CAPSTONE DESIGN

Space Log 최종 보고 및 방학 계획

목차

201321333 한재선, 201021340 원용률

2016-06-26

1. 최종 보고
   A. 클라이언트
      i. 회원가입

- 우선 멀티 플레이 게임 방식이므로 회원가입에 따른 관리가 필요함을 느껴 회원가입 페이지를 만들게 되었다. 처음에는 폼의 형식으로 action을 주었을 시에 해당 URL을 통하여 서버에 전송하였지만 회원가입시에만 form을 이용하여 전달하는 방식은 자칫 socket.io를 이용한 방식과 혼란을 조장할 수 있기 때문에 고심 끝에 form 형태가 아닌 jquery를 이용하여 click시 소캣을 열어 데이터를 주고 받는 방식으로 일반화 시켰다. 따라서 index.html 페이지를 만들어 회원가입 시에 socket.io를 이용한 서버와의 통신을 하여 해당 플레이어어의 아이디, 비밀번호, 그리고 이메일 주소를 넘겨주는 작업을 하였으며 여기에서 처음엔 공성전 방식을 생각하여 이메일 주소를 통한 공성전 알림을 생각하였으나 중간 회의 결과 게임 방식의 전환. 즉, 행성의 약탈 및 함선 자체의 실시간 슈팅 게임 요소가 주가 되었기 때문에 이메일이 필요 없게 되어 추후 이메일 부분과 로그인 부분을 통합하는 것도 생각하고 있다. 왜냐하면 따로 가시적인 폼을 만들어봤자 의미가 없다고 판단하였기 때문이다. 따라서 로직의 수정이 아닌 디자인적인 수정을 약간 가미할 계획을 가지고 있다.

결론적으로 회원가입은 서버와의 통신이 제대로 되는 것을 테스트 결과 확인하였다.

      ii. 로그인 및 로그아웃

- 로그인의 경우, 가입한 유저의 데이터를 받아 일치 및 불일치의 여부를 판단하여 메인 게임 페이지에 들어가는 방식이므로 크게 어려움은 없었으나 한 가지 문제가 있었는데, 그것이 바로 페이지 간의 값 전달 방식 해결 문제였다. 처음에는 인터넷 페이지의 특성상 페이지가 reload되거나 페이지 간에 이동을 하였을 시에 같은 소켓이 유지되지 않는 다는 특성의 자료를 읽고 나서(왜냐하면 일반적인 인터넷 연결은 한 번 연결을 맺으면 계속 유지하는 방식이 아닌 연결 했다가 다시 끊고 다시 연결하는 방식이므로) http protocol을 기반으로 한 web socket자체도 그리할 거라 예상하였다.

그리하여 값을 전달 시에 동일 값이 전달이 안될 수도 있을거라는 생각을 하여 회원가입 페이지 로그인 시에 해당 값을 가진 url을 main page에 전달하는 방식의 해결책을 고안하였는데 결론적으로는 여러 시도 끝에 해결을 하였으나 url을 보낼 때마다 값의 변경에 따른 주소의 split을 해야하기 때문에 번거로움이 많았다. 따라서 교수님과의 대화결과 아래와 같은 방식을 생각할 수 있었다.

URL을 이용한 값의 전달.

: 일반적인 홈페이지를 만들 때 값을 전달하는 방식으로 사용하기 때문에 불용.

Cookie를 이용한 값의 전달.

: 큰 값을 이용한 방식이 아니며, 영구적 저장을 이용한 방식이 아님.

localStorage를 이용한 값의 전달.

SessionStorage를 이용한 값의 전달.

: 둘 다 key : value를 이용하는 것이기 때문에 mongo디비와 javascript의 object 방식의 호환이 잘 됨. 따라서 최종적으로 이 방식을 사용. 이 중 클라에서의 페이지간 값 전달 방식에 적합하다고 여긴 것이 localSorage이므로 이 방식을 쓰기로 하였음.

따라서 localStorage를 이용하여 값 전달을 하여 자체적으로 받는 것이 낫다 판단하여 이 방식을 쓴 결과 아주 간단하게 값이 전달되는 것을 확인하였다.
로그인의 경우 위와 같이 회원가입과 로그인의 시스템. 즉, index.html 파일 안에서 조장하는 시스템의 전체적인 로직은 전송하는 형식과 받는 형식상의 상이함만이 있을 뿐이지 외향적인 구조는 같은 구조이므로 외향적인 디자인을 조금 수정할 계획이다.
로그아웃의 경우 서버로부터 내 유저 아이디 정보를 보내고 해당 정보를 서버에서 로그아웃에 대한 결과 메시지를 전송하여 정상적 로그아웃 또는 로그아웃이 안 되는 경우의 에러를 분기로 나누어 다루었다.

iii. 중복로그인

- 로그인 및 로그아웃 문제를 해결해놓고 보니 하나의 문제가 발생하였는데, 그것이 바로 중복 로그인에 의한 같은 아이디로의 다중 접속 문제였다. 이 문제가 생긴 이유는 클라이언트에서 접속 시에 해당 아이디가 접속해 있는지 접속해 있지 않은 지의 여부를 서버 내 데이터베이스에 추가하여 그 여부를 통하여 로그인 및 로그아웃을 제어해야 하는데 그 경우를 생각 못해서 생긴 문제였다.
결론적으로 해당 문제는 상기와 같은 데이터베이스 컨트롤 및 그에 따른 로직 생성으로 해결하였다.

iv. 미개척 행성의 정보 받아서 화면에 출력

- 임 방식 자체가 함선 간 슈팅 게임을 하면서 미개척된 행성을 하나씩 점령하여 자원의 우위를 통한 함선 업그레이드 방식의 실시간 전략 게임이므로 서버에서 만들어진 미 개척 행성의 정보를 받아 해당 화면에 보여줄 필요가 있다. 따라서 클라이언트의 경우 이러한 미개척 행성 데이터베이스를 받아서 division tag안 division tag의 형식으로 뿌려주는 것을 예상하고 작업을 진행하였으나 위치에 따른 tag의 행성이 아닌 지속적으로 리스트화가 되는 발생이 생겨서 모든 것을 document단위로 handling하는 것에 문제가 생겼다. 따라서 미개척 행성의 경우만 canvas내에 띄우도록 작업을 하였는데 교수님과 상의 해본 결과 이것은 div tag의 특성에 따른 문제가 아닌 position에 따른 handling의 잘못으로 일어난 문제이며, 또한 canvas의 경우 실시간 데이터 갱신 시 원활한 이미지 뷰가 안될 가능성도 생기기 때문에 좀 더 생각해 보고 추후 canvas로 그리는 것이 아

닌 tag별로 접근하여 핸들링 하는 방식으로 전환하는 것을 생각하고 있다.

v. 플레이어 함선 방향키에 따른 이동

- 게임이란 것이 플레이 할 시에 자신 만의 아바타. 즉, 자신이 열과 성을 다하여 성장 시킬 수 있는 것이 필요하다. 이것이 게임에서의 기초적인 요소라 볼 수 있는데, 그러기 위해 함선 자체의 컨트롤이 우선이라고 만단하여 기본적인 방향키에 따른 이동을 구현하였다. 하지만 여러 가지 문제점이 생겼는데, 그 중 하나가 우리가 생각한 좌우키의 움직임 시 해당 각도를 틀고 위, 아래 키를 눌렀을 시에 해당 각도로 전진 및 후진하는 방식에서의 문제점 이었다. 이 방식자체가 문제점이 아닌 좌우키의 움직임 시에는 설정한 각도를 회전하여 해당 좌표를 통해 직선 간 움직임 보여야 하는데 여기에서 수학적 지식이 부족하여 처음에는 css 스타일의 transform을 이용하면 그대로 각도 회전을 하였다. 이에 따라 해당 이미지 오브젝트의 회전은 성공하였으나 중요한 것은 제자리 회전이므로 각의 회전에 따른 좌표생성을 하지 못한다는 것이다. 이렇게 된다면 그 좌표를 계산하여 벡터 계산에 따른 직진 및 후진이던지 평행이동을 통한 직선 간 거리 계산에 따른 움직임도 불가능라기 때문에 바람직하지 않은 방식이다.

  해결책이 생각나지 않던 도중 교수님의 조언을 통해 rotation transform matrix. 즉, 회전 변환 행렬 공식을 통해 해당 이미지 물체의 한 좌표를 기준으로 삼아 해당 각으로 이동하여 해당 좌표를 그 좌표 방향으로 갱신하면서 직선 움직임을 꾀하는 것을 방학 목표중 하나로 삼고 있다.

B. 서버

i. 회원가입

- 클라이언트로부터 정보를 받을 때 사용하는 메시지는 'join_msg'.
- MEMBER Collection 에서 위의 정보 중 'username'이 존재하는지 조회하고 이미 해당 username 이 존재하면, 클라이언트에게 전송할 때 join_res 메시지로 {response : 'false'}를 전송 / 존재하지 않으면 MEMBER Collection에 받은 정보를 모두 저장한 뒤 , {response : 'true'}를 전송

ii. 로그인

- 클라이언트로부터 정보를 받을 때 사용하는 메시지는 'login_msg'
- DB MEMBER Collection에서 위의 정보(username , password)를 조회
- 일치하는 정보가 있으면 소켓 메시지 'login_res'를 클라이언트에게 전송할 때 {response : 'true'}를 전송 / 없으면 {response : 'false'} 를 전송

iii. 로그아웃 및 중복로그인

- 처음 구현한 loginForm.js 에서는 유저가 로그인하여 접속을 했는데도 다른 웹 브라우저 창에서 같은 아이디로 로그인이 되는(중복로그인) 문제가 발생
- 위 문제를 해결하기 위해 MEMEBR Collection에 accessing 필드를 추가하여 현

재 해당 아이디가 로그인 상태인지 아닌지를 true, false 로 표시.
- 회원가입을 할 때 기본적으로 accessing 필드 값은 false 로 저장하여 접속하지 않은 상태임을 표시.
- 유저가 로그인을 시도할 때 해당 아이디의 accessing 필드 값을 확인. Accessing 필드 값이 false 일 경우 로그인 가능 / true 일 경우 로그인 불가능.
- 유저가 로그아웃 버튼을 누르면 클라이언트로부터 logout_msg 메시지로 로그아웃하려는 유저의 정보를 받고 MEMBER Collection에서 일치하는 데이터를 찾아 accessing 필드값을 false 로 업데이트하고 업데이트 성공하면 logout_res 메시지로 {response : 'true'}를 전송 / 실패하면 {response : 'false'} 를 전송

iv. 행성 랜덤 생성

- 행성의 고유 번호인 planet_id 필드를 생성하고 행성번호를 정수형 숫자로 넣었었는데 굳이 이렇게 할 필요가 없음. MongoDB에서는 Collection에 Document를 추가할 때마다 각 Document에 고유 아이디 값(필드 명은 '_id')이 주어지는데 이 _id 값은 고유한 값이므로 충분히 행성 고유 번호 역할을 할 수 있음. 즉, _id 필드 = planet_id 필드
- 새로 생성하는 행성의 위치와 자원양을 랜덤으로 생성하기 위해 random 함수를 사용
- random 함수만 사용하면 일정 패턴이 반복되기 때문에 행성의 위치가 겹치는 문제가 발생.
- getTime 함수로 현재 시간을 획득하고 이 시간을 행성 생성 수식에 추가하여 행성의 위치가 겹치지 않게 함.

v. 행성정보 전송

- PLANET Collection 에서 모든 document를 조회하여 객체형식으로 클라이언트에게 모두 전송.

2. 방학 계획

A. 방학 중 진행 할 부분 목록

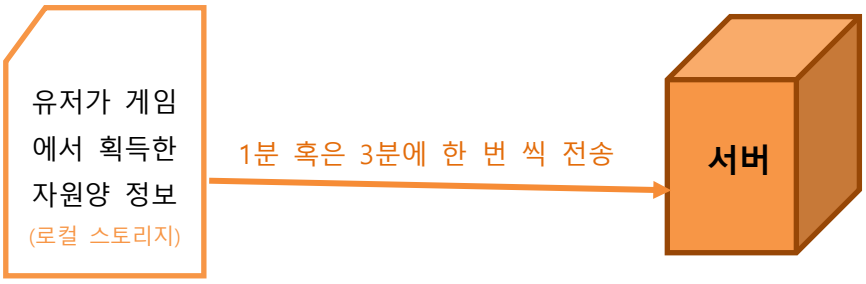
i. 유저가 이동함에 따라 화면도 이동하기. 그리고 게임 종료시에 유저의 마지막 좌표값을 서버에게 전송하여 MEM_INFO Collection의 좌표값 없데이트 하기.

username	exp	mineral	gas	unknown	Location_x	Location_y
					★	★

ii. 유저가 개척한 행성 정보를 받아서 MEM_PLANET Collection에 입력 및 업데이트

행성번호	유저 ID

iii. 클라이언트는 유저가 개척하여 얻은 자원량 정보를 로컬스토리지에 저장하고 있다가 일정 시간이 지나면(ex. 1분 혹은 3분) 현재까지 유저가 얻은 자원량 정보를 서버에게 한꺼번에 전송. 이렇게 하는 이유는 유저가 자원을 획득하는 속도는 1초에 각 자원을 10씩 얻는데 (즉, mineral +10 , gas + 10, unknown + 10 . 이런 식으로 증가) 매번 자원량의 변동이 있을 때마다 그 정보를 계속 서버에게 전송하는 것은 비효율 적이므로 1분이나 3분에 한 번씩 누적 획득량을 서버에게 전송하는 게 좋다고 판단했다.



iv. 유저가 개척해서 얻은 자원량 정보를 받아서 MEM_INFO Collection에 해당 유저의 정보 업데이트

username	exp	mineral	gas	unknown	Location_x	Location_y
	★	★	★	★		

v. 서버에서는 상점에 들어갈 여러 큐브 데이터를 MISSILE Collection 과 SHIELD Collection에 저장하고, 클라이언트에게 이 데이터들을 전송하여 상점에 물품 목록이 뜨도록 하기

missile	offensive

shield	defensive

vi. MEM_SHIP Collection 에 유저가 구입한 방어 큐브와 발사체 큐브의 공격력과 방어력을 계산한 값과 함선의 내구도 정보를 저장

username	durability	defensive	Offensive

B. 진행 계획 달력

* 방학 중 각자의 휴가 계획이 잡히면 계획의 일부를 수정할 수 있습니다.

7월

// : 보완 작업

i ii iii iv v vi vii viii ix : 위의 리스트 항목 번호

일	월	화	수	목	금	토
					1 i	2 i
3 i	4 i	5 i	6 i	7 i	8 i	9 i
10 ii	11 ii	12 ii	13 ii	14 ii	15 ii	16 iii
17 iii	18 iii	19 iii	20 iii	21 iii	22 iii	23 iii
24 iii	25 //	26 //	27 //	28 //	29 //	30 //
31 //						

8월

// : 보완 작업

i ii iii iv v vi vii viii ix : 위의 리스트 항목 번호

일	월	화	수	목	금	토
	1 iv	2 iv	3 iv	4 iv	5 iv	6 iv
7 iv	8 v	9 v	10 v	11 v	12 v	13 v
14 v	15 vi	16 vi	17 vi	18 vi	19 vi	20 vi
21 vi	22 //	23 //	24 //	25 //	26 //	27 //
28 //	29	30	31			