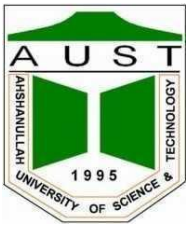
*Ahsanullah University of Science & Technology*
Department of Computer Science & Engineering




# A Helicopter

## Computer Graphics Lab (CSE 4204)

### Project Final Report



| **Submitted By:** | |
|---|---|
| Chandrima Sarker | 20200104131 |
| Tanzima Ahsan | 20200104152 |

## Project Requirements:

This project simulates a 3D helicopter scene where the camera, lighting, and object animations can be interactively controlled by the user. Below are the essential requirements:

**3D Models:**

- A detailed 3D model of a helicopter body.
- A separate 3D model for the helicopter's wings to allow independent rotation.

**Interactions:**

- Keyboard: For controlling the camera's movement around the helicopter.
- Mouse: For adjusting the light's position around the helicopter.

**Animations:**

Rotation of the helicopter's wings during the animation cycle.

**Web Browser:**

A browser with WebGL support is required to render and interact with the 3D scene.

## Software Platform:

**Three.js:** The core JavaScript library used to render the 3D objects and enable user interactions in the browser using WebGL.

**Version:** The latest version of Three.js.

**Modules used:**

- Scene setup and rendering.
- Object loading and manipulation (for the helicopter body and wings).
- Camera handling and perspective view.
- Lighting models and interaction with light sources.
- Animation and transformation functions for dynamic behavior.

**WebGL:** A browser-based API used by Three.js to render 3D scenes efficiently without requiring additional plugins.

**Development Environment:** Visual Studio Code (VS Code) is used to develop and debug the Three.js code.

**Browser:** Any modern browser with WebGL support, such as Google Chrome or Mozilla Firefox.

## Project Features:

This project incorporates several key features, each enhancing the user's interaction with the 3D scene and creating a realistic simulation.

### 3.1. 3D Objects

**Helicopter Body:**

- A static 3D object representing the helicopter's body.
- This object serves as the central visual element of the project, around which the camera and light rotate.

We used 9 different 3D objects in the helicopter body.

**1. Fuselage:** Represents the main body of the helicopter.

**2. Cockpit:** Represents the cockpit of the helicopter.

**3. Main Rotor:**Represents the main rotor of the helicopter.

**4. Tail Boom:** Represents the tail boom of the helicopter.

**5. Tail Rotor:** Represents the tail rotor at the end of the tail boom.

**6. Skid1:** One of the landing skids for the helicopter.

**7. Skid2:** The other landing skid.

**8. SkidConnector1:** Connector for the landing skids.

**9. SkidConnector2:** Another connector for the skids.

**Helicopter Wings:**

- A separate 3D object representing the wings of the helicopter.
- The wings are animated to continuously rotate around their central axis to mimic real helicopter behavior.
- The rotation is implemented using Three.js `rotateY`, `rotateZ` methods within the `requestAnimationFrame` loop, which keeps the animation smooth and synchronized with the rendering.

### 3.2. Keyboard Interaction (Camera Control)

The camera is programmed to move around the helicopter based on keyboard input. The user can explore the scene from different angles using arrow keys or 'W' and 'S' keys.

**Implementation:**

In this project, keyboard interactions were implemented to control the camera's position using the following keys:

**1. Arrow Keys :**

- Arrow Up (`↑`): Moves the camera upward (increases the camera's Y position).
- Arrow Down (`↓`): Moves the camera downward (decreases the camera's Y position).
- Arrow Left (`←`): Moves the camera to the left (decreases the camera's X position).
- Arrow Right (`→`): Moves the camera to the right (increases the camera's X position).

**2. W and S Keys :**

- W : Moves the camera forward (decreases the camera's Z position).
- S : Moves the camera backward (increases the camera's Z position).

These interactions allow the user to move the camera around the scene, providing control over the view of the 3D environment.

## 3.3. Mouse Interaction (Light Control)

The light position can be dynamically controlled using mouse movements. This simulates a rotating light source around the helicopter, casting varying shadows and reflections as it moves. While dragging the mouse, the system calculates how much the mouse has moved. This movement is used to adjust the position of a light source in the scene. Essentially, moving the mouse while holding the button down will rotate the light around the scene.

**Implementation:**

**1. Mouse Down Event :** Begins tracking mouse movements by setting a flag (`isDragging`) to `true` when the mouse is pressed.

**2. Mouse Move Event:** Calculates the change in mouse position (`deltaMove`) as the user drags the mouse. This movement is used to create a rotation effect for the light. Applies a rotation to the light's position using a quaternion derived from the mouse movement, which updates the light’s direction dynamically.

**3. Mouse Up and Mouse Leave Events:** Stops tracking mouse movements by resetting the `isDragging` flag to `false` when the mouse button is released or when the mouse leaves the window.

**4.Storing Previous Mouse Position**: The current mouse position is stored in (`previousMousePosition`) after each movement, so the next move can calculate the delta (difference).

This interaction allows users to change the direction of the light source by dragging the mouse across the screen.

### 3.4. Lighting Model

The lighting model uses:

**1. Ambient Light:** Provides uniform lighting across the scene, ensuring objects are visible even in shadowed areas.

**2. Directional Light:** Simulates sunlight, casting shadows and creating highlights by illuminating objects from a specific direction.

**3. Shadow Configuration:** Adjusts the quality and appearance of shadows cast by lights, contributing to realistic visual effects.

### 3.5. Animation (Wings Rotation)

The helicopter's wings are animated to rotate around a fixed axis, simulating the behavior of helicopter blades in motion.

**Implementation:**

- Main Rotor (`mainRotor`): Rotates around the Y-axis using `mainRotor.rotation.y += mainRotorSpeed`.
- Tail Rotor (`tailRotor`): Rotates around the Z-axis using `tailRotor.rotation.z += tailRotorSpeed`.

### 3.6. Animation (Body Movement)

The helicopter’s x position is incremented by 0.02 units per frame within the animation loop for the body movement.

### 3.7. Helicopter’s Sound

We created an audio listener and attached it to the camera, allowing the sound to be heard from the camera's perspective.

### 3.8. Scene and Rendering

For scene rendering, the following are used:

1. **`THREE.WebGLRenderer`**:
    - It renders the scene using WebGL and is responsible for drawing the scene to the canvas. Antialiasing is enabled for smoother edges.
2. **`renderer.setSize()`**:
    - Sets the size of the rendering canvas to match the window dimensions.
3. **`renderer.setPixelRatio()`**:
    - Adjusts the renderer's pixel ratio based on the device's pixel density for better clarity.
4. **`renderer.shadowMap.enabled`**:
    - Enables shadow mapping, allowing objects to cast and receive shadows.
5. **`renderer.shadowMap.type`**:
    - Sets the shadow map type to a soft shadow effect for more realistic shadows.
6. **`renderer.render()`**:
    - Renders the scene from the perspective of the camera.

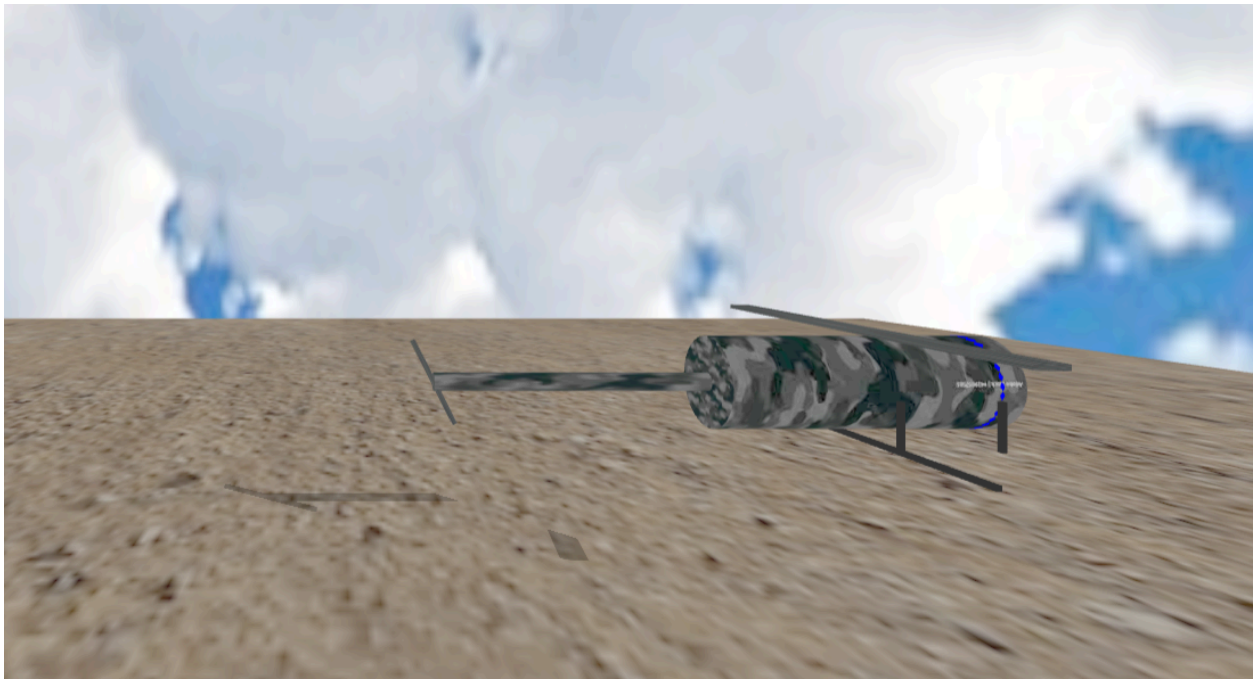
These components work together to display the 3D scene on the web page.

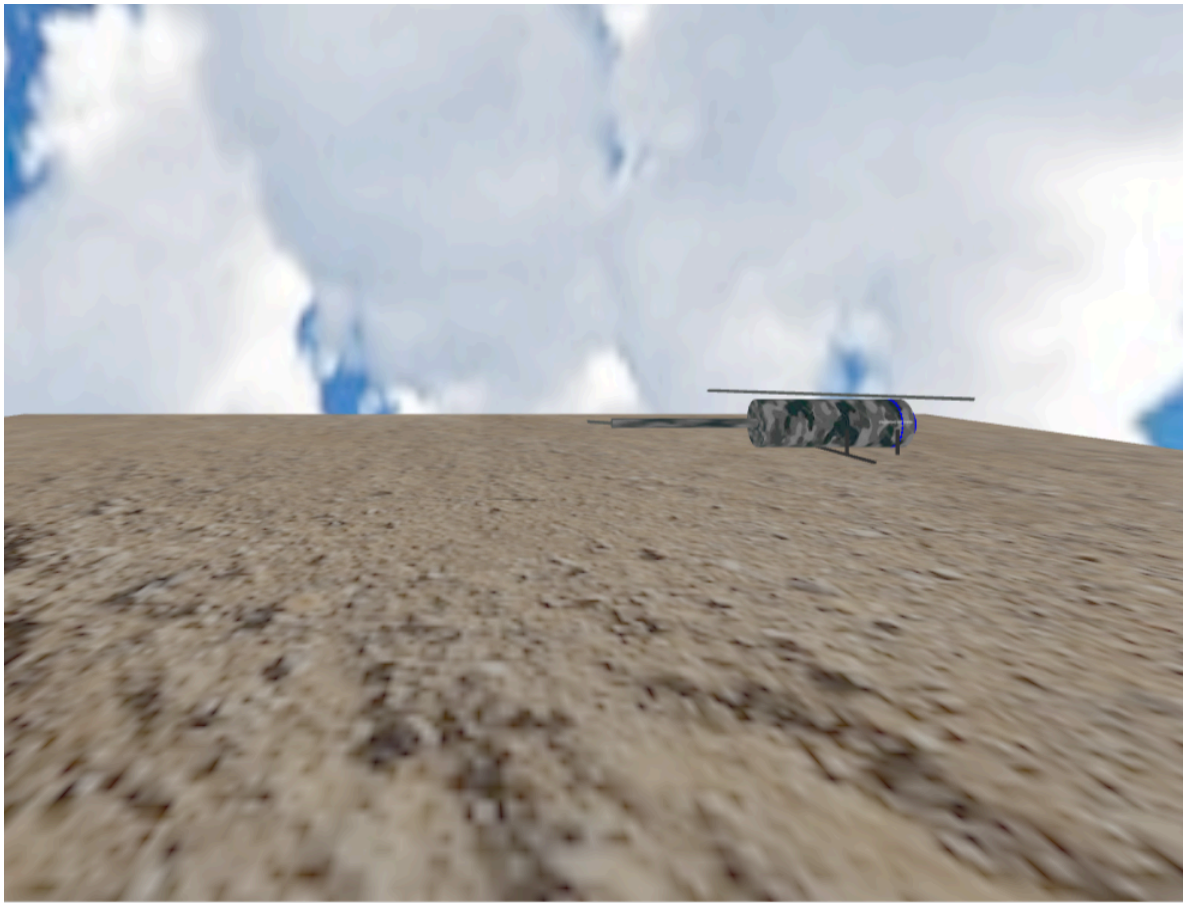
**Attached table with all our required/additional features and classified them into three categories: Implemented, Partially Implemented and Not Implemented.**

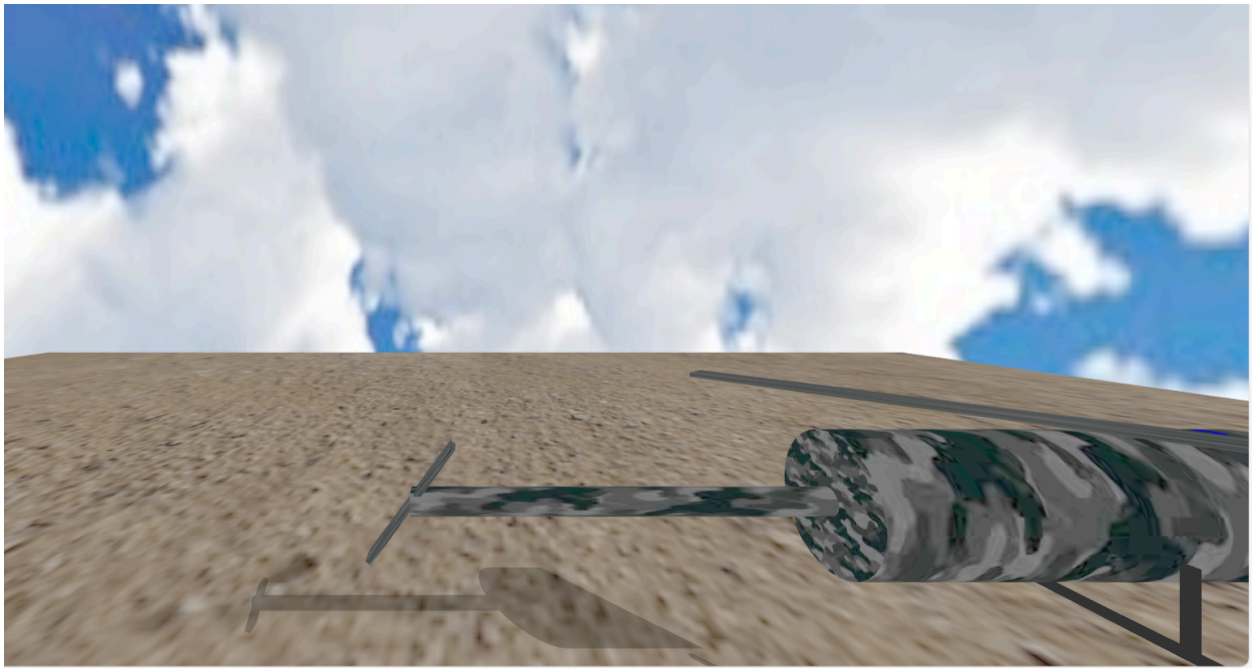
| # | Features | Status |
|---|---|---|
| 1 | Helicopter’s body [with texture] | Implemented |
| 2 | Helicopter’s wings [with texture] | Implemented |
| 3 | Wings rotation | Implemented |
| 4 | Light position rotation | Implemented |
| 5 | Camera movement | Implemented |
| 6 | Helicopter’s movement | Implemented |
| 7 | Helicopter’s sound | Implemented |

Table 01: Project Feature Table

**Snapshots:**







## Contribution:

| ID | Code Contribution | Report Contribution |
|---|---|---|
| **ID:** 20200104131 | Helicopter shape, Keyboard interaction, Helicopter animation | Project requirement, Software platform, Snapshots, Future work |
| **ID:** 20200104152 | Helicopter Sound, Mouse interaction, Ground rotation, Scene Setup | Project features, Future work |

## Future Work:

Here are some potential features that we plan to implement in the future to enhance this project:

1. Helicopter Controls: Add interactive controls for the helicopter, allowing it to take off, land, and move in different directions using keyboard inputs or an on-screen interface.

2. Physics Simulation: Introduce physics-based movement, such as gravity, inertia, and wind effects, to make the helicopter's motion more realistic.

3. Collision Detection: Add collision detection to prevent the helicopter from passing through objects or the ground.

4. Environment Interaction : Add objects like buildings or trees that the helicopter can interact with, enhancing the realism of the scene.

These features would expand the complexity and interactivity of the project, offering a more engaging experience.