# Cover Page

# 

# Table of Contents

# Introduction / Game feature description

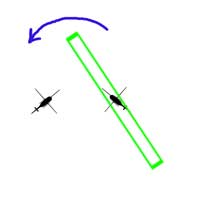
**Objective**

**Controls**

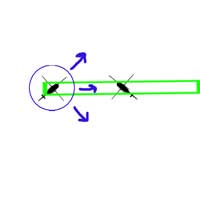
**The Missile Intercept**

**bla bla**

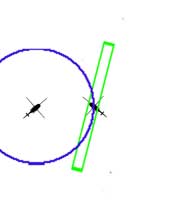
**The radar system**



The player/owner craft sweeps the area with a bounding box.  This meant to represent a radar sweep of the area.



If the sweep collides with another craft, that craft spawns a bounding sphere that will continue to expand until it expires.  This is the represent the ping back.



Once the owner/player craft receives the ping they begin to log the origin and signature of the received ping and from this the tracking process can begin.

**The multiplayer**

Master server

UDP lan broadcast / listen discovery

# Works used in accomplishing this project

Algorithm for finding the thing,

Unity networking tutorial

UDP multicasting

Unity Detonator pack (avail and free to use from the unity asset store

the models = Blend swap CC Zero

# Originality Statement

add and sign

# Weekly project forms

Issues faced

**Unit testing**

One of the biggest issues faced surrounded the fact that unity classes than inherit from monobehaviour cannot be instantiated as the “new” keyword is not supported. This made testing difficult and in some cases impossible. A solution I proposed and tried was to abstract as much behaviours and states and put them in a concrete class which my base monobehaviour class could then instantiate. Having a concrete class meant that it would be possible to create unit tests. Unfortunately a lot of features used in my application made use of monobehaviour and put them out of the scope of being tested completely. Examples being the networking and Radar systems developed in the application.

**Version Control**

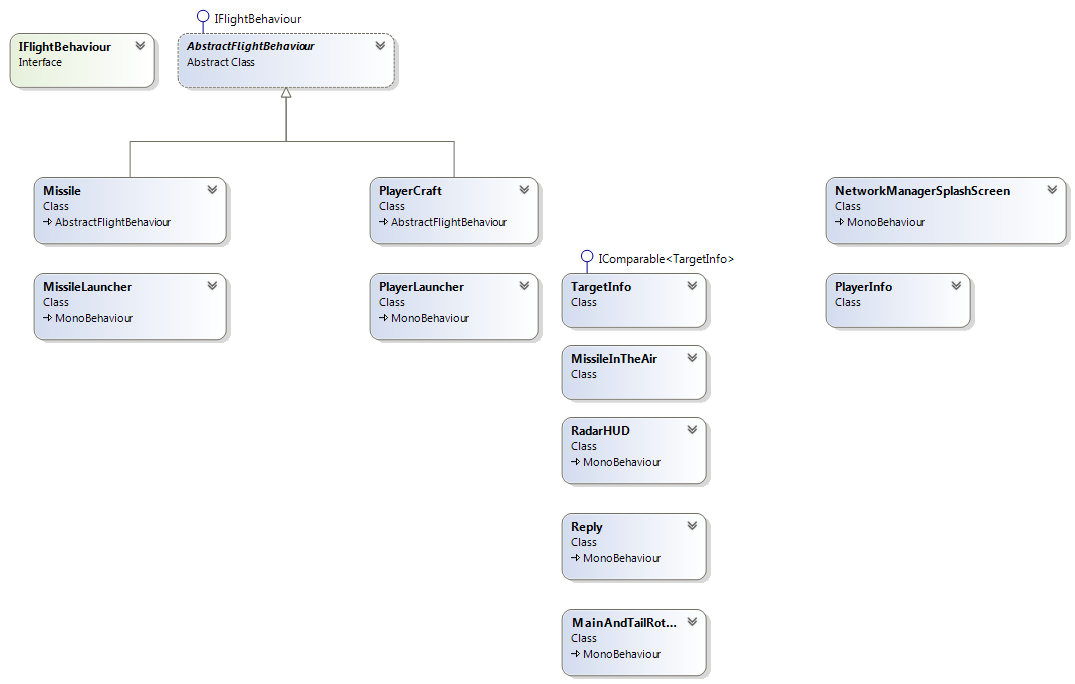
Unity has made the use version control difficult when it comes to their indie package making it very difficult to collaborating on assets. The issue surrounds the fact that all assets are stored as binaries so when it comes to version control it is impossible for the VC system to distinguish who made what changes and will only overwrite the previous change. With careful management it is possible to work together having specific rules on who can work on what.

Unity pro is required to force text so full version control is possible. However for the indie package they introduced meta files that are stored alongside each file allowing the version control system know that are changes.

For my project I chose to use GIT for version control

(Source control, Mono (crashing) and Visual studio, testing, importing projects, RegisterHost FPS drop, Multicasting Vbox issue)

# Class Diagram



# The Code

# Screenies



