

**웹 프로그래밍**

**팀 프로젝트 과제 #1**

eCampus-breaker



2021 May 28

강관주 201811232

서동균 202011306

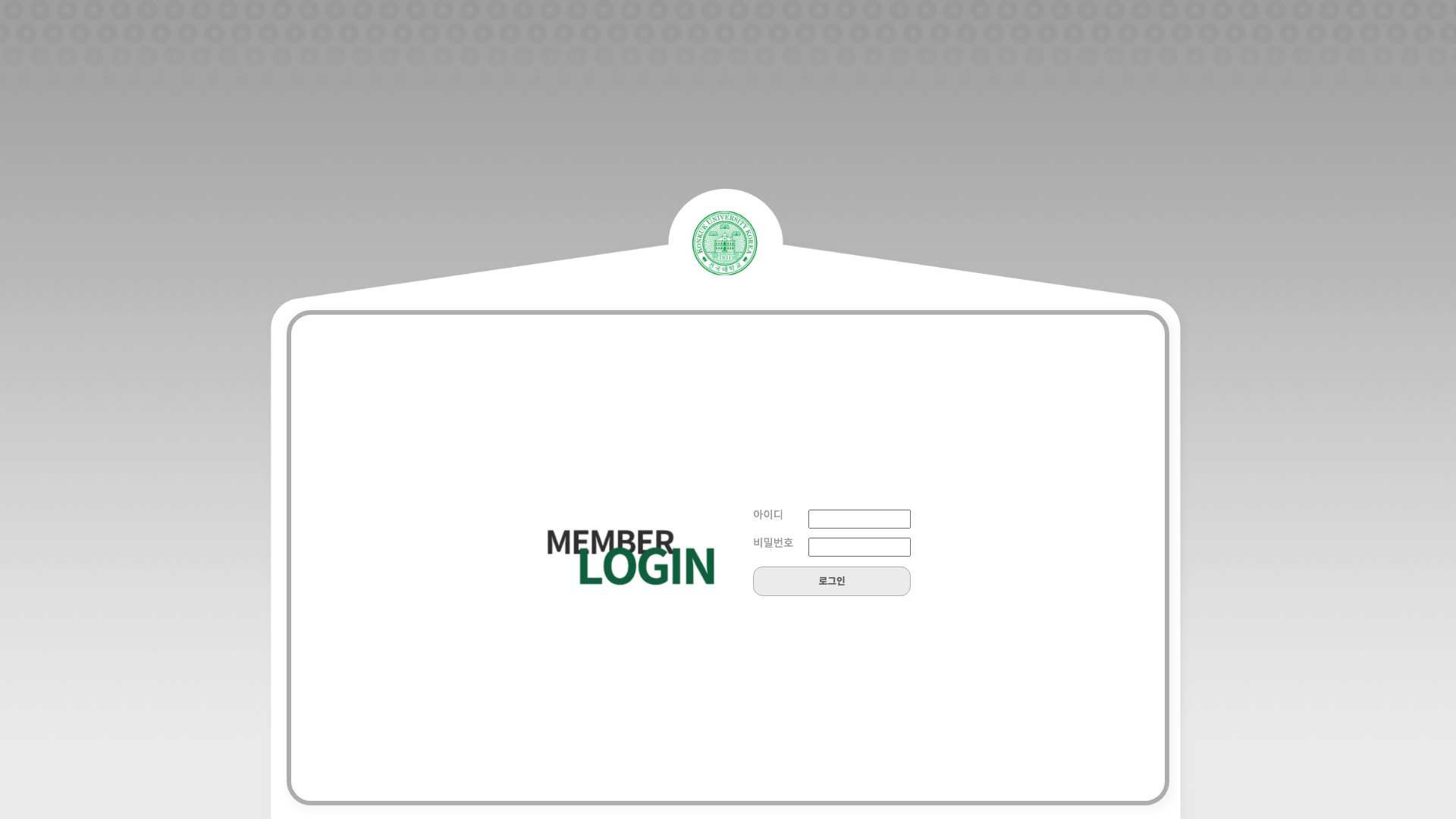
전진호 202011361

정은아 202011365

파리나 202013555

**1. 게임 시나리오 및 게임 동작 방법 (매뉴얼)**

1-1. 로그인 페이지



접속 시 로그인을 할 수 있는 페이지가 로드된다. 브라우저의 화면 확대 정도를 80% 또는 90%로 설정하고 보는 것을 권장한다. ecampus 및 건국대학교 원스탑 서비스 계정을 입력하면 사용자의 이름이 alert 되면서 메뉴 페이지를 로드한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

아이디를 잘못 입력하게 되면 입력한 정보와 일치하는 사용자가 없다는 alert를, 비밀번호를 잘못 입력하게 되면 로그인이 실패하였으며 n회 이상 실패시 접속 불가라는 alert를 띄우게 된다.

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

1-2. 메뉴 페이지

텍스트이(가) 표시된 사진

자동 생성된 설명

메뉴 페이지는 상단에 사용자의 프로필 이미지와 이름, 하단에 대학생 모드, 대학원생 모드, 랭킹, 설정으로 이루어져 있다. footer에는 로고와 제작한 팀원의 이름이 있으며, 로고를 클릭하면 ecampus 화면으로 이동한다.

1-3. 대학생 모드



좌측 상단에 위치한 Score는 현재까지 획득한 점수, 중간에 위치한 Level은 현재 게임의 단계를 의미한다. 우측 상단에 위치한 Timer는 게임 경과 시간을 초단위로 보여주고, Life는 남은 목숨을 의미한다.

**시나리오**

eCampus 화면을 배경으로 쌓여가는 과제들을 없애는 벽돌깨기 게임이다. 일정 시간마다 계속 추가 과제들이 생기며, 모든 과제를 없애면 다음 단계로 올라간다. 건국대학교 학사정보시스템의 정보를 API로 받아와 사용자가 현재 수강하고 있는 과목들로 과제 이름이 구성된다. 단계가 올라갈수록 더 빠르게 추가 과제가 생기며, 2단계와 3단계부터는 어려운 과제들이 추가가 된다. 어려운 과제는 아래와 같이 더 진한 색상의 테두리로 표시가 된다.

텍스트이(가) 표시된 사진

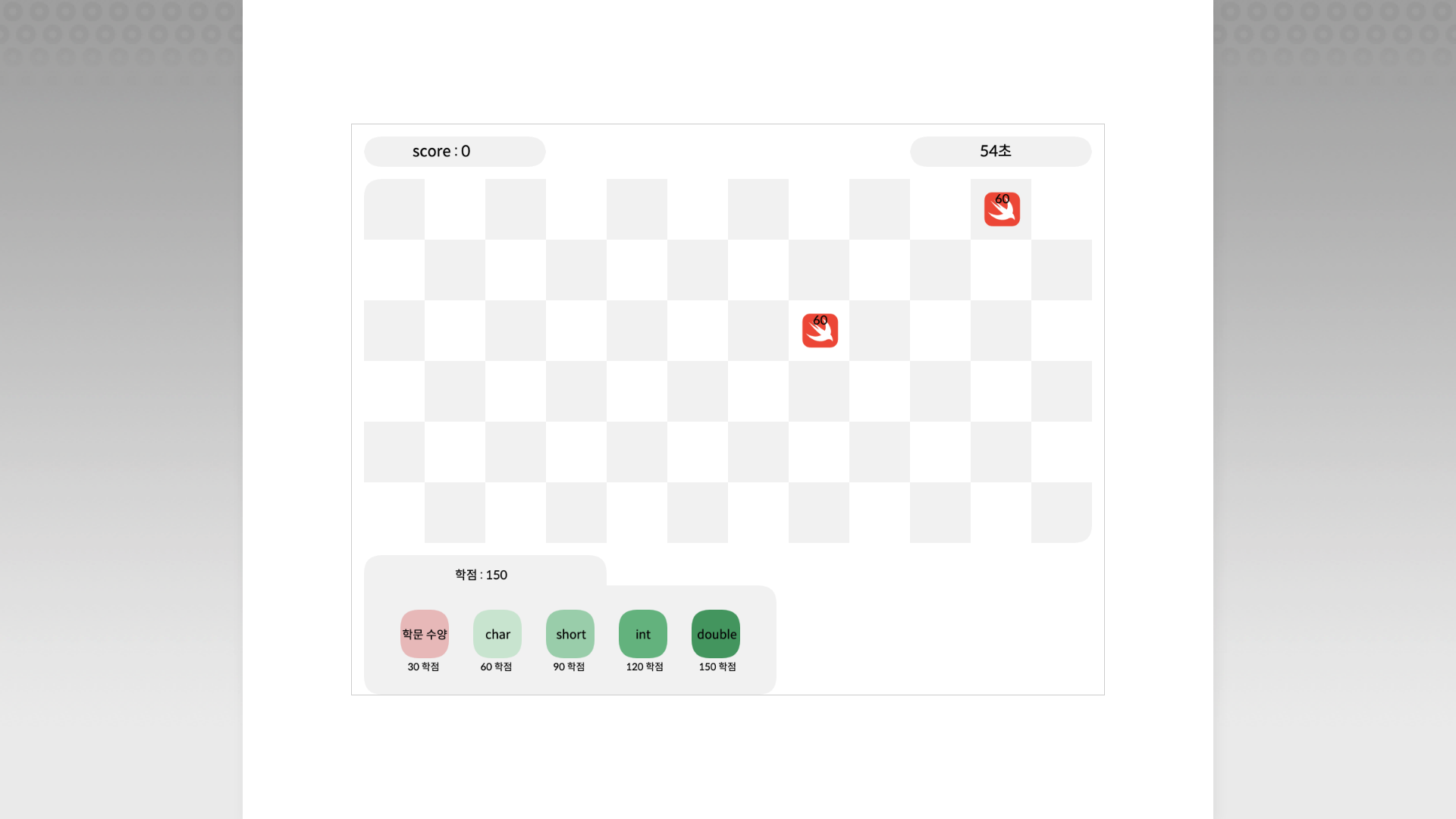
자동 생성된 설명

**동작방법**

마우스를 움직여 패드를 움직인다. 공이 패드와 부딪히면 부딪힌 부분에 따라 공이 움직이는 각도가 바뀐다. 공은 벽이나 벽돌에 부딪히면 방향이 바뀌고 부딪힌 벽돌은 내구도를 전부 잃으면 부서지게 된다. 일반 과제는 회색 테두리, 내구도가 2인 어려운 과제는 검은 테두리를 갖고 있으며, 어려운 과제는 공이 한번 부딪혀도 사라지지 않고 일반 과제로 바뀐다.

일정 시간마다 벽돌이 생성되는데 화면에 있는 모든 벽돌을 없애면 다음 단계로 올라가며, 총 3단계로 구성되어 있다. 아이템은 일정 시간마다 무작위 위치에서 생성되어 아래로 떨어진다. 아이템은 패드를 움직여 획득할 수 있고, 빨간 아이템은 공의 크기를 크게 만들고 파란 아이템은 패드의 길이를 크게 만드는 효과를 가진다.

1-4. 대학원생 모드



화면 중앙 게임 canvas에 적들이 우측에서 생성되어 좌측으로 이동하며, 적들이 좌측에 닿게 되면 게임이 종료된다. 하단에는 설치할 수 있는 터렛이 위치한다.

**시나리오**

eCampus 화면을 배경으로 다가오는 과제들을 막는 디펜스 게임이다. 터렛을 설치하여 다가오는 과제를 저지한다. 60초동안 과제를 놓치지 않으면 다음 단계로 올라간다. 1, 2, 3단계로 구성되어 있으며, 단계가 올라갈수록 과제의 속도가 빨라진다.

**동작방법**

마우스를 움직여 하단의 터렛을 클릭하고 격자무늬를 클릭하면 터렛을 설치할 수 있다. 터렛 설치에는 학점이라는 비용이 사용된다. 기본 150학점이 주어진다. 학점은 과제 제거 또는 [학문 수양] 터렛을 설치하여 획득할 수 있다.

터렛은 5종류 [학문 수양], [char], [short], [int], [double]가 있다. 터렛의 체력은 모두 100이다.

* [학문 수양]: 3초당 +20학점, 가격: 30학점
* [char]: 공격력: 10, 가격: 60학점
* [short]: 공격력: 20, 가격: 90학점
* [int]: 공격력: 30, 가격: 120학점
* [double]: 공격력: 40, 가격: 150학점

과제는 5종류 [PYTHON], [SWIFT], [JAVA], [R], [MYSQL]이 있다.

* [PYTHON] 체력: 30
* [SWIFT] 체력: 60
* [JAVA]: 90
* [R] 체력: 120
* [MYSQL] 체력: 150

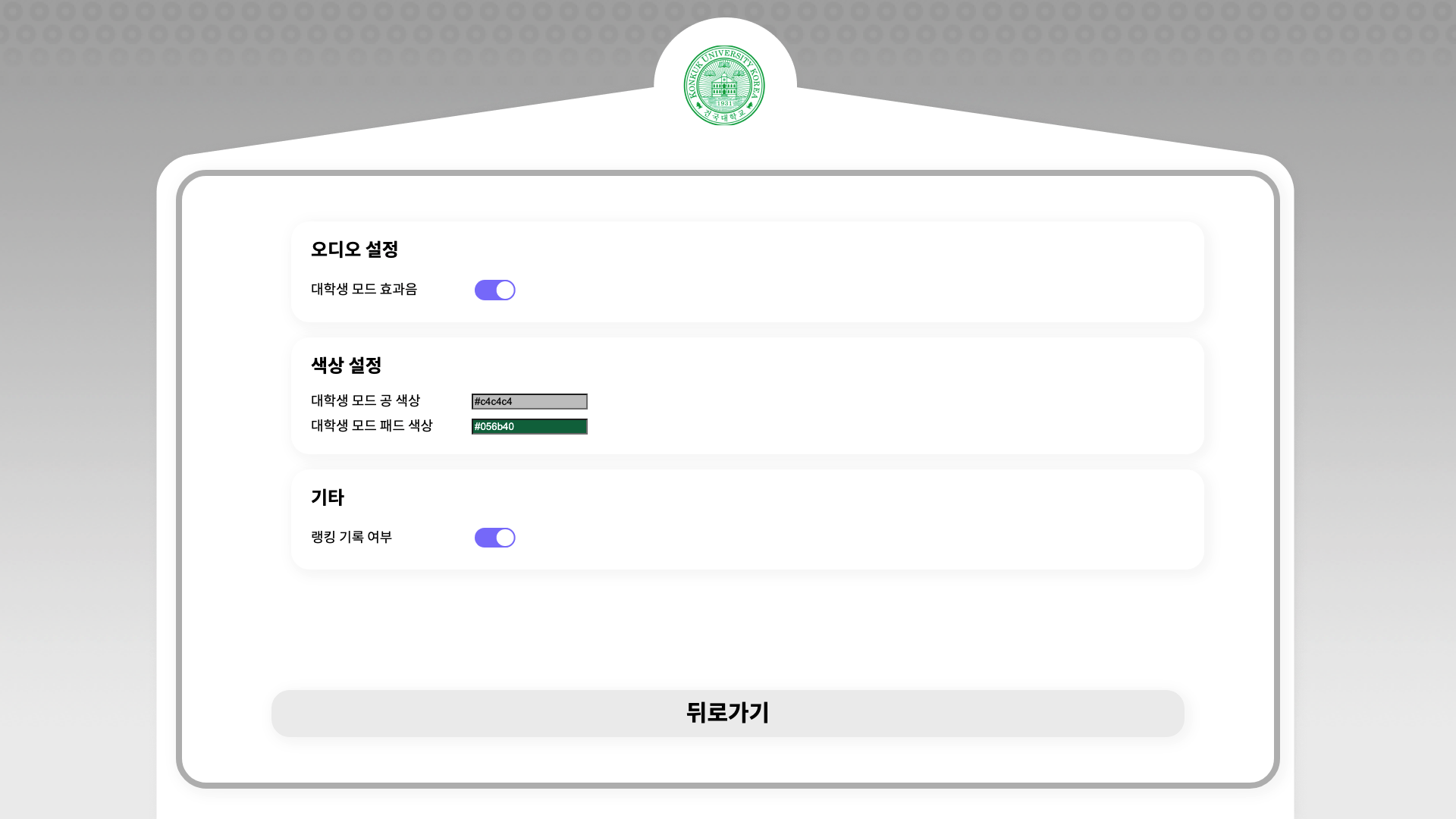
터렛은 코드를 발사하여 과제를 제거한다. 코드에 부딪힌 과제는 터렛의 공격력만큼 체력이 낮아진다. 이후 과제의 체력이 0이 되면 파괴된다. 과제에 부딪힌 터렛은 체력이 서서히 낮아지며, 0이 되면 파괴된다.

1-5. 랭킹 페이지



좌측에는 대학생 모드의 랭킹이, 우측에는 대학원생 모드의 랭킹이 보여진다. 랭킹은 순위, 이름, 점수를 보여주며 5위까지 출력한다. 페이지를 새로고침하지 않아도, 랭킹이나 점수에 변화가 생기면 감지하여 다시 출력한다. 뒤로가기 버튼을 눌러 메뉴 페이지로 이동할 수 있다.

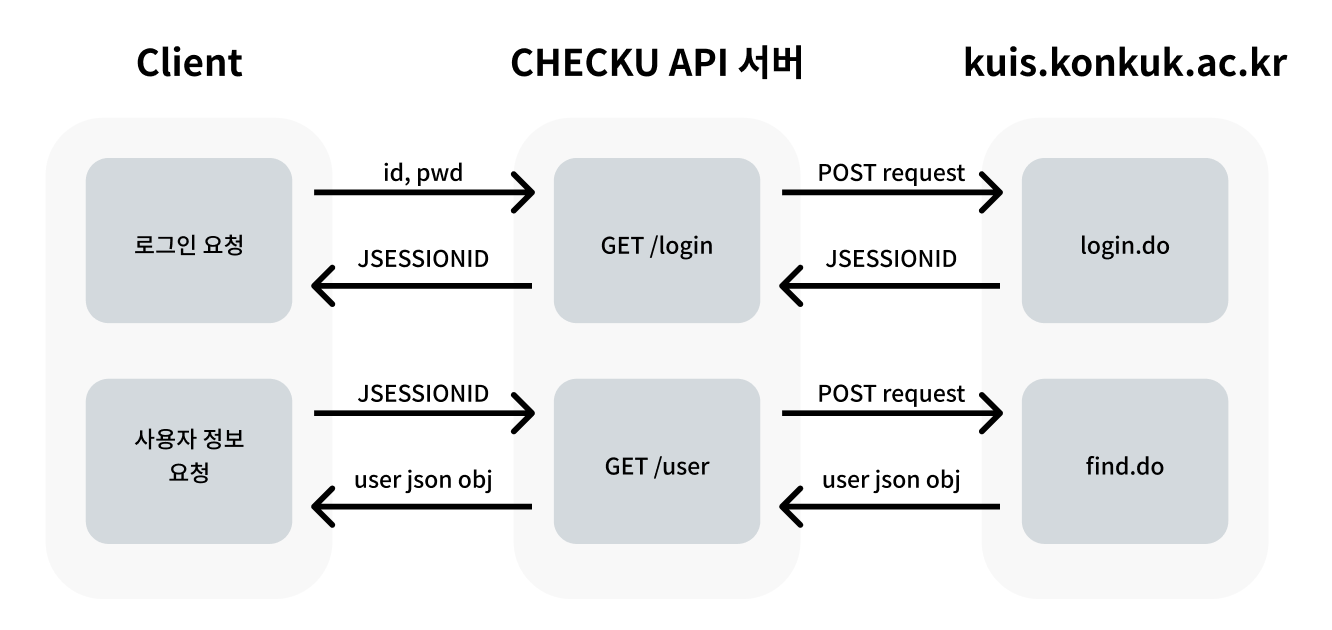
1-6. 설정 페이지



오디오, 색상, 기타 설정을 바꿀 수 있다. 오디오 설정은 대학생 모드의 효과음을 키거나 끌 수 있다. 색상 설정은 대학생 모드에서의 공과 패드 색상을 바꿀 수 있으며 아래와 같이 직접 Hex 색상 코드를 입력하거나 팔레트에서 선택하여 바꿀 수 있다. 기타 설정은 랭킹 기록 여부를 키거나 끌 수 있으며 끄게 되면 랭킹보다 높은 점수를 획득해도 점수를 기록하지 않는다.

**2. 주요 기능 및 구현을 위한 기술적 상세 명세**

2-1. 로그인 기능



로그인 기능은 Django Rest Framework를 이용해 구현하였다. POST 방식으로 <https://kuis.konkuk.ac.kr/Login/login.do>에 headers와 사용자 계정 정보를 담은 payload를 checku.site/login으로 전송하면 서버에서 JSESSIONID를 반환하여 준다. 전달 받은 JSESSIONID는 마치 사용자가 kuis 학사정보시스템에 접속한 것 처럼 믿게 해준다. 이를 이용하여 사용자의 정보를 반환하는 학사정보시스템의 엔드포인트로 접속하면 해당하는 json 객체를 얻을 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**menu.js:19-32**

ES6에서 지원하는 fetch API를 이용해 API 주소로 데이터를 요청하였다. Fetch는 Promise 객체를 AJAX와 함께 wrapping 하여 반환하므로 아래와 같이 위 함수들을 처리하는 login function은 async/await 으로 구현하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**menu.js:41-64**

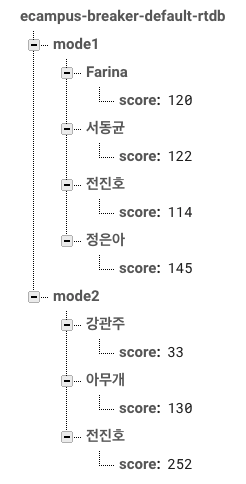
login function에 async 키워드를 추가하여 이 함수가 Promise 객체를 다룸을 선언하고 getUser, getJSESSIONID, getSubjects로 부터 awiat을 통해 전달 받도록 구현했다. JSESSIONID에 ERRMSG가 포함되어 있으면 아이디 정보가 존재하지 않거나 비밀번호가 존재하지 않는 경우이므로 ERRCODE를 alert한다. 만약 계정 정보가 일치하는 사용자가 있다면 getUser를 통해 사용자 정보를 조회하는데, API 구현 상의 문제로 요청이 실패하는 경우가 있었다. 그래서 만약 실패하는 경우 성공할 때까지 다시 JSESSIONID를 발급 받고 getUser를 실행한다. 이와 같은 문제로 로그인 요청이 1800ms에서 3000ms까지 지연되는 현상이 발생한다.

2-2. API 서버 배포

외부에서도 API 서버에 접속할 수 있게 하기 위해 Amazon Web Service(이하 AWS)를 이용해 서버를 배포하였다. 서버로는 Nginx를, gateway interface로는 Django에 내장된 WSGI를 사용했다.

그러나 라이브 코딩 협업 도구 CodeSandbox에서 외부 http 서버로 요청을 보내자 Mixed Content 오류로 인해 접속이 안되는 현상이 발생하였다. 따라서 Route 53과 Certificate Manager를 이용해 https 접속도 받을 수 있도록 추가적인 작업을 진행하였다. Route 53은 다른 도메인으로부터 리디렉션 받을 수 있도록 하는 DNS 서비스이며, https, http 요청을 받도록 레코드를 추가하는 작업을 수행했다. Certificate Manager은 사설 SSL 인증서를 발급해주는 서비스이다. Nginx에서도 https를 리디렉션 해주는 코드를 추가하고 서버를 재구동하여 https 요청을 처리할 수 있도로 작업을 마무리 하였다.



2-3. 랭킹 시스템 구현

백엔드 없이 랭킹 시스템을 구현하기 위해 Firebase의 realtime database를 이용했다. 따라서 CDN을 body 태그 하단에 붙여 넣어 SDK를 사용해야 했다. 사용자가 한 번도 플레이하지 않아 점수가 기록되어 있지 않거나, 이전 점수를 넘은 경우 랭킹에 추가하거나 갱신되도록 구현하였다. 랭킹을 불러오는 경우 orderByChild로 오름차순으로 db의 snapshot을 전달 받아 배열로 저장하고 역순으로 만들어 출력하여 구현하였다.

대학생 모드와 대학원생 모드에서 게임이 종료되고 restart 하기 전에 score를 saveRank(mode, score)에 전달 하여 현재의 점수를 랭킹에 기록하도록 구현했다.

텍스트이(가) 표시된 사진

자동 생성된 설명2-4. 설정 기능 구현

설정 기능 중 color picker와 기록 저장 유무 변경 기능은 다음과 같이 index.html에서 전역 변수로 설정하여 구현하였다. 색상 변경을 위한 Color picker는 spectrum(<https://seballot.github.io/spectrum/>) 을 이용하여 구현했으며, MIT License이기 때문에 소스 코드의 공개 없이 이용할 수 있었다.

2-5 대학생 모드. 타이머, 스코어링, 목숨 기능 및 출력

setInterval()를 이용해 타이머를 움직이다. 0부터 시작하고 하나씩 증가한다. 그리고 난이도를 증가할 때 다시 0부터 시작한다. 스코링을 (ball.dy = -ball.dy; b.status--;)공이 벽돌과 충돌 때 마다 하나씩 증가한다. 못숨을 이 조건으로 (ball.y + ball.radius + ball.dy > canvas.height) 공을 놓칠 때 목숨 하나 없다. showGameStats(text, textX, textY) 함수로 타이머, 스코링, 목숨 기능을 출력한다.

2-6. 마우스 위치에 따른 패드 움직임

document.addEventListener("mousemove", mouseMoveHandler, false);를 이용해 마우스가 캔버스에서 패드너비의 절반 만큼 안에 있으면 그에 따라 움직이도록 했다.

2-7 대학생 모드. 공의 움직임

위, 옆, 아래(목숨 잃음)의 벽 충돌, 벽돌 충돌, 패드 충돌 시의 공의 움직임을 처리하였다. 위의 벽과 충돌 시 공의 **dy\*=-1**으로, 옆의 벽과 충돌시에는 **dx\*=-1**으로 방향을 전환시킨다. 공이 아래 벽과 충돌하면 LIFE—으로 생명을 감소시킨다.

패드와 충돌 시(if문을 이용해 공의 위치가 패드의 위치 내부와 겹치면)에는 패드 중심으로부터 얼마나 떨어진 부분에 충돌했는지에 따라 dx는 sin을 dy는 cos연산으로 공의 움직임의 각도를 정해준다.

**var collidePoint = ball.x - (paddle.x + paddle.width / 2);**

**collidePoint = collidePoint / (paddle.width / 2);**

**var angle = (collidePoint \* Math.PI) / 3;**

**ball.dx = ball.speed \* Math.sin(angle);**

**ball.dy = -ball.speed \* Math.cos(angle);**

2-8 대학생 모드. 아이템

5초 마다 나오는 아이템을 패드를 움직여 획득(효과: 패드의 길이 증가, 공의 크기 증가)할 수 있다. 아이템 Math.random()으로 아이템의 생성위치, 떨어지는 속도, 아이템의 종류 결정한다. item.status로 아이템 종류 정보 저장하며 0이면 획득되거나 사라진 아이템, 1이면 공 크기 증가, 2이면 패드 길이 증가를 나타낸다. 아이템 배열에 주기마다 새 아이템 추가하며, status가 0인 아이템은 배열에서 제거한다. 5초 주기는 setInterval()로 설정하였다.

2-9 대학생 모드. 난이도

1에서 3단계까지 구성되어 있으며, 모든 벽돌을 없애면 다음 단계로 진행된다.

**function showGameStats(text, textX, textY) {**

**for (var r = 0; r < brick.row; r++) {**

**for (var c = 0; c < brick.column; c++) {**

**isLevelDone = isLevelDone && bricks[r][c].status === 0;**

**}**

벽돌을 다 제거된 것을 확인하고 난이도를 증가한다. 난이도에 따라 벽돌 출력하기 위해 loop() 함수를 구현했다.

2-10 대학생 모드. 벽돌 생성

기본 벽돌은 4x8, 총 32개로 이루어져 있으며 각 단계에서 일정 주기 마다 벽돌을 생성한다. 벽돌에는 사용자가 현재 수강증인 과목 중에서 무작위로 선택되어 수강과목명이 표시되며 단계에 따라 내구도가 2인 어려운 과제 블럭을 생성한다.

1단계: 생성주기 35초, 내구도2 벽돌 X,

2단계: 생성주기 32초, 내구도2 벽돌 2개

3단계: 생성주기 30초, 내구도2 벽돌 3개

**bricks[r][c] = {**

**x: c \* (brick.padding + brick.width) + brick.offSetLeft,**

**y: r \* (brick.padding + brick.height) + brick.offSetTop,**

**status: 1,**

**name: subject[randNum]**

**}**

기본 벽돌은 위의 코드를 통해 간격에 맞게 위치 좌표(x,y) 설정되며, status는 내구도를 나타낸다.

추가 벽돌 생성은 아래의 코드와 같이 남아 있는 벽돌을 brick.padding + brick.height만큼 내린 다음, 맨 윗줄에 추과 과제 블럭을 출력한다. 2, 3단계에서는 추가 벽돌 중 2개(3개)는 내구도가 2인 벽돌을 랜덤으로 생성한다.

**for (var r = brick.row - 1; r > 0; r--) {**

**for (var c = 0; c < brick.column; c++) {**

**bricks[r][c] = {**

**x: bricks[r - 1][c].x,**

**y: bricks[r - 1][c].y + (brick.padding + brick.height),**

**status: bricks[r - 1][c].status,**

**name: bricks[r - 1][c].name**

**};**

**}**

**}**

벽돌의 내구도를 2로 설정하여 위치하는 코드는 아래와 같다.

**for (var i = 0; i < 2; i++) {**

**enhanced[i] = Math.floor(Math.random() \* 8);**

**for (var j = 0; j < i; j++) {**

**if (enhanced[i] === enhanced[j]) {**

**i--;**

**}**

**}**

**}**

2-11 대학생 모드. 게임 종료 조건

모든 벽돌 제거(3단계까지 클리어)되거나 벽돌이 패드에 닿거나 목숨을 모두 잃게 되면 게임을 종료하도록 구현했다.

**if (LEVEL >= MAX\_LEVEL) {**

**showYouWin();**

**GAME\_OVER = true;**

**return;**

**}**

2-12 대학생 모드. 루프

공의 움직임 및 출력, 패드 움직임 및 출력, 안 깨진 벽돌 출력, 충돌 이벤트들, 목숨유무, 레벨 클리어 여부, 아이템 움직임 및 획득은 매순간 변화하고 체크해주어야 하는 요소들이므로 아래와 같은 loop 함수를 구현하여 실행되도록 구현했다. 아이템 생성, 추가 벽돌 생성 등은 setInterval()을 이용하여 각자 주기에 맞추어 반복한다.

**function loop(){**

**requestAnimationFrame(loop);**

**}**

2-13 대학원생 모드. 스테이지의 흐름

**관련 전역 변수**

**var score = 0; // 점수**

**var resources = 150; // 학점(자원)**

**var gameTime = 5; // 게임 시간 : 60초**

**var frame = 0; // frame 100 = 1초보다 조금 느림**

**var start; // 게임 시작 시간 (start = new Date();)**

**var gameStatus = 0; // 게임 상태 : 0(게임 중), 1(패배), 2(승리)**

**var level = 0; // 게임 난이도 : 0(게임 시작 전), 1(easy), 2(normal), 3(hard)**

로그인 후 [대학원생 모드] 버튼은 누르면 게임 설명 알림창이 뜬 후 startGame() 함수를 불러오게 된다.

**function startGame() {**

**gameStatus = 0; // 게임 상태 : ‘게임 중’으로 변경**

**frame = 0;**

**start = new Date(); // 게임 시작 시간 저장**

**if (level === 0) {**

**level = 1; // easy 모드**

**}**

**draw();**

**}**

draw()는 화면 출력, 게임 동작 등을 모두 실행시키는 함수이다.

승리와 패배, 난이도 증가는 게임 상태를 의미하는 전역 변수 **gameStatus**에 의해 조절되며, 함수 handleGameStatus()에 의해 게임 승리 또는 패배 시 동작을 실행하게 된다.

**var gameStatus = 0; // 게임 상태 : 0(게임 중), 1(패배), 2(승리)**

**var level = 0; // 게임 난이도 : 0(게임 시작 전), 1(easy), 2(normal), 3(hard)**

**function handleGameStatus() {**

**if (gameStatus === 1) { // 패배 시**

**level = 0; // 레벨 초기화 (게임 시작 전 상태로 되돌림)**

**restart(); // 초기화 함수**

**alert("GAME OVER");**

**score = 0; // 점수 초기화**

**menuContainer.style.display = "flex";**

**mode2.style.display = "none";**

**}**

**if (gameStatus === 2) { // 승리 시**

**if (level === 1) { // easy 모드 승리**

**level++; // -> normal 모드**

**restart();**

**alert("LEVEL 1 CLEAR\n(확인을 누르면 LEVEL 2 시작)");**

**startGame(); // 게임 시작**

**} else if (level === 2) { // normal 모드 승리**

**level++; // -> hard 모드**

**restart();**

**alert("LEVEL 2 CLEAR\n(확인을 누르면 LEVEL 3 시작)");**

**startGame(); // 게임 시작**

**} else if (level === 3) { // hard 모드 승리**

**level = 0; // 게임 시작 전 상태로 되돌리기**

**restart();**

**alert("GAME COMPLETE\n획득 점수 : " + score + "점");**

**score = 0; // 점수 초기화**

**menuContainer.style.display = "flex";**

**mode2.style.display = "none";**

**}**

**}**

**}**

2-14 대학원생 모드. 터렛

터렛은 총 5가지로, 배열을 생성한 뒤 인덱스로 각각 접근하여 사용하였다.

**관련 변수**

**var defenders = []; // 생성된 터렛 객체 저장**

**var price = [30, 60, 90, 120, 150]; // 터렛 각각의 가격**

**var power = [20, 10, 20, 30, 40]; // 학문수양+20학점, 터렛의 공격력들**

터렛의 기본 틀로 Defender 클래스를 생성하고, 이 클래스를 상속하여 각 터렛의 기능을 구현하는 클래스 5개를 생성하였다. 터렛들은 “click”에 대한 EventListener와 handleDefenders() 함수를 통해 생성, 동작 실행, 제거된다.

상속된 클래스들의 함수

1) **draw**() : 각 터렛에 맞는 색과 명칭 출력

2) **update**() : 각 터렛의 동작 구현

2-14-1. Study 클래스 : 학문 수양

update() 함수가 불려질 때마다 멤버 변수 second의 값이 +1되는데, 이 값을 150으로 나눴을 때 0인 경우 학점에 20을 더한다. (3초에 한번씩 +20학점)

**update() {**

**this.second++;**

**if (this.second % 150 === 0) {**

**resources += power[0]; // 0 : 학문 수양 터렛의 고유 번호**

**}**

2-14-2. char, short, int ,double 클래스

update() 함수가 호출될 때마다 멤버 변수 timer의 값이 +1되는데, 이 값을 100으로 나눴을 때 0인 경우 코드(발사체)를 발사한다. (2초에 한번씩 코드 발사)

**projectiles.push(new Projectile(this.x + 70, this.y + 40, power[터렛 고유 번호]));**

**// 각 공격력에 맞는 발사 코드 생성**

2-15 대학원생 모드. 터렛의 생성

사용자가 메뉴에서 터렛을 클릭한 후, 생성하고자 하는 영역에 클릭을 하였을 때 터렛이 생성된다. 마우스 클릭을 통해 동작되므로 **canvas.addEventListener("click", function () {…** 내부에 해당 기능을 구현하였다.

**var tNum = 0; // 선택한 터렛의 번호를 저장하는 전역 변수**

tNum은 초기에 0으로 초기화되며, 사용자가 메뉴에서 터렛을 선택한 경우 그 터렛의 고유 번호를 저장하여 생성한 후에 다시 0으로 초기화된다.

EventListener 내 함수는 다음과 같은 순서로 진행된다

**if (tNum !== 0) { // 메뉴에서 터렛을 선택한 후에만 작동**

**1. 마우스의 x 좌표를 이용하여 터렛이 생성될 수 있는 좌표 계산**

**2. 마우스의 y 좌표를 이용하여 터렛이 생성될 수 있는 좌표 계산**

**(터렛 생성구역을 벗어났다면 터렛 생성 불가능)**

**(이미 터렛이 생성된 곳엔 터렛 생성 불가능)**

**3. 가진 resource가 선택한 터렛의 비용보다 같거나 큰지 확인**

**4. 현재 tNum에 맞는 터렛 객체 생성 (defenders[] 배열에 추가)**

**5. resource –= 선택 터렛의 비용**

**6. tNum = 0; // 사용자 선택 초기화**

**}**

2-16 대학원생 모드. 터렛의 출력과 제거

**handleDefenders()** 함수를 통해 터렛을 출력 및 제거한다. 함수가 호출되면 defenders[] 배열에 저장되어 있는 모든 터렛을 다시 그리고, 각 기능(학점 추가 또는 총알 생성)을 수행하도록 한다. 적과 터렛이 충돌했을 경우 터렛의 체력을 감소시키고, 체력이 0이 되었을 경우 해당 터렛을 defenders[] 배열에서 삭제한다.

2-17 대학원생 모드. 터렛 발사체 구현

**class Projectile {**

**constructor(x, y, power) {this.power = power; this.speed = 5; }**

**update() {this.x += this.speed;}**

**draw() { }**

**}**

constructor(x, y, power)에서 입력 받은 power를 통해 터렛의 공격력을 결정한다. update ()를 통해 발사체가 speed만큼 x좌표를 우측으로 움직이고, draw()로 Projectile의 형태를 캔버스에 그려 발사체를 구현하였다.

2-18 대학원생 모드. 발사체 이동, 삭제

projectiles[i].**update**(): 발사체의 x좌표를 speed만큼 우측으로 움직인다.

projectiles[i].**draw**(): 변경된 x좌표를 이용하여 projectiles[i]의 형태를 캔버스에 그린다.

**if (enemies[j] && projectiles[i] && collision(projectiles[i], enemies[j])) {**

**enemies[j].health -= projectiles[i].power;**

**projectiles.splice(i, 1);**

**i--;**

**}**

collision()을 통해 발사체와 적 충돌을 감지한다. 충돌 시 적의 체력을 projectiles[i]의 공격력만큼 감소시키며 이후 발사체는 제거된다.

**if (projectiles[i] && projectiles[i].x > canvas.width - 40) {**

**projectiles.splice(i, 1);**

**i--;**

**}**

발사체가 격자무늬를 벗어날 시 제거하는 코드이다.

2-19 대학원생 모드. Enemy(과제) 구현

**class Enemy {**

**constructor(verticalPosition, index) {this.y = verticalPosition;**

**this.speed = speed[level -1];**

**this.movement = this.speed;**

**this.health = enemyHealth[index];**

**this.maxHealth = this.health;**

**this.index = index;**

**}**

**update() {this.x -= this.movement;}**

**draw() {var image = new Image();**

**image.src = enemyImg[this.index];**

**ctx.drawImage(image, this.x, this.y);**

**}**

**}**

constructor(verticalPosition, index)에서 입력 받은 verticalPosition을 통해 y좌표를 결정한다. 속도는 레벨에 따라, 체력은 입력 받은 index를 통해 결정된다. update()를 통해 과제가 movement만큼 x좌표를 좌측으로 움직인다. draw()에서는 index에 의해 캔버스에 그려지는 Enemy의 형태가 결정된다.

2-20 대학원생 모드. Enemy(과제) 생성, 이동, 삭제 : **handleEnemies()**

enmies[i].**update**(): 발사체의 x좌표를 movement만큼 좌측으로 움직인다.

enemies[i].**draw**(): 변경된 x좌표를 이용하여 enemies[i]의 형태를 캔버스에 그린다.

if (enemies[i].x < x) {gameStatus = 1;}: 적이 왼쪽 벽에 닿으면 게임이 종료된다.

**if (enemies[i].health <= 0) {**

**var gainedResources = enemies[i].maxHealth / 10;**

**resources += gainedResources;**

**score += gainedResources;**

**enemies.splice(i, 1);**

**i--;**

**}**

**var now = new Date();**

**var sec = Math.floor((now - start) / 1000); : 게임 후 지난 시간**

과제의 체력이 0이하면 과제가 생성되었을 때의 체력을 10으로 나눈 만큼 학점과 점수가 증가된다. 이후 과제는 제거된다.

**if (frame % 300 === 0) {**

**let verticalPosition = Math.floor(Math.random() \* 6) \* height + 3 \* y + scoreHeight;**

**var num;**

**if (sec <= 10) {num 2;}**

**else if (sec <= 30) {num = 4;}**

**else {num = 5;}**

**let randomIndex = Math.floor(Math.random() \* num);**

**enemies.push(new Enemy(verticalPosition, randomIndex));**

**}**

frame % 300 === 0을 통해 약 6초에 한 번씩 과제를 생성한다. 과제의 y좌표인 verticalPosition은 랜덤하게 설정된다. if (sec <= 10) {num 2;}로 게임 시작 후 10초 이전에는 [PYTHON], [SWIFT]만 생성한다. else if (sec <= 30) {num = 4;}로 30초 이전에는 [PYTHON], [SWIFT], [JAVA], [R]만 생성한다. 30초 이후에는 모든 종류의 적이 생성된다. randomIndex는 정해진 num에 따라 랜덤하게 설정된다. 이후 verticalPosition과 randomIndex를 통해 과제를 생성한다.

2-20 대학원생 모드. 충돌 기준 : collision(first, second)

**if( !( first.x > second.x + second.width ||**

**first.x + first.width < second.x ||**

**first.y > second.y + second.height ||**

**first.y + first.height < second.y))**

**{return true;}**

접촉하지 않는 조건에 논리연산자! 을 사용하여 접촉하면 collision(first, second)가 true를 반환하게 하였다.

**3. 토의 사항**

**강관주**

터렛, 발사체, 적 생성은 쉬웠으나, 삭제하는 방법이 떠오르지 않아서 오류가 많았다. 하지만, 배열을 전역으로 선언하고 그 배열에 객체들을 집어넣고, 삭제한다는 방법을 사용한 후로는 훨씬 쉬워졌다. 충돌 후 삭제를 어디서 할지 정하는 것도 생각보다 까다로웠다. 게임 종료, 재시작 과정에서 초기화에 어려움을 겪었다. 패배나 단계에 따라 초기화 해야 하는 변수를 팀원과 상의하여 정해서 마무리할 수 있었다.

**서동균**

게임 내 요소들에 대한 연산, 사용자 조작에 의한 이벤트 처리, 그리고 그것들에 따른 실시간 화면 출력, 이 세 요소를 함께 생각하며 프로그램을 구현하는 것이 어려웠다. 특히 디버깅 시, 내부 연산 문제인지, 화면 출력의 문제인지를 특정하기가 쉽지 않아 더욱 그랬다. 벽돌 추가 생성 기능을 처음 구현할 때 기능이 잘 작동하지 않았는데, 내부 연산 부분에서의 문제라 생각하여 많은 시간을 할애했지만, 출력 부분에서의 문제였다. 한편으로는 세 가지 요소를 반복처리 하하는 것이 힘들었는데, 기능에 따라 setInterval(), setTimeout(), 재귀를 이용한 requestAnimationFrame() 알맞게 선택하여 사용해야만 했다.

**전진호**

API 단계에서 사용자의 평문 패스워드가 url 상에서 노출이 되기 때문에 보안이 취약하게 되어 url encoding을 거쳤으면 좋지 않았을까 하는 아쉬움이 든다. 서버에서는 계정 정보를 저장하여 보관하지 않지만, 다른 사람들이 사용했을 때는 자신의 계정 정보가 유출된다고 생각할 수 있다는 걱정도 된다. API에서 JSESSIONID를 부여받은 뒤에 user 정보 요청을 보냈을 때 세션을 잃어버리는 경우가 있었는데, 이에 대한 원인을 찾지 못한 것도 아쉽다. 클라이언트에서 여러번 요청을 보내는 것으로 해결을 했지만, 만약 사용자가 많아지게 되면 그에 따른 요청이 배로 늘어나게 된다는 문제점이 있다. Firebase 프로젝트를 추가하기 위해 body 태그 하단에서 cdn으로 불러왔는데, 보안 정보가 그대로 노출되어지는 문제도 아쉽다고 생각한다.

하나의 프로젝트를 여럿이서 협업하는 방법에 대해서도 많은 고민을 했다. Git을 사용하여 각 기능마다 브랜치를 나누어 협업하는 것이 일반적인 방법이지만 짧은 기간 내에 git을 활용하기는 어렵다고 생각했다. 그래서 codesandbox라는 online code editor로 각자 실시간으로 코드를 작성하는 협업 방식을 채택했는데, 아주 마음에 들었다. 구현해야할 기능들도 적절히 분배했기 때문에 결론적으로 협업이 잘 이루어진 것 같다.

**정은아**

처음에 구현을 기획했을 때는 이 프로그램을 과연 짤 수 있을지 막연 했으나 팀원과 함께 만들어가니 완성할 수 있었다. 구현 중 가장 힘들었던 부분은 게임 승리 또는 패배 시 동작이었다. 코드가 길다 보니 게임을 다시 시작하기 위해서는 초기화 함수를 어디에 넣어야 정상 작동을 하는지, 어느 요소를 초기화하고 어느 요소를 남겨야 할지 헷갈려서 코드를 다시 한번 정리 했어야 했다.

가장 아쉬운 부분은 학문 수양 터렛이 3초마다 학점을 더하는 부분이다. 게임 시간 측정하는 것처럼 시간을 정확히 측정하는 것으로 구현하려고 했는데, 이렇게 하면 학점이 3초마다 딱딱 20학점씩 더해지는 것이 아니라 이상한 숫자가 좌르륵 더해졌다. 그래서 어쩔 수 없이 멤버 변수 second을 통해 구현하게 되었다.

그래도 이번 조별 과제를 통해 혼자서는 절대로 구현하지 못했을 게임을 구현해서 뿌듯하다.

**파리나**

각 함수를 구현 때 다른 부분과 잘 연결하기 위해 좀 어려웠다. 그리고 아이템 구현할 때 setInterval() 사용하기 위해 아직도 문제가 있다. 아직도 그거에 대한 완전히 이해하지 않은 것을 알게 되었다. 그리고 난이도를 증가할 때 출력하는 부분을 반복할 수 있도록 많은 문제 생겼어요. 여러 것을 시도할 수 있어서 수업 때 배운 것을 연습도 할 수 있고 새로운 것도 배울 수 있다.

**4. 기여도**

**강관주 20%**

* class Enemy
* class Projectile
* drawPattern1()
* drawPattern2()
* drawPattern3()
* handleGameGrid()
* handleProjectiles()

**정은아 20%**

* class Cell
* class Defender
* drawScore()
* printTime()
* drawTime()
* drawMenu()
* printResources()

**강관주, 정은아 공동 구현**

* drawSetting()
* handleDefenders()
* handleEnemies()
* handleGameStatus()
* startGame()
* restart()
* collision()
* mouse 이벤트 처리

**서동균 22%**

- 아이템: 5초 마다 나오는 아이템을 패드를 움직여 획득(효과: 패드의 길이 증가, 공의 크기 증가)

- 벽돌 생성: 기본 벽돌 4x8개, 단계에 따른 일정 주기 마다 벽돌 생성, 벽돌에 무작위 수강과목명 표시, 단계에 따라 내구도2인 벽돌도 생성

- 게임 종료 조건: 벽돌이 패드에 닿음

- 마우스 위치에 따른 패드 움직임

- 공의 움직임: 위, 옆, 아래(목숨 잃음)의 벽 충돌, 벽돌 충돌, 패드 충돌

**파리나 15%**

- 타이머, 스코어링, 목숨 기능 및 출력

- 난이도: 1~3단계, 모든 벽돌을 없애면 다음 단계로

- 게임 종료 조건: 모든 벽돌 제거(3단계까지 클리어) ,목숨을 모두 잃음

- 마우스 위치에 따른 패드 움직임

- 공의 움직임: 위, 옆, 아래(목숨 잃음)의 벽 충돌, 벽돌 충돌, 패드 충돌

**전진호 23%**

* 학사정보시스템 API 서버 구현 : <https://github.com/Jinho1011/KuHub-API>
* 랭킹 시스템 구현 : ranking.js, Firebase 사용
* 설정 기능 구현 : setting.js
* 메뉴 기능 구현 : menu.js
* UI/UX 디자인 : Figma 사용<https://www.figma.com/file/OEmdiuRmPrGWnnuzonWUu4/%EC%9B%B9%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D?node-id=149%3A2>
* 퍼블리싱 : index.hmtl, index.css, menu.css, reset.css 작성