IMPLEMENTATION

**REPORT**

**Planet**

**-Fps game simulator-**

|  |  |
| --- | --- |
| 과목명 | 컴퓨터그래픽스실습(1) |
| 담당교수 | 김예진 |
| 조원1 | B577029 이준영 |
| 조원2 | B577041 하태훈 |
| 제출일 | 2019.06.18 |



목차

**1. 어플리케이션 소개3**

1.1 소개3

1.2 조작법3

**2. 영할 분담 4**

**3. 기능 소개4**

3.1 Height map 기반으로 지형 생성4

3.2 Color map을 이용하여 지형 맵핑 4

3.3 Color map을 이용하여 미니맵 UI 4

3.4 쿼드트리를 이용한 지형 분할 5

3.5 지형 멀티 텍스쳐링 5

3.6 노드기반 지형 높이에 따른 카메라 이동 5

3.7 공전과 자전 6

3.8 파티클 카메라 주위 적용 6

3.9 Sky Dome & 비트맵 구름6

**3. 기타5**

1. **어플리케이션 소개**

|  |  |  |
| --- | --- | --- |
| **제목** | | |
| Planet | | |
|  | | |
| **내용** | 얼음 행성을 directX를 이용해서 구현. 행성의 사실성을 높이기 위해 적절한 위치에 안테나, 부품 상자 등을 배치. 그 외에도 자전과 공전을 하는 행성이나 눈이 내리는 듯한 효과도 적용.  또한 단순한 행성 구현에서 끝나지 않고 사용자가 게임을 하는 듯한 경험을 주기 위해 카메라를 따라다니는 총기, 미니맵, UI 등의 다양한 장치를 추가해 FPS(first person shooter)게임과 유사한 경험을 할 수 있도록 제작. | |
| **조작법** | | |
| 키보드 WASD : 이동 | | 마우스 이동 : 카메라 회전  마우스 좌클릭 : 총 발사 |
|  | | mouse icon png에 대한 이미지 검색결과 |

1. **역할 분담**

|  |  |
| --- | --- |
| **구현 기능** | |
| **이준영** | 1. 지형  2. 높이 기반 이동  3. 모델링 배치  4. 미니맵 구현  5. 총기 사운드 |
| **하태훈** | 1. Sky Dome  2. Snow Particle  3. Camera Following Gun  4. BGM  5. 공전과 자전을 하는 행성  6. UI |

1. **기술 소개**

**3.1 Height map 기반으로 지형 생성**

|  |
| --- |
| **소개** |
| 0~255의 회색 색상의 밝기 값으로 지형의 높낮이를 표현하고 이를 삼각형 메쉬로 지형을 렌더링한다. |

**3.2 Color map을 이용하여 지형 맵핑**

|  |
| --- |
| **소개** |
| Height map으로 불러온 각 정점에 대해 컬러맵 이미지의 각 정점의 색상을 1:1 매핑시켜 지형의 각 정점에 색상 정보를 넣어 색상을 추가한다. |

**3.3 Color map을 이용하여 미니맵 UI**

|  |
| --- |
| **소개** |
| Color map을 미니맵의 지형 이미지로 사용한다. 또한, 작은 크기의 비트맵 이미지를 이용하여 카메라의 위치정보를 업데이트하여 사용자의 위치 정보를 미니맵 상에 표시한다. |

**3.4 쿼드트리를 이용한 지형 분할**

|  |
| --- |
| **소개** |
| 지형을 4개의 사각형으로 구역을 분할하고 하나의 구역 내에 폴리곤의 수가 일정 수 이하일 때까지 구역을 분할한다. 이 때 Frustum 클래스를 이용하여 보이지 않는 부분을 제거한다. |

**3.5 지형 멀티 텍스쳐링**

|  |
| --- |
| **소개** |
| 일반 텍스쳐만으로는 지형을 평평한 표면으로 렌더링하므로 만족스러운 렌더링을 얻지 못한다. 따라서, 범프 맵핑을 사용한다. 노말맵의 각 픽셀 라이팅 정보를 이용하여 보다 사실적인 표면을 표현할 수 있다. |

**3.6 노드기반 지형 높이에 따른 카메라 이동**

|  |
| --- |
| **소개** |
| 쿼드트리를 이용하여 분할한 구역을 하나의 노드라고 한다. 카메라, 즉 사용자가 해당 노드에 위치하고 있는지 판별하고, 노드 안의 폴리곤의 높이에 따라 카메라의 높이를 조정한다. |

**3.7 공전과 자전**

|  |
| --- |
| **소개** |
| DirectX에서 제공하는 기본 변환 행렬은 크게 비례, 회전, 이동의 3종류의 변환이 존재한다. 공전과 자전을 구현하기 위해서는 이 3종류의 변환 중 회전 행렬을 사용해야한다. |
| **구현에서 주의점** |
| **C = SRT**  기본 행렬의 변환은 위의 식처럼 비례\*회전\*이동의 순서로 계산한다. 하지만 여기서 공전을 할할 때 SR1(이 행성의 회전)TR2(축이 되는 행성의 회전)를 사용해서 계산해야 한다. |

**3.8 파티클 카메라 주위 적용**

|  |
| --- |
| **소개** |
| 파티클은 기본적으로 특정 범위를 지정해 그 범위내에 한가지 이미지를 다양한 위치 생성해각기 다른 효과를 적용시키는 것을 기본으로 한다. |
| **구현에서 주의점** |
| 눈이 내리는 효과를 구현하기 위해 처음에는 게임 속 맵 전체에 눈 파티클을 적용하려고 하였으나 이는 연산의 비효율성으로 프레임 드랍을 유발함. 이를 해결하기 위해 파티클의 생성 범위를 축소 시키고 카메라의 이동에 따라 파티클의 생성 위치로 이동하는 방식으로 구현. |

**3.9 Sky Dome & 비트맵 구름**

|  |
| --- |
| **소개** |
| 스카이 돔은 3가지를 주의해서 구현해야한다. 첫번째로 카메라가 어디로 이동하더라도 스카이 돔은 카메라를 둘러 싸야하기 때문에 스카이돔의 위치카 카메라의 위치에 따라 실시간으로 변해야한다.. 두번째로 카메라는 스카이돔 안에 있기에 뒷표면 컬링을 꺼야한다. 마지막으로 거리에 상관없이 스카이 돔이 모든 것을 덮게 하기 위해서 Z버퍼를 꺼야한다. |