MICROOH 麦可网

Android-从程序员到架构师之路

出品人：Sundy
讲师：高焕堂（台湾）

http://www.microoh.com

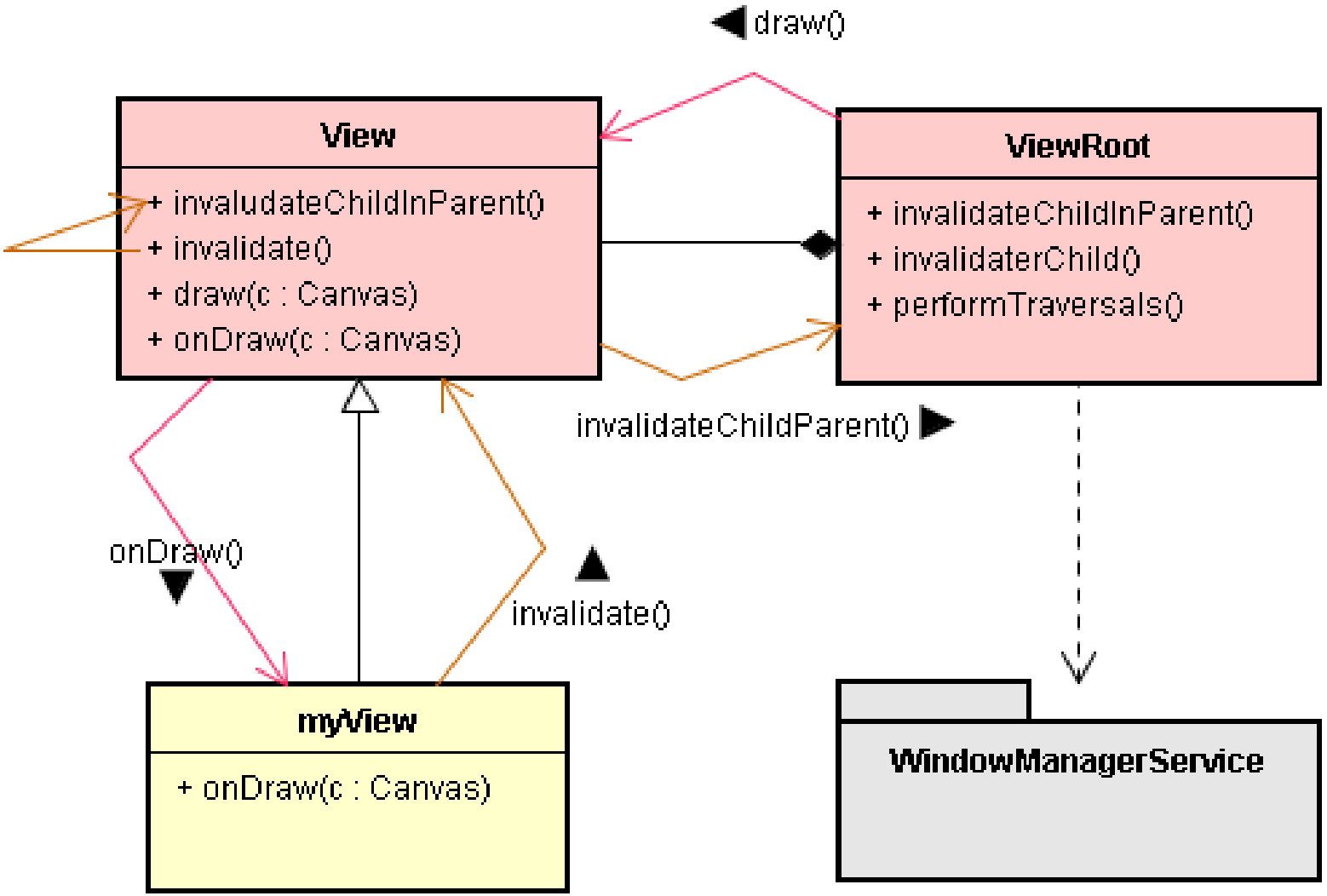
B03_b

应用Android的UI框架(b)

-- 以设计游戏循环(GameLoop)为例

By 高煥堂

3、使用UI线程的MQ(Message Queue)


draw()
View
+ invaludateChildInParent()
+ invalidate()
+ draw(c : Canvas)
+ onDraw(c : Canvas)
ViewRoot
+ invalidateChildInParent()
+ invalidaterChild()
+ performTraversals()
invalidateChildParent()
onDraw()
invalidate()
myView
+ onDraw(c : Canvas)
WindowManagerService

```
// myView.java
// .....

public class myView extends View {
   // .....
   @Override protected void onDraw(Canvas canvas) {
           super.onDraw(canvas);
           // .....
           // canvas.drawRect(....);
          invalidate();
    }
}
```

- 我们可以透过Message方式来触发UI线程去調用invalidate()函数，而达到重新执行onDraw()来进行重复绘图和刷新画面的动作。

```
// myView.java
//……..
public class myView extends View {
    private Paint paint= new Paint();
    private int line_x = 100, int line_y = 100;
    private float count = 0;
    private myHandler h;

    myView(Context ctx)
        {   super(ctx);
            h = new myHandler();
        }
```

```
@Override protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    if( count > 12) count = 0;
    int x = (int) (75.0 * Math.cos(2*Math.PI * count/12.0));
    int y = (int) (75.0 * Math.sin(2*Math.PI * count/12.0));
    count++;
    canvas.drawColor(Color.WHITE);
    paint.setColor(Color.RED);
    paint.setStrokeWidth(3);
    canvas.drawLine(line_x, line_y, line_x+x, line_y+y, paint);
    paint.setStrokeWidth(2);
    paint.setColor(Color.BLUE);
```

```
    canvas.drawRect(line_x-5, line_y - 5, line_x+5, line_y + 5, paint);
    paint.setColor(Color.YELLOW);
    canvas.drawRect(line_x-3, line_y - 3, line_x+3, line_y + 3, paint);

    h.removeMessages(0);
    Message msg = h.obtainMessage(0);
    h.sendMessageDelayed(msg, 1000);
   }
  class myHandler extends Handler {
        @Override  public void handleMessage(Message msg) {
             invalidate();
            }
      };
  }
```

- 使用sendMessageDelayed()函数来暂停一下，延迟数秒钟才传递 Message给UI线程。

4、诞生一个小线程，担任游戏线程

- 刚才是由UI线程来丢Message到自己的MQ里；也就是UI线程丢Message给自己。同一样地，也可以由其它线程来丢Message到UI线程的MQ里，来触发UI线程去調用invalidate()函数。

```
// myView.java
// ……..
public class myView extends View {
    private Paint paint= new Paint();
    private int line_x = 100, line_y = 100;
    private float count = 0;
    private myHandler h;

     myView(Context ctx) {   super(ctx);   h = new myHandler();   }
     @Override protected void onDraw(Canvas canvas) {
              super.onDraw(canvas);
             if( count > 12) count = 0;
             int x = (int) (75.0 * Math.cos(2*Math.PI * count/12.0));
             int y = (int) (75.0 * Math.sin(2*Math.PI * count/12.0));
             count++;
```

```
canvas.drawColor(Color.WHITE);
paint.setColor(Color.RED);
paint.setStrokeWidth(3);
canvas.drawLine(line_x, line_y, line_x+x, line_y+y, paint);
paint.setStrokeWidth(2);
paint.setColor(Color.BLUE);
canvas.drawRect(line_x-5, line_y - 5, line_x+5, line_y + 5, paint);
paint.setColor(Color.MAGENTA);
canvas.drawRect(line_x-3, line_y - 3, line_x+3, line_y + 3, paint);
//------------------------------
myThread t = new myThread();
t.start();
}
```

```
        // 诞生一个小线程，担任游戏线程，负责回圈控制
        class myThread extends Thread{
              public void run() {
                        h.removeMessages(0);
                        Message msg = h.obtainMessage(0);
                         h.sendMessageDelayed(msg, 1000);
                }
          };

         class myHandler extends Handler {
         @Override public void handleMessage(Message msg) {
                  invalidate(); // call onDraw()
          }};
}
```

- UI线程诞生一个小线程，并且由该小线程去执行myThread类别里的run()函数。接着，这新线程执行到指令：

```
h.removeMessages(0);
Message msg = h.obtainMessage(0);
h.sendMessageDelayed(msg, 1000);
```

- 延迟数秒钟才传递 Message给UI线程(丢入UI线程的MQ里)。
- 当UI线程发现MQ有个Message，就去执行myHandler类别里的handleMessage()函数。就触发UI线程去調用invalidate()函数了。

5、小线程(非UI线程)
調用postInvalidate()

- 刚才的小线程传递Message给UI线程(丢入UI线程的MQ里)，触发UI线程去調用invalidate()函数。Android提供一个postInvalidate()函数来替代上述的动作。由小线程直接去調用postInvalidate()函数，就能间接触发UI线程去調用invalidate()函数了。

```
// myView.java
//……
public class myView extends View {
      //…….
     @Override protected void onDraw(Canvas canvas) {
               //……..
               myThread t = new myThread();
               t.start();
             }
      class myThread extends Thread{
               public void run() {
                        postInvalidateDelayed(1000);
             }};
}
```

- 由小线程直接去調用postInvalidate()函数；就相当于，由小线程传递Message给UI线程，触发UI线程去調用invalidate()函数。

~ Continued ~