Scalable Rendering for Graphics and Game Engines

Antonio Chica Calaf
achica@cs.upc.edu

Marc Comino Trinidad
mcomino@cs.upc.edu

PROJECT STATEMENT

Students must deliver one or multiple C++ projects implementing a series of functionalities. During each laboratory session we will introduce one basic and one advanced functionality (amounting to a total of 3 + 2). To achieve the maximum grade, each student will have to implement all 3 basic and at least 1 advanced functionalities.

All project must support reading from PLY (http://paulbourke.net/dataformats/ply/) format and exporting the generated models to PLY or OBJ format. You can find some test models on the path /assig/rrmm-miri/models.

Session 1

- Load and draw models using OpenGL 3 (**vertex arrays, vertex buffer objects, vertex array objects, ...)**
- Implement an interface element to allow drawing N x N copies of the same object, **using instancing.**
- Implement an interface element to be able to display the **framerate**.

https://learnopengl.com/Model-Loading/Mesh
http://www.songho.ca/opengl/gl_vertexarray.html
http://www.songho.ca/opengl/gl_vbo.html
https://www.khronos.org/opengl/wiki/Vertex_Specification#Vertex_Array_Object
https://learnopengl.com/Advanced-OpenGL/Instancing

Session 2

Basic

- Use **vertex clustering** on a **regular grid** to compute simplified version of a loaded model.
  - Take the mean as the representative vertex for each cell.
  - Generate and store at least 4 different **level of details.**

Advanced

- Use an **octree** to generate all the **level of details** at the same time.

Session 3

Basic

- Improve the **vertex clustering** algorithm by picking the vertex representative using **quadric error metrics**.

http://eigen.tuxfamily.org/
https://dl.acm.org/citation.cfm?id=258849

Advanced

- Improve the **vertex clustering** algorithm by implementing the shape preserving algorithm described in the section 4.1 of Willmott *et al.*

https://dl.acm.org/citation.cfm?id=2018347

DELIVERY

Please upload a single zip file by **April 23rd**, named after your username. For instance: marc.comino.zip

The zip file should contain:

- A compilable and executable project. This includes:
  - All the required .c, .cc, .cpp, .h, .hpp, .ui, etc. files needed to compile your application.
  - A Makefile, CMakeLists or similar script that is able to compile and generate an executable file out of your source files.
  - For **linux** submissions: A list of the dependencies needed to compile your application.
- A short report explaining the implemented functionalities.
  - The report must describe which functionalities have been implemented and which of the different projects contain them. It should be clear which classes implement the different functionalities.
  - I personally recommend to elaborate the report using Microsoft Word or Latex or Google Docs.