Canvas với

Phần 1 (khá giống bài trước)

1. Tạo project.
2. Tạo 1 class mới tên “Draw2d.java” extends từ view, viết hàm tạo và override lên hàm onDraw

```
public class Draw2d extends View{

	public Draw2d(Context context) {
		super(context);
		// TODO Auto-generated constructor stub
	}
	@Override
	protected void onDraw(Canvas canvas) {
		// TODO Auto-generated method stub
		super.onDraw(canvas);
	}
}
```

3. Quay về file java chính, tạo đối tượng và sửa lại setContentView

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

		//tao doi tuong
		Draw2d d=new Draw2d(this);
		setContentView(d);
```

4. Trong file Draw2d.java tạo ra 1 biến bitmap, trong hàm tạo lấy tài nguyên cho bitmap như sau:

```
	Bitmap mybitmap;
	public Draw2d(Context context) {
		super(context);
		// TODO Auto-generated constructor stub
		mybitmap=BitmapFactory.decodeResource(getResources(), R.drawable.ic_launcher);
	}
```

5. Trong onDraw vẽ hình lên như sau:

```
	@Override
	protected void onDraw(Canvas canvas) {
		// TODO Auto-generated method stub
		super.onDraw(canvas);
		canvas.drawColor(Color.BLACK);
		canvas.drawBitmap(mybitmap,10,10, null);

	}
```

6. Chạy chương trình và thấy trên vùng vẽ (canvas) có 1 hình được vẽ (đến bước này thấy vẫn còn giống ý tưởng từ bài trước).

Phần 2:

7. Ta sẽ thay đổi để lớp Draw2d kế thừa từ lớp SurfaceView thay cho lớp View. Lớp SurfaceView cho phép ta vẽ mọi thứ trên vùng vẽ mà không cần layout và file xml. Nó còn giúp ta tùy chọn hoạt hình tốt hơn.

Thay đổi lớp Draw2d để kế thừa từ lớp SurfaceView.

```
public class Draw2d extends SurfaceView
```

8. Chạy chương trình và ta không còn thấy hình nữa do hàm onDraw không còn được chạy nữa. Để SurfaceView cập nhập và vẽ lại liên tục ta cần phải implement một view thead mà nó sẽ điều khiển việc gọi hàm vẽ của ta. Có thể làm bằng cách khác nhưng dùng với Theard ta có nhiều điều khiển hơn.
   Để đơn giản ta sẽ implement từ interface tên SurfaceHolder.Callback và có 3 phương thức ta cần overrider

```
public class Draw2d extends SurfaceView implements SurfaceHolder.Callback
```

Và override lên 3 hàm

```
    public void surfaceChanged(SurfaceHolder holder, int format, int width,
                int height) {
          // TODO Auto-generated method stub

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

    }
    public void surfaceDestroyed(SurfaceHolder holder) {
          // TODO Auto-generated method stub

       }
```

9. Để đăng kí class có thể callback ta khai báo thêm trong hàm tạo

```
          getHolder().addCallback(this);
```

10. Sửa lại hàm onDraw, bỏ dòng @override ở trên hàm onDraw và sử tên hàm lại thành doDraw. Thực hiện điều này để ta kiểm soát hoàn toàn việc vẽ.

```
    protected void doDraw(Canvas canvas) {
          // TODO Auto-generated method stub
          super.onDraw(canvas);
          canvas.drawColor(Color.BLACK);
          canvas.drawBitmap(mybitmap,10,10, null);

    }
```

11. Xây dựng 1 class mới tên ViewThread kế thừa từ Thread để điều khiển.

```
public class ViewThread extends Thread{

    private Draw2d mdraw2d;
    private SurfaceHolder mHolder;
    private boolean mRun=false;

    public ViewThread(Draw2d d)
    {
          mdraw2d=d;
          mHolder=mdraw2d.getHolder();
    }
    public void setRunning(boolean r)
```

```
    {
        mRun=r;
    }
    @Override
    public void run() {
        // TODO Auto-generated method stub
        super.run();
        Canvas canvas=null;
        while(mRun)
        {
            canvas=mHolder.lockCanvas();
            if(canvas!=null)
            {
                mdraw2d.doDraw(canvas);
                mHolder.unlockCanvasAndPost(canvas);
            }
        }

    }
}
```

Trong class này ta có hàm tạo nhận vào 1 đối tượng Draw2d, biến mHolder để nắm điều khiển của Draw2d.

Ta có hàm setRunning để gán trạng thái đang chạy hay ngủm rùi của Thread.

Hàm run là quan trọng nhất. Nó sẽ xét trong khi còn chạy thì nó sẽ gọi hàm lockCanvas(). Hàm này sẽ cho phép bắt đầu sửa trên canvas, nó sẽ trả về canvas để sẵn sàng vẽ, nếu trả về null nếu surface không tạo và sửa được.

Khi canvas sẵn sàng nó gọi hàm doDraw để vẽ (hàm doDraw do ta sử tên của hàm onDraw mà ra) sau đó nó gọi hàm unlock để kết thúc việc vẽ trên Canvas và xuất ra màn hình.

12. Quay về file Draw2d.java ta khai báo 1 biến ViewThread.

```
    private ViewThread mthread;
```

13. Trong hàm tạo khởi tạo đối tượng ViewThread.

```
        mthread=new ViewThread(this);
```

14. Trong hàm surfaceCreated (hàm này chạy khi surface được tạo ra) xét nếu luồn chưa sống thì tạo nó, set cho nó trang thái chạy là true và bắt đầu chạy nó. Nếu nó có rồi thì thôi. Nếu không làm điều này thì khi ta tạo ra một instance mới và chạy thread thì ta không thể sử dụng lại đối tượng theard đang có bởi vì tại thời điểm đó chúng ta không biết chắc thread đã được chạy hay chưa và một theard instance không thể chạy nhiều hơn 1 lần
    Trong hàm surfaceDestroyed xét nó đang sống thì gán trạng thái đang chạy lại thành false. Vì khi Activity nó chuyển sang trạng thái nền thì ta cũng cho theard kết thúc bằng cách dừng vòng lặp.

```
    public void surfaceCreated(SurfaceHolder holder) {
        // TODO Auto-generated method stub
        if(!mthread.isAlive())
        {
            mthread=new ViewThread(this);
            mthread.setRunning(true);
            mthread.start();
        }
```

```
    }
    public void surfaceDestroyed(SurfaceHolder holder) {
           // TODO Auto-generated method stub
           if(mthread.isAlive())
                  mthread.setRunning(false);

    }
```

15. Chạy chương trình và ta vẫn thấy chỉ có 1 hình được vẽ ra giống phần trước. Nhưng giờ ta đã dùng đến khái niệm 2Dgraphic.