**Elemental Arena II**



Session: 2022 – 2026

**Submitted by:**

M.Abdullah Iftikhar 2022-CS-05

**Submitted To:**

Sir Laeeq Khan

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

Contents

[Introduction: 3](#_Toc141599948)

[OOP Concepts: 3](#_Toc141599949)

[Design Pattern Implementation: 3](#_Toc141599950)

[CRC Diagram: 4](#_Toc141599951)

[Class Details: 4](#_Toc141599952)

[Conclusion: 4](#_Toc141599953)

[Complete Code: 5](#_Toc141599954)

[**https://youtu.be/T7O63ECsEiU**](https://youtu.be/T7O63ECsEiU)

**Link of GameVideo**

## Introduction:

It starts with our main character named Death who live in the world of souls(NetherLand) Whose purpose is to prevent Dark Forces and lost souls from Reaching .our dimension across the multiverse. Death is Pure Evil himself,, All he wants is Power and dominance over everything,, but unlike other dark forces ,he doesn’t pray on weak, he always tries to find new challenges and opponents,,, death is also looking for special keys with are source of great power for him,,, he gets these keys by Killing waves of dark Forces,,SO whenever some dark Force tries to cross over Death Stops it,,and he will continue to do so till all the evil that remains in the multiverse is him ,

## OOP Concepts:

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 types.(ie, vertical,horizontal,smart and random).

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).

Bullet is a parent Class with different bullets as childclass(ie,,, chasing bullet, enemyBullet,PlayerBullet, all with different move mathods and attributes and some common attributes and mathods in bullet(ie,,, in ParentClass))

RandomObject Class that is used to generate RandomObjects in game,,like Pallets or the key that is generated at the end of each level

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.

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:

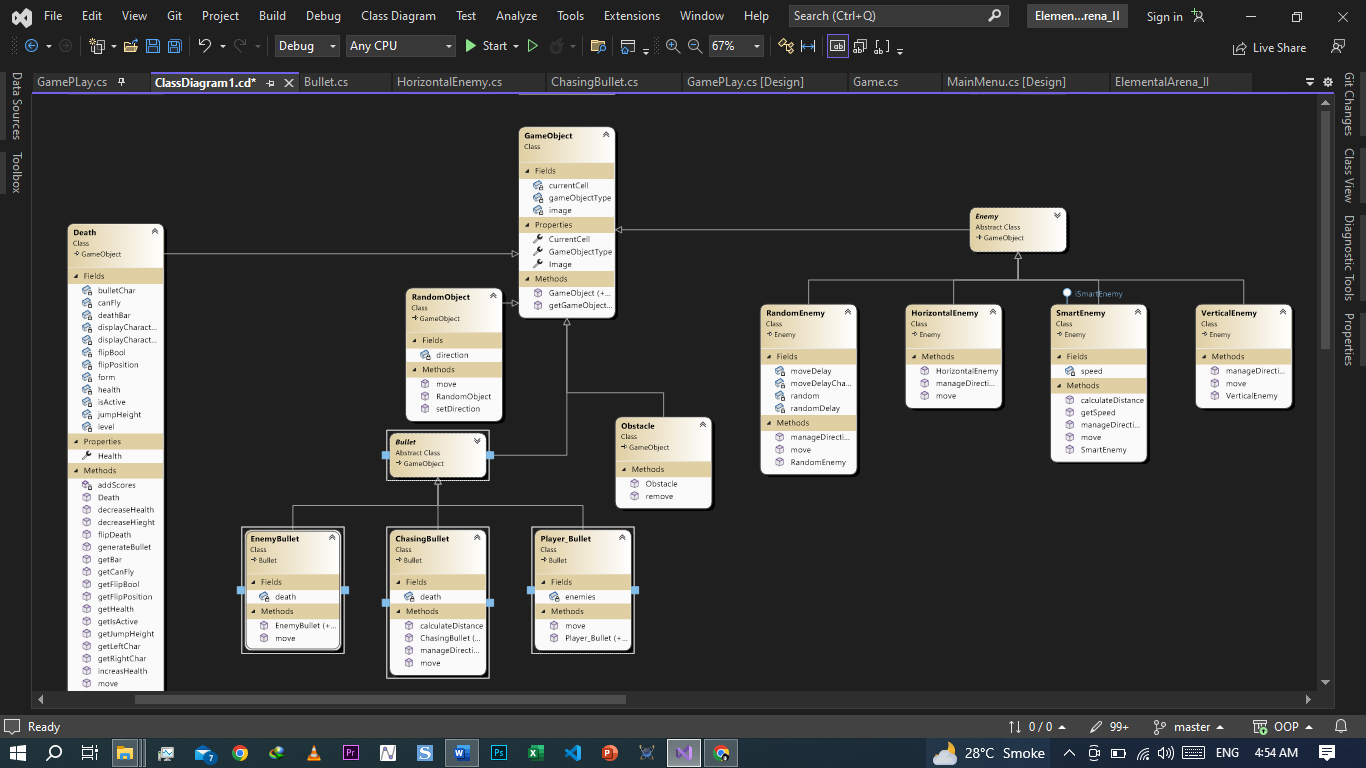
It has a class of game that returns the type of character and images depending upon their display character. It has a class of Gamecell which set the width and height of a cell at the of its creation and has a function that returns the next cell. It has a class of game object which determine its type 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. Bullet Class is also abstract and has three child classes, Chasing,Enemy and player bullet each having different move functionality,,

**Impotant Features**

**The code is written highly efficiently,, tho it can be made more efficient ,,,but as it is right now,,**

* **It manages all stages in a single form,,**
* **it takes 10 mins to add a new level,,(You just need to decide with enemy should be present in next stage,**
* **Main Characer Death can fly in some stages and can Jump in others depending on a bool attribute canFly,,,**
* **Enemy also can fire normal and chasing bullets depending on attribute bool ischasingBullet,,**
* **Any number of stages can be included with much ease**
* **Random reward generation(Bullets destroy reward)**
* **After killing all the enemies in a stage,, a key is generated at the end of it,,next level starts after taking the key,(bullets cant destroy key)**
* **Auto enemy and player health bar generation**
* **EnemyAnd Players flip there direction depending on there direction of movement**

## CRC Diagram:



## 

## Class Details:

GameObject class contains the information about the game object which a cell holds. GameCell contains information about a cell and set its height and width when created. Game class returns the type of character and its image based on the display character. GameGrid 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..Enemy 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.

Bullet Class is abstract,, which is a parent class and ChasingBullet, EnemyBullet & Player Bullets are child classes

## Conclusion:

This is a GUI based 2d game. It has different types of move functions and collision detection eward generation. It achieves many of its goals. It is still lacking in some places but still it is good. I learned so many new things while working on this project. It covers many of the c# concepts. I face some difficulties in starting while converting it into GUI but I was able to face them and make this project. It I quite an achievement for me.

## Complete Code:

**GameGL Classes:**

**GameGrid**

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;

if (type == GameObjectType.OBSTACLE)

{

loadObstacles(cell, displayIamge);

}

}

}

fp.Close();

}

void loadObstacles(GameCell cell, Image img)

{

Obstacle obs = new Obstacle(img, cell);

Game.obstacles.Add(obs);

}

}

**GameCell**

internal class GameCell

{

int row;

int col;

GameObject currentGameObject;

GameGrid grid;

PictureBox pictureBox;

const int width = 55;

const int height = 55;

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 GameGrid getGrid()

{

return grid;

}

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; }

}

**GameObject**

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 == '|' || displayCharacter == '%' || displayCharacter == '#')

{

return GameObjectType.WALL;

}

if (displayCharacter == '!')

{

return GameObjectType.V\_WALL;

}

if (displayCharacter == '&')

{

return GameObjectType.OBSTACLE;

}

if (displayCharacter == '.')

{

return GameObjectType.REWARD;

}

if (displayCharacter == 'F' || displayCharacter == 'f')

{

return GameObjectType.PLAYER;

}

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; }

}

**Game**

internal class Game

{

public static GameGrid grid;

public static Death death;

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

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

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

public static List<RandomObject> key = new List<RandomObject>();

public static int score = 0;

public static int lives = 5;

public static int enemyDamage = 1;

public static int playerDamage = 5;

public static bool keyBool = false;

public static bool levelClear = false;

public static int level;

public static int change;

private static int rewardDelay = 0;

public static List<Obstacle> obstacles = new List<Obstacle>();

public static void increaseScore(int scoreAdded)

{

score = score + scoreAdded;

}

public static void generateKey()

{

if(keyBool == true)

{

Image keyImg = GameGL.Game.getGameObjectImage('?');

GameCell keyCell = Game.grid.getCell(1, 14);

RandomObject key = new RandomObject(GameObjectType.KEY, keyImg, keyCell);

Game.key.Add(key);

keyBool = false;

}

}

public static void generateRandomReward()

{

rewardDelay++;

if(rewardDelay % 30 == 0)

{

Random rand = new Random();

int x = rand.Next(1, 11);

int y = rand.Next(1, 24);

Image rewardImg = GameGL.Game.getGameObjectImage('@');

GameCell rewardCell = Game.grid.getCell(x, y);

GameObject obj = new GameObject(GameObjectType.REWARD, rewardImg);

if (rewardCell.nextWallCell(GameDirection.Down).CurrentGameObject.GameObjectType == GameObjectType.WALL)

{

rewardCell.setGameObject(obj);

}

}

}

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 = ElementalArena\_II.Properties.Resources.blackHorizontal;

}

if (displayCharacter == '|')

{

img = ElementalArena\_II.Properties.Resources.blackVertical;

}

if (displayCharacter == '&')

{

img = ElementalArena\_II.Properties.Resources.Red\_Wall;

}

if (displayCharacter == 'd')

{

img = ElementalArena\_II.Properties.Resources.DeathLeft;

}

if (displayCharacter == 'D')

{

img = ElementalArena\_II.Properties.Resources.DeathRight;

}

if (displayCharacter == 'e')

{

img = ElementalArena\_II.Properties.Resources.DarkLeft;

}

if (displayCharacter == 'E')

{

img = ElementalArena\_II.Properties.Resources.DarkRight;

}

if (displayCharacter == 'c')

{

img = ElementalArena\_II.Properties.Resources.BlackDragonLeft;

}

if (displayCharacter == 'C')

{

img = ElementalArena\_II.Properties.Resources.BlackDragonRight;

}

//starting enemies

if (displayCharacter == 'w')

{

img = ElementalArena\_II.Properties.Resources.WildEnemyLeft;

}

if (displayCharacter == 'W')

{

img = ElementalArena\_II.Properties.Resources.WildEnemyRight;

}

if (displayCharacter == 'i')

{

img = ElementalArena\_II.Properties.Resources.IceEnemyLeft;

}

if (displayCharacter == 'I')

{

img = ElementalArena\_II.Properties.Resources.IceEnemyRight;

}

if (displayCharacter == 'f')

{

img = ElementalArena\_II.Properties.Resources.fireEnemyLeft;

}

if (displayCharacter == 'F')

{

img = ElementalArena\_II.Properties.Resources.fireEnemyRight;

}

//Dragons

if (displayCharacter == 'r')

{

img = ElementalArena\_II.Properties.Resources.RazorLeft;

}

if (displayCharacter == 'R')

{

img = ElementalArena\_II.Properties.Resources.RazorRight;

}

if (displayCharacter == 't')

{

img = ElementalArena\_II.Properties.Resources.DarkDragonLeft;

}

if (displayCharacter == 'T')

{

img = ElementalArena\_II.Properties.Resources.DarkDragonRight;

}

if (displayCharacter == 'y')

{

img = ElementalArena\_II.Properties.Resources.AngerDragonLeft;

}

if (displayCharacter == 'Y')

{

img = ElementalArena\_II.Properties.Resources.AngerDragonRight;

}

//Teir 3

if (displayCharacter == 'l')

{

img = ElementalArena\_II.Properties.Resources.LavaQueenLeft;

}

if (displayCharacter == 'L')

{

img = ElementalArena\_II.Properties.Resources.LavaQueenRight;

}

if (displayCharacter == 'k')

{

img = ElementalArena\_II.Properties.Resources.SamuraiLeft;

}

if (displayCharacter == 'K')

{

img = ElementalArena\_II.Properties.Resources.SamuraiRight;

}

if (displayCharacter == 'j')

{

img = ElementalArena\_II.Properties.Resources.ReaperLeft;

}

if (displayCharacter == 'J')

{

img = ElementalArena\_II.Properties.Resources.ReaperRight;

}

//tier 2

if (displayCharacter == 'h')

{

img = ElementalArena\_II.Properties.Resources.WarriorLeft;

}

if (displayCharacter == 'H')

{

img = ElementalArena\_II.Properties.Resources.WarriorRight;

}

if (displayCharacter == 'm')

{

img = ElementalArena\_II.Properties.Resources.FireSoldierLeft;

}

if (displayCharacter == 'M')

{

img = ElementalArena\_II.Properties.Resources.FireSoldierRight;

}

if (displayCharacter == 'n')

{

img = ElementalArena\_II.Properties.Resources.FireLordLeft;

}

if (displayCharacter == 'N')

{

img = ElementalArena\_II.Properties.Resources.FireLordRight;

}

//tier 1

if (displayCharacter == 'b')

{

img = ElementalArena\_II.Properties.Resources.SkeletonDragonLeft;

}

if (displayCharacter == 'B')

{

img = ElementalArena\_II.Properties.Resources.SkeletonDragonRight;

}

if (displayCharacter == 'z')

{

img = ElementalArena\_II.Properties.Resources.DarknessLeft;

}

if (displayCharacter == 'Z')

{

img = ElementalArena\_II.Properties.Resources.DarknessRight ;

}

//FireBalls

if (displayCharacter == '1')

{

img = ElementalArena\_II.Properties.Resources.SinisterBall;

}

if (displayCharacter == '2')

{

img = ElementalArena\_II.Properties.Resources.GreenBall;

}

if (displayCharacter == '3')

{

img = ElementalArena\_II.Properties.Resources.waterBall;

}

if (displayCharacter == '4')

{

img = ElementalArena\_II.Properties.Resources.fireBall;

}

if (displayCharacter == '5')

{

img = ElementalArena\_II.Properties.Resources.IceBall;

}

if (displayCharacter == '6')

{

img = ElementalArena\_II.Properties.Resources.electricFire;

}

if (displayCharacter == '7')

{

img = ElementalArena\_II.Properties.Resources.DarkFire;

}

if (displayCharacter == '@')

{

img = ElementalArena\_II.Properties.Resources.reward;

}

if (displayCharacter == '?')

{

img = ElementalArena\_II.Properties.Resources.key;

}

return img;

}

public static void generateHorizontalEnemy(Form form, bool isChasingBullet, char left, char right, char bulletChar, int health, Death death, GameDirection direction, char imageChar, int x, int y)

{

Image Img = GameGL.Game.getGameObjectImage(left);

GameCell enemyStartCell = Game.grid.getCell(x, y);

HorizontalEnemy enemy = new HorizontalEnemy(form, isChasingBullet, left, right, bulletChar, health, death, direction, Img, enemyStartCell);

Game.enemies.Add(enemy);

}

public static void generateVerticalEnemy(Form form, bool isChasingBullet, char left, char right, char bulletChar, int health, Death death, GameDirection direction, char imageChar, int x, int y)

{

Image Img = GameGL.Game.getGameObjectImage(left);

GameCell enemyStartCell = Game.grid.getCell(x, y);

VerticalEnemy enemy = new VerticalEnemy(form, isChasingBullet, left, right, bulletChar, health, death, direction, Img, enemyStartCell);

Game.enemies.Add(enemy);

}

public static void generateRandomEnemy(Form form, bool isChasingBullet, char left, char right, char bulletChar, int health, Death death, GameDirection direction, char imageChar, int x, int y)

{

Image Img = GameGL.Game.getGameObjectImage(left);

GameCell enemyStartCell = Game.grid.getCell(x, y);

RandomEnemy enemy = new RandomEnemy(form, isChasingBullet, left, right, bulletChar, health, death, direction, Img, enemyStartCell);

Game.enemies.Add(enemy);

}

public static void generateSmartEnemy(Form form, bool isChasingBullet, char left, char right, char bulletChar, int health, Death death, GameDirection direction, char imageChar, int x, int y)

{

Image Img = GameGL.Game.getGameObjectImage(left);

GameCell enemyStartCell = Game.grid.getCell(x, y);

SmartEnemy enemy = new SmartEnemy(form, isChasingBullet, left, right, bulletChar, health, death, direction, Img, enemyStartCell);

Game.enemies.Add(enemy);

}

}

**Death (Main Player)**

internal class Death : GameObject

{

bool isActive;

private int health;

private string flipPosition = "Right";

private bool flipBool = false;

private int jumpHeight = -1;

private char displayCharacterRight;

private char displayCharacterLeft;

private char bulletChar;

private bool canFly;

ProgressBar deathBar;

GamePLay form;

int level;

public Death(GamePLay form,char left,char right,char bulletChar,bool canFly, int health, Image image, GameCell startCell) : base(GameObjectType.PLAYER, image)

{

this.health = health;

this.CurrentCell = startCell;

this.form = form;

this.displayCharacterLeft = left;

this.displayCharacterRight = right;

this.bulletChar = bulletChar;

this.canFly = canFly;

deathBar = new ProgressBar();

deathBar = new ProgressBar();

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

deathBar.ForeColor = Color.Green;

deathBar.BackColor = Color.Black;

deathBar.Style = ProgressBarStyle.Continuous;

form.Controls.Add(deathBar);

}

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

public void setDisplayChars(char left, char right)

{

this.displayCharacterLeft = left;

this.displayCharacterRight = right;

}

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;

}

}

public void setCanFly(bool canFly)

{

this.canFly = canFly;

}

public bool getCanFly()

{

return canFly;

}

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 = Game.getGameObjectImage(displayCharacterLeft);

}

else if (flipPosition == "Right")

{

this.CurrentCell.PictureBox.Image = Game.getGameObjectImage(displayCharacterRight); ;

}

}

public ProgressBar getBar()

{

return deathBar;

}

public void setBarValue()

{

if(health >=0)

{

deathBar.Value = health;

}

}

public void setBarPosition()

{

deathBar.Top = this.CurrentCell.X \* 55;

deathBar.Left = this.CurrentCell.Y \* 55;

}

public void generateBullet()

{

Player\_Bullet b = new Player\_Bullet();

Image bullet = GameGL.Game.getGameObjectImage(bulletChar);

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 Player\_Bullet(Game.enemies,GameDirection.Right, bullet, startBullet);

b.setIsActive(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 Player\_Bullet(Game.enemies,GameDirection.Left, bullet, startBullet);

b.setIsActive(true);

Game.bullets.Add(b);

}

}

}

public char getRightChar()

{

return displayCharacterRight;

}

public char getLeftChar()

{

return displayCharacterLeft;

}

}

**Enemy**

abstract internal class Enemy : GameObject

{

protected ProgressBar enemyBar = new ProgressBar();

protected int health;

protected Death death;

protected string flipPosition = "Right";

protected bool flipBool = false;

Form form;

bool isActive;

protected char displayCharacterRight;

protected char displayCharacterLeft;

protected char bulletChar;

protected bool isChasingBullet;

protected GameDirection direction;

protected GameDirection faceDirection;

protected int bulletDelay = 1;

public Enemy(Form form,bool isChasingBullet,char left,char right,char bulletChar, int health, GameObjectType gameObjectType, Image image) : base(GameObjectType.ENEMY, image)

{

this.health = health;

this.displayCharacterLeft = left;

this.displayCharacterRight = right;

this.bulletChar = bulletChar;

this.form = form;

this.isChasingBullet = isChasingBullet;

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 \* 55;

enemyBar.Left = this.CurrentCell.Y \* 55;

}

void generateChasingBullet()

{

if (bulletDelay % 8 == 0)

{

Image bullet = GameGL.Game.getGameObjectImage(bulletChar);

GameCell startBullet = new GameCell();

startBullet = this.CurrentCell.nextCell(faceDirection);

Bullet b = new ChasingBullet();

b = new ChasingBullet(death, faceDirection, bullet, startBullet);

b.setIsActive(true);

Game.enemyBullets.Add(b);

}

bulletDelay++;

}

void generateNormalBullet()

{

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

{

if (bulletDelay == 2)

{

this.setFlipBool(true);

}

if (bulletDelay % 3 == 0)

{

this.setFlipBool(false);

EnemyBullet b = new EnemyBullet();

Image bullet = GameGL.Game.getGameObjectImage(bulletChar);

GameCell startBullet = new GameCell();

startBullet = this.CurrentCell.nextCell(faceDirection);

b = new EnemyBullet(death, faceDirection, bullet, startBullet);

b.setIsActive(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)

{

if (isChasingBullet == true)

{

generateChasingBullet();

}

else if (isChasingBullet == false)

{

generateNormalBullet();

}

}

}

public char getRightChar()

{

return displayCharacterRight;

}

public char getLeftChar()

{

return displayCharacterLeft;

}

public GameDirection getDirection()

{

return direction;

}

public void flipEnemy()

{

if (flipPosition == "Left")

{

this.CurrentCell.PictureBox.Image = Game.getGameObjectImage(displayCharacterLeft);

faceDirection = GameDirection.Left;

}

else if (flipPosition == "Right")

{

this.CurrentCell.PictureBox.Image = Game.getGameObjectImage(displayCharacterRight);

faceDirection = GameDirection.Right;

}

}

protected void enemyDirection()

{

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

{

flipPosition = "Left";

faceDirection = GameDirection.Left;

}

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

{

flipPosition = "Right";

faceDirection = GameDirection.Right;

}

}

}

interface iSmartEnemy

{

double calculateDistance(GameCell nextcell);

}

**HorizontalEnemy**

internal class HorizontalEnemy : Enemy

{

public HorizontalEnemy(Form form,bool isChasingBullet,char left,char right,char bulletChar, int lives, Death death, GameDirection direction, Image image, GameCell startCell) : base(form,isChasingBullet,left,right,bulletChar, lives, GameObjectType.ENEMY, image)

{

this.direction = direction;

this.CurrentCell = startCell;

this.death = death;

}

public override GameCell move()

{

if(this.getIsActive() == true)

{

if (death.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 (death.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";

}

}

}

**VerticalEnemy**

public VerticalEnemy(Form form, bool isChasingBullet, char left, char right, char bulletChar, int lives, Death death, GameDirection direction, Image image, GameCell startCell) : base(form,isChasingBullet, left, right, bulletChar, lives, GameObjectType.ENEMY, image)

{

this.direction = direction;

this.CurrentCell = startCell;

this.death = death;

}

public override GameCell move()

{

if (this.getIsActive() == true)

{

if (death.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;

}

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

{

manageDirections();

}

if (currentCell != nextCell)

{

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

{

currentCell.setGameObject(previousObject);

}

this.setFlipBool(true);

}

}

else if (death.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 = "Left";

base.enemyDirection();

}

}

}

**RandomEnemy**

internal class RandomEnemy: Enemy

{

int randomDelay;

int random;

int moveDelay = 11;

bool moveDelayChanger;

public RandomEnemy(Form form, bool isChasingBullet, char left, char right, char bulletChar, int lives, Death death, GameDirection direction, Image image, GameCell startCell) : base(form,isChasingBullet, left, right, bulletChar, lives, GameObjectType.ENEMY, image)

{

this.direction = direction;

this.CurrentCell = startCell;

this.death = death;

}

public override GameCell move()

{

if (this.getIsActive() == true)

{

if (death.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++;

}

}

**SmartEnemy**

internal class SmartEnemy : Enemy ,iSmartEnemy

{

int speed;

public SmartEnemy(Form form, bool isChasingBullet, char left,char right,Char bulletChar, int lives, Death death, GameDirection direction, Image image, GameCell startCell) : base(form,isChasingBullet, left,right,bulletChar, lives, GameObjectType.ENEMY, image)

{

this.direction = direction;

this.CurrentCell = startCell;

this.death = death;

this.speed = 1;

}

public int getSpeed()

{

return speed;

}

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 (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((death.CurrentCell.X - nextcell.X), 2) + Math.Pow((death.CurrentCell.Y - nextcell.Y), 2));

}

**Bullet**

abstract internal class Bullet : GameObject

{

protected bool isActive = false;

protected GameDirection direction;

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

{

}

public GameDirection getDirection()

{

return direction;

}

public Bullet()

{

}

public abstract GameCell move();

public void setIsActive(bool set)

{

this.isActive = set;

}

public bool getIsActive()

{

return isActive;

}

}

**PlayerBullet**

internal class Player\_Bullet : Bullet

{

private List<Enemy> enemies;

public Player\_Bullet(List<Enemy> enemies,GameDirection direction, Image image, GameCell startCell) : base(GameObjectType.PLAYER\_BULLET, image)

{

this.direction = direction;

this.CurrentCell = startCell;

this.enemies = enemies;

}

public Player\_Bullet()

{

}

public override GameCell move()

{

if (getIsActive() == true)

{

GameCell currentCell = this.CurrentCell;

GameCell nextCell = currentCell.nextCell(direction);

GameCell nextCell2 = currentCell.nextWallCell(direction);

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

{

this.CurrentCell = nextCell;

}

GameObject previousObject = nextCell.CurrentGameObject;

GameObject nextObject = nextCell2.CurrentGameObject;

if (currentCell != nextCell)

{

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

{

currentCell.setGameObject(Game.getBlankGameObject());

}

}

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

{

if (nextObject.GameObjectType == GameObjectType.ENEMY)

{

Game.increaseScore(1);

foreach (Enemy 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.PLAYER\_BULLET)

{

enemy.decreaseHealth(Game.playerDamage);

}

if (obj2.GameObjectType == GameObjectType.PLAYER\_BULLET)

{

enemy.decreaseHealth(Game.playerDamage);

}

}

}

currentCell.setGameObject(Game.getBlankGameObject());

this.setIsActive(false);

}

return nextCell;

}

return null;

}

}

**EnemyBullet**

internal class EnemyBullet : Bullet

{

Death death;

public EnemyBullet(Death death, GameDirection direction, Image image, GameCell startCell) : base(GameObjectType.ENEMY\_BULLET, image)

{

this.direction = direction;

this.CurrentCell = startCell;

this.death = death;

}

public EnemyBullet()

{

}

public override GameCell move()

{

if (getIsActive() == true)

{

GameCell currentCell = this.CurrentCell;

GameCell nextCell = currentCell.nextCell(direction);

GameCell nextCell2 = currentCell.nextWallCell(direction);

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

{

this.CurrentCell = nextCell;

}

GameObject previousObject = nextCell.CurrentGameObject;

GameObject nextObject = nextCell2.CurrentGameObject;

if (currentCell != nextCell)

{

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

{

currentCell.setGameObject(Game.getBlankGameObject());

}

}

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

{

if (nextObject.GameObjectType == GameObjectType.PLAYER)

{

death.decreaseHealth(Game.enemyDamage);

}

currentCell.setGameObject(Game.getBlankGameObject());

this.setIsActive(false);

}

return nextCell;

}

return null;

}

}

**ChasingBullet**

internal class ChasingBullet : Bullet

{

Death death;

public ChasingBullet(Death death, GameDirection direction, Image image, GameCell startCell) : base(GameObjectType.ENEMY\_BULLET, image)

{

this.direction = direction;

this.CurrentCell = startCell;

this.death = death;

}

public ChasingBullet()

{

}

public override GameCell move()

{

if (getIsActive() == true)

{

manageDirections();

GameCell currentCell = this.CurrentCell;

GameCell nextCell = currentCell.nextCell(direction);

GameCell nextCell2 = currentCell.nextWallCell(direction);

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

{

this.CurrentCell = nextCell;

}

GameObject previousObject = nextCell.CurrentGameObject;

GameObject nextObject = nextCell2.CurrentGameObject;

if (currentCell != nextCell)

{

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

{

currentCell.setGameObject(Game.getBlankGameObject());

}

}

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

{

if (nextObject.GameObjectType == GameObjectType.PLAYER)

{

death.decreaseHealth(Game.enemyDamage);

}

currentCell.setGameObject(Game.getBlankGameObject());

this.setIsActive(false);

}

return nextCell;

}

return null;

}

public void manageDirections()

{

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

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

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

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

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

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

{

this.direction = GameDirection.Left;

}

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

{

this.direction = 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((death.CurrentCell.X - nextcell.X), 2) + Math.Pow((death.CurrentCell.Y - nextcell.Y), 2));

}

}

**GameDirection**

public enum GameDirection

{

Left,

Right,

Up,

Down

}

**GameObjectType**

public enum GameObjectType

{

WALL,

PLAYER,

ENEMY,

REWARD,

ENEMY\_BULLET,

PLAYER\_BULLET,

V\_WALL,

OBSTACLE,

KEY,

NONE

}

**RandomObject**

internal class RandomObject : GameObject

{

GameDirection direction;

public RandomObject(GameObjectType type, Image image, GameCell startCell) : base(type, image)

{

this.CurrentCell = startCell;

}

public GameCell move()

{

GameCell currentCell = this.CurrentCell;

GameCell nextCell = currentCell.nextCell(direction);

GameCell nextCell2 = currentCell.nextWallCell(direction);

GameObject previousObject = nextCell.CurrentGameObject;

if(nextCell2.CurrentGameObject.GameObjectType != GameObjectType.WALL)

{

this.CurrentCell = nextCell;

}

if (currentCell != nextCell)

{

currentCell.setGameObject(Game.getBlankGameObject());

}

return null;

}

public void setDirection(GameDirection direction)

{

this.direction = direction;

}

}

**Obstacles**

internal class Obstacle : GameObject

{

public Obstacle(Image image, GameCell startCell) : base(GameObjectType.OBSTACLE, image)

{

this.CurrentCell = startCell;

}

public void remove()

{

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

this.CurrentCell.setGameObject(blankGameObject);

}

}