ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра комп‘ютерної інженерії

Пояснювальна записка
до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему:

«РОЗРОБКА МОБІЛЬНОЇ ГРИ ЖАНРУ 2Д РИБОЛОВЕЦЬКА ПРИГОДА ВИКОРИСТОВУЮЧИ UNITY 2020 НА МОВІ C#»

Виконав: студент 5 курсу, групи ППЗ-52
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)
Руденко О .О
(прізвище та ініціали)
Керівник **Н**егоденко О.В
(прізвище та ініціали)
Рецензент
(прізвище та ініціали)

Київ – 2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти - «Бакалавр»
Напрям підготовки -121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
О.В. Негоденко
"__" 2021 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ
Руденко Олегу Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка мобільної гри жанру 2д риболовецька пригода використовуючи Unity 2020 на мові C#»

Керівник роботи Негоденко Олена Василівна
затверджені наказом вищого навчального закладу від **12.03.** 2021 № 65.
2. Строк подання студентом роботи 01.06.2021
3. Вхідні дані до роботи:
У дослідженні було використанно середовище розробки Unity. Для створення спрайтів було використано програмне забезпечення Photoshop та Paint.net

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).
4.1 Аналіз та характеристики програмного продукту а також опис взаємодіґ компонентів та модулів.
4.2 Здійснити вибір розробки середовища та використання мов програмування
4.3 Здійснити вибір розробки середовища та використання актуальних мов
5. Перелік графічного матеріалу
1.Титульний лист
2. Об'єкт, предмет та мета дослідження
3. Порівняльна характеристика аналогів
4.Архітектура програмного додатку

6. Дата видачі завдання 19.04.21

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврсь коїроботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	19.04.21– 25.04.21	
2	Аналіз предметної області	25.04.21– 26.04.21	
3	Аналіз засобів програмної реалізації	27.04.21– 30.04.21	
4	Концепція та архітектура ПЗ	30.04.21– 05.05.21	
5	Вступ, висновки, реферат	05.05.21– 07.05.21	
6	Розробка презентації	07.05.21– 20.05.21	
7	Здача роботи	01.06.21	

Студент _____ Руденко Олег Олександрович
(підпис) (прізвище та ініціали)

Керівник роботи _____ Негоденко Олена Василівна
підпис) (прізвище та ініціали)

РЕФЕРАТ

РОЗРОБКА МОБІЛЬНОЇ ГРИ ЖАНРУ 2Д РИБОЛОВЕЦЬКА ПРИГОДА ВИКОРИСТОВУЮЧИ UNITY 2020 НА МОВІ C#

Об’єкт сфери дослідження – розробка гри для смартфонів.

Предмет дослідження – технології розробки 2-д гри для смартфону жанру пригода

Мета дослідження – розробити прототип мобільної гри жанру 2D-пригода засобами середовища Unity

Для того щоб досягнути поставленого завдання необхідно:
1)Вибрати жанр та платформу для гри
2) Придумати сценарій та концепцію основних елементів гри
3) Обрати та вивчити засіб реалізацій проекту
4)Підготувати необхідні для гри файли
5)Реалізація гри

ЗМІСТ

УМОВНІ ПОЗНАЧЕННЯ

UML - Unified Modeling Language

IDE- інтегроване середовище розробки

ВСТУП

Об’єкт сфери дослідження – розробка гри для смартфонів.

Предмет дослідження – технології розробки 2-д гри для смартфону жанру пригода

Мета дослідження – розробити прототип мобільної гри жанру 2D-пригода засобами середовища Unity

Для того щоб досягнути поставленого завдання необхідно:
1)Вибрати жанр та платформу для гри
2) Придумати сценарій та концепцію основних елементів гри
3) Обрати та вивчити засіб реалізацій проекту
4)Підготувати необхідні для гри файли
5)Реалізація гри

Відеоігри – це один з видів активного дозвілля багатьох людей по всьому світу. Діти та дорослі трактують відеоігри як форму проведення вільного часу, що рівноцінне читанню, перегляду фільму або відвідуванню театру. Високоякісні сучасні відеоігри виконують не лише розважальну функцію, а й навчальну та виховну. Тому, попит на розважальне програмне забезпечення спонукає видавців відеоігор видаватиїх ще більше та більше

1.АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність теми

Індустрія відеоігор виникла порівняно нещодавно, близько 30 років тому , але незважаючи на це їй уже вдалося перетворитися на велику галузь з доходом у кілька мільярдів доларів на рік.Пояснити таке раптове зростання популярності віртуальних розваг дуже просто: вона залежить від широкого використання комп'ютерних технологій, в тому числі пов'язана з появою Інтернету. Завдяки цьому ,навідмінно від інших видів розваг, комп'ютерні ігри є більш доступними для кінцевого користувача.

Для гри гравець просто потребує комп'ютера або ігрової приставки та копії самої гри .Завдяки мережі інтернет вам не потрібно виходити з дому, щоб отримати копію гри. Крім того, споживачеві не потрібно мати спеціальних знань для того щоб обрати гру яка підходить саме йому,в той час як для інших розваг необхідно розбиратися як мінімум в необхідному обладнання.

Слід також зазначити, що останнім часом ігри перестають бути лише засобами для відпочинку та розваг . Наприклад, сьогодні завдяки використанню ігрових технологій створюються спеціальні імітаційні комплекси, які необхідні для підготовки фахівців у різних галузях: від лісоруба до пілота.

На жаль, у Україні все трохи інакше. Ігрова індустрія в Україні як і в більшості інших пострадянських країн дуже слабо розвинена. Це тому, що культура комп'ютерних розваг прийшла занадто пізно і почала розвиватись зовсім нещодавно .Через це , незважаючи на відносно великий попит, ми маємо дуже малу кількість компаній-розробників, здатних конкурувати з іноземними компаніями.

Тому розвиток технологій у цьому напрямку можна розглядати як одну з найперспективніших, особливо в нашій країні

1.2 Аналіз та характеристика аналогів

Для розробки гри необхідно спочатку продумати концепцію та прототип.Для цього спочатку ознайомимся з аналогами мого проекту

1)Ace Fishing

Рис 1.2.1 .Гемплей Ace Fishing

Ace Fishing - це один з найпопулярніших додатків. У ньому є налаштування для підбору снастей, поїздки, лову декількох риб і т. Д. Також в програмі встановлено таблиця лідерів, в ній можна порівняти досягнення зі світовими значеннями. Грати тут складно, але механіка приносить задоволення, тим більше що в додатку присутня пара основних тактик риболовлі, необхідних для полювання.

Гравець відвідує екзотичні місця, намагається виловити всі види риб, які там живуть. Улов можна виростити або продати дорожче. Але для цього потрібно набратися терпіння. Як і в реальному житті, мешканці морських глибин не ловляться миттєво

За рідкісні і великі екземпляри платять більше грошей. Але і зловити їх складніше. Для цього потрібно найкраще обладнання і удача. Втім, неможливо постійно задовольнятися лише дрібними рибками. Рано чи пізно це набридне. А який справжній рибалка не побажає випробувати себе в сьогоденні змаганні з представниками фауни.

Отримані гроші можна витрачати на придбання кращих снастей. В асортименті є різна кількість приманок, лісок і ін. Багато з них є одноразовими, а інші треба ремонтувати і теж на них витрачати накопичені кошти. Чим кращі результати, тим швидше можна поліпшити свою екіпіровку і рухатися далі.

В додаток вбудований лічильник витрати енергії. Одна рибалка триває не більше

півгодини. Так що завжди є можливість перепочити, перш ніж продовжити.

У певні моменти потрібно швидко натискати на клавіші, підсікати і витягувати рибу. Реалістичність знаходиться на гідному рівні. Граючи можна відчути хвилювання, виловлюючи великого або рідкісного мешканця морських глибин. Це якраз те, що потрібно, якщо сьогодні не вистачає часу вибратися на справжній водойму.

2)Let's Fish

Рис 1.2.2.Гемплей Let's Fish

Let's Fish - це один з найбільших додатків про риболовлю. У ньому представлено понад 60 озер і 650 видів риб, безліч снастей для лову . На відміну від конкурентів, ця програма має багато користувачів та режими: змагання, турніри і таблицю лідерів.

3)Rapala Fishing

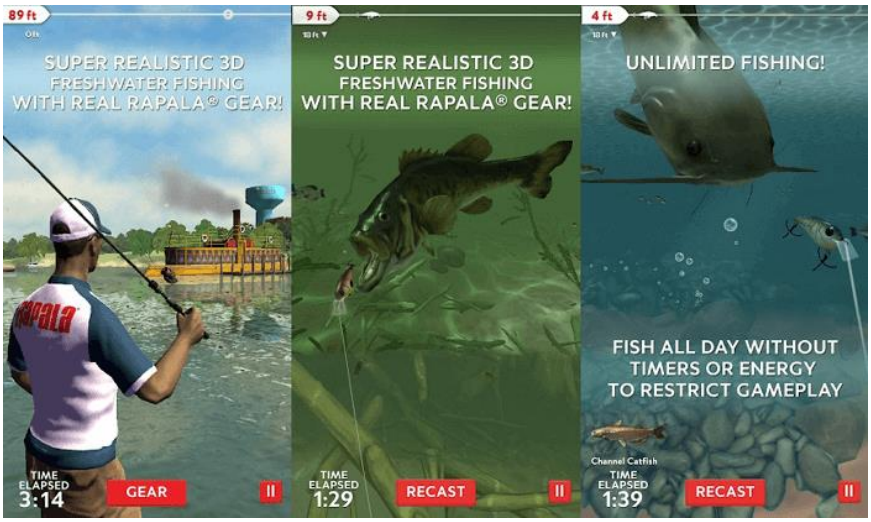


Рис 1.2.3 .Гемплей Rapala Fishing

Rapala Fishing - це гра в стилі аркади, розроблена компанією Concrete Software з Rapala. Вона націлена на реалістичний досвід. Гра не ідеальна, але непогана. Розробники можуть похвалитися тим, що в ній немає енергії для риболовлі, таймерів, а лише проводяться щоденні турніри.

Присутня реалістична графіка з насиченими і змішаними квітами. Як і в інших варіантах, значна увага приділена підводним сценам. Виконані вони на гідному рівні: від витонченого руху морських мешканців до бульбашок повітря і помутнілої від мулу води. Звуковий супровід також органічно вписується в ігровий процес.

Риболовля відбувається в цікавих локаціях. Можна вибрати одиночне подорож або турнір, а також ознайомитися з результатами гравців. При індивідуальній риболовлі виконуються певні завдання на час.

Також є режим вилову певних видів. Він підходить для новачків, охочих дізнатися більше про морську фауну. Піймавши будь-якої тип, можна дізнатися про нього чимало цікавого. Цілком можливо, що ця інформація стане в нагоді і в реальному житті.

С початку закидають приманку. Як тільки вона досягає поверхні води, камера перемикається на підводну зйомку. Коли риба проковтне гачок, потрібно намотувати волосінь: обережніше (якщо робити це слабо, то видобуток може вислизнути) і не сильно швидко (натяг може привести до обриву оснастки).

Щоб приманка зацікавила рибу, потрібно смикати нитку в різні боки. Коли жертва заковтує гачок, то починає активно намагатися звільнитися. Тут і починається найцікавіше.

І ще одна важлива деталь - кожен улов приносить гроші.

4)Fishing Hook

Рис 1.2.4 .Гемплей Fishing Hook

Fishing Hook - це одна з найбільш реалістичних ігор, але вона досить проста. У ній проводять більшу частину часу на віртуальному човні з вудкою в воді. Механіка трохи дивна, але нескладна. Спійману рибку можна відпустити, продати або залишити собі. У кожної особини своя вартість, яка залежить від виду, довжини і ваги. У програмі також є досягнення, ранжування і підтримка до 16 мов.

5)iFishing

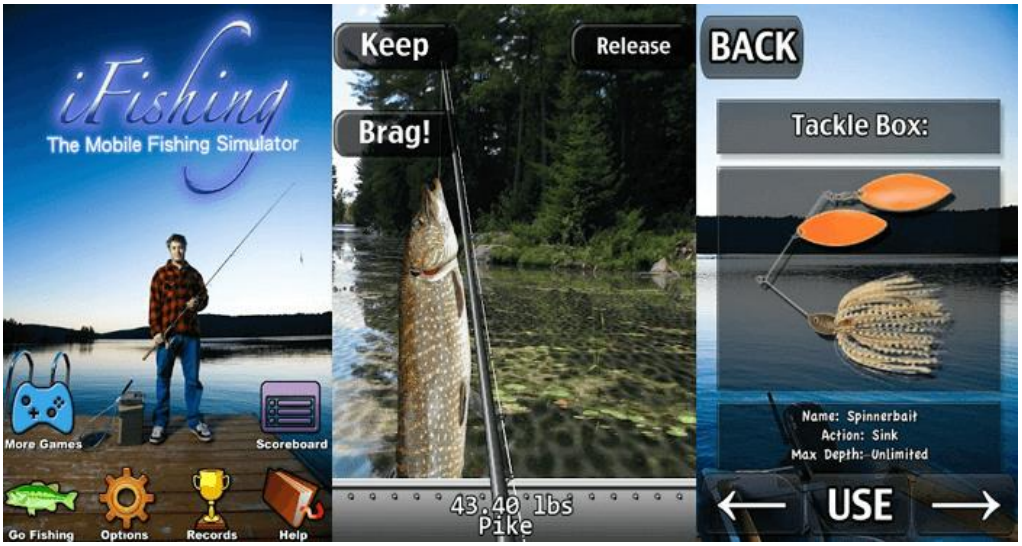


Рис 1.2. 5 .Гемплей iFishing

iFishing - цей мобільний симулятор дозволяє відчути себе рибалкою, перетворюючи смартфон в вудку. У ньому також присутні багато індикаторів, що вказують, де є риба, куди закидати мережі, яку приманку використовувати, коли давати особинам

перепочинок.

Як тільки починається гра, користувач відправляється на турнір. Його головна мета - це виловити якомога більше риби. Потім відбувається зважування улову, від ваги якого залежить місце в турнірній таблиці.

Для риболовлі можна вибрати час і озеро. У додатку є 15 водойм зі всілякими пейзажами.

Коли користувач вперше вирушає на озеро, йому потрібно розвідати рибні косяки радіолокатором, розташованим під човном. Але в основному варто триматися подалі від глибоководних вирів. Часто хороша ловля буває біля берега.

Цікаво зроблені елементи управління. Щоб закинути гачок потрібно натиснути і утримувати кнопку кидка, одночасно хитаючи телефон, і відпустити, щоб розмотати волосінь. Тут потрібно бути обережним і стежити за чистотою рук. Якщо вони слизькі, смартфон може полетіти в стіну.

Повзунок у нижній частині екрана контролює швидкість намотування нитки, її можна потягнути, керуючи смартфоном (до себе або убік). Нитка буде рухатися відповідно телефону.

Також можна вибрати глибину занурення приманки, тип снастей і інші параметри, які впливають на те, кого пощастить зловити. В основному на ловлю йде щука, судак і окунь.

Одна з головних особливостей програми це його різноманіття. Процес не обмежується закиданням гачка і витягуванням улову. Спочатку можна покататися на човні по озеру, зупинитися в одному місці і тільки потім перейти до лову риби.

Звуковий супровід відповідає тому, що робиться в даний момент. Коли проводять розвідку на водоймі радіолокатором, то чується гул мотора. Сидячи з вудкою, можна почути спів птахів, крякання качок. У бібліотеці є звуки, що закидається волосіні, накручуваної котушки. Все це робить ігровий процес дуже реалістичним.

6) Master Bass Angler

Рис 1.2. 6 .Гемплей Master Bass Angler

На відміну від інших симуляторів, Master Bass Angler не змусить довго чекати, щоб щось зловити. Риба клює відразу після того, як закидається оснащення. У симуляторі є кілька місцевостей з усього США, різні відновлення інвентарю. Існує багато користувачів версія для змагання з іншими гравцями.

7)Ridiculous Fishing

Ridiculous Fishing оснащена ретро-графікою і простою механікою. Тут кидають свою приманку і намагаються зачепити стільки риби, скільки можна витягнути. Є також предмети, капелюхи, забавні і абсурдні варіанти снастей. Єдиним недоліком є те, що останнє оновлення додатка було в 2015 році.

Ця версія відрізняється від попередніх. У ній спочатку закидають гачок в воду і виловлюють рибу. Як тільки вона досягає поверхні, її потрібно підкинути в повітря і розстріляти. За кожну знищену таким чином особина нараховуються гроші. Для цього використовують пістолети, дробовики, міні-гармати, базуки і інше потужна зброя.

Для знищення багатьох риб потрібно кілька пострілів. Тому бажано клікати швидше, інакше видобуток встигне впасти назад у воду і поплисти. Вони навіть відлітають крізь хмари і повз Місяця в космічний простір.

Купівля нових лісок дає можливість закидати приманку більш глибоко, щоб зловити велику і цінну рибу. Бур дозволяє розрізати рибу та інші перешкоди під час спуску. За це теж отримують винагороду.

Як рибалки виступає хлопець на ім'я Біллі. Він закидає наживку на глибину, потім

намагається одночасно витягнути на поверхню якомога більше риби. Коли гачок рухається вниз, гаджет потрібно нахиляти так, щоб огинати риб і досягти максимальної глибини.

8)Fishing Break

Рис 1.2.7 .Гемплей Fishing Break

У Fishing Break гравці ловлять сотні риб з восьми різних місць. Також існують досягнення, лідери і швидкий, але складний режим. Графіка проста, але на рівні. Насправді реалізм може зробити це додаток гірше. У таку рибну ловлю зазвичай грають, чекаючи черги в магазині або поки пройде реклама по телевізору. Не варто очікувати від неї чогось більшого.

Хоча ця програма має агресивну стратегію монетизації, вона виглядає гідна. Можливо, це виглядає дивно для 2D на мобільному пристрої, але з нею дійсно можна зануритися в захоплюючу історію. Всчі спрайт оброблені в стилі аніме.

Щоб витягнути рибу, потрібно постійно регулювати силу натягу волосіні. При її ослабленні, особина зірветься з гачка, а при надмірному натягу лопне нитку. Особливо важко регулювати ведення оснастки на вищих рівнях, коли доводиться виловлювати велику рибу. Ігровий процес поступово стає складним.

Але в цілому багато чого зрозуміло навіть без тривалого навчання:

закинути волосінь;

підвести гачок до риби;

дати їй заковтнути наживку.

У додатку немає складних алгоритмів, які потрібно запам'ятати. Відсутні терміни,

характеристики риб, особливості різних водойм. Потрібно просто натискати на кнопку і проводити по екрану пальцем. Ось і все управління.

Як правило, риба на гачку натягує вудку сильніше. Тому необхідно дотримуватися «золоту середину», дати їй видихнути, поступово натягуючи і послаблюючи нитку. Коли рибка знесилиться, її витягують на поверхню, фотографуються з уловом і підраховують зароблені гроші.

9) Fishing Diary

Рис 1.2.8 .Гемплей Fishing Diary

У Fishing Diary є знаряддя і рибальські снасті. Завдання рибалки - зібрати більше риби. Щоб полегшити полювання, в додатку встановлені бонуси.

Отже на телефон та смартфон доступно безліч ігор про риболовлю.Але 2д ігор зовсім мало.Тому було вирішено розробити саме такий додаток.

2. ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ ЗАСТОСУНКУ

2.1 Загальний алгоритм реалізації проекту

Що ж , почнемо з алгоритму розробки гри .Він мало чим відрізняється від алгоритму розробки будь-якого іншого програмного продукту і включає в себе такі етапи життєвого циклу як :

1) проектування;

2) розробка;

3) видання і підтримка.

На етапі проектування визначаються мета гри і засоби її розробки.При визначенні мети визначаються ідея, жанр гри. ідея- це те, що буде спонукати гравця грати в створювану гру, і вона дуже тісно пов'язана з жанром.

Таким чином, визначивши основні ідеї гри, жанр буде підібраний практично відразу. Визначившись з жанром і ідеєю гри, наступним кроком буде вибір сетінг. Сеттінг - це те де буде відбуватися сама гра .Він може сильно полегшити розробку сценарію для гри, тому його краще вибирати заздалегідь і грунтуючись на смаки цільової аудиторії.

До засобів розробки в першу чергу відносять написання програмного коду та вибір ігрового рушія. Від його вибору залежить як швидкість самої розробки, так і працездатність самого продукту надалі. Код в першу чергу залежить від платформи, для якої буде створюватися гра.

Ігровий рушій відповідає за опис фізики об'єктів, рендер графіки і інше.

2.2 Вибір жанру гри

Розділяють 8 основних жанрів відеоігор, які у свою чергу діляться ще на декілька піджанрів.

1).Action - жанр комп'ютерних ігор, в яких успіх гравця у великій мірі залежить від його швидкості реакції і здатності швидко приймати тактичні рішення. Дія таких ігор розвивається дуже динамічно і вимагає напруження уваги і швидкої реакції на що

відбуваються в грі події. При цьому в якості основного засобу прогресу в грі, як правило, використовують будь-яку зброю. Поділяється на шутери, аркади, файтинги та стелс-ігри. Цей жанр є найпопулярнішим та займає близько 70% ринку відеоігор.

3D-шутер (англ. 3D Shooter) - в іграх даного типу гравець, як правило , діючи поодинці , повинен знищувати ворогів за допомогою зброї ближнього бою (як правило холодного) і стрілецької зброї (найчастіше вогнепальної зброї та енергетичної) , для досягнення певних цілей на даному рівні , зазвичай , після досягнення заданих цілей гравець , переходить на наступний рівень. Ворогами часто є: бандити (напр. Max Payne) , нацисти (напр. в Return to Castle Wolfenstein) та інші «погані хлопці», а також всілякі інопланетяни, мутанти і монстри (напр. Doom , Half- life , Duke Nukem 3D , Serious Sam).Існують шутери від першї особи. У шутерах від першої особи (англ. First person shooter, FPS) гравець не бачить персонажа з боку - він спостерігає за подіями від особи персонажа - «очима персонажа» (англ. First person look), і спостережувана гравцем картина збігається з тим, що «бачить» персонаж.

У шутерах від третьої особи (англ. Third person shooter, TPS) гравець бачить персонаж з боку з фіксованою (зазвичай зі спини) або довільної точки зору (англ. Third person look). У ряді ігор реалізована можливість перемикання перший / третя особа і фіксована / довільна камера.

Фа́йтинг (від англ. Fighting - бій , бійка , поєдинок , боротьба) - жанр комп'ютерних ігор, що імітують рукопашний бій малого числа персонажів в межах обмеженого простору , званого ареною. Файтинг близькі до ігор жанру «побий їх усіх», проте між ними існують відмінності. Так, в більшості файтингов гравцеві не потрібно переміщатися по довгому рівню і не можна вийти за межі арени, а бій складається з непарного числа окремих раундів і не є безперервним. Менш значними і необов'язково присутніми ознаками жанру є використання численних шкал для зображення життєво важливих показників персонажів і промальовування бійців на арені в профіль.

Слешер англ. Slasher від англ. Slash - рубати , різати) або Чоппер (англ. Chopper , буквально рубилово, від англ. Chop - рубати) - особливий різновид комп'ютерних та відеоігор, дуже схожа на жанр побий їх усіх, але спеціалізується на холодному зброю. Найпопулярніші слешери розробляються насамперед для ігрових консолей, так як геймпад набагато більш підходить для даного типу гри - не більше десяти активно використовуються кнопок та їх комбінації роблять повнорозмірні клавіатури комп'ютерів

громіздкими і надлишковими, а двох -трьох кнопкові миші - недостатніми.Не слід плутати Slasher с Hack and slash, визначення Hack and slash є більш широким і в нього крім слешерів потрапляють також ігри типу Diablo і Dungeon Siege , які слешером не є.

Аркада (англ. Arcade) - гра , в якій гравцеві доводиться діяти швидко , покладаючись в першу чергу на свої рефлекси і реакцію. Ігровий процес простий і не змінюється протягом гри. Аркади характеризуються розвиненою системою бонусів : нарахування очок , поступово відкриваються елементи гри і т. д.Термін « аркада » по відношенню до комп'ютерних ігор виник у часи ігрових автоматів , які встановлювалися в торгових галереях (arcades) . Ігри на них були простими в освоєнні (щоб залучити побільше граючих) . Згодом ці ігри перекочували на ігрові приставки (консолі) і до цих пір є основним жанром на них.

Стелс-екшен (англ. Stealth action) - це жанр комп'ютерних ігор, в яких гравцеві потрібно непомітно переміщатися, ховатися, таємно і непомітно вбивати ворогів, і уникати виявлення, щоб виконати місію. Незважаючи на те, що цей жанр порівняно старий, на даний момент існує не так багато ігор, що класифікуються як стелс-екшени. Більш часто зустрічаються ігри, що містять елементи стелса. Найбільшу популярність жанр отримав завдяки серіям ігор Thief, Metal Gear, Hitman, Splinter Cell.

2).Симулятор - імітатори , механічні або комп'ютерні , що імітують управління яким-небудь процесом , апаратом або транспортним засобом. Найчастіше зараз слово « симулятор » використовується стосовно до комп'ютерних програм (зазвичай іграм). За допомогою комп'ютерно -механічних симуляторів , абсолютно точно відтворюють інтер'єр кабіни апарату , тренуються пілоти , космонавти , машиністи високошвидкісних поїздів.Симулятор - програмні та апаратні засоби , що створюють враження дійсності , відображаючи частину реальних явищ і властивостей у віртуальному середовищі .

3).Стратегії (англ. Strategy) - ігри, що вимагають планування і вироблення певної стратегії для досягнення якоїсь конкретної мети , наприклад , перемоги у військовій операції . Гравець управляє не одним персонажем , а цілим підрозділом , підприємством або навіть всесвіту. Розрізняють походовом або покрокові стратегічні ігри (Turn- Based Strategy , TBS) , де гравці по черзі роблять ходи , і кожному гравцеві відводиться необмежену або обмежену (залежно від типу і складності гри) час на свій хід , і стратегічні ігри в реальному часі (Real Time Strategy , RTS) , в яких всі гравці виконують свої дії одночасно, і хід часу не переривається.

4).Пригода (англ. Adventure) - гра-розповідь у якій керований гравцем герой просувається по сюжету і взаємодіє з ігровим світом за допомогою застосування предметів, спілкування з іншими персонажами і рішення логічних задач.

5).Музична гра (англ. Rhythm game , rhythm action) - жанр комп'ютерних ігор , де наріжним ставиться музична складова , а від гравця потрібно наявність почуття ритму.Ігри даного жанру беруть за основу танці або виконання групою музичних композицій. Гравці повинні відповідно до того , що демонструється на екрані , натискати певні кнопки або виконувати танцювальні рухи. У разі успіху нараховуються очки . Багато музичні ігри також пропонують багато користувачів режими , де гравці або б'ються один з одним на рахунок , або грають в одній команді , уособлюючи собою групу.Деякі ігри жанру дозволяють використовувати стандартні геймпади або комп'ютерної миші, як , наприклад у грі osu ! , Однак , більшість з них вимагають спеціальні контролери , виконані у вигляді музичних інструментів. Для багатьох танцювальних ігор випускають особливі килимки з реагують на натискання областями.

6).Комп'ютерна рольова гра (англ. Computer Role - Playing Game (CRPG або RPG) - жанр комп'ютерних ігор , заснований на елементах ігрового процесу традиційних настільних рольових ігор. У рольовій грі гравець управляє одним або декількома персонажами , кожен з яких описаний набором чисельних характеристик , списком здібностей і вмінь ; прикладами таких характеристик можуть бути хіт- поінти (англ. hit points , HP) , показники сили , спритності , захисту , ухилення , рівень розвитку того чи іншого навику і т.п. У ході гри вони можуть змінюватися. Одним з характерних елементів ігрового процесу є підвищення можливостей персонажів за рахунок поліпшення їх параметрів та вивчення нових здібностей .

7).Пазли (англ. Puzzle) та інші. У некомп'ютерною головоломці роль арбітра, що стежить за дотриманням правил, грає або сам гравець (пасьянс), або деякий механічний пристрій (кубик Рубіка). З появою комп'ютерів можливості головоломок розширилися, так як написати комп'ютерну програму простіше, ніж сконструювати механічний пристрій.Головоломки, як правило, не вимагають реакції від гравця (проте багато ведуть рахунок часу, витраченого на вирішення).

8).Interactive fiction (IF, буквальний переклад - інтерактивна література; текстові квести; adventure - пригодницька гра) - жанр комп'ютерних ігор, в якому спілкування з гравцем здійснюється за допомогою текстової інформації. Розвиток цього жанру, у

зв'язку з низьким вимогою до ресурсів, почалося досить давно, і не припинилося навіть з появою графічних ігор. Існують два види інтерфейсу - інтерфейс з введенням тексту з клавіатури або інтерфейс у вигляді меню, де гравець вибирає дію з декількох запропонованих (англ. CYOA, Choose Your Own Adventure).
Найбільш цікавим жанром є жанр пригода

2.3 Вибір середовища розробки

Ігровий рушій – це готова архітектура, яку розробники використовують для розробки та запуску гри

Середньостатистичний ігровий рушій надає розробникам спосіб додавати такі речі, як:

1)фізика

2)керування

3)рендерінг

4)необхідні скрипти

5)виявлення колізій

6)штучний інтелект і т.п. без необхідності їх програмування Отже проаналізувавши ринок виділив для себе 3 ігрових рушія Unity, Unreal та Godot.Почнемо з аналіу кожного з них:

1) **Unity**

Рис 2.3.1. логотип Unity

Unity - найпопулярніший ігровий рушій на сьогоднішній день. З'явився в 2005 році. Велике поширення отримав з виходом версії Unity3d. На цьому рушії зроблено велику

кількість мобільних ігор та ігор в steam. У тому числі і інді.

У Unity найбільша кількість розробників, у більшості компаній процеси заточені саме під Unity (так як там використовується мова C #). Крім того готових асетів у рушія приблизно стільки ж, скільки сумарно у всіх інших розглянутих в цій статті. Ассет - це готові рішення та матеріали : 3d моделі, картинки, рівні і навіть цілі і гри які можна використовувати в своїй грі..

Недоліки та переваги рушія

Переваги:

1) Найбільше ком'юніті

2) Вбудована реклама та аналітика від самої Unity

3) Мультиплатформеність

4) Наявність безкоштовної версії

5) Найбільший вибір асетів

6) Найбільший вибір навчальних матеріалів, курсів і книг

7) C #. Потужний і зрозумілий

8) Сам рушій має великі можливості як в 2d, так і в 3d

Мінуси:

1) У порівнянні з іншими рушіями не надто зручна структура і інтерфейс

2) Файли на виході важать багато.

3) Помилки, які тягнуться з версії у версію

4) Навіть при наявності інструменту візуального програмування Playmaker і великої кількості асетів для зовсім новачків рушій буде занадто складний

5) 3d графіка непогана, але таки пристойно поступається Unreal.
2) Unreal Engine

Рис 2.3.2. логотип Unreal Engine

Unreal Engine - ігровий рушій компанії Epic Games, орієнтований на AAA-проекти і проекти в 3D. Він умовно безкоштовний при некомерційному застосуванні, але якщо проект приносить більше $ 3 тис. на квартал - розробникам потрібно платити авторські відрахування в розмірі 5% від виручки. Перше, про що варто згадати в огляді UE, це, звичайно, графічний потенціал, що включає підтримку навіть DirectX 12. Не можна забувати про платформ цього SDK: на ньому можна створювати ігри для ПК, консолей, смартфонів, планшетів.

Unreal дає розробнику великий набір простих в освоєнні і інтуїтивно зрозумілих інструментів. C ++ накладає мінімум обмежень під час написання скриптів, а система візуального програмування Blueprint полегшує прототипування або написання скриптів руками новачків. Створювати елементи гри можна наочно, переміщаючи об'єкти, без ручного введення коду.

Плюси та мінуси

Плюси:

1) Відмінна графіка

2) Хороші інструменти для моделювання рівнів

3) Велике ком'юніті і велика кількість туторіали

4) Візуальний скриптинг Blueprints з коробки

5) Безкоштовний

6) Відмінний набір готових Ассетів . Менше ніж у Unity, але все якісні

Мінуси :

1) Займає багато місця сама програма і її гри

2) Високий поріг входу, який частково знижено візуальним програмуванням

3) Інтерфейс досить громіздкий з десятками елементів. Хоча якщо звикнути - логічніше ніж в Unity

4) Є проблеми з продуктивністю

3)Godot

Рис 2.3.3. логотип Godot

Godot Engine - відкритий багатоплатформовий 2D і 3D ігровий рушій під ліцензією MIT, який розробляється спільнотою Godot Engine Community. До публічного релізу у вигляді відкритого ПЗ рушій використовувався усередині деяких компаній Латинської Америки. Середовище розробника працює на Linux, OS X, Windows, BSD і Haiku і може експортувати ігрові проекти на ПК, консолі, мобільні і веб-платформи. Отже проаналізувавши 3 рушії найбільш зручним для новачка в розробці ігор є Unity

3.МОДЕЛЮВАННЯ ТА РОЗРОБКА ЗАСТОСУНКУ

3.1 Постановка задачі та опис можливостей гри

Для аналізу та розробки гри створенно UML-діаграми

UML використовується на кожному процесі життєвого циклу. UML підтримує багато видів діаграм, такі як:

1. Class Diagram.
2. Package Diagram.
3. Component Diagram.
4. Deployment Diagram.
5. Collaboration Diagram.
6. Object Diagram.
7. Composite Structural Diagram.

Мова UML являється простим концептуальним рішенням для розробки різних застосунків. Процес об’єктно – орієнтованого аналізу з використанням всіх можливостей включає одну важливу особливість : включати в діаграму потрібно тільки ті окремі моделі, які є об’ємними та важливими у порівнянні з тими, що також існують у проекті, тільки вважаються другорядними, щоб не навантажувати процес аналізу.

Були створенні такі діаграми як діаграма прецендентів та діаграма діяльності . Діаграми прецедентів дозволяють візуалізувати поведінка системи, підсистеми або класу, щоб користувачі могли зрозуміти, як їх використовувати, а розробники - реалізувати відповідний елемент.

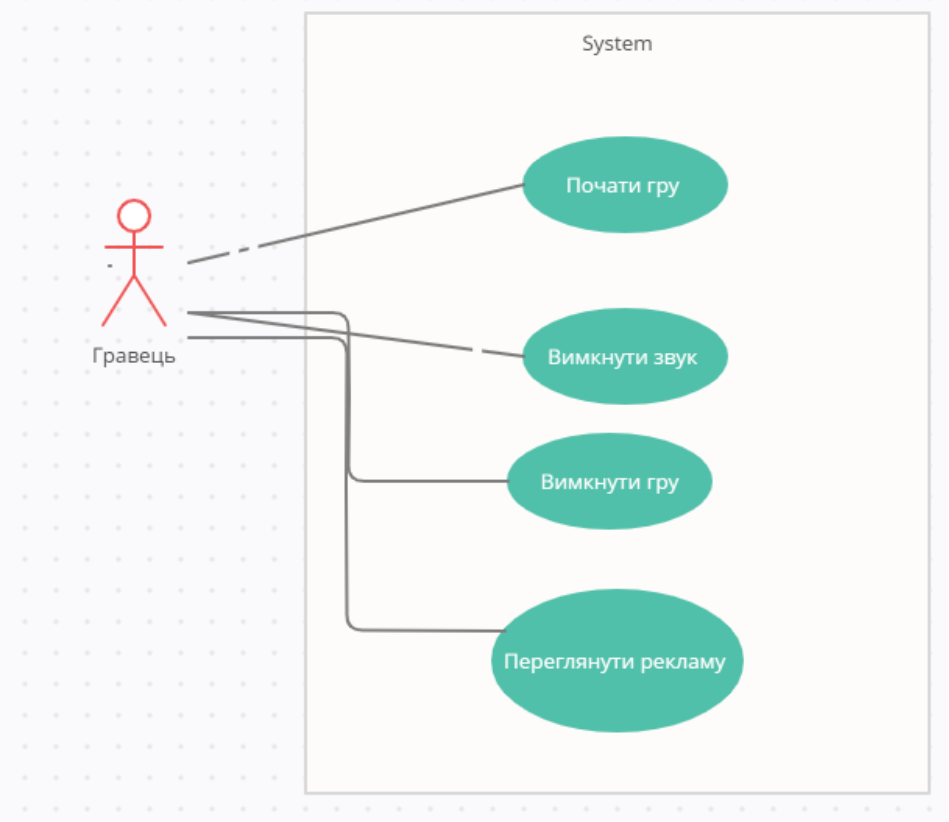


Рис 3.1.1. діаграма прецендентів

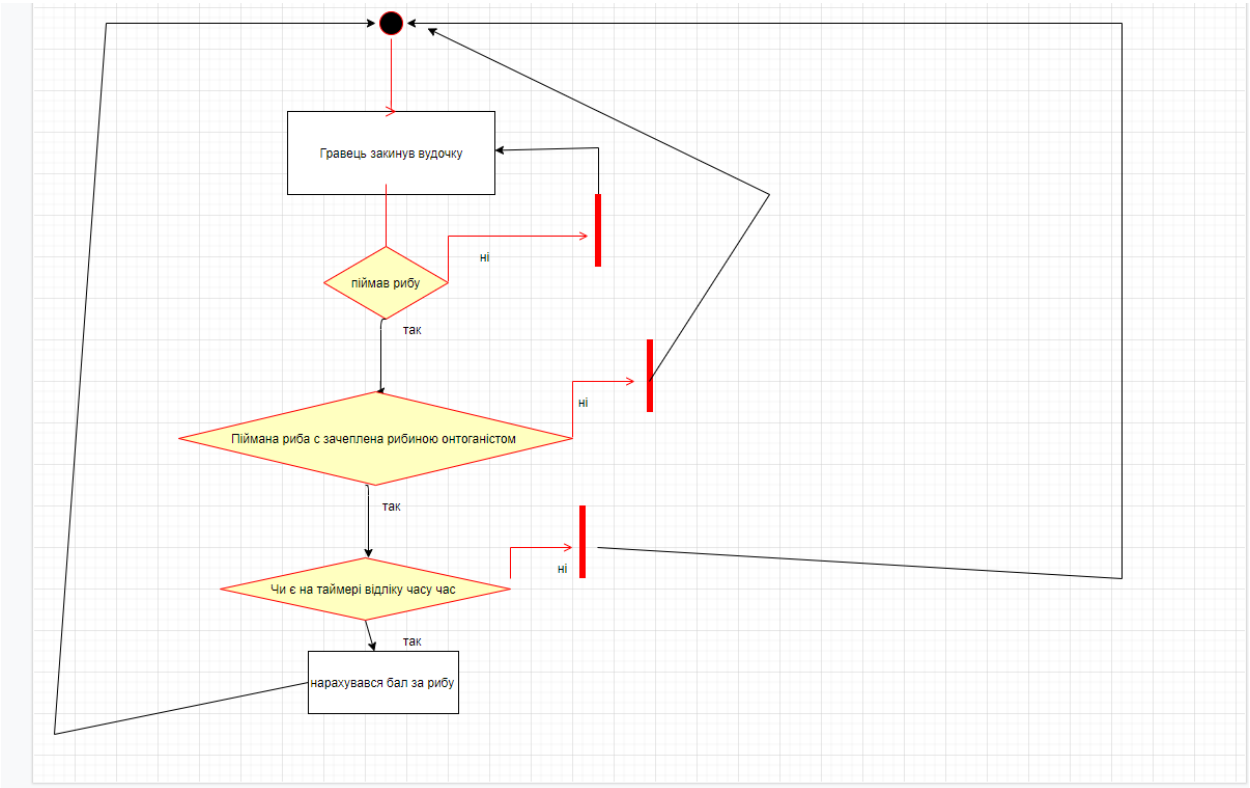


Рис 3.1.2. . діаграма діяльності

3.2 Розробка гри

Розробка кожного за стосунку починається з розробки його прототипу Розробимо прототип та задумку самої гри.За задумко. Головний персонаж -Жорж закидає вудочку в річку та ловить рибу.Він бажає спіймати рибу своєї мрії

Спочатку розробимо прототипи дизайну головних меню та гемплею

Рис 3.2.1. Макет гейплеюю

На основі даного макету було зроблено таке меню

Рис 3.2.2.Головна сцена продукту.

Далі я почав малювати самі моделі та писати код даного розділу.Ось що вийшло –

Рис 3.2.3.Головний персонаж

Також були розроблена сама локація (бекграунд)

Рис 3.2.4 Головна локація

Були реалізовані Моделі самих риб

Рис 3.2.5.Спрайт риби онтогоніста

Вихідний код даного персонажу

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

public class Blackfish : MonoBehaviour {
 public bool up;
 //public float Speedup;
 public bool down;
 //public float Speeddown ;
 public float SPEED;
 public float STOP;
 public float HighSpeed;
 public float SD;
 public float SU;


 public GameObject MainCam;

 public bool Hunt;
 // Use this for initialization
 void Start()
 {
        up = false;
        down = true;
 }

 // Update is called once per frame
 void Update()
 {
        MainCam = GameObject.Find ("Main Camera");
        if (Hunt == false) {
            if (transform.position.x >= MainCam.transform.position.x + 5f) {
```

```
            transform.localScale = new Vector3 (-1, 1, 1);
            up = false;
            down = true;
        }
        if (transform.position.x <= MainCam.transform.position.x - 5f) {
            transform.localScale = new Vector3 (1, 1, 1);
            up = true;
            down = false;
        }
        if (up == true) {
            SU += SPEED * Time.deltaTime;
            if (SU >= HighSpeed) {
                SU = HighSpeed;
            }
            transform.position += new Vector3 (SU * Time.deltaTime, 0, 0);
        }

        if (up == false) {
            SU -= STOP * Time.deltaTime;
            if (SU < 0) {
                SU = 0;
            }
            transform.position += new Vector3 (SU * Time.deltaTime, 0, 0);
        }
        ////////////////////////////////////////////////////////////////////////


        if (down == true) {
            SD += SPEED * Time.deltaTime;
            if (SD >= HighSpeed) {
                SD = HighSpeed;
            }
```

```
                transform.position -= new Vector3 (SD * Time.deltaTime, 0, 0);
            }

            if (down == false) {
                SD -= STOP * Time.deltaTime;
                if (SD <= 0) {
                    SD = 0;
                }
                transform.position -= new Vector3 (SD * Time.deltaTime, 0, 0);
            }
        }
    }

    void OnTriggerEnter2D(Collider2D coll)
    {

        if (coll.gameObject.tag == "Fish2")
        {
            Hunt = true;
            GetComponent<Animator> ().SetInteger("Run" , 1);
            StartCoroutine(Idel());
        }
    }
    IEnumerator Idel(){
        yield return new WaitForSeconds (2f);
        Hunt = false;
        GetComponent<Animator> ().SetInteger("Run" , 0);
    }
}
```

При виконанні даного скрипта якщо рибка яка піднята на крючок стикається з “чорною рибою” то вона перетворюється в скелет

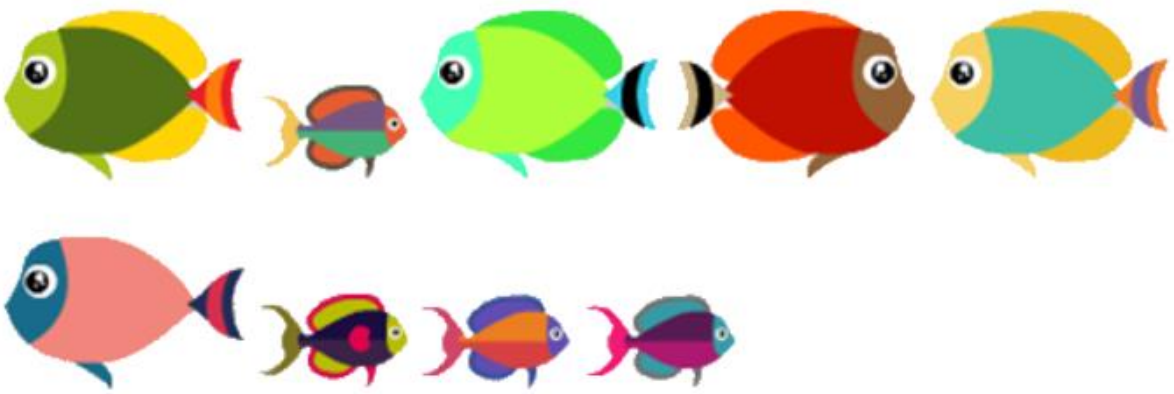


Рис 3.2.6.Спрайти риб

Вихідний код даного персонажу

```
public GameObject Fish2;
public GameObject MainCam;
// Use this for initialization
void Start()
{
    up = false;
    down = true;
}

// Update is called once per frame
void Update()
{
    MainCam = GameObject.Find("Main Camera");
    if (transform.position.x >= MainCam.transform.position.x + 5f)
    {
        transform.localScale = new Vector3(1, 1, 1);
        up = false;
        down = true;
    }
    if (transform.position.x <= MainCam.transform.position.x - 5f)
```

```
{
    transform.localScale = new Vector3(-1, 1, 1);
    up = true;
    down = false;
}
if (up == true)
{
    SU += SPEED * Time.deltaTime;
    if (SU >= HighSpeed)
    {
        SU = HighSpeed;
    }
    transform.position += new Vector3(SU * Time.deltaTime, 0, 0);
}

if (up == false)
{
    SU -= STOP * Time.deltaTime;
    if (SU < 0)
    {
        SU = 0;
    }
    transform.position += new Vector3(SU * Time.deltaTime, 0, 0);
}
//////////////////////////////////////////////////////////////////////////////


if (down == true)
{
    SD += SPEED * Time.deltaTime;
    if (SD >= HighSpeed)
    {
```

```
                SD = HighSpeed;
            }
            transform.position -= new Vector3(SD * Time.deltaTime, 0, 0);
        }

        if (down == false)
        {
            SD -= STOP * Time.deltaTime;
            if (SD <= 0)
            {
                SD = 0;
            }
            transform.position -= new Vector3(SD * Time.deltaTime, 0, 0);
        }
    }

    void OnTriggerEnter2D(Collider2D coll)
    {

        if (coll.gameObject.tag == "Hunt")
        {
                GameObject.Find ("Manage").GetComponent<Manage> ().Supply = true;
                GameObject.Find ("Rope").GetComponent<Rope> ().Fishing = false;
            Instantiate(Fish2, new Vector3(transform.position.x, transform.position.y ,-1f),
Quaternion.identity) ;
            Destroy(gameObject);
        }
    }
}

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

```
using UnityEngine;

public class Fish2 : MonoBehaviour {

   public bool up;

   public bool down;

   public float SU;
  public GameObject Bone;
   public GameObject Target;

   void Start () {

      up = false;
      down = true;
   }

  // Update is called once per frame
  void Update () {
      Target = GameObject.Find("TARGET");
      transform.position = Vector3.MoveTowards(transform.position,
Target.transform.position, 10);
      if (transform.eulerAngles.z >=320&& transform.eulerAngles.z <= 350)
      {

        up = false;

      }
      if (transform.eulerAngles.z >= 170 && transform.eulerAngles.z <= 200)
      {
```

```
            up = true;

        }
        if (up == true)
        {

            transform.eulerAngles += new Vector3(0, 0, SU * Time.deltaTime);
        }

        if (up == false)
        {

            transform.eulerAngles += new Vector3(0, 0,-SU * Time.deltaTime);
        }




    }
    void OnTriggerEnter2D(Collider2D coll)
    {

        if (coll.gameObject.tag == "BlackFish")
        {
            if (GameObject.Find("Manage").GetComponent<Manage>().Sound == 0)
            {
                GameObject.Find("Eat").GetComponent<AudioSource>().Play();
            }
                    GameObject.Find ("TARGET").GetComponent<Target> ().Hunt = 0;
                    Instantiate(Bone, new Vector3(transform.position.x, transform.position.y ,-
1f), Quaternion.identity) ;
            Destroy(gameObject);
```

```
        }
        if (coll.gameObject.tag == "Player")
        {
            if (GameObject.Find("Manage").GetComponent<Manage>().Sound == 0)
            {
                GameObject.Find("Coin").GetComponent<AudioSource>().Play();
            }
            GameObject.Find ("Manage").GetComponent<Manage> ().Score += 1;
                    GameObject.Find ("barlife").GetComponent<Barlife> ().Life = true;
                    GameObject.Find ("TARGET").GetComponent<Target> ().Hunt = 0;
            Destroy(gameObject);
        }
    }
}
```

Далі було створено сюжетні сцени гри

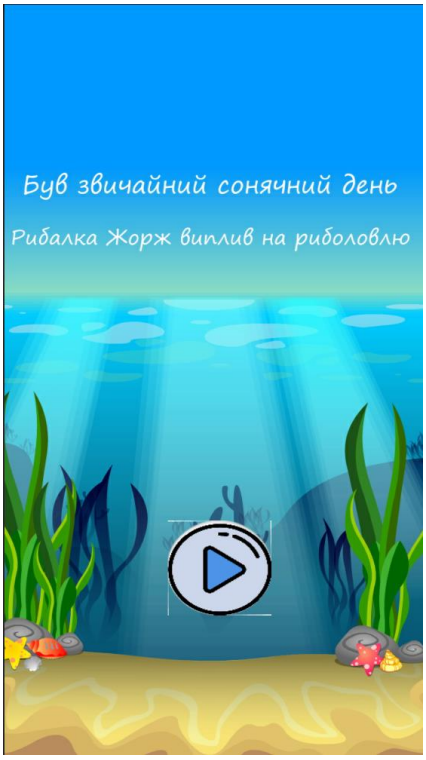


Рис 3.2.8..Перша сцена проекту

Рис 3.2.9.Друга сцена проекту

Рис 3.2.10.Третя сцена проекту

Перемикання між даними сценами відбуваються за допомогою програмного коду кнопки .А саме: :

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

```
public class levelScript : MonoBehaviour {
    public int level;
  // Use this for initialization
  void Start () {

  }

    // Update is called once per frame
    [System.Obsolete]
    void Update () {
      if (Input.GetMouseButtonDown(0))
        {
            Application.LoadLevel(level);

        }
  }
}
```

В властивостях кожного button-у вказано номер сцени на яку необхідно перемкнути її Наступна сцена являється головною гемплейною сценою

Рис 3.2.11.кнопка запуску гри

При натисканні на кнопку Play реалізовано запуск самого гемплею

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

```
public class Play : MonoBehaviour {
  public bool up;
  //public float Speedup;
  public bool down;
  //public float Speeddown ;
  public float SPEED;
  public float STOP;
  public float HighSpeed;
  public float SD;
  public float SU;

  public GameObject Items;
  public GameObject Barlife;
  // Use this for initialization
  void Start () {
        up = false;
        down = true;
  }

  // Update is called once per frame
  void Update () {
        if (transform.localScale.x >= 0.9f)
        {

              up = false;
              down = true;
        }
        if (transform.localScale.x <= 0.8f)
        {

              up = true;
              down = false;
```

```
        }
        if (up == true)
        {
            SU += SPEED * Time.deltaTime;
            if (SU >= HighSpeed)
            {
                SU = HighSpeed;
            }
            transform.localScale += new Vector3(SU * Time.deltaTime, SU *
Time.deltaTime, 0);
        }

        if (up == false)
        {
            SU -= STOP * Time.deltaTime;
            if (SU < 0)
            {
                SU = 0;
            }
            transform.localScale += new Vector3(SU * Time.deltaTime, SU *
Time.deltaTime, 0);
        }
        ////////////////////////////////////////////////////////////////////////////


        if (down == true)
        {
            SD += SPEED * Time.deltaTime;
            if (SD >= HighSpeed)
            {
                SD = HighSpeed;
            }
```

```
                transform.localScale -= new Vector3(SD * Time.deltaTime, SD *
Time.deltaTime, 0);
            }

            if (down == false)
            {
                SD -= STOP * Time.deltaTime;
                if (SD <= 0)
                {
                    SD = 0;
                }
                transform.localScale -= new Vector3(SD * Time.deltaTime, SD *
Time.deltaTime, 0);
            }
        }
        void OnMouseDown(){
            Barlife.SetActive (true);
            GameObject.Find ("Manage").GetComponent<Manage> ().Startgame = 1;
            Destroy (Items);
        }
    }
```

Рис 3.2.12..кнопка управління звуком

Реалізовна кнопка відмикання та вмикання звуку using System.Collections; using System.Collections.Generic;

```
using UnityEngine;


public class Sound : MonoBehaviour {
  public GameObject S;
  public GameObject M;
  // Use this for initialization
  void Start () {


  }


  // Update is called once per frame
  void Update () {
        if (GameObject.Find ("Manage").GetComponent<Manage> ().Sound == 0) {
              M.GetComponent<SpriteRenderer> ().enabled = false;
              S.GetComponent<SpriteRenderer> ().enabled = true;
        }
        if (GameObject.Find ("Manage").GetComponent<Manage> ().Sound == 1) {
              M.GetComponent<SpriteRenderer> ().enabled = true;
              S.GetComponent<SpriteRenderer> ().enabled = false;
        }
  }
  void OnMouseDown(){
```

```
            GameObject.Find ("Manage").GetComponent<Manage> ().Sound += 1;

        }

    }
```

Також були реалізовані додаткові скріпти логіки гри та поведінки риб в цілому

Наприклад скрипт ловлі риби:

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

    public class Fishing : MonoBehaviour {
     public GameObject Rope;
     // Use this for initialization
     void Start () {
            Rope = GameObject.Find("Rope");
     }

     // Update is called once per frame
     void Update () {

     }
     void OnMouseDown(){
            if (GameObject.Find ("TARGET").GetComponent<Target> ().Hunt == 0) {
                if (Rope.transform.localScale.y <= 1 && GameObject.Find
("Manage").GetComponent<Manage> ().Startgame == 1) {
                    Rope.GetComponent<Rope> ().Fishing = true;
                }
            }
```

Рис 3.2.13. повзунок відліку часу

Реалізований повзунок часу відліку до закінчення раунду головним персонажем

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

public class Barlife : MonoBehaviour {
  public float Speed;
  public bool Life;


  // Use this for initialization
  void Start () {

  }

  // Update is called once per frame
  void Update () {
        if (transform.localScale.x >= 1) {
              transform.localScale = new Vector3 (1, 1, 1);
        }
        if (Life == true) {
              transform.localScale += new Vector3 (6f*Time.deltaTime, 0, 0);
              Life = false;
        }
        if (GameObject.Find ("Manage").GetComponent<Manage> ().Startgame == 1) {
              Speed = 1.3f * Time.deltaTime;
              transform.localScale -= new Vector3 (Speed * Time.deltaTime, 0, 0);
```

```
                    if (transform.localScale.x <= 0) {
                        transform.localScale = new Vector3 (0, 1, 1);
                        GameObject.Find ("Manage").GetComponent<Manage> ().End =
true;
                        //GameObject.Find ("Manage").GetComponent<Manage>
().Startgame = 0;

                    }
                }

        }
    }
```

Скрипт загальної логіки в цілому

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

public class Manage : MonoBehaviour {
  public int Startgame;
  public int Score;
  public int BestScore;
  public int Sound;
  public int R;
  public int Randomfish;
  public GameObject Again;
  public GameObject Fish;
  public bool Supply;
  public float TimeScale;
  public bool End;
  public GameObject F1;
  public GameObject F2;
  public GameObject F3;
```

```
public GameObject F4;
public GameObject F5;
public GameObject F6;
public GameObject F7;
public GameObject F8;
public GameObject F9;
public GameObject F10;
public GameObject F11;
public GameObject F12;
public GameObject F13;
public GameObject Black1;
public GameObject Black2;
public GameObject Black3;
  public GameObject Bubble;
  public GameObject Tolid;
  public float H;
  public int Once;
  public bool PlayMusic;
// Use this for initialization
void Start () {
      InvokeRepeating("Globe", 3f, 5f);
      Score = 0;
          TimeScale = 1;
          BestScore = PlayerPrefs.GetInt ("BestScore", BestScore);


          Sound = PlayerPrefs.GetInt ("Sound", Sound);
}

// Update is called once per frame
void Update () {
```

```
    if (Startgame == 0)
{
    if (Sound == 0&&PlayMusic==false)
    {
        GameObject.Find("Soundgame1").GetComponent<AudioSource>().Play();
        PlayMusic = true;
    }
    if (Sound == 1)
    {
        GameObject.Find("Soundgame1").GetComponent<AudioSource>().Stop();
        PlayMusic = false;
    }

}
if (Startgame == 1)
{
    GameObject.Find("Soundgame1").GetComponent<AudioSource>().Stop();
    if (Sound == 0 && PlayMusic == true)
    {
        GameObject.Find("Soundgame2").GetComponent<AudioSource>().Play();
        GameObject.Find("Bird").GetComponent<AudioSource>().Play();
        PlayMusic = false;
    }
    if (Sound == 1)
    {
        GameObject.Find("Soundgame2").GetComponent<AudioSource>().Stop();
        GameObject.Find("Bird").GetComponent<AudioSource>().Stop();
        PlayMusic = true;
    }

}
if (Score >= 10) {
```

```
        Black1.SetActive (true);
}
if (Score >= 20) {
        Black2.SetActive (true);
}
if (Score >= 40) {
        Black3.SetActive (true);
}
if (Score >= 9999) {
        Score = 9999;
}
if (Score <= 0) {
        Score = 0;
}
Time.timeScale = (float)TimeScale;
if (End == true) {
        Once += 1;
        if (Once == 1) {
                TimeScale = 0.2f;
                StartCoroutine (Endgame ());
        }
}
if (Sound >= 2) {
        Sound = 0;
}




if (Supply == true) {
        Randomfish = Random.Range (1, 14);
        if (Randomfish == 1) {
```

```
        Fish = F1;
}
if (Randomfish == 2) {
        Fish = F2;
}
if (Randomfish == 3) {
        Fish = F3;
}
if (Randomfish == 4) {
        Fish = F4;
}
if (Randomfish == 5) {
        Fish = F5;
}
if (Randomfish == 6) {
        Fish = F6;
}
if (Randomfish == 7) {
        Fish = F7;
}
if (Randomfish == 8) {
        Fish = F8;
}
if (Randomfish == 9) {
        Fish = F9;
}
if (Randomfish == 10) {
        Fish = F10;
}
if (Randomfish == 11) {
        Fish = F11;
}
```

```
                if (Randomfish == 12) {
                        Fish = F12;
                }
                if (Randomfish == 13) {
                        Fish = F13;
                }
                R = Random.Range (0, 2);
                if (R == 1) {
                        Instantiate (Fish, new Vector3 (GameObject.Find ("Main
Camera").transform.position.x-5.5f, Random.Range (-0.5f, -5.2f), 0), Quaternion.identity);
                }
                if (R == 0) {
                        Instantiate (Fish, new Vector3 (GameObject.Find ("Main
Camera").transform.position.x+5.5f, Random.Range (-0.5f, -5.2f), 0), Quaternion.identity);
                }
                Supply = false;
        }
        if (Score >= BestScore) {
                BestScore = Score;
        }

        PlayerPrefs.SetInt ("BestScore", BestScore);

        PlayerPrefs.SetInt ("Sound", Sound);
    }
    IEnumerator Endgame(){
        yield return new WaitForSeconds (1f);

        Instantiate(Again, new Vector3(0,0 ,-5f), Quaternion.identity) ;
        StartCoroutine (T ());
    }
    IEnumerator T(){
```

```
            yield return new WaitForSeconds (0.2f);
            TimeScale = 0;
        }
        void Globe()
        {
            H = Random.Range(0.3f, 1f);
            Tolid=Instantiate(Bubble, new Vector3(Random.Range(-4f, 4f),-5f, 0),
Quaternion.identity)as GameObject;
            Tolid.transform.localScale = new Vector3(H, H, H);
            H = Random.Range(0.3f, 1f);
            Tolid = Instantiate(Bubble, new Vector3(Random.Range(-4f, 4f), -5f, 0),
Quaternion.identity) as GameObject;
            Tolid.transform.localScale = new Vector3(H, H, H);
            H = Random.Range(0.3f, 1f);
            Tolid = Instantiate(Bubble, new Vector3(Random.Range(-4f, 4f), -5f, 0),
Quaternion.identity) as GameObject;
            Tolid.transform.localScale = new Vector3(H, H, H);
            H = Random.Range(0.3f, 1f);
            Tolid = Instantiate(Bubble, new Vector3(Random.Range(-4f, 4f), -5f, 0),
Quaternion.identity) as GameObject;
            Tolid.transform.localScale = new Vector3(H, H, H);
            H = Random.Range(0.3f, 1f);
            Tolid = Instantiate(Bubble, new Vector3(Random.Range(-4f, 4f), -5f, 0),
Quaternion.identity) as GameObject;
            Tolid.transform.localScale = new Vector3(H, H, H);
            H = Random.Range(0.3f, 1f);
            Tolid = Instantiate(Bubble, new Vector3(Random.Range(-4f, 4f), -5f, 0),
Quaternion.identity) as GameObject;
            Tolid.transform.localScale = new Vector3(H, H, H);  }
```

ВИСНОВКИ

Під час створення програмного продукту , було пройдено такі етапи як :

1. Аналіз та характеристика предметної області проекту та існуючі аналоги світового ринку.
2. На етапі проектування програмного забезпечення був зроблений аналіз середи розробки, на якому робився проект, чому використовується саме вона, та її переваги.
3. На завершальному етапі було створено прототип додатку з майже всім необхідним функціоналом.

В подальшому я сподіваюсь доопрацювати даний проект та привести його до фінального стану

СПИСОК ЛІТЕРАТУРИ

1. C # Programming Guide. [Електронний ресурс] URL: https://msdn.microsoft.com/en-us/library/67ef8sbd.aspx

2. Flower M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. - USA: Addison-Wesley Publishing Company, 2004. - 150 p.

3. Gold J. Object-Oriented Game Development. - UK: Pearson Education Limited, 2004. - 404 p.

4. Gregory J. Game Engine Architecture. - USA: A K Peters / CRC Press, 2009. - 864 p

5. Kaner C., Bach J., Pettichord B. Lessons Learned in Software Testing. - USA: Wiley, 2001. - 320 p.

6. Tidwell J. Designing Interfaces. - USA: O'Reilly Media, 2004. - 578 p.

7. Unity3D Manual. [Електронний ресурс] URL: http://docs.unity3d.com/Manual/index.html (дата звернення: 29.04.2017).

8. Вігенс К., Бітті Д. Розробка вимог до програмного забезпечення.

9. Офіційний сайт Unity3D. [Електронний ресурс] URL: https://unity3d.com/ru

10 Рамбо Дж. UML 2.0. Об'єктно-орієнтоване моделювання розробка. - СПб .: Пітер, 2007. - 544 с.

Додаток А

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка мобільної гри жанру 2Д риболовецька пригода використовуючи Unity 2020 на мові C#

Виконано студент 5 курсу
Групи ППЗ-52
Руденко О.О
Керівник роботи
Негоденко Олена Василівна

Київ – 2021

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи дослідження розробки прототипу мобільної гри жанру 2D-пригода засобами середовища Unity
Об'єкт дослідження -розробка 2D гри для смартфонів засобами середовища Unity

Предмет дослідження - технології розробки 2-д гри для смартфону жанру пригода

АНАЛОГИ

Назва гри	Характеристика		
	2D	Статистика гравців	Сюжет
FishingLegend	Так	Так	Так
Fishing Break	Так	Ні	Ні
Ridiculous Fishing	Так	Ні	Ні

3

ТЕХНІЧНІ ЗАВДАННЯ

1. Дослідження методів розробки ігор з застосуванням рушія Unity
2. Спроєктувати продукт на базі проведених досліджень
3. Розробити програмну реалізацію переміщення гравця та риб
4. Створення алгоритмів для обрахування рекорду в грі
5. Розробка локації гри
6. Проаналізувати відповідність розробленого продукту початковим вимогам

4

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Ігровий рушій Unity

Мова програмування C#

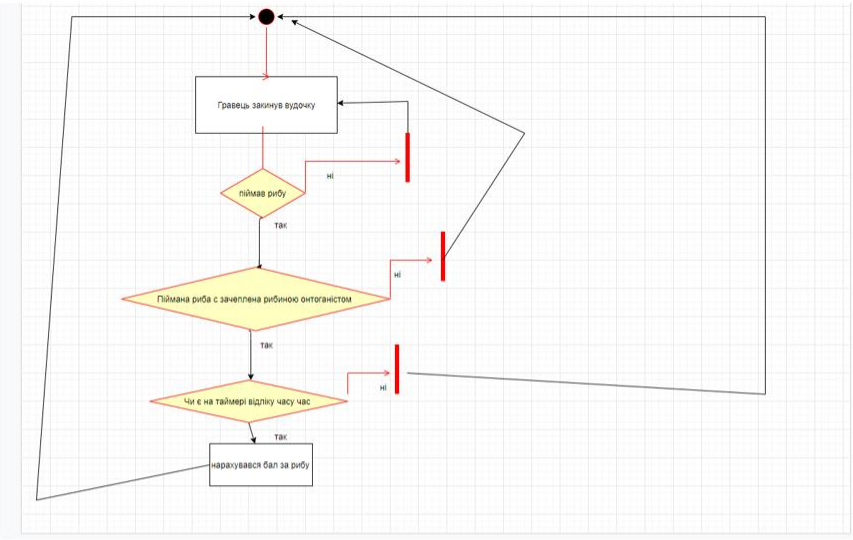
Інтегроване середовище розробки Visual Studio

Цільова платформа розробки - Android

5

Моделюваня інтерфейсу гри

6



Діаграма діяльності

7

Відео з додатку

8

ВИСНОВКИ

Під час створення програмного продукту , було пройдено такі етапи як :

1. Аналіз та характеристика предметної області проекту та існуючі аналоги світового ринку.
2. На етапі проектування програмного забезпечення був зроблений аналіз середи розробки, на якому робився проект, чому використовується саме вона, та її переваги.
3. Було створено додаток длясмартфоназ усім необхідним функціоналом

ДЯКУЮ ЗА УВАГУ!