УДК 004.032.26

Разработка простой игры - настольный теннис в Visual Studio

Ульянов Егор Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье описывается процесс разработки игры -настольного тенниса в Visual Studio с использованием языка программирования C# и библиотеки Windows Presentation Foundation (WPF). Целью данной статьи является создание игры настольный теннис. Практическим результатом является разработанная игра.
Ключевые слова: C#, Visual Studio, игра, настольный тенис

Development of a simple game - table tennis in Visual Studio

Ulianov Egor Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article describes the process of developing a table tennis game in Visual Studio using the C# programming language and the Windows Presentation Foundation (WPF) library. The purpose of this article is to create a table tennis game. The practical result is a developed game.
Keywords: C#, Visual Studio, game, table tennis

1. Введение

1.1 Актуальность исследования

Самым популярным видом досуга многих людей на сегодняшний момент являются компьютерные игры. Виртуальные миры открывают нам огромные возможности: позволяют отыгрывать различные роли, играя за главного героя, многие игры способствуют развитию интеллекта, внимания, реакции, пространственной ориентации и логического мышления. Разработкой видеоигр может заниматься как один человек, так и команда. Создание игры - это продолжительный и трудоёмкий процесс, состоящий из самых разнообразных этапов, включающий в себя как технические, так и творческие моменты.

1.2 Обзор исследований

В своей работе И. Ю. Просвирнина создала приложение «Морской бой», обладающее игровым искусственным интеллектом, в котором предусмотрен режим игры «Игрок против компьютера» [1]. В статье Н. А. Базеевой и

Д.С.Лебедева описано начало игровой индустрии, а также рассмотрены особенности языков программирования для разработки игр [2]. В своей работе Р.Ф. Гайнуллин, В.А. Захаров, Е.А. Аксенова изучили инструмент для разработки двух- и трёхмерных игр – Unity 3D [3]. В.И. Вахрушев в работе представил подробную инструкцию по разработке игры «Расстановка мебели» на движке Unity с использованием языка программирования C# [4].

1.3 Цель исследования

Цель исследования – применяя возможности среды разработки Visual Studio и языка программирования C#, создать игру настольный теннис, со всеми, необходимыми для полноценной игры, механиками.

2. Материалы и методы

Для создания игры будем использовать программное обеспечение Visual Studio, а также язык программирования C#.

Для проекта использовалась Windows Presentation Foundation (WPF), платформа пользовательского интерфейса для создания клиентских приложений для настольных систем. Платформа разработки WPF поддерживает широкий набор компонентов для разработки приложений, включая модель приложения, ресурсы, элементы управления, графику, макет, привязки данных, документы и безопасность. Эта платформа является частью платформы .NET.

3 Результаты и дискуссия

Начнём разработку игры, с создания WPF приложения. Сразу назовем проект «Pong» (рис. 1).

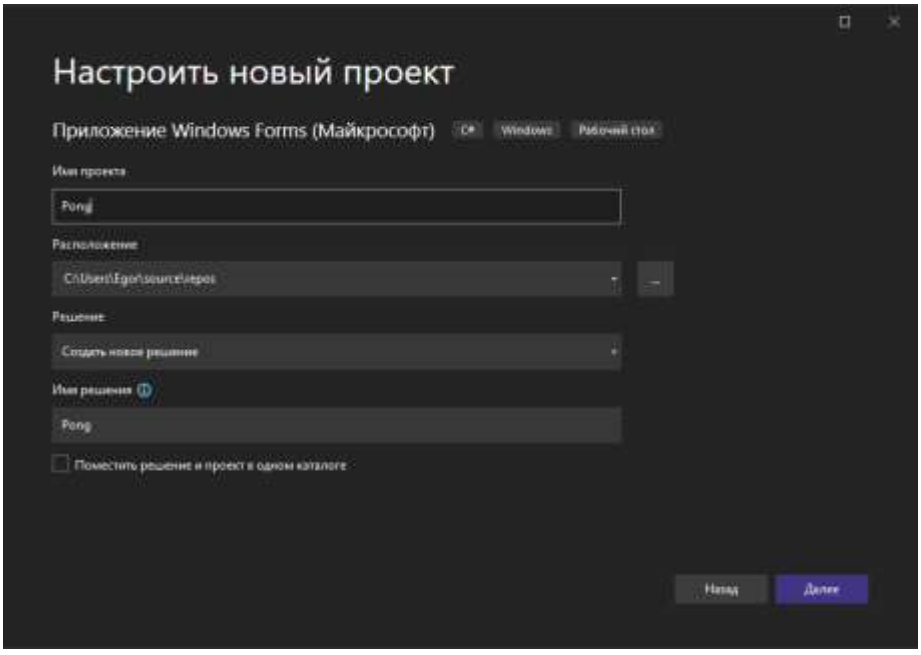

Рисунок 1. Создание проекта

Для начала разместим необходимые компоненты на форму (рис.2).

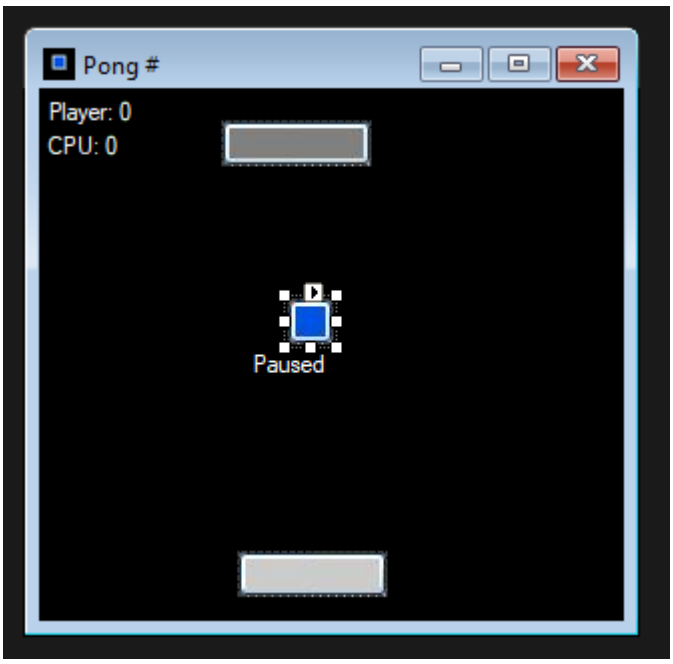

Рисунок 2. Добавление компонентов на форму

Далее приступаем к написанию кода. Переменные, объявленные в начале, используются для хранения скорости шара, последней позиции мыши и позиции ракетки компьютера, а также для хранения счета и границ поля. Конструктор Form1() выполняет инициализацию игры - устанавливает начальные значения переменных, делает кнопки не кликабельными, скрывает текст паузы и настраивает двойную буферизацию, чтобы экран не мерцал при отрисовке (рис.3-4).

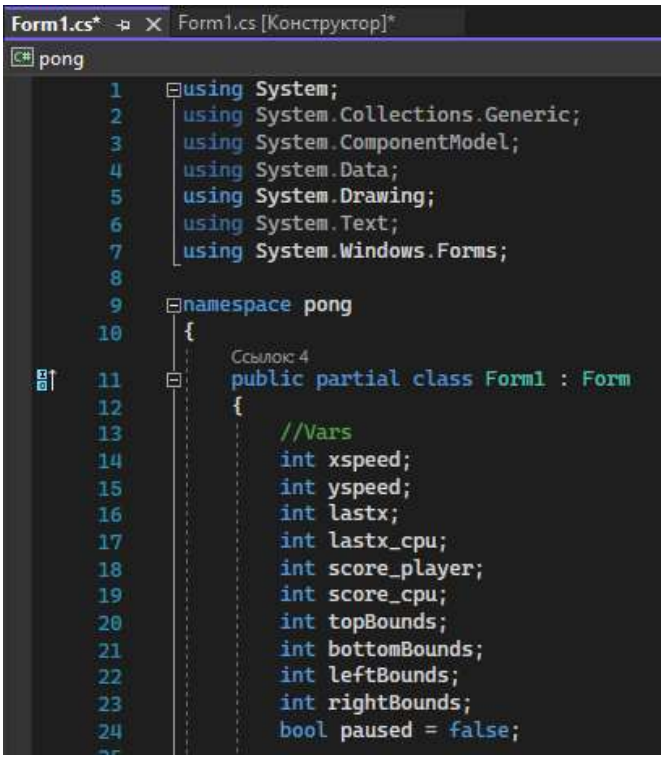

Рисунок 3. Создание переменных

```
Ссылок: 1
public Form1()
{
    InitializeComponent();


    xspeed = 2;
    yspeed = 2;


    ball.Enabled = false;
    paddle.Enabled = false;


    lastx = MousePosition.X;
    lastx_cpu = paddle2.Location.X;


    score_player = 0;
    score_cpu = 0;


    topBounds = 0;
    bottomBounds = this.Height;
    leftBounds = 0;
    rightBounds = this.Width;


    pause_txt.Visible = false;


    this.SetStyle(ControlStyles.UserPaint | ControlStyles.AllPaintingInWmPaint | ControlStyles.OptimizedDoubleBuffer, true);
}
```

Рисунок 4. Добавление необходимых функций

Теперь переходим к логике игры, код на рисунке 5-8 обрабатывает движение мяча и ракеток, изменение скорости мяча при ударе о ракетку, обработку выхода мяча за границы игрового поля и изменение очков игроков. Код использует таймер (событие EventArgs e), который вызывается регулярно и вызывает метод moveBall(). Также код проверяет, не приостановлена ли игра - переменная paused.

```
Ссылок: 1
private void moveBall(object sender, EventArgs e)
{

    topBounds = 0;
    bottomBounds = this.Height-23;
    leftBounds = 0;
    rightBounds = this.Width;


    if (!paused)
    {

        paddle.Location = new Point((int)(MousePosition.X - paddle.Width), paddle.Location.Y);
        ball.Location = new Point(ball.Location.X + xspeed, ball.Location.Y + yspeed);


        if (ball.Location.X > paddle2.Location.X)
        {
            paddle2.Location = new Point(paddle2.Location.X + 3, paddle2.Location.Y);
        }
        else
        {
            paddle2.Location = new Point(paddle2.Location.X - 3, paddle2.Location.Y);
        }


        if (ball.Location.X < leftBounds)
        {
            xspeed *= -1;
            while (ball.Location.X - 1 < leftBounds)
            {
                ball.Location = new Point(ball.Location.X + 1, ball.Location.Y);
            }
        }


        if (ball.Location.X + ball.Width > rightBounds)
        {
            xspeed *= -1;
            while (ball.Location.X + 1 > rightBounds)
            {
                ball.Location = new Point(ball.Location.X - 1, ball.Location.Y);
            }
        }
```

Рисунок 5. Написание основной логики игры

Рисунок 6. Написание основной логики игры

```
if (ball.Location.Y < paddle2.Location.Y + paddle2.Height && ball.Location.X > (int)(paddle2.Location.X - ball.Width / 2))
{
    yspeed *= -1;
    xspeed = Math.Abs(paddle.Location.X - lastx_cpu);
    if (xspeed > 4)
    {
        xspeed = 4;
    }
    else if (xspeed < -4)
    {
        xspeed = -4;
    }
    else if (xspeed == 0)
    {
        Random a = new Random();
        if (a.NextDouble() > .5)
        {
            xspeed = 2;
        }
        else
        {
            xspeed = -2;
        }
    }
    while (ball.Location.Y - 1 < paddle2.Location.Y + paddle2.Height)
    {
        ball.Location = new Point(ball.Location.X, ball.Location.Y + 1);
    }
}
```

Рисунок 7. Написание основной логики игры

```
if (ball.Location.Y > bottomBounds)
{
    ball.Location = new Point(120, 100);
    Random b = new Random();
    if (b.NextDouble() > .5)
    {
        xspeed = 2;
    }
    else
    {
        xspeed = -2;
    }
    yspeed = -2;
    score_cpu++;
    points2.Text = "CPU: " + score_cpu;
}
else if (ball.Location.Y < topBounds)
{
    ball.Location = new Point(120, 100);
    Random b = new Random();
    if (b.NextDouble() > .5)
    {
        xspeed = 2;
    }
    else
    {
        xspeed = -2;
    }
    yspeed = 2;
    score_player++;
    points1.Text = "Player: " + score_player;
}
lastx = MousePosition.X;
lastx_cpu = paddle2.Location.X;
```

Рисунок 8. Написание основной логики игры

Создаем функцию, отвечающую за перемещение платформ, паузу в игре и обработку событий. Метод movePaddles отвечает за перемещение платформ и передвигает по верхней и нижней части игрового поля. Метод pause переключает флаг paused на противоположное состояние и отображает текстовый блок pause_txt, если игра на паузе (рис.9).

```
private void movePaddles(object sender, EventArgs e)
{

    paddle.Location = new Point(paddle.Location.X, bottomBounds - 46);
    paddle2.Location = new Point(paddle2.Location.X,topBounds + 12);
    pause_txt.Location = new Point((int)rightBounds / 2 - pause_txt.Width / 2, (int)bottomBounds / 2 - pause_txt.Height / 2);
}

Ссылок: 1
private void pause(object sender, EventArgs e)
{

    if (!paused)
    {
        paused = true;
        pause_txt.Visible = true;
    }
    else
    {
        paused = false;
        pause_txt.Visible = false;
    }
}

Ссылок: 0
protected override void OnPaint(PaintEventArgs pe)
{

}

Ссылок: 1
private void pause_txt_Click(object sender, EventArgs e)
{

}
```

Рисунок 9. Создание функций

Осталось только проверить игру. Жмем кнопку «Play» (рис.10-12).

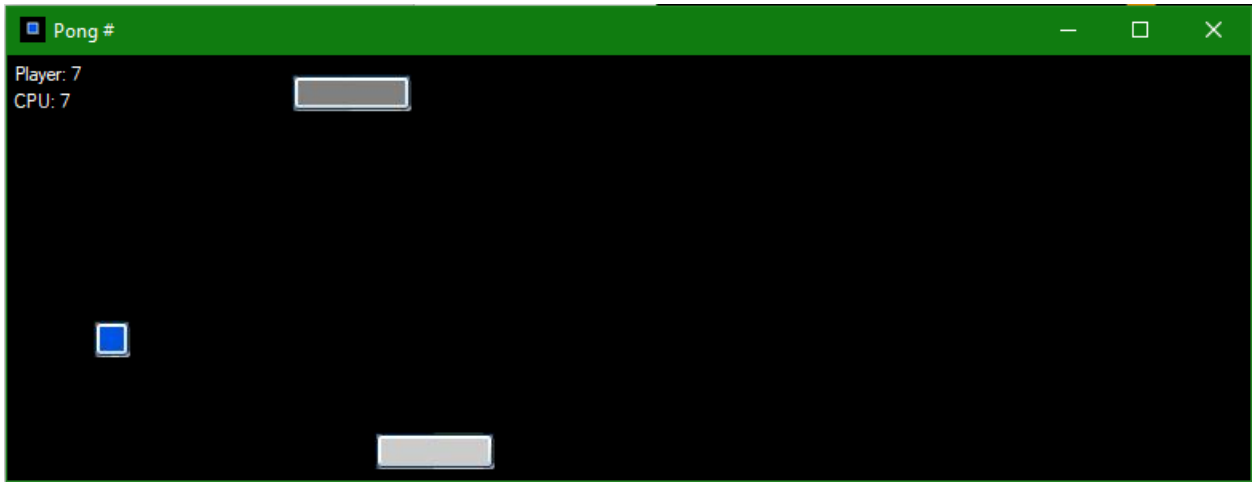

Рисунок 10. Геймплей

Рисунок 11. Геймплей

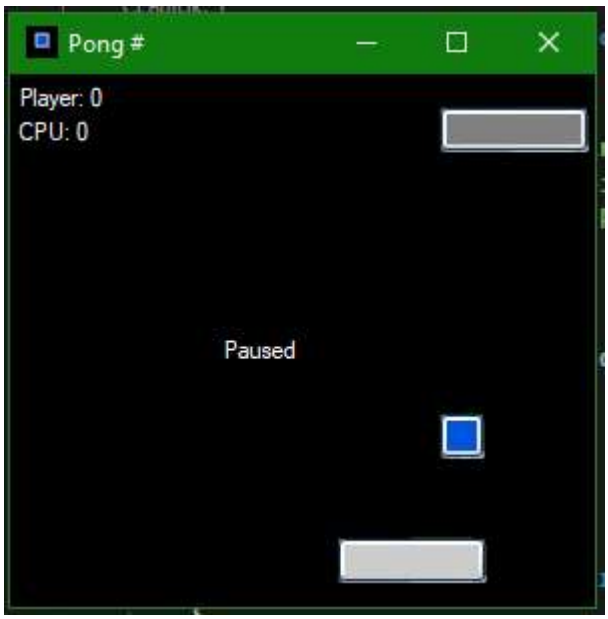

Рисунок 12. Функция паузы

4 Выводы

Были проанализированы существующие аналоги и методы разработки, а также выбрана среда разработки. Для реализации поставленной задачи отлично подошла разработка с помощью Visual Studio и языка программирования C#. Такой выбор заметно упростил разработку игры, так как в интернете имеется достаточное количество документации. Во время создания игры был полученный ценный опыт работы с этим средством разработки. В итоге была разработана игра «Pong». Тесты игра прошла хорошо, были исправлены незначительные ошибки. Созданная игра имеет потенциал к развитию, а именно: добавление новых функций; улучшение интерфейса; увеличение количества контента, добавление мультиплеера.

Библиографический список

1. Просвирнина И. Ю., Егунова А. И., Аббакумов А. А. Среда разработки

Microsoft Visual Studio на примере создания игры "морской бой" // Интеграционные процессы в науке в современных условиях. 2017. С. 123-125.
2. Базеева Н. А., Лебедев Д. С. Языки программирования для создания игр //E-Scio. 2019. №4. С. 31-39.
3. Гайнуллин Р.Ф., Захаров В.А., Аксенова Е.А. Создание 2d игры на Unity 3D 5.4 // Вестник современных исследований. 2018. №4. С. 78-82.
4. Вахрушев В.И. Реализация игры-головоломки "Расстановка мебели" с использованием unity и языка C#// Развитие современной науки: теоретические и прикладные аспекты. 2016. С.23-25.