Controls

Button	Function
Q	Quit the Game
←	Turn Ship Left
→	Turn Ship Right
^	Accelerate Ship
Right CTL	Fire Bullet

Program Flow

Initialisation

At the start of the game's running, the setup() method is called, which is responsible for initialising the necessary objects and variables. First, the ODE physics world, which is responsible for handling all simulations of motion and collision, is created and the necessary physics objects are initialised. Then, all necessary objects (which model components such as the player, asteroids and barriers) are initialised. These objects include an ODE physics body and an OF Bounding box (for display or, if a collada model is used, for more complicated graphical transormations). Finally, the OpenFrameworks camera (for capturing the drawn OF objects) and light (to properly illuminate the drawn objects) are configured before the first draw loop is called.

Update

The update() method is called directly before each draw loop and is responsible to updating the physics, positions and rotations of all the objects in the game. All of the instantiated objects are updated for the new positions and rotations of their ODE counterparts (e.g. if a force is being exerted on the ship, then the OpenFrameworks objects will need to be updated for this change). Furthermore, the update() method is also where the collision call-back function is . After all collisions have been processed, any destroyed asteroids or bullets are re-initialised in their respective arrays.

Collision Resolution

The collision resolution function evaluates several conditional statements to decipher which objects the colliding ODE-Geoms belong to and then execute the appropriate response. If a player and an asteroid collide, the game is ended and the finishing score is outputted to the console. If a player or asteroid collide with a barrier, their respective physics object is mirrored across the arena, to simulate the open borders effect of *Asteroids* (Atari Corporation, 1979). If a bullet and an asteroid collide, both objects are marked for destruction will eventually be re-initialised when the next update function is called. Finally, if a bullet and a barrier collide, the bullet is marked for destruction also.

Draw

As well as drawing the appropriate shape, size, position and rotation of each object into the arena, the draw() method is also responsible for colouring the background, drawing the plane and setting the positioning of the light. The camera is also rotated and positioned so that it is always directly behind, and facing the same direction as, the player.

At the end of the draw() method, the lights and camera are ended and the Score is displayed using a string stream at the top-centre of the screen.

Evaluation

First, due to difficulty implementing the physics and time-constraints near the end of the games development, it was not possible to correctly model the constraint resolution between asteroids. Ideally, if two asteroids were to collide in the arena, they would move away from each other, similar to a real collision in space. However, when the contact joints were created and their resolutions calculated, the game experienced several different 'program-ending' error. Despite experimentation with the different contact parameters (such as softness), an appropriate solution was not able to be developed in the time-frame.

Furthermore, in its current state, *Dogfight* would need further design and coding to be considered fully-developed. Components such as a start and end screen, while not integral, create a more well rounded experience for the player, and including them in later iterations of *Dofight* could be beneficial. Also, more realistic animations (e.g. explosions when asteroids are hit and even splitting into two asteroids) could also be programmed to make the game more appealing to the player.