МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Навчальний посібник
з дисципліни
"Програмне забезпечення мультимедіа"

Тернопіль - 2011

Р.П. Шевчук // Навчальний посібник з дисципліни „Програмне забезпечення мультимедіа”, для студентів напрямку „Програмна інженерія”. — Тернопіль, 2011. — 174 с.

Укладач: Шевчук Руслан Петрович, к.т.н., доцент кафедри комп’ютерних наук ТНЕУ

Відповідальний за випуск: Дивак Микола Петрович, д.т.н., професор., завідувач кафедри комп’ютерних наук ТНЕУ

Рецензенти:

Завідувач кафедри комп’ютерних
систем та мереж Тернопільського
національного технічного університету
імені Івана Пулюя
доктор технічних наук, професор С.А. Лупенко

Доцент кафедри спеціалізованих
комп’ютерних систем Тернопільського
національного економічного університету
кандидат технічних наук, доцент В.В. Яцків

Затверджено на засіданні кафедри комп’ютерних наук ТНЕУ.
Протокол № 17 від «14» червня 2011 р.

ЗМІСТ

РОЗДІЛ 1. ВСТУП В МУЛЬТИМЕДІА ТЕХНОЛОГІЇ

1.1. Вступ в мультимедіа

Термін мультимедіа – латинського походження, що поширився за рахунок англомовних джерел (“*multi*” - множинний, складний та “*media*” - середовище, засіб, спосіб). У буквальному перекладі термін “мультимедіа” означає “множинний засіб” або “багато середовищ”. Як одне слово термін “мультимедіа” почали використовувати у 60-ті роки XX сторіччя.

У 1980 році було розроблено перший мультимедійний програмний продукт – комп’ютерний гороскоп.

У 1988 році Європейською Комісією, що займається проблемами впровадження і використання нових технологій було офіційно затверджено назву нової технології - мультимедіа. Ідейним прототипом виникнення технології мультимедіа вважають “Концепцію організації пам’яті MEMEX”, запропоновану в 1945 році американським вченим Ваннівером Бушем. Концепція передбачає пошук інформації за змістовними характеристиками, а не за формальними принципами (порядковим номером, індексом, алфавітом і т.п.).

Стрімкий розвиток мультимедіа розпочався у 80-тих роках XX століття. Білл Гейтс був одним з перших, хто реалізував на практиці ідею мультимедійного комерційного продукту, який акумулював три принципи мультимедіа:

- подання даних за допомогою динамічної комбінації множин середовищ;
- наявність кількох сюжетних ліній у змісті програмного продукту;
- розвинений художній дизайн програмного інтерфейсу і засобів навігації.

З розвитком комп’ютерних технологій, програмісти стали використовувати термін “мультимедіа” для визначення комп’ютерних програм та продуктів, які містять в собі аудіо, зображення, відео, текст, анімацію та гіпертекст.

У сучасній науковій та технічній літературі зустрічаються різні визначення терміну “мультимедіа” залежно від того, де і для кого передбачається його використання [1]. Так наприклад, Машбиць Ю.І трактує “мультимедіа” як багатоканальне середовище, що видає інформацію в різноманітних модальностях [2]. У Шликової О.В. мультимедіа – “полісередовище”, єдиний простір, який в синкретичному вигляді представляє різні види та способи надання інформації (текст, графіку, звук, тощо) [3]. Деякі автори тлумачать мультимедіа як сучасну інформаційну технологію, що об’єднує за допомогою комп’ютерних засобів графічне та відео зображення, звук і інші спеціальні ефекти [4].

Узагальнивши найбільш часті визначення терміну “мультимедіа”, під цим терміном будемо розуміти сукупність технологій інтерактивного

оброблення даних, що представляються у формі аудіо, зображення, відео, тексту, анімації та гіпертексту.

Технології мультимедіа визначають сукупність методів та програмно-апаратних засобів, інтегрованих з метою оброблення, зберігання та відтворення мультимедійних даних, що використовують зоровий, слуховий та тактильний канал надходження інформації до людини.

Сьогодні виділяють три типи мультимедіа технологій:

1. Автономні – технології оброблення, зберігання та відтворення мультимедійних даних, що не прив'язані до конкретних програмно-апаратних засобів. Дані сформовані завдяки цим технологіям зберігається на відеокасетах, комп'ютерних дисках, флеш картах, зовнішніх вінчестерах та інших засобах зберігання інформації. Для зчитування даних необхідне певне обладнання: відеомагнітофони, програвачі компакт дисків, персональні комп'ютери.
2. Телемовні – технології, що використовують спеціалізоване обладнання (кабельні системи, супутники) для роботи із мультимедійними даними. До технологій цього типу відносять: супутникове телебачення, цифрове телебачення, телетекст, стільниковий зв'язок.
3. Телекомунікаційні – технології оброблення, зберігання та відтворення мультимедійних даних в реальному часі, що використовують сучасне телекомунікаційне обладнання. До технологій цього типу належать: комп'ютерна телефонія, IP-телефонія, мультимедіа-конференції, call-центри, онлайн-служби та інші. У всіх цих технологіях регламентована передача мультимедійних даних за допомогою цифрового сигналу.

Розрізняють лінійний та нелінійний спосіб відтворення мультимедіа даних. При лінійному способі, процес відтворення даних відбувається без участі людини (наприклад при перегляді відео, людина жодним чином не може вплинути на його зміст). Нелінійний спосіб (гіпермедіа) дає змогу людині брати активну участь в процесі відтворенні мультимедійних даних. Участь людини в даному процесі називається інтерактивністю. Такий спосіб взаємодії людини та комп'ютера найбільш повно представлений у комп'ютерних іграх.

Як приклад лінійного та нелінійного способу відтворення даних, можна розглядати мультимедійну презентацію. Якщо презентація в режимі слайд-шоу, показується аудиторії, то цей спосіб подання даних називається лінійним, тому що відсутні можливості впливу аудиторії на процес відтворення. У випадку "живої" презентації, аудиторія має змогу ставити доповідачеві запитання та взаємодіяти з ним в інший спосіб. Це дозволяє доповідачеві відходити від теми презентації, щоб більш детально зупинитись на цікавих частинах доповіді. Таким чином, "жива презентація" представляє нелінійний (інтерактивний) спосіб відтворення даних.

Технології мультимедіа представляють сенсорну інформацію в максимально близькій для людини формі через програмне та апаратне

забезпечення персонального комп'ютера, а також автономні мультимедійні пристрої.

На рисунку 1.1 наведено графічне відображення загальної структури мультимедіа.

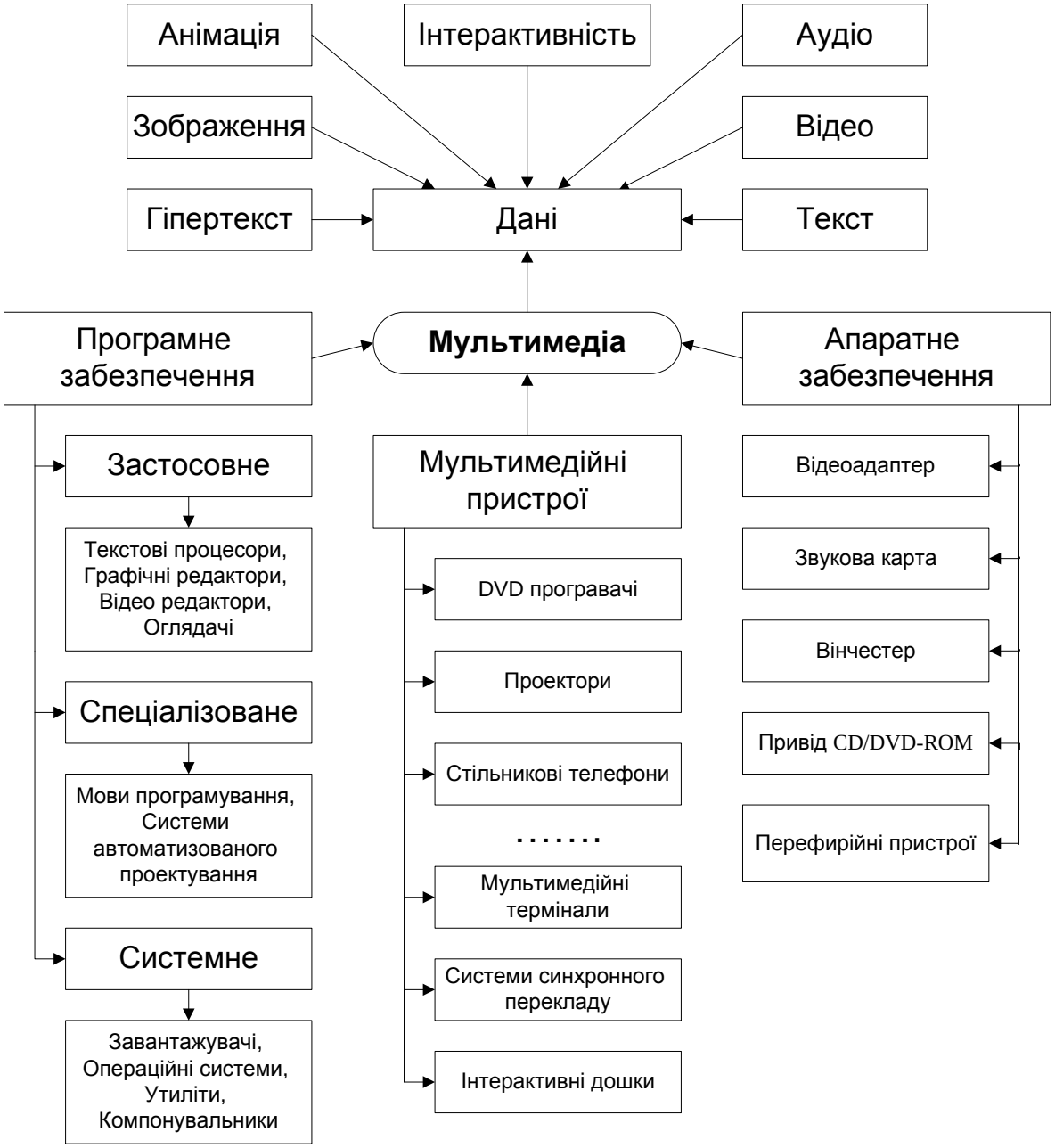

Рисунок 1.1 – Загальна структура мультимедіа

Як видно з рисунку 1.1, мультимедіа визначає комплекс програмних, апаратних засобів та мультимедійних пристроїв, які дають змогу користувачеві працювати в інтерактивному режимі з різнотипними даними, що організовані у вигляді єдиного інформаційного середовища.

Апаратне забезпечення мультимедіа надає безпосередній доступ до даних завдяки стандартним пристроям персонального комп'ютера: відеоадаптерам, моніторам, дисководам жорстких дисків, приводам CD/DVD-ROM, звуковим картам та перефирійним пристроям.

Програмне забезпечення мультимедіа (ПЗМ) визначає набір кодових інструкцій для оброблення мультимедійних даних центральним процесором. ПЗМ поділяється на:

- застосовне (прикладне) забезпечує розв`язання певної користувацької задачі та потребує безпосередньої взаємодії між користувачем та комп'ютером. Прикладами застосунків є текстові процесори, графічні редактори, відео редактори, оглядачі;

- спеціалізоване забезпечує створення мультимедійних додатків. Прикладами спеціалізованого ПЗМ є мови програмування, системи автоматизованого проектування;

- системне забезпечує інфраструктуру, на якій можуть працювати прикладні програми, тобто воно керує і контролює апаратне забезпечення. Прикладами системного ПЗМ є завантажувачі, операційні системи, драйвери, утиліти, компонувальники.

Мультимедійні пристрої – обладнання, призначене для роботи з мультимедіа даними. Прикладами такого обладнання є: мультимедійні термінали, DVD-програвачі, стільникові телефони, проектори, системи синхронного перекладу, інтерактивні дошки та інші.

Основними складовими мультимедіа є: текст, графіка, анімація, аудіо, відео, гіпертекст та інтерактивність. Кожна з цих складових має свої особливості і правила використання.

1.2. Історія розвитку мультимедіа технологій

Рисунок 1.2 – Малюнок на стіні печери

Більшість дослідників бачать витоки сучасних мультимедіа технологій у верхньому палеоліті, коли первісні люди почали використовувати стіни печер для малюнків вдалого полювання (рисунок 1.2). Вже тоді людина знала примітивні технології створення та зберігання графічної інформації. Використовуючи різні природні фарбники первісна людина створювала на стінах печер справжні художні твори, фарби яких зберігають свій колір багато тисячоліть. Безумовно, свої малюнки кроманьйонець створював в ритуальних цілях, сподіваючись на успіх в найближчому полюванні. Але, разом з тим, малюнки були першою технологією передачі і зберігання знань.

Наступний етап розвитку мультимедіа технологій - клинопис стародавніх шумерів (рисунок 1.3). Більше 3000 років до н.е. в межиріччі Тігра і Евфрата

існувала розвинена землеробська цивілізація, яка вимагала створення складних іригаційних систем і каналів. Для зберігання і передачі інформації шумери створили свою писемність - клинопис, щось середнє між малюнками і буквами. Писемність була дуже складною, вивчити і користуватися нею могли тільки представники привілейованих класів. Цивілізація шумерів була однією з найрозвиненіших в ті далекі тисячоліття.

Рисунок 1.3 – Клинопис шумерів

Приблизно в 1750-1670 рр. до н.е. у Стародавній Греції з'являється лінійна писемність, яка близька до нашого розуміння. Трохи пізніше, в Китаї активно розвинулась писемність за допомогою ієрогліфів, у якій практично кожне слово та вираз мали свій знак.

Алфавіт, тобто писемність за допомогою певної кількості знаків, кожен з яких позначав звук, вперше почали використовувати фінікійці близько 1100 років до н.е.

Рисунок 1.4 – Перший папір

Наступним етапом розвитку мультимедіа технологій стала поява паперу (рисунок 1.4). Папірус і пергамент були дорогими, в порівнянні з камінням чи глиною, але вони дозволяли записувати інформацію без особливих зусиль. Пергамент, до того ж, дозволяв стирати попередні записи і використовувався повторно. Проте вже в 105 році нашої ери в Китаї з'явився папір, а через 625 років і перша друкарня. У 853 році в Китаї з'явилася перша книга, надрукована друкарським способом. Це був великий крок вперед у розвитку медіа-технологій.

У 1450 році в Німеччині друкарським способом було надруковано перші книги. У 1500 році в Європі налічувалось близько 20 млн. книг найрізноманітнішого змісту. А в 1620 році у Амстердамі з'явилося перше регулярне щотижневе видання.

Наступний етап розвитку технологій мультимедіа почався через декілька сторіч. У 1822 році Нієпс провів перші досліди із фотографуванням зображенням. Через 19 років англієць Ф. Телбот удосконалив технологію Нієпса, запропонувавши проектувати негативне зображення на поверхню, покриту солями срібла. Фотографія стала досить популярною серед артистів та вчених.

1 вересня 1794 року були передані перші повідомлення з Ліля до Парижа по оптичному телеграфу, а в 1837 році з'являється електричний телеграф і

відома азбука Морзе. Досить швидко за допомогою телеграфу навчилися передавати не тільки слова, але і прості малюнки.

4 лютого 1876 року в Бостоні близько 14 годин Грейхем Белл, викладач притулку для глухонімих, подав патент, який описував апарат, здатний передавати звук на відстань. Белл активно використовував свій винахід і вже у 1881 році в США було 400 телефонних станцій і 132 000 абонентів. Майже одночасно з телефоном була розроблена і технологія механічного звукозапису. У 1877 році Едісон в США і Шарль Грос у Франції зробили звуковідтворюючий апарат — фонограф. На рухомий циліндр, покритий тонкою фольгою, була записана популярна американська пісня.

Ввечері 28 грудня 1895 у Парижі було показано перші “рухливі зображення” (рисунок 1.5). Цього вечора брати Люмьєр відтворили на екрані перші кіносюжети - “Робочі виходять з воріт заводу”, “Купання дитини”. Кіно розвивалося дуже швидко. У США між 1905 і 1909 роками було побудовано більше 10000 кінотеатрів. Можна сказати, що кіно стало прообразом мультимедіа, особливо після 1927 року, коли його вперше озвучили.

У 1906 році молодий американець Форест отримав патент на пристрій, який дозволяв передавати комплексний сигнал. І 2 квітня 1908 року вперше в радіоефірі зазвучав людський голос: “Алло, алло, з Вами говорить радіостанція з Ейфелевої башти в Парижі”.

Оскільки радіомовлення і кіно розвивалися паралельно, то рано чи пізно мало відбутись об'єднання цих технологій в одному пристрої, здатному передавати зображення на відстань. Ідеї, що стосувались телебачення, висловлювалися багатьма вченими ще в середині XIX століття. Саме слово “телебачення” з'явилося в 1900 році на Всесвітній виставці в Парижі. Але тільки в 1925 році з'явилися більш-менш повні системи передачі і прийому телевізійного зображення.

В результаті розвитку технологій друку, кіно, телебачення і радіомовлення знання, інформація і культура стали доступні найширшим верствам населення.

Однак докорінні зміни в розвитку мультимедіа технологій внесла поява комп'ютерів, після чого почалось лавиноподібне зростання кількості мультимедіа технологій. Кожного року з’являлись десятки нових та удосконалених технологій мультимедіа.

У 1948 році введений в дію перший в світі комп'ютер з програмою, що зберігається - "Манчестерський Марк-1", створений англійськими вченими Томом Кілбурном і Фредді Вільямсом з Манчестерського університету.

Невдовзі з'явився перший накопичувач на магнітній стрічці, пристрій IBM 726. Щільність запису становила 100 символів на дюйм, швидкість 75 дюймів за секунду.

Фірмою IBM були розроблені плаваючі магнітні головки на повітряній подушці. Винахід дозволив створити новий тип пам'яті - дискові

запам'ятовуючі пристрої. Це - перший жорсткий диск, що коштував більше за мільйон доларів.

Перші запам’ятовуючі пристрої на дисках з'явилися в машинах IBM 305 і RAMAC-650.

У лабораторії Bell Labs створили пристрій для передачі даних по телефонних лініях.

В 1965 році термін “мультимедіа” вперше було використано для опису проекту «Вибухове Пластикове Неминуче» (Exploding Plastic Inevitable) — події, яка поєднувала концерт рок музики, фільм, експериментальне освітлення та виставу. Серію цих вистав організував Енді Ворхол, засновник художньої школи поп-арту.

У грудні 1968 року Полом Сеффо організовано першу мультимедіа конференцію. На цій конференції відеопотік, що направлявся по радіоканалу з Альто, висвітлював основні моменти роботи Девіда Енгельбарта в Стенфордському дослідницькому університеті. Було показано основні технологій нової інформаційної ери: інтерактивне програмування, спільне використання баз даних, відеоконференції, навігація у віртуальних просторах.

У 1976 році молоді американці Стів Джобс і Стів Возняк організували підприємство по виготовленню персональних комп'ютерів "Apple", призначених для великого кола непрофесійних користувачів. У 1977 році запущено в масове виробництво персональний комп'ютер PET.Apple-2, що містив процесор 6502, операційну систему, мову програмування Basic, оперативну пам'ять, два ігрових електронних пульта, інтерфейс для приєднання до касетного магнітофона і систему кольорової графіки.

У 1986 році широкого розповсюдження набули комп’ютери ATARI ST, які досить довго займали нішу в області бюджетних рішень для оброблення музики.

У березні 1989 р. Чи Тім Бернерс з CERN запропонував керівництву міжнародного європейського наукового центру концепцію нової розподільної інформаційної системи, яку назвав World Wide Web. Гіпертекстова технологія повинна була дозволити легко переходити з одного документа в інший. У 1990 році ці пропозиції були прийняті, і проект стартував.

На початку 90-х років широко почали використовуватись гібридні мультитрекові системи, що підтримували MIDI формат та могли створювати аудіо записи.

Бурхливий розвиток мережі Інтернет дозволив створити ряд технологій, які до сьогодні не втрачають своєї популярності. Зокрема, у грудні 1992 року у Великобританії запущена технологія SMS, що дозволила передавати текст з персонального комп'ютера на мобільний телефон в мережі GSM компанії Vodafone. У 1993 році вперше була реалізована можливість передачі голосових повідомлень через мережу з пакетною комутацією. Дана технологія отримала назву VoIP (Voice over IP).

У 1994 році вироблено основні положення стандарту DVB-C (цифрового кабельного телебачення) та DVB-S (цифрового супутникового телебачення). Роботу над стандартом цифрового наземного (ефірного) телебачення DVB-T було завершено в 1996 році.

У 1996 році для платформи Apple Macintosh реалізовано нову революційну технологію Virtual Studio Technology (VST), яка дозволила використовувати 24 аудіо-трека та необмежену кількість MIDI-треків. Завдяки VST стала доступна обробка аудіо даних в режимі реального часу.

У березні 1996 року фірма Intel вперше представила інформацію про технологію MMX (Matrix Math Extensions [instruction set] - набір команд для розширення матричних математичних операцій). Дана технологія корпорації Intel реалізована у 1997 році в процесорах Pentium для підтримки мультимедіа .

У 1998 році групою компаній: Ericsson, IBM, Intel, Nokia та Toshiba створена технологія бездротового зв'язку Bluetooth, що дала змогу передавати мультимедіа дані зі швидкістю 64 Кбіт/сек.

В кінці 1998 року затверджено стандарт MPEG-4, який забезпечував передачу відео та звуку по вузькосмугових каналах передачі даних.

Інформатизація суспільства і бурхливий розвиток мультимедійних технологій, що спостерігається в XXI столітті все більше впливає на процеси у всіх галузях роботи людини. Практично кожен день у світі з’являються нові та удосконалюються перспективні технології роботи з мультимедійними даними.

Сьогодні створюються технології, які дозволять повністю погрузити людину у віртуальний світ. Вирішуються питання механізмів смаку, запаху та тактильних дій, які дозволять мозку людини в реальному часі взаємодіяти з мультимедіа даними.

1.3. Складові мультимедіа

У даному підрозділі розкриваються характеристики та правила використання основних складових мультимедіа.

На рисунку 1.5 представлено класифікацію мультимедіа складових.
Як видно з рисунку 1.5, мультимедіа складові поділяються на:

- зображення: векторні, растрові;
- аудіо: звукові сигнали, мовні сигнали;
- відео: аналогове, цифрове;
- анімація: графічна, об’ємна, комп’ютерна;
- текст, основу якого формують шрифти;
- гіпертекст;
- інтерактивність.

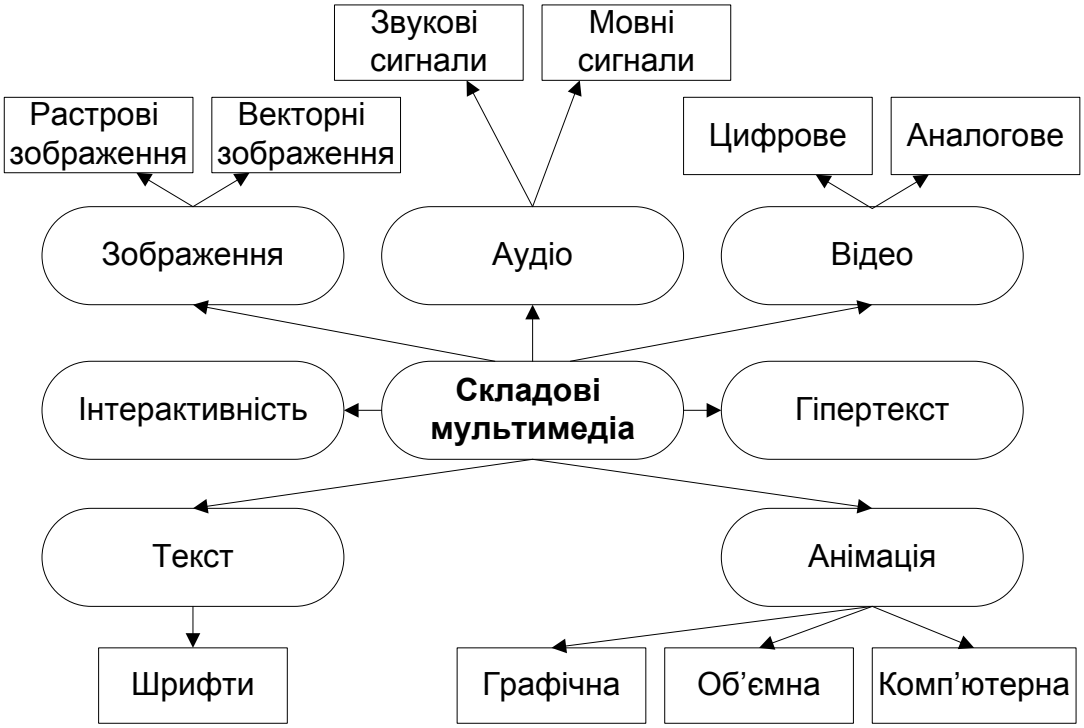

Рисунок 1.5 – Класифікація мультимедіа складових

1.3.1. Зображення

Зображення - відтворення виду, форми і кольору предмета. Зображення створюється, обробляється і відображається за допомогою комп'ютерної графіки, яка використовує апаратні і програмні засоби обчислювальної техніки. Для відображення комп'ютерного зображення використовують монітор, принтер, плотер тощо.

Розрізняють два способи створення зображень - растровий і векторний і, відповідно, два види комп'ютерної графіки - растрову і векторну.

Растрова графіка представляється в машинній пам'яті у вигляді растру. Растрове зображення будується як набір елементарних точок (пікселів) розфарбованих тим чи іншим кольором. Кожен піксель має координати. В найпростішому випадку, коли використовується чорно-біле зображення для опису кольору достатньо одного двійкового розряду: 0 – чорний; 1 – білий. Для 256-колірного зображення необхідно вісім розрядів на кожен піксель ($256=2^8$). Найскладніші, фотореалістичні кольорові зображення потребують до 24 розрядів для представлення пікселя. У зв'язку з цим розмір файлів з растровими зображеннями досить швидко збільшується при великій глибині кольору. Також якість растрового зображення суттєво залежить від розміру пікселя, який в свою чергу, визначає роздільну здатність монітора. Растрове зображення характеризується наступними параметрами:

- розміром зображення (вимірюється в пікселях, міліметрах, дюймах і т. д.);
- розширенням - кількість точок на одиницю (дюйм);

- кількістю переданих кольорів або глибиною кольору;
- форматом зберігання - спосіб зберігання графічної інформації.

Оброблення растрових зображень виконується растровими графічними редакторами.

Векторна графіка створює зображення за правилами векторної алгебри з точок, ліній, поверхонь, тобто об’єктів які можна описати математичним рівнянням. Наприклад, для опису будь-якого кола необхідно всього три числових значення: радіус, координати центру і товщина лінії. Завдяки цьому, векторне зображення має ряд переваг в порівнянні з растровим:

- зміна масштабу без втрати якості і практично без збільшення розмірів вихідного файлу;
- велика точність (до сотої частки мікрона);
- невеликий розмір файлу;
- висока якість друку;
- відсутність проблем з експортом векторного зображення в растрове;
- можливість редагування кожного елемента зображення окремо.

Типовими примітивними об’єктами векторної графіки є: лінії, багатокутники, окружності, еліпси, криві Безьє, текст та інші.

Наведемо приклад, що ілюструє як може бути описаний відрізок прямої у різних форматах:

- у векторному форматі задаються координати початку і кінця, колір та товщина лінії;
- у растровому форматі задаються координати та колір кожної точки, що входить у цей відрізок.

У комп'ютерних системах використовуються такі типи зображень:

- Двоколірні (бінарні) - 1 біт на піксель. Найчастіше зустрічаються чорно-білі зображення.

- Півтонові - градації сірого або іншого кольору. Наприклад, 256 градацій (1 байт на піксель).

- Кольорові зображення. Використовуються від 2 біт на піксель і вище. Глибина кольору 16 біт на піксель (65536 кольорів) отримала назву High Color, для 24 біт на піксель (16,7 млн. кольорів) - True Color. В комп'ютерних графічних системах використовують і більшу глибину кольору - 32, 48 біт на піксель і вище.

1.3.2. Аудіо

Аудіо (лат. audio “чую”) - загальний термін, що стосується звукових технологій. Найчастіше під терміном аудіо розуміють звук, записаний на носії інформації. Складовими аудіо є звукові та мовні сигнали.

Звуковий сигнал - коливальний рух частинок пружного середовища, що поширюється у вигляді хвиль у газоподібному, рідкому чи твердому

середовищі. Людина чує звуковий сигнал з частотами від 16 Гц до 20 кГц. Звуки з частотами до 16 Гц називаються інфразвуком, вище 20000 Гц - ультразвуком. Наука, що вивчає звукові сигнали, називається акустикою.

Характеристиками звукового сигналу є:

- частота - кількість коливань певної точки звукової хвилі в секунду. Одному циклу коливання в секунду відповідає величина 1 Гц (1/с);
- довжина хвилі – характеристика плоскої періодичної хвилі, що позначає найменшу відстань між точками простору, в яких хвиля має однакову фазу;
- амплітуда – найбільше значення величини звукової хвилі, яка періодично змінюється;
- швидкість – відношення переміщення звукового сигналу до проміжку часу, за яке це переміщення відбувалось. Швидкість звуку залежить від середовища, через яке проходять звукові хвилі і визначається його параметрами. Швидкість звуку в повітрі при нормальних умовах становить 330 м/с, вона може змінюватися в залежності від температури. В твердих тілах та рідинах звуковий сигнал розповсюджується швидше;
- висота – суб'єктивна оцінка якості звукового сигналу. Висота звуку визначається частотою коливань основного тону (незалежно від амплітуди коливань складових) і фіксується як нота. Чим більша частота коливань, тим вищий звук;
- гучність – інтенсивність звукового відчуття, що викликана звуковим сигналом у людини з нормальним слухом. Гучність залежить від сили звуку і виражається кількістю децибел, на які даний звук перевищує за інтенсивністю звук, прийнятий за поріг чутності.

На рисунку 1.6 наведено класифікацію звукових сигналів.

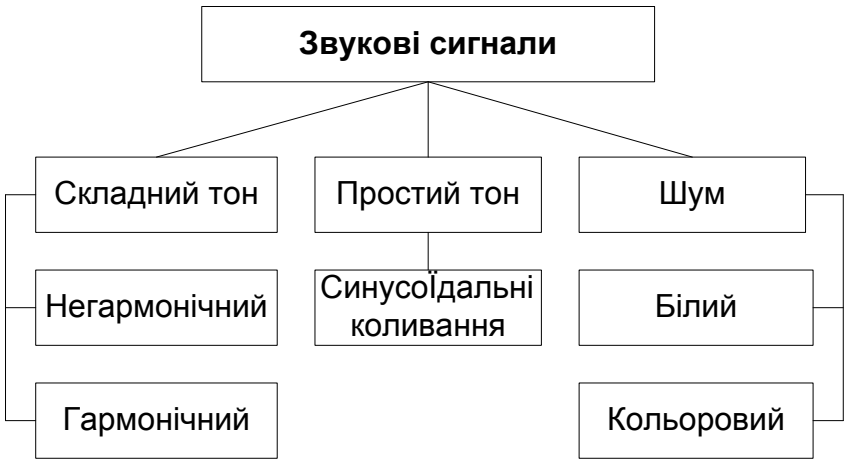

Рисунок 1.6 – Класифікація звукових сигналів

На рисунку 1.6 звукові сигнали класифіковано за об'єктивними характеристиками акустичної хвилі:

- простий тон (синусоїдні коливання);
- складний тон:

- гармонічний (визначеної звуковисотності, що складається з основного тону та обертонів);

- негармонічний(приблизно визначеної звуковисотності, що складається з основного тону та негармонічних обертонів);

- шум:

- білий шум (хаотичні коливання, спектральні складові розміщуються рівномірно по всьому діапазону);

- кольоровий шум (хаотичні коливання, спектральні складові розміщуються нерівномірно по всьому діапазону, як правило з поступовим зменшенням інтенсивності від низьких до високих частот).

Простий звуковий сигнал представляється зазвичай напругою або струмом, що змінюється в часі по синусоїдальному закону. Амплітуда відповідає гучності звуку, частота - висоті звуку. Для представлення звукового сигналу в цифровому вигляді аналоговий сигнал перекодовують, запам'ятовуючи параметри сигналу через певні проміжки часу в структурі даних певного розміру.

Якість запису звукового сигналу характеризується: частотою дискретизації, розміром структури даних, кількістю каналів (стерео, моно, квадро).

Для відтворення цифрового звукового сигналу використовують зворотне перетворення цифрового сигналу в аналоговий або синтез аналогового сигналу на основі цифрового запису.

Мовний сигнал – хвиля, яка формується голосовим апаратом людини. Наука, що вивчає мовні сигнали називається фонетикою.

Для формування хвилі певної якості, сигнал проходить шлях від джерела в мовному тракті та порушує дію партикулярних органів, які працюють як фільтри, що змінюються в часі. Артикулярні органи накладають обмеження на швидкість зміни сигналу. Вони використовують функцію згладжування для гладкого зчеплення окремих базових фонетичних одиниць у складний мовний потік.

На рисунку 1.7 представлено схему голосового апарату людини, який формує мовний сигнал.

В силу своєї специфіки мовні сигнали розглядаються з трьох точок зору:

- акустичної, оскільки мовний сигнал є акустичним явищем;

- фізіологічної, оскільки мовний сигнал є продуктом діяльності нервової системи та формується голосовим апаратом людини;

- лінгвістичної, оскільки за допомогою мовних сигналів відбувається спілкування.

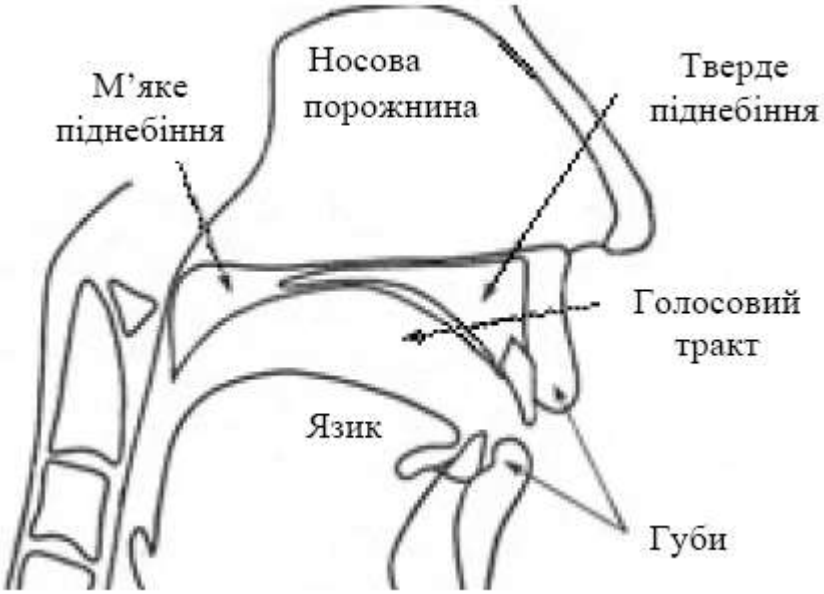

Рисунок 1.7 – Схема голосового апарату людини

З точки зору акустики, що є найбільш близькою до мультимедіа, мовний сигнал представляє собою коливання пружного середовища та має відповідний спектр, інтенсивність і діапазон. Математичну модель мовного сигналу можна представити у вигляді генераторів тонового і білого шуму та групи фільтрів, модуляторів і ключів (рот, ніс, мова, губи), що забезпечують фільтрацію і формування відчуття звуку.

Мовний апарат людини при генерації сигналу використовує фізичні принципи для отримання різних типів звуків: голосний, шиплячий голосний, змішані шиплячі-тонові звуки, вибуховий голосний, пауза, зміна параметрів артикуляції, інтонація.

Найбільш відомою характеристикою мовного сигналу є основний тон – найнижчий тон, що створює джерело сигналу. Ця характеристика є звичайною частотною модуляцією сигналу, параметри якої легко вимірюються.

У комп'ютерних системах, в більшості випадків, цифрові мовні сигнали стискаються.

1.3.3. Відео

Відео (від лат. video - дивлюся, бачу) - під цим терміном розуміють широкий спектр технологій запису, оброблення, передачі, зберігання й відтворення візуального і аудіовізуального матеріалу. У побутовому значенні відео означає відеоматеріал, телесигнал або кінофільм, записаний на фізичному носії (відеокасеті, відеодиску, компакт диску й т. п.).

Відео поділяють на аналогове та цифрове. Аналогове відео визначається відеосигналом, в якому вихідний параметр змінюється як неперервна функція вхідного сигналу, а його значення знаходиться будь де у визначених межах.

Цифрове відео визначається цифровим відеосигналом, що може бути створений комп’ютерним способом або шляхом перетворення з аналогового відеосигналу.

Відеосигнал характеризується наступними параметрами: кількість кадрів в секунду, розгортка, роздільна здатність, співвідношення сторін екрана, кількість кольорів, кольорова розрядність, бітова швидкість (ширина відеопотоку).

Кількість кадрів на секунду - це число нерухомих зображень, що змінюють одне одне впродовж однієї секунди відеоматеріалу, створюючи ефект руху об'єктів на екрані. Чим більша частота кадрів на секунду, тим плавнішим і природнішим буде здаватися рух. Мінімальний показник, за якого рух буде сприйматися однорідним - приблизно 10 кадрів на секунду (це значення індивідуальне для кожної людини). У традиційному плівковому кінематографі використовується частота 24 кадри на секунду. Системи телебачення PAL й SECAM використовують 25 кадрів на секунду, а система NTSC використовує 29,97 кадри на секунду. Комп'ютерні оцифровані відеоматеріали гарної якості, як правило, використовують частоту 30 кадрів на секунду. Верхня гранична частота мерехтіння, що сприймається людським мозком, в середньому становить 39-42 Герца й індивідуальна для кожної людини. Деякі сучасні професійні камери можуть знімати з частотою до 120 кадрів на секунду.

Розгортка - процес перетворення оптичного зображення в послідовність сигналів. Розгортка відеоматеріалу може бути прогресивною або черезрядковою. При прогресивній розгортці всі горизонтальні лінії зображення (рядки) відображаються одночасно. При черезрядковій розгортці показуються почергово парні й непарні рядки. Черезрядкову розгортку було розроблено для відтворення зображення на кінескопах, а зараз її використовують для передачі відео по “вузьким” каналах. PAL, SECAM та NTSC - це системи із черезрядковою розгорткою. Нові цифрові стандарти телебачення, наприклад, HDTV використовують прогресивну розгортку.

Роздільна здатність - фізична кількість колонок та рядків пікселів, що створює дисплей. Звичайний аналогова телевізійна роздільна здатність складає 720×576 пікселів для стандартів PAL і SECAM, при частоті кадрів 50 Герц (одне поле, 2×25); і 648×486 пікселів для NTSC, при частоті 60 Герц (одне поле, 2×29,97). В позначенні 648×480 першим числом позначається кількість точок у горизонтальній лінії (горизонтальне розділення), а другим числом кількість самих ліній (вертикальне розділення). Новий стандарт високоякісного цифрового телебачення HDTV передбачає роздільну здатність до 1920×1080 при частоті 60 Герц з прогресивною розгорткою.

Співвідношення ширини й висоти кадру - найважливіший параметр у будь-якому відео. З 1910 року кінофільми мали співвідношення сторін екрана 4:3 (4 одиниці ширини та 3 одиниці висоти; іноді це записується як 1,33:1 або

1,33). Телебачення успадкувало це співвідношення і майже всі аналогові телесистеми мали співвідношення сторін екрана 4:3. Комп'ютерні монітори також успадкували телевізійний стандарт сторін. У 1950-х роках уява про 4:3 суттєво змінилася. Справа в тому, що поле зору людини має співвідношення аж ніяк не 4:3. Оскільки у людини 2 ока, розташовані на одній горизонтальній лінії, то поле зору людини наближається до співвідношення 2:1. Щоб наблизити форму кадру до природного поля зору людини, було запропоновано стандарт 16:9 (1,78). До кінця XX століття, після ряду додаткових досліджень у цій області, почали з'являтися більш радикальні співвідношення сторін кадру: 1,85, 2,20 і аж до 2,35. На рисунку 1.8 представлено список стандартних роздільних здатностей та співвідношень сторін екрана різних форматів.

Кількість кольорів і кольорова розрядність описується кольоровими моделями. Для стандарту PAL застосовується колірна модель YUV, для SECAM модель YDbDr, для NTSC модель YIQ, у комп'ютерній техніці застосовується в основному RGB (і αRGB), рідше HSV, а в друкарській техніці CMYK. Кількість кольорів, що може показати монітор або проектор залежить від якості монітора або проектора. Людське око може сприйняти, по різних підрахунках, від 5 до 10 мільйонів відтінків кольорів. Кількість кольорів у відеоматеріалі визначається числом бітів, відведеним для кодування кольору кожного пікселя. 1 біт дозволяє закодувати 2 кольори (як правило, чорний і білий), 2 біти - 4 кольори, 3 біти - 8 кольорів, …, 8 бітів - 256 кольорів(28=256), 16 бітів - 65 536 кольорів (216), 24 біти -16 777 216 кольорів (224).

Ширина (або швидкість) відеопотоку - кількість оброблюваних бітів відеоінформації за секунду часу. Чим вища ширина відеопотоку, тим краща якість відео. Наприклад, для формату VideoCD ширина відеопотоку складає приблизно 1 Мбіт/с, а для DVD складає приблизно 5 Мбіт/с. Формат цифрового телебачення HDTV використовує ширину відеопотоку біля 10 Мбіт/с. За допомогою швидкості відеопотоку дуже зручно оцінювати якість відео при його передачі через Інтернет. Розрізняють два види керування шириною потоку у відеокодеку - постійний бітрейт (англ. constant bit rate, CBR) і змінний бітрейт (англ. variable bit rate, VBR).

Якість відео вимірюється за допомогою формальних метрик, таких, як PSNR або SSI, або з використанням суб'єктивного порівняння із залученням експертів.

Суб'єктивна якість відео виміряється за наступною методикою:

- вибираються відеопослідовності для використання в тесті;
- вибираються параметри системи вимірювання;
- вибирається метод показу відео й підрахунку результатів виміру;
- запрошується необхідне число експертів (звичайно не менше 15);
- проводиться тест;
- підраховується середня оцінка на основі оцінок експертів.

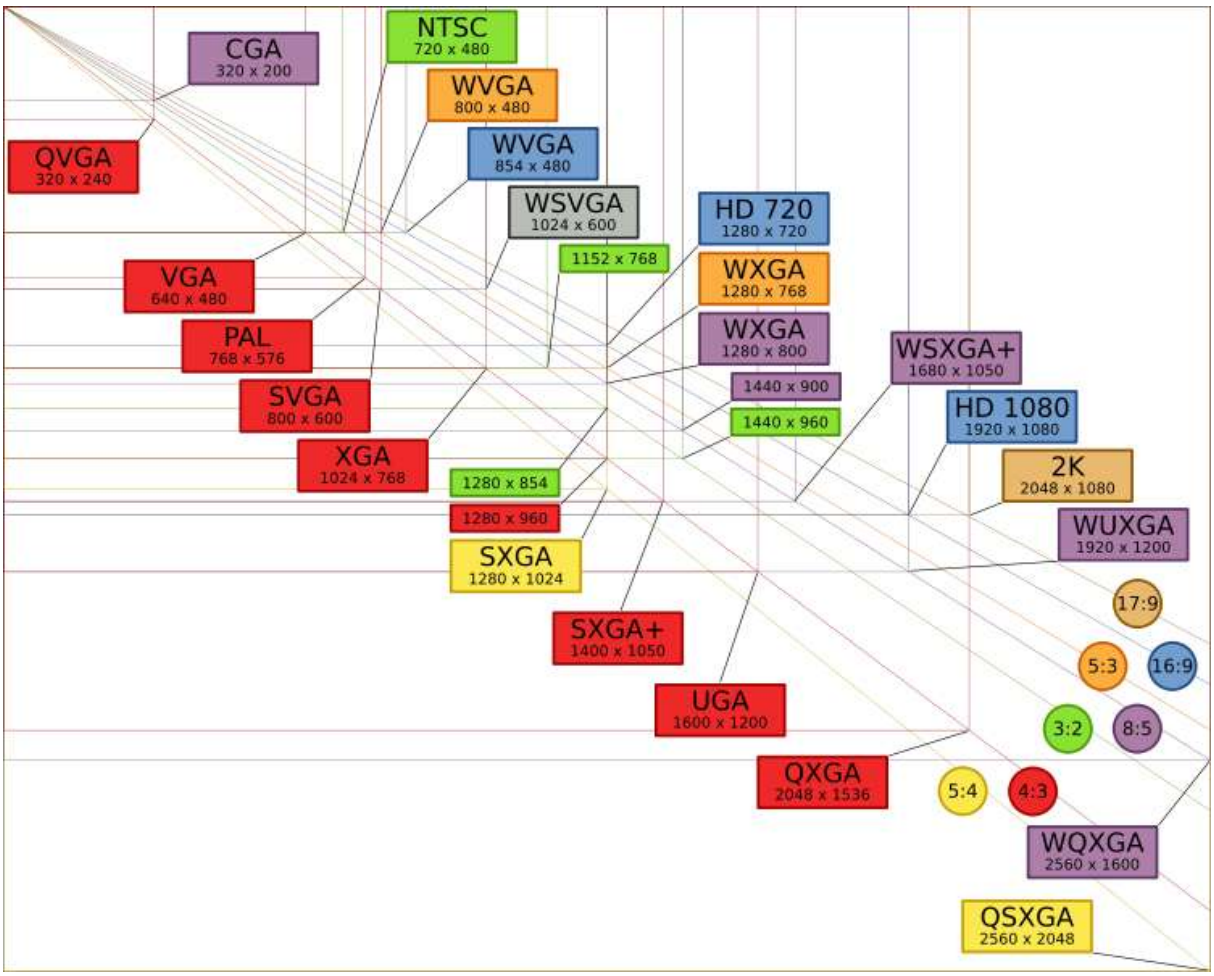

Рисунок 1.8 – Роздільні здатності відео форматів

Кілька методів суб'єктивної оцінки описані в рекомендаціях ITU-T BT.500. Один із найбільш популярних методів оцінки - DSIS (Double Stimulus Impairment Scale), при якому експертам спочатку показують вихідний відеоматеріал, а потім оброблений. При цьому експерти оцінюють якість обробки, варіюючи свої оцінки від «оброблення непомітне» і «оброблення покращує відеозображення» до «оброблений відеоматеріал сильно дратує».

1.3.4. Гіпертекст

Гіпертекст – інтерактивний нелінійний документ, окремі вузли якого зв’язані між собою за допомогою гіперзв'язків чи гіперпосилань. Нелінійність гіпертексту полягає в тому, що після прочитання кожного із фрагментів тексту, з’являється множинність можливостей вибору подальшого прочитання інших елементів тексту. Гіпертекст забезпечує функціонування великих обсягів текстової, графічної та інших видів інформації у віртуальному просторі.

Інформативність гіпертексту пов’язана з використанням системи посилань. Зокрема гіпертекст характеризується специфічними параметрами, які залежать від особливостей побудови системи посилань у гіпертексті:

- інтрагіпертекстовість, коли посилання з одного вузла на інший зв’язуються з межах одного тексту;

- інтергіпертекстовість, коли посилання з одного вузла на інший зв’язуються в межах окремих текстів;

- екстрагіпертекстовість, коли з посилання одного вузла на інший зв’язуються між окремими гіпертекстами з межах одного тексту.

Найпопулярнішим прикладом гіпертексту є World Wide Web, у якому веб-оглядач переміщує користувача з одного документу в інший. Для створення гіпертекстових документів у Інтернеті розроблено спеціальну мову розмітки гіпертексту HTML.

1.3.5. Анімація

Анімація - процес створення серії знімків, малюнків, кольорових плям, ляльок або силуетів в окремих фазах руху, за допомогою якого під час показу їх на екрані виникає враження оживлення форм.

Відомо такі види анімації:

- графічна – класичний вид анімації, де об’єкти малюються вручну;

- об’ємна – об’єкти є окремими елементами матеріального світу (пластилінова анімація, лялькова анімація, сипка анімація);

- комп’ютерна – вид анімації в якому об’єкти створюються за допомогою комп’ютерної графіки (2D анімація, 3D анімація).

За методом анімування виділяють наступні види технологій:

- покадрова технологія – це технологія за якою кожен кадр малюється окремо;

- технологія “ключових кадрів” - полягає в створенні ключових кадрів, а інші кадри, що розміщені між ними малюються автоматично;

- технологія “захоплення руху” - відносно молода технологія, у якій об'єкти рухаються або змінюють форму внаслідок аналогічних дій реальними істотами або неживими об'єктами, до яких прикріплено датчики, що фіксують рух в просторі та передають дані до комп'ютера.

За типом параметрів об’єктів виділяють такі технології:

- руху - технології, що дозволяють передати рух об'єкта або його частин;

- форми “морфінг” - технології зміни форми. Часто використовуються для перетворення одного об'єкта в інший;

- кольорова анімація - технології трансформації забарвлення об'єкта.

1.3.6. Текст

Текст - друкарський шрифт, кегель якого рівний 20 пунктам. Будь-яка інформація, зображена за допомогою символів клавіатури комп'ютера. Це впорядкований набір речень, який призначений для того, щоб виразити певну

суть. Текстом може бути, наприклад стаття, звіт, наказ, інформативний лист, рекламний лист та інші документи.

Характеристики тексту:

- Кернінг - процес зміни розмірів міжбуквенних пропусків (інтервалів) між сусідніми буквами для покращення зовнішнього вигляду і легкості у читанні тексту. Цей параметр відповідає за індивідуальну роботу з кожною буквою і підбір її місцезнаходження залежно від вибраного шрифту, малюнку самої букви і її сусідніх букв, смислового навантаження слова і т. д. Значення кернінга встановлюється у відсотках від ширини пропуску шрифту.

- Трекінг - пропорційна зміна міжбуквенних і міжрядкових інтервалів при незмінному форматі набору.

- Інтерліньяж - міжбуквенний інтервал, відстань між базовими лініями сусідніх рядків.

- Міжбуквенний інтервал - відстань між окремими символами тексту.

- Міжрядковий інтервал визначає відстань між рядками в абзаці.

Шрифт - повний комплект друкарських літер певного типу й рисунку, необхідний для набору якого-небудь тексту. Шрифти характеризуються наступними параметрами:

- гарнітурою: об`єднання різних за кеглем та рисунком, але однаковим за характером накреслення шрифтів;

- нахилом: прямий, похилий, курсив;

- насиченістю: контурний, світлий, нежирний, напівжирний, жирний, наджирний;

- шириною: надвузький, вузький, нормальний, широкий, надширокий;

- ілюміновкою: з контуром, з тінню, штриховий, орнаментований, негативний, кольоровий;

- розміром (кеглем) в пунктах (1 пункт = 1/72 дюйма).

Найбільш популярними на сьогодні є два стандарти класифікації шрифтів: ГОСТ 3489.1-71 та німецький індустріальний стандарт.

Класифікація друкарських шрифтів, відповідно до ГОСТ 3489.1-71:

- група рублених шрифтів. Це шрифти, які не мають засічок;

- група шрифтів з ледь помітними засічками. Це гарнітури з дещо потовщеними кінцями вертикальних штрихів;

- група медієвальних шрифтів - гарнітури з помірною контрастністю штрихів, з засічками у вигляді плавного потовщення кінців основних штрихів, що наближаються за своєю формою до трикутника, переважно з похилими осями круглих літер;

- група брускових шрифтів - гарнітури з неконтрастними або малоконтрастними штрихами з довгими засічками, що сполучаються з основними штрихами під прямим кутом або з легким закругленням;

- група звичайних шрифтів - гарнітури з контрастними штрихами, з довгими тонкими засічками, що сполучаються з основними штрихами під

прямим кутом, інколи з легким заокругленням, округлі літери з вертикальними осями;

- група нових малоконтрастних шрифтів - гарнітури, що мають малоконтрастні штрихи з довгими засічками, переважно з закругленими кінцями, які сполучаються з основними штрихами під прямим кутом або з легким закругленням;

- група додаткових шрифтів, до цієї групи належать такі шрифти, побудова та характер малюнків яких дуже відрізняється від шрифтів шести основних груп.

1.3.7. Інтерактивність

Інтерактивність – безпосередня взаємодія користувача з персональним комп'ютером, що може відображатись у вигляді запиту або діалогу.

Відомі технології часткової та повної інтерактивності. До часткової інтерактивності належать технології, які забезпечують збереження інформації у структурованому вигляді - банки даних, бази даних. Інформація у цих технологіях надається як послуга, при цьому користувачу не дозволяється вводити нову інформацію. До повної інтерактивності належать технології, які забезпечують прямий доступ до великих обсягів інформації. Цей вид технологій об'єднує всі форми комунікації за допомогою комп'ютера: електронну пошту, телеконференцзв'язок, синхронний та асинхронний зв'язок, IP-телефонію та інші.

Інтерактивність цікава для користувачів мультимедіа не тільки тим, що суб'єкт може впливати на результат, але й тим, що суб'єкт несвідомо ототожнює себе творцем проекту.

1.4. Напрями застосування технологій мультимедіа

Напрями застосування сучасних мультимедійних технологій розширюються дуже швидко. Вже сьогодні немає такої галузі, де б не використовувались технології або складові мультимедіа.

Серед численних напрямів та галузей застосування мультимедіа технологій можна виділити декілька основних:

1. Освіта і навчання. Використання мультимедіа в освіті і навчанні передбачає особисте та бізнесове використання технологій мультимедіа. В майбутньому значення цієї області використання мультимедіа зростатиме, оскільки знання, що забезпечують високий рівень професійної кваліфікації потрібно постійно підвищувати. Навчання з використанням мультимедійних технологій проходить переважно у сфері виробництва та освіті. Економія часу, необхідного для вивчення конкретного матеріалу, в середньому складає 30%, а одержані знання зберігаються в пам'яті людини значно довше. Широкого

поширення в Internet набули системи дистанційного навчання і прийому іспитів. По електронній пошті студенти отримують завдання і консультації, а також літературу і методичні матеріали. Після вивчення запропонованого матеріалу і здачі декількох контрольних робіт студент зобов'язаний пройти онлайн-іспит (безпосередньо спілкуючись з викладачем в чаті або телеконференції), або відповідаючи на запитання, що з'являються на web-сторінці. Якщо всі іспити успішно складені, студент отримує поштою сертифікат або диплом.

2. Маркетинг і реклама. Не секрет, що зростання обороту коштів спостерігається в тих рекламних агентствах, які використовують для презентацій фірм мультимедіа технології. Галузь вітринної реклами є класичним прикладом використання мультимедіа. За допомогою такої реклами клієнти мають можливість самостійно отримувати інформацію, їх що цікавить. Наприклад, це можуть бути операційні зали банків, де таким чином виводиться інформація за пропозиціями кредитів, різним банківським операціям, зали на виставках і ярмарках, зали автосалонів, бюро подорожей, аеропорти, залізничні вокзали. Таким довідковими системами можна користуватися і в неробочий час, якщо екран знаходиться за скляною вітриною та оснащений клавіатурою. Перевагою цих систем є швидка реакція на отримання інформації та створення додаткової позитивної реклами товару, а також отримання статичної інформації по попиту в даній області ринку.

3. Системи для комп’ютерного моделювання. Системи призначені для комп’ютерного моделювання дозволяють досить природно представити реальність за допомогою рухомого зображення і звуку у поєднанні з інтерактивною здатністю такої системи. За допомогою таких систем моделюються танкові та повітряні битви, проводяться тренування пілотів. Взаємодія людини і комп'ютера дозволила створити кібернетичний простір, що представляється віртуальним тривимірним світом, який динамічно реагує на інтерактивне спілкування з користувачем. Віртуальні системи створюються, як правило, на базі комп'ютера і програм CAD (Computer Aided Design – проектування за допомогою комп'ютера). Такі системи вже не новина на споживчому ринку, і тепер замість простого спостереження за комп'ютерною грою або відеофільмом можна повністю зануритися в світ віртуальної реальності і за допомогою рукавичок та шолома не тільки дивитися, але і активно втручатися в події, що відбуваються на екрані.

4. Системи орієнтування. Останнім часом розробляються потужні програмні продукти, які можуть інтерактивно використовувати картографічний матеріал на основі банків даних. Для прикладу, розглянемо програму формування дорожніх маршрутів у якій користувач для отримання довідкової інформації вказує початковий і кінцевий пункти бажаного маршруту. Програма обчислює маршрут поїздки або альтернативні відрізки дороги – у разі пробки на дорозі – з такими параметрами, як загальна довжина маршруту, кілометраж

окремих відрізків, відгалуження, зупинні пункти і так далі. При бажанні можна отримати точний план вулиць по маршруту проходження в кінцеву точку. При використанні систем в туристичному обслуговуванні інформація про маршрут подорожі може супроводжуватися відповідними картинами та звуком.

5. Інформаційно-довідкові системи. Серед сучасних систем існує достатня кількість розроблених і тривалий час працюючих інформаційно-довідкових систем, що розроблені з використанням мультимедійних технологій. Серед закордонних можна виділити такі як інформаційна служба CAS Американського хімічного товариства, Он-лайнова версія автоматизованої системи аналізу та пошуку біомедичної літератури Medline, серед вітчизняних - Ескізний проект картографічної інформаційної системи та банку географічних назв для забезпечення потреб державного і регіонального управління "ГеоПростір", АВЕРС-Бухгалтерія, Довідник по новому бухгалтерському обліку та ЛІГА:ЗАКОН фірми «Самсон», бази даних та інформаційно-довідкові системи патентного бюро України. Більшість таких систем використовують гіпертекст та забезпечують дуже швидкий доступ до потрібної інформації.

6. Системи архівування і документування. Інформація, яка раніше зберігалася на плівках та дискетах, тепер розміщується на вінчестерах і CD/DVD дисках. При зберіганні даних використовуються різні системи архівації, більшість з яких керують текстом, графікою, окремими зображеннями і звуком за допомогою банків даних і розміщують їх на різних носіях інформації. Одна з найважливіших областей застосування мультимедіа технологій – це керування документами, договорами, рахунками, службовим листуванням і так далі. Мультимедійні дані заносяться на носій з одноразовим записом та можуть бути зчитані у будь-який момент часу. Банки зображень, які використовуються переважно в науково-технічній області, зберігають величезну кількість цифрових зображень, на підставі яких, можна провести комплексну статистичну обробку зображень та інше.

Контрольні запитання.

1. Поясніть, як Ви розумієте термін “мультимедіа”?
2. Назвіть та охарактеризуйте типи мультимедіа технологій?
3. Які Ви знаєте способи відтворення мультимедіа даних?
4. Наведіть приклад лінійного способу відтворення мультимедіа даних?
5. Наведіть приклад нелінійного способу відтворення мультимедіа даних?
6. Апаратне забезпечення мультимедіа та його приклади?
7. Класифікація програмного забезпечення мультимедіа?
8. Виділіть основні віхи в історії розвитку мультимедіа?
9. Класифікація мультимедіа складових?
10. Які Ви знаєте види комп’ютерної графіки?

11. Наведіть характеристики растрового зображення?
12. Наведіть характеристики векторного зображення?
13. Назвіть відомі типи зображень у комп'ютерних системах?
14. Що Ви розумієте під терміном "аудіо"? Складові аудіо?
15. Наведіть характеристики звукових сигналів?
16. Класифікація звукових сигналів?
17. Що таке мовний сигнал? Які особливості його формування?
18. Наведіть характеристики мовного сигналу?
19. Класифікація відеосигналів?
20. Охарактеризуйте основні параметри відеосигналу?
21. Назвіть методи оцінки якості відеосигналу?
22. Що таке гіпертекст? Наведіть приклади гіпертекстових систем.
23. Що таке анімація? Охарактеризуйте основні види анімації.
24. Класифікація технологій анімації?
25. Текст та його характеристики?
26. Параметри шрифтів?
27. Класифікація друкарських шрифтів?
28. Охарактеризуйте технології інтерактивності?
29. Дайте коротку характеристику основним галузям використання мультимедіа.

Використана література.

1. Воген Т. Мультимедиа: Практическое руководство./Пер. с англ.; Худ. обл. М.В. Драко. - Мн.: ООО "Попурри", 1997.
2. Михаэль Кирмайер. Мультимедиа. BHV- Санкт-Петербург, 1994
3. Гейтс Б. «Дорога в будущее». М.: Русская мысль, 1996.
4. Schmuck C. Introduction au multimedia. Technologies et marches. Paris: AFNOR, 1995.
5. Рудометов Е., Рудометов В. Аппаратные средства и мультимедиа (справочник). -С.Птб.: Питер, 2000. -416 с.
6. Порев В.Н. Компьютерная графика. -С.Птб.: BHV, 2002. -432 с.
7. Пінчук О. Проблема визначення мультимедіа в освіті: технологічний аспект // Нові технології навчання. – К., 2007. – Вип. 46. – С 55–58.
8. Всемирный доклад по образованию, 1998 г.: Учителя, педагогическая деятельность и новые технологии / ЮНЕСКО. – Париж: ЮНЕСКО, 1998. – 175 с
9. Основи нових інформаційних технологій навчання: [посібник для вчителів] / за ред. Ю.І. Машбиця. – К. : ІЗМН, 1997. – 264 с.
10. Шлыкова О.В. Культура мультимедиа:Учебное пособие.-М.:Фаир-Пресс, 2004. - 415 с.
11. Інформатика: Комп`ютерна техніка. Комп`ютерні технології: Підручник/ За ред. Пушкаря О.І.- К.: Академія, 2002.- 704 с.

РОЗДІЛ 2. ЗБЕРІГАННЯ МУЛЬТИМЕДІА ДАНИХ

2.1. Зберігання цифрових зображень

2.1.1. Основи зберігання зображень

Цифрове зображення – це зображення, збережене в двійковому коді. Зображення зберігається в цифровій запам'ятовуючій системі як файл, представлений прямокутною таблицею пікселів. Концептуально, монохромне зображення є функцією *f(x,y)*, дискретизованою двомірною сіткою. Кожне вибіркове значення називається пікселем. Для прикладу, у таблиці 2.1 представлено послідовність пікселів у файлі з монохромним зображенням, а на рисунку 2.1 – наведено саме цифрове зображення, що описується даною таблицею.

Таблиця 2.1

Послідовність пікселів у файлі з монохромним зображенням

1	2	3	..	256
257	*258*	259	..	512
513	514	515	..	768

Рисунок 2.1 – Монохромне цифрове зображення

В наведеному прикладі, файл складається тільки з градації сірих пікселів. Для визначення розміру файлу із зображенням необхідно кількість рядків матриці, кількість стовпців матриці та кількість байт, якими представляється один піксель перемножити між собою. Для нашого прикладу розмір файлу буде дорівнювати: 256 рядків*256 стовбців*1 байт/піксель = 65536 байт.

При виконанні за допомогою комп'ютера будь-яких операцій над зображеннями необхідно мати опис зображення, наданий у вигляді чисел. Розглянемо опис (кодування) кольорів числами.

Для двоколірного (бінарного) зображення, наприклад чорно-білого, досить одного двійкового числа (0 - чорний колір, 1- білий).

Для півтонових (сірих) зображень досить вказувати інтенсивність за допомогою одного числа. Наприклад, 0 може відповідати чорному, максимальне значення інтенсивності означає білий колір, а проміжні значення інтенсивності відповідають рівням сірого.

Для кольорових зображень використовують декілька чисел для опису характеристики кольору. Зазвичай використовують три числа, при цьому кожному кольору відповідає його положення у тривимірному просторі. Одиниця виміру вздовж кожної осі визначається кольоровою моделлю.

2.1.2. Кольорові моделі

Кольорова модель - це модель конкретизованої класифікації гами світлових кольорів прийнятних для людини, котра дає можливість класифікувати конкретний колір для подальшої можливості його відтворення.

Людське око може сприймати світлове випромінювання в діапазоні довжин хвиль від 380 до 770 нм – одночасно близько 10 тисяч різних кольорів. Хвилі різної довжини сприймаються оком неоднаково. Найбільш відчутним є зелений колір, потім йде червоний, а за ним – синій. Краще розрізняються кольори ближче розміщених об’єктів, ніж віддалених. Погано сприймається колір дуже маленьких об’єктів. Можливість розрізняти кольори є індивідуальною. На сприйняття кольору впливає також спосіб його відтворення: одні і ті ж зображення, візуалізовані на різних пристроях, мають різний вигляд.

Колір є важливим засобом підсилення враження при сприйнятті зображень і підвищення їх інформаційної насиченості. Відчуття кольору формується людським мозком в результаті аналізу світлового потоку, що попадає на сітчатку ока від об’єктів, які випромінюють або відбивають світло.

В відповідності до принципів формування зображення адитивним чи субтрактивним методами розроблені способи розділення відтінку кольору на складові компоненти, які називають кольоровими моделями. В адитивних моделях нові кольори утворюються шляхом додавання основного кольору до чорного. Змішування всіх основних кольорів дає білий колір, якщо значення їх інтенсивностей максимальні, і чорний, якщо вони дорівнюють нулю. Адитивні моделі використовуються в пристроях, які випромінюють світло. В субстрактивних моделях нові кольори утворюються шляхом віднімання основного кольору від білого. В цьому випадку змішування всіх основних кольорів дає чистий чорний колір, якщо значення їх інтенсивностей

максимальні, і чистий білий, якщо вони дорівнюють нулю. Субстрактивні моделі використовуються в пристроях, які відбивають світло. Найбільш поширеними є колірні моделі RGB, RGBA, CMY, HSB, CIE Lab.

Найбільш простою і природною моделлю представлення кольору є RGB. У даній моделі колір кодується градаціями складових каналів (Red, Green, Blue). Інші кольори та відтінки отримуються змішуванням визначеної кількості кожного з основних кольорів (рисунок 2.2).

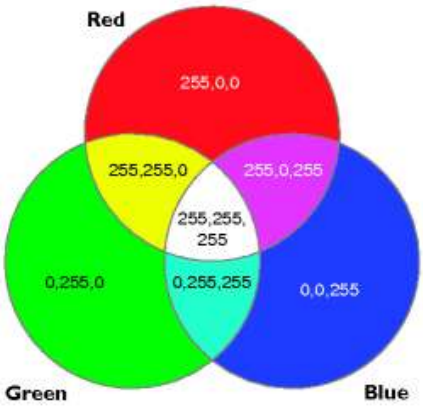

Рисунок 2.2 – Кольоровий простір моделі RGB

При збільшенні величини градації будь-якого каналу - зростає його інтенсивність під час синтезу. Кількість градацій кожного каналу залежить від бітового значення RGB. Зазвичай використовують 24-х бітну модель, у котрій визначається по 8 біт на кожен канал, і тому кількість градацій дорівнює 255, що дозволяє закодувати 2553=16.5 млн. кольорів. Ця модель називається адитивною і безпосередньо реалізується в сучасних сканерах, електронно-променевих трубках моніторів, телевізорах, цифрових фотоапаратах. Модель RGB широко використовується у Windows, зокрема відомі системні функції отримання повного кольору по його складових і виділення інтенсивності N-ної компоненти (N∈[R,G,B]) кольору GetNValue (RGB_Value). У якості одиниць виміру для RGB можуть використовуватись різні величини. В комп'ютерних системах часто використовуються умовні одиниці, наприклад, дробові від 0 до 1 або цілі від 0 до 255 і тому подібні. Умовність тут полягає в тому, що кольори не визначаються якимись абсолютними величинами, а все залежить від властивостей графічного пристрою відображення, його точності. Перевагами моделі RGB є:

- апаратна сумісність із графічними пристроями відтворення зображень;
- велика кольорова гамма, близька до можливостей людського зору;
- доступність багатьох функцій обробки зображення у програмах растрової графіки.

Модель RGB має і недоліки:

- кольори на екранах графічних пристроїв можуть відрізнятися від отриманих при виділені кольором;

- існує взаємозалежність колірних каналів (при збільшенні яскравості одного каналу в інших каналах яскравість зменшується).

Розвитком RGB-моделі є модель RGBA (Red, Green, Blue, Alpha), що дозволяє враховувати прозорість елементів зображення (канал Alpha).

На відміну від колірної моделі RGB, у якій для створення кольору використовується джерело світла, колірна модель CMYK ґрунтується на властивостях поглинання світла фарбами, якими виконується друк на папері. Коли біле світло потрапляє на напівпрозорі фарби, частина спектра поглинається. Результатом комбінування чистої блакитної (C), пурпурової (M) та жовтої (Y) фарб є чорний колір через поглинання або субтракцію всіх кольорів. Саме з цієї причини ці кольори називаються субтрактивними. Комбінування цих фарб для відтворення кольорів носить назву чотирьохфарбовий друк.

Практика показала, що модель CMYK адекватно описує принцип дії класичного друкарського друку, де кольорові зображення отримуються нанесенням на паперовий лист чотирьох фарб різної щільності.

На рисунку 2.3 зображено кольоровий простір моделі CMYK.

Рисунок 2.3 – Кольоровий простір моделі CMYK

Кожен колір в CMYK описується сукупністю чотирьох чисел, які називають кольоровими координатами. Кожне з цих чисел є відсотком фарби даного кольору у складовій колірної комбінації. Приклад: для отримання темно-помаранчевого кольору слід змішати 30 % фарби cyan, 45 % фарби magenta, 80 % фарби yellow і 5 % кольори black. Цей колір можна записати таким чином: (30,45,80,5). Іноді користуються іншим позначенням: C30M45Y80K5.

Недоліками колірної моделі CMYK є вузький діапазон кольорів, неточне відображення кольорів CMYK на моніторі і більша (в порівнянні з RGB) витрата пам'яті для реалізації.

Міжнародною комісією з освітлення в 1931 році розроблена і затверджена міжгалузевим стандартом колірна модель, яка після уточнення і доопрацювання отримала назву Lab (L*a*b). Ця модель розроблялася з метою

усунення недоліків моделей RGB і CMYK. Модель не прив'язана до пристроїв репродукції світла.
Будь-який колір в моделі визначається значенням яскравості L (Lightness) і двома хроматичними координатами - a і b (рисунок 2.4).

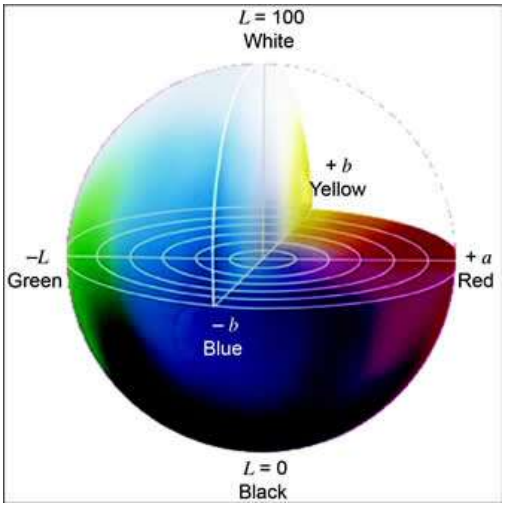

Рисунок 2.4 – Кольоровий простір моделі Lab

Хроматична координата a приймає всі значення кольору по колірному кругу - від зеленого до червоного. Координата b - від блакитного до жовтого. У природі не існує випромінювачів, які могли б відтворити діапазон колірних значень хроматичних координат a і b, тому модель застосовується в теоретичних дослідженнях, при обмінах інформацією про колір і для синтезу кольору в комп'ютерних програмах. Найбільшою перевагою моделі є її широкий колірний діапазон: система Lab передає всі кольори видимої частини спектру.

У моделі HSB всі кольори визначаються трьома координатами: відтінком (Hue), насиченістю (Saturation) і яскравістю (Brightness) (рисунок 2.5). Назва моделі утворена по перших буквах англійських назв колірних координат.

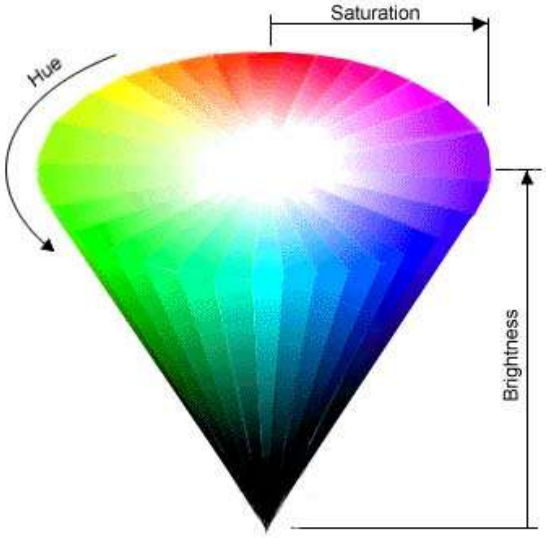

Рисунок 2.5 – Кольоровий простір моделі HSB

Колірним тоном або відтінком називається спектрально-чистий колір певної довжини хвилі, наприклад чистий червоний або чистий зелений. Насиченість описує чистоту кольору. Один і той же тон може бути тьмяним або насиченим. Зміну насиченості можна представити як розбавлення чистого хроматичного кольору білим або сірим. Яскравість характеризує інтенсивність, енергію кольору. Зміну яскравості можна представити як змішуванням чистого тону і чорного кольору. Із зменшенням відсотка чорного освітленість збільшується.

Модель HSB дуже зручна для користувача. У ній можна синтезувати нові кольори і отримувати різні варіанти заданого кольору, спираючись на інтуїцію і зображення кольору. Наприклад, ми знаємо, що чистий синій колір лежить на колірному крузі під кутом 240 градусів. Якщо потрібно змістити тон у бік пурпурного відтінку, то для цього досить збільшити кут повороту. Подібну стратегію синтезу кольору неможливо реалізувати в моделі RGB, оскільки важко передбачати наслідки навіть невеликих змін колірних координат. Ще однією безперечною перевагою системи HSB є її незалежність від апаратури.

У таблиці 2.2 подано значення кольорів у різних колірних моделях.

Таблиця 2.2

Значення первинних і ахроматичних кольорів

	RGB	CMY	HSB
Червоний	255, 0, 0	0, 255, 255	0, 240, 120
Жовтий	255, 255, 0	0, 0, 255	40, 240, 120
Зелений	0, 255, 0	255, 0, 255	80, 240, 120
Синьо-зелений	0, 255, 255	255, 0, 0	120, 240, 120
Синій	0, 0, 255	255, 255, 0	160, 240, 120
Червоно-синій	255, 0, 255	0, 255, 0	200, 240, 120
Чорний	0, 0, 0	255, 255, 255	160, 0, 0
Відтінки сірого	63, 63, 63 127, 127, 127 191, 191, 191	191, 191, 191 127, 127, 127 63, 63, 63	160, 0, 59 160, 0, 120 160, 0, 180
Білий	255, 255, 255	0, 0, 0	160, 0, 240

2.1.3. Графічні формати файлів

Для зберігання зображень використовують декілька десятків форматів файлів. Деякі з них розроблені окремими фірмами під конкретні програмні

засоби, інші створені науково-дослідними установами, у більшій чи меншій мірі пов’язаними співпрацею з Міжнародною організацією стандартів. Деяка частина з них стала стандартами і використовується в більшості графічних програм. За типами графічні формати файлів можна розділити на:

- растрові формати – призначені для зберігання растрових даних;
- векторні формати – призначені для зберігання векторних даних;
- метафайлові формати – можуть зберігати як растрові, так і векторні дані;
- формати сцени – містять додатково інструкції, що дозволяють програмі візуалізації відновити зображення цілком;
- формати анімації – прості дозволяють відображати зображення в циклі одне за іншим, а більш складні зберігають початкове зображення та різниці між двома зображеннями, які послідовно відображаються;
- мультимедійні формати – призначені для зберігання даних різних типів (графіки, звуку, відео) в одному файлі;
- тривимірні формати – містять опис форми і кольору об’ємних моделей.

Далі приведемо коротку загальну характеристику найбільш розповсюджених форматів цифрових зображень.

Формат GIF (Graphics Interchange Format – формат взаємообміну графікою) - 8-бітний растровий графічний формат, що розроблявся для мереж з низькими швидкостями передачі даних. Формат використовує до 256 чітких кольорів із 24-бітного діапазону RGB. GIF розроблено компанією CompuServe у 1987 році, і з того часу він набув широкої популярності у всесвітній павутині завдяки своїй відносній простоті та мобільності. Одними із головних особливостей формату є підтримка анімації та прозорості. У форматі використовується алгоритм стиснення LZW, який полягає в стисненні ряду однакових символів в один символ, помножений на кількість повторень. Анімаційні файли GIF дозволяють в одному файлі зберігати декілька зображень, які послідовно відтворюються. Остання версія формату GIF89a дозволяє виконувати черезрядкове завантаження зображень і створювати малюнки з прозорим фоном. Обмежена кількість кольорів обумовлює його використання переважно в електронних публікаціях. До переваг динамічних файлів GIF відносять невеликий об’єм файлу за рахунок стискання (до 40%), він не вимагає постійного зв’язку з сервером і повторного звертання до сервера, його просто розмістити на сторінці.

Формат JPEG - растровий формат зберігання графічної інформації, який використовує алгоритми стиснення з втратами, призначений для зменшення розмірів файлів, що мають плавні переходи кольорових тонів і відтінків. Дозволяє регулювати співвідношення між мірою стискання файлу і якістю зображення. JPEG стискає зображення, зберігаючи його повну чорно-білу

версію і більшу частину колірної інформації. На відмінність від алгоритму стискання GIF, який аналізує файли по рядках, JPEG розбиває зображення на області близьких кольорів. Формат JPEG найширше використовується для зберігання цифрових фотографій або графіки зі складними тінями та ефектами освітлення. Цифрові зображення у форматі JPEG використовуються в Web для зберігання фотографій товарів, об'ємних зображень і графіки з ефектами освітлення.

Формат PNG (Portable Network Graphics – мережева графіка, що переноситься) - растровий формат збереження графічної інформації, що використовує алгоритм стиснення без втрати якості. PNG був створений для заміни формату GIF, графічним форматом, який не потребує ліцензії для використання. Формат PNG підтримує три типи зображень – кольорові з глибиною кольору 8 або 24 біти і чорно-білі з градацією 256 відтінків сірого. Стиснення даних здійснюється без втрат, передбачено 254 рівня альфа-каналу та черезрядкову розгортку. Вважається, що PNG забезпечує краще стиснення, ніж GIF (на 10–30 %). Специфікація формату PNG включає можливості автоматичної корекції кольорів при перенесенні зображень між апаратними платформами і ефектів змінної прозорості.

Формат ICO призначений для зберігання значків файлів. Один ICO-файл містить один або декілька значків, розмір і колір кожного з яких задається окремо. Значки зберігаються в природному кольорі (True Color, глибина кольору 24 біт), High Color (глибина кольору 15 біт), або з фіксованою палітрою. По своїй структурі зображення у файлі ICO найбільш близькі до BMP, але принципово відрізняються від них наявністю додаткового зображення - маски, що накладається на задній план та дозволяє реалізувати повну прозорість малюнка.

Формат TIFF (Tagged Image File Format) – графічний формат, розроблений компанією Aldus (сучасна Adobe) у 1987 році, як один з базових універсальних форматів представлення високоякісних зображень, які використовуються у поліграфічній галузі. Забезпечує зберігання чорно-білих зображень та зображень з глибиною кольору 8, 16, 24 і 32 біт. Підтримується більшістю графічних, верстальних і дизайнерських програм та переноситься між платформами IBM PC і Apple Macintosh. Починаючи з версії 6.0 в форматі TIFF можна зберігати відомості про маски зображень. TIFF підтримує декілька алгоритмів стиснення без втрат якості: PackBits, LZW, CCITT Fax group.

Формат BMP – растровий bitmap-формат операційної системи Windows, який запам'ятовує одно і багатокольорові (RGB) ілюстрації у формі Pixel. Дозволяє використовувати палітри в 2, 16, 256 кольорів або повну палітру в 16 млн. кольорів. У даний час - один з найпоширеніших графічних форматів. Підтримується практично всіма графічними програмами. Із-за своєї простоти вимагає для виводу дуже мало системних ресурсів, тому основне його призначення - зберігання зображень користувача інтерфейсу операційної

системи. Зокрема, саме у форматі BMP зберігаються системні "шпалери", заставки, іконки і тому подібне.

Формат PCX – апаратно-залежний растровий формат, що використовується графічним редактором Paintbrush, текстовими редакторами і настільними видавничими системами типу Word і Ventura Publisher. Формат підтримує палітри 2, 16, 256 кольорів або повну палітру в 16 млн. кольорів. Призначається для зберігання інформації у файлі в такому ж вигляді, як і у відеоплаті. В даний час не конкурує з форматами, які підтримують краще стискання: GIF, JPEG і PNG.

Формат WMF - проміжний формат обміну для 16-бітових програм в ОС Windows. Формат підтримує векторну і растрову графіку у середовищі Windows, використовуючи палітри в 65 тис. і 16 млн. кольорів. У файлі з розширенням WMF використовуються команди опису графіки аналогічні командам ОС Windows для побудови графічних зображень. Файли цього формату відкриваються як векторними, так і растровими графічними редакторами. Однак, відсутність засобів для роботи зі стандартизованими палітрами кольорів, що прийняті в поліграфії, та інші недоліки обмежують його використання.

Формат CGM підтримує векторну і растрову графіку з використанням повної палітри (16 млн. кольорів) та палітри зі змінною кількістю кольорів. Він орієнтований на складні та високохудожні зображення, створює компактні файли та підтримує зберігання більше одного зображення у файлі.

Формат EPS описує як векторні, так і растрові зображення на мові PostScript фірми Adobe, яка є універсальною. В файлі одночасно може зберігатись як векторна, так і растрова графіка, шрифти, контури обтравки (маски), параметри калібрування обладнання, профілі кольору. Для відображення векторного вмісту використовується формат WMF, а растрового - TIFF. Екранна копія цього формату тільки в загальних рисах відображає реальне зображення. Дійсне зображення можна побачити тільки після друку, за допомогою спеціальних програм перегляду, або після перетворення файлу в формат PDF в застосунках Acrobat Reader та Acrobat Exchange.

Формат PDF – відкритий формат файлу, створений і підтримуваний компанією Adobe Systems, для представлення двовимірних документів у незалежному від пристрою відтворення та роздільної здатності вигляді. Кожен PDF файл може містити повну інформацію про 2D документ, таку як: тексти, зображення, векторні зображення, відео, інтерактивні форми та ін. Потужний алгоритм стискання з засобами керування підсумковою роздільною здатністю зображень забезпечує компактність файлів та їх високу якість. В грудні 2007 року, формат PDF було затверджено в якості стандарту ISO 32000.

Формат PSD - є одним з найпотужніших форматів за можливостями зберігання растрової графічної інформації. Він дозволяє запам’ятовувати параметри пластів, каналів, міри прозорості, множини масок і підтримує 48-

бітове кодування кольору, розділення кольорів і різноманітні моделі кольору. Однак відсутність ефективного алгоритму стискання інформації приводить до великого об'єму файлів. Відкривається файл у цьому форматі графічним редактором Photoshop.

Формат PhotoCD розроблений фірмою Kodak для зберігання цифрових растрових зображень високої якості. Внутрішня структура файлу забезпечує зберігання зображень з фіксованими величинами роздільної здатності, тому розміри будь-яких файлів незначно відрізняються один від одного і знаходяться в діапазоні 4-5 Мбайт. Кожній роздільній здатності присвоєно власний рівень, що відраховується від базового (512×768 точок).

Формат CDR - основний формат векторного графічного редактора CorelDRAW. Формат CDR став універсальним для інших програм завдяки використанню окремої компресії для векторних і растрових зображень, можливості вбудовувати шрифти, величезному робочому полю 45х45 метрів, підтримці багатосторінковості.

2.2. Зберігання аудіо

2.2.1. Аудіозапис та його особливості

Аудіозапис - процес запису звукової або мовної інформації з метою її збереження та подальшого відтворення. Аудіозапис базується на зміні фізичного стану або форми різних ділянок носія* запису - магнітної стрічки, грамофонної платівки, цифрових носіїв інформації тощо.

Аудіозапис є окремим випадком запису і відтворення інформації і здійснюється двома способами: акустичним і електроакустичним. У першому способі звукові коливання безпосередньо керують роботою приладу, що впливає на носій запису, в другому — спочатку перетворяться мікрофоном в електричні коливання, потужність яких підвищується підсилювачем до необхідного значення, після чого електричні коливання поступають в прилад, що впливає на носій, який безпосередньо проводить запис та зберігання аудіо.

Запис може бути здійсненим на аналогові та цифрові носії інформації у різних форматах. У випадку аналогового запису, електричний сигнал являє собою «аналог» акустичних коливань, в якому періодичні зміни звукового тиску можна представити як зміни електричної напруги, які фіксуються на магнітній стрічці або фотоплівці. У випадку цифрового звукозапису, електричний сигнал після підсилення поступає на аналого-цифровий перетворювач (АЦП), який перетворює аналоговий сигнал на цифровий, який можна зберегти у будь-якому форматі на цифровому носії інформації.

Розглянемо детальніше основні способи аудіозапису.

Механічний аудіозапис - система запису аудіо за допомогою зміни форми носія при механічній дії на нього. Механічний запис аудіо був першою практичною системою аудіозапису. При механічному записі звуку голка або

різець видавлює або вирізує на поверхні рухомого носія канавку, форма якої відповідає формі записуваних звукових коливань (рисунок 2.6).

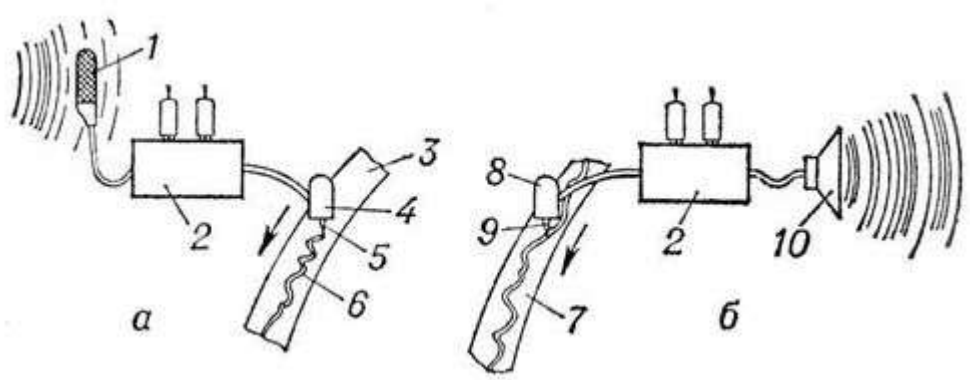

Рисунок 2.6 - Схема механічного аудіозапису (а) і його відтворення (б): 1 - мікрофон; 2 - підсилювач електричних коливань; 3 - носій запису; 4 - рекордер; 5 - різець; 6 - доріжка запису (канавка); 7 - механічна фонограма; 8 - звукознімач; 9 - грамофонна голка; 10 - гучномовець.

В процесі відтворення електропрогравачем грамофонна голка, рухаючись по звивині канавки, повторює ці коливання і передає їх або мембрані, випромінюючій звук через рупор, або електромеханічному перетворювачу звукознімача, що виробляє електричні сигнали. Механічний аудіозапис вперше практично здійснено в 1877 американським винахідником Т.А. Едісоном, що побудував фонограф із записом аудіо сигналів на валу, обернутому олов'яною фольгою. Надалі фольга була замінена воском. Механічний звукозапис на грамофонних платівках набув широкого поширення через простоту і зручність відтворення звуку в домашніх умовах.

Переваги механічного запису - масове тиражування грамплатівок, їх відносна дешевизна і простота звернення, а також можливість надійного зберігання запису тривалий час в металевих оригіналах (матрицях), основні недоліки - порівняно швидкий знос грамплатівки із-за безпосереднього механічного контакту грамофонної голки з нею, неможливість монтажу і стирання запису.

З появою цифрових технологій механічний запис звуку практично втратив своє застосування.

При фотографічному аудіозаписі у такт із звуковими коливаннями змінюється (модулюється) сила або форма світлового променя, падаючого на рухому кіноплівку. В результаті звук виявляється «сфотографованим». Після хімічного прояву на плівці утворюється затемнена доріжка запису, прозорість або ширина якої змінюється по довжині плівки відповідно до закономірності записаного коливання. Для відтворення аудіозапису фотографічну фонограму, яка рухається з тією ж швидкістю, з якою рухалася плівка при записі, просвічують променем світла, що проходить крізь доріжку запису і падаючим на фотоелемент, який перетворює коливання сили світла в електричні

коливання. Схема фотографічного аудіозапису і його відтворення наведена на рисунку 2.7.

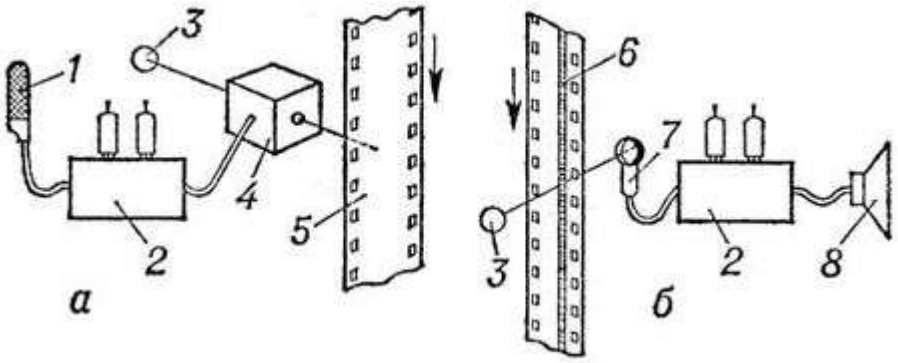

Рисунок 2.7 - Схема фотографічного аудіозапису (а) і його відтворення (б): 1 - мікрофон; 2 - підсилювач електричних коливань; 3 - джерело світла; 4 - модулятор світла; 5 - носій запису (кіноплівка); 6 - доріжка запису (фотографічна фонограма); 7 - фотоелемент; 8 - гучномовець.

Прообразом апаратів фотографічного аудіозапису є фотографофон, виготовлений у 1901 німецьким інженером Е. Румером. Фотографічний аудіозапис застосовували головним чином в звуковому кіно.

При магнітному аудіозаписі в такт із звуковими коливаннями намагнічуються окремі ділянки носія, що рухається через магнітне поле. Поле створюється магнітною головкою, через обмотку якої проходять посилені електричні струми мікрофону. При відтворенні відбувається зворотне перетворення: рухома магнітна фонограма порушує в магнітній головці електричні сигнали. Схема магнітного аудіозапису наведена на рисунку 2.7.

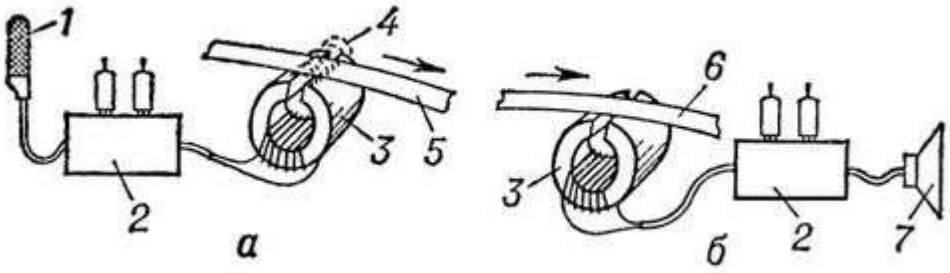

Рисунок 2.7 - Схема магнітного аудіозапису (а) і її відтворення (б): 1 - мікрофон; 2 - підсилювач електричних коливань; 3 - магнітна головка; 4 - магнітне поле головки; 5 - носій запису; 6 - магнітна фонограма; 7 - гучномовець.

Перший апарат для магнітного звукозапису на сталевий дріт (телеграфон) був запропонований в 1898 данським інженером В. Паульсеном. З 40-50-х рр. 20 століття набув поширення магнітний звукозапис на магнітну

стрічку за допомогою магнітофонів, які стали простими і зручними апаратами для виробництва звукозапису у домашніх умовах.

Цифровий аудіозапис полягає у перетворенні аудіо сигналу в цифрову форму, що виконується шляхом вимірювання миттєвих значень його амплітуди через рівні проміжки часу і представленні отриманих значень, у вигляді послідовності чисел. Ця процедура називається аналого-цифровим перетворенням, а пристрій для її реалізації - аналого-цифровим перетворювачем (АЦП). АЦП працює у заданій частоті дискретизації та розрядності модуляції. Зберігання цифрового аудіо здійснюється на компакт-дисках, iPod, жорстких дисках або будь-яких інших цифрових носіях інформації. Часто для зменшення об'єму аудіо файлі застосовується компресія

2.2.2. Аудіоносії

Аудіоносії - носії інформації, що слугують для зберігання звукових або мовних сигналів з метою наступного відтворення. На рисунку 2.4 подана класифікація аудіоносіїв.

Рисунок 2.8 – Класифікація аудіоносіїв

З появою цифрових технологій зберігання інформації, механічні та аналогові поступово втрачають свою популярність.

До найбільш вживаних та популярних аудіоносіїв можна віднести:

1. Digital audio tape - аудіоносій, розроблений компаніями Sony й Philips у середині 1980-х. Зовні він нагадує зменшену у два рази компакт-касету, оскільки являє собою магнітну стрічку 4 мм завширшки, укладену в захисний пластиковий корпус розміру 73 мм × 54 мм × 10.5 мм. Запис на магнітну стрічку - цифровий, при цьому використається 16-бітна

імпульсно-кодова модуляція без стиснення, як і в CD, а частота дискретизації може бути 48, 44.1 або 32 кГц. Запис здійснюється без втрати якості вихідного сигналу.

2. Компакт-диск - переносний оптичний диск для збереження цифрової інформації, діаметром 12 см (стандарт) або 6-8 см (міні-CD), використовується для запису значних обсягів інформації - аудіо-, відеопродукції, даних тощо. Зчитується компакт-диск лазерним променем з мікроскопічних канавок, на яких розміщено цифровий код. Під час програвання лазерний промінь зчитує код і створює сигнал, що практично не відрізняється від звучання оригіналу. Компакт-диски використовуються з жовтня 1982 року, масовий випуск почався з 1983 року. Вони розроблялися для запису музики, пізніше стало можливим використовувати запис інших типів даних. На 2009 рік компакт-диски залишаються стандартом для комерційного аудіозапису.

3. DVD - носій інформації у вигляді диска, зовні схожий з компакт-диском, однак має можливість зберігати більше інформації за рахунок використання лазера з меншою довжиною хвилі, ніж для звичайних компакт-дисків. Спеціально для запису аудіо інформації розроблено DVD-Audio - цифровий формат DVD, створений для високоякісного відтворення аудіо інформації. Диск формату DVD-Audio дозволяє записувати фонограми з різною кількістю звукових каналів. Підтримка багатоканального звуку є важливою перевагою DVD-Audio перед попередніми форматами. Існують дві версії формату DVD-Audio: просто DVD-Audio - тільки для звукового змісту й DVD-Audio - для звуку з додатковою інформацією. звукові доріжки у форматі DVD-Audio розташовуються в каталозі AUDIO_TS диска.

4. Міні-диск - магніто-оптичний носій інформації, вперше представлений компанією «Sony» 12 січня 1992 року. Використовується для зберігання будь-якого виду цифрових даних. Найбільше широко міні-диски використаються для зберігання аудіо інформації. Технологія міні-диск досі використовується деякими виробниками аудіо систем (в основному це «Sony», «Sharp» й «Aiwa»), однак, широкого поширення вона не отримала.

5. SACD (англ. Super Audio Compact Disc) - компакт-диск нового покоління, винайдений, як і CD, фірмами Sony і Philips. При запису SACD використовується формат DSD, що теоретично дає неперевершену якість звучання. Запис на SACD може містити від 1 до 6 звукових каналів. Для відтворення SACD потрібен спеціальний програвач, сумісний із цим форматом.

На диску може бути додатковий CD (тільки стерео) шар для сумісності зі звичайними програвачами.

6. Портативні аудіоносії з вмонтованою постійною пам’яттю. Прикладами таких носіїв є цифрові диктофони, mp3-плеєри, пристрої для читання аудіо книг, КПК, стільникові телефони та інші.

2.2.3. Формати аудіо файлів

Проведемо класифікацію та огляд найбільш поширених форматів зберігання аудіо файлів.

На рисунку 2.9 представлено класифікацію форматів аудіо файлів.

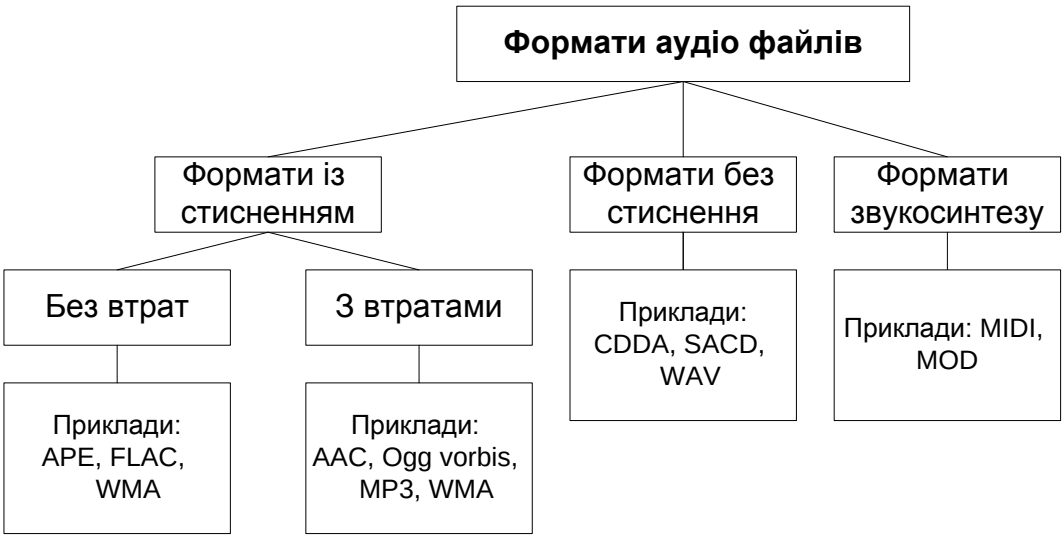

Рисунок 2.9 – Класифікація форматів аудіо файлів

Формат APE або Monkey's Audio - популярний формат кодування цифрового аудіо без втрат. Поширюється безкоштовно разом з відкритим вихідним кодом і набором програмного забезпечення для кодування і відтворення, а також плагінами до популярних плеєрів. Файли Monkey's Audio використовують наступні розширення: .ape для зберігання аудіо і .apl для зберігання метаданих. Незважаючи на відкритий вихідний код, Monkey's Audio не є вільним, тому що його ліцензія накладає значні обмеження на використання. Офіційно кодек Monkey's Audio випускається тільки для платформи Windows, хоча існує ряд неофіційних кодеків для GNU/Linux й Mac OS X, які в більшості випадків дозволяють лише стискати файли, перетворюючи їх в інший формат.

Формат FLAC (англ. Free Lossless Audio Codec - вільний аудіо формат стиснення без втрат) - популярний вільний кодек для стиснення аудіо даних. На відміну від кодеків з втратами Ogg Vorbis, MP3 і AAC, не видаляє ніякої інформації з аудіопотоку і підходить як для прослуховування музики на високоякісній звуковідтворювальній апаратурі, так і для архівації

аудіоколекції. На сьогодні формат FLAC підтримується багатьма аудіо-застосунками.

Формат AAC (англ. Advanced Audio Coding) - стандартний формат стиснення із втратами для аудіо файлів, розроблений як альтернатива формату MP3. AAC використовує два основних принципи кодування, для зменшення кількості даних необхідних для передачі високоякісного цифрового аудіо: вилучення неінформативних складових сигналу та вилучення надлишковість у кодованому аудіо сигналі.

Формат Ogg Vorbis - вільний формат стиснення аудіо даних, який був розроблений компанією Xiph.Org Foundation у 2002 році. Ogg Vorbis використовує власну психоакустичну модель при компресії з втратами для досягнення високих ступенів стиснення інформації. За замовчуванням Ogg Vorbis використає змінний бітрейт, при цьому його значення не обмежені і можуть варіюватися при мінімальних настройках на 1 kbps, а при максимальних - від 400 kbps до 700 kbps. Гнучкою є і частота дискретизації - користувачам надається будь-який вибір у межах від 2 кгц до 192 кгц. Така модель дозволяє отримати кращу якість відтворення при рівному ступеню стиснення. На сьогодні Ogg Vorbis вважається другим за популярністю форматом стиснення аудіо даних після mp3. Він широко використовується в файлообмінних мережах для передачі аудіо файлів.

Формат MP3 - ліцензований формат файла для зберігання аудіо-інформації. Сьогодні без перебільшення можна говорити, що розроблений майже два десятиліття назад формат стиснення аудіо даних MP3 є найбільш популярним форматом збереження аудіо на комп'ютері. MP3 є форматом стиснення з втратами, тобто частина аудіо інформації, яку (згідно психоакустичної моделі) вухо людини сприйняти не може або сприймається не всіма людьми, із запису віддаляється безповоротно. Ступінь стиснення можна варіювати, зокрема в межах одного файлу. Інтервал можливих значень бітрейту складає 8 - 320 кбіт/с. MP3 є лідером по поширеності, але при цьому не є кращим по технічних параметрах. MP3 непридатний для професійного використання музикантами через те, що дані стискаються з втратами, і при кожному редагуванні файлу якість погіршується.

Формат WMA (англ. Windows Media Audio) - ліцензований формат файла, розроблений компанією Microsoft для зберігання і трансляції аудіо-інформації. З самого початку формат WMA позиціювався як альтернатива MP3, але сьогодні Microsoft протиставляє йому формат AAC. Стандарт WMA є закритим стандартом і немає опису внутрішньої структури аудіо даних. Номінально формат WMA характеризується хорошою здатністю до стискання даних, що дозволяє йому «обходити» формат MP3 і конкурувати по цих параметрах з форматами Ogg vorbis і AAC.

Формат CDDA дає змого зберігати звукові сигнали з використанням PCM 44100 Гц, 16-біт стерео. Максимальний час всіх записів складає 74 хвилини,

мінімальний час треку - 2 секунди, максимальна кількість треків – 99. Звукові дані у форматі CDDA зберігають на носіях типу Audio CD.

Формат WAV розроблений компаніями Microsoft та IBM, який використовується для зберігання аудіо високої якості. Аудіо дані зберігаються у вигляді вибірки, над якою не проводиться жодної компресії. Розміри аудіо-файлу формату WAV є дуже великими і для зберігання секунди аудіо сигналу необхідно 1,4 мегабайти вільного місця. У структурі WAV файлу немає додаткової інформації, оскільки цей формат майже не використовується для зберігання великої кількості аудіо інформації. WAV формат представляє собою просту сукупність значень відліків сигналу, що подається на вхід цифро-аналогового перетворювача. Зазвичай це 8 або 16 біт. Значення відліків сигналу можуть подаватись в один (моно), два (стерео) або декілька каналів. Для відтворення вибірки на приймачі необхідно знати: бітрейт, частоту вибірки, число каналів.

Формат MIDI - стандарт передачі інформації між електронними музичними інструментами, розроблений 1983 року, що уможливлює комунікацію електромузичних інструментів, комп'ютера та іншого MIDI-сумісного обладнання, здійснювати з одного інструменту управління іншими. MIDI не передає звукової інформації - натомість MIDI працює з "повідомленнями", такими як висота та динаміка взятої на інструменті ноти, контрольні сигнали для таких параметрів як гучність, панорама, сигнали відліку часу для синхронізації темпу тощо. MIDI описує апаратний інтерфейс, який дозволяє з'єднувати синтезатори та комп'ютери різних виробників, описує протоколи зв'язку для передачі даних від одного пристрою іншому. MIDI-пристрої можуть взаємодіяти з програмним секвенсером, та посилати інформацію на синтезатор звукової карти комп'ютера. Запис та відтворення MIDI базується на пакетах даних, кожний з яких відповідає MIDI – повідомленню.

Формат MOD розроблений для створення, збереження та відтворення музичних композицій на ПК Amiga. Свою назву формат отримав від того, що став першим форматом, який зберігає свої фрагменти (наприклад, семпли) в інших файлах (принцип модульності). Файли цього формату мають, як правило, розширення. mod. Кожен файл формату MOD містить в собі оцифровані записи реального звучання інструментів - семпли. Композитор, що пише в форматі MOD, використовує програму, яка називається трекером, в якій вказує, який саме інструмент, в який час, який нотою і який з октав має прозвучати. Послідовність нот записується в список - трек, а кілька паралельних треків утворюють блок (паттерн). Сукупність паттернів утворює модуль - файл у форматі MOD.

2.3. Зберігання відео

2.3.1. Типи та формати відео носіїв

Відомо, що відеоматеріали можуть бути аналоговими або цифровими. В залежності від цієї особливості можна виділити три типи носіїв відео матеріалів: аналогові відеокасети, цифрові відеокасети та цифрові оптичні дискові носії. Для кожного з цих типів носіїв розроблено формати для їх запису та зберігання. На рисунку 2.6 наведено типи відео носіїв та їх формати.

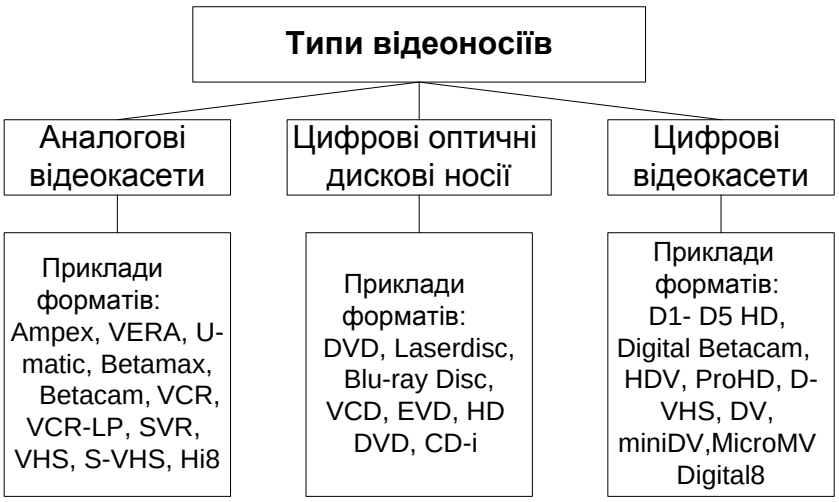

Рисунок 2.10 – Типи та формати відео носіїв

Опишемо найбільш поширені на сьогодні формати запису відео даних.

VHS (Video Home System) - найпоширеніший формат аналогового запису відеокасет розроблений японською компанією JVC в 1976 році. Запис здійснюється на відеокасету з плівкою шириною 12,6 мм та часом запису до 6 годин. До недавнього часу це був найбільш поширений формат відеозапису, перевагою якого був моментальний перегляд відеоматеріалу на відеомагнітофоні. Камери формату VHS досить громіздкі і на сьогодні вони морально застаріли. Останній великий постачальник VHS-касет Distribution Video Audio (Флоріда, США) оголосив про те, що партія відеокасет, випущена в жовтні 2008 року, була останньою.

Hi8 – формат запису відеокасет розроблений в 1989 році призначений для побутової й напівпрофесійної апаратури із записом на 8 мм. металопорошкову стрічку. Роздільна здатність зображення у цьому форматі 380*420. Тривалість запису - 120 хвилин, звук - Hi-Fi Stereo. У відеокамери для запису цього формату вмонтовують S-video роз'єм

для якісного виведення зображення. Hi8 є найкращим вибором для любителів якісного домашнього відео.

Betacam - сімейство форматів професійної відеозапису, що розробляються фірмою Sony. Базується на магнітному записі на 1/ 2 дюймову стрічку відеокасети. Протягом багатьох років широко використвується у телекомпаніях. У цьому форматі всі три складові яскравості і дві кольоророздільності передаються окремо одна від одної, що дає змогу досягти високої якості аналогового відео.

miniDV - цифровий напівпрофесійний формат відео, створений за рахунок спрощення та здешевлення професійного формату DV. Використовує intraframe-стискання у якому кожен кадр стискається незалежно, і ніяк не пов'язаний з попереднім або наступним. При стисненні кадр розбивається на

блоки 8×8 пікселів, і кожен з таких блоків стискається індивідуально, у відповідності з алгоритмом дискретного косинусного перетворення. Загальний коефіцієнт стиснення DV - 5:1. На кадр записаний у цьому форматі припадає по 12 похилих доріжок запису, при цьому зображення рівномірно розподіляється між ними. Обє'м MiniDV складає близько 13 гігабайт на одну годину відео. Форматі MiniDV використовуються спеціальні касети miniDV (ширина стрічки - 6,35 мм, швидкість - 18,831 мм/с), які можуть відтворюватися або з відеокамери, або на спеціальному цифровому відеомагнітофоні.

Digital8 – цифровий DV-відеоформат створений в 1999 році для запису на 8-ми мм магнітну стрічку та призначений для споживчого сектора DV. Відеокамери для запису цього формату орієнтовані на любителів та мають спрощену функціональність. Крім того для формату Digital8 випускалися переносні відеомагнітофони «Video Walkman».

VHS-D – цифровий відеоформат розроблений JVC спільно з Hitachi, Matsushita і Philips. Принцип запису відео у цьому форматі і касети аналогічні до формату VHS (але з більш якісною та дорогою стрічкою). Можливий запис сигналу як стандартної, так і високої чіткості. Основним перевагою VHS-D є

можливість запису кодованого сигналу високої чіткості із супутникової або кабельної трансляції, оскільки запис ведеться в потоковому форматі. На касету E-240 в режимі запису LS-3 поміщається понад 17 годин відео з якістю близькою до стандартної телетрансляції. Недоліками цього формату є відсутність виходу DV і входу RGB.

Laserdisc (LD) - перший комерційний оптичний

носій даних із записаним відеоматеріалом для домашнього перегляду. Однак, незважаючи на технологічну перевагу над VHS і Betamax, Laserdisc не був успішни на світовому ринку: в основному був розповсюджений у США та Японії. Технології, відпрацьовані у цьому форматі були використані в CD і DVD дисках. На сьогодні LaserDisc повністю поступився місцем DVD.

Blu-ray Disc або скорочено BD - це чергове покоління формату оптичних дисків, що використовується для зберігання відео високої чіткості (з роздільною здатністю 1920x1080 точок) і даних з підвищеною щільністю. Blu-ray використовує синій лазер з довжиною хвилі 405 нм. Через те, що на дисках Blu-Ray дані розташовані занадто близько до поверхні, перші версії дисків були вкрай чутливі до подряпин та інших зовнішніх механічних впливів через що вони вкладалися в пластикові картриджі. Вирішення цієї проблеми з'явилося в січні 2004, з появою нового полімерного покриття, що дало дискам неймовірний захист від подряпин і пилу. Сьогодні механічна міцність робочої поверхні BD в 100 разів більше, ніж DVD. Зменшення захисного шару дозволило збільшити масштабованість BR-дисків. Так, розробники планують незабаром підвищити кількість реєструючих шарів до 8, тим самим збільшивши загальну ємність до 200 GB. У форматі істотно перероблено механізм захисту вмісту від копіювання. Тепер в Blu-ray реалізована підтримка AACS (Advanced Access Content System), яка використовує 128-бітовий ключ, що змінюється через кожні 6 Kb записаних на диск даних. Крім того, AACS містить у собі механізм MMC (Mandatory Managed Copy), що дозволяє створити копію ліцензійного диска для перегляду його з HDD домашнього медіацентру або на портативному плеєрі, але, який запобігає її поширенню. Для додаткового захисту служить функція ROM-Mark, за допомогою якої кожному BR-диску надається свій унікальний ідентифікатор, що перешкоджає створенню його нелегальних копій, тому що робиться це на етапі виготовлення і тільки на сертифікованому обладнанні.

HD DVD (High Definition DVD) - технологія запису від компанії Toshiba (в співдружності з компаніями NEC і Sanyo). HD DVD подібний до технології Blu-ray Disc, що також використовує такі диски стандартного розміру (120 міліметрів в діаметрі) і синій лазер із довжиною хвилі 405 нанометрів. Одношаровий HD DVD має ємність 15 ГБ, двошаровий - 30 ГБ. Toshiba також анонсувала тришаровий диск, який зберігатиме 45 ГБ даних.

2.3.2. Мультимедійні контейнери

Мультимедійний контейнер - формат файлів, що може містити дані різних типів, стиснених різними кодеками і дозволяє зберігати аудіо, відео і текстову інформацію в єдиному файлі. Мультимедійні контейнери відкриваються більшістю медіаплеєрів. Зазвичай, в контейнерах знаходяться дані від різних кодеків. Поширений мультемедійний відео контейнер AVI може, наприклад, містити потік відео закодований кодеком Xvid в формат MPEG-4 і потік аудіо закодований LAME в формат MP3. Деякі із контейнерів можуть містити додаткову інформацію, таку як структура меню, або додаткові потоки аудіо. Інші контейнери можуть містити лише аудіо дані. Запис потоків аудіо та відео даних в один файл виконується мультиплексором. Під час програвання потік ділиться на аудіо та відео дані в демультиплексорі а потім передаються на відповідний кодек для декодування.

Охарактеризуємо найпоширеніші формати-контейнери:

1. Формат ASF (Advanced Systems Format) - розроблений фірмою Microsoft формат файлів, що містять потокове аудіо і відео. Формат може містити дані, закодовані за допомогою різних кодеків і підтримує синхронізацію потоків. ASF придатний як для локального відтворення відео, так і для передачі та відтворення по комп'ютерних мережах. Зазвичай використовується розширення файлу -. asf, а тип MIME - video / x-ms-asf. Крім того, файли, що містять аудіо інформацію, мають розширення .wma, а відеофайли - .wmv. Розширення. asf зазвичай використовується для файлів, що містять дані, закодовані кодеками сторонніх (не Microsoft) розробників. Особливістю формату ASF є можливість відтворення безпосередньо в момент завантаження даних по мережі, тобто потокового відтворення. Для використання даного формату необхідно ліцензування у фірми Microsoft.

2. Формат AVI (Audio Video Interleave) - RIFF-медіаконтейнер, вперше використаний Microsoft в 1992 році. Формат файлів з розширенням .avi відомий як медіа контейнер, який може містити відео та аудіо дані, стиснуті з використанням різних комбінацій кодеків, що дозволяє синхронно відтворювати відео зі звуком. Так, якщо MP3 і JPG файли побудовані на використанні тільки основного виду компресії даних (MPEG Audio Layer 3 і JPEG), AVI файл може містити різні види скомпресованих даних (наприклад, DivX - відео + WMA - аудіо або Indeo - видео + PCM - аудіо), залежно від того, який кодек використовується для кодування / декодування. AVI-файли можуть містити різні види стислих даних, наприклад DivX для відеоінформації та MP3 для аудіо. Всі AVI файли виглядають однаково «зовні» (мають розширення. AVI), але «всередині» вони можуть дуже сильно відрізнятись.

3. Matroska (Матрьошка) - проект, спрямований на створення відкритого, гнучкого, кросплатформеного формату мультимедійного контейнера та набору інструментів і бібліотек для роботи з даними. Цей проект є розвитком проекту MCF, однак він базується на EBML (Extensible Binary Meta Language - розширювана двійковий метамова). Використання EBML дозволяє розширити

формат без втрат сумісності зі старими програмами. Розширення файлів Matroska: .mkv - для відео (з субтитрами і звуком), .mka - для аудіофайлів і .mks - для субтитрів. Контейнер Matroska може містити велику кількість потоків аудіо, відео і субтитрів, та дає змогу зберігати в одному файлі цілий фільм. Matroska є відкритим проектом (open standard). Це означає, що для персонального використання вона абсолютно безкоштовна. Вихідний код всіх бібліотек, створених групою розробників проекту Matroska, поширюється на умовах ліцензії LGPL.

4. RealMedia - пропрієтарний формат потокового мовлення, який належить фірмі «RealNetworks Products and Services». У цьому форматі зберігається аудіо та відео інформація в мережі Інтернет. Файли RealMedia зазвичай мають розширення *.rm, *.ram або *.rmvb.

Основні переваги формату:

- дає змогу здійснювати довільне «перемотування» відео файлів з http серверів по осі часу, у тому числі і під час роботи через проксі;

- забезпечує прийнятну якість зображення і розбірливість мови при наднизьких бітрейта відео потоку;

- низький трафік, необхідний для трансляції відео по каналах зв'язку;

- низькі процесорні потужності, необхідні для відтворення потоку з низьким бітрейтом;

- сумісність старих файлів і потоків з новими версіями плеєра;

- можливість включення в потік слайдшоу, текстової інформації, інтерактивних елементів.

5. Формат Ogg Media являє собою контейнер для зберігання потоків даних, таких як відео, аудіо, і субтитри. Він забезпечує необхідні засоби для надійного транспортування файлу, контролю цілісності файлу, мінімізації кількості переміщень по файлу під час відтворення декількох потоків. Контейнер Ogg Media - модифікація контейнера Ogg, розрахованого на вільні кодеки, що підтримуються Xiph.Org. Формат часто використовуваних з контейнером AVI, за допомогою DirectShow.

6. Формат FLV (Flash Video) - формат файлів, медіаконтейнер, який використовується для передачі відео через Інтернет. Використовується такими сервісами відеохостингу, як YouTube, Google Video, Вконтакте, RuTube та іншими. FLV-файл - це бітовий потік, який є частиною відеостандарту H.263. Flash Player 8 і більше нові версії підтримують потокове відео On2 TrueMotion VP6, що дає змогу забезпечити більш якісне зображення, особливо при використанні низького бітрейта. Звук у FLV як правило закодований в MP3, однак іноді можуть використовуватися Nellymoser codec або ADPCM аудіоформат. У версії Flash Player 10 Beta доданий відкритий кодек SPEEX.

7. Формат TS - формат контейнера який інкапсулює пакети елементарних потоків та інших даних. Транспортний потік TS є форматом для передачі аудіо та відео даних, які описані в MPEG2. Мета розробки цього формату -

мультиплексування аудіо і відео даних і синхронізація їх виходу. Транспортний потік відкриває можливості для виправлення помилок транспортних засобів, таких як DVB і ATSC.

8. Формат MP4 (MPEG-4 Part 14) є мультимедійним контейнером. Був створений для цифрового телебачення та пересилання мультимедіа через Інтернет. Гарантує добру якість за низької швидкості пересилання. Файли цього формату мають розширення *.mp4. Формат MP4 уможливлює контроль швидкості пересилання і положення спектру сигналу, а також виправлення помилок. Формат передбачає не тільки зберігання аудіо та відео, а ще й анімованого / інтерактивного вмісту. Звукова доріжка в MP4 може бути моно, стерео і багатоканальна. Підтримуються формати звуку: MP3, Ogg Vorbis, WMA, AAC, VQF, AC3 та інші. Недоліки контейнера: слабка підтримка з боку апаратних DVD-плеєрів.

9. 3GP - формат даних створений для «мобільного відео». Використовується у мобільних телефонах. Адаптований до пропускних здатностей каналів зв'язку мобільної комунікація, як наслідок — якість зображення на другому місці. 3gp зберігає відео як MPEG-4 або H.263. Аудіо зберігається у форматах AMR-NB або AAC-LC.

2.3.3. Стандарти відеоданих

На сьогоднішній день існує багато способів захоплення, збереження і відтворення відеоданих на комп'ютері. З появою комп'ютерного цифрового відео стихійно стали виникати найрізноманітніші формати представлення відеоданих, що спочатку привело до деякої плутанини і викликало проблеми сумісності. Однак в останні роки завдяки зусиллям Міжнародної організації по стандартизації (ISO - International Standards Organisation) та експертної групи по кінематографії (MPEG - Moving Picture Expert Group) вироблені єдині стандарти на стандарти відеоданих.

Стандарт MPEG-1 розроблений в 1992 році для передачі відеоданих по низькошвидкісних мережах та для запису відеоданих на компакт диски у форматі Video-CD. Максимально можлива ширина відеопотоку обмежувалась порогом в 1150 Кб/с. Відеодані записані у стандарті MPEG-1 для систем PAL/SECAM забезпечують роздільну здатність 352*288 пікселів з 25 кадрами в секунду. Аудіо частина у стандарті містить стереозвук з частотою дискретизації 44,1 кГц, стиснутий згідно з MPEG-1 Layer II. Повнометражний фільм, записаний у цьому стандарті, займає два компакт-диски в стандарті Video-CD. Якість зображення на Video-CD дисках знаходиться на рівні VHS відеокасети. Стандант MPEG-1 використовується у форматах *.mpg та *.vcd. До переваг стандарту відносять:

1) широка розповсюдженість, що дає змогу відтворювати його на будь-якому комп’ютері;

2) MPEG-1 має стандартну ширину відеопотоку - 1150 Кбіт/с, роздільна здатність - 352x288 пікселів (для PAL/SECAM), 25 кадрів в секунду;

3) записані на компакт диск відеодані можна переглядати на різних мультимедійних пристроях (VCD, DVD програвачі та інші).

До недоліків стандарту відносять:

1) неефективні алгоритми стиснення відеоданих та звуку;

2) якість MPEG-1 можна змінювати тільки за рахунок збільшення ширини відеопотоку, що призводить до збільшення розміру кінцевого файлу і підвищує вимоги до швидкодії комп'ютера.

Стандарт MPEG-2 випущений у 1995 р. для оброблення відеоданих з якістю телевізійних трансляцій та шириною відеопотоку від 3 до 15 Мбіт/с. Сьогодні цей стандарт асоціюється з DVD дисками, поява яких викликала масове виробництво бюджетних DVD програвачів. Типова роздільна здатність для систем PAL/SECAM складає 720*576, при 25 кадрах на секунду, або 640*480 при 30 кадрах на секунду для систем NTSC. У порівнянні з MPEG-1, в аудіо частини додана підтримка багатоканального звуку (Dolby Digital 5.1, DTS). Збільшення ширини відеопотку і використання удосконаленого алгоритму стиснення відеопотоку покращило якість зображення MPEG-2, в порівнянні з VideoCD. MPEG-2 сьогодні використовується також у цифровому супутниковому телебаченні. Стандант MPEG-2 використовується у форматах *.mpg та DVD. До переваг стандарту відносять:

1) підтримка високої роздільної здатності та ширини відеопотоку;
2) підтримка черезрядкової розгортки, що робить стандарт оптимальним для роботи з відеоданими, які мають високу роздільну здатність;
3) за рахунок дуже високої ширини відео потоку стандарт MPEG-2 дає змогу змонтувати відеодані високої якості.

До недоліку стандарту MPEG-2 відносять високі вимоги до швидкодії комп'ютера.

Стандарт MPEG-4 почали розробляти ще в першій половині 90-х років минулого століття. У грудні 1999 року був представлений реліз цього стандарту, що одержав офіційний статус стандарту ISO / IEC. MPEG-4 задумувався, як спосіб передачі потокових медіа-даних, по каналах з низькою пропускною спроможність. Застосування у MPEG-4 більш складних алгоритмів компресії дозволило розміщати повнометражні фільми тривалістю півтори-дві години в прийнятній якості всього на одному компакт-диску. При однаковій ширині відеопотоку і певних умовах стиснення, якість зображення в MPEG-4 можна порівнювати з якістю зображень у MPEG-1 або MPEG-2. Застосування нових алгоритмів стиснення вимагає збільшення вимог до обчислювальних ресурсів, необхідних для якісної декомпресії зображення у стандарті MPEG-4. Для прикладу, на більшості компакт-дисків з фільмами у форматі MPEG-4 у системних вимогах зазначено PII-400, в той час як MPEG-1

відтворюється на комп'ютері з процесором P-100. До переваг стандарту відносять:

1) можливість створення відео з високою якістю і, відносно невисокою шириною відеопотоку, що робить можливим перегляд відео на більшості комп'ютерів і спрощує публікацію відео в Інтернеті;

2) стиснення відео практично будь-якої роздільної датності при відносно низькій ширині відео потоку;

3) перегляд на комп'ютері за допомогою стандартних засобів перегляду відео.

Стандарт H.261 – стандарт відеокодека організації ITU-T, затверджений в 1990 році для передачі інформації по мережі ISDN. H.261 належить до сімейства стандартів H.26x, алгоритм кодування якого розроблявся для передачі відео зі швидкістю в діапазоні від 40 кБіт/с до 2 МБіт/с. Стандартом підтримується два формати фреймів: CIF та QCIF із схемою 4:2:0.

Стандарт H.263 розроблений ITU-T в рамках проекту, який завершився в 1995/1996. Стандартом визначається формат даних для відеоконференцій через канали зв'язку з низькою пропускною спроможністю.

H.263 розроблявся як еволюційне вдосконалення на основі досвіду отриманого від розробки попереднього стандарту ITU-T — H.261, та стандартів MPEG-1 та MPEG-2. Розробку першої версії стандарту було завершено в 1995 р., перша версія була адекватною заміною стандарту H.261 на всіх бітрейтах.

Наступною вдосконаленою версією кодека, розробленою групою VCEG ITU-T (разом із MPEG) після H.263 є стандарт H.264, також відомий як AVC та MPEG-4 частина 10. Оскільки H.264 має істотні переваги над H.263, стандарт H.263 тепер вважається застарілим. Більшість продуктів для відеоконференцій мають підтримку стандартів H.264, та H.263 і H.261

2.4. Зберігання гіпертекстових документів

Гіпертекстовий документ в комп'ютерному виконанні - це файл (текст, графічне зображення чи інший фрагмент інформації), що має в своїй структурі посилання на інші файли (документи). Для реалізації можливостей гіпертексту використовують спеціальні мови розмітки документів у яких використовується набір анотацій до тексту, що надає інструкції стосовно структури тексту чи його відображення.

Гіпертекстові системи мають спеціальні програмні засоби побудови гіпертекстових зв'язків. Самі гіпертекстові посилання зберігаються в спеціальних форматах чи складають спеціальні файли. Такий підхід виправдовує себе для локальної системи, але не для розподіленої на безлічі різних комп'ютерних платформ. У Інтернет гіпертекстовий документ - це

звичайні ASCII- файл, який можна підготувати в будь-якому текстовому редакторі.

Найпоширенішим форматом зберігання та представлення гіпертекстових документів є HTML. Звичайно, HTML файли – це статичні документи. Використовуючи шлюзи у форматі HTML можна представити динамічну інформацію, наприклад вибірку з бази даних. Кожен документ HTML - це послідовність символів з репертуара. HTML використовує найбільш повну кодову таблицю UCS (Universal Character Set). Проте, однієї кодової таблиці недостатньо для того, щоб браузери могли правильно відтворювати документи HTML. Для цього браузерам потрібно «знати» специфічну кодову таблицю документа, яку автор має зазначати завжди в елементі meta із параметром charset. За замовчуванням використовується кодова таблиця ISO-8859-1, відома також як Latin-1. Розмітка в HTML складається з чотирьох основних компонентів: елементів (та їхніх атрибутів), базових типів даних, символьних мнемонік та декларації типу документа. У HTML гіпертекстові посилання вбудовані в тіло документа і зберігаються як його частина. Часто в системах застосовують спеціальні формати зберігання даних для підвищення ефективності доступу. Для перегляду HTML-розмітки документа можна використовувати будь-який текстовий редактор. Для перегляду документу відтвореного за правилами HTML-розмітки використовується оглядач.

Формат SGML визначає мову розмітки для документів. Спочатку SGML був розроблений для можливості спільного використання документів, що мають читатися машинами, у великих урядових та аерокосмічних проектах. Також він широко використовується в друкувальній та видавничій сфері, але складність документа ускладнила його широке розповсюдження для повсякденного використання. Три основні частини SGML документа, це SGML декларація; Document Type Definition; Зміст SGML-документа, принаймні, повинен бути кореневий елемент. SGML надає множину варіантів синтаксичної розмітки для використання різними прикладними програмами. Змінюючи SGML Declaration можна навіть відмовитись від використання кутових дужок, хоча, цей синтаксис вважається стандартним, так званим concrete reference syntax. SGML чітко визначає загальний синтаксис введення елементів розмітки до документу, а також окремі синтаксичні правила вживання тегів. Це дозволяє авторам створювати та використовувати різноманітну розмітку, обираючи найбільш відповідні за змістом теги, та називаючи їх зрозумілою своєю мовою.

Формат XML визначає набір базових лексичних та синтаксичних правил для побудови мови описання інформації шляхом застосування простих тегів. Цей формат достатньо гнучкий для того, аби бути придатним для застосування в різних галузях. Іншими словами, запропонований стандарт визначає метамову, на основі якої, шляхом запровадження обмежень на структуру та зміст документів визначаються специфічні, предметно-орієнтовані мови

розмітки даних. Ці обмеженння описуються мовами схем таких як DTD, RELAX NG або XML Schema. Прикладами мов, основаних на XML є: RSS, MathML, GraphML, XHTML, Scalable Vector Graphics, і також XML Schema.

2.5. Зберігання комп'ютерної анімації

Комп'ютерна анімація зберігається в універсальних графічних файлах (наприклад, у форматі GIF) у вигляді набору незалежних зображень, або в спеціалізованих файлах відповідних пакетів анімації (3ds Max, Blender, Maya) у вигляді текстур і окремих елементів, або в форматах, призначених для перегляду (FLIC) та застосування в іграх (Bink). Також, анімація може зберігатися у форматах, призначених для зберігання відео (наприклад, MPEG-4, AVI, MOV).

Розглянемо найбільш поширені формати зберігання комп'ютерної анімації.

Формат 3DS зберігає анімаційні файли створені в професійній програмній системі для роботи з трьохмірною графікою Autodesk 3ds MAX. Файл 3DS складається з блоків, кожен з яких містить корисні дані і, можливо, підблоки. Більшість блоків містить або дані, або підблоки, хоча є і змішані блоки.

Формат VRML - стандартний формат файлів для зберігання та демонстрації тривимірної інтерактивної векторної графіки, що використовується в Інтернет. У цьому форматі посилання можуть бути пов'язані з графічними компонентами, таким чином, що веб-браузер може отримувати веб-сторінку або новий VRML-файл з мережі Інтернет тоді, коли користувач клацає по будь-якому графічному компоненті. Рух, звуки, освітлення та інші аспекти віртуального світу можуть з'являтися як реакція на дії користувача або ж на інші зовнішні події, наприклад таймери. Особливий компонент Script Node дозволяє додавати програмний код (наприклад, Java або JavaScript (ECMAScript)) до VRML-файлу. VRML-файли зазвичай називаються світами і мають розширення *.wrl (наприклад: island.wrl). Хоча VRML-світи використовують текстовий формат вони часто можуть бути стиснуті з використанням алгоритму компресії gzip для того, щоб їх можна було передавати по мережі за менший час. Більшість програм тривимірного моделювання можуть зберігати об'єкти та сцени у форматі VRML.

MNG (Multiple-image Network Graphics) - формат графічних файлів для створення анімованих зображень. Підтримується в Mozilla/NN6 і Konqueror. Форматом MNG є спрощений растровий аналог Adobe Flash: кожен кадр складається з великої кількості шарів, які можна рухати один щодо одного, масштабувати і обрізати. Через це підтримка формату MNG досить складна. Також визначено дві спрощені версії специфікації: MNG-LC (низька складність) і MNG-VLC (дуже низька складність). Вони дозволяють реалізувати часткову підтримку формату MNG, щоб зменшити складність

реалізації програми. Це важливо для пристроїв з дуже обмеженими ресурсами: мобільні телефони і тому подібне. Кожен з кадрів може бути закодований в MNG як з втратою інформації (JPEG-компресія), так і без втрат (компресія LZ77, вживана в PNG).

Формат SVG - специфікація основаної на XML мови розмітки та формат файлів для двовимірної векторної графіки, як статичної, так і анімованої та інтерактивної. SVG може бути виключно декларативним, або містити описання сценаріїв. Зображення можуть містити зовнішні посилання шляхом застосування простих XLink-ів. Ця специфікація є відкритим стандартом, розробленим робочою групою англ. SVG Working Group. Анімація реалізована в SVG за допомогою мови SMIL (Synchronized Multimedia Integration Language), розробленої консорціумом W3C. Підтримуються скриптові мови на основі специфікації ECMAScript. SVG-елементами можна управляти за допомогою JavaScript. Застосування скриптів і анімації в SVG дозволяє створювати динамічну і інтерактивну графіку. У SVG забезпечується подієва модель, відстежуються події (завантаження сторінки, зміна її параметрів, події миші, клавіатури тощо). Анімація може запускатися по певній події (наприклад «onmouseover» або «onclick»), що додає графіці інтерактивність. У кожного елементу є свої власні події, до яких можна прив'язувати окремі скріпти.

SWF - формат Flash-файлів (анімації, ігор та інтерактивних додатків. SWF-файли можна переглядати за допомогою різних вільних плеєрів, наприклад, Gnash або swfdec. В основі формату лежить векторний морфінг, тобто плавне перетікання одного ключового кадру в іншій, що дає змогу робити складні мультиплікаційні сцени, задаючи лише кілька ключових кадрів для кожного персонажу. Flash використовує мову програмування ActionScript. Основний недолік Flash-додатків збережених у форматі SWF - висока вимогливість до ресурсів процесора. Специфікація SWF версії 4 була відкрита, але описи наступних версій продавалися тільки з підпискою про нерозголошення, і їх було заборонено використовувати для створення програвачів Flash.

2.6. Зберігання текстових даних

2.6.1. Структура текстових даних

Текстовими даними як правило називаються послідовності з підмножини знаків, що включає тільки друковані знаки (літери, цифри, розділові знаки) і деякі керуючі знаки (пропуски, табуляції, переведення рядка). Існують методи (наприклад, UUENCODE), що дозволяють представити в текстовому форматі довільні дані будь-якого формату.

Вимога до можливості розуміння вмісту даних людиною вносить додаткову надмірність до подання даних. Наприклад, число 123, для кодування якого достатньо одного байта, в текстовому вигляді кодується декількома

цифровими символами - так, в десятковій системі числення для цього потрібно три знака (123), у двійковій - сім знаків (1111011), в шістнадцятковій - два (7B).

Текстові дані можуть поділятися на рядки. На деяких платформах (в основному, в операційних системах сімейства Unix) розбивка на рядки кодується одним керуючим знаком з кодом 10 в таблиці ASCII. В інших ОС (MS-DOS, Microsoft Windows) - парою керуючих знаків з кодами 13 та 10. У Mac OS розбиття кодується одним знаком з кодом 13. Таке розбиття керуючими знаками аналогічне до принципу роботи друкарських машинок, через які здійснювалось введення в перших персональних комп'ютерах.

Також, знаки розбиття рядків використовувалися для керування механічними принтерами. На деяких платформах розбивка на рядки виконувась інакше - текст представлявся у вигляді послідовності записів фіксованої довжини, для чого більш короткі стрічки доповнювалися потрібною кількістю пробілів. Саме так представлялись дані на перфокартах, які служили засобом введення і зберігання даних.

Для більшої частини комп'ютерного обладнання і програмного забезпечення не важливий тип даних, які обробляються. Однак багато мережевих протоколів розраховані на роботу тільки з текстовими даними і не можуть обробляти довільну послідовність байтів.

2.6.2. Формати текстових файлів

В силу своєї простоти текстові файли нерідко використовуються для зберігання інформації (наприклад, для логів). Текстовий формат служить основою для багатьох більш спеціалізованих форматів. Наприклад: TeX, XML, вихідні тексти мов програмування.

Охарактеризуємо найбільш розповсюджені текстові формати.

Формат DOC (Document) - двійковий текстовий формат, розроблений компанією Microsoft у 1989 році на платформі IBM PC для текстового процесору Microsoft Word. Файли документів з розширенням. DOC представляють собою складні об'єкти, організовані за правилами структурованого сховища. Фактично, структуроване сховище - це окрема файлова система від Microsoft, приблизно така ж, як FAT або NTFS. Використовують її дуже багато Windows-додатків різних виробників, що використовують технологій OLE / COM / ActiveX. Сьогодні практично всі офісні програми підтримують даний формат файлів.

Open Document Format (OpenDocument, ODF) - відкритий формат файлів документів для зберігання й обміну офісними документами, доступними для редагування, в тому числі текстовими документами (нотатки, звіти, книги), електронними таблицями, рисунками, базами даних, презентаціями. Цей стандарт розроблений індустріальною спільнотою OASIS і базується на XML-

форматі, створеному для OpenOffice.org. 3 травня 2006 року прийнятий як міжнародний стандарт ISO/IEC 26300.

Office Open XML (OOXML) - міжнародний стандарт формату файлів для електронних документів, такі як електронні таблиці, діаграми, презентації та текстові документи, що базується на XML. Компанія Microsoft початково розробила специфікацію формату як наступника бінарних форматів Microsoft Office. Пізніше специфікація була передана до Ecma International для розробки стандарту Ecma-376, під керівництвом Ecma International Technical Committee. Стандарт Ecma 376 було опубліковано в грудні 2006. 01 квітня 2008 року компанія Microsoft оголосила про схвалення стандарту в ISO. У формі формату використовуєть стиснення файлів у відповідності до Open Packaging Convention. Стандарт спільно та публічно розроблений різними організаціями, доступний усім і може використовуватись без обмежень. Він є альтернативою приватним закритим форматам, включно з DOC, XLS і PPT (формати вживані у Microsoft Office), а також формату «Microsoft Office Open XML» (із цим форматом пов’язані різні ліцензійні обмеження, які не дають можливості використовувати його конкурентам).

Формат RTF (Rich Text Format) - вільний кроссплатформовий формат зберігання розмічених текстових документів, запропонований компанією Microsoft і іншими розробниками. Перша версія стандарту RTF з'явилася в 1987 році. Rtf-формат підтримуються більшістю сучасних текстових редакторів. Текст у форматі RTF кодується 8-бітними символами. Символи можуть кодуватися двома способами: кодами в рамках зазначеної таблиці кодування символів, або кодами в Юнікоді.

Формат INI використовується для зберігання звичайних текстових файлів, які можна редагувати та переглядати за допомогою будь-якого текстового редактора. INI-файл - це файл конфігурації, що містить дані налаштувань для Microsoft Windows, Windows NT та інших застосунків. INI-файл може містити: порожні рядки, коментарі, заголовки розділів, значення параметрів.

Формат DjVu використовує технологію стискання зображення з втратами, розроблену компанією AT&T спеціально для зберігання відсканованих документів — книг, журналів, рукописів та ін., де наявна велика кількість формул, схем, рисунків та рукописних символів, котрі роблять повноцінне розпізнавання такого документа надзвичайно складним та трудоємним. Цей формат є ефективним рішенням, коли необхідно передати всі особливості оформлення документа. Наприклад в історичних документах важливим є не тільки зміст, а й колір, фактура паперу, його дефекти: тріщини, сліди від згинів, клякси, залишені сліди предметів тощо.

Electronic Publication - відкритий формат електронних версій книг, розроблений Міжнародним форумом з цифровим публікацій IDPF. Файли цього формату мають розширення *.epub. Формат дозволяє видавцям

виробляти і поширювати цифрову публікацію в одному файлі, забезпечуючи сумісність між програмним і апаратним забезпеченням, необхідним для відтворення незашифрованих цифрових книг та інших публікацій. Zip-архів контейнера ePub містить тексти у форматах xHTML, HTML або PDF, опис видання в XML, поруч у папках - графіка, включаючи векторну (SVG), і вбудовані шрифти, таблиці стилів і т.д.

FictionBook - формат подання електронних версій книг у вигляді XML-документів, де кожен елемент книги описується своїми тегами. Стандарт забезпечує сумісність з будь-якими пристроями і форматами. XML дозволяє легко створювати документи, готові до безпосереднього використання і програмної обробки (конвертації, зберігання, управління) в будь-якому середовищі. Документи, зазвичай мають розширення Fb2, можуть містити структурну розмітку основних елементів тексту, деяка кількість інформації про книгу, а також можуть містити вкладення з двійковими файлами, у яких можуть зберігатися ілюстрації або обкладинка.

Формат BBeB (Broadband eBook) - формат для зберігання і відображення текстових даних, розроблений компанією Sony. Специфікація BBeB розроблена з метою вмістити текст великих книг (понад 250 сторінок) в маленький файл і вбудувати систему захисту від копіювання Open MG, щоб не допустити необмежене копіювання контенту.

Контрольні запитання

1. Що ви розумієте під цифровим зображенням?
2. Як визначити розмір файлу із цифровим зображенням?
3. Особливості сприйняття кольору людським оком.
4. Дайте визначення терміну «кольорова модель».
5. Охарактеризуйте основні колірні моделі.
6. Особливості зберігання растрових зображень та приклади форматів.
7. Особливості зберігання векторних зображень та приклади форматів.
8. Назвіть основні типи графічних форматів файлів.
9. Аудіозапис та його особливості.
10. Основні способи аудіо запису.
11. Наведіть класифікацію аудіоносіїв.
12. Аудіоносії та їх приклади.
13. Класифікація форматів аудіофайлів та їх приклади.
14. Аналогові відеокасети та приклади їх форматів.
15. Цифрові оптичні дискові носії та приклади їх форматів.
16. Цифрові відеокасети та приклади їх форматів.
17. Охарактеризуйте найпоширеніші формати-контейнери.
18. Стандарти відеоданих сімейства MPEG.
19. Стандарти відеоданих ITU-T.

20. Особливості зберігання гіпертекстових документів.
21. Формати зберігання гіпертекстових документів.
22. Особливості зберігання комп’ютерної анімації.
23. Структура та особливості зберігання текстових даних.
24. Формати текстових файлів.

Використана література

1. Борзенко А.Е., Фёдоров А.Г. Мультимедиа для всех. – 2-е изд. – М.: "КомпьютерПресс", 1996. – 252с.
2. Порев В.Н. Компьютерная графика. -С.Птб.: BHV, 2002. - 432 с.
3. Загуменнов А.П. Компьютерная обработка звука. -М.: ДМК, 2000. - 384 с.
4. Кречман Д.Л., Пушков А.И. Мультимедиа своими руками. – СПб.: БХВ Санкт-Петербург, 2003. – 538 с.
5. Загуменов А.П. Компьютерная обработка звука. – М.: ДМК, 2004. – 384 с.
6. Кинтцель Т. Руководство программиста по работе со звуком: Пер. с англ. – М.: ДМК Пресс, 2003. – 432 с.
7. Ковалгин Ю.О., Вологдин Э.И. Цифровое кодирование звуковых сигналов. – М.: ДМК Пресс, 2003. – 542 с.
8. Петелин Р.Ю., Петелин Ю.В. Музыкальный компьютер. Секреты мастерства. 2-е изд., перераб. и доп. – СПб., БХВ-Петербург; 2004. – 688 с.
9. Кирмайер М. Мультимедиа: Пер. с нем. – СПб.: БХВ Санкт-Петербург, 1999. – 192 с.
10. Окрасса М.А. Director 8.5/MX — М.: ДМК, 2003. — 432 с.
11. Титтел Э., Сандерс К., Скотт Ч., Вольф П. Создание VRML-миров: пер. с англ. – К.: Издательская группа БХВ, 1997. – 320 с.
12. Фролов М.И., Музыченко И.Ю. Мультимедиа для Windows. – М.: Майор, 2003. – 192 с.
13. Бондаренко М.Ф., Помазанов С.В., Шубін І.Ю. Програмні засоби створення мультимедіа – Навч. посібник, Харків: СМІТ, 2005. – 185 с.
14. Шубін І.Ю. Засоби мультимедіа в інформаційних технологіях. – Навч. посібник, Харків: СМІТ, 2005. – 118 с.

РОЗДІЛ 3. АЛГОРИТМИ СТИСНЕННЯ МУЛЬТИМЕДІА ДАНИХ

3.1. Особливості стиснення мультимедійних даних

Стиснення даних базується на усуненні надлишку інформації, яка міститься у вихідних даних. Прикладом надлишку є повторення в тексті фрагментів (наприклад, слів природної або машинної мови). Подібний надлишок зазвичай усувається заміною повторюваних послідовностей більш коротким значенням (кодом). Інший вид збитковості пов'язаний з тим, що деякі значення в даних, що стискаються зустрічаються частіше інших, при цьому можна замінювати дані, що часто зустрічаються більш короткими кодами, а ті, що рідко більш довгими (ймовірнісне стиснення). Стиснення даних, які не мають властивості надлишку (наприклад випадковий сигнал чи шум), неможливе. Так само зазвичай неможливо стиснути зашифровану інформацію.

Отже, стиснення даних - це процедура перетворення даних, яка проводиться з метою зменшення їх розміру чи об’єму. Розрізняють два методи стиснення даних: з втратами та без втрат.

Стиснення даних без втрат дає змогу відновити вихідні дані без спотворення та часто використовується при обробці та збереженні комп'ютерних програм і текстових даних. Методи стиснення даних без втрат дають змогу відновити перетворену інформацію з точністю до біта. Для кожної з складових мультимедіа, як правило, існують свої алгоритми стиснення без втрат.

Стиснення даних з втратами - метод стиснення даних, при якому вихідні дані відрізняються від оригіналу, проте вони можуть бути корисним для використання. Стиснення із втратами найчастіше використовується для графічних, відео та аудіо даних, а також для потокової передачі. В цьому контексті такі методи часто називаються кодеками.

Існують дві основних схеми стиснення із втратами:

- у трансформуючих кодеках фрейми даних діляться на невеликі сегменти та трансформуються в новий базисний простір для квантування. Результати квантування стискаються ентропійними методами;

- у кодеках з передбаченням попередні і/або наступні фрейми даних використовуються для того, щоб передбачити поточний фрейм. Помилка між передбаченими даними і реальними разом з додатковою інформацією про дані квантується і стискається.

У деяких системах ці дві схеми комбінуються шляхом використання трансформуючих кодеків для стиснення помилкових сигналів, згенерованих на стадії передбачення.

Перевага методів стиснення із втратами над методами стиснення без втрат полягає в тому, що перші забезпечують набагато кращий коефіцієнт стиснення, задовольняючи при цьому поставлені вимоги.

Коли методи стиснення даних застосовуються до файлів, то часто замість терміну "стиснення даних" вживають термін "архівування даних", стиснений варіант даних називають архівом, а програмні засоби, що реалізують методи стиснення називаються архіваторами.

3.2. Алгоритми стиснення зображень

Типове зображення, отримане цифровою фотокамерою, має розширення 2048×1024, для передачі кольору зазвичай використовується 24 біта/піксель. Розмір такого зображення приблизно 6 мегабайт, при цьому якість його гірша, ніж на звичайній фотографії 10×15. Для забезпечення кращої якості зображення потрібно в 5-6 разів збільшувати розширення. Тому дуже актуальними є алгоритми стиснення зображень, які дають змогу отримати якісне зображення при відносно невеликому розмірі файла.

У даному питанні наведено короткий огляд основних ідей алгоритмів стиснення зображень та їх реалізацій.

3.2.1. Кодування довжин серій

Кодування довжин серіїв (run-length encoding, RLE) або кодування повторів - найпростіший алгоритм стиснення даних, який оперує послідовностями, в яких один і той же символ зустрічається кілька разів підряд. При стисненні, рядок однакових символів, замінюється рядком, який містить сам символ, що повторюється, і кількість його повторів.

Для наочного пояснення алгоритму розглянемо зображення, що містить простий чорний текст на суцільному білому фоні. Як приклад приведемо довільний рядок зображення в чорно-білому варіанті у якому *B* представляє чорний піксель, а *W* - білий:

WWWWWWWWWWWWBWWWWWWWWWWWWBBBWW

Якщо ми застосуємо алгоритм кодування довжин довжин серіїв до цього рядка, то отримаємо наступне: 12W1B12W3B2W.

Останній запис інтерпретується як «дванадцять W», «одна B», «дванадцять W», «три B» і т. д. Таким чином, код представляє початкові 30 символів у вигляді всього лише 12.

Розглянемо ще один випадок, у якому вхідний рядок складається з множини символів, які не повторюються:

Вхідний рядок: ABCABCABCABCDDEFFFFFFFFF
Результат кодування: 1A1B1C1A1B1C1A1B1C1A1B1C2D1E8F

Наведений приклад демонструє, що об’єм кодування більший за вхідну послідовність. Для вирішення цієї проблеми алфавіт, в якому записані довжини серій, розділяється на дві (зазвичай рівні) частини. Алфавіт цілих чисел можна розділити на позитивні і негативні. Позитивні використовують для запису кількості однакових символів, що повторюються, а негативні - для запису кількості неоднакових символів. У цьому випадку результат кодування попереднього вхідного рядка буде наступний:

Результат кодування: -1ABCABCABCABC2D1E8F

Очевидно, що таке використання такого алгоритму ефективне для даних, що містять велику кількість серій повторів.

Алгорititm є досить ефективним для графічних зображень у форматі “байт на піксел” (PCX , BMP, TGA, TIFF).

Недоліки алгориму RLE очевидний - низька пристосованість до розповсюджених типів файлів, наприклад, текстових: у загальному випадку реально стиснути можна лише ланцюжки проміжків на початку абзаців. Саме тому цей метод можна ефективно використовувати лише у комбінації з вторинним кодуванням.

3.2.2. Алгоритм Хаффмана

Алгоритм Хаффмана - адаптивний жадний алгоритм оптимального префіксного кодування алфавіту з мінімальною надмірністю, який був розроблений в 1952 році доктором Масачусетського технологічного інституту Девідом Хаффманом .

Алгоритм складається з двох основних етапів:

- побудова оптимального кодового дерева;
- побудова відображення код-символ на основі побудованого дерева.

Розглянемо принцип роботи алгоритму Хаффмана на прикладі. Нехай вхідний алфавіт складається з чотирьох символів: a, b, c, d, частоти яких відповідно дорівнюють 1/2, 1/4, 1/8, 1/8.

Кодування Хаффмана для цього алфавіта подається таблицею 3.1.

Таблиця 3.1

Кодування Хаффмана

Символ	Частота	Вхідне кодування	Вихідне кодування
A	½	00	0
B	¼	01	10
C	1/8	10	110
D	1/8	11	111

Наведена таблиця дає змогу розмістити символи алфавіту на відповідних вершинах дерева Хаффмана (рисунок 3.1).

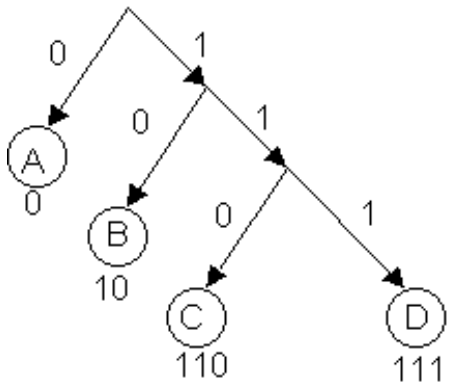

Рисунок 3.1 - Дерево Хаффмана

Стиснемо вхідну послідовність символів abaaacb, подану на вході як 00 01 00 00 00 10 01 у відповідності з алгоритмом Хаффмана. Результатом буде 0 10 0 0 0 110 10 (проміжки додано для зручності читання). Отже, 14 біт на вході дали 11 біт на виході.

Перевагами алгоритму Хаффмана є досить висока швидкість та коефіцієнт стиснення. Алгоритм Хаффмана має мінімальну надлишковість за умови, що кожен символ кодується окремим ланцюжком в алфавіті {0,1}.

Недоліком алгоритму є залежність коефіцієнту стиснення від близькості ймовірностей символів до від'ємних ступенів числа 2, це пов'язано з тим, що кожен символ кодується цілою кількістю біт. Найбільш помітно це під час кодування двосимвольного алфавіту: в цьому випадку стиснення неможливе, оскільки незважаючи на відмінності ймовірностей символів алгоритм фактично округляє їх до ½.

Властивості алгоритму Хаффмана:

- код Хаффмана оптимальний (у сенсі результату найменшої довжини в межі) в класі алгоритмів, що використовують префіксний код;
- таблицю частот символів можна будувати для кожної вхідної послідовності свою; використовувати фіксовану таблицю; будувати динамічно по поточній послідовності символів.

3.2.3. Алгоритм Лемпеля – Зіва – Велча

Алгоритм Лемпеля – Зіва – Велча (Lempel-Ziv-Welch, LZW) - це універсальний алгоритм стиснення даних без втрат, створений Абрахамом Лемпелем, Якобом Зівом і Тері Велчем.

Процес стиснення виглядає досить просто. Ми послідовно зчитуємо символи з вхідного потоку і перевіряємо, чи є в створеній нами таблиці стрічок

така стрічка. Якщо стрічка є, то ми зчитуємо наступний символ, а якщо стрічки немає, то ми заносимо в потік код для попередньо знайденої стрічки, заносимо стрічку в таблицю і починаємо пошук знову.

Алгоритму стиснення Лемпеля – Зіва – Велча для 256-и восьми бітових символів буде виглядати наступним чином:

```
початок алгоритму кодування
    Кладемо в словник символи a, b, c . під номерами 0 .. 255, відповідно.
    стрічка = пуста
    поки (потік не порожній)     // дивимось вхідний потік
        стрічка += наступний елемент
        якщо (стрічка є в словнику) то    продовжити;
        інакше
            // код(стрічка 0) — код, відповідний стрічка0 в словнику
        вивести код(стрічка – останній елемент)
        добавити_в_словник(стрічка)
        рядок = останній_елемент(стрічка)
        кінець якщо
    кінець поки
кінець алгоритму кодування
```

Реалізовуючи цей алгоритм, треба враховувати, що будь-яка комірка словника, за винятком найперших, які містять односимвольні ланцюжки, зберігає копію деякої іншої комірки, до кінця якої приписано один символ: використовується літерно-інкрементний словник. Таким чином, алгоритм LZW має важливу властивість префіксності: якщо деякий ланцюжок S міститься в словнику, то всі його префікси – також.

Графічне представлення наведеного алгоритму кодування подано на рисунку 3.2.

Алгоритму декодування на вході потрібно тільки закодований текст, оскільки він може відтворити відповідну таблицю перетворення безпосередньо по закодованому тексту.

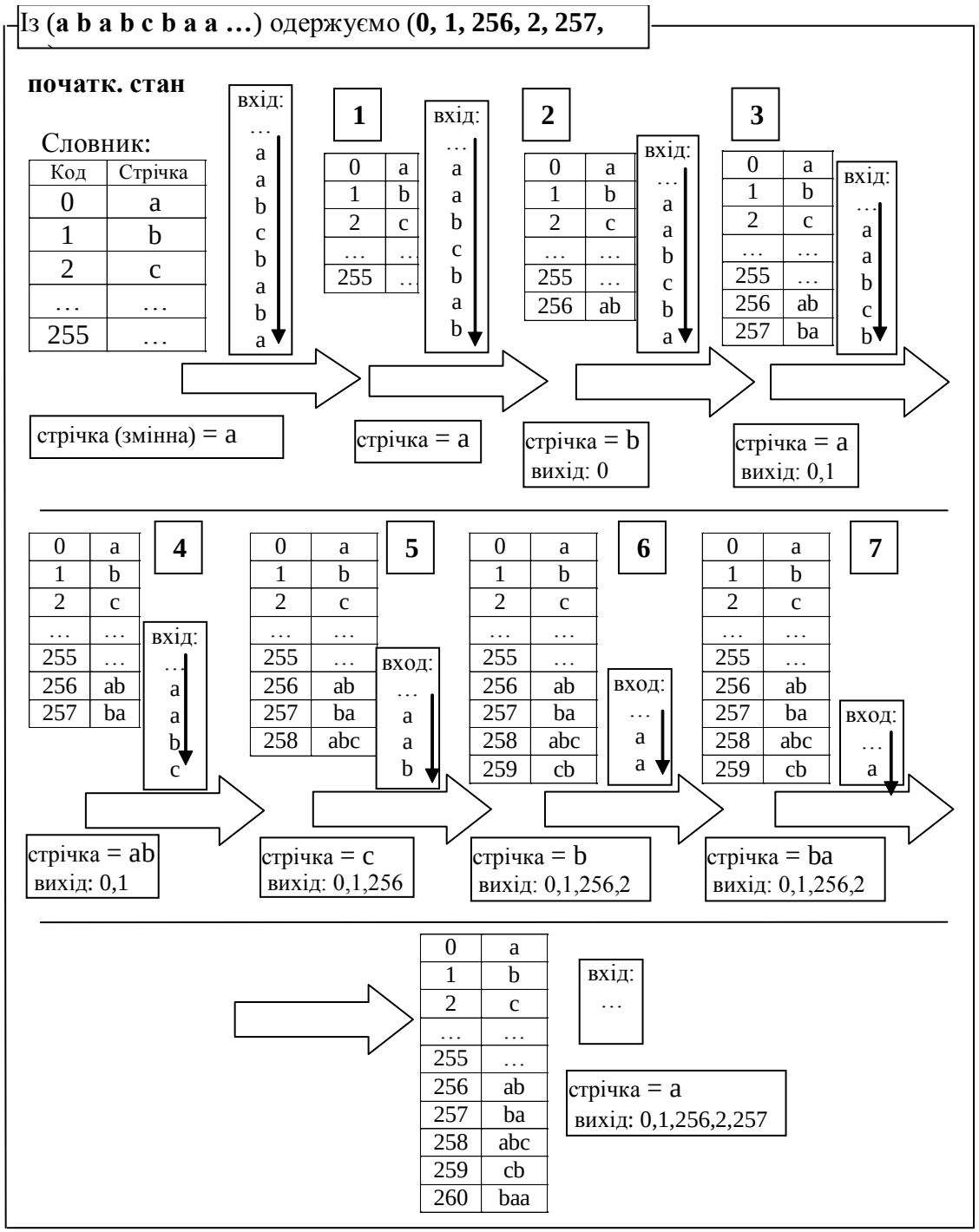

Рисунок 3.2 Графічне представлення алгоритму кодування LZW

початок алгоритму декодування
Кладемо в словник символи a, b, c під номерами 0 .. 255, відповідно.
поки (потік не порожній)
код = наступний код з потоку
1якщо (у словнику є слово для коду)
вивести вміст_словника(за кодом)
додати в словник(вміст_словника(за попереднім_кодом)+
+ перш_символ(вміст_словника(за кодом)))
2інакше
стрічка = вміст_словника(за попереднім_кодом)+
+ перш_символ(вміст_словника(за попереднім_кодом))

вивести(стрічку)
додати в словник(стрічку)
кінець якщо
кінець поки
кінець алгоритму декодування

Графічне представлення алгоритму декодування подано на рисунку 3.3.

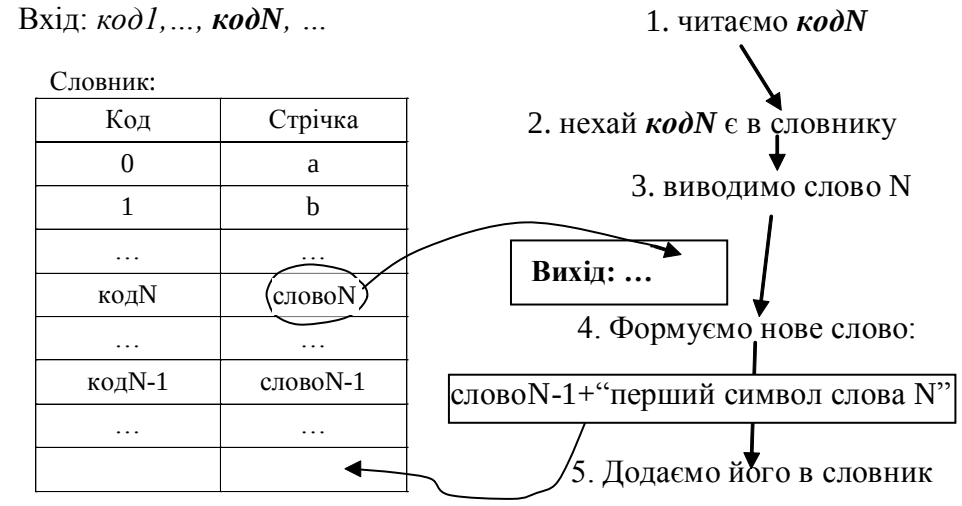

Рисунок 3.3 – Крок роботи алгоритму декодування LZW у випадку 1

Алгоритм Лемпеля – Зіва – Велча використовується у графічних форматах GIF, TIFF, PDF та вирізняється високою швидкістю роботи при кодуванні та декодуванні, невибагливістю до пам’яті та простою апаратної реалізації. Недолік алгоритму – менший коефіцієнт компресії порівняно зі схемою двоступеневого кодування.

3.2.4. Алгоритм JPEG

Алгоритм JPEG розроблений групою експертів в області фотографії (Joint Photographic Expert Group) спеціально для стиснення 24-бітних і напівтонових зображень у 1991 році. Цей алгоритм прекрасно обробляє зображення з безперервними тонами, у яких близькі пікселі мають схожі кольори. Алгоритм JPEG базується на дискретно-косинусному перетворенні (ДКП), що застосовується до матриці непересічних блоків зображення, розміром 8x8 пікселів. ДКП розкладає ці блоки по амплітудам деяких частот. У результаті виходить матриця, в якій багато коефіцієнтів, як правило, близькі до нуля, що дає змогу їх представити в грубій числовій формі, тобто в квантованому вигляді без істотної втрати якості зображенння.

Розглянемо роботу алгоритму детальніше. Припустимо, що стискається повнокольорове 24-бітове зображення. У цьому випадку, при стисненні зображення алгоритмом JPEG, можна виділити наступні етапи:

Етап 1. Переводимо зображення з колірної моделі RGB у модель YCbCr за допомогою наступного виразу:

$$\begin{vmatrix} Y \\ Cb \\ Cr \end{vmatrix} = \begin{vmatrix} 0{,}299 & 0{,}587 & 0{,}114 \\ 0{,}5 & -0{,}4187 & -0{,}0813 \\ 0{,}1687 & -0{,}3313 & 0{,}5 \end{vmatrix} \begin{vmatrix} R \\ G \\ B \end{vmatrix} + \begin{vmatrix} 0 \\ 128 \\ 128 \end{vmatrix} \quad (3.1)$$

Зворотнє перетворення легко виходить шляхом множення оберненої матриці на вектор $[Y, Cb,Cr]^T – [0,128,128]^T$:

$$\begin{vmatrix} R \\ G \\ B \end{vmatrix} = \begin{vmatrix} 1 & 0 & 1{,}402 \\ 1 & -0{,}34414 & -0{,}71414 \\ 1 & 1{,}772 & 0 \end{vmatrix} \left(\begin{vmatrix} Y \\ Cb \\ Cr \end{vmatrix} - \begin{vmatrix} 0 \\ 128 \\ 128 \end{vmatrix} \right) \quad (3.2)$$

Етап 2. Розбиваємо вихідне зображення на матриці розміром 8x8. Формуємо з кожної матриці три робочі матриці ДКП - по 8 біт окремо для кожної компоненти. При великих коефіцієнтах стиснення блок 8x8 розкладається на компоненти YCbCr у форматі 4:2:0, тобто, компоненти для Cb і Cr беруться через точку по рядках і стовпцях.

Етап 3. ДКП блоків зображення 8x8 пікселів. Формально пряме ДКП для блоку 8x8 можна записати у вигляді:

$$Y(u,v) = \frac{1}{4} \sum_{i=0}^{7} \sum_{j=0}^{7} A(u)A(v)X(i,j) \cos\left(\frac{\pi(i+0{,}5)}{8} u \right) \cos\left(\frac{\pi(j+0{,}5)}{8} v \right) \quad (3.3)$$

У результаті ДКП отримуємо матрицю, в якій коефіцієнти в лівому верхньому кутку відповідають низькочастотної складової зображення, а в правому нижньому - високочастотної.

Етап 4. Квантування. На цьому кроці відбувається відкидання частини інформації. Тут кожне число з матриці ділиться на спеціальне число з «таблиці квантування», а результат округляється до найближчого цілого:

$$Y^q(u,v) = Round\left(\frac{Y(u,v)}{q(u,v)} \right) \quad (3.4)$$

Причому для кожної матриці Y, Cb і Cr можна задавати свої таблиці квантування. Алгоритм JPEG навіть допускає використання власних таблиць квантування, які необхідно передавати декодеру разом із стисненими даними. Зрозуміло, що користувачеві складно самостійно підібрати 64 коефіцієнта, тому алгоритм JPEG використовує два підходи для матриць квантування. Перший полягає в тому, що в алгоритмом JPEG включені дві

рекомендовані таблиці квантування: одна для яскравості, друга для кольору. Другий підхід полягає в синтезі (обчисленні на льоту) таблиці квантуванні, що залежить від одного параметра R, що задається користувачем.

На етапі квантування здійснюється управління ступенем стиснення, і відбуваються найбільші втрати. Зрозуміло, що задаючи таблиці квантування з великими коефіцієнтами, ми отримаємо більше нулів і, отже, більший ступінь стиснення.

З квантуванням пов'язані і специфічні ефекти алгоритму. При великих значеннях кроку квантування втрати можуть бути настільки великі, що зображення розпадеться на квадрати однотонні 8x8. У свою чергу втрати у високих частотах можуть проявитися в так званому «ефекті Гіббса», коли навколо контурів з різким переходом кольору утворюється хвилеподібний «німб».

Етап 5. Перетворення матриці 8x8 в 64-елементний вектор за допомогою «зигзаг»-сканування (рисунок 3.4).

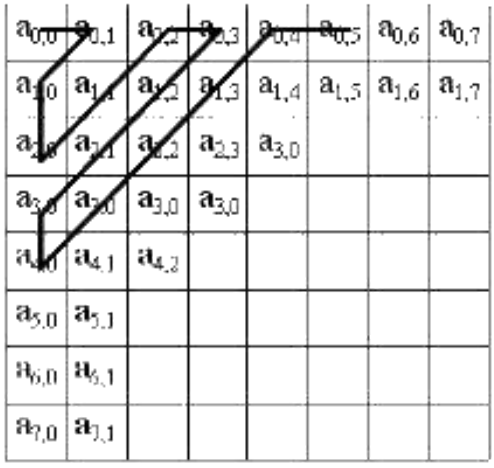

Рисунок 3.4 - «Зигзаг»-сканування

У результаті на початку вектора, як правило, будуть записуватися ненульові коефіцієнти, а в кінці - утворюватися ланцюжка з нулів.

Етап 6. Перетворення вектора за допомогою модифікованого алгоритму RLE, на виході якого отримуємо пари типу (пропустити, число), де «пропустити» є лічильником нулів, що пропускаються, а «число» - значення, яке необхідно поставити у наступну клітинку.

Наприклад, вектор 1118 3 0 0 0 -2 0 0 0 0 1 ... буде згорнутий в пари (0,1118) (0,3) (3, -2) (4,1)

Слід зазначити, що перше число перетвореної компоненти, по суті, дорівнює середній яскравості блоку 8x8 і носить назву DC-коефіцієнта. Аналогічно для всіх блоків зображення. Ця обставина наводить на думку, що коефіцієнти DC можна ефективно стиснути, якщо запам'ятовувати не їх абсолютні значення, а відносні у вигляді різниці між DC коефіцієнтом

поточного блоку і DC коефіцієнтом попереднього блоку, а перший коефіцієнт запам'ятати так, як він є. При цьому впорядкування коефіцієнтів DC можна зробити, наприклад, так (рисунок 3.5).

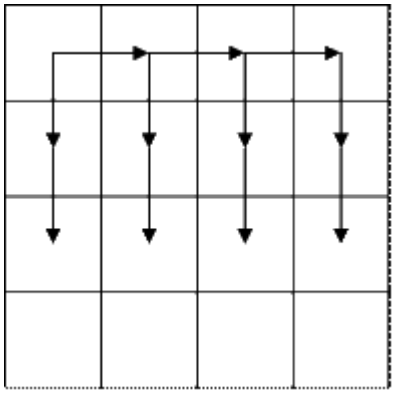

Рисунок 3.5 – Схема впорядкування DC коефіцієнтів

Решта коефіцієнти, які називаються AC-коефіцієнтами зберігаються без змін.

Етап 7. Впорядковуємо отриманні пари за допомогою нерівномірних кодів Хаффмана з фіксованою таблицею. Причому для DC та AC коефіцієнтів використовуються різні коди, тобто різні таблиці з кодами Хаффмана (див. рис. 3.5).

Процес відновлення зображення в цьому алгоритмі повністю симетричний. Алгоритм JPEG дозволяє стискати зображення в 10-15 разів без помітних візуальних втрат.

3.3. Алгоритми стиснення аудіо

Стиснення аудіо застосовується для зменшення обсягу аудіофайлів або для зменшення смуги пропускання потокового аудіо. Сьогодні розроблено декілька десятків алгоритми стиснення аудіо сигналів, які структурно та семантично відрізняються один від одного. Алгоритми стиснення аудіо можна поділити на алгоритми стиснення мовних сигналів та алгоритми стиснення звукових сигналів.

Завданням будь-якого алгоритму стиснення мовних сигналів є отримання цифрової послідовності відліків, яка вимагає мінімальної швидкості передачі і з якої декомпресор зможе відновити мовний сигнал з мінімальними втратами. Стиснення мовних сигналів відбувається з врахуванням особливостей мовної та фонетичної структури сигналу. Виділяють три класи алгоритмів стиснення мовних сигналів: стиснення форми мовного сигналу; стиснення параметрів мовного тракту людини; гібридне стиснення. Прикладами алгоритмів, які належать до першого класу, є: PCM, DPCM, ADPCM, DM, ATC та інші. До

другого класу відносять алгоритми: LPC, MBE, а також смугові, формантні, фонемні, ортогональні та гомоморфні вокодери. До третього класу відносять алгоритми: RELP, MPLPC, CELP та інші.

Алгоритми стиснення звукових сигналів реалізуються у комп'ютерних програмах, що називаються аудіокодеками. Розробка спеціальних алгоритмів стиснення звукових сигналів вмотивована тим, що загальні алгоритми стиснення мовних сигналів неефективні для роботи зі звуком та працюють в реальному часі. Розрізняють алгоритми стиснення звуку без втрат (англ. lossless), що дає змогу відновлювати вихідні дані без спотворень, та алгоритми стиснення з втратами (англ. lossy), при якому таке відновлення неможливе. Алгоритми стиснення звукових сигналів із втратами дають більшу ступінь стиснення на відміну від алгоритмів стиснення без втрат. У більшості випадків алгоритми стиснення звукових сигналів є складовою форматів звукових файлів (APE, FLAC, AAC, MP3, Musepack Ogg Vorbis, WMA та інші).

Оскільки кількість алгоритмів кожного класу є досить великою, тому в даному розділі розглянемо тільки загальні особливості роботи алгоритмів кожної групи.

3.3.1. Алгоритми стиснення форми мовного сигналу

Принцип роботи алгоритмів класу стиснення форми мовного сигналу, полягає у відновлені декомпресором форми сигналу. Відомо, що сусідні значення відліків слабо відрізняються одне від одного, тому з високою точністю можна спрогнозувати значення будь-якого відліку мовного сигналу на основі значень декількох попередніх відліків. При побудові алгоритмів стиснення названа закономірність використовується двома способами. По-перше існує можливість зміни параметри квантування в залежності від характеристик мовного сигналу. У цьому випадку крок квантування вимірюється, що дає змогу кодеру звузити динамічний діапазон сигналу. Крім того, деякі алгоритми виконують зміну параметрів квантування в рамках мовних складів, а інші змінюють ці параметри на основі аналізу статичних даних про амплітуду сигналу, одержану за відносно короткий інтервал часу.

Найпростішим алгоритмом класу стиснення форми МС є імпульсно-кодова модуляція (PCM). Більш складнішим алгоритмом є диференціальна ІКМ (DPCM). До класифікаційних ознак DPCM можна віднести наявність блоку лінійного прогнозування авторегресійних послідовностей (прогнозувальник) та використання багаторівневого (більше двох рівнів) квантувальника. DPCM кодує різницю між значенням поточного відліку і оцінкою значень попередніх відліків. Системи з DPCM, на відміну від PCM, забезпечують на порядок більшу завадостійкість при аналогічній якості мовного сигналу.

Дельта-модуляція (DM) представляє собою частковий випадок DPCM з використанням однорозрядного квантувальника. У DM приймачу передаються тільки квантовані значення різниці між поточним відліком та його прогнозованим значеням, причому, квантування виконується лише по двох рівнях. Необхідно відмітити, що DM забезпечує найкращий коефіцієнт стиснення серед усіх алгоритмів стиснення мовних сигналів. Кодеки алгоритму DM не втрачають працездатності при виникненні одиночних помилок та характеризуються простотою побудови компресора та декомпресора.

Одним з найпоширеніших алгоритмів класу стиснення форми мовного сигналу є адаптивно-диференціальна імпульсно-кодова модуляції (ADPCM) та його різновиди. Оскільки в аналоговому мовному сигналів неможливі різкі стрибки інтенсивності, то кодеки ADPCM стискають не саме значення амплітуди мовного сигналу, а його зміну в порівнянні з попереднім значенням, що дає змогу зменшити число розрядів для кодування відліку. Для представлення відліку мовного сигналу в форматі ADPCM значення зміни рівня сигналу стискається до чотирьохзначного числа, при цьому частота вимірювання амплітуди сигналу зберігається незмінною. Такий підхід забезпечує прийнятну якість мовлення ($3,5 \leq MOS \leq 4$) на швидкостях 16-32 Кб/с.

У алгоритмі підсмугового стиснення мовних сигнал фільтрується на декілька підсмуг. Кожен підсмуговий сигнал адаптивно стискається. Кількість біт для стиснення сигналу, визначається відповідно до кількості біт квантування призначених критерієм сприйняття. При стиснені кожного підсмугового сигналу шуми квантування обмежуються своєю підсмугою. Найкращі характеристики спостерігаються при збільшенні числа частотних піддіапазонів, а також при динамічній зміні кількості біт на вибірку від одного піддіапазону до іншого.

Алгоритми цього класу дають змогу ефективно стискати будь-які сигнали, а їх функціонування не залежать від типу сигналів. Їх перевагами є: стійкість до широкого діапазону характеристик джерела сигналу, простота реалізації, невисокі швидкості передачі. Недоліком алгоритмів цього класу є низька ефективність при роботі з сигналами, в яких спостерігаються різкі стрибки амплітуди.

3.3.2. Алгоритми стиснення параметрів мовного тракту людини

В основі принципів побудови алгоритмів класу стиснення параметрів мовного тракту людини закладено класичну модель створення мови, відповідно до якої всі голосові зв’язки являються джерелом збудження, а голосовий тракт – нелінійним фільтром, який формує спектр мовного сигналу. При цьому параметри джерела та фільтра -нестаціонарні. Кодеки, побудовані

згідно з класом стиснення параметрів мовного тракту людини називають вокодерами.

Робота вокодерів базується на моделі джерела, з якого отримується інформація про параметри мовного сигналу. Результатом стиснення є коди параметрів джерела мовного сигналу. Тип вокодера залежить від способів знаходження моделі системи та її параметрів.

При вокодерному стисненні мовний тракт людини представляється нелінійним фільтром із змінними в часі параметрами, що збуджується джерелом білого шуму (при формуванні приголосних звуків) або послідовністю тонових імпульсів (при формуванні голосних звуків) (рисунок 3.6).

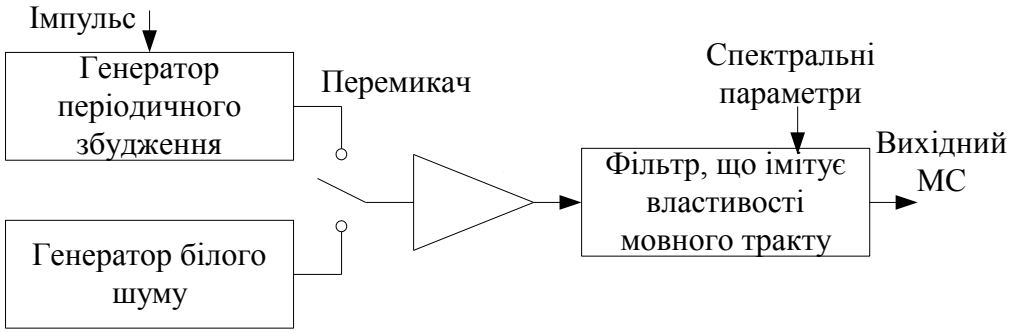

Рисунок 3.6 - Схема моделі функціонування мовного тракту людини

Компресор вокодерах обробляє наступну інформацію: параметри фільтра, який формує мовний сигнал; вказівник на голосний/приголосний звук; потужність сигналу збудження; висота тону для голосних звуків.

За принципами аналізу та синтезу мовного сигналу, вокодери поділяють на смугові (канальні), формантні, фонемні, ортогональні та гомоморфні.

Одними з перших алгоритмів класу стиснення параметрів мовного тракту людини були канальні вокодери, які використовують слабку чуттєвість слуху людини до незначних фазових зсувів сигнал. Для елементарних відрізків мовного сигналу за допомогою вузькосмугових фільтрів визначається амплітудний спектр. Сигнали з виходів фільтрів детектуються, пропускаються через фільтри низької частоти, дискретизуються та піддаються двійковому стисненню. Таким чином визначаються: параметри мовного тракту людини, висота тону, збудження мовного сигналу, вказівник на голосний/приголосний звук. Комп'ютерні засоби реалізації канальних вокодерів розробляють в цифровій та аналоговій формі. Канальні вокодери забезпечують задовільну якість мовлення на швидкостях порядку 2,4 Кб/с.

У формантних вокодерах огинаюча спектру мовного сигналу визначається комбінацією формант (резонансних частот голосового тракту). Основні параметри формант - центральна частота, амплітуда та ширина.

У ортогональних вокодерах огинаюча спектру розкладається в ряд по вибраній системі ортогональних базисних функцій. Обчислені коефіцієнти цього розкладання передаються декомпресору.

Гомоморфне оброблення сигналів дає змогу представити мовний сигнал як тимчасову згортку імпульсної перехідної характеристики мовного тракту з сигналом збудження. У частотній області згортка імпульсної перехідної характеристики відповідає добутку частотної характеристики мовного тракту та спектру сигналу збудження.

Вокодери, що базуються на принципах алгоритмів лінійного прогнозування представляють мовний тракт людини лінійним фільтром з безперервно імпульсною перехідною характеристикою, у якому кожне чергове значення відліку сигналу визначається як лінійна комбінація значень попередніх відліків. У такому вокодері мовний сигнал ділиться на блоки, для кожного з яких визначаються коефіцієнти фільтру прогнозування. Ці коефіцієнти квантуються та передаються декомпресору. Потім сигнал пропускається через фільтр, частотна характеристика якого зворотня частотній характеристиці мовного тракту. На виході фільтру отримується помилка прогнозування. У результаті таких перетворень набагато виразніше виявляється довготривала кореляція в сигналі, що забезпечує точніше визначення значення частоти основного тону та дає змогу виділити вказівник на голосний/приголосний звук. Швидкості кодів для вокодерів побудованих на базі алгоритмів лінійного передбачення при задовільній якості мовлення, змінюються від 1 Кб/с до 2,4 Кб/с.

Недоліком алгоритмів класу стиснення параметрів мовного тракту людини є синтетична якість мови. При цьому навіть суттєве збільшення швидкості передачі практично не покращує якості мови. Вокодери вмонтовують в обладнання спеціалізованих систем зв’язку, де головне не якість мови, а високий коефіцієнт стиснення.

3.3.3. Алгоритми гібридного стиснення

Алгоритми гібридного стиснення поєднують переваги алгоритмів класу стиснення форми мовного сигналу та стиснення параметрів мовного тракту людини.

Алгоритми стиснення мовного сигналу класу гібридного стиснення є найбільш ефективними серед відомих алгоритмів стиснення. Вони використовують відому інформацію про мовний сигнал для покращення якості мовлення та зменшення швидкості результуючого коду. Зокрема при стисненні мовних сигналів використовується замаскований слуховий шум, слухова частота розширення, слухова фаза нечутливості, складові зміни енергії, довготермінові звукові властивості тракту, крок квантування.

У алгоритмах класу гібридного стиснення, мовний сигнал дискретизується з метою одержання необхідних параметрів мови. Однак, замість безпосереднього стиснення висоти, модуляції та інших характеристик мовного сигналу, вони використовуються для синтезу фрагменту сигналу, з якого вони були отримані. Синтезований фрагмент мови, звичайно, тривалістю від 10 до 30 мс., порівнюється з початковим сигналом. Якщо вони співпадають з прийнятним допуском, то параметри мови залишаються без змін. Якщо ж реальний та синтезований фрагменти мовного сигналу відрізняються більш ніж на задану величину, то параметри мови коректуються для досягнення необхідного збігу. Кінцевим етапом процедури зворотного зв'язку є аналіз шляхом синтезу: редагування параметрів мови для забезпечення синтезу форми мовного сигналу, найбільш наближеної до його початкової форми. Після визначення значень параметрів сигналу вони співставляються з характеристиками кодової книги. При виявленні збігу, замість значення параметра, використовується його положення в кодовій книзі, що суттєво зменшує обсяг інформації для передачі.

За описаними вище принципами працюють алгоритми збудження на основі кодових книг (CELP). Кодеки CELP на швидкостях до 4,8 Кб/с забезпечують задовільну якість мовлення. Головним недоліком кодеків типу CELP є висока часова та апаратна складність виконання.

Для одержання стисненого мовного сигналу кодеки з регулярним імпульсним збудженням (RPE) використовують лінійний прогнозувальник, який покращує показник кореляції між відліками вхідного сигналу. Якщо прогнозування виконується добре, то на виході прогнозувальника формується практично білий шум з рівномірним спектром. Для декомпресії стисненого мовного сигналу отриманий білий шум з рівномірним спектром фільтрують фільтром лінійного прогнозування. Проте, через наявність в мовному сигналі квазіперіодичних формантних складових, лінійний прогнозувальник не може усунути довготривалої кореляції з висотою тону формант, і вони будуть явно відображатись у спектрі помилки прогнозування.

У кодеках з багатоімпульсним збудженням (MPE) в якості сигналу збудження використовується послідовність з чотирьох - шести коротких імпульсів. Тимчасове положення кожного імпульсу та їх амплітуди визначаються в процесі виконання процедури аналізу шляхом синтезу, яка виконується до досягнення максимальної кореляції між початковим та синтезованим сигналом. Параметри імпульсів збудження, що мінімізують помилку прогнозування, підбирають послідовно. Для забезпечення прийнятної якості мови при швидкості коду близько 10 Кб/с достатньо задати положення імпульсів з кроком близько 1 мс., і точністю амплітуд до 5 %.

Алгоритми класу гібридного стиснення мовного сигналу забезпечують найкращий коефіцієнт стиснення та прийнятну якість мови при низьких швидкостях передачі. Крім того, вони характеризуються високою

обчислювальною стійкістю та складністю виконання. Алгоритми гібридного класу стиснених мовного сигналу рекомендується використовують в спеціалізованих системах зв’язку та в системах зв’язку загального призначення.

3.3.4. Алгоритми стиснення звукових сигналів

У даному підрозділі розглянемо два класи алгоритмів стиснення звукових сигналів: з втратами та без втрат.

Стиснення звукових сигналів із втратами має надзвичайно широке застосування. Окрім комп'ютерних програм, стиснення з втратами використовується в потоковому аудіо, DVD, цифровому телебаченні, радіо та потоковому медіа, Інтернеті.

Особливістю алгоритмів цього класу є використання психоакустики для виявлення компонентів звучання, що не сприймаються слухом людини. Прикладом можуть слугувати високі частоти, які сприймаються лише при достатній їх потужності, або тихі звуки, що виникають одночасно після голосніших звуків і тому маскуються ними - такі компоненти звучання можуть бути передані менш точно, або і взагалі не передаватись. Для здійснення маскування сигнал із часової послідовності відліків амплітуди перетворюється на послідовність спектрів звуків, в яких кожен компонент спектру стискається окремо. Для здійснення такого перетворення використовуються методи швидкого перетворення Фур'є, модифікованого дискретно-косинусного перетворення, квадратурно-дзеркальних фільтрів або інші. Загальний обсяг інформації при такому перекодуванні лишається незмінним. Стиснення в певній частотній області може полягати в тому, що замасковані або нульові компоненти не запам'ятовуються взагалі, або кодуються з меншою кількістю біт. Наприклад, частотні компоненти до 200 Гц та понад 14 кГц можуть бути закодовані з 4-бітною розрядністю, тоді як компоненти в середньому діапазоні - 16 бітною розрядністю. Результатом такої операції є кодування із середньою розрядністю 8-біт, проте результат буде значно кращим ніж при кодуванні всього діапазону частот з 8-бітною розрядністю. Очевидно, що перекодовані з низькою роздільністю фрагменти спектру вже не можуть бути точно відновлені.

Головним параметром алгоритмів стиснення звукових сигналів з втратами є бітрейт, що визначає ступінь стиснення файлу та його якість. Найпоширенішими форматами стиснення з втратами є: AAC, ADPCM, ATRAC, Dolby AC-3, MP2, MP3, Musepack Ogg Vorbis, WMA та інші.

Складність стиснення звуку без втрат полягає в тому, що записи звуку є надзвичайно складними у своїй структурі. Одним з стандартних методів стиснення - пошук взірців і їх повторень - не ефективний для більш хаотичних даних, якими є оцифрований звук. Найпоширенішими форматами стиснення

без втрат є: Free Lossless Audio Codec (FLAC), Apple Lossless, MPEG-4 ALS, Monkey's Audio, TTA.

3.4. Алгоритми стиснення відео

Стиснення відео являє собою зменшення кількості даних, які використовуються для представлення відеопотоку. Алгоритми стиснення відео дозволяють ефективно зменшувати потік даних, необхідний для відтворення, передачі та зберігання відео.

У загальному випадку стиснення відео є компромісом між економією дискового простору, якістю відео, і вартістю апаратного забезпечення, необхідного для декомпресії відео в реальному часі.

Стиснення відео ґрунтується на двох важливих принципах. Перший – це просторова надлишковість, притаманна кожному кадру відеоряду. Другий принцип базується на тому факті, що більшу частину часу кожен кадр подібний до свого попередника (часова надлишковість). Таким чином, типовий алгоритм стиснення відео розпочинається з стиснення першого кадру з допомогою декількох алгоритмів стиснення зображення. Потім кодується кожен наступний кадр, знаходяться розбіжності між цим кадром і його попередником та стискається їх збіжність. Якщо новий кадр сильно відрізняється від попереднього, то його можна стискати незалежно від інших відео кадрів.

Виділяють алгоритми стиснення відео з втратами та без втрат. Стиснення відео без втрат виконується шляхом заміни оригінальної послідовності бітів іншою послідовністю, що містить опис оригінальної послідовності. При цьому скорочення відбувається за рахунок фрагментів, що повторюються, і фрагментів, що містять закономірний ряд змін. У разі потреби отримати початковий потік даних його відновлюють за потоком-описом. Проте використовуючи алгоритми стисненням без втрат неможливо досягти високих коефіцієнтів стиснення на реальному (не штучному) відео. З цієї причини майже все відео стискаються з використання алгоритмів стиснення з втратами.

Стиснення відео із втратами виконують за двома схемами:

1. У трансформувальних кодеках фрейми зображень трансформуються в новий базисний простір і виконується квантування. Трансформація може здійснюватися або для всього фрейму цілком (як, наприклад, у схемах на основі wavelet-перетворення), або поблоково (характерний приклад – JPEG). Результат потім стискається методами ентропії.

2. У кодеках, які використовують метод передбачення, попередні й наступні дані використовуються для того, щоб передбачити поточний семпл зображення. Помилка між передбаченими даними і реальними разом з додатковою інформацією квантується і кодується.

У деяких системах обидві схеми комбінуються через використання трансформуючих кодеків для стиснення помилкових сигналів, згенерованих на стадії передбачення.

Більшість сучасних алгоритмів стиснення відео використовують дискретно косинусне перетворення, вейвлет-перетворення або фрактальне стиснення для усунення просторової та часової надлишковості відео файлів.

3.4.1. Дискретне косинусне перетворення

На сьогодні майже всі алгоритми стиснення відео (наприклад, стандарти, прийняті ITU-T або ISO) використовують дискретне косинусне перетворення (ДКП) або його модифікації для усунення просторової надлишковості відео даних.

Найчастіше у при стисненні відео потоку використовується двовимірне ДКП, що послідовно застосовується до блоків зображення розмірністю 8×8 пікселів. Дискретне косинусне перетворення обчислює 64 (8×8=64) коефіцієнти, які потім квантуються, забезпечуючи тим самим реальне стиснення відео потоку. У більшості зображень значна частина коефіцієнтів ДКП через свою незначну вагу після квантування обнуляються. Цю властивість ДКП і покладено в основу алгоритмів стиснення, які використовують ДКП.

Дискретне косинусне перетворення реалізується множенням вектора на матрицю перетворення. При цьому матриця оберненого перетворення з точністю до множника дорівнює транспонованій матриці. Матриці вибирають таким чином, щоб перетворення було ортонормованим, а постійний множник дорівнював одиниці. В комп'ютерних програмах це не завжди так.

Змінні довжини сигналу приводять до необхідності використання різних типів ДКП. Нижче наведено матриці для перших чотирьох типів:

$$\text{DCT-1}_n = \left[\cos kl\tfrac{\pi}{n-1}\right]_{0\le k,l<n} \tag{3.5}$$

$$\text{DCT-2}_n = \left[\cos k(l+\tfrac{1}{2})\tfrac{\pi}{n}\right]_{0\le k,l<n} \tag{3.6}$$

$$\text{DCT-3}_n = \left[\cos(k+\tfrac{1}{2})l\tfrac{\pi}{n}\right]_{0\le k,l<n} \tag{3.7}$$

$$\text{DCT-4}_n = \left[\cos(k+\tfrac{1}{2})(l+\tfrac{1}{2})\tfrac{\pi}{n}\right]_{0\le k,l<n} \tag{3.8}$$

де *k, 1= 1,...,n; n* – кількість вибірок в блоці.

Перетворення 3.6 (двовимірне ДКП) найчастіше зустрічається в практичних застосуваннях завдяки властивості "ущільнення енергії".

Звісно, використання ДКП на блоці з *n* вибірок потребує n^2 операцій множення та сумування. Проте завдяки рекурсивній структурі матриці ДКП реально буде потрібна набагато менша кількість математичних операцій, а саме $n*log_2(n)$. Ця властивість робить ДКП практично незамінним при застосовнні на сучасних математичних процесорах персональних комп'ютерів.

ДКП використовується у таких форматах: JPEG, MPEG, MPEG-1, MPEG-2, MPEG-4, H.261, H.263, H.263+.

Основними недоліками алгоритмів стиснення з ДКП є:

- “блочність” за високої компресії;
- закруглення кутів та випадкове розмивання гострих країв зображення;
- висока апаратна сладність.

3.4.2. Алгоритм фрактального стиснення

Основа алгоритму фрактального стиснення - виявлення самоподібних ділянок у зображенні. У відповідності з даним алгоритмом зображення розбивається на множину непересічних рангових підзображень, що визначають множину доменних підзображень, які перекриваються. Для кожного рангового блоку алгоритм стиснення знаходить найоптимальніший доменний блок і афінне перетворення, яке переводить доменний блок в даний ранговий блок. Структура зображення відображається в систему рангових блоків, доменних блоків і перетворень.

Ідея полягає в наступному: припускають, що вихідне зображення є нерухомою точкою відображення, яке потрібно стиснути. Тоді можна замість самого зображення запам'ятати його відображення, для відновлення якого достатньо декілька разів застосувати це відображення до будь-якого стартового зображення.

По теоремі Банаха, такі ітерації завжди призводять до нерухомої точки, тобто до вихідного зображення. На практиці вся складність цього алгоритму полягає в знаходженні по зображенню найкращого відображення та зберігання його у компактному вигляді.

Алгоритми фрактального стисння є найбільш перспективними на сьогоднішній день, оскільки вони забезпечують коефіцієнт стиснення в межах 2-2000. Крім того, при розархівуванні зображення можна виконувати масштабування. Алгоритм орієнтований на повноколірні зображення та зображення в градаціях сірого кольору.

3.4.3. Дискретне вейвлет перетворення

Дискретне вейвлет перетворення (ДВП) – алгоритм, що ґрунтується на передаванні сигналу, наприклад зображення через пару фільтрів: низькочастотний і високочастотний. Низькочастотний фільтр видає грубу форму вихідного сигналу. Високочастотний фільтр видає сигнал різниці або додаткової деталізації. У свою чергу, результат на виході високочастотного фільтра (додатковий сигнал деталізації) може бути підданий тій же процедурі й так далі.

Простим прикладом ДВП є ДВП Хаара.

Вхідний сигнал x[n] є множиною вибірок з індексом n. Низькочастотний фільтр Хаара (Haar Low Pass Filter) являє собою арифметичне середнє двох вибірок вхідного сигналу *x[n]*:

$$g[n] = \frac{1}{2}\left(x[n] + x[n+1]\right) \tag{3.9}$$

Високочастотний фільтр Хаара є середньою різницею двох вибірок:

$$h[n] = \frac{1}{2}\left(x[n+1] - x[n]\right). \tag{3.10}$$

Зазначимо, що:

$$x[n] = g[n] - g[n]x[n+1] = g[n] + h[n]. \tag{3.11}$$

Вихідні послідовності *g[n]* і *h[n]* містять надлишкову інформацію. Таким чином, зрозуміло, що для відтворення вхідного сигналу *x[n]* досить узяти лише парні або лише непарні його вибірки. Як правило, беруться парні вибірки. Таким чином, вхідний сигнал *x[n]* включає в себе лише елементи з вибірок, наприклад парних:

$$g[0], g[2], g[4]....$$

$$h[0], h[2], h[4]....$$

$$x[0] = g[0] - h[0];\;\; x[1] = g[0] + h[0];$$

$$x[2] = g[2] - h[2];\;\; x[3] = g[2] + h[2] \text{ і т. д.} \tag{3.12}$$

Вихід низькочастотного фільтра є грубою аналогією вихідного сигналу. Якщо вихідним сигналом є зображення, то на виході низькочастотного фільтра буде розмите зображення з низькою роздільною здатністю. Вихід високочастотного сигналу додає деталі до зображення. У поєднанні з виходом низькочастотного фільтра може бути відтворене вхідне зображення. Грубу форму вихідного сигналу (сигнал на виході низькочастотного фільтра) інколи називають базовим рівнем (base layer), а додатковий сигнал деталізації –рівнем покращення (enhancement layer). Сигнал на виході високочастотного фільтра *h[n]* може бути пропущений знову через пару фільтрів, і процес таким чином може повторюватись до того часу, поки не буде досягнута достатня міра деталізації вихідного сигналу *x[n]*. Однак зрозуміло, що при такому підході щодо стиснення ефекту не буде. Перетворення просто відтворить таку саму кількість бітів, яка була у вихідному сигналі.

Вихідні значення називають коефіцієнтами перетворення, або коефіцієнтами wavelet-перетворення. Перетворення Хаара використовують здебільшого для стиснення зображень. Для інших цілей використовуються складніші фільтри перетворень. Стиснення досягається завдяки квантуванню

додаткового сигналу деталізації. Далі для отриманих коефіцієнтів перетворення використовують процедуру ймовірнісного кодування (ентропії).

Необхідно відмітити, що відносна якість зображень, стиснених з використанням ДВП, перевищує якість зображень, стиснених за допомогою ДКП, за одних і тих самих коефіцієнтів стиснення.

Алгоритми стиснення з ДВП мають такі недоліки:

- більшість статичних та динамічних зображень стиснених за допомогою алгоритму ДВП, не мають характерної для алгоритму ДКП блокової структури;
- відбувається заокруглення гострих контурів зображення, що породжує контурний шум або ефект Гіббса.

3.4.4. Алгоритм векторного квантування

Суть алгоритму, що базується на векторному квантуванні, полягає в розбитті зображення на блоки пік селів (розміром 4×4 пікселів у колірній схемі YUV для компресорів Indeo і Cinepak). Часто такі блоки виявляються схожими один на одного, що дає змогу компресору ідентифікувати клас подібних блоків і замінити їх одним загальним блоком. Крім того, генерується двійкова таблиця (карта) таких загальних блоків з найкоротших кодових слів. При декомпресії декодер, використовуючи таблицю, збирає зображення поблоково із загальних блоків. Алгоритм допускає апроксимацію реальних блоків зображення до загального блока, що їх об’єднує. Процес стиснення характеризується високою часовою та апаратною складністю, оскільки кодеру необхідно виявляти належність кожного блока зображення до будь-якого загального блока. Процес декодування зводиться до побудови зображення за заданою картою із загальних блоків та не вимагає багато обчислювальних ресурсів, що дозволяє його дуже швидко виконати. Таблицю або карту називають кодовою книгою, а двійкові коди, що входять у неї - кодовими словами.

Найкращий коефіцієнт стиснення відео потоку отримується завдяки зменшенні кількості класів загальних блоків, що дає змогу зменшити розмір кодової книги. По мірі зменшення розміру кодової книги якість відео погіршується.

Для прикладу порівняємо три блоки 4×4.

Блок 1:	Блок 2:	Блок 3:
128 128 128 128	128 127 128 128	128 127 126 128
128 128 128 128	128 128 128 128	128 128 128 128
128 128 128 128	128 128 127 128	127 128 128 128
128 128 128 128	128 128 128 128	128 128 128 128

Відмінності в наведених блоках для людського ока непомітні. Отже блоки 2 і 3 можна замінити першим. Тоді кодова книга матиме такий вигляд:

Кодова книга [1] = 128 128 128 128
128 128 128 128
128 128 128 128
128 128 128 128

Важливою особливістю алгоритму векторного квантування є те, що для стиснення декількож кадрів відео може використовуватись одна і таж кодова книга.

Недоліки векторного квантування:

- процес стиснення характеризується високою апаратною складністю і практично неможливий без спеціального обладнання;
- спостерігаються блокові спотворення за високих коефіцієнтів стиснення.

Контрольні запитання

1. Що Ви розумієте під стисненням даних?
2. Методи стиснення даних: з втратами та без втрат.
3. Стиснення даних з використанням алгоритму RLE. Наведіть приклад стиснення.
4. Стиснення даних з використанням алгоритму Хаффмана. Наведіть приклад стиснення.
5. Стиснення даних з використанням алгоритму LZW.
6. Охарактеризуйте алгоритм JPEG.
7. Класифікація алгоритмів стиснення аудіо.
8. Класифікація та приклади алгоритмів стиснення мовних сигналів.
9. Алгоритми стиснення звукових сигналів: особливості побудови.
10. Алгоритми стиснення мовного тракту людини.
11. Алгоритми стиснення параметрів мовного тракту людини.
12. Алгоритми гібридного стиснення мовних сигналів.
13. Особливості стиснення відео.
14. Охарактеризуйте алгоритм дискретного-косинусного перетворення.
15. Алгоритм фрактального стиснення.
16. Охарактеризуйте алгоритм дискретного вейвлет перетвореннння.
17. Охарактеризуйте алгоритм векторного квантування.

Використана література.

1. Chu W.C. Speech coding algorithms Foundation and Evolution of Standardized Coders / W.C. Chu. – New Jersey : John Wiley & Sons Inc, 2003. – 578 p.
2. Д. Ватолин, А. Ратушняк, М. Смирнов, В. Юкин. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. - Диалог-МИФИ, 2002. - С. 384.

3. Д. Сэломон. Сжатие данных, изображения и звука. - М.: Техносфера, 2004. - С. 368.
4. Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. Алгоритмы: построение и анализ. - 2-е изд. - М.: «Вильямс», 2006. - 1296 с.
5. Ананий В. Левитин. Глава 9. Жадные методы: Алгоритм Хаффмана // Алгоритмы: введение в разработку и анализ = Introduction to The Design and Analysis of Aigorithms. — М.: «Вильямс», 2006. - С. 392-398.
6. А. Мельник. Р. Шевчук. Порівняльний аналіз алгоритмів стиснення мовних сигналів // Вісник національного університету «Львівська політехніка» Комп'ютерні системи і мережі №523. – Львів, 2004. – С. 109 – 117.

РОЗДІЛ 4. ПРОГРАМНІ ІНТЕРФЕЙСИ ДЛЯ СТВОРЕННЯ МУЛЬТИМЕДІА ЗАСТОСУНКІВ

4.1. Графічна бібліотека OpenGL

Графічна бібліотека OpenGL (Open Graphics Library) - це базовий стандарт, що визначає незалежний від мови програмування крос-платформовий програмний інтерфейс (API) для написання застосунків, що використовують 2D та 3D комп'ютерну графіку. Даний інтерфейс містить понад 250 функцій, які можуть використовуватися для малювання складних тривимірних сцен з простих примітивів. Широко застосовується індустрією комп'ютерних ігор і віртуальної реальності, у графічних інтерфейсах (Compiz, Clutter), при візуалізації наукових даних, в системах автоматизованого проектування тощо. Фактично, бібліотека OpenGL являє собою два набори команд: визначення графічних об'єктів та керування їхнім відображенням у буфері екрана.

Графічний стандарт OpenGL розроблений і затверджений у 1992 році дев'ятьма фірмами, серед яких Digital Equipment Corporation, Hewlett-Packard Corporation, IBM Corporation, Silicon Corporation, Sun Microsystems Inc., Microsoft. В основу стандарту була покладена бібліотека IrisGL, розроблена Silicon Graphics.

Основними перевагами OpenGL є:

- стабільність - внесені в стандарт зміни попередньо анонсуються й реалізуються таким чином, щоб гарантувати нормальну роботу вже написаного програмного забезпечення;
- надійність – усі додатки, що використовують OpenGL, гарантують однаковий візуальний результат, що не залежить від апартного та програмного забезпечення персонального комп’ютера;
- мобільність – додатки, що використовують OpenGL, можуть запускатися на персональних комп'ютерах, робочих станціях чи суперкомп'ютерах;
- простота використання – OpenGL добре структурована. Її драйвери включають інформацію про базові графічні пристрої.

4.1.1. Система координат і проекції у OpenGL

У комп'ютерній графіці найчастіше для задання місця розташування тривимірного об'єкта використовується ортогональна тривимірна декартова система координат. Якщо позитивний напрямок осі Z спрямовано вбік спостерігача, то така система називається правосторонньою, в іншому випадку – лівосторонньою (рисунок 4.1).

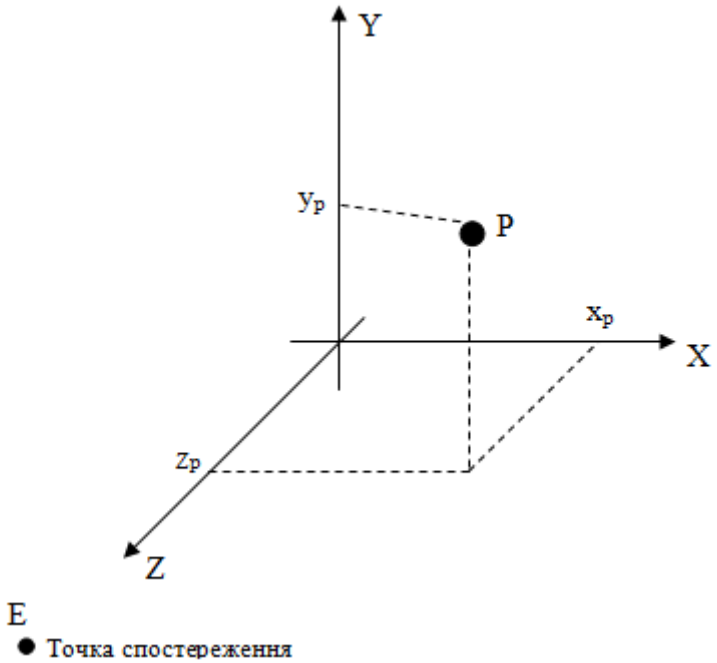

Рисунок 4.1 – Правостороння декартова система координат

Бібліотека OpenGL використовує три види систем координат:

- глобальну;
- видову;
- екранну.

Глобальна система координат – це правостороння тривимірна декартова система координат. Видова система координат – це лівостороння система координат. Екранна система координат – це плоска декартова система координат із природнім напрямом осей (вісь Y спрямована нагору, вісь X – вправо).

На рисунку 4.2 зображені системи координат, які використовуються бібліотекою OpenGL.

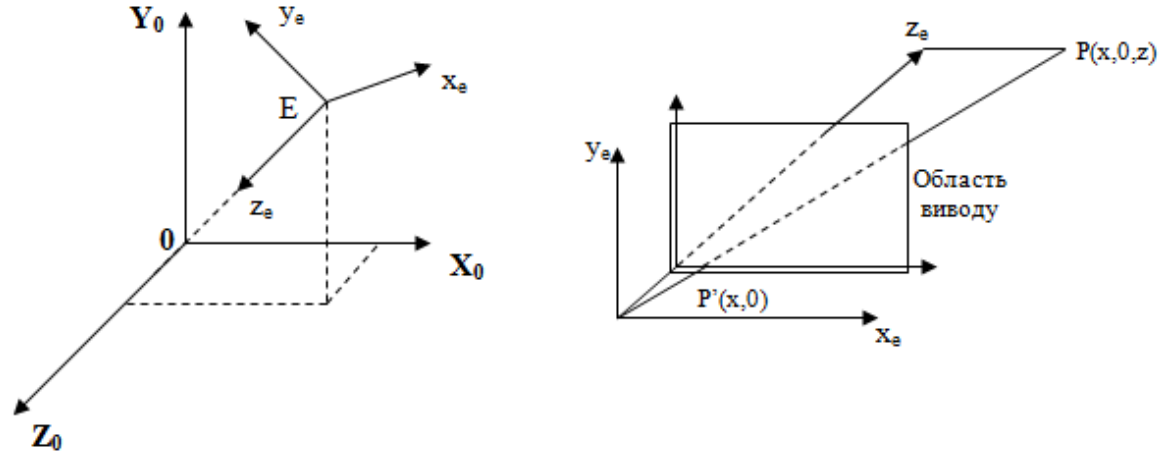

Рисунок 4.2 – Системи координат OpenGL

При цьому глобальна система координат з’єднується з положенням об'єкта чи сцени, видова – із положенням спостерігача, причому початок її координат – точка E збігається з оком спостерігача. І, нарешті, екранна система координат сполучається з плоским екраном, на якому спостерігач розглядає проекцію сцени.

Механізм перетворення видових координат в екранні, реалізований у OpenGL, наведено на рисунку 4.3.

Рисунок 4.3 – Процес виведення тривимірної графіки в OpenGL

У процесі виведення тривимірної графічної інформації задається видимість об’єкта у глобальному просторі, його проекція на графічну площину та область виведення на екран.

У 3D-графіці всі об'єкти описуються в просторі трьома координатами, однак спостереження за ними ведеться через екран монітора, що є плоским (двовимірним). Ця невідповідність усувається шляхом введення проекцій, що відображають об'єкти на двовимірній графічній проекційній площині.

У залежності від відстані між спостерігачем і графічною площиною розрізняють два основних класи проекцій:

- рівнобіжна, якщо відстань між спостерігачем і графічною площиною нескінченна. При визначенні рівнобіжної проекції вказується напрям проектування (рисунок 4.4);
- центральна, якщо відстань між спостерігачем і графічною площиною кінцева. При описі такої проекції явно задається центр проекції (рисунок 4.5).

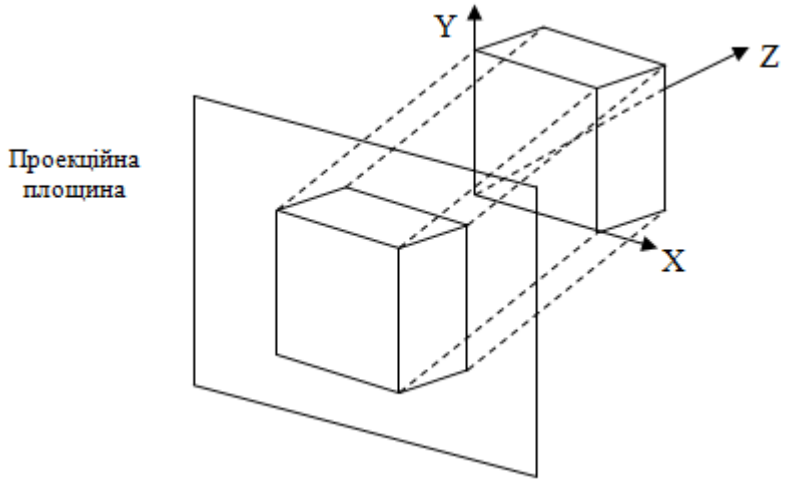

Рисунок 4.4 – Рівнобіжна проекція

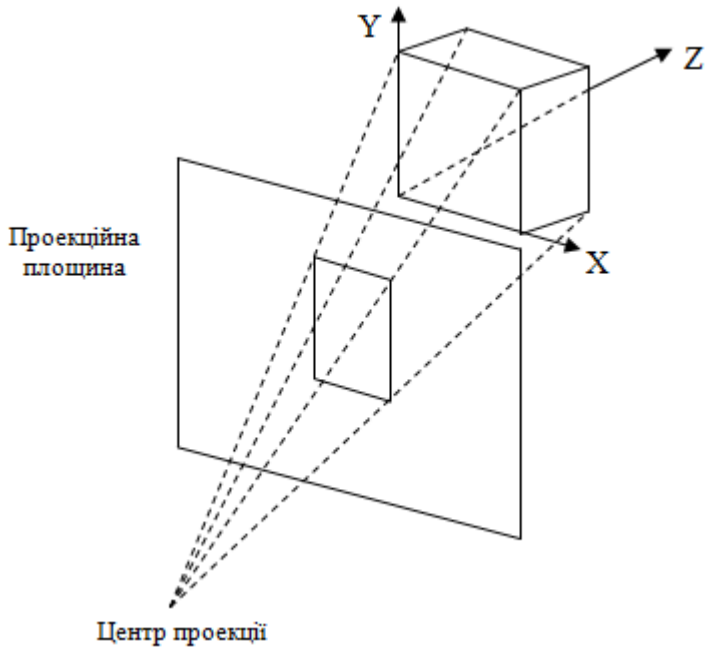

Рисунок 4.4 – Центральна проекція

Існують наступні види проектування:

- ортографічний – об'єкт, що відображається, проектується під кутом 90^0 з дотриманням точних співвідношень висоти та ширини;
- аксонометричний - об'єкт, що відображається, належить до умовної прямокутної (ортогональної) просторової системи координат, осі якої паралельні основним розмірам зображуваного об'єкта. Основою аксонометричного зображення предмета є аксонометричні координати його характерних точок і аксонометричний масштаб;
- перспективний – всі точки об’єкта, що відображається, проектуються на площину проекцій променями, що проходять через одну точку, яка називається центром проектування. Даний вид проектування застосовується при спостереженні за всією сценою та

найбільш наближений до дійсності, тому що перспектива є невід'ємною частиною людського світосприйняття.

Таким чином, усі дані про об'єкт комп'ютер зберігає у формі наборів з 3 координат, а на екрані відображається лише його плоска проекція. Для створення реалістичних образів необхідно враховувати вплив перспективи. У силу цього бібліотека OpenGL для задання вершин використовує однорідні координати. В однорідних координатах положення точки P(x,y,z) записується як P(W(x, W(y, W(z, W) або P(x, y, z, W), де W – масштабний множник. При W = 1 виходить звичайна декартова система координат.

4.1.2. Синтаксис команд OpenGL

Командами в OpenGL називаються функції чи процедури. Багато еквівалентні по діях команди можуть розрізнятися по типах і кількості аргументів. Для опису таких команд введено наступний синтаксис:

rtype CommandName [1 2 3 4] [b s i f d ub us ui] [v](atype arg).

Команди OpenGL представляються чотирма параметрами – ім'ям та трьома символами:

- *rtype* визначає тип значення, що повертається та вказується в явному виді для кожної команди;
- *CommandName* – ім'я команди, таке, наприклад, як glRotate
- *[1234]* – цифра, що показує число аргументів команди;
- *[b s i f d ub us ui]* – символи, що визначають тип аргументу;
- *[v]* – буква, що показує, що як аргумент використовується покажчик на масив значень;
- *atype i ard* визначаються типом та числом аргументів відповідно.

Допускаються наступні типи аргументів (таблиця 4.1).

Таблиця 4.1

Типи аргументів OpenGL

Символ	Тип OpenGL*	Тип C(C++)
b	GLbute	Char
s	GLshort	Short
i	GLint	Int
f	GLfloat	Float
d	GLdouble	Double
ub	GLubute	unsigned byte
us	GLushort	unsigned short
ui	GLuint	unsigned int

Наприклад, команда визначення кольору glColor* може мати одну з нижче перерахованих форм запису:

```
glColor3d(1,1,1);
glColor3f(0.5,0.5,0.5);
glColor4b(1,1,1,0).
```

4.1.3. Ініціалізація OpenGL

Для реалізації бібліотеки OpenGL в операційних системах Windows фірма Microsoft використовує спеціальну структуру PIXELFORMATDESCRIPTOR, що використовується для задання формату пікселів:

```
typedef struct tagPIXELFORMATDESCRIPTOR
{
   WORD  nSize;
   WORD  nVersion;
   DWORD dwFlags;
   BYTE  iPixelType;
   BYTE  cColorBits;
   BYTE  cRedBits;
   BYTE  cRedShift;
   BYTE  cGreenBits;
   BYTE  cGreenShift;
   BYTE  cBlueBits;
   BYTE  cBlueShift;
   BYTE  cAlphaBits;
   BYTE  cAlphaShift;
   BYTE  cAccumBits;
   BYTE  cAccumRedBits;
   BYTE  cAccumGreenBits;
   BYTE  cAccumBlueBits;
   BYTE  cAccumAlphaBits;
   BYTE  cDepthBits;
   BYTE  cStencilBits;
   BYTE  cAuxBuffers;
   BYTE  iLayerType;
   BYTE  bReserved;
   DWORD dwLayerMask;
   DWORD dwVisibleMask;
   DWORD dwDamageMask;
} PIXELFORMATDESCRIPTOR;
```

Для ініціалізації бібліотеки необхідно заповнити наступні поля:

- nSize - розмір структури, встановлюється оператором sizeof(PIXELFORMATDESCRIPTOR);

- nVersion - номер версії бібліотеки. Це значення має бути рівним одиниці;
- dwFlags - набір бітових прапорів, що визначають властивості пікселів буфера. Для більшості додатків оптимальним значенням буде наступна логічна комбінація: PFD_DRAW_TO_WINDOW | PFD_SUPPORT_OPENGL | PFD_DOUBLEBUFFER;
- iPixelType - режим відображення кольорів. Найчастіше використовується режим, що задається значенням PFD_TYPE_RGBA – кожен колір визначається чотирма значеннями: інтенсивністю червоного, зеленого, синього і ступенем прозорості (альфа);
- cColorBits - визначає число бітових площин кольору в кожному колірному буфері;
- cDepthBits - розмір буфера глибини (вісь Z). Наприклад, 32.

Інші поля даної структури використовуються для отримання спеціальних ефектів і по замовчувані повинні бути заповнені нулями.

Для роботи зі структурою PIXELFORMATDESCRIPTOR у Win32 API реалізовано цілий ряд функцій. Найбільш важливими з них є ChoosePixelFormat і SetPixelFormat.

int ChoosePixelFormat(HDC hdc, CONST PIXELFORMATDESCRIPTOR ppfd);*

Ця функція перевіряє, чи підтримує заданий формат пікселів (ppfd), пристрій, який задається дескриптором (hdc). Якщо пристрій не підтримує такий формат пікселів, то функція поверне значення 0. В іншому випадку функція поверне індекс формату пікселів, що найбільше повно задовольняє заданим параметрам.

Після того як знайдено необхідний формат пікселів, його потрібно встановити на пристрої.

bool SetPixelFormat(HDC hdc, int iPixelFormat, CONST PIXELFORMATDESCRIPTOR ppfd);*

Другий параметр задає індекс формату пікселів (отриманий з виклику попередньої функції). У випадку помилки повертається значення false.

Після того як необхідний формат пікселів встановлений на заданому пристрої, необхідно створити контекст відтворення і зробити його активним. Контекст відтворення OpenGL (rendering context) – це спеціальна змінна типу HGLRC, що використовується для зв'язку OpenGL і віконних систем сімейства Windows (GDI).

Для роботи з контекстом відтворення в Win32 API реалізовано функції:

- wglCreateContext - створює контекст відтворення для заданого пристрою (hdc). У випадку помилки (наприклад, не заданий формат пікселів на поточному пристрої) буде повернуте значення NULL. Приклад: HGLRC wglCreateContext(HDC hdc).
- wglMakeCurrent – встановлює поточний контекст відтворення (hglrc) на заданому пристрої виведення (hdc). В один момент часу тільки один контекст

відтворення може бути активним. У випадку помилки, повертається значення false. Приклад: bool wglMakeCurrent(HDC hdc, HGLRC hglrc).

- wglDeleteContext - видаляє заданий контекст відтворення (hglrc). Її використовують у тому випадку, якщо контекст відтворення більше не потрібно (наприклад, при закритті вікна). Приклад: bool wglDeleteContext(HGLRC hglrc).

Таким чином, для ініціалізації графічної бібліотеки OpenGL необхідно виконати наступні стандартні дії:

1. Визначити необхідний формат пікселів і перевірити його сумісність із заданим пристроєм (наприклад, вікном виводу).
2. Створити контекст відтворення.
3. Зробити контекст відтворення активним.

При завершенні роботи з OpenGL необхідно виконати наступні стандартні дії:

1. Зробити контекст відтворення не активним.
2. Видалити контекст відтворення.

Приклад: Ініціалізація OpenGL

```
// Оголошення та ініціалізація формату пікселів
static PIXELFORMATDESCRIPTOR pfd=        // pfd повідомляє Windows яким буде виведений піксель
 {
  sizeof(PIXELFORMATDESCRIPTOR),         // розмір дескриптора даного формату пікселів
  1,                // Номер версії
  PFD_DRAW_TO_WINDOW |           // Формат для вікна
  PFD_SUPPORT_OPENGL |           // Формат для OpenGL
  PFD_DOUBLEBUFFER,              // Формат для подвійного буфера
  PFD_TYPE_RGBA,                 // Необхідний RGBA формат
  bits,                 // Вибирається біт глибини кольору
  0, 0, 0, 0, 0, 0,             // Ігнорування кольоровичх бітів
  0,                // Відсутній буфер прозорості
  0,                // Зсунутий біт ігнорується
  0,                // Відсутній буфер накопичення
  0, 0, 0, 0,                   // Біти накопичення ігноруються
  32,                // 32 бітний Z-буфер (буфер глибини)
  0,                // Відсутній буфер трафарета
  0,                // Відсутні додаткові буфери
  PFD_MAIN_PLANE,                // Головний шар рисування
  0,                // Зарезервовано
  0, 0, 0                // Маски шарів ігноруються
 };
  int                           PixelFormat;
  HGLRC                         hrc;

  // Перевірка сумісності формату пікселів і пристрою (у даному випадку – форми TForm)
  PixelFormat = ChoosePixelFormat(Canvas->Handle, &pfd);
```

```
// Встановлення формату пікселів
SetPixelFormat(Canvas->Handle, PixelFormat, &pfd);
// Створення контексту відтворення
hrc = wglCreateContext(hdc);
// Встановлення контексту в активний стан
wglMakeCurrent(Canvas->Handle, hrc);
```

Приклад: Завершення роботи з OpenGL

```
wglMakeCurrent(NULL, NULL);    // Відключення контексту відтворення
wglMakeCurrent(Canvas->Handle, hrc);    // Видалення контексту
```

При програмуванні з використанням C++ Builder для графічного виводу на формі (клас TForm бібліотеки візуальних компонентів VCL) із використанням OpenGL необхідно спочатку підключити відповідний файл-заголовок gl.h. Робиться це директивою препроцесора #include <gl.h>.

Для програмування процесу малювання на формі необхідно перевизначити оброблювачі наступних подій Windows: WM_CREATE, WM_DESTROY, WM_PAINT і WM_RESIZE. Для цього в інспекторі об'єктів C++ Builder необхідно в закладці Events знайти й перевизначити оброблювачі відповідних подій класу TForm – OnCreate, OnDestroy, OnPaint і OnResize.

Таким чином, загальну структуру додатка, що використовує OpenGL, можна представити в такий спосіб (рисунок 4.5).

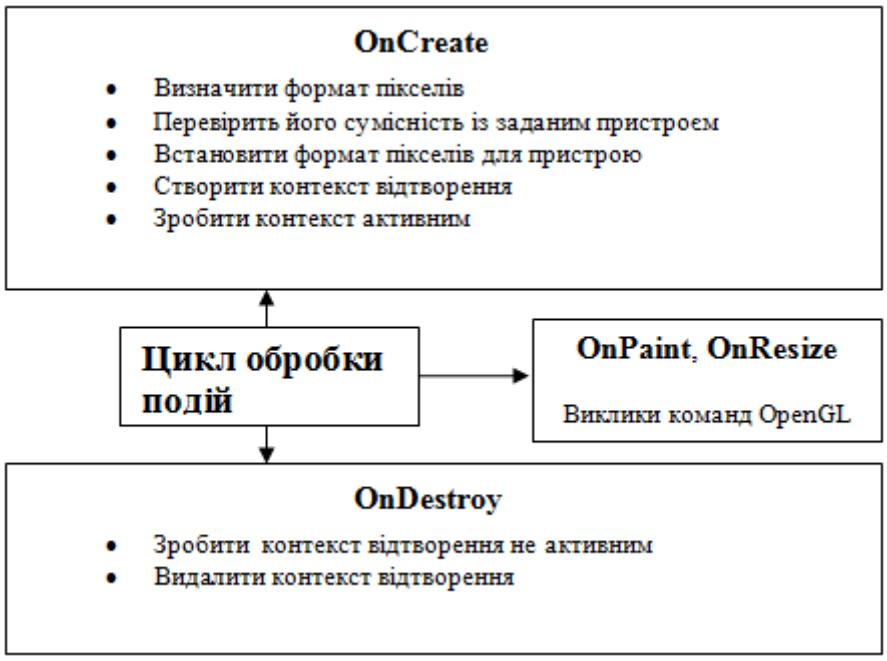

Рисунок 4.5 – Структура додатка, що використовує OpenGL

4.1.4. Матриці перетворення координат

Усі перетворення над координатами бібліотека OpenGL робить із використанням відповідних матриць перетворення (розмірністю 4×4).

Передбачено три типи матриць – виду, проекцій і текстури. Перш ніж почати роботу з кожною з них, її необхідно зробити активною:

void glMatrixMode(Glenum mode) // робить поточною задану параметром mode матрицю.

Параметр mode може приймати одне з наступних значень:

- GL_MODELVIEW – робить активною матрию виду;
- GL_PROJECTION - робить активною матрию проекцій;
- GL_TEXTURE - робить активною матрию текстури.

Наприклад, команда glMatrixMode(GL_MODELVIEW) робить активною видову матрицю перетворення.

Перш, ніж використовувати матрицю, її потрібно ініціалізувати:

void glLoadIdentity(void) // замінює поточну матрицю перетворення на одиничну (яку називають матрицею ідентичності).

При виконанні різних перетворень часто виникає необхідність збереження значень поточної матриці з можливістю її відновленням. Для цього слугують наступні команди:

void glPushMatrix(void); // запам'ятовує матрицю у стеці
void glPopMatrix(void); // *відновлення значення поточної матриці*

Бібліотека OpenGL дає змогу виконувати наступні видові перетворення:

- обертання (glRotate*);
- перенос (glTranslate*);
- масштабування (glScale*).

void glRotate[f d](GLtype angle, GLtype x, GLtype y, GLtype z) // здійснює поворот вектора проти часової стрілки на кут angle (у градусах) щодо точки (x,y,z).

Наприклад, після виконання команди glRotatef(45,0,0,1) усі об'єкти будуть зображені повернеными на кут 45^0 щодо осі Z.

void glTranslate[f d](GLtype x, GLtype y, GLtype z) // здійснює перенесення об'єкта на відстані x,y і z вздовж осей X,Y і Z відповідно.

Наприклад, команда glTranslatef(1,0,0) зрушує об'єкт на відстань 1 уздовж осі абсцис.

void glScale[f d](GLtype x, GLtype y, GLtype z) // здійснює масштабування об'єкта вздовж кожної з координатних осей на значення, які задані відповідними параметрами.

4.1.5. Введення та виведення проекцій

Область виведення проекцій визначається як прямокутник із заданими координатами верхнього лівого кута, а також шириною та висотою у пікселях. По замовчуванні, область виводу OpenGL збігається з клієнтської (робочої) частиною вікна, у яке виводиться зображення. Для її задання використовується спеціальна команда glViewport:

void glViewport(GLint x, Glint y, Glint width, Glint height) // Ця команда задає перетворення з нормалізованих координат у віконні. Причому, x і y – задають координати верхнього лівого кута області виводу, а width і height – відповідно її висоту й ширину.

Наприклад, команда glVievport(0,0, ClientWidth / 2, ClientHeight / 2) задає ліву половину клієнтської частини вікна як область виведення.

Після того як встановлена область виведення, потрібно встановити проекційну (картинну) площину й обсяг видимості. Для різних типів проекції це здійснюється по-різному.

Для задання ортографічної проекції необхідно викоти наступні дії:

- здати області виведення;
- встановити активною та ініціалізувати матрицю проекцій;
- встановити матриці перспективи та обсяг видимості.

Для виконання останнього етапу використовується команда

void glOrtho(Gldouble left, Gldouble right, Gldouble bottom, Gldouble top, Gldouble near, Gldouble far)// параметри left і right визначають координати лівої і правий вертикальних, а bottom і top – нижньої й верхній горизонтальних площин відсікання. Відстань до ближньої і дальньої площин відсікання визначається параметрами near і far.

Приклад: Завання ортографічної проекції

```
// Визначення області виведення
glViewport(0,0, ClientWidth, ClientHeight)
// Встановлення активною матрицю проекції
glMatrixMode(GL_PROJECTION);
// Її ініціалізація
glLoadIdentity();
glOrtho(-10, 10, -10, 10, 20, 50);
// Встановлення активною матрицю виду
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
// Поворот на 45 градусів навколо осі Y
glRotatef(45,0,1,0);
```

Встановлення перспективної проекції аналогічне ортографічній за винятком обсягу видимості – для цього типу проектування визначається усічений конус видимості. У OpenGL реалізовано спеціальну команду для створення матриці перспективи:

void glFrustum(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble znear, GLdouble zfar), // Параметри left і right визначають координати лівої і правий вертикальних, a bottom і top – нижньої і верхній горизонтальних площин відсікання. Відстань до ближньої і дальньої площин відсікання по глибині визначається параметрами znear і zfar (які повинні завжди бути позитивними).

Вважається, що спостерігач знаходиться в точці з координатами (0,0,0). Точність представлення глибини кольру залежить від значень, визначених у znear і zfar.

Приклад: Задання перспективної проекції
glViewport(0,0, ClientWidth, ClientHeight) // Визначення області виводу
glMatrixMode(GL_PROJECTION); // Встановлення активною матрицю проекції
glLoadIdentity();// Її ініціалізація
glFrustum(-2, 2, -2, 2, 2, 50); // Встановлення матриці виду активною
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();

Оскільки перспективна проекція використовується досить часто, у бібліотеку OpenGL реалізовано команди gluPerspective і gluLookAt, що полегшують її задання.

void gluPerspective(Gldouble angley, Gldouble aspect, Gldouble znear, Gldouble zfar) // визначає усічений конус видимості у видовій системі координат. Параметр angley задає кут видимості (у градусах) у напрямку осі Y. У напрямку X кут видимості задається через відношення aspect, що визначається співвідношенням сторін області виведення. Параметри znear, zfar визначають відстань від спостерігача до ближньої і дальньої площин відсікання.

void gluLookAt(GLdouble eyex, GLdouble eyey, GLdouble eyez, GLdouble centerx, GLdouble centery, GLdouble centerz, GLdouble upx, GLdouble upy, GLdouble upz); // створює матрицю виду, що залежить від точки спостереження (дозволяє в явному виді задати положення спостерігача сцени). Параметри eyex, eyey, eyez задають координати точки спостереження, centerx, centery, centerz – центр сцени, upx, upy, upz – вектор "нагору", що визначають позитивний напрямок осі Y сцени.

Приклад 5. Задання перспективної проекції і точки спостереження
glViewport(0,0, ClientWidth, ClientHeight); // Визначення області виводу
glMatrixMode(GL_PROJECTION);
glLoadIdentity();

gluPerspective(40.0, ClientWidth / ClientHeight, 0.1, 25); // Задання конуса видимості
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0, 0, 3, 0, 0, 0, 0, 1, 0); // Задання точки спостереження і центра сцени

Як правило, задання області виведення і виду проекції здійснюється при зображенні сцени один раз, тому вищенаведені приклади задання області видимості і способу проектування зазвичай містяться в оброблювачі події WM_CREATE – OnCreate класу TForm.

4.1.6. Колір у OpenGL

У OpenGL використовується колірна модель RGB. Однак для встановлення конкретного кольору допускається використання четвертого компонента –альфи, що визначає ступінь прозорості кольору. Така колірна схема називається RGBA. Причому альфа може змінюватися від 0.0 (повна прозорість) до 1.0 (повна непрозорість, значення за замовчуванням).

Для роботи в режимі RGBA вирисовується наступна команда:

void glColor[3 4][b s i f d][v](GLtype components) //

Наприклад, для встановлення жовтого кольору необхідно виконати команду glColor3b(1,1,0).

Після того як колір встановлено, він розповсюджується на всі об’єкти, що створюється. При малюванні суцільних примітивів (наприклад, багатокутників) допускається використання плавного сполучення кольорів між різнобарвними вершинами. Даний режим включається командою glEnable(GL_SMOOTH).

4.1.7. Графічні примітиви OpenGL

Примітиви – це базові елементи з яких будуються графічні об'єкти. До примітивів у OpenGL відносяться точки, лінії, багатокутники та бітові масиви.

Для задання примітиву в OpenGL передбачено спеціальні командні дужки glBegin – glEnd, між якими міститься група команд, що задає примітив:

void glBegin(GLenum mode);
void glEnd(void);

Параметр mode визначає тип примітива, що буде будуватися з описаних між командними дужками вершин.

Таблиця 4.2

Типи примітивів OpenGL

Значення mode	Опис
GL_POINTS	Кожна вершина розглядається як окрема точка
GL_LINES	Кожна пара вершин розглядається як незалежний відрізок. Остання непарна вершина ігнорується.
GL_LINE_STRIP	Малюється зв'язна послідовність відрізків. Кінець попереднього відрізка є початком наступного.
GL_LINE_LOOP	Аналогічно попередньому за винятком того, що остання вершина з'єднується з першою (замкнута послідовність відрізків)
GL_TRIANGLES	Кожна трійка вершин розглядається як незалежний трикутник. Зайві вершини (одна чи дві ігноруються).
GL_TRIANGLE_STRIP	Малюється група зв'язних трикутників, що мають загальну грань.
GL_TRIANGLE_FAN	Малюється група зв'язних трикутників. Перші три вершини – перший трикутник. Перша, третя і четверта – другий і т.д.
GL_QUADS	Кожна група з чотирьох вершин розглядається як незалежний чотирикутник. Зайві вершини ігноруються.
GL_QUAD_STRIP	Малюється група зв'язних чотирикутників (аналогічно GL_TRIANGLE_STRIP)
GL_POLYGON	Малюється окремий опуклий багатокутник, заданий усіма вершинами.

Як вже відзначалося раніше, вершина – це точка в тривимірному просторі. Для її визначення в бібліотеці реалізована спеціальна команда glVertex:

void glVertex[234] [v] (type coord)

Виклик будь-якої команди glVertex визначається чотирма аргументами: x,y,z,w, що задають однорідні координати вершини. При цьому дотримуються наступні умови:

- glVertex2 – задає координати x і y, z = 0,w = 1;
- glVertex3 – задає координати x , y, z , а w = 1;
- glVertex4 – задає всі чотири координати x , y, z і w.

Для задання товщини точки в OpenGL використовується команда:

void glPointSize(GLfloat size) // size задає значення товщини (за замовчуванням приймається 1.0).

Приклад: Зображення червоної точки в просторі товщиною 2.5

```
glPointSize(2.5);
glColor3b(1,0,0);
glBegin(GL_POINTS);
  glVertex3d(1,1,1);
```

```
glEnd();
```

Виведення ліній не складніше ніж малювання точок. Для задання ширини ліній використовується команда:

```
void glLineWidth(GLfloat width);
```

Приклад: Зображення двох не зв'язаних між собою відрізків різного кольору

```
glLineWidth(2.0);
glBegin(GL_LINES);
  glColor3f(0.5,0.5,1);
  glVertex2f(-1,0);
  glVertex2f( 1,0);
  glColor3f(1,0.5,0);
  glVertex2f(0,-1);
  glVertex2f(0,1);
glEnd();
```

Для демонстрації використання багатокутних примітивів розглянемо наступні приклади:

Приклад: Зображення різнобарвного трикутника

```
glEnable(GL_SMOOTH); // Встановлюєм режим плавного сполучення кольорів
// Задаємо координати вершин і їх кольорів
glBegin(GL_TRIANGLES);
  glColor3b(1,0,0);
  glVertex2f(0,0);
  glColor3b(0,1,0);
  glVertex2f( 1,0);
  glColor3b(0,0,1);
  glVertex2f(1,1);
glEnd();
```

Приклад: Зображення двох білих суміжних чотирикутників

```
glBegin(GL_QUAD_STRIP);
  glVertex3f(0,0,0);
  glVertex3f( 1,0,0);
  glVertex3f(1,1,0);
  glVertex3f(0,1,0);
  glVertex3f(1,1,1);
  glVertex3f(0,1,1);
glEnd();
```

Приклад: Зображення сірого п'ятикутника

```
glColor3f(0.5,0.5,0.5);
glBegin(GL_POLYGON);
```

```
    glVertex3f(0,0,0);
    glVertex3f( 1,0,0);
    glVertex3f(1,1,0);
    glVertex3f(0,1,0);
    glVertex3f(-1,0,0);
  glEnd();
```

Для зображення растрових примітивів використовується команда:

*void glDrawPixels(GLsizei width, GLsizei height, GLenum format, GLenum type, const GLvoid *pixels); // параметри width і height задають висоту і ширину бітового масиву в пікселях, format – визначає спосіб збереження бітів у масиві (GL_RGBA – колірна модель), type – тип елементів бітового масиву (наприклад, GL_UNSIGNED_BYTE) і pixels – вказує адресу, починаючи з якої, зберігається бітовий образ.*

При відображенні растровий примітив виводиться відносно поточної позиції растра, що задається командою

void glRasterPos[234] [v] (type coord) // coord – координати початкової позиції растра.

Приклад: Зображення растрового примітива

```
// Задання елементів бітового масиву
GLubyte bmp[] = {
                  0x00, 0x00, 0x00, 0x00,
                  0x00, 0xFF, 0xFF, 0x00,
                  0x00, 0xFF, 0xFF, 0x00,
                  0x00, 0x00, 0x00, 0x00,
                };

    // Задання початкової позиції растра
glRasterPos2i(0,0);
glDrawPixels(4,4,GL_RGBA,GL_UNSIGNED_BYTE, bmp); // Виведення растрового примітива
```

Бібліотека OpenGL містить ряд додаткових квадратичних примітивів, таких як циліндр, сфера, диск, сектор та інші (gluCylinder, gluSphere, gluDisk, gluPartialDisk, gluNewQuadric).

Перш ніж використовувати такі примітиви, їх потрібно створити за допомогою команди

GLUquadricObj gluNewQuadric(void), // повертає вказівник на квадратичний примітив. Потім вказівник використовується для створення безпосередньо конкретної фігури.*

Приклад: Зображення сфери
```
// Оголошення вказівника  на квадратичний примітив
GLUquadricObj* quadObj;
quadObj = gluNewQuadric();// Створення примітива
// Зображення сфери радіусом 1.5, що складається з 16 меридіанів і 16 паралелей
gluSphere (quadObj, 1.5, 16, 16);
gluDeleteQuadric(quadObj); // Видалення об'єкта й очищення пам'яті
```

Робота з іншими квадратичними об'єктами ведеться аналогічно роботі зі сферою.

4.2. Програмний інтерфейс DirectX

DirectX - це набір API функцій, розроблених для простого і ефективного вирішення завдань пов'язаних з ігровим та відеопрограмуванням під Microsoft Windows.

Практично всі частини DirectX API є наборами COM-сумісних об'єктів.

В цілому, DirectX підрозділяється на:

- DirectX Graphics (DirectDraw: інтерфейс виведення растрової графіки; Direct3D (D3D): інтерфейс виведення тривимірних примітивів).
- DirectInput - інтерфейс для обробки даних, що поступають з клавіатури, миші, джойстика, ігрових контролерів, тощо.
- DirectPlay - інтерфейс мережевої комунікації ігор.
- DirectSound - інтерфейс низькорівневої роботи із звуком (формату Wave).
- DirectMusic - інтерфейс відтворення музики у форматах Microsoft.
- DirectShow - інтерфейс для вводу/виводу аудіо- і/або відео- даних.
- DirectSetup - частина, що відповідає за встановлення DirectX.
- DirectX Media Objects - реалізує функціональну підтримку потокових об'єктів (наприклад, кодери/декодери).

4.2.1. Основи DirectX Graphics

4.2.1.1. Теорія малювання тривимірних зображень в DirectX Graphics

Вся графічна система DirectX базується на тривимірному інтерфейсі виведення тривимірних об’єктів - Direct3D. Тому все, що робиться з DirectX Graphics, викладено в термінології тривимірної графіки.

У цьому підрозділі викладено термінологію тривимірної графіки і теорією малювання тривимірних зображень.

Все, що малюється за допомогою Direct3D складається з полігонів (polygon) - це зазвичай трикутна фігура, утворена трьома точками (вершинами). Вершина (vertex) - це мінімальна одиниця в тривимірній графіці; єдина точка (з координатами) розташована у двомірному або тривимірному просторі. Об’єднання двох вершин створюють ребра (лінії), а три ребра утворюють полігон.

Три з'єднаних ребра утворюють грань (polygon face), а весь малюнок називається сіткою (mesh) або моделлю (сітка - це результат об'єднання всіх базових об'єктів: вершин, ребер і полігонів). Взаємозв'язок між вершинами, ребрами, полігонами і сітками показано на рисунку 4.6.

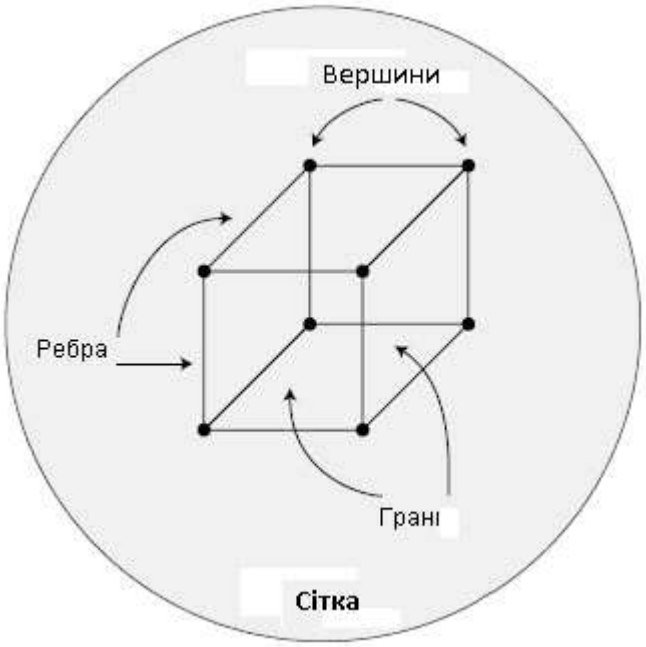

Рисунок 4.6 - З'єднання точок для створення тривимірного об'єкта

Щоб представлення об'єктів виглядало більш реалістичним, Direct3D використовує матеріали (materials) для зафарбовування порожніх полігонів сітки зазначеним кольором. Матеріали подаються у вигляді комбінації колірних компонент, а також необов'язкових растрових зображень – текстур (texture), які в процесі візуалізації наносяться на поверхню полігону.

Системи координат. Система координат, що використовується у Direct3D зображена на рисунку 4.7.

Координата X направлена зліва направо від розташованого в крайній лівій позиції початку координат. Координата Y направлена зверху вниз від розташованого на самому верху початку координат. Координати збільшуються в позитивному напрямку до іншого кінця їх проміжків. У Direct3D двомірні координати зазвичай називають перетвореними координатами (transformed coordinates) тому що вони представляють собою кінцеві координати, що використовуються для малювання об'єкта на екрані.

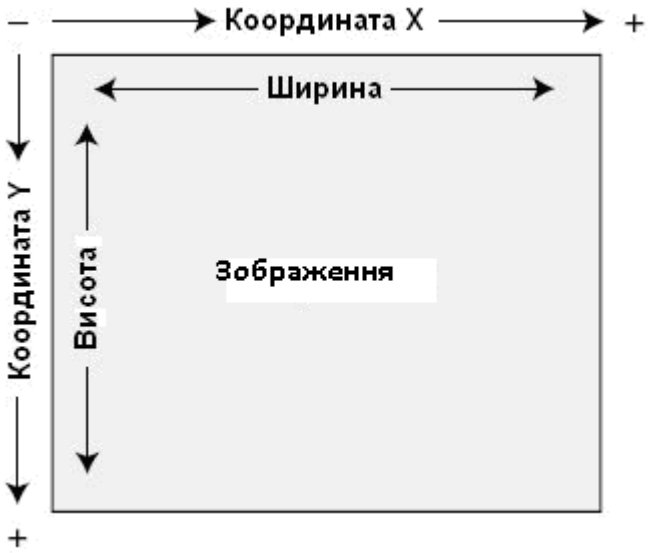

Рисунок 4.7 – Система координат, що використовується у Direct3D

У тривимірному просторі додається додаткова координата -координата Z (рисунок 4.8).

Рисунок 4.8 – Тривимірна система координат

Зазвичай координата Z представляє глибину зображення. Координата Z може використовуватися двома способами: коли позитивний напрям слідує від спостерігача або, коли позитивний напрямок слідує до спостерігача. Ці два варіанти називаються лівосторонньої (left-handed) і правосторонньою (right-handed) координатними системами.

У тривимірному просторі все вимірюється в таких координатних системах - двомірної для зображень та екрану і тривимірної для всього іншого. Отже, якщо потрібно визначити (за допомогою тривимірних координат) точку

простору, що знаходиться перед вами (по осі Z), трохи правіше (по осі X) і на рівні очей (по осі Y), ви можете задати для неї координати X = 100, Y = 50, Z = 200. Ці координати представляють точку, що знаходиться в 100 одиницях праворуч від вас, в 50 одиницях над землею і в 200 одиницях перед вами.

У випадку двомірних координат ви можете сказати, що картина на стіні має 200 пікселів в ширину і 200 пікселів у висоту. Центр цієї картини буде в точці X = 100, Y = 100, а верхній лівий кут - у точці X = 0, Y = 0. Тривимірні координати називають неперетвореними координатами (untransformed coordinates), оскільки вони не представляють кінцевих координат. З іншого боку, двовимірні координати називають перетвореними координатами (transformed coordinates), оскільки вони безпосередньо відображаються в координати точок екрана.

Розробка об'єктів. Розробляти об'єкти, такі як сітки і моделі (і навіть плоскі двомірні зображення), необхідно починати з рівня вершин. У кожної вершини є призначені їй координати X, Y і Z. Ви можете задати ці координати трьома способами: в екранному просторі (screen space) використовуючи перетворені координати, в просторі моделі (model space) використовуючи неперетворенні координати, і в глобальному просторі (world space) також використовуючи неперетворенні координати.

Екранний простір використовується для того, щоб відобразити вершини на реальні координати точок екрана. Простір моделі (локальний простір - local space) посилається на координати, виміряні відповідно до довільної точки відліку, якою часто є центр моделі. Вершини в локальному просторі належать моделі, і програміст можете переміщати їх, надаючи об'єкту необхідну форму. Перед візуалізацією об'єкту точки перетворюються з локального простору в глобальний. При візуалізації об'єкту координати глобального простору перетворюються в координати екранного простору.

Вершини, розміщені в глобальному просторі визначають кінцеве місце розташування, що використовується при візуалізації об'єкта. Глобальний простір - це позиція фіксованої точки в тривимірному просторі. Для прикладу розглянемо в якості сітки людину. Суглоби людини - це вершини, визначені в локальному просторі, оскільки їхні координати задані щодо центру вашої грудної клітини. Коли людина переміщується по будинку (у глобальному просторі), координати суглобів змінюються у просторі, але відносно центру вашого тіла залишаються постійними, як це показано на рисунку 4.9.

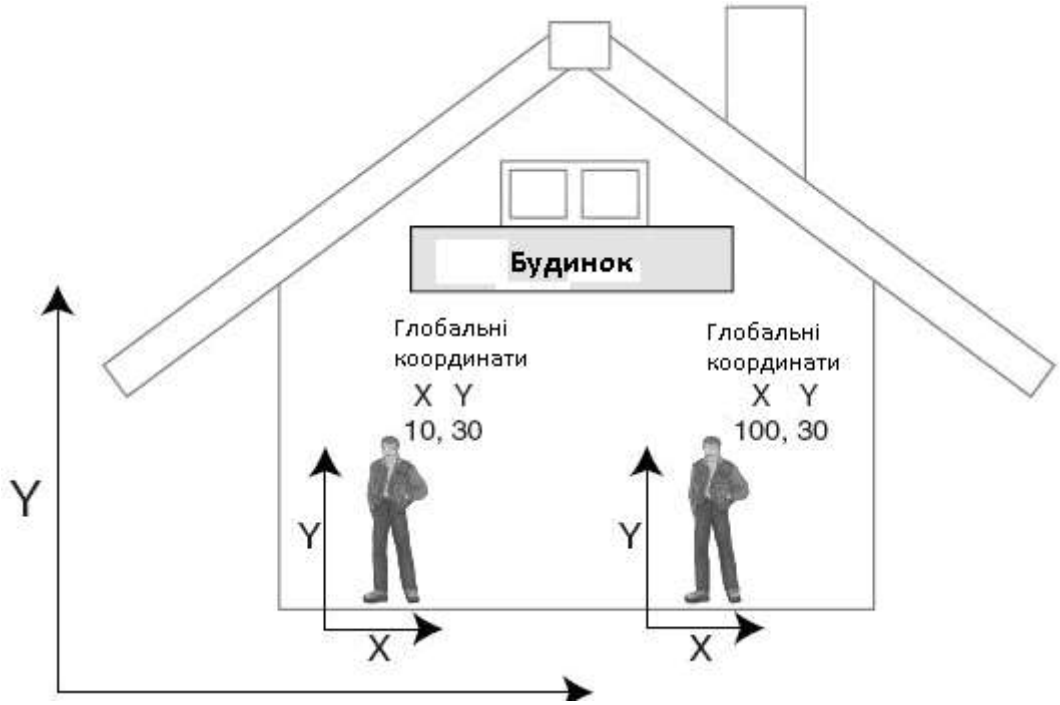

Рисунок 4.9 – Приклад глобальних координат

Після вибору типу координат для малювання об'єкта (в екранному, локальному або глобальному просторі), розміщуються вершини (нумерація здійснюється у порядку їх розміщення). Потім ці вершини з’єднуються в групи по три для створення трикутних граней. На рисунку 4.10 показано набір полігональних граней створених шляхом групування вершин.

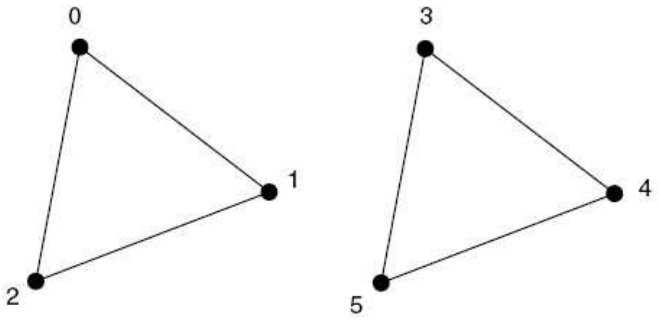

Рисунок 4.10 – Набір полігональних граней

Списки, смуги та віяла. При створенні полігональних граней обов'язково має врахуватись спільне використання вершин. Набір полігональних граней може бути в одній з трьох категорій: списки трикутників, смуги трикутників і віяла трикутників.

Список трикутників (triangle list) - це набір граней, що не мають спільних вершин, тому в кожного полігону своє тріо вершин. Смуга трикутників (triangle strip) - це набір граней з загальними вершинами в якому у кожного полігону загальне ребро з іншим. Віяло трикутників (triangle fan) виходить в

тому випадку, якщо кілька граней спільно використовують одну й ту ж вершину. Ці три категорії показані на рисунку 4.11.

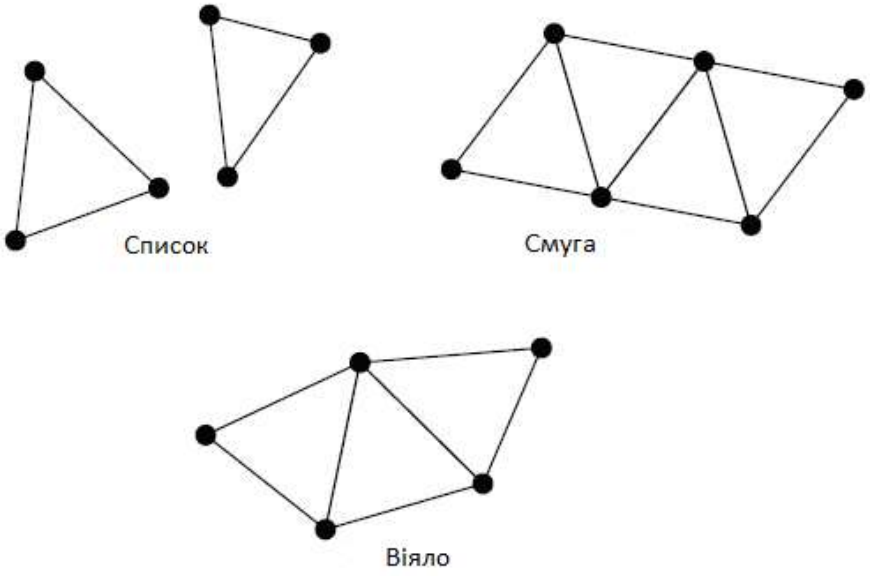

Рисунок 4.11 – Приклад спільного використання вершин

У списку трикутників, на відміну від смуг і віяла трикутників, вершини спільно не використовуються. Застосування смуг і віяла скорочує кількість вершин, що заощаджує пам'ять і збільшує швидкість візуалізації

Для візуалізації точок і ліній використовуються відповідно одна і дві вершини.

Порядок вершин. При візуалізації полігонів, дуже важливим є порядок, який використовувався при визначенні вершин, тому що необхідно визначити, яка сторона грані є лицьова, а яка - зворотною. Вершини, що визначаються грань прийнято розміщувати за годинниковою стрілкою (рисунок 4.12).

У системах тривимірної графіки грані оберненні до спостерігача зворотною стороною зазвичай не малюються і ігноруються в процесі візуалізації. Цей процес називається відкиданням зворотних граней (backface culling) і є одним з основних способів оптимізації. У правосторонній системі координат все виглядає з точністю до навпаки, і лицьовою стороною доспостерігача звернені ті грані, у яких вершини йдуть проти годинникової стрілки.

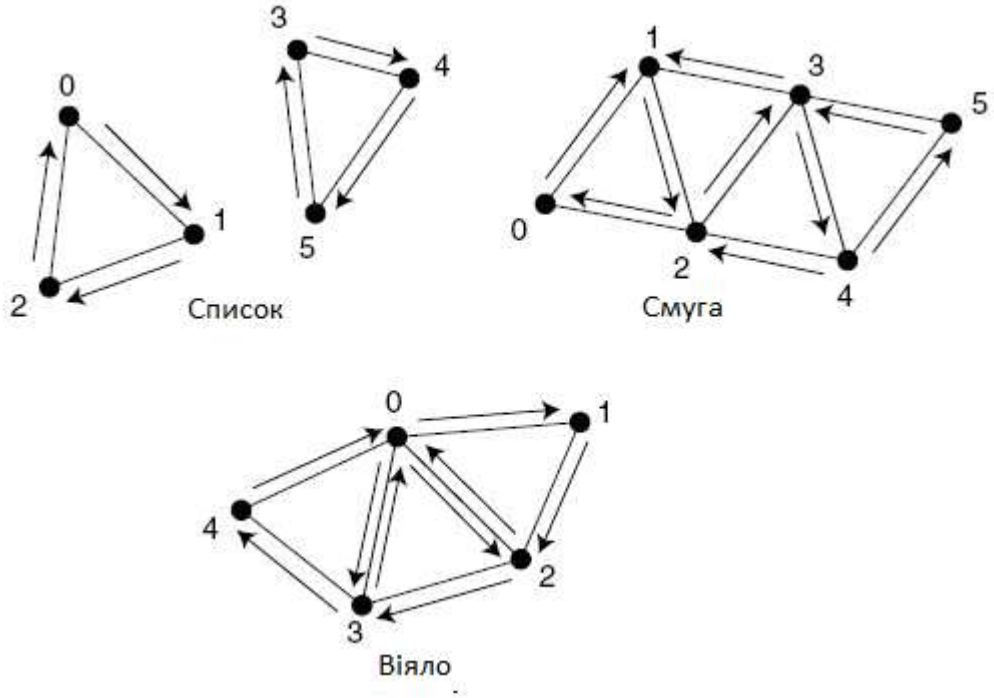

Рисунок 4.12 – Особливості розміщення вершин граней

Фарбування полігонів. У Direct3D використовується дві схеми розфарбовування полігонів. Перша схема використовує визначення матеріалів, які є простими кольорами. Для цих матеріалів задаються їх розсіювана, фонова та відображена колірні компоненти. Розсіюванний (diffuse) і фоновий (ambient) колір зазвичай один і той же колір - колір забарвлення об'єкту. Колір відбивача (specular) - це колір відблисків, що з'являються на об'єкті, якщо освітити його близько розташованим джерелом світла.

Друга схема - накладення текстур (texture mapping), розфарбовує полігони використовуючи зображення.Сама текстура є зображенням, що завантажуються з файлу із растровою графікою. Це растрове зображення натягується або накладається повторюваної мозаїкою на поверхню полігону.

Перетворення. Після визначення моделі (або набіру полігонів), можна розмістити її в потрібному місці простору. На рисунку 4.13 показано кілька моделей розміщених у трьохвимірному просторі.

Спостерігач можете переміщати, масштабувати та обертати будь-який об'єкт у потрібному напрямі використовуючи одну модель для малювання декількох об'єктів з різною орієнтацією.

Такі дії, як переміщення, обертання і масштабування називаються перетвореннями. Для того, щоб перенести об'єкт з його простору моделі в готову до перегляду систему координат необхідний набір перетворень.

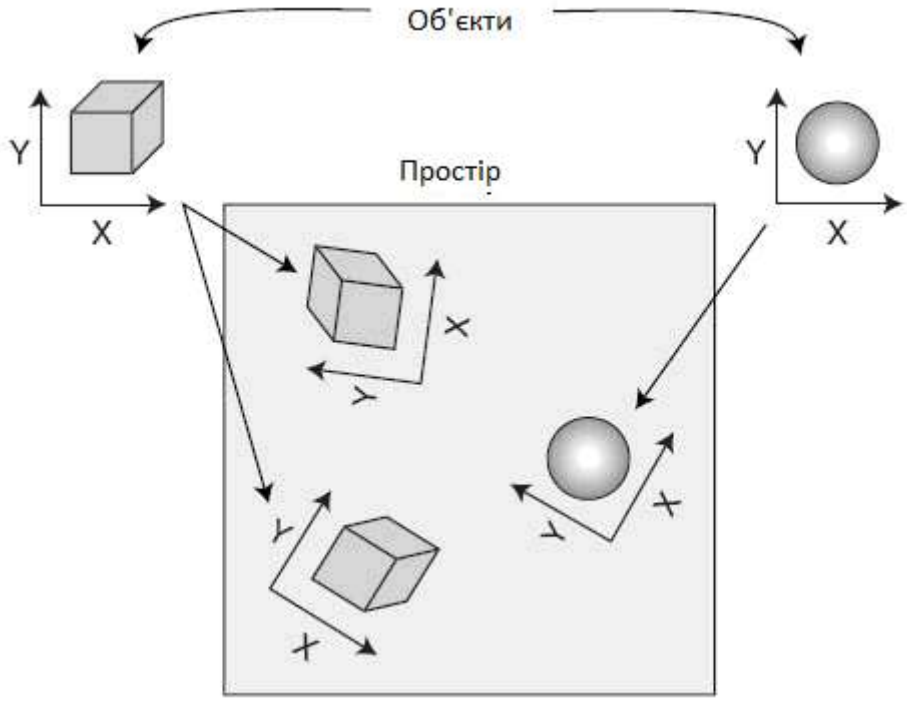

Рисунок 4.13 – Розміщення об’єктів в тривимірному просторі

Перше з них, глобальне перетворення (world transformation), застосовується для перетворення об'єкта з його локальних коодинат у глобальні координати з використанням масштабування, обертання щодо осей X, Y і Z, і переміщення. Наступне перетворення – це перетворення виду (view transformation), яке орієнтує всі об'єкти відносно точки перегляду в тривимірному світі, тобто перетворює глобальні координати в координати виду.

Перетворення виду можна сприймати, як літаючу камеру у вашому тривимірному просторі. Перетворення виду переміщає об'єкти простору в ситему координат, центром якої є камера.

Остання перетворення – це перетворення проекції (projection transformation), яке перетворює тривимірний простір у його двомірне зображення. Це перетворення можна асоціювати з об'єктивом камери із змінною фокусною відстанню, що настроюються кутами огляду і різними ефектами.

4.2.1.2. Робота з матрицями у Direct3D

Створення матриць. У Direct3D матриці можуть бути будь-якого розміру, однак для кращого розуміння принципів роботи з матрицями розглянемо матриці розмірністю 4 × 4, тобто такі у яких чотири рядки і чотири стовпці. Direct3D зберігає матриці у вигляді структури D3DMATRIX:

```
typedef struct _D3DMATRIX {
  D3DVALUE _11, _12, _13, _14;
  D3DVALUE _21, _22, _23, _24;
```

```
    D3DVALUE _31, _32, _33, _34;
    D3DVALUE _41, _42, _43, _44;
} D3DMATRIX;
```

Для заповнення матриці перетвореннями можна використати бібліотеку D3DX у якій для цих цілей зарезервовано об'єкт D3DXMATRIX.

Кожне перетворення, що використовується має власну матрицю, за винятком обертання для якого потрібно три матриці (по одній для кожної осі). Це означає, що потрібно п'ять матриць перетвореннь: обертання відносно осі X, обертання відносно осі Y, обертання відносно осі Z, переміщення і масштабування. Наведемо набір функцій, що використовується для ініціалізації матриць обертання:

```
D3DXMATRIX *D3DXMatrixRotationX(
    D3DXMATRIX *pOut, // результуюча матриця
    FLOAT Angle);     // кут повороту навколо осі X
D3DXMATRIX *D3DXMatrixRotationY(
    D3DXMATRIX *pOut, // результуюча матриця
    FLOAT Angle);     // кут повороту навколо осі Y
D3DXMATRIX *D3DXMatrixRotationZ(
    D3DXMATRIX *pOut, // результуюча матриця
    FLOAT Angle);     // кут повороту навколо осі Z
```

Передавши кожній з перерахованих вище функцій порожню матрицю і значення кута повороту (у радіанах, що представляє кут повороту відносно відповідної осі), ми отримаємо матрицю, заповнену необхідними для роботи значеннями.

Наступна функція створює матрицю переміщення, яка застосовується для пересування об'єктів:

```
D3DXMATRIX *D3DXMatrixTranslation(
    D3DXMATRIX *pOut, // результуюча матриця
    FLOAT x,          // координата по осі X
    FLOAT y,          // координата по осі Y
    FLOAT z);         // координата по осі Z
```

Координати - це фактичні зміщення об'єкту відносно початку системи координат. Переміщення використовується для перетворення об'єкту з координат його локального простору у глобальні координати. Тепер представляємо функцію, яка масштабує об'єкт по його осях:

```
D3DXMATRIX *D3DXMatrixScaling(
    D3DXMATRIX *pOut, // результуюча матриця
    FLOAT sx,         // масштаб по X
    FLOAT sy,         // масштаб по Y
    FLOAT sz);        // масштаб по Z
```

Звичайний масштаб об'єкту дорівнює 1.0. Щоб збільшити розмір об'єкту в два рази потрібно задати значення 2.0, а щоб зробити об'єкт наполовину менше

його оригінального розміру, задається значення 0.5. Часто при роботі з матрицями застосовується спеціальний тип матриці - одинична матриця (identity matrix). Одинична матриця дуже корисна коли комбінуються дві матриці, але не змінюються їх початкові значення.

Для створення одиничної матриці використовують наступну функцію:

```
D3DXMATRIX *D3DXMatrixIdentity(D3DXMATRIX *pOut);
```

Наведемо декілька прикладів використання функцій:

```
D3DXMATRIX matXRot, matYRot, matZRot;
D3DXMATRIX matTrans, matScale;
// Задаємо поворот на 45 градусів (.785 радіан)
D3DXMatrixRotationX(&matXRot, 0.785f);
D3DXMatrixRotationY(&matYRot, 0.785f);
D3DXMatrixRotationZ(&matZRot, 0.785f);
// Задаємо переміщення в точку 100,200,300
D3DXMatrixTranslation(&matTrans, 100.0f, 200.0f, 300.0f);
// Збільшуємо розмір об'єкту у два рази по усіх осях
D3DXMatrixScaling(&matScale, 2.0f, 2.0f, 2.0f);
```

Комбінування матриць. Після заповнення різних матриць значеннями для перетворень, їх можна застосувати до кожної вершини. Фактично, можна зробити ще простіше, скомбінувати окремі матриці, що містять значення для переміщення, обертання і масштабування в єдину матрицю, перемноживши їх між собою. Ця процедура називається конкатенацією матриць, і є основою оптимізації усіх матричних обчислень.

Створивши єдину матрицю для кадру можна використовувати її для кожної вершини у сцені. Застосування цієї матриці до вершини дає аналогічний ефект послідовному застосуванню окремих матриць. Комбінувати матриці можна за допомогою функції D3DXMatrixMultiply:

```
D3DXMATRIX *D3DXMatrixMultiply(
  D3DXMATRIX      *pOut, // Результуюча матриця
  CONST D3DXMATRIX *pM1,  // Початкова матриця 1
  CONST D3DXMATRIX *pM2); // Початкова матриця 2
```

Передавши дві матриці в параметрах pM1 і pM2 отримуємо результуючу матрицю pOut, обчислену як результат множення перших двох матриць. Зупинимося на прикладі матриць переміщення, масштабування і обертання, створених в попередньому підрозділі, і скомбінуємо їх разом в одну матрицю, що представляє всі перетворення.

```
D3DXMATRIX matResult; // Результуюча матриця
// Очищаємо результуючу матрицю, роблячи її одиничною
D3DXMatrixIdentity(&matResult);
// Множимо на матрицю масштабування
D3DXMatrixMultiply(&matResult, &matResult, &matScale);
```

```
// Множимо на матриці обертання
D3DXMatrixMultiply(&matResult, &matResult, &matXRot);
D3DXMatrixMultiply(&matResult, &matResult, &matYRot);
D3DXMatrixMultiply(&matResult, &matResult, &matZRot);
// Множимо на матрицю переміщення
D3DXMatrixMultiply(&matResult, &matResult, &matTrans);
```

Інший метод перемножування матриць між собою - використання оператора множення (*) об'єкту D3DXMATRIX:

```
matResult = matXRot * matYRot * matZRot * matTrans;
```

Порядок комбінування матриць дуже важливий. У попередньому прикладі комбінування матриць здійснювалось в наступному порядку: масштабування, поворот навколо осі X, поворот навколо осі Y, поворот навколо осі Z і переміщення. Комбінування матриці в будь-якому іншому порядку може привести до небажаних результатів.

Перехід від локальних координат до координат виду. Для того, щоб вершина могла використовуватися для візуалізації грані, вона має бути перетворена з локальних координат (неперетворених координат) у глобальні координати. Глобальні координати потрібно перетворити в координати виду, а їх у свою чергу спроектувати в двомірні координати.

Локальні координати перетворюються у глобальні за допомогою глобальної матриці. Ця матриця містить перетворення, необхідні для розміщення об'єкту у тривимірному просторі. Друга матриця перетворення, яка використовується для перетворення об'єкту в координати виду, називається матрицею виду. Остання, матриця проекції, використовується для перетворення тривимірних координат з координат виду в перетворену вершину, яка застосовується для візуалізації графіки.

Створюючи глобальну матрицю і матрицю виду необхідно приділити увагу порядку в якому відбувається комбінування окремих матриць. Для глобального перетворення окремі матриці перетворень комбінуються в наступному порядку:

$$R = S * X * Y * Z * T$$

де R - це результуюча матриця, S - це матриця масштабування, X - матриця обертання навколо осі X, Y - матриця обертання відносно осі Y, Z - матриця обертання відносно осі Z, T - матриця переміщення.

У матриці виду окремі матриці перетворення повинні комбінуватися в наступному порядку:

$$R = T * X * Y * Z$$

На рисунку 4.14 показано шлях вершини через різні перетворення до завершального набору координат для малювання.

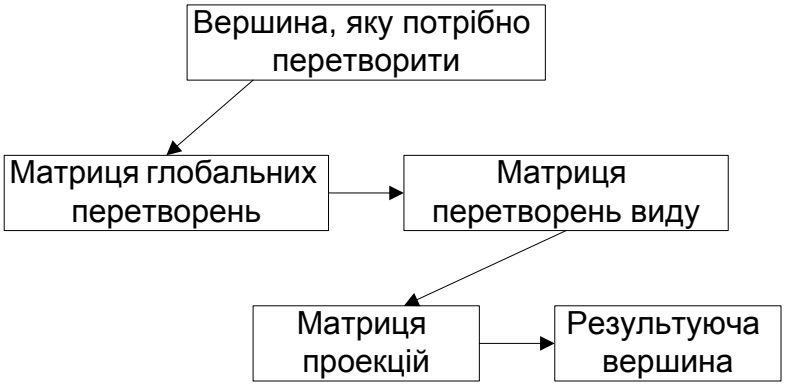

Рисунок 4.14 – Перетворення вершини за допомогою відповідних матриць

4.2.1.3. Особливості малювання об'єктів

Налаштування формату вершин. Direct3D дає змогу визначати вершини різними способами. Для створення довільних структур даних вершини виконується налаштування формату вершин (flexible vertex format, FVF), у результаті якого визначаються дані необхідні для опису вершин (тривимірні координати, двомірні координати, колір…).

Як тільки структура FVF створена, програміст оголошує дескриптор FVF (FVF descriptor), який є комбінацією прапорців та вершин.

Приведений нижче фрагмент коду містить структуру даних вершини:

```
typedef struct {
   FLOAT    x, y, z, rhw; // Двомірні координати
   FLOAT    x, y, z;      // Тривимірні координати
   FLOAT    nx, ny, nz;   // Нормаль
   D3DCOLOR diffuse;      // Розсіюваний колір
   FLOAT    u, v;         // Координати текстури
} sVertex;
```

Нормаль (normal) - це набір координат, що визначає напрям, який може використовуватися тільки спільно з тривимірними координатами. Як видно з прикладу, єдиним конфліктуючим елементо є координати, тому необхідно вибрати, який набір координат (двомірні або тривимірні) залишити.

Єдиною реальною відмінністю між двомірними і тривимірними координатами є додавання змінної rhw, яка є аналогом однорідного W. Це значення зазвичай представляє відстань від точки перегляду до вершини вздовж осі Z. В більшості випадків змінній rhw присвоюється значення 1.0.

Структура sVertex використовує дані типу FLOAT (числа з плаваючою точкою).

D3DCOLOR - це значення типу DWORD, яке застосовується в Direct3D для зберігання значень кольору. Щоб створити значення кольору типу D3DCOLOR можна використати одну з двох функцій: D3DCOLOR_RGBA або D3DCOLOR_COLORVALUE :

```
D3DCOLOR D3DCOLOR_RGBA(Red, Green, Blue, Alpha);
```

D3DCOLOR D3DCOLOR_COLORVALUE(Red, Green, Blue, Alpha);

Кожна функція отримує чотири параметри, кожен з яких задає величину окремої колірної складової, включаючи значення альфа-компоненти (прозорість). Для функції D3DCOLOR_RGBA ці значення мають знаходитися в діапазоні від 0 до 255, а для функції D3DCOLOR_COLORVALUE - в діапазоні від 0.0 до 1.0. Якщо використовуються непрозорі кольори, то для альфа-компоненти задається значення 255 (або 1.0).

Як приклад припустимо, що у структуру даних вершини необхідно включити тільки тривимірні координати і розсіювану складову кольору:

```
typedef struct {
  FLOAT    x, y, z;
  D3DCOLOR diffuse;
} sVertex;
```

Наступним етапом конструювання FVF є створення дескриптора FVF з використанням комбінації прапорців, перерахованих в таблиці 4.3.

Таблиця 4.3

Прапорці дескриптора формату вершин, що налаштовуються

Прапорець	Опис
D3DFVF_XYZ	Включені тривимірні координати
D3DFVF_XYZRHW	Включені двомірні координати
D3DFVF_NORMAL	Включена нормаль (вектор)
D3DFVF_DIFFUSE	Включена розсіювана складова кольору
D3DFVF_TEX1	Включений один набір координат текстури

Щоб описати дескриптор FVF необхідно комбінувати відповідні прапорці (мається на увазі, що потрібно використовувати тривимірні координати і розсіювану складову кольору):

```
#define VertexFVF (D3DFVF_XYZ | D3DFVF_DIFFUSE)
```

Використання буфера вершин. Після створення структури даних вершини та дескриптора, створюється об'єкт, який містить масив вершин. Direct3D для цього надає два об'єкти: IDirect3DVertexBuffer9 і IDirect3DIndexBuffer9.

При роботі із списками трикутників об'єкт IDirect3DVertexBuffer9 зберігає як мінімум по три вершини для кожного мальованого полігону (вершини впорядковані за годинниковою стрілкою). У смуги трикутників для малювання першого полігону використовується три вершини, а для малювання кожного наступного полігону використовується одна додаткова вершина. Для віяла трикутників задається одна центральна вершина, а для малювання кожного полігону зберігаються дві додаткові вершини.

Полігон може використовувати одну, дві або три вершини, залежно від того, що саме малюється. Для точок потрібно одну вершину, для ліній - дві, а для трикутників - три вершини.

На рисунку 4.15 показано особливоті зберігання та упорядкування вершин.

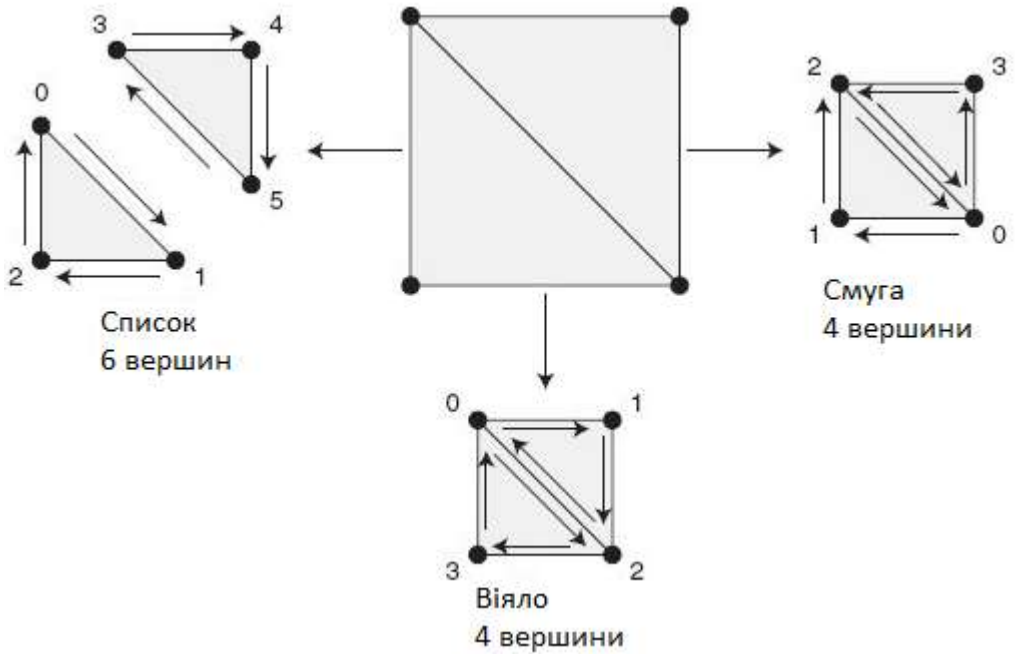

Рисунок 4.15 – Особливості зберігання та упорядкування вершин

На рис. 4.15 показано квадрат, представлений трьома різними способами. Перший спосіб - використання списку трикутників, який вимагає для представлення квадрата шість вершин, - по три вершини для кожного з двох трикутників. Другий спосіб - використання смуги трикутників з використанням тільки чотирьох вершин. Перші три вершини утворюють першу грань, а остання вершина використовується для формування другої грані. Для третього способу впорядковування, віяло трикутників, також використовується чотири вершини. Проте, у випадку з віялом, перша вершина задає основу віяла, а наступні вершини утворюють грані.

Створення буферу вершин. Для створення буферу вершин, використовується об'єкт IDirect3DDevice9, який ініціалізується наступним чином:

```
HRESULT IDirect3DDevice9::CreateVertexBuffer(
  UINT Length,        // кількість вершин, помножена на  розмір структури даних вершини
  DWORD Usage,     // 0
  DWORD FVF,       // дескриптор FVF
  D3DPOOL Pool,    // D3DPOOL_MANAGED
  IDirect3DVertexBuffer **ppVertexBuffer, // буфер вершин
```

```
   HANDLE *pHandle); // вкажіть NULL
```

У функції CreateVertexBuffer є тільки один параметр - прапорець Usage, який повідомляє Direct3D як використовуватиметься пам'ять, призначена для зберігання вершин.

Наведемо невеликий приклад створення буфера вершин, що містить чотири вершини (побудований на основі створеного раніше формату вершин, який містить тільки тривимірні координати і розсіювану складову кольору:

```
// g_pD3DDevice = об'єкт пристрою
// sVertex      = структура даних вершини
// VertexFVF    = дескриптор FVF
IDirect3DVertexBuffer9 *pD3DVB = NULL;
// Створення буфера вершин
if(FAILED(g_pD3DDevice ->CreateVertexBuffer(sizeof(sVertex) * 4
                    D3DCREATE_WRITEONLY, VertexFVF
                    D3DPOOL_MANAGED, &pD3DVB, NULL))) {
   // Виникла помилка
}
```

При створенні буфера вершин необхідно вказати відповідний розмір буфера (перший параметр у функції CreateVertexBuffer).

Блокування буфера вершин. Перед додаванням вершин до об'єкту буфера вершин, необхідно заблокувати пам'ять, яку використовує буфер. Це гарантує, що пам'ять використана для зберігання цих вершин буде доступна. Потім використовується покажчик на область пам'яті для доступу до пам'яті буфера вершин. Блокуючи пам'ять буфера вершин можна отримати покажчик на неї за допомогою функції методу Lock об'єкту буфера:

```
HRESULT IDirect3DVertexBuffer9::Lock(
   UINT  OffsetToLock, // зміщення до початку області, що блокується     UINT
SizeToLock,  // скільки байт блокувати, 0 = всі
   VOID  **ppbData,    // покажчик на покажчик (для доступу до даних)
   DWORD Flags);       // 0
```

Програміст вказує зміщення від початку буфера до тієї області до якої хочете отримати доступ (у байтах), а також розмір області, що блокується, в байтах (0, якщо блокується весь буфер).

Наведемо приклад функції, яка блокує весь буфер вершин:

```
// pD3DVB = об'єкт буфера вершин
BYTE *Ptr;
// Блокування пам’яті буфера вершин і отримання покажчика на неї
if(FAILED(pD3DVB ->Lock(0, 0, (void**)&Ptr, 0))){
   // Виникла помилка
}
```

Буфер вершин при створенні якого був вказаний прапорець D3DCREATE_WRITEONLY недоступний для читання, а доступний тільки для

запису. Для читання даних з буфера вершин потрібно параметру Usage функції CreateVertexBuffer присвоїти значення 0.

Після завершення доступу до буфера вершин викликається функція IDirect3DVertexBuffer9::Unlock:

```
HRESULT IDirect3DVertexBuffer9::Unlock();
```

Розблокування буфера вершин гарантує, що Direct3D може безпечно почати його використання, знаючи буфері дані не модифікувались.

Заповнення вершин. Маючи структуру даних вершини, дескриптор і заблокований буфер можна скопіювати необхідну кількість вершин у буфер вершин.

Продовжимо раніше наведений приклад і використовуючи визначений формат вершин (який містить тривимірні координати і розсіювану складову кольору), створимо локальний набір вершин в масиві:

```
sVertex Verts[4] = {
   {- 100.0f,  100.0f, 100.0f, D3DCOLOR_RGBA(255,255,255,255)},
   {  100.0f,  100.0f, 100.0f, D3DCOLOR_RGBA(255,  0,  0,255)},
   {  100.0f, - 100.0f, 100.0f, D3DCOLOR_RGBA(  0,255,  0,255)},
   { - 100.0f, - 100.0f, 100.0f, D3DCOLOR_RGBA(  0,  0,255,255)}
};
```

Блокуючи буфер вершин отримаємо покажчик на пам'ять буфера вершин, що дає змогу скопіювати локальні дані вершин (та після закінчення розблокувати буфер):

```
// pD3DVB = об'єкт буфера вершин
BYTE *Ptr;
// Блокуємо пам'ять буфера вершин і отримуємо покажчик на неї
if(SUCCEEDED(pD3DVB ->Lock(0, 0, (void**)&Ptr, 0))){
 // Копіюємо локальні вершини в буфер вершин
memcpy(Ptr, Verts, sizeof(Verts));
 // Розблоковуємо буфер вершин
 pD3DVB ->Unlock(); }
```

Наведений приклад ілюструє створення буфера вершин і заповнення його даними. Тепер залишилося тільки призначити джерело потокових даних і задати вершинний шейдер для обробки інформації про вершини.

Потік вершин. Direct3D дає змогу передавати вершини візуалізації через декілька каналів, що називаються потоками вершин (vertex stream). Об'єднуючи декілька потоків вершин в єдиний потік можна добитися хороших результатів.

Щоб вказати, які дані вершин передаватимуться в потік необхідно використати функцію IDirect3DDevice9::SetStreamSource:

```
HRESULT IDirect3DDevice9::SetStreamSource(
```

```
    UINT StreamNumber,  // 0
    IDirect3DVertexBuffer9* pStreamData, // Об'єкт буфера вершин
    UINT OffsetInBytes, // Зміщення (у байтах) цих вершин в буфері вершин.
    UINT Stride);       // Розмір структури даних вершини
```

Все, що необхідно робити для встановлення джерела потоку вершин - викликати функцію, передавши їй покажчик на об'єкт буфера вершин і вказати кількість байт, що будуть використанні для зберігання структури даних вершини (функція sizeof).

Якщо в буфері вершин зберігається більше однієї групи полігонів (наприклад, декілька смуг трикутників), програміст може в параметрі OffsetInBytes вказати зміщення в байтах до початку необхідної групи вершин. Наприклад, якщо при використанні буфера вершин для зберігання двох смуг трикутників (перша починається із зміщення 0, а друга - із зміщення 6), можна намалювати другу смугу трикутників, вказавши відповідне зміщення (у даному прикладі - 6).

Для нашого прикладу роботи з буфером вершин, приведеного в попередньому підпункті необхідно використати наступний код:

```
 // g_pD3DDevice = об'єкт пристрою
// pD3DVB       = буфер вершин
if(FAILED(g_pD3DDevice->SetStreamSource(0, pD3DVB,
                              0, sizeof(sVertex)))) {
// Виникла помилка
}
```

Вершинні шейдери. Для виконання завершального етапу використання вершин для малювання графіки необхідно познайомитися з концепцією вершинних шейдерів. Вершинний шейдер (vertex shader) - це механізм, що виконує завантаження і оброблення вершин, у який входить модифікація координат вершини, застосування кольорів і туману, а також оброблення інших компонентів даних вершини.

Існує дві форми вершинних шейдерів. Це може бути фіксований вершинний шейдер (у який вбудована вся функціональність, необхідна в переважній більшості випадків) або програмований вершинний шейдер (у якому записан власна процедура для модифікації даних вершини перед візуалізацією на екрані).

Для використання фіксованого вершинного шейдера для вершин потрібно передати FVF дескриптор вершини у функцію IDirect3DDevice9::SetFVF:

```
 HRESULT IDirect3DDevice9::SetFVF(
   DWORD FVF); // Дескриптор FVF вершини
// g_pD3DDevice = об'єкт пристрою
// VertexFVF    = дескриптор FVF вершини
if(FAILED(g_pD3DDevice ->SetFVF(VertexFVF))){
```

```
    // Виникла помилка
}
```

Тепер потрібно вказати записати перетворення, необхідні для розміщення вершини у глобальній системі координат. Звичайно, це необхідно робити тільки у тому випадку, якщо використовуються тривимірні координати.

Глобальні перетворення. Перетворення локальних координат вершин у відповідні координати глобального простору здійснюється за допомогою спеціальних перетворень. На рисунку 4.16 показано особливості переміщення вершин куба з локального простору в глобальний за допомогою спеціальних перетворень.

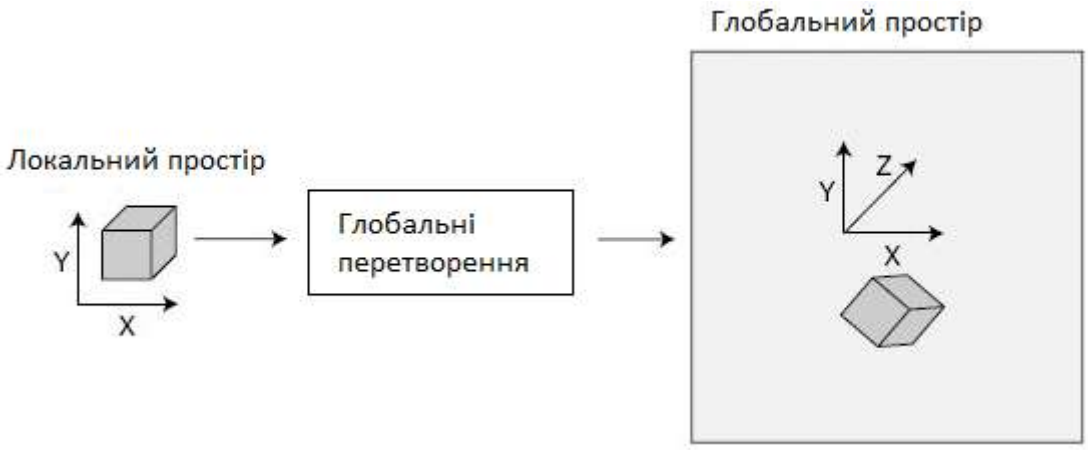

Рисунок 4.16 – Перетворення координат куба з локальних у глобальні

4.2.2. Відтворення аудіо засобами DirectX Audio і DirectShow

DirectX Audio складається з двох компонентів: DirectSound і DirectMusic. DirectSound - основний компонент, який застосовується для відтворення цифрового аудіо. DirectMusic виконує оброблення аудіо форматів та відправляє їх для відтворення компоненту DirectSound.

4.2.2.1. Особливості роботи з DirectSound

Для роботи з DirectSound необхідно створити COM-об'єкт, який служитиме інтерфейсом для аудіо обладнання. Створений COM-об'єкт дає можливість працювати з індивідуальними аудіо буферами для зберігання даних.

Дані з цих аудіо буферів змішуються у головному буфері мікшування і відтворюються у заданому аудіо форматі. Аудіо формати відтворення можуть відрізнятися частотою дискретизації, кількістю каналів та розрядності вибірки. Допустимі частоти - 8 000 Гц, 11 025 Гц, 22 050 Гц і 44 100 Гц. Для кількості каналів є два варіанти: один канал для монофонічного звуку або два канали для стереофонічного. Параметри розрядності також обмежуються

двома варіантами: 8 біт для низькоякісного відтворення аудіо та 16 біт для високоякісного.

За замовчуванням, якщо не виконувалось ручне налаштування вручну, для первинного аудіо буфера DirectSound встановлюють наступні параметри: 22 050 Гц, 8-розрядна вибірка, стереофонічний звук.

У процесі відтворення звуку можна модифікувати аудіо канали для зміни висоту звуку, змінювати гучність і панорамування звуку, і навіть зациклювати звук.

Для відтворення аудіо використовується метод потокового відтворення при якому в буфер завантажується невеликий фрагмент аудіо даних, а коли його відтворення закінчиться у буфер завантажується наступний фрагмент даних. Цей процес триває, поки весь аудіо файл не буде відтворено.

Для того щоб використовувати у проекті функції DirectSound і DirectMusic треба включити заготовочні файли dsound.h і dmusici.h, а також додати до списку компонування файл бібліотеки dsound.lib. Також при компонуванні потрібно підключити бібліотеку dxguid.lib, оскільки в ній визначені деякі корисні елементи DirectSound.

У таблиці 4.4 наведено основні COM-інтерфейси DirectSound.

Таблиця 4.4

COM-інтерфейси DirectSound

Назва інтерфейсу	Опис інтерфейсу
IDirectSound8	Інтерфейс головного об'єкта DirectSound.
IDirectSoundBuffer8	Об'єкт первинного та вторинного аудіо буферів. Зберігає дані і керує їх відтворенням.
IDirectSoundNotify8	Об'єкт повідомлення. Повідомляє додатком про досягненні заданої позиції в аудіо буфері.

На рисунку 4.17 показано взаємозв’язки між цими об'єктами.

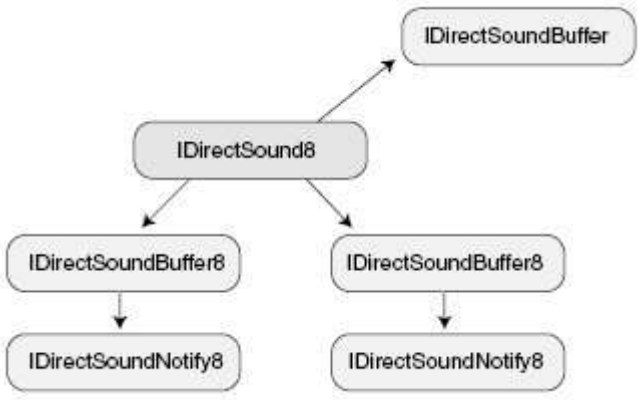

Рисунок 4.17 – Взаємозв’язок між об’єктами DirectSound, що використовується при відтворенні аудіо

IDirectSound8 - це головний інтерфейс з якого створюється аудіо буфер (IDirectSoundBuffer8). Потім аудіо буфер може створити власний інтерфейс повідомлення (IDirectSoundNotify8), який використовується для позначення позиції в аудіо буфері після досягнення якої буде надіслано повідомлення.

Ініціалізація DirectSound. Першим кроком програміста під час роботи з DirectSound є підключення заголовку dsound.h та бібліотеки dsound.lib. Після цього можна приступати до першого етапу використання DirectSound - створення об'єкта IDirectSound8, який є головним інтерфейсом аудіо обладнання. Робиться це за допомогою функції DirectSoundCreate8:

```
HRESULT WINAPI DirectSoundCreate8(
LPCGUID lpcGuidDevice, // Вкажіть NULL (аудіо пристрій по замовчуванню)
LPDIRECTSOUND8 * ppDS8, // Створюваний об'єкт
LPUNKNOWN pUnkOuter); // NULL - не використовується
```

Функція DirectSoundCreate8, подібно до всіх інших функцій DirectSound, повертає DS_OK, якщо вона завершилася успішно, або код помилки в іншому випадку. Використовуючи функцію DirectSoundCreate8 і глобальний екземпляр об'єкта IDirectSound8, можна ініціалізувати об'єкт аудіо системи наступним чином:

```
IDirectSound8 * g_pDS; // Глобальний об'єкт
IDirectSound8 if (FAILED (DirectSoundCreate8 (NULL, & g_pDS, NULL)))
 {// Помилка}
```

Встановлення рівня кооперації. Наступний етап ініціалізації – встановлення рівня кооперації (cooperative level) об'єкта IDirectSound8. Рівень кооперації використовується для того, щоб з'ясувати, як буде здійснюватися спільне з іншими додатками використання ресурсів звукової карти.

Для встановлення рівня кооперації використовується функція IDirectSound8::SetCooperativeLevel.

Існує чотири рівня кооперації (таблиця 4.5) кожному з яких відповідає власний макрос, визначений у DirectSound.

Таблиця 4.5

Рівні кооперації DirectSound

Рівень	Макрос	Опис
Нормальний	DSSCL_NORMAL	Звичайний рівень; дозволяє всім програмам звертатися до звукової карти, використовуючи формат відтворення по замовчуванню: 8 біт, 11025 Гц, 1 канал (моно). Цей формат не може бути змінений.
Пріоритетний	DSSCL_PRIORITY	Те ж, що і нормальний, але дозволяє змінювати формат відтворення.
Монопольний	DSSCL_EXCLUSIVE	Монопольне використання звукової карти; ніякі інші

		програми не можуть використовувати звуковий пристрій, поки програма активна. Задається формат відтворення.
Первинний запис	DSSCL_WRITEPRIMARY	Професійний рівень, що надає повний контроль над системою. Ви отримуєте доступ тільки до первинного аудіо буферу (вторинні аудіо буфери відключені).

Наведемо прототип функції IDirectSound8:: SetCooperativeLevel:

IDirectSound8:: SetCooperativeLevel (HWND hwnd, // Дескриптор батьківського вікна
DWORD dwLevel); // Рівень кооперації з таблиці 4.2

Рівень кооперації в більшості випадків визначається типом додатку. Для повноекранних додатків використовується монопольний рівень. В інших випадках рекомендується використовувати пріоритетний рівень.

Наведемо приклад встановлення пріоритетного рівня кооперації з використанням раніше ініціалізованого об'єкту IDirectSound8:

// G_pDS = раніше ініціалізований об'єкт IDirectSound8
// hWnd = раніше ініціалізований дескриптор батьківського вікна if (FAILED (g_pDS-> SetCooperativeLevel (hWnd, DSSCL_PRIORITY))) {/ / Помилка}

Встановлення формату відтворення. При використанні всіх рівнів кооперації (крім нормального), останнім етапом ініціалізації DirectSound буде захоплення керування первинним аудіо буфером і встановлення формату відтворення для системи. Цей процес ділиться на дві частини: спочатку використовується об'єкт IDirectSound8 для створення інтерфейсу буфера, а потім використовуєте інтерфейс для модифікації формату.

Створення об'єкта первинного звукового буфера. Об'єкт IDirectSoundBuffer представляє первинний аудіо буфер. Аудіо буфер (як первинний, так і вторинний) створює функція IDirectSound8:: CreateSoundBuffer, яка виглядає наступним чином:

HRESULT IDirectSound8:: CreateSoundBuffer (LPCDSBUFFERDESC pcDSBufferDesc, // Опис буфера
*LPDIRECTSOUNDBUFFER * ppDSBuffer, // Створюваний об'єкт буфера*
LPUNKNOWN pUnkOuter); / / NULL - не використовується

У параметрі DSBufferDesc передається покажчик на структуру DSBUFFERDESC, що зберігає інформацію про створюваний буфер. Для первинного буфера не використовуються всі можливості, надані об'єктом звукового буфера, але тим не менш поглянемо на його структуру:

typedef struct {DWORD dwSize; // Розмір структури
DWORD dwFlags; // Прапори, що описують можливості буфера

DWORD dwBufferBytes; // Розмір аудіо буфера
DWORD dwReserved; // Не використовується, встановіть 0
LPWAVEFORMATEX lpwfxFormat; / / Формат відтворення
GUID guid3DAlgorithm; // GUID_NULL (Алгоритм 3D-відтворення)}
*DSBUFFERDESC, * LPDSBUFFERDESC;*

Призначення полів зрозуміле із їх, за винятком lpwfxFormat, який є покажчиком на структуру, яка описує формат відтворення створюваного буфера.

Змінна dwFlags містить набір прапорців, що визначають можливості створюваного буфера. У таблиці 4.6 перераховані всі доступні для використання прапорці.

Таблиця 4.6

Прапорці для створення звукового буфера

Прапорці	Опис
DSBCAPS_CTRL3D	У буфера є тривимірні можливості.
DSBCAPS_CTRLFREQUENCY	Дозволяє міняти частоту в ході відтворення буфера.
DSBCAPS_CTRLFX	Буфер дозволяє обробку ефектів.
DSBCAPS_CTRLPAN	У буфера є можливість позиціонування.
DSBCAPS_CTRLPOSITIONNOTIFY	Буфер може використовувати повідомлення.
DSBCAPS_CTRLVOLUME	Дозволяє на льоту регулювати гучність для буфера.
DSBCAPS_GETCURRENTPOSITION2	Дозволяє запитувати буфер про місцезнаходження позиції відтворення.
DSBCAPS_GLOBALFOCUS	Створює глобальний аудіо буфер, це значить, що відтворений звук буде чутно навіть якщо активна інша програма.
DSBCAPS_LOCDEFER	Дозволяє буферу використовувати апаратні і програмні ресурси.
DSBCAPS_LOCHARDWARE	Змушує використовувати апаратні ресурси, такі як мікшер і пам'ять звукової карти.
DSBCAPS_LOCSOFTWARE	Змушує використовувати програмні ресурси, такі як програмне мікшування і системна пам'ять.
DSBCAPS_MUTE3DATMAXDISTANCE	Вимикає тривимірний звук, якщо його джерело знаходиться дуже далеко від слухача.
DSBCAPS_PRIMARYBUFFER	Робить створюваний буфер первинним звуковим буфером (використовується у всіх рівнях кооперації, крім нормального).
DSBCAPS_STATIC	Якщо можливо, буфер буде розміщений у пам'яті звукової карти.
DSBCAPS_STICKYFOCUS	Примушує буфер продовжувати відтворення, коли користувач перемикається на іншу програму, що не використовує DirectSound.

Надалі будемо використовувати тільки прапорці DSBCAPS_PRIMARYBUFFER і DSBCAPS_CTRLVOLUME. Ці прапорці повідомляють DirectSound про створення інтерфейсу первинного звукового буфера та забезпечують можливість регулювання гучності.

Необхідно відмітити, що деякі прапорці не можна використовувати для первинного буфера (прапорець контролю частоти)

Наведено приклад створення інтерфейсу первинного звукового буфера:

```
IDirectSoundBuffer g_pDSPrimary; // Глобальний доступ
DSBUFFERDESC dsbd; // Опис буфера
ZeroMemory (& dsbd, sizeof (DSBUFFERDESC)); / / Обнуляємо структуру
dsbd.dwSize = sizeof (DSBUFFERDESC); // Встановлюємо розмір структури
dsbd.dwFlags = DSBCAPS_PRIMARYBUFFER | DSBCAPS_CTRLVOLUME;
dsbd.dwBufferBytes = 0; // Не задаємо розмір буфера
dsbd.lpwfxFormat = NULL; // Не задаємо формат
if (FAILED (g_pDS-> CreateSoundBuffer (& dsbd, & g_pDSPrimary, NULL))) {
// Помилка}
```

Встановлення формату. Тепер, коли ми отримали можливість управляти первинним аудіо буфером, прийшов час встановити формат відтворення. Для цього є кілька варіантів, однак рекомендується використовувати осмислені значення, такі як 11 025 Гц, 16-розрядна вибірка, моно або 22 050 Гц, 16-розрядна вибірка, моно.

Вибираючи формат, постарайтеся не використовувати стереофонію. Вона створює непотрібне навантаження на процесор, оскільки справжні стереофонічні звуки досить складно записати. Також потрібно завжди використовувати 16-розрядну вибірку, оскільки її якість значно вища, ніж у 8-розрядної. Що стосується частоти, то чим вона вища - тим краще, але не варто вибирати частоту вище 22 050 Гц. Навіть звуки з CD-якістю прекрасно відтворюються на частоті 22 050 Гц без помітних втрат.

Формат відтворення встановлюється функцією IDirectSoundBuffer:: SetFormat:

```
HRESULT IDirectSoundBuffer:: SetFormat (LPCWAVEFORMATEX pcfxFormat);
```

Функція SetFormat повинна викликатися тільки для об'єкта первинного звукового буфера.

Перший і єдиний аргумент - це покажчик на структуру WAVEFORMATEX, що містить відомості про формат, який встановлюється:

```
typedef struct {WORD wFormatTag; // Вкажіть WAVE_FORMAT_PCM
WORD nChannels; // 1 для моно, 2 для стерео
DWORD nSamplesPerSec; // Частота дискретизації
DWORD nAvgBytesPerSec; // Кількість байт в секунду для формату
WORD nBlockAlign; // Розмір даних вибірки
WORD wBitsPerSample ; // 8 або 16
```

```
WORD cbSize; / / Не використовується
} WAVEFORMATEX;
```

Змінна nBlockAlign - це кількість байтів, що використовується для кожної вибірки аудіо. Вона ініціалізується наступним чином:

```
nBlockAlign = (nBitsPerSample / 8) * nChannels;
```

Змінна nAvgBytesPerSec - це кількість байт, необхідних для зберігання однієї секунди звуку:

```
nAvgBytesPerSec = nSamplesPerSec * nBlockAlign;
```

Наведено приклад встановлення формату 22 050 Гц, 16 біт, моно:

// G_pDSPrimary = раніше ініціалізований глобальний об'єкт первинного звукового буфера

```
WAVEFORMATEX wfex;
ZeroMemory (& wfex, sizeof (WAVEFORMATEX));
wfex.wFormatTag = WAVE_FORMAT_PCM;
wfex.nChannels = 1; // моно
wfex.nSamplesPerSec = 22050; // 22050 Гц
wfex.wBitsPerSample = 16; // 16 біт
wfex.nBlockAlign = (wfex.wBitsPerSample / 8) * wfex.nChannels;
wfex.nAvgBytesPerSec = wfex.nSamplesPerSec * wfex.nBlockAlign;
 if (FAILED (g_pDSPrimary-> SetFormat (& wfex) ))
{/ / Помилка}
```

Запуск первинного аудіо буфера. Запуск відтворення первинного буфера на початку програми дає змогу заощадити час процесора.

Перед тим, як показати запуск відтворення буфера необхідно ознайомитись з принципом роботи первинного аудіо буфера. Оскільки ресурси пам'яті обмежені, особливо у звукової карти, використовуваний буфер даних може бути будь-якого розміру (навіть всього кілька сотень байт). З цієї причини в якості первинного і вторинних аудіо буферів використовуються кільцеві буфери.

Приклад кільцевого буфера зображений на рисунку 4.18.

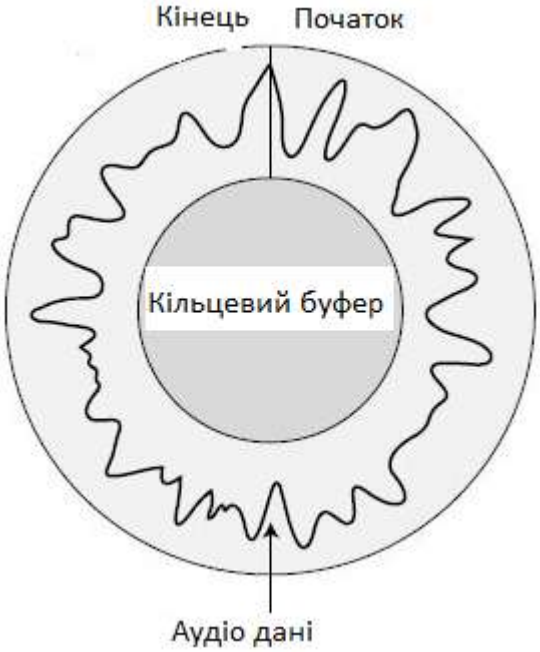

Рисунок 4.18 - Кільцеві буфери завжди замкнені, з'єднуючи початок з кінцем буфера, щоб звуки могли постійно ходити по колу для безперервного відтворення

Незважаючи на те, що буфер даних - це одновимірний масив, його кінець замикається на початок. Це потужна техніка, що дозволяє економити значні обсяги пам'яті.

Аудіо, що відтворюється змішуються в первинному звуковому буфері, який є кільцевим буфером даних. При досягненні кінця аудіо буфера, звук зациклюється на початок буфера і його відтворення продовжується без паузи. Щоб використовувати можливість зациклення буфера необхідно явно включити зациклене відтворення; інакше відтворення буфера буде зупинено після досягнення його кінця.

Щоб почати відтворення звукового буфера (з необов'язковою можливістю зацикленого відтворення) потрібно викликати функцію аудіо буфера Play , яка виглядає так:

```
HRESULT IDirectSoundBuffer8:: Play (DWORD Reserved1, // Повинен бути 0
DWORD Priority, // Пріоритет мікшування - використовується 0
DWORD dwFlags); // Прапори відтворення
```

Аргумент dwFlags може приймати два значення: 0 для одноразового відтворення звукового буфера і зупинки при досягненні кінця і DSBPLAY_LOOPING - повідомляє про необхідність постійно при досягненні кінця повертатися до початку буфера для безперервного відтворення.

Для первинного аудіо буфера практично завжди вимагається зациклене відтворення, яке можна запрограмувати наступним чином:

```
if (FAILED (g_pDSPrimary-> Play (0, 0, DSBPLAY_LOOPING)))
{/ / Помилка}
```

Після закінчення роботи з первинним аудіо буфером (і зі звуковою системою в цілому), треба зупинити її, викликавши функцію IDirectSoundBuffer:: Stop , у якої немає аргументів:

```
if (FAILED (g_pDSPrimary-> Stop ())) {/ / Помилка}
```

Використання вторинних аудіо буферів. Далі на черзі створення вторинних аудіо буферів, що містять реальні звукові дані, які можна відтворити. У DirectSound відсутні обмеження щодо кількості створюваних вторинних аудіо буферів та їх одночасного відтворення.

Для досягнення цього необхідно заповнити первинний аудіо буфер звуковими даними, що знаходяться у вторинних аудіо буферах (рисунок 4.19).

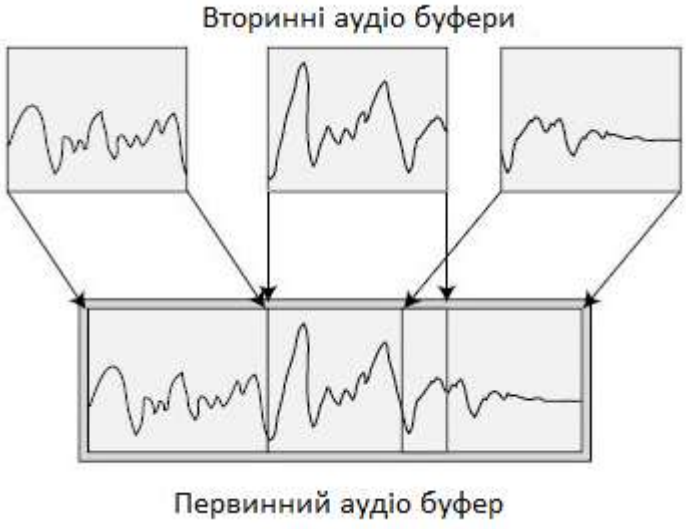

Рисунок 4.19 - Перед початком відтворення вторинні аудіо буфери мікшуються у первинному аудіо буфері

Вторинні аудіо буфери використовують об'єкт IDirectSoundBuffer8, дуже схожий на об'єкт IDirectSoundBuffer.

У створенні вторинних аудіо буферів є одна відмінність - при ініціалізації необхідно встановити формат відтворення. Це означає, що буфер може використовувати тільки один формат. Якщо формат потрібно змінити, то звільняється буфер і створюється інший формат.

Для зберігання формату використовується структура WAVEFORMATEX, а для опису можливостей буфера структура DSBUFFERDESC.

Наведемо приклад створення вторинного звукового буфера, що використовує формат 22 050 Гц, 16 біт, моно.

```
// G_pDS = раніше ініціалізований об'єкт
```

```
IDirectSound8 IDirectSoundBuffer8 * g_pDSBuffer; // Глобальний об'єкт 8 версії
IDirectSoundBuffer * pDSB; // Локальний аудіо буфер
// Ініціалізуємо структуру WAVEFORMATEX
 WAVEFORMATEX wfex; ZeroMemory (& wfex, sizeof (WAVEFORMATEX));
wfex.wFormatTag = WAVE_FORMAT_PCM ;
wfex.nChannels = 1; // моно
wfex.nSamplesPerSec = 22050; // 22050 Гц
wfex.wBitsPerSample = 16; // 16 біт
wfex.nBlockAlign = (wfex.wBitsPerSample / 8) * wfex.nChannels;
wfex.nAvgBytesPerSec = wfex.nSamplesPerSec * wfex.nBlockAlign;
// Ініціалізіруем структуру DSBUFFERDESC DSBUFFERDESC dsbd;
ZeroMemory (& dsbd, sizeof (DSBUFFERDESC)); // Обнуляємо структуру
dsbd.dwSize = sizeof (DSBUFFERDESC); // Задаємо розмір
dsbd.dwFlags = DSBCAPS_CTRLFREQUENCY | DSBCAPS_CTRLVOLUME |
DSBCAPS_CTRLPAN;
dsbd.dwBufferBytes = wfex.nAvgBytesPerSec * 2; // 2 секунди
dsbd.lpwfxFormat = &wfex; // Створюємо першу версію об'єкта
if (FAILED (g_pDS-> CreateSoundBuffer (& dsbd, & pDSB, NULL)))
{/ / Відбулася помилка}
else {
// Отримуємо 8 версію інтерфейсу
if (FAILED (pDSB-> QueryInterface (IID_IDirectSoundBuffer8, (void **) & g_pDSBuffer))) {
// Помилка. Звільняємо перший інтерфейс
pDSB- > Release ();
} else {
// Звільняємо вихідний інтерфейс - все успішно!
pDSB-> Release ();}
}
```

Завантаження звукових даних у буфер. У розпорядженні звукових буферів є пара функцій:

- IDirectSoundBuffer8:: Lock, яка блокує буфер аудіо даних і повертає вказівник на область даних,
- IDirectSoundBuffer8:: Unlock звільняє ресурси, які використовуються під час операції блокування.

Блокуючи буфер відбувається підготовка його для запису. Наступним етапом є повідомлення буфера по зсув (у байтах), починаючи з якого необхідно почати доступ, і про кількість байт до яких необхідно доступ. У свою чергу програміст отримуєте два покажчики на дані і дві змінні, які повідомляють скільки є даних.

Два покажчика та два розміри отримується тому що аудіо буфер є кільцевим і для блокування необхідної кількості байт може знадобитися повернутися до початку буфера. Перший покажчик - це запитана позиція, а перший розмір - це кількість байт від цієї позиції до кінця буфера. Другий покажчик зазвичай встановлюється на початок буфера, а другий розмір - це залишкова кількість заблокованих байт, які знаходяться за кінцем звукового

буфера. На рисунку 4.20 показано приклад буфера даних з двома покажчиками і розмірами.

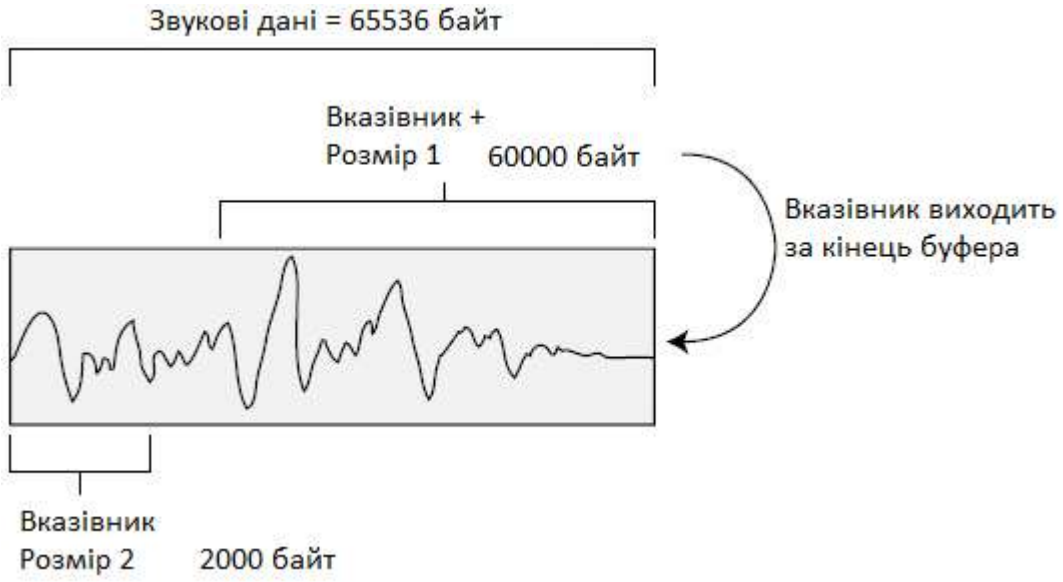

Рисунок 4.20 - Буфер даних, розміром 65536 байт, в якому блокуються для доступу 62000 байт. Перший покажчик використовується для доступу до 600000 байт, а другий покажчик - для доступу до решти 2000 байт

Ось як виглядає прототип функції блокування:

```
HRESULT IDirectSoundBuffer8:: Lock (
DWORD dwOffset, // Зсув в буфері до початку блокуючої області
DWORD dwBytes, // Кількість заблокованих байт
LPVOID * ppvAudioPtr1, // Покажчик на покажчик першого блоку даних
LPDWORD * ppwAudioBytes1, // Покажчик на розмір першого блоку
LPVOID * ppvAudioPtr2, // Покажчик на покажчик другого блоку даних
LPDWORD * ppwAudioBytes2, // Покажчик на розмір другого блоку
DWORD dwFlags); // Прапорці блокування
```

Тут слід звернути увагу на декілька особливостей. По-перше треба передати пару покажчиків (зведених до типу LPVOID *), які вказують на відповідні дані в заблокованому аудіо буфері. Якщо розмір блокуючого простору перевищує розмір фрагмента буфера, що залишився і виникає ефект закольцування, то використовується два покажчики.

Також два покажчика потрібно на пару змінних типу DWORD, у які буде занесено (функцією Lock) кількість байт аудіо буфера, до яких надається доступ через два різних покажчики на дані.

По-друге, змінна dwFlags може приймати три значення: 0 для блокування фрагмента, DSBLOCK_FROMWRITECURCOR для блокування від поточної позиції запису і DSBLOCK_ENTIREBUFFER для блокування всього буфера, ігноруючи зазначені зміщення і розмір.

Найкраще присвоювати змінній dwFlags значення 0 - це гарантує блокування тільки вказаної кількість байт у заданому місці.

Необхідно відмітити, що тривале блокування буфера призводить до небажаних ефектів.

Наведемо приклад блокування всього буфера і заповнення його випадковими числами:

```
// G_pDSBuffer = раніше ініціалізований вторинний аудіо буфер
char * Ptr;
DWORD Size;
if (SUCCEEDED (g_pDSBuffer-> Lock (0, 0, (void **) & Ptr, (DWORD *) & Size, NULL, 0, DSBLOCK_ENTIREBUFFER) )) {
for (long i = 0; i <Size; i + +)
Ptr [i] = rand ()% 65536;
```

Розблокування та звільнення буфера виконується за допомогою функції IDirectSoundBuffer8:: Unlock:

```
HRESULT IDirectSoundBuffer8:: Unlock (
LPVOID pvAudioPtr1, // Перший покажчик на дані
DWORD dwAudioBytes1, // Перший розмір
LPVOID pvAudioPtr2, // Другий покажчик на дані
DWORD dwAutioBytes2); // Другий розмір
```

Необхідно передати значення (а не покажчики на них), отримані при блокуванні буфера:

```
if (FAILED (g_pDSBuffer-> Unlock ((void *) Ptr, Size, NULL, 0)))
{/ / Помилка}
```

Відтворення аудіо буфера. Буфер розблоковано і дані завантажені - настав час відтворювати аудіо. Для відтворення вторинного аудіо буфера викликається та ж функція, яка застосовувалася для відтворення первинного буфера і була описана раніше. Проте цього разу не потрібно зациклювати звук, тому прапорець DSBPLAY_LOOPING потрібно вимкнути.

Наступним етапом є повідомлення звукового буфера з якої позиції всередині нього треба починати відтворення. Зазвичай, перше відтворення звуку починається з його початку. Однак зупинка звуку не скидає позицію відтворення, оскільки програміст може включити паузу, а потім викликати функцію відтворення знову, щоб продовжити прослуховування з того місця, на якому в останній раз зупинився. Встановлення позиції відтворення виконується за допомогою наступної функції:

```
HRESULT IDirectSoundBuffer8:: SetCurrentPosition (DWORD dwNewPosition);
```

У функції всього один аргумент - зміщення, з якого потрібно почати відтворення звуку. Зміщення має бути вирівняне на розмір блоку вибірки, який був заданий при створенні буфера. Якщо кожного разу при відтворенні користувач хоче слухати звук з початку, можна використовувати наступний код:

```
// G_pDSBuffer = ініціалізований аудіо буфер із завантаженими даними
if (SUCEEDED (g_pDSBuffer-> SetCurrentPosition (0))) {
if (FAILED (g_pDSBuffer-> Play (0,0,0))) {
// Помилка}}
```

Щоб зупинити відтворення використовується функція IDirectSoundBuffer8:: Stop:

```
if (FAILED (g_pDSBuffer-> Stop ())) {
// Помилка}
```

Зміна гучності, позиціонування і частоти. Якщо при створенні буфера були вказані відповідні прапорці, то у процесі відтворення аудіо можна змінювати гучність, позиціонування і частоту цього буфера.

DirectSound відтворює аудіо з тією гучністю, з якою вони були записані. Він не підсилює звуки, щоб зробити їх голоснішими, оскільки це завдання апаратури.

DirectSound тільки робить звук тихішим. Це виконується шляхом задання рівня звуку в сотих частках децибел в діапазоні від 0 (повна гучність) до -10 000 (тиша).

Для зміни гучності потрібно викликати наступну функцію:

```
HRESULT IDirectSoundBuffer8:: SetVolume (LONG lVolume);
```

Єдиний аргумент цієї функції це рівень гучності в сотих частках децибел. Як приклад налаштування гучності наведемо приклад, який зменшить гучність на 25 децибел:

```
// G_pDSBuffer = раніше ініціалізований аудіо буфер
if (FAILED (g_pDSBuffer-> SetVolume (-2500))) {
// Помилка}
```

Позиціонування - це можливість пересувати центр відтворення між правим і лівим динаміками (як показано на рисунку 4.21).

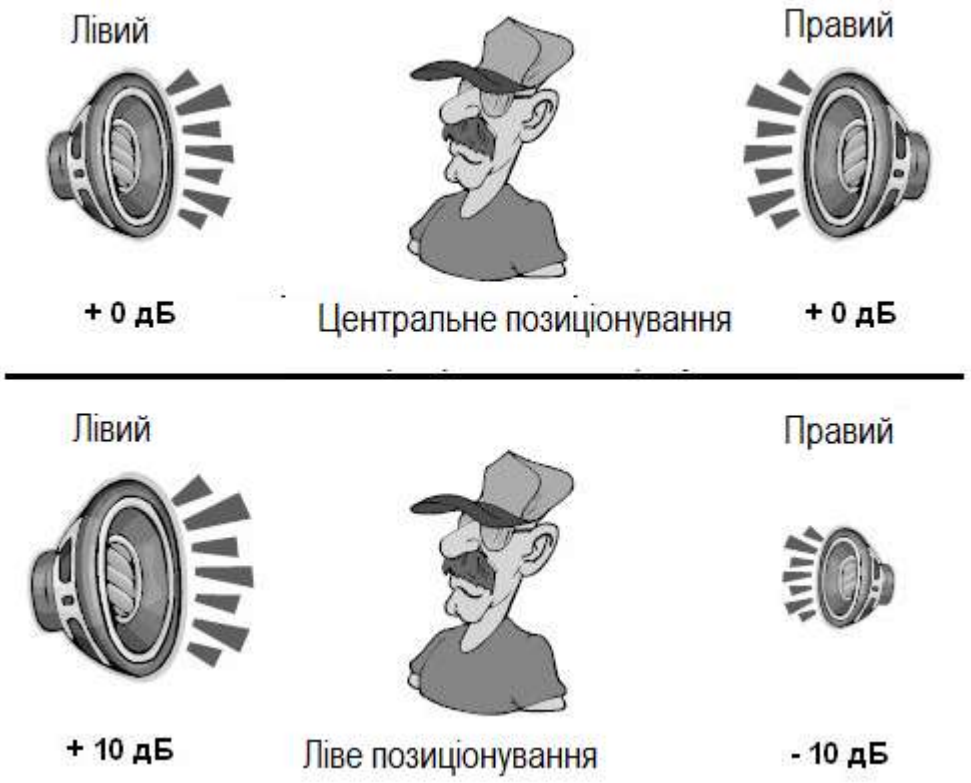

Рисунок 4.21 – Приклад позиціонування

Зазвичай динаміки відтворюють звук з однаковою гучністю (вимірюється в децибелах, дБ). Позиціонування зменшує гучність в одному з динаміків і збільшує в іншому, що призводить до виникнення псевдотривимірних ефектів.

Позиціонування визначає, наскільки звук буде зміщений вліво або вправо. Крайній лівій позиції (працює тільки лівий динамік) відповідає значення -10000, а крайній правій (працює тільки правий динамік) відповідає 10000. Всі проміжні значення відповідають певному балансу між правим і лівим каналами.

DirectSound визначає два макроси, що представляють крайній лівий і правий крайній рівні позиціонування це DSBPAN_LEFT і DSBPAN_RIGHT відповідно.

Для роботи з позиціонуванням використовується функція:

HRESULT IDirectSoundBuffer8:: SetPan (LONG lPan);

У аргументі lPan заданої функції встановлюється необхідний рівень позиціонування. Для прикладу встановимо значення позиціонування буфера рівним -5000, що зменшить гучність правого динаміка на 50 дБ:

```
// G_pDSBuffer = раніше ініціалізований аудіо буфер
if (FAILED (g_pDSBuffer-> SetPan (-5000)))
 {// Помилка}
```

Щоб під час відтворення можна було керувати позиціонуванням звуку, при створенні аудіо буфера необхідно вказати прапорець DSBCAPS_CTRLPAN.

Зміна частоти. Зміна частоти відтворення аудіо буфера змінює висоту звуку та дозволяє змінити тембр його звучання.

Для встановлення частоти використовується наступна функція:

```
HRESULT IDirectSoundBuffer8:: SetFrequency (DWORD dwFrequency);
```

У аргументі dwFrequency наведеної функції необхідно встановити потрібне значення частоти:

```
// G_pDSBuffer = раніше ініціалізований аудіо буфер
if (FAILED (g_pDSBuffer-> SetFrequency (22050)))
{/ / Помилка}
```

Необхідно відмітити, що зміна частоти відтворення викликає ефект стиснення звукової хвилі через що вона відтворюється за менший час (рисунок 4.22).

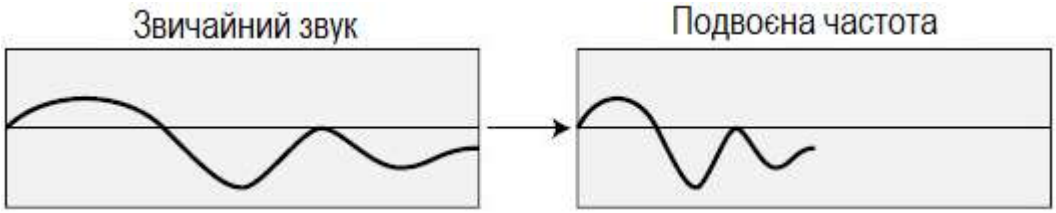

Рисунок 4.22 – Ілюстрація прикладу подвоєння частоти дискретизації

Втрата фокусу. Часто при роботі з аудіо додатки операційної системи можуть «викрасти» ресурси. Для відновлення цих ресурсів потрібно викликати функцію IDirectSoundBuffer8:: Restore.

Наприклад, якщо втрачено буфер, його можна відновити (і всю пов'язану з буфером пам'ять) використовуючи наступний код:

```
// G_pDSBuffer = раніше ініціалізований аудіо буфер, що був втрачений
if (FAILED (g_pDSBuffer-> Restore ()))
{// Помилка}
```

Для уникнення втрати ресурсів потрібно використовувати прапорець DSBCAPS_LOCKSOFTWARE, який дозволяє використовувати системну пам'ять.

Використання повідомлень. Як було сказано вище, повідомлення - це маркери у аудіо буфері при досягненні яких генерується сигнал про подію, що

сталася. Працюючи з повідомленнями програміст отримуєте можливість дізнатися, що відтворення аудіо завершене або припинене. Найчастіше повідомлення використовуються для потокового відтворення великих аудіо файлів.

Повідомлення використовують об'єкт IDirectSoundNotify8. Метою цього об’єкту - відзначати позиції у аудіо буфері і згенерувати події для програми, які можуть оброблятися в циклі обробки повідомлень або в окремому потоці.

Позиції визначаються за їх зміщення в буфері (рисунок 4.23) або за допомогою макросу DSBPN_OFFSETSTOP, що позначає паузу або завершення відтворення.

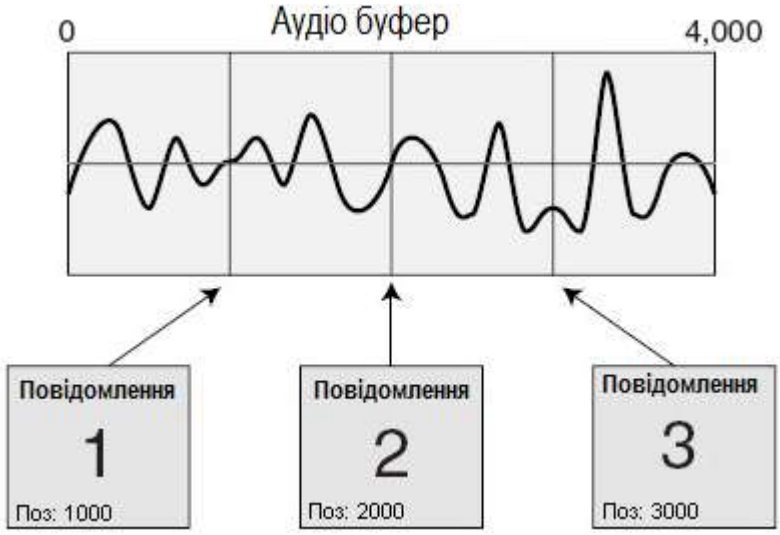

Рисунок 4.23 - Повідомлення можуть бути розміщені (шляхом зазначення зсуву) в будь-якому місці аудіо буфера

Зміщення в аудіо буфері має бути вирівняне за розміром блоку вибірки. Також повідомлення повинні бути впорядковані від менших зміщень до більших і в жодному разі два повідомлення не можуть використовувати один і той же зсув.

Наприклад, якщо розмір блоку дорівнює 2 (монофонічний звук, 16 біт) і програміст встановлює зсув 4 та 5, то відбудеться збій, тому що зміщення 4 і 5 відповідають одній вибірці.

Об'єкт IDirectSoundNotify8 викликається через об'єкт IDirectSoundBuffer8:

```
// G_pDSBuffer = раніше ініціалізований вторинний аудіо буфер
IDirectSoundNotify8 * g_pDSNotify;
if (FAILED (g_pDSBuffer-> QueryInterface (IID_IDirectSoundNotify8, (void **) & g_pDSNotify)))
{/ / Помилка}
```

Для використання повідомлень при створенні аудіо буфера необхідно задати прапорець DSBCAPS_CTRLPOSITIONNOTIFY.

У інтерфейсу повідомлення є тільки одна функція:

HRESULT IDirectSoundNotify8:: SetNotificationPositions (
DWORD dwPositionNotifies, // Кількість повідомлень
LPCDSBPOSITIONNOTIFY pcPositionNotifies); // Масив зсувів

Параметр pcPositionNotifies є вказівником на масив структур DSBPOSITIONNOTIFY. Наведемо опис вказаної структури:

typedef struct {DWORD dwOffset; // Зсув або макрос
DSBPN_OFFSET HANDLE hEventNotify; // Дескриптор події
*} DSBPOSITIONOTIFY, * LPCDSBPOSITIONNOTIFY;*

У події є два стани - відбулося (встановлено) або не відбулось (скинуто). Створюючи події необхідно вказати змінну дескриптора і провести її ініціалізацію наступним чином:

HANDLE hEvent; hEvent = CreateEvent (NULL, FALSE, FALSE, NULL);

Після завершення використання події, слід викликати CloseHandle (hEvent) для звільнення ресурсів.

Найкращим способом зберігання подій є масив.

Розглянемо приклад встановлення події і зсуву повідомлення. Для цього використаємо аудіо буфер розміром 65536 байт і створимо дві події, що представляють середину і кінець буфера відповідно:

/ / G_pDSBNotify = раніше ініціалізований об'єкт повідомлення
HANDLE g_hEvents [2]; // Глобальний дескриптор
DSBPOSITIONNOTIFY dspn [2]; // Два локальних зсуви для встановлення
g_hEvents [0] = CreateEvent (NULL, FALSE, FALSE, NULL);
g_hEvents [1] = CreateEvent (NULL, FALSE, FALSE, NULL);
dspn [0]. dwOffset = 32768; // Маркер середини
dspn [0]. hEventNotify = g_hEvents [0];
dspn [1]. dwOffset = DSBPN_OFFSETSTOP; // Маркер кінця звуку
dspn [1]. hEventNotify = g_hEvents [1];
if (FAILED (g_pDSBNotify-> SetNotificationPositions (2, dspn)))
{/ / Помилка}

Тепер буфер готовий, можна запустити відтворення аудіо буфера і дозволити події. Далі слід просто спостерігати за подіями, очікуючи їх сигналу. Для очікування подій використовується функція WaitForMultipleObjects:

DWORD WaitForMultipleObjects (DWORD nCount, // Кількість очікуваних подій (не більше 64)
*CONST HANDLE * lpHandles, // Масив дескрипторів очікуваних подій*
BOOL fWaitAll, // FALSE (не чекати всі)
DWORD dwMilliseconds); // INFINITE (чекати завжди)

Аргумент nCount зберігає кількість сканованих подій, а lpHandles - масивом дескрипторів подій, які сканує функція. Повернувшись з цієї функції ми отримуємо номер позиції події, що сталася в масиві.

Функція WaitForMultipleObjects може повернути значення WAIT_FAILED, яке вказує що під час очікування події сталася помилка. У такому випадку потрібно перезапустити очікування.

Для того щоб отримати з повернутого значення номер події потрібно відняти значення WAIT_OBJECT_0 з повернутого функцією значення, і ви отримаєте число з діапазону від 0 до кількості подій мінус одиниця.

Наведемо приклад функції, яка відтворює аудіо буфер і чекає виникнення раніше встановлених подій:

// Функції передається ініціалізований аудіо буфер до встановлених подіями

```
// g_Events = раніше ініціалізований масив подій з двох елементів: HANDLE g_Events [2];
void PlayItAndWait (IDirectSoundBuffer8 * pDSB) {
DWORD RetVal, EventNum; // Починаємо відтворення аудіо з початку буфера
pDSBuffer-> SetCurrentPosition (0);
pDSBuffer-> Play (0,0,0); while (1) {
while ((RetVal = WaitForMultipleObjects (2, g_Events, FALSE, INFINITE))! = WAIT_FAILED) {
EventNum = RetVal - WAIT_OBJECT_0; // Перевіряємо подію завершення звуку і перериваємо цикл
if (EventNum == 1) break;}} // Зупинка відтворення
pDSBuffer-> Stop ();}
```

Єдина проблема з наведеним вище прикладом відтворення звуку полягає в тому, що код повинен бути поміщений в головному циклі обробки повідомлень програми і, таким чином, в ньому треба сканувати стандартні повідомлення Windows.

Використання потоків для сканування подій. Відомо що для закриття потоку він має всередині себе викликати функцію ExitThread. Але як потік дізнається, коли це треба робити, якщо він зайнятий нескінченним скануванням списку повідомлень? Рішення - додати ще одну подію, яка буде повідомляти про те, що треба закрити потік.

Щоб вручну згенерувати подію використовується наступний виклик з дескриптором події SetEvent (hEvent).

Щоб скинути подію використовується виклик функції ResetEvent (hEvent).

Повернемося до циклу сканування подій, але тепер додамо функцію для відтворення аудіо буфера, для паузи і підтримку потоків:

```
HANDLE g_Events [2]; // Глобальні події
```

```
IDirectSoundNotify8 * g_pDSBNotify; // Глобальний об'єкт повідомлення
HANDLE g_hThread; // Дескриптор потоку
BOOL g_Active = FALSE; // Прапор активності потоку
BOOL g_Playing = FALSE; // Прапор відтворення звуку
void PlaySound (IDirectSoundBufffer8 * pDSBuffer) {
DSBPOSITIONNOTIFY dspn [1];
DWORD ThreadId; // Зупиняємо звук, якщо він вже відтворюється
if (g_Playing == TRUE) StopSound (pDSBuffer); // Отримуємо об'єкт повідомлення
pDSBuffer-> QueryInterface (IID_IDirectSoundNotify8, (void **) & g_pDSBNotify); / / Створюємо події і потік
g_hEvents [0] = CreateEvent (NULL, FALSE, FALSE, NULL);
g_hEvents [1] = CreateEvent (NULL, FALSE, FALSE, NULL);
g_hThread = CreateThread ( NULL, 0, LPTHREAD_START_ROUTINE) MyThread, NULL, 0, & ThreadId);
// Встановлюємо позицію повідомлення
dspn [0]. dwOffset = DSBPN_OFFSETSTOP;
dspn [0]. hEventNotify = g_hEvents [0];
g_pDSBNotify-> SetNotificationPositions (1, dspn );
// Починаємо відтворення і встановлюємо прапорець
pDSBuffer-> SetCurrentPosition (0);
pDSBuffer-> Play (0, 0, 0); g_Playing = TRUE;
}
void StopSound (IDirectSoundBuffer8 * pDSBuffer) {pDSBuffer-> Stop ();
g_Playing = FALSE;
// Очищаємо подію звукового буфера і сигналізуємо про закриття потоку
while (g_Active == TRUE) {ResetEvent (g_Events [0]); SetEvent (g_Events [1]);}
// Звільняємо всі ресурси
g_pDSBNotify-> Release ();
CloseHandle (g_hEvents [0]);
CloseHandle (g_hEvents [1]);
CloseHandle (g_hThread);}
DWORD WINAPI MyThread (void * lpParameter) {
DWORD RetVal, EventNum; g_Active = TRUE;
while (1) {
while ( (RetVal = WaitForMultipleObjects (2, g_Events, FALSE, INFINITE)! = WAIT_FAILED) {
EventNum = RetVal - WAIT_OBJECT_0;
// Перевіряємо, чи закривати потік
if (EventNum == 1) ExitThread (0);
// Звук зупинено - скидається прапорець
if (EventNum == 1) {g_Playing = FALSE;}}}}
```

Викликавши функцію PlaySound та передавши їй аудіо буфер можна прослухати сформований у прикладі звук. Для звільнення ресурсу і закриття потоку викликається функція StopSound.

Завантаження аудіо в буфер. Найпростіший спосіб завантаження аудіо є використання WAV формату звукових файлів від Microsoft.

WAV-файли починаються з невеликого заголовка, за яким знаходяться необроблені аудіо дані.

Наведемо приклад структури, яка створена для того щоб зберти заголовок WAV файлу для подальшого використання:

```
typedef struct sWaveHeader {
{char RiffSig [4]; // 'RIFF'
long WaveformChunkSize; // 8 char
WaveSig [4]; // 'WAVE'
char FormatSig [4]; // 'fmt'
long FormatChunkSize; // 16 short
FormatTag; // WAVE_FORMAT_PCM
short Channels; // кількість каналів
long SampleRate; // частота вибірки
long BytesPerSec; // байт на секунду
short BlockAlign; // вирівнювання блоку вибірки
short BitsPerSample; // біт у вибірці
char DataSig [4]; // 'data'
long DataSize; / / розмір даних
} sWaveHeader;
```

Більшість WAV файлів при збереженні використовують заголовок, аналогічний до наведеного у структурі sWaveHeader, однак іноді в цей заголовок вставляються додаткові блоки. Наприклад, перед блоком даних може бути вставлений блок коментарів.

Для обробки заголовка треба відкрити WAV файл і відразу ж зчитати з нього дані. Структура буде містити всю інформацію, необхідну для визначення формату аудіо, а також розмір даних для подальшого читання.

Створимо декілька функцій, які завантажують WAV файл у створюваний вторинний аудіо буфер. Перша функція читає і аналізує заголовок WAV файлу, створюючи по ходу аудіо буфер, друга читає звукові дані і поміщає їх у аудіо буфер.

Перша функція CreateBufferFromWAV, отримує вказівник на відкритий WAV файл, а також покажчик на структуру sWaveHeader, яка буде заповнена даними заголовка, отриманими з файлу. Після повернення з функції CreateBufferFromWAV ми отримаємо вказівник на новий створений об'єкт IDirectSoundBuffer8, який готовий прийняти аудіо дані, одержані після виклику функції LoadSoundData. Наведемо код цих двох функцій:

```
// G_pDS = раніше ініціалізований об'єкт IDirectSound8
IDirectSounndBuffer8 * CreateBufferFromWAV (FILE * fp, sWaveHeader * Hdr) {
IDirectSoundBuffer * pDSB; IDirectSoundBuffer8 * pDSBuffer; DSBUFFERDESC dsbd;
WAVEFORMATEX wfex;
// Читаємо заголовок з початку файлу
fseek (fp, 0, SEEK_SET) ;
fread (Hdr, 1, sizeof (sWaveHeader), fp);
// Перевіряємо поля сигнатур, повертаємося у разі помилки
if (memcmp (Hdr-> RiffSig, "RIFF", 4) | |
 memcmp (Hdr-> WaveSig , "WAVE", 4) | |
```

```
memcmp (Hdr-> FormatSig, "fmt", 4) | |
memcmp (Hdr-> DataSig, "data", 4))
return NULL;
// Встановлюємо формат відтворення
ZeroMemory- (& wfex, sizeof (WAVEFORMATEX));
wfex.wFormatTag = WAVE_FORMAT_PCM;
wfex.nChannels = Hdr-> Channels;
wfex.nSamplesPerSec = Hdr-> SampleRate;
wfex.wBitsPerSample = Hdr-> BitsPerSample;
wfex.nBlockAlign = wfex.wBitsPerSample / 8 * wfex . nChannels;
wfex.nAvgBytesPerSec = wfex.nSamplesPerSec * wfex.nBlockAlign;
// Створюємо аудіо буфер, використовуючи дані заголовка
ZeroMemory (& dsbd, sizeof (DSBUFFERDESC)); dsbd.dwSize = sizeof (DSBUFFERDESC);
dsbd.Flags = DSBCAPS_CTRLVOLUME | DSBCAPS_CTRLPAN |
DSBCAPS_CTRLFREQUENCY;
dsbd.dwBufferBytes = Hdr-> DataSize;
dsbd.lpwfxFormat = &wfex;
if (FAILED (g_pDS-> CreateSoundBuffer (& dsbd, & pDSB, NULL)))
return NULL;
// Отримуємо нову версію інтерфейсу
if (FAILED (pDSB- > QueryInterface (IID_IDirectSoundBuffer8, (void **) & pDSBuffer))) {
pDSB-> Release ();
return NULL;}
// Повертаємо інтерфейс
return pDSBuffer;
} BOOL LoadSoundData (IDirectSoundBuffer8 * pDSBuffer, long LockPos, FILE * fp, long Size )
{BYTE * Ptr1, * Ptr2;
DWORD Size1, Size2;
if (! Size) return FALSE;
// Блокуємо аудіо буфер з заданої позиції
if (FAILED (pDSBuffer-> Lock (LockPos, Size, (void **) & Ptr1, & Size1, (void **) & Ptr2, &
Size2, 0)))
return FALSE;
// Читаємо дані
fread (Ptr1, 1, Size1, fp);
if (Ptr2! = NULL) fread (Ptr2, 1, Size2, fp) ;
// розблокуємо буфер
pDSBuffer-> Unlock (Ptr1, Size1, Ptr2, Size2);
// Повертаємо прапор успіху
return TRUE;}
```

Наведемо приклад функції, яка використовує функції CreateBufferFromWAV і LoadSoundData для завантаження WAV файлу. Після повернення з показаної нижче функції LoadWAV одержимо готовий для роботи аудіо буфер:

```
IDirectSoundBuffer8 * LoadWAV (char * Filename) {
IDirectSoundBuffer8 * pDSBuffer;
sWaveHeader Hdr;
FILE * fp;
// Відкриваємо вихідний файл
```

```
if ((fp = fopen (Filename, "rb "))== NULL) return NULL;
// Створюємо аудіо буфер
if ((pDSBuffer = CreateBufferFromWAV (fp, & Hdr)) == NULL) {
fclose (fp);
return NULL;}
// Читаємо дані
fseek (fp, sizeof (sWaveHeader), SEEK_SET);
LoadSoundData (pDSBuffer, 0, fp , Hdr.DataSize);
// Закриваємо вихідний файл
fclose (fp);
// Повертаємо новий аудіо буфер з  записаним у нього звуком return pDSBuffer;}
```

Потокове відтворення аудіо. Секрет потокового відтворення полягає у використанні зацикленого відтворення, що забезпечує безперервний звук без пауз, і постійне завантаження нових звукових даних на заміну тим, які вже відіграли. Для здійснення такого відтворення необхідно встановити декілька маркерів у аудіо буфері. Коли відтворений звук проходить один з цих маркерів, ви отримуєте сигнал, що прийшов час завантажувати наступну аудіо вибірку. Такий процес дозволяє гарантувати, що коли відтворення повернеться до початку буфера там вже будуть знаходитись нові аудіо дані. На рисунку 4.24 показано аудіо буфер з чотирма потоковими маркерами, що повідомляють коли мають бути завантажені нові дані.

Рисунок 4.24 – Графічне відображення потокового відтворення аудіо

Як видно з рисунку 4.24 у вторинному аудіо буфері є чотири потокових маркера. Коли відтворення досягає одного з цих маркерів, аудіо буфер сигналізує, що треба завантажувати нові звукові дані в тільки що відтворену секцію

Вище була розглянута функція LoadSoundData, яка завантажує аудіо дані у визначений буфер. У потоці, що обробляє події повідомлень,

використовується функція завантаження для підтримки потоку даних замість завершених, забезпечуючи заповнення буфера аудіо інформацією. Для виконання цих дій:

1. Створюємо аудіо буфер, скажімо, розміром 65536 байт.
2. Встановлюємо чотири позиції повідомлень (в кінці кожної чверті аудіо буфера).
3. Завантажуємо в буфер стільки даних, скільки поміститься.
4. Запускаємо відтворення аудіо буфера, встановивши прапор DSBPLAY_LOOP.

Щоб визначити, яка подія відзначає кінець аудіо, визначається модуль розміру аудіо за розміром буфера (залишок від ділення розміру аудіо на розмір буфера). Розділіть значення модуля на чотири (кількість повідомлень) щоб визначити, яка з подій використовується для оповіщення кінцевої позиції звуку.

4.2.2.2. Особливості роботи з DirectMusic

DirectMusic використовується для відтворення музики з файлів MIDI (файли з розширенням .MID), файлів DirectMusic (файли з розширенням .SGT) і цифрових записів пісень, що зберігаються у WAV форматі (файли з розширенням .WAV).

У кожного з перелічених форматів музичних файлів є свої переваги і недоліки. Найкраще робота DirectMusic проявляється, при роботі з файлами з розширенням .SGT. Пісня у цьому форматі складається з невеликих музичних паритур, які можуть випадково відтворюватися одна за одною в різних акордах. Випадковий вибір партитур і акордів означає робить музику унікальною

Перевага використання файлів MIDI - їх широка підтримка.В Інтернеті можна знайти тисячі пісень у цьому форматі, і сотні програмних пакетів для створення MIDI-музики. У цьому форматі замість звичайних інструментів можна використовувати будь-які записані в цифровій формі звуки.

Використання цифрових записів інструментів гарантує ідентичність звучання звуків на всіх комп'ютерах.

DirectMusic має можливість створювати власні інтерфейси DirectSound або використовувати вже створені.

Використання SGT та MIDI форматів має одну спільну перевагу (можливість змінювати темп відтворення), що додає можливість прискорювати і сповільнювати музику, відповідно до сюжету на екрані. Коли на екрані розгортаються активні дії темп музики можна збільшити, і навпаки.

Початок роботи з DirectMusic. Перший етап використання DirectMusic - це створення головного об'єкту (виконавця), який представляє музичну

систему. Потім, створюється об'єкт-завантажувач, який завантажує всі основні музичні файли. Взаємодія між цими двома об'єктами показано на рисунку 4.25.

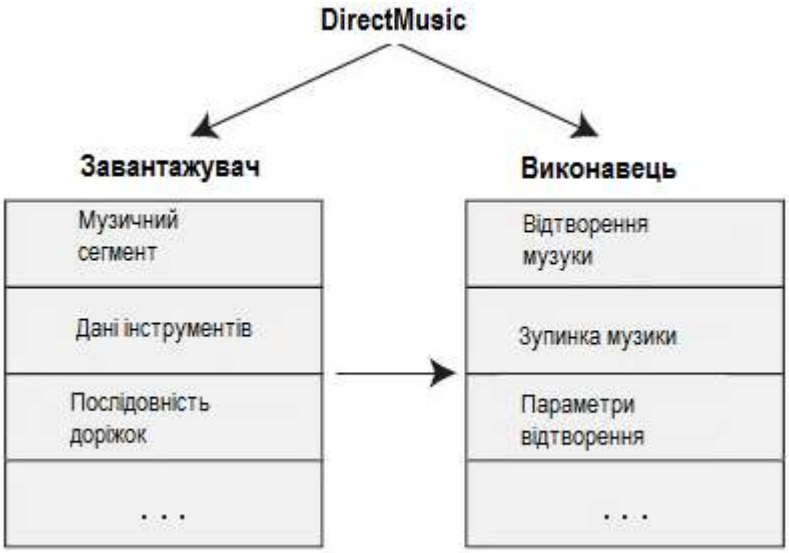

Рисунок 4.25 - Об'єкт завантажувача надає дані, необхідні об'єкту виконавця для відтворення музики

Наступним етапом є завантаження реальних музичних сегментів в об'єкти сегментів. Для створення довгих або більш динамічних пісень можна завантажити декілька сегментів і відтворювати їх один за іншим. Далі буде використовуватись тільки один сегмент, який представляє цілісну музичну композицію.

У DirectMusic програміст самостійно ініціалізовує COM-систему використовуючи фунцію CoInitialize (NULL). Кожному викликом цієї функції повинен відповідати виклик для завершення роботи COM - CoUninitialize (), який зменшує лічильник посилань використовують COM. Коли значення лічильника стане рівним 0 - COM-система буде видалена з пам'яті. Це підвищує ефективність використання пам'яті і всі COM-об'єкти використовують цю процедуру.

Для створення об'єкту використовується функція CoCreateInstance.

Створення об'єкта виконавця.

Об'єкт виконавця - основний об'єкт, з яким працює програміст. Рекомендується використовувати тільки один об’єкт виконавця. Для створення об'єкта виконавця оголошується екземпляр об'єкта IDirectMusicPerformance8, після чого викликається функція CoCreateInstance, як показано нижче:

```
// Глобальний об'єкт виконавця
IDirectMusicPerformance8 * g_pDMPerformance;
CoCreateInstance (CLSID_DirectMusicPerformance, NULL,
CLSCTX_INPROC, IID_IDirectMusicPerformance8, (void **) &
g_pDMPerformance);
```

Функція CoCreateInstance повертає значення S_OK , якщо виклик завершено успішно. Будь-яке інше значення свідчить про помилку.

Об'єкт виконавця необхідно ініціалізувати. При цьому створюються об'єкти DirectMusic і DirectSound, вони створюють аудіо буфери і встановлюють параметри відтворення. Також встановлюється аудіо-шлях за замовчуванням, по якому відтворюється музика. Стандартні параметри передбачають використання 128 каналів (інструментів), включення стереофонії і ефекту луни (відбиття звуку від об'єктів). Наведемо функцію, яка виконує описані дії:

```
HRESULT IDirectMusicPerformance8:: InitAudio (
IDirectMusic ** ppDirectMusic, // NULL
IDirectSound ** ppDirectSound, // NULL
HWND hWnd, // Дескриптор батьківського вікна
DWORD dwDefaultPathType, // Тип аудіо-шляху за замовчуванням використовується DMUS_APATH_SHARED_ STEREOPLUSREVERB
DWORD dwPChannelCount, // Кількість каналів - використовується 128
DWORD dwFlags, // DMUS_AUDIOF_ALL (дозволити всі музичні можливості)
DMUS_AUDIOPARAMS * pParams); // NULL (структура параметрів)
```

Наведемо приклад виклику функції:

```
/ / G_pDMPerformance = раніше створений об'єкт виконавця
if (FAILED (g_pDMPerformance-> InitAudio (NULL, NULL, hWnd, DMUS_APATH_SHARED_STEREOPLUSREVERB, 128, DMUS_AUDIOF_ALL, NULL)))
{/ / Помилка}
```

Створення об'єкта завантажувача. Наступний етап використання DirectMusic - створення об'єкта завантажувача. Цей об'єкт є системою кешування, прискорює завантаження даних і виконує завантаження необхідних музичних файлів.

Завантажувач представляє об'єкт IDirectMusicLoader8, який створюється за допомогою наступного коду:

```
IDirectMusicLoader8 * g_pDMLoader; / / Глобальний об'єкт завантажувача
CoCreateInstance (CLSID_DirectMusicLoader, NULL, CLSCTX_INPROC, IID_IDirectMusicLoader8, (void **) & g_pDMLoader);
```

Відмітимо, що один додаток може містити тільки один об'єкт IDirectMusicLoader8.

Наступний етап використання завантажувача - вказівки йому, в якому каталозі слід шукати файли. На цей каталог посилаються як на каталог пошуку за замовчуванням. Зазвичай при завантаженні одного музичного файлу, такого як MIDI-файл, встановлення каталогу за замовчуванням не потрібно, оскільки завантажувачу повідомляється повний шлях до файлу. Однак для файлів

рідного формату DirectMusic об'єкт завантажувача повинен знати, де шукати допоміжні файли.

Для встановлення каталогу пошуку за замовчуванням використовується функція IDirectMusicLoader8:: SetSearchDirectory:

```
HRESULT IDirectMusicLoader8:: SetSearchDirectory (REFGUID rguidClass, / / Клас (GUID_DirectMusicallTypes)
WCHAR * pwszPath, / / Шлях до каталогу (широкі символи)
BOOL fClear); / / FALSE - очистити кеш
```

Як видно з прикладу, функції потрібен тільки один параметр - встановлений шлях до каталогу для пошуку. Щоб оголосити рядок широких символів використовується наступний код:

```
WCHAR * Text = L "Testing";
```

Щоб перетворити рядок звичайних символів в рядок широких символів використовується функція mbstowcs:

```
char Text [] = "Roleplaying is fun!"; // Буфер вихідного тексту
WCHAR WText [256]; // Буфер для перетвореного тексту
// Перетворимо 256 символів з джерела в приймач
mbstowcs (WText, Text, 256);
```

Наведемо приклад встановлення поточного каталогу в якості каталогу пошуку за замовчуванням:

```
// G_pDMLoader = раніше ініціалізований об'єкт завантажувача
CHAR strPath [MAX_PATH]; // Поточний шлях
WCHAR wstrPath [MAX_PATH]; // Буфер широких символів
GetCurrentDirectory (MAX_PATH, strPath);
mbstowcs (wstrPath, strPath, MAX_PATH);
if (FAILED (g_pDMLoader -> SetSearchDirectory (GUID_DirectMusicAllTypes, wstrPath, FALSE)))
{/ / Помилка}
```

Робота з музичними сегментами. Об'єкт завантажувача DirectMusic (як показано на рис. 4.26) завантажує музику і дані інструментів та створює об'єкт IDirectMusicSegment8.

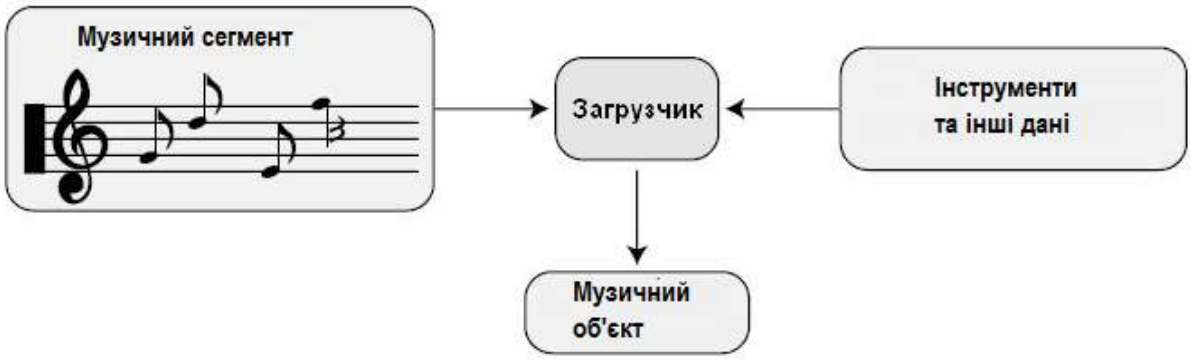

Рисунок 4.26 - Об'єкт завантажувача відповідає за отримання даних, таких як музичні партитури і дані інструментів, необхідні для створення музичного об'єкта

Процес завантаження складається з двох етапів - спочатку завантажується музичний сегмент, що містить ноти для відтворення, а потім музичні інструменти.

Перший етап - ініціалізація об'єкта структури DMUS_OBJECTDESC, що містить інформацію про вміст завантаження (про музику):

```
typedef struct {
DWORD dwSize; // Розмір структури
DWORD dwValidData; // Прапори, що визначають дійсні поля
GUID guidObject; // Унікальний GUID об'єкта
GUID guidClass; // CLSID_DirectMusicSegment
FILETIME ftDate; // Дата останнього редагування об'єкта
DMUS_VERSION vVersion; // Структура, яка містить інформацію про версію
WCHAR wszName [DMUS_MAX_NAME]; // Назва об'єкту
WCHAR wszCategory [DMUS_MAX_CATEGORY]; // Категорія об'єкта
WCHAR wszFileName [DMUS_MAX_FILENAME]; // Ім'я файлу для завантаження
LONGLONG llMemLength; // Розмір даних в пам'яті
LPBYTE pbMemData; // Покажчик на дані в пам'яті
IStream * pStream; // Інтерфейс потоку для завантаження
} DMUS_OBJECTDESC;
```

Більшість полів в DMUS_OBJECTDESC можна ігнорувати. Поле dwValidData зберігає комбінацію прапорів, що повідомляють завантажувачу, які поля в структурі використовуються. Наприклад, якщо ви хочете використовувати wszFilename і guidClass, встановіть відповідні прапори. Список прапорів наведено в таблиці 4.7.

Таблиця 4.7

Прапори dwValidData

Прапор	Опис
DMUS_OBJ_CATEGORY	Дійсно значення wszCategory .

DMUS_OBJ_CLASS	Дійсно значення guidClass .
DMUS_OBJ_DATE	Дійсно значення ftDate .
DMUS_OBJ_FILENAME	Дійсно значення wszFileName .
DMUS_OBJ_FULLPATH	wszFileName містить повний шлях до об'єкта.
DMUS_OBJ_LOADED	Об'єкт уже завантажений.
DMUS_OBJ_MEMORY	Об'єкт в пам'яті. Дійсні значення llMemLength і pbMemData .
DMUS_OBJ_NAME	Дійсно значення wszName .
DMUS_OBJ_OBJECT	Дійсно значення guidObject .
DMUS_OBJ_STREAM	Дійсно значення pStream .
DMUS_OBJ_URL	wszFileName представляє адресу URL.
DMUS_OBJ_VERSION	Дійсно значення vVersion .

Структура DMUS_OBJECTDESC передається функції IDirectMusicLoader8::GetObject, що забезпечує завантаження і розміщення в об'єкті сегмента всіх пов'язаних файлів даних. Наведемо прототип функції:

```
HRESULT IDirectMusicLoader8:: GetObject (
LPDMUS_OBJECTDESC pDesc, // Покажчик на структуру
DMUS_OJBECTDESC REFIID riid, // IID_IDirectMusicSegment8
LPVOID FAR * ppv); / / Покажчик на новий завантажений об'єкт
```

Написаний код дає вказівку функції GetObject використовувати структуру, яка ініціалізувала і завантажила дані в об'єкт IDirectMusicSegment8. Звичайно, спершу потрібно оголосити об'єкт сегмента, щоб мати справу з функцією, яка обробляє завантаження фрагмента музики і повертає об'єкт сегмента (або NULL, якщо виникла помилка).

// G_pDMLoader = раніше ініціалізований об'єкт завантажувача з встановленим каталогом пошуку

```
IDirectMusicSegment8 * LoadSong (char * Filename)
{DMUS_OBJECTDESC dmod;
IDirectMusicSegment8 * pDMSegment;
ZeroMemory (& dmod, sizeof (DMUS_OBJECTDESC));
dmod.dwSize = sizeof (DMUS_OBJECTDESC);
dmod . guidClass = CLSID_DirectMusicSegment;
dmod.dwValidData = DMUS_OBJ_CLASS | DMUS_OBJ_FILENAME |
DMUS_OBJ_FULLPATH;
 mbstowcs (dmod.wszFileName, Filename, MAX_PATH);
if (FAILED (g_pDMLoader-> GetObject
(& dmod, IID_IDirectMusicSegment8, (LPVOID) & pDMSegment)))
return NULL;
// Завантаження завершено
```

return p_DMSegment;}

Завантаження інструментів. При використанні формату MIDI завантажувач DirectMusic встановлює інструменти по замовчуванню

Інструменти називаються модифікаторами (patches), а набір модифікаторів називається DLS-даними інструментів, які включаються до колекції інструментів. Модифікатори нумеруються послідовністю із трьох значень: старшого значущого байта (most significant byte , MSB), молодшого значущого байта (least significant byte , LSB) і номери модифікатора (patch number).

Основні модифікатори MIDI стандартизовані, так що модифікатор з номером 1 (піаніно) завжди буде ставитися до піаніно. Якщо потрібно використовувати новий модифікатор піаніно, досить просто завантажити його з DLS-колекції. DirectMusic поставляється з колекцією інструментів, відповідної специфікації General MIDI створеною компанією Roland.

Для завантаження DLS-колекції необхідно отримати об'єкт IDirectMusicCollection8 через об'єкт завантажувача. Для цього використовується функція IDirectMusicLoader8:: GetObject. Наведемо приклад функції, яка завантажує DLS-колекцію і повертає вказівник на об'єкт завантаженої колекції для подальшої роботи:

```
IDirectMusicCollection8 * LoadDLSCollection (char * Filename) {
DMUS_OBJECTDESC dmod; IDirectMusicCollection8 * pDMCollection;
ZeroMemory (& dmod, sizeof (DMUS_OBJECTDESC));
dmod.dwSize = sizeof (DMUS_OBJECTDESC);
dmod.guidClass = CLSID_DirectMusicCollection;
dmod.dwValidData = DMUS_OBJ_CLASS | DMUS_OBJ_FILENAME | DMUS_OBJ_FULLPATH;
 mbstowcs (dmod.wszFileName, Filename, MAX_PATH);
if (FAILED (g_pDMLoader-> GetObject (& dmod, IID_IDirectMusicCollection8, (void **) pDMCollection))) return NULL; / / повертається покажчик на об'єкт колекції return
IDirectMusicCollection8;}
```

Коли колекція завантажена її ще потрібно призначити сегменту. Це робиться шляхом встановлення розташування сегмента треку за допомогою функції IDirectMusicSegment8:: SetParam:

```
HRESULT IDirectMusicSegment8:: SetParam (
REFGUID rguidType, // GUID встановлюваного параметра
DWORD dwGroupBits, // На який трек має ефект (0xFFFFFFFF)
DWORD dwIndex, // 0
MUSIC_TIME mtTime, // Коли застосовувати зміни - використовуйте 0
void * pParam); // Новий параметр або NULL якщо не потрібно
```

Іноді потрібно використовувати пропоновану за замовчуванням колекцію, що досягається викликом GetObject зі значенням GUID GUID_DefaultGMCollection в поле об’єкта класу:

```
IDirectMusicCollection8 * GetDefaultCollection () {
DMUS_OBJECTDESC mod;
IDirectMusicCollection8 * pDMCollection;
ZeroMemory (& dmod, sizeof (DMUS_OBJECTDESC));
dmod.dwSize = sizeof (DMUS_OBJECTDESC);
dmod.guidObject = GUID_DefaultDMCollection;
dmod.dwValidData = DMUS_OBJ_OBJECT;
if (FAILED (g_pDMLoader- > GetObject (& dmod, IID_IDirectMusicCollection8, (void **) pDMCollection)))
return NULL;
return pDMCollection;}
```

Виклик показаної вище функції GetDefaultCollection створить об'єкт колекції інструментів, що містить DLS-дані інструментів за замовчуванням.

Налаштування для MIDI. Все пісня в пам'яті (з інструментами) і майже готова до використання. Залишилось лише пара проблем, які слід вирішити. По-перше, оскільки система повинна підготувати себе у відповідності зі специфікацією General MIDI, вам необхідно повідомити системі чи є завантаження MIDI-файлом.

Щоб повідомити DirectMusic, що сегмент є MIDI-файлом потрібно встановити параметри сегмента треку функцією IDirectMusicSegment8:: SetParam:

```
pDMSegment-> SetParam (GUID_StandardMidiFile, 0xFFFFFFFF, 0, 0, NULL);
```

Показаний нижче виклик функції SetParam можна помістити у функцію LoadSongFile (після того, як пісня повністю завантажена):

```
if (FAILED (g_pDMLoader-> GetObject (& dmod, IID_IDirectMusicSegment8, (LPVOID) & pDMSegment)))
return NULL; // Завантаження завершено
// Встановлюємо ознаку MIDI-файлу
if (strstr (Filename, ". mid")! = NULL)
pDMSegment -> SetParam (GUID_StandardMidiFile, 0xFFFFFFFF, 0, 0, NULL);
return p_DMSegment;}
```

Налаштування інструментів. Наступний етап підготовки сегмента до відтворення – налаштування даних інструментів шляхом їх завантаження в об'єкт виконавця. Це виконується за допомогою викликуIDirectMusicSegment8:: Download :

*HRESULT IDirectMusicSegment8:: Download (IUnknown * pAudioPath);*

Цей виклик тільки для MIDI-файлів, оскільки він змінює спосіб сприйняття музичної інформації. Єдиний параметр функції - аудіо-шлях для якого завантажуються дані інструментів. У даному випадку це об'єкт виконавця, так що використовуйте наступний код:

if (FAILED (g_pDMSegment-> Download (g_pDMPerformance))) {/ / Помилка}

Після завершення роботи з музичним сегментом, необхідно виконати функцію IDirectMusicSegment8:: Unload , що звільняє дані інструментів. Виконується це після того, як зупиняється відтворення сегмента і завершується робота з ним або коли перемикається колекція інструментів. Виклик ідентичний IDirectMusicSegment8:: Download:

if (FAILED (g_pDMSegment-> Unload (g_pDMPerformance))) {/ / Помилка}

Використання циклів і повторів. Останній крок перед відтворенням - встановлення точок повтору і кількості повторень циклу. Якщо необхідно повторити ритм декілька разів, то треба встановити початкову та кінцеву точки циклу (як показано на рис. 4.27), а потім задати кількість повторень циклу.

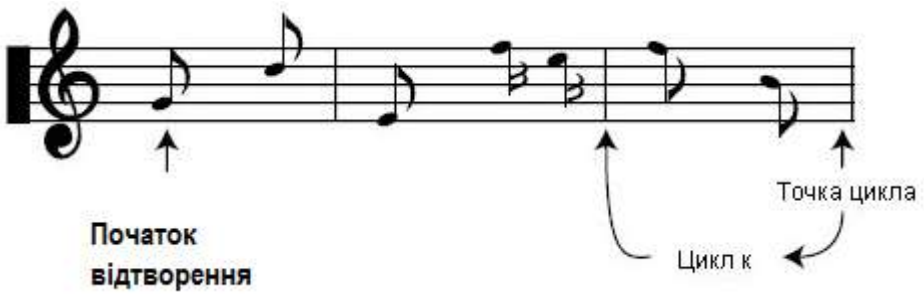

Рисунок 4.27 – Робота з циклами і повторами у DirectMusic

Встановлення точок циклу - це призначення функції IDirectMusicSegment8:: SetLoopPoints :

HRESULT IDirectMusicSegment8:: SetLoopPoints (MUSIC_TIME mtStart, MUSIC_TIME mtEnd);

Зверніть увагу на використання MUSIC_TIME - одиниць виміру часу, що застосовуються в DirectMusic. Це вимір часу, що базується на темпі пісні, а не на таймері.

Після встановлення точок циклу необхідно вказати кількість повторів, за допомогою функції IDirectMusicSegment8:: SetRepeats , у якої всього один параметр - кількість повторень циклу пісні:

pDMSegment-> SetRepeats (0); / / Відтворення пісні один раз (немає циклів)

Відтворення та зупинка сегмента. Об'єкт виконавця відтворює сегмент за допомогою функціїIDirectMusicPerformance8:: PlaySegmentEx :

HRESULT IDirectMusicPerformance8:: PlaySegmentEx (
*IUnknown * pSource, // Відтворений сегмент*
*WCHAR * pwzSegmentName, // NULL - не використовується*
*IUknown * pTransition, // Сегмент переходу - використовуйте NULL*
DWORD dwFlags, // Прапори зміни поведінки
__int64 i64Starttime, // Коли починати відтворення
// 0 - негайно
*IDirectMusicSegmentState ** ppSegmentState, // Покажчик на об'єкт, отримує об'єкт, стану сегменту*
*IUnknown * pFrom, // NULL*
*IUnknown * pAudioPath); // Використовуваний аудіо-шлях або NULL для аудіо-шляху за замовчуванням*

Швидко розпочати відтворення сегмента можна за допомогою наступного фрагмента коду:

if (FAILED (g_pDMPerformance-> PlaySegmentEx (g_pDMSegment, NULL, NULL, 0, 0, NULL, NULL, NULL))) {/ / Помилка}

Щоб зупинити відтворення сегмента використовуйте виклик IDirectMusicPerformance:: Stop :

HRESULT IDirectMusicPerformance8:: Stop (
*IDirectMusicSegment * pSegment, // зупинка сегменту*
*IDirectMusicSegmentState * pSegmentState, // Стан для зупинки*
MUSIC_TIME mtTime, / / Час зупинки (0 - негайно)
DWORD dwFlags); / / Поведінка часу зупинки

Інший метод зупинки відтворення сегмента показано нижче:

if (FAILED (g_pDMPerformance-> Stop (g_pDMSegment, NULL, 0, 0))) {/ / Помилка}

Вивантаження даних сегменту. Після того як сегмент зупинено та завершена робота з ним, необхідно вивантажити дані інструментів:

pDMSegment-> Unload (g_pDMPerformance);

Також потрібно звернутися до завантажувача для звільнення кешованих даних за допомогою виклику IDirectMusicLoader8:: ReleaseObjectByUnknown :

*HRESULT IDirectMusicLoader8:: ReleaseObjectByUnknown (IUnknown * pObject);*

Метод ReleaseObjectByUnknown отримує один параметр - покажчик на об'єкт сегмента. Коли вивантаження проведено можна звільнити COM-об'єкт сегмента. Ось два виклики, які це роблять:

g_pDMLoader-> ReleaseObjectByUnknown (pDMSegment); pDMSegment-> Release ();

Наведемо приклад очищення кеша, яке потрібно виколати після вивантаження:

g_pDMLoader-> ClearCache (GUID_DirectMusicAllTypes);

Дані кеша вивантажуються тільки коли новий сегмент, який буде завантажуватися не потребує кешованої інформації.

Зміна музики. З музикою можна виконувати ряд дій, включаючи зміну рівня гучності, зміну темпу і застосування спец ефектів шляхом використання об'єкта звукового буфера DirectSound. Давайте поглянемо на кожен з методів.

Програміст може змінювати два параметри гучності - загальну гучність об'єкта виконавця (і музичної системи в цілому) та індивідуальну гучність відтворення окремого сегмента. Як показано на рисунку 4.28, кожен сегмент піддається зміні гучності, коли передається об'єкту виконавця. Об'єкт виконавця впливає на загальну гучність.

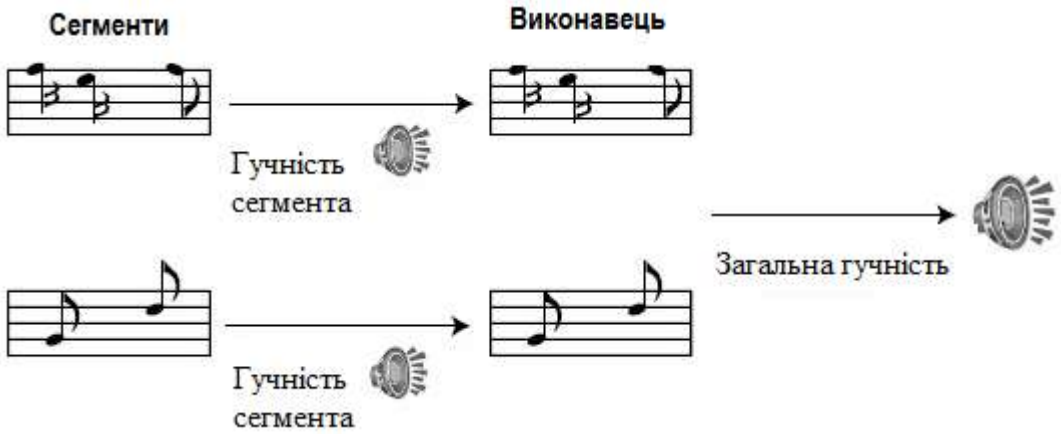

Рисунок 4.28 – Особливості роботи з гучністю

Гучність виконавця (основна гучність) являє собою глобальний параметр і для її встановлення використовується виклик IDirectMusicPerformance8:: SetGlobalParam :

HRESULT IDirectMusicPerformance8:: SetGlobalParam (
REFGUID rguidType, // Встановлюваний глобальний параметр
*void * pParam, // Нове значення параметра*
DWORD dwSize); // Розмір даних параметра

Параметр rguidType - це GUID глобального параметра, який потрібно встановити, в даному випадку GUID_PerfMasterVolume.

Параметр pParam - це рівень гучності, який потрібно встановити. Значення dwSize - це розмір довгого цілого, тобто розмір змінної, яка використовується для зберігання рівня гучності. DirectMusic використовує два макроси з іменами DMUS_VOLUME_MIN (-200 децибел) і DMUS_VOLUME_MAX (+20 децибел), що представляють мінімальний і максимальний рівень гучності відповідно. Використовуючи значення, розташовані між значеннями цих двох макросів можна задати ступінь загасання в децибелах.

Спростити завдання рівня гучності можна створивши формулу, яка дозволить використовувати для завдання рівня гучності відсотки, а не абсолютні значення. Відсотки змінюються в діапазоні від 0 до 100, де 0 представляє відсутність звуку, а 100 - максимально посилений звук.

Ось невеличка функція, яку можна використовувати для визначення основного рівня гучності, задаючи значення у відсотках від 0 до 100:

```
BOOL SetMasterVolume (long Level) {
long Volume, Range; / / Обчислюємо діапазон рівнів гучності і формуємо нове значення
Range = labs (DMUS_VOLUME_MAX - DMUS_VOLUME_MIN); / / 220
Volume = DMUS_VOLUME_MIN + Range / 100 * Level; / / Встановлюємо новий рівень гучності
if (FAILED (g_pDMPerformance-> SetParam (GUID_PerfMasterVolume, & Volume, sizeof (long))))
return FALSE;
return TRUE;}
```

Встановлення гучності музичного сегменту виконується шляхом перехоплення інтерфейсу аудіо-шляху та його подальшого використання для встановлення нової гучності. Отримання покажчика полягає у простому зверненні до наступної функції:

```
HRESULT IDirectMusicPerformance8:: GetDefaultAudioPath (IDirectMusicAudioPath8 ** ppAudioPath);
```

Функція отримує єдиний параметр - покажчик на використовуваний об'єкт IDirectMusicAudioPath8. Коли покажчик на аудіо-шлях отриманий, можна скористатися функцієюIDirectMusicAudioPath8:: SetVolume :

```
HRESULT IDirectMusicAudioPath8:: SetVolume (
long lVolume, // Встановлюваний рівень гучності
DWORD dwDuration); // Час виконання зміни (мілісекунди)
```

Рівень гучності варіюється від -600 (тиша) до 0 (повна гучність). Посилення тут немає. Щоб надмірно не навантажувати процесор, час

виконання зміни має бути дорівнює 0. Присвоєння dwDuration значення 0 також повідомляє музичну систему про те, що гучність треба змінити негайно.

Темп вимірюється в ударах за хвилину і звичайне його значення дорівнює 120. DirectMusic дозволяє змінювати темп різними способами. Найпростіший шлях – налаштувати основний темп виконавця шляхом задання коефіцієнта масштабування. Наприклад, коефіцієнт масштабування 0.5 вдвічі сповільнить темп, а коефіцієнт 2.0 подвоїть його.

Це виконується шляхом встановлення глобального параметра. Наведемо приклад невеликої функції для задання темпу:

```
BOOL SetTempo (long Percent) {
float Tempo;
Tempo = (float) Percent / 100.0f;
 if (FAILED (g_pDMPerformance-> SetGlobalParam (GUID_PerfMasterTempo, (void *) &
Tempo,
sizeof (float)))
return FALSE;
return TRUE;}
```

Єдина проблема тут в тому, що прояв ефекту зміни темпу може зайняти пару секунд через синхронізацію такту. Також необхідно пам’ятати, що SetTempo впливає на загальний темп, і мінятися буде темп усіх відтворюваних сегментів. Завершивши роботу з піснею необхідно повернути нормальне значення темпу (1.0 або 100%).

Останнім у довгому списку музичних можливостей йде захоплення об'єкта звукового буфера DirectSound, використовуваного для синтезування інструментів і музики. Це можна зробити отримавши об'єкт аудіо-шляху та скористатись ним для захоплення інтерфейсу звукового буфера. Наведемо приклад функції для виконання цих дій:

```
HRESULT IDirectMusicAudioPath8:: GetObjectInPath (
DWORD dwPChannel, // DMUS_PCHANNEL_ALL (канали для пошуку)
DWORD dwStage, // DMUS_PATH_BUFFER (етап шляху)
DWORD dwBuffer, // 0 (індекс у ланцюжку буферів)
REFGUID guidObject, //
GUID_NULL (клас об'єкта)
DWORD dwIndex, // 0 (індекс об'єкта в буфері)
REFGUID iidInterface, // GUID бажаного об'єкта
void ** ppObject); / / Покажчик на створюваний об'єкт
```

Щоб захопити об'єкт звукового буфера достатньо вказати його GUID і надати покажчик. Показана нижче функція отримує заданий за замовчуванням шлях з об'єкта виконавця та надає об'єкт IDirectSoundBuffer8 , з яким можна експериментувати:

```
IDirectSoundBuffer8 * GetSoundBuffer () {
```

```
IDirectMusicAudioPath8 * pDMAudioPath;
IDirectSoundBuffer * pDSB;
IDirectSoundBuffer8 * pDSBuffer; // Отримуємо аудіо-шлях за замовчуванням
if (FAILED (g_pDMPerformance-> GetDefaultAudioPath (& pDMAudioPath)))
return NULL; // Створюємо об'єкт IDirectSoundBuffer і звільняємо об'єкт аудіо-шляху
if (FAILED (pDMAudioPath-> GetObjectInPath (DMUS_PCHANNEL_ALL,
DMUS_PATH_BUFFER, 0, GUID_NULL, 0, IID_IDirectSoundBuffer, (LPVOID *) & pDSB))) {
pDMAudioPath-> Release ();
return FALSE;
} pDMAudioPath-> Release (); // запитуємо новий об'єкт звукового буфера і повертаємо
його
if (FAILED (pDSB-> QueryInterface (IID_IDirectSoundBuffer8, (void **) & pDSBuffer))) {
pDSB-> Release ();
return FALSE;} pDSB-> Release ();
return pDSBuffer;}
```

Змінити показану вище функцію для запиту інтерфейсу тривимірного звукового буфера просто, змінивши ідентифікатор інтерфейсу на IID_IDirectSound3DBuffer8.

Контрольні запитання

1. Що таке OpenGL: переваги та недоліки.
2. Види систем координат в OpenGL.
3. Синтаксис команд OpenGL.
4. Етапи ініціалізації OpenGL.
5. Що ви розумієте під проекцією: приклади програмування.
6. Як представляється колір в OpenGL?
7. Назвіть інтерфейси DirectX.
8. Для чого використовується інтерфейс Direct3D?
9. Система координат в Direct3D.
10. Наведіть приклади представлення об’єктів в різних системах координат Direct3D.
11. Як відбувається фарбування полігонів у Direct3D.
12. Які ви знаєте фунції Direct3D.для роботи з матрицями?
13. Особливості зберігання та упорядкування вершин в Direct3D.
14. Особливості роботи з DirectSound.
15. Що таке COM інтерфейси DirectSound. Наведіть приклади.
16. Особливості роботи з форматом файлів в DirectSound.
17. Аудіо буфери та процес мікшування.
18. Наведіть приклади позиціонування.
19. Як відбувається потокове відтворення аудіо в DirectSound.
20. Етапи використання DirectMusic.
21. Робота з музичними сегментами.
22. Як завантажити інструменти в DirectMusic?

Використана література

1. Direct X 9. Осваиваем 3D-пространство: Ален Торн — Санкт-Петербург, НТ Пресс, 2007 г.- 288 с.
2. DirectX 10 - это просто. Программируем графику на C++ (+ CD-ROM): Алексей Попов — Санкт-Петербург, БХВ-Петербург, 2008 г.- 464 с.
3. Managed DirectX 9 с управляемым кодом. Программирование игр и графика (+ CD-ROM): Том Миллер — Санкт-Петербург, КомБук, 2005 г.- 400 с.
4. Графика в формате DirectX 9. Полное руководство по использованию 3D-пространства: Ален Торн — Санкт-Петербург, НТ Пресс, 2007 г.- 288 с.
5. Инструментальные средства программирования и отладки шейдеров в DirectX и OpenGL (+ CD-ROM): Станислав Горнаков — Санкт-Петербург, БХВ-Петербург, 2005 г.- 256 с.
6. OpenGL. Руководство по программированию: М. Ву, Т. Девис, Дж. Нейдер, Д. Шрайнер — Санкт-Петербург, Питер, 2006 г.- 624 с.
7. Графика трехмерной компьютерной игры на основе OpenGL: А. В. Боресков — Москва, Диалог-МИФИ, 2004 г.- 384 с.
8. Инструментальные средства программирования и отладки шейдеров в DirectX и OpenGL (+ CD-ROM): Станислав Горнаков — Санкт-Петербург, БХВ-Петербург, 2005 г.- 256 с.
9. Компьютерная графика. Полигональные модели: Е. В. Шикин, А. В. Боресков — Санкт-Петербург, Диалог-МИФИ, 2005 г.- 464 с.
10. Разработка и отладка шейдеров (+ CD-ROM): Алексей Боресков — Санкт-Петербург, БХВ-Петербург, 2006 г.- 488 с.
11. Расширения OpenGL (+ CD-ROM): Алексей Боресков — Санкт-Петербург, БХВ-Петербург, 2005 г.- 688 с.

ТЕСТОВІ ПИТАННЯ

1. Виберіть найбільш точне визначення терміну «мультимедіа»
   - сукупність технологій інтерактивного оброблення даних, що представляються у формі аудіо, зображення, відео, тексту, анімації, гіпертекст;
   - багатоканальне середовище, що видає інформацію в різноманітних форматах;
   - сучасна інформаційна технологія, що об'єднує за допомогою комп'ютерних засобів графічне та відео зображення, звук і інші спеціальні ефекти ;
   - технологія передачі даних.
2. Термін "мультимедіа" почали використовувати у:
   - 60-ті роки XX століття;
   - 50-ті роки XX століття;
   - 70-ті роки XX століття;
   - 90-ті роки XX століття.
3. Виберіть типи мультимедіа технологій:
   - автономні;
   - телемовні;
   - телекомунікаційні;
   - локальні;
   - глобальні;
4. Основними способами відтворення мультимедіа даних є:
   - нелінійний;
   - лінійний;
   - паралельний;
   - покадровий;
5. Гіпермедіа дозволяє людині брати участь у…
   - процесі відтворення мультимедійних даних;
   - процесі створення мультимедійних даних;
   - процесі редагування мультимедійних даних;
   - процесі стиснення мультимедійних даних;
6. Який із процесів можна класифікувати, як лінійний спосіб відтворення мультимедіа:
   - комп'ютерна гра;
   - читання електронної книги;
   - перегляд флеш анімації;
   - перегляд Інтернет сторінки.
7. Для створення мультимедійних додатків застосовується:
   - прикладне програмне забезпечення;
   - спеціалізоване програмне забезпечення;

- системне програмне забезпечення;
- драйвери.

8. Що з переліченого є складовими мультимедіа:
   - текст;
   - відео;
   - комп’ютерна гра;
   - тактильні відчуття.
9. Набором елементарних точок можна представити:
   - відео кліп;
   - векторне зображення;
   - анімацію;
   - растрове зображення.
10. Параметрами растрового зображення є:
   - глибина кольору;
   - розширення;
   - піксель;
   - роздільна здатність монітора.
11. Примітивними об’єктами векторної графіки є:
   - еліпс;
   - точка;
   - парабола;
   - лінії.
12. До мультимедіа складових входять:
   - зображення: векторні, растрові;
   - аудіо: звукові сигнали, мовні сигнали;
   - відео: аналогове, цифрове;
   - бази даних: реляційні, ієрархічні.
13. Для опису кольору чорно-білого зображення достатньо:
   - одного двійкового розряду;
   - 16 біт на піксель;
   - 4 біти на піксель;
   - 1 байт на піксель.
14. Півтонові зображення – це:
   - зображення з градацією сірого або іншого кольору;
   - зображення з глибиною кольору 16 біт на піксель;
   - зображення з глибиною кольору 24 біти на піксель;
   - зображення з глибиною кольору 32 біти на піксель.
15. Під терміном «аудіо» розуміють:
   - мовний чи звуковий сигнал, записаний на носії інформації;
   - будь-який мовний сигнал;
   - звук, відтворений з носія інформації;
   - шум.

16. Звуковий сигнал – це:
- коливальний рух частинок пружного середовища, що поширюється у вигляді хвиль у газоподібному, рідкому чи твердому середовищі;
- плоска періодична хвиля, що позначає найменшу відстань між точками простору, в яких вона має однакову фазу;
- звукова хвиля, яка періодично змінюється;
- сигнал з частотами від 16 Гц до 20 кГц.

17. В силу своєї специфіки мовні сигнали розглядаються з точок зору:
- акустичної;
- фізіологічної;
- фізичної;
- артикуляційної;
- інтонаційної.

18. Під терміном відео розуміють:
- широкий спектр технологій запису, оброблення, передачі, зберігання й відтворення візуального і аудіовізуального матеріалу;
- динамічну зміну зображення;
- динамічну зміну зображень з відсутністю звукового супроводу;
- фільми, мультиплікацію, анімацію.

19. Відеосигнал характеризується наступними параметрами:
- кількість кадрів в секунду;
- розгортка;
- роздільна здатність;
- кількість каналів;
- якість звукового супроводу;

20. В методиці вимірювання суб'єктивної якості відео присутні етапи:
- вибираються відеопослідовності для використання в тесті;
- вибираються параметри системи вимірювання;
- вибирається метод показу відео й підрахунку результатів виміру;
- проводиться тест;
- повторюється тест;

21. Розгортка – це:
- процес перетворення оптичного зображення в послідовність сигналів;
- дія протилежна до згортки;
- верхня гранична частота мерехтіння, що сприймається людським мозком;
- фізична кількість колонок та рядків пікселів, що створює дисплей.

22. Гіпертекст – це:
- інтерактивний нелінійний документ, окремі вузли якого зв’язані між собою за допомогою гіперзв'язків чи гіперпосилань;
- текст великого розміру;
- розмітка сторінки;
- текст оформлений в таблицях.

23. Гіпертекст характеризується такими специфічними параметрами, які залежать від особливостей побудови системи посилань:
- інтрагіпертекстовість;
- екстрагіпертекстовість;
- супергіпертекстовість;
- ультрагіпертекстовість.

24. Гіпертекст забезпечує функціонування великих обсягів інформації:
- текстової;
- графічної;
- математичної;
- бази даних;
- суб’єктивної.

25. Для створення гіпертекстових документів у Інтернеті розроблено спеціальну мову розмітки гіпертексту:
- HTML;
- JavaXML;
- JavaScript;
- GWBasic.

26. Анімація – це:
- процес створення серії знімків, малюнків, кольорових плям, ляльок або силуетів в окремих фазах руху, за допомогою якого під час показу їх на екрані виникає враження оживлення форм;
- процес перетвореннння кінофільмів в мультиплікації;
- відеоролик на мові SID;
- зображення в форматі TIFF;

27. Відомо такі види анімації:
- графічна;
- об’ємна;
- комп’ютерна;
- фрактальна;
- фігурна.

28. За типом параметрів об’єктів виділяють такі анімаційні технології:
руху
- форми (морфінг);
- кольорова анімація;
- стабілізації;
- контролю координат.

29. Технологія “захоплення руху” – це:
- технологія, у якій об'єкти рухаються або змінюють форму внаслідок аналогічних дій реальними до яких прикріплено датчики;
- класична технологія, у якій об’єкти рухаються динамічно;
- технологія в якій будь-який рух об’єктів забороняється;

- технологія у якій датчики пересуваються на об’єктах.

30. Текст – це:
- будь-яка інформація, зображена за допомогою символів клавіатури комп'ютера;
- сукупність букв;
- класичне представлення неграфічної інформації;
- інформація записана на будь-якому носії даних.

31. Характеристики тексту:
- кернінг;
- харкінг;
- інтерліньяж;
- абзац;
- фішінг.

32. Шрифт – це:
- повний комплект друкарських літер певного типу й рисунку, необхідний для набору якого-небудь тексту;
- теж що і текст;
- міжрядковий інтервал;
- міжбуквенний інтервал.

33. Шрифти характеризуються наступними параметрами:
- гарнітурою;
- нахилом;
- насиченістю;
- змістом;
- виглядом.

34. Інтерактивність – це:
- безпосередня взаємодія користувача з персональним комп’ютером, що може відображатись у вигляді запиту або діалогу;
- період часу між діяльністю деякої системи;
- зв’язки між чужорідними об’єктами;
- безпосередня взаємодія користувача з мережею Інтернет.

35. До часткової інтерактивності належать технології, які забезпечують збереження інформації у структурованому вигляді:
- банки даних;
- бази даних;
- гіпертекст;
- посилання;
- псевдографіка.

36. До повної інтерактивності належать технології:
- які забезпечують прямий доступ до великих обсягів інформації;
- які забезпечують прямий доступ до малих обсягів інформації;
- які забезпечують непрямий доступ до великих обсягів інформації;

- які забезпечують непрямий доступ до малих обсягів інформації.

37. Інтерактивність об’єднує такі форми комунікації за допомогою комп’ютера:
- електронну пошту;
- телеконференцзв’язок;
- синхронний та асинхронний зв’язок;
- діалог;
- факсимільний зв'язок.

38. Серед напрямів та галузей використання мультимедіа технологій можна виділити:
- народне господарство;
- маркетинг і реклама;
- системи для комп’ютерного моделювання;
- системи орієнтування;
- підводне плавання.

39.Зображення зберігається в цифровій запам’ятовуючій системі як:
- файл представлений прямокутною таблицею пік селів;
- файл з набором графічних чисел;
- файл з списком бітів;
- файл з векторами.

40.При виконанні за допомогою комп'ютера будь-яких операцій над зображеннями необхідно мати опис зображення, який представлений у вигляді…
- чисел;
- букв;
- знаків і букв;
- символів.

41.Кольорова модель – це…
- це модель конкретизованої класифікації гами світлових кольорів прийнятних для людини, котра дає можливість класифікувати конкретний колір для подальшої можливості його відтворення;
- це модель набору кольорів та потокових значень;
- це набір світлових кольорів, які можна відтворити в системі команд ПК;
- це модель класифікації гами колорів та потокових значень.

42.Колір є важливим засобом (чого)?
- підсилення враження при сприйнятті зображень і підвищення їх інформаційної насиченості;
- підсилення яскравості зображень;
- відтворення зображення в уяві людини;
- візуалізації об’єктів.

43.Найбільш простою і природною моделлю представлення кольору є…
- CMYK;
- CMY;

- RGB;
- RBG.

44.Кількість градацій кожного каналу залежить від:
- від кількості кольорів;
- бітового значення RGB;
- розміру RGB;
- байтового значення.

45.Які переваги RGB?
- апаратна сумісність із графічними пристроями відтворення зображень;
- велика кольорова гамма;
- невеликий розмір файлу;
- високий коефіцієнт стиснення.

46.Що з перераховоного відноситься до графічних моделей?
- RGB;
- CMYK;
- SDD;
- HSB;
- PSD;
- JPEG.

47.Які недоліки моделі RGB?
- кольори на екранах графічних пристроїв можуть відрізнятися від отриманих при виділені кольором;
- існує взаємозалежність колірних каналів;
- великий розмір файлу;
- велика кольорова гама.

48.Сукупністю скількох чисел описується колір в моделі CMYK:
- 3;
- 4;
- 5;
- 64.

49.Скількома координатами визначаються кольори у моделі HSB?
- 2;
- 6;
- 3;
- 1.

50.Колірним тоном або відтінком називається -
- спектрально-чистий колір паралельний осі двох кольорів;
- спектрально-чистий колір певної довжини хвилі;
- спектрально-темний колір певної довжини хвилі;
- візуалізована проекція зображення.

51.Які з перелічених форматів є графічними?
- растрові формати;

- формати сцени;
- аудіо формати;
- текстові формати.

52.Яки формати файлів відносяться до графічних?
- BMP;
- JPEG;
- MPEG;
- GOF;
- PhotoDVD.

53.Формат GIF це -
- 8-бітний растровий графічний формат, що розроблявся для мереж з низькими швидкостями передачі даних;
- 16-бітний растровий графічний формат, що розроблявся для мереж з високими швидкостями передачі даних;
- 8-бітний векторний графічний формат, що розроблявся для мереж з низькими швидкостями передачі даних;
- 8-бітний формат рухомих векторних зображень.

54.Формат JPEG це -
- векторний формат зберігання графічної інформації, який використовує алгоритм Зіва-Лемпеля-Велча;
- растровий формат зберігання графічної інформації, який використовує алгоритми стиснення з втратами, призначений для зменшення розмірів файлів, що мають плавні переходи кольорових тонів і відтінків;
- растровий формат зберігання аудіо інформації;
- немає правильної відповіді.

55.Як розшифровується формат PNG?
- Protected Native Graphics;
- Portable Network Graphics;
- Portable Netdot Graphics;
- Portable Network Gradients.

56.Для чого призначений формат ICO?
- для запису дисків;
- для відтворення системних зображень;
- для зберігання високо роздільної графіки;
- для зберігання значків файлів.

57.Хто є розробником формату TIFF (Tagged Image File Format) ?
- Adobe;
- Microsoft;
- Apple;
- Abobe.

58.Найпоширенішим растровим форматом є…
- BMP;

- RAR;
- DjVu;
- BMP2000.

59.PCX це -
- апаратно-залежний графічний формат;
- немає вірних відповідей;
- апаратно-незалежний растровий формат, що використовується графічним редактором Paint;
- апаратно-залежний растровий формат, що використовується графічним редактором Paintbrush.

60.Який формат описує як векторні, так і растрові зображення на мові PostScript фірми Adobe?
- EPS;
- VCD;
- CDR;
- BMP.

61.Формат CGM це -
- формат який підтримує векторну і растрову графіку з використанням повної палітри (16 млн. кольорів) та палітри зі змінною кількістю кольорів;
- формат кольорів;
- формат збереження даних;
- формат векторної графіки з моделлю кольору RGB.

62.Який з графічних форматів є одним з найпотужніших форматів за можливостями зберігання растрової графічної інформації?
- PSD;
- CDR;
- AI;
- JPEG2000.

63.Якою фірмою був розроблений формат PhotoCD?
- Kodak;
- Sony;
- Honda;
- Nikon.

64.Для чого використовують формат PhotoCD ?
- для запису диску з фото;
- для зберігання цифрових растрових зображень високої якості;
- для зберігання монотонних цифрових зображень;
- для створення мультимедійних презентацій із цифрових зображень.

65.Основний формат векторного графічного редактора CorelDRAW є…
- CDR;
- CDRS;

- CDE;
- PSD.

66.Яке з перелічених визначень найбільш повно характеризує термін «Аудіозапис»?
- процес запису звуку на диск, платівку тощо;
- процес запису звукової або мовної інформації з метою її збереження та подальшого відтворення;
- процес запису звукової або мовної інформації з метою її розповсюдження;
- всі перечисленні відповіді.

67.Якими двома способами здійснюється відтворення аудіозапису?
- акустичним і електромеханічним;
- прямим і непрямим;
- внутрішнім і зовнішнім;
- акустичним і електроакустичним.

68.Як розшифровується АЦП?
- автономний центр процесора;
- аналого-цитатний перетворювач;
- аналого-цифровий перетворювач;
- аналого-цифровий процесор;

69.Що таке механічний аудіозапис?
- система запису аудіо за допомогою зміни форми носія при механічній дії на нього;
- система запису аудіо за допомогою механічного плеєра;
- система запису аудіо за допомогою зміни форми формату аудіо;
- відсутня правильна відповідь.

70.Які переваги механічного запису?
- масове тиражування грамплатівок, їх відносна дешевизна і простота звернення;
- висока якість звуку;
- невисокі затрати процесорної пам’яті;
- невеликий розмір файлу.

71.Як здійснюється цифровий аудіозапис?
- аудіо сигнал перетворюється в числово-знакову форму;
- аудіо сигнал перетворюється в цифрову форму, що виконується шляхом вимірювання миттєвих значень його амплітуди через рівні проміжки часу і представленні отриманих значень, у вигляді послідовності чисел;
- аудіо хвиля перетворюється в масив десяткових чисел та зиписується у файл певної структури;
- немає вірних відповідей.

72.Що таке аудіоносії?
- музичний програвач;

- це носії інформації, що слугують для зберігання звукових або мовних сигналів з метою наступного відтворення;
- грамплатівка;
- носій інформації на якому зберігається мультимедійна інформації.

73.Чим DVD відрізняється від компакт-диску?
- об’ємом;
- можливістю багатоканального запису;
- формою.

74.Якою компанію вперше було розроблено міні-диск?
- Toshiba;
- ZAZ;
- Philips;
- Sony.

75.Який диск використовує формат DSD для запису?
- DVD;
- SACD;
- MiniDisc;
- CD.

76.Яки з перерахованих нижче пристроїв не відносяться до аудіоносіїв?
- цифрові диктофони;
- mp3-плеєри;
- пристрої для читання аудіо книг;
- стільникові телефони;
- всі відносяться до аудіоносіїв.

77.Формат Ogg Vorbis це -
- вільний формат стиснення аудіо даних;
- формат не стисненого аудіо;
- формат стиснення відео даних;
- немає правильної відповіді;

78.Як розшифровується формат WMA?
- Word Master Audio;
- Winballs Media Audio;
- Windows Magic Audio
- Windows Media Audio.

79.Який формат використовується для зберігання аудіо високої якості?
- WAV;
- AAC;
- OGG;
- WAT.

80.Відеоматеріали можуть бути…
- аналоговими та цифровими;
- цифровими та оптичними;

- аналоговими та оптичними;
- цифровими, аналоговими та оптичними.

81.Типи носіїв відео матеріалів:
- аналогові відеокасети;
- грамплатівки;
- аудіокасети;
- цифрові оптичні дискові носії.

82.VHS – це…
- напоширеніший формат аналогового запису відеокасет розроблений компанією JVC;
- напоширеніший формат цифрового запису відеокасет розроблений компанією JVC;
- напоширеніший формат цифрового запису відеокасет розроблений компанією Sony;
- напоширеніший формат аналогового запису відеокасет розроблений компанією Sony.

83.Основні формати аналогових відеокасет…
- VHS;
- VHS-K;
- Hi8;
- maxiDV;
- Digital7.

84.Основні формати цифрових відеокасет це…
- VHS-D;
- HD DVD;
- miniDV;
- Blu-ray Disc.

85.Основні формати цифоровиx оптичних дискових носіїв …
- Blu-ray Disc;
- Digital8;
- miniDV;
- Digital7.

86.Blu-ray Disc — це…
- чергове покоління формату оптичних дисків, що використовується для зберігання; відео високої чіткості і даних з підвищеною щільністю;
- аналоговий носій, що використовується для зберігання відео високої чіткості і даних з підвищеною щільністю;
- перший комерційний оптичний носій даних із записаним відеоматеріалом для домашнього перегляду;
- цифровий напівпрофесійний формат відео, створений за рахунок спрощення та здешевлення професійного формату DV.

87.Синій лазер для запису інформаціїї використовують у таких форматах…

- AAC;
- Laserdisc;
- Blu-ray Disc;
- miniDVD.

88.Мультимедійний контейнер — це …
- формат файлів, що може містити дані різних типів, стиснених різними кодеками і дозволяє зберігати аудіо, відео і текстову інформацію в єдиному файлі;
- формат файлів, що може містити дані одного типу, які стиснені одним кодеком і дозволяє зберігати аудіо, відео і текстову інформацію в єдиному файлі;
- набір програм для запису і читання мультимедійних файлів;
- послідовності з підмножини знаків, що включає тільки друковані знаки (літери, цифри, розділові знаки) і деякі керуючі знаки (пропуски, табуляції, переведення рядка).

89.ASF …
- розроблений фірмою Microsoft формат файлів, що містять потокове аудіо і відео;
- RIFF-медіаконтейнер, вперше використаний Microsoft в 1992 році;
- проект, спрямований на створення відкритого, гнучкого, кросплатформеного формату мультимедійного контейнера та набору інструментів і бібліотек для роботи з даними;
- формат потокового мовлення, який належить фірмі «RealNetworks Products and Services».

90.AVI — це …
- медіаконтейнер, вперше використаний Microsoft в 1992 році;
- формат контейнера який інкапсулює пакети елементарних потоків та інших даних;
- формат даних створений для «мобільного відео»;
- медіаконтейнер, який використовується для передачі відео через Інтернет;

91.Розширення фалів Matroska …
- .mkv;
- .ska;
- .mka;
- .tro.

92.Matroska — це …
- проект, спрямований на створення відкритого, гнучкого, кросплатформеного формату мультимедійного контейнера та набору інструментів і бібліотек для роботи з даними;
- формат файлів, медіаконтейнер, який використовується для передачі відео через Інтернет;
- формат даних створений для «мобільного відео»;

- пропрієтарний формат потокового мовлення, який належить фірмі «RealNetworks Products and Services».

93.Основна перевега формату RealMedia …
- забезпечує прийнятну якість зображення та розбірливість мови при наднизьких бітрейта відео потоку;
- містить дані, стиснуті з використанням різних комбінацій кодеків, що дозволяє синхронно відтворювати відео зі звуком;
- дає змогу здійснювати довільне стиснення звуку;
- можливість мультиплексування аудіо і відео даних і синхронізація їх виходу.

94.FLV — це …
- формат файлів, медіаконтейнер, який використовується для передачі відео через Інтернет;
- формат контейнера який інкапсулює пакети елементарних потоків текстових даних;
- формат даних створений для «мобільного аудіо»;
- пропрієтарний формат потокового мовлення, який належить фірмі «RealNetworks Products and Services»;

95.В 3GP аудіо зберігається у форматах…
- AMR-NB;
- WMA;
- MP3;
- Ogg Vorbis;
- AAC-LC.

96.3GP — це…
- формат даних створений для «мобільного відео»;
- формат контейнера який інкапсулює пакети елементарних потоків та інших даних;
- RIFF-медіаконтейнер, вперше використаний Microsoft в 1992 році;
- формат файлів, медіаконтейнер, який використовується для передачі відео через Інтернет.

97.Формати, які використовуються в мережі Інтернет для потокового відео це …
- FLV;
- AVI;
- WAV;
- Matroska;
- RealMedia.

98.Гіпертекстовий документ - це …
- файл (текст, графічне зображення чи інший фрагмент інформації), що має в своїй структурі посилання на інші файли (документи);
- файл, що може містити дані різних типів, стиснених різними кодеками і дозволяє зберігати аудіо, відео і текстову інформацію в єдиному файлі;

- файл для зберігання та демонстрації тривимірної інтерактивної векторної графіки, що використовується в Інтернет;
- файли анімованих зображень.

99. Найпоширенішим форматом зберігання та представлення гіпертекстових документів є:
- C++;
- HTML;
- Java;
- JavaScript.

100. У HTML гіпертекстові посилання …
- вбудовані в тіло документа і зберігаються як його частина;
- записані в окремий файл і при необхідності читаються з файлу;
- зберігаються в базі даних на стороні сервера;
- не використовуються.

101. Для перегляду документу відтвореного за правилами HTML-розмітки використовується…
- Microsoft Visual Studio;
- Mozilla Firefox;
- Builder C++;
- Internet Explorer;
- Текстовий редактор.

102. Формат SGML визначає…
- мову розмітки для документів;
- формат мультимедійного файлу;
- шаблон мультимедійного файлу;
- тип документу.

103. Прикладами мов, основаних на XML є:
- MPASM;
- C++;
- XHTML;
- Java;
- Matlab.

104. Формати зберігання комп'ютерної анімації…
- MOV;
- Matroska;
- SWF;
- AVI.

105. VRML - це…
- стандартний формат файлів для зберігання та демонстрації тривимірної інтерактивної векторної графіки, що використовується в Інтернет;
- формат графічних файлів для створення анімованих зображень;
- специфікація основаної на XML мови розмітки та формат файлів для двовимірної векторної графіки, як статичної, так і анімованої та інтерактивної;
- формат Flash-файлів.

106. SWF - це…
- формат файлів для зберігання тривимірної інтерактивної векторної графіки;
- формат графічних файлів для створення растрових зображень;
- специфікація основаної на XML мови розмітки документів;
- формат Flash-файлів (анімації, ігор та інтерактивних додатків).

107. Комп'ютерна анімація зберігається…
- в універсальних графічних файлах у вигляді набору незалежних зображень, або в спеціалізованих файлах відповідних пакетів анімації;
- в мультимедійних контейнерах у вигляді набору відео файлів;
- в гіпертекстових документах, у вигляді посилань правил для побудови анімації;
- усі відповіді вірні.

108. Текстовими даними називаються…
- послідовності з підмножини знаків, що включає тільки друковані знаки (літери, цифри, розділові знаки) і деякі керуючі знаки (пропуски, табуляції, переведення рядка);
- файл (текст, графічне зображення чи інший фрагмент інформації), що має в своїй структурі посилання на інші файли (документи);
- набір команд, які виконуються послідовно;
- послідовність аудіо або відео сигналів.

109. DOC - це…
- двійковий текстовий формат, розроблений компанією Microsoft у 1989 році на платформі IBM PC для текстового процесору Microsoft Word;
- відкритий формат файлів документів для зберігання й обміну офісними документами
- вільний кроссплатформовий формат зберігання розмічених текстових документів;
- міжнародний стандарт формату файлів для електронних документів, такі як електронні таблиці, діаграми, презентації та текстові документи, що базується на XML.

110. INI-файл може містити:
- порожні рядки;
- керуючі структуру XML;
- гіперпосилання;
- коментарі.

111. FictionBook - це…
- формат подання електронних версій книг у вигляді XML-документів, де кожен елемент книги описується своїми тегами;
- відкритий формат електронних версій книг;
- формат для зберігання і відображення текстових даних;
- вільний кросплатформовий формат зберігання розмічених текстових документів.

112. BbeB - це…
- формат для зберігання і відображення текстових даних, розроблений компанією Sony;

- формат подання електронних версій книг у вигляді XML-документів, де кожен елемент книги описується своїми тегами;
- відкритий формат електронних версій книг, розроблений Міжнародним форумом з цифровим публікацій IDPF;
- файл конфігурації, що містить дані налаштувань для Microsoft Windows, Windows NT та інших застосунків.

113. Розрізняють методи стиснення даних:
- з втратами;
- без втрат;
- вінрарні;
- потокові;
- фіксовані.

114. Стиснення без втрат найчастіше використовується для даних:
- графічних;
- відео;
- аудіо;
- потокової передач;
- текстових.

115. Перевага методів стиснення із втратами над методами стиснення без втрат полягає в тому, що:
- перші забезпечують набагато кращий коефіцієнт стиснення, задовольняючи при цьому поставлені вимоги;
- другі забезпечують набагато кращий коефіцієнт стиснення, задовольняючи при цьому поставлені вимоги;
- перші забезпечують набагато гірший коефіцієнт стиснення, задовольняючи при цьому поставлені вимоги;
- другі забезпечують набагато гірший коефіцієнт стиснення, задовольняючи при цьому поставлені вимоги.

116. Стиснення даних- це:
- процедура перетворення даних, яка проводиться з метою зменшення їх розміру чи об’єму;
- метод перетворення даних;
- процес конвертації даних з одного формату в інший;
- використання архіватора WinRAR та подібних.

117. Особливостями зображення, отриманого цифровою фотокамерою є:
- розширення 2048×1024;
- передача кольору 24 біта/піксель;
- розмір приблизно 6 мегабайт;
- якість гірша, ніж на звичайній фотографії 10×15;
- формат BMP.

118. До основних алгоритмів стиснення зображень відносять:
- кодування довжин серій;
- алгоритм Хаффмана;
- кодування торсіонними полями;
- алгоритм Форда-Фалкерсона.

119. Для забезпечення кращої якості зображення, отриманого цифровою фотокамерою, потрібно:
   - в 5-6 разів збільшувати розширення;
   - використовувати програму IrfanView;
   - в 5-6 разів зменшувати розширення;
   - використовувати програму Adobe Photoshop.
120. Перспективні алгоритми стиснення зображень:
   - дають змогу отримати якісне зображення при відносно невеликому розмірі файла;
   - дають змогу отримати якісне зображення при великому розмірі файла;
   - зберігають зображення в форматі JPEG;
   - дають змогу отримати якісне зображення в форматі JPEG.
121. Алгоритм кодування довжин серій також називають:
   - кодування зворотів;
   - run-length encoding;
   - RLE;
   - алгоритм Фібоначчі;
   - алгоритм Хаффмана.
122. Алгорититм кодування довжин серій неефективні для графічних зображень у форматах:
   - PCX;
   - BMP;
   - TGA;
   - PDF;
   - JPEG.
123. Результатом кодування по алгоритму RLE (run-length encoding)вхідного рядка «ABCABCABCABCDDEFFFFFFFF» буде:
   - 1A1B1C1A1B1C1A1B1C1A1B1C2D1E8F;
   - 1A2B1C1A1B1C1A2B1C1A2B1C2D1E8F;
   - 2A1B1C7A1B1C1A1B1C1A1B1C2D1E7F;
   - 1A1B5C1A1B1C1A1B1C1A1B1C2D1E6F.
124. Недоліком алгориму RLE (run-length encoding) є:
   - низька пристосованість до розповсюджених типів файлів;
   - низька швидкодія;
   - потреба в великій кількості пам’яті;
   - складність та незручність у використанні.
125. Алгоритм Хаффман складається з таких основних етапів:
   - побудова оптимального кодового дерева;
   - побудова відображення код-символ на основі побудованого дерева
   - стиснення вхідної послідовності символів;
   - стиснення вихідної послідовності символів;
   - побудова результуючих даних.
126. В таблиці кодування Хаффмана відсутні такі колонки даних:
   - символ;
   - частота;

- вхідне кодування;
- вихідне кодування;
- результат.

127. Таблиця кодування Хаффмана дає змогу розмістити символи алфавіту на:
- відповідних вершинах дерева Хаффмана;
- результуючу таблицю;
- вихідний файл;
- вихідне кодування даних з відповідною частотою.

128. Алгоритм Хаффмана має мінімальну надлишковість за умови, що:
- кожен символ кодується окремим ланцюжком в алфавіті {0,1};
- кожен символ кодується окремим ланцюжком в алфавіті {a,b,c,..,z};
- кожен символ додається в вершини дерева Хаффмана;
- розмір вхідних даних достатньо великий і містить часті їх повторення.

129. Алгоритм Лемпеля – Зіва – Велча використовується у графічних форматах:
- GIF;
- TIFF;
- ICO;
- BMP;
- PSD.

130. У процесі стиснення в алгоритмі Лемпеля – Зіва – Велча якщо шукана стрічка є, то ми зчитуємо наступний символ, а якщо стрічки немає, то:
- ми заносимо в потік код для попередньо знайденої стрічки, заносимо стрічку в таблицю і починаємо пошук знову;
- припиняємо пошук;
- заносимо стрічку в таблицю і починаємо пошук знову;
- ми заносимо в потік код для попередньо знайденої стрічки і продовжуємо пошук.

131. Виділяють такі класи алгоритмів стиснення мовних сигналів:
- стиснення форми мовного сигналу;
- стиснення параметрів мовного тракту людини;
- гібридне стиснення;
- стиснення такту сигналу;
- стиснення частоти сигналу.

132. Прикладами алгоритмів, які належать до класу стиснення мовних сигналів є:
- PCM;
- DPCM;
- LPC;
- MBE.

133. До класу стиснення параметрів мовного тракту людини відносять алгоритми:
- LPC;
- MBE;
- ATC;
- PCM.

134. До класу гібридного стиснення відносять алгоритм:
- MPLPC;

- АТС;
- РСМ;
- МВЕ.

135. Завданням будь-якого алгоритму стиснення аудіо сигналів є:
- отримання цифрової послідовності відліків, яка вимагає мінімальної швидкості передачі і з якої декомпресор зможе відновити аудіо сигнал з мінімальними втратами;
- отримання цифрової послідовності відліків, яка вимагає мінімальної швидкості передачі і з якої декомпресор зможе відновити аудіо сигнал з максимальними втратами;
- отримання цифрової послідовності відліків, яка вимагає максимальної швидкості передачі і з якої декомпресор зможе відновити аудіо сигнал з максимальними втратами;
- отримання цифрової послідовності відліків, яка вимагає максимальної швидкості передачі і з якої декомпресор зможе відновити аудіо сигнал з мінімальними втратами.

136. До класифікаційних ознак DPCM можна віднести:
- наявність блоку лінійного прогнозування авторегресійних послідовностей;
- використання багаторівневого квантувальника;
- використання однорівневого квантувальника;
- використанням однорозрядного квантувальника.

137. Перевагами алгоритмів класу стиснення форми мовного сигналу є:
- стійкість до широкого діапазону характеристик джерела сигналу;
- простота реалізації;
- сумісність;
- підтримка форматів збереження.

138. Найкращі характеристики алгоритмів класу стиснення форми мовного сигналу спостерігаються при:
- збільшенні числа частотних піддіапазонів;
- динамічній зміні кількості біт на вибірку від одного піддіапазону до іншого;
- підсумуванні сигналу;
- зменшенні шумів;
- зміні рівня сигналу.

139. Найпростішим алгоритмом класу стиснення форми МС є:
- імпульсно-кодова модуляція (PCM);
- диференціальна ІКМ (DPCM);
- дельта-модуляція (DM);
- адаптивно-диференціальна імпульсно-кодова модуляція (ADPCM).

140. Частковий випадок DPCM з використанням однорозрядного квантувальника представляє собою:
- дельта-модуляція (DM);
- імпульсно-кодова модуляція (PCM);
- диференціальна ІКМ (DPCM);
- адаптивно-диференціальна імпульсно-кодова модуляція (ADPCM).

141. При вокодерному стисненні мовний тракт людини представляється:

- нелінійним фільтром із змінними в часі параметрами, що збуджується джерелом білого шуму;
- спектром із змінними в часі параметрами, що збуджується джерелом білого шуму;
- чорним ящиком із змінними в часі параметрами, що збуджується джерелом чорного шуму;
- генератором білого шуму.

142. За принципами аналізу та синтезу мовного сигналу, вокодерами є:
- смугові;
- формантні;
- базисні;
- дуплексні.

143. Робота вокодерів базується на:
- моделі джерела, з якого отримується інформація про параметри мовного сигналу;
- кодах параметрів джерела мовного сигналу;
- способах знаходження моделі системи та її параметрів;
- нелінійному фільтрі, який формує спектр мовного сигналу.

144. Вокодери, що базуються на принципах алгоритмів лінійного прогнозування представляють мовний тракт людини:
- лінійним фільтром з безперервно імпульсною перехідною характеристикою;
- нелінійним фільтром з безперервно імпульсною перехідною характеристикою;
- нелінійним фільтром, який формує спектр мовного сигналу;
- нелінійним фільтром, який формує спектр мовного сигналу.

145. В алгоритмах гібридного стиснення при стисненні мовних сигналів не використовується:
- замаскований слуховий шум;
- слухова частота розширення;
- слухова фаза нечутливості;
- відкритий слуховий шум.

146. Головним недоліком кодеків типу CELP є:
- висока часова складність;
- апаратна складність;
- алгоритмічна несумісність;
- невелика швидкість;
- несумісність.

147. Алгоритми гібридного класу стиснених мовного сигналу рекомендується використовувати в:
- спеціалізованих системах зв’язку;
- домашніх акустичних системах;
- професійних студіях звукозапису;
- телефонії.

148. Для одержання стисненого мовного сигналу кодеки з регулярним імпульсним збудженням (RPE) використовують:

- лінійний прогнозувальник, який покращує показник кореляції між відліками вхідного сигналу;
- нелінійний вокодер, який покращує показник кореляції між відліками вхідного сигналу;
- послідовність з чотирьох - шести коротких імпульсів;
- послідовність з чотирьох - шести довгих імпульсів.

149. У кодеках з багатоімпульсним збудженням (MPE) в якості сигналу збудження використовується:
- послідовність з чотирьох - шести коротких імпульсів;
- лінійний прогнозувальник, який покращує показник кореляції між відліками вхідного сигналу;
- нелінійний прогнозувальник, який покращує показник кореляції між відліками вхідного сигналу.

150. У алгоритмах класу гібридного стиснення, мовний сигнал дискретизується з метою:
- одержання необхідних параметрів мови;
- безпосереднього стиснення висоти;
- синтезу фрагменту сигналу;
- досягнення необхідного збігу.

Навчально-методичне видання

Програмне забезпечення мультимедіа

Навчальний посібник

Відповідальний за випуск:
Дивак М. П., д. т. н., професор, декан факультету комп'ютерних інформаційних технологій

Підписано до друку 12.06.2011 р.
Формат 60х84/16. Папір офсетний.
Друк офсетний. Зам. № 2-1389
Умов.-друк. арк. 12,42.
Тираж 100 прим.

Видавництво Тернопільського національного
економічного університету "Економічна думка"
46000, Тернопіль, вул. Львівська, 3, тел. 43-22-18
E-mail: edition@tane.edu.ua

Свідоцтво про внесення суб'єкта видавничої справи
до державного реєстру видавців ДК № 3467 від 23.04.2009 р.

Віддруковано ФО-П Шпак В. Б.
Свідоцтво про державну реєстрацію В02 № 924434 від 11.12.2006 р.
Свідоцтво платника податку: Серія Е № 897220
м. Тернопіль, вул. Просвіти, 6.
тел. 8 097 299 38 99
E-mail: tooums@ukr.net