**Tank Busters**



Session: 2022 – 2026

**Submitted by:**

Syed Muhammad Danish 2022-CS-46

**Submitted To:**

Sir Laeeq Khan

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

Contents

[1. Introduction: 3](#_Toc141610283)

[2. OOP Concepts: 3](#_Toc141610284)

[2.1 Association: 3](#_Toc141610285)

[2.2 Inheritance: 4](#_Toc141610286)

[2.3 Polymorphism 4](#_Toc141610287)

[2.4 Abstraction: 4](#_Toc141610288)

[3. Design Pattern Implementation: 4](#_Toc141610289)

[3.1 BL (Business Layer): 4](#_Toc141610290)

[4. CRC Diagram: 5](#_Toc141610291)

[5. Class Details: 6](#_Toc141610292)

[6. Conclusion: 7](#_Toc141610293)

[Complete Code: 7](#_Toc141610294)

## **Youtube Video Link**

https://youtu.be/dZ7ENnl8sM0

## **Introduction**:

By the advent of different technologies, the humans started to fight for dominance, making it difficult to create peace. These fights become so common that they were becoming trend of that era. Once a city named (SEEHO) attacked the neighboring city. But in response having more technologies the other city defeated them and took over the SEEHO also. So, in despise the SEEHO fighters again planned to attack the other city with more technology and tanks..............................................................CapPrice with a tank shape body that loves to gain dominace. He spends his days exploring the maze, and finding planning for how gain dominance. However, CapPrice’s life is disrupted by the arrival of four SEEHO fighters named Makrov, Makrovalpha, MakrovKnight, and MakrovRen. These fighters terrorize the maze and always lurk around every corner, trying to catch CapPrice and take over the city.--.............

Despite the danger, CapPrice refuses to give up. He knows that if he can find a safe way through the maze, he can turn the tables on the SEEHO fighters and kill them for points. So, CapPrice sets out on a quest to find a safe route and save his city. As CapPrice navigates the maze, he faces many obstacles and challenges. But he never loses hope and he never gives up. With his wits and cunning, he outmaneuvers the Tanks becoming stronger and more powerful with each victory. CapPrice eventually succeeds in finding a safe way through the maze. He becomes a hero, known for his bravery, determination, and never-say-die spirit. The Tanks may try to stop him from time to time, but CapPrice always perseveres and triumphs in the end. The legend of CapPrice lives on, inspiring new generations of players to explore the maze, defeat the Tanks and save the maze. CapPrice remains a timeless classic

that captivates and entertains players of all ages.

## **OOP Concepts:**

## 2.1 Association:

There is an association between Player/Enemy/Bullets and the game Cell. For a player or Enemy to exist, they must have a cell which is a Game cell. This is aggregation. There is also an association between Player/Enemy/Bullets with game direction.

## 2.2 Inheritance:

There is an inheritance between the player, enemy, bullets, and game object with the game Object as the parent class. It also has an inheritance between an enemy and its type.

## 

## 2.3 Polymorphism

There is Static and Dynamic Polymorphism in my project.

## 2.4 Abstraction:

Enemy class is abstract which leads to override the move function in its type. As all type of enemies have a unique pattern of moving(horizontal, vertical, random , smart).

By using oop data management and security improves and it’s easy to add new components with the help of classes than simple pf.

# Design Pattern Implementation:

## 3.1 BL (Business Layer):

* It has a class of game that returns the type of character and images depending upon their display character.
* There is a class of Gamecell which set the width and height of a cell at the of its creation and has a function that returns the next cell.
* It has a class of game object which determine its type and its image and its current cell.
* It has a class of game grid that load the grid from the file and maintain it.
* It has two enum classes of game direction and game object type.
* Then it has a class of player, bullets, and enemy which are inherited from the game object and control the move function.
* Enemy class is abstract which further inherits horizontal, vertical, and random enemies which have their move function accordingly.

# CRC Diagram:

## 



# Class Details:

**Game Object:** This Class contains the information about the game object which a cell holds.

**Game Cell:** It contains information about a cell and set its height and width when created.

**Game :** Game class returns the type of character and its image based on the display character.

**Game Grid:** It contains the information about grid like its rows and columns and loads the grid from file to form. The player, Bullet, and enemy are inherited from GameObject as they are an object of the game.  
**Makrov:** Makrov class is an abstract class with an abstract function of the move. Horizontal, vertical, and random enemies are inherited from it with their own move function. Enemies has only 1 live.

**CapPrice:** This class is an main class of player ship that is inherited with GameObject. The ship has 3 lives after one live ends the other start.

**Tanks:** This class is also abstract , that is further inherited to the bullet class of player and enemy.

**Random Object:** This class generates random object of green colour and when the player takes that object his score increases.

**MakrovRandom:** this class generated random enemy.

**MakrovSmart:** this class generated Chasing enemy which moves left and right. Making the game more interesting.

**MakrovHorizontal:** there are also horizontal enemies in the game.

**MakrovVertical:** there are also Vertical enemies in the game.

# Conclusion:

This 2D game project is a great accomplishment, demonstrating my proficiency in C# and my ability to learn and overcome challenges. While it still has room for improvement, it already incorporates various C# concepts and functionalities, showcasing your skills.This project showcases an impressive understanding of GUI-based development, implementing different types of move functions and collision detection. Even though I initially faced difficulties when converting it into a GUI, my determination and problem-solving skills enabled you to overcome these obstacles successfully. Overall, this 2D game project is an excellent showcase of my abilities and grip on C#. With some further refinement and fine-tuning, it has the potential to become even more professional and impressive.

## Complete Code:

**GameGL**:

internal class CapPrice:GameObject

{

bool isActive;

private int health;

private string flipPosition = "Right";

private bool flipBool = false;

private int jumpHeight = 0;

ProgressBar Caphealth;

Main form;

int level;

public CapPrice(Main form, int health, Image image, GameCell startCell) : base(GameObjectType.PLAYER, image)

{

this.health = health;

this.CurrentCell = startCell;

this.form = form;

Caphealth = new ProgressBar();

Caphealth = new ProgressBar();

Caphealth.Size = new Size(30, 7);

Caphealth.ForeColor = Color.Green;

Caphealth.BackColor = Color.Black;

Caphealth.Style = ProgressBarStyle.Continuous;

form.Controls.Add(Caphealth);

}

public int Health { get => health; set => health = value; }

public int getHealth()

{

return health;

}

public void setHealth(int health)

{

this.health = health;

}

public void setIsActive(bool set)

{

this.isActive = set;

}

public bool getIsActive()

{

return isActive;

}

public void increasHealth()

{

health = health + 5;

}

public void decreaseHealth(int damage)

{

health = health - damage;

}

public GameCell move(GameDirection direction)

{

GameCell currentCell = this.CurrentCell;

GameCell nextCell = currentCell.nextCell(direction);

addScores(nextCell);

if (nextCell.CurrentGameObject.GameObjectType == GameObjectType.KEY)

{

Game.levelClear = true;

Game.increaseScore(10);

}

this.CurrentCell = nextCell;

if (currentCell != nextCell)

{

currentCell.setGameObject(Game.getBlankGameObject());

}

return nextCell;

}

private void addScores(GameCell nextCell)

{

if (nextCell.CurrentGameObject.GameObjectType == GameObjectType.REWARD)

{

Game.score = Game.score + 5;

Game.palletcount--;

}

}

public int getJumpHeight()

{

return jumpHeight;

}

public void setJumpHeight(int height)

{

this.jumpHeight = height;

}

public void decreaseHieght()

{

jumpHeight--;

}

public string getFlipPosition()

{

return flipPosition;

}

public void setFlipPosition(string position)

{

flipPosition = position;

}

public bool getFlipBool()

{

return flipBool;

}

public void setFlipBool(bool flip)

{

flipBool = flip;

}

public void flipDeath()

{

if (flipPosition == "Left")

{

this.CurrentCell.PictureBox.Image = TankBusters.Properties.Resources.p1;

}

if (flipPosition == "Right")

{

this.CurrentCell.PictureBox.Image = TankBusters.Properties.Resources.p1\_flip;

}

}

public ProgressBar getBar()

{

return Caphealth;

}

public void setBarValue()

{

if (health >= 0)

{

Caphealth.Value = health;

}

}

public void setBarPosition()

{

Caphealth.Top = this.CurrentCell.X \* 46;

Caphealth.Left = this.CurrentCell.Y \* 50;

}

public bool getFlip()

{

return flipBool;

}

public void setFlip(bool flip)

{

flipBool = flip;

}

public void generateBullet()

{

PriceTanks b = new PriceTanks();

Image bullet = GameGL.Game.getGameObjectImage('Q');

GameCell startBullet = new GameCell();

if (this.getFlipPosition() == "Right")

{

GameCell next = this.CurrentCell.nextWallCell(GameDirection.Right);

if (next.CurrentGameObject.GameObjectType == GameObjectType.NONE)

{

startBullet = this.CurrentCell.nextCell(GameDirection.Right);

b = new PriceTanks(Game.enemies, GameDirection.Right, bullet, startBullet);

b.setIsLive(true);

Game.bullets.Add(b);

}

}

else if (this.getFlipPosition() == "Left")

{

GameCell next = this.CurrentCell.nextWallCell(GameDirection.Left);

if (next.CurrentGameObject.GameObjectType == GameObjectType.NONE)

{

startBullet = this.CurrentCell.nextCell(GameDirection.Left);

b = new PriceTanks(Game.enemies, GameDirection.Left, bullet, startBullet);

b.setIsLive(true);

Game.bullets.Add(b);

}

}

}

}

internal class Game

{

public static GameGrid grid;

public static CapPrice cap;

public static List<Tanks> bullets = new List<Tanks>();

public static List<Makrov> enemies = new List<Makrov>();

public static List<MakrovTanks> enemyBullets = new List<MakrovTanks>();

public static int score = 0;

public static int lives = 5;

public static int enemyDamage = 1;

public static int playerDamage = 5;

public static int palletcount = 6;

public static bool levelClear = false;

public static int level ;

private static int rewardDelay = 0;

public static void increaseScore(int scoreAdded)

{

score = score + scoreAdded;

}

public static GameObject getBlankGameObject()

{

GameObject blankGameObject = new GameObject(GameObjectType.NONE, null);

return blankGameObject;

}

public static Image getGameObjectImage(char displayCharacter)

{

Image img = null;

if (displayCharacter == '#' || displayCharacter == '%' || displayCharacter == '@')

{

img = TankBusters.Properties.Resources.wall;

}

if (displayCharacter == '!')

{

img = TankBusters.Properties.Resources.com\_resize\_unscreen;

}

if (displayCharacter == '|')

{

img = TankBusters.Properties.Resources.wallv;

}

if (displayCharacter == '$')

{

img = TankBusters.Properties.Resources.wallv;

}

if (displayCharacter == 'p' || displayCharacter == 'P')

{

img = TankBusters.Properties.Resources.p1;

}

if (displayCharacter == 'F') // enemy fire

{

img = TankBusters.Properties.Resources.fire\_pic;

}

if (displayCharacter == 'Q') // player fire

{

img = TankBusters.Properties.Resources.image\_removebg\_preview\_\_4\_;

}

if (displayCharacter == 'h')

{

img = TankBusters.Properties.Resources.e1;

}

if (displayCharacter == 'v')

{

img = TankBusters.Properties.Resources.e2\_flip;

}

if (displayCharacter == '.')

{

img = TankBusters.Properties.Resources.pf\_picc\_unscreen;

}

if (displayCharacter == 'x')

{

img = TankBusters.Properties.Resources.e3;

}

if (displayCharacter == 'r')

{

img = TankBusters.Properties.Resources.e4\_flip;

}

return img;

}

}

internal class GameCell

{

int row;

int col;

GameObject currentGameObject;

GameGrid grid;

PictureBox pictureBox;

const int width = 50;

const int height = 46;

public GameCell(int row, int col, GameGrid grid)

{

this.row = row;

this.col = col;

pictureBox = new PictureBox();

pictureBox.Left = col \* width;

pictureBox.Top = row \* height;

pictureBox.Size = new Size(width, height);

pictureBox.SizeMode = PictureBoxSizeMode.Zoom;

pictureBox.BackColor = Color.Transparent;

this.grid = grid;

}

public GameCell()

{

}

public void setGameObject(GameObject gameObject)

{

currentGameObject = gameObject;

pictureBox.Image = gameObject.Image;

}

public GameCell nextCell(GameDirection direction)

{

if (direction == GameDirection.Left)

{

if (this.col > 0)

{

GameCell ncell = grid.getCell(row, col - 1);

if (ncell.CurrentGameObject.GameObjectType == GameObjectType.NONE || ncell.CurrentGameObject.GameObjectType == GameObjectType.REWARD || ncell.CurrentGameObject.GameObjectType == GameObjectType.KEY)

{

return ncell;

}

}

}

if (direction == GameDirection.Right)

{

if (this.col < grid.Cols - 1)

{

GameCell ncell = grid.getCell(this.row, this.col + 1);

if (ncell.CurrentGameObject.GameObjectType == GameObjectType.NONE || ncell.CurrentGameObject.GameObjectType == GameObjectType.REWARD || ncell.CurrentGameObject.GameObjectType == GameObjectType.KEY)

{

return ncell;

}

}

}

if (direction == GameDirection.Up)

{

if (this.row > 0)

{

GameCell ncell = grid.getCell(this.row - 1, this.col);

if (ncell.CurrentGameObject.GameObjectType == GameObjectType.NONE || ncell.CurrentGameObject.GameObjectType == GameObjectType.REWARD || ncell.CurrentGameObject.GameObjectType == GameObjectType.KEY)

{

return ncell;

}

}

}

if (direction == GameDirection.Down)

{

if (this.row < grid.Rows - 1)

{

GameCell ncell = grid.getCell(this.row + 1, this.col);

if (ncell.CurrentGameObject.GameObjectType == GameObjectType.NONE || ncell.CurrentGameObject.GameObjectType == GameObjectType.REWARD || ncell.CurrentGameObject.GameObjectType == GameObjectType.KEY)

{

return ncell;

}

}

}

return this; // if can not return next cell return its own reference

}

public GameCell nextWallCell(GameDirection direction)

{

if (direction == GameDirection.Left)

{

if (this.col > 0)

{

GameCell ncell = grid.getCell(row, col - 1);

return ncell;

}

}

if (direction == GameDirection.Right)

{

if (this.col < grid.Cols - 1)

{

GameCell ncell = grid.getCell(this.row, this.col + 1);

return ncell;

}

}

if (direction == GameDirection.Up)

{

if (this.row > 0)

{

GameCell ncell = grid.getCell(this.row - 1, this.col);

return ncell;

}

}

if (direction == GameDirection.Down)

{

if (this.row < grid.Rows - 1)

{

GameCell ncell = grid.getCell(this.row + 1, this.col);

return ncell;

}

}

return this; // if can not return next cell return its own reference

}

public int X { get => row; set => row = value; }

public int Y { get => col; set => col = value; }

public GameObject CurrentGameObject { get => currentGameObject; set => currentGameObject = value; }

public PictureBox PictureBox { get => pictureBox; set => pictureBox = value; }

}

public enum GameDirection

{

Left,

Right,

Up,

Down

}

internal class GameGrid

{

GameCell[,] cells;

int rows;

int cols;

public GameGrid(String fileName, int rows, int cols)

{

//Numbers of rows and cols should load from the text file

this.rows = rows;

this.cols = cols;

cells = new GameCell[rows, cols];

this.loadGrid(fileName);

}

public GameCell getCell(int x, int y)

{

return cells[x, y];

}

public int getRows()

{

return rows;

}

public int getCols()

{

return cols;

}

public int Rows { get => rows; set => rows = value; }

public int Cols { get => cols; set => cols = value; }

void loadGrid(string fileName)

{

StreamReader fp = new StreamReader(fileName);

string record;

for (int row = 0; row < this.rows; row++)

{

record = fp.ReadLine();

for (int col = 0; col < this.cols; col++)

{

GameCell cell = new GameCell(row, col, this);

char displayCharacter = record[col];

GameObjectType type = GameObject.getGameObjectType(displayCharacter);

Image displayIamge = Game.getGameObjectImage(displayCharacter);

GameObject gameObject = new GameObject(type, displayIamge);

cell.setGameObject(gameObject);

cells[row, col] = cell;

}

}

fp.Close();

}

}

internal class GameObject

{

GameObjectType gameObjectType;

GameCell currentCell;

Image image;

public GameObject()

{

}

public GameObject(GameObjectType type, Image image)

{

this.gameObjectType = type;

this.Image = image;

}

public static GameObjectType getGameObjectType(char displayCharacter)

{

if (displayCharacter == '.')

{

return GameObjectType.REWARD;

}

if (displayCharacter == 'p' || displayCharacter == 'P')

{

return GameObjectType.PLAYER;

}

if (displayCharacter == '%' || displayCharacter == '#')

{

return GameObjectType.WALL;

}

if (displayCharacter == '|' || displayCharacter == '@')

{

return GameObjectType.V\_WALL;

}

if (displayCharacter == '$')

{

return GameObjectType.OBSTACLE;

}

if (displayCharacter == '!')

{

return GameObjectType.FAN;

}

return GameObjectType.NONE;

}

public GameObjectType GameObjectType { get => gameObjectType; set => gameObjectType = value; }

public GameCell CurrentCell

{

get => currentCell;

set

{

currentCell = value;

currentCell.setGameObject(this);

}

}

public Image Image { get => image; set => image = value; }

}

public enum GameObjectType

{

WALL,

PLAYER,

ENEMY,

REWARD,

BULLET,

V\_WALL,

OBSTACLE,

KEY,

PLAYERBULLET,

FAN,

NONE

} internal abstract class Makrov:GameObject

{

protected ProgressBar enemyBar = new ProgressBar();

protected int health;

protected CapPrice Captain;

protected string flipPosition = "Right";

protected bool flipBool = false;

Form form;

bool isActive;

protected GameDirection direction;

protected GameDirection faceDirection;

protected int bulletDelay = 1;

public Makrov(Form form, int health, GameObjectType gameObjectType, Image image) : base(GameObjectType.ENEMY, image)

{

this.health = health;

this.form = form;

enemyBar = new ProgressBar();

enemyBar.Size = new Size(30, 7);

enemyBar.ForeColor = Color.Red;

enemyBar.BackColor = Color.Black;

enemyBar.Style = ProgressBarStyle.Continuous;

isActive = true;

form.Controls.Add(enemyBar);

}

public abstract GameCell move();

public string getFlipPosition()

{

return flipPosition;

}

public void setFlipPosition(string position)

{

flipPosition = position;

}

public bool getFlipBool()

{

return flipBool;

}

public void setFlipBool(bool flip)

{

flipBool = flip;

}

public void setIsActive(bool set)

{

this.isActive = set;

}

public bool getIsActive()

{

return isActive;

}

public int getHealth()

{

return health;

}

public void increasHealth()

{

health = health + 5;

}

public void decreaseHealth(int damage)

{

health = health - damage;

}

public ProgressBar getBar()

{

return enemyBar;

}

public void setBarValue()

{

if (health >= 0)

{

enemyBar.Value = health;

}

}

public void setBarPosition()

{

enemyBar.Top = this.CurrentCell.X \* 46;

enemyBar.Left = this.CurrentCell.Y \* 50;

}

void generateNormalBullet()

{

if (Captain.CurrentCell.X == this.CurrentCell.X)

{

if (bulletDelay == 2)

{

this.setFlipBool(true);

}

if (bulletDelay % 3 == 0)

{

this.setFlipBool(false);

MakrovTanks b = new MakrovTanks();

Image bullet = TankBusters.Properties.Resources.fire\_pic;

GameCell startBullet = new GameCell();

startBullet = this.CurrentCell.nextCell(faceDirection);

b = new MakrovTanks(Captain, faceDirection, bullet, startBullet);

b.setIsLive(true);

Game.enemyBullets.Add(b);

}

bulletDelay++;

}

else

{

bulletDelay = 1;

}

}

public void generateBullet()

{

GameCell currentCell = this.CurrentCell;

GameCell nextCell = currentCell.nextWallCell(faceDirection);

if (nextCell.CurrentGameObject.GameObjectType == GameObjectType.NONE)

{

generateNormalBullet();

}

}

public abstract void flipEnemy();

protected void enemyDirection()

{

if (Captain.CurrentCell.Y < this.CurrentCell.Y)

{

flipPosition = "Left";

faceDirection = GameDirection.Left;

}

else if (Captain.CurrentCell.Y > this.CurrentCell.Y)

{

flipPosition = "Right";

faceDirection = GameDirection.Right;

}

}

}

interface iSmartEnemy

{

double calculateDistance(GameCell nextcell);

}

internal class MakrovHorizontal:Makrov

{

public MakrovHorizontal(Main form, int lives, CapPrice Captain, GameDirection direction, Image image, GameCell startCell) : base(form, lives, GameObjectType.ENEMY, image)

{

this.direction = direction;

this.CurrentCell = startCell;

this.Captain = Captain;

}

public override GameCell move()

{

if (this.getIsActive() == true)

{

if (Captain.CurrentCell.X != this.CurrentCell.X)

{

GameCell currentCell = this.CurrentCell;

GameCell nextCell = currentCell.nextCell(direction);

GameCell nextCell2 = currentCell.nextWallCell(direction);

GameObject previousObject = nextCell.CurrentGameObject;

if ((nextCell.CurrentGameObject.GameObjectType != GameObjectType.KEY))

{

this.CurrentCell = nextCell;

}

GameCell downCell = nextCell.nextWallCell(GameDirection.Down);

if (currentCell == nextCell || (nextCell.CurrentGameObject.GameObjectType == GameObjectType.KEY))

{

manageDirections();

}

if (currentCell != nextCell)

{

if ((nextCell.CurrentGameObject.GameObjectType != GameObjectType.KEY))

{

currentCell.setGameObject(previousObject);

}

if (downCell.CurrentGameObject.GameObjectType == GameObjectType.V\_WALL)

{

manageDirections();

}

this.setFlipBool(true);

}

}

else if (Captain.CurrentCell.X == this.CurrentCell.X)

{

base.enemyDirection();

}

}

return null;

}

public void manageDirections()

{

if (direction == GameDirection.Left)

{

direction = GameDirection.Right;

flipPosition = "Right";

}

else if (direction == GameDirection.Right)

{

direction = GameDirection.Left;

flipPosition = "Left";

}

}

public override void flipEnemy()

{

if (flipPosition == "Left")

{

this.CurrentCell.PictureBox.Image = TankBusters.Properties.Resources.e1;

faceDirection = GameDirection.Left;

}

else if (flipPosition == "Right")

{

this.CurrentCell.PictureBox.Image = TankBusters.Properties.Resources.e1\_flip;

faceDirection = GameDirection.Right;

}

}

}

internal class MakrovRandom:Makrov

{

int randomDelay;

int random;

int moveDelay = 11;

bool moveDelayChanger;

public MakrovRandom(Form form,int lives, CapPrice Captain, GameDirection direction, Image image, GameCell startCell) : base(form, lives, GameObjectType.ENEMY, image)

{

this.direction = direction;

this.CurrentCell = startCell;

this.Captain = Captain;

}

public override GameCell move()

{

if (this.getIsActive() == true)

{

if (Captain.CurrentCell.X == this.CurrentCell.X && moveDelayChanger == true)

{

moveDelay = 0;

moveDelayChanger = false;

}

if (moveDelay > 10)

{

manageDirections();

GameCell currentCell = this.CurrentCell;

GameCell nextCell = currentCell.nextCell(direction);

GameObject previousObject = nextCell.CurrentGameObject;

if ((nextCell.CurrentGameObject.GameObjectType != GameObjectType.KEY))

{

this.CurrentCell = nextCell;

}

if (currentCell != nextCell)

{

if ((nextCell.CurrentGameObject.GameObjectType != GameObjectType.KEY))

{

currentCell.setGameObject(previousObject);

}

}

this.setFlipBool(true);

if (moveDelay > 30)

{

moveDelayChanger = true;

}

}

moveDelay++;

}

return null;

}

public void manageDirections()

{

if (randomDelay % 5 == 0)

{

Random r = new Random();

random = r.Next(4);

}

base.enemyDirection();

if (random == 0)

{

direction = GameDirection.Right;

}

else if (random == 1)

{

direction = GameDirection.Left;

}

else if (random == 2)

{

direction = GameDirection.Up;

}

else if (random == 3)

{

direction = GameDirection.Down;

}

randomDelay++;

}

public override void flipEnemy()

{

if (flipPosition == "Left")

{

this.CurrentCell.PictureBox.Image = TankBusters.Properties.Resources.e4\_flip;

faceDirection = GameDirection.Left;

}

else if (flipPosition == "Right")

{

this.CurrentCell.PictureBox.Image = TankBusters.Properties.Resources.e4;

faceDirection = GameDirection.Right;

}

}

}

internal class MakrovSmart:Makrov,iSmartEnemy

{

int speed;

public MakrovSmart(Form form, int lives, CapPrice Captain, GameDirection direction, Image image, GameCell startCell) : base(form, lives, GameObjectType.ENEMY, image)

{

this.direction = direction;

this.CurrentCell = startCell;

this.Captain = Captain;

this.speed = 1;

}

public int getSpeed()

{

return speed;

}

public override void flipEnemy()

{

if (flipPosition == "Left")

{

this.CurrentCell.PictureBox.Image = TankBusters.Properties.Resources.e3;

faceDirection = GameDirection.Left;

}

else if (flipPosition == "Right")

{

this.CurrentCell.PictureBox.Image = TankBusters.Properties.Resources.e3\_flip;

faceDirection = GameDirection.Right;

}

}

public override GameCell move()

{

if (getIsActive() == true)

{

speed++;

if (speed % 3 == 0)

{

manageDirections();

GameCell currentCell = this.CurrentCell;

GameObjectType nexttype = currentCell.nextCell(direction).CurrentGameObject.GameObjectType;

GameCell nextCell = currentCell.nextCell(direction);

if ((nextCell.CurrentGameObject.GameObjectType != GameObjectType.KEY))

{

this.CurrentCell = nextCell;

}

enemyDirection();

if (nextCell.CurrentGameObject.GameObjectType == GameObjectType.PLAYER)

{

this.Captain.decreaseHealth (Game.enemyDamage);

}

if (currentCell != nextCell)

{

if ((nextCell.CurrentGameObject.GameObjectType != GameObjectType.KEY))

{

currentCell.setGameObject(Game.getBlankGameObject());

}

}

setFlipBool(true);

return nextCell;

}

}

return null;

}

public void manageDirections()

{

double[] distance = new double[4] { 10000, 10000, 10000, 10000 };

if (this.CurrentCell.nextCell(GameDirection.Left).CurrentGameObject.GameObjectType != GameObjectType.WALL)

{

distance[0] = calculateDistance(this.CurrentCell.nextCell(GameDirection.Left));

}

if (this.CurrentCell.nextCell(GameDirection.Right).CurrentGameObject.GameObjectType != GameObjectType.WALL)

{

distance[1] = calculateDistance(this.CurrentCell.nextCell(GameDirection.Right));

}

if (this.CurrentCell.nextCell(GameDirection.Up).CurrentGameObject.GameObjectType != GameObjectType.WALL)

{

distance[2] = calculateDistance(this.CurrentCell.nextCell(GameDirection.Up));

}

if (this.CurrentCell.nextCell(GameDirection.Down).CurrentGameObject.GameObjectType != GameObjectType.WALL)

{

distance[3] = calculateDistance(this.CurrentCell.nextCell(GameDirection.Down));

}

if (distance[0] <= distance[1] && distance[0] <= distance[2] && distance[0] <= distance[3])

{

this.direction = GameDirection.Left;

flipPosition = "Left";

faceDirection = GameDirection.Left;

}

if (distance[1] <= distance[0] && distance[1] <= distance[2] && distance[1] <= distance[3])

{

this.direction = GameDirection.Right;

flipPosition = "Right";

faceDirection = GameDirection.Right;

}

if (distance[2] <= distance[0] && distance[2] <= distance[1] && distance[2] <= distance[3])

{

this.direction = GameDirection.Up;

}

if (distance[3] <= distance[0] && distance[3] <= distance[1] && distance[3] <= distance[2])

{

this.direction = GameDirection.Down;

}

}

public double calculateDistance(GameCell nextcell)

{

return Math.Sqrt(Math.Pow((Captain.CurrentCell.X - nextcell.X), 2) + Math.Pow((Captain.CurrentCell.Y - nextcell.Y), 2));

}

}

internal class MakrovTanks : Tanks

{

CapPrice captian;

public MakrovTanks(CapPrice captian, GameDirection direction, Image image, GameCell startCell) : base(GameObjectType.BULLET, image)

{

this.direction = direction;

this.CurrentCell = startCell;

this.captian = captian;

}

public MakrovTanks()

{

}

public override GameCell move()

{

if (getIsLive() == true)

{

GameCell currentCell = this.CurrentCell;

GameCell nextCell = currentCell.nextCell(direction);

GameCell nextCell2 = currentCell.nextWallCell(direction);

if (nextCell.CurrentGameObject.GameObjectType != GameObjectType.REWARD)

{

this.CurrentCell = nextCell;

}

GameObject previousObject = nextCell.CurrentGameObject;

GameObject nextObject = nextCell2.CurrentGameObject;

if (currentCell != nextCell)

{

if (nextCell.CurrentGameObject.GameObjectType != GameObjectType.REWARD)

{

currentCell.setGameObject(Game.getBlankGameObject());

}

else

{

CurrentCell.setGameObject(Game.getBlankGameObject());

this.isActive = false;

}

}

if (currentCell == nextCell || nextCell.CurrentGameObject.GameObjectType == GameObjectType.KEY)

{

if (nextObject.GameObjectType == GameObjectType.PLAYER)

{

captian.decreaseHealth(Game.enemyDamage);

}

currentCell.setGameObject(Game.getBlankGameObject());

this.setIsLive(false);

}

return nextCell;

}

return null;

}

}

internal class MakrovVertical:Makrov

{

public MakrovVertical(Form form, int lives, CapPrice Captain, GameDirection direction, Image image, GameCell startCell) : base(form, lives, GameObjectType.ENEMY, image)

{

this.direction = direction;

this.CurrentCell = startCell;

this.Captain = Captain;

}

public override GameCell move()

{

if (this.getIsActive() == true)

{

if (Captain.CurrentCell.X != this.CurrentCell.X)

{

GameCell currentCell = this.CurrentCell;

GameCell nextCell = currentCell.nextCell(direction);

GameCell nextCell2 = currentCell.nextWallCell(direction);

GameObject previousObject = nextCell.CurrentGameObject;

if ((nextCell.CurrentGameObject.GameObjectType != GameObjectType.WALL))

{

this.CurrentCell = nextCell;

}

if (currentCell == nextCell || (nextCell.CurrentGameObject.GameObjectType == GameObjectType.WALL))

{

manageDirections();

}

if (currentCell != nextCell)

{

if ((nextCell.CurrentGameObject.GameObjectType != GameObjectType.WALL))

{

currentCell.setGameObject(previousObject);

}

this.setFlipBool(true);

}

}

else if (Captain.CurrentCell.X == this.CurrentCell.X)

{

base.enemyDirection();

// generateBullet();

}

}

return null;

}

public void manageDirections()

{

if (direction == GameDirection.Down)

{

direction = GameDirection.Up;

base.enemyDirection();

}

else if (direction == GameDirection.Up)

{

direction = GameDirection.Down;

flipPosition = "Right";

base.enemyDirection();

}

}

public override void flipEnemy()

{

}

}

internal class PriceTanks:Tanks

{

private List<Makrov> enemies;

public PriceTanks()

{

}

public PriceTanks(List<Makrov> enemies, GameDirection direction, Image image, GameCell startCell) : base(GameObjectType.PLAYERBULLET, image)

{

this.direction = direction;

this.CurrentCell = startCell;

this.enemies = enemies;

}

public override GameCell move()

{

if (getIsLive() == true)

{

GameCell currentCell = this.CurrentCell;

GameCell nextCell = currentCell.nextCell(direction);

GameCell nextCell2 = currentCell.nextWallCell(direction);

if (nextCell.CurrentGameObject.GameObjectType != GameObjectType.REWARD)

{

this.CurrentCell = nextCell;

}

GameObject previousObject = nextCell.CurrentGameObject;

GameObject nextObject = nextCell2.CurrentGameObject;

if (currentCell != nextCell)

{

if ((nextCell.CurrentGameObject.GameObjectType != GameObjectType.REWARD))

{

currentCell.setGameObject(Game.getBlankGameObject());

}

else

{

CurrentCell.setGameObject(Game.getBlankGameObject());

this.isActive = false;

}

}

if (currentCell == nextCell || nextCell.CurrentGameObject.GameObjectType == GameObjectType.KEY)

{

if (nextObject.GameObjectType == GameObjectType.ENEMY)

{

Game.increaseScore(1);

foreach (Makrov enemy in enemies)

{

GameCell next = enemy.CurrentCell.nextWallCell(GameDirection.Left);

GameObject obj = next.CurrentGameObject;

GameCell next2 = enemy.CurrentCell.nextWallCell(GameDirection.Right);

GameObject obj2 = next2.CurrentGameObject;

if (obj.GameObjectType == GameObjectType.PLAYERBULLET)

{

enemy.decreaseHealth(Game.playerDamage);

}

if (obj2.GameObjectType == GameObjectType.PLAYERBULLET)

{

enemy.decreaseHealth(Game.playerDamage);

}

}

}

currentCell.setGameObject(Game.getBlankGameObject());

this.setIsLive(false);

}

return nextCell;

}

return null;

}

}

internal abstract class Tanks : GameObject

{

protected bool isActive = false;

protected GameDirection direction;

public Tanks(GameObjectType gameObjectType, Image image) : base(gameObjectType, image)

{

}

public GameDirection getDirection()

{

return direction;

}

public Tanks()

{

}

public abstract GameCell move();

public void setIsLive(bool set)

{

this.isActive = set;

}

public bool getIsLive()

{

return isActive;

}

}

**Forms:**

public partial class Form3 : Form

{

int count = 0;

public Form3()

{

InitializeComponent();

}

private void timer1\_Tick(object sender, EventArgs e)

{

count++;

if (count == 3)

{

this.Dispose();

}

}

}

public partial class Loading : Form

{

public Loading()

{

InitializeComponent();

}

private void timer1\_Tick(object sender, EventArgs e)

{

guna2ProgressBar1.Value += 10;

if (guna2ProgressBar1.Value >= 100)

{

this.Hide();

this.timer1.Stop();

Main f = new Main();

f.Show();

}

}

}

public partial class Main : Form

{

int count = 0;

public Main()

{

InitializeComponent();

}

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

{

MakrovBulletsMovements();

CaptainBulletsMovements();

}

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

{

if (Keyboard.IsKeyPressed(Key.LeftArrow))

{

Game.cap.move(GameDirection.Left);

if (Game.cap.getFlipPosition() == "Right")

{

Game.cap.setFlipPosition("Left");

}

Game.cap.setFlip(true);

}

else if (Keyboard.IsKeyPressed(Key.RightArrow))

{

Game.cap.move(GameDirection.Right);

if (Game.cap.getFlipPosition() == "Left")

{

Game.cap.setFlipPosition("Right");

}

Game.cap.setFlip(true);

}

if (Game.cap.CurrentCell.nextWallCell(GameDirection.Down).CurrentGameObject.GameObjectType == GameObjectType.FAN)

{

Game.cap.setJumpHeight(2);

}

if (Keyboard.IsKeyPressed(Key.Space))

{

Game.cap.generateBullet();

}

if (Game.cap.getJumpHeight() > 0)

{

Game.cap.move(GameDirection.Up);

Game.cap.setFlip(true);

Game.cap.decreaseHieght();

}

else if (Game.cap.CurrentCell.nextWallCell(GameDirection.Down).CurrentGameObject.GameObjectType == GameObjectType.NONE)

{

Game.cap.move(GameDirection.Down);

Game.cap.setFlip(true);

}

foreach (Makrov enemy in Game.enemies)

{

if (enemy.getIsActive() == true)

{

enemy.move();

enemy.generateBullet();

if (enemy.getFlipBool() == true)

{

enemy.flipEnemy();

enemy.setFlipBool(false);

}

enemy.setBarPosition();

enemy.setBarValue();

}

if (enemy.getHealth() <= 0)

{

enemy.setIsActive(false);

GameObject gameObj = Game.getBlankGameObject();

this.Controls.Remove(enemy.getBar());

enemy.CurrentCell.setGameObject(gameObj);

}

}

for (int x = 0; x < Game.enemies.Count; x++)

{

if (Game.enemies[x].getIsActive() == false)

{

Game.enemies.Remove(Game.enemies[x]);

}

}

if (Game.cap.getHealth() == 0)

{

if (Game.lives > 0)

{

Game.cap.setHealth(100);

Game.lives--;

}

else if (Game.lives == 0)

{

GameTimer.Enabled = false;

MessageBox.Show("GameOver");

this.Dispose();

//MainMenu m = new MainMenu();

}

}

if (Game.cap.getFlip() == true)

{

Game.cap.flipDeath();

Game.cap.setFlip(false);

}

Game.cap.setBarPosition();

Game.cap.setBarValue();

label2.Text = "Score: " + Game.score;

//lives.Text = "Lives: " + Game.lives;

if (Game.enemies.Count == 0 && Game.palletcount == 0)

{

Game.levelClear = true;

if (Game.levelClear == true)

{

GameTimer.Enabled = false;

if (Game.level < 2)

{

Game.level++;

Game.levelClear = false;

this.Dispose();

Main g = new Main();

if (Game.level == 2 )

{

Form3 f = new Form3();

f.Show();

f.BringToFront();

Thread.Sleep(10000);

}

g.Show();

//

}

else if (Game.level >= 2)

{

MessageBox.Show("GameComplete");

Game.levelClear = false;

this.Dispose();

}

// MessageBox.Show("WHTA");

}

// levelchanger();

}

}

void CaptainBulletsMovements()

{

foreach (Tanks b in Game.bullets)

{

b.move();

}

for (int x = 0; x < Game.bullets.Count; x++)

{

if (Game.bullets[x].getIsLive() == false)

{

Game.bullets.RemoveAt(x);

}

}

}

void MakrovBulletsMovements()

{

foreach (Tanks b in Game.enemyBullets)

{

b.move();

}

for (int x = 0; x < Game.enemyBullets.Count; x++)

{

if (Game.enemyBullets[x].getIsLive() == false)

{

Game.enemyBullets.RemoveAt(x);

}

}

}

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

{

levelchanger();

}

void levelchanger()

{

if (Game.level == 0)

{

Game.enemyDamage = 3;

Game.grid = new GameGrid("maze2.txt", 15, 27);

Image Image = GameGL.Game.getGameObjectImage('P');

GameCell startCell = Game.grid.getCell(12, 23);

Game.cap = new CapPrice(this, 100, Image, startCell);

Image Img = GameGL.Game.getGameObjectImage('h');

GameCell h1 = Game.grid.getCell(1, 12);

MakrovHorizontal enemy = new MakrovHorizontal(this, 100, Game.cap, GameDirection.Left, Img, h1);

Game.enemies.Add(enemy);

Image Img2 = GameGL.Game.getGameObjectImage('h');

GameCell h2 = Game.grid.getCell(7, 12);

MakrovHorizontal enemy3 = new MakrovHorizontal(this, 100, Game.cap, GameDirection.Right, Img2, h2);

Game.enemies.Add(enemy3);

Image Img1 = GameGL.Game.getGameObjectImage('v');

GameCell v1 = Game.grid.getCell(4, 1);

MakrovVertical enemy1 = new MakrovVertical(this, 100, Game.cap, GameDirection.Down, Img1, v1);

Game.enemies.Add(enemy1);

printMaze(Game.grid);

}

if (Game.level == 1)

{

Game.palletcount = 6;

Game.enemyDamage = 5;

Game.grid = new GameGrid("maze.txt", 15, 27);

Image Image = GameGL.Game.getGameObjectImage('P');

GameCell startCell = Game.grid.getCell(13 , 23);

Game.cap = new CapPrice(this, 100, Image, startCell);

Image Img = GameGL.Game.getGameObjectImage('x');

GameCell s1 = Game.grid.getCell(1, 12);

MakrovSmart e = new MakrovSmart(this, 100, Game.cap, GameDirection.Left, Img, s1);

Game.enemies.Add(e);

Image Img2 = GameGL.Game.getGameObjectImage('r');

GameCell h2 = Game.grid.getCell(7, 12);

MakrovRandom e3 = new MakrovRandom(this, 100, Game.cap, GameDirection.Right, Img2, h2);

Game.enemies.Add(e3);

printMaze(Game.grid);

}

else if (Game.level == 2 )

{

Game.enemyDamage = 5;

Game.grid = new GameGrid("maze1.txt", 15, 27);

Image Image = GameGL.Game.getGameObjectImage('P');

GameCell startCell = Game.grid.getCell(12, 23);

Game.cap = new CapPrice(this, 100, Image, startCell);

Image Img = GameGL.Game.getGameObjectImage('h');

GameCell h1 = Game.grid.getCell(1, 12);

MakrovHorizontal enemy = new MakrovHorizontal(this, 100, Game.cap, GameDirection.Left, Img, h1);

Game.enemies.Add(enemy);

Image Img2 = GameGL.Game.getGameObjectImage('h');

GameCell h2 = Game.grid.getCell(5, 16);

MakrovHorizontal enemy3 = new MakrovHorizontal(this, 100, Game.cap, GameDirection.Right, Img2, h2);

Game.enemies.Add(enemy3);

Image Img1 = GameGL.Game.getGameObjectImage('v');

GameCell v1 = Game.grid.getCell(4, 1);

MakrovVertical enemy1 = new MakrovVertical(this, 100, Game.cap, GameDirection.Down, Img1, v1);

Game.enemies.Add(enemy1);

Image Img3 = GameGL.Game.getGameObjectImage('v');

GameCell v2 = Game.grid.getCell(2, 24);

MakrovVertical enemy2 = new MakrovVertical(this, 100, Game.cap, GameDirection.Down, Img3, v2);

Game.enemies.Add(enemy2);

Image Img4 = GameGL.Game.getGameObjectImage('x');

GameCell s1 = Game.grid.getCell(12, 12);

MakrovSmart enemy4 = new MakrovSmart(this, 100, Game.cap, GameDirection.Left, Img4, s1);

Game.enemies.Add(enemy4);

Image Img5 = GameGL.Game.getGameObjectImage('r');

GameCell r = Game.grid.getCell(7, 19);

MakrovRandom enemy5 = new MakrovRandom(this, 100, Game.cap, GameDirection.Right, Img5, r);

Game.enemies.Add(enemy5);

Image Img6 = GameGL.Game.getGameObjectImage('r');

GameCell r1 = Game.grid.getCell(11, 4);

MakrovRandom enemy6 = new MakrovRandom(this, 100, Game.cap, GameDirection.Right, Img6, r1);

Game.enemies.Add(enemy6);

printMaze(Game.grid);

}

}

void printMaze(GameGrid grid)

{

for (int x = 0; x < grid.Rows; x++)

{

for (int y = 0; y < grid.Cols; y++)

{

GameCell cell = grid.getCell(x, y);

this.Controls.Add(cell.PictureBox);

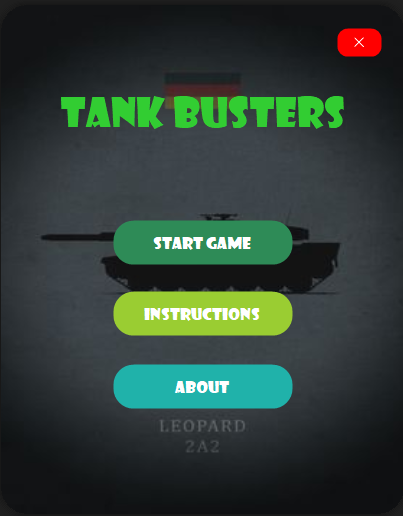
}

}

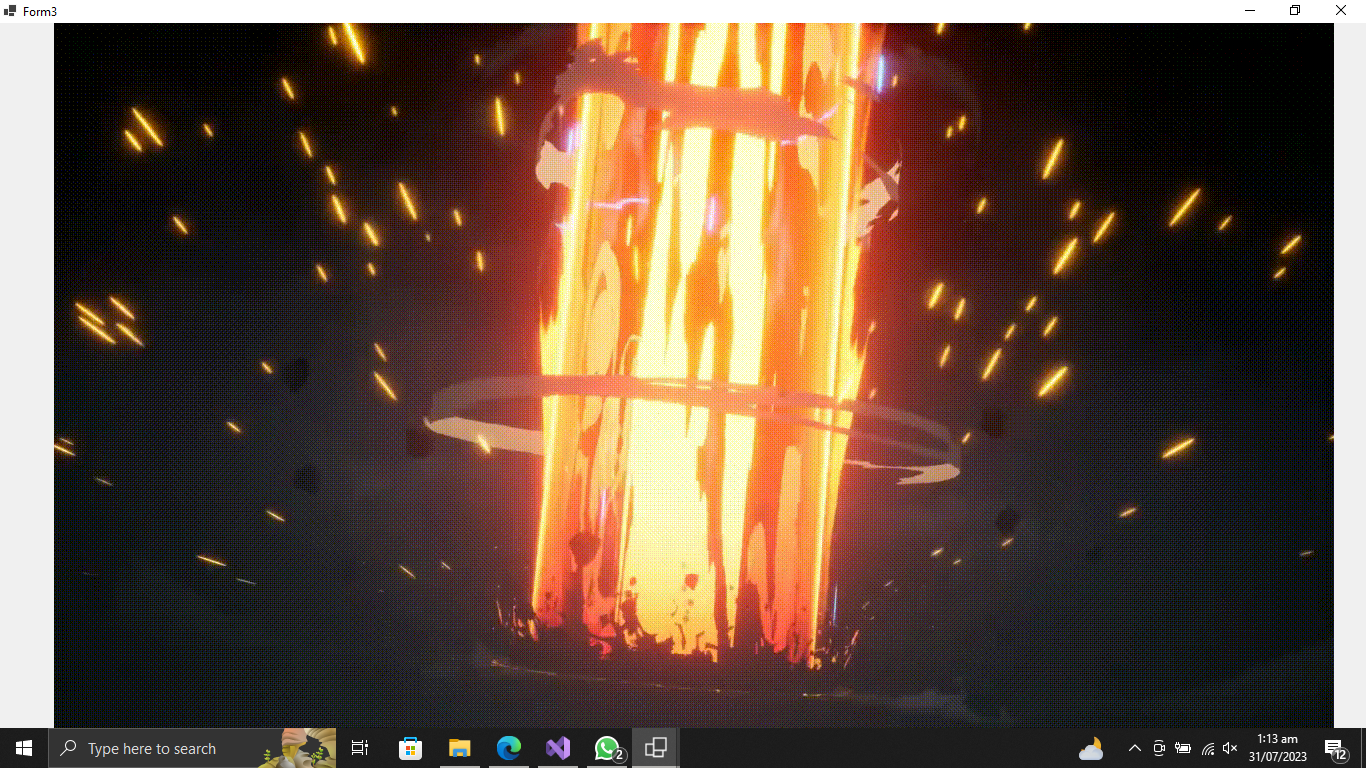
}

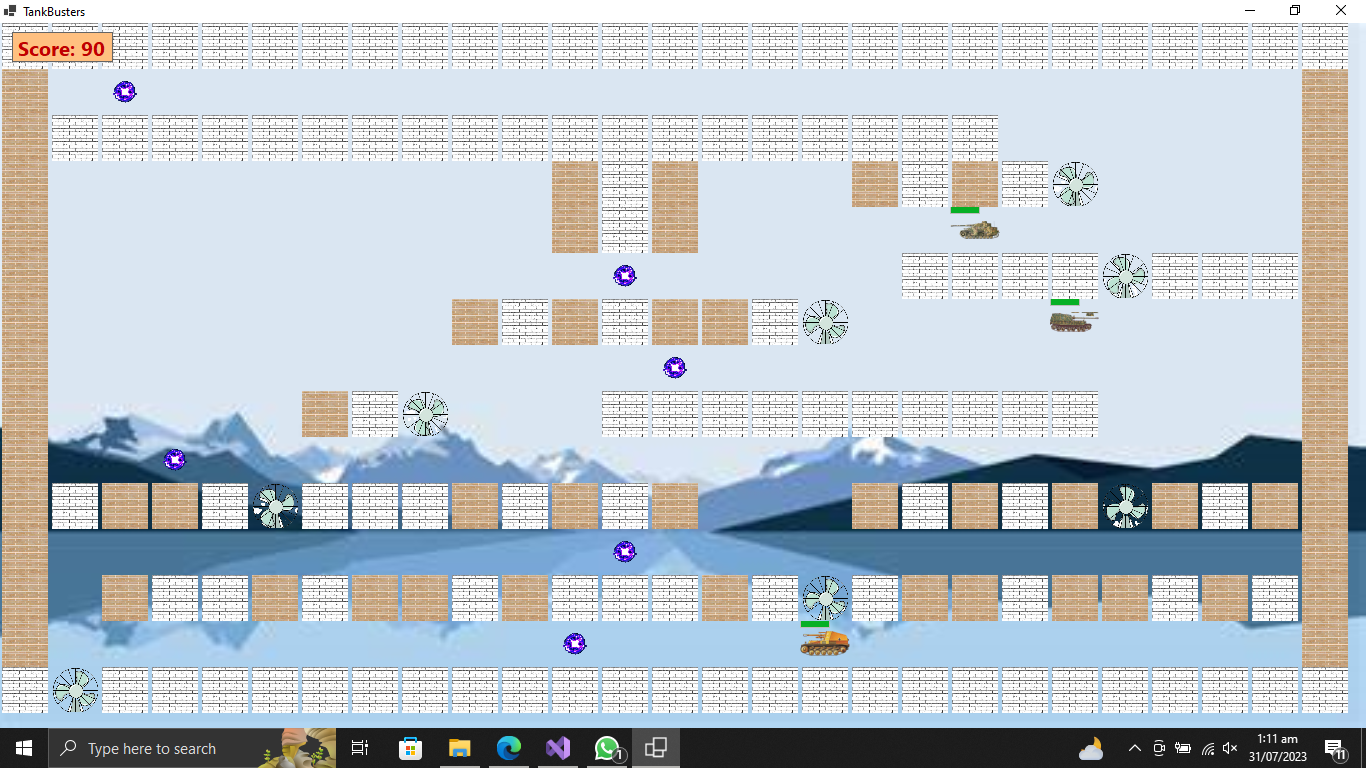
}

## Frameworks:

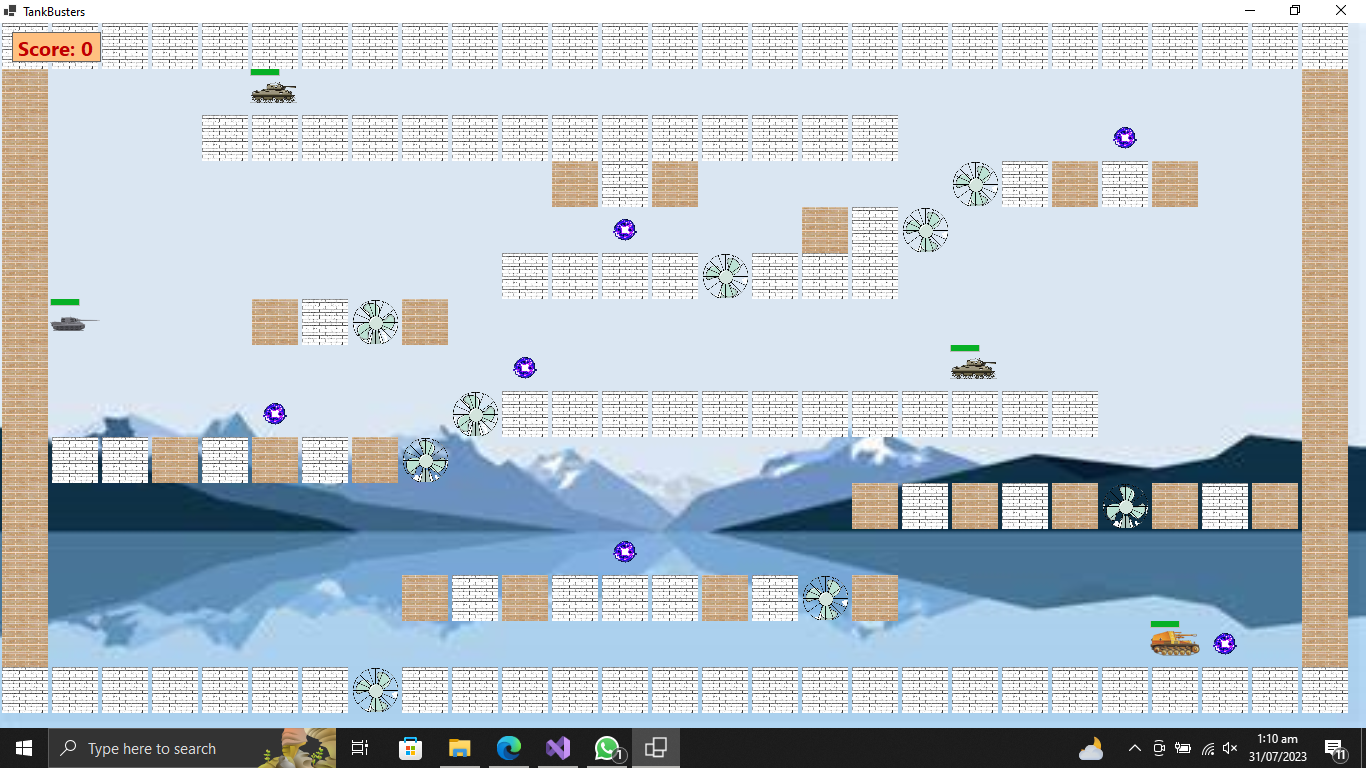
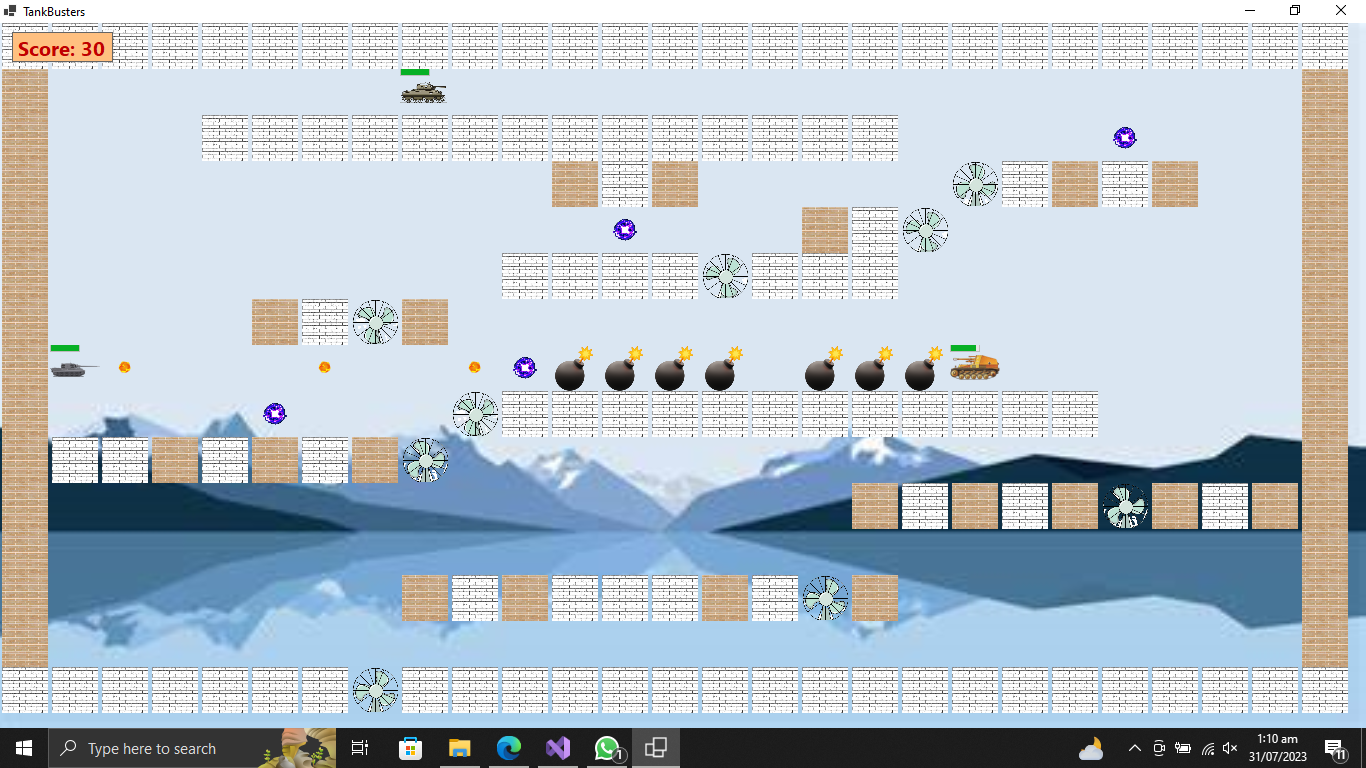


**Last Level:**

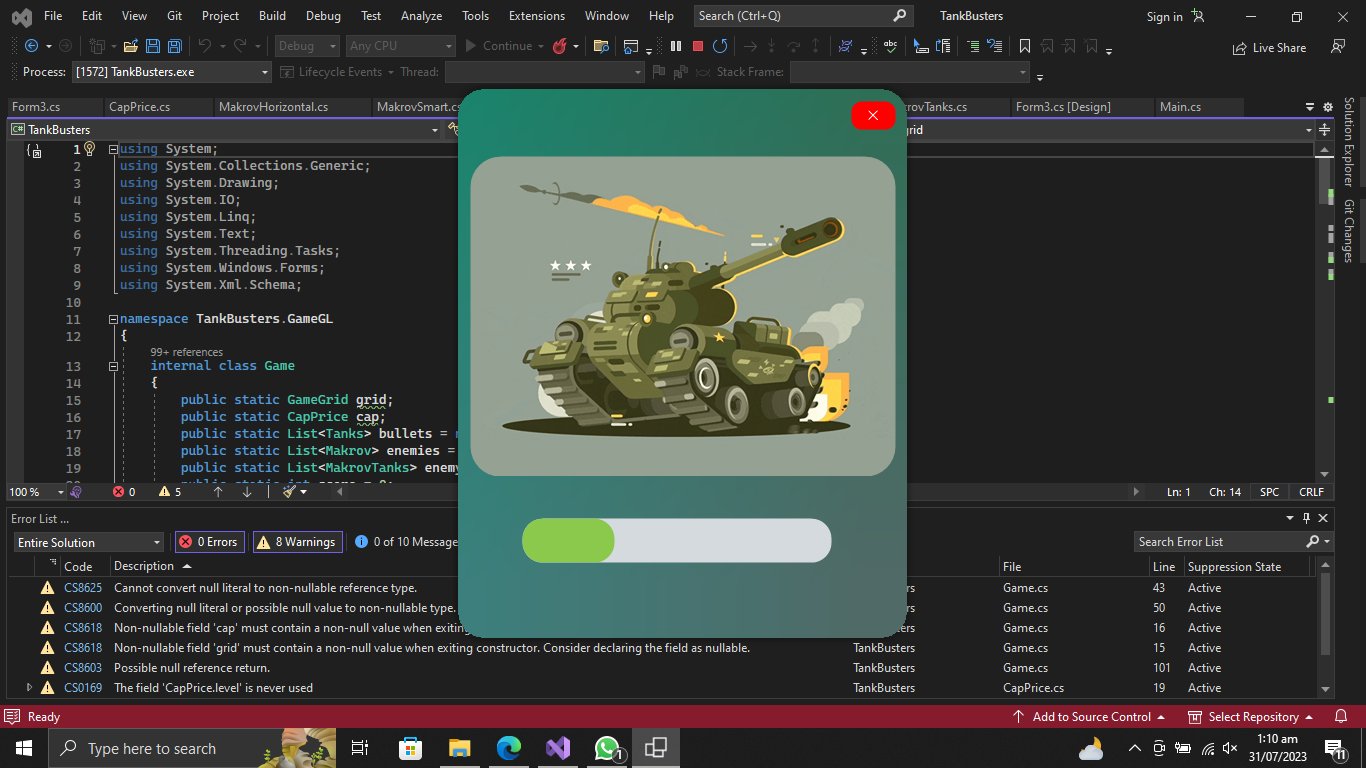


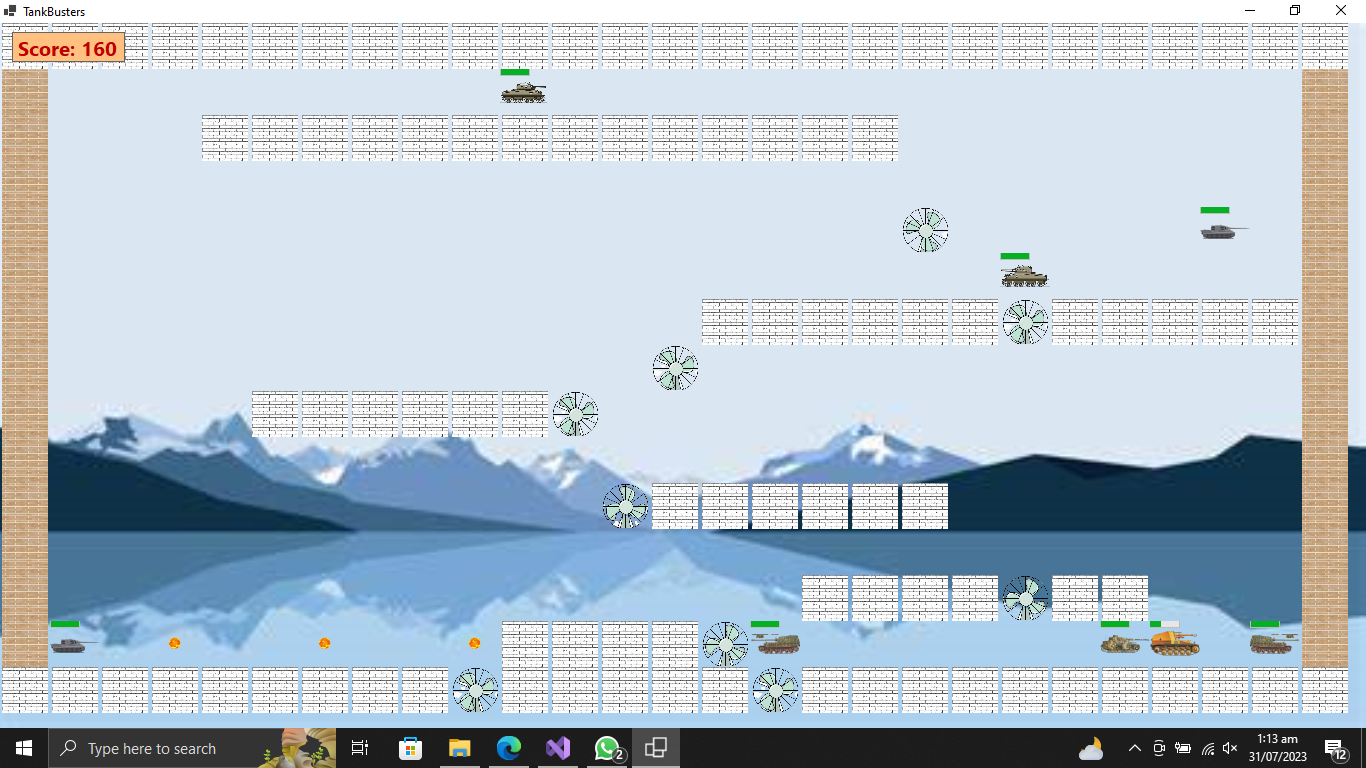
**Level 2:**

**Level 1:**

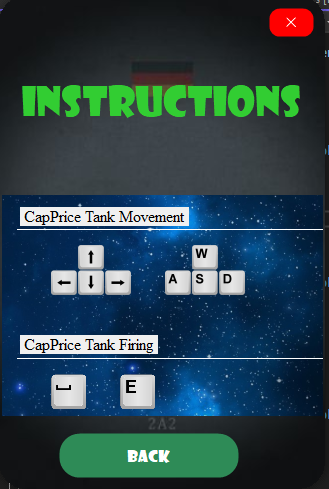


**Loading:**

 **Level 3:**



**Instructions:**

  
**Ending:**

