How To Build and Run Cocos2d-x Applications on Samsung Smart TV

Published 2014-10-28 | (Compatible with SDK 4.5,5.0,5.1 and 2013,2014 models)

> This document aims at guiding Cocos2d-x developers to port and deploy their apps on Samsung Smart TV.

Contents

Cocos2d-x is the most popular cross-platform 2d game engine in the world. It is free and open-source and many developers across the globe use it in their games and other rich-graphic applications. This tutorial demonstrates how to port and deploy existing Cocos2d-x applications on Samsung Smart TV by using Portable Native Client (PNaCl) Technology.

Scope

The intended audience of this guide are Cocos2d-x application developers who want to run their Cocos2d-x applications on Samsung Smart TV.

Prerequisites

To build Cocos2d-x and its dependent libraries, NaCl SDK PPAPI v31 (pepper_31) will be required.
Visit **https://developers.google.com/native-client/dev/sdk/download** for instructions on how to download NaCl SDK.
Visit **http://stackoverflow.com/questions/12964666/installing-chrome-native-client-sdk** in case of any issue while following the above link.

To test the application, a Samsung Smart TV or its Emulator running in VirtualBox will be required.

Development Environment

Use a Linux machine to build the Cocos2d-x libraries and applications. Emulator provided along with SDK could be used to debug and test the applications before deploying them onto TV.

Source Code

The build instructions mentioned later in this tutorial are for Cocos2d-x v2.1.5. They may or may not work with other versions. Source code of Cocos2d-x v2.1.5 can be downloaded from **here** .

The following table explains the directory structure of the source code.

Directory	Contents
cocos2dx	This is the main directory which contains the source code of cocos2d-x framework.
CocosDenshion	It is a sound engine based on OpenAL which provides audio support to Cocos2d-x apps.
external	It contains all the 3rd party libraries which are used by Cocos2d-x.
samples	As the name indicates, it contains demo applications for developers' reference. Cpp/HelloCpp is the most basic HelloWorld example and TestCpp contains the usages of all Cocos2d-x classes.

How to Build

Building Cocos2d-x Library

1. Download cocos2d-x v2.1.5 source code from **https://github.com/cocos2d/cocos2d-x/archive/cocos2d-x-2.1.5.zip** and extract it.

   ```
   wget https://github.com/cocos2d/cocos2d-x/archive/cocos2d-x-2.1.5.zip
   unzip cocos2d-x-2.1.5.zip
   ```
2. Download Cocos2d-x PNaCl patch from **here**, unzip and copy it to the cocos2d-x-2.1.5 directory.

   ```
   unzip cocos2dx-pnacl.zip
   cp -r cocos2dx-pnacl cocos2d-x-2.1.5/
   ```
3. Navigate to the cocos2d-x-2.1.5/cocos2dx-pnacl directory.

   ```
   cd cocos2d-x-2.1.5/cocos2dx-pnacl
   ```
4. Apply the Cocos2d-x PNaCl patch to the Cocos2d-x source code by executing the configure-pnacl.sh script. This script configures the Makefiles of Cocos2d-x and other external libraries for cross-compilation with PNaCl toolchain and also creates widgets for Cpp samples that will be deployed on TV as explained later in **Running Cocos2d-x Sample Application** section.

   ```
   source configure-pnacl.sh
   ```
5. Now, go back to the Cocos2d-x root,ie, cocos2d-x-2.1.5 directory.

   ```
   cd ..
   ```
6. Navigate to the cocos2dx/proj.nacl folder.

   ```
   cd cocos2dx/proj.nacl
   ```
7. Export NACL_SDK_ROOT variable with the path to NaCl SDK's pepper API version 31 and prefix PATH variable with path to the aforementioned SDK's PNaCl toolchain.

   ```
   export NACL_SDK_ROOT= folder
   export PATH=$NACL_SDK_ROOT/toolchain/linux_pnacl/bin/:$PATH
   ```
8. Execute **make** command to build the Cocos2d-x library with PNaCl toolchain.

   ```
   make
   ```
   This will build a static cocos2d-x library in Release mode. The name of the lib file is libcocos2d.a and it will be created in NACL_SDK_ROOT/lib/pnacl/Release folder.

   To build the library in DEBUG mode, execute make **DEBUG=1** command. It will create libcocos2d.a in NACL_SDK_ROOT/lib/pnacl/Debug folder.

   ```
   make DEBUG=1
   ```

Building CocosDenshion

CocosDenshion is a sound engine based on OpenAL which provides audio support to Cocos2d-x apps. It is used by many Cocos2d-x applications.

Steps to build it are as follows.

1. Execute steps 2, 3, 4, 5 and 7 of the **Building Cocos2d-x Library** section, if not done already.
2. Navigate to the CocosDenshion/proj.nacl folder present in the Cocos2d-x root, ie, cocos2d-x-2.1.5 directory.

cd CocosDenshion/proj.nacl

3. Execute **make** command to build the CocosDenshion library with PNaCl toolchain.

   make

   This will build a static library in Release mode. The name of the lib file will be libcocosdenshion.a and it will be created in NACL_SDK_ROOT/lib/pnacl/Release folder.

   To build the library in DEBUG mode, execute **make DEBUG=1** command. It will create libcocosdenshion.a in NACL_SDK_ROOT/lib/pnacl/Debug folder.

   make DEBUG=1

Building Box2D

Box2D is a 2D rigid body simulation library for games. Cocos2D-x programmers can use it in their games to make objects move in believable ways and make the game world more interactive.

Steps to build it are as follows.

1. Execute steps 2, 3, 4, 5 and 7 of the **Building Cocos2d-x Library** section, if not done already.
2. Navigate to the external/Box2D/proj.nacl folder present in the Cocos2d-x root, ie, cocos2d-x-2.1.5 directory.

   cd external/Box2D/proj.nacl

3. Execute **make** command to build the Box2D library with PNaCl toolchain.

   make

   This will build a static library in Release mode. The name of the lib file will be libbox2d.a and it will be created in NACL_SDK_ROOT/lib/pnacl/Release folder.

   To build the library in DEBUG mode, execute **make DEBUG=1** command. It will create libbox2d.a in NACL_SDK_ROOT/lib/pnacl/Debug folder.

   make DEBUG=1

Building Chipmunk

Chipmuk, like Box2D, is also a game physics library, which Cocos2D-x programmers to make objects in their games behave like real life objects which can be affected by gravity, collide into other objects, bounce around, etc.

Steps to build it are as follows.

1. Execute steps 2, 3, 4, 5 and 7 of the **Building Cocos2d-x Library** section, if not done already.
2. Navigate to the external/chipmunk/proj.nacl folder present in the Cocos2d-x root, ie, cocos2d-x-2.1.5 directory.

   cd external/Box2D/proj.nacl

3. Execute **make** command to build the Chipmunk library with PNaCl toolchain.

   make

   This will build a static library in Release mode. The name of the lib file will be libchipmunk.a and it will be created in NACL_SDK_ROOT/lib/pnacl/Release folder.

   To build the library in DEBUG mode, execute **make DEBUG=1** command. It will create libchipmunk.a in NACL_SDK_ROOT/lib/pnacl/Debug folder.

   make DEBUG=1

Building Cocos2d-x Sample Application

Now that we've built the libraries, let's move ahead to building applications that use them.

Please note that this tutorial explains building and deploying Cocos2d-x sample applications as PNaCl widgets. If you are unfamiliar with PNaCl widgets, please refer to the guide on **How to create sample PNaCl application**.

The sample applications are present in samples folder. Only the Cpp samples can be built through the procedure mentioned in this tutorial.

The following steps demonstrate how to build the TestCpp sample which contains basic implementations of many Cocos2d-x API's.

1. Go to the samples/Cpp/TestCpp/proj.nacl folder present in the Cocos2d-x root directory.

   cd samples/Cpp/TestCpp/proj.nacl

2. Execute **make** command.

```
make
```

It will build the application in Release mode and create the below mentioned output files in bin/Release folder.

Build the application pexe (TestCpp_pnacl.pexe) with PNaCl toolchain and translate it for x86-32, x86-64 and arm architectures to create TestCpp_x86-32.nexe, TestCpp_x86-64.nexe and TestCpp_arm.nexe respectively.

Create a Native Client manifest file (TestCpp.nmf) that points to these nexe's for corresponding architectures.

To build the application in DEBUG mode, execute **make DEBUG=1** command. It will create the above mentioned output files in bin/Debug folder.

```
make DEBUG=1
```

Running Cocos2d-x Sample Application

In this section, we will learn how to deploy the TestCpp application on TV and SDK.

1. In the proj.nacl folder of every Cpp sample, you can find a widget folder, for eg, TestCppWidget folder in TestCpp/proj.nacl and HelloCppWidget folder in HelloCpp/proj.nacl. This folder contains the Resources (images, fonts, etc) required by the application and the following set of files that constitute a Smart TV widget.

   config.xml

   widget.info

   index.html

   JavaScript files

   For detailed information regarding these files, please visit **this** page.
2. Copy the three nexe's (ie, TestCpp_x86-32.nexe, TestCpp_x86-64.nexe and TestCpp_arm.nexe) and TestCpp.nmf files to the TestCppWidget folder. That's it, your widget is ready. Just copy the TestCppWidget folder to the Apps folder of your emulator and run it.

Related Documents

How to create sample PNaCl application