Cocos2d-xで作る物理演算ゲーム

軌跡の点線を入れてみる③編

= 2016年5月28日 =

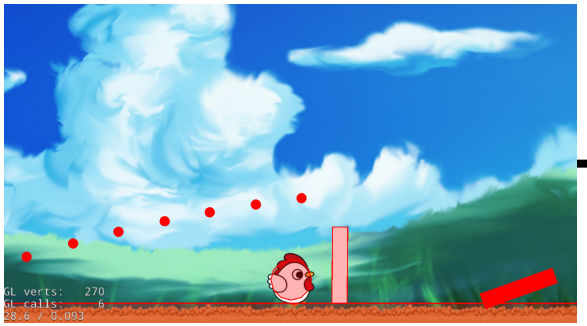
ここまでのソースプログラムはここ

http://monolizm.com/sab/src/AngryChicken18.zip

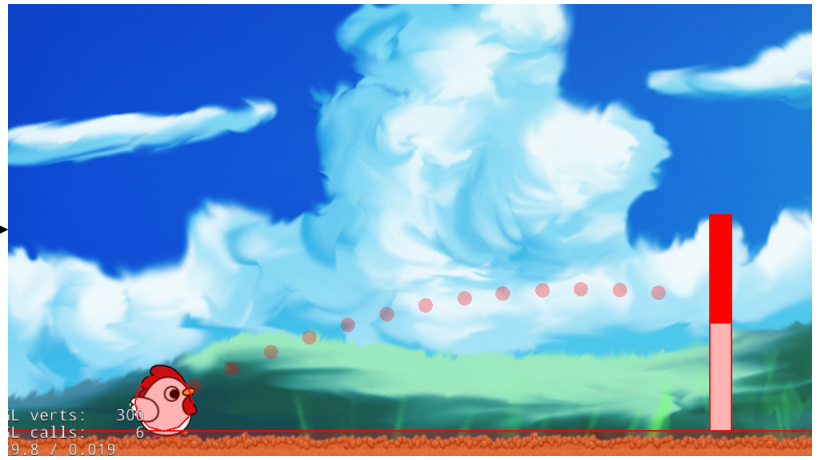
GETだぜ！

今回は次のプレイに移っても
前のゲームの軌跡が残る
ようにします。

コレ



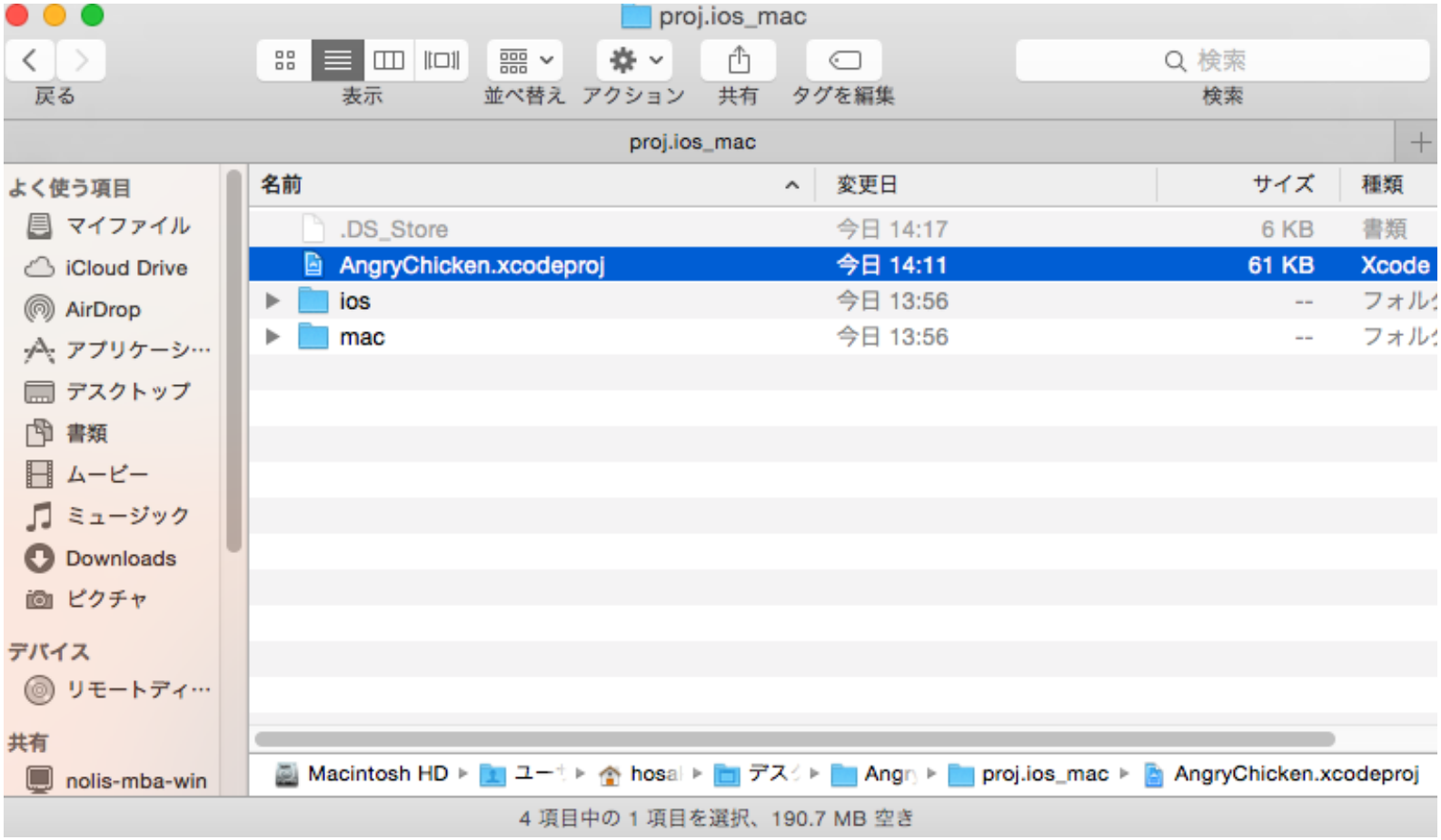
プレイ終了後・・・



最初からやると、前のゲームの点線が！

まずは起動しよう

AngryChickenをxcodeで実行。



まずは必要な処理を考えてみる

①前ゲームの点情報を保持する。
②その点情報をうっすらと表示する。

①前ゲームの点情報の保持

まずパッと思いつくやり方は以下の2つが考えられる。

1. 描画している点の位置情報はDrawNodeクラスが持っている。なので、
   次のゲームに行くときに、そのDrawNodeクラスを履歴用として
   ディープコピーして使う。

2. 履歴用のDrawNodeクラスを用意するところまでは同じ。
   「点情報を登録」を通常のDrawNodeクラスと同じように保持する
   処理を入れていく。

今回は2を採用。
1. はディープコピーするにはコピーコンストラクタや、copyメソッドをオーバーライドが必要なのでコストが高い。

ディープコピーとは？

例えば、同じクラスから生成されたインスタンスA，Bがある。A＝Bと代入したとき、クラスの持っているメンバ変数はコピーされる。ただし！メンバ変数がポインタメンバ変数だった場合、アドレスがコピーされるだけなので、両クラスとも同じ実態を参照する。

こうなると怖いことが今後起こる。Aを解放したとき、そのメンバがさす実態は解放される。まだBは生きているので、そのメンバから参照しようとした場合「解放されているので」不正な参照となってしまう。このレベルのコピーを「浅いコピー」という。

本来は実態も含めてコピーしたいわけで。そのコピーの方法を「深いコピー」という。「深いコピー」をするには「コピーコンストラクタ」や「copy」メソッドを自前でオーバーライドする必要がある。そして、DrawNodeは対応していないので、どうにもならない。

■詳しく知りたいなら簡単な例はここを参照↓
http://d.hatena.ne.jp/joynote/20100621/1277114747

こうなった

```
// on "init" you need to initialize your instance
bool MainGame::init()
{
~~~~~~~~~~~
~~~~~~~~~~~
~~~~~~~~~~~

    _commonInit();

    //履歴用0525
    this->_historyDraw = DrawNode::create();
    this->_historyDraw->retain();

    return true;
}
```

まずはinitメソッドで、履歴表示用のDrawNodeを生成。
メモリを自己管理するのでretain()で参照カウントを1上げとく。（つまり明示的にrelease()する必要がある）

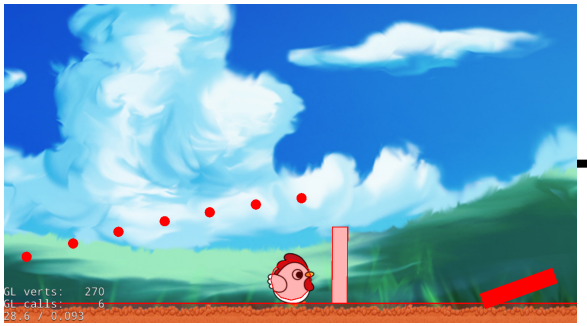
```
void MainGame::update(float delta)
{
～～～～～
～～～～～
～～～～～
            // 点線を一定間隔で打つ
            auto* draw = dynamic_cast<DrawNode*>(this->getChildByTag(LOCUS_OBJTAG));
            if ( draw != nullptr )
            {//
                if ( (*(bool*)draw->getUserData()) == true )
                {
                    auto* charSprite = (Sprite*)this->getChildByTag(CHAR_OBJTAG);
                    draw->drawDot(charSprite->getPosition(), 10, Color4F(1.f,0.f,0.f,1.f));

                    // 0525
                    _historyDraw->drawDot(charSprite->getPosition(), 10, Color4F(1.f,0.f,0.f,0.3f));
                }
            }
～～～～～
～～～～～
}
```

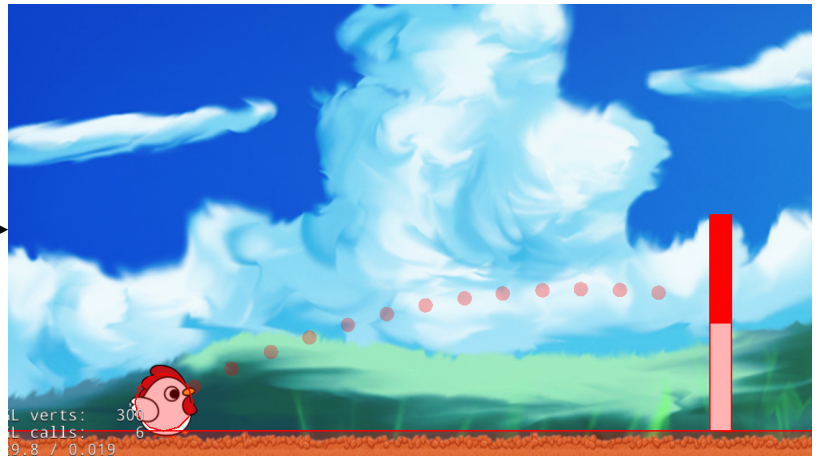
そして、現在点を打つ時に一緒に同じ位置にうっていく。
こうすれば実質コピーしたことと同じ。
本当は非表示にしたりした方がいいけど、同じ位置なのでとりあえずそんなことはほっとく。

実行してみよう！

こうなる



プレイ終了後・・・



最初からやると、前のゲームの点線が！

んが、、、、

ずっと履歴が残ってしまう･･･。とりあえず前Ｇのみだけにしたい。それは次回に。。。

まとめ

「浅いコピー」「深いコピー」の２つの違いを知っておくこと！C++に置いてはまりポイントの１つなので、その概念はしっかりと抑えておこう！

次回は物理演算Chipmunk

軌跡の点線を入れてみる④編

ここまでのソースプログラムはここ

http://monolizm.com/sab/src/AngryChicken20.zip

ご清聴ありがとうございました。