МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ



Кафедра «СМАРТ технологии»

Лабораторная работа № 6:

«Создание интерактивных приложений»

По дисциплине: «Программирование и алгоритмизация на языках

высокого уровня»

Группа 211-328

Студент Пан Филипп Юльевич

Дата \_\_\_\_\_\_\_\_\_\_\_

Преподаватель Идиатуллов Тимур Тофикович

2022

Задание

L06\_6. Интерактивная симуляция спортивной игры (Ping Pong)

1. Разработать приложение для визуализации движения круглого объекта (мяча) в плоскости экрана с расчетом упругой модели столкновений с прямоугольными препятствиями (битами) и стенкамиграницами зоны движения. Зона движения (поле) должна иметь прямоугольную форму размером не менее 640х480 пикселей. Левая и правая граница поля считаются зонами вылета мяча (воротами). Размер формы приложения должен быть фиксированный. При расчете движения необходимо учитывать отскока мяча от плоских поверхностей. Трением пренебречь. Изменением скорости (модуля вектора скорости) движения шайбы также пренебречь.

2. В режиме остановки симуляции должна быть реализована возможность настройки размеров движущегося объекта и препятствий, а также ворот. Сброс мяча производится в центре поля с произвольным направлением вектора движения. Скорость мяча также настраивается в режиме остановки симуляции. Также при необходимости проводится сброс накопленных очков (игра начинается

сначала).

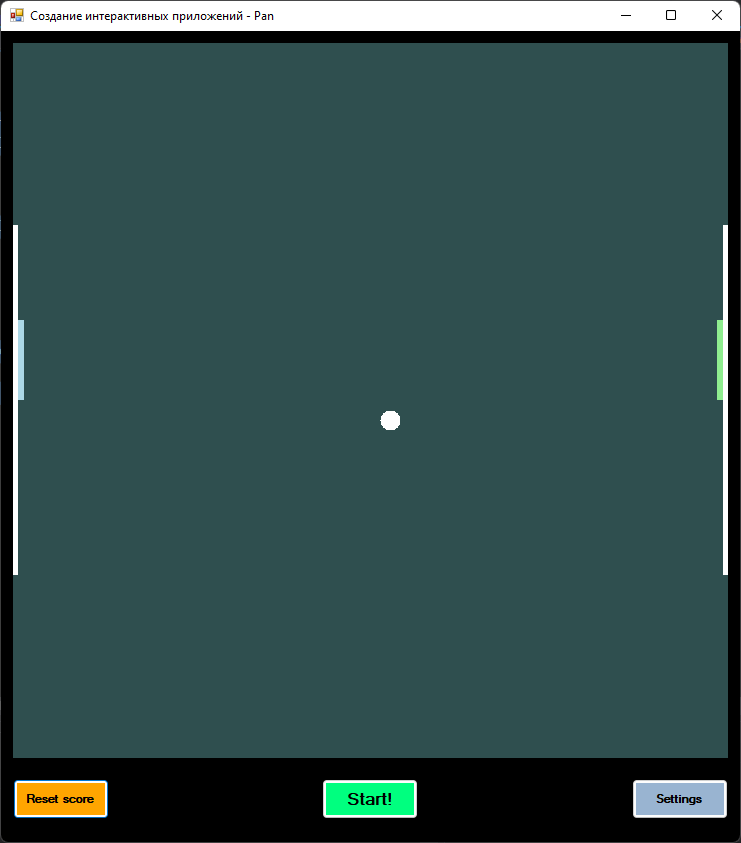
3. Установка (перемещение) препятствий должна выполняться нажатием кнопок клавиатуры (левая бита) и движением указателя мыши (правая бита). Биты должны свободно перемещаться вверх и вниз вдоль своей границы поля. При проходе мяча в створ ворот должно засчитываться игровое очко одному из игроков и проводиться сброс мяча от середины биты проигравшего игрока в сторону противоположной границы поля с произвольным вектором движения.

Ход работы

В ходе работы было разработано приложение на языке C# (Windows Forms) для визуализации динамических процессов и обеспечивающих интерактивное взаимодействие с пользователем. Реализовано моделирование процессов, определенных индивидуальным заданием, с использованием обработчиков событий таймера. Обеспечена возможность включения и отключения симуляции, а также изменения параметров процессов во время остановки симуляции. Реализована возможность интерактивного взаимодействия с симуляцией в соответствии с индивидуальным заданием.

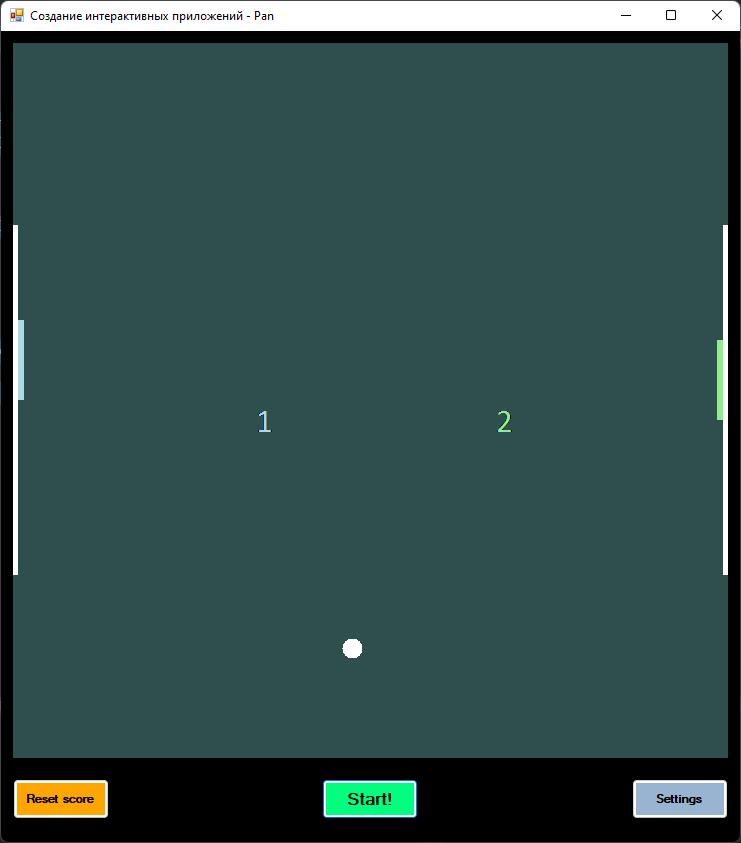
**Описание алгоритма работы программы**

При запуске программы открывается главное окно (Рисунок 1). На нем видно исходные параметры игры – размеры ворот, шарика, платформ, и их положение.



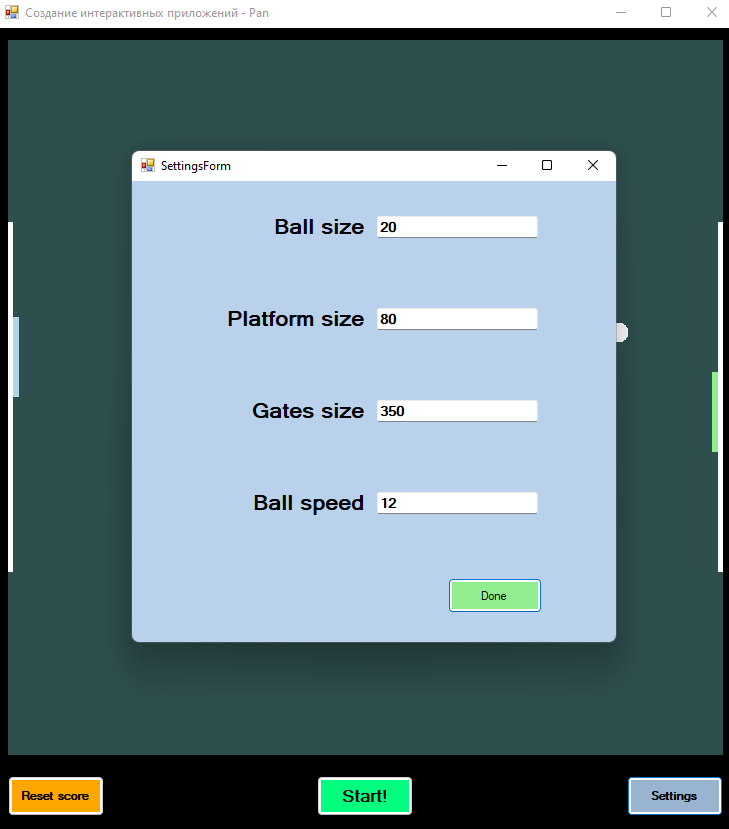
Рисунок

Если нажать на кнопку «Старт» (Рисунок 2), то запустится игра, а на экране отобразится счет. Шарик начинает двигаться со скоростью 4, установленной по умолчанию в произвольном направлении.



Рисунок

При нажатии на кнопку «Настройки» открывается окно с возможностью настроить параметры игры (Рисунок 3). При нажатии на кнопку «Сброс» счет сбрасывается, а все элементы возвращаются в исходное положение.



Рисунок

Работа системы

Мячик отскакивает от черных стенок и платформ. Чтобы забить сопернику гол, необходимо попасть в белые ворота. Левая платформа перемещается вверх и вниз при помощи клавиш W и S соответственно. Для управления правой платформой используется мышь – при перемещении курсора вниз или наверх, платформа будет следовать за ним.

Приложение А. Листинг кода

int BallSize = 20, PlatformSize = 80, GatesSize = 350, Speed = 4, // params for settings

ScoreP1 = 0, ScoreP2 = 0, PosY\_P1, PosY\_P2, PosY\_M, ballPosX, ballPosY, ballSpeedX = 1, ballSpeedY = 1, GatePosY;

bool p1\_up = false, p1\_down = false, p2\_up = false, p2\_down = false;

Graphics graphics; Bitmap image;

Random rnd = new Random();

System.Timers.Timer timer;

public MainForm()

{

InitializeComponent();

SetTimer();

GatePosY = (pbPlayGround.Height - GatesSize) / 2;

DefaultSettings();

}

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

{

switch (e.KeyCode)

{

case Keys.W:

p1\_up = false;

break;

case Keys.S:

p1\_down = false;

break;

}

}

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

{

switch (e.KeyCode)

{

case Keys.W:

p1\_up = true;

p1\_down = false;

break;

case Keys.S:

p1\_up = false;

p1\_down = true;

break;

}

}

private void pbPlayGround\_MouseEnter(object sender, EventArgs e)

{

p2\_down = false;

p2\_up = false;

}

private void pbPlayGround\_MouseMove(object sender, MouseEventArgs e)

{

PosY\_M = e.Y;

if(PosY\_M-3 > PosY\_P2 + PlatformSize / 2)

{

p2\_up = false;

p2\_down = true;

}

else if (PosY\_M + 3 < PosY\_P2 + PlatformSize / 2)

{

p2\_up = true;

p2\_down = false;

}

else

{

p2\_up = false;

p2\_down = false;

}

}

private void SetTimer()

{

timer = new System.Timers.Timer(16);

timer.Elapsed += Timer\_Elapsed;

timer.AutoReset = true;

timer.Enabled = false;

}

private void Timer\_Elapsed(object sender, ElapsedEventArgs e)

{

image = new Bitmap(pbPlayGround.Width, pbPlayGround.Height);

graphics = Graphics.FromImage(image);

DrawBall(ballPosX, ballPosY, BallSize, new SolidBrush(Color.White));

DrawPlatforms(1, PosY\_P1, 10, PlatformSize, new SolidBrush(Color.LightBlue));

DrawPlatforms(pbPlayGround.Width - 11, PosY\_P2, 10, PlatformSize, new SolidBrush(Color.LightGreen));

DrawGates(0, GatePosY, 5, GatesSize, new SolidBrush(Color.White));

DrawGates(pbPlayGround.Width - 5, GatePosY, 5, GatesSize, new SolidBrush(Color.White));

DrawScore(ScoreP1, ScoreP2);

MovePlatforms();

MoveBall();

pbPlayGround.Image = image;

return;

throw new NotImplementedException();

}

private void btnStart\_Click(object sender, EventArgs e)

{

ballSpeedX = rnd.Next(-Speed, +Speed);

if(rnd.Next(1,6) %2 == 0) ballSpeedY = Convert.ToInt32(Math.Sqrt(Math.Pow(Speed, 2) - Math.Pow(ballSpeedX, 2)));

else ballSpeedY = -Convert.ToInt32(Math.Sqrt(Math.Pow(Speed, 2) - Math.Pow(ballSpeedX, 2)));

timer.Enabled = true;

if (ballSpeedX == 0) ballSpeedX += 1;

if (ballSpeedY == 0) ballSpeedY += 1;

}

private void btnSettings\_Click(object sender, EventArgs e)

{

if(timer.Enabled) timer.Stop();

SettingsForm form = new SettingsForm(BallSize, PlatformSize, GatesSize, Speed);

form.ShowDialog();

if (form.BS > 0) BallSize = form.BS;

if (form.PS > 0) PlatformSize = form.PS;

if (form.GS > 0) GatesSize = form.GS;

if (form.S > 0) Speed = form.S;

GatePosY = (pbPlayGround.Height - GatesSize) / 2;

DefaultSettings();

}

private void DefaultSettings()

{

ballPosX = pbPlayGround.Width / 2 + BallSize;

ballPosY = pbPlayGround.Height / 2 + BallSize;

PosY\_P1 = pbPlayGround.Height / 2 - PlatformSize;

PosY\_P2 = pbPlayGround.Height / 2 - PlatformSize;

image = new Bitmap(pbPlayGround.Width, pbPlayGround.Height);

graphics = Graphics.FromImage(image);

pbPlayGround.Image = image;

DrawBall(ballPosX, ballPosY, BallSize, new SolidBrush(Color.White));

DrawPlatforms(1, PosY\_P1, 10, PlatformSize, new SolidBrush(Color.LightBlue));

DrawPlatforms(pbPlayGround.Width - 11, PosY\_P2, 10, PlatformSize, new SolidBrush(Color.LightGreen));

DrawGates(0, GatePosY, 5, GatesSize, new SolidBrush(Color.White));

DrawGates(pbPlayGround.Width-5, GatePosY, 5, GatesSize, new SolidBrush(Color.White));

ballSpeedX = rnd.Next(-3, 3);

ballSpeedY = rnd.Next(-3, 3);

}

private void DrawPlatforms(int x, int y, int w, int h, SolidBrush col)

{

graphics.FillRectangle(col, new Rectangle(x, y, w, h));

}

private void DrawGates(int x, int y, int w, int h, SolidBrush col)

{

graphics.FillRectangle(col, new Rectangle(x, y, w, h));

}

private void DrawBall(int x, int y, int size, SolidBrush col)

{

graphics.FillEllipse(col, new Rectangle(x-size/2, y-size/2, size, size));

}

private void DrawScore(int p1, int p2)

{

graphics.DrawString(p1.ToString(), new Font("Calibri", 25), new SolidBrush(Color.LightBlue), new Point(pbPlayGround.Width / 2 - 120, pbPlayGround.Height / 2));

graphics.DrawString(p2.ToString(), new Font("Calibri", 25), new SolidBrush(Color.LightGreen), new Point(pbPlayGround.Width / 2 + 120, pbPlayGround.Height / 2));

}

private void MovePlatforms()

{

if (p1\_down && PosY\_P1 < pbPlayGround.Height - PlatformSize) PosY\_P1 += 5;

if (p1\_up && PosY\_P1 > 0) PosY\_P1 -= 5;

if (p2\_down && PosY\_P2 < pbPlayGround.Height - PlatformSize) PosY\_P2 += 5;

if (p2\_up && PosY\_P2 > 0) PosY\_P2 -= 5;

if (PosY\_M - 3 < PosY\_P2 + PlatformSize / 2) p2\_down = false;

if (PosY\_M + 3 > PosY\_P2 + PlatformSize / 2) p2\_up = false;

}

private void MoveBall()

{

ballPosX += ballSpeedX;

ballPosY += ballSpeedY;

if (ballPosY + BallSize / 2 > pbPlayGround.Height-5 || ballPosY + BallSize / 2 < 0) ballSpeedY = -ballSpeedY;

if(ballPosX + ballSpeedX > pbPlayGround.Width - 5)

{

if (ballPosY > PosY\_P2 && ballPosY < PosY\_P2 + PlatformSize)

{

ballSpeedX = -Math.Abs(ballSpeedX);

}

else if (ballPosY > GatePosY && ballPosY < GatePosY + GatesSize)

{

ScoreP1++;

ballPosX = pbPlayGround.Width - 15;

ballPosY = PosY\_P2 + PlatformSize / 2;

ballSpeedX = -Math.Abs(ballSpeedX);

} else

{

ballSpeedX = -Math.Abs(ballSpeedX);

}

}

if(ballPosX + ballSpeedX < 5)

{

if (ballPosY > PosY\_P1 && ballPosY < PosY\_P1 + PlatformSize)

{

ballSpeedX = Math.Abs(ballSpeedX);

}

else if (ballPosY > GatePosY && ballPosY < GatePosY + GatesSize)

{

ScoreP2++;

ballPosX = 15;

ballPosY = PosY\_P1 + PlatformSize / 2;

ballSpeedX = Math.Abs(ballSpeedX);

}

else

{

ballSpeedX = Math.Abs(ballSpeedX);

}

}

}

private void btnReset\_Click(object sender, EventArgs e)

{

if (timer.Enabled) timer.Stop();

ScoreP1 = 0;

ScoreP2 = 0;

DefaultSettings();

}