#### **Unit 5 Assignment**

#### **1. Week’s Activities and Observations**

This week was quite engaging as it delved deeper into 3D graphics rendering, particularly focusing on using **Three.js** to create a 3D methane molecule model. The week's activities required us to work with basic shapes, lighting, and transformations, applying principles learned in previous weeks to create a visually accurate representation of methane (CH₄). This assignment helped me strengthen my understanding of geometric shapes and their manipulation in a 3D space.

One surprising aspect was the realization of how essential lighting and shading are to rendering realistic images. At the beginning of the unit, I didn’t anticipate how much of a difference proper lighting makes. I spent a lot of time tweaking the light sources and shadowing effects to make the scene look visually appealing. This deepened my appreciation of how lighting is not just an aesthetic enhancement but a fundamental part of rendering that influences object visibility, depth perception, and realism.

The greatest challenge I encountered was ensuring that the bonds between the methane atoms (represented by cylinders) were correctly positioned and perpendicular to the surface of the carbon atom. Initially, I struggled with using transformations like rotation and translation to get the cylinders in place. To overcome this, I re-visited the concept of quaternion rotations in Three.js and used step-by-step testing to manually adjust and visualize each bond's alignment. Another challenge was handling user interaction using mouse controls for rotating the molecule. I implemented a combination of event listeners and user input event handling to allow smooth manipulation of the molecule without impacting the rest of the scene.

Reflecting on my challenges, I realize how important it is to have a solid grasp of vector mathematics when working with 3D objects. Small details, like getting the bonds correctly aligned, required precise mathematical transformations that took some time to fully understand.

#### **2. Comparison of Rendering in Photoshop vs. Three.js**

Rendering in graphical applications like **Photoshop** is fundamentally different from rendering in a 3D engine like **Three.js**. Photoshop, a 2D raster-based application, primarily focuses on the manipulation and editing of pixels. When you "render" an effect in Photoshop—such as adding a drop shadow or a gradient—you’re applying pre-determined filters and effects that modify the appearance of existing pixels. In essence, rendering in Photoshop is a two-dimensional operation where the user edits or enhances the visual output through pixel manipulation.

In contrast, **Three.js** and other 3D graphics engines deal with rendering in a three-dimensional space. The process here is much more complex because the scene being rendered exists in a virtual 3D environment. Objects in Three.js are represented by vertices and faces that form meshes. The position of these objects, the lighting, shadows, camera angles, and textures all affect the rendering process. For instance, rendering the methane molecule required positioning it in a virtual space, lighting it from different angles, casting shadows on a plane, and enabling user interaction to rotate the object.

One key difference between these two types of rendering is how **perspective** and **depth** are handled. In Photoshop, there’s no inherent notion of depth unless you simulate it through effects like drop shadows or gradients. However, in Three.js, depth is an integral part of the rendering pipeline. The camera’s perspective, combined with light sources and shadows, provides a true sense of 3D depth that’s dynamically calculated as the user interacts with the scene.

Moreover, **real-time rendering** in Three.js is vastly different from Photoshop’s non-real-time rendering. Photoshop processes and applies effects directly to static images, whereas Three.js generates images in real time based on the object’s geometry, the position of light sources, and the camera’s position. This means the final rendered output in Three.js changes dynamically as objects or light sources move, while in Photoshop, you must apply specific changes manually to modify the image.

Another significant contrast is the **interaction and manipulation** of objects. In Photoshop, manipulation is largely mouse-driven for 2D adjustments, such as resizing or rotating layers. On the other hand, in Three.js, objects can be transformed in a 3D space along the x, y, and z axes. The interactions involve not just scaling or rotating but also setting up cameras and lighting to capture the right angle for rendering.

To summarize, while both rendering in Photoshop and Three.js involve modifying visual elements, their underlying processes are entirely different. Photoshop focuses on pixel-based manipulation within a 2D space, while Three.js involves complex geometric and lighting calculations to render objects within a 3D environment. This week’s assignment highlighted the importance of understanding these differences, especially when working with a dynamic, real-time rendering engine like Three.js.

#### **References:**

* Danchilla, B. (2012). *Beginning WebGL for HTML5*. Apress.
* Three.js Documentation. (n.d.). *Three.js Fundamentals*. Retrieved from <https://threejs.org/docs/index.html>
* Adobe. (n.d.). *Photoshop User Guide*. Retrieved from<https://helpx.adobe.com/photoshop>