Chúng ta sẽ bắt đầu từng bước để xây dựng game Mario mini trên Android Studio, sử dụng **Java**, **Canvas**, **SharedPreferences**, và **SQLite**, theo trình tự từ nền tảng đến tính năng nâng cao. Bước đầu tiên là tạo project và lớp hiển thị đồ họa.

🛠️ Bước 1: Tạo Project & Thiết lập giao diện cơ bản

1.1 Tạo Project Android Studio

- **Ngôn ngữ**: Java
- **Activity**: `Empty Activity`
- **Tên Activity**: `GameActivity.java`
- **Package name**: `com.example.mariogame`

1.2 Cập nhật `AndroidManifest.xml` để chạy full màn hình

```
<application ... >
    <activity
        android:name=".GameActivity"
        android:theme="@style/Theme.AppCompat.NoActionBar">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
```

1.3 Tạo lớp `GameView.java`

```
public class GameView extends SurfaceView implements Runnable {
    private Thread gameThread;
    private boolean isPlaying = true;
    private Paint paint;
    private SurfaceHolder surfaceHolder;

    public GameView(Context context) {
        super(context);
        surfaceHolder = getHolder();
        paint = new Paint();
    }

    @Override
    public void run() {
        while (isPlaying) {
            if (!surfaceHolder.getSurface().isValid()) continue;

            Canvas canvas = surfaceHolder.lockCanvas();
            canvas.drawColor(Color.CYAN); // nền trời
            canvas.drawText("Loading Mario...", 100, 100, paint);
            surfaceHolder.unlockCanvasAndPost(canvas);
        }
    }

    public void pause() {
        isPlaying = false;
        try {
            gameThread.join();
```

```
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    public void resume() {
        isPlaying = true;
        gameThread = new Thread(this);
        gameThread.start();
    }
}
```

1.4 Sử dụng `GameView` trong `GameActivity.java`

```
public class GameActivity extends AppCompatActivity {
    private GameView gameView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        gameView = new GameView(this);
        setContentView(gameView);
    }

    @Override
    protected void onPause() {
        super.onPause();
        gameView.pause();
    }

    @Override
    protected void onResume() {
        super.onResume();
        gameView.resume();
    }
}
```

Bước 2: Thêm nhân vật Mario và vẽ bằng Canvas

2.1 Thêm hình ảnh Mario vào dự án

- Đặt file ảnh `mario.png` vào thư mục `res/drawable`.

2.2 Cập nhật `GameView.java` để vẽ Mario

```
public class GameView extends SurfaceView implements Runnable {
    private Thread gameThread;
    private boolean isPlaying;
    private Paint paint;
    private SurfaceHolder surfaceHolder;
    private Bitmap marioBitmap;
    private int marioX = 100, marioY = 800; // vị trí Mario
    private int marioVelocityY = 0;

    public GameView(Context context) {
        super(context);
        surfaceHolder = getHolder();
        paint = new Paint();
        marioBitmap = BitmapFactory.decodeResource(getResources(),
R.drawable.mario);
```

```
    }

    @Override
    public void run() {
        while (isPlaying) {
            if (!surfaceHolder.getSurface().isValid()) continue;

            update(); // cập nhật logic game
            draw();   // vẽ lại khung hình
        }
    }

    private void update() {
        marioY += marioVelocityY;
        if (marioY > getHeight() - marioBitmap.getHeight()) {
            marioY = getHeight() - marioBitmap.getHeight();
        }
    }

    private void draw() {
        Canvas canvas = surfaceHolder.lockCanvas();
        canvas.drawColor(Color.CYAN); // nền trời
        canvas.drawBitmap(marioBitmap, marioX, marioY, paint);
        surfaceHolder.unlockCanvasAndPost(canvas);
    }

    public void pause() {
        isPlaying = false;
        try {
            gameThread.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    public void resume() {
        isPlaying = true;
        gameThread = new Thread(this);
        gameThread.start();
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        if (event.getAction() == MotionEvent.ACTION_DOWN) {
            marioVelocityY = -30; // nhảy lên
        } else if (event.getAction() == MotionEvent.ACTION_UP) {
            marioVelocityY = 10; // rơi xuống
        }
        return true;
    }
}
```

✅ **Kết quả:** Mario sẽ nhảy lên khi bạn chạm màn hình và rơi xuống khi nhả tay ra.

Bước 3: Thêm chướng ngại vật & kiểm tra va chạm

3.1 Thêm hình ảnh chướng ngại vật

- Thêm ảnh `obstacle.png` vào thư mục `res/drawable`.

3.2 Khai báo biến và khởi tạo chướng ngại vật

Thêm vào `GameView.java`:

```
private Bitmap obstacleBitmap;
private int obstacleX, obstacleY;
private int obstacleSpeed = 20;
```

Trong constructor:

```
obstacleBitmap = BitmapFactory.decodeResource(getResources(),
R.drawable.obstacle);
obstacleX = getWidth() + 200; // bắt đầu từ ngoài màn hình
obstacleY = getHeight() - 200; // gần đáy
```

3.3 Cập nhật `update()` để di chuyển chướng ngại vật

```
private void update() {
    marioY += marioVelocityY;
    if (marioY > getHeight() - marioBitmap.getHeight()) {
        marioY = getHeight() - marioBitmap.getHeight();
    }

    obstacleX -= obstacleSpeed;
    if (obstacleX + obstacleBitmap.getWidth() < 0) {
        obstacleX = getWidth(); // reset vị trí chướng ngại vật
    }

    // Kiểm tra va chạm
    if (Rect.intersects(getRectMario(), getRectObstacle())) {
        isPlaying = false;
        // Thêm xử lý kết thúc game nếu cần
    }
}
```

3.4 Thêm hàm tạo `Rect` cho va chạm

```
private Rect getRectMario() {
    return new Rect(marioX, marioY, marioX + marioBitmap.getWidth(), marioY
+ marioBitmap.getHeight());
}

private Rect getRectObstacle() {
    return new Rect(obstacleX, obstacleY, obstacleX +
obstacleBitmap.getWidth(), obstacleY + obstacleBitmap.getHeight());
}
```

3.5 Vẽ chướng ngại vật trong `draw()`

```
canvas.drawBitmap(obstacleBitmap, obstacleX, obstacleY, paint);
```

✓ **Kết quả:** Chướng ngại vật sẽ trượt từ phải sang trái. Nếu Mario va vào chướng ngại vật, game sẽ dừng lại.

Bước 4: Thêm hệ thống điểm và lưu cấp độ bằng SharedPreferences

---

4.1 Thêm điểm số và cấp độ vào GameView

```
private int score = 0;
private int level = 1;
private SharedPreferences prefs;
private SharedPreferences.Editor editor;
```

Trong constructor:

```
prefs = getContext().getSharedPreferences("game_data",
Context.MODE_PRIVATE);
editor = prefs.edit();
level = prefs.getInt("level", 1); // nếu chưa có thì mặc định level 1
```

4.2 Cập nhật điểm trong `update()` và xử lý qua chướng ngại vật

```
if (obstacleX + obstacleBitmap.getWidth() < marioX && !scored) {
    score += 10;
    scored = true;
    if (score % 50 == 0) {
        level++;
        editor.putInt("level", level);
        editor.apply();
        obstacleSpeed += 5; // tăng tốc độ chướng ngại vật
    }
}
if (obstacleX + obstacleBitmap.getWidth() < 0) {
    obstacleX = getWidth();
    scored = false;
}
```

4.3 Vẽ điểm và level trong `draw()`

```
paint.setColor(Color.BLACK);
paint.setTextSize(60);
canvas.drawText("Score: " + score, 50, 100, paint);
canvas.drawText("Level: " + level, 50, 180, paint);
```

✓ **Kết quả:** Khi Mario vượt qua chướng ngại vật, điểm sẽ tăng. Mỗi 50 điểm, level tăng và chướng ngại vật chạy nhanh hơn. Dữ liệu level sẽ được lưu lại khi thoát game.

Bước 5: Thêm SQLite để lưu lịch sử chơi game

5.1 Tạo lớp `GameDatabaseHelper`

Tạo file `GameDatabaseHelper.java`:

```
public class GameDatabaseHelper extends SQLiteOpenHelper {
    private static final String DB_NAME = "game.db";
    private static final int DB_VERSION = 1;

    public GameDatabaseHelper(Context context) {
        super(context, DB_NAME, null, DB_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("CREATE TABLE IF NOT EXISTS play_history (" +
                "id INTEGER PRIMARY KEY AUTOINCREMENT, " +
                "score INTEGER, " +
                "level INTEGER, " +
                "play_date TEXT)");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        db.execSQL("DROP TABLE IF EXISTS play_history");
        onCreate(db);
    }

    public void saveResult(int score, int level) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put("score", score);
        values.put("level", level);
        values.put("play_date", new SimpleDateFormat("yyyy-MM-dd HH:mm",
Locale.getDefault()).format(new Date()));
        db.insert("play_history", null, values);
        db.close();
    }
}
```

5.2 Sử dụng SQLite trong `GameView`

Thêm thuộc tính:

```
private GameDatabaseHelper dbHelper;
```

Trong constructor:

```
dbHelper = new GameDatabaseHelper(getContext());
```

5.3 Ghi kết quả khi game kết thúc

Trong phần xử lý va chạm:

```
if (Rect.intersects(getRectMario(), getRectObstacle())) {
    isPlaying = false;
```

```
    dbHelper.saveResult(score, level);
}
```

✅ **Kết quả:** Mỗi lần người chơi va chạm và kết thúc game, kết quả sẽ được lưu vào CSDL SQLite: điểm, level, ngày giờ.

Bước 6: Tạo Menu chính và xem lịch sử chơi

6.1 Tạo giao diện `activity_main.xml`

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    android:padding="24dp">

    <Button
        android:id="@+id/btnStart"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bắt đầu chơi" />

    <Button
        android:id="@+id/btnHistory"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Xem lịch sử" />
</LinearLayout>
```

6.2 Tạo `MainActivity.java`

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        findViewById(R.id.btnStart).setOnClickListener(v ->
                startActivity(new Intent(this, GameActivity.class)));

        findViewById(R.id.btnHistory).setOnClickListener(v ->
                startActivity(new Intent(this, HistoryActivity.class)));
    }
}
```

6.3 Tạo `activity_history.xml`

```
<ListView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/listView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

6.4 Tạo `HistoryActivity.java`

```
public class HistoryActivity extends AppCompatActivity {
    ListView listView;
    GameDatabaseHelper dbHelper;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_history);
        listView = findViewById(R.id.listView);
        dbHelper = new GameDatabaseHelper(this);
        showData();
    }

    private void showData() {
        SQLiteDatabase db = dbHelper.getReadableDatabase();
        Cursor cursor = db.rawQuery("SELECT * FROM play_history ORDER BY id
DESC", null);
        ArrayList<String> records = new ArrayList<>();

        while (cursor.moveToNext()) {
            int score = cursor.getInt(cursor.getColumnIndex("score"));
            int level = cursor.getInt(cursor.getColumnIndex("level"));
            String date =
cursor.getString(cursor.getColumnIndex("play_date"));
            records.add("Score: " + score + " | Level: " + level + " | " +
date);
        }

        cursor.close();
        db.close();

        ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
android.R.layout.simple_list_item_1, records);
        listView.setAdapter(adapter);
    }
}
```

6.5 Cập nhật `AndroidManifest.xml`

```
<activity android:name=".HistoryActivity" />
<activity android:name=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
    </intent-filter>
</activity>
```

✅ **Kết quả:** Khi mở ứng dụng, bạn có thể:

- **Bắt đầu game** từ menu chính.
- **Xem danh sách các lượt chơi cũ** với điểm, level, ngày giờ.

Tuyệt! Bây giờ ta sẽ bổ sung:

🎉 Bước 7: Thêm Dialog chiến thắng và nhạc chúc mừng

7.1 Thêm nhạc chiến thắng

- Thêm file `victory.mp3` vào thư mục `res/raw`.

7.2 Khai báo `MediaPlayer` trong `GameView`

```
private MediaPlayer victorySound;
```

Trong constructor:

```
victorySound = MediaPlayer.create(getContext(), R.raw.victory);
```

7.3 Kiểm tra điều kiện qua level trong `update()`

Giả sử mỗi level cần 100 điểm:

```
if (score >= level * 100 && !levelCompleted) {
    levelCompleted = true;
    victorySound.start();

    new Handler(Looper.getMainLooper()).post(() -> {
        new AlertDialog.Builder(getContext())
            .setTitle("🎉 Hoàn thành Level " + level)
            .setMessage("Chúc mừng! Bạn đã vượt qua level " + level)
            .setCancelable(false)
            .setPositiveButton("Chơi tiếp", (dialog, which) -> {
                level++;
                obstacleSpeed += 5;
                score = 0;
                editor.putInt("level", level);
                editor.apply();
                levelCompleted = false;
                obstacleX = getWidth();
            })
            .show();
    });
}
```

7.4 Khai báo `levelCompleted` để tránh gọi lại dialog nhiều lần

```
private boolean levelCompleted = false;
```

✅ **Kết quả:** Khi đạt đủ điểm của một level, sẽ có:

- **Nhạc chiến thắng phát ra**
- **Dialog hiện chúc mừng và nút chơi tiếp**

- **Tăng level và tốc độ**, reset lại điểm và vị trí chướng ngại vật

---