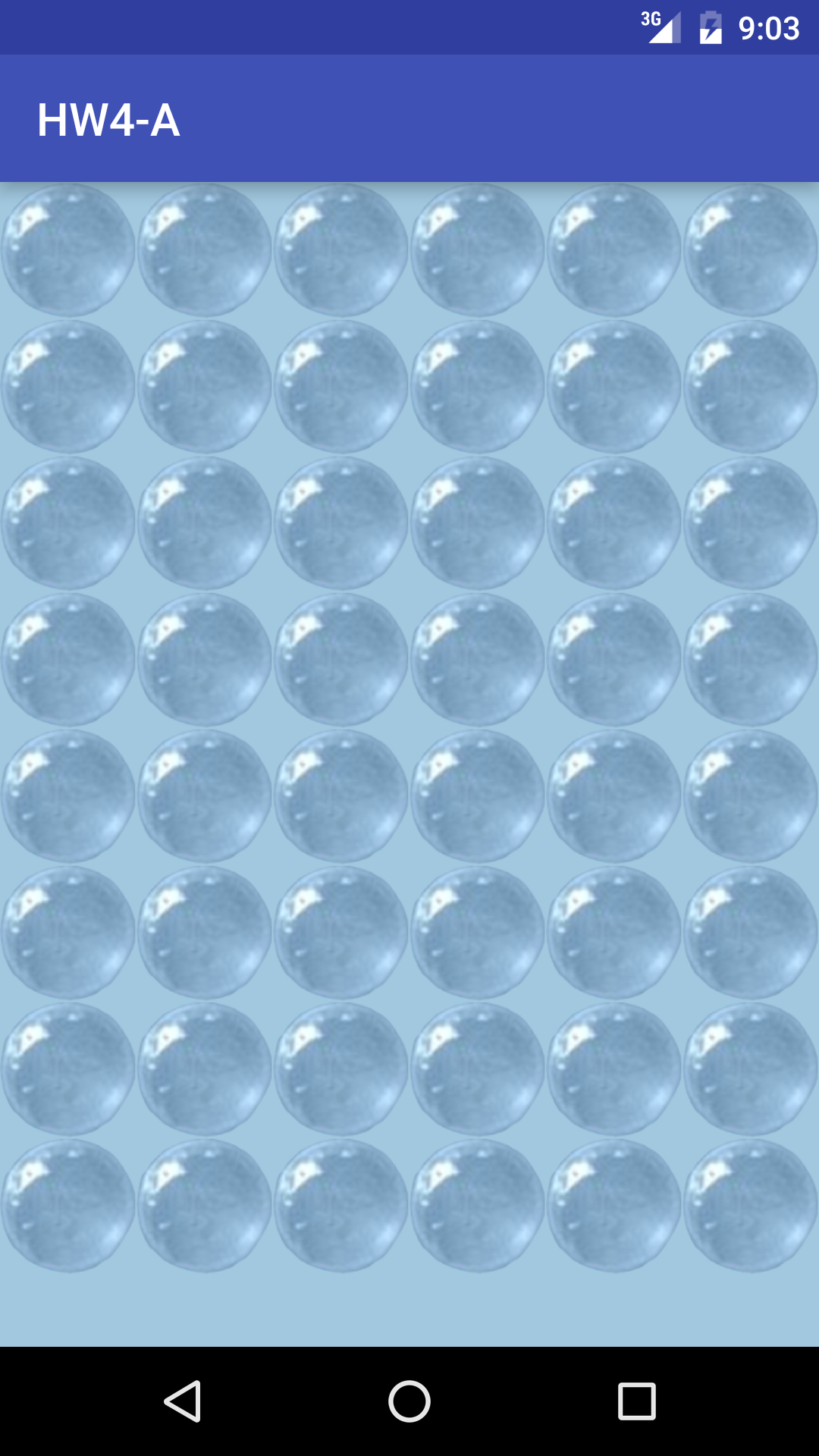
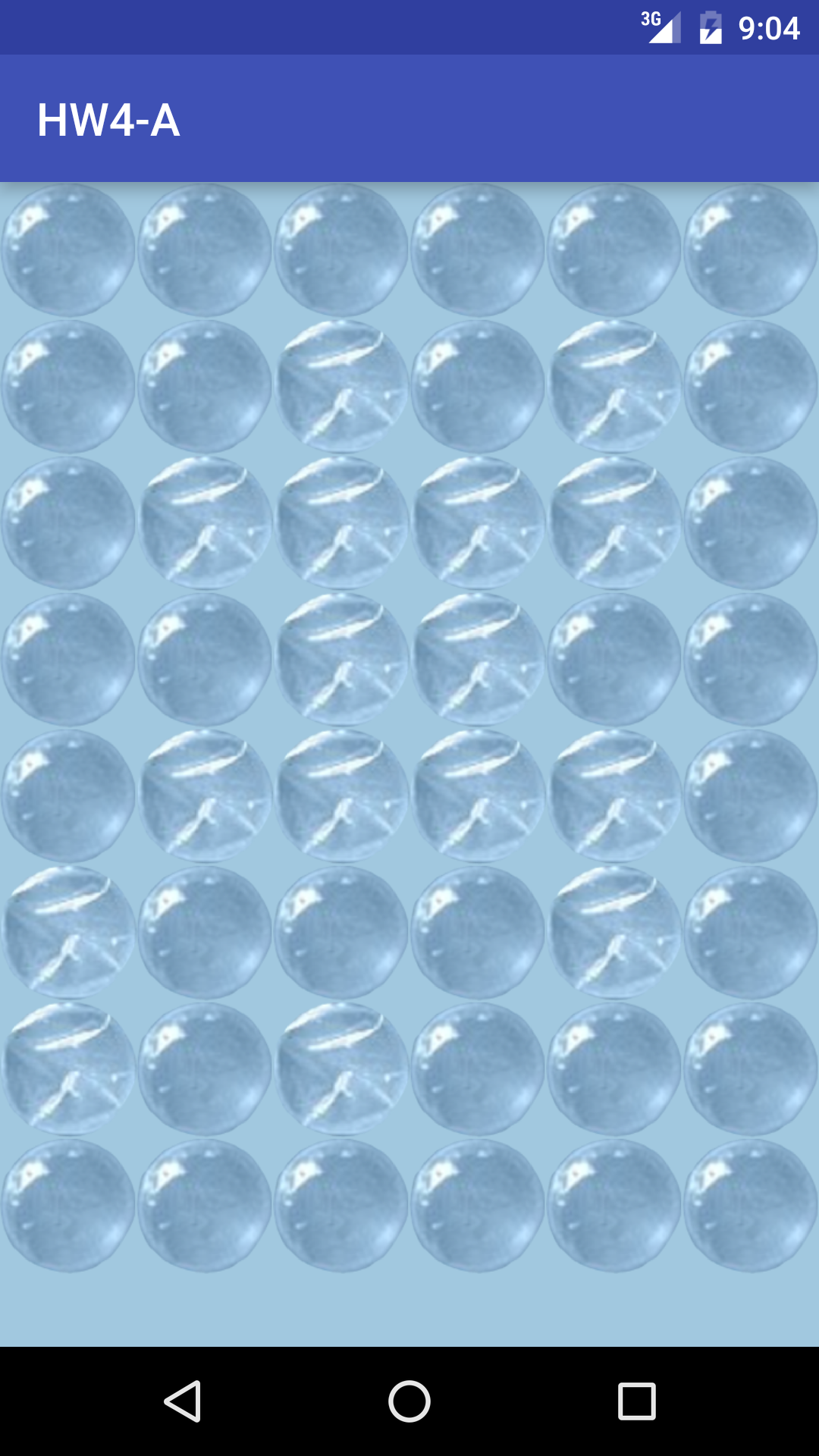
HW4-A

**package** com.example.hansol.hw4\_a;  
  
**import** android.content.Context;  
**import** android.content.res.Configuration;  
**import** android.content.res.Resources;  
**import** android.graphics.Bitmap;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.util.DisplayMetrics;  
**import** android.util.Log;  
**import** android.view.MotionEvent;  
**import** android.os.Bundle;  
**import** android.graphics.BitmapFactory;  
**import** android.graphics.Canvas;  
**import** android.graphics.Paint;  
**import** android.graphics.Rect;  
**import** android.view.View;  
  
**public class** MainActivity **extends** AppCompatActivity {  
 **private** Bitmap **mbok**;  
 **private** Bitmap **mbokbok**;  
 **private** Paint **mPaint**;  
 **private boolean**[][] **check**= **new boolean**[8][6];  
 **private int**[][] **mode** = **new int**[8][6];  
 **private float swidth**;  
 **int windowHeight**;  
 **int windowWidth**;  
 **int imgwidth**;  
 **int imgheight**;  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 MyView vw = **new** MyView(**this**);  
 setContentView(vw);  
 }  
 **protected class** MyView **extends** View {  
  
 **public** MyView(Context context) {  
 **super**(context);  
 init();  
 }  
 **public void** init() {  
 **for**(**int** i = 0 ; i < 8 ; i++){  
 **for**(**int** j = 0 ; j < 6 ; j++){  
 **check**[i][j] = **true**;  
 **mode**[i][j] = 1;  
 }  
 }  
 **mPaint** = **new** Paint();  
 *//mbok is before it pop, and mbokbok is after it poped.* Resources res = getResources();  
 **mbok** = BitmapFactory.*decodeResource*(res, R.drawable.***bok***);  
 **mbokbok** = BitmapFactory.*decodeResource*(res, R.drawable.***bokbok***);  
  
 **imgwidth** = **mbok**.getWidth();  
 **imgheight** = **mbok**.getHeight();  
  
 DisplayMetrics dm = getApplicationContext().getResources().getDisplayMetrics();  
 **int** statusBarheight = getStatusBarHeight();  
  
 *//get right windowsize for windowWidth/6 and show 8 size and 6 size for whole screen* **windowHeight**= dm.**heightPixels** - statusBarheight;  
 **windowWidth** = dm.**widthPixels**;  
 **swidth** = (**float**)**windowWidth**/6;  
  
 Log.*i*(**"TAG11"**,**windowWidth** +**" "**+**windowHeight** );  
 }  
 **public boolean** onTouchEvent(MotionEvent event) {  
 **int** endcheck = 0 ;  
 **if** (event.getAction() == MotionEvent.***ACTION\_UP***) {  
 *// get the coordinates* **float** x = event.getX();  
 **float** y = event.getY();  
 *//change state for already poped, if it;s poped already, finish the loop* **for**(**int** i =0 ; i< 8 ; i++){  
 **for**(**int** j = 0 ; j < 6 ; j++){  
 **if** ((**int**)((**swidth**\*j + **swidth**/2)-x)\*(**int**)((**swidth**\*j + **swidth**/2)-x) + (**int**)((**swidth**\*i + **swidth**/2)-y)\* (**int**)((**swidth**\*i + **swidth**/2)-y) <(**int**)((**swidth**/2) \* (**swidth**/2)) ) {  
 **if**(**check**[i][j] == **true**) {  
 **check**[i][j] = **false**;  
 **int** r = (**int**) (Math.*random*() \* 4);  
 **mode**[i][j] = r;  
 endcheck = 1;  
 invalidate();  
 **break**;  
 }  
 }  
 }  
 **if**(endcheck == 1){  
 endcheck = 0;  
 **break**;  
 }  
 }  
  
 }  
 *// indicates that the event was handled* **return true**;  
 } *// end of onTouchEvent* **public void** onDraw(Canvas canvas) {  
 canvas.drawColor(0xffA1C8DF);  
 **for**(**int** i = 0 ;i < 8 ; i++){  
 **for**(**int** j = 0 ; j < 6 ; j++){  
 **if**(**check**[i][j])  
 canvas.drawBitmap(**mbok**, **new** Rect(0, 0, **imgwidth**, **imgheight**), **new** Rect((**int**)**swidth**\*j,(**int**)**swidth**\*i ,(**int**)(**swidth**\*j+**swidth**), (**int**)(**swidth**\*i +**swidth**)), **mPaint**);  
 **else** {  
 canvas.drawBitmap(**mbokbok**, **new** Rect(0, 0, **imgwidth**, **imgheight**), **new** Rect((**int**) **swidth** \* j, (**int**) **swidth** \* i, (**int**) (**swidth** \* j + **swidth**), (**int**) (**swidth** \* i + **swidth**)), **mPaint**);  
 }  
 }  
 }  
 }  
 }  
 **public int** getStatusBarHeight(){  
 **int** statusHeight = 0;  
 **int** screenSizeType = (getApplicationContext().getResources().getConfiguration().**screenLayout** &  
 Configuration.***SCREENLAYOUT\_SIZE\_MASK***);  
 **if**(screenSizeType != Configuration.***SCREENLAYOUT\_SIZE\_XLARGE***) {  
 **int** resourceId = getApplicationContext().getResources().getIdentifier(**"status\_bar\_height"**, **"dimen"**, **"android"**);  
 **if** (resourceId > 0) {  
 statusHeight = getApplicationContext().getResources().getDimensionPixelSize(resourceId);  
 }  
 }  
 **return** statusHeight;  
 }  
}

HW4-B

**package** com.example.hansol.hw4\_b;  
  
**import** android.content.Context;  
**import** android.content.res.Resources;  
**import** android.graphics.Bitmap;  
**import** android.graphics.BitmapFactory;  
**import** android.os.Bundle;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.util.AttributeSet;  
**import** android.util.Log;  
**import** android.view.MotionEvent;  
**import** android.view.SurfaceHolder;  
**import** android.view.SurfaceView;  
**import** android.graphics.Canvas;  
**import** android.graphics.Color;  
**import** android.graphics.Paint;  
**import** android.graphics.Rect;  
**import** android.widget.Toast;  
  
**public class** MainActivity **extends** AppCompatActivity {  
  
 **private** Paint **paint** = **new** Paint(Paint.***ANTI\_ALIAS\_FLAG***);  
 *//declare variables for mazes* **int**[][] **maze** = **new int**[15][15];  
 **boolean**[][] **wasHere** = **new boolean**[15][15];  
 **int startX** = 1, **startY** = 1;  
 **int endX** = 13, **endY** = 13;  
 **int cix**, **ciy**, **wallsize**;  
 **boolean objectmove** = **false**;  
 **private** Bitmap **wall**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 MySurfaceView mySurfaceView = **new** MySurfaceView(**this**);  
 setContentView(mySurfaceView);  
 }  
  
 *//use a variable for get ramdon and figure out wall or road.  
 //use 'MakingMaze' get ture, then maze will done for mapping random* **public void** MakingMaze() {  
 **int** a;  
 *//if it is start point or end point, it will be marked by road.  
 //declare all side part for all wall  
 //if a=0 for road, if a=1 for wall.* **for** (**int** i = 0; i < 15; i++) {  
 **for** (**int** j = 0; j < 15; j++) {  
 a = (**int**) (Math.*random*() \* 2);  
 **if** ((i == 1 && j == 1) || (i == 13 && j == 13)) {  
 **maze**[i][j] = 1;  
 } **else** {  
 **if** (i == 0 || j == 0 || i == 14 || j == 14) {  
 **maze**[i][j] = 2;  
 } **else** {  
 **if** (a == 0) {  
 **maze**[i][j] = 1;  
 } **else** {  
 **maze**[i][j] = 2;  
 }  
 }  
 }  
 }  
 }  
 }  
  
 *// create Maze 1 for road 2 for wall  
 // sets boolean Arrays to default values* **public boolean** solveMaze() {  
 MakingMaze();  
 **for** (**int** row = 0; row < 15; row++)  
 **for** (**int** col = 0; col < 15; col++) {  
 **wasHere**[row][col] = **false**;  
 }  
 **boolean** b = checkendpoint(**startX**, **startY**);  
  
 **return** b;  
 }  
  
 **public boolean** checkendpoint(**int** x, **int** y) {  
 **if** (x == **endX** && y == **endY**)  
 **return true**; *// If you reached the end* **if** (**maze**[x][y] == 2 || **wasHere**[x][y])  
 **return false**;  
  
 *//check all side of roal and figure out start point and end point.* **wasHere**[x][y] = **true**;  
 **if** (x != 0)  
 **if** (checkendpoint(x - 1, y)) {  
 **return true**;  
 }  
 **if** (x != 15 - 1)  
 **if** (checkendpoint(x + 1, y)) {  
 **return true**;  
 }  
 **if** (y != 0)  
 **if** (checkendpoint(x, y - 1)) {  
 **return true**;  
 }  
 **if** (y != 15 - 1)  
 **if** (checkendpoint(x, y + 1)) {  
 **return true**;  
 }  
 **return false**;  
 }  
  
  
 **public class** MySurfaceView **extends** SurfaceView **implements** SurfaceHolder.Callback {  
 SurfaceHolder **mHolder**;  
 **int side**, **side\_y**;  
 Canvas **cacheCanvas**;  
 Bitmap **backBuffer**;  
 **int width**, **height**;  
 Paint **paint**;  
 Context **context**;  
 **int lastX**, **lastY**, **currX**, **currY**;  
 **boolean isDeleting**;  
  
 **public** MySurfaceView(Context context) {  
 **super**(context);  
 **this**.**context** = context;  
 init();  
 }  
 **public** MySurfaceView(Context context, AttributeSet attrs) {  
 **super**(context, attrs);  
 **this**.**context** = context;  
 init();  
 }  
 **void** init() {  
 **mHolder** = getHolder();  
 **mHolder**.addCallback(**this**);  
 **boolean** makemaze;  
 **do** {  
 makemaze = solveMaze();  
 } **while** (!makemaze);  
 }  
 **void** remakemaze(){  
 **boolean** makemaze;  
 **do** {  
 makemaze = solveMaze();  
 } **while** (!makemaze);  
 }  
 @Override  
 **public boolean** onTouchEvent(MotionEvent event) {  
 **super**.onTouchEvent(event);  
 **int** action = event.getAction();  
  
 **switch** (action & MotionEvent.***ACTION\_MASK***) {  
 **case** MotionEvent.***ACTION\_DOWN***:  
  
 **lastX** = (**int**) event.getX();  
 **lastY** = (**int**) event.getY();  
 **if** ((**cix** - **lastX**) \* (**cix** - **lastX**) + (**ciy** - **lastY**) \* (**ciy** - **lastY**) < (**int**) (**side** / 3) \* (**side** / 3)) {  
 **objectmove** = **true**;  
 **cix** = **lastX**;  
 **ciy** = **lastY**;  
 }  
 **break**;  
  
 *//if touch sensor for twice, make sure it doesn't work.* **case** MotionEvent.***ACTION\_MOVE***:  
 **if** (**isDeleting**)  
 **break**;  
 **if** (**objectmove**) {  
 **currX** = (**int**) event.getX();  
 **currY** = (**int**) event.getY();  
 **int** x = **currX** / **side**;  
 **int** y = **currY** / **side\_y**;  
 **if** (y == 13 && x == 13) {  
 **objectmove** = **false**;  
 **cix** = **side** + (**int**) **side** / 2;  
 **ciy** = **side\_y** + (**int**) **side\_y** / 2;  
 Toast.*makeText*(getApplicationContext(), **" Complete! "**, Toast.***LENGTH\_SHORT***).show();  
 remakemaze();  
 } **else if** (**maze**[y][x] == 2) {  
 *//Reset plaer when it touch wall* **objectmove** = **false**;  
 **cix** = **side** + (**int**) **side** / 2;  
 **ciy** = **side\_y** + (**int**) **side\_y** / 2;  
 Toast.*makeText*(getApplicationContext(), **"Don't touch wall!\nReset the game."**, Toast.***LENGTH\_SHORT***).show();  
 }  
 *//cacheCanvas.drawLine(lastX, lastY, currX, currY, paint);* **else** {  
 **cix** = **currX**;  
 **ciy** = **currY**;  
 }  
 **break**;  
 }  
 **case** MotionEvent.***ACTION\_UP***:  
 **if** (**isDeleting**)  
 **isDeleting** = **false**;  
 **objectmove** = **false**;  
 **break**;  
 **case** MotionEvent.***ACTION\_POINTER\_DOWN***:  
 **cacheCanvas**.drawColor(Color.***WHITE***);  
 **isDeleting** = **true**;  
 **break**;  
 **case** MotionEvent.***ACTION\_POINTER\_UP***:  
 **break**;  
 }  
 draw();  
 **return true**;  
 }  
  
  
 *// override the methods for the UI SurfaceHolder. call back* @Override  
 **public void** surfaceCreated(SurfaceHolder holder) {  
  
 **width** = getWidth();  
 **height** = getHeight();  
 Log.*i*(**"TAG"**, **"draw: "** + **height**);  
 **side** = (**int**) (**width** / 15);  
 **side\_y** = (**int**) (**height** / 15);  
 **cix** = **side** + (**int**) **side** / 2;  
 **ciy** = **side\_y** + (**int**) **side\_y** / 2;  
  
 Log.*i*(**"TAG"**, **side** + **" "** + **width** + **" "** + **cix**);  
 **cacheCanvas** = **new** Canvas();  
 **backBuffer** = Bitmap.*createBitmap*(**width**, **height**, Bitmap.Config.***ARGB\_8888***);  
 **cacheCanvas**.setBitmap(**backBuffer**);  
 **cacheCanvas**.drawColor(Color.***WHITE***);  
  
 **paint** = **new** Paint();  
 **paint**.setColor(Color.***BLUE***);  
 **paint**.setStrokeWidth(10);  
 **paint**.setStrokeCap(Paint.Cap.***ROUND***);  
 **paint**.setStrokeJoin(Paint.Join.***ROUND***);  
  
 Resources res = getResources();  
 **wall** = BitmapFactory.*decodeResource*(res, R.drawable.***wall***);  
 **wallsize** = **wall**.getWidth();  
 draw();  
 }  
  
 @Override  
 **public void** surfaceChanged(SurfaceHolder holder, **int** format, **int** width, **int** height) {}  
  
 @Override  
 **public void** surfaceDestroyed(SurfaceHolder holder) {}  
  
 *//draw start point and end point for black color  
 //draw road for white color  
 //draw wall for image in drawable* **protected void** draw() {  
 **backBuffer** = Bitmap.*createBitmap*(**width**, **height**, Bitmap.Config.***ARGB\_8888***);  
 **cacheCanvas**.setBitmap(**backBuffer**);  
 **cacheCanvas**.drawColor(Color.***WHITE***);  
 **for** (**int** i = 0; i < 15; i++) {  
 **for** (**int** j = 0; j < 15; j++) {  
 **if** ((i == 1 && j == 1) || (i == 13 && j == 13)) {  
 **paint**.setColor(Color.***BLACK***);  
 **cacheCanvas**.drawRect(**side** \* j, **side\_y** \* i, **side** \* j + **side**, **side\_y** \* i + **side\_y**, **paint**);  
 } **else if** (**maze**[i][j] == 1) {  
 **paint**.setColor(Color.***WHITE***);  
 **cacheCanvas**.drawRect(**side** \* j, **side\_y** \* i, **side** \* j + **side**, **side\_y** \* i + **side\_y**, **paint**);  
 } **else** {  
 **paint**.setColor(Color.***BLACK***);  
 **cacheCanvas**.drawBitmap(**wall**, **new** Rect(0, 0, **wallsize**, **wallsize**),  
 **new** Rect(**side** \* j, **side\_y** \* i, **side** \* j + **side**, **side\_y** \* i + **side\_y**), **paint**);  
 }  
 }  
 }  
 **paint**.setColor(Color.***RED***);  
  
  
  
 **cacheCanvas**.drawCircle(**cix**, **ciy**, (**int**) **side** / 3, **paint**);  
 Canvas canvas = **null**;  
 **try** {  
 canvas = **mHolder**.lockCanvas(**null**);  
 canvas.drawBitmap(**backBuffer**, 0, 0, **paint**);  
 } **catch** (Exception ex) {  
 ex.printStackTrace();  
 } **finally** {  
 **if** (**mHolder** != **null**) **mHolder**.unlockCanvasAndPost(canvas);  
 }  
 }  
 }  
}

