МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Кафедра комп’ютерних наук

**КУРСОВИЙ ПРОЕКТ**

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

з дисципліни: “Технології програмування”

на тему:

Розробка гри на Unity на тему 2D Top-down Shooter

студента ІІІ курсу групи КН-19-2

спеціальності 122 “Комп’ютерні науки”

Копилова Владислава Руслановича

(прізвище, ім’я та по-батькові)

Керівник: В.Л. Левківський

Дата захисту: “29” грудня 2021р.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Г.В. Марчук

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.Л. Левківський

(підпис) (прізвище та ініціали)

Житомир – 2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Факультет інформаційно-комп’ютерних технологій

Кафедра комп’ютерних наук

Освітній рівень: бакалавр

Спеціальність 122 «Комп’ютерні науки»

“ЗАТВЕРДЖУЮ”

Зав. кафедри КН

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

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_ 20\_\_р.

ЗАВДАННЯ

НА КУРСОВИЙ ПРОЕКТ СТУДЕНТУ

Копилову Владиславу Руслановичу

1. Тема роботи: Розробка гри на Unity на тему 2D Top-down Shooter

керівник роботи: Левківський Віталій Леонідович

2. Строк подання студентом: “ ” травня 2021р.

3. Вихідні дані до роботи: Розробити гру на Unity

4. Зміст розрахунково-пояснювальної записки(перелік питань, які підлягають розробці)

- аналіз методів та засобів вирішення задачі

- проектування та розробка програмного додатку

- опис роботи з програмним додатком та його тестування

5. Перелік графічного матеріалу(з точним зазначенням обов’язкових креслень)

1. Презентація до КП.

2. Посилання на репозиторій:https://gitlab.com/Tamskray/course\_work\_unity.git

6. Консультанти розділів проекту (роботи)

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посади консультанта | Підпис, дата | |
| завдання  видав | завдання прийняв |
| 1, 2 ,3 | Левківський В.Л. |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

7. Дата видачі завдання “ 25 ” вересня 2021 р

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назва етапів курсової роботи | Строк виконання етапів роботи | Примітки |
| 1 | Постановка задачі | 25.09.21 |  |
| 2 | Пошук, огляд та аналіз аналогічних розробок | 25.11.21 |  |
| 3 | Формулювання технічного завдання | 01.12.21 |  |
| 4 | Опрацювання літературних джерел | 04.12.21 |  |
| 5 | Проектування структури | 18.12.21-21.12.21 |  |
| 6 | Написання програмного коду | 21.12.21-27.12.21 |  |
| 7 | Налагодження | 25.12.21-27.12.21 |  |
| 8 | Написання пояснювальної записки | 27.12.21-28.12.21 |  |
| 9 | Захист | 29.12.21 |  |

**КАЛЕНДАРНИЙ ПЛАН**

**Студент** \_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.Р. Копилов

(підпис) (прізвище та ініціали)

**Керівник роботи** \_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.Л. Левківський

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до курсового проекту на тему «Розробка гри на Unity на тему 2D Top-down Shooter» складається з переліку умовних скорочень, вступу, трьох розділів, висновків, списку використаної літератури та додатку.

Текстова частина викладена на **28** сторінках друкованого тексту.

Пояснювальна записка має **27** сторінки додатків. Список використаних джерел містить **10** найменувань і займає **1** сторінку. В роботі наведено **23** рисунків. Загальний обсяг роботи – **35** сторінок.

У першому розділі було проаналізовано потреби та визначено альтернативні програмні продукти.

У другому розділі проведено проектування гри та розробка алгоритмів и скриптів.

У третьому розділі проведено опис можливостей та тестування продукту.

Висновок містить в собі результати виконаної роботи створення гри на платформі Unity.

У додатку представлений лістинг розробленого програмного продукту.

**ЗМІСТ**

[**ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ** 6](#_Toc91550754)

[**ВСТУП** 7](#_Toc91550755)

[**РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ** 8](#_Toc91550756)

[**1.1** **Аналіз існуючого програмного забезпечення за тематикою курсового проекту (роботи).** 8](#_Toc91550757)

[**1.2** **Концепт-документ (Concept Document)** 12](#_Toc91550758)

[**Висновки до першого розділу** 12](#_Toc91550759)

[**РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ** 13](#_Toc91550760)

[**2.1 Дизайн-документ (Game Design)** 13](#_Toc91550761)

[**2.2 Розробка архітектури ігрового додатку** 15](#_Toc91550762)

[**2.3 Опис (графічне представлення) основних алгоритмів** 17](#_Toc91550763)

[**2.4 Основні етапи реалізації гри** 18](#_Toc91550764)

[**Висновки до другого розділу** 21](#_Toc91550765)

[**РОЗДІЛ 3. ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ** 22](#_Toc91550766)

[**3.1 Опис роботи з програмним додатком (Опис інтерфейсу)** 22](#_Toc91550767)

[**3.2 Тестування роботи програмного забезпечення** 25](#_Toc91550768)

[**Висновки до третього розділу** 26](#_Toc91550769)

[**ВИСНОВКИ** 27](#_Toc91550770)

[**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ** 28](#_Toc91550771)

[**ДОДАТОК** 29](#_Toc91550772)

# **ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ**

ХП — кількість здоров’я гравця

ПК — персональний комп’ютер

WASD — відповідно клавіші «W», «A», «S», «D»

ПЗ - програмне забезпечення

НІП – не ігровий персонаж

STG – жанр гри, абревіатурою позначаються всі ігри про стрілянину

RPG - жанр комп'ютерних та відеоігор, де основою ігрового процесу є відігрівання певної ролі

# **ВСТУП**

У цій курсовій роботі буде наведено процес створення гри, розроблення 2D TopDown Shooter з елементами roguelike на платформі Unity.

TopDown Shooter або Shoot 'em up (з англ. - «перестріляй їх усіх»), також shmup і STG - різновид жанру шутер комп'ютерних ігор. У цих іграх ігровий персонаж рухається у чотирьох напрямках та розстрілює безліч ворогів, ухиляючись одночасно від перешкод. У питанні, які саме ігрові елементи найбільш характерні для жанру, загальновизнана думка відсутня. Для деяких рецензентів такими є наявність у грі космічного апарату та певних типів рухів персонажа, інші вважають доречним ширше трактування визначення, включаючи жанр гри з персонажами, що переміщаються пішки, і з різною графічною перспективою. Ігри в жанрі shoot 'em up відомі вимогливістю до реакції гравця та необхідністю запам'ятовувати ігрові рівні та послідовності ворожих атак.

Roguelikeабо rogue-подібні ігри — це піджанр рольових відеоігор, визначними особливостями якого є випадкове, процедурне створення рівнів, покроковий ігровий процес, плиткова (тайлова) або ASCII-графіка та перманентна смерть персонажа у випадку поразки.

***Актуальність теми:***  ця тема для курсового проекту є актуальною, адже ігри з кожним роком стають все більш популярнішими, а геймдев розвивається з блискавичною швидкістю. На виході ми отримаємо інді гру, що означає, що таку гру може розробити меленька група людей чи навіть одна людина, що є досить важливим навичком у сучасному геймдеві.

***Мета:*** метою дослідження є цікава гра, в яку можна було б просто пограти чи взяти за основу для більш великого проекту.

***Об’єктом дослідження*** – є гра, її створенння на Unity, з використанням мови програмування C#.

***Предметом дослідження*** – вивчення нових можливостей роботи на платформі Unity та різних технологій програмування.

# **РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ**

## **Аналіз існуючого програмного забезпечення за тематикою курсового проекту (роботи).**

Для пошуку та аналізу аналогічних ігр, нам спочатку потрібно виділити основні критерії за якими ми можемо визначити потрібну гру для дослідження.

Було виділено такі критерії як:

* 2D
* Можливість пересування у 4 строни, вид камеру – зверху
* Присутня механіка стрільби та декілька видів зброї
* Присутня велика кількість супротивників на рівнях
* Декілька рівнів, рівень складається з кімнат
* Випадкова генерація рівнів

Отже було виділено такі приклади аналогічного програмного забезпечення:

1) *Enter the Gungeon* — відеогра в жанрі roguelike, розроблена компанією Dodge Roll і випущена команією Developer Gigital. Гра розповідає про мандрівників які спустилися в Gungeon (від gun — гармата і dungeon — підземелля) з однією метою, щоб знайти зброю, що вб'є їхнє минуле.



Рис.1.1. Enter the Gungeon

Гра належить до двох основних жанрів Roguelike(було описано вище) та Shoot 'em up.



Рис.1.2. Скриншот з гри (супротивники)

Shoot 'em up («перестріляти їх усіх») — жанр відеоігор, в яких гравець, керуючи персонажем або технічним засобом, бореться з великою кількістю ворогів за допомогою стрілянини. Процес гри часто зображений в дуже стилізованої манері. У короткому варіанті, поширеному серед фанатів жанру, має назву «шмап» (shmup). У Японії такі ігри відомі як «shooting games» («ігри зі стріляниною», також використовується абревіатура «STG»). На піку популярності вони часто називалися просто як «shooter», тобто «стрілялка» і в більшості були двовимірними. Але з появою тривимірних ігор жанр став більш збірним.

Переваги:

* Дуже динамічна, потребує швидкої реації гравця
* Має цікаву стилістику
* Дуже багато зброї, видів супротивників та босів
* Класний саундтрек

Недоліки:

* Доступна тільки на ПК
* Для когось може здатись складною

2) *The Binding of Isaac* - комп'ютерна гра у жанрі action-adventure з елементами roguelike та шутера з видом зверху. 2D-RPG з видом зверху, гра, де гравець керує Ісааком або іншими розблокованими персонажами, досліджуючи підвальні підземелля. Механіка та зовнішнє оформлення гри нагадує ранні ігри серії The Legend of Zelda, але з додаванням елемента випадковості, незворотної смерті та процедурно генерованих рівнів – механік, характерних для жанру roguelike. На кожному поверсі підземелля гравець змушений боротися із монстрами у кожній кімнаті, щоб просуватися далі. По дорозі він може збирати гроші, щоб купити спорядження в магазині, ключі, щоб відкривати скарбниці, нову зброю та різні посилення, щоб збільшити свою перевагу перед ворогами. На кожному поверсі підземелля є бос, якого потрібно вбити для отримання можливості переміститися на наступний рівень.



Рис.1.3. Скриншот з гри The Binding of Isaac



Рис.1.4. Скриншот з гри The Binding of Isaac, битва с боссом

Переваги:

* Має цікаву стилістику
* Інтригуючий сюжет
* багато видів супротивників та босів

Недоліки:

* Доступна тільки на ПК
* Для когось може здатись складною, через випадкову генерацію рівнів
* Для когось може бути висока ціна

3) *Soul Knight* - це пригодницький екшен, в якому гравці повинні досліджувати підземелля, збирати унікальну зброю та боротися з величезною кількістю супротивників. На вибір гравцям дається понад 170 одиниць озброєння, випадково створені рівні, а також різноманітні грабельні персонажі, кожен з яких має власні навички. Відноситься до жанру 2d TopDown Shooter та Roguelike



Рис.1.5. Скриншот з гри Soul Knight

Переваги:

* Багато персонажів
* Багато видів зброї, супротивників та босів
* Можна грати будь де, адже гра доступна на телефоні
* Безкоштовна

Недоліки:

* Немає порту на ПК
* Для когось може здатись складною, через випадкову генерацію рівнів

## **Концепт-документ (Concept Document)**

1. *Жанр і аудиторія:* гра належить до двох основних жанру shoot 'em up з елементами roguelike, основна аудиторія це юнаки та підлітки (від 14 до 25 років)

2. *Основні особливості гри:* bullet hell, зброя, супротивники, герой

3. *Геймплей:* проходити рівні, вбивати ворогів, знищувати босів, шукати новую зброю, досліджувати рівні підземелля, шукати НІП-ів, пройти усі рівні

4. *Сюжет:* головний герой хоче довести свою силу та пройти випробування у dungeon, на його шляху зустрінуться вороги, він навіть потрапить у інший світ де змушений буде довести свою стійкість..

5. *Платформа:* ПК, Android

## **Висновки до першого розділу**

У ході виконання першого розділу була проаналізована основна задача, засіб її вирішення. Були проаналізовані аналогічні ігри. Були визначені основні технічні завдання, яких необхідно дотримуватись при створенні своєї гри, для опимальної та зручної роботи, та реалізації потрібного функціоналу.

# **РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

## **2.1 Дизайн-документ (Game Design)**

Гра розроблена в стилі випробування в інтерактивному форматі. Тобто гравець повинен проходити рівні, щоб пройти гру.

Кінцева мета гри – пройти усі випробування, тобто усі рівні.

Гра реалізована наступним чином: існує головний персонажі (навколо якого і відбуваються усі дії) та один другорядний персонаж, який буде напрявляти головного героя та розповідати про світ.

Сцена, що описана вище, буде розроблена у стилі 2D Top-down, як і усі інші.

Управління персонажем здійснюється за допомогою клавіш WASD на клавіатурі. WS - відповідають за пересування по вертикалі, а AD – по горизонталі. Також можна користуватись стрілочками. Присутній порт на Android, тому у грі присутнє управління за допомогою стіків.

Для частини 2D-гри розроблены декілька сцена, на якій розташована земля по якій рухається персонаж та присутні стінки, за які він не може зайти.

Також, персонаж від початку має 10 ХП, яке відповідає за життя героя.

Анімація об’єктів здійснюється за допомогою «спрайтів». Основним гейм-об’єктом є головний персонаж, який має два види анімації – стан спокою та біг. Другим анімованим об’єктом є персонаж-«монстр» – супротивник, він має дільки анімацію бігу. Третій анімований персонаж – це мастер-гід, має лише анімацію спокою. Та останній це бос, має такі анімації: анімація входу до першої та другої фази, дві анімації стану спокою, дві анімаціїї стрибку та анімація смерті.



Рис.2.1. Зображення персонажу



Рис.2.2. Зображення ворога

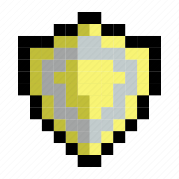
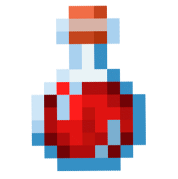


Рис.2.3. Зображення сцілюючого зілля та щита



Рис.2.4. Зображення боса

Якщо ворог або бос атакує героя, то другий втрачає життя до моменту, поки кількість ХП не буде меншою за нуль.

Персонаж може пройти перший навчальий рівень, знайшовши портал – перехід на другий. Щоб пройти другий рівень, герой повинен знайти ключ на карті та активувати наступний портал. Для пройдення третього рівня, потрібно буде перемогти боса.

Графіка скалається з пакету різних спрайтів. Пакет було завантажено з AS.

Спрайти для 2D Top-down shooter були завантаженні з: <https://assetstore.unity.com/packages/2d/environments/pixel-art-top-down-basic-187605>

Пак спрайтів для додавання «джойстик» управління: <https://assetstore.unity.com/packages/tools/input-management/joystick-pack-107631>

Інші спрайти або малюнки були зроблені самостійно.

## **2.2 Розробка архітектури ігрового додатку**

На рисунку 2.5. показані ігрові обєкти і їх компоненти.

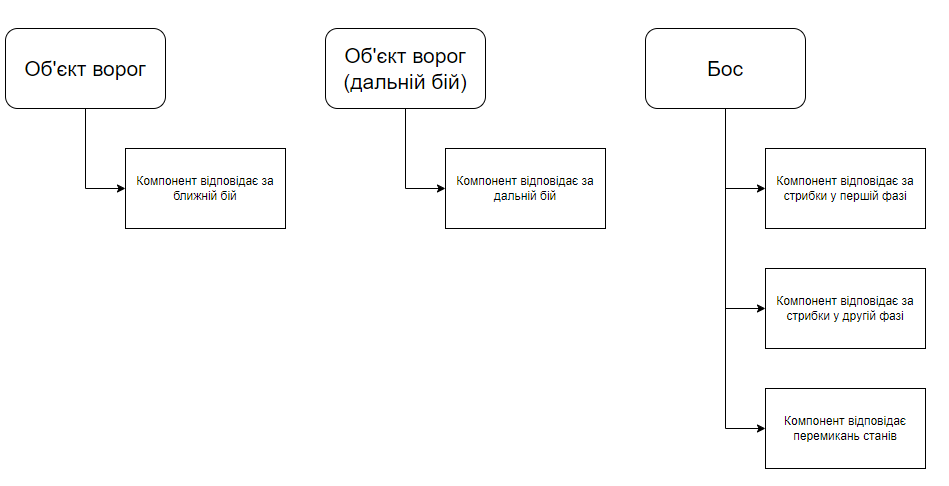


Рис.2.5. Схема ігрових об’єктів

На рисунку 2.6. представлена схема переходів між сценами ігрового додатку.

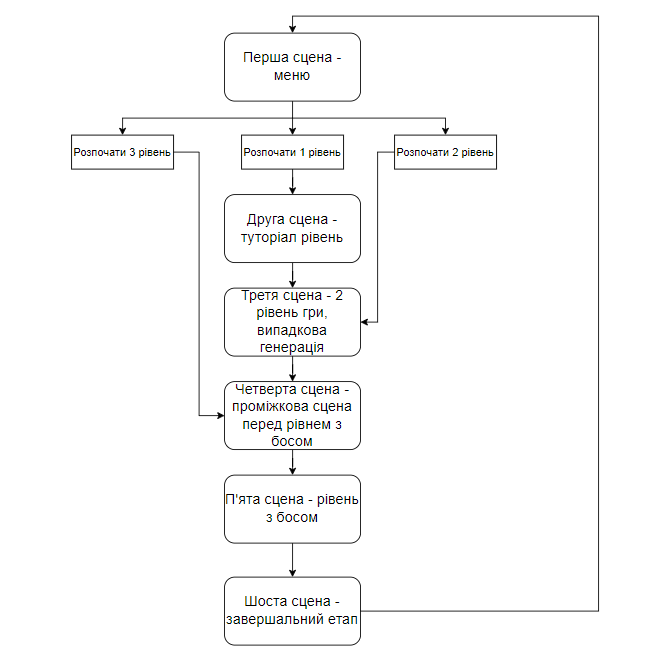


Рис.2.6. Схема сцен і переходів

Як тільки гравець обере який рівень розпочати, відкриється відповідна сцена.

Друга сцена – туторіал, навчальний рівень, який ознайомить гравця з основними ігровими механіками, ворогами, предметами. На рівні герой зустріне НІП-а, з який можна поговорити.

Третя сцена – рівень з випадковою генерацією кімнат. На рівні генерується ключ, який потрібен для відкриття портала у останній кіманаті.

Четверта сцена – проміжкова сцена, можна побалакати с НІП-ом та перейти на наступну сцену.

П’ята сцена – рівень с босом, щоб пройти на наступну сцену, потрібно його спочатку подолати

Шоста сцена – останній рівень, гравця зустріне декілька ворогів та потів НІП, який привітає з проходженням випробування.

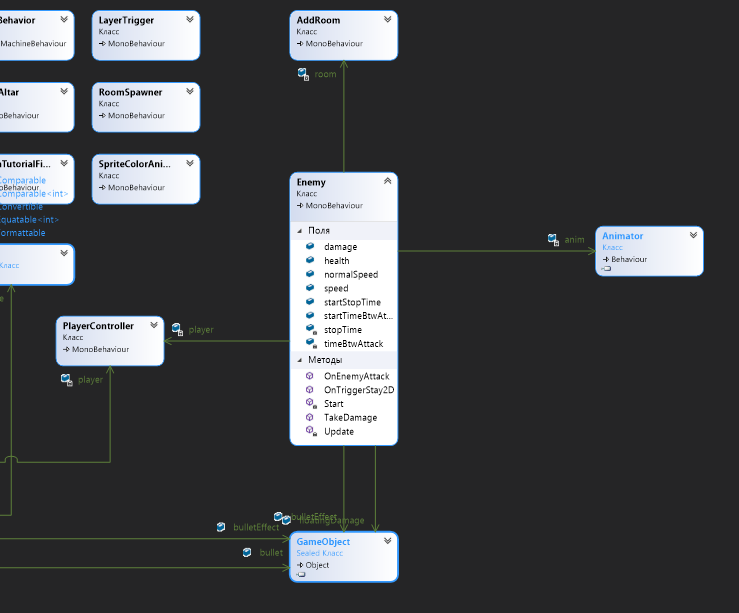


Рис.2.7. Частина діаграми класів

## **2.3 Опис (графічне представлення) основних алгоритмів**

Опишемо основні алгоритми робои анімацій, починаючи з гравця

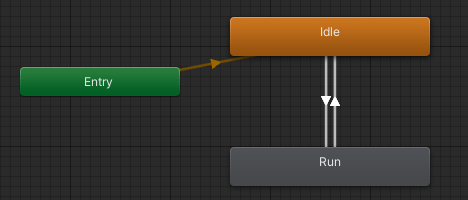


Рис.2.8. Алгоритм роботи анімаціїї гравця

Зображено 3 блоки – Entry, Run, Idle. Блок Entry означає початок анімації, тобто, при вмиканні сцени з цією анімацією, анімація разпочинає свою дію, і відбуваеться перехід до анімації Idle, що є стандартною анімацією, яка спрацюває, коли персонаж не виконує ніяких дій, а просто стоїть. Якщо гравець виконує маніпуляції над персонажем (тобто біг), то зі стану Idle відбуваєьтся переход до стану Run

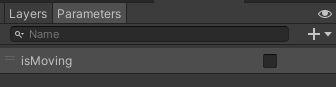


Рис.2.9. Параметр для визначення бігу гравця

За допомогою параметра isMoving ми визначаємо коли персонаж рухається, а коли стоїть на місці.

Для ворогів, НІП-а та інших об’єктів які мають анімацію, наприклад портал – в аніматорі лише два блоки Entry та одразу перехід до анімації (у ворога – це біг, у НІП-а і портала це стан спокою)

Алгоритм роботи анімації боса дещо складніший

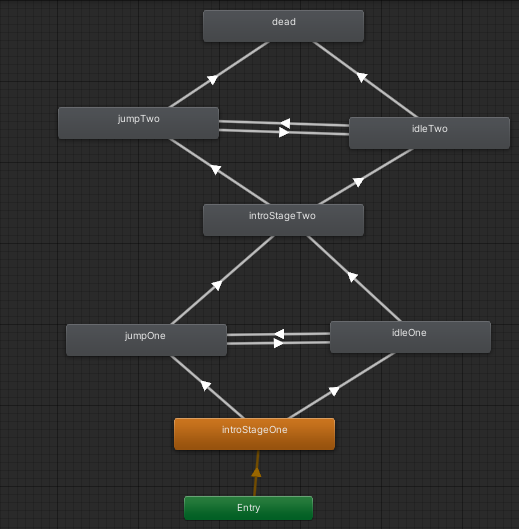


Рис.2.10. Алгоритм роботи анімаціїї боса

Має 8 блоків. Основі блоки це перша та друга фаза, з них бос може входити до відповідних станів спокою та стрибків. Зі станів спокою може переходити до стрибків. Якщо помирає, то зі стану спокою або стрибка другої фази переходить до стану смерті.

## **2.4 Основні етапи реалізації гри**

У даному пункті опишемо деякі реалізовані методи.

Наприклад пересування персонажа у методі Update(). Спочатку з вказаного перечислення ми робимо перевірку, якщо вид керування – ПК, то в змінну moveInput вказуємо горизонтальні та вертикальні шляхи. Якщо вид керування – Android, то визначаємо теж саме, але за допомогою методів joystick. Далі у змінній moveVelocity визначаємо остаточну швидкість пересування.

Лістинг коду:

if(controlType == ControlType.PC)

{

moveInput = new Vector2(Input.GetAxisRaw("Horizontal"), Input.GetAxisRaw("Vertical"));

}

else if(controlType == ControlType.Android)

{

moveInput = new Vector2(joystick.Horizontal, joystick.Vertical);

}

moveVelocity = moveInput.normalized \* moveSpeed;

Опишемо метод Update() та DestroyBullet() у скрипті Bullet. За допомогою луча рейкасту ми будемо визначето у що влучає пуля. Якщо вона влучає в колайдер, то іде перевірка у колайдер якого об’єкта вона влучила. Якщо колайдер об’єкта має тег Enemy або EnemyBasic, то викликаємо метод класу Enemy або EnemyBasic – TakeDamage, де вказуємо який урон отримає ворог. Якщо колайдер об’єкта має тег Player та ця пуля, це постіл ворога (булева змінна enemyBullet визначає чи це пуля ворога) то викликаємо метод класу PlayerController – ChangeHealth, у якій зменшуємо кількість ХП гравця. Далі пуля знищується. Якщо ж рейкаст луч не знайшов колайдер, то пуля буде рухатись у напрямку пострілку, поки не знайде колайдер.

У методі знищення пулі – ми спочатку створюємо об’єкт, це буде ефект влучання. Далі вже знищуємо пулю.

Лістинг коду:

void Update()

{

RaycastHit2D hitInfo = Physics2D.Raycast(transform.position, transform.up, distance, whatIsSolid);

if (hitInfo.collider != null)

{

if (hitInfo.collider.CompareTag("Enemy"))

{

hitInfo.collider.GetComponent<Enemy>().TakeDamage(damage);

}

if (hitInfo.collider.CompareTag("EnemyBasic"))

{

hitInfo.collider.GetComponent<EnemyBasic>().TakeDamage(damage);

}

if (hitInfo.collider.CompareTag("Player") && enemyBullet)

{

hitInfo.collider.GetComponent<PlayerController>().ChangeHealth(-damage);

}

DestroyBullet();

}

transform.Translate(Vector3.right \* Time.deltaTime \* speed);

}

public void DestroyBullet()

{

Instantiate(bulletEffect, transform.position, Quaternion.identity);

Destroy(gameObject);

}

Також опишемо метод Spawn для генерації кімнат у другому рівну. Маємо чотири строни створення кімнат. Якщо кімнати в одному з чотирьох сторін немає, то визначаємо напрямок, обираємо одну випадкову з префабів кімнат данного напрямку та створюємо її. В кінца змінюємо булеву змінну spawn на true, тим самими позначаючи, що в цьому напрямку вже є кімната.

Лістинг коду:

public void Spawn()

{

if(spawned == false)

{

if(direction == Direction.Top)

{

rand = Random.Range(0, variants.topRooms.Length);

Instantiate(variants.topRooms[rand], transform.position, variants.topRooms[rand].transform.rotation);

}

else if (direction == Direction.Bottom)

{

rand = Random.Range(0, variants.bottomRooms.Length);

Instantiate(variants.bottomRooms[rand], transform.position, variants.bottomRooms[rand].transform.rotation);

}

else if (direction == Direction.Right)

{

rand = Random.Range(0, variants.rightRooms.Length);

Instantiate(variants.rightRooms[rand], transform.position, variants.rightRooms[rand].transform.rotation);

}

else if (direction == Direction.Left)

{

rand = Random.Range(0, variants.leftRooms.Length);

Instantiate(variants.leftRooms[rand], transform.position, variants.leftRooms[rand].transform.rotation);

}

spawned = true;

}

}

## **Висновки до другого розділу**

У ході виконання другого розділу було спроектовано загальний функціонал гри, описано її. Зроблено основні алгоритми та були написані скрипти.

# **РОЗДІЛ 3. ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ**

## **3.1 Опис роботи з програмним додатком (Опис інтерфейсу)**

У даному пункті опишемо перший рівнь, який є начальним (ознайомчим) та покажемо головне меню, де маємо вибір рівнів.



Рис.3.1. Головне меню гри

На першому рівні є підказки, щодо управління, перші вороги, розмова з НІП-ом

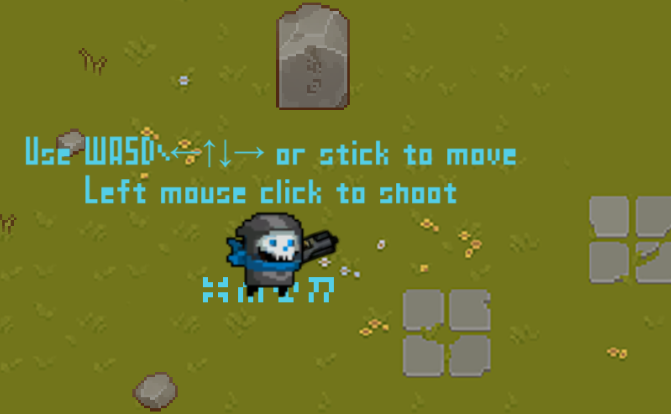


Рис.3.2. Підказки на рівні коли персонаж стоїть на руні (на місці де працює підказка)



Рис.3.3. Розмова з НІП-ом

При зустічи з ворогами ми можемо їх стріляти.

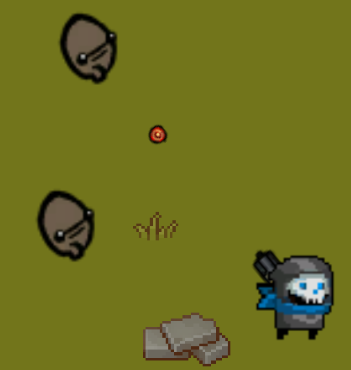
 

Рис.3.4. Срільба у ворога, еффект влучання

На карті можуть зустрічатися зілля сцілювання та щити

Рис.3.5. Зілля сцілення, його еффект

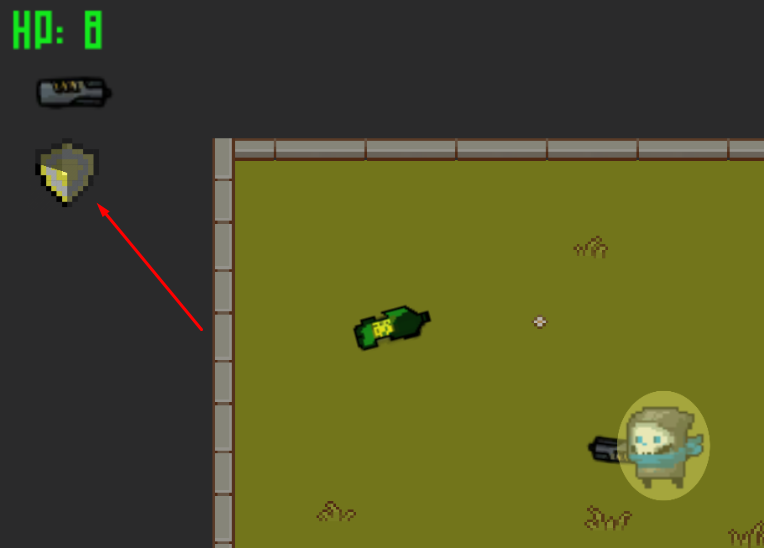
 

Рис.3.6. Щит та його таймер дії

Також можемо знайти другу зброю та взяти її



Рис.3.7. Інша зброя

Ми можемо отримувати урон від ворогів, якщо ХП буде менше нуля, то рівень почнется спочатку

Рис.3.8. Бій з ворогами та їх атака, втрата ХП героя

## **3.2 Тестування роботи програмного забезпечення**

Проведемо тестування гри

|  |  |  |  |
| --- | --- | --- | --- |
| **Сцена** | **Вхідні дані** | **Очікувані дані** | **Спостережені дані** |
| Головне меню | Натиск на кнопку 1-3 рівень | Перехід на відповідну сцену | Перехід на відповідну сцену |
| Усі рівні | Натиск на клавіши «A» та «D» | Пересування по горизонталі | Пересування по горизонталі |
| Натиск на клавіши «W» та «S» | Пересування по вертикалі | Пересування по вертикалі |
| Натиск на ліву кнопку миші | Постріл | Постріл |
| Натиск на клавішу Q | Зміна зброї (якщо гравець має дві зброї) | Зміна зброї (якщо гравець має дві зброї) |
| Підбирання зілля сцілення | Додавання певної кількості ХП | Додавання певної кількості ХП |
| Підбирання щита | Накладання щита на героя, неможливість йому нанести урон | Накладання щита на героя, неможливість йому нанести урон |
| Вхід у портал | Перехід на наступну локацію | Перехід на наступну локацію |
| Влучення пулею ворога у гравця або атака ворога ближнього бою | Втрата ХП гравця | Втрата ХП гравця |
| Натиснення кнопки початку діалогу з НІП-ом | Діалог з НІП-ом | Діалог з НІП-ом |
| 2 рівень | Підбирання ключа | Додавання ключа до предметів гравця | Додавання ключа до предметів гравця |
| Натиск на клавішу E | Якщо є ключ та гравець стоїть біля двері, то відкриття дверей | Якщо є ключ та гравець стоїть біля двері, то відкриття дверей |

При проведенні тестування гри – помилок не було знайдено

## **Висновки до третього розділу**

В ході виконання третього розділу було протестовано гру. Були протестовані всі можливості гри та ігровий процес. Функціонал був розписаний та продемонстрований на відповідних рисунках. Під час роботи різного роду проблем не виникало.

## **ВИСНОВКИ**

В першому розділі курсового проекту було проаналізовано аналогічні ігри та було створено концепт-документ гри.

В другому розділі курсового проекту було проведено проектування та розробку гри. Було створено дизайн-документ, створено основні скрипти та графічно представлено алгоритми.

В третьому розділі курсового проекту було описано роботу з програмгим додатком, було описано його інтерфейс та можливості. Було проведено тестування роботи гри.

В результаті виконання курсового проекту отримано гру, яка повністю відповідає запланованому функціоналу на початку проектування.

# **СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Сайт Metanit, даний сайт присвячений різним мовам і технологій програмування, комп'ютерів, мобільних платформ і ІТ-технологій. Режим доступу до ресурсу: <https://metanit.com/>
2. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения / Роберт Мартин. – СПб: Питер, 2019. – 351 с.
3. Fandom вікі по грі Enter the Gungeon. Режим доступу: <https://enter-the-gungeon.fandom.com/ru/wiki/Enter_the_Gungeon_%D0%92%D0%B8%D0%BA%D0%B8>
4. Learn ztu. Курс Технології програмування – Режим доступу: https://learn.ztu.edu.ua/course/view.php?id=4167
5. Unity. Documentation Version: 5.3 (switch to 5.4b) [Електронний ресурс] / Unity. – 2015. – Режим доступу до ресурсу: <https://docs.unity3d.com/ru/530/Manual/AnimationGlossary.html>.
6. Unity. Documentation Version: 2020.3 (LTS) [Електронний ресурс] / Unity. – 2017. – Режим доступу до ресурсу: <https://docs.unity3d.com/Manual/index.html>.
7. YouTube канал Blackthornprod присвячений створеню ігр – Режим доступу: https://www.youtube.com/channel/UC9Z1XWw1kmnvOOFsj6Bzy2g
8. YouTube канал maxter присвячений створеню ігр на Unity та різної ігрвої тематики – Режим доступу: https://www.youtube.com/c/Maxters/videos
9. YouTube канал Brackeys присвячений створеню ігр на Unity та програмуванню на мові C# та інших – Режим доступу: https://www.youtube.com/channel/UCYbK\_tjZ2OrIZFBvU6CCMiA
10. YouTube канал Six Dotприсвячений створеню різних ігр на Unity – Режим доступу: https://www.youtube.com/user/IxxHATExxBUFFERING/videos

# **ДОДАТОК**

**PlayerController**

Лістинг коду:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.SceneManagement;

public class PlayerController : MonoBehaviour

{

private Rigidbody2D rb;

[Header("Controls")]

public Joystick joystick;

public ControlType controlType;

[SerializeField] float moveSpeed;

public enum ControlType { PC, Android }

Animator anim;

private Vector2 moveInput;

private Vector2 moveVelocity;

public static bool facingRight = true;

private bool keyButtonPushed;

[Header("Health")]

public float health;

public Text healthDisplay;

public GameObject potionEffect;

[Header("Shield")]

public GameObject shield;

public Shield shieldTimer;

public GameObject shieldEffect;

[Header("Weapon")]

public List<GameObject> unlockedWeapons;

public GameObject[] allWeapons;

public Image weaponIcone;

[Header("Key")]

public GameObject keyIcon;

public GameObject wallEffect;

[Header("Tips")]

public Text movementTip;

public Text buffsTip;

public float timerStart = 10;

private bool messageShown;

void Start()

{

rb = GetComponent<Rigidbody2D>();

anim = GetComponentInChildren<Animator>();

if(controlType == ControlType.PC)

{

joystick.gameObject.SetActive(false);

}

}

void Update()

{

if(messageShown)

timerStart -= Time.deltaTime;

if (timerStart < 0)

{

movementTip.gameObject.SetActive(false);

buffsTip.gameObject.SetActive(false);

messageShown = false;

timerStart = 3;

}

if(controlType == ControlType.PC)

{

moveInput = new Vector2(Input.GetAxisRaw("Horizontal"), Input.GetAxisRaw("Vertical"));

}

else if(controlType == ControlType.Android)

{

moveInput = new Vector2(joystick.Horizontal, joystick.Vertical);

}

moveVelocity = moveInput.normalized \* moveSpeed;

if (!facingRight && moveInput.x > 0)

{

Flip();

}

else if (facingRight && moveInput.x < 0)

{

Flip();

}

if(health <= 0)

{

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);

facingRight = true;

}

if(Input.GetKeyDown(KeyCode.Q))

{

SwitchWeapon();

}

if (Input.GetKeyDown(KeyCode.E))

{

OnKeyButtonDown();

}

}

private void FixedUpdate()

{

rb.MovePosition(rb.position + moveVelocity \* Time.fixedDeltaTime);

anim.SetBool("isMoving", (Mathf.Abs(moveInput.x) > 0 || Mathf.Abs(moveInput.y) > 0));

}

void Flip()

{

facingRight = !facingRight;

Vector3 Scaler = transform.localScale;

Scaler.x \*= -1;

transform.localScale = Scaler;

}

private void OnTriggerEnter2D(Collider2D other)

{

if(other.CompareTag("Potion"))

{

ChangeHealth(5);

Instantiate(potionEffect, other.transform.position, Quaternion.identity);

Destroy(other.gameObject);

}

else if(other.CompareTag("Shield"))

{

if(!shield.activeInHierarchy)

{

shield.SetActive(true);

shieldTimer.gameObject.SetActive(true);

shieldTimer.isCoolDown = true;

Instantiate(shieldEffect, other.transform.position, Quaternion.identity);

Destroy(other.gameObject);

}

else

{

shieldTimer.ResetTimer();

Instantiate(shieldEffect, other.transform.position, Quaternion.identity);

Destroy(other.gameObject);

}

}

else if(other.CompareTag("Weapon"))

{

for (int i = 0; i < allWeapons.Length; i++)

{

if(other.name == allWeapons[i].name)

{

unlockedWeapons.Add(allWeapons[i]);

}

}

SwitchWeapon();

Destroy(other.gameObject);

}

else if (other.CompareTag("Tips"))

{

if(timerStart > 0)

{

movementTip.gameObject.SetActive(true);

messageShown = true;

}

}

else if (other.CompareTag("Tip2"))

{

if (timerStart > 0)

{

buffsTip.gameObject.SetActive(true);

messageShown = true;

}

}

else if (other.CompareTag("Key"))

{

keyIcon.SetActive(true);

Destroy(other.gameObject);

}

}

public void OnKeyButtonDown()

{

keyButtonPushed = !keyButtonPushed;

}

private void OnTriggerStay2D(Collider2D other)

{

if(other.CompareTag("Door") && keyButtonPushed && keyIcon.activeInHierarchy)

{

Instantiate(wallEffect, other.transform.position, Quaternion.identity);

keyIcon.SetActive(false);

other.gameObject.SetActive(false);

keyButtonPushed = false;

}

}

public void ChangeHealth(int healthValue)

{

if(!shield.activeInHierarchy || shield.activeInHierarchy && healthValue > 0)

{

health += healthValue;

healthDisplay.text = "HP: " + health;

}

else if(shield.activeInHierarchy && healthValue < 0)

{

shieldTimer.ReduceTimer(healthValue);

}

}

public void SwitchWeapon()

{

for(int i = 0; i < unlockedWeapons.Count; i++)

{

if(unlockedWeapons[i].activeInHierarchy)

{

unlockedWeapons[i].SetActive(false);

if (i != 0)

{

unlockedWeapons[i - 1].SetActive(true);

weaponIcone.sprite = unlockedWeapons[i - 1].GetComponent<SpriteRenderer>().sprite;

}

else

{

unlockedWeapons[unlockedWeapons.Count - 1].SetActive(true);

weaponIcone.sprite = unlockedWeapons[unlockedWeapons.Count - 1].GetComponent<SpriteRenderer>().sprite;

}

weaponIcone.SetNativeSize();

break;

}

}

}

}

**GunController**

Лістинг коду:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class GunController : MonoBehaviour

{

public GunType gunType;

SpriteRenderer sprite;

private PlayerController player;

public float offset;

public GameObject bullet;

public Transform shotPoint;

private float rotZ;

public enum GunType {Default, Enemy};

private float timeBtwShots;

public float startTimeBtwShots;

private Vector3 difference;

public Joystick joystick;

void Start()

{

player = GameObject.FindGameObjectWithTag("Player").GetComponent<PlayerController>();

sprite = GetComponent<SpriteRenderer>();

if(player.controlType == PlayerController.ControlType.PC)

{

joystick.gameObject.SetActive(false);

}

}

void Update()

{

if(gunType == GunType.Default)

{

if (PlayerController.facingRight)

{

sprite.flipX = false;

}

else if (!PlayerController.facingRight)

{

sprite.flipX = true;

}

//

if (player.controlType == PlayerController.ControlType.PC)

{

Vector3 difference = Camera.main.ScreenToWorldPoint(Input.mousePosition) - transform.position;

rotZ = Mathf.Atan2(difference.y, difference.x) \* Mathf.Rad2Deg;

}

else if (player.controlType == PlayerController.ControlType.Android && Mathf.Abs(joystick.Horizontal) > 0.3f || Mathf.Abs(joystick.Vertical) > 0.3f)

{

rotZ = Mathf.Atan2(joystick.Vertical, joystick.Horizontal) \* Mathf.Rad2Deg;

}

}

else if(gunType == GunType.Enemy)

{

Vector3 difference = player.transform.position - transform.position;

rotZ = Mathf.Atan2(difference.y, difference.x) \* Mathf.Rad2Deg;

}

transform.rotation = Quaternion.Euler(0f, 0f, rotZ + offset);

if (timeBtwShots <= 0)

{

if (Input.GetMouseButton(0) && player.controlType == PlayerController.ControlType.PC || gunType == GunType.Enemy)

{

Shoot();

}

else if(player.controlType == PlayerController.ControlType.Android)

{

if(joystick.Horizontal != 0 || joystick.Vertical != 0)

{

Shoot();

}

}

}

else

{

timeBtwShots -= Time.deltaTime;

}

}

public void Shoot()

{

Instantiate(bullet, shotPoint.position, shotPoint.rotation);

timeBtwShots = startTimeBtwShots;

}

}