|  |  |
| --- | --- |
| **Project title** | **Star Wars: Rebels Resurrect** |
| **Author(s)** | **Iordache Paul-Cristian** |
| **Group** | **30424** |

# Task Description

Title: Star Wars: Rebels Resurrect

Description:

Star Wars: Rebels Resurrect is an engaging Java-based space shooter game that brings the iconic Star Wars universe into the realm of classic arcade gameplay. Players take control of a Rebel Alliance starfighter, battling against the formidable Imperial forces in two intense levels.

Level 1: Space Skirmish

In the first level, players navigate the vastness of space, encountering Imperial TIE Fighters and other iconic ships from the Star Wars universe. The goal is to accumulate 30 points by successfully shooting down enemy spacecraft. Players will face increasingly challenging waves of enemies, testing their piloting and shooting skills.

Level Transition:

Upon reaching the 30-point threshold, players advance to the second level, signaling a critical shift in the battle.

Level 2: Approaching the Death Star

Level 2 intensifies the challenge as players find themselves closer to the ominous Death Star. The iconic space station looms in the background, adding a palpable sense of urgency to the mission. The Imperial forces become even more relentless, and players must navigate through a barrage of enemy attacks while avoiding collisions with the massive Death Star.

Objective:

The ultimate objective remains the same - survive the onslaught, accumulate points, and strike a blow against the Empire. Success in Level 2 not only requires skillful piloting but also strategic decision-making to evade obstacles and choose the right moments to engage enemies.

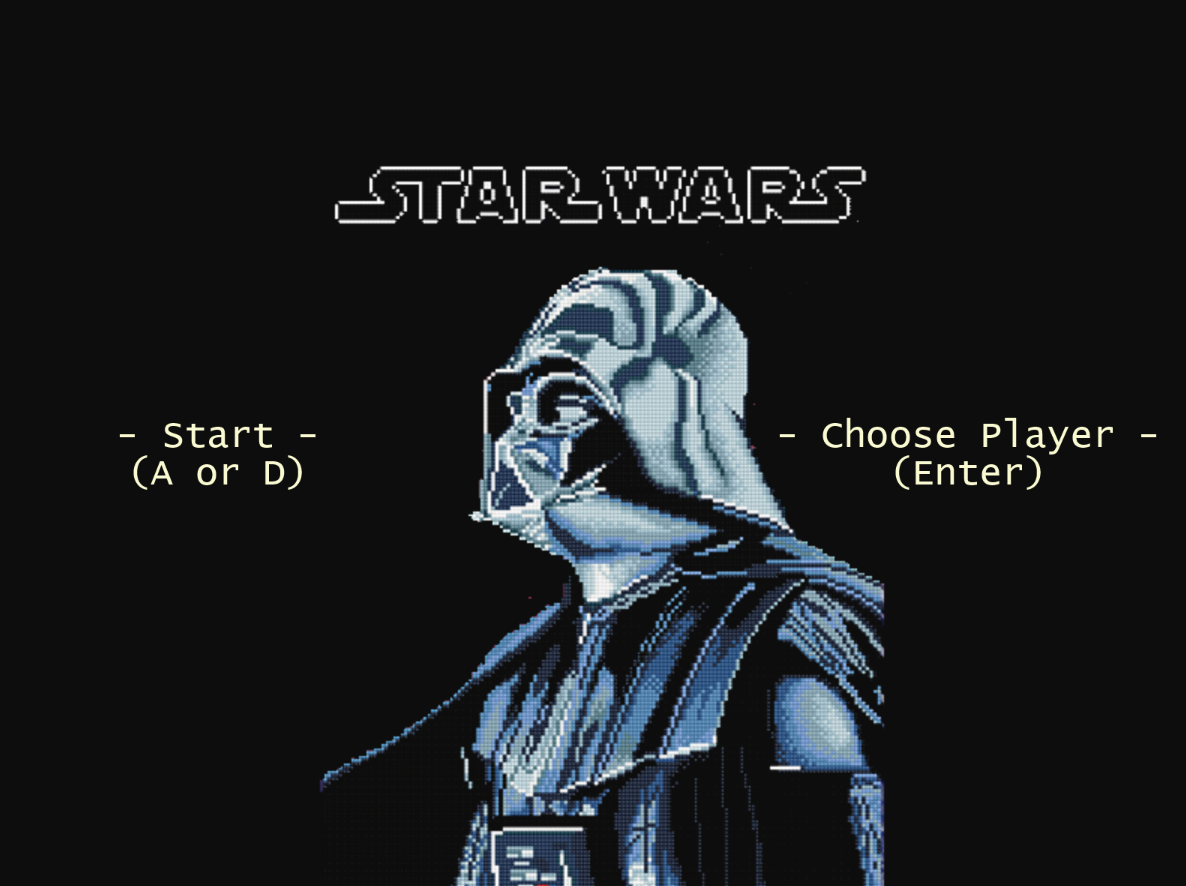
***Prior to opening the game files, javafx-sdk-21.0.1 needs to be imported***

***How to start the game***: Open SpaceInvaders.java

(file: ..\SpaceInvaders\src\application\SpaceInvaders.java)

*Select “SpaceInvaders” in Run Configurations and Run the Program*

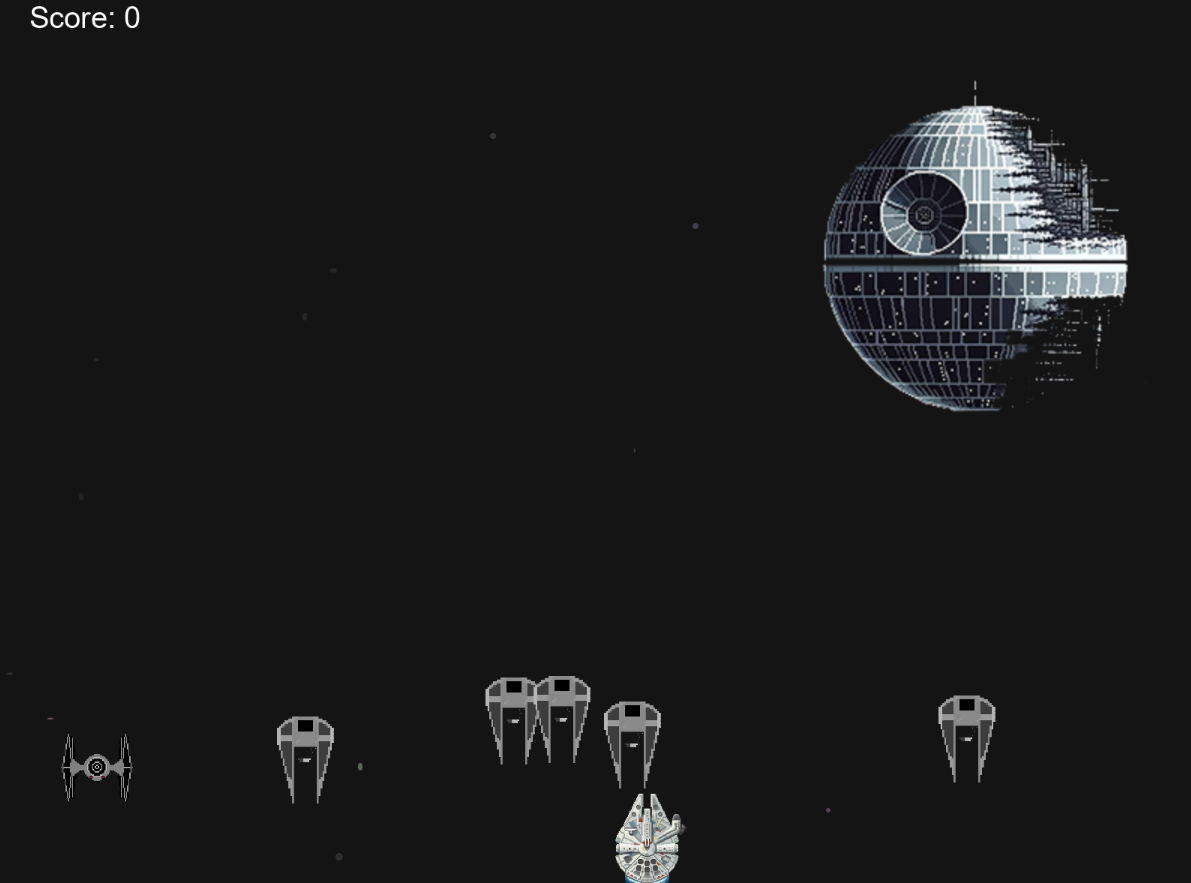
1. You will be met with this starting screen:



1. Player Selection (Press Enter)



1. Chose a player or pressed A or D directly from the starting menu (see 1)





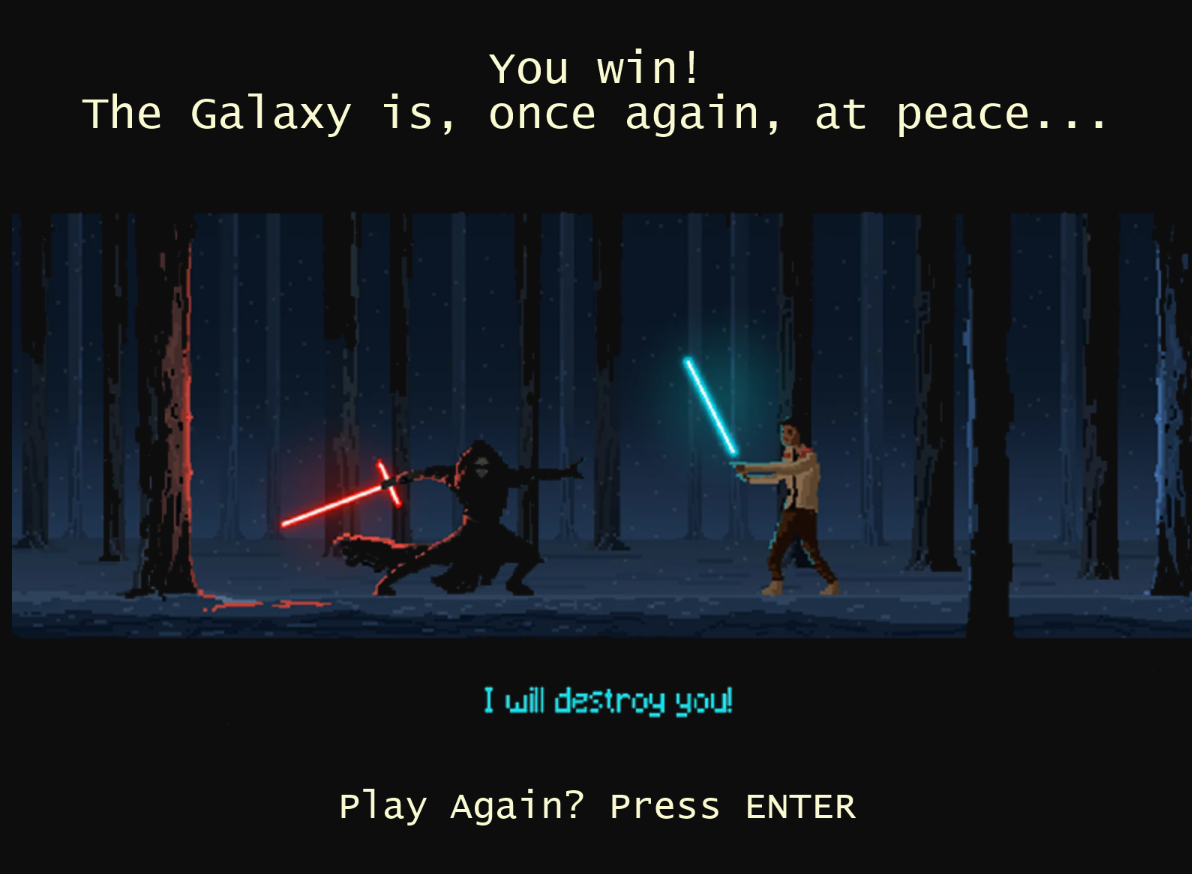
1. Geeting 30 points should take you to this screen



1. Pressing “ENTER” takes you to the next level



1. Acquiring 30 points, once again, transports you to the winning screen



1. Collision with an enemy means GAME OVER



# Class Discovery

|  |  |
| --- | --- |
| **Player** | |
| Moves the player around horizontally  Evades obstacles  Shoots the enemy  Changes the universe whether or not it transitions to the next level | Laser  Enemy  Laser  Universe  SpaceInvaders |

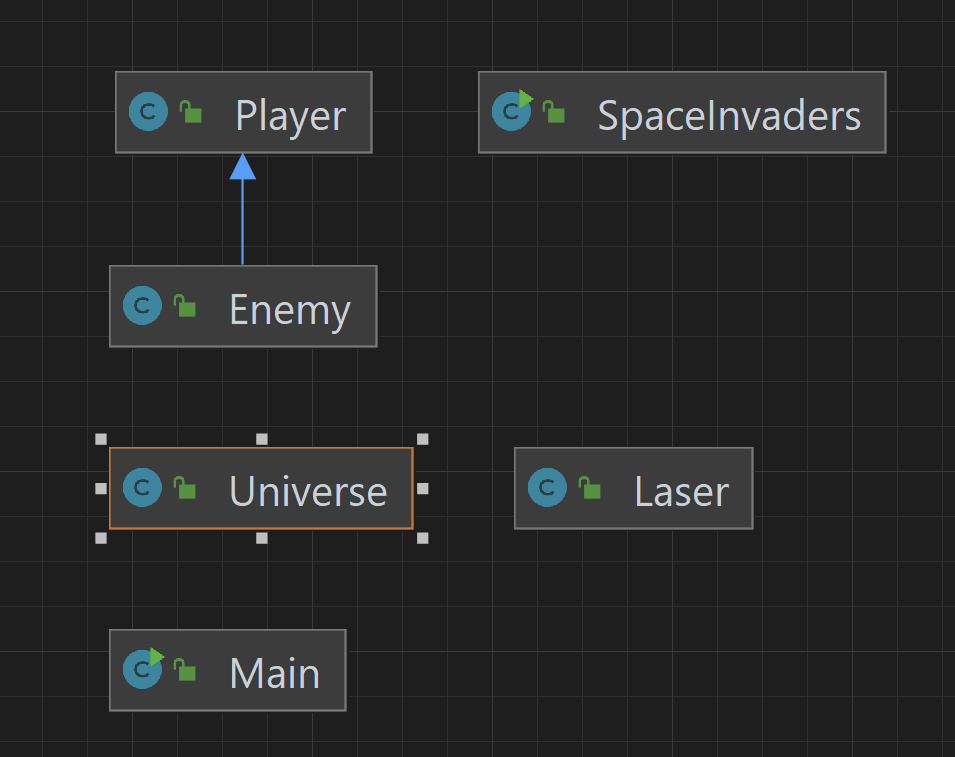
|  |  |
| --- | --- |
| **Enemy** | |
| Goes towards enemy  Needs to be destroyed in order to get points | Player  SpaceInvaders |

|  |  |
| --- | --- |
| **Laser** | |
| It is the laser shot by the player | Laser  Enemy  SpaceInvaders |

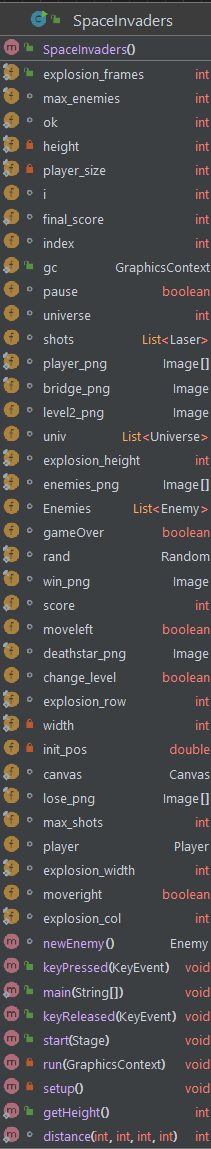
|  |  |
| --- | --- |
| **Universe** | |
| Computes the background stars | Player  SpaceInvaders |

|  |  |
| --- | --- |
| **SpaceInvaders** | |
| Sets the stage  Changes the level  Draws the player  Draws the starting and ending scene  Defines the movement logic | Player  Enemy  Universe  Laser |

# Class Diagram

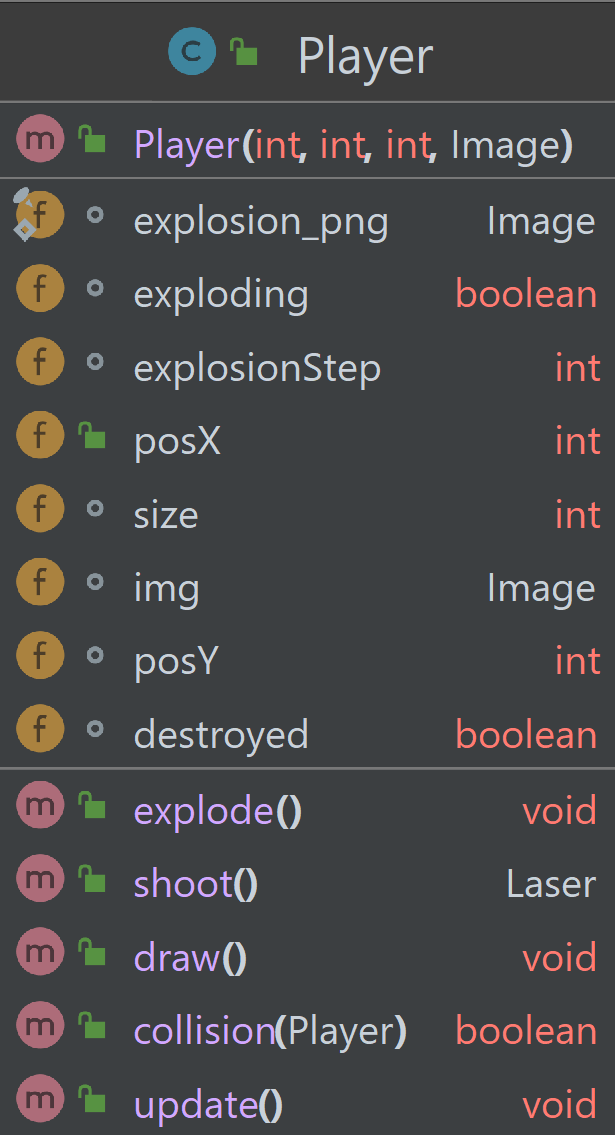


**SPACE INVADERS CLASS**



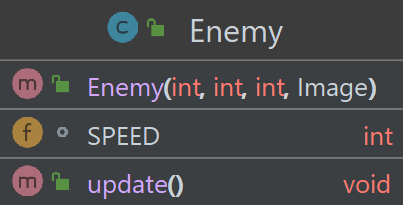
* Change\_level (Boolean): enables level changing.
* Explosion\_width (int): takes a .png file and is the set standard for the width sectioning.
* Enemies (List<Enemy>): list of enemies spawned.
* Pause (Boolean): enables/disables pause mode.
* Canvas (Canvas): it sets the canvas
* Lose/Enemies/Win/Player/Bridge\_png (Image): importing .png files
* Player (Player): player object
* gameOver (Boolean): flag for game over
* moveleft (Boolean): flag for moving left
* explosion\_height (int): takes a .png file and is the set standard for the height sectioning.
* Index (int): index for players
* Shots (List<Laser>): player lasers
* Moveright (Boolean): flag for moving right
* Max\_enemies (int): maximum number of enemies
* Rand (Random): randomly generated number
* Final\_score (int): changes the score if the game is not over
* Deathstar\_png (Image): Image of Death Star
* Level2\_png (Image): background Death Star .png file
* Explosion\_row (int): delimiting the rows of a .png file to replicate an animation
* Explosion\_col (int): delimiting the columns of a .png file to replicate an animation
* Gc (GraphicsContext): Graphics object used in writing sentences, displaying images and more
* Univ (List<Universe>): Background Image
* Ok (int): scene changing variable
* Max\_shots (int): maximum number of shots
* I (int): index for end game screen
* Explosion\_frames (int): frames of .png file
* Score (int): number of enemies destroyed, regardless of the game over flag
* Universe (int): color of background
* Void keyPressed (KeyEvent): method for keys pressed
* Int getHeight(): getter for private height variable
* Void start(Stage): sets all elements before launching
* Void main(): Actual start
* Void run(Graphics Context): method to run all the levels, transitions and the logic for the game
* Void keyReleased(KeyEvent): method for keys released, used in smooth movement
* Enemy newEnemy(): spawns new enemy object
* Int distance(int, int, int , int):

**PLAYER CLASS**



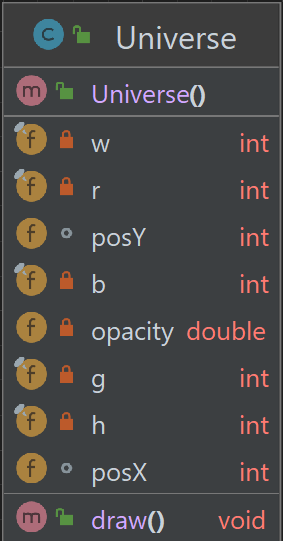
* Constructor Player (int, int, int, Image): Constructor for the Player object
* Explosion\_png (Image): explosion .png file imported
* Exploding (Boolean): flag to check whether or not the player has been defeated
* explosionStep (int): integer used in iterating through the .png file frames
* posX (int): horizontal position of the Player
* size (int): size of the Player (x and y based)
* img (Image): Player image
* posY (int): initial vertical position of the Player
* destroyed (Boolean): flag to start the destruction animation
* void explode(): method for destruction
* Laser shoot(): spawn laser
* Void draw(): draw Player model (default or exploding)
* Boolean Collision(Player): checks if the player collided with an enemy
* Void update(): update player’s model

**ENEMY CLASS**



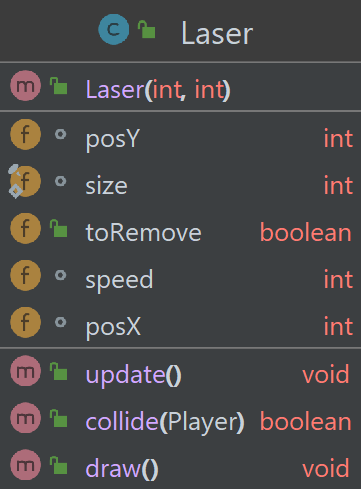
THIS CLASS INHERITS FROM THE “PLAYER” CLASS

* Constructor Enemy (int, int, int, Image): Constructor for the Enemy Object (uses super)
* Speed (int): speed of the enemy, based on the score
* Void update(): update the model of the enemy (default or exploding)



* Constructor Universe(): Constructor for the Universe object
* w h (width and height) (int): spanning of the universe
* posY, posX (int): position of the stars
* r, g, b (int): RGB for the stars (variations of gray)
* void draw(): draw the universe

**LASER CLASS**



* Constructor Laser (int, int): Constructor for the Laser object
* posY, posX (int): vertical and horizontal position of the laser
* size (int): size of the laser – width and height
* toRemove (Boolean): flag to remove the item from the screen
* speed (int): speed of the laser
* void update(): update the laser
* Boolean collide (Player): check whether or not the laser collided with the player
* Void draw(): draw the actual laser