SOFT351 Programming for Entertainment Systems: Main Assignment

This is an individual piece of work. *"Create something which does something"*. A Negotiated project.

You must create a working prototype using the DirectX API preferably something which does something interesting. As with last year, you may also use OpenGL as long as you use "new OpenGL", the OpenGL which looks very similar to DirectX (all vertex buffers, index buffers and shaders) or even Vulkan. Whichever way, you must put on a hair shirt and use an unmanaged language (C++). You are pretty much on your own if you use OpenGL or Vulkan, though.

Please see the end of this document regarding the hand-in date, which is just at the end of the semester, as well as the section on the demonstration, which is mandatory, not desirable.

Start by filling in a proposal form (alongside this doc). The proposal is not assessed as such; it is intended to concentrate the mind, and I would like to know that you intend to do before you get too far into the production process.

Your prototype need not be a playable game, see below for a list of suggestions. The only rule is that your masterpiece must use the DirectX API (or OpenGL or Vulkan now) and an unmanaged language as the basis of your work. You may use other higher level tools for assistance – for instance you may use “Photoshop” to produce the textures files, and 3D modelling tools such as “3D Max” to produce meshes. And you can also “nick” such resources from any source you like; this is a SOFT module, not a 3D modelling module. You can also use any other classes or frameworks you like (as long as your framework is not of such a high level that it turns DirectX/OpenGL/Vulkan into something completely different). See the rules of engagement.

We have done no more than scratch the surface of the DirectX API. Almost all the material we have covered is related to 3D graphics, but there is nothing that says that your assignment has to be in 3D space. You can use this assignment to explore particular parts of DirectX; 3D space, sound, networking, media player etc. I do not expect anybody to produce a product which includes all of these elements!

Your *prototype need not be a playable game*; indeed, some of the good submissions in the past have been more like sample code to demonstrate a particular effect or effects.

Originality is not one of by strong points, and the following suggestions for your prototype are by no means exhaustive.

- An artificial life simulation. Examples might be:
  - A "boids" simulation (see separate document). This is probably too easy on its own.
  - A variation on the (Artificial Life simulation) herbivores/predators theme (the herbivores eat grass, which grows when you make it rain, the predators eat the herbivores. Both reproduce when they meet another of their own kind).

- A game. Most in the past have been a variant on a 3D shoot-everything-in-sight game.

- A variant on an old 2D favourite with a new twist. I can't believe I am the first to think of this. Make a 3D Pacman and put the viewer on the player. It is suddenly a very different game as you can't see over the walls to where the ghosts are. Maybe you can jump to look over the walls.

- Something like the Microsoft sample programs, which demonstrate a particular effect or effects. E.g. there has been much written (and quite a bit of sample code) on dynamic generation of scenery. Not only terrain, trees and other similar things. If your trees are loaded from pre-built meshes they will all look the same. Programmatically generated trees could all look different.

- One effect I would like to see, and which is on my "to do" list, is a simple physics sample where a ball (or something more complicated) rolls down a hill and hits a block wall. the wall smashes and falls in a realistic way.

- Experiment with "unusual" (now quite usual) ways to interact with a "game" / DirectX / a computer. For example, I have some Wii wands (old tech now) and Bluetooth adapters which you can borrow. Or, if you know, say, Android or iOS, see if you can control the computer from a mobile phone, or any of the myriad of input devices, including gesture control.

- This is very easy using a game engine, but not so in DirectX: generate some terrain, maybe massive terrain. Then create a buggy like thing, and drive it around the terrain so that it accurately follows the contours. Then create a world which you can explore.

- A more serious application of the material in this module, such as a computer aided training tool or teaching tool.

You are welcome to use my high level classes, the Thing3D, FlatThing3D etc., but I would like you to get inside them and modify them, or write your own equivalents. Believe me, there is plenty of room for improvement in the coding of these classes!

XNA / Managed DirectX

Several have asked if it is possible to use C# / XNA rather than C++ / DirectX. Sorry, **no XNA or C# in 2017/18**. And no managed DirectX either (yes, that still exists). I know you would achieve more in XNA than you would in DirectX, but part of what I want you to get from this module is learning to work in an unmanaged environment.

Software Engineering Issues

In addition, you should demonstrate that you have considered some of the more mainstream computing issues discussed in this module. For example, performance (speed) is often the goal here, but that is not always compatible with conventional good practice in software engineering, such as OO, modularity, encapsulation etc. You should demonstrate that you have considered this, and you should tell me what compromises you have made (if any) in order to reconcile issues of performance against good software engineering practice.

I would also like you to consider some more advanced performance optimisations, such as the culling of non-visible objects.

Or, alternatively, is you have sacrificed some performance for the sake of a purer and more general computing solution, say what you have done and why.

Put this in your brief (brief-ish) report.

Rules of Engagement

You may use any from any sources (e.g. my code, Frank Luna, MS sample code, "gamedev", Nvidia) of program code which you can adapt. But you **must** tell me what you started with and what you have done to it.

You can also use 3rd party libraries this time around, e.g. a mesh loader (though you might already have one of those that you are pleased with), maybe a loader that can handle file formats that allow animated meshes (there are many proprietary file formats here). You could also use a physics library such as a Bullet. Be very careful about using 3rd party libraries and frameworks that abstract you so far away from DirectX (or OpenGL or Vulkan) that that they make it look as if you are programming in something completely different. If in doubt ask

There **must** also be a substantial contribution from yourself. For example, just taking one of my example demos and changing the meshes is not enough. You may also recycle parts of the first assignment. If you are really pleased with your loader or flying creature (or both!), you may use it in this assignment. Think carefully about how you incorporate it, though. Maybe make it into a self-contained class. And finally, I reserve the right to ask you questions about the program code you submit.

Hand In

- Create a single ".zip" archive containing the whole Visual Studio project (not just the executable), with the solution I should open clearly indicated (put that in the write-up below). Don't forget any to include any other resources required; meshes, sounds etc. Submit your archive via the DLE electronic submission system. And don't forget to remove the "debug", "ipch" and ".sdf" files and folders.

- Also include in your archive a brief report on your prototype, which can be a Word document, PDF or equivalent. This isn't a formal exercise in how to document software. However, the report should address

  - What I am looking at from an end user's point of view. How do I work it? Which solution (if there are more than one) do I open?
  - What I am looking at from a programmer's point of view. How your program fits together The flow through your program. This is *not* intended to be an exercise in formally documenting a software project.
  - Anything else which will help me understand how your prototype works.
  - The software engineering issues, such as the trade-off between performance and good practice, which I have asked you to address.
  - What (if anything) you started with (see rules of engagement), where you got it, and what, in the final product, is your own.
  - A (brief) evaluation of what you think you have achieved, and what (if anything) you would do differently, knowing what you now know. Feel free to blow your trumpet! Draw my attention to anything you are particularly pleased with.

Don't go mad on the paperwork; a few pages are enough. This is not intended to be an exercise in how to formally document software, but without some information of this type it is virtually impossible to see how your code works. **Please address the "what you started with" bullet point, as without this information it may not be possible to give your work a grade.**

Assessment

This coursework contributes 60% to the module's overall grade. The module's learning outcomes, as defined in the Definitive Module Record, assessed here are: LO1. Demonstrate a knowledge of the concepts (including mathematical concepts) of programming in a high performance real-time graphics environment; LO2. Design, code and demonstrate a working prototype.

Demonstration – Now counts 30% of this assignment **(well, a negative mark if you don't do it really)**

You must, at some point, demonstrate your prototype to me. Email me to arrange a time. The demonstration can take place before the hand-in date, or shortly afterwards. However, I suggest that you do a demonstration *before the formal submission date*; that allows you to get some pointers (pun intended) before you make the final submission. You may make small tweaks to your work, fixing a bug, that sort of thing, between the formal submission and the demo. I reserve the right to define "small tweak", and I also reserve the right to revert to the project you submitted on the DLE if I think you are taking liberties. Demos can run into the exam period if necessary, you will be given a "final call for demos" reminder.

It does not matter if your prototype will not work on a standard university computer. As long as you can show it working on something, that will suffice. Again, if in doubt, please ask.

The demonstration is the only component of this coursework against which I have allocated a definitive mark. In fact, it is more like a negative mark (-30%) if you don't complete the demonstration. This is because I need to convince myself that your prototype is your own work, and I will ask you questions relating to how your program code works. If you don't seem to know your way around the project, I reserve the right to modify your overall assignment grade accordingly. Please don't get stressed about this; if you have written the program, you have nothing to fear.

Due In before: End of semester (a little negotiation is possible). January 2018

Submit via the DLE electronic coursework submission system.

Demos can take place either before of after the formal submission, see above.

Please note: at busy times, the DLE can become very slow. The slowness of the system is not a valid reason for late submission, so please don't leave it to the last 5 minutes! Get a submission "in the bank" early, you can submit as many times as you wish up to the deadline.

Also note, only DLE submissions can be accepted. the DLE can time out if you are submitting big archives over slow connections. Reduce the size of your submission by removing the folders "Debug" and "pch" (if you are using pre-compiled headers) and also the "intellisense" database file.

Dr. Nigel Barlow
November 2017