Iain Cotter S1507974 Games Programming 2

Games Software Development

*I confirm that the code contained in this file (other than that provided or authorised) is all my own work and has not been submitted elsewhere in fulfilment of this or any other award*.

*Signature*.

Iain Cotter

Content

[**Cam**](#_get7avc1jgze) **3**

[Initialising the Camera](#_ovjt68jbratt) 3

[**DisplayWindow**](#_q9q2abjjlpza) **3**

[Screen Setup](#_h3w7idi7l1hi) 3

[Initialising Display](#_w7gn78vvn5uf) 3

[Clear Display](#_11z3r4dweaq7) 4

[Swap Buffer Swaps to other rendered screen. This if done properly should reduce likelihood of tearing and artefacts.](#_k7lct8hp6pcq) 4

[**Mesh**](#_2qf0rg1p8bsb) **4**

[Load Model File](#_hpaotw16k1g2) 4

[Initialising Model](#_p3rjjsdhcx6j) 5

[Render Model](#_sjthnfa6694w) 5

[Set Sphere for Collision](#_62tf7dvs9wkn) 6

[**Shader**](#_td143bgdfwj7) **6**

[Initialising](#_ldg7f0bdqchb) 6

[Loading shader file](#_v09ouy6rds7) 7

[Creating individual Shader](#_8kqhfl7db21n) 8

[Updating Shader uniforms](#_szmf5h1yh181) 8

[Binding shader program](#_lf55ml8kralc) 8

[Vertex Shader](#_8kpx3nhxhtkd) 9

[Fragment Shader](#_jds2pkydtji9) 9

[**Sound**](#_e6yr78vjru0v) **10**

[Initialises](#_707zjsoprulc) 10

[Plays](#_nf5op2vyosym) 10

[Stops](#_xwjxxdz2tpu0) 10

[**Tex**](#_rrhz0v6h3tp6) **10**

[Initialise Texture](#_xy242pukk28i) 11

[Bind Texture](#_9gvbc74bqh1d) 11

[**Transform**](#_ws2n7f54xnls) **12**

[Get Model Matrix](#_on8v6bfdsgyi) 12

[Set Transforms](#_89iz7vr3zzlo) 12

[**TheGame**](#_h5l2992lgac0) **12**

[Constructor](#_43cifpczow8t) 12

[Initialise](#_irupdxy00jrz) 13

[Game Loop](#_w1z24cno8uvv) 14

[Collision](#_ws3ugjqxnc8b) 15

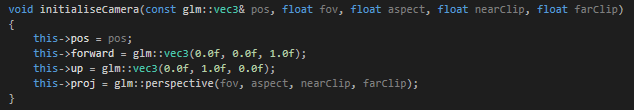
[Key Input](#_t8yjcqbkpvp4) 16

[Render](#_8kub9odrle2d) 16

[References](#_seg13n8iuwe5) 18

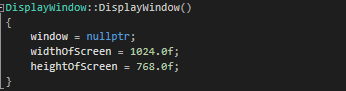
## Cam

### Initialising the Camera

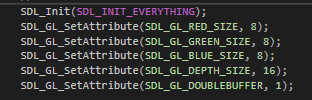
Cam holds values for position of the viewable camera in the virtual space, direction of the camera, sets projection i.e. the representation from 3D to a 2D screen. Sets the field of view, size of screen’s aspect ratio and the viewable space the closest and farthest.

## DisplayWindow

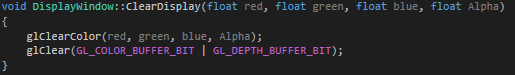
### Screen Setup

 Sets the width and height of screen as it is to appear on monitor.

### Initialising Display

 Sets colors and z buffer a number of bits. Also sets up double buffering (rendering on a background window then swapping windows; improves system performance).  Enables z buffering, not rendering obscured vertices, and back face culling, only render what is pointing at user i.e. only what they can see. Sets background color.

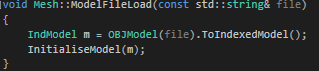
### Clear Display

 Sets color of screen background and the value that it is to be rendered depending on bits allocated to both color and depth buffer.

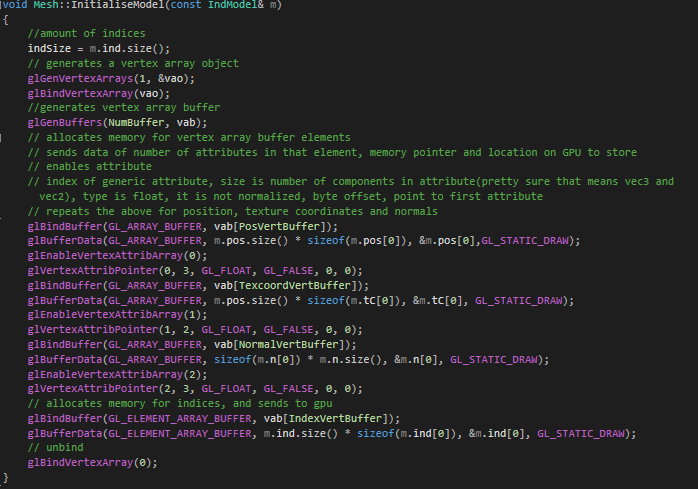
### Swap Buffer Swaps to other rendered screen. This if done properly should reduce likelihood of tearing and artefacts.

## Mesh

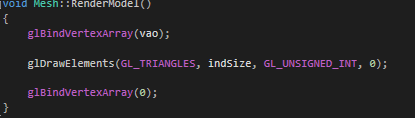
### Load Model File

 Load the from file (explained in section 4) and setup the model to be rendered.

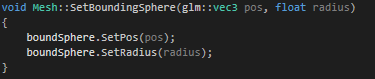
### Initialising Model

 Generates buffers and binds them to memory locations. Sets individual values for position, texture coords and normal in the vertex array buffer, as well as induces of model. Unbinds as to avoid accidental writes.

### Render Model

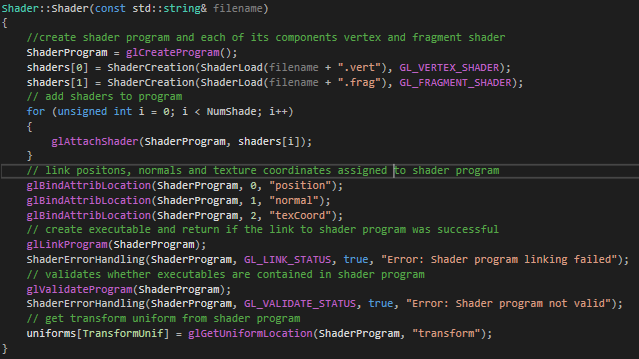
 Gets vertex object and draws its each face in triangles between points, gets the number of indices to draw, uses type of unsigned ints (positive integers) and index location is passed through. Then it unbinds again to prevent error.

### Set Sphere for Collision

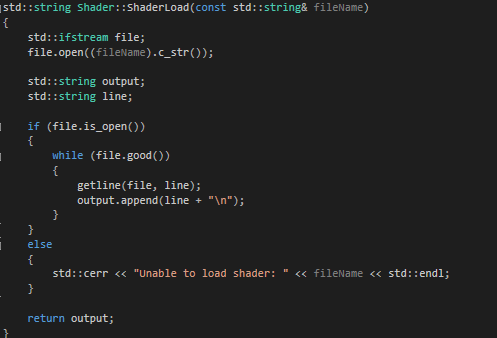
 Sets the sphere position and and its radius. This allows it to be changed to best suit the gameplay.

## Shader

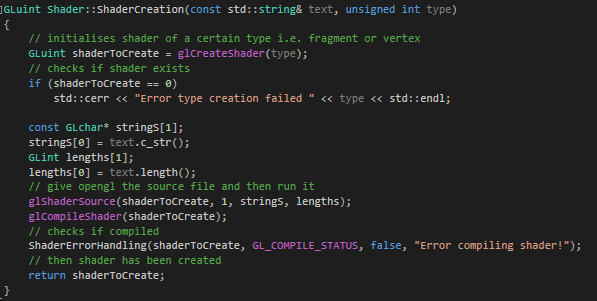
### Initialising

 Creates shader program and assigns premade vertex and fragment (section 4.2), then for every shader assigns shader to shader program. Then links varying values in shader program. Creates and verifies executable as well as validates shader program to ensure it runs. Transform uniform is made equal to shader programs queried transform value.

### Loading shader file

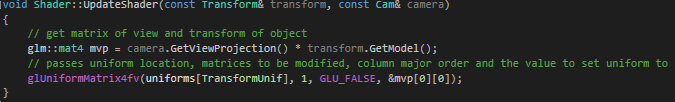
 Gets file name then changes it to c\_string and opens it. Loops while there is no error flag in ifstream. Adds current line to output. Once output holds all line returns it.

### Creating individual Shader



Creates shader of type either vertex or fragment, important since each exists at different step in pipeline. Gets the text of shader then adds to opengl so it can create the file and compile it, returns if compilation failed. Returns shader created.

### Updating Shader uniforms



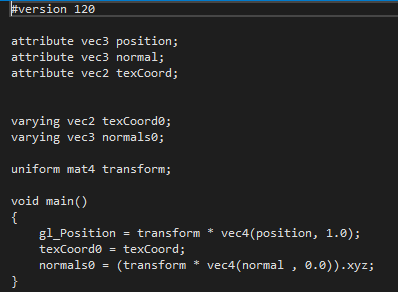
Gets the matrix view projection by multiplying the camera view projection and the transform of the object. The shader is then fed this value to its uniform transform so that it renders the object.

### Binding shader program



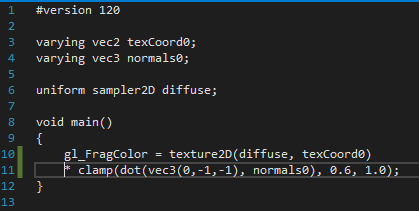
(Called in TheGame section 7) This tells opengl to use this program to render the scene.

### Vertex Shader



Positional data of a vertex.

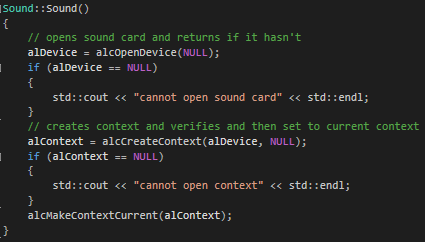
### Fragment Shader



Gets the positional data and the decides the color that should be represented.

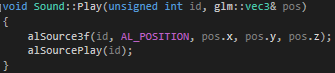
## Sound

### Initialises



Gets sound card and creates context for the audio.

### Plays



Audio file id is passed in and set to a position, then audio is played at that position.

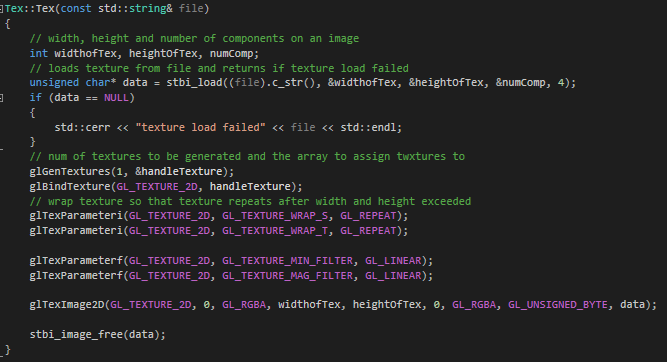
### Stops



Audio file of id is stopped at source.

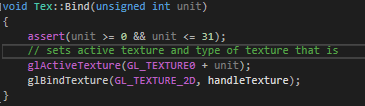
## Tex

### Initialises Texture



Creates data variable char and makes it equal to return of stbi\_load class, returns if failed. Generates texture and binds to memory so that it can be modified. Set texture to repeat. Pass in the values of type (2D texture), level of detail (0 being base value), sets color format to use, width of texture, height of texture, border (always 0), pixel data format, type of unsigned byte and pointer to memory location.

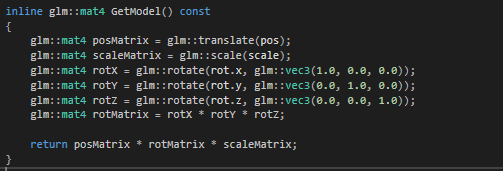
### Bind Texture



Checks texture is one of the 32 textures. Sets active texture to unit and changes unit to current texture.

## Transform

### Get Model Matrix



This returns matrix for shader program (section 4.4).

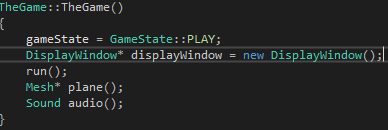
### Set Transforms



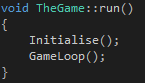
Set transforms for position, rotation and scale. This allows object manipulation.

## TheGame

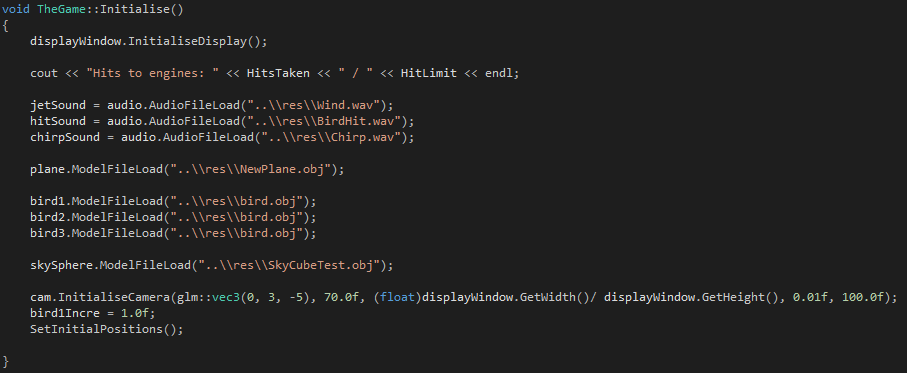
### Constructor



Sets game state to playing so that when game loop occurs it will always not be EXIT (section 8.3), sets display window, run method to initialise variables and start game loop. Mesh and audio device are initialised.



### Initialise



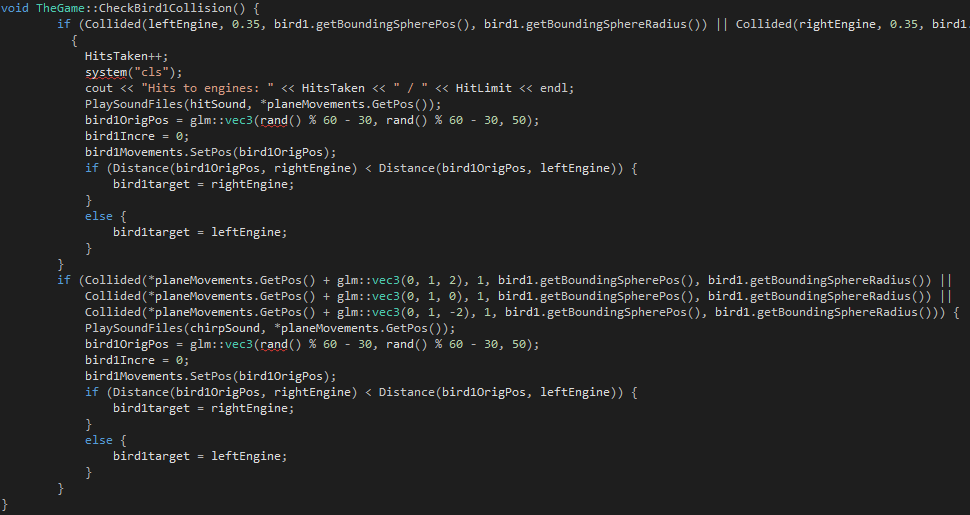
Creates display window (section 2.2). Loads audio files and model files. Initialises camera and its components, sets starting positions

### Game Loop

### 

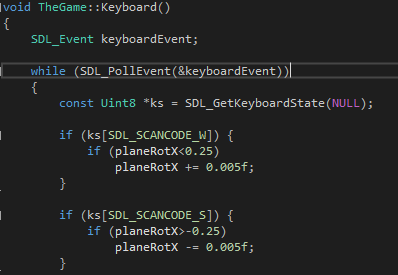
While the state isn’t exit then play jet sound. Check constraints that health has not depleted, and has not collided or left screen space. Continually runs input method (section 8.5). Renders the screen (section 8.6)

### Collision



For every bird first check if in plane proximity and then runs a method similar to this (different for each model as they are bir1, bird2, bird3). This method checks if collided with either engine and increments hit if true, then plays sound and randomises the start position then resets the bird and gives it a new target. If bird hit main plane body, then it is just randomised and reset.

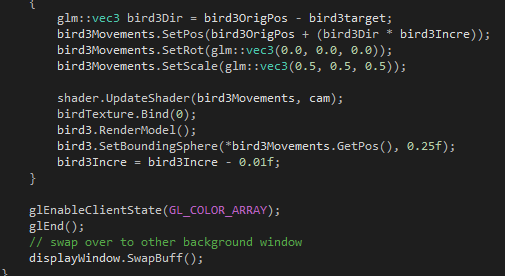
### Key Input



Uses keystates to get the unavailable keys so and then increments or decrements the rotational values accordingly as to which is currently being pressed. This was done with keystates as it allows multiple inputs.

### Render



Loads shaders and textures. Loads sphere object, that has a negative scale to allow it be viewable from the inside, then textures that object. Then draws that object. The planes position and rotation are changed by the rotational increment affected by user (section 8.6), sets sphere of plane that will act bird proximity detection to plane. Finds the engine position at each iteration of render method. The bird’s directions are calculated from origin position and target. Then moved along path by increment amount.

Does this for all 3 birds and then swaps the windows to display the background window.

### References

Thebennybox

[**https://www.youtube.com/watch?v=NS980twY1ZE&index=12&list=PLEETnX-uPtBXT9T-hD0Bj31DSnwio-ywh**](https://www.youtube.com/watch?v=NS980twY1ZE&index=12&list=PLEETnX-uPtBXT9T-hD0Bj31DSnwio-ywh) **20/03/2014 [Accessed 08/01/18]**

KeyStates

[**http://lazyfoo.net/tutorials/SDL/18\_key\_states/index.php**](http://lazyfoo.net/tutorials/SDL/18_key_states/index.php)

**23/02/14 [Accessed 19/12/17]**

Plane 3D Model

<https://www.turbosquid.com/3d-models/free-plane-3d-model/647078>

**03/01/12 [Accessed 28/11/17]**

Sound

<https://freesound.org/>

**[Accessed 02/01/18]**