3D Model Loading Library: A Personal Reflection

Rowan Stratton

Southern New Hampshire University

CS-330-R1881 Comp Graphic and Visualization 23EW1

Professor Jeff Phillips

October 22, 2023

My Choices and Their Rationales:

On the Objects I Selected: I opted for 3D models in a JSON-like format, possibly referencing GLTF, after much consideration. My reasoning? GLTF is widely recognized and is known for its adaptability. This choice ensures that the library can cater to an extensive array of 3D assets, granting me and any potential users a lot more freedom in design.

On Providing the Right Functionality: I was determined to make sure my library could handle 3D models with multiple nodes, along with their transformations and mesh data. This might sound ambitious, but I was aiming for intricate 3D scenes, not just stand-alone models. The challenge was real, but it felt right for what I wanted to achieve.

Moving Around My 3D World:

How Users Can Navigate: Although the focus of my code leaned towards model loading, I did drop hints about potential interactions with a Camera class. If I had to elaborate based on standard 3D rendering practices:

- **Moving Around**: I'd probably allow users to use keys like W, A, S, D for basic movements. It feels intuitive.
- **Looking Around**: Mouse movements for this one. I've always found it natural to look around with a slight drag of the mouse.
- **Getting Closer or Further Away**: A mouse scroll could work wonders for zooming in and out. Maybe specific keys if I want to spice things up.

Camera and Its Controls: A Camera class would likely be at the heart of rendering the 3D space from the viewer's angle. I imagine this class handling the view and projection matrices to make sure every object pops out just right. Integrating various input devices would definitely be on the cards. Keyboards, mice, even touchscreens, should play well with the Camera class for smooth navigation.

Functions I'm Proud Of:

1. **loadMesh(unsigned int indMesh)**:
    - **What's It For?**: When I was writing this, I wanted a neat way to pluck out a specific mesh from a 3D model using its JSON index.
    - **Why It's Handy**: Anytime I want to fiddle with a particular mesh, this is the tool. It's like having a Swiss army knife for mesh manipulation.
2. **traverseNode(unsigned int nextNode, glm::mat4 matrix)**:

- **What's It For?**: This was all about diving deep into nodes in hierarchical 3D models. I wanted it to sift through nodes, apply those intricate transformations, and then get the associated meshes.
- **Why It's Handy**: This is my go-to for any model with a node-based structure. It's like having a universal key for any kind of 3D asset.

3. **getData()**:
    - **What's It For?**: Getting the binary data was crucial. This function is the backstage pass to all that parsing action.
    - **Why It's Handy**: It's foundational. Anytime I need to sneak a peek at the binary data, this is the magic spell.
4. **groupFloatsVecX(std::vector<float> floatVec)**:
    - **What's It For?**: I needed a way to turn a flat list of floats into organized vectors. These functions were my attempt at taming that chaos.
    - **Why It's Handy**: They're my go-to set whenever I'm dealing with any data extraction. It's like organizing a messy drawer. So satisfying.

Wrapping it up, this 3D Model Loading Library was a personal challenge. I wanted something adaptable, user-friendly, and something that felt like 'me'. Through trials, errors, and lots of coffee, I believe I've made something I can be proud of. It caters to an array of 3D models, remains intuitive, and offers a platform for both developers and users to explore and create.