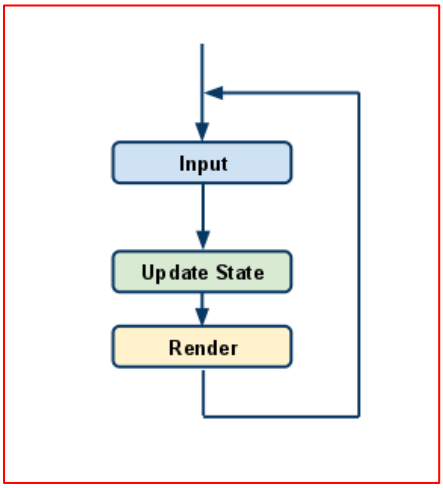
Bài 0 - Cấu trúc game

Dù đơn giản thế nào thì 1 game cũng cần phải có cấu trúc cơ bản như sau:

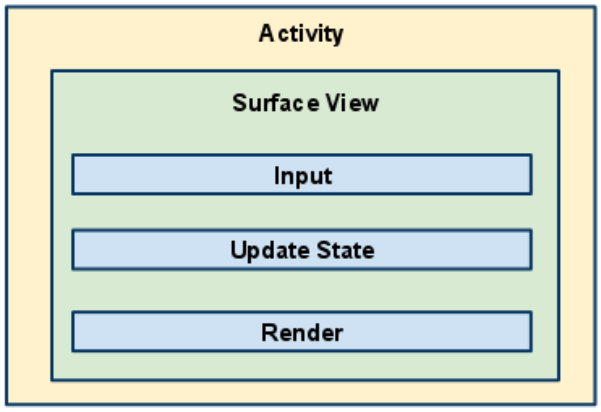
1. Xử lý đầu vào (input).
2. Cập nhật trạng thái bên trong.
3. Render ra màn hình (có thêm phần phát âm thanh, rung….)

Ba việc này lặp đi lặp lại tạo thành loop game.



Update và Render được nhóm lại 1 cách logic.

Tất cả mọi thứ trong Android đều xảy ra trong Activity. Activity sẽ tạo ra một View. View là nơi mọi thứ sẽ xảy ra. Nó là nơi nhận việc chạm tay và cũng là nơi hiển thị hình ảnh và dữ liệu. Tưởng tượng Activity là một vật chứa, chứa 1 tờ giấy (View), trên tờ giấy ta có thể vẽ nhiều thứ. Ta sẽ dùng bút chì để vẽ lên tờ giấy. Chạm tay là một hành động vật lý xảy ra trên tờ giấy và kết quả của sự tương tác trên view sẽ là hình ảnh. Trong Activity sẽ có diagram như sau:



Phần 1 : Xây dựng cấu trúc 2D

1. Tạo project mới đặt tên vd là “Cautruccoban”
2. Tạo 1 file class mới tên “GamePanel.java”. Kế thừa từ lớp SurfaceView. Xây hàm tạo và tạo override lên hàm onDraw (chú ý hàm này nằm trong View).

```
public class GamePanel extends SurfaceView {

	public GamePanel(Context context) {
		super(context);
		// TODO Auto-generated constructor stub
	}

	@Override
	protected void onDraw(Canvas canvas) {
		// TODO Auto-generated method stub
		super.onDraw(canvas);
	}
}
```

SurfaceView là một bề mặt để vẽ và có thể nhúng vào các View. Ta có thể điều khiển kích thước bề mặt vẽ này. Surface như 1 bề mặt được đặt chồng lên trên một cửa sổ (cửa sổ này là view chứa nó).

Việc truy cập đến bề mặt vẽ này được cung cấp thông qua interface tên SurfaceHolder, truy xuất bằng cách gọi getHolder(). Thực thi từ interface tên SurfaceHolder.Callback để truy xuất thông tin về sự thay đổi của bề mặt vẽ.

Bề mặt vẽ sẽ được tạo trong khi cửa sổ SurfaceView được visible. Ta phải ghi đè lên 2 hàm surfaceCreated() và surfaceDestroyed() để xử lý khi bề mặt vẽ được vẽ và được hủy (khi cửa sổ ẩn và hiện).

Một trong những chức năng của lớp này là cung cấp 1 bề mặt vẽ và trên đó cho phép một thread thứ cấp có thể render lên màn hình. Khi đó ta phải biết thêm về luồng (thread).

Tất cả SurfaceView và SurfaceHolder.CallBack sẽ được gọi từ luồng đang chạy cửa sổ SurfaceView (thường là luồng chính của chương trình). Do đó ta cần phải đồng bộ (synchronize) với các trạng thái chạm của luồng vẽ.

Ta phải chắc rằng luồng vẽ chỉ tác động vào bề mặt vẽ chỉ khi nó còn hợp lệ (giữa SurfaceCreated() và SurfaceDestroys().

3. Cho lớp GamePanel được implements từ SurfaceHolder.Callback. Sau đó ta phải triển khai 3 hàm là surfaceChanged(), surfaceCreated() và surfaceDestroyed().

```
public class GamePanel extends SurfaceView implements SurfaceHolder.Callback{

	public GamePanel(Context context) {
		super(context);
		// TODO Auto-generated constructor stub
	}

	@Override
	protected void onDraw(Canvas canvas) {
		// TODO Auto-generated method stub
		super.onDraw(canvas);
	}

	public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2, int arg3) {
		// TODO Auto-generated method stub

	}

	public void surfaceCreated(SurfaceHolder arg0) {
		// TODO Auto-generated method stub

	}
```

```
    public void surfaceDestroyed(SurfaceHolder arg0) {
        // TODO Auto-generated method stub

    }
}
```

4. Trong hàm tạo của GamePanel cần bổ sung thêm:

```
    public GamePanel(Context context) {
        super(context);
        // TODO Auto-generated constructor stub
        getHolder().addCallback(this);
        setFocusable(true);
    }
```

getHolder().addCallback(this) dùng để gán lớp hiện tại (GamePanel) như là một hanlder cho các sự kiện xảy ra trên bề mặt vẽ.

setFocusable(true) làm cho GamePanel có thể focus. Nghĩa là nó có thể truy xuất focus do đó nó có thể nắm bắt và xử lý sự kiện.

5. Giờ ta sẽ gán bề mặt vẽ "GamePanel" lên Activity. Mở file java chính (tên CautruccobanActivity.java) và sửa hàm onCreate lại như sau:

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

        //cua so khong co thanh title
        requestWindowFeature(Window.FEATURE_NO_TITLE);

        //tao doi tuong
        GamePanel d=new GamePanel(this);
        setContentView(d);
        //setContentView(R.layout.main);
    }
```

Chú ý: hàm setContentView bây giờ không truyền vào 1 layout dạng XML nữa mà là một GamePanel.

6. Sau hàm requestWindowFeature có thể bổ sung thêm để fullscreen như sau:

```
        //cua so khong co thanh title
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        //full man hinh
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);
```

Phần 2: Tạo game loop

1. Tiếp theo ta sẽ tạo ra một thread mà nó sẽ là game loop . Tạo một class mới tên MainThread kế thừa từ thread. Trong class vừa tạo tạo ra một biến tên "running" để lưu trạng thái của thread. True là đang chạy. Viết một hàm setRunning để gán lại biến "running". Tiếp theo override lên hàm run.

```
public class MainThread extends Thread{
```

```
    private boolean running;

    public void setRunning(boolean r)
    {
            running=r;

    }

    @Override
    public void run() {
            // TODO Auto-generated method stub
            super.run();

            while(running)
            {
                    ////// cap nhat lai trang thai game
                    /////  render du lieu ra man hinh
            }
    }
}
```

Trong hàm run. Trong khi biến cờ "running" còn true sẽ thực hiện một vòng lặp vĩnh viễn. Trong vòng lặp này ta sẽ cập nhật lại trạng thái game cũng như render dữ liệu ra màn hình.

2. Hiện tại Thread vừa tạo chưa được chạy, ta sẽ cho nó bắt đầu chạy khi màn hình vừa được load lên. Mở file "GamePanel.java".
    a. Khai báo 1 đối tượng tên "thread" thuộc lớp "MainThread".
    b. Trong hàm tạo khởi tạo đối tượng "thread".
    c. Trong hàm surfaceCreated cho thread chạy.
    d. Trong hàm surfaceDestroyed thì hủy thread.

```
public class GamePanel extends SurfaceView implements SurfaceHolder.Callback{

    private MainThread thread; //a. khai bao bien thread

    public GamePanel(Context context) {
            super(context);
            getHolder().addCallback(this);
            thread =new MainThread(); //b. khoi tao bien thread
            setFocusable(true);
    }

    @Override
    protected void onDraw(Canvas canvas) {
            super.onDraw(canvas);
    }

    public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2, int arg3) {
            // TODO Auto-generated method stub

    }

    public void surfaceCreated(SurfaceHolder arg0) {
            //c. gán trạng thái cho thread và kích cho thread chay
            thread.setRunning(true);
            thread.start();
    }
```

```
    public void surfaceDestroyed(SurfaceHolder arg0) {
        //d. huy thread
        if(thread.isAlive())
            thread.setRunning(false);
        }
}
```

Phương thức surfaceDestroyed được gọi trực tiếp trước khi surface bị hủy. Trong hàm ta sẽ xét xem luồng MainThread còn sống không, nếu còn sống ta sẽ gọi hàm setRunning và gán về false.

Phần 3: Test game loop

1. Mở file Mainthread.java trong hàm run viết như sau:

```
    @Override
    public void run() {
        // TODO Auto-generated method stub
        long dem=0L;
        super.run();
        while(running)
        {
            dem++;
            ////// cap nhat lai trang thai game
            /////  render du lieu ra man hinh
            Log.d("testloop:", "loop "+ dem);
        }
    }
```

Trong hàm run ta tạo ra biến đếm và trong vòng lặp while ta cho biến đếm tăng lên rồi dùng lệnh Log.d để xuất ra Logcat.

2. Tương tự bên file "cautrucobanActivity.java" trong hàm onDestroy ta cũng xuất ra logcat để test khi kết thúc chương trình.

```
    protected void onDestroy() {
        // TODO Auto-generated method stub

        super.onDestroy();
        Log.d("looptest", "huy thread");

    }
```

3. Chạy và xem kết quả trên Logcat bằng cách Trên Eclipse chọn Window -> Show View -> LogCat để mở hộp thoại LogCat. Chạy chương trình và xem kết quả loop liên tục trên logcat. Tắt ct để thấy onDestroy được chạy.