**SPACE WAR**



Session 2022-2026

**Submitted By:**

Mohsin Ali 2022-CS-193

**Supervised By:**

Prof. Dr. Awais Hassan

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

**Table of Contents**

[**STORY OF GAME:** 2](#_Toc141615333)

[**GAME CHARACTERS DESCRIPTION:** 3](#_Toc141615334)

[**CLASS DIAGRAM** 3](#_Toc141615335)

[**WIREFRAMES** 4](#_Toc141615336)

[**RULES AND INSTRUCTIONS** 7](#_Toc141615337)

[**COMPLETE CODE** 7](#_Toc141615338)

[**Program.cs** 7](#_Toc141615339)

[**Form1.cs** 7](#_Toc141615340)

[**Game.cs** 12](#_Toc141615341)

[**Start.cs** 14](#_Toc141615342)

[**Bullets.cs** 14](#_Toc141615343)

[**CollisionDetection.cs** 15](#_Toc141615344)

[**Enemy.cs** 16](#_Toc141615345)

[**GameCell.cs** 16](#_Toc141615346)

[**GameDirection.cs** 18](#_Toc141615347)

[**GameGrid.cs** 18](#_Toc141615348)

[**GameObject.cs** 19](#_Toc141615349)

[**GameObjectType.cs** 20](#_Toc141615350)

[**GameObject.cs** 20](#_Toc141615351)

[**HorizontalEnemy.cs** 21](#_Toc141615352)

[**RandomGhost.cs** 22](#_Toc141615353)

[**SmartEnemy.cs** 23](#_Toc141615354)

[**VerticalGhost.cs** 25](#_Toc141615355)

# **VIDEO LINK:**

<https://youtu.be/M4f1uvxICPM>

<https://youtube.com/@MohsinAli-sq1mo>

# **STORY OF GAME:**

Jack was exploring space in his spaceship. He was on a mission. Suddenly he got attacked by so many enemy ships named as Yutani, Tricky, Fresh, Spike and many more. They had advance space ships and are too dangerous. They just want to kill Jack. So Jack started to fight against them in order to save his life. These enemies could move horizontally, vertically, random and one of them keep on chasing Jack. So Jack keep on fighting these enemies to save his life.

# **GAME CHARACTERS DESCRIPTION:**

o **Jack :** Jack is the main player of the game. He can fire bullets and can move in any direction.

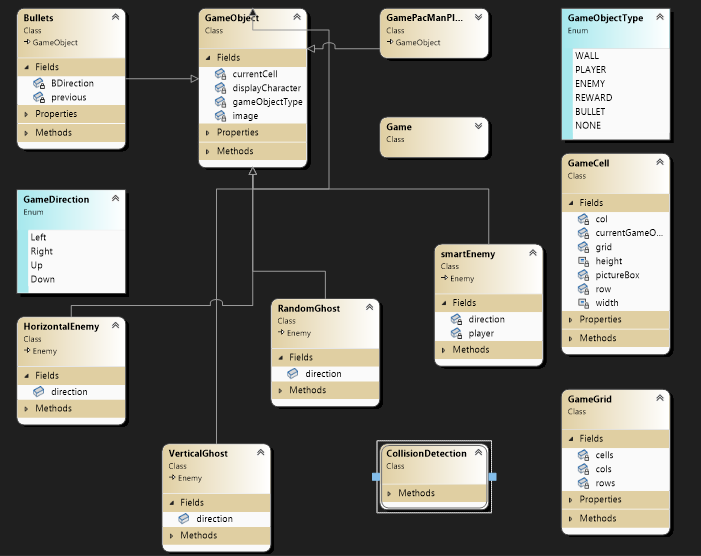
o **Yuatni :** Yutani is enemy who moves horizontally and can fire bullets.

o **Tricky :** Tricky is enemy who moves vertically and can fire bullets.

o **Fresh :** Fresh is enemy who moves randomly in any direction and can fire bullets.

o **Spike :** Spike is enemy who chases Jack according to his movements and can also fire bullets.

# **CLASS DIAGRAM**



# **WIREFRAMES**









# **RULES AND INSTRUCTIONS**

Following are the rules of the game :

* Game consists of only one level.
* Player Jack has only one life, if it ends than game is over.
* Enemies also have only one life.
* Player has to kill all the enemies in order to win the game.
* If player’s bullet collide with enemies, score increases and enemies health decreases.
* If enemy’s bullet collide with player, plyer’s health decrease.
* If player dies, game is over.

# **COMPLETE CODE**

## **Program.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace PacManGUI

{

static class Program //////////////////// Program.cs

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

//Application.Run(new Form1());

Application.Run(new start());

}

}

}

## **Form1.cs**

public partial class Form1 : Form //////////////////// Form1.cs

{

GameGrid grid;

int score = 0;

GamePacManPlayer pacman;

Enemy horizontalGhost;

Enemy verticalGhost1;

Enemy verticalGhost3;

Enemy randomGhost;

Enemy smartEnemy;

int smartspeed = 0;

int enemyBulletDelay = 0;

List<Enemy> ghosts = new List<Enemy>();

List<Bullets> bullets = new List<Bullets>();

List<Bullets> enemyBullets = new List<Bullets>();

public Form1()

{

InitializeComponent();

}

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

{

grid = new GameGrid("newMaze.txt", 17, 31);

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

Image horizontalGhostImage = Game.getGameObjectImage('H');

Image verticalGhostImage1 = Game.getGameObjectImage('V');

Image verticalGhostImage2 = Game.getGameObjectImage('v');

Image randomGhostImage = Game.getGameObjectImage('R');

Image smartEnemyImage = Game.getGameObjectImage('S');

GameCell startCell = grid.getCell(13, 26);

GameCell HghostStartCell = grid.getCell(1, 9);

GameCell VghostStartCell1 = grid.getCell(10, 1);

GameCell VghostStartCell3 = grid.getCell(1, 29);

GameCell RghostStartCell = grid.getCell(6, 16);

GameCell SEnemyStartCell = grid.getCell(3,3);

pacman = new GamePacManPlayer(pacManImage, startCell);

horizontalGhost = new HorizontalEnemy(horizontalGhostImage, HghostStartCell);

verticalGhost1 = new VerticalGhost(verticalGhostImage2, VghostStartCell1);

verticalGhost3 = new VerticalGhost(verticalGhostImage1, VghostStartCell3);

randomGhost = new RandomGhost(randomGhostImage, RghostStartCell);

smartEnemy = new smartEnemy(smartEnemyImage, SEnemyStartCell,pacman);

ghosts.Add(horizontalGhost);

ghosts.Add(verticalGhost1);

ghosts.Add(verticalGhost3);

ghosts.Add(randomGhost);

printMaze(grid);

gameEnd();

}

void gameEnd()

{

if(FlapHealth.Value <= 0)

{

gameLoop.Enabled = false;

this.Close();

gameEnd end = new gameEnd();

end.ShowDialog();

this.Show();

}

}

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);

// printCell(cell);

}

}

}

static void printCell(GameCell cell)

{

Console.SetCursorPosition(cell.Y, cell.X);

Console.Write(cell.CurrentGameObject.DisplayCharacter);

}

void generatePlayerBullet()

{

Image bImage = Game.getGameObjectImage('b');

GameCell bCell = grid.getCell(pacman.CurrentCell.X - 1, pacman.CurrentCell.Y);

Bullets bullet = new Bullets(GameObjectType.BULLET, bImage, bCell, GameDirection.Up);

bullets.Add(bullet);

}

void generateVertical1Bullets()

{

Image bImage = Game.getGameObjectImage('B');

GameCell bCell = grid.getCell(verticalGhost1.CurrentCell.X, verticalGhost1.CurrentCell.Y + 1);

Bullets bullet = new Bullets(GameObjectType.BULLET, bImage, bCell, GameDirection.Right);

enemyBullets.Add(bullet);

}

void generatehHorizontalBullets()

{

Image bImage = Game.getGameObjectImage('!');

GameCell bCell = grid.getCell(horizontalGhost.CurrentCell.X + 1, horizontalGhost.CurrentCell.Y );

Bullets bullet = new Bullets(GameObjectType.BULLET, bImage, bCell, GameDirection.Down);

enemyBullets.Add(bullet);

}

void generateVertical3Bullets()

{

Image bImage = Game.getGameObjectImage('B');

GameCell bCell = grid.getCell(verticalGhost3.CurrentCell.X, verticalGhost3.CurrentCell.Y - 1);

Bullets bullet = new Bullets(GameObjectType.BULLET, bImage, bCell, GameDirection.Left);

enemyBullets.Add(bullet);

}

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

{

if(Keyboard.IsKeyPressed(Key.LeftArrow)) {

GameCell cell = pacman.move(GameDirection.Left);

PacmanCollision(cell);

}

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

GameCell cell = pacman.move(GameDirection.Right);

PacmanCollision(cell);

}

if (Keyboard.IsKeyPressed(Key.UpArrow)){

GameCell cell = pacman.move(GameDirection.Up);

PacmanCollision (cell);

}

if (Keyboard.IsKeyPressed(Key.DownArrow)){

GameCell cell = pacman.move(GameDirection.Down);

PacmanCollision(cell);

}

if (Keyboard.IsKeyPressed(Key.Space))

{

generatePlayerBullet();

}

foreach (var g in ghosts)

{

GameCell cell = g.Move();

if(CollisionDetection.PlayerDetection(cell))

{

DecreasePlayerHealth(10);

}

}

if(smartspeed == 5)

{

GameCell cell = smartEnemy.Move();

if (CollisionDetection.PlayerDetection(cell))

{

DecreasePlayerHealth(10);

}

smartspeed = 0;

}

if(enemyBulletDelay == 5)

{

generateVertical1Bullets();

generateVertical3Bullets();

generatehHorizontalBullets();

enemyBulletDelay = 0;

}

enemyBulletDelay++;

smartspeed++;

MovePlayerBullets();

MoveEnemyBullets();

gameEnd();

}

private void IncreseScore()

{

score++;

lblScore.Text = "Score" + " " + score;

}

private void PacmanCollision(GameCell cell)

{

if(CollisionDetection.EnemyDetection(cell))

{

DecreasePlayerHealth(10);

}

}

private void StopGameLoop()

{

gameLoop.Stop();

lblGameOver.Visible = true;

}

private void MovePlayerBullets()

{

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

{

GameCell cell = bullets[i].Move();

if(CollisionDetection.EnemyDetection(cell))

{

IncreseScore();

bullets.Remove(bullets[i]);

}

else if(CollisionDetection.WallDetection(cell))

{

bullets.Remove(bullets[i]);

}

}

}

private void MoveEnemyBullets()

{

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

{

GameCell cell = enemyBullets[i].Move();

if(CollisionDetection.PlayerDetection(cell))

{

DecreasePlayerHealth(5);

enemyBullets.Remove(enemyBullets[i]);

}

else if(CollisionDetection.WallDetection(cell))

{

enemyBullets.Remove(enemyBullets[i]);

}

}

}

private void DecreasePlayerHealth(int value)

{

if(FlapHealth.Value>0)

{

FlapHealth.Value -= value;

}

}

//private void GameOver()

//{

// if(FlapHealth.Value <= 0)

// {

// gameLoop.Enabled = false;

// lblGameOver.Visible = true;

// }

//}

//private void GameOver()

//{

// //this.Hide();

// //Form gameOver = new GameOver();

// //gameOver.ShowDialog();

//}

}

## **Game.cs**

public class Game // Game.cs

{

public static GameObject getBlankGameObject(){

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

return blankGameObject;

}

public static GameObject getPlayerGameObject()

{

GameObject blankGameObject = new GameObject(GameObjectType.PLAYER, PacManGUI.Properties.Resources.Flap);

return blankGameObject;

}

public static GameObject getEnemyGameObject()

{

GameObject blankGameObject = new GameObject(GameObjectType.ENEMY, PacManGUI.Properties.Resources.verticalEnemy);

return blankGameObject;

}

public static GameObject getRewardGameObject()

{

GameObject blankGameObject = new GameObject(GameObjectType.REWARD, PacManGUI.Properties.Resources.pallet);

return blankGameObject;

}

public static Image getGameObjectImage(char displayCharacter)

{

Image img = null;

if (displayCharacter == '%')

{

img = PacManGUI.Properties.Resources.blocks1;

}

if(displayCharacter == '|')

{

img = PacManGUI.Properties.Resources.block4;

}

if (displayCharacter == '+')

{

img = PacManGUI.Properties.Resources.block2;

}

if (displayCharacter == '^')

{

img = PacManGUI.Properties.Resources.\_20;

}

if (displayCharacter == '-')

{

img = PacManGUI.Properties.Resources.block3;

}

if (displayCharacter == '\_')

{

img = PacManGUI.Properties.Resources.\_32;

}

if (displayCharacter == '!')

{

img = PacManGUI.Properties.Resources.laserRed13;

}

if (displayCharacter == '#')

{

img = PacManGUI.Properties.Resources.sideWall;

}

if (displayCharacter == 'b')

{

img = PacManGUI.Properties.Resources.playerLaser;

}

if (displayCharacter == 'B')

{

img = PacManGUI.Properties.Resources.enemyBullet;

}

if (displayCharacter == '.')

{

img = PacManGUI.Properties.Resources.pallet;

}

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

img = PacManGUI.Properties.Resources.playerShip;

}

if(displayCharacter == 'H')

{

img = PacManGUI.Properties.Resources.horizontalEnmy;

}

if (displayCharacter == 'V')

{

img = PacManGUI.Properties.Resources.enemyVerticalRight;

}

if (displayCharacter == 'v')

{

img = PacManGUI.Properties.Resources.enemyVerticalLeft;

}

if (displayCharacter == 'R')

{

img = PacManGUI.Properties.Resources.random;

}

if(displayCharacter == 'S')

{

img = PacManGUI.Properties.Resources.samrt;

}

return img;

}

}

## **Start.cs**

public partial class start : Form ///////////// start.cs

{

public start()

{

InitializeComponent();

}

private void startgaem\_Click(object sender, EventArgs e)

{

this.Hide();

Form1 form = new Form1();

form.ShowDialog();

this.Show();

}

}

## **Bullets.cs**

internal class Bullets : GameObject //////////// Bullets.cs

{

GameDirection BDirection;

GameObject previous = Game.getBlankGameObject();

public Bullets(GameObjectType type, Image image, GameCell cell, GameDirection direction) : base(type, image)

{

this.CurrentCell = cell;

BDirection = direction;

}

public GameDirection Direction { get { return BDirection; } }

public GameCell Move()

{

GameCell currentCell = this.CurrentCell;

GameCell nextCell = currentCell.nextCell(BDirection);

GameCell nextCopy = new GameCell(nextCell.X, nextCell.Y, nextCell.gameGrid);

nextCopy.setGameObject(nextCell.CurrentGameObject);

GameObject nextObject = nextCell.CurrentGameObject;

this.CurrentCell = nextCell;

if (nextCell != null && currentCell != nextCell)

{

if (previous.GameObjectType == GameObjectType.REWARD)

{

currentCell.setGameObject(Game.getRewardGameObject());

}

else if (previous.GameObjectType == GameObjectType.NONE)

{

currentCell.setGameObject(Game.getBlankGameObject());

}

previous = nextObject;

}

else

{

currentCell.setGameObject(Game.getBlankGameObject());

}

return nextCopy;

}

}

## **CollisionDetection.cs**

internal class CollisionDetection /////////// collisionDetection.cs

{

public static bool EnemyDetection(GameCell cell)

{

if (cell.CurrentGameObject.GameObjectType == GameObjectType.ENEMY)

{

return true;

}

return false;

}

public static bool WallDetection(GameCell cell)

{

if (cell.CurrentGameObject.GameObjectType == GameObjectType.WALL)

{

return true;

}

return false;

}

public static bool PlayerDetection(GameCell cell)

{

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

{

return true;

}

return false;

}

}

## **Enemy.cs**

internal abstract class Enemy : GameObject //////// Enemy.cs

{

protected float originalSpeed = new float();

protected float speed;

public abstract GameCell Move();

public GameObject previous = Game.getBlankGameObject();

public Enemy(Image image,GameCell startCell) :base(GameObjectType.ENEMY,image)

{

this.CurrentCell = startCell;

}

public Enemy(Image image, float speed) : base(GameObjectType.ENEMY, image)

{

this.speed = speed;

}

public float OriginalSpeed { get => originalSpeed; set => originalSpeed = value; }

public float Speed { get => speed; set => speed = value; }

}

## **GameCell.cs**

public class GameCell ////////////// GameCell.cs

{

int row;

int col;

GameObject currentGameObject;

GameGrid grid;

PictureBox pictureBox;

const int width = 40;

const int height = 40;

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 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(this.row, this.col-1);

if (ncell.CurrentGameObject.GameObjectType != GameObjectType.WALL) {

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.WALL)

{

return ncell;

}

}

}

if (direction == GameDirection.Up)

{

if (this.row > 0)

{

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

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

{

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.WALL)

{

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; }

public GameGrid gameGrid { get => grid; set => grid = value; }

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

}

## **GameDirection.cs**

public enum GameDirection //////////// GameDirection

{

Left,

Right,

Up,

Down

}

## **GameGrid.cs**

public class GameGrid ////////// GameGrid.cs

{

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 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();

}

}

## **GameObject.cs**

public class GameObject ///////// GameObject.cs

{

char displayCharacter;

GameObjectType gameObjectType;

GameCell currentCell;

Image image;

public GameObject(GameObjectType type, Image image)

{

this.gameObjectType = type;

this.Image = image;

}

public GameObject(GameObjectType type, char displayCharacter)

{

this.gameObjectType = type;

this.displayCharacter = displayCharacter;

}

public static GameObjectType getGameObjectType(char displayCharacter)

{

if (displayCharacter == '|' || displayCharacter == '%' || displayCharacter == '#' || displayCharacter == '+' || displayCharacter == '^' || displayCharacter == '-') {

return GameObjectType.WALL;

}

if (displayCharacter == '.') {

return GameObjectType.REWARD;

}

return GameObjectType.NONE;

}

public char DisplayCharacter { get => displayCharacter; set => displayCharacter = value; }

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; }

}

## **GameObjectType.cs**

public enum GameObjectType //////////// GameObjectType

{

WALL,

PLAYER,

ENEMY,

REWARD,

BULLET,

NONE

}

## **GameObject.cs**

class GamePacManPlayer : GameObject ////// GamePacManPlayer

{

public GamePacManPlayer(Image image,GameCell startCell):base (GameObjectType.PLAYER,image)

{

this.CurrentCell = startCell;

}

public GameCell move(GameDirection direction)

{

GameCell currentCell = this.CurrentCell;

GameCell nextCell= currentCell.nextCell(direction);

GameCell temp = new GameCell(nextCell.X,nextCell.Y,nextCell.gameGrid);

temp.setGameObject(nextCell.CurrentGameObject);

this.CurrentCell = nextCell;

if (currentCell != nextCell)

{

currentCell.setGameObject(Game.getBlankGameObject());

}

return temp;

}

}

## **HorizontalEnemy.cs**

internal class HorizontalEnemy : Enemy ///// HorizantalEnemy

{

public GameDirection direction = GameDirection.Right;

public HorizontalEnemy(Image image, GameCell startCell) : base(image,startCell)

{

}

public override GameCell Move()

{

GameCell currentCell = this.CurrentCell;

GameCell nextCell = currentCell.nextCell(direction);

GameCell temp = new GameCell(nextCell.X,nextCell.Y,nextCell.gameGrid);

temp.setGameObject(nextCell.CurrentGameObject);

GameObject nextObject = nextCell.CurrentGameObject;

this.CurrentCell = nextCell;

if (currentCell != nextCell)

{

if(previous.GameObjectType == GameObjectType.REWARD)

{

currentCell.setGameObject(Game.getRewardGameObject());

}

else

{

currentCell.setGameObject(Game.getBlankGameObject());

}

previous = nextObject;

}

else

{

ChangeDirection();

}

return temp;

}

public void ChangeDirection()

{

if (direction == GameDirection.Right)

{

direction = GameDirection.Left;

}

else if (direction == GameDirection.Left)

{

direction = GameDirection.Right;

}

}

}

## **RandomGhost.cs**

internal class RandomGhost : Enemy //////// RandomGhost

{

public GameDirection direction = GameDirection.Down;

public RandomGhost(Image image, GameCell startCell) : base(image, startCell)

{

}

public int GetRandomDirection()

{

Random rnd = new Random();

int num = rnd.Next(4);

return num;

}

public override GameCell Move()

{

int dir = GetRandomDirection();

GameCell currentCell = this.CurrentCell;

GameCell nextCell = currentCell.nextCell(direction);

GameCell temp = new GameCell(nextCell.X, nextCell.Y, nextCell.gameGrid);

temp.setGameObject(nextCell.CurrentGameObject);

GameObject nextObject = nextCell.CurrentGameObject;

this.CurrentCell = nextCell;

if (currentCell != nextCell)

{

if (currentCell != nextCell)

{

if (previous.GameObjectType == GameObjectType.REWARD)

{

currentCell.setGameObject(Game.getRewardGameObject());

}

else

{

currentCell.setGameObject(Game.getBlankGameObject());

}

previous = nextObject;

}

}

if(dir == 0)

{

direction = GameDirection.Up;

}

else if(dir == 1)

{

direction = GameDirection.Down;

}

else if (dir == 2)

{

direction = GameDirection.Left;

}

else if (dir == 3)

{

direction = GameDirection.Right;

}

return temp;

}

}

## **SmartEnemy.cs**

internal class smartEnemy : Enemy //////////// SmartEnemy.cs

{

GameDirection direction;

GameObject player;

public smartEnemy(Image image, GameCell startCell) : base(image, startCell)

{

}

public smartEnemy(Image image, GameCell start, GameObject pacman) : base(image, start)

{

this.CurrentCell = start;

this.player = pacman;

}

public void changeDirection()

{

double[] distance = new double[4] { 1000000, 1000000, 1000000, 1000000 };

GameCell nextUp = this.CurrentCell.nextCell(GameDirection.Up);

GameCell nextDown = this.CurrentCell.nextCell(GameDirection.Down);

GameCell nextRight = this.CurrentCell.nextCell(GameDirection.Right);

GameCell nextLeft = this.CurrentCell.nextCell(GameDirection.Left);

//if (nextLeft.CurrentGameObject.GameObjectType == GameObjectType.WALL || nextLeft.CurrentGameObject.GameObjectType == GameObjectType.REWARD )

{

distance[0] = getDistance(GameDirection.Left);

}

//if (nextRight.CurrentGameObject.GameObjectType != GameObjectType.WALL && nextRight.CurrentGameObject.GameObjectType != GameObjectType.REWARD)

{

distance[1] = getDistance(GameDirection.Right);

}

//if (nextDown.CurrentGameObject.GameObjectType != GameObjectType.WALL && nextDown.CurrentGameObject.GameObjectType != GameObjectType.REWARD)

{

distance[2] = getDistance(GameDirection.Down);

}

//if (!(nextUp.CurrentGameObject.GameObjectType == GameObjectType.WALL || nextUp.CurrentGameObject.GameObjectType == GameObjectType.REWARD))

{

distance[3] = getDistance(GameDirection.Up);

}

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

{

direction = GameDirection.Left;

}

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

{

direction = GameDirection.Right;

}

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

{

direction = GameDirection.Down;

}

else

{

direction = GameDirection.Up;

}

}

public override GameCell Move()

{

GameCell currentCell = this.CurrentCell;

GameCell nextCell = currentCell.nextCell(direction);

GameCell nextCopy = new GameCell(nextCell.X, nextCell.Y, nextCell.gameGrid);

nextCopy.setGameObject(nextCell.CurrentGameObject);

GameObject nextObject = nextCell.CurrentGameObject;

this.CurrentCell = nextCell;

if (nextCell != null && currentCell != nextCell)

{

if (previous.GameObjectType == GameObjectType.REWARD)

{

currentCell.setGameObject(Game.getRewardGameObject());

}

else

{

currentCell.setGameObject(Game.getBlankGameObject());

}

changeDirection();

previous = nextObject;

}

else

{

changeDirection();

}

return nextCopy;

}

public double getDistance(GameDirection direction)

{

return Math.Sqrt(Math.Pow((player.CurrentCell.X - this.CurrentCell.nextCell(direction).X), 2) + Math.Pow((player.CurrentCell.Y - (this.CurrentCell.nextCell(direction).Y)), 2));

}

}

## **VerticalGhost.cs**

internal class VerticalGhost : Enemy ///////////// Vertical Ghost

{

public GameDirection direction = GameDirection.Down;

public VerticalGhost(Image image, GameCell startCell) : base(image, startCell)

{

}

public override GameCell Move()

{

GameCell currentCell = this.CurrentCell;

GameCell nextCell = currentCell.nextCell(direction);

GameCell temp = new GameCell(nextCell.X, nextCell.Y, nextCell.gameGrid);

temp.setGameObject(nextCell.CurrentGameObject);

GameObject nextObject = nextCell.CurrentGameObject;

this.CurrentCell = nextCell;

if (currentCell != nextCell)

{

if (currentCell != nextCell)

{

if (previous.GameObjectType == GameObjectType.REWARD)

{

currentCell.setGameObject(Game.getRewardGameObject());

}

else

{

currentCell.setGameObject(Game.getBlankGameObject());

}

previous = nextObject;

}

}

else if (direction == GameDirection.Down || CurrentCell.CurrentGameObject.GameObjectType == GameObjectType.WALL)

{

direction = GameDirection.Up;

}

else if (direction == GameDirection.Up || CurrentCell.CurrentGameObject.GameObjectType == GameObjectType.WALL)

{

direction = GameDirection.Down;

}

return temp;

}

}