Super Animation Converter Help Doc

Intro

Super Animation Converter is a simple SWF file(a file format supported by Adobe Flash Player software) converter. It can parse and extract transformation data from SWF file. With these transformation data and the right images, you can reproduce the animation in any game engine just as Adobe Flash Player can do.

The native animation solution in Cocos2d game engine is “frame by frame” animation, which means if there are 20 frames in your animation, you need 20 pieces of image(texture), and draw these images one by one on the screen to create animation effect. There are two flaws in this solution:

1. High resource consuming, one frame one image;
2. Hard to make smooth animation, since no interpolation between frames, which means it “jumps” directly from the current frame to the next frame.

So I try to provide a better animation solution which is based on Super Animation Converter for Cocos2d game engine. The idea is very simple:

1. Use Flash as your animation editor, create any animation you want in it, then export these animation as SWF file.
2. Parse the SWF file with Super Animation Converter, get the transformation data and images from the SWF file.
3. Reproduce the animation in Cocos2d game engine with the transformation data and images.

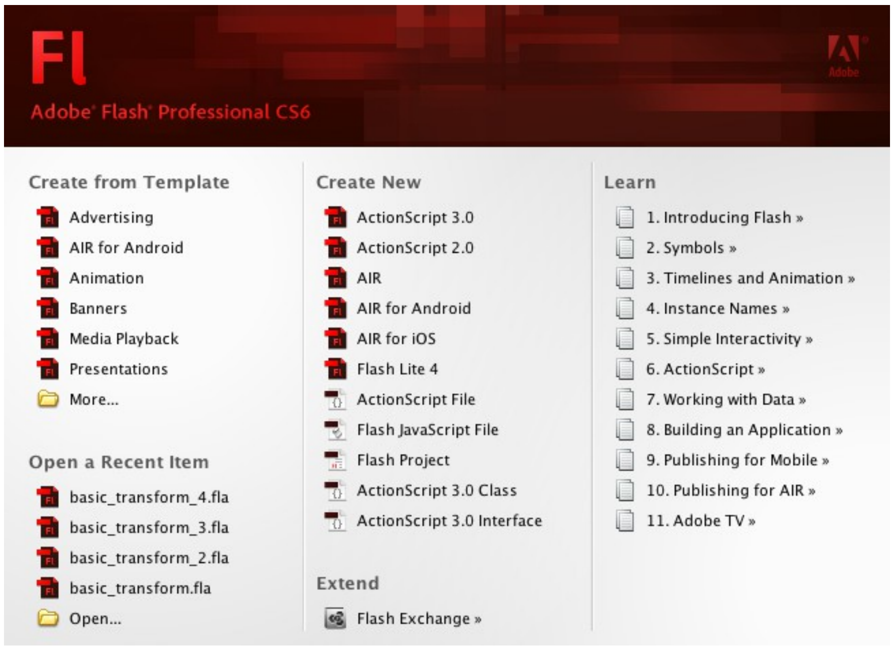
I will introduce rules & steps you need to follow when creating animation in Flash, and source code files need to be imported to the existed Cocos2d game engine in the next chapter.

Rules & Steps in Flash

I assume that you have the basic knowledge about Flash, you should know some key concept, like Stage, Layer, Key Frame, Symbol and Classic Tween, etc.

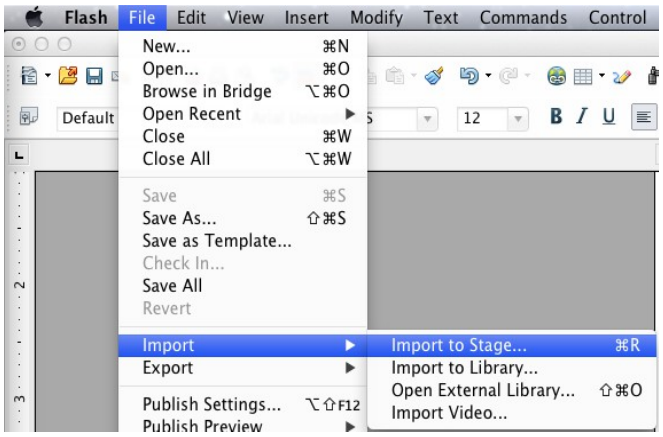
Before you start to create you animation, one key rule you should know: Super Animation Converter bases on **Graphic Symbol** in Flash, which means **Graphic Symbol** is the only symbol you can place on the stage. **Button** & **Movie Clip**, any V**ector Graphics** are NOT supported by Super Animation Converter.

Open the Flash, my version is CS6. Since we only use the basic function in Flash, you can use any old version you have. Create new ActionScript 3.0 file, save it as "basic_transform.fla".

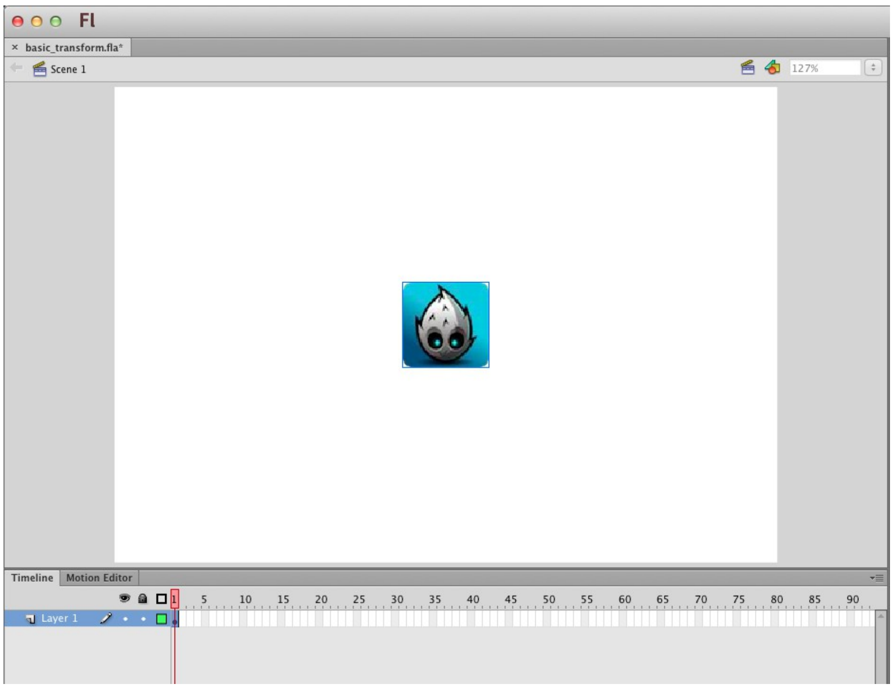


Let's import the little blue logo of Cocos2d-x into the editor, select "File/Import/Import to Stage".

Please remember that we only support PNG image right now.



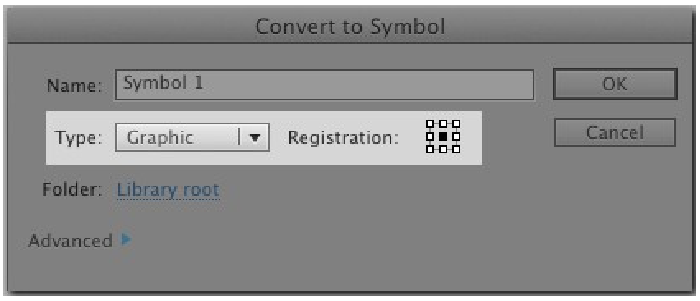
Now you can see the little blue logo in the stage.



Remember, we only support **Graphic Symbol**. So you need to convert the little blue logo to graphic symbol firstly. Select the logo in the stage, right mouse button, select "Convert to Symbol..." on popup menu.

In the "Convert to Symbol" dialog box, please make sure:

1. Type must be Graphic;
2. Registration must be center. I always assume that the origin of symbol is located at the center. If the transformation of your animation is wrong, maybe because the registration is not center.



Convert to Symbol

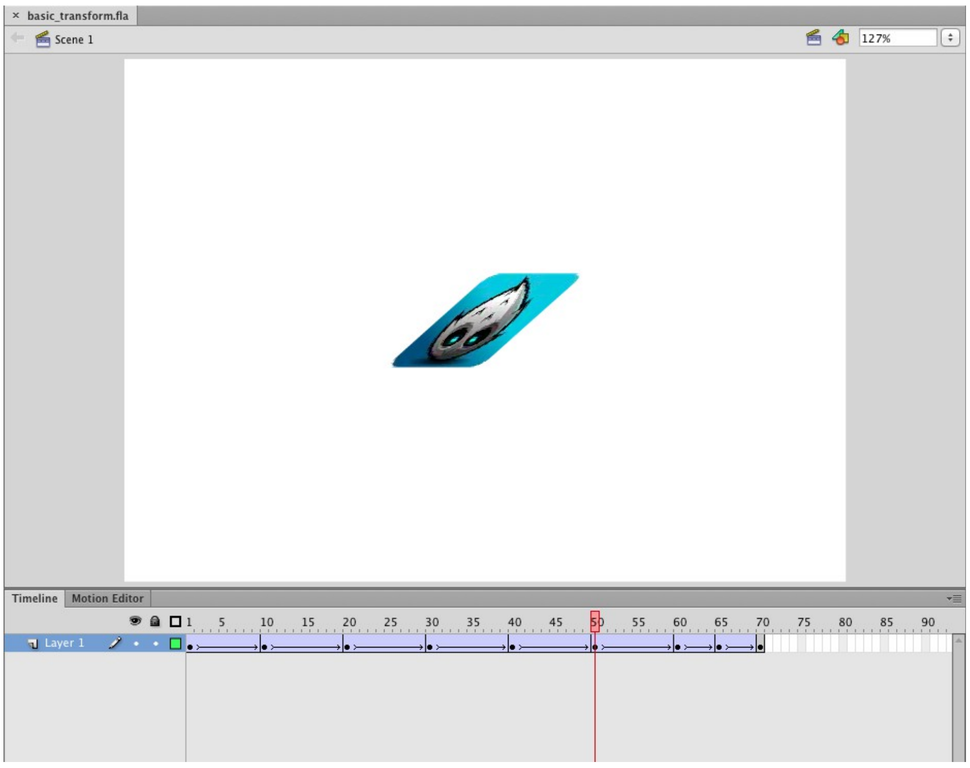
Name: Symbol 1

Type: Graphic

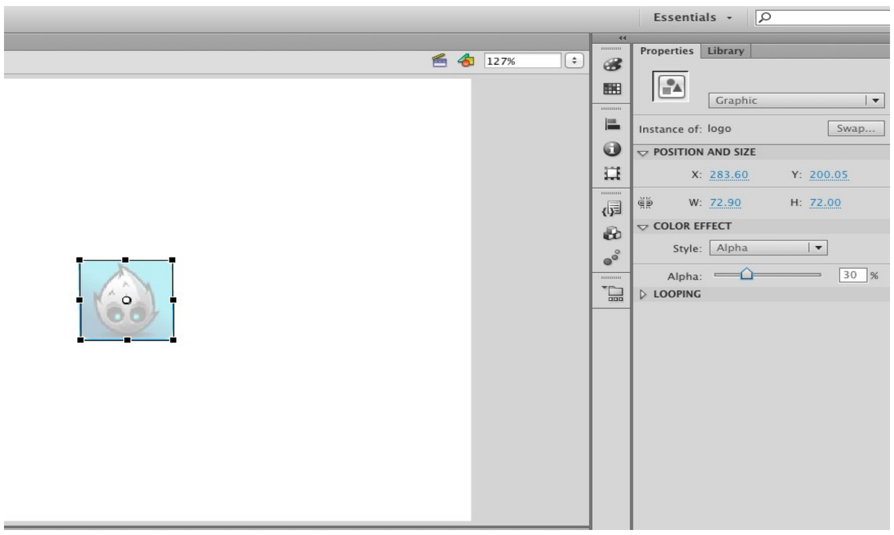
Registration:

Folder: Library root

Once you have converted the image into Graphic Symbol, you can apply any animation effect on it. Let's add some basic transformation to the logo which includes scale up & down, skew, rotation and translation.



Super Animation Converter also support Alpha effect. So let's add fade out/in effect for the little blu logo.



Instance of: logo

POSITION AND SIZE

X: 283.60 Y: 200.05

W: 72.90 H: 72.00

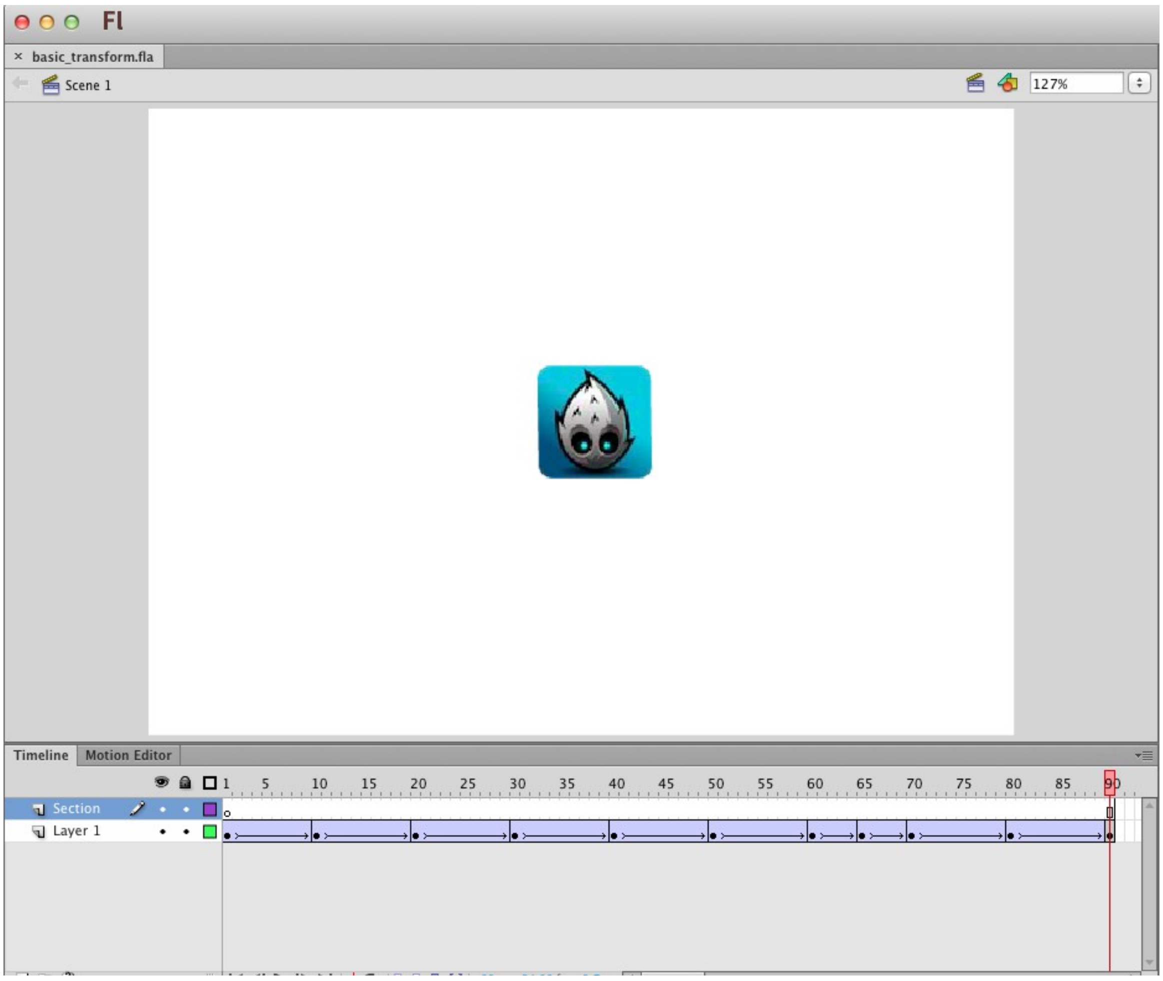
COLOR EFFECT

Style: Alpha

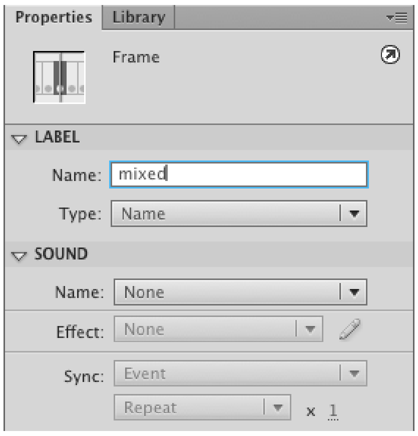
Alpha: 30 %

OK, our animation seems ready now. But let's assume a scenario: a character may have multiple actions, like idle action, battle action. We want to play "battle" action when character is battling, otherwise play "idle" action. Usually, artist like to create all these actions in one animation file. So in order to separate animation, Super Animation Converter introduces "**Animation Section**".

Create a new layer, rename it as "Section", and place it on the top. Super Animation Converter use this layer to separate animation.



"Section" layer should not include any drawable item. We use "**Label**" property of "**Blank Keyframe**" to specify the start of an animation section. Since we already have one blank keyframe at the 1st frame of "Section" layer, just select it, name it as "mixed".



LABEL

Name: mixed

Type: Name

SOUND

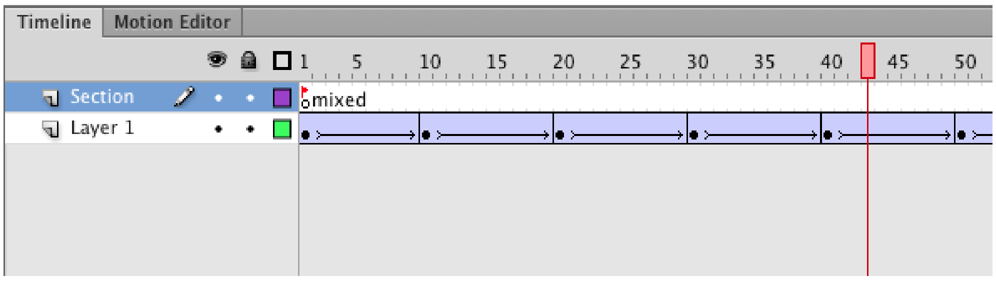
Name: None

Effect: None

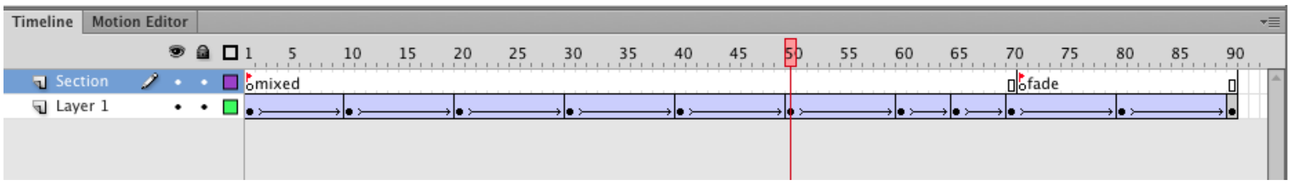
Sync: Event

Repeat x 1

Enter, you will see that there is a little red flag and a name "mixed" on the 1st frame, which means you have added the section label successfully.

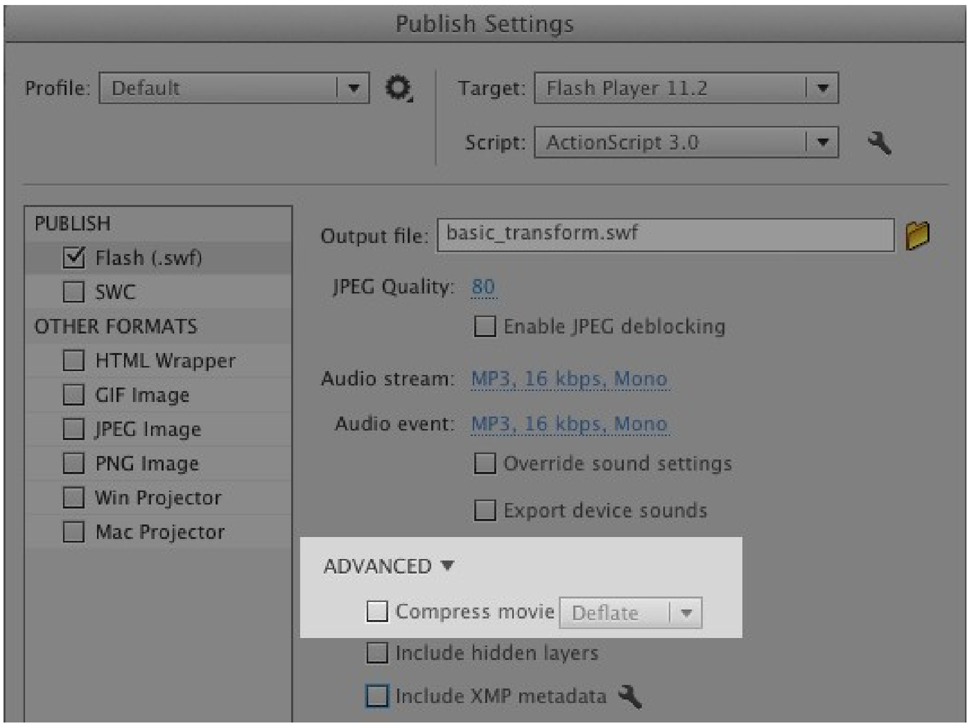


Let's separate the second section. Assume that the second section begins at the 71st frame, select the 71st frame on the "Section" layer, right mouse button, the select "Insert Blank Keyframe" on the popup menu. Then edit the label name of the new added blank keyframe, name it as "fade".



Now we've separated the animation into two sections. The first section named "mixed", which begins at the 1st frame, ends at the previous frame of "fade" label. The second section named "fade", which begins at the frame of "fade" label, ends at the tail of the whole animation.

Ok, everything is done, it's time to publish our animation as SWF file. Select "File/Publish Settings", make sure on thing in the "Publish Settings" dialog box: "Compress movie" is NOT selected in "ADVANCED". Then click "Publish" button, a new file named "basic_transform.swf" will be created in the folder of "basic_transform.fla".



Publish Settings

Profile: Default

Target: Flash Player 11.2

Script: ActionScript 3.0

PUBLISH
- [✓] Flash (.swf)
- [ ] SWC

OTHER FORMATS
- [ ] HTML Wrapper
- [ ] GIF Image
- [ ] JPEG Image
- [ ] PNG Image
- [ ] Win Projector
- [ ] Mac Projector

Output file: basic_transform.swf

JPEG Quality: 80

- [ ] Enable JPEG deblocking

Audio stream: MP3, 16 kbps, Mono

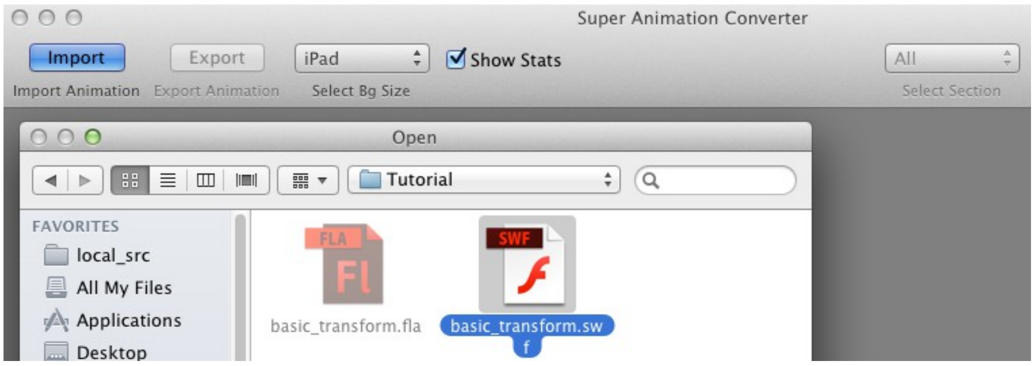
Audio event: MP3, 16 kbps, Mono

- [ ] Override sound settings
- [ ] Export device sounds

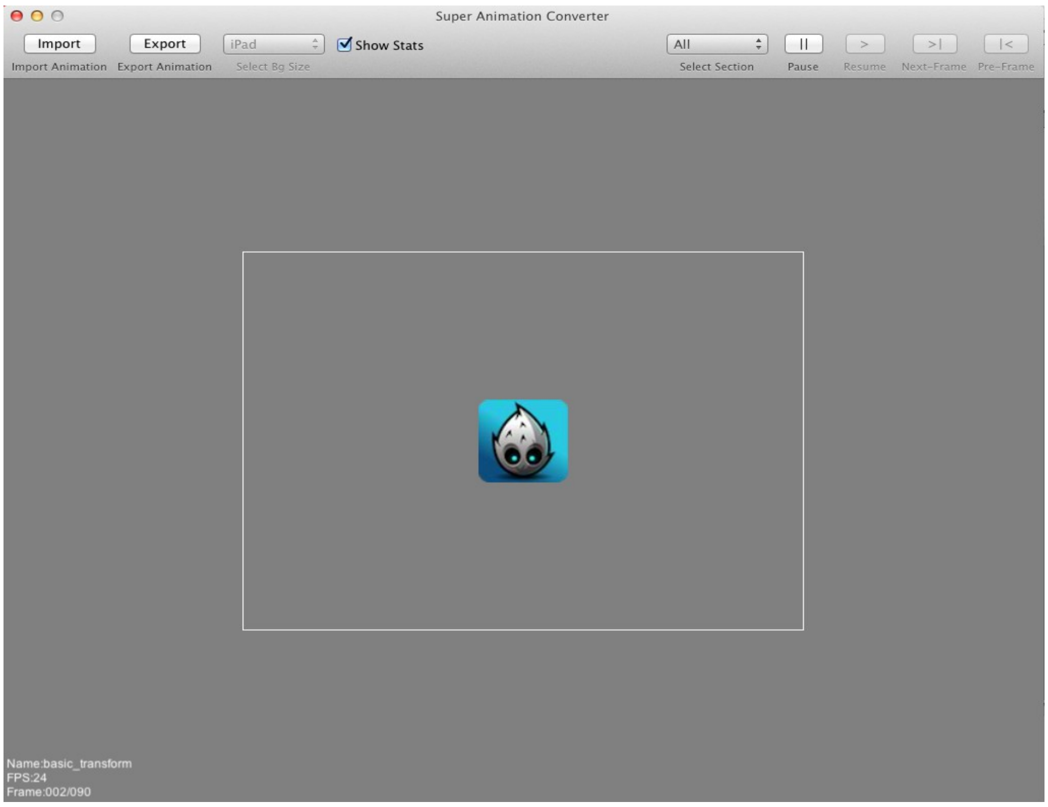
ADVANCED ▼

- [ ] Compress movie Deflate
- [ ] Include hidden layers
- [ ] Include XMP metadata

Run Super Animation Converter, click “Import”, import “basic_transform.swf”.

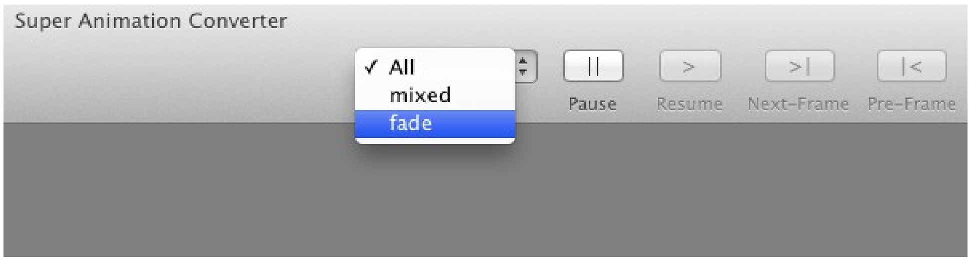


If everything goes well, you can see the animation start to run.



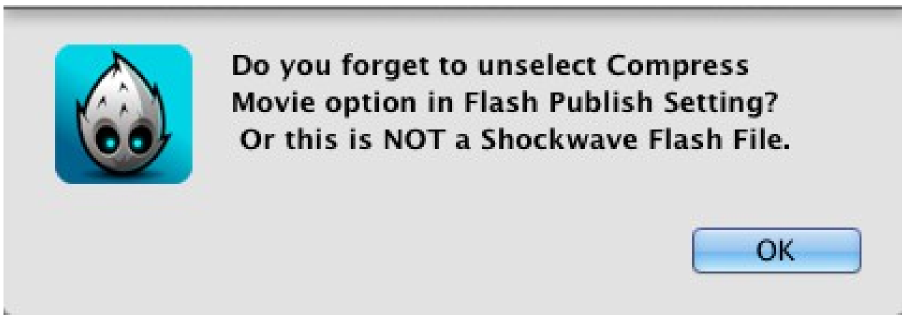
Name:basic_transform
FPS:24
Frame:002/090

Let's have a quick look at the Super Animation Converter. On the left bottom corner, there are some stats data, includes animation file name, FPS and the current frame number, etc. On the right top corner, there are some operation buttons and a pop-up list named “Select Section”. Click the “Select Section” pop-up list, you can see all the animation sections. Super Animation Converter play the whole animation repeatedly by default. You can select any animation section to play in the pop-up list.



I think you will also notice that there is a white frame on the grey background. This white frame stands for the stage dimension in the Flash. And I will use it as the content size of "SuperAnimNode" in the cocos2d game engine.

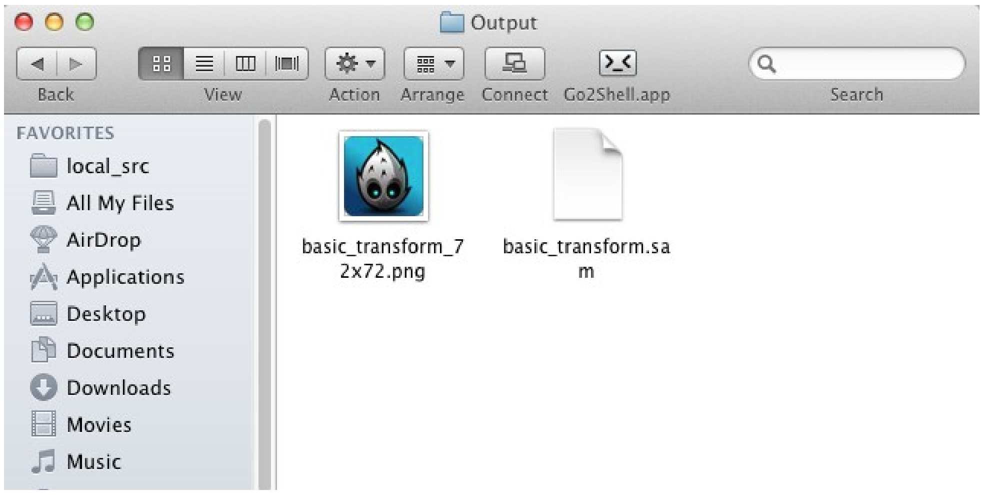
If something goes wrong after you click "Import" button, Super Animation Converter will you, like



Do you forget to unselect Compress Movie option in Flash Publish Setting?
Or this is NOT a Shockwave Flash File.

Which indicates that you didn't unselect "Compress movie" in the "Publish Settings".

Ok, if everything goes OK, it's time to export the animation. Just click the "Export" button, then select a folder.



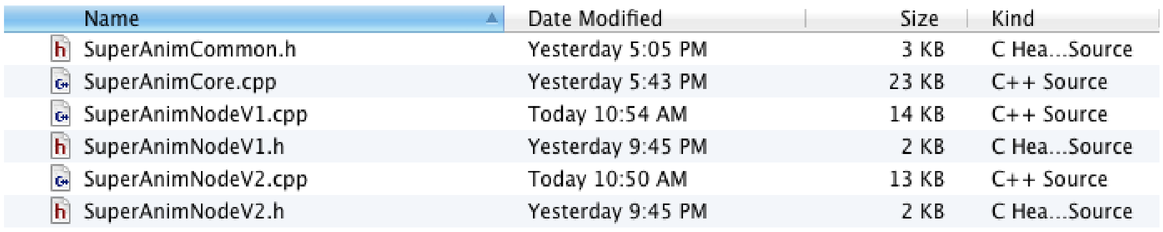
As you see in the previous screenshot, we only have one image in the output since only one image is used in the animation. Besides, we have a **.sam**(short for Super Animation) file in the output, which includes transformation data of each frame in the animation. These two files are all we need in Cocos2d game engine. You may say that the animation is too simple. Yes, but you can do anything you want in Flash as long as you follow the rules I mentioned. You will see a complicate animation in my sample code, a fish, which can spit out some bubbles:-).

Implementation in Cocos2d-x engine

You can get my code & resources from:

1. github website, the link is https://github.com/raymondlu/super-animation-samples , where I will maintain my code.
2. src_media.zip file in the Resource folder of Super Animation Converter app package, which includes all the source files you need to add into the existed Cocos2d-x game engine, and resources needed in the sample I mentioned. Since I cannot update these source code & resources as I want, the first way is recommended.

We need to import 4 files into Cocos2d-x engine. Open the src folder, you will files below:



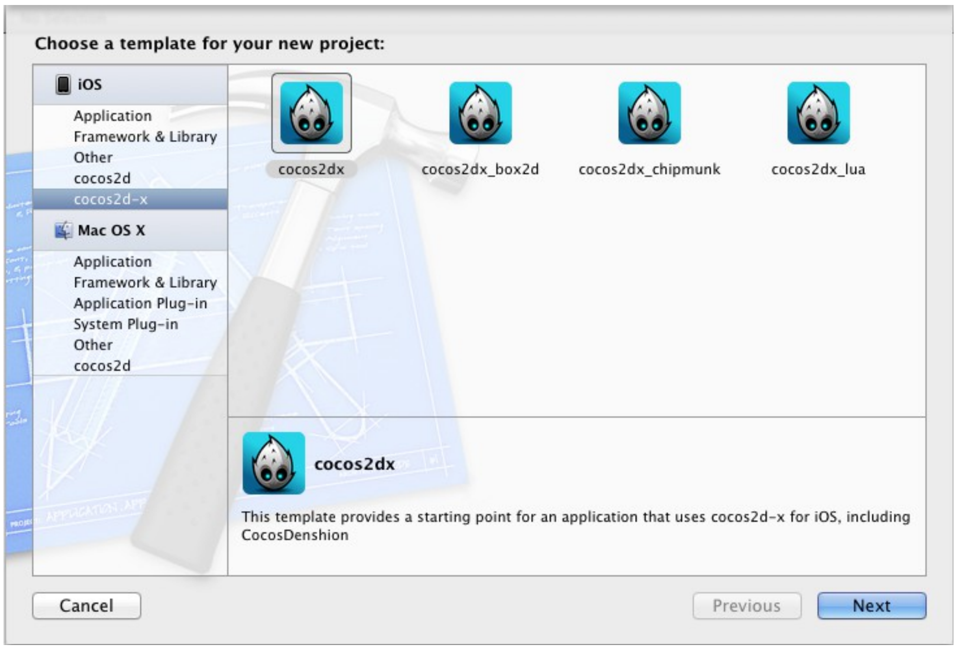
Name	Date Modified	Size	Kind
SuperAnimCommon.h	Yesterday 5:05 PM	3 KB	C Hea...Source
SuperAnimCore.cpp	Yesterday 5:43 PM	23 KB	C++ Source
SuperAnimNodeV1.cpp	Today 10:54 AM	14 KB	C++ Source
SuperAnimNodeV1.h	Yesterday 9:45 PM	2 KB	C Hea...Source
SuperAnimNodeV2.cpp	Today 10:50 AM	13 KB	C++ Source
SuperAnimNodeV2.h	Yesterday 9:45 PM	2 KB	C Hea...Source

SuperAnimCommon.h: which includes all data structure definition used in animation, like frame, matrix, and functions used when playing animation. (You don't have to know about these stuff.)

SuperAnimCore.cpp: which implements some core functions, like **.sam** file parser. (You don't have to know about these stuff.)

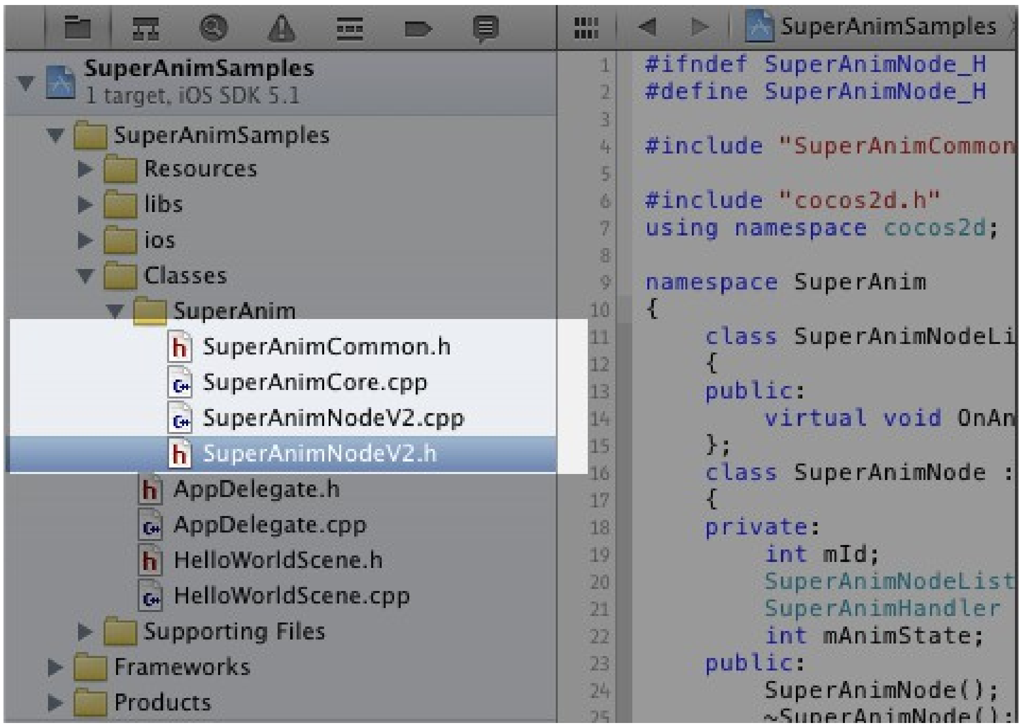
SuperAnimNodeV*.h(cpp): which implement a class named **SuperAnimNode**. **SuperAnimNode** is the only class you need to know. You will use this class to control the behavior of the animation. Since there are lots of changes between Version 1.0 and Version 2.0 of Cocos2d-x, I implement two versions of **SuperAnimatonNode**. If your Coscos2d-x is lower than Version 2.0, please use SuperAnimNodeV1.h(cpp), otherwise, use SuperAnimNodeV2.h(cpp).

Let's create a Cocos2d-x project. My Cocos2d-x is Version 2.0.



This template provides a starting point for an application that uses cocos2d-x for iOS, including CocosDenshion

Add **SuperAnimCommon.h**, **SuperAnimCore.cpp** and **SuperAnimNodeV2.h(cpp)**, these 4 files into the project(Remember my engine is Version 2.0).



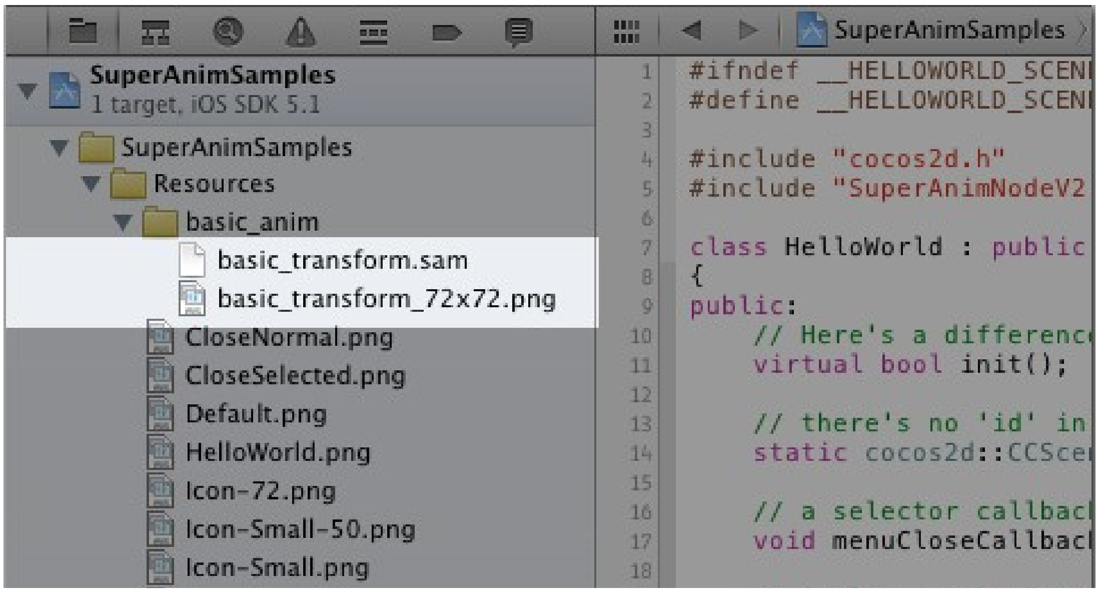
```
#ifndef SuperAnimNode_H
#define SuperAnimNode_H

#include "SuperAnimCommon

#include "cocos2d.h"
using namespace cocos2d;

namespace SuperAnim
{
    class SuperAnimNodeLi
    {
    public:
        virtual void OnAn
    };
    class SuperAnimNode :
    {
    private:
        int mId;
        SuperAnimNodeList
        SuperAnimHandler
        int mAnimState;
    public:
        SuperAnimNode();
        ~SuperAnimNode();
```

Add Super Animation Converter exported files into Resources group.



SuperAnimSamples

```
SuperAnimSamples
1 target, iOS SDK 5.1
  SuperAnimSamples
    Resources
      basic_anim
        basic_transform.sam
        basic_transform_72x72.png
      CloseNormal.png
      CloseSelected.png
      Default.png
      HelloWorld.png
      Icon-72.png
      Icon-Small-50.png
      Icon-Small.png
```

```
#ifndef __HELLOWORLD_SCEN
#define __HELLOWORLD_SCEN

#include "cocos2d.h"
#include "SuperAnimNodeV2

class HelloWorld : public
{
public:
    // Here's a differenc
    virtual bool init();

    // there's no 'id' in
    static cocos2d::CCSce

    // a selector callbac
    void menuCloseCallbac
```

OK, since code & resource are all ready, let't start to program. First, let's edit the HelloWorld.h.

```
#include "cocos2d.h"

#include "SuperAnimNodeV2.h"

class HelloWorld : public cocos2d::CCLayer, public SuperAnim::SuperAnimNodeListener
{
public:
   virtual bool init();
   static cocos2d::CCScene* scene();
   LAYER_CREATE_FUNC(HelloWorld);
    void OnAnimSectionEnd(int theId, std::string theLabelName);
private:
        SuperAnim::SuperAnimNode* mAnimNode;
};
```

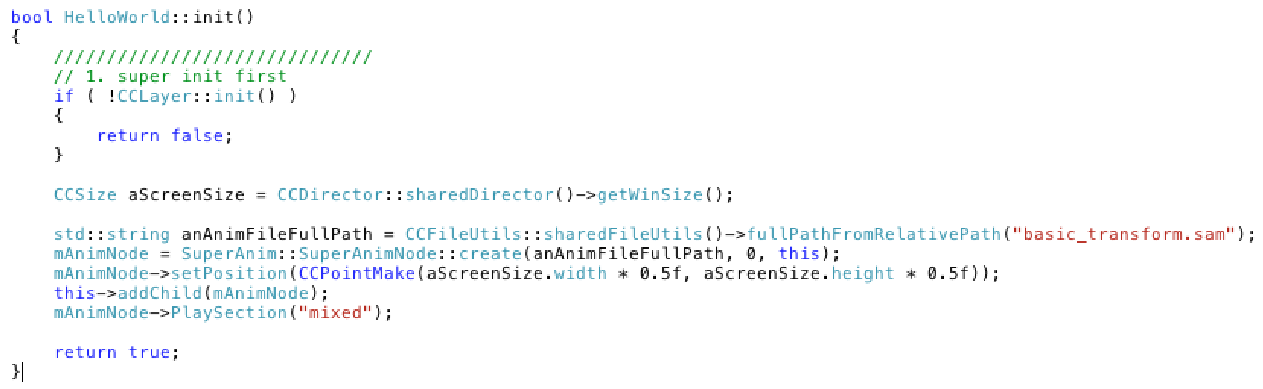
I want to make the HelloWorld become the listener of **SuperAnimNode**, so let HelloWorld inherit from **SuperAnimNodeListener:**

```
 class SuperAnimNodeListener
{
public:
        virtual void OnAnimSectionEnd(int theId, std::string theLabelName){}
};
```

When **SuperAnimNode** reach the end of a section, it will call this method to inform its listener.

I also add a SuperAnimNode pointer member for easy reference to the instance of SuperAnimNode.

Turn to HelloWorld.cpp, let's edit the init method.



```
bool HelloWorld::init()
{
    //////////////////////////////
    // 1. super init first
    if ( !CCLayer::init() )
    {
        return false;
    }

    CCSize aScreenSize = CCDirector::sharedDirector()->getWinSize();

    std::string anAnimFileFullPath = CCFileUtils::sharedFileUtils()->fullPathFromRelativePath("basic_transform.sam");
    mAnimNode = SuperAnim::SuperAnimNode::create(anAnimFileFullPath, 0, this);
    mAnimNode->setPosition(CCPointMake(aScreenSize.width * 0.5f, aScreenSize.height * 0.5f));
    this->addChild(mAnimNode);
    mAnimNode->PlaySection("mixed");

    return true;
}|
```

Before create SuperAnimNode instance, we need to get the full path of animation data file(.sam file).

```
mAnimNode = SuperAnim::SuperAnimNode::create(anAnimFileFullPath, 0, this);
```

I set the "id" parameter as 0, you can set other values. If you have multiple **SuperAnimNode** instances, this "id" value is the only identifier. For "listener" of **SuperAnimNode**, I set it to "this" object, our HelloWorld object, then add the **SuperAnimNode** instance as a child of **HelloWorld** .

I want to play the "mixed" section, so call **PlaySection** method, and set the parameter to "mixed". Ok, you can try to run the program now.

```
mAnimNode->PlaySection("mixed");
```

Wait for a second, what to do after finishing the "mixed" animation section? You need to tell **SuperAnimNode**. So I override the OnAnimSectionEnd method as:

```
void HelloWorld::OnAnimSectionEnd(int theId, std::string theLabelName){
        if (theId == 0) {
          if (theLabelName == "mixed") {
                    // goto fade section
                    mAnimNode->PlaySection("fade");
          } else if (theLabelName == "fade"){
                    // goto mixed section
                    mAnimNode->PlaySection("mixed");
          }
        }
}
```

Which means if finishing the "mixed" section, go on to play "fade" section, and vice versa. You can do anything you want in this method. Then you will see that our little blue logo runs repeatedly:

Now, you have the basic knowledge about the new animation solution for Cocos2d-x engine. Based on this idea, you can do any animation you want in Cocos2d-x. If you want to learn more, please refer to my sample code, and feel free to contact me, if you have any issue, my e-mail address: Raymond-Lu@live.com.