



# OPEN SOURCE SOFTWARE PROJECT

## 중간보고서

Team HotSource

2020 년 11 월 30 일

산업시스템공학과 2016112587 김택원

산업시스템공학과 2016112548 김정률

건축공학전공 2017112547 전영인

## 목차

---

### 초기 제안 사항

#### 교수님 피드백

1. 랭킹시스템
2. 여백삭제

### 개선 사항

1. 게임 종료 후 오류 개선
2. 게임 진행 오류 개선
3. 최고 점수 갱신
4. 시간 단축에 대한 추가 점수 부여
5. 게임 시작 전 준비 시간 부여
6. 음소거 기능

### 진행 중인 사항

1. 쌓인 블록 구현
2. Frame Resizing

### 프로젝트 일정 및 역할 분담

1. 프로젝트 진행 일정
2. 역할 분담

## 초기 제안 사항



1. 게임 종료 후 오류 개선
2. 게임 진행 오류 개선
3. 최고 점수 갱신
4. 시간 단축에 대한 추가 점수 부여
5. 쌓은 블록 구현
6. 음소거 기능

## 교수님 피드백

### 1. 랭킹시스템

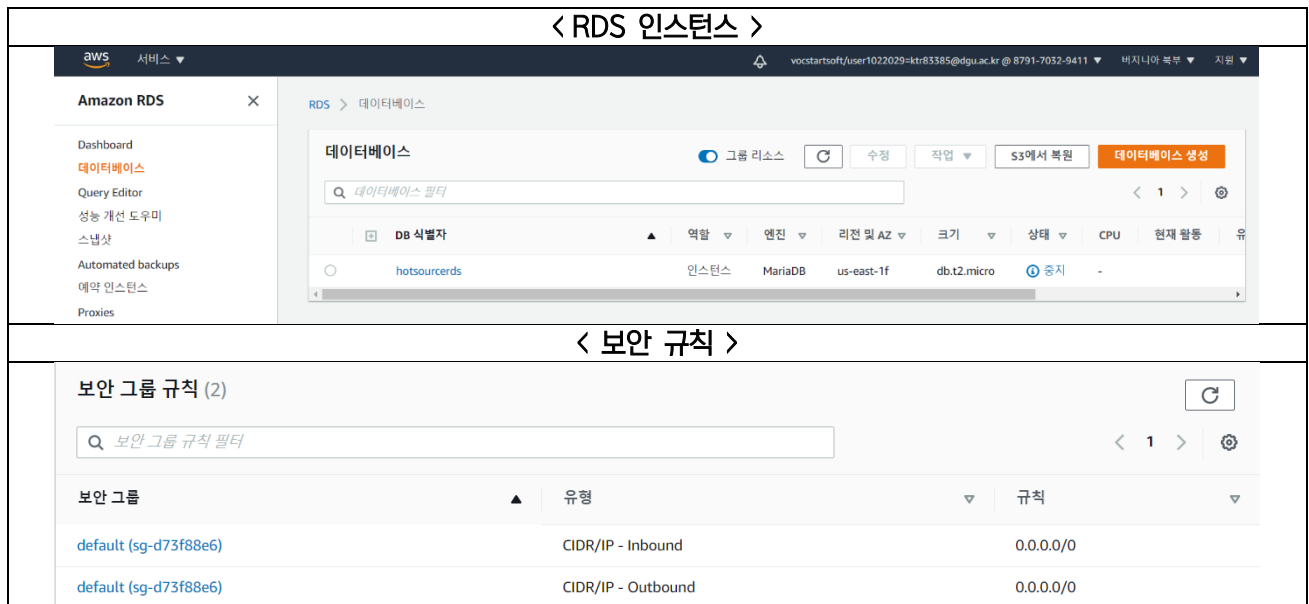
```
last_src [2020-2-OSSP-CP-HotSource-3 main]
> com.ok.ai
> com.ok.classes
> com.ok.gamedb
  > DBInsert.java
  > DBSelect.java
  > MyDB.java
> com.ok.images
> com.ok.main
> JRE System Library [jdk-14.0.2]
score.txt
```

기존에는 위 사진에 나온 것처럼 score.txt 파일을 사용하여 랭킹 현황을 갱신했다. 이렇게 텍스트 파일로 유저들의 점수를 관리하면 문제가 발생하는데, 그것은 바로 각자의 게임프로그램에서 랭킹 현황이 다르다는 점이다. 어느 유저에게는 <그림 1> 과 같은 화면을 보여주고, 다른 유저에게는 <그림 2>와 같은 화면을 보여줄 수 있다는 것이다.

| <그림 1>   | <그림 2>   |
|--|--|
|  <pre>ID : adfasdfadsf, MY SCORE : 280 1등! ID : jungreul, SCORE : 2550 2등! ID : jungreul, SCORE : 1842 3등! ID : OSSP, SCORE : 1635 4등! ID : jugnfajskf, SCORE : 1154 5등! ID : jungreul, SCORE : 1125 6등! ID : jungreulZZZZ, SCORE : 1099 7등! ID : jungreul97, SCORE : 900 8등! ID : jugreul456456, SCORE : 812 9등! ID : adsfadsf, SCORE : 375 10등! ID : jungreul_test, SCORE : 360</pre> |  <pre>ID : OSSP, MY SCORE : 1635 1등! ID : jungreul, SCORE : 2550 2등! ID : jungreul, SCORE : 1842 3등! ID : OSSP, SCORE : 1635 4등! ID : jungreul, SCORE : 1125 5등! ID : jungreulZZZZ, SCORE : 1099 6등! ID : jungreul97, SCORE : 900 7등! ID : jugreul456456, SCORE : 812 8등! ID : adsfadsf, SCORE : 375 9등! ID : jungreul_test, SCORE : 360 10등! ID : rlawjdff, SCORE : 276</pre> |

위와 같은 문제를 교수님께서 지적하시면서, 이를 해결하려면 랭킹을 관리하는 서버가 필요할 것이라고 말씀해 주셨다. 따라서 저희는 AWS 를 떠올렸고, RDS 라는 것을 통해 해결할 수 있었다.

AWS RDS 인스턴스 생성, 외부에서도 접속할 수 있도록 보안 규칙 설정(인바운드, 아웃바운드)

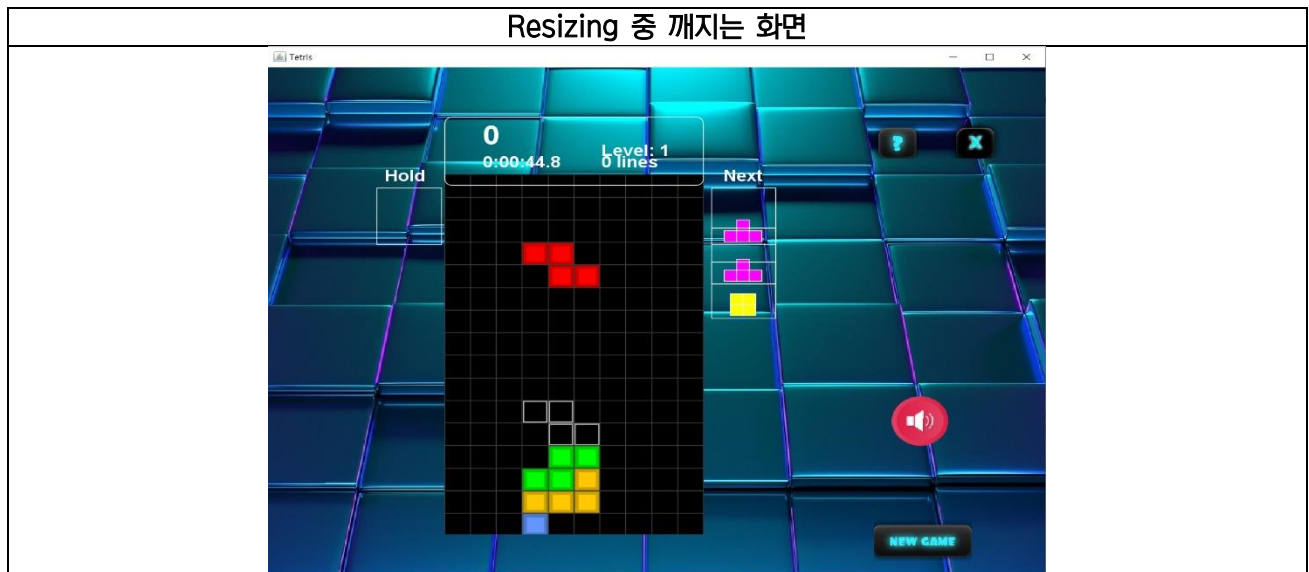


이렇게 DB 서버를 구축하고 나서, JDBC 를 사용하여 java 소스코드 내에서 DB 를 활용할 수 있도록 하여 랭킹 시스템을 관리하는 것으로 개선하였다.

## 2. 여백삭제



여백을 삭제하는 것까지는 좋았으나, 기존에 잘 되던 frame resizing 이 기존 설정한 frame 사이즈를 넘어서면 frame 안 내용물이 깨지는 오류를 발견했다. 이를 최종 발표 전까지는 수정할 예정이다.



## 개선 사항

### 1. 게임 종료 후 오류 개선

#### 문제

완전히 게임을 끝내지 않은 상태에서 메뉴 화면으로 돌아가 다시 게임을 시작하려고 했을 때 키보드와 마우스를 이용하여 해당 화면을 벗어날 수 없었다. 이를 개선하여 다시 게임에 접속하였을 때 정상적으로 게임에 참여할 수 있도록 한다.

#### 해결방법

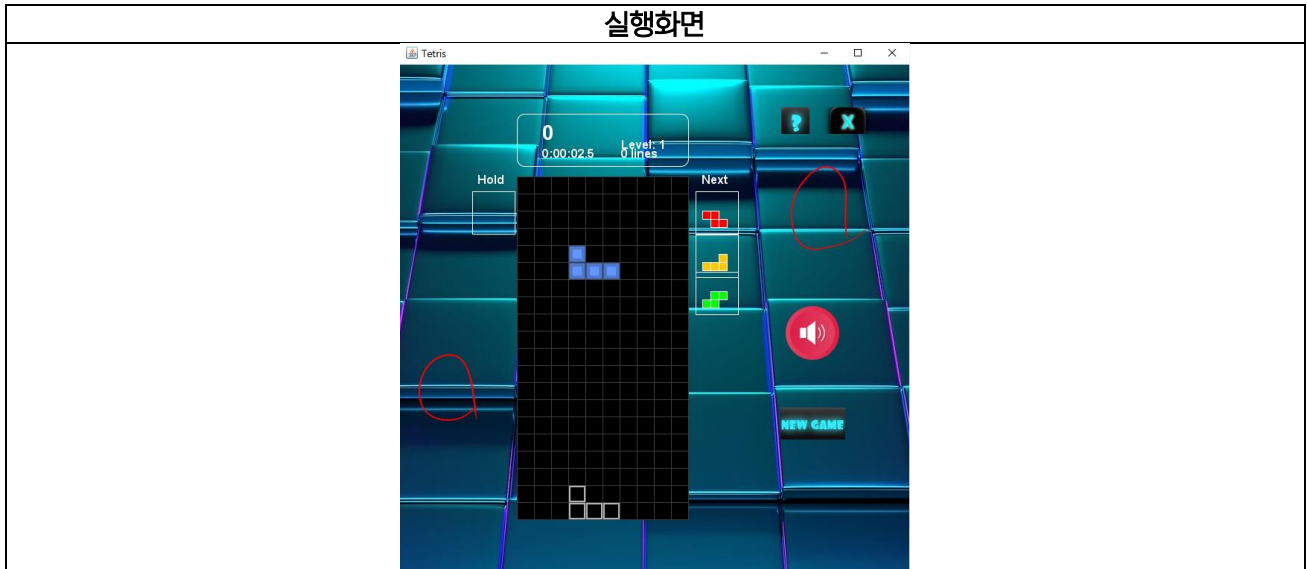
ActionListener를 각 버튼에 직접 지정해 주었다.



## 2. 게임 진행 오류 개선

### 문제

기존에 게임 진행 중 마우스의 포커싱이 밖으로 나갔다가 오거나, 게임의 배경(빨간 동그라미 부분)을 클릭하면 게임이 진행이 되지 않고 키보드가 먹통이 되는 오류가 있었다.



### 해결방법

기존에 마우스로 배경을 클릭하게 되면 게임의 포커싱이 화면으로 향하게 된다는 것을 알고 TetrisRenderer.java 에 배경에 포커싱이 맞춰진 것을 false 로 바꾸고 다른 버튼들 모두 포커싱을 주지 않았더니 해결되었다.

### 구현 코드

```
background.setSize(getMinimunSize());
background.setBorderPainted(false);
background.setContentAreaFilled(false);
background.setFocusPainted(false);
background.setFocusable(false);
background.setVisible(true);
frame.add(background); //add game play screen background image
frame.setLocationRelativeTo(null);
```

## 3. 최고 점수 갱신

### 문제

기존 프로젝트에서 score board 에 최고 점수 및 랭킹이 제대로 기록되지 않음을 확인하였다. 따라서 score board 에 취득점수에 맞는 랭킹과 점수를 기록하여 다른 플레이어들과 경쟁하면서 성취감을 느낄 수 있도록 한다.

## 해결방안

게임을 이용하는 모든 사용자의 데이터를 이용하여 공통된 랭킹 시스템을 구현하기 위해 RDS(AWS)를 사용하였다.

데이터베이스를 다루는 새로운 패키지를 생성하여 다음과 같이 구성하였다.



MyDB 에는 insert 와 select 에 사용되는 리소스들을 관리하기 용이하도록 따로 정리하였다.

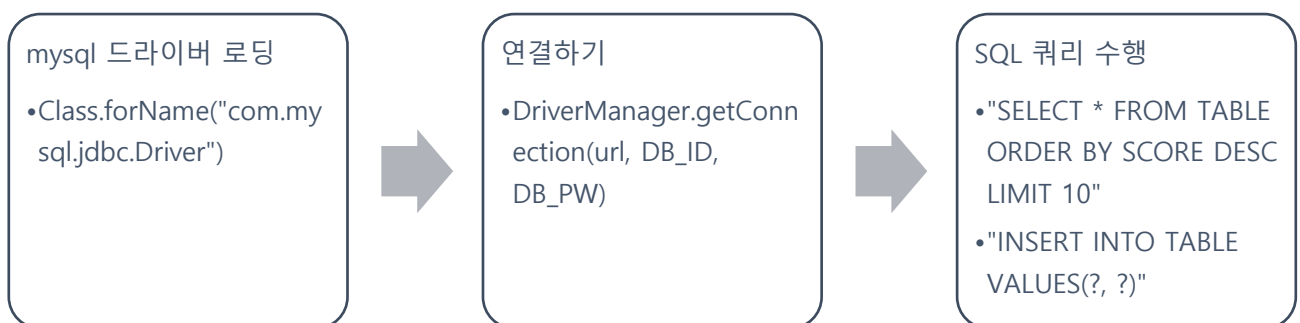
```
package com.ok.gameDB;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;

public class Game_db {
    static Connection conn = null;
    static PreparedStatement pstmt = null;
    static Statement stmt = null;
    static ResultSet rs = null;

    // RDS SETTING
    static String url = "jdbc:mysql://hoursegamedb.rds.amazonaws.com/test?serverTimezo";
    static String DB_ID = "yin9931";
    static String DB_PW = "yin9931";
    static String TABLE = "`hoursegamedb`.`gamedb`";
}
```

DBSelect 를 통해 유저의 데이터 중 상위 10 개를 가져와 프로젝트 내부의 score.txt 파일의 내용을 지우고 새로 작성하여 저장한다. DBInsert 를 통해 유저의 데이터를 RDS 서버로 전송한다.



## 4. 시간 단축에 대한 추가 점수 부여

### 문제

기존의 프로젝트에서는 두 줄 이상의 블록을 한 번에 없앴을 때 combo score 를 부여한다. 본 프로젝트에서는 기존의 combo score 에 Speed Bonus Score 점수를 추가한다. 즉 블록을 빠르게 내리는 특정 버튼을 사용하면 보너스 점수를 부여한다. 이를 통해 플레이어가 추가 점 수 획득을 위하여 빠른 속도로 게임을 즐기도록 유도하여 속도감 있는 테트리스 게임을 제공한다.

### 해결방안

| 구현 코드  | 실행화면 |
|--|------|
| <pre>30 public void firmDrop() 31 { 32     if(1 &lt;= ty &amp;&amp; ty &lt; 5) { 33         score += (FirmDropPlusScore - (ty * DropDelayMinusScore)); 34     } 35     else if(0 &gt;= ty) { 36         score += FirmDropPlusScore; 37     } 38     int oldy = ty; 39     while (pieceLegal() == LEGAL) 40         ty++; 41     ty--; 42     if (oldy != ty) 43     { 44         lastMoveRotate = false; 45         delays = 0; 46     } 47 } 48 }</pre> |      |

## 5. 게임 시작 전 준비 시간 부여

### 문제

기존의 프로젝트에서는 start 버튼을 누르는 동시에 게임이 시작된다. 플레이어가 집중력 있게 게임에 임할 수 있는 환경을 제공하기 위하여 게임을 시작하기 전에 카운트 다운 효과를 추가한다.

### 해결방안

자바의 Timer, TimerTask 객체를 이용하여 게임을 잠시 멈추고, countdown\_number 의 값을 1 씩 감소하게 한다. countdown\_number 가 [1, 3]의 범위 안에 있을 때 테트리스 구역의 중심에 countdown\_number 값을, 0 일 때 “go!”를 띄워주도록 코드를 작성하였다.



## 구현 코드

```
// Tetris.java 232 line
private int countdown_number = 3;

// Tetris.java 564 line
protected void countdown() {
    countdown_number = 3;

    // 게임 멈추기
    paused = true; // stop game
    Timer count_timer = new Timer();
    TimerTask paused_task = new TimerTask() {

        @Override
        public void run() {
            paused = false; // game start!
        }
    };
    paused_timer.schedule(paused_task, delay*(getcount()+1)); //run after 1200*4ms

    // 카운트 다운
    // 1200초 뒤에 시작하고 1200초 간격으로 실행
    Timer count_timer = new Timer();
    TimerTask count_task = new TimerTask() {
        @Override
        public void run() {
            //카운트 다운 도중 게임을 종료하는 경우를 고려하여 dead==false일때만 TimerTask를 진행
            if (countdown_number>0 && dead==false) {
                countdown_number--;
            }
            else {
                countdown_number = -1;
                count_timer.cancel(); // countdown_number이 0보다 작아지면 count_timer를 종료
            }
        }
    };
    count_timer.schedule(count_task, delay, delay);
}
```

```
// Tetris.java 1108 line
// 카운트 다운 도중 게임을 종료하는 경우를 고려하여 dead==false인지 확인
// countdown_number>0일 때는 countdown_number값을 띄우기
else if (countdown_number>0 && dead==false) {
    g.setColor(new Color(0, 0, 0, 80));
    g.fillRect(x, y, FIELD_W, FIELD_H);

    g.setFont(F_PAUSE);
    FontMetrics m = g.getFontMetrics();
    int wid = m.stringwidth("aaa");// "aaa"의 너비만큼 자리 확보

    g.setColor(new Color(0, 0, 0, 120));
    RoundRectangle2D rect = new RoundRectangle2D.Float(x + FIELD_W / 2 - wid / 2 - 15, y,
    g.fillRect(rect);
    g.setColor(Color.WHITE);
    g.draw(rect);

    g.setColor(C_NOTICE);
    drawCentered(g, countdown_number+"", x + FIELD_W / 2, y + 5 + FIELD_H / 2);
}

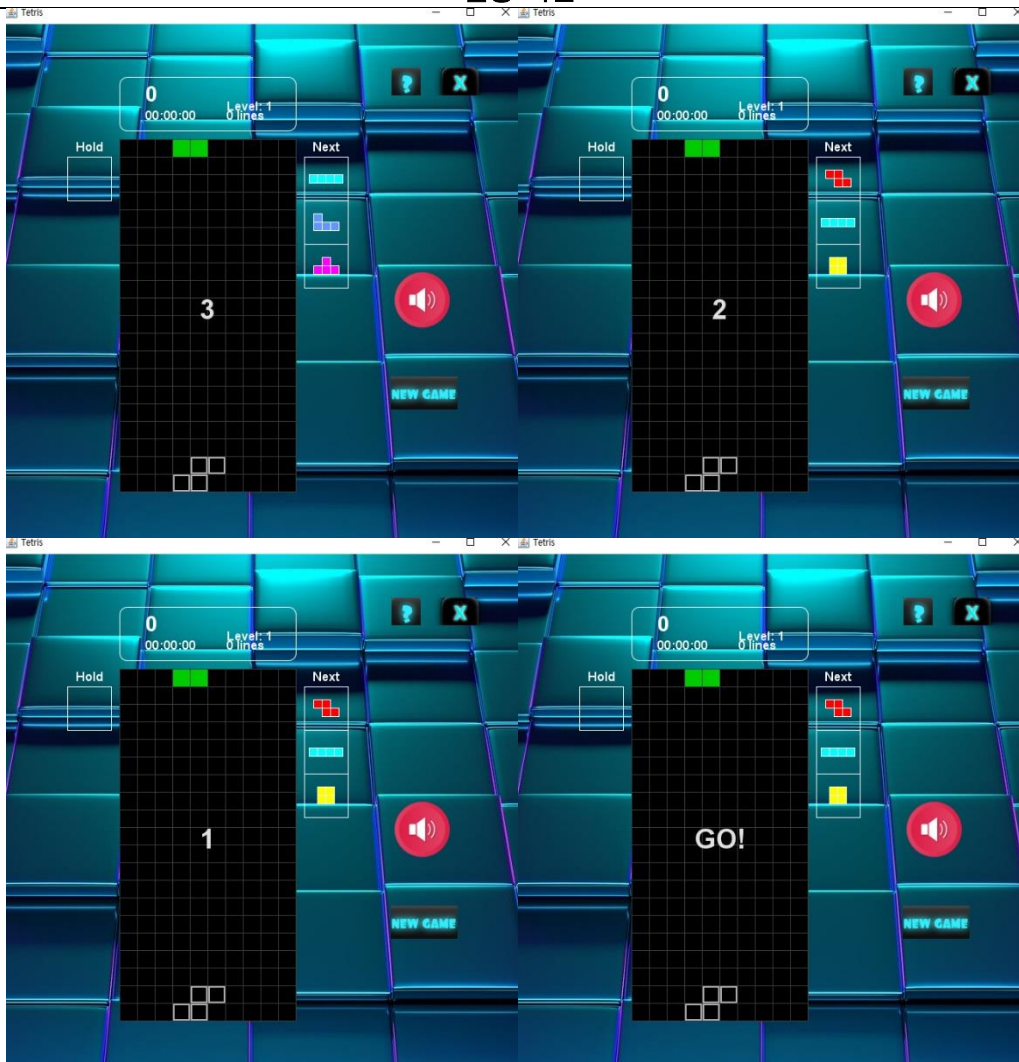
// 카운트 다운 도중 게임을 종료하는 경우를 고려하여 dead==false인지 확인
// countdown_number==0일 때는 GO!값을 띄우기
else if (countdown_number==0 && dead==false) {
    g.setColor(new Color(0, 0, 0, 80));
    g.fillRect(x, y, FIELD_W, FIELD_H);

    g.setFont(F_PAUSE);
    FontMetrics m = g.getFontMetrics();
    int wid = m.stringwidth("aaa");// "aaa"의 너비만큼 자리 확보

    g.setColor(new Color(0, 0, 0, 120));
    RoundRectangle2D rect = new RoundRectangle2D.Float(x + FIELD_W / 2 - wid / 2 - 15, y,
    g.fillRect(rect);
    g.setColor(Color.WHITE);
    g.draw(rect);

    g.setColor(C_NOTICE);
    drawCentered(g, "GO!", x + FIELD_W / 2, y + 5 + FIELD_H / 2);
}
```

## 실행 화면



## 6. 음소거 기능

### 문제

기존의 게임에서는 게임 실행 중의 배경음과 게임 종료 시 효과음을 삽입하였다. 이에 본 프로젝트에서는 음소거 기능 구현하고자 한다. 플레이어들이 적절한 효과음으로 게임에 몰입할 수 있도록 하고, 다양한 플레이 환경을 제공한다.

### 해결방안

```
public class BGM {
    private Clip clip;
    private boolean availableFile = true;
    public BGM() {
        File bgm = new File("../Sound/bgm_TheFatRat.wav");
        AudioInputStream stream;
        AudioFormat format;
        DataLine.Info info;

        try {
            stream = AudioSystem.getAudioInputStream(bgm);
            format = stream.getFormat();
            info = new DataLine.Info(Clip.class, format);
            clip = (Clip)AudioSystem.getLine(info);
            clip.open(stream);
            availableFile = true;
        } catch (Exception e) {
            System.out.println("err : " + e);
            availableFile = false;
        }
    }
    public void play() {
        if (availableFile==false) return;
        clip.setFramePosition(0);
        clip.start();
    }
    public void stop() {
        if (availableFile==false) return;
        clip.stop();
    }
}
```

우선 기존의 없었던 BGM 생성자를 만들어 주었습니다. 생성자는 노래 파일을 재생시킬 수 있도록 파일을 받아온다. 그후, BGM class 에 bgm 시작 메서드, 정지 메서드 만들어서 추가하였다.

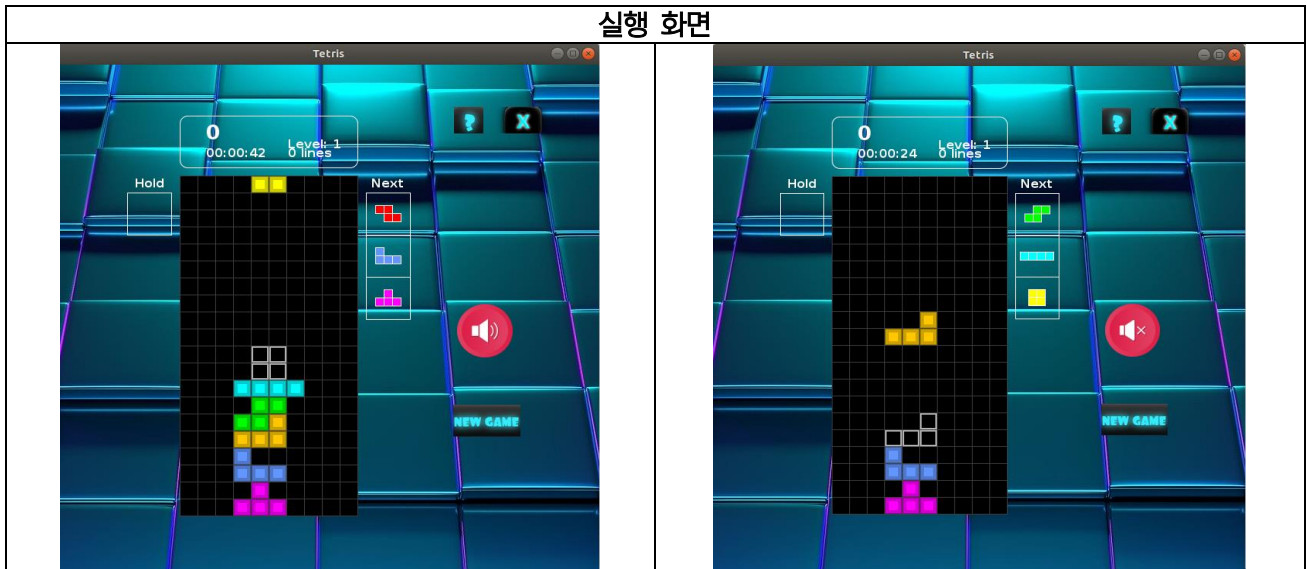
```
muteButton.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
        mute();
        muteButton.setVisible(false);
        soundButton.setVisible(true);
    }
});
soundButton.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
        mute();
        soundButton.setVisible(false);
        muteButton.setVisible(true);
    }
}
```

이후 게임 진행관련 클래스인 TetrisRenderer.java 에 음소거 버튼과 음악 재생 버튼을 추가해 주었다. 각 버튼을 클릭 시 mute() 메서드가 실행됨을 알 수 있다.

```
private void mute() {
    if(soundplay) {
        bgm_sound.stop();
        soundplay = false;
    }else {
        bgm_sound.play();
        soundplay = true;
    }
}
```

음소거 메서드 입니다. 버튼 클릭 시 음소거 중이면 음악을 재생 시키고 음악이 재생 중 이면 음소거 모드로 바꾼다.

실행 화면



## 진행 중인 사항

### 1. 쌓인 블록 구현

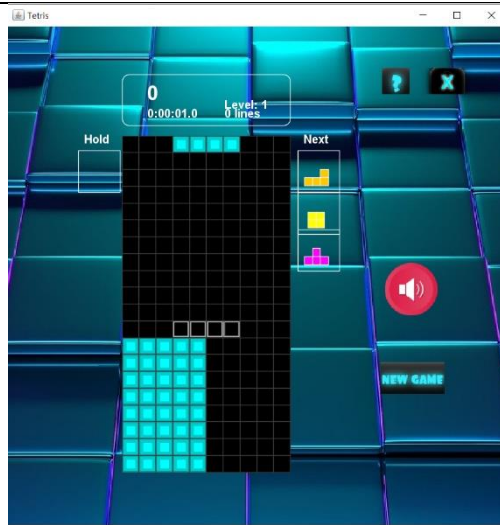
#### 문제

기존의 프로젝트에서는 컨테이너가 비어 있는 상태에서 게임을 시작한다. 다양한 게임방법을 제공하여 플레이어의 흥미를 더하기 위해 컨테이너에 블록들이 쌓여 있는 상태에서 게임을 시작할 수 있게 한다.

#### 진행사항

블록을 쌓는 데까지는 성공했지만, 이를 유저가 블록이 있을 지 없을 지를 선택할 수 있도록 하는 것은 아직 진행 중이다. 또한 쌓여져 있는 모양도 다양하게 구현할 예정이다.

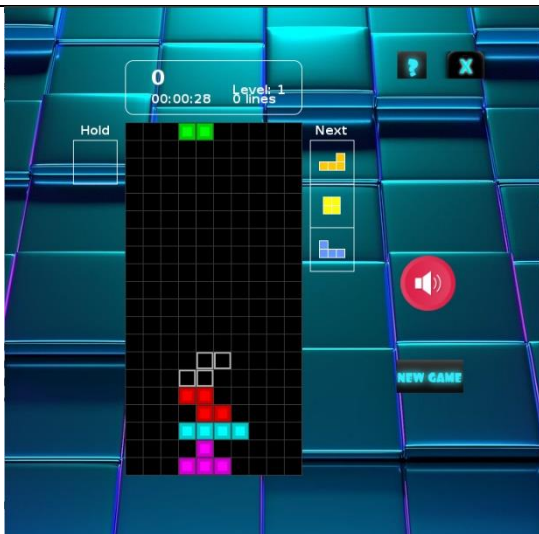
## 실행 화면



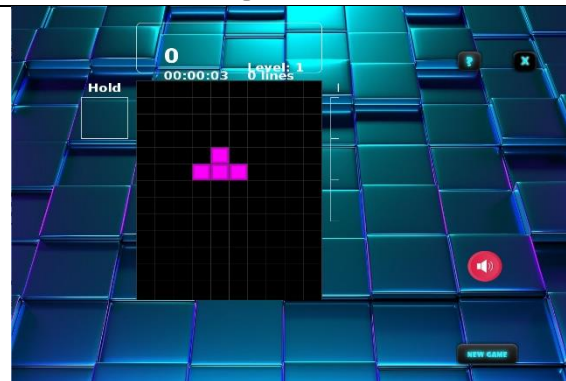
## 2. Frame Resizing

문제

### 초기 게임 화면



### 화면 창 크기 조절시



진행 사항

- 현재: 전체 창의 크기는 조절이 가능
- 진행 예정: 게임화면은 일정 크기 이상 커지지 않는 오류 개선할 예정( 플레이 화면 여백 삭제)

# 프로젝트 일정 및 역할 분담

## 1. 프로젝트 진행 일정

| ACTIVITY               | PLAN START | PLAN DURATION | ACTUAL START | ACTUAL DURATION | PERCENT COMPLETE | PERIODS |    |    |    |    |    |    |             | PARTICIPANTS |
|------------------------|------------|---------------|--------------|-----------------|------------------|---------|----|----|----|----|----|----|-------------|--------------|
|                        |            |               |              |                 |                  | 9       | 10 | 11 | 12 | 13 | 14 | 15 |             |              |
| 프로젝트 주제 선정 및 제안서 작성    | 9          | 1             | 9            | 1               | 100%             |         |    |    |    |    |    |    | 김택원,김정률,전영인 |              |
| 게임 오류 사항 수정            | 10         | 2             | 10           | 2               | 100%             |         |    |    |    |    |    |    | 김택원,김정률,전영인 |              |
| 블록 모드 구현               | 12         | 1             | 13           | 2               | 50%              |         |    |    |    |    |    |    | 김택원         |              |
| 시작 전 카운트 구현            | 12         | 1             | 12           | 1               | 100%             |         |    |    |    |    |    |    | 전영인         |              |
| 기록 갱신                  | 12         | 1             | 12           | 1               | 100%             |         |    |    |    |    |    |    | 김정률         |              |
| 시간 단축에 따른 추가 점수 부여     | 13         | 1             | 13           | 1               | 100%             |         |    |    |    |    |    |    | 김택원,전영인     |              |
| 배경 음악                  | 13         | 1             | 13           | 1               | 100%             |         |    |    |    |    |    |    | 김정률         |              |
| 오류 사항 수정 및 해결 못한 기능 추가 | 14         | 1             | 14           | 1               | 100%             |         |    |    |    |    |    |    | 김택원,김정률,전영인 |              |
| 검토 및 피드백 / 디버깅         | 15         | 1             | 15           | 1               | 100%             |         |    |    |    |    |    |    | 김택원,김정률,전영인 |              |

## 2. 역할 분담

김택원(팀장)

| 프로젝트 관리   | 기능 구현   |
|---|---|
| <ul style="list-style-type: none"> <li>● GitHub ReadMe 관리</li> <li>● 진행상황 발표</li> </ul> | <ul style="list-style-type: none"> <li>● AWS DB 구축</li> <li>● 게임 중 오류 사항 수정</li> <li>● 보너스 점수 구현</li> <li>● 쌓인 블록 모드</li> <li>● Frame Resizing</li> </ul> |

김정률

| 프로젝트 관리   | 기능 구현   |
|---|---|
| <ul style="list-style-type: none"> <li>● 보고서 작성 및 관리</li> <li>● 프로젝트 라이선스 관리</li> </ul> | <ul style="list-style-type: none"> <li>● 음소거 기능 구현</li> <li>● 게임 중 오류 사항 수정</li> <li>● 사용자 편의를 위한 주석 및 불필요 코드 정리</li> <li>● exe 실행 파일 제작</li> <li>● Frame Resizing</li> </ul> |

전영인

| 프로젝트 관리  | 기능 구현  |
|--|--|
| <ul style="list-style-type: none"> <li>● GitHub document 관리</li> <li>● PPT 제작</li> </ul> | <ul style="list-style-type: none"> <li>● 시작 시 카운트 다운 구현</li> <li>● 게임 중 오류 사항 수정</li> <li>● JDBC 활용한 랭킹 시스템 구축</li> <li>● 배경화면 UX/UI 개선</li> <li>● 불필요한 코드 정리</li> </ul> |