Toon Boom Game SDK
C++ Backend Documentation

The C++ Backend for the Toon Boom Game SDK is a portable data structure that is meant to be integrated with third party game engines. There are currently two implementations available. The first one is an integration with Cocos2dx that creates sprites compatible with the engine and animate them, while letting the engine handle the rendering. The second one integrates with Unity. In this integration, the backend handles the sprite rendering itself on all supported platforms.

CoreCpp

CoreCpp is the common library used in the Toon Boom Game SDK. This library handles the parsing of the XML data structure and also parsing and writing of the binary data structure.

The package contains:

- CoreCpp/Base - Common classes.
- CoreCpp/Bin - Binary data structure IO.
- CoreCpp/Render - Rendering data structure.
- CoreCpp/Support - Third party classes.
- CoreCpp/Tree - Tree data structure.
- CoreCpp/TreeView - Tree data structure visualizer.
- CoreCpp/Xml - XML data structure IO.

The common library in itself is not sufficient to render sprites. It is meant to be used as a basis for any C++ implementation. The following example demonstrates how to load an existing project data structure from file and how to retrieve its associated sprite sheets.

```
// The clip data contains a collection of renderable hierarchies and
// is the main data structure used to describe a single scene.
RD_ClipDataCore *clipCore = new RD_ClipDataCore;

// Load clip data.  First ask binary parser to load the file and fallback
// to XML parser if file does not exist
if (!BIN_ProjectLoader::loadStageClip( folder, clipName, clipCore ))
  XML_ProjectLoader::loadStageClip( folder, clipName, clipCore );

// Iterate on node tree hierarchies.  Every node tree is a separate
// top level hierarchy in Harmony and has its own sprite sheet.
unsigned count = clipCore->count();
for ( unsigned idx=0 ; idx<count ; ++idx )
{
  TV_NodeTreeViewPtr_t nodeTreeView = clipCore->nodeTreeView( idx );
```

```
  STD_String sheetName;
  STD_String resName = “ipad”;
  if (nodeTreeView->spriteSheet(sheetName))
  {
    RD_SpriteSheetCore *spriteSheetCore = new RD_SpriteSheetCore;

    // Load the sprite sheet mapping.  Since this can be handled by the game
    // engine itself, the image is not loaded in this data structure.
    if (!BIN_ProjectLoader::loadSpriteSheet( folder, sheetName, resName, spriteSheetCore ))
      XML_ProjectLoader::loadSpriteSheet( folder, sheetName, resName, spriteSheetCore );

    // The sprite sheet should be stored in a shared container as it will be
    // reused through the different clip data objects.
    (...)
  }
}

(...)
```

Sample uses of CoreCpp data structures

Cocos2dx Backend

The current Cocos2dx version used in this backend is version 2.1.4.

The package contains:

- Cocos2dx/Core - Common sources for Cocos2dx projects.
- Cocos2dx/Apps/Previewer - Game Previewer sources.
  - Cocos2dx/Apps/Previewer/proj.mac - XCode project for Mac OSX.
  - Cocos2dx/Apps/Previewer/proj.ios - XCode project for iOS.
- Cocos2dx/Apps/SampleGame - Sample Game sources.
  - Cocos2dx/Apps/SampleGame/proj.mac - XCode project for Mac OSX.
  - Cocos2dx/Apps/SampleGame/proj.ios - XCode project for iOS.
- Cocos2dx/cocos2dx - Cocos2dx 2.1.4 precompiled headers and libraries.
- Cocos2dx/neon - Neon (WebDAV client) precompiled headers and libraries.
- Cocos2dx/openssl - OpenSSL precompiled headers and libraries.

* Cocos2dx backend does not have support for newer features implemented in the Toon Boom Game SDK. As such, it will not completely support scenes with deformations, cutters or metadata.

```
// Load clip data
if( !RD_ClipDataManager::instance()->isExist(clipName) )
{
  RD_ClipDataCore *clipCore = new RD_ClipDataCore();

  if ( BIN_ProjectLoader::loadStageClip( folder, clipName, clipCore ) == false )
    XML_ProjectLoader::loadStageClip( folder, clipName, clipCore );
```

```
  RD_ClipData *clipData = new RD_ClipData( clipCore );
  RD_ClipDataManager::instance()->addClipData( clipName, clipData );
}

// Create a Cocos2dx renderable object using specified clip data.
RD_Clip *clip = RD_Clip::create( RD_ClipDataManager::instance()->getClipData(clipName) );

// Load sprite sheets associated with this clip.
clip->build( folder );
// Loop clip at 24 frames per second.
clip->loopAnimation( 24.0f );
```

Sample creation of a Cocos2dx renderable layer

Unity Backend

The Unity backend will work with Unity version 4.5.4 or higher. A C# scripting interface has also been implemented for Unity to properly communicate with its C++ plugin.

The plugin is located in the folder “Unity/RendererPlugin”. To compile the plugin, you can use the qmake tool to generate a platform specific Makefile. qmake is included with Qt, which may be downloaded at http://download.qt.io. Any version of Qt 4 should provide a qmake tool compatible with the current package.

Alternatively, solutions for multiple platforms are also available:

- Unity/RendererPlugin/proj.mac - XCode project for Mac OSX.
- Unity/RendererPlugin/proj.ios - XCode project for iOS.
- Unity/RendererPlugin/proj.win32 - Visual Studio 2010 solution for Windows.
- Unity/RendererPlugin/jni - Compilation scripts for Android.

The sources for the plugin are organized as follows:

- HarmonyRenderer - Plugin entry point.
- Base - Common classes.
- CoreCpp - Reference to CoreCpp.
- Image - Image data structure and io.
- Render - Rendering data structures and platform specific rendering code.
- Shaders - Vertex and fragment shaders.
- Support - Third party libraries.

Xml2Bin

The Xml2Bin utility converts the XML data structure generated through the Toon Boom Harmony software to a compressed binary data structure.

The package contains:

- Utills/macosx - Precompiled binary for Mac OSX.
- Utils/win32 - Precompiled binary for Windows.
- Utils/Xml2Bin - Xml2Bin sources.
  - Utils/Xml2Bin/proj.mac/Xml2Bin.xcodeproj - XCode project for Mac OSX.
  - Utils/Xml2Bin/proj.win32/Xml2Bin.sln - Visual Studio 2010 solution for Windows.