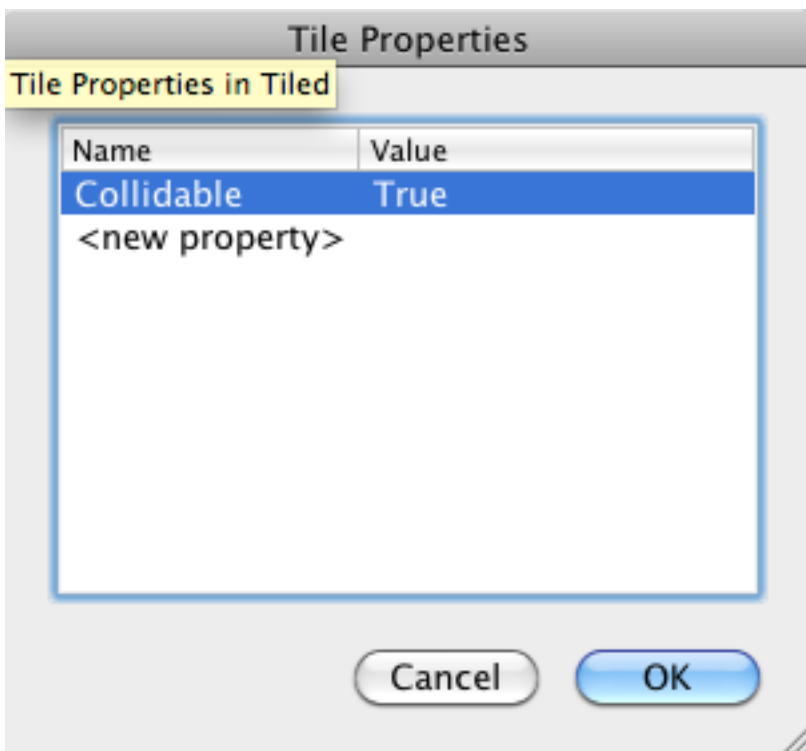
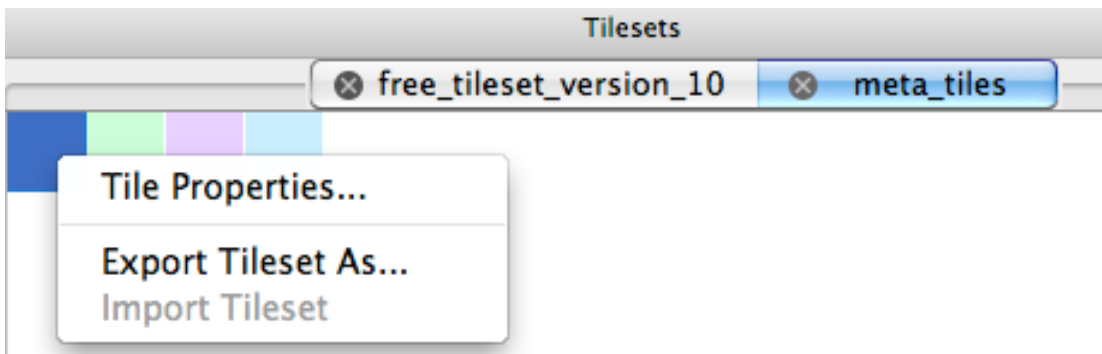




Building an iPhone Tile-based Game using Cocos2D 2

We will continue the previous app by extending with collision walls, add collectible items that player can pick up. Add sound effects.

1. Lets add a collision wall around the map to act as boundaries to where the player can move. For this we need to add another layer using the meta tile set. Open up the tmx in Tiled, create a new layer and name it "Meta". Make sure the meta_tiles tile set is imported. Let's select the red tile and denotes it as a collision tile where player cannot pass through. We have other meaning for other colored tiles. Right click properties on the red tile, set a property called Collidable and value set to True.



2. With the meta layer selected, draw around the map to make those spaces you do not want the player to move through. Save the map and add these codes to HelloWorldScene.

```
// Inside the HelloWorld class declaration
CCTMXLayer *_meta;

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

```
// Right after the implementation section
@synthesize meta = _meta;

// In dealloc
self.meta = nil;

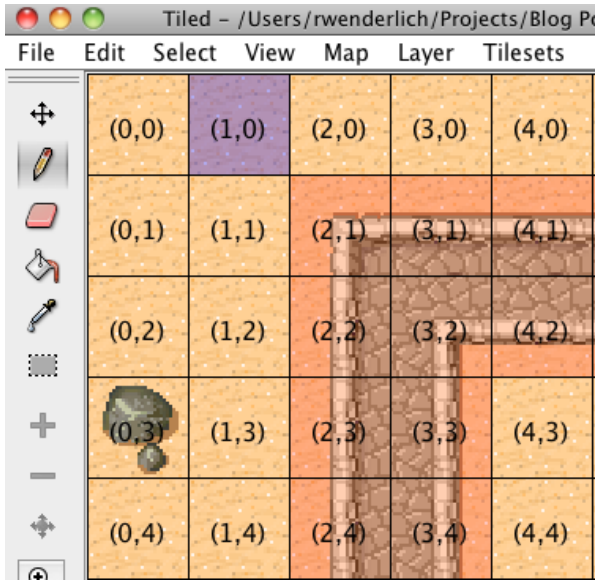
// In init, right after loading background
self.meta = [_tileMap layerNamed:@"Meta"];
_meta.visible = NO;

// Add new method
- (CGPoint)tileCoordForPosition:(CGPoint)position {
    int x = position.x / _tileMap.tileSize.width;
    int y = ((_tileMap.mapSize.height * _tileMap.tileSize.height) - position.y) /
    _tileMap.tileSize.height;
    return ccp(x, y);
}

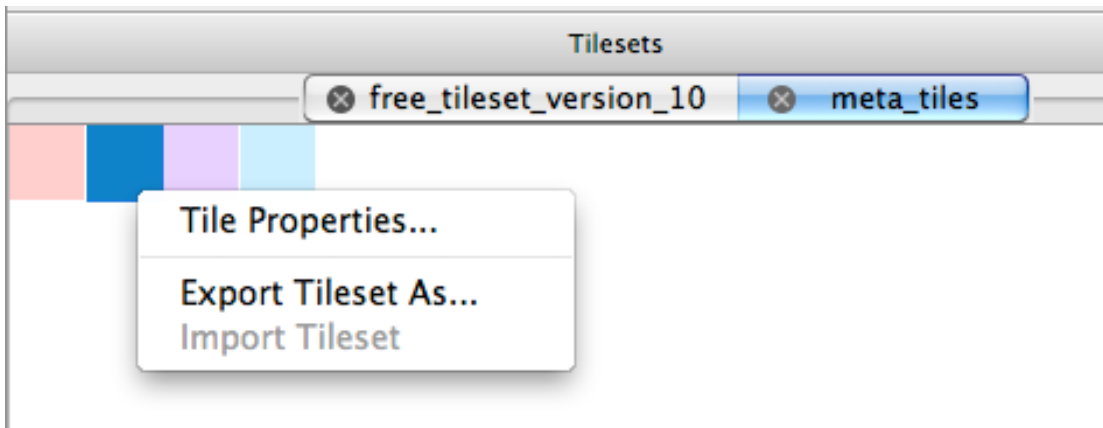
- (CGPoint)positionCoordForTile:(CGPoint)position {
    int x = position.x * _tileMap.tileSize.width;
    int y = -(position.y * _tileMap.tileSize.height - (_tileMap.mapSize.height *
    _tileMap.tileSize.height));
    x = x + _tileMap.tileSize.width/2; // offset
    y = y - _tileMap.tileSize.height/2; // offset
    return ccp(x, y);
}
```

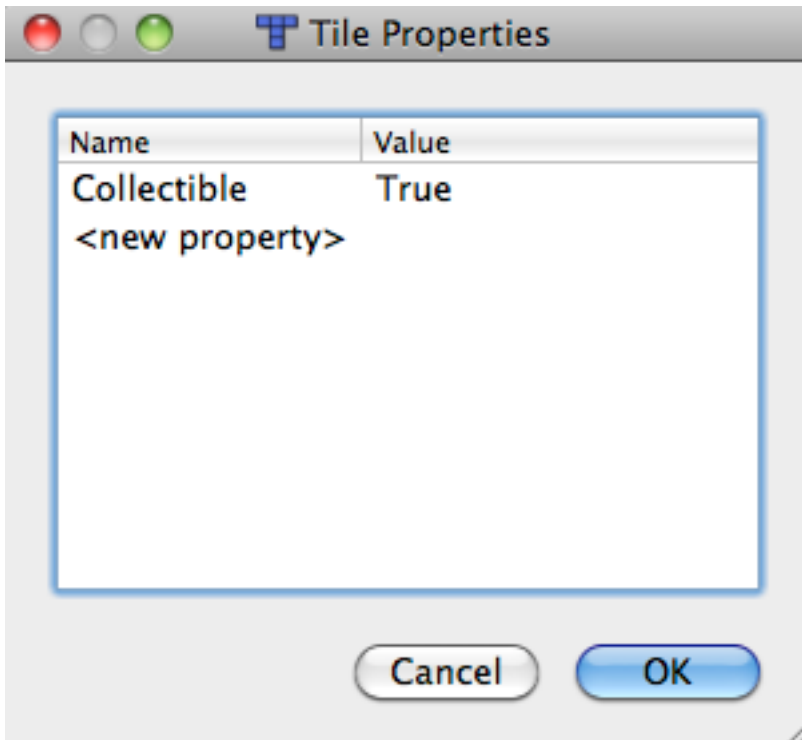
Note we have create 2 new methods. This is because the coordinate space between the tile map in game and the coordinate system in the iphone itself is different. In the tile map, it is a tile-coordinates, where as iphone is pixel-coordinates.



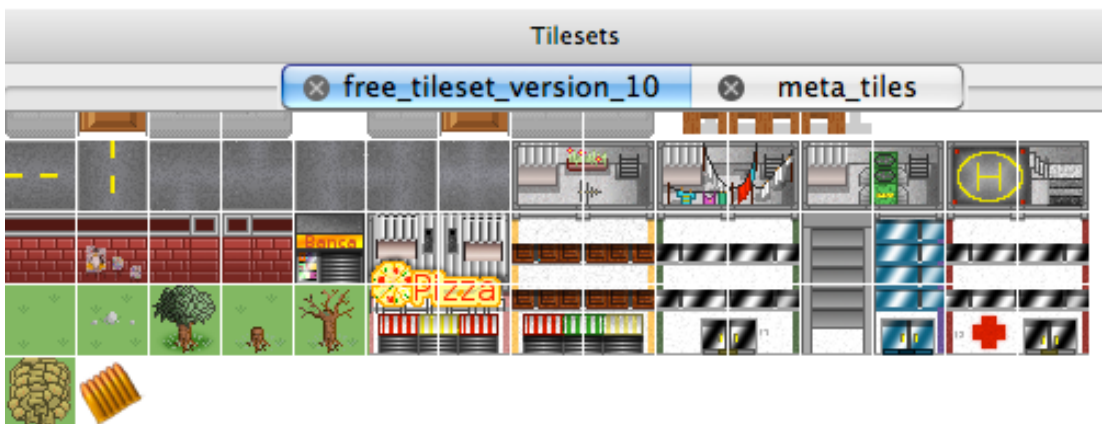


3. Lets edit the map again. This time we shall add a new layer called items. Add a couple of collectibles to the map under this layer. We need to mark these items as collectible. To do this we need to paint every collectibles green using the green tile in meta_tiles. Right click properties the green tile in meta_tiles, set the following.

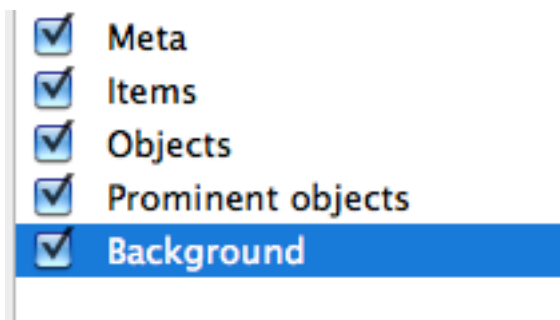




4. With the items layer selected, add in items using the free_tileset_version_10. For this example we shall use the very last item in the tileset that looks like an ammo cartridges image.



5. Make sure the layers are in the order, if not either move up or down the appropriate layers.



6. Select the meta layer, start placing the green tile on top of your collectibles.



7. Save the tmx file. I have already created mine as the latest police&thief.tmx in resource folder.



8. Add these code to HelloWorldScene.

```
// Inside the HelloWorld class declaration
CCTMXLayer *_items;

// After the class declaration
@property (nonatomic, retain) CCTMXLayer *items;

// Right after the implementation section
@synthesize items = _items;
```



```

// In dealloc
self.items = nil;

// In init, right after loading background
self.items = [_tileMap layerNamed:@"Items"];

// Add new method
- (CGPoint)tileCoordForPosition:(CGPoint)position {
    int x = position.x / _tileMap.tileSize.width;
    int y = ((_tileMap.mapSize.height * _tileMap.tileSize.height) - position.y) /
_tileMap.tileSize.height;
    return ccp(x, y);
}

- (CGPoint)positionCoordForTile:(CGPoint)position {
    int x = position.x * _tileMap.tileSize.width;
    int y = -(position.y * _tileMap.tileSize.height - (_tileMap.mapSize.height *
_tileMap.tileSize.height));
    x = x + _tileMap.tileSize.width/2; // offset
    y = y - _tileMap.tileSize.height/2; // offset
    return ccp(x, y);
}

```

9. Modify setPlayerPosition into,

```

-(void)setPlayerPosition:(CGPoint)position {

    CGPoint tileCoord = [self tileCoordForPosition:position];
    int tileGid = [_meta tileGIDAt:tileCoord];
    if (tileGid) {
        NSDictionary *properties = [_tileMap propertiesForGID:tileGid];
        if (properties) {
            NSString *collision = [properties valueForKey:@"Collidable"];
            if (collision && [collision compare:@"True"] == NSOrderedSame) {
                return;
            }
            NSString *collectible = [properties valueForKey:@"Collectible"];
            if (collectible && [collectible compare:@"True"] == NSOrderedSame) {
                [_meta removeTileAt:tileCoord];
                [_items removeTileAt:tileCoord];
            }
        }
    }

    CGPoint positionCoord = [self positionCoordForTile:tileCoord];
    position = positionCoord;
    //_player.position = position;
    CCAction *action = [CCMoveTo actionWithDuration:0.25 position: position];
    [_player runAction:action];
}

```

10. Clean all targets and Build and Go. Refer to PoliceAndThief2-1.zip for the completed source code so far.



11. Lets display an ammo count on the game hud layer. To do this by adding to the HelloWorldScene.h, HelloWorld layer a variable to store the count. We declare in HelloWorld layer because it can then be accessed by the gamehud layer.

```
int _numCollected;

@property (nonatomic, assign) int numCollected;

@synthesize numCollected = _numCollected;
```

12. Declare a function in GameHud layer,

```
- (void)numCollectedChanged:(int)numCollected;
```

13. In the setPlayerPosition method in HelloWorld implementation, modify by adding 2 lines of code,

```
NSString *collectible = [properties valueForKey:@"Collectible"];
if (collectible && [collectible compare:@"True"] == NSOrderedSame) {
    [_meta removeTileAt:tileCoord];
    [_items removeTileAt:tileCoord];
    self.numCollected += 6;
    [_hud numCollectedChanged:_numCollected];
}
```

14. Declare a label in GameHud layer in HelloWorldScene.h for displaying the ammo count.

```
CCLabelTTF *label;
```

15. Right at the top of the init method of the HelloWorld implementation, add,

```
CGSize winSize = [[CCDirector sharedDirector] winSize];
label = [CCLabelTTF labelWithString:@"0" dimensions:CGSizeMake(50, 20)
        alignment:UITextAlignmentRight
        fontName:@"Verdana-Bold"
        fontSize:18.0];
label.color = ccc3(0,0,0);
int margin = 10;
label.position = ccp(winSize.width - (label.contentSize.width/2)
                    - margin, label.contentSize.height/2 + margin);
[self addChild:label];
```

16. Implement the method in GameHud layer implementation.

```
- (void)numCollectedChanged:(int)numCollected {
    [label setString:[NSString stringWithFormat:@"%d", numCollected]];
}
```



17. Add sound effects like for example when the player picks up the ammo. Cocos2D comes built with an audio framework. Import the Audio Engine library.

```
#import "SimpleAudioEngine.h"
```

18. We need to initialize the sounds by adding these lines in init method in HelloWorld layer implementation.

```
[[SimpleAudioEngine sharedEngine] preloadEffect:@"pickup.wav"];  
[[SimpleAudioEngine sharedEngine] preloadEffect:@"ricochet.wav"];  
[[SimpleAudioEngine sharedEngine] preloadEffect:@"death.wav"];
```

19. At setPlayerPosition method modify by adding

```
[[SimpleAudioEngine sharedEngine] playEffect:@"pickup.wav"];
```

```
self.numCollected += 6;  
[_hud numCollectedChanged:_numCollected];  
[[SimpleAudioEngine sharedEngine] playEffect:@"pickup.wav"];
```

20. Drag the 3 audio files pickup.wav and ricochet.wav and death.wav to resources in xcode. We are use the 2 other sounds later.

21. Build and Go. Refer to PoliceAndThief2-2.zip for the source code so far.



Next we are going to add some shooting and animation effects using spritesheets.



22. Lets add shooting effects. Drag and drop bullet0.png to Resources in xcode.

23. We have to enable touch events in our HelloWorld layer. Do this by putting this line in init method in HelloWorld,

```
self.isTouchEnabled = YES;
```

24. And implement the following touch events methods.

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

-(BOOL) ccTouchBegan:(UITouch *)touch withEvent:(UIEvent *)event
{
    return YES;
}

-(void) ccTouchEnded:(UITouch *)touch withEvent:(UIEvent *)event
{
    if (self.numCollected<=0) {
        return;
    }

    CGPoint touchLocation = [touch locationInView: [touch view]];
    touchLocation = [[CCDirector sharedDirector] convertToGL: touchLocation];
    touchLocation = [self convertToNodeSpace:touchLocation];

    CGPoint diff = ccpSub(touchLocation, _player.position);
    if (diff.y > -15 && diff.y < 15 && diff.x > 15)
    {
        // Create a projectile and put it at the player's location
        CCSprite *projectile = [CCSprite spriteWithFile:@"bullet0.png"];
        projectile.position = _player.position;
        [self addChild:projectile];

        // Determine where we wish to shoot the projectile to
        int realX = (_tileMap.mapSize.width * _tileMap.tileSize.width) +
        (projectile.contentSize.width/2);
        int realY = projectile.position.y;
        CGPoint realDest = ccp(realX, realY);

        // Determine the length of how far we're shooting
        float length = realX - projectile.position.x;
        float velocity = 480/8; // 480pixels/8sec
        float realMoveDuration = length/velocity;

        // Move projectile to actual endpoint
        id actionMoveDone = [CCCallFuncN actionWithTarget:self
selector:@selector(projectileMoveFinished:)];
        [projectile runAction:
        [CCSequence actionOne:
        [CCMoveTo actionWithDuration: realMoveDuration
        position: realDest]
        two: actionMoveDone]];
        [_projectiles addObject:projectile];
    }
    else if (diff.y > -15 && diff.y < 15 && diff.x < -15)
    {
```

```

// Create a projectile and put it at the player's location
CCSprite *projectile = [CCSprite spriteWithFile:@"bullet0.png"];
projectile.position = _player.position;
[self addChild:projectile];

// Determine where we wish to shoot the projectile to
int realX = -(_tileMap.mapSize.width * _tileMap.tileSize.width) -
(projectile.contentSize.width/2);
int realY = projectile.position.y;
CGPoint realDest = ccp(realX, realY);

// Determine the length of how far we're shooting
float length = -(realX - projectile.position.x);
float velocity = 480/8; // 480pixels/8sec
float realMoveDuration = length/velocity;

// Move projectile to actual endpoint
id actionMoveDone = [CCCallFuncN actionWithTarget:self
selector:@selector(projectileMoveFinished:)];
[projectile runAction:
 [CCSequence actionOne:
 [CCMoveTo actionWithDuration: realMoveDuration
                position: realDest]
                two: actionMoveDone]];
[_projectiles addObject:projectile];
}
else if (diff.x > -15 && diff.x < 15 && diff.y < -15)
{
// Create a projectile and put it at the player's location
CCSprite *projectile = [CCSprite spriteWithFile:@"bullet0.png"];
projectile.position = _player.position;
[self addChild:projectile];

// Determine where we wish to shoot the projectile to
int realX = projectile.position.x;
int realY = -(_tileMap.mapSize.height * _tileMap.tileSize.height) -
(projectile.contentSize.height/2);
CGPoint realDest = ccp(realX, realY);

// Determine the length of how far we're shooting
float length = -(realY - projectile.position.y);
float velocity = 480/8; // 480pixels/8sec
float realMoveDuration = length/velocity;

// Move projectile to actual endpoint
id actionMoveDone = [CCCallFuncN actionWithTarget:self
selector:@selector(projectileMoveFinished:)];
[projectile runAction:
 [CCSequence actionOne:
 [CCMoveTo actionWithDuration: realMoveDuration
                position: realDest]
                two: actionMoveDone]];
[_projectiles addObject:projectile];
}
else if (diff.x > -15 && diff.x < 15 && diff.y > 15)
{
// Create a projectile and put it at the player's location
CCSprite *projectile = [CCSprite spriteWithFile:@"bullet0.png"];
projectile.position = _player.position;
[self addChild:projectile];

// Determine where we wish to shoot the projectile to
int realX = projectile.position.x;
int realY = (_tileMap.mapSize.height * _tileMap.tileSize.height) +
(projectile.contentSize.height/2);
CGPoint realDest = ccp(realX, realY);

// Determine the length of how far we're shooting
float length = realY - projectile.position.y;
float velocity = 480/8; // 480pixels/8sec
float realMoveDuration = length/velocity;

```

```

    // Move projectile to actual endpoint
    id actionMoveDone = [CCCallFuncN actionWithTarget:self
selector:@selector(projectileMoveFinished:)];
    [projectile runAction:
    [CCSequence actionOne:
    [CCMoveTo actionWithDuration: realMoveDuration
    position: realDest]
    two: actionMoveDone]];
    [_projectiles addObject:projectile];
}
else {
    return;
}

self.numCollected--;
[_hud numCollectedChanged:_numCollected];
[[SimpleAudioEngine sharedEngine] playEffect:@"ricochet.wav"];
}

```

25. There is quite a few steps we have done here. We first determine how far apart the touch position is away from the player position, if it is above a certain threshold, we determine it is a shoot event. Next we determine if the touch is left, right, up or down from the player position. Hence we done a bit of calculation to set a projectile of the shot in the direction of the touch and set a certain projectile speed. We added a projectile sprite and animate the movement.

26. We should declare an array in HelloWorldScene.m under HelloWorld layer,

```
NSMutableArray *_projectiles;
```

27. And in the init method in HelloWorld, initialize the array.

```
_projectiles = [[NSMutableArray alloc] init];
```

28. Finally implement the method in HelloWorld implementation to remove the projectile sprite when the sprite finishes animation action.

```

- (void) projectileMoveFinished:(id)sender {
    CCSprite *sprite = (CCSprite *)sender;
    [self removeChild:sprite cleanup:YES];
    [_projectiles removeObject:sprite];
}

```

29. Later I will explain why we use the array to store reference to all the projectiles. Let's Build and Go. Refer to source code PoliceAndThief2-3.zip so far.

30. First we need to implement a collision method to test if any projectile hit an enemy target. Create this timer in init method in HelloWorld.



```
[self schedule:@selector(testCollisions:)];
```

31. Implement the testCollision method.

```
- (void)testCollisions:(ccTime)dt {  
  
    NSMutableArray *projectilesToDelete = [[NSMutableArray alloc] init];  
  
    // iterate through projectiles  
    for (CCSprite *projectile in _projectiles) {  
        CGRect projectileRect = CGRectMake(  
            projectile.position.x - (projectile.contentSize.width/2),  
            projectile.position.y - (projectile.contentSize.height/2),  
            projectile.contentSize.width,  
            projectile.contentSize.height);  
  
        NSMutableArray *targetsToDelete = [[NSMutableArray alloc] init];  
  
        // iterate through enemies, see if any intersect with current projectile  
        for (CCSprite *target in _enemies) {  
            CGRect targetRect = CGRectMake(  
                target.position.x - (target.contentSize.width/2),  
                target.position.y - (target.contentSize.height/2),  
                target.contentSize.width,  
                target.contentSize.height);  
  
            if (CGRectIntersectsRect(projectileRect, targetRect)) {  
                [targetsToDelete addObject:target];  
            }  
        }  
  
        // delete all hit enemies  
        for (CCSprite *target in targetsToDelete) {  
            [_enemies removeObject:target];  
            [self removeChild:target cleanup:YES];  
            [[SimpleAudioEngine sharedEngine] playEffect:@"death.wav"];  
            ///////////////////////////////////////////////////////////////////  
            ///////////////////////////////////////////////////////////////////  
        }  
  
        if (targetsToDelete.count > 0) {  
            // add the projectile to the list of ones to remove  
            [projectilesToDelete addObject:projectile];  
        }  
        [targetsToDelete release];  
    }  
  
    // remove all the projectiles that hit.  
    for (CCSprite *projectile in projectilesToDelete) {  
        [_projectiles removeObject:projectile];  
        [self removeChild:projectile cleanup:YES];  
    }  
    [projectilesToDelete release];  
}
```

32. Build and Go.

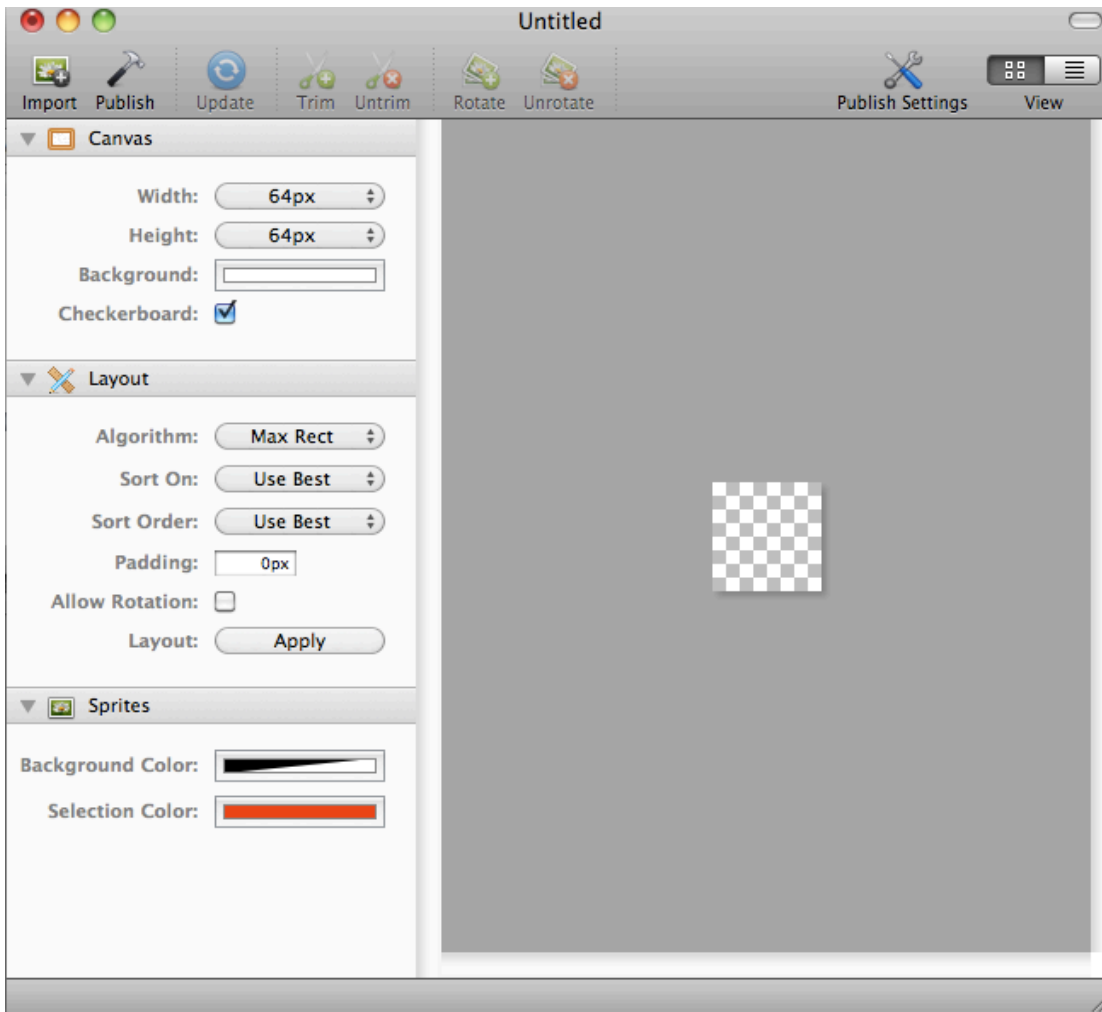
33. Finally we shall learn how to animate sprites using a sprite sheet. For example we animate how an enemy die. Basically to create an animation using sprite sheet we need to create a sprite sheet png image file, and a plist that store all the coordinates where each individual image is placed on the sprite sheet. Lets create a sprite sheet like the following. We can use a free tool call Zwoptex from

'zwoptexapp.com/'



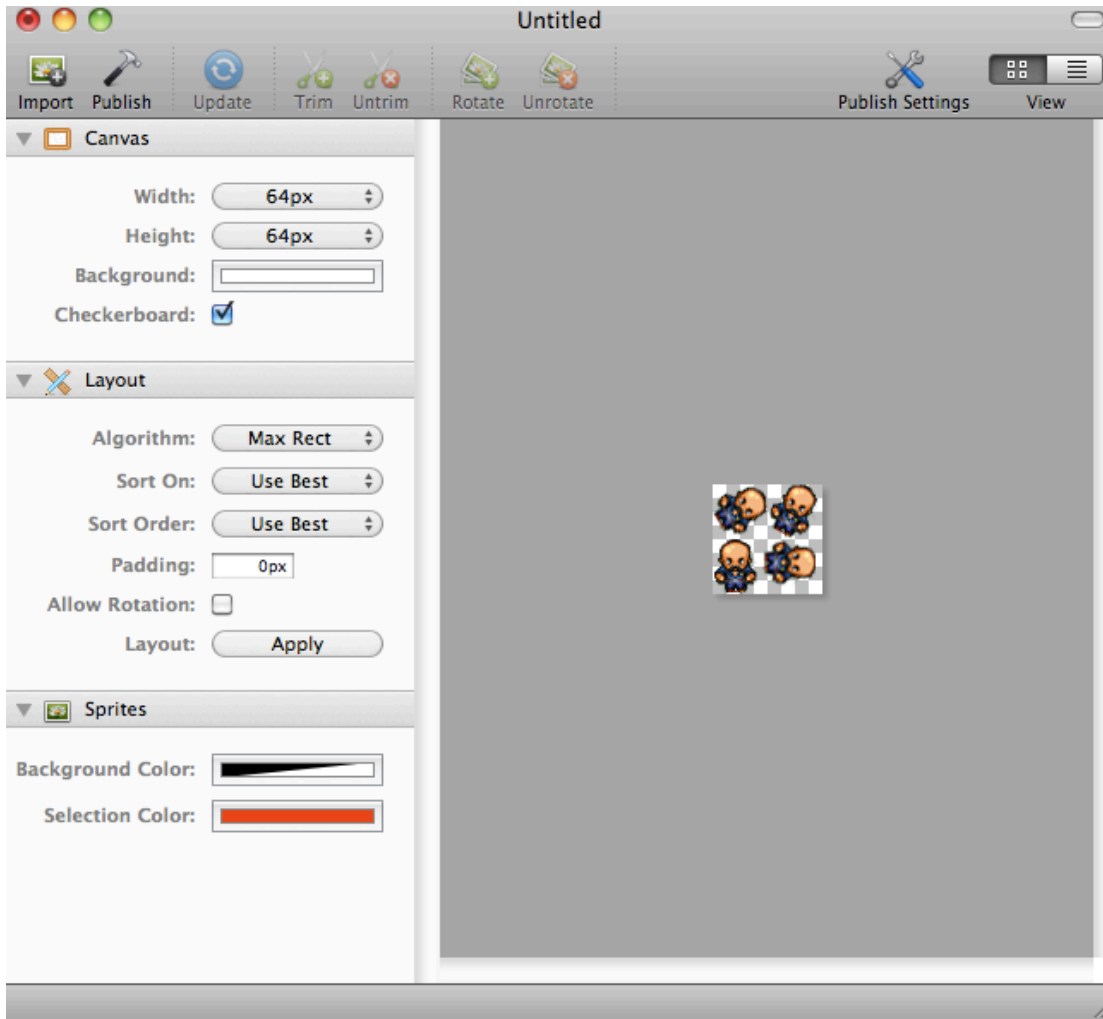


34. Open Zwoptex, set the canvas 64 x 64. Set the padding to 0 pixels.



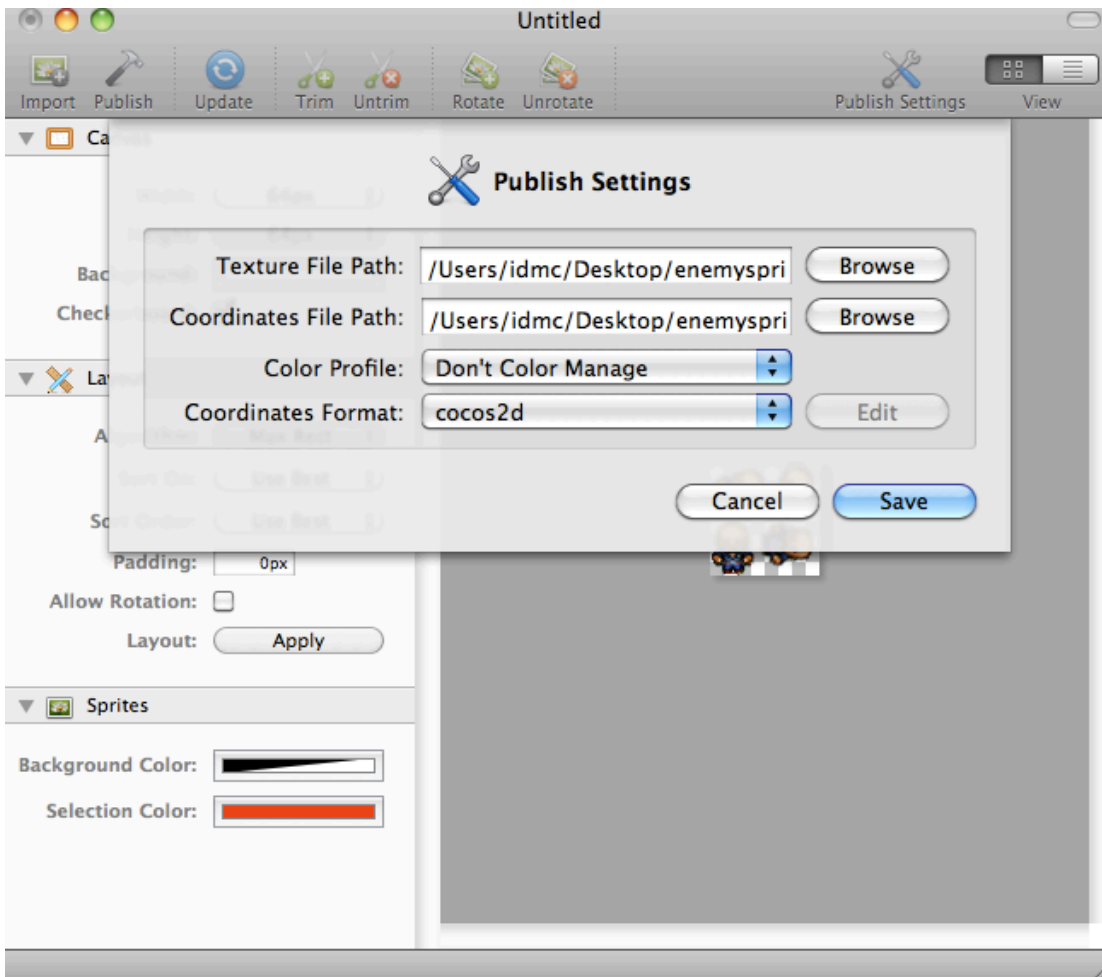
35. Click import and import the 4 individual enemy images. Click apply layout.





36. Click publish. Make sure the coordinate format is in cocos2d.





37. Drag the generated enemysprites.png and enemysprites.plist to Resources in xcode. Implement the following codes in testCollision method under in between the // region.

```

////////////////////////////////////
CCSpriteFrameCache * cache = [CCSpriteFrameCache sharedSpriteFrameCache];
[cache addSpriteFramesWithFile:@"enemysprites.plist"];
CCAnimation * animation = [[CCAnimation alloc] initWithName:@"idle" delay:1/8.0];
for ( int i=1; i < 5; ++i )
{
    NSString * fname = [NSString stringWithFormat:@"enemy%01i.png", i];
    [animation addFrame:[cache spriteFrameByName: fname]];
}
CCSprite *targetenemy = [CCSprite spriteWithSpriteFrameName:@"enemy1.png"];
id action = [CCAnimate actionWithAnimation: animation restoreOriginalFrame:NO];
[targetenemy runAction:action];
targetenemy.position = target.position;
[self addChild:targetenemy];
id actionDone = [CCCallFuncN actionWithTarget:self selector:@selector(enemydiefinish)];
[targetenemy runAction:
    [CCSequence actionOne:
        [CCMoveTo actionWithDuration: 1.0
            position: targetenemy.position]
        two: actionDone]];
////////////////////////////////////

```

38. Implement the method,



```
-(void)enemydiefinish:(id)sender {  
    CCSprite *sprite = (CCSprite *)sender;  
    [self removeChild:sprite cleanup:YES];  
}
```

