

Joseph Kenyon

Version: 1.0

**OVERVIEW:**

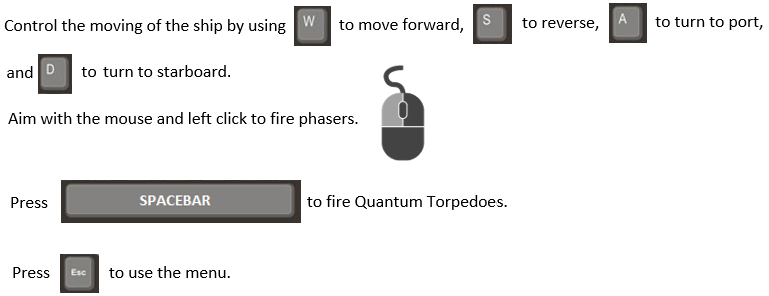
Stark Trek Commander is a Science Fiction Combat Flight Simulation game based on the Star Trek: The Next Generation TV and film series. The player commands the Starship Enterprise-E and battles against an enemy ship.

**GAME DESCRIPTION:**

Objective – The objective of the game is to wear down the enemy ship’s shield and destroy it before the enemy can do the same to you.

Gameplay – Each starship has an ovular shield surrounding their ship that will stop incoming projectiles before reaching the hull. The shield can be worn down on any one of its sides and then projectiles will penetrate the shield and damage the hull. The player flies around their starship, damaging the other starship with Phasers and Quantum Torpedoes.

Controls:

****

**USER MANUAL:**

Upon entering the game, the user will see this start-up Screen:



Figure : Main Menu

The player can select “Play” or “Exit Game”. If the latter is selected, the program is terminated. If the player decides to play, they can then select their difficulty. Easy causes the player to deal more damage and prevents the enemy ship from regenerating shields. Hard makes the player take more damage from attacks.



Figure : Difficulty Selection

Upon selecting the difficulty the player will immediately enter the space arena and begin battle. To move around, the player must use the W, A, S, D keys for forward, backward, left-turn, and right-turn respectively. The play must use these keys to evade enemy attacks like the green disruptor blasts in figure 3.

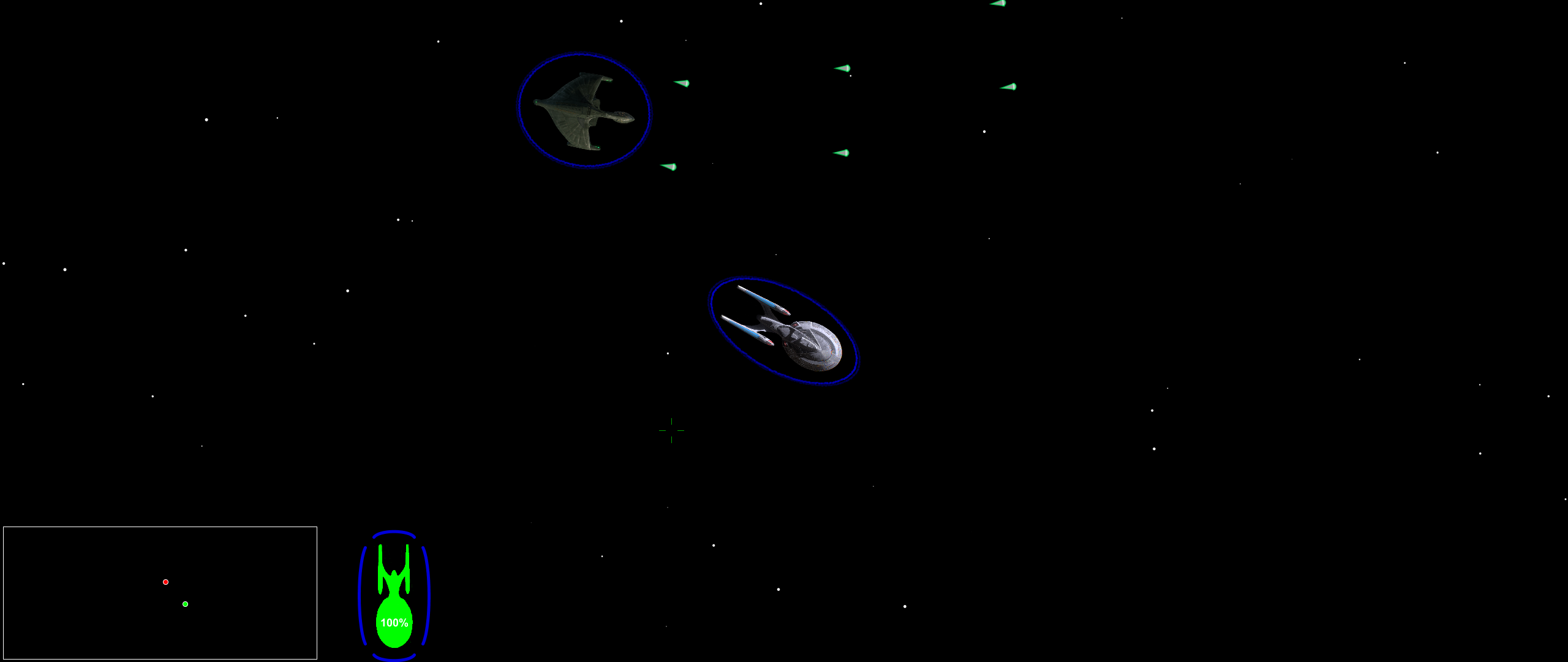


Figure : Disruptor Blasts

The primary weapon of the enterprise is the Phaser. The phaser is fired by clicking the mouse and is fired in the direction of the cursor. The Phaser is designed to deal a lot of damage to shields, but is weak against the hull of the enemy ships.

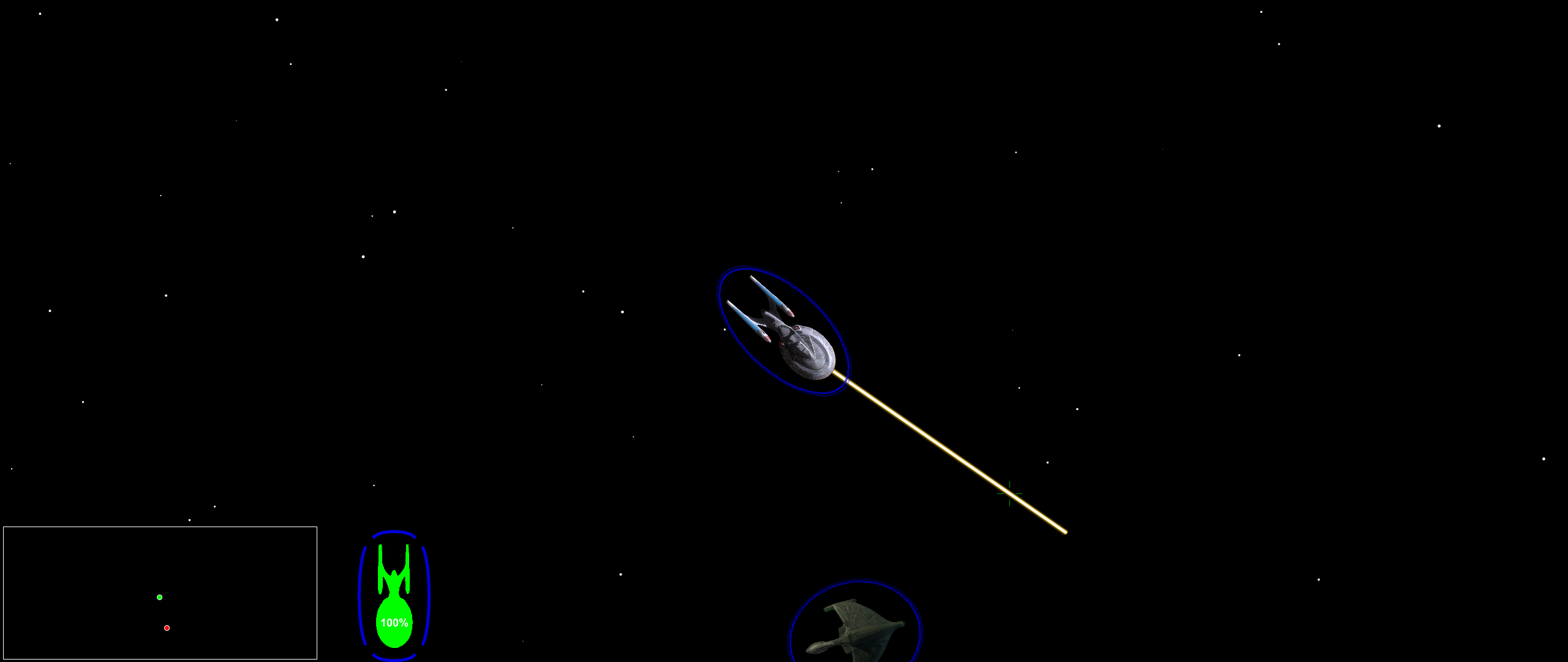


Figure : Phaser Blasts

The secondary weapon of the enterprise is the Quantum Torpedo. This torpedo is weak against shields but is powerful against the enemy ships hull. The optimal strategy is to wear down their shield with phasers and then fire a Torpedo into the shield gap.



Figure : Quantum Torpedo

The ships shields can be redistributed to a section running low by clicking the desired section on the shield indicator in the bottom left.



Figure : Shield redistribution

If a ship is struck enough times in a section with its shields down, the ship will explode and victory is achieved!



Figure : Victory Conditions

**CODE STRUCTURE:**

The program is organized by a series of .pde files:

Main.pde, Effect.pde, Enterprise.pde, HUD.pde, Miscellaneous.pde, Objects.pde, Projectile.pde, Ship.pde, Sound.pde, Use\_Input.pde and Weapon.pde.

Main.pde is the central file of the project and holds all of the global variables, the game loop, and is used to call all of the other methods throughout the program. It contains the standard setup() and draw() functions, and also a method called startGame() which is used to start a new game;

Effect.pde contains a class called Effect that is used for visual effects like explosions. This class has a constructor and a draw method, which updates the effect frame and draws the effect.

HUD.pde is used for drawing all of the HUD elements like the mini-map and Menu screens.

* drawMiniMap() draws the mini-map in the bottom left corner of the screen.
* drawShipStatus() draws the green or red enterprise in the bottom portion of the screen alerting the user to the status of their ship.
* The Menu Class is used for drawing menus and has the following functions: update\_draw() which updates and draws the Menu screen. The Menu class also contains Containers which contain Buttons. This hierarchy allows for easily switching between options in the menu.
* createReplayMenu() is used to create the menu that the user sees when they win, asking them if they want to play again or exit the game.

Miscellaneous.pde lives up to its name justand contains a few odds and ends.

Objects.pde contains a super class called Object which is extended by other classes to be explained later.

Projectile.pde contains all of the classes used for projectiles like phasers, disruptors and torpedos.

* The Projectile Class has the standard update() and draw() methods, as well as collideWithShield() which returns a Boolean determined by whether or not this projectile has collided with a shield, collideWithShip() which does the same but for ship hulls, move() which moves the projectile according to its rotation and returns -1 if its out of range and should be terminated.
* The Phaser Class extends Projectile and is used for differences the phaser has from other projectiles. The same goes for the Quantum Torpedo Class.

Ship.pde contains the Enterprise class which extends the Ship class. The Ship class contains the following methods:

* RunAI() which directs the action of the enemy ship.
* Update() which normalizes the shields, fires all weapons if inputted by the user, and calls the update() methods of all projectiles emanating from the ship.
* Fire() fires the current weapon and spawns the appropriate projectile for that weapon.
* ClosestShieldPoint() returns the closest shield impact point from the specified coordinates.
* ShieldImpact() handles the ship’s shield being fired upon and lowers its health.
* ShipImpact() does the same but for hull impacts.
* ShieldRadiusX() and ShieldRadiusY() return the radius of the shield in their respective directions.
* GetShieldX() and getShieldY() return the location of the shield point at the specified index.
* ShieldAverage() returns the average value of all the shield points.
* NormalizeShields() averages out the shield values and regenerates damaged sections.

Sound.pde handles sounds and its method play() is a way of preventing duplicated sounds from playing at the same time.

User\_Input.pde handles all of the Key and Mouse events. The runKeyboard() method moves the ship according to whether or not the movement keys are down or not.

Weapon.pde contains the Weapon Class which is assigned to Ships to help facilitate the projectiles that the ships spawn.

* CanFire() returns true or false depending on whether or not that weapon has been fired too recently or not to be fired again.
* Fire() spawns the desired projectile, based on its weapon type.

**ASSETS USED:**

Images:

* Enterprise.PNG and Romulan\_warbird.PNG are screenshot of a 3D model found at: <https://sketchfab.com/3d-models/enterprise-e-c68e0b0a794446a1ae04ebf564b41b16> and <https://sketchfab.com/3d-models/romulan-warbird-7dd1ca506d2342fdb19b7eb7d17b03dd>
* Enterprise\_hull.PNG was created from Enterprise.PNG via color correction.
* Explosion\_sprite\_sheet.PNG was found at: <https://www.seekpng.com/ipng/u2q8t4i1o0t4a9u2_drawn-explosions-sprite-explosion-sprite-sheet-doom/>
* Logo.PNG was created by me using a Star Trek font I found and by color correcting this image: <https://www.pinterest.com/pin/395331673510497529/>
* Quantum\_torpedo.PNG is an image I made in photoshop.

Audio:

* battle\_music.wav is an edited down version of the track “The Battle Begins” from the movie *Star Trek: Nemesis*.
* Menu\_theme.wav is an edited down version of the track “A New Beginning” from the movie *Star Trek: Nemesis*.
* All other sound effects were found on: <http://www.trekcore.com/audio/>