| **비단길(Silk Road)** | |
| --- | --- |
| **12주차** | **2024.03.10 (일) ~ 2024.03.16 (토)** |

**■ 주간 회의**

**■ 다음 주 세부 계획**

**▶ [권순원]**

* 파티클 시스템 계속해서 제작
* 하드코딩한 네트워크 관련 코드를 계속해서 정리하기

**▶ [박준영]**

* 중간발표 스테이지에 쓸 맵 오브젝트 배치 완료 및 적용
* ai 컴포넌트 구현

**▶ [엄장헌]**

* 애니메이션 인스턴싱에 따른 문제 해결

**■ 작업 일지**

**▶ [권순원]**

이번 주 계획을 작성할 때 맵 불러오기가 완료되면 구현하기로 했던 위치를 조정해주는 네트워크 기능을 제작했다. 생각보다 맵과 캐릭터 구현이 빨리 될 것 같아 후에 작업을 편하게 하기 위해 하드코딩으로 일단 만들어 테스트를 했다.

처음에는 게스트가 연결되면 플레이어를 생성하는 방향으로 구현을 시작했다. 게스트가 연결되면 각 클라이언트에 서로의 플레이어 정보를 생성하는 방식으로 구현하고 있었는데 각 클라이언트마다 객체의 아이디가 다를 수 있기 때문에 이를 서로 전달해주거나 같게 맞춰줘야 했다. 하지만 현재 프레임워크에는 오브젝트이 아이디를 임의로 변경해주는 기능이 없었고, 이런 기능은 다른 팀원과 상의가 필요할 것 같았다. 그리고 현재는 플레이어에 들어가는 정보다 많이 없으니 상관 없지만 추후 플레이어가 제대로 구현된다면 네트워크 접속을 할 때마다 플레이어를 만드는 것은 좋은 방법이 아니겠다는 생각을 했다.

때문에 게임이 시작할 때 플레이어를 모두 제작하는 방향으로 바꾸어 구현을 했다. 우리 게임에는 최대 3명의 플레이어가 등장하는 것으로 결정되어 있기 때문에 데이터를 공유하면서 3명의 플레이어의 동시에 만들게 되니 위에 방법보다 더 이득이 있을 것이라 생각했다.



아직은 플레이어 클래스가 따로 없기 때문에 임시 객체로 만들었다(추후 다른 팀원이 만든 플레이어 모델을 넣을 예정). 그리고 플레이어 아이디를 객체의 아이디와 별도로 두어 호스트 플레이와 게스트 플레이어의 플레이어 아이디를 결정해 두었다. 이렇게 제작한 이유는 컴포넌트를 끄고 켜는 기능이 없었기 때문에 직접 컴포넌트에 아이디를 주어 플레이어의 네트워크 상태(호스트, 게스트1, 2)에 따라 움직일 수 있는 플레이어를 변경하게 하기 위함이다. 이 부분은 오브젝트 자체와 컴포넌트를 활성화/비활성화 할 수 있는 기능을 추가하면서 변경할 것이다.

이후는 기존의 만들어 놓은 네트워크 구현에 따라 정보를 교환하고 갱신하게 된다.

한 가지 바뀐부분은 정보를 갱신하는 Update()부분인데, 플레이어의 경우 호스트인지 게스트인지에 따라 정확한 아이디가 생겼기 때문에 직접적으로 아이디로 접근해 위치를 갱신해 주는 것으로 변경했다.



여러번 테스트를 하면서 몇 가지 문제점이 있었다.

1. 프레임 저하
2. 게스트 클라이언트의 간헐적인 에러(튕김)

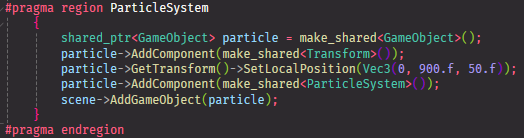
1번의 경우 아직 주고받는 패킷의 형태가 위치만 주고 받는 임시 데이터이기 때문에 문제가 발생했다고 예상하고 있기 때문에, 콘텐츠 제작을 하면서 주고 받는 데이터를 정리하고 네트워크 통신 횟수를 조정한 다음에 다시 확인해야할 문제인것 같았다.

2번의 경우 정확한 문제점을 찾지 못 했다., 예상하기로는 게스트가 호스트 서버에 접속하는 쓰레드에서 NetworkManager에 접근하는데, 이때 메인 쓰레드에서도 NetworkManager에 접근하여 동기화를 제대로하지 못해 생긴 문제가 생긴것이 아닌가 생각을 하고 있다.

위에 예상한 것이 맞다면 서버 제작 경험이 부족해 까먹고 처리를 해주지 못한것이기 때문에 쉽게 해결할 수 있을 것이라고 생각하지만, 만약 다른 곳에 원인이 있다면… 또 새로운 디버그 모험을 떠나야 할 것 같다.

원래 계획했던 파티클 시스템은 약간 문제가 생겼다. 일단 맨 처음한 생각은 ‘유니티에 파티클 데이터 파일을 게임에 그대로 넣어 사용하겠다’였다. 방학 때 만든 프레임워크에 파티클과 3D게임프로그래밍을 들으면서 해봤던 파티클을 기반으로 만들어 보고 나서, 다시 유니티 파티클 prefeb을 보았다.





그런데 가만히 생각해보니 파티클을 제대로 구현하기 위해서는 쉐이더 파일이 필요하고 유니티 파티클에는 바로 프로젝트에 적용할 수 있는 형태의 쉐이더 파일이 없었다. 내가 제대로 못 찾았을 가능성을 아직 배제할 수는 없지만, 방학 때 공부하겠다고 봤을 때도 그렇고 이번에도 원하는 형태의 쉐이더 파일은 구할 수 없었다는 것으로 보아 다른 형태로 구현되어 있을 가능성이 큰 것 같다. 그렇다보니 맨 처음 내가 생각했던대로 쉽게 같은 파일로 프로젝트에 적용을 하진 못할 것 같다는 생각이 들었다.

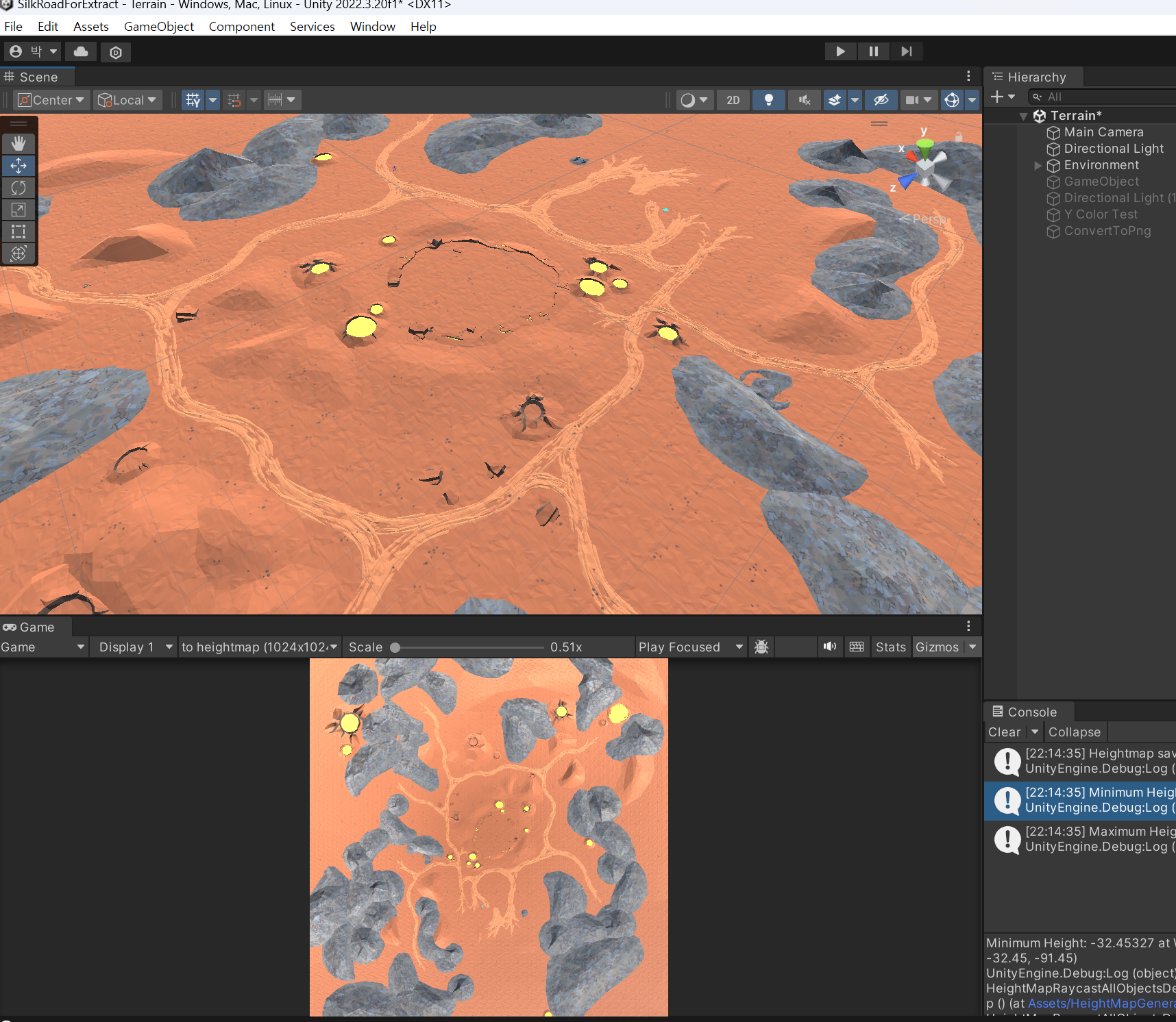
때문에 살짝 방향을 바꾸어 유니티에서 모델 fbx를 바이너리 파일로 만들어 추출했던 것처럼 파티클 prefab도 스크립트를 이용해 추출하는 방법을 생각해봐야 겠다.

**▶ [박준영]**

중간발표 스테이지에 적용될 L1단계의 맵을 제작 및 적용시키는데 성공했다.

생각한 것 보다 여러 우여곡절이 있었다.

우선 터레인 전역에 적용할 맵 전체 텍스쳐를 추출했다.

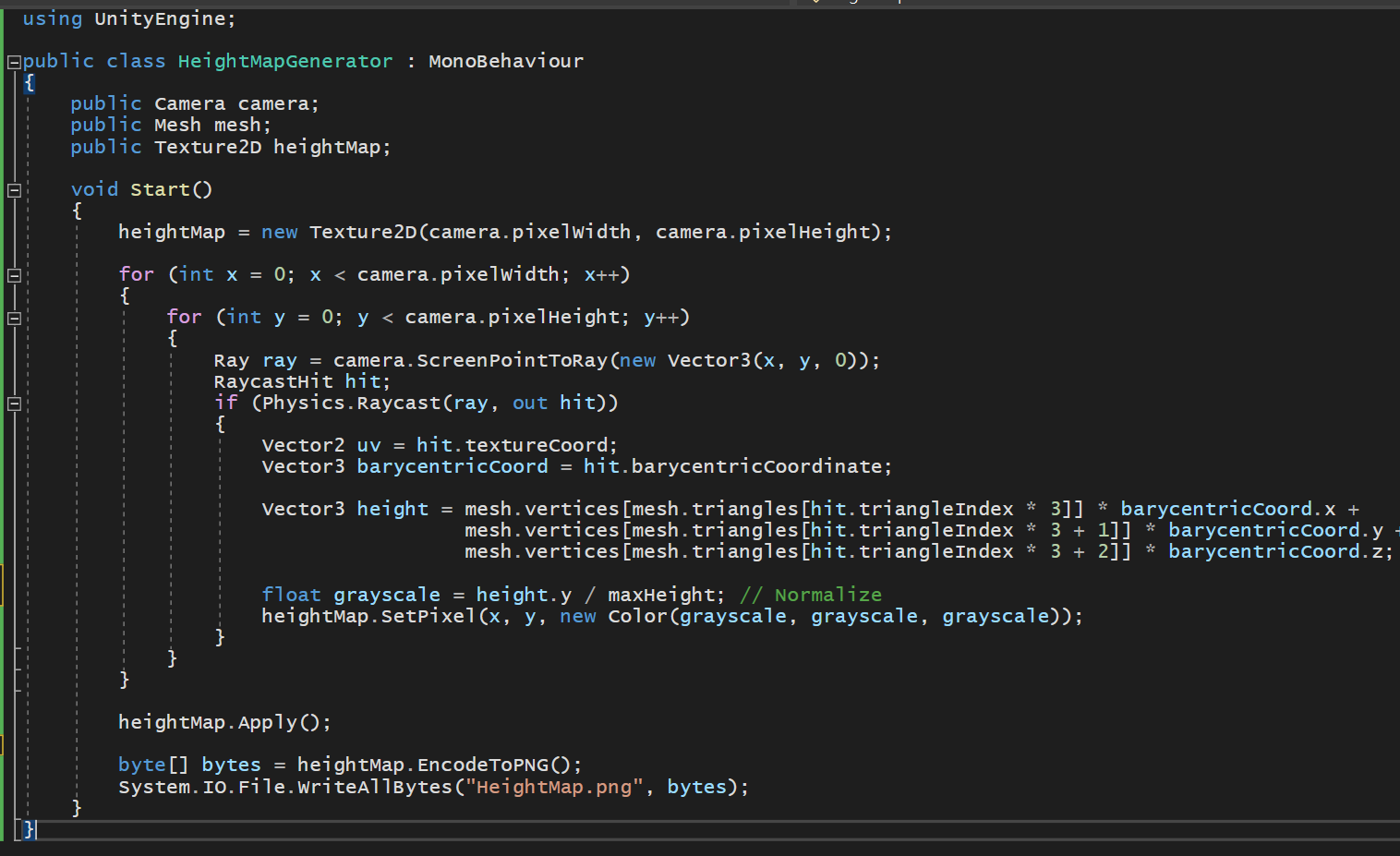
****

500m x 500m의 크기의 맵에 적용해야 하다 보니 자연스럽게 보이려면 텍스쳐의 해상도가 적어도 4k x 4k급 이상이 되어야했다.

제작했던 맵에서 게임플레이 시 불필요할 것 같던 지형지물을 삭제하고 다듬었다.

그 후, 직교투영으로 설정한 메인카메라를 공중에 배치한 후, 500x500 만큼의 영역을 촬영하도록 설정하고, Start시 지정한 render texture에 카메라의 view를 쓰고(이 때, 마테리얼의 Albedo 컬러만 적용되게 설정했다.)

이렇게 해서 내가 제작한 맵의 텍스쳐를 원하는 해상도의 PNG파일로 뽑아내는 것 까지는 완료했으나, 높이맵 생성에서 살짝 애를 먹었다. 처음엔 시험삼아 맵 텍스쳐 파일을 단순히 포토샵의 흑백 및 곡선 조정으로 높이맵을 만들어 보려고 시도했다. 과거에 이런 방식으로 재질을 표현하는 높이맵이나 노말맵을 제작하는걸 보고 따라해봤는데 내가 만들어야 했던 전체 맵을 담는 높이맵에는 전혀 적합하지 않았다. 그 이후, unity의 ray기능을 사용하여 높이맵의 각 픽셀에 해당하는 좌표 지점을 레이캐스트하여 최초로 hit하는 지점의 높이값을 0~1사이의 값으로 정규화하여 grayscale로 출력하도록 하였다.

****

아무리 생각해도 잘 작동해야 정상인데, 결과물로 나온 높이맵이 온통 까맣거나 색상이 희멀건하게 뜨는 등의 문제가 계속 발생하였다.

결국 ray가 hit하지 않는 위치, ray가 hit한 위치 중 최고 높이 지점과 최저 높이 지점등을 출력해서 확인해 본 결과, 거의 눈에 보이지 않을 정도로 작은 구멍이 몇군데 맵상에 뚫려있던 것을 확인했고, 이 지점에서 다른 높이값들과 비정상적으로 큰 차이가 있는 높이값이 발생하여 정규화 과정에서 다른 높이들이 거의 드러나지 못하게 하던 것이다.

이 지점들을 포착하여 잘 막고 정상적으로 높이맵을 뽑아내었다.****

지금껏 수업에선 제공된 높이맵을 적용하는 것만 해보았지, 내가 직접 오브젝트를 배치해 만든 맵을 높이맵으로 변환하는 과정은 처음이어서 조금 삽질을 했다.

더불어 unity에서 배치한 맵의 최대, 최저 고도를 출력하도록 하여 우리 프로젝트에서 터레인의 스케일을 이 값을 기반으로 설정하도록 하였다.

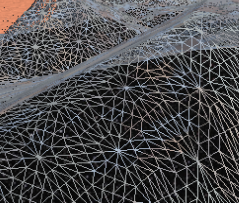
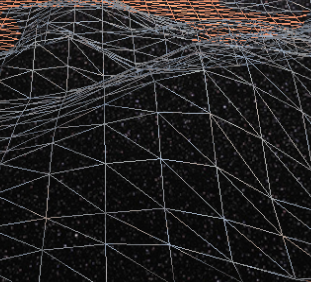
실제 적용 시에 캐릭터들의 발이 지형에 빠지는 문제가 발견되었지만 장헌이가 빠르게 찾아내 해결해주어서 금방 적용가능 하였다.

현재는 프레임워크의 최대 오브젝트 개수 문제로 인해 정체되었던 L2 오브젝트 배치 작업을 수행중이다. 방학동안 만들었던 맵 오브젝트(각종 트리거 개체) Export 시스템을 기반으로 생성한 JSON파일(XML로 바꿀까 고민중)을 읽어들여 실제 메시렌더러가 포함된 게임오브젝트들을 지정된 위치에 배치하는 것이다. 물론 AI개발도 병행해야 한다. 이제 컨텐츠들을 개발할 초석들이 어느정도 갖춰졌으니 게속 열심히 개발해야겠다.

**▶ [엄장헌]**

이번 주에 준영이가 작업해서 준 전체 지형의 높이맵과 텍스쳐를 받았다. 이를 저번주에 내가 만든 터레인에 적용을 해봤다. 하지만 완만한 터레인이였던 내 지형에서는 잘 보이지 않던 터레인과 물체의 높이차가 확인되었다. 예를 들어 캐릭터의 무릎까지 터레인에 잠긴다던가, 공중에 떠있는등의 문제를 발견했고, 이유를 찾았다.

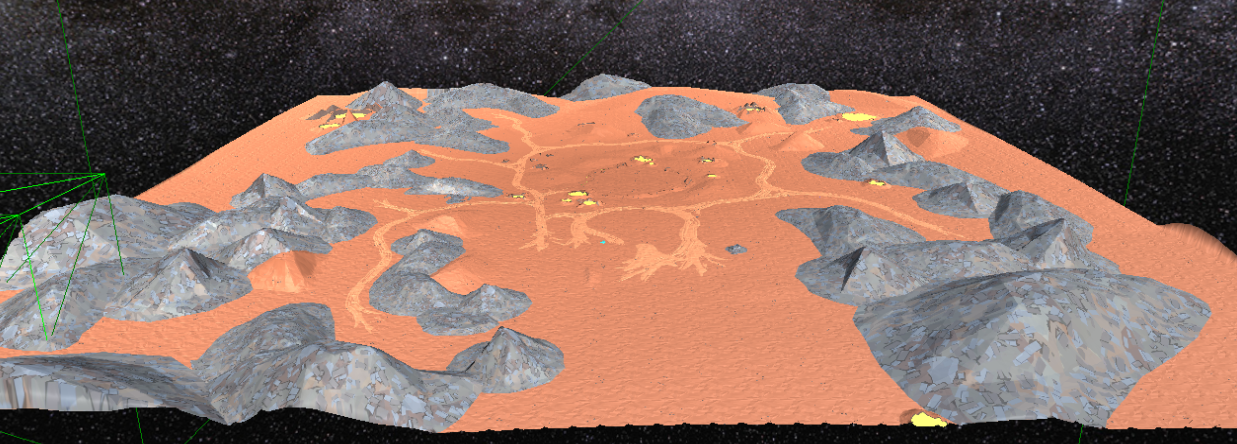
처음에는 테셀레이션으로 인한 높이차라 생각했다. 다만 테셀레이션 때문에 발생했다고 하기에는 잠기는 높이차가 굉장히 심했고,다른 이유가 있었다.



우리의 터레인은 삼각형 두개로 이루어진 사각형들의 집합 매쉬에 높이맵을 적용하게 되는데 하나의 사각형을 이루는 삼각형은 한 평면에 있지 않는 경우가 많다. 하지만 나의 높이계산을 위한 선형보간은 한 평면에서의 계산이여서 발생했다.

따라서 어느 삼각형인가에 따라 값을 바꿔서 계산을 하였다.





적용 완료된 사막 스테이지의 전체 지형이다.

그 다음으로 진행한 것은 애니메이션작업이다. 이전에 루트 모션이 적용이 안되었다. 예를 들어 넘어져야한다면 root의 값이 변경되며 회전하며 넘어져야하지만, 이것의 애니메이션이 적용되지 않아 선채로 넘어지게 된다. 따라서 루트 모션을 적용할 필요성을 느꼈고 계층구조의 형태를 한 단계 추가해 따로 값을 넣는 것으로 해결하였다..



이전에 어색하게 동작하던 애니메이션이 깔끔하게 적용되는 것을 확인했다.

다만 문제가 있다면 인스턴싱을 이용해서 한번의 쉐이더에 데이터를 전해주는 것으로 매쉬를 그리게 되는데 이 경우 하나의 객체의 애니메이션 값을 바꿔도 전체가 똑같이 움직이게 된다. 따라서 잠시 인스턴싱을 끄고 해보니, 이 경우는 프레임 드랍이 굉장히 크게 일어났다. 따라서 인스턴싱을 적용한 상태로 위치값만 따로 보내주던 방식에 애니메이션 프레임이나, 몇번째 애니메이션인지 등의 정보만 보내는 식으로 적용하면 될 것 같은데 아직은 잘 안되고 있다.