포트폴리오

목차

1. DirectX12를 이용한 컴퓨터 그래픽스 기술 구현 프로젝트
2. IOCP, Unity를 이용한 게임
3. Unreal4를 이용한 레이싱 게임

이름 : 남주영

번호 : 010-7735-0943

GitHub : https://github.com/Juyeong11/portfolio

DirectX12를 이용한 컴퓨터 그래픽스 기술 구현 프로젝트

컴퓨터 그래픽스 기술에 대한 이해를 위해

구현해보고 싶은 기술 선정 후 계획 수립 후 제작

목표

구현할 기술 선정

- 그림자 맵
- 동적 파티클
- 지형 동적 LOD
- 블러링

계획

구현하기 위해 필요한 정보 수집

- 그림자 맵 생성 방법
- 동적 파티클을 위한 스트림 아웃단계 사용방법
- 테셀레이션 단계를 사용하기 위한 도메인 셰이더 헐 셰이더 사용방법
- 포스트 프로세싱을 위한 계산 셰이더 사용방법

구현

구현하며 생기는 버그를 해결

- 동적으로 변하는 지형 위를 플레이어가 밟지 못하고 떠다니던 문제
- 스트림 아웃단계에 생성된 정점이 입력 조립단계에서 확인이 안되던 문제

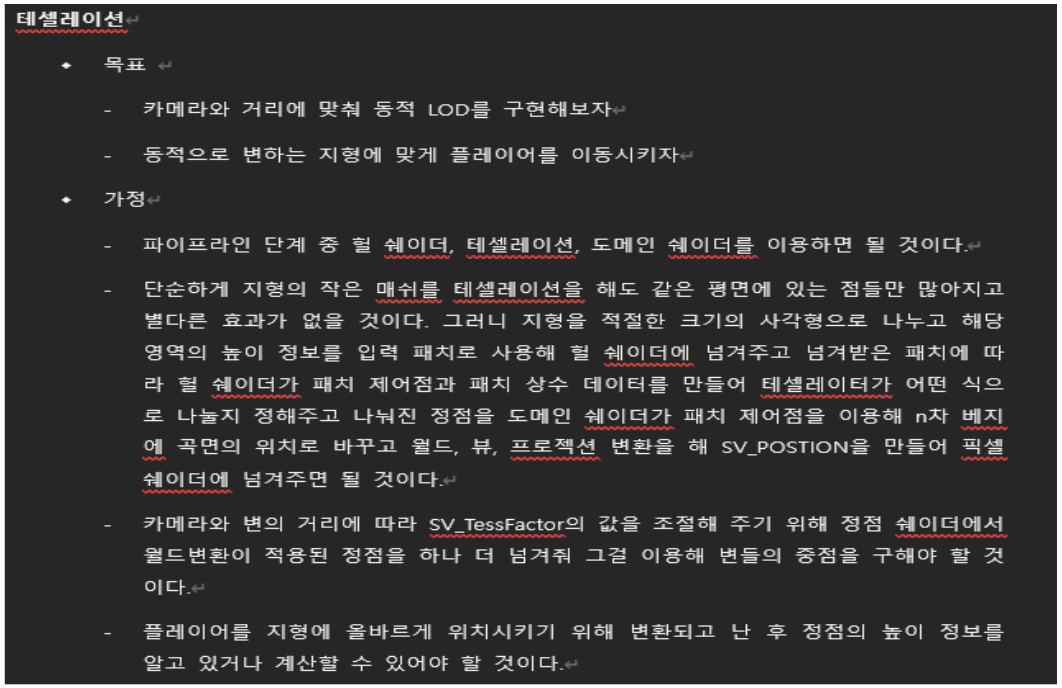
프로젝트의 소스코드와 자세한 설명을 보고 싶으시다면 아래의 링크를 클릭해주세요

https://github.com/Juyeong11/portfolio/tree/main/DirectX12/D3D12_2

진행 과정

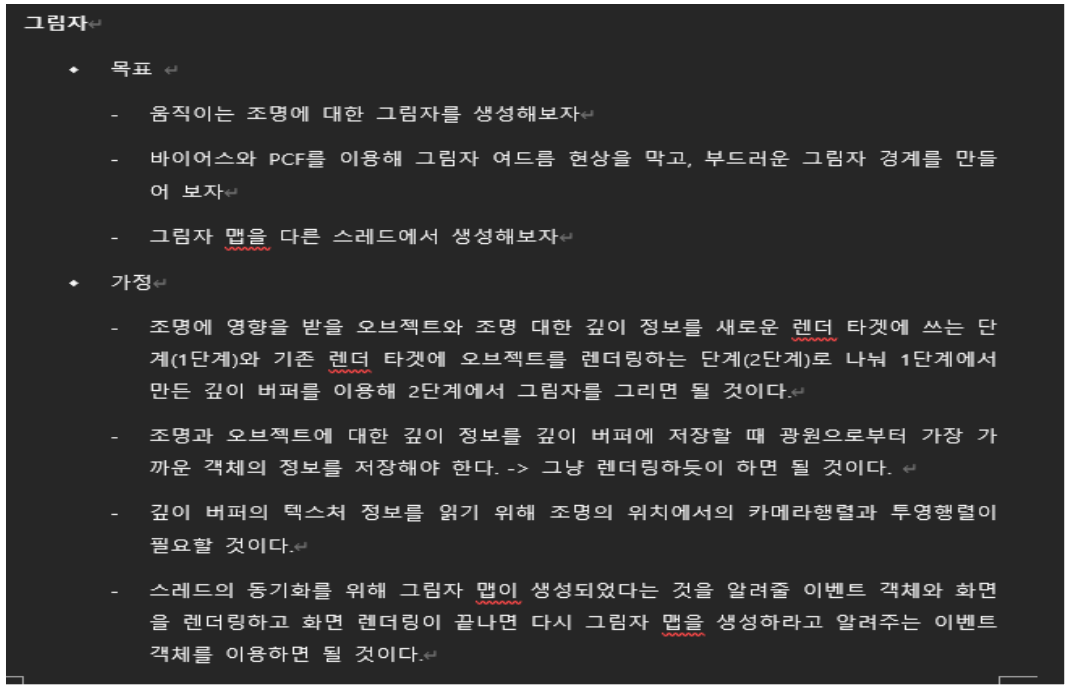
목표 설정 및 계획 수립

구현할 기술을 정하고,
필요한 정보를 찾아 구현 해야 할 사항을 정리하며 진행했습니다.



테셀레이션

- 목표
  - 카메라와 거리에 맞춰 동적 LOD를 구현해보자
  - 동적으로 변하는 지형에 맞게 플레이어를 이동시키자
- 가정
  - 파이프라인 단계 중 헐 쉐이더, 테셀레이션, 도메인 쉐이더를 이용하면 될 것이다.
  - 단순하게 지형의 작은 매쉬를 테셀레이션을 해도 같은 평면에 있는 점들만 많아지고 별다른 효과가 없을 것이다. 그러니 지형을 적절한 크기의 사각형으로 나누고 해당 영역의 높이 정보를 입력 패치로 사용해 헐 쉐이더에 넘겨주고 넘겨받은 패치에 따라 헐 쉐이더가 패치 제어점과 패치 상수 데이터를 만들어 테셀레이터가 어떤 식으로 나눌지 정해주고 나눠진 정점을 도메인 쉐이더가 패치 제어점을 이용해 n차 베지에 곡면의 위치로 바꾸고 월드, 뷰, 프로젝션 변환을 해 SV_POSTION을 만들어 픽셀 쉐이더에 넘겨주면 될 것이다.
  - 카메라와 변의 거리에 따라 SV_TessFactor의 값을 조절해 주기 위해 정점 쉐이더에서 월드변환이 적용된 정점을 하나 더 넘겨줘 그걸 이용해 변들의 중점을 구해야 할 것이다.
  - 플레이어를 지형에 올바르게 위치시키기 위해 변환되고 난 후 정점의 높이 정보를 알고 있거나 계산할 수 있어야 할 것이다.



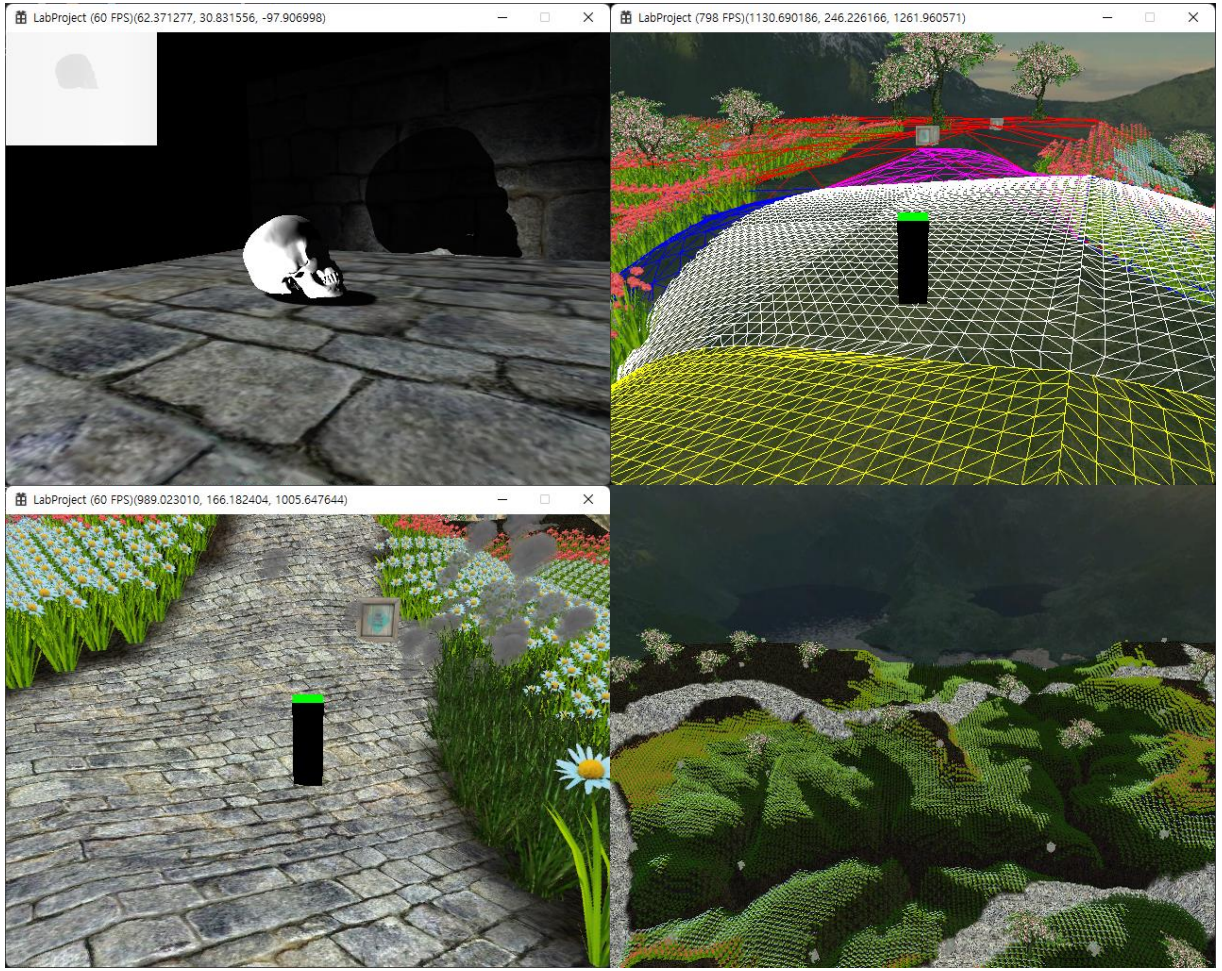
그림자

- 목표
  - 움직이는 조명에 대한 그림자를 생성해보자
  - 바이어스와 PCF를 이용해 그림자 여드름 현상을 막고, 부드러운 그림자 경계를 만들어 보자
  - 그림자 맵을 다른 스레드에서 생성해보자
- 가정
  - 조명에 영향을 받을 오브젝트와 조명 대한 깊이 정보를 새로운 렌더 타겟에 쓰는 단계(1단계)와 기존 렌더 타겟에 오브젝트를 렌더링하는 단계(2단계)로 나눠 1단계에서 만든 깊이 버퍼를 이용해 2단계에서 그림자를 그리면 될 것이다.
  - 조명과 오브젝트에 대한 깊이 정보를 깊이 버퍼에 저장할 때 광원으로부터 가장 가까운 객체의 정보를 저장해야 한다. -> 그냥 렌더링하듯이 하면 될 것이다.
  - 깊이 버퍼의 텍스처 정보를 읽기 위해 조명의 위치에서의 카메라행렬과 투영행렬이 필요할 것이다.
  - 스레드의 동기화를 위해 그림자 맵이 생성되었다는 것을 알려줄 이벤트 객체와 화면을 렌더링하고 화면 렌더링이 끝나면 다시 그림자 맵을 생성하라고 알려주는 이벤트 객체를 이용하면 될 것이다.

해당 파일은 여기서 보실 수 있습니다.
https://github.com/Juyeong11/portfolio/blob/main/DirectX12/D3D12_2/DirectX12_%EC%84%A4%EB%AA%85%EB%AC%B8%EC%84%9C2.pdf

진행 결과

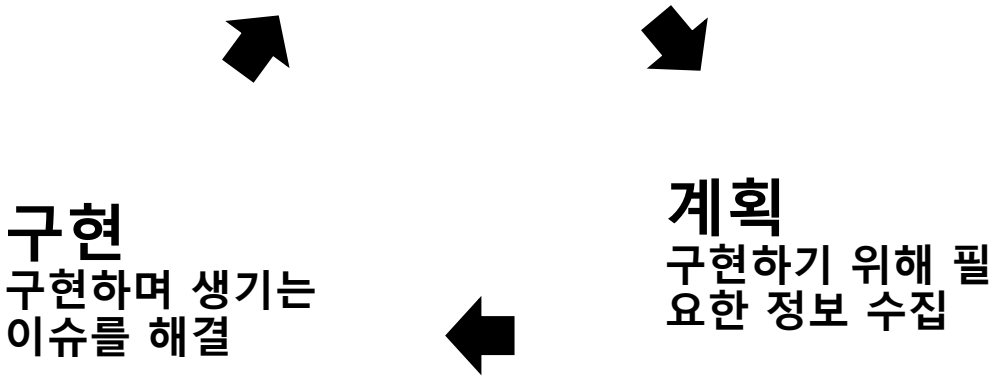
기본적인 기술을 구현해보면서 얻은 정보를 바탕으로 **더 발전된 형태의 기술**을 찾아보며 이해할 수 있었습니다.



- "DirectX12를 이용한 3D 게임 프로그래밍 입문"과 "게임 프로그래밍을 위한 3차원 그래픽스" 두 책을 통해 **기술 구현을 위한 정보를 찾고 구현해** 보며 **DirectX12 API에 대한 숙련도와 컴퓨터 그래픽스 기술 구현을** 위한 지식을 얻었습니다.
- 구현 중 생기는 여러 오류를 해결하며 **디버깅 능력을** 키웠습니다.
- 디버깅을 쉽게 하기 위해서 로직을 최대한 단순화 하고 그것을 그대로 구현하며 **코드의 흐름을 정확히 이해하며** 제작하였습니다.
- 그림자의 구현을 위해 여러 자료를 찾아보던 중 **여러 개의 그림자 맵을 거리에 따라 사용하는 Cascade shadow**에 대해 찾아볼 수 있었습니다.
- 플레이어와 거리에 따라 변화하는 **지형의 찢어짐을 방지하기 위해 균열부분에 삼각형을 추가하는 방법**을 찾아볼 수 있었습니다.
- **포스트 프로세싱을 스텐실 버퍼를 이용해** 원하는 오브젝트에 적용하는 방법을 찾아 볼 수 있었습니다.

IOCP, Unity를 이용한 게임 제작
다중 접속 게임에 대한 이해를 위해

팀원과 소통하며 요구사항을 파악하고 필요한 기능을 구현



구현 사항

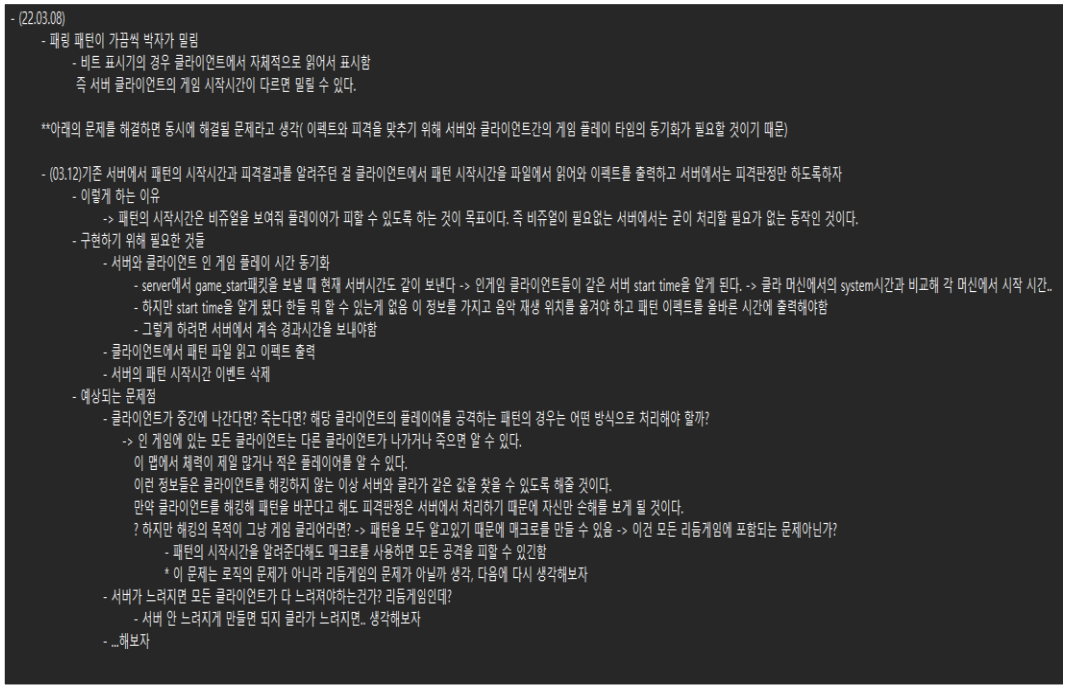
- C#과 C++ 간 통신
- 7개의 스레드를 이용한 로직, 패킷 처리
- DB를 이용한 로그인 및 플레이어 정보 수정
- 타이머 큐를 이용한 이벤트 처리
- A*를 이용한 길 찾기
- 플레이어 시야 처리
- 플레이어 스킬
- 보스 스킬
- 인던

프로젝트의 소스코드와 자세한 설명을 보고 싶으시다면 아래의 링크를 클릭해주세요
https://github.com/Juyeong11/Graduation_project/tree/main/BeatSlimeServer

진행 과정

구현 과정

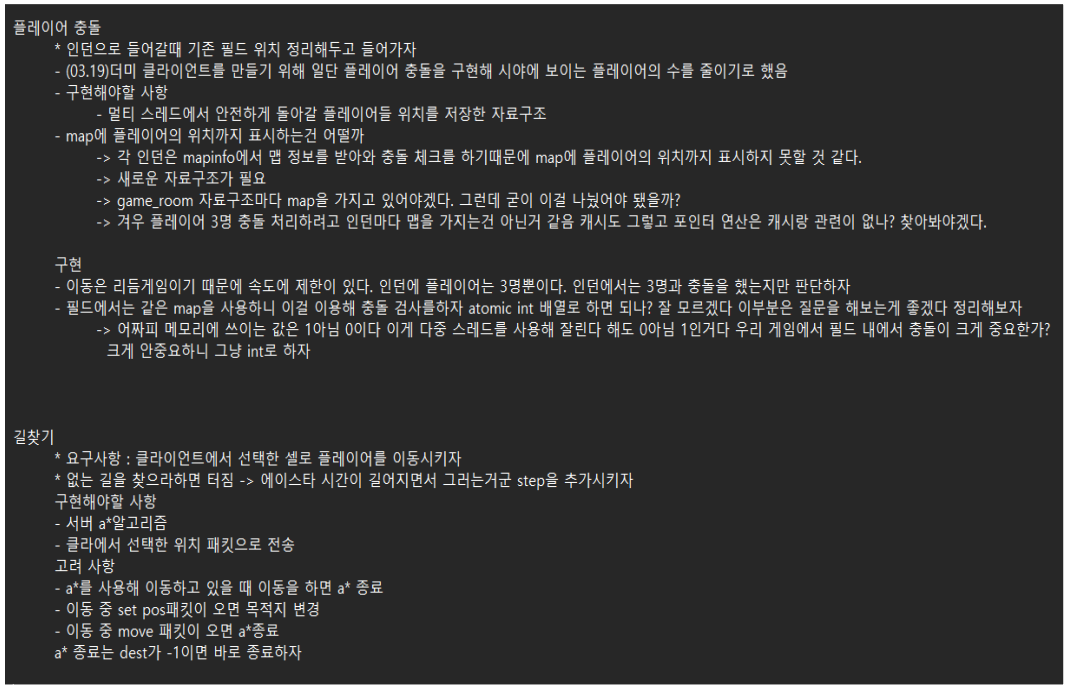
요구사항을 파악하고
필요한 기능 정리하며 구현했습니다.



- (22.03.08)
  - 패킹 패턴이 가끔씩 박자가 밀림
    - 비트 표시기의 경우 클라이언트에서 자체적으로 읽어서 표시함
      즉 서버 클라이언트의 게임 시작시간이 다르면 밀릴 수 있다.

  **아래의 문제를 해결하면 동시에 해결될 문제라고 생각(이펙트와 피격을 맞추기 위해 서버와 클라이언트간의 게임 플레이 타임의 동기화가 필요할 것이기 때문)

  - (03.12)기존 서버에서 패턴의 시작시간과 피격결과를 알려주던 걸 클라이언트에서 패턴 시작시간을 파일에서 읽어와 이펙트를 출력하고 서버에서는 피격판정만 하도록하자
    - 이렇게 하는 이유
      -> 패턴의 시작시간은 비쥬얼을 보여줘 플레이어가 피할 수 있도록 하는 것이 목표이다. 즉 비쥬얼이 필요없는 서버에서는 굳이 처리할 필요가 없는 동작인 것이다.
    - 구현하기 위해 필요한 것들
      - 서버와 클라이언트 인 게임 플레이 시간 동기화
        - server에서 game_start패킷을 보낼 때 현재 서버시간도 같이 보낸다 -> 인게임 클라이언트들이 같은 서버 start time을 알게 된다. -> 클라 머신에서의 system시간과 비교해 각 머신에서 시작 시간..
        - 하지만 start time을 알게 됐다 한들 뭐 할 수 있는게 없음 이 정보를 가지고 음악 재생 위치를 옮겨야 하고 패턴 이펙트를 올바른 시간에 출력해야함
        - 그렇게 하려면 서버에서 계속 경과시간을 보내야함
      - 클라이언트에서 패턴 파일 읽고 이펙트 출력
      - 서버의 패턴 시작시간 이벤트 삭제
    - 예상되는 문제점
      - 클라이언트가 중간에 나간다면? 죽는다면? 해당 클라이언트의 플레이어를 공격하는 패턴의 경우는 어떤 방식으로 처리해야 할까?
        -> 인 게임에 있는 모든 클라이언트는 다른 클라이언트가 나가거나 죽으면 알 수 있다.
        이 맵에서 체력이 제일 많거나 적은 플레이어를 알 수 있다.
        이런 정보들은 클라이언트를 해킹하지 않는 이상 서버와 클라가 같은 값을 찾을 수 있도록 해줄 것이다.
        만약 클라이언트를 해킹해 패턴을 바꾼다고 해도 피격판정은 서버에서 처리하기 때문에 자신만 손해를 보게 될 것이다.
        ? 하지만 해킹의 목적이 그냥 게임 클리어라면? -> 패턴을 모두 알고있기 때문에 매크로를 만들 수 있음 -> 이건 모든 리듬게임에 포함되는 문제아닌가?
          - 패턴의 시작시간을 알려준다해도 매크로를 사용하면 모든 공격을 피할 수 있긴함
          * 이 문제는 로직의 문제가 아니라 리듬게임의 문제가 아닐까 생각, 다음에 다시 생각해보자
      - 서버가 느려지면 모든 클라이언트가 다 느려져야하는건가? 리듬게임인데?
        - 서버 안 느려지게 만들면 되지 클라가 느려지면.. 생각해보자
      - ...해보자



플레이어 충돌
  * 인던으로 들어갈때 기존 필드 위치 정리해두고 들어가자
  - (03.19)더미 클라이언트를 만들기 위해 일단 플레이어 충돌을 구현해 시야에 보이는 플레이어의 수를 줄이기로 했음
  - 구현해야할 사항
    - 멀티 스레드에서 안전하게 돌아갈 플레이어들 위치를 저장한 자료구조
  - map에 플레이어의 위치까지 표시하는건 어떨까
    -> 각 인던은 mapinfo에서 맵 정보를 받아와 충돌 체크를 하기때문에 map에 플레이어의 위치까지 표시하지 못할 것 같다.
    -> 새로운 자료구조가 필요
    -> game_room 자료구조마다 map을 가지고 있어야겠다. 그런데 굳이 이걸 나눴어야 됐을까?
    -> 겨우 플레이어 3명 충돌 처리하려고 인던마다 맵을 가지는건 아닌거 같음 캐시도 그렇고 포인터 연산은 캐시랑 관련이 없나? 찾아봐야겠다.

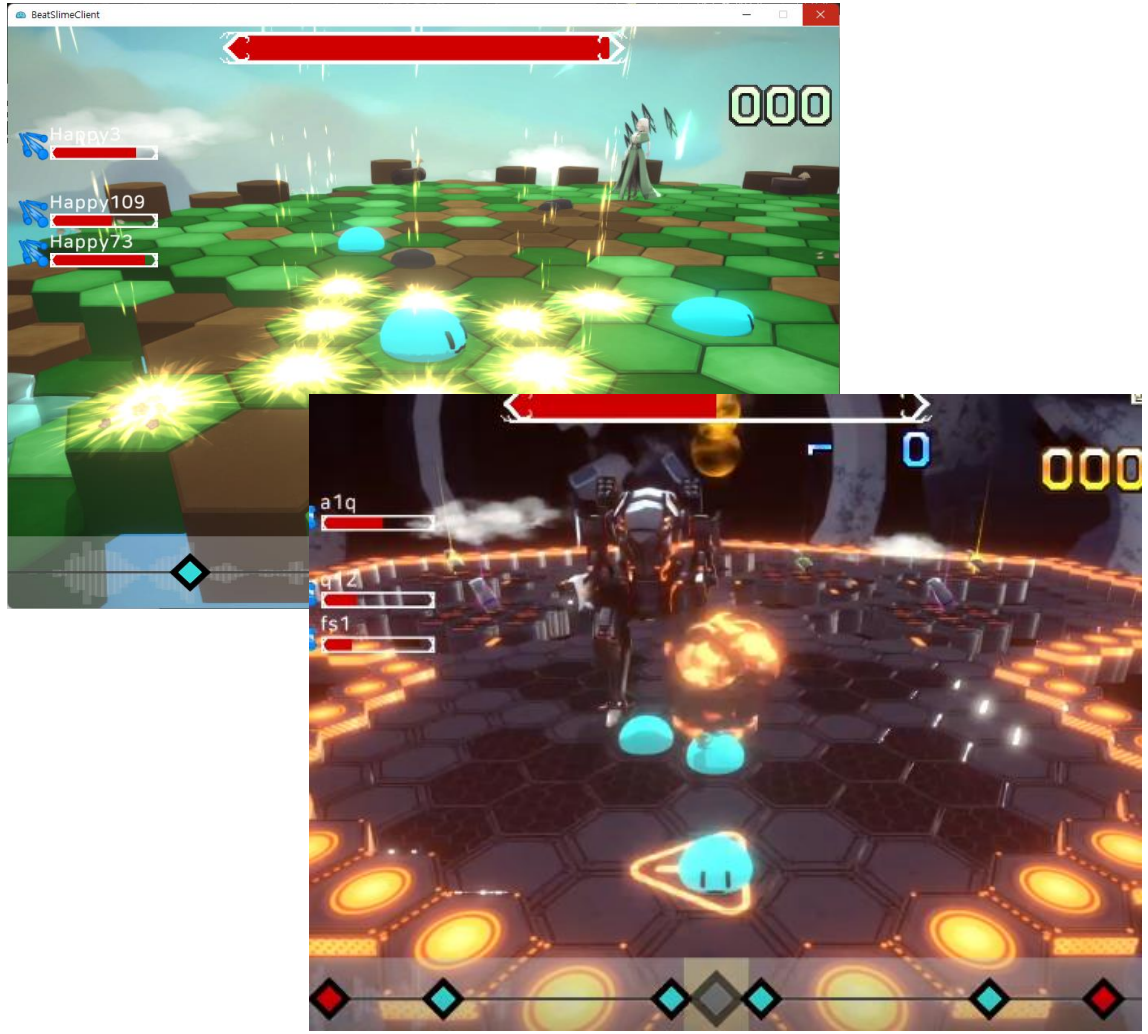
  구현
  - 이동은 리듬게임이기 때문에 속도에 제한이 있다. 인던에 플레이어는 3명뿐이다. 인던에서는 3명과 충돌을 했는지만 판단하자
  - 필드에서는 같은 map을 사용하니 이걸 이용해 충돌 검사를하자 atomic int 배열로 하면 되나? 잘 모르겠다 이부분은 질문을 해보는게 좋겠다 정리해보자
    -> 어짜피 메모리에 쓰이는 값은 1아님 0이다 이게 다중 스레드를 사용해 잘린다 해도 0아님 1인거다 우리 게임에서 필드 내에서 충돌이 크게 중요한가?
    크게 안중요하니 그냥 int로 하자

길찾기
  * 요구사항 : 클라이언트에서 선택한 셀로 플레이어를 이동시키자
  * 없는 길을 찾으라하면 터짐 -> 에이스타 시간이 길어지면서 그러는거군 step을 추가시키자
  구현해야할 사항
  - 서버 a*알고리즘
  - 클라에서 선택한 위치 패킷으로 전송
  고려 사항
  - a*를 사용해 이동하고 있을 때 이동을 하면 a* 종료
  - 이동 중 set pos패킷이 오면 목적지 변경
  - 이동 중 move 패킷이 오면 a*종료
  a* 종료는 dest가 -1이면 바로 종료하자

해당 파일은 여기서 보실 수 있습니다.
https://github.com/Juyeong11/Graduation_project/blob/main/BeatSlimeServer/Server%20%ED%95%B4%EC%95%BC%ED%95%A0%20%EC%9D%BC.txt

진행 결과

오랜 기간동안 팀원들과 프로젝트를 진행하며 협업에서
소통 능력과 협업 능력을 키울 수 있었습니다.



- 서버 제작 외 **이펙트 출력과 로봇 애니메이션 제작**을 하였습니다.
- 다중 접속 게임의 서버를 제작하며 멀티 스레드 환경을 코딩해보는 경험을 할 수 있었습니다.
- i5-10400 CPU 기준으로 1500명의 더미 플레이어의 요청을 100ms의 응답시간내에 처리할 수 있습니다.

Unreal4를 이용한 레이싱 게임 제작

평소 만들어보고 싶었던 드리프트를 구현해보기 위해

차량 물리에 대해 찾아보고 직접 차량 컴포넌트를 제작해 사용

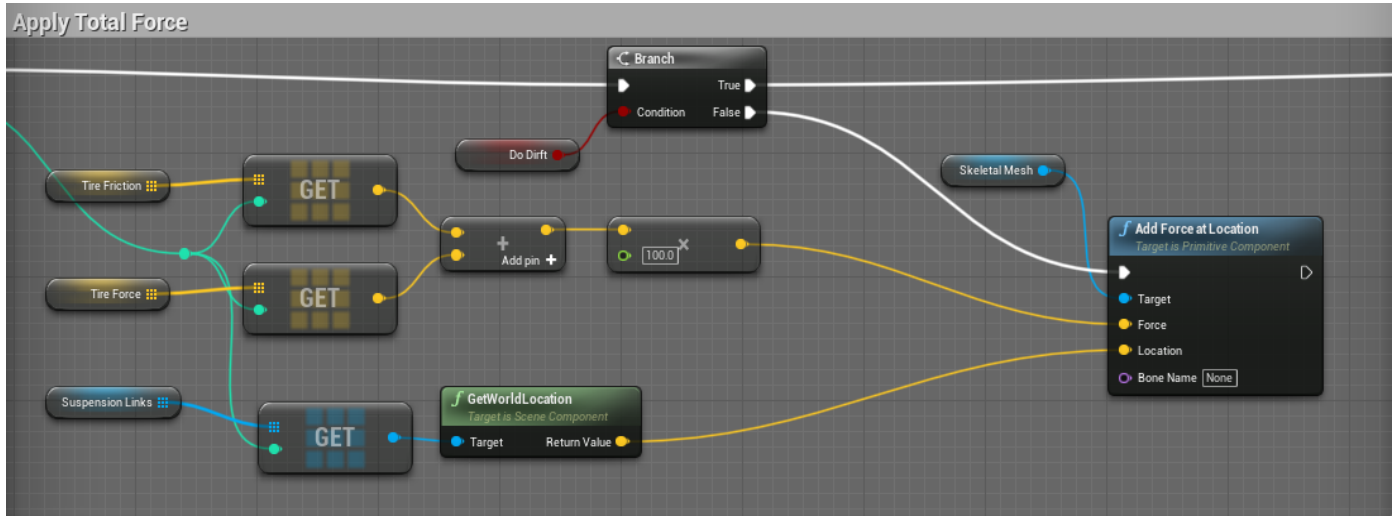
구현 사항

- 광선 충돌을 이용한 차량 서스펜션 구현
- 차량 물리량 계산해 이동 및 브레이크 구현
- Cascade Particle 시스템을 이용한 여러 파티클 제작
- 포스트 프로세싱과 스텐실 버퍼를 이용해 속도감을 위한 블러 효과 제작
- 스플라인을 따라가며 플레이어와 경주하는 AI
- 텍스쳐 uv좌표를 이용해 미니맵 및 차량 계기판 UI 제작
- 전남 영암에 있는 f1 서킷의 크기를 맞춰 기존 LandScape 에서 수정한 맵
- LandScape와 차량 모델은 언리얼 애셋을 사용하였습니다.

진행 과정

구현 과정

레이싱 게임에서 중요한 속도감과 조작감을 표현하기 위해 여러 게임을 찾아보며 만들었습니다.



타이어의 방향에 따른 추가된 힘의 방향, 차량 속도에 비례한 마찰력을 차량 매쉬에 전달하는 블루 프린트 노드입니다.

- 언리얼에서 제공해주는 Vehicle컴포넌트를 사용해 차량을 만들었지만 너무 사실적인 움직임과 런타임에 마찰 계수를 조정할 수 없는 문제 등 제약이 있어 해당 컴포넌트로는 레이싱 게임에서의 드리프트를 표현할 수 없을 것 같다 판단해 **직접 물리 계산을 통해 간단한 차량 엑터 클래스를 제작**하였습니다.

- 드리프트의 경우는 회전 시 차량의 무게중심을 옮기고 마찰력을 변경하는 등 다양한 방법을 시도해 보았지만 카트라이더와 같은 **드리프트를 구현하는데 실패해** 드리프트를 하면 계산되고 있는 물리 값을 무시하고 현재 차량의 진행 방향과 속도를 얻어와 원하는 방향으로 강제로 바꾸고 차량을 회전해 주는 방법을 사용했습니다.

해당 게임의 플레이 영상과 설명은 여기서 보실 수 있습니다.
https://youtu.be/I2YxXXBI4BA