Programming Assignment #3

Due Date: Tuesday, April 29, 11:59 PM

Thus far, you have rendered virtual objects in the form of force fields and implicit surfaces defined by mathematical equations. These objects have had the nice property of simple collision detection.

In this assignment, you will see that an unstructured point set ("point cloud"), such as those acquired from a laser range scanner or an RGB-depth camera (e.g. Microsoft Kinect), can also be haptically rendered as an implicitly defined surface. However, some of these data sets can be very large, and you will need to implement a method for accelerating collision detection in order to meet the millisecond time allotment for haptic rendering.

This assignment is worth a total of 15 points, with an additional possibility of earning a "bonus" designation. Since the bonus is a binary state, it will only be awarded to submissions that do an exemplary job of meeting the requirements.

Getting Started

You will need to have completed assignment #2 before starting this assignment, as you will be using the implicit surface rendering algorithm you developed previously to render point sets now. Once again, start by making a copy of the "application" template from the CHAI3D package. Download the additional starting file archive (CS277-Assignment3-Code.zip) from the course web site and unpack it into your local folder, replacing the main application.cpp with the one provided again. The archive contains one additional class in the files PointSet.cpp/h, which you will have to add to your Visual Studio or Xcode project.

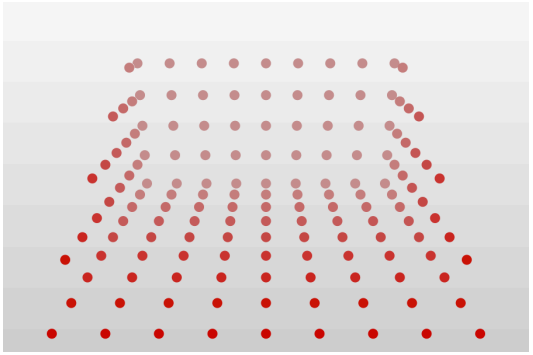
This assignment also comes with point set data files (CS277-Assignment3-Data.zip), which you will also want to download, unpack, and put into your project folder (and/or the CHAI3D bin folder, depending on where you normally like to run your program from. If things are in the right place, you should be able to run the application, which loads and visually renders the "ledge" (p1-ledge.ply) point set on the screen for you.

The starting source files are set up much as they were last time. The scene graph, which contains your interaction cursor and a solitary point set object, is created in application.cpp. The PointSet class derives from the cMesh class, works much like the ImplicitMesh class you used last time. In fact, you can probably start the assignment by copying your implicit surface rendering code from assignment #2 into the computeLocalInteraction() method of the PointSet class. The bulk of your work in this assignment will be to devise and implement a method for computing the implicit function and its gradient using contributions from the point set.

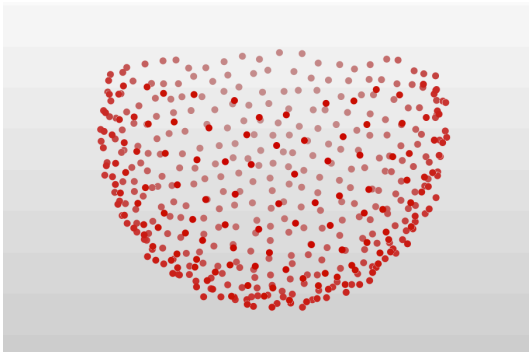
Part I - Rendering Unstructured Point Sets (4 points)

Use either the metaballs/soft objects (Wyvill *et al.* 1986) method or the point set surface approximation method (Adamson & Alexa 2003) discussed in class to haptically render the "ledge" (p1-ledge.ply) and "bowl" (p1-bowl.ply) point sets, shown below. These data are small enough that you can iterate through the entire point set at every time step to sum up contributions from individual points.

Select an appropriate radius of influence for the points and a threshold (if appropriate) so that you can feel a nice surface on each of the objects without popping through. You should be able to feel the flat surfaces and the corner on the ledge, and be able to touch both the inside and the outside of the bowl. Add to your program a means of switching between the different point sets.



ledge



bowl

Notes & Hints:

- The metaballs method may be a little easier to implement, test, and debug, as it more closely resembles mathematical functions you worked with in the previous assignment. However, if you intend to pursue the bonus, you will probably need to implement the point set surface approximation method.

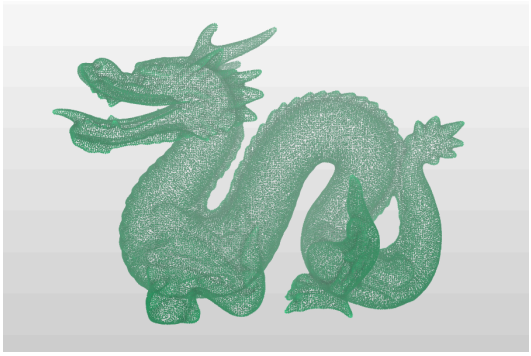
Part II - Accelerating Collision Queries (8 points)

Try rendering the "dragon" (p2-dragon.ply) point set now. Unless you have already taken steps to accelerate point queries, you will notice that the haptic rate drops dramatically. This is bad! (If you haven't yet experienced what slow haptic rendering feels like, here's your chance.)

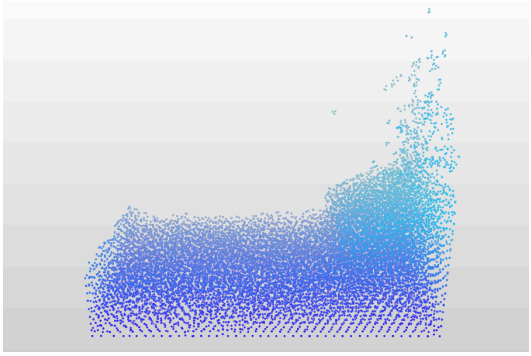
Implement one of the collision query acceleration methods discussed in class to improve your algorithm so that it can render the "dragon", "fluid" (p2-fluid.ply), and "desk" (p2-desk.ply) point sets, shown below. Your algorithm must be able to render all of these point sets at a haptic update rate of 1000 Hz or greater to receive full credit.

The point sets in this part have a much higher point density than those from Part I. Thus, you will have to adjust the radius of influence of the points appropriately in order to feel the surface features on these objects. Your algorithm should be able to render fine

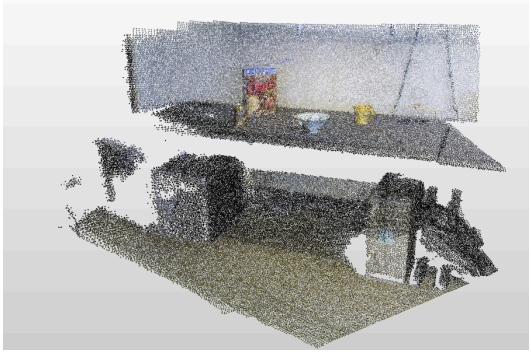
details in the surface geometry without allowing the avatar to pop through. Add a means to switch between rendering each of these three point sets as well.



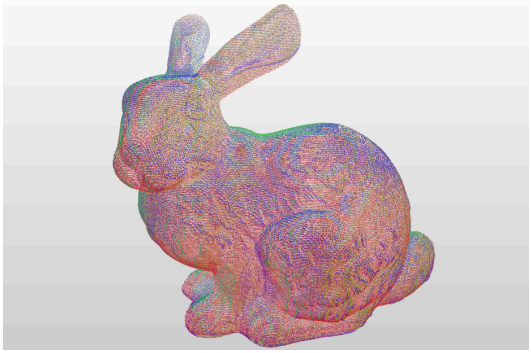
dragon



fluid



desk



bunny (bonus)

Notes & Hints:

- Think carefully about what query your acceleration data structure must answer. It may even help to write it down precisely in words before choosing a collision structure that can answer that query *efficiently*, or at least be modified to do so.
- If you like, you may examine the implementations of the collision data structures in the CHAI3D library or similar data structures from other sources. However, the implementation of the collision acceleration algorithm and data structure used here must be your own code. *Submitting code you did not write for this assignment is plagiarism, which is a violation of the Honor Code!* Please cite any external sources you referred to in a large capacity (but did not copy).
- These data may require a reasonably good graphics processor to render visually. If you notice your visual update rate (FPS) drop, try to find a more powerful computer to work on (if possible), or come talk to one of the instructors.
- Remember to test the "ledge" and "bowl" point sets again before submitting your assignment. You'll want to make sure that the improvements you made for Part II did not break the rendering of the objects in Part I!

Bonus Part - Variable Density Point Sets

Many unstructured point sets tend to have points spaced at a fairly uniform density, but sometimes we will encounter a point set whose density varies greatly from one location to another. These point sets can be significantly more difficult to render. Improve (or rewrite) your point set rendering algorithm to render the "bunny" (pb-bunny.ply) and "room" (pb-room.ply, not shown) data provided.

The bunny point set is a combination of multiple range images captured by a Cyberware 3030MS optical triangulation scanner, with each distinct image shown in a separate color. You should be able to touch and feel fine structural details on the surface of the bunny near the eyes, nose, ears, and near its feet. The avatar should not pop through the surface during haptic interaction, although there are two real holes on the bottom of the bunny that the avatar may legitimately pass through.

The room point set is a merged sequence of RGB-Depth images of a living room captuerd with a Microsoft Kinect. You should be able to feel the flat surfaces of the sofa, chairs, floor, and table. Even though they are quite small, you should still be able to touch the objects on the coffee table. Naturally, there are gaps in the data where the camera could not see, and it is fine if the proxy passes through those.

The bonus will only be awarded to submissions that do an exemplary job of rendering these additional real-world point sets. Note that these points sets are quite large, so you may need a reasonably powerful graphcis processor to render them visually.

Assignment Questions (3 points)

These questions help you think about various aspects of implementing the assignment, and it may be helpful to read through and think about them before starting on your code. You may include a separate file in your submission with your answers, or simply append your answers to your "readme" file.

1. With the methods we studied, the quality of haptic feedback for these unstructured point sets can depend heavily on selecting an appropriate radius of influence for the points. What happens if you use a radius that is too small? Too large? Briefly comment on how you selected your radii in this assignment.
2. Think about the method you chose to render the point sets (i.e. metaballs vs. point set surface approximation). Of the point sets provided, is there a specific one that your method performs poorly with? What about one that it renders particularly well? Also comment on whether you think the other rendering method would have similar strengths and weaknesses, or different ones.
3. In Part II, what algorithm or data structure did you choose to use to accelerate collision queries? Briefly describe your reasoning for choosing the method you did. If you are rendering a point set containing n points, what would be the expected and worst case time complexities of your algorithm for answering a collision query? What is the running time for constructing your data structure?

Submitting the Assignment

You should normally just submit the source code for any file(s) you modified or created. Please do not send us back the CHAI3D libraries or any other large compiled binaries! We will compile your program with a pristine copy of CHAI3D when we grade your assignment. We clearly have a copy of the point set data files as well, so please do not include copies of those with your submission.

Include a "readme" file with your submission. Please indicate which platform and development environment you used to do the assignment, in case we run into any problems compiling or running your program. Also give brief instructions on how to use your software, and any gotchas we may encounter while grading it. You may also write in this document anything else you'd like us to know about your submission.

Completed assignments should be emailed as attachments to cs277.2014@gmail.com before 11:59 PM of the due date. Please indicate any late days used. If your files are abnormally large, you may send a web download link, drop off a copy on a flash drive, or make a suitable alternate arrangement.

References

Leeper, A., Chan, S., & Salisbury, K. Point clouds can be represented as implicit surfaces for constraint-based haptic rendering. In *Proc. IEEE International Conference on Robotics and Automation*, 5000–5005 (2012).

Wyvill, G., McPheeters, C., & Wyvill, B. Data structure for soft objects. *The Visual Computer*, 2(4), 227–234 (1986).

Adamson, A., & Alexa, M. Approximating and Intersecting Surfaces from Points. In *Proc. Eurographics Symposium on Geometry Processing*, 230–239 (2003).

Data Sources:

- The "dragon" and "bunny" data sets were obtained from the Stanford 3D Scanning Repository (http://graphics.stanford.edu/data/3Dscanrep/).
- The "desk" and "room" data were obtained from the University of Washington's RGB-D Scene Dataset (http://rgbd-dataset.cs.washington.edu).
- The "fluid" data is the result of a smoothed particle hydrodynamics simulation provided courtesy of the High Performance Computing Center North (HPC2N) at Umeå University in Sweden.