МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Інститут післядипломної освіти
Кафедра інформаційних технологій

Кваліфікаційна робота бакалавра

на тему: Розробка додатку "Космічні війни"

Виконав студент групи КН-5
спеціальності 122 Комп'ютерні науки
Тахіров Тейфур Бахтіяр огли

Керівник Бучинська І.В.,
доктор філософії, доцент

Рецензент Ковальчук В.В.
д.ф.-м.н., професор

Одеса 2022

ЗМІСТ

ВСТУП

IT-фахівець широке поняття, що поєднує у собі представників багатьох професій, які працюють у галузі інформаційних технологій. Це розробники, програмісти, адміністратори мереж та баз даних, фахівці з робототехніки, модератори, інформаційної безпеки, web-дизайнери та 3D-аніматори. З проникненням інформаційних технологій у нові сфери діяльності, з'являються нові професії для IT-фахівців. За даними на сьогоднішній день і думку багатьох аналітиків [1] фахівці даної галузі є затребуваними та будуть затребувані у найближчому майбутньому. Зокрема, сьогодні у світі з'являється багато перспективних ігрових розробок та проектів.

GameDev (вимовляється "геймдев") – скорочення від Game Development (розробка ігор). У процес розробки ігрового додатку входить: розробка дизайну ігрового процесу (геймплею), програмування ігрового движка або використання готових рішень, розробка візуального концепту та його складових, створення графіки та моделювання об'єктів, створення музичного та звукового супроводу.

Ігрова індустрія продовжує розвиватися і з кожним роком її прибуток на ринку збільшується як у світі, так і, в Україні Таким чином, ідея створення цікавої та нової гри є перспективним напрямком.

Метою бакалаврської роботи є розробка гри «Космічні війни» з гнучкими налаштуваннями, в жанрі Аркада у жанрі «Аркади» з метою розвитку моторики та реакції користувача.

Для досягнення цієї мети мають бути вирішені такі завдання:

1) здійснити постановку ігрового завдання;
2) виконати огляд аналогів і програмних засобів розробки комп'ютерної гри;
3) спроектувати програму;
4) реалізувати та скласти документацію (посібник користувача) на додаток.

Кваліфікаційна робота бакалавра складається з вступу, 5 розділів, висновків 55 сторінок, 18 джерел, 35 рисунків, 2 таблиці.

1 РОЗРОБКА МУЛЬТИМЕДІЙНОГО ІГРОВОГО ДОДАТКУ

В результаті кваліфікаційної роботи бакалавра потрібно створити гру в жанрі "Аркада". Гра повинна бути зроблена у 2D-стилі. При запуску програми гравець має певну кількість здоров'я та нульовий лічильник очок. Гравець за допомогою пострілів (атаки) повинен знищувати ворогів, що з'являються, і не допускати їх пересування до нижньої межі екрану. Якщо ворог дійде до кордону або зачепить гравця, здоров'я гравця зменшиться. При знищенні ворога (потраплянням у нього кулі) лічильник очок збільшується. Коли здоров'я гравця впаде до певного рівня (нуля), гра закінчиться. Надалі користувач може запустити гру знову.

Особливістю гри є те, що користувач може змінювати текстури гравця, ворогів та куль. Запускаючи гру, користувач може перейти в меню налаштувань гри та явно вказати текстури (картинки) моделей ворогів та гравця. Якщо користувач не зайде в меню налаштувань або не вкаже свої моделі, то буде використано стандартні параметри (моделі).

Основні положення гри:

- головна мета гри – набрати найбільшу кількість очок;
- на початку гри гравець має показник здоров'я, що дорівнює 3;
- якщо ворог досягне нижньої межі гри, показник очок зменшиться на одиницю;
- за досягнення показника здоров'я значення 0 – гра закінчується;
- щоб збільшити показник очок на одиницю гравцю, потрібно знищити одну ворожу одиницю;
- для знищення ворога гравець повинен використати кулю;
- ворог постійно прагне досягти нижньої межі гри.

2 ОБГРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЇ І ОПЕРАЦІЙНОЇ СЕРЕДОВИЩЕ

2.1 Аналіз ігрової промисловості

В теперешній час ігрова індустрія [2] – це один із найширших сегментів цифрового контенту у світі. Ця сфера постійно розвивається і зростає (рис 1). Так, за підсумками 2021 року –ігровий ринок у світі склав близько 100 млдр доларів.

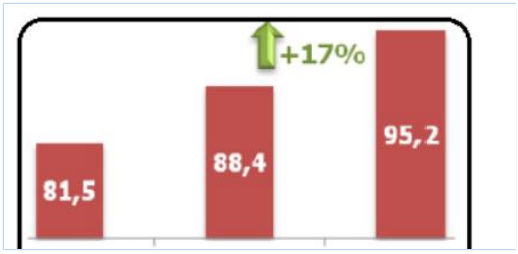

Рисунок 1 – Світовий ринок ігор у 2017-2021 роках

За результатами 2019-2020 року, ігровий ринок в Україні є значущим напрямком, який розвивається. При цьому обсяг ігрового ринку продовжує просуватис з 2012 року (рис 2) [3]. Ринок ігор Україні є досі привабливим інвестиційно.

Рисунок 2 – Динаміка зростання ринку гри 2012 по 2020 рік

Ринок тільки недавно став розвиватися у цьому напрямі. Частка України у світовому ринку ігор становила близько 1.1% (рис 3), проте стабільне зростання Російського ігрового ринку залучає ігрові компанії заходу.

2.2 Вибір основнної платформи

У теперешній час є можливість виділити 3 основні ігрові платформи:

- персональні комп'ютери;
- мобільні пристрої;
- консолі.

З появи комп'ютера почалося зародження ігор, тому основною аудиторією ігор є користувачі ПК. Тому, за даними на 2015 [4], більше половини ігрового ринку належить платформі персональних комп'ютерів (рис 3).

Рисунок 3 – Ігровий ринок на різних платформах у 2015 р.

При аналізі ринку персональних комп'ютерів було здійснено

приблизний підрахунок величини операційних систем, що використовуються на них [5]. Результати аналізу представлені на рис. 4.

З рис. 4 видно, що основна більшість користувачів ПК обирають ОС сімейства Windows.

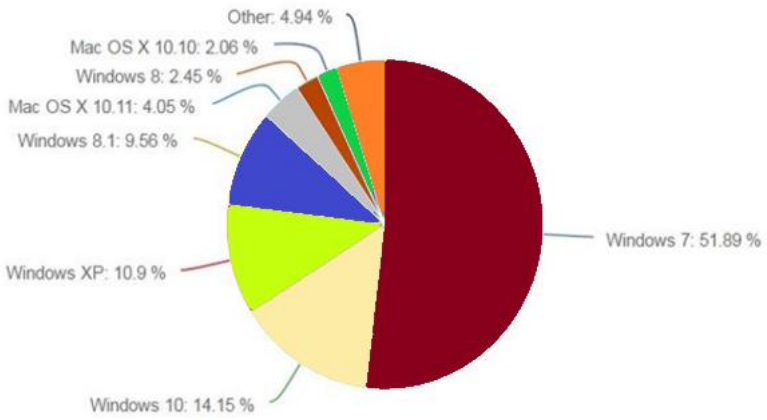

Рисунок 4 – Операційні системи, які використовуються

2.3 Вибір жанру гри

У світі на сьогодні виділяють такі жанри ігор:

– азартні;
– бойовики;
– спортивні;
– стратегії;
– аркади.
– головоломки;

За різними даними більша частка користувачів віддають перевагу жанру "Аркада" [6], тому що аркади є досить популярними щодо інших жанрів з наступних причин (рис. 5):

1. Жанр аркади не вимагають дуже потужних технічних характеристик:

Не у всіх користувачів є можливість купувати сучасні та потужні пристрої з відповідними характеристиками. Багато користувачів відзначають,

що чим потужніша гра, тим більше вона гальмує. Таким чином, ігровий процес є досить скрутним і неприємним користувачеві, що відбиває бажання грати і, тим самим, зменшує рейтинг програм, що в свою чергу зменшує зацікавленість нових користувачів (гравців)встановлювати цю програму. Найчастіше аркади не вимагають серйозних технічних характеристик, тому є можливість користувачам насолоджуватися ігровим процесом без будь-яких затримок, зависань та інших проблем, пов'язаних із продуктивністю.

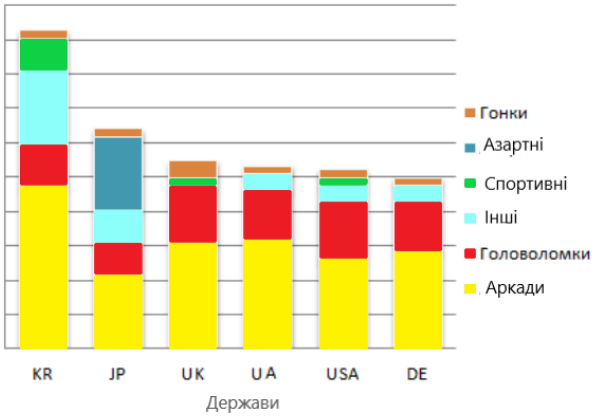

Рисунок 5 – Статистика популярністі ігрових жанрів

2. Жанр аркади найчастіше не займає багато часу:

Усі користувачі ігрових додатків (гравці) займаються якоюсь діяльністю (працюють, навчаються тощо). Таким чином, гравці не завжди мають вільний час, щоб проводити його за іграми. Аркади створені для людей, які не мають можливості проводити за грою достатню кількість часу. Зазвичай цей жанр дозволяє гравцеві зайняти його час на якийсь період. Тим самим більшість користувачів називають аркади – «ТаймКіллерами», так як вони є цікавими і незамінними помічниками весело провести період часу.

3. Аркади цікаві:

Чимало користувачів відзначають, що жанр аркада різносторонній. Він може включати логічні загадки, красиву графіку або цікавий ігровий процес (геймплей), тобто. не дозволяють гравцеві занудьгувати.

Таким чином, жанр для гри «Космічні війни» вирішено вибрати «Аркада».

2.4 Вибір програмної складової програми

В даний час існує безліч програмного забезпечення, що дозволяє розробляти власні проекти різного характеру, у тому числі комп'ютерних ігрових додатків. Виділяють дві основні групи з них:

– використання програмного компонента (комплексу), що дозволяє створювати ігровий додаток (ігровий двигун);

– створення власного «движка» за допомогою мови програмування та спеціалізованих бібліотек.

Розглянемо кожну з груп докладніше.

2.4.1 Опис движка Unity 3D

Unity 3D [7] – це потужний інструмент для створення програм та ігор під різні платформи. Є можливість створювати як 2D, і 3D проекти. Є підтримка DirectX і OpenGL, підтримує багато різних форматів даних. Unity 3D підтримує такі мови програмування як C#, JavaScript. Основне робоче вікно Unity 3D представлене рис. 6.

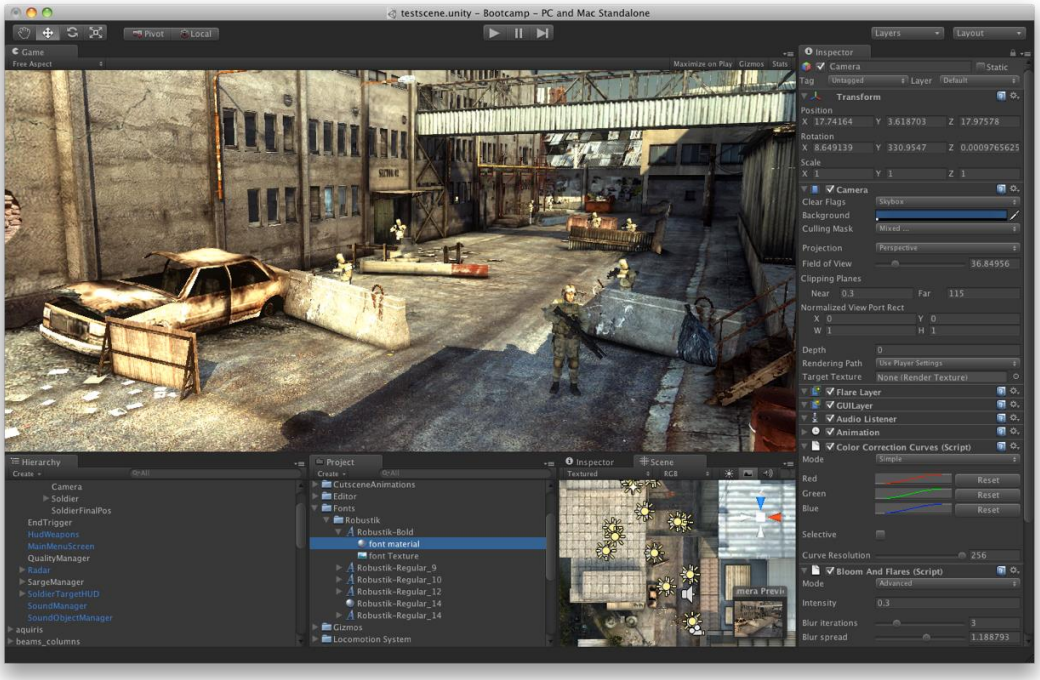

Рисунок 6 – Робоче вікно ігрового движка Unity 3D

Переваги:

- невеликі витрати ресурсів (порівняно з іншими "движками");
- кросплатформність;
- велике російськомовне суспільство;
- регулярні поновлення.

Недоліки:

- має малий набір інструментів;
- потребує пристойних технічних комплектуючих;
- платна комерційна ліцензія;
- незручна робота при створенні 2D - Програми.

2.4.2 Опис движка CryEngine

CryEngine [8] – найпотужніший із сучасних ігрових движків, забезпечує фотореалістичну графіку з підтримкою DirectX 12 та шейдерів (тіней). Третя версія "движка" створена в 2009 році, розповсюджується платно. П'ята версія "движка" вийшла наприкінці 2016 року і має умовно безкоштовну ліцензію. Основним напрямом цього засобу є створення ігор жанру "Шутер". Робоче вікно CryEngine представлене рис. 7.

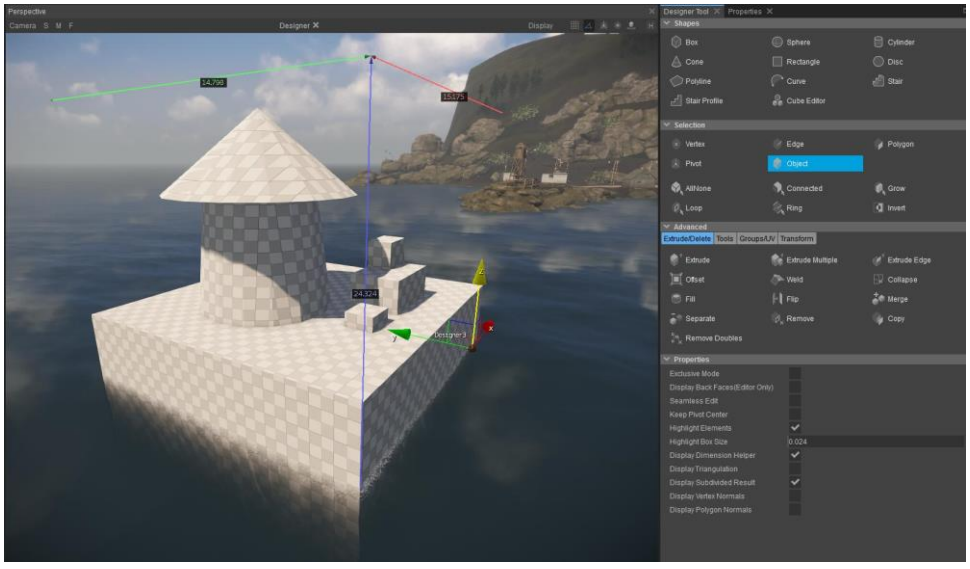

Рисунок 7 – Робоче вікно ігрового двигуна CryEngine

Переваги:

- високі показники графіки;
- кросплатформність.

Недоліки:

- має малий набір інструментів;
- спрямовано створення певного жанру ігор;
- самий ресурсомісткий «движок» з усіх представлених.

2.4.3 Опис движка Unreal Engine

Unreal Engine[9] – ігровий «движок», що розробляється та підтримується компанією Epic Games. Надає великий набір інструментарію для створення 2D та 3D проектів. Починаючи з четвертої версії, – поширюється безкоштовно. Однак, поки розробник не випустить свій перший комерційний продукт на основі UE4. Є головним двигуном більшості сучасних ігор. Робоче вікно ігрового "движка" Unreal Enginge представлене рис. 8.

Переваги:

- високі показники графіки;
- кросплатформність;
- великі та регулярні оновлення;
- різноманітний інструментарій.

Недоліки:

- високі технічні вимоги;
- складність у освоєння;
- мала кількість російськомовної документації;
- комерційну ліцензію.

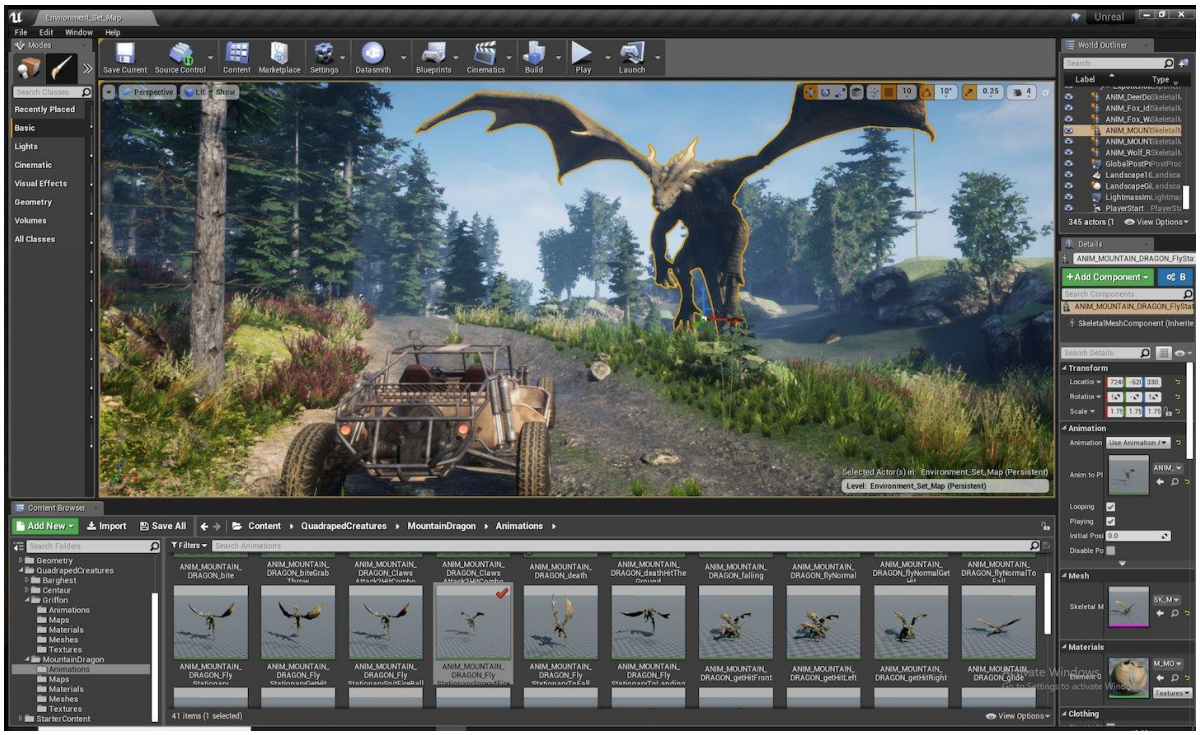

Рисунок 8 – Робоче вікно ігрового двигуна Unreal Engine

Всі розглянуті ігрові двигуни в основному призначені для створення 3D додатків і вимагають високі показники технічних вимог, тому використовувати їх недоцільно (у багатьох користувачів ПК відсутні сучасні та потужні комплектуючі). Перейдемо до розгляду іншої групи програмного забезпечення для створення ігрової програми.

2.5Аналіз мов програмування (API)

Для початку варто визначитися яку мову програмування використовувати [10]. На сьогоднішній день виділяють такі мови програмування як основу подальшої розробки:

2.5.1 Опис C

C# дозволяє стартувати розробку швидше, але це дозволяє швидше отримати прототип рішення. Швидкість розробки на початкових етапах проекту значно вища порівняно з іншими мовами.

C# спроектований бути кросплатформним, проте його розвиток не

пішов у цьому напрямку, тому під Windows утворилася досить повна

.net інфраструктура, а інших платформах рівноцінної інфраструктури не з'явилося.

При розробці невеликих проектів продуктивність C# не поступається іншим мовам програмування, однак за збільшення вихідного коду, алгоритмів і т.д. – швидкість роботи програм значно падає.

C# має пристойну кількість бібліотек зі старту, що істотно полегшує розробку.

2.5.2 Опис HTML та JavaScript

Є незамінними під час створення простих веб-додатків або ігор, проте на цьому переваги цих коштів закінчуються.

2.5.3 Опис C++

Є основною мовою програмування розробки ігор, оскільки дозволяє здійснити повний контроль за засобами і логікою. Є абсолютним рекордсменом з продуктивності, по кросплатформенності, за сумісністю та за кількістю існуючих бібліотек серед інших мов програмування. Однак багато програмістів виділяють головний його недолік - складність освоєння цієї мови. Зведемо отримані дані у табл. 1.

Таблиця 1 –Порівняння мовпрограмування

Критерій	**C #**	**JavaScript/HTML**	**C++**
Продуктивність	3☆	2☆	5☆ (максимальна)
Кількість бібліотек	3☆	2☆	5☆
Зручність програмування	4☆ (легка)	3☆	3☆
Складність освоєння	4☆ (легка)	4☆	3☆
Кросплатформеність	3☆	5☆ (повна)	4☆

Кожна мова програмування добре себе показують у певних задачах.

Наприклад: веб-розробка (JavaScript) не має прихильності до операційної системи; якщо пріоритетом розробки стоїть швидкість (отримання прототипу докладання), слід використовувати C#. Як мову програмування для створення ігор було вирішено використовувати мову C++, так як він підтримує величезну кількість бібліотек і має хорошу продуктивність.

На мові C++ існують два основні API для створення графіки:

1) SFML [11]

Проста та кросплатформова мультимедійна бібліотека. Включає модулі для програмування ігор і мультимедійних засобів. Складається з п'яти модулів:

- System – модуль керує часом та потоками. Є базовим, тобто потрібним для всіх модулів;
- Window – керування вікнами та виведення інформації користувачеві;
- Graphics – виробляє виведення інформації (зображень, геометричних фігур);
- Audio – бібліотека для роботи зі звуком;
- Network – бібліотека для роботи із мережею.

2) Qt Graphics [12]

Більш потужний інструмент (порівняно з SFML). Входить до складу бібліотеки Qt, яка являє собою інструментарій розробки ПЗ мовою програмування C++ і має кросплатформність. Qt надає програмісту як зручний набір бібліотек класів, а й певну модель розробки додатків, певний каркас їх структури, що істотно знижує появу помилок у додаток (відпливу пам'яті, незакриті файли тощо.). Бібліотека Qt має власну IDE (середовищем розробки) – Qt Creator, що значно спрощує розробку додатків. Оболонка Qt Creator представлена рис. 9.

Існує спосіб об'єднати Qt і SFML, однак, як засіб розробки, був обраний Qt, так як він має ширший функціонал.

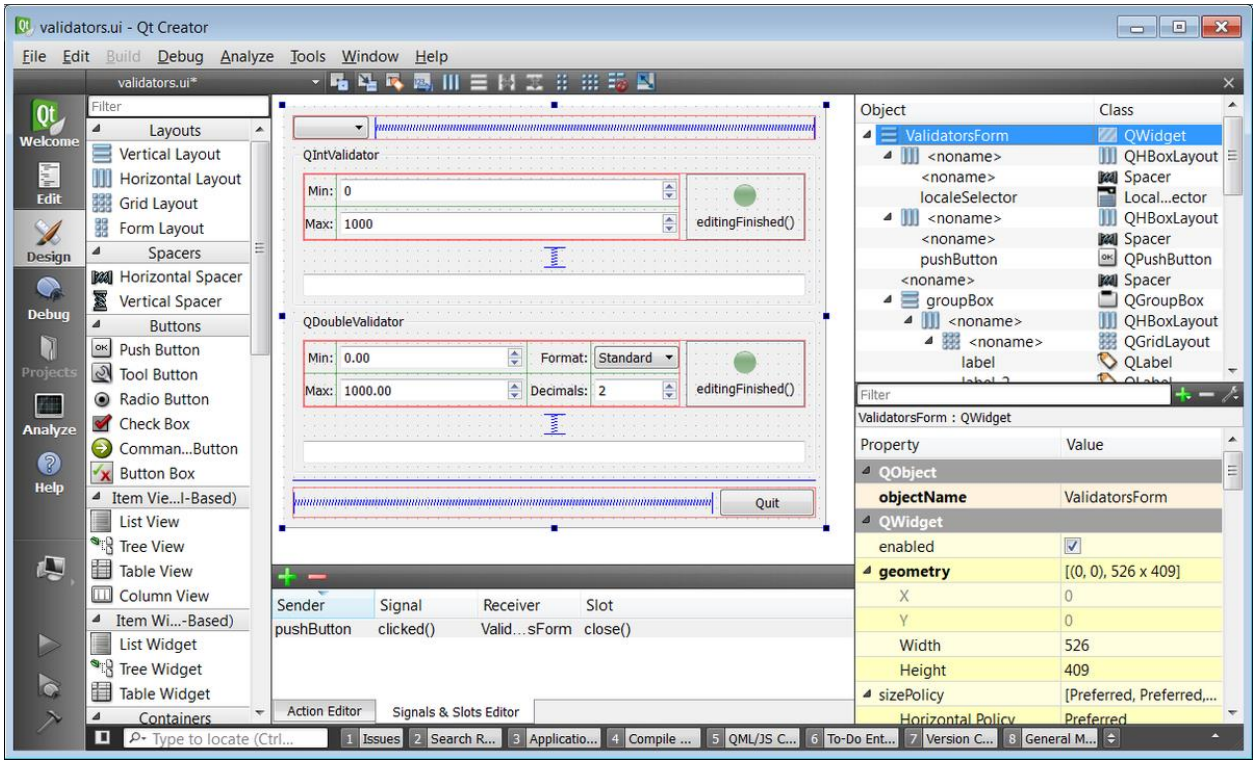

Рисунок 9 – IDE Qt Creator

3 ОГЛЯД АНАЛОГІВ

На сьогоднішній день існує безліч ігрових програм жанру аркада. Розглянемо найпоширеніші з них.

3.1 Аналіз Space Invaders

Найпопулярнішою аркадою є Space Invaders (рис. 10), яка була однією з перших у цьому жанрі [13]. Мета гри – знищити всіх ворогів на екрані. Спочатку гра мала просту графіку та показник здоров'я, при досягненні нульового значення якого – гра завершувалася, проте пізніше вона удосконалилася (були введені нові колірні схеми та показник очок) (рис. 11).

Рисунок 10 – Скріншот гри Space Invaders

Недоліки:

- користувач не має змоги змінювати ігрові моделі;
- користувач не може змінювати логіку роботи гри (зміна складності рівня, зміна траєкторії пересування ворогів);
- у першій версії гри був відсутній показник здоров'я;
- гра має примітивні моделі.

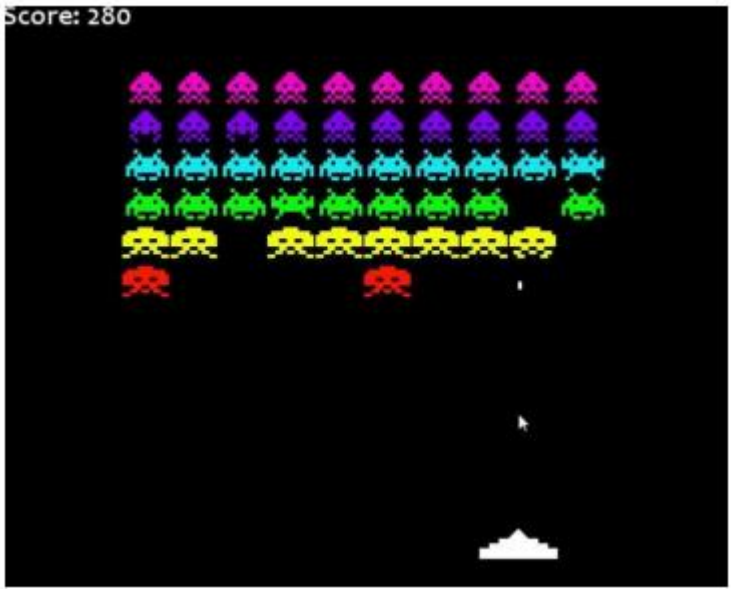

Рисунок 11 – Скріншот покращеної Space Invaders

3.2 Аналіз Radiant

Аналог гри Space Invaders випущений у 2010 році. Ціль гри [14] – знищити всіх ворогів на екрані (рис. 12). Головна особливість гри – покращена 2D графіка. У міру збільшення часу гри збільшується і складність – з'являються більше ворогів, однак, у користувача немає можливості змінювати моделі гри.

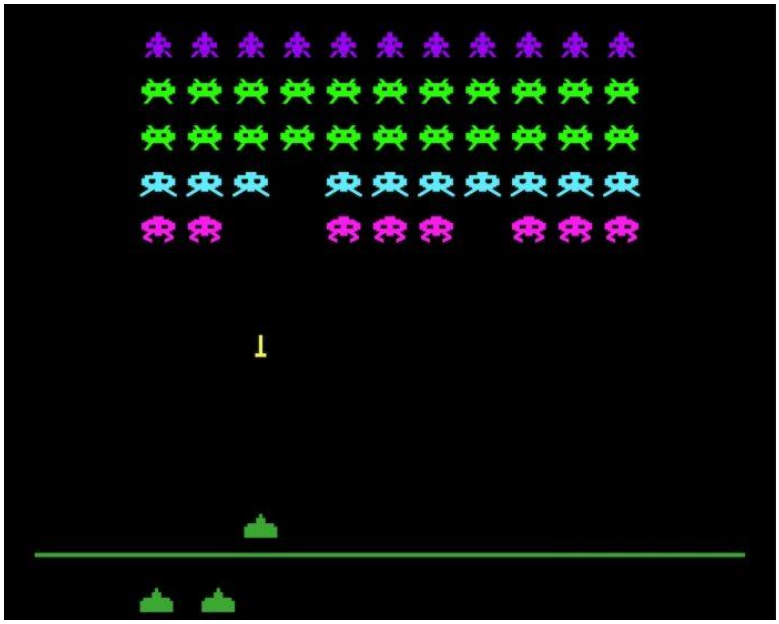

Рисунок 12 – Скріншот гри Radiant

Переваги:

– покращена 2D-графіка;
– є наростання складності в міру проходження.

Недоліки:

– у грі присутні лише стандартні однотипні моделі;
– складністьприсутній, проте, користувача немає можливості вибрати її самому;
– відсутня можливість обрати звуковий супровід.

3.3 Аналіз Luxor

"Luxor"[15] – серія аркадних комп'ютерних ігор, розроблених компанією MumboJumbo і виконані в 2D – стилі, що базуються на міфології Стародавнього Єгипту. Завдання гравця – винищити сферичних об'єктів, що з'являються на екрані, до того, як вони досягнуть піраміди (рис. 13).

У грі є адаптивна складність (складність збільшується з рівнем), однак, користувач не має можливості змінювати кулі (ігрові об'єкти) на певний колір або інші фігури.

Рисунок 13 – Скріншот гри "Luxor"

Переваги:

– присутні елементи головоломок (гра розвиває логіку);
– є наростання складності в міру проходження.

Недоліки:

– у грі є тільки шароподібні моделі. У користкувача намає можливості замінити їх;
– складність присутня, але, у користувача немає можливості вибрати її самому.

3.4 Аналіз Beyond Space

«Beyond Space» [16] – космічна аркада для Android, випущена у 2015 році. Головна особливість гри – 3D-графіка. Головна мета гри – знищити інші кораблі. Є можливість кастомізувати корабель (змінювати колір корабля). У грі присутні місії різного рівня складності – користувач переходить до легкого режиму до складного (рис. 14).

Рисунок 14 – Скріншот гри «Beyond Space»

Переваги:

– барвиста 3D графіка;

– присутня часткова зміна моделі корабля (кастомізація).

Недоліки:

– гра потребує потужних технічних характеристик;
– гра неоптимізована (присутні проблеми зпродуктивністю);
– складність присутня, але у користувача немає можливості вибрати певний її рівень для кожної місії.

3.5 Опис Чужий у космосі

"Чужий у космосі" [17] – це гра-аркада, випущена в 2013 році. Хоробрий космодесантник має провести вчених у безпечні бункери, доки до них не дісталися місцеві жуки переростки. Для виконання цього непростого завдання гравцеві дозволено користуватися всілякою зброєю: бензопилою, ракетами, кулями.

Гра має більш продуману графіку (порівняно з першими трьома іграми), а також має систему покращення зброї (на модель гравця додаються нові елементи, а також змінюється модель кулі).

Переваги:

– продумана 2D-графіка;
– присутня часткова зміна моделі корабля зброї (Кастомізація);
– є наростання складності в міру проходження.

Недоліки:

– складність присутня, однак, у користувача немає можливості задати її на певному рівні;
– відсутня можливість вибрати звуковий супровід у грі.
– у грі не можна вибрати певні моделі ворога.

Рисунок 15 – Скріншот гри «Чужий у космосі»

В результаті огляду вище перерахованихігрових рішень було виділено їх функціональні складові та винесено до табл. 2. При аналізі розглянутих ігор було виділено такі особливості жанру:

– аркади включають або набір місій, або нескінченний режим
– ігри;
– основна мета більшості аркад - набрати найбільшу кількість
– окулярів;
– щоб збільшити показник очок необхідно знищити модель ворога за допомогою функції випуску кулі (атаки);
– майже у всіх аркадах є показник здоров'я, який зменшується або по досягненню моделі ворога певної місцевості, або при досягненні самого гравця.

Таблиця 2 –Порівняння особливостей ігор жанру «Аркада»

Назва гри	**Графіка**	**Сюжет**	**Можливість змінювати моделі**	**Можливість змінювати поведінку ворогів**
Space Invaders	2D	Нескінченний режим	-	-
Radiant	2D	Нескінченний режим	-	-/+ Присутня збільшення складності
Luxor	2D	Набір місій	-	-/+ Присутнє збільшення складності
Beyound Space	3D	Набір місій	-/+ Додавання нових модулів	-/+ Присутнє збільшення складності
Чужий у космосі	2D	Нескінченний режим	-/+ Додавання нових модулів	-

З табл. 2 представлено, що більшість аркад не надають користувачеві можливість змінювати поведінку ворогів чи явно ставити рівень складності, і навіть змінювати моделі у грі.

4 РОЗРОБКА ДОДАТКУ

Розробка гри складалася з чотирьох етапів:

4.1 Розробка концептуальної схеми

4.1.1 Формування цілей та завдання проекту

На цьому етапі продумується функціонал та можливості. Формується жанр та інші особливості гри. Як жанр гри було обрано напрям «Аркада», т.к. вони досить поширені, не потребують великого часу гри (геймплея) і дозволяють гравцям скоротати час (ТаймКіллери). Таким чином, на даному етапі для гри "Космічні війни" були сформовані такі твердження:

- гра представлятиме собою Арконойд (жанр «Аркада»);
- у грі присутні гравець, вороги, зброя (кулі);
- у грі врахована система окулярів та здоров'я;
- за досягнення нульового показника здоров'я - гра закінчиться;
- мета гри ─ набрати найбільшу кількість очок.

При аналізі інших аркад було виявлено такі судження:

- майже у всіх аркадах є адаптивне нарощування складності в міру проходження, однак, явне завдання складності відсутнє;
- у раніше розглянутих проектах відсутня можливість змінювати логіку поведінки ворогів (напрямок руху);
- у деяких іграх є часткова зміна моделей гри, але, найчастіше, ці зміни незначні.
- у всіх розглянутих примірниках відсутня можливість вибору звукового супроводу.

Таким чином, як особливості проекту «Космічні війни» виступатимуть такі функції:

1) вигрі буде можливість змінювати стандартні моделі на користувальницькі. Таким чином, користувач (гравець) зможе змінювати:

- модель гравця;

– модель ворога;

– модель кулі;

– заднє тло (BackGround).

2) у грі буде можливість змінювати поведінку ворогів (напрямок руху).

3) у грі буде можливість змінювати рівень складності;

4) вігре буде присутня можливість змінити звуковий і музичний супровід.

4.1.2 Розробка UML-діаграм

Для проектування програми був використаний мову графічного опису об'єктного моделювання UML [18], так як він дозволяє продумати та описати архітектуру проекту без програмного коду, при цьому не потребує особливих знань для їхнього розуміння. Як основний засіб опису проекту були обрані та побудовані діаграми використання, які показують функціональність майбутньої системи. Варіанти використання показують можливі взаємини між системою та користувачем. Вони відображають зовнішній інтерфейс системи та вказують форму того, що система має зробити. Діюча особа (актор) – це елемент, який не є системою, а взаємодіє з нею, через особливий інструмент – варіант У ході проектування було виділено наступний актор:

Гравець ─ користувач програми, який має повний доступ до функцій відеоігри. Може брати участь у грі, керувати налаштуваннями програми.

Загальна діаграма використання представлена рис. 16

Рисунок 16 – Загальна UML діаграма використання

Розпишемо кожен варіант використання детальніше. Для цього треба подати діаграми відносин варіантів використання. При запуску рівня користувачем, він може зробити певні дії, представлені на рис. 17.

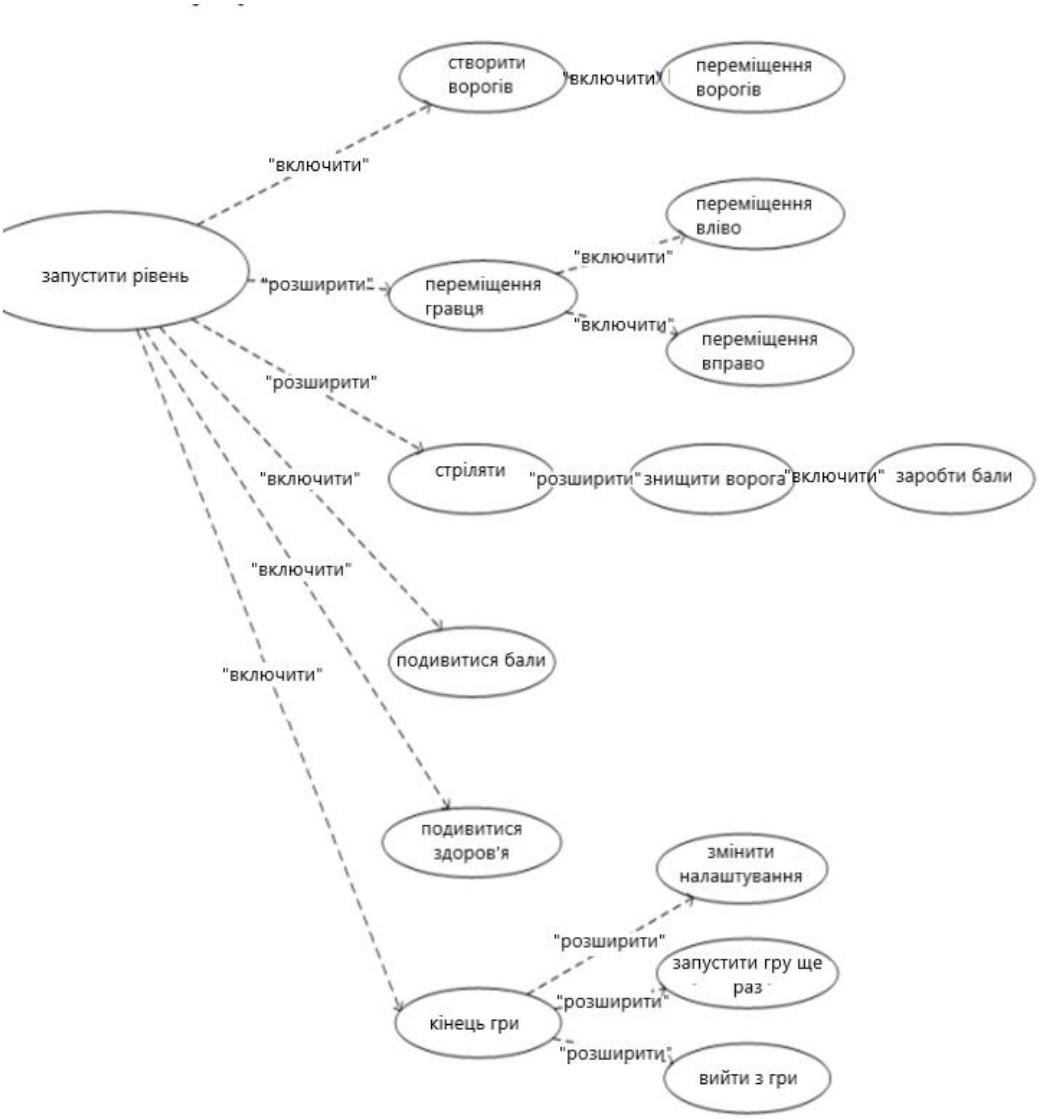

Рисунок 17 – Діаграма відношення варіантів використання для дії «Запустити рівень»

Як інший варіант користувач може використовувати дію "Запустити налаштування візуальних ефектів". В даному варіанті використання у дійової особи (гравця) є можливість настроїти візуальні компоненти прироження. При налаштуванні деяких компонентів необов'язково налаштовувати деякі функції, позначені ставленням «розширити». Діаграма відносин варіанта

використання «Запустити налаштування візуальних ефектів» представлена рис. 18

Рисунок 18 – Діаграма відношення варіантів використання для дії «Запустити налаштування візуальних ефектів»

У варіанті використання "Запустити налаштування логіки", користувачеві надається можливість змінювати складність гри, а також налаштовувати траєкторію пересування ворогів та дію "Woopsy".

Woopsy – спеціальна функція, що викликає вказану гравцем картинку для досягнення певної дії (по досягненню певної кількості очок або через деякий час, вказане користувачем). Діаграма відносин варіанти використання «Запустити налаштування логіки» представлена рис. 19.

Рисунок 19 – Діаграма відношення варіантів використання для дії "Запустити налаштування логіки"

4.1.3 Розробка бізнес сутностей та логіки

В результаті проектування програми «Космічні війни» було виділено бізнес сутності та логіка, представлені в табл. 3.

Таблиця 3 – опис сутностей та логіки

Бізнес сутності		Бізнес логіка	
1	Гравець	1	Калькулятор очок
2	Куля	2	Відстеження положення кулі
3	Показник здоров'я	3	Відстеження становища ворогів
4	Показник очок	4	Відстеження колізій (перетину куль та ворогів)
5	Вороги	5	Інтерфейс для надання к-сті очок
6	Текстури	6	Інтерфейс для надання к-сті здоров'я
7	Музика	7	Інтерфейс головного меню
		8	Інтерфейс роботи з вибором текстур
		9	Інтерфейс налаштування логіки ворогів

4.2 Розробка гри із використанням стандартних об'єктів

4.2.1 Створення Scene та View

Клас QGraphicsScene надає поверхню для керування великою кількістю графічних 2D елементів. QGraphicsScene надає функціональність, яка дає змогу визначати положення цих елементів, а також дозволяє визначати видимість компонентів усередині довільної області сцени. Щоб додати будь-який елемент на сцену, використовується метод addItem.

Клас QGraphicsView є віджет для відображення вмісту QGraphicsScene. Він також дозволяє відобразити всю сцену цілком або тільки певну її частину. Щоб вказати сцену, яку відображатиме View, потрібно використовувати метод setScene. Щоб відобразити View, використовується метод show().

Клас QGraphicsRectItem дозволяє створити прямокутний елемент, який буде використовуватися як модель. За допомогою методу setRect (x, y, width, height) встановлюється місцезнаходження та розміри прямокутника:

- x – координата x прямокутника;
- y – координата y прямокутника;
- width – ширина прямокутника;
- height – Висота прямогульника.

Взаємодія трьох вище перерахованих класів можна побачити рис. 20. Нижче представлений код створює сцену, вікно виведення (View) та прямокутник розмірами 100 на 100. Результат роботи коду можна побачити на рис. 21.

```
QGraphicsScene * scene = new QGraphicsScene(); QGraphicsRectItem * rect = new
QGraphicsRectItem();
rect->setRect(0,0,100,100); scene->addItem(rect);
QGraphicsView * view = new QGraphicsView(scene);
view->show();
```

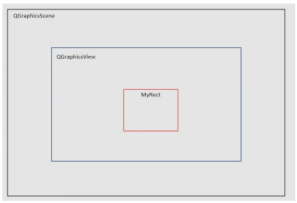

Рисунок 20 – Взаємодія Scene, View, RectItem

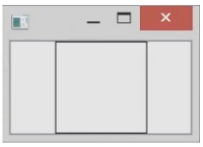

Рисунок 21 – Результат роботи коду

4.2.2 Створення моделі ворогів та кулі

Графічне уявлення у бібліотеках є декартову систему координат. Таким чином, щоб змінити положення елементів та їх геометрію потрібно скористатися двома величинами: x-координати та y-координати.

Як модель ворога і кулі виступатиме QGraphicsRectItem, який є прямокутним елементом.

Вороги повинні з'являтись у верхній частині екрана гри. Для визначення позиції моделі ворога (поява ліворуч, праворуч або серединою) використовується функція rand(), яка генерує розташування моделі ворога перед її появою. Якщо модель ворога пересуватиметься косою траєкторією, слід враховувати місце її появи (верхній лівий/правий кут).

За допомогою функції setPos (double x, double y) – встановлюється позиція елемента, де:

- x – координата x встановлюваного елемента;
- y – координата y встановлюваного елемента.

4.2.3 Обробка колізій

При взаємодії (перетин) моделі ворога і кулі, обидві моделі повинні бути

видалені зі сцени та пам'яті. Для цього використовується спеціальний масив, який включає моделі представлені на сцені, а також спеціально створену функцію, яка перевіряє перетин моделей певних фігур. При перетині моделей фігури – функція видаляє елементи зі сцени та звертається до певних (пересічених) моделей масиву та видаляє їх.

Таким чином, пересічені моделі втрачають видимість (видаляються зі сцени) і видаляються з пам'яті. Як приклад на рис. 22 представлений алгоритм логіки роботи при перетині моделі ворога та кулі.

Рисунок 22 – Алгоритм обробки при перетині ворога та кулі

При взаємодії (перетин) моделі ворога та гравця зі сцени повинна видалятися тільки модель ворога, але при цьому показник здоров'я гравця

повинен зменшитися.

4.2.4 Створення таймерів

Мета гри ─ знищити ворогів, що з'являються, за допомогою куль. Для генерації ворогів використовується таймер QTimer. Клас QTimer надає повторювані та одноразові таймери.

QTimer – це високорівневий програмний інтерфейс для таймерів. Для роботи з ним слідує:

– створити елемент QTimer;

– підключити його сигнал тайм-ауту до відповідного слота (викликати функцію переповнення таймера);

– викликати метод start().

Для цілей пересування моделей слід створити таймер, при переповненні якого положення кулі і ворога буде змінюватися (ворог пересуватиметься до нижньої частини екрану, а куля до верхньої). Нижче наведено приклад створення таймера для пересування ворога.

1. Створення елемента QTimer під назвою «timer»:

```
QTimer * timer = New QTimer();
```

2. Підключення таймера до слота (функції) «move»:

```
connect(timer,SIGNAL(timeout()),this,SLOT(move()));
```

3. Увімкнення таймера методом "start(msec)", де msec – кількість мілісекунд до переповнення таймера:

```
timer->start(msec);
```

4.3 Завантаження текстур, музики, моделей

На цій стадії всі звичайні об'єкти QGraphicsRectItem (прямокутники) замінюються на графічні моделі. У програмі ««Космічні війни»» потрібні такі

типи моделей:

– спрайти моделей;

– музика та звуки;

– відео.

Файлова структура додатка представлена рис. 23.

1) Sprites

– Enemy – стандартна модель ворога. Є 2D спрайтом космічного корабля з корпусом червоного кольору.

– Bullet – стандартна модель кулі, яку випускає гравець. Як спрайт кулі була обрана ракета.

– Player – стандартна модель гравця (користувача). Являє собою космічний корабель із корпусом зеленого кольору.

– BackGround – заднє тло гри. Як тло було обрано малюнок космосу.

Основні моделі гри наведено рис. 3.3.2.

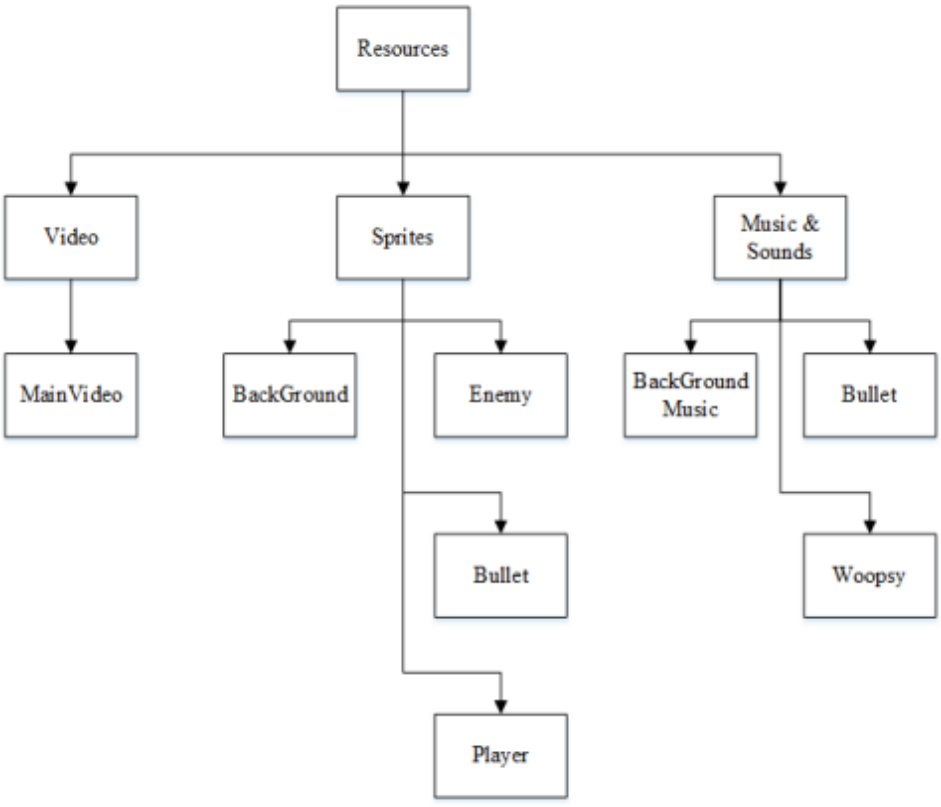

Рисунок 23 – Файлова структура програми «Космічні війни»

Рисунок 24 – Основні моделі програми «Космічні війни»

2) Music & Sounds

– Bullet – звук, який з'являється в результаті запуску кулі гравцем (у стандартній моделі – ракета).

– BackGroundMusic – музика, яка програється як фон.

– Woopsy – звук, що здається при включенні функції «Woopsy».

3) Video (MainVideo – заставка у головному вікні програми, являє собою анімацію корабля, що летить до червоного карлика).

4.4 Розробка логічної складової

4.4.1 Розробка рівнів складності

В ігровому додатку «Космічні війни» потрібно розробити різні рівні складності.

– стандартна складність (складність, приякої у користувача немає труднощів у процесі гри. Даний вид складності є окремим випадком заданої складності).

– задана складність (користувач може задати рівень складності показником від 1 до 10, при цьому 10 рівень складності є найважчим для проходження, а 1 найлегшим).

– адаптивна складність (цей рівень складності змінюватиметься в залежності від успіхів гравця в процесі гри. Таким чином складність

змінюватиме своє значення від найлегшого показника до найскладнішого і навпаки).

Як приклад рис. 25 представлений алгоритм роботи адаптивної складності.

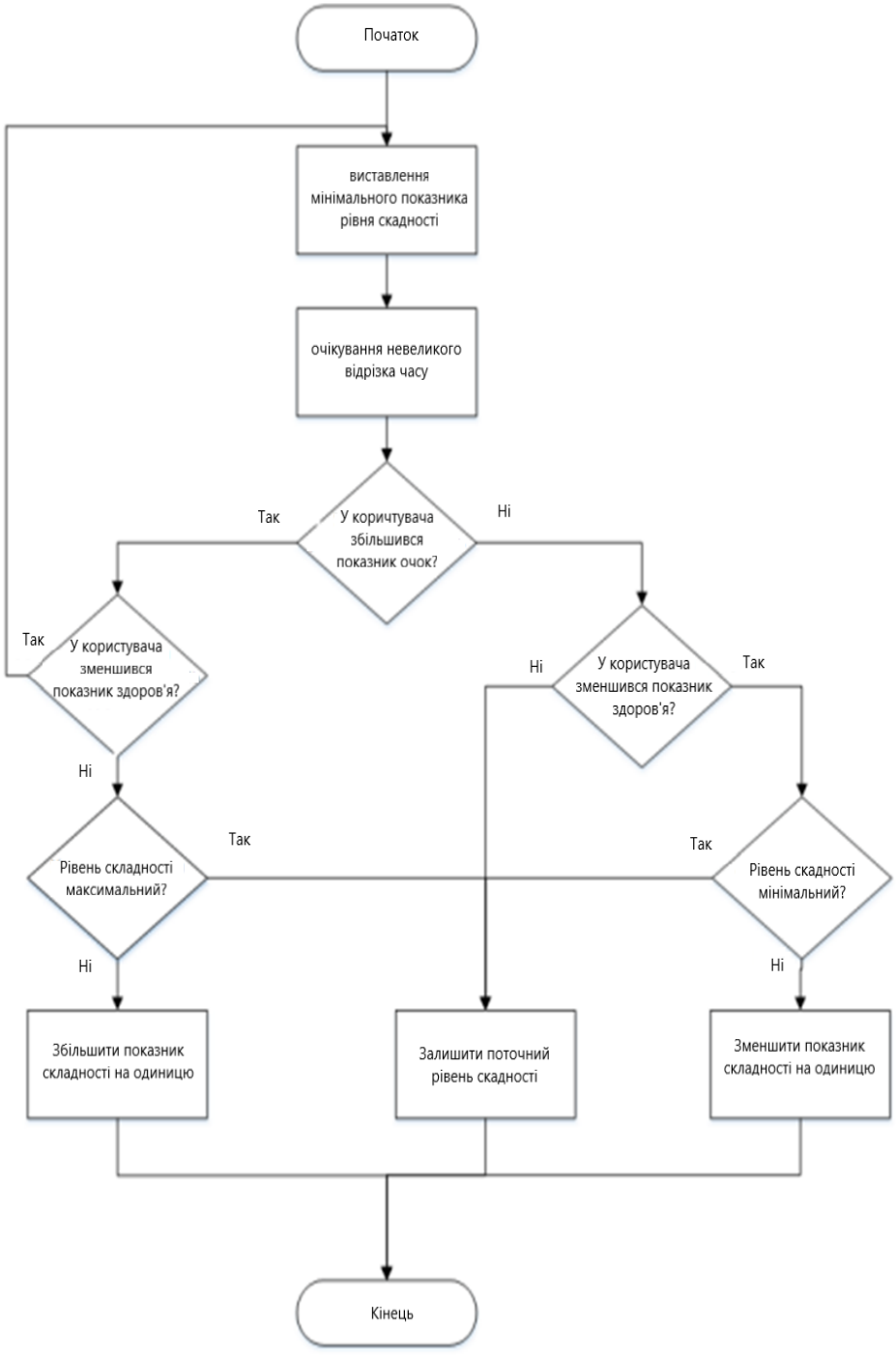

Рисунок 25 – Алгоритм роботи адаптивної складності

4.4.2 Розробка типів пересування

Можна виділити три види переміщення ворожих об'єктів (рис. 26):

– пересування прямою траєкторією;

– пересування косою траєкторією;

– пересування за гармонійним законом.

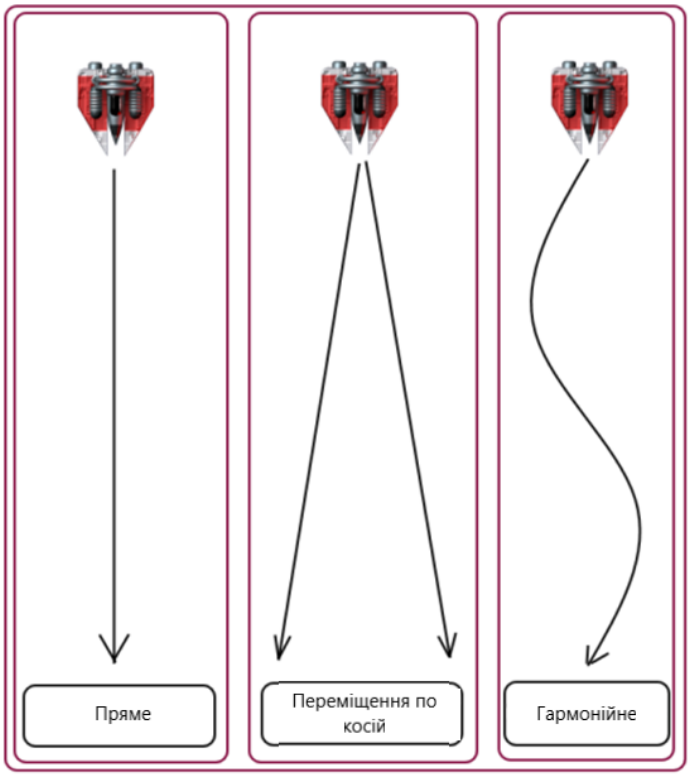

Рисунок 26 – Види пересування ворожих об'єктів

4.4.2.1 Пересування об'єктів прямою

Для пересування об'єкта прямою траєкторією слід зміщувати її по координаті Y через особливості системи координат графічного представлення (рис. 3.2.2). Таким чином, щоб перемістити ворожу модель з верхньої позиції екрана в нижню, скористаємося такою функцією:

```
SetPos (x(),y() + offset);
```

Де offset – величина усунення щодо координати Y

4.4.2.2 Пересування об'єктів косою траєкторією

Для пересування об'єкта косою траєкторією слід зміщувати її по координаті Y і X. При цьому значення зміщення Y повинно перевищувати значення зсуву X в кілька разів. Також слід враховувати місце появи ворожого об'єкта. Якщо об'єкт з'явився в лівій верхній частині екрана, то його кінцевий пункт призначення нижній правий кордон і навпаки. Таким чином, функція пересування набуде наступного вигляду:

```
SetPos (x() ± offsetX, y() + offsetY);
```

Де offsetX (Y) – зміщення координати X (Y).

4.4.2.3 Пересування об'єктів по гармонійному коливанню

Щоб об'єкт пересувався за гармонічним законом, скористаємося тригонометричною функцією – синусом (рис 27). З малюнка видно, що збільшити зсув щодо координати X – слід змінити значення параметра A (амплітуди коливання). Для збільшення зміщення щодо координат Y – потрібно змінити параметр частоти w (циклічна частота). Параметром початкової фази (f0) знехтуємо (встановимо значення 0).

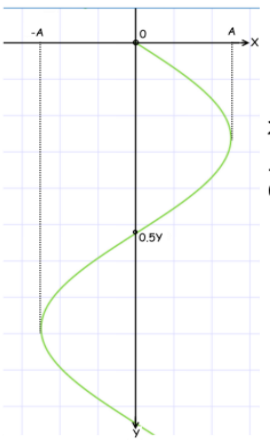

Рисунок 27 – Тригонометрична функція синуса

Тоді функція пересування набуде наступного вигляду:

```
SetPos (x() ± A*sin(y()*w), y() + offsetY);
```

де A – усунення щодо координати X;

w – параметр циклічної частоти;

offsetY – Зміщення по координаті Y.

В результаті процесу розробки були виконані такі завдання:

– виділено основні положення гри (тип гри, жанр тощо);

– вироблено основні цілі та завдання гри;

– продумані основні можливості та особливості гри;

– складеніUML-діаграмифункціональної складника програмної системи;

– вивчено технології роботи з графічними компонентами;

– для роботи зі звуковими компонентами системи були вивченівідповідні бібліотеки;

– продумано логічні компоненти системи;

– розроблено різні рівні складності ігрової програми;

– розроблені різні типи пересування, а також розрахованіформули кожного типу переміщення ворожих одиниць;

– були знайдені та використані безкоштовні моделі для програмної системи;

– розроблено інтерфейси для взаємодії з користувачем (завантаження текстур, вибір режиму складності/траєкторії);

На рис. 28 представлена коротка діаграма класів продукту «Космічні війни», на якій можна побачити наступне: є основні класи – ворога (Enemy), гравця (Player), кулі (Enemy), показника здоров'я (Health ») та очок («Score»). За допомогою інтерфейсу взаємодії з користувачем (GetFile) відбувається вибірка текстур деяких компонентів, а також, за бажанням користувача, звукової складової. Після чого всі отримані дані (якщо дані не отримані – використовуються стандартні компоненти) використовуються у класі «Game», який здійснює ініціалізацію всіх компонентів гри, запускає її, а також слідкує

за зовнішніми сигналами від користувача (натискання клавіш).

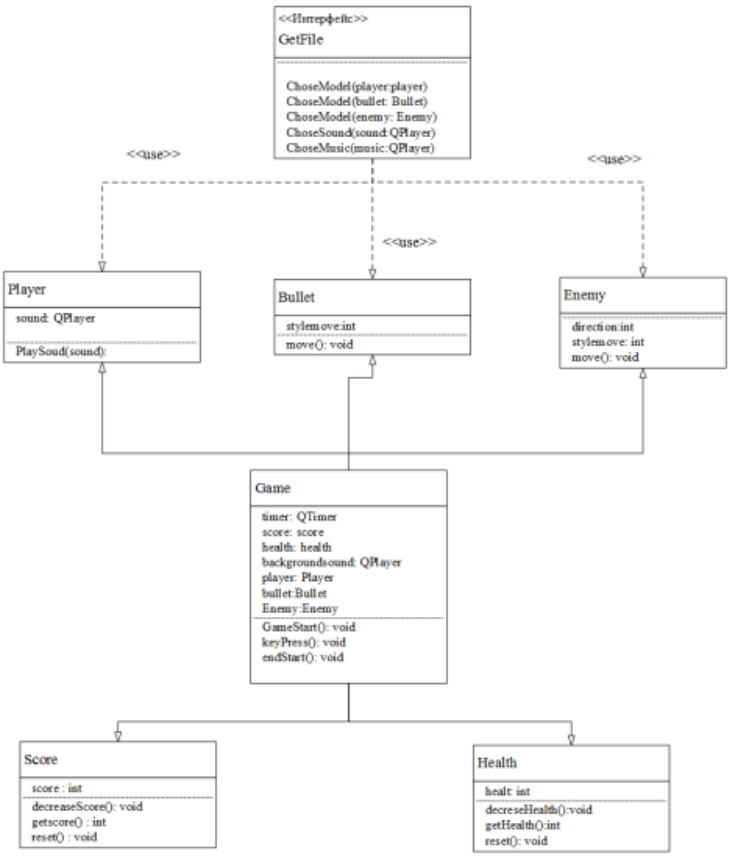

Рисунок 28 – Діаграма класів програмного ігрового продукту «Космічні війни»"

5 ОПИС ДОДАТКУ «КОСМІЧНІ ВІЙНИ»

5.1 Загальний характеристики

«Космічні війни» – це гра в якій гравцеві потрібно виживати, знищувати якомога більше ворогів і заробити якомога більше балів.

При запуску програми «Космічні війни» користувач бачить основне меню програми, представлене на рис. 29. Взаємодія з елементами меню здійснюється за допомогою комп'ютерної миші – наводимо курсор на потрібний елемент і натискаємо на ліву кнопку миші.

Основні пункти меню:

– грати;

– налаштування текстур;

– настроювання логіки;

– вихід.

При виборі користувачем кнопки Грати з'являється вікно, в якому здійснюється основний процес гри (рис. 30).

Рисунок 29 – Головне меню

Рисунок 30 – Основне вікно гри

Основні елементи керування:

– клавіша «Вліво» – пересування моделі гравця вліво;
– клавіша «Вправо» – пересування моделі гравця вправо;
– клавіша «Пробіл» – випуск кулі моделлю гравця.

У верхній лівій частині екрана представлено три показники:

– "Score" – рахунок гравця (показник очок);
– “BestScore” – найкращий показник очок;
– “Health” – показник здоров'я (на початку гри цей показник дорівнює трьом).

Після закінчення гри, тобто коли досягненно нульовий показника здоров'я (Health = 0) гравцеві виводиться вікно рахунку (рис 31), у якому гравець має вибрати наступні дії:

– «Так!» – почати гру знову, при цьому всі налаштування гри зберігаються;
– "Налаштування" – перейти у вікно налаштувань графічних компонентів;
– "Логіка" – перейти у вікно налаштувань логіки;
– "Вихід" – закрити гру.

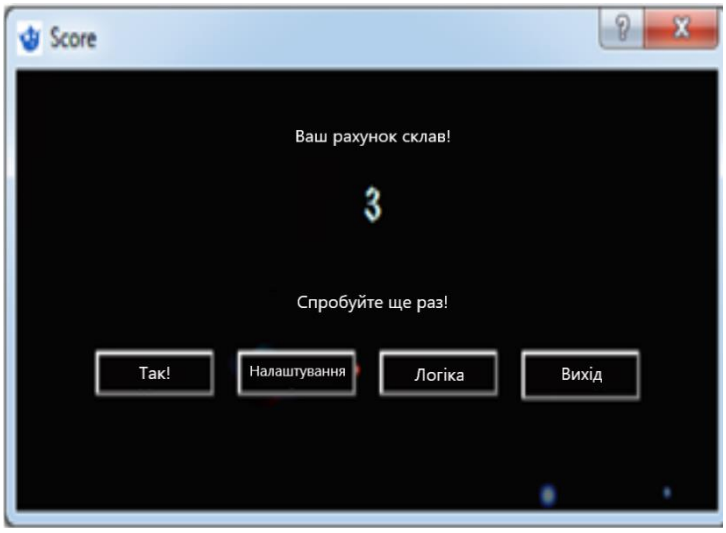

Рисунок 31 – Вікно рахунку

При виборі пункту «Налаштування» користувачеві відкривається вікно, представлене рис. 32. У цьому вікні гравець може налаштувати:

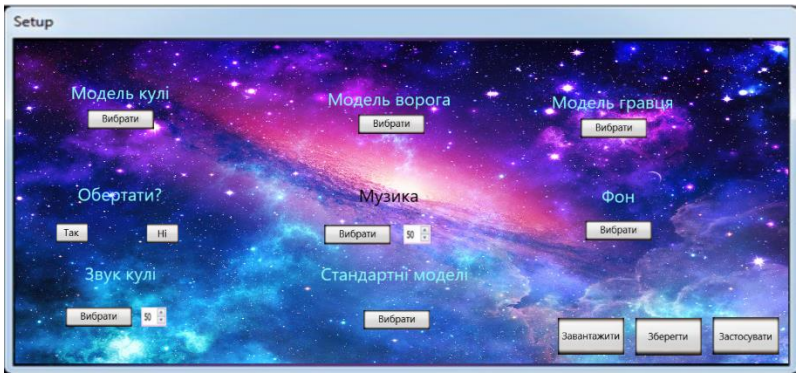

Рисунок 32 – Меню установок

– текстуру моделі гравця;
– текстуру моделі кулі;
– текстуру моделі ворога;
– фон гри;
– звук кулі (а також його гучність);
– музику гри (а також її гучність);

Також у гравця є можливість зберегти вибрані налаштування за допомогою кнопки «Зберегти» та в майбутньому завантажити їх за

допомогою кнопки "Завантажити".

По закінченні всіх налаштувань гравець повинен натиснути кнопку «Застосувати», розташовану в нижньому правому куті, щоб зміни зміни набрали чинності.

При виборі пункту «Логіка» користувачеві виводиться вікно, представлене рис. 33. У цьому вікні гравець може налаштувати наступні функції.

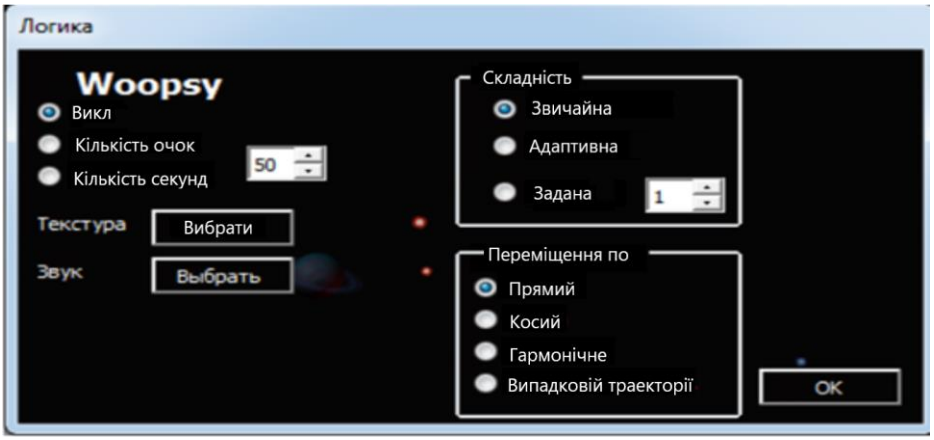

Рисунок 33 – Меню опцій логіки

5.1.1 Опис функції "Woopsy"

Ця функція виводить картинку в нижньому правому кутку та відтворює звуковий сигнал, які можна вказати через кнопки «Текстура-Вибрати» та "Звук-вибрати". Функція Woopsy спрацьовує за певною умовою, яку може задати гравець:

– якщо вибрати пункт «Кількість очок», то функція активується для досягнення кількох очок, вказаних поруч (праворуч) від даної установки.
– якщо вибрати пункт «Кількість секунд», то функція активується через вказану кількість секунд, після запуску гри.

Спочатку (у стандартних налаштуваннях) цю функцію вимкнено.

5.1.2 Характеристика складності

Гравець має можлтвість вибирати один із трьох рівнів складності:

– звичайна складність – оптимальний рівень складності, не змінюється під час гри.
– адаптивна складність – на початку гри рівень складності дорівнює стандартному, однак залежно від гри користувача, рівень складності може збільшуватися або зменшуватися.
– задана складність – користувач може вибрати рівень складності від 1 до 10, при цьому рівень складності не буде змінюватися протягом всієї гри.

5.2 Налаштування моделей

В якості прикладу налашьування програми виконуємо наступні дії:

– замінимо модель гравця;
– замінимо модель ворога;
– замінимо модель кулі;
– замінимо тло гри;
– збережемо налаштування;

Для цього необхідно запустити додаток «Космічні війни» та перейти в меню "Налаштування" (рис. 34).

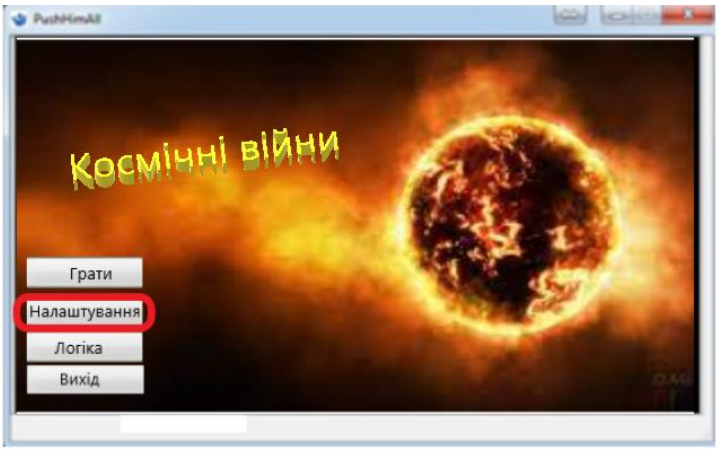

Рисунок 34 – Перехід у вікно налаштувань

Для заміни моделі кулі потрібно натиснути на кнопку "Вибрати" (рис. 35) і після цього перед користувачем відкривається нове вікно, в якому він має можливість вказати потрібну модель.

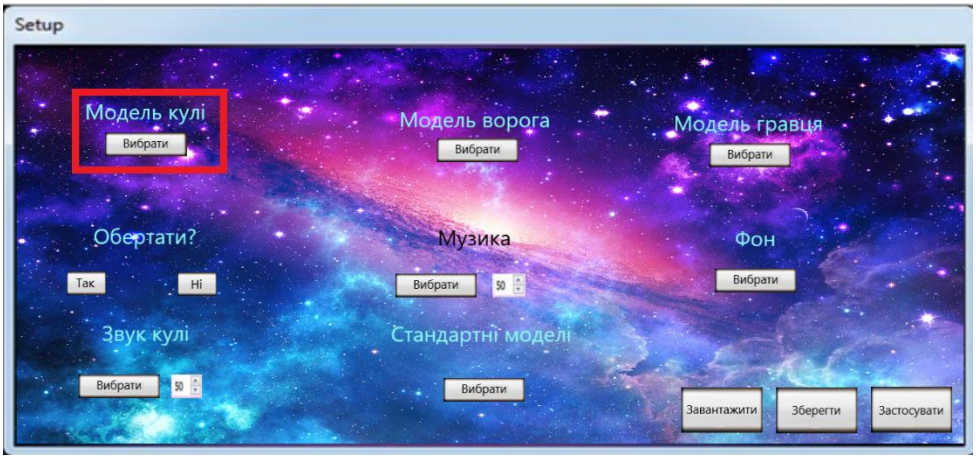

Рисунок 35 – Вікно налаштувань

Виконаємо подібні дії і для інших моделей.

– як модель ворога вкажемо файл "mask".

– як модель гравця вкажемо файл «Корабль».

– як фон вкажемо файл "kosmos".

допомогою кнопки "Зберегти" зберігаємо налаштуванняю.

Для того, щоб зміни набули чинності, потрібно натиснути кнопку "Застосувати".

Після цього гравець побачить головне вікно програми. Тепер можна запустити гру за допомогою кнопки Грати і подивитися на результат.

Як видно, всі налаштування успішно застосовуються (рис. 36).

Щоб завантажити раніше збережені установки, потрібно перейти в меню «Налаштування» та натиснути на кнопку «Завантажити».

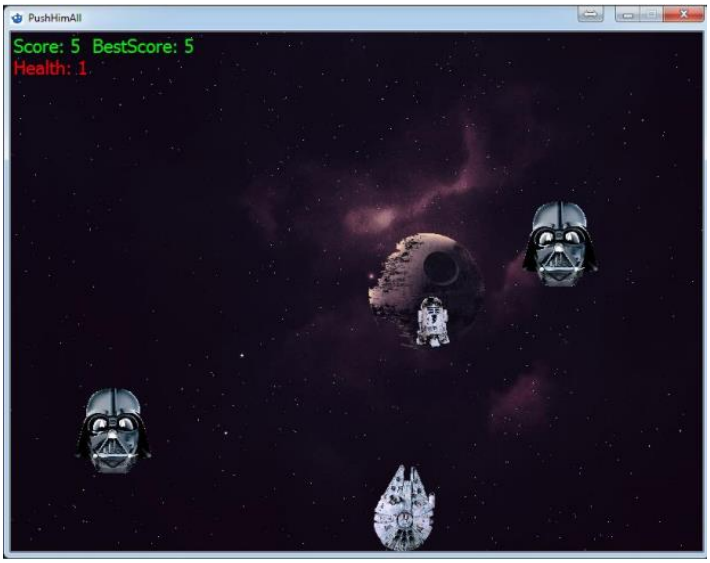

Рисунок 36 – Результат застосованих налаштувань

Також користувач може вибрати опцію «Обертати». Ця функція дозволяє обертати стандартну або задану користувачем (гравцем) модель кулі. Щоб модель кулі оберталася, потрібно вибрати відповідний пункт у меню налаштувань (рис. 35). "Так" – активує функцію обертання. "Ні" – відключає цю функцію (модель кулі не обертається).

Користувач має можливість вказувати музику в грі за допомогою пункту «Музика» та вказувати її гучність, також звук (і його гучність) кулі за допомогою пункту «Звук кулі».

Для гри зі стандартними моделями потрібно натиснути кнопку

"Вибрати" у пункті "Стандартні моделі". Після цього всі налаштування користувача скинуться, а гра набуде стандартних значень.

5.3 Налаштування логіки

Ігровий додаток «Космічні війни» також дозволяє змінювати поведінку ворогів, зокрема:

– змінювати складність гри;

– змінювати траєкторію пересування ворогів;

Для того, щоб задати дані параметри, потрібно перейти з головного меню ігрової програми в меню «Логіка» або після завершення гри,

натиснувши кнопку «Логіка» на вікні рахунку.

Користувач (гравець) Можна задати траєкторію пересування ворожих одиниць. Спочатку всі вороги рухаються прямою траєкторією, однак, користувач може змінити її на наступні варіанти:

- пересування прямою
- Ворожа одиниця створюється у верхній частині ігрового вікна і починає своє пересування в нижню частину екрана прямою траєкторією.
- пересування по косій
- Залежно від положення створення ворожої одиниці вона почне пересуватися в нижній лівий або правий кут ігрового вікна.
- пересування гармонікою

У цьому варіанті ворожа одиниця змінюватиме своє становище протягом усього шляху за гармонічним законом. У цьому варіанті потрапити за метою стає значно складніше, цим тренуються прогностичні можливості гравця.

Усі ворожі одиниці матимуть власний закон пересування, які перераховані вище (за прямою, косою, гармонікою). Цей режим є найважчим, оскільки потрібно прогнозувати траєкторію кожної ворожої одиниці одноразово.

ВИСНОВКИ

Частка користувачів ПК та інтернет аудиторії зростає щодня. В основному користувачі використовують ПК як розважальну платформу. Отже, ігрова аудиторія продовжує зростати, цим мотивуючи розробників створювати нові ігрові проекти.

У рамках кваліфікаційної роботи бакалавра було вивчено та опрацьовано технологію розробки ігор шляхом створення ігрового додатку «Космічні війни» у жанрі Аркада з можливістю змінювати моделі гри, а також логіку поведінки ворогів. Додаток був розроблений програмною мовою C++.

Для досягнення цієї мети було вирішено такі завдання:

- здійснено постановку ігрового завдання;
- проведено аналіз аналогових програмних засобів розробки комп'ютерної гри;
- здійснено пошук та підбір безкоштовних моделей;
- повторені та поглиблені знання у мові програмування C++;
- вивчено роботу з деякими бібліотеками Qt;
- спроектований додаток;

ПЕРЕЛІК ПОСИЛАНЬ

1. Професії майбутнього. [Електронний ресурс]. URL: http://www.proforientator.ru/publications/articles/detail.php?ID=5454 (дата звернення 12.05.2022).

2. Аналіз ринку ігор у світі 2014-2016 років [Електронний ресурс]. URL: http://json.tv/ict_video_watch/analiz-rynka-igr-v-rossii-i- mire-2014-2016-gg-tekuschaya-situatsiya-prognozy-igroki-proekty-i-tendentsii. (дата звернення 02.03.2022).

3. Ринок ігор у світі з 2010 по 2016 рік [Електронний ресурс]. URL: http://json.tv/ict_telecom_analytics_view/rynok-igr-v-rossii-i-mire-2010-2016-gg-20141121113425. (дата звернення 05.03.2022).

4. Worldwide Digital Games Market [Електронний ресурс]. URL: https://twitter.com/_SuperData/status/675337478514024448. (дата звернення 06.03.2022).

5. Top Operating System Software [Електронний ресурс]. URL: http://www.jegsworks.com/lessons/ComputerBasics/lesson8/lesson8-3.html. (дата звернення 05.03.2017).

6. Unity [Електронний ресурс]. URL: https://unity3d.com/ua. (дата звернення 15.03.2017).

7. Популярні ігри [Електронний ресурс]. URL: https://habrahabr.ru/company/mailru/blog/181974/. (дата звернення 12.03.2022).

8. CRYENGINE – The complete solution for next generation game evelopment [Електронний ресурс]. URL: https://www.cryengine.com. (дата звернення 20.03.2022).

9. What is Unreal Engine 4? [Електронний ресурс]. URL: https://www.unrealengine.com/. (дата звернення 18.03.2017).

10. Game programming [Електронний ресурс]. URL: goo.gl/rxJ4xzc. (дата звернення 20.03.2022).

11. Qt Graphics Framework [Електронний ресурс]. URL:http://doc.qt.io/qt-5/graphicsview.html. (дата звернення 15.03.2022).

12. What is SFML? [Електронний ресурс]. URL: https://www.sfml-dev.org/grl-whatis. (дата звернення 16.03.2022).

13. Space Invaders. [Електронний ресурс]. URL: https://ua.wikipedia.org/wiki/SpaceInvaders. (дата звернення 13.05.2022).

14. Радіант. [Електронний ресурс]. URL: http://www.hexag.net/radiant/. (дата звернення 13.05.2022).

15. Luxor. [Електронний ресурс]. URL: https://ru.wikipedia.org/wiki/Luxor_Game/. (дата звернення 14.05.2022).

16. Beyond Space. [Електронний ресурс]. URL: http://store.steampowered.com/app/297111/. (дата звернення 15.05.2022).

17. Чужий у космосі. [Електронний ресурс]. URL: http://onlineguru.ru/15104/view.html/. (дата звернення 13.04.2022).

18. Діаграма варіантів використання як концептуальне уявлення бізнес-системи у процесі її розробки. [Електронний ресурс]. URL: http://www.intuit.ru/studies/courses/32/32/lecture/1004 (Дата звернення 10.02.2022

ДОДАТОК

Уривок листингу програми

"MainWindow.h" - заголовний файл головного меню

```
#ifndef
MANWINDOWS_H
#define
MANWINDOWS_H
#include
<QManWindows>


namespace Uii

{

class ManWindows;

}

class ManWindows : public QManWindows{

    Q_OBJECT

public:

    ~ManWindows();

private slotss:

    void on_ExitButton_clicked();//кнопка "Вихід"

    void restartVideo();    //Зациклювання анімованого
    меню
    void on_GameButton_clicked();// Кнопка "Грати" -
    запуск
    ігор
    voidon_SettingButton_clicked();
                //кнопка«Налаштування»                -
відкриття налаштувань текстури та музики.

    void on_BtnForLogic_clicked();//кнопка "Логіка" -
відкриття налаштувань логіки


};
```

"MainWindow.c" - вихідний код головного меню

```
//Підключення бібліотек
#include "MainWindow.h"
#include "ui_MainWindow.h"
#include "Game.h"
#include "Settings.h"
#include <QMovie>
#include <QLabel>
#include <QGraphicsVideoItem>
#include <QGraphicsView>
#include <QGraphicsScene>
#include <QMediaPlayer>
#include <QPushButton>
#include <QFileDialog>
#include <QMessageBox>
#include <QUrl>
#include <QFile>

//Оголошення змінних
extern Game * game;
QMediaPlayer *player=new QMediaPlayer(); int clickplay = 0;

//Ініціалізація меню
//Включення анімованого меню

MainWindow::MainWindow(QWidget *parent) : QMainWindow(parent),

            uii(new Uii::ManWindows)

            uii->setupUii(this);
            this->setFixedSize(this->size().width(),this-
               >size().height());
               setAttribute(Qt::WA_DeleteOnClose);
               QGraphicsVideoItem *item=new
               QGraphicsVideoItem; item->setSize(ui-
               >playerScreen->size());
               player->setVideoOutput(item);
               QGraphicsScene*scene=newQGraphicsScene(0,0,ui-

   >playerScreen->size().width(),ui->playerScreen-
               >size().height()); ui->playerScreen-
               >setScene(scene);
               ui->playerScreen->scene()->addItem(item);
               player-
               >setMedia(QUrl::fromLocalFile("E:\Diplom.mp4"))
               ; player->setVolume(100);
               player->play();

        connect(player,SIGNAL(stateChanged(QMediaPlayer::State)),thi
```

```
          s,SLOT(re startVideo()));
          }
          MainWindow::~MainWindow()
          {
              delete ui;
          }
          //Кнопка «Вихід» - закриття програми
          void MainWindow::on_pushButton_2_clicked()
          {
              exit(0);
          }

          // Анімоване меню
          void MainWindow::on_pushButton_3_clicked()
{
              clickplay++; player->stop(); QString filename;
             player-
              >setMedia(QUrl::fromLocalFile(filename)
              ); player->play();
              clickplay--;
          }
          void MainWindow::restartVideo()
          {
              if(!clickplay) player->play();
          }
          //Кнопка "Грати" - запуск гри
          void MainWindow::on_pushButton_clicked()
          {
              clickplay++; player->stop(); delete player; game = new
              Game(); game-> show ();
              this->close(); clickplay--;
          }
          //Кнопка "Налаштування" - відкриття вікна налаштування текстур,
          музики.
          void MainWindow::on_SettingButton_4_clicked()
          {
              Settings *setup = new Settings();
              setup->setAttribute(Qt::WA_DeleteOnClose);
              setup->setFixedSize(setup->size().width(),setup-
   >size().height());
setup->show();
          }
          //Кнопка "Логіка" - відкриття вікна логіки програми
          void MainWindow::on_LogicButton_clicked()
          {
              Settings *setup = new Settings();
              setup->setAttribute(Qt::WA_DeleteOnClose);
              setup->setFixedSize(setup->size().width(),setup-
   >size().height());
              setup->show();
```

```
}
```