
MADE BY SANG HYEOK KIM
GAME LAND
PORTFOLIO
START

INDEX

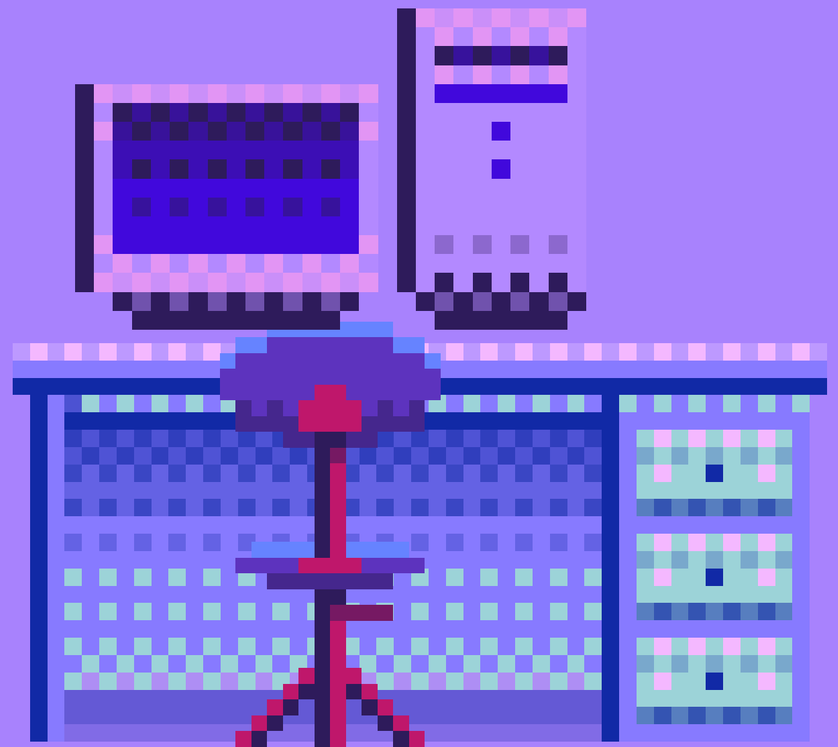
프로젝트 소개

What is Game Land?

1. PURPOSE OF PROJECT

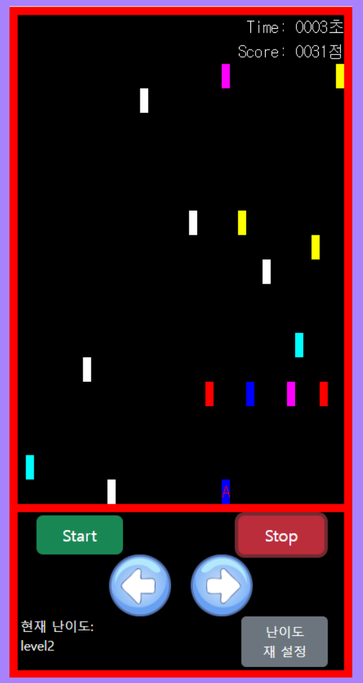
보유한 기술을 기반으로

사용자가 직접 플레이 가능한 게임 홈페이지 구현

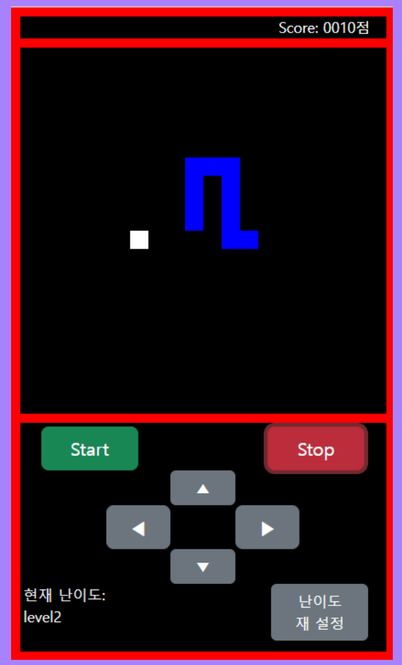


2. CORE OF PROJECT

AVOID GAME



SNAKE GAME



USER RANKING

미니 게임 플레이 기능

게임 기록 랭킹화

3. DEVELOPMENT ENVIRONMENT

eclipse(2022-12)

Java(jdk17)

Oracle 21c XE

Spring Boot(3.1.0)

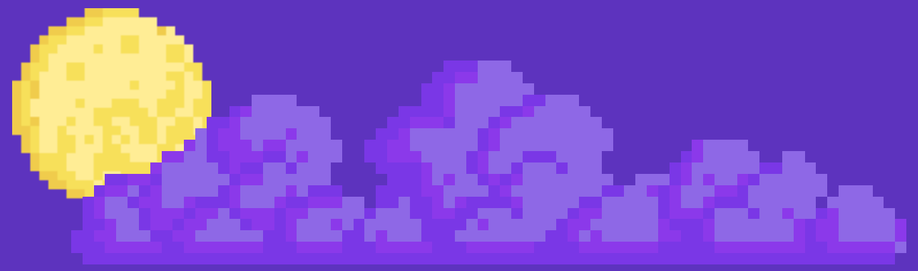
3. DEVELOPMENT ENVIRONMENT - DETAIL

Plug in
- STS(Spring Tool Suite)
- Tern, Node.js

웹 페이지 구현
- html5
- css3 & bootstrap5
- ECMAscript6

라이브러리 및 프레임 워크
- Gradle(Build Tool)
- Spring Security(보안 관련 처리)
- 암호화: BCryptPasswordEncoder
- Spring Web(Tomcat 사용)
- Lombok(@data와 같은 Annotation)
- Validation(사용자 입력 데이터 검증)
- Page Helper(랭킹 페이지 구현)
- JSP

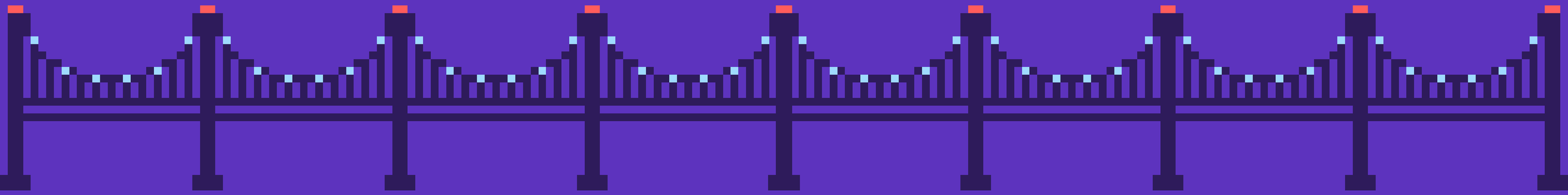
- Scott 유저 생성
- SQL Developer 사용
=>테이블 생성 및
테스트 진행



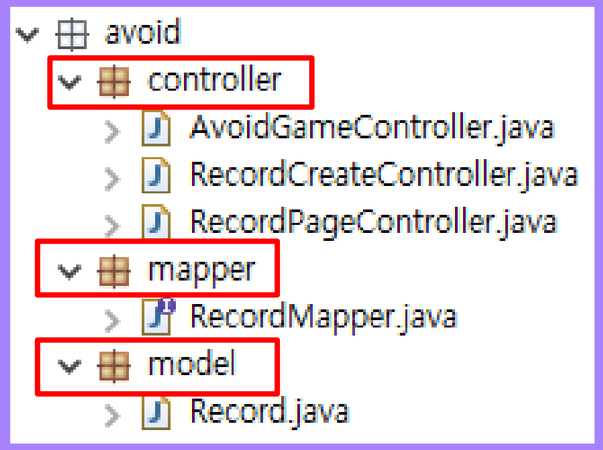
02

프로젝트 구성도

Let's see a project!



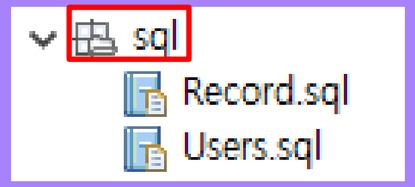
MVC MODEL



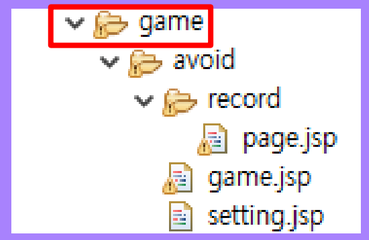
CONTROLLER

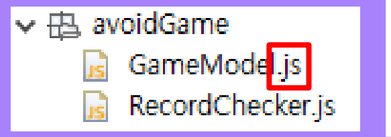
MODEL

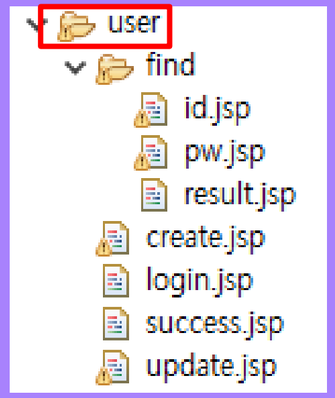
DB



VIEW

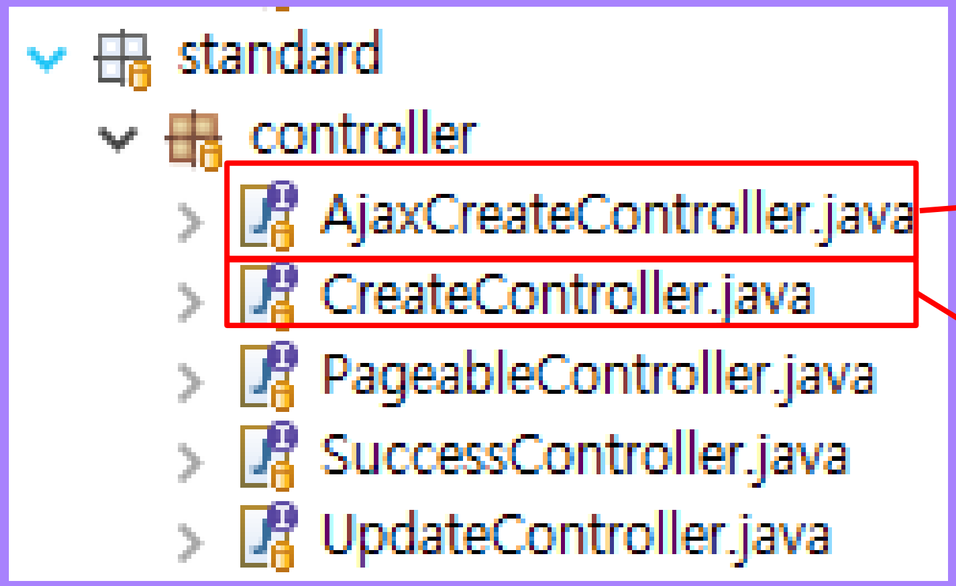






MVC MODEL

STANDARD CONTROLLER




```java
//Ajax로 받은 데이터 처리 관련 interface
public interface AjaxCreateController<T,K,V> {

    @GetMapping("/create")
    void create(Model model, HttpServletRequest request);

    @PostMapping("/create")
    ResponseEntity<T> create(@RequestBody Map<K, V> requestBody);

}
```

```java
// 계정 생성 관련 interface
public interface CreateController<DTO> {

    @GetMapping("/create")
    void create(Model model, HttpServletRequest request);

    @PostMapping("/create")
    String create(@Valid DTO dto, BindingResult binding, Model model, HttpServletRequest request, RedirectAttributes attr);

}
```

Standard Controller interface를 구현
=> 표준에 맞추어 구현 클래스를 생성함

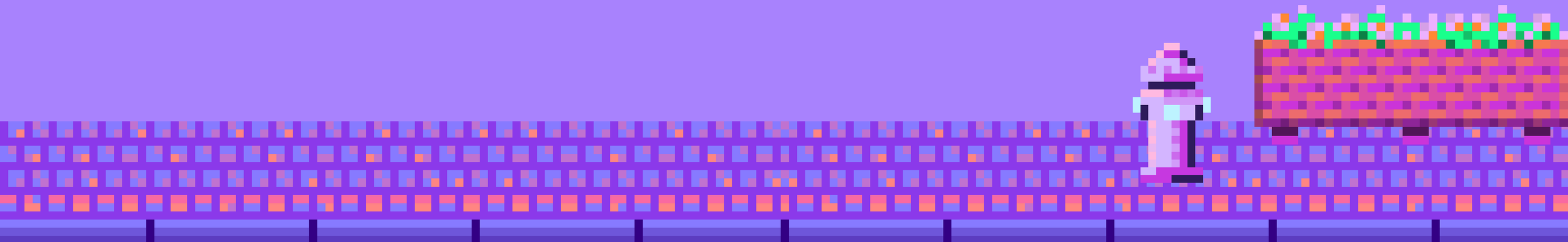

03
핵심 코드 소개
프로젝트 소개
데이터 파이프라인

GAME ALGORITHM

1. AVOID GAME

```
//run 메소드 반복 수행
timerChecker[this.blockNumber] = this.timer = setTimeout(() => {
    this.run(object, timerChecker, target, width, height, blockMap,
}, this.cycle);
```

setTimeout으로
run 메소드 반복 호출



GAME ALGORITHM

1. AVOID GAME

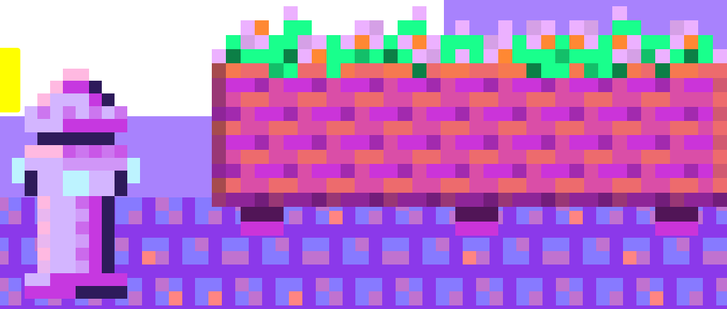
```
//반복적으로 수행되면서 캐릭터와 block들을 조작하는 메소드
run(object, timerChecker, target, width, height, blockMap, columnChecker, elapseTimeId, recordChecker, setFinalScore, finalScoreTag) {

    //블록을 1칸 아래로 이동 후 바뀐 위치를 blockMap에 저장
    this.hide(object);
    this.line++;
    if(blockMap[this.line-1][this.column-1]==null){
        blockMap[this.line-2][this.column-1]=null; //캐릭 있던 곳을
        blockMap[this.line-1][this.column-1]=this;
    } else { //캐릭터와 부딪힌 경우 게임 종료
        for(let i=0; i<timerChecker.length; i++){
            clearTimeout(timerChecker[i]);
        }
        clearInterval(elapseTimeId);
        this.gameEnd(recordChecker);
        return;
    }
    object.style.left = width*(this.column-1) + 'px';
    object.style.top = height*(
    this.show(object);
```

이전 위치의 HTML 태그의 display 속성을 ‘none’으로 하여 숨김

위치 정보는
20X40의 2차원 배열인
blockMap에 저장

이동한 위치에 캐릭터가
존재할 경우 게임 종료

현재 위치의 HTML 태그의 display 속성을 ‘inline’으로 하여 화면에 출력



GAME ALGORITHM

2. SNAKE GAME

```
//isStart가 true일 때만 run 메소드를 반복 수행
if(this.isStart){
    objects[objects.length-1].timerId = setTimeout(()=>{
        this.run(objectTags, objects, target, t, width, height, blockMap, cycle)
    }, cycle);
}
```

setTimeout으로
run 메소드 반복 호출

GAME ALGORITHM

2. SNAKE GAME

```
//반복적으로 수행되면서 캐릭터와 target을 조작하는 메소드
run(objectTags, objects, target, t, width, height, blockMap, cycle) {

    //Game객체들의 line과 column을 순서대로 저장(moveOthers 메소드에 대입하기 위함)
    let previousLine = [];
    let previousColumn = [];
    for(let e of objects){
        previousLine.push(e.line);
        previousColumn.push(e.column);
    }

    //objects배열의 첫 번째 객체인 head를 1칸 이동
    objects[0].moveHead(objectTags[0], objects, target, t, width, height, blockMap);

    //길이가 2이상이면 head가 아닌 나머지 객체들을 자신의 바로 앞의 line, c
    if(objects.length > 1)
        for(let i=1; i<objects.length; i++){
            objects[i].moveOthers(objectTags[i], previousLine[i-1], previousColumn[i-1], width, height, blockMap);
        }
```

캐릭터의 맨 앞은 방향에 따라 1칸 이동,

나머지 블록들은
자신의 바로 앞의 블록의 위치로 이동

GAME ALGORITHM

2. SNAKE GAME

```
//캐릭터가 어떤 방향으로 target에 접근했는가에 따라 조건처리하여 새로운 Game 객체를 return
getNewCharacter(object) {
    let character = new Game(object.line, object.column, object.direction);
    if(character.direction == "left") //왼쪽으로 접근 시 1칸 왼쪽에 새 블록 생성
        character.column -= 1;
    else if(character.direction == "top") //위로 접근 시 1칸 위쪽에 새 블록 생성
        character.line -= 1;
    else if(character.direction == "right") //오른쪽으로 접근 시 1칸 오른쪽에 새 블록 생성
        character.column += 1;
    else if(character.direction == "bottom") //아래쪽으로 접근 시 1칸 아래쪽에 새 블록 생성
        character.line += 1;
    return character;
}
```

Target에 부딪혔을 때
새로운 블록이 생성되는 방식

GAME ALGORITHM

3. DATA TRANSFER

```
//게임이 종료되면 실행되는 메소드
gameEnd(recordChecker) {

    //게임이 종료되었을 때 Start버튼과 Stop버튼 누르지 못하게 막음
    document.querySelector("#btnStart").disabled = "true";
    document.querySelector("#btnStop").disabled = "true";

    //게임 종료 관련 팝업 출력
    const popup = document.querySelector('.popup');
    popup.style.display = 'inline';

    //username, level, score, elapseTime에 랭킹 관련 기록들을 담아서 서버로 전송
    const url = '/game/avoid/record/create';
    const data = {
        userName: recordChecker.userName,
        level: recordChecker.level,
        score: recordChecker.score-1, //부딪힐 때도 점수가 1점 추가되기 때문에 -1(맵핑처리)
        elapseTime: recor
    };
    const options = {
        method: 'POST',
        headers: {
            'Content-Type
        },
        body: JSON.stringify(data)
    };

    //로그인해야 정보가 서버로 전송되도록 조건 처리
    if(recordChecker.userName != "anonymousUser")
        fetch(url, options)
            .then(response => console.log(response))
            .catch(error => {
                console.log(error)
            });
}
```

게임 종료 시 로그인 한 유저의 기록이
Ajax 방식으로 전송
(fetch 메소드 사용)

```
@Override
public ResponseEntity<String> create(Map<String, String> requestBody) {
    var userName = requestBody.get("userName");
    var level = Integer.parseInt(requestBody.get("level"));
    var score = Integer.parseInt(requestBody.get("score"));
    var elapseTime = Integer.parseInt(requestBody.get("elapseTime"));
    var record = Record.builder()
                        .userName(userName)
                        .score(score)
                        .elapseTime(elapseTime)
                        .gameLevel(level)
                        .build();

    //조건처리: 각 레벨 당 데이터를 최대 100개로 유지(랭킹 순).
    var minRecord = mapper.selectLastRowOfxLevel(level);
    var cnt = mapper.countAllOfxLevel(level);
    if(cnt<100)
        mapper.insertRecord(record);
    else {
        if(minRecord.getScore()<score || minRecord.getScore()==score && minRecord.getElapseTime()<elapseTime) {
            mapper.deleteMinRecordOfxLevel(level);
            mapper.insertRecord(record);
        }
    }
    return ResponseEntity.ok("Data created successfully");
}
```

Model(Record)에 유저의 기록을 담음

Mapper로 DB에 데이터를 삽입

```
@Insert("""
        insert into record
        (user_name, score, elapse_time, game_level)
        values
        (#{record.userName}, #{record.score}, #{record.elapseTime}, #{record.gameLevel})
        """)
int insertRecord(@Param("record") Record record);
```

GAME ALGORITHM

3. DATA TRANSFER

```sql
-- Avoid Game 랭킹 테이블 제거
DROP TABLE RECORD;

-- Avoid Game 랭킹 테이블 생성
CREATE TABLE RECORD(
    USER_NAME    VARCHAR2(10 CHAR),
    SCORE        NUMBER,
    ELAPSE_TIME  NUMBER,
    GAME_LEVEL   NUMBER
);

-- Snake Game 랭킹 테이블 제거
DROP TABLE RECORD2;

-- Snake Game 랭킹 테이블 생성
CREATE TABLE RECORD2(
    USER_NAME    VARCHAR2(10 CHAR),
    SCORE        NUMBER,
    GAME_LEVEL   NUMBER
);

COMMIT;
```

DB, 기록 관리 테이블

```sql
DROP TABLE USERS;

CREATE TABLE USERS (
     ID           VARCHAR2(20) CONSTRAINT USERS_ID_PK PRIMARY KEY,
     PASSWORD     VARCHAR2(100 CHAR) NOT NULL,
     ROLE         VARCHAR2(20  CHAR),
     NAME         VARCHAR2(20  CHAR),
     PHONE_NUMBER CHAR(13 CHAR)
);

COMMIT;
```

DB, 유저 관리 테이블

GAME ALGORITHM

4. RANKING PAGE

```java
package com.example.standard.controller;

import org.springframework.ui.Model;

//페이지 처리 관련 interface
public interface PageableController {

    @GetMapping("/page/{pageNum}/{pageSize}")
    String page(HttpServletRequest request, @PathVariable("pageNum") int pageNum, @PathVariable("pageSize") int pageSize, Model model);
}

```

```java
//Avoid Game 랭킹 페이지 처리 Controller
@Controller
@RequestMapping("/game/avoid/record")
public class RecordPageController implements PageableController{

    @Autowired
    RecordMapper mapper;

    @Autowired
    ObjectMapper json;

    @Override
    public String page(HttpServletRequest request, int pageNum, int pageSize, Model model) {
        var level = Integer.parseInt(request.getParameter("level"));
        model.addAttribute("level", level);

        PageHelper.startPage(pageNum, pageSize);
        var list = mapper.selectPageByGameLevel(level);
        var paging = PageInfo.of(list, 10);
        model.addAttribute("list", list);
        model.addAttribute("paging", paging);

        return "game/avoid/record/page";

    }
```

url 정보로 페이지를 생성

```java
@Select("""
            select *
              from record
             where game_level = #{gameLevel}
             order by score desc,
                      elapse_time desc
        """)
Page<Record> selectPageByGameLevel(@Param("gameLevel") int gameLevel);
```

Mapper로 조회한 객체들을 list에 넣음

04

결과물 소개

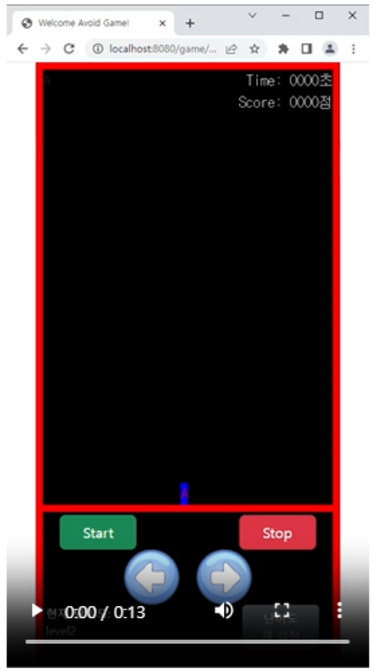
Let’s test a project!

초기 화면

Welcome Game Land!

*랭킹 등록을 원하시면 로그인 후 플레이 해주세요!

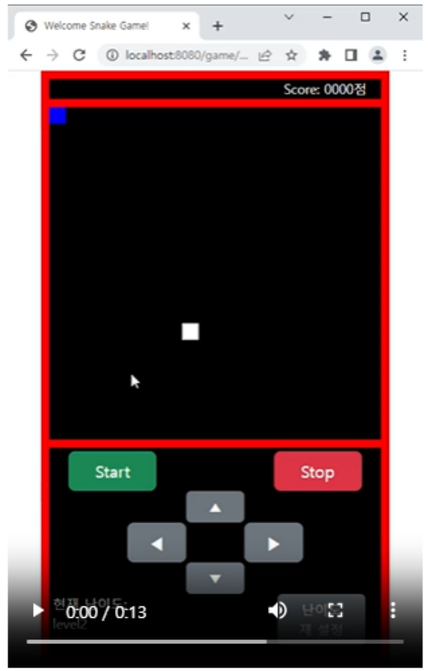
로그인



Avoid Game

떨어지는 블록에 부딪히지 말고
끝까지 생존하세요!

블록피하기 게임 플레이! 랭킹



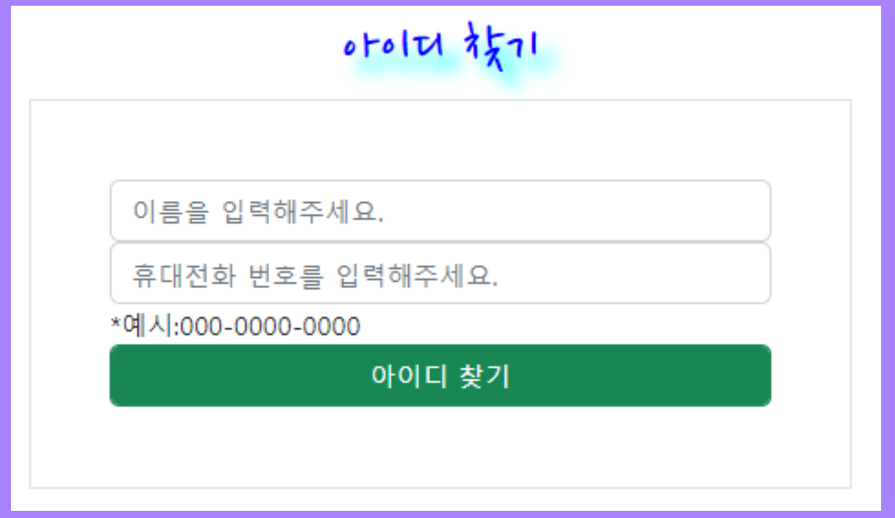
Snake Game

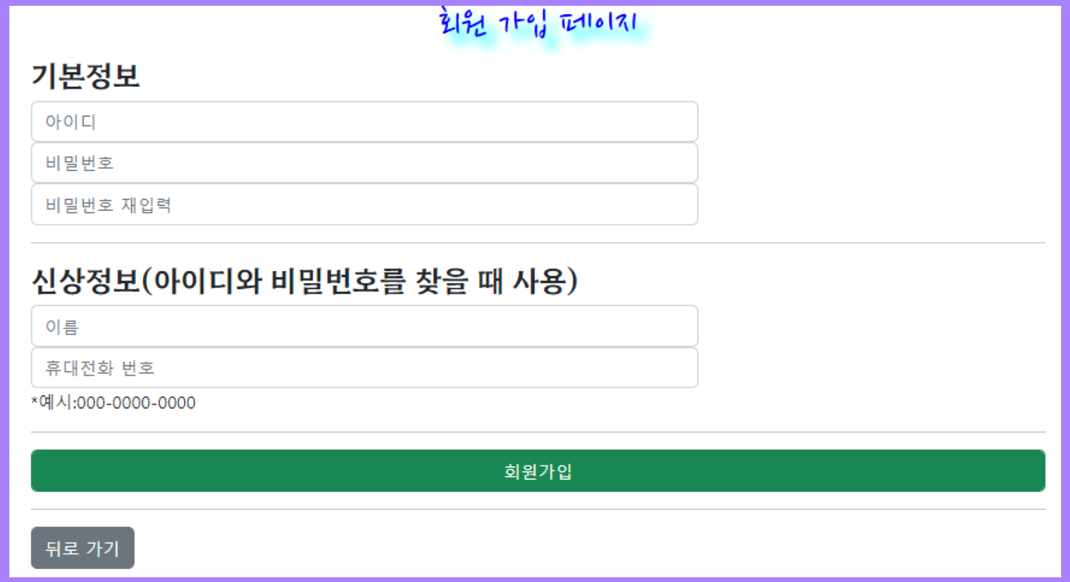
먹이를 찾아서 몸집을 키우세요!

스네이크 게임 플레이! 랭킹

계정 관련 페이지 1


Game Land Login
ID
아이디를 입력하세요.
Password
비밀번호를 입력하세요.
자동 로그인
로그인
회원가입
아이디 찾기
비밀번호 찾기


아이디 찾기
이름을 입력해주세요.
휴대전화 번호를 입력해주세요.
*예시:000-0000-0000
아이디 찾기


회원 가입 페이지
기본정보
아이디
비밀번호
비밀번호 재입력
신상정보(아이디와 비밀번호를 찾을 때 사용)
이름
휴대전화 번호
*예시:000-0000-0000
회원가입
뒤로 가기


비밀번호 찾기
아이디를 입력해주세요.
이름을 입력해주세요.
전화번호를 입력해주세요.
*예시:000-0000-0000
비밀번호 찾기

계정 관련 페이지 2

아이디 찾기

id 조회 결과입니다.

당신의 id는 java입니다.

로그인 페이지로 이동

비밀번호 찾기

임시 비밀번호를 발급해드렸습니다.

임시 비밀번호는 e75a4792 입니다.
로그인 하신 후 꼭 비밀번호를 변경해주시기 바랍니다.

로그인 페이지로 이동

회원정보 수정

기본정보

java

••••

••••

신상정보

java

000-0000-0000

*예시:000-0000-0000

회원정보 수정

메인 화면으로

회원 정보 수정이 완료되었습니다.

메인 화면으로

게임 소개 및 셋팅 페이지

Block 피하기 게임 세계에 오신 것을 환영합니다.

간단설명:

하늘에서 떨어지는 수 많은 블록들을 방향키를 이용해 피하시면 됩니다.

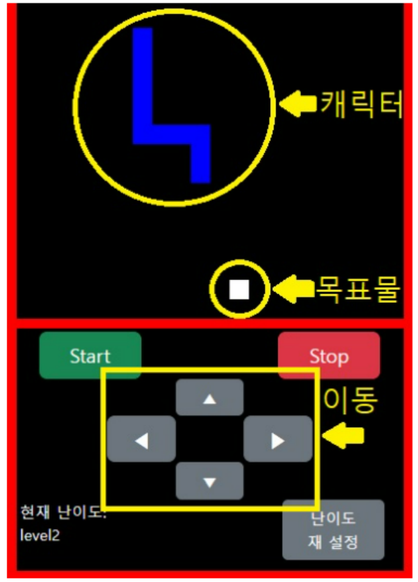
1. 가운데에 A문자가 적혀있는 블록이 캐릭터 입니다.
2. 화살표 버튼이나 키보드의 방향키로 캐릭터를 조작합니다.
3. 하늘에서 수 많은 블록들이 떨어집니다. 캐릭터와 부딪히면 게임종료.
4. 블록이 바닥까지 떨어졌을 때 점수가 1점 추가 됩니다.
5. 난이도는 3단계까지 있으며,
   난이도에 따라 블록의 속도와 수가 달라집니다.
6. 1단계: 쉬움, 2단계: 보통, 3단계: 어려움
7. Start 버튼을 눌러 게임을 시작하고,
   Stop 버튼을 눌러 일시정지 할 수 있습니다.
8. 난이도 선택 후 플레이 해주세요!

플레이 할 난이도를 선택해주세요.

Level1 Level2 Level3

메인 화면으로

Snake 게임 세계에 오신 것을 환영합니다.



간단설명:

목표물을 향해 이동하여 캐릭터의 몸집을 키우시면 됩니다.

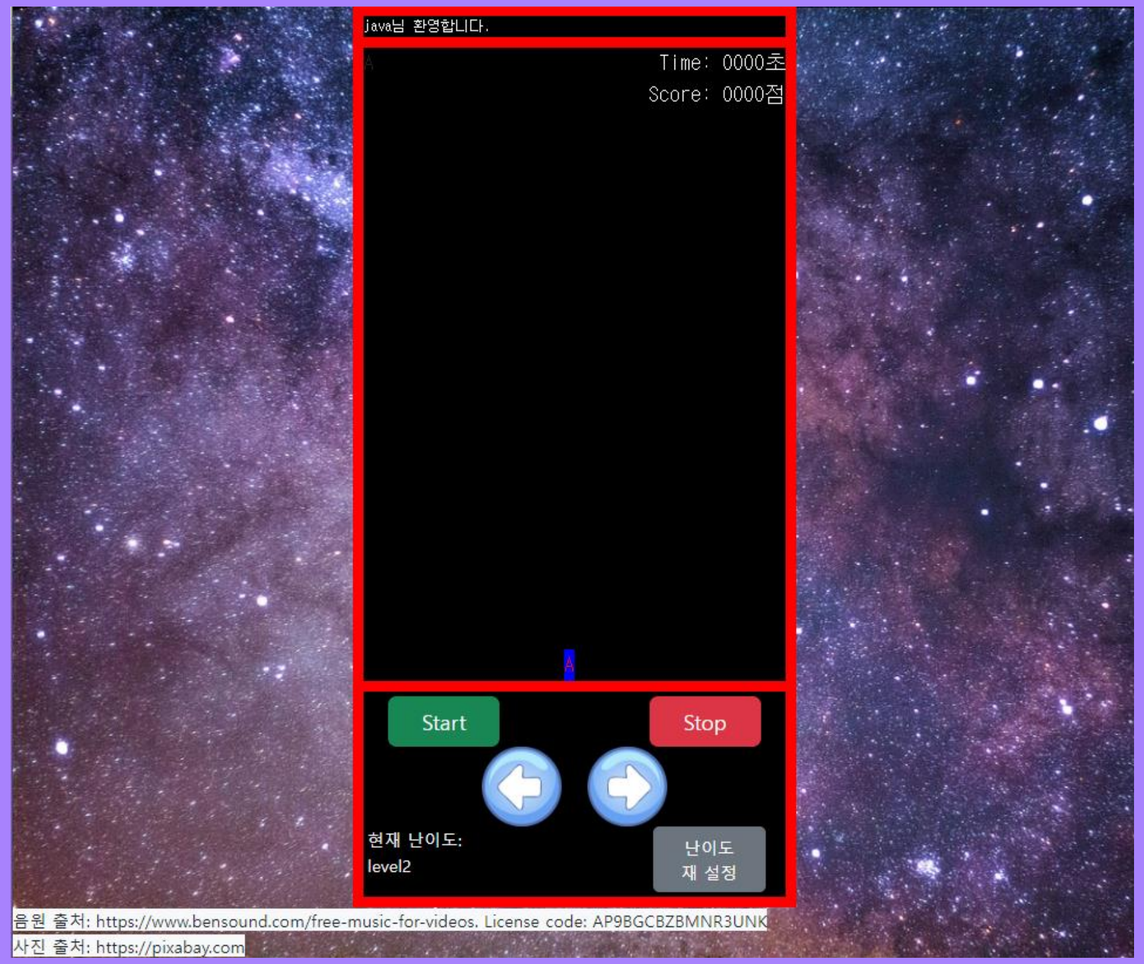
1. 파란색 블록이 캐릭터, 흰색 블록이 목표물입니다.
2. 게임이 시작되면 캐릭터는 일정한 속도로 이동합니다.
3. 화살표 버튼이나 키보드의 방향키로 캐릭터의 방향을 조절합니다.
4. 캐릭터가 목표물을 잡아 먹으면 길이가 길어지고, 1점이 추가됩니다.
5. 캐릭터가 벽에 부딪히거나, 자기 자신과 만나면 게임 종료.
6. 난이도는 3단계까지 있으며,
   난이도에 따라 캐릭터의 속도가 달라집니다.
7. 1단계: 쉬움, 2단계: 보통, 3단계: 어려움
8. Start 버튼을 눌러 게임을 시작하고,
   Stop 버튼을 눌러 일시정지 할 수 있습니다.
9. 난이도 선택 후 플레이 해주세요!

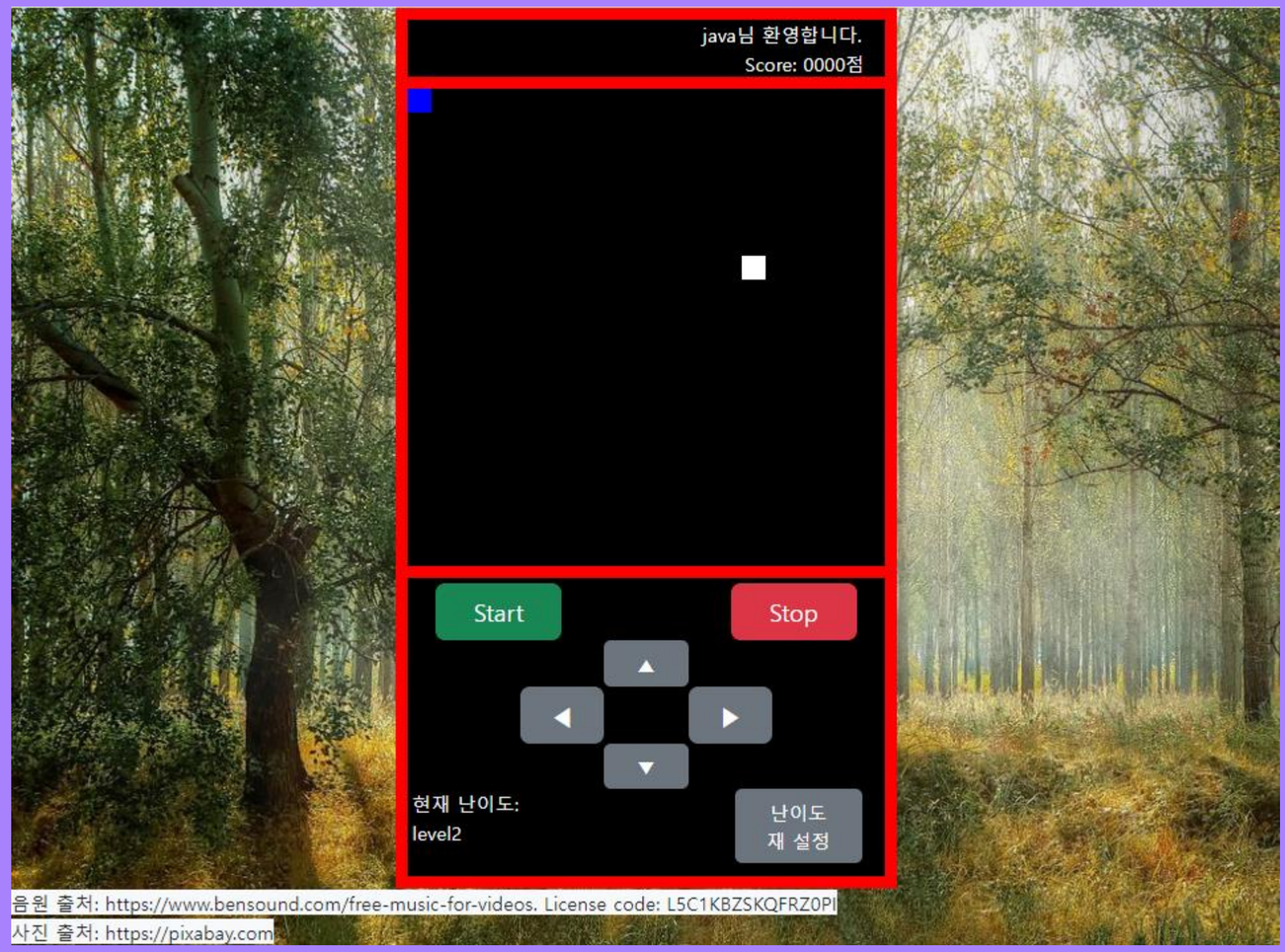
플레이 할 난이도를 선택해주세요.

Level1 Level2 Level3

메인 화면으로

게임 플레이 페이지


java님 환영합니다.
Time: 0000초
Score: 0000점
Start
Stop
현재 난이도:
level2
난이도
재 설정
음원 출처: https://www.bensound.com/free-music-for-videos. License code: AP9BGCBZBMNR3UNK
사진 출처: https://pixabay.com


java님 환영합니다.
Score: 0000점
Start
Stop
현재 난이도:
level2
난이도
재 설정
음원 출처: https://www.bensound.com/free-music-for-videos. License code: L5C1KBZSKQFRZ0PI
사진 출처: https://pixabay.com

게임 랭킹 페이지

TOP100 랭킹

*순위 산정 기준: 점수 높은 순, 점수 같을 시 생존 시간 높은 순

Level1 | Level2 | Level3

등수	닉네임	점수	생존 시간	난이도
1	java	251점	13초	3
2	java	249점	12초	3
3	java	209점	10초	3
4	java	184점	7초	3
5	java	69점	3초	3
6	java	50점	2초	3
7	java	34점	1초	3
8	java	33점	1초	3
9	java	31점	1초	3
10	java	26점	1초	3

<< Previous 1 2 Next >>

게임으로 메인 화면으로

AVOID GAME

TOP100 랭킹

Level1 | Level2 | Level3

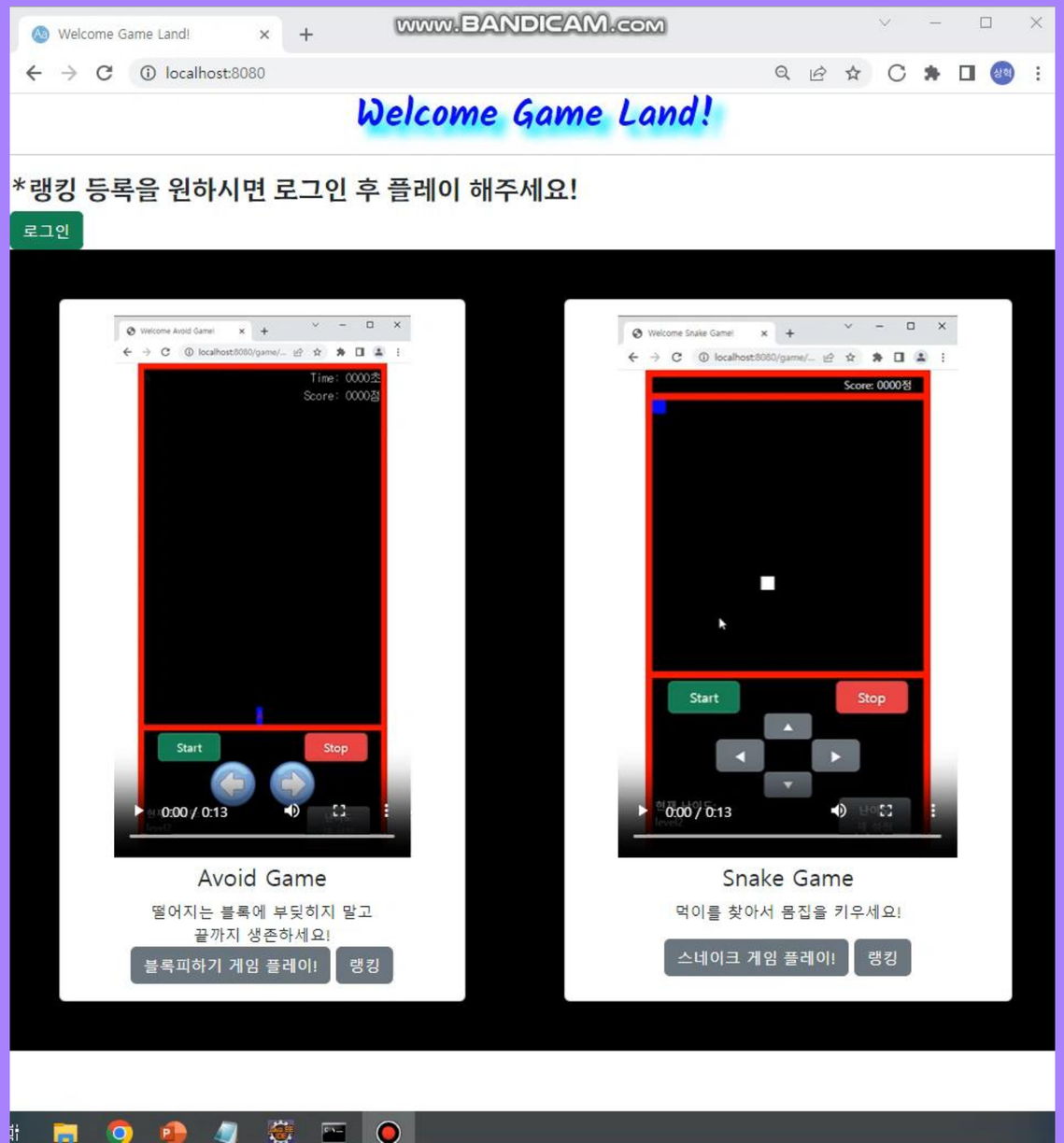
등수	닉네임	점수	난이도
1	java	17점	3
2	java	7점	3
3	java	7점	3
4	java	5점	3
5	java	5점	3
6	java	4점	3
7	java	3점	3
8	java	3점	3
9	java	2점	3
10	java	2점	3

<< Previous 1 2 Next >>

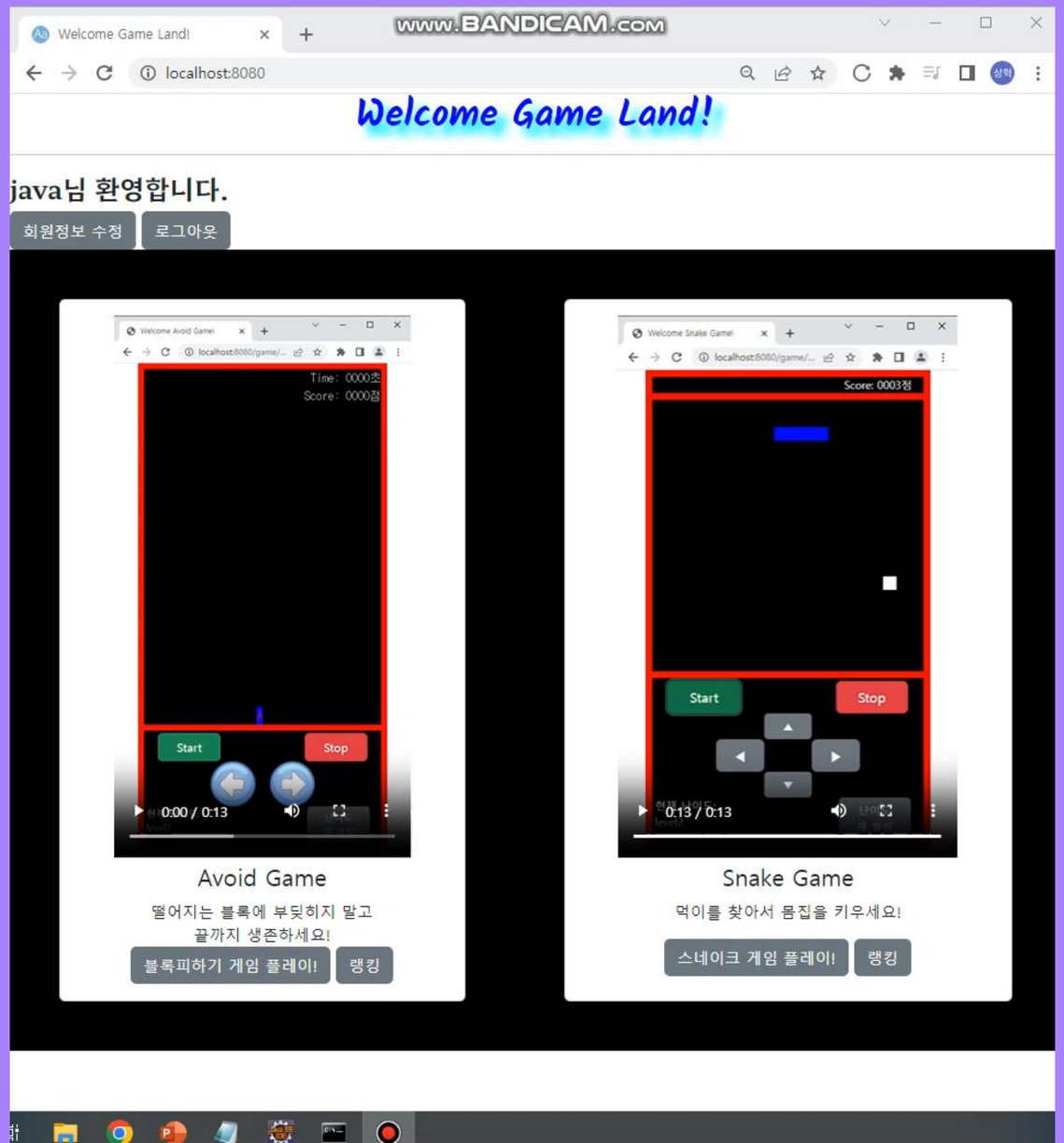
게임으로 메인 화면으로

SNAKE GAME

시연 동영상 1 계정 관련 조작



시연 동영상 ㄹ 게임 플레이




THANK YOU