Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Объектно-ориентированные технологии программирования и стандарты проектирования

ОТЧЕТ

по лабораторной работе

Тема работы: Простейшие графические примитивы

Выполнил

студент: гр. 251003 Дедов Н. Ю,

Проверил: Алексеев И.Г.

Минск 2024

Класс DisplayObject

public abstract class DisplayObject

{

protected int xP, yP; // точка привязки

protected int x1, y1; // Левый верхний угол клиентской части

protected int x2, y2; // Правый нижний угол клиентской части

protected Color fillColor;

protected Color borderColor;

protected int borderWidth;

protected int x1Bord, y1Bord; // Левый верхний угол рамки

protected int x2Bord, y2Bord; // Правый нижний угол рамки

protected float velocity;

protected float velocityX, velocityY;

protected int accelerationX, accelerationY;

protected float initialVelocityX, initialVelocityY;

protected float time0;

protected float angle=45;

protected float radians;

public DisplayObject(int x1, int y1, int x2, int y2, Color fillColor, Color borderColor, int borderWidth, int acccelX, int acccelY,int initialVelocity,int initialVelocityY)

{

this.x1 = x1;

this.y1 = y1;

this.x2 = x2;

this.y2 = y2;

this.fillColor = fillColor;

this.borderColor = borderColor;

this.borderWidth = borderWidth;

this.accelerationX = acccelX;

this.accelerationY = acccelY;

this.velocity = velocity;

this.radians = angle \* (float)Math.PI / 180;

this.initialVelocityX = (int) velocity \* (float)Math.Cos(radians);

this.initialVelocityY = (int)velocity \* (float)Math.Sin(radians);

velocityX = initialVelocityX;

velocityY = initialVelocityY;

CalculateBoundingRectangle();

}

public DisplayObject(int x1, int y1, int x2, int y2, Color fillColor, Color borderColor, int borderWidth)

{

this.x1 = x1;

this.y1 = y1;

this.x2 = x2;

this.y2 = y2;

this.fillColor = fillColor;

this.borderColor = borderColor;

this.borderWidth = borderWidth;

CalculateBoundingRectangle();

}

public virtual void Draw(Graphics g)

{

}

protected void SetBoundingRectangle(int x1BordN, int y1BordN, int x2BordN, int y2BordN)

{

x1Bord += x1BordN;

y1Bord += y1BordN;

x2Bord += x2BordN;

y2Bord += y2BordN;

}

protected virtual void CalculateBoundingRectangle()

{

}

public void Move(float deltaTime)

{

velocityX += accelerationX \* deltaTime;

velocityY += accelerationY \* deltaTime;

float deltaX = (int)deltaTime \* ((int)velocityX + (int)(accelerationX) \* (int)deltaTime / 2);

float deltaY = (int)deltaTime \* ((int)velocityY + (int)(accelerationY) \* (int)deltaTime / 2);

time0 += deltaTime;

ShiftCoords(deltaX, deltaY);

}

protected abstract void ShiftCoords(float deltaX, float deltaY);

}

Класс GameField

public class GameField : DisplayObject

{

private DisplayObject[] \_displayObjects = new DisplayObject[80];

private int \_p = 0;

public GameField(int x1, int y1, int x2, int y2, Color fillColor, Color borderColor, int borderWidth)

: base(x1, y1, x2, y2, fillColor, borderColor, borderWidth)

{

}

public DisplayObject[] CreateArray(int Len)

{

DisplayObject[] array = new DisplayObject[Len];

return array;

}

public void Add(DisplayObject o)

{

\_displayObjects[\_p++] = o;

}

public void Delete(DisplayObject o)

{

int indexToRemove = Array.IndexOf(\_displayObjects, o);

for (int i = indexToRemove; i < \_displayObjects.Length - 1; i++)

\_displayObjects[i] = \_displayObjects[i + 1];

\_p--;

}

public override void Draw(Graphics g)

{

base.Draw(g);

using (SolidBrush brush = new SolidBrush(fillColor))

{

g.FillRectangle(brush, x1, y1, (x2 - x1), (y2 - y1));

}

using (Pen pen = new Pen(borderColor, borderWidth))

{

g.DrawRectangle(pen, x1, y1, (x2 - x1), (y2 - y1));

}

}

protected override void CalculateBoundingRectangle()

{

int topLeftX = x1 - borderWidth;

int topLeftY = y1 - borderWidth;

int boundingWidth = (x2 - x1) + borderWidth;

int boundingHeight = (y2 - y1) + borderWidth;

SetBoundingRectangle(topLeftX, topLeftY, boundingWidth, boundingHeight);

}

protected override void ShiftCoords(float deltaX, float deltaY)

{

}

Класс Rectangle

public class Rectangle : DisplayObject

{

private PictureBox pictureBox;

private static Random random = new Random();

private int width, height;

public Rectangle(int x1, int y1, int x2, int y2, Color fillColor, Color borderColor, int borderWidth, PictureBox pictureBox, int acccelX, int acccelY, int initialVelocityX, int initialVelocityY)

: base(x1, y1, x2, y2, fillColor, borderColor, borderWidth, acccelX, acccelY, initialVelocityX, initialVelocityY)

{

this.width = x2 - x1;

this.height = y2 - y1;

this.pictureBox = pictureBox;

CalculateBoundingRectangle();

}

public override void Draw(Graphics g)

{

base.Draw(g);

using (SolidBrush brush = new SolidBrush(fillColor))

{

g.FillRectangle(brush, x1, y1, width, height);

}

using (Pen pen = new Pen(borderColor, borderWidth))

{

g.DrawRectangle(pen, x1, y1, width, height);

}

}

protected override void ShiftCoords(float deltaX,float deltaY)

{

x1 += (int)deltaX;

y1 += (int)deltaY;

x2 += (int)deltaX;

y2 += (int)deltaY;

if (!pictureBox.ClientRectangle.Contains(new System.Drawing.Rectangle(x1, y1, width, height)))

{

x1 = random.Next(width, pictureBox.ClientRectangle.Width - width);

y1 = random.Next(height, pictureBox.ClientRectangle.Height - height);

velocityX = initialVelocityX;

velocityY = initialVelocityY;

}

CalculateBoundingRectangle();

pictureBox.Invalidate();

}

protected override void CalculateBoundingRectangle()

{

int topLeftX = x1 - borderWidth;

int topLeftY = y1 - borderWidth;

int boundingWidth = (x2 - x1) + borderWidth;

int boundingHeight = (y2 - y1) + borderWidth;

SetBoundingRectangle(topLeftX, topLeftY, boundingWidth, boundingHeight);

}

}

Класс Rhombus

public class Rhombus : DisplayObject

{

private PictureBox pictureBox;

private static Random random = new Random();

public Rhombus(int x1, int y1, int x2, int y2, Color fillColor, Color borderColor, int borderWidth, PictureBox pictureBox, int acccelX, int acccelY, int initialVelocityX, int initialVelocityY)

: base(x1, y1, x2, y2, fillColor, borderColor, borderWidth, acccelX, acccelY, initialVelocityX,initialVelocityY)

{

this.x1 = x1;

this.y1 = y1;

this.x2 = x2;

this.y2 = y2;

this.pictureBox = pictureBox;

CalculateBoundingRectangle();

}

public override void Draw(Graphics g)

{

base.Draw(g);

int width = x2 - x1;

int height = y2 - y1;

int midX = x1 + width / 2;

int midY = y1 + height / 2;

Point top = new Point(midX, y1);

Point right = new Point(x2, midY);

Point bottom = new Point(midX, y2);

Point left = new Point(x1, midY);

using (SolidBrush brush = new SolidBrush(fillColor))

using (Pen pen = new Pen(borderColor, borderWidth))

{

g.FillPolygon(brush, new Point[] { top, right, bottom, left });

g.DrawPolygon(pen, new Point[] { top, right, bottom, left });

}

}

protected override void ShiftCoords(float deltaX,float deltaY)

{

x1 += (int)deltaX;

y1 += (int)deltaY;

x2 += (int)deltaX;

y2 += (int)deltaY;

if (!pictureBox.ClientRectangle.Contains(new System.Drawing.Rectangle(x1, y1, x2 - x1, y2 - y1)))

{

int width = x2 - x1;

int height = y2 - y1;

x1 = random.Next(0, pictureBox.ClientRectangle.Width - width);

y1 = random.Next(0, pictureBox.ClientRectangle.Height - height);

x2 = x1 + width;

y2 = y1 + height;

velocityX = initialVelocityX;

velocityY = initialVelocityY;

}

CalculateBoundingRectangle();

pictureBox.Invalidate();

}

protected override void CalculateBoundingRectangle()

{

int topLeftX = x1 - borderWidth;

int topLeftY = y1 - borderWidth;

int boundingWidth = (x2 - x1) + borderWidth;

int boundingHeight = (y2 - y1) + borderWidth;

SetBoundingRectangle(topLeftX, topLeftY, boundingWidth, boundingHeight);

}

}

Класс Square

public class Square : DisplayObject

{

private int side;

private PictureBox pictureBox;

private static Random random = new Random();

public Square(int x1, int y1, int x2, int y2, Color fillColor, Color borderColor, int borderWidth, PictureBox pictureBox, int acccelX, int acccelY, int initialVelocityX, int initialVelocityY)

: base(x1, y1, x2, y2, fillColor, borderColor, borderWidth, acccelX, acccelY, initialVelocityX, initialVelocityY)

{

this.pictureBox = pictureBox;

side = (x2 - x1) < (y2 - y1) ? (x2 - x1) : (y2 - y1);

CalculateBoundingRectangle();

}

public override void Draw(Graphics g)

{

base.Draw(g);

using (SolidBrush brush = new SolidBrush(fillColor))

{

g.FillRectangle(brush, x1, y1, side, side);

}

using (Pen pen = new Pen(borderColor, borderWidth))

{

g.DrawRectangle(pen, x1, y1, side, side);

}

}

protected override void ShiftCoords(float deltaX,float deltaY)

{

x1 += (int)deltaX;

y1 += (int)deltaY;

x2 += (int)deltaX;

y2 += (int)deltaY;

if (!pictureBox.ClientRectangle.Contains(new System.Drawing.Rectangle(x1, y1, side, side)))

{

x1 = random.Next(0, pictureBox.ClientRectangle.Width - side);

y1 = random.Next(0, pictureBox.ClientRectangle.Height - side);

x2 = x1 + side;

y2 = x1 + side;

velocityX = initialVelocityX;

velocityY = initialVelocityY;

}

CalculateBoundingRectangle();

pictureBox.Invalidate();

}

protected override void CalculateBoundingRectangle()

{

int topLeftX = x1 - borderWidth;

int topLeftY = y1 - borderWidth;

int boundingWidth = side + borderWidth;

int boundingHeight = side + borderWidth;

SetBoundingRectangle(topLeftX, topLeftY, boundingWidth, boundingHeight);

}

Класс DiagonalLine

public class DiagonalLine : DisplayObject

{

private PictureBox pictureBox;

private static Random random = new Random();

public DiagonalLine(int x1, int y1, int x2, int y2, Color fillColor, Color borderColor, int borderWidth, PictureBox pictureBox, int acccelX, int acccelY,int initialVelocityX,int initialVelocityY)

: base(x1, y1, x2, y2, fillColor, borderColor, borderWidth, acccelX, acccelY, initialVelocityX, initialVelocityY)

{

this.x2 = x2;

this.y2 = y2;

this.pictureBox = pictureBox;

}

public override void Draw(Graphics g)

{

base.Draw(g);

using (Pen pen = new Pen(fillColor, borderWidth))

{

g.DrawLine(pen, x1, y1, x2, y2);

}

}

protected override void ShiftCoords(float deltaX,float deltaY)

{

x1 += (int)deltaX;

y1 += (int)deltaY;

x2 += (int)deltaX;

y2 += (int)deltaY;

if (!pictureBox.ClientRectangle.Contains(new System.Drawing.Rectangle(Math.Min(x1, x2), Math.Min(y1, y2), (x2 - x1), y2 - y1)))

{

int width = x2 - x1;

int height = y2 - y1;

x1 = random.Next(pictureBox.ClientRectangle.Width);

y1 = random.Next(pictureBox.ClientRectangle.Height);

x2 = x1 + width;

y2 = y1 + height;

velocityX = initialVelocityX;

velocityY = initialVelocityY;

}

CalculateBoundingRectangle();

pictureBox.Invalidate();

}

protected override void CalculateBoundingRectangle()

{

int topLeftX = Math.Min(x1, x2) - borderWidth;

int topLeftY = Math.Min(y1, y2) - borderWidth;

int boundingWidth = Math.Abs(x2 - x1) + borderWidth;

int boundingHeight = Math.Abs(y2 - y1) + borderWidth;

SetBoundingRectangle(topLeftX, topLeftY, boundingWidth, boundingHeight);

}

}

Класс Triangle

public class Triangle : DisplayObject

{

private int x3, y3;

private PictureBox pictureBox;

private static Random random = new Random();

public Triangle(int x1, int y1, int x2, int y2, Color fillColor, Color borderColor, int borderWidth, PictureBox pictureBox, int acccelX, int acccelY, int initialVelocityX, int initialVelocityY)

: base(x1, y1, x2, y2, fillColor, borderColor, borderWidth, acccelX, acccelY, initialVelocityX, initialVelocityY)

{

this.x1 = x1;

this.y1 = y1;

this.x2 = x2;

this.y2 = y2;

this.x3 = x1 + (x2 - x1) / 2;

this.y3 = y1;

this.pictureBox = pictureBox;

CalculateBoundingRectangle();

}

public override void Draw(Graphics g)

{

base.Draw(g);

using (SolidBrush brush = new SolidBrush(fillColor))

{

g.FillPolygon(brush, new[] { new Point(x1, y2), new Point(x3, y1), new Point(x2, y2) });

}

using (Pen pen = new Pen(borderColor, borderWidth))

{

g.DrawPolygon(pen, new[] { new Point(x1, y2), new Point(x3, y1), new Point(x2, y2) });

}

}

protected override void ShiftCoords(float deltaX,float deltaY)

{

x1 += (int)deltaX;

y1 += (int)deltaY;

x2 += (int)deltaX;

y2 += (int)deltaY;

x3 += (int)deltaX;

y3 += (int)deltaY;

if (!pictureBox.ClientRectangle.Contains(new System.Drawing.Rectangle(Math.Min(x1, Math.Min(x2, x3)), Math.Min(y1, Math.Min(y2, y3)), x2-x1, y2-y1)))

{

int width = x2 - x1;

int height = y2 - y1;

x1 = random.Next(pictureBox.ClientRectangle.Width);

y1 = random.Next(pictureBox.ClientRectangle.Height);

x2 = x1 + width;

y2 = y1 + height;

x3 = x1 + width / 2;

y3 = y1;

velocityX = initialVelocityX;

velocityY = initialVelocityY;

}

else {

}

CalculateBoundingRectangle();

}

protected override void CalculateBoundingRectangle()

{

int lowestX = Math.Min(x1, Math.Min(x2, x3)) - borderWidth;

int lowestY = Math.Min(y1, Math.Min(y2, y3)) - borderWidth;

int highestX = Math.Max(x1, Math.Max(x2, x3));

int highestY = Math.Max(y1, Math.Max(y2, y3));

int width = highestX - lowestX + borderWidth;

int height = highestY - lowestY + borderWidth;

SetBoundingRectangle(lowestX, lowestY, width, height);

}

}

Класс Oval

public class Oval : DisplayObject

{

private PictureBox pictureBox;

private static Random random = new Random();

private int width, height;

public Oval(int x1, int y1, int x2, int y2, Color fillColor, Color borderColor, int borderWidth, PictureBox pictureBox, int acccelX, int acccelY,int initialVelocityX, int initialVelocityY)

: base(x1, y1, x2, y2, fillColor, borderColor, borderWidth, acccelX, acccelY,initialVelocityX, initialVelocityY)

{

this.width = x2 - x1;

this.height = y2 - y1;

this.pictureBox = pictureBox;

CalculateBoundingRectangle();

}

public override void Draw(Graphics g)

{

base.Draw(g);

using (SolidBrush brush = new SolidBrush(fillColor))

{

g.FillEllipse(brush, x1, y1, width, height);

}

using (Pen pen = new Pen(borderColor, borderWidth))

{

g.DrawEllipse(pen, x1, y1, width, height);

}

}

protected override void ShiftCoords(float deltaX,float deltaY)

{

x1 += (int)deltaX;

y1 += (int)deltaY;

x2 += (int)deltaX;

y2 += (int)deltaY;

if (!pictureBox.ClientRectangle.Contains(new System.Drawing.Rectangle(x1, y1, width, height)))

{

x1 = random.Next(width, pictureBox.ClientRectangle.Width - width);

y1 = random.Next(height, pictureBox.ClientRectangle.Height - height);

velocityX = initialVelocityX;

velocityY = initialVelocityY;

}

CalculateBoundingRectangle();

pictureBox.Invalidate();

}

protected override void CalculateBoundingRectangle()

{

int topLeftX = x1 - borderWidth;

int topLeftY = y1 - borderWidth;

int boundingWidth = (x2 - x1) + borderWidth;

int boundingHeight = (y2 - y1) + borderWidth;

SetBoundingRectangle(topLeftX, topLeftY, boundingWidth, boundingHeight);

}

}

Класс Circle

public class Circle : DisplayObject

{

private int radius;

private PictureBox pictureBox;

private static Random random = new Random();

public Circle(int x1, int y1, int x2, int y2, Color fillColor, Color borderColor, int borderWidth, PictureBox pictureBox, int acccelX, int acccelY,int initialVelocityX,int initialVelocityY) : base(x1, y1, x2, y2, fillColor, borderColor, borderWidth, acccelX, acccelY,initialVelocityX,initialVelocityY)

{

this.radius = Math.Min(x2 - x1, y2 - y1) / 2;

this.pictureBox = pictureBox;

}

public override void Draw(Graphics g)

{

base.Draw(g);

int topLeftX = x1 - radius;

int topLeftY = y1 - radius;

using (SolidBrush brush = new SolidBrush(fillColor))

{

g.FillEllipse(brush, topLeftX, topLeftY, radius \* 2, radius \* 2);

}

using (Pen pen = new Pen(borderColor, borderWidth))

{

g.DrawEllipse(pen, topLeftX, topLeftY, radius \* 2, radius \* 2);

}

}

protected override void ShiftCoords(float deltaX,float deltaY)

{

x1 += (int)deltaX;

y1 += (int)deltaY;

x2 += (int)deltaX;

y2 += (int)deltaY;

if (!pictureBox.ClientRectangle.Contains(new System.Drawing.Rectangle(x1 - radius, y1 - radius, radius \* 2, radius \* 2)))

{

x1 = random.Next(radius, pictureBox.ClientRectangle.Width - radius);

y1 = random.Next(radius, pictureBox.ClientRectangle.Height - radius);

x2 = x1 + radius\*2;

y2 = x1 + radius\*2;

velocityX = initialVelocityX;

velocityY = initialVelocityY;

}

CalculateBoundingRectangle();

pictureBox.Invalidate();

}

protected override void CalculateBoundingRectangle()

{

int diameter = Math.Min(x2 - x1, y2 - y1) + borderWidth;

int topLeftX = x1 + (x2 - x1 - diameter) / 2 - borderWidth;

int topLeftY = y1 + (y2 - y1 - diameter) / 2 - borderWidth;

SetBoundingRectangle(topLeftX, topLeftY, topLeftX+diameter, topLeftY+diameter);

}

}

Класс InvertedTriangle

public class InvertedTriangle : DisplayObject

{

private int x3, y3;

private PictureBox pictureBox;

private static Random random = new Random();

public InvertedTriangle(int x1, int y1, int x2, int y2, Color fillColor, Color borderColor, int borderWidth, PictureBox pictureBox, int acccelX, int acccelY, int initialVelocityX, int initialVelocityY)

: base(x1, y1, x2, y2, fillColor, borderColor, borderWidth, acccelX, acccelY,initialVelocityX,initialVelocityY)

{

this.x1 = x1;

this.y1 = y1;

this.x2 = x2;

this.y2 = y2;

this.x3 = x1 + (x2 - x1) / 2;

this.y3 = y1 + (y2 - y1);

CalculateBoundingRectangle();

this.pictureBox = pictureBox;

}

public override void Draw(Graphics g)

{

base.Draw(g);

using (SolidBrush brush = new SolidBrush(fillColor))

{

g.FillPolygon(brush, new[] { new Point(x1, y1), new Point(x1 + (x2 - x1), y1), new Point(x3, y3) });

}

using (Pen pen = new Pen(borderColor, borderWidth))

{

g.DrawPolygon(pen, new[] { new Point(x1, y1), new Point(x1 + (x2 - x1), y1), new Point(x3, y3) });

}

}

protected override void ShiftCoords(float deltaX,float deltaY)

{

x1 += (int)deltaX;

y1 += (int)deltaY;

x2 += (int)deltaX;

y2 += (int)deltaY;

x3 += (int)deltaX;

y3 += (int)deltaY;

if (!pictureBox.ClientRectangle.Contains(new System.Drawing.Rectangle(x1, y1, (x2 - x1), (y3 - y1))))

{

int width = x2 - x1;

int height = y3 - y1;

x1 = random.Next(pictureBox.ClientRectangle.Width - width);

y1 = random.Next(pictureBox.ClientRectangle.Height - height);

x2 = x1 + width;

y2 = y1;

x3 = x1 + width / 2;

y3 = y1 + height;

velocityX = initialVelocityX;

velocityY = initialVelocityY;

}

CalculateBoundingRectangle();

pictureBox.Invalidate();

}

protected override void CalculateBoundingRectangle()

{

int topLeftX = Math.Min(x1, x1 + (x2 - x1) / 2) - borderWidth;

int topLeftY = Math.Min(y1, y1 + (y2 - y1)) - borderWidth;

int boundingWidth = Math.Abs((x2 - x1)) + borderWidth;

int boundingHeight = Math.Abs((y2 - y1)) + borderWidth;

SetBoundingRectangle(topLeftX, topLeftY, boundingWidth, boundingHeight);

}

}

Класс Program

internal static class Program

{

/// Главная точка входа для приложения.

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}