

**LECTURE : Computer Graphics**

**LECTURER : Öner Barut**

21118080059 Ahmet Muhittin Gürkan

**3D Home Decoration Simulator - Project Report**

Executive Summary

This project presents a comprehensive web-based 3D home decoration simulator developed using the Babylon.js framework. The application enables users to create and customize virtual room environments by placing, manipulating, and managing furniture items in a realistic 3D space. The simulator provides an intuitive interface for interior design experimentation with real-time rendering capabilities.

Project Overview

Objectives

- Create an interactive 3D room environment.

- Implement a furniture placement and manipulation system.

- Provide realistic lighting and material rendering.

- Develop a user-friendly interface for design tasks.

- Demonstrate modern web-based 3D graphics capabilities.

Technical Architecture

Development Environment

- IDE: Visual Studio Code

- Programming Languages: HTML5, CSS3, JavaScript ES6

- 3D Graphics Engine: Babylon.js 5.x

- Model Format: GLB (glTF Binary)

- Browser Requirements: WebGL 2.0 support

System Components

The project is organized with a main `index.html` file serving as the application interface. Core application logic, exceeding 1,200 lines of JavaScript, resides in `js/main.js`. 3D furniture assets are stored in the `models/` directory, including `sofa.glb`, `table.glb`, `chair.glb`, `lamp.glb`, and `bed.glb`. Material textures such as `floor.jpg`, `wall.jpg`, various fabric and wood textures, `tv.jpg`, and `window\_view.jpg` are located in the `textures/` directory.

Core Features Implementation

1. 3D Room Environment

Room Structure

- A complete enclosed room featuring four walls, a floor, and a ceiling.

- Dimensions are 10x10x5 units (length x width x height), providing realistic proportions for furniture placement.

- Camera collision detection is implemented for room boundaries.

Materials and Textures

- The floor is textured with a wood-like appearance, utilizing 4x4 UV scaling for detail.

- Walls are textured with realistic surface materials.

- All surfaces interact correctly with the lighting system.

- The floor and walls are configured to receive shadows.

2. Interactive Elements

Static Window with View

- A two-panel window is implemented on one of the walls, providing a view to the outside.

- This window is a static element and does not open or close.

- It includes a frame, glass material for the panes, and a texture (`window\_view.jpg`) depicting an outdoor scene, enhancing the room's ambiance.

Entry Door

- A detailed entry door model is present, complete with a door panel, frame, and handle.

- It is positioned in the front wall, providing a realistic entrance point.

Television

- A wall-mounted television unit is included.

- It features a frame and a screen displaying a static image texture (`tv.jpg`).

Shelving

- Multiple wall-mounted shelves are procedurally generated.

- Shelves include decorative items like books and vases, each with distinct materials.

3. Furniture Management System

Available Furniture Types

- Sofa: Blue fabric seating furniture.

- Table: Wooden dining/coffee table.

- Chair: Individual seating with custom texturing.

- Lamp: Ceiling-mounted lighting fixture that adds a dynamic light source to the scene.

- Bed: Bedroom furniture, rendered with its original model materials.

Placement System

- A click-to-place mechanism allows users to position furniture on the floor surface.

- Automatic positioning and appropriate scaling are applied for each furniture type upon placement.

- The system provides real-time visual feedback during the placement process.

Manipulation Features

- Placed furniture can be selected by clicking on it, indicated by a blue transparent highlight box.

- Selected furniture can be rotated in 45-degree increments using the 'R' key.

- Selected furniture can be removed from the scene using the 'DELETE' key.

4. Lighting and Rendering System

Main Lighting

- A central point light source is implemented, with its intensity adjustable via a UI slider (0.0-1.0 range).

- The position of the main light can be configured (center, left, right) through a UI dropdown.

- The light has a warm color temperature (1.0, 0.95, 0.85 RGB values).

Ambient Lighting

- A hemisphere light provides general scene illumination, softening shadows.

- Its intensity is also adjustable via a UI slider.

Dynamic Lighting

- When a "Lamp" furniture item is placed, it introduces an additional spot light source into the scene, parented to the lamp model.

- This light is directed downwards and has a warm color.

Shadow System

- Real-time shadow generation is active, with a 1024x1024 shadow map resolution.

- Blur exponential shadow mapping is used for softer shadow edges.

- All furniture objects are configured to cast shadows, and the ground/walls receive them.

5. Vegetation and Decoration

Plant System

- Three distinct types of procedurally generated plants are available for decoration:

- Ficus: Features a branching trunk structure and multiple spherical leaf clusters.

- Palm: Characterized by a central trunk and radiating fronds composed of smaller leaflet segments.

- Monstera: Known for its large, distinctively shaped leaves.

- Each plant includes a pot and soil, rendered with appropriate materials.

- Plants are typically placed in room corners to enhance realism.

6. Camera Controls

Navigation

- Standard WASD keyboard keys are used for first-person camera movement (forward, backward, strafe left/right).

- Mouse input controls the camera's view direction (look around).

- Gravity is applied to the camera for more realistic vertical movement.

- Collision detection prevents the camera from passing through walls and other objects.

- A "Reset Camera" button in the UI returns the camera to a predefined default position and orientation.

7. User Interface

Layout

- Instructions Panel: Positioned on the top-left, providing guidance on controls.

- UI Panel (Settings): Located on the top-right, containing controls for light intensity, light position, ambient light, camera reset, and clearing all furniture.

- Furniture Panel: Positioned at the bottom-center, displaying buttons for selecting each of the five furniture types. Each button is color-coded and labeled.

- Loading Screen: An initial loading screen with a progress bar is displayed while assets are being loaded.

8. Asset Loading and Management

Model Loading

- 3D models in GLB format are loaded asynchronously.

- A template-based system is used for furniture: one master model is loaded for each type, and instances are cloned for placement, optimizing performance.

- Progress is tracked during loading, updating the loading screen.

- Error handling is in place for model loading failures, and a timeout (15 seconds) prevents indefinite waiting.

Technical Implementation Details

Core JavaScript Architecture (`main.js`)

Scene Management (`createScene` function)

- Initializes the Babylon.js scene, engine, and canvas.

- Sets up the primary camera with controls, gravity, and collision properties.

- Creates main light sources (point and hemispheric) and the shadow generator.

- Generates the ground mesh with its material and texture.

Wall and Static Element Creation (`createWalls`, `addDoor`, `addOpenableWindow`, `addSimpleShelf`, TV setup)

- Procedurally generates the room walls, ceiling, door, window, shelves, and TV.

- Applies materials and textures to these static elements.

Furniture System (`loadFurnitureMeshes`, `placeFurniture`, `selectFurnitureObject`, `createSelectionHighlight`, `deleteFurniture`)

- `loadFurnitureMeshes`: Asynchronously loads GLB models for each furniture type, creating invisible templates.

- `placeFurniture`: Clones a furniture template, sets its position, scale, rotation, and material, and adds it to the scene.

- `selectFurnitureObject` and `createSelectionHighlight`: Handle the selection of placed furniture and display a highlight mesh.

- `deleteFurniture`: Removes the selected furniture and its highlight from the scene.

Interaction Handling (`setupInteractions`)

- Manages user input from mouse clicks (for placement and selection) and keyboard presses (camera movement, furniture rotation/deletion).

- Binds UI elements (sliders, buttons, dropdowns) to their respective JavaScript functions.

Plant Generation (`addRealisticPlant`, `createFicusPlant`, `createPalmPlant`, `createMonsteraPlant`)

- Procedurally creates different types of plants with pots, soil, trunks, branches, and leaves.

Challenges and Solutions

Model Loading Synchronization

- Challenge: Ensuring models were fully loaded before being used as templates.

- Solution: Utilized Babylon.js SceneLoader callbacks and an asset tracking system (`assetLoaded` function, `totalAssets`, `loadedAssets`) to manage asynchronous loading and trigger scene readiness only when all essential assets were available.

Furniture Positioning and Scaling

- Challenge: GLB models often come with varying inherent scales and pivot points.

- Solution: Applied specific scaling and initial rotation adjustments within the `loadFurnitureMeshes` (for templates) and `placeFurniture` functions for each furniture type to ensure they appear correctly sized and oriented in the scene. For complex items like chairs, dynamic bounding box calculations were used post-placement to adjust height and ground positioning.

Selection System for Complex Objects

- Challenge: Furniture models consist of multiple meshes parented to a root transform node. Picking needed to identify the main furniture group.

- Solution: Implemented a selection logic (`selectFurnitureObject`) that, upon picking a mesh, traverses up its parent hierarchy to find the main `TransformNode` representing the entire furniture item, excluding template nodes.

Performance with Multiple Objects and Shadows

- Challenge: Maintaining a smooth frame rate as more furniture (and thus more shadow casters and complex geometries) are added to the scene.

- Solution: Employed a template system for furniture to reduce redundant geometry data. Optimized shadow generator settings (e.g., `useBlurExponentialShadowMap`, `blurKernel`, 1024 map size). Ensured proper disposal of meshes and materials when furniture is deleted.

User Interface Responsiveness

- Challenge: Ensuring UI controls correctly update scene parameters and provide immediate feedback.

- Solution: Used direct event listeners on HTML UI elements, calling JavaScript functions that modify Babylon.js scene properties (e.g., light intensity, camera position) directly.

Future Enhancement Possibilities

- Save/Load Scene State: Allow users to save their room designs and load them later, possibly using JSON for scene serialization.

- Expanded Furniture Library: Introduce a wider variety of furniture models and categories.

- Material Customization: Enable users to change colors and textures of furniture and room surfaces.

- Measurement Tools: Implement tools to measure distances and areas within the 3D space.

- Drag-and-Drop Placement: Allow users to drag furniture items from a palette directly into the scene.

- Undo/Redo Functionality: Provide a history system for user actions.

- Advanced Lighting Options: More types of lights, and more detailed control over light properties.

Conclusion

The 3D Home Decoration Simulator successfully demonstrates the application of the Babylon.js framework to create an interactive and engaging web-based 3D experience. It provides users with a functional set of tools for basic interior layout and visualization. Key achievements include the implementation of a dynamic furniture placement system, realistic lighting with shadows, procedurally generated decorative elements like plants, and an intuitive user interface for controlling the environment and its contents. The project effectively balances features with performance, running smoothly in modern web browsers. It serves as a solid foundation for a more advanced interior design application.

Project Information

- Development Framework: Babylon.js 5.x

- Core Logic: `js/main.js` (approx. 1600+ lines)

- 3D Assets: 5 furniture models (GLB), 8+ textures (JPG)

- Browser Support: Modern browsers with WebGL 2.0 (Chrome, Firefox, Edge, Safari)

- Performance Target: 60 FPS on standard desktop hardware

Github Link : https://github.com/ahmetmuhittingrkn/GraphProject