Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

Кафедра програмного забезпечення

**Пояснювальна записка**

до магістерської кваліфікаційної роботи

на тему: «»

Виконала: студентка ІІ курсу,

групи 1ПЗ-16м

спеціальності 8.05010301 –

Програмне забезпечення систем

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Рецензент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Вінниця - 2018 року

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

Кафедра програмного забезпечення

Освітньо-кваліфікаційний рівень – магістр

Спеціальність 8.05010301 – Програмне забезпечення систем

**ЗАТВЕРДЖУЮ**

**Завідувач кафедри ПЗ**

**Пєтух А.М.**

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_року

**З А В Д А Н Н Я**

**НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ**

Польовій Марії Валеріївні

1. Тема роботи: **Розробка web-додатку для онлайн реєстрації пацієнтів на прийом до лікаря.**

керівник роботи: Войтко Вікторія Володимирівна, к.т.н., доцент кафедри ПЗ,

затверджені наказом вищого навчального закладу від “\_\_”\_\_\_\_\_20\_\_року №\_\_\_

2. Строк подання студентом роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Вихідні дані до роботи: операційна система – Windows XP/7/8; середовище програмування Visual Studio 2013; мова програмування – C#, SQL, JavaScript;

технологія –.NET

4. Зміст розрахунково-пояснювальної записки: обґрунтування доцільності розробки автоматизованої системи; варіантний аналіз та обґрунтування вибору засобів реалізації автоматизованої системи; розробка моделей та структури автоматизованої системи; розробка програмного забезпечення автоматизованої системи; тестування роботи автоматизованої системи; економічна частина.

5. Перелік графічного матеріалу: мета роботи, об’єкт та предмет дослідження; основні задачі дослідження; наукова новизна, практичне значення; high-level архітектура web-додатку; модулі автоматизованої системи; приклад вигляду розробленої програми; висновки, публікації, апробації.

6. Консультанти розділів роботи

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посада  консультанта | Підпис, дата | |
| завдання видав | завдання  прийняв |
| 1-5 | Войтко В.В., к.т.н., доцент кафедри ПЗ |  |  |
| 6 | Бальзан М.В., к.е.н., доц. каф. ЕПВМ |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

7. Дата видачі завдання\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів магістерської кваліфікаційної  роботи | Строк виконання етапів роботи | Примітка |
| 1 | Технічне обґрунтування доцільності розробки web-додатку онлайн реєстрації | 20.09–25.09 |  |
| 2 | Варіантний аналіз та обґрунтування вибору засобів реалізації web-додатку | 26.09–30.09 |  |
| 3 | Розробка моделей системи | 01.10–10.10 |  |
| 4 | Розробка програмного забезпечення системи | 11.10–27.10 |  |
| 5 | Тестування програмного забезпечення системи | 28.10–04.11 |  |
| 6 | Економічна частина | 05.11–10.11 |  |

**Студент \_\_\_\_\_\_\_\_\_\_\_\_ Польова М. В.**

( підпис ) (прізвище та ініціали)

**Керівник магістерської кваліфікаційної роботи \_\_\_\_\_\_\_\_\_\_\_\_\_ Войтко В.В.\_\_**

( підпис ) (прізвище та ініціали)

АНОТАЦІЯ

ABSTRACT

ЗМІСТ

[ВСТУП 9](#_Toc502066891)

[1 АНАЛІЗ СУЧАСНОГО СТАНУ ПИТАННЯ 10](#_Toc502066892)

[1.1 Актуальність розробки ігрових додатків 10](#_Toc502066893)

[1.2 Аналіз жанрів ігрових додатків та обґрунтування вибору 10](#_Toc502066894)

[1.3 Порівняльний аналіз існуючих аналогів 18](#_Toc502066895)

[1.4 Аналіз методів розв’язання поставленої задачі 21](#_Toc502066896)

[1.5 Постановка задач дослідження 21](#_Toc502066897)

[1.6 Висновки 21](#_Toc502066898)

[2 РЗРОБКА МОДЕЛЕЙ ІГРОВОГО ДОДАТКУ 22](#_Toc502066899)

[2.1 Розробка моделей та структур даних 22](#_Toc502066900)

[2.2 Розробка нечітких правил 22](#_Toc502066901)

[2.3 Розробка структури інтерфейсу 27](#_Toc502066902)

[2.4 Висновки 27](#_Toc502066903)

[3 ВАРІАНТНИЙ АНАЛІЗ ТА ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ ІГРОВОГО ДОДАТКУ 28](#_Toc502066904)

[3.1 Аналіз особливостей розробки ігрового додатку 28](#_Toc502066905)

[3.2 Варіантний аналіз вибору мови розробки ігрового додатку 28](#_Toc502066906)

[3.3 Варіантний аналіз вибору середовища розробки ігрового додатку 28](#_Toc502066907)

[3.4 Варіантний аналіз вибору ігрового рушія 28](#_Toc502066908)

[3.5 Варіантний аналіз вибору бібліотеки для роботи з нечіткою логікою 28](#_Toc502066909)

[3.6 Висновки 28](#_Toc502066910)

[4 РОЗРОБКА ІГРОВГО ДОДАТКУ 29](#_Toc502066911)

[4.1 Аналіз методів тестування 29](#_Toc502066912)

[4 ТЕСТУВАННЯ SCROLLER-ГРИ 30](#_Toc502066913)

[4.1 Аналіз методів тестування 30](#_Toc502066914)

[4.2 Інструкція тестування 31](#_Toc502066915)

[4.3 Висновки 31](#_Toc502066916)

[ВИСНОВКИ 32](#_Toc502066917)

[ПЕРЕЛІК ПОСИЛАНЬ 33](#_Toc502066918)

[ДОДАТОК А Технічне завдання 37](#_Toc502066919)

[ДОДАТОК Б Лістинг вихідного коду ігрової системи 40](#_Toc502066920)

[ДОДАТОК В Ілюстративний матеріал 41](#_Toc502066921)

# ВСТУП

До 5 ст

# 1 АНАЛІЗ СУЧАСНОГО СТАНУ ПИТАННЯ

## 1.1 Актуальність розробки ігрових додатків

Відеогра – електронна гра, в яку можна відтворювати на комп'ютерному пристрої, наприклад: персональний комп'ютер, ігрова консоль або мобільний телефон. Залежно від платформи, відеоігри можуть бути розподілені на комп'ютерні ігри та консольні ігри. Проте останнім часом поява соціальних мереж, смартфонів і планшетів представила нові категорії ігор, такі як мобільні та соціальні ігри. Відеоігри пройшли довгий шлях з часу появи перших ігор у 1970-х. Сьогоднішні відеоігри пропонують фото-реалістичну графіку та імітують дійсність до ступеня, що вражає у багатьох випадках.

Ринок комп’ютерних ігор – це мільярди доларів та багато років існування. У 2016 році сегмент відеоігор в США оцінювався в 17,68 млрд доларів США. В цьому році споживачі ігор витрачають приблизно вдвічі більше на вміст ігор, аксесуари, обладнання. Широкої популярності ігрові додатки набрали серед старшого покоління людей [https://www.statista.com/topics/868/video-games/].

Популярності набирає постачання контенту через інтернет, що надає більше можливостей, як і для розробників, так і для клієнтів. Популярними платформами розповсюдження ігрового контенту через інтернет є Steam, Origin, gog.com, humble bundle, uplay. Зберігання та розповсюдження програмних продуктів на електронних носіях відходить у минуле. Сьогодні достатньо лише мати ліцензійний ключ, а доступ до контенту гри можна отримати з багатьох джерел за допомогою інтернету. Статистика наведена на рисунку 1.

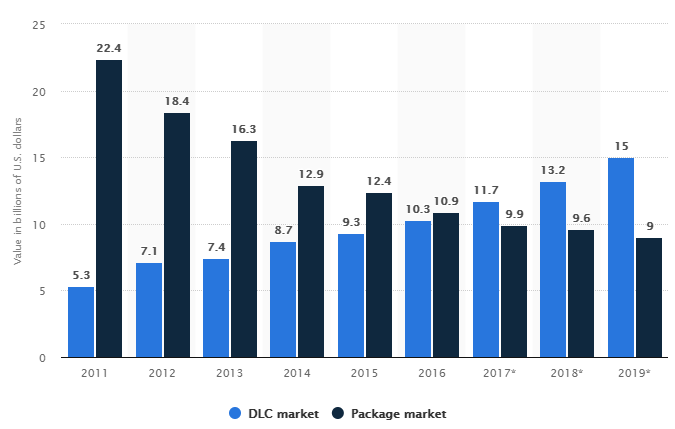


Рисунок 1 – Статистика розповсюдження ігрових додатків через інтернет

Згідно опитуванню споживачів щодо того, чи надають вони перевагу тій чи іншій платформі, персональні комп’ютери займають друге місце, поступившись першим місцем мобільній (рисунок 2).

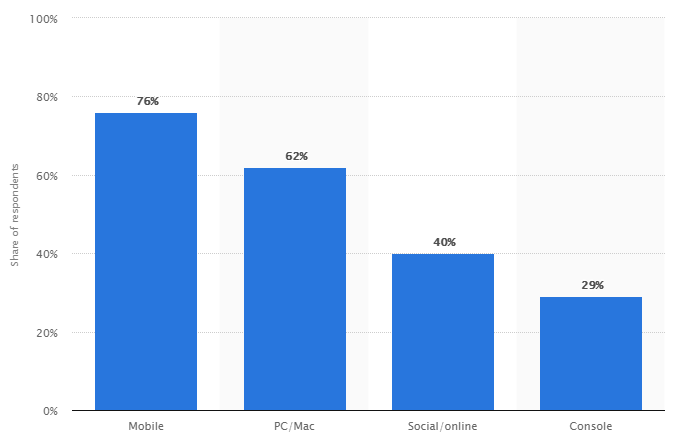


Рисунок 2 – Рейтинг ігрових платформ

Серед багатьох процвітаючих представників індустрії відеоігор є три основні розробники ігрових пристроїв, що розвивалися десятиліттями і залишаються на верхівці рейтингу з 2015 року. До них належать: Sony, Microsoft і Nintendo. PlayStation 4 від Sony є бестселером серед нинішніх поколінь консолей. До кінця 2015 року Sony продала 16,75 млн. одиниць популярної консолі.

Компанія Microsoft за поточний рік через онлайн-транзакції отримала 21,2 млрд доларів США (рисунок 3).



Рисунок 3 – Статистика річного доходу компанії Microsoft через онлайн-транзакції

Отже, ринок відеоігор активно розвивається, охоплює нові платформи та пристрої.

## 1.2 Аналіз жанрів ігрових додатків та обґрунтування вибору

Більшість відеоігор належать до певної категорії. Ігровий продукт може належати до декількох жанрів за рахунок їх комбінування. Частина виходить за рамки будь-якого жанру. Якщо новий продукт стає достатньо популярним, розробляються нові аналоги, то з'являється новий жанр.

Розглянемо поширені на сьогодні жанри відеоігор.

Шутер є одним з перших жанрів відеоігор. Першою грою в жанрі вважається «Spacewar» (рисунок 1).

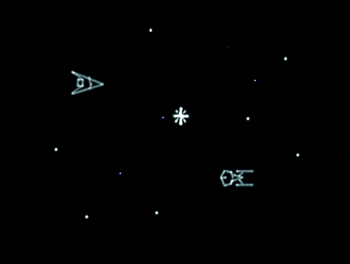


Рисунок 1 – Відеогра «Spacewar»

Головною ціллю шутерів є знищення ворогів за допомогою різноманітної зброї. З розвитоком технологій та можливістю відображати 3-х вимірні об’єкти на екрані популярними стали FPS (First-person shooter – шутер від першої особи). Одним із засновників жанру є відеогра «Doom» 1993 року, але вона не є 3-х вимірною грою.

В пригодницьких іграх гравець виступає в ролі протагоніста, що переживаю історію. Гра наповнена різного роду загадками та завданнями. Яскравим прикладом є «Indiana Jones and the Emperor's Tomb» (рисунок 2)



Рисунок 2 – Відеогра «Indiana Jones and the Emperor's Tomb»

Жанр платформер започатковано грою «Donkey Kong» (рисунок 3).

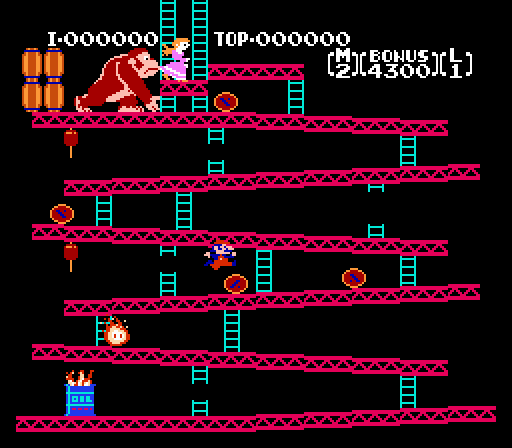


Рисунок 3 – Відеогра «Donkey Kong»

Головною ідеєю жанру є платформи з перешкодами, що повинен подолати гравець. В основу ігрового процесу покладено переміщення персонажа.

Розвиваючись від настільної гри, як «Dungeons and Dragons» 1974 року, RPG (Role-Playing Game – рольова гра) – це особливий тип пригодницької гри, зазвичай що складається з трьох основних елементів:

1. конкретний квест (завдання);
2. процес покращення персонажа за допомогою досвіду;
3. керування предметами.

Жанр головоломки націлений на логіку, здатність гравця розпізнавати патерни та доповнювати слова. Головоломки поширені серед всіх вікових категорій і різні за рівнем складності. До жанру належить відома у світі гра «Tetris» (рисунок 4).



Рисунок 4 – Відеогра «Tetris»

Відеоігри симулятори призначені для відтворення певного досвіду, наприклад: керування автомобілем, потягом, літаком, спортивною командою, тощо. Популярності набув різновид CMS (Construction and management simulation – симуляція будівництва та управління), де гравець повинен будувати та керувати певною інфраструктурою. Відомою є серія ігор «SimCity» та «Transport Tycoon» (рисунок 5).

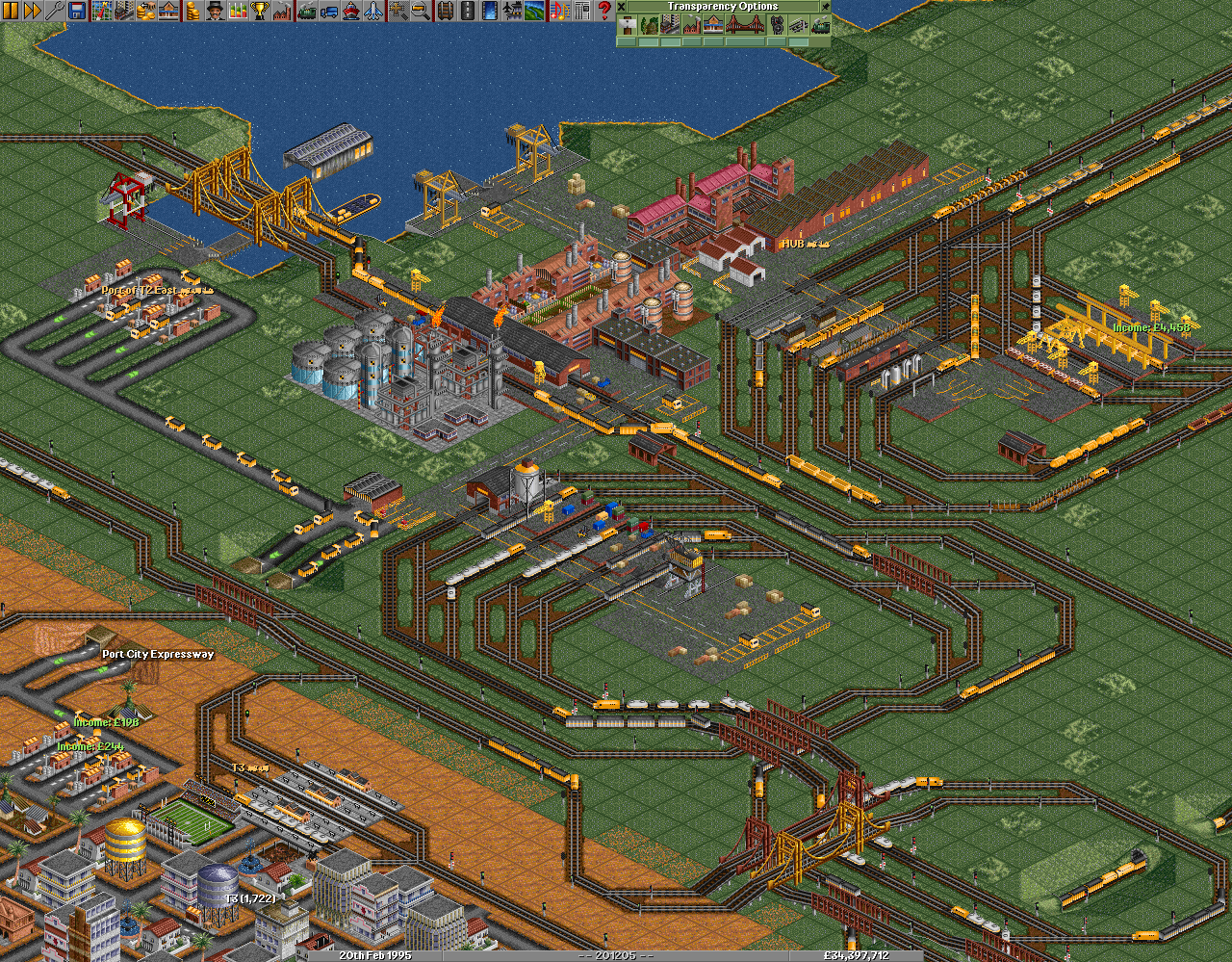


Рисунок 5 – Відеогра «Transport tycoon»

Першою спортивною відеогрою був Tennis for Two, де двоє гравців грали в теніс на осцилографі. На сьогодні існує безліч спортивних відеоігор, де гравець може формувати команду та безпосередньо брати участь у самій грі. Популярною серією спортивних ігор є «FIFA» – гра, де гравець у ролі тренера та менеджера команди, а також, безпосередньо керує командою під час матчів (рисунок 6).



Рисунок 6 – Відеогра «FIFA 2018»

Стратегії класифікуються на піджанри:

* RTS (Real-time strategy – стратегія в реальному часі);
* RTT (Real-time tactics – тактика в реальному часу);
* TBS (Turn-based strategy – покрокова стратегія);
* TBT (Turn-based tactics – покрокова тактика);
* MOBA (Multiplayer online battle arena – багатокористувацька онлайн-арена);
* TD (Tower defense – башта оборони).

Основною механікою стратегій в реальному часі є побудова бази та накопичення військової потужності для подолання ворога, увага звертається на безпосереднє керування кожним воїном.

Значний вклад у розвиток сучасних RTS зробила компанія Blizzard Entertainment. Популярними є серія фентезійних відеоігор «WarCraft» та серія науково-фантастичних ігор «StarCraft». «StarCraft» перша відеогра, що набрала широкої популярності в кіберспорті. Перший сезон було проведено в 1999 році, в Північній Кореї. На рисунку 7 наведено ігровий процес «StarCraft».



Рисунок 7 – Відеогра «StarCraft»

До покрокових стратегій відносяться «Heroes of Might and Magic», «Disciples», «M.A.X.»

Тактичні відеоігри зосереджуються увагу гравця на кожному воїні окремо, гравець не займається накопичення ресурсів на розбудовую бази. Прикладами покрокових тактичних ігор є серія ігор «X-Com», «UFO», «Jagged Alliance». Представником тактичних ігор в реальному часу є популярна серія ігор «Total War».

В іграх MOBA існує дві сторони, що з’єднані між собою декількома дорогами, кожен гравець керує одним персонажем. Ціль гри – знищити ворожий табір. В ході бою гравцям допомагають воїни, якими керує штучний інтелект. Відомою на весь світ є відеогра «Dota 2», що взяла свій початок, як звичайна мапа в грі «WarCraft 3». На сьогодні по «Dota 2» проходять масштабні змагання по всьому світу.

Однією з перших ігор в жанрі Tower Defense є «Rampart» (рисунок 8)



Рисунок 8 – Відеогра «Rampart»

В «Rampart» гравець будує стіни, і кожен замок, що повністю оточений стіною, а також територія, належать гравцю. На етапі будівництва стіни задаються випадково. На території замку гравець будує башти. Залежно від того, яка площа території належить гравцю, він отримує певну кількість залікових балів, від яких залежить кількість башт, які може побудувати власник замку. Після кожної хвилі гравець повинен відновити стіни, щоб замки з баштами належали йому. Гравець програє у випадку, якщо йому не належить жоден із замків.

Жанр ігор Tower Defense набрав широкої популярності серед комп’ютерних ігор на всіх платформах. Існує безліч Tower Defense сценаріїв в таких іграх, як: «Stacraft» та «Warcraft 3».

ДОПИСАТИ про приклади.

Отже, стратегічні відеоігри мають багато можливостей для використання нечіткої логіки, оскільки в них безліч параметрів, формули для розрахунків яких можна замінити розрахунками з використанням нечіткої логіки. Піджанром стратегії обрано Tower Defense для спрощення розрахунків, оскільки гравець не має можливості безпосередньо керувати воїнами.

1. https://www.thocp.net/software/games/reference/genres.htm

## 1.3 Порівняльний аналіз існуючих аналогів

Популярними на сьогодні ігри в жанрі Tower Defense є:

* Dungeon Defenders.
* Orcs Must Die! Unchained.
* Kingdom Rush.

Dungeon Defenders – це поєднання класичної Tower Defense гри з елементами RPG з можливістю гри по мережі до 4 гравців. Гравець вибирає одного з чотирьох героїв: маг, воїн, стрілець, монах. Кожен клас має унікальні спеціальні здібності, які поєднуються з конкретними оборонними спорудами. Наприклад, монах може створювати аури, які погіршують показники ворогів: сповільнення, зменшення пошкодження, спрямовування ворогів одне на одного. Також монах може лікувати союзників. Воїн, з іншого боку, будує захисті споруд, щоб зупинити ворогів і дати іншим героям завдати більшого пошкодження. Ціль гри – відбити всі хвилі і захистити об’єкт [1].

На рисунку 2 зображено ігровий процес Dungeon Defenders.



Рисунок 2 – Ігровий процес Dungeon Defenders

Orcs Must Die! Unchained – це Tower Defense з елементами MOBA. У грі бере участь дві команди, кожна з яких повинна оборонятися від орків та героїв ворожої команди. Кожен гравець вибирає героя певного класу. Кожен герой може будувати перешкоди. Мапа складається з двох віддзеркалених частин, тому жодна з команд не має переваг над іншою. В ході гри орки команди можуть вдосконалюватися, як і самі герої. Ціль гри – знищити ворожий портал, з якого виходять ворожі орки [3].

На рисунку 3 зображено ігровий процес Orcs Must Die! Unchained.



Рисунок 3 – Ігровий процес Orcs Must Die! Unchained

Kingdom Rush – це класична Tower Defense гра. Однією з особливостей гри є башти-казарми. З кожної казарми на поле виходять декілька воїнів, які зупиняють ворожі війська, що дозволяє іншим баштам завдати більшого пошкодження (рисунок 4).



Рисунок 4 – Ігровий процес Kingdom Rush

Кожен клас башти має своє дерево розвитку, кожна башта має певний набір характеристик, що можна вдосконалювати декілька разів. Також, в арсенальні гравця є метеоритний дощ та виклик підкріплення. Між рівнями гравець має можливість вдосконалювати параметри башт та метеоритного дощу, підкріплення. Перед кожним рівнем гравець вибирає героя, яким він зможе керувати. Кожен рівень має різного роду досягнення, деякі з них неможливо досягнути за одне проходження, що спонукає гравця проходити рівень декілька разів [5].

Спочатку гра була орієнтована на мобільні платформи, але згодом адаптована під PC та Web. Станом на сьогодні існує декілька розширень гри, що вводять нові башти, локації, героїв.

1. <http://www.ign.com/articles/2011/10/31/dungeon-defenders-review>
2. <https://www.slant.co/topics/1983/viewpoints/15/~pc-tower-defense-games~dungeon-defenders>
3. <http://www.ign.com/articles/2011/10/05/orcs-must-die-review>
4. <https://www.slant.co/topics/1983/viewpoints/6/~pc-tower-defense-games~orcs-must-die-unchained>
5. <http://www.ign.com/articles/2012/01/31/kingdom-rush-review>
6. <https://www.slant.co/topics/1983/viewpoints/2/~pc-tower-defense-games~kingdom-rush>

## 1.4 Аналіз методів розв’язання поставленої задачі

Розробка та застосування нечіткого виведення включає в себе декілька етапів:

1. Визначення множини вхідних змінних.
2. Формування бази правил.
3. Фазифікація вхідних лінгвістичних змінних.
4. Агрегування попередніх умов.
5. Активізація попередніх висновків.
6. Акумулювання висновків.
7. Дефазифікація вихідних лінгвістичних змінних [http://nrsu.bstu.ru/chap27.html].

Лінгвістична змінна визначається набором <B, T, X, G, M>.

B – найменування лінгвістичної змінної.

T – множина значень лынгвыстичноъ змынноъ(терм-множина), що являє собою найменування нечітких змінних та область визначення кожної з них на множині Х. Т – базова терм-множина.

G – синтаксична функція, що дозволяє генерувати нові терми з елементів терм-множини Т.

М – семантична функція, що дозволяє сформувати відповідну нечітку множину для кожної базової терми та терми, що була створена функцією G [книга].

Нові лінгвістичні змінні формуються за допомогою операторів: «Не», «Дуже», «Більш-менш».

Оператору «Не» відповідає формула:

|  |  |
| --- | --- |
|  | (x.1) |

Оператору «Дуже» відповідає операція концентрації:

|  |  |
| --- | --- |
|  | (x.1) |

Оператору «Більш-менш» відповідає операція розтягнення:

|  |  |
| --- | --- |
|  | (x.1) |

Фазифікація – це застосування функції приналежності до чітких значень змінних для визначення їх ступеню істинності до тої чи іншої терми.

В процесі агрегування примітивні умови об’єднуються за допомогою операцій кон’юнкції та диз’юнкції. Одним з підходів до операторів об’єднання та перетину полягає в їх визначенні в класі трикутних норм та конорм.

Трикутна норма (Т-норма) – дійсна бінарна функція Т: [1, 0] × [0, 1] → [0, 1], що задовольняє наступним вимогам:

1. Т(0, 0) = 0; Т(µА, 1) = µА; Т(1, µА) = µА – обмеженість;
2. Т(µА, µВ) ≤ Т(µC, µD), якщо µА ≤ µC, µB ≤ µD – монотонність;
3. Т(µА, µВ) = Т(µВ, µА) – комутативність;
4. Т(µА, Т(µB, µC)) = Т((µА, µB), µC) – асоціативність;

Трикутна конорма (Т-конорма, S-норма) – дійсна бінарна функція S: [1, 0] × [0, 1] → [0, 1], з властивостями:

1. S(1, 1) = 1; S(µА, 0) = µА; S(0, µА) = µА – обмеженість;
2. S(µА, µВ) ≥ S(µC, µD), якщо µА ≥ µC, µB ≥ µD – монотонність;
3. S(µА, µВ) = S(µВ, µА) – комутативність;
4. S(µА, Т(µB, µC)) = S((µА, µB), µC) – асоціативність;

В таблиці 1.1 наведено приклади норм та конорм [http://sernam.ru/book\_gen.php?id=19].

Таблиця 1.1

Приклади трикутних норм

|  |  |  |  |
| --- | --- | --- | --- |
| № | Т(a, b) | S(a, b) | Умови |
| 1 | min(a, b) | max(a, b) |  |
| 2 | ab | a+b-ab |  |
| 3 | max(a+b-1, 0) | min(a+b, 1) |  |
| 4 | якщо b = 1, то a  якщо a = 1, то b  якщо a, b ≠ 1, то 0 | якщо b = 0, то a  якщо a = 0, то b  якщо a, b ≠ 0, то 0 |  |
| 5 |  |  | y > 0 |
| 6 |  |  | α є [0,1] |
| 7 |  |  | p ≥ 1 |
| 8 |  |  | λ ≥ -1 |

Активізація полягає в находженні степені істинності кожного з елементарних висловлювань (попередніх висновків).

Ціль акумуляції полягає в об’єднанні всіх ступенів істинності попередніх висновків для отримання функції приналежності кожної з вихідної змінної. Зазвичай об’єднання відбувається за допомогою операції максимум.

Дефазифікація – процес отримання чіткого значення з функції приналежності вихідної лінгвістичної змінної [http://nrsu.bstu.ru/chap27.html].

Існує декілька методів приведення до чіткості:

1. Центроїдний метод, непереривний варіант:

|  |  |
| --- | --- |
|  | (x.1) |

де C(z) – функція приналежності вихідної змінної.

Дискретний варіант:

|  |  |
| --- | --- |
|  | (x.1) |

1. Перший максимум:

|  |  |
| --- | --- |
|  | (x.1) |

Береться мінімальне чітке значення, в якому функція приналежності досягає свого максимуму.

1. Середній максимум:

|  |  |
| --- | --- |
|  | (x.1) |

де G – множина елементів, в яких досягається максимум.

Дискретний варіант:

|  |  |
| --- | --- |
|  | (x.1) |

1. Критерій максимуму. Чітке значення береться довільно з множини елементів, в яких досягається максимальне значення функції:

|  |  |
| --- | --- |
|  | (x.1) |

1. Висотна дефазифікація. Елементи області визначення Ω, для яких значення функції приналежності менше, ніж деякий рівень α, в розрахунок не приймаються:

|  |  |
| --- | --- |
|  | (x.1) |

де Сα – нечітка множина α-рівня.

Отже, кожний етап нечіткого виведення має декілька варіантів його рішення. Ефективним методом є комбінування тих чи інших правил та методів в залежності від поставлених цілей.

## 1.5 Постановка задач дослідження

## 1.6 Висновки

# 2 РОЗРОБКА МОДЕЛІ ІГРОВОГО ДОДАТКУ

## 2.1 Розробка моделей та структур даних

Основними ігровими одиницями в є воїн, башта та замок.

Воїн – основна бойова одиниця сторони, що нападає, описується наступним набором характеристик:

* здоров’я (Healthw);
* захист від фізичного пошкодження (Armorph);
* захист від магічного пошкодження (Armormg);
* розмір (Size);
* швидкість (Velocity);
* потужність (Power).

Здоров’я – ціле беззнакове число. Якщо рівень здоров’я воїна падає до 0 – він гине.

Захист від фізичного пошкодження – ціле беззнакове число в діапазоні [0; 20]. Враховується при виведенні коефіцієнта фізичного пошкодження. Впливає на результуючу швидкість воїна.

Захист від магічного пошкодження – ціле беззнакове число в діапазоні [0; 20]. Впливає на коефіцієнт магічного пошкодження.

Розмір – переліковий тип, що приймає два значення: «Малий», «Великий». Розмір суттєво впливає на результуючу швидкість воїна, а також, враховується при виведенні коефіцієнту фізичного пошкодження.

Швидкість – беззнакове число з плаваючою комою, описує швидкість переміщення воїна в одиницях за секунду.

Потужність – беззнакове ціле число, що характеризує пошкодження, що завдасть воїн замку, у випадку, якщо дійде до нього.

Башта – основна бойова одиниця сторони, що захищається, має наступні властивості:

* фізичне пошкодження (Damageph);
* магічне пошкодження (Damagemg);
* фізична атака (Attackph);
* магічна атака (Attackmg);
* тип фізичної атаки (AttackTypeph);
* час перезарядки (RechargeTime).

Фізичне пошкодження – пара цілих беззнакових чисел, що характеризує мінімальне та максимальне фізичне пошкодження.

Магічне пошкодження - пара цілих беззнакових чисел, що характеризує мінімальне та максимальне магічне пошкодження.

Фізична атака – ціле беззнакове число в діапазоні [0; 20], що враховується при виведенні коефіцієнта фізичного пошкодження.

Магічна атака – ціле беззнакове число в діапазоні [0; 20], що враховується при виведенні коефіцієнта магічного пошкодження.

Тип фізичної атаки – ціле беззнакове число, являє собою бітову маску: перший біт – стрілкова атака, другий – вибухова атака. Впливає на коефіцієнт фізичного пошкодження.

Час перезарядки - беззнакове число з плаваючою комою, характеризую час перезарядки башти після пострілу в секундах.

Замок – ціль обох сторін. Сторона нападника повинна знищити його, сторона захисту – відбити всі атаки від нього. Замок має наступні характеристики:

* здоров’я (Healthc).

Здоров’я – ціле беззнакове число.

## 2.2 Розробка нечітких правил

Правила нечіткої логіки будуть використані для виведення параметрів:

* коефіцієнта фізичного пошкодження;
* коефіцієнта магічного пошкодження;
* швидкості воїна;

Результуюче пошкодження, що може завдати башта воїну:

|  |  |
| --- | --- |
| Damage = Damaget×Damagec , | (x.1) |

де Damaget – пошкодження башти;

Damagec – коефіцієнт пошкодження.

Коефіцієнт пошкодження виводиться з нечітких правил.

Для розрахунку магічного пошкодження необхідно ввести лінгвістичні змінні: Attackmg – магічна атака; Defensemg – захист від магічного пошкодження; Damagemgc – коефіцієнт магічного пошкодження.

Визначимо множину термів для вхідної лінгвістичної змінної Attackmg: Adept, Wizard, Master.

Функція приналежності терму Adept задається трапецією: 0; 3; 4; 9.

Функція приналежності терму Wizard задається трапецією: 4; 9; 12; 20.

Функція приналежності терму Master задається прямою: 12; 20.

На рисунку 1 наведено функції приналежності лінгвістичної змінної Attackmg до термів.



Рисунок 1 – Функції приналежності лінгвістичної змінної Attackmg

Визначимо множину термів для вхідної лінгвістичної змінної Defensemg: Sensitive, Resistance, Antimage.

На рисунку 2 наведено функції приналежності лінгвістичної змінної Defensemg до термів.



Рисунок 2 – Функції приналежності нечіткої змінної Defensemg

Функція приналежності терму Sensitive задається прямою: 8; 0.

Функція приналежності терму Resistance задається трапецією: 0; 8; 12; 20.

Функція приналежності терму Antimage задається прямою: 12; 20.

Визначимо множину термів вихідної лінгвістичної змінної Damagemgc: Normal, Critical, Double.

Функція приналежності терму Normal задається прямою: 0,0; 1,0.

Функція приналежності терму Critical задається прямою: 0,8; 1,5.

Функція приналежності терму Double задається прямою: 1,0; 2,0.

На рисунку 3 наведено функції приналежності лінгвістичної змінної Damagemgc до термів.



Рисунок 3 – Функції приналежності нечіткої змінної Damagemgc

Для кон’юнкція обрано T-норму мінімум.

Для диз’юнкції – S-норму максимум.

Для активізації – Т-норму мінімум.

Для акумулювання вихідної змінної Damagemgc – S-норму максимум.

Для дефазифікації вихідної змінної Damagemgc – правило першого максимуму.

Всі правила виведення коефіцієнта магічного пошкодження наведені в таблиці 1.

Таблиця Х.1

Правила виведення коефіцієнта магічного пошкодження

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attackmg Defensemg | Adapt | Wizard | Master | Дуже Master |
| Більш-менш  Sensitive | Damagemgc є Normal | Damagemgc є дуже Normal | Damagemgc є Critical | Damagemgc є Double |
| Sensitive | Damagemgc є 0.8×Normal | Damagemgc є Normal | Damagemgc є дуже Normal | Damagemgc є 0.9×Critical |
| Resistance | Damagemgc є 0.5×Normal | Damagemgc є 0.6×Normal | Damagemgc є 0.75×Normal | Damagemgc є 0.9×Normal |
| Antimage | Damagemgc є  0×Normal | Damagemgc є 0.2×Normal | Damagemgc є 0.3×Normal | Damagemgc є 0.4×Normal |

На перетині стовпця і рядка формується правило з кон’юнкцією функцій приналежності до відповідних терм лінгвістичних змінних Attackmg та Defensemg. Для кращого регулювання виведення введено вагові коефіцієнти. Коефіцієнти множаться на ступінь істинності передумов правил.

Чітке значення коефіцієнта пошкодження виводиться з формули:

|  |  |
| --- | --- |
| dc0 = min(d|C(d) = maxu(C(u))), | (x.1) |

де C(u) – функція приналежності, отримана в результаті акумуляції всіх правил.

Функція приналежності C(u) отримується з формули:

|  |  |
| --- | --- |
| C(u) = maxij(min[min(Ci(Defensemg); Cj(Attackmg))×Wij; Cij(Damagemgc)]), | (x.1) |

де Ci(Defensemg) – функція приналежності лінгвістичної змінної Defensemg до і-го терму;

Cj(Attackmg) – функція приналежності лінгвістичної змінної Attackmg до j-го терму;

Wij – ваговий коефіцієнт правила i-го рядка та j-го стовпця;

Cij(Damagemgc) – функція приналежності лінгвістичної змінної Damagemgc правила i-го рядка та j-го стовпця;

## 2.3 Розробка структури інтерфейсу

## 2.4 Висновки

# 3 ВАРІАНТНИЙ АНАЛІЗ ТА ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ ІГРОВОГО ДОДАТКУ

## 3.1 Аналіз особливостей розробки ігрового додатку

Розробка відеоігор - це сфера, що поєднує в собі області комп'ютерної науки, математики та виробництва. Відповідно ролі в ігровій індустрії різноманітні [1]:

* гейм продюсер;
* художник 2d, 3d;
* програміст;
* графічний дизайнер;
* креативний директор;
* сценарист;
* аудіо фахівець;
* левел дизайнер.

На сьогодні існує багато інді-студій. Інді-ігри – це ігри, що створенні одним чи декількому розробниками без будь-якої фінансової підтримки. В таких студіях один розробник може виконувати декілька ролей.

Розробка відеогри складається з наступних етапів [2]:

1. Розробка концепту гри. Концепт містить в собі жанр гри, опис ігрового процесу, особливості, сценарій, цільову аудиторію, цільові платформи, графік робіт, командні вимоги, аналіз ризиків.
2. Документ ігрового дизайну. Документ описує особливості гри в деталях і є відправним пунктом для розробників. Він може змінюватися протягом всього проекту.
3. Розробка прототипу. Ціллю прототипа є перевірка головних механік гри, без занурення в деталі. В прототипі не використовуються контент високої якості, він може містити примітивні фігури.
4. Дизайн гри. На поточному етапі визначається контент гри, його обсяг.
5. Програмування.
6. Створення рівнів.
7. Наповнення рівнів звуками.
8. Тестування.

Головним інструментом розробки ігрових додатків є ігровий рушій. Існують рушії, що дозіоляють розробити гру, не маючи навиків програмування. Також, необхідне програмне забезпечення для роботи з моделями, спрайтами, текстурами, скелетами, анімаціями, звуками.

Для організації роботи, як і для розробки будь-якого іншого програмного забезпечення, необхідно інструменти для керування задачами, тестування, автоматизації процесів.

Необхідно налагодити процеси експорту всіх ресурсів в ігровий рушій, збирання проміжних виконуваних файлів для тестування.

Отже, розробка відеоігор охоплює широкий спектр робіт.

## 3.2 Варіантний аналіз вибору мови розробки ігрового додатку

Існує безліч парадигм програмування [3]:

* Імперативне: програмування заданої послідовності команд, що змінюють стан.
* Декларативне: програмування результату, а не те, як його отримати.
* Структуроване: програмування за допомогою послідовних конструкцій, розгалуження та циклів.
* Функціональне: програмування за допомогою функцій, відсутній глобальний стан.
* Комбінаторне: програмування без змінних.
* Об’єктно-орієнтоване: порограмування шляхом визначення об’єктів, що взаємодіють одне з одним, кожен об’єкт має свій стан.
* Подіє-орієнтоване: програмування з слухачами асинхронних дій.
* Потоко-орієнтоване: програмування процесу взаємодію одне з одним за допомогою визначених каналів.
* Логічне: програмування визначається множиною правил та фактів.
* Обмеженнями: програмування з використанням обмежень, базується на пошуку рішень, що відповідають обмеженню.
* Рефлекторне: програмування з використанням коду програми, як даних, що дозволяє змінювати структури самої програми на етапі виконання.
* Масиво-орієнтоване: програмування роботи з даними, як з масивами.

Для зручності розробки ігрового додатку мова програмування повинна підтримувати об’єктно-орієнтоване програмування.

Для порівняння обрано мови, що широко використовуються для розробки на PC та підтримують парадигму об’єктно-орієнтованого програмування: С++, C# (таблиця 3.1).

Таблиця 3.1

Порівняльний аналіз мов програмування

|  |  |  |
| --- | --- | --- |
| Критерій | С++ | C# |
| 1 | 2 | 3 |
| Процедурне програмування | Підтримується | Не підтримується |
| Об’єктно-орієнтоване програмування | Підтримується | Підтримується |
| Множинне наслідування | Наслідування від декількох класів; можливе віртуальне наслідування | Наслідування лише від одного класу та реалізація більше одного інтерфейсу |
|  |  |  |
|  |  | Продовження таблиці 3.1 |
| 1 | 2 | 3 |
| Можливості роботи з пам’яттю | Об’єкти можна створювати як на кучі (динамічно), та і на стеці (статично); вказівники надають можливість доступитися до пам’яті. | Всі об’єкти створюються динамічно; посилання зберігають на стеці; об’єкти, що не використовуються, знищуються автоматично |
| Константні аргументи методів та функцій | Підтримується, оскільки const надає лише права на зчитування змінної. | Не підтримується, оскільки const вказує на константу етапу компіляції |
| Шаблони зі змінною кількістю параметрів | Підтримується | Не підтримується |
| Портативність | Виконуваний файл можна зібрати на будь-яку платформу | Файл збирається за допомогою CLR (Common Language Runtime) в PE (Portable Executable) і виконується на .NET |

На С++ існує ризик витоку пам’яті, проте розширенні можливості для роботи з нею дозволяють організувати роботу з динамічною пам’яттю під потреби проекту, що покращить продуктивність додатку. Також, суттєвим є портативність мови.

Отже, для розробки ігрового додатку обрано мову програмування С++.

## 3.3 Варіантний аналіз вибору середовища розробки ігрового додатку

Інтегроване середовище розробки (IDE) – програмне забезпечення, що надає розширенний фукнціонал під час розробки: структурування проекту, підсвітка коду, синтаксична перевірка, автоматизований процес збирання проекту, автодоповнення коду, аналізатори коду, налагодження коду, тощо.

Для розробки на С++ середовище розробки повинне підтримувати автоматизований процес компілювання. Компіляція С++ коду складається з наступних етапів [4]:

1. Препроцесор копіює вміст включених файлів заголовків у файл вихідного коду, генерує код макросу та замінює символьні константи, визначені за допомогою #define їх значеннями.
2. Розширений файл вихідного коду збирається в мову асемблера платформи.
3. Асемблерний код збирається в об'єктний код платформи.
4. Файли об'єктного коду зв'язуються разом з файлами об'єктного коду, на які вони посилаються (використання функцій, класів, глобальних змінних з іншого файлу).

Розглянемо IDE, що підтримують описану можливість:

* Microsoft Visual Studio;
* Code::Blocks.

Microsoft Visual Studio є стандартною IDE від корпорації Microsoft. Воно підтримує широкий асортимент мов, а також забезпечує велику різноманітність інструментів тестування та сумісності. Воно також пов’язане з рядом сервісів Microsoft, що робить його основним вибором для розробки на ПК [5].

Особливості Microsoft Visual Studio [5]:

* має безкоштовні версії;
* підтримка багатьох мов програмування;
* підтримка Microsoft;
* широкий набір можливостей налаштування вікон;
* підсвітка синтаксису;
* автодоповнення;
* відлагодження;
* інтегрованість з багатьма сервісами Microsoft;
* статичний аналізатор коду;
* генерація UML діаграм з коду;
* профайлер продуктивності.

Code::Blocks – безкоштовне, портативне інтегроване середовище розробки з відкритим вихідним кодом для розробки на С, С++ та Fortran. Побудоване навколо архітектури плагінів, тобто, може легко бути налаштована під індивідуальні потреби.

Особливості Code::Blocks [5]:

* безкоштовне;
* доступне на всі операційні системи;
* гнучкі можливості налаштування;
* відлагодження;
* підсвітка синтаксису;
* автодоповнення.

Однією з переваг Microsoft Visual Studio є профайлер продуктивності, що допоможе знайти вузькі місця в коді, що дуже важливо для розробки ігрового додатку.

Отже, для розробки ігрового додатку обрано IDE Microsoft Visual Studio.

## 3.4 Варіантний аналіз вибору ігрового рушія

Ігровий рушій – одне з ключових програмних забезпечень в процесі розробки ігрового додатку. Він пропонує компоненти для багаторазового використання: завантаження, відображення та анімація моделей, визначення колізії між об’єктами, фізика, обробка сигналів вводу, графічний інтерфейс користувача, штучний інтелект, редагування ландшафту, робота зі звуком [6].

Особливістю рушія є те, що він бере на себе всю специфічну роботу з апаратним забезпеченням для різних платформ. Це дає можливість художникам, розробникам рівнів, програмістам зосередитися на самій грі, на її ігровому процесі, не вникаючи в специфіку реалізації тої, чи іншої фукнціональної характеристики на різні платформи.

Ігровий додаток буде реалізований в двовимірному просторі з наступних причин:

* логіка гри не потребує третього виміру;
* для розробки тривимірних моделей необхідні додаткові інструменти;
* ігровий процес жанру Tower Defense базується на стратегічному мисленні, а не на графічному наповненні.

Популярними ігровими рушіями для розробки 2d ігор на сьогодні є: Cocos2d та Unity.

Cocos2d має безліч розширень, одне з широко використовуваних - Cocos2d-x. Cocos2d-x - багатоплатформенний ігровий рушій з відкритим вихідним кодом, і є абсолютно безкоштовним. Він надає можливість публікувати ігри для мобільних пристроїв і настільних комп'ютерів, зокрема iPhone, iPad, Android, Kindle, Windows і Mac. Cocos2d-x написано на C++, тому його можна збирати на будь-якій платформі. Оскільки Cocos2d-x з відкритим вихідним кодом – він розвивається надзвичайно швидко. Cocos2d-x версії 3 підтримує C++ 11. Він також підтримує 3D і має покращений процес рендерингу [7].

В таблиці 3.2 наведено порівняльний аналіз Cocos2d-x та Unity [7].

Таблиця 3.2

Порівняльний аналіз Cocos2d-x та Unity

|  |  |  |
| --- | --- | --- |
| Критерій | Cocos2d-x | Unity |
| Мови  програмування | C++, Lua, javascript | C#, javascript |
| Поріг входження | Середній, оскільки використовується досить багато стороннього програмного забезпечення | Низький |
| Онлайн підтримка | Значна і стрімко розвивається; недолік: значна частина книг китайською мовою | Дуже висока, власний магазин з готовими скріптами та ресурсами |
| Стороннє програмне забезпечення, плагіни | Підтримка зовнішнього програмного забезпечення (Tiledmap, bitmap, тощо), прості у використанні | Плагіни заплутані і важкі у використанні |
| Платформи | Windows, Mac OS, Linux, Android, iOS, HTML5, Tizen, | Windows, Mac OS, Linux, Android, iOS, HTML5, Tizen,  PS4, Xbox One, Wii |
| Успішні проекти | BadLand, Castle Clash, Hill Climb Racing, Geometry Dash Lite, Flow Free | Rust, Superhot, Firewatch, Deadlock, Slender |
| Доступність | Відкрий вихідний код, безкоштовне комерційне використання | Закритий вихідний код, умовно безкоштовне комерційне використання |

Однією з переваг Cocos2d-x є підтримка C++, за рахунок відкритого коду, рушій можна легко розширити під власні потреби, також, він споживає менше ресурсів. Unity надає широкий набір функціональних характеристик, проте для розробки двовимірної Tower Defense гри рушій буде ресурсозатратним.

Отже, для розробки ігрового додатку обрано ігровий рушій Cocos2d-x.

## 3.5 Варіантний аналіз вибору бібліотеки для роботи з нечіткою логікою

Нечітка логіка як розділ математики бере свій початок у 1965 році і була введена Лотфі Заде. З розвитком програмування з’явилося багато реалізацій нечіткої логіки. Універсальним математичним інструментом, що надає можливості роботи з нечіткою логікою є MATLAB. Проте для розробки гри не потрібні всі можливості, що надає MATLAB.

Тому розглянемо деякі бібліотеки:

* Fuzzy Lite (FL);
* Free Fuzzy Logic Library (FFLL);
* Embedded Fuzzy Logic Library (EFLF).

В таблиці 3.3 наведено порівняльний аналіз бібліотек.

Таблиця 3.3

Порівняльний аналіз бібліотек нечіткої логіки

|  |  |  |  |
| --- | --- | --- | --- |
| Критерій | FL | FFLL | EFLF |
| Графічний редактор | Платний | Відсутній | Відсутній |
| Мова нечіткої логіки | Підтримується | Підтримується | Відсутня |
| Підтримка обчислювальних платформ | Відсутня | Відсутня | Arduino |
| Парадигма програмування | ООП | ООП | ООП, Процедурна |

Free Fuzzy Logic Library призначена для розробки штучного інтелекту, тому є добре оптимізована, має інтерфейс для роботи із С-кодом. Бібліотека вже не підтримується розробником.

Embedded Fuzzy Logic Library розроблена для використання з обчислювальними платформами, підтримка розробників не значна.

Fuzzy Lite має активну онлайн підтримку, надає широкий спектр функціональних можливосте для нечіткого виведення (фазифікація, агрегація, способи активізації, акумулювання, дефазифікація, t-, s-норми). Також, бібліотека підтримує мову для створення нечітких правил: Fuzzy Lite Language.

Отже, для розробки нечітких правил та використання їх в ігровому додатку обрано бібліотеку Fuzzy Lite.

## 3.6 Висновки

Література

1. <https://www.internationalstudent.com/study-video-game-development/what-is-video-game-development/>
2. <https://en.wikipedia.org/wiki/Video_game_development>
3. <http://cs.lmu.edu/~ray/notes/paradigms/>
4. <http://faculty.cs.niu.edu/~mcmahon/CS241/Notes/compile.html>
5. Procedural Content Generation for C++ Game Development.
6. <https://www.gamecareerguide.com/features/529/what_is_a_game_.php>
7. Конфа

# 4 РОЗРОБКА ІГРОВГО ДОДАТКУ

## 4.1 Розробка рівня

Для розробки рівня гри використано Tiled Map Editor.

Tiled Map Editor – це 2D-редактор рівня, що призначений для розробки контенту відеогри. Його основна функція полягає в редагуванні карт, що складається з плиток різних форм, також підтримує розміщення зображень, способи анотування рівня з додатковою інформацією, що може бути використана в грі [1].

Для заповнення рівня використовується Tileset (набір плиток). Спрайт лист для заповнення рівня, зображений на рисунку 4.1.



Рисунок 4.1 – Спрайт лист для рівня

Додамо його до набору плиток в редактор (рисунок 4.2).

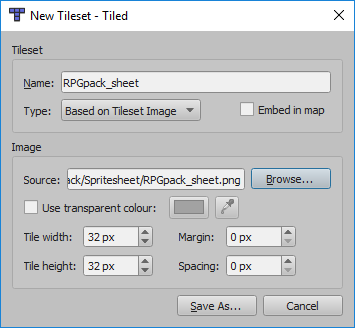


Рисунок 4.2 – Вікно додавання нового набору плиток

Плитка висотою 32 пікселі та довжиною 32 пікселі, в спрайт листі відступів між плитками немає. Редактор автоматично розіб’є малюнок на елементи по 32х32 пікселі, після чого їх можна буде вибирати та встановлювати в рівні.

Кожен рівень може складатися з декількох шарів. Створемо шар Ground та зафарбуємо його травою. Щоб зафарбувати весь шар одним елементом, можна скористатися інструментом для заповнення, що знаходиться на панелі інструментів (рисунок 4.3).



Рисунок 4.3 – Інструмент для заповнення

Перед заповненням рівня спочатку створемо місцевість «Road». Необхідно задати рамки для місцевості, як прямі, так і на поворотах (рисунок 4.4).

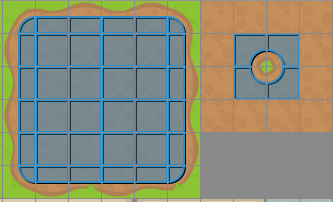


Рисунок 4.4 – Рамки для місцевості «Road»

Редактор буде автоматично підбирати необхідні плитки границі місцевості. Створемо новий шар для дороги. Створимо дві дороги, що прямують до будівлі. Для будівлі заведемо інші шари, один – для стін, другий – для даху, третій – для вікон (рисунок 4.5).

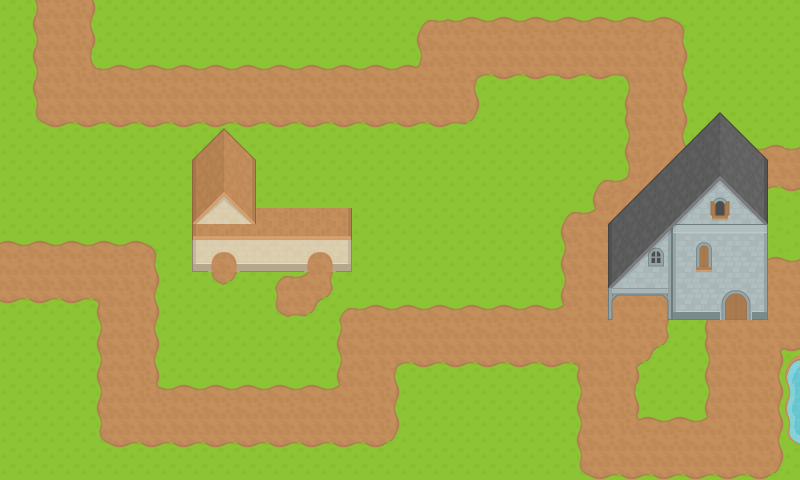


Рисунок 4.5 – Рівень з дорогами та будівлями

Будівля, що знаходиться праворуч є ціллю гравця. Tiled Map Editor дозволяє надавати властивості об’єктам. Створимо окремий шар об’єктів, на якому будуть розміщенні точки генерації воїнів для кожної дороги. Ще один шар необхідний для розташування ключових точок, по яким будуть рухатися воїни. Кожна ключова точка матиме власний індекс та індекс наступної точки (рисунок 4.6).

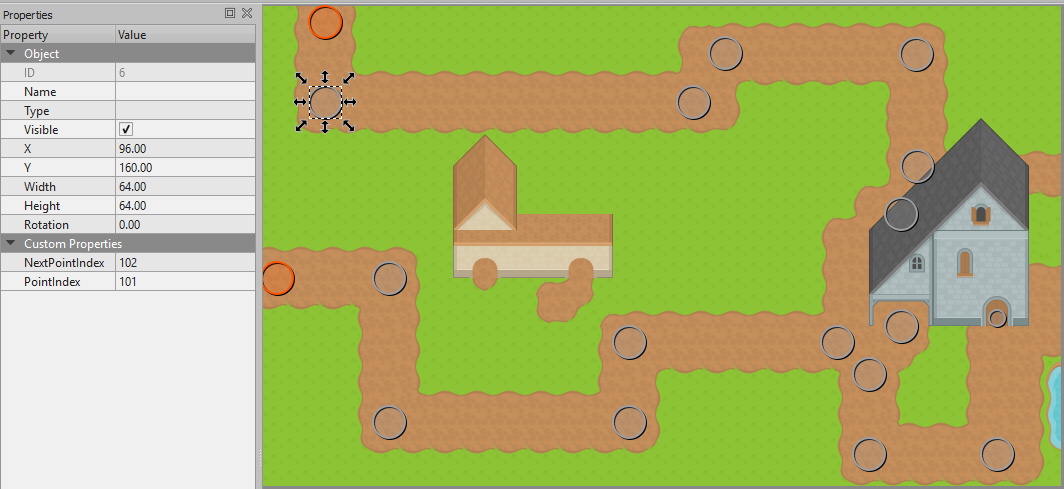


Рисунок 4.6 – Ключові точки шляхів

Червоним колом зображені точки генерації воїнів, сірим – ключові точки. Як тільки воїн дійте до останнього малого кола на дверях будуник – будинок втрачає здоров’я. Наступний індекс кінцевої ключової точки менший 0, що необхідно для ідентифікація кінця шляху. Кожен воїн генерується на випадковій позиції червоного кола. Кожна наступна точка вибирається випадково з кола наступної ключової точки. Це надасть різноманіття пересуванню воїнів.

Зробимо те ж саме для місць будівничтва башт. Тільки в цьому випадку створемо окрему місцевість (Рисунок 4.7).

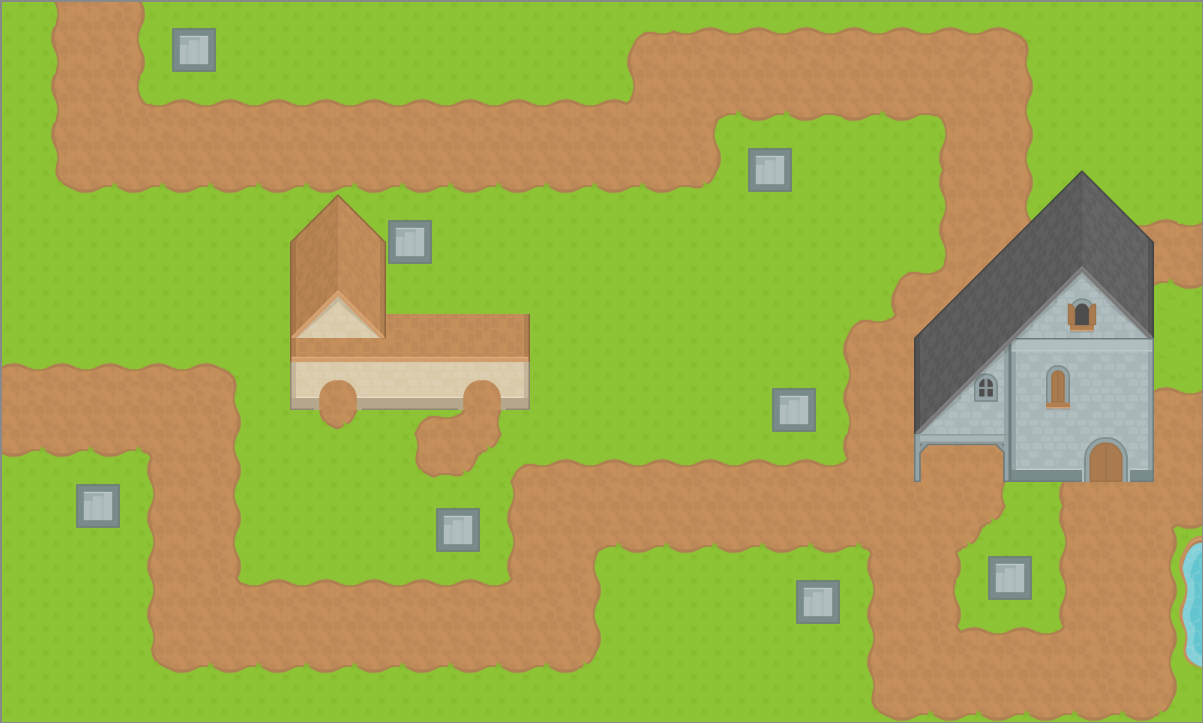


Рисунок 4.7 – Місця для побудови башт

Додамо ще два шари «Trees» - для дерев, та «Fence» - для огорожі та діжок (рисунок 4.8).

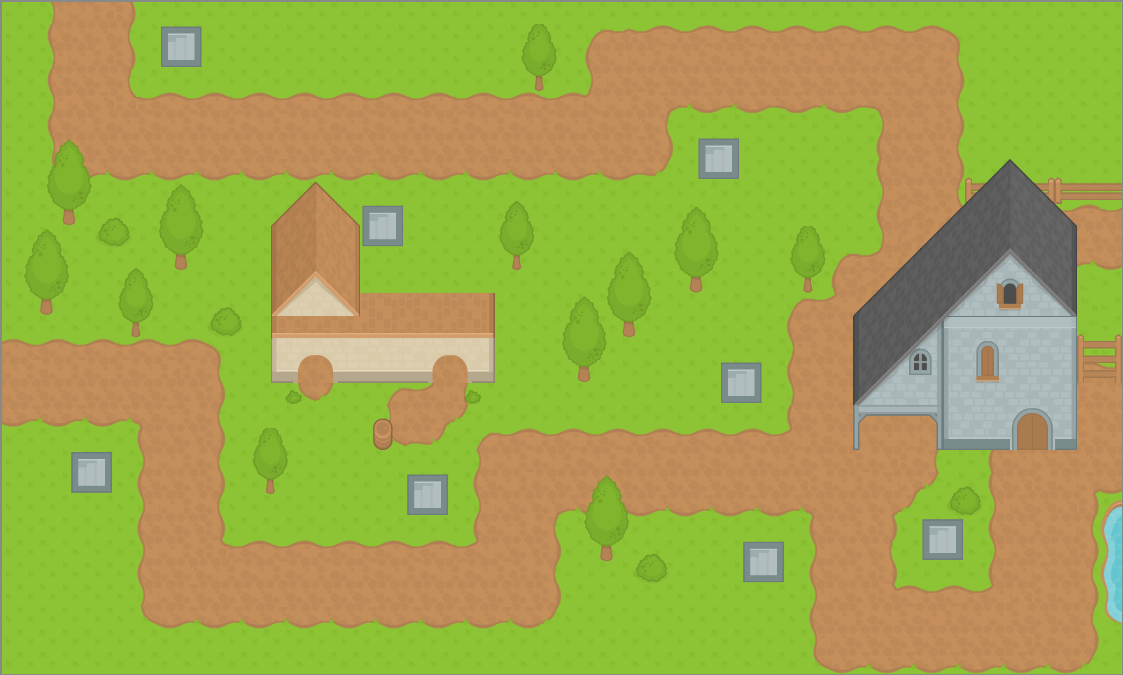


Рисунок 4.8 – Кінцевий варіант першого рівня

На рисунку 4.9 зображено шари рівня.

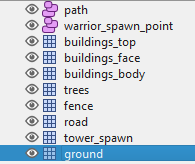


Рисунок 4.9 – Шари рівня

Cocos2d має готову реалізацію для завантаження файлів, експортованих з Tiled Map Editor. Для цього слід використати клас TMXTiledMap, створити через статичний метод create, передавши шлях до файлу tmx.

## 4.2 Розробка анімацій

Розглянемо процес створення анімаці на основі звичайного воїна (рисунок 4.10) [3].



Рисунок 4.10 – Воїн

Анімація – це послідовність кадрів. Cocos2d-x програє анімацію з файлів plist. Файл містить xml, що описує кожен кадр анімації, та позицію відповідного спрайта в спрайт листі. На рисунку 4.11 наведено опис другого кадру анімації бігу.

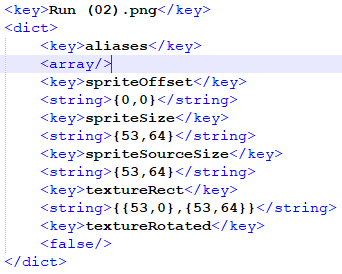


Рисунок 4.11 – Опис кадру анімації

Відповідно біля кожного plist файл повинен бути спрайт лист з такою ж назвою, як і plist. Для пакування спрайтів в листи використовується програмне забезпечення Texture Packer. Додаток підтримує cocos2d-x, тому експортує одразу два файли: plist та png (рисунок 4.12).

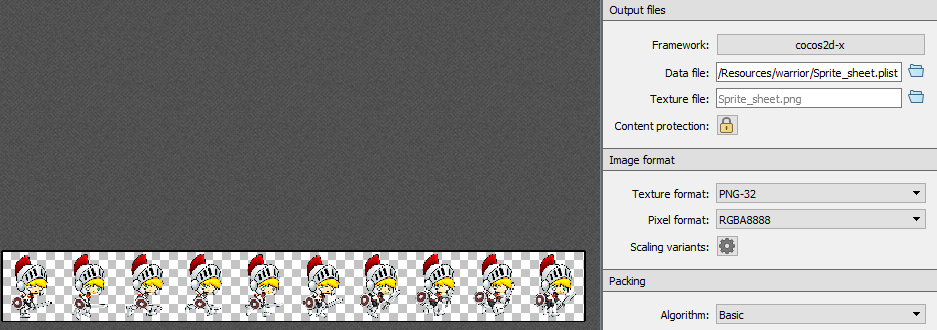


Рисунок 4.12 – Експорт спрайтів в plist

Для завантаження спрайт листа необхідно використати клас SpriteFrameCache, він завантажить всі кадри з plist, створити Animation на основі вестора кадрів, а також action на основі анімації (рисунок 4.13).



Рисунок 4.13 – Завантаження plist

## 4.х Висновки

Література

1. <http://doc.mapeditor.org/en/latest/manual/introduction/>
2. <https://opengameart.org/content/rpg-pack-base-set>
3. <https://www.gameart2d.com/the-knight-free-sprites.html>

# 5 ТЕСТУВАННЯ ІГРОВОГО ДОДАТКУ

## 4.1 Аналіз методів тестування

Тестування – методика підвищення якості, підкріплена багатьма дослідженнями і багатим досвідом розробки комерційних додатків. Існує безліч видів тестування: одні зазвичай виконують самі розробники, а інші - спеціалізовані групи. Види тестування перераховані нижче [27]:

* Блоковим тестуванням називають тестування повного класу, методу або невеликого програми, написаної одним програмістом або групою, що виконується окремо від інших частин системи [27].
* Тестування компонента - це тестування класу, пакету, невеликого додатки або іншого елемента системи, розробленого кількома програмістами або групами, що виконується в ізоляції від інших частин системи [27].
* Інтеграційне тестування - це спільне виконання двох або більше класів, пакетів, компонентів або підсистем, створених кількома програмістами або групами [27].
* Регресивним тестуванням називають повторне виконання тестів, спрямоване на виявлення дефектів у програмі, що вже пройшла цей набір тестів [27].
* Тестування системи - це виконання програмного забезпечення в його остаточної конфігурації, інтегрованого з іншими програмними та апаратними системами [27].

Існує два основних методи тестування: тестування «білої скриньки», тестування «чорної скриньки» [28].

Одним із способів вивчення поставленого питання є дослідження стратегії «чорної скриньки», тестування з керуванням за даними, або тестування з керуванням по входу-виходу. При використанні цієї стратегії програма розглядається як чорна скринька. Таке тестування має на меті з'ясування обставин, в яких поведінка програми не відповідає її специфікації. Тестові ж дані використовуються тільки у відповідності зі специфікацією програми (без урахування знань про її внутрішню структуру) [28].

Стратегія «білої скриньки», або стратегія тестування, керованого логікою програми, дозволяє досліджувати внутрішню структуру програми. В цьому випадку тестувальник отримує тестові дані шляхом аналізу логіки програми При використанні даної стратегії часто не використовується специфікація програми [28].

Для тестування програмного додатку було обраного стратегію «чорної скриньки» [28].

## 4.2 Інструкція тестування

## 4.3 Висновки

# ВИСНОВКИ

В бакалаврській дипломній роботі було проаналізовано сучасний стан та жанри мобільних ігор. Проведено порівняльний аналіз аналогів з ігровим додатком, що розробляється. В ході аналізу встановлено переваги додатку над аналогами, розглянуто актуальність розробки. Проаналізовано методи розв’язання проблеми. Для реалізації додатку обрано ігровий двигун Unity3D.

Розроблено структуру інтерфейсу додатку. Визначено всі панелі головного меню. Встановлено множину основних моделей, що складають гру, визначено їх функціонал. Розроблено діаграму класів відповідно до моделей. Обрано патерни проектування для ефективної розробки. Розроблено алгоритм подолання нескінченності.

Проведено варіантний аналіз засобів реалізації. В ході аналізу для розробки ігрового додатку було обрано мову програмування C# та IDE Visual Studio 2015.

Розроблено додаток “Unknown enemy” з використанням готових компонентів Unity3D та створенням власних скриптів. В грі реалізовано наступні функціольні характеристики:

* Досягнення.
* Система рейтингу.
* Можливість знищення ворогів.
* Магазин та внутрішньо-ігрова валюта.
* Випадкова генерація рівня.

Програмне забезпечення протестовано на різних мобільних пристроях. В ході тестування помилок виявлено не було. Для тестування було обрано метод «чорної скриньки».

# ПЕРЕЛІК ПОСИЛАНЬ

1. Відеогра. Матеріал з Вікіпедії — вільної енциклопедії. – [Електронний ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/Відеогра>
2. Почему Видеоигры так популярны сегодня? – [Електронний ресурс]. – Режим доступу:

<http://teachmen.ru/others/articles/pochjemu-vidjeoigry-tak-populjarny-sjegodnja.html>

1. Відеоігри можуть бути корисними для здоров’я – [Електронний ресурс]. – Режим доступу:

<http://zhyvyaktyvno.org/index.php/news/pogratisya-zalyubki>

1. Мобільний телефон. Матеріал з Вікіпедії — вільної енциклопедії. – [Електронний ресурс]. – Режим доступу:

<https://uk.wikipedia.org/wiki/Мобільний_телефон>

1. Разработка мобильных приложений. – [Електронний ресурс]. – Режим доступу:

<http://pkstudio.com.ua/standarty-kachestva-produkcii/>

1. Мобильные игры сегодня и завтра. – [Електронний ресурс]. – Режим доступу:

<https://www.facebook.com/appintopru/posts/254186671395043>

1. Мобильные игры для Андроид: лучшие представители по жанрам - [Електронний ресурс]. – Режим доступу:

<http://softdroid.net/luchshie-mobilnye-igry-dlya-android>

1. Мобільна гра Fly Happy Bird. – [Електронний ресурс]. – Режим доступу:

<https://play.google.com/store/apps/details?id=com.trifingger.flyhappybird>

1. Мобільна гра RETRY. – [Електронний ресурс]. – Режим доступу:

<https://play.google.com/store/apps/details?id=com.rovio.retry>

1. Мобільна гра Square Jump. – [Електронний ресурс]. – Режим доступу:

<https://play.google.com/store/apps/details?id=com.helusion.squarejump>

1. Єлісеєнко О.Є. Порівняльний аналіз сучасних гральних рушіїв / О.Є. Єлісеєнко – Вінниця: ВНТУ, 2016.
2. Яворович О.Ю. Unity3d як універсальний інструмент розробки ігрових додатків / О.Ю. Яворович – Вінниця: ВНТУ, 2016.
3. C#. Матеріал з Вікіпедії — вільної енциклопедії. – [Електронний ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/C_Sharp>
4. Johnson B. Professional Visual Studio 2012. - John Wiley & Sons, 2013. – 1104 p.
5. Зиборов В. В. MS Visual C++ 2010 в среде .NET. Библиотека программиста. – СПб: Питер, 2012. – 320с.: ил.
6. Новые возможности Visual Studio 2015. – [Електронний ресурс]. – Режим доступу: <https://msdn.microsoft.com/ru-ru/library/bb386063.aspx>
7. Возможности среды программирования MonoDevelop. – [Електронний ресурс]. – Режим доступу: <https://www.ibm.com/developerworks/ru/library/l-Mono_2/>
8. Системне програмне забезпечення. Користувальницький інтерфейс. – [Електронний ресурс]. – Режим доступу:

<http://ua-referat.com/Користувальницький_інтерфейс>

1. Програмування по-українськи. Об'єктно-орієнтоване програмування. – [Електронний ресурс]. – Режим доступу:

<http://programming.in.ua/programming/basisprogramming/25-oop.html>

1. Прата С. Язык программирования С++. Лекции и упражнения, 6-е изд.: Пер. с англ. – М.: ООО “И.Д. Вильямс”, 2012. – 1248 с.
2. Хокинг Джон. Unity в действии. Мультиплатформенная разработка на C# / Пер. с англ. И. Рузмайкиной. – СПб.: Питер, 2016. – 336с.
3. Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя. 2-е изд.: Пер. с англ. Мухин Н. – М.: ДМК Пресс, 2006. – 496 с.: ил.
4. Борьба с 2D-физикой в Unity на примере бесконечной игры. Хабрахабр. – [Електронний ресурс]. – Режим доступу:

<https://habrahabr.ru/post/248391/>

1. Will Goldstone. Unity 3.x. Game Development Essentials. Game development with C# and Javascript. – Birmingham: Packt publishing, 2011. – 488p.
2. Краткий обзор нового Unity UI с примерами организации интерфейса. Хабрахабр. – [Електронний ресурс]. – Режим доступу:

<https://habrahabr.ru/post/246605/>

1. Image. Unity documentation. – [Електронний ресурс]. – Режим доступу: <http://docs.unity3d.com/Manual/script-Image.html>
2. Поняття про тестування програмного забезпечення. – [Електронний ресурс]. – Режим доступу: <http://helpiks.org/5-101392.html>
3. Степанченко И.В. Методы тестирования программного обеспечения: Учебное пособие / ВолгГТУ, Волгоград, 2006. – 74с.

ДОДАТКИ

# ДОДАТОК А Технічне завдання

Технічне завдання на бакалаврську дипломну роботу

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ

д.т.н., професор \_\_\_\_\_\_ Пєтух А.М.

"\_\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2016 року

Технічне завдання

на бакалаврську роботу за напрямом підготовки

6.050103 – Програмна інженерія

Керівник бакалаврської роботи:

\_\_\_\_\_\_\_\_\_\_\_\_ к.т.н., доц. Катєльніков Д. І.

"\_\_\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2016 р.

Виконaв:

\_\_\_\_\_ студент гр.1ПІ-12б Яворович О. Ю.

"\_\_\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2016 р.

Вінниця ВНТУ 2016

1. Галузь застосування – ігрова система.

2. Підстава для розробки.

Підставою для розробки бакалаврської дипломної роботи є рішення засідання кафедри програмного забезпечення (протокол №\_\_ від «\_\_»\_\_\_\_\_\_\_\_\_ 2016 року).

3. Мета та призначення розробки.

Мета виконання бакалаврської дипломної роботи – процес подолання нескінченності в scroller-грі. Призначення роботи – розробка ігрового процесу scroller-гри з нескінченними рівнями.

4. Джерела розробки:

* завдання на бакалаврську дипломної роботу;

5. Технічні вимоги:

5.1 Вимоги до середовища розробки:

* операційна система – Windows 7/8/10;
* ігровий двигун – Unity3D;
* середовище розробки – Visual Studio 2015;
* мова програмування – C#.

5.2 Мінімальні вимоги до програмного забезпечення:

* операційна система – Android 4.0 і вище;
* процесор – ARM Cortex-A9 1 ГГц Dual Core і вище;
* оперативна пам’ять – 1ГБ і вище;
* графічний процесор – PowerVR SGX531 Ultra і вище.

6. Конструктивні вимоги.

Графічна та текстова документація повинна відповідати всім діючим стандартам України.

7. Перелік технічної документації, що пред’являється по закінченню робіт:

* пояснювальна записка до БДР;
* лістинг програми.

8. Стадії та етапи розробки:

|  |  |
| --- | --- |
| Назва етапів бакалаврської кваліфікаційної роботи | Строк виконання етапів роботи |
| Аналіз сучасного стану та обґрунтування актуальності розробки | 09.02.16 – 15.02.16 |
| Аналіз існуючих аналогів | 16.02.16 – 23.02.16 |
| Аналіз ігрових двигунів | 24.02.16 – 02.03.16 |
| Розробка структури та моделей ігрової системи “Unknown enemy” | 03.03.16 – 01.04.16 |
| Розробка алгоритму подолання нескінченності | 02.04.16 – 09.04.16 |
| Розробка алгоритму генерації рівня | 10.04.16 – 17.04.16 |
| Розробка ігрової системи “Unknown enemy” | 18.04.16 – 23.05.16 |
| Тестування ігрової системи | 24.05.16 – 07.06.16 |
|  |  |

9. Порядок контролю та прийняття.

Виконання етапів бакалаврської дипломної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту.

Корегування технічного завдання допускається з дозволу керівника бакалаврської дипломної роботи.

# ДОДАТОК Б Лістинг вихідного коду ігрової системи

# ДОДАТОК В Ілюстративний матеріал

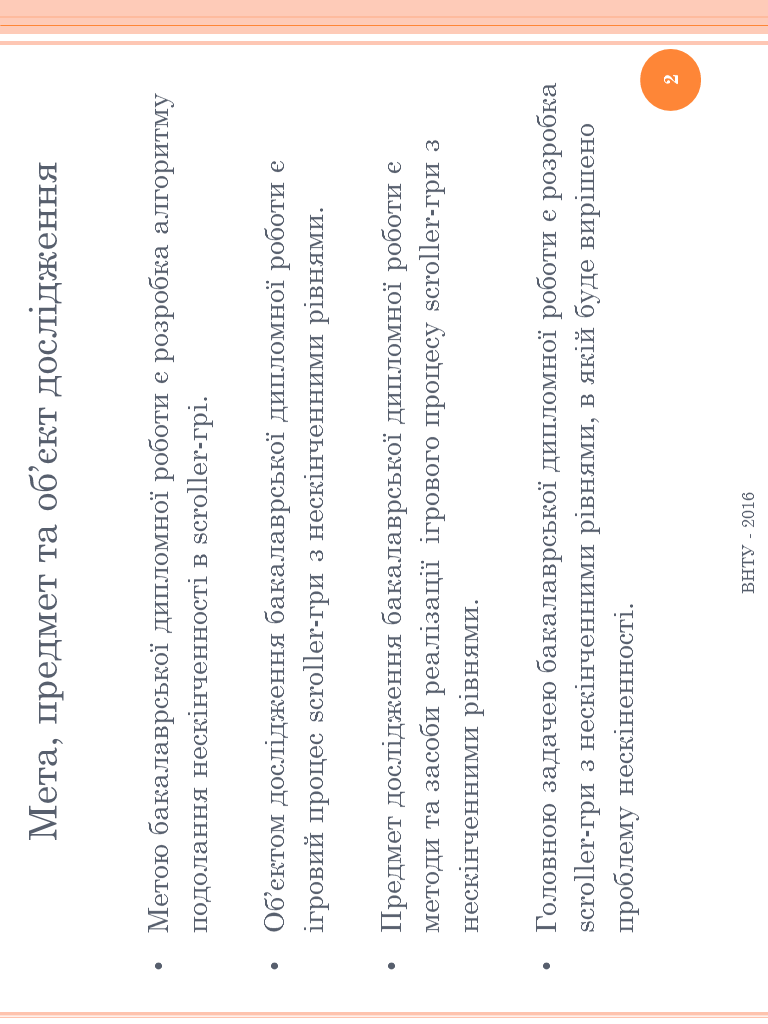
ІЛЮСТРАТИВНИЙ МАТЕРІАЛ ДО ЗАХИСТУ БАКАЛАВРСЬКОЇ РОБОТИ

|  |  |  |
| --- | --- | --- |
| Завідувач кафедри ПЗ, д.т.н., професор | \_\_\_\_\_\_\_\_\_\_\_\_ | А. М. Пєтух |
| Науковий керівник, к.т.н., доцент | \_\_\_\_\_\_\_\_\_\_\_\_ | Д. І. Катєльніков |
| Рецензент, к.т.н., доцент | \_\_\_\_\_\_\_\_\_\_\_\_ | О. К. Колесницький |
| Виконавець, студент групи 1ПІ-12б | \_\_\_\_\_\_\_\_\_\_\_\_ | О. Ю. Яворович |

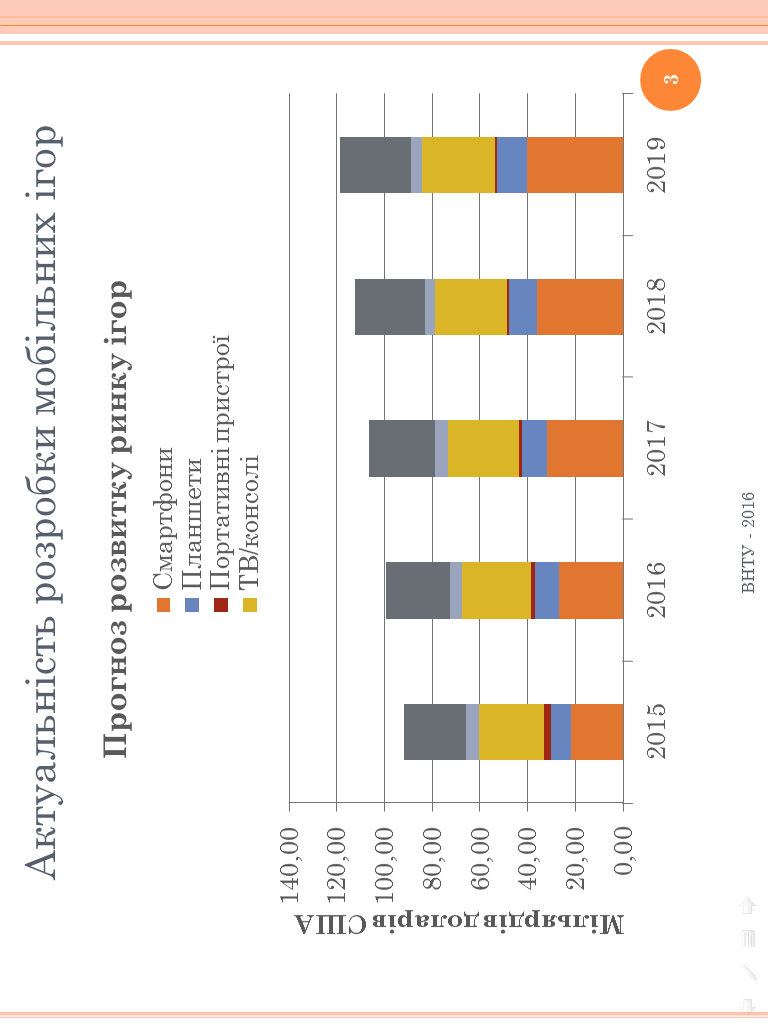
Плакат 1 - Тема, автор, науковий керівник бакалаврської дипломної роботи



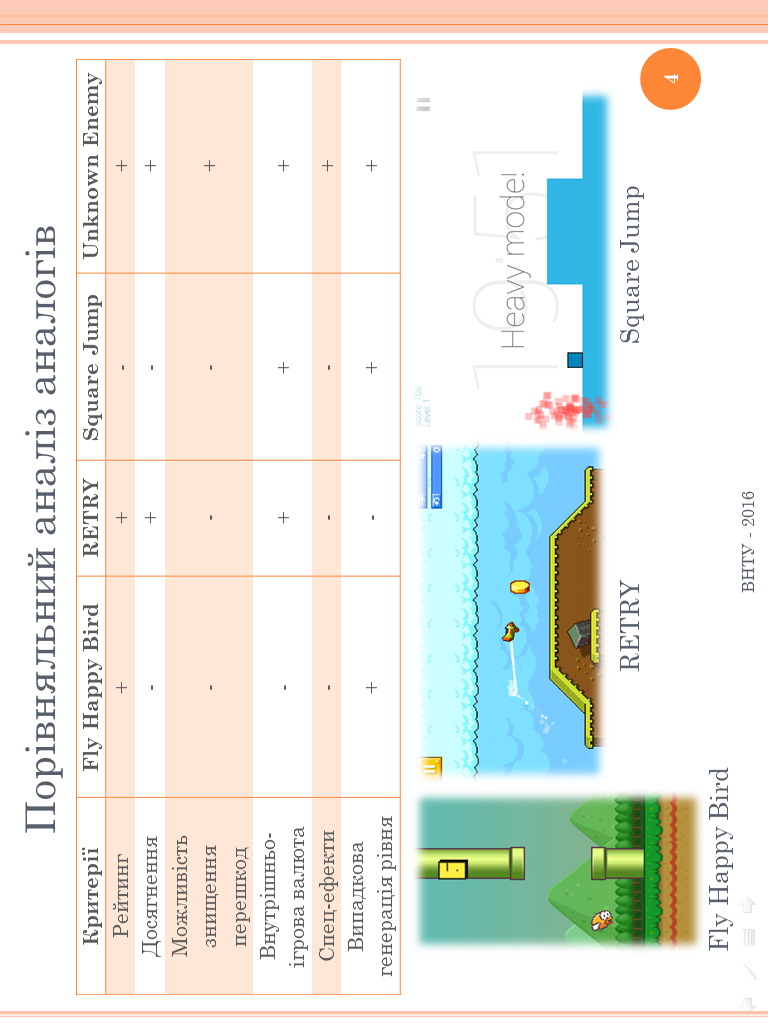
Плакат 2 - Мета, предмет та об’єкт дослідження



Плакат 3 - Актуальність розробки мобільних ігор



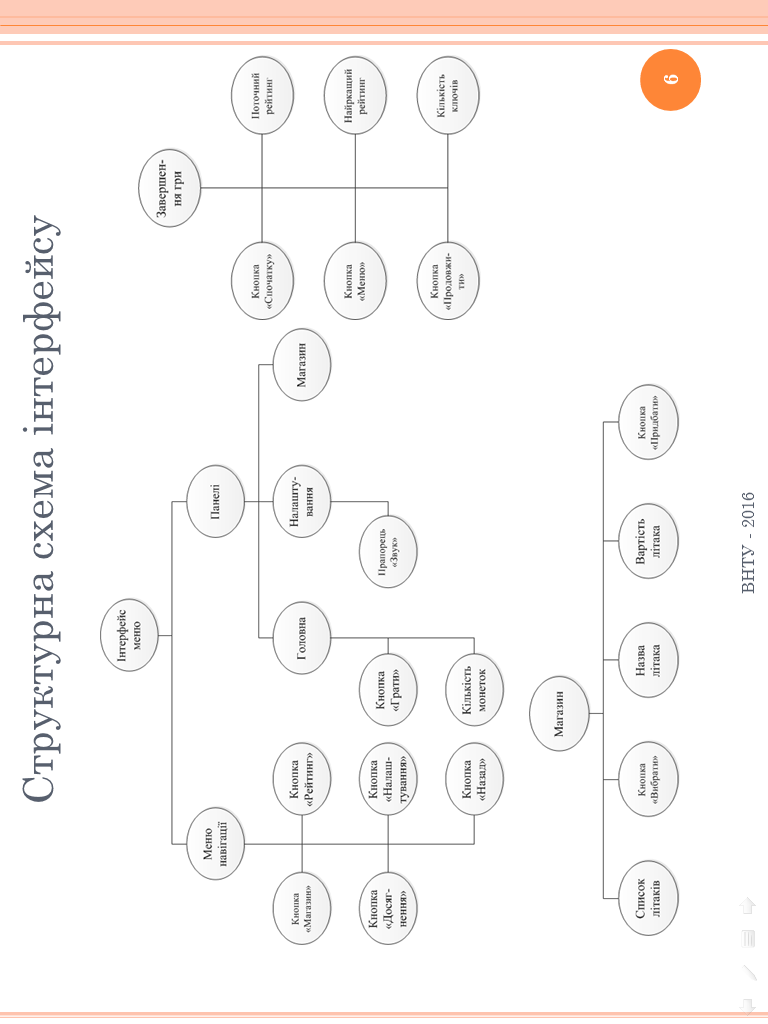
Плакат 4 - Порівняльний аналіз аналогів



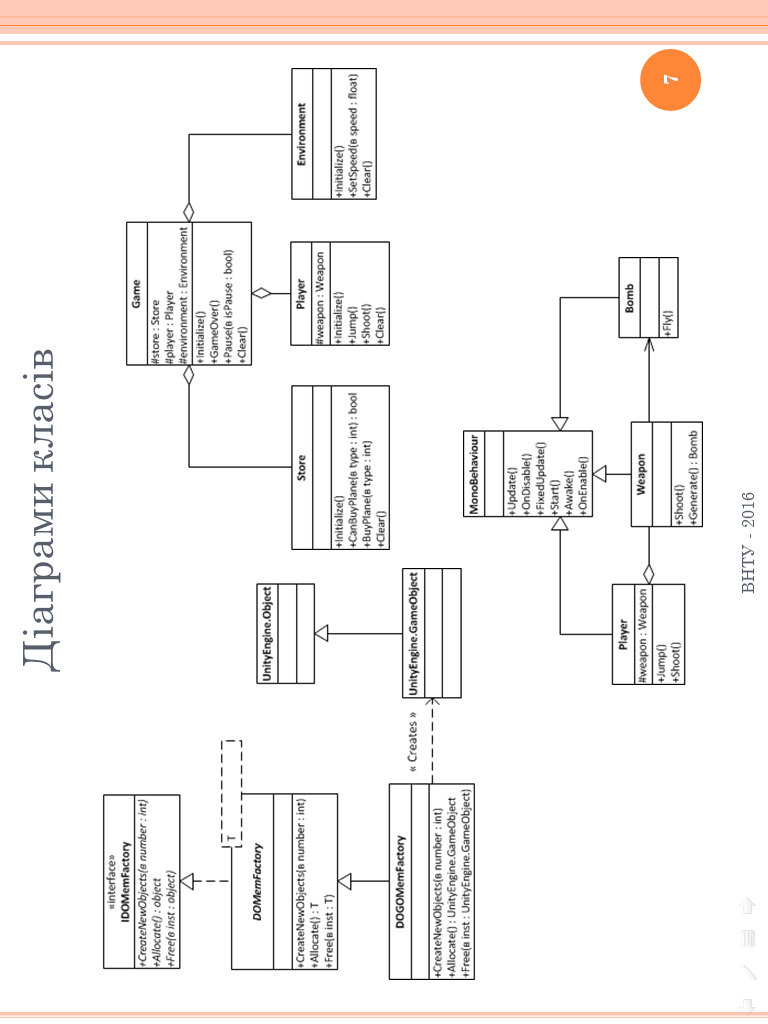
Плакат 5 - Аналіз ігрових двигунів



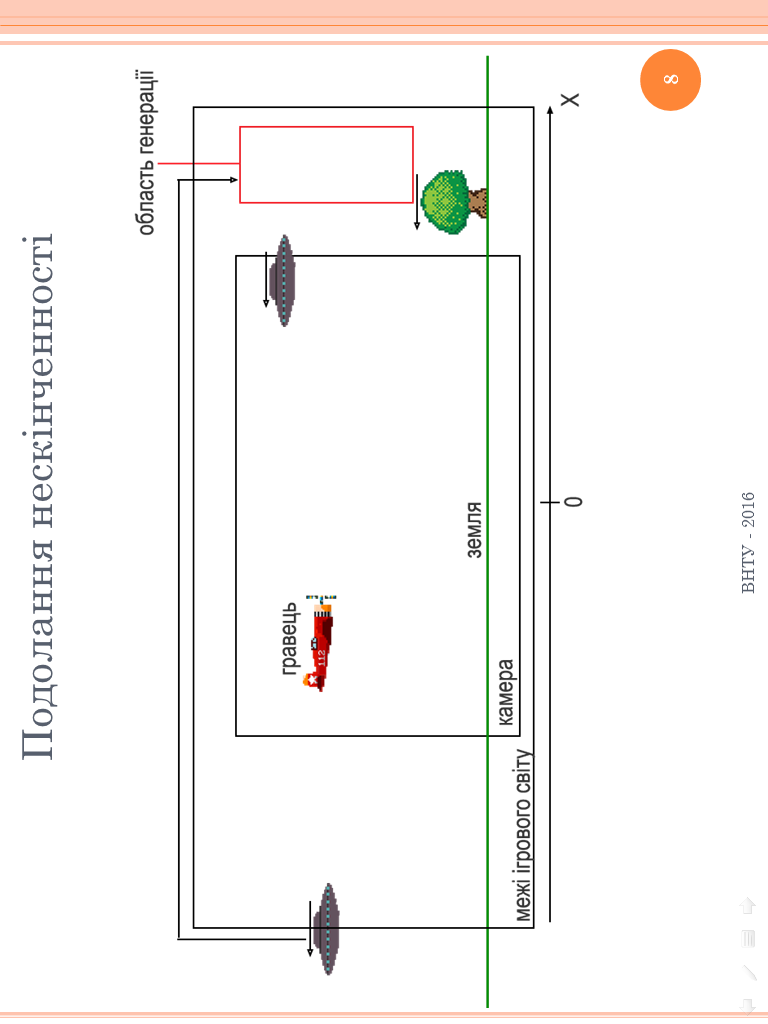
Плакат 6 - Структурна схема інтерфейсу



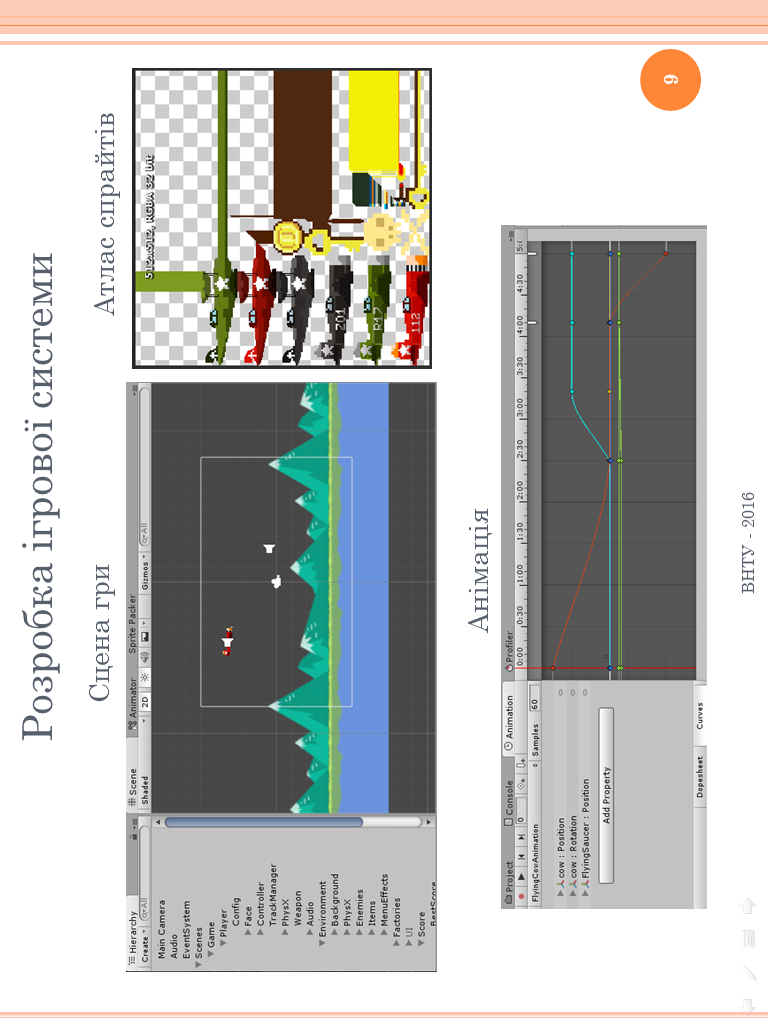
Плакат 7 - Діаграми класів



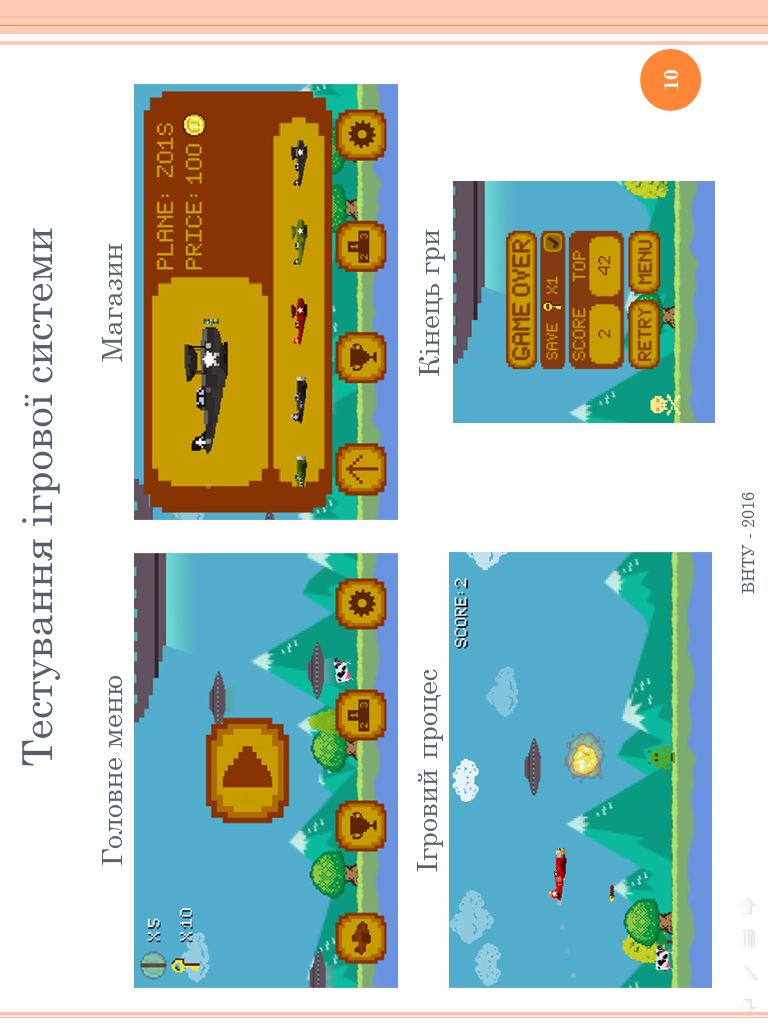
Плакат 8 - Подолання нескінченності



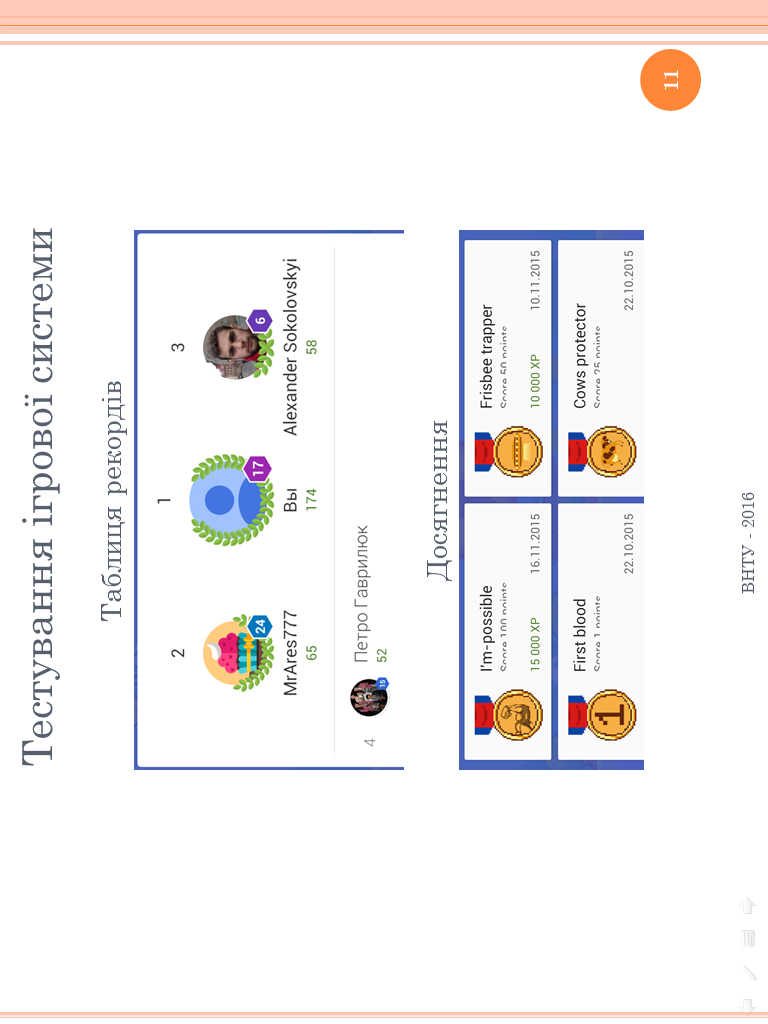
Плакат 9 - Розробка ігрової системи



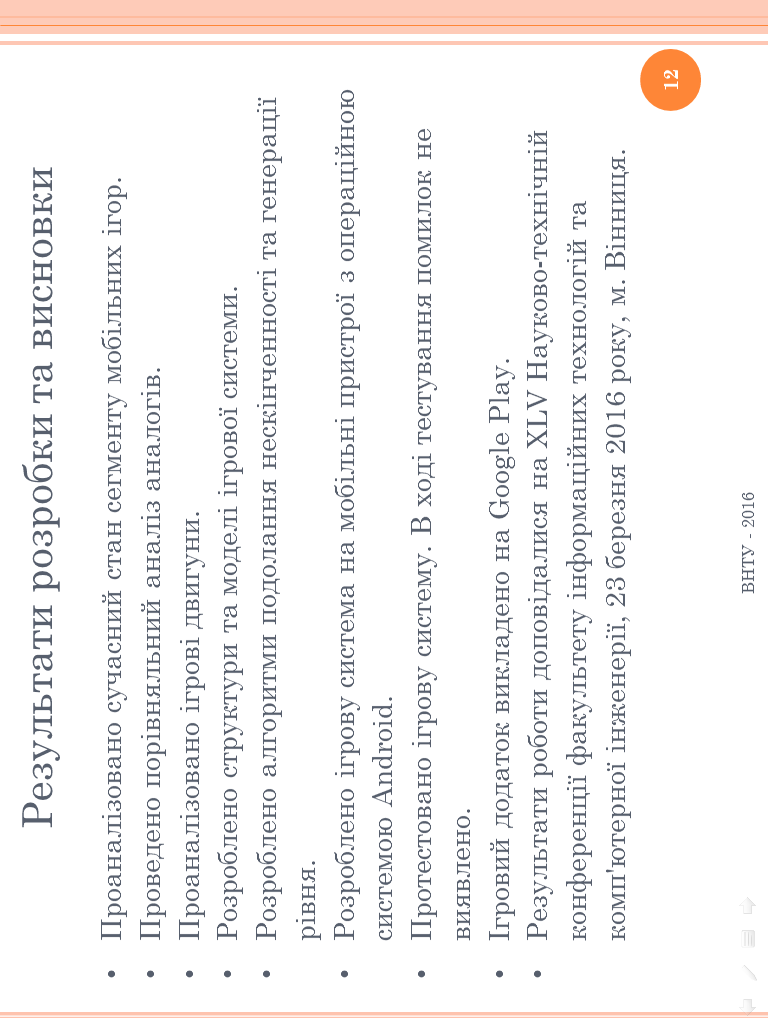
Плакат 10 - Тестування ігрової системи



Плакат 11 - Тестування ігрової системи



Плакат 12 - Результати розробки та висновки



Плакат 13 - Дякую за увагу

