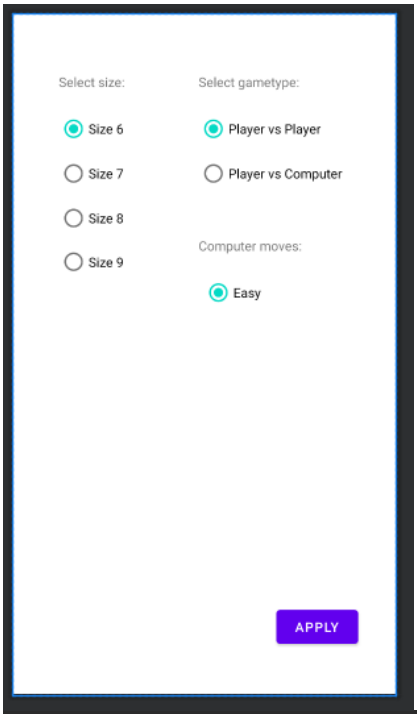
Gebze Technical University

CSE 222 Winter Project Report

DERVİŞ ALİ DUMAN

1801042626

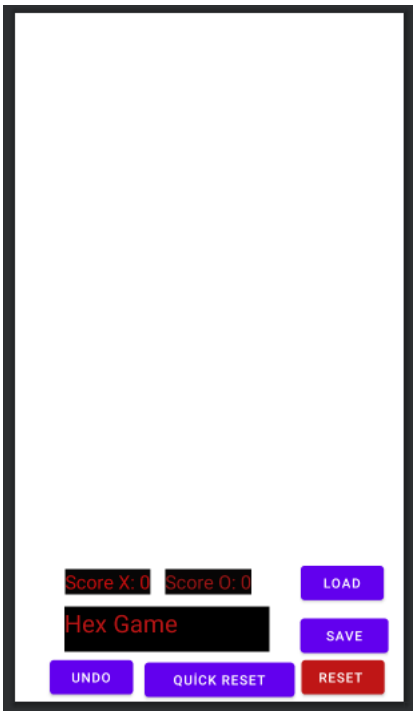
Code screenshots:



Main Activity: Here user can select board size and game type as player versus player or player versus computer.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Button b = (Button)findViewById(R.id.ApplyButton);
    b.setOnClickListener(new View.OnClickListener(){    //opening gameplay with selected radio buttons
        @Override
        public void onClick(View v) {

            Intent a = new Intent( packageContext: MainActivity.this, Game.class);
            a.putExtra( name: "board_size",board_size);
            a.putExtra( name: "game_type",game_type);
            finish();
            startActivityForResult(a, requestCode: 1);   //opening activity Game
        }
    });
}
```

Here if user selects raido buttons and apllys them. There will be open game with thoose informations.



Activity Game: Here all this gameplay occuring. There will be buttons created via Game.java file.

Game.java:

I choose Relative layout because I wanted to create 2D button array and put them on the layout.

```
RelativeLayout relativeLayout;
```

```
relativeLayout = findViewById(R.id.relative_main);
relativeLayout.setBackgroundColor(Color.rgb( red: 35,  green: 10,  blue: 10));
```

Code that below takes size and game type from main activity.

```
Intent a = getIntent();
board_size = a.getIntExtra( name: "board_size", defaultValue: 6);  //taking input from main activity
game_type = a.getIntExtra( name: "game_type", defaultValue: 1);
```

Then I created button and added them relative layout and button array one by one.

```
for (int i = 0; i < board_size; i++) {  //making buttons and adding interface and layout
    for (int j = 0; j < board_size; j++) {
        control_table[i][j]=0;
        Button button = new Button( context: this);
        int width = Resources.getSystem().getDisplayMetrics().widthPixels-180 ;

        RelativeLayout.LayoutParams layoutParams = new RelativeLayout.LayoutParams((int)(width/(board_size*1.5)), (int)(width/(board_size*1.5)));

        button.setText("");
        button.setTextSize(9);
        int sum=(i)+board_size*j;
        button.setId(sum);
        button.setOnClickListener(Clicked(i,j,button));
        button.setTranslationX((i)*(int)(width/(board_size*1.5)+4)+j*((int)(width/(board_size*1.5)+4)/2));
        button.setTranslationY((j+1)*(int)(width/(board_size*1.5)+4));

        buttons[i][j]=button;

        buttons[i][j].setBackgroundColor(Color.rgb( red: 150,  green: 25,  blue: 25));

        relativeLayout.addView(button, layoutParams);
    }
}
```

Quick reset button calls same activity with same parameters.

```
Button b2 = (Button)findViewById(R.id.qckreset);
b2.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View v) {    //resets board quickly
        Intent a = new Intent( packageContext: Game.this,Game.class);
        a.putExtra( name: "board_size",board_size);
        a.putExtra( name: "game_type",game_type);

        startActivityForResult(a, requestCode: 1);
        finish();

    }
});
```

Undo button .

```
Button b3 = (Button)findViewById(R.id.undobutton);
b3.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View v) {  //BUTTON UNDO, undos mooves
        if(move_count>0){
            buttons[all_move_logs[move_count-1][0]][all_move_logs[move_count-1][1]].setBackgroundColor(Color.rgb( red: 150, green: 25, blue: 25));
            buttons[all_move_logs[move_count-1][0]][all_move_logs[move_count-1][1]].setText("");
            move_count-=1;
        }
        if(game_type==2){
            buttons[all_move_logs[move_count-1][0]][all_move_logs[move_count-1][1]].setBackgroundColor(Color.rgb( red: 150, green: 25, blue: 25));
            buttons[all_move_logs[move_count-1][0]][all_move_logs[move_count-1][1]].setText("");
            move_count-=1;
        }

        fixTurns();
    }
});
```

Player makes move and saving moves for undo and saving game like this.

```
if (player1_turn) {              //player 1 plays
    if(buttons[i][j].getText()=="") {
        buttons[i][j].setBackgroundColor(Color.rgb( red: 25,  green: 25,  blue: 255));
        buttons[i][j].setText("X");

        lastx_coordinate=j+1;
        lasty_coordinate=i+1;

        all_move_logs[move_count][0]=i; //move logs saving
        all_move_logs[move_count][1]=j;
        all_move_logs[move_count][2]=0;
        move_count+=1;

        reset_control_table();
        scoreX=0;
        is_game_done_for_user1(lastx_coordinate,lasty_coordinate); //wincheck

        scoretextX.setText("Score X: "+scoreX);
```

Wincheck: It Works for last move of last player. It won't do job that souldn't do. It is very effective code.

```
if(flag1==1 && flag2 == 1){ //wincheck with flags
    informationtext.setText("WINNER X");
    Button b3 = (Button)findViewById(R.id.undobutton);
    b3.setEnabled(false);

    DetermineWinnerWay();
}
else if(game_type==1){

    informationtext.setText("Player2's Turn");
}
else{                                          //Computer plays
    informationtext.setText("Computer's Turn");
```

```
//checks for last move of user1 for if game end or not
public void is_game_done_for_user1(int coordinate_x,int coordinate_y){
    scoreX+=1;
    control_table[coordinate_y-1][coordinate_x-1]=1;


    for(int i=1;i<=board_size;i++){ //if we go most right or most left
        if(coordinate_y == 1 && coordinate_x==i)
            flag1=1;
        if(coordinate_y == board_size && coordinate_x==i)
            flag2=1;
    }
    //Moving around recursively
    if(coordinate_y-2>=0 && control_table[coordinate_y - 2][coordinate_x-1] == 0 && buttons[coordinate_y-2][coordinate_x-1].getText()=="X")
        is_game_done_for_user1(coordinate_x, coordinate_y: coordinate_y-1);
    if(coordinate_y-2>=0 && coordinate_x <board_size && control_table[coordinate_y-2][coordinate_x] == 0 && buttons[coordinate_y-2][coordinate_x].getText()=="X")
        is_game_done_for_user1( coordinate_x: coordinate_x+1, coordinate_y: coordinate_y-1);
    if(coordinate_x < board_size && control_table[coordinate_y-1][coordinate_x] == 0 && buttons[coordinate_y-1][coordinate_x].getText()=="X")
        is_game_done_for_user1( coordinate_x: coordinate_x+1,coordinate_y);
    if(coordinate_y < board_size && control_table[coordinate_y][coordinate_x-1] == 0 && buttons[coordinate_y][coordinate_x-1].getText()=="X")
        is_game_done_for_user1(coordinate_x, coordinate_y: coordinate_y+1);
    if(coordinate_y < board_size && coordinate_x- 2 >= 0 && control_table[coordinate_y][coordinate_x-2] == 0 && buttons[coordinate_y][coordinate_x-2].getText()=="X")
        is_game_done_for_user1( coordinate_x: coordinate_x-1, coordinate_y: coordinate_y+1);
    if(coordinate_x-2 >= 0 && control_table[coordinate_y - 1][coordinate_x -2] == 0 && buttons[coordinate_y-1][coordinate_x-2].getText()=="X")
        is_game_done_for_user1( coordinate_x: coordinate_x-1,coordinate_y);

}
```

Plays AI : ATTENTION: this code for minimax is not my work and pdf says that I can use code from open source. Even though I used open source code. My AI is powerless. I don’t know why because I didn’t understand minimax theorem exactly.

```
playAI();       //plays AI

/*buttons[lasty_coordinate-1][lastx_coordinate-1].setBackgroundColor(Color.rgb(25, 255, 25));
buttons[lasty_coordinate-1][lastx_coordinate-1].setText("O");*/
reset_control_table();
scoreO=0;
is_game_done_for_user2(lastx_coordinate,lasty_coordinate); //wincheck
scoretextO.setText("Score O: "+scoreO);

all_move_logs[move_count][0]=lasty_coordinate-1; //move logs saving
all_move_logs[move_count][1]=lastx_coordinate-1;
all_move_logs[move_count][2]=1;
move_count+=1;
```

Save button: Saves the size, moves and gametype in the save.txt.

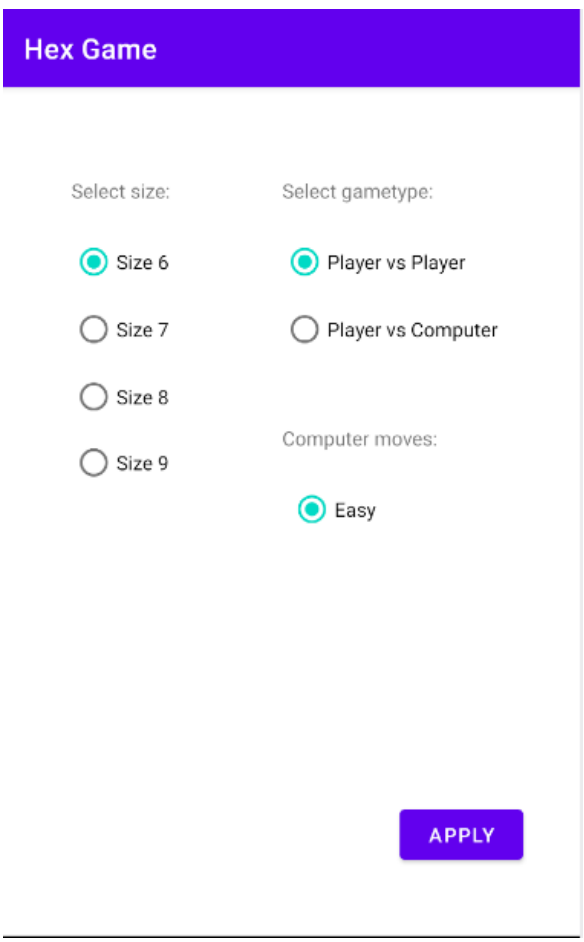
```
Button b4 = (Button)findViewById(R.id.savebutton);
b4.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View v) {   //SAVES THE GAME
        TextView informationtext= (TextView)findViewById(R.id.informationtext);
        String FILE_NAME = "save.txt";
        String text = "";
        FileOutputStream foutput = null;

        text+=""+board_size;    //Saving size, game type, move count
        text+="\n";
        text+=""+game_type;
        text+="\n";
        text+=""+move_count;
        text+="\n";

        for (int i = 0; i < move_count; i++){ //writing moves
            text+=""+all_move_logs[i][0];
            text+="\n";
            text+=""+all_move_logs[i][1];
            text+="\n";
            text+=""+all_move_logs[i][2];
            text+="\n";
        }

        try {
            foutput = openFileOutput(FILE_NAME, MODE_PRIVATE);
            foutput.write(text.getBytes());
            informationtext.setText("SAVED");
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
```

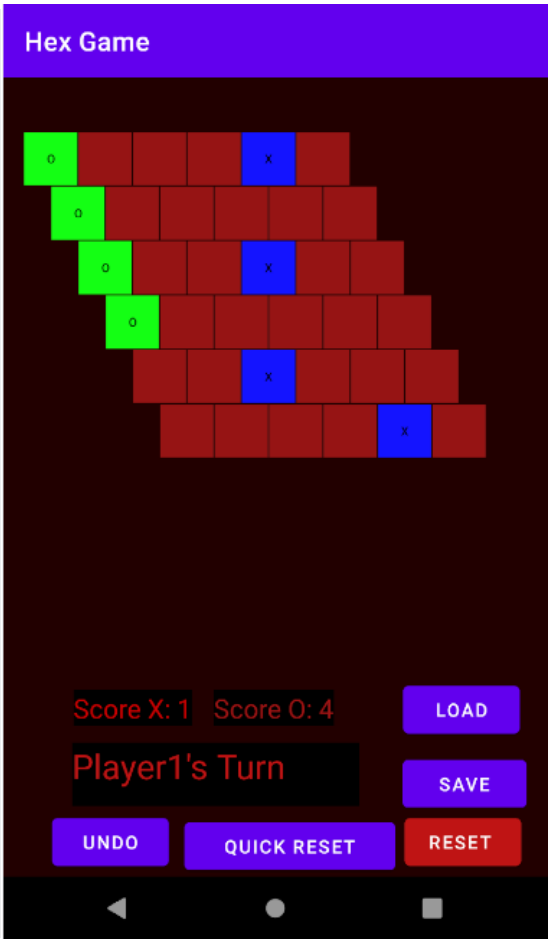
Load button: Loads the game with save.txt

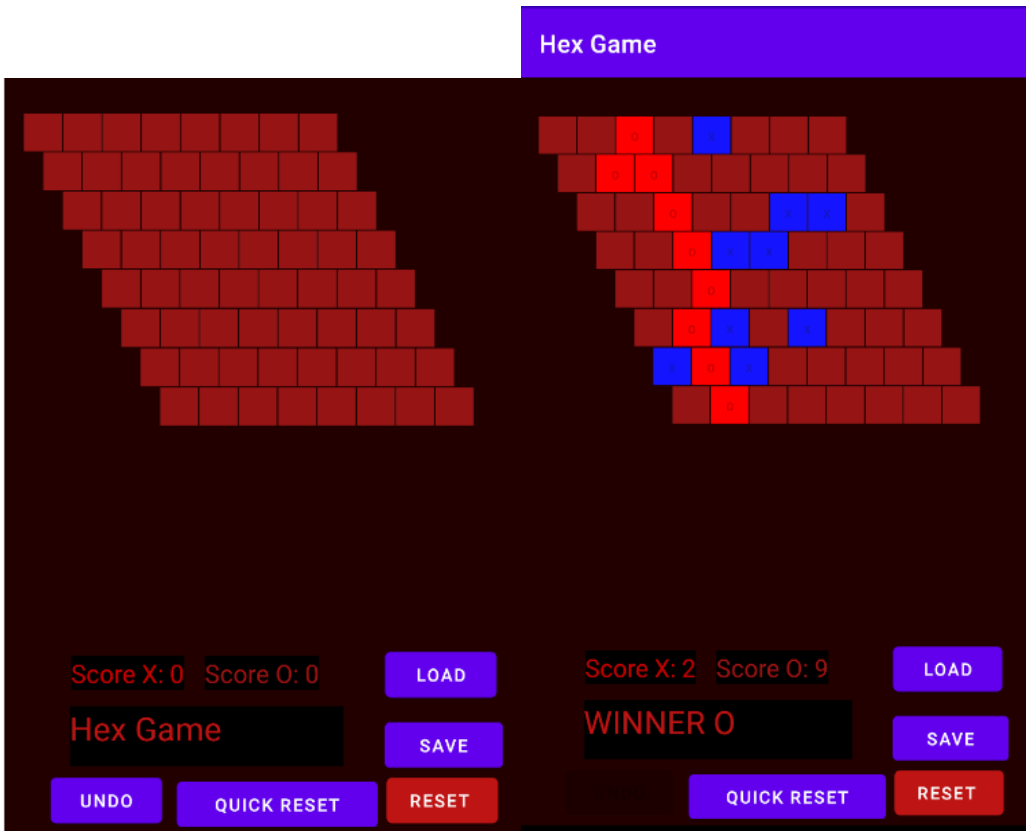
```
Button b5 = (Button)findViewById(R.id.real_load_button);
b5.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View v) {           //LOADİNG BOARD
        TextView informationtext= (TextView)findViewById(R.id.informationtext);
        FileInputStream finput = null;
        try {
            finput = openFileInput( name: "save.txt");
            InputStreamReader isr = new InputStreamReader(finput);
            BufferedReader br = new BufferedReader(isr);

            String line;

            int x;

            if ((line = br.readLine()) != null){
                x= Integer.parseInt(line);
                reset_game(x);  //reset with given size
            }

            if ((line = br.readLine()) != null){    //taking gametype
                x= Integer.parseInt(line);
                game_type=x;
            }

            if ((line = br.readLine()) != null){  //taking move count
                x= Integer.parseInt(line);
                move_count=x;
            }
```

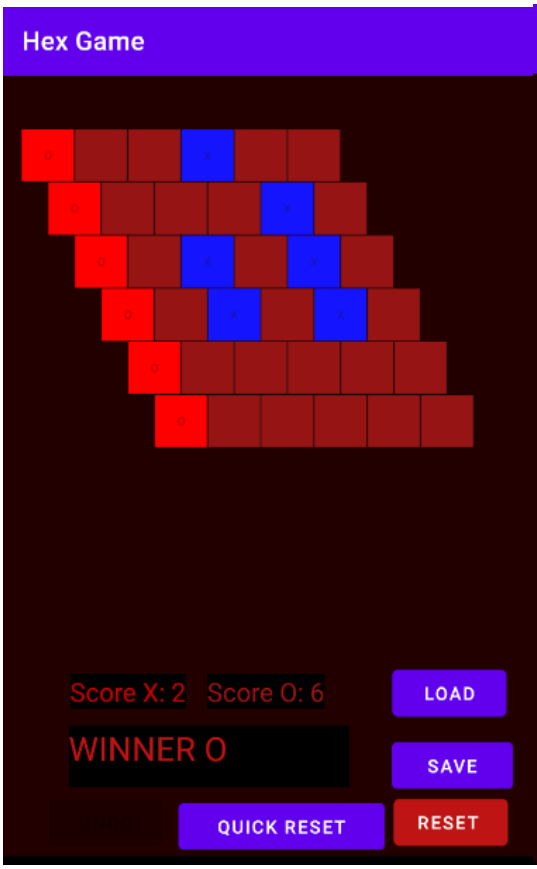
```
    for(int i=0 ; (line = br.readLine()) != null ; i++){ //taking board
        x= Integer.parseInt(line);
        all_move_logs[i][0]=x;
        line = br.readLine();
        x= Integer.parseInt(line);
        all_move_logs[i][1]=x;
        line = br.readLine();
        x= Integer.parseInt(line);
        all_move_logs[i][2]=x;
    }

    for(int i=0; i<move_count;i++){  //placing board

        if(all_move_logs[i][2] == 0){
            buttons[all_move_logs[i][0]][all_move_logs[i][1]].setText("X");
            buttons[all_move_logs[i][0]][all_move_logs[i][1]].setBackgroundColor(Color.rgb( red: 25, green: 25, blue: 255));
        }
        else{
            buttons[all_move_logs[i][0]][all_move_logs[i][1]].setText("O");
            buttons[all_move_logs[i][0]][all_move_logs[i][1]].setBackgroundColor(Color.rgb( red: 25, green: 255, blue: 25));
        }


    }

    fixTurns();
```

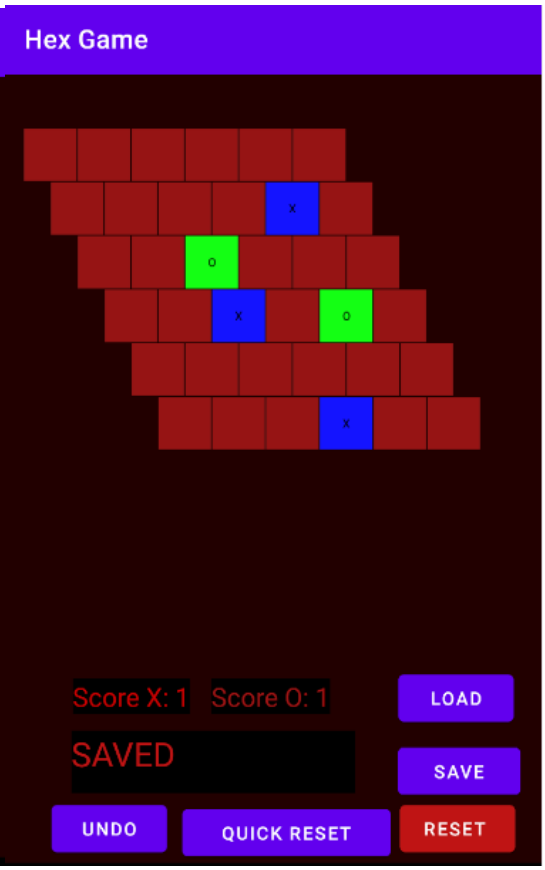
GAMEPLAY SCREENSHOTS:








Hex Game
Score X: 2
Score O: 6
WINNER O
LOAD
SAVE
QUICK RESET
RESET


Hex Game
Score X: 1
Score O: 1
SAVED
LOAD
SAVE
UNDO
QUICK RESET
RESET