**Car Racing: With Frameworks**



Session: 2021 – 2025

**Submitted by:**

2021-CS-61 Muhammad Waseem

**Supervised by:**

Dr. Awais Hassan

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

**Table of Contents:**

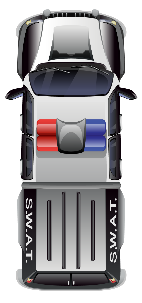
|  |  |
| --- | --- |
| Content | Page# |
| Description | 2 |
| Characters Description | 3 |
| Rules | 4 |
| Goal | 5 |
| Frameworks | 6 |
| Class Diagram | 7 |
| Sequence Diagram | 8 |
| Example | 9 |
| Complete Code | 10 |

**Description:**

Car Racing 2D is the top-down racing game in which the player is the driver. The Player Car can move forward, backward, Left and Right. The enemy cars will be coming from opposite sides. If the car collides with any of the enemy cars is health decreases by 25%. Enemy cars will be coming randomly. If the health becomes Zero the ends. There are HP as well if the players take HP its health increases by 25%.

**Game Characters:**

The game Characters include Player Car, Enemy Cars and HP bonuses. If the player car collides with enemy cars, its health decreases and by taking Hp its health increases.

**Rules:**

If the car collides with other enemy cars its health decreases. If the car collides with HP its health increases. If health approaches to Zero then game ends and the Player loses and if the player reaches a score of 200, he wins.

**Goal:**

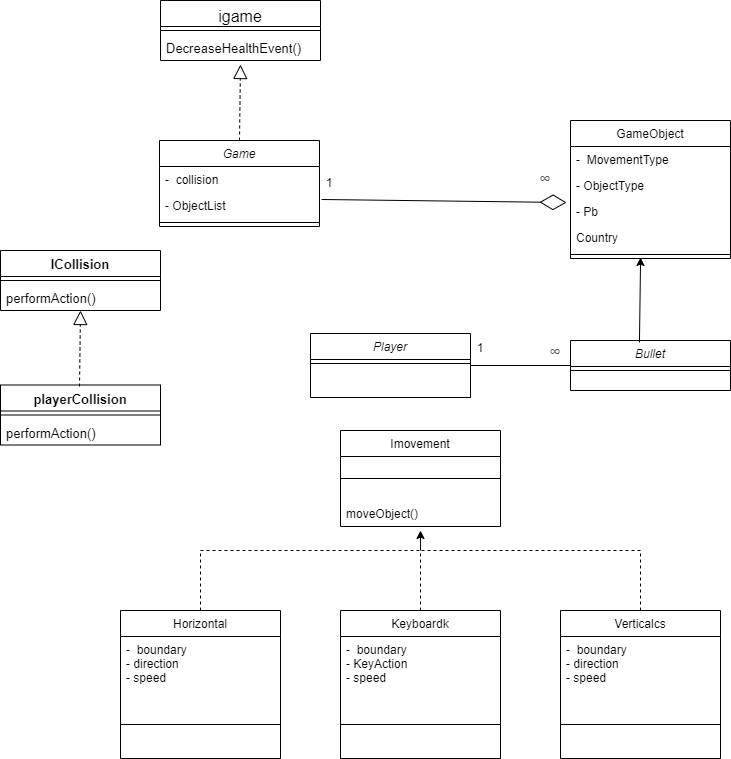
The Goal of the game is to score 200. If the player reaches the goal he wins otherwise h loses the game.

**Features of Frameworks:**

This Game Framework provides three features:

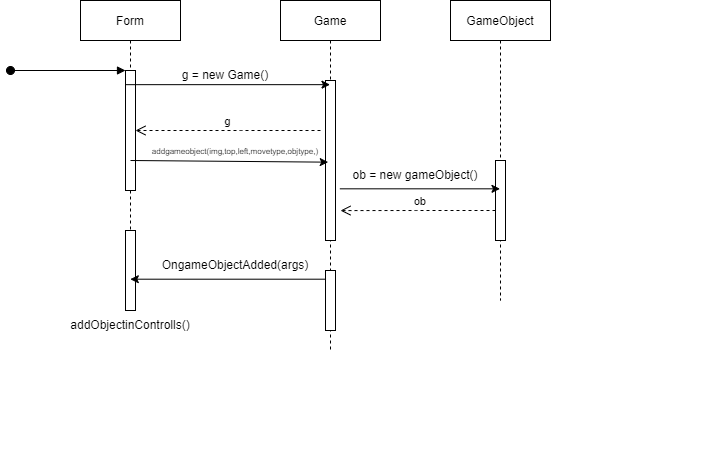
1. Movement Framework. (Vertical, horizontal, Keyboard)
2. Collision Framework.
3. Firing Framework.

**Class Diagram:**

****

**Sequence Diagram:**

**Add Game Object:**

****

**How to Use Framework:**

**Movement:**

This framework has all the basic movements of vertical, horizontal and with keyboard.

For using movement follow these steps:

* Create an object of **Game** Class.
* There is a function of **addgameobject** in Game class.
* You have to call it and Pass the arguments in the following way:

g.addGameObject(image, top, left, new movementtype(boundary, speed, "up or down"), "ObjectType")

Movement Types are:

* Vectical
* Horizontal
* Keyboard
* Event of addgameobject is made you have to call a function created by yourself of AddintoControls().
* At last you have to call an update fuction present in the Game class in Timetick event.

**Collision:**

* You have to make as many **Collision** Object as you want.
* Pass the following arguments in Constructor:

Collosion c = new Collosion("ObjecttType", "ObjectType", new PlayerCollosion())

* There are Events made for increasehealth , decreases health, increase score you have to call your self made functions according to the event.

**Complete Code:**

**Form:**

Game g;

Playerr p = new Playerr(Properties.Resources.Car\_2\_01, 380, 300);

Random rand = new Random();

System.Drawing.Point boundary = new System.Drawing.Point(300, 600);

System.Drawing.Point boundary2 = new System.Drawing.Point(275,5);

int health = 100;

int score = 0;

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

g = new Game();

g.OnObjectAdded += new EventHandler(AddIntoControls);

g.DecreaseHealth += new EventHandler(decreaseHealth);

g.IncreaseScore += new EventHandler(incScore);

g.increaseHealth += new EventHandler(IncreaseHealth);

p.AddBulletIntoControls += new EventHandler(addbulletsinControls);

g.addGameObject(p.Img, p.Top, p.Left, new Keyboardk(boundary2, 20), "player");

Collosion c = new Collosion("player", "enemy", new PlayerCollosion());

g.AddCollosion(c);

}

private void addbulletsinControls(object sender, EventArgs e)

{

this.Controls.Add((PictureBox)sender);

}

private void IncreaseHealth(object sender, EventArgs e)

{

health += 25;

}

private void incScore(object sender, EventArgs e)

{

score += 5;

}

private void decreaseHealth(object sender, EventArgs e)

{

this.Controls.Remove((PictureBox)sender);

health = health - 25;

PgBar.Value = health;

}

private void AddIntoControls(object sender, EventArgs e)

{

this.Controls.Add((PictureBox)sender);

}

private void Loop\_Tick(object sender, EventArgs e)

{

lblscore.Text = "Score" + score;

g.update();

EnemyCars();

getHealth();

g.DeleteObjects(this);

p.update();

p.DeleteFire(this);

if (health <= 0)

{

GameOver();

}

if (Keyboard.IsKeyPressed(Key.Space))

{

p.CreateBullet(Properties.Resources.laserBlue03);

}

}

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

{

g.KeyPressed(e.KeyCode);

}

public Image GetEnemyImage()

{

Image img = null;

int num = rand.Next(1, 4);

if (num == 1)

{

img = Properties.Resources.Car\_2\_011;

}

if (num == 2)

{

img = Properties.Resources.red;

}

else if (num == 3)

{

img = Properties.Resources.Car\_3\_01;

}

return img;

}

public void EnemyCars()

{

int num = rand.Next(0, 100);

if (num == 1)

{

Image e1 = GetEnemyImage();

g.addGameObject(e1, 5, 500, new Verticalcs(boundary, 18, "down"), "enemy");

}

if (num == 5)

{

Image e1 = GetEnemyImage();

g.addGameObject(e1, 5, 425, new Verticalcs(boundary, 18, "down"), "enemy");

}

if (num == 10)

{

Image e1 = GetEnemyImage();

g.addGameObject(e1, 5, 260, new Verticalcs(boundary, 18, "down"), "enemy");

}

if (num == 20)

{

Image e1 = GetEnemyImage();

g.addGameObject(e1, 5, 330, new Verticalcs(boundary, 18, "down"), "enemy");

}

}

public void getHealth()

{

int num = rand.Next(0, 200);

if (num == 28)

{

Image e1 = Properties.Resources.HP\_Dot;

g.addGameObject(e1, 5, 260, new Verticalcs(boundary, 16, "down"), "health");

}

if (num == 29)

{

Image e1 = Properties.Resources.HP\_Dot;

g.addGameObject(e1, 5, 260, new Verticalcs(boundary, 16, "down"), "health");

}

}

public void GameOver()

{

this.Close();

GameOver f = new GameOver();

f.Show();

}

**Frameworks:**

**Game class:**

public Game()

{

ObjectList = new List<GameObject>();

collosion = new List<Collosion>();

}

public void update()

{

int s = DetectCollosion();

foreach (GameObject ob in ObjectList)

{

ob.update();

}

}

public void addGameObject(Image img, int top, int left,Imovment MovementType,string ObjectType)

{

GameObject ob = new GameObject(img, top, left,MovementType,ObjectType);

ObjectList.Add(ob);

OnObjectAdded?.Invoke(ob.Pb1 , EventArgs.Empty);

}

public void KeyPressed(Keys keycode)

{

foreach (GameObject ob in ObjectList)

{

if(ob.MovementType1.GetType() == typeof(Keyboardk))

{

Keyboardk k = (Keyboardk)ob.MovementType1;

k.CheckKeyPressed(keycode);

}

}

}

public int DetectCollosion()

{

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

{

for(int j = 0; j < ObjectList.Count; j++)

{

if (ObjectList[i].Pb1.Bounds.IntersectsWith(ObjectList[j].Pb1.Bounds))

{

foreach (Collosion c in collosion)

{

if ((ObjectList[i].ObjectType1 == c.G1) && (ObjectList[j].ObjectType1 == c.G2))

{

c.Behaviour.PerformAction(this, ObjectList[i], ObjectList[j]);

if (ObjectList[i].ObjectType1 == "enemy")

{

ObjectList.RemoveAt(i);

return 1;

}

if (ObjectList[i].ObjectType1 == "health")

{

ObjectList.RemoveAt(i);

IncreaseHealthEvent();

return 1;

}

if (ObjectList[j].ObjectType1 == "enemy")

{

ObjectList.RemoveAt(j);

return 1;

}

if (ObjectList[j].ObjectType1 == "health")

{

ObjectList.RemoveAt(j);

IncreaseHealthEvent();

return 1;

}

}

}

}

}

}

return 0;

}

public void DecreaseHealthEvent(PictureBox PlGameObject)

{

DecreaseHealth?.Invoke(PlGameObject , EventArgs.Empty);

}

public void AddCollosion(Collosion c)

{

collosion.Add(c);

}

public void DeleteObjects(Form f)

{

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

{

if(ObjectList[i].Pb1.Top >= f.Height && ObjectList[i].ObjectType1 != "player")

{

ObjectList.RemoveAt(i);

IncreaseScore?.Invoke(" ", EventArgs.Empty);

}

}

}

public void IncreaseHealthEvent()

{

increaseHealth?.Invoke(" ",EventArgs.Empty);

}

**GameObject class:**

public class GameObject

{

private PictureBox Pb;

private Imovment MovementType;

private string ObjectType;

public GameObject(Image img , int top , int left, Imovment MovementType,string ObjectType)

{

Pb1 = new PictureBox();

Pb1.Image = img;

Pb1.Height = 60;

Pb1.Width = 35;

Pb1.SizeMode = PictureBoxSizeMode.StretchImage;

Pb1.BackColor = Color.Transparent;

Pb1.Top = top;

Pb1.Left = left;

this.MovementType1 = MovementType;

this.ObjectType1 = ObjectType;

}

public PictureBox Pb1 { get => Pb; set => Pb = value; }

public string ObjectType1 { get => ObjectType; set => ObjectType = value; }

internal Imovment MovementType1 { get => MovementType; set => MovementType = value; }

public void update()

{

Pb.Location = MovementType.MoveObject(Pb.Location);

}

}

**Collision class:**

public class Collosion

{

private string g1;

private string g2;

private Icollosion behaviour;

public Collosion(string g1, string g2, Icollosion behaviour)

{

this.G1 = g1;

this.G2 = g2;

this.Behaviour = behaviour;

}

public Icollosion Behaviour { get => behaviour; set => behaviour = value; }

public string G1 { get => g1; set => g1 = value; }

public string G2 { get => g2; set => g2 = value; }

}

**Horizontal class:**

public class Horizontal : Imovment

{

private Point boundary;

private int speed;

private string direction;

public Horizontal(Point boundary , int speed , string direction)

{

this.boundary = boundary;

this.speed = speed;

this.direction = direction;

}

public Point MoveObject(Point location)

{

if(location.X <= 0)

{

direction = "right";

}

else if (location.X + 50 > boundary.X)

{

direction = "left";

}

if(direction == "left")

{

location.X -= speed;

}

if(direction == "right")

{

location.X += speed;

}

return location;

}

}

**Keyboardk class:**

public class Keyboardk : Imovment

{

private Point boundary;

private int speed;

private string KeyAction;

public Keyboardk(Point boundary , int speed)

{

this.boundary = boundary;

this.speed = speed;

}

public void CheckKeyPressed(Keys keyCode)

{

if(keyCode == Keys.Up)

{

KeyAction = "up";

}

if(keyCode == Keys.Down)

{

KeyAction = "down";

}

if(keyCode == Keys.Left)

{

KeyAction = "left";

}

if(keyCode == Keys.Right)

{

KeyAction = "right";

}

}

public Point MoveObject(Point location)

{

if (KeyAction != null)

{

if (KeyAction == "up" && location.Y >= boundary.Y)

{

location.Y -= speed;

}

if (KeyAction == "down" && location.Y <= 375)

{

location.Y += speed;

}

if (KeyAction == "left" && location.X > boundary.X)

{

location.X -= speed;

}

if (KeyAction == "right" && location.X < 485)

{

location.X += speed;

}

KeyAction = null;

}

return location;

}

}

**Player Collision:**

public class PlayerCollosion : Icollosion

{

public void PerformAction(Igame game , GameObject g1 , GameObject g2)

{

GameObject ob = null;

if(g1.ObjectType1 == "enemy")

{

ob = g1;

}

else if(g2.ObjectType1 == "enemy")

{

ob = g2;

}

game.DecreaseHealthEvent(ob.Pb1);

}

}

**Playerr:**

public class Playerr

{

private Image img;

private int top;

private int left;

private List<PictureBox> bulletList = new List<PictureBox>();

public event EventHandler AddBulletIntoControls;

public Playerr(Image img,int top, int left)

{

this.Img = img;

this.Top = top;

this.Left = left;

}

public void CreateBullet(Image img)

{

PictureBox b = new PictureBox();

b.Image = img;

b.Top = this.top;

b.Left = this.left;

b.BackColor = Color.Transparent;

BulletList.Add(b);

AddBulletIntoControls?.Invoke(b, EventArgs.Empty);

}

public void update()

{

foreach (PictureBox p in bulletList)

{

p.Top -= 10;

}

}

public void DeleteFire(Form f)

{

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

{

if (BulletList[i].Top >= f.Height)

{

BulletList.RemoveAt(i);

}

}

}

public Image Img { get => img; set => img = value; }

public int Top { get => top; set => top = value; }

public int Left { get => left; set => left = value; }

public List<PictureBox> BulletList { get => bulletList; set => bulletList = value; }

}

**Icollision interface:**

public interface Icollosion

{

void PerformAction(Igame game , GameObject g1 , GameObject g2);

}

**Imovement interface:**

public interface Imovment

{

Point MoveObject(Point location);

}