Progressive Delta Edits on Subdivision Surfaces

Laura Devendorf
Final Project CS 284 – Fall 2012
Professor Carlo Sequin
https://github.com/Devendork/DeltaChangeSubdivision.git

Abstract

This project explores ways in which global and local changes can be made on subdivision surfaces by adding offset values to vertices at various levels in the subdivision. For instance, one can make edits to the overall shape of the object by adjusting vertices on the control mesh of an early subdivision step and small local details can be created by adjusting vertex positions at later subdivision stages. A GUI is provided to interface with the algorithm and assist in the free-form modeling of objects.

Introduction

This project is motivated in part by a lecture I attended at the FabLearn conference held at Stanford in October of this year. The conference was devoted to looking at ways in which fabrication technologies could contribute to school curricula. Mike Eisenberg, professor at CU Boulder, presented a talk that took a critical look at the way the fabrication tools we use now and discussed how we could use these tools specifically in the context of learning. His example focused specifically on 3D printing and argued that going to a website, downloading a model, and 3D printing it was not a sufficient learning experience. He encouraged researchers to look at ways in which students can engage with the models, create narratives with them and form more immersive learning experiences that leverage both the technology and what fundamental concepts we can learn from such technology.

In this project, I attempted to build a GUI interface for modeling that could be accessible by someone in the 8th grade. Throughout the design and implementation of system, I focused on accessibility and user-feedback. What I have built is moving toward giving younger students the ability to create and modify models that they send to a 3D printer for manufacturing.

Implementation

The system I build consists of a graphical user interface and underlying code that allows users to add progressive delta edits to an existing mesh.

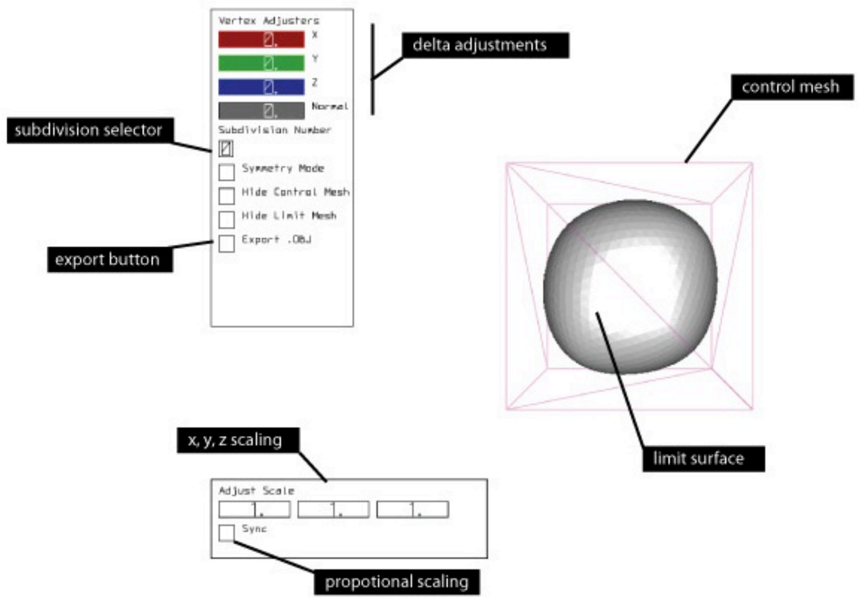

Figure 1: GUI overview

Graphical UI
The graphical elements such as sliders and buttons were implemented using GLV (Graphics View Library). More information on GLV can be found at http://mat.ucsb.edu/glv/

Each of the user interface elements is explained below and shown in Figure 1:

- **Limit Surface**: The limit surface shows the final form the shape will take on after 5 subdivisions. This is the model that can be exported and printed on a 3D printer.
- **Control Mesh**: The control mesh represents the mesh at some user-defined level of subdivision. The user selects points on this control mesh in order to manipulate the limit surface.
- **Subdivision Selector**: moving this number between 0 and 5 will reveal what the mesh looked like at that particular subdivision step. At each subdivision step, the user can select points on the control mesh to adjust the shape of the limit surface.
- **Delta Adjustors**: These sliders allow the user to move a selected vertex or vertices along the X,Y, and Z axis. The colors of the sliders correspond to the color of axis that is visualized on top selected points. An additional slider allows the user to move the point along the vertex normal.
- **XYZ Scaling**: Adjusting values in the X, Y, Z scaling boxes allows one to globally scale the model in the X, Y or Z direction according to the value selected in the number selector box.
- **Proportional Scaling:** If one wishes to scale the shape evenly in all dimensions, they can click the box labeled "Sync" in order to make all of the scaling values the same and proportionally scale the mesh.
- **Symmetry Mode:** If the symmetry mode button is checked the then control mesh visualization changes to allow the user to select a face. Faces that are neighbors and co-planar will automatically be selected along with the selected face. Selecting a face in symmetry mode mirrors the mesh across the plane defined by the face normal. The symmetry operation is intended to take place before other delta edits. After a plane of symmetry has been defined, the program will ensure that if one vertex is selected, it's symmetric vertex or vertices are also selected.
- **Hide Control Mesh:** Clicking this button will hide the control mesh allowing the user to visualize and inspect only the limit surface.
- **Hide Limit Mesh:** Clicking this button will hide the limit surface so that the user can see the control mesh more clearly. This is helpful in scenarios where the limit mesh occludes the control mesh. In order to visualize the structure of the control mesh more clearly and assist in point selection, the triangles of the control mesh are drawn to be opaque when the limit surface is not being rendered.
- **Export OBJ:** Clicking the export .obj button exports a file representing the limit surface in obj format.
- **Mouse Controls:** The user can drag the mouse in order to move the camera around the object. If one drags the mouse while the right click button is depressed, the camera will zoom in and out of the scene. To select vertices to adjust, the user can hover and click over vertices in the control mesh. Using the delta adjustors, they can modify and visualize the mesh as they do so. To deselect vertices, one can hit click on a selected vertex to deselect it.
- **Key Commands:** Pressing the 'x' key will automatically deselect all vertices.

Algorithm
Charles Loop described the subdivision scheme implemented in this project in detail in his thesis entitled, "Smooth Subdivision Surfaces Based on Triangles." Given a triangle mesh as an .obj file, the program reads the input and constructs an initial mesh. To construct the levels of subdivision, the program copies the previous level mesh, subdivides according to Loop's rules, and then offsets any vertices by "Delta," and updates data associated with the mesh (vertex normal, face normals, etc). A "Delta" object stores the offset of the vertex as an x,y,z position. When a vertex is offset by its normal, the x,y,z position is derived by multiplying the magnitude of movement along the normal by the normalized vertex normal.

A class entitled "MeshManager" stores a pointer to a mesh at every level of subdivision. When delta edits are added to the control mesh at stage X, the MeshManager updates all subdivisions for stages later than X. Currently, this step proceeds by calling the subdivision process over again with the new point positions but it could be accomplished more efficiently by constructing a dependency graph between vertices at varying levels of the subdivision.

When a user defines a plane of symmetry, the face normal of the selected face is used as a plane of reflection. All points in the mesh and not on the plane are reflected across the plane. The mesh stores an adjacency list of vertices in the original mesh to the corresponding vertex on the symmetric mesh. Using this information, faces are defined by adding faces that are mirrors to the original faces and adjusted to maintain a counter-clockwise orientation. The face and vertex data of the symmetric mesh run through a function that constructs adjacency information for vertices and faces.

To support open meshes, slight adaptations were made to Loop's rules. I use the same formulation for positioning vertices. In the case where the number of incident edges is 3, the mesh shows some visible warping which can be offset by a delta change (see figure 2). To place a vertex along an edge that is on a boundary, the new vertex position is obtained by using the existing faces information in place of the missing face. In practice, this didn't lead to any noticeable deformations in the mesh.

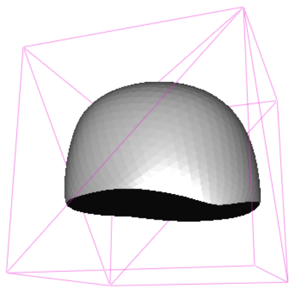

Figure 2: An open mesh in the GUI. Notice a "dip" in the boundary where the number of edges incident a vertex was 3

Building with Delta Edits:
The process of modeling a surface with delta edits on subdivision surfaces leads to a process of free-form modeling. Having real-time feedback was crucial in visually constructing a surface and making interesting new forms. What I particularly enjoyed about the application was the ability to model surfaces that mimicked the shapes and properties of clay. Freedom in the ability to make delta edits leads to a freedom to form shapes in an improvisational fashion, leaving room for unexpected results and surprises to arise. In Figure 3, I outline the process of constructing a shape using the GUI and algorithm I developed.

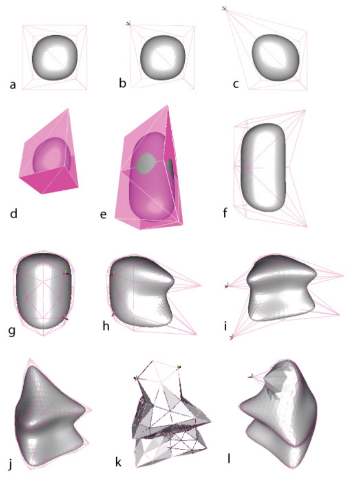

Figure 3: a) the original mesh is loaded. b) a point is selected along the original mesh. c) the point is adjusted using the delta sliders. d) symmetry mode is enabled. e) a face is selected for a symmetry operation. f) symmetry mode is turned off. g) a point is selected and the symmetric point is automatically selected as well. h) the points are adjusted by augmenting the value along the normal. i, j) a second and third point is selected and adjusted along the normal. k) the "Hide Limit Mesh" button is pressed making the control mesh opaque in order to ease selection. l) A finishing touch is added.

Design Challenges
As I dove deeper into the development of the interface new challenges continued to arise. The particular method of implementing symmetry that I chose may be more limiting than helpful. An alternative scheme to select any three points that define a plane and then slice and reflect the mesh across that plane was suggested as a more constructive solution. Similarly, using the suggested scheme would make rotational symmetry an equally feasible option that currently isn't available.

One challenge I faced in particular was due to my lack of experience building interfaces from scratch in OpenGL. I must admit that I underestimated the challenge of this task and much time went into debugging picking and camera operations that a more experienced user may have been able to handle with greater ease. The user interface tools are sufficient for getting the point across that certain operations exist but are not quite at a level to be accessible to an 8th grade student. Preforming studies with this target user group would be instrumental in developing robust interface solutions.

Visualizing the mesh at deeper levels of subdivision becomes a challenge since the points fall very close to one another. Better tools for zooming in on dense regions of points which maintaining a vision of the over all surface using local and global views could be helpful in solving this problem.

Conclusion
I present an algorithm and user interface that allows for delta edits on subdivision surfaces. The process and interface allow for a kind of free-form modeling that could be instrumental in creating more involved and creative interactions on meshes. Real-time updates of how the limit mesh changes as a result of changes at a particular level of subdivision produces a new familiarity with the algorithm and its processes. I am confident that the ability to make these new familiarities and visual understandings of algorithmic processes could be a positive contribution to discussions on how we learn from using CAD tools.