2015 年 11 月 19 日（木）実施

SurfaceView による高速描画

SurfaceView とは

SurfaceView は，View の階層の中に描画専用の面を埋め込むためのクラスである。SurfaceView を継承したクラスでは，View を継承したクラスと異なり，描画処理専用の**スレッド**（並列処理に対応した OS 上でのプログラムの最小の実行単位；Thread）を主要なスレッドとは別に割り当てることが可能で，高速な描画が可能となる。

課 題

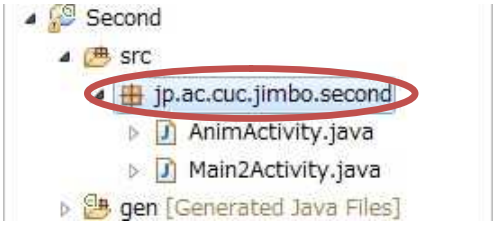
今回は，SurfaceView を継承した新規のクラスを作成し，高速描画の基本を学ぶ。

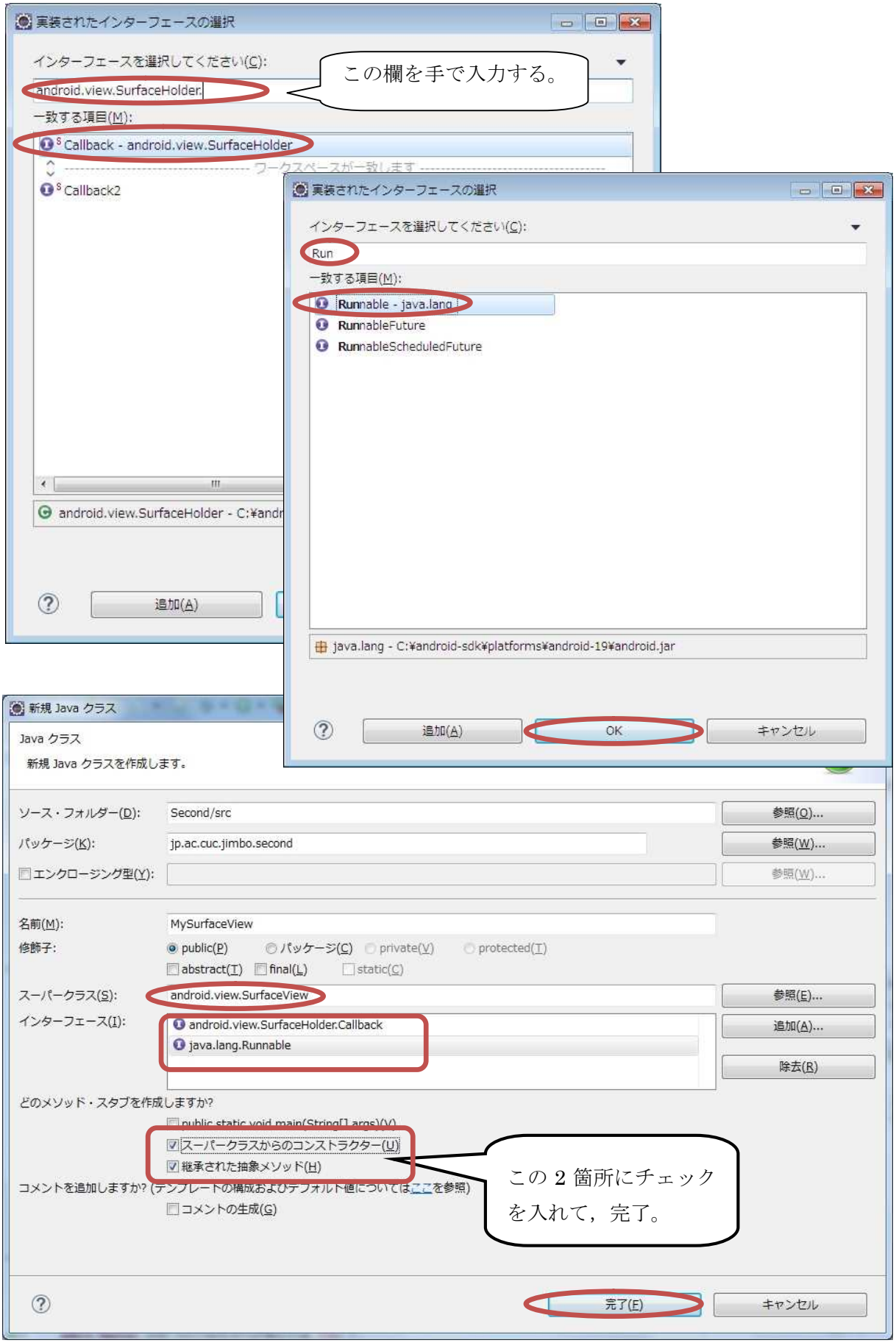
Android アプリの作成

Eclipse を起動し，パッケージ・エクスプローラーでパッケージ名を選択し，『ファイル』→『新規』→『クラス』と選択して，『次へ』をクリックする。

『名前』を「MySurfaceView」，『スーパークラス』を「android.view.SurfaceView」に書き換える。

『インターフェース』で『追加』ボタンを押し，「android.view.SurfaceHolder.Callback」を追加し（次のページの図），再度『追加』ボタンを押して「java.lang.Runnable」を追加する。




実装されたインターフェースの選択
インターフェースを選択してください(C):
android.view.SurfaceHolder
この欄を手で入力する。
一致する項目(M):
Callback - android.view.SurfaceHolder
Callback2
android.view.SurfaceHolder - C:¥andr
追加(A)
実装されたインターフェースの選択
インターフェースを選択してください(C):
Run
一致する項目(M):
Runnable - java.lang
RunnableFuture
RunnableScheduledFuture
java.lang - C:¥android-sdk¥platforms¥android-19¥android.jar
追加(A)
OK
キャンセル
新規 Java クラス
Java クラス
新規 Java クラスを作成します。
ソース・フォルダー(D):
Second/src
参照(O)...
パッケージ(K):
jp.ac.cuc.jimbo.second
参照(W)...
エンクロージング型(Y):
名前(M):
MySurfaceView
修飾子:
public(P)
パッケージ(C)
private(V)
protected(T)
abstract(T)
final(L)
static(C)
スーパークラス(S):
android.view.SurfaceView
参照(E)...
インターフェース(I):
android.view.SurfaceHolder.Callback
java.lang.Runnable
追加(A)...
除去(R)
どのメソッド・スタブを作成しますか?
スーパークラスからのコンストラクター(U)
継承された抽象メソッド(H)
コメントを追加しますか? (テンプレートの構成およびデフォルト値についてはここを参照)
コメントの生成(G)
この 2 箇所にチェックを入れて，完了。
完了(F)
キャンセル

MySurfaceView.java

```
package jp.ac.cuc.jimbo.second;

import android.content.Context;
import android.util.AttributeSet;
import android.view.SurfaceHolder;
import android.view.SurfaceHolder.Callback;
import android.view.SurfaceView;

public class MySurfaceView extends SurfaceView implements Callback, Runnable {

    public MySurfaceView(Context context) {
        super(context);
        // TODO 自動生成されたコンストラクター・スタブ
    }

    public MySurfaceView(Context context, AttributeSet attrs) {
        super(context, attrs);
        // TODO 自動生成されたコンストラクター・スタブ
    }

    public MySurfaceView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        // TODO 自動生成されたコンストラクター・スタブ
    }

    @Override
    public void run() {
        // TODO 自動生成されたメソッド・スタブ

    }

    @Override
    public void surfaceCreated(SurfaceHolder holder) {
        // TODO 自動生成されたメソッド・スタブ

    }

    @Override
    public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {
        // TODO 自動生成されたメソッド・スタブ

    }

    @Override
    public void surfaceDestroyed(SurfaceHolder holder) {
        // TODO 自動生成されたメソッド・スタブ

    }

}
```

この様に自動生成されるので，それぞれのメソッドを実装する。

MySurfaceView.java

```
package jp.ac.cuc.jimbo.second;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.util.AttributeSet;
import android.view.SurfaceHolder;
import android.view.SurfaceHolder.Callback;
import android.view.SurfaceView;

public class MySurfaceView extends SurfaceView implements Callback, Runnable {
    private SurfaceHolder holder;
    private Thread thread;
    private int x, y, xadd = 5, yadd = 10;
    private final int r = 10;

    public MySurfaceView(Context context) {
        super(context);
        surfaceInit();
    }

    public MySurfaceView(Context context, AttributeSet attrs) {
        super(context, attrs);
        surfaceInit();
    }

    public MySurfaceView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        surfaceInit();
    }

    private void surfaceInit() {
        x = 15;
        y = 20;
        holder = getHolder();
        holder.addCallback(this);
    }

    @Override
    public void run() {
```

```
40    @Override
41    public void run() {
42        Canvas canvas = null;
43        Paint p = new Paint();
44        p.setColor(Color.YELLOW);
45
46        while (thread != null) {
47            try {
48                canvas = holder.lockCanvas();
49                if (canvas != null) {
50                    canvas.drawColor(Color.CYAN);
51                    canvas.drawCircle(x, y, r, p);
52                }
53            } finally {
54                if (canvas != null) {
55                    holder.unlockCanvasAndPost(canvas);
56                }
57            }
58            x += xadd;
59            y += yadd;
60            if (x <= r || x > 500 - r)
61                xadd = -xadd;
62            if (y <= r || y > 700 - r)
63                yadd = -yadd;
64        }
65    }
66
67    @Override
68    public void surfaceCreated(SurfaceHolder holder) {
69        thread = new Thread(this);
70        thread.start();
71    }
72
73    @Override
74    public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {
75    }
76
77    @Override
78    public void surfaceDestroyed(SurfaceHolder holder) {
79        thread = null;
80
81    }
82
83 }
84
```

MySurfaceView. java に上の図の色付きの枠で囲った箇所を付け加える。

1) 橙色の枠内　MySurfaceView クラスのフィールド ・・・ クラス内の複数のメソッドで使用。SurfaceHolder のインスタンス holder，Thread のインスタンス thread，円の中心座標 x，y，円の中心座標の増分 xadd，yadd，円の半径 r

2) 紫色の枠内　コンストラクタ MySurfaceView から自作のメソッド surfaceInit を呼び出して初期化。

3) 水色の枠内　getHolder メソッドで SurfaceHolder（描画面を保持，制御するためのインターフェイス）を取得し，addCallback メソッドでホルダーholder にコールバックインターフェイスを付加する。

4) 濃赤色の枠内　Canvas を用いて，run メソッド内に描画処理を記述。

【描画に必要な 4 つの要素】

(1) ビットマップ　　ピクセル（画素）を保持
(2) キャンバス　　描画コール（ビットマップへの書き出し要請）に対応
(3) 描画プリミティブ　描画領域，パス，テキスト，ビットマップ等
(4) ペイント　　画像の色及びスタイル

① Canvas のインスタンス canvas, Paint のインスタンス p を作成し, p に黄色をセットする。
② thread が空でない，即ち描画面が破棄されない限り描画を繰り返す。
(1) lockCanvas メソッドは，描画面上でピクセルの編集を開始する。戻り値のキャンバスは描画面のビットマップへの描画に利用可能となる。
(2) drawColor メソッドは，キャンバスの描画面全体のビットマップに特定の色をセットする。ここではシアン色をセットしている。
(3) drawCircle メソッドは，中心座標，半径，ペイントを指定して，円を描く。ここでは，ペイントは色のみ指定して，スタイルを指定していないので，黄色く塗り潰される。線のみを描画するためには，「p.setStyle(Style.STROKE);」の記述を追加しておく必要がある。
(4) unlockCanvasAndPost メソッドは，ピクセルの編集を終了する。このメソッドが呼ばれると，画面上に描画面のピクセルが表示されるが，描画面の内容は失われる。
(5) x, y の座標を xadd, yadd だけ移動する。但し，x, y が半径以下になった場合及び x が 500−半径，y が 700−半径を超えた場合は xadd，yadd の符号を反転する。
5) 明赤色の枠内　描画用スレッドを生成し，開始する。
6) 黄色の枠内　描画面が破棄された時点で，描画用スレッドに null を設定して，run メソッドの while の継続条件を満たさなくなる様にして，繰り返しを止める。

それぞれの色の枠内を次に示す。

```
private SurfaceHolder holder;
private Thread thread;
private int x, y, xadd = 5, yadd = 10;
private final int r = 10;
```

```
surfaceInit();
```

```
private void surfaceInit() {
    x = 15;
    y = 20;
    holder = getHolder();
    holder.addCallback(this);
}
```

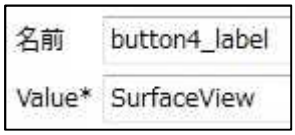
```
Canvas canvas = null;
Paint p = new Paint();
p.setColor(Color.YELLOW);

while (thread != null) {
    try {
        canvas = holder.lockCanvas();
        if (canvas != null) {
            canvas.drawColor(Color.CYAN);
```

```
                canvas.drawCircle(x, y, r, p);
            }
        } finally {
            if (canvas != null) {
                holder.unlockCanvasAndPost(canvas);
            }
        }
        x += xadd;
        y += yadd;
        if (x <= r || x > 500 - r)
            xadd = -xadd;
        if (y <= r || y > 700 - r)
            yadd = -yadd;
    }
```

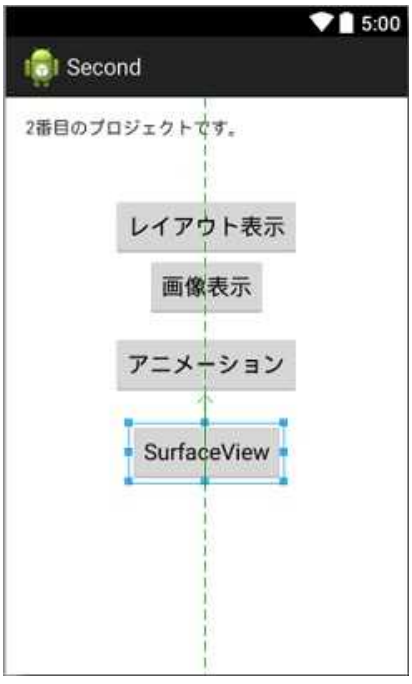
```
thread = new Thread(this);
thread.start();
```

```
thread = null;
```

『res』→『values』と展開し，『strings.xml』を開き，『追加』をクリックする。『String』を選択して OK をクリックし，『名前』を「button4_label」，『Value』を「SurfaceView」と入力して，保管する。



activity_main2.xml にボタンを配置する。ボタンの Text には『button4_label』を設定する。



Main2Activity. java を開く。

onCreate メソッド中に Button ウィジェット button4 のインスタンス btn4 を作成し，それに働きかけるイベントリスナーを付け加える。イベントリスナーでは，MySurfaceView を生成して画面にセットする，drawSurfaceView メソッド（自作する）を呼び出す。

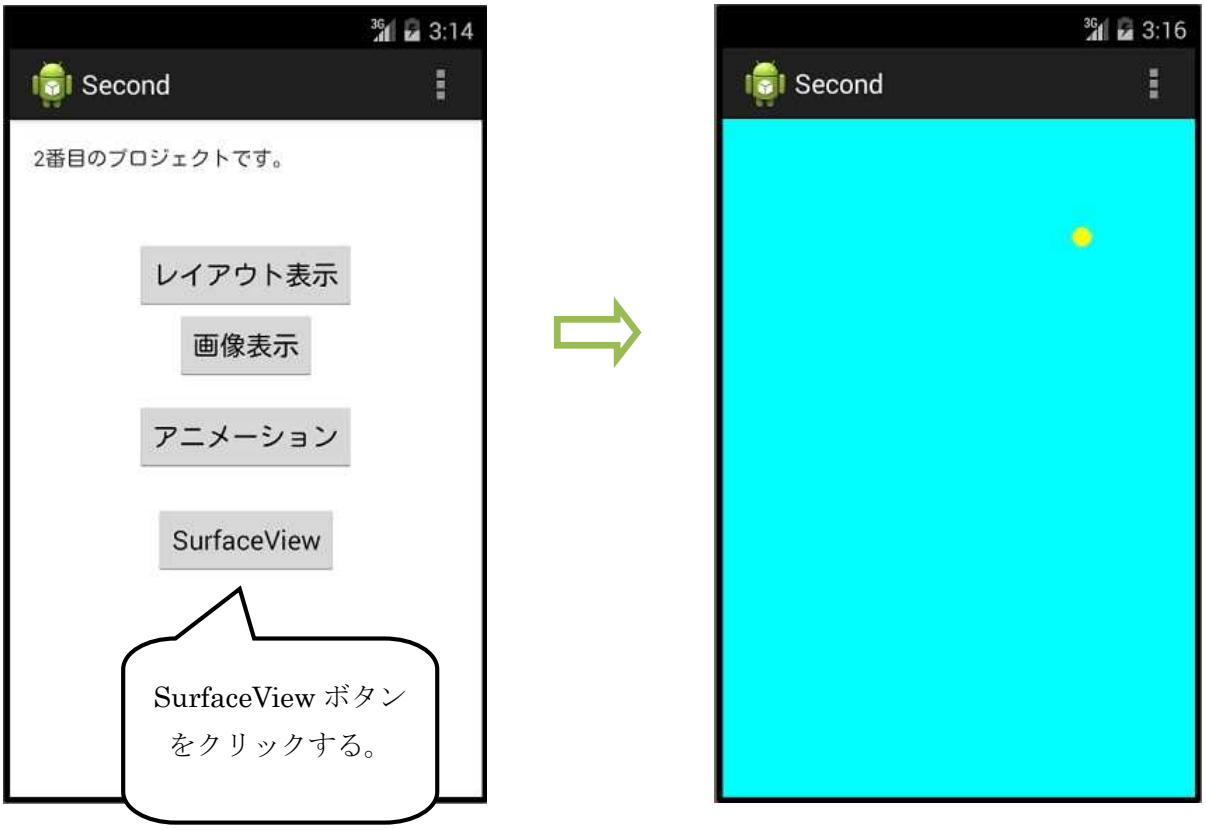
```
Button btn4 = (Button)this.findViewById(R.id.button4);
btn4.setOnClickListener(
    new View.OnClickListener() {

        @Override
            public void onClick(View v) {
                drawSurfaceView();
            }
        }
);
```

Main2Activity.java

```
10
11  public class Main2Activity extends Activity {
12
13      @Override
14      protected void onCreate(Bundle savedInstanceState) {
15          super.onCreate(savedInstanceState);
16          setContentView(R.layout.activity_main2);
17          Button btn = (Button)this.findViewById(R.id.button1);
18          btn.setOnClickListener(
19              new View.OnClickListener() {
20
21                  @Override
22                  public void onClick(View v) {
23                      drawLayouts();
24                  }
25              }
26          );
27          Button btn2 = (Button)this.findViewById(R.id.button2);
28          btn2.setOnClickListener(
29              new View.OnClickListener() {
30
31                  @Override
32                  public void onClick(View v) {
33                      setContentView(R.layout.image);
34                  }
35              }
36          );
37          Button btn3 = (Button)this.findViewById(R.id.button3);
38          btn3.setOnClickListener(
39              new View.OnClickListener() {
40
41                  @Override
42                  public void onClick(View v) {
43                      Intent intent = new Intent();
44                      intent.setClassName("jp.ac.cuc.jimbo.second", "jp.ac.cuc.jimbo.second.AnimActivity");
45                      startActivity(intent);
46                  }
47              }
48          );
49          Button btn4 = (Button)this.findViewById(R.id.button4);
50          btn4.setOnClickListener(
51              new View.OnClickListener() {
52
53                  @Override
54                  public void onClick(View v) {
55                      drawSurfaceView();
56                  }
57              }
58          );
59      }
60
61      public void drawSurfaceView() {
62          setContentView(new MySurfaceView(this));
63      }
64
```

```
public void drawSurfaceView() {
    setContentView(new MySurfaceView(this));
}
```

『保管』のアイコンをクリックして，Main2Activity. java を上書き保存し，実行ボタンをクリックする。



提出物：

1）画面のレイアウト設定ファイル activity_main2. xml

2）MySurfaceView のソースファイル MySurfaceView. java

3）Main2Activity のソースファイル Main2Activity. java