ЗМІСТ

*Змн*.

*Лист*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*6*

08-23.БДР.002.00.000.ПЗ

*Розроб.*

*Гайдучок М.А*

*Перевір.*

*Мартинюк Т.Б.*

*Реценз.*

*Месюра В.І.*

*Н. Контр.*

*Дзюбенко В.В.*

*Затверд.*

*Мартинюк Т.Б.*

Розробка комплексного мультимедійного забезпечення для створення відеоуроків у системі дистанційного навчання

*Літ.*

*Акрушів*

*79*

*ВНТУ, 1КІ-12Б*

[ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 8](#_Toc453623512)

[ВСТУП 9](#_Toc453623513)

[1 ОСНОВИ ДИСТАНЦІЙНОГО НАВЧАННЯ 11](#_Toc453623514)

[1.1 Мета і задачі системи дистанційної освіти 11](#_Toc453623515)

[1.2 Суть дистанційного навчання та його переваги 16](#_Toc453623516)

[1.3 Особливості сприйняття інформації. Відеоуроки 21](#_Toc453623517)

[1.3.1 Аудіовізуальні засоби навчання 22](#_Toc453623518)

[1.3.2 Поняття відеонавчання 25](#_Toc453623519)

[1.3.3 Переваги відеоуроків в процесі навчання 27](#_Toc453623520)

[1.4 Методи реалізації відеуроків. 29](#_Toc453623521)

[2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ 31](#_Toc453623522)

[2.2 Основи проектування програмного забезпечення. UML 31](#_Toc453623523)

[2.3 Діаграма варіантів використання (use case diagram) 33](#_Toc453623524)

[2.4 Основні компоненти. Діаграма компонентів (component diagram) 37](#_Toc453623525)

[2.3.1 Головне вікно 39](#_Toc453623526)

[2.3.2 Механізм FBO 41](#_Toc453623527)

[2.3.3 Система ефектів 42](#_Toc453623528)

[2.3.4 Система відображення 42](#_Toc453623529)

[2.3.5 Система управління блоками 43](#_Toc453623530)

[2.5 Діаграма класів (classes diagram) 43](#_Toc453623531)

[2.5 Діаграма розгортання (deployment diagram) 45](#_Toc453623532)

[3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ 47](#_Toc453623533)

[3.1 Обґрунтування вибору мови програмування 47](#_Toc453623534)

[3.1.1 Вибір мови програмування 47](#_Toc453623535)

[3.1.2 Вибір бібліотеки класів 49](#_Toc453623536)

[3.2 Обґрунтування вибору бібліотеки та кодека обробки аудіо/відео 52](#_Toc453623537)

[3.3 Обґрунтування вибору бібліотеки виводу зображення 55](#_Toc453623538)

[3.4 Програмна реалізація системи кодування/декодування відео/аудіо 56](#_Toc453623539)

[3.5 Реалізація системи рендерингу та анімації 59](#_Toc453623540)

[3.6 Реалізація системи захвату пера 65](#_Toc453623541)

[3.7 Реалізація системи управління елементами сцени 68](#_Toc453623542)

[3.8 Тестування програмного продукту 71](#_Toc453623543)

[ВИСНОВКИ 74](#_Toc453623544)

[ЛІТЕРАТУРА 76](#_Toc453623545)

[ДОДАТКИ 79](#_Toc453623546)

*Змн.*

*Арк.*

*№ докум.*

*Підпис*

*Дата*

*Арк.*

*7*

08-23.БДР.002.00.000.ПЗ

# ВСТУП

В останні 20-30 років медицина і біологія вступили у нову фазу свого розвитку. Нагромадження величезних масивів кількісних даних і доступність

обчислювальної техніки посилили математизацію біології та медицини. Сьогодні вже неможливо собі уявити наукові дослідження в галузі біології та медицини, які виконані без статистичного аналізу отриманих фактичних даних. Біостатистика стала суттєвим кроком у русі медичного дослідження від описів окремих випадків до експериментів з групами контролю, і, нарешті, до великомасштабних рандомізованих контрольованих досліджень, яким тепер надається перевага як стандартам наукового доказу.

Разом з тим, методи статистичної обробки отриманих результатів настільки різноманітні, що найчастіше досліднику дуже складно розібратися у них і вибрати адекватні поставленим завданням статистичні критерії. Тривалий час аналіз медичних даних могли здійснювати лише високопрофесійні фахівці, оскільки це вимагало серйозної попередньої підготовки. З появою обчислювальної техніки і вдосконаленням сучасних програм оброблення й аналізу даних статистична обробка піднялася на новий рівень. Тепер дослідник-медик може й не мати математичної підготовки.

Досить оперувати статистичними поняттями і, найголовніше, правильно вибрати метод аналізу [5], володіти навичками роботи на персональному комп'ютері та освоїти новітні пакети програмного забезпечення для статистичного оброблення й аналізу біомедичних досліджень.  **ОРГАНІЗАЦІЯ**

В даному розділі висвітлено поняття дистанційного навчання як системи навчання, що з'явилася нещодавно і у даний час починає все більш вагомо заявляти про себе як про нову високотехнологічну форму навчання. Саме тому варто проаналізувати мету та задачі, які ставить перед собою така система, а також визначити її переваги перед іншими. А враховуючи, що кожен хто вивчає будь яке явище, хоче побачити та почути його, якщо не на власні очі, то принаймні на відео, не дозволяє обійти стороною один із найвідоміший та найперспективніший методів дистанційного навчання - відеоуроки. Не менш важливим є дослідження можливостей створення таких уроків.

1.1 Аналіз технічного завдання.

За мету поставлено задачу візуального аналізу біомедичних даних, попередньо підготовлених і сформованих у багатоканальному wav файлі. Аналіз здійснюватиметься візуально, за допомогою графіків різного роду, які відображають форму та спектр сигналу до і після виконання фільтрації. Одночасно можна здійснювати аналіз по декільком каналам.

3. Вимоги до технічних характеристик комплексу.

3.1 Частота сигналів довільна.

3.2 Тип вхідних даних : wav файл

3.3 В залежності від інформації в заголовку у wav файлі програма повинна оброблювати різну кількість каналів з різною частотою дискретизації та розрядністю представлення даних.

3.4 Передбачити можливість виведення до 16 різних, одночасних графічних представлень, для кожного з яких має бути можливість вибору типу фільтра.

3.5 Кожне представлення повинно мати можливість одночасного виведення форми сигналу та його спектру із можливістю масштабування по обом осям кожного графіка.

3.6 Програма має виконувати фільтрацію різними типами фільтрів в залежності від вибору користувача.

3.7 Має буди передбачено можливість виведення на графіки амплітуд та спектрів як вхідного так і відфільтрованого сигналів.

3.8 Програма повинна мати можливість записувати у wav файл відфільтрований сигнал або його частину у заданому часовому проміжку.

3.9 Програма має мати можливість створювати вкладки для виведення графічної інформації.

3.10 Кожна вкладка повинна мати можливість задання часового інтервалу. Також має бути можливість задання одного часового інтервалу для всіх вкладок.

1.2 Розгляд сучасного програмного забезпечення для візуалізації аналізу біомединих даних

В основі оброблення й аналізу даних лежать математичні методи, які за останні півстоліття істотно не змінилися, проте відповідне програмне забезпечення за цей час зазнало істотних змін. Зі зміною поколінь ЕОМ мінялися й програмні засоби обробки та аналізу даних. І якщо можливості перших ЕОМ з аналізу даних не перевершували можливості сучасних середніх калькуляторів, то в 70-і роки вже з'явилися пакети прикладних програм, які містили практично всі ті математичні методи оброблення, які входять до складу й сучасних пакетів. Розвиток пакетів здійснювався шляхом вдосконалювання технології й аналізу (табл. 1).

На сьогоднішній день число пакетів для оброблення інформації досягає кількох десятків, серед яких зарубіжні пакети, такі як: SYSTAT, STATGRAPHICS, BMDP, SPSS, SAS, CSS, Statistica, а також вітчизняні: STADIA, ЕВРІСТА, МЕЗОЗАВР, САНІ, КЛАСС-МАСТЕР, СІГАМД тощо.

Більшість комп'ютерних статистичних програм не є чисто медичними прикладними програмами, оскільки більшість методів статистичного аналізу є універсальними й можуть застосовуватися не лише в різних галузях медичної статистики, але й у найрізноманітніших галузях людської діяльності. Наприклад, з погляду формальної логіки статистичний прогноз інфекційної захворюваності й прогноз курсу долара — це та ж сама задача, а тому вона може вирішуватися за допомогою одних і тих же пакетів прикладних програм.

Основну частину наявних пакетів для оброблення даних можна віднести до трьох категорій: спеціалізовані пакети, пакети загального призначення і професійні пакети.

Спеціалізовані пакети, як правило, містять методи з одного-двох розділів статистики або методи, що використовуються в конкретній предметній галузі (наприклад, Мезозавр — програма аналізу часових рядів). Спеціалізовані пакети застосовуються для вирішення вузького кола завдань з використанням спеціальних методів статистичного аналізу. Експлуатація цих програм вимагає високого рівня підготовки користувача в галузях певних розділів статистики.

Професійні пакети призначені для користувачів, які мають справу із надзвичайно великими обсягами даних або вузькоспеціалізованими методами аналізу. Пакети загального призначення або універсальні (Statgraphics, SPSS, Statistica, Excel, STADIA тощо) є найбільш зручними для користувача-початківця завдяки відсутності орієнтації на специфічну предметну галузь, широкому діапазону статистичних методів і дружньому інтерфейсу користувача. Вони більш доступні для практики й можуть використовуватися широким колом фахівців різного профілю. Практично всі задачі, які стосуються оброблення й аналізу медико-біологічних досліджень, можуть бути вирішені за допомогою універсальних пакетів.

Пакет Statgraphics розроблявся ще для роботи в середовищі DOS, a потім був адаптований до операційної системи Windows і отримав нову назву Statgraphics Plus. Сучасний пакет STATGRAPHICS PLUS - це досить потужна статистична програма, яка містить більше 250 статистичних функцій. За своїми характеристиками пакет займає проміжне місце між SPSS і Statistica.

Пакет SPSS (Statistical Package For Social Science) - це один із найбільш часто використовуваних пакетів статистичної обробки медико-біологічних даних (http://www.spss.com/). Цей пакет створювався ще для «великих» електронно-обчислювальних машин і послідовно переводився для роботи в середовищі DOS, а потім Windows. Пакет SPSS досить потужний і добре відпрацьований, наближається за своїми можливостями до професійних пакетів, і реалізація статистичних процедур добре пристосована до практичної роботи.

Поряд з пакетом SPSS, великою популярністю в освітніх та наукових закладах США користується пакет STATA. Це професійний статистичний програмний пакет з data-management system, який досить часто застосовують для біомедичних цілей.

Програма добре документована, видається спеціальний журнал для користувачів системи. Офіційний сайт <http://www.stata.com/>. Не менш популярним інструментом розробки користувацьких додатків і не лише в медицині, а й в бізнесі, економіці, фінансах, промисловості тощо є інтегрована система аналізу та управління даними STATISTICA [1].

Пакет Statistica спеціально створювався для роботи в середовищі Windows і відповідає всім стандартам Windows, що дозволяє зробити аналіз високоінтерактивним. STATISTICA складається з набору модулів, в кожному з яких зібрані тематично пов'язані групи процедур, що мають високу швидкість і точність обчислень. Система STATISTICA містить повний набір класичних методів аналізу даних [2]: від основних методів статистики до просунутих методів, що дозволяє гнучко організувати аналіз. Дані системи STATISTICA легко конвертувати в різні бази даних і електронні таблиці. Ця система відрізняється найбільш розвиненим інтерфейсом із користувачем і багатими графічними можливостями, підтримує високоякісну графіку, що дозволяє ефектно візуалізувати дані і проводити графічний аналіз. Гнучка і

потужна технологія доступу до даних дозволяє ефективно працювати як з таблицями даних на локальному диску, так і з віддаленими сховищами даних. Система STATISTICA є відкритою системою містить мови програмування, які дозволяють розширювати систему, запускати її з інших Windows-додатків, наприклад, з Excel.

Електронна таблиця Excel найпоширеніша, оскільки повністю русифікована і найбільш доступна, вона встановлюється автоматично при інсталяції пакета MS Office. Електронна таблиця Excel тісно інтегрується з іншими програмами пакета MS Office, наприклад, MS Word і PowerPoint, а тому найчастіше використовується при оформленні результатів роботи. MS Excel - це електронна таблиця з достатньо потужними математичними можливостями, проте деякі статистичні функції є просто додатковими, а тому розрахунки, зроблені за допомогою вбудованих окремих формул, не завжди визнаються авторитетними біомедичними журналами [4]. MS Excel, як правило, використовується при найпростішому статистичному аналізі даних. Окрім того, в MS Excel неможливо побудувати якісні наукові графіки. MS Excel добре підходить для накопичення даних, попередніх статистичних прикидок, для побудови деяких видів діаграм, проте остаточний статистичний аналіз бажано робити в програмах, які спеціально створені для цих цілей. Існує макрос-додаток XLSTAT-Pro (http://www.xlstat.com) для MS Excel, який включає в себе більше 50 статистичних функцій, включаючи аналіз виживаності.

Програма вітчизняної розробки STADIA включає в себе всі необхідні статистичні функції, призначені для статистичного аналізу даних. Проте ця програма фактично не змінюється з 1996 року, а тому графіки і діаграми, побудовані за допомогою STADIA, виглядають в сучасних презентаціях архаїчно. До позитивних якостей програми можна віднести російськомовний інтерфейс і наявність книг, що описують роботу [3]. Зі сторінки http://www.protein.bio.msu.su/ ~ akula / index.htm можна взяти демо-версію STADIA. Слід зазначити, що всі ці пакети постійно оновлюються і з кожним роком з'являються їх нові версії.

При виборі пакета для аналізу даних можна виділити два аспекти:

а) початковий вибір пакета аналізу;

б) поточний вибір при переході на більш сучасний, більш потужний пакет. Підходи в обох випадках дещо відрізняються.

У першому випадку на вибір накладаються такі обмеження:

1. Можливості комп'ютера.

2. Можливості одержання установчої версії пакета.

3. Характеристики пакета.

Що стосується першого пункту, то варто вибирати найбільш сучасні версії пакетів із тих, що можуть бути встановлені на наявний комп'ютер. Другий пункт очевидний - вибирати можна з тих пакетів, що доступні. Що стосується характеристик пакета, то тут варто розглянути такі аспекти: \

а) обчислювальні можливості,

б) зручність роботи,

в) складність освоєння.

Обчислювальні можливості. У випадку, коли необхідно обробляти медичні дані помірних обсягів (до декількох тисяч спостережень) стандартними статистичними методами, то найкраще використовувати універсальні пакети. Якщо дивитися з позицій лікаря-дослідника, то всі сучасні універсальні статистичні пакети за своїми обчислювальними можливостями повністю відповідають можливим потребам (Statistica, SPSS, Statgraphics Plus та інші пакети, що працюють в операційній системі Windows). Проте завжди варто переконатися, що обраний пакет містить необхідні методи обробки.

Зручність роботи. Всі сучасні пакети досить зручні в роботі (коли вони вже освоєні). Складність освоєння. За складністю освоєння пакети дещо розрізняються і тут варто віддати перевагу русифікованим пакетам або пакетам, з яких є доступна література або є ймовірність пройти курс навчання. Варто зауважити, що без крайньої необхідності (неможливість забезпечити необхідну обробку даних) не бажано змінювати обраний і освоєний пакет аналізу, тому що це призведе до значного збільшення трудовитрат.

Що стосується заміни пакета на більш сучасну версію, то тут є дві крайності:

1. Прагнення до постійного відновлення, установки самих останніх версій пакетів, як правило, віднімає багато сил, не дозволяє виробитися корисним стереотипам дій, у той же час не приводить до суттєвого зростання можливостей.

2. З іншого боку, уподобання застарілих пакетів найчастіше не дозволяє повною мірою використовувати можливості сучасної техніки і програмного забезпечення. Існує деякий емпіричний оптимум, що може визначатися зразковим терміном експлуатації пакета в 2-3 роки, після закінчення котрого доцільно здійснювати перехід до більш сучасних пакетів. При цьому переважно вибирати чергову версію того ж пакета, що використовувався раніше. Запам'ятайте: спадкоємність значно полегшує процес освоєння пакета.

Висновок

Широке впровадження сучасних комп'ютерних технологій і застосування пакетів прикладних програм докорінно змінило процес оброблення й аналізу медичних даних. Завдяки використанню комп' ютерів і сучасного програмного забезпечення, оброблення й аналіз медико-біологічних даних став набагато легшим. При цьому для застосування основних статистичних методів оброблення медичних даних лікарю не потрібно заглиблюватися в

складність математичних процедур, а варто зрозуміти, для чого і як ці методи використовуються, а також вміло використовувати обраний пакет приклад-

них програм.

# ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

Процес проектування складного програмного забезпечення починається з уточнення його структури, тобто визначення структурних компонентів і зв'язків між ними. Тому в даному розділі слід розглянути сучасні принципи проектування ПЗ, а також спроектувати програмне забезпечення для візуалізації біомедичних сигналів, як на фізичному так і логічному рівні. Результатом проектування має стати набір діаграм які розкривають можливості для користувача, і демонструють компоненти, які потрібні для розробки та реалізації потрібного функціоналу.

* 1. Основи проектування програмного забезпечення. UML

Основне завдання логічного проектування при об'єктному підході - розробка класів для реалізації об'єктів, отриманих при об'єктної декомпозиції, що передбачає повний опис полів і методів кожного класу. Фізичне проектування при об'єктному підході включає проектування об'єднання класів і інших програмних ресурсів в програмні компоненти та розміщення цих компонентів на конкретних обчислювальних установках.

Пакетом при об'єктному підході називають сукупність описів класів і інших програмних ресурсів, в тому числі пакетів. Об'єднання в пакети використовують тільки для зручності створення великих проектів, кількість класів в яких вели-ко. При цьому в один пакет зазвичай збирають класи та інші ресурси єдиного призначення. Діаграма пакетів показує, з яких частин складається проектована програм-ва система, і як ці частини залежать один від одного [13].

Моделі представлення елементів програмного забезпечення при об'єктному підході засновані на предметах і явищах реального світу. В основі всіх моделей лежить опис необхідного поводження розробляємого програмного продукту, тобто його функціональності, але ця поведінка пов'язується з станами елементів (об'єктів) конкретної предметної області.

Таким чином, на етапі аналізу ставляться два завдання:

* уточнити необхідну поведінку розробляється ПО;
* розробити концептуальну модель його предметної області з точки зору поставлених завдань.

В основі об'єктного підходу до розробки ПЗ лежить об'єктна декомпозиція, тобто поданні елементів ПЗ у вигляді сукупності об'єктів, в процесі взаємодії яких за допомогою передачі повідомлень і відбувається виконання потрібних функцій та поставлених завдань.

Розробка програмного забезпечення таким методом є складним у реалізації ну значно простіше для сприйняття, тому для розробки в наш час використовують спеціалізовану мову - універсальна мова моделювання (unified modeling language) UML. Ця мова, авторами якого є Г. Буч, Д. Рамбо і А. Джакобсон, не так давно був визнаний стандартним засобом опису об'єктних розробок.

Повний опис розробки з використанням UML включає кілька моделей, які характеризують певний аспект проектованої системи:

* + модель використання - є опис функціональності ПО з точки зору усіх користувачів, у будь якій формі (користувач, програміст, інша система, тощо);
  + логічна модель - описує ключові абстракції ПО (класи, інтерфейси, і т.п.), тобто елементи, що забезпечують необхідну функціональність;
  + модель реалізації - визначає реальну організацію програмних модулів;
  + модель процесів - відображає організацію обчислень і оперує поняттями «процеси» і «потоки». Вона дозволяє оцінити продуктивність, маштабування і надійність ПЗ;
  + модель розгортання - показує особливості розміщення програмних компонентів на конкретному обладнанні.

Всього UML пропонує дев'ять діаграм, основними з яких є:

* діаграми варіантів використання;
* діаграми класів та пакетів;
* діаграми послідовностей дій та станів;
* діаграми компонентів;
* діаграми розміщення.

Розглянемо діаграмами варіантів використання, компонентів, класів та розміщення. У діаграмами пакетів немає необхідності через використання безкоштовного фреймворку, а внутрішні зв’язки допоможуть відобразити діаграмами класів та компонентів. Діаграмами станів та послідовності дій будуть представлені у формі алгоритмів [14].

* 1. Діаграма варіантів використання (use case diagram)

Призначення даної діаграми полягає в наступному: проектована програмна система представляється у формі так званих варіантів використання, з якими взаємодіють зовнішні сутності або актори. При цьому актором або дійовою особою називається будь-який об'єкт, суб'єкт або система, що взаємодіє з модельовану бізнес-системою ззовні. Це може бути людина, технічний пристрій, програма або будь-яка інша система, яка служить джерелом впливу на модельовану систему так, як визначить розробник.

Варіант використання служить для опису сервісів, які система надає актору. Іншими словами кожен варіант використання визначає набір дій, які виконуються системою при роботі з актором. При цьому не важливо як це буде реалізовано.

Розглядаючи діаграму варіантів використання в якості моделі бізнес-системи, можна асоціювати її з "чорним ящиком". Концептуальний характер цієї діаграми проявляється в тому, що докладна деталізація діаграми або включення в неї елементів фізичного рівня уявлення на початковому етапі проектування швидше має негативний характер, оскільки визначає способи реалізації поведінки системи. Ці аспекти повинні бути свідомо приховані на діаграмі варіантів використання. Тому будуть розглядатися основні вимоги до програмного майбутнього програмного продукту, без розгляду їх реалізації [13].

Розробка продукту передбачає створення зручного багатофункціонального комплексного додатку для аналізу вхідного багатоканального сигналу, який можна було використовувати в не лише в медичних цілях. Програма повинна мати велику кількість вбудованих інструментів для візуалізації вхідного сигналу, його модифікації, а саме головне порівняння та аналізу вихідних даних. Крім того має бути можливість розширення функціоналу по мірі потреб, що потребувало написання добре підтримуємого коду.

Отже основним користувачем звісно виступає особа, яка саме і бере участь у процесі аналізу вхідного сигналу. Вона має отримати доступ до великої кількості інструментів, які мають бути представленні зручним користувацьким інтерфейсом. Крім того кожен інструмент повинен бути не лише простий у використанні, а й бути максимально конфігуруватися як користувачем під час роботи так і програмістом з подальшим розвитком продукту [15].

Ці інструменти можна розбити на категорії повезені з призначенням:

* інструменти керування проектом:

1. створення проекту (створення файлу проекту, який містить службову інформацію, а також внутрішню архітектуру проекту, налаштування і шляхи до зовнішніх файлів які не вдалося інтегрувати);
2. редагування проекту (можливість редагувати раніше створений проект, враховуючи особливості поточної версії програми);
3. резервування проекту (можливість відновити проект вразі внутрішніх конфліктів, або його пошкодження).

* інструменти керування блоками (відео фрагменти):

1. формування блоків певного типу (додавання до сцени певного мультимедійного об’єкту);

В то самий час окрім особи яка проводить створення відеоуроку, програма може використовуватися програмістами для відлагодження та розширення функціоналу програми. Згідно проаналізованого матеріалу було побудовано, UML діаграму варіантів використання, яка наведена на рисунку 2.1 [13-15].



Рисунок 2.1 – Діаграма варіантів використання

* 1. Основні компоненти. Діаграма компонентів (component diagram)

Для створення конкретної фізичної системи необхідно реалізувати всі елементи логічного представлення в конкретні матеріальні сутності. Для опису таких реальних сутностей призначений інший аспект модельного уявлення, а саме - фізичне представлення моделі. У контексті мови UML це означає сукупність пов'язаних фізичних сутностей, включаючи програмне і апаратне забезпечення, а також персонал, які організовані для виконання спеціальних завдань.

Для реалізації системи необхідно розробити вихідний текст програми на мові програмування. При цьому вже в тексті програми передбачається організація програмного коду, що визначається синтаксисом мови програмування і передбачає розбиття вихідного коду на окремі модулі.

Однак вихідні тексти програми ще не є остаточною реалізацією проекту, хоча і служать фрагментом його фізичного представлення. Програмна система може вважатися реалізованою в тому випадку, коли вона буде здатна виконувати функції свого цільового призначення. А це можливо, тільки якщо програмний код системи буде реалізований у формі виконуваних модулів, бібліотек класів і процедур, стандартних графічних інтерфейсів, файлів баз даних. Саме ці компоненти є базовими елементами фізичного представлення системи в нотації мови UML.

Повний проект програмної системи являє собою сукупність моделей логічного і фізичного уявлень, які повинні бути узгоджені між собою. У мові UML для фізичного представлення моделей систем використовуються так звані діаграми реалізації, які включають в себе дві окремі канонічні діаграми: діаграму компонентів і діаграму розгортання [13].

Діаграма компонентів, на відміну від раніше розглянутих діаграм, описує особливості фізичного представлення системи. Діаграма компонентів дозволяє визначити архітектуру розроблюваної системи, встановивши залежності між програмними компонентами, в ролі яких може виступати вихідний, бінарний і виконуваний код. У багатьох середовищах розробки модуль або компонент відповідає файлу. Пунктирні стрілки, що з'єднують модулі, показують відношення взаємозалежності, аналогічні тим, які мають місце при компіляції початкового програмного коду. Основними графічними елементами діаграми компонентів є компоненти, інтерфейси і залежності між ними.

Особливістю програмного забезпечення, що розроблється, є його функціональна навантаженість. Така форма передбачає використання великої кількості інструментів, а також складних елементів користувацького інтерфейсу, що утворюють окремі модулі із власними залежностями один від одного. Зв'язок між модулями має бути тісний, але частина з них маж бути більш незалежна для того щоб можна було додавати новий функціонал, без серйозних змін в інших компонентах [14].

Оскільки програма має працювати з відео та аудіо контентом, то варто використовувати бібліотеку або ряд бібліотек для стандартизованого пакування, розпакування, кодування та декодування аудіо та відео контейнерів. Для уникнення проблем пов’язаних з типізацією даних та використання з іншими бібліотека ми, слід створити обгортку на вибрану бібліотеку, а також розділи процес пакування та розпакування.

Для виводу графіки на екран варто використовувати бібліотеку яка б дозволила використовувати апаратні ресурси машини для виводу графіки (особливо відеоадаптер). Це дозволить розвантажити центральний процесор і дасть розширити функціонал програми без втрати швидкодії. Як із попереднюю бібліотеко слід обгорнути бібліотеку, а також розробити додаткові модулі для виводу складних графічних елементів, приведених до об’єктів [15].

Оскільки програма матиме досить складний і багатий візуальний інтерфейс слід вибрати framework (набір бібліотек), який не лише дозволив використовувати стандартні елементи користувацького інтерфейсу (user interface UI), а й спростити роботу з мовою програмування і дати можливість створювати власні елементи UI за допомогою одної із багатьох мов розмітки та задання стилю.

Крім вище сказано потрібно щоб вибраний framework мав можливість роботу з різними типами даних. Не менш важливим моментом є можливість переносу коду на інші платформи і максимальна інтегрованість об’єктних принципів програмування.

Згідно проаналізованого матеріалу та технічного завдання було побудовано діаграму компонентів зображену на рисунку 2.2.

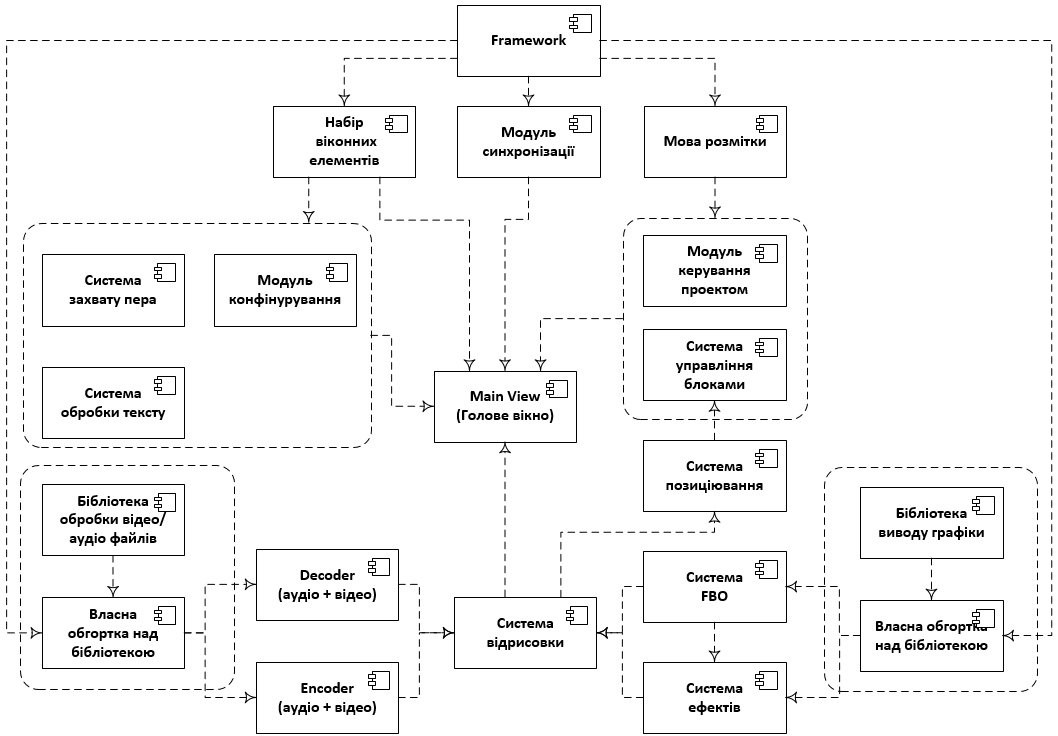


Рисунок 2.2 – Діаграма компонентів

Розглянемо основні представлені блоки, як окремі, автономні одиниці.

* + 1. Головне вікно

Головне вікно має представлятися як ключовий елемент користувацького інтерфейсу. Він має давати доступ до більшості інструментів програми, які мають бути доступні звичайному користувачу завдяки інтегрованому меню та панелі швидкого доступу. Набір інструментів з наступним функціоналом повинні ділитися на наступні категорії:

* File:
  1. new project (створення нового проекту);
  2. open project (відкриття уже готового проекту);
  3. save project (зберігання нових змін);
  4. open document (завантаження текстового документу);
  5. save document (збереження текстового документу);
  6. save as document;
  7. exit (безпечний вихід із програми).
* Edit (управління текстовими елементами програми):
  1. undo;
  2. redo;
  3. cut;
  4. copy;
  5. paste;
  6. select all;
  7. find.
* settings (елементи кастомізації UI):
  1. program font;
  2. program text color;
  3. canvas font.
* board:
  1. show;
  2. hide;
  3. hide edit box (приховування елементів позиціювання);
  4. play;
  5. pause;
  6. stop;
  7. record to file (запис у вихідний файл);
  8. send to YouTube (відвантаження на сервіс YouTube).
* help:
* about (інформацію про програму);
* aboutQt (відображення ліцензійної згоди).

Усі елементи меню та кнопок швидкого доступу мають супроводжувати підказками. Крім того інтерфейс має бути однаково себе поводити на різних дисплеях з різною роздільною здатністю, що стосується усіх елементів користувацького інтерфейсу.

Крім доступу до інструментів внутрішні механізми мають відслідковувати діяльність користувача, а також контролювати правильність роботи інших елементів програми. Діяльність користувача відслідковується для аналізу змін у проекті, для видачі повідомлень про можливу втрату даних вразі внесення змін у проект.

* + 1. Механізм FBO

Більшість елементів у сцені мають власні умови виводу на екран. Такі елементи перед тим як виводитись у мають пройти індивідуальну обробку, після чого елемент буде інтегрована в сцену і знову буде застосована уже пост обробка. Для цього був розроблений механізм для гнучкої закадрового відтворення, який отримав назву Frame buffer object (FBO) [16].

Даний механізм застосовується в багатьох аспектах комп'ютерної графіки, таких як пост ефекти, тіні, віддзеркалення, освітлення, а також в технології відомої як R2VB (Render to Vertex Buffer), на якій базуються багато вершинних ефекти, починаючи від побудова рельєфу поверхні по карті висот і до скелетної анімації на GPU. Окремо варто відзначити можливість виконувати паралельні обчислення на GPU, які так само не можливі без технології рендеринга в текстуру.

Для організації такої системи потрібно реалізувати такий механізм на рівні об’єктів, які мають бути тісно пов’язані з бібліотекою виводу зображення. Такі об’єкти мають мати можливість легко створити такий FBO, а також легко здійснювати переключення між ними і виводити їх на головну сцену.

* + 1. Система ефектів

Система ефектів має являти собою набір класів який дозволив би накласти каскад візуальних ефектів на будь які елементи, що мають візуальне представлення. Дана система має бути тісно повязеною з системою FBO. Кожен ефект має представлятися шейдерним блоком, що складається з вершинного та піксельного шейдерів, або являти собою послідовність трансформацій об’єкту. Всі ефекти повинні конфігуруватися, тобто містити змінні для налаштувань, а також підтримувати можливість відображення згідно часу.

При написанні (створенні) програмістами інших типів ефектів (з іншою поведінкою), всім надається однаковий набір параметрів таких як: загальний час відтворення(локальний), поточний час відтворення (локальний), роздільна здатність та інші необхідні уніфіковані параметри. Ефект може застосовуватися до будь якого об’єкту, навіть якщо він немає графічного представлення, в такому випадку ефект не буде мати ніякого впливу.

Крім внутрішньої системи має бути розроблений користувацький інтерфейс для керування ефектами (створення, видалення, зміна черговості та параметрів). Інтерфейс має підлаштовуватися під тип ефекту та під його властивості враховуючи тип властивості.

* + 1. Система відображення

Основним призначення є вивід підготовленого зображення на екран. Як і система ефектів являється, системою класів яка тісно пов’язана з системою FBO та ефектів. У даній системі містить основні буфери для фінального виводу, в які і відбувається занесення FBO усіх елементів сцени.

Крім того дана система має містити в собі елементи позиціювання, такі як сітка розмітки та контейнера для об’єкта сцени для його розміщення, масштабування та обертання.

* + 1. Система управління блоками

Дана система являє собою складний користувацький інтерфейс, який дозволяє додавати елементи різних типів у сцену, а також здійснювати їх позиціювання в часі та редагування параметрів. Зміна позиції в часі має здійснюватися методом перетаскування.

Має бути можливість додати новий елемент в будь яке місце часового проміжку, крім того мають бути додаткові інструменти, такі як прилипання, заборона на додавання та групування об’єктів на сцені. Даний інтерфейс має підтримувати додавати на сцену велику кількість елементів з урахування одночасного відтворення багатьох елементів.

* 1. Діаграма класів (classes diagram)

Клас - це основний будівельний блок ПЗ. Це поняття присутнє і в обєктно-орієнтованих мовах програмування, тобто між класами UML і програмними класами є відповідність, що є основою для автоматичної генерації програмних кодів. Кожен клас має назву, атрибути і операції. Клас на діаграмі показується у вигляді прямокутника, розділеного на 3 області. У верхній міститься назва класу, в середній - опис атрибутів (властивостей), в нижній - назви операцій - послуг, що надаються об'єктами цього класу.

Атрибути класу визначають склад і структуру даних, що зберігаються в об'єктах цього класу. Кожен атрибут має ім'я і тип, що визначає, які дані він представляє. При реалізації об'єкта в програмному коді для атрибутів буде виділена пам'ять, необхідна для зберігання всіх атрибутів, і кожен атрибут матиме конкретне значення в будь-який момент часу роботи програми.

Для кожного атрибута класу можна задати видимість:

* Відкритий - атрибут видно для будь-якого іншого класу («+»);
* Захищений - атрибут видно для нащадків даного класу («~»);
* Закритий - атрибут непомітний зовнішніми класами і може використовуватися тільки об'єктом який його містить («-»).

Останнє значення дозволяє реалізувати властивість інкапсуляції даних. Наприклад, оголосивши всі атрибути класу закритими, можна повністю приховати від зовнішнього світу його дані, гарантуючи відсутність несанкціонованого доступу до них. Це дозволяє скоротити число помилок в програмі. При цьому будь-які зміни в складі атрибутів класу ніяк не позначяться на решті частини ПЗ [13,14].

Клас містить оголошення операцій, що представляють собою визначення запитів, які повинні виконувати об'єкти даного класу. Кожна операція має сигнатуру, яка містить ім'я операції, тип значення і список параметрів, який може бути порожнім. Реалізація операції у вигляді процедури - це метод, що належить класу. Для операцій, як і для атрибутів класу, визначено поняття «видимість». Закриті операції є внутрішніми для об'єктів класу і недоступні з інших об'єктів. Решта утворюють інтерфейсну частину класу і є засобом інтеграції класу в ПЗ

Класи можуть перебувати у певних відношеннях або зв’язках. Розглядаються бінарні асоціації, в яких об’єкт з кожної сторони відіграє свою роль. На діаграмах класів зазвичай показуються асоціації та узагальнення.

Є багато видів залежностей між класами: деякий клас-клієнт може використовувати певний сервіс (операцію) іншого класу; класи можуть бути пов’язані відношенням трасування, коли один трансформується в другий унаслідок певного процесу життєвого циклу, наприклад, клас аналізу перетворюється в клас проекту, а потім у клас реалізації. Один клас може бути уточненням другого.

Діаграма класів може належати до екземплярів класу, суперкласів (абстрактних класів) або підкласів (конкретних класів). У кожному з конкретних прикладів на іконці класу перед його назвою зазначається його стереотип («підклас», «суперклас» тощо), при цьому за замовчуванням вважається клас. Для стереотипів, які позначають відношення, фіксованими є такі: «асоціація», «наслідування», «екземпляризація», «узагальнення», «розширення» та інші.

Згідно ТЗ та розроблених діаграм компонентів та діаграм використання було розроблено діаграму класів (рис.2.3) без урахування атрибутів, оскільки більшість класів є занадто навантаженими для відображення у такій формі.

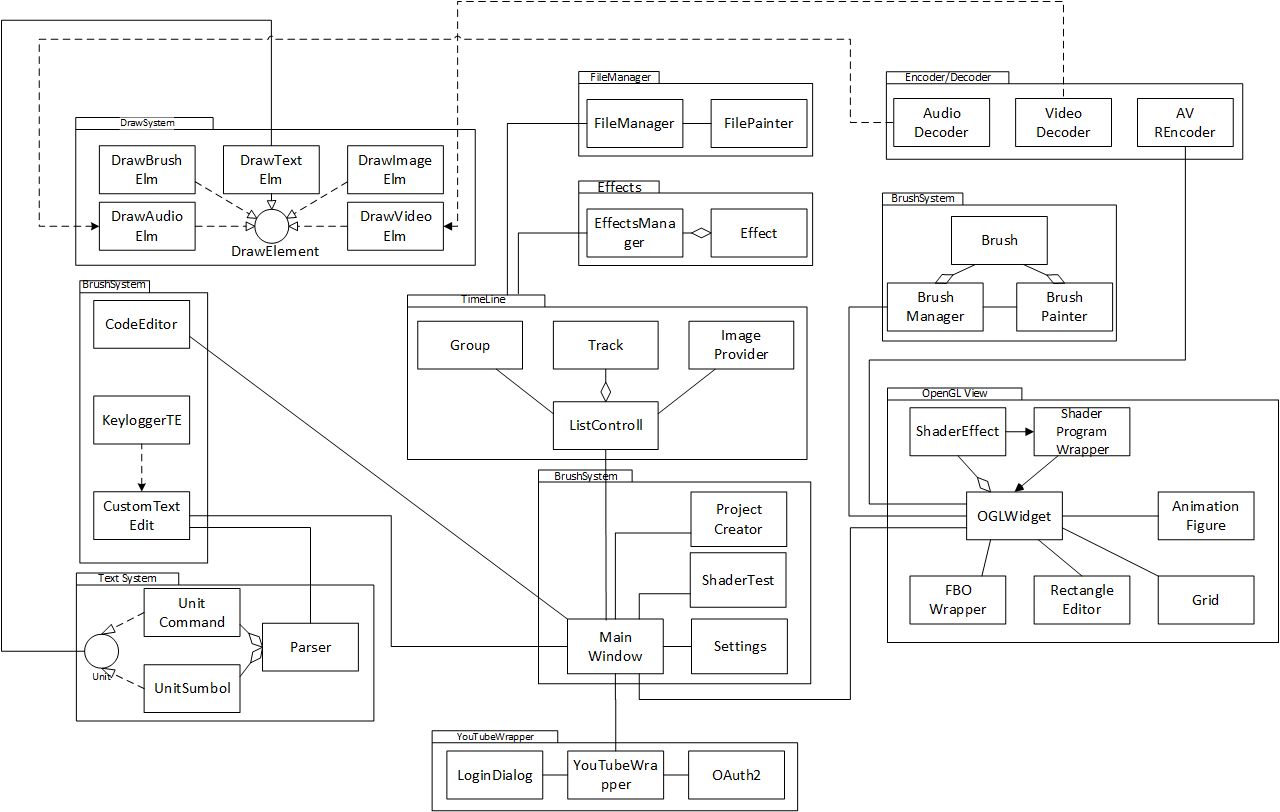


Рисунок 2.3 – Діаграма класів

# РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

В даному розділі виконується вибір мови та бібліотек, необхідність яких обґрунтована в попередньому розділі. Також у розділі описано розробку та тестування таких ключових модулів програми як: системи кодування та декодування аудіо та відео потоку, систему відображення, ефектів і захвату пера, а також системи управління сценою, тобто її контентом.

3.1 Обґрунтування вибору мови програмування

Реалізація системи обумовлена технічним завданням у додатку А. Для написання програми необхідно проаналізувати і обрати мову, що врахує усі вимоги до мови, обґрунтування яких було здійснено під час проектування програмного продукту.

Згідно проаналізованого матеріалу у попередньому розділі, існує потреба у використання набору бібліотек, які б полегшили написання користувацького інтерфейсу, а також надали можливість написання власних елементів UI. Саме тому при виборі мови варто звернути увагу не лише переваги мови, а й на можливості найпоширеніших фреймворків для даних мов, а також їх використання у корпоративних рішеннях.

3.1.1 Вибір мови програмування

Assembler – мова програмування низького рівня, що представляє собою формат запису машинних команд, зручний для сприйняття людиною. Мова асемблера забезпечує базові програмні абстракції: зв'язування частин програми і даних через мітки з символьними іменами й директивами [18].

С++ – стандартизована об'єктно-орієнтована мова програмування. Вона є ефективною для створення системного програмного забезпечення, та прикладних програм. Має зручні механізми управління пам’яттю та даними. Крім того має велику базу підготовленого коду (бібліотек, фреймворків, тощо). Швидкодія програм написаних на С++ дуже висока [19].

C# – мова програмування, що поєднує об'єктно-орієнтовані і контекстно-орієнтовані концепції. Дана мова працює на базі технології .NET. Мова має строгу статичну типізацію. C# не підтримує множинне наслідування класів або виведення типів. Швидкодія програм написаних на С# досить висока [20].

PHP – рефлексивний мова програмування, розроблена як інструмент для створення динамічних Web-сторінок і роботи з базами даних. PHP є скриптовою мовою і виділяється серед інших завдяки своїй простоті, швидкості виконання, багатій функціональності.

Java – об'єктно-орієнтована мова програмування, що спочатку розроблялася для побутової електроніки, але згодом стала використовуватися для написання аплетів, додатків і серверного програмного забезпечення. Мова Java зародилася як частина проекту створення передового програмного забезпечення для різних побутових приладів [21].

Для вибору розглянуто такі критерії, як наявність об’єкто-орієнтовної парадигми, застосування багатопоточності, багатоплатформеність, швидкість виконання, оскільки вони є найактуальнішими при виборі мови. Не менш важливим є і кількість підготовленого коду. З таблиці 3.1 можна зробити висновок, що найкращою мовою для реалізації даного продукту є С++, оскільки ця мова має більше переваг ніж інші.

Таблиця 3.1 – Порівняння сучасних мов програмування

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Можливості | Assembler | C++ | C# | PHP | Java |
| Об’єктно-орієнтована | – | + | + | + | + |
| Динамічні масиви | – | + | + | + | + |
| Багатопоточність | – | + | + | + | + |
| Узагальнене програмування | – | + | + | + | + |
| Багатоплатформеність | – | + | +/– | + | + |
| Статична типізація | – | + | + | – | + |
| Швидкість виконання | + | + | + | +/– | +/– |

3.1.2 Вибір бібліотеки класів

Оскільки такі мови як Assembler та PHP не призначені для створення масштабних програмних продуктів із складним користувацьким інтерфейсом, то розглядати їх в подальшому не варто. Проаналізуємо найбільш поширені бібліотеки класів для мов С++, С# та Java.

Бібліотекою класів для мови C# є .NET Framework, що являє собою колекцію типів, які тісно інтегруються із середовищем CLR. Бібліотека класів є об'єктно-орієнтованою; надаючи типи, з яких керований код користувача може успадковувати функції. Це не тільки спрощує роботу з типами .NET Framework, але також зменшує час, що витрачається на вивчення нових засобів платформи .NET Framework. Крім того, компоненти незалежних виробників можна легко поєднувати з класами платформи .NET Framework. Наприклад, в класах колекцій .NET Framework реалізується набір інтерфейсів, які можна використовувати для розробки призначених для користувача класів колекцій [20].

Як і очікується від об'єктно-орієнтованої бібліотеки класів, типи .NET Framework дозволяють вирішувати типові завдання програмування, включаючи роботу з рядками, збір даних, підключення до баз даних і доступ до файлів. На додаток до цих звичайним завданням бібліотека класів містить типи, що підтримують багато спеціалізовані сценарії розробки.

Технологія WPF (Windows Presentation Foundation) є частина екосистеми платформи .NET і являє собою підсистему для побудови графічних інтерфейсів.

Якщо при створенні традиційних додатків на основі WinForms за отрисовку елементів управління і графіки відповідали такі частини ОС Windows, як User32 і GDI +, то додатки WPF засновані на DirectX. У цьому полягає ключова особливість рендеринга графіки в WPF: використовуючи WPF, значна частина роботи по відображенні графіки, як найпростіших кнопочок, так і складних 3D-моделей, лягати на графічний процесор відеоадаптера, що також дозволяє скористатися апаратним прискоренням графіки.

Однією з важливих особливостей є використання мови декларативною розмітки інтерфейсу XAML, заснованого на XML: ви можете створювати насичений графічний інтерфейс, використовуючи або декларативне оголошення інтерфейсу, або код на керованих мовах C # і VB.NET, або поєднувати і те, і інше.

Для мови Java новим, але найбільш перспективним фреймворком є JavaFX. JavaFX - платформа для створення RIA (RichInternetApplication), дозволяє будувати уніфіковані додатки з насиченим графічним інтерфейсом користувача для безпосереднього запуску з-під операційних систем, роботи в браузерах і на мобільних телефонах, в тому числі, які працюють з мультимедійним вмістом [21].

У розробки на платформі JavaFX є наступні переваги:

* оскільки JavaFX написана на Java, Java розробники зможуть використовувати свої вміння в розробці JavaFX додатків;
* оскільки Java дуже популярна, досить легко знайти досвідчених Java розробників, які швидко і ефективно приступлять до розробки JavaFX додатків;
* використання Java як клієнтської і серверної платформи знижує складність бізнес-системи;
* через зниження складності системи розробка коштує дешевше;
* платформа JavaFX поставляється бібліотекою класів і середовищем розробки широковживаних бізнес-додатків для всіх платформ, що підтримують Java;

Qt - це крос-платформна бібліотека C ++ класів для створення графічних користувацьких інтерфейсів (GUI). Ця бібліотека повністю об'єктно-орієнтована, що забезпечує легке розширення можливостей і створення нових компонентів. Вона підтримує величезну кількість платформ, таких як Windows, Linux, Sun Solaris, HP-UX, Digital Unix, IBM AIX, SGI IRIX і багато інших. З моменту першої появи комерційної версії Qt в 1996 році з'явилася кілька сотень відомих по всьому світу додатків.

Для бібліотеки створення GUI об'єктно-орієнтована мова є єдино придатним засобом, а стандартна об'єктна модель C ++ забезпечує ефективну і швидку розробку програм, нарощування необмежених можливостей і швидку модернізацію. Природно через це в якості базової мови для бібліотеки обраний C ++. Але на додаток до можливостей самої мови в бібліотеці Qt додані декілька хороших можливостей:

* потужний механізм комунікації між об’єктами;
* механізм створення властивостей об'єктів;
* підтримку подій і фільтрів подій;
* переклад рядків для підтримки інтернаціоналізації;
* підтримку внутрішніх таймерів, які дозволяють інтегрувати багато завдань для складних GUI;
* ієрархічні дерева об'єктів, які є свого роду генеалогічними деревами.
* "безпечні" вказівники, які автоматично встановлюються в NULL при видаленні об'єкта, на який веде посилання.

Qt Quick - новий перспективний інструмент Qt, що дозволяє проектувати UI додатки на простому javascript-подібній декларативній мові (QML), використовуючи всі можливості Qt Animation Framework і навіть більше. Завдяки Qt Quick, створення UI і коду стають повністю незалежними етапами розробки. Qt Quick відкриває нові можливості при переносі на інші платформи і кастомізації додатків, скорочує терміни розробки і виводить UI на якісно новий рівень [22].

Всі ці можливості реалізовані у вигляді спеціальних класів C ++. Поява нових сучасних вимог до призначених для користувача інтерфейсів тягне за собою і появу нових нестандартних механізмів, які сама мова програмування забезпечити не здатний, а лише після створення спеціалізованих класів.

Найбільш привабливим рішенням згідно проаналізованого матеріалу є використання бібліотеки класів Qt оскільки вона дає використовувати всі переваги мови С++, яка є найшвидшою хоча і не самою зручною із розглянутих мов програмування. Вибрана платформа є не новою, тому її надійність значно спрощує розробку програмного продукту, а інструмент Qt Quick та мова QML надають можливість створити складний користувацький інтерфейс, як незалежну одиницю.

Не менш важливим критерієм є те що бібліотека підтримує, велику кількість платформ для розгортання з максимальною адаптивністю до звичних компонентів, а також безкоштовне ліцензування більшості компонентів для корпоративних рішень.

* 1. Обґрунтування вибору бібліотеки та кодека обробки аудіо/відео

Написання бібліотеки для кодування/декодування з нуля дуже трудомістка задача, довго реалізовувати підтримку великої кількості форматів, кодеків і інших нюансів. Тому варто використовувати сторонні бібліотеки, такі як libVLC, FFmpeg, MFT, DirectShow.

MFT та DirectShow розроблені компанією Microsoft для операційної системи Windows, тому їх використання обмежує перенос на інші платформи.

FFmpeg та libVLC - набір вільних бібліотек з відкритим вихідним кодом, які дозволяють записувати, конвертувати і передавати цифрові аудіо- та відеозаписи в різних форматах. При чому libVLC більшою мірою призначена для декодування і мало оптимізована для кодування відео. FFmpeg же включає libavcodec - бібліотеку кодування і декодування аудіо та відео і libavformat - бібліотеку мультиплексування і демультиплексування в медіаконтейнера.

FFmpeg розроблений під ОС на основі Linux, однак може бути скомпільовано під багато інших операційні системи. Поширюється під ліцензіями GNU LGPL або GNU GPL. Основним недоліком відкритого коду є відсутність якісної документації та прикладів роботи. Тому в якості документації будуть використовуватися внутрішні коментарі розробників проекту.

Оскільки бібліотека FFmpeg надає можливості як декодування так кодування, то саме її найбільш доцільно використовувати при розробці даного програмного продукту [5,23].

Аудіо контент є досить важливим, але високих вимог до нього не виставляється. Причиною цього є те що основним елементом композиції є голос диктора, або фон. Тому був безконкурентно вибраний найбільш популярний аудіо кодек MP3, який забезпечує високу якість при низькій вазі вихідного файлу, саме при таких умовах використання.

Основними критерієм при виборі відео кодека є якість, вага вихідного файлу та легкість подальшого декодування, швидкість кодування відходить на задній план. Причина такого рішення, пояснюється подальшим розташуванням файлу для доступу з мережі Інтернет, що значно обмежує канал передачі даних, а враховуючі, що відео курс може складатися з декількох сотень відео слід враховувати обмеження і вільного простору на сервері де буде розміщуватися контент. Останній же критерій обґрунтовується широким використанням мобільних пристроїв для перегляду відео контенту, що значно обмежує у обчислювальній потужності машини кінцевого користувача [23].

Згідно останнього критерію існує два найбільш популярних відео кодека, які широко використовуються у мультимедійних пристроях H264 та MPEG4. У кодеку MPEG-4 стиснення здійснюється не тільки індивідуально для кожного кадру, але і для серії кадрів (міжкадрове стиснення), тому що великі обсяги даних в серії кадрів часто залишаються незмінними. Однак це добре при несуттєвих змінах в кадрі, простіше кажучи, якщо рух практично відсутній. Але для об'єктів з інтенсивністю подій, великим потоком людей і автомобілів, важливо отримати якісне зображення в моменти саме руху. Ця необхідність привела до розробки нового прогресивного формату H.264 (або MPEG-4 V10) [6].

H.264 дозволяє формувати високоякісний відеосигнал зі значно меншим цифровим потоком, ніж MPEG-4, але при цьому вимоги до продуктивності процесора досить високі. Але формат H.264 дозволяє зменшити розмір файлу з цифровим відео більш, ніж на 80% в порівнянні з форматом M-JPEG.4 при рівних показниках якості.

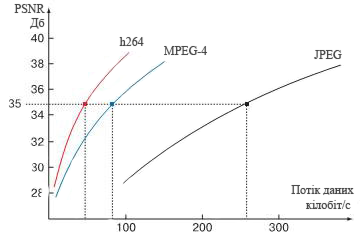


Рисунок 3.1 – Графік залежності якості від швидкості передачі даних

Графік рисунку 3.1 являє собою приклад порівняння цифрових потоків зображень, переданих камерами в кодеках JPEG, MPEG-4 і H .264. По вертикалі - «пікове значення сигнал-шум» (PSNR), що визначає «якість» стислих відеозображень, а по горизонталі - цифровий потік. Згідно даного графіка випливає, що кодек H264 показує значно кращі результати при одинакові пропускній спроможності. Що важливо, оскільки вихідний файлу має передавати із сервера на віддалений пристрій і саме така особливість наддасть можливість значно розширити аудиторію, тому вибір відео кодека падає саме на нього [5,6].

* 1. Обґрунтування вибору бібліотеки виводу зображення

Серед найбільш популярних бібліотек, які спрощують роботу з виводом графіки та забезпечують можливість переносу на нові платформи є наступні:

1. OpenGL
2. DirectX
3. Vulkan

OpenGL — (англ. Open Graphics Library — відкрита графічна бібліотека) — специфікація, що визначає незалежний від мови програмування крос-платформовий програмний інтерфейс (API) для написання застосунків, що використовують 2D та 3D комп'ютерну графіку. Цей 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 [24].

Vulkan - багатоплатформовий API для 2D і 3D графіки, вперше представлений Khronos Group в рамках конференції GDC 2015.

Vulkan API спочатку був відомий як «нове покоління OpenGL» або просто «glNext», але після анонса компанія відмовилася від цих назв на користь назви Vulkan. Як і OpenGL, Vulkan дозволяє з високою продуктивністю відображати в реальному часі різні додатки з 3D графікою, такі як ігри чи інтерактивні книги, на всіх платформах, а також забезпечує більш високу продуктивність і менше навантаження на процесор, аналогічно Direct3D 12 і Mantle. Vulkan заснований на технологіях AMD в Mantle [25].

DirectX — це набір API функцій, розроблених для простого і ефективного вирішення завдань, пов'язаних з ігровим і відеопрограмуванням під Microsoft Windows. Найширше використовується при написанні відеоігор. Пакет засобів розробки DirectX під Microsoft Windows безкоштовно доступний на сайті Microsoft. Незважаючи на те, що зараз DirectX включений в стандартний набір ПЗ Windows, свіжі версії DirectX часто поставляються разом з іграми. Це пов'язано з регулярним оновленням API, і версія, включена в ОС Windows часто є не останньою. Існує часткова підтримка для OS Linux [26].

Серед усіх вище наведених найбільш зручним у використанні та у підтримці являється DirectX від компанії Microsoft. Головним недоліком серед інших обмежені можливості при переносі на інші платформі. Vulkan є доволі новим API, хоча один з найбільш перспективним у області 2D графіки. Але через відсутність достатньої документації і нестабільність, вирішено використовувати бібліотеку OpenGL.

3.4 Програмна реалізація системи кодування/декодування відео/аудіо

Оскільки для обробка відео та аудіо контенту використовується бібліотека FFmpeg, згідно діаграми компонентів наведеної в попередньому розділі, було вирішено створити так звану обгортку над даною бібліотекою. Такою обгорткою став каскад класів Encoder, EncoderPrivate, EncoderGlobal, AV\_REncoder, AudioDecoder, VideoDecoder [5,6].

Клас EncoderPrivate відповідає саме за створення вихідного відео файлу. Головний функціонал даного класу це налаштування та ініціалізація вибраного кодека для стискання контенту. Не менш важливою здачею даного класу є підготовка вхідних даних для стиснення, так званий «ресеплінг», який здійснює перетворення вхідного зображення у відповідний кольоровий формат представлення даних, зміну розширення зображення, а також зміни частоти дискретизації, кількості каналів і формату представлення аудіо даних. Більшість параметрів для пакування було винесено у клас EncoderGlobal, який містить велику кількість заготовлених констант. Переважна більшість полів класу є об’єктами та вказівниками на об’єкти бібліотеки FFmpeg, процес ініціалізації якої, для кодування відео та аудіо, складається з етапів наведених на рисунку 3.2.

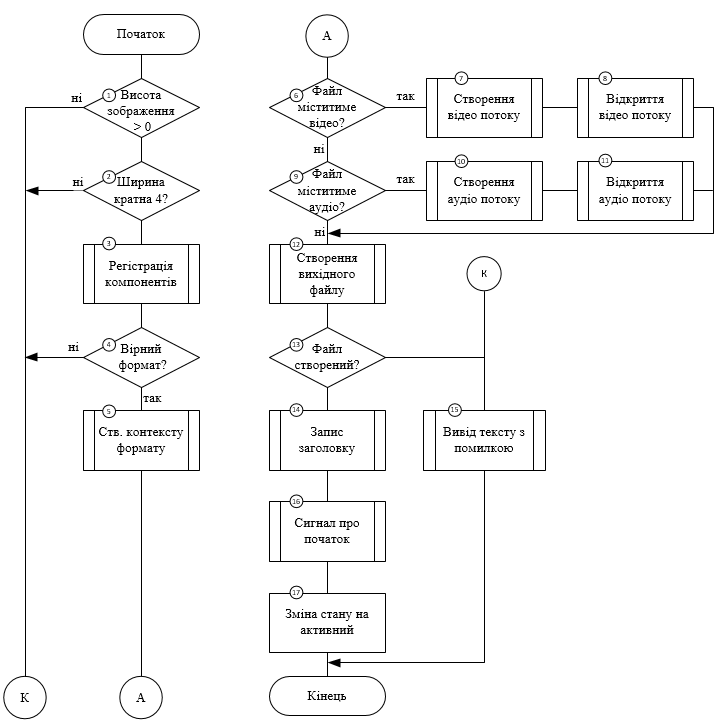


Рисунок 3.2 – Алгоритм ініціалізації бібліотеки для кодування відео/аудіо

Створення відео/аудіо потоку відбувається методом av\_new\_stream, параметром якого є AVFormatContext, який зберігає в собі усі наявні потоки. Метод виділяє пам'ять та ініціалізує потік, після чого варто налаштувати кодек (тип кодеку, формат пікселів, розширення вхідного зображення) для даного потоку (відео чи аудіо), за допомого коду наведеного нижче коду:

m\_videoCodecContext = m\_videoStream->codec;

m\_videoCodecContext->codec\_id = m\_outputFormat->video\_codec;

m\_videoCodecContext->codec\_type = AVMEDIA\_TYPE\_VIDEO;

m\_videoCodecContext->width = videoSize().width();

m\_videoCodecContext->height = videoSize().height();

m\_videoCodecContext->pix\_fmt = (PixelFormat)outputPixelFormat();

Для підвищення продуктивності процес пакування був винесений в окремий потік, а контроль здійснюється за допомогою допоміжного класу Encoder, при чому зв'язок між потоками реалізований стандартними методами Qt, а саме сигналами, слотами та мета об’єктами. Нижче наведений фрагмент коду, який здійснює виклик функції та передачу параметрів між об’єктами, що знаходяться в різних потоках.

QMetaObject::invokeMethod(d\_ptr, "encodeVideoFrame", Qt::QueuedConnection, Q\_ARG(QImage, frame), Q\_ARG(int, pts));

Вище наведений виклик методу encodeVideoFrame, в тілі якого і виконується кодування відео. Перед початком кодування зображення приводиться до потрібного формату (розширення та піксельний формат) за допомогою методу convertImage тіло якого наведено нижче:

m\_imageConvertContext = sws\_getCachedContext(m\_imageConvertContext, image.width(), image.height(), (PixelFormat)inputFormat, image.width(), image.height(), m\_videoCodecContext->pix\_fmt, SWS\_BICUBIC, NULL, NULL, NULL);

sws\_scale(m\_imageConvertContext, srcplanes, srcstride,0, image.height(), m\_videoPicture->data, m\_videoPicture->linesize);

Після чого зображення обробляється методом avcodec\_encode\_video, в якому відбувається формування відео пакета із низки таких зображень, який потім буде записаний у вихідний файл.

Крім того вище згаданий клас Encoder виконує захват зображень підготовленого у QGLWidget, методом отримання даних з подвійного буферу, а також виконує синхронізацію та зведення звукових доріжок. У свою чергу класи AudioDecoder, VideoDecoder виконують декодування відео і працюють у зворотному порядку.

* 1. Реалізація системи рендерингу та анімації

Головним елементом для виводу графіки є удосконалений клас QGLWidget який отримав назву OGLWidget. Він являє собою елемент візуального інтерфейсу (widget), який зазвичай використовується в стандартних компонентах користувацького інтерфейсу. Клас був розширений функціями OpenGL 3.0, методом наслідування від класу QOpenGLFunctions\_3\_0, що дало змогу доступитися до низькорівневого функціоналу відеоадаптера та значно розширити можливості стандартного класу за допомогою систем FBO, захвату пера, позиціювання, ефектів та їх тестування.

Система FBO реалізована на основі структури FBOWrapperStruct, що містить розширення текстури буферу, вказівники на кадровий буфер, буфер глибини, текстуру, а також статус даного FBO. Структура наведена нижче:

struct FBOWrapper{

GLuint frameBuffer;

GLuint depthStensilBuffer;

GLuint bindedTexture;

int errorStatus= -1;

int tWidth=0,tHeight=0;

};

Крім структури OGLWidget містить ряд функції, які полегшують процес переключення між FBO та виводом їх в основний буфер. Оскільки механізм FBO зберігає контент у текстурі, то її відображення виконується за допомогою методу drawTexture, який має наступні параметри:

* int x, y, z (координати х та y та глибина z елемента);
* int width, height (висота та ширина);
* GLuint texture (вказівник на текстуру);
* int angle (кут повороту елемента);
* float scaleX, scaleY (масштаб елемента по х та у);

Значення для цих параметрів отримуються з структури FBOWrapperStruct та об’єкту, що буде відображений. Сама ж функція складається з наступних етапів:

1. Перехід у режим відображення текстур і підключення до вибраної текстури:  
    glEnable(GL\_TEXTURE\_2D);

qglColor(Qt::white);

glBindTexture(GL\_TEXTURE\_2D, texture);

1. Збереження поточної матриці відображення, за допомогою методу glPushMatrix;
2. Застосування трансформацій:

glScalef(scaleX, scaleY, 1.f) (маштабування);

glRotatef(angle, 0, 0, 1) (поворот);

1. Вивід поточної текстури по координатам за допомого методу drawRect;
2. Відновлення матриці, методом glPopMatrix.

Кожен елемент сцени представлений об’єктом класу DrawElement, який містить поля які позначають його позицію в часі та просторі, а також дають можливість вивести його контент враховуючи трансформації та набір ефектів. Це досягається інкапсуляцією усіх складових, які повинен мати елемент сцени. Сам же контент, що відображатиметься на екрані від імені нового компоненту, підготовлюється у перевизначеному методі draw, який викликається у методі paint, який у свою чергу обгортає контент у набір із ефектів і виконується в основному циклі самої системи відображення. Алгоритм роботи paint показано на рисунку 3.3.

Даний клас реалізований в ролі інтерфейсу, що дає змогу наповняти його різним контентом і створювати різні компоненти сцени (анімований текст, зображення, тощо), при цьому залишати процес рендерингу унікальним і не змінювати внутрішні механізми відображення.

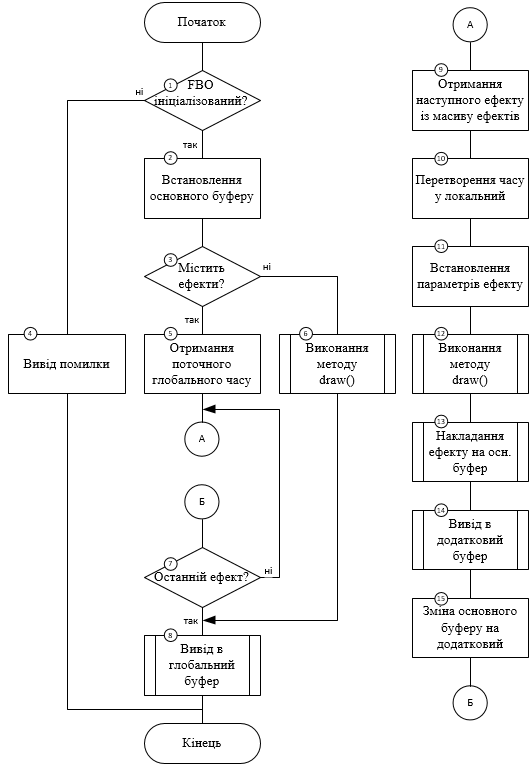


Рисунок 3.3 – Алгоритм відображення на сцені об’єкт DrawElement

Система ефектів побудована на шейдерах. Шейдер (англ. Shader) — це програма для одного із ступенів графічного конвеєра, що використовується в тривимірній графіці для визначення остаточних параметрів об'єкта чи зображення. Вона може включати в себе довільної складності опис поглинання та розсіювання світла, накладення текстури, віддзеркалення і заломлення, затінення, зміщення поверхні і ефекти пост-обробки. Програмовані шейдери гнучкі та ефективні. Складні на вигляд поверхні можуть бути відтворені за допомогою простих геометричних форм. Наприклад, шейдери можуть бути використані для малювання поверхні із тривимірної керамічної плитки на абсолютно плоскій поверхні.

Шейдери були розроблені для того, щоб мати змогу відтворювати складні відеоефекти з атомарних операцій. Попередниками шейдерів були процедурна генерація текстур (що широко застосовувалася в Unreal для створення анімованих текстур води і вогню) і мультитекстуровання. Але і ці механізми не забезпечують такої гнучкості, як шейдери. В даний час шейдери поділяються на чотири типи: вершинні, геометричні, параллаксні і фрагментні (піксельні). Але головна їх особливість, що вони розкривають весь потенціал відеоадаптера.

Вершинний шейдер оперує даними вершин Багатогранників. До таких даних, зокрема, відносяться координати вершини в просторі, текстурні координати, тангенс-вектор, вектор бінормалі, вектор нормалі. Вершинний шейдер може бути використаний для видового і перспективного перетворення вершин, генерації текстурних координат, розрахунку освітлення і т. д.

Фрагментний шейдер працює з фрагментами зображення. Під фрагментом зображення в даному випадку розуміється піксель, якому поставлено у відповідність деякий набір атрибутів, таких як колір, глибина, текстурні координати. Фрагментний шейдер використовується на останній стадії графічного конвеєра для формування фрагмента зображення.

Оскільки шейдер це програма, то як і всі програми пишеться за допомогою мови програмування. Шейдерна мова OpenGL носить назву GLSL (The OpenGL Shading Language). GLSL заснована на мові ANSI C. Більшість можливостей мови ANSI C збережено, також до них додано векторні та матричні типи даних, що часто застосовуються при роботі з тривимірною графікою. У контексті GLSL шейдером називається незалежно компільована одиниця, написана цією мовою. Це дає змогу динамічно завантажувати шейдери, після чого їх компілювати.

Код фрагментного шейдеру, що накладає прозору маску на усі пікселі наведено нище:

uniform sampler2D textureSampler;

uniform float animationKey;

uniform bool reverse;

void main(void){

vec4 col = texture2D(textureSampler, vec2(gl\_TexCoord[0]));

float alpha = animationKey;

if (reverse)alpha=1.0-alpha;

if (alpha>col.a)alpha=col.a;

gl\_FragColor = vec4(col.rgb,alpha);

}

Ключове слово, uniform, біля змінних надає можливість змінювати їх значення із зовні, за допомогою інструментів OpenGL. У головній функції спершу ми отримуємо колір поточного пікселя у форматі 4D вектора (RGB + ALPHA), методом texture2D, після чого в залежності від моменту часу встановлюється його прозорість, встановлюючи нове значення змінної gl\_FragColor.

Для роботи з шейдерами створено клас ShaderProgramWrapper. Основною його функцією є перетворити код у програму та застосувати її до контенту. Процес компіляції виконує метод initShader параметрами якого може бути шлях до файлів з кодом або ж сам код програми. Для створення об’єкту шейдеру слід скористатися метод glCreateShader параметром якого є тип шейдеру:

GLuint vertexShaderObj = glf->glCreateShader(GL\_VERTEX\_SHADER);

Після створення об’єкту слід завантажити код програми та скомпілювати:

glf->glShaderSource(vertexShaderObj, 1, code, codeLength);

glf->glCompileShader(vertexShaderObj);

Робота ж із самими ефектами із сторони користувача виконується за допомогою спеціального вікна UI наведеного на рисунку 3.4.

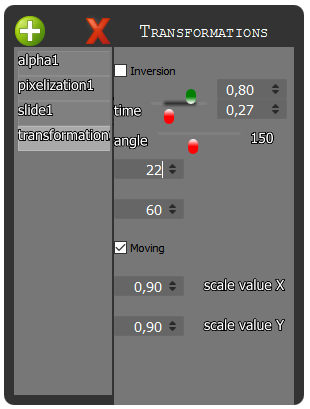


Рисунок 3.4 – Вікно роботи з ефектами

Вікно ділиться на три частини: кнопки управління, список ефектів та форми зміни параметрів ефекту.

Головний цикл роботи цієї системи обробляється в методі paintGL. Він обробляється по таймеру з фіксованою частотою у 5мс, з метою розвантаження центрального процесора та надати можливість обробити іншим не менш важливим компонентам програми. Основна мета цього методу це вивід усіх елементів сцени (службових таких як сітка та користувацьких) на екран монітора та внутрішній буфер програми, з метою його подальшого використання для утворення вихідного файлу. Алгоритм даного методу зображений на рисунку 3.5.

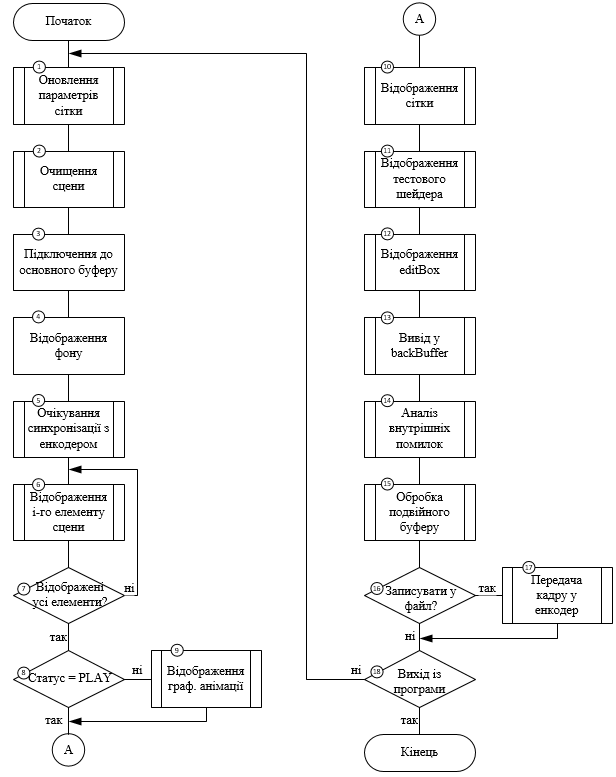


Рисунок 3.5 – Цикл роботи системи відображення

3.6 Реалізація системи захвату пера

Система захвату пера представляє собою звичайну програму для малювання, але з можливістю захвату послідовності та швидкості з якою користувач малює той чи інший об’єкт. Це дає можливість створювати власні анімовані зображення, які можна використати для того, щоб привернути увагу глядача до того чи іншого елементу сцени, або ж здійснити плавний перехід від одного елементу до іншого.

Процес малювання відбувається на вікні системи відображення, тому існує тісний зв'язок між цими системами. Для того щоб розпочати створення анімованої фігури, потрібно перейти у відповідний режим роботи, за допомогою меню чи кнопки на панелі швидкого доступу головного вікна програми. Сам процес відбувається за допомогою віртуальної кісточки із власним узором, в ролі якого виступає підготовлене зображення. Кісточка має ряд налаштувань в тому числі можливість змінити узор, його колір (тощо).

Для налаштування кісточки розроблено спеціальний користувацький інтерфейс, з можливістю перегляду налаштувань (верхня частина вікна), який наведено на рисунку 3.6.

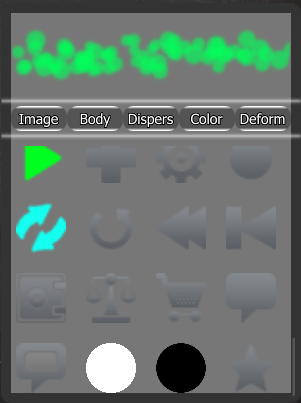


Рисунок 3.6 – Вікно вибору узору для створення анімованої фігури

Наведений вище інтерфейс містить декілька вкладок для різних налаштувань кісточки. На рисунку 3.5 зображена вкладка Image - вибору узору. Вкладка Body відповідає за налаштування розміру, чіткості, прозорості кісточки, а Dispers – за налаштування розносу та густоти кісточки при мазкові. Деякі налаштування можна відключати, тоді вони відображаються червоним.

Схожі зміни вносять параметри вкладки Deform : що вносить автоматичні деформацію розміру, кута повороту та зміну перспективи під час непереривного рисування. Колір узору можна змінити у вкладці Color. Колір задається у форматі HSL за допомогою повзунків Hue, Saturation, Light та у форматі RGB у текстовому полі. Процес налаштування та вигляд цих вікон зображено на рисунку 3.7.

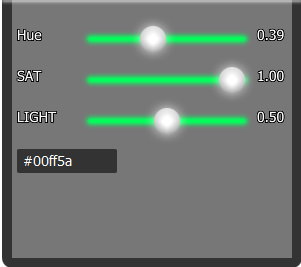
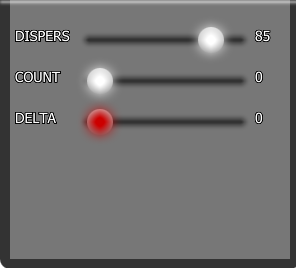
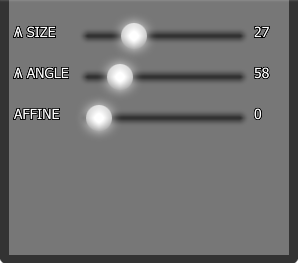
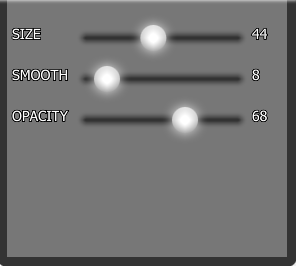


Рисунок 3.7 – Процес налаштування кісточки

Після налаштування кісточки можна розпочинати процес створення анмованої фігури, під час якої можна змінювати усі налаштування кісточки в тому числі узор. Після завершення процедури малювання можна переглянути отриману анімацію і вразі задоволенням результату зберегти її на диск. Після чого анімацію можна використовувати у проекті і застосовувати її як звичайний елемент сцени. Приклад створеної та використаної анімації наведено на рисунку 3.8.

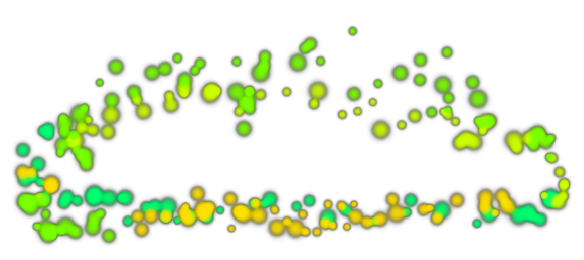


Рисунок 3.8 – Приклад створеної анімації

3.7 Реалізація системи управління елементами сцени

Дана система являє собою складний користувацький інтерфейс написаний, як і вікно налаштування кісточки описане вище, на основі QML і отримав назву TimeLine. Головними компонентами є треки, кнопки керування треком та елементи сцени. TimeLine має елементи управління у вигляді декількох панелей, зображених на рисунку 3.9.

Вказівник

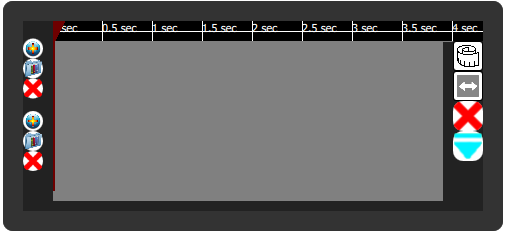
Часова шкала

Авто

розмір

Додати

текст



Додати

трек

Видалити

трек

Примусове

переміщення

Видалити

останній елемент

Додати елемент з бібліотеки

Рисунок 3.9 – Елементи управління вікна TimeLine

Елементи управління праворуч є статичними та глобальними для усієї сцени, а ті що ліворуч є особистими для кожного треку. Кнопка авто розміру вмикає автоматичний підгін часу закінчення чи початку блоку (елементи сцени, зображені на рисунку 3.9) згідно сусідніх у інших треках, що дозволяє вирівнювати часові інтервали, робота функції інтелектуальна. Кнопка примусового переміщення відповідає за дозвіл переміщення блоку у місця де немає достатнього вільного часового простору, в такому разі сусідній праворуч за ним блок та інші будуть переміщенні на інтервал, якого не вистачало для переміщення блоку. Вказівник вказує на поточний час відтворення.

Треки виконують роль шарів, що дозволяють відтворювати декілька елементів одночасно. В загальному треки виглядають так як показано на рисунку 3.10.

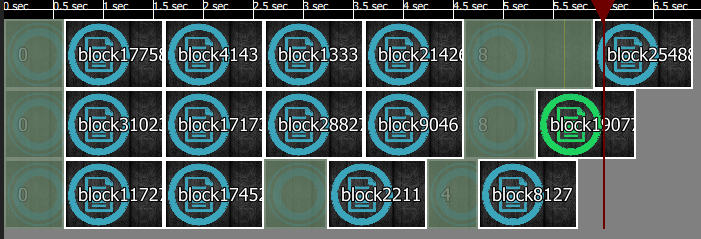


Рисунок 3.10 – Вигляд набору треків

Розташування елементів здійснювання за допомогою шкали зображеної у верхній частині TimeLine. Шкала може масштабуватися з кроком 0,6 секунди, та відображатися в секундах та мілісекундах. Від довжини блоку та місцем його розташування залежить час початку його відтворення та тривалість. Для того щоб змінити час початку відтворення, слід перетягнути елемент у відповідну позицію, а щоб змінити протяжність потягнути за один із кінців елемента. Причому процес перетягування супроводжується підсвічуванням зеленого кольору треку у який буде переміщено елемент.

За замовчуванням елементи які розташовані нижче, відображаються на сцені поверх тих елементів які знаходяться у вищому трекові. Це пов’ялено з тим, що стандартна глибина елемента прямо пов’язана з номером треку. Але параметр елемента можна змінити завдяки контекстному меню зображеному на рисунку 3.11.

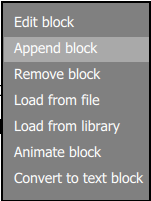
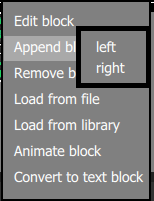


Рисунок 3.11 – Контекстне меню з відкритим пунктом «Append block» елементу сцени

Пункт «Append block» дозволяє додати, ще один текстовий блок праворуч або ліворуч від поточного, «Remove block» - видалити, «Convert to text block» - перетворити у текстовий тип блоку, «Animate block» - налаштувати анімацію. Наступні два дозволяють замінити цей блок новим, який буде завантажений з бібліотеки (рис. 3.12) або з файлу.

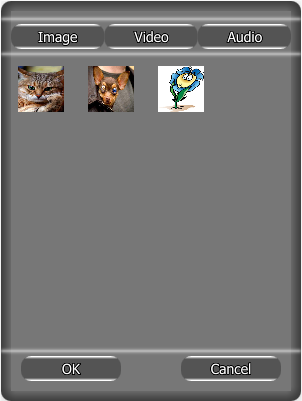


Рисунок 3.12 – Вигляд мультимедійної бібліотеки користувача

Перший пункт викликає модальне вікно (рис. 3.13), що дозволяє змінити основні налаштування блоку : розмір, позиція в часі та просторі (в тому числі глибину), а також задати ім’я блоку.

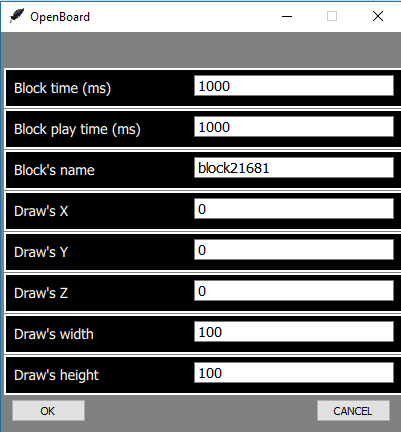


Рисунок 3.13 – Вікно налаштування блоку

3.8 Тестування програмного продукту

Першою перевіркою програми є перевірка на запуск. Результат запуску програми зображено на рисунку 3.14. З’являється вікно з вибором режиму роботи.

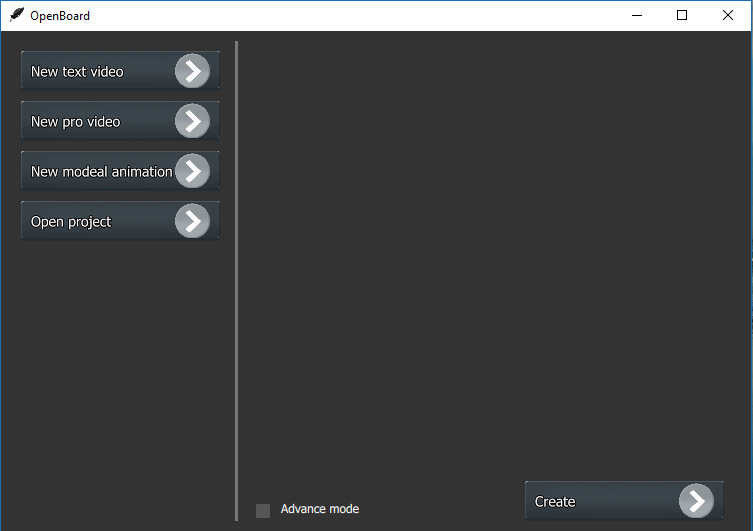


Рисунок 3.14 – Вигляд стартового вікна

В даному вікні користувач має вибрати, який проект він хоче створити або відкрити уже створений. Якщо вікно закрити, то програма продовжить роботу у режимі за замовчуванням (створить пустий проект у пам’яті та відкриє його) та відкриє головне вікно, що зображене на рисунку 3.15.

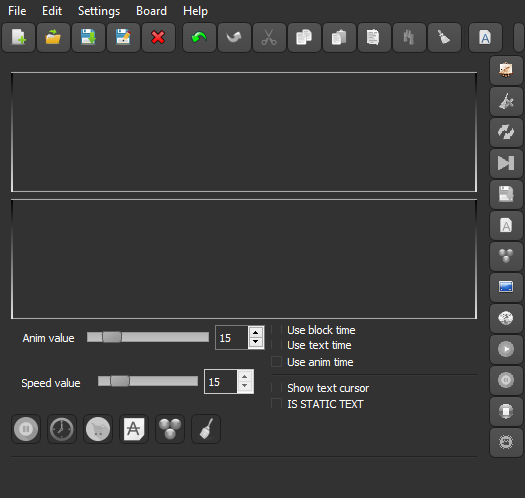


Рисунок 3.15 – Вигляд стартового вікна

Після чого слід натиснути на синю піктограму у правій частині вікна. Це призведе до запуску модулів керування елементами сцени та відображення (рис 3.16).

У сцену було додано декілька елементів різних типів, після чого проект був вдало збережений та знову завантажено з диску. Увімкнувши попередній перегляд почалось відтворення, із урахуванням розміщенням блоків (елементів) у часі, звук який був у сцену відтворювався. Було вибрано режим формування відеофайлу, в результаті якого утворилося вікно для вказання назви та розташування файлу.

В результаті після запуску запису, вікно TimeLine автоматично було приховано, для збереження обчислювальних ресурсів. Для перевірки було вирішено поставити перегляд на паузу, після чого відновити відтворення. Програма успішно опрацювала таку послідовність дій. На виході було отримано 5,5 секундне тестове відео у форматі mp4.

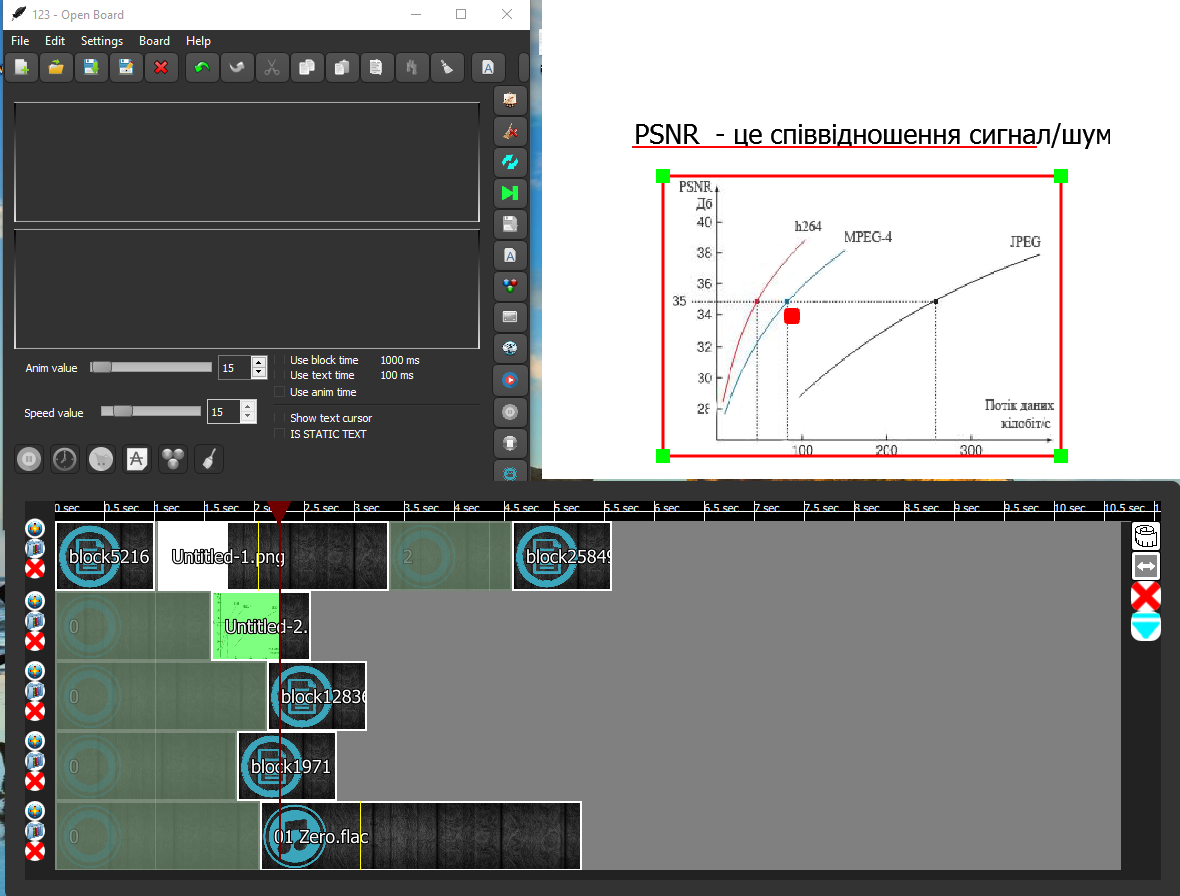


Рисунок 3.16 – Робота програми в штатному режимі

Після того, як було отримано вихідний відео файл, програму було закрито, але перед закриттям було виведено повідомлення про те, що не усі зміни були збережені із пропозицією їх збереження. Програма успішно виконала усі поставлені задачі та коректо завершила свою роботу.

Висновки

В даному розділі було проаналізовано вимоги до мови програмування та для бібліотеки класі, на основі було вибрано зв’язку С++ та Framework Qt. Після чого було здійснено вибір бібліотек для виводу зображення та кодування та декодування відео та аудіо. На основі вибраних бібліотек було розроблені модуль для кодування та декодування відео/аудіо, системи відображення, захвату пера та управління сценою.

В результаті було розроблено програмний продукт, який був успішно протестований.

# ВИСНОВКИ

У даній бакалаврській дипломній роботі виконано проектування та розробка комплексного програмного забезпечення для створення відеоуроків.

В результаті аналізу системи дистанційного навчання було визначено його суть, мета та призначення. Також було описано можливості та необхідність застосування такого методу навчання у різних умовах та різних прошарках населення.

Проаналізовано, поняття та місце комп’ютерної графіки та відео уроків у досліджуваній області. Дослідження проводилось з сторони особливостей сприйняття інформації людиною, що дало можливість розподілити види засобів навчання за типами сприйняття. Було розглянуті найпопулярніші підходи для створення відеоуроків, їхні недоліки та переваги. З розглянутого матеріалу було виділено всі переваги таких підходу для проведення дистанційного навчання.

Розглянуто особливості проектування програмного забезпечення з використанням об’єктно орієнтованого підходу за допомогою універсальної мови проектування UML. Побудовано діаграму використання, що відображає можливості користувача, а також ряд діаграм для визначення необхідного набору вимог та інструментів для розробки програмного продукту.

Обґрунтовано вибір набору бібліотек для кодування та декодування аудіо/відео потоків, виводу зображення та спрощенням обміну службовою інформацією між об’єктами програми. Використовуючи вибрані бібліотеки розроблені різноманітні модулі програми для кодування та декодування, відображення та формування відео та аудіо контенту, створення та відображення ефекті, анімації тексту та фігур. Розробка таких модулів надала можливість створювати власні та використовувати уже створені анімовані фігури у різних форматах, інтегрувати відео та аудіо фрагменти у сцену та багато іншого, що дозволить зробити відеоурок більш динамічним та таким що запам’ятовується.

Розроблена потужна система управління елементами сцени, в якій реалізовано можливості групування, дублювання елементів та їх властивостей, авто-корегування позицій та інші корисні функції, що дозволили підвищити швидкість розробки складних та навантажених сцен, з яких складаються відеоуроки. Також присутня система управління проекту, яка дозволяє не лише зберегти проект в максимально зручному форматі, а забезпечити обернену сумісність з проектами створених в попередніх версія програми, а вбудована система резервування дозволить відновити проект в разі його втрати або пошкодження.

Для роботи групами в офісі було розроблено систему синхронізації, що дозволить обмінюватися власними надбаннями з колегами в реальному часі, що також значно заощадить час розробки проекту, шляхом використання уже підготовленого мультимедійного контенту.

Визначено, що програмна реалізація системи дозволить оптимізувати процес підготовки, створення та розповсюдження відеоуроків, що значною мірою прискорить розвиток дистанційного навчання.

Результати дослідження бакалаврської дипломної роботи доповідались та обговорювались на науково-технічна конференцї професорсько-викладацького складу, співробітників та студентів Вінницького національного технічного університету (м. Вінниця, Україна, 2016р) та опубліковані у збірниках тез даної конференції.

# ЛІТЕРАТУРА

1. Вимятнін В.М. Iнформаційно-технологічне забезпечення : ДО / В.М. Вимятнін. – Відкрита і дистанційна освіта, 2000. – 84 с.
2. Гороховський О.І. Інформаційна технологія доставки контенту у системі комп’ютеризованої підготовки спеціалістів : монографія / О.І. Гороховський, О.Д. Азаров, Т.І. Трояновська. – Вінниця : ВНТУ, 2016 - 160 с.
3. Клименко В.В. Впровадження дистанційного навчання у вищих навчальних закладах / В.В. Клименко, М.А. Гайдучок. // Інноваційні технології в процесі підготовки фахівців. Матеріали Міжнародної науково-практичної інтернет конференції 03-04 квітня 2016 року : збірник наукових праць / Міністерство освіти і науки України, Вінницький національний технічний університет [та інш.]. — Вінниця: ВНТУ, 2016. – С. 77–78. – ISBN 978-966-641-663-94
4. Гайдучок М.А. Впровадження інформацiйно-комунiкацiйних технологiй у професійній освіті: переваги та недоліки, проблематика / М.А. Гайдучок, Зінчук Р.С. // Інноваційні технології в процесі підготовки фахівців. Матеріали Міжнародної науково-практичної інтернет конференції 03-04 квітня 2016 року : збірник наукових праць / Міністерство освіти і науки України, Вінницький національний технічний університет [та інш.]. — Вінниця: ВНТУ, 2016. – С. 43–46. – ISBN 978-966-641-663-94
5. Гайдучок М.А. Розробка комплексного мультимедійного додатку для створення відеокурсів [Електронний ресурс] / М.А. Гайдучок // XLV Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії. – 2016. – Режим доступу до тез : <http://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2016/paper/view/1147>
6. Гайдучок М.А. Розробка кросс-платфоремного додатку для захвату і передачі аудіо/відео потоку в режимі реального часу [Електронний ресурс] / М.А. Гайдучок // XLV Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії. – 2016. – Режим доступу до тез : http://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2016/paper/view/1148
7. Гороховський О.І. Методичні аспекти створення навчальної літератури для дистанційного навчання / О.І. Гороховський – К.,2007. – 543 с.
8. Выготский Л.С. Педагогическая психология / Л.С.Выготский, В.В. Давыдов. – М.: Педагогика, 2006. – 143 с.
9. Воронин Ю.А. Технические и аудиовизуальные средства обучения: Учебное пособие / Ю.А.Воронин. – Воронеж: Воронежский государственный педагогический университет, 2005. – 50 с.
10. Григорьев С.Г. Мультимедиа в образовании / С.Г.Григорьев, В.В. Гриншкун. – М.: Педагогика, 2008. – 68 с.
11. Комарова Н.И. Технические и аудиовизуальные средства обучения. Программа для студентов гуманитарных факультетов педагогических ВУЗов / Н.И. Комарова. – М.: МГПУ, 2004. – 74 с.
12. Носкова Т.Н. Аудиовизуальные технологии в образовании / Т.Н.Носкова. – СПб.: СПбГУКиТ, 2004. – 50 с.
13. Буч Г., Якобсон А., Рамбо Дж. UML. Классика CS / С. Орлов. – 2-е изд. – СПб.: Питер, 2006. — 736 с. — [ISBN 5-46900-599-2](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:%D0%98%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8_%D0%BA%D0%BD%D0%B8%D0%B3/5469005992).
14. Крэг Ларман Применение UML 2.0 и шаблонов проектирования – 3-е изд. / Крэг Ларман – М.: Вильямс, 2006. — 736 с. — ISBN 0-13-148906-2.
15. Блохнин С.М., Резников Ф.А. Цифровые видеокамеры, видеомонтаж и фабрика видеодисков дома: «Ulead Mediastudio Pro 7» / Блохнин С.М. – М.: Лучшие книги., 2004. – 58 с.
16. Боресков Алексей. Разработка и отладка шейдеров. / А. Боресков – БХВ-Петербург, 2006. – 488 с. – ISBN 5-94157-712-5.
17. Комп’ютерні мережі : навчальний посібник / [Азаров О. Д.,Захарченко С. М., Кадук О. В. та ін.] — Вінниця : ВНТУ, 2013. – 371 с. –ISBN 978-966-641-543-4
18. Семеренко В.П. Системне програмування мовою Асемблера. Лабораторний практикум. Част.1. / В.П. Семеренко – Вінниця: ВНТУ,  
    2003. –89 с.
19. Семеренко В. П. Системне програмне забезпечення. Системне об’єктно-орієнтоване програмування мовою С++. Курсове проектування. Самостійна та індивідуальна робота студентів / В. П. Семеренко. – Вінниця : ВНТУ, 2015. – 195 с.
20. Трей Нэш C# 2010 : ускоренный курс для профессионалов («Accelerated C# 2010») / Трей Нэш – М.: Вильямс, 2010. – с. 592. – ISBN 978-5-8459-1638-9
21. Герберт Шилдт Java 8. Полное руководство, 9-е издание = Java 8. The Complete Reference, 9th Edition. / Шилдт Герберт – М.: «Вильямс», 2015. – 1376 с. – ISBN 978-5-8459-1918-2.
22. Макс Шлее Qt 5.3 Профессиональное программирование на C++. / Шлее Макс – СПб.: «БХВ-Петербург», 2015. – с. 928. – ISBN 978-5-9775-3346-1.
23. FFmpeg API Documentation [Електронний ресурс] // Електронна документація. – Режим доступу: <http://www.ffmpeg.org/documentation.html>
24. Рэнди Дж. Рост OpenGL. Трёхмерная графика и язык программирования шейдеров. Для профессионалов. Питер, 2005. ISBN 5-469-00383-3, 0-321-19789-5
25. Vulkan 1.0.16 - A Specification [Електронний ресурс] // Специфікація стандарту. – Режим доступу: https://www.khronos.org/registry/vulkan/  
    specs/1.0/pdf/vkspec.pdf
26. Фленов М.Е. DirectX и C++. Искусство программирования / М.Е. Фленов – БХВ-Петербург, 2006 – с. 384

# ДОДАТКИ