Realtime Rendering

Assignment 1

Graphics Performance, Microbenchmarking and Optimisation using VBOs

Assessment: 33%

Due Date: End week 4 - Fri Aug 11 9pm

Clarifications and Updates

- 24/7/2017: Added tristrips.

Aims

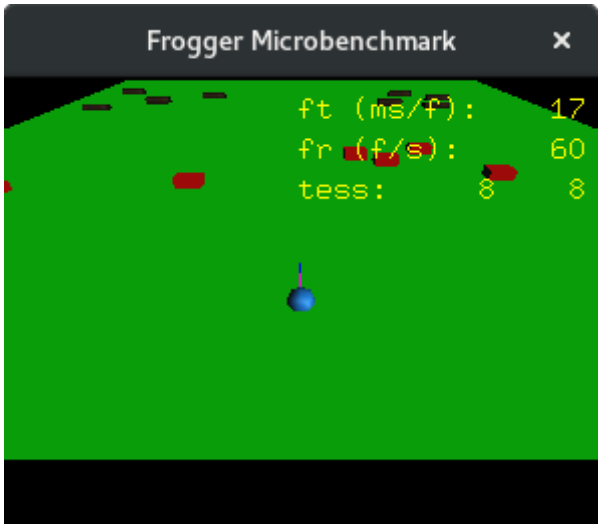
This assignment is intended to introduce students to graphics performance, measurement, benchmarking and optimisation, primarily through the use of vertex buffer objects (VBOs) and associated APIs. SDL is to be used for the UI and event handling instead of GLUT, although not for performance reasons.

The assignment may be undertaken in pairs or individually. By undertaking the assignment in pairs students can gain deeper understanding by discussing performance results they observe.

Problem Description

In computer graphics - real-time applications in particular - performance is always of interest, or, more accurately, image quality and performance tradeoffs: better images take longer - but how much better for how much longer. In this assignment techniques for improved rendering performance using buffers are used and measured.

In the I3D assignments procedural generation was used for shapes such as planes, spheres and cylinders, and for surfaces, possibly a water surface or terrain in the scene. For investigating graphics performance. The assignment also uses procedurally generated objects, more specifically, the same as used for the frogger assignment in I3D in semester 1.



Task

There are three components to the assignment:

- Modifying and extending the supplied Frogger micro benchmark base code frogger microbenchmark (which is a slightly modified version of the 'simple' frogger reference implementation from I3D) to use SDL and vertex buffer objects (VBOs) along with various interactive controls. This code base must be used so performance results can be compared.
- Conducting performance experiments, including benchmarks.
- Writing a report.

All performance measurements are to be done in the Sutherland Lab locally - so that results can be compared - and the aim is that much of it can be done in tutorial time.

The assignment requires you compare peformance of immediate mode with use of VBOs for both static and animated geometry. Use independent triangles for rendering, and optionally explore using triangle strips, with one strip per row or column (or stack or slice), but *not* one strip for an entire object using degenerate vertices. For VBOs use glDrawElements.

Your program should have a performance meter or 'perfmeter' as a toggleable on screen display (OSD) showing frame-rate, frame-time, triangle-rate (triangles/second). It should implement the following controls:

- Tesselation increase/decrease ('+/-' keys), using doubling/halving from a min of 8X8 and a max of 1024x124 (two million polygons - with polygons about 1 pixel in size for 1920x1080 HD resolution).
- Wireframe or filled rendering can be selected
- Use of immediate mode or vertex buffer objects can be selected.
- Lighting can be enabled and number of lights increased/decreased up to 9. Use a default <0, 0, 1> directional light for the first light, and then directional lights from the corners of a unit cube, e.g. <1, 1, 1>, <-1, 1, 1> etc.
- Toggle to have the perfmeter print to the console instead of using an OSD (which can affect performance in itself).

Measure performance and other impacts of:

- Increasing tesselation.
- Using immediate mode, vertex arrays and vertex buffer objects.
- Changing rendering mode from wireframe to filled

- Turning lighting on and off, and specifying or not specifying normals accordingly (they are not needed if lighting is disabled).
- Increasing the number of lights, up to a maximum of 9.

Use 1920x1080 (HD) resolution with the OSD turned off - print to the console instead.

As part of the investigation you are specifically required to find how many lit shaded triangles (polygons) can be drawn at a 'real-time frame rate' of around 60 fps, i.e. in 1/60th of a second using the fastest rendering approach.

You are also to specifically find how long a scene with 1-2 million polygons (triangles) takes to render.

Optional Extras

Add the ability to use triangle strips as well as individual triangles. Are they faster?

With vsync disabled, modify the program so that a steady frame rate of around 60 fps is rendered whilst varying rendering mode and parameters. Use a main loop with an idle function as discussed in the SDL lecture. In the 'idle' routine game logic would normally be peformed including game logic, AI, physics, etc but for this assignment idle() should just sleep (use nanosleep()).

Your report must present results and a discussion of your investigation. It must include graphs of performance - one main graph showing performance of immediate mode, VAs and VBOs for increasing number of polygons for both static and animated geometry, i.e. three lines. And be careful of the scale used, make sure it is linear, even though the tesselation doubling means the dnumber of polygons increases exponentially.

You are encouraged to ask for help if you have any problems, including using the online channel. Good luck.

Submission

Before you submit anything, read through the assignment specifications again carefully and check that you have followed all instructions. Your code must be written in either C or C++ use SDL2 and compile and run on the Sutherland machines.

What

You must submit all source code, as well as a readme file and the report. If you have worked in pairs only submit one assignment but specify both group members in your readme file.

Marking guide

Subject to (minor) change:

- 50% -- Functionality
- 40% -- Report (max 4 pages)
- 10% -- Code