Android Game Programing Tips

株式会社タイトー
ON!AIR事業本部
コンテンツ企画部　開発課
山田俊一

自己紹介

- 2005年入社　その後の経歴
  - Vodafone/Softbank　DoCoMo（Java）
    - iアプリ
    - JavaTMアプリ、S!アプリ
  - Webサイト管理（html, perl, MySQL...）
    - なんだかいろいろ

ゲームってどうやって動いてんの？(1/2)

■ パラパラアニメ

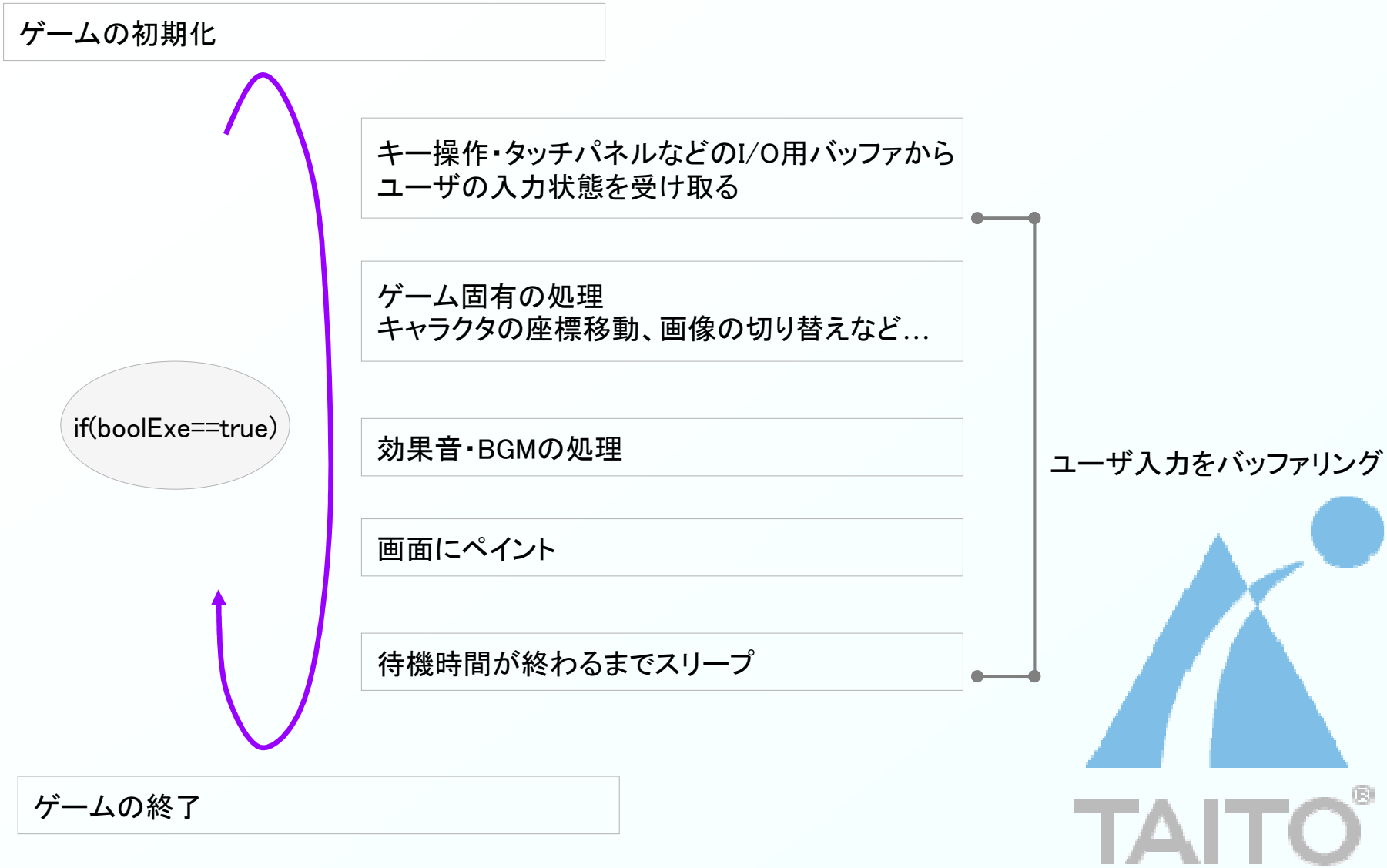
- 1秒間に xx回のループ ： xx fps(frame per second)
  - 例：20fps＝1ループは50ms
    各種計算処理に15ms -> 35msスリープ
    各種計算処理に80ms -> スリープ無し，処理落ち発生中.

■ 実例

- アーケード・コンシューマは 30fps or 60fps
- 携帯アプリは10～30fps
- 傍目に見てカクカクが目立つのは 8fps 以下
- TVアニメは24fpsで製作

ゲームってどうやって動いてんの？(2/2)

■ 簡易図



AndroidのView、いろいろあるよ(1/4)

- View
  - onDraw(Canvas oCanvas)ループ
  - ○　UIキットのようなものを利用し、複数機種へのレイアウト崩れを防止
  - ×　処理が重い
  - やっぱりツール向け

```
//=== method ========================================
/**
 * View#onDraw()
 **/
//===================================================
public void onDraw()
{
  /*
   * バッファ処理・数値計算など、各種処理を記載
   */
  update_userInput();
  update_game();
  update_sound();

  //画面への描画を実行する
  //onDraw()終了時に次のonDrawが発生＝無限ループ状態
  invalidate();

  //スリープ処理
  game_sleep(lFinishTime - lStartTime);

  return;
}
```

AndroidのView、いろいろあるよ(2/4)

- SurfaceView
  - サブクラス作成、別スレッドでループ管理
  - ○　描画処理が軽い！
  - ×　アプリのライフサイクルが分かりにくい
  - ×　UIキットが使えない(かも)
  - ゲーム向き

AndroidのView、いろいろあるよ(3/4)

- SurfaceView, sample

```
public class MyView extends SurfaceView implements SurfaceHolder.Callback
{
  //=== implements ==================================
  /**
   * implements : surface生成
   */
  //=================================================
  public void surfaceCreated(SurfaceHolder oSH)
  {
    m_oDrawThread = new DrawThread( SurfaceHolder );
    m_oDrawThread.start();
    return;
  }
  ////////////////////////////////////////////////////
  /**
   * SUB CLASS
   *   DrawThread
   *
   *   Surfaceを用いた描画用スレッド
   **/
  ////////////////////////////////////////////////////
  class DrawThread extends Thread
  {
    public void run()
    {
      while(boolGameExecute == true)
      {
        //ゲーム処理
      }
      return;
    }
  }
}
```

AndroidのView、いろいろあるよ(4/4)

- OpenGLを組み込む場合...
  - SurfaceView＋OpenGL描画用スレッド
    - 更に設計が煩雑になる
    - ライフサイクルが分かりにくい
    - Canvas, Drawableを用いた描画処理ができず、iアプリ・S!アプリからの移植性が低い
  - SDK1.5にある新API GLSurfaceView に期待？
    - OpenGL ESアプリケーションを作りやすくする　...らしい

Android固有の問題と解法
システム全般の巻

システム編:ライフサイクル注意点

- アプリをしっかりと終了させよう！
  - Backボタンに任せる
  - Activity#finish()を発行する
    - ゾンビアプリが多いよー
- 割り込み処理に対応しよう！
  - Activity#onPause()
  - Activity#onResume()
  - 何も処理しないと、電話が来てもBGMが鳴り続けるよ

システム編:バイブレーション機能

- manifest.xmlの修正も必要

```
<manifest>
  <uses-permission android:name="android.permission.VIBRATE" />
</manifest>
```

```
/**
 * View#onDraw(Bundle)
 */
public void onCreate( Bundle savedInstanceState )
{
    //バイブレーション機能を取得
    oVibrator = (Vibrator)getSystemService(Context.VIBRATOR_SERVICE);
}
/**
 * lTime : 0～負数で強制停止
 */
protected void setVibrator(long lTime)
{
    if( oVibrator != null )
    {
      if(lTime > 0)
      {
        oVibrator.vibrate(lTime);
      }
      else
      {
        oVibrator.cancel();
      }
    }
    return;
}
```

システム編:ユーザI/O (1/2)

■ キー入力

- onKeyDown() ： return true →キー入力を無効化
  - Backボタン、Homeボタンを無効化したい場合に。

```
public boolean onKeyDown(int iKeyCode, KeyEvent oKeyEvent)
{
  if(oKeyEvent.getAction() == KeyEvent.ACTION_DOWN)
  {
    // KEY : BACK
    if(iKeyCode == KeyEvent.KEYCODE_BACK)
    {
        m_iKeyBuf |= KEY_SYS_BACK;
        return true;
        //return super.onKeyDown(iKeyCode, oKeyEvent);
    }
  }

  return super.onKeyDown(iKeyCode, oKeyEvent);
}
```

システム編:ユーザI/O (2/2)

■ トラックボール

- ゲームによっては、値そのまま処理すると違和感が出てくるため
  加速度をバッファリングするなど、何かしらの補正をつけるといいかも
  - たとえば...
    0.1f -> 1 pix移動
    0.3f -> 5 pix
    0.5f -> 10 pix
    1.0f -> 50 pix

■ タッチパネル

- 重い！激重！
  - 効果的な対応策、無し...

システム編：Manifest.xml Tips

- configChanges
  - 端末の縦横回転時に実行アプリケーションの再起動を行わない

- screenOrientation
  - Portrait/Landscapeの固定を行う

```
<activity android:name=".HogeHogeAppName"
          android:label="@string/app_name"
          android:configChanges="orientation|keyboardHidden"
          android:screenOrientation="portrait">
```

Android固有の問題と解法
音の巻

サウンド編:再生形式(1/2)

■ Wav　MP3

- 再生できるけど、再生開始に時間がかかる（毎回バッファリングしてるっぽい？）
  - BGMのループが途切れやすい
  - シューティングのショット音など、SE(効果音)の連続再生ができない

- 無理に連続再生すると、致命的エラーでアプリが落ちる！
  - それどころか、OSすら落とす

→ダメじゃん！

サウンド編:再生形式(1/2)

■ OGG

- Ogg Vorbis
- 圧縮音声フォーマット、ライセンスフリー

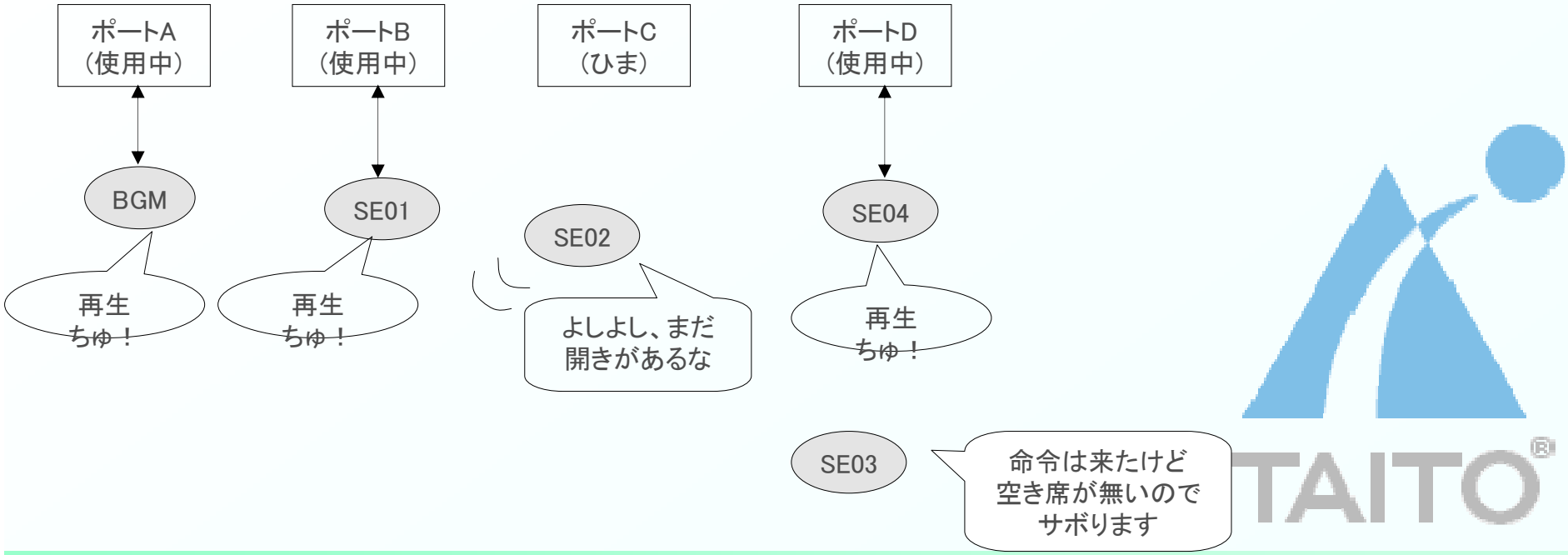
- MediaPlayer#start() ですぐに再生開始！
- SEの連続再生もいけるね！

→SEもBGMもOggで！

サウンド編：G1限定のお話、かも

- SEを頭から再生しなおす場合、MediaPlayer#seekTo(0)
  - わざわざpause()する必要無し

- ポート（=同時に可能再生数）は4つ！
  - API側には制限が無いので、ソフトウェア側でポートを管理
    - 勝手に空きポートにサウンドデータが割り振られるので、最大同時再生数が 4 になれば良い
    - implements OnCompletionListener, onCompletion()
  - iアプリもS!アプリもポート4つなので、経験者なら楽！



Android固有の問題と解法
グラフィックの巻

グラフィック編:こまごまとしたこと

- 処理の重さ＝描画範囲
  - 描画範囲、描画する画像領域が広いほど重い
    - onDraw()を用いる場合、Invalidate(x,y,w,h)で書き換え範囲を限定するのも手
  - Png、無駄なヘッダが入っていると処理が重い
    - たぶん、律儀に全ヘッダ解析＋エンコードしてるのだろう
- 画像描画はDrawalbeを使おう
  - createBitmapはDrawableよりもメモリを消費するため、2D画像回転など、限定した処理で用いるほうがよい
- 拡大縮小はアンチエイリアスがかかるので綺麗
  - ただし、すごく処理が重い
  - 前もって拡大縮小済みの画像を持っていたほうがいい
- 透過pngだけでなく、アルファpngにも対応
  - 面白い演出ができるかも

その他

Eclipse＆エミュレータとうまく付き合う

■ 重い！
- PC最高スペックだとさくさく？
- 補完機能は使い方次第では便利。親切すぎて邪魔にもなる…
- 使い慣れたエディタで作成→Eclipseでバグチェックがいいかも

■ 再現度が高い
- エミュレータでバグ、処理落ちがあるなら、実機でも同じことが起きる
  - タッチパネル関連、ホントどうしてくれよう
- トラックボールの操作は
  - エミュレータではアウトプットは1.0f単位
  - 実機でのアウトプットは0.1f程度
- 3Dに関してはエミュレータと実機で異なる部分が多々あるため、実機中心で開発しよう

ログは便利

- だが...
  - Eclipse経由で、他人のアプリのエラーログ確認ができしてしまう

- 製品版アプリではandroid.util.Logを外しやすい設計を！
  - ログ出力専用関数を作っておき、間に挟んでおくなど

```
public void outLog(String sMes)
{
  //*
  Log.w(TAG, "[warn] "+sMes);
  //*/
  return;
}
```

今後の為に：機能を切り分けよう

- プラットフォームによって異なる
  - ユーザデータのセーブ＆ロード
  - 画像表示
  - サウンド

- どのプラットフォームでも共通
  - ゲームアルゴリズム

以上！

- ご静聴ありがとうございました