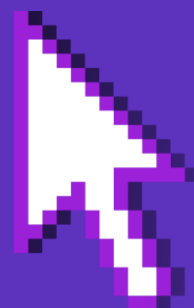


RELEASED



MADE BY SANG HYEOK KIM

# GAME LAND PORTFOLIO



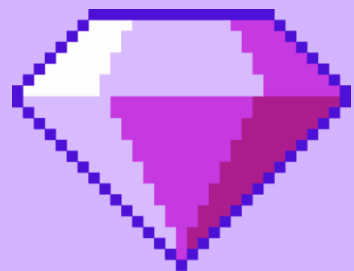
START



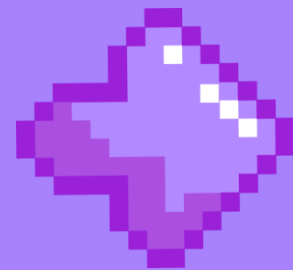
# INDEX

01

프로젝트 소개



프로젝트 목표  
프로젝트 핵심 내용  
개발 환경 및 도구

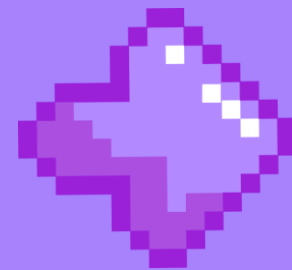


02

프로젝트 구성도



전체 구조 보기  
MVC MODEL  
DATA BASE

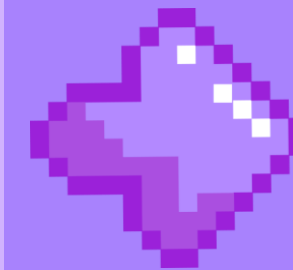


03

핵심 코드 소개



AVOID GAME  
SNAKE GAME  
DATA TRANSFER  
RANKING PAGE

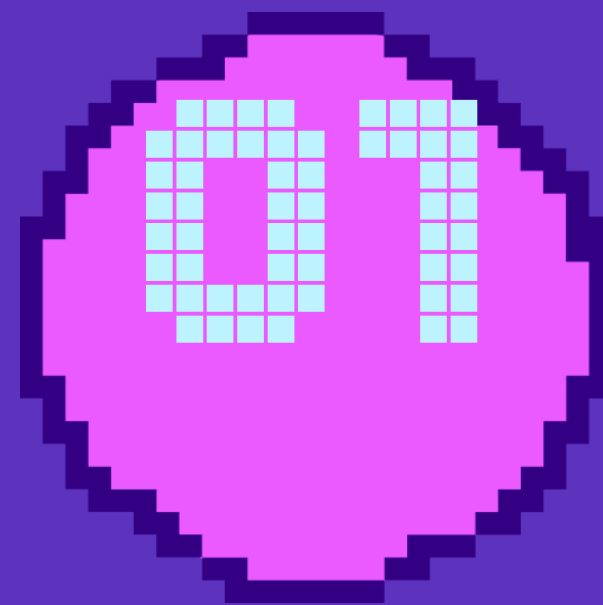


04

결과물 소개

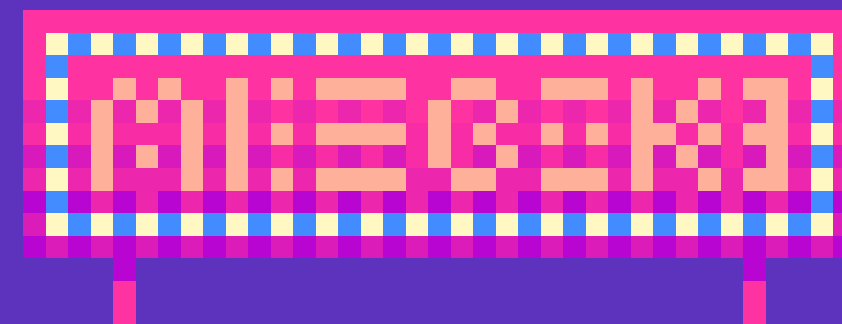
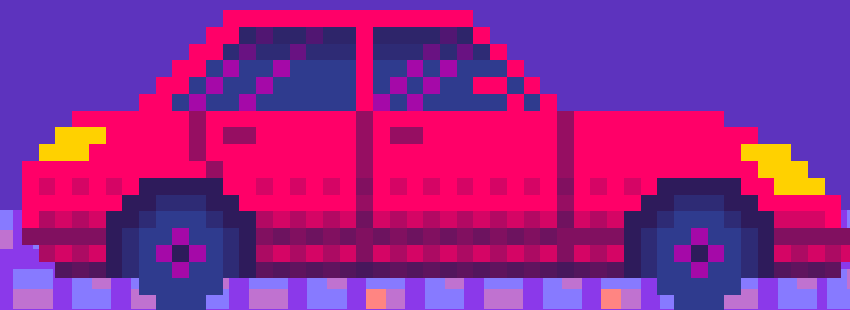
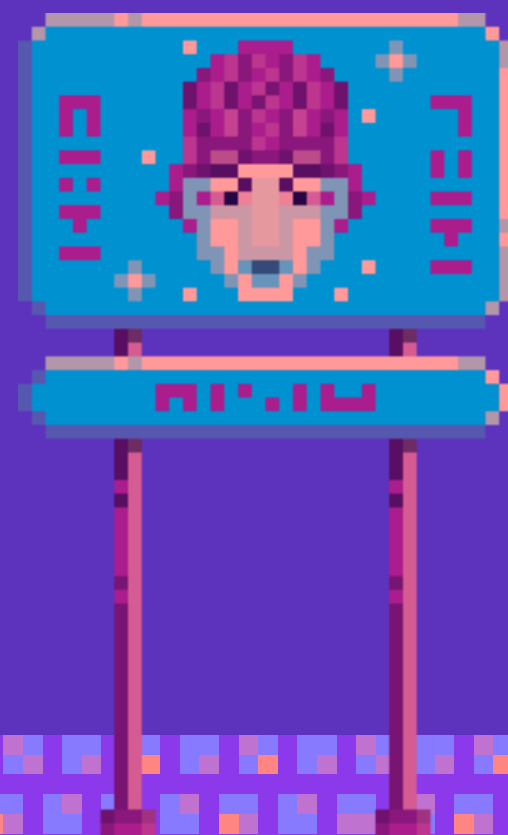


화면 구성  
시연 동영상



# 프로젝트 소개

What is Game Land?

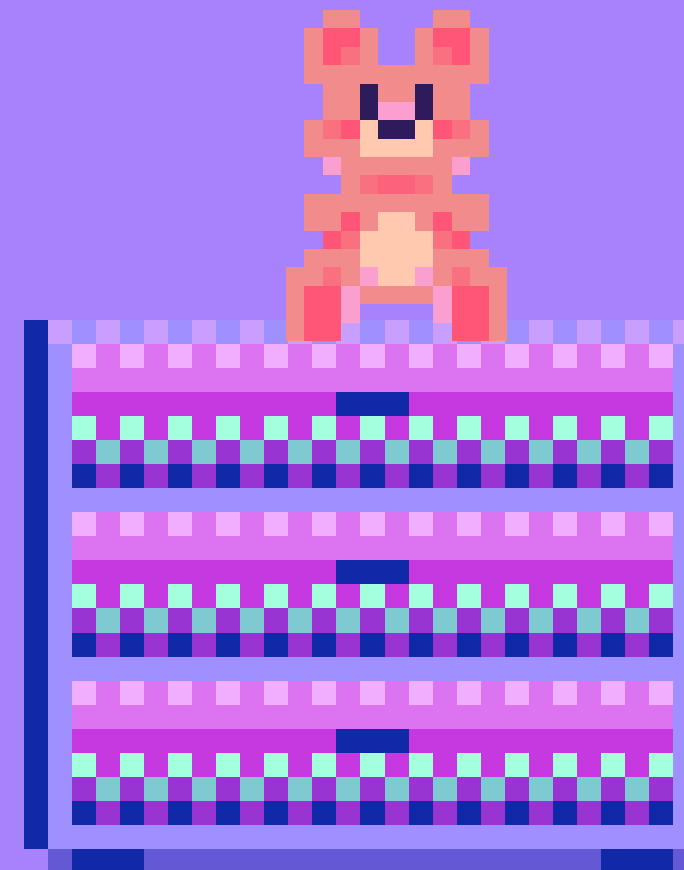
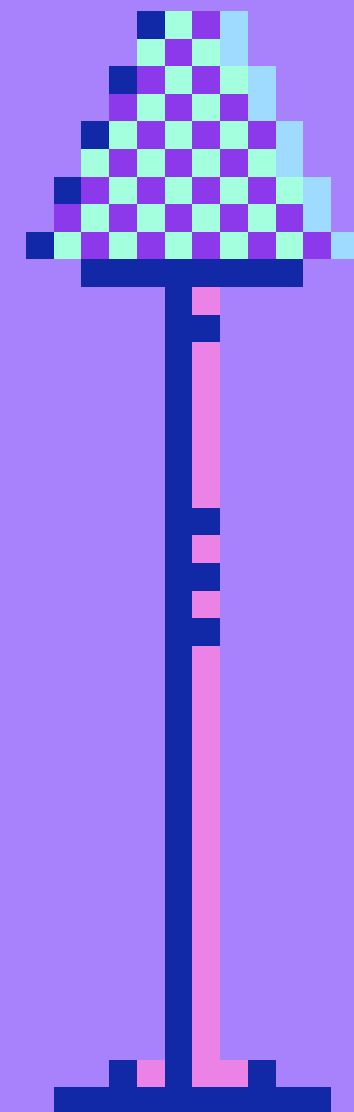
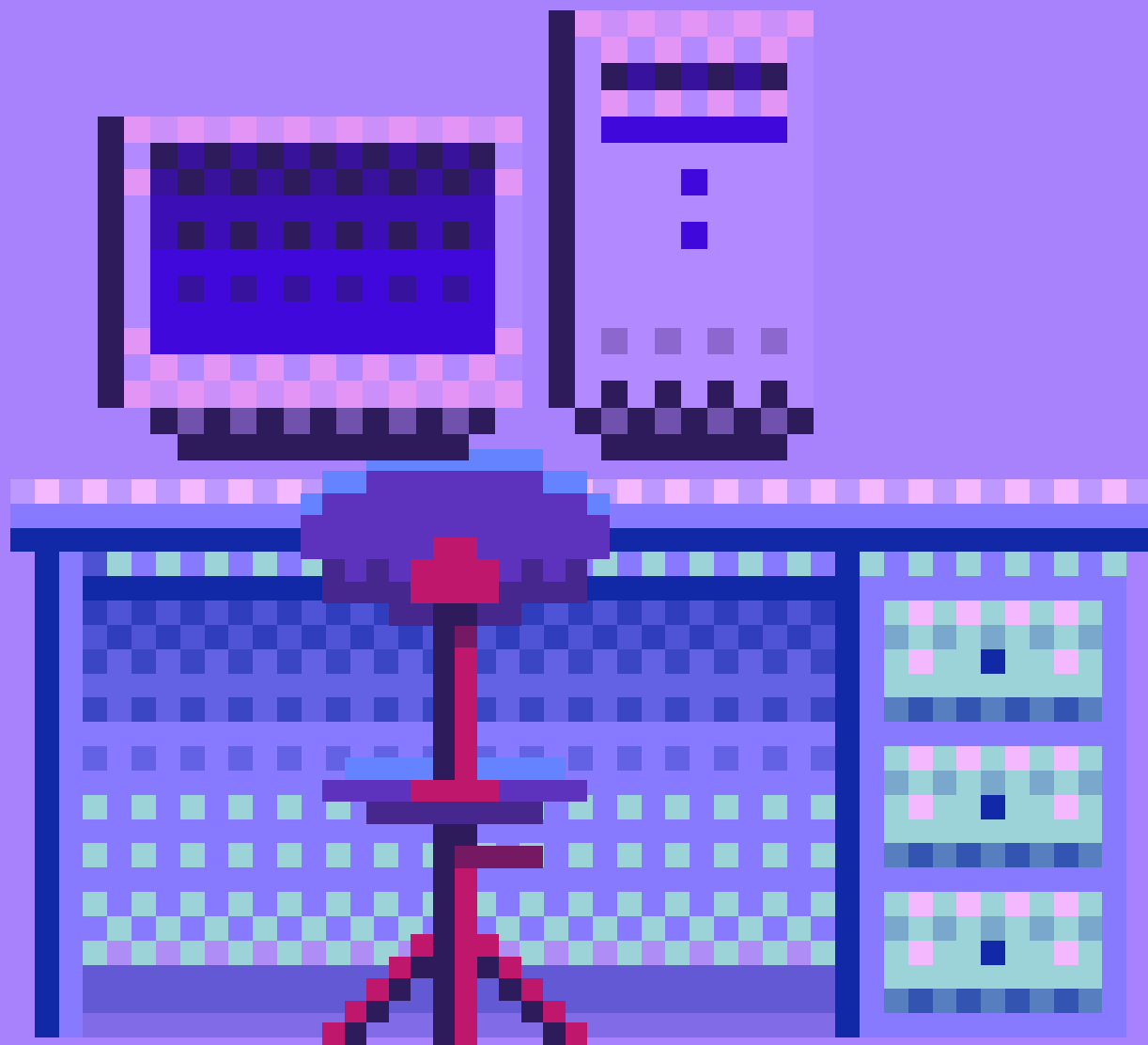


# 1. PURPOSE OF PROJECT

보유한 기술을 기반으로

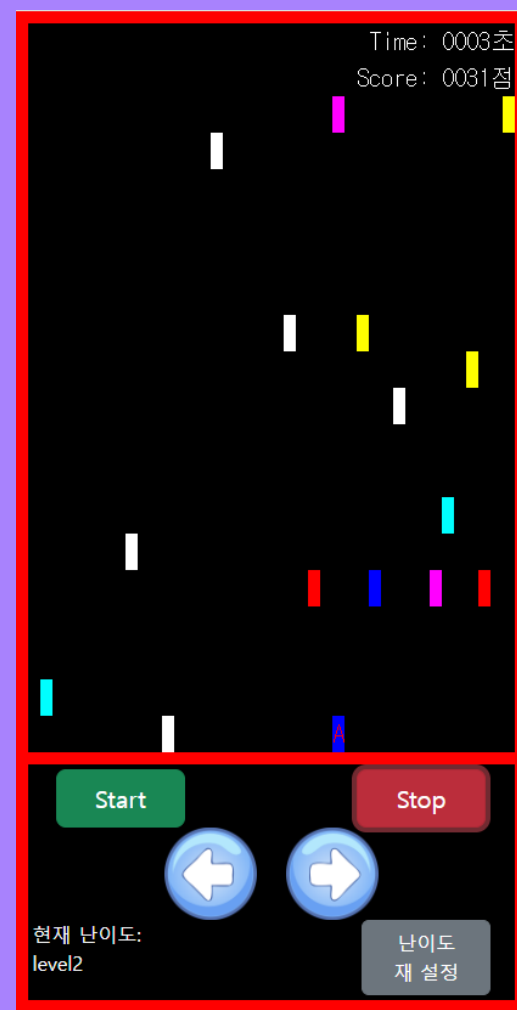


사용자가 직접 플레이 가능한 게임 홈페이지 구현



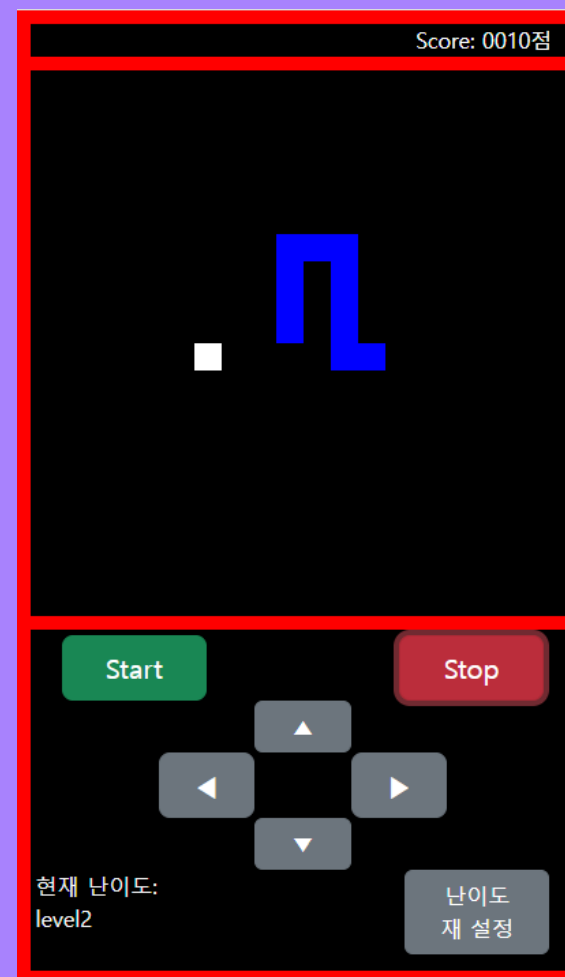
## 2. CORE OF PROJECT

AVOID  
GAME



미니 게임 플레이 기능

SNAKE  
GAME



USER

RANKING

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	254점	20초	2
2	test	133점	11초	2
3	test	114점	9초	2
4	test	107점	8초	2
5	java	78점	6초	2
6	test	67점	5초	2
7	java	66점	5초	2
8	test	65점	5초	2
9	java	47점	4초	2
10	java	44점	4초	2
<< Previous 1 2 Next >>				
게임으로		메인 화면으로		

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

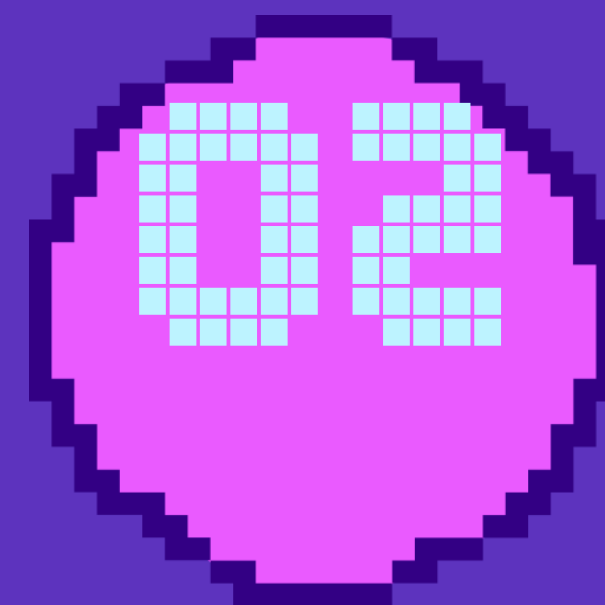
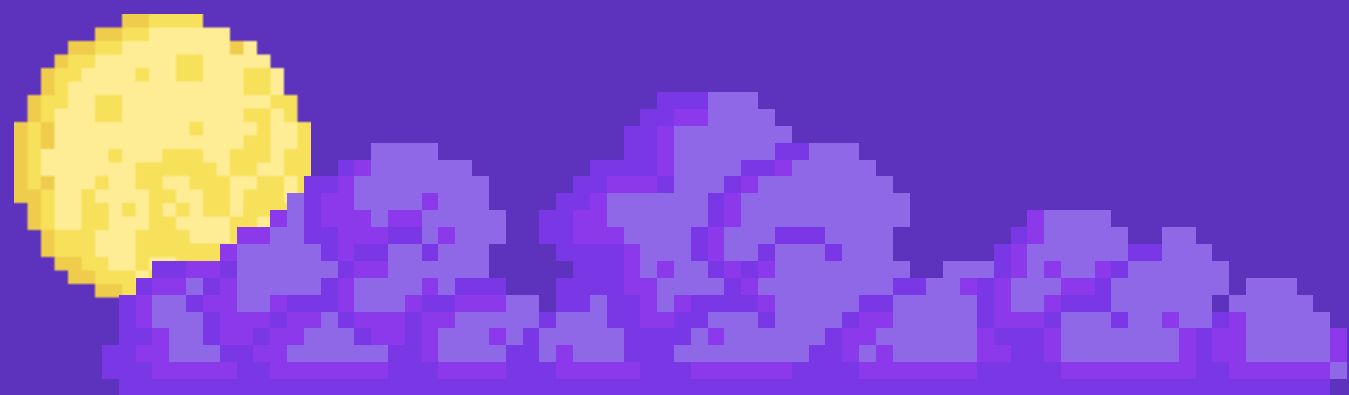
- Gradle(Build Tool)
- ORM: Mybatis
- Spring Security(보안 관련 처리)
- 암호화: BCryptPasswordEncoder
- Spring Web(Tomcat 사용)
- Lombok(@data와 같은 Annotation)
- Validation(사용자 입력 데이터 검증)
- Page Helper(랭킹 페이지 구현)
- JSP

ORACLE®



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

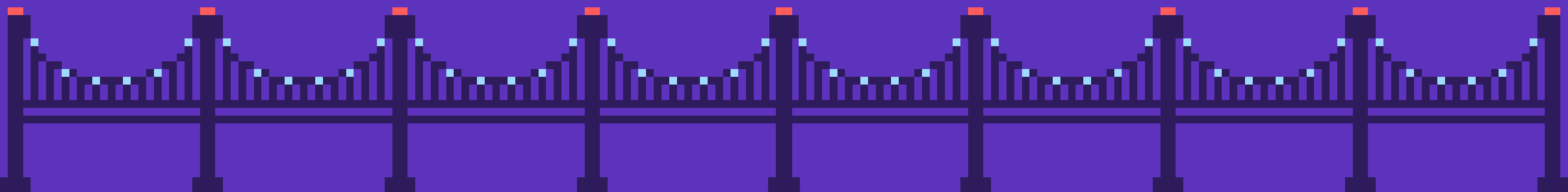




# 프로젝트 구성도

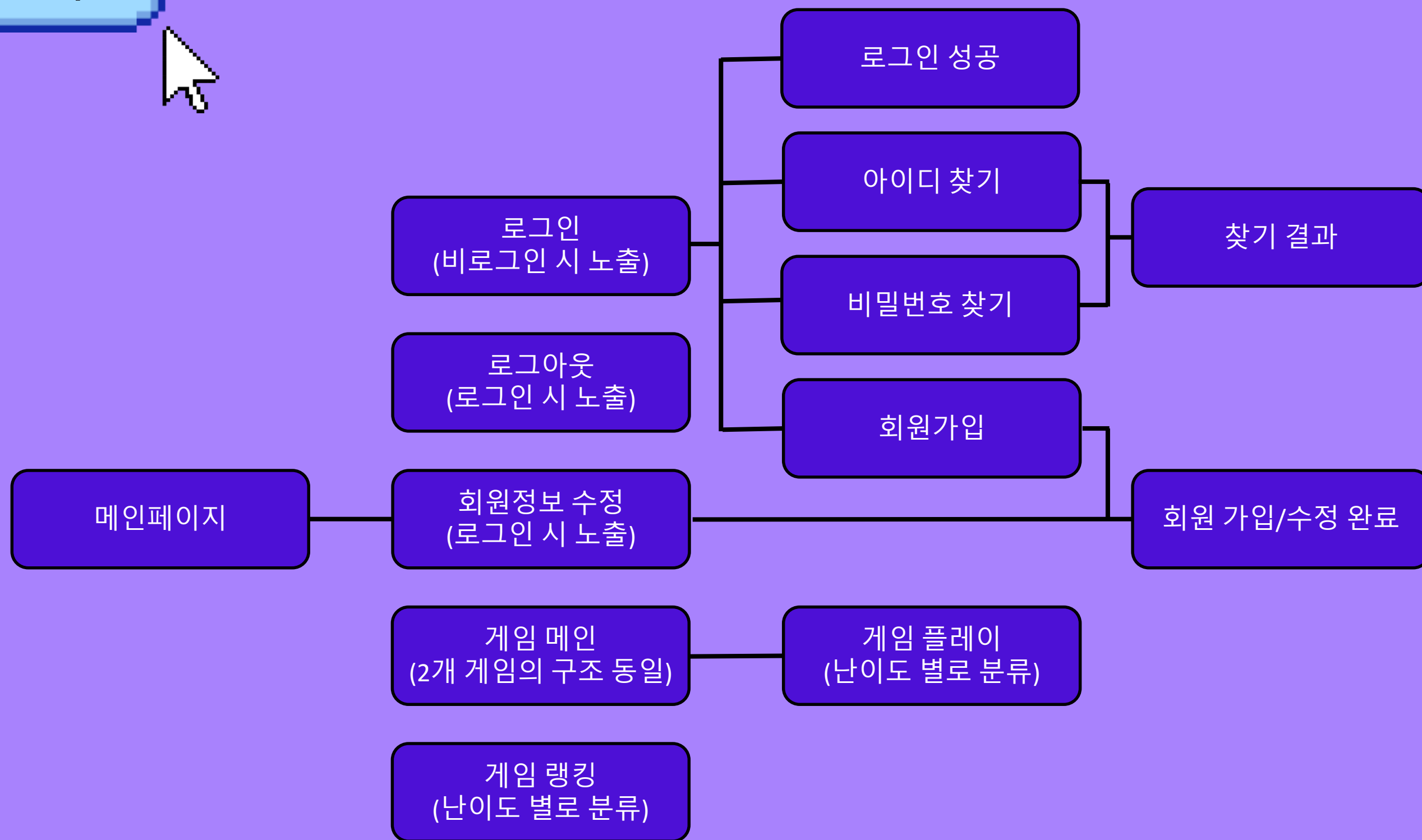


Let's see a project!





## 전체 구조 보기



## MVC MODEL

CONTROLLER

DB

MODEL

VIEW

- avoid
  - controller
    - AvoidGameController.java
    - RecordCreateController.java
    - RecordPageController.java
  - mapper
    - RecordMapper.java
  - model
    - Record.java

- sql
  - Record.sql
  - Users.sql

- game
  - avoid
    - record
      - page.jsp
      - game.jsp
      - setting.jsp

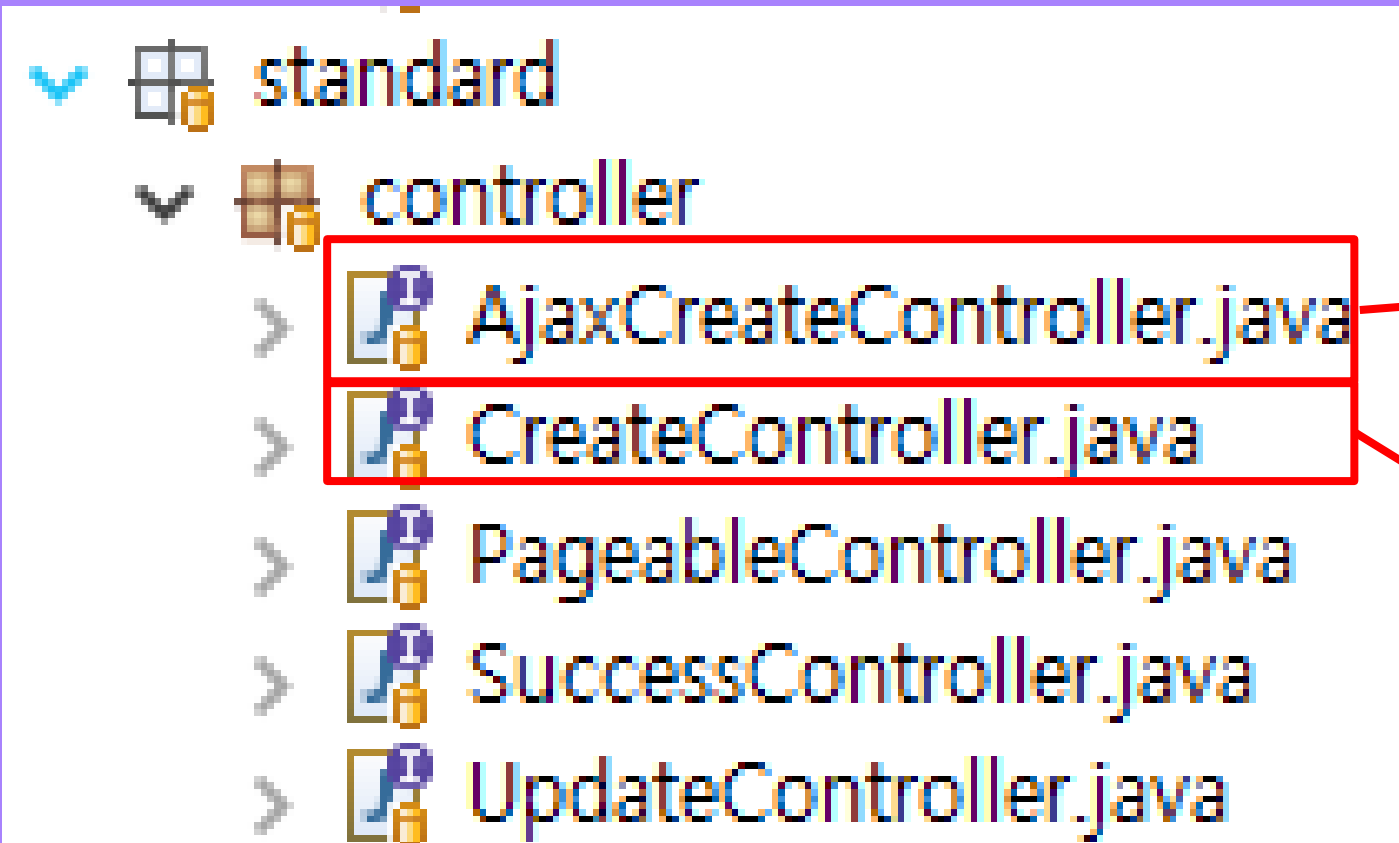
- avoidGame
  - GameModel.js
  - RecordChecker.js

- user
  - find
    - id.jsp
    - pw.jsp
    - result.jsp
    - create.jsp
    - login.jsp
    - success.jsp
    - update.jsp



## MVC MODEL

# STANDARD CONTROLLER



```
13 //Ajax로 받은 데이터 처리 관련 interface
14 public interface AjaxCreateController<T,K,V> {
15
16     @GetMapping("/create")
17     void create(Model model, HttpServletRequest request);
18
19     @PostMapping("/create")
20     ResponseEntity<T> create(@RequestBody Map<K, V> requestBody);
21
22 }
```

```
12 // 계정 생성 관련 interface
13 public interface CreateController<DTO> {
14
15     @GetMapping("/create")
16     void create(Model model, HttpServletRequest request);
17
18     @PostMapping("/create")
19     String create(@Valid DTO dto, BindingResult binding, Model model, HttpServletRequest request, RedirectAttributes attr);
20
21 }
```

Standard Controller interface를 구현  
=> 표준에 맞추어 구현 클래스를 생성함



DATA BASE





SCOTT.RECORD	
USER_NAME	VARCHAR2 (10 CHAR)
SCORE	NUMBER
ELAPSE_TIME	NUMBER
GAME_LEVEL	NUMBER

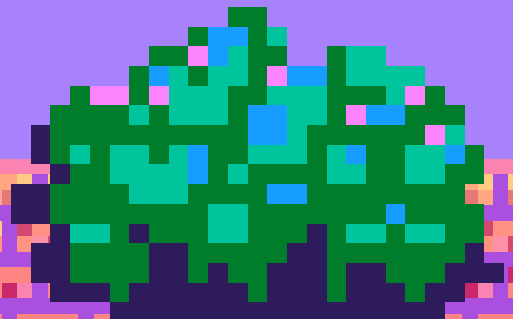
RECORD TABLE  
블록 피하기 게임의 기록들을 저장

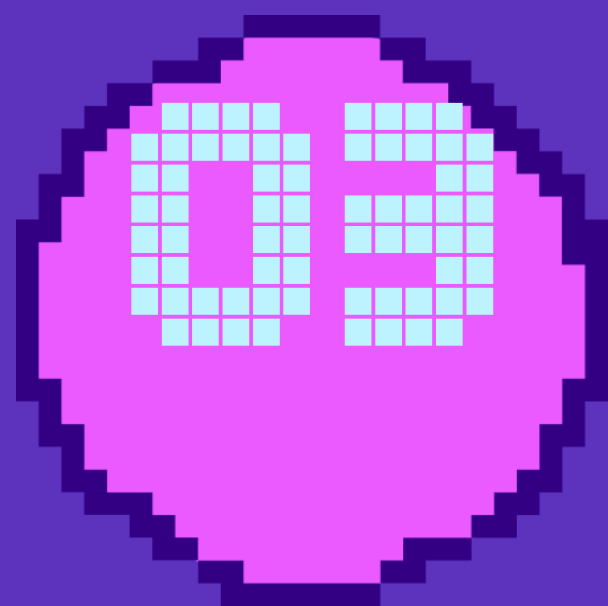
SCOTT.RECORD2	
USER_NAME	VARCHAR2 (10 CHAR)
SCORE	NUMBER
GAME_LEVEL	NUMBER

RECORD2 TABLE  
스네이크 게임의 기록들을 저장

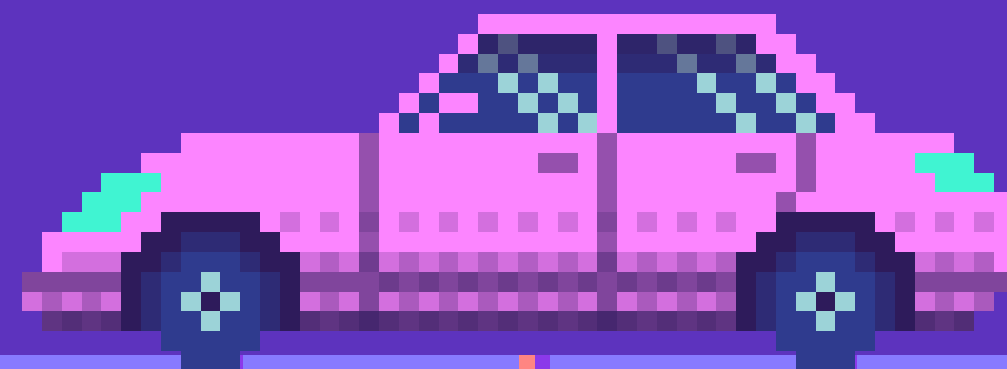
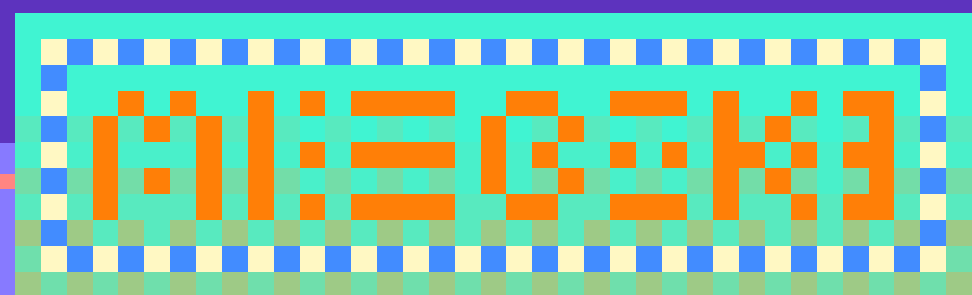
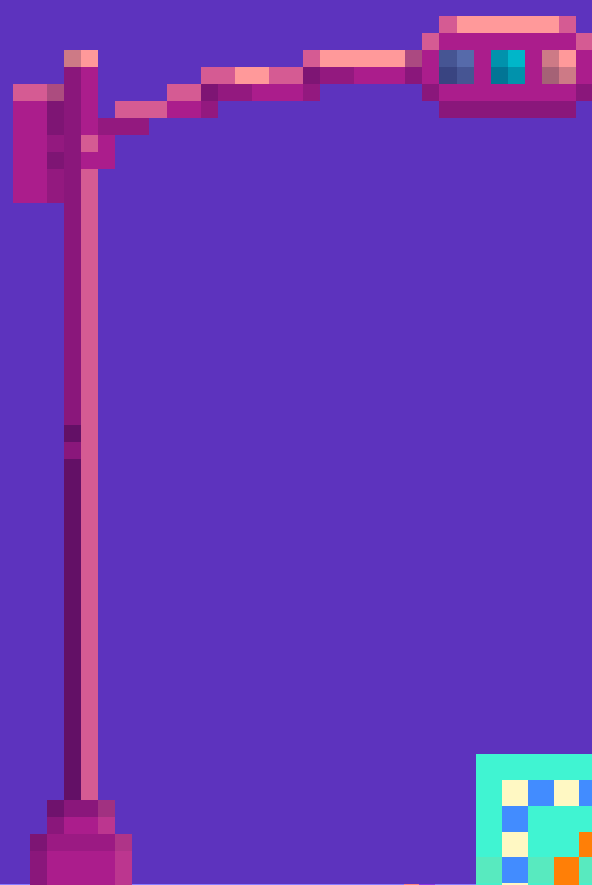
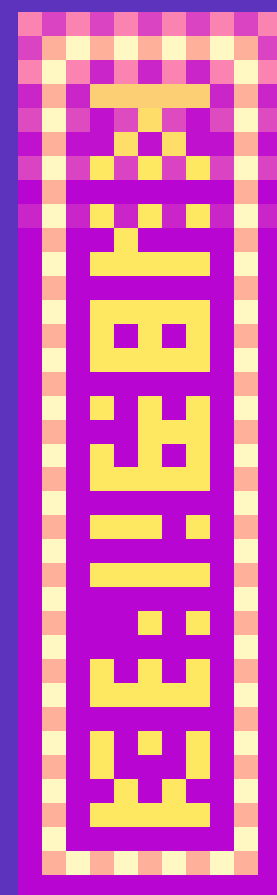
SCOTT.USERS			
P	*	ID	VARCHAR2 (20 BYTE)
	*	PASSWORD	VARCHAR2 (100 CHAR)
		ROLE	VARCHAR2 (20 CHAR)
		NAME	VARCHAR2 (20 BYTE)
		PHONE_NUMBER	CHAR (13 CHAR)
 USERS_ID_PK (ID)			
 USERS_ID_PK (ID)			

USERS TABLE  
사용자 계정과 관련된 정보들을 저장





# 핵심 코드 소개

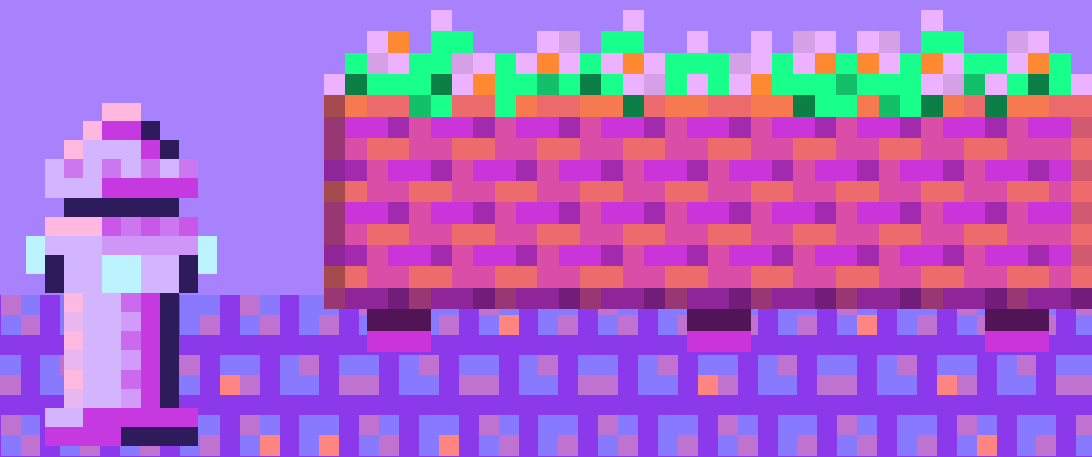


# GAME ALGORITHM

## 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, elapsedTimeId, recordChecker, setFinalScore, finalScoreTag) {`

//블록을 1칸 아래로 이동 후 바뀐 위치를 blockMap에 저장

`this.hide(object);`

이전 위치의 HTML 태그의 display 속성을 'none'으로 하여 숨김

`this.line++;`

`if(blockMap[this.line-1][this.column-1]==null){`

`blockMap[this.line-2][this.column-1]=null; // 원래 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';`

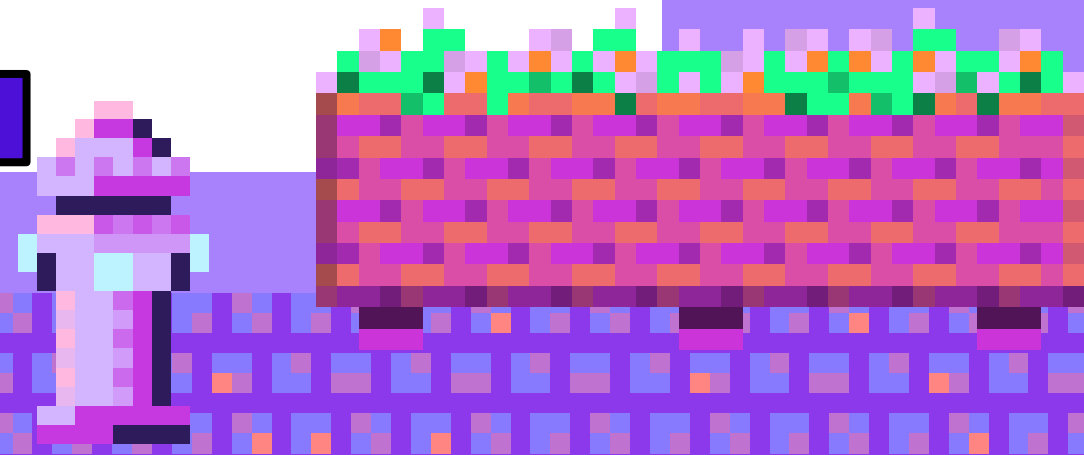
`object.style.top = height*(`

`this.show(object);`

현재 위치의 HTML 태그의 display 속성을 'inline'으로 하여 화면에 출력

위치 정보는  
20X40의 2차원 배열인  
blockMap에 저장

이동한 위치에 캐릭터가  
존재할 경우 게임 종료



# 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, elapsedTime에 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)
        elapsedTime: recordChecker.elapsedTime
    };

    const options = {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        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 elapsedTime = Integer.parseInt(requestBody.get("elapsedTime"));
    var record = Record.builder()
        .userName(userName)
        .score(score)
        .elapsedTime(elapsedTime)
        .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.getElapsedTime() < elapsedTime) {
            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, elapsed_time, game_level)
values
(#{record.userName}, #{record.score}, #{record.elapsedTime}, #{record.gameLevel})
""")
int insertRecord(@Param("record") Record record);
```

# GAME ALGORITHM

## 3. DATA TRANSFER

```
1  -- Avoid Game 랭킹 테이블 제거
2  DROP TABLE RECORD;
3
4  -- Avoid Game 랭킹 테이블 생성
5  CREATE TABLE RECORD(
6      USER_NAME  VARCHAR2(10 CHAR),
7      SCORE      NUMBER,
8      ELAPSE_TIME NUMBER,
9      GAME_LEVEL  NUMBER
10 );
11
12 -- Snake Game 랭킹 테이블 제거
13 DROP TABLE RECORD2;
14
15 -- Snake Game 랭킹 테이블 생성
16 CREATE TABLE RECORD2(
17     USER_NAME  VARCHAR2(10 CHAR),
18     SCORE      NUMBER,
19     GAME_LEVEL  NUMBER
20 );
21
22 COMMIT;
```

DB, 기록 관리 테이블

```
1  DROP TABLE USERS;
2
3  CREATE TABLE USERS (
4      ID          VARCHAR2(20) CONSTRAINT USERS_ID_PK PRIMARY KEY,
5      PASSWORD    VARCHAR2(100 CHAR) NOT NULL,
6      ROLE        VARCHAR2(20 CHAR),
7      NAME        VARCHAR2(20 CHAR),
8      PHONE_NUMBER CHAR(13 CHAR)
9  );
10
11 COMMIT;
```

DB, 유저 관리 테이블

# GAME ALGORITHM

## 4. RANKING PAGE

```
1 package com.example.standard.controller;
2
3 import org.springframework.ui.Model;
4
5 //페이지 처리 관련 interface
6 public interface PageableController {
7
8     @GetMapping("/page/{pageNum}/{pageSize}")
9     String page(HttpServletRequest request, @PathVariable("pageNum") int pageNum, @PathVariable("pageSize") int pageSize, Model model);
10 }
11
```

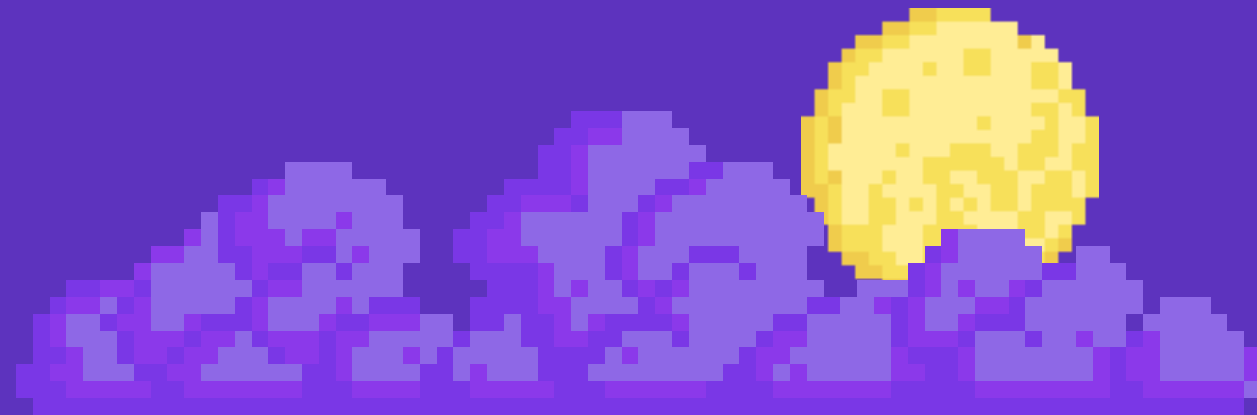
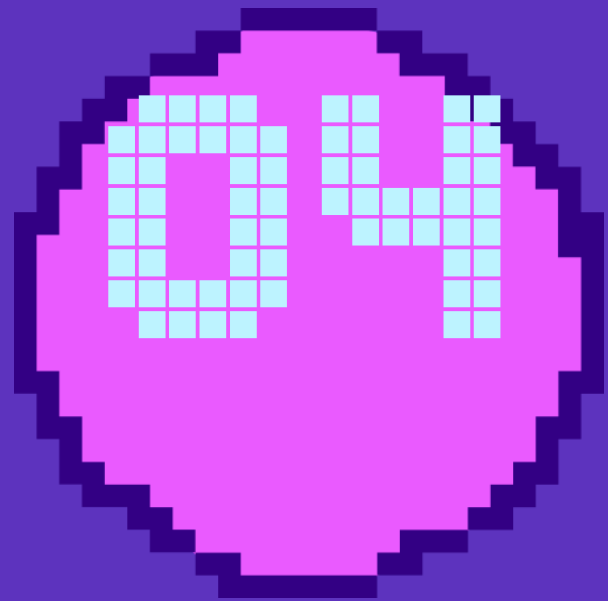
```
16 //Avoid Game 랭킹 페이지 처리 Controller
17 @Controller
18 @RequestMapping("/game/avoid/record")
19 public class RecordPageController implements PageableController{
20
21     @Autowired
22     RecordMapper mapper;
23
24     @Autowired
25     ObjectMapper json;
26
27     @Override
28     public String page(HttpServletRequest request, int pageNum, int pageSize, Model model) {
29         var level = Integer.parseInt(request.getParameter("level"));
30         model.addAttribute("level", level);
31
32         PageHelper.startPage(pageNum, pageSize);
33         var list = mapper.selectPageByGameLevel(level);
34         var paging = PageInfo.of(list, 10);
35         model.addAttribute("list", list);
36         model.addAttribute("paging", paging);
37
38         return "game/avoid/record/page";
39     }
40 }
41
```

url 정보로 페이지를 생성

Mapper로 조회한 객체들을 list에 넣음

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





# 결과물 소개



Let's test a project!

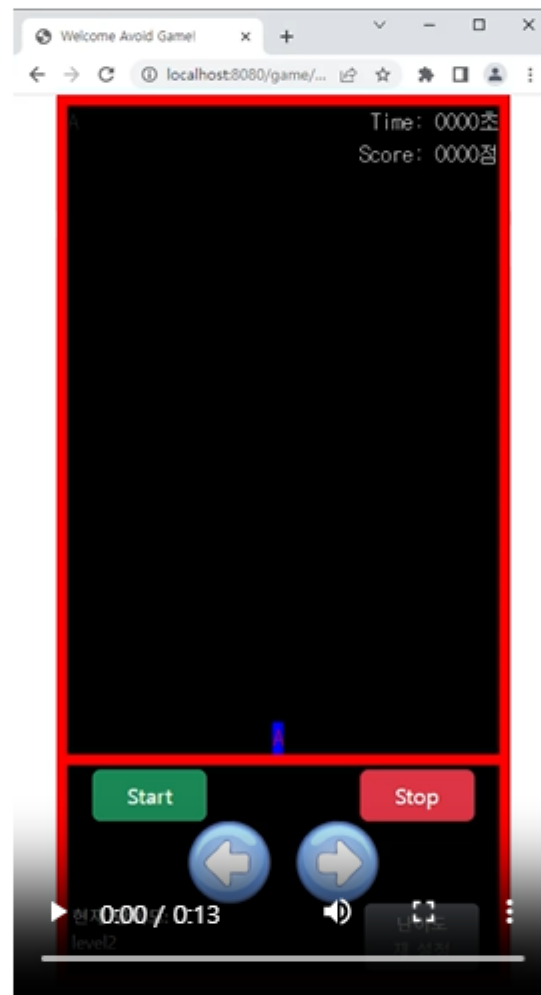


# 초기 화면

Welcome Game Land!

\*랭킹 등록을 원하시면 로그인 후 플레이 해주세요!

로그인

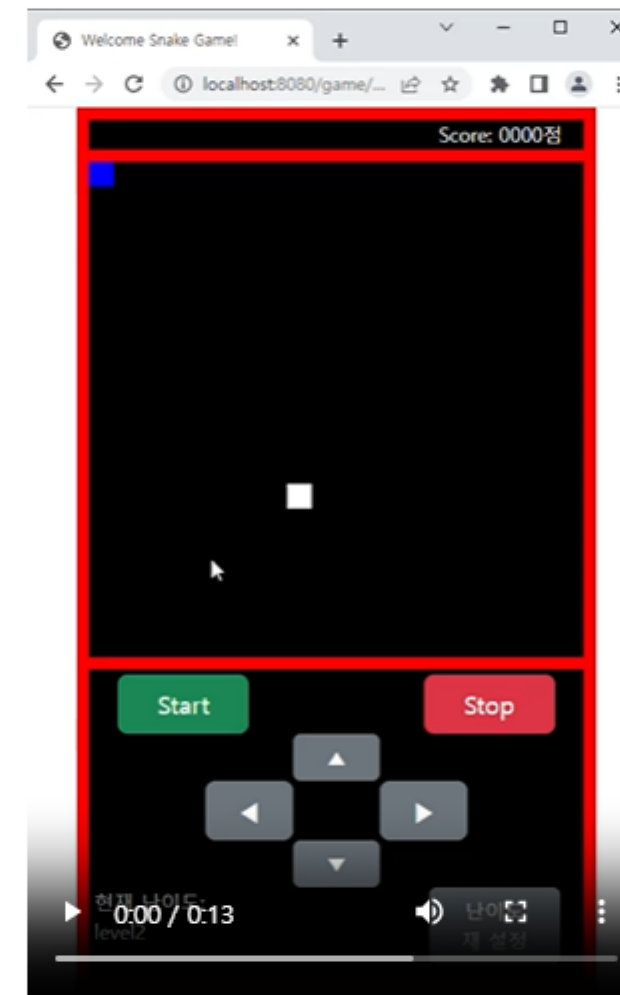


Avoid Game

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

### 신상정보

\*예시:000-0000-0000

[회원정보 수정](#)

[메인 화면으로](#)

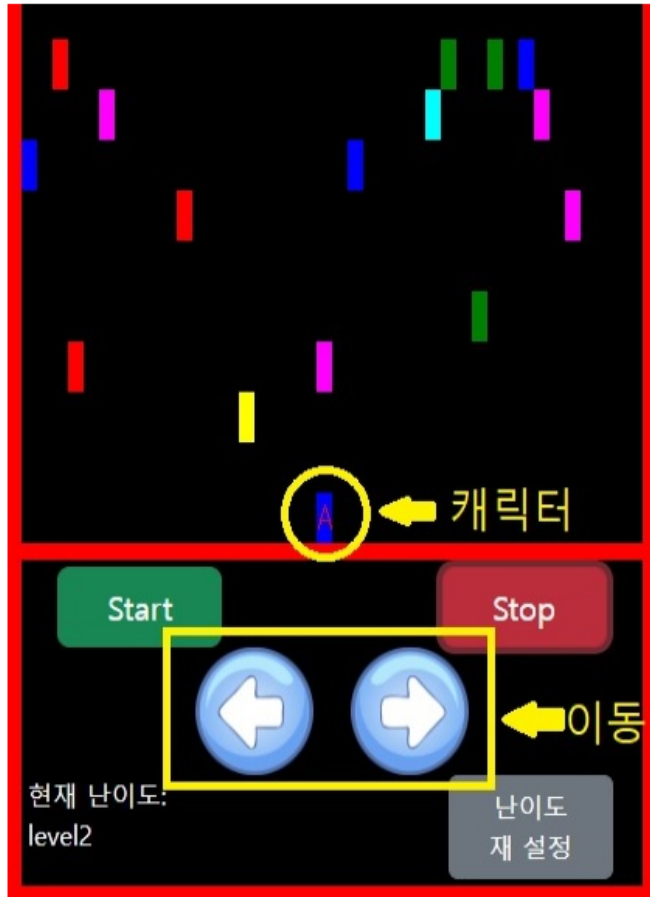


회원 정보 수정이 완료되었습니다.

[메인 화면으로](#)

# 게임 소개 및 셋팅 페이지

## Block 피하기 게임 세계에 오신 것을 환영합니다.



간단설명:

하늘에서 떨어지는 수 많은 블록들을 방향키를 이용해 피하시면 됩니다.

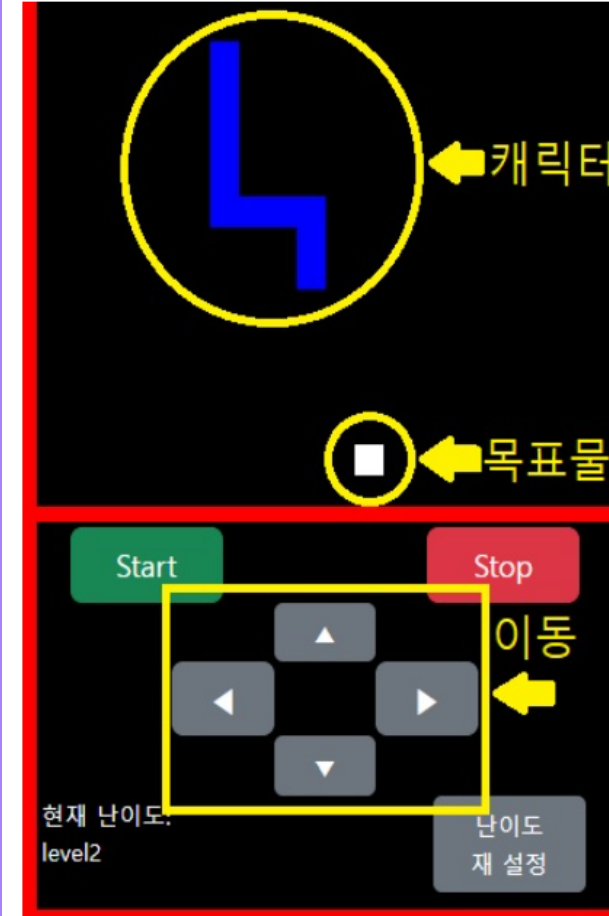
1. 가운데에 A문자가 적혀있는 블록이 캐릭터 입니다.
2. 화살표 버튼이나 키보드의 방향키로 캐릭터를 조작합니다.
3. 하늘에서 수 많은 블록들이 떨어집니다. 캐릭터와 부딪히면 게임종료.
4. 블록이 바닥까지 떨어졌을 때 점수가 1점 추가 됩니다.
5. 난이도는 3단계까지 있으며, 난이도에 따라 블록의 속도와 수가 달라집니다.
6. 1단계: 쉬움, 2단계: 보통, 3단계: 어려움
7. Start 버튼을 눌러 게임을 시작하고, Stop 버튼을 눌러 일시정지 할 수 있습니다.
8. 난이도 선택 후 플레이 해주세요!

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 집어 삼키면 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!

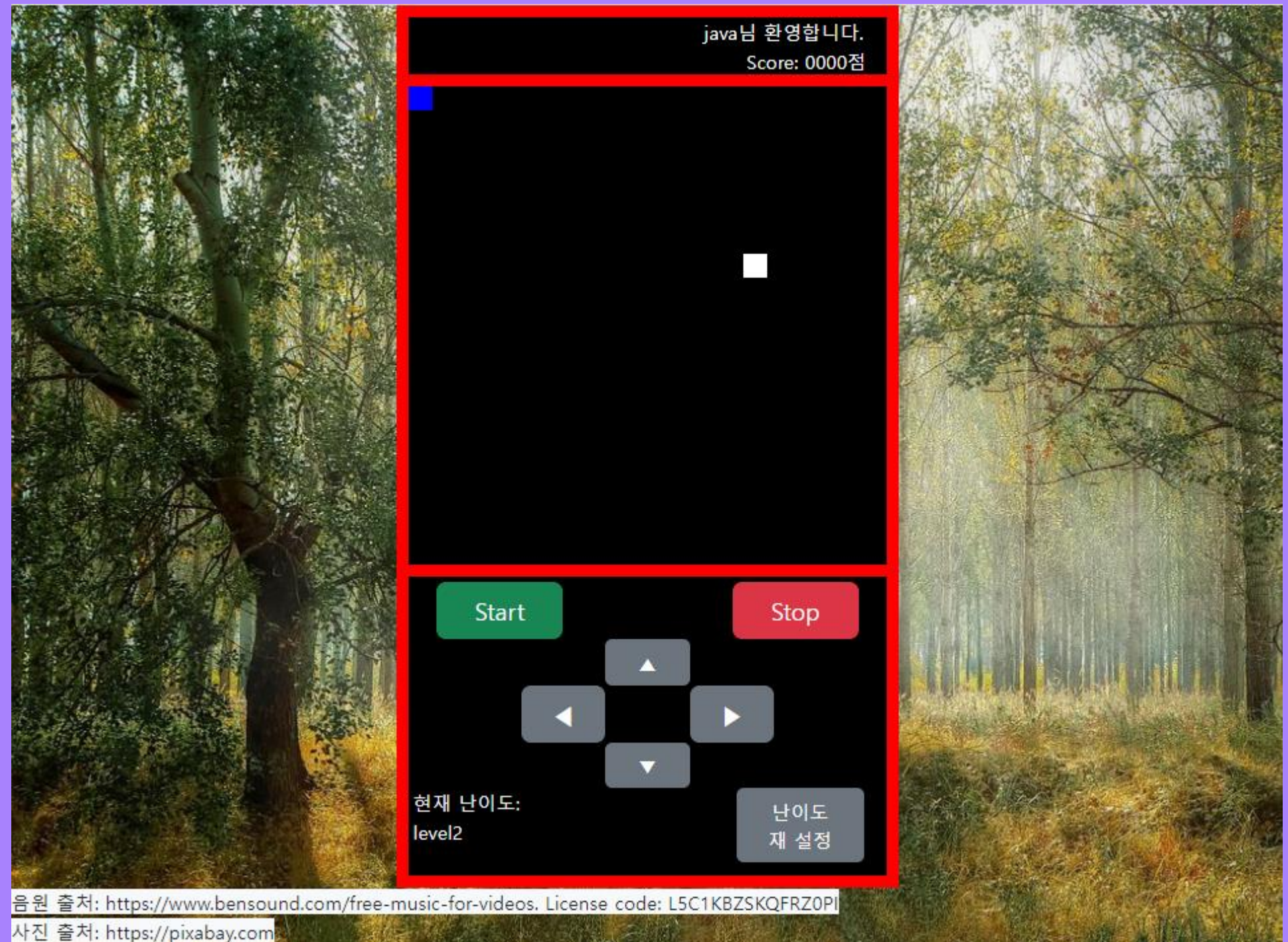
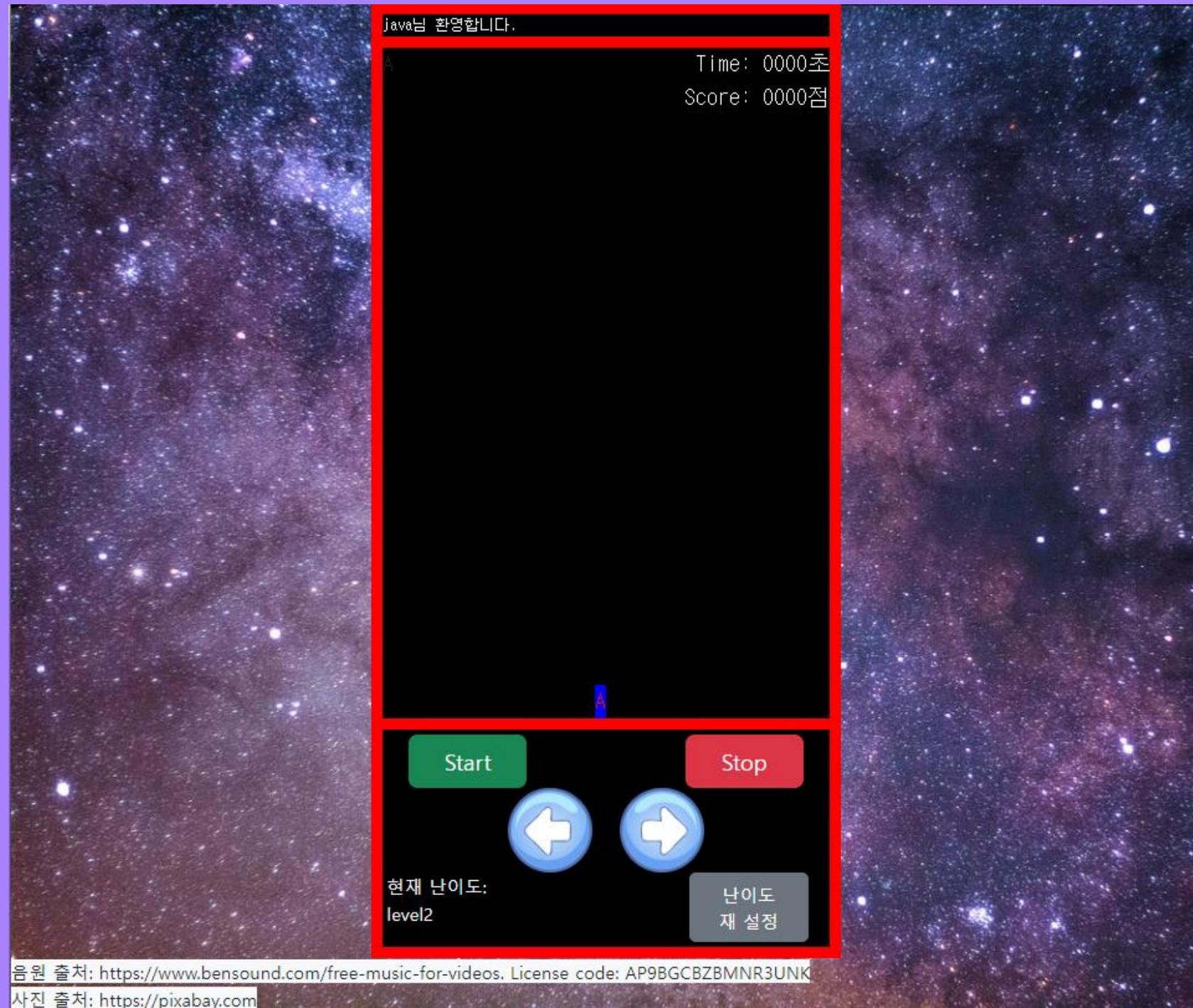
플레이 할 난이도를 선택해주세요.

Level1 Level2 Level3

메인 화면으로



# 게임 플레이 페이지





# 게임 랭킹 페이지

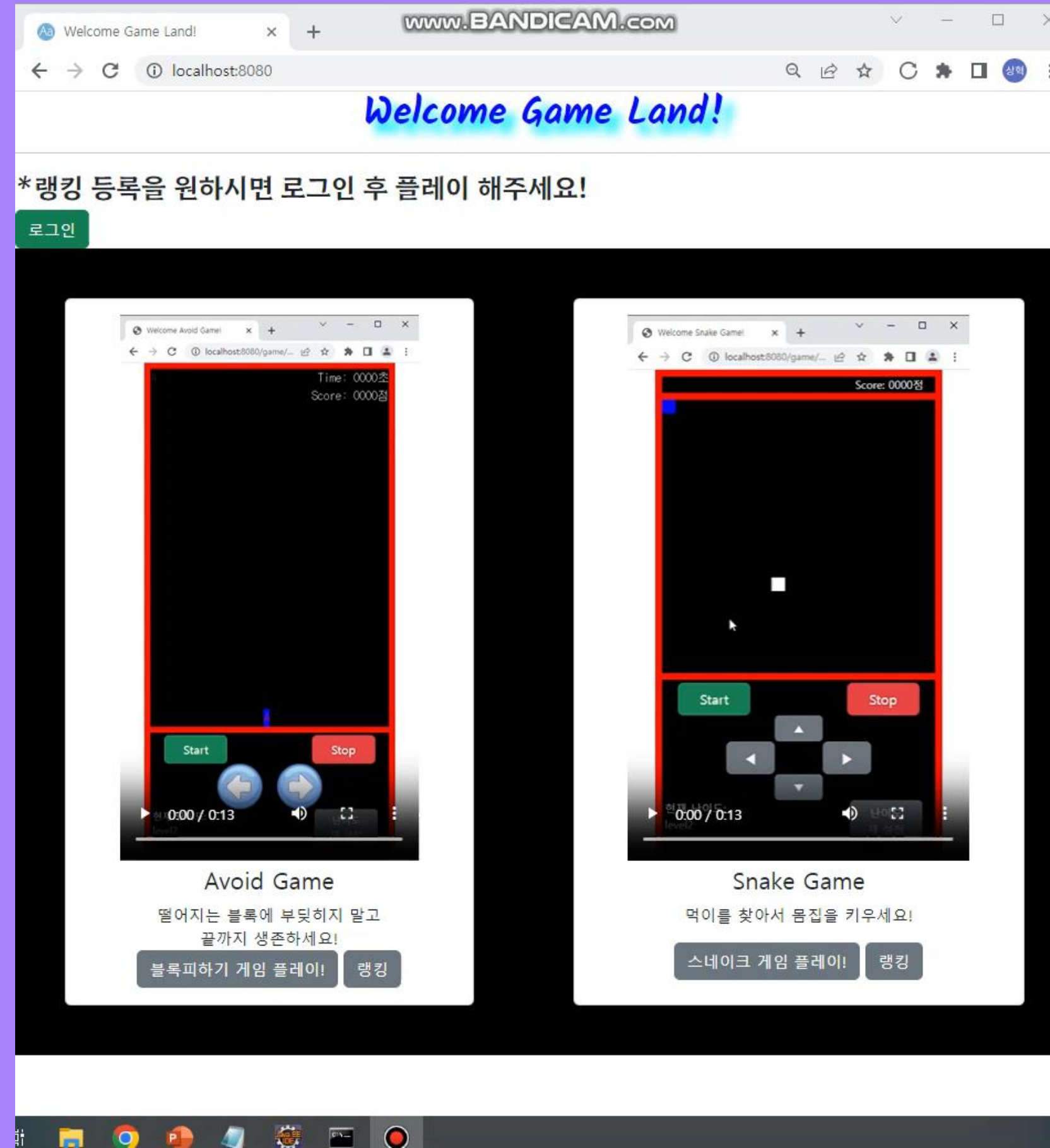
TOP100 랭킹				
*순위 산정 기준: 점수 높은 순, 점수 같을 시 생존 시간 높은 순				
Level1		Level2		Level3
등수	닉네임	점수	생존 시간	난이도
1	java	251점	13초	3
2	java	249점	12초	3
3	java	209점	10초	3
4	java	184점	7초	3
5	java	69점	3초	3
6	java	50점	2초	3
7	java	34점	1초	3
8	java	33점	1초	3
9	java	31점	1초	3
10	java	26점	1초	3
<div>&lt;&lt; Previous 1 2 Next &gt;&gt;</div>				
<div>게임으로 메인 화면으로</div>				

AVOID GAME

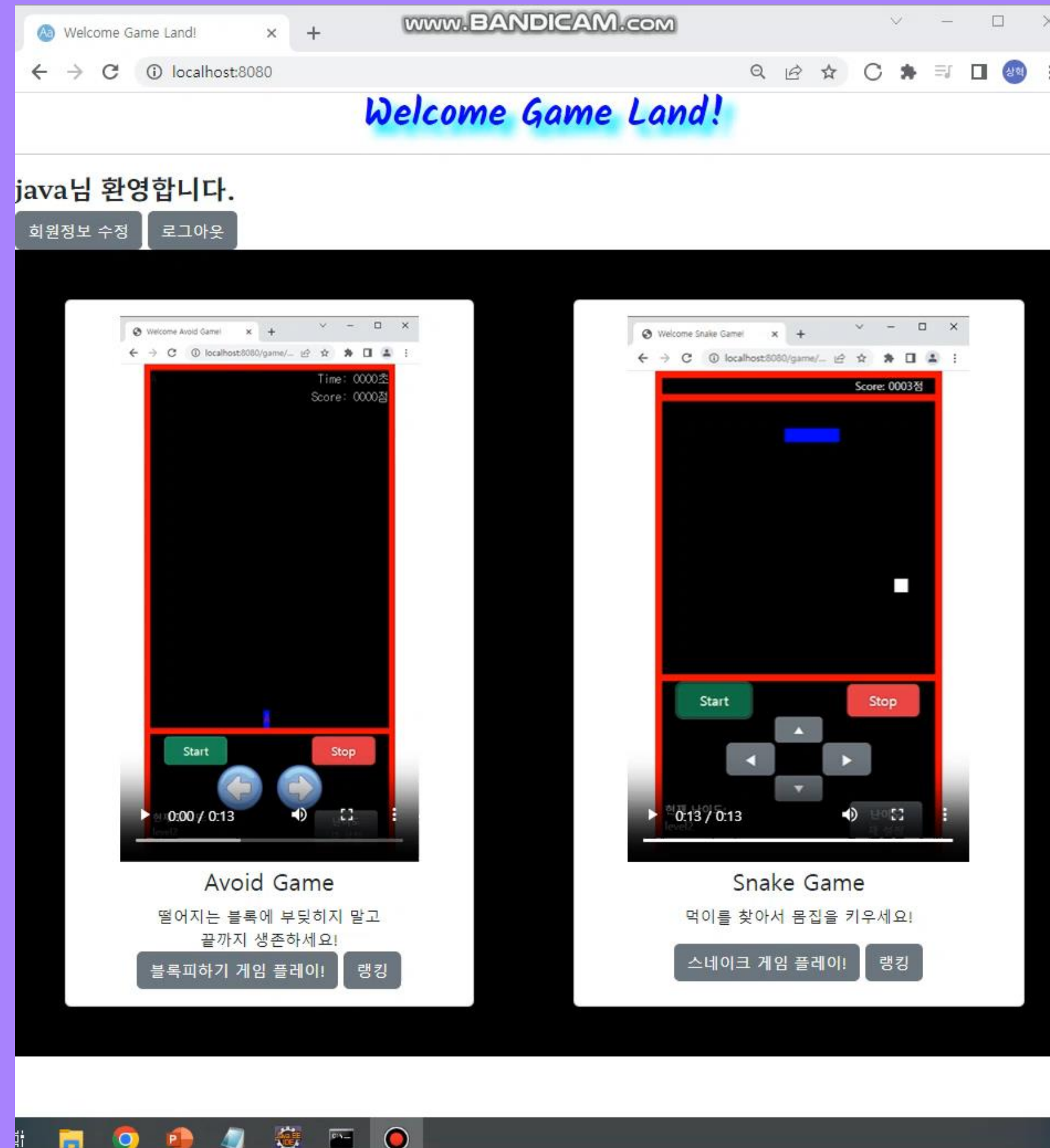
TOP100 랭킹			
Level1		Level2	Level3
등수	닉네임	점수	난이도
1	java	17점	3
2	java	7점	3
3	java	7점	3
4	java	5점	3
5	java	5점	3
6	java	4점	3
7	java	3점	3
8	java	3점	3
9	java	2점	3
10	java	2점	3
<div>&lt;&lt; Previous 1 2 Next &gt;&gt;</div>			
<div>게임으로 메인 화면으로</div>			

SNAKE GAME

# 시연 동영상 1 계정 관련 조작



# 시연 동영상 2 게임 플레이



THANK YOU

