Computer Graphics — Practicals MT2018

This practical is designed to gently introduce you to 3D computer graphics, including a basic understanding of the rasterization pipeline, cameras, transformations, creation and manipulation of geometry, materials and lighting in virtual space. By the end, you should have developed **a basic 3D loader** that can run in your web browser. **There are two practical assignments.**

A basic three.js loader: Due: week 4.

- S: load in a cube and a sphere inside a Cornell box (a grey box with one red wall and one green wall), then texture map the objects and add an interactive a camera model using three.js.
- S+: add phong lighting and have the objects translate, rotate and scale in the scene – feel free to be creative here.

Key concepts to understand in assignment 1:

- **Camera** – how to set up and configure a virtual camera.
- **Rasterisation pipeline** – how to draw a primitive object on screen.
- **Transformation** – how to manipulate primitive objects and a virtual camera.

Requirements for an S:

- **A working camera** that can **translate smoothly from point A to B using camera controls of your choice**. (e.g. first-person or orbital controls). Any camera controls implemented should be fit for your broader purpose: if you intend to develop a first-person game or a racing game, make sure the camera matches reasonable expectations in the genre.
- **A working renderer** that can draw polygonal objects. For this part, it is sufficient to hard-code individual triangles to be drawn on screen.
- **Ability to scale, rotate and translate** a polygonal object **based on keyboard or mouse input, along Cartesian axes as well as an arbitrary axis.**

A basic webgl version to load geometry and materials from a file. Due: Week 8.

- S: load geometry, materials (from a file) and implement your own camera using WebGL without three.js
- S+: add lighting, bump mapping or some other additional visual effect, or make a simple(!) game out of the application.

Requirements for an S:

- **Ability to load in multiple (at least 3) arbitrary mesh objects of a format of choice (e.g. JSON or OBJ). Objects must be more complex than a primitive object.**
  - Can be done from a JSON file or OBJ file, for instance. Example libraries include:
    - http://graphics.stanford.edu/data/3Dscanrep/
    - https://github.com/nasa/NASA-3D-Resources/tree/master/3D%20Models
  - A simple OBJ box and a gem can be downloaded from:
    - https://www.cs.ox.ac.uk/teaching/internal/courses/2017-2018/graphics/material.html
  - You are not expected to 3D model the object in Maya, so feel you are free to download your models from the web.
- **Ability to texture map loaded geometry.**

Both have to be signed off by end of week 7, but are recommended to be completed at the times shown.

For all parts: any substantial progress falling short of the S requirements will count as S-. Anything less than substantial progress will count as S- or "no submission" at the discretion of the demonstrator.

Jassim Happa, Stefano Gogioso

The reason we use javascript is to ensure that you can achieve a lot with relatively little programming. Production code is often written in C/C++, and there is something to be said for the associated performance boost, but all key concepts are the same—including linear algebra, geometry, materials and lighting. Should you still want to develop your application in another language—e.g. because you have strong ambitions to work as a graphics programmer after your degree and/or feel more comfortable with C/C++ or other performant languages—we won't object to you doing so, as long as you **let the demonstrators know first.**

Some resources:

- Practicals from the last two years may serve as a starting point:
    - https://www.cs.ox.ac.uk/teaching/internal/courses/2016-2017/graphics/material.html
- Basics of javascript debugging: https://www.w3schools.com/js/js_debugging.asp
- https://webglfundamentals.org/
- https://webgl2fundamentals.org/
- https://www.tutorialspoint.com/webgl/webgl_cube_rotation.htm
- https://threejs.org/
- https://experiments.withgoogle.com/chrome?tag=WebGL
- https://developer.mozilla.org/en-US/docs/Learn/WebGL
- https://www.awwwards.com/websites/webgl/
- https://stemkoski.github.io/Three.js/index.html

In order to run assets (3D models and materials not created in javascript) in a browser, it is necessary that the assets be passed via a webserver. A simple way to achieve this would be to run a local webserver using python, e.g. by calling "`python -m http.server 8000 --bind 127.0.0.1`" in a terminal widow (make sure that the command is run in the same folder as your index.html file, and that the terminal window remains open throughout).

Debugging tools are becoming more widely available to test and obtain feedback for web applications. Chrome/Chromium is recommended—webkit has greater rendering performance during debugging—but Firefox will work fine as well these days. Remember: F12 (for debugging) is your friend!

This is a very open-ended practical: students are free to tackle the problem how they see fit, demonstrating independent thinking and proposing their own ideas for how to tackle loader development. A large part of the challenge will become more obvious in the latter part of the practical: as complexity in a graphics application grows, maintaining code becomes increasingly challenging. The purpose of this practical is also for students to develop and maintain a large codebase of their own. For any clarification, please consult with your demonstrator.