CompSci715 Assignment 3, 2002

Visualization & Physically-based Modelling

Due: 11am, Mon 16th September

Getting Started

Download and unzip *Ass3Source.zip* from the web server. The contents of the folder *Modelling* are for part A and the contents of the folder *Visualization* are for part C of this assignment. The *geometry* folder is the usual *geometry* package, with a few slight changes, and must be on the classpath for all stages of the assignment.
Indicated marks are out of a total of 35. This assignment contributes 5.5% to your final mark.

A. Physically Based Modelling [10 marks]

The directory *Modelling* contains most of the animation code for the mass-spring system covered in the lecture handout.

1. The implementations of the *getState* and *getStateDerivative* methods of the Ma*ssSpringSystem* class have mysteriously vanished. Re-implement them, and check that the pendulum works. Typical parameters for the execution, with a rather "springy" 5-element pendulum, are given as defaults in the *Parameters* dialogue box.

   [5 marks]

2. Suggest a way of informally estimating whether a time step is small enough for the differential equation solution to be stable. Use your method to determine approximately the maximum time step that can be used with the above pendulum parameters (other than parameter 3!) while still maintaining stability. Obtain answers for:

   (i) An Euler method

   (ii) A mid-point method

   (iii) A fourth-order Runge-Kutta method

   [3 marks]

3. Repeat part (C.2) with a stiffness of 250 rather than 25. Make a few intelligent remarks about the results of parts (C.2) and (C.3).

   [2 marks]

B. Volume Visualization - Written Questions (can be written out by hand, or done electronically, whichever you prefer) **[10 marks]**

1. Let a, b, c and d denote values of a 2D density function (scalar field) at positions (0,0), (1,0), (0,1) and (1,1) respectively. Derive an expression $\rho(x,y) = f(x,y,a,b,c,d,)$ for the bilinearly interpolated density at location (x,y), where x and y are both in the range 0 – 1.

   [2 marks]

2. Show from your answer to question 1 that the bilinearly-interpolated density variation along a line parallel to either axis is linear.

   [1 mark]

3. Again with reference to question 1, suppose *a* and *d* are both negative, while *b* and *c* are both positive. Suppose further that we want to draw a contour line for the ρ (x,y) = 0 isovalue. The contour intersects all four edges of the square, and if we are going to approximate the contour with two straight-line segments, there are two alternative ways of connecting the edge crossings. Sketch the two alternatives. Then, using your answer to question 2, derive and explain a test that you could perform to allow you to decide which of the two connection schemes was the correct way to approximate the *topology* of f(x,y).

   [4 marks]

4. Briefly discuss the relevance of your answer to question 3 to the problem of dealing with the so-called "ambiguous cases" in the Marching Cubes algorithm. Don't get too detailed, e.g. you *don't* have to identify and discuss all ambiguous cases in the Marching Cubes table.

   [3 marks]

C. Volume Visualization – Programming [15 marks]

The directory *Visualization* of the assignment source code contains a largish program called *VolViz* that was one of Richard's research projects. The program was used to investigate the usage of particle systems for volume visualization. In addition to this it has the capability of generating and displaying standard isosurfaces using a Marching Cube type of algorithm. You are not expected to understand any of the particle system stuff, but you *are* expected to understand the Marching Cube code, which is all in the package *Isosurface*. Please realise that *VolViz* was a research project with lots of bugs buried in it. You're welcome to explore beyond the prescribed area for this assignment, but don't be surprised if you get a crash or an apparently infinite loop. For example, if you try to create a good polygonization of the rounded polyhedron volume, it will take you at least 6 hours of computation!

To get a feel for the program compile and run it. If the program code is in the folder *c:\Ass3\Visualization* then you must set the classpath to specifically include *c:\Ass3\Visualization* and also *c:\Ass3\Visualization\vecmath.jar* and the *Magician*

classes. *vecmath.jar* is the Java3D vector math library[1]. The single command *javac VolViz.java* should compile everything. Note that under some circumstances this might not work with the JDK1.3 compiler (even if you use version 1.3.1_01). In this case use the *oldjavac* compiler, rather than *javac*.

Run the VolViz application and select *Sphere* from the volume menu. From the *Displayables* menu, select *Isosphage Group*. Click OK in the pop-up dialog box, and then type the number 10000 into the "# particles" text field in the control panel. Terminate with <RETURN>. You should see 10000 particles buzzing around on a sphere. Change the "isolevel" text field to 0.1 and notice that the particles all move onto a smaller sphere. Set the isolevel back to 0.0 and select *Torus* from the *Volumes* menu. Now you should see all the particles on the surface of a torus.
Look at the source code in *Volume.java*. Ignore the *ParameterList* stuff, which you're not expected to understand. You'll see that a *Volume* is essentially just a double-valued field over R^3. Look at *Sphere.java* to see how an instance of *Volume* is implemented. Set the volume back to a sphere. Click delete on the *Isophage* control panel. Select *Isosurface* from the *Displayable* menu. Look at the control panel to see if you can figure out what it's meant to do. Click *ComputeSurface*. You should see a polygonal version of the isosurface that the particles were buzzing around on. But … oh dear – it doesn't seem to be working. Maybe that nasty cyber-gremlin that eats lines of source code has been at it again.

1. Study the code in the *Isosurface* package. Identify the places where the cyber gremlin has been (all are in *Isosurface.java*). Fix up the code, and get it working on
   the sphere and the torus.

   [5 marks]

2. Look at the *TrilinearField* code in the *Volumes* package. It, too, seems to have been nibbled at by the vermin. Repair it. Note that the data to be entered into the automatically-generated control panel for the volume are field values at the vertices of a cube. The first row is the values at (0,0,0), (1,0,0), (0,1,0) and (1,1,0), in that order, while the second row is the values at (0,0,1), (1,0,1), (0,1,1) and (1,1,1). Whenever the user changes a value in one of the control panel fields and types <RETURN> the complete set of values is copied via the *setParameters* method into the *TrilinearField* instance variables. [You are not expected to understand that mechanism, which was set up to provide quick easy prototyping of new classes].

   Check out your code by testing with simple planar isosurface cases, such as is given by the default values. Don't forget to terminate your editing of vertex values in the *TrilinearField* control panel with <RETURN> *BEFORE* clicking *ComputeSurface*. Otherwise the changed values will not be propagated into the program.

   [3 marks]

---

[1] Most of the *VolViz* library uses the Java3D vector routines rather than the *Geometry3d* library used in the previous assignments. But this shouldn't be an issue for this assignment, since you're not hacking at any of the geometry.

3. Investigate the shape of the trilinearly interpolated isosurface (with isovalue 0) for the following vertex values: {-1, -1, -0.1, 1, 0.9, -1, -1, X} where X is a value in the range 0.5 to 2.0. At what value of X (to 3-figure accuracy) does the topology of the isosurface change? Generate an image called *IsosurfaceImage.jpg* (e.g. via *PaintShop*) of the isosurface at the critical point. Which of the two topologies would the Marching Cube code that you've been given generate?

[3 marks]

4. Also in the zip archive are two files *Pelvis.dat* and *Pelvis.vol*. These are CT scans of a human pelvis. Use the *Volume File* option from the *Volume* menu of *VolViz* to load the *Pelvis* data set – you should select the *Pelvis.vol* file (which is a text file describing the organization of *Pelvis.dat*) rather than *Pelvis.dat*. Generate isosurface images for both bone and skin. What isolevels did you choose? Get the images into image files *Bone.jpg* and *Skin.jpg*.
   Note that the way *VolViz* generates an isosurface from a sampled data set is a little odd. Normally we just "march the cubes" through the given data set. In *VolViz*, however, the given data set is used to provide a continuous field which is then resampled at the specified grid resolution to produce a new sample set for Marching Cubes. This has the advantage of allowing relatively fast previewing on a coarse grid, but it is obviously a waste of time to resample the data set when trying to display it at its maximum resolution. Also additional aliasing artifacts can be introduced. Don't worry about subtleties like aliasing effects in this final assignment question, but do be aware that you won't be able to get a smooth surface on the bone due to the way sampling and the gradient reconstruction process.

[4 marks]

Handing In

Submit via the electronic drop box all the java source files you have modified (Isosurface.java, TrilinearField.java, MassSpringSystem.java), plus the images for questions C.3 and C.4. Submit the answers to all the other questions either hand-written or in a doc/txt/html/pdf file electronically. Make sure that you have included the answers to the two questions buried in C.3 and to the question in C.4.

☺ *Enjoy!*