Computer Graphics Project2 : Spline & Subdivision Simulator

2023-15725 곽민서

1. Introduction

이번 프로젝트는 주어진 obj 파일을 읽어서 Spline과 Subdivision을 구현하는 것이다. Spline과 Subdivision은 모두 매끄러운 곡면을 구현하기 위한 수학적 도구로, 주어진 점을 통해 더 많은 점을 만들어 나가는 것을 기본원리로 한다.

구현한 Spline은 Bezier Spline과 Bspline으로, 내부로 점을 늘려나가 정교한 곡면을 만들며, 구현한 Subdivision은 Catmull Subdivision으로 외부로 점을 늘려나가 일부만으로 정교한 도형을 만들게 해준다.

여기에 단순한 구현을 넘어서 좀 더 다양한 그래픽의 구현을 위하여 마우스 이동, 키보드 이동, obj 저장과 gui 구현 등 추가적인 구현을 이루었다.

2. Implementation

수정한 문서는 main.py, shader.py, control.py 총 3개라고 볼 수 있다.

이 장에선 수학적 구현만을 보기로 한다.

1) main.py

main.py는 초기값을 설정하는 역할을 수행한다. .obj파일을 읽어와 Vertices와 Indices의 관한 정보를 파싱한다. 이 정보를 바탕으로 Spline이면 16개의 Control Point의 프레임을, Subdivision이면 Cage 프레임을 생성한다. 이 점들을 바탕으로 추가적인 Vertices와 Indices를 생산하는데, Change of Basis 행렬 연산을 이용해 Vertices를 구하고, 이로 인해 생기는 인접한 4점씩 모은 Quad의 Indices를 얻는데, 이 Indices는 3개씩 모아서 삼각형 폴리곤으로 면을 구현하고, 2개씩 모아서 프레임을 만든다. Subdivision에 대해서도 정해진 Catmull Subdivision 알고리즘에 따라 새로운 Vertices와 Indices를 생성하여 프레임과 면을 구현한다. 이때, CPU연산의 한계로 Subdivision의 반복횟수를 의미하는 STEP은 2~3이 한계이다.

2) shader.py

Shader.py는 초기 구현 이후의 동작을 담당한다. 주로 Control point의 움직임이 가장 많이 다루어지는데, 각 Control Point에 있는 16개의 작은 정사각형이 Control Point를 시각화한다. 마우스가 Control Point에 다가가면 정사각형이 커져서 조작을 할 수 있도록 해주는데, 이는 각 좌표의 WC를 DC까지 바꿔서 이를 마우스 위치와 대조하여 구한다. 마우스를 누를 때 Control Point 근처에 있다면, 그 Control Point를 움직이기로 결정한것이다. 그리고, 초기 마우스 위치를 저장해놓는다. 마우스를 움직이면, 마우스의 초기위치에 대한 움직임의 스크린상의 벡터(DC상의 dz은0 으로 한다)를 WC의 벡터로 바꿔, 이를 Control Point에 Translation을 취한다. 이때, 바로 달라진 control point에 따라 Spline 상의 모든 점을 다 업데이트 해준다. 이로써 마우스의 움직임에 따라 Control Point가 움직이며, 드래그를 구현하였다. 마우스를 다시 때면 다시 사용가능 상태로 바꾼다. 이는 Spliine의 Control Point만 가능하게 설정한다.

3) control.py

키보드와 마우스의 인터럽트 장치의 입력을 구현하게 해준다. 또한, esc로 정상적인 종료가 일어날때, 마지막에 있던 Spline 데이터를 .obj의 형태로 저장하여 사용가능하게 한다.

3. Additional Implementation

1) Tkinter GUI

더 편안한 초기설정을 위하여 Tkinter GUI를 이용하였다. Tkinter은 pip install tk로 설치가 가능하며, 간단한 gui를 구현하게 해준다. 우선 어떤 파일을 어떤 모드로 실행할 것인지 결정하기 위하여 라디오버튼 모듈을 적용하여 모드를 선택할 수 있다. 또한 RGB 값을 스케일 모듈을 사용하여 지정할 수 있는데, 이는 곡면의 색을 바꿀 수 있게 해준다. 버튼을 누르면 주어진 설정에 맞게 pyglet을 실행시키고, X버튼으로 비정상종료를 하면 pyglet실행없이 종료한다.

2) 키보드 입력

Pyglet상에서 3D Graphic Simulator으로써의 기능을 강화하기 위하여 키보드 입력을 만들었다. WASD를 통한 전후좌우 움직임, QERF을 통한 회전, TG를 통한 확대, 축소가 구현되어 있다. 모두 View_matrix와 View_target를 변경하여 구현하였다. 모두 사용자가 바라보는 시점을 기준으로 하여 움직이며, W는 전진, S는 후진, A는 좌로 움직임, D는 우로 움직임이며, View_matrix와

View_target모두를 똑같이 Transition시킨다. Q는 좌로 회전, E는 우로 회전, R은 위로 회전, F는 아래로 회전으로, 현재 위치는 변경 없이 방향만 변경하여 구현한다. T는 확대, G는 축소로 view_target방향으로의 size 변경으로 구현한다.

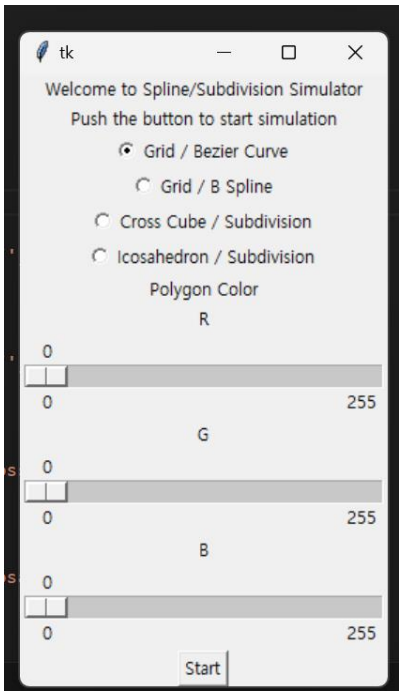
3. Unimplemented

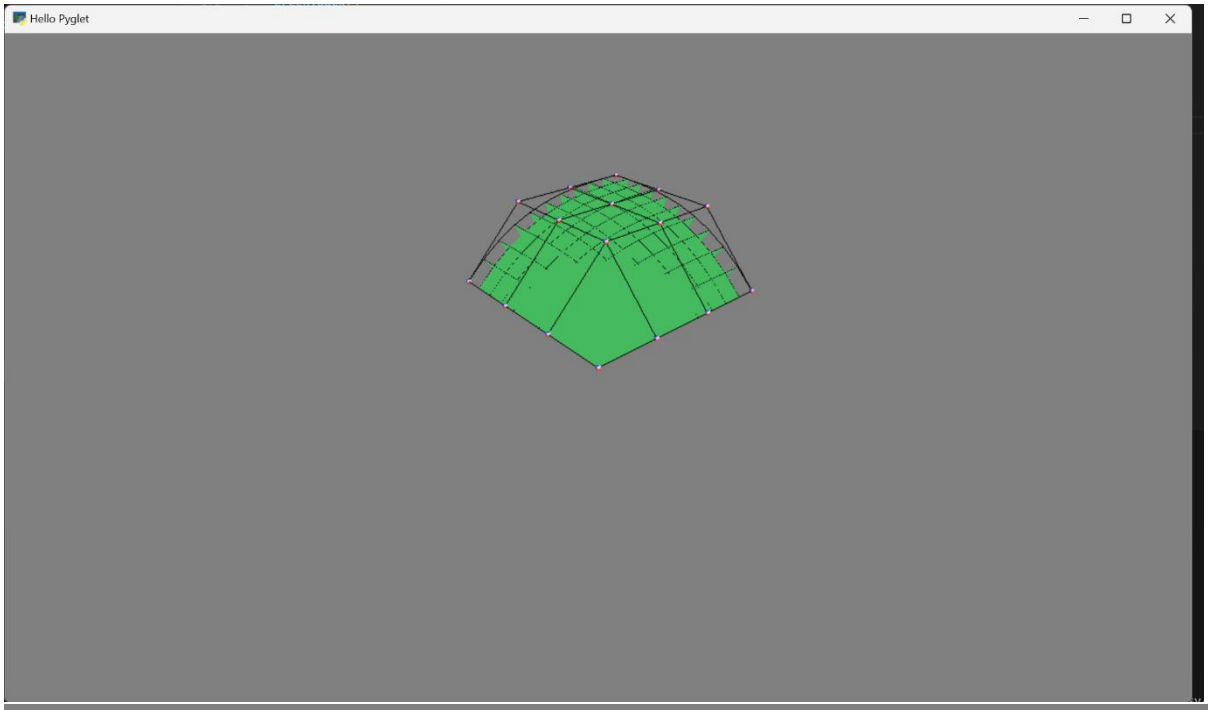
Antialiasing과 Rasterization에 관해선 대처가 미흡하다. Open Surface는 SAMPLE이 적으면, 모서리에서 보는 각도에 따라 끝이 갈라져 보이며, 이는 추가적인 구현을 요구한다. 또한, 곡면과 프레임이 자주 겹쳐 프레임이 보이기도, 안보이기도 한다.

또한, 드래그 움직임이 매끄럽지 못하다. 한번의 드래그에 모든 Control Point의 Spline을 새로 계산해야하는 만큼 드래그에 제약이 걸리기도 한다.

이외에도 여러가지 버그가 생길 수 있다.

5. Screeenshots




Hello Pyglet