Thực hành

# Viết chương trình vẽ khi touch.

1. Tạo ra một class mới tên SingleTouch và cho lớp này kế thừa từ View, sau đó import class và phát sinh hàm tạo.

**public** **class** SingleTouch **extends** View{

**public** SingleTouch(Context context, AttributeSet attrs) {

**super**(context, attrs);

// **TODO** Auto-generated constructor stub

}

}

1. Trong class vừa tạo khai báo 2 biến toàn cục là Paint (bút vẽ) và Path (đường vẽ)

Paint paint=**new** Paint();

Path path=**new** Path();

1. Trong hàm tạo khởi tạo giá trị cho biến Paint như sau:

paint.setAntiAlias(**true**);//khu rang cua

paint.setStrokeWidth(6f);//do day

paint.setColor(Color.*RED*);//gan mau

paint.setStyle(Paint.Style.*STROKE*);

paint.setStrokeJoin(Paint.Join.*ROUND*);

1. Override lên hàm onDraw để vẽ.

@Override

**protected** **void** onDraw(Canvas canvas) {

// **TODO** Auto-generated method stub

**super**.onDraw(canvas);

canvas.drawPath(path, paint);

}

1. Override lên hàm onTouchEvent. Trong hàm đầu tiên lấy ra tọa độ X,Y chính là vị trí chạm. Sau đó xét nếu chạm tay (down) thì nhảy paint đến vị trí đó, nếu di chuyển (move) vẽ từ vị trí trước đến vị trí move, nhả tay ra (up) thì không làm gì nữa. Cuối cùng gọi invalidate() để vẽ lại (chú ý hàm invalidate và postInvalidate). Cuối cùng nhớ trả về true (bỏ trả về mặc định đi).

@Override

**public** **boolean** onTouchEvent(MotionEvent event) {

// **TODO** Auto-generated method stub

**float** eventX=event.getX();

**float** eventY=event.getY();

**switch**(event.getAction())

{

**case** MotionEvent.*ACTION\_DOWN*:

path.moveTo(eventX, eventY);

**break**;

**case** MotionEvent.*ACTION\_MOVE*:

path.lineTo(eventX,eventY);

**break**;

**case** MotionEvent.*ACTION\_UP*:

**break**;

**default**: **return** **false**;

}

invalidate();//ve lai view

//neu ve lai view tu UI-thread dung invalidate()

//neu ve lai tu non-UI Thread dung view.postInvalidate()

**return** **true**;

//return super.onTouchEvent(event);

}

1. Quay lại file java chính. Trong hàm onCreate sửa lại hàm setContextView như sau:

@Override

**protected** **void** onCreate(Bundle savedInstanceState) {

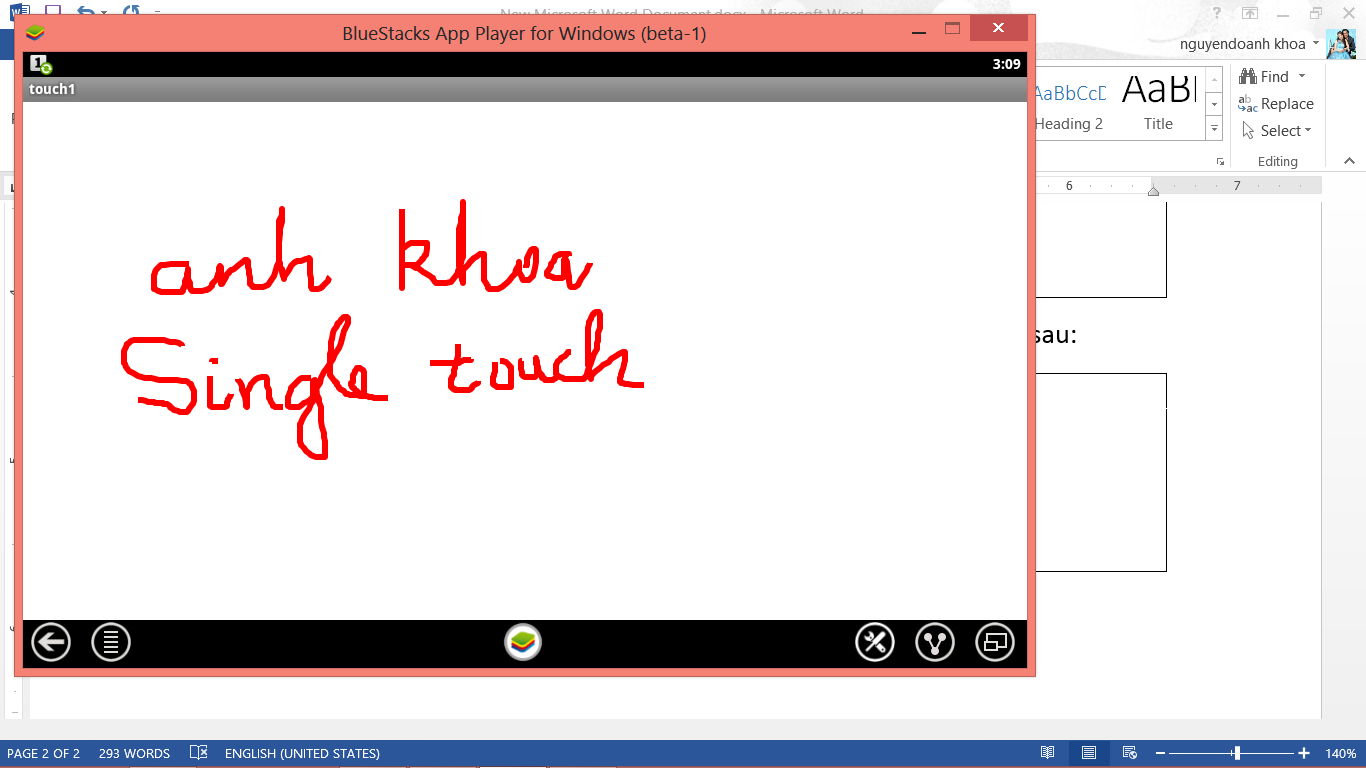
**super**.onCreate(savedInstanceState);

//setContentView(R.layout.activity\_main);

setContentView(**new** SingleTouch(**this**,**null**));

}

1. Chạy chương trình và xem kết quả.



# Viết chương trình vẽ và duy chuyển vòng tròn khi chạm các ngón tay (Multitouch)

Tương tự như bài 1 nhưng sửa lại lớp SingleTouch như sau:

1. Khai báo các biến toàn cục gồm hai arraylist để chứa tập tọa độ x,y của các ngón tay khi chạm và một arraylist istouch để chứa trạng thái touch (thật ra arraylist này không cần nhưng để an toàn thôi).

Paint paint=**new** Paint();

ArrayList<Float> x=**new** ArrayList<Float>();

ArrayList<Float> y=**new** ArrayList<Float>();

ArrayList<Boolean> istouch=**new** ArrayList<Boolean>();

1. Mệt nhất là hàm onTouchEvent (xem thêm ghi chú bên trong hàm)

// ta phai chu ý các diem sau:

// 1.lop MotionEventCompat nhung thu vien ho tro vao la

// import android.support.v4.view.MotionEventCompat;

// 2. de tien ta dung .getActionIndex de lay index cua su kien

// index nay chinh la index cua cac ngon tay, vd: cham 1 ngon

// ta co index=0, cham them cai nua thi index=1.....

// 3. moi lan ta cham vao ta se lay du lieu gom toado x, y va

// mang istouch, khi di chuyen ta phai duyet tat ca cac index (cac

// diem cham va cap nhat lai toa do cho cac diem cham.

// 4. su kien up rat ac, moi lan xong thi no lai cap nhat lai index

// nen index duoc cap nhat lien tuc neu ta bo ngon tay ra. Tui da

// dung array va bi pha san phai sua thanh ArrayList de co the de

// dang remove 1 phan tu o giua.

@Override

**public** **boolean** onTouchEvent(MotionEvent event) {

// **TODO** Auto-generated method stub

**int** action = MotionEventCompat.*getActionMasked*(event);

**int** index = MotionEventCompat.*getActionIndex*(event);

//int actionpointid=event.getPointerId(index);

**switch**(action)

{

**case** MotionEvent.*ACTION\_DOWN*:

istouch.add(**true**);

x.add(event.getX(index));

y.add(event.getY(index));

**break**;

**case** MotionEvent.*ACTION\_POINTER\_DOWN*:

istouch.add(**true**);

x.add(event.getX(index));

y.add(event.getY(index));

**break**;

**case** MotionEvent.*ACTION\_MOVE*:

**int** count=event.getPointerCount();

**for**(**int** i=0;i<count;i++)

{

index=i;

x.set(index,event.getX(index));

y.set(index,event.getY(index));

}

**break**;

**case** MotionEvent.*ACTION\_UP*:

istouch.remove(index);

x.remove(index);

y.remove(index);

**break**;

**case** MotionEvent.*ACTION\_POINTER\_UP*:

istouch.remove(index);

x.remove(index);

y.remove(index);

**break**;

**default**: **return** **false**;

}

invalidate();

// bao co thay doi va yeu cau ve lai view

//neu ve lai view tu UI-thread dung invalidate()

//neu ve lai tu non-UI Thread dung view.postInvalidate()

**return** **true**;

//return super.onTouchEvent(event);

}

1. Cuối cùng là hàm onDraw. Hàm này được chạy liên tục mỗi lần chạy sẽ duyệt tất cả các vị trí chạm đã được lưu trữ và vẽ hình tròn tại vị trí đó.

@Override

**protected** **void** onDraw(Canvas canvas) {

// **TODO** Auto-generated method stub

**super**.onDraw(canvas);

**for**(**int** i=0;i<x.size();i++)

**if**(istouch.get(i)==**true**)

canvas.drawCircle(x.get(i), y.get(i),50f, paint);

}

# Viết chương trình Zoom hình ảnh bằng đa chạm

1. Tạo project mới. Trong project tạo một class mới tên MyImageZoom kế thừa từ view. Thêm một hàm tạo và khai báo các biến gồm một Drawable để chứa hình, một biến float tên tile để lưu tỉ lệ zoom, một đối tượng tên scale thuộc lớp ScaleGestureDetector lớp này hỗ trợ cho việc co giản kích thước (zoom).

**public** **class** MyImageZoom **extends** View{

Drawable hinh;

**float** tile=1.0f;

ScaleGestureDetector scale;

**public** MyImageZoom(Context context) {

**super**(context);

// **TODO** Auto-generated constructor stub

}

}

1. Trong hàm tạo gắn biến hinh là một hình lấy từ tài nguyên resouce, ta lấy mặc định một hình là icon (sẽ viết hàm cập nhật hình sau). Dùng hàm setBounds để gán biến độ left top là 0 0 để gán góc trên bên trái còn right và bottom bằng chiều ngang và chiều cao của hình (ta có thể dùng nó để định lại kích thước hình nếu muốn). Sau đó gắn sự kiện cho đối tượng ScaleGestureDetector là sukienscale , ta sẽ viết class sukienscale sau.

**public** MyImageZoom(Context context) {

**super**(context);

// **TODO** Auto-generated constructor stub

hinh=context.getResources().getDrawable(R.drawable.*ic\_launcher*);

setFocusable(**true**);

//left,top,right,bottom

hinh.setBounds(0, 0, hinh.getIntrinsicWidth(), hinh.getIntrinsicHeight());;

scale=**new** ScaleGestureDetector(context,**new** sukienscale());

}

1. Viết thêm một hàm để gán lại hình. Trong hàm khi gán hình mới ta cũng cần gọi lại setBounds để cập nhật kích thước theo hình mới.

**public** **void** setImage(**int** id)

{

hinh=getResources().getDrawable(id);

hinh.setBounds(0, 0, hinh.getIntrinsicWidth(), hinh.getIntrinsicHeight());;

}

1. Tiếp theo ta xây dựng một class con tên sukienscale kế thừ từ SimpleOnScaleGestureListener (Class này sẽ hỗ trợ cho việc co giãn). Bên trong ta override lên hàm onScale tức là khi dùng ngón tay co giản thì sự kiện này xảy ra. Nó được nhận vào một detector.

Dùng detector.getScaleFactor để lấy về tỉ lệ co giản từ scale event trước đó đến scale event hiện tại. Nó tương tự như ([getCurrentSpan()](file:///C:\Program%20Files%20(x86)\Android\android-sdk\docs\reference\android\view\ScaleGestureDetector.html#getCurrentSpan()) / [getPreviousSpan()](file:///C:\Program%20Files%20(x86)\Android\android-sdk\docs\reference\android\view\ScaleGestureDetector.html#getPreviousSpan())). Nếu 2 ngón tay co lại nó sẽ trả về số <0 (vd: 0.99343) nếu 2 ngón tay giãn ra nó sẽ trả về số >1 (vd:1.00434). lấy biến tile nhân số này và gán vào biến tile ta sẽ có số tile nhỏ dần nếu co lại và lớn dần nếu giản ra.

Để khống chế giá trị lớn và nhỏ nhất của biến tỉ lệ đầu tiên ta lấy min giữa tile và 5.0 nên sẽ không vượt 5.0 sau đó lấy max 0.1 với giá trị đó nên sẽ không nhỏ hơn 0.5. Cách viết này giống với

|  |
| --- |
| **if**(tile>=5.0f) tile=5.0f;  **if**(tile<=0.1f) tile=0.1f; |

Nhưng dùng như mã bên dưới sẽ nghệ thuật hơn ;-) . Sau khi có tile xong ta goi hàm để vẽ lại.

**class** sukienscale **extends** SimpleOnScaleGestureListener{

@Override

**public** **boolean** onScale(ScaleGestureDetector detector) {

// **TODO** Auto-generated method stub

tile=tile\*detector.getScaleFactor();

Log.*d*("tile", detector.getScaleFactor()+" "+ tile);

tile=Math.*max*(0.1f,Math.*min*(tile,5.0f));

invalidate();

**return** **true**;

//return super.onScale(detector);

}

}

1. Tiếp theo ta override lên hàm onDraw để làm việc vẽ lại. Ở đây có 2 hàm là save và restore để lưu và phục hồi canvas vào stack (ở đây thật sự không cần nhưng cứ dùng). Chủ yếu là hàm scale để co kích thước canvas rồi gọi draw để vẽ hình lên canvas. Hàm scale nhận vào tỉ lệ co giản ở đây ta truyền biến tile, và biến này đã được class sự kiện ở trên cập nhật dựa vào 2 ngón tay co giãn.

@Override

**protected** **void** onDraw(Canvas canvas) {

// **TODO** Auto-generated method stub

**super**.onDraw(canvas);

canvas.save();//luu lai gioi han hien tai

canvas.scale(tile,tile);

hinh.draw(canvas);

canvas.restore();//phuc hoi lai trang thai

}

1. Tiếp theo ta override lên hàm onTouchEvent để bắt sự kiện khi touch, trong hàm này ta gọi tiếp onTouchEvent của đối tượng ScaleGestureDetector để chuyển tiếp sự kiện qua cho sự kiện của đối tượng scale.

@Override

**public** **boolean** onTouchEvent(MotionEvent event) {

// **TODO** Auto-generated method stub

scale.onTouchEvent(event);

invalidate();

**return** **true**;

//return super.onTouchEvent(event);

}

1. Cuối cùng quay về chương trình chính và sửa hàm onCreate như sau (chú ý thay hình cho phù hợp):

@Override

**protected** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

MyImageZoom i=**new** MyImageZoom(**this**);

i.setImage(R.drawable.*apple*);

setContentView(i);

//setContentView(R.layout.activity\_main);

}

1. Chạy chương trình trên máy thật để thấy kết quả của việc zoom.

# Move hình ảnh (phát triển từ bài zoom lên)

1. Thực hiện tiếp từ bài trên (bài zoom nhé). Trong class MyImageZoom ta khai báo thêm các biến toàn cục như sau:

**float** x,y;//luu x,y cua event

**float** last\_touch\_x,last\_touch\_y;//luu vi tri down truoc do

**float** move\_x,move\_y;

1. Trong hàm onTouchEvent, sau sự kiện scale ta bổ xung thêm phần bắt các hành động down, move như sau:

……..

scale.onTouchEvent(event);

**int** action=event.getAction();

**switch** (action)

{

**case** MotionEvent.*ACTION\_DOWN*:

x=event.getX();

y=event.getY();

last\_touch\_x=x;

last\_touch\_y=y;

**break**;

**case** MotionEvent.*ACTION\_MOVE*:

x=event.getX();

y=event.getY();

**if**(!scale.isInProgress())//neu khong phai la scale

{

**float** lech\_x=x-last\_touch\_x;

**float** lech\_y=y-last\_touch\_y;

move\_x+=lech\_x;

move\_y+=lech\_y;

}

last\_touch\_x=x;

last\_touch\_y=y;

**break**;

**case** MotionEvent.*ACTION\_UP*:

**break**;

**default**:

**break**;

}

……………..

1. Trong hàm onDraw sau lệnh canvas.scale ta thêm lệnh translate như code bên dưới, Chú ý: hàm translate khi di chuyển đối tượng còn phải phụ thuộc vào tỉ lệ tile nếu không chia cho tile nếu chưa zoom ta move chính xác nhưng zoom lên ta move sẽ bị sai:

…………………………………………

canvas.scale(tile,tile);

canvas.translate(move\_x/tile, move\_y/tile);

hinh.draw(canvas);

…………………………………………

1. Chạy chương trình để thấy zoom và move đối tượng đã được thực hiện tốt.

Chú ý:

Bài lab trên chỉ mang tính học tập. Muốn tốt hơn ta nên sử dụng thư viện có sẵn tên TouchImageView.

Các bạn có thể download tại địa chỉ bên dưới

<https://github.com/MikeOrtiz/TouchImageView>