Events & Touch Detection in Cocos2D-X

**This is not a step-by-step tutorial. Please ensure you download the code from GitHub and use it as a reference. I will introduce methods and variables but expect you to look for these in the source code file to supplement what is written in this short document.**

Editing HelloWorldScene as everyone will have that. Note that HelloWorld is the name of the class (not HelloWorldScene, this is the file name) and that HelloWorld inherits from a Layer!

Word of warning – we are going to be setting up callback methods and your compiler will complain if it does not have all valid methods. Please ignore any build errors until you have finished implementing EVERYTHING!

**In the HelloWorldScene HEADER file:**

Add 4 methods to detect the touches we need to capture. These are virtual methods and can be found in the header file.

onTouchBegan();

onTouchEnded();

onTouchMoved();

onTouhCancelled();

I also add a label to display touch information

Cocos2d::Label labelTouchInfo;

**Now we move into the HelloWorldScene CPP file**

Time for some additional includes. I plan to allow the user to touch any part of the screen, move the label to that section of the screen, change the message and change the colour of the text based on randomly generated numbers. We need to include the header files for random number generation. These are:

#include <cstdlib>

#include <ctime>

Now we need to initialise the pseudo-random seed, I have done this near the top of the Init method using the following statement:

srand(time(NULL));

Again, in the Init method (but a little further down this time), right after we add the rootnode to the layer I create the Label we will use to give the user feedback. Look for the line

labelTouchInfo = Label::createWithSystemFont("Touch or click on the screen", "Arial", 30);

This really should be a self-explanatory statement, you should also be able to use your skills of deductive reasoning to figure out what the three parameters we pass to the method mean.

Right after that statement I set the ORIGINAL position of the label to the centre of the screen with the following statement:

labelTouchInfo->setPosition(cocos2d::Vec2(Director::getInstance()->getVisibleSize().width / 2, Director::getInstance()->getVisibleSize().height / 2));

Now we have the code that sets up a listener for touches, adds the callback methods that should be called given a certain touch event and then we add these events to the main event dispatcher, which is responsible for calling any class listening for the events. I will not explain this here, please go to http://www.cocos2d-x.org/wiki/EventDispatcher\_Mechanism

Have a GOOD look at the following code

// Create a custom event listener

auto touchListener = EventListenerTouchOneByOne::create();

// Assign the event methods to the event listener (known as callbacks)

touchListener->onTouchBegan = CC\_CALLBACK\_2(HelloWorld::onTouchBegan, this);

touchListener->onTouchEnded = CC\_CALLBACK\_2(HelloWorld::onTouchEnded, this);

touchListener->onTouchMoved = CC\_CALLBACK\_2(HelloWorld::onTouchMoved, this);

touchListener->onTouchCancelled = CC\_CALLBACK\_2(HelloWorld::onTouchCancelled, this);

/\* For more information on the eventdispatcher mechanism (and how events in Cocos work in general) go to http://www.cocos2d-x.org/wiki/EventDispatcher\_Mechanism \*/

// Add the event listener to the event dispatcher

\_eventDispatcher->addEventListenerWithSceneGraphPriority(touchListener, this);

The final thing we need to do in our Init() method is add the label we created to the layer so it can be displayed

this->addChild(labelTouchInfo);

Now we need to implement the method definitions we created in the header file at the start. I have done this with the following code.

bool HelloWorld::onTouchBegan(cocos2d::Touch\* touch, cocos2d::Event\* event)

{

}

void HelloWorld::onTouchEnded(cocos2d::Touch\* touch, cocos2d::Event\* event)

{

}

void HelloWorld::onTouchMoved(cocos2d::Touch\* touch, cocos2d::Event\* event)

{

}

void HelloWorld::onTouchCancelled(cocos2d::Touch\* touch, cocos2d::Event\* event)

{

}

Within these methods, we write the code that will be executed WHEN THE EVENT IN QUESTION IS CALLED. I have only added code to the onTouchBegan method. Let’s look at the code now.

// Create three random numbers that will be the colour of our text

GLubyte red = rand() % 255;

GLubyte green = rand() % 255;

GLubyte blue = rand() % 255;

// Create the colour based on our random numbers

cocos2d::Color3B myColour = cocos2d::Color3B(red, green, blue);

labelTouchInfo->setPosition(touch->getLocation());

labelTouchInfo->setString("You pressed HERE!");

labelTouchInfo->setColor(myColour);

return true;

The first three statements of this method are concerned with creating random numbers, between the values of 0 and 255, that will be used as our Red, Green & Blue component for the colour of the text.

The 4th statement concerns CREATING the Colour value.

Statements 5 & 6 set the position and text of the label respectively.

Finally, we set the colour of the text using our colour value.

# Additional Work Based on This Text

In Cocos, you cannot just attach events and listeners to sprites. You will have to implement code similar to what has been provided then determine if your touch->getLocation intersects with your Sprite’s boundingbox. Sprite DOES have a getBoundingBox() method so this should be trivial to implement.

If you do not know the mechanical difference between onTouchBegan and onTouchEnded why not **cut and paste** the code from onTouchBegan (leave the return true in though) and move it to the onTouchEnded() method. Run the project and see the difference.

# Other questions

I have had other questions relating to CC\_CALLBACK2, what it is and how it works. CC\_CALLBACK” is a macro created in C++ and the tutorial that all this code has been based on can be found at <http://www.gamefromscratch.com/post/2014/10/03/Cocos2d-x-Tutorial-Series-Handling-Touch-and-Mouse-Input.aspx>

This tutorial also gives a good explanation of what CC\_CALLBACK2 is, it shows you how to generate touch events and touch move events only when pressing a sprite and it also shows how to multi-touch. There is no excuse for not implementing at least basic touch functionality.

The last question I have often been asked is about the “Game Loop” and how we update and draw the additional sprites you will likely add. I will NOT answer this succinctly here because you have already been given a code example of how this works. If you are wondering WHERE or WHEN you received this information I suggest you (re)download Paul’s Flappy bird example. The pipes in this example move from the right hand side of the screen to the left side of the screen and are then re-instantiated. Paul wrote an Update() method and importantly scheduledUpdate() within his code which tells Cocos that this particular class should be updated and therefore needs an update method.