CTT535 - Phát Triển Phần Mềm cho Thiết Bị Di Động
Báo cáo đồ án thực hành
Đề tài Ứng dụng đồ họa

Số thứ tự nhóm: 30

Tên nhóm: VPK2013

MSSV : 1352035

Họ và tên : Trần Hoàng Vũ

MSSV : 1352034

Họ và tên : Ang Tony Vincent

Email liên hệ : angtonyvincent@gmail.com

Điện thoại liên hệ : 0963 218 224

Nội dung chính

1. Các kĩ thuật đã tìm hiểu

1.1. OpenGL ES

1.1.1. Giới thiệu OpenGL ES

OpenGL ES là viết tắt của Open Graphics Library for Embedded System, tức là thư viện đồ họa mở cho hệ thống nhúng.

Đây là một tiêu chuẩn kỹ thuật đồ họa trong Android dùng để tạo ra các giao diện lập trình cho các ứng dụng đồ họa trong không gian 2 chiều (2D) và 3 chiều (3D).

Các phiên bản OpenGL ES tính đến hiện tại và các phiên bản Android hỗ trợ tương ứng

Phiên bản OpenGL ES	Phiên bản Android hỗ trợ
1.0 & 1.1	Android 1.0 và cao hơn
2.0	Android 2.2 (API 8) và cao hơn
3.0	Android 4.3 (API 18) và cao hơn
4.0	Android 5.0 (API 21) và cao hơn

1.1.2. Cách thiết lập môi trường lập trình

Yêu cầu là phải có IDLE (ở đây là Android Studio), Android NDK và thiết bị sử dụng phải hỗ trợ OpenGL ES.

Bước 1 : Khai báo sử dụng Open GL ở tập tin Manifest

```
<uses-feature android:glEsVersion="0x00020000" android:required="true" />
```

Bước 2 : Tạo một màn hình MainActivity cho đồ họa OpenGL ES

```
public class MainActivity extends Activity {

    private GLSurfaceView mGLView;
```

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Create a GLSurfaceView instance and set it
        // as the ContentView for this Activity.
        mGLView = new MyGLSurfaceView(this);
        setContentView(mGLView);
    }
}
```

Bước 3 : Xây dựng đối tượng MyGLSurfaceView để vẽ

```
class MyGLSurfaceView extends GLSurfaceView {

    private final MyGLRenderer mRenderer;

    public MyGLSurfaceView(Context context){
        super(context);

        // Create an OpenGL ES 2.0 context
        setEGLContextClientVersion(2);

        mRenderer = new MyGLRenderer();

        // Set the Renderer for drawing on the GLSurfaceView
        setRenderer(mRenderer);
    }
}
```

Bước 4 : Xây dựng lớp MyGLRenderer để điều khiển

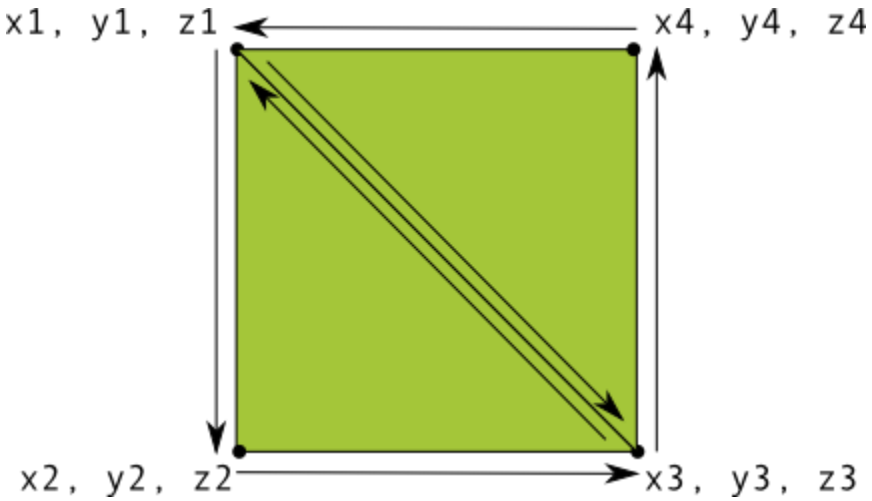
```
public class MyGLRenderer implements GLSurfaceView.Renderer {

    public void onSurfaceCreated(GL10 unused, EGLConfig config) {
        // Set the background frame color
        GLES20.glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    }

    public void onDrawFrame(GL10 unused) {
        // Redraw background color
        GLES20.glClear(GLES20.GL_COLOR_BUFFER_BIT);
    }
```

```
    public void onSurfaceChanged(GL10 unused, int width, int height) {
        GLES20.glViewport(0, 0, width, height);
    }
}
```

Nhận xét : Đây là cách thiết kế khá giống với mô hình MVC (Model – Control – View) quen thuộc.

1.1.3. Định nghĩa hình và vẽ hình (quan trọng)

Giả sử, ta có hình vuông sau



Bước 1 : Định nghĩa hình vuông trong OpenGL ES qua 4 đỉnh của nó

+ Đỉnh bên trái phía trên

+ Đỉnh bên trái phía dưới

+ Đỉnh bên phải phía trên

+ Đỉnh bên phải phía dưới

```
public class Square {

    private FloatBuffer vertexBuffer;
    private ShortBuffer drawListBuffer;

    // number of coordinates per vertex in this array
    static final int COORDS_PER_VERTEX = 3;
    static float squareCoords[] = {
            -0.5f,  0.5f, 0.0f,   // top left
            -0.5f, -0.5f, 0.0f,   // bottom left
```

```
             0.5f, -0.5f, 0.0f,   // bottom right
             0.5f,  0.5f, 0.0f }; // top right

    private short drawOrder[] = { 0, 1, 2, 0, 2, 3 }; // order to draw vertices

    public Square() {
        // initialize vertex byte buffer for shape coordinates
        ByteBuffer bb = ByteBuffer.allocateDirect(
        // (# of coordinate values * 4 bytes per float)
                squareCoords.length * 4);
        bb.order(ByteOrder.nativeOrder());
        vertexBuffer = bb.asFloatBuffer();
        vertexBuffer.put(squareCoords);
        vertexBuffer.position(0);

        // initialize byte buffer for the draw list
        ByteBuffer dlb = ByteBuffer.allocateDirect(
        // (# of coordinate values * 2 bytes per short)
                drawOrder.length * 2);
        dlb.order(ByteOrder.nativeOrder());
        drawListBuffer = dlb.asShortBuffer();
        drawListBuffer.put(drawOrder);
        drawListBuffer.position(0);
    }
}
```

Bước 2 : Khởi tạo hình vuông trong lớp MyGLRenderer

```
public class MyGLRenderer implements GLSurfaceView.Renderer {

    ...
    private Square   mSquare;

    public void onSurfaceCreated(GL10 unused, EGLConfig config) {
        ...
        // initialize a square
        mSquare = new Square();
    }
    ...
}
```

Bước 3 : Viết hàm draw() vẽ hình vuông

```
private int mPositionHandle;
private int mColorHandle;
```

```
private final int vertexCount = squareCoords.length / COORDS_PER_VERTEX;
private final int vertexStride = COORDS_PER_VERTEX * 4; // 4 bytes per vertex

public void draw() {
    // Add program to OpenGL ES environment
    GLES20.glUseProgram(mProgram);

    // get handle to vertex shader's vPosition member
    mPositionHandle = GLES20.glGetAttribLocation(mProgram, "vPosition");

    // Enable a handle to the square vertices
    GLES20.glEnableVertexAttribArray(mPositionHandle);

    // Prepare the square coordinate data
    GLES20.glVertexAttribPointer(mPositionHandle, COORDS_PER_VERTEX,
                                 GLES20.GL_FLOAT, false,
                                 vertexStride, vertexBuffer);

    // get handle to fragment shader's vColor member
    mColorHandle = GLES20.glGetUniformLocation(mProgram, "vColor");

    // Set color for drawing the square
    GLES20.glUniform4fv(mColorHandle, 1, color, 0);

    // Draw the square
    GLES20.glDrawArrays(GLES20.GL_SQUARES, 0, vertexCount);

    // Disable vertex array
    GLES20.glDisableVertexAttribArray(mPositionHandle);
}
```

Bước 4 : Gọi hàm draw() trong onDrawFrame()

```
public void onDrawFrame(GL10 unused) {
    ...

    mSquare.draw();
}
```

Nhận xét : Đã hiển thị thành công hình vuông.

1.1.4. Sử dụng hình chiếu và góc nhìn

Bước 1 : Định nghĩa hình chiếu

```
// mMVPMatrix is an abbreviation for "Model View Projection Matrix"
private final float[] mMVPMatrix = new float[16];
private final float[] mProjectionMatrix = new float[16];
private final float[] mViewMatrix = new float[16];

@Override
public void onSurfaceChanged(GL10 unused, int width, int height) {
    GLES20.glViewport(0, 0, width, height);

    float ratio = (float) width / height;

    // this projection matrix is applied to object coordinates
    // in the onDrawFrame() method
    Matrix.frustumM(mProjectionMatrix, 0, -ratio, ratio, -1, 1, 3, 7);
}
```

Bước 2 : Định nghĩa góc nhìn

```
@Override
public void onDrawFrame(GL10 unused) {
    ...
    // Set the camera position (View matrix)
    Matrix.setLookAtM(mViewMatrix, 0, 0, 0, -3, 0f, 0f, 0f, 0f, 1.0f, 0.0f);

    // Calculate the projection and view transformation
    Matrix.multiplyMM(mMVPMatrix, 0, mProjectionMatrix, 0, mViewMatrix, 0);

    // Draw shape
    mSquare.draw(mMVPMatrix);
}
```

Bước 3 : Sử dụng hình chiếu và góc nhìn

```
public class Square {

    private final String vertexShaderCode =
        // This matrix member variable provides a hook to manipulate
        // the coordinates of the objects that use this vertex shader
        "uniform mat4 uMVPMatrix;" +
        "attribute vec4 vPosition;" +
```

```
        "void main() {" +
        // the matrix must be included as a modifier of gl_Position
        // Note that the uMVPMatrix factor *must be first* in order
        // for the matrix multiplication product to be correct.
        "  gl_Position = uMVPMatrix * vPosition;" +
        "}";

    // Use to access and set the view transformation
    private int mMVPMatrixHandle;

    ...
}
```

Bước 4 : Sửa lại hàm draw() cho phù hợp

```
public void draw(float[] mvpMatrix) { // pass in the calculated transformation matrix
    ...

    // get handle to shape's transformation matrix
    mMVPMatrixHandle = GLES20.glGetUniformLocation(mProgram, "uMVPMatrix");

    // Pass the projection and view transformation to the shader
    GLES20.glUniformMatrix4fv(mMVPMatrixHandle, 1, false, mvpMatrix, 0);

    // Draw the square
    GLES20.glDrawArrays(GLES20.GL_SQUARES, 0, vertexCount);

    // Disable vertex array
    GLES20.glDisableVertexAttribArray(mPositionHandle);
}
```

1.1.5. Thêm hiệu ứng hình ảnh (quan trọng)

Bước 1 : Thêm hiệu ứng xoay hình

```
private float[] mRotationMatrix = new float[16];
public void onDrawFrame(GL10 gl) {
    float[] scratch = new float[16];

    // Create a rotation transformation for the square
    long time = SystemClock.uptimeMillis() % 4000L;
    float angle = 0.090f * ((int) time);
    Matrix.setRotateM(mRotationMatrix, 0, angle, 0, 0, -1.0f);
```

```
    // Combine the rotation matrix with the projection and camera view
    // Note that the mMVPMatrix factor *must be first* in order
    // for the matrix multiplication product to be correct.
    Matrix.multiplyMM(scratch, 0, mMVPMatrix, 0, mRotationMatrix, 0);

    // Draw the square
    mSquare.draw(scratch);
}
```

Các bước tiếp theo ở phần sau.

1.1.6. Xử lí các sự kiện chạm (quan trọng)

Bước 1 : Cài đặt xử lí sự kiện chạm

```
private final float TOUCH_SCALE_FACTOR = 180.0f / 320;
private float mPreviousX;
private float mPreviousY;

@Override
public boolean onTouchEvent(MotionEvent e) {
    // MotionEvent reports input details from the touch screen
    // and other input controls. In this case, you are only
    // interested in events where the touch position changed.

    float x = e.getX();
    float y = e.getY();

    switch (e.getAction()) {
        case MotionEvent.ACTION_MOVE:

            float dx = x - mPreviousX;
            float dy = y - mPreviousY;

            // reverse direction of rotation above the mid-line
            if (y > getHeight() / 2) {
              dx = dx * -1 ;
            }

            // reverse direction of rotation to left of the mid-line
            if (x < getWidth() / 2) {
              dy = dy * -1 ;
            }
```

```
            mRenderer.setAngle(
                    mRenderer.getAngle() +
                    ((dx + dy) * TOUCH_SCALE_FACTOR));
            requestRender();
    }

    mPreviousX = x;
    mPreviousY = y;
    return true;
}
```

Bước 2 : Cập nhật lại hình vẽ qua hàm setRenderMode()

```
public MyGLSurfaceView(Context context) {
    ...
    // Render the view only when there is a change in the drawing data
    setRenderMode(GLSurfaceView.RENDERMODE_WHEN_DIRTY);
}
```

Bước 3 : Hiển thị góc xoay

```
public class MyGLRenderer implements GLSurfaceView.Renderer {
    ...

    public volatile float mAngle;

    public float getAngle() {
        return mAngle;
    }

    public void setAngle(float angle) {
        mAngle = angle;
    }
}
```

Bước 4 : Áp dụng việc xoay hình

```
public void onDrawFrame(GL10 gl) {
    float[] scratch = new float[16];


    // Create a rotation for the square
    // long time = SystemClock.uptimeMillis() % 4000L;
```

```
    // float angle = 0.090f * ((int) time);
    Matrix.setRotateM(mRotationMatrix, 0, mAngle, 0, 0, -1.0f);


    // Combine the rotation matrix with the projection and camera view
    // Note that the mMVPMatrix factor *must be first* in order
    // for the matrix multiplication product to be correct.
    Matrix.multiplyMM(scratch, 0, mMVPMatrix, 0, mRotationMatrix, 0);


    // Draw the square
    mSquare.draw(scratch);
}
```

Nhận xét : Hình vuông sẽ xoay theo chiều kim đồng hồ hay ngược lại tùy vào cách người dùng di chuyển trên màn hình.

1.2. Drawables

1.2.1. Giới thiệu Drawables

Drawables là một đối tượng đồ họa được hỗ trợ trong Android.

Nằm trong gói : android.graphics.drawable (chỉ cần khai báo là dùng được)

Hỗ trợ các lớp : BitmapDrawable, ShapeDrawable, PictureDrawable, LayerDrawable…

1.2.2. Sử dụng Drawables

1.3. Canvas

1.3.1. Giới thiệu Canvas

Canvas là một đối tượng đồ họa 2 chiều (2D) được hỗ trợ trong Android.

Ta phải tùy biến lại lớp con của android.view.View để sử dụng, tức là kế thừa lại lớp View.

Mã nguồn mẫu

```
import android.view.View;


public class ClassName extends View {
	// required constructor
	public ClassName(Context context, AttributeSet attrs) {
		super(context, attrs);
	}

	// this method draws on the view
	@Override
	protected void onDraw(Canvas canvas) {
		super.onDraw(canvas);

		// drawing code
	}
} // end ClassName
```

1.3.2. Hàm vẽ có sẵn

drawARGB(alpha, r, g, b) // tô màu

drawArc(..., paint) // vẽ một cung tròn

drawBitmap(…, paint) // vẽ một hình Bitmap

drawCircle(…, paint) // vẽ một hình tròn

drawLine(…, paint) // vẽ một đường thẳng

c.drawOval(…, paint) // vẽ một hình Oval hay hình tròn

drawPath(…, paint) // vẽ một đường nối

drawRect(…, paint) // vẽ một hình chữ nhật hay hình vuông

drawText(…, paint) // vẽ chuỗi kí tự

Nhận xét : Tất cả các hàm trên đều cần một biến màu vẽ (paint).

1.3.3. Sơn và màu vẽ

Cú pháp

Paint name = new Paint();

name.setARGB(alpha, red, green, blue);

Ví dụ

Paint purple = new Paint();

purple.setARGB(255, 255, 0, 255);

purple.setStyle(Style.FILL_AND_STROKE);

1.3.4. Xử lí sự kiện chạm

Cú pháp

```
@Override
public boolean onTouchEvent(MotionEvent event) {
        float x = event.getX();
        float y = event.getY();

        if (event.getAction() == MotionEvent.ACTION_DOWN) {
                // code to run when finger is pressed
        }

        return super.onTouchEvent(event);
}
```

2. Đồ án cuối kì : Game How to draw pattern?

2.1. Các chế độ chính

2.1.1. Chế độ đăng nhập – đăng xuất

Người dùng có thể đăng nhập vào trò chơi bằng tài khoản Google của mình và mọi thông tin của người dùng sẽ được hệ thống ghi nhận vào Firebase.

Lưu ý : thiết bị cần phải được kết nối mạng để thực hiện chức năng này.

2.1.2. Chế độ hướng dẫn (Tutorial) – Chưa hoàn thành

Người chơi có thể xem hướng dẫn để nắm được cách chơi và lối chơi.

Hệ thống sẽ mở 1 màn hình gồm 4 điểm và vẽ thử hình chữ U để người chơi tập vẽ theo.

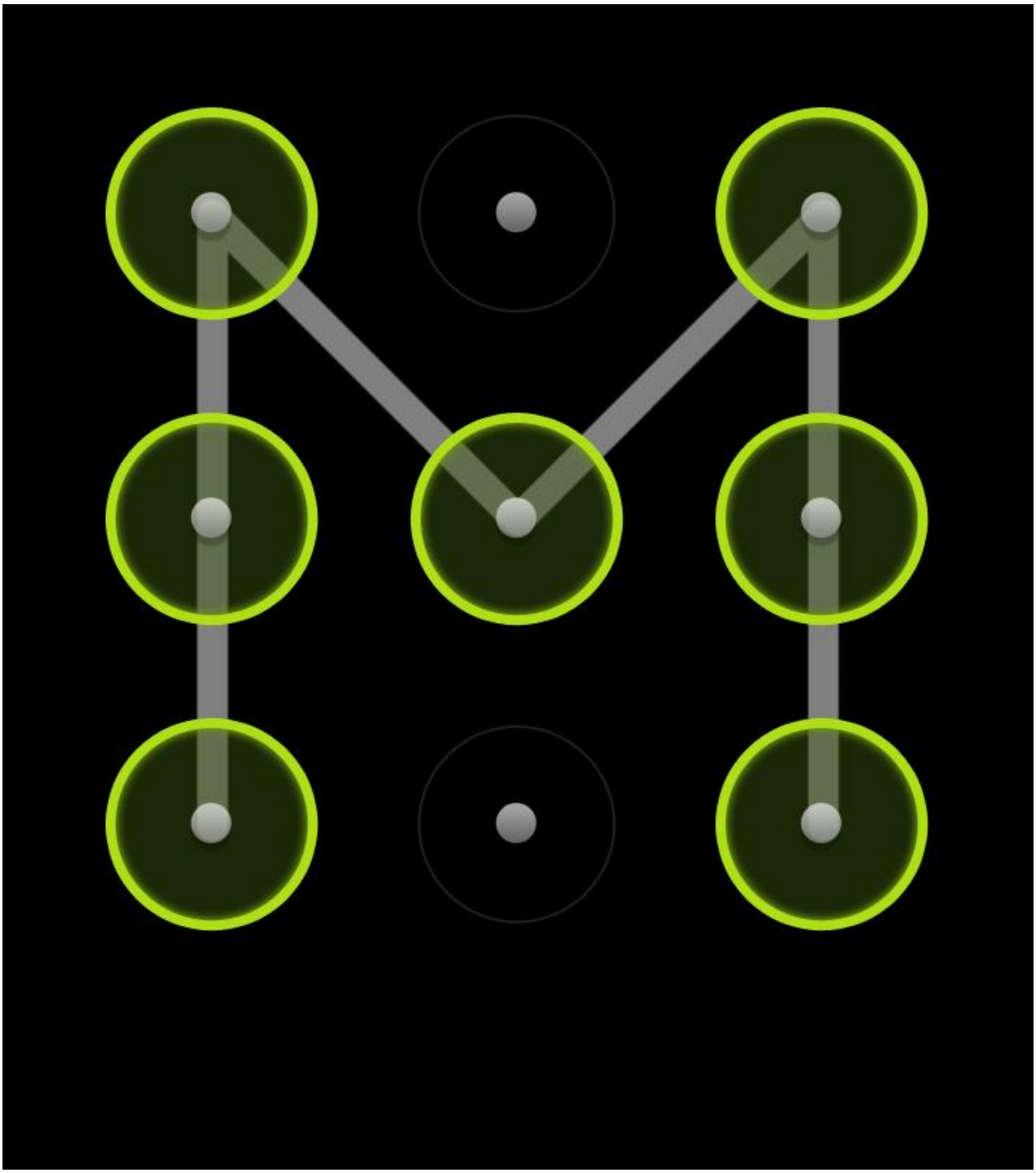
⇨ Màn hình hướng dẫn vẽ chữ U

2.1.3. Chế độ chơi đơn (Single player mode)

Hệ thống yêu cầu người chơi làm một yêu cầu nào đó

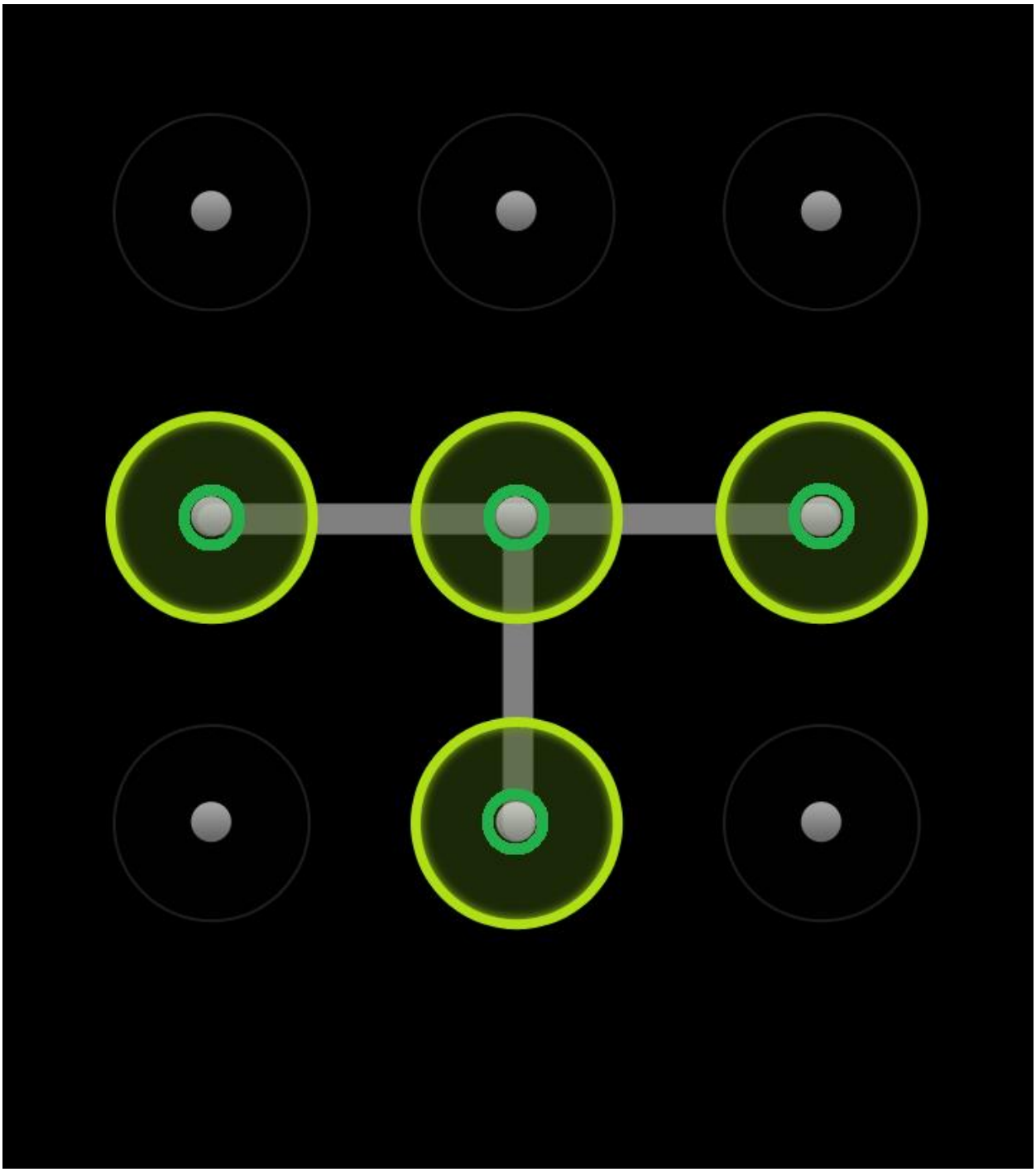
- **Vẽ lại theo một hình mẫu cho trước**

Ví dụ : vẽ lại hình chữ M sau đây



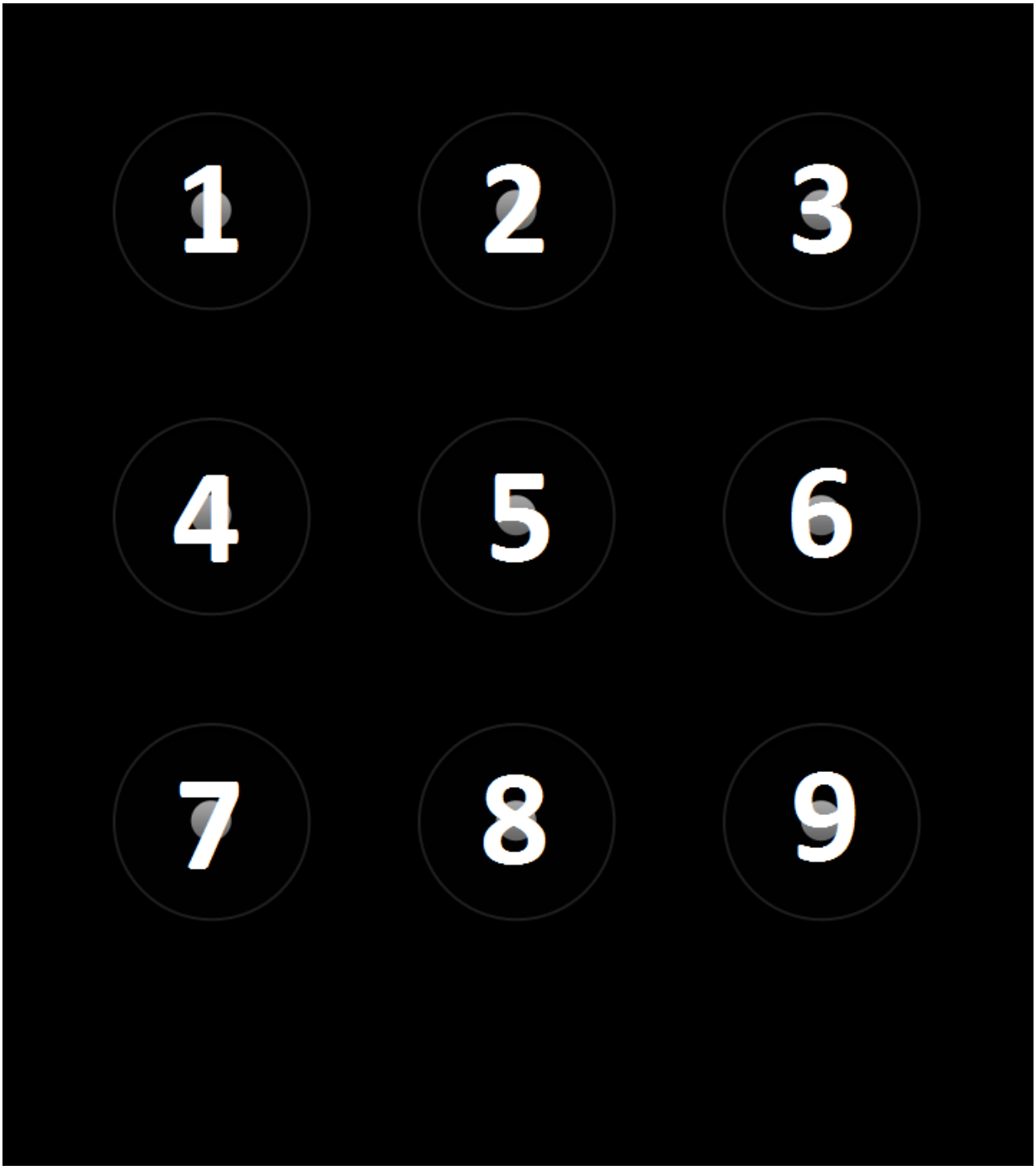
- **Tìm đường đi ngắn nhất đi qua các đỉnh thỏa mãn một điều kiện nào đó – Chưa hoàn thành**

Ví dụ : chỉ đi qua các đỉnh tô xanh



- **Giải một câu đố bất kì - Chưa hoàn thành**

Ví dụ : Giải phép toán sau ? + ? + ? = 15



Người chơi phải vẽ bằng cách nối các điểm trên bản đồ (mỗi điểm chỉ đi qua đúng một lần duy nhất) và chỉ được dùng 1 nét vẽ duy nhất (tính từ lúc đặt ngón tay xuống màn hình đến lúc nhấc ngón tay lên).

Người chơi chỉ được vẽ trong một khoảng thời gian (tính bằng giây) giới hạn cho trước. Trò chơi kết thúc khi hết thời gian vẽ hoặc người chơi nhấn nút bỏ cuộc.

Độ khó tăng dần bằng cách sau

+ Tăng độ phức tạp của hình mẫu, tức là tăng số lượng đỉnh cần đi qua (Passed points)

Ví dụ : 2 => 3 => 4 …

+ Tăng kích thước bản đồ (Map), tức là tăng số lượng điểm.

Ví dụ : 2x2 => 3x3 => 4x4 …

2.1.4. Chế độ qua màn – Chưa hoàn thành

- **Các màn chơi dùng bản đồ gồm 4 điểm (Map 2x2)**

Số đỉnh cần đi qua trong hình mẫu

+ Tối thiểu : 2

+ Tối đa : 4

Thời gian : 3 - 5 giây

Level	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Point	2	2	2	3	3	3	4	4	4	4

- **Các màn chơi dùng bản đồ gồm 9 điểm (Map 3x3)**

Số đỉnh cần đi qua trong hình mẫu

+ Tối thiểu : 4

+ Tối đa : 9

Thời gian : 5 - 10 giây

Level	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Point	4	4	4	5	5	5	6	6	6	7
Level	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
Point	7	7	8	8	8	8	9	9	9	9

- **Các màn chơi dùng bản đồ gồm 16 điểm (Map 4x4)**

Số đỉnh cần đi qua trong hình mẫu

+ Tối thiểu : 9

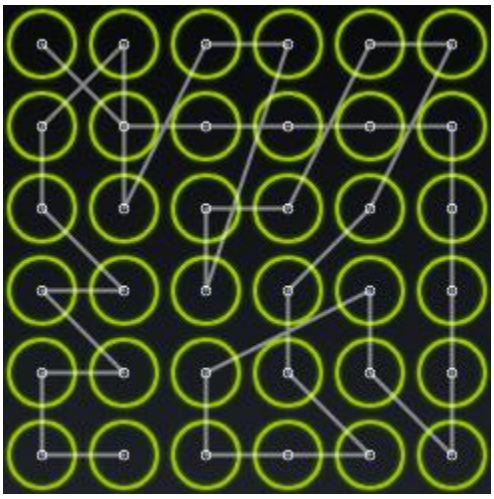
+ Tối đa : 16

Thời gian : 10 - 15 giây

Level	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**
Point	9	9	9	10	10	10	11	11	11	11
Level	**41**	**42**	**43**	**44**	**45**	**46**	**47**	**48**	**49**	**50**
Point	12	12	12	12	13	13	13	13	14	14
Level	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**
Point	14	14	15	15	15	15	16	16	16	16

- **Các màn chơi dùng bản đồ gồm rất nhiều điểm (Map 5x5 and more)**

Thiết kế sau, có thể là phiên bản nâng cấp trong tương lai



2.1.5. Chế độ thưởng (Reward) – Chưa hoàn thành

Người chơi thắng mỗi vòng sẽ được cộng một lượng tiền nhất định (Coin). Đây là đơn vị tiền tệ duy nhất trong trò chơi và có thể dùng để mua vật phẩm trong Shop.

Khi vượt qua các vòng 5, 10, 15, 20… người chơi sẽ được quyền mở khóa 1 vật phẩm (Item) có độ hiếm ngẫu nhiên. Vật phẩm có độ hiếm càng cao thì sẽ khó mở ra hơn nhưng sẽ có giá trị cao hơn.

Sau khi trò chơi kết thúc, hệ thống sẽ hiện các phần thưởng mà người chơi đạt được gồm có tiền và các vật phẩm tương ứng.

2.1.6. Chế độ nhiều người chơi (Multi player mode)

Người chơi có thể tham gia thi đấu với những người chơi khác, cũng có thể mời bạn bè của mình, tạo thành 1 giải đấu. Người chiến thắng là người có thành tích cao nhất (tức là vượt qua nhiều màn nhất). Hệ thống sẽ xác định người thắng cuộc khi các người cùng chơi khác đã bị loại.

Ví dụ : Giải đấu gồm 5 người chơi A, B, C, D, E. Sau một thời gian chơi, E và D dừng lại ở màn 5, C dừng lại ở màn 10 và B dừng lại ở màn 15. Khi A vượt qua màn 15 thì A là người chiến thắng và A không cần chơi tiếp các màn sau nữa.

Hệ thống màn chơi và hệ thống trao thưởng sẽ giữ nguyên như mục 1.2 và 1.3.

Ngoài ra, người chiến thắng sẽ được thưởng thêm, tỉ lệ với độ lớn của giải đấu.

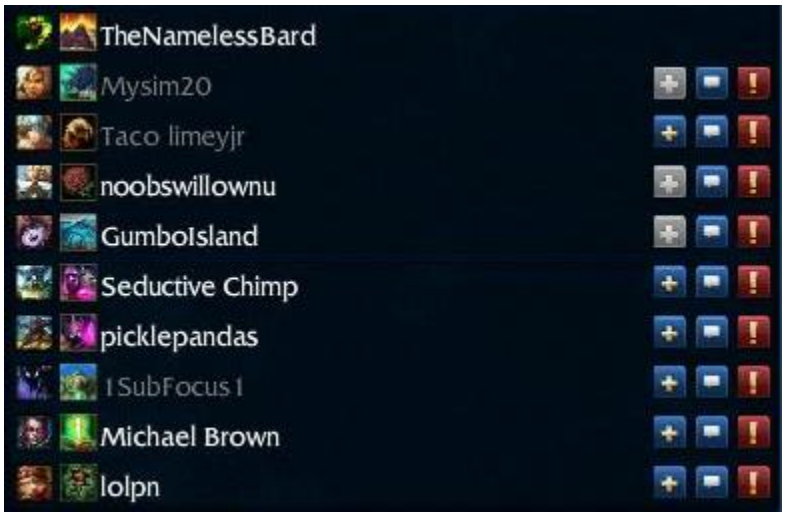
Players	1st	2nd	3rd
>= 5	1 Item		
>= 10	2 Items	1 Item	
>= 15	3 Items	2 Items	1 Item

Ví dụ : Theo ví dụ trên thì A sẽ được thưởng thêm 1 vật phẩm.

2.1.7. Chế độ kết bạn – Chưa hoàn thành

Người chơi có thể kết bạn với những người chơi khác ở chung giải đấu sau khi đã kết thúc phần chơi của mình.

⇨ Màn hình kết thúc giải đấu và hiển thị nút kết bạn



Ngoài ra, người chơi có thể kết bạn với người chơi khác bằng cách dùng id / username của người kia.

2.1.8. Chế độ mua và trao đổi vật phẩm – Chưa hoàn thành

Shop là nơi người chơi có thể mua thêm vật phẩm bằng tiền hoặc trao đổi những vật phẩm đang có.

Bảng quy chiếu các vật phẩm trong Shop dựa trên độ hiếm của chúng

Rarity	Price	Trade
Common	1 Coin	
Rare	10 Coins	10 Common item
Super Rare	100 Coins	10 Rare item
Ultra Rare	1000 Coins	10 Super rare item

2.2. Use-case diagram

Người chơi có thể

+ Đăng nhập dùng tài khoản Google của mình (người chơi phải đăng nhập để thực hiện các chức năng khác)

+ Kết bạn với người chơi khác

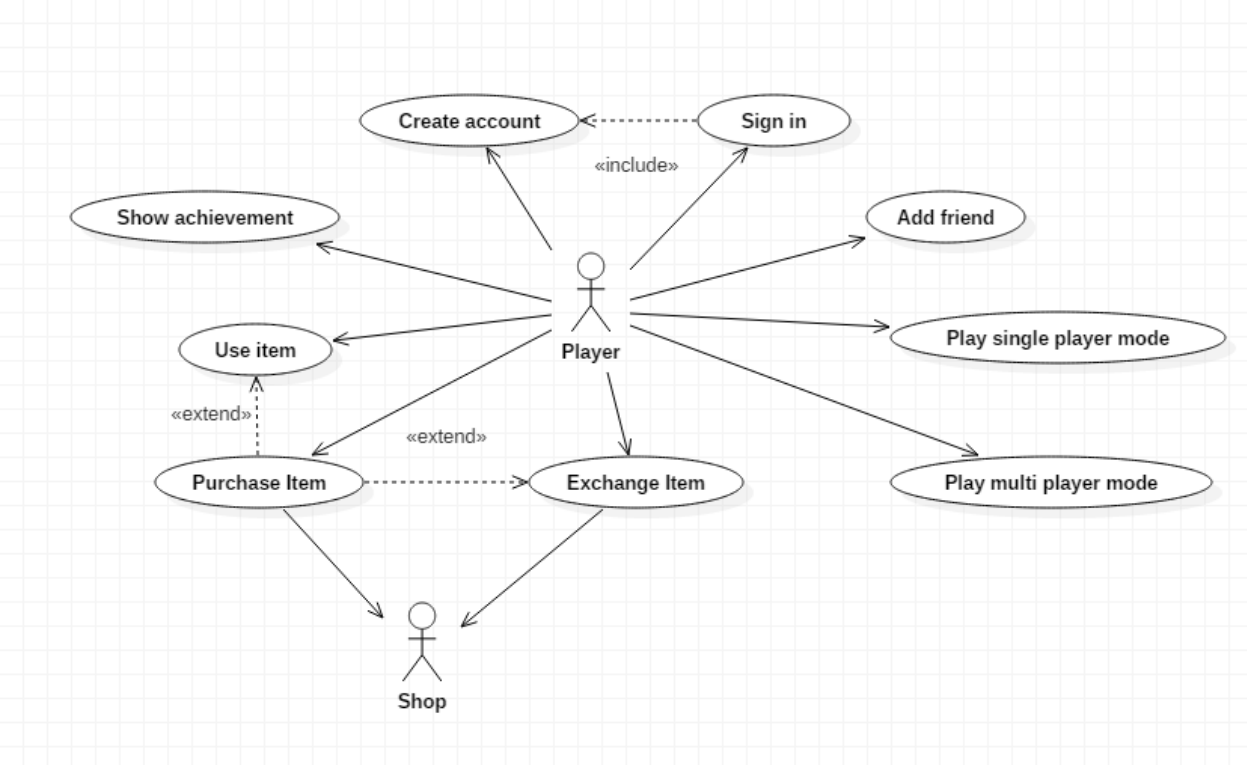
+ Chơi chế độ đơn, cố gắng đạt thành tích cao nhất

+ Chơi chế độ nhiều người, thi đấu với các người chơi khác

+ Kheo thành tích mà người chơi đạt được

+ Sử dụng vật phẩm đang có trong Kho

+ Mua thêm vật phẩm hoặc trao đổi vật phẩm trong Cửa hàng



2.3. Class

2.3.1. Vật phẩm (Item)

Mã vật phẩm, tên vật phẩm, độ hiếm và loại bao gồm

+ Hình nền (Cover)

+ Hình đại diện (Avatar)

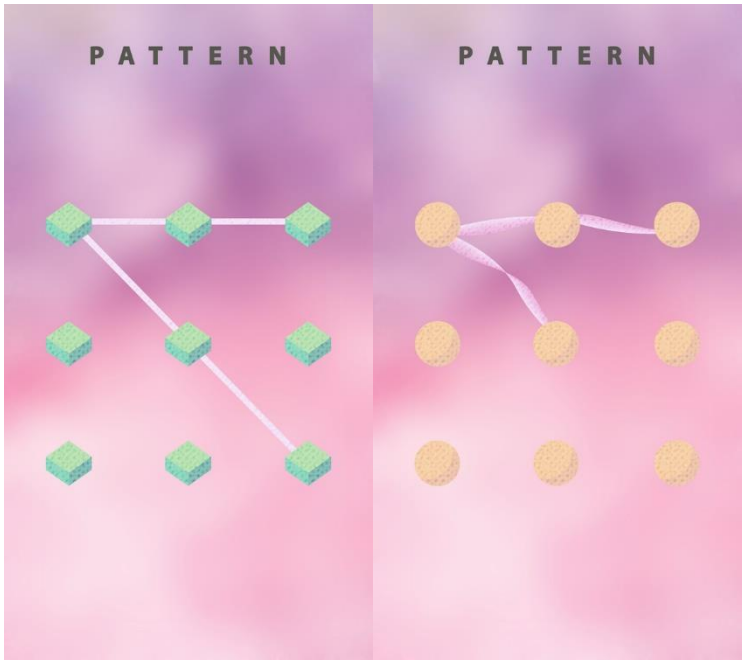
+ Hình nét vẽ (Line)

+ Hình điểm vẽ (Point)

Ví dụ : Ta có 2 mẫu vật phẩm sau để sử dụng

+ Hình thứ 1 (bên trái) có điểm vẽ hình hộp và nét vẽ thẳng tấp

+ Hình thứ 2 (bên phải) có điểm vẽ hình cầu và nét vẽ uốn lượn



2.3.2. Người chơi (Player)

Thông tin tài khoản : ID (tự động cấp), Gmail, Nickname

Thông tin cá nhân : họ tên, sinh nhật, địa chỉ, điện thoại

Thông tin quá trình chơi : kỉ lục (màn chơi cao nhất đã vượt qua), số tiền đang có, danh sách vật phẩm đang sở hữu, danh sách bạn bè.

2.3.3. Shop

Danh sách vật phẩm đang bán

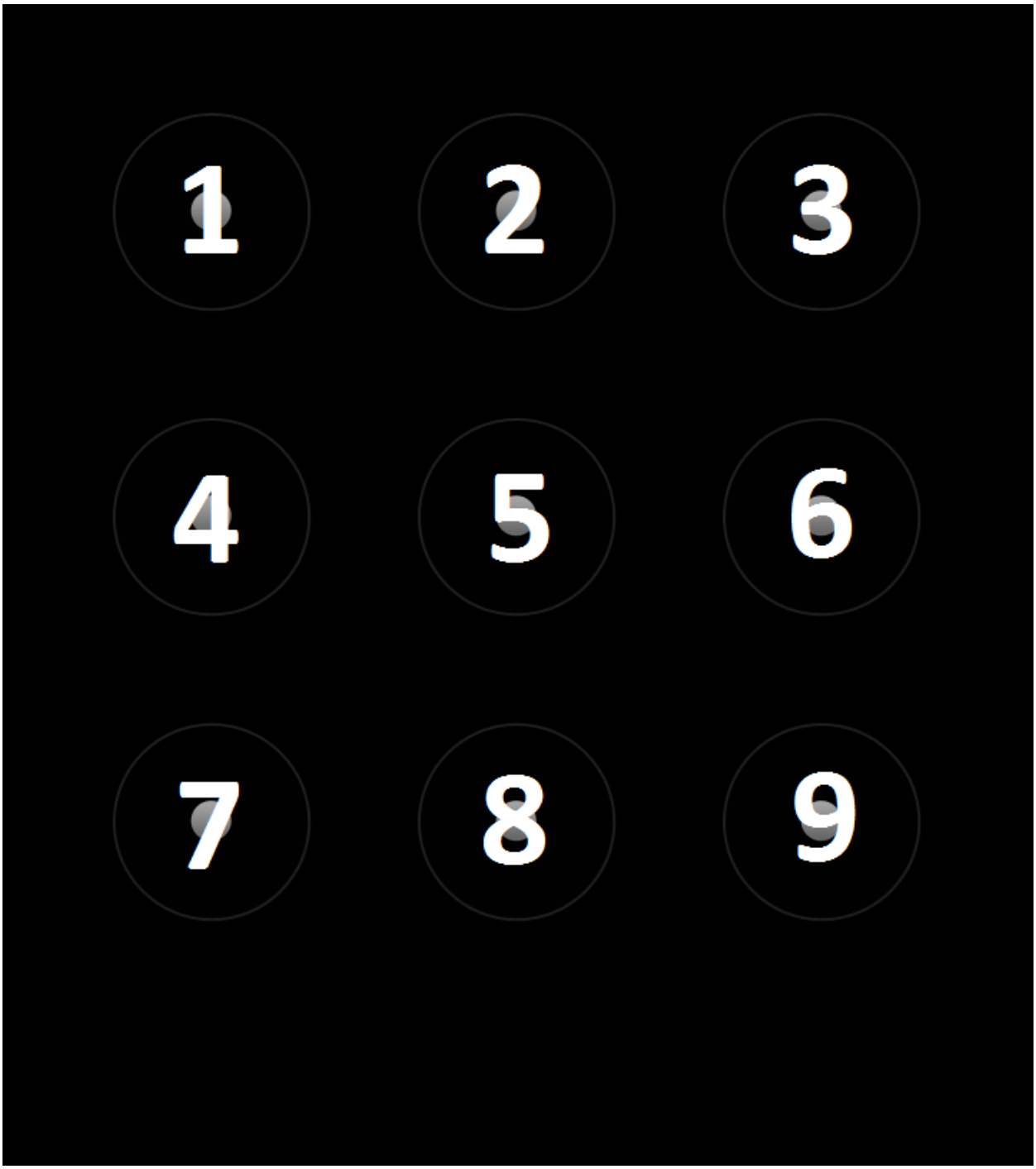
Danh sách vật phẩm có thể trao đổi

Ghi chú : 2 danh sách trên khác nhau hoàn toàn

2.3.4. Điểm (Point)

Là điểm để vẽ trên màn hình

Mỗi điểm trên bản đồ sẽ được đánh số từ trái sang phải và từ trên xuống dưới như sau



Do đó, ta có thể quy hình mẫu sang dạng chuỗi số để xử lí và lưu trữ dễ dàng hơn. Dễ thấy là một hình mẫu luôn có 2 cách vẽ (trừ một số trường hợp ngoại lệ).

Ví dụ : Chữ M trong bản đồ 3x3 ở trên sẽ có giá trị 7415369 hoặc 9635147

2.3.5. Bản đồ (Map)

Danh sách toàn bộ các điểm trên bản đồ

Ghi chú : bản đồ luôn có dạng hình vuông (số lượng hàng bằng đúng số lượng cột)

2.3.6. Hình mẫu (Pattern)

Danh sách các điểm tạo ra 1 hình mẫu nào đó.

Ví dụ : hình chữ cái như C, O, N… hình con vật như con cá, con chim…

2.4. Database

Các hình mẫu để vẽ theo

Mã số (ID)	Số điểm tối đa có thể đi qua	Độ khó	Tên hình ảnh
001	4	1	p123.png
…	…	…	…
010	9	3	p12345.png
…	…	…	…
100	9	5	p123456789.png

Giải thích

+ ID sẽ do hệ thống tự động cung cấp

+ Số điểm tối đa có thể đi qua tương ứng với kích thước bản đồ. Ví dụ, bản đồ 2x2 thì sẽ là 4 điểm.

+ Độ khó của hình mẫu được đánh giá từ 1 đến 5 để sau này hiển thị theo dạng số sao trên màn hình chơi.

+ Tên hình ảnh được lưu theo dạng chuỗi số (bắt đầu bằng chữ “p” để phù hợp với quy định của Android Studio) và có phần mở rộng là .png.

2.5. Interface

Lưu ý : Giao diện sẽ được trình bày theo dạng Chữ hiển thị (Tên biến trong tập tin xml).

Ví dụ : Nút A_B_C (a_b_c) tức là nút này sẽ hiển thị chuỗi A_B_C và tên biến trong tập tin xml là a_b_c.

2.5.1. Game loading

Giao diện khi game được mở lên

2.5.2. Login

Trên màn hình LoginActivity có các thành phần sau

+ Nút Sign in (sign_in_button) được hiện lên, ở chính giữa phía dưới màn hình để người dùng có thể đang nhập vào trò chơi

+ Nút Sign out (sign_out_button) bị ẩn đi, ở bên trái phía dưới màn hình

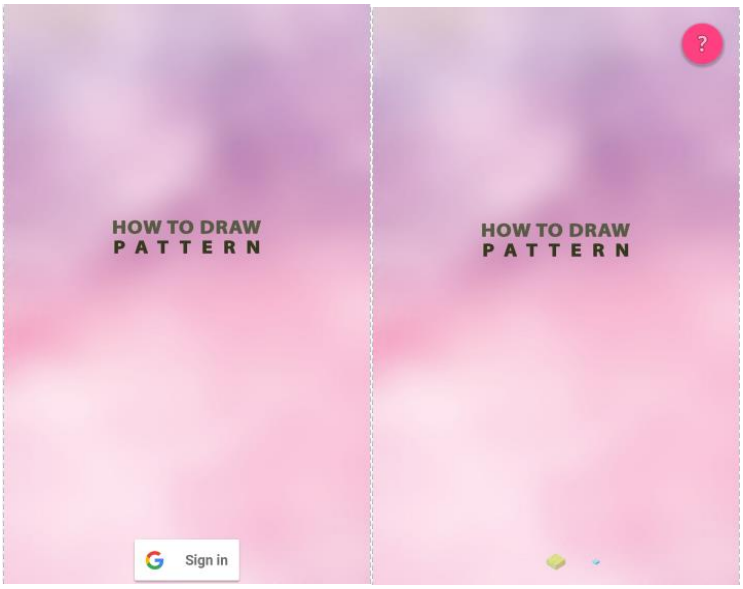
+ Nút Let’s rock (play_game_button) bị ẩn đi, ở bên phải phía dưới màn hình

Sau khi người dùng đăng nhập bằng tài khoản Google của mình thì

+ Nút Sign in sẽ bị ẩn đi

+ Nút Sign out sẽ hiện lên để người dùng có thể đăng xuất khỏi trò chơi

+ Nút Let’s rock sẽ hiện lên để người dùng có thể vào Main Menu



2.5.3. Main Menu

Trên màn hình MainActivity bao gồm nhiều thẻ (Tab)

+ Single Player : chơi chế độ đơn

+ Multiplayer : chơi chế độ nhiều người

+ Shop : mua và trao đổi vật phẩm

+ Tutorial : xem hướng dẫn chơi

Phía dưới là thanh truy cập nhanh (Navigation bar)

Người dùng có thể vuốt sang trái hoặc sang phải để chuyển giữa các thẻ và nhấn vào màn hình để chọn (nhấn vào đâu cũng được).

Ngoài ra, còn có một nút hình dấu chấm hỏi để hiển thị hướng dẫn (hình phía trên).

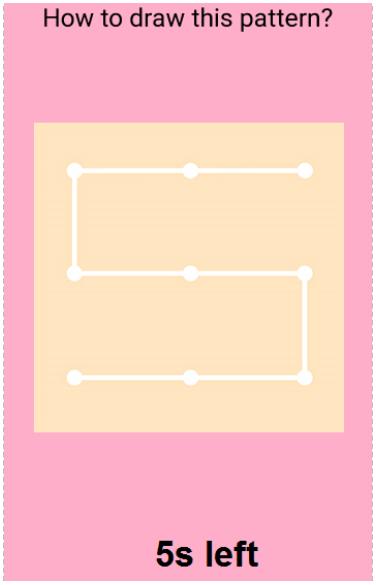
2.5.4. Single player mode

Bước 1 : Màn hình PatternActivity sẽ hiển thị hình mẫu cần vẽ.

+ Chuỗi kí tự (không có id) nằm ở phía trên màn hình chỉ để hiển thị “How to draw this pattern?”.

+ Hình mẫu (pattern_image) nằm ở ở chính giữa để hiển thị hình mẫu cần vẽ.

+ Chuỗi kí tự (pattern_time_text) nằm ở phía dưới màn hình dùng để hiển thị thời gian còn lại trước khi hình mẫu biến mất.



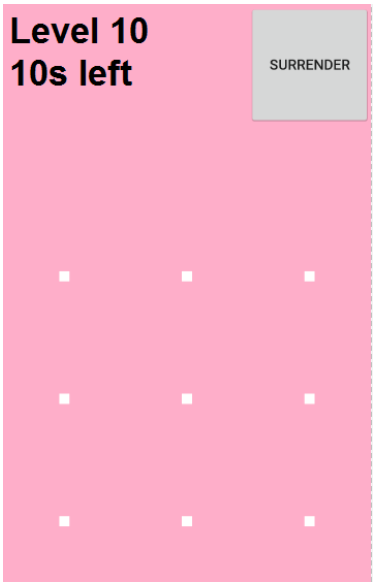
Bước 2 : Màn hình GameplayActivity sẽ hiển thị giao diện cho người chơi vẽ

+ Chuỗi kí tự (gameplay_level_text) nằm ở bên phải phía trên màn hình dùng để hiển thị cấp độ hiện tại của trò chơi

+ Chuỗi kí tự (gameplay_time_text) nằm ở phía dưới chuỗi trên dùng để hiển thị thời gian vẽ còn lại

+ Nút Surrender (gameplay_surrender_button) nằm ở bên phải phía trên màn hình dùng để giúp người chơi đầu hàng và thoát khỏi màn chơi đó

+ Đố tượng PatternDrawer (pattern) nằm ở phía dưới màn hình sẽ cung cấp giao diện để người dùng vẽ hình



2.5.5. Game over

Màn hình ResultActivity sẽ hiện ra khi người dùng đầu hàng hoặc thời gian vẽ đã hết. Một thời gian ngắn sau đó, hệ thống sẽ tự động chuyển sang màn hình MainActivity.

Chuỗi kí tự (không có id) nằm ở chính giữa màn hình chỉ để hiển thị “Game over”.

3. Tài liệu tham khảo

OpenGL ES

https://developer.android.com/guide/topics/graphics/opengl.html

Canvas & Drawables

https://developer.android.com/guide/topics/graphics/2d-graphics.html