CS 450/550 – Project 6 Report

Pig-in-the-Python (GLSL Shaders)

Student: Haochuan Zhang
Email: `zhanhaoc@oregonstate.edu`

Date: November 26, 2025

Project Overview

This project replaces the texture-mapped solar-system scene from Project 5 with a single animated snake model and implements a fully programmable shading pipeline. Using GLSL vertex and fragment shaders, I created the "pig-in-the-python" effect—a smooth, localized bulge that travels along the snake's X-axis over time. Per-fragment Phong lighting was implemented entirely in the fragment shader.

The bulge's center, width, and height are controlled through uniform variables passed from C++, and its motion is driven by a cyclic time variable derived from GLUT's elapsed time. The model geometry (from `snakeH.obj`) is deformed in the vertex shader using a smoothstep-based pulse function. All lighting (ambient, diffuse, specular) is computed per pixel in the fragment shader.

What I Built

- **Scene Simplification:**
  Project 5's ten-object solar system was removed. The scene now contains a single `ObjectInfo` bound to `snakeH.obj`, rendered without textures and lit solely through the custom GLSL pipeline.
- **GLSL Integration:**
  Activated the OSU `glslprogram.cpp` loader and added two new shader files:
  - `PigInPython.vert` – vertex deformation + lighting vectors
  - `PigInPython.frag` – per-fragment Phong shading

  The shader program is created during initialization and bound before drawing.
- **Pig-in-the-Python Deformation:**
  The vertex shader computes a bulge pulse using two smoothstep functions:

  ```
  pulse = smoothstep(PigD - PigW/2, PigD, x)
        - smoothstep(PigD, PigD + PigW/2, x);
  ```

  The snake's YZ coordinates are scaled by:

  ```
  MCvertex.yz *= (1.0 + pulse * uPigH);
  ```
- **Time-Driven Animation:**
  A cyclic time variable (`PigTime01` in [0,1)) drives:
  - Bulge position `uPigD` traveling from tail (-13) to head (+9)

  - Bulge height `uPigH` using a sine modulator (optional)

  The direction toggles with key `p`, and height animation toggles with `h`.

- **Per-Fragment Phong Lighting:**
  The fragment shader computes ambient, diffuse, and specular terms using interpolated vectors:

  ```
  varying vec3 vN, vL, vE;
  ```

  Constants (Ka, Kd, Ks, Shininess) are defined in-shader. The object is rendered using a bright orange material to visualize shading clearly.

- **Runtime Controls (P6-specific additions):**
  - `p`: Toggle bulge direction (tail→head / head→tail)
  - `h`: Toggle height animation (sine)
  - `1`: Camera follow-mode on the snake
  - `0`: Return to orbiting free-camera
  - `b`: Toggle light-beam visualizer (for debugging)

- **HUD Overlay:**
  Updated to show Pig direction, height animation state, GLSL program status, and the current view zoom. Colored labels match the P5 aesthetic.

Implementation Details

Vertex Shader (PigInPython.vert)

- Inputs: `gl_Vertex`, `gl_Normal`, `LightPosition`
- Uniforms: `uPigD`, `uPigH`, `uPigW`
- Outputs: `vN`, `vL`, `vE`
- Computes deformation using smoothstep, applies to YZ, and outputs the transformed position through the model-view-projection matrix.

Fragment Shader (PigInPython.frag)

- Performs per-pixel Phong lighting using normalized vN, vL, vE.
- Uses material constants:

  ```
  Ka, Kd, Ks, Color, SpecularColor, Shininess
  ```

- Produces smooth specular highlights along the bulge crest.

CPU Side: Time and Uniforms

Uniforms are set every frame:

```
PigProgram.SetUniformVariable("uPigD", PigD);
PigProgram.SetUniformVariable("uPigH", PigH);
PigProgram.SetUniformVariable("uPigW", PigWidth);
```

Bulge position:

```
PigD = -13 + PigTime01 * (9 + 13);    // tail → head
```

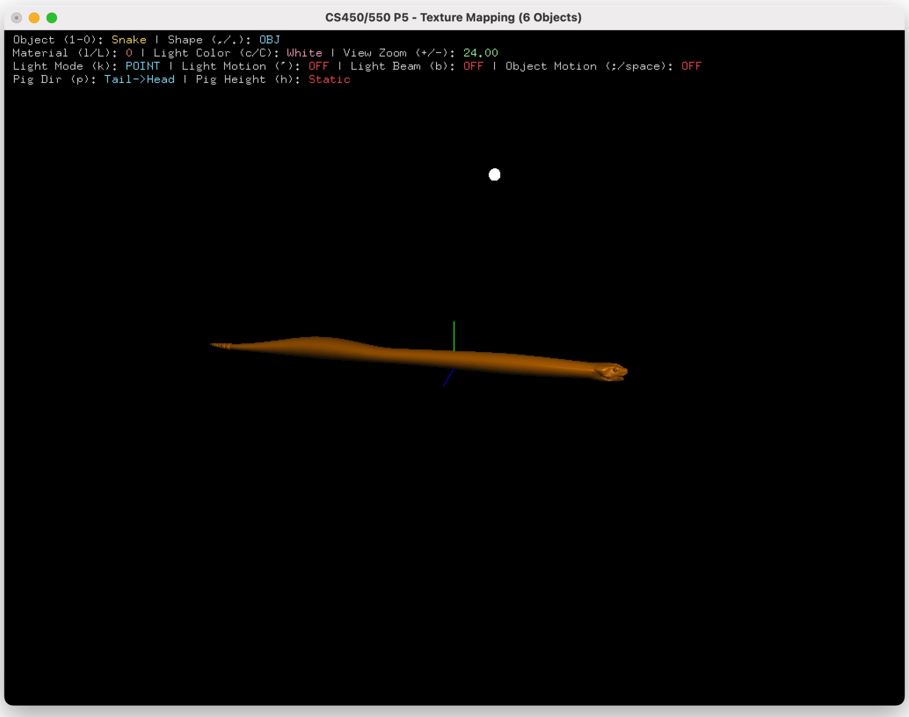
Height animation (optional):

```
PigH = 1.0 + 0.8 * sin(2*pi*PigTime01);
```

A 5-second cycle is used for clear visualization.

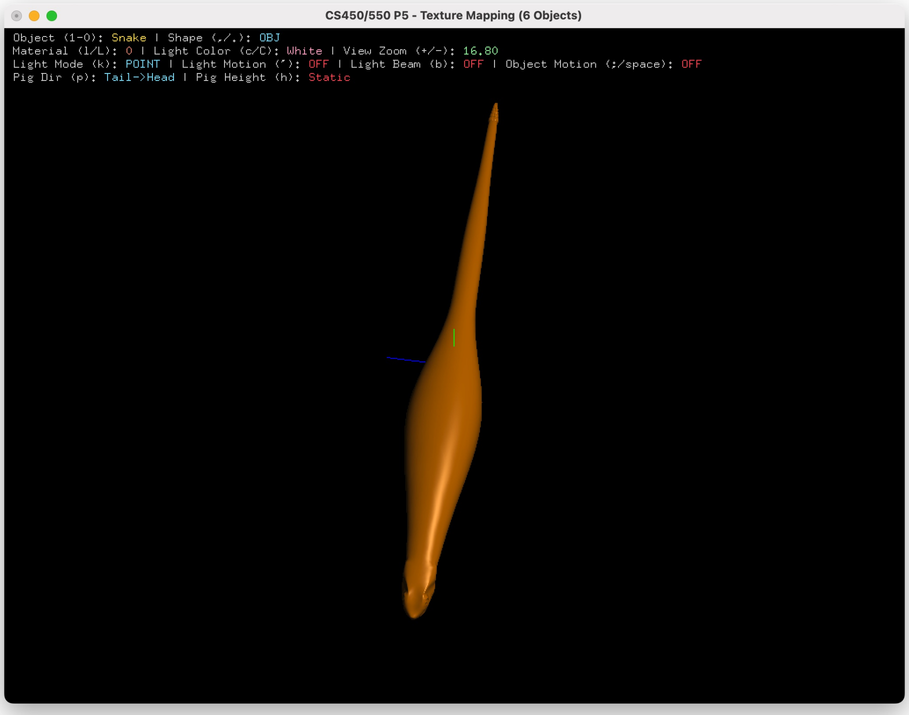
Results and Demonstration

Screenshots



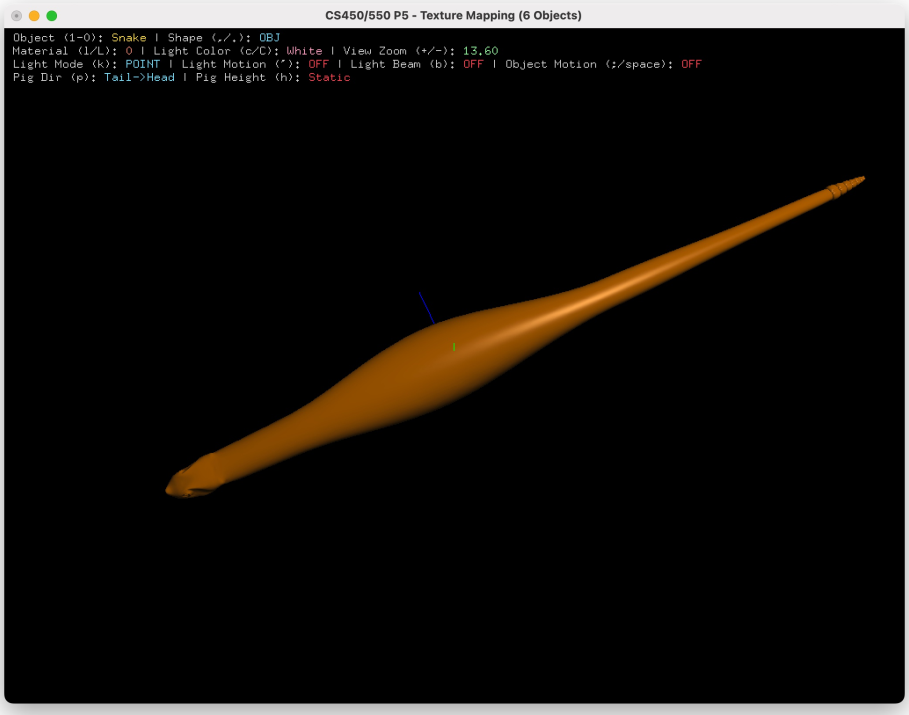
```
Object (1-0): Snake | Shape (,/.): OBJ
Material (l/L): 0 | Light Color (c/C): White | View Zoom (+/-): 24.00
Light Mode (k): POINT | Light Motion ('): OFF | Light Beam (b): OFF | Object Motion (;/space): OFF
Pig Dir (p): Tail->Head | Pig Height (h): Static
```

Screenshot 1: Baseline snake with no bulge (far zoom).



```
Object (1-0): Snake | Shape (,/.): OBJ
Material (l/L): 0 | Light Color (c/C): White | View Zoom (+/-): 16.80
Light Mode (k): POINT | Light Motion ('): OFF | Light Beam (b): OFF | Object Motion (;/space): OFF
Pig Dir (p): Tail->Head | Pig Height (h): Static
```

Screenshot 2: Mid-travel bulge at higher zoom.



```
Object (1-0): Snake | Shape (,/.): OBJ
Material (l/L): 0 | Light Color (c/C): White | View Zoom (+/-): 13.60
Light Mode (k): POINT | Light Motion ('): OFF | Light Beam (b): OFF | Object Motion (;/space): OFF
Pig Dir (p): Tail->Head | Pig Height (h): Static
```

Screenshot 3: Bulge near the tail, viewed from camera-follow mode.

Demo Video

A narrated demonstration is available at:
`https://media.oregonstate.edu/media/t/1_gp7yfd23` (unlisted Kaltura link)

The video shows bulge motion, height animation toggling, GLSL shading behavior, and interaction with different viewing angles.

Key Features Summary

- Smoothstep-driven "pig-in-the-python" deformation
- Time-dependent bulge travel with optional height animation
- Full conversion to GLSL programmable shading
- Per-fragment Phong lighting with clear specular response
- Minimal scene design focused on deformation + shading clarity
- HUD feedback and interactive bulge direction switching

Reflection

This assignment required transitioning from OpenGL's fixed pipeline into a fully programmable shader workflow. Implementing deformation in the vertex shader clarified how model-space geometry flows through the pipeline, while per-fragment Phong shading demonstrated the benefits of fine-grained lighting control. The smoothstep pulse made the deformation visually clean and stable, and CPU-driven animation tied the effect together.

Overall, Project 6 provided a deeper understanding of GLSL, uniform communication, and real-time geometric manipulation.