|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **주차** | 11주차 | **기간** | 2024.12.10  ~2024.12.24 | **지도교수** | 이형구 교수님 |
| 이번주 한일 요약 | 10주차: 캐릭터 업데이트를 비동기에서 tick으로 바꾸기  11주차: 여러명의 클라이언트 받아서 화면에 출력하기 | | | | |

<10주차 작업>

#include "Containers/Queue.h"  
TQueue<FCharacterState> ReceivedDataQueue;  
언리얼에서 사용하는 queue를 사용하여 이 큐에 플레이어   
이동들을 저장해서 tick에서 사용한다.

receiveData에서 받은 데이터를  
ReceivedDataQueue.Enqueue(\*ReceivedState); 로 큐에 저장하도록 바꾸고

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

tick에서 queue에 있는 데이터로 캐릭터를 업데이트 한다.

하지만 이 작업은 다음주에 여러 클라이언트를 관리하느라 다시 비동기로 바뀌게 되며 쓸모없게 되었다.

<11주차 작업>

이제 멀티를 지원하기 위해 여러명의 클라이언트를 접속시키는 작업을 해야한다.

TArray<SOCKET> ClientSockets; // 여러 클라이언트 소켓 관리   
TMap<SOCKET, AClientCharacter\*> ClientCharacters; // 클라이언트별 캐릭터 관리   
FCriticalSection ClientSocketsMutex; // 소켓 리스트에 대한 스레드 안전성 보장

멀티를 지원하기 위해 여러명의 데이터를 저장하기 위해 위 변수들을 추가 해준다.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

그리고 이제 데이터를 받거나 업데이트 하는 것을 소켓을 받아서 해당 소켓에 해당하는 캐릭터를 업데이트 하도록 한다.

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

바뀐 점은 Accept할때 받은 클라이언트 정보를 배열에 저장한다.

{ FScopeLock Lock(&ClientSocketsMutex); ClientSockets.Add(NewClientSocket); }  
이렇게 Lock을 중괄호로 묶으면 동기화를 위해 접근을 막는 행동이 저 범위 안에서만 행해진다고 한다.

Accpet에서 받은 인자로 ReceiveData에서 해당 클라이언트를 업데이트하는  
SpawnOrUpdateClientCharacter를 호출한다. 실패할 경우 새로생긴 CloseClientSocket를 호출한다.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

CloseClientSocket에서는 그 클라이언트에 관련된 정보와 소켓을 지운다.

SpawnOrUpdateClientCharacter에서 바뀐것도 인자를 받은 해당 클라이언트의 캐릭터를 움직이게 한 것 밖에 없다.

<결과물>구름, 하늘, 조각상, 야외이(가) 표시된 사진

자동 생성된 설명

이렇게 여러명의 클라이언트가 접속하면 여러 개의 캐릭터가 생성되고, 각각의 움직임을 받아서 화면에 출력하게 된다.

|  |  |  |  |
| --- | --- | --- | --- |
| **문제점 정리** | 클라이언트간에 겹치기가 되는거 보니 충돌처리가 안되고 있나보다. | | |
| **해결방안** | 충돌처리 관련해서 찾아서 추가해보기 | | |
| **다음주차** | 12주차 | **다음기간** | 2024.12.24  ~2024.12.31 |
| **다음주 할 일** | 충돌처리 공부하기 | | |
| **지도 교수**  **Comment** |  | | |