Airplane Simulation Report

2021204045 이성민

전경 사진

Sky Class의 변경점

Sky Class는 하늘을 구현한다. 그러나 기존의 방식은 구름이 한 쪽에 몰려 있어 왼쪽 하늘에 위치하는 C, 오른쪽 하늘에 위치하는 c를 각각 생성하여 다른 값은 동일하게 설정하되 z 포지션만 다르게 설정하여 하늘에 Cloud가 고루 퍼질 수 있도록 한다.

```
c.mesh.position.z = Math.random()*700;
C.mesh.position.z = -Math.random()*700;
```

sky 사진

Sea Class와 moveWaves의 동작 방식

Sea Class는 바다를 구현하며, 바다의 파동을 표현하기 위해 moveWaves function을 사용한다.

주요 기능:

- **Sea Class**:
  - **생성자**: 바다를 나타내는 CylinderGeometry를 생성하고, 파도의 데이터를

저장하기 위해 각 정점(vertex)에 대해 랜덤 각도와 속도를 포함하는 waves 배열을 초기화.

- **Material**: MeshPhongMaterial을 사용해 바다의 재질을 정의하고, 반투명(.8)하게 설정.
- **Mesh**: 생성된 기하학과 재질을 사용해 Mesh를 만들고 그림자를 받을 수 있게 설정.

moveWaves :

- 파도의 위치를 업데이트합니다. 각 프레임마다 waves배열의 데이터를 기반으로 각 정점의 위치를 조정.
- 정점 위치를 조정한 후 positionBuffer.needsUpdate를 true로 설정하여 변경사항이 renderer에 반영.
- 바다 메시의 회전을 조정하여 바다 전체가 회전.

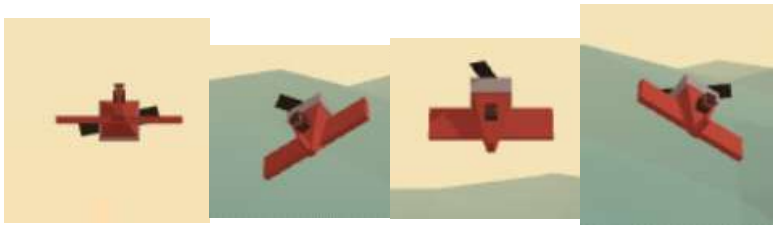
sea 사진

airplane 동작 - 'W', 'A', 'S', 'D' 키 사용

'W', 'A', 'S', 'D' 키를 이용해 airplane을 조작하는 방식으로 구현되어 있다. 주요 로직은 updatePlane function에 포함되어 있다.

주요 기능:

- **키 입력 추적**: keyboard 객체를 사용해 키 입력 상태를 추적. onKeyDownonKeyUp이벤트 리스너가 이를 담당.
- **airplane 위치 및 회전 조정**:
  - 'W' 키: airplane이 위로 이동. Umax = 500으로 Y값의 최댓값을 조절.
  - 'A' 키: airplane이 왼쪽으로 이동. Lmax = -Rmax = -300으로 Z값의 최솟값을 조절.
  - 'S' 키: airplane이 아래로 이동. Dmax = 70으로 Y값의 최솟값을 조절.
  - 'D' 키: airplane이 오른쪽으로 이동. Rmax = 300으로 Z값의 최댓값을 조절.
  - 전체적으로 moveDistance = 30으로 이동속도를 조절.
- **이동 및 회전 구현**:
  - airplane의 위치와 회전을 부드럽게 조정하기 위해 targetYtargetZ값을 사용하고 easing값을 적용.
  - airplane의 mesh의 pilot을 추가하여 pilot 또한 airplane과 같이 자연스러운 회전.(airplane.mesh.add(pilot.mesh)를 이용);
  - Quaternion을 사용해 airplane의 자연스러운 회전을 구현하며, 키 입력에 따라 적절한 축(axis)으로 회전.
  - 'W', 'S'키는 Z축, 'A', 'D'키는 X축, 또한 버튼을 누르지 않았을 때 각도가 0으로 서서히 회전(airplane.mesh.quaternion.slerp를 이용)



각각 'W', 'A', 'S', 'D'키를 눌렀을 때 airplane이 이동하면서 자연스럽게 회전

카메라 동작 - 'o' (FPS), 'p' (Topview) 키 사용

카메라 시점은 'O'와 'P' 키를 통해 전환됩니다. 각 시점 전환은 updatePlane함수에 정의된 로직에 따라 구현된다.

주요 기능:

- 'O' 또는 'P'키를 눌렀을 때, 현재 camera의 position(startPosition), 현재 카메라의 lookAt(startLookAt)에서 endPosition, endLookAt으로 서서히 변경.
- **'O' 키**:

  - Top-View 모드에서, FPS (First-Person Shooter) 모드로 자연스러운 전환
  - 보이지 않고, 회전하지 않는 sphere가 airplane과 같은 위치에 위치.
  - camera가 보이지 않는 sphere를 추적 -> airplane을 추적하는 효과.
  - cameraType이 1에서 0으로 전환.

  - 해당 카메라 시점 전환은 (sphere.mesh.add(camera);)를 실행한 뒤에 실행.
  - 이에 따라 자연스러운 카메라 시점 전환을 위해 airplane이 어디에 위치하든 Top-View 모드의 카메라 좌표인 (-100, 750, 0)를 startPosition에 입력 필요.
  - 또한 카메라 시점 전환 종료 시 lookAt이 airplane에 위치해야 함.
  - startPosition.set(**-100, 750 - airplane.mesh.position.y, -airplane.mesh.position.z**) -> endPosition.set(**-300, 200, 0**);
  - startLookAt.set(**0, 100, 0**); -> endLookAt.set(**0, airplane.mesh.position.y, airplane.mesh.position.z**);

  - **문제점:** 시점 전환 직전의 airplane의 위치를 저장하기에 시점 전환 동안에 airplane이 움직이면 그 움직인 만큼의 position만큼의 lookAt 오류 발생.
  - **해결방법:** camera 시점 전환 동안의 움직임만큼 lookAt을 변경.

```
if (cameraType === 0) {
    if (keyboard[87] && airplane.mesh.position.y < Umax) { // 'W'키
        endLookAt.y += sphereMoveDistance;
    } ...  //다른 키도 같은 방법으로 설정('A', 'D'키는 position.z 사용)
}
```

- **해결방법2:** 혹시 예상치 못한 오류로 카메라 전환을 끝마쳤는데도 카메라가 airplane의 정가운데를 향하지 않을 시, 카메라 전환이 끝나는 60프레임 뒤 camera의 lookAt을 airplane을 향하게 다시 변경.

```
executeAfterFrames(() => {
    camera.lookAt(0, airplane.mesh.position.y, airplane.mesh.position.z);
}, 60);
```

- **'P' 키**:

  - FPS (First-Person Shooter) 모드에서, Top-View 모드로 자연스러운 전환
  - 카메라가 위에서 아래로 비행기를 내려다보는 시점으로 이동
  - camera sphere 추적 해제, 고정 시점으로 전환
  - 카메라 타입이 0에서 1로 전환.

  - 해당 카메라 시점 전환은 (sphere.mesh.remove(camera);)를 실행한 뒤에 실행.
  - 이에 따라 자연스러운 카메라 시점 전환을 위해 airplane이 어디에 위치하든 airplane을 추적하는 camera의 좌표, lookAt을 startPosition,startLookAt에 입력 필요.
  - startPosition.set(**-300, 200 + airplane.mesh.position.y, airplane.mesh.position.z**) -> endPosition.set(**-100, 750, 0**);
  - startLookAt.set(**0, airplane.mesh.position.y, airplane.mesh.position.z**); -> endLookAt.set(**0, 100, 0**);

요약

이 코드에서는 'W', 'A', 'S', 'D' 키를 사용해 비행기를 조작하고, 'O'와 'P'키를 통해 카메라 시점을 전환한다. Sea Class는 파도를 시뮬레이션하며, moveWave function을 통해 파도의 움직임을 구현하며, 기존의 Sky Class를 수정하여 하늘에 고루 cloud를 생성한다.