**Київський національний торговельно-економічний університет**

**Кафедра комп'ютерних наук та інформаційних систем**

**ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЕКТ**

на тему:

# «Розробка мобільної 2D гри з процедурною генерацією рівня»

Студентки 4 курсу, 10 групи, Кубасова Кристина Олександрівна

спеціальності 122 «Комп'ютерні науки»

_____
*підпис студента*

Науковий керівник Кандидат фізико-математичних наук Філімонова Тетяна Олегівна

_____
*підпис керівника*

Гарант освітньої програми кандидат технічних наук, доцент Демідов Павло Георгійович

_____
*підпис керівника*

**Київ 2021**

**Київський національний торговельно-економічний університет**

Факультет інформаційних технологій

Кафедра комп'ютерних наук та інформаційних систем

Спеціальність 122 «Комп'ютерні науки»

**Затверджую**

Зав. кафедри _____Пурський О.І.

« » 2020р.

**Завдання**

**на випускну кваліфікаційну роботу (проект) студентці**

**Кубасова Кристина Олександрівна**

(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи (проекту)

*«Розробка мобільної 2D гри з процедурною генерацією рівнів»*

Затверджена наказом ректора від *«15 грудня» 2020р. № 3780*

2. Строк здачі студентом закінченої роботи *31 травня 2021 року*

3. Цільова установка та вихідні дані до роботи

*Мета роботи: художньо-технічно спроектувати концепт гри і реалізувати його за допомогою спеціалізованого програмного забезпечення.*

*Об'єкт дослідження: мобільна гра.*

*Предмет дослідження: методи проектування та розробки мобільної гри.*

4. Перелік графічного матеріалу_____

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

| Розділ | Консультант (прізвище, ініціали) | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | Завдання прийняв |
| 1 | Філімонова Т.О. | 22.12.2020 р. | 22.12.2020 р. |
| 2 | Філімонова Т.О. | 22.12.2020 р. | 22.12.2020 р. |
| 3 | Філімонова Т.О. | 22.12.2020 р. | 22.12.2020 р. |

6. Зміст випускної кваліфікаційної роботи (проекту) (перелік питань за кожним розділом)

*ВСТУП*

*РОЗДІЛ 1. Аналітична частина*

*1.1 Вибір жанру гри з прикладами існуючих розробок.*

*1.2 Ігри в жанрі раннер.*

*1.2.1 Crossy Road.*

*1.2.2 Temple Run 2.*

*1.2.3 Jetpack Joyride.*

*1.3 Ігри з процедурної генерацією.*

*1.3.1 Dark Lands.*

*1.3.2 Nyan Cat: Lost In Space.*

*1.3.3 BadLand.*

*1.3.4 GoNNER.*

*1.4 Вибір програмного забезпечення для реалізації проекту.*

*1.4.1 Графічні редактори.*

*1.4.2 Середовище моделювання об'єктів.*

*СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ*

*ДОДАТКИ*

7. Календарний план виконання роботи

| № Пор | Назва етапів випускної кваліфікаційної роботи | Строк виконання етапів роботи | |
|---|---|---|---|
| | | За планом | фактично |
| 1 | 2 | 3 | 4 |
| *1* | *Вибір теми випускної кваліфікаційної роботи* | *05.10.2020* | *05.10.2020* |
| *2* | *Розробка та затвердження завдання на випускну кваліфікаційну роботу* | *18.12.2020* | *18.12.2020* |
| *3* | *Вступ* | *03.02.2021* | *03.02.2021* |
| *4* | *1. Аналітична частина* | *26.02.2021* | *26.02.2021* |
| *5* | *2. Проектна частина* | *06.04.2021* | *06.04.2021* |
| 6 | *3. Розробка* | *12.05.2021* | *12.05.2021* |
| 7 | *Висновки* | *14.05.2020* | *14.05.2020* |
| 8 | *Здача випускної кваліфікаційної роботи на кафедру науковому керівнику* | *20.05.2020* | *20.05.2020* |
| 9 | *Попередній захист випускної кваліфікаційної роботи* | *26.05.2020* | *26.05.2020* |
| 11 | *Виправлення зауважень, зовнішнє рецензування випускної кваліфікаційної роботи* | *27.05.2020* | *27.05.2020* |
| 12 | *Представлення готової зшитої випускної кваліфікаційної роботи на кафедру* | *31.05.2020* | |
| 13 | *Публічний захист випускної кваліфікаційної роботи* | *За розкладом роботи ЕК* | |

8. Дата видачі завдання «22» грудня 2020 р.

9. Керівник випускної кваліфікаційної роботи (проекту)

Філімонова Т.О.
*(прізвище, ініціали, підпис)*

10. Гарант освітньої програми Демідов П.Г.
*(прізвище, ініціали, підпис)*

11. Завдання прийняв до виконання студент-дипломник Кубасова К. О.
*(прізвище, ініціали, підпис)*

12. Відгук керівника випускної кваліфікаційної роботи (проекту)

Керівник випускної кваліфікаційної роботи (проекту)

_____ 31.05.2021 р.

*(підпис, дата)*

**13. Висновок про випускну кваліфікаційну роботу (проект)**

Випускна кваліфікаційна робота (проект) студента Кубасова К.О.
*(прізвище, ініціали)*

може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми_____ Демідов П.Г.
*(підпис, прізвище, ініціали)*

Завідувач кафедри _____ Пурський О.І.
*(підпис, прізвище, ініціали)*

«_____»_____2021 р.

# АНОТАЦІЯ

Кубасова К. О. «Розробка мобільної 2D гри з процедурною генерацією рівнів»

В рамках цієї дипломної роботи було проведене дослідження і вивчена робота спеціалізованого програмного забезпечення.

Як результат був створені художньо та технічно спроектований концепт гри та комплексна розробка практичної її частини, а саме мобільна гра в жанрі раннер з процедурної генерацією рівня, за допомогою спеціалізованого програмного забезпечення. Для розробки були використані такі програмні продукти, як графічний редактор Adobe Photoshop, ігровий движок Unity та середовище розробки програмного забезпечення Microsoft Visual Studio.

КЛЮЧОВІ СЛОВА: МОБІЛЬНА ГРА, ГРА, ГЕЙМ-ДЕВЕЛОПМЕНТ, C#, UNITY, PHOTOSHOP, ГЕЙМ-ДИЗАЙН

SUMMARY

Kubasova KO "Development of a mobile 2D game with procedural generation of levels"

As a result, an artistically and technically designed concept of the game and a comprehensive development of its practical part was created, namely a mobile game in the genre of runner with procedural level generation, using specialized software. Software products such as Adobe Photoshop graphics editor, Unity game engine, and Microsoft Visual Studio software development environment were used for development.

KEY WORDS: MOBILE GAME, GAME, GAME-DEVELOPMENT, C #, UNITY, PHOTOSHOP, GAME-DESIGN

ЗМІСТ

# ВСТУП

З розвитком глобалізації та науково-технічним прогресом, що вийшли на найвищий, порівняно з усією історією світу, рівень, почали з'являтися очевидні результати роботи науковців: комп'ютери, смартфони, автоматизовані пристрої, одним словом – комп'ютерні технології. Це стало безумовно переломним моментом в історії і послужило накращим чином по відношенню до розвитку більшої частини галузей промисловості. За минулі чотири десятиліття індустрія комп'ютерних наук встигла не просто зародитися і здобути популярність, а дістатися й перевершити, відносно здобутих грошей, куди більш широковідому індустрію кіно. Хоча й починалося все дуже просто.

Розвиток комп'ютерних технологій потягнуло за собою розвиток ігор, чим приваблювало все більше і більше людей. На початку комп'ютерні ігри користувалися особливим попитом у людей з 7 до 15 років, розроблялися довго та геймплей був максимально простий [1]. А на сьогоднішній день, комп'ютерна техніка досягла такого рівня розвитку, що дозволяє програмістам розробляти реалістичні ігри з гарним звуковим і графічним супроводом, за порівняно короткий термін та мати успіх у людей всіх вікових категорій.

**Мета:** художньо-технічно спроектований концепт гри та комплексна розробка практичної її частини за допомогою спеціалізованого програмного забезпечення.

**Завдання:**

- аналіз доступної літератури стосовно розробки мобільних ігор;
- вибір жанру проекту;
- дослідження існуючих схожих розробок;
- розробка концепт мобільної гри;
- вибір програмного забезпечення для реалізації проекту;
- розробка продукту.

**Об'єкт дослідження:** процес розробки мобільної гри.

**Предмет дослідження:** методи проектування та розробки мобільної гри в жанрі "раннер".

**Методи дослідження:**

- загальнонауковий аналітичний метод;
- методи моделювання;
- методи розробки мобільних та комп’ютерних ігор.

**Практичне значення:** отримання навичок створення і розробки мобільного додатку, з використанням спеціалізованого програмного забезпечення (ПЗ), розвиток власних навичок та умінь для подальшого розвитку у сфері розробки софту, геймдевелопменту та геймдизайну.

В рамках цієї дипломної роботи була проведена дослідження і вивчена робота спеціалізованого програмного забезпечення та розроблена гра у жанрі раннер з нуля.

# РОЗДІЛ 1.
# АНАЛІТИЧНА ЧАСТИНА

## 1.1 Вибір жанру гри з прикладами існуючих розробок.

Перед тим як розпочати роботу над проектом необхідно вибрати жанр майбутнього проекту. Мобільні ігри можна класифікувати по різним аспектам, але по необхідності ми роздивимося найважливіший з них: жанр [2]. За допомогою Таблиці 1.1, було вирішено робити раннер з процедурною генерацією рівня.

Для того, щоб продукт мав успіх, його треба зробити цікавим. Отже, необхідно проаналізувати існуючі схожі розробки: виокремити найкращі раннери за останні декілька років, дізнатися відгуки користувачів, виділити цікаві механіки та дослідити плюси і мінуси.

*Таблиця 1.1*

**Жанри ігор**

| **Гра дії** | | |
|---|---|---|
| **Назва жанру** | **Опис** | **Приклади ігор** |
| Аркада | Швидко виконувати які-небудь дії (збирати, стріляти, бігати і т.д.). Безтурботний процес порушується присутністю на рівнях ворогів або пасток. Найчастіше вид зверху, рідше - вид збоку. Як правило не має сюжету. | «Pacman», «Lode Runner», «Bomberman» |
| Шутер | Головна задача – знищення усього, що рухається. Кількість боєприпасів майже завжди необмежена, тому стрілянину можна вести постійно [3]. | «Call of Duty», «Medal of Honor: Allied Assault», |
| Раннер | Ігри, де персонаж постійно біжить/летить/повзе вперед по теоретично нескінченного ігровому світу, проходячи повз перешкоди. | «Subway Surfers», «Jetpack Joyride» |
| Спорт | Управління людським тілом, загнане в певні рамки правил, які дотримуються з максимальною ретельністю. Територія також обмежена, не можна залишити поле або стадіон. | «NBA’09», «FIFA’09» |
| Симулятор | Симулювання реальних ситуацій. Більшу частину екрана займає внутрішній вигляд кабіни симульованої техніки. | «Silent Hunter 3», «Microsoft Flight Simulator» |
| Гонки | Змагання на певному виді техніки, зазвичай гоночні автомобілі. | «Need for Speed: Underground», «Test Drive Unlimited», «Ballistic» |

| | | |
|---|---|---|
| Файтинг | Ігри імітуючі бійки, рукопашні бої. | «Mortal Kombat», «Street Fighter IV», «Tekken» |
| Платформер | Гравцеві потрібно пересуватися по світу, зазвичай відбувається в 2D просторі, стрибати, ухилятися від пасток. | «Super Meat Boy», «N», «Super Mario» |
| Слешер | Подолання перешкод у вигляді ворогів, за допомогою дій «ухилення» і «знищення». | «Darksiders: Wrath of War», «Devil May Cry 4» |
| Екшн | Знищення ворогів, рівних по силі. | «Half-Life», «BioShock» |

## 1.2 Ігри в жанрі раннер.

Раннер (англ. Runner) – жанр мобільних ігор, в яких основною рисою ігрового процесу є невпинний рух. Тобто особливістю такого жанру є постійний рух по горизонталі, до того моменту, поки гравець не гине [4].

### 1.2.1 Crossy Road

Один з найбільш популярних нескінченних раннерів – був і залишається досі (Рис. 1.1). Його геймплей заснований на ідеях класичної гри Frogger, де замість бігу передбачені стрибки [5].


Рис.1.1 - Заставка гри Crossy Road

Головним героєм на початку гри виступає курочка, яка хоче перебігти дорогу, при проходженні гри розблоковуються нові персонажі. Перешкодами в цій грі служать дороги, по яким мчать машини, і річки з колодами. Персонажі й світ створені і анімовані за допомогою 3D технологій.

### 1.2.2 Temple Run 2

Мало не найвідоміший Раннер в історії ігор [6]. Сюжет розроблений в стилі

Індіани Джонса: «Візьми ідол, якщо наважишся», - говорить вам гра на старті (Рис. 1.2).

.

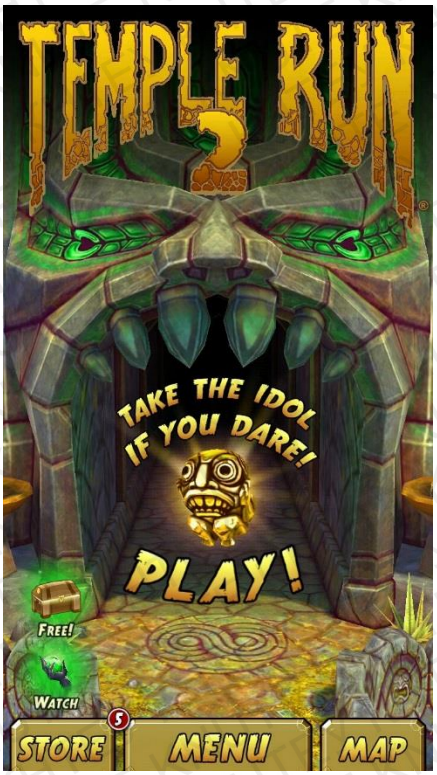


Рис.1.2 – стартове вікно гри Temple Run 2

Та тільки-но ви торкаєтеся до екрану, як за героєм починає гнатися величезна горила. Персонаж стрибає, сковзаєте, ухиляється, збирає монетки, відкриває навички. Щоб гравець не нудьгував, автори згодом наповнювали гру різноманітними локаціями і кожна радувала новими фішками і особливим міфічним чудовиськом замість горили (Рис. 1.3).


Рис.1.3 - заставка гри Temple Run 2

Хоча сьогодні Temple Run 2 і виглядає застарілою, азарт вона викликає неабиякий. Персонажі й світ створені і анімовані за допомогою 3D технологій.

### 1.2.3 Jetpack Joyride

Цей 2D-раннер постійно оновлюється і тому до сих пір користується популярністю [5]. Польоти по підземним лабораторіям за допомогою реактивного ранця, збір монет і покращення персонажа виглядає захоплююче (Рис. 1.4).


Рис.1.4 - заставка гри Jetpack Joyride

Гра використовує просту, в один дотик, систему контролю реактивного ранця: коли гравець натискає на будь-якому місці сенсорного екрану, то ранець спрацьовує і головний герой піднімається; коли гравець відпускає, то реактивний ранець вимикається, головний персонаж приземляється та починає бігти по землі. Гравець не контролює його швидкість, тому що він постійно в русі, але може контролювати його рух уздовж вертикальної осі.

## 1.3 Ігри з процедурної генерацією.

Процедурна генерація – це метод алгоритмічного створення даних за допомогою комбінацій алгоритмів, поєднаних із випадковістю. У відеоіграх він використовується для автоматичного чи напівавтоматичного створення великої кількості контенту: перепон, ворогів, текстур [7].

Таким чином за допомогою процедурної генерації світ гри стає більш наповненим та неповторюваним в плані послідовності локацій, а також полегшує

розробнику роботу автоматизованим заповненням текстур світу.

### 1.3.1 Dark lands

Раннер з битвами в світі тіней і величних пейзажів (Рис. 1.5). Присутня оптимізація характеристик головного героя, щоб пройти режим пригоди або вижити в нескінченному режимі, що теж є хорошим аспектом, бо таке розподілення розширює коло користувачів (Рис. 1.6).

Рис.1.5 - заставка гри Dark lands


Рис.1.6 - скріншот з гри Dark lands

Гра генерує світ впродовж усього проходження. Головний герой сам біжить вправо, ним можна підстрибнути, пригнутися, зупинитися, атакувати й прикритися щитом. Вороги різноманітні і мають різні здібності, як і боси.

Щодо інтерфейсу, справа кнопка меню-паузи зліва вгорі вказана смужка здоров'я, а поруч з нею пройдена відстань з початку гри.

### 1.3.2 Nyan Cat: Lost In Space

Цей раннер підкупає користувача своїм дизайном, бо головним героєм виступає котик з тілом печивка, за котрим шлейфом тягнеться райдуга (Рис. 1.7) [8]. Має декілька модів (режимів), які відрізняються текстурами (Рис. 1.8).


Рис.1.7 - заставка гри Nyan Cat: Lost In Space


Рис.1.8 - скріншот з гри Nyan Cat: Lost In Space

Класичний мод – оригінальний режим, у якому котик Ньян стрибає на платформах, оминаючи перешкоди.

Існує 4 різні типи платформ: ковбасна, крихка платформа з торту, золота і слизька платформа з торту.

### 1.3.3 BadLand

Серія 2D екшн-пригодницьких ігор, основна мета яких – пролетіти маленькою птахоподібною істотою через ліс (Рис. 1.9).


Рис.1.9 - заставка гри BadLand

Гра проходить в чотири етапи, які складаються із світанку, полудня, сутінків та ночі, кожен із окремою кольоровою гамою та новою темою пасток. Гравець пересувається головним героєм за допомогою натисканням по екрану. Персонажі й світ створені і анімовані за допомогою 2D стилю [9].

### 1.3.4 GoNNER

Двомірний платформер в жанрі аркада-екшен з процедурною генерацією світу, непростими босами, секретами та захоплюючим сюжетом (Рис. 1.10).

Особливістю гри представляються різні старти та кінцівки, які можна розблокувати за допомогою декількох проходжень, також існує чотири світи, до яких можна перейти, вбиваючи поточного боса кінця світу [10].


Рис.1.10 - заставка гри GoNNER

## 1.4 Вибір програмного забезпечення для реалізації проекту.

Серед існуючого ПЗ, необхідно підібрати найбільш комфортний та придатний для розробки проекту. Потрібно підбирати програми з зручним інтерфейсом, широким функціоналом, вільною ліцензією та можливістю інтегрувати файли в необхідне розширення.

### 1.4.1 Графічні редактори.

Планується використання лише одного редактору:

Adobe Photoshop — графічний редактор для редагування растрових зображень [11]. Особливості:

- величезний інструментарії для операції створення і обробки зображень;
- висока якость обробки графічних зображень;
- зручність й простота у використанні;
- можливості до автоматизації обробки растрових зображень;
- механізми роботи з кольоровими профілями;
- великий набір команд фільтрації, за допомогою яких можна створювати найрізноманітніші художні ефекти.

Так як гра буде малюватися від руки, саме цей графічний редактор краще усього підійде для створення та перенесення у цифровий формат текстур світу та

персонажів.

### 1.4.2 Середовище моделювання об’єктів.

Ігровий рушій (англ. Game engine) — центральна програмна частина будь-якої відеогри, яка відповідає за всю її технічну сторону, дозволяє полегшити розробку гри шляхом уніфікації та систематизації її внутрішньої структури [12]. Важливим значенням рушія є можливість створення багатоплатформових ігор.

Основну функціональність гри зазвичай забезпечує її рушій, до якого входить «візуалізатор» (рушій рендерингу), фізичний рушій, звук, система скриптів, анімація, ігровий штучний інтелект, мережевий код, керування пам’яттю, багатонитевість і графічні сцени.

#### 1.4.2.1 Unity

Unity – це більше, ніж движок, це середовище для розробки комп'ютерних ігор, в якому об'єднані різні програмні засоби, що використовуються при створенні ПЗ – текстовий редактор, компілятор, відладчик і так далі [13]. При цьому, завдяки зручності використання, Unity робить створення ігор максимально простим і комфортним, а мультиплатформність движка дозволяє розробників охопити якомога більшу кількість ігрових платформ і операційних систем.

Переваги:

- дає можливість розробляти ігри, не вимагаючи для цього якихось особливих знань;
- наявність величезної бібліотеки ассетів(об’єктних ресурсів) і плагінів, за допомогою яких можна значно прискорити процес розробки гри;
- підтримка величезної кількості платформ, технологій, API. Тобто створені на движку ігри можна легко перенести між ОС Windows, Linux, OS X, Android, iOS, на консолі сімейств PlayStation, Xbox, Nintendo, на VR- і AR-пристрої;
- доступна фізика твердих тіл, регдолл і тканин, система Level of Detail(тривимірна графіка), колізії між об'єктами, складні і прості

анімації;

- доступний безкоштовно, але така версія движка доступна лише, якщо проект, створений з її допомогою, не приносить розробнику більше 100 тисяч долларів на рік.

Недоліки:

- виникають труднощі з реалізацією складних проектів;
- повільність;
- на виході навіть сама легка гра буде важити багато.

#### 1.4.2.2 CryEngine

Переваги:

- просунуті можливості по розробці відеоігор і підтримкою самих передових технологій;
- дозволяє створювати ігри з майже фотореалістичною графікою;
- технологія трасування променів на движку, яка працює на відеокартах AMD і Nvidia і не вимагає потужності графічних чіпів RTX [14].

Недоліки:

- складність збірки білда;
- наявність багів в редакторі;
- скромний вибір ассетів;
- обмеження для розробки мережевої гри;
- відсутність хорошої тех. підтримки і активного ком'юніті;
- складний в освоєнні.

#### 1.4.2.3 Unreal Engine

Переваги:

- дуже гнучкий і універсальний;
- містить безліч інструментів, які полегшують роботу з ним;
- має велику колекцію ассетів (платних і безкоштовних);
- використовує передові технології [15].

Недоліки:

- проблематично створювати великі безшовні світи, розраховані на безліч гравців і наявністю штучного інтелекту;
- розроблений для професіоналів;
- змушує більше працювати над оптимізацією ігор.

Після проведеного порівняння було вирішено використовувати ігровий движок Unity.

### 1.5 Загальний алгоритм реалізації проекту.

Алгоритм реалізації проекту включає в себе наступні кроки:

1) Створення та продумування деталей проекту.
2) Малювання текстур та бонусів.
3) Малювання персонажа та інтерфейсу.
4) Обробка текстур та бонусів у графічному редакторі.
5) Обробка персонажа у графічному редакторі.
6) Анімація та налаштування персонажа та додаткових об’єктів в ігровому движку.
7) Налаштування початку та анімації локацій.
8) Написання скриптів генерації бонусів та перепон під час гри.
9) Написання інших скриптів і налаштування інтерфейсу.
10) Налаштування кінцевого меню.
11) Створення візуальних ефектів інтерфейсу.
12) Створення головного меню.
13) Додавання звуків і музики.
14) Зменшення розміру проекту та його компіляція.

**Висновки**

Після проведеного дослідження по вибору жанру гри було вирішено робити гру-раннер. Також було розглянуто та проаналізовано існуючі схожі розробки і відібрано, як приклад, три різних гри в жанрі раннер та ще чотири гри з процедурною генерацією задля виокремлення цікавих механік, сюжетних, графічніх та анімаційних рішень. Таким чином шанс, що продукт буде більш успішним для розробника та цікавим для користувача виріс.

Окрім того було проведене порівняння та опис кращих існуючих ПЗ, а саме: ігрових рушіїв та графічних редакторів. Як результат дослідження, було вибрано найкомфортніші у використанні продукти: ігровий движок Unity та графічний редактор Photoshop.

# РОЗДІЛ 2.
# ПРОЕКТНА ЧАСТИНА

## 2.1 Актуальність проекту.

З розвитком глобалізації та науково-технічним прогресом, з'явилися мобільні пристрої, які об'єктивно стали невід'ємною частиною життя кожної прогресивної людини. Дуже велика кількість користувачів, будь то діти чи люди за 70, проводять свій час за будь-якого роду іграми у вільний час.

Головна причина такої популярності це ескапізм. Уже багато століть люди намагаються втекти від нудної, неприємної буденності за допомогою історій, книжок, фільмів, ігор.

Крім того, книжки та фільми, як надійний спосіб ескапізму, це, звичайно, добре, але ігри дають більше можливостей розробнику взаємодіяти з споживачем.

Також дуже важливим фактором стає швидкість сучасного світу, наповненого стресами, багатьом людям просто необхідно швидко та якісно відключити мозок та робити якісь абстраговані, зазвичай автоматичні, дії. До речі, саме тому такий жанр як Клікер [16], ігровий процес в якому складається з виконання простих дій, таких як багаторазове натискання на екран, залишається до сих пір популярним.

До недавнього часу однією з головних причин актуальності ігор був ринок, який потребував більше цікавих розробок в даному напрямку. Але з початком пандемії у багатьох розробників, ігроробів-початківців, а також у людей, яким просто нудно, з'явилося більше часу, тому ринок став перенасичений іграми різних жанрів та якостей. Та я вважаю, що там, де закриваються одні двері, відкриваються інші, а отже, карантин це не тільки вільний час у розробників, але й у майбутніх користувачів. Люди більше проводять часу вдома, таким чином, більше шукають можливості абстрагуватися від проблем сьогодення і грають в ігри.

Особливим даний продукт зробить графіка та анімація, так як усі фони,

перепони, персонаж, меню і так далі будуть намальовані від руки.

### 2.2 Характеристика потенційної аудиторії проекту.

Цільовою аудиторією проекту будуть люди, які мають характеристики згідно Таблиці 2.1 [17].

*Таблиця 2.1*

**Характеристика аудиторії**

| Показник | Характеристика | |
|---|---|---|
| Стать | Жіноча | Чоловіча |
| Співвідношення | 60% | 40% |
| Вік | 5 - 35 | 5 - 18 |
| Інтереси | Казуальні ігри, слов'янська міфологія, етнічна тематика. | |
| Мета гри | Зайняти вільний час, потрапити в казковий міфічний світ. | |
| Психологічний тип | Накопичувачі(Achievers) – люди, яких стимулює внутрішньо-ігровий прогрес [18]. | |

Більшою мірою проект розрахований на жіночу аудиторію 5 - 35 років, з метою зайняти їх вільний час та потрапити у казковий світ, наповнений міфічними та етнічними оберегами та амулетами. А бажання накопичувати внутрішньо-ігрові досягнення буде спонукати користувачів повернутися до гри знову.

### 2.3 Концепція.

Далі будуть розглянуті ідеї, варіації головних аспектів гри.

#### 2.3.1 Назва і сюжет.

Дуже важливо створити назву, що зможе правильно відобразити суть гри. Саме від неї буде залежати подальше сприйняття проекту гравцями.

Варіанти:

1) Світилка;
2) Вогник;
3) Міф;
4) Twinkle ( з англ. – вогник, мерехтіння).

Було вирішено вибрати назву «Світилка».

Сюжет гри зосереджений на головному герої, який повинен пройти якомога більшу відстань у лісі, в якому живе, поки не помре і під контролем гравця намагається минати усі перешкоди, попутно збираючи скарби.

**2.3.2 Ігрова логіка.**

Головний герой має пройти якомога більшу відстань ігрового світу. На цьому шляху гравець буде зустрічати перепони на локаціях, що змінюються. Також будуть збиратися скарби у вигляді золотих монеток, з накопиченням певної кількості монеток персонаж зможе поставити щит на деякий час і не боятися, що помре при зустрічі з перешкодою.

З початку гри у головного героя є 3 HP (з англ. Health Points – пункти здоров'я), при зустрічі з перепонами знімається 1HP.

**2.3.3 Локації.**

Усі локації необхідно зробити максимально природніми, щоб підходити під сюжет. Сезон – початок осені, середина вересня. Тому були вибрані 2 основні локації та 1 додаткова:

- Ліс – ранок осіннього лісу, сонце тільки-но вийшло і проглядає між кронами жовтогарячих дерев. Локація наповнена гніздами птахів, магічними амулетами та оберегами.
- Болото – локація виглядає вже не так сонячно, як минула. Озеро, з якого виглядають то кущі, то колода, то очерет. З дерев звисають лози й майже не дають сонцю дістати до землі. На невисоких деревах поселилися магічні обереги.
- Нічний ліс (додаткова) – ця локація виглядає так само як і перша, але її наповнюють світлячки та місячне сяйво. Додатковою локація є тому, що

вона діє лише після 22:00 і дійсна до 8:00 реального часу.

Для локацій були розглянуті референси і приблизні концепти. Слідуючи представленим ескізам, створювалася кінцева графіка (Рис. 2.1).

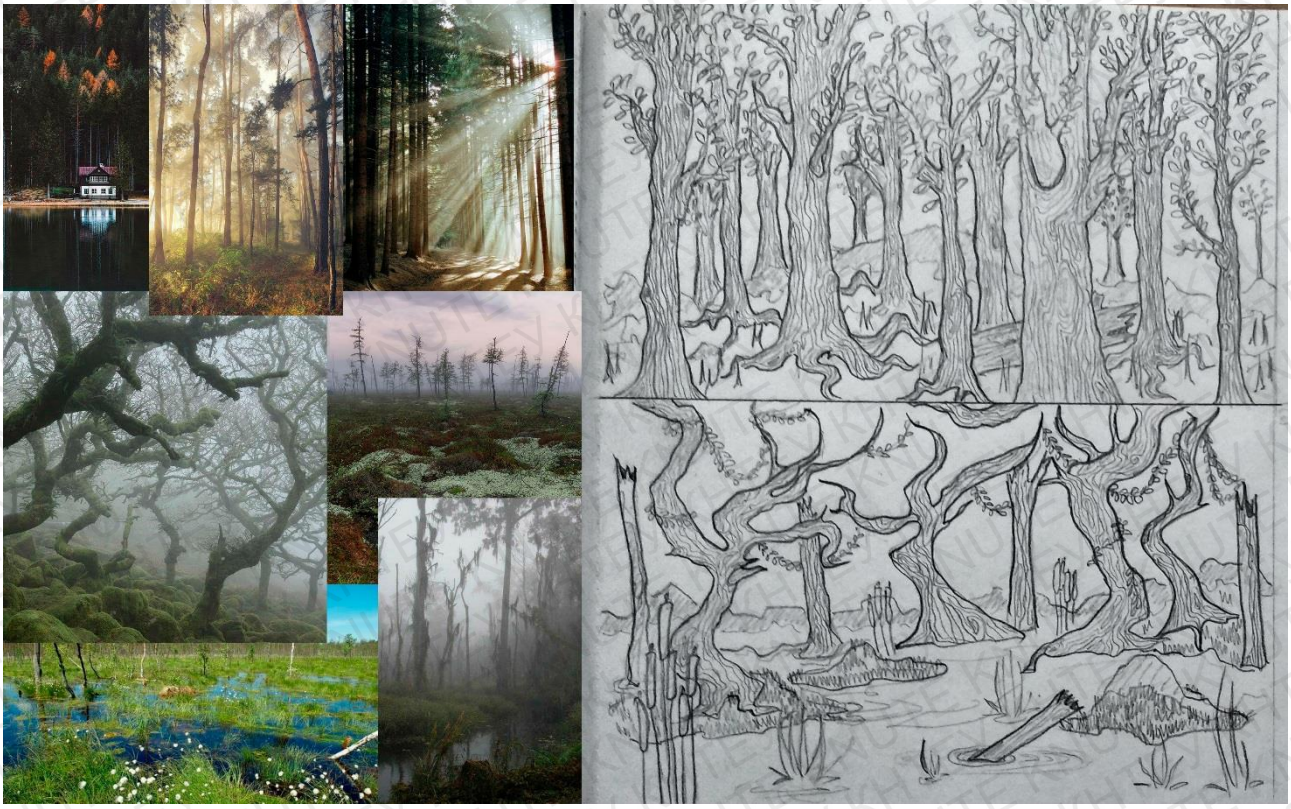

Рис.2.1 - концепти і готові малюнки локацій

### 2.3.4 Головний герой.

Було проведене дослідження по пошуку цікавих міфічних істот України, було декілька варіантів на вибір:

- Аука – лісний дух, що заводить людей в нетрі;
- Берегиня – дух, пов’язаний з культом дерев і рослинності;
- Індрик-звір – міфічна тварина, цар лісу.
- Курдуші – злі духи, що заражають людей хворобами;
- Пущовик – лісовий дух, зупиняє пожежі, не випускає людей з лісу.
- Світилка.

Вибір впав на останнього міфологічного персонажа, тому далі викладено більше інформації про нього.

Головним героєм є персонаж українського фольклору Світилка [19]. За народними віруваннями, світилка з'являється восени і, в залежності від повір’я,

є  чи то блукаючими духами потопельників і вішальників, чи то злим духом, що при зустрічі з людиною викликає у неї переляк і доводить до смерті, чи добрим духом, що допомагає людині вибратися з трясовини або показує дорогу вночі. З’являється цей персонаж на полях, з яких вже зібрали урожай, та в лісах,  на болотах і кладовищах, на трьох останніх світилка вказує на закопані там скарби. Саме тому персонаж гри мандрує лісом, болотом і нічним лісом, збираючи монетки.

Виглядатиме Світилка як маленький вогник з очами (Рис. 2.2).

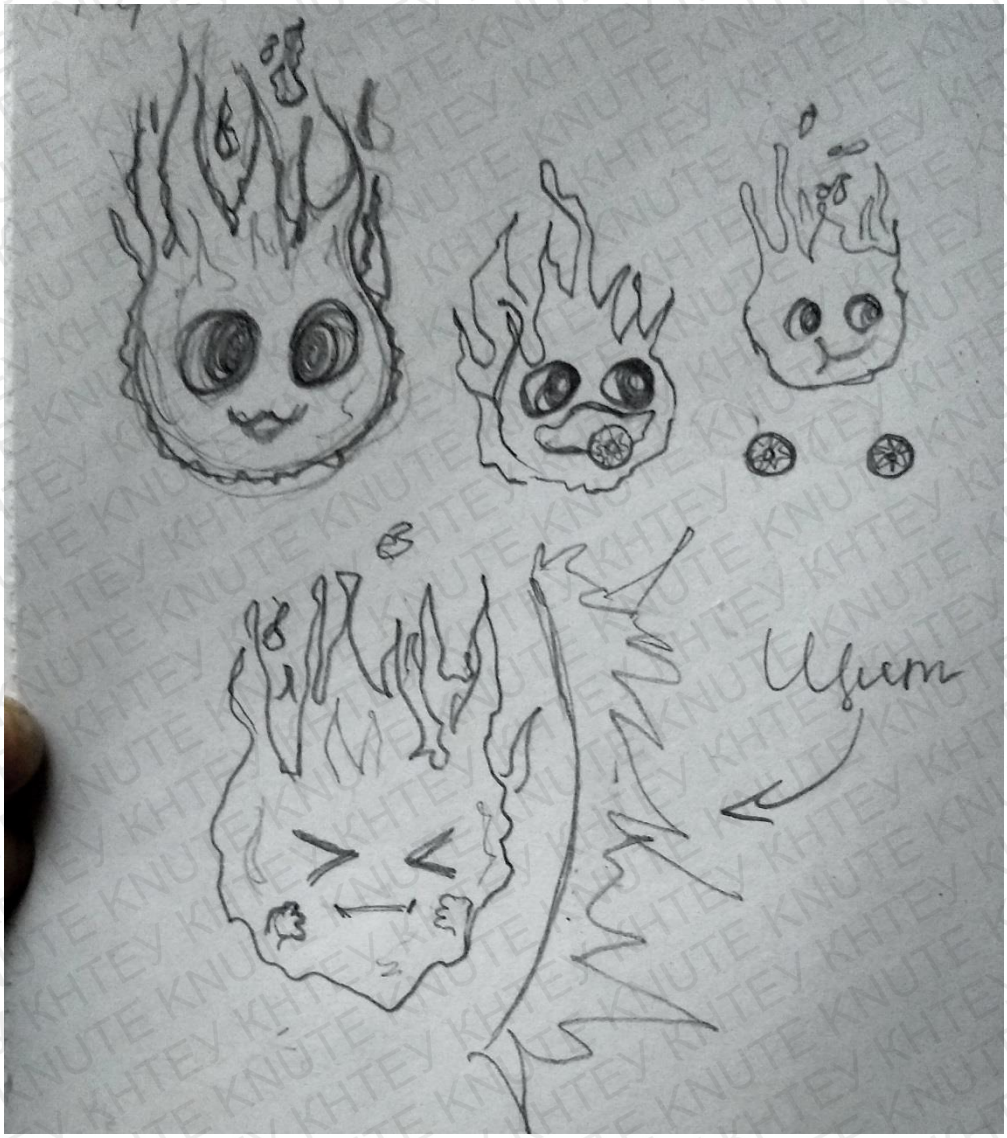


Рис.2.2 - концепт-арти головного  героя

### 2.3.5 Об'єкти, що завдають шкоди.

Є декілька локацій та перешкоди, які викладені у Таблиці 2.2.

*Таблиця 2.2*

**Перешкоди**

| Локація | Місце на екрані | Перешкода |
|---|---|---|
| Болото/ Ліс / Нічний ліс | Верх | Лоза/ Гніздо |
| | Середина | Паличний чоловічок |
| | Низ | Колода |

## Висновки

Актуальним проект робить сильна потреба людей у ескапізмі, особливо у часи максимального стресу через пандемію. Так як гра буде намальована від руки, вона зацікавить користувачів та легко перенесе їх у світ казки.

Більшою мірою проект розрахований на жіночу аудиторію 5 - 35 років, з метою зайняти їх вільний час та потрапити у казковий світ, наповнений міфічними та етнічними оберегами та амулетами.

Також були розглянута концепція гри з такими аспектами як назва, сюжет, ігрова логіка, локації, головний персонаж та перешкоди. Головний герой Світилка мандрує лісом та болотом, збираючи скарби та оминаючи перепони.

Таким чином у розділі було викладено актуальність проекту, досліджені і викладені основні характеристики потенційної аудиторії та головні складові проекту.

# РОЗДІЛ 3.
# РОЗРОБКА

Після створення концепції гри, слідує етап розробки. Саме тому далі буде викладений опис аспектів стосовно розробки з детальними описами виконаних дій.

## 3.1 Візуальна частина.

Візуальна складова гри ледь не найважливіша частина гри, адже саме від неї багато в чому залежить сприйняття гри гравцем. Уся візуальна частина була відтворена вручну олівцем та розмальована за допомогою програми Adobe Photoshop.

### 3.1.1 Головний персонаж.

Головний герой – це невід'ємна частина гри, яка повинна містити у собі все, щоб сподобатися гравцю: візуальну частину, передісторію, анімації тощо.

Створення мого персонажа почалося з концепту: дух Світилка має вигляд вогника, в залежності від повір'я може бути злим чи добрим духом. Був вибраний незлий тип персонажу і зроблений максимально «сонячним» та добрим.

Робота над Світилкою почалася з пари замальовок (Рис. 2.2), після чого були створені три основних малюнка персонажа, щоб пізніше створити покадрову анімацію горіння вогника.

Відштовхуючись від цих основних трьох зображень були створені і інші покадрові анімації з додаванням певних елементів задля набуття персонажем різних емоцій (Рис. 3.1).

Далі створювались та налаштовувались необхідні анімації за допомогою машини станів анімацій в движку Unity (Рис. 3.2).

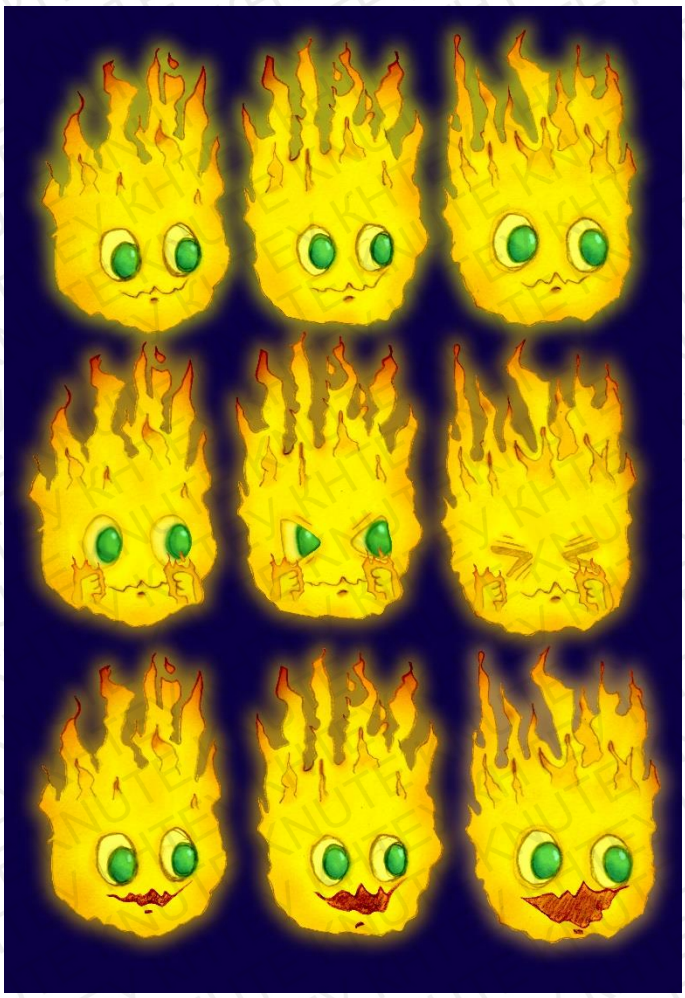

Рис. 3.1 - покадрова анімація головного персонажа

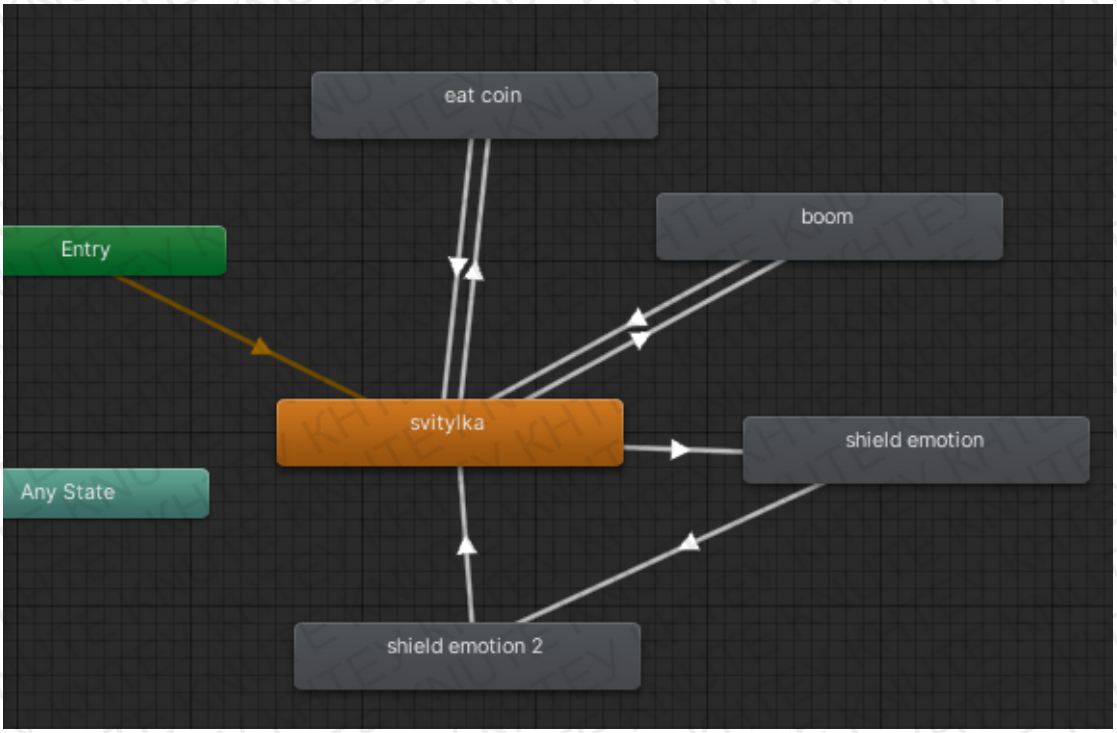


Рис. 3.2 - машина станів анімації головного персонажа

### 3.1.2 Оточення.

Оточення складається з фонів та перепон, що рухаються з різними швидкостями, для створення ефекту глибини та руху (Рис. 3.3). У кодовій частині фонам заданий час зміни з одного фону на інший, таким чином кожні 2 хвилини «Туманне болото» змінює «Сонячний осінній ліс», а після 22-ї години локального часу вмикається фон «Нічний ліс».

Рис. 3.3 - зображення перешкод

### 3.1.3 Інтерфейс.

Інтерфейс – це індикатори, що показують гравцеві кількість пунктів здоров'я, пройденого шляху, зібраних речей тощо. Зазвичай розташовуються на екрані під час гри таким чином, щоб не перекривати сам процес гри собою та бути корисним гравцеві.

У проекті було створено мінімальний інтерфейс, який містить у собі:

- Показник пунктів здоров'я;
- Кількість пройденого шляху;
- Кнопку щита, що з'являється при певних умовах.

### 3.1.4 Меню програшу.

Меню програшу з'являється на екрані, коли усі пункти здоров'я персонажа закінчуються. Воно також мінімальне (Рис. 3.4), має:

- Текст «Game Over» (з англ. – «Гра Завершена»);
- Кнопку запуску гри заново («Рестарт»);

- Кнопку виходу до головного меню.

Кнопки мають у собі звукові та візуальні ефекти, чим оживляють мінімалістичне меню.


Рис. 3.4 – меню програшу

### 3.1.5 Головне меню.

Головне меню викликається при запуску гри та з меню програшу (Рис. 3.5). Має всього дві активні кнопки, які також мають звукові та візуальні ефекти : «Нова гра» (для початку гри) та «Вийти» (для виходу з гри).

Візуально являє собою такий собі лісовий язичний вівтар с підвішеними букетами польових квітів та ароматичними паличками, дим яких анімований. На


Рис. 3.5 – головне меню

задньому плані, в залежності від локального часу, пропливає чи денний ліс, чи нічний. Анімації в меню створюють особливу атмосферу оживлення місця.

### 3.2 Написання коду.

Ігрові об'єкти можуть контролюватися за допомогою компонентів, які приєднуються до них у середовищі Unity, але незважаючи на те, що вбудовані компоненти дуже різнобічні, їх можливостей часто недостатньо для реалізації особливостей геймплея. Щоб виправити це, Unity дозволяє використовувати свої власні компоненти, створюючи скрипти, на одній з підтримуваних середовищем мов (C# або UnityScript). Вони дозволяють активувати ігрові події, змінювати параметри компонентів тощо.

Була вибрана мова програмування C#.

#### 3.2.1 Задній план.

Скрипт заднього плану має назву «BG», скорочено з англійської background (Додаток А). Складається з:

- Механіки руху фону;
- Реалізації зміни фону через деякий час;
- Реалізації зміни фону на нічний режим за локальним часом.

Є посилання на компонент об'єкта анімація.

#### 3.2.2 Персонаж.

Скрипт персонажа називається «Player» (Додаток Б). Він містить у собі:

1) Механіку головного героя: швидкість та дальність переміщення по ігровому полю, кількість життя, анімації.
2) Механіку щита, прив’язаного до головного героя: активація кнопки, таймера та анімації.
3) Накопичення монеток при контакті з гравцем.
4) Необхідні прив’язки аудіо-ефектів.
5) Активацію меню програшу.
6) Активацію скрипта управління свайпами.

Є посилання на компоненти об'єкта: аніматор, фізичне тіло, колайдер.

### 3.2.3 Перешкоди та монетки.

Перешкоди та монетки за типом своєї реалізації схожі. Мають компонент об'єкта колайдер та додаткові точки для рандомної генерації.

Скрипти мають назву «Barriers» (Додаток В) і «Coin» (Додаток Г).

У першому задається:

- швидкість переміщення об’єктів;
- реалізація зіткнення з гравцем у тому числі з завдаванням йому шкоди;
- реалізація зіткнення зі щитом.

Скрипт Coin має менше реалізацій, бо більша частина прописана у скрипті персонажа:

- швидкість переміщення об'єктів:
- знищення об'єкту при активації меню програшу;
- реалізація ефектів при зіткненні з гравцем та щитом.

### 3.2.4 Генерація об'єктів.

Як було зазначено в попередньому пункті для процедурної генерації об’єктів було створено додаткові точки, з яких, за допомогою скриптів «Spawner» (Додаток Ґ) та «SpawnerCoins» (Додаток Д), з’являються необхідні зображення перешкод чи монет.

Скрипт «Spawner» містить у собі реалізацію:

- рандомної появи об’єктів;
- задання необхідних об'єктів точкам;
- зменшення часу звуження інтервалу.

Скрипт «SpawnerCoins» має таку саму структуру та реалізації, окрім зменшення часу, після якого інтервал не стає вужче.

### 3.2.5 Інше.

Також в проекті присутні наступні скрипти, які коротко описані в Таблиці 3.1 (Додаток Е):

*Таблиця 3.1*

**Короткий опис скриптів**

| Назва скрипта | Реалізації |
|---|---|
| «Shield» | реалізація таймера щита та вимкнення анімацій персонажа з вимкненням щита. |
| «Point» | скрипт додається точкам, що використовуються в генерації об’єктів, для задання їм необхідних об’єктів |
| «SceneManagement» | використовується для зміни сцен у головному меню та самій грі. За допомогою цього скрипта, натиснувши, наприклад, на кнопку «Нова гра» у головному меню увімкнеться сама гра. |
| «Score» та «Destroyer» | структура скриптів однакова, але перший використовується для підрахунку пройдених перешкод, а другий – для видалення їх. «Destroyer» використовується, бо якщо не видаляти об’єкти після виходу зі сцени, через деякий час гра почне дуже підвисати. |
| «SwipeManager» | створений для управління персонажем за допомогою свайпів на телефоні. |

### 3.3 Музичний супровід.

Для звукового супроводу було проведене дослідження сучасних українських гуртів, які створюють музику жанрів, пов'язаних з етнічною музикою різних народів, особливо українського.

Після проведеного дослідження, було виділено декілька позицій та обрані композиції і звуки до них:

1) Головна тема:
   - Меню – денна, нічна;
   - Гра – денна, нічна.
2) Звук ввімкнення щита.
3) Звук натиску на активні кнопки.
4) Звук з’їдання монетки.

5) Звук зіткнення з перешкодами.

Головна музична тема проекту була створена українським музичним етногуртом ДахаБраха [20]. Різниця між темою в меню та грою полягає в тому, що композиція в меню має вокал, а у грі вона обрізана. Різниця між денною та нічною версіями: денна має на фоні спів пташок, а нічна – звук цвіркунів.

Звуки були створені шляхом доробки вихідних файлів, узятих з сайту freesound.org, що поширює аудіофайли для озвучки безкоштовно [21].

**3.4 Підсумок проекту.**

В ході розробки проекту було створено:

- унікальну візуальну складову об’єктів та текстур;
- 2 сцени;
- 3 фонових зображення;
- 14 спрайтів для анімацій та 5 анімаційних станів персонажа;
- 14 скриптів;
- 2 алгоритма генерації об’єктів;
- 4 спрайти перешкод;
- 2 спрайти та анімацію монетки;
- 2 ігрових меню;
- 2 анімаційні ефекти для перешкод та монетки;
- 4 активних кнопки;
- анімація щита.
- анімація “оживлення” фону у головному меню.

**Висновки**

У розділі був викладені аспекти стосовно розробки з детальними описами виконаних дій.

Візуальна складова є дуже важливою частиною гри, тому задля набуття унікальності уся візуальна частина була відтворена вручну олівцем та розмальована за допомогою програми Adobe Photoshop, а саме: фонові зображення, зображення головного героя, перешкод, щита, бонусів, інтерфейсу,

головного меню та меню програшу. За допомогою мови програмування C#, середовища для створення ігор Unity та середовища розробки програмного забезпечення Microsoft Visual Studio були прописані та задані поведінка усіх об’єктів гри. Також було проведене дослідження з пошуку підходящого аудіо-супроводу та задано деяким об’єктам проекту для кращого відтворення атмосфери гри.

# ВИСНОВКИ

Хоча й на сьогоднішній день індустрія мобільних та комп'ютерних ігор дуже розвинута та є безпосередньою частиною нашої повсякденності, в Україні ця індустрія, порівняно з іншими країнами, розвивається повільно. Тому була вибрана саме така тема дослідження.

В роботі були опрацьовані існуючі матеріали на тему гейм-девелопменту та гейм-дизайну. Був проведений аналіз доступної літератури Також були проглянуті існуючі схожі розробки.

Був вибраний жанр створюваного проекту, розроблений концепт гри і аналіз та вибір доступного програмного забезпечення для реалізації проекту.

Проведена робота дозволила отримати досвіт у багатьох областях, пов'язаних з дослідженням необхідної інформації та розробкою ігор. Наприклад, написання алгоритмів на мові програмування C#, створення концептів персонажа, деталей гри, інтерфейсу, створення анімацій та важливий заключний етап створення гри – налаштування та збірка проекту в ігровому движку.

Як результат був створений художньо та технічно спроектований концепт гри та комплексна розробка практичної її частини, а саме мобільна гра в жанрі раннер з процедурної генерацією рівня, за допомогою спеціалізованого програмного забезпечення. Для розробки були використані такі програмні продукти, як графічний редактор Adobe Photoshop, ігровий движок Unity та середовище розробки програмного забезпечення Microsoft Visual Studio.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. История компьютерных игр. Как все начиналось. [Електронний ресурс]. – Режим доступу: https://www.computer-museum.ru/games/genesis.htm
2. Жанры компьютерных игр (общая схема). [Електронний ресурс]. – Режим доступу: https://gamesisart.ru/TableJanr.html#Horror/
3. Что за жанр? Endless (infinity) running game. [Електронний ресурс]. – Режим доступу: https://stopgame.ru/blogs/topic/85445#:~:text=%D0%9A%D0%B0%D0%BA%20%D0%B3%D0%BE%D0%B2%D0%BE%D1%80%D0%B8%D1%82%20Wiki%2C%20Endless%20(Infinity,%D0%B2%20%D0%BD%D0%B5%D0%B3%D0%BE%20%D0%BA%D1%80%D0%BE%D0%B2%D0%BE%D0%B6%D0%B0%D0%B4%D0%BD%D0%BE%D1%81%D1%82%D0%B8%20%D0%B8%20%D0%BE%D0%B1%D0%BD%D0%B0%D0%B6%D0%B5%D0%BD%D0%BA%D0%B8
4. О мобильных раннерах и их классификации. [Електронний ресурс]. – Режим доступу: https://dtf.ru/mobile/40836-o-mobilnyh-rannerah-i-ih-klassifikacii
5. В какие бесконечные раннеры стоит поиграть? Подборка самых достойных. [Електронний ресурс]. – Режим доступу: https://app-time.ru/game-parad/v-kakie-beskonechnye-rannery-stoit-poigrat-podborka-samyh-dostoynyh
6. 10 лучших мобильных раннеров. [Електронний ресурс]. – Режим доступу: https://m.igromania.ru/article/30803/10_luchshih_mobilnyh_rannerov_Ot_Temple_Run_2_i_Talking_Tom_Hero_Dash_do_Rayman_Jungle_Run_i_Super_Mario_Run.html
7. Процедурна генерація. [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Процедурна_генерація
8. Nyan Cat: Lost in Space (app). [Електронний ресурс]. – Режим доступу: https://nyancat.fandom.com/wiki/Nyan_Cat:_Lost_in_Space_(app)#Description
9. Badland (video game). [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Badland_(video_game)

10. Gonner. [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Gonner
11. Adobe Photoshop. [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/Adobe_Photoshop
12. Ігровий рушій. [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Ігровий_рушій
13. Движок Unity – особенности, преимущества и недостатки. [Електронний ресурс]. – Режим доступу: https://cubiq.ru/dvizhok-unity/
14. Движок CryEngine – особенности, преимущества и недостатки. [Електронний ресурс]. – Режим доступу: https://cubiq.ru/dvizhok-cryengine/
15. Движок Unreal Engine – особенности, преимущества и недостатки. [Електронний ресурс]. – Режим доступу: https://cubiq.ru/dvizhok-unreal-engine/
16. Инкрементальная игра. [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%BA%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0
17. Для кого эта игрушка или как определить целевую аудиторию. [Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/253895/
18. Психотипы Бартла и балансировка аудитории. [Електронний ресурс]. – Режим доступу: https://habr.com/ru/company/mailru/blog/263839/
19. Кононенко О.А. УКРАЇНСЬКА МІФОЛОГІЯ ТА КУЛЬТУРНА СПАДЩИНА. Міфологічні уявлення, вірування, обряди, легенди та їхні відлуння у фольклорі і пізніших звичаях українців, братів-слов'ян та інших народів/ О.А. Кононенко – Харків: Фоліо, 2011. – 716 с.
20. ДахаБраха [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/ДахаБраха
21. Звуки [Електронний ресурс]. – Режим доступу: https://freesound.org

# Скрипт BG

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System;

public class BG : MonoBehaviour
{
    public float speed;
    public float endX;
    public float startX;
    public int dateTimeStart;
    public float time;
    public GameObject bgForest, bgSwamp, bgNightForest;
    bool bgForest_stat, bgSwamp_stat;
    private Animation bgForest_anim;
    private Animation bgSwamp_anim;
    public GameObject daySound;
    public GameObject nightSound;
    private void Start()
    {
        dateTimeStart = DateTime.Now.Minute;

        bgSwamp = GameObject.FindWithTag("Swamp");
        bgForest = GameObject.FindWithTag("Forest");

        bgSwamp_anim = bgSwamp.GetComponent<Animation>();
        bgForest_anim = bgForest.GetComponent<Animation>();

        bgSwamp_stat = true;
        bgForest_stat = false;

        time = DateTime.UtcNow.ToLocalTime().Hour;
        if (time >= 8 && time < 22) // денний режим
        {
            Instantiate(daySound, transform.position, Quaternion.identity);

            bgNightForest.SetActive(false);
            bgForest.SetActive(true);
            bgSwamp.SetActive(true);
        }
        else if (time >= 22 || time < 8) // нічний режим
        {
            Instantiate(nightSound, transform.position, Quaternion.identity);

            bgSwamp.SetActive(false);
            bgForest.SetActive(false);
            bgNightForest.SetActive(true);
        }

    }

    private void Update()
    {
       transform.Translate(Vector2.left * speed * Time.deltaTime);

        if (transform.position.x < endX) // рух фону
        {
            Vector2 pos = new Vector2(startX, transform.position.y);
            transform.position = pos;
```

```
}
        int speedNum = Mathf.RoundToInt(speed * 5f);

        if (DateTime.Now.Minute > dateTimeStart + 2)
        {
            if (bgSwamp_stat) // активація фону лісу
            {
                bgSwamp_anim.Play("kill swamp");
                bgForest_anim.Play("create forest");

                bgSwamp_stat = false;
                bgForest_stat = true;

            }
            else if (bgForest_stat) // активація фону болота
            {
                bgForest_anim.Play("kill forest");
                bgSwamp_anim.Play("create swamp");

                bgSwamp_stat = true;
                bgForest_stat = false;
            }
            dateTimeStart = DateTime.Now.Minute;

            if (DateTime.Now.Minute == 0) // обнулення таймера
            {
                dateTimeStart = 0;
            }
        }
    }
}
```

# Скрипт Player

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class Player : MonoBehaviour
{
    public GameObject Svitylka;
    private Vector2 targetPos;
    public float Yincriment;
    [Header("Controls")]
    public float speed;
    public float maxHeight;
    public float minHeight;
    public Animator animator;
    [Header("Health")]
    public float health;
    public int numberofHearts;
    public Image[] hearts;
    public Sprite fullHeart;
    public Sprite emptyHeart;
    [Header("Shield")]
    public GameObject shield;
    public GameObject shieldIcon;
    public Shield shieldTimer;
    [SerializeField] public int coins;
    public GameObject coinSound;
    public GameObject shieldSound;
    [Header("The End")]
    public GameObject panel;

    public float time;

    private void Start()
    {
        Svitylka.SetActive(true);

        // Свайп
        SwipeManager.instance.MoveEvent += MovePlayer;
    }

    private void OnTriggerEnter2D(Collider2D other)
    {
        if (other.CompareTag("Coin"))
        {
            Instantiate(coinSound, transform.position, Quaternion.identity);
            animator.SetTrigger("Emotion 3");
            coins++;
            Destroy(other.gameObject);

        }

         if (other.CompareTag("Barrier"))
         {
             animator.SetTrigger("Emotion 4");
         }
    }
```

```
    private void Update()
    {
        // Здоров'я
        if (health < 1)
        {
            panel.SetActive(true);
            Destroy(gameObject);
            Destroy(shieldTimer.gameObject);
          Destroy(shieldIcon.gameObject);

        }
        if (health > numberofHearts)
        {
            health = numberofHearts;
        }
        for (int i = 0; i < hearts.Length; i++)
        {
            if (i < Mathf.RoundToInt(health))
            {
                hearts[i].sprite = fullHeart;
            }
            else { hearts[i].sprite = emptyHeart; }

            if (i < numberofHearts)
            {
                hearts[i].enabled = true;
            }
            else
            {
                hearts[i].enabled = false;
            }
        }

        // Переміщення
        transform.position = Vector2.MoveTowards(transform.position, targetPos, speed *
Time.deltaTime);

        // Активація щита
        if (coins == 3)
        {
            shieldIcon.SetActive(true);
        }
    }

    // Переміщення
    void MovePlayer(bool[] swipes)
    {
        if (swipes[(int)SwipeManager.Direction.Up] && transform.position.y < maxHeight)
        {
            targetPos = new Vector2(transform.position.x, transform.position.y +
Yincriment);
        }
        else if (swipes[(int)SwipeManager.Direction.Down] && transform.position.y >
minHeight)
        {
            targetPos = new Vector2(transform.position.x, transform.position.y -
Yincriment);

        }
    }
```

```
    // Щит
    public void Shield()
    {
        Instantiate(shieldSound, transform.position, Quaternion.identity);

        shieldIcon.SetActive(false);

        shield.SetActive(true);
        shieldTimer.gameObject.SetActive(true);
        shieldTimer.isCooldown = true;

        animator.SetBool("Emotion 1", true);
        animator.SetBool("Emotion 2", true);

        coins = 0;

    }
}
```

# Скрипт Barriers

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Barriers : MonoBehaviour
{
    public int damage = 1;
    public float speed;
    public GameObject effect;
    public GameObject sound;
    private void Update()
    {
        transform.Translate(Vector2.left * speed);  // швидкість переміщення об'єктів
    }
    private void OnTriggerEnter2D(Collider2D other)
    {
        if(other.CompareTag("Player")) // реалізація зіткнення з гравцем
        {
                Instantiate(sound, transform.position, Quaternion.identity);

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

            other.GetComponent<Player>().health -= damage; // завдавання шкоди гравцеві
            Destroy(gameObject);

        }
         if (other.CompareTag("Shield")) // реалізація зіткнення зі щитом
        {
            Instantiate(sound, transform.position, Quaternion.identity);

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

            Destroy(gameObject);
         }
    }
}
```

# Скрипт Coin

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Coin : MonoBehaviour
{
    public float speed;
    public GameObject effect1;

    public GameObject panel;
    private void Update()
    {
        transform.Translate(Vector2.left * speed); // швидкість переміщення об'єктів

        if (panel.activeSelf) // знищення об'єкту при активації меню програшу
        {
            Destroy(gameObject);
        }
    }
    private void OnTriggerEnter2D(Collider2D other)
    {
        if (other.CompareTag("Player")) // при зіткненні з гравцем викликається ефект
зірочок
        {
            Instantiate(effect1, transform.position, Quaternion.identity);

        }
        if (other.CompareTag("Shield")) // при зіткненні зі щитом викликається ефект зірочок
        {
            Instantiate(effect1, transform.position, Quaternion.identity);

        }
    }
}
```

# Скрипт «Spawner»

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Random = System.Random;

public class Spawner : MonoBehaviour
{
    private float timeBtwSpawn; // час між появами перешкод
    public float startTimeBtwSpawn; // старотовый інтервал появи перешкод
    public float decreaseTime;
    public float minTime = 1f; // час, після якого інтервал не стає вужче
    public List<GameObject> obstacles;
    public List<Transform> points;
    Random random = new Random();

    private void Update()
    {
        if (timeBtwSpawn <= 0)
        {
            var obstNum = random.Next(obstacles.Count); // рандомна поява
            Instantiate(obstacles[obstNum], points[obstNum].position, Quaternion.identity);
// задання небхідних об'єктів точкам

            timeBtwSpawn = startTimeBtwSpawn;

            if (startTimeBtwSpawn > minTime) //  зменшення часу звуження інтервалу
            {
                startTimeBtwSpawn -= decreaseTime;
            }
        }
        else
        {
            timeBtwSpawn -= Time.deltaTime;
        }
    }
}
```

# Скрипт «SpawnerCoins»

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Random = System.Random;

public class SpawnerCoins : MonoBehaviour
{
    private float timeBtwSpawn; //  час між появами
    public float startTimeBtwSpawn; //  старотовый інтервал появи перешкод

    public List<GameObject> obstacles;
    public List<Transform> points;

    Random random = new Random();

    private void Update()
    {
        if (timeBtwSpawn <= 0)
        {
            var coinNum = random.Next(obstacles.Count); // рандомна поява
            Instantiate(obstacles[coinNum], points[coinNum].position, Quaternion.identity);
// задання небхідних об'єктів точкам

             timeBtwSpawn = startTimeBtwSpawn;
        }
        else
        {
            timeBtwSpawn -= Time.deltaTime;
        }
    }
}
```

*Додаток Е*

# Допоміжні скрипти

*Додаток Е.1*

## Скрипт «Shield»

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class Shield : MonoBehaviour
{
    public float cooldown;
    [HideInInspector] public bool isCooldown;
    private Image shieldImage;
    private Player player;
    public Animator animator;

    void Start()
    {
        shieldImage = GetComponent<Image>();
        player = GameObject.FindGameObjectWithTag("Player").GetComponent<Player>();
        isCooldown = true; // ввімкнення таймера
    }
     void Update()
    {
        if(isCooldown) // реалізація таймера
        {
            shieldImage.fillAmount -= 1 / cooldown * Time.deltaTime;
            if(shieldImage.fillAmount <= 0) // вимикання таймера та допоміжних об'єктів
            {
                shieldImage.fillAmount = 1;
                isCooldown = false;
                player.shield.SetActive(false);

                gameObject.SetActive(false);

                animator.SetBool("Emotion 2", false);
                animator.SetBool("Emotion 1", false);
            }
        }
    }
    public void ResetTimer() // скидання таймеру
    {
        shieldImage.fillAmount = 1;
    }
}
```

*Додаток Е.2*

## Скрипт «Point»

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Point : MonoBehaviour
{
   public GameObject image;
```

```
  private void Start()
    {
        Instantiate(image, transform.position, Quaternion.identity); // поле для додавання
об'єкта
    }
}
```

*Додаток Е.3*

## Скрипт «SceneManagement»

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class SceneManagement : MonoBehaviour
{
    public GameObject sound;
    public void PlayGame() // перехід на сцену гри
    {
        Instantiate(sound, transform.position, Quaternion.identity);
        SceneManager.LoadScene("The Game");
    }
    public void QuitGame() // вихід з гри
    {
        Instantiate(sound, transform.position, Quaternion.identity);
        Application.Quit();
    }
    public void GoToMenu()// перехід на сцену головного меню
    {
        Instantiate(sound, transform.position, Quaternion.identity);
        SceneManager.LoadScene("Menu");
    }
}
```

*Додаток Е.4*

## Скрипт «Score»

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class Score : MonoBehaviour
{
    public int score;
    public Text scoreDisplay;
    private void Update()
    {
        scoreDisplay.text = score.ToString(); // підрахунок виводиться у заданому місці
текстом
    }
    private void OnTriggerEnter2D(Collider2D other)
    {
        if(other.CompareTag("Barrier")) // при зіткненні рахункового колайдера з перешкодою
додається пункт пройдених першкод
        {
            score++;
        } } }
```

*Додаток Е.5*

## Скрипт «Destroyer»

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Destroyer : MonoBehaviour
{
    private void OnTriggerEnter2D(Collider2D other) // при зіткненні рахункового колайдера з перешкодою чи монеткою об'єкт монетки чи перешкоди видаляється зі сцени
    {
        if (other.CompareTag("Barrier"))
        {
            Destroy(other.gameObject);
        }
        if (other.CompareTag("Coin"))
        {
            Destroy(other.gameObject);
        }
    }
}
```

*Додаток Е.6*

## Скрипт «SwipeManager»

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.EventSystems;
public class SwipeManager : MonoBehaviour
{
    public static SwipeManager instance;
    public enum Direction { Left, Right, Up, Down };
    bool[] swipe = new bool[4];
    Vector2 startTouch;
    bool touchMoved;
    Vector2 swipeDelta;
    const float SWIPE_THRESHOLD = 50;
    public delegate void MoveDelegate(bool[] swipes);
    public MoveDelegate MoveEvent;
    public delegate void ClickDelegate(Vector2 pos);
    public ClickDelegate ClickEvent;

    Vector2 TouchPosition() { return (Vector2)Input.mousePosition; } // визначення місця дотику
    bool TouchBegan() { return Input.GetMouseButtonDown(0); } // визначення початкового місця дотику
    bool TouchEnded()  { return Input.GetMouseButtonUp(0); }// визначення кінцевого місця дотику
    bool GetTouch() { return Input.GetMouseButton(0); } //  визначення дотику

    void Awake()
    {
        instance = this; // при вмиканні викликається класс
    }
    void Update() // перевірка кожного дотику
    {
        if (EventSystem.current.IsPointerOverGameObject()) return;
        // визначення старту і фінішу
        if (TouchBegan())
        {
```

```
            startTouch = TouchPosition();
            touchMoved = true;
        }
        else if (TouchEnded() && touchMoved == true)
        {
            SendSwipe();
            touchMoved = false;
        }
        // вимірювання відстані свайпа
        swipeDelta = Vector2.zero;
        if(touchMoved && GetTouch())
        {
            swipeDelta = TouchPosition() - startTouch;
        }
        //перевірка свайпа
        if(swipeDelta.magnitude > SWIPE_THRESHOLD)
        {
            if(Mathf.Abs(swipeDelta.x) > Mathf.Abs(swipeDelta.y))
            {
                // ліво/право
                swipe[(int)Direction.Left] = swipeDelta.x < 0;
                swipe[(int)Direction.Right] = swipeDelta.x > 0;

            }
            else
            {
                // вверх/вниз
                swipe[(int)Direction.Down] = swipeDelta.y < 0;
                swipe[(int)Direction.Up] = swipeDelta.y > 0;
            }
            SendSwipe();
        }
    }
    void SendSwipe() // реалізація свайпа
    {
        if (swipe[0] || swipe[1] || swipe[2] || swipe[3])
        {
            Debug.Log(swipe[0] + "|" + swipe[1] + "|" + swipe[2] + "|" + swipe[3]);
            MoveEvent?.Invoke(swipe); // перевірка на нульове значення
        }
        else
        {
            Debug.Log("Click");
            ClickEvent?.Invoke(TouchPosition()); // перевірка на нульове значення
        }
        Reset();
    }
    private void Reset()
    {
        startTouch = swipeDelta = Vector2.zero; // скидання позицій, станів
        touchMoved = false;
        for(int i = 0; i < 4; i++)
        {
            swipe[i] = false;
        }
    }
}
```