VIET NAM NATIONAL UNIVERSITY, HO CHI MINH CITY

UNIVERSITY OF SCIENCE

FACULTY OF INFORMATION TECHNOLOGY

PROJECT REPORT

FOREST ECOSYSTEM SIMULATION

3D Forest Environment using Three.js

Student: Thái Hoàng Bảo Khanh

Student ID: 21126071

Supervisor: Võ Thế Hào – Phạm Thanh Tùng

Year 2025

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to my supervisor for the continuous support, patience, motivation, and immense knowledge throughout this project. Their guidance helped me in all the time of research and development of this project.

I would also like to thank the faculty members of the Department of Information Technology for their valuable lectures and guidance during my academic journey. The knowledge and skills I acquired have been instrumental in completing this project.

My sincere thanks also go to my classmates and friends who provided support, encouragement, and valuable feedback during the development of this project. Their insights and suggestions have significantly contributed to improving the quality of this work.

I am also grateful to the open-source community, particularly the developers of Three.js, for creating and maintaining such powerful tools that made this project possible.

Last but not least, I would like to thank my family for their unconditional love, support, and encouragement throughout my studies. Without their support, this achievement would not have been possible.

DECLARATION

I hereby declare that this project report entitled "Forest Ecosystem Simulation" is my own original work and has not been submitted previously for any academic degree or diploma.

All the information, data, and results presented in this report are authentic and have been obtained through legitimate means. Any sources of information used in this project have been properly cited and acknowledged in the references section.

I understand that any violation of the above statement may result in disciplinary action, including cancellation of my degree, in accordance with the rules and regulations of the university.

Date: 23/12/2025

Thái Hoàng Bảo Khanh

INTRODUCTION

Computer graphics has become an integral part of modern technology, finding applications in various fields such as gaming, simulation, education, and visualization. The ability to create realistic and interactive 3D environments has opened up new possibilities for representing and understanding complex systems.

This project, titled "Forest Ecosystem Simulation," aims to create a simple yet visually appealing 3D representation of a forest environment using web-based technologies. The project serves as an educational exercise in understanding the fundamentals of 3D graphics programming, including scene composition, geometry creation, material application, and lighting effects.

The primary motivation behind this project is to explore the capabilities of Three.js, a powerful JavaScript library for creating 3D graphics in web browsers. By building a forest ecosystem from basic geometric primitives, this project demonstrates how complex visual scenes can be constructed from simple building blocks.

The forest simulation includes various elements commonly found in natural environments, such as trees of different types, rocks, bushes, mushrooms, flowers, and grass. Additionally, the project implements a day/night cycle simulation that changes the lighting and atmosphere of the scene based on the time of day, adding a dynamic element to the otherwise static environment.

This report documents the entire development process of the Forest Ecosystem Simulation project. It begins with an overview of the project objectives and technologies used, followed by a detailed analysis and design phase. The implementation section provides in-depth explanations of the code and techniques used to create various elements of the scene. Finally, the report concludes with an evaluation of the results achieved and suggestions for future improvements.

The report is organized into four main chapters:

- Chapter 1: Overview - Introduces the project, its objectives, scope, and the technologies used.

- Chapter 2: Analysis and Design - Presents the requirements analysis and system design.

- Chapter 3: Implementation - Details the technical implementation of all components.

- Chapter 4: Results and Conclusion - Evaluates the outcomes and discusses future work.

CHAPTER 1: OVERVIEW

1.1. Project Introduction

The Forest Ecosystem Simulation project is a web-based 3D graphics application that creates an interactive visualization of a forest environment. The project utilizes Three.js, a cross-browser JavaScript library and API used to create and display animated 3D computer graphics in a web browser using WebGL.

The concept of the project is to build a stylized, low-poly forest scene using only primitive geometric shapes. This approach not only simplifies the development process but also creates a distinctive visual style that is both aesthetically pleasing and computationally efficient. The low-poly aesthetic has gained popularity in recent years, particularly in indie games and artistic visualizations.

The forest scene consists of multiple elements that together create a cohesive natural environment. These elements include various types of trees (pine trees, round trees, tall pines, and shrub trees), rocks of different sizes and colors, bushes, mushrooms, flowers, and grass patches. Each element is constructed using basic Three.js geometries such as CylinderGeometry, ConeGeometry, SphereGeometry, and IcosahedronGeometry.

One of the key features of this project is the implementation of a day/night cycle simulation. This feature dynamically changes the lighting conditions, sky color, and overall atmosphere of the scene based on a simulated time system. Users can control the time through an interactive control panel, allowing them to observe the forest at different times of day, from sunrise to sunset and through the night.

The project is designed to be entirely client-side, requiring no server infrastructure. It runs directly in modern web browsers that support WebGL, making it easily accessible and deployable. The single HTML file contains all the necessary code, including the Three.js library loaded from a CDN (Content Delivery Network).

1.2. Objectives

The primary objectives of the Forest Ecosystem Simulation project are as follows:

1.2.1. Educational Objectives

- To gain practical experience with Three.js and WebGL-based 3D graphics programming.

- To understand the fundamentals of 3D scene composition, including camera setup, lighting, and object placement.

- To learn how to create and manipulate 3D geometries and materials programmatically.

- To explore techniques for creating dynamic visual effects such as day/night cycles.

1.2.2. Technical Objectives

- To create a complete 3D forest scene using only primitive geometric shapes (no external 3D models).

- To implement realistic lighting that responds to the time of day.

- To develop an interactive user interface for controlling the simulation.

- To optimize the scene for smooth performance in web browsers.

1.2.3. Visual Objectives

- To achieve a cohesive low-poly, stylized aesthetic throughout the scene.

- To create variety in the forest elements through randomization of sizes, positions, and colors.

- To implement convincing atmospheric effects including fog and color transitions during day/night changes.

1.3. Scope and Limitations

1.3.1. Project Scope

The scope of this project encompasses the following aspects:

- Creation of a 3D forest environment with multiple types of vegetation and natural objects.
- Implementation of a complete lighting system with ambient, directional, and hemisphere lights.
- Development of a day/night cycle that affects all visual aspects of the scene.
- Creation of an interactive control panel for time manipulation.
- Implementation of camera controls for scene exploration (orbit, zoom, pan).
- Real-time shadow rendering for enhanced visual realism.

1.3.2. Limitations

The project has the following limitations:

- No complex animations such as wind effects on trees or moving wildlife.
- No realistic ecosystem simulation (growth, interaction between elements).
- No procedural terrain generation - the ground is a flat plane.
- No weather effects such as rain, snow, or clouds.
- No audio elements or ambient sounds.
- Performance may vary depending on the user's hardware and browser capabilities.

These limitations are intentional and align with the project's focus on learning fundamental 3D graphics concepts rather than creating a production-ready application.

1.4. Technologies Used

This section describes the main technologies and tools used in the development of the Forest Ecosystem Simulation project.

Table 1.1: Technologies and Tools Used

Technology	Version	Purpose
Three.js	0.160.0	3D graphics library
HTML5	5	Document structure
CSS3	3	Styling and layout
JavaScript	ES6+	Programming logic
WebGL	2.0	Graphics rendering API

1.4.1. Three.js

Three.js is a cross-browser JavaScript library and application programming interface (API) used to create and display animated 3D computer graphics in a web browser using WebGL. Three.js allows the creation of graphical processing unit (GPU)-accelerated 3D animations using the JavaScript language as part of a website without relying on proprietary browser plugins.

Key features of Three.js used in this project include:

- Scene graph management for organizing 3D objects hierarchically.
- Built-in geometry primitives (Cylinder, Cone, Sphere, Icosahedron, Plane).
- Material system with support for physically-based rendering (PBR).
- Multiple light types (Ambient, Directional, Hemisphere).
- Shadow mapping for realistic shadows.
- OrbitControls for intuitive camera manipulation.

The architecture of Three.js follows a scene graph pattern where all objects are organized in a tree-like structure. The main components include the Scene (container for all objects), Camera (defines the viewpoint), Renderer (draws the scene to the screen), and various Objects (meshes, lights, groups).

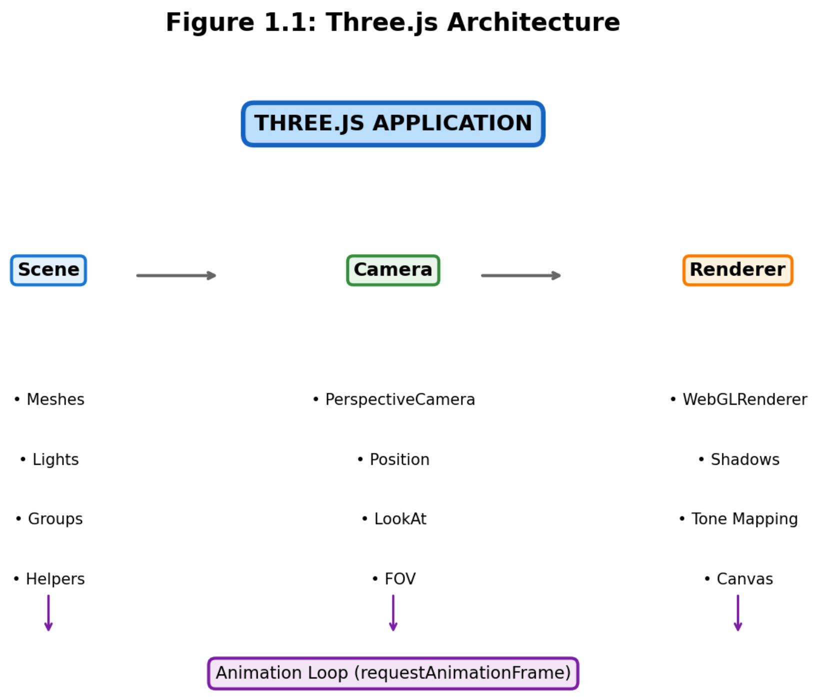


Figure 1.1: Three.js Architecture

1.4.2. Primitive Geometry

Three.js provides a variety of built-in geometry primitives that can be used to construct complex 3D objects. This project utilizes the following primitive geometries:

CylinderGeometry: Used for creating tree trunks and flower stems. The geometry is defined by top radius, bottom radius, height, and the number of radial segments. By varying the top and bottom radii, tapered cylinders can be created for more natural-looking tree trunks.

ConeGeometry: Used for creating conical tree foliage, particularly for pine trees. The geometry is defined by radius, height, and radial segments. Multiple cones of decreasing size are stacked to create layered pine tree shapes.

SphereGeometry: Used for round tree foliage, bushes, mushroom caps, and flower components. The geometry is defined by radius, width segments, and height segments. Lower segment counts create a more faceted, low-poly appearance.

IcosahedronGeometry: Used for creating rocks. The icosahedron's angular faces naturally create a rough, stone-like appearance. When combined with flatShading material property, it produces an attractive low-poly rock aesthetic.

PlaneGeometry: Used for the ground surface. A simple flat plane serves as the forest floor, providing a base for all other objects in the scene.

1.4.3. MeshStandardMaterial

MeshStandardMaterial is a physically-based rendering (PBR) material in Three.js that provides realistic lighting interactions. It is the primary material type used throughout this project for all forest objects.

Key properties of MeshStandardMaterial used in this project:

- color: The base color of the material (e.g., brown for tree trunks, green for foliage).
- roughness: Controls how rough the surface appears (0 = mirror-like, 1 = completely diffuse). Most natural objects use high roughness values (0.8-0.9).
- metalness: Controls how metallic the surface appears (0 = non-metal, 1 = metal). Natural objects use very low metalness values (0.0-0.1).
- flatShading: When enabled, gives the object a faceted, low-poly appearance by not interpolating normals between faces.

The combination of these properties allows for the creation of visually distinct materials for different object types while maintaining a cohesive aesthetic across the entire scene.

Table 1.2: Primitive Geometry Comparison

Object Type	**Color**	**Roughness**	**FlatShading**
Tree Trunk	Brown (0x8B4513)	0.9	No
Foliage	Green (0x228B22)	0.8	Yes
Rocks	Gray (0x696969)	0.9	Yes
Bushes	Dark Green (0x2E8B57)	0.8	Yes
Flowers	Various	0.6	No

CHAPTER 2: ANALYSIS AND DESIGN

2.1. Requirements Analysis

This section presents a detailed analysis of the requirements for the Forest Ecosystem Simulation project. The requirements are categorized into functional and non-functional requirements.

2.1.1. Functional Requirements

Functional requirements define the specific behaviors and functions that the system must provide:

Table 2.1: Functional Requirements

ID	Requirement	Priority
FR01	Display 3D forest scene with trees, rocks, and vegetation	High
FR02	Allow camera rotation, zoom, and pan controls	High
FR03	Implement day/night cycle simulation	High
FR04	Provide time control panel (play/stop)	High
FR05	Display current simulation time	Medium
FR06	Allow manual time adjustment via slider	Medium
FR07	Show scene statistics (object counts)	Low
FR08	Adjust simulation speed	Low

2.1.2. Non-Functional Requirements

Non-functional requirements specify criteria that can be used to judge the operation of the system:

Table 2.2: Non-Functional Requirements

ID	**Requirement**	**Specification**
NFR01	Performance	Maintain 30+ FPS on modern browsers
NFR02	Compatibility	Support Chrome, Firefox, Safari, Edge
NFR03	Responsiveness	Adapt to different screen sizes
NFR04	Load Time	Initial load under 5 seconds
NFR05	Code Quality	Clean, readable, maintainable code
NFR06	Visual Quality	Consistent low-poly aesthetic

2.1.3. Use Case Analysis

The primary user of the system is anyone who wants to view and interact with the forest simulation. The main use cases include:

- View Forest Scene: User can observe the 3D forest environment from different angles.

- Navigate Scene: User can rotate, zoom, and pan the camera to explore the scene.

- Control Time: User can start, stop, and adjust the simulation time.

- Adjust Speed: User can change how fast time passes in the simulation.

- View Statistics: User can see information about the objects in the scene.

2.2. System Design

2.2.1. System Architecture

The Forest Ecosystem Simulation follows a modular architecture organized around the Three.js rendering pipeline. The system consists of several interconnected components that work together to create the interactive 3D experience.

The main architectural components are:

- Initialization Module: Sets up the Three.js scene, camera, and renderer.
- Lighting Module: Manages all light sources including sun, moon, ambient, and hemisphere lights.
- Object Factory Module: Contains functions for creating various forest objects (trees, rocks, bushes, etc.).
- Day/Night Cycle Module: Controls the time simulation and updates lighting/colors accordingly.
- UI Module: Handles the control panel and user interactions.
- Animation Loop: The main render loop that updates and draws the scene each frame.

Figure 2.1: System Architecture Diagram

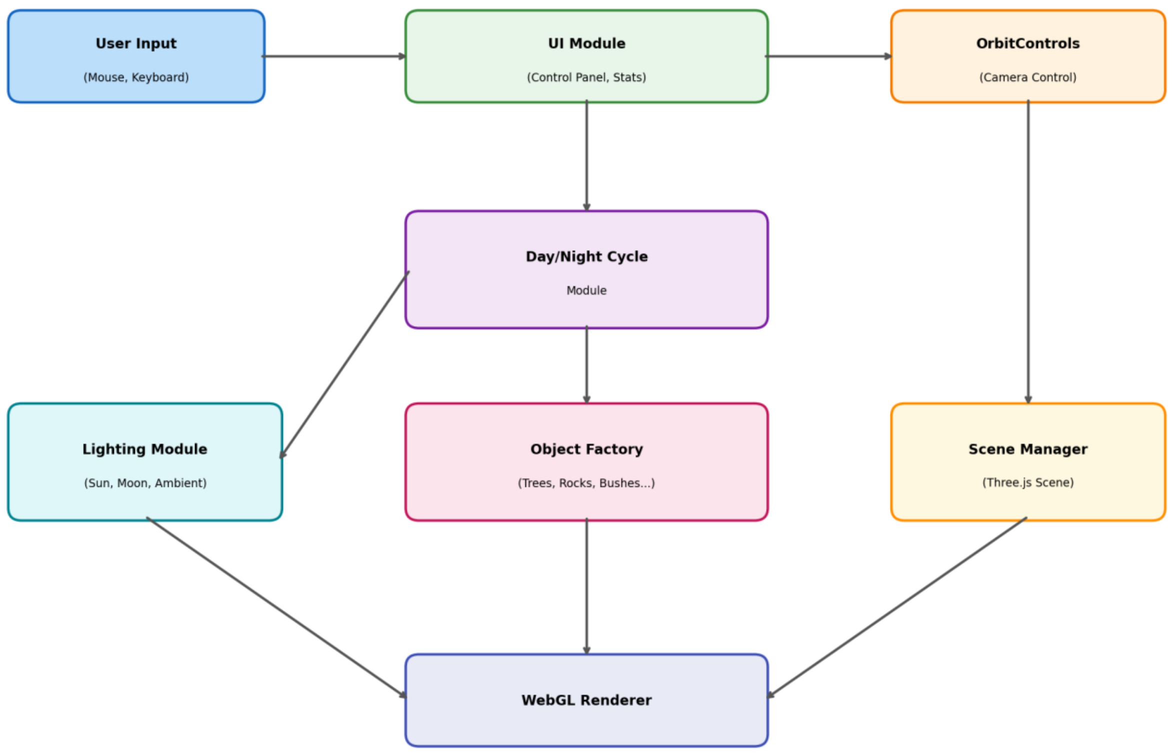


Figure 2.1: System Architecture Diagram

The data flow in the system follows this pattern: User input is captured by the UI module, which updates the simulation state. The animation loop continuously checks the state and calls the Day/Night Cycle module to update lighting. Object positions remain static, but their appearance changes based on lighting conditions. Finally, the renderer draws the updated scene to the screen.

2.2.2. Object Design

Each type of object in the forest scene is designed using a combination of primitive geometries. This section describes the design approach for each object type.

Tree Design:

Trees are composite objects consisting of a trunk and foliage. The trunk is created using CylinderGeometry with a slight taper (larger radius at the bottom). The foliage varies by tree type - pine trees use stacked ConeGeometry shapes while round trees use SphereGeometry. Each tree type has parameters for height, width, and color variation to create natural diversity.

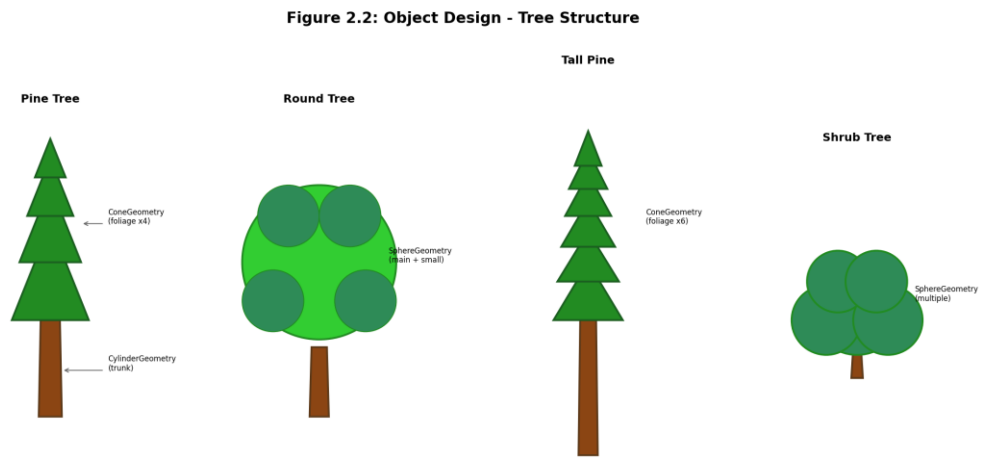


Figure 2.2: Object Design - Tree Structure
Pine Tree
Round Tree
Tall Pine
Shrub Tree
ConeGeometry (foliage x4)
CylinderGeometry (trunk)
SphereGeometry (main + small)
ConeGeometry (foliage x6)
SphereGeometry (multiple)

Figure 2.2: Object Design - Tree Structure

Rock and Bush Design:

Rocks use IcosahedronGeometry as the base shape. The geometry is randomly rotated and non-uniformly scaled to create natural variation. The flatShading property gives rocks their characteristic angular appearance. Rock clusters are created by grouping multiple rocks of varying sizes together.

Bushes are created using multiple SphereGeometry objects grouped together. The overlapping spheres create a natural, organic bush shape. Varying the sizes and positions of the spheres within each bush creates unique instances.

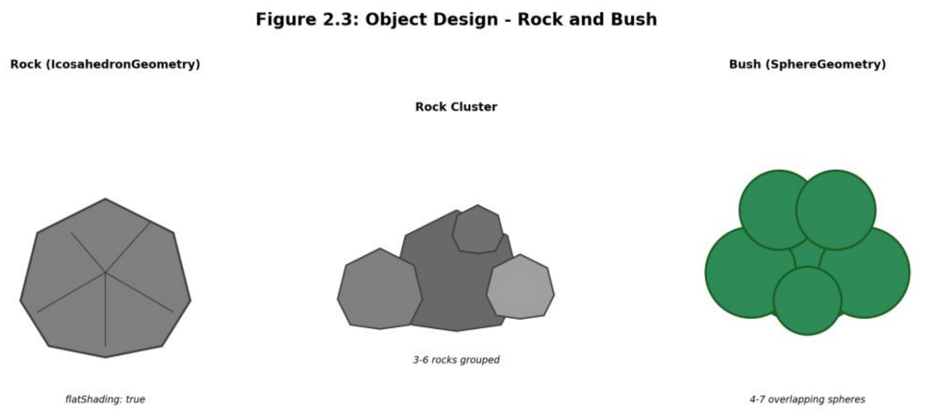


Figure 2.3: Object Design - Rock and Bush

2.3. Interface Design

The user interface is designed to be intuitive and non-intrusive, allowing users to focus on the 3D scene while having easy access to controls.

2.3.1. Layout Design

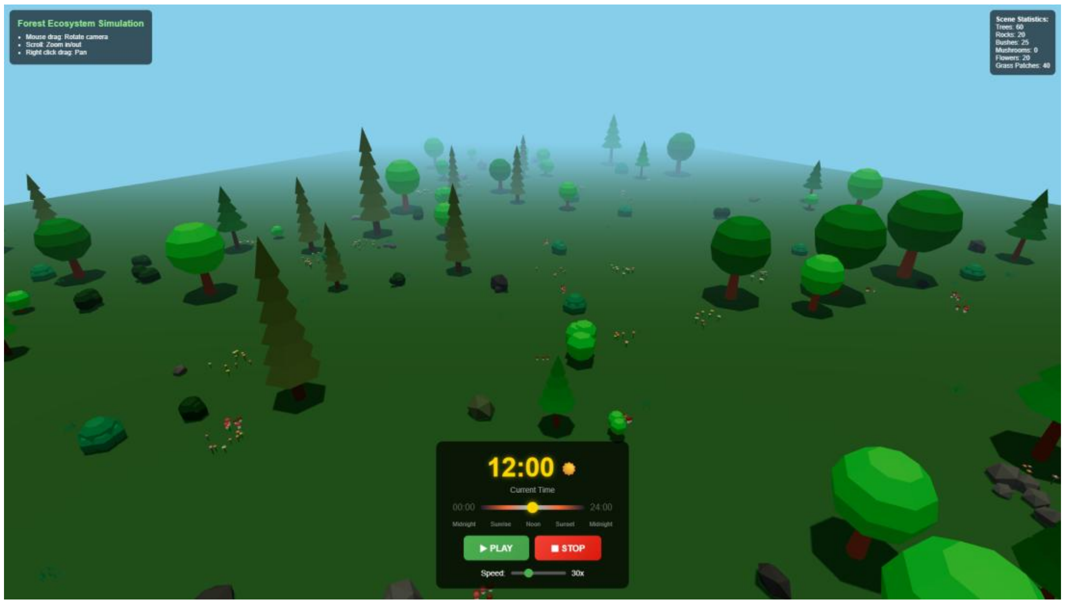
The interface consists of three main areas:

- Main Viewport: The 3D scene occupies the entire browser window, providing an immersive viewing experience.

- Information Panel (Top-Left): Displays the project title and camera control instructions.

- Statistics Panel (Top-Right): Shows the count of different objects in the scene.

- Control Panel (Bottom-Center): Contains all time simulation controls.



Forest Ecosystem Simulation
- Mouse drag: Rotate camera
- Scroll: Zoom in/out
- Right click drag: Pan

Scene Statistics:
Trees: 60
Rocks: 20
Bushes: 25
Mushrooms: 0
Flowers: 20
Grass Patches: 40

Figure 2.4: User Interface Overview

2.3.2. Control Panel Design

The control panel is the primary interface for user interaction with the simulation. It includes:

- Time Display: Large digital clock showing the current simulation time in 24-hour format.
- Sun/Moon Indicator: An emoji that changes based on the time of day.
- Time Slider: Allows direct manipulation of the simulation time from 00:00 to 24:00.
- Play/Stop Buttons: Controls for starting and stopping the automatic time progression.
- Speed Slider: Adjusts how fast time passes in the simulation (1x to 100x speed).

The control panel uses a dark semi-transparent background to ensure visibility against varying scene backgrounds while not obstructing too much of the view.

CHAPTER 3: IMPLEMENTATION

3.1. Project Structure

The Forest Ecosystem Simulation is implemented as a single HTML file that contains all necessary code. This approach simplifies deployment and ensures the project can be run by simply opening the file in a web browser.

The file structure is as follows:

- HTML Structure: Contains the document structure, including div elements for UI panels.

- CSS Styles: Embedded styles for all UI elements and layout.

- Import Map: Configures the module imports for Three.js from CDN.

- JavaScript Module: Contains all application logic, organized into logical sections.

Figure 3.1: Project File Structure

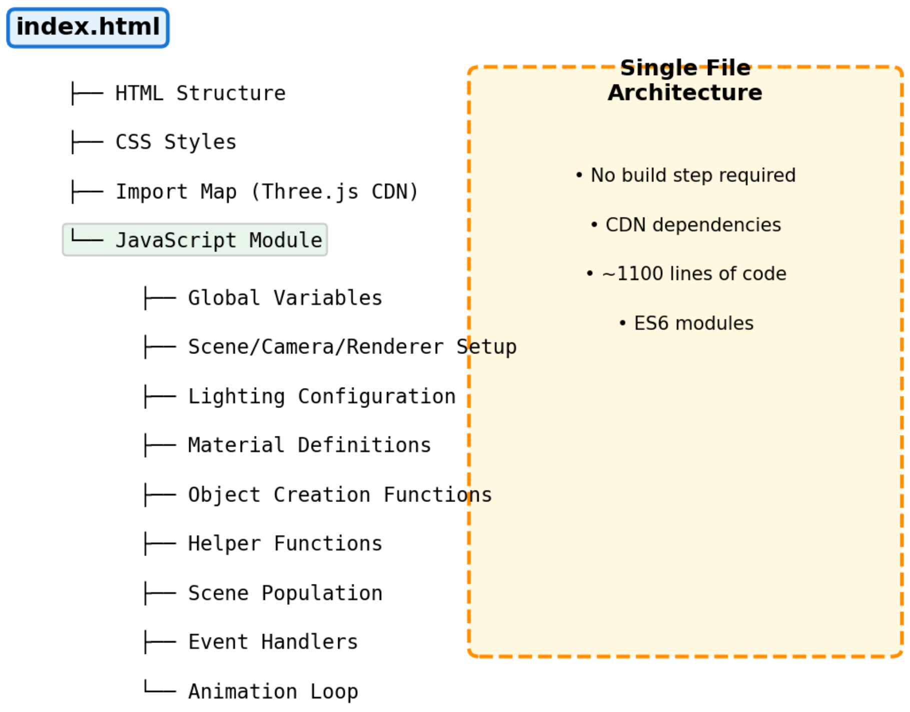


Figure 3.1: Project File Structure

The JavaScript code is organized into the following logical sections:

- Global variables and state management
- Scene, camera, and renderer setup
- Lighting configuration
- Material definitions
- Object creation functions
- Helper functions
- Scene population
- Event handlers
- Animation loop

3.2. Scene, Camera, Renderer Setup

The foundation of any Three.js application is the proper setup of the scene, camera, and renderer. This section describes the configuration of these core components.

3.2.1. Scene Configuration

The scene serves as the container for all 3D objects, lights, and other elements. Key configuration includes:

- Background Color: Set to sky blue (0x87CEEB) for daytime, dynamically changed during day/night cycle.
- Fog: Linear fog is added to create depth and atmosphere. The fog color matches the background and creates a sense of distance.

3.2.2. Camera Configuration

A PerspectiveCamera is used to provide a natural viewing perspective. The camera is configured with:

- Field of View: 60 degrees, providing a balanced view that is neither too narrow nor too wide.
- Aspect Ratio: Dynamically calculated based on window dimensions.
- Near Plane: 0.1 units, allowing close-up viewing of objects.
- Far Plane: 1000 units, ensuring distant objects are visible.
- Initial Position: (20, 15, 20), providing an elevated perspective of the scene.

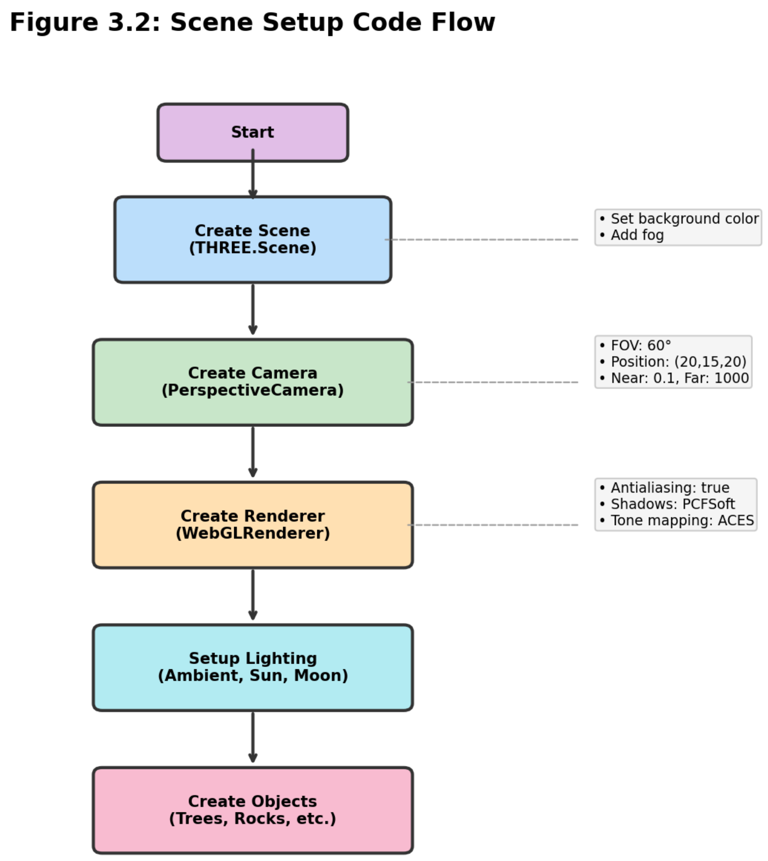


Figure 3.2: Scene Setup Code Flow
Start
Create Scene (THREE.Scene)
• Set background color
• Add fog
Create Camera (PerspectiveCamera)
• FOV: 60°
• Position: (20,15,20)
• Near: 0.1, Far: 1000
Create Renderer (WebGLRenderer)
• Antialiasing: true
• Shadows: PCFSoft
• Tone mapping: ACES
Setup Lighting (Ambient, Sun, Moon)
Create Objects (Trees, Rocks, etc.)

Figure 3.2: Scene Setup Code Flow

3.2.3. Renderer Configuration

The WebGLRenderer is configured for optimal visual quality:

- Antialiasing: Enabled to smooth jagged edges on geometry.
- Pixel Ratio: Set to device pixel ratio (capped at 2) for sharp rendering on high-DPI displays.
- Shadow Map: Enabled with PCFSoftShadowMap for smooth, realistic shadows.
- Tone Mapping: ACES Filmic tone mapping for cinematic color reproduction.
- Exposure: Dynamically adjusted based on time of day.

3.2.4. OrbitControls

OrbitControls from Three.js examples library provides intuitive camera manipulation:

- Damping: Enabled for smooth camera movement.
- Distance Limits: Minimum 5 units, maximum 100 units to prevent extreme zoom levels.
- Polar Angle Limit: Restricted to prevent camera from going below the ground plane.

3.3. Lighting System

The lighting system is crucial for creating a realistic and dynamic visual experience. Multiple light types are combined to achieve the desired effect.

3.3.1. Light Types Used

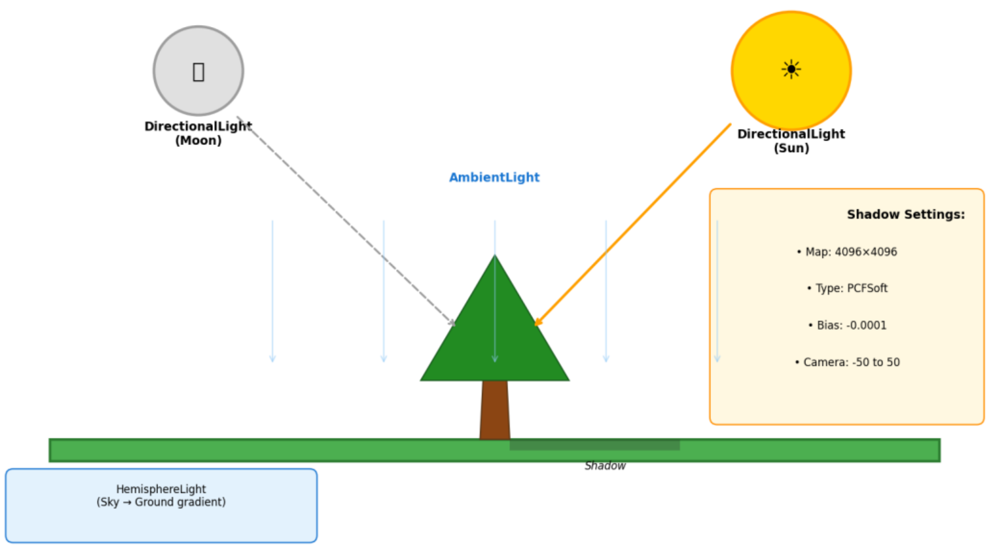
Ambient Light: Provides overall illumination to ensure no areas are completely dark. Intensity varies with time of day.

Directional Light (Sun): Simulates sunlight coming from a specific direction. Configured with shadow mapping for realistic shadows. Position changes during day/night cycle to simulate sun movement.

Directional Light (Moon): Secondary directional light that activates during night time to provide subtle moonlight illumination.

Hemisphere Light: Provides gradient lighting from sky color (top) to ground color (bottom), simulating natural environmental lighting.

Figure 3.3: Lighting Configuration



Shadow Settings:

- Map: 4096×4096
- Type: PCFSoft
- Bias: -0.0001
- Camera: -50 to 50

Figure 3.3: Lighting Configuration

3.3.2. Shadow Configuration

Shadows add significant depth and realism to the scene. The sun light is configured to cast shadows with the following parameters:

Table 3.1: Shadow Configuration Parameters

Parameter	**Value**
Shadow Map Size	4096 x 4096 pixels
Shadow Camera Near	0.5 units
Shadow Camera Far	100 units
Shadow Camera Bounds	-50 to 50 units
Shadow Bias	-0.0001

Objects are configured individually for shadow behavior - most objects cast shadows, while the ground plane receives shadows but does not cast them.

3.4. Forest Objects Creation

3.4.1. Ground Plane

The ground plane provides the foundation for the forest scene. It is created using PlaneGeometry with dimensions of 100x100 units, providing ample space for all forest objects.

The ground uses MeshStandardMaterial with a dark green color (0x3d6b35) to represent grass-covered earth. The plane is rotated 90 degrees around the X-axis to lie flat on the horizontal plane. Shadow receiving is enabled to display shadows cast by trees and other objects.

3.4.2. Trees

Four distinct tree types are implemented, each with unique characteristics:

Pine Tree (createPineTree): Classic conifer design with a tapered cylindrical trunk and multiple stacked cones for foliage. Four cone layers of decreasing size create the characteristic pine shape. Height and color vary randomly for natural diversity.

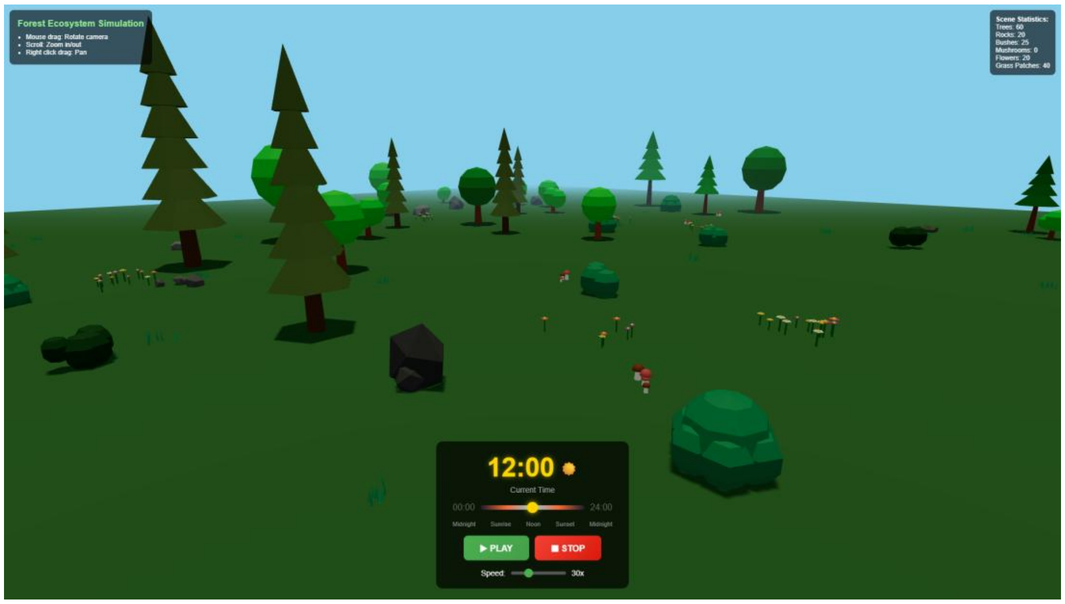
Round Tree (createRoundTree): Deciduous tree design with a trunk and spherical foliage. The main foliage sphere is surrounded by smaller spheres to create a fuller appearance. This tree type represents broad-leafed trees like oaks or maples.

Tall Pine (createTallPine): A taller, more slender variant of the pine tree with six foliage layers. This tree type adds vertical variety to the forest canopy.

Shrub Tree (createShrubTree): A small, bushy tree with a short trunk and multiple overlapping spheres for foliage. This type fills the understory layer of the forest.

Table 3.2: Tree Types and Parameters

Tree Type	**Trunk Height**	**Foliage Type**	**Layers**
Pine Tree	2.5 units	Cone	4
Round Tree	1.8 units	Sphere	1 + 4 small
Tall Pine	4.0 units	Cone	6
Shrub Tree	0.8 units	Sphere	5 random



- Mouse drag: Rotate camera
- Scroll: Zoom in/out
- Right click drag: Pan

Scene Statistics:
Trees: 60
Rocks: 20
Bushes: 25
Mushrooms: 0
Flowers: 20
Grass Patches: 40

Figure 3.4: Tree Models Detail

3.4.3. Rocks

Rocks are created using IcosahedronGeometry, which provides an angular, crystalline appearance perfect for representing stones. Two rock creation functions are implemented:

Single Rock (createRock): Creates an individual rock with random rotation and non-uniform scaling to make each rock unique. Optionally adds a smaller rock nearby for added natural variation.

Rock Cluster (createRockCluster): Groups 3-6 rocks of varying sizes together to create natural rock formations.

Four rock materials are defined with different colors: gray, dark gray, light gray, and mossy (greenish-gray). The flatShading property is enabled for all rock materials to enhance the angular, low-poly appearance.

3.4.4. Bushes

Bushes add ground-level vegetation to the forest. Two bush types are implemented:

Standard Bush (createBush): Creates a bush from 4-7 overlapping spheres of varying sizes, positioned randomly within a small area to create an organic shape.

Large Bush (createLargeBush): A more structured bush with a central sphere surrounded by six smaller spheres arranged in a circular pattern.

Bush materials use dark green colors with flatShading enabled for consistency with the low-poly aesthetic.

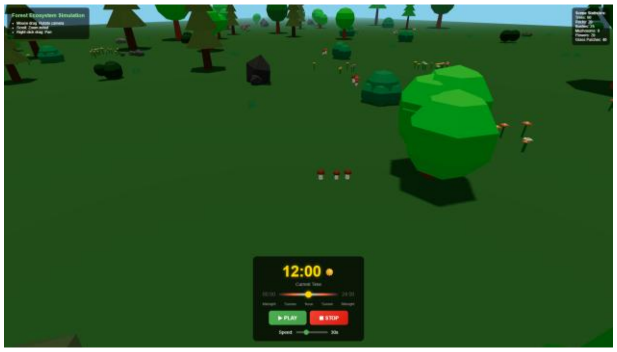


Figure 3.5: Rock and Bush Models

3.4.5. Other Details (Mushrooms, Flowers, Grass)

Smaller details add richness and life to the forest floor:

Mushrooms: Consist of a thin cylindrical stem topped with a hemisphere cap. Created individually and grouped into clusters of 3-7 mushrooms. Cap colors alternate between red and brown.

Flowers: Feature a thin green stem, yellow center sphere, and five petals arranged radially. Petal colors include red, yellow, white, and purple, adding color variety to the scene. Flowers are grouped into patches of 5-13 flowers.

Grass Patches: Collections of 8-16 small cones representing grass blades. Each blade has slight random rotation to simulate natural growth patterns.

3.5. Day/Night Cycle Simulation

The day/night cycle is one of the most visually impactful features of the simulation. It creates a dynamic environment that changes continuously as time progresses.

3.5.1. Time System

Time in the simulation is tracked as minutes since midnight (0-1440, representing 00:00 to 24:00). This simple numeric representation makes calculations straightforward while allowing precise time display.

The time can progress automatically when the simulation is playing, or be manually controlled via the time slider. The speed of time progression is adjustable from 1x to 100x real-time.

3.5.2. Sun and Moon Movement

The sun and moon are represented as visible spheres that move across the sky. Their positions are calculated using trigonometric functions based on the current time:

- Sun position: Calculated using sine and cosine of the time angle, creating a circular path.
- Moon position: Opposite to the sun, ensuring it rises as the sun sets.
- Visibility: Each celestial body becomes invisible when below the horizon.

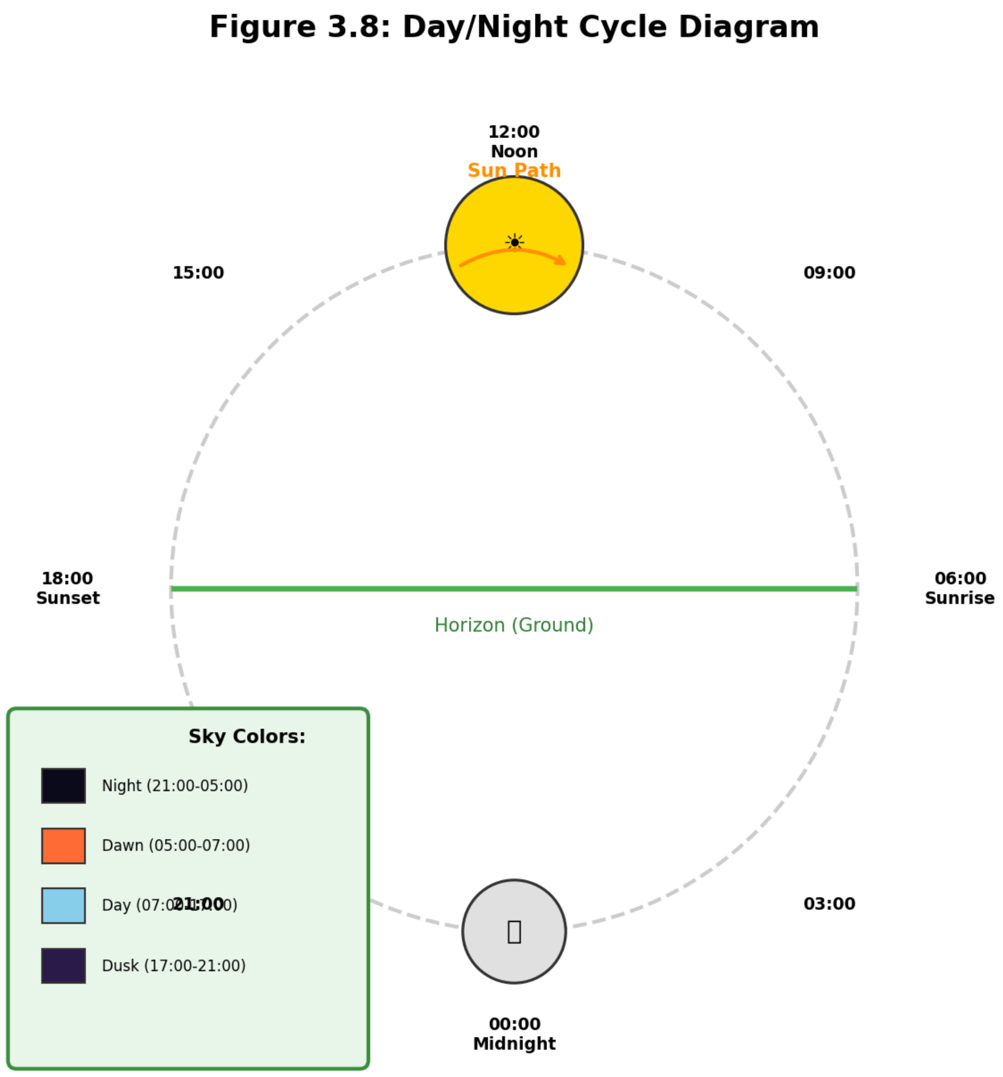


Figure 3.6: Day/Night Cycle Diagram

3.5.3. Color Transitions

The sky color and lighting change smoothly throughout the day/night cycle:

Table 3.3: Color Scheme for Day/Night Cycle

Time Period	**Hours**	**Sky Color**
Night	21:00 - 05:00	Dark Blue (0x0a0a1a)
Dawn	05:00 - 07:00	Orange (0xff6b35)
Morning	07:00 - 09:00	Light Blue (0x87CEEB)
Day	09:00 - 17:00	Sky Blue (0x87CEEB)
Evening	17:00 - 19:00	Orange (0xff6b35)
Dusk	19:00 - 21:00	Purple (0x2a1a4a)

Colors are interpolated smoothly between these key values using Three.js's Color.lerp() function, creating gradual transitions rather than abrupt changes.

3.5.4. Lighting Intensity Changes

Light intensities vary based on time of day:

- Daytime: Sun light at full intensity (1.0), ambient light moderate (0.4).
- Dawn/Dusk: Reduced sun intensity, warm orange tint to sun color.
- Night: Sun light off, moon light at low intensity (0.3), minimal ambient light (0.1).

The renderer's tone mapping exposure is also adjusted to simulate the eye's adaptation to different light levels.

3.6. Time Control Panel

The time control panel provides users with complete control over the simulation's temporal aspects.

3.6.1. Panel Components

The control panel includes the following interactive elements:

Time Display: A large digital clock (48px font) showing the current time in HH:MM format. The golden color and glow effect make it easily readable against varying backgrounds.

Sun/Moon Indicator: An emoji character that changes based on time of day - sun during day, sunrise/sunset during transitions, moon at night.

Time Slider: A range input spanning 0-1440 (minutes in a day). The slider track is color-coded to show the progression from night through day and back to night.

Play/Stop Buttons: Green play button starts automatic time progression, red stop button pauses it. Buttons have hover effects for visual feedback.

Speed Slider: Adjusts time speed from 1x to 100x with real-time display of current speed value.

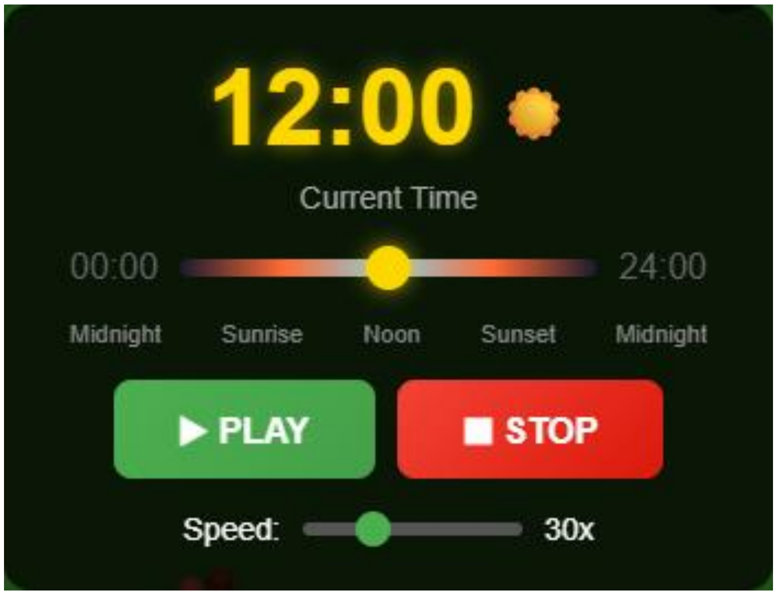


Figure 3.7: Time Control Panel Interface

3.6.2. Event Handling

User interactions are handled through JavaScript event listeners:

- Play Button Click: Sets isPlaying flag to true, enabling automatic time progression in the animation loop.
- Stop Button Click: Sets isPlaying flag to false, pausing time progression.
- Time Slider Input: Directly updates currentTime variable and refreshes the display.
- Speed Slider Input: Updates timeSpeed variable and display label.

3.6.3. Animation Loop Integration

The animation loop is the heart of the real-time simulation. It uses requestAnimationFrame for smooth 60fps updates and performs the following tasks each frame:

- Calculate delta time since last frame for consistent timing.
- If playing, increment currentTime by delta * timeSpeed.
- Wrap time to stay within 0-1440 range.
- Update time display and day/night cycle visuals.
- Update OrbitControls for smooth camera movement.
- Render the scene.

CHAPTER 4: RESULTS AND CONCLUSION

4.1. Achieved Results

The Forest Ecosystem Simulation project has successfully achieved its primary objectives. This section summarizes the key accomplishments and outcomes of the project.

4.1.1. Completed Features

All planned features have been implemented successfully:

- Complete 3D forest scene with diverse vegetation types.
- Four distinct tree types with natural variation in size and color.
- Rock formations including single rocks and clusters.
- Ground-level vegetation: bushes, mushrooms, flowers, and grass.
- Fully functional day/night cycle with smooth transitions.
- Interactive control panel with all planned functionality.
- Responsive camera controls for scene exploration.
- Real-time shadow rendering.
- Scene statistics display.

4.1.2. Scene Statistics

The final scene contains a substantial number of objects, creating a rich and detailed environment:

Table 4.1: Object Statistics in Scene

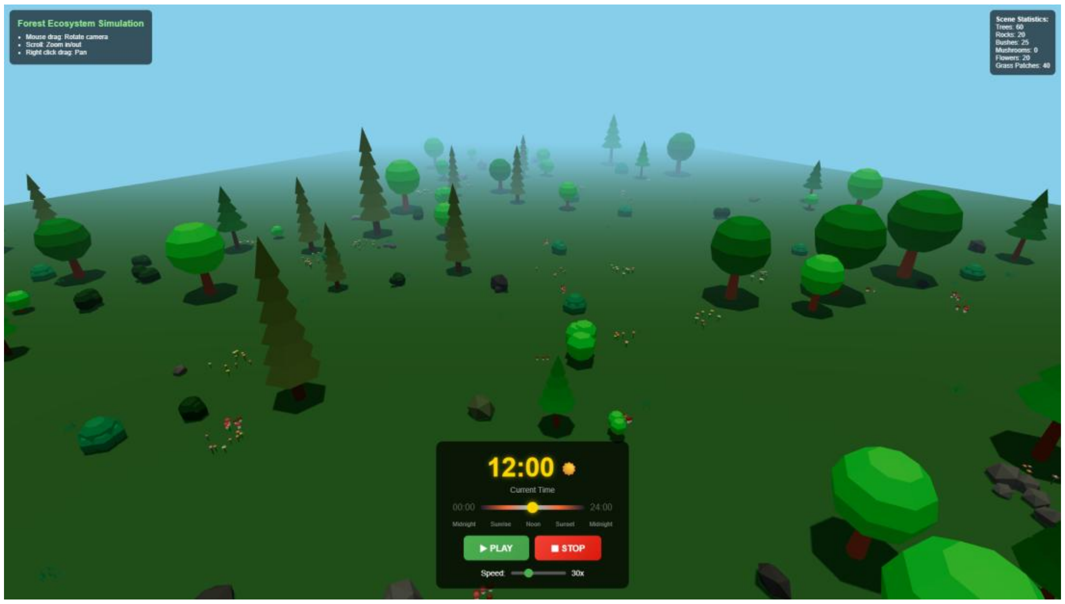
Object Type	**Count**
Trees	~60
Rocks	~20
Bushes	~25
Mushroom Clusters	~15
Flower Patches	~20
Grass Patches	~40

The total number of individual 3D objects in the scene exceeds 500, considering that trees, clusters, and patches contain multiple geometric primitives.

4.2. Screenshots

The following screenshots demonstrate the visual results of the simulation at different times of day.

4.2.1. Daytime View



Scene Statistics:
Trees: 60
Rocks: 20
Bushes: 25
Mushrooms: 0
Flowers: 20
Grass Patches: 40

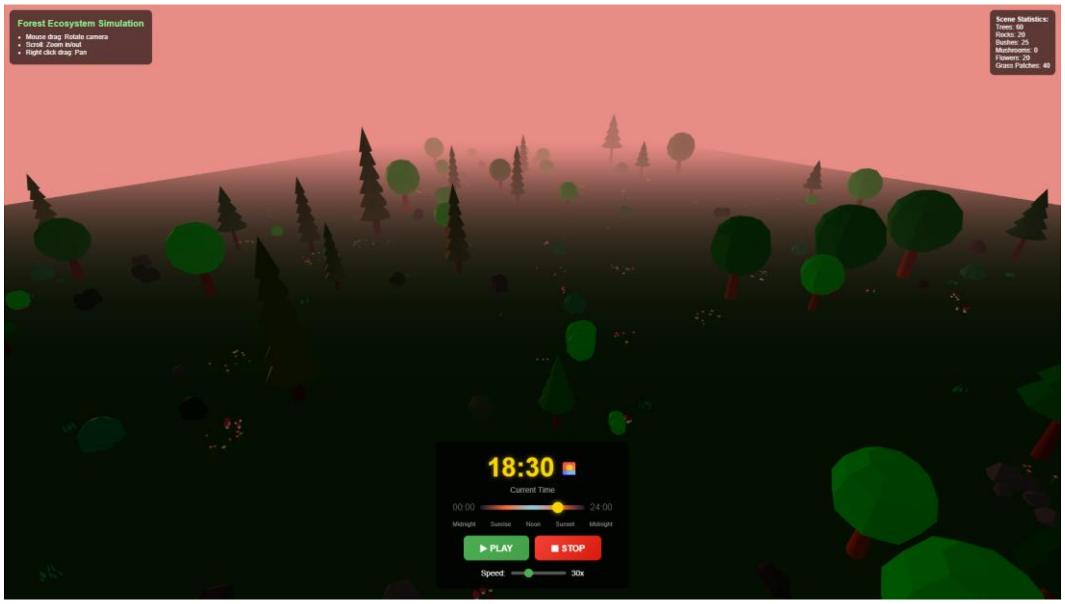
12:00

Speed: 30x

Figure 4.1: Final Scene - Daytime View

The daytime view showcases the full color palette of the forest. Trees display various shades of green, rocks show their natural gray tones, and flowers add colorful accents. Shadows are crisp and well-defined under the bright sunlight.

4.2.2. Sunset View


Forest Ecosystem Simulation
- Mouse drag: Rotate camera
- Scroll: Zoom in/out
- Right click drag: Pan

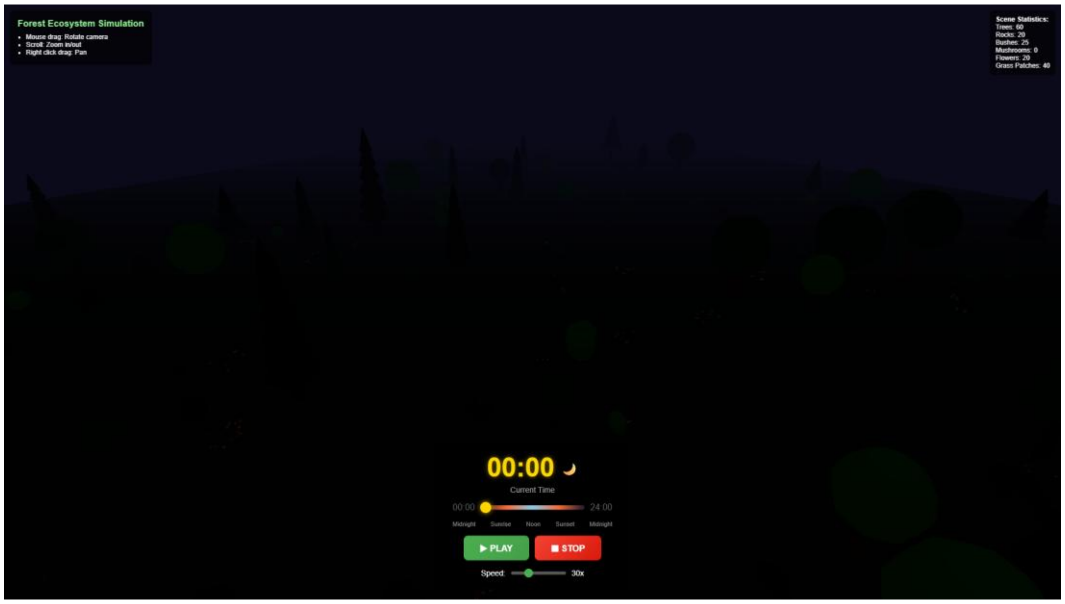
Scene Statistics:
Trees: 60
Rocks: 20
Bushes: 25
Mushrooms: 0
Flowers: 20
Grass Patches: 40

18:30
Current Time
Speed: 30x

Figure 4.2: Final Scene - Sunset View

During sunset, the scene takes on warm orange and pink tones. The sun is visible near the horizon, casting long shadows across the forest floor. This demonstrates the smooth color transition of the day/night cycle.

4.2.3. Night View

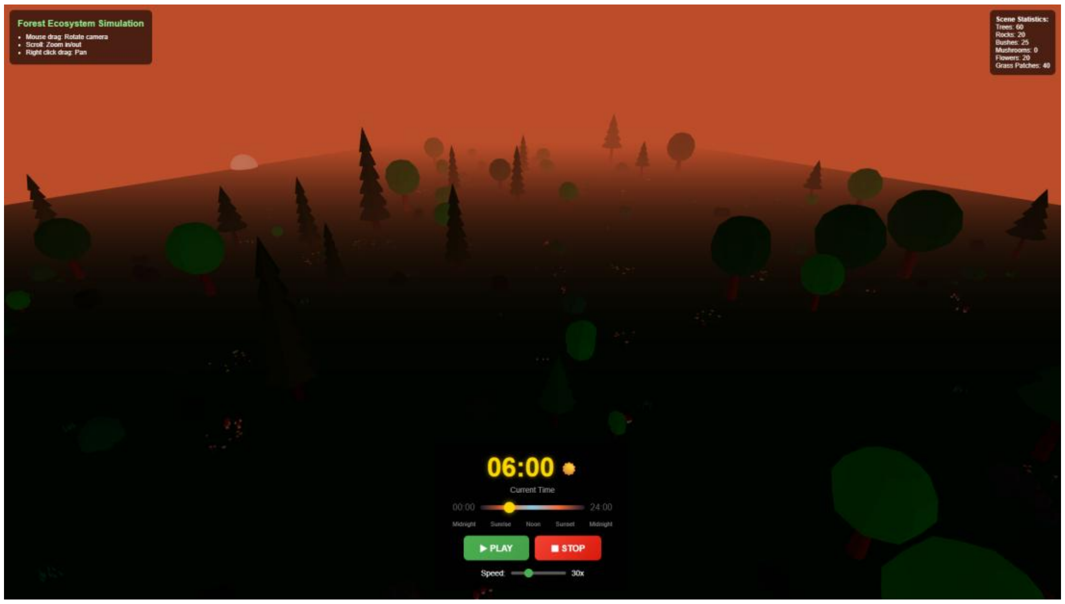


Scene Statistics:
Trees: 60
Rocks: 20
Bushes: 25
Mushrooms: 0
Flowers: 20
Grass Patches: 40

Figure 4.3: Final Scene - Night View

The night view presents a dramatically different atmosphere. The dark blue sky, reduced visibility through fog, and subtle moonlight create a mysterious and peaceful nighttime forest scene.

4.2.4. Dawn View

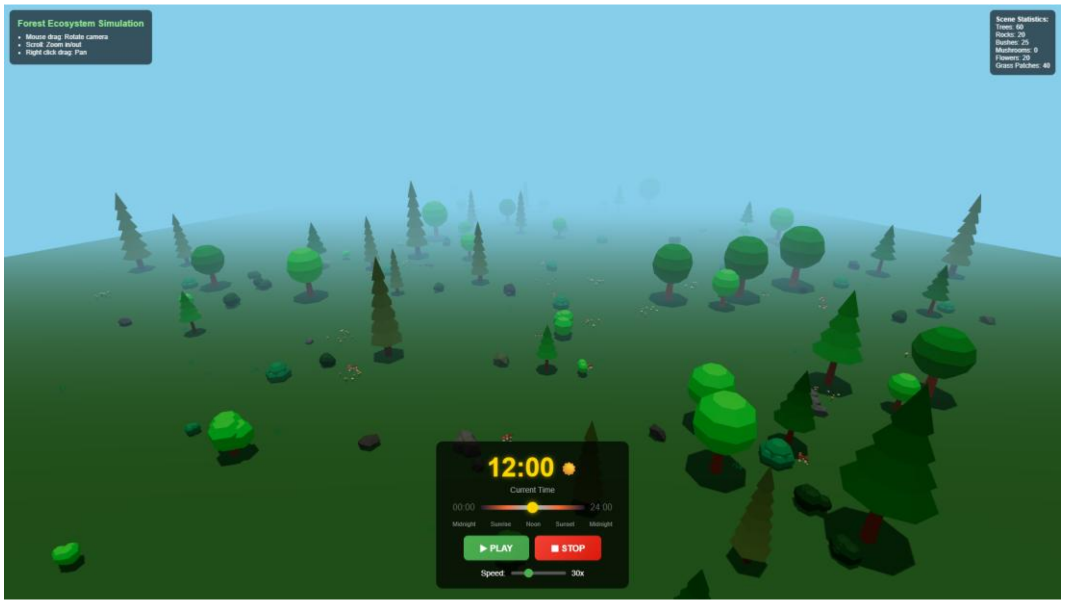


Scene Statistics:
Trees: 60
Rocks: 20
Bushes: 25
Mushrooms: 0
Flowers: 20
Grass Patches: 40

Figure 4.4: Final Scene - Dawn View

The dawn view captures the magical moment when the sun rises. The sky transitions from dark to warm orange hues, creating a beautiful atmosphere that signals the start of a new day in the forest.

4.2.5. Wide Overview



Scene Statistics:
Trees: 60
Rocks: 20
Bushes: 25
Mushrooms: 0
Flowers: 20
Grass Patches: 40

Figure 4.5: Wide Overview Shot

This wide overview shot shows the entire forest scene from an elevated perspective, demonstrating the scale and distribution of objects across the environment.

4.3. Evaluation

4.3.1. Advantages

The project demonstrates several strengths:

- Clean, consistent low-poly aesthetic that is visually appealing and distinctive.
- Smooth and convincing day/night cycle with realistic color transitions.
- Good performance - maintains high frame rates on modern hardware.
- Intuitive user interface that does not obstruct the view.
- Well-organized, readable code that could be extended or modified.
- No external dependencies beyond Three.js - simple deployment.
- Natural variation in objects prevents the scene from looking repetitive.

4.3.2. Limitations

Areas where the project could be improved:

- Static objects - no animations such as swaying trees or moving wildlife.
- Flat terrain - no hills, valleys, or terrain variation.
- No water features such as streams, ponds, or lakes.
- No weather effects or seasonal changes.
- Limited interactivity beyond camera movement and time control.
- No audio elements to enhance immersion.

4.3.3. Performance Analysis

Performance testing was conducted on various hardware configurations:

Table 4.2: Performance Metrics

Hardware	**Browser**	**FPS**
High-end Desktop (RTX 3070)	Chrome	60 (capped)
Mid-range Laptop (GTX 1650)	Firefox	55-60
Integrated Graphics (Intel UHD)	Edge	30-45

The simulation maintains acceptable performance across a range of hardware, though users with integrated graphics may experience reduced frame rates.

4.4. Future Development

Several enhancements could be implemented in future versions of the project:

4.4.1. Visual Enhancements

- Animated elements: Wind effects on trees and grass, flying birds, wandering animals.
- Procedural terrain generation with hills and valleys.
- Water features including a lake or stream with reflections.
- Weather effects: rain, snow, clouds, lightning.
- Seasonal changes affecting tree colors and vegetation.
- Particle effects for falling leaves, fireflies, or dust motes.

4.4.2. Technical Improvements

- Level of Detail (LOD) system for distant objects.
- Instanced rendering for improved performance with many similar objects.
- Texture mapping for more detailed surfaces.
- Post-processing effects such as bloom, depth of field, and color grading.

4.4.3. Interactive Features

- First-person walking mode for immersive exploration.
- Object interaction and information display.
- Customization options for forest density and composition.
- Save and load functionality for custom scenes.

4.5. Conclusion

The Forest Ecosystem Simulation project has successfully achieved its goal of creating an interactive 3D visualization of a forest environment using Three.js and primitive geometric shapes. The project demonstrates that compelling visual experiences can be created using simple building blocks when combined thoughtfully.

Through the development of this project, valuable experience was gained in 3D graphics programming, including scene composition, lighting techniques, material configuration, and animation loops. The implementation of the day/night cycle provided insight into creating dynamic, time-based visual effects.

The low-poly aesthetic adopted for this project proved to be both visually appealing and technically efficient. The consistent application of flatShading materials and carefully chosen color palettes created a cohesive visual style that distinguishes the project.

The project serves as a foundation that could be expanded into more complex simulations. The modular code structure allows for easy addition of new object types or features. The techniques learned can be applied to other 3D visualization projects.

In conclusion, the Forest Ecosystem Simulation successfully meets its educational objectives while producing a visually pleasing and interactive application. The project demonstrates the capabilities of modern web-based 3D graphics and provides a platform for future exploration and development.

REFERENCES

[1] Three.js Documentation. (2025). Three.js - JavaScript 3D Library. Retrieved from https://threejs.org/docs/

[2] Dirksen, J. (2018). Learn Three.js: Programming 3D animations and visualizations for the web with HTML5 and WebGL (3rd ed.). Packt Publishing.

[3] Parisi, T. (2012). WebGL: Up and Running. O'Reilly Media.

[4] Mozilla Developer Network. (2025). WebGL: 2D and 3D graphics for the web. Retrieved from https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API

[5] Three.js Examples. (2025). OrbitControls. Retrieved from https://threejs.org/examples/#misc_controls_orbit

[6] OpenGL Wiki. (2025). Vertex Specification. Retrieved from https://www.khronos.org/opengl/wiki/

[7] Computer Graphics: Principles and Practice (3rd ed.). Hughes, J.F., et al. (2013). Addison-Wesley.

[8] Real-Time Rendering (4th ed.). Akenine-Moller, T., et al. (2018). A K Peters/CRC Press.

APPENDIX

Appendix A: Source Code

The complete source code for the Forest Ecosystem Simulation is contained in a single HTML file (index.html). The file includes:

- HTML structure for the user interface
- CSS styles for layout and visual design
- JavaScript code for Three.js scene setup and management
- Object creation functions for all forest elements
- Day/night cycle implementation
- User interaction handlers

The source code is available in the project repository and totals approximately 1100 lines of code.

Key code sections include:

- Lines 1-170: HTML structure and CSS styles
- Lines 171-300: Scene, camera, renderer, and lighting setup
- Lines 301-550: Material definitions
- Lines 551-950: Object creation functions
- Lines 951-1050: Scene population and statistics
- Lines 1051-1115: Event handlers and animation loop

Appendix B: User Guide

B.1. System Requirements

- Modern web browser with WebGL support (Chrome, Firefox, Safari, Edge)
- Dedicated or integrated graphics capable of WebGL 2.0
- Internet connection (for loading Three.js from CDN)

B.2. Running the Simulation

- Open index.html in a web browser
- Wait for the scene to load (typically 2-3 seconds)
- The forest scene will appear with default settings

B.3. Camera Controls

- Left Mouse Drag: Rotate the camera around the scene
- Right Mouse Drag: Pan the camera
- Mouse Wheel: Zoom in and out

B.4. Time Controls

- Play Button: Start automatic time progression
- Stop Button: Pause time progression
- Time Slider: Drag to set specific time of day
- Speed Slider: Adjust simulation speed (1x - 100x)

The current time is displayed in 24-hour format at the top of the control panel, along with an indicator showing whether it is day or night.