User Guide

To install the game, simply extract the files into a single folder and click the main executable.

The game is a simple 2D competitive dogfighter, wherein two players attempt to shoot each other down. A player wins when he has taken the other player's lives.

Controls:

Player One

Forward Thrust - Up Cursor Key/Up D-Pad (X360)
Reverse Thrust - Down Cursor Key/Down D-Pad (X360)
Turn Left - Left Cursor Key/Left D-Pad (X360)
Turn Right - Right Cursor Key/Right D-Pad (X360)
Shoot - Space/"A" button (X360)

Player Two

Forward Thrust - W Key/Up D-Pad (X360)
Reverse Thrust - S Key/Down D-Pad (X360)
Turn Left - A Key/Left D-Pad (X360)
Turn Right - D Key/Right D-Pad (X360)
Shoot - Shift/"A" button (X360)

Known Issues:

Most things implemented in the game at present work. However, not all features are implemented. There is no opening menu or "win" screen, though the victory is recorded in the debug output. Sound is unimplemented.

Maintenance Guide

RVisualize, the Visualisation component.

The purpose of this component is to handle the loading and drawing of textures, including reading a list of texture data from a text file.
This component uses the singleton method to make itself accessible to all components, though the majority of its interactions are via the World component (detailed below) and by the REntity Render() method. The majority of functions would be usable in any game that uses the HAPI API, though some extension may be needed depending on the specific game design.
Classes and methods used in this component:
RVisualize: The main class of the Visualisation component. It is a Singleton and as such only one instance of it may exist at any one time.
The largest structures in the component are _spriteContainer, which is a STL Vector containing pointers to RSprite, and _spriteNameContainer, which is a map of strings to integers, specifically the ID numbers of RSprites within the prior structure. I have chosen to set sprite access up in this manner to aid user-friendliness as well as machine-friendliness, as you can access the ID directly in constant time, or via the string in logarithmic+constant time, which also massively aids readability of datafiles while keeping most routine access of the sprite container at a high speed.
Only the visualisation component interacts with RSprites directly, loading them into memory, unloading them at destruction, and using their draw methods.
The visualisation component is black boxed and should remain such to maintain portability.

RWorldModel, the World component.

The world component handles all entities, as well as managing user input through the RInput class. It is also a Singleton, though this is primarily to make sure multiple instances of the world are not created rather than to facilitate access to the world's components from all areas.
Most of the classes used in the World component are derived from REntity, which encompasses all objects that have position, including cameras (used to track the two players seperately), although many functions are under the jurisdiction of a variety of Manager classes, including the Bullet Manager for keeping track of bullets, the Stage Manager for holding all entities used in the current stage, and the HUD Manager, for holding entities that need to be rendered irrespective of the camera system.
Most of World is black boxed and as such would be fairly reusable in any shooting game, with different RPlayers with whatever special requirements the other game needs, etc.