

Создание игры «Змейка» на языке программирования C# в Visual Studio

Саяпин Никита Анатольевич

Приамурский государственный университет им.Шолом-Алейхема

Студент

Аннотация

Цель статьи — создание компьютерной версии игры «Змейка» на C# с графическим интерфейсом с использованием Windows Forms. Реализация позволяет игроку управлять змейкой, собирать еду и избегать столкновений на поле 20x15, с обработкой ввода через клавиатуру и динамической отрисовкой. Проект выполнен в Visual Studio, демонстрируя работу с .NET Framework, событиями, графикой и таймерами. Игра подходит для обучения программированию и развлечения.

Ключевые слова: Змейка, C#, Windows Forms, Visual Studio, .NET Framework, игра.

Creating a Snake game in the C# programming language in Visual Studio

Sayapin Nikita Anatolyevich

Sholom-Aleichem Priamursky State University

Student

Abstract

This article aims to develop a computer version of the classic "Snake" game using C# with a graphical user interface based on Windows Forms. The implementation enables the player to control the snake, collect food, and avoid collisions on a 20x15 grid, with keyboard input handling and dynamic rendering. The project is built in Visual Studio, showcasing the use of .NET Framework, events, graphics, and timers. The game is suitable for educational and entertainment purposes.

Keywords: Snake, C#, Windows Forms, Visual Studio, .NET Framework, game.

1. Введение

1.1 Актуальность

Создание игры «Змейка» на языке C# с использованием Windows Forms в среде Visual Studio актуально, так как проект наглядно демонстрирует применение фундаментальных концепций программирования, включая алгоритмизацию, обработку событий, управление графическим интерфейсом и реализацию игровой логики. Он позволяет изучить работу с динамическими структурами данных, таймерами и графикой, способствуя развитию навыков разработки интерактивных приложений.

1.2 Обзор исследований

В статье А.С. Маринчук описывается создание игры Arkanoid в среде Microsoft Visual Studio на языке программирования C# [1]. Д.А. Колмыков в своей работе освещает ключевые шаги создания игры, включая проектирование архитектуры, реализацию игровой логики, разработку пользовательского интерфейса и тестирование программы [2]. А.А. Эрдман в своей статье рассматривает процесс создания 2D игры без использования игровых движков по типу Unity3D, Unreal Engine 5 и подобных [3]. В работе Д.А. Шарова показан подход, позволяющий конечному пользователю самому задавать "правила игры". Разбирается структура фрагмента БД и приводится пример кода на языке C#, использующий eXpress Persistent Objects и визуальные компоненты от DevExpress [4]. В работе А.А. Эрмана рассмотрен процесс создания графического приложения для ОС Windows на примере видеоигры «Змейка» на базе платформы WPF [5].

1.3 Цель исследования

Целью исследования является разработка компьютерной версии классической игры «Змейка», позволяющей игроку управлять змейкой на поле 20x15 для сбора еды и избегания столкновений с границами и собственным телом.

2 Материалы и методы

В данном исследовании используется среда Visual Studio и язык программирования C# с фреймворком Windows Forms для создания графического интерфейса. Применяются библиотеки .NET Framework для обработки событий, управления графикой и реализации игровой логики.

2.1 Подготовка

Создание компьютерной игры «Змейка» было реализовано в среде разработки Visual Studio. На первом этапе разработки было выполнено создание нового проекта с использованием шаблона «Windows Forms App» на платформе .NET Framework версии 4.8. Проект получил название SnakeGame, после чего была задана директория для сохранения проекта. Среда разработки автоматически сформировала структуру проекта, включающую основные файлы: Program.cs, Form1.cs и Form1.Designer.cs. Основная логика приложения размещалась в файле Form1.cs (рис. 1).

Рисунок 1 – Окно создания проекта

3 Результаты

На следующем этапе проведена инициализация основных переменных, отвечающих за реализацию игровой механики. Заданы константы, определяющие размеры игрового поля (20×15 клеток) и ячеек, а также переменные для хранения состояния змейки (в виде списка координат), текущего положения еды, направления движения, флага завершения игры и таймера, управляющего игровым циклом (рис. 2).

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;

Ссылка: 2
class SnakeGame : Form
{
    private List<Point> snake = new List<Point>();
    private Point food;
    private int dx = 1, dy = 0;
    private bool gameOver = false;
    private Random random = new Random();
    private Timer gameTimer = new Timer();
    private const int cellSize = 20;
    private const int boardWidth = 20;
    private const int boardHeight = 15;
```

Рисунок 2 - Объявление переменных класса

Конструктор формы отвечал за первичную настройку графического интерфейса и инициализацию игры (рис. 3). В методе инициализации создавалось игровое поле заданного размера, конфигурировался внешний вид формы и подключались основные обработчики событий, обеспечивающие отклик интерфейса на действия пользователя и таймерные события (рис. 4).

```
Ссылка: 1
public SnakeGame()
{
    this.Text = "Змейка";
    this.ClientSize = new Size(boardWidth * cellSize, boardHeight * cellSize);
    this.FormBorderStyle = FormBorderStyle.FixedSingle;
    this.MaximizeBox = false;
    this.KeyDown += new KeyEventHandler(OnKeyDown);

    // Настройка таймера
    gameTimer.Interval = 100; // Скорость игры
    gameTimer.Tick += new EventHandler(UpdateGame);
    gameTimer.Start();

    InitializeGame();
}
```

Рисунок 3 – Конструктор класса SnakeGame

```
private void InitializeGame()
{
    snake.Clear();
    snake.Add(new Point(boardWidth / 2, boardHeight / 2)); // Начальная позиция змейки
    SpawnFood();
}
```

Рисунок 4 – Инициализация игры

Игровая логика сосредоточена в двух основных методах-обработчиках. Метод OnKeyDown отвечал за обработку клавиш клавиатуры, позволяя изменять направление движения змейки. Изменение направления происходит при условии, что новое направление не противоположно текущему, тем самым предотвращая немедленное столкновение (рис. 5).

```
private void OnKeyDown(object sender, KeyEventArgs e)
{
    // Изменение направления, только если не противоположное текущему
    if (e.KeyCode == Keys.Up && dy == 0) { dx = 0; dy = -1; }
    else if (e.KeyCode == Keys.Down && dy == 0) { dx = 0; dy = 1; }
    else if (e.KeyCode == Keys.Left && dx == 0) { dx = -1; dy = 0; }
    else if (e.KeyCode == Keys.Right && dx == 0) { dx = 1; dy = 0; }
}
```

Рисунок 5 – Обработчик события нажатия клавиш

Основной игровой цикл реализован в методе UpdateGame, который вызывался таймером через каждые 100 мс. Данный метод проверяет возможность столкновений змейки со стенками или с её телом, завершает игру при возникновении таких условий, а также обновляет положение змейки и состояние игровой еды. В случае поедания еды вызывается метод SpawnFood

для генерации новой позиции. Графическое обновление интерфейса производится путём вызова `Invalidate()` (рис. 6).

```
private void UpdateGame(object sender, EventArgs e)
{
    if (gameOver)
    {
        gameTimer.Stop();
        MessageBox.Show("Игра окончена!", "Змейка");
        return;
    }

    var head = snake[0];
    var newHead = new Point(head.X + dx, head.Y + dy);

    // Проверка на столкновение со стенами
    if (newHead.X < 0 || newHead.X >= boardWidth || newHead.Y < 0 || newHead.Y >= boardHeight)
    {
        gameOver = true;
        return;
    }

    // Проверка на столкновение с собой
    if (snake.Contains(newHead))
    {
        gameOver = true;
        return;
    }

    // Движение змейки
    snake.Insert(0, newHead);
    if (newHead == food)
        SpawnFood();
    else
        snake.RemoveAt(snake.Count - 1);

    Invalidate(); // Перерисовка формы
}
```

Рисунок 6 - Метод `UpdateGame`

Для визуализации игрового процесса реализован метод `OnPaint`, в котором с использованием класса `Graphics` происходит отрисовка змейки, еды и вспомогательной сетки. Змейка отображалась в виде зелёных прямоугольников, еда — красным эллипсом, а сетка — серыми линиями, обеспечивающими чёткое деление игрового поля (рис. 7).

```
protected override void OnPaint(PaintEventArgs e)
{
    base.OnPaint(e);
    Graphics g = e.Graphics;

    // Отрисовка змейки
    foreach (var segment in snake)
        g.FillRectangle(Brushes.Green, segment.X * cellSize, segment.Y * cellSize, cellSize, cellSize);

    // Отрисовка еды
    g.FillEllipse(Brushes.Red, food.X * cellSize, food.Y * cellSize, cellSize, cellSize);

    // Отрисовка сетки
    for (int x = 0; x <= boardWidth; x++)
        g.DrawLine(Pens.Gray, x * cellSize, 0, x * cellSize, boardHeight * cellSize);
    for (int y = 0; y <= boardHeight; y++)
        g.DrawLine(Pens.Gray, 0, y * cellSize, boardWidth * cellSize, y * cellSize);
}
```

Рисунок 7 – Метод отрисовки `onPaint`

После завершения реализации всех компонентов проекта была произведена сборка и запуск приложения с помощью комбинации клавиш `Ctrl + F5`. Интерфейс приложения состоял из поля размером 20×15 клеток,

управление происходило с помощью клавиш со стрелками, а целью игры являлось поедание еды при избежание столкновений (рис. 8).

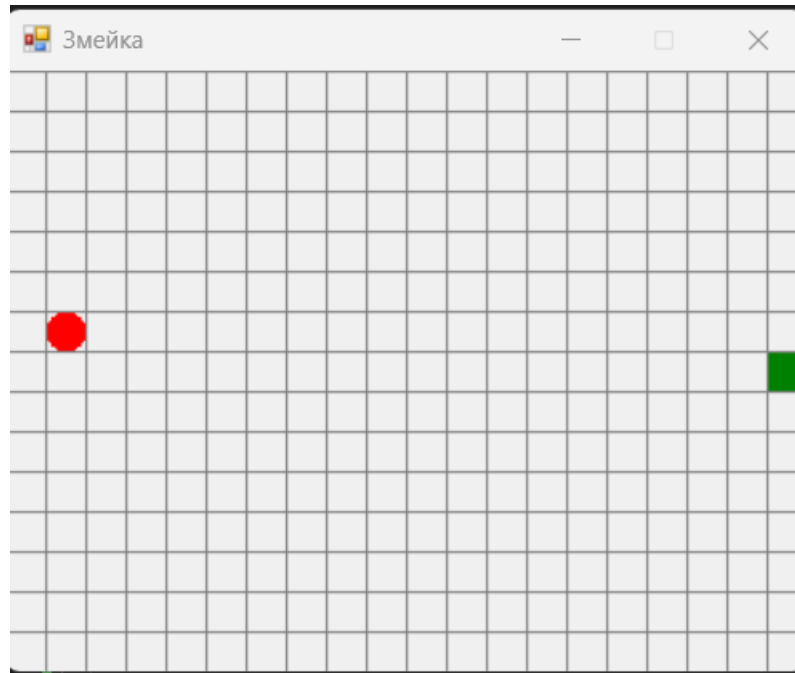


Рисунок 8 – Интерфейс игры в начальном состоянии

В процессе игры поведение змейки изменялось: при каждом поедании еды длина змейки увеличивалась, что повышало сложность управления. Интерфейс обеспечивал визуальную чёткость элементов за счёт контрастных цветов и сетки. Пример состояния интерфейса во время активной игры с удлинённой змейкой представлен на рис. 9.

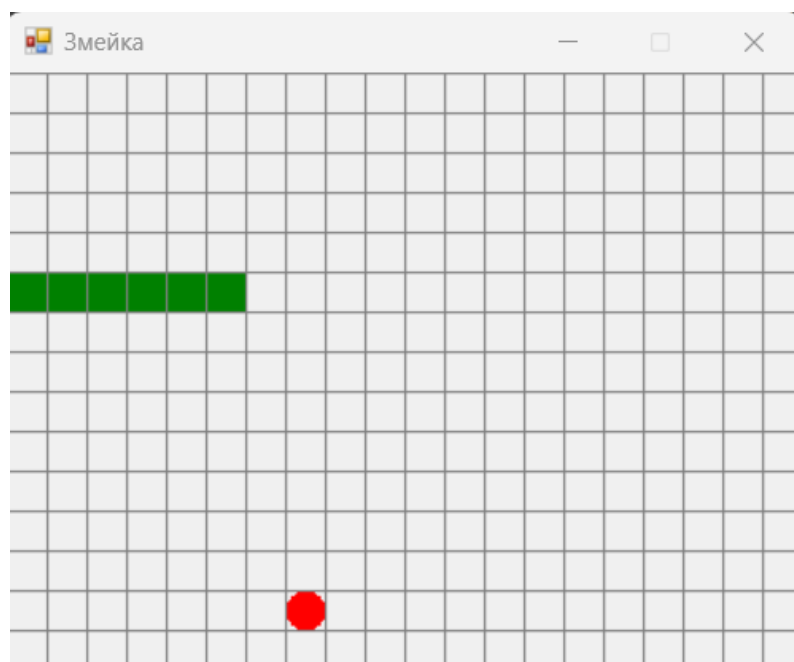


Рисунок 9 – Интерфейс игры с удлинённой змейкой

4 Выводы

Разработка настольного приложения «Змейка» с использованием платформы Windows Forms и языка программирования C# продемонстрировала практическое применение принципов объектно-ориентированного программирования, управления событиями и работы с графическим интерфейсом. Реализация игровых механизмов через таймер, динамическую отрисовку и взаимодействие с пользователем позволила создать функциональное и визуально понятное приложение. Проект подтвердил эффективность использования .NET Framework и среды Visual Studio для быстрого прототипирования интерактивных приложений и разработки игровых систем базового уровня.

Библиографический список

1. Маринчук А. С. Создание игры Arkanoid в Microsoft Visual Studio на языке программирования C //Постулат. 2021. №. 1.
2. Колмыков Д. А. Создание игры Распан на языке C //Прогрессивные технологии и экономика в машиностроении: сборник трудов XV Всероссийской научно-практической конференции для студентов и учащейся молодежи, г. Юрга, 11-13 апреля 2024 г. Томский политехнический университет, 2024. С. 244-247.
3. Эрдман А. А. Создание 2D игры на языке программирования C# с помощью библиотеки WinForms //Постулат. 2024. №. 8.
4. Шаров Д. А. Конструктор для создания связей между объектами иерархической (древовидной) сущности на базе eXpress Persistent Objects (XPO) и WinForms Controls от DevExpress //RSDN Magazine. 2009. №. 3. С. 43-47.
5. Эрдман А. А. Создание графического приложения с помощью языка программирования C# на примере видеоигры //Постулат. 2024. №. 8.