* **[조작법]**

|  |  |
| --- | --- |
| **키** | **설명** |
| **W** | **앞으로 이동** |
| **A** | **왼쪽으로 이동** |
| **S** | **뒤로 이동** |
| **D** | **오른쪽으로 이동** |
| **F2 / F3** | **자유 시점(플레이어를 그리지 않는다) / 플레이어 시점** |
| **M** | **해가 떠오르게 한다.** |
| **N** | **달이 떠오르게 한다.** |
| **P** | **플레이어의 위치를 미로의 입구로 이동한다.** |
| **F** | **손전등을 켠다** |
| **G** | **손전등을 끈다.** |
| **ESC** | **프로그램 종료** |

* **[사용한 자료구조]**

**.......**

CScene

**ObjectTag::**

**Background**

**ObjectTag::**

**Objects**

**ObjectTag::**

**Bullet**

**ObjectTag::**

**Particle**

**m\_pShaders**

**ObjectTag::**

**Count**

**ObjectTag::**

**Bullet**

**ObjectTag::**

**Objects**

**ObjectTag::**

**Background**

**m\_pppObjects**

**ObjectTag::**

**Particle**

**m\_npObjects[ObjectTag]**

**CGameObject\*: -**

**.......**

**CRotatingObject\*: 50**

**.......**

**CRotatingObject\*: 50**

**.......**

**CRotatingObject\*: 500**

**.......**

* 사용한 자료구조는 3D Game Programming 1의 3번 과제와 같다. 오브젝트의 3차원 포인터 배열에 세로축엔 오브젝트의 종류, 가로축엔 같은 오브젝트의 개수를 담아서 같은 종류의 오브젝트들마다 하나의 인스턴싱 셰이더를 통해 animate 및 render연산을 수행했다.
* **[프로그램 요구사항]**

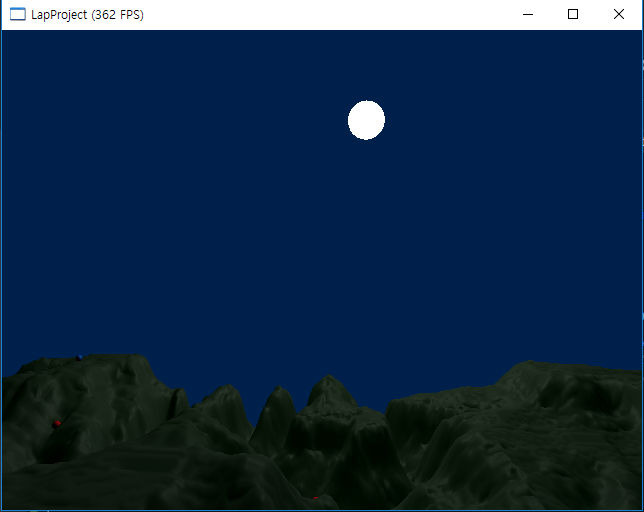
1. **태양과 달이 있고, 지형 중심을 기준으로 공전하면서 가상공간에 낮과 밤을 만든다.**

* 가정

1. 태양과 달은 구로 표현한다.
2. 태양과 directional light, 달은 point light로 표현한다.
3. 태양과 달은 180도 반대편에 위치한다.

* 구현

1. 태양과 달은 샘플 코드에 선언 되어있던 CRevolvingObject를 사용했으며 mesh로는 CSphereMeshIlluminated를 사용했다.
2. Material의 Emissive를 설정하여 구가 빛이 없어도 밝아 보이게 하였다.
3. 태양과 달의 공전은 CRevolvingObject의 Animate에서 처리하였다. 월드 matrix의 41, 42, 43부분에 terrain의 중심 위치가 (800, 0, 800)이므로 (1200\*cos + 800, 1200\*sin, 800)으로 설정하고 cos, sin의 세타값을 매 프레임마다 fTimeElapsed를 누적시켜 CRevolvingObject가 공전하게 만들었다.
4. 태양과 달의 cos, sin 세타의 초기값을 각각 0, 3.14로 설정하여 태양과 달이 180도 반대편에 위치하게 하였다.
5. m\_pLights의 1번 2번을 각각 directional light, point light로 설정하고 scene의 animate에서 매 프레임마다 position을 각각 CRevolvingObject 의 position으로 갱신하면서 directional light의 direction을 normalize((800, 0, 800) – position)로 구한 벡터로 설정하여 빛의 방향이 지형의 중심으로 가도록 하였다.



1. 태양과 달의 y값이 0보다 작으면 비활성화하고 0보다 크면 활성화하여 낮과 밤을 나타냈다.

달

태양

1. **지형과 오브젝트들은 태양과 달빛을 받아서 빛 반사를 한다.**

* 가정

1. 플레이어를 제외한 오브젝트들의 메시를 모두 illuminated로 만든다.
2. 빛 계산은 cube mesh의 정점이 러프하므로 hlsl에서 픽셀라이팅으로 계산한다.
3. 샘플 코드에 나머지는 다 구현되어 있으므로 그대로 사용한다.

* 구현

1. 루트 시그니처에 D3D12\_ROOT\_PARAMETER\_TYPE\_SRV 의 ShaderVisibility를 D3D12\_SHADER\_VISIBILITY\_ALL로 변경하고 shader.hlsl의 #define \_WITH\_VERTEX\_LIGHTING를 주석처리하여 정점라이팅에서 픽셀라이팅으로 변경한다.
2. Cube와 Sphere의 mesh는 샘플 코드에 이미 illuminated가 구현 되어있으므로 기존의 diffused mesh를 이용하는 cube와 sphere 오브젝트들을 illuminated cube, illuminated, sphere로 변경한다.
3. terrain mesh를 생성할 때 terrain image의 GetHeightMapNormal함수를 이용하여 기존의 diffused vertex를 illuminated vertex로 변경한다.
4. **키 조작으로 낮과 밤 상태를 조절할 수 있다.**

* 가정

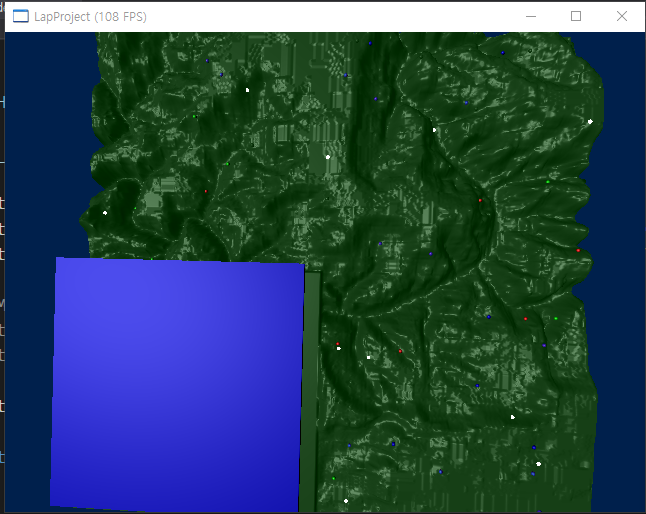
1. M 키를 누르면 낮이 된다.
2. N 키를 누르면 밤이 된다.

* 구현

1. CRevolvingObject의 누적되고있는 세타를 M을 누르면 태양은 0.0, 달은 3.14로 설정한다.
2. CRevolvingObject의 누적되고있는 세타를 N을 누르면 태양은 3.14, 달은 0.0로 설정한다.

낮

밤



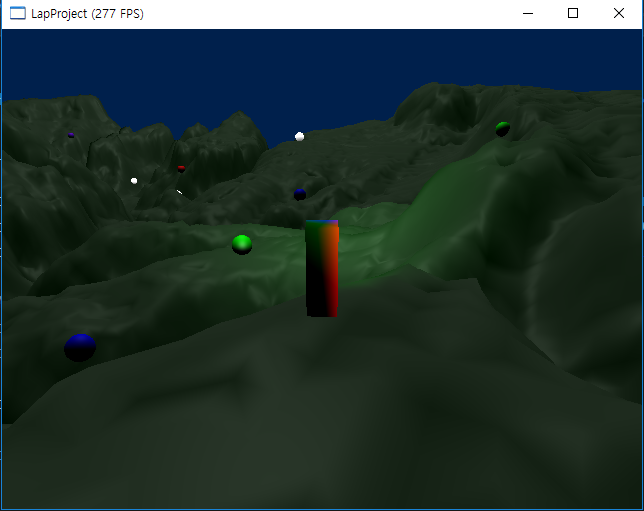
1. **지형 위에 50개 이상의 구들이 존재하고 각각의 구는 서로 다른 방향으로 이동하며 구끼리 충돌하면 튕겨나간다.**

* 가정

1. 3D Game Programming 1의 3번 과제의 구들을 기반으로 한다.
2. 미로 안에서 플레이어를 쫓아다니던 구들을 지형의 랜덤한 위치에서 생성되게 하고 플레이어를 쫓지 않게 한다.
3. 구들의 초기 이동 벡터는 랜덤이다.
4. 구들은 지형 밖으로 나가지 않고 경계에 도달하면 이동 벡터가 반사 방향으로 변경된다.
5. 구들은 실내 지형이 있는 곳에 들어갈 수 없고 경계에 도착하면 이동 벡터가 반사된다.
6. 구 끼리 충돌하면 방향 벡터가 충돌한 구와 반대 방향으로 변경된다.

* 구현

1. 씬의 AnimateObjects함수에서 구가 플레이어를 쫓게 만드는 부분과 실내 지형의 벽과 충돌하는 부분을 삭제한다.
2. PhysicsProcessing함수에서 충돌 후 후처리부분에 setdirection (normalize(object->getposition() – collider->getposition())) 또는 그 반대인 colliderPos – objectPos 를 추가하여 구끼리 충돌했을 때 충돌한 구의 반대 방향으로 이동 방향이 바뀌도록 한다.
3. 구가 지형의 좌우 경계에 도달하면 이동 벡터의 x를 –x로 반전시키고 상하 경계에 도달하면 이동 벡터의 z를 –z로 반전시킨다.
4. 구가 실내 지형이 존재하는 지형의 1/4 크기의 구역 경계의 위쪽에 도달하면 이동 벡터의 z를 –z로 반전시키고 오른쪽에 도달하면 이동 벡터의 x를 –x로 반전시켰다.
5. 구가 생성되는 위치는 x: 800~1500, z: 800~1500으로 설정하였고 이동 벡터도 xyz를 모두 -1000~1000의 랜덤한 정수를 넣고 normalize하여 설정하였다.
6. 구의 이동 벡터가 설정될 때 마다 구의 회전축을 이동 벡터와 (0,1,0)의 외적 벡터로 설정하여 구가 이동할 때 이동하는 방향으로 구르게 하였다.



굴러다니는 구들

1. **지형 어느 위치에 실내 지형이 있고 실내 지형에 20개 이상의 조명이 있으며 플레이어가 실내에 들어가면 조명이 활성화되고 실내에는 태양과 달이 영향을 줄 수 없다.**

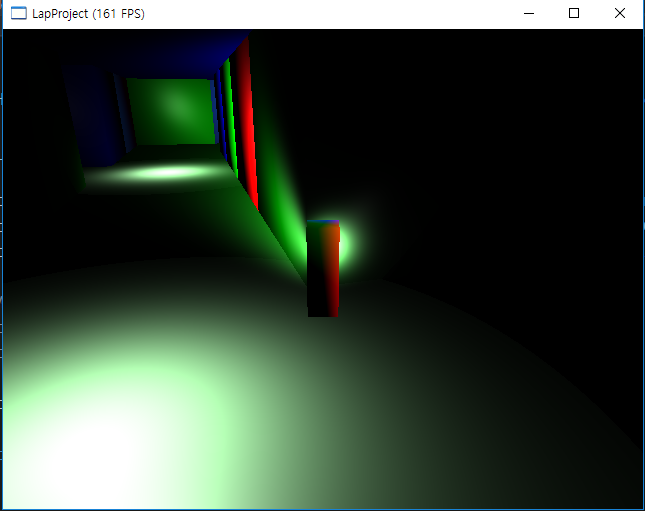
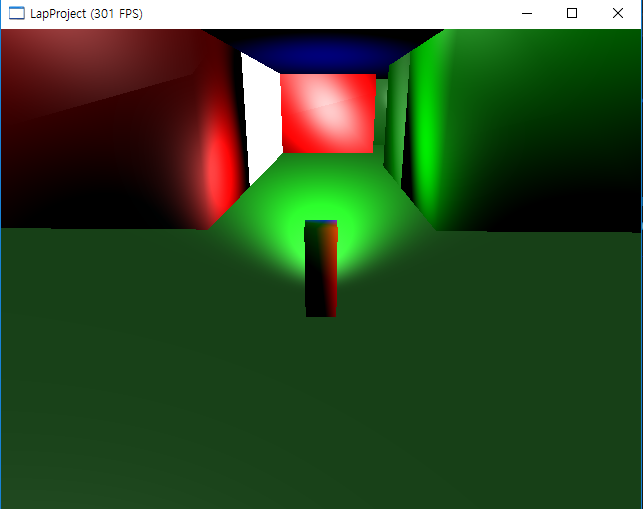
* 가정

1. 실내 지형은3D Game Programming 1의 3번 과제의 미로를 기반으로 한다.
2. 태양, 달, 플레이어 손전등을 제외한 실내 지형의 조명은 27개이며 미로의 pathes 위치에 일정 간격으로 존재한다.
3. 플레이어가 실내 지형으로 진입하면 27개의 실내 조명이 활성화되고 태양과 달의 light가 비활성화 된다.
4. 실내 조명은 spot light만 사용한다.

* 구현

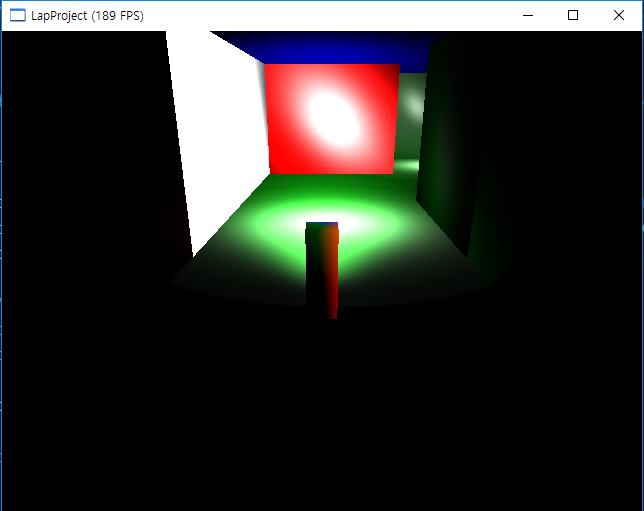
1. 미로 지붕의 크기가 지형의 1/4이 되도록 terrain의 scale을 (6.5, 2, 6.5) 로 조정하고 terrain의 1/4의 높이 값을 100으로 고정하여 미로가 위치한 지형을 약간 높게 평탄화 시켰다.
2. 미로의 길 개수를 실내조명 개수로 나눈 값을 stride로 두어 미로의 길 내부에서 일정간격마다 조명이 위치할 수 있도록 설정하였다.
3. 미로 지붕 크기를 기준으로 지붕의 x, z범위 안에 플레이어가 들어오고 지붕의 y보다 플레이어가 아래에 있다면 플레이어가 실내에 들어왔다고 판단을 하여 실내 조명을 모두 활성화하고 태양과 달을 비활성화 하였다.
4. 실내 조명은 미로의 길 천장 중심에 위치시켰고 0, -1, 0 방향으로 빛을 쏘도록 설정하였다.

실내 조명



실내 진입 후

실내 진입 전



1. **플레이어는 손전등을 가지고 있으며 손전등은 키고 끄고 할 수 있다.**

* 가정

1. 샘플 코드에 있는 손전등을 그대로 사용하였다.
2. F키를 누르면 손전등이 활성화된다.
3. G키를 누르면 손전등이 비활성화 된다.

* 구현

1. F키와 G키로 손전등인 light[0]의 enable플래그를 토글하도록 하였다.
2. 손전등의 범위를 300으로 늘리고 m\_fPhi를 70, m\_fTheta를 35로 조정하여 손전등의 밝기와 퍼지는 정도를 더 크게 하였다.

손전등 끔

손전등 켬

