Android ゲームプログラミング

○　第６回　タッチイベント　○

タッチイベント

・タッチスクリーン端末では、ボタンがないためタッチで操作する
・Androidでは、タッチするとタッチイベントが発生する
・タッチイベントには、画面に指をつける、指を放す、指をつけたまま移動する、の３つがある
・onTouchEventメソッドを定義することにより、タッチイベントを処理できる
・ViewクラスでsetClickable(true)を実行しないと「指をつける」イベントしか取得できない

概要

ほとんどのタッチスクリーン端末のボタンは、電源、音量、ホームといった端末の基本的な操作をするもので、アプリケーションを操作するためのものはありません。代わりに、画面を指でふれながらジェスチャーを行うことにより、さまざまな操作が行えるようになっています。

タッチスクリーンのもっとも基本的な操作方法にタッチとドラッグがあります。タッチは指で画面にふれる、ドラッグは画面を指でふれたまま移動するというものです。

Androidでは、これらの操作が行われると、タッチイベントが発生します。基本的なタッチイベントとして「指をつける」「指を放す」「指をつけたまま移動する」の３つが定義されています。これらのイベントを処理するには、ActivityクラスもしくはViewクラス(これらを継承したクラスを含む)でonTouchEventメソッドを定義(オーバーライド)する必要があります。

onTouchEventメソッド

タッチイベントが発生すると、onTouchEventメソッドが呼び出されます。このメソッドは、そのままでは基底クラスのActivityクラスおよびViewクラスが内部的に処理しているだけですが、派生クラスでオーバーライドすることにより、アプリケーション独自の動作をさせることができます。
多くのゲームで使われるSurfaceViewクラスもViewクラスを継承して定義されているので、onTouchEventメソッドを定義することができます。このメソッドが呼び出されると、引数にタッチイベント情報が格納されたMotionEventクラスの変数が渡されます。

また、Viewクラスでは、デフォルトでは「指をつける」イベントしか取得できないようになっています。ViewクラスのsetClickableメソッドにtrueを指定することにより、すべての基本イベントが取得できるようになります。falseを指定するともとの状態に戻ります。タッチイベントを処理したくない場面でfalseを指定する、という使い方もできます(「指をつける」イベントは発生します)。

MotionEventクラス

onTouchEventメソッドには、タッチイベントの情報が格納されたMotionEventクラスの変数が渡されます。MotionEventクラスには、イベント情報を取得するために、いくつかのメソッドが定義されています。

getAction()	タッチイベントの種類
getX()	イベントが起きたときのタッチｘ座標
getY()	イベントが起きたときのタッチｙ座標
getDownTime()	押されていた時間(ms単位)
getEventTime()	イベントの継続時間(ms単位)
getEdgeFlags()	スクリーンの端かどうかの判定
getSize()	タッチされている範囲、サイズ
getPressure()	タッチの圧力・強さ

getActionメソッドは、発生したイベントを以下の定数で返します。onTouchEventメソッドでは、getActionメソッドの値をswitch文やif文で調べ、イベントにあった処理を行うようにします。

```
MotionEvent.ACTION_DOWN      画面に指をふれた
MotionEvent.ACTION_MOVE      画面に指をふれたまま移動している
MotionEvent.ACTION_UP        画面から指を持ち上げた
MotionEvent.ACTION_CANCEL    端末がタッチ操作をキャンセルしたと判定したときに発生
MotionEvent.ACTION_OUTSIDE   ターゲットの範囲外をタッチした
```

「画面を指で押した後、放した」という操作が行われた場合は、以下のようにタッチイベントが発生します。

```
ACTION_DOWN…画面にふれた際、一度だけ発生
     ↓
ACTION_MOVE…画面にふれながら指を少しでも動かすと繰り返し発生(動かさないと発生しません)
     ⋮        ※エミュレータでは、指の代わりのマウスをまったく動かさない、はできますが、
ACTION_MOVE     実機で指をまったく動かさないのはほぼ不可能なので、繰り返し発生します
     ↓
ACTION_UP  …画面から指を放した際に発生
```

端末によっては、最後のACTION_UPが発生せず、ACTION_CANCELになってしまうものがあります。「指を放した」イベントを処理する場合、どのような端末でも対応できるように、ACTION_UPとACTION_CANCELの両方に処理を記述しておきます。

コーディング例

```
// SurfaceViewクラスコンストラクタ
public AndroidGameSurfaceView(Context context) {
    super(context);
    getHolder().addCallback(this);
    setClickable(true);
}

@Override
public boolean onTouchEvent(MotionEvent event) {
    switch(event.getAction()) {
    case MotionEvent.ACTION_DOWN:
        // 画面にふれたときの処理
        mTouchX = event.getX(); // mTouchXはタッチｘ座標を保存するためのフィールド変数
        mTouchY = event.getY(); // mTouchYはタッチｙ座標を保存するためのフィールド変数
        return true;

    case MotionEvent.ACTION_MOVE:
        // 画面にふれたまま指を動かしたときの処理
        return true;

    case MotionEvent.ACTION_UP:
    case MotionEvent.ACTION_CANCEL:
        // 画面から指を放したときの処理
        return true;
    }

    return super.onTouchEvent(event);
}
```

課　　題

画面をドラッグすると、指の動きに併せてキャラクターが移動するプログラムを作成しましょう。

(１)タッチ座標を格納する変数を宣言します。以下の変数を適切な場所に追加しましょう。

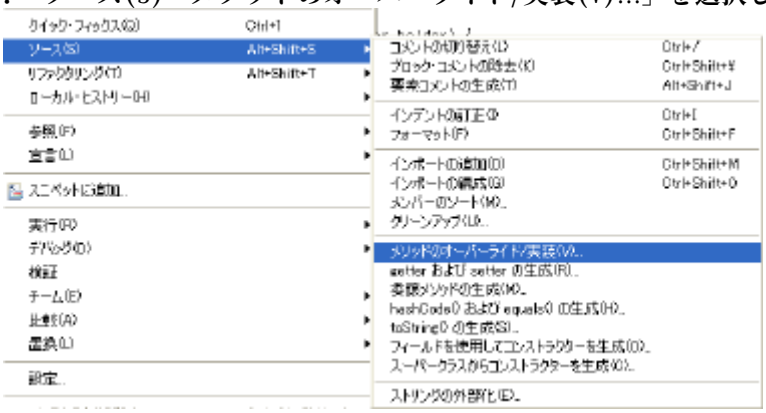
```
private float   mTouchX, mTouchY;
```

onTouchEventメソッドでは、この変数にタッチ座標を設定するだけにします。具体的な処理は、doUpdateメソッドに記述するようにします。onTouchEventメソッドに具体的な処理を記述すると、doUpdateとonTouchEventメソッドの２カ所に内部処理が分散してしまい、メンテナンス性が低下してしまうからです。

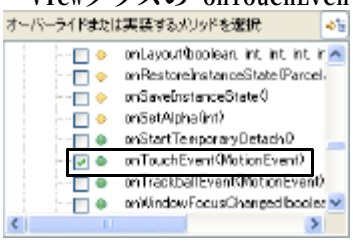
(２)onTouchEvenメソッドをオーバーライドし、定義します。

１．Eclipseソースエディタの onTouchEventメソッドを作成したい行で右クリックします。

２．「ソース(S)→メソッドのオーバーライド/実装(V)...」を選択します。



３．「オーバーライドまたは実装するメソッドを選択」ダイアログが表示されます。
Viewクラスの"onTouchEvent(MotionEvent)"にチェックを入れ、



４．以上で、onTouchEventメソッドのひな形が作成されます。

(３)同期処理(排他処理)を行うようにします。

onTouchEventメソッドとゲーム処理は別スレッドのため、同時に実行される場合があります。そのため、同じ変数を書き換えながら読み取る、といったことが起こり得ます。このような場合、同期を取ることにより、問題が起こらないようにすることができます。複数のスレッド間で同期を取るようにすると、同時に実行されては困る処理をひとつずつ行うことができるようになります。たとえば、タッチイベントとゲームの内部処理(タッチ情報をもとに行う処理)を同時に行わず、片方を処理した後、もう片方の処理を行う、ということができます。

Javaでは、同期を取る方法のひとつとしてsynchronizedブロック(またはメソッド)があります。synchronizedで囲まれた部分は同時に実行されません。また、synchronizedブロックに突入する際、クラスの変数(オブジェクト)にロックを掛けることができます。ロックは１つのスレッドしか掛けられないので、ロックの要求が複数あった場合、ロックを掛けられなかったスレッドは待ち状態となります。この性質を利用し、タッチイベントと内部処理の「タッチ情報をもとに行う処理」を同時に実行できないようにします。

１．onTouchEventメソッドを以下のように変更します。

```
@Override
public boolean onTouchEvent(MotionEvent event) {
    synchronized(mLock) {
        // TODO Auto-generated method stub
        return super.onTouchEvent(event);
    }
}
```

synchronizedブロックに突入する際に、変数mLockにロックが掛けられ、ブロックから抜けるときに解除されます。doUpdateメソッドにもすでに同様のブロックがあります。ロックは１つしか掛けられないので、複数ロックの要求があったとき＝onTouchEventとdoUpdateメソッド(のブロック)が同時に実行されようとしているときは、一方は実行され、もう一方はロック解除まで待つようになります。

２．onTouchEventメソッドの適切な場所に、タッチイベントを処理するプログラムを追加しましょう。

(４)キャラクター座標を管理する変数を宣言します。以下の変数を適切な場所に追加しましょう。

```
private float   mCharaX, mCharaY;
```

(５)doUpdateメソッドの適切な場所に、タッチ座標をもとに行う処理を追加しましょう。

(６)doDrawメソッドを変更し、キャラクター座標にキャラクター画像が表示されるようにしましょう。