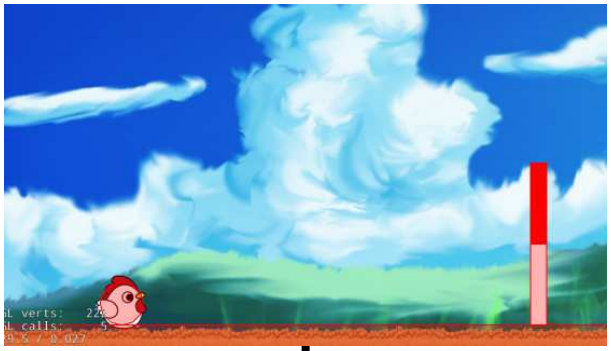
Cocos2d-xで作る物理演算ゲーム
引っ張り具合に上限をつける編

= 2016年1月23日 =

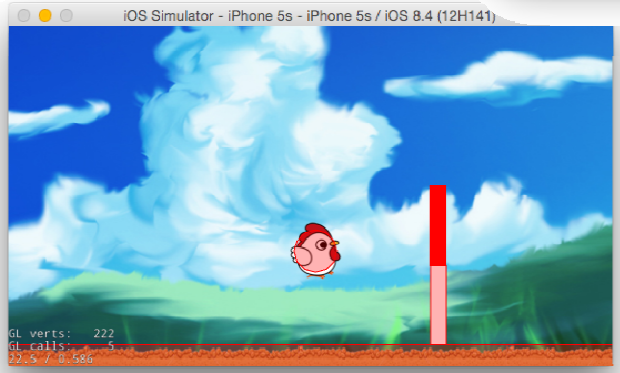
前回のあらすじ



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

```
collision pre begin!!
collision pre end!!
collision pre end!!
collision pre end!!
collision pre begin!!
collision pre begin!!
collision pre begin!!
collision pre end!!
collision pre end!!
collision pre end!!
collision pre begin!!
collision pre begin!!
collision pre begin!!
collision pre end!!
collision pre end!!
collision pre end!!
```



飛ぶ！

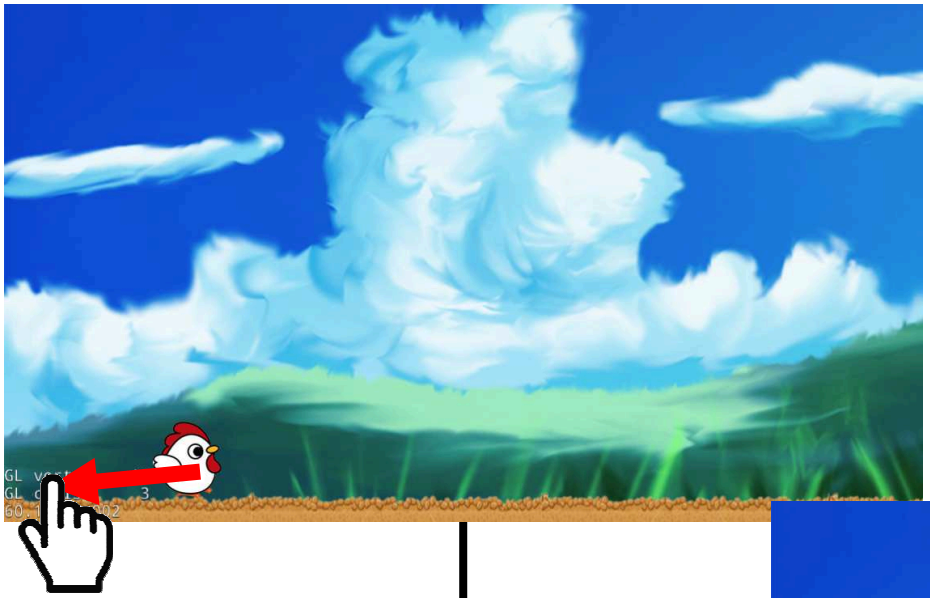
ここまでのソースプログラムはここ

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

ＧＥＴだぜ！

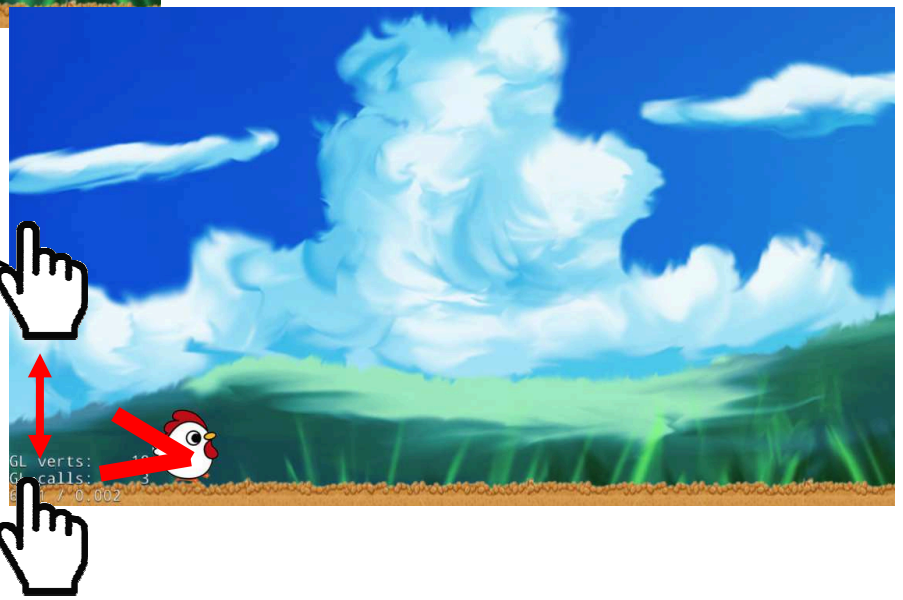
今回は引っ張り具合に
制限を付けます

今回はこちら



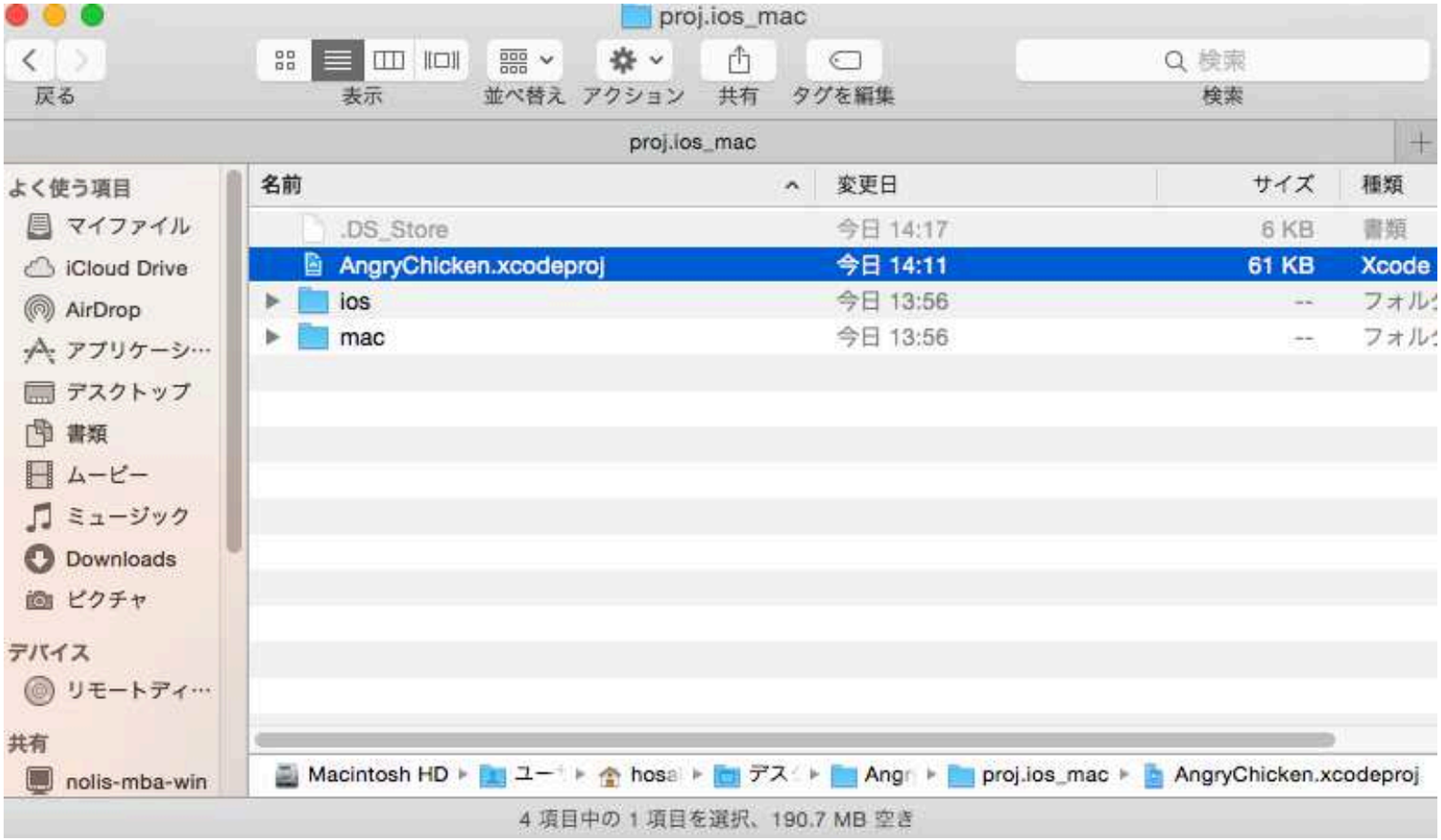
チキンにタッチして大きく引っ張っても…

それ以上は伸びない！



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引っ張っている線の距離を求める
②一定距離以上ならそれ以上伸ばさない

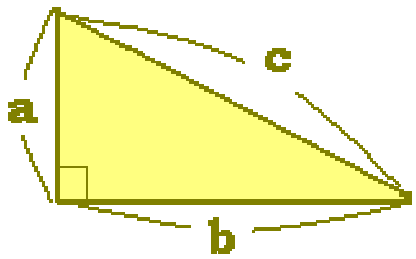
①引っ張っている線の距離を求める

つまり赤線の距離を求める・・・どうやったら求めることができるのか？
まずは冷静に情報を整理してみる。

鶏の座標とタッチしている座標、つまりその2点間の距離を求めるということ。
「2点間の距離を求める」にはどうすればよいか？

①引っ張っている線の距離を求める

「三平方の定理」を使うのだ！

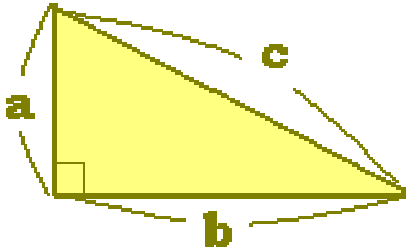


「三平方の定理」とは、ピタゴラスさんが作った、以下の方程式で直角三角形の斜辺の長さがわかる方程式。

$$a^2+b^2=c^2$$

つまりこんなイメージ。

①引っ張っている線の距離を求める



ってことでまずはa, bを求める必要がある。

これは簡単。

bを求めるには、タッチ座標のxから、鶏のxを引けば求まる。つまり
[b = タッチ座標x　－　鶏座標x]である。aもまったく同じ理屈で。

①引っ張っている線の距離を求める

とりあえずどこに書くかは置いておくと、プログラムではこうなる。

```
// キャラの位置を取得
Sprite* charSprite = (Sprite*)this->getChildByTag(CHAR_OBJTAG);
Vec2 charSpritePos = charSprite->getPosition();

// タッチ位置を取得
Vec2 touchLocation = touch->getLocation();

//長さを求める
float distance = touchLocation.getDistance(charSpritePos);
```

本当は前述のような計算を自分でする必要があるが、そこはライブラリがあるので、それを使ってしまおう。Vec2クラス（x,yの成分を持つクラス）にはgetDistanceというメソッドがある。自分の保持しているx,y座標と引数で渡されたVec2の距離を求めてくれる便利なメソッドなのだ。

あとはこの長さが一定距離だったら伸ばさないということをすればいい。

②一定距離以上ならそれ以上伸ばさない

画面をドラッグしている最中、線分描画で使用しているを変数「_currentTouchPoint」を一定距離以上なら更新しないという処理はどうだろうか？

```
void MainGame::_onTouchMoved(Touch* touch, Event* event)
{
    // キャラの位置を取得
    Sprite* charSprite = (Sprite*)this->getChildByTag(CHAR_OBJTAG);
    Vec2 charSpritePos = charSprite->getPosition();

    // タッチ位置を取得
    Vec2 touchLocation = touch->getLocation();

    //長さを求める
    float distance = touchLocation.getDistance(charSpritePos);
    if ( distance < 130 )
    {
        // 更新用に退避
        _currentTouchPoint = touch->getLocation();
        _currentTouchPoint = this->convertToNodeSpace(_currentTouchPoint);
    }
}
```

これならタッチ位置と鶏の位置の距離が130を超えたら、_currentTouchPointは、それ以上更新されず、130の距離を保つことができる！

実行してみよう！

うまくいった・・・？

確かに130以上になると線は伸びない。が、やってみるとすぐに問題点がわかる。130以上の状態に伸びないのはいいにしても、「角度は変えることができない」必要がある。

これは問題である。やっていて気持ち悪いし普通は角度は変えたいであろう。

何が問題なのか？

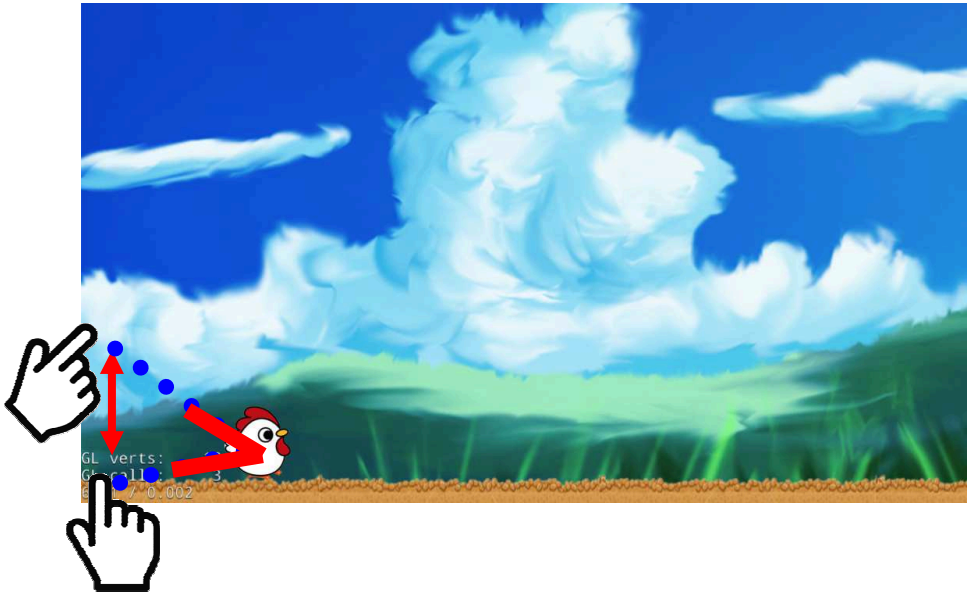
距離が１３０以上ではタッチ座標を更新しないので当たり前の挙動だ。なのでこのメソッドの内容は元に戻すとしよう。

どうすればいいか？

このような場合は少し根本ロジックを考えなおしてみよう。もっとシンプルに考えてみる。

単純に考えれば「表示する線分を一定の距離以上なら、１３０にする」というだけにだ。

つまりこーゆーことだ。本来は青線まで線分が伸びるが、その線分を表示する際に「130以上なら130の長さにする」（つまり赤線）ということだ。

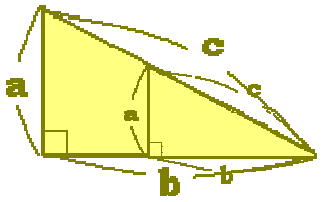


そうなるとまた課題が出てくる。線分を表示するには２点の座標が必要である。もちろん鶏の座標は解っているが、赤線の位置のｘ、ｙ座標は求まっていない。これをどうにかしなければいけない。

これさえ求めれば無事に赤線を表示することができる。

どうすればいいか？

まずはわかっている情報をまとめてみる。



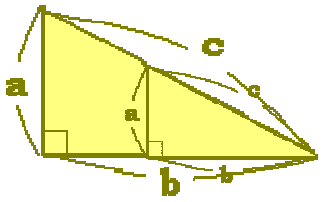
大きい方の三角形の3辺は求まっている・・・

1．小さい直角三角形の斜辺cは130である。
2．タッチ位置座標はわかっている
3．鶏の座標はわかっている。
4．タッチ位置座標と鶏の座標で形成される直角三角形
　　（大きい直角三角形）の3辺（a,b, c）はわかっている

ここから小さい三角形のa, bを求めることができないか？

できる！！

見ての通り２つの三角形は**相似**であるので、「対応する辺の長さの比は全て互いに等しい」



上記の図で言うと小さい三角形のc=130, 大きい三角形のc（ここではCとする）は求まっている。なので、比率を求めることができる。「h=130/C」

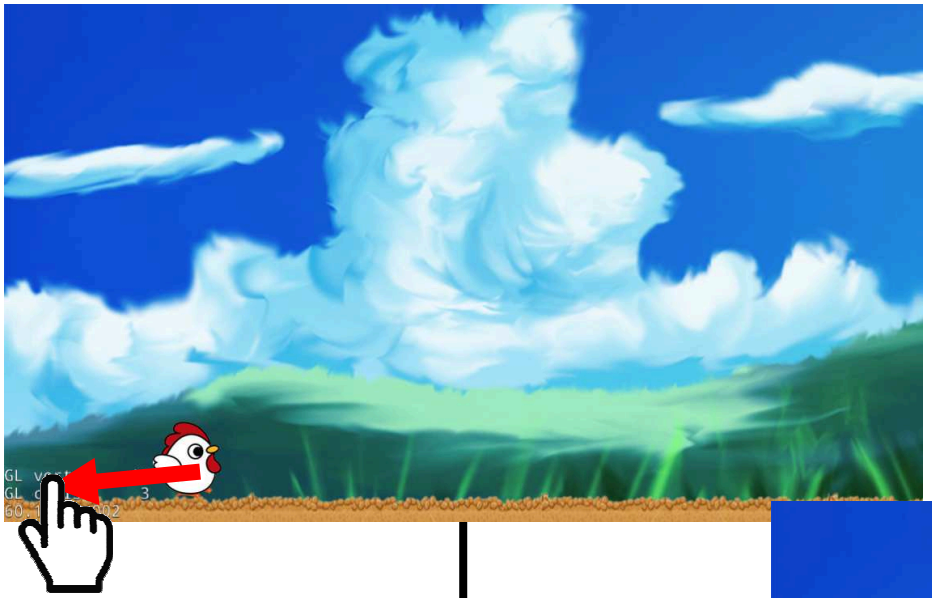
で前述の通り比率は全て互いに等しいので小さな三角形のbを求めるには、大きな三角形のb（ここではBとする）とすると次の計算式で求める。
「b=B*h」

となる。aも同様。

プログラムで書くとこうなる。

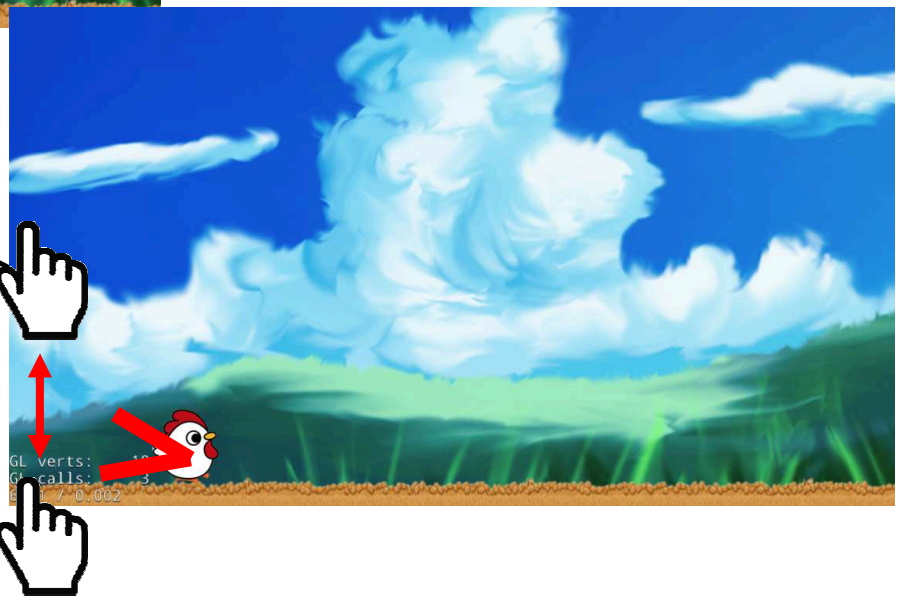
```
void MainGame::update(float delta)
{
    auto* draw = dynamic_cast<DrawNode*>(this->getChildByTag(PULL_LINE_OBJTAG));
    if ( draw != nullptr )
    {//存在した場合＝ひっぱり中なのでひっぱりラインを更新
        draw->clear();

        // 鶏の座標を取得
        Sprite* charSprite = (Sprite*)this->getChildByTag(CHAR_OBJTAG);
        Vec2 charSpritePos = charSprite->getPosition();

        // 辺a, bを求める
        auto tVec = _currentTouchPoint-charSpritePos;

        //長さを求める
        float distance = _currentTouchPoint.getDistance(charSpritePos);
        if ( distance > 130 )
        {
            // 斜辺の比率を求める。
            float ratio = 130/distance;

            // 辺a,bに比率をかけて小さい三角形のa,bを求める
            tVec* = ratio;

            // tVecは鶏の位置が原点の座標系となっているので鶏の位置に足しこむとx,yが求まる。
            _currentTouchPoint = charSpritePos+tVec;
        }
        draw->drawSegment(Vec2(charSpritePos.x, charSpritePos.y),
                                Vec2(_currentTouchPoint.x, _currentTouchPoint.y), 2.0f, Color4F(.5f, 1.0f, .7f, 1.0f));
    }
}
```

さて、実行してみよう！



チキンにタッチして大きく引っ張っても…

それ以上は伸びない！
さらに上下に指を移動させると長さをキープしたまま線分の角度が変わる！



次回は物理演算Chipmunk
やり直し処理を入れてみる編

ここまでのソースプログラムはここ

http://monolizm.com/sab/src/AngryChicken14.zip

ご清聴ありがとうございました。