iPhoneでも ぷにゅう してみよう

さきら

すまべん! 関西#3　@大阪 / 2009.05.16

本日のメニュー

- 自己紹介
- iPhone OSプラットフォーム
- 「ぷにゅう」って何?
- iPhone OS開発環境
- アプリケーション配布
- プログラミング上の注意
- ぷにゅうの物理モデル
- その他注意点
- ぷにゅうの今後とまとめ

自己紹介

- TK-80に憧れた小学生時代
- 6809, i386, C, Javaな学生時代
- Java, Ruby, Mathematicaなお仕事
- C++, C, Javaな趣味

iPhone OSプラットフォームの特徴

- iPod touch/iPhone
- 480×320 3.5inch LCD
- 11.5mm×62.1mm×12.3mm / 133g
- ARM11 VFP搭載 (8個までのfloat同時計算!)
- 412MHz / 533MHz (iPod touch 2G)
- メモリ116MB / ストレージ 4G～32GB

iPhone OSプラットフォームの特徴

- 無線LAN (802.11b/g)
- 加速度センサー
- マルチタッチ
- カメラ / GPS
- OpenGL ES 1.1対応
- 開発言語はObjective-C 2.0
  - Android G1に搭載されてるキーボードや地磁気センサーはなし

「ぷにゅう」って何?

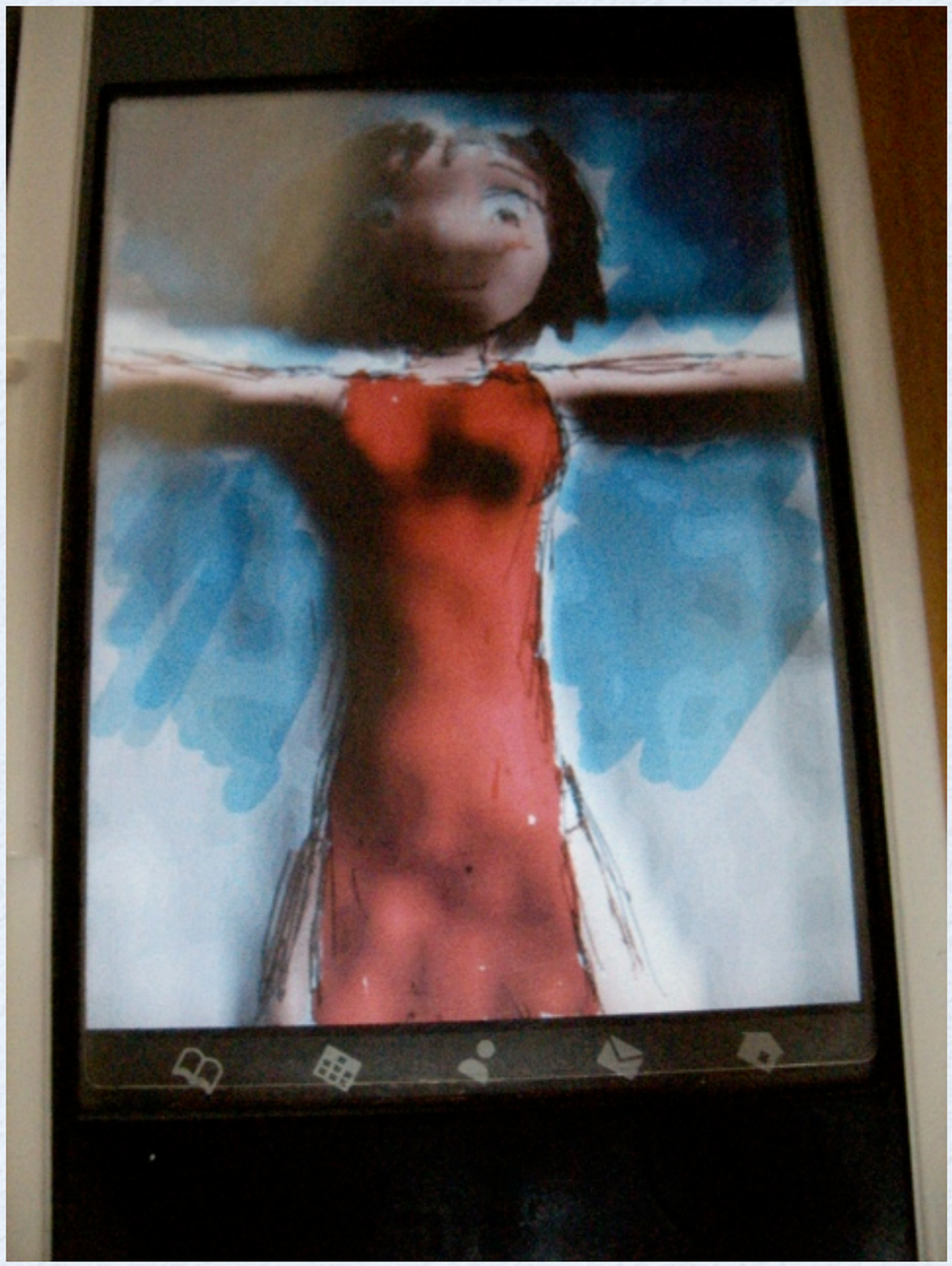
- Project PNUE (**P**arametrized **N**atural **U**neven skin **E**xpression)
  - 西暦2000年に発足。**ほっぺたを「ぷにゅう」**
  - 軟体同士や軟体と剛体の接触時の軟体の変形
  - 重力や圧力を考慮した軟体の変形
  - などなどの変形に伴う疑似レンダリング
  - これらを「少ないリソースで」実現する
- 最初の成果物, Zaurus版「ぷにゅう」は2004～2005年に開発/公開

「ぷにゅう」って何?

物理計算もレンダラーも
float, doubleを使わず
全部自前で計算

マルチタッチはないけど
圧力感知はあるよ

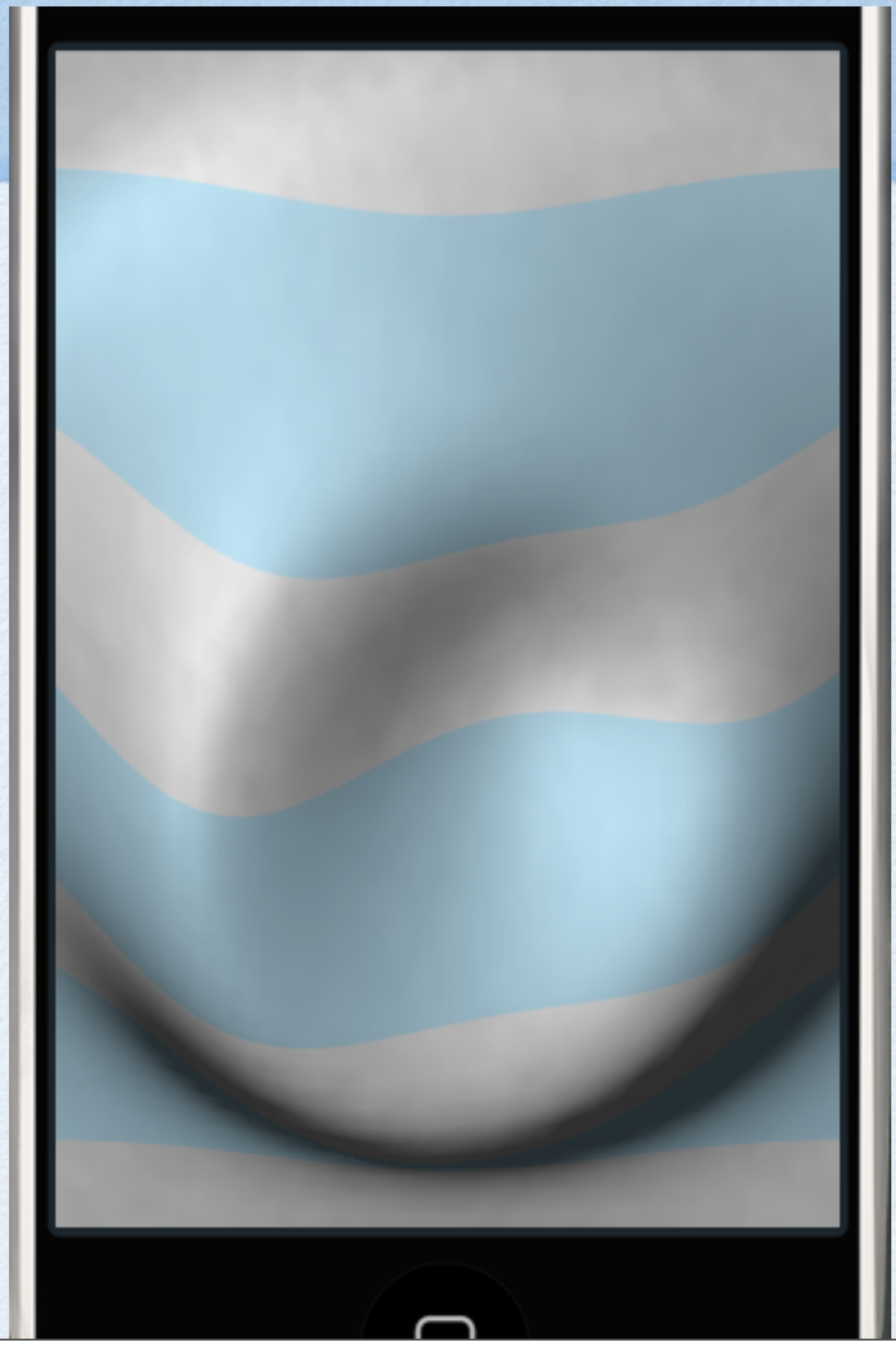
簡易モデラーも搭載



「ぷにゅう」って何?

- Zaurusでは浮動小数点演算も使えなかった
  - iPhoneにはVFPがある!
- Zaurusではフレームバッファに直に描画
  - iPhoneにはOpen GL ES 1.1がある!
  - つまりハードウェアで結構処理してくれる

「ぷにゅう」
って何?



開発環境

- jailbreakingについては省略
- Appleから公式のiPhone SDKが無料で提供
  - Mac OS X/Leopard/intel でしか動きません
  - XcodeとInterface Builderが主
  - エディタ, デバッガ, シミュレーター, パフォーマンス解析ツールなどなど
  - 基本キーバインディングはEmacs風

iPhone API

- APIリファレンスとかガイドラインとか公式ドキュメントは充実してます → ただし英語
- Objective-CやCocoaの部分についてはブログなどにも結構多かったので、そっちが有効
- iPhone APIの公式日本語版も徐々に出てきた
  → 全然足りないけど
- NDA解除以来、iPhone API絡みのブログなども増えてきました

参入するチャンスかも!!

開発環境 (デバッグとか)

- iPhoneシミュレーター
  - エミュレータじゃないよ、シミュレーターだよ
  - 実機との動作が違う事が結構あります
  - 加速度センサーは使えません
  - マルチタッチは指二本までな上に特定の動きのみ

実機での動作確認とか

- Appleに年間約1万円のお布施
  - もろもろ込みで考えると高くないという考えも?
  - USBで繋ぎます
  - デバッガや解析ツールなどはシミュレーター同様
  - 電子鍵が必要。
  - 開発用/adhoc配布用/AppStore配布用各々にプロビジョニングプロファイルが必要(正直面倒)

アプリケーションの配布

- ad hoc配布
  - アプリケーション毎に50台までの実機にインストール可能
  - 一般のウェブサイトとかからダウンロード可能
  - ベータ試験とかに向いてる?
- AppStore配布
  - 有償配布にはこっちが必要
  - Appleによる審査。Androidと違うよ (3〜7日?)
  - 忘れちゃいけない「Pending Contract問題」

プログラミングについて

- Objective-C 2.0　～基本はC言語～
  - iPhone用にはGCありません!
  - retain, release, autoreleaseと仲良くね
    - リファレンスカウンタがあるだけ C++ の new, delete よりマシ?
    - autoreleaseは「GCもどき」
  - [sakira joinTo:[SumaBen alloc] initWithId:3]];とかいう記法には慣れるしかない

Objective-C++でも大丈夫

- C++で書いてもいいんです!
- 拡張子を「.mm」にしてXcodeにimportするだけ!
- 拡張子「.m/.cpp/.mm」全部混在OK!
- Appleもちゃんと認めてます!
- iPhoneのAPI呼び出しの部分だけは諦めてObjective-Cにしましょう…

```
@interface Hoge {
  public:
    int bar;
}
@property int bar;
```

```
Hoge* foo = [[Hoge
alloc] init];
```

としたらどっち?

```
foo.bar = 1; ?
foo->bar = 1; ?
```

プログラミング (浮動小数点演算)

iPhoneにはちゃんとfloatやdoubleの並列計算が可能なFPU載ってます。

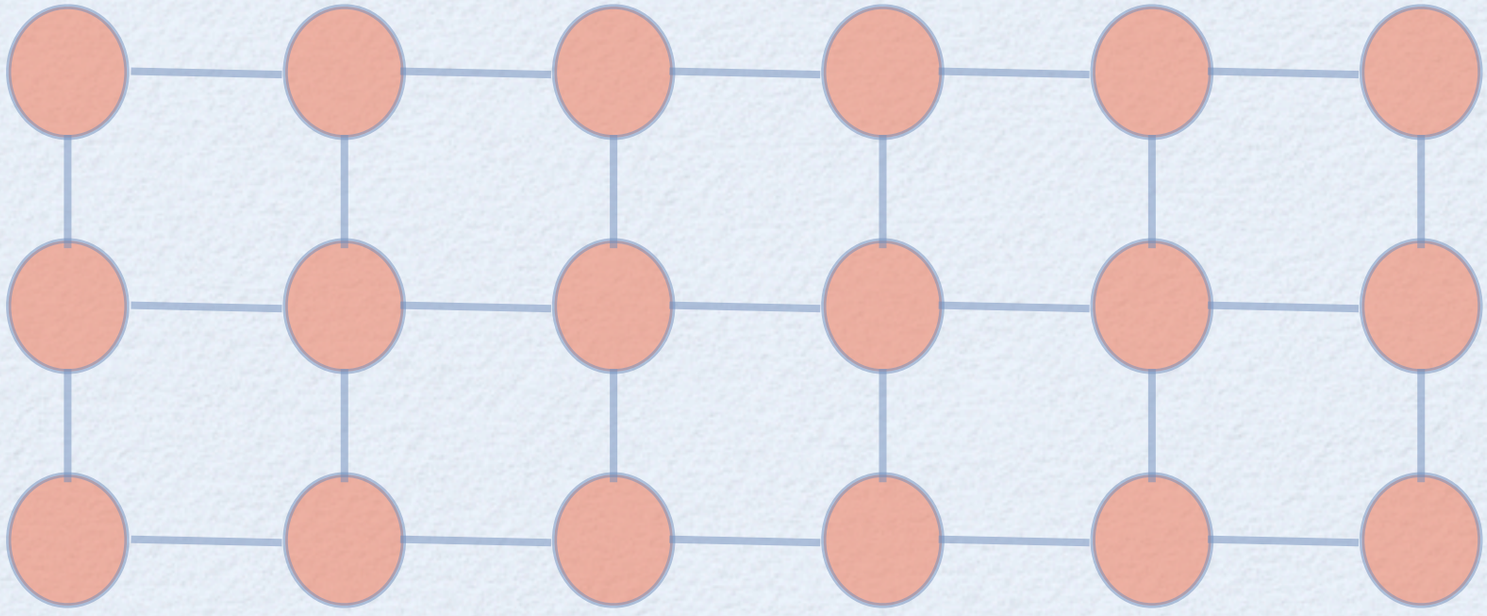
- でも注意点もあります
  - Thumbは使うと遅くなります
    - Thumbとは: ARM社のページから引用
    - コードは小さくなるけど、float, doubleを頻繁に使うのなら避けるべし
  - Xcodeの中のコンパイルオプションでon/off可
- それにしても cmath の pow(a, b) とかおかしい...

OpenGL ES 1.1

- 組み込み機器用のOpenGLがOpenGL ES
- 1.1ではハードウェアによる固定機能パイプライン
  - 2.0ではプログラマブルシェーダ。面白そうだけど今日の話とは関係ないので略
- 他の3DCG APIと多分考え方は一緒
- 頂点座標をGLFloat[]とかで指定、法線ベクトルを指定、ポリゴンを指定、光源を指定、マテリアルを指定、テクスチャを指定、視点と視野を指定 → 描画命令実行
- アフィン変換とかも使えるよ!

物理モデル

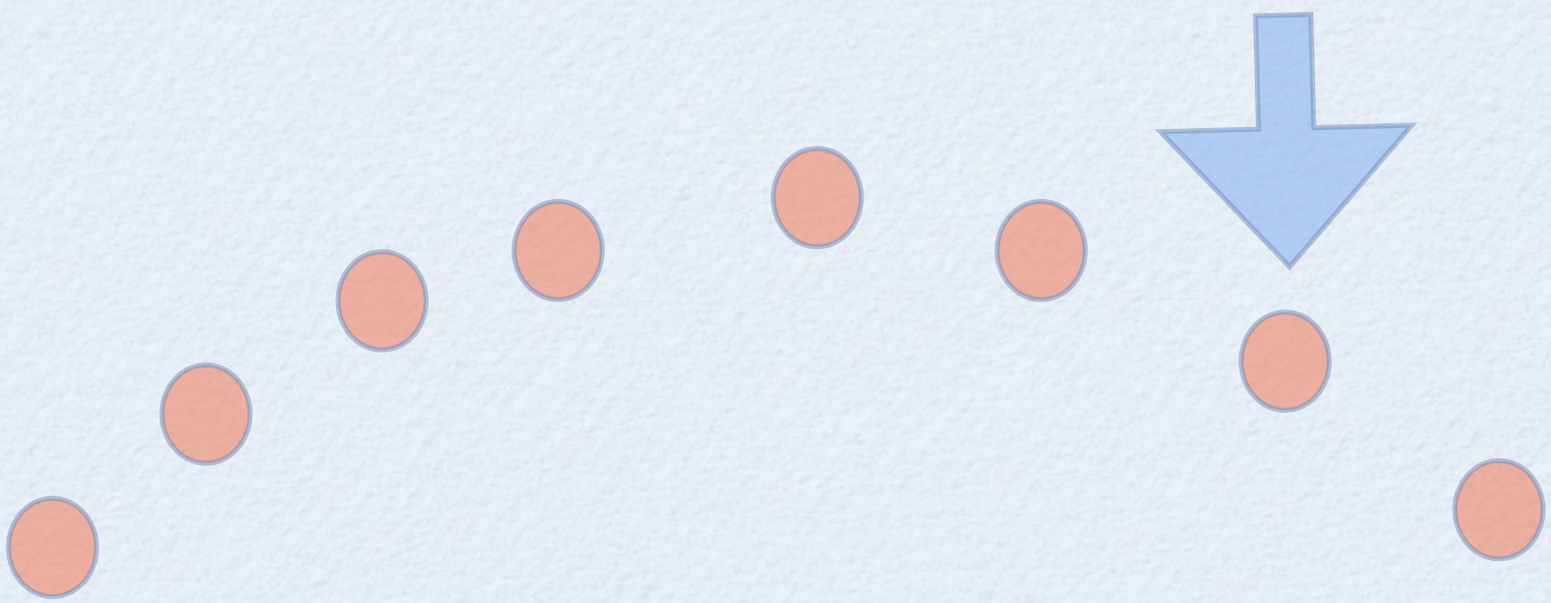
- $F = -kx$, $F = ma$
  - 今回使ってる運動方程式は基本的にはこれだけ
  - 単位系は統一しよう (s, m, kg)
    - ぷにゅうでは: 縦横各々0.2m, 高さ0.1m, 比重 800kg/m^3, 重力加速度 9.8 m/s^2
- ばね連結モデルを試してみた→使いものにならず



50×50×10のメッシュ

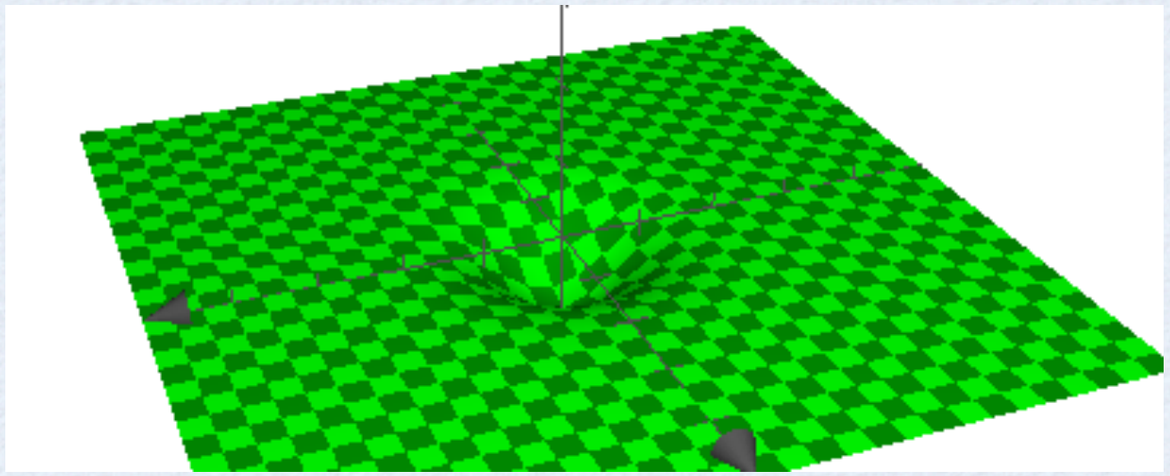
物理モデル

- 運動方程式の適用先をメッシュから変更
  - 重力加速度 → 物体の重心
  - タッチによる変形 → 最初にタッチした表面上の点
- 他の表面メッシュ(50×50)については、運動方程式で動く点からの距離に応じて
  - 近いとたくさんうごく：遠いとあまりうごかない



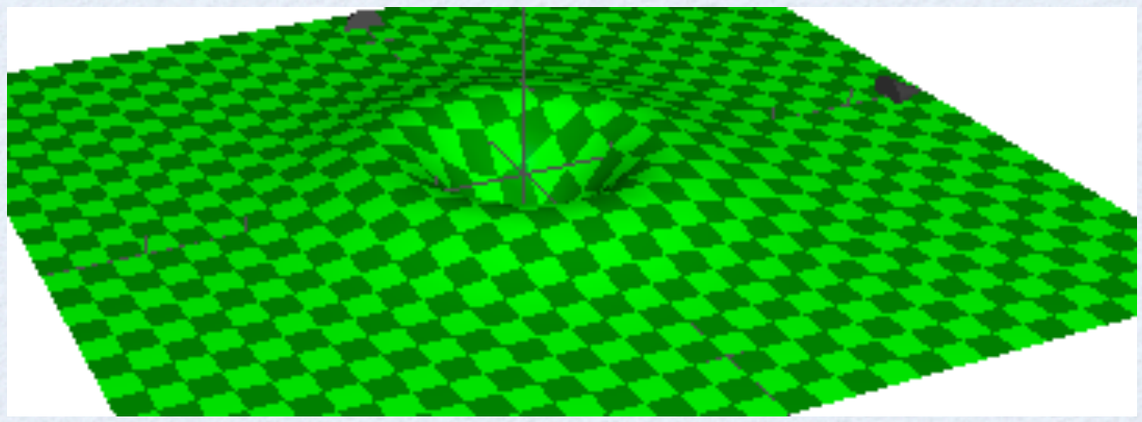
凹みの形

最初は $y = -a\ exp(-\ bx^2)$ 的な数式使ってました



でも凹むだけ
じゃつまんない。
つまめないし。

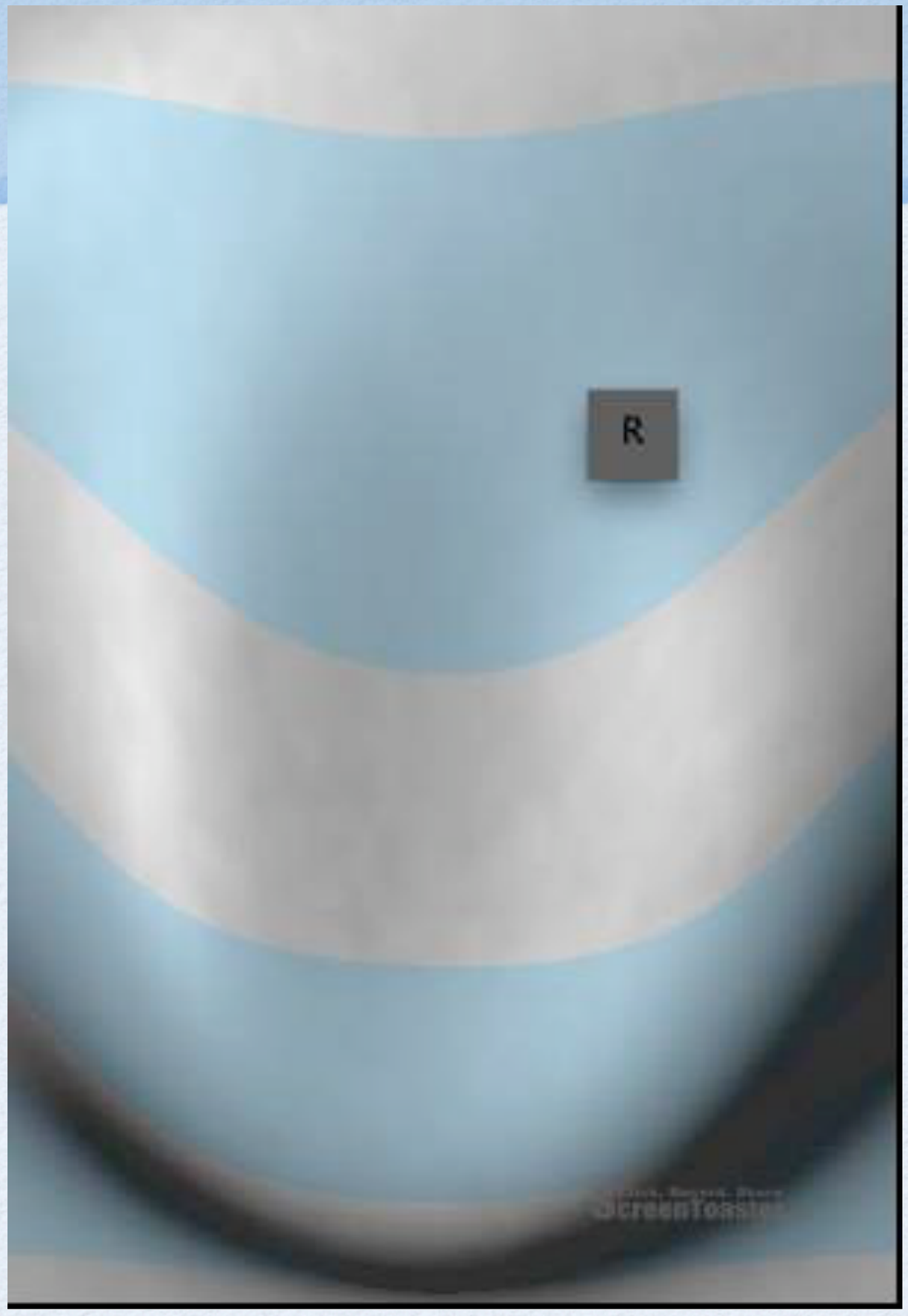
で、 $y = -a\ exp(-\ bx^2)+c\ exp\ (-\ dx^2)$ に変えました



凹むけど、
周りはその分
盛り上るの。

体積一定になるように a, b, c, d を設定

で、摘める
ようになる



加速度センサー

- 実は簡単

```
- (void)applicationDidFinishLaunching:(UIApplication *)application {

  [[UIAccelerometer sharedAccelerometer] setUpdateInterval:(1.0 / AccelerometerFrequency)];
  [[UIAccelerometer sharedAccelerometer] setDelegate:self];
```

これして、

```
- (void)accelerometer:(UIAccelerometer*)accelerometer didAccelerate:(UIAcceleration*)acceleration {
  //Use a basic low-pass filter to only keep the gravity in the accelerometer values
  if (glView) {
    glView.accel0 = acceleration.x * FilteringFactor + glView.accel0 * (1.0 - FilteringFactor);
    glView.accel1 = acceleration.y * FilteringFactor + glView.accel1 * (1.0 - FilteringFactor);
    glView.accel2 = acceleration.z * FilteringFactor + glView.accel2 * (1.0 - FilteringFactor);
  }
}
```

これやって、

後はOpenGLのアフィン変換に渡してあげるだけ。

加速度センサー

シミュレーターでは加速度センサー効きません。

→こんな感じに対処すればいいかと。

```
  #if TARGET_IPHONE_SIMULATOR
    [_pnue_trans setAccelDirectionAt:0 toX:0.0 toY:1.0 toZ:0.0];
  #else
    [_pnue_trans setAccelDirectionAt:0 toX:accel0 toY:-accel1
toZ:accel2 * 0.5f];
  #endif
```

マルチタッチ

- UIViewの中で self.multipleTouchEnabled = YES; にしたら
- touchBegan, touchMoved, touchEndedに渡されるNSSet (成分はUITouchのインスタンス) に現在タッチ中の全タッチイベントが入ります
- タッチ中はUITouchのインスタンスの中身は変わりますが、そのポインタは変化しません

  → ポインタを追跡してどのタッチがどこに移動したか判別する事が可能です
- (残念ながら)後はガシガシ書くだけ!

ぷにゅうの今後

- より一般的な形状もサポート
  - だとしたら3Dモデラーも内蔵しなくちゃ
- やわらかさとか変えられるように
- 表面のすべすべさも変えられるように

- データ形式が問題
- ユーザーインターフェースもすごく問題

まとめ

- Objective-Cはそんなにこわくない
- OpenGL ES 1.1はコピペで対処 (行列の復習しよう!)
- 加速度センサーはコピペで簡単
- マルチタッチは難しくはないけど面倒かも