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 + 3). To achieve the maximum grade, each student will have to implement all 3 basic and at least 2 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 4

Basic

- Implement a scene with multiple model instances. Dynamically select the **level of detail** for each model using the **time-critical** rendering algorithm to ensure a target frame-rate.
    - Estimate your graphics card **maximum throughput** (triangles per second).
    - Compute maximum number of triangles that your graphics card can process while **ensuring 30 fps**.
    - Estimate the benefit of each model as **d / 2^LD**, where *L* is the **level of detail**, *D* is the distance between the object and the viewpoint and *d* is the diagonal of its bounding box.
    - Maximize the total benefit while ensuring the target frame-rate.

https://dl.acm.org/citation.cfm?id=319365

Advanced

- Implement **hysteresis transition.**

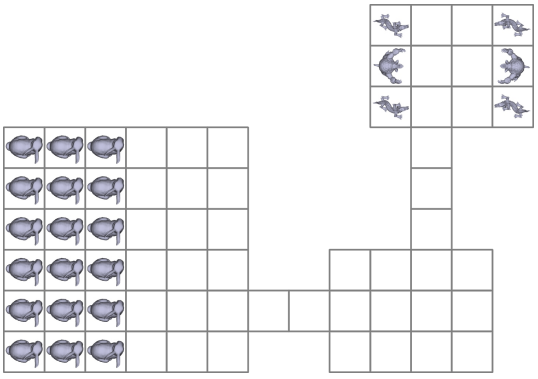
Session 5

Basic

- Implement a scene representing a museum using a tile-based representation. It must have at least three rooms.

Advanced

- Design a complex floorpan.



Session 6

Basic

- Precompute cell-to-cell visibility using **random visibility sampling** on a separate application. Use this information during museum visualization.

Advanced

- Optimize the process using **quadtree ray traversal** and/or **supercover bresenham** (http://eugen.dedu.free.fr/projects/bresenham/).

DELIVERY

Please upload a single zip file by **June 11th**, 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.
- A live presentation of your project. This must take place on either the laboratory class on June 12th.