Final Project

Handout date: 11/19/2019
Submission deadline: 12/17/2019, 23:59 EST
Demo date: 12/16/2019 9AM-5:30PM - By Schedule (a link will be provided)

This homework accounts for 27.5% of your final grade.
Projects considered to be of medium difficulty will have extra points of 12.5% of your final grade (but not more than 80% when added to the grades of the first, second and third assignments).
Projects considered to be of hard difficulty will have extra points of 25.0% of your final grade (but not more than 80% when added to the grades of the first, second and third assignments).

Goal of this exercise

This is the final project of the class. You are free to decide what you want to work on, as long as it follows the guidelines below.

Submission

1. Make sure your code is compiling and working correctly on a Unix (Linux) environment
2. Add a report in pdf format.
3. Add all your code and libraries used to a zip file.
4. Upload the zip file from the previous step to NYU's Newclasses final assignment entry.
5. Make sure to do all the previous steps before the deadline.

1 Project Guidelines

You have to prepare a 1 or 2 pages overview of the project, detailing what you want to do and what is the relation with the topics learned in the class. This document has to be sent to the instructors, and it has to be approved before Tuesday 26th November. The grade of the final project will be given depending on the correctness and completeness of the graphics part of the project (wrt the project overview). Only the parts that are related to computer graphics will be graded.

The extra points, when present, will be given in the same proportion of your project grade. For instance, if you go for a medium difficulty project and get half of the final project grade (13.75%), then you will get half of the extra points for a medium difficulty project: 6.25%.

Programming Languages. You can use C++, Java, or Python for the last assignment.

Software Libraries. You are free to use any software library you want, as long as it is detailed in the project overview and approved by the instructors.

Minimal Requirements. The project must use either ray tracing or rasterization to render a scene. The project must use at least one technique that has not been already covered in the previous assignments (i.e. texture mapping, advanced shaders, procedural geometry/materials, virtual reality, raster transparency, raster shadows, or any other technique that is approved by the instructors).

2 Examples of Possible Projects

- Bump, Normal, and Parallax mapping (Difficult: Easy)
  Your application must display the three techniques, and you must be able to show their differences.
- Real-time Relief Mapping (Difficult: Medium, Extra points: 12.5%)
  You must implement the following technique: https://www.cs.purdue.edu/cgvlab/courses/434/434_Spring_2013/lectures/References/DepthImagesForRenderingSurfaceDetail.pdf.
- Real-time refraction (Difficult: Medium, Extra points: 12.5%)
  You must implement the following technique: http://cwyman.org/papers/sig05_approxISRefr.pdf.
- Shadows visualizer (Difficult: Medium, Extra points: 12.5%)
  Your application must be able to render at least two different types of shadow techniques, e.g., shadow mapping and volume shadows.
- Text rendering (Difficult: Hard, Extra points: 25%)
  Using a texture atlas approach, rendering texts in 2D and 3D. Extra points will be given if your application is capable of rendering different types of fonts or alphabets.
  A good start is the FreeType project.
- Volume rendering (Difficult: Medium, Extra points: 12.5%)
  Using ray tracing: Visualize simple volumes datasets.
  Using OpenGL: interactive volume rendering of simple datasets.
- Simple drive/flight simulator (Difficult: Easy)
  You must be able to render the street (for the driving simulator), buildings, trees, etc. and control the car on the map. You must use textures for the road and buildings.
  You must be able to render the sky and clouds (billboards) and other planes on air.

- Mesh editor (Difficult: Medium, Extra points: 12.5%)
  You must be able to render complex meshes and allow the user to select and edit vertices on these meshes.
  The user must be able to translate, delete and merge vertices (delete an edge (also known as edge collapse).)
- Bezier patches (surfaces) / NURBS visualizer (Difficult: Medium, Extra points: 12.5%)
  Your application must be capable of rendering solid and wireframe versions of these surfaces.
- Ray tracing revolution solids (Difficult: Easy)

- Monte Carlo Ray Tracing (Distribution ray tracing) (Difficult: Medium, Extra points: 12.5%)
  Update your first assignment to a Monte Carlo ray tracing.
  Your ray tracing must be able to render area lights and depth of field effects.
- Path tracing (Difficult: Hard, Extra points: 25.0%)
  Update your first assignment to path tracing.
- Image Processing in OpenGL (Difficult: Easy)
  Your application must be capable of applying image processing filters in real-time. Examples of filters: Gaussian, Sobel, Morphological, Convolution, etc.

- Real-time Physically Based Rendering Shading (Difficult: Medium, Extra Points: 12.5%)
  Your application must be capable of rendering different BRDFs https://learnopengl.com/PBR/Theory https://marmoset.co/posts/basic-theory-of-physically-based-rendering/.
- High Dynamic Range (HDR) Image-Based Lighting (Difficult: Medium/Hard, Extra points: 20.0%)
  See https://www.pauldebevec.com/RNL/.
- Voxel Rendering (Difficult: Medium, Extra Points: 12.5%)
  Implement an application capable of loading a Minecraft map and render it in real-time (correct colors, textures, and voxels must be handled).
- Real-Time Fur Shading (Difficult: Medium/Hard, Extra Points: 20%)
  You must implement the following technique: http://hhoppe.com/fur.pdf.
- VR
- Skeletal animation (Difficult: Medium, Extra points: 12.5%)
  The correct mesh deformation and wrapping must be done too.
- Implement Distance Field Fonts in OpenSpace (Difficult: Medium, Time consuming project, Extra points: 20%)

You must add Distance Fields Text Rendering to OpenSpace https://steamcdn-a.akamaihd.net/apps/valve/2007/SIGGRAPH2007_AlphaTestedMagnification.pdf.

- Flights Visualizer in OpenSpace (Difficult: Easy, Time consuming project, Extra points: 12.5%)
  You must access the global flights' database and obtain the data of the current planes on air (flying) and display them on OpenSpace.
  You must create your renderable class for the flights' paths and handle the real-time updating of the positions of the airplanes.

- Black Hole Rendering in OpenSpace (Difficult: Hard, Math-intensive, Time consuming project, Extra points: 30%)
  You must create the rendering of black holes in OpenSpace. You must have a math-oriented strong background (you will need to read and understand general relativity equations). Also, you will need to figure out a way to render the results in real-time.
  Renderings like the Gargantua Black Hole in the movie Interstellar. https://arxiv.org/pdf/1502.03808.pdf and http://www.its.caltech.edu/~kip/INTERSTELLAR/BrandBlackBoards/blackboards.html.
  You can try a procedural approach: https://www.shadertoy.com/results?query=black+hole.