



OPEN SOURCE SOFTWARE PROJECT 최종보고서

Team HotSource

2020 년 12 월 9 일

산업시스템공학과 2016112587 김택원
산업시스템공학과 2016112548 김정률
건축공학전공 2017112547 전영인

목차

프로젝트 개요

프로젝트 주요 목표

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
7. 쌓인 블록 구현
8. Frame Resizing

프로젝트 일정 및 역할 분담

1. 프로젝트 진행 일정
2. 역할 분담

프로젝트 개요

기존 프로젝트는 주제가 테트리스로, 2019 년 2 학기 OSSP 에서 OS(Open Source ISE)에서 진행했다.

프로젝트의 원저자는 PSNB92 이고 MIT 라이선스에 기반하여 2017 년 11 월 프로젝트를 생성하였다.

2 차 저자는 PAIS Team 이다. OS 팀은 MIT 라이선스에 기반하여 메인스크린의 UI 를 변경하였고, 기존에 존재했던 Tetris with AI 모드 저작권자에게 일정 금액을 지불해야 하고, OS 이전 팀이 소스를 구해 왔던 저작권자 홈페이지가 닫혀 있어 다른 사람에 의해 배포 및 변경을 허용하지 않아 AI 모드를 삭제했다.

또한 응용프로그램 실행창을 사용자가 마우스를 이용하여 늘리거나 줄일 때 이에 맞게 해상도가 변경되는 Frame Resizing 기능과 테트리스 게임이 종료되었을 때 아이디를 입력하면 아이디와 점수를 기록하여 순위를 매기는 랭킹 시스템을 메인스크린에 랭킹 버튼이 나타나도록 하였다. 마지막으로 BGM -The Fat Rat - Prelude 을 도입했다.

프로젝트 주요 목표

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
 7. 배경화면 UX/UI 개선
- + FeedBack 반영
8. 랭킹시스템 개선 (AWS RDS 활용)
 9. Frame Resizing & 여백 삭제

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

ActionListener 를 각 버튼에 직접 지정해 주었다.

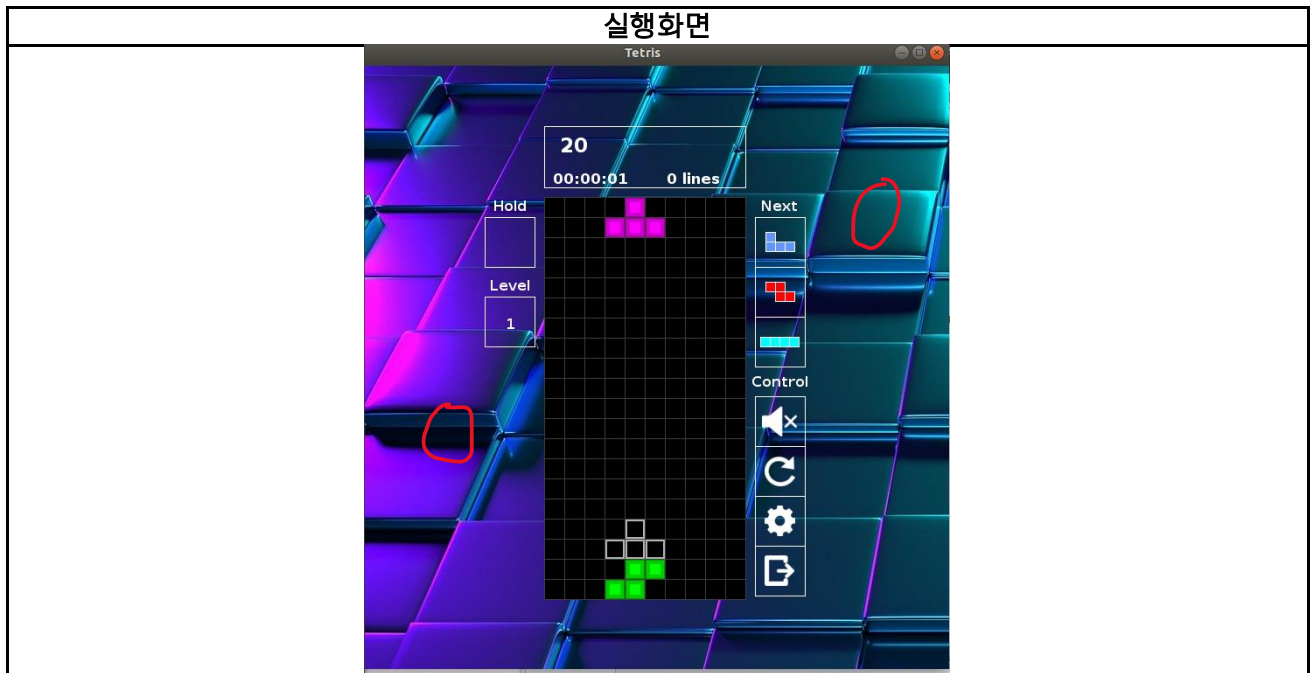
구현 코드

```
// TetrisRenderer.java 114 line
// Add click actionEvent
keyButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        launchKeyDialog();
    }
});
newButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        launchNewGameDialog();
    }
});
homeButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        game.die();
        frame.dispose();
        main = new TMain();
    }
});
```

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

```
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. 따라서 이를 개선하여 score board 에 취득점수에 맞는 랭킹과 점수를 기록하여 다른 플레이어들과 경쟁하면서 성취감을 느낄 수 있도록 한다.

그림 1

ID : adfsadfadsf, 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

그림 2

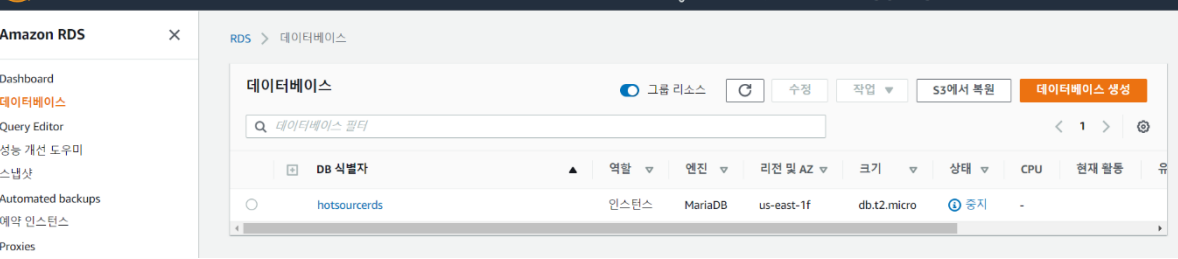
ID : OSSP, MY SCORE : 1635

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

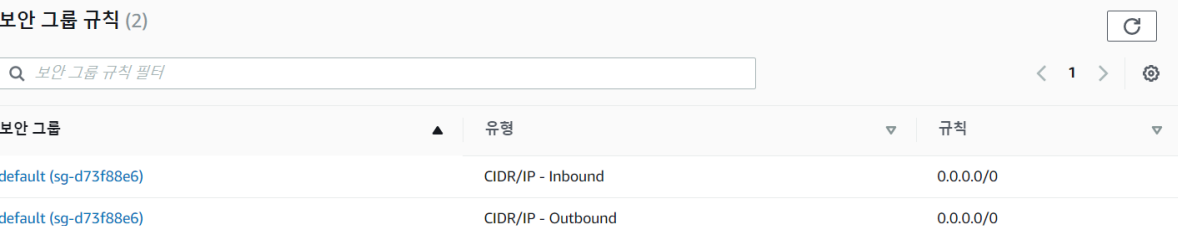
해결방안

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

RDS 인스턴스

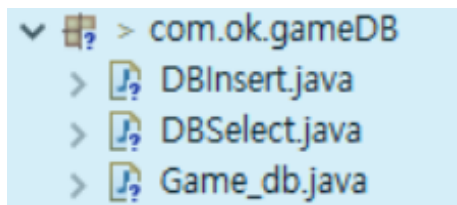


보안 규칙



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

이렇게 DB 서버를 구축하고 나서, 데이터베이스를 다루는 새로운 패키지를 생성하여 다음과 같이 구성하였다.



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

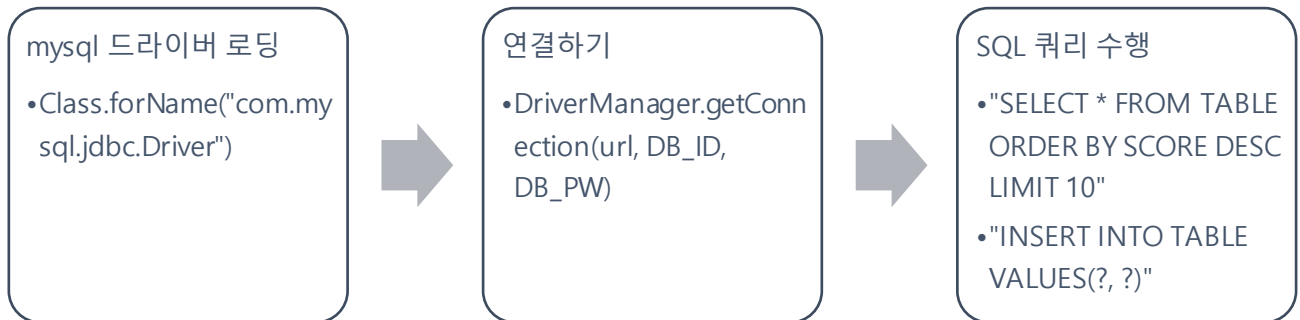
```
package com.ok.gameDB;

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://horsourcegamedb.rds.amazonaws.com/test?serverTimezo";
    static String DB_ID = "yin9931";
    static String DB_PW = "yin9931";
    static String TABLE = "`horsource_gamedb`.`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.

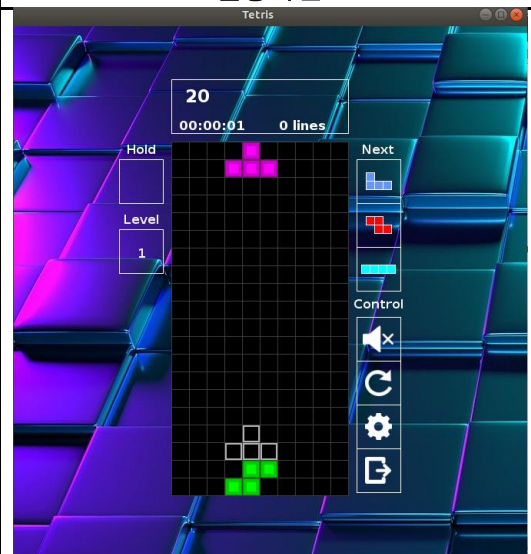
해결방안

구현 코드

```

30 public void firmDrop()
31 {
32     if(1 <= ty && ty < 5) {
33         score += (FirmDropPlusScore - (ty * DropDelayMinusScore));
34     }
35     else if(0 >= 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
  
```

실행화면



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_number의 값을 1씩 감소하게 한다. countdown_number가 [1, 3]의 범위 안에 있을 때 테트리스 구역의 중심에 countdown_number 값을, 0일 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 paused_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 - 15, wid + 30, 30);
    g.fill(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);
}

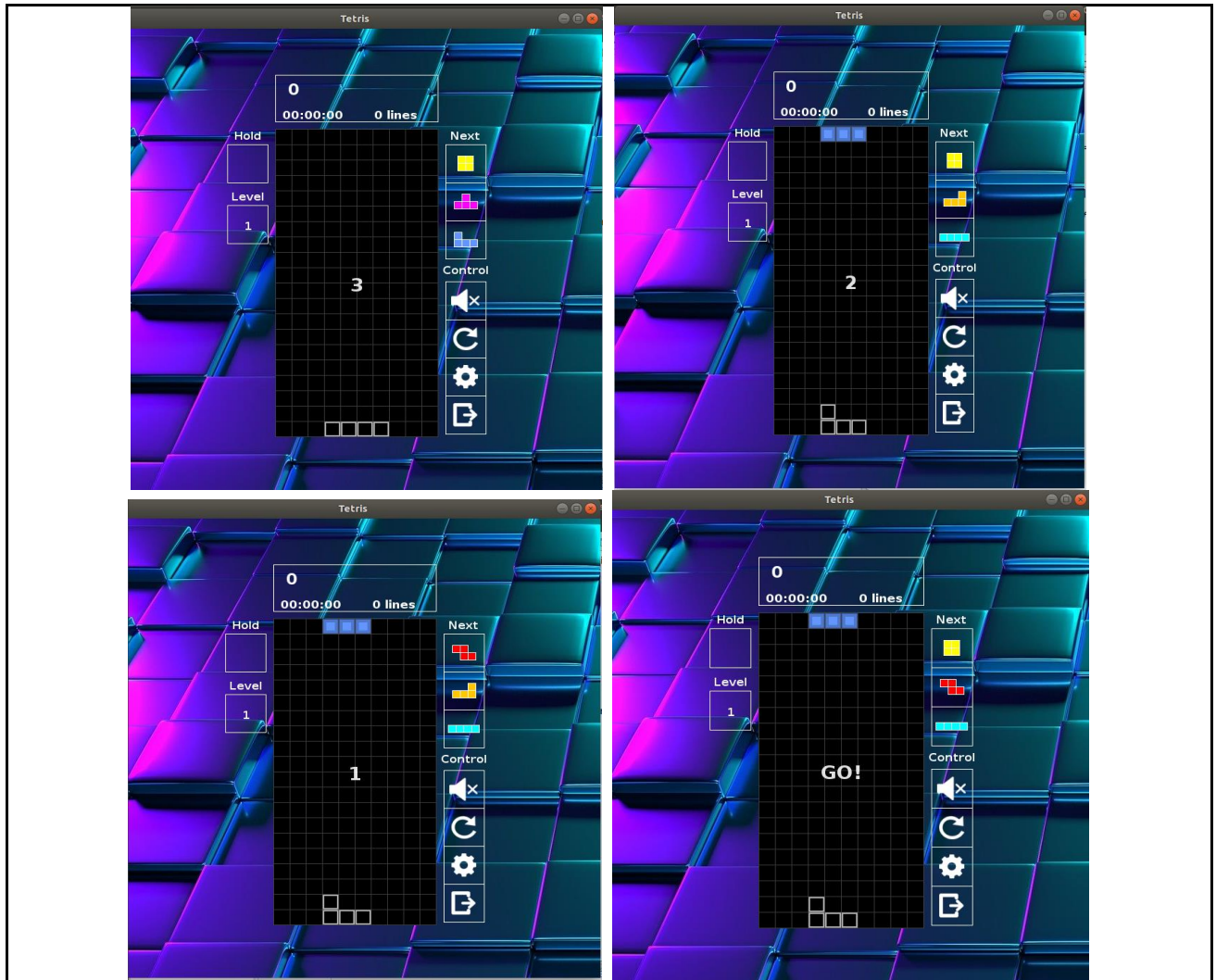
// 카운트 다운 도중 게임을 종료하는 경우를 고려하여 dead==false인지 확인
// countdown_number==0일 때는 GO!값을 띄우기
else if (countdown_number==0 && dead==false) {
    g.setColor(new Color(0, 0, 0, 80));
    g.fillRect(X, Y, FIELD_W, FIELD_H);

    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 - 15, wid + 30, 30);
    g.fill(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();
    }
}
```

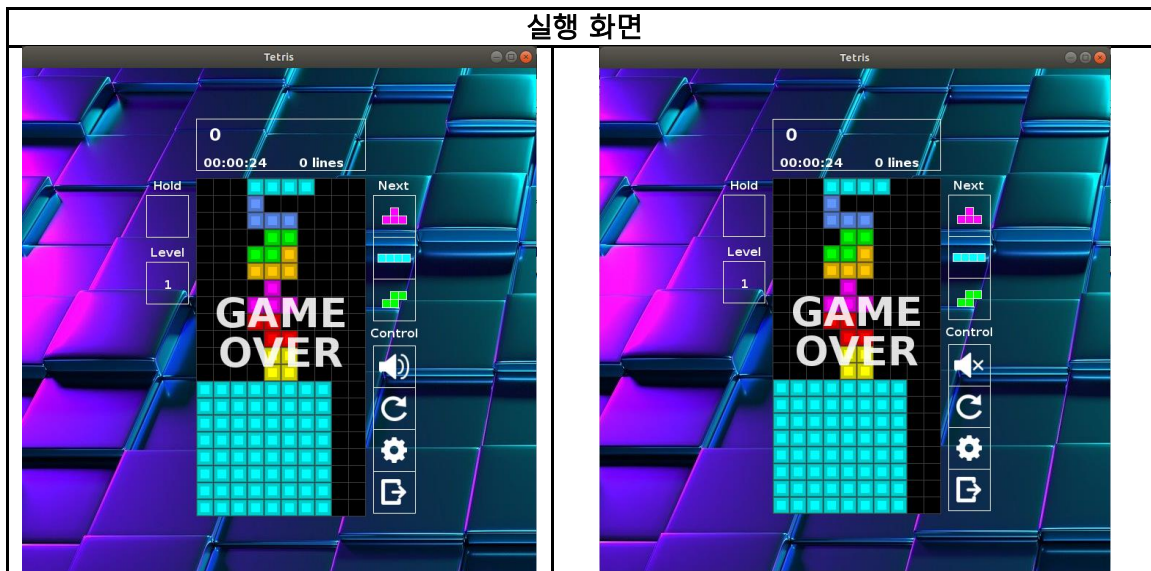
우선 기존의 없었던 BGM 생성자를 만들어 주었습니다. 생성자는 노래 파일을 재생시킬 수 있도록 파일을 받아온다. 그후, BGM class 에 bgm 시작 메서드, 정지 메서드 만들어서 추가하였다.

```
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.

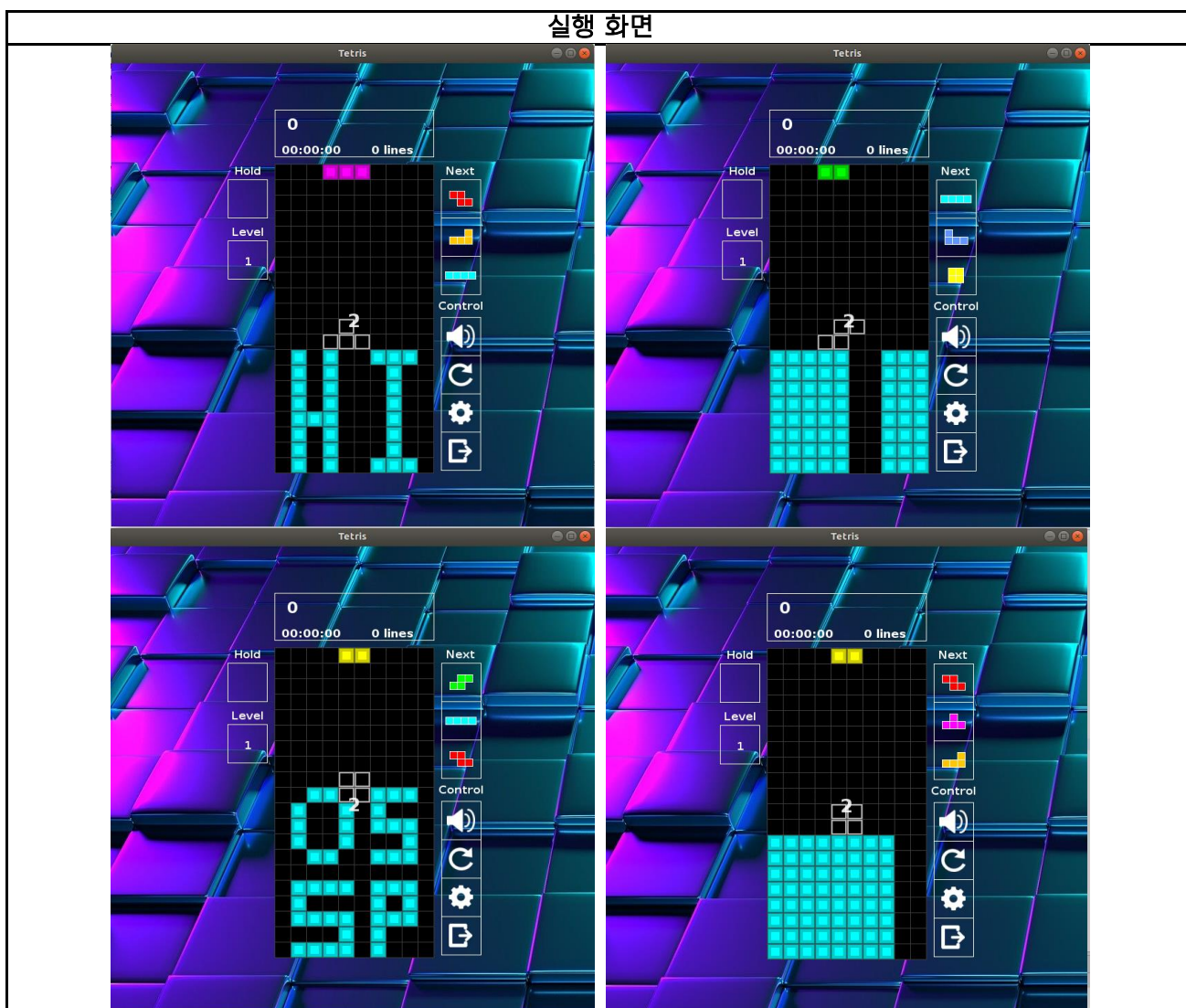
```
private void mute() {
    if(soundplay) {
        bgm_sound.stop();
        soundplay = false;
    }else {
        bgm_sound.play();
        soundplay = true;
    }
}
```

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



7. 쌓인 블록 구현

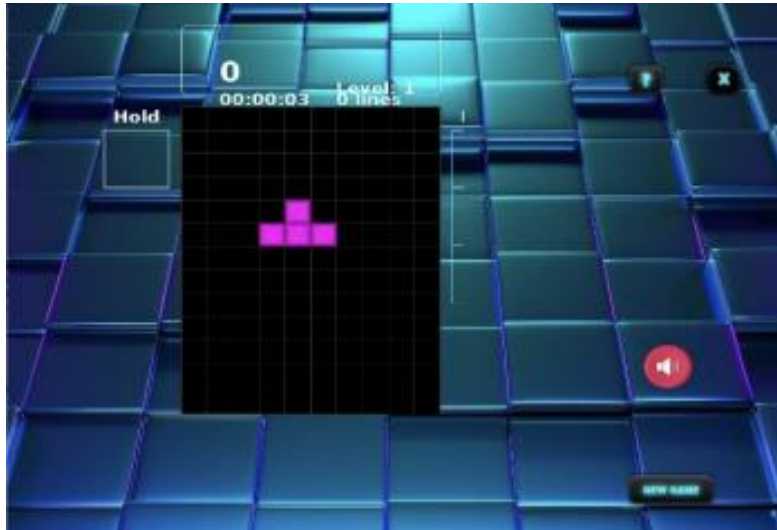
기존의 프로젝트에서는 컨테이너가 비어 있는 상태에서 게임을 시작한다. 다양한 게임방법을 제공하여 플레이어의 흥미를 더하기 위해 컨테이너에 블록들이 쌓여 있는 상태에서 게임을 시작할 수 있게 한다.



8. Frame Resizing

문제

기존의 게임에서는 전체 창 크기 조절은 가능하였으나 게임화면이 늘어나지를 않아 많은 여백이 생겼다. 그리고 게임화면이 창의 크기를 키우게 되면 배경 뒤로 이미지가 사라졌었다.



해결방안

삭제 : `frame.add(background);`

추가 : `g.drawImage(backgroundImage.getImage(), 0, 0, null);`

게임화면이 제한된 크기로 추가가 되어있어서 더 이상 늘어나지 않는 오류를 창의 크기에 맞게 비례해서 게임 화면이 그려지도록 코드를 수정하였다.

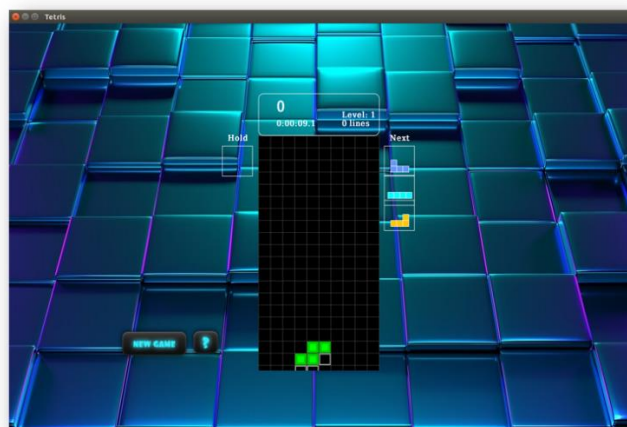
```
game.setSOR_W(frame.getSize().width/homeButtonLocationXCoefficient,frame.getSize().height);
game.setDSP_W(frame.getSize().width/gameDSPCoefficient);
game.drawTo(Graphics2D(g), (int)(frame.getSize().width - game.xoffset*2 - game.boxsize*2 - game.FIELD_W)/2, (int)(frame.getSize().height/8));
```

게임을 그리는 메서드의 파라미터를 전체화면의 Width와 Height 로 설정하여 게임화면이 조절 가능하도록 구현하였다. 이후 모든 모니터의 크기가 가로보다는 세로가 짧다는 가정하에 게임의 높이가 모니터의 세로의 크기보다 더 늘어나지 않게 하였다. 모니터의 세로의 길이가 되면 전체창은 양옆으로만 늘어날 수 있게 되며 이에 따라 게임화면도 양옆으로만 이동 가능하게 조정하였다.

구현 코드	블록 크기 조정
<p>사용자 모니터의 Height 보다 창의 크기가 늘어나는 이후부터는 게임화면의 크기가 고정되고 늘어나는 창의 x 축으로 이동합니다.</p> <pre> if(FIELD_H<=real_y) { g.fillRect(x, y, FIELD_W, FIELD_H); //게임창 구성 field_w=FIELD_W; field_h= FIELD_H; sqr = SQR_W; dsp = DSP_W; }else { g.fillRect(x, y, field_w, field_h); } </pre>	<pre> for (int j = 0; j < H; j++) { if (board[i][j] == 0) continue; if(FIELD_H<=real_y) { g.setColor(COLORS[board[i][j]]); g.fillRect(x + i * SQR_W, y + j * SQR_W, SQR_W, SQR_W); g.setColor(C_PIECE_HIGHLIGHT); g.fillRect(x + i * SQR_W, y + j * SQR_W, SQR_W, SQR_W); g.setColor(COLORS[board[i][j]]); g.fillRect(x + i * SQR_W + 6, y + j * SQR_W + 6, SQR_W - 11, SQR_W - 11); g.setColor(C_SHADOW); g.drawRect(x + i * SQR_W + 1, y + j * SQR_W + 1, SQR_W - 2, SQR_W - 2); }else { g.setColor(COLORS[board[i][j]]); g.fillRect(x + i * sqr, y + j * sqr, sqr, sqr); g.setColor(C_PIECE_HIGHLIGHT); g.fillRect(x + i * sqr, y + j * sqr, sqr, sqr); g.setColor(COLORS[board[i][j]]); g.fillRect(x + i * sqr + 6, y + j * sqr + 6, sqr - 11, sqr - 11); g.setColor(C_SHADOW); g.drawRect(x + i * sqr + 1, y + j * sqr + 1, sqr - 2, sqr - 2); } } </pre>
격자 크기 조정	Hold 박스 조정
<pre> g.setColor(C_BORDER); for (int i = 0; i < W; i++) { for (int j = 0; j < H; j++) if(FIELD_H<=real_y) { g.drawRect(x + i * SQR_W, y + j * SQR_W, SQR_W, SQR_W); sqr = SQR_W; }else { g.drawRect(x + i * sqr, y + j * sqr, sqr, sqr); } } </pre>	<pre> // holdBox와 hold글씨 그리기 g.setColor(Color.WHITE); g.setFont(F_UI); if(FIELD_H<=real_y) { drawCentered(g, "Hold", x - xoffset - boxsize/2, y + g.getFont().getSize()); g.drawRect(x - xoffset - boxsize, y + yoffset, boxsize, boxsize); hold = g.getFont().getSize(); }else { drawCentered(g, "Hold", x - xoffset - boxsize/2, y + hold); g.drawRect(x - xoffset - boxsize, y + yoffset, boxsize, boxsize); } // holdBox안의 테트리스 그리기 if (stored != -1) drawTetrimino(g, stored, x - xoffset - boxsize/2, y + yoffset + boxsize/2, blocksize); // hold </pre>
대기 블록창/상단 보드판 조정	버튼위치 조정
<pre> for (int i = 0; i < AHEAD; i++) { //대기하는 블록 창 g.setColor(Color.WHITE); if(FIELD_H<=real_y) { g.drawRect(x + FIELD_W + xoffset, y + boxsize*i+ yoffset, boxsize, boxsize); drawTetrimino(g, fMoves[i], x + FIELD_W + xoffset+ boxsize/2, y + boxsize*i+ boxsize/2 + yoffset, blocksize); }else { g.drawRect(x +field w + xoffset, y + boxsize*i+ yoffset, boxsize, boxsize); drawTetrimino(g, fMoves[i], x + field_w + xoffset+ boxsize/2, y + boxsize*i+ boxsize/2 + yoffset, blocksize); } } //보드판(상단) g.setColor (Color.WHITE); if(FIELD_H<=real_y) { g.drawRect(x, y-DSP_W-yoffset/2, SQR_W*10, DSP_W); } g.drawRect(x, y-dsp-yoffset/2, sqr*10, dsp); </pre>	<pre> int right_x = FIELD_H<=real_y ? x + FIELD_W + xoffset : x +field_w + xoffset; int right_y = (int) (y + boxsize*(AHEAD+1)); g.setColor (Color.WHITE); drawCentered(g, "Control", right_x + boxsize/2, right_y-yoffset/2); TetrisRenderer.muteButton.setSize(boxsize,boxsize); TetrisRenderer.muteButton.setLocation(right_x, right_y); g.drawRect(right_x, right_y, boxsize, boxsize); TetrisRenderer.soundButton.setSize(boxsize,boxsize); TetrisRenderer.soundButton.setLocation(right_x, right_y); g.drawRect(right_x, right_y, boxsize, boxsize); TetrisRenderer.newButton.setSize(boxsize,boxsize); TetrisRenderer.newButton.setLocation(right_x, right_y +boxsize); g.drawRect(right_x, right_y +boxsize, boxsize, boxsize); TetrisRenderer.keyButton.setSize(boxsize,boxsize); TetrisRenderer.keyButton.setLocation(right_x, right_y +boxsize*2); g.drawRect(right_x, right_y +boxsize*2, boxsize, boxsize); TetrisRenderer.homeButton.setSize(boxsize,boxsize); TetrisRenderer.homeButton.setLocation(right_x, right_y +boxsize*3); g.drawRect(right_x, right_y +boxsize*3, boxsize, boxsize); </pre>

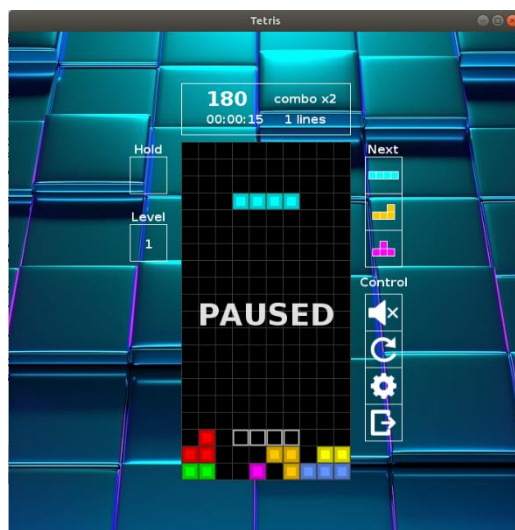


배경화면 UX/UI 개선 문제



해결방안

버튼을 적절한 아이콘으로 변경하고, 테트리스 구역 옆에 창을 컨트롤 할 수 있는 버튼을 쌓았다.



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	100%								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	50%								김택원,김정률,전영인	

2. 역할 분담

김택원(팀장)

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

김정률

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

전영인

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

3. 참고문헌

“프로젝트 OS 팀 github”, <https://github.com/CSID-DGU/2019-2-OSSPC-OS-3.git>

“프로젝트 PAIS 팀 github”, <https://github.com/CSID-DGU/2019-1-OSSPC-PAIS-1>

“원작자 PSNB92 github”, <https://github.com/PSNB92/Tetris.git>

FrameResizing , <https://cadaworld.tistory.com/24?category=684527>

FrameResizing, <http://zetcode.com/tutorials/javaswingtutorial/resizablecomponent/>, Feb 1, 2019