3D SKETCH-BASED PIXEL WORLD

QIYUE ZHANG (10131658)
JIAN LI (30023214)

Abstract. This project focuses on the recreation of natural landscapes through 3D sketching, by using strategically placed cameras to capture the movement of a pen with a specially colored tip.

1. introducion

In this project, we have created a program that models elements found in natural landscapes. The program takes a file containing a set of data points as input, and models them as different elements such as clouds, trees, and leaves based on the modeling technique, colors, and points specified in the file.

2. proposed goal

Our goal was to create a sketch-based 3D modeling application through the use of iPhone cameras and a simple pen tool. The application will enable users to create their vision of a natural landscape with minimal knowledge of the back-end, using equipment that is easy to obtain and set-up.
We plan to get 3D point data by using two iPhones of the same model. By placing them on planes perpendicular to each other, the appearance of 3D space can be achieved.
The drawing tool to be used may be any pen or stick with a red tip, so that the iPhone cameras may look for a specific set of RGB values to collect on and ignore any other values. After obtaining the data, the coordinates will be passed to the C++ application through TCP packet. The C++ application will render data set by applying the proper tool-kit API to be built. The whole model should be able to be output as a raw file and take a the file back as input for further sketching.

3. proposal vs. final project

In the initial proposal, we laid out the basic idea of the application we wanted to achieve, as well as some general requirements, and while the general idea of modeling landscapes has remained the same,there have been changes that we decided to make

once the actual implementation have started.

Section 3.1. Consistencies

3.1.1. *Modeling*

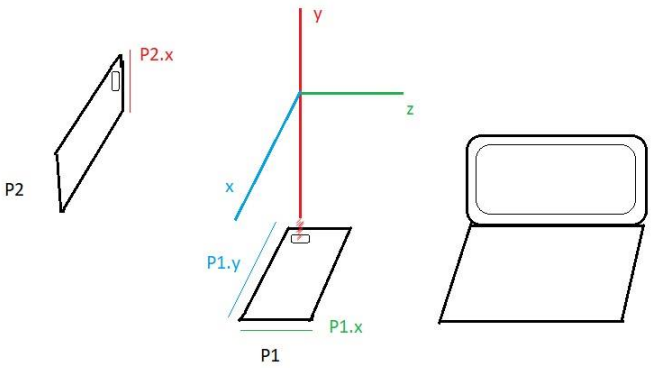
We were able to use a given set of point data to model elements of natural landscapes such as land, trees, and grass. This is consistent with what we initially set out to achieve. However, due to the lack of time, we chose to not apply texture mapping to our elements.

3.1.2. *Data Set I/O*

In our proposal, we wanted to have the user be able to export the scenes that they have drawn and import the data back to work on it at a later time. We were able to make special files containing data sets that can be read in as renderable objects. However, due to some unavoidable changes, we no longer support the exporting of data sets.

Section 3.2. Differences

The main difference between our final deliverable and the proposal is that we have decided to spend small part of time on the user interaction of using cameras to capture our data, but instead focus on using a special custom file containing the data needed for some renderable object(s) that can be imported and rendered.
Initially, we had the idea of setting up the two iPhones so that one is face-down on a flat surface with its camera facing up, and the other is held up, perpendicular to the first phone, as shown below.



However, since the drawing tool that we are is a pen that is completely red, we

ran into some troubles with detecting which part of the pen was the tip. In the end, we tried to solve this issue by reading pixels from the first rows in the x and y axes of the iphone $P1$ and making that the data point reflecting the tip of the pen. However, this method soon proved itself to be unstable as it makes the human interaction awkward, and makes errors and inconsistencies inevitable.
Therefore, we have since decided to get out input from a custom input file, and drawing based on data read from the file.

4. IMPLEMENTATION

Section 4.1. Technology Used

In our attempt to obtain camera data, we used the video stream from iOS hardware to get the frames captured from which we can read and process the data.
After we decided to remove the iPhone camera component from the project, we wrote custom files that can be processed to model different objects. This file contains data sets with the following information in each set:

- Sets of control points to be processed as a set of curves
- A variable specifying the modeling technique to be used
- Any transformation (translations, rotations, scalings) that should be applied to the object
- Any additional operations such as duplicate last object, object coloring, and make new object

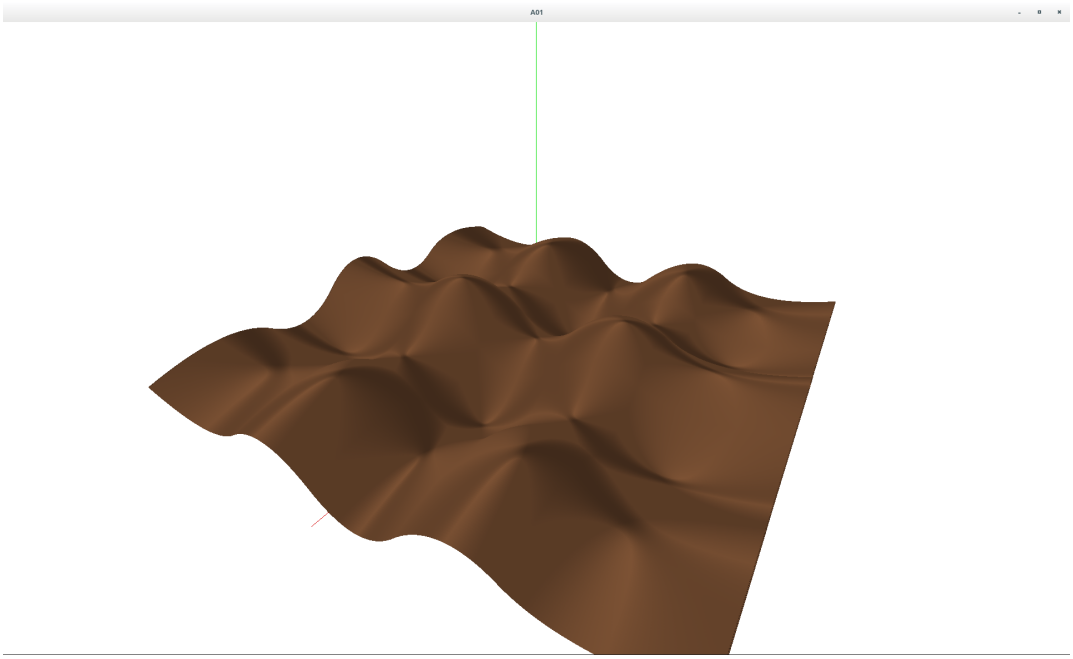
Section 4.2. Modeling Techniques Used

In order to model our objects, we used mainly surface modeling techniques learned in class, specifically, we used many curve-based surfaces, with B-spline curves as the basis curves of the surfaces.

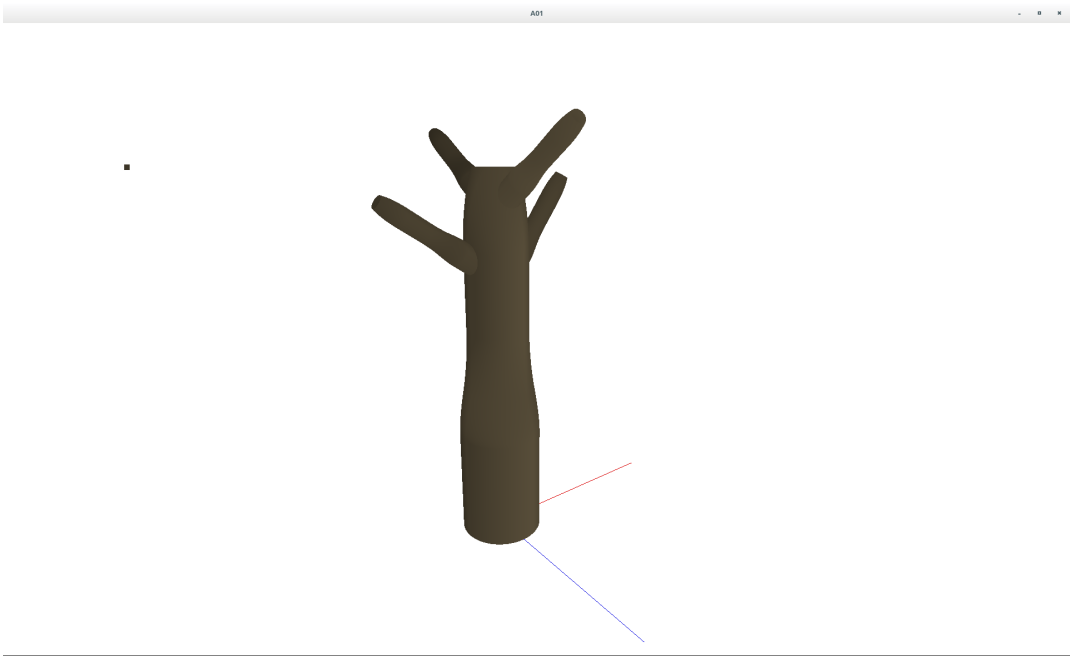
In terms of the actual landscape elements, we managed to model the following:

- *Land*: For land, the use of a bilinear blending surface seemed the most suitable since it can model a somewhat realistic looking landscape with elevation by using just four curves. In the input script, the user will specifying 4 sets of control points corresponding to 4 curves $P_0(u), P_1(u), Q_0(u), Q_1(u)$, and use the variable O3 to denote that a bilinear blending surface should be modeled. Furthermore, since there are 4 joints on the edge and it is

hard for user to make joints coincide, making joints between adjacent edges interpolated is an user friendly feature.
E.g.

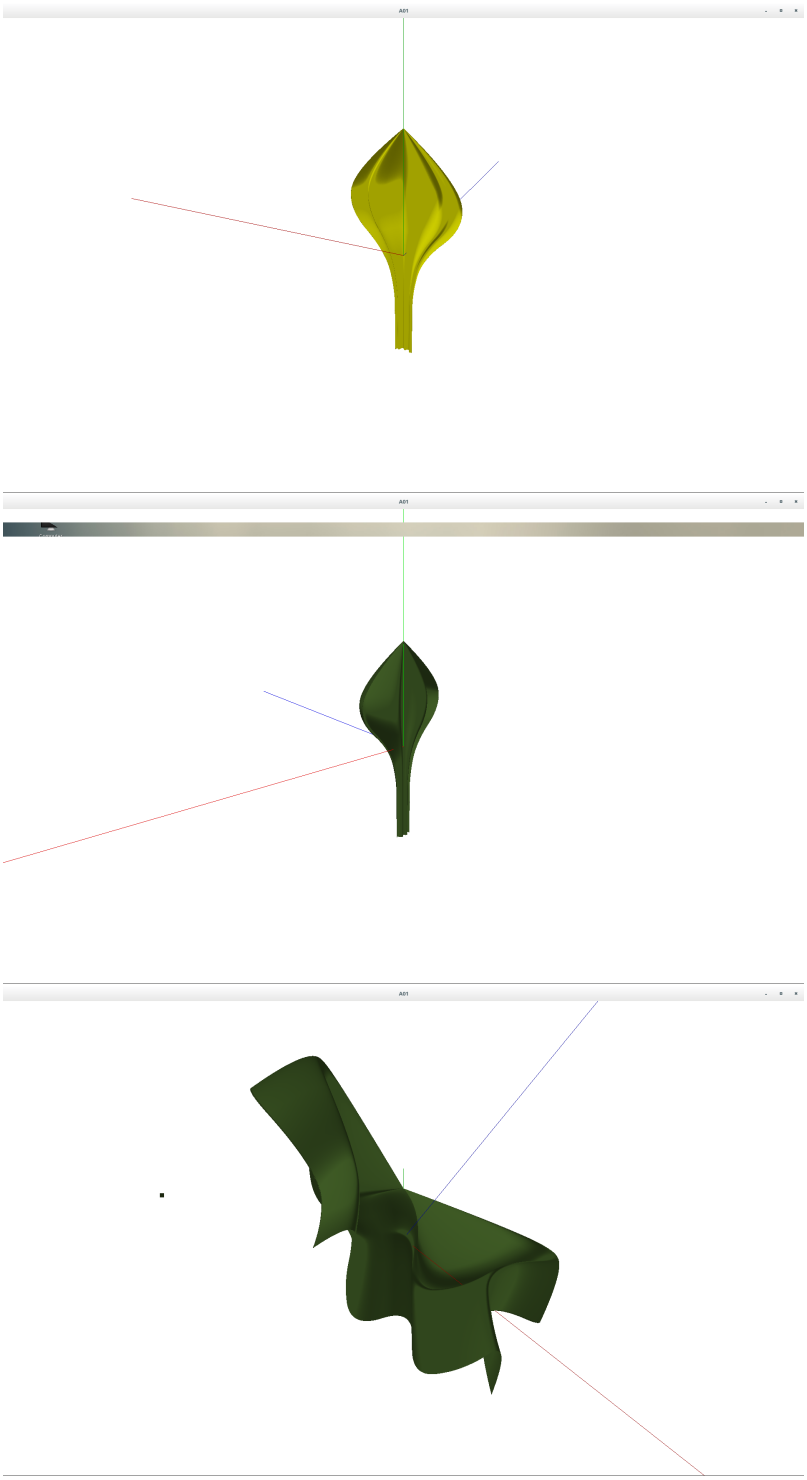


- *Trees*:
  - Tree trunks and branches
    Tree trunks are modeled using a rotational blending surface, in the input script, the user specifies a two sets of control points corresponding to silhouette curves, and use the variable O4 to denote that a rotational blending surface should be modeled. Branches are modeled by using scaled versions of tree trunks with rotation and translations applied. Since control points got from user is 3D but $q_r(u), q_l(u)$ should be co-planner 2D curves, we discard z-axis coordinate data of control points to obtain 2D curves.
    E.g.

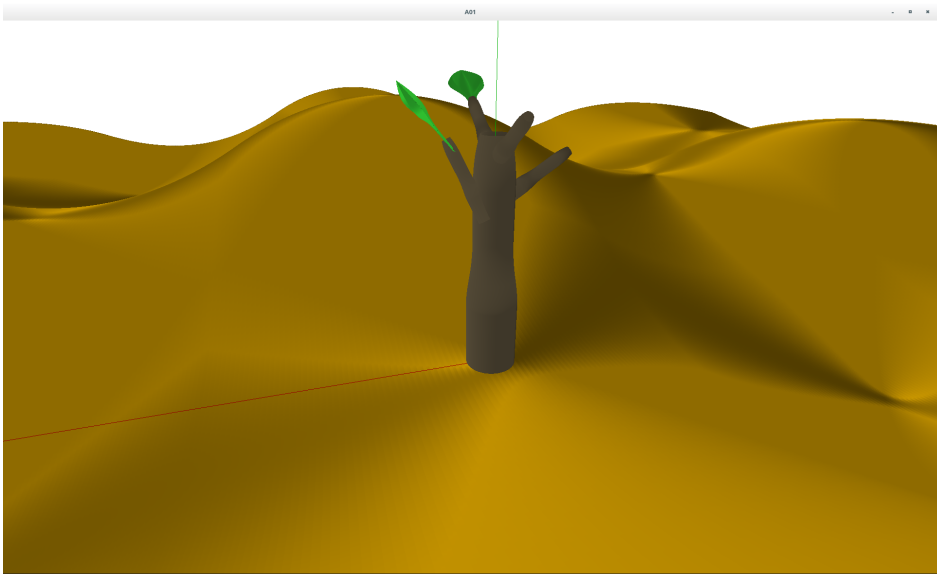


– Leaves

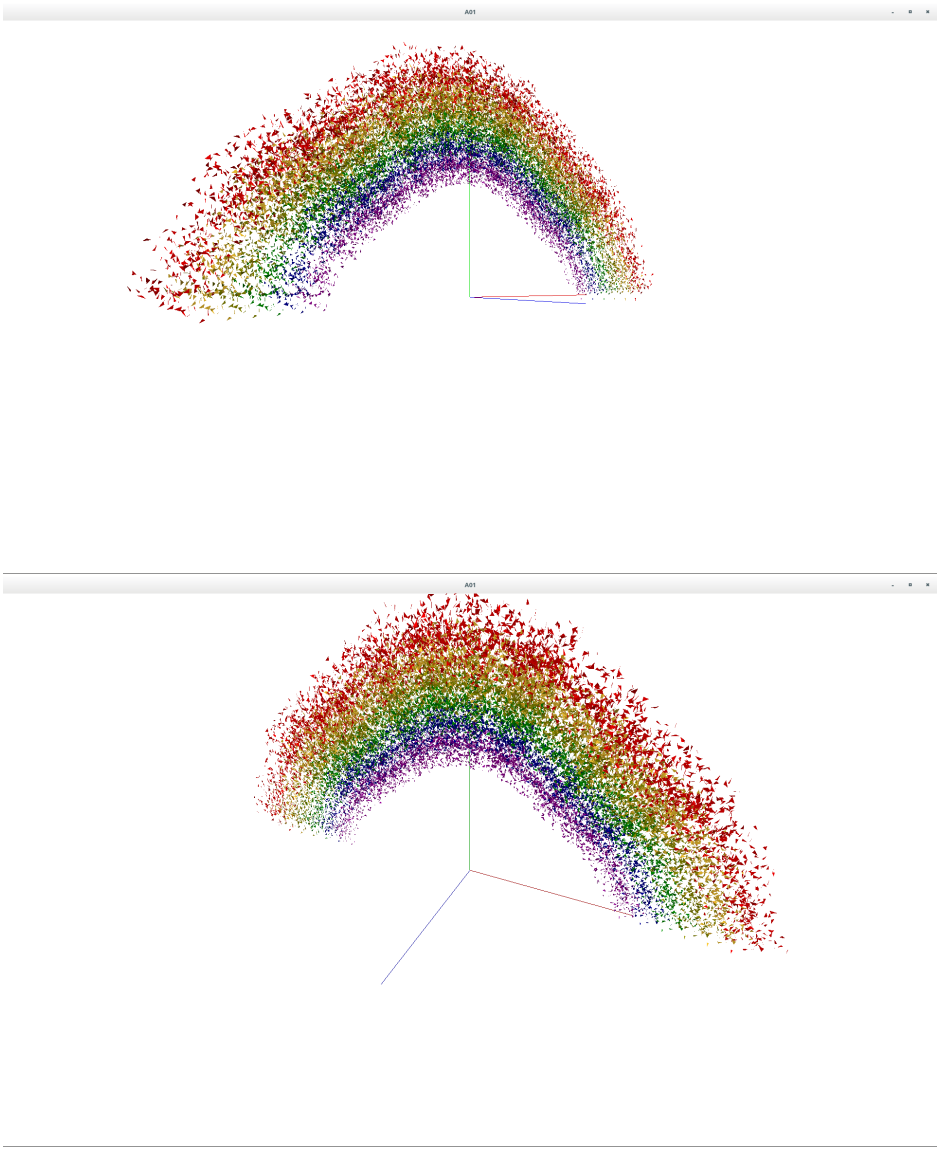
Leaves are modeled using a cross-sectional surface. In the input script, the user specifies three sets of control points with two sets corresponding the the left and right silhouette curves, and the third corresponding to the cross-section, and use the variable O5 to denote that a cross-sectional surface should be modeled.



To make a full tree, we applied scaling and transformations to leaves and branch objects, as shown below.



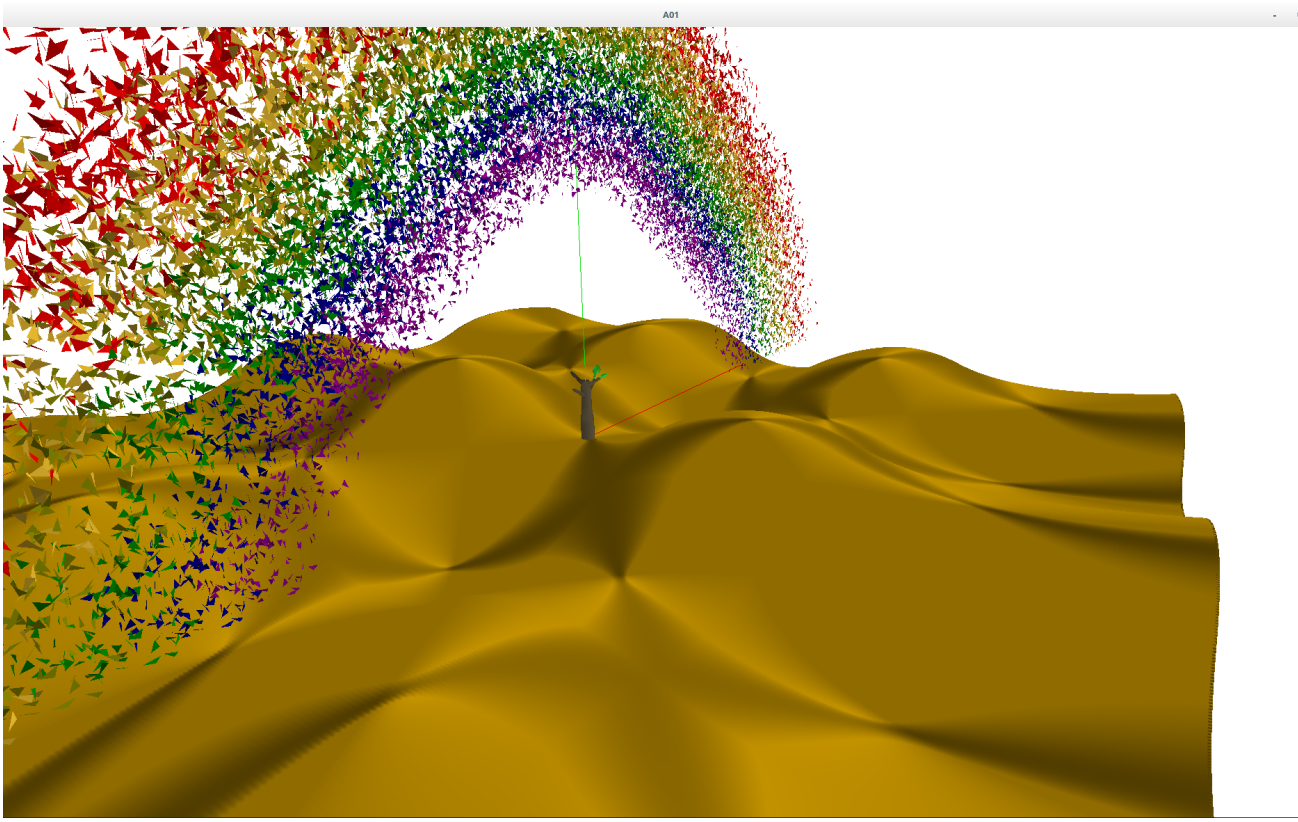
- Clouds and Rainbows
  Clouds are modeled using a random small triangles along curve model called spray model. In the input script, the user specifies one set of control points, and use the variable O6 to denote that a spray should be modeled along specified curve.
  Rainbows are modeled as special clouds. Input script may include arcs that adjacent to each other with different colors.
  E.g.



Section 4.3. Rendering

We are using the Phong shading model to render our objects. After generating the mesh for our surfaces, calculation of normals for each OpenGL primitive is hard. However, we were able to finally find pattern for the mesh sequence. By properly reversing the direction of normals calculated by cross production of three consecutive verities, normals work well for diffuse lights and specular lights.

5. DEMO SCENES



6. CONCLUSION

Overall, even though we were not able to carry out some of the project requirements such as interactive user input using iPhones, we achieved modeling of landscape scenes through a script based user input, and were successful in implementing multiple curve-based surfaces and using them each to their advantage to model different types of elements.

References

[1] Faramarz Samavati*CPSC 589 Lecture Notes*.