Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет» (ПНИПУ)**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

Тема: «Создание игры “Flappy Bird”»

Выполнил

Студент группы РИС-22-2б

Прядеин И.А.

Проверил доцент кафедры

ИТАС

Петренко А.А.

Пермь 2023

**Введение**

“Flappy Bird” - это аркада, разработанная вьетнамским программистом Донгом Нгуеном в 2013 году. Игра имеет минималистичный дизайн и высокую сложность, что привлекло внимание множества игроков со всего мира. Однако в 2014 году Нгуен решил удалить игру, объяснив это тем, приложение вызывает чрезмерную зависимость у пользователя. После этого появилось много аналогов “Flappy Bird”, которые пытались повторить или усовершенствовать основную идею.

Цель работы - создание аркадной игры “Flappy Bird” с использованием среды разработки Unity.

Задачи:

* Изучить основы работы с Unity и Photoshop
* Создать все необходимые модели в Photoshop
* Реализовать графический интерфейс, механику полёта и столкновения с трубами

1. **Анализ предметной области**

Unity - кроссплатформенная среда разработки, позволяющая создавать приложения, работающие на более чем 25 различных платформах, включающих персональные компьютеры, игровые консоли, мобильные устройства и другие.

Редактор Unity имеет простой Drag&Drop интерфейс, состоящий из различных окон, благодаря чему можно производить отладку игры прямо в редакторе. Для написания скриптов используется язык C#.  
Проект делится на сцены, содержащие игровые миры и наборы объектов. Объекты, в свою очередь содержат наборы компонентов, с которыми взаимодействуют скрипты.

Photoshop - многофункциональный растровый графический редактор, имеющий некоторые векторные инструменты. Программа позволяет обрабатывать фотографии, рисовать иллюстрации, создавать анимации, разрабатывать макеты сайтов и многое другое.

1. **Технология реализации**

Реализация логики игры выполнена с помощью написания скриптов на языке программирования C#.

Скрипт “Fly Slime Script” (Рис.1) отвечает за управление каплей воды в воздухе. Он использует компонент “Rigidbody2D” для придания капле импульса вверх при нажатии кнопки мыши или клавиши пробела. Для этого он обращается к свойству “velocity” и устанавливает его значение равным вектору, направленному вверх, умноженному на параметр “velocity”. Таким образом, капля подпрыгивает вверх при каждом нажатии.

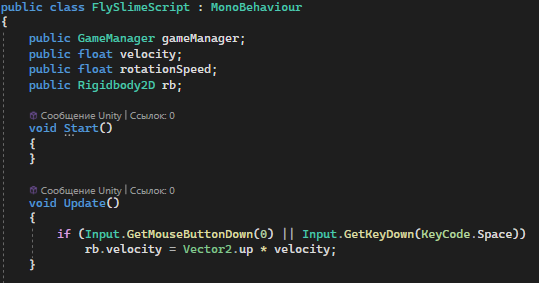


Рис.1 - Скрипт “Fly Slime Script”

Кроме того, скрипт “Fly Slime Script” изменяет угол поворота капли в зависимости от её вертикальной скорости. Для этого он использует компонент “Transform” и метод “Quaternion.Euler”, который позволяет задать поворот объекта по трем осям. Скрипт умножает вертикальную скорость капли на параметр “rotationSpeed” (Рис.2) и передаёт полученное значение в качестве угла поворота по оси Z. Таким образом, капля наклоняется вперёд при падении и назад при подъёме.



Рис.2 - Метод “FixedUpdate”

Наконец, скрипт “Fly Slime Script” обрабатывает столкновение капли с другими объектами. Для этого он использует событие “On Collision Enter 2D” (Рис.3), которое срабатывает при контакте с объектами, имеющими компоненты “Rigidbody2D” и “Collider2D”. Скрипт вызывает метод “Game Over” из скрипта “Game Manager”, который останавливает игру и показывает экран проигрыша.

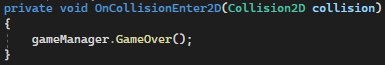


Рис.3 - Метод “OnCollisionEnter2D”

Такие объекты как “Ground” и “Two Pipes” при касании с каплей обращаются к скрипту “Game Manager” (Рис.4), который вызывает метод “Game Over” и останавливает игру. Для возобновления игры используется кнопка “Replay”, которая загружает сцену заново с помощью компонента “Scene Manager”.

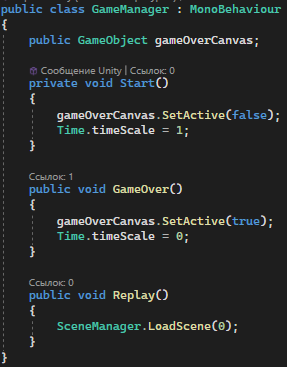


Рис.4 - Скрипт “Game Manager”

Для создания препятствий в виде труб используется скрипт “Spawn Pipe Script” (Рис.5), который в определённые промежутки времени создаёт новые объекты “Pipe” с разными высотами и уничтожает их через некоторое время. Скорость движения труб задаётся скриптом “Move Pipes Script” (Рис 6.), который изменяет позицию объектов по оси X с помощью компонента “Transform”.

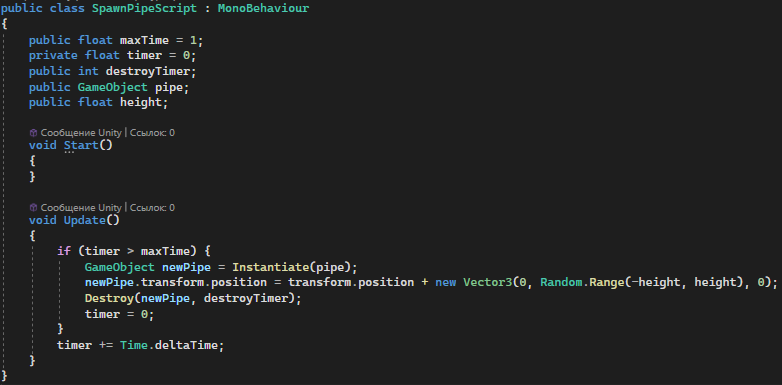


Рис.5 - Скрипт “SpawnPipeScript”

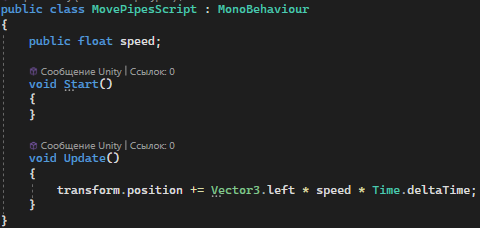


Рис.6 - Скрипт “Move Pipes Script”

Для подсчёта очков используется статическая переменная “score” (Рис.7), которая увеличивается на единицу при прохождении каплей через трубы. Для этого используется скрипт “add Score Script” (Рис 8.), который реагирует на событие “On Trigger Enter 2D” и вызывает метод “Score.score++”. Отображение текущего счёта осуществляется с помощью компонента “Text”, который обращается к скрипту “Score” и выводит значение переменной “score” на экран.

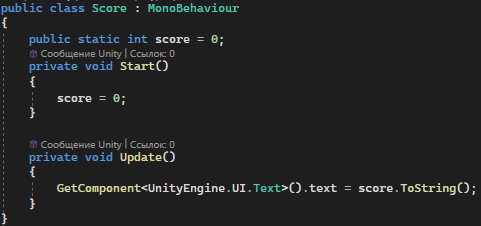


Рис.7 - Скрипт “Fly Slime Script”

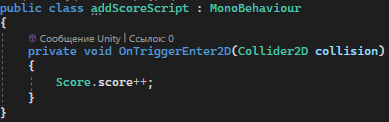


Рис.8 - Скрипт “Add Score Script”

1. Руководство пользователя

При запуске игры открывается основное игровое окно (Рис. 9) с каплей в середине.

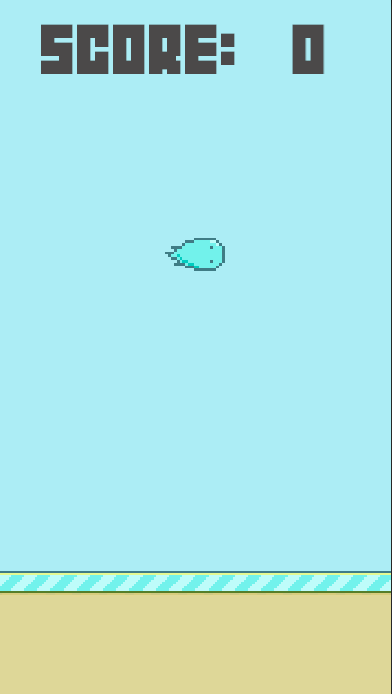


Рис.9 - Основное окно игры

Для управления используется две клавиши: левая кнопка мыши или “Spacebar”. В верхней части окна расположено текущее значение счёта. Основная задача - достижение как можно большего количества очков, минуя встречающиеся на пути трубы.

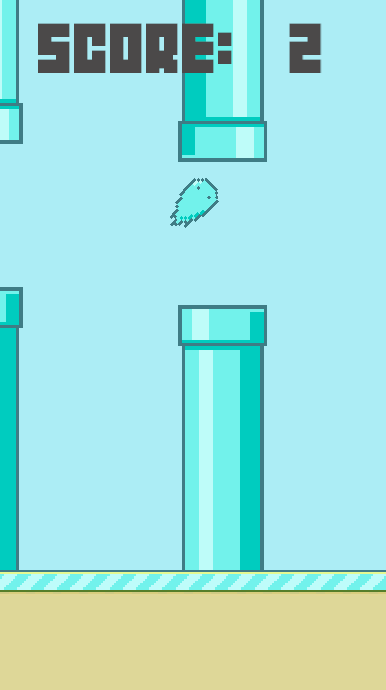


Рис.10 - Игровой процесс

В случае поражения открывается соответствующее окно (Рис.11) с кнопкой перезапуска.

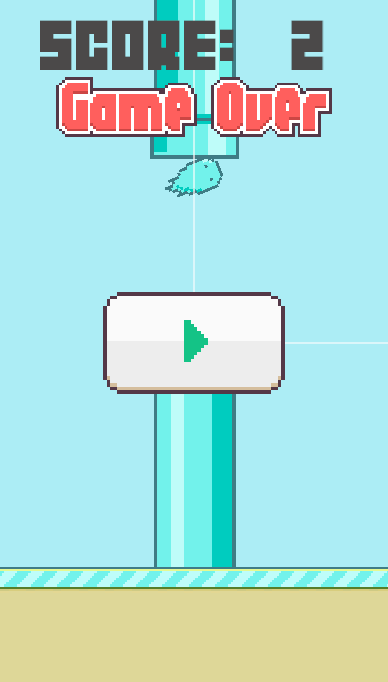


Рис.11 - Окно окончания игры

**Заключение**

В процессе выполнения работы был произведён анализ и изучения основ работы с Unity и Photoshop, также реализация графического интерфейса, механики полёта и столкновения с трубами.

Это реализация игры “Flappy Bird” выполненная на платформе “Windows” с использованием среды разработки Unity и созданием своих моделей текстур на редакторе Photoshop. Проект может быть доработан выпуском на другие платформы и введением рекордного счёта.

**Вывод**

Во время разработки игры “Flappy Bird” я познакомился со средой разработки Unity, а также графическим редактором Photoshop, выявил для себя преимущества в понятном и удобном интерфейсе, работе со скриптами, импорта спрайтов и сборки проекта Unity. Также отметил многофункциональность и удобство работы с 2D пиксельной графикой.

**Список использованных источников**

1. Руководство от Microsoft по Unity - URL: <https://dotnet.microsoft.com/en-us/apps/games/unity>
2. Руководство пользователя Unity - URL: <https://docs.unity3d.com/Manual/UnityManual.html>
3. Документация по C# - URL: <https://learn.microsoft.com/en-us/dotnet/csharp/?WT.mc_id=dotnet-35129-website>

**Приложение А**

**Исходный код программы**

**Скрипт “GameManager”:**

using UnityEngine;

using UnityEngine.SceneManagement;

public class GameManager : MonoBehaviour

{

public GameObject gameOverCanvas;

private void Start()

{

gameOverCanvas.SetActive(false);

Time.timeScale = 1;

}

public void GameOver()

{

gameOverCanvas.SetActive(true);

Time.timeScale = 0;

}

public void Replay()

{

SceneManager.LoadScene(0);

}

}

**Скрипт “SpawnPipeScript”:**

using UnityEngine;

public class SpawnPipeScript : MonoBehaviour

{

public float maxTime = 1;

private float timer = 0;

public int destroyTimer;

public GameObject pipe;

public float height;

void Start()

{

}

void Update()

{

if (timer > maxTime) {

GameObject newPipe = Instantiate(pipe);

newPipe.transform.position = transform.position + new Vector3(0, Random.Range(-height, height), 0);

Destroy(newPipe, destroyTimer);

timer = 0;

}

timer += Time.deltaTime;

}

}

**Скрипт “MovePipesScript”:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UIElements;

public class MovePipesScript : MonoBehaviour

{

public float speed;

void Start()

{

}

void Update()

{

transform.position += Vector3.left \* speed \* Time.deltaTime;

}

}

**Скрипт “Fly Slime Script”:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class FlySlimeScript : MonoBehaviour

{

public GameManager gameManager;

public float velocity;

public float rotationSpeed;

public Rigidbody2D rb;

void Start()

{

}

void Update()

{

if (Input.GetMouseButtonDown(0) || Input.GetKeyDown(KeyCode.Space))

rb.velocity = Vector2.up \* velocity;

}

private void FixedUpdate()

{

transform.rotation = Quaternion.Euler(0, 0, rb.velocity.y \* rotationSpeed);

}

private void OnCollisionEnter2D(Collision2D collision)

{

gameManager.GameOver();

}

}

**Скрипт “Score”:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Score : MonoBehaviour

{

public static int score = 0;

private void Start()

{

score = 0;

}

private void Update()

{

GetComponent<UnityEngine.UI.Text>().text = score.ToString();

}

}

**Скрипт “Add Score Script”:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class addScoreScript : MonoBehaviour

{

private void OnTriggerEnter2D(Collider2D collision)

{

Score.score++;

}

}

**Приложение Б**

**Результат работы программы**

