Assignment 4 COMP 557 Winter 2015
Volume Rendering

Posted: Thurs., April 2
Due: Wed., April 15 at 23:59

General Information

- This assignment is worth 6 % of your final course grade.
- Milena Scaccia (milena.scaccia@mail.mcgill.ca) will handle the office hours and grading. Office hours will be posted on the public web page.
- Use the myCourses discussion board (Assignments/A4) for clarification questions.
- Do not change any of the given code. Add/change code only in the two methods that you are asked to complete.
- Submit a file `VolumeRendering.py` to the myCourses Assignment/A4 folder. If you have any issues that you wish the TA to be aware of when grading, then include them as a separate comment with your submission.
- **Late assignments** will be accepted up to only 3 days late and will be penalized by *1 point per day on your final course grade.* If you submit one minute late, this is equivalent to submitting 23 hours and 59 minutes late, etc.

  Note that the assignment has 60 points, so the late penalty is 10 points per day.

User Interface

All windows provide similar interactions as in previous assignments. The only mouse buttons really necessary are the left or middle buttons which, for this assignment, behave the same and rotate the view via Arcball. Alternatively, you can use Ctrl+($\leftarrow$/ $\rightarrow$ $\uparrow$/ $\downarrow$) for Arcball rotate left/right/up/down.

Questions

Volume rendering is a useful technique for visualizing three dimensional arrays of volumetric data obtained through sampling, simulation or modeling techniques. In this assignment you will create a simple volume renderer using OpenGL's 3D texturing methods. Your task will be to complete the methods `transfer_func()` and `draw_volume()` in order to achieve visualization of features using transfer functions for the first method, and proper slicing, compositing for the second method.

1. **`transfer_func()` (30 points)**

   You will write a set of transfer functions to reveal the object in the bounding box, and highlight features. The starter code allows for three transfer functions to be applied at run-time using keyboard inputs '0', '1', '2', '3', each of which invoke a transfer function. Key '0' is implemented to revert to the default transfer function of the starter code. Your task is to write transfer functions '1', '2', and '3'. To help you identify intensities that you might wish to highlight, `display_histogram()` displays the distribution of values in the data set. You may also look at individual slices using an image viewer/editor to examine pixel intensities of the different materials in the slice. (Use the discussion board to suggest viewers.)

   1) Write a binary threshold transfer function which highlights the surface of the head i.e. the skin.
   2) Write a transfer function which highlights the skeletal structure of the skull.
   3) Write a transfer function of your own to try to highlight one or more parts of the head e.g. the brain, the eyes, the teeth, etc.

   Note that it is not always possible to design transfer function to exactly isolate a chosen part.

2. **`draw_volume()` (30 points)**

   The starter code sets up a 3D RGBA texture and a bounding box. `draw_volume()` sets up the initial position of the slices relative to the initial camera position.

   Your task is to complete this method so that the slices always face the camera (eye). That is, the texture volume and the bounding box will stay fixed in world coordinates, but the slices that sample the volume must be re-oriented to face the camera as the viewer moves. If you do not do this, then the rendering will be of poor quality when the slices are oblique to the line of sight and the rendering will be incorrect if the object is rotated more than 90 degrees since the slices will be rendered in the wrong order.

   There are several possible solutions. One simple solution uses `glMatrixMode(GL_TEXTURE)` which allows you to transform the 3D texture coordinates using the same operations that you use in matrix mode `GL_MODELVIEW`, namely `glRotate, glTranslate, glScale`.

   Alternatively, you can use the `GeomTransform.py` transformations which is closer to what you would do in modern OpenGL.

   Whatever you do, you must document it. If the grader (Milena) cannot quickly understand your solution by reading your comments, she will take marks off.

Notes

- You may use `self.view.eye` to access the position of the eye, and `self.view.lookat` to access the lookat vector.
- You may wish to vary `NUMSLICES` when testing e.g. setting it to a small number e.g. 5 allows you to better see the individual slices as you rotate the cube relative to the observer.
- The data's bounding box is centered at $(0, 0, 0)$, so that x, y, z-coordinates go from -0.5 to 0.5 but texture coordinates instead go from 0 to 1. The starter code takes care of translating the texture coordinates by 0.5, but feel free to modify how this is done. The texture layers are drawn from back to front.
- The starter code implements a default transfer function where all RGBA values are set to 255. Therefore you will see a filled in cube until question 1 of the assignment is answered.
- The second window displays the view from an external camera to show the viewer transformation of the first window.
- You may wish to comment out `display_histogram()` when you are not using it.