**МІНІСТЕРТСВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Національний університет водного господарства та природокористування**

**Факультет прикладної математики та комп’ютерно-інтегрованих систем**

**Кафедра прикладної математики**

**Курсова робота**

на тему:

«Створення 3D гри в середовищі Unity 3D»

**Виконав: студент групи**

**КНІТ-21**

**Єфімчук Вадим**

**Перевірив: ст. викладач**

**Жуковський В. В.**

**Рівне 2018**

***Зміст***

[ВСТУП 4](#_Toc514095677)

[Життєвий цикл розроблюваної](#_Toc514095678) [гри (програми) 7](#_Toc514095679)

[**Аналіз** 7](#_Toc514095680)

[Ідея розробки і предметна область 7](#_Toc514095681)

[Функціональні можливості та постановка вимог 9](#_Toc514095682)

[**Проектування** 10](#_Toc514095683)

[Вибір середовищ реалізації 10](#_Toc514095684)

[**Алгоритм розв'язання задачі** 11](#_Toc514095685)

[**Тестування** 13](#_Toc514095686)

[**Впровадження** 13](#_Toc514095687)

[Інструкція використання гри 14](#_Toc514095688)

[Меню 14](#_Toc514095689)

[Гра (Правила) 14](#_Toc514095690)

[Керування 15](#_Toc514095691)

[Вікно поразки 16](#_Toc514095692)

[Коди поведінки гри й персонажа 17](#_Toc514095693)

[**Скрипти** 17](#_Toc514095694)

[MainMenu 17](#_Toc514095695)

[LookAt 17](#_Toc514095696)

[Game 17](#_Toc514095697)

[Player 17](#_Toc514095698)

[Food 18](#_Toc514095699)

[LoadLevel 18](#_Toc514095700)

[**UML діаграми** 19](#_Toc514095701)

[MainMenu 19](#_Toc514095702)

[Player 19](#_Toc514095703)

[LoadLevel 20](#_Toc514095704)

[Текст програми 21](#_Toc514095705)

[MainMenu 21](#_Toc514095706)

[LookAt 21](#_Toc514095707)

[Game 22](#_Toc514095708)

[Player 23](#_Toc514095709)

[Food 24](#_Toc514095710)

[LoadLevel 24](#_Toc514095711)

[Висновки 25](#_Toc514095712)

[Список використаної літератури 26](#_Toc514095713)

[**Інтернет посилання, які було використано в процесі створення гри 27**](#_Toc514095715)

# 

# ВСТУП

На даний момент комп’ютерна сфера є досить розвинутою. Існує безліч різних програмних забезпечень, ігор, додатків… Майже в кожного є комп’ютери і смартфони, які останнім часом, можуть частково замінити повноцінний ПК.

Можна допустити, що кожен, хто працював за комп’ютером або тримав у руках телефон, знає, що таке відео-гра.

Приклади одних з найстаріших ігор:

* Змійка
* Spacewar
* Тетріс
* Bounce

Технології дуже швидко рухаються вперед. Відповідно, що ігрова індустрія також розвивається. За останні 20-30 років різниця між ігровими продуктами колосальна.

Деякі ігри допомагають в житті. Наприклад, допомагають швидше розвиватися і розвинути в собі потрібні якості…

Позитивні риси ігор:

* Розвиток абстрактного мислення і логіки
* Швидкість реакції
* Уважність
* Дрібна моторика
* Допомагають в навчанні
* Вивчення іноземних слів, мов
* Вчать орієнтуватися на карті

З цього можна дійти висновку, що ігри можуть навчити аналізувати ситуацію і дуже швидко реагувати на неї. Однак, не від всіх може бути такий результат.

Негативні риси ігор:

* Пропаганда
* Псування зору
* Психічна залежність
* Копія рухів персонажа (Більшість у дітей.)
* Займають багато часу
* Відривають гравця від суспільства (Онлайн-ігри не є повноцінною заміною.)
* …

Хороша і погана сторона є, як і у всього, в комп’ютерних іграх. Важливо пам’ятати, що все добре в міру.

Виходячи з позитивних і негативних рис, буде формуватися завдання і мета проекту.

*Завдання проекту:*

1. Створити модель проекту й визначитися з платформою для гри
2. Вибрати середовище реалізації гри
3. Створити візуальну частину (Ту частину, яку буде бачити користувач)
4. Створити саму гру в раніше вибраному середовищі і зробити гральне поле у вигляді студмістечка НУВГП.
5. Написати коди поведінки персонажів гри
6. Налаштування гри й ігрових характеристик
7. Оптимізація гри
8. Додати звуковий супровід(музику).
9. Провести випробовування гри й усунути можливі недоліки.

*Мета проекту:*

1. Зробити гру позитивною, цікавою для користувача з достойним музичним супроводом.
2. Уникнути негативних рис відео-ігор
3. Допомогти гравцеві приємно провести час
4. Розвинути користувачу деякі позитивні риси характеру

*Дана відео-гра має володіти наступними функціональними можливостями*:

1. Меню
2. Кнопка “Почати гру”
3. Кнопка “Вийти з гри”
4. Гра
5. Підраховувати кількість набраних очок
6. Вікно, що буде відображатися при поразці гравця і через певний час буде відкриватися меню

*Суть гри:*

Суть гри полягає в тому, щоб збирати даним об’єктом(Персонажем) інші об’єкти(Їжу). Аналог гри “Змійка”.“Їжа” буде випадати випадково, на будь яку точку ігрового поля.

Також необхідно буде уникати інших ігрових об’єктів. Ті об’єкти будуть розставлені так, що будуть віддалено нагадувати карту(модель) студмістечка Національного університету водного господарства та природокористування (НУВГП).

При взаємодії з ними гравцеві буде присвоєна поразка й відбудеться перекидання на вікно поразки, й через певний час до меню.

# Життєвий цикл розроблюваної

# гри (програми)

## **Аналіз**

### **Ідея розробки і предметна область**

Ідеєю розроблювальної гри є створення простої розваги для користувача(гравця). Гра має бути максимально простою і зрозумілою для користувача будь якого віку.

Для себе я поставив мету опанувати середовище розробки ігор Unity3D.

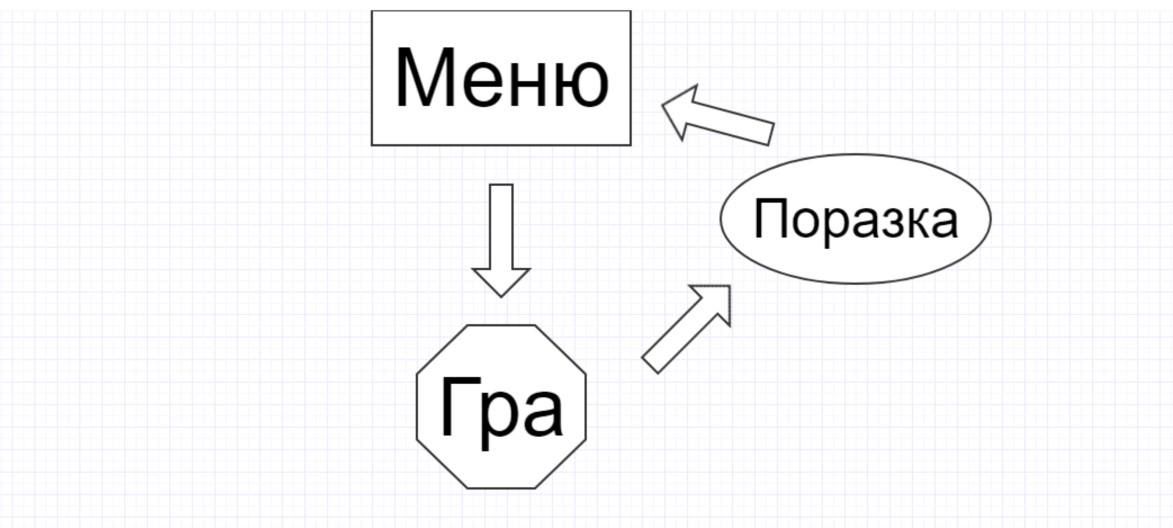
Сама ж гра є результатом моїх знань і навичок в програмуванні, дизайні…

Швидше за все, цей програмний продукт буде цікавий для дітей, бо, насамперед, простота не буде вимагати ніяких особливих вмінь і знань.

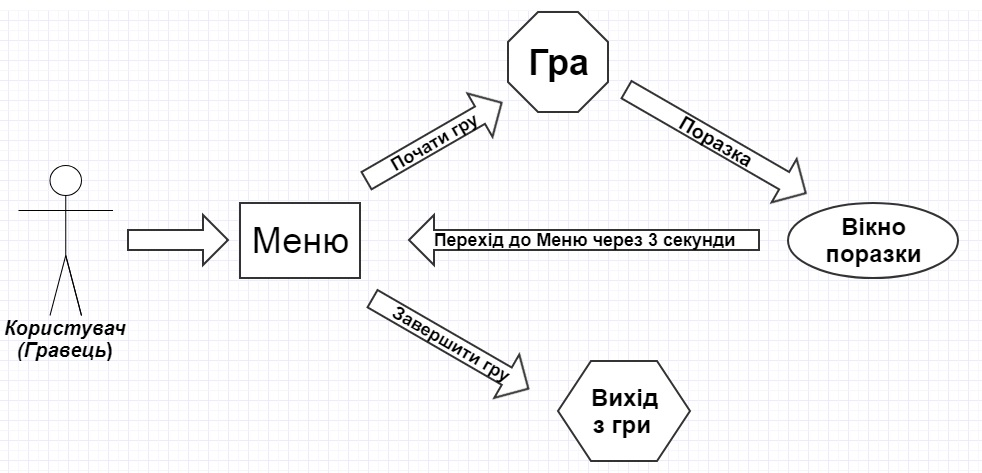
Гра не матиме серверної частини, а буде розміщуватися й виконуватися лише на стаціонарному персональному комп’ютері. Спочатку вона буде тільки під операційну систему Windows.

З часом, у майбутньому, є в планах модернізувати її та реалізувати на інших операційних системах та платформах. Наприклад: Mac, Linux, Android, iOS… Також викласти на GooglePlay для розповсюження.

Схема гри:



Діаграма прецедентів:



### **Функціональні можливості та постановка вимог**

Модулі гри такі: інтерфейс, керування персонажем, підрахунок кількості набраних очок, звуковий супровід(Музика).

**Інтерфейс:**

Його основне завдання – бути максимально простим і зрозумілим, мати мінімум кнопок і приємну картинку.

**Керування персонажем:**

Керування буде здійснюватися лише двома кнопками: Стрілка вліво, Стрілка вправо. Персонаж буде рухатися прямо самостійно із заданою, сталою швидкістю. Це зроблено з метою якомога більше спростити гру, розуміння гри й код поведінки персонажа.

**Підрахунок кількості набраних очок:**

Завдання гри – набрати якомога більше очок. Тому потрібно реалізувати код взаємодії персонажа з “їжею”. Він буде підраховувати скільки очок було набрано.

**Звуковий супровід(Музика):**

Звук є невід’ємною частиною будь якої гри, тому слід ретельно підібрати її. Це так звана дизайнерська частина.

## **Проектування**

### **Вибір середовищ реалізації**

Середовищем розробки гри була обраний потужний, багатоплатформовий інструмент для розробки додатків та ігор Unity 3D. Коди поведінки об’єктів були написані на мові C# в середовищі Microsoft Visual Studio 2015. Для реалізації дизайну інтерфейсу були використані текстури з інтернету, які були оброблені в Adobe Photoshop cs6.

**Середовище розробки:** Unity 2018.1.0f2 (64-bit)

**Мова програмування:** C# 6.0

**Версія .NET Framework:** 3.5

**При розробці було використано також:**

* Visual Studio 2015
* Adobe Photoshop cs6

## **Алгоритм розв'язання задачі**

Задача полягає в тому, аби було найбільш просто і зрозуміло. Тобто

Меню-Гра-Вікно поразки. Це все буде реалізовано за допомогою методів класу MonoBehaviour на C#.

if (GUILayout.Button(startButtonText, buttonStyle, GUILayout.MinHeight(buttonMinHeight)))

{

SceneManager.LoadScene("Level");

}

Це код перевірки натискання на кнопку і загрузки сцени гри.

Далі в нас має бути генерація їжі у випадковому місці:

public static void GenerateNewFood()

{ ... }

І при цьому вона не повинна стикатися з іншими предметами.

Наступним треба реалізувати код поведінки персонажа. Якщо він взаємодіє їжею, то ми генеруємо нову, якщо ні, то загружаємо вікно поразки.

public void OnControllerColliderHit(ControllerColliderHit hit)

{

if (\_testing)

{

Food food = hit.collider.GetComponent<Food>();

if (food != null)

{

// з’їдаємо їжу якщо взаємодіяли з нею.

food.Eat();

AddTile();

}

else

{

// Врізалися в блок, або стіну

SceneManager.LoadScene ("GameOver");

}

\_testing = false;

}

}

Також необхідно налаштувати камеру, з’єднати її з намиш персонажем. Її основна мета – слідкувати за ним і його рухами. Це зроблено заради зручності гри.

// ціль на яку дивиться камера

public Transform MyPlayer;

public void Update()

{

if (MyPlayer != null)

{

/\* дивиться завжди на ціль, яку ми пізніше поставимо у програмі в спец рядок скрипту \*/

transform.LookAt(MyPlayer);

}

}

А далі найпростіше – після загрузки вікна поразки, закривати його через декілька секунд і відкривати меню

public IEnumerator Start()

{

// Затримка на, задану раніше, кількість секунд

yield return new WaitForSeconds(SecTime);

// Загрузка сцени Меню

SceneManager.LoadScene("MainMenu");

}

## **Тестування**

В результаті тестування не було виявлено помилок та багів, бо гра є досить простою. Також було покращено освітлення, замінено деякі текстури, надписи й шрифти.

Однак виникла проблема на фінальній стадії, де потрібно було “збудувати” гру (Зліпити всі сцени до купи). Не правильно працювала основна сцена гри. При її запуску зразу був перехід на фінальну сцену. Це все було виправлено і тепер гра правильно функціонує.

Повністю замінений весь звуковий супровід. Було підібране щось у більш веселому і драйвовому стилі.

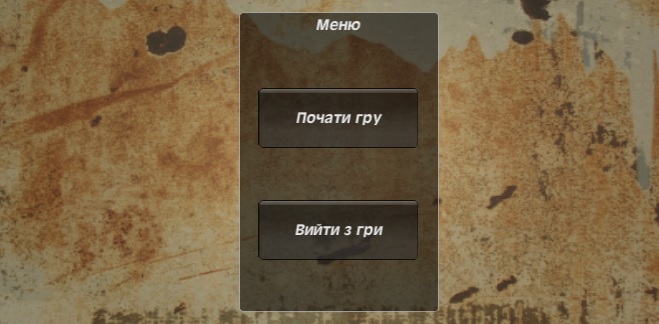
## **Впровадження**

Для цього проекту є не досить важливим етапом. Так як основною метою створення гри було навчитися працювати з Unity 3D. На майбутнє є в планах поширити її в GooglePlay.

# Інструкція використання гри

### **Меню**

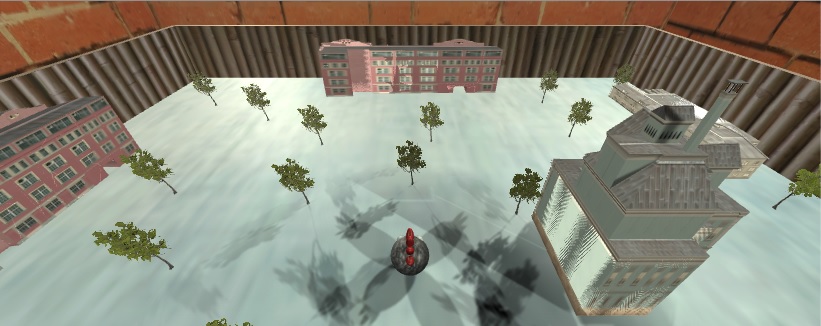
Меню має такий досить простий вигляд:



Все елементарно просто. Тиснемо “Вийти з гри” й наша програма закриється. І якщо бажаємо розпочати гру – тиснемо “Почати гру”. Далі нас перекине на сцену самої гри.

### **Гра (Правила)**

Гра має такий вигляд:



Правила гри є досить простими. Необхідно збирати синій шар(Їжу, вона появляється у випадковому місці)

E:\Папка\Воднік\2 курс\2 семестр\Курсова робота\Їжа.jpg

При цьому потрібно не забувати, що не можна торкатись будинків-блоків та стін нашого грального поля. У випадку торкання буде поразка і ввімкнення сцени поразки.



Також, для цікавості є рахівник, що буде підраховувати кількість зібраної персонажем їжі.

E:\Папка\Воднік\2 курс\2 семестр\Курсова робота\Рахівник.jpg

### **Керування**

Керування також є досить простим. Воно потребує всього дві кнопки:

* Стрілка вліво, для повороту вліво
* Стрілка вправо, для повороту вправо

Персонаж в автоматичному режимі рухається вперед із заданою йому швидкістю.

### **Вікно поразки**

Вікно поразки з написом “Грі кінець” з'являється у випадку взаємодії персонажа з блоками або стінами і означає, що гру завершено.

Вікно поразки:



# Коди поведінки гри й персонажа

## **Скрипти**

Всього було створено шість скриптів:

1. MainMenu
2. LookAt
3. Game
4. Player
5. Food
6. LoadLevel

Зараз детально розглянемо кожен з них.

### **MainMenu**

В цьому скрипті ми створюємо дві кнопки “Почати гру”, “Закінчити гру”. І при натискані на кожну з них відбувалася певна дія.

### **LookAt**

В цьому скрипті реалізовано слідування ігрової камери за нашим персонажем, аби той не міг сховатися з її поля зору.

### **Game**

Це, напевно, найголосніший скрипт цієї гри. Тут реалізонано:

1. Загрузка рівня
2. Їжа, що з’являється у випадковому місці
3. Рахівник

### **Player**

Цей код використовується для появи персонажа в заданих координатах ігрового поля, керування персонажа. І перевірка на дотик з “Їжею” або стінами чи блоками. Якщо взаємодія з їжею, то рахівнику плюс один. В іншому випадку перехід до сцени “Вікно поразки”.

### **Food**

В цьому скрипті реалізована поява і знищення “Їжі”, коли з нею взаємодіє персинаж. Тут також вказано, який шаблон(префаб) треба використати, й які розміри має мати їжа

### **LoadLevel**

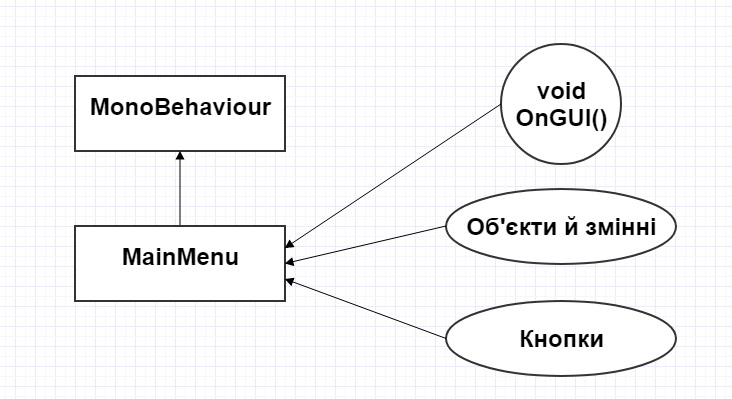
Найпростіший скрипт. У ньому реалізований лиже перехід на сцену “Меню” через заданий проміжок часу.

## **UML діаграми**

Тут будуть представлені лише декілька(Три) UML діаграм.

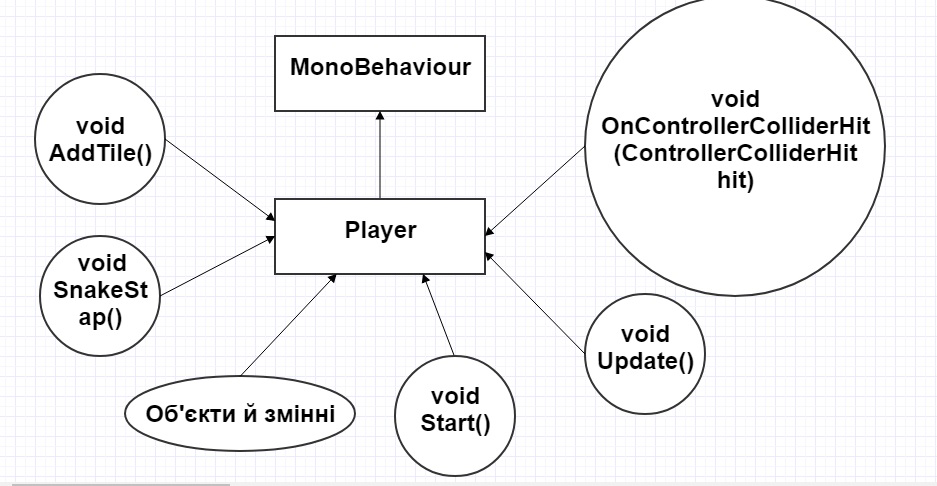
### **MainMenu**

UML діаграма 1:



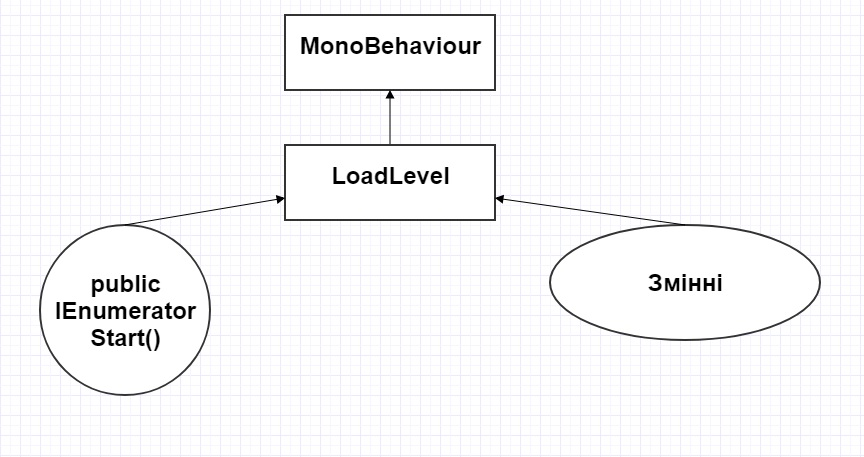
### **Player**

UML діаграма 2:



### **LoadLevel**

UML діаграма 3:



# Текст програми

Тут вже будуть представлені всі коди цієї гри. Однак вони будуть фрагментами.

### **MainMenu**

public class MainMenu : MonoBehaviour {

public Vector2 SizeMenu = new Vector2(500, 300);

public float MinHeightButton = 60f;

public Font FontСaption;

public Font FontButton;

// текст меню

public string MenuText = "Меню";

public string StartText = "Почати Гру";

public string ExitText = "Вийти з Гри";

public void OnGUI()

{

Rect rect = new Rect(

Screen.width / 2f - SizeMenu.x / 2,

Screen.height / 2f - SizeMenu.y / 2,

SizeMenu.x,

SizeMenu.y);

…

### **LookAt**

public class LookAt : MonoBehaviour {

public Transform MyPlayer;

public void Update()

{

if (MyPlayer != null)

{

transform.LookAt(MyPlayer);

}

}

}

### **Game**

public class Game : MonoBehaviour {

...

private string \_pointsString;

private int \_lastPonts = -1;

public void Awake()

{

points = 0;

Food.GenerateNewFood();

}

public void Update()

{

if (\_lastPonts == points) return;

\_lastPonts = points;

\_pointsString = "Рахунок: " + points.ToString("0000");

}

...

}

### **Player**

public class Player : MonoBehaviour

{

public float speed = 1;

private bool \_testing = false;

...

private CharacterController \_controller;

public void AddTile() {

Vector3 Position = transform.position;

if (BodySnake.Count > 0) {

Position = BodySnake[BodySnake.Count - 1].transform.position;

}

Position.y+=10f;

GameObject Body = Instantiate(SnakeBody, Position, Quaternion.identity) as GameObject;

BodySnake.Add(Body);

}

public void Start()

{

BodySnake.Clear();

for (int i = 0; i < 5; i++) AddTile();

\_controller = GetComponent<CharacterController>();

}

public void Update()

{

SnakeStap();

float horizontal = Input.GetAxis("Horizontal");

transform.Rotate(rotationSpeed \* Time.deltaTime \* horizontal,0, 0);

\_testing = true;

\_controller.Move(transform.forward \* speed \* Time.deltaTime );

}

...

public void OnControllerColliderHit(ControllerColliderHit hit)

{

if (\_testing)

{

Food food = hit.collider.GetComponent<Food>();

if (food != null)

{

food.Eat();

AddTile();

}

else

{

SceneManager.LoadScene ("GameOver");

}

\_testing = false; ...

### **Food**

public static void GenerateNewFood()

{

GameObject food = (GameObject)Instantiate(Resources.Load("Prefabs/Food", typeof(GameObject)));

while (true)

{

food.transform.position = new Vector3(Random.Range(-40, 41), 0, Random.Range(-40, 41));

Bounds foodBounds = food.GetComponent<Collider>().bounds;

bool intersects = false;

foreach (Collider objectColiider in FindObjectsOfType(typeof(Collider)))

{

if (objectColiider != food.GetComponent<Collider>())

{

if (objectColiider.bounds.Intersects(foodBounds))

{

intersects = true;

break;

}

}

}

if (!intersects)

{

break;

}

}

### **LoadLevel**

public class LoadLevel : MonoBehaviour {

public float SecTime = 3;

public string levelName;

public IEnumerator Start()

{

yield return new WaitForSeconds(SecTime);

SceneManager.LoadScene("MainMenu");

}

}

# Висновки

У процесі розробки даної відео-гри вдалося опанувати багатоплатформовий редактор для створення 2D і 3D ігор - Unity 3D. Також було досліджено базовий клас для створення скриптів на C# - MonoBehaviour. Він містить дуже велику кількість готових функцій, які можна використати у своїх скриптах. Завдяки цьому класу не довелося створювати нічого нового, а лише використовувати вбудовані функції.

Також було вирішено, поставлені нами раніше, задачі:

* Створити модель проекту й визначитися з платформою для гри
* Вибрати середовище реалізації гри
* Створити візуальну частину (Ту частину, яку буде бачити користувач)
* Створити саму гру в раніше вибраному середовищі
* Написати коди поведінки персонажів гри
* Налаштування гри й гральних характеристик
* Оптимізація гри й звуковий супровід(музику).
* Провести випробовування гри й усунути можливі недоліки

В наступних версіях гри очікується:

* Реалізувати зберігання гри
* Список лідерів, аби було видно хто набрав найбільше очок
* Додати різноманіття
* Розділити гру на рівні
* Покращити графічну частину й створити нові моделі

# Список використаної літератури

1. Герберт Шилдт. “C# полное руководство”. Перекл. з англ. — М.: ООО "И.Д. Вильямс", 2011. — 1056 с.
2. Шарп Д. “Microsoft Visual C#. Подробное руководство” — СПб.: Питер, 2017. — 848 с.
3. Н. Культин. “microsoft visual c# в задачах и примерах”. - БХВ-Петербург 2015. – 320 с.

\*Більшість інформації було взято, і переглянуто прикладів, з ресурсів Інтернету.

## **Інтернет посилання, які було використано в процесі створення гри**

1. Пошукова система - <https://www.google.com.ua/>
2. Описання функцій і методів класу MonoBehaviour - <https://docs.unity3d.com/ru/530/ScriptReference/MonoBehaviour.html>
3. Офіційний сайт Unity - <https://unity3d.com/ru>
4. Туторіали Unity - <https://unity3d.com/ru/learn/tutorials>
5. Туторіал приклад - <https://unity3d.com/ru/learn/tutorials/s/roll-ball-tutorial>
6. Ще один приклад роботи - <https://habrahabr.ru/post/264611/>
7. 2D Гра, детальна інструкція - <http://websketches.ru/blog/2d-igra-na-unity-podrobnoye-rukovodstvo-p1>
8. Відео-уроки з користування і створення ігор - <https://itproger.com/course/2d-unity-game>
9. Руководство Unity - <https://docs.unity3d.com/ru/current/Manual/index.html>
10. Створення платформера 2D Unity - <https://www.youtube.com/watch?v=CAPVBTKk3Ww>
11. Текстури - <http://junior3d.ru/texture.html>
12. Звуковий супровід - <http://zk.fm/>
13. Малювання діаграм - <https://cameralabs.org/5200-onlajn-redaktor-dlya-skhem-grafikov-i-diagramm-gliffy>