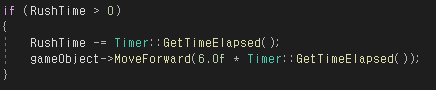
**2018180020 박재우(클라이언트) 21주차 기록**

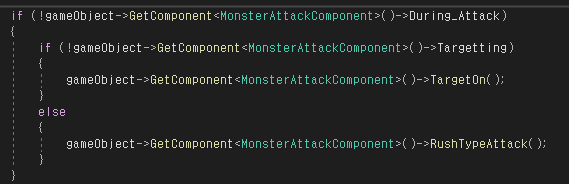
**돌진 몬스터 FSM**

저번주에 다 못한 돌진형의 몬스터를 만든다.

돌진 몬스터는 그냥 돌진해서 플레이어랑 부딪히게 만든다. 애니메이션은 대쉬하는 애니메이션으로 설정한다. 공격하는 동안에 빠르게 앞으로 나아가 돌진을 하는 것 같은 느낌을 주도록 한다. MonsterAttack 컴포넌트에서 RushTime 변수를 추가한다. RushTime이 돌진 몬스터가 공격하는 시간이다. RushTime을 1초로 설정했는데 1초동안 몬스터가 앞으로 나아가게 된다.



RushTime이 0보다 클 때 프레임 시간을 계속해서 빼주면서 몬스터를 앞으로 옮겨준다. 옮겨주는 값은 6.0 \* ElapsedTime인데 RushTime이 1초이기 때문에 1초동안 6 만큼 움직이게 된다. 정리하면 공격했을 때 몬스터는 1초동안 6m를 앞으로 나아가는 것이다.



FSM의 Attack 함수를 다른 몬스터와는 다르게 수정했다. 다른 몬스터는 During\_Attack에 따라 공격을 했지만 돌진 몬스터는 추가로 TargetOn 함수를 부르게 된다. TargetOn 함수는 내부에서 TargetCoolTime 변수에 프레임 시간을 빼주고 TargetCoolTime이 0이 되면 Targetting을 true로 바꿔 FSM에서 RushTypeAttack함수가 불릴 수 있도록 했다. Target CoolTime에 의해 몬스터는 TargetCoolTime이 0이 될 동안 공격을 할 수 없다. 이렇게 한 이유는 근거리나 원거리가 사정거리에 들어올 때까지 대비할 시간이 충분하지만 돌진은 사정거리에 들어오자마자 빠른 속도로 밀고 들어오기 때문에 플레이어가 대비할 수 없다. 게임 기획상 돌진 몬스터의 공격은 매우 치명적이기 때문에 피할 수 없으면 난이도가 너무 높아진다. 그렇기 때문에 1초정도 가만히 서있다가 바라보는 방향으로 빠르게 돌진하도록 해 피할 준비를 할 시간을 줄 수 있게 했다. 이 부분은 마지막에 밸런스 관련해 고치게 되면 수치를 조절하도록 한다.

**Tessellation**

지형이 계단처럼 각진 현상을 수정하기 위해 테셀레이션을 해준다. 테셀레이션을 하기 위해 Hull Shader와 Domain Shader가 추가된다. 비교를 위해 Vertex Shader와 Pixel Shader는 기존의 Shader코드와 같은 동작을 하지만 PS의 인풋이 다르다. VS의 인풋과 아웃풋은 동일하지만 PS는 인풋이 Hull Shader와 Domain Shader를 거친 결과물이 들어가게 된다. PS만 인풋을 바꾼 버전을 추가한다.

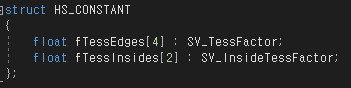


VS\_TERRAIN\_OUTPUT 이 인풋이었던 것을 DS\_TERRAIN\_OUTPUT으로 변경한 새로운 PS를 만들었다.

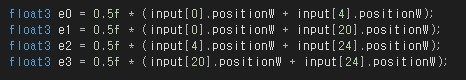
Hull Shader는 폴리곤을 여러 패치로 나누는 쉐이더이다. 우리는 사각형을 25개의 패치로 나누도록 한다.



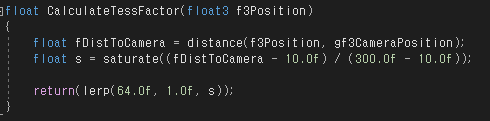
InputPatch에 VS에서 계산된 결과와 25를 넣어 패치를 25개 만든다. 여기에 필요한 제어점은 파이프라인에서 자동으로 생성된다. 우리는 한 변을 몇 개로 나누고 내부의 사각형을 얼마나 만드는지 정해주면 된다. 이를 지정하기 위한 상수를 계산하는 함수를 만든다.



터레인이 넓기 때문에 모든 지형을 전부 테셀레이션을 하기보단 플레이어 주변만 테셀레이션을 해주는 것이 효율적이기 때문에 카메라 거리에 따라 값을 동적으로 지정해 주도록 한다.



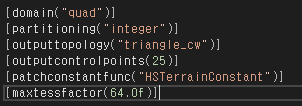
패치가 25개이기 때문에 각 변을 알 수 있게 각 꼭지점의 패치의 위치를 더하고 0.5를 곱해 중간 값을 구한다. 이 값이 한 변의 위치이다.



구한 변의 위치를 이용해 카메라와의 거리를 구한 다음 saturate 함수를 이용해 가중치를 구한다. 거리가 300이 넘어가면 1 이상이 된다. saturate함수는 0과 1사이로 제한하는 함수이므로 값은 1이 된다. 만약 거리가 10보다 작으면 음수가 되어 0보다 작아지고 saturate함수를 적용하면 0이된다. 즉, 테셀레이션을 최대로 하는 최소거리가 10이며 테셀레이션을 적용하는 최대 거리는 300이다. 300을 넘어가면 테셀레이션 팩터가 1이 되어 테셀레이션을 하지 않게 된다.

HS에서 e0, e1, e2, e4를 CalculateTessFactor 함수로 계산한 결과를 fTessEdges의 각 인덱스에 저장한다.

모든 패치의 위치의 중간 값을 구해 똑같이 CalculateTessFactor 함수로 계산한 결과를 fTessInsides에 저장한다.



HS에서 어떻게 처리하는 지 정의해준다. 위에서부터 차례대로 패치의 유형, 분할법, 출력 프리미티브 유형, 출력 제어점 개수, 사용할 패치함수 이름, 최대 테셀레이션 인자이다.

우리 게임은 사각형 패치로 분할하고 테셀레이션 인자보다 크거나 같은 자연수만큼 분할(integer)하며 프리미티브는 시계방향의 삼각형으로, 출력 제어점은 25개이고 사용할 함수는 HSTerrainConstant이고 테셀레이션 인자는 최대 64개이다.

이렇게 계산된 출력값은 테셀레이터에 전달되어 새로운 정점이 생성된다.



DS는 테셀레이터에서 정점을 생성할 때마다 호출되는 쉐이더이다. 입력으로 HS단계에서 계산된 결과를 받는다.

DS에서 공식을 이용해 보간 해서 값을 계산해 PS에 넘겨준다.

쉐이더를 완성하고 실행해 봤더니 지형이 그려지지 않았다. D3D12 오류를 전부 해결하고 관련 오류가 뜨지 않는 것을 보아 쉐이더에 값이 제대로 들어가지 않는 문제인 듯하다.

코드 한 줄 한 줄 읽으면서 어디가 문제인지 찾아야겠다.