LẬP TRÌNH ANDROID

Giảng Viên: Phùng Mạnh Dương

Trường ĐH Công nghệ, ĐHQGHN

CHƯƠNG 8
Graphics, Touch, Gesture và Multimedia

Vẽ đồ họa 2D

- Khi thiết bị muốn hiển thị đồ họa 2D trên màn hình, có 2 cách để thực hiện:
  - Vẽ dùng View
  - Vẽ dùng Canvas
- Vẽ dùng View:
  - Đơn giản nhưng kém linh hoạt
  - Dùng cho đồ họa đơn giản và ít hoặc không cần cập nhật
- Vẽ dùng Canvas:
  - Phức tạp hơn nhưng đa dạng và linh hoạt hơn
  - Dùng cho đồ họa phức tạp, cần cập nhật thường xuyên

Vẽ dùng View với lớp Drawable

- Drawable (đồ họa) biểu diễn những thứ có thể vẽ được như Bitmap, màu, hình dạng...
- Một số lớp drawable (đồ họa):
  - Lớp ShapeDrawable: biểu diễn các hình như chữ nhật, oval...
  - Lớp BitmapDrawable: biểu diễn một ma trận các điểm ảnh
  - Lớp ColorDrawable: biểu diễn màu sắc
- Để vẽ dùng View, tạo 1 đối tượng đồ họa rồi gắn nó vào ImageView. ImageView sau đó sẽ vẽ đối tượng đồ họa.
- Có thể thực hiện bằng XML hoặc viết mã Java
- VD: GraphicsBubbleXML và GraphicsBubbleProgramm

Lớp ShapeDrawable

- Dùng để vẽ các hình đơn giản.
- Các hình khác nhau được biểu diễn bởi lớp con tương ứng của lớp ShapeDrawable:
  - PathShape: Đoạn thẳng và đường cong
  - RectShape: Hình chữ nhật
  - OvaleShape: Hình Oval và hình tròn
- VD: GraphicsShapeDrawXML và GraphicsShapeDrawProgram

Vẽ dùng Canvas

- Cần 4 thứ để vẽ dùng Canvas:
  - Bitmap: ma trận điểm ảnh để vẽ (tờ giấy để vẽ)
  - Canvas: thực thi việc vẽ vào bitmap (cây bút)
  - Drawing Primitive: thể hiện nội dung vẽ cụ thể như Rect, path, Text, Bitmap (Hình cần vẽ)
  - Paint: để thiết lập màu sắc và kiểu dáng

Drawing Primitive

- Canvas hỗ trợ nhiều hàm vẽ:
  - drawText()
  - drawPoints()
  - drawColor()
  - drawOval()
  - drawBitmap()

Paint Object

- Lớp paint cho phép thiết lập các tham số khi vẽ như:
  - setStrokeWidth(): độ rộng nét vẽ
  - setTextSize()
  - setColor()
  - setAntiAlias(): làm mịn biên ảnh
- VD: GraphicsPaint

Canvas

- Canvas là một cơ chế để vẽ vào một bitmap
- Có thể lấy canvas thông qua:
  - Đối tượng View
  - Đối tượng SurfaceView (lớp con của View)
- Dùng đối tượng View:
  - Khi không cần thường xuyên update đồ họa
  - Tạo một lớp con kế thừa lớp View
  - Hệ thống sẽ cung cấp canvas khi gọi hàm onDraw()
- Dùng đối tượng SurfaceView
  - Khi thường xuyên update đồ họa
  - Tạo một lớp con kế thừa lớp SurfaceView
  - Tạo một luồng riêng cho việc vẽ
  - Khi chạy, ứng dụng có thể truy cập canvas để vẽ linh hoạt
- VD: GraphicsCanvasBubble

Lớp SurfaceView

- Để sử dụng SurfaceView, tạo một lớp con và cài đặt giao diện SurfaceHolder.Callback
- Để sử dụng lớp con vừa tạo, ta cần:
  - Thiết lập SurfaceView
  - Vẽ vào SurfaceView

Thiết lập Surfaceview

- Dùng hàm getHolder() của SurfaceView để lấy tham chiếu tới Surface
- Đăng ký hàm các hàm callback của SurfaceView bằng cách gọi hàm addCallback().
- Các hàm callback bao gồm:
  - surfaceCreate(): được gọi khi Surface được tạo
  - surfaceChanged()
  - surfaceDestroy()
- Tạo luồng thực thi việc vẽ trong surfaceCreated():
  - Các hàm callback của SurfaceHolder thường được gọi từ luồng chính do đó cần đồng bộ truy cập vào dữ liệu cần bởi cả 2 luồng

Vẽ vào SurfaceView

- Khóa Canvas:
  - SurfaceHolder.lockCanvas()
- Thực thi việc vẽ:
  - Canvas.drawBitmap()
- Mở khóa Canvas:
  - SurfaceHolder.UnlockCanvasAndPost()
- VD: GraphicsBubbleCanvasSurfaceView

Animation

- Animation là sự thay đổi các đặc tính của View trong một khoảng thời gian:
  - Kích thước
  - Vị trí
  - Độ trong suốt
  - Hướng
- Để làm Animiation dễ dàng hơn, Android cung cấp các lớp hỗ trợ:
  - Lớp TransitionDrawable: Chuyển trạng thái giữa 2 view
  - Lớp AnimationDrawable: tạo animation theo từng frame
  - Lớp Animation: tạo tween animation (ta chỉ ra các khung hình và thời điểm cần Animation và Android tự nội suy các điểm trung gian)

Lớp TransitionDrawable

- Định nghĩa đối tượng đồ họa 2 lớp
- Khi hiển thị, người dùng nhìn thấy lớp thứ nhất rồi sau đó mới thấy lớp thứ 2.
- VD: GraphicsTransitionDrawable

Lớp AnimationDrawable

- Animate một chuỗi các hình trong đó mỗi hình sẽ hiển thị trong một khoảng thời gian nhất định.
- VD: GraphicsFrameAnimation

Lớp Animation

- Một chuỗi các biến đổi được đặt vào một View.
- Ứng dụng có thể tùy biến thời gian animation để tạo hiệu ứng thay đổi liên tục hoặc tức thời.
- Có thể kết hợp các hiệu ứng để tạo animation phức tạp hơn.
- VD: GraphicsTweenAnimation

Property Animation

- Android phát triển một hệ thống hỗ trợ thay đổi các đặc tính của một đối tượng tổng quát theo thời gian.
- Hệ thống gồm một số thành phần:
  - ValueAnimator: lớp chính để điều khiển animation
  - TimeInterpolator: xác định giá trị thay đổi theo thời gian như thế nào (đều, nhanh dần, chậm dần).
  - AnimatorUpdateListerner: định nghĩa hàm onAnimationUpdate được gọi mỗi khi một animation mới được tạo
  - TypeEvaluator: định nghĩa hàm sẽ được gọi để đặt giá trị tại một thời điểm animation nhất định.
  - AnimatorSet: cho phép kết hợp các animation riêng lẻ thành animation phức tạp

Ví dụ

- GraphicsValueAnimator
- GraphicsViewPropertyAnimator

Touch và Gesture

- MotionEvents
- Xử lý chạm (Touch)
- Cử chỉ (Gesture)

MotionEvent

- Android sử dụng lớp MotionEvent để biểu diễn chuyển động trên thiết bị như bút, trackball, chuột hay ngón tay.
- Mỗi chuyển động bao gồm một số thông tin:
  - Action code: loại chuyển động
  - Action value: vị trí và đặc điểm của chuyển động như thời điểm, nguồn tạo, vị trí, lực chạm...
  - Ta chỉ tập trung vào sự kiện chạm/nhấn vào màn hình cảm ứng.

MultiTouch

- Đa số thiết bị hiện nay là cảm ứng đa điểm (MultiTouch): có thể theo dõi nhiều chạm tại cùng một thời điểm.
- Màn hình MultiTouch tạo ra một vết chuyển động với mỗi nguồn chạm.
- Mỗi nguồn chạm được gọi là 1 pointer.
- Mỗi pointer có 1 ID duy nhất chừng nào nó còn active
- Trong một số trường hợp, Android sẽ nhóm nhiều pointer thành một MotionEvent.
  - Trong trường hợp này, mỗi pointer (ngoài ID duy nhất) có thêm 1 chỉ số Index và chỉ số này có thể thay đổi theo thời gian.

Một số Action trong MotionEvent

- ACTION_DOWN: 1 ngón tay đã bắt đầu chạm màn hình
- ACTION_POINTER_DOWN: Đã có 1 ngón tay chạm màn hình và giờ có thêm 1 ngón nữa.
- ACTION_POINTER_UP: 1 ngón tay ngừng chạm màn hình
- ACTION_MOVE: Ngón tay di chuyển trên màn hình
- ACTION_UP: Ngón tay cuối cùng ngừng chạm màn hình
- ACTION_CANCEL: Hủy gesture hiện tại

Các hàm xử lý MotionEvent

- getActionMasked(): trả về action code gắn với MotionEvent
- getActionIndex(): trả về chỉ số của Pointer gắn với action code đó. VD: nếu action code là ACTION_POINTER_DOWN, ta dùng hàm này để lấy chỉ số của pointer vừa chạm.
- getPointerId(int pointerIndex): lấy ID duy nhất gắn với chỉ số pointer.
- getPointerCount(): lấy số pointer gắn với MotionEvent
- getX/getY(int pointerIndex): trả về tọa độ X/Y của pointer gắn với chỉ số đang xét.
- findPointerIndex(int pointerId): trả về chỉ số của pointer có ID đang xét.

Xử lý sự kiện chạm bằng Listener

- Muốn xử lý MotionEvent xảy ra trên 1 view nào đó có thể thực hiện giống xử lý click:

```
View.setOnTouchListener(new OnTouchListener()
    @Overrride
    public boolean onTouch(View v, MotionEvent event) {
    ....
    return true/false;
});
```

  - Hàm callback onTouch() được gọi khi có sự kiện chạm xuất hiện như chạm, thả, vuốt... : trả về true nếu đã xử lý sự kiện, ngược lại trả về false

Xử lý nhiều sự kiện chạm

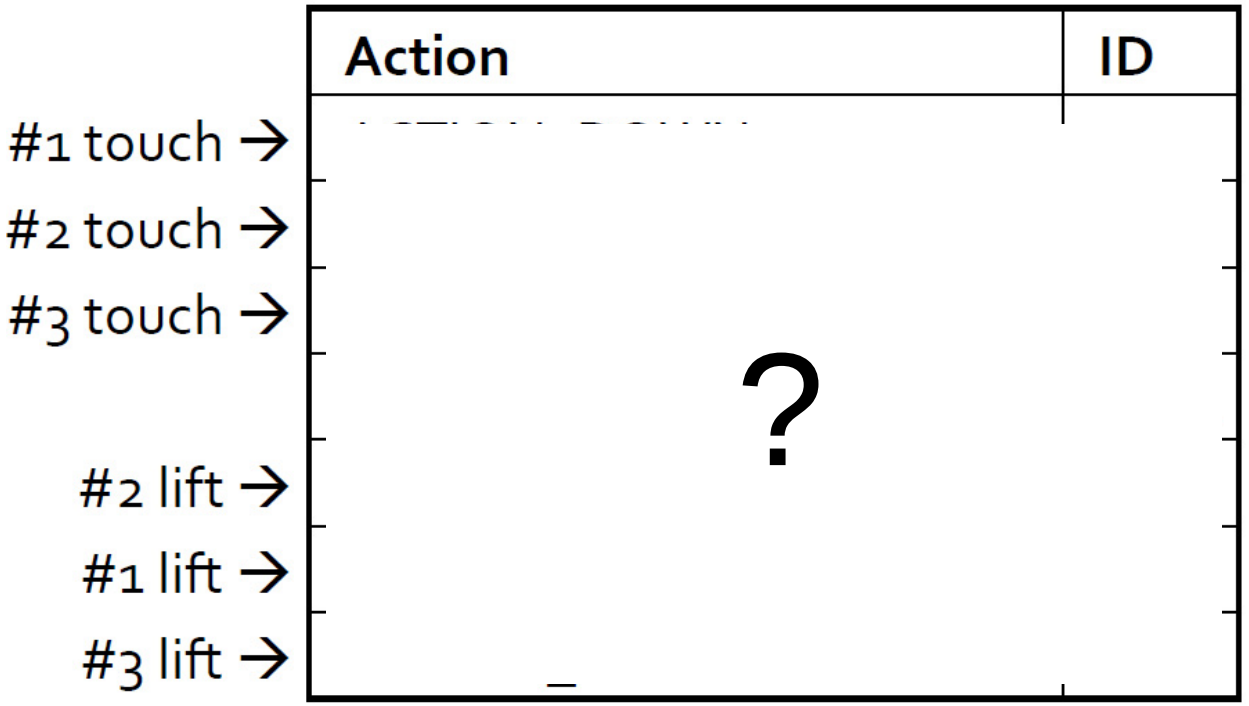
- Nhiều chạm kết hợp lại sẽ tạo thành 1 cử chỉ.
- Ứng dụng cần xác định và xử lý các tổ hợp chạm này.
- VD1: double tap sẽ bao gồm:
  - ACTION_DOWN, ACTION_UP
  - ACTION_DOWN, ACTION_UP

  xảy ra nhanh liên tiếp.
- VD2: Gesture

	Action	IDs
#1 touch →	ACTION_DOWN	0
	ACTION_MOVE ...	0
#2 touch →	ACTION_POINTER_DOWN	1
	ACTION_MOVE ...	0,1
#1 lift →	ACTION_POINTER_UP	0
#2 lift →	ACTION_UP	1

Xử lý nhiều sự kiện chạm

- VD3:



- Ứng dụng: TouchIndicateTouchLocation

GestureDetector

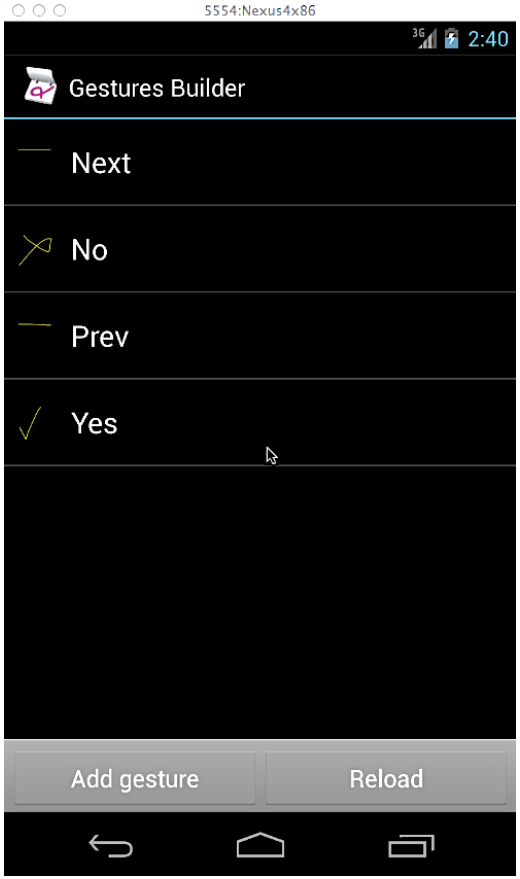
- Lớp GestureDetector dùng để phát hiện các cử chỉ chạm phổ biến.
- VD: Chạm đơn, chạm kép, vuốt nhanh
- Để sử dụng GestureDetector:
  - Tạo đối tượng GestureDetector
  - Thiết lập giao diện GestureDetector.SimpleOnGestureListener
  - Override hàm onTouchEvent(). Hàm này sẽ được gọi khi chạm xảy ra trên Activity.
- VD: TouchGestureViewFlipper

Tạo Gesture tùy ý

- Ứng dụng GestureBuilder cho phép tạo và lưu Gesture tự định nghĩa.
- Sau đó sử dụng GestureLibraries để nạp gesture và nhận dạng khi nào người dùng sử dụng gesture đó.
  - Dùng GestureOverlayView trong layout
  - Phần Overlay này sẽ nhận gesture của người dùng và thực thi code trong ứng dụng để xử lý.

VD: TouchGesture

- Dùng GestureBuilder trong emulator tạo gesture:
  - Gesture tạo ra được lưu ở /mnt/sdcard/gestures
  - Dùng DDMS download file xuống PC
  - Copy file vào thư mục /res/raw



Multimedia

- Các lớp hỗ trợ Multimedia
- Chơi nhạc
- Xem phim
- Sử dụng camera

Multimedia

- Android hỗ trợ nhiều định dạng media cho phép chơi nhạc, ghi âm, hiện hình ảnh, xem phim.
- Các lớp sẽ học:
  - AudioManager & SoundPool
  - MediaPlayer
  - Camera

Lớp AudioManager

- Quản lý âm thanh như âm lượng, hiệu ứng âm thanh, chế độ chuông.
- Ứng dụng sẽ lấy tham chiếu tới AudioManager bằng cách:

```
Context.getSystemService(Context.AUDIO_SERVICE)
```

- Sau đó, ứng dụng có thể:
  - Nạp và chạy các hiệu ứng âm thanh
  - Điều chỉnh âm lượng
  - Điều khiển phần cứng ngoại vi như làm câm micro, bật tai nghe bluetooth.

Lớp SoudPool

- Để biểu diễn một nhóm (playlist) các file audio.
- Cho phép trộn và chơi nhiều file audio.
- VD: AudioVideoAudioManager

Lớp MediaPlayer

- Điều khiển chơi audio hoặc video từ file hoặc stream.
- Cho phép ứng dụng và người dùng điều khiển quá trình chơi audio/video.
- Một số hàm hay dùng:
  - setDataSource(): chọn nguồn audio/video
  - prepare(): nạp audio/video
  - start()
  - pause()
  - seekTo()
  - stop()
  - release(): giải phóng tài nguyên sử dụng bởi MediaPlayer

Lớp VideoView

- Là lớp con của SurfaceView dùng để hiển thị file video.
- Có thể nạp video từ nhiều nguồn.
- Cung cấp các tùy chọn hiển thị và chức năng tiện ích
- VD: AudioVideoVideoPlay

Lớp Camera

- Cho phép ứng dụng truy cập Camera service để quản lý Camera trên thiết bị.
- Qua dịch vụ này, ứng dụng có thể:
  - Quản lý thiết lập chụp ảnh
  - Start/stop preview
  - Chụp ảnh/quay phim

Sử dụng Camera

- Cần camera permission:

```
<uses-permission
        android:name="android.permission.CAMERA" />
<uses-feature
        android:name="android.hardware.camera" />
<uses-feature
        android:name="android.hardware.camera.autofocus" />
```

- Sau đó thực hiện các bước:
  - Lấy tham chiếu Camera
  - Thiết lập tham số camera
  - Thiết lập hiển thị preview
  - Bắt đầu preview
  - Chụp ảnh và xử lý dữ liệu ảnh
  - Giải phóng camera

VD: AudioVideoCamera