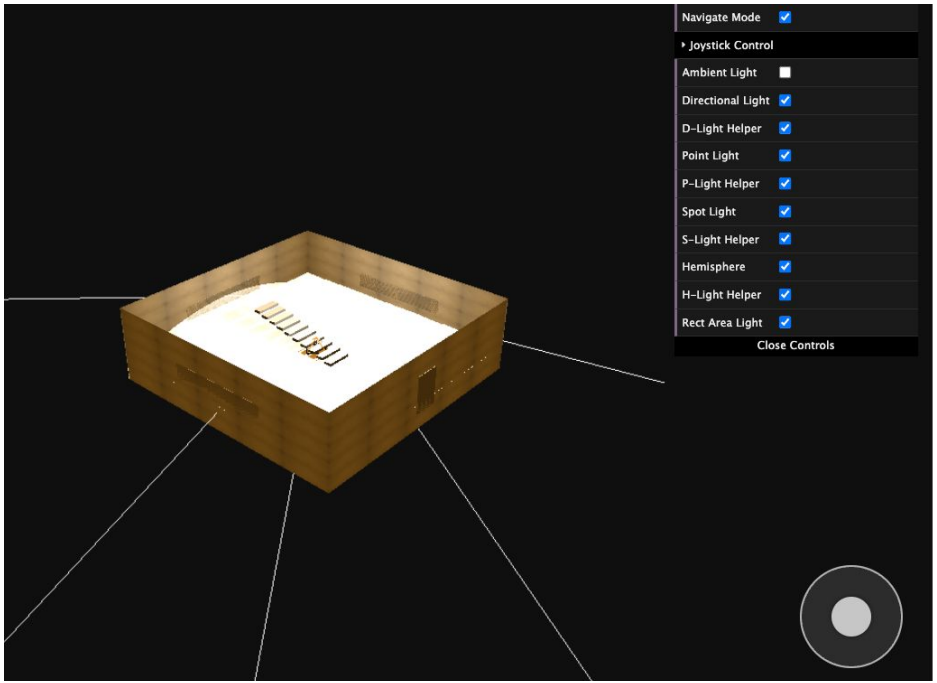
Computer Graphics #3

Shading + Texture Mapping

Seungwonlee

Summary



01 —— Shading and texture mapping

02 —— Navigate with trackpad and keyboard

03 —— Control lightSource with GUI

Scene, Camera, Renderer

- Scene: The 3D environment holding all objects like walls, lights , etc.
- Camera: Captures the scene from a chosen viewpoint, enabling perspective control.
- Renderer: Transforms the 3D scene into a 2D image displayed in the browser.

```
	const scene =
new THREE.Scene();
	scene.background =
new THREE.Color(0x0f0f0f);

	const camera =
new THREE.PerspectiveCamera(45,
window.innerWidth / window.innerHeight, 0.01, 1000);
	camera.position.set(0, 20, 40);

	const renderer =
new THREE.WebGLRenderer({  });
	renderer.setSize(window.innerWidth,
window.innerHeight);
	renderer.shadowMap.enabled =
true;

document.body.appendChild(renderer.domElement);
```

GUI , trackpad with control circle Controls

- The controls object has a property navigate that can be toggled, and it is labeled “Navigate Mode” in the GUI.
- New folder in the GUI named “Joystick Control” to group related controls
- add controls for joystickX and joystickY properties of the controls object, allowing values between -1 and 1
- Function updates the camera’s position based on the joystick inputs if “Navigate Mode” is enabled

```
gui.add(controls, 'navigate').name("Navigate Mode");
const joystickFolder = gui.addFolder('Joystick Control');
joystickFolder.add(controls, 'joystickX', -1, 1).name("X-Axis");
joystickFolder.add(controls, 'joystickY', -1, 1).name("Y-Axis");

function applyJoystickControls() {
    if (controls.navigate) {
        camera.position.z -= controls.joystickY * controls.moveSpeed;
        camera.position.x += controls.joystickX * controls.moveSpeed;
    }
}
const trackpad = document.getElementById('trackpad');
const controlCircle = document.getElementById('controlCircle');
let isDragging = false;
let offsetX = 0, offsetY = 0;

controlCircle.addEventListener('mousedown', (event) => {
    isDragging = true;
    offsetX = event.clientX - controlCircle.getBoundingClientRect().left;
    offsetY = event.clientY - controlCircle.getBoundingClientRect().top;
});
```

Applying Textures-1

- Textures on surfaces
- TextureLoader: Importing textures into the scene.
- RepeatWrapping: Repeats texture for seamless coverage.

Applying Textures-2

- VSCode Live Server Extension: the Live Server extension. This will launch your project in a local server environment:
- Open the HTML file in VSCode.
- Right-click the file and select “Open with Live Server”.
- This can solving the CORS issue.
- Web application needs to fetch data from an API hosted on a different domain. CORS provides a way to relax these restrictions securely

Access to image at 'file:///Users/users/StudywithThreeJS/StudywithThreeJS/Modeling/wall_texture.jpg' from origin 'null' has been blocked by **CORS policy**: Cross origin requests are only supported for protocol schemes: chrome, chrome-extension, chrome-untrusted, data, http, https, isolated-app. Understand this error
wall_texture.jpg:1

Failed to load resource: net::ERR_FAILEDUnderstand this error

Materials

- Texture: Uses floor texture for the floor’s appearance.
- Material Type: MeshPhong Material is used for shiny surfaces with specular highlights.
- Texture: Uses wall texture for the wall’s appearance.
- Double-sided: side: THREE. DoubleSide ensures the material is rendered on both sides of the wall geometry.
- Transparency: transparent: true, and opacity: 0.5 make the window semi-transparent.

```
const materials = {

        floor: new THREE.MeshPhongMaterial({ map: floorTexture }),

        wall: new THREE.MeshPhongMaterial({ map: wallTexture, side: THREE.DoubleSide }),

        window: new THREE.MeshPhongMaterial({ color: 0x878787, transparent: true, opacity: 0.5 }),

        door: new THREE.MeshPhongMaterial({ color: 0x8B8B8B }),

        roof: new THREE.MeshPhongMaterial({ color: 0xffffff }),

        stair: new MeshPhongMaterial({ color: 0x606060 })

    };
```

Room Structure Functions

- createFloor(): Creates the floor mesh and sets its position.
- createWall(): Creates walls at specified locations and orientations.
- createWindow(): Creates a window with optional rotation for side placement.
- createDoor(): Creates a door at the front of the room.
- createRoof(): Creates a roof to cover the room.
- createStairs(): Creates a staircase within the room.

```
function setupRoom() {
    createFloor();
    createWalls();
    createWindows();
    createDoor();
    createRoof();
    createStairs();
}
```

Lighting and Shadows

- Ambient Light: Provides overall, base illumination.
- Directional Light: Mimics sunlight with adjustable direction and shadows.
- Point Light: Creates a localized light source with adjustable position.
- Spot Light: Casts a focused beam of light with adjustable direction and spread.
- Hemisphere Light: Provides a gradient light, simulating the sky and ground light. It does not cast shadows.
- Rect Area Light: Light from a rectangular area, useful for simulating large light sources like windows. It can cast soft shadows.

```
function setupLights() {
    createAmbientLight();
    createDirectionalLight();
    createPointLight();
    createSpotLight();
    createHemisphereLight();
    createRectAreaLight();
}
```