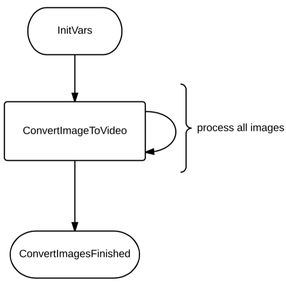
Version	Change	Date
1.5	Support importing audio file	2015-06-24

Images2Video is an assistant library for iOS and Android. This library can help to convert serial images into a mp4 format video.

Now you can attach the audio file with the video file!!

We wrapper these plugins by *VideoConverter.cs* and publish three functions, *InitVars*, *ConvertImageToVideo* and *ConvertImagesFinished*.

The following flowchart is how the script work.



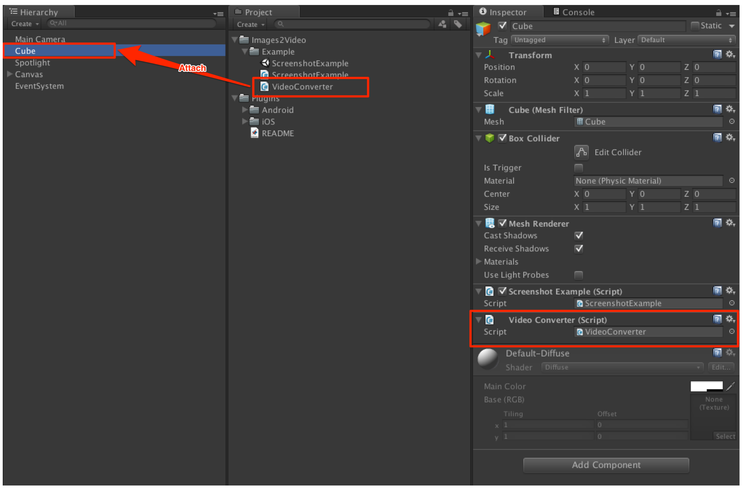
1. You need to call *InitVars*, which define the width and height of the images, fps, callback function name, audio path, shortest clip boolean value, and attached game object name.
2. Then you need to call *ConvertImageToVideo* to pass the image data and frame indicator for the loop.
3. After pass all image data, you need to call *ConvertImagesFinished* and wait for callback function being triggered.

First of all, we need the serial of images to generate the video file.

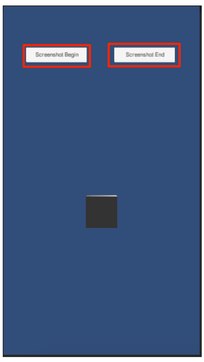
In our example, we create the cube rotating animation, and two buttons to trigger the cube start and stop rotating.
While rotating, take a screenshot in *FixedUpdate* function.
When pressing Screenshot End button, start converting all screenshots to the video file and save to the camera roll.

The following tutorial will show you how to use this script step by step.

Step1. Create the scene, *ScreenshotExample.unity*. add the script for controlling the Attached this script, *VideoConverter.cs* to the game object. In our example, we attach the script to Cube game object.



Step2. Create two buttons for start and stop rotating animation. Set up their click event functions for ScreenshotBegin and ScreenshotEnd which are in the ScreenshotExample.cs.



Step3. Follow the above flowchart, when Screenshot Begin button is clicked, we call *InitVars* function to initialise required parameters.

VideoConverter.cs snippet

```
/// <summary>
/// Inits the converter variables.
/// </summary>
/// <param name="gameObjName">The attached game object name.</param>
/// <param name="callbackMethod">The callback method which will be called by UnitySendMessage.</param>
/// <param name="audioPath">The audio path which you wnat to attach, if you don't want to attach the audio file, set as empty string</param>
/// <param name="width">The video width.</param>
/// <param name="height">The video height.</param>
/// <param name="frameRate">The video frame rate.</param>
/// <param name="shortestClip">Enable the shortest clip while generating the video file.</param>
public void InitVars(String gameObjName, String callbackMethod, String audioPath, int width, int height, int frameRate, bool shortestClip)
```

ScreenshotExample.cs snippet

```
if (gameObject == null)
    _videoConverter.InitVars("default", "ConvertEnd", path, _imageWidth, _imageHeight, _frameRate, true);
else
    _videoConverter.InitVars(gameObject.name, "ConvertEnd", path, _imageWidth, _imageHeight, _frameRate, true);

Time.captureFramerate = _frameRate;
_keepGoing = true;
```

Step4. While the cube is rotating, we call *Application.CaptureScreenshot* function in FixedUpdate() to capture the screen.

```
// Update is called once per frame
void FixedUpdate ()
{
    if(!_keepGoing)
        return;

    RotateCube();

    string _name = String.Format("{0}/{1:D04}_shot.png", _folder, Time.frameCount);
    Debug.Log("screenshot file path : " + _name);
    Application.CaptureScreenshot(_name, 1);
}
```

Step5. Then when Screenshot End button is clicked, we begin to pass the image data byte array into *ConvertImageToVideo*. After passing all images data, call *ConvertImagesFinished* to begin converting the video.

```
public void ScreenshotEnd() {
    if (Application.platform != RuntimePlatform.OSXEditor) {
        ResetParameters();

        StopRotate();

        _videoConverter.DisplayProgress("Converting", "Processing");

        /* Prepare for convert processing */
        DirectoryInfo _info = new DirectoryInfo(_path);
        FileInfo[] _files = _info.GetFiles("*.*");
        int _indicator = 0 ;
        foreach (FileInfo _file in _files) {
            byte[] _bytes = File.ReadAllBytes(_path + "/" + _file);
            _videoConverter.ConvertImageToVideo(_bytes, _indicator);

            _indicator++;
        }

        _videoConverter.ConvertImagesFinished();
    }
}
```

Step6. After finishing, the callback function, *ConvertEnd*, will be called, the video file also will be saved into camera roll.

If you have any questions, please send to <unity@starktech.com.tw>.

2014 Stark Info Co., Ltd.