COMP08076
Programming Native App Interaction

- Module coordinator: John Nixon
  - john.nixon@uws.ac.uk, room E259, x3617

- Lecture 7
  - Threads
  - SurfaceView
  - Runnable Interface

Threads

- A **thread** in computer science is short for a *thread of execution*. Threads are a way for a program to divide (termed *"split"*) itself into two or more simultaneously (or pseudo-simultaneously) running tasks.
- The Android apps we have created so far are single-threaded.
- When an application is launched, the system creates a thread of execution for the application, called **main thread** or **UI thread**
- This thread is in charge of dispatching events to the appropriate user interface widgets, including drawing events.
- The system does *not* create a separate thread for each instance of a component. All components that run in the same process are instantiated in the UI thread, and system calls to each component are dispatched from that thread.

Problems with single threads

- Doing lengthy processing, for example when an application is calculating it cannot process any more messages.
- Doing background processing: Some tasks may not be time critical, but need to execute continuously.
- Doing I/O work: I/O to disk or to network can have unpredictable delays. Threads allow you to ensure that I/O latency does not delay unrelated parts of your application.
- Limited use in games
  - Restricted to turn based games

Multiple Threads

- “solves” the problems above
- Can be more efficient (particularly if computer has multiple processors)

- Allows gameplay to exist in a separate thread
  - What we will be using

- Does need to be managed
  - Not a big issue for us as the code we will adopt does this

View Class, again – from lecture 3

- This class represents the basic building block for user interface components. A View occupies a rectangular area on the screen and is responsible for drawing and event handling. View is the base class for *widgets*, which are used to create interactive UI components (buttons, text fields, etc.).
- http://developer.android.com/reference/android/view/View.html

- Extends object

- Has subclasses
  - ImageView
  - SurfaceView
  - TextView
  - Etc.

- ImageView has subclass ImageButton
- TextView has subclasses Button, EditText etc
- Button has subclasses CheckBox, RadioButton etc

SurfaceView

http://developer.android.com/reference/android/view/SurfaceView.html

- Provides a dedicated drawing surface
- The Surface will be created for you while the SurfaceView's window is visible
- Methods to implement
  - surfaceCreated(SurfaceHolder)
  - surfaceDestroyed(SurfaceHolder)
  - surfaceChanged(SurfaceHolder, int,int, int)
- Call onDraw() explicitly
  - Not called automatically as with custom view
- onDraw() creates the SurfaceView’s output – i.e. what the user sees.

GameActivity (same as for View)

```
public class GameActivity extends Activity {
    GameView GV;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        GV = new GameView(this);
        setContentView(GV);
    }
}
```

GameView

- public class GameView extends SurfaceView implements Runnable {

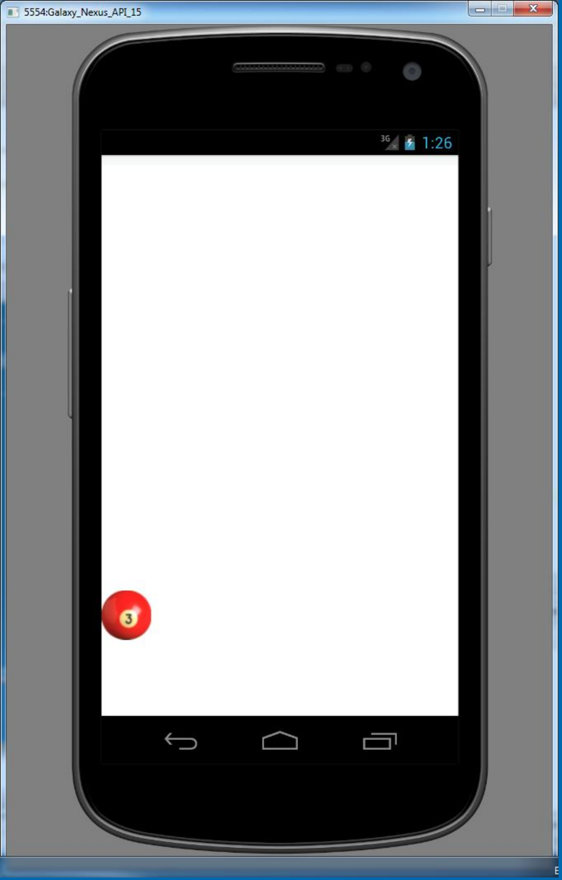
- Set up new thread
  - thread = new Thread(this);
  - thread.start()

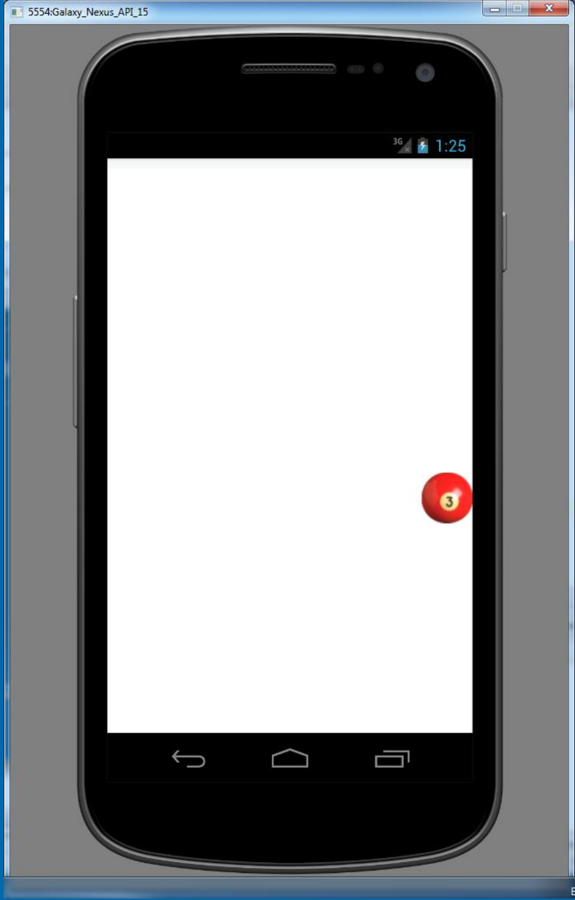
- Thread has run() method
  - Acts as a loop while it is running
  - Use to call an update() and onDraw

- Note – need to call onDraw() explicitly, unlike with View
- Declare game assets, such as bitmaps, in GameView

```
@Override
   public void run() {
       while (running) {
           Canvas c = null;
           try {
               c = getHolder().lockCanvas();
               synchronized (getHolder()) {
                   update();
                   onDraw(c);
               }
           } finally {
               if (c != null) {
                   getHolder().unlockCanvasAndPost(c);
               }
           }
       }
   }
```

```
@Override
   protected void onDraw(Canvas canvas) {
       canvas.drawColor(Color.BLACK);
       if (x < getWidth() – ball1.getWidth()) {
           x++;
       }
       canvas.drawBitmap(ball1, x, 10, null);
   }
```





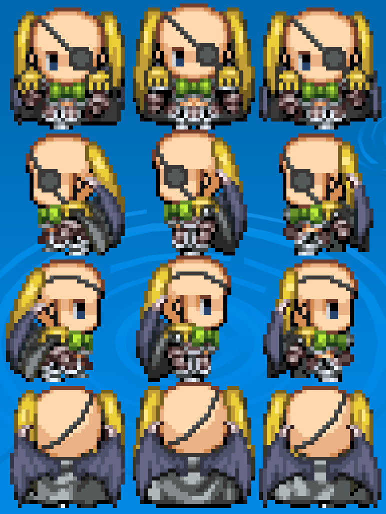
➢ ball.getWidth()	width of ball graphic
➢ getWidth()	width of screen

➢ Tests for position:
- if x < 0
- if x + ball.getWidth() > getWidth()

Sprites

- Sprite
  - A small or elusive supernatural being
  - A two dimensional image or animation that is integrated into a larger scene

- Sprite Class
  - A base class for different types of objects in a game – a java file
- Sprite Sheet
  - Collection of sprite images on a single sheet



Making a sprite for the graphically challenged

- http://translate.google.co.uk/translate?hl=en&sl=ja&u=http://www.famitsu.com/freegame/tool/chibi/index2.html&ei=_Pn-ScGQLt-2jAez2-msAw&sa=X&oi=translate&resnum=1&ct=result&prev=/search%3Fq%3Dfamitsu%2Bchibi%26hl%3Den%26safe%3Doff%26sa%3DG

Sprite class

- New java file
- Represents the sprite on the screen
  - Needs a bitmap to display
  - Needs to know where to display it

- Can have own update() and onDraw() methods
  - So can move itself
  - Can draw itself

  - Move the things the sprite needs to do into the Sprite class.

```
import android.graphics.Canvas;

public class Sprite {
    private int x = 0;// and other declarations
    private GameView gameView;
    private Bitmap bmp;
 //Constructor
    public Sprite(GameView gameView, Bitmap bmp) {
        this.gameView=gameView;
        this.bmp=bmp;
    }

    private void update() {
        if (x > gameView.getWidth() - bmp.getWidth() - xSpeed)
//movement stuff
    }

    public void onDraw(Canvas canvas) {
        update();
        canvas.drawBitmap(bmp, x , 10, null);
    }
```

- private Sprite sprite;
- sprite = new Sprite(this, ball);

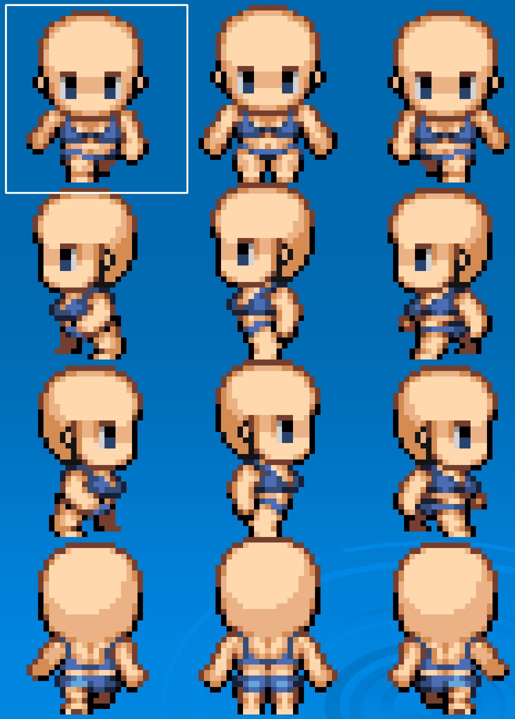
- In GameView

```
@Override
    protected void onDraw(Canvas canvas) {
        canvas.drawColor(Color.BLACK);
        sprite.onDraw(canvas);
    }
```

Animated sprite

- Rect
- currentFrame

- Down
- Left
- Right
- Up



Randomness

- Sprite starting position
  - x = rnd.nextInt(gameView.getWidth() - width);
  - y = rnd.nextInt(gameView.getHeight() - height);

- Sprite speed
  - xSpeed = rnd.nextInt(10)-5;
  - ySpeed = rnd.nextInt(10)-5;

  - xSpeed = rnd.nextInt(MAX_SPEED * 2) - MAX_SPEED;
  - ySpeed = rnd.nextInt(MAX_SPEED * 2) - MAX_SPEED;

Assessment 2

- For this part of the assessment you are to specify and implement an app with a minimum of 3 activities
  - A landing page activity that allows navigation to the two other game activities off the app and some explanation of what the games are
  - A View turn based game that involves interaction with graphics, a score of some sort and a game over notification when the user has successfully completed the game.
  - A SurfaceView based action game involving interaction with and between sprites. This is most likely to be based on the examples used in the labs.

- The **specification** for your product must **concisely** explain its features (and mention any that don't work!), any instructions or rules not included in the product itself and a description of user triggered and internal events. **The specification must also include references to sources of code or ideas.**

- The work submitted must be your own work. Reference **must** be made in the specification and comments in the code to any sources of code or ideas used. You are not permitted to submit work that results from the completion of online, book or other tutorials – these will not be considered your own work – however you may incorporate referenced ideas into your work. **Your work will not be marked if you do not submit a specification document.**