# Geometry Viewer Implementation Plan

## Current Status Analysis

From the provided code, you have:

- Basic OpenGL initialization

- Simple camera setup

- Basic GameObject class

- Transform component with basic operations

- Project structure with three projects (Engine, Editor, Game)

## Required Features & Implementation Plan

### 1. Core Architecture (Priority: High)

- [x] Basic project structure

- [ ] Component system implementation

- Create base Component class

- Implement Transform, Mesh, and Material components

- Add component management to GameObject

- [ ] Scene management system

- Create Scene class to manage GameObjects

- Implement scene hierarchy

### 2. Model Loading (Priority: High)

- [ ] Assimp integration

- Add Assimp library to project

- Create ModelLoader class

- Implement FBX loading

- Extract mesh data (vertices, indices, normals, UVs)

- [ ] Mesh component

- Store mesh data

- OpenGL buffer management

- Normal visualization

- [ ] Initial baker\_house.fbx loading

### 3. Texture System (Priority: High)

- [ ] DevIL integration

- Add DevIL library

- Create TextureLoader class

- [ ] Material component

- Texture management

- Checker texture generation

- Support for DDS/PNG formats

- [ ] Drag & drop support for textures

### 4. Camera System (Priority: Medium)

- [ ] Extend current Camera class

- Implement orbit controls

- Add focus functionality

- Speed modifier with SHIFT

- [ ] Input system

- WASD movement

- Right-click free look

- Alt+Left click orbit

- Mouse wheel zoom

- 'F' key focus

### 5. Editor UI (Priority: Medium)

- [ ] Main menu

- File operations

- View toggles

- About window

- GitHub link

- [ ] Console window

- Log system

- Assimp/DevIL output capture

- [ ] Configuration window

- FPS graph

- Module settings

- Hardware info

- [ ] Hierarchy window

- GameObject list

- Selection system

- [ ] Inspector window

- Component display

- Transform values

- Mesh information

- Texture properties

- Normal visualization toggle

### 6. Basic Shapes (Priority: Low)

- [ ] Primitive generation

- Cube

- Sphere

- Cylinder

- Custom shapes menu

## Implementation Order

1. Core Architecture

```

Week 1:

- Component system

- Scene management

- Basic GameObject hierarchy

```

2. Model Loading

```

Week 1-2:

- Assimp integration

- Mesh component

- Basic model loading

```

3. Texture System

```

Week 2:

- DevIL setup

- Basic texture loading

- Material component

```

4. Camera Controls

```

Week 2-3:

- Camera movement

- Orbit controls

- Input system

```

5. Editor UI

```

Week 3-4:

- Window framework

- Basic windows

- Inspector/Hierarchy

```

6. Polish & Testing

```

Week 4:

- Bug fixing

- Performance optimization

- Documentation

```

## Project Structure

```

SpaghettiEngine/

├── Core/

│ ├── Component.h/.cpp

│ ├── GameObject.h/.cpp

│ └── Transform.h/.cpp

├── Graphics/

│ ├── Mesh.h/.cpp

│ ├── Material.h/.cpp

│ └── Camera.h/.cpp

├── Resources/

│ ├── ModelLoader.h/.cpp

│ └── TextureLoader.h/.cpp

└── Editor/

├── Windows/

│ ├── ConsoleWindow.h/.cpp

│ ├── HierarchyWindow.h/.cpp

│ └── InspectorWindow.h/.cpp

└── EditorLayer.h/.cpp

**Key Technical Considerations**

1. Memory Management:

* Use smart pointers for resource management
* Implement proper cleanup in destructors
* Track and free OpenGL resources

1. Performance:

* Minimize state changes in rendering
* Batch similar operations
* Use efficient data structures

1. Architecture:

* Keep modules loosely coupled
* Use events for communication
* Follow SOLID principles

1. Error Handling:

* Robust error checking for file operations
* Graceful handling of invalid inputs
* Clear error messages in logs