Flash to Unity

Standards and good practices

Monday, October 7th, 2013

Abstract:

The following document summarizes a series of standards and good practices that should be followed when working with Flash to Unity. It’s heavily recommended, yet not mandatory, for the user to get accustomed to the aforementioned standards and always practice caution when deviating from the norm.

Table of Contents

Introduction

Flash to Unity is a tool that allows game developers to import animations made in Flash to the Unity 3D Game Engine. When coding with Flash to Unity, it's advisable to follow a simple series of standards and good practices in order to improve the performance and usability of the tool. The rest of the document will detail a list of suggestions the user is encouraged to follow when using Flash to Unity.

Standards and Good Practices

In order to ensure Flash to Unity's correct execution, the developers recommend following the next guidelines:

1. If your Flash project is relatively big, you may want to separate it into several sub-projects to be imported separately. This will not only serve to divide workloads and allow for multiple people to work in the same project, but will also prevent memory leaks on Flash's side.
2. For optimal performance, the camera should be set at point (0, 0, -10), with a size of 1. It's recommended to create a camera directly from the F2U Menu.
3. While some naming conventions are not actually enforced or detected by Flash to Unity, it would be wise to follow them.
4. Don't use ambiguous or misleading labels on your projects. While Flash to Unity is smart enough to associate labels with their corresponding frames and timelines, most humans may have troubles differentiating two distinct labels with very similar names. In addition, try not to add labels outside the "labels" layer as they will, under no circumstances, be exported into Unity.
5. Convert any Flash effect into key frames whenever applicable before importing your animation into Unity. Flash to Unity supports some well-known effects like classic tweening, yet not every single effect is supported; the only way to make sure your desired effect will be supported is to convert it into key frames.

6. Use PNG images on your Flash; the usage of this format will make the importing process faster.
7. It's advisable to create a test scene in your project where animators can safely test out their work; this will not only allow them to see how their animations will look like, but will also minimize the chances of some important aspect of the project breaking down due to untested material.
8. It's recommended for the team to agree on each texture's size beforehand. The latter will ensure all textures will be at their optimum size. Moreover, it will reduce the amount of pixelation or blurring that resizing the image might provoke.
9. Be aware of easily avoidable/fixable errors that may arise during the usage of Flash to Unity. Some common problems related to the improper usage of this tool include:
    a. Invalid texture atlas' settings.
    b. Animated sprites having color transformations. They are only partially supported, and must be used with care. Currently, they can only be activated from code.
    c. Animations referencing symbols outside the textures or animations folder. Animations may only reference textures from the "textures" folder and other movie clips on the "animations" folder.
    d. Incorrectly declared symbols. Remember to declare all textures as graphic symbols and all animations as movie clip symbols.
10. Create a single texture per each background, so the textures will have only a keyframe with the background.

11. Avoid adding the backgrounds in textures that have images with transparencies.
12. Once you have imported the .xfl by using the Flash importer tool (Tools -> F2U -> Flash -> Import XFL), find the material automatically created for the background, and set the shader to: 'F2U -> Sprite -> CG -> Background', find the texture atlas (png) automatically created and set the texture format to 'RGB 24 bit."

13. Separate the animations of your .XFL project into several folders representing each of the scenes you are going to create. This allows Flash to Unity to better organize the generated assets in your Unity project.
14. Prefix the name of your textures with the name of the scene they are going to be used. This lets you find the associated atlas more easily later on.

Creating background textures

- Create a single texture per each background, so the textures will have only a key frame with the background.
- Avoid adding the backgrounds in textures that have images with transparencies.
- Once you have imported the .xfl by using the Flash importer tool (Tools -> F2U -> Flash -> Import XFL), find the material automatically created for the background, and set the shader to: ‘F2U -> Sprite -> CG -> Background’, find the texture atlas (png) automatically created and set the texture format to: ‘RGB 24 bit’.

The goal of this process is to improve the rendering performance using a better shader for backgrounds and also to improve the memory required for backgrounds removing the alpha channel (removing the alpha channel will reduce the memory size required for the background).

It’s a good practice to create a separate .XFL file for every scene to be create, mainly because Flash tends to crash and corrupt .XFL files if they are sufficiently big, but also because it’s dangerous if two different people try to modify the same project at the same time, even if they are modifying different parts of the .XFL project.

On cameras and 3D

The usage of 3D cameras to display Flash to Unity components is heavily discouraged; several issues may arise, and objects may not display properly. In order to use Flash to Unity in a 3D environment, at least two cameras must exist: an orthographic camera, used exclusively to display 2D elements generated with Flash to Unity, and another camera that draws everything else. The orthographic camera must be the scene's main camera, and it's recommended to use the one Flash to Unity builds when selecting the "GameObject > Create Other > F2U > Camera2D" option. It's important to set one of more of the cameras' clear flags (the latter if using more than two), most commonly the orthographic one, to "don't clear", as to allow the other elements to be displayed properly.