Computer Graphics: CS 6360
Spring 2013 Assignments

Prof. Sukhendu Das
TA's: Ankit Shrivastava, Prateek Shrivastava
www.cse.iitm.ac.in/~vplab/computer_graphics.html

February 16, 2013

Contents

1 Common Assignments (All Compulsory)

These assignments are to be done individually. Demonstrate the working demo to the TA's. Send email to confirm date and time (prateek.shrvstv@gmail.com or ankit.jec.jbp@gmail.com). You must bring your own laptop for the demo. In case you don't have a laptop, you can request for a machine in VPLab, atleast 3-4 days before the deadline.

Requirements: OpenGL 2.0 ready machine preferably having Nvidia graphic accelerator.

1.1 Import a .blend file to OpenGL

Marks: 9

Deadline: 23rd February 2013.

The aim of this assignment is to get you familiarized with OpenGl as well as prepare you for future assignments.

The objective is to load a .blend file (default file format for Blender) in Vertex Buffer Objects and render it using openGL 2.0 or higher. You are **allowed to use any API** for the assignment. You are also allowed to **export .blend file to other formats** if you like.

The blender model given will have mesh as well as armature data in it. The assignment will not be considered complete if you fail to load the armature/mesh data from the blend file. To know more about blender Google it, there is much information available. **Some sample .blend files can be downloaded from the course web page.**

References

1. Computer Graphics using OpenGL; 2nd edn.; F. S. Hill Jr.; Pearson Education, 2003

2. The OpenGL Programming Guide - The Redbook, seventh Edition, Addison-Wesley

3. Blender is a open source modeling software and can be found at `www.blender.org/`

NOTE: This assignment 1.1 will be helpful for the task of team assignments to be done later.

1.2 Create .geo file for a basic solid object and test it on Gmsh

Marks: 3

Deadline: 15 March. [Tentative]

Aim of this assignment is to attempt basic CSG operations to generate new shapes.

Input: Different views of a basic solid object will be given in jpeg format. (To be given by the TA's)
Output: A .geo file which will generate the desired solid.

Testing : Gmsh is a freely available software to read .geo files; use it to check your output.
Get Gmsh at `http://geuz.org/gmsh/`

To learn more about .geo file format go here.`http://geuz.org/gmsh/doc/texinfo/gmsh.html#File-formats`

1.3 Surface construction

Marks: 3

Deadline : April 1st Week.

Use the theory taught in class as reference and write simple OpenGl programs to render a wireframe of Cubic Splines for a set of N points (for N = 6 - 50) and Bezier Surface for 16 control points. **Use of any API for this assignment is not allowed**, and ofcourse you can use OpenGL, freeglut, and glew (`http://www.cse.iitm.ac.in/~vplab/courses/CG/opengl_start.html`).

Input: Control Points.

Output: Render the Surface or spline, and show wire frame models.

2 Team Assignments

Marks: 25

Submission Deadline: Around 25th April 2013 (Tentative)

1. A Team can have a maximum of 3 members.
2. A team should be registered before starting the work.
3. Registration can be done by meeting the TA's and informing them about the choice of assignment, no later than **22nd February 2013.**
4. Teams can select from any one of the 7 assignments given in this section.
5. No two teams are allowed to do the same assignments, allotment will be done on first come first serve basis.
6. All Team Assignments require the use of assignment 1.1 as starting point.
7. Use of API's for any task in the assignment is allowed.
8. You are required to use OPENGL only.
9. Programming language used should be C++.

Submission Procedure

1. Create a presentation for the work you have done. Clearly mention the **input, output, implementation methodology,** and any improvements that you have made in the existing technology.
2. Each team will be given **15 mins** to present their work.
3. During the presentation you are required to show a working demo of the assignment you have selected.
4. Your assignments will not be evaluated if you don't have a Demo.
5. You are required to submit a report along with the demo. Format of the report can be found at the course webpage.

6. Create a CD/DVD containing all the **dependencies, references, presentation slides, results and source code** for all the assignments (including the individual assignments). Add a Readme.txt file for setting up your project from scratch.

7. Create separate folder for individual assignments for each team member on the CD/DVD. Add a Readme.txt file for setting up your project from scratch.

2.1 Comparison of Continuous Collision detection (CCD) Algorithms

Marks: 20

Use of API's for implementation of collision detection algorithms is allowed.
The computational cost of a collision detection algorithm depends not only on the complexity of the basic interference test used, but also on the number of times this test is applied. Therefore, it is crucial to apply this test only at those instants and places where a collision can truly occur.

Several strategies have been developed to this end: (1) to "Find a lower time bound for the First collision", (2) to reduce the pairs of primitives within objects susceptible of interfering, and (3) to cut down the number of object pairs to be considered for interference. These strategies rely on distance computation algorithms, hierarchical object representations, orientation-based pruning criteria, and space partitioning schemes.

Input : The number of objects, layout, motion pattern etc.
Output : You are required to come up with a detailed comparison of the following CCD algorithms :

1. OBB
2. k-dop
3. KDtree
4. Oct-tree

Use as many metrics that you can find in literature to compare these.

References

1. Jimenez, Pablo, Federico Thomas, and Carme Torras. *"3D collision detection: a survey."* Computers & Graphics 25.2 (2001): 269-285.
2. Kockara, S., *"Collision detection: A survey."* Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on. IEEE, 2007.
3. Min Tang, Dinesh Manocha, Sung-Eui Yoon, Peng Du, Jae-Pil Heo, and Ruofeng Tong, VolCCD: Fast Continuous Collision Culling between Deforming Volume Meshes, ACM Transaction on Graphics, 30, 5, Article 111 (October 2011), 15 pages.

2.2 Chemical formula visualizer

Marks: 22

Biomolecules, such as proteins and nucleic acids (DNA and RNA), are involved in every aspect of cellular function. Often times, understanding their structure is key to understanding their function. In the past, crystallographers and biologists created detailed real-world models, called Corey- Pauling-Koltun models, using wooden or synthetic spheres to represent atoms and sticks to represent bonds. Today, these models of protein structures, referred to as space-filling and ball-stick models, have been adopted in computer graphics systems to create visual representations. Your job is as follows:
Input: A Chemical Formula eg. C_6H_{12} (Cyclohexane).
Output: Visualize it using ball-stick models and Space filled models.
Your demo should produce smooth edges around the intersection of spheres and cylinders.

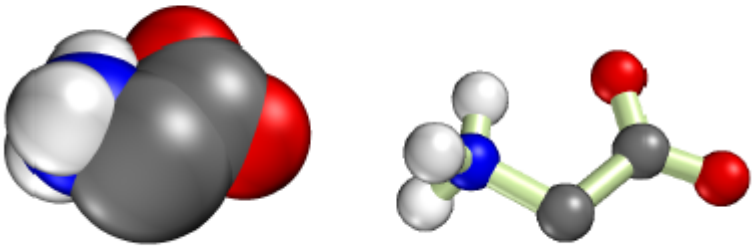

Figure 1: Example of Glycine molecule: Left - space-fill model; Right - Ball Stick model

References

1. Pranav D Bagur, Nithin Shivashankar and Vijay Natarajan, *Improved Quadric Surface Impostors for Large Bio-Molecular Visualization*, Proceedings of the 8th Indian Conference on Vision, Graphics and Image Processing, 16-19 Dec. 2012, Bombay, India.

2. Molecular Visualization Freeware `www.umass.edu/microbio/rasmol/`

2.3 Texture mapping

Marks: 25

Texture mapping algorithms use mesh parameterization methods to find an optimal map for the vertices of a 3D model in texture space. These techniques vary in the properties they try to optimize such as stretch and skewness of the texture when mapped onto the surface. While most of them do well in terms of quality, they tend to be computationally intensive for large mesh models, which limits their use in interactive applications.Use a greedy alternative that is significantly faster than current algorithms and achieves comparable quality [1].
Input: Texture image and a polygonal mesh.
Output: Textured surface of a model with minimum seam errors and to generate self tile-able textures for use in conjunction with your texture mapping algorithm.

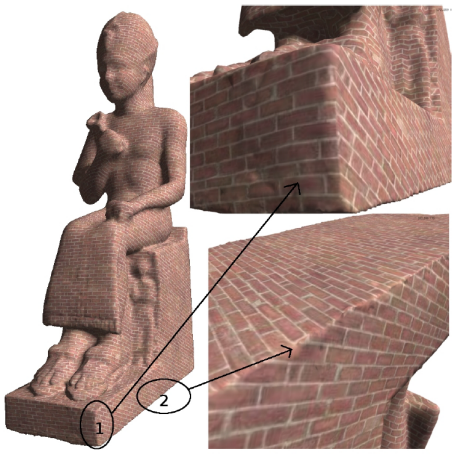

Figure 2: Example of a texture on Model

References

1. Vikram Pratap Singh, Anoop M. Namboodiri *Efficient Texture Mapping by Homogeneous Patch Discovery* Proceedings of the 8th Indian Conference on Vision, Graphics and Image Processing, 16-19 Dec. 2012, Bombay, India.

2. Survey of texture mapping `www.cs.cmu.edu/~ph/texsurv.pdf`

3. NeHe Productions: Texture Mapping nehe.gamedev.net/tutorial/texture_mapping/12038/

4. Texture Mapping Tutorial http://www.glprogramming.com/red/chapter09.html

5. Texture Mapping Tutorial http://www.cse.msu.edu/~cse872/tutorial4.html

2.4 Stadium crowd rendering

Marks: 23

Many sports titles feature stadia that require some degree of crowd rendering technology. Your job is to create an animation for portion of a stadium with a hooting crowd.

Input: Models for various hooting pose's and Stadium portion.

Output: The implementation should have controls to increase or decrease crowd density. Various crowd pose's are desirable. Use your imagination to create an interesting crowd.

Figure 3: Examples of Scenes, you are required to render.

References

1. Wolfgang Engel, Chapter 3, GPU Pro-3 Advanced rendering techniques, CRC Press.

2. `www.crcnetbase.com/doi/abs/10.1201/b11642-5`

3. Crowd simulation with CUDA - Greenleaf `www.greenleaf.dk/projects/cudacrowd`

2.5 Hybrid Ray-tracing on GPU

Marks: 22 + 3

Parametric surfaces are used widely in Computer Aided Design (CAD) and other fields. They provide a compact and effective representation of geometrical shapes for engineering, graphics, etc. The most powerful feature of parametric surfaces is their ability to stay curved and smooth even when viewed at close distances. Parametric bicubic patches of the Bezier form is the most popular among the many possibilities and is popular in many engineering and scientific applications. Direct ray tracing of parametric patches has natural advantages over rendering their tessellations. Implement a scheme for interactive ray tracing of scenes with multiple objects, bounces, soft shadows, etc. using the technique described above.
Input: Polygonal mesh model and light positions.
Output: A scene with complex shapes (not just cubes and spheres) showing effects of soft shadows, color bleeding etc.

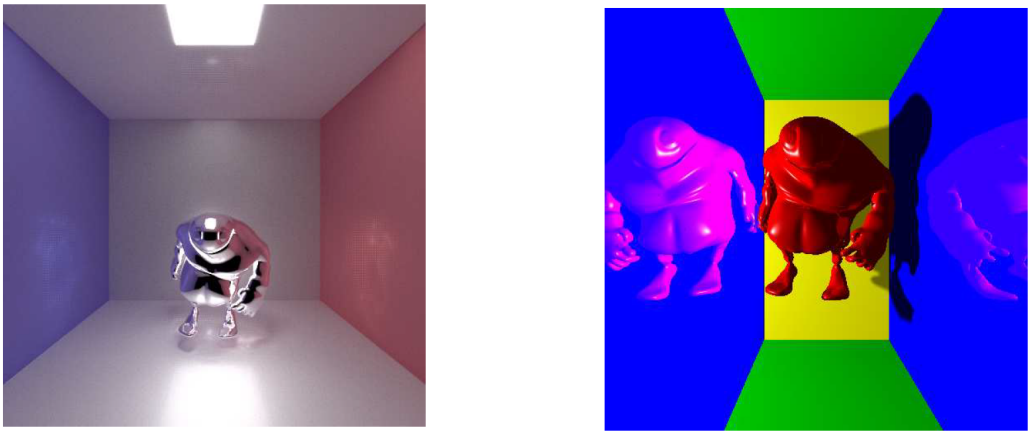

Figure 4: Examples of Scenes you are required to render.

References

1. Rohit Nigam and P J Narayanan, *Hybrid Ray Tracing And Path Tracing of Bezier Surfaces using a Mixed Hierarchy*, Proceedings of the 8th Indian Conference on Vision, Graphics and Image Processing, 16-19 Dec. 2012, Bombay, India.

2. `graphics.stanford.edu/papers/i3dkdtree/`

2.6 Object modeling using Constructive Solid Geometry

Marks: 26

CSG gives a high-level description of geometry. It is intuitive because its syntax resembles the way humans may describe objects in space. Reverting for a moment to two dimensions, the object , " A Simple Two-Dimensional Object " could, sacrificing any mathematical accuracy, be described as "a circle with four rectangles sticking out its sides" or "two rectangles crossed on top of a circle". Having a computer come up with such a description is surprisingly hard, mainly because humans, unlike computers, can "see" the circle in spite of the fact that its characteristic circumference is interrupted by rectangles. As it turns out, having the same computer generate a geometry from such a description is relatively simple.
Input : Polygonal meshes of basic 3-D shapes and a scene graph.
Output : Creating and Rendering of objects using Boolean set operations.

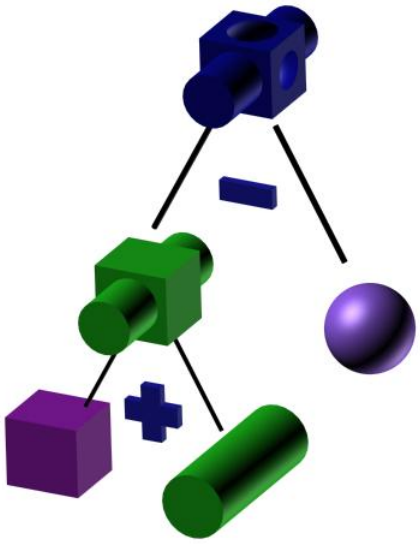

Figure 5: A typical scene graph

References

1. Carve is a C++ library designed to perform regularized boolean operations between two arbitrary polygonal meshes. `http://code.google.com/p/carve/`

2. Library intended to provide a set of useful functions to deal with 3D surfaces meshed with interconnected triangles. gts.sourceforge.net/index.html

3. http://cse.csusb.edu/tong/courses/cs520/notes/mesh.php

4. Constructive Solid Geometry tutorial http://www.codecreator.net/csg/csg.html.

5. CSG in MesoRD http://mesord.sourceforge.net/man/mesord/docbook/ch04.html

2.7 Face modeling with Opengl

Marks : 23 + 4

Task : Simulate the face of person using the frontal 2D image and 3D model of a face mesh. Since the depth information is not available we will actually approximate the 3D model of the 2D image, using the texture mapping, which is provided in OpenGL after finding a appropriate function, for mapping the pixels of image on the vertices of the face. Three points each on the wireframe and the image are used for mapping. Lighting and shading is used to create a more realistic look.

Input: .wrl/blend file for 3D wireframe of face and images (.jpeg/.bmp) of faces (frontal poses).

Output: Face at different tilt, yaw and roll angles; also change light position and observe the effect.

Your implementation should be capable of the following:

1. Reading .wrl/.ply/.blend file and rendering them in OpenGL.
2. Face image (.jpeg or .bmp) must be mapped on a 3D model.
3. Mapping function must be developed.
4. Lighting and shading should be applied using openGL.
5. Several GUI options should be provided to change the light position etc and for saving the bitmap.
6. Basic animations like realistic movement of eye, eyebrows will deserve extra credit.

References

1. Rick Parent. Computer Animation :Algorithm & Techniques, Second Edition ,Morgan Kaufman Publishers,2008.
2. Donald Hearn, M. Pauline Baker. Computer Graphics C Version ,Second Edition. Prentice Hall,2008
3. `http://nehe.gamedev.net`
4. VRML tutorial `http://www.edcenter.sdsu.edu/vrml`.
5. `http://www.winprog.org`
6. `http://en.wikipedia.org/wiki/PLY_(file_format)`