|  |  |  |  |
| --- | --- | --- | --- |
| 도둑들 | | | |
| **4월 4주** | **2023. 4. 23 ~ 2023. 4. 29** | **작성자** | **정극훈** |
| **이번 주**  **목표** | **김혁동: 에셋 맵 제작, 맵 데이터 서버에 전달, 서버 도와주기**  **정극훈: 플레이어의 행동에 따른 애니메이션 출력, 로딩한 애니메이션을 여러 객체에 적용, 공격버튼을 추가하고 공격시 파티클 출력**  **이도영: MOVE 패킷 동기화 완료, 맵 제작 종료 시 충돌체크**  **서버 오류코드 변경** | | |
| **4/23**  **일** | **김혁동:** 맵 에셋을 구매함. 기존에 3개의 후보군이 있었으나, 기획을 담당한 도영이와 이야기 해본 결과, 하나는 지원받을 수 있는 금액 이상의 금액을 요구하였고, 다른 하나는 내용물이 부실한 것으로 보이므로 나머지 하나의 후보를 선택하여 구매함.    <실제 구매한 에셋>  구매한 에셋을 바탕으로 맵 제작을 시작하였음.  유니티에서 제작하여 Export 한 뒤, 블렌더에서 병합하여 FBX 데이터를 넣을 예정이었으나, 이 에셋에서 바닥과 벽이 5m x 5m인 Prefab 밖에 없다는 것을 알게 됨. 우리 게임은 맵의 크기를 300m x 300m 를 기준으로 만들려고 했기 때문에, 한 번에 엄청나게 많은 양의 바닥과 벽을 넣어야 함. 이렇게 되면 각각의 바닥과 벽이 적어도 2개의 폴리곤을 가지게 되고, 바닥과 벽 뿐만이 아닌 많은 양의 모델을 넣어야 하는데 바닥과 벽에 많은 폴리곤을 할애할 수 없다는 생각이 들어 직접 제작하기로 마음먹음.  로고이(가) 표시된 사진  자동 생성된 설명  <에셋에 있는 바닥. 5m x 5m의 크기를 가지고 있음.>  혹시 몰라서 실제로 100m x 100m의 바닥만 배치한 이후 블렌더로 가져가 데이터를 병합한 결과 13600 폴리곤이라는 엄청나게 많은 수의 폴리곤 수가 되는 것을 볼 수 있었음. 그 이외에도 정점의 노말이 이상하게 추출되는 현상도 있어 이 방식은 시간이 많이 걸릴 것으로 보였음.  텍스트, 바닥, 타일을 붙인, 보도이(가) 표시된 사진  자동 생성된 설명  <100m x 100m 크기의 바닥>    <Tris가 폴리곤 수. 엄청나게 많은 폴리곤이 할애되게 된다.>  **정극훈:** 드디어 인풋렉 문제를 해결했다. 다른 졸작팀은 어떻게 입력을 받나 확인했더니 GetAsyncKeyState를 사용해서 입력을 받고 있었다. 이 함수는 키보드 입력이 발생하면 메시지큐를 거치지 않고 바로 해당 키가 다운되었다고 알려준다. 도둑들 코드에서는 GetKeyBoardState는 모든 가상키의 상태 데이터를 복사해서 반환해주는데 이 경우에 키보드의 입력과 마우스의 입력을 동시에 복사해서 반환한다. 그런데 SetCursorPos가 실행될 경우 키보드와 마우스의 입력이 전부 지연된다. 그래서 인풋렉 문제가 발생했던 것이다. 이 GetKeyBoardState를 GetAsyncKeyState로 바꾸었더니 인풋렉이 사라졌다. GetAsynceKeyState는 입력이 발생할 때마다 바로 상태 데이터를 반환하기 때문에 중간에 SetCursorPos가 실행되더라도 키보드 입력이 밀리지 않는다. 정말 예상치 못하게 해결방법이 나왔다.  그래도 임시방편이 아닌 완전한 해결책이 나왔으므로 인풋렉 문제는 더 이상 신경 쓰지 않아도 될 것이다. 아마 이부분에 문제가 다시 생긴다면 아마 최적화 문제가 되겠지만 아직까진 신경 쓰지 않아도 될 부분이다. | | |
| **4/24**  **월** | **김혁동:** 서버 관련해서 도영이의 요청으로 팀원이 같이 모여 디버깅을 했음. 패킷을 전달받은 이후, 데이터를 넣어주지만 막상 그 데이터를 활용하려고 하는 순간 데이터가 기존 값으로 초기화되는 문제가 있음.  텍스트이(가) 표시된 사진  자동 생성된 설명  디버깅을 하던 도중, this로 받아온 패킷 전달용 변수와 엔진에서 함수를 통해 받아오는 패킷 전달용 변수의 포인터 주소가 다른 것을 확인함.  (this -> 0x000002b62297c9b0, 함수 -> 0x000002b605c89350)  같은 데이터가 두 개의 주소를 가지는 경우는 어떤 방식으로든 여러 번 만들어졌다는 것이고, 기본적으로 해당 클래스의 변수를 저장할 때에 ComPtr, shared\_ptr이나 unique\_ptr의 형태로 저장했기 때문에 패킷 전달용 클래스가 두 개 이상 생성되고 있을 확률이 굉장히 높았음. 이를 바탕으로 코드를 뒤져 본 결과, 오류가 생기는 이유를 확인할 수 있었음.  해당 오류는 실제로 데이터를 전달받는 곳과, 해당 데이터를 가져오려는 곳이 달라서 생기는 오류였음. 실제로, 가장 처음 게임을 실행할 때에 생성하는 Network 라는 클래스에서 가져오는 패킷 전달용 클래스와 프레임워크를 관장하는 Engine 클래스에 멤버 변수로 지정된 패킷 전달 변수가 있었고, 실제로 네트워킹을 통한 패킷 데이터는 Network 클래스 측에서 받고 전달해 주었으나 클라이언트에서 해당 데이터를 사용하려고 할 때에는 Engine 클래스에 선언된 데이터를 사용하였기 때문에 데이터에 문제가 있었던 것임.    <Network 클래스에 선언된 패킷 전달용 멤버 변수>    <Engine 클래스에 선언된 패킷 전달용 멤버 변수>  해당 오류는 서버의 구조 전반적으로 문제가 있었으므로, 서버 담당인 도영이에게 이 부분에 대한 수정을 요청함.  **이도영:**  Client와 Engine 두개의 프로젝트에 나누어져 있는 서버 관련 코드들을 Engine 프로젝트에 통합.  통합한 이유는 server에서 data를 send 하고 recv 할 때에 Client 프로젝트에서 recv 한 뒤 패킷을 분해하여, 패킷에 맞는 Engine안의 객체로 가서 실행하게 되는데, 이 과정에서 코드의 오류가 발생할 확률이 높았기 때문에 통합하려 함.  move에서 한 오류를 고치기 위해 시간투자가 오래 걸렸는데 Client 프로젝트와 Engine에서 공통된 객체를 선언된 shared\_ptr 과 unique\_ptr이 있었고 객체가 두개 만들어지고 말았습니다. server에서 client로 데이터를 보내고 이 데이터를 뺴서 쓰려고 해도, client에서 저장된 객체가 아닌 engine 프로젝트에서 선언된 객체에서 데이터를 빼 쓰려다 보니 move가 실행되지 않는 오류가 발생했습니다. 결국 오류를 찾긴 했지만, 프로젝트가 달라 디버깅에도 많은 어려움을 겪었기에 이번 기회에 Client와 Engine을 합치게 되었습니다.  이 과정에서 include 경로 오류가 발생하였고 이 경로 오류는 서로 다르게 포함된 경로에서 같은 이름의 폴더가 있어 발생한 오류였습니다. 폴더의 이름을 변경한 뒤 다시 빌드하니 오류는 해결되었습니다. | | |
| **4/25**  **화** | **정극훈:** 인스턴스 생성을 위해 이전에 사용하던 방법으로 경찰 캐릭터를 생성했다. 그런데 경찰 캐릭터를 2개 생성했더니 캐릭터가 화면에 출력되지 않았다.    현재 캐릭터는 SceneManager의 LoadScene에서   1. LoadFBX로 meshData에 FBX 데이터는 넣는다. 2. meshData를 기본으로 Transform과 Animation과 같은 기본적으로 들어가야 할 컴포넌트를 Instantiate로 추가해서 gameObjects를 만든다. 3. 밑에서 나머지 컴포넌트나 스크립트를 추가한다.   이와 같은 방식으로 만드는데 처음에는 LoadFBX를 중복으로 실행해서 발생하는 문제인지 생각해보았다. 그런데 LoadFBX를 안하면 FBX데이터를 못 가져오므로 이 경우는 문제가 아니였다. 나머지 부분은 캐릭터를 생성하고 컴포넌트를 추가하는 부분이므로 역시 중복으로 생성하는 문제는 아니다. 그래서 Instantiate 함수를 확인해봤다.    Transform과 MeshRenderer, Animator 컴포넌트를 추가하고 MeshRenderer에 마테리얼을 설정한다.  이 부분에서는 FBX 데이터를 건드리는 부분은 없고 Transform이나 Animation에서 문제가 발생하지 않았으므로 Instantiate 함수도 문제가 없는 것으로 확인했다.  그렇다면 LoadFBX에서 뭔가 문제가 있는 것 같다. 이 부분은 다음에 확인할려고 한다.  **이도영:**  점프를 할 시에 부자연스럽게 점프가 되어지는 오류가 발생함.  Server에서 지속적으로 데이터를 보내게 되는데 Jump를 할 시 이동 되어지는 y값이 move패킷를 통해 받는 y값한테 잡아 먹히게 되어 강제로 y값이 변경되는 오류 였습니다.  move패킷에서 w a s d 이동일 때에는 y 좌표 값을 삭제함. 점프를 계산하는 계산식을 client 로 옮긴 뒤 점프할 시 플레이어의 상태를 나타내는 값을 보내 id 값에 따라 점프를 하는지 안 하는지 판단만 하게 하여 점프의 오류를 수정하였습니다. 추후 Client 담당 팀원이 Jump의 알고리즘을 변경하면 다시 서버로 가져오기로 하였습니다.  Client 가 Disconect 해도 server의 id에는 값이 남아 있고 다른 id 값의 변화가 일어나면 지속적으로 변화된 값을 disconnect 된 id에 보내주려 함.  Client에서 게임을 종료해도 Disconnect 하지 않아 지속 적으로 없는 아이디 값을 찾아 데이터를 저장하려 함. Disconnect하면 자료구조안에 저장된 데이터를 삭제한 뒤 reset 하여 free 상태이니 다른 id 값이 들어와도 된다 라는 표시를 하려함.  ESC를 누를 때에 Disconnect를 통해 room의 obj\_list에서 삭제한 후 ST\_FREE TYPE을 저장시켜 준다.  ST\_FREE는 현재 비어 있다는 뜻으로 게임중인 list는 ST\_INGAME으로 나타내어 준다. | | |
| **4/26**  **수** | **김혁동:**  텍스트이(가) 표시된 사진  자동 생성된 설명  기본적인 벽과 바닥에 대한 모델링이 끝남. 그러나 아직 텍스쳐가 마련되지 않았으므로 직접 그리기 위해 기획자인 도영이와 상의가 필요함.  도영이의 말로는 바닥은 회색 바닥에 벽은 아이보리의 느낌을 원하는 것 같았으므로 그렇게 작업하기로 함. 그렇게 텍스쳐를 입히고 유니티에 옮김.    이제부터 이 곳에 구매한 에셋들을 채워 넣어야 하나, 도영이가 만든 맵 구조를 보았을 때에, 에셋들의 크기가 많이 작아 보이므로 이 부분을 도영이와 상의하고 나서 추가하기로 하였음. 그리고 에셋 배치 작업을 하기 전에 이 상황에서 콜라이더를 외부로 빼내는 작업을 해보는 것도 나쁘지 않아 보임.  도영이와 상의를 한 결과, 맵의 크기가 너무 커 기획대로 진행하면 너무 비어 보일 수 있다는 이야기가 나왔음. 그래서 일단은 유니티에서 확인한 결과 180m x 180m의 크기로 진행을 하고, 나중에 크기를 줄이거나 늘리면서 생각을 해보자는 이야기를 하게 되었음. 또한 대략적인 각각의 구역의 컨셉에 대해 이야기를 나누었고, 각각의 구역에 대해 카페, 의류점, 오락실 등의 컨셉을 가지고 만들어 가기로 결정함.  유니티에서 콜라이더 데이터를 추출하기 위해, 유니티에서의 파일 입출력에 대해서 알아보기 시작함. 유니티의 파일 입출력은 System.IO에 들어있는 FileStream을 통해 가능하다고 함. 우리 게임의 경우 OBB형식의 박스 콜라이더만을 사용하여 충분히 모든 충돌을 구현할 수 있을 것으로 보이므로, 신 내부에 있는 모든 박스 콜라이더를 가져와 해당 콜라이더의 센터와 스케일 등을 가져오면 될 것으로 보임.  콜라이더 데이터를 추출할 수 있게 됨. 기본적으로 콜라이더로부터 center와 size 값을 받아와 저장하는 형식으로 진행됨. 4개의 콜라이더를 추가해 추출을 진행해본 결과 정상적으로 진행되는 것을 확인하였음.  텍스트이(가) 표시된 사진  자동 생성된 설명  <콜라이더 데이터 추출 코드>  테이블이(가) 표시된 사진  자동 생성된 설명<추출된 콜라이더 데이터>  그러나, 이런 방식으로 진행하게 되면 콜라이더는 해당 콜라이더 컴포넌트를 가지는 게임오브젝트 자기 자신의 Position을 기준으로 center와 size를 정하기 때문에 콜라이더를 삽입할 때에 0, 0, 0의 위치에서 콜라이더만 옮기는 형식으로 진행해야 하기 때문에 시간이 걸릴 수 있을 것 같음. 그러므로 다른 방식을 택하는 것이 좋아보임. 아마도 유니티 엔진 내부에 콜라이더의 월드 기준 중심 좌표를 얻을 수 있는 방법이 있을 것임.  유니티에서 월드 공간에서 해당 콜라이더가 차지하는 부분에 대한 데이터는 ‘bounds’라는 곳에 저장되어 있는 것으로 보임. 그러므로 콜라이더의 bounds를 통해 실제 좌표를 가져오는 방식으로 다시 한번 코드를 작성하기로 함. 이 부분은 이전 코드에 “bounds.”을 추가해주는 것으로 바로 구현할 수 있으므로 바로 변경하여 데이터를 확인했음.  텍스트이(가) 표시된 사진  자동 생성된 설명  <변경한 코드>  텍스트이(가) 표시된 사진  자동 생성된 설명  <실제 월드 좌표에 맞춰 변환된 데이터>  이것으로 콜라이더 데이터를 추출하는 것은 끝났으므로, 맵을 만들고 해당 맵에 콜라이더를 추가하기만 하면 될 것으로 보임.  유니티에서 반 정도 완성한 맵을 문제가 있을 확률이 있다고 판단하여 블렌더에서 하나로 합쳐 실제 졸업작품 데이터에 적용시켜 보았다. 크게 문제는 없는 것으로 보이므로 이 상태로 진행하면 될 것으로 보인다.  <맵이 제대로 표시 됨>  그러나 생각보다 에셋들의 크기가 큰 문제가 있음. 이정도면 사람의 크기를 키우던가 에셋의 크기를 줄여야 하는데, 에셋의 크기를 줄이게 되면 구역이 너무 비어 보일 수 있는 문제가 있을 수 있고, 사람의 크기를 키우면 본래 계획했던 기획과 다른 부분이 생기게 되어 이 부분도 기획자에게 물어봐야 할 것 같음.  **이도영:**  AI 관련 코드 추가 MoveObj를 상속 받는 Enemy 객체에서 AI를 관리하게 된다. Timer에 의해서 움직이게 되며 init할 때에는 type room\_id, SetPos, 값을 얻게 된다.  멤버함수는 SetSpawn(), Reset(), MoveTime(), DoMove(), DoPrecvMove(),가 있으며 AI의 초기화 무브 타임 관련 함수 전 이동 관련함수, 스폰 위치 설정 맵과 함께 충돌체크 데이터를 받게 되면 충돌체크 까지 추가할 예정 | | |
| **4/27**  **목** | **김혁동:** 맵을 전부 제작 완료함. 콜라이더 데이터도 게임 스타트만 누르면 바로 export되므로 실질적으로 맵은 완성됨. 그러나, 가운데에 다이아몬드를 어떻게 처리할 지 몰라서 이 부분도 기획자에게 물어봐야 함.**텍스트, 교통, 배, 선박이(가) 표시된 사진  자동 생성된 설명**  **정극훈:** 저번에 미뤄뒀던 LoadFBX를 확인했다.    FBX가 있는 경로를 path로 받고 해당 경로의 메쉬데이터가 있는지 확인한 후 있다면 그 데이터를 그대로 리턴, 없다면 해당 경로의 FBX 데이터를 메쉬데이터에 넣고 Add함수로 meshData를 추가한 후 데이터를 리턴한다.  여기서 아래쪽에 메쉬데이터를 추가하는 부분은 캐릭터를 한 개만 생성할 때에 정상적으로 출력됐으므로 문제가 아니라는 것을 확인했고 이 부분에서 if문 부분을 제외했더니 일단은 캐릭터가 정상적으로 출력됐다. 하지만 Instance 생성을 위해 Add로 처음 생성한 메쉬데이터를 저장하고 해당 데이터를 재사용할 때 Load를 하지 않고 저장한 데이터를 바로 쓰기 위해 존재하는 코드인데 여기서 문제가 발생했다는 것은 저장을 제대로 못하거나 저장된 데이터를 제대로 가져오지 못해서 발생하는 문제인 것으로 추측된다. 일단은 서버에서 여러 명의 유저가 서로의 캐릭터를 확인할 수 있는 코드가 필요했기에 if문 부분을 제외하고 Instance를 사용하지 않고 캐릭터를 사용하는 쪽으로 코드를 바꾸었다.  **이도영:**  중간발표에 사용할 PPT를 제작함. | | |
| **4/28**  **금** | **정극훈:** 우선 애니메이션 출력코드를 살펴보았다. 이전에 드래곤에 애니메이션을 넣었을 때 TestDragon 스크립트에 숫자키 입력에 따른 애니메이션 출력 코드를 구현했었다.  해당 코드를 보니 애니메이션 출력 코드 자체는 매우 간단했다.    인덱스 번호를 받으면 해당번호에 맞는 애니메이션을 출력한다.  나는 몇번 인덱스에 어떤 애니메이션이 있는지 알고 있으므로 인덱스를 따로 계산해서 넣지 않고 해당 인덱스번호를 바로 넣어서 애니메이션을 실행할 것이다.  우선 플레이어가 공격할 때 관련된 기능들을 수행하는 PlayerAttack 함수를 만들었다.  지금은 공격할 때 공격 애니메이션을 출력하는 코드를 넣었다. 처음에 단순히 애니메이션을 실행하는 코드만을 넣으니 공격을 하는 마우스 좌클릭을 눌렀을 때 공격 애니메이션의 종료와 상관없이 항상 공격 애니메이션을 처음부터 실행했다.  내가 원하는 것은 공격이 끝날 때까지 다른 애니메이션이 재실행되지 않도록 하는 것이므로 공격의 상태를 나타내는 \_attackState 변수를 추가했다.  공격버튼을 눌렀을 때와 공격 중인 상태를 구분해서 공격 중에 다시 공격을 시도해도 애니메이션이 다시 나오지 않도록 만들었다. | | |
| **4/29**  **토** | **김혁동:** FBX를 인스턴싱할 때에 문제를 발견함. 하나의 FBX 메쉬를 여러 개 생성할 때에, 생성되지 않는 문제를 발견하였음. 해당 메쉬는 다른 메쉬를 만들었을 때에 그 위치에 생성되는 것을 확인하였음.  텍스트, 밤하늘이(가) 표시된 사진  자동 생성된 설명  <같은 메쉬의 FBX를 여러 개 만들었을 때에 모델이 표시되지 않음>  텍스트이(가) 표시된 사진  자동 생성된 설명  <경찰 FBX를 추가하는 경우 여러 개 만든 도둑이 추가됨 (같은 위치이므로 하나인지 여러 개 인지 확인은 불가능.)>  임시적으로 같은 FBX 데이터를 여러 개로 만들어 각각의 FBX를 읽어오는 방법으로 진행하기로 하였음. 이 상태로 진행하면서 FBX 인스턴싱에 대한 디버그도 진행할 예정임.  **정극훈:** 저번에 공격 애니메이션을 출력하는 코드를 구현했으니 오늘은 나머지 애니메이션을 캐릭터의 움직임에 맞게 출력하고 서로 충돌하지 않도록 만들었다.    \_moveState변수는 캐릭터가 이동할 때 true, 멈출 때는 false가 된다. \_moveState가 false이고 \_attackState가 0일때 정지상태의 애니메이션을 출력하도록 하였다.    이동할 때는 이동버튼이 눌렸을 때 이동상태의 애니메이션이 출력되지만 공격 중일 땐 공격 애니메이션이 출력될 때까지 이동 애니메이션이 출력되지 않도록 하였다.    점프의 경우에도 이전에 만들었던 공격과 동일한 방식으로 구현하였다.  경찰의 애니메이션의 경우 모션은 도둑과 동일하므로 인덱스만 조정하면 동일한 코드로 경찰의 애니메이션도 구현이 가능하다. | | |
| **이번 주 문제점 및 해결방안** | **클라이언트: 하나의 FBX 메쉬를 여러 개 생성할 때에, 생성되지 않는 문제를 발견하였음.** | | |
| **다음 주 계획** | **김혁동: FBX 인스턴싱 문제 해결**  **이도영: 서버 동기화**  **정극훈: 파티클 코드 개선, FBX 인스턴싱 문제 해결** | | |
| **비고** | Github 도둑들 주소: <https://github.com/rmrgns/Thieves.git>  Github 졸업작품 회의록 주소: <https://github.com/rmrgns/gameproject_proceedings.git> | | |