



Building an iPhone Tile-based Game using Cocos2D

We will use the Cocos2D framework to create a 2D game on iPhone. It's a light weight open source 2D engine that has sprite support, graphical effects, animations, physics, sound engines and a lot more.

Do we all remember games like Legend of Zelda?



Or final fantasy?





We are going to learn how to create such tiled-based game.

1. Download Cocos2D.

http://www.cocos2d-iphone.org/wiki/doku.php/release_notes:0_99_5#download

This is the latest cocos2D on ios4 support, (xcode 3.2.4 onwards) previous versions are for previous version of xcode.

2. After downloading, install it as a plug in to xcode by using the following.

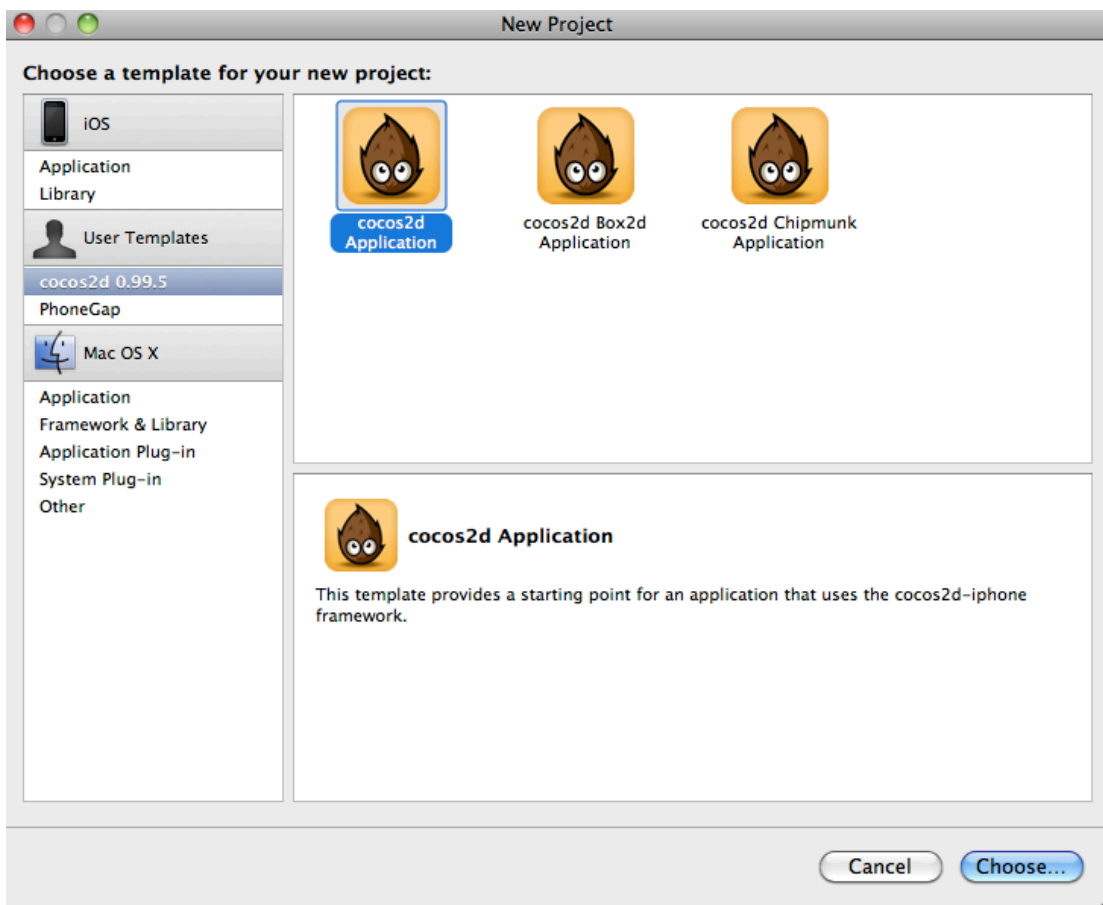
1. open Terminal
2. in the terminal enter "cd " (with a trailing space and without quotes)
3. drag your unarchived cocos2D folder onto Terminal window - it should read something like
idmc2s-imac:~ idmc\$ cd /Users/idmc/Desktop/cocos2D/cocos2d-iphone-0.99.4
4. Hit enter
5. sudo sh install_templates.sh -f
6. Hit enter

It will start installing the cocos2D templates and give it a while before it finally prompted "done".

In this workshop, the cocos2d plugin has already been installed on your mac, unfortunately I cannot package this as a distributable.

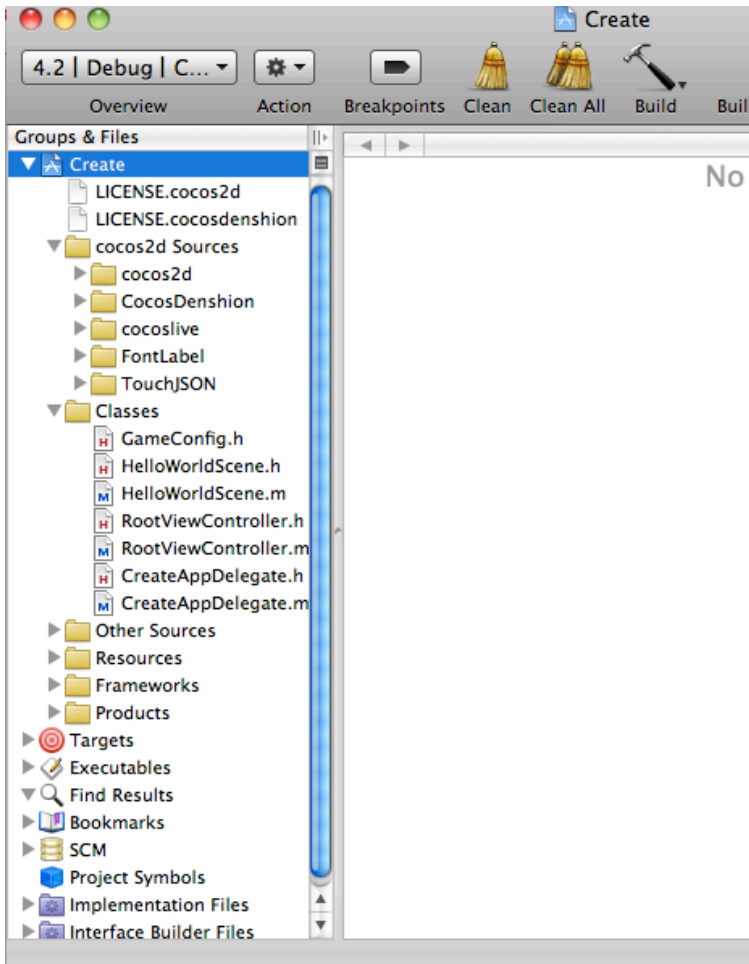


3. Open up xcode create a new project. Choose a cocos2d application.



The following class structure is created for you in the xcode. The essential cocos2D libraries are in the top left column as default. By default it will create a HelloWorldScene. Cocos2D is organized into the concept of “scenes”, like levels of a game. You can have a game menu scene, main action scene, battle scene, game over scene or game complete scene. Inside a scene you can have a number of layers, layers can contain nodes such as sprites labels, menus and more.





4. Build and Go. You should see a HelloWorld message like the following.

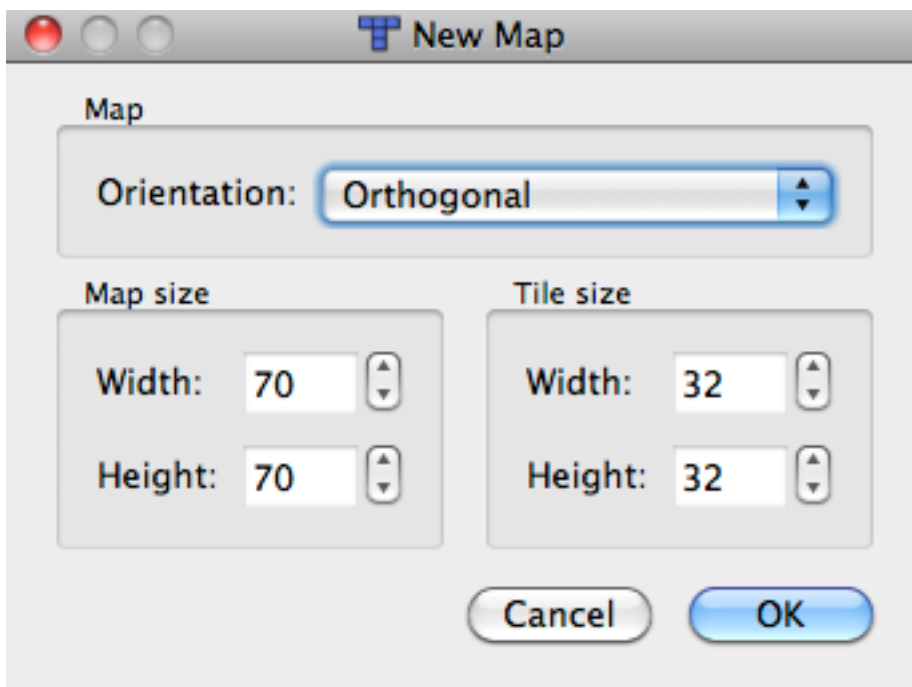


5. Now we are going to learn how to make a map using a tilemap. There is an open source Tiled Map Editor called Tiled Qt. Lets download this and start to make a map with Tiled.

<http://www.mapeditor.org/>



6. Open up Tiled, lets create a tiled map of dimension 70 x 70 tiles, width and height of a tile is 32 x 32.



7. For this exercise, we need to import a tile set called “free_tileset_version_10.png” which already created in resource folder. Basically a tile set is an image file, which serves as a template to create a tiled map. You can design your own. The size of every tile is designed to be 32 x 32 pixels exactly. We also import a tile set called “meta_tiles.png”. This will be used to create special tiles, which depicts a special meaning on the map later on.

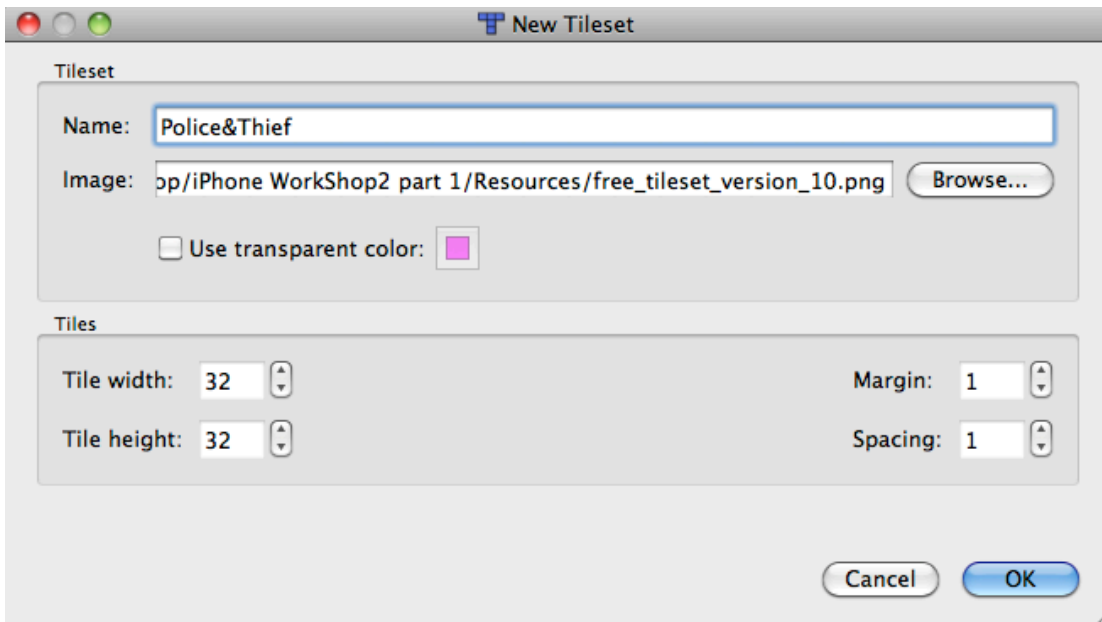




See more at silvianeto.net/tag/tileset
 Free for use under
 Creative Commons Attribution Share-Alike License.



8. Click on “Map” in menu bar, “New Tileset”, use the following.



9. Feel free to draw your own map. Use the tools to place, fill, eraser.



10. Zoom in and Zoom out the map using shortcuts.

View	Map	Layer
Show Grid		⌘G
Zoom In		⌘+
Zoom Out		⌘-
Normal Size		⌘0

11. Save the layer, which you drew your map on, as "Background". After creating your map, save as a TileMap.tmx file. I have created mine called Police&Thief.tmx.





12. Add the tmx file as well as the “free_tileset_version_10.png “ and “meta_tiles.png “ to Resources in xcode. Add the following code to HelloWorldScene. Make sure the name of the map correspond to the name you give to your tmx.

```
// Inside the HelloWorld class declaration
CCTMXTiledMap *_tileMap;
CCTMXLayer *_background;

// After the class declaration
@property (nonatomic, retain) CCTMXTiledMap *tileMap;
@property (nonatomic, retain) CCTMXLayer *background;
```

```
// Right after the implementation section
@synthesize tileMap = _tileMap;
@synthesize background = _background;

// In dealloc
.tileMap = nil;
.background = nil;
```



```

// Replace the init method with the following
-(id) init
{
    if( (self=[super init] ) ) {

        self.tileMap = [CCTMXTiledMap tiledMapWithTMXFile:@"TileMap.tmx"];
        self.background = [_tileMap layerNamed:@"Background"];

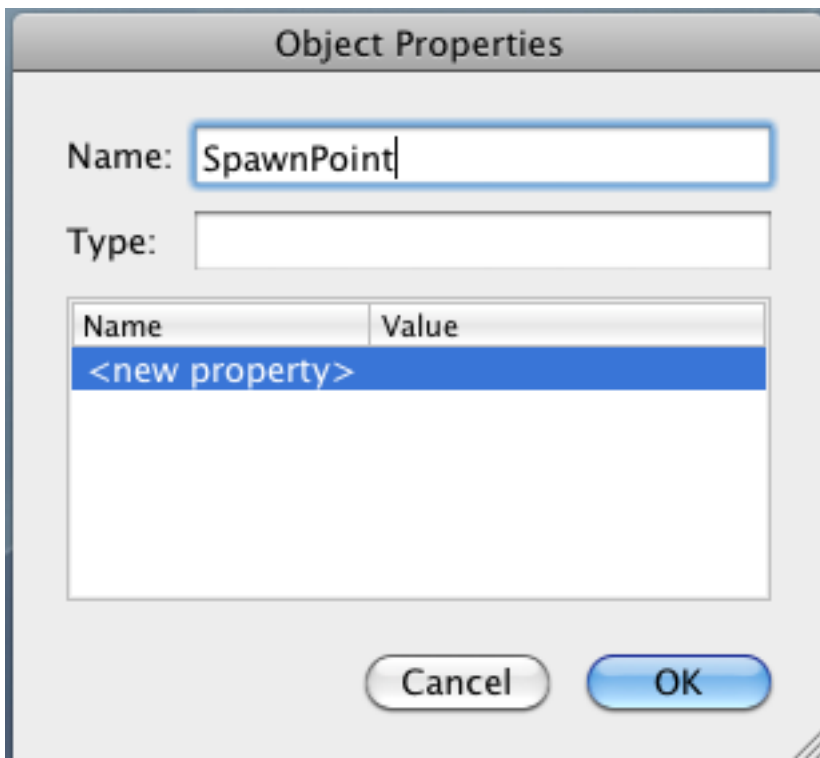
        [self addChild:_tileMap z:-1];

    }
    return self;
}

```

12. You have loaded the map into the HelloWorldScene. Build and Go to see what is the result. Soon we will create another layer called the object layer to spawn some player and enemy sprites onto the map, as well as a meta layer to specify where the player can move (or will collide) in the map, and add an items layer for items which the player can collect.

13. From now on, we will now work with the tmx in the xcode resource. First, to add the player layer, open up the tmx in Tiled. Go to the menu bar and click "Layer\Add Object Layer". Name the layer as "Objects". Click somewhere on the map and right click properties and give it a name of "SpawnPoint". This will be the spawn point of the player.



... re the tmx, and lets add a player image to the xcode. Drag and drop the Player1.png to the xcode. e following code to HelloWorldScene.



```
// Inside the HelloWorld class declaration
CCSprite *_player;

// After the class declaration
@property (nonatomic, retain) CCSprite *player;
```

```
// Right after the implementation section
@synthesize player = _player;

// In dealloc
self.player = nil;

// Inside the init method, after setting self.background
CCTMXObjectGroup *objects = [_tileMap objectGroupNamed:@"Objects"];
NSAssert(objects != nil, @"'Objects' object group not found");
NSMutableDictionary *spawnPoint = [objects objectForKey:@"SpawnPoint"];
NSAssert(spawnPoint != nil, @"SpawnPoint object not found");
int x = [[spawnPoint valueForKey:@"x"] intValue];
int y = [[spawnPoint valueForKey:@"y"] intValue];

self.player = [CCSprite spriteWithFile:@"Player.png"];
_player.position = ccp(x, y);
[self addChild:_player];

[self setViewpointCenter:_player.position];
```

```
-(void)setViewpointCenter:(CGPoint) position {

    CGSize winSize = [[CCDirector sharedDirector] winSize];

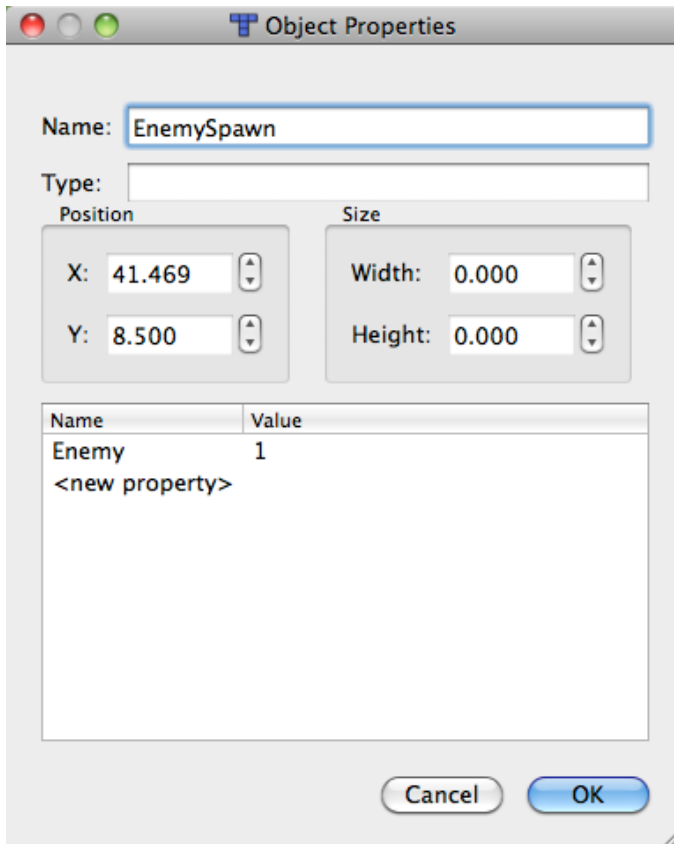
    int x = MAX(position.x, winSize.width / 2);
    int y = MAX(position.y, winSize.height / 2);
    x = MIN(x, (_tileMap.mapSize.width * _tileMap.tileSize.width)
        - winSize.width / 2);
    y = MIN(y, (_tileMap.mapSize.height * _tileMap.tileSize.height)
        - winSize.height/2);
    CGPoint actualPosition = ccp(x, y);

    CGPoint centerOfView = ccp(winSize.width/2, winSize.height/2);
    CGPoint viewPoint = ccpSub(centerOfView, actualPosition);
    self.position = viewPoint;
}
```

What we have done is to retrieve the object layer and create a sprite object representing the player, onto the screen. We set the view to center at the player.



15. Lets add also a enemy objects. Open the tmx in Tiled, select objects layer, click somewhere on the map not too far off to the player and right click properties and give it a name of "EnemySpawn". Set a new property name called Enemy and value 1. This will be the spawn point of an enemy.



16. Drag the "enemy1.png" file to xcode and setup the following codes.

Declare an NSMutableArray to store references to enemies object.

```
NSMutableArray *_enemies;
```

In the init method after creating the player.

```
NSMutableDictionary * spawnPoint_e;
_enemies = [[NSMutableArray alloc] init];
for (spawnPoint_e in [objects objects]) {
    if ([[spawnPoint_e valueForKey:@"Enemy"] intValue] == 1){
        x = [[spawnPoint_e valueForKey:@"x"] intValue];
        y = [[spawnPoint_e valueForKey:@"y"] intValue];
        [self addEnemyAtX:x y:y];
    }
}
```

Create a method to add enemy sprites to the screen as well as add the object reference to the _enemies array.

```
-(void)addEnemyAtX:(int)x y:(int)y {
    CCSprite *enemy = [CCSprite spriteWithFile:@"enemy1.png"];
    enemy.position = ccp(x, y);
```

```

    [self addChild:enemy];
    [_enemies addObject:enemy];
}

```

17. Lets stops here and Build and Go. At this point refer to source code PoliceAndThief.zip



18. Lets try to move the player around the map. Before that, we create another layer in the HelloWorldScene. Open HelloWorldScene.h above the @interface HelloWorld, add.

```

@class HelloWorld;
@interface GameHud : CCLayer
{
    HelloWorld *_gameLayer;
}

@property (nonatomic, retain) HelloWorld *gameLayer;

@end

```

19. Open HelloWorldScene.m above the @implementation HelloWorld, add.

```

@implementation GameHud
@synthesize gameLayer = _gameLayer;

CCMenuItem *righton;
CCMenuItem *downon;
CCMenuItem *lefton;
CCMenuItem *upon;
CCMenuItemToggle *toggleItemright;
CCMenuItemToggle *toggleItemdown;
CCMenuItemToggle *toggleItemleft;
nuItemToggle *toggleItemup;

```



```

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

        righton = [[CCMenuItemImage itemFromNormalImage:@"arrowright-pressed.png"
selectedImage:@"arrowright-pressed.png" target:nil selector:nil] retain];
        toggleItemright = [CCMenuItemToggle itemWithTarget:self
selector:@selector(moveButtonTapped:) items:righton, nil];
        CCMenu *toggleMenuright = [CCMenu menuWithItems:toggleItemright, nil];
        toggleMenuright.position = ccp(460, 160);
        [self addChild:toggleMenuright];

        downon = [[CCMenuItemImage itemFromNormalImage:@"arrowdown-pressed.png"
selectedImage:@"arrowdown-pressed.png" target:nil selector:nil] retain];
        toggleItemdown = [CCMenuItemToggle itemWithTarget:self
selector:@selector(moveButtonTapped:) items:downon, nil];
        CCMenu *toggleMenudown = [CCMenu menuWithItems:toggleItemdown, nil];
        toggleMenudown.position = ccp(240, 20);
        [self addChild:toggleMenudown];

        lefton = [[CCMenuItemImage itemFromNormalImage:@"arrowleft-pressed.png"
selectedImage:@"arrowleft-pressed.png" target:nil selector:nil] retain];
        toggleItemleft = [CCMenuItemToggle itemWithTarget:self
selector:@selector(moveButtonTapped:) items:lefton, nil];
        CCMenu *toggleMenuleft = [CCMenu menuWithItems:toggleItemleft, nil];
        toggleMenuleft.position = ccp(20, 160);
        [self addChild:toggleMenuleft];

        upon = [[CCMenuItemImage itemFromNormalImage:@"arrowup-pressed.png"
selectedImage:@"arrowup-pressed.png" target:nil selector:nil] retain];
        toggleItemup = [CCMenuItemToggle itemWithTarget:self
selector:@selector(moveButtonTapped:) items:upon, nil];
        CCMenu *toggleMenuup = [CCMenu menuWithItems:toggleItemup, nil];
        toggleMenuup.position = ccp(240, 300);
        [self addChild:toggleMenuup];

    }
    return self;
}

- (void)moveButtonTapped:(id)sender
{
    CCMenuItemToggle *toggleItem = (CCMenuItemToggle *)sender;
    if (toggleItem.selectedItem == righton) {
        [_gameLayer moveright];
    }
    if (toggleItem.selectedItem == downon) {
        [_gameLayer movedown];
    }
    if (toggleItem.selectedItem == lefton) {
        [_gameLayer moveleft];
    }
    if (toggleItem.selectedItem == upon) {
        [_gameLayer moveup];
    }
}

@end

```

20. Declare in `@interface HelloWorld`, add property and synthesize, release at the end.

```

GameHud *_hud;
- property (nonatomic, retain) GameHud *hud;

```

```
@synthesize hud = _hud;

self.hud = nil;
```

21. In @implementation HelloWorld, +(id) scene method add,

```
GameHud *hud = [GameHud node];
[scene addChild: hud];
layer.hud = hud;
hud.gameLayer = layer;
```

22. Add the following methods to receive the button actions and execute the moving action.

```
-(void)setPlayerPosition:(CGPoint)position {
    //_player.position = position;
    CCAction *action = [CCMoveTo actionWithDuration:0.25 position: position];
    [_player runAction:action];
}

- (void)moveright
{
    CGPoint playerPos = _player.position;
    playerPos.x += _tileMap.tileSize.width;

    if (playerPos.x <= (_tileMap.mapSize.width * _tileMap.tileSize.width) &&
        playerPos.y <= (_tileMap.mapSize.height * _tileMap.tileSize.height) &&
        playerPos.y >= 0 &&
        playerPos.x >= 0 )
    {
        [self setPlayerPosition:playerPos];
    }

    [self setViewpointCenter:_player.position];
}

- (void)movedown
{
    CGPoint playerPos = _player.position;
    playerPos.y -= _tileMap.tileSize.height;

    if (playerPos.x <= (_tileMap.mapSize.width * _tileMap.tileSize.width) &&
        playerPos.y <= (_tileMap.mapSize.height * _tileMap.tileSize.height) &&
        playerPos.y >= 0 &&
        playerPos.x >= 0 )
    {
        [self setPlayerPosition:playerPos];
    }

    [self setViewpointCenter:_player.position];
}

- (void)moveleft
{
    CGPoint playerPos = _player.position;
    playerPos.x -= _tileMap.tileSize.width;

    if (playerPos.x <= (_tileMap.mapSize.width * _tileMap.tileSize.width) &&
        playerPos.y <= (_tileMap.mapSize.height * _tileMap.tileSize.height) &&
        playerPos.y >= 0 &&
        playerPos.x >= 0 )
    {
```

```

    [self setPlayerPosition:playerPos];
}

[self setViewpointCenter:_player.position];
}

- (void)moveup
{
    CGPoint playerPos = _player.position;
    playerPos.y += _tileMap.tileSize.height;

    if (playerPos.x <= (_tileMap.mapSize.width * _tileMap.tileSize.width) &&
        playerPos.y <= (_tileMap.mapSize.height * _tileMap.tileSize.height) &&
        playerPos.y >= 0 &&
        playerPos.x >= 0 )
    {
        [self setPlayerPosition:playerPos];
    }

    [self setViewpointCenter:_player.position];
}
}

```

23. Build and Go. Try moving around the map. Refer to source code PoliceAndThief-2.zip. Note we still have a problem here. The Player can move anywhere on the map with no restriction.

