UE20CS352 : Object Oriented Analysis & Design with Java

SPACE INVADERS GAME

Rehan Ganapathy MU - PES2UG20CS269
Saakshi H Srinivasan - PES2UG20CS290
Sakshi Hulageri - PES2UG20CS300

Synopsis:

The Java project implementing Space Invaders is a classic arcade-style game where the player controls a spaceship and must defend the Earth from an invading alien army.
The player's spaceship can move horizontally at the bottom of the screen and shoot missiles to destroy the alien spaceships. The alien spaceships move in a predetermined pattern and shoot back at the player.
The graphics and sound effects are implemented using the JavaFX library. The game features a main menu and high score tracking. The project includes unit testing and error handling to ensure a robust and stable game.

The project uses object-oriented programming principles, including inheritance and polymorphism, to organize and manage the game's various entities. The code is well-documented and adheres to coding best practices.

Use Case:

Use case Specifications:

Name: Space Invaders
Summary: Players will be able to control the spaceship at the bottom of the screen, which moves horizontally. They can shoot missiles at the aliens to destroy them. The player dies when the alien ship collides with them. The player scores by shooting aliens.
Actors: Player, Game engine
Pre conditions: The player has to start the game.
Description:
- The player starts a new game.
- Player can move spaceship left or right using the arrow keys or touch controls.
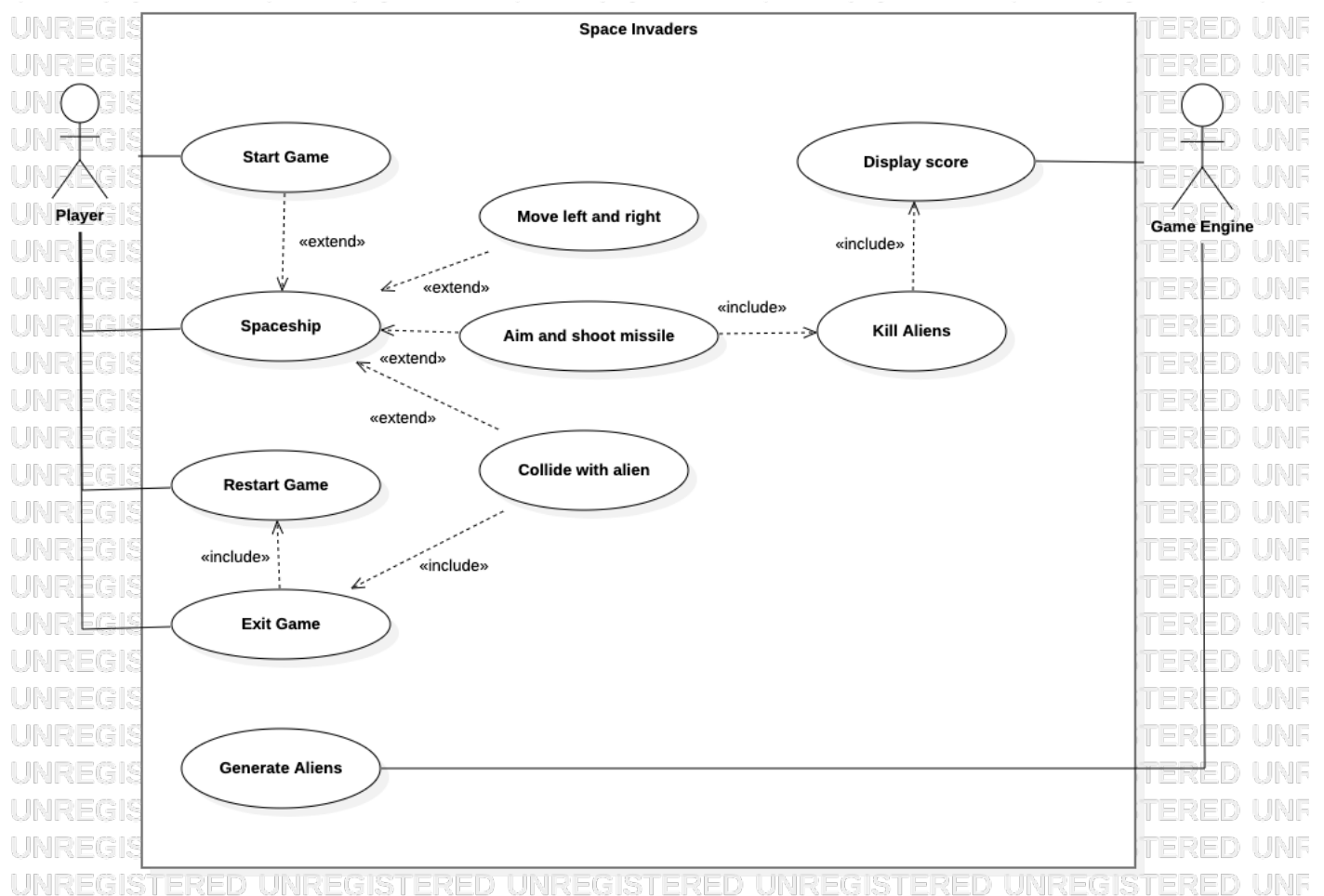
- The aliens descend towards the spaceship.
- The player shoots missiles at the aliens to destroy them.
- If the alien collides with the player, the player dies.
- The game ends when the player dies and the player can choose to restart the game.

Exceptions: If the game encounters a critical error during gameplay, such as a crash or freeze, the game will automatically save the player's progress and exit to the main menu.
Alternate Flows: Game pauses if the mouse pointer goes outside the boundaries of the game screen.
Post conditions: The player has ended the game and can choose to restart.

Use case Diagram:

Class Diagram:

Game

+score: int
+player: Player
+alien_list: Alien
+missile_list: Missile

+start()
+restart()
+quit()
+moveSpaceship()
+moveAliens()
+checkCollisions()
+updateScore()

Alien

+speed: int
+x_pos: int
+y_pos: int
+sprite: Image

+move()
+collide(Spaceship)

Missile

+color: Color
+x_pos: double
+y_pos: double
+speed: int

+update(): void
+out_of_bounds(): boolean
+collide(Alien): boolean

Spaceship

+x_pos: int
+y_pos: int
+speed: int
+sprite: Image

+moveLeft()
+moveRight()
+shoot()
+die()