3D 게임프로그래밍

과제02 설명 문서

〈과제02 테셀레이션, 파티클 이펙트, 평면 거울〉

제출일: 2021. 11. 29

2018182021 윤성주

목차

1. 프로젝트 소개

2. 프로젝트 목표

3. 구현 내용 설명 및 가정

4. 코드 설명

5. 사용한 자료구조와 알고리즘, 디자인 패턴

6. 프로젝트의 매커니즘

7. 조작법

8. 프로젝트 플레이 화면, 지형 전경

1. 프로젝트 소개

지형을 동적 테셀레이션을 사용해 구현하고, 빌보드, 블렌딩된 눈(snow) 파티클을 기하쉐이더를 통해 생성한다. 지형 텍스처를 눈을 사용해 눈이 쌓여 있는 느낌을 주고, 실내환경 안에는 평면거울을 구현하였다.

2. 프로젝트 목표

① 지형을 동적 테셀레이션을 사용하여 렌더링한다. 'D'를 누르면 테셀레이션이 보인다.

② 기하쉐이더를 이용하여 빌보드, 블렌딩된 파티클을 1만개 생성한다.

③ 실내 환경을 오브젝트의 은면제거를 통해 구현하고, 'I'를 누르면 실내로 이동한다.

④ 실내에는 오브젝트 1개와 평면거울을 설치하여 평면거울에 오브젝트가 반사되어 보이도록 한다.

⑤ 이 프로젝트를 통해 지형의 동적 테셀레이션의 과정을 이해한다.

⑥ 이 프로젝트를 통해 쉐이더로 연결, 쉐이더 코드의 사용 등을 익힌다. 또한 기하쉐이더를 구현해봄으로써 기하쉐이더의 실행과정을 이해한다.

3. 구현 내용 설명 및 가정

◆ **가정**

✓ 플레이어는 모든 오브젝트와 충돌하지 않는다.

✓ 지형을 타는 플레이어는 지형을 벗어나지 않는다. (플레이어가 실내환경으로 이동할 때는 벗어난다.)

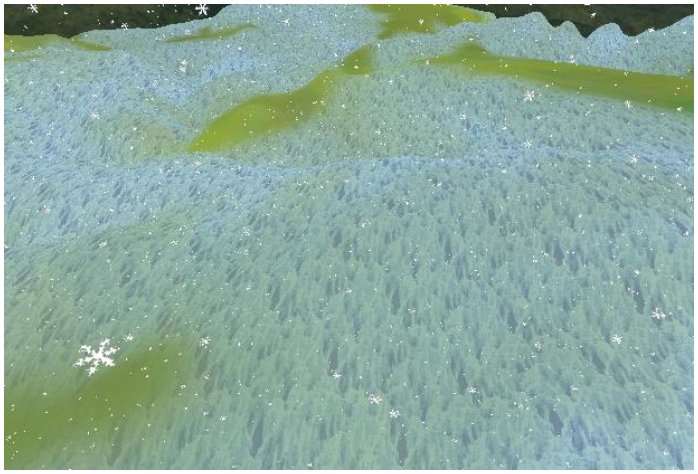
✓ 파티클은 계속해서 아래로 떨어지며 파티클은 (0, y, 0) -〉 (1000, y, 1000) 구역에서만 렌더된다.

✓ 평면거울에는 상자하나만 반사되고, 흑백으로 반사된다.

◆ 구현 내용 및 기능 설명

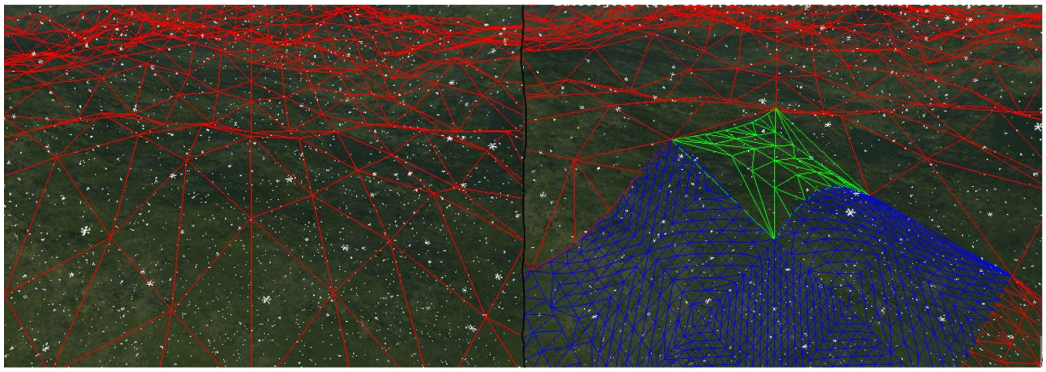
■ 지형

① 테셀레이션 터레인



(텍스처링된 지형)

✓ 디테일 텍스처링으로 통해 눈이 쌓인 느낌을 주었다.

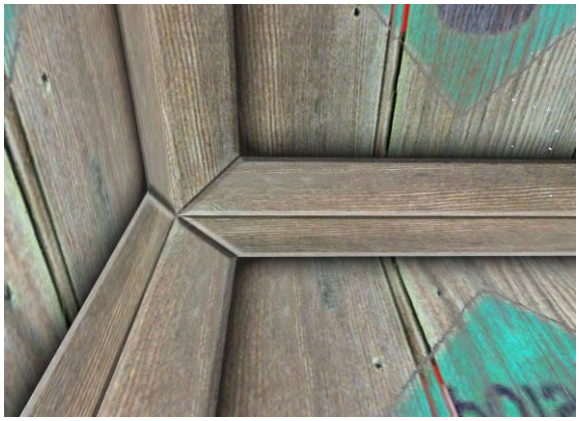


(동적 테셀레이션된 지형 (좌-카메라가 멀 때, 우- 카메라가 가까워질때))

✓ 카메라로부터의 거리에 따라 정점의 개수를 조절하는 동적 테셀레이션을 구현하였다. 왼쪽은 카메라를 멀리 두었을 때 정점의 개수가 적은 상태이며, 왼쪽은 카메라를 가까이 두었을 때 카메라에서 가까운 부분은 정점의 개수가 늘어난 상태이다.

✓ ‘D’를 누르면 동적 테셀레이션을 볼 수 있다.

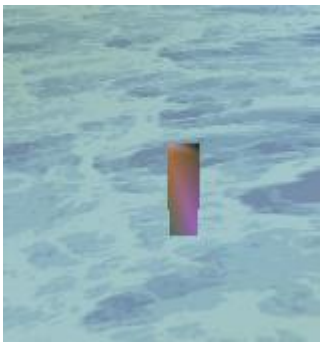
② 실내 지형



✓ 상자 오브젝트를 렌더링할 때 은면을 FRONT로 하여 내부에서 텍스처가 보이도록 했다. 플레이어가 상자 안에 들어가 있으면 실내에 있는 느낌이 든다.

✓ 'I'를 누르면 플레이어는 실내로 이동한다.

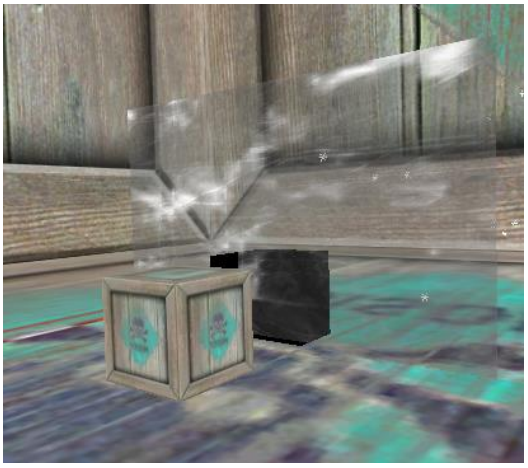
■ **플레이어**



✓ 지형을 타도록 함 (테셀레이션된 지형을 정확하게 타지 못함.)

✓ 마우스로 시점 변경 가능하고 상하좌우 키로 카메라 시점에 따라 이동한다.

■ **평면거울**



- ✓ 상자 오브젝트 1개가 평면거울에 반사된다.
- ✓ 평면거울은 흑백으로 렌더링 된다. (버그)

■ **파티클 이펙트**

- ✓ 기하쉐이더를 이용하여 파티클 하나 렌더링할 때 1개의 정점을 가지고 렌더하였으며, 1만개의 눈 파티클을 렌더링하였다.
- ✓ 파티클은 y축 방향으로 내려온다.
- ✓ 파티클은 (x, z)좌표 (0, 0) ~ (1000, 1000) 구간에만 생성된다.

4. 코드 설명

프로젝트는 샘플 프로젝트 중 LabProject09-1-0(Terrain Tessellation)을 기반으로 만들었으며 프로젝트 중 프로젝트에 이미 있던 터레인, 플레이어를 제외한 모든 오브젝트는 직접 추가하였다.

〈Player, DDSTextureLoader12, Camera, Timer, UILayer, Object 소스 및 헤더〉

① 기존 샘플 프로젝트에서 따로 수정하지 않음.

〈GameFramework 소스코드 및 헤더파일〉

① 실내 지형 이동 입력받기

- ✓ 'I' 누르면 플레이어의 위치를 실내지형의 위치로 이동.

〈Mesh 소스코드 및 헤더파일〉

① CSnowVertex 클래스 추가 (부모 클래스: CVertex)

- ✓ 눈 파티클 생성 시 사용하는 클래스. 멤버로 크기를 갖는다. CVertex에 이미 위치는 있으므로 따로 만들지 않음. 이 클래스의 변수가 쉐이더로 연결된다.

② CSnowBillboardMesh 클래스 추가 (부모 클래스: CMesh)

- ✓ 눈 파티클 메쉬. 생성 시 생성되는 개수만큼의 CSnowVertex 객체를 만들고, 이 객체를 CreateBufferResource()을 통해 쉐이더로 연결한다.
- ✓ 기하쉐이더로 정점을 늘리므로 눈 파티클 하나 당 하나의 정점만 넘겨준다.

③ CB_SNOW_INFO 구조체 추가

- ✓ 업로드 힙에 연결한 구조체. 눈 파티클의 위치와 사이즈를 담는다.

〈Scene 소스코드 및 헤더파일〉

① BuildObjects() 함수

- ✓ HeightMapTerrain, SkyBox, 3개의 쉐이더를 생성한다.
- ✓ 3개의 쉐이더는 객체를 생성하는 ObjectsShader, 눈 파티클을 생성하는 SnowBillboardObjectsShader, 평면거울을 생성하는 MirrorShader이다.
- ✓ MirrorShader의 경우 현재씬 객체와 ObjectsShader 객체를 저장하도록 Set 해준다.

② CreateGraphicsRootSignature()함수

- ✓ DescriptorRanges에 눈 파티클 텍스처와 거울 텍스처를 설정해준다. 각각 쉐이더에서 gSnowTextureArray(t4~t8), gtxtMirror(t9) 변수와 연결된다 각각 DescriptorRanges의 5, 6번 인덱스에 저장된다.
- ✓ CB_SNOW_INFO 구조체를 쉐이더에 연결하기 위해 CBV를 생성한다. 6번 루트 파라미터에 연결한다.
- ✓ 위에서 DescriptorRanges 5, 6번 인덱스에 저장된 디스크립터 테이블을 각각 11, 12 번 루트 파라미터에 연결한다.

〈Shader 소스코드 및 헤더파일〉

① CObjectsShader 클래스

- ✓ BuildObjects(): 거울에 반사할 객체하나, 실내 지형 객체 하나 총 2개의 오브젝트를 만든다. 두 객체의 텍스처는 같고, 메쉬도 큐브텍스처 메쉬를 사용한다. 이때 생성한 텍스터는 7번 루트 파라미터에 연결한다.
- ✓ CreateShader(): 생성된 두 개의 객체는 각각 다른 파이프라인상태를 가지므로 2개의 파이프라인 상태를 만들고 하나는 기존과 동일(0번 인덱스), 하나는 레스터라이저에서 은면제거를 FRONT로 하도록 설정한다.(1번 인덱스) (실내 지형은 안쪽 면에 텍스처가 보여야 하므로)
- ✓ Render(): 거울에 반사할 객체는 0번 인덱스의 파이프라인상태를 사용하고, 실내 지형은 1번 인덱스의 파이프라인상태를 사용한다.

② CMirrorShader 클래스

- ✓ 스텐실 버퍼에 그릴 거울, 반사된 객체, 거울 뒷면, 거울 총 4개의 파이프라인 상태를 설정한다.
- ✓ Build Objects(): 거울 객체를 텍스처링된 사각형 메쉬로 만든다.
- ✓ Render(): 거울 뒷면 렌더링 -〉 거울 스텐실 버퍼에 렌더링 -〉 반사된 객체 렌더링 -〉 거울 렌더링.
  이때 반사 평면은 (0, 0, -1)이다. 거울 뒷면을 렌더링하고 깊이 스텐실 초기화 하여 거울 뒷면이 보이지 않도록 한다. (버그 있음)
- ✓ CreatePixelShader_Mirror(): 거울 텍스처를 사용하는 거울 픽셀쉐이더를 연결한다.

③ CSnowBillboardObjectsShader 클래스

- ✓ BuildObjects(): CSnowBillboardMesh로 객체를 생성하고, 이때 11번 루트 파라미터에 눈 파티클 텍스처 5개를 연결한다.
- ✓ CreateInputLayout(): CB_SNOW_INFO 구조체의 멤버의 위치, 크기를 InputLayout으로 설정한다.
- ✓ CreateShaderVariables(),UpdateShaderVariables():
  m_pd3dcbSnowBillboard을 업로드힙에 연결하고, 갱신한다.
- ✓ CreateBlendState(): 텍스처가 색상이 없는 부분은 블렌딩되도록 상태를 설정한다.
- ✓ CreateGeometryShader(): 기하 쉐이더 연결한다.

〈Shaders.hlsl〉

① cbuffer cbSnowInfo: 눈파티클의 위치와 사이즈를 담음

② gSnowTextureArray: 눈 파티클의 텍스처들을 담음

③ gtxtMirror: 거울의 텍스처를 담음

④ PSMirror: 거울 객체의 픽셀 쉐이더.

⑤ VS_SnowBillboard: 눈 파티클 정점 쉐이더

⑥ GS_SnowBillboard: 눈 파티클 기하 쉐이더. 하나의 정점을 4개로 늘린다.

⑦ PS_SnowBillboard: 눈 파티클 픽셀 쉐이더. 조명을 적용한다.

5. 사용한 자료구조와 알고리즘, 디자인 패턴

① 배열로 객체 관리

6. 프로젝트의 매커니즘

① 플레이어가 지형 중간에 생성된다.

② 눈 파티클이 아래로 떨어진다.

③ 'I'를 누르면 실내 지형으로 이동한다.

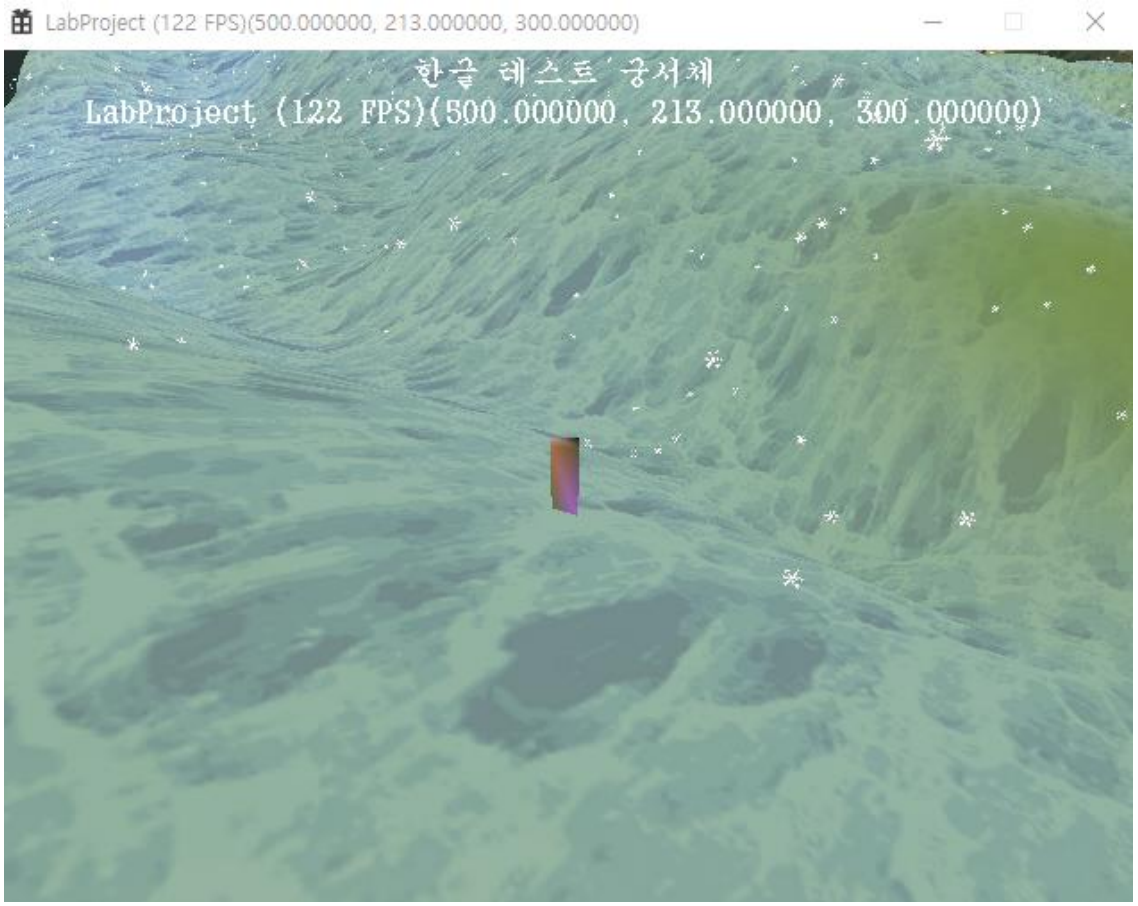
④ 거울에 상자 객체가 반사된다.

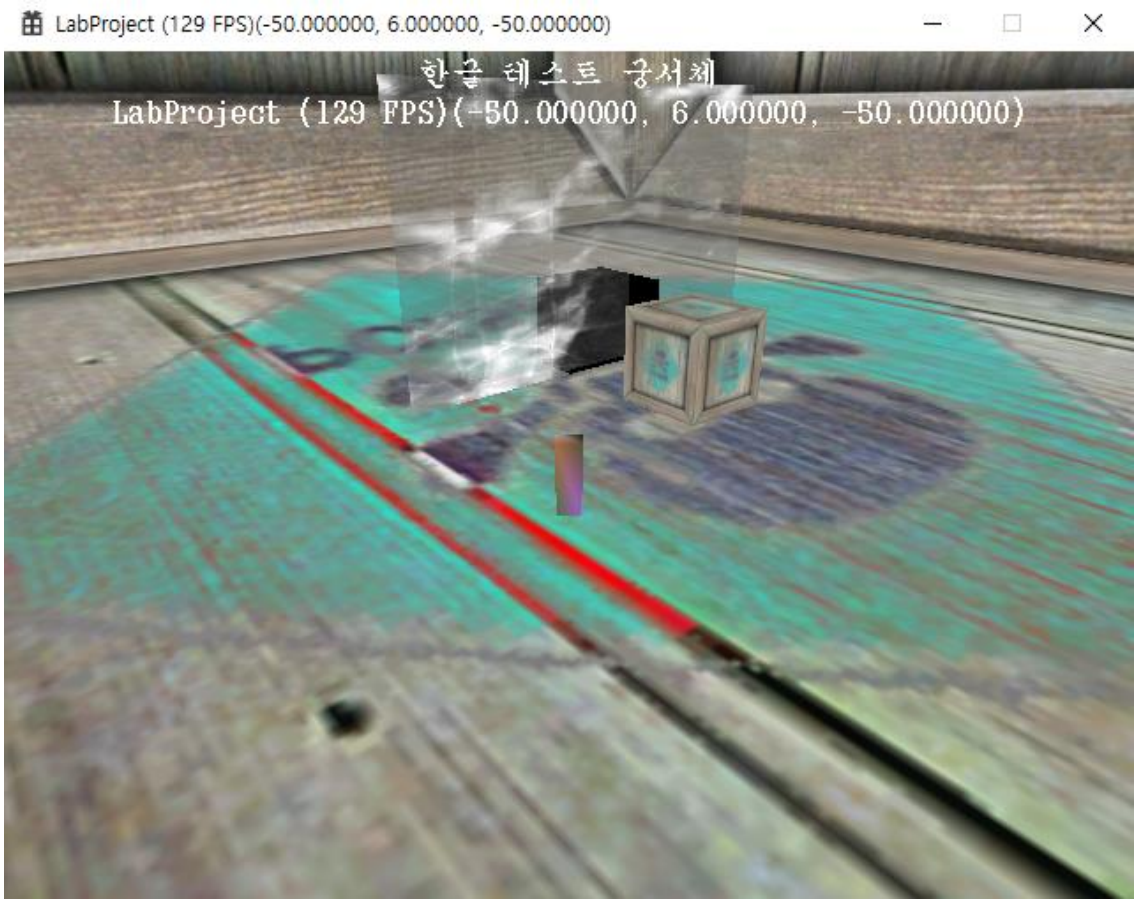
7. 조작법

① 키보드 이동키로 이동

② 마우스 좌클릭으로 회전

8. 프로젝트 플레이 화면, 지형 전경


LabProject (122 FPS)(500.000000, 213.000000, 300.000000)
한글 테스트 궁서체
LabProject (122 FPS)(500.000000, 213.000000, 300.000000)


LabProject (129 FPS)(-50.000000, 6.000000, -50.000000)
한글 테스트 궁서체
LabProject (129 FPS)(-50.000000, 6.000000, -50.000000)