|  |  |  |  |
| --- | --- | --- | --- |
| 도둑들 | | | |
| **5월 4주** | **2023. 5. 21 ~ 2023. 5. 27** | **작성자** | **정극훈** |
| **이번 주**  **목표** | **정극훈: UI는 forward쉐이더로 생성하도록 변경, FBX모델의 그림자 구현, 콜라이더 데이터를 기반으로 OBB 추가**  **이도영: OBB 오류 수정, 슬라이딩 벡터 추가**  **김혁동: 게임 로직 구현** | | |
| **5/21**  **일** | **이도영:** 충돌처리 오류해결   * 1. 오류내용   + 충돌되지만 충돌되지 않는 곳도 충돌이 되어진다. * 해결 방법   + center값만 받도록 하여 center값에 충돌되는 순간 충돌되어지는 position을 받아와 그 값을 collision data 와 일일이 비교를 해보니 collision data에 값이 없었음.   + 모든 collision data를 받은 것이 아니었기 때문에 충돌이 되지 않는 것이었음.   + 충돌은 잘 이루어지고 있었다. * 2. 오류내용   + 충돌하고자 하는 기물의 충돌이 완벽히 이루어지지는 않는다. * 해결 방법   + Collision data를 받을 때 default 크기가 달랐던 오류가 있었다.   + 받아온 extent 값에 1000배를 해야만 정확한 충돌이 이루어졌음.   + Extent 값을 조금씩 올리며 1000배에 해당하는 것을 확인한 후 충돌처리 오류를 해결함. | | |
| **5/22**  **월** | **김혁동:** 5/22 ~ 5/27  게임 시작 전, **로그인부터 방에 들어가 레디까지의 패킷을 제작**하고, 해당 패킷을 받았을 때 진행해야 할 프로세스들에 대해 작성함.  클라이언트에서 서버로 가는 프로토콜에는 다음과 같은 패킷들이 추가되었음.  **CS\_PACKET\_ENTER\_ROOM** -> 클라이언트가 방에 들어가는 것을 선택하였을 때에 주는 패킷. 서버에서는 이 패킷을 받으면 들어가려고 하는 룸의 ID가 실제로 들어갈 수 있는 상태에 있는지 확인한 뒤, 접속이 가능하면 해당 ID의 룸에 플레이어의 데이터를 추가하고 SC\_PACKET\_ENTER\_ROOM\_OK를 보내줌. 보내지지 않은 경우에는 SC\_PACKET\_ERROR를 통해 어떤 문제로 인해 접속이 불가했는지 알려주도록 하였음.  **CS\_PACKET\_LEAVE\_ROOM** -> 클라이언트가 방에서 나가는 경우 주는 패킷. 이 패킷을 받은 경우 해당 클라이언트가 들어가 있는 룸을 찾아 해당 룸에서 플레이어의 데이터를 제거해 줌.  **CS\_PACKET\_PLAYER\_READY** -> 클라이언트가 룸에서 Ready를 선택했을 때 보내는 패킷. 서버는 이 패킷을 받으면 룸에서 READY를 하고 있는 플레이어를 관리하는 컨테이너에 데이터를 넣어주도록 하였음.  **CS\_PACKET\_PLAYER\_CANCLE\_READY** -> 클라이언트가 룸에서 Ready 해제를 선택했을 때 보내는 패킷. 서버는 이 패킷을 받으면 룸에서 READY를 하고 있는 플레이어를 관리하는 컨테이너에서 데이터를 제거하도록 하였음.  **CS\_PACKET\_PLAYER\_LOG\_OUT** -> 클라이언트가 로그아웃을 한 뒤, 다시 다른 아이디로 로그인을 할 수 있기 때문에 추가한 패킷. 클라이언트의 상태를 Accept하였던 상태로 다시 초기화 한 뒤, SC\_PACKET\_PLAYER\_LOG\_OUT\_OK를 보내주도록 하였음.  **CS\_PACKET\_REQUEST\_ROOMS\_DATA\_FOR\_LOBBY** -> 클라이언트가 로비에 들어왔을 때, Lobby에서 보여줄 룸 들의 데이터를 받기 위한 패킷임. 서버는 이 패킷을 받으면 룸들의 데이터를 보내주도록 함.  **CS\_PACKET\_REQUEST\_ROOMS\_DATA\_FOR\_ROOM** -> 클라이언트가 룸에 들어왔을 때, Room에서 보여줄 플레이어들의 데이터를 받기 위한 패킷임. 서버는 이 패킷을 받으면 룸 안에 있는 플레이어들의 데이터를 보내주도록 하였음.  서버에서 클라이언트로 가는 프로토콜에는 다음과 같은 패킷이 추가되었음.  **SC\_PACKET\_ENTER\_ROOM** -> 클라이언트에서 Enter를 받았으면, 해당 룸에 있는 플레이어 들에게도 룸에 들어갔다는 것을 알려야 하므로 그 때 사용하는 패킷.  **SC\_PACKET\_ENTER\_ROOM\_OK** -> 클라이언트가 룸 입장에 성공했다는 것을 알려주는 패킷.  **SC\_PACKET\_LEAVE\_ROOM** -> 클라이언트가 룸에서 나간 것을 다른 플레이어 들에게도 알려 줘야 하므로 그 때 사용하는 패킷.  **SC\_PACKET\_PLAYER\_READY** -> 클라이언트가 Ready를 했으면, 다른 플레이어 들에게도 알려 줘야 하므로 그 때 사용하는 패킷.  **SC\_PACKET\_PLAYER\_CANCLE\_READY** -> 클라이언트가 Ready를 취소했으면, 다른 플레이어 들에게도 알려줘야 하므로 그 때 사용하는 패킷.  **SC\_PACKET\_LOG\_OUT\_OK** -> 로그아웃이 성공했는지 알려주는 패킷. 로그아웃 후 다른 아이디로 접속할 수도 있으므로 추가한 패킷임.  **SC\_PACKET\_ALL\_PLAYER\_LOAD\_END** -> 게임이 시작될 때, 모든 플레이어가 로딩이 끝난 것을 알려주는 패킷. 이 패킷을 보내게 되면 룸 안의 모든 플레이어가 게임에 접속하게 됨.  **SC\_PACKET\_ROOMS\_DATA\_FOR\_LOBBY** -> 클라이언트에게 룸 데이터 들의 요청을 받았을 때, 해당 데이터를 보내주는 패킷. 하나씩 보내주므로 WSASend가 많이 불려 느려질 수 있으므로 나중에 바꿔야 할 수도 있음.  **SC\_PACKET\_ROOMS\_DATA\_FOR\_LOBBY\_END** -> 하나씩 보내는 데이터이므로 데이터가 끝났음을 알려줘야 함.  **SC\_PACKET\_ROOMS\_DATA\_FOR\_ROOM** -> 위와 같이 클라이언트에게 룸 내부의 데이터의 요청을 받았을 때, 해당 데이터를 보내주는 패킷. 이 부분도 똑같이 WSASend가 많이 불려질 수 있음.  **SC\_PACKET\_ROOMS\_DATA\_FOR\_ROOM\_END** -> 마찬가지로 하나씩 보내는 데이터 이므로 데이터가 끝났음을 알려줘야 한다.  **SC\_PACKET\_ERROR** -> 무언가의 문제로 클라이언트가 하려고 했던 동작이 서버에서 문제가 발생하였을 때 보내는 패킷.  이외에도, 서버에서 Lobby를 추가하기 위하여, Lobby라는 클래스를 추가하여 플레이어의 데이터를 관리하도록 하였음. Lobby 클래스에서도 플레이어 데이터의 추가 및 삭제, 데이터 리셋 등의 처리를 할 수 있도록 해 두었음. 그리고 플레이어가 가질 수 있는 상태의 가짓수가 추가되었으므로, 따로 INROOM이라는 State와 INROOMREADY라는 State를 추가하였음.  또한, 클라이언트에서는 Room 상태와 Lobby상태를 표시할 수 있게 만들기 위해서 Scene 클래스를 상속받는 Room Scene과 Lobby Scene을 새로운 클래스로 만들었음. 이 부분은 각각 Room와 Lobby에서 관리해야 할 룸 정보와 플레이어 데이터를 저장하고 관리하기 위해 추가하였음.  LobbyScene에서 임시적으로 룸 접속 등을 하기 위하여, o 버튼을 누르면 빈 방, 숫자 버튼을 누르면 숫자에 맞는 ID의 방으로 들어가도록 설정하였음.  아직 RoomScene에 대한 작성이 완료되지 않았고, 게임이 시작되는 부분이나, 룸에 접속하고 퇴장할 때에 대한 관리가 완벽할 것 같지 않으므로, 해당 부분을 다시 한번 확인하여 문제가 없게 한 뒤, 게임이 시작한 이후의 게임 로직에 대해 처리를 해야할 것으로 보임. 또한, 도영이의 충돌체크가 완료되는 대로, 맵과 플레이어의 상호작용에 대한 처리도 진행할 수 있을 것으로 보임.  **이도영:** 맵 수정 및 슬라이딩 벡터 시작   * Map의 비어 보이는 이유는 “넣을 것만 넣었다.”이기 때문이라 생각하였습니다. * 손기훈 교수님의 기획 수업을 들었을 때 교수님의 말을 빌리면 “게임적 허용”이 필요하다. * 저희 게임은 꾸밈 공간이란 개념이 없습니다. 모든 곳이 플레이 공간이기 때문에 플레이만을 생각하여 맵이 비어 보이는 결과를 초래했다 생각하게 되었습니다. * 그렇기 때문에 플레이공간에 현실처럼 맵을 제작하는 것이 아닌 게임적 허용으로 플레이 공간을 더욱 채워야 한다. 이에 따른 맵을 제작을 다시 진행하며 레벨디자인에 관한 것은 게임내 플레이어의 자유도를 높이는 것으로 하기로 결론 지었습니다. * <원래 맵> * 현재 맵 상태에서 9구획으로 나눈다고 생각만 하였지 1구획안에서도 구획이 나뉠 필요가 있습니다. 맵이 큰 만큼 1구획을 어울리게 끔만 하는 것이 아니라 어울리지 않더라도 그 구획 안에서도 zone을 나눈다 생각하며 다시한번 제작하게 되었습니다. * 그 구획은 벽으로 아예 나누는 것이 아닌 에셋을 통해 나누며, 더욱더 복잡해지고 시야에도 방해가 되는 맵은 플레이어의 2단점프과 같은 이동을 추가하여 레벨디자인 측면에는 변화가 없도록 하였다. * 맵부터 만든 후 그에 따른 에셋을 찾는다 생각하며 제작하기로 마지막 결론을 내서 맵을 수정함. | | |
| **5/23**  **화** | **이도영:** 슬라이딩 벡터 코드 작성   * 슬라이딩 벡터.   + 슬라이딩 벡터는 플레이어가 예를 들어 벽과 충돌을 하였을 때 충돌한 지점에서 멈추는 것 이 외에도 플레이어의 운동하는 방향에 따라 벡터 값을 계산하여 이동하도록 하는 것입니다.   + 미끄러짐 벡터 ( Sliding Vector )   + 기본 공식에 관한 그림입니다. P가 플레이어이면 이에 대한 입사벡터 P를 n에 투영시켜 이에 관한 투영벡터를 얻게 됩니다. 이 때 얻은 투영 벡터를 우리의 이동방향 P에서 빼 주면 슬라이딩벡터 S를 얻을 수 있습니다.   + 이를 통해 얻은 슬라이딩 벡터 S가 벽을 따라 이동하게 됩니다.   + 이에 대한 공식은 S = P - n(P·n) 입니다.   + 플레이어가 이동할 때에 충돌되는 순간 슬라이딩 벡터를 계산하는 코드로 들어가도록 하였습니다. 이를 통해 sliding 벡터 pos를 얻으면 이 좌표를 그대로 플레이어의 position에 저장되도록 하였습니다.   + 플레이어가 이동하기전 좌표와 이동한 좌표를 통해 플레이어 이동방향에 따른 벡터를 구한 뒤 벽에 center 값과 충돌한 곳을 이용한 법선 벡터를 구해 공식에 넣어 슬라이딩 벡터를 구현하였다.   + 오류발생     - 오류내용     - 충돌을 하였을 때에 슬라이딩 벡터를 통해서 나오는 좌표가 잘못 표기되어 나옴.     - 오류 수정 내용     - 디버깅을 통해 멤버함수에 대입되는 값들을 살펴보았을 때는 잘못되지 않았기에 계산식에 문제점이 있다고 생각하여 계산식에 대해 살펴보았으나 계산식도 공식 그대로 코드로 옮긴 것 이기에 문제점이 없다고 생각하였다.     - 그래서 들어가는 값이 아니라 근본적으로 공식에 대입할 때 처음부터 잘못 생각한 것이 있는지에 대해 보게 되었다. | | |
| **5/24**  **수** | **정극훈:** UI는 Deferred쉐이더로 생성할 필요가 없으므로 Forward쉐이더로 생성하도록 변경했다. Resources class에 여러 쉐이더들이 미리 만들어져 있으므로 로그인화면에서 사용하는 UI의 쉐이더를 바꾸는 것은 간단했다.  텍스트, 스크린샷, 디스플레이이(가) 표시된 사진  자동 생성된 설명 | | |
| **5/25**  **목** | **이도영:** 슬라이딩 벡터 오류발생   * 23일에 계속 보았던 오류에 대해 보니 충돌 지점에 대한 법선 벡터를 구해야 되는데 충돌한 기물의 center값에 대해 법선 벡터를 구하고 있었다. 다시 말하면 충돌지점을 맞추지 않아 나오는 좌표 값이 바뀌게 된 것이다. 이를 해결하기 위해 충돌한 지점과 충돌 법선 벡터를 구해서 다시 대입하여야 한다. * 이때에 플레이어의 속도 벡터를 기반으로 슬라이딩 벡터를 구하는 것이 가능하다. * 계산식들이 멤버함수로 너무 많이 생기는 것 같아 연산자를 통해 vector에 추가하며 충돌 벡터의 코드를 리팩토링하게 됨. * 이를 통한 입사벡터와 법선 벡터를 구함으로써 더욱 직관적이고 효율적인 코드 작성 * 아직도 슬라이딩 벡터의 오류가 발생하는 중   + 오류내용   + 부딪치면 y 좌표가 자꾸만 줄어드는 오류가 발생   + 이 또한 대입하는 것에 대한 문제라 생각하고 이 오류에 대한 수정을 하였으나 제대로 동작하지 않음.   **정극훈:** OBB를 시각적으로 표현하기 위한 사전작업으로 CreateDefaultMaterial에 바운딩박스를 와이어프레임으로 생성하는 마테리얼을 추가했다.  스크린샷, 3D 모델링, PC 게임, 비디오 게임 소프트웨어이(가) 표시된 사진  자동 생성된 설명 | | |
| **5/26**  **금** | **정극훈:** FBX모델의 그림자 구현을 위해 기존의 그림자 생성 코드를 전체적으로 살펴봤다. 우선 FBX를 사용하지 않는 객체의 경우 정상적으로 그림자가 출력되는 것을 확인했다. 그림자 생성 코드는 문제가 없다고 판단해서 FBX모델이 플레인 객체 위에서 그림자가 생기는지 확인했다.  처음에는 그림자가 생성되지 않아서 FBX모델의 경우 그림자를 생성하지 못하는 건가 고민했었는데 그림자가 생성됬을 때 그림자를 보기 편하게 Directional Light의 밝기를 줄였는데 플레인 객체 위에 FBX모델의 그림자가 생성되는 것을 확인했다. Directional Light의 밝기를 최소로 하고 확인하니 그림자가 더욱 선명하게 보였다. 다행이도 코드에 문제가 있던 것이 아니라 빛의 밝기가 너무 강해서 그림자가 거의 보이지 않는 것을 그림자가 생성되지 않았다고 착각한 것이였다.  그림자가 생성되는 것을 확인했으니 맵 위에 그림자가 정상적으로 생성되는지 확인했다. 확인결과 플레인 객체 위에서 생성됬을 때와 똑같은 결과가 출력됐다.  **실루엣이(가) 표시된 사진  중간 신뢰도로 자동 생성된 설명**  처음에는 그림자가 생성되지 않아서 FBX모델의 경우 그림자를 생성하지 못하는 건가 고민했었는데 그림자가 생성됬을 때 그림자를 보기 편하게 Directional Light의 밝기를 줄였는데 플레인 객체 위에 FBX모델의 그림자가 생성되는 것을 확인했다. Directional Light의 밝기를 최소로 하고 확인하니 그림자가 더욱 선명하게 보였다. 다행이도 코드에 문제가 있던 것이 아니라 빛의 밝기가 너무 강해서 그림자가 거의 보이지 않는 것을 그림자가 생성되지 않았다고 착각한 것이였다.  그림자가 생성되는 것을 확인했으니 맵 위에 그림자가 정상적으로 생성되는지 확인했다. 확인결과 플레인 객체 위에서 생성됬을 때와 똑같은 결과가 출력됐다.  그림자는 출력은 되는데 두 가지 문제가 생겼다.   1. 그림자가 일정범위 내에서만 생성된다. 2. 그림자의 모양이 FBX모델의 기본모션으로 고정되어있다.   첫 번째 문제의 경우 그림자를 보여주는 쉐도우카메라의 FOV가 작아서 발생한 문제로 추정된다. 쉐도우카메라의 FOV값을 키웠더니 그림자가 생성되는 범위가 확연히 늘어났다. 하지만 그림자의 선명도가 떨어지는 문제가 발생했다. 이 부분은 FOV를 키웠으니 당연히 따라오는 문제였지만 그림자가 생성되는 범위가 늘어났을 뿐이지 생성범위 밖에서는 그림자가 짤리는 문제는 그대로이다. 일단은 당장 해결해야 하는 문제는 아니므로 추후에 해결방법을 찾아보기로 결정했다.  두 번째 문제의 경우 그림자의 형태가 도둑의 애니메이션과는 별개로 도둑의 FBX모델의 기본자세로만 출력된다. 이 문제는 어느 부분이 문제인지 도저히 예상이 안가는 문제다. 그림자가 Idle상태의 형태로 출력됐다면 쉐도우카메라가 애니메이션이 바뀌는 것을 몰라서 발생한 문제라도 추측할 수 있었겠지만 기본상태의 형태에서 도둑의 애니메이션을 아무리 바꿔봐도 그림자가 바뀌지 않았다. 이 부분은 김혁동과 문제를 공유해서 해결하기로 생각했다.  **이도영:** 슬라이딩 벡터 오류 및 레이 캐스팅 코드  슬라이딩 벡터의 오류는 결론적으로 법선벡터의 문제인것으로 드러남.  친구들과 얘기하면 이에 대한 문제점을 파악했으면 오류수정은 본인이 한다 함.   * 레이캐스팅 * 출발점과 방향을 정의 * 레이를 씬안으로 쏨 이때 레이는 주로 3D 공간에서 직선 형태로 이동한다. * 레이와 씬 안의 객체들과의 교차점을 검사, 레이가 씬 안의 객체와 교차하는지 어떤 객체와 교차하는지 판별 * 해당 정보를 활용하여 작업을 수행.   이러한 순서로 레이캐스팅이 이루어짐 총알 발사할때에 레이캐스팅을 이용할 계획이라 미리 레이캐스팅에 대해 학습하며 기본 알고리즘을 작성 | | |
| **5/27**  **토** | **도둑들 주간회의**  충돌처리 진행상황 확인 및 차후 계획 회의  김혁동의 게임 로직 구현 상황 확인  2주간의 진행상황과 계획서의 내용을 비교  그림자 문제 공유 | | |
| **이번 주 문제점 및 해결방안** | **클라이언트: 하나의 FBX 메쉬를 여러 개 생성할 때에, 생성되지 않는 문제 수정중**  **도둑의 그림자의 형태가 변하지 않는 문제** | | |
| **다음 주 계획** | **정극훈: 그림자 문제 수정, OBB의 시각적 구현 완성, 클라이언트 문제 수정**  **이도영: 충돌처리 오류 수정 및 Astar 코드 구현**  **김혁동: 게임 로직 구현** | | |
| **비고** | Github 도둑들 주소: <https://github.com/rmrgns/Thieves.git>  Github 졸업작품 회의록 주소: <https://github.com/rmrgns/gameproject_proceedings.git> | | |