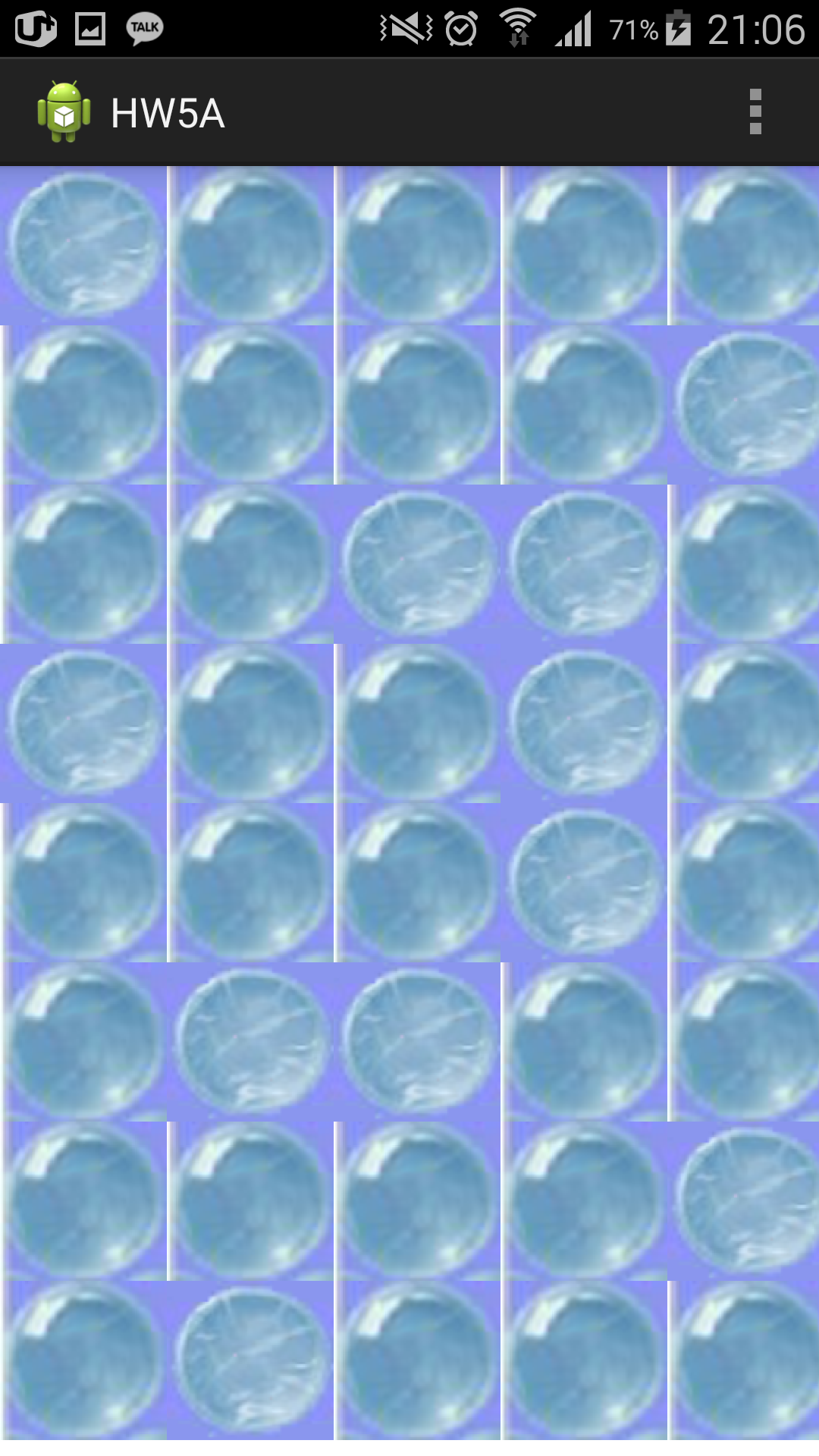
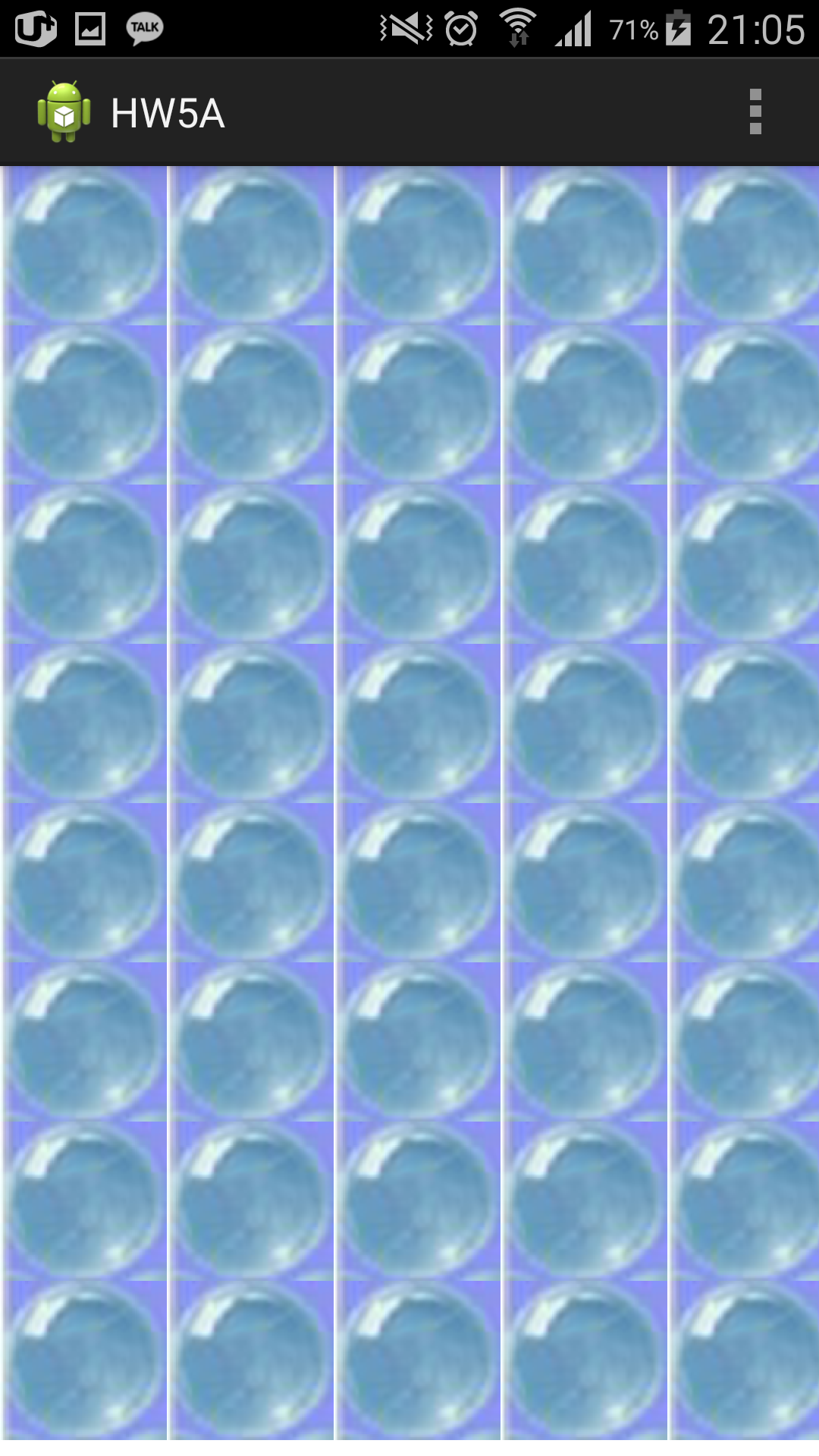
**Assignment 5**

**201133216 정유석**

**HW. #5-A: Bbok Bbok (Bubble)**

**ScreenShot 1**



**<First> <Some Touch>**

**Code 1.**

**MainActivity**

package com.example.hw5a;

import android.app.Activity;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

public class MainActivity extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.main, menu);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

// Handle action bar item clicks here. The action bar will

// automatically handle clicks on the Home/Up button, so long

// as you specify a parent activity in AndroidManifest.xml.

int id = item.getItemId();

if (id == R.id.action\_settings) {

return true;

}

return super.onOptionsItemSelected(item);

}

}

**MyDraw**

package com.example.hw5a;

import android.content.Context;

import android.content.res.Resources;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.graphics.Canvas;

import android.graphics.Paint;

import android.util.AttributeSet;

import android.view.MotionEvent;

import android.view.View;

import android.widget.Toast;

//Extends View class

public class MyDraw extends View {

//Make paint & bitmap

private Paint mPaint;

private Bitmap Pre, Post;

//It is used to check user click

boolean[][] check = new boolean[8][5];

public void init(){

//Initialize paint

mPaint=new Paint();

//get resource route

Resources res = getResources();

//Initialize bitmap

Pre=BitmapFactory.decodeResource(res,R.drawable.bubble);

Post=BitmapFactory.decodeResource(res,R.drawable.bbok);

}

//Default Constructor

public MyDraw(Context c){

super(c);

//Initialize this view class

init();

}

//Default Constructor

public MyDraw(Context c, AttributeSet a){

super(c,a);

//Initialize this view class

init();

}

protected void onDraw(Canvas canvas){

int i,j;

int temp1=0, temp2=0;

//set bitmap

Bitmap sm = Bitmap.createScaledBitmap(Pre,220,210,false);

Bitmap sm2 = Bitmap.createScaledBitmap(Post,220,210,false);

canvas.save(); //It looks like push

//Draw View class

for(i=0; i<8; i++){

for(j=0; j<5; j++){

if(check[i][j]==false)

canvas.drawBitmap(sm,temp1,temp2,mPaint);

else

canvas.drawBitmap(sm2,temp1,temp2,mPaint);

temp1 = temp1 + 220;

}

temp1 = 0;

temp2 = temp2 + 210;

}

//It looks like pop

canvas.restore();

}

public boolean onTouchEvent(MotionEvent event){

int i,j;

//If user click bubble, then change to bbok

if(event.getAction()==MotionEvent.ACTION\_UP){

float x = event.getX();

float y = event.getY();

for(i=0; i<8; i++){

if(y<=210\*(i+1) && y>210\*i){

for(j=0; j<5; j++){

if(x<=220\*(j+1) && x>220\*j){

check[i][j]=true;

invalidate();

}

}

}

}

}

return true;

}

}

**activity\_main.xml**

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

xmlns:tools=*"http://schemas.android.com/tools"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:orientation=*"vertical"* >

<!-- Declear View Class -->

<com.example.hw5a.MyDraw

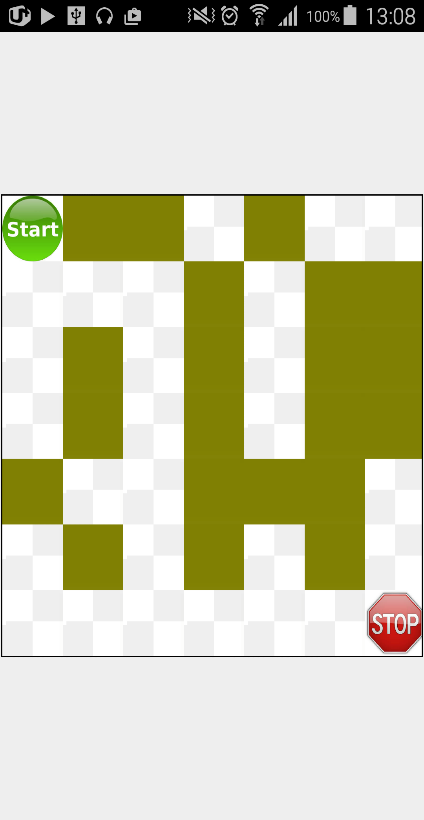
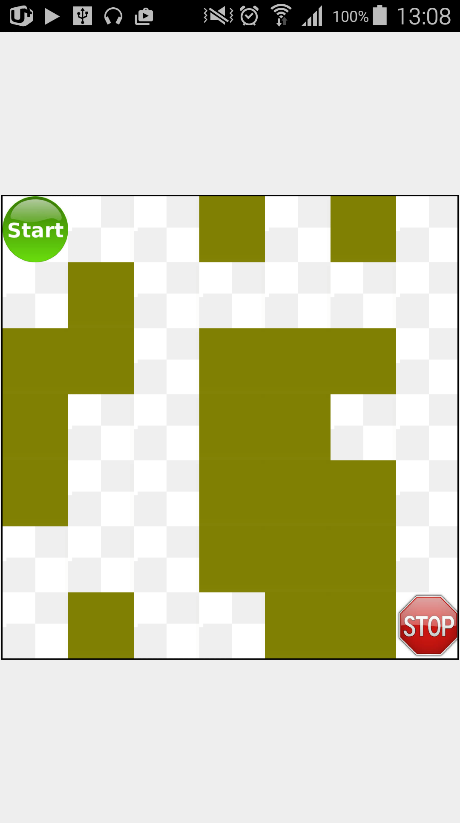
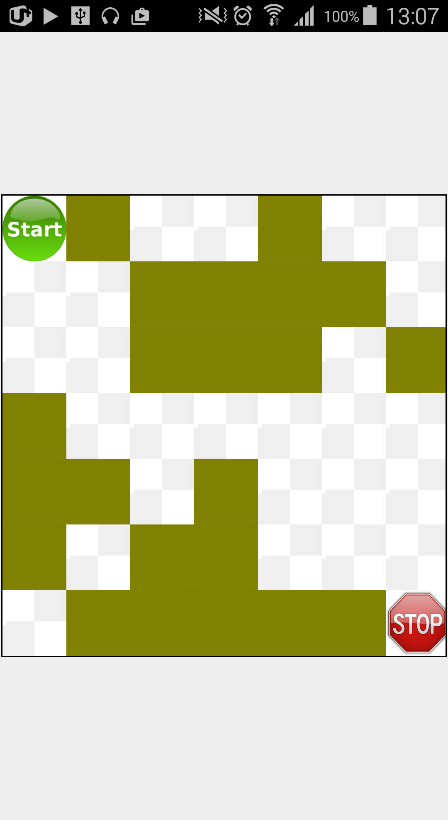
android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"* />

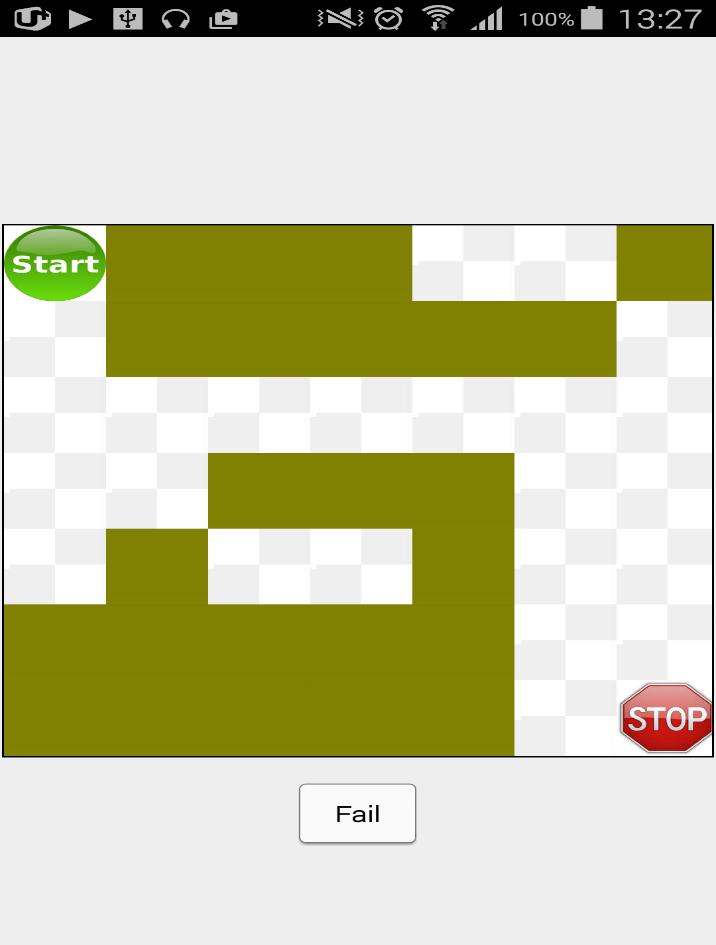
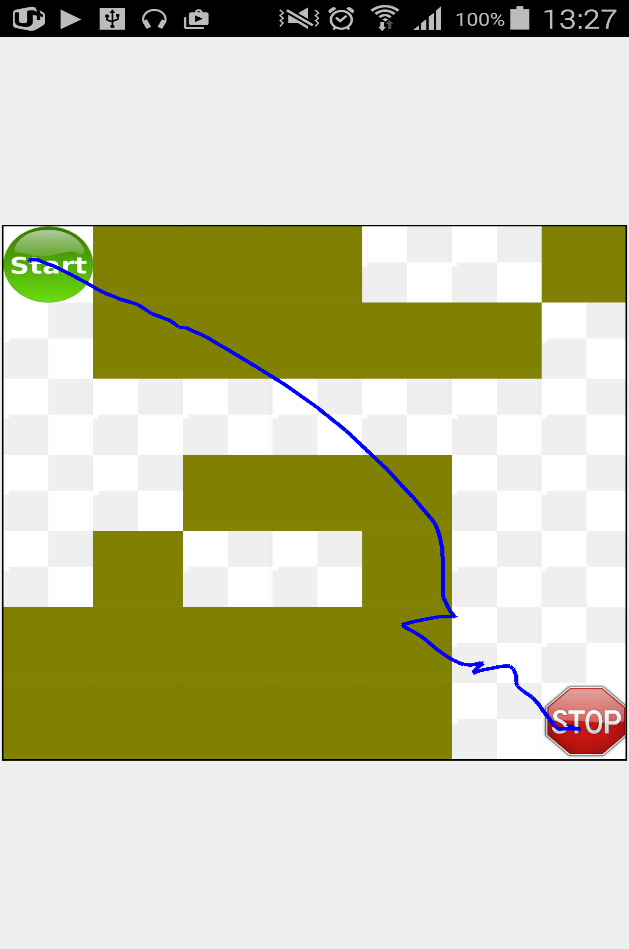
</LinearLayout>

**HW. #5-B: Mazes (60Pt)**

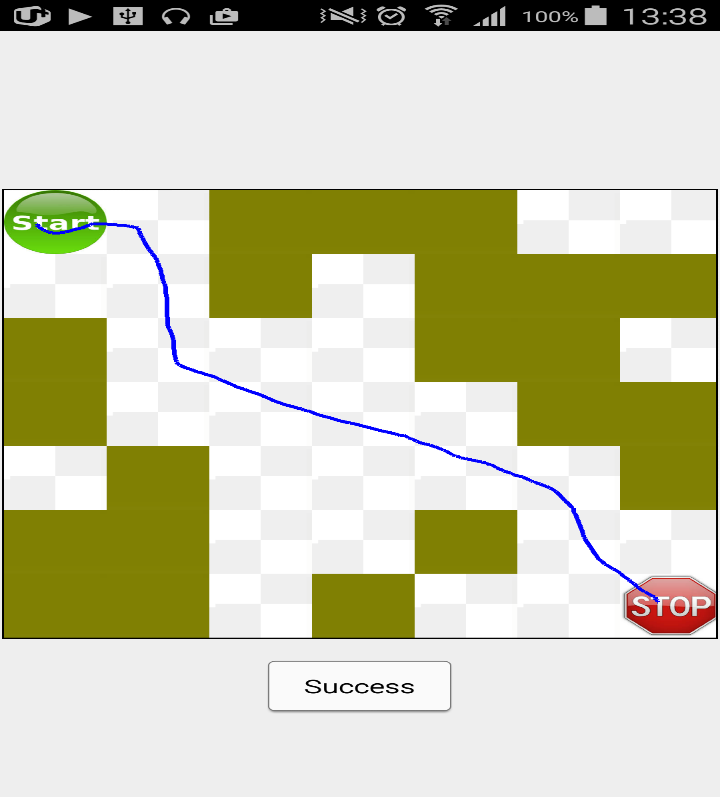
**ScreenShot 2**



**<First Screen (Random generation)>**



**<Fail Screen>**



**<Success Screen>**

**Code 2.**

**MainActivity**

**package** com.example.mymaze;

**import** java.util.ArrayList;

**import** android.annotation.SuppressLint;

**import** android.app.ActionBar.LayoutParams;

**import** android.app.Activity;

**import** android.content.Context;

**import** android.content.Intent;

**import** android.graphics.Bitmap;

**import** android.graphics.Canvas;

**import** android.graphics.Color;

**import** android.graphics.Paint;

**import** android.graphics.PixelFormat;

**import** android.graphics.Rect;

**import** android.graphics.drawable.BitmapDrawable;

**import** android.os.Bundle;

**import** android.util.AttributeSet;

**import** android.view.Display;

**import** android.view.MotionEvent;

**import** android.view.SurfaceHolder;

**import** android.view.SurfaceView;

**import** android.view.ViewManager;

**import** android.view.Window;

**import** android.view.WindowManager;

**import** android.widget.ImageView;

**import** android.widget.TableLayout;

**import** android.widget.TableRow;

**import** android.widget.Toast;

**public** **class** MainActivity **extends** Activity {

**boolean** start, end, collision;

LineSurface ds;

**int** i, j;

Bitmap Wall, Start, End, Road; //Set bitmap (Wall, Start/End, Road)

**int**[][] maze; //Maze

ImageView startImage, endImage; //Set image through bitmap

ArrayList<ImageView> blockImage = **new** ArrayList<ImageView>();

**int**[] imageXY = **new** **int**[2];

TableLayout table;

@SuppressLint("NewApi")

**private** **void** addSurfaceView() {

// add additional layer(SurfaceView) through addContentView

ds = **new** LineSurface(**this**);

ds.setZOrderOnTop(**true**);

ds.getHolder().setFormat(PixelFormat.***TRANSPARENT***);

addContentView(ds, **new** LayoutParams(LayoutParams.***MATCH\_PARENT***, LayoutParams.***MATCH\_PARENT***));

}

**private** **void** removeSurfaceView() {//Delete additional layer

((ViewManager) ds.getParent()).removeView(ds);

}

@Override

**protected** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

//InXtialize all condition

collision = start = end = **false**;

setContentView(R.layout.***activity\_main***); //Set maze background

addSurfaceView(); //Add SurfaceView to activity\_main(table layout)

// Set maze X,Y line

Intent intent = getIntent();

**if** (intent.getIntExtra("x", 0) < 5)

i = 7;

**else**

i = intent.getIntExtra("x", 7);

**if** (intent.getIntExtra("y", 0) < 5)

j = 7;

**else**

j = intent.getIntExtra("y", 7);

// Make maze

maze = **new** **int**[i][j];

//Check posiblity of maze, if not satisfy maze, recreate maze

MazeSolver ms = **new** MazeSolver(i, j, maze);

// Initial

**do** {

**for** (**int** nX = 0; nX < i; nX++) {

**for** (**int** nY = 0; nY < j; nY++) {

**if** ((nX == 0 && nY == 0) || (nX == i - 1 && nY == j - 1)) {

maze[nX][nY] = 0;

} **else** {

maze[nX][nY] = ((**int**) (Math.*random*() \* 10) % 2); //0 혹은 1로 배열을 채움.

}

}

}

} **while** (!ms.find(0, 0));//If it satisfy MazeSolver, escape while loop

// Set maze location to center

Display display = ((WindowManager) getSystemService(***WINDOW\_SERVICE***))

.getDefaultDisplay();

**int** tableSize = display.~~getWidth~~();

**int** cellWidth = tableSize / i;

// regulate bitmap

Wall = Bitmap.*createScaledBitmap*(((BitmapDrawable) getResources()

.getDrawable(R.drawable.***block***)).getBitmap(), cellWidth,

cellWidth, **false**);

Start = Bitmap.*createScaledBitmap*(((BitmapDrawable) getResources()

.getDrawable(R.drawable.***start***)).getBitmap(), cellWidth,

cellWidth, **false**);

End = Bitmap.*createScaledBitmap*(((BitmapDrawable) getResources()

.getDrawable(R.drawable.***stop***)).getBitmap(), cellWidth,

cellWidth, **false**);

Road = Bitmap.*createScaledBitmap*(((BitmapDrawable) getResources()

.getDrawable(R.drawable.***road***)).getBitmap(), cellWidth,

cellWidth, **false**);

//get TableLayout

table = (TableLayout) findViewById(R.id.***mTable***);

// Draw maze to table using drawable image

**for** (**int** nY = 0; nY < j; nY++) {

TableRow r = **new** TableRow(getApplicationContext());

**for** (**int** nX = 0; nX < i; nX++) {

ImageView temp = **new** ImageView(getApplicationContext());

**if** ((nY == 0 && nX == 0)) {// Start locate left above

temp.setImageBitmap(Start);

startImage = temp;

} **else** **if** ((nX == i - 1 && nY == j - 1)) { //End locate right bottom

temp.setImageBitmap(End);

endImage = temp;

} **else** {

**if** (maze[nX][nY] == 1) { //If maze[nX][nY] == 1 --> It represents wall

temp.setImageBitmap(Wall);

blockImage.add(temp);

}**else**

temp.setImageBitmap(Road);//maze[nX][nY] == 0 --> Road

}

r.addView(temp);

}

table.addView(r);

}

}

//SurfaceView

**public** **class** LineSurface **extends** SurfaceView **implements**

SurfaceHolder.Callback {

Canvas cacheCanvas;

Paint paint;

Context context;

Bitmap backBuffer;

**int** width, height, clientHeight;

SurfaceHolder mHolder;

**boolean** notify = **true**;

**public** LineSurface(Context context) {

**super**(context);

**this**.context = context;

inXt();

}

**public** LineSurface(Context context, AttributeSet attrs) {

**super**(context, attrs);

**this**.context = context;

inXt();

}

**private** **void** inXt() {

mHolder = getHolder();

mHolder.addCallback(**this**);

}

**public** **void** surfaceChanged(SurfaceHolder holder, **int** format, **int** width, **int** height) {}

**public** **void** surfaceCreated(SurfaceHolder holder) {

width = getWidth();

height = getHeight();

cacheCanvas = **new** Canvas();

backBuffer = Bitmap.*createBitmap*(width, height,Bitmap.Config.***ARGB\_8888***);

cacheCanvas.setBitmap(backBuffer);

cacheCanvas.drawColor(Color.***TRANSPARENT***); //Because of bottom of table layout

paint = **new** Paint(); //Set paint

paint.setColor(Color.***BLUE***);

paint.setStrokeWidth(7);

draw();

}

**public** **void** surfaceDestroyed(SurfaceHolder holder) {}

**int** lastX, lastY, currX, currY;

**boolean** isDeleting;

@SuppressLint("NewApi")

@Override

**public** **boolean** onTouchEvent(MotionEvent event) {

**super**.onTouchEvent(event);

**int** action = event.getAction(); //get current event

**switch** (action & MotionEvent.***ACTION\_MASK***) {

**case** MotionEvent.***ACTION\_DOWN***: //If click, store current position

lastX = (**int**) event.getX();

lastY = (**int**) event.getY();

**break**;

**case** MotionEvent.***ACTION\_MOVE***: //If move, draw line

**if** (isDeleting)

**break**;

currX = (**int**) event.getX();

currY = (**int**) event.getY();

//Only draw in maze

**if** (currX >= table.getX()

&& currX <= table.getX() + table.getWidth()

&& currY >= table.getY()

&& currY <= table.getY() + table.getHeight()) {

cacheCanvas.drawLine(lastX, lastY, currX, currY, paint);

lastX = currX;

lastY = currY;

// If pass the start image, then convert boolean value.

startImage.getLocationInWindow(imageXY);

**if** (imageXY[0] <= currX

&& imageXY[0] + startImage.getWidth() >= currX

&& imageXY[1] <= currY

&& imageXY[1] + startImage.getHeight() >= currY) {

**int** z = imageXY[0]+startImage.getWidth();

**int** x = imageXY[1]+startImage.getHeight();

start = **true**;

}

// It same above, end boolean value convert

endImage.getLocationInWindow(imageXY);

**if** (imageXY[0] <= currX

&& imageXY[0] + endImage.getWidth() >= currX

&& imageXY[1] <= currY

&& imageXY[1] + endImage.getHeight() >= currY) {

end = **true**;

}

// Process of collision to wall. If occur collision to wall, convert boolean value.

**for** (**int** i = 0; i < blockImage.size(); i++) {

ImageView b = blockImage.get(i);

b.getLocationInWindow(imageXY);

**if** (imageXY[0] <= currX

&& imageXY[0] + b.getWidth() >= currX

&& imageXY[1] <= currY

&& imageXY[1] + b.getHeight() >= currY) {

collision = **true**;

}

}

draw(); // Draw SurfaceView

} **else** {

// If user over the maze, program set action\_up

MotionEvent up\_event = MotionEvent.*obtain*(0, 0, MotionEvent.***ACTION\_UP***, 0, 0, 0);

dispatchTouchEvent(up\_event);

}

**break**;

**case** MotionEvent.***ACTION\_UP***: //If action up

**if** (notify) {

**if** (start && end && !collision) { //If satisfy all condition safe, show toast "Success"

Toast.*makeText*(getApplicationContext(), "Success", Toast.***LENGTH\_SHORT***).show();

notify = **false**;

isDeleting = **true**;

} **else** { //If not satisfy, show toast "Fail"

Toast.*makeText*(getApplicationContext(), "Fail", Toast.***LENGTH\_SHORT***).show();

cleanUp();

}

}

**break**;

}

**return** **true**;

}

**private** **void** cleanUp() {

//If user doesn't satisfy all condition, clean the maze

collision = start = end = **false**; //InXtialize

removeSurfaceView();

addSurfaceView();

}

**protected** **void** draw() { //Draw re surfaceview

**if** (clientHeight == 0) {

clientHeight = getClientHeight();

height = clientHeight;

cacheCanvas.drawColor(Color.***TRANSPARENT***);

}

Canvas canvas = **null**;

**try** {

canvas = mHolder.lockCanvas(**null**);

// Draw bitmap in back buffer

canvas.drawBitmap(backBuffer, 0, 0, paint);

} **catch** (Exception ex) {

ex.printStackTrace();

} **finally** {

**if** (mHolder != **null**)

mHolder.unlockCanvasAndPost(canvas);

}

}

// get client position

**private** **int** getClientHeight() {

Rect rect = **new** Rect();

Window window = ((Activity) context).getWindow();

window.getDecorView().getWindowVisibleDisplayFrame(rect);

**int** statusBarHeight = rect.top;

**int** contentViewTop = window.findViewById(Window.***ID\_ANDROID\_CONTENT***)

.getTop();

**int** titleBarHeight = contentViewTop - statusBarHeight;

**return** statusBarHeight + titleBarHeight;

}

}

// class LineSurface

//get solution of exit maze. (recursive)

**public** **class** MazeSolver {

**public** **int**[][] maze;

**public** **int** width, height;

**int** row;

**int** col;

**boolean** a,b,c,d;

**public** MazeSolver(**int** width, **int** height, **int**[][] maze){ //get size

**this**.width = width;

**this**.height = height;

**this**.maze = maze.clone();

}

**public** **boolean** find(**int** row, **int** col) { //Find the path

**this**.maze[row][col] = 3;

**if** (row == width-1 && col == height-1) {

**return** **true**;

}**else** **if**(row > width || col > height)

**return** **false**;

**if** (row + 1 < width && maze[row + 1][col] == 0) {

**return** a = find(row + 1, col);

}

**if** (a == **false** && col +1 < height && maze[row][col + 1] == 0) {

**return** b = find(row, col + 1);

}

**if** (a == b== **false** && row > 1 && maze[row - 1][col] == 0) {

**return** c = find(row -1, col);

}

**if** (a == b == c == **false** &&col > 1 && maze[row][col - 1] == 0) {

**return** d = find(row, col - 1);

}

**return** **false**;

}

}

}// MazeSolver

**activity\_main.xml**

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

xmlns:tools=*"http://schemas.android.com/tools"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:gravity=*"center"*

tools:context=*"${relativePackage}.${activityClass}"* >

<TableLayout

android:id=*"@+id/mTable"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_margin=*"1dp"*

android:background=*"@drawable/maze"*

android:padding=*"1dp"* >

</TableLayout>

</LinearLayout>

**maze.xml**

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<shape

xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:shape= *"rectangle"* >

<solid android:color=*"#fff"*/>

<stroke android:width=*"1dp"* android:color=*"#000"*/>

</shape>

**AndroidManifest.xml**

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<manifest xmlns:android=*"http://schemas.android.com/apk/res/android"*

package=*"com.example.mymaze"*

android:versionCode=*"1"*

android:versionName=*"1.0"* >

<uses-sdk

android:minSdkVersion=*"11"*

android:targetSdkVersion=*"21"* />

<application

android:allowBackup=*"true"*

android:icon=*"@drawable/ic\_launcher"*

android:label=*"@string/app\_name"*

android:theme=*"@android:style/Theme.Holo.NoActionBar"*>

<activity

android:name=*".MainActivity"*

android:label=*"@string/app\_name"* >

<intent-filter>

<action android:name=*"android.intent.action.MAIN"* />

<category android:name=*"android.intent.category.LAUNCHER"* />

</intent-filter>

</activity>

</application>

</manifest>