Deakin University

Object Oriented Development

OnTrack Submission

Different Robots

Submitted By:
Akashdeep Akashdeep
s223040483
2023/05/02 01:57

Tutor:
Son Nguyen

Outcome	**Weight**
Build Programs	♦♦♦♦♦

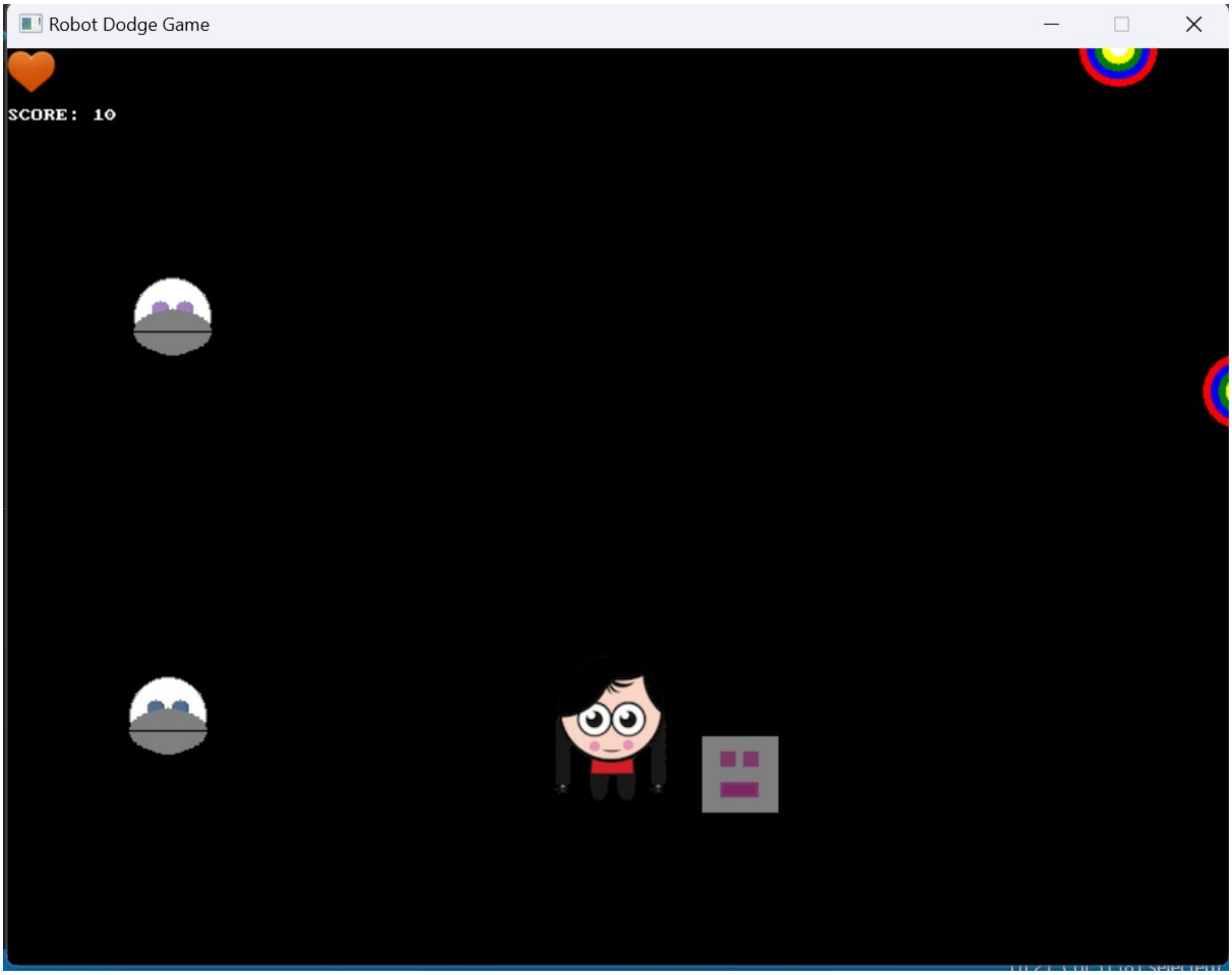
Learned how to create different types of robots and use them in the game

May 2, 2023

Produced by Doubtfire

```
using System;
using SplashKitSDK;
using System.IO;
using System.Collections.Generic;

public class RobotDodge
{
    private Player _Player;
    //private Robot _TestRobot;
    private Window _GameWindow;
    private List<Robot> _Robots = new List<Robot>();
    private List<Robot> _removedRobots = new List<Robot>();

    private List<Bullet> _Bullets = new List<Bullet>();

    private List<Bullet> _removedBullets = new List<Bullet>();

    private Bitmap HeartBitmap = new Bitmap("Heart", "heart.png");
    public SplashKitSDK.Timer myTimer;




    public bool Quit
    {
        get
        {
            return _Player.Quit;
        }
    }

    public RobotDodge(Window window)
    {
        _GameWindow = window;
        _Player = new Player(window);
        //_TestRobot = RandomRobot();
        SplashKit.LoadMusic("background", "background.mp3");
        SplashKit.LoadSoundEffect("gameover", "gameover.wav");
        SplashKit.LoadSoundEffect("background", "background.wav");
        SplashKit.PlayMusic("background");
        myTimer = new SplashKitSDK.Timer("My Timer");
        myTimer.Start();

    }

    public void HandleInput()
    {
        _Player.HandleInput();
        _Player.StayOnWindow(_GameWindow);
    }

    public void Draw()
    {
        _GameWindow.Clear(Color.Black);
        foreach (Robot robot in _Robots)
        {
            robot.Draw();
        }

        _Player.Draw();
        foreach (Bullet bullet in _Bullets)
        {
            bullet.Draw();
        }
        DisplayHUD();
        if (_Player.Lives <= 0)
        {
            _GameWindow.Clear(Color.Black);
            Bitmap _GameOver = new Bitmap("Game Over", "gameover.png");
            SplashKit.DrawBitmap(_GameOver, 200, 100);
            SplashKit.StopMusic();
            SplashKit.PlaySoundEffect("gameover");

        }
        _GameWindow.Refresh(60);
    }


    public Robot RandomRobot()
    {
        // _TestRobot = new Robot(_GameWindow, _Player);
        //return _TestRobot;
        Robot _RandomRobotOne = new Robot.Boxy(_GameWindow, _Player);
        Robot _RandomRobotTwo = new Robot.Roundy(_GameWindow, _Player);
        Robot _RandomRobotThree = new Robot.Custom(_GameWindow, _Player);

        double randomNumber = SplashKit.Rnd(900);
        if (randomNumber < 300)
        {
            return _RandomRobotOne;
        }
        else if (randomNumber > 300 & randomNumber < 600)
        {
            return _RandomRobotTwo;
        }
        else
        {
            return _RandomRobotThree;
        }
    }
    public void Update()
    {
        CheckCollisions();
        _Player.Score = Convert.ToInt32(myTimer.Ticks / 1000);
        foreach (Robot robot in _Robots)
        {
            robot.Update();
        }
        //add random number of robots into the list
        double randomNumber = SplashKit.Rnd(1000);
        if (randomNumber < 25)
        {
            _Robots.Add(RandomRobot());
        }

        if (SplashKit.MouseClicked(MouseButton.LeftButton))
        {
            _Bullets.Add(AddBullet());

            SplashKit.PlaySoundEffect("bullet");
        }
        foreach (Bullet bullet in _Bullets)
        {
            bullet.Update();
        }
    }

    public Bullet AddBullet()
    {
        Bullet _RandomBullet = new Bullet(_GameWindow, _Player);
        return _RandomBullet;
    }

    private void CheckCollisions()
    {

        foreach (Robot robot in _Robots)
        {

            //check the player and robot collision to remove the robot from main list
            if (_Player.CollidedWith(robot) || robot.IsOffscreen(_GameWindow))
            {
                _removedRobots.Add(robot);
            }
            if (_Player.CollidedWith(robot) & _Player.Lives > 0)
            {
                _Player.Lives = _Player.Lives - 1;
            }
            //check the bullet and robot collision
            foreach (Bullet bullet in _Bullets)
            {
                if (bullet.BulletCollidedWith(robot))
                {
                    _removedBullets.Add(bullet);
                    _removedRobots.Add(robot);
                }
                if (bullet.IsOffscreen(_GameWindow))
                {
                    _removedBullets.Add(bullet);
                }
            }
        }
        foreach (Robot robot in _removedRobots)
        {
            _Robots.Remove(robot);
        }
        foreach (Bullet bullet in _removedBullets)
        {
            _Bullets.Remove(bullet);
        }

    }

    public void DisplayHUD()
    {
        DrawHearts(_Player.Lives);
        SplashKit.DrawText("SCORE: " + _Player.Score, Color.White, 0, 40);
    }

    public void DrawHearts(int numberOfHearts)
    {
        int heartX = 0;
        for (int i = 0; i < numberOfHearts; i++)
        {
            if (heartX < 300)
            {
                SplashKit.DrawBitmap(HeartBitmap, heartX, 0);
                heartX = heartX + 40;
            }
        }
    }

}
```

```
using System;
using SplashKitSDK;

public abstract class Robot
{
    public double X
    {
        get;
        set;
    }
    public double Y
    {
        get;
        set;
    }
    public Color MainColor
    {
        get;
        set;
    }

    private Vector2D Velocity { get; set; }

    public int Width
    {
        get
        {
            return 50;
        }
    }

    //read only property
    public int Height
    {
        get
        {
            return 50;
        }
    }

    public Circle CollisionCircle
    {
        get
        {
            return SplashKit.CircleAt(X+25, Y+25, 20);
        }
    }

   public Robot(Window gameWindow, Player player)
    {
        //Initial position of robot
        //Randomly pick.. Top/bottom or Left/Right
        if (SplashKit.Rnd() < 0.5)
        {
            //We picked...Top/Bottom

            //Start by picking the random position left to right (X)
            X = SplashKit.Rnd(gameWindow.Width);

            //Now work out if we are top or bottom?
            if (SplashKit.Rnd() < 0.5)
            {
                Y = -Height; //Top...so above top
            }
            else
            {
                Y = gameWindow.Height; //Bottom so below bottom
            }
        }
        else
        {
            //We picked..Left // Right
            Y = SplashKit.Rnd(gameWindow.Height);

            if (SplashKit.Rnd() < 0.5)
            {
                X = -Width;
            }
            else
            {
                X = gameWindow.Width;
            }
        }

        //process to add the velocity to robot
        const int SPEED = 4;

        //Get a point from Robot
        Point2D fromPt = new Point2D()
        {
            X = X,
            Y = Y
        };

        //Get a point from Player
        Point2D toPt = new Point2D()
        {
            X = player.X,
            Y = player.Y
        };

        //Calculate the direction to head
        Vector2D dir;
        dir = SplashKit.UnitVector(SplashKit.VectorPointToPoint(fromPt, toPt));

        //Set the speed and assign the velocity
        Velocity = SplashKit.VectorMultiply(dir, SPEED);
        //generate a random color for robot
        MainColor = Color.RandomRGB(200);

    }
 public bool IsOffscreen(Window screen)
    {
        return (X < -Width || X > screen.Width || Y < -Height || Y > screen.Height);
    }

 public abstract void Draw();
    public void Update()
    {
        X = X + Velocity.X;
        Y = Y + Velocity.Y;
    }


internal class Boxy : Robot
{
    public Boxy(Window gameWindow, Player player) : base(gameWindow, player)
    {

    }
    public override void Draw()
    {
        double eyeY, mouthY;
        double leftX = X + 12;
        double rightX = X + 27;
        eyeY = Y + 10;
        mouthY = Y + 30;

        SplashKit.FillRectangle(Color.Gray, X, Y, Width, Height);
        SplashKit.FillRectangle(MainColor, leftX, eyeY, 10, 10);
        SplashKit.FillRectangle(MainColor, rightX, eyeY, 10, 10);
        SplashKit.FillRectangle(MainColor, leftX, mouthY, 25, 10);
        SplashKit.FillRectangle(MainColor, leftX + 2, mouthY + 2, 21, 6);
    }
}

internal class Roundy : Robot
{
    public Roundy(Window gameWindow, Player player) : base(gameWindow, player)
    {
    }
    public override void Draw()
    {
        double leftX, midX, rightX;
        double midY, eyeY, mouthY;

        leftX = X + 17;
        midX = X + 25;
        rightX = X + 33;

        midY = Y + 25;
        eyeY = Y + 20;
        mouthY = Y + 35;

        SplashKit.FillCircle(Color.White, midX, midY, 25);
        SplashKit.DrawCircle(Color.Gray, midX, midY, 25);
        SplashKit.FillCircle(MainColor, leftX, eyeY, 5);
        SplashKit.FillCircle(MainColor, rightX, eyeY, 5);
        SplashKit.FillEllipse(Color.Gray, X, eyeY, 50, 30);
        SplashKit.DrawLine(Color.Black, X, mouthY, X + 50, Y + 35);
    }
}
   public class Custom : Robot
{
    public Custom(Window gameWindow, Player player) : base(gameWindow, player)
    {
    }
    public override void Draw()
    {
        double leftX, midX, rightX;
        double midY, eyeY, mouthY;

        leftX = X + 17;
        midX = X + 25;
        rightX = X + 33;

        midY = Y + 25;
        eyeY = Y + 20;
        mouthY = Y + 35;

        SplashKit.FillCircle(Color.Red, midX, midY, 25);
        SplashKit.FillCircle(Color.Blue, midX, midY, 20);
        SplashKit.FillCircle(Color.Green, midX, midY, 15);
        SplashKit.FillCircle(Color.Yellow, midX, midY, 10);
        SplashKit.FillCircle(Color.White, midX, midY, 5);
    }
}


}
```


Robot Dodge Game
SCORE: 10