cocos2dで かんたんな シューティングゲームを つくろう

STAGE 1
自己紹介

ZuQ9Nn(ずきゅ〜ん)

普段はPHPで業務システムや

Webサイトを構築

STAGE 2
今回の環境

- Xcode4.2
- iOS 4.3
- cocos2d 1.0.1
- インストールとかは説明しないのでググってください。

STAGE 3

今回のゲームの仕様

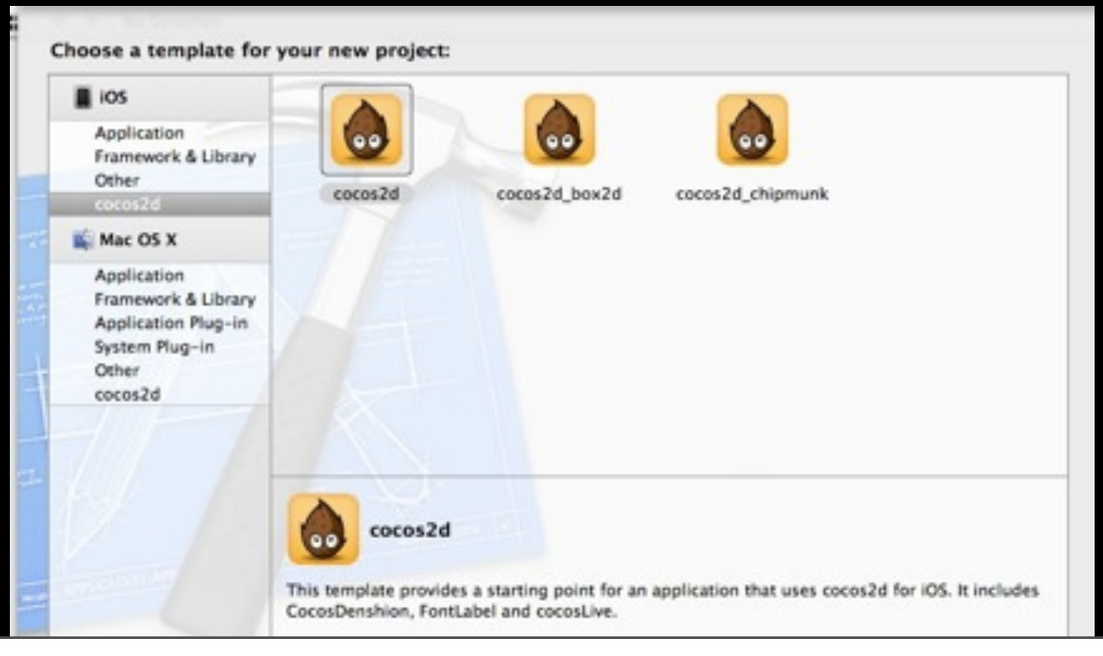
- 自機は画面をタップし左右にのみ動く
- 自機はタップで移動しながら攻撃
- 敵はランダムに出現
- 敵は攻撃しない
- 敵を10体倒せばGAME CLEAR
- GAME OVERは無い

STAGE 4
キャラクターの表示

- 画面にキャラクターを表示させてみる
- キャラクターの画像を用意
- 画像の形式はpngがいいらしい
- 今回は下記の戦闘機を表示する

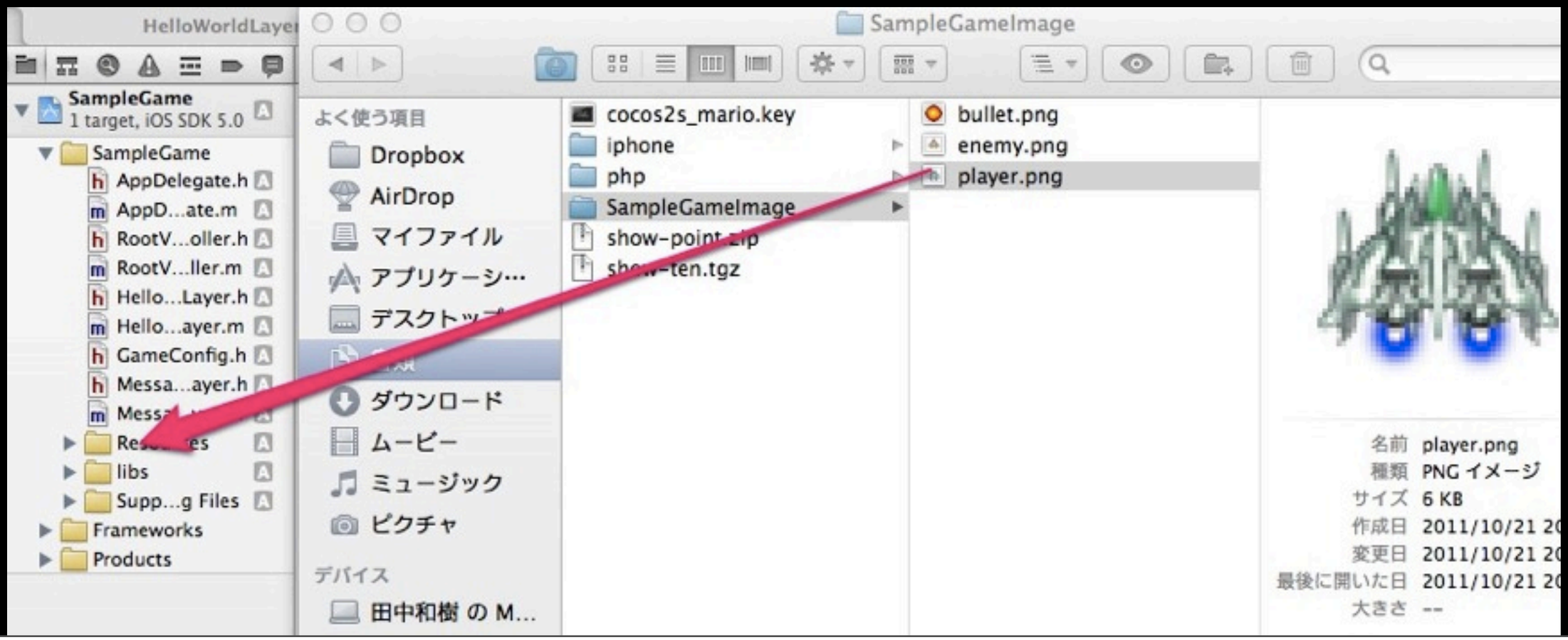
STAGE 5
プロジェクトの作成

- cocos2dをインストールするとテンプレートが選択できるようになっているので、coco2dのテンプレートを選択
- プロジェクト名をSampleとして保存



STAGE 6
プロジェクトの作成

- キャラクターの画像をドラッグアンドドロップでプロジェクトのResourcesフォルダにコピー



STAGE 7
ソースの記述

- 今回は自動で作成された
  HelloWorldLayer.mのinitメソッドを変更して、コードを記述していきます

STAGE 8
ソースの 中身

```
-(id) init {

    if((self=[super init])) {

        CCSprite *player = [CCSprite
                            spriteWithFile:@"player.png"];

        CGSize winSize = [CCDirector sharedDirector].winSize;
        player.position = ccp(winSize.width / 2,
                              player.contentSize.height);

        [self addChild:player];
    }
  return self;
}
```

STAGE 9
ソースの説明

- cocos2dは基本的にsingletonオブジェクトでautorelease

- 座標は左下が(0,0)

STAGE 10

Spriteとは

- Spriteは、基本的に絵を表示する(キャラクタ、障害物、アイテム etc..)
- Spriteの基準点は中心
- 単位はpoint

STAGE 11
キャラクターを動かす

- タッチイベントを有効にするにはinitメソッドで、self.isTouchEnabled = YES

- タッチイベントは今回ccTouchesEndedで処理
- タッチの位置はlocationInViewで取得できるが通常のiPhoneと座標がことなるためconvertToGLで変換する必要がある

STAGE 12
プログラムの注意点

- ccTouchesEndedの中で、spriteの座標を変えると残念な感じになる

- 今回はinitメソッドでscheduleUpdateをよびだす
- updateメソッドをオーバーライドする

STAGE 13

updateメソッドの中身

```
-(void) update:(ccTime)delta {

    CGSize winSize = [CCDirector sharedDirector].winSize;

    if(location.x < winSize.width / 2) {
        player.position = ccp(player.position.x - 2, player.position.y);
    }

    if(player.position.x <= player.contentSize.width) {
        player.position = ccp(player.contentSize.width,
                              player.position.y);
    }

    if(location.x > winSize.width / 2) {
        player.position = ccp(player.positon.x + 2, player.position.y);
    }

    if(player.position.x >= winSize.width - player.contentSize.width) {
        player.position = ccp(winSize.width - player.contentSize.width,
```

STAGE 14
攻撃してみよう

- 弾丸の画像をResourceにドラッグアンドドロップでコピー

- spriteを作成するまでは、キャラクター表示と同じだが、今回は自機をタップしたときに攻撃するのでinitメソッドではなくccTouchEndedメソッドで弾丸のspriteを作成する

STAGE 15
移動のアニメーション

- 弾丸は自機から画面の上へ移動させる
- 今回は弾丸のspriteを移動するのにCCMoveToを使用する
- spriteのrunActionメソッドで移動を実行
- しかし、CCMoveToのみすると残念な結果になるので画面の上に移動したらspriteを削除する処理を実行

STAGE 16

敵キャラの表示、移動

- 敵キャラの表示、移動は弾丸とほぼ同じ、敵はランダムで表示させたいので最初のposionとCCMoveTomのx座標をランダムにする
- initメソッドで敵を表示させると敵が1体、1度しか表示されない、次々と現れてほしいのでinitでscheduleメソッドを呼び出す
- 今回は、敵を表示させるaddTargetメソッドを作成、addTargetをscheduleメソッドで実行

STAGE 17

addTargetメソッドの中身

```
(void) addTarget:(ccTime)dt {

   CCSprite *target = [CCSprite spriteWithFile:@"target.png"];

   CGSize winSize = [CCDirector sharedDirector].winSize;
   target.position = ccp(target.contentSize.width +
                         (rand()%winSize.width), winSize.height
                         + target.contentSize.height);
   target.filpY = YES;

   [self addChild:target];

    id action = [CCMoveTo actionWithDuration:2
                   position :ccp(target.contentSize.width +
                         (rand()%winSize.width), 0);

    id actionDone = [CCCallFuncN actionWithTarget:
                    self selector @selector(spriteMoveFinished:)];

    [target runAction:[[CCSequence actions:action, actionDone, nil]];
```

STAGE 18

当たり判定

- 弾丸と敵があたったときに当たり判定を判断し、あったっていた場合は弾丸と敵を画面から消す処理を入れる
- 今回当たりの判断には矩形の重なりを判断するCGRectInsertSectsRectを用いる
- 弾丸を発射しているupdateメソッドのなかで処理を記述する

STAGE 19
当たり判定

- まずは、弾丸と敵を配列に追加
- updateメソッドであたり判定を判断してあたった弾丸と敵をlayerから削除

STAGE 20

updateメソッドの中身

```
NSMutableArray *bulletToDelete = [[NSMutableArray alloc  init];

   for (CCSprite *bullet in bulletList) {

       CGRect bulletRect = CGRectMake(
                           bullet.position x  bullet.position y,
                           bullet.contentSize width  bullet.contentSize height);

       NSMutableArray  targetToDelete = [[NSMutableArray alloc init];

       for (CCSprite *target in targetList) {

           CGRect targetRect = CGRectMake(
                               target.position x  target.position y
                               target.contentSize width, target.contentSize height);

           if (CGRectIntersectsRect bulletRect, targetRect)) {
               [targetToDelete addObject target];
           }


       for (CCSprite *target in targetToDelete) {
           [targetList removeObject target];
           [self removeChild:target cleanup YES];


       if (targetToDelete.count > 0) {
           [bulletToDelete addObject:bullet];
       }

       [targetToDelete release];


   for (CCSprite *bullet in bulletToDelete) {
        bulletList removeObject:bullet];
       [self removeChild:bullet cleanup YES];


   [bulletToDelete release];
```

STAGE 21
Sceneの追加

- GAME CLEARを表示するために
  Sceneを追加する
- Xcodeのプロジェクトを右クリック
  New Fileを選択、CCLayerのsubclass
  を選択、クラス名をMessageLayerと
  する

STAGE 22

SceneとLayer

- Sceneは画面
- SceneにLayerを複数、重ねて行く

STAGE 23
音楽をならそう

- SimpleAudioEngineをインポート
- SimpleAudioEngineは、本当にSimpleで音楽をならす、止めるくらいにしか出来ない。
- 音楽ファイルはwavが良いらしい
- 今回はinitメソッドでSimpleAudioEngineのplayBackgroundMusicを実行

STAGE 24
まとめ

- いわゆる、ツクール系のツールではないので、cocos2dだけあれば、簡単にゲームを作れる訳ではない
- プログラマは画像と音楽が無いとつらい
- ゲームのロジック、数学、物理の知識が別途必要
- OpenGL ESを直接触るよりかは、数段簡単

LAST STAGE
最後に

- 日本語の書籍や情報も徐々に多くなってきました。
- すぐに、パッと出来上がるのが好きな人じゃなくて、あくまでもプログラムを書くのが好きな人向け
- ご清聴ありがとうございました。

Congratulations