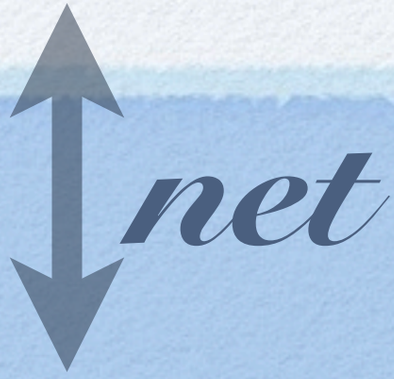
Google App Engine



iPhone & Android

さきら

すまべんのかんさい #7　@大阪 / 2010.02.06

おしながき

- モバ絵ツールのはなし
- この話のネタ
- Google App Engineの紹介
- 実装についての注意

モバ絵ツール

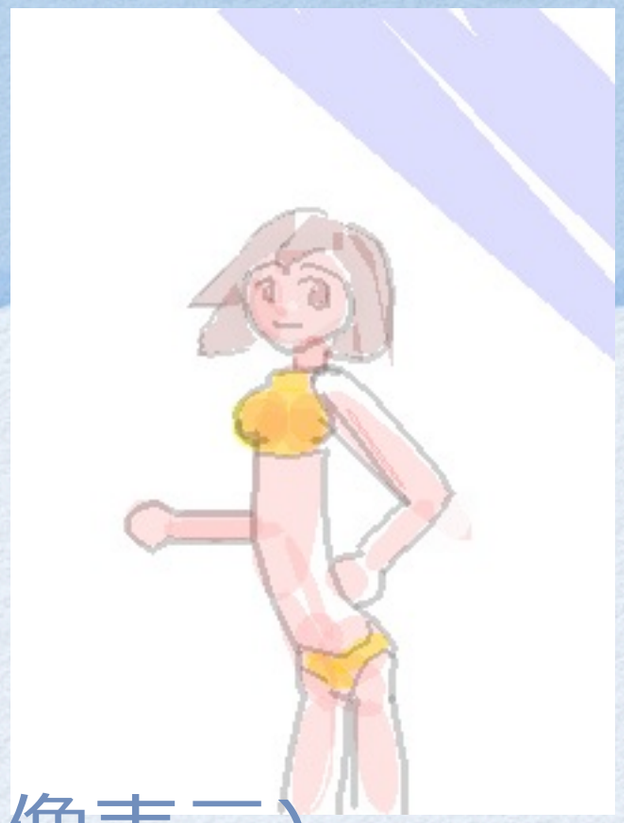
「え? モノ絵ですか?」

- あんなタイトルで お絵描きツールの話
- Linux Zaurusでも、
  - もともとはRubyスクリプト触ってた人
  - でも待ってても誰もモノ絵ツール作ってくれる気配がなかった

Petite Peinture (for Zaurus)

- 2003/06～
- Linux Zaurus
  (SL-A300～SLC3x00)
- C++/Qt
- ユーザーインターフェースは独自
- 筆圧感知、多機能
- 「小さい絵」のフランス語 愛称「ぷちぱん」

Pocket Peinture



- 2004/05～
- AH-K3001V (DDI Pocket(当時))
- JavaScript (クライアント:キー入力, 画像表示)
- Ruby / C++ (サーバ:セッション管理, 描画)
- 12キーだけで操作 (かなり無理があった)
- ある意味今回の話の契機
- 「画材をポケットに入れて持ち運ぼう」

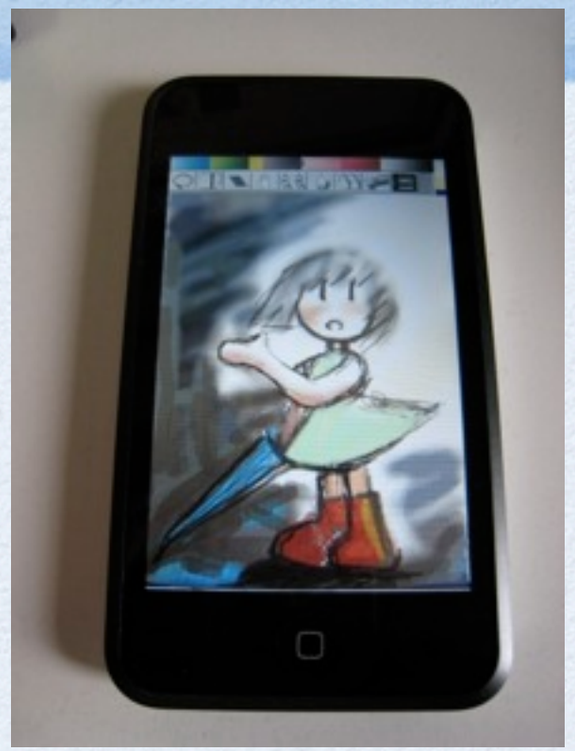
SM Peinture

- 2006/08～
- UIQ3 (Sony Ericsson M600iなど)
- Java / CLDC + MIDP + NokiaUI
- タッチパネルあるしレイヤーも
  付いてるので案外まとのもに使えます
- ただ、UIQ3のユーザ数が少ない
- 「Symbian MIDP Peinture」の略

Mollo Peinture

- 2006/11～
- Palm OS 4以降 (WristPDAがターゲット)
- C++, C
- モノクロだけどレイヤーもあるよ
- 小さい画面に描くための
  ユーザーインターフェース
- 「easy paint」のフランス語

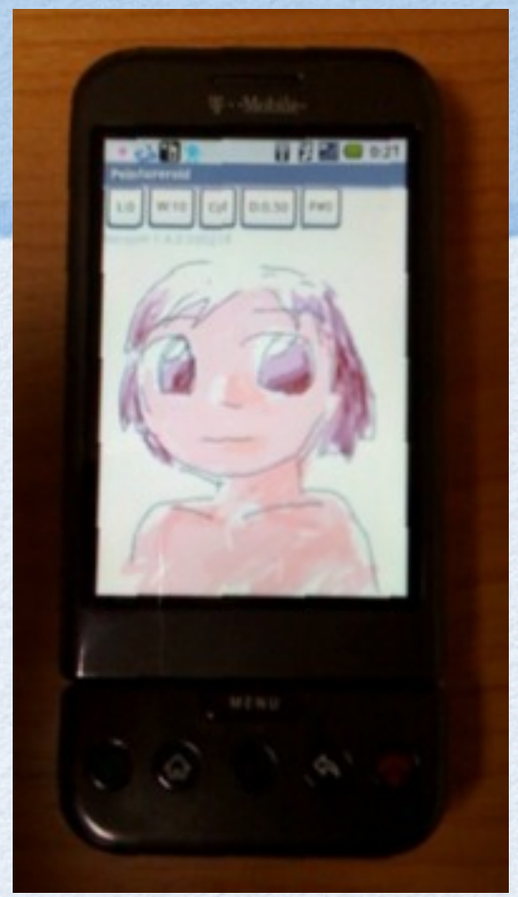
Petite Peinture for iPhone



- 2008/07～
- iPhone, iPod touch
- Objective-C++

- オリジナルぷちぱんの移植からスタートし、マルチタッチやスムーズなズームなどiPhone OSの機能に
- オリジナル版の機能の全ては移植しきれていない
- 最近「ぷちぱん」と言えばこっち

Peintureroid



- 2008/11～
- Android
- Java
- 筆圧感知やマルチタッチ(G1で確認)にも対応 (でも不完全)
- もともとは「Android Peinture」という名前だった
- 「Peinture + roid」で「ぱんちゅえろいど」

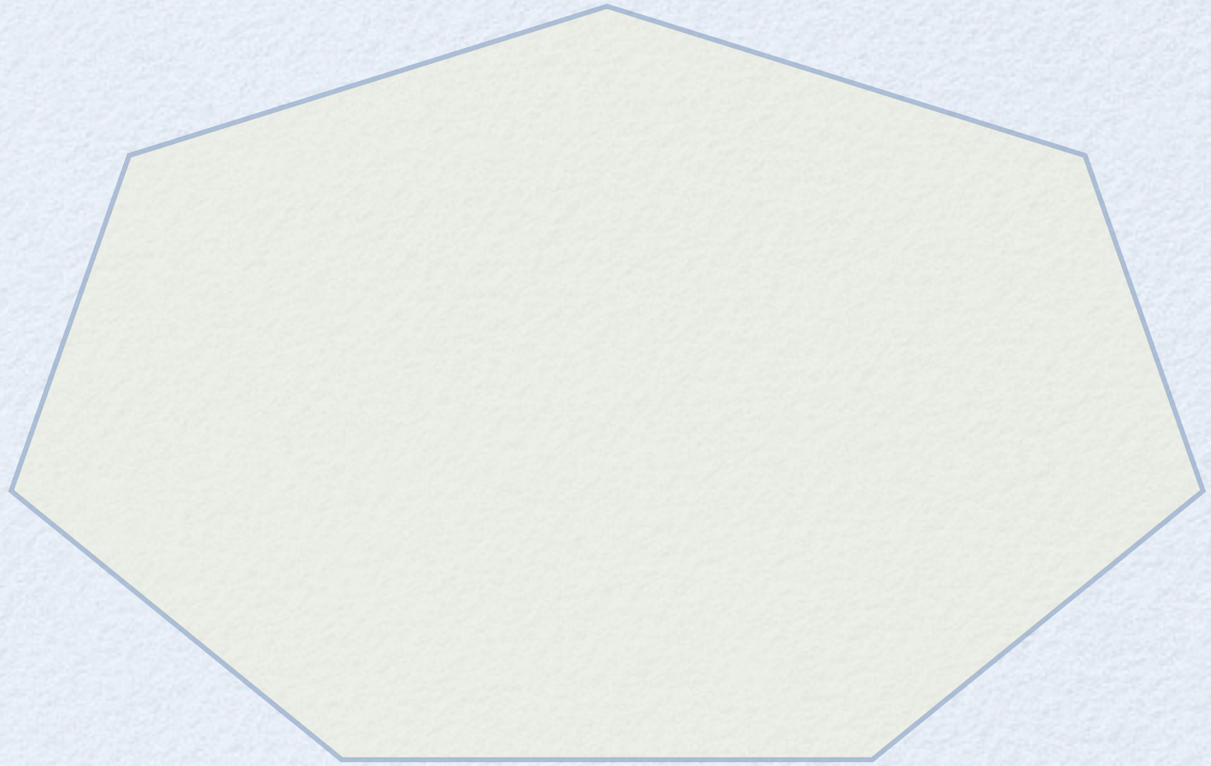
モバ絵としてのプラットフォーム

	Palm	Linux Zaurus	iPhone	Android
スタイラス	○	○		○ 一部のみ
筆圧感知		○		○
マルチタッチ			○	○
なめらかな線		○	△	○
描画演算速度		△	○	○

今日紹介するものの概要

今日は何のはなしだっけ?

- クラウドとの連携で絵を描く



クラウド上の
サーバ

クライアント

ストロークに含まれる情報

- 色、幅、濃度、レイヤー番号、時刻
- キャンバスID
- ユーザID
- 頂点座標の配列

これらの情報をひたすら雲の上に蓄積していく

ソフトの特徴と名前

- サーバ上で一つの絵を成す
  (クライアントだけでも一応動作)
- ストロークが描かれるたびに
  空を飛んでいくイメージ
- Peinture over the Air → 「Air Peinture」

Air Peinture クライアント

- iPhone, iPod touch用クライアント
  - Objective-C++
  - ストロークはC++ (SPStroke.{h,cpp})
- Android用クライアント
  - Java
  - ストロークも当然Java (SPStroke.java)

Air Peinture サーバ

- ストロークはユニコード文字列として表現
- CGIのような仕組みでサーバは実現可能
  - 入力も文字列、出力も文字列
- Google App Engineを採用
  - Pythonで構築 (たった233行)

Google App Engine

Google App Engine

- 2008年に公開
- ウェブアプリホスティングサービス
  - 個人でも利用可能 (無料からスタート)
    - 認証に携帯メールを利用
- PythonとJavaならすぐに利用可能
  - RubyなどもJRuby経由で動くらしい

Google App Engine 料金

- とりあえずGoogleのアカウントと携帯電話があればウェブアプリ作成できます
- 「課金が有効にされていないアプリケーションには、永続的ストレージが 500 MB、月に約 500 万ページビューの処理に十分な CPU と帯域幅が割り当てられます。」
- この辺りは日本語で書いてあるのでぜひ読んで
- 課金を有効にして使用リソースの上限を上げることはできます

Google App Engine 書き方

- CGIだと思って書けばOK

```
def get(self):
    self.response.out.write('Hello world!')
```

で「Hello world!」が表示されます

- こんな感じでgetとpostを実装すればOK
- cronもあるそうです (試してません)

Google App Engine データ保存

- ファイルに保存とかいう概念ありません
- Datastoreと呼ばれるリレーショナルじゃないデータベースを使います
- INSERT, DELETE, SELECT *しかできないデータベースみたいなもの

GAE Datastore Properties

- 各項目のプロパティ
  - String, Boolean, Integer, Float, DateTime, User, Email, Referenceなど19種類
    - 並び換え順序も定義済み
  - String(文字列 str, unicode)やByteString(バイナリ列)は500バイトまで
  - Text(文字列 str, unicode)やBlob(バイナリ列)に長さ制限はないけど、フィルタや並び換えに使えない

GAE Datastore 制限事項

- 一行は1MB以内
  - JPEGやPNGなどの画像ファイルはBlobに入れれば大丈夫
- 1回のクエリで件数は1,000件まで、30秒以内
- 順序指定は一つの項目だけ
- 制限は多いけど何とか工夫してみましょう

実装についての注意

(クライアント編)

Air Peinture for iPhone

- 去年の宿題 (1)

```
@interface Hoge {
@public
   NSString* str;
}
 @property(retain) NSString* str;
```

として Hoge* foo = [[Hoge alloc] init];

としたらどっち?

foo.str = another_str; ?

foo->str = another_str; ?

だから self.str = another_str; も意味あります

Air Peinture for iPhone

- 去年の宿題 (2) (Objective-C++)

```
class CPPClass {
   ....
};
```

```
@interface ObjCClass {
   CPPClass cppc;
}
```

としたらcppcではデフォルトコンストラクタで生成されたCPPClassインスタンスが代入されるようになりました。(iPhone SDKのバージョンにより)

Air Peinture for iPhone

- 非同期HTTP通信

どこにどんな
HTTP通信するのか

```
NSURLConnection *conn;
conn = [[NSURLConnection alloc]
          initWithRequest:request
                 delegate:my_delegate
         startImmediately:YES];
```

通信中の
データ受信とか
受け持ち

これで非同期通信大丈夫。

通信自身は面倒見てくれる。

Air Peinture for iPhone

- 通知される側 (主スレッド)

```
[[NSNotificationCenter defaultCenter]
 addObserver:self
    selector:@selector(get_stroke:)
        name:@"StrokeGot"
      object:agent];
```

- 通知する側 (通信スレッド)

```
NSNotification* notifi = [NSNotification
  notificationWithName:@"StrokeGot"
                object:self];
[[NSNotificationQueue defaultQueue]
     enqueueNotification:notifi
            postingStyle:NSPostWhenIdle];
```

Air Peinture for Android

- 非同期HTTP通信

```
class HTTPConnectDelegate extends Thread {
  HttpURLConnection http_connect;
  .....
```

普通にThread作って
その中で普通にHTTPの
通信するだけなんで省略。

Air Peinture for Android

- 通知される側 (主スレッド)

  特になし

- 通知する側 (通信スレッド)

```
Handler handler = new Handler();
handler.post(new Runnable() {
  public void run () {
    canvas.get_stroke();
  }});
```

実装についての注意

(サーバ編)

Air Peinture Server

- 文字列処理ばかりなのでPythonにしました
- やるべきこと
  - ストロークを受信したら蓄積 (INSERT)
  - 蓄積したストロークをまとめて送信 (SELECT)
  - キャンバスIDやユーザIDの識別とかは要るけど、基本的にはこんだけ

Air Peinture Server

- ストローク用テーブルの定義

```
class StrokeProperties(db.Model):
    canvas_id = db.StringProperty()
    user_id = db.StringProperty()
    server_time = db.IntegerProperty(default = 0)
    rawtext = db.TextProperty()
```

Air Peinture Server

- SQLとGQLはどうせ別モンなんで、
  GQL使わない方針で。こんな感じ。

DBのテーブル名

```
query = StrokeProperties.all()
query.filter('canvas_id = ', canvas_id)
query.filter('server_time >= ', last_time)
query.order('server_time')
return query.get()
```

さいごに

- せっかく通信機能付いてるんだから
  サーバに繋ごうよ
- とりあえず始めるなら
  Google App Engine は便利っぽいよ