Assignment 3 - Geometry

This assignment makes you practice geometric algorithms in computer graphics. For your in-depth understanding of this area, it also covers a few related concepts that might be new to you. You should use any resources (e.g., books, search engines, calculators, and etc.) that can help you accomplish it.

The assignment provides a few code skeletons for corresponding tasks, please check https://github.com/mimuc/cg1-ss20.

Task 1: Geometric Representations

There is no “best” representation. Some geometric representation works better depending on the task. For instance, it is not a good decision to use polygon mesh to represent fluid. This task helps you walk through different geometric representations.

a) Given a sphere (A) and a cuboid (B), the Constructive Solid Geometry (CSG) operations result in geometries using AND (\(\cap\)), OR (\(\cup\)), NOT (\(\neg\)), and XOR (\(\oplus\)) that shown in Figure 1. Write the corresponding CSG formula for each subfigure.

Figure 1: CSG operations were performed in each of the four figures.

b) Figure 2 shows a tetrahedron with given 4 vertices. Fill the vertex list and face list table (Coordinates can be omitted) using Face-Vertex mesh representation to express the given tetrahedron.

Figure 2: Using Face-Vertex table to represent a tetrahedron.
Assignment 3: Geometry

The hilly terrain is usually represented as an unflattened surface, as illustrated in Figure 3, and there is a demo\(^1\) for you. It is common to use noise (e.g., Perlin noise\(^2\)) for generating random content, the terrain is a classic example.

![Figure 3: An example of a hilly terrain.](image)

**c)** To implement the terrain, which representation is more suitable than the others?

**d)** Which geometry do you need for the implementation of the terrain? And how do you modify this object for creating the terrain?

**e)** In the code skeleton, you can find four files: `main.js`, `renderer.js`, `perlin.js`, and `terrain.js`. Use the given Perlin noise generator to implement the // TODO: in `terrain.js`.

**f)** Why are triangles mostly used as basic elements comparing to complex polygons in 3D rendering? Name three reasons.

**g)** Why quad-meshes are commonly used in 3D modeling? Name three reasons.

Answer text questions in the `README.md` of the code skeleton, then include your answers and implementation in a folder called “task01”. Exclude the installed dependencies (folder `node_modules`) in your submission.

**Task 2: Bézier Curves Interpolation**

In the lecture, you learned Bézier Curves and the basic idea of the de Casteljau algorithm regarding how to draw a Bézier curve recursively. Let’s now heading from theory to practice, and implement it eventually.

Given control points \(P_0(1, 4), P_1(3, 1), P_2(6, 2), P_3(8, 6)\) of a cubic Bézier curve, shown in Figure 4.

\(^1\)http://www.medien.ifi.lmu.de/lehre/ss20/cg1/demo/3-geometry/terrain/
\(^2\)http://en.wikipedia.org/wiki/Perlin_noise
a) Draw the point on the Bézier curve determined by the given control points where parameter $t = 0.5$.

To implement the de Casteljau algorithm, we need to calculate the formula for each point on a Bézier curve.

b) Given two points $b_0(x_0, y_0), b_1(x_1, y_1)$. Show $b_1(t)(x, y)$ such that

$$\frac{|b_0 b_1|}{|b_1 b_1|} = \frac{t}{1-t}, t \in [0, 1]$$

where $|\cdot|$ represents the distance between two points.

c) In the provided code skeleton, you can find three JS files main.js, renderer.js, and bazier.js. Combining with the results you have in b), look for // TODO: in bazier.js, and implement the de Casteljau algorithm. If your implementation is correct, your scene should similar to Figure 5. There is an interactive demo \(^3\) to help you understand the scene, you can move the camera using the right button of your mouse.

d) Name three properties of Bézier curves.

---

\(^3\)http://www.medien.ifi.lmu.de/lehre/ss20/cg1/demo/3-geometry/bezier/
e) What constraints two connected piecewise Bézier curves to be “seamless”?

f) Name the reason why do we need piecewise Bézier curves other than a single curve with many control points?

g) Why do we have 16 control points for a bicubic Bézier patch?

Answer text questions in the README.md of the given code skeleton, then include your answers and implementation in a folder called “task02”. **Exclude** the installed dependencies (folder node_modules) in your submission.

**Task 3: Mesh Sampling**

In the lecture, you learned the basic idea of the algorithm vertex clustering to perform mesh simplification for downsample. To practice the algorithm, a mesh is given in the Figure 6a.

a) Apply the vertex clustering algorithm, draw a simplified mesh in Figure 6b.

b) What is the reduction ratio in terms of a number of triangles?

c) What drawback of the algorithm is showed in your result?

The goal of mesh simplification is to reduce the number of polygons while preserving its shape. In contrast, one can increase the number of polygons using subdivision for upsample. Mesh subdivision subdivides each polygon into smaller faces that better approximate the smooth surface.

An interesting question is: What could happen if we repeat these two operations as shown in the Figure 7?
There is an interactive demo 4 for you to explore. Implement this process in the provided code skeleton using existing three.js implementation: SubdivisionModifier5 and SimplifyModifier6.

d) What algorithms are used in three.js corresponding to mesh simplification and subdivision?

e) Assume the subdivision number is 2, and the reduction ratio of number of vertices is 95%. In the code skeleton, you can find main.js, renderer.js, and bunny.js. Implement the // TODO: in bunny.js. Then present a picture for $M_{10}$.

f) What did you observed from your result and what could you conclude from it?

Answer text questions in the README.md of the provided code skeleton, then include your answers and implementation in a folder called “task03”. Exclude the installed dependencies (folder node_modules) in your submission.

4http://www.medien.ifi.lmu.de/lehre/ss20/cg1/demo/3-geometry/bunny/
5https://threejs.org/examples/?q=modifier#webgl_modifier_subdivision
6https://threejs.org/examples/?q=modifier#webgl_modifier_simplifier
Submission

- Participation in the exercises and submission of the weekly exercise sheets is voluntary and not a prerequisite for participation in the exam. However, participation in an exercise is a good preparation for the exam (the content is the content of the lecture and the exercise).

- For non-coding tasks, write your answers in a Markdown file. Markdown is a simple mark-up language that can be learned within a minute. A recommended the Markdown GUI parser is typora (https://typora.io/), it supports parsing embedded formula in a Markdown file. You can find the syntax reference in its Help menu.

- Please turn your solution as a ZIP file via Uni2Work (https://uni2work.ifi.lmu.de/) before the deadline. We do not accept group submissions.

- Your solution will be corrected before the discussion. Comment your code properly, organize the code well, and make sure your submission clearer can help us to give you maximum feedback.

- If we discover cheating behavior or any kind of fraud in solving the assignments, you will be withdrawn for the entire course! If that happens, you can only rejoin the course next year

- If you have any questions, please discuss it with your fellow students first. If the problem cannot be resolved, please contact your tutorial tutor or discuss it in our Slack channel.