SOFT351

Assignment 1

Joseph Kellaway

10503639

# Version Information

## Campus

Microsoft Visual Studio Professional 2015 – Version 14.0.25431.01 Update 3

DirectX SDK June 2010

## Home

Microsoft Visual Studio Enterprise 2015 – Version 14.0.25431.01 Update 3

DirectX SDK June 2010

# End User Guide

## Part 1

The user is presented with a 3rd person perspective of a winged-bear which they can move around the screen. The arrow keys are used to rotate the bear and alter the pitch that it is angled in. The W and S keys are used to move the bear forwards and backwards respectively. When turning at near maximum speed the bear will lean into the turn to mimic the animal shifting its weight. To prevent the user from losing the bear off-camera, the F4 key can be pressed to toggle a camera mode that follows the bear. The spacebar can be pressed to make the bear roar. The F1 key can be used to toggle whether these commands are shown on-screen whilst using the program. When angled skyward, the bear can use W and S to fly. Whilst flying the wings flap, a flapping sound is made periodically and the bear always leans whilst turning. If neither momentum key is pressed when flying, the bear glides whilst slowly losing momentum and falling to the ground. If the bear is angled towards the ground, it straightens upright when it lands.

## Part 2

The user is presented with a revolving pig. It is loaded from an “.obj” file and displayed on the screen. To change the pig to another shape, change the file loaded at line 494. To change the texture applied to the shape that is loaded, change the file loaded at 583.

# Programmer’s guide

## Part 1

Global variables that are relevant and the same for all objects are set up. This includes all of the objects required for setting up the Direct3D environment that the bear program runs in. Global variables added to the Microsoft example code starts at line 78 with:

* Meshes (to render later), the projection matrix.
* A diffuse only pixel shader for loading, objects that won’t display shadows correctly.
* Boolean flags to detect keyboard input.
* A variable to alter the rate at which the skybox spins.
* Angles (in radians) that are level with the ground and the y-value of the ground itself.
* Speed increment due to gravity (currently incremented per frame, so an arbitrary scale to reduce it is included) – used for falling.
* Air density for slowing down objects in the air. A value that represents earth’s air pressure at 20 degrees Celsius.
* A value to raise the camera above the bear when in follow mode, with a stabiliser value that scales based on the angle of the bear to reduce camera movement when the Y-direction is changed.
* A “Bear” – an object class designed to hold all of the attributes and actions specific to what the bear can do.
* Light colours and an arbitrary shininess value (used when doing per-pixel reflection effects).

Thing3D is currently used as an abstract class of variables extended by the Bear class. It has been set up to include variables required for every item in the world to be an instantiation of it (currently the skybox and floor). It contains the following variables with the means to retrieve and set them from the main class:

* Location, rotation and scale variables and a vector to hold the initial direction.
* Matrices for calculating how changes to these variables should work and be represented.
* Pointers to shaders, buffers, the mesh and a struct for holding world view projection and world matrices (passed to the shader) so that the object can render itself (not currently in use due to an error passing the ID3D11InputLayout pointer). Pointers are used so that all Thing3Ds can share the same shaders rather than have shaders for every single item in the world (as this would make adding items inefficient for performance).

Thing3D also holds constructor and destructor methods as well as methods to prepare and render itself, though these are currently not in use due to the issue with the ID3D11InputLayout pointer.

The Bear object is a Thing3D with extra functionality. It holds the following additional variables with the means to retrieve and set them from the main class:

* Current speed and fall speeds to indicate how fast the bear is moving.
* Maximum forward and reverse speeds to stop the user from giving the bear too much momentum.
* A drag coefficient and mass used when calculating how much the bear should be slowed.
* A tilt value in radians for whilst turning.
* Maximum climb and descent values in radians to prevent the horrible interaction with going beyond the middle of the ceiling/floor.
* A current and resting position for the wings (in radians), used for calculating the angle to display the flapping wings at.

Bear turning – standard calculations

* Leaning
* Gimbal lock prevention

Bear movement – standard calculations

* Gravity
* Slow down

Skybox

Shadow

## Part 2

Load shape

Load mesh

Load material file

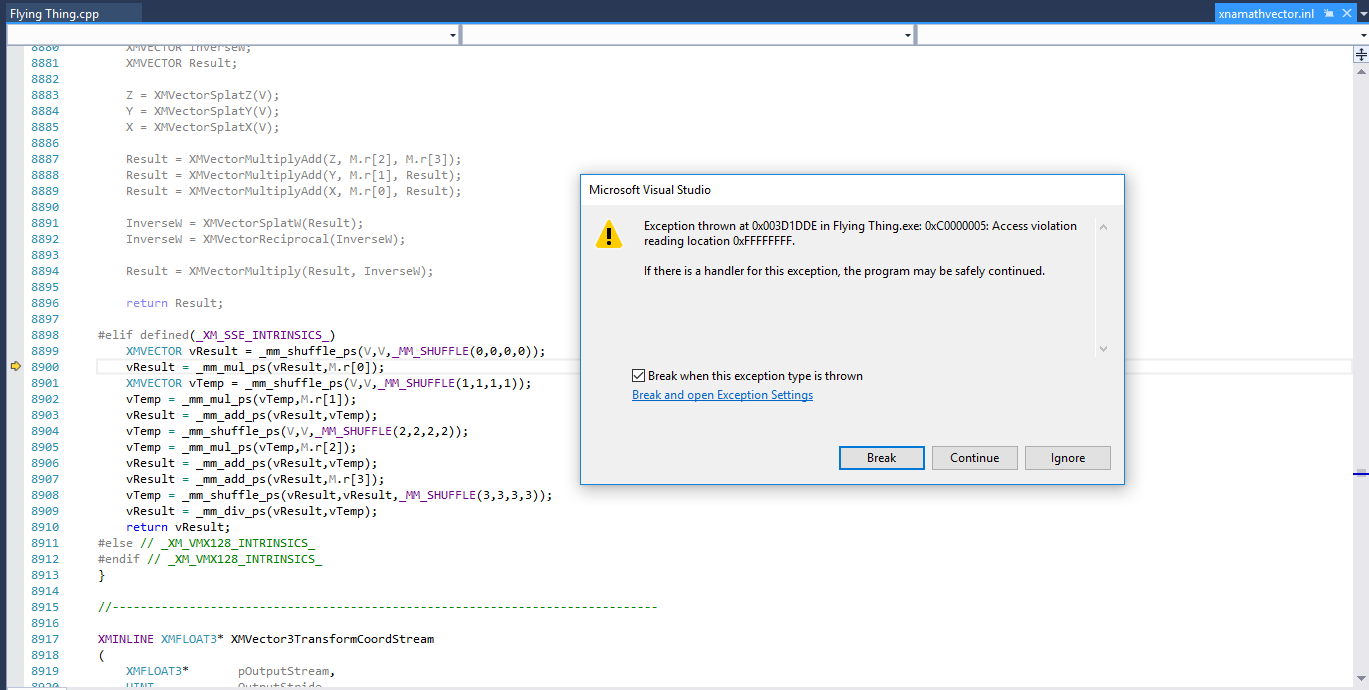
# Additional Notes to the Programmer

Currently, there is an is occasionally an exception thrown on startup of part 1. The exception declares that there is an access violation regarding memory location 0xFFFFFFFF and loads a line in

# Evaluation

# Appendices

## Appendix 1



Occasional error on start-up of part 1.