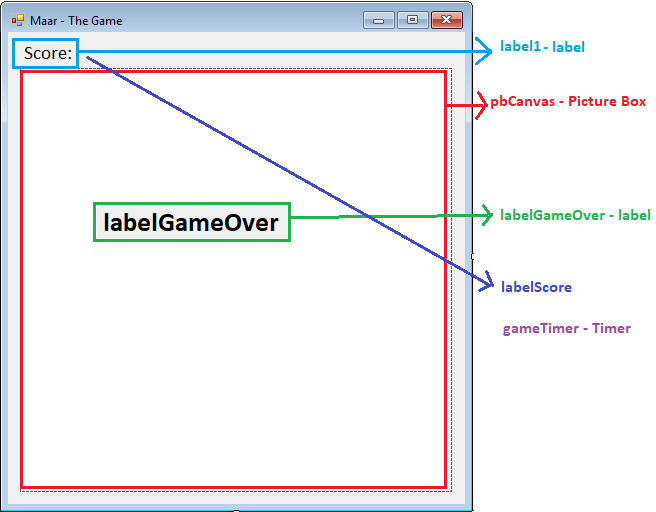
**Maar – The Game**

**Group Members:**

* Muhammad Shahzeb Khan (36518)
* Yasir Ali (39549)
* Muhammad Ismail (38698)

**Form Design:**



**Source Code:**

# **Circle Class:**

namespace Maar\_TheGame

{

class Circle

{

public int X { get; set; }

public int Y { get; set; }

public Circle()

{

X = 0;

Y = 0;

}

}

}

# **Input Class:**

using System.Collections;

using System.Windows.Forms;

namespace Maar\_TheGame

{

internal class Input

{

//Load list of available keyboard buttons

private static Hashtable keyTable = new Hashtable();

//Perform a check to see if a particular keyboard button is pressed

public static bool KeyPressed(Keys key)

{

if(keyTable[key]==null)

{

return false;

}

return (bool)keyTable[key];

}

//Detect if a keyboard button is pressed

public static void ChangeState(Keys key,bool state)

{

keyTable[key] = state;

}

}

}

# **Settings Class:**

namespace Maar\_TheGame

{

public enum Direction

{

Up,

Down,

Left,

Right

};

public class Settings

{

public static int Width { get; set; }//Width of Circle

public static int Height { get; set; }//Height of Circle

public static int Speed { get; set; }//How fast the player character moves

public static int Score { get; set; }//Total Score of the Game

public static int Points { get; set; }//Determine how many points will be added each time character eats food

public static bool GameOver { get; set; }//If TRUE than the game will end

public static Direction direction { get; set; }

public Settings()

{

Width = 16;

Height = 16;

Speed = 12;

Score = 0;

Points = 100;

GameOver = false;

direction = Direction.Down;

}

}

}

# **Form1 Class:**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;

namespace Maar\_TheGame

{

public partial class Form1 : Form

{

private List<Circle> Snake = new List<Circle>();//Character

private Circle food = new Circle();

public Form1()

{

InitializeComponent();

//set settings to default

new Settings();

//set game speed and start timer

gameTimer.Interval = 1000 / Settings.Speed;

gameTimer.Tick += UpdateScreen;

gameTimer.Start();

//Start new Game

StartGame();

}

private void StartGame()

{

labelGameOver.Visible = false;

//Set settings to default

new Settings();

//Create a new player object

Snake.Clear();

Circle head = new Circle { X = 10, Y = 5 };

//head.X = 10;

//head.Y = 5;

Snake.Add(head);

labelScore.Text = Settings.Score.ToString();

GenerateFood();

}

//place random food object/game

private void GenerateFood()

{

int maxXPos = pbCanvas.Size.Width / Settings.Width;

int maxYPos = pbCanvas.Size.Height / Settings.Height;

Random random = new Random();

food = new Circle { X = random.Next(0, maxXPos), Y = random.Next(0, maxYPos) };

//food.X = random.Next(0, maxXPos);

//food.Y = random.Next(0, maxYPos);

}

public void UpdateScreen(object sender, EventArgs e)

{

//check for Game Over

if(Settings.GameOver==true)

{

//check if Enter is pressed

if(Input.KeyPressed(Keys.Enter))

{

StartGame();

}

}

else

{

if(Input.KeyPressed(Keys.Right) && Settings.direction!=Direction.Left)

Settings.direction = Direction.Right;

else if (Input.KeyPressed(Keys.Left) && Settings.direction != Direction.Right)

Settings.direction = Direction.Left;

else if (Input.KeyPressed(Keys.Up) && Settings.direction != Direction.Down)

Settings.direction = Direction.Up;

else if (Input.KeyPressed(Keys.Down) && Settings.direction != Direction.Up)

Settings.direction = Direction.Down;

MovePlayer();

}

pbCanvas.Invalidate();//It will delete all the data on screen and on it again

}

private void pbCanvas\_Paint(object sender, PaintEventArgs e)

{

Graphics canvas = e.Graphics;

if (!Settings.GameOver)

{

//set colour of snake

Brush snakeColour;

//Draw snake

for (int i = 0; i < Snake.Count; i++)

{

if (i == 0)

snakeColour = Brushes.Black; //Draw Head

else

snakeColour = Brushes.Green; //Rest of the Body

//Draw snake

canvas.FillEllipse(snakeColour,

new Rectangle(Snake[i].X \* Settings.Width,

Snake[i].Y \* Settings.Height,

Settings.Width, Settings.Height));

//Draw food

canvas.FillEllipse(Brushes.Red,

new Rectangle(food.X \* Settings.Width,

food.Y \* Settings.Height,

Settings.Width, Settings.Height));

}

}

else

{

string gameOver = "\tGAME OVER!\nYour final Score is: " + Settings.Score + "\nPress Enter to try again";

labelGameOver.Text = gameOver;

labelGameOver.Visible = true;

}

}

private void MovePlayer()

{

for (int i = Snake.Count-1; i>=0 ; i--)

{

//Move head

if(i==0)

{

switch(Settings.direction)

{

case Direction.Right:

Snake[i].X++;

break;

case Direction.Left:

Snake[i].X--;

break;

case Direction.Up:

Snake[i].Y--;

break;

case Direction.Down:

Snake[i].Y++;

break;

}

//Get maximum X and Y Pos

int maxXPos = pbCanvas.Size.Width / Settings.Width;

int maxYPos = pbCanvas.Size.Height / Settings.Height;

//Detect collission with game borders.

if (Snake[i].X < 0 || Snake[i].Y < 0

|| Snake[i].X >= maxXPos || Snake[i].Y >= maxYPos)

{

Die();

}

//Detect collission with body

for (int j = 1; j < Snake.Count; j++)

{

if (Snake[i].X == Snake[j].X &&

Snake[i].Y == Snake[j].Y)

{

Die();

}

}

//Detect collision with food piece

if (Snake[0].X == food.X && Snake[0].Y == food.Y)

{

Eat();

}

}

else

{

Snake[i].X = Snake[i - 1].X;

Snake[i].Y = Snake[i - 1].Y;

}

}

}

private void Form1\_KeyDown(object sender, KeyEventArgs e)

{

Input.ChangeState(e.KeyCode, true);

}

private void Form1\_KeyUp(object sender, KeyEventArgs e)

{

Input.ChangeState(e.KeyCode, false);

}

private void Eat()

{

//Add circle to body

Circle circle = new Circle

{

X = Snake[Snake.Count - 1].X,

Y = Snake[Snake.Count - 1].Y

};

Snake.Add(circle);

//Update Score

Settings.Score += Settings.Points;

labelScore.Text = Settings.Score.ToString();

GenerateFood();

}

private void Die()

{

Settings.GameOver = true;

}

}

}

# **Program Class:**

using System;

using System.Windows.Forms;

namespace Maar\_TheGame

{

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

**Output:**

