Mobile Programing (MS)

Term Project : App Develop Planning 담당 교수 : 정우진 교수님

모바일시스템공학과 김태경 (32211203)
2024.12.17

앱 개발 보고서 : Button Whacker

1. 개요

게임 이름: Button Whacker

개발 목적: 안드로이드 스튜디오를 이용한 모바일 앱 개발의 실습 프로젝트로, UI/UX 설계, 이벤트 처리, 쓰레드의 관리 등 모바일 개발의 기본 개념을 학습하는데 중점을 두었습니다.

게임 설명: Button Whacker는 5x5 그리드에서 무작위로 나타나는 버튼을 터치하여 점수를 획득하는 간단한 캐주얼 게임입니다. 불이 들어오는 버튼을 클릭하여 점수를 얻고, 잘못된 버튼을 클릭하면 감점됩니다. 제한 시간 내에 버튼를 클릭하지 못하거나 점수가 0 이하로 떨어지면 게임이 종료됩니다.

2. 주요 기능 설명 (Key Features)

기능 1: 5x5 버튼 배열

- 게임의 핵심 인터페이스는 5x5 그리드로 배치된 버튼 배열입니다. 각 버튼은 무작위로 활성화되며, 플레이어는 시간 내에 활성화된 버튼을 클릭해야 합니다.
- 버튼 배열은 GridLayout을 사용해 쉽게 구현되었으며, 각 버튼은 Button 위젯으로 정의되었습니다. Random 클래스를 이용하여 무작위로 하나의 버튼이 선택되면, 그 버튼이 노란색으로 활성화됩니다. 이는 setBackgroundColor 메서드를 이용하여 색상을 변경함으로써 시각적으로 불이 들어온 것처럼 표현하였습니다.

기능 2: 점수 관리

- Button Whacker의 점수 시스템은 게임의 핵심 요소입니다. 게임 시작 시 플레이어는 100점으로 시작하며, 올바른 버튼을 클릭하면 10점이 증가하고, 잘못된 버튼을 클릭하거나 클릭하지 않으면 10점이 감점됩니다.
- 실시간 점수는 화면 우측 상단의 TextView로 표시되며, 점수가 0 이하로 떨어지면 게임이 종료됩니다. 점수 계산 로직은 간단하면서도 직관적으로 설계되었습니다.

기능 3: 난이도 조절

- 난이도는 Easy, Normal, Hard 세 가지가 제공되며, 난이도에 따라 버튼 클릭 제한 시간이 달라집니다. 난이도는 PopupMenu를 통해 선택할 수 있으며, 선택한 난이도에 따라 버튼이 활성화된 후 제한 시간이 달라집니다. Easy 모드에서는 2초, Normal은 1초, Hard는 0.5초로 설정됩니다.
- 난이도는 메인 화면에서 PopupMenu를 통해 선택할 수 있으며, 선택된 난이도에 따라 게임의 시간 제한이 조정됩니다.

기능 4: 게임오버 처리

- 플레이어가 제한 시간 내에 버튼을 클릭하지 못하거나 점수가 0 이하로 떨어지면 게임은 종료됩니다. 게임 오버가 발생하면 GameOverActivity로 전환되며, 최종 점수가 표시됩니다. 게임오버 화면에서는 다시 시작하거나 메인 화면으로 돌아갈 수 있는 옵션을 제공합니다.

3. UI/UX 설계 (UI/UX Design)

메인 화면

- 메인 화면에는 "게임 시작", "난이도 선택", "게임 종료" 버튼이 배치되어 있습니다.
- 난이도 선택 버튼을 클릭하면 PopupMenu가 열리고, Easy, Normal, Hard 중 하나를 선택할 수 있습니다.
- 게임 시작 버튼을 클릭하면 GameActivity로 전환되며, 선택한 난이도에 따라 게임이 시작됩니다.

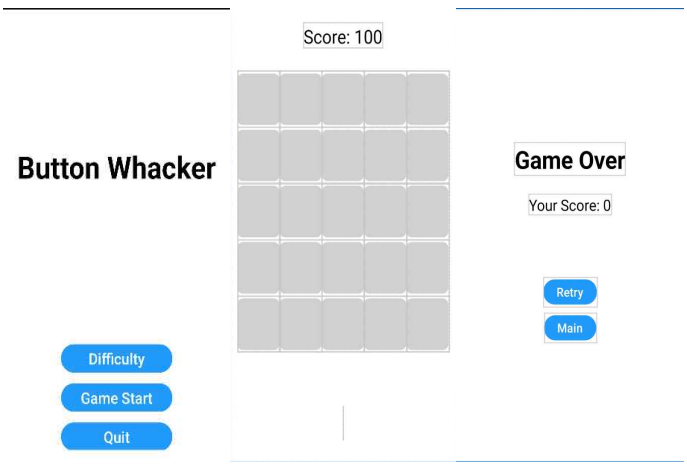
게임 화면

- 게임 화면은 5x5 그리드로 구성된 버튼 배열과 점수 표시 텍스트뷰로 구성됩니다.
- 상단에는 플레이어의 점수가 실시간으로 표시되며, 가운데에는 버튼 배열이 배치되어 있습니다. 버튼은 클릭 가능할 때 노란색으로 표시되며, 클릭하면 색상이 변합니다.

게임 오버 화면

- 플레이어가 제한 시간 내 버튼을 클릭하지 못했거나 점수가 0 이하로 떨어지면 게임 오버 화면이 표시됩니다. 최종 점수는 게임오버 화면에서 확인할 수 있으며, "재시작" 버튼을 클릭하여 게임을 다시 시작하거나, "메인 화면" 버튼을 클릭해 메인 화면으로 돌아갈 수 있습니다.

레이아웃 사진 ->



4. 기술 설계 및 코드 분석 (Technical Design & Code Analysis)

4.1 MainActivity (메인 화면)

기능 : 게임의 메인 화면으로, 사용자에게 난이도 선택, 게임 시작, 종료 기능을 제공합니다.

4,1,1 세부 기능 요약

- 난이도 선택: PopupMenu를 통해 사용자가 난이도를 선택할 수 있습니다. 선택된 난이도는 Intent를 사용하여 GameActivity로 전달됩니다.
- 게임 시작: "Game Start" 버튼을 눌러 게임을 시작할 수 있습니다.
- 게임 종료: "Quit" 버튼을 눌러 애플리케이션을 종료할 수 있습니다.

4.1.2 MainActivity 코드 분석

1) **난이도 선택**

```
// 난이도 선택 PopupMenu를 표시할 버튼
Button selectDifficultyButton = findViewById(R.id.select_difficulty_button);
selectDifficultyButton.setOnClickListener(new View.OnClickListener()
{
    @Override
    public void onClick(View v)
    {
        showDifficultyPopupMenu(v); // PopupMenu 표시
    }
});
```

selectDifficultyButton은 난이도를 선택할 수 있는 버튼입니다. 버튼 클릭 시 showDifficultyPopupMenu() 메서드를 호출하여 난이도를 선택할 수 있는 팝업 메뉴를 화면에 표시합니다.

2) **PopupMenu를 통한 난이도 선택**

```
// 난이도 선택 PopupMenu를 생성
private void showDifficultyPopupMenu(View view)  1 usage
{
    // PopupMenu 객체 생성
    PopupMenu popupMenu = new PopupMenu( context: MainActivity.this, view);
    // 메뉴 항목을 추가
    popupMenu.getMenu().add("Easy");
    popupMenu.getMenu().add("Normal");
    popupMenu.getMenu().add("Hard");

    // 메뉴 항목 클릭 시 이벤트 처리
    popupMenu.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener()
    {
        @Override
        public boolean onMenuItemClick(MenuItem item)
        {
            // 선택한 항목에 따른 난이도 설정
            difficultyLevel = item.getTitle().toString();
            Toast.makeText( context: MainActivity.this,  text: "Selected Difficulty: "
                    + difficultyLevel, Toast.LENGTH_SHORT).show();
            return true;
        }
    });

    // PopupMenu 표시
    popupMenu.show();
}
```

- PopupMenu는 화면에 난이도 선택을 위한 메뉴를 표시하는 역할을 합니다. 메뉴 항목은 "Easy", "Normal", "Hard"로 구성되어 있으며, 선택된 항목은 difficultyLevel 변수에 저장됩니다.

- 사용자가 난이도를 선택하면 선택한 난이도가 Toast 메시지로 표시되며, 이후 게임 시작 시 이 난이도가 게임에 반영됩니다.

3) **게임 시작**

```
// 게임 시작 버튼
Button startGameButton = findViewById(R.id.start_game_button);
startGameButton.setOnClickListener(new View.OnClickListener()
{
    @Override
    public void onClick(View v)
    {
        // 난이도를 GameActivity로 전달하여 게임 시작
        Intent intent = new Intent( packageContext: MainActivity.this, GameActivity.class);
        intent.putExtra( name: "difficulty", difficultyLevel); // 난이도를 전달
        startActivity(intent);
    }
});
```

- startGameButton은 게임을 시작하는 버튼입니다. 버튼 클릭 시 Intent를 생성하여 GameActivity로 전환되며, Intent에 선택된 난이도를 전달합니다.
- putExtra() 메서드를 사용해 난이도를 전달하고, startActivity(intent)로 화면을 전환합니다.

4) **게임 종료**

```
// 게임 종료 버튼
Button exitGameButton = findViewById(R.id.exit_game_button);
exitGameButton.setOnClickListener(new View.OnClickListener()
{
    @Override
    public void onClick(View v)
    {
        finish(); // 현재 액티비티 종료
    }
});
```

- exitGameButton은 게임을 종료하는 버튼입니다. 버튼을 클릭하면 finish() 메서드가 호출되어 MainActivity가 종료됩니다.

4.2 GameActivity (게임 화면)

기능: 게임의 핵심 로직을 담당하는 화면입니다.

4.2.1 세부 기능 요약

- 5x5 버튼 배열: 무작위로 버튼을 활성화하고, 플레이어가 버튼을 클릭했을 때 점수를 증가시키거나 감소시키는 로직을 처리합니다.
- 점수 시스템: 점수는 실시간으로 업데이트되며, 잘못된 버튼을 클릭하거나 제한 시간 내에 버튼을 클릭하지 못할 경우 감점됩니다.
- 난이도에 따른 제한 시간 조정: 난이도에 따라 버튼 클릭 제한 시간이 다르게 설정되며, 시간 초과 시 게임 오버 처리가 됩니다.
- 게임 오버 처리: 플레이어가 버튼을 클릭하지 않거나 점수가 0 이하로 떨어지면 게임이 종료되고 GameOverActivity로 전환됩니다.

4.2.2. GameActivity 코드 분석

1) **난이도에 따른 시간 제한 설정**

```
// 난이도에 따른 시간 설정
switch (difficulty)
{
    case "Easy":
        timeLimit = 2000; // Easy = 2초
        break;
    case "Normal":
        timeLimit = 1000; // Normal = 1초
        break;
    case "Hard":
        timeLimit = 500; // Hard = 0.5초
        break;
}
```

- MainActivity에서 전달된 난이도를 기준으로 각 난이도에 따른 시간 제한을 설정합니다. Easy 모드에서는 2초, Normal 모드에서는 1초, Hard 모드에서는 0.5초로 설정되어 플레이어의 반응 속도를 테스트할 수 있습니다.

2) **5x5 버튼 배열 생성 및 클릭 이벤트 처리**

```
// 5x5 버튼 배열 설정 및 초기화 (+색깔 : 회색 설정)
GridLayout buttonGrid = findViewById(R.id.button_grid);
for (int row = 0; row < 5; row++)
{
    for (int col = 0; col < 5; col++)
    {
        String buttonID = "button_" + (row * 5 + col);
        int resID = getResources().getIdentifier(buttonID, defType: "id", getPackageName());
        buttons[row][col] = findViewById(resID); // 버튼 연결

        // 버튼 색깔 설정 : 회색
        buttons[row][col].setBackgroundColor(getResources().getColor(android.R.color.darker_gray));

        // 버튼 클릭 이벤트 처리
        buttons[row][col].setOnClickListener(v -> handleButtonClick((Button) v));
    }
}
```

- GridLayout을 사용하여 5x5 버튼 배열을 생성합니다. 각 버튼은 setOnClickListener()를 통해 클릭 이벤트를 처리할 수 있도록 설정됩니다. 사용자가 버튼을 클릭하면 handleButtonClick() 메서드가 호출되어 클릭한 버튼이 올바른지 확인합니다.

3) **버튼 클릭 처리**

```
// 버튼 클릭 이벤트 처리
private void handleButtonClick(final Button clickedButton)  1 usage
{
    // 클릭한 버튼이 활성화된 버튼인지 확인
    if (clickedButton == activeButton)
    {
        // 올바른 버튼 클릭 시 파란색으로 0.3초간 표시
        clickedButton.setBackgroundColor(getResources().getColor(android.R.color.holo_blue_light));
        score += 10; // 점수 증가
        updateScore();
        handler.removeCallbacks(gameOverRunnable); // 타이머 취소

        // 0.3초 후 버튼 색상 원래대로 돌리기
        handler.postDelayed(() ->
        {
            clickedButton.setBackgroundColor(getResources().getColor(android.R.color.darker_gray));
            activateRandomButton(); // 새로운 버튼 활성화
        }, delayMillis: 300); // 0.3초 후 실행
```

- 사용자가 버튼을 클릭했을 때, 해당 버튼이 활성화된 버튼인지 확인하고, 올바른 버튼일 경우 점수를 증가시키고 잘못된 버튼일 경우 감점합니다.

```
    } else {
        // 잘못된 버튼 클릭 시 빨간색으로 0.3초간 표시
        clickedButton.setBackgroundColor(getResources().getColor(android.R.color.holo_red_light));
        score -= 10; // 잘못된 클릭으로 감점
        updateScore();

        // 0.3초 후 버튼 색상 원래대로 돌리기
        handler.postDelayed(() ->
        {
            clickedButton.setBackgroundColor(getResources().getColor(android.R.color.darker_gray));
            checkGameOver();
        }, delayMillis: 300); // 0.3초 후 실행
    }
}
```

- 올바른 버튼을 클릭하면 0.3초 후 새로운 무작위 버튼을 활성화하고, 잘못된 버튼을 클릭했을 경우 게임 오버 여부를 확인합니다.

4) **게임 오버 처리**

```
private void gameOver()  2 usages
{
    // 게임 오버 처리
    Intent gameOverIntent = new Intent( packageContext: GameActivity.this, GameOverActivity.class);
    gameOverIntent.putExtra( name: "FINAL_SCORE", score); // 최종 점수 전달
    gameOverIntent.putExtra( name: "difficulty", getIntent().getStringExtra( name: "difficulty"));
    startActivity(gameOverIntent);
    finish(); // 현재 액티비티 종료
}
```

- 플레이어의 점수가 0 이하로 떨어지면 게임 오버가 처리됩니다. Intent를 사용하여 GameOverActivity로 전환하며, 현재 점수를 전달합니다.

4.3 GameOverActivity (게임 오버 화면)

기능: 게임이 종료된 후 최종 점수를 보여주고, 재시작하거나 메인 화면으로 돌아가는 기능을 제공합니다.

4.3.1 세부 기능 요약

- 최종 점수 표시: TextView를 사용해 게임 종료 시 플레이어의 최종 점수를 화면에 표시합니다.
- 재시작 버튼: 사용자가 "Retry" 버튼을 클릭하면 게임이 다시 시작됩니다.
- 메인 화면으로 돌아가기: 사용자가 "Main" 버튼을 클릭하면 메인 화면으로 돌아갑니다.

4.3.2 GameOverActivity 코드 분석

1) **최종 점수 표시**

```
// Intent로부터 점수 불러오기
Intent intent = getIntent();
int finalScore = intent.getIntExtra( name: "FINAL_SCORE",  defaultValue: 0);

// 점수 표시 TextView
TextView scoreTextView = findViewById(R.id.final_score_text);
scoreTextView.setText("Your Score: " + finalScore); // 최종 점수 표시
```

- GameActivity에서 전달된 최종 점수를 받아와 TextView에 표시합니다.

2) **재시작 버튼**

```
// 재시작 버튼 (GameActivity로 다시 돌아가기)
Button restartButton = findViewById(R.id.restart_button);
restartButton.setOnClickListener(v ->
{
    Intent restartIntent = new Intent( packageContext: GameOverActivity.this,
            kr.ac.dankook.mobsys.buttonwhacker.GameActivity.class);
    restartIntent.putExtra( name: "difficulty", getIntent().getStringExtra
            ( name: "difficulty")); // 난이도를 다시 전달
    startActivity(restartIntent);
    finish(); // 현재 액티비티 종료하고 게임 재시작
});
```

- "Retry" 버튼을 클릭하면 게임을 다시 시작할 수 있습니다. GameActivity로 다시 이동하며, 기존에 선택한 난이도도 함께 전달됩니다.

3) **메인 화면으로 돌아가기**

```
// 메인 화면으로 돌아가기 버튼
Button mainMenuButton = findViewById(R.id.main_menu_button);
mainMenuButton.setOnClickListener(v ->
{
    Intent mainMenuIntent = new Intent( packageContext: GameOverActivity.this,
            kr.ac.dankook.mobsys.buttonwhacker.MainActivity.class);
    startActivity(mainMenuIntent);
    finish(); // 메인 화면으로 돌아가기
});
```

- "Main" 버튼을 클릭하면 MainActivity로 돌아가며, 새로운 게임을 시작할 수 있습니다.

5. 작업중 오류 및 버그 해결

5.1 **오류 1: 버튼 색상 처리 문제**

오류 내용:

버튼을 클릭할 때, 잘못된 클릭 시 빨간색, 올바른 클릭 시 파란색으로 0.3초 동안 표시되도록 설정했지만, 같은 버튼에 연속으로 불이 들어오는 경우 빨간색 또는 파란색이 제대로 표시되지 않고, 0.7초 동안 노란색 불만 표시되는 문제가 발생했습니다.

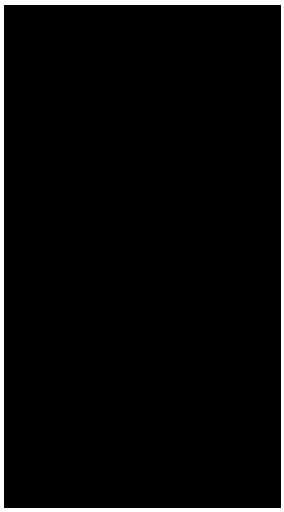
해결 방법:

클릭 후 빨간색이나 파란색으로 표시되는 0.3초 동안 버튼이 유지된 후에야 새로운 노란색 불이 들어오도록 0.3초 딜레이를 추가했습니다. 이를 통해 시각적 피드백이 명확하게 표시되도록 조치했습니다.

결과:

버튼 클릭 시 빨간색 또는 파란색으로 0.3초 동안 표시된 후 새로운 버튼이 활성화되어 시각적 피드백이 명확해졌고, 문제없이 게임이 진행되었습니다.

5.2 **오류 2: 에뮬레이터 검은 화면 문제**



오류 내용:

Android Studio에서 에뮬레이터를 실행할 때 가끔 검은 화면이 나타나고 시간이 지나도 변경되지 않는 문제가 발생했습니다.

해결 방법:

에뮬레이터를 강제로 종료한 후 AVD Manager에서 Cold Boot Now 기능을 사용해 재부팅하여 문제를 해결했습니다.

방법:

Android Studio 상단에 있는 AVD Manager 클릭 또는 Tools > AVD Manager 선택.
실행 중인 에뮬레이터의 우측에 있는 더보기 아이콘을 클릭 후 Cold Boot Now 선택.

결과:

에뮬레이터가 정상적으로 부팅되었으며, 검은 화면 문제 없이 앱이 실행되었습니다.

5.3 **오류 3: 게임 시작 시 로딩 문제**

오류 내용:

게임 시작 후 로딩 중에 버튼에 불이 들어오는 카운트가 진행되어 게임이 원활하게 시작되지 않았습니다. 로딩 시간이 충분히 제공되지 않아 게임 플레이에 영향을 주었습니다.

해결 방법:

1안 : resume을 사용한 해결을 시도했으나 코드 수정이 많이 필요하여 하지 않았습니다.

게임 시작 전에 3초 동안 카운트다운을 표시하는 기능을 추가하여, 로딩 중에 불이 들어오지 않도록 했습니다. 이를 통해 게임이 시작되기 전에 플레이어에게 충분한 준비 시간을 제공하였습니다.

결과:

카운트다운 기능이 추가되어 게임 시작 전에 충분한 로딩 시간이 제공되었고, 플레이어는 원활하게 게임을 시작할 수 있었습니다.

버그 해결 코드

```
// 카운트 다운 3, 2, 1 후에 게임 시작
private void startCountdown()  1 usage
{
    countdownText.setText("3");
    handler.postDelayed(() -> countdownText.setText("2"),  delayMillis: 1000);
    handler.postDelayed(() -> countdownText.setText("1"),  delayMillis: 2000);
    handler.postDelayed(() ->
    {
        countdownText.setText(""); // 카운트 다운 완료 후 텍스트 제거
        activateRandomButton();
    },  delayMillis: 3000);
}
```

5.4 **오류 4: No speakable text present 오류**

오류 내용:

XML에서 버튼을 구성할 때 No speakable text present라는 경고가 발생했습니다. 이 오류는 접근성 문제로, 버튼에 시각적 또는 청각적 정보를 제공하지 않는다는 의미입니다.

해결 방법:

이 경고는 앱의 실제 실행에 영향을 주지 않는 경고 메시지였으므로 게임의 작동에는 문제가 없다고 판단하여, 무시하고 개발을 진행하였습니다.

결과:

오류 메시지가 남아있었지만, 앱 실행에는 전혀 문제가 없음을 확인하였고, 게임은 문제없이 정상적으로 작동했습니다.

6. 게임 테스트 (Game Test)

테스트 시나리오

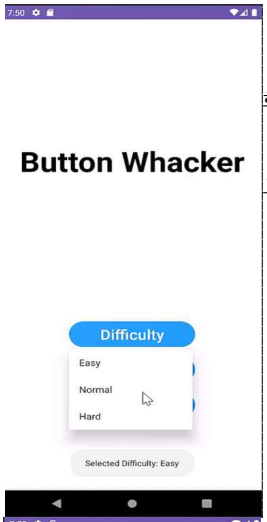
1) 메인 화면의 버튼들의 기능 작동 확인
    1.1) 난이도 조정 버튼 및 작동 로그 확인
    1.2) GameStart 버튼 작동 확인
    1.3) Quit 버튼 작동 확인
2) 게임 화면의 기능 작동 테스트
    2.1) 시작 카운트 다운
    2.2) 버튼들의 색깔 변화기능 확인
    2.3) 실시간 점수 변화 확인
3) 게임 오버 화면의 기능 테스트
    3.1) 최종 점수 확인
    3.2) Retry 버튼 작동 확인
    3.3) Main 버튼 작동 확인

* 메인 화면

* 난이도 및 로그

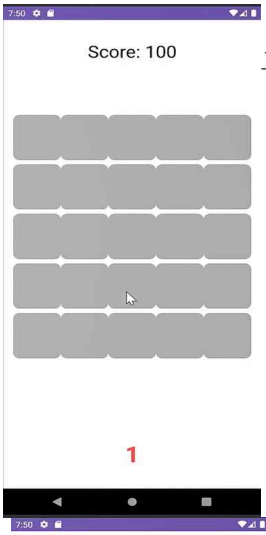
* 카운트 다운

* 파란불, 점수 확인



*
확인

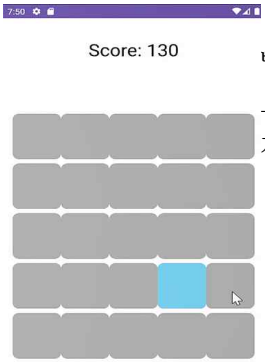
후



노란,

* 게임

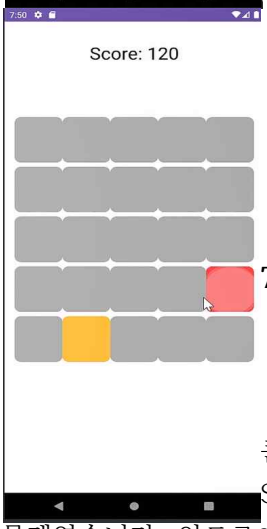
*

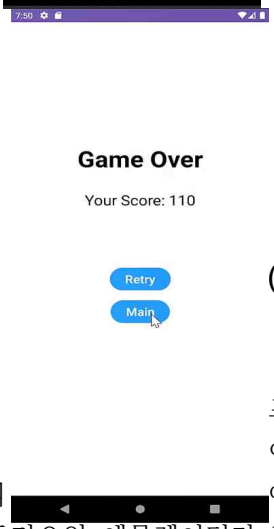


빨간불

오버 화면

게임 종료





7. 결론 (Conclusion)

느낀점

이번 프로젝트에서 가장 큰 어려움은 Android Studio의 에뮬레이터 자체의 문제였습니다. 안드로이드 스튜디오의 에뮬레이터가 작성한 코드에 문제가 있지 않음에도 검은 화면을 유지하며 작동하지 않아, 코드에서부터 문제가 있다고 착각하여 많은 시간을 낭비하게 되었고, 인터넷에서 확인하였던 간단한 해결책(버튼에 텍스트나 사운드를 추가하는 방법)은 버튼의 기능적과 맞지 않았습니다. 결국 에뮬레이터 문제는 무시하고, 앱 자체의 작동에는 지장이 없도록 조치하여 프로젝트를 진행한 후 마무리했습니다.

비슷한 문제로는 게임 시작 후 로딩 도중에 게임의 카운트다운이 진행되어 게임이 정상적으로 시작되지 않는 문제였습니다. 코드 자체에는 오류가 없었지만, 로딩 중에 게임이 시작되는 현상은 실행 환경에 따라 달라질 수 있다는 것을 깨달았습니다. 다양한 사용자의 환경을 고려해 UI에 카운트다운을 추가함으로써 문제를 해결했고, 사용자 친화적인 경험을 제공하기 위해 추가적인 수정도 진행했습니다.

이번 프로젝트를 통해 모바일 앱 개발의 여러 측면을 직접 체험할 수 있었고, UI/UX 설계와 이벤트 처리, 난이도 조정, 점수 관리 등의 기능을 성공적으로 구현했습니다. 아직 여러 프로그램을 많들어 본 것이 아니지만, 취업 후에도 간간히 취미로서 1인 개발의 영역에서 간단한 앱의 개발을 지속적으로 해볼만 하다는 생각이 들었습니다.

추가적으로 간단한게 구현해보고 싶은 기능이 있다면 테마와 디자인의 영역이 게임의 분위기에 영향을 많이 주는 것 같아 신경을 많이 써야 한다는 것을 느껴 좀 더 다듬어진 UI를 구현하고 싶다는 생각과, 게임의 기능적 요소로는 개인의 점수를 기록해두는 리더보드의 구현입니다. 이 추가구현 요소를 떠올렸을 당시에는 시간적 여건이 되지않아 구현하지 않았지만 충분히 시도해볼만한 아이디어라고 생각했습니다.

8. 참고문헌

* 안드로이드 스튜디오 에뮬레이터 검은화면 오류 해결
  https://gradler.tistory.com/28

* PopupMenu 만들기
  https://cheonjoosung.github.io/blog/an-popup-menu

* java handler 개념 및 구현
  https://ju-hy.tistory.com/62

* 그 외 모바일프로그래밍 과목 정우진 교수님 강의자료 참고