

JACOS2D-X

The free opensource cross platform game engine

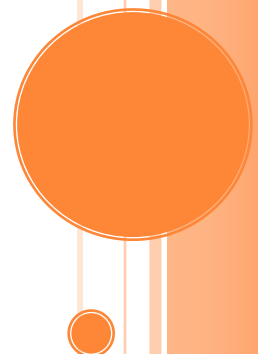
Do you want to develop games in smart phones? Do you want your game to be cross platform? And do you want to use Javascript for your game development, ...

Thai-Duong Nguyen

nguyen.duong@jacos2d-x.org

www.jacos2d-x.org

3/10/2013



Contents

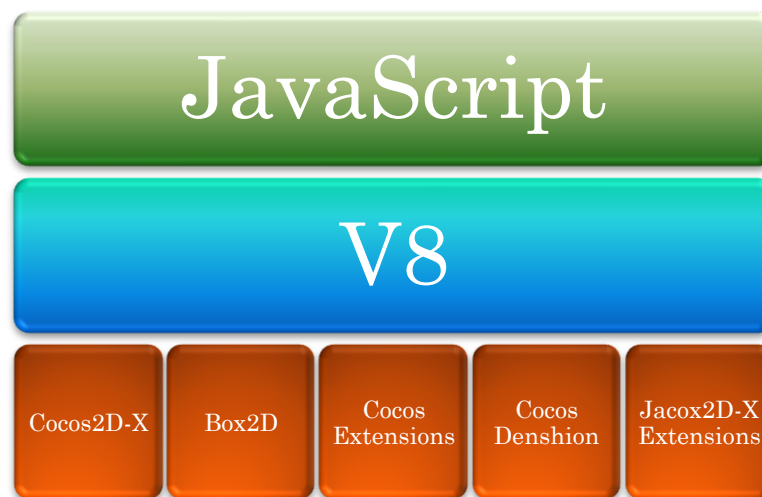
OVERVIEW	2
PREREQUISITES.....	3
START THE ADVENTURE	4
ARRIVED AT THE FIRST STATION	17
CONTINUE... ..	19
THANKS.....	19

JACOS2D-X

The free opensource cross platform game engine

OVERVIEW

Jacos2D-X is a game engine developed based on the popular Cocos2D-X engine. It is built on top of the super Google's V8 Javascript engine, it means that you can use Javascript for your game or application development. The image below will describe how Jacos2D-X works in detail:



Jacos2D-X inherits almost every library of Cocos2D-X:

1. **The graphical library:** help you create Scene, Layer, Sprite, etc, and your desired animations. However, differed from Cocos2D-X, because you develop game in Javascript, there are several libraries are added to increase the flexibility.
2. **The Box2D physics engine:** this is a very popular, high performance physics engine. You can use Box2D to create great physics effects for your game.
3. **The Cocos Extensions library:** this is the library for UI elements, loading .ccbi generated by CocosBuilder. With Jacos2D-X, the CocosBuilder will come in very handy to design scenes in your game.
4. **The sound FX and Music Cocos Denshion library:** help you bring sound and music effects to your game. Besides, it can be used to change the volumn, turn on/off sound and music as your game request.
5. **The Jacos2D-X extensions library:** include APIs for file IO, process binary data. Currently, I am working on this library to provide more helpful APIs.

Almost every simple operation is executed at the native layer, so the performance will be high. The libraries are quite sufficient to bring whatever your creation become real.

Jacos2D-X provides you a remotely game loading mechanism. On other words, you only have to enter the URL to your game source code (Javascript); then it will be loaded to your device for execution. This mechanism has a special meaning in **game development** as you don't have to make build, load it into device, and run it like before. All you have to do is just a simple operation like refreshing a web page. It means that you can definitely store your game in any game sever as you want. All is as simple as input the URL and there we run. Notice that you use this advantage for **development phase** only.

Jacos2D-X uses Google's V8 Javascript engine. Why is V8 used here while SpiderMonkey Javascript engine has already been integrated to Cocos2D-X? The problem here is about performance, V8 has well managed this problem – the most essential issue of Javascript. In order to run in iOS, I had to spend so much effort porting another iOS version of V8. Presently, I am still improving this version to have higher performance.

At this point, I think you all had an overview about Jacos2D-X. In the following sections, I will present you how to develop with this engine in detail. However, you must prepare something before we can start. Let's go!!!

PREREQUISITES

1. **Download Jacos2d-x:** Visit www.jacos2d-x.org to download the latest version.
2. **Compile and run for Android:**
 - At first, visit the home page of [Android NDK](http://developer.android.com/ndk/) to get and install Android NDK. The detail instructions about how to install in MAC OS, Linux, and Windows are already given there.
 - Download and install Eclipse – the vesion supporting JavaScript at <http://www.eclipse.org>
 - In the console window (Cygwin is needed in case you are using Windows), move to the folder of Jacos2D-X. Move to the directory Jacos2d-x/proj.android. Run the script **build_native.sh** to export the file **game.so**.
 - Next, open Eclipse then import 2 projects: Jacos2d-x/proj.android and cocos2dx/platform/android/java. Compile and run Jacos2d-x/proj.android
3. **Compile and build for iOS:** All you have to do is opening the project jacos2dx.xcodeproj in Jacos2d-x/proj.ios and then build it just as every other XCode projects.
4. **Compile and run for Windows:** Run VC 2010, open project Jacos2dx-win32.vc2010.sln then build it as usual.

5. **Config autocomplete for Eclipse:** You can use *jacos2dx_dummy_lib.js* which is under documents folder to do this.
6. **Setup the webserver:** You can use Apache, IIS, or any other ones that you're familiar (mine is EasyPHP).

START THE ADVENTURE

1. Hello World program

As usual, we will start by the popular Hello World program. Firstly, please go to folder **www** of EasyPHP and create a file so called helloworld.js with the following content:

```
printf("Hello World!");
```

OK. Don't be sad if this simple statement disappointed you. This is just the first statement to print out a message to the console. Next, I will describe how to display the "Hello World" in your game screen.

```
printf("Hello World!");

var CCScene = cocos2d.CCScene;
var CCLayer = cocos2d.CCLayer;
var CCLabelTTF = cocos2d.CCLabelTTF;


//get director
var director = cocos2d.CCDirector.sharedDirector();

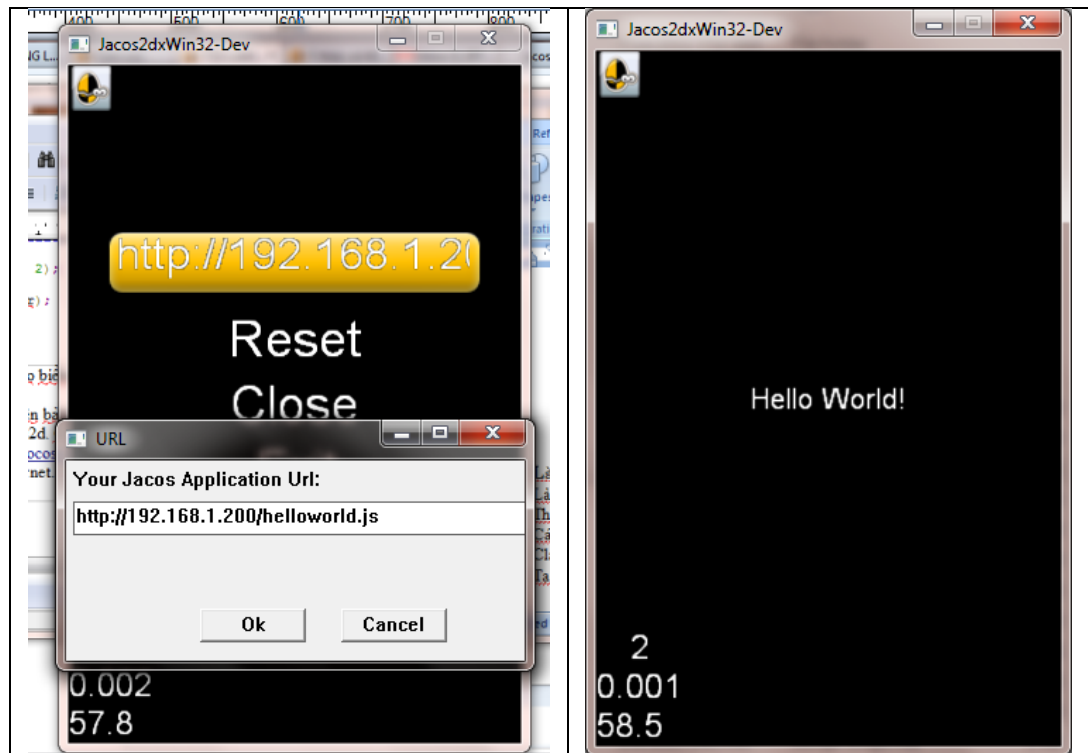
//create main scene and layer
var helloWorldScene = CCScene.create();
var helloWorldLayer = CCLayer.create();

//create label
var lbHelloWorld = CCLabelTTF.create("Hello World!", "Arial",
20);

lbHelloWorld.setPosition(480 / 2, 320 / 2);
helloWorldLayer.addChild(lbHelloWorld);
helloWorldScene.addChild(helloWorldLayer);

director.pushScene(helloWorldScene);
```

Please run Jacos2D-X, click the Jacos2D-X icon  and enter a URL to the file helloworld.js as the below figure:



Now, click Reset to start the program.

2. Work with Node and Sprite

Node is the most basic graphical element in Jacos2D-X. Similar to Sprite, it's an element to display images. You can rotate, scale, move a Sprite or Node object at ease. The below code snippet will describe how to use Node and Sprite:

```
var CCScene = cocos2d.CCScene;
var CCLayer = cocos2d.CCLayer;
var CCSprite = cocos2d.CCSprite;

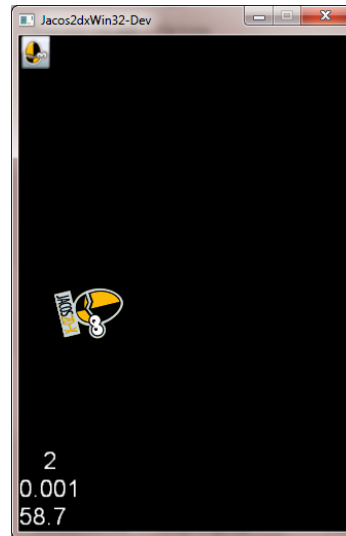
//get director
var director = cocos2d.CCDirector.sharedDirector();

//create main scene and layer
var helloWorldScene = CCScene.create();
var helloWorldLayer = CCLayer.create();

//create sprite
var sprite = CCSprite.create("jacos2dx.png");

sprite.setPosition(480 / 2, 320 / 2);
helloWorldLayer.addChild(sprite);
helloWorldScene.addChild(helloWorldLayer);

director.pushScene(helloWorldScene);
```



3. Work with Layer

Layer is an important element to build a Scene. You can place any Node in a Layer as you want. Using layers will help you easily manage other graphical elements. Jacos2D-X also supports CCLayer class just as Cocos2D-X to provide this function.

4. How to call a callback and use the schedule

In game development, there are many cases that you need to use callback functions. For instance, when you touch a Menu Item, schedule a task, or execute a call from an Action. Jacos2D-X also supports callback functions with Javascript. You will need 2 parameters when making a callback: target and selector. Selector is the callback function, and target is the object which owns that callback function. Following example will describe how to make a callback:

```
var menuItem1 = CCMenuItemImage.create("img1.png", "img2.png",
this, function() {
    printf("menu item 1 is touched");
});
```

In some circumstances, you don't have to pass in the target parameter. In such cases, target will be the primary object which is in charge of calling callback. For example, when you schedule a task for a Node:

```
var MyNode = SubClass("MyNode", cocos2d.CCNode, function() {
    this._base();
    this._x = 0;
    var onTick = function(dt) {
        this._x++;
        if (this._x >= 1000)
            this._x = 0;
    };
    this.schedule(onTick); //target is this
});
```

If you are familiar with Javascript, you indeed must know how to create a callback with the bind(object) function as below:

```
set_call_back(function() {
    this._x++;
}).bind(this);
```

Jacos2D-X also supports this bind function in Native. As below:

```
bind(_function, object[, param1, param2, ....]);
```

This function returns a new function that has been binded with your desired object. Below is an example:

```
var newFunc = bind(function() {
    printf("My name: " + this._name);
    printf("Age: " + this._age);
}, { _name: "Jacos2D-x", _age : 18 });
newFunc();
```

or

```
bind(function(scale) {
    printf("size: " + (this._size * scale));
}, { _size : 2 })(10.0);
```

5. Class and inheritance

- Create new class: there are many ways to create a Class in Javascript. With Jacos-2D-X, you can use any one of them:

```
var MyClass = function(x, y, z) {
    //private member
    var _member1;
    var _member2;
    //private methods
    var _method1 = function() {
    };
    var _method2 = function() {
    };
    //var constructor
    (function(x, y, z) {

    })(arguments);

    //public methods
    this.methodA = function() {
    };
    //public members
    this.m_memberA = 1;
};
MyClass.prototype.methodB = function() {};
MyClass.prototype.methodC = function() {};
MyClass.prototype.m_member3 = 5;
```

- To inherit from a given class, Jacos2D-X provides you an extreme tool to do it so called SubClass. This function help you

quickly create a new class by inheriting an existing one with high performance. Here's how to use it:

```
var MyClass = SubClass("MyClass", ParentClass, function(arg1,
arg2, arg3,...) { //constructor
    this._super(arg1, arg2, arg3,...);
}, [get_parent_method]);
```

Where `_super()` is the default method which is added by `SubClass` to call the super class's constructor. Note that `this._super()` must be called at first to initialize super class's elements. It is also possible to pass some parameters to the `_super()` method. For example:

```
this._super(1, "Hello");
this._super(x, y, z);
```

`get_parent_method` is a Boolean parameter. This parameter will be very useful if your class call many super class's methods. If it is set to true, the call `this._super()` will return an object with the original method of parent's class; also, `this._super` refers to this object instead of parent's constructor. I will explain it and how to use it in more detail later. However, you have to carefully consider whether or not using this feature because it may slow down the performance. If you don't pass this parameter, the `SubClass` will take the default value of it: false.

```
var ClassA = function() {
    this.printString = function(s) {
        printf(s);
    };
    this.getName = function() {
        return "I'm ClassA!";
    };
}
ClassA.prototype.setAge = function(age) {this._age = age;}
ClassA.prototype.getAge = function() {return this._age;};

//ClassB inherit from ClassA
var ClassB = SubClass("ClassB", ClassA, function() {
    this._super();
    this.getName = function() { //overwrite method
        return "I'm ClassB!";
    };
});
```

- Call the inherited method. Suppose that you want to call the method `getName` of `ClassA` from inside of `ClassB`'s `getName` function. How to do that? There are two options for you:

Option 1

```
var ClassB = SubClass("ClassB", ClassA, function() {
    this._super();
```

```

    var old_getName = this.getName;
    this.getName = function() { //overwrite method
        var parentName = this._call(old_getName); //call
ClassA's method
        return "I'm ClassB! My parent is " + parentName;
    };
});
ClassB.prototype.getAge = function() {
    return 20;
};
ClassB.prototype.getFamilyAge = function() {
    var parentAge = this._call(ClassA.prototype.getAge,
this);
    return this.getAge() + parentAge;
}

```

Option 2

```

var ClassB = SubClass("ClassB", ClassA, function() {
    this._super();
    var old_getName = this.getName;
    this.getName = function() { //overwrite method
        var parentName = this._call(this._super.getName);
//call ClassA's method
        return "I'm ClassB! My parent is " + parentName;
    };
}, true); //added true parameter
ClassB.prototype.getAge = function() {
    return 20;
};
ClassB.prototype.getFamilyAge = function() {
    var parentAge = this._call(this._super.getAge);
};

```

SubClass creates a default `_call` method, this is how to use it:

```

this._call(function_object, [arg1, arg2, arg3,...])

```

This method will call `function_object` as a member of this object.

Note that when creating a new class, it's best to use `SubClass`. Even if the class is not inherited from any other one, you still should use `SubClass` and extend the `Object` class.

6. Create and use your own library

When developing a software or game project, you may have to write many files. Splitting up the files also helps the management task easier, especially for the project with many contributors. For this reason, `Jacos2D-X` also enables you to define your own library as below:

File `tree_lib.js`

```

var Tree = SubClass("Tree", Object, function() {
    this._base();
});
Tree.prototype.makeLeaf = function(leafCount) {};
Tree.prototype.setColor = function(color) {};

```

```

Tree.prototype.hasFlower = function() {return false;};

var Rose = SubClass("Rose", Tree, function() {
    this._base();
});
Rose.prototype.hasFlower = function() {return true;};

//Export the Classes you want to use as libraries
Exports(Tree, Rose);

```

Use the method Exports(Class1[, Class2, Class3, Class4,...]) to export classes that you want to be the library.

This is how to use the library tree_lib.js (suppose you place tree_lib.js in the same directory with main.js)

```

Import("tree_lib.js/*");

var rose1 = new Rose();
printf("Rose has flower? " + rose1.hasFlower());

```

Import function is used to import a given library. You can import 1, more or even all of the Class within a library. For example, if you want to use the Rose class only:

```

Import("tree_lib.js/Rose");

```

Or if you want to import both Rose and Tree classes

```

Import("tree_lib.js/Rose");
Import("tree_lib.js/Tree");

```

Or even more simple like this:

```

Import("tree_lib.js/Rose, Tree");

```

To import all classes in tree_lib.js

```

Import("tree_lib.js/*");

```

Even importing a library from the internet:

```

Import("http://www.test.com/www_lib.js/*");

```

Not only importing your own library is possible, but also you can import native libraries such as cocos2d, box2d ... For example:

```

//import class CCNode
Import("cocos2d/CCNode");

//import CCSprite, CCLayer, CCScene
Import("cocos2d/CCSprite, CCLayer, CCScene");

//Import all
Import("Box2D/*");

```

Assume you have two files as below:

lib_a.js

```

Import("lib_b.js/*");

var ClassA = SubClass("ClassA", Object, function() {
    printf("this is classA");
});

ClassA.prototype.createClassB = function() {
    return new ClassB();
}
Exports(ClassA);

```

lib_b.js

```

//import all classes
Import("lib_a.js/*");

var ClassB = SubClass("ClassB", ClassA, function() {
    printf("this is classA");
});

Exports(ClassB);

```

In this case, you won't be able to import because there is a recursive error, and Jacos2D-X will prompt you about it. However, you can still do this by using the Import function explicitly (don't use /*).

```

//must use
Import("lib_b.js/ClassB");

//don't use
Import("lib_b.js/*");

```

Action in details:

```

//lib_a.js
Import("lib_b.js/ClassB");

var ClassA = SubClass("ClassA", Object, function() {
    printf("this is classA");
});

ClassA.prototype.createClassB = function() {
    return new ClassB();
}
Exports(ClassA);

//-----
//lib_b.js
Import("lib_a.js/ClassA");

```

```

var ClassB = SubClass("ClassB", ClassA, function() {
    printf("this is classA");
});
Exports(ClassB);

```

Recursive error will be gone completely when you declare explicitly like above.

7. Using CocosBuilder

- Creating CocosBuilder project
Jacos2D-X is currently support CocosBuilder version 3.0 alpha. You can download and learn how to create a project with it at: www.cocosbuilder.com. CocosBuilder also supports integration with Javascript. In Jacos2D-X, there is already a library to load .ccbi files generated by CocosBuilder.
- Integerate with Javascript
In order to load .ccbi files, please do as below:

```

var ccbReader = new
CCBReader(CCNodeLoaderLibrary.sharedCCNodeLoaderLibrary());
var scene =
ccbReader.createSceneWithNodeGraphFromFile(fileName);
CCDirector.sharedDirector().pushScene(scene);

```

Where, filename is the name of .ccbi file to be loaded. In the cocos dragan sample (based on the sample cocos dragon at cocosbuilder.com), I described carefully how to develop a game by using Jacos2D-X and CocosBuilder.

8. Using the Box2D physics engine library

Jacos2D-X fully supports the Box2D physics engine. You may figure out how to use it through my samples or API reference.

9. Sound FX and Music

Jacos2D-X also supports sound FX and music effect through Cocos Denshion library. It consists of many methods such as: play effect, play background, set volumn, etc. Once again, you may read my samples or the API reference when using this library.

10.Process binary data

If you are familiar with Java, you must have used Stream at least once. Jacos2D-X also provides Streams to interact with different kinds of data. Streams are used for IO operations and are described more carefully in API reference.

11.Open an Http connection

If you want to open an Http connection, it can be done easily in Jacos2D-X. Please see the following example:

```

Import("cocos2d.extension/CCHttpRequest, CCHttpClient,
CCHttpResponse");
var request = new CCHttpRequest();
request.setRequestType(CCHttpRequest.kHttpGet);
request.setUrl("http://google.com");
request.setResponseCallback(this, function(sender, response) {
    var stream = response.getInputStream();
    var data = "";
    while (stream.available() > 0)
    {
        //data += stream.readUTF8Line(jacos2dx._CR_LF);
        printf(stream.readUTF8Line(jacos2dx._CR_LF));
    }
    printf("data: " + data);
});

var httpClient = CCHttpClient.getInstance();
httpClient.send(request);

```

12. File IO in Jacos2D-X

File is a special IO stream which is used for durable storage in the devices. Below is an example about how to read/write file with Jacos2D-X:

Create a new file and write some texts:

```

Import("cocos2d/CCFileUtils");
Import("jacos2dx/*");

var writeablePath = CCFileUtils.sharedFileUtils().getWriteablePath();
printf("Path: " + writeablePath);

var fs = new FileStream(writeablePath + "myfile.txt",
FileStream.MODE_CREATE);
var os = fs.openOutputStream();

os.writeUTF8Line("Helloworld", jacos2dx._CR_LF);
os.flush();
fs.close();
//os.writeUTF8Line("cannot write this line!", jacos2dx._CR_LF);

```

Read file :

```

Import("cocos2d/CCFileUtils");
Import("jacos2dx/*");

var writeablePath = CCFileUtils.sharedFileUtils().getWriteablePath();

var fr = new FileStream(writeablePath + "myfile.txt",
FileStream.MODE_OPEN_READ);
var is = fr.openInputStream();
var s = is.readUTF8Line(jacos2dx._CR_LF);
printf("data: " + s);
fr.close();

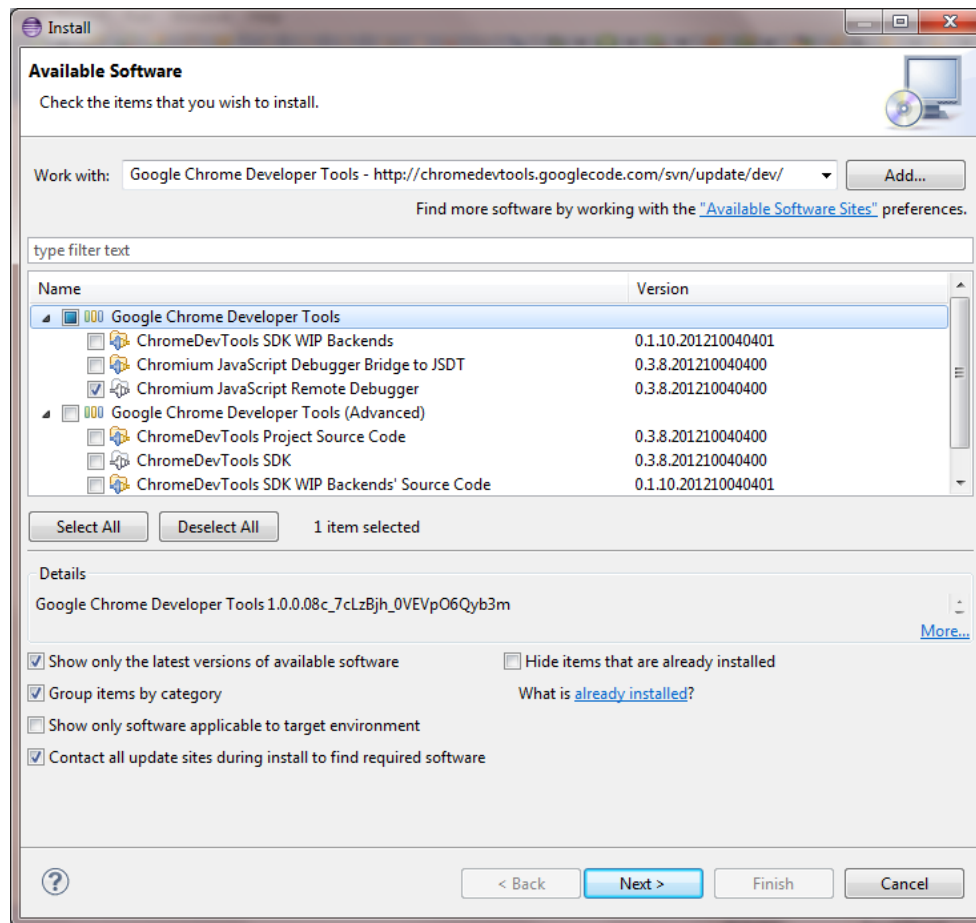
```

13. Use Eclipse's debug tool

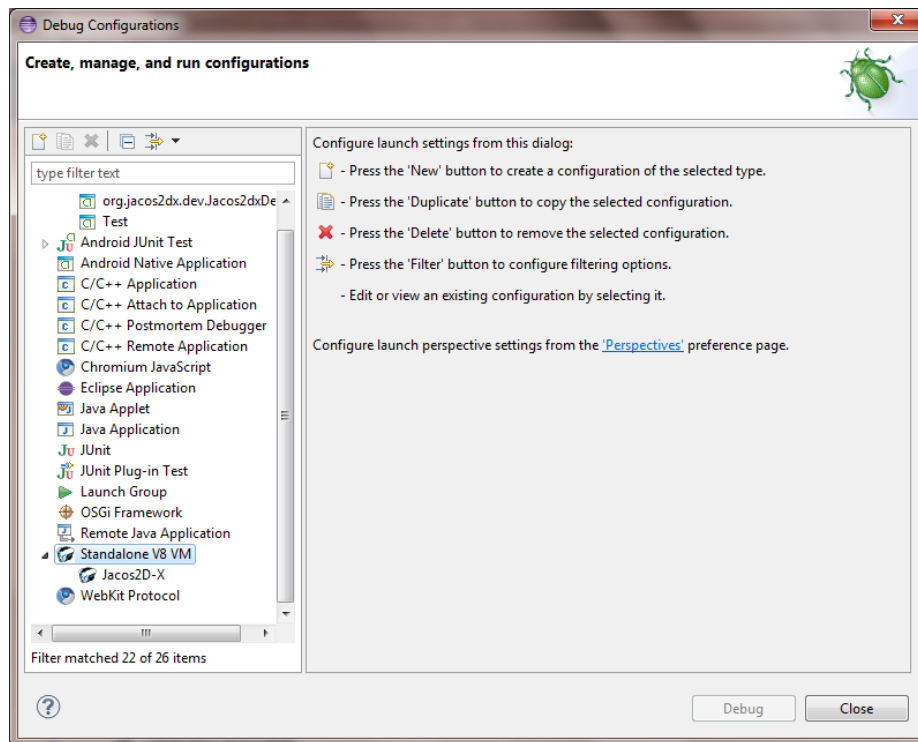
Open Eclipse, then open menu Help/ Install New Software. In the Work with text field, please enter:

Google Chrome Developer Tools - <http://chromedevtools.googlecode.com/svn/update/dev/>

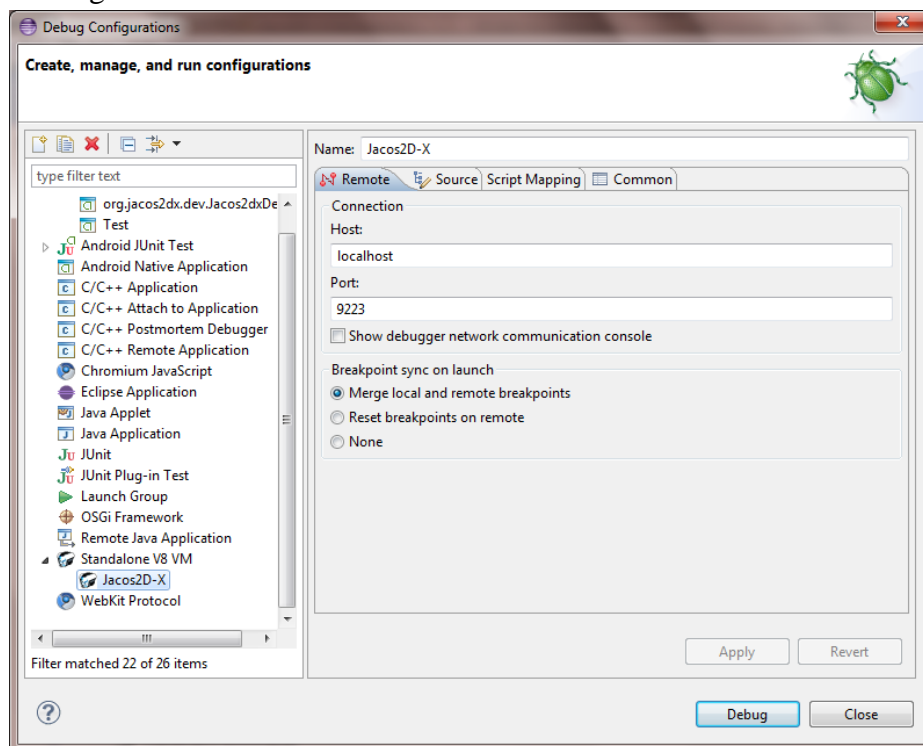
Please select as the below image and then press Next to install. Make sure your internet connection is stable during the installation process!



Open menu Run/Debug Configurations. See **Standalone V8 VM**



Double click to Standalone V8 VM to create new debug configuration like the following:

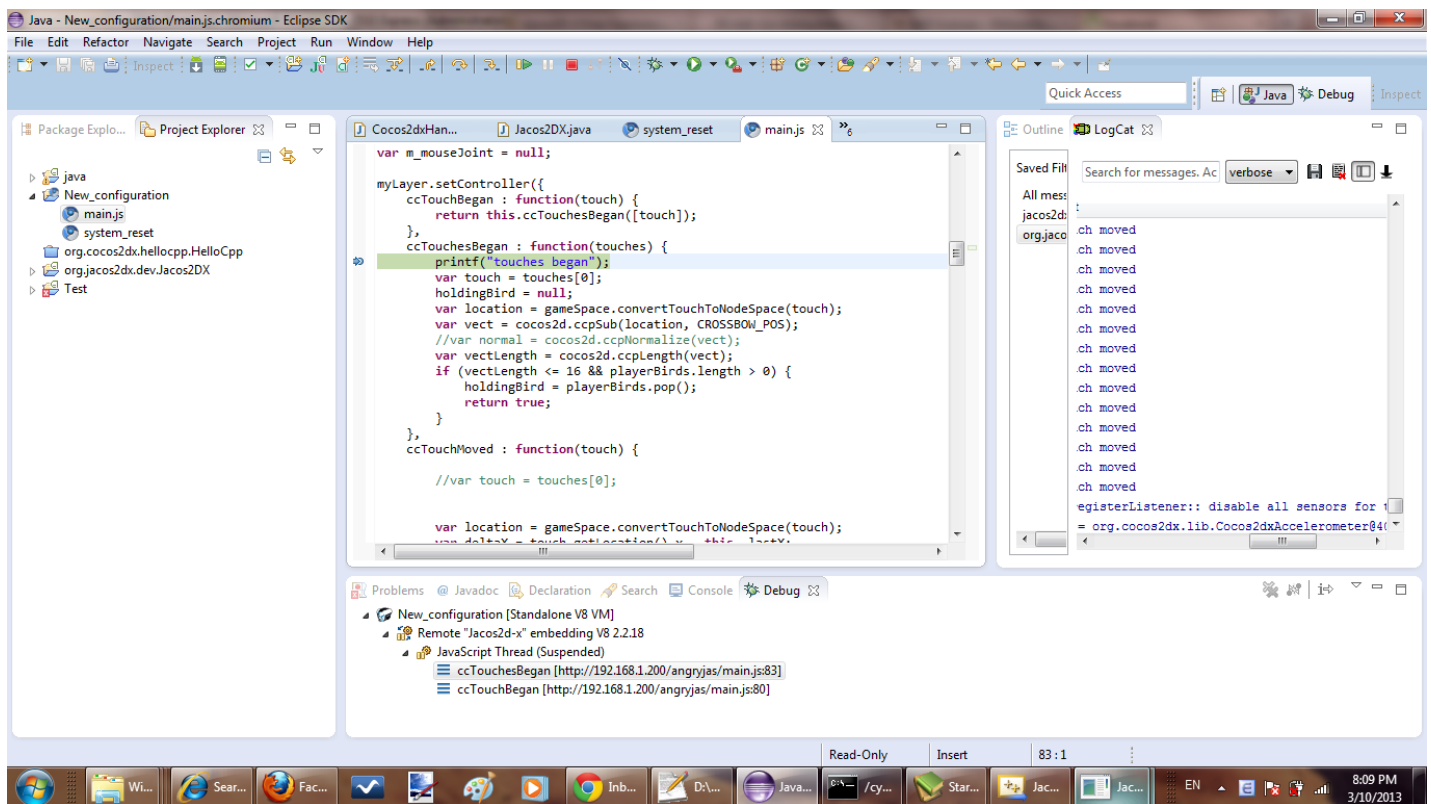


The port you use for debugging is **9223**

Rightnow, please open project by Visual C++. Switch build target to **Remote_Debug**. Build and run Jacos2D-X.

Come back to Eclipse and click **“Debug”** on above dialog. After some seconds, eclipse will connect to V8 debugger successfully

Now, back to Jacos2D-X. Enter your URL which points to application you want to debug and touch Reset. You will see a debug screen like this:



Now, you can use Eclipse's debugging tools

ARRIVED AT THE FIRST STATION

1. Packing the program

Packing Jacos2D-X is extremely simple. You only have to copy all of your project's files (.js resource files, game assets, etc) from www folder to **Resource/jacos2dx_entry** directory. Then build Jacos2D-X again:

- If you want to build for window. Please switch build target to **Distribute** and rebuild again.
- If you want to build for Android. You still have to use Cygwin, but go to **distribute.android** folder and run `./build_native.sh` to make game.so. Now, run Eclipse and import these projects: ***Jacos2d-x/distribute.android*** and ***cocos2dx/platform/android/java***. Compile and run ***Jacos2d-x/distribute.android***
- For iOS, you only switch target to Jacos2DX-Distribute and rebuild it.

2. Global Native functions of Jacos2D-X

No.	Name	Description	Sample
1	printf(string)	Print out a string to the cosole. This function is very usefull for debugging purpose.	printf("Hello World!");
2	setOrientation()	Set the screen orientation to be landscape or portrait.	Set landscape setOrientation(0); Set portrait setOrientation(1);
3	SubClass	Create a new class by inheriting a given class.	See the above example
4	bind	Bind an object to a function. The function is to be called as a member of the binded object.	See the above example
5	Import	Import one or more Class from a library (the default library or library from developers - js).	See the above example
6	Exports	Export one or more Class in a library.	See the above example
7	garbageCollect	Signal the system that a grapage collection is needed to be done. Otherwise, garbage collection will be done automatically when necessary.	garbageCollect();
8	GetJacosInfor	Get current Jacos2D-X information	The information is packed as an object like: <pre>{ version : "0.8.288", author : "Nguyen Thai-Duong", release_date : "03/13/2013", release_at : "Ha Dong district, Hanoi City, Vietnam" };</pre>

CONTINUE...

Because I had no much time to write more, thus I didn't show you many details of Jacos2D-X. Infact, I want to tell you many things about Jacos2D-x. I think every people also be able to develop game. Even you are not an IT engineer, or you are a school pupil☺. I will guide you how to use it in the next times. Thanks for your attention and see you in the next books.

THANKS

- Cocos2D-X community – www.cocos2d-x.org
- Cocosbuilder – www.cocosbuilder.com
- Thu-May Nguyen – artist.
- Duc-Anh Nguyen – supporter.
- Anh-Dung Phan – supporter.
- Dinh-Phuc Nguyen – supporter.
- Son-Thuy Pham – supporter.
- And other friends who help me to develop this Engine.

Hanoi, Spring 2013.