**КУРСОВА РАБОТА**

по Системи и Технологии за Мултимедия

Тема: Игра наподобяваща играта Pong

изработил **Васил Ясенов Божурски**

група: **41**

Факултетен номер: **121214045**

1. ВЪВЕДЕНИЕ

Курсовата работа цели имплементация на игра с правила подобни на класическата игра Pong. Реализацията е постигната чрез WPF приложение според изисканията и представлява игра за двама играча, като управлението на “хилките” на всеки играч става посредством клавиатурата.

1. РЕАЛИЗАЦИЯ

Некви къси думи тука.

2.1. **Ball.cs**

Класът **Ball** описва параметрите на топката. Това са позиция спрямо **x** и **y** координатата и към кой играч лети топката. Ако **isDirectionRight** е **true**,то топчето лети към десния играч, в противен случай е обратното.

class Ball

{

private double xPosition;

private double yPosition;

private bool isDirectionRight;

public double XPosition { get => xPosition; set => xPosition = value; }

public double YPosition { get => yPosition; set => yPosition = value; }

public bool IsDirectionRight { get => isDirectionRight; set => isDirectionRight = value; }

}

2.2. **ViewModel.cs**

**ViewModel** класът държи информация за двете “хилки”, с които играят играчите, победите на всеки играч, както и за топчето, с което играят. Той имплементира интерфейсът **INotifyPropertyChanged** за да позволи директно обновление във View-то при промяна на някоя стойност.

class ViewModel : INotifyPropertyChanged

{

private int leftPadPosition = 180;

private int rightPadPosition = 180;

private int leftResult = 0;

private int rightResult = 0;

private Ball ball = new Ball { XPosition = 380, YPosition = 210, IsDirectionRight = true };

public int LeftPadPosition

{

get { return leftPadPosition; }

set

{

leftPadPosition = value;

OnPropertyChanged("LeftPadPosition");

}

}

**\*Останалите property-та са изпуснати, за да се спести място в документацията.**

public void changeBallDirection()

{

IsBallDirectionRight = !IsBallDirectionRight;

}

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropertyChanged(string propertyName)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

}

2.3. **MainWindow.xaml**

**MainWindow** описва изгледа на игралното поле. Той представлява един обект от тип **Canvas**, в който са изрисувани останалите елементи на играта. Това са двете “хилки” представляващи обекти от тип **Rectangle**, топчето (**Ellipse**) и точките на всеки играч (2 обекта от тип **Label**).

<Window x:Class="PongProject.MainWindow"

**\*част от кода е изпуснат**

KeyDown="MainWindow\_OnKeyDown"

Title="Ping-Pong" Height="503" Width="824" ResizeMode="NoResize">

<Canvas Width="800" Height="475" Name="MainCanvas" Background="Black">

<Rectangle Height="80" Width="20" Fill="#EEEEEE" Canvas.Top="{Binding LeftPadPosition}"/>

<Rectangle Height="80" Width="20" Fill="#EEEEEE" Canvas.Top="{Binding RightPadPosition}" Canvas.Left="780"/>

<Line X1="400" X2="400" Y2="500" Y1="0" StrokeThickness="2" Stroke="#EEEEEE"/>

<Ellipse Width="20" Height="20" Canvas.Left="{Binding BallXPosition}" Canvas.Top="{Binding BallYPosition}" StrokeThickness="0.1" Fill="#EEEEEE"/>

<Label Canvas.Left="349" Canvas.Top="35" Content="{Binding LeftResult}" Height="auto" FontSize="40" Foreground="#EEEEEE"/>

<Label Canvas.Left="421" Canvas.Top="35" Content="{Binding RightResult}" Height="auto" FontSize="40" Foreground="#EEEEEE"/>

</Canvas>

</Window>

2.4. **MainWindow.xaml.cs**

Съдържа същинската част на играта. Обработва входните сигнали от клавиатурата и се грижи за игралната логика. Използва класа **ViewModel**, за да работи с “хилките” и топчето.

public partial class MainWindow : Window

{

private ViewModel vm = new ViewModel();

DispatcherTimer timer = new DispatcherTimer();

**\*Инициализира компонента, задава контекста да е vm и създава таймер, който на 10 милисекунди изпълнява GameTickCalculation**

public MainWindow()

private double angle = 155;

private double speed = 5;

private int padSpeed = 7;

**\*Изчислява новата позиция на топчето. Пресмята дали то е ударило дадена стена или пък е влезнало във вратата на един от играчите. Грижи се за точките и за рестартирането на играта в случай, че някой играч направи точка.**

void GameTickCalculation(object sender, EventArgs e)

**\*Връща позицията на топчето на средата на полето.**

private void GameResetBallPosition()

**\*Използва се, когато трябва да се изчисли новата посока на топчето, когато то се удари в някоя хилка.**

private void ChangeAngle()

**\*Проверява дали топчето е ударило хилка.**

private bool CheckCollision()

**\*Взима входни данни дали даден клавиш е натиснат. Играч 1 играе с W и S, а пък играч 2 със стрелките нагоре и надолу.**

private void MainWindow\_OnKeyDown(object sender, KeyboardEventArgs e)

**\*Проверява дали хилката е стигнала края на полето и не и позволява да излезе извън него.**

private int verifyBounds(int position, int change)

}