2D Graphics with Canvas

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

香港科技大學
THE HONG KONG UNIVERSITY OF
SCIENCE AND TECHNOLOGY

2D Drawing With Canvas

- Suited for applications which require specialized drawing and/or control of the animation of graphics
- Canvas provides the interface to the actual surface upon which graphics can be drawn
  - Drawn to a bitmap which is placed into the window
- Canvas class offers a set of drawing methods:
  - drawBitmap(), drawRect(), drawText() …
- For applications not requiring significant amount of processing or frame-rate speed, create a custom view component and draw on it with a Canvas in View.onDraw()

2D Canvas on a View

- Extend the View class and define the onDraw() callback method
  - Use the onDraw() method to perform all calls to draw on the canvas
  - Android framework calls onDraw() to update the screen
  - Use invalidate() to trigger your view to be drawn
  - Inside your View component's onDraw(), use the Canvas given to you for all your drawing, using various Canvas.draw...() methods, or other class draw() methods that take your Canvas as an argument
  - Once your onDraw() is complete, the Android framework will use your Canvas to draw a Bitmap handled by the system.

Shooting Game Exercise

- Off to our next exercise
  - Starting from the Basic Graphics Exercise, develop a shooting Game
  - Use of onDraw() and invalidate() to generate motion on the screen

2D Graphics with Canvas on SurfaceView

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

香港科技大學
THE HONG KONG UNIVERSITY OF
SCIENCE AND TECHNOLOGY

2D Canvas on SurfaceView

- SurfaceView is a special subclass of View that offers a dedicated drawing surface within the View hierarchy
  - Offers this drawing surface to an application's secondary thread, so that the application isn't required to wait until the system's View hierarchy is ready to draw
  - Instead, a secondary thread that has reference to a SurfaceView can draw to its own Canvas at its own pace.

2D Canvas on SurfaceView

- In your code extend the SurfaceView class
- Need to implement the SurfaceHolder.Callback
- Also the interface requires three additional methods to be implemented: surfaceCreated(), surfaceChanged(), and surfaceDestroyed()
  - Important so that you know when you can start drawing, whether you need to make adjustments based on new surface properties, and when to stop drawing and potentially kill some tasks.

2D Canvas on SurfaceView

- The surface object is handled via a SurfaceHolder which you get when you initialize the SurfaceView, by calling getHolder()
- Inform the SurfaceHolder that you will handle the callbacks by calling holder.addCallback(this). Then override the Callback methods within your SurfaceView: surfaceCreated(), surfaceChanged(), and surfaceDestroyed()
- Create a secondary thread that will control all the drawings to the surface, and pass it the SurfaceHolder
  - Within the thread get hold of the canvas: surfaceHolder.lockCanvas()
  - Then make modifications to the canvas
  - Finally, surfaceHolder.unlockCanvasAndPost(canvas)

Shooting Game with Music Exercise

- Off to our next exercise
  - Extending the Shooting Game with background music
  - Using the MediaPlayer Class
    - Initializing the player
    - Controlling playback using MediaPlayer methods
  - Menu items in the toolbar/actionbar