МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский Авиационный Институт»
(Национальный Исследовательский Университет)

Институт: №8 «Информационные технологии и прикладная
математика»
Кафедра: 806 «Вычислительная математика и программирование»

Реферат
по курсу «Фундаментальная информатика»
I семестр
Тема:
«Реализация игры «Тетрис» на языке программирования C#»

| | |
|---|---|
| Группа | М8О-109Б-22 |
| Студент | Горохов М.С. |
| Преподаватель | Сысоев М.А. |
| Оценка | |
| Дата | |

Москва, 2022

**Содержание**

# 1. Предметная область

## 1.1Введение

В целях изучения языка программирования и оттачивания навыков была создана небольшая игра «Тетрис». В ходе работы был выбран язык программирования C#, создан проект, подключены необходимые для создания игры модули и библиотеки, продумана логика и интерфейс игры. Все это позволило написать небольшую игру.

В связи с поставленной целью в данной работе решаются следующие задачи:

1. Ознакомление со средой разработки C#
2. Освоение языка программирования C
3. Создание графического интерфейса
4. Создание объектов
5. Реализация логики игры
6. Управление поведением объектов
7. Взаимодействие с пользователем
8. Ознакомление с интерфейсом
9. Работа с классами

## 1.2 Правила игры

Случайные фигурки тетрамино падают сверху в прямоугольный стакан шириной 10 и высотой 20 клеток. В полете игрок может поворачивать фигурку на 90° и двигать ее по горизонтали. Также можно «сбрасывать» фигурку, то есть ускорять ее падение, когда уже решено, куда фигурка должна упасть. Фигурка летит до тех пор, пока не наткнется на другую фигурку или дно стакана, если при этом заполнился горизонтальный ряд из 10 клеток, он пропадает и все, что выше него, опускается на одну клетку. Игра заканчивается, когда новая фигурка не может поместиться в стакан. Игрок получает очки за каждый заполненный ряд, поэтому его задача – заполнять ряды, не заполняя сам стакан как можно дольше, чтобы таким образом получить как можно больше очков.

## 2. Реализация

### 2.1 Среда разработки

Написание данной игры было в среде разработки VisualStudio.
VisualStudio – это многофункциональная программа, которую можно использовать для различных аспектов разработки программного обеспечения. Помимо стандартного редактора и отладчика, VisualStudio включает в себя компиляторы, средства выполнения кода, графические конструкторы и многие другие функции для упрощения процесса разработки программного обеспечения. Данная среда разработки была выбрана в основном за эти достоинства:

- Автоматизация работы. По умолчанию Visual Studio формирует код по мере ввода, автоматически вставляя необходимые отступы и применяя цветовое кодирование для выделения элементов типа комментариев.
- Возможности отладки
- Является одной из самых популярных сред разработки

### 2.2 Создание интерфейсов

В первую очередь для данного проекта необходимо создать окно, на котором будут происходить все действия игры, таблицы с кнопками управления игрой, кнопки для новой игры и выхода из приложения. Для настройки окна приложения открываем MainWindow.xaml и меняем характеристики полей Title(название окна) и Height(высота окна).
Для разметки игрового поля создается контейнер DockPanel, который содержит внутри себя StackPanel и Grid.
StacPanel – это более простой элемент компоновки. Он располагает все элементы в ряд или по горизонтали, или по вертикали в зависимости от ориентации.
DockPanel – этот контейнер прижимает свое содержимое к определенной стороне внешнего контейнера. Для этого у вложенных элементов надо установить сторону, к которой они будут прижиматься с помощью свойства DockPanel.Dock.

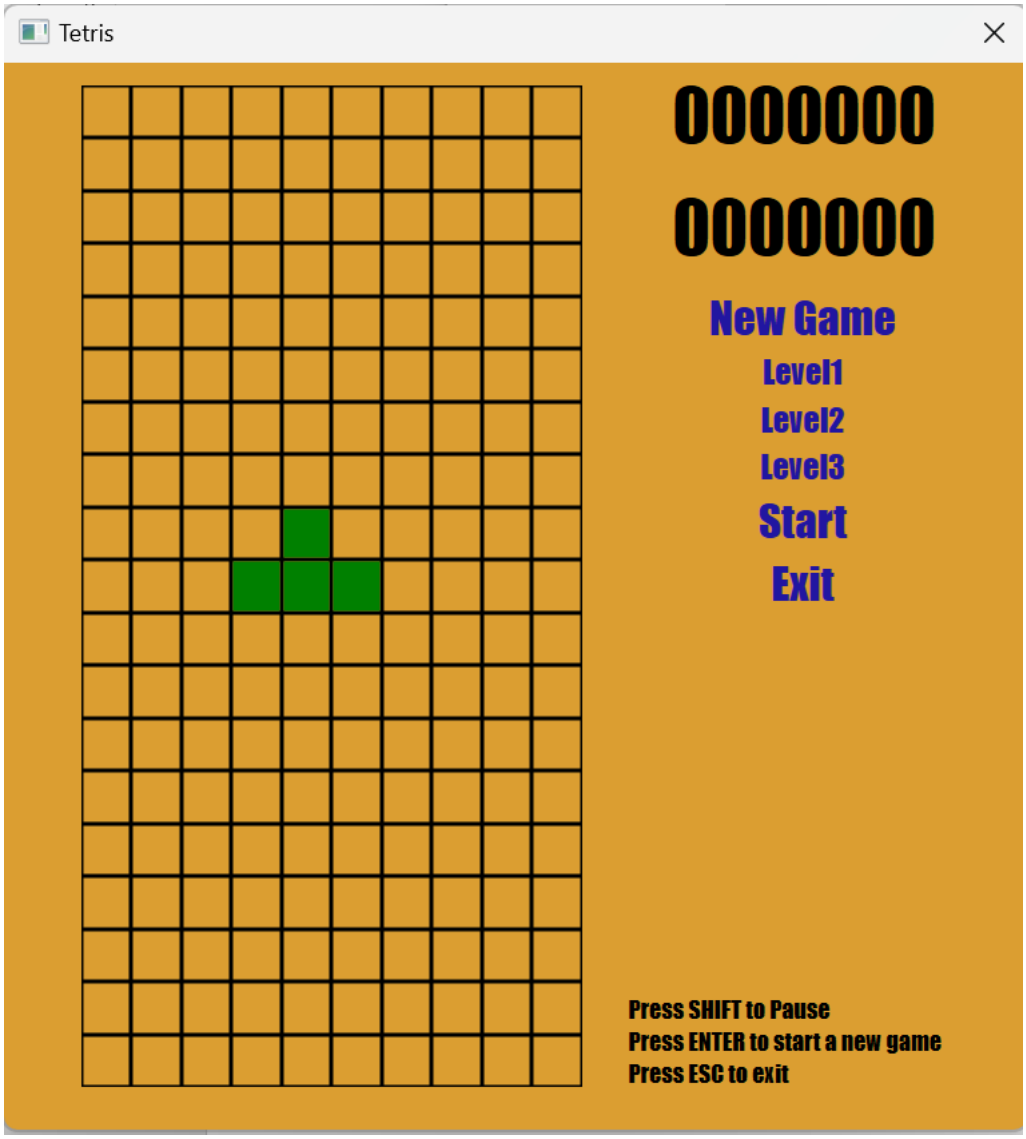
Grid – это наиболее мощный и часто используемый контейнер, напоминающий обычную таблицу. Он содержит столбцы и строки, количество которых задается разработчик. Для определения строк используем RowDefinitions, а для определения столбцов – свойство ColumnDefinitions.

В Grid задается высота, ширина и отступ.

Само игровое поле будет состоять из 10 столбцов и 20 строк.

Также мы будем использовать 1 тип кнопок и 1 тип меток.

Итоговый вид реализации интерфейса:




### 2.3 Написание логики

Класс MainWindow : Window:

Данный класс главный, и он отвечает за запуск игры, за таймер, за конец игры, за паузу и за выбор нажатия каждой из кнопок.

Класс Tetris

```
        private Point currPosition;
```

Игра тетрис состоит из фигурок, нам необходимо создать переменную, которая представляет упорядоченную пару целых чисел – координат X и Y, определяющую точку на двумерной плоскости.

```
        private Point[] currShape;
```

Это массив таких точек для дальнейшего представления самой фигурки.

```
        private Brush currColor;
```

Данная переменная необходима для заливки фигурки.

```
        private bool rotate;
```

Логическая переменная поворота фигурки.

```
        public Point CurrPosition { get { return currPosition; } }
```

Получает и возвращает положение фигурки.

```
        public Point[] CurrShape { get { return currShape; } }
```

Получает и возвращает саму фигурку.

```
        public Brush CurrColor { get { return currColor; } }
```

Получает и возвращает цве фигурки.

```
        currShape = setRandomShape();
```

Получает случайную фигурку.

Класс Board.

Для начала нужно создать игровое поле, которое состоит из строк и столбцов. Для этого служат переменные:

```
        public int Rows { get; set; }
```

Переменная, отвечающая за количество строк.

```
        public int Cols { get; set; }
```

Переменная, отвечающая за количество столбцов.

Так же для поддержания интереса к игре мы задаем счетчики очков и собранных линий.

```
public int Score { get; set; }
public int LinesFilled { get; set; }
```

### 2.4 Формирование тетрамино

Пришло время создать сами тетрамино. Формируются через public class Tetris, где задаются соответствующие переменные, описанные выше. Методы для смещения тетрамино:

```
public void moveLeft()
{
   currPosition.X -= 1;
}

public void moveRight()
{
   currPosition.X += 1;
}

public void moveDown()
{
   currPosition.Y += 1;
}

public void moveRotate()
{
   if (rotate)
   {
      for (int i = 0; i < currShape.Length; i++)
      {
         double x = currShape[i].X;
```

```
                currShape[i].X = currShape[i].Y * -1;
                currShape[i].Y = x;
            }
        }
    }
```

Метод случайного появления и сами тетрамино:

```
    private Point[] setRandomShape()
    {
        Random rand = new Random();
        switch (rand.Next() % 7)
        {
            case 0:
                rotate = true;
                currColor = Brushes.Cyan;
                return new Point[]
                {
                    new Point(0, -1),
                    new Point(-1, -1),
                    new Point(1, -1),
                    new Point(2, -1)
                };
            case 1:
                rotate = true;
                currColor = Brushes.Blue;
                return new Point[]
                {
                    new Point(1, -1),
                    new Point(-1, 0),
                    new Point(0, 0),
                    new Point(1, 0)
```

```
        };
    case 2:
        rotate = true;
        currColor = Brushes.Cornsilk;
        return new Point[]
        {
            new Point(0, 0),
            new Point(-1, 0),
            new Point(1, 0),
            new Point(-1, -1)
        };
    case 3:
        rotate = false;
        currColor = Brushes.Yellow;
        return new Point[]
        {
            new Point(0, 0),
            new Point(0, -1),
            new Point(1, 0),
            new Point(1, -1)
        };
    case 4:
        rotate = true;
        currColor = Brushes.Green;
        return new Point[]
        {
            new Point(0, 0),
            new Point(-1, 0),
            new Point(0, -1),
            new Point(1, 0)
```

```
                };
            case 5:
                rotate = true;
                currColor = Brushes.Purple;
                return new Point[]
                {
                    new Point(0, 0),
                    new Point(-1, 0),
                    new Point(1, 0),
                    new Point(1, 1)
                };
            case 6:
                rotate = true;
                currColor = Brushes.Red;
                return new Point[]
                {
                    new Point(0, -1),
                    new Point(-1, -1),
                    new Point(0, 0),
                    new Point(1, 0)
                };
            default:
                return null;
        }
    }

  }
}
```

Наша фигурка состоит из нескольких точек, которые в свое время содержат соответствующие координаты. Так же фигурке задается свой определенный цвет, чтобы их можно было различать на поле.

### 2.5 Создание поля игры

Как было указано выше, поле будет состоять из 20 столбцов и 10 строчек. Плюс ко всему поле будет содержать отдельное окно, где будет количество собранных полных линий и количество очков.

```
public int Rows { get; set; }
public int Cols { get; set; }
public int Score { get; set; }
public int LinesFilled { get; set; }
private Tetris currTetris;
private Label[,] BlockControls;

private Brush NoBrush = Brushes.Transparent; - цвет поля
private Brush SilverBrush = Brushes.Black; - цвет сетки
```

Как можно заметить поле мы создаем через public class Board.

Метод для ограничения поля:

```
public Board(Grid TetrisGrid)
{
   Rows = TetrisGrid.RowDefinitions.Count;
   Cols = TetrisGrid.ColumnDefinitions.Count;
   Score = 0;
   LinesFilled = 0;

   BlockControls = new Label[Cols, Rows];
   for (int i = 0; i < Cols; i++)
   {
      for (int j = 0; j < Rows; j++)
      {
```

```
                BlockControls[i, j] = new Label();
                BlockControls[i, j].Background = NoBrush;
                BlockControls[i, j].BorderBrush = SilverBrush;
                BlockControls[i, j].BorderThickness = new Thickness(1, 1, 1, 1);
                Grid.SetRow(BlockControls[i, j], j);
                Grid.SetColumn(BlockControls[i, j], i);
                TetrisGrid.Children.Add(BlockControls[i, j]);
            }
        }
        currTetris = new Tetris();
        currTetrisDraw();
    }
```

Данный метод ограничивает возможность тетрамино выйти за границы поля. Здесь используется цикл for и создаются ему рамки нашими столбами и строками. Так же тут задаются размеры границ нашему полю, размеры границ сетки.

### 2.6 Создание логики для кнопок

Создание логики для кнопок реализуем в классе:

```
public partial class MainWindow : Window
```

Нам понадобится функция таймера и класс Board. Поэтому объявляется:

```
    DispatcherTimer Timer;
    Board myBoard;
```

Создаем метод остановки игры и вывода очков:

```
    private void GameTick(object sender, EventArgs e)
    {
        try
        {
            Score.Content = myBoard.Score.ToString("0000000");
            Lines.Content = myBoard.LinesFilled.ToString("0000000");
            myBoard.CurrTetrisMoveDown();
```

```
            if (myBoard.checkLastRow())
            {
                Timer.Stop();
                MainGrid.Children.Clear();
                Lines.FontSize = 15;
                Lines.Content = "Your score is " + Score.Content;
                Score.Content = "GAME OVER!";
                Score.Width = 184;
                Score.Foreground = Brushes.Red;
            }
        }
        catch { }
    }
```

Прописываем логику для кнопок «New Game», «Level1», «Level2», «Level3», «Pause» :

```
    private void GameStart()
    {
        Score.Foreground = Brushes.Black;
        Lines.FontSize = 35;
        Score.Width = 150;
        MainGrid.Children.Clear();
        myBoard = new Board(MainGrid);
        Timer.Start();
    }

    private void GamePause()
    {
        if (Timer.IsEnabled)
        {
            Pause.Content = "Start";
```

```
            Timer.Stop();
        }
        else
        {
            Pause.Content = "Pause";
            Timer.Start();
        }
    }
```

Теперь назначим эти методы соответствующим кнопкам:

```
    private void NewGame_Click(object sender, RoutedEventArgs e)
    {
        Button b = (Button)sender;
        if ((string)(b.Content) == "New Game")
            Timer.Interval = new TimeSpan(0, 0, 0, 0, 600);
        else if ((string)(b.Content) == "Level1")
            Timer.Interval = new TimeSpan(0, 0, 0, 0, 600);
        else if ((string)(b.Content) == "Level2")
            Timer.Interval = new TimeSpan(0, 0, 0, 0, 400);
        else if ((string)(b.Content) == "Level3")
            Timer.Interval = new TimeSpan(0, 0, 0, 0, 300);
        GameStart();
    }

    private void Pause_Click(object sender, RoutedEventArgs e)
    {
        GamePause();
    }

    private void Exit_Click(object sender, RoutedEventArgs e)
    {
```

```
        Close();
    }
```

С увеличением уровня увеличивается скорость падения тетрамино.

Теперь все кнопки игры реализованы.

### 2.7 Логика для клавиш

Осталось реализовать логику для клавиш, чтобы управлять тетрамино в игре.

Делаем это так:

```
    private void HandleKeyDown(object sender, KeyEventArgs e)
    {
        switch (e.Key)
        {
            case Key.Left:
                if (Timer.IsEnabled) myBoard.CurrTetrisMoveLeft();
                break;
            case Key.Right:
                if (Timer.IsEnabled) myBoard.CurrTetrisMoveRight();
                break;
            case Key.Down:
                if (Timer.IsEnabled) myBoard.CurrTetrisMoveDown();
                break;
            case Key.Up:
                if (Timer.IsEnabled) myBoard.CurrTetrisMoveRotate();
                break;
            case Key.Enter:
                GameStart();
                break;
            case Key.LeftShift:
                GamePause();
                break;
            case Key.Escape:
```

```
                Close();
                break;
            default:
                break;
        }
    }
```

## 3. Заключение

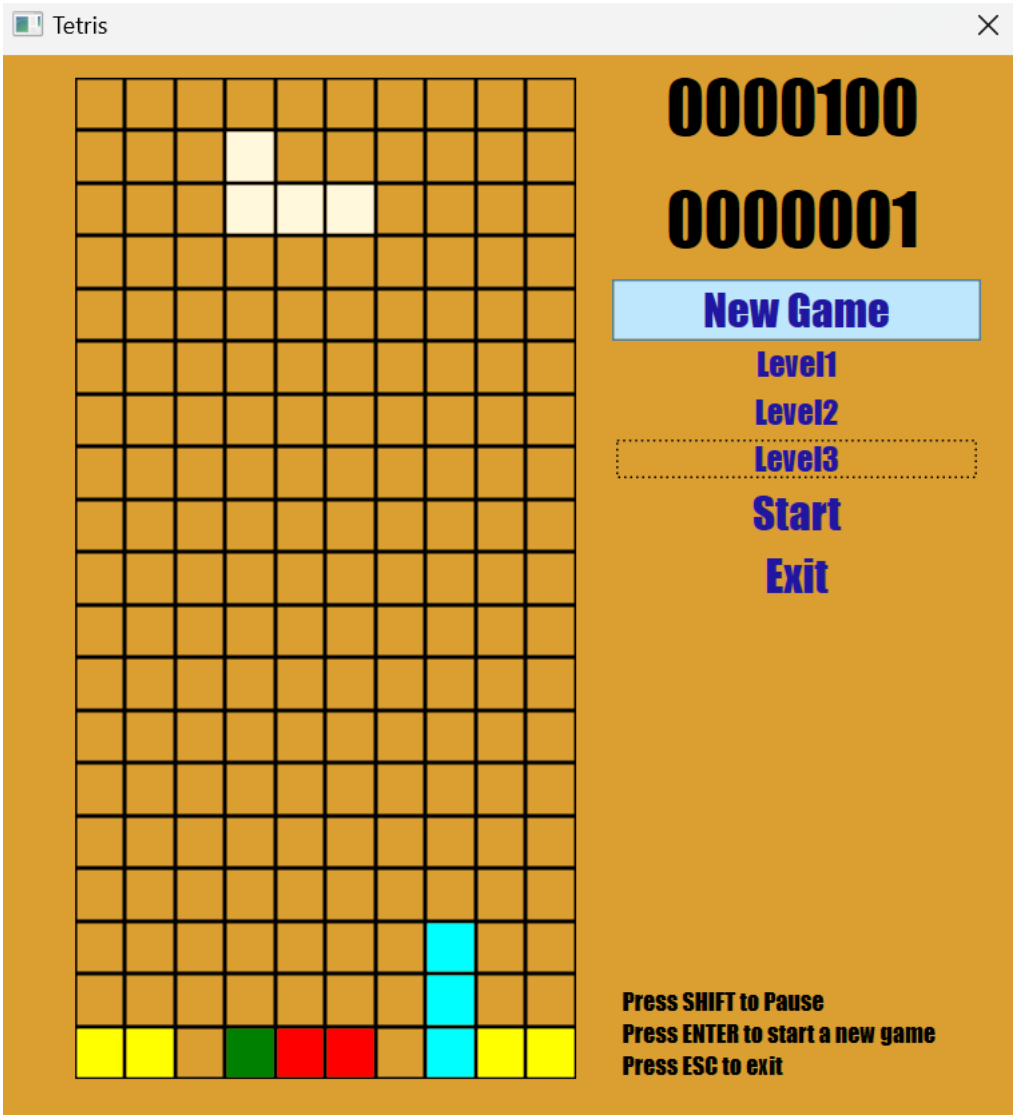
Данная работа помогла мне ближе познакомится с языком программирования C#, и это работа мне очень понравилась. Я ознакомился и научился работать с языком разметки XAML, а также платформой пользовательского интерфейса WPF. Изучил новые библиотеки и написал игру.

**Список литературы**

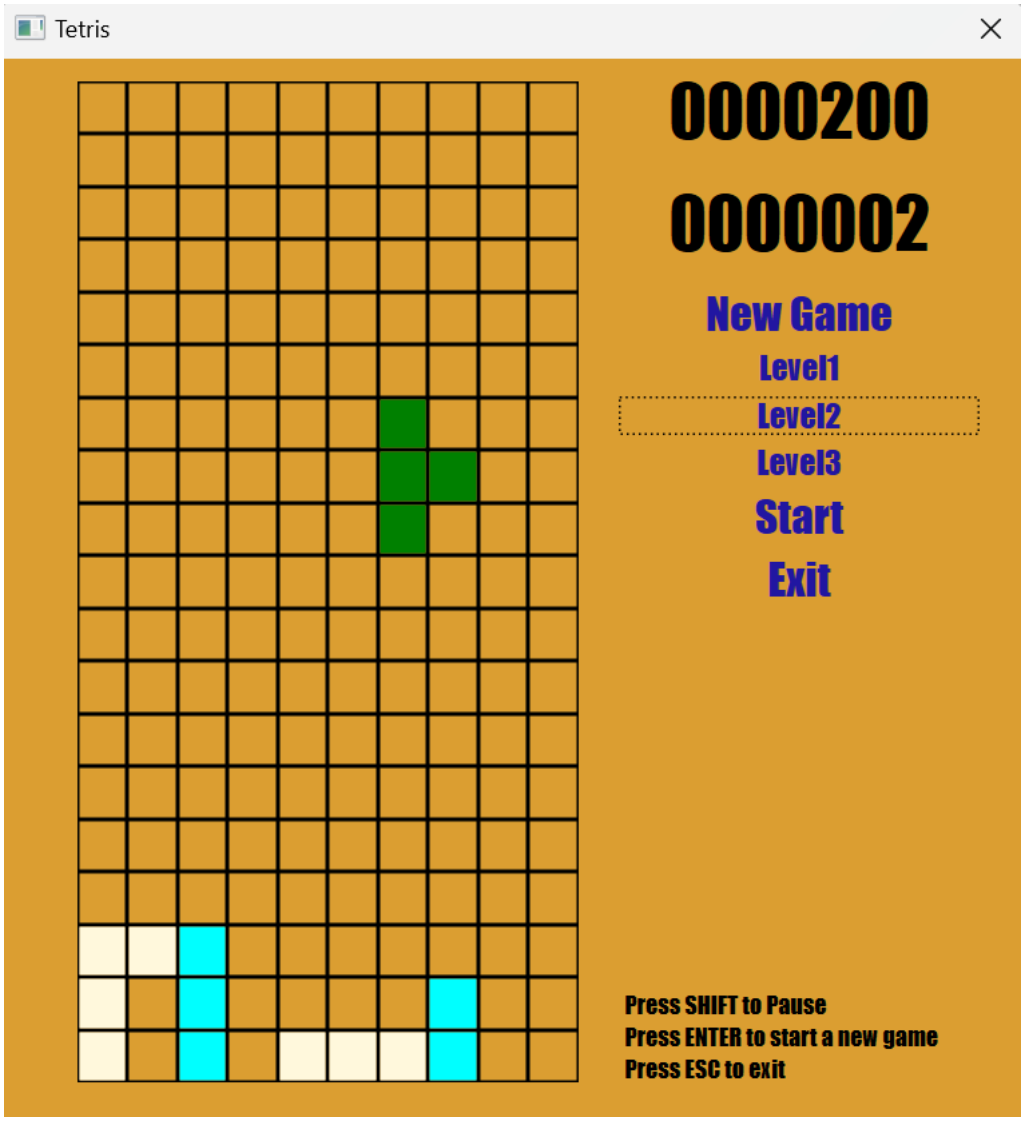
C# на примерах. П.В. Евдокимов (Книга, год издания 2016)

Справочник C# (Электронный ресурс) https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/

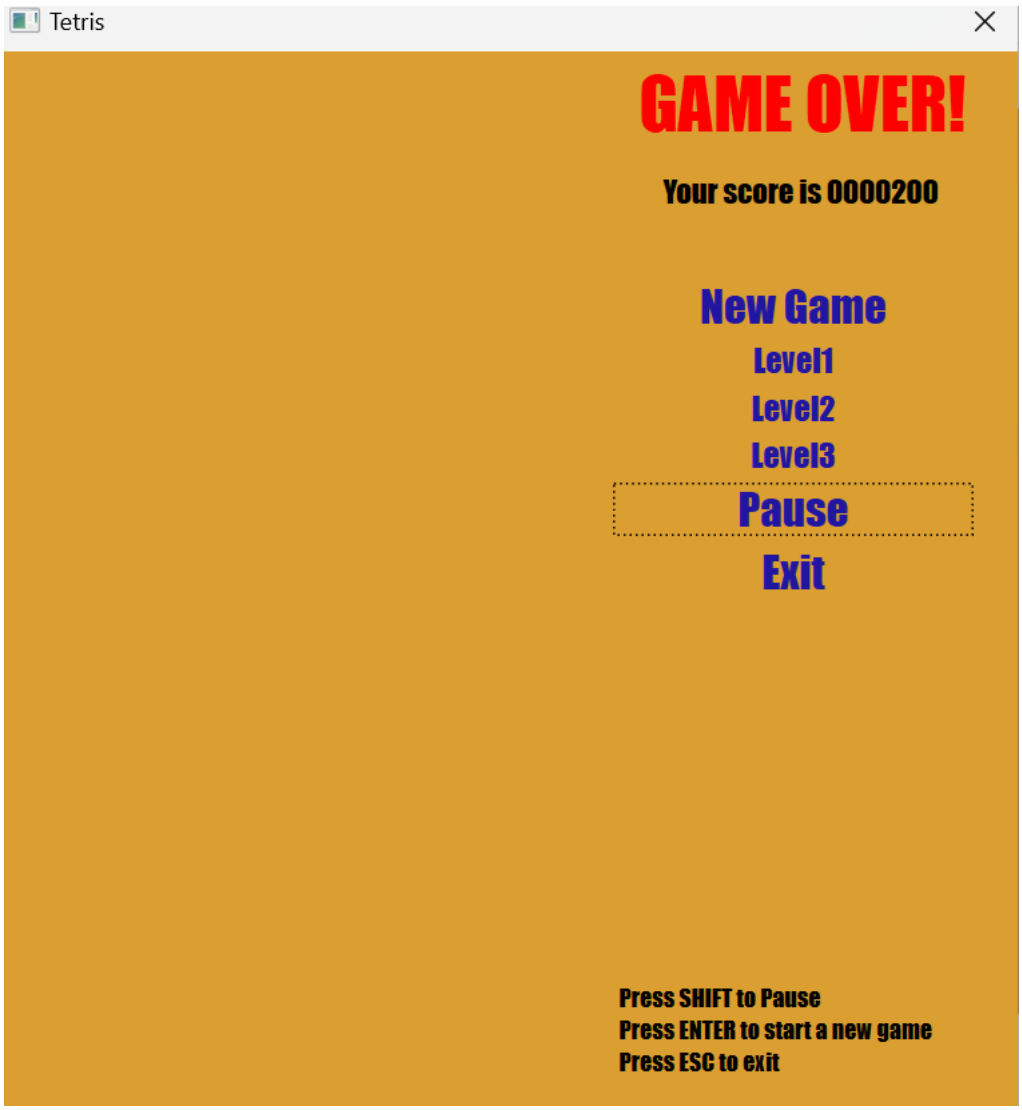
Руководство по языку программирования C# (Электронный ресурс) https://metanit.com/sharp/general.php


Tetris
0000100
0000001
New Game
Level1
Level2
Level3
Start
Exit
Press SHIFT to Pause
Press ENTER to start a new game
Press ESC to exit

Первые очки




Процесс игры




Поражение в игре