Interactive Computer Graphics
Final Project Report

TeddyCam

A Sketching Interface for 3D Freeform Design

陳勁宇、林湧達、陳艾苓

Abstract
Education for preschool children becomes highly mentioned nowadays. As a result, in order to enhance the creation ability of the children, the importance of do-things-in-hand increases. The old school way to achieve the goal like using LEGO may be a good but costly solution. A novel way of in-air sketching is presented in this report. To encourage children to create all kinds of objects within low cost, our program provides a platform for kids to do freeform sketches. Result of this project shows that a reasonable model can be generated, and the operation of mesh construction can be done within 3 seconds.

Keywords: Teddy System, Unity 3D, Poly2Tri, computer graphics, virtual Reality, augmented reality, 3D modeling, wand-based, gestures, chordal axes, inflation

1 INTRODUCTION

Teddy System [1] is a noted work of Takeo Igarashi in 1999, and it makes users draw 2D strokes on the screen and construct 3D models in real time. In this project, we combine the in-air sketching method and Teddy System.

By using the webcam to capture the red object holding by the users, they can draw freeform strokes in air and then get the inflated models of the silhouette surrounded by the strokes. We implemented this work with Unity™, and we construct a scene for our prototype system. In the scene, the world is still under construction, and the users are set to be the God, the Creator, and they can overlook the Wild from a Sky Window and make rocks in any shape drawn by his or her magic wand. In addition, the rock will drop to the ground immediately after created. (Figure 1)

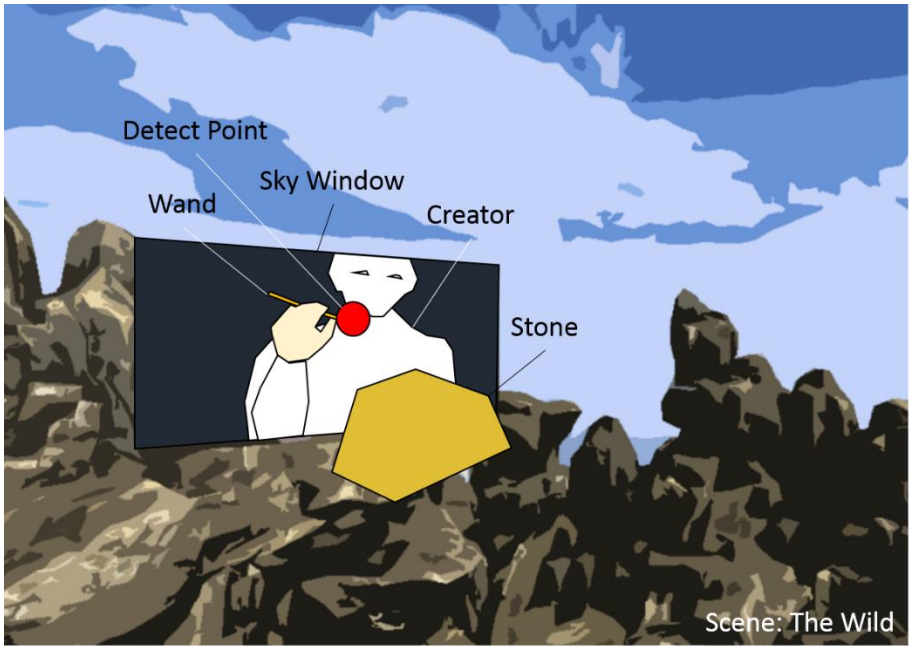


Figure 1: Our platform

2 MOTIVATION

In the beginning, we come up with an idea that what if one can draw any freeform object by fingers in the air and throw it out like a snowball. We thought it would be an amusing interaction that you can throw something VISIBLE on somebody but harmless. On the other hand, it seems that you got the magic power.

To achieve this goal, we planned to do an in-air interactive system based on Augmented Reality. After discussing with our instructor, we got to know that Teddy System would be a great reference for us. In the end, we decided to implement the Teddy System but with the in-air sketching method. Nevertheless, this project is the first step for us, and this is more like a prototype of our original idea.

3 ALGORITHM

3.1 Drawing

In order to detect user drawing, the user has to stick a red object to his finger, then we can detect what the user draw through tracking red object. We use unity's native WebCamTexture, multithreading and Opencvsharp [2]. First, turn on the webcam, and keep on capturing frames from webcam. We use threshold operation provided by OpenCV. Before analyzing captured image, we define the threshold like a filter, if the pixel in the image over the threshold, we regards it as a red object, then filter it to white color. Therefore, we can track red object from webcam.

3.2 Polygon to Triangles

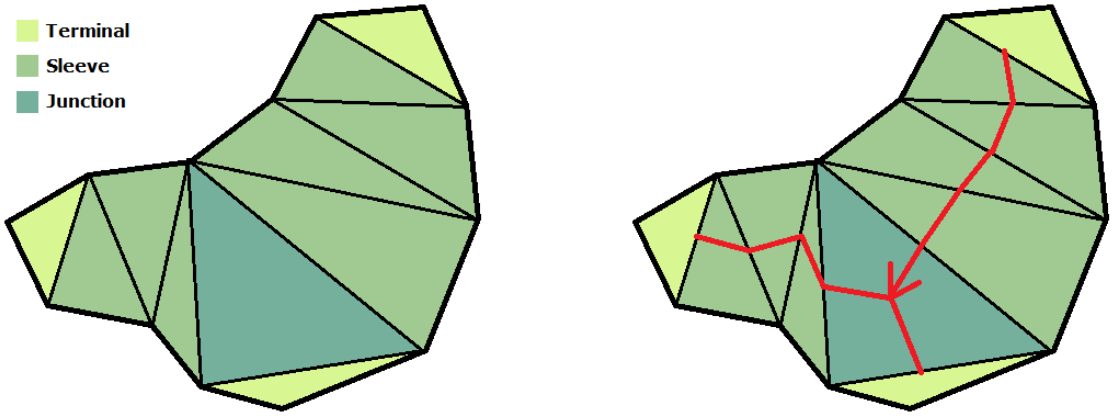
In order to show meshes in Unity3D, the detected polygon (Figure 2a) needs to be transformed into a collection of triangles (Figure 2b). The triangles generated here are not the ones that is put into the meshes, since we may modified them later. For example, pruning the branches will replace some triangles, rearranging the triangles will cut some triangles into pieces, while elevation changes the z-axis value of some triangles. The tool we use in this step is Poly2Tri [3], which is an open source. Instead of using the compliable source code, our implementation prone to use the binary executable file, and execute it inside Unity3D using C# script.

a) Input polygon b) Triangulated polygon

Figure 2: Poly2Tri

3.3 Chordal Axis

A novel way of finding Chordal Axis is presented in this report. The triangles are classified into three categories, Terminal triangles with two external edges, Sleeve triangles with one external edge and Junction triangles with zero external edge (Figure 3a). First we run through all the triangles to find the Terminal triangles. For each Terminal triangle, a track started from it to a Junction triangle is generated by finding the next connected triangle iteratively. After all tracks are built (Figure 4), an alternative Chordal Axis is finished (Figure 3b), with Sleeve triangles between two Junction triangles are excluded. Notice that a single Junction triangle may be the ending node of several tracks.



a) Types of triangles b) Chordal Axis

Figure 3: Chordal Axis

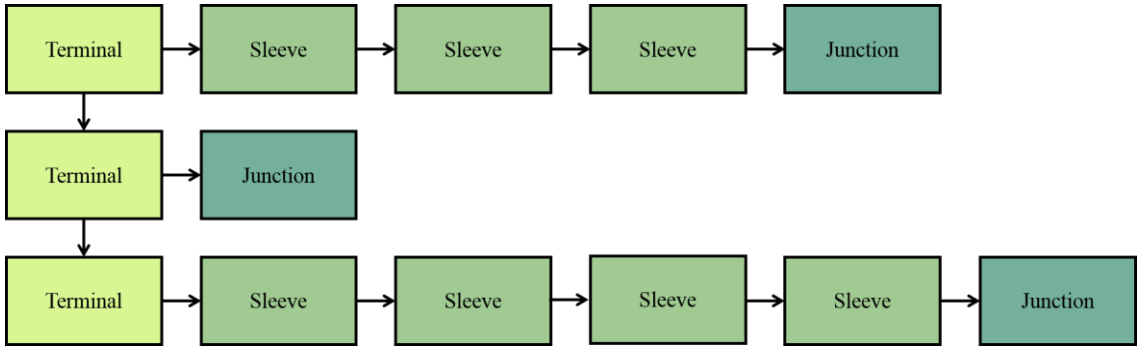


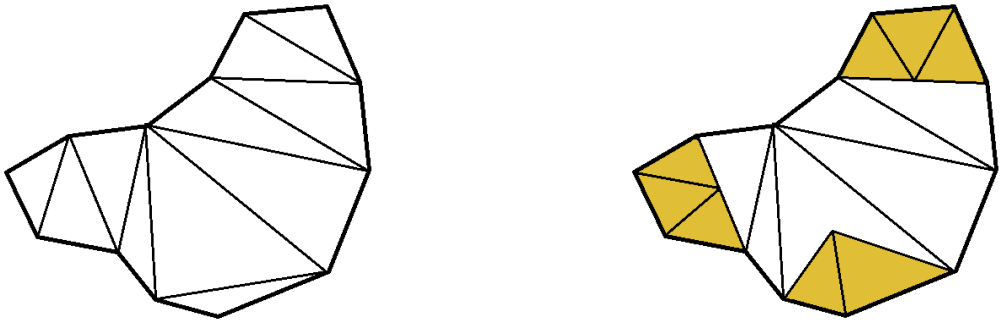
Figure 4: Trace table data structure

3.4 Pruning Branches

To make some sharp edges smoother, the Chordal Axis track built in 3.3 must be traced. Starting from a Terminal triangle, we draw a semicircle with a diameter based on the interior edge of the triangle. If all the vertices of this triangle lies in or within the semicircle, then the interior edge is removed and the triangle is merged with the next connected triangle, which generates a polygon. We do the same operations on the polygon, including drawing a semicircle using the interior edge and evaluate the positions of all vertices with respect to the semicircle to determine whether to merge with the next triangle. This operation will iterate from a Terminal triangle and through

several Sleeve triangles, and ended when (1) meeting a Junction triangle or (2) if any single vertex lies outside the semicircle.

To pare rough edges, the tracked triangles will be replaced by a fan, which is also composed of several triangles, only with a different arrangement. For the (1) ending case mentioned before, the fan is radiating from the center of the Junction triangle, while the fan is radiating from the midpoint of the last interior edge in the (2) case.

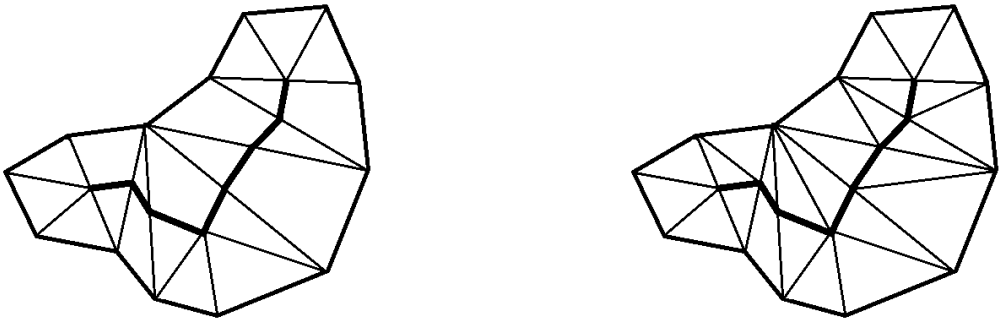


a) Original triangulated polygon b) Pruned polygon

Figure 5: Pruning Branches

3.5 Rearranging Triangles

The spine goes along the midpoints of the interior edges from the radiation point to the center of the Junction triangle, and also between the centers of two Junction triangles (Figure 6a). The triangles are then divided by the spine, which will generate some quadrangles. In order to build meshes in Unity3D, each quadrangle is cut into two triangles (Figure 6b). Also, all points along the spine are added into a list for the preparation of elevation in 3.6.



a) Spine of the triangles b) Rearranged triangles

Figure 6: Rearrangement

3.6 Elevation and Mesh Generation

For each point along the spine, the distance between all interior edges from it to the exterior vertices are evaluated, and the point is then elevated along the z-axis by the average of all these distances. Now all triangles are generated in one side. Finally, a closed and symmetric model is retrieved by simply reflecting the triangles to the other side.

4 RESULTS

Using Unity3D as our platform, we bind the algorithm through 3.2 to 3.6 in a GameObject with MeshRenderer. The GameObject is then set as a prefab, so that multiple meshes can be rendered. Our program works as follow:

(1) Hold A when drawing in the air with a red object in hand (Figure 7)

(2) Press S to start render the model (Around 3 seconds to complete, depends on the size.) (Figure 8)

Due to the limitation of input format in the Poly2Tri binary executable file and the feature of polygon, the drawing gesture must be in clockwise rotation without intersections.

Figure 7: The God is waving his wand!

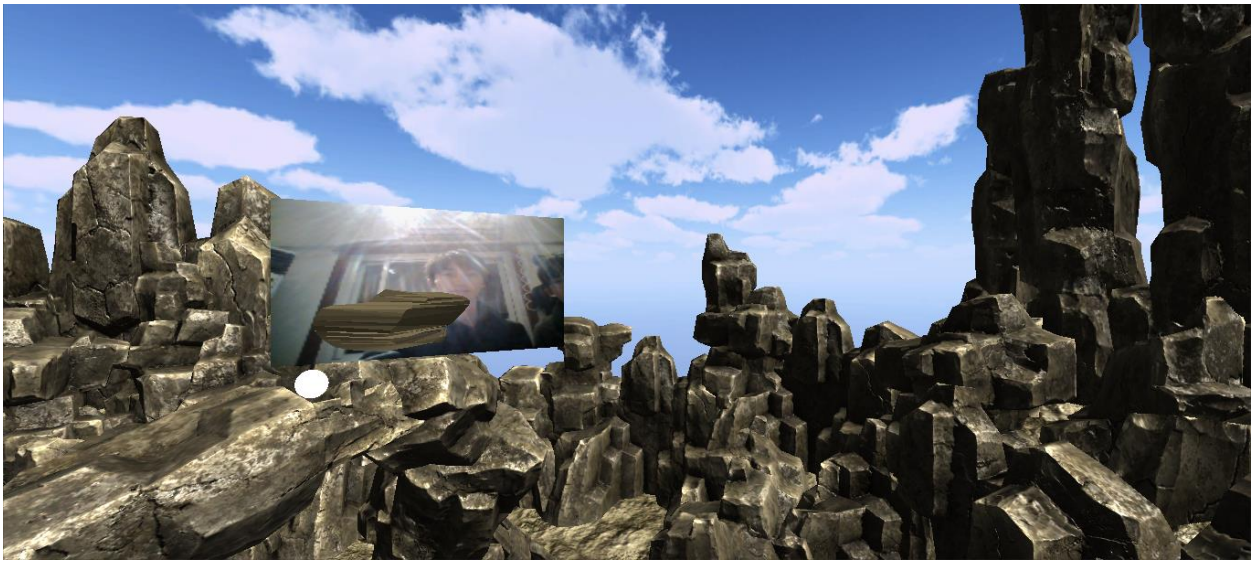


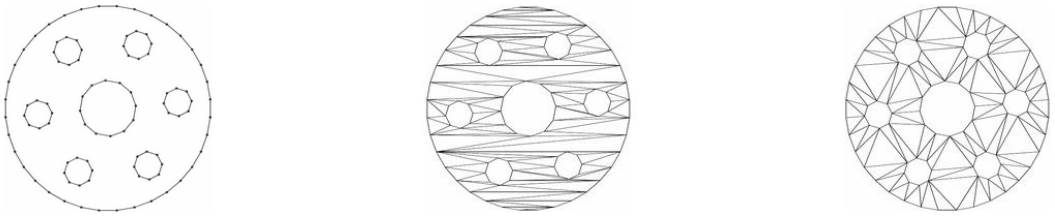
Figure 8: The stone is generated!

5 DISCUSSION

From the result, we can see there are still some limitation in our work. For the input, the in-air drawing method makes the users have poor control of the drawing shape, especially when the trace doesn't display on the screen. The detection of the red-spot is actually not so accurate, and the resulting point can be influenced by the environment light or the resolution of the webcam. As a consequences, the detected spot may vibrate and make the resulting shape far from smooth.

As for the output, we can notice that the resulting model differs from the result of Teddy System, because the Poly2Tri binary executable we used here is not the constrained Delaunay triangulation (CDT) (Figure 9c) but normal triangulation (Figure 9b).

Lastly, the execution time is 3 seconds. The main cost is due to the file IO of writing file for the Poly2Tri executable and read the output file. If we use the CDT library, then we can kill two birds with one stone.



a) Input polygon b) Normal triangulation c) CDT Triangulation

Figure 9: Comparison of two triangulation[1]

6 FUTURE WORK

To achieve a better user experience, the drawing track should be displayed on the screen to let users have better control of the drawing shape. We can also use end-point detection rather than red-spot detection, and this can easily be done with depth camera or Leap Motion™. With the end-point detection, the magic power would significantly upgrade to "wandless" level. On top of that, we can implement the system on head-mounted display like Oculus™ or other glasses, which will make the system be more like an Augmented Reality.

In the modeling part, first, CDT library can be imported to make the models more reasonable than not resemble the stones. Second, after the inflation, the edges are still not replaced by quarter oval, so our result is still polyhedron. Last, the motion of the created model is only gravity now, and users cannot have further interaction with the model. We hope the animation or the ability to throw the model can be added to the models in the future.

[1] http://sites-final.uclouvain.be/mema/Poly2Tri/

7 REFERENCE

1. IGARASHI, T., MATSUOKA, S., AND TANAKA, H. 1999. Teddy: A sketching interface for 3d freeform design. SIGGRAPH, 409–416. ISBN 0-20148-560-5. Held in Los Angeles, California.
2. https://github.com/yura415/unity-opencvsharp
3. Wu Liang, “Poly2Tri: Fast and Robust Simple Polygon Triangulation With/Without Holes by Sweep Line Algorithm”. Centre for Systems Engineering and Applied Mechanics (CESAME) 2005, University Catholique de Louvain, http://sitesfinal.uclouvain.be/mema/Poly2Tri/