2D sketch to 3D Transform

Team 07

ANANYA GUPTA and SAUMIL LAKRA

1 INTRODUCTION

The advent of 3D modeling systems has undeniably revolutionized the way we create digital objects and designs. However, the complexity of these systems has made the process of designing free-form surfaces, such as stuffed animals and rotund objects, a challenging and time-consuming endeavor.

This paper introduces an innovative solution – a sketching interface that enables the rapid and intuitive creation of 3D models. The core concept of this interface revolves around the use of free-form strokes as a means of expression in design. Users can effortlessly draw 2D free-form strokes on the screen, outlining the silhouette of the desired object. With a few strokes, the system automatically generates a plausible 3D polygonal surface model, eliminating the need for users to manipulate control points or engage in complex editing operations. Even those new to 3D modeling can swiftly create expressive and intricate 3D models with this system. Moreover, the resulting models possess a distinctive hand-crafted quality that is often elusive in traditional modeling software. This paper delves into the details of the sketching interface and the underlying algorithms for transforming 2D strokes into 3D shapes.

Additionally, it discusses the implementation of the prototype system, Teddy, which facilitates real-time mesh construction on standard PCs using Java. While Teddy is designed for the rapid construction of approximate models rather than precise editing, it boasts an intuitive and engaging pen-and-ink rendering interface that encourages creative exploration. The applications of Teddy extend beyond character animation, promising to revolutionize various domains. Its user-friendly approach has the potential to facilitate rapid prototyping in design, foster educational and recreational use among non-professionals and children, and enhance real-time communication on pen-based systems. With its ease of use and broad potential, Teddy represents a significant step forward in 3D modeling technology

2 LITERATURE REVIEW

Teddy allows real time rendering without a dedicated 3D rendering hardware. The paper aims to produce a 3D object out of a 2D object drawn on the screen(canvas). The first step would be to render the mesh structure of the object(Delaunay Triangulation[3]) and then find the spine of the shape using the chordal axis[2]. This would give us the spine of the shape(prune insignificant branches) drawn. Now we elevate the spines connecting the new points to the edge vertices of the shape. We repeat this process to generate the bottom volume of the object.

To implement rotation we use the same process we did in assignment 2(u,v,w vectors and rotate the camera from a fixed distance(can be changed) from the object. To draw curves on the surface i.e. to implement painting on the surface, we project the curve on the surface of the polygon and subdivide it into segments,one for each polygon. Then we splice the segments by using a camera over a bounded plane.

For extrusion, the 2D extruding stroke is projected onto a plane perpendicular to the object surface and the base ring is swept along the projected extruding stroke.The plane for projection is found using the base ring's center of

Authors' address: Ananya Gupta, ananya21012@iiitd.ac.in; Saumil Lakra, saumil21097@iiitd.ac.in.

gravity and normal of the base ring. Copies of the base ring are created along the extruding stroke, perpendicular to the direction of the extrusion, and are resized to fit within the stroke. This is done by advancing two pointers (left and right) along the extruding stroke starting from both ends[1]. The cutting algorithm is based on the painting algorithm. Each line segment of the cutting stroke is projected onto the front and back facing polygons. For smoothing,First, the system translates the objects into a coordinate system whose Z-axis is parallel to the normal of the ring. System creates a 2D polygon by projecting the ring onto the XY-plane in the newly created coordinate system,and triangulates the polygon.Then each vertex is elevated along the Z-axis to create a smooth 3D surface.[4]

3 MILESTONES

S. No.	Milestone	Member
	Final evaluation	
1	Implementation of canvas (Done)	Ananya Gupta
2	Delaunay Triangulation (Done)	Saumil Lakra
3		
i)	Find and elevate the spine points (Done)	Ananya Gupta
ii)	Complete the object (Done)	Saumil Lakra
5	Rotation (Done)	Saumil Lakra
5	Painting	Ananya Gupta
6	Cutting	Ananya Gupta
7	Extrusion	Saumil Lakra
8	Smoothening of object (Done)	Ananya Gupta

4 APPROACH

4.1 Implementation of canvas

We took user defined control points through our interface and stored them in our points vector. This helped us build the outer boundary of the 2-D figure we were constructing.

4.2 Delaunay triangulation

Next,we imported a library called Triangle in our code. Triangle has functions which aid to perform constrained delaunay triangulation(CDT). We first removed the duplicate control points with the help of set in cpp in a seperate function and added them to a new array called controlPointswithoutdups. Then we described our triangulate function to generate a node or poly file. This file stores all points, their attributes, whether they are boundary points and segment information as well. It gets generated automatically each time the project is run. Next we prescribed some keys for specific functions like left mouse button for adding points, right for moving the camera and tab button to do triangulation after atleast 3 points are drawn. Next we described our readele function which reads the .ele file generated line by line and pushes the points in a vector in a set of 3 as per the triangles constructed in triangulation.

4.3 Finding the spine and elevation

Once our 2-D figure has been triangulated using CDT, we have to find its chordal axis and spine- (press escape) and triangulate it further-(press tab). So to find the spine we first made a function to check whether an edge connecting two points was internal or not. It was internal if it was common between two triangles(which we iterated through our readele vector). Next we found out the midpoints of all the internal edges and then constructed a set to avoid duplicacy. These midpoints were pushed in the spine vector as well and displayed on screen. On pressing tab, further triangulation could be made as mentioned in the paper.

4.4 Pruning

Before the elevation of the spine, pruning has to be done to get fan triangulation which are required when the inflation algorithm has to implemented.

4.5 Completing the 3D mesh

Next, we had to elevate the spine. So we shifted our z coordinate of the spine by a fixed distance symmetrically up and down the normal, to get two new points. This distance was the same as that to the external vertices. Then we drew circular arcs to interpolate the elevated points to the external vertices. This formed a basic structure of our 3-D mesh. Then we triangulated all the points on the arc and did it symmetrically down as well. This lead to our construction of the 3D mesh.

5 CHALLENGES

We faced many challenges which first and foremost was understanding how the triangle library worked. We had to research on the syntax followed. Because of our main while loop running again and again there was duplicacy in our points which we had to resolve in order to correctly generate our poly file. To solve this challenge we used the set data structure to parse these points beforehand.

Second, we faced challenge of CDT while reading .ele file. The triangulation seemed to work sometimes and other times it wouldn't especially when we increased our number of points. Then figuring out how to perform delaunay triangulation by exploiting the poly and ele files.

Converting the code to 3d (Mesh assignment) was challenging and to configure the points entered by the user and make sure the world coordinated and mouse coordinates are same was difficult.

Pruning algorithm was posing problems when executed, we tried implementing it by first finding all the internal edges of the surface. Then for each edge we tried to find the vertices that lied on its side that contained its 3rd vertex. Then for each internal edge we tried to find if there were points that lied outside of the semicircle drawn by taking the internal edge as parameter. The issue was faced here was to identify an algorithm that could guide us to traverse all the internal edges in a sequential order (The code for this is present in prune.cpp)

REFERENCES

Igarashi, T. (1999). Teddy: A Sketching Interface for 3D Freeform Design. In Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99), pp. 409-416.