**User Story: #101 - Research and Implement 3D Model Optimization Techniques**

**1. Polygon Reduction Techniques - [#102]**

**Objective**: Research ways to reduce the number of polygons in 3D models to improve performance without losing significant visual quality.

**Three.js Approach**:

* Blender Decimation Modifier: Blender’s Decimate Modifier is a great way for us to simplify models while keeping their overall shape. Once we’ve reduced the polygon count in Blender, we can easily export the model in a format like glTF or OBJ, which integrates with Three.js.
* LOD in Three.js: While Three.js doesn’t offer automatic polygon reduction, it does support Level of Detail (LOD), which allows us to create different versions of a model (high-poly, mid-poly, low-poly). Three.js will automatically switch between these versions based on how far the camera is from the object. This helps us reduce the complexity of distant objects, keeping everything running smoothly.

**Personal Recommendation**: I recommend that we use Blender’s Decimate Modifier first to streamline our models, then take advantage of Three.js’s LOD system to handle performance optimization. It’s a bit more hands-on than other engines, but it works well and keeps our project lightweight.

**2. Texture Compression Techniques - [#103]**

Objective: Research methods to reduce the file size of textures to improve memory usage and load times while maintaining visual quality.

**Three.js Approach:**

* Texture Compression in Three.js: Three.js doesn’t handle texture compression automatically, but there are some excellent tools we can use to manage this. I’d suggest we work with KTX2 or Basis Universal. These formats allow us to compress textures so they load faster and use less memory, all without losing too much visual quality. We’d compress the textures before loading them into Three.js, ensuring they’re optimized across different platforms.
* Mipmaps: Three.js handles mipmaps automatically, which is a great help. Mipmaps improve performance by reducing the resolution of textures when objects are far away, lowering memory usage and improving load times.

**Personal Recommendation**: We should pre-compress our textures using KTX2 or Basis Universal before loading them into Three.js. After that, we can rely on Three.js to handle mipmapping and further optimize performance without additional effort on our part.

**3. Level of Detail (LOD) Techniques - [#104]**

**Objective**: Explore techniques to adjust the level of detail of 3D models dynamically to improve performance without sacrificing too much visual fidelity.

**Three.js Approach:**

* LOD in Three.js: Three.js offers built-in support for LOD, which allows us to create different versions of the same model with varying levels of detail. For example, we can use low-poly models when they’re far away and switch to high-poly models as they get closer. Three.js takes care of switching between these versions based on the camera’s distance, keeping the performance optimized without requiring manual intervention from us.

**Personal Recommendation:** I recommend that we utilize Three.js’s LOD system. It’s straightforward to implement, and once we create the different versions of our models (which we can easily do in Blender), Three.js will handle the rest for us.

**Best Workflow Summary for User Story #101 with Three.js:**

1. **Polygon Reduction: We simplify our models in Blender using the Decimate Modifier**, then import them into Three.js. For larger scenes, we’ll use the LOD system to manage different versions of the same model.
2. **Texture Compression: We compress textures using KTX2 or Basis Universal** before loading them into Three.js, and then let Three.js handle mipmapping to ensure everything loads quickly and efficiently.
3. **LOD Implementation: We’ll take advantage of Three.js’s LOD functionality** to automatically switch between different levels of detail based on the camera’s distance.

**User Story: #105 - Research and Propose Mechanisms for User Feedback**

**1. Research Various User Feedback Collection Methods (#106)**

**Objective**: Explore and document different methods for collecting feedback from users to understand their experiences and needs.

**Approach**:

* Since our project is web-based, we can easily integrate feedback collection directly into the webpage. We can embed tools like Google Forms or use a quick feedback pop-up that appears after users interact with the 3D models. This makes it convenient for users to share their experiences right as they engage with our product.
* For more detailed insights, we can use tools like Google Analytics or Hotjar to track how users are interacting with our 3D content. These tools can show us where users click, how long they spend on certain models, and if they’re having any trouble navigating our project, all without needing direct input from the user.

**Personal Recommendation**: I’d suggest that we start by embedding a simple feedback form on the webpage so users can easily share their thoughts. At the same time, we should use event-tracking tools like Google Analytics to get more in-depth data on how users are interacting with our models.

**2. Evaluate Tools for Storing and Managing Feedback (#107)**

**Objective**: Identify and assess tools that can help store, manage, and organize feedback efficiently.

**Approach**:

* Google Forms or SurveyMonkey are perfect for collecting feedback. They’re simple to set up, and they allow us to collect both structured and open-ended responses from users.
* To organize and manage the feedback, I’d recommend we use Airtable or Trello. Airtable is great for managing large amounts of data, combining the power of a database with the ease of a spreadsheet. Trello, on the other hand, is excellent for tracking feedback and turning it into manageable tasks for our team.
* If we want to capture real-time feedback, we can integrate tools like Zendesk or Intercom directly into our webpage. This allows users to submit live feedback or support requests while they interact with our content.

**Personal Recommendation**: We should use Google Forms or SurveyMonkey to collect feedback and Airtable to organize it. If the project is smaller, Trello would be a simpler way to track feedback and convert it into actionable tasks for us.