Bài 1 - Hiện hình lên vùng canvas- tại vị trí touch

Phát triển tiếp từ bài 0

Phần 1: Truyền GamPanel cho MainThread.

Trong Mainthread ta cần phải có GamePanel để điều khiển và truy xuất cũng như khóa vùng vẽ do đó ta cần phải truyền gamepanel vào cho Mainthread.

1. Trong file "MainThread.java" ta thêm 2 biến và xây hàm tạo như sau:

```
    private SurfaceHolder surfaceholder;
    private GamePanel gamepanel;
    private boolean running;

    public MainThread(SurfaceHolder surfaceholder, GamePanel gamepanel)
    {
           this.surfaceholder=surfaceholder;
           this.gamepanel=gamepanel;
    }
```

File MainThread cần phải có gamePanel để để có thể khóa surface khi vẽ lên, điều đó cần phải thông qua surfaceholder. Thật ra ta có thể không cần surfaceholder truyền vào vì khi có gamepanel ta vẫn có thể dùng gamepanel.getHolder() để truy đến surfaceholder của nó.

2. Do đã sửa hàm tạo của MainThread nên ta phải quay về GamePanel.java nơi gọi MainThread để sửa lại. Mở GamePanel.java ngay tại hàm tạo khi khởi tạo MainThread sửa lại như sau:

```
           thread =new MainThread(getHolder(),this); //b. khoi tao bien thread
```

Phần 2: Vẽ hình ảnh lên vùng canvas.

1. Mở file GamePanel.java. Tạo đối tượng Bitmap toàn cục, trong hàm tạo dùng BitmapFactory để lấy tài nguyên hình.

```
    Bitmap bitmap;
    public GamePanel(Context context) {
           super(context);
           getHolder().addCallback(this);
           thread =new MainThread(getHolder(),this); //b. khoi tao bien thread
           setFocusable(true);
           bitmap=BitmapFactory.decodeResource(this.getResources(),R.drawable.ic_launcher);
    }
```

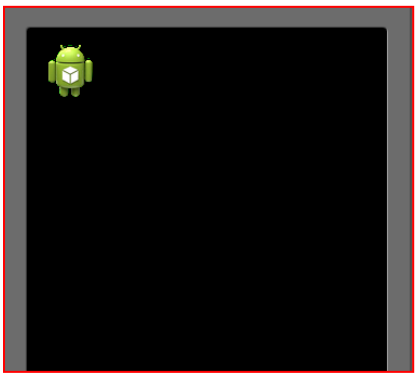
2. Trong hàm onDraw dùng drawBitmap để vẽ lên vùng canvas

```
    protected void onDraw(Canvas canvas) {
           super.onDraw(canvas);
           canvas.drawBitmap(bitmap, 10, 10, null);
    }
```

3. Mở file MainThread.java. Sửa lại hàm run gọi onDraw của MainThread.

```
public void run() {
	// TODO Auto-generated method stub
	super.run();
	long dem=0L;
	Canvas canvas=null;
	while(running)
	{
		//1.cap nhat lai trang thai game //2.render du lieu ra man hinh
		canvas=surfaceholder.lockCanvas();
		if(canvas!=null)
		{
			gamepanel.onDraw(canvas);
			surfaceholder.unlockCanvasAndPost(canvas);
		}
		Log.d("testloop", "loop "+ (dem++));
	}
}
```

4. Chạy chương trình và thấy hình được vẽ ra màn hình.



Phần 3: Hiển thị hình tại vị trí chạm

Phần trên ta đã hiển thị hình tại tọa độ (10,10) . Tiếp theo ta sẽ sửa lại để hình sẽ được vẽ tại vị trí ta chạm tay vào.

1. Trong file "GamePanel.java" ta cần khai báo 2 biến để lưu trữ lại tọa độ khi ta chạm tay vào.

```
int mX;//toa do X khi cham tay
int mY; //toa do Y khi cham tay
```

2. Trong hàm onDraw ta sửa lại để vẽ hình tại tọa độ mX,mY

```
protected void onDraw(Canvas canvas) {
	super.onDraw(canvas);
	canvas.drawBitmap(bitmap, mX, mY, null);
}
```

3. Vậy tọa độ mX, mY này ở đâu ra, đó chính là tọa độ khi ta chạm tay vào. Ta sẽ override lên hàm onTouchEvent. Đối tượng event sẽ có hàm getX và getY cho ta lấy tọa độ vị trí được chạm

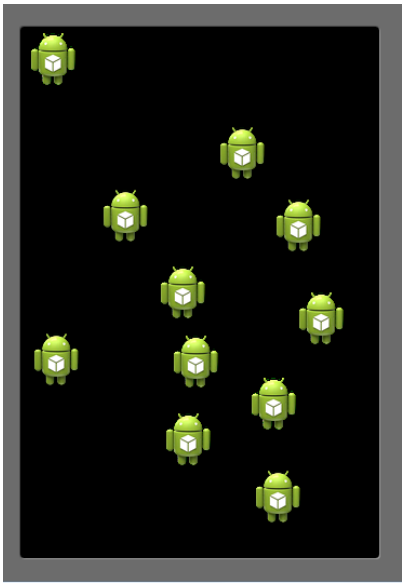
   Chú ý: hàm onTouchEvent thuộc View do đó khi dùng tool để override phải kéo xuống vùng View chứ không phải SurfaceView

```
    @Override
    public boolean onTouchEvent(MotionEvent event) {
           // TODO Auto-generated method stub
           mX=(int)event.getX();
           mY=(int)event.getY();
           return super.onTouchEvent(event);
    }
```

4. Chạy chương trình để thấy chạm vị trí nào thì hình sẽ hiện ra ở vị trí đó. Nhưng lưu ý rằng tọa độ luôn tính góc trên bên trái nên khi ta chạm thì nó sẽ tính vị trí chạm và vẽ hình từ góc trên bên trái vị trí chạm. Muốn vẽ hình ngay tâm của vị trí chạm ta sửa lại 1 tí.

```
    @Override
    public boolean onTouchEvent(MotionEvent event) {
           // TODO Auto-generated method stub
           mX=(int)event.getX() - mybitmap.getWidth()/2;
           mY=(int)event.getY() - mybitmap.getHeight()/2;
           return super.onTouchEvent(event);
    }
```

5. Chạy chương trình để thấy ta chạm vị trí nào sẽ vẽ hình có tâm ngay tại vị trí ta chạm.



6. Nhưng ta thấy rằng ta chạm vào nhiều vị trí thì hình của ta vẫn giữ lại do vẫn vẽ tiếp lên vùng vẽ trước đó. Do đó để vẽ lại ta phải vẽ lại toàn bộ. Ở đây đơn giản ta chỉ cần vẽ canvas lại với vùng màu đen. Trong hàm onDraw trước khi vẽ bitmap ta vẽ vùng canvas với một nền đen.

```
    protected void onDraw(Canvas canvas) {
           super.onDraw(canvas);
           canvas.drawColor(Color.BLACK);
           canvas.drawBitmap(bitmap, mX, mY, null);

    }
```

7. Chạy lại để thấy kết quả ta chạm vị trí nào thì hình sẽ được vẽ ở vị trí đó (hình cũ mất đi). Giả sử muốn giữ lại nhiều hình ở nhiều lần chạm ta sẽ làm bằng phương pháp khác.

Phần 4: Đưa việc xử lý hình ảnh thành đối tượng.

Class GamePanel chỉ nên quản lý chung việc vẽ chứ không nên dùng để vẽ. Ta nên xây dựng thành class riêng.

1. Các bước ở phần 2 và 3 cần phải sửa lại để mã lệnh quay về như phần 1. Trong class GamePanel:
   a. Bỏ 3 biến bitmap, mX, mY trong phần khai báo biến.
   b. Trong hàm tạo bỏ lệnh khởi tạo cho biến bitmap.
   c. Trong hàm onDraw bỏ lệnh canvas.drawBitmap.
   d. Trong hàm onTouchEvent bỏ 2 lệnh lấy tọa độ mX, mY.
2. Giờ ta sẽ tách phần bitmap ra thành 1 class riêng. Tạo 1 class mới có tên Element.java và viết mã như sau:

```
public class Element {

	Bitmap bitmap; //hinh anh
	int mX; //toa do x
	int mY; //toa do y

	public Element(Resources res, int x,int y)
	{
		bitmap=BitmapFactory.decodeResource(res,R.drawable.ic_launcher);
		mX=x-bitmap.getWidth()/2;
		mY=y-bitmap.getHeight()/2;
	}
	public Element(Resources res, int x, int y, int idHinh)
	{
		bitmap=BitmapFactory.decodeResource(res,idHinh);
		mX=x-bitmap.getWidth()/2;
		mY=y-bitmap.getHeight()/2;
	}
	public void doDraw(Canvas canvas)
	{
		canvas.drawBitmap(bitmap, mX,mY, null);
	}
}
```

Ta thấy rằng mã lệnh của class thật ra rất giống phần ta đã bỏ nhưng được tách ra để xây thành đối tượng. Ở đây ta viết hàm doDraw đảm trách việc vẽ đối tượng do đó hàm onDraw ở GamePanel giờ chỉ quản lý việc vẽ chứ không trực tiếp vẽ nữa.

3. Quay về GamePanel.java để gọi đối tượng Element và vẽ. Đầu tiên (trong GamePanel nhé) khai báo biến toàn cục Element.

```
    Element myelement;
```

4. Trong hàm onTouchEvent khởi tạo đối tượng myelement và truyền vào tọa độ ngay tại vị trí chạm để đối tượng element vẽ ngay tại vị trí đó.

```
    @Override
    public boolean onTouchEvent(MotionEvent event) {
	// TODO Auto-generated method stub

	myelement=new Element(getResources(),(int)event.getX(),(int)event.getY());
	return super.onTouchEvent(event);
    }
```

5. Trong hàm onDraw ta không chủ động vẽ mà kêu hàm vẽ của myelement. Nhớ phải xét xem myelement đã được khởi tạo hay chưa, vì trong MainThread thì hàm onDraw được gọi liên tục từ khi bắt đầu. Những khi bắt đầu ta chưa chạm nên myelement chưa có thì lấy gì mà vẽ.

```
    protected void onDraw(Canvas canvas) {
            super.onDraw(canvas);
            canvas.drawColor(Color.BLACK);

            if(myelement!=null)
                    myelement.doDraw(canvas);
    }
```

6. Bắt sự kiện vẽ khi chạm tay. Trong hàm onTouch viết như sau:

```
    public boolean onTouchEvent(MotionEvent event) {
            // TODO Auto-generated method stub
            if(myelement==null)
            {
                    myelement=new Element(getResources(),(int)event.getX(),(int)event.getY());
                    Log.d("abc","khoi tao dau tien");
                    return true;
            }
            else
            {
                    myelement.mX=(int)event.getX()-myelement.bitmap.getWidth()/2;
                    myelement.mY=(int)event.getY()-myelement.bitmap.getHeight()/2;
            }

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

Trong hàm onTouch này lần đầu tiên (chưa có đối tượng myelement) thì khởi tạo, nếu có rồi thì cập nhật lại tọa độ cho myelement.

Chú ý: return phải trả về true chứ không gọi super

7. Hàm onTouch đã bao gồm các trạng thái down (nhấn xuống), up (nhả ra), nhấn và kéo (move) nên ta chạy và thấy rằng không chỉ chạm chổ nào hình được vẽ chổ đó mà khi nhấn và di chuyển hình sẽ di chuyển theo luôn. Ta có thể kĩ hơn viết riêng từng trường hợp để dễ tùy biến về sau cho từng trang thái như sau:

```
public boolean onTouchEvent(MotionEvent event) {
            // TODO Auto-generated method stub
            if(myelement==null)
            {
                    myelement=new Element(getResources(),(int)event.getX(),(int)event.getY());
                    Log.d("abc","khoi tao dau tien");
                    return true;
            }
            else
            {
                    myelement.mX=(int)event.getX()-myelement.bitmap.getWidth()/2;
                    myelement.mY=(int)event.getY()-myelement.bitmap.getHeight()/2;
            }

            if(event.getAction()==MotionEvent.ACTION_DOWN)
            {
                    myelement.mX=(int)event.getX()-myelement.bitmap.getWidth()/2;
```

```
                myelement.mY=(int)event.getY()-myelement.bitmap.getHeight()/2;
                Log.d("abc","ddddddddddddddddddddddddddddown");
        }
        if(event.getAction()==MotionEvent.ACTION_UP)
        {
                myelement.mX=(int)event.getX()-myelement.bitmap.getWidth()/2;
                myelement.mY=(int)event.getY()-myelement.bitmap.getHeight()/2;
                Log.d("abc","uuuuuuuuuuuuuuuuuuuuuuuuuuuup");
        }
        if(event.getAction()==MotionEvent.ACTION_MOVE)
        {
                myelement.mX=(int)event.getX()-myelement.bitmap.getWidth()/2;
                myelement.mY=(int)event.getY()-myelement.bitmap.getHeight()/2;
                Log.d("abc","mmmmmmmmmmmmmmmmmmmmmmmmmmmmmove");
        }

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

8. Chạy ct để thấy kết quả, touch ở đâu hình được vẽ ở đó, có thể touch và kéo để di chuyển đối tượng hình.