Dark Water

Dear reader,

The coursework attempt for the Graphics Module has taken the name Dark Water – you will see why when you run it. The programme relies heavily on the Resources folder and will not function without its contents. It is a basic two player game with one destructible volcano that fires boulders in random directions unless a player is nearby. The boulders are thrown at an interval of 4 seconds. The players' boats are normally more agile, so this may seem unfair, however the player cannot fire another shot until 10 seconds have passed from the last time they attempted it (and 30 seconds for sniper mode). Throughout the sea are also scattered 5 boxes with random effects on the one that will pick them up. Some beneficial, some less so. They are mapped to keys F1 through F8 for Player 1 and 1 through 8 (not the numeric keypad ones) for Player 2. They are considered cheats and as such they may be enabled or disabled using F11. Keys F9 and F10 implement a basic save/load functionality respectively. Pause and Esc will pause the game. In the event Esc has been pressed, the user will be given the option to exit by pressing it again. If not, they will have to press Pause to return to the game.

The game has a split-screen view. Player 1 moves with the arrow keys, must hold down the spacebar to enter into sniper mode and uses the right Ctrl key to fire. Player 2 moves with the W, A, D keys (plus S in sniper mode), uses the Z key for sniper view and fires with the left Ctrl key.

The game was developed partially on an ATi Mobility Radeon machine and on the one of the lab. The same code seemed to have the effect to crash on the machine that did not develop it. It was developed and under Windows 7 x64. If it does not run for you, you might need to compile it for your own machine. The source code and all needed libraries should be included.

One may perceive that the game has a huge loading time. That is because it loads models with a very very large amount of faces – in fact, the ultimate artefact is made up of some 90,000 of faces.

The method chosen to represent damage is merely the shifting of the object's faces along the direction of their normal vectors by a factor depending on the amount of damage received. The display list is then recompiled, so as to facilitate things. When a player or the island dies, the display list is reverted to a grid-like form, a sorry reminder of something that used to be there.

The lighting is merely a rotation of a light source at the edge of the world (non-local viewer model) as though to resemble a sun. The water is animated by its texture coordinates being shifted per frame. All this is done procedurally – the programme does not use shaders.

To communicate with the user, the programme uses the FTGL font library which was taken from http://homepages.paradise.net.nz/henryj/code/.

A word on the effects of the boxes collected from the sea; they are as follows:

1. Healing. +25. Or rather -25 of damage. This does not affect the display of the ship. It looks brand new.
2. Poison. -25 damage.
3. Transformation. The ship transforms. Into the teapot. For 30 seconds. This was accomplished by shifting the faces along their normals.
4. Well, I oughtn't really disclose this one... It's an idea I copied from Eternal Darkness.
5. The ultimate artefact. It will be used with your next sniper shot. It has quite an area of effect and does 200 damage to everything it hits, so don't miss when you shoot with it! Also, you don't want to be too close to that either.
6. Double damage. The player's shots cause twice as much damage. The effect lasts 30 seconds.
7. Rapid fire. Or rather, the player can shoot again as soon as their previous missile has expired. No extra cooldown interval. Lasts 30 seconds.
8. The plague. 50 Damage.

There are all sorts of things one could implement to expand this, but for lack of time this has not been possible. As documentation to the code, let it be mentioned in a nutshell that there is a grid of classes and an expansion of the previous assignment's code so it can display polygons and read .obj model files. The Entities (and their descendants) are made to point to a model and use that for their calculations. Finally, there is a rather chaotic network of global variables, which, again for lack of time, it has not been possible to discipline.

For sound, the OpenAL sound engine has been used. You may need to install the runtime, if sound does not seem to work for you (should be included with the source code). Sounds and graphics have been taken off the web; I have created none of them. They may be found at

- http://incompetech.com/m/c/royalty-free/index.html

- http://www.partnersinrhyme.com/pir/PIRsfx.shtml
- http://www.3dsnap.com/archives.php?dir=Appliances%20and%20Household%20Items/&sort=d&sortMode=f
- http://www.turbosquid.com/ and
- http://www.cgtextures.com/.

I do however own the rest of the code as it has been developed mostly from scratch in the context of my MSc course. As such, you are free to take, use and alter the code, but you must indicate where you have found it and are not allowed under any circumstances to sell whatever you may have created that includes part of it.

With all due respect,

Euthyme A. Geordie