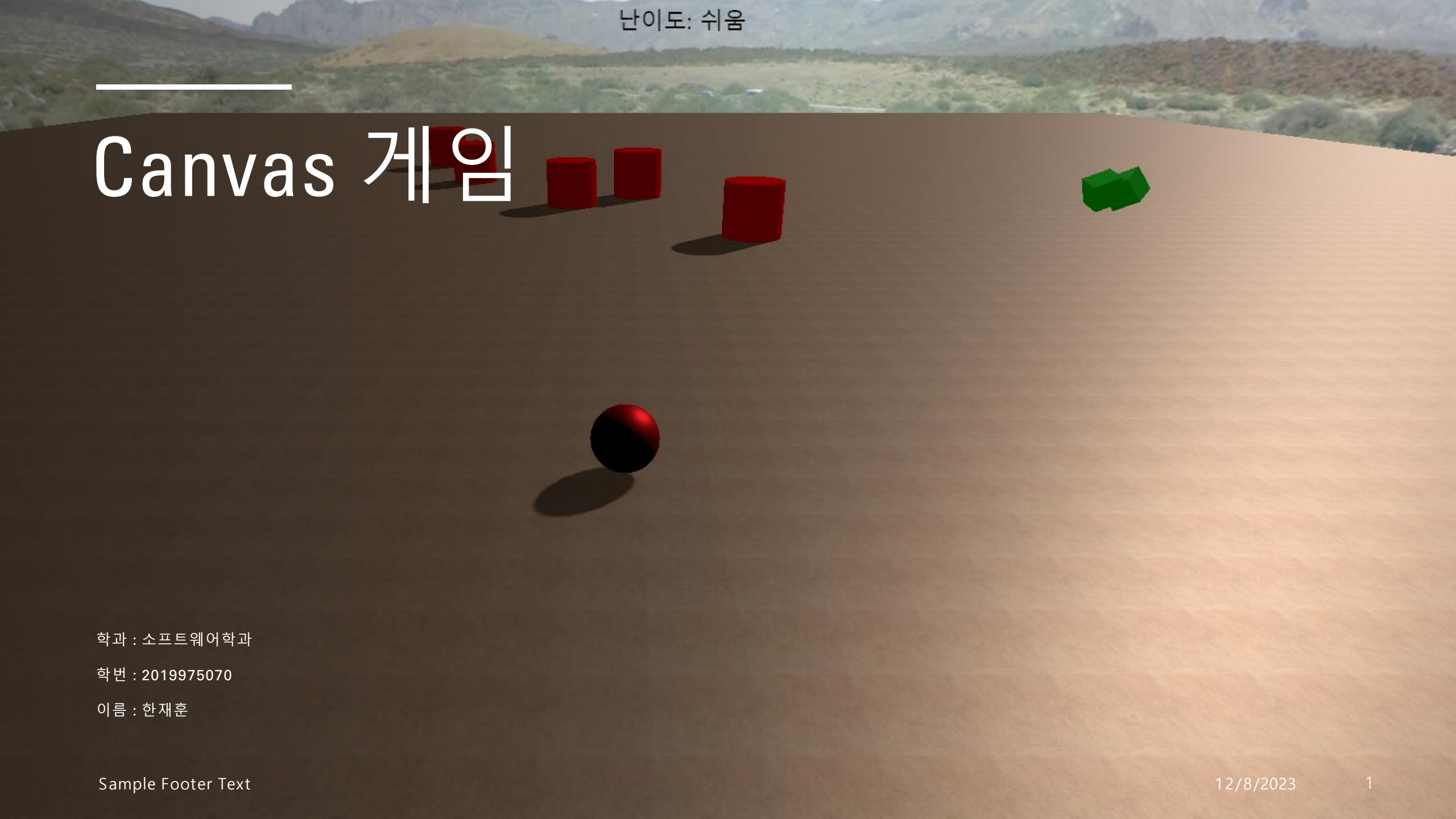


Canvas 게임

학과 : 소프트웨어학과

학번 : 2019975070

이름 : 한재훈

목차

- **게임 소개**
  - Title.html, Scene.html, Gameover.html
- **사용한 라이브러리 설명**
  - Three.js,Cannon.js,howler.js,GLTFLoader.js
- **주요함수 설명**

게임 소개

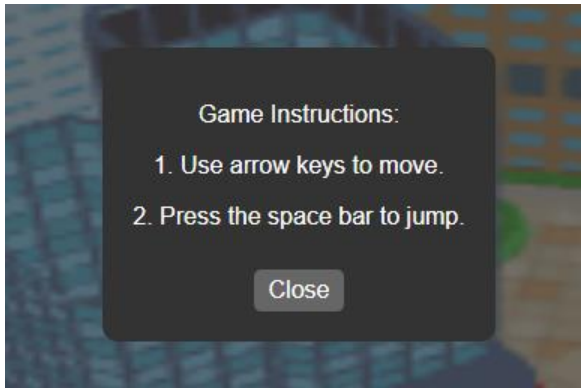
초기 계획

- 초기 계획은 Three.js를 활용한 뱀서라이크 게임 제작이었습니다. 하지만 Three.js를 처음 다루어 보고 제작 시간간 역시 짧아 아쉽게도 적이 랜덤한 위치에서 생성되고, 생성된 적이 플레이어를 쫓아오는 로직만 구현하였습니다. 원본 소스는 따로 없으며, Three.js의 기능을 공부할겸 아무것도 없는 상태에서 하나하나 구현해보았습니다.

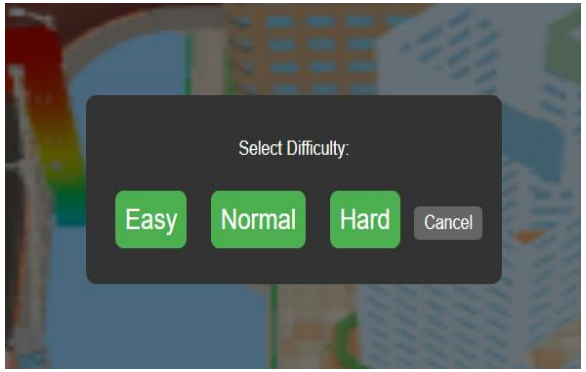
TITLE.HTML

타이틀 화면

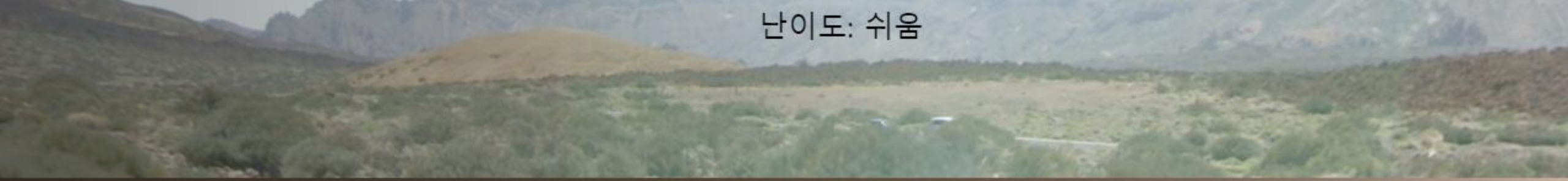
- 이 HTML 파일은 게임 타이틀 화면을 구현한 것입니다. Three.js와 Cannon.js 라이브러리를 사용하여 3D 그래픽스와 물리 엔진을 적용하였고, Howler.js 라이브러리를 통해 배경음악을 재생합니다. 캔버스를 통해 GLTF 3D 모델 파일을 GLTF 로더 라이브러리를 통해 로드하여 렌더링 하였고 rotation을 통해 회전하는 애니메이션이 재생 됩니다. 좌측에는 각각 "Start Game", "Game Instructions", "Exit Game" 버튼이 있습니다.
- **Start Game :** 버튼을 눌렸을때 난이도 선택 모달이 표시되고 난이도를 선택하면 난이도의 값을 Local Storage에 저장한뒤 값을 Scene.html에 전달합니다다.
- **Game Instrucitons :** 버튼을 눌렸을때 게임 설명 모달이 표시됩니다. "Close" 버튼을 누르면 닫힙니다.
- **Exit Game :** html을 닫습니다.



<게임 소개 버튼을 눌렸을때>



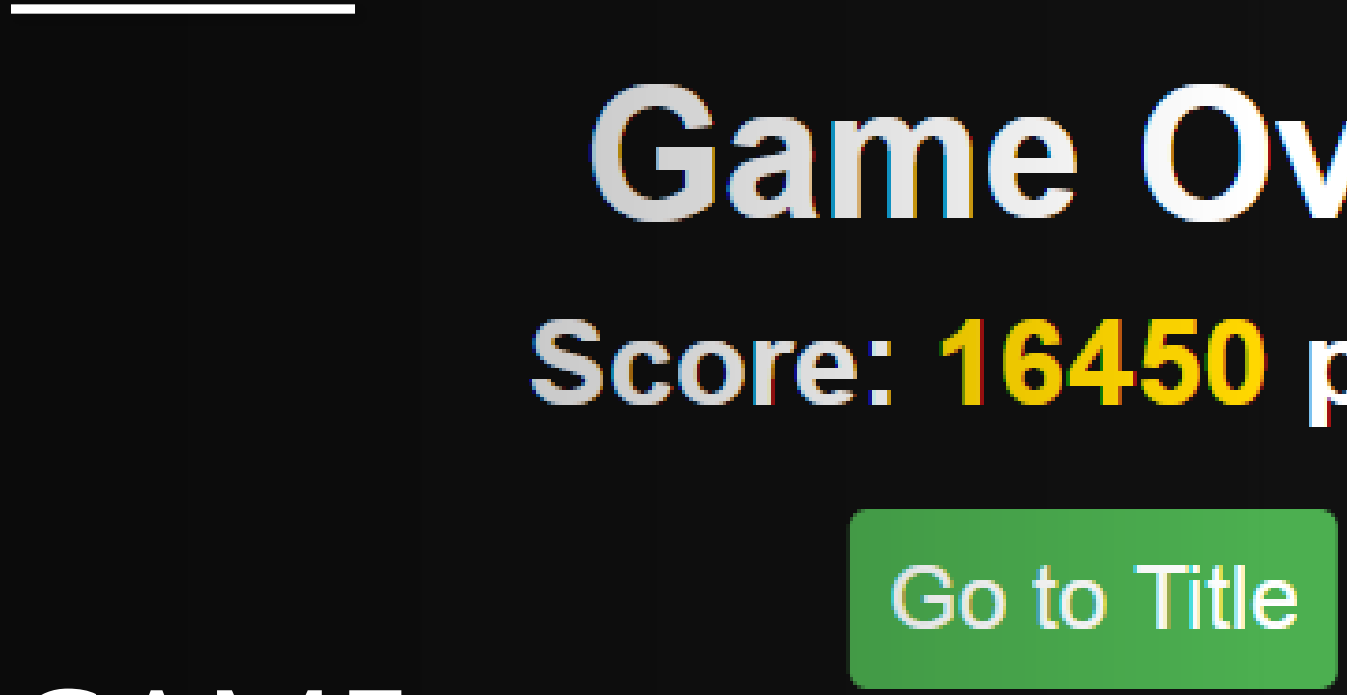
<게임 시작 버튼을 눌렸을때>



SCENE.HTML

Scene.html

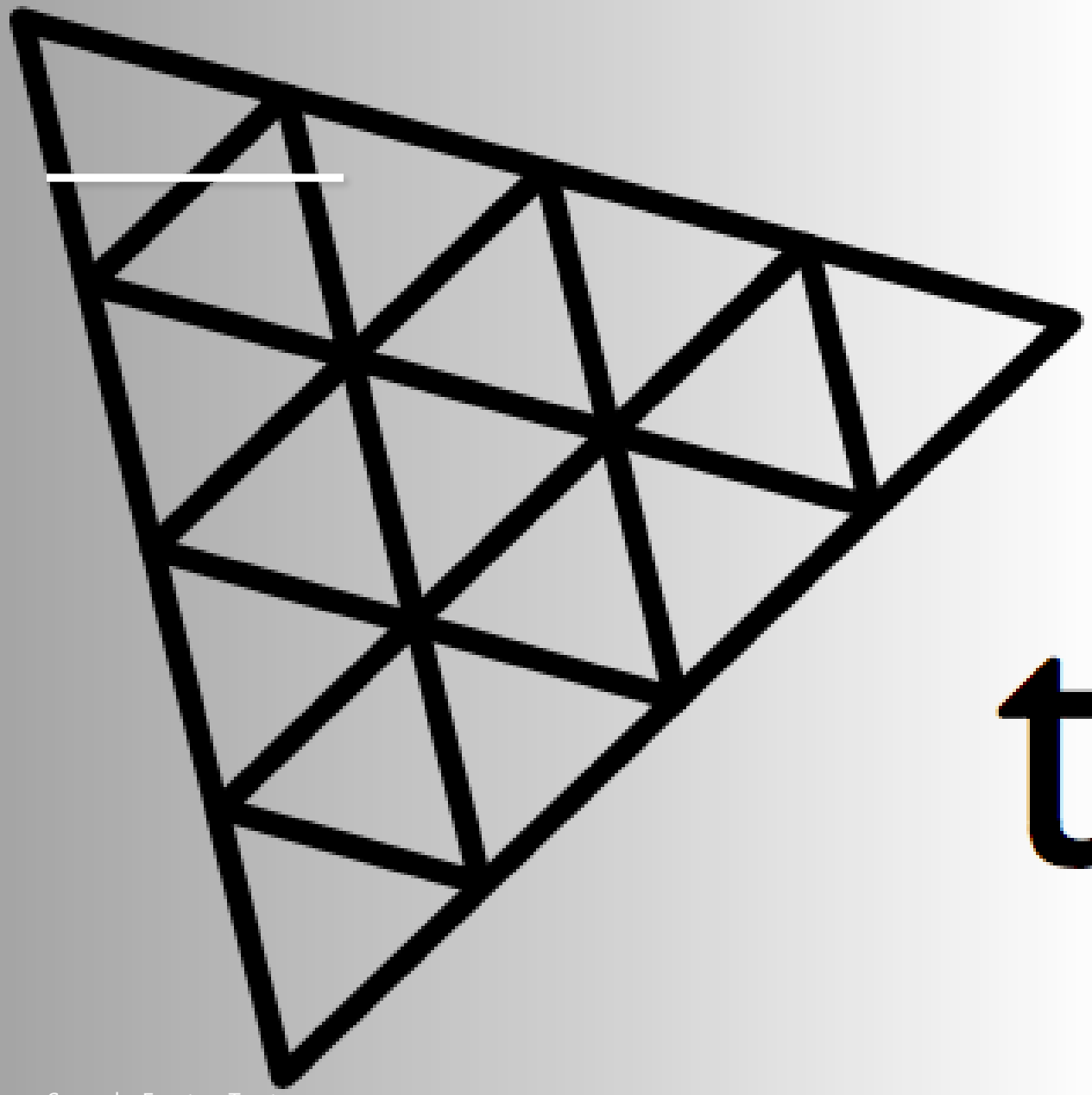
- HTML: <canvas> 요소를 포함하고 있어, 이 공간에서 게임이 진행됩니다.
- CSS: body의 margin을 0으로 설정하여, 페이지의 여백을 없애고 게임 화면을 최대한 크게 만듭니다. 또한, 게임이 진행되는 <canvas> 요소의 너비와 높이를 100%로 설정하여 화면 전체를 사용하도록 합니다.
- JavaScript: Three.js, GLTFLoader, Cannon.js 등의 라이브러리를 로드합니다. 이 라이브러리들을 통해 3D 그래픽스를 생성하고 물리 엔진을 적용합니다. 또한, 'objects.js'와 'soundManager.js', 'howler.core.min.js', 'main.js' 스크립트를 로드합니다.
- 난이도에 따라 적들이 생성되는 속도와 움직임이 다르고 1초마다 100점씩 점수가 오릅니다. 만약 땅에서 떨어지거나 적과 충돌하면 게임은 종료됩니다. 중간중간 생성되는 아이템을 먹으면 2000점의 추가 점수를 얻을 수 있습니다.



GAME OVER.HTML

Gameover.html

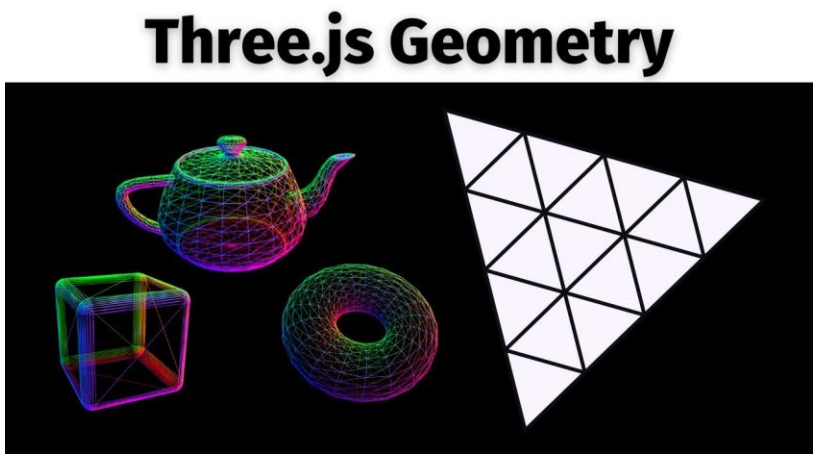
- Scene.html에서 마지막으로 죽었을때의 저장된 Score값을 받아와서 화면에 띄워줍니다. Go to Title 버튼을 누르면 Title.html로 이동합니다.



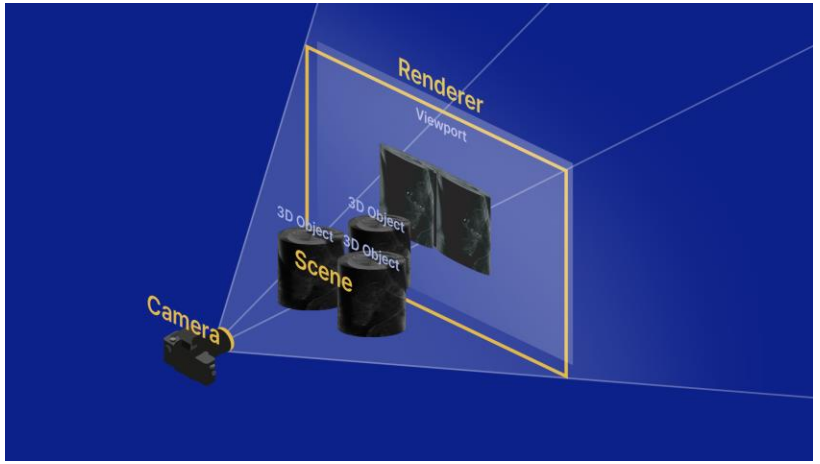
사용한
라이브러리 설명
three.js

Three.js

- **Three.js**는 웹에서 3D 그래픽스를 생성하고 표시하는 데 사용되는 JavaScript 라이브러리입니다. **WebGL(Web Graphics Library)**을 기반으로 작동하며, 복잡한 3D 그래픽스를 쉽게 만들 수 있도록 도와줍니다.
- **Three.js**를 사용하면 웹에서 실시간 3D 애니메이션, 인터랙티브 3D 그래픽스, 복잡한 3D 모델 등을 만들 수 있습니다. 또한, 다양한 형태의 3D 오브젝트, 카메라, 빛, 재질, 텍스처 등의 요소를 제어할 수 있습니다.
- **Three.js**는 개발자가 **WebGL**의 복잡한 세부 사항을 걱정하지 않고 3D 그래픽스를 만들 수 있게 해주므로, 3D 웹 애플리케이션 개발에 매우 유용합니다. 이 라이브러리를 사용하면 웹 브라우저에서 실행되는 복잡한 3D 그래픽스 애플리케이션을 빠르고 효율적으로 만들 수 있습니다.



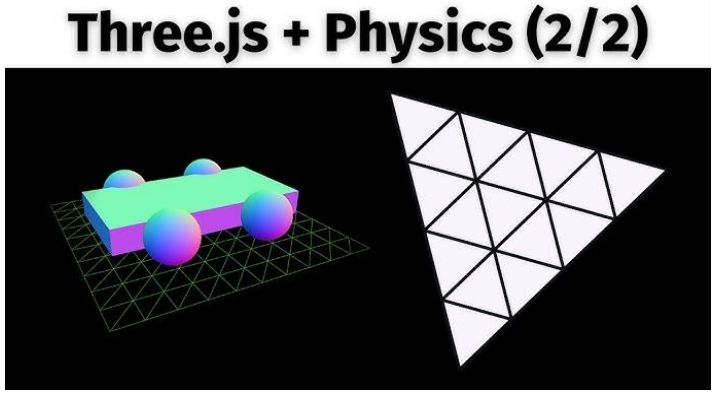
<각종 3D mesh>



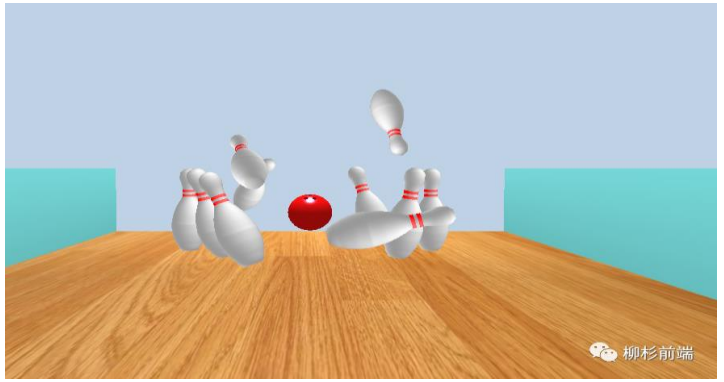
<기본 뼈대 구조>

Cannon.js

- **Cannon.js**는 웹에서 실시간 3D 물리 시뮬레이션을 가능하게 하는 JavaScript 라이브러리입니다.
- **강체 동역학**: 물리학에서 강체는 변형이나 회전이 없는 이상한 물체를 의미합니다. Cannon.js는 이러한 강체의 **움직임을 시뮬레이션**하는 것을 지원합니다.
- **충돌 검출**: 두 물체가 접촉했는지 아닌지를 판단하는 기능입니다. 이를 통해 물체 간의 상호작용을 구현할 수 있습니다.
- **다양한 형태의 충돌체**: Cannon.js는 구체, 상자, 원기둥 등 다양한 형태의 충돌체를 지원합니다. 이를 통해 다양한 모양의 물체를 물리 시뮬레이션에 포함시킬 수 있습니다.
- **제약 조건**: 두 물체를 서로 연결하거나, 특정 방향으로의 움직임을 제한하는 등의 제약 조건을 설정할 수 있습니다.
- 이러한 기능들을 통해 Cannon.js는 웹에서 복잡한 3D 물리 시뮬레이션을 구현하는 데 매우 유용한 도구입니다. Three.js와 같은 3D 그래픽 라이브러리와 함께 사용되어, 물리적 상호작용을 포함한 실시간 3D 애플리케이션을 만드는 데 널리 활용됩니다.



<렌더링 + 물리를 추가함>



<ex 볼링게임>

howler.js

- Howler.js는 웹에서 오디오를 효율적으로 처리하기 위한 JavaScript 라이브러리입니다.
- 오디오 재생: 웹에서 오디오 파일을 재생하는 기능을 제공합니다. 다양한 형식의 오디오 파일을 지원하며, 여러 사운드를 동시에 재생할 수 있습니다.
- 볼륨 제어: 오디오의 볼륨을 제어할 수 있습니다. 전체 볼륨 뿐만 아니라 개별 사운드의 볼륨도 조절할 수 있습니다.
- Spatial audio: 3D 위치 기반 오디오 효과를 제공합니다. 이를 통해 사용자가 사운드의 위치를 인지할 수 있게 해주며, 이는 게임과 같은 인터랙티브 애플리케이션에서 중요한 역할을 합니다.
- 여러 오디오 소스 관리: 여러 오디오 클립을 동시에 관리하고 제어할 수 있습니다. 예를 들어, 배경음악과 효과음을 동시에 재생하면서 별도로 제어할 수 있습니다.
- 크로스 브라우저 지원: 모든 현대적인 웹 브라우저를 지원하며, 브라우저마다 오디오 처리 방식의 차이를 최소화하여 일관된 오디오 경험을 제공합니다.

GLTFLoader.js

- GLTFLoader.js는 Three.js 라이브러리에서 제공하는 GLTF(GL Transmission Format) 형식의 3D 모델을 로드하는데 사용되는 로더입니다.
- GLTF는 효율적인 런타임 전송을 위해 만들어진 3D 모델 형식으로, 즉, 웹에서 3D 콘텐츠를 전송하고 로드하는 데 최적화되어 있습니다. 이 형식은 3D 모델뿐만 아니라 애니메이션, 스킨, 텍스처, 재질, 라이팅 등 3D 장면을 구성하는 거의 모든 요소를 포함할 수 있습니다.
- GLTFLoader.js를 사용하면 다음과 같은 과정으로 GLTF 형식의 3D 모델을 로드할 수 있습니다:
- GLTFLoader 인스턴스를 생성합니다.
- `load` 메소드를 호출하여 3D 모델 파일의 URL을 전달합니다.
- `load` 메소드의 두 번째 인자로 로드가 완료되었을 때 호출되는 콜백 함수를 전달합니다. 이 함수는 로드된 3D 모델을 인자로 받습니다.
- 콜백 함수 내에서 로드된 3D 모델을 장면에 추가합니다.
- 이러한 기능을 통해 GLTFLoader.js는 웹에서 3D 콘텐츠를 효율적으로 로드하고 표시하는 데 중요한 역할을 합니

```
sound.play();

//점수 변수
var score = 0;
//적의 HP 변수
var enemyHP = 3;


//난이도
const storedDifficulty = localStorage.getItem("difficulty");

// Cannon.js world
const world = new CANNON.World();
world.gravity.set(0, -9.8, 0); // 중력을 설정

// Three.js scene
const scene = new THREE.Scene();
scene.background = new THREE.Color(0x004fff); // Hex 코드로 하늘색을 나타내는 값

// Three.js renderer
const renderer = new THREE.WebGLRenderer({ canvas: document.getElementById("gameCanvas") });
renderer.setSize(window.innerWidth, window.innerHeight);
renderer.shadowMap.enabled = true; // 그림자를 만들도록 설정
document.body.appendChild(renderer.domElement);

//캔버스 위에 점수를 표시할 HTML 요소를 추가
const scoreElement = document.createElement("div");
scoreElement.style.position = "absolute";
scoreElement.style.fontWeight = "bold";
scoreElement.style.top = "20px";
scoreElement.style.left = "15px";
scoreElement.style.color = "black";
scoreElement.style.fontSize = "30px";
```

주요 함수 설명

주요함수 (Title.html)

- 이 HTML 파일은 게임 페이지를 구성하고 있으며, 주요 함수들에 대한 설명은 다음과 같습니다:
- **`startGame()`**: 이 함수는 게임 시작 버튼을 클릭했을 때 호출됩니다. 함수가 호출되면 '난이도 선택' 모달을 표시합니다.
- **`startGameWithDifficulty(difficulty)`**: 이 함수는 특정 난이도를 선택했을 때 호출됩니다. 함수가 호출되면 선택한 난이도를 localStorage에 저장하고, 해당 난이도를 쿼리 파라미터로 포함하여 Scene.html 페이지로 리다이렉션합니다.
- **`exitGame()`**: 이 함수는 게임 종료 버튼을 클릭했을 때 호출됩니다. 함수가 호출되면 현재 창을 닫습니다.
- **`showGameInstructions()`**: 이 함수는 게임 설명서 버튼을 클릭했을 때 호출됩니다. 함수가 호출되면 게임 설명서 모달을 표시합니다.
- **`closeModal()`**: 이 함수는 게임 설명서 모달의 '닫기' 버튼을 클릭했을 때 호출됩니다. 함수가 호출되면 게임 설명서 모달을 숨깁니다.
- 그리고 이 파일에는 3D 모델을 로드하고, 화면에 그리며, 애니메이션을 적용하는 등의 코드도 포함되어 있습니다. 이런 코드들은 Three.js와 GLTFLoader.js, Cannon.js 라이브러리를 사용하여 작성되었습니다.

주요함수 (objects.js)

- **이 코드는 여러 가지 객체를 생성하는 함수들을 정의하고 있습니다. 각 함수에 대한 설명은 다음과 같습니다:**
- **`createFloor(scene, width, height, x, y, z, color)`**: 주어진 크기, 위치, 재질로 바닥을 생성하는 함수입니다. 생성된 바닥 객체는 반환됩니다.
- **`createBall(scene, radius, x, y, z, color)`**: 주어진 반지름, 위치, 재질로 공을 생성하는 함수입니다. 생성된 공 객체는 반환됩니다.
- **`createBox(scene, width, height, depth, x, y, z, color)`**: 주어진 크기, 위치, 재질로 상자를 생성하는 함수입니다. 생성된 상자 객체는 반환됩니다.
- **`createCylinder(scene, radiusTop, radiusBottom, height, x, y, z, color)`**: 주어진 크기, 위치, 재질로 원통을 생성하는 함수입니다. 생성된 원통 객체는 반환됩니다.
- **`createLight(scene, color, x, y, z)`**: 주어진 색상과 위치로 빛을 생성하는 함수입니다. 생성된 빛 객체는 반환됩니다.
- **`createTexture(src)`**: 주어진 2D 텍스처 이미지의 경로로 텍스처를 생성하는 함수입니다. 생성된 텍스처 객체는 반환됩니다.
- **`createModel(scene, src, scale, rotation, restitution, friction, x, y, z, callback)`**: 주어진 속성으로 3D 모델을 로드하고, 모델이 로드되면 콜백 함수를 호출하는 함수입니다.
- **`createText(text, size)`**: 주어진 텍스트와 폰트 크기로 3D 텍스트를 생성하는 함수입니다. 생성된 텍스트 객체는 반환됩니다.

주요함수 (main.js) -1-

- 이 코드는 게임의 메인 로직을 담고 있습니다. 주요 함수들에 대한 설명은 다음과 같습니다:
- **`updateScore()`**: 점수를 1씩 증가시키고, 점수를 표시하는 HTML 요소를 갱신합니다.
- **`createEnemy()`**: 필드 내의 랜덤한 위치에 적을 생성합니다. 이 함수는 `setInterval`에 의해 주기적으로 호출됩니다.
- **`updateEnemies()`**: 모든 적들이 플레이어를 향해 이동하도록 합니다.
- **`checkCollisions()`**: 플레이어와 적 사이의 충돌을 감지합니다. 충돌이 감지되면 게임이 종료됩니다.
- **`createItem(x, y, z)`**: 지정된 위치에 아이템을 생성합니다. 이 아이템은 플레이어가 획득할 수 있습니다. 아이템의 회전 속성과 발광 색상, 그리고 깜빡거리는 효과도 저장합니다.
- **`spawnItem()`**: 필드 내의 랜덤한 위치에 아이템을 생성합니다. 이 함수는 `setInterval`에 의해 주기적으로 호출됩니다.
- **`updateItemsRotation()`**: 모든 아이템이 회전하도록 합니다. 이때 회전과 동시에 봘광하며 깜빡거리는는 blink() 함수 역시 호출하여 줍니다.

주요함수 (main.js) -2-

- **checkItemCollisions():** 플레이어와 아이템 사이의 충돌을 감지합니다. 이때 저는 아이템과 플레이어의 위치의 거리를 계산하여 충돌이 감지되면 점수가 증가하고, 해당 아이템이 제거하게 로직을 구성하였습니다.
- **gameOver():** 게임이 종료되었음을 알리고, Gameover.html 페이지로 이동합니다.
- **checkGameOver():** 플레이어가 일정 구간(y축값) 밖으로 떨어졌는지 확인합니다. 만약 그렇다면 게임이 종료됩니다. 이때 점수를 LocalStorage로 저장합니다.
- **animate():** 게임의 애니메이션 루프입니다. 이 함수는 `requestAnimationFrame`에 의해 주기적으로 호출되며, 게임 상태를 갱신하고 그래픽을 렌더링합니다. 60프레임 이므로 1초에 60번 업데이트 됩니다.
- **handleKeyboardInput():** 키보드 입력을 처리합니다. 이 함수는 애니메이션 루프 내에서 호출되며, 플레이어의 이동, 점프, 공격 등을 처리합니다.
- **attack():** 플레이어가 공격하는 함수입니다. 공격 버튼을 누르면 총알을 발사합니다. (제대로 구현 x)
- **updateBullets():** 발사된 총알들의 위치를 갱신합니다. 이 함수는 `setInterval`에 의해 주기적으로 호출됩니다.

주요함수 (SoundManager.js)

- 이 코드는 사운드 관리 클래스인 'SoundManager'를 정의하고 있습니다. 각 메서드는 다음과 같은 역할을 수행합니다:
- **`constructor()`**: SoundManager의 인스턴스를 초기화하는 생성자 메서드입니다.
- **`initAudioContext()`**: 오디오 컨텍스트를 초기화하는 메서드입니다. 오디오 컨텍스트가 아직 초기화되지 않았다면 새로운 오디오 컨텍스트를 생성합니다.
- **`loadSound(filePath, callback)`**: 파일 경로를 통해 오디오 파일을 로드하는 메서드입니다. 오디오 파일이 로드되면, 오디오 데이터를 디코딩하고, 이를 audioBuffer에 저장합니다. 오디오 파일 로드나 디코딩에서 오류가 발생하면 오류 메시지와 함께 콜백 함수를 호출합니다.
- **`play()`**: audioBuffer에 저장된 오디오를 재생하는 메서드입니다. audioBuffer가 존재하면, 오디오 컨텍스트에서 버퍼 소스를 생성하고, 이 소스를 오디오 컨텍스트의 출력 노드에 연결합니다. 이후 소스를 시작하여 오디오를 재생합니다.
- 마지막으로, SoundManager 클래스의 인스턴스를 생성하고, 이를 soundManager 변수에 할당합니다. 이 인스턴스는 이 파일을 import하는 다른 파일에서 사용할 수 있도록 export됩니다.

주요함수 (Gameover.html)

- 이 HTML 문서는 게임 오버 화면을 표시하고, 게임 오버 사운드를 재생하며, 게임 종료 시점에서 플레이어의 점수를 표시하는 역할을 합니다. 주요 함수는 다음과 같습니다:
- **`showGameOverScreen(score)`:** 게임 오버 화면을 표시하고, 플레이어의 점수를 표시하는 함수입니다. 'gameOverScreen' 요소의 display 스타일을 'block'으로 설정하여 표시하고, 'score' 요소의 textContent를 인자로 받은 점수로 설정합니다. 또한, 'gotoTitleButton' 요소에 클릭 이벤트 리스너를 등록하여, 클릭 시 'gotoTitle' 함수를 호출하도록 설정합니다.
- **`gotoTitle()`:** 타이틀 화면으로 이동하는 함수입니다. window.location.href를 'Title.html'로 설정하여 페이지를 이동시킵니다.
- **`handleGameOver()`:** 게임 오버 발생 시 호출되는 함수입니다. localStorage에서 'score'를 가져와서, 'showGameOverScreen' 함수에 전달하여 게임 오버 화면을 표시합니다. 만약 'score'를 찾을 수 없다면, 콘솔에 오류 메시지를 출력합니다.
- 이 스크립트는 localStorage에서 'score'를 가져와서 'handleGameOver' 함수를 호출하므로, 게임이 오버되었을 때 이 페이지로 이동하면, 자동으로 플레이어의 점수를 표시하는 게임 오버 화면이 표시됩니다.

그 외 요소들

- 3D 배경을 나타내는 SkyBox 추가
- 플레이어를 타겟으로 한 Perspective 카메라 추가
- Directinal Light 추가 및 그림자 추가
- 난이도 설정 및 Score Ui 추가

참고 사이트

- https://threejs.org/docs/ - Three.js Docs 이 문서에서 기본적인 Three.js 기능을 살펴 볼 수 있습니다.
- https://sketchfab.com/feed - 무료 3D 모델링 다운 사이트 입니다. 이곳에서 타이틀의 모델링을 가져왔습니다.
- https://www.youtube.com/@simondev758 - Three.js를 기반으로 게임 제작 강의를 해주시는 외국 유튜버 입니다. 이분의 강의를 듣고 어느정도 개념(카메라,라이팅,매쉬등)을 숙지하여 제작하였습니다.