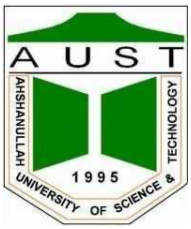
Ahsanullah University of Science & Technology
Department of Computer Science & Engineering



An Asteroid Field

Computer Graphics Lab (CSE 4204)

Project Final Report

Submitted By:	
Israt Jahan Anika	20210104136

Project Requirements:

Three.js Framework:

- **Rendering 3D Objects**: The **Three.js** library is used to render 3D objects like the spaceship and asteroids, as well as handle scene creation, camera controls, and lighting setup.
- **Library Import**: The **Three.js** library is properly imported using an ES6 module system, which ensures that all dependencies are correctly loaded and the scene is rendered smoothly.

3D Scene Setup:

 Scene Creation: The scene simulates a **space environment** where a spaceship is surrounded by rotating asteroids. The background uses a starfield with **space-dust textures**.

 Geometry: The scene includes 3D objects such as:

- **Asteroids**: Created using **DodecahedronGeometry** to give them an irregular shape, making them look more realistic and natural.
- **Spaceship**: A spaceship is modeled using a combination of **CylinderGeometry** for the body and **BoxGeometry** for the wings.
- **Starfield**: A large sphere is used as the background, mapped with a space texture, to give the illusion of being in outer space.

Camera and Movement:

 PerspectiveCamera: A **PerspectiveCamera** is used to simulate a 3D viewpoint from which users can explore the asteroid field.

 Camera Controls:

- **Keyboard**: The **WASD** keys or **Arrow keys** allow users to move the spaceship forward, backward, and side-to-side.
- **Mouse**: The mouse controls the camera's orientation, allowing users to rotate and look around the scene.
- **Camera Reset**: The **R key** resets the camera's position to its default setup.

 Camera Follows Spaceship: The camera follows the spaceship's movement and adjusts its position accordingly.

Lighting Setup:

 Ambient Light: An **AmbientLight** is used to provide general illumination across the scene, ensuring no object is fully dark.

 Directional Light: A **DirectionalLight** simulates sunlight and casts shadows, giving the scene depth and realism.

 Point Light: A **PointLight** is attached to the spaceship to simulate the glow from the spaceship's engine.

 Spotlight: A dynamic **spotlight** follows the spaceship, illuminating its surroundings and creating a visual emphasis on the moving spaceship.

Interactive Textures:

 Spaceship Textures: The spaceship has multiple textures that can be changed interactively by clicking on the spaceship. Each click switches between the spaceship textures (spaceship-blue1, spaceship-yellow, and spaceship-blue2).

 Asteroid Texture: The asteroids are mapped with a **spaceship-yellow texture** for a rocky, space-like look.

Asteroid Creation:

- **Geometry**: **Asteroids** are created using **DodecahedronGeometry** with random perturbations applied to their vertices, making them look irregular and natural.
- **Shaders**: Custom **vertex** and **fragment shaders** are used for the asteroids, providing:
    - **Displacement**: The vertex shader adds subtle organic movement to the asteroids.
    - **Lighting**: The fragment shader calculates the lighting effect based on the asteroid's surface normal, adding realistic ambient and diffuse light.

User Interface Controls:

- **Spaceship Texture Change**: The spaceship's texture can be changed by clicking on the spaceship. The mouse interaction allows users to switch between different spaceship designs.
- **Movement and Camera Controls**: The user can control the movement of the spaceship using the keyboard (WASD/Arrow keys) and rotate the camera with the mouse.
- **Real-time Updates**: The **R** key resets the camera and the **WASD/Arrow keys** allow for spaceship control, making the scene interactive.

Animation Loop:

 Smooth Animation: The animation loop is created using **requestAnimationFrame**, which ensures smooth rendering of the scene.

 Asteroids Movement: The asteroids move and rotate continuously in space, adding dynamic elements to the scene. Their movement is randomized, with their position and rotation being updated in each frame.

 Spaceship Rotation: The spaceship smoothly rotates as the user controls it, following the movement and look direction.

Additional Features:

- **Responsive Rendering**: The **renderer** adjusts to window resizing, ensuring the scene looks good across different screen sizes.
- **Interaction with the Scene**: Users can interact with the scene by moving the spaceship with the keyboard and clicking to change its texture. The camera follows the spaceship's movement, providing an immersive experience as the user explores the asteroid field.

Software Platform:

Three.js:

 Three.js is a powerful **JavaScript library** used to create and render 3D content directly in the browser.

 It handles the rendering of **3D models**, **lighting**, **materials**, **camera controls**, and **scene management** for the project.

 The library is integrated into the project via the **CDN link**:

- Three.js CDN link

Web Browser:

 The project is designed to run directly in a **web browser**, which requires **WebGL** support for rendering 3D graphics.

 The following browsers are recommended as they support **WebGL** and **Three.js rendering** out-of-the-box:

- **Google Chrome**
- **Mozilla Firefox**
- **Microsoft Edge**

HTML:

The core structure of the project is built using **HTML5**.

- It contains the **canvas** element for rendering the 3D scene.
- It also includes **user interface elements**, such as **control buttons** and **status indicators**.
- The **Three.js** code is integrated into the HTML structure to manage and display the 3D scene.

CSS:
 CSS is used to style the page elements, such as:

- The **control panel**.
- The **loading screen**.
- **Status display** elements.

 The **CSS** ensures the page is visually appealing and **responsive**, adapting to different screen sizes and resolutions.

JavaScript:

 JavaScript is the primary programming language used for writing the **logic** and **interactivity** of the project.

 It handles:

- **Scene creation**: Setting up the 3D environment, camera, lighting, and objects.
- **Animation loop**: Creating a continuous rendering loop to animate objects in the scene.
- **User input**: Handling **keyboard** and **mouse interactions** for controlling the camera and the spaceship.
- **Texture management**: Applying and switching textures on objects like the spaceship and asteroids.

VS Code IDE:

 The code for the project is written and tested using **VS Code**.

 It provides helpful features such as:

- **Syntax highlighting** for easy code readability.
- **Code linting** to detect errors or potential issues in the code.
- **Debugging** capabilities to troubleshoot and improve the code.
- Integration with version control tools like **Git** to track and manage code changes.

Project Features:

1. Spaceship

How it works:

- The **spaceship** is created using **Three.js geometries** like **CylinderGeometry** for the body and **BoxGeometry** for the wings and engine.
- The **spaceship texture** changes when the user interacts with it. The textures used are **spaceship-blue1**, **spaceship-yellow**, and **spaceship-blue2**, which can be changed by clicking on the spaceship.
- The spaceship's position is updated in real-time based on **keyboard interactions** (WASD keys) and its orientation changes to match its velocity.
- A subtle rotation animation is applied to the spaceship to give it a more dynamic feel as it moves through the scene.

2. Asteroids

How it works:

 The **asteroids** are created using **DodecahedronGeometry** to simulate irregular rock-like shapes.

 Each asteroid is equipped with **custom shaders** for vertex and fragment shading, which adds a natural movement and lighting effect, simulating a more organic and dynamic look.

 The **asteroids** are placed randomly in the scene, and they rotate and move based on random speed and direction.

 The **textures** for asteroids are applied using **ShaderMaterial** to add realistic surface details.

3. **Keyboard Interaction**

How it works:

 The **camera movement** is controlled by **WASD keys**:

- **'W'**: Moves the camera forward.
- **'S'**: Moves the camera backward.
- **'A'**: Moves the camera left.
- **'D'**: Moves the camera right.

 Camera rotation can be done using the **arrow keys** for rotating the camera in all directions (up, down, left, right).

 'E' and 'Q' keys allow vertical movement of the camera, adjusting the camera height.

4. **Mouse Interaction**

How it works:

- The **mouse controls** allow the camera to **rotate** based on the mouse movement:
  - **X-axis movement** rotates the camera horizontally (left and right).
  - **Y-axis movement** rotates the camera vertically (up and down), with limitations to avoid flipping the camera upside down.
- When the user clicks on the spaceship, the **spaceship's texture** changes to the next one in the sequence. This interaction gives the user a way to control the spaceship's appearance dynamically.

5. **Animation**

How it works:

- The **animation loop** is set up using **requestAnimationFrame()**, ensuring smooth rendering of the scene at optimal frame rates.
- The **spaceship** rotates slightly to simulate a dynamic look as it moves through space.
- The **asteroids** also rotate and move in the scene. Their movement is random, and their rotation speeds are dynamically calculated, making them feel natural and alive.
- The **camera** follows the spaceship in a smooth, dynamic way, ensuring the player always has a good view of the spaceship and surrounding asteroids.

The table below outlines the key features implemented in An Asteroid Field project.

No.	Features	Status
1	Spaceship	Implemented
2	Asteroids	Implemented
3	Keyboard Interaction	Implemented
4	Mouse Interaction	Implemented
5	Animation	Implemented

Table 01: Project Feature Table

Snapshots: The snapshots below showcase key elements of the Asteroid Field Project, providing visual representations of the project's core features. These images highlight the dynamic aspects of the scene, including the spaceship, rotating asteroids, and the interactive texture-changing system. The snapshots also demonstrate the effects of camera movement, keyboard controls, and mouse interactions, illustrating how the user navigates and interacts within the 3D environment. Additionally, the rotation and lighting effects on the spaceship and asteroids emphasize the immersive nature of the asteroid field simulation.

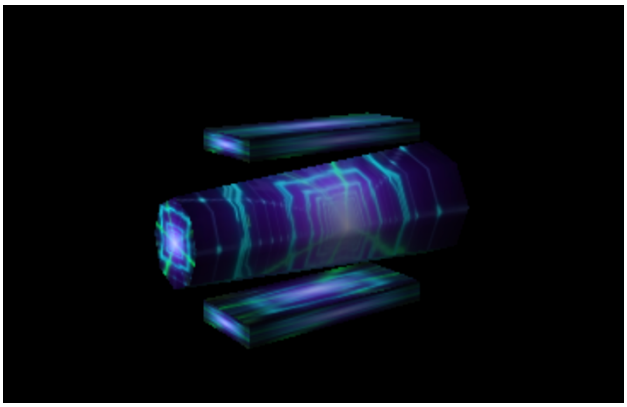


Figure: Spaceship

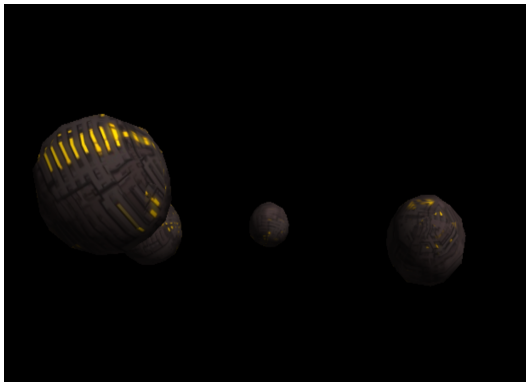


Figure: Asteroid

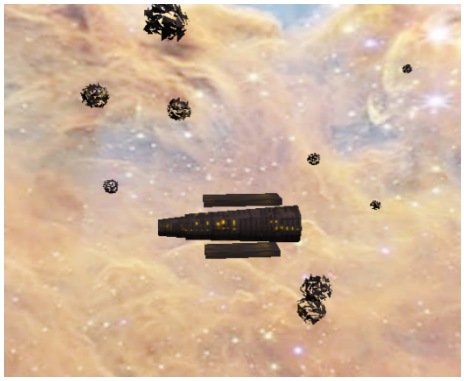


Figure : Left Rotation with Astronoids.

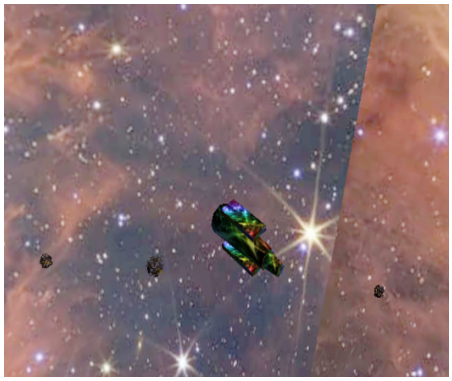
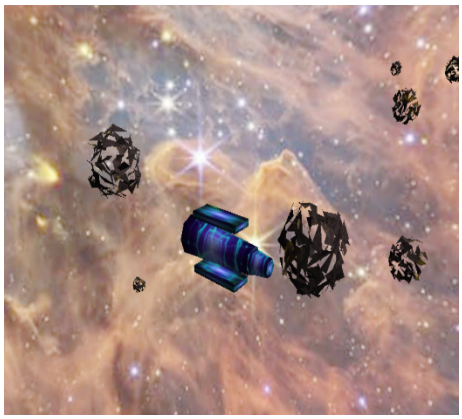


Figure : Right Rotation with Astronoids.



Future Work:

- **Improved Textures and Materials**:
  - Future versions of the project could incorporate **higher-quality textures** for both the **spaceship** and **asteroids**, enhancing their realism. This could include the use of **normal maps** or **bump mapping** for more detailed surfaces.

- **Advanced Lighting Effects**:
  - Implement **interactive lighting** where users can control the **intensity**, **color**, and **direction** of the lights in the scene. This would allow users to further personalize the visual experience of the asteroid field.

- **User-Driven Content**:
  - Allow users to upload their own **spaceship models** and **asteroid textures** to customize the environment. This would add a layer of personalization, making the project more interactive and engaging for users.

- **Enhanced Animation**:
  - Add **dynamic character animations** for the spaceship, such as rotating thrusters, moving wings, or more complex spaceship behaviors.
  - Extend **camera animations** to include more advanced features, such as **guided tours** or **cinematic fly-throughs** that can smoothly animate through the asteroid field and space.