Cocos2d-xで作る物理演算ゲーム
壊す対象を置いて
当たり判定をとろう編

= 2015年11月28日 =

前回のあらすじ

チキンにタッチしてスライドして離す！

飛ぶ！

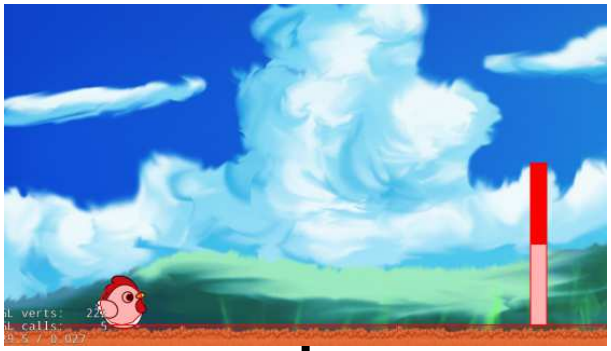
ここまでのソースプログラムはここ

http://monolizm.com/sab/src/AngryChicken11.zip

ＧＥＴだぜ！

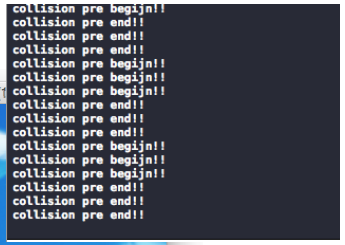
今回は壊す対象との
衝突判定の検知します。

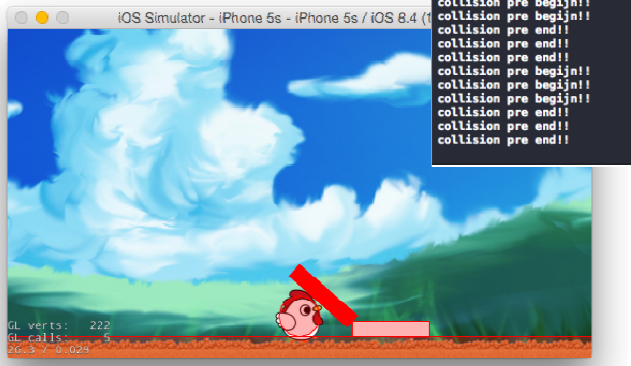
こんな感じを目指す

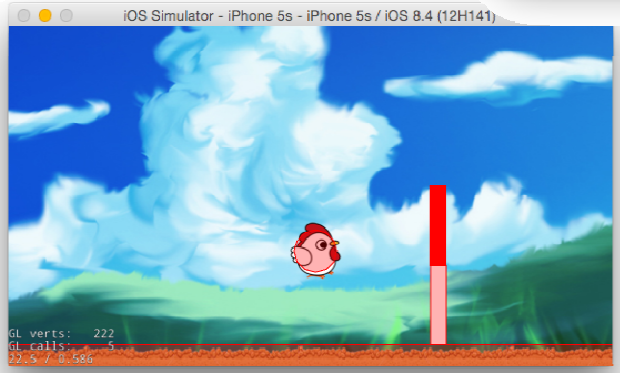


対象物を置いて。。。

ひとまずコンソールに衝突応答として
メッセージを表示



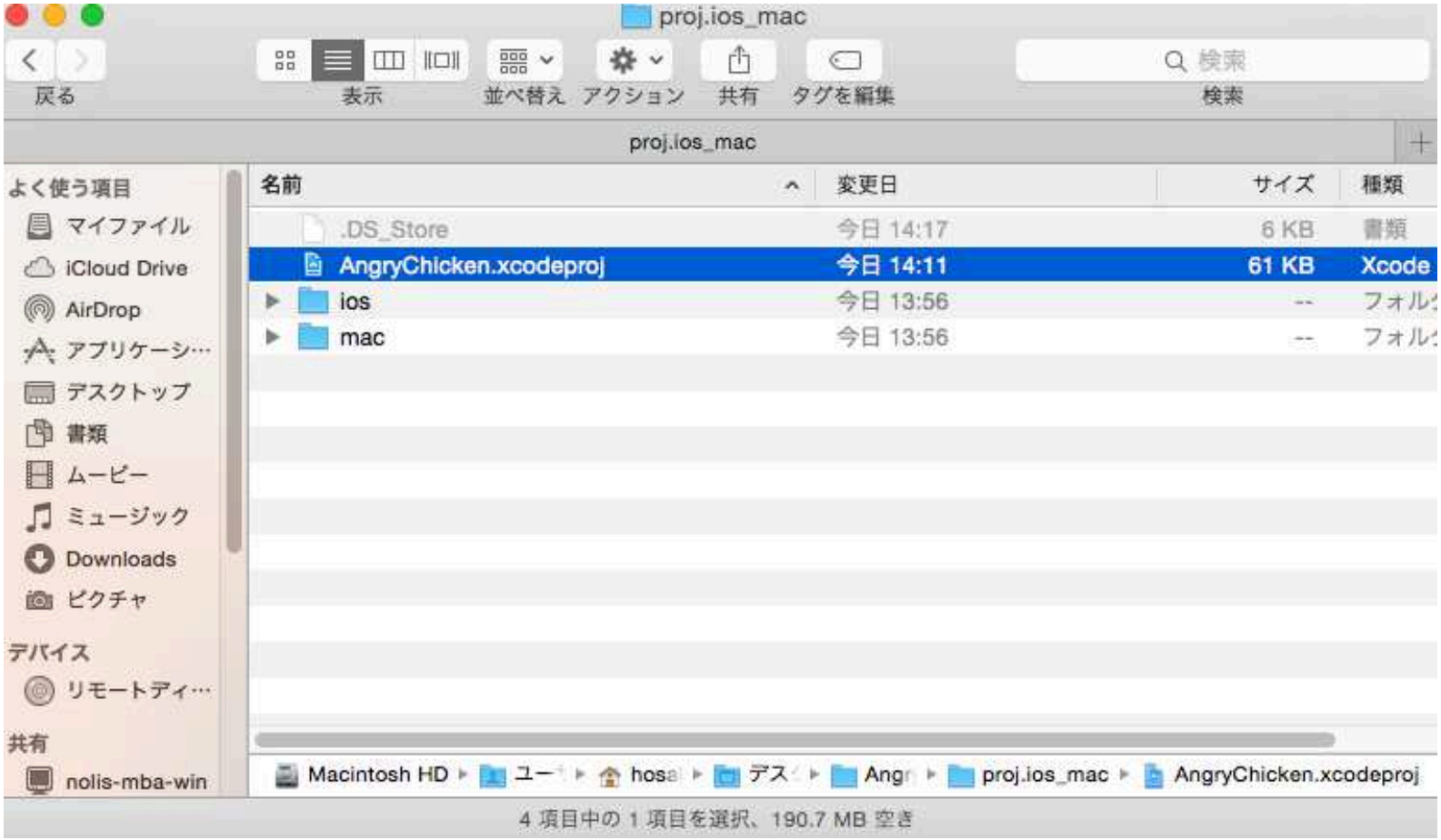




飛ぶ！

まずは起動しよう

AngryChickenをxcodeで実行。



①ブロックを作る

画像は用意せず矩形を自前で作ることにしよう。

```
// 1つ目
auto square = Sprite::create();                          // Spriteを生成
square->setTextureRect(Rect(0, 0, 30, 150));             // 矩形テクスチャ指定
square->setPosition(1000, 100);                          // 右の端に配置
square->setColor(Color3B(255,255,255));                  // 色
auto blockPb = PhysicsBody::createBox(square->getContentSize());          // 物理空間オブジェクト生成
blockPb->setMass(200.0);                                 // 重さ
blockPb->setContactTestBitmask(true);                    // とりあえず後ほど説明
square->setPhysicsBody(blockPb);                         // spriteと紐づけ
this->addChild(square);                                  // シーンに登録

// 2つ目　1つ目の上に載せちゃおう。
square = Sprite::create();
square->setTextureRect(Rect(0, 0, 30, 150));
square->setPosition(1000, 250);
square->setColor(Color3B(255,0,0));
blockPb = PhysicsBody::createBox(square->getContentSize());
blockPb->setMass(200.0);
blockPb->setContactTestBitmask(true);                    // 後ほど説明
square->setPhysicsBody(blockPb);
this->addChild(square);
```

このままだと地面、鶏、ブロックの干渉具合がイマイチ。なので、摩擦や重さなど諸々調整して想定する動きにいったん調整しよう。

調整できるパラメータの調整

剛体を生成する際に指定できる「Material」クラスこれで内部パラメータが決まる。

```
material = PHYSICSBODY_MATERIAL_DEFAULT;
material.density   = 10.0f; // 密度
material.restitution = 0.4f; // 反発係数
material.friction    = 10.9f; // 摩擦係数

auto* charaPb = PhysicsBody::createCircle(40, material);
```

剛体を生成後、属性を色々と設定する。「PhysicsBody」クラス

```
auto* floorPb =
        PhysicsBody::createBox(floor->getContentSize(), material);
charaPb->setMass(50.0f);
floorPb->setDynamic(false);
floorPb->setContactTestBitmask(true);
floor->setPhysicsBody(floorPb);
```

相互のパラメータで挙動の結果が変わるので適時微調整をしていこう。

衝突検知・・・のまえにこぼれ話

剛対のサイズを視覚化する。

```
Scene* MainGame::createScene()
{
    // 'scene' is an autorelease object
    auto scene = Scene::createWithPhysics();

    // 重力を設定
    auto* world = scene->getPhysicsWorld();
    world->setGravity(Vect(0, -150));

    // 'layer' is an autorelease object
    auto layer = MainGame::create();

    //  11.22 物理オブジェクトにを可視的にしてくれるデバックモード
    scene->getPhysicsWorld()->setDebugDrawMask(PhysicsWorld::DEBUGDRAW_ALL);
    ～～～～～～
```

これでSpriteとの関係性や当たり判定の状況など、様々な面でやくにたつ。

うっすらと赤い半透明の矩形がそれ。

②衝突検知メソッド登録

衝突判定リスナーをイベントディスパッチャーに登録

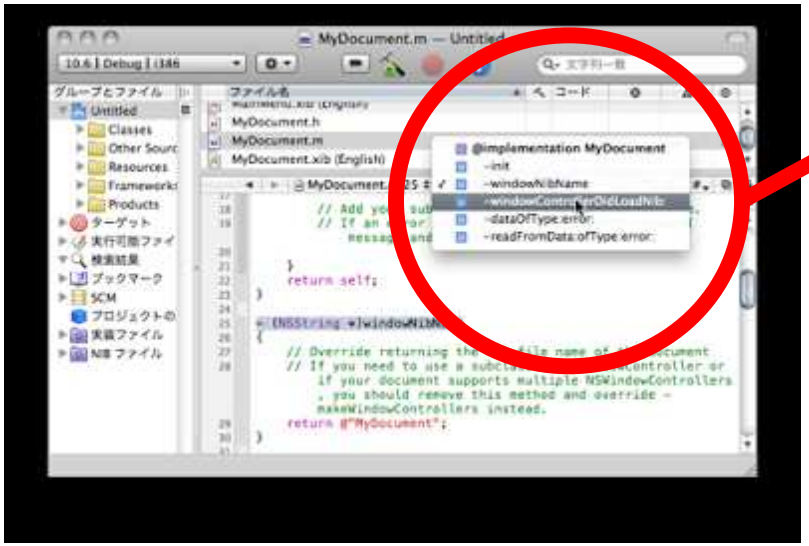
```
//////////////////////////////
// 物理演算衝突検知メソッド登録 11.22
auto contactListener = EventListenerPhysicsContact::create();
contactListener->onContactBegin = CC_CALLBACK_1(MainGame::_onContactBegin, this);
contactListener->onContactPreSolve = CC_CALLBACK_2(MainGame::_onContactPreSolve, this);
contactListener->onContactPostSolve = CC_CALLBACK_2(MainGame::_onContactPostSolve, this);
contactListener->onContactSeparate = CC_CALLBACK_1(MainGame::_onContactSeparate, this);
this->getEventDispatcher()->addEventListenerWithFixedPriority(contactListener, 10);
```

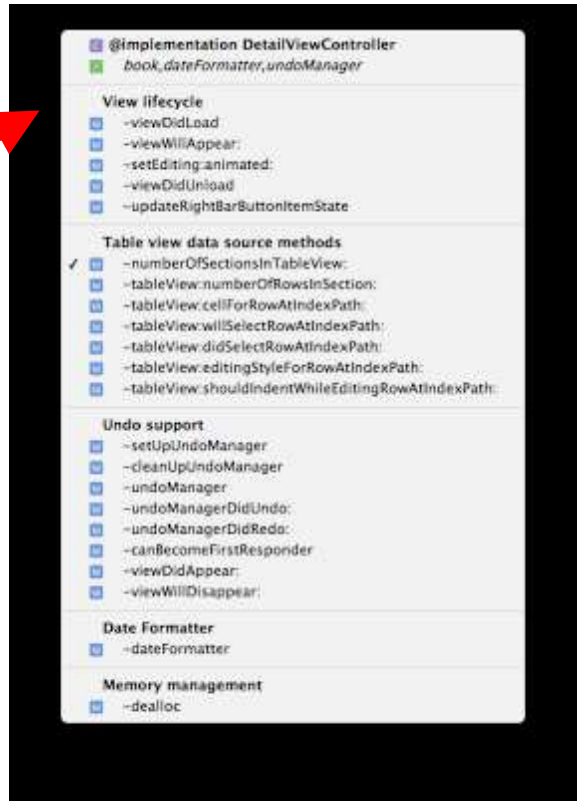
衝突検知をするメソッドを定義　＊もちろん.hにも浅間を描こうね。

```
#pragma mark Collision call

bool MainGame::_onContactBegin(PhysicsContact& contact)
{
    log("collision begijn!!");
    return true;
}

bool MainGame::_onContactPreSolve(PhysicsContact& contact, PhysicsContactPreSolve& solve)
{
    log("collision pre begijn!!");
    return true;
}

void MainGame::_onContactPostSolve(PhysicsContact& contact, const PhysicsContactPostSolve&
solve)
{
    log("collision pre end!!");
}

void MainGame::_onContactSeparate(PhysicsContact& contact)
{
    log("collision end!!");
}
```

またこぼれ話。みんな使ってますかね？

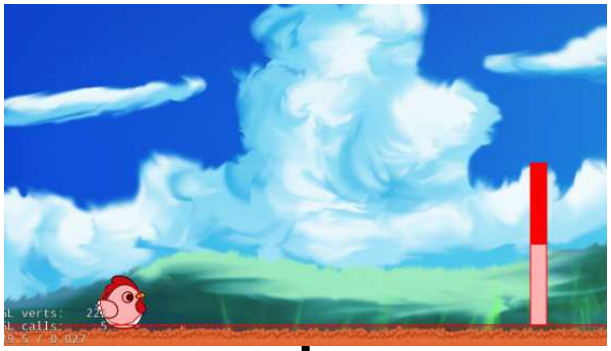
#pragma mark - XXXXXX





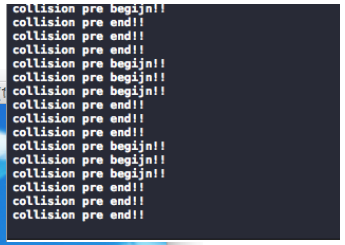
ソースコードに #pragma mark を書くとその内容が Function Menu に表示される

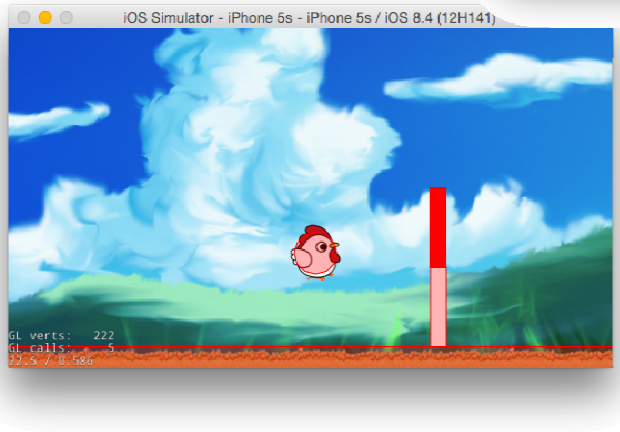
さて、実行してみよう！



対象物を置いて。。。

ひとまずコンソールに衝突応答としてメッセージを表示





飛ぶ！

次回は物理演算Chipmunk
引っ張り具合に上限をつける編

ここまでのソースプログラムはここ

http://monolizm.com/sab/src/AngryChicken12.zip

ご清聴ありがとうございました。