웹 프로그래밍 TP 최종 보고서
Realtime TPS Shooting Game 만들기
(구현 코드 총 : 2214 라인) *(wc -l 이용)

1. 본 과제의 개요

웹 프로그래밍 과목에서 배운 HTML5 JAVASCRIPT 기술에 소켓 통신을 더하여 실시간 통신으로 이루어지는 게임을 구현했다. 플레이어는 TCP 연결을 통해 접속을 하고, 같은 방에 모두 캐릭터를 생성해 플레이 한다. 제한 시간 안에 고득점 하는 것을 목표로 하는 간단한 슈팅 게임이다. 브라우저만 있으면 가볍게 즐길 수 있는 것이 특징이며, 채팅을 이용해 간단한 소통을 할 수 있도록 했다.

2. 본 시스템 구현

1. 3D 그래픽에 대한 간략한 설명

가. 렌더링 파이프라인

3D 그래픽을 그리기 위해서는 몇가지 단계를 거치는데 아래와 같은 과정을 거친다

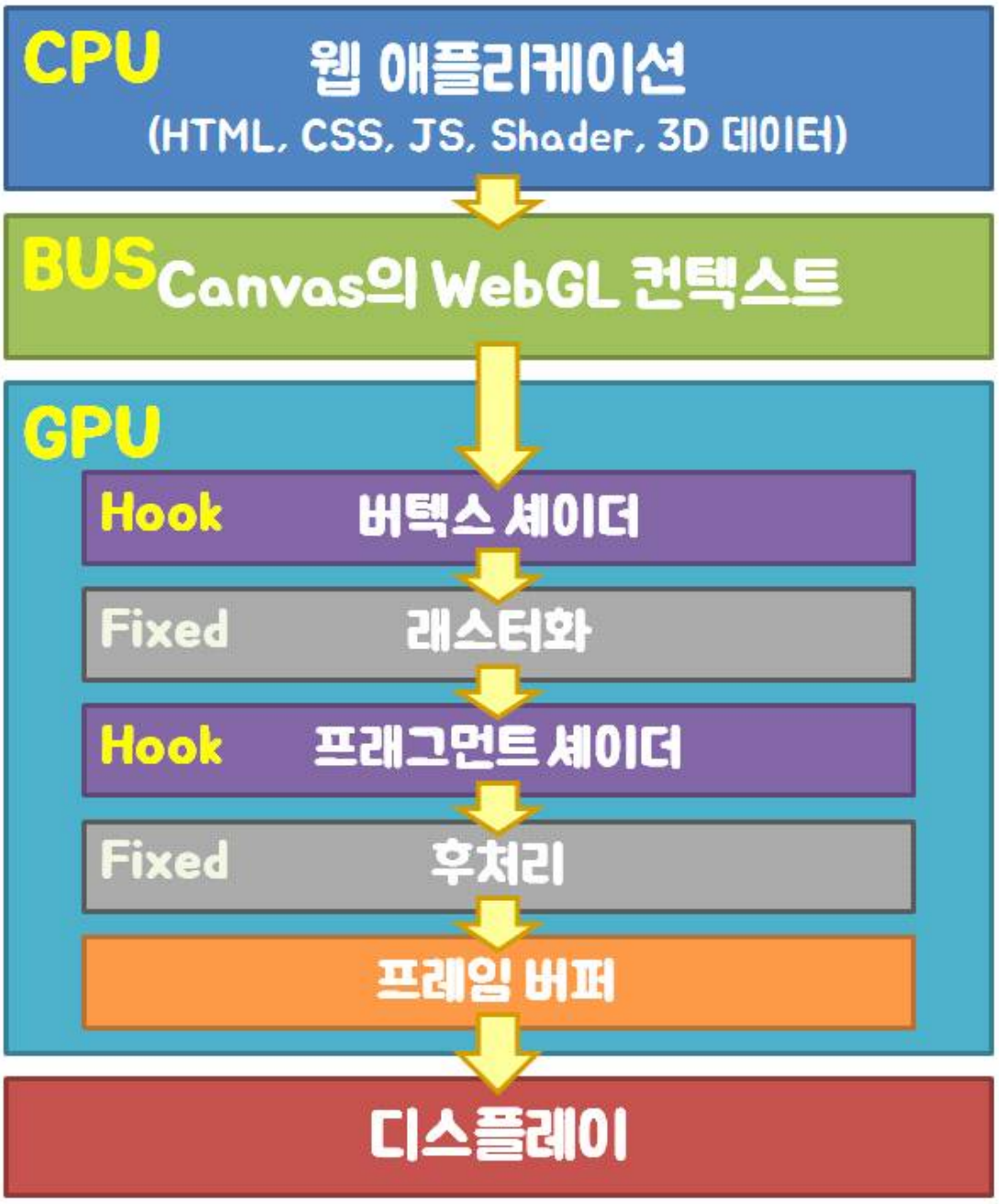

그림 1 Web GL Rendering Pipeline

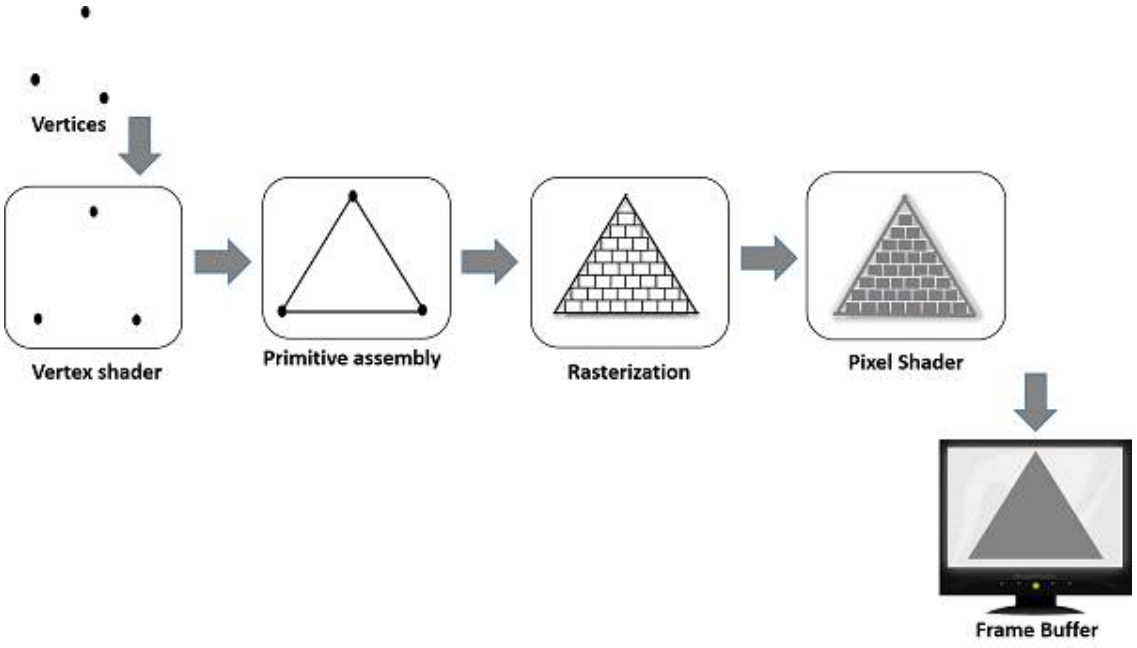

그림 2 Processing of Vertices

간단히 설명하면
Vertex Shader 단계에서 화면에 점을 찍어준다
Primitive Assembly 단계에서 엣지를 생성한다
Rasterization 단계에서 데이터를 유한한 픽셀로 옮기는 처리를 한다
Pixel Shader에서 픽셀을 색칠해준다.

특히 Vertex Shading 과정에서는 3차원 공간 내의 Vertices가 Camera에 투영되었을 때 어디에 위치하는지 행렬로 계산하여 그 위치에 그려낸다

3d 물체의 점들은 그것의 고유한 원점을 기준으로 상대적인 좌표를 모두 가지고 있다. 이것을 가상 공간의 좌표로 옮긴다. 다음으로 카메라의 위치, 보는 방향을 이용해 점들의 좌표를 새로이 지정한다. 다음, 절두체의 공간으로 한번 더 걸러낸다. 마지막으로, Viewport라고도 하는 실제 화면의 좌표에 대응시킨 좌표로 옮겨서 실제 화면에 보여지게 한다

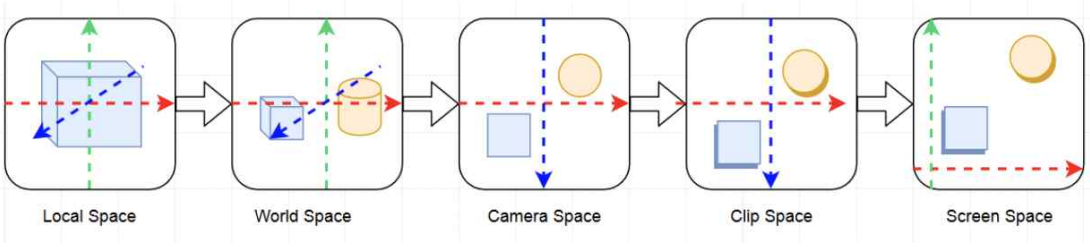

그림 3 Vertex Processing

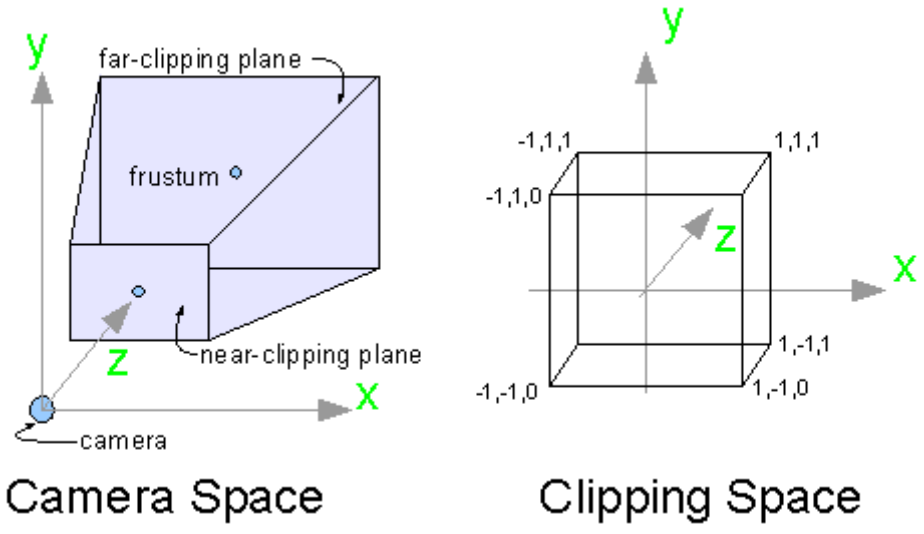

그림 4 Camera Space to Clip Space

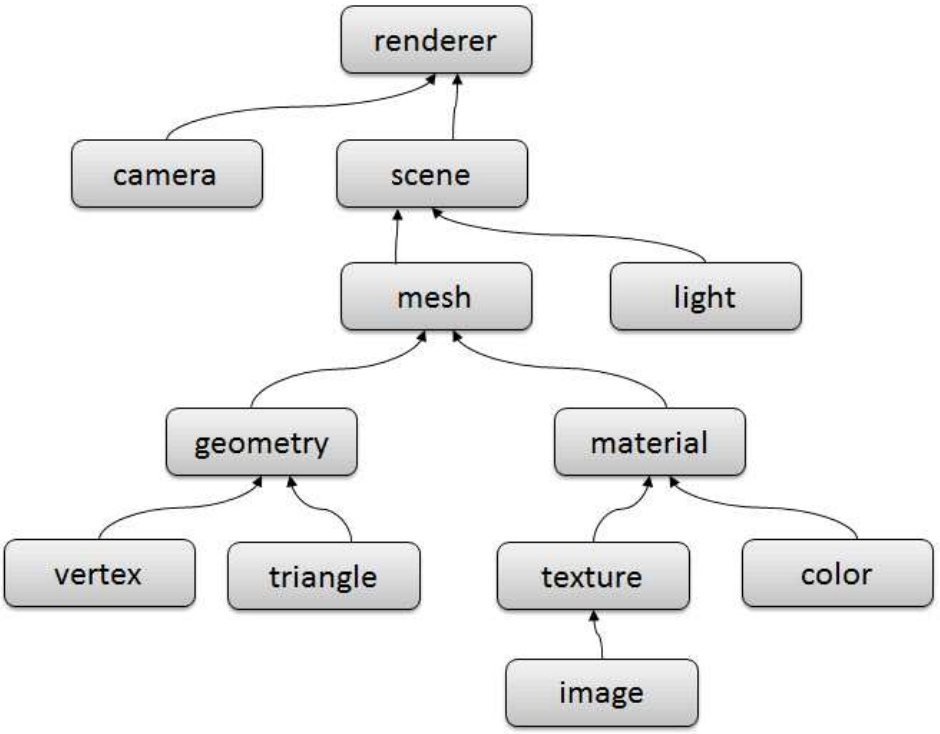

그림 5 Three.js library class hierarchy

나. Renderer

Renderer는 라이브러리로 만든 3차원의 물체를 화면에 그려내는 역할을 맡는다. 즉, 우리가 렌더러가 요구하는 알맞는 형식만 지켜서 Scene에 넣어주면 위에 설명한 부분에 대한 처리를 모두 해주는 것이다. 일반적인 라이브러리는 이 렌더러에 대해서도 복잡한 정의를 해야하지만, 이 프로젝트에서 이용한 라이브러리는 생성만 하면 된다.

다. Camera

3차원 공간을 2차원 디스플레이에 보여주기 위한 행렬 변환을 위해 존재하는 오브젝트이다. 가상의 3차원 공간에 존재하면서 어디를 보고 있으며, 얼마나 멀리까지 볼 수 있는지 등을 정의하고 있는 클래스이다

라. Scene

그래픽스에서 3차원의 가상 공간을 나타낼 때 쓰는 클래스이다. 생성한 모든 물체는 이 Scene 안에 생성된다

마. Mesh

Vertex와 Edge 그리고 Three js에선 Material까지 포함되어있는 3차원 물체를 나타내는 클래스이다. Scene 내부에서 형태를 갖추고 있으며, Material에 따라 Light에 대해 반응할 수 있다.(이 부분은 렌더러가 처리한다)

바. Geometry

Mesh의 골격을 생성하는 Vertex와 Edge로 구성된 껍데기이다. 쉽게 말하자면 그림 2 와이어 프레임 같은 것이다

사. Material

Mesh의 색상을 표현해주는 클래스이다. 재질이라고도 하며, 일반적인 Material의 경우 난반사광, 전반사광, 주변광 등에 대한 반사값을 가지고 있으며, 텍스쳐를 입혀 이용할 수도 있다

아. Light

가상의 빛이다. 점광원, 스팟라이트 등으로 구분되며 Material이 있으면 이것의 반사값을 정해주어 카메라에 보이게 하는 역할이다

자. Texture

이미지 파일이다. Material에 넣어 메시에 그림을 넣을 수 있다

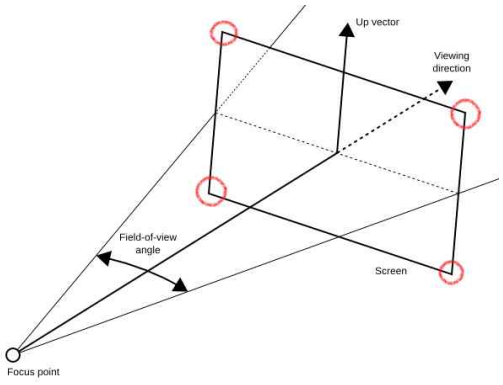

그림 6 Perspective Camera

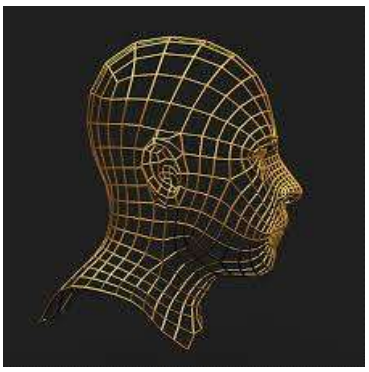

그림 7 와이어프레임

2. 그래픽 시스템 구현 설명

3d 그래픽 구현을 위해 Three.js 라이브러리를 이용했다. 캔버스에 보다 쉽게 삼차원 그래픽을 그릴 수 있다. 먼저 Renderer를 생성하고, 폭과 높이를 지정해준다. 그리고 이 Renderer를 붙인 캔버스를 html에 생성한다. Renderer를 정의했으므로 Camera와 Scene을 만든다.
그림 3와 같은 Perspective Camera를 생성한다. Camera의 FOV, min max depth, 좌표를 정한다. Scene을 만들고 거기에 카메라를 붙여준다.
Scene은 특별한 오브젝트의 관리를 위해 THREE.Scene을 상속받은 새로운 GScene을 이용했다. 이 상속받은 GScene은 Camera와 LoadingScene, ObjectController를 가지고 있다. 새로운 Object를 생성하고 화면에 보여주기 위해서는 Scene에 add를 해야하고, 만일 움직이는 Object라면 ObjectController를 같이 add 해주어야한다. ObjectController가 삭제되면, Object도 같이 삭제된다

3. 게임 시스템

가. 로그인창

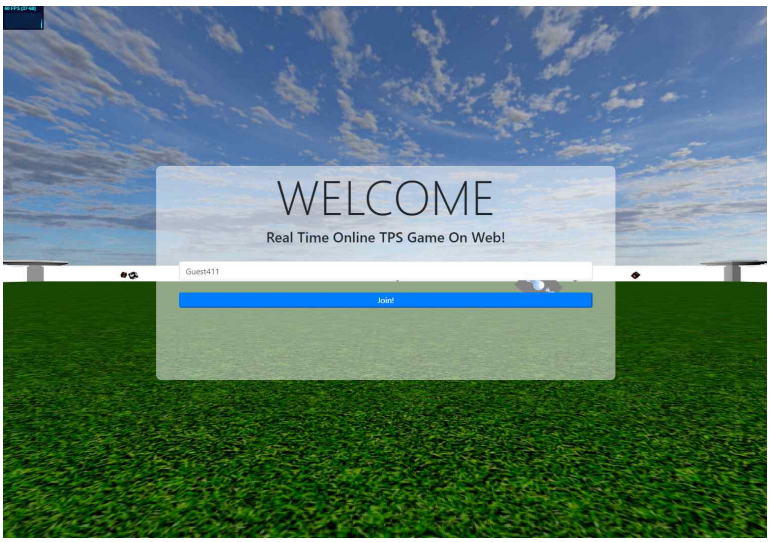

그림 8 로그인 화면

로그인 창이다. 첫 접속시 랜덤으로 닉네임이 부여되며, 영문자와 숫자를 사용할 수 있다. 로그인 창 뒤로는 실제 서버의 상황이 계속 갱신되어 보인다. Join 버튼을 누르면 게임 서버에 캐릭터 생성 요청이 가고, 서버는 캐릭터를 생성하여 클라이언트에게 돌려준다. 클라이언트는 자신의 캐릭터를 컨트롤러에 연결하여 화면에 띄운다. 캐릭터가 생성된 시점부터 게임은 진행된다.
게임 서버에 자신을 등록하는 것과 동시에, 채팅 서버에도 따로 연결이 되어 채팅을 할 수 있게된다.

나. 인게임 시스템

그림 9 인게임 그래픽

1) 플레이어 조작

가) 플레이어 움직임

플레이어는 WASD로 움직이는 기본적인 슈팅 게임의 조작법을 따른다. 그리고 플레이어의 움직임의 경우 클라이언트에서 처리하고 그 결과값을 서버에 전송한다. 공격을 하는 것은 미사일을 생성해 발사를 하는 것이므로 플레이어의 화면에 큰 영향을 미치지 않으나, 움직임의 경우 미세한 지연에도 화면은 매우 떨리므로 플레이어는 불편함을 느낄 수 밖에 없다. 그래서 클라이언트에서 움직이는 좌표를 계산해 서버에 전달하게 했다.

나) 플레이어 공격

미사일은 투사체로 속도를 갖는다. 3D 화면의 중앙에 크로스헤어가 가리키는 곳에다가 쏘아야 하기 때문에, Raycast를 이용해 화면의 중심이 가리키는 곳의 좌표를 구했다.

Raycast는 고비용 연산이기도 하면서, 판정하지 말아야 할 오브젝트도 있다. 예를 들어 자신의 플레이어에 에임이 맞추어져 있다고 생각해보면, Raycast가 자신의 캐릭터의 머리 부근의 좌표로 설정될 것이고, 총알이 이상한 방향으로 날아갈 것이다. 몇 가지 이유로 Raycast할 오브젝트는 리스트를 따로 두어 관리했다. 클릭을 하면 서버에 이벤트를 날려 주고, 서버에서 미사일을 생성하고 다시 클라이

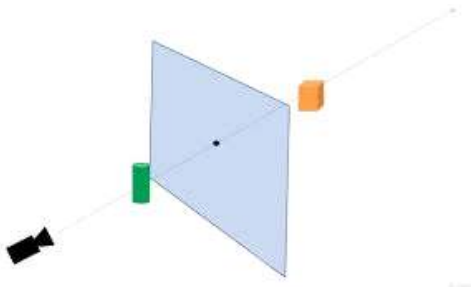

그림 10 Raycast

언트에게 생성된 미사일의 좌표를 넘겨준다. 클라이언트는 생성된 미사일을 Scene에 등록하고 사용자에게 보여주기만 하면 된다.

다) 플레이어의 피격 및 사망

플레이어와의 충돌 판정은 일반적인 충돌인 둘 사이의 거리를 이용하였다. 이는 서버에서 계산을 하고 클라이언트에 결과만 보내주도록 했다. 플레이어가 시간이 다 되거나, 체력이 전부 소모되면 사망한다. 사망을 하면 죽는 모션이 나오고, 잠시후 캐릭터는 서버에서 제거되고 곧 클라이언트에서도 제거된다.

2) UI

가) 체력바

중앙 하단에는 체력 바가 표시되어있다. 체력은 캐릭터 정보를 서버에서 생성 시 정해진다. 실시간으로 서버에서 체력을 갱신해주므로 클라이언트도 서버의 데이터를 받아서 적용만 해주는 역할을 맡고 있다.

나) 점수

상대방을 맞추거나 죽이고, 아이템 박스를 먹으면 점수가 오른다. 회복 아이템은 습득시 역으로 점수가 깎이게 된다. 죽이는 것이 굉장히 많이 점수가 오르므로 플레이어에게 공격적인 플레이 스타일을 강요하게 된다.

다) 시간

시간은 첫 캐릭터 생성 시간을 기준으로 3분이 주어진다. 회복 아이템을 먹을 시 약간의 시간이 늘어난다.

라) 속성

현재 자신의 속성을 나타낸다. 다른 속성을 먹으면 다른 이미지로 바뀐다 Animate.css로 효과를 주어서 전환시 뒤집어지게 만들었다

마) 채팅

채팅은 채팅 서버에 연결되어 진행된다. 채팅을 치면 현재 시간과 자신의 닉네임이 나오고 채팅 내용이 나온다. 이외에도 로그인, 로그아웃, 데스의 로그도 여기에 뜬다.

다. 게임 오버

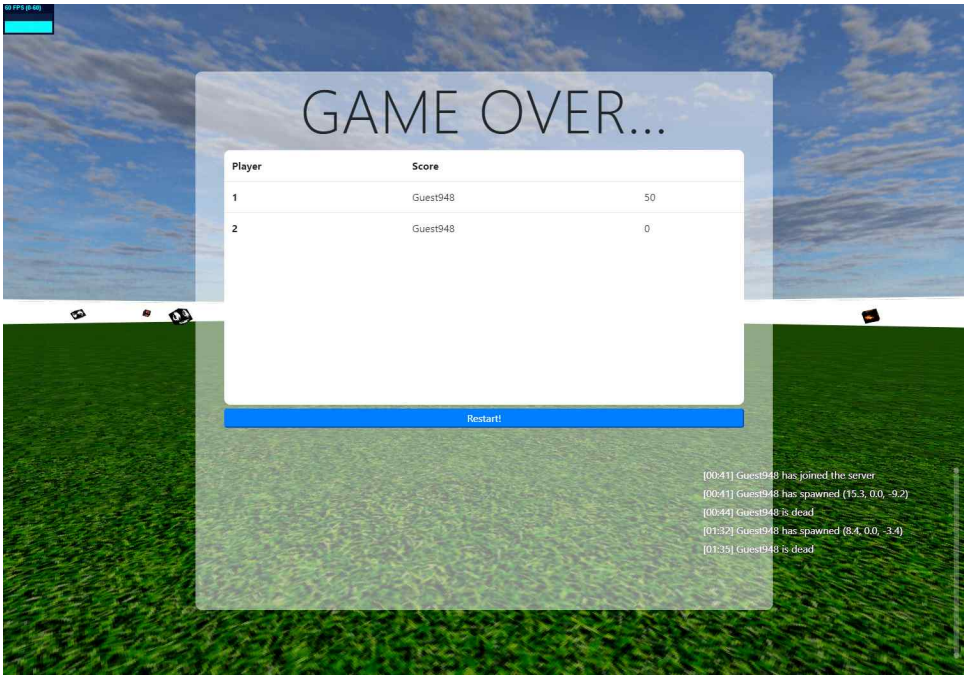

그림 11 게임 오버 창

체력이 다되거나 시간이 다 되면 캐릭터는 사망하며, 점수가 기록으로 남는다. 같은 닉네임이라도 계속해서 기록에 남게 된다

라. 서버

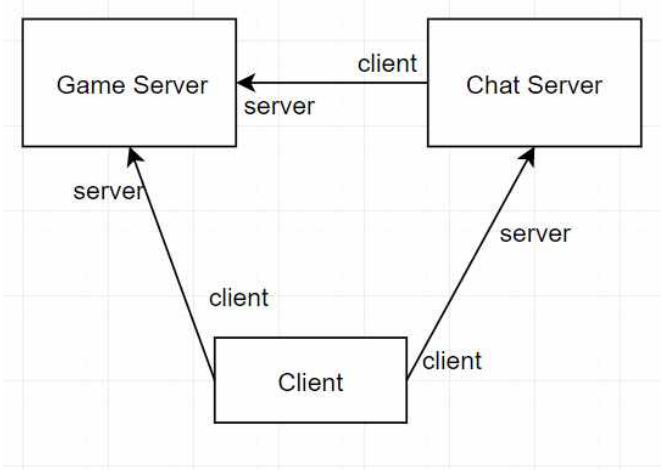

그림 12 게임 통신 구조

클라이언트는 게임 접속 시 게임 서버와 채팅 서버에 동시에 연결하게 된다. 게임 서버에서는 게임 접속, 플레이어 사망 등의 로그만 뜨고, 채팅 서버에서는 실제 유저들의 대화에 대한 로그까지 뜬다. 채팅 서버는 사용자의 채팅을 처리한다. 그리고 게임 서버에 대한 클라이언트로 연결되어, 게임 이벤트 발생 시 자신도 이벤트를 받아 채팅 서버에 뿌려주는 역할도 맡고 있다.

3. 구현한 본 시스템의 특징

```
// collided player and bullet
for(let userid in players){
  let playerPos = copyVector3(players[userid].position);
  playerPos.y+=0.13;
  bullets = bullets.filter((bullet)=>{
    if(bullet.player == userid){return true;}
    let length = getEuclideDistance(playerPos, bullet.position);
    if(length < 0.3 && players[userid].hp > 0)
    {
      fureDamageForPlayer(players[userid],bullet);
      addCCForPlayer(players[userid],bullet);

      willBeRemoveBullets.push(bullet);

      if(players[userid].hp <= 0){
        players[bullet.player].score += 1000;
      }
      else{
        players[bullet.player].score += 10;
      }
      return false;
    }
    return true;
  });
```

기존의 슈팅 게임에는 속성에 의한 상성이 존재하지 않는다. 총 같은 경우도 한번 시작하면 바꾸는 것도 어렵다. 이것을 개선하여 게임 상에 아이템을 이용해 계속해서 속성을 바꿔나갈 수 있으며, 이를 이용해 전혀 다른 게임 양상을 만들 수도 있을 것이다.

```
socket.on('updatePlayer',data=>{

  if(players[socket.id] != undefined &&
    players[socket.id].time > 0 &&
    players[socket.id].hp > 0)
  {
    players[socket.id].time = players[socket.id].endTime - Date.now();
```

플레이어는 생존 시간이 정해져 있으므로, 아무리 잘하더라도 일정 시간 이상 게임에 같은 캐릭터로 머무를 수 없다. 시간제한은 3분으로 정했다. 회복 아이템으로 조금씩은 늘어나지만, 결국 오래 머무르지 못하고 죽게 된다. 이를 통해 하나의 캐릭터로 무한히 다른 사람을 죽이는 것을 막아 누구나 어렵지 않게 비슷한 위치에서 게임을 시작할 수 있게끔 했다.

4. 최종 구현 시스템이 이전에 설계 I 보고서와 비교하여 다른 점

랭킹, 채팅, 로그인 시스템을 추가적으로 구현하였습니다

5. 참고 사이트 및 소스

used library
three.js
bootstrap
animate.css
node.js
socket.io

https://threejs.org/docs/
three js 라이브러리 함수 참고
https://webdoli.tistory.com/36
three js 라이브러리 호출 참고
https://threejs.org/examples/
three js example code 호출 참고
https://gist.github.com/cdata/f2d7a6ccdec071839bc1954c32595e87
gltf model clone 함수 이용
https://poiemaweb.com/nodejs-socketio
node js socket io 인터페이스 참고