1. Game Scene Kevin Version
(Other codes are the same with first version)
Interface:

```
#import <Foundation/Foundation.h>
#import <AVFoundation/AVFoundation.h>
#import <AudioToolbox/AudioToolbox.h>
#import "cocos2d.h"

@interface GameScene : CCLayer {
    CCSprite *_character;
    CCSprite *_punchEffect;
    CCSprite *_background;
    CCSprite *_woopsie;
    CCAction *_hitAction;
    CCAction *_punchLeft;
    CCAction * _idle;
    CCAction *_centerJab;
    CCAction *_action;
    NSInteger score;
}
+(id) scene;
@property (nonatomic, retain) CCSprite * character;
@property (retain) CCSprite * woopsie;
@property (retain) CCSprite * background;
@property (retain) CCAction * punchLeft;
@property (retain) CCAction * idle;
@property (retain) CCAction * action;
@property (retain) CCAction * centerJab;
@property (nonatomic, retain) CCAction * hitAction;
@property (retain) CCSprite * punchEffect;
@property (retain) AVAudioPlayer * punchPlayer;
@property (retain) AVAudioPlayer * musicPlayer;
@property (retain) AVAudioPlayer * woopsPlayer;

@property NSInteger *score;


-(id) initWithCharacter: (int)mode ;

@end
```

Implementation:

```
#import "GameScene.h"

#import "MainMenuScene.h"
@implementation GameScene

//the character loading depends on the mode
//mode == 1 : loading picture related to piggy
```

```
//mode == 2 : loading picture related to frog
@synthesize character = _character;
@synthesize punchLeft = _punchLeft;
@synthesize woopsie = _woopsie;
@synthesize background = _background;
@synthesize hitAction = _hitAction;
@synthesize idle = _idle;
@synthesize centerJab = _centerJab;
@synthesize action = _action;
@synthesize score = _score;
@synthesize punchEffect = _punchEffect;
@synthesize punchPlayer = _punchPlayer;
@synthesize woopsPlayer = _woopsPlayer;
@synthesize musicPlayer = _musicPlayer;

double timePunched;

+(id) scene
{
	// 'scene' is an autorelease object.
	CCScene *scene = [CCScene node];

	// 'layer' is an autorelease object.
	GameScene *layer = [GameScene node];

	// add layer as a child to scene
	[scene addChild: layer];

	// return the scene
	return scene;
}
int numCombos = 0;
int score = 0;
double endTime = 40.0;
bool paused = false;

CGPoint previousTouch;

- (id) initWithCharacter:(int)mode {
    self = [super init];
    if (self != nil) {
        [[CCSpriteFrameCache sharedSpriteFrameCache] addSpriteFramesWithFile:@"mrspiggy.plist"];
	CCSpriteBatchNode * spriteSheet = [CCSpriteBatchNode batchNodeWithFile:@"mrspiggy.png"];

	self.background = [CCSprite spriteWithFile:@"gameBackground.png" rect:CGRectMake(0, 0, [CCDirector
sharedDirector].winSize.width, [CCDirector sharedDirector].winSize.height)];
	self.background.position =  ccp([CCDirector sharedDirector].winSize.width/2, [CCDirector
sharedDirector].winSize.height/2);


	[self addChild:_background];
	[self addChild:spriteSheet];
	NSMutableArray * hitAnimFrames = [NSMutableArray array];
```

```
	for(int i = 1; i  <= 5; i ++){
	   [hitAnimFrames addObject:
	    [[CCSpriteFrameCache sharedSpriteFrameCache] spriteFrameByName:[NSString
stringWithFormat:@"pigCenterPunch%d.png",i]]];

	}
	CCAnimation *hitanim = [CCAnimation animationWithFrames:hitAnimFrames delay: 0.1f];
	self.hitAction =[CCAnimate actionWithAnimation: hitanim restoreOriginalFrame:NO];

	NSMutableArray * centerJabAnims= [NSMutableArray array];
	for(int i = 1; i  <= 3; i ++){
	   [centerJabAnims addObject:
	    [[CCSpriteFrameCache sharedSpriteFrameCache] spriteFrameByName:[NSString
stringWithFormat:@"pigCenterJab%d.png",i]]];
	}

	CCAnimation * centerjabanim = [CCAnimation animationWithFrames:centerJabAnims delay: 0.1f];
	self.centerJab =  [CCAnimate actionWithAnimation:centerjabanim restoreOriginalFrame:NO];

	NSMutableArray * leftpunches= [NSMutableArray array];
	for(int i = 1; i  < 6; i ++){
	   [leftpunches addObject:
	    [[CCSpriteFrameCache sharedSpriteFrameCache] spriteFrameByName:[NSString
stringWithFormat:@"pigLeftJab%d.png",i]]];
	}

	CCAnimation * leftpunchanim = [CCAnimation animationWithFrames:leftpunches delay: 0.05f];
	self.punchLeft = [CCAnimate actionWithAnimation:leftpunchanim restoreOriginalFrame:NO];


	NSMutableArray * idles= [NSMutableArray array];
	for(int i = 1; i  < 5; i ++){
	   [idles addObject:
	    [[CCSpriteFrameCache sharedSpriteFrameCache] spriteFrameByName:[NSString
stringWithFormat:@"pigidle%d.png",i]]];

	}

	CCAnimation * idleanims = [CCAnimation animationWithFrames:idles delay: 0.175f];
	self.idle = [CCRepeatForever actionWithAction: [CCAnimate actionWithAnimation:idleanims
restoreOriginalFrame:NO]];
	self.action = self.idle;


	CGSize winSize = [CCDirector sharedDirector].winSize;
	self.character = [CCSprite spriteWithSpriteFrameName:@"pigidle1.png"];


	self.punchEffect = [CCSprite spriteWithFile:@"punchEffect1.png" rect:CGRectMake(0, 0, 60, 60)];
	self.punchEffect.position =  ccp(self.punchEffect.contentSize.width/2, winSize.height/2);
	_punchEffect.visible = !_punchEffect.visible;
```

```
        self.woopsie = [CCSprite spriteWithFile:@"woops.png" rect:CGRectMake(0, 0, 111, 182)];
        self.woopsie.position =  ccp(self.punchEffect.contentSize.width, winSize.height);
        [self.woopsie setPosition:CGPointMake([CCDirector sharedDirector].winSize.width/2,
self.woopsie.boundingBox.size.height)];
        _woopsie.visible = !_woopsie.visible;




        [self addChild:_woopsie];


        [self addChild:_punchEffect];

        _character.position = ccp(winSize.width/2, winSize.height/2);
        [spriteSheet addChild:_character];
        [_character runAction:_action];
        self.isTouchEnabled = YES;
        spriteSheet.tag = 152;


        timePunched = CACurrentMediaTime();
        [self LoadAVPlayers];
        [self loadGameUI];
        previousTouch = CGPointMake(0.0f,0.0f);
    }
    return self;
}

-(void) LoadAVPlayers
{

    NSString * path = [[[NSBundle mainBundle] resourcePath] stringByAppendingString:@"/slap.wav"];
    NSError * err;
    self.punchPlayer = [[AVAudioPlayer alloc] initWithContentsOfURL:[NSURL fileURLWithPath:path ]
error:&err];

    if(err){
        NSLog(@"%@ sound failed with reason: %@", path, [err localizedDescription]);
    }    else{
        self.punchPlayer.delegate = self;

    }

    [self.punchPlayer prepareToPlay];

    path = [[[NSBundle mainBundle] resourcePath] stringByAppendingString:@"/beaker4.wav"];

    self.woopsPlayer = [[AVAudioPlayer alloc] initWithContentsOfURL:[NSURL fileURLWithPath:path ]
error:&err];

    if(err){
```

```
        NSLog(@"%@ sound failed with reason: %@", path, [err localizedDescription]);
    }    else{
        self.woopsPlayer.delegate = self;

    }

    [self.woopsPlayer prepareToPlay];
    path = [[[NSBundle mainBundle] resourcePath] stringByAppendingString:@"/GameTheme.mp3"];

    self.musicPlayer = [[AVAudioPlayer alloc] initWithContentsOfURL:[NSURL fileURLWithPath:path ]
error:&err];

    if(err){
        NSLog(@"%@ sound failed with reason: %@", path, [err localizedDescription]);
    }    else{
        self.musicPlayer.delegate = self;

    }

    //    [self.musicPlayer prepareToPlay];
    self.musicPlayer.numberOfLoops = -1;
    [self.musicPlayer play];


}
CCLabelBMFont* scoreLabel;
CCLabelBMFont* comboLabel;
CCMenuItemSprite * pauseBtn;
CCLabelBMFont * endTimeLabel;

-(void) pauseGame{
    if(![CCDirector sharedDirector].isPaused) {
        [[CCDirector sharedDirector] pause];
        [self.musicPlayer pause];
        [pauseBtn setColor:ccBLUE];

    }
    else{
        [[CCDirector sharedDirector] resume];
        [self.musicPlayer play];
        [pauseBtn setColor:ccWHITE];
    }

    paused = !paused;
}

-(void) loadGameUI{
    scoreLabel = [CCLabelBMFont labelWithString:@"0" fntFile:@"hud_font.fnt"];
    CCLabelBMFont * scoreTHING = [CCLabelBMFont labelWithString:@"SCORE :"
fntFile:@"hud_font.fnt"];
    CCLabelBMFont * comboThing = [CCLabelBMFont labelWithString:@"COMBO :"
fntFile:@"hud_font.fnt"];
    comboLabel = [CCLabelBMFont labelWithString:[NSString stringWithFormat:@"%d",numCombos]
```

```
fntFile:@"hud_font.fnt"];


    endTimeLabel = [CCLabelBMFont labelWithString:[NSString stringWithFormat:@"TIME LEFT :%d",40]
fntFile:@"hud_font.fnt"];

    [endTimeLabel setScale:1.5f];
    [comboThing setScale:2.0f];
    [comboLabel setScale:2.0f];
    [scoreTHING setScale:2.0f];
    [scoreLabel setScale:2.0f];
    [endTimeLabel setColor:ccWHITE];
    [scoreLabel setColor:ccWHITE];
    [scoreTHING setColor:ccWHITE];
    [comboLabel setColor:ccWHITE];
    [comboThing setColor:ccWHITE];

    pauseBtn = [CCMenuItemSprite itemFromNormalSprite:[CCSprite spriteWithFile:@"pauseBtn.png"
rect:CGRectMake(0,0, 60, 60)] selectedSprite:[CCSprite spriteWithFile:@"pauseBtn.png"
rect:CGRectMake(0,0, 60, 60) ]  target:self selector:@selector(pauseGame)];


    CCMenuItemLabel * scoreThingLabel = [CCMenuItemLabel itemWithLabel:scoreLabel];
    CCMenuItemLabel * SCORETHINGLABEL1 = [CCMenuItemLabel itemWithLabel:scoreTHING];
    CCMenuItemLabel * comboThingLabel = [CCMenuItemLabel itemWithLabel:comboThing];
    CCMenuItemLabel * comboThingLabel1 = [CCMenuItemLabel itemWithLabel:comboLabel];

    CCMenu* menu = [CCMenu menuWithItems:scoreThingLabel, SCORETHINGLABEL1, comboThingLabel,
comboThingLabel1, pauseBtn, [CCMenuItemLabel itemWithLabel:endTimeLabel], nil];
    [self addChild:menu];
    [menu setPosition:ccp([CCDirector sharedDirector].winSize.width/2, 10.0f)];
    [SCORETHINGLABEL1 setPosition:ccp(-[CCDirector sharedDirector].winSize.width/2 +50.0f, 40.0f)];
    [scoreThingLabel setPosition:ccp(50.0f, 40.0f)];
    [comboThingLabel setPosition:ccp(-[CCDirector sharedDirector].winSize.width/2 +50.0f, 70.0f)];
    [comboThingLabel1 setPosition:ccp(50.0f,70.0f)];
    [pauseBtn setPosition:ccp(0.0f, 20.0f)];
    [endTimeLabel setPosition: ccp(0.0f, 100.0f)];
}

-(void) registerWithTouchDispatcher
{
        [[CCTouchDispatcher sharedDispatcher] addTargetedDelegate:self priority:0 swallowsTouches:YES];
}


bool flippedX = false;
CCAction * currentAction;
-(void) handlePunch: (CGPoint) pos{
    if(!paused){
        if(numCombos > 0){
            [_character stopAction:currentAction];
        }
        if(self.punchPlayer.playing){
```

```
            self.punchPlayer.currentTime = 0;
        }
        [self.punchPlayer prepareToPlay];
        [self.punchPlayer play];
        float spriteCenter = _character.boundingBox.origin.x + _character.boundingBox.size.width/2;
        if(pos.x >  spriteCenter + _character.boundingBox.size.width/7 && pos.x <  spriteCenter +
_character.boundingBox.size.width*.90){
            if(!flippedX)    {
                flippedX = _character.flipX = true;
            }
            currentAction = _punchLeft;
            [_character runAction:_punchLeft];
            score += 10;
        }
        else if(pos.x < spriteCenter - _character.boundingBox.size.width/7 && pos.x > spriteCenter -
_character.boundingBox.size.width*.90){
            if(flippedX)    {
                flippedX = _character.flipX = false;
            }
            currentAction = _punchLeft;
            [_character runAction:_punchLeft];
            score += 10;

        }
        else if(pos.x > spriteCenter - _character.boundingBox.size.width/7 && pos.x < spriteCenter +
_character.boundingBox.size.width/7 )    {
            currentAction = _centerJab;
            [_character runAction:_centerJab];
            score += 10;

        }
        _punchEffect.position = pos;
        _punchEffect.visible = !_punchEffect.visible;
        [NSTimer scheduledTimerWithTimeInterval:0.3f target:self selector:@selector(handlePunchEffect)
userInfo:nil repeats:NO];
        [self schedule:@selector(updateCombo) interval:0];//using zero for the interval will just sync us with the
default refresh, usually every 1/60 second

        scoreLabel.string = [NSString stringWithFormat:@"%i",score ];
    }
}

-(void) handlePunchEffect{
    _punchEffect.visible = !_punchEffect.visible;
}

-(void) hideBeeker{
    [self.woopsie setVisible:FALSE];
}

bool touchBegan = false;
-(BOOL) ccTouchBegan: (UITouch *) touch withEvent:(UIEvent *) event {
    if(!touchBegan) {
```

```
        touchBegan = true;
    }


    return YES;
}
-(void) ccTouchesMoved:(NSSet *)touches withEvent:(UIEvent *)event{

}
double deltaTime;
-(void)updateCombo{
    if(!paused){
        if(!timePunched){
            timePunched = CACurrentMediaTime();
        }
        double deltaTouchTime = CACurrentMediaTime() - timePunched;
        if(deltaTouchTime >= 0.5f){
            numCombos = 0;
            comboLabel.string = [NSString stringWithFormat:@"%i",numCombos ];
        }
        double currentTime = CACurrentMediaTime() - deltaTime;
        endTime -= deltaTouchTime/60;
        int tempEnd = (int)endTime;
        endTimeLabel.string = [NSString stringWithFormat:@"TIME LEFT :%d", tempEnd];
        deltaTime = currentTime;
        if(tempEnd == 0){
            MainMenuScene * ms = [MainMenuScene node];
            [self.musicPlayer stop];
            [[CCDirector sharedDirector] pushScene:ms];

            endTime = 40;
            score = 0;
        }
    }

}

-(void) ccTouchEnded:(UITouch *)touch withEvent:(UIEvent *)event {
    if(!paused){
        double deltaTouchTime = CACurrentMediaTime() - timePunched;
        timePunched = CACurrentMediaTime();


        touchBegan = false;
        CGPoint touchPos = [touch locationInView: [touch view]];
        touchPos = [[CCDirector sharedDirector] convertToGL: touchPos];
        touchPos = [self convertToNodeSpace:touchPos];
        if(!CGRectContainsPoint(_punchEffect.boundingBox, touchPos)){
            if(CGRectContainsPoint(_character.boundingBox, touchPos)) {
                if(deltaTouchTime < 0.3f)
                {
                    ++numCombos;
                    if(numCombos%10 == 0){
```

```
                score += numCombos*5;
            }
            if(numCombos% 30 == 0){
                [self.woopsie setVisible:TRUE];
                [NSTimer scheduledTimerWithTimeInterval:1.0f target:self selector:@selector(hideBeeker)
userInfo:nil repeats:NO];
                [self.woopsPlayer play];
                endTime += 2.0;
            }
            comboLabel.string = [NSString stringWithFormat:@"%i",numCombos ];
        }

        [self handlePunch:touchPos];
    }
  }
}
```

```
-(void)dealloc
{
    self.character = nil;
    [self.punchEffect release];
    [self.action release];
    [self.punchLeft release];
    [self.centerJab release];
    [self.hitAction release];
    [self.action release];
    [self.punchPlayer release];
    [self.musicPlayer release];
    [self.woopsPlayer release];
    [self.idle release];
    self.score = nil;
        [super dealloc];
}

@end
```