|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **주차** | 16주차 | **기간** | 2023.10.17~  2023.10.23 | **지도교수** | (서명) |
| 이번주 한일 요약 | 게임 서버 프로그래밍 교과서 책 마지막 10장 읽기  프로토타입 게임 외부에서 접속 가능하게 해결하기 | | | | |

<상세 수행내용>

게임 서버 프로그래밍 교과서 책 10장 읽었음. 분산 서버 구조 사례에 대한 것이었음.

프로토타입에서 서버접속을 위한 UI을 제작함

텍스트, 스크린샷, 소프트웨어, 디자인이(가) 표시된 사진

자동 생성된 설명

Host를 누르면 실행중인 컴퓨터에서 서버를 열게 됨. 포트번호는 7777 고정

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

Join을 누르면 주소를 적는 창이 나오고 입력하여 Enter를 누르면 주소에 맞는 서버에 접속하게 됨. (유튜브 영상에선 127.0.0.1을 입력하였음. 내 컴퓨터 Public IP로 접속가능한 것을 확인함)

스크린샷, 만화 영화, 3D 모델링, 애니메이션이(가) 표시된 사진

자동 생성된 설명

ESC를 눌렀을 시 뜨는 UI. 아직 창 모드를 윈도우와 전체화면 바꾸는 것만 구현하였고, 서버 나가기와 게임 종료 버튼이 구현되어있음.

추가로 다른 서버 책 ‘멀티플레이어 게임 프로그래밍’을 읽으며 간단히 직렬화에 대해서 배움.

서버 분들이 알고있으면 좋을거 같은 ‘압축’에 관한 정보를 간략하게 적어 봄.

들어가기전 이 책에선 직렬화 방식을 메모리 스트림을 통해 송수신하는데 본인은 큐로 이해했음.

희소 배열 압축: 보낼 필요가 없는 정보를 제거

이름을 저장할 128바이트 문자 배열이 있을 때, 그 중 5바이트만 쓴다면 123바이트는 보낼 필요가 없다. 5바이트를 먼저 기록 후 문자열을 기록한다.

사실 문자열을 나타낼땐 std::string을 최적화 관점에서 추천하고 있다.

엔트로피 인코딩: 데이터 압축에 있어 출현하는 데이터의 예측 가능성 정도가 얼마나 높고 낮은가에 따라 압축률이 달라진다는 이론

물론 게임 차원에서 CPU 자원을 엔트로피 분석에 쓰는건 비효율적이지만 어느정도 간단한 방법으로 효율적인 엔트로핑 인코딩을 할 수 있는 경우가 있다.

예를 들어, 좌표를 나타내는 Vector3 자료형이 있을 때, 점프를 거의 안한다고 가정하자.

그러면 y는 대부분이 0일 것이고 직렬화 할때 bool 값을 통해 y가 0인지 아닌지 구분하게 한다(bool 값은 당연히 스트림에 1비트만 작성된다).

y==0이면 1비트만 더 쓰고(총 64(x, z좌표) + 1(bool) = 65), 아니라면 y좌표를 기록한다(32 + 32 + 1 + 32 = 97).

이로써 바닥에 있는 시간이 아닐때보다 더 많을 수록 이득이 커진다.

하지만 이렇게 계속 코드를 짜기엔 복잡해지므로 개별 필드를 하나하나 최적화 하는 것이 아닌 있는 그대로 직렬화 후, 허프만 코딩(zip 혹은 zlib, lz4), 산술 코딩, 감마 코딩, RLE 인코딩을 하는 방식을 널리 쓴다.

고정 소수점: 정수를 미리 정해둔 상수로 나누어 쓰는 숫자

엔트로피 인코딩 예시에 이어서 x, z 좌표를 최적화 하고 싶다. 월드 크기는 4000 \* 4000일 때 x, z는 [-2000, 2000]이 되는 것이다.

그리고 좌표이동의 정밀값은 0.1이면 충분하다는 결론이 났다면, x축에 가능한 값의 최대 개수는 (2000 - (-2000)) / 0.1 + 1 = 40001개 이다.

40001은 16(log40001 = 15.3)개 비트로 표현이 가능하므로 기존 float형 32비트에서 반으로 줄어든 크기로 표현이 가능해졌다.

그러므로 총 좌표 크기는 16 + 16 + 1 (+ 엔트로피 인코딩 사용 시 y좌표 평균값)이 된다.

기하 압축: 여러 기하(geometry) 자료형을 압축하는 기법. 이 책에선 사원수와 변환 행렬을 다룸

사원수로 회전을 표현할 때 정규화하여 [-1, 1]로 만들고 네 성분을 다 제곱하고 더했을시 1이 나온다는 사실을 이용한다.

3성분을 고정소수점으로 표시하고 1비트를 추가로 적어 4번째 성분의 부호를 나타낸다. 후에 4번째 성분 값은 1 - (세 성분 다 더한 값)으로 구한다.

이로써 float형 4개 128비트를 정밀도를 아주 조금 훼손하고 49 (16 \* 3 + 1)비트로 직렬화가 가능해진다.

변환 행렬은 원래 float형 16개로 구성 되지만 아핀 행렬이라면 이동 3, 회전 사원수 1, 스케일 3개로 10개의 float만 사용하면 된다.

이때 스케일은 1이 아닌 경우가 대부분이라 유니폼(세 성분이 다 같음)인지를 비트로 표기 후 유니폼한 경우 세 성분 중 하나만 직렬화 하면 된다.

언리얼의 멀티플레이 지원 FSocket에 대해서도 잠깐 다뤄보았음. 네트워크 게임 프로그래밍에서 배운 WinSock은 Window 종속이지만 언리얼의 FSocket을 쓰면 플랫폼에 종속받지 않는다하여 따로 프로젝트를 파서 테스트 중임.

추가적으로 클라와 함께 열리는 서버가 아닌 단독 서버(전용 서버)를 열기위해 작업 중이나, 언리얼 코드 빌드 과정에서 오류가 생겨 해결하는 중에 있음. 하지만 직접 만든 서버를 구동한다면 불필요한 작업이라 생각 되므로 너무 시간을 많이 소요될 시 안 할 수도 있음.

진행 상황 유튜브 링크: <https://youtu.be/uNwGsUF6qrE>

|  |  |  |  |
| --- | --- | --- | --- |
| **문제점 정리** | 전용 서버가 만들어지지 않는다. | | |
| **해결방안** | 구글링 또는 포기 | | |
| **다음주차** | 18주차 | **다음기간** | 2023.10.24~2023.10.30 |
| **다음주 할일** | 새로운 서버 책에서 새롭게 습득할 내용 있는지 확인하기(읽어보기)  프로토타입 준비  언리얼의 FSocket 공부해보기 | | |
| **지도 교수**  **Comment** |  | | |