DropBox-Link: https://www.dropbox.com/sh/axd9visyb71kqy8/AADwSIPlV7YNs05AO4agqQsda?dl=0

Coursework Part B

*Game Development*

*“Space Arena”*

*Due date: 5.12.2016*

***Student:***

*Name: Sören Schreiber*

*Kingston ID: K1549360*

*Module leader: Dr. Vasils Argyriou(VA)*

*Lecturers: -Dr. Vasilis Argzriou* *(VA)*

*Marker: - Dr. Vasilis Argzriou* *(VA)*

# Table of Contents

[Table of Contents 2](#_Toc468721252)

[Introduction 3](#_Toc468721253)

[Assignment 3](#_Toc468721254)

[“Space Arena” 3](#_Toc468721255)

[Game Design 3](#_Toc468721256)

[Game Controls 7](#_Toc468721257)

[User Interface 7](#_Toc468721258)

[Player Control 7](#_Toc468721259)

[AI 9](#_Toc468721260)

[Game Controller 9](#_Toc468721261)

[Implemented tasks 9](#_Toc468721262)

[Source code 11](#_Toc468721263)

[Scripts 11](#_Toc468721264)

[Player Control 11](#_Toc468721265)

[AI 18](#_Toc468721266)

[Game Controller 24](#_Toc468721267)

[Others 28](#_Toc468721268)

[Results 37](#_Toc468721269)

[Bugs/Problems 37](#_Toc468721270)

[Not Implemented Features 38](#_Toc468721271)

[Annex 40](#_Toc468721272)

[Table of Figures 40](#_Toc468721273)

[Table of Tables 40](#_Toc468721274)

[References 42](#_Toc468721275)

# Introduction

The subject of second assignment for the module 3D games development is the development of a game or tool for any platform and using any of the technologies introduced during the course of the module.

## Assignment

The Games has to implement at least 1 of the following tasks:

1. Background. Include a (moving) background
2. Machinery. Put some interesting objects into your scene. To attract good marks, use non-trivial 2D/3D modelling, texturing or provide it with some kind of animated behaviour.
3. Vehicles/Moving Objects. Conceptually close to the machinery, but vehicles should be able to move around.
4. AI. If you’d like to try your hand in artificial intelligence, you are welcome! This is an open-ended topic.
5. Collision detection. You will have to introduce some kind of scenery to collide with. Collision detection should restrict players or NPCs motion based on contact conditions with other objects (e.g. walls).
6. Shooting. You will have to introduce some kind of target (enemy monsters?) and test the player’s accuracy. Extra bonus for moving targets!
7. Setting. This includes game title screen, menus, scoring, timing, status, hall of fame etc.
8. Anything else. You must obtain a go ahead from your Tutor if you have any other interesting idea.
9. Focus on a specific part and improve it as much as possible.

# “Space Arena”

## Game Design

“Space arena” is a single player oriented sci-fi shooter game set in various space arenas.

It is designed for mobile android devices.

The player controls his ship with input for speed, direction and targeting and is opposed by enemy fighters.

The game is set in stages and has 3 difficulty levels.

-Stages: before each stage the player is reset to the centre of the map as well as randomly select a new arena to be fought in. Each completed stage increases the difficulty factor by increasing the number of enemy fighters opposing the player.

-Difficulty levels: “Space Arena” has 3 difficulty settings (easy, medium, hard).

-Easy:

-Player: The player has 3 times the normal health and shields

The available lives of the player are 3 at the start.

-AI: The AI uses less of the maximum speed of its ship. The weapon control will select preferably the short range cannon. Rockets should be rare.

-Medium:

-Player: The player has 2 times the normal health and shields

The available lives of the player are 2 at the start.

-AI: The AI uses almost its maximum speed of its ship (still giving the player an edge). The weapon control will select preferably the short range cannon and beam weapon. Rockets will be more common.

-Hard:

-Player: The player has the same health and shields as the enemy fighters.

The player will start with only one life.

-AI: The AI uses its maximum speed of the ship. The weapon control will select equally between the 3 available weapon types.

-Arena: The Arenas are vast combat areas which contain different or no hazards as well as decorative object to make the void look a bit more alive.

The hazards are 3 different types of nebulae:

-Corrosion:

|  |  |
| --- | --- |
| Figure - Corrosion Nebula | This nebula will damage the hull of every ship within. This will only occur if the ships shields are down.  The nebula will prevent shields from regenerating. |

Table - Corrosion Nebula

-Energy:

|  |  |
| --- | --- |
| Figure - Energy Nebula | This nebula will damage only the shields of a ship caught within its effect radius. No shield regeneration will work while the ship remains in the nebula. |

Table - Energy Nebula

-Plasma:

|  |  |
| --- | --- |
| Figure - Plasma Nebula | This type combines the first two into a deadly hazard.  It will damage both shields and hull, as well as denying shield regeneration |

Table - Plasma Nebula

-Power-Up: There are 3 types of power ups available during the game. Each of them has a chance of dropping when a ship is destroyed.

-Health:

|  |  |
| --- | --- |
| Figure - Health Power Up | This power up will regenerate the hull of the ship which is picking it up |

Table - Health Power-Up

-Shields:

|  |  |
| --- | --- |
| Figure - Shield Power Up | Pick up this power up to replenish the shields energy of the ship |

Table - Shields Power-Up

-Lives:

|  |  |
| --- | --- |
| Figure - Life Power Up | A Power up that will increase the players remaining lives. |

Table - Lives Power-Up

While the lives power up can only be obtained by the player, every power up can be picked up and used (picking up “Live” will just remove the power up from the field) by a passing AI ship.

After the death of a player the ship will be respawned after 3 seconds at its last location.

|  |  |
| --- | --- |
| Figure - Ship without shields | Figure - Ship with shields |

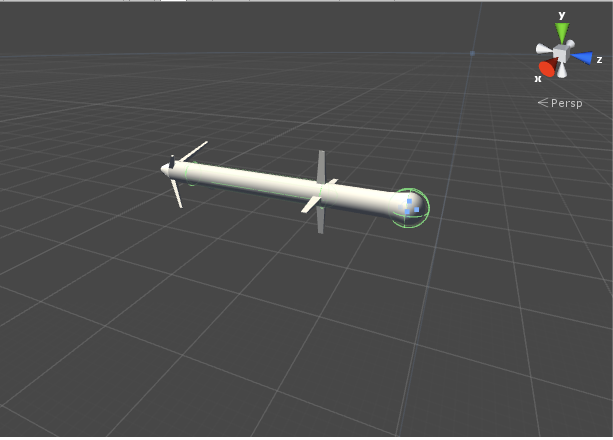


Figure - Rocket

At the end of the game the player can enter into the communal high score list if the game score was high enough to beat one of the 3 previous scores.

A save and load function is implemented as well as a pause option.

The Original design for the game included an Option for it to be played in AR mode. This caused a lot of unresolved problems and was removed for the final version of the game (For more information see Not Implemented Features).

## Game Controls

### User Interface

|  |  |  |
| --- | --- | --- |
| User Interface | Controls | Description |
| Figure - Main UI | -New Game  -Load Game  -Achievements  -Quit | Start Interface of the game. Provides controls to start a new game, load a previously saved game, view high scores and close the application |
| Figure - Select difficulty | -Easy  -Medium  -Hard  -Back to Main | Here the player can select between the 3 difficulties or return to the main screen |
| Figure - Pause game | -Continue  -Save Game  -Load Game  -Back to Main | This UI shows during a paused game.  It allows the user to save the current game, to load a previous game, continue playing or to quit the game unsaved and return to the main screen |
| Figure - High score | -Close | Shows the current high scores |
| Figure - New high score | -Submit | Form to enter a new high score to the game |

Table - User Interface

### Player Control

The final game implements basic joystick controls and button inputs using the device touchscreen. These control operate flying targeting and weapon control.

|  |  |  |
| --- | --- | --- |
| Control | Function | Description |
| LeftJoy-X | Ball(Left/Right) | Controls player ships rotation around its Y-Axis |
| LeftJoy-Y | Ball(Forward/back) | Controls rotation around ships X-Axis |
| RightJoy-X | Camera(Left/right) | Moves the crosshair left and right |
| RightJoy-Y | Camera(up/down) | Moves the crosshair up and down |
| Right Slider | Speed | Regulates player speed |
| Figure - Fire button | Fire | Fires selected weapon at target |
| Figure - Blaster cannon    Figure - Beam weapon    Figure - Rockets | Weapon selection | Switches to next weapon.  Cycles though all 3 weapon types   * Cannon * Beam * Rocket |
| Figure - Rocket targeting | Target selection | Selected next target within range (rockets only) |
| Figure - Pause | Pause | Button press will pause the game |
| Figure - Score Indicator | Current Score | Displays the current player score |
| Figure - Remaining Lives | Remaining lives | Show the players remaining lives |
| Bars at bottom of screen | Health/Shield Bars | Display players current health and shields |

Table - Player Controls

### AI

Every AI opponent has the same basic controls as the player (with exception of pause and crosshair movement).

All of these control inputs are being regulated by the AI script, which turns and accelerates the ship towards its target location, and the AI weapon control script, which regulates targeting weapon selection and fire rate.

This AI operates through a finite state machine, selecting appropriate actions for the different states of the AI.

More information can be found in the AI script section.

### Game Controller

The game controller is responsible for regulating the actual game (adjusting health bars, provide stage and ship information and interface control).

Neither player nor AI actively interact with the game controller. Only information is being exchanged.

## Implemented tasks

* Background/Map: The Nebula Skybox presents a visually nice background for the game. In addition, each map contains certain elements like nebulae or planets/suns/moons to fill the void within the game areas.
* AI: The game features an AI which works in two separate states controlled by a state machine
  + Patrol

Patrol between pre-set points on the map trying to find the player

* + Engage

Engaging the target in combat. Firing its weapons and following the player until contact is lost

* Shooting: The game includes 3 types of weapons:
  + Blaster

A prefab with is moved by using a force impulse towards the target location.

Will be destroy after moving a certain distance

* + Beam

A ray cast weapon which incrementally increases in length until it reaches its maximum or until it hits an object

* + Rocket

A self-contained prefab weapon which will home onto its target until it is reached or the rocket flew for a certain amount of time

* UI: The game contains multiple user interfaces and menus.
  + The User Interface provides all the control surfaces for the player to control the game as well as informational text elements.
  + The menus let the player navigate through the game options and functions.
* Machinery/Moving objects:
  + Particle systems:

Every map contains 1 or more particle systems designed for visual effects.

* + - Fields of stars: To fill the void and provide reference to movement
    - Nebulae: Visual objects with hazardous effects to ships within them
    - Planetary objects/systems: Solley visual provide no interaction with the player or AI. AI ships will try to avoid any object that can revive ray casts.
    - Explosions: Animated particle effects used to show destruction of a ship.
* Collison detection: The arena is not protected by an invisible wall. All AI ships and the player are locked in place by a cubical clamping function which will not allow movement outside the arena.

There are multiple collision detectors for nebulae effects and hit detection

* Lighting/sound: Weapons have their own sound emitters playing weapons fire when used.
* Setting: The game entails multiple menus throughout the application. For more information, see User Interface in the Game Controls section

# Source code

The following will detail all scripts used throughout the game. Each section will highlight one game script, detailing its variables and functions and possible problems, as well as not implemented features and know bugs/issues.

## Scripts

### Player Control

* Script :: PlayerControl

The player control script handles all input of the user related to the motion of the ship.

To Signal a turning motion it tilts the ship up to a 40\* angle depending on the input value. The engine particle system and the turn rate is connected to the throttle input increasing as speed is being increase by the player.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| GC | Private game controller | Reference to the game controller script |
| scanPerimiter | Public mesh renderer | Mesh component of the scan perimeter |
| Ship | Public transform | Transform of the ship object |
| LeftEngine | Public particle system | Left engines particle system |
| RightEngine | Public particle system | Right engines particle system |
| speed | Private float | Current player speed |
| maxSpeed | Public float | Maximum player speed |
| turnSpeed | Public float | Players turn speed |
| setStartSize | Private float | Start size for engine particles |
| setLifeTime | Private float | Life time of engine particles |

Table - Variables: Player Control

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| Start | -Input:  NONE  -Output:  NONE | Called at the creating of the object.  Assigns the game controller, and turns off the scanPerimiter mesh.  Set particle start size and lifetime |
| Update | -Input:  NONE  -Output:  NONE | Called every frame.  If the game is not paused, moves the player ship in accordance with the user input.  Tilts the ship component of the player to signal turning.  Clamps the player position within a 20000 units sided cube.  Controls the engine particle size to correspond to the current player speed |
| getSpeed | -Input:  NONE  -Output:  float | Return speed value to calling script |

Table - Functions: Player Control

* Script :: TargetControl

This script functions similar to the player control. Here the input for the targeting point is being handled.

The player crosshair is locked in a cone in front of the player between=-40 (left/right) and +-20 (up/down).

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| Player | Public gameobject | Reference to the player object |
| Target | Public gameobject | Reference to the crosshair |
| Camera | Public gameobject | Reference to the game camera |
| GC | Private game controller | Reference to the game controller |
| targetAngle | Public float | Maximum move angle |
| TargetHor | Private float | User input x-axis |
| targetVer | Private float | User input y-axis |

Table - Variables:Target Control

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| Start | -Input:  NONE  -Output:  NONE | Called at the start of the game.  Assigns both camera and game controller |
| Update | -Input:  NONE  -Output:  NONE | Called every frame.  When the game is not paused, move the target in accordance with the user input inside the angle restriction and follow the player object.  Turn the crosshair to look at the camera at all times |

Table - Functions: Target Control

* Script :: WeaponControl

Weapon control handles any input related to weapon systems.

Here the player can swap between the available weapons, target new enemies with rockets and fire his selected weapon type onto the target (crosshair or AIship).

Rockets can only be fired if the enemy is within +-90\* in front of the player.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| Rocket | Public gameobject | Rocket prefab |
| Cannon | Public gameobject | Blaster prefab |
| Crosshair | Public gameobject | Crosshair object |
| GC | Public gamecontroller | Game controller |
| rocketPods | Public gameobject[] | Array of rocket locations |
| cannonPods | Public gameobject[] | Array of cannon locations |
| beamEmitters | Public gameobject[] | Array of beam emitters |
| GameUI | Public gameobject | UI gameobject |
| Button | Public image | Weapon switch button |
| Firebutton | Public image | Weapon fire button |
| cycleTarget | Public image | Switch target button |
| rocketMat | Public sprite | Image for rockets |
| beamMat | Public sprite | Image for beam weapons |
| cannonMat | Public sprite | Image for cannon weapons |
| currentRocketScript | Private rockets | Script of currently selected rocket |
| currentBeamScript | Private beamweapons | Script of currently selected beam emitter |
| currentCannonScript | Private cannonShotscript | Script of currently selected cannon shot |
| selectedRocketPod | Private int | Indicator for current rocket pod |
| selectedCannonPod | Private int | Indicator for current cannon pod |
| selectedBeamEmitter | Private int | Indicator for current beam emitter |
| Rocket\_reload | Public float | Reload time for rockets |
| timeOfFire | Private float | Float value for last time a weapon was fired |
| rocketTarget | Public List<gameobjects> | List of all available targets for the rocket |
| lockedTarget | Private gameobject | Currently selected target gameobject |
| currentTarget | Private int | Current target number |
| Weapon\_select | Public int | Currently selected weapon |
| weaponRange | Private float | Max weapon range |

Table - Variables: Weapon Control

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| Start | -Input:  NONE  -Output:  NONE | Called at creation of gameobject.  Sets all default values for variables and finds needed gameobjects |
| update | -Input:  NONE  -Output:  NONE | Called every frame.  If the game is not paused read user input for weapon control, switches weapon types and manipulates the Game UI |
| spawnRocket | -Input:  GameObject  -Output:  NONE | Spawns a new rocket after reload delay in given rocket pod |
| fireBeam | -Input:  GameObject  -Output:  NONE | Fires beam weapon onto target from selected beam emitter  Selects next beam emitter |
| fireCannon | -Input:  GameObject  -Output:  NONE | Fires cannon onto target from current cannon pod. Selects next cannon pod |
| fireRocket | -Input:  GameObject  , GameObject  -Output:  NONE | Fires rocket connected to selected rocket pod.  Selects next rocket pod |
| cycleTarget | -Input:  NONE  -Output:  NONE | Selects next target in list and switches target lock on off |
| showButton | -Input:  NONE  -Output:  NONE | Show cycle target button |
| hideButton | -Input:  NONE  -Output:  NONE | Hide cycle target button |
| setWeaponRange | -Input:  NONE  -Output:  NONE | Turn crosshair on and off |
| isTargetInWeaponAngle | -Input:  Float, vector3  -Output:  bool | Reads if target is within given targeting angle |

Table - Functions: Weapon Control

* Script :: ShipEventController

The ship event controller handles the same things for both player and AI.

It handles the ships status from hull to shield points as well as power up spawns on death and explosions. It also turns the shield mesh on/off and handles the status bars.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| GC | Public gamecontroller | Game controller script |
| GameUI | Public gameobject | In game UI reference |
| healthPowerUp | Public gameobject | Power UP prefab |
| shieldsPowerUp | Public gameobject | Power UP prefab |
| livesPowerUp | Public gameobject | Power UP prefab |
| shieldStrength | Public float | Max ship shields |
| currentShieldStrength | Private float | Current shield power |
| hullStrength | Public float | Maximum hull strength |
| currentHullStrength | Private float | Current hull strength |
| Shields | Publich mesh renderer | Mesh renderer for shield effect |
| Explosion | Public gameobject | Explosion prefab |
| hitTime | Private float | Time of last hit received |
| Shields | Private bool | Is shield active |
| regenDelay | Private float | Time before shields regenerate |
| healthBar | Public slider | Slider element to indicate health |
| shieldsBar | Public slider | Slider element to indicate shields |

Table - Varaibles: Ship Event Handler

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| Start | -Input:  NONE  -Output:  NONE | Finds the game controller and Game UI.  Also sets values to default when gameobject is pawned |
| Update | -Input:  NONE  -Output:  NONE | If the game is not paused handles shields effects and shield regeneration |
| OnHitByWeapon | -Input:  Float, float  -Output:  NONE | Sets new hit time enables shields if possible starts the damage calculation and handles death events |
| triggerShield | -Input:  float  -Output:  NONE | Turns off shield mesh when time has run out |
| dmgCalculation | -Input:  Float, float  -Output:  NONE | Calculate and apply damage to the ship |
| rocketLock | -Input:  NONE  -Output:  NONE | Toggle rocket lock image |
| getShipInformation | -Input:  NONE  -Output:  List<float> | Returns a list of ship information and weather its an AI ship or the player |
| adjustBars | -Input:  NONE  -Output:  NONE | Adjust bar indicators to match values |
| regenShields | -Input:  NONE  -Output:  NONE | Regenerate shields over time |
| healHull | -Input:  float  -Output:  NONE | Add health to hull |
| healShields | -Input:  float  -Output:  NONE | Add power to shields |
| addLife | -Input:  NONE  -Output:  NONE | Call add life function from game controller |
| selfDestroy | -Input:  Int, int  -Output:  NONE | Destroy this game object, spawn explosion prefab and spawn power-ups |
| spawnPoint | -Input:  NONE  -Output:  Vector3 | Calculate power up spawn point around ships location |
| getShipInformationToSave | -Input:  NONE  -Output:  Float[] | Returns ship information in order to be saved |
| setShipInformation | -Input:  String, string  -Output:  NONE | Sets ship information from given save |
| readTargetLock | -Input:  NONE  -Output:  Bool | Returns if target lock is on or off |

Table - Functions: Ship Event Handler

* Script :: PlayerTargetScanner

This is a script added to the scanner game object. IT sole purpose I to detect and handle enemies moving into range and giving this information to the weapon control

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| Wpc | public weaponControl | Reference to the players weapon control script |

Table - Variables: Player Target Scanner

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| Start | -Input:  NONE  -Output:  NONE | Called at the creation of the game object.  Sets the scripts collision detection to ignore anything that is not a ray cast target |
| OnTriggerEnter | -Input:  Collider  -Output:  NONE | If a AI ship enters the trigger the add function of the weapon script is called |
| OnTriggerStay | -Input:  Collider  -Output:  NONE | If a AI ship spawns inside the trigger the add function of the weapon script is called |
| OnTriggerExit | -Input:  Collider  -Output:  NONE | Calles the exitTrigger function as a coroutine |
| exitTrigger | -Input:  Collider  -Output:  IEnumerator | If a AI ship leaves the trigger the remove function of the weapon script is called after a delay of 2 second to prevent “flickering” if the object is right on the triggers edge |

Table - Functions: Player Target Scanner

* Script :: Camerascript

The camera script replaces the original AR idea.

The script creates a 3rd person follow camera.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| Player | Private GameObject | Contains a reference to the player gameobject |

Table - Variables:Camera Script

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| Update | -Input:  NONE  -Output:  NONE | Called every frame  Assigns the player variable if its empty.  Positions this object in the centre of the player |

Table - Functions: Camera Script

### AI

* Script :: AIShip

This script is the AI equivalent of the player control. It runs on a finite state machine which controls the state of the AI (patrol, engage). Depending on the current state of the AI a different code is executed.

The patrol part moves the ship back and forth between randomly selected waypoints across the map. While moving the ship will avoid obstacles and scan for enemies.

If an enemy is found the state switches to engage.

During engage the ship will follow and attack its target while moving around obstacles and scanning if target is still valid. To create a more “realistic” movement behaviour, the AI has a randomly chosen veer away function, which will make the ship turn away from its target to reengage a few moments later. Should the target get destroyed or leave the scanner area, the AI will return to its patrol movement.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| GC | Public game controller | Game controller script reference |
| FSMrunning | Private bool | Is state machine active |
| Patrol, Engage | Public enum state | All possible states for AI |
| State | Public state | Current state of AI |
| Alive | Public bool | Is AI alive |
| scanPerimiter | Public mesh renderer | Mesh of scan perimeter |
| patrolPoints | Public gameobject[] | Array of all map waypoints |
| waypointCounter | Private int | Current waypoint |
| Scan | Public AIScanner | AIScanner script |
| Target | Private Gameobject | Target gameobject |
| Destination | Private Vector 3 | Destination for navigation |
| Obstacle | Private transform | Detected obstacle |
| currentWaypoint | Private vector 3 | Current waypoint position |
| distError | Private float | Required distance from destination |
| Buffer | Private bool | Is destination buffered |
| ovDestination | Private bool | Is destination changed |
| escapeDir | Public List<Vector3> | All escape direction to avoid obstacle |
| PC | Private player control | Player control script |
| Ship | Public transform | AI ship transform |
| leftEngine | Public particle system | Particle system for left engine |
| rightEngine | Public particle system | Particle system for right engine |
| setStartSize | Private float | Start size for particles |
| setLifeTime | Private float | Start lifetime for particles |
| maxSpeed | Public float | Max AI speed |
| currentSpeed | Public float | Current AI speed |
| turnSpeed | Public float | AI turn speed |
| wingspan | Public float | AI ship wingspan |

Table - Variables: AI Ship

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| Start | -Input:  NONE  -Output:  NONE | Called at creation of object.  Finds needed scripts, sets default values and starts finite sate machine |
| finiteStateMachine | -Input:  NONE  -Output:  IEnumerator | AI State handler |
| update | -Input:  NONE  -Output:  NONE | If AI has patrol points and the FSM is off start FSM |
| Turn | -Input:  NONE  -Output:  NONE | Turns ship towards destination |
| Throttle | -Input:  float  -Output:  NONE | Moves ship towards destination and handles the engine particle systems |
| scanForEnemy | -Input:  NONE  -Output:  NONE | Checks for enemies in scanner range and sets state accordingly |
| Patrol | -Input:  NONE  -Output:  NONE | Patrol the systems waypoints while scanning for player and avoiding obstacles |
| Engage | -Input:  NONE  -Output:  NONE | Engages and follows the player until destroyed or loses target |
| avoidObstacle | -Input:  Vector 3, float  -Output:  NONE | Check for obstacles in ships path and finds a way around them |
| Rays | -Input:  Vector 3, float  -Output:  RaycastHit[] | Raycasts for avoiding function |
| navigateAroundObject | -Input:  Collider  -Output:  NONE | Finds escape directions around obstacle |
| findShortest | -Input:  NONE  -Output:  Vector 3 | Select shortest escape direction |
| setPartolPoints | -Input:  Gameobject[]  -Output:  NONE | Find all patrol points on the map and add them to the AI ship for use |
| veerAway | -Input:  NONE  -Output:  NONE | This function turns the AI away from the player at random making it look like attack runs rather then follow and shoot. |

Table - Functions: AI Ship

* Script :: AIWeaponControl

An automated weapon selection and firing script. The AI calculates a leading point in front of the target to shot at. This lead pointer can have a variance (amount depends on difficulty) to simulate miss targeting.

The weapons are chosen at random with decreasing likelihood for rockets and beam, when difficulty is not hard. This selection happens after a fixed period but does not actually swap every time.

The firing of the weapon has a build in delay making sure there are pauses between each shot fired. This time is a random number in a range. This range depends on the difficulty selected.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| GC | Public gamecontroller | Game controller |
| Scanner | Public AIScanner | AI scanner script |
| leadIndicator | Private vector3 | Position of lead indicator |
| targetSpeed | Private float | Speed of engaged enemy |
| Target | Private gameobject | Target game object |
| Rocket | Public gameobject | Rocket prefab |
| Cannon | Public gameobject | Blaster prefab |
| rocketPods | Public gameobject[] | Array of rocket locations |
| cannonPods | Public gameobject[] | Array of cannon locations |
| beamEmitter | Public gameobject[] | Array of beam emitter |
| currentRocketScript | Private rockets | Script of currently selected rocket |
| currentBeamScript | Private beamweapons | Script of currently selected beam emitter |
| currentCannonScript | Private cannonShotscript | Script of currently selected cannon shot |
| selectedRocketPod | Private int | Indicator for current rocket pod |
| selectedCannonPod | Private int | Indicator for current cannon pod |
| selectedBeamEmitter | Private int | Indicator for current beam emitter |
| Rocket\_reload | Public float | Reload time for rockets |
| rocketTarget | Public List,Gameobjet> | All enemies within scanner range |
| lockedTarget | Private gameobject | Locket target for rockets |
| currentTarget | Private int | Current target number |
| weaponRange | Private float | Max weapon range |
| readyToFire | Private bool | Can weapon be fired |
| lastShotTime | Private float | Time of last weapon shot |
| delayBetweenShots | Private float | Time between AI shots |
| maxDelayBetweenShots | Private float | Maximum delay |
| minDelayBetweenShots | Private float | Minimum delay |
| rocketWeaponRange | Private float | Range of rockets |
| cannonWeaponRange | Private float | Range of cannons |
| beamWeaponRange | Private float | Range of beam weapon |
| weaponToFire | Private int | Selected weapon |
| timeSinceSwitch | Private float | Time of last weapon switch |
| preSet | Private bool | Is AI pre-set |

Table - Varaibles: AI Weapon Control

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| Start | -Input:  NONE  -Output:  NONE | Finds game controller at creation of object and sets default values |
| Update | -Input:  NONE  -Output:  NONE | If the game is not pause the update function handles the AIs decision making on when and what to shoot |
| getLeadPoint | -Input:  NONE  -Output:  Vector3 | Calculates lead point to target |
| getDistanceToTargetPoint | -Input:  Vector3  , Vector3  -Output:  float | Calculates and returns distance to target point |
| fireCannon | -Input:  GameObject  -Output:  NONE | Fires cannon onto target from current cannon pod. Selects next cannon pod |
| FireBeam | -Input:  GameObject  -Output:  NONE | Fires beam weapon onto target from selected beam emitter  Selects next beam emitter |
| fireRocket | -Input:  GameObject  , GameObject  -Output:  NONE | Fires rocket connected to selected rocket pod.  Selects next rocket pod |
| spawnRocket | -Input:  GameObject  -Output:  NONE | Spawns a new rocket after reload delay in given rocket pod |
| decideOnWEapon | -Input:  NONE  -Output:  Int | Picks a random weapon to be used.  Probability depends on difficulty |
| isTargetInWeaponAngle | -Input:  Float, vector3  -Output:  bool | Reads if target is within given targeting angle |
| addVariance | -Input:  Vector3  -Output:  Vector3 | Adds a random variance to the lead point |

Table - Functions: AI Weapon Control

* Script :: AIScanner

This script has a similar function as the Player scanner.

If finds and sends the target(in this case only players) to the AI weapon script

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| Targetfound | Private bool | Boolean indicating if the player has been found |
| Target | Private Gameobject | Reference to the player game object |

Table - Variables: AI Scanner

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| Start | -Input:  NONE  -Output:  NONE | Called at creation of gameobject.  Set target found to false and ignores collisions with non raycast targets |
| OnTriggerEnter | -Input:  Collider  -Output:  NONE | Sets target to player and target found to true if player enters the scan zone |
| OnTriggerStay | -Input:  Collider  -Output:  NONE | Sets target to player and target found to true if player spawns in the scan zone |
| OnTriggerExit | -Input:  Collider  -Output:  NONE | Sets target to null and target found to false if player leaves the scan zone |
| getTargetFound | -Input:  NONE  -Output:  Bool | Returns targetFound Boolean |
| getTarget | -Input:  NONE  -Output:  Gameobject | Returns target gameobject |

Table - Functions: AI Scanner

* Script :: ShipEventController

SEE SHIPEVENT HANDLER IN PLAYER SECTION

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| SEE SHIPEVENT HANDLER IN PLAYER SECTION | | |

Table - Variables: Ship Event Handler

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| SEE SHIPEVENT HANDLER IN PLAYER SECTION | | |

Table - Functions: Ship Event Handler

### Game Controller

* Script :: GameController

The game controller script is the heart of the application. It regulates everything that happens in the game.

For menu selection it has a running state machine which selects the appropriate UI for the user interactions. The update method reads and handles every input besides player and weapon control.

The game controller is responsible for saving/loading a game, as well as deleting old stage elements and creating new games/stages for the player.

Player and AI death are also being handled here triggering respawns and “end of game” (player) or score increases and stage progression (AI).

The game controller script also manipulates in game UI elements to indicate remaining lives, score, stage and enemy count.

AI Spawn positions as well as map selection is being handled randomly with in the script (no AI can spawn closer than 5000 units from the player.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| Target | Public Gameobject | Gameobject of target |
| Alive | Private bool | Is application running |
| Paused | public bool | Is game paused |
| PlayerPrefab | Public Gameobject | Prefab of player |
| AIShipPrefab | Public Gameobject | Prefab of AI |
| Menus | Public List<Canvas> | List of all Menus |
| Highscore1 | Public text | Text in high score window |
| Highscore2 | Public text | Text in high score window |
| Highscore3 | Public text | Text in high score window |
| lifesIndicator | Public text | Play UI text indicator |
| scoreIndicator | Public text | Play UI text indicator |
| stageIndicator | Public text | Play UI text indicator |
| enemyIndicator | Public text | Play UI text indicator |
| Name | Public inputField | Input option for new high score |
| Start, difficultSelect, Pause, Play, Achievment, SetHighscore, Quit | Public enum Menu | List of all possible menu selections |
| currentMenu | Private Menu | Current menu state |
| currentCanvas | Private int | Current Canvas indicator |
| StarSystems | Public list<Gameobject> | List of all maps |
| currentMap | Private gameobject | Current map gameobject |
| Score | Private int | Int for player score |
| Lifes | Private int | Int for player lives |
| gameAlive | Private bool | Is game being played |
| Difficulty | Private int | Selected difficulty |
| curentStarSystem | Private int | Map count indicator |
| stageNumber | Private int | Current stage |
| enemyCount | Private int | Current enemy count |
| Total kills | Private int | Total kills since install – NOT IMPLEMENTED |
| sessionKills | Private int | Highest session kills since install – NOT IMPLEMENTED |
| powerupsPickedUp | Private int | Total power pick-ups since install – NOT IMPLEMENTED |
| SystemVisited | Private int | Total map seen since install – NOT IMPLEMENTED |
| AIStatus | Private string | Encoded AI information |
| PlayerOrientation | Private string | Encoded player information |
| High1 | Private string | Value of high score 1 |
| High2 | Private string | Value of high score 2 |
| High3 | Private string | Value of high score 3 |
| Scoreposition | Private int | Position on high score list |

Table - Variables: Game Controller

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| Start | -Input:  NONE  -Output:  NONE | Called at start of application.  Sets default values and start the menu state machine |
| Update | -Input:  NONE  -Output:  NONE | Handles user input on UI every frame.  Also handles end of game, end of stage and game information displayed on the UI |
| Pause | -Input:  NONE  -Output:  NONE | Set pause true |
| Unpause | -Input:  NONE  -Output:  NONE | Set pause false |
| FSM | -Input:  NONE  -Output:  IEnumerator | State machine handling UI selection and actions to be taken |
| newGame | -Input:  int  -Output:  bool | Starts a new game with given difficulty |
| saveGame | -Input:  NONE  -Output:  bool | Saves current game |
| loadGame | -Input:  NONE  -Output:  bool | Loads last saved game |
| loadHighscores | -Input:  NONE  -Output:  NONE | Loads all high scores into UI |
| saveHighscores | -Input:  NONE  -Output:  NONE | Saves high scores |
| setHighscores | -Input:  String, int, int  -Output:  NONE | Put new high score into list |
| checkHighscores | -Input:  int  -Output:  int | Checks position of current score and moves old high scores where needed |
| Deathplayer | -Input:  Vector3, Quaternion  -Output:  NONE | Player Death event handles life count and respawn |
| respawnPlayer | -Input:  Vector3, Quaternion  -Output:  IEnumerator | Spawn new player at last position after a 3 second delay |
| addLifes | -Input:  NONE  -Output:  NONE | Increases player lives by 1 |
| killedEnemy | -Input:  NONE  -Output:  NONE | Handles enemy death event.  Sets enemy count -1 |
| addToScore | -Input:  int  -Output:  NONE | Increases score depending on difficulty |
| encodeAIInformation | -Input:  NONE  -Output:  String | Encode AI information for save function |
| LoadAIInformation | -Input:  String,  Gameobject[]  -Output:  NONE | Loads all information for AI and spawns ships accordingly |
| encodePlayerPosition | -Input:  NONE  -Output:  string | Encode player information for save function |
| loadPlayerInformation | -Input:  string  -Output:  NONE | Loads all information for player and spawns ships accordingly |
| readDifficulty | -Input:  NONE  -Output:  Int | Return difficulty to calling script |
| createNewStage | -Input:  Int, int  -Output:  NONE | Create new stage |
| loadStarSystem | -Input:  Int  -Output:  Gameobject | Loads selected star system |
| spawnEnemy | -Input:  Int, gameobject[], gameobject  -Output:  NONE | Spawns enemies at random locations around the map |
| DestroyMap | -Input:  NONE  -Output:  NONE | Destroy all element of the current map, including player and AI |

Table - Functions: Game Controller

### Others

* Script :: BeamWeapon

The beam weapon consists of a line renderer for the visible part of the beam, plus a ray cast for hit detection.

This script builds and removes the beam step by step, giving it a motion effect. On hit with a collider or on reaching its maximum range the beam is removed.

The weapon can only fire if the beam has been completely removed.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| Sound | Public audio source | Audio source reference |
| Speed | Public float | Speed of beam |
| Gun | Public transform | Beam emitter transform |
| target | Private vector3 | Crosshair position |
| Dist | Public float | Maximum fire range |
| Firing | Public bool | Is weapon firing |
| Removing | Public bool | Is beam being removed |
| GC | Private game controller |  |
| Hitlength | Private float | Distance to hit point |
| Counter | Private float | Step counter |
| X | Private float | Point on beam |
| LR | Private line renderer | Line renderer component |
| pointOnLine | Private vector3 | Position of line point |
| Hit | Private RaycastHit | Hit information |
| readyToFire | Private bool | Can weapon be fired |
| damageHull | Private float | Weapon damage to hull |
| damageShields | Private float | Weapon damage to shields |

Table - Variables: Beam Weapon

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| Start | -Input:  Collider  -Output:  NONE | Assign game controller, sound source and line renderer  Set values for damage and ready to fire |
| update | -Input:  Collider  -Output:  NONE | Either builds or removes the beam if game is not paused |
| onFire | -Input:  Vector3  -Output:  NONE | Public call from weapon control.  If weapon can be fired, set firing towards target into motion |
| OnHit | -Input:  GameObject  -Output:  NONE | If Ray cast hits a collider.  Apply damage to object and prepare for beam removal |
| Fire | -Input:  Vector3  -Output:  NONE | Gradually increases the beam and ray cast length until either a hit is detected or the max weapon range has been reached |
| removeBeam | -Input:  NONE  -Output:  NONE | Inverse process of fire function.  Removing the beam step by step till hit point |
| getMaxFireDistance | -Input:  NONE  -Output:  Float | Return maxFireDistance to calling script |

Table - Functions: Beam Weapon

* Script :: CannonShotsScript

The actual cannon shot is a rigidBody prefab with a particle system. The firing script

Turns the blast to look at its targets current location and applies an impulse force to shoo tit to that spot.

The blast gets destroyed on collision or after reaching its maximum weapon range.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| Speed | Public float | Weapon speed |
| Weapon | Public rigidBody | Physics body of gun shot |
| Range | Public float | Max range of weapon |
| Start | Private Vector3 | Start position of shot |
| fireFrom | Private gameobject | Firing parent object |
| damageShields | Private float | Damage a hit deals to the targets shields |
| damageHull | Private float | Damage a hit deals to the targets hull |

Table - Varaibles: Cannon Shot Script

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| Start | -Input:  NONE  -Output:  NONE | Ignores collision with non-ray cast targets.  Rotates the prefab to look at the target and set the damage for hull and shields |
| Update | -Input:  NONE  -Output:  NONE | Destroys the object if it has passed its maximum range |
| onFire | -Input:  Vector3, GameObject  -Output:  NONE | Set fireFrom and looks at target.  Sets the start position before applying an impulse force to the object propelling it forward.  Play weapon sound |
| OnTriggerEnter | -Input:  Collider  -Output:  NONE | If blast collides with object.  If its not the firing object destroy the shot and apply damage to the ship |
| getMaxFireRange | -Input:  NONE  -Output:  float | Returns max weapon range to calling script |

Table - Functions: Cannon Shot Script

* Script :: Rockets

The rocket implemented into the game is a homing missile, finding its way towards the target. The script detaches the rocket form the ship, move it downwards before activating the movement of the rocket. After the rocket is a certain distance away from the firing ship it starts homing onto the target until it hits are the flight timer expires.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| sound | Public Audio source | Audio source of the rocket object |
| Detonator | Public float | Time until self-detonation |
| damageHull | Private float | Damage dealt to the hull |
| damageShields | Private float | Damage dealt to the shields |
| Speed | Public float | Rocket speed |
| rotSpeed | Public float | Rocket turn rate |
| Fired | Public bool | Is fired |
| GC | Private game controller | Game controller script |
| Detached | Private bool | Is detached from carrier |
| Homing | Private bool | Is homing onto target |
| Range | Private float | Distance from player before homing |
| maxRange | Private float | Maximum fire range |
| Target | Private transform | Transform of locket target |
| Player | Private transform | Transform of carrier |
| Engine | Public particle system | Particle system for engine |
| Glow | Public particle system | Particle system for engine glow |
| targetEventController | Private ShipEventController | Target event controller |

Table - Variables: Rockets

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| start | -Input:  NONE  -Output:  NONE | Assigns the game controller and audio source.  Reads the player position and set the default values for the variables |
| Update | -Input:  NONE  -Output:  NONE | Either detaches, moves away or homes the rocket if the game is not paused |
| OnHit | -Input:  GameObject  -Output:  NONE | Apply damage to hit object and destroy rocket |
| fireRocket | -Input:  Gamobject  -Output:  NONE | If rocket has a valid target, set target values, fire the rocket and play the sound |
| Detach | -Input:  NONE  -Output:  NONE | Detaches the rocket from the carrier and starts the engine |
| homingOnTarget | -Input:  NONE  -Output:  NONE | As long as target is valid moves and rotates towards target until within 1 unit=>apply hit |
| moveAwayFromPlayer | -Input:  NONE  -Output:  NONE | After detaching moves rocket away before starting to home on target |
| getMaxFireRange | -Input:  NONE  -Output:  Float | Return max fire range to calling script |

Table - Functions: Rockets

* Script :: PowerUps

The power up script handles trigger events when a ship passes through one of the power ups. This applies its effect to the ship that pick it up.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| GC | Public game controller | Contains reference to the game controller script |
| Health, shields, lifes | Public enum type | List of all possible power ups |
| Type | Private type | Type of this power up |
| shieldValue | Private float | Heal value for shields |
| hullValue | Private float | Heal value for hull |

Table - Variables: Power Ups

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| Start | -Input:  NONE  -Output:  NONE | Called at creation of the game object.  Sets the type of power up to the correct selection |
| OnTriggerEnter | -Input:  Collider  -Output:  NONE | Destroys the gameobject and applies the appropriate effect to the ship that picked it up |

Table - Functions: Power Ups

* Script :: StarField

This script creates and displace static stars(particles) inside a sphere, covering the map.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| Stars | Private ParticleSystem.Particle[] | Array of all particles to be displayed in this particle system |
| Starfield | Private ParticleSystem | The particle system to show the particles with |
| maxStars | Private int | Number of maximum particles |
| starSize | Private flat | Size of particles |
| Distance | Private float | Max distance from centre |

Table - Variables Star Field

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| Start | -Input:  NONE  -Output:  NONE | Called at creation of the gameobject.  Sets particle values, gets the particle system component and create the particle field |
| Update | -Input:  NONE  -Output:  NONE | Called every frame.  If no particle field is formed, creates a new particle field.  Spawns all particles on the map |
| CreateStarField | -Input:  NONE  -Output:  NONE | Creates a field of particles of size starSize with random positions within a sphere with radius Distance. |

Table - Functions: Star Field

* Script :: NebulaField

Similar to the StarField this script creates static nebulae, thereby saving resources to make the game run smoother, since no particles have to be emitted or particle movement calculated.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| Stars | Private ParticleSystem.Particle[] | Array of all particles to be displayed in this particle system |
| Starfield | Private ParticleSystem | The particle system to show the particles with |
| maxStars | Private int | Number of maximum particles |
| starSize | Private flat | Size of particles |
| Distance | Private float | Max distance from centre |

Table - Variables: Nebula Field

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| Start | -Input:  NONE  -Output:  NONE | Called at creation of the gameobject.  Sets particle values, gets the particle system component and create the particle field |
| Update | -Input:  NONE  -Output:  NONE | Called every frame.  If no particle field is formed, creates a new particle field.  Spawns all particles on the map |
| CreateStarField | -Input:  NONE  -Output:  NONE | Creates a field of particles of size starSize with random positions within a sphere with radius Distance. |

Table - Functions: Nebula Field

* Script :: Nebula

The Nebulae on the map are hazardous objects damaging ships within them. This script applies the effect of each separate nebula type to any ship that is within its effect area.

These effects are being triggered every 3 seconds.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| nebulaType | Private type | Contains type of this nebula |
| Corrosion, energy, plasma | Private enum type | Collection of possible nebula types |
| dmgHull | Private float | Damage the nebula will deal to the hull |
| dmgShields | Private float | Damage the nebula will deal to the shields |
| Delay | Private float | Delay between damage ticks |
| lastTrigger | Private float | Last tick time in seconds |
| Ships | Private List<Gameobjects> | List containing all ships within the nebula |

Table - Variables: Nebula

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| Start | -Input:  NONE  -Output:  NONE | Called at creation of gameobject.  Set the damage values and the nebula type to the correct selection |
| Update | -Input:  NONE  -Output:  NONE | Called every frame.  Applies the nebula effect when the tick delay has expired |
| OnTriggerEnter | -Input:  Collider  -Output:  NONE | Adds a ship to the list when it enters the nebula |
| OnTriggerExit | -Input:  Collider  -Output:  NONE | Removes a ship form the list when it leaves the nebula |
| applyNebulaEffect | -Input:  NONE  -Output:  NONE | Send damage message to all ships currently within the nebulas effect range and set the lastTrigger to now |
| findWantedParent | -Input:  Gameobject  -Output:  Gameobject | “FOR AR PURPOSES”  A recursive algorithm finding a wanted parent object.  Root was not working due to everything being a child of the target image |

Table - Functions: Nebula

* Script :: PlanataryObjectMovingScript

This script handles the rotation of map objects in 3d Space.

* + Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| GC | Public GameControler | Reference to the game controller script |
| rotSpeed | Public float | Rotation speed for object |

Table - Variables: Planetary Object Moving Script

* + Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| Start | -Input:  NONE  -Output:  NONE | Finds and assigns the game controller script |
| Update | -Input:  NONE  -Output:  NONE | Rotates the object around its Y-axis with speed\*delta time |

Table - Functions: Planetary Object Movement Script

* Not Implemented

This script was supposed to add indicators to the game screen, marking any enemy currently inside the camera view with squares and indicate all other enemies with arrows on the side of the screen.

Due to implementation issues this script never worked properly and was therefore removed from the current game.

* + Script :: Indicator
    - Variables:

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| targetSquare | Public game object | UI Sprite to mark enemy fighter |
| targets | Private List<GameObject> | List of all enemies on the map |

Table - Variables: Indicator

* + - Functions:

|  |  |  |
| --- | --- | --- |
| Function | Input/Output | Description |
| update | -Input:  NONE  -Output:  NONE | Run through all fighters and display indicator where appropriate. Also removes invalid targets from the list |
| DisplayIndicator | -Input:  GameObject  -Output:  NONE | Display indicator on screen on position of enemy fighter if visible |
| createIndicator | -Input:  GameObject  -Output:  GameObject | Create instance of UI sprite to display and return the created object |

Table - Functions: Indicator

# Results

The final version of the game can be viewed in the game play video provided and by building/copying the project/.apk onto an android device.

The following screenshots were taken out of the editors play option.

They show map components and visual elements.

|  |  |
| --- | --- |
| Figure - Player inside nebula | Figure - Player shooting blaster |
| Figure - Approaching 2 enemies | Figure - Energy Nebula |
| Figure - Active ship shields | Figure - Planet system + Nebula |
| Figure - Player firing a rocket | Figure - Planet system |

Table - InGame Screenshots

## Bugs/Problems

* Performance issues on mobile devices. The nebulas make the game lag sometimes
* AI sometimes enters an orbiting motion around its destination which prevents advancing of the script.
* Killing an AI sometime counts as 2 kills instead of 1

## Not Implemented Features

The game has a few features which couldn’t be implemented in time or could not be coded to function properly and where therefor removed from the final game. This part will outline all of these cut features and explain what went wrong or what would be needed to implement.

* Achievements:

The Achievement system is nothing complicated to implement but due to problems with the maps and AI had to be cut.

The idea included a list of achievements next to the high score which would be obtained by completing tasks or by reaching milestones (i.e. kill X enemies, reach stage X, pick up X power-ups, play for X hours, etc.)

Each achievement would have had a name and a according picture assigned to it.

Some elements can still be found in the game controller script, marked with NOT IMPLEMENTED.

* Target Indicators:

The current version of the game is missing a feature to better navigate the map and find enemies. A target indicator was supposed to fill this function but the placement on the UI caused multiple problems to occur, one of which was that while the sprite was spawned in the right spot to indicate the target (visible and checkable in the editor) it did not show up in the actual game. All attempts to fix the script before the deadline failed.

The indicator script is still located in the project and can be view.

* AR mode:

This project was started with the idea in mind to create a AR mode to make the player look from the outside. This was part of the game for a long time. Ultimately there were issues regarding the spawning inside the target image which would require a slow working recursive algorithm to find the correct parent object, as well as an issue which occurred with turned off mesh renderers inexplicably turning on by default when prefabs where spawned into the game making the game unplayable since no ship could be seen.

* Different Ship types:

This would have been a simple task of creating more prefabs of ships with different speed, turn rate, hull/shields, weapon strengths and weapon numbers/types.

Due to time issues there was no time to model and texture new ships and no free ones could be found on the Asset store.

* More detailed weapon systems:

The current weapon system contains a total of 3 weapon types (Blaster, beam, homing-rocket). Originally more weapons like machine guns, sidewinder rockets or smart drones were to be implemented for bigger more powerful ship classes. With the scrubbing of multiple ship types this idea was also removed.

Implementing this would mean to primarily create prefabs for bullets or rockets, while the implementation of a smart drone (search and destroy) would take creating a new script for drone control.

* RPG mode/Explorer mode: A Story mode giving the player choices on where to go, to gather resources to build better ships and weapons, Story

# Annex

## Table of Figures

[Figure 1 - Corrosion Nebula 4](#_Toc468720896)

[Figure 2 - Energy Nebula 4](#_Toc468720897)

[Figure 3 - Plasma Nebula 5](#_Toc468720898)

[Figure 4 - Health Power Up 5](#_Toc468720899)

[Figure 5 - Shield Power Up 5](#_Toc468720900)

[Figure 6 - Life Power Up 5](#_Toc468720901)

[Figure 7 - Ship without shields 6](#_Toc468720902)

[Figure 8 - Ship with shields 6](#_Toc468720903)

[Figure 9 - Rocket 6](#_Toc468720904)

[Figure 10 - Main UI 7](#_Toc468720905)

[Figure 11 - Select difficulty 7](#_Toc468720906)

[Figure 12 - Pause game 7](#_Toc468720907)

[Figure 13 - High score 7](#_Toc468720908)

[Figure 14 - New high score 7](#_Toc468720909)

[Figure 15 - Fire button 8](#_Toc468720910)

[Figure 16 - Blaster cannon 8](#_Toc468720911)

[Figure 17 - Beam weapon 8](#_Toc468720912)

[Figure 18 - Rockets 8](#_Toc468720913)

[Figure 19 - Rocket targeting 8](#_Toc468720914)

[Figure 20 - Pause 8](#_Toc468720915)

[Figure 21 - Score Indicator 8](#_Toc468720916)

[Figure 22 - Remaining Lives 8](#_Toc468720917)

[Figure 23 - Player inside nebula 37](#_Toc468720918)

[Figure 24 - Player shooting blaster 37](#_Toc468720919)

[Figure 25 - Approaching 2 enemies 37](#_Toc468720920)

[Figure 26 - Energy Nebula 37](#_Toc468720921)

[Figure 27 - Active ship shields 37](#_Toc468720922)

[Figure 28 - Planet system + Nebula 37](#_Toc468720923)

[Figure 29 - Player firing a rocket 37](#_Toc468720924)

[Figure 30 - Planet system 37](#_Toc468720925)

## Table of Tables

[Table 1- Corrosion Nebula 4](#_Toc468720926)

[Table 2 - Energy Nebula 4](#_Toc468720927)

[Table 3 - Plasma Nebula 5](#_Toc468720928)

[Table 4 - Health Power-Up 5](#_Toc468720929)

[Table 5 - Shields Power-Up 5](#_Toc468720930)

[Table 6 - Lives Power-Up 5](#_Toc468720931)

[Table 7 - User Interface 7](#_Toc468720932)

[Table 8 - Player Controls 9](#_Toc468720933)

[Table 9 - Variables: Player Control 11](#_Toc468720934)

[Table 10 - Functions: Player Control 12](#_Toc468720935)

[Table 11 - Variables:Target Control 12](#_Toc468720936)

[Table 12 - Functions: Target Control 12](#_Toc468720937)

[Table 13 - Variables: Weapon Control 13](#_Toc468720938)

[Table 14 - Functions: Weapon Control 15](#_Toc468720939)

[Table 15 - Varaibles: Ship Event Handler 15](#_Toc468720940)

[Table 16 - Functions: Ship Event Handler 17](#_Toc468720941)

[Table 17 - Variables: Player Target Scanner 17](#_Toc468720942)

[Table 18 - Functions: Player Target Scanner 17](#_Toc468720943)

[Table 19 - Variables:Camera Script 18](#_Toc468720944)

[Table 20 - Functions: Camera Script 18](#_Toc468720945)

[Table 21 - Variables: AI Ship 19](#_Toc468720946)

[Table 22 - Functions: AI Ship 20](#_Toc468720947)

[Table 23 - Varaibles: AI Weapon Control 22](#_Toc468720948)

[Table 24 - Functions: AI Weapon Control 23](#_Toc468720949)

[Table 25 - Variables: AI Scanner 23](#_Toc468720950)

[Table 26 - Functions: AI Scanner 24](#_Toc468720951)

[Table 27 - Variables: Ship Event Handler 24](#_Toc468720952)

[Table 28 - Functions: Ship Event Handler 24](#_Toc468720953)

[Table 29 - Variables: Game Controller 25](#_Toc468720954)

[Table 30 - Functions: Game Controller 28](#_Toc468720955)

[Table 31 - Variables: Beam Weapon 28](#_Toc468720956)

[Table 32 - Functions: Beam Weapon 29](#_Toc468720957)

[Table 33 - Varaibles: Cannon Shot Script 30](#_Toc468720958)

[Table 34 - Functions: Cannon Shot Script 30](#_Toc468720959)

[Table 35 - Variables: Rockets 31](#_Toc468720960)

[Table 36 - Functions: Rockets 32](#_Toc468720961)

[Table 37 - Variables: Power Ups 32](#_Toc468720962)

[Table 38 - Functions: Power Ups 32](#_Toc468720963)

[Table 39 - Variables Star Field 32](#_Toc468720964)

[Table 40 - Functions: Star Field 33](#_Toc468720965)

[Table 41 - Variables: Nebula Field 33](#_Toc468720966)

[Table 42 - Functions: Nebula Field 34](#_Toc468720967)

[Table 43 - Variables: Nebula 34](#_Toc468720968)

[Table 44 - Functions: Nebula 35](#_Toc468720969)

[Table 45 - Variables: Planetary Object Moving Script 35](#_Toc468720970)

[Table 46 - Functions: Planetary Object Movement Script 35](#_Toc468720971)

[Table 47 - Variables: Indicator 35](#_Toc468720972)

[Table 48 - Functions: Indicator 36](#_Toc468720973)

[Table 49 - InGame Screenshots 37](#_Toc468720974)

## References

* Standart Assest (from the Asset store)
* Different UI sprites for Player Control(google image search)
* Purple Nebula(Asset Store)
* Blaster audio (<http://soundbible.com/472-Laser-Blasts.html>)
* Beam weapon audio (http://www.soundboard.com/sb/sound/71572)
* Rocket audio (<http://soundbible.com/1794-Missle-Launch.html>)
* Vuforia