**MARMARA UNIVERSITY**

**DEPARTMENT OF COMPUTER ENGINEERING**



CSE1042.1

TERM PROJECT

***Project Name:***

**Fighting in The Space**

## *Project Group:*

## Onur Sabit Salman – 150114501

**Mehmet Tuğberk Çelik - 150113057**

INSTRUCTOR: Assoc. Prof. Dr. ALİ FUAT ALKAYA

This project aims that when people play this game, they can remember their old stories because this game look like a mobil game(nokia 3310) which is played 15 years ago. Also, begin to think more faster and systematic.

**About Fighting in The Space**

Fighting in The Space look like a mobil game which is played on nokia 3310 before 15 years ago. In this game player have three kind of space ship and he/she can pick one space ship in options menu. When game is started, player’s aim is that kill ol of the enemies and bosses.

**Fighting in The Space**

In this program, we used twelve classes that are named as GameRun , GameWindow , StartWindow , SettingsWindow , Ammo , Boom , SpaceShip , BoomSound , FastAmmoSound , GameSound , LaserSound and StrongAmmoSound.

1)GameRun

GameRun is our first class. When the game is run, this class work firstly because this class have main method. And main method included StartWindow object.

Method: **StartWindow** **frame** = **new** StartWindow();

2)StartWindow

StartWindow is our second class but player or user see first this window. When run the program or click the FITS.exe which created by us player will see 5 labels. These labels are Fighting in The Space , START, LOAD, SETTINGS and QUIT. When player click the START, opening GameWindow class and player can play the game.

These codes creating this labels :

**JLabel** titleLabel = **new** JLabel("FIGHTING IN THE SPACE");

**JLabel** lblStart = **new** JLabel("START");

**JLabel** lblLoad = **new** JLabel("LOAD");

**JLabel** lblSettings = **new** JLabel("SETTINGS");

**JLabel** lblQuit = **new** JLabel("QUIT");

When player click the LOAD, opening game screen which is saved before. When player click the SETTINGS, opening SettingWindow. When player click the QUIT, the game will close.

We set up background image with this code:

**JLabel** backgroundImage = **new** JLabel(**new** ImageIcon("Pictures/1.gif"));

These are our panels code:

**JPanel** contentPanel = **new** JPanel();

**JPanel** menuPanel = **new** JPanel();

3)GameWindow

When click the START label openin GameWindow. Firstly, we created a panel.

Code: **private** **JPanel** contentPane;

Then we created 4 spaceship object and boom object.

Code: **private** **SpaceShip** gemi,enemy,boss1,boss2;

**private** **Boom** boom ;

Then we created a integer values to set a screen size.

Code: **private** **final** **int** SCREENHEIGHT= 900;

**private** **final** **int** SCREENWIDTH= 1200;

Then we created 3 arraylist enemyList, ammoList and boomList.

Code: **private** **ArrayList**<SpaceShip> enemyList = **new** ArrayList<SpaceShip>();

**private** **ArrayList**<Ammo> ammoList = **new** ArrayList<Ammo>();

**private** **ArrayList**<Boom> boomList = **new** ArrayList<Boom>();

Then we created labels and bolean valuse to different aims. For example, scoreLbl’s aim is that seeing player score.

Code: **private** **JLabel** bigText = **new** JLabel();

**private** **boolean** paused = **false**;

**private** **JLabel** scoreLbl = **new** JLabel();

**private** **JLabel** lifeLbl = **new** JLabel();

**private** **JButton** btnTryAgain = **new** JButton("PLAY AGAIN");

**private** **JButton** btnExit = **new** JButton("QUIT");

Then we created 2 different construction. First construction to normal START game. Second construction to LOAD game.

Then we created init because we dont want to write same code in two construction and we think we need init. İnit generally included GUI desing.

Then we created 2 get Array list method

Code: **public** **ArrayList**<SpaceShip> **getEnemyList**(){

**return** enemyList;

}

**public** **ArrayList**<Boom> **getBoomList**(){

**return** boomList;

}

Then we created TimerListener to move somethings which we want to move.

In the TimerListener automatic creating enemies and their ammos. And they moved specific direction. When score equal to 300, creating weak boss but when score equal to 1000, creating final boss and it is really strong.

Then we created MyController and we extends KeyAdapter to use keyboard.

And when player press the VK\_UP player’s spaceship moved up. When player press the VK\_DOWN , VK\_RIGHT or VK\_LEFT player’s spaceship moved down, right or left. When player press the VK\_A , VK\_S or VK\_D , spaceship fired a ammo but their types are different.

We have a small trick. When you press VK\_L , your life being 10 and if you know this trick you can be immortal.

Code: **else** **if**(e.getKeyCode() == **KeyEvent**.***VK\_L***){

life = 10;

lifeLbl.setText("Life :" + life);

Then we created removeCrashed and it’s aim is check the intersects.

Addition, when player click left-up File, opening 3 choses.

Save = Player can save the game.

Pause = Player can pause the game.

Exit = Player can exit the game.

4)SettingsWindow

When click the SETTINGS label openin SettingsWindow. Firstly, we created a panel.

Code: **private** **JPanel** contentPane;

Then we created 4 labels to different aims. For example, lblBack’s aim is that back to StartWindow.

Code: **JLabel** lblSettings = **new** JLabel("SETTINGS");

**JLabel** lblSpaceShips = **new** JLabel("SPACE SHIPS =");

**JLabel** lblBack = **new** JLabel("<- BACK");

**private** **final** **JLabel** lblNewLabel = **new** JLabel("HOW TO PLAY");

We created ComboBoxEvents and implements ActionListener to change spaceship. We have 3 kind of spaceship different features.

Code: **public** **class** **ComboBoxEvents** **implements** ActionListener {

**public** **void** **actionPerformed**(**ActionEvent** e) {

**if**(e.getSource().equals(shipType)){

selectedShip = shipType.getSelectedIndex();

}

Then we created getSelectedShip to send which spaceship is selected.

Code: **public** **int** **getSelectedShip**() {

**return** selectedShip;

}

5)SpaceShip

This class directly object. Firstly we created array ImageIcon to see ship’s images.

Code: **private** **static** **ImageIcon** [] *gemiImage* = {**new** ImageIcon("Pictures/space1.png"),**new** ImageIcon("Pictures/space2.png"),

**new** ImageIcon("Pictures/space3.png"),**new** ImageIcon("Pictures/enemy.png"),

**new** ImageIcon("Pictures/boss1.png"), **new** ImageIcon("Pictures/boss2.png")};

Then we created array width and height to set size ships.

Code: **public** **static** **final** **int** [] ***WIDTH*** = {54,121,100,49,121,100};

**public** **static** **final** **int** [] ***HEIGHT*** = {38,100,100,64,100,100};

Then we created array step to move ships.

Code: **public** **static** **final** **int** [] ***STEP*** = {15,15,15,5,15,15};

Then we created array numbers like a 0,1,2,3,4,5 because we need to pick arrays members.

Then we created construction.

Then we created getType to learn the space ship type.

Code: **public** **int** **getType**() {

**return** type;

Then we created setType to set the space ship type.

Code: **public** **void** **setType**(**int** type) {

**this**.type = type;

Then we created different get and set methods to learn or set sometings.

6)Ammo

This class directly object. Firstly we created array ImageIcon to see ammo’s images.

Code: **private** **static** **ImageIcon** [] *kursunImage* = {**new** ImageIcon("Pictures/blasterTwo.png"),**new** ImageIcon("Pictures/bullet2.png"),

**new** ImageIcon("Pictures/laser.png"),**new** ImageIcon("Pictures/blaster.png")};

Then we created array width and height to set size ammos.

Code: **private** **static** **final** **int** [] ***WIDTH*** = {29,90,80,29};

**private** **static** **final** **int** [] ***HEIGHT*** = {50,39,3,50};

Then we created array step to move ammo.

Code: **private** **static** **int** [] *STEP* = {10,15,45,10};

Then we created array delay to speed ammo.

Code: **private** **static** **int** [] *DELAY* = {10,30,20,10};

Then we created array numbers like a 0,1,2,3 because we need to pick arrays members.

Then we created construction.

Then we created setLocation to set ammo’s x and y.

Code: **public** **void** **setLocation**(**int** x,**int** y){

**super**.setLocation(x, y);

**this**.x = x;

**this**.y = y;

Then we created get a lot of method to learn sometings.

7)Boom

This class directly object. Firstly we created array ImageIcon to see ammo’s images.

Code: **private** **static** **ImageIcon** [] *explosionImage* = {**new** ImageIcon("Pictures/explosion.gif")};

Then we created array width and height to set size boom.

Code: **private** **static** **final** **int** [] ***WIDTH*** = {142};

**private** **static** **final** **int** [] ***HEIGHT*** = {200};

Then we created contruction.

Then we created setLocation to set boom’s x and y.

Code: **public** **void** **setLocation**(**int** x,**int** y){

**super**.setLocation(x, y);

**this**.x = x;

**this**.y = y;

Then we created get a lot of method to learn sometings.

Note: We created array and this array have 1 member because we planning add different kind of explosions but later we dont want to add.

8)GameSound/BoomSound/FastAmmoSound/LaserSound/StrongAmmoSound

These classes have same code. The only differences is named of sounds.

These code playing music.

Code:

**URL** **url** = **this**.getClass().getClassLoader().getResource("StarWars.wav");

*clip* = **AudioSystem**.*getClip*();

**AudioInputStream** **audio** = **AudioSystem**.*getAudioInputStream*(url);

*clip*.open(audio);

*clip*.start();