Федеральное государственное автономное образовательное учреждение
высшего образования
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра «Информационные системы и технологии»

Направление подготовки/ специальность: 09.03.02 «Информационные системы и технологии»

ОТЧЕТ

по проектной практике

Студент: Кривоносова Варвара Владимировна_____ Группа: __241-337____

Место прохождения практики: Московский Политех, кафедра «ИиИТ»

Отчет принят с оценкой _____ Дата _____

Руководитель практики: Меньшикова Наталия Павловна_____

Москва 2025

Отчет по вариативной части проектной (учебной) практике

Вариативная часть задания включает в себя:

1. Практическая реализация технологии:

- Выполнить все задачи базовой части.
- Выбрать любую технологию (Build a C#/WPF RPG). Проведите исследование: изучите, как создать выбранную технологию с нуля, воспроизведите практическую часть.
- Создать подробное описание в формате Markdown, включающее:
    - Последовательность действий по исследованию предметной области и созданию технологии.
    - Напишите техническое руководство по созданию этой технологии, ориентированное на начинающих.
    - Включите в руководство:
    - Пошаговые инструкции.
    - Примеры кода.
- Создание технического руководства по созданию проекта на выбранную тему;
- Создание видеопрезентации выполненной работы;
- Написание документации проекта в формате Markdown;
- Подготовка финального отчёта.

Финальный отчет по разработке игры "Flappy Bird" на WPF (C#)

Разработка игры "Flappy Bird" на платформе WPF (Windows Presentation Foundation) с использованием языка C# представляла собой комплексный процесс, включавший несколько последовательных этапов.

На начальном подготовительном этапе было проведено детальное изучение механики оригинальной игры Flappy Bird, что позволило выделить ключевые элементы геймплея, такие как управление птицей, система физики с гравитацией, а также алгоритм генерации и движения препятствий. Особое внимание было уделено проектированию архитектуры будущего приложения, где центральное место заняли такие компоненты как игровой цикл на основе DispatcherTimer, система обработки столкновений с использованием класса Rect и метода IntersectsWith, а также механизм обработки пользовательского ввода для реагирования на нажатия клавиш Space и R. Параллельно велась работа по подготовке графических ресурсов, включая создание и подбор изображений для персонажа птицы, декоративных облаков, труб-препятствий и фоновых элементов.

Основной этап разработки начался с реализации базового функционала в среде WPF. Было создано главное окно приложения MainWindow.xaml, использующее Canvas в качестве контейнера для всех игровых элементов. Каждый графический объект, включая птицу, трубы и облака, был представлен элементом Image с соответствующими тегами (obs1, obs2, obs3 для труб и clouds для облаков), что впоследствии упростило их обработку в коде.

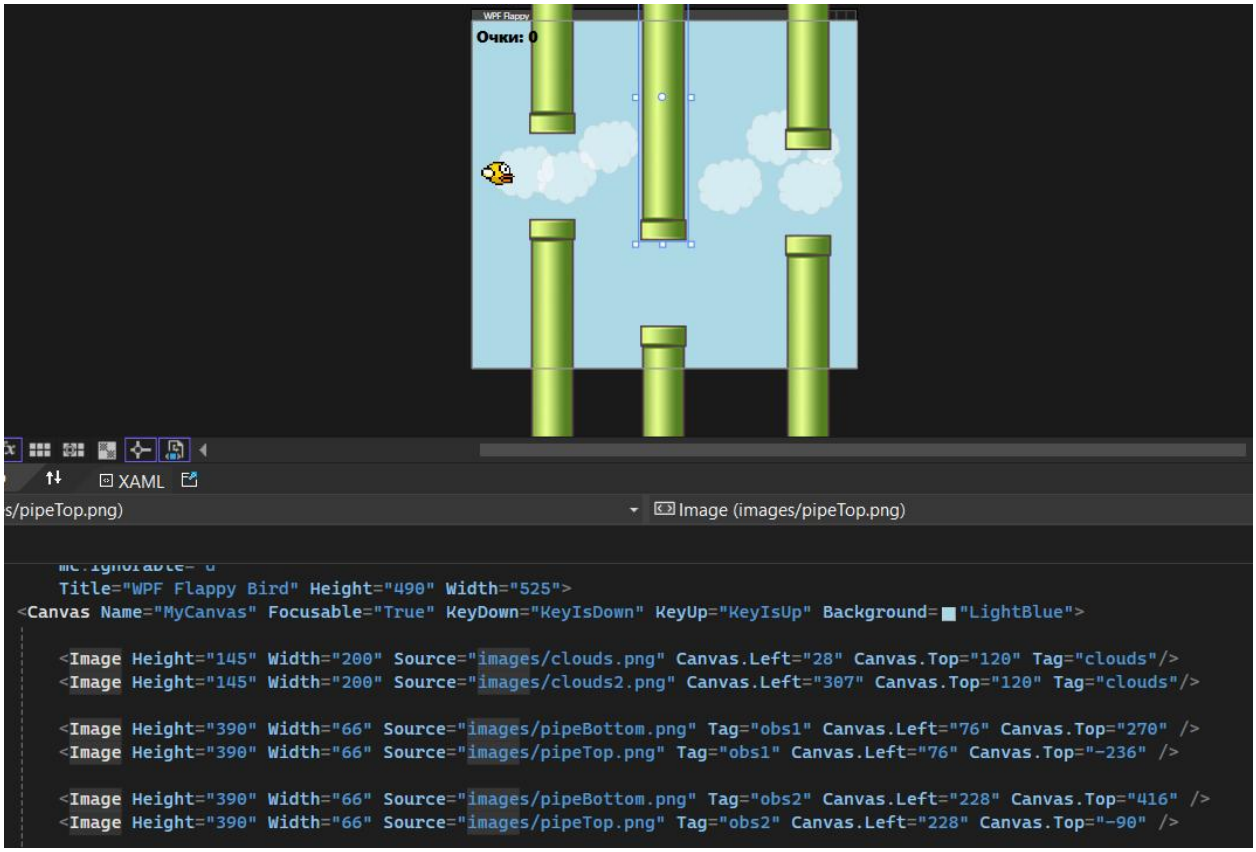

Рисунок 1 Расположение игровых объектов на Canvas: птица (flappyBird), трубы-препятствия (obs1, obs2, obs3) и декоративные облака (clouds).

Сердцем игровой логики стал DispatcherTimer с интервалом 20 миллисекунд, обеспечивающий плавную анимацию на 50 кадрах в секунду.

```
public partial class MainWindow : Window
{
    //Основной игровой цикл
    DispatcherTimer gameTimer = new DispatcherTimer();
    double score;
    int gravity = 8;    // Базовая гравитация (падение птицы вниз)
    bool gameOver;
    Rect flappyBirdHitBox;
```

Рисунок 2 Инициализация игрового цикла (DispatcherTimer) и объявление ключевых переменных для управления состоянием игры.

В обработчике таймера MainEventTimer была реализована основная игровая механика: движение птицы под действием гравитации, перемещение препятствий и облаков, а также постоянная проверка столкновений.

```
private void MainEventTimer(object sender, EventArgs e)
{
    txtScore.Content = "Очки: " + score;

    flappyBirdHitBox = new Rect(Canvas.GetLeft(flappyBird), Canvas.GetTop(flappyBird), flappyBird.Width - 12, flappyBird.Height);

    Canvas.SetTop(flappyBird, Canvas.GetTop(flappyBird) + gravity);

    if (Canvas.GetTop(flappyBird) < -30 || Canvas.GetTop(flappyBird) + flappyBird.Height > 460)
    {
        EndGame();
    }

    foreach (var x in MyCanvas.Children.OfType<Image>())
    {
        if ((string)x.Tag == "obs1" || (string)x.Tag == "obs2" || (string)x.Tag == "obs3")
        {
            Canvas.SetLeft(x, Canvas.GetLeft(x) - 5);

            if (Canvas.GetLeft(x) < -100)
            {
                Canvas.SetLeft(x, 800);

                score += .5;
            }

            Rect PillarHitBox = new Rect(Canvas.GetLeft(x), Canvas.GetTop(x), x.Width, x.Height);

            if (flappyBirdHitBox.IntersectsWith(PillarHitBox))
            {
                EndGame();
            }
        }
        if ((string)x.Tag == "clouds")
        {
            Canvas.SetLeft(x, Canvas.GetLeft(x) - 1);
            if (Canvas.GetLeft(x) < -250)
            {
                Canvas.SetLeft(x, 550);
                score += .5;
            }
        }
    }
```

Рисунок 3 Алгоритм работы игрового цикла: применение гравитации к птице (Canvas.SetTop), перемещение труб (Canvas.SetLeft) и проверка пересечений (IntersectsWith).

Система управления была построена на обработчиках событий KeyIsDown и KeyIsUp, где клавиша Space отвечала за "прыжок" птицы (временное изменение гравитации на отрицательное значение), а клавиша R - за перезапуск игры после проигрыша.

```
private void KeyIsDown(object sender, KeyEventArgs e)
{
    if (e.Key == Key.Space)
    {
        flappyBird.RenderTransform = new RotateTransform(-20, flappyBird.Width / 2, flappyBird.Height / 2);
        gravity = -8;           // "Прыжок" - временная отрицательная гравитация
    }

    if (e.Key == Key.R && gameOver == true)
    {
        StartGame();
    }
}

Ссылок: 1
private void KeyIsUp(object sender, KeyEventArgs e)
{
    flappyBird.RenderTransform = new RotateTransform(5, flappyBird.Width / 2, flappyBird.Height / 2);

    gravity = 8;            // Возврат к обычной гравитации
}
```

Рисунок 4 Обработчики событий клавиатуры KeyIsDown и KeyIsUp с изменением гравитации (gravity) и поворотом птицы (RotateTransform).

На этапе реализации игровой механики особое внимание было уделено физике движения и системе коллизий. Гравитационный эффект достигался постоянным увеличением координаты Y птицы на значение переменной gravity (равное 8 по умолчанию). Для более точного и "честного" геймплея была реализована система HitBox, где область столкновений птицы сознательно делалась на 12 пикселей уже оригинального изображения. Проверка столкновений выполнялась как с границами игрового поля (верхняя и нижняя границы Canvas), так и с препятствиями-трубами. Система начисления очков была построена на принципе "прохождения" между трубами - за каждую успешно преодоленную пару труб игрок получал 0.5 очка, что отображалось в реальном времени в элементе Label txtScore.

Механизм генерации препятствий обеспечивал их постоянное движение слева направо с последующим "телепортированием" обратно за правую границу экрана при полном исчезновении из видимой области.

```
private void StartGame()
{
    MyCanvas.Focus();

    int temp = 300;

    score = 0;

    gameOver = false;

    Canvas.SetTop(flappyBird, 190);

    foreach (var x in MyCanvas.Children.OfType<Image>())
    {
        if ((string)x.Tag == "obs1")
        {
            Canvas.SetLeft(x, 500);
        }
        if ((string)x.Tag == "obs2")
        {
            Canvas.SetLeft(x, 800);
        }
        if ((string)x.Tag == "obs3")
        {
            Canvas.SetLeft(x, 1100);
        }

        if ((string)x.Tag == "clouds")
        {
            Canvas.SetLeft(x, 300 + temp);
            temp = 800;
        }
    }
```

Рисунок 5 Механизм генерации препятствий

Этап тестирования и отладки включал тщательную проверку всех игровых механик. Были проведены многочисленные тесты для балансировки геймплея, в ходе которых подбирались оптимальные значения скорости движения препятствий, силы гравитации и параметров HitBox. Особое внимание уделялось плавности анимации и отсутствию визуальных артефактов при движении объектов. В процессе отладки были выявлены и исправлены различные проблемы, включая некорректное определение столкновений в угловых случаях и редкие ситуации с одновременным наложением нескольких объектов.

На финальном этапе разработки были проведены работы по улучшению визуальной составляющей игры. Реализована анимация поворота птицы при помощи RotateTransform, что добавляло реалистичности движениям персонажа. Интерфейс игры был дополнен стилизованным отображением счета с использованием крупного жирного шрифта. Логика управления состоянием игры была вынесена в отдельные методы StartGame() и EndGame(), что сделало код более структурированным и читаемым. StartGame() отвечал за сброс всех позиций и параметров в начальное состояние, в то время как EndGame() останавливал игровой процесс и выводил сообщение о завершении игры.

В результате проведенной работы был создан полностью функциональный клон игры Flappy Bird, демонстрирующий плавную анимацию, точную физику движения и корректную систему подсчета очков. Все элементы игры работают согласованно, обеспечивая стабильный игровой процесс. Код проекта был тщательно документирован, что облегчает его дальнейшую поддержку и развитие. В процессе разработки были глубоко освоены ключевые аспекты работы с WPF, включая использование XAML для описания интерфейса, работу с графическими элементами, обработку пользовательского ввода и реализацию игровой логики на C#. Полученный опыт может служить основой для создания более сложных игровых проектов с использованием технологий .NET.

Листинг программы:

Код в окне Mainwindow.xaml:

```
<Window x:Class="WPF_Flappy_Bird.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:WPF_Flappy_Bird"
        mc:Ignorable="d"
        Title="WPF Flappy Bird" Height="490" Width="525">
    <Canvas Name="MyCanvas" Focusable="True" KeyDown="KeyIsDown" KeyUp="KeyIsUp"
Background="LightBlue">

        <Image Height="145" Width="200" Source="images/clouds.png" Canvas.Left="28"
Canvas.Top="120" Tag="clouds"/>
        <Image Height="145" Width="200" Source="images/clouds2.png"
Canvas.Left="307" Canvas.Top="120" Tag="clouds"/>

        <Image Height="390" Width="66" Source="images/pipeBottom.png" Tag="obs1"
Canvas.Left="76" Canvas.Top="270" />
        <Image Height="390" Width="66" Source="images/pipeTop.png" Tag="obs1"
Canvas.Left="76" Canvas.Top="-236" />

        <Image Height="390" Width="66" Source="images/pipeBottom.png" Tag="obs2"
Canvas.Left="228" Canvas.Top="416" />
        <Image Height="390" Width="66" Source="images/pipeTop.png" Tag="obs2"
Canvas.Left="228" Canvas.Top="-90" />

        <Image Height="390" Width="66" Source="images/pipeBottom.png" Tag="obs3"
Canvas.Left="426" Canvas.Top="292" />
        <Image Height="390" Width="66" Source="images/pipeTop.png" Tag="obs3"
Canvas.Left="426" Canvas.Top="-214" />

        <Image Name="flappyBird" Height="36" Width="50"
Source="/images/flappyBird.png" Stretch="Fill" Canvas.Top="190" Canvas.Left="10" />

        <Label Name="txtScore" FontSize="22" FontWeight="ExtraBold" Content="Очки:
0" />

    </Canvas>
</Window>
```

Код в окне Mainwindow.xaml.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Windows.Threading;

namespace WPF_Flappy_Bird
{
    public partial class MainWindow : Window
    {
        //Основной игровой цикл
        DispatcherTimer gameTimer = new DispatcherTimer();
        double score;
        int gravity = 8;    // Базовая гравитация (падение птицы вниз)
        bool gameOver;
        Rect flappyBirdHitBox;

        public MainWindow()
        {
            InitializeComponent();

            gameTimer.Tick += MainEventTimer;
            gameTimer.Interval = TimeSpan.FromMilliseconds(20);
            StartGame();
        }

        private void MainEventTimer(object sender, EventArgs e)
        {
            txtScore.Content = "Очки: " + score;

            flappyBirdHitBox = new Rect(Canvas.GetLeft(flappyBird),
Canvas.GetTop(flappyBird), flappyBird.Width - 12, flappyBird.Height);

            Canvas.SetTop(flappyBird, Canvas.GetTop(flappyBird) + gravity);

            if (Canvas.GetTop(flappyBird) < -30 || Canvas.GetTop(flappyBird) +
flappyBird.Height > 460)
            {
                EndGame();
            }

            foreach (var x in MyCanvas.Children.OfType<Image>())
            {
                if ((string)x.Tag == "obs1" || (string)x.Tag == "obs2" ||
(string)x.Tag == "obs3")
                {
                    Canvas.SetLeft(x, Canvas.GetLeft(x) - 5);

                    if (Canvas.GetLeft(x) < -100)
                    {
                        Canvas.SetLeft(x, 800);

                        score += .5;
```

```
                    }

                    Rect PillarHitBox = new Rect(Canvas.GetLeft(x),
Canvas.GetTop(x), x.Width, x.Height);

                    if (flappyBirdHitBox.IntersectsWith(PillarHitBox))
                    {
                        EndGame();
                    }
                }
                if ((string)x.Tag == "clouds")
                {
                    Canvas.SetLeft(x, Canvas.GetLeft(x) - 1);
                    if (Canvas.GetLeft(x) < -250)
                    {
                        Canvas.SetLeft(x, 550);
                        score += .5;
                    }
                }
            }
        }

        private void KeyIsDown(object sender, KeyEventArgs e)
        {
            if (e.Key == Key.Space)
            {
                flappyBird.RenderTransform = new RotateTransform(-20,
flappyBird.Width / 2, flappyBird.Height / 2);
                gravity = -8;           // "Прыжок" - временная отрицательная
гравитация
            }

            if (e.Key == Key.R && gameOver == true)
            {
                StartGame();
            }
        }

        private void KeyIsUp(object sender, KeyEventArgs e)
        {
            flappyBird.RenderTransform = new RotateTransform(5, flappyBird.Width /
2, flappyBird.Height / 2);

            gravity = 8;            // Возврат к обычной гравитации
        }

        private void StartGame()
        {
            MyCanvas.Focus();

            int temp = 300;

            score = 0;

            gameOver = false;

            Canvas.SetTop(flappyBird, 190);

            foreach (var x in MyCanvas.Children.OfType<Image>())
            {
                if ((string)x.Tag == "obs1")
                {
                    Canvas.SetLeft(x, 500);
                }
                if ((string)x.Tag == "obs2")
```

```
                {
                    Canvas.SetLeft(x, 800);
                }
                if ((string)x.Tag == "obs3")
                {
                    Canvas.SetLeft(x, 1100);
                }

                if ((string)x.Tag == "clouds")
                {
                    Canvas.SetLeft(x, 300 + temp);
                    temp = 800;
                }
            }

            gameTimer.Start();
        }

        private void EndGame()
        {
            gameTimer.Stop();
            gameOver = true;
            txtScore.Content += " Игра окончена! Нажмите R - начать заново.";

        }
    }
}
```