МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Ижевский государственный технический университет имени М.Т. Калашникова»

(ФГБОУ ВО «ИжГТУ имени М.Т. Калашникова»)

Факультет «Информатика и вычислительная техника»

Кафедра «Программное обеспечение»

Направление подготовки 09.03.04 «Программная инженерия»

ОТЧЕТ ПО ПРАКТИКЕ

|  |
| --- |
|  |
| *(наименование типа практики)* |
| *(полное наименование профильной организации)*  Выполнил обучающийся /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / |
| *(подпись) (ФИО, курс, номер группы)*  Дата сдачи отчета: « » 201 г. |
| Дата аттестации « » 201 г. |
| Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Руководитель практики от  ИЖГТУ имени М.Т. Калашникова / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ |
| *(подпись) (И.О. Фамилия, должность, ученая степень)*  Заведующий кафедрой /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ |
| *(подпись) (И.О. Фамилия, должность, ученая степень)* |

**ВВЕДЕНИЕ**

Актуальность: Разработка компьютерных игр – одна из самых быстрорастущих областей IT-индустрии, что делает навыки в этой сфере крайне востребованными. Игра «Flappy Bird» выбрана для проекта благодаря своей популярности и простоте, что позволяет изучить ключевые аспекты создания игр, такие как механики, физика и анимация. Проект по созданию «Flappy Bird» помогает развить фундаментальные навыки разработки игр, актуальные в условиях роста популярности мобильных игр и быстрой эволюции технологий.

Цель: создание функциональной версии игры «Flappy Bird», демонстрирующей мои навыки в области программирования, проектирования игровых механик и использования инструментов для разработки игр. В процессе разработки я планировал изучить и применить различные технологии и методы, необходимые для создания игры.

Задачи:

1. Изучение и выбор подходящего игрового движка для реализации проекта.
2. Разработка концепции игры, включающей основные механики и графику.
3. Программирование основных игровых механик, таких как управление персонажем, генерация препятствий и обработка столкновений.
4. Создание и интеграция графических элементов, обеспечивающих визуальную составляющую игры.
5. Проведение тестирования и отладки игры для выявления и исправления возможных ошибок.
6. Подготовка документации, описывающей процесс разработки, использованные технологии и достигнутые результаты.

1. **ОСНОВНАЯ ЧАСТЬ**

1.1 Словесное описание

Проект по разработке игры «Flappy Bird» был выполнен в рамках учебной практики с целью освоения базовых принципов разработки компьютерных игр и приобретения практических навыков программирования и проектирования игровых механик.

Постановка задачи

Задачей проекта является создание функциональной версии игры «Flappy Bird», включающей разработку всех основных игровых элементов: управление персонажем, генерацию препятствий, обработку столкновений и отображение игровой статистики. Основные этапы выполнения проекта включают:

1. Изучение и выбор игрового движка.
2. Проектирование игрового процесса и графики.
3. Реализация основных механик игры.
4. Интеграция графических элементов.
5. Проведение тестирования и отладки игры.

Входные данные

Для выполнения проекта были использованы следующие входные данные и инструменты:

1. Игровой движок Unity.
2. Графические ресурсы для персонажа, препятствий и фона.
3. Руководства по пользованию движком Unity.

Ожидаемый результат

В результате выполнения проекта ожидается получить полноценную игру «Flappy Bird» с следующими характеристиками:

1. Игрок управляет персонажем, который избегает препятствия на своем пути.
2. Препятствия генерируются случайным образом и двигаются с постоянной скоростью.
3. Столкновение с препятствием приводит к завершению игры.
4. Игра имеет понятный интерфейс и управление.
5. Графика соответствует оригинальной игре «Flappy bird».

1.2 Математическая постановка

Проект по разработке игры "Flappy Bird" включает использование ряда математических методов и подходов, которые обеспечивают корректную работу игровых механик. В данном разделе рассматриваются основные математические аспекты, применяемые при разработке этой игры.

Обоснование математического метода решения

Основные математические задачи в разработке игры "Flappy Bird" связаны с реализацией движения персонажа, генерацией препятствий и обработкой столкновений. Для этого используются методы аналитической геометрии, теории вероятностей и численного анализа.

Применяемые утверждения:

1. Прыжок

При каждом нажатии на экран вертикальная скорость птицы мгновенно изменяется, задаваясь определенным значением .

1. Генерация препятствий

Препятствия генерируются случайным образом, и их вертикальная позиция определяется случайной величиной , которая может быть задана с помощью равномерного распределения:

Где a и b - минимальное и максимальное значения вертикальной позиции препятствия.

1. Движение птицы

Птица движется под действием силы тяжести и силы, создаваемой игроком при нажатии на экран (прыжок).

Формулировка теорем и аксиом:

1. Теорема о равномерном движении:

В отсутствие внешних сил (например, сила прыжка) объект (птица) движется с постоянным ускорением (сила тяжести).

1. Теорема о случайном распределении.

При генерации препятствий с помощью равномерного распределения, каждое значение в заданном диапазоне [a,b] имеет равную вероятность быть выбранным.

Применение этих математических методов и формул позволяет обеспечить корректное и реалистичное поведение игровых объектов, что является основой для создания увлекательного игрового процесса в "Flappy Bird".

1.3 Описание программной реализации

Программная реализация игры "Flappy Bird" включает несколько основных компонентов: управление персонажем, генерация препятствий, обработка столкновений и отображение игрового интерфейса. Ниже приведено описание алгоритма и основные шаги реализации проекта.

Алгоритм решения поставленной задачи в виде словесного описания:

1. Инициализация игры:

* Настройка игрового окна и основных параметров.
* Загрузка графических ресурсов.

1. Главный игровой цикл:

* Обработка пользовательского ввода.
* Обновление состояния игрового мира.
* Генерация и отрисовка игровых объектов.
* Проверка условий завершения игры.

1. Управление персонажем(птицей):

* Обработка нажатия на кнопку для прыжка.
* Обновлении позиции птицы.

1. Генерация и обновление препятствий:

* Создание новых препятствий через опредленные интервалы времени.
* Обновление позиций существующих препятствий.
* Удаление препятствий, вышедших за границы игрового окна.

1. Обработка столкновений:

* Проверка пересечения птицы с препятствием.
* Проверка пересечения птицей границами игрового окна.
* Завершение игры при обнаружении столкновения.

1. Отображение игрового интерфейса:

* Отрисовка птицы, препятствий и фона.
* Отображение кнопки начала игры и других элементов интерфейса.
  1. Полный листинг программы

1. Скрипт игрока(player.cs)

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

// Класс Player отвечает за управление поведением игрока (птицы) в игре.

public class Player : MonoBehaviour

{

// Переменная для хранения скорости прыжка.

public float velocity = 2.4f;

// Приватная переменная для ссылки на Rigidbody2D компонента.

private Rigidbody2D rigidbody;

// Ссылка на объект GameManager, чтобы управлять игровыми событиями.

public GameManager gameManager;

// Переменная для отслеживания состояния игрока (жив или мертв).

public bool isDead = false;

// Метод, который вызывается при столкновении с другим объектом.

private void OnCollisionEnter2D(Collision2D collision)

{

// Устанавливаем флаг, что игрок мертв.

isDead = true;

// Вызываем метод GameOver() из GameManager, чтобы завершить игру.

gameManager.GameOver();

}

// Метод Start вызывается перед первым обновлением кадра.

void Start()

{

// Получаем компонент Rigidbody2D, прикрепленный к объекту игрока.

rigidbody = GetComponent<Rigidbody2D>();

}

// Метод Update вызывается один раз за кадр.

void Update()

{

// Проверяем, была ли нажата клавиша пробела.

if (Input.GetKeyDown("space"))

{

// Устанавливаем вертикальную скорость игрока, чтобы он прыгнул вверх.

rigidbody.velocity = Vector2.up \* velocity;

}

}

}

1. Скрипт для движения препятствий:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

// Класс obstacle отвечает за поведение препятствий в игре.

public class obstacle : MonoBehaviour

{

// Переменная для хранения скорости перемещения препятствия.

public float speed;

// Метод Update вызывается один раз за кадр.

void Update()

{

// Перемещаем препятствие влево с заданной скоростью.

// Умножаем скорость на Time.deltaTime, чтобы обеспечить плавное перемещение независимо от частоты кадров.

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

}

}

1. Скрипт для генерации новых препятствий(spawner.cs)

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

// Класс spawner отвечает за генерацию препятствий в игре.

public class spawner : MonoBehaviour

{

// Время между появлением новых препятствий.

public float queueTime = 3f;

// Внутренняя переменная для отслеживания времени.

private float time = 0f;

// Префаб препятствия, который будет создан.

public GameObject obstacle;

// Высота, в пределах которой препятствия могут появляться случайным образом.

public float height;

// Метод Update вызывается один раз за кадр.

void Update()

{

// Если прошло достаточно времени с последнего появления препятствия.

if (time > queueTime)

{

// Создаем новое препятствие.

GameObject go = Instantiate(obstacle);

// Устанавливаем позицию нового препятствия с учетом случайной высоты.

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

// Сбрасываем таймер.

time = 0;

// Уничтожаем препятствие через 10 секунд, чтобы предотвратить переполнение памяти.

Destroy(go, 10);

}

// Увеличиваем таймер на время, прошедшее с последнего кадра.

time += Time.deltaTime;

}

}

1. Скрипт для начала, окончания игры(GameManager.cs)

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.SceneManagement;

// Класс GameManager отвечает за управление состоянием игры.

public class GameManager : MonoBehaviour

{

// Кнопка старта игры.

public GameObject startButton;

// Ссылка на игрока.

public Player player;

// Текстовый элемент для отображения отсчета времени до завершения игры.

public Text gameOverCountdown;

// Таймер для отсчета времени после завершения игры.

public float countTimer = 5;

// Метод Start вызывается перед первым обновлением кадра.

void Start()

{

// Скрываем текст отсчета до завершения игры.

gameOverCountdown.gameObject.SetActive(false);

// Останавливаем время в игре.

Time.timeScale = 0;

}

// Метод Update вызывается один раз за кадр.

private void Update()

{

// Проверяем, мертв ли игрок.

if (player.isDead)

{

// Отображаем текст отсчета до завершения игры.

gameOverCountdown.gameObject.SetActive(true);

// Уменьшаем таймер отсчета.

countTimer -= Time.unscaledDeltaTime;

}

// Обновляем текст отсчета.

gameOverCountdown.text = "Game Over";

// Если таймер отсчета дошел до нуля, перезапускаем игру.

if (countTimer < 0)

{

RestartGame();

}

}

// Метод для запуска игры.

public void StartGame()

{

// Скрываем кнопку старта.

startButton.SetActive(false);

// Запускаем время в игре.

Time.timeScale = 1;

}

// Метод для завершения игры.

public void GameOver()

{

// Останавливаем время в игре.

Time.timeScale = 0;

}

// Метод для перезапуска игры.

public void RestartGame()

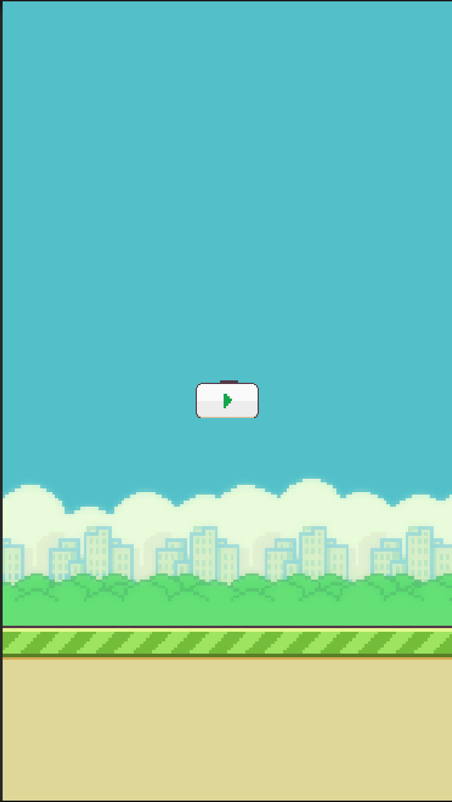
{

// Перезагружаем сцену с именем "fdfh".

SceneManager.LoadScene("fdfh");

}

}

Скриншоты игры:



**ЗАКЛЮЧЕНИЕ**

Работа над проектом по созданию игры "Flappy Bird" позволила мне значительно углубить свои знания в области разработки игр на платформе Unity. В процессе выполнения проекта я получил ценные практические навыки, которые включают управление игровыми объектами, обработку пользовательского ввода, генерацию препятствий.

Одним из наиболее важных аспектов работы было понимание и реализация механики движения персонажа. Управление птицей с помощью силы гравитации и прыжков потребовало использования базовых принципов физики и математики. Реализация обработки столкновений с препятствиями и границами экрана была еще одной важной задачей.

Особенно полезным опытом было использование Unity для управления игровыми объектами и сценами. Я научился создавать и настраивать препятствия, управлять состоянием игры с помощью, а также использовать различные методы Unity для обновления и отображения состояния игры в реальном времени.

Проект также помог мне развить навыки программирования на языке C#. Написание скриптов дало мне лучшее понимание синтаксиса и особенностей C#.

В будущем можно улучшить этот проект, добавив новые уровни сложности, расширив функционал игры и добавив новые элементы интерфейса. Например, можно добавить систему очков и достижений, разнообразить дизайн и анимацию, а также реализовать поддержку многопользовательского режима.

В целом, работа над проектом принесла мне много положительных впечатлений. Я получил ценные знания и навыки, которые непременно пригодятся мне в будущем.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1) Руководство Unity на Русском // UnityHub URL: https://unityhub.ru/manual/index (дата обращения: 22.05.2024).

2) Flappy Bird Sprites // WeeklyHow URL: https://weeklyhow.com/downloads/11-24-19/Flappy%20Bird%20Sprites.png (дата обращения: 22.05.2024).

3) Управление игровыми объектами // Unity Documentation URL: https://docs.unity3d.com/ru/530/Manual/ControllingGameObjectsComponents.html (дата обращения: 22.05.2024).

4) Создание и уничтожение игровых объектов // Unity Documentation URL: https://docs.unity3d.com/ru/530/Manual/CreateDestroyObjects.html (дата обращения: 23.05.2024).