User Documentation

Game Object Inheritance Tree:

Created an inheritance tree to define the different categories of objects there are in the game, defining behaviors common to all of that classes type, creating attributes common to all of that class's type and creating methods useful for that class's type.

The hierarchy is:

SpaceObject – Common to all objects in the game, inherits from MonoBehavior class script.

NonInteractiveObject – Objects that do not use physics, not having a RidgidBody.

Item – Objects that Player's can pick up and use.

InteractiveObject – Objects that use physics, having a RidgidBody and Collider

DestructableObject – Objects that have health, thus can die from damage.

Player – The player's ship, updates biased on the player's inputs.

IndestructableObject – Objects that don't have health but use physics.

DestructableObjects, IndestructableObjects and NonInteractiveObjects have methods that are called when the object is created, destroyed, each update and when it collides with other objects. These are used to describe the game object's behavior at the appropriate times.

Items have methods for when they are picked-up, dropped and each update they are held.

DestructableObjects:

Asteroid – Breaks apart into smaller asteroids when dealt enough damage. Dies if its small enough that the damage done is greater than its health.

Blob – Breaks apart into smaller pieces when dealt damage. Combines with other Blobs when they touch. Dies if its small enough that the damage done is greater than its health.

HomingMine – Locks on to a DestructableObject not on the same team if a target gets too close, then moves towards the taget. Looses the target if its gets too far away. When it has no target, it stops. Explodes, dealing damage, when it touches an InteractiveObject.

HomingMissile – Locks on to the first DestructableObject not on the same team that is straight in-front of it then moves towards the target. When it has not target, it moves forward. Explodes, dealing damage, when it touches an InteractiveObject.

LazerShooter – Locks on to a DestructableObject not on the same team if a target gets too close. Then it follows the target, keeping some distance, and periodically shooting at where it will be given its current velocity. Does not change velocity if it has no target.

MineLayer – Moves forward, periodically laying HomingMines behind it. Turns away from InteractiveObjects on a different team if they get too close.

Rammer – Finds the closest DestructableObject not on the same team and moves towards where it will be biased on its current velocity, then finds the new closest DestructableObject not on the same team and keeps doing this. Deals damage to DestructableObjects not on the same team when it collides with them.

RandomTurner – Turns one direction for a random amount of time, then turns the other direction for a random amount of time, ect. Accelerates forward until it reaches it maximum speed. Deals damage to DestructableObjects not on the same team when it collides with them.

RubberyDebris – Moves in a direction (not changing its speed) until it hits an InteractiveObject, then moves in the mirror direction at the same speed. Deals damage to DestructableObjects not on the same team when it collides with them.

SlowTurner – Finds the closest DestructableObject not on the same team and rotates towards it a certain amount and accelerates forward. Deals damage to DestructableObjects not on the same team when it collides with them.

IndestructableObjects:

GravityWell – Pulls all InteractiveObjects towards it biased on its gravity and the squared distance between it and the other InteractiveObject. When an InteractiveObject touches it, it will push it away from it and deal damage to it if it is on a different team.

LazerShot – Moves forward at its starting speed until it hits an InteractiveObject, when it then deals damage to what it hit then gets destroyed.

NonInteractiveObjects:

LazerBeam – Attaches to a given SpaceObject at a given relative position at a given relative angle. Extends to a set length and damages any DestructableObjects it collides with.

SpaceDust – Tiles a space dust texture across the Level to give the Player something non-moving to reference his/her movement against. Moves when it gets too far from the center of the play area but always enough that its tiles lineup. Does not interact with anything.

Items:

HomingMines – Allows the player to create a HomingMine every so often and only a certain number of the mines can be in the level at once.

HomingMissles – Allows the player to create a HomingMissle every so often.

LazerSword – Crates a LazerBeam attached to the player by pressing its button. Holding down its button rotates the LazerBeam around the player. Double pressing its button removes the LazerBeam.

MultiShooter – Allows the player to shoot multiple LazerShots at once, every so often, evenly spaced over an angle. Holding down its button will increase the spread angle.

Level:

Level is an abstract class that keeps track of everything in the current Level, like game objects in the Level, Level settings, duration, gamebounds, ect. Implements features common to all levels, like create, restartLevel, nextLevel, save, saveReplay, getLevel, loadLevel and loadReplay. Also provides helper methods for things commonly done by Levels, like createObject. Derived classes have to implement abstract methods createLevel and updateLevel to implement things specific to that Level. createLevel is called once, at level creation and is mainly used to add the objects that are in the Level. updateLevel can be used for logic specific to the level and is called every game update.

CameraController:

The CameraController class Controls the game's orthographic camera. It allows the players to zoom in and out, makes sure all players are on screen and makes sure the screen does not go outside of the game's bounds.

Controls, PlayerControls, PlayerInput and Key:

The Controls class is a singleton class that keeps track of each of the player's controls and inputs with multiple PlayerControls instances. It sets the default controls and deals with saving and loading the controls and inputs to and from files. A certain player's controls can be accessed from the players array using (player's number – 1) as the index.

The PlayerControls class keeps track of one player's controls and inputs. It has methods to update the inputs from the player or from a string (used for replay). It also has methods make the players controls into strings and set the controls form a string. It also provides assessors to easily get the player's current inputs (Capital named assessors represent keys just pressed and lowercase represent keys held down). This class also keeps a list containing the history of the Player's input for the current Level.

The PlayerInput class represents one player's input for one frame. It has a constructor to create the PlayerInput from a string and implements the ToString method to represent the player's inputs as a string.

The Key class was made to have a unified way to deal with Unities KeyCodes and Axises, since Unity uses them in different ways. So, it represents a KeyCode or Axis and can get a float representing an Axis value (buttons will return 1 or 0) or a bool pressed value (an Axis will return 1 after it reaches a certain threshold, 0 otherwise). It has a static method to find the activated key (KeyCode or Axis).

IngameInterace and PlayerPanel:

The IngameInterface sets text for the current Level's duration, current Level's number and current Level's object's progress. It allows messages to be displayed to the user for a given number of seconds by calling the static method displayMessage(string message, float durationSecs). It activates or deactivates PlayerPanels biased on which players are currently playing the level.

The PlayerPanel sets the heal text and the health-bar biased on its player's health and maxHealth. For each item of the player it represents, it will show the item image and show the name of the key set to that item.

Player Controls:

The default controls are listed at the end of this paper. The Player's ship is able to move using the Forward, Backward, Straif Right and Straif Left keys. There is a setting to allow the movement to be relative, which will cause the Player's ship to in a direction relative the direction it is pointed, or absolute, which will cause the Player's ship to move in the direction on the screen directly. The Player's ship can turn using the Turn Left and Turn Right keys if input setting is set to turn or point in a direction using the Turn Left, Turn Right, Turn Up, Turn Down keys if the input setting is set to point.

Using a particular item's key will use the item. Different Items are used in different ways; some do things when you hold them down, some when you first press it, some when you press it twice quickly. Holding down Pickup/Drop when moving over an Item will pick it up, putting it in the first available item slot. Holding down Pickup/Drop and holding down an Item key when moving over an Item will pick it up in that slot. Holding down Pickup/Drop and holding pressing an Item key with an Item in it will drop the Item.

Pressing the Shoot key will shoot the Player's ship's laser, creating a LazerShot in front of the Player's ship.

Pressing the Pause key will bring up the Pause Game menu, where the player can resume the Level, restart the Level, go to the Options Menu or quit the Level (and go back to the MainMenu).

Pressing the ZoomIn and ZoomOut keys will zoom the game's camera in or out.

General Game Behavior:

All InteractiveObjects will collide with each other and has physics applied to them using Unity's physics engine. This physics engine also updates their position biased on its velocity that it also controls. This physics engine also updates its rotation biased on its angular velocity that it also controls. Velocity and angular velocity can be modified by scripts without problem but modifying rotation and position directly can cause problems with the physics engine.

When SpaceObjects collide with each other, they may interact with each other in their scripts in different ways, like by dealing damage to the other. If any DestructableObject (SpaceObjects with health) has a health of 0 or less, it will be destroyed and removed from the Level. When all Player's are

destroyed, the Level is lost and the GameOver screen is brought up, where the player can save the replay, restart the level or return to the main menu.

When all enemy DestructableObjects are destroyed, by default (Levels can have different win conditions) the Level is completed, an auto-save is created and the LevelComplete menu is displayed, where the user can save his progress, save the replay, start the next Level or return to the main menu. When the last Level has been completed, the Victory menu is displayed instead, with the same options except start the next Level.

Default Controls

P1:	
Forward	W
Backward	S
Straif Left	A
Straif Right	D
Turn Up	P
Turn Down	Semicolon
Turn Left	L
Turn Right	Quote
Item 1	E
Item 2	O
Item 3	R
Item 4	I
Item 5	F
Item 6	K
Item 7	G
Item 8	J
Item 9	C
Item 10	Comma
Pickup/Drop	Right Shift
Shoot	Spacebar
Pause	Escape
Zoom In	Left Shift
Zoom Out	Left Control
P2:	**(Xbox360 Controller)**
Forward	+Y Axis
Backward	-Y Axis
Straif Left	+X Axis
Straif Right	-X Axis
Turn Up	+Y Axis2
Turn Down	-Y Axis2
Turn Left	+X Axis2
Turn Right	-X Axis 2
Item 1	A Button
Item 2	B Button
Item 3	X Button
Item 4	Y Button
Item 5	DPad Down

Item 6	DPad Up	Pickup/Drop	Right Trigger
Item 7	DPad Left	Shoot	Left Trigger
Item 8	DPad Right	Pause	Start
Item 9	Left Stick Press	Zoom In	Left Bumper
Item 10	Right Stick Press	Zoom Out	Right Bumper

Temporary Options Controls (until the options menu is created)

F1: Toggle how the edge of the screen is managed.

F2: Toggle the controller names from Xbox360 to generic.

F3: Toggle relative/direct movement for all players.

F4: Toggle turn/point for all players

F5: Lower Interface Alpha

F6: Increase interface Alpha

F7: Lower HealthBar Alpha

F8: Increase HealthBar Alpha