

# Cocos2d-xで作る物理演算ゲーム

## 軌跡の点線を入れてみる④編

= 2016年6月25日 =

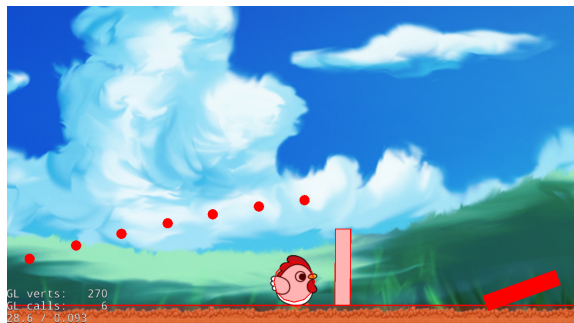
ここまでのソースプログラムはここ

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

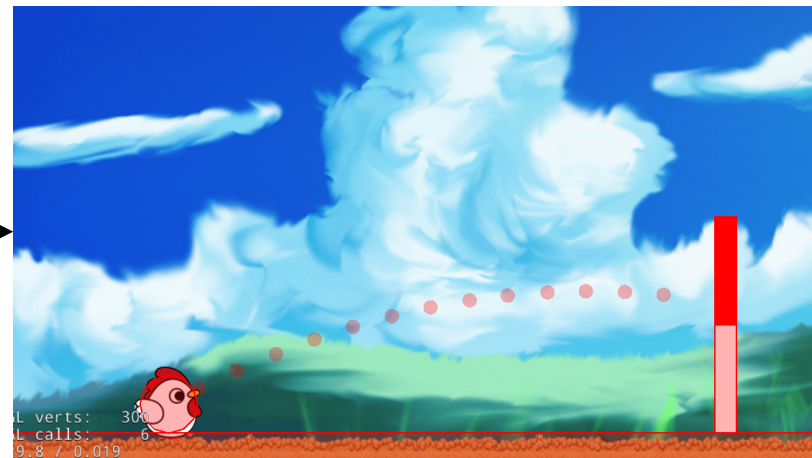
**GET**だぜ！

今回は  
前のゲームの軌跡がだけ残る  
ようにします。

# コレ



連続してプレイしても...



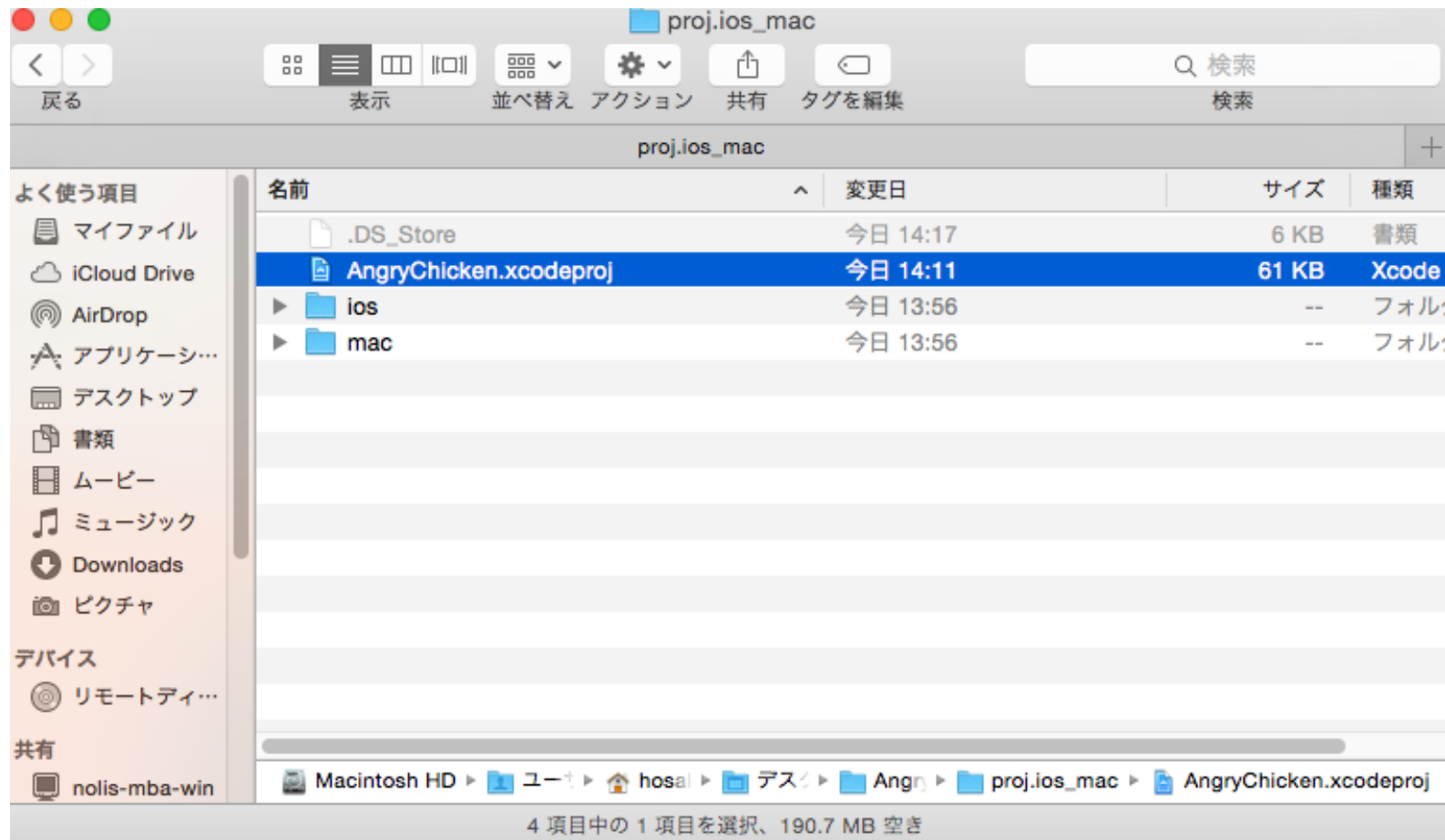
常に前ゲームの点線だけが！

# 前回の問題点

今までの軌跡履歴が全部残ってしまう・・・。  
それを今回修正します。

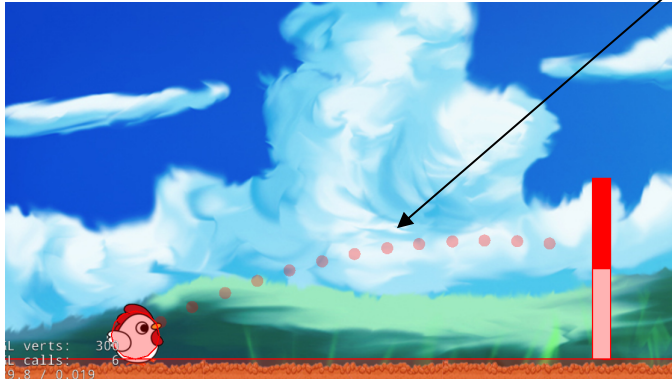
# まずは起動しよう

AngryChickenをxcodeで実行。



# まずは必要な処理を考えてみる

問題点を改善するには、「前ゲーム軌跡表示用」と「現在ゲーム軌跡記録用」の2つを同時に保持する必要がある。



前ゲーム軌跡表示用点線情報

さらにこれからプレイする軌跡  
情報も保存しなければならない。

# 処理の流れを整理

- ①ゲーム開始: 2つDrawNodeを準備(1つは「A」記録用、1つは「B」表示用とする)  
↓
- ②1ゲーム目はじく: 「A」が軌跡を記録(この時点ではBは未使用)  
↓
- ③2ゲーム目待機: 「A」の軌跡情報を表示、「B」を記録用にする  
↓
- ④2ゲーム目はじく: 「B」が軌跡を記録。Aはそのまま軌跡情報を表示  
↓
- ⑤3ゲーム目待機: 「B」の軌跡情報を表示、「A」を記録用にする  
↓
- ⑥3ゲーム目はじく: 「A」が軌跡を記録。Bはそのまま軌跡情報を表示  
↓
- (③～⑥を繰り返す)

つまり、2つのDrawNodeクラスを「記録用」「表示用」で切り替えながら、使用すればいい。

# 実装・・・の前にちょっと一呼吸

ここで一息して考える。このまま力技でDrawNodeクラスを2つ使って、処理ロジックをMainGameScene.cppに記載してもいいのだけど、ちょっとクラスとしてしっかりとまとめよう。

というわけで、  
DrawNodeの2つを保持して、それらをいい感じに入れ替えながら描画制御してくれるクラスを作ろう

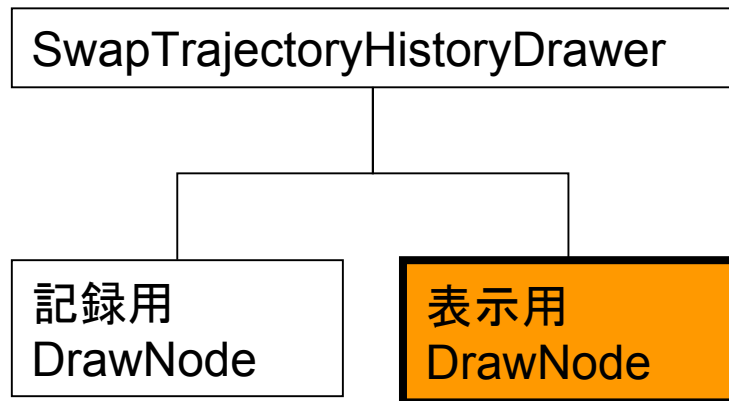
## ・SwapTrajectoryHistoryDrawerクラス

を作る

# こんな感じ

## ヘッダファイル

Nodeクラスを継承して描画ツリーに、ぶら下げることができるようにする。こいつ自体は何も描画しないが、こいつにぶら下がる「表示用DrawNode」が表示をする。



```
// SwapTrajectoryHistoryDrawer.hpp
// AngryChicken
#ifndef SwapTrajectoryHistoryDrawer_hpp
#define SwapTrajectoryHistoryDrawer_hpp

#include <stdio.h>
#include "extensions/cocos-ext.h"
#include "cocostudio/cocostudio.h"
#include "ui/CocosGUI.h"

class SwapTrajectoryHistoryDrawer : public cocos2d::Node
{
public:
    SwapTrajectoryHistoryDrawer();
    virtual ~SwapTrajectoryHistoryDrawer();

    static SwapTrajectoryHistoryDrawer* create();

    void swap();
    void pushDot(const Vec2 &pos, float radius, const Color4F &color);

private:
    int _currentIndex;

    enum {
        BUFF_0,
        BUFF_1
    };

    DrawNode* _historyDraw[2];
};

#endif /* SwapTrajectoryHistoryDrawer_hpp */
```

\* 実装部分はソースを見てね

# 主なメソッド説明

## **pushDot**

現在「記録用」として使われているバッファに点を記憶する。

## **swap**

記録用・描画用のバッファを切り替える。また自動的に記録用となったバッファはクリアされる。

このクラスをインスタンス化して、pushDotを複数回呼び出したのち、新しいゲームになるタイミングでswapをするという流れ。

# こんな感じ

```
void MainGame::_commonInit()
{
    // 初期化
    this->_touchEndedFlag = false; // 0218
    ~~~~~
    ~~~~~

    // 0617
    this->addChild(this->_swapTHDrawer);

    //////////////////////////////////////
    // 画面スクロール登録
    auto size = bg->getContentSize();
    ~~~~~
    ~~~~~
}
```

基本的にはもともと書いてあった  
DrawNode部分の記載を置き換えた  
くらい。

```
bool MainGame::init()
{
    ~~~~~
    ~~~~~

    // 0617
    this->_swapTHDrawer = SwapTrajectoryHistoryDrawer::create();
    this->_swapTHDrawer->retain();

    _commonInit();

    return true;
}

void MainGame::update(float delta)
{
    ~~~~~
    ~~~~~

    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));

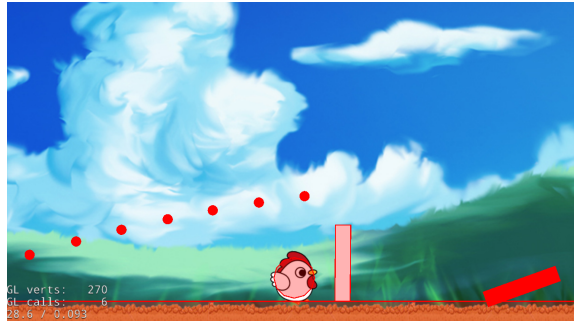
        // 0617
        this->_swapTHDrawer->pushDot(charSprite->getPosition(), 10,
        Color4F(1.f,0.f,0.f,0.3f));
        ~~~~~
        ~~~~~

        if ( velocity.length() < 1 )
        {
            // リセットする。
            _commonInit();

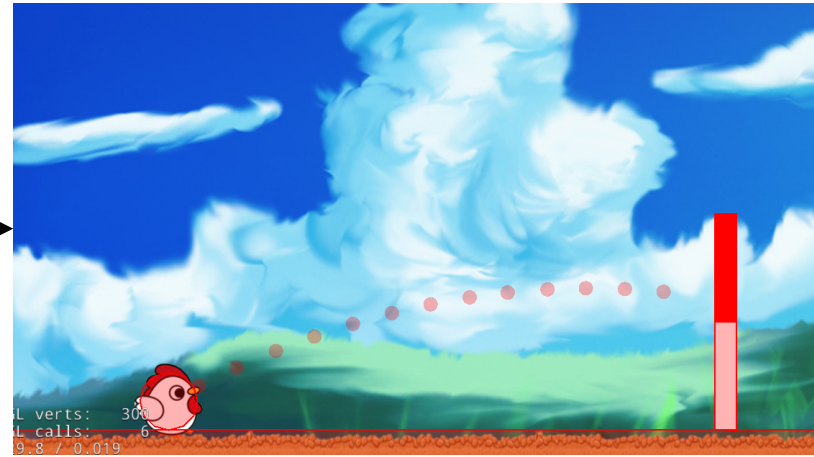
            // 0617
            this->_swapTHDrawer->swap();
        }
        ~~~~~
        ~~~~~
    }
}
```

実行してみよう！

# こうなる



プレイ連続しても・・・



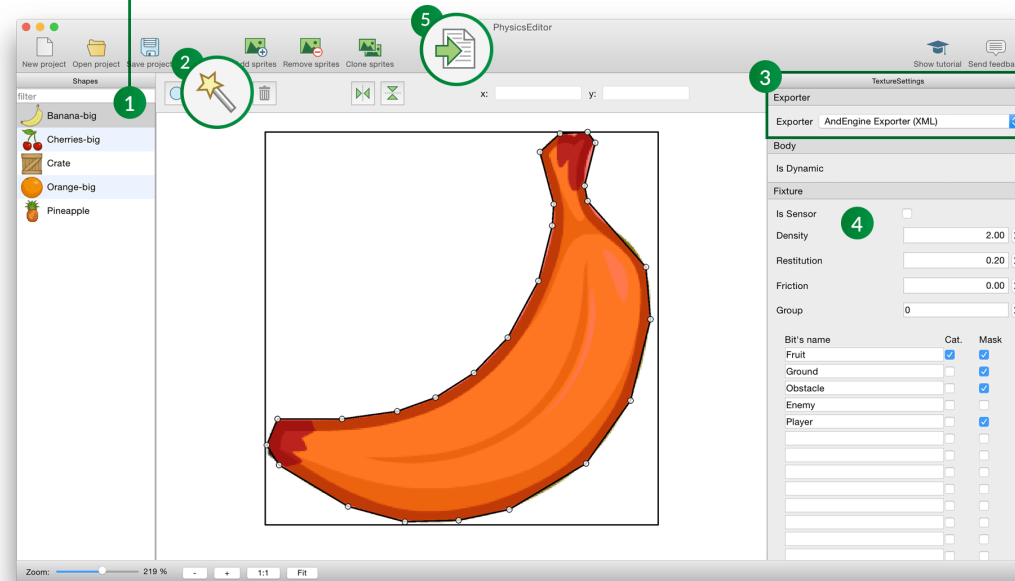
常に前ゲームの点線だけが！

# まとめ

描画ツリーのNodeを独自で作るときは、必ずNodeクラスを継承すること。

# 次回は物理演算Chipmunk

## Physics editor編



ここまでのソースプログラムはここ

<http://monolizm.com/sab/src/AngryChicken21.zip>

ご清聴ありがとうございました。