**Tecnológico de Costa Rica**

Escuela de Ingeniería en Computación

**Fair Play Labs**

# “Flash To Unity”

Informe final de la Práctica de Especialidad para optar por el título de bachillerato en Ingeniería en Computación

Gabriel Fallas Carrera

San José, noviembre, 2013

## Resumen

El presente documento constituye el informe final del proyecto de Práctica de especialidad, curso para optar por el grado de Bachiller en Ingeniería en Computación.

El objetivo principal de este informe es presentar y dar a conocer el estado final del proyecto “Flash To Unity”. “Flash To Unity” inicialmente era una herramienta para exportar animaciones de Flash a Unity, brindaba un SDK para leer las animaciones y reproducirlas en Unity, pero ambas funcionalidades se hacían en procesos separados de manera que no existía un “pipeline” que integrara los distintos procesos y de esta forma facilitara la exportación de Flash a Unity.

Ahora la herramienta es un “plugin” para el motor de videojuegos Unity3D, el cual integra todos los procesos del “pipeline” en el mismo “plugin” y brinda más funcionalidades que su versión anterior de esta manera se facilita el proceso de importación. El objetivo principal del proyecto ha consistido en el desarrollo de mejoras entre las cuales la integración de los procesos era primordial, tal como se planeó, también cabe destacar que uno de los objetivos principales de la herramienta ha sido el facilitar el desarrollo de videojuegos en 2D.

En las secciones 4 y 5, respectivamente, se incluye la descripción del problema retomado del primer informe con breves ajustes de su contenido actualizado a la fecha y la solución final implementada basada en la sección 4 retomada del segundo informe.

En la sección 6 se encuentran las conclusiones y experiencias personales de lo que he aprendido durante el período de la práctica de la especialidad.

Adicionalmente, al final del informe se incluye el plan de trabajo del proyecto.



**Palabras claves:** Unity3D; SDK; videojuegos 2D; Flash; pipeline; plugin.

## Contenido

# 1. Contexto del proyecto

## 1.1. Empresa

La empresa donde el proyecto de práctica de especialidad es llevado a cabo es Fair Play Labs S.A.

Fair Play Labs es un estudio de desarrollo de juegos localizado en Costa Rica, fundado en el año 2003, se enfoca en desarrollar juegos no violentos para usuarios de diversas edades, desde niños de 8 años hasta adultos mayores de más de 80 años.

El estudio posee una amplia experiencia en las áreas de diseño de juegos, arte, software, música y efectos de sonido. Ofrece servicios de: Diseño de juegos, desarrollo de juegos para PS3, PSP, iOS, Android, Web (Flash/Facebook), PC y Mac.

***Misión***

Desarrollar juegos divertidos para personas con edades desde 8 hasta 80 años, los cuales son innovadores y no violentos, destacándonos en plataformas móviles, y divirtiéndonos haciéndolos! Convertirnos en una gran empresa para trabajar.

***Visión***

Ser reconocido como una de las mejores empresas en la industria de entretenimiento Digital Interactive, que combina la creatividad, la innovación y la diversión

***Valores***

- Equidad
- Creatividad e innovación
- Diversión y calidad
- Entusiasmo y compromiso
- Confidencialidad y perseverancia

- Aprendizaje y humildad
- Reconocimiento y orgullo
- Trabajo en equipo y solidaridad
- Integridad y honestidad

***Departamentos o unidades de la empresa***

- Administración
- Producción
- Arte
- Música
- **Desarrollo de software: departamento donde se realizará el proyecto de práctica de especialidad**
- Aseguramiento de calidad

***Contacto de la empresa***

- Dirección: Edificio La Nacional, segundo piso, Zapote, San José, Costa Rica
- Email: info@fairplaylabs.com
- Sitio web: http://www.fairplaylabs.com/
- Teléfono: (506) 2226-1250
- Fax: (506) 2227-1904

***Organigrama de la empresa***

En la siguiente figura se muestra la organización de la empresa:

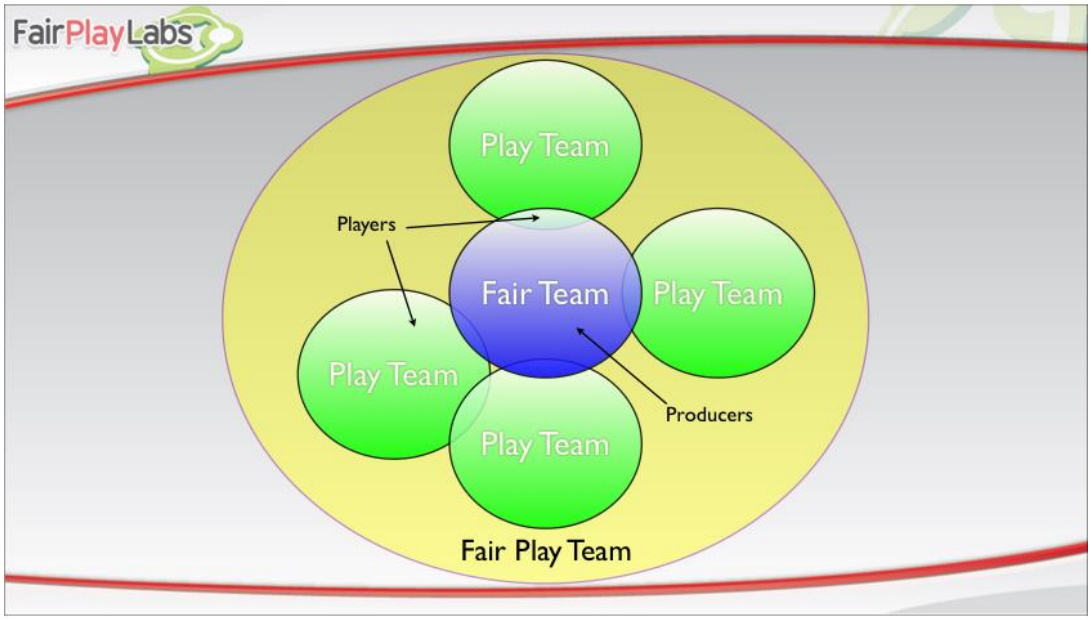


**Figura 1.1 Organigrama Fair Play Labs**

En Fair Play Labs no existe un organigrama jerárquica de los departamentos, por lo que la organización de la empresa se basa en un grupo principal, Fair Team, el cual lo componen 4 Producers, grupos de trabajo llamados Play Team, los cuales se componen de un Producer, desarrolladores, al menos un artista y un desarrollador de QA.

## 1.2. Antecedentes del proyecto

El proyecto se denomina “FlashToUnity”, el estudio lo desarrolló inicialmente como un software de utilidad para facilitar a los artistas la exportación de animaciones hechas en Adobe Flash al motor de videojuegos Unity3d y así simplificar el proceso de desarrollo general del videojuego.

La herramienta ha sido y sigue siendo muy útil, ya que funciona como exportador de animaciones y “assets”, los cuales son los componentes principales de las animaciones y requeridos en Unity3d para permitir el desarrollo de los videojuegos.

El objetivo general de este proyecto en la Práctica de Especialidad es el diseño y desarrollo de mejoras a esta herramienta, las mejoras se requieren para optimizar su uso interno y ofrecerla como un “plugin” en el “Marketplace” de la tienda de Unity 3D.

# 2. Descripción del problema

## 2.1. Problema

Actualmente la versión anterior de la herramienta “Flash to Unity” consta de dos componentes: el “exporter” que toma los archivos de Flash, y genera archivos con los datos y la metadata requeridos para representar estas animaciones en Unity. Y una serie de SDKs en Unity que utilizan estos archivos en tiempo de ejecución para generar las animaciones utilizando OpenGL ES. Ambos componentes están desarrollados en C#.

Aun se carece de un “importer” para las animaciones desde Unity y de una integración total de los componentes de la herramienta para ser utilizados en Unity3d, por lo que resulta tedioso y no integral el proceso de exportación de Flash a Unity.

## 2.2. Solución

La solución consiste una serie de mejoras importantes que se implementaran a la herramienta, la próxima versión de la herramienta será la 2, las principales mejoras que se esperan tener son:

- La integración de la herramienta como un complemento de utilidades (plugin) al editor del Motor de Videojuegos Unity3d, esta mejora se compone de la parte principal de la herramienta, el cual es el Importador para las animaciones y de otra serie de mejoras importantes que se listan y detallan a continuación:
    1. Integración del exportador y generador de bitmaps - necesario para crear las texturas atlas utilizadas para las animaciones y sprites, ambos componentes de Flash to Unity.

2. Integración e implementación de un importador de Fonts – consiste en que la herramienta pueda importar fonts (escribiéndolos) en archivos con un formato predeterminado para que sean interpretados y utilizados por el componente de Texto de “Flash to Unity”.
3. Integración e implementación del uso de los componentes de “Flash to Unity” (SDK) - Consiste en agregar un menú para que los artistas y desarrolladores puedan agregar los componentes Sprite, Animation, Sound, componentes de UI y Localization directamente a las escenas de los juegos.
4. Integración e implementación del importador de animaciones – Consiste en integrar e implementar un importador de animaciones, el cual es el encargado de tomar las animaciones creadas por los artistas en Flash e importarlas hacia Unity, escribiéndolas y leyéndolas en un formato determinado para su uso en el componente de Flash to Unity, animation.
5. Diseño e integración de documentación – Para esta versión se integrara una documentación para los desarrolladores, la documentación incluirá toda la información detallada de los componentes de “Flash to Unity” y también contará con ejemplos de códigos que sirvan como guía.

## 2.3. Patrocinadores

**Tabla 2.1 Patrocinadores del proyecto**

| Puesto | Nombre | Labores que realiza | Responsabilidades en el proyecto |
|---|---|---|---|
| Cliente | Fair Play Labs | Proporcionar los requerimientos del proyecto | Supervisar el proyecto, espeficicar nuevos requisitos. |
| Gerente general | Claudio Pinto | Supervisión del proyecto. | Supervisar el proyecto y coordinar reuniones con la profesora asesor. |
| Producer | Christian Sánchez | Administración, planificación y control del proyecto. | Planificar y definir tareas u objetivos a realizar, signar tiempo de realización a las tareas y revisar los avances semanales. |
| Artista digital | Roberto Dobles | Creación de los recursos del sistema (imágenes y animaciones) | Crear y modificar los recursos digitales necesarios para los ejemplos integrados en el proyecto. |
| Ingeniero senior de desarrollo | Leonardo Rojas | Desarrollo, asesoría y apoyo al practicante en cuestiones técnicas. | Desarrollar, asesorar y apoyar al practicante en cuestiones técnicas. |
| Aseguramiento | No | Realización de | Realizar control de calidad y |

| | | | |
|---|---|---|---|
| de calidad | asignado | pruebas a los proyectos antes de salir a la venta. | pruebas generales del sistema. |
| Practicante Desarrollador | Gabriel Fallas Carrera | Desarrollo del proyecto. | Desarrollar las tareas u objetivos asignados, investigar las tecnologías a utilizar y asegurar la calidad del proyecto. |
| Desarrollador | José Solano | Desarrollo del proyecto | Desarrollar las tareas u objetivos asignados, investigar las tecnologías a utilizar y asegurar la calidad del proyecto. |
| Profesora asesora | Gaudy Esquivel | Supervisión práctica de especialidad. | Velar por el desarrollo de la práctica de la especialidad del estudiante y por el cumplimiento de responsabilidades del mismo. |

## 2.4. Necesidades y expectativas

**Tabla 2.2 Necesidad - 01**

| **Identificación** | **Necesidad - 01** |
|---|---|
| Necesidad | Integración del exportador y generador de bitmaps |
| Prioridad | Alta |

| | |
|---|---|
| Problema | Se necesita integrar el proceso de generación de bitmaps a la herramienta. |
| Solución actual | El proceso se debe realizar manualmente. |
| Solución Propuesta | Integrar la automatización del proceso de generación de bitmaps a la herramienta. |

**Tabla 2.3 Necesidad - 02**

| **Identificación** | **Necesidad - 02** |
|---|---|
| Necesidad | Integración e implementación de lectura y escritura de Fonts |
| Prioridad | Media |
| Problema | Se necesitan implementar e integrar el proceso para importar Fonts, este proceso requiere de la escritura y lectura de los archivos que contienen los Fonts, los Fonts son necesarios para el componente de Texto. |
| Solución actual | No existe un importador de Fonts. |
| Solución Propuesta | Implementar e integrar la importación de Fonts por medio de la escritura y su interpretación por medio de su lectura. |

**Tabla 2.4 Necesidad - 03**

| **Identificación** | **Necesidad - 03** |
|---|---|
| Necesidad | Integración e implementación del uso de los componentes de Flash to Unity (SDK) |
| Prioridad | Alta |

| | |
|---|---|
| Problema | Se requiere que los desarrolladores puedan hacer uso de los componentes de Flash to Unity para agregarlos a las escenas de los juegos. |
| Solución actual | El menú existente aún no contiene todos los componentes. |
| Solución Propuesta | Implementar e integrar una opción de menú para poder agregar los componentes que se usan para las escenas de los juegos. |

**Tabla 2.5 Necesidad - 04**

| **Identificación** | **Necesidad - 04** |
|---|---|
| Necesidad | Integración e implementación del importador de animaciones |
| Prioridad | Alta |
| Problema | Se requiere que los desarrolladores y artistas puedan usar importar las animaciones de Flash a Unity. Se necesita un importador desde Unity. |
| Solución actual | El proceso se realiza manualmente, se deben ejecutar varios programas y herramientas para completar el proceso de exportación. |
| Solución Propuesta | Implementar e integrar una opción de menú para que los artistas y desarrolladores puedan importar las animaciones hechas en Flash a Unity, las animaciones se escriben y leen con el componente animation. |

| **Identificación** | **Necesidad-05** |
|---|---|
| Necesidad | Diseño e integración de documentación |
| Prioridad | Alta |
| Problema | Se requiere que la herramienta tenga una documentación de ayuda para los desarrolladores que la utilizarán. |
| Solución actual | No existe documentación para los desarrolladores, documentación de las clases y diagramas de las clases. |
| Solución Propuesta | Diseñar e implementar una documentación de todos los componentes de Flash to Unity, además se debe integrar al menú de la herramienta para que los desarrolladores puedan acceder a la documentación, la cual estará almacenada en un servidor. |

## 2.5. Perspectiva, supuestos y dependencias del producto

**Perspectiva**

Se proyecta implementar la herramienta “Flash to Uniy” como un “plugin” para Unity3d, además de las mejoras ya mencionadas, de manera que se permita facilitar y controlar con más eficiencia el desarrollo de videojuegos que incluyen arte hecha con la herramienta Adobe Flash.

**Supuestos**

La herramienta “Flash To Unity” está desarrollada en el lenguaje C# y se seguirá desarrollando en el mismo lenguaje con los mismos estándares de código.

Se contara con sprites y animaciones hechas y arte nuevo para realizar casos de prueba e integrarlos en las escenas que irán incluidas en la herramienta.

**Dependencias**

Para realizar el “pipeline” completo es necesario que se cuente con los siguientes programas:

- Texture Packer: Programa para crear texturas.
- Adobe Flash: Programa para crear animaciones.
- Unity3d: Programa para crear videojuegos, la herramienta a desarrollar se integra a este programa como un “plugin”.

## 2.6. Requerimientos no funcionales

A continuación se muestra una lista de los requerimientos no funcionales del proyecto:

- Todos los recursos de la herramienta deben estar en un solo paquete de Unity3D.
- Se debe seguir con el estándar de código de las versiones anteriores y el estándar establecido por la empresa.

- La interfaz debe ir integrada como una opción de menú adicional en la herramienta Unity3d.
- Se deberan incluir escenas que muestren y ejemplifiquen el uso y la funcionalidad de la herramienta.

### 2.7. Características generales

**Tecnologías utilizadas:**

- Apple MacOS y Microsoft Windows
- Apple iOS y Google Android
- Lenguaje C# y Java
- Adobe Flash
- ActionScript
- Unity3D
- OpenGL ES
- Subversion
- JIRA

## 3. Análisis de los Riesgos

**Tabla 3.1 Análisis del riesgo - 01**

| **Identificación** | **Riesgo-01** |
|---|---|
| Riesgo | Nuevos requerimientos o cambios de requerimientos |
| Descripción | Cambios muy grandes solicitados por el *cliente durante el proceso de desarrollo de las mejoras de la herramienta. |
| Categoría | Diseño y requerimientos |
| Posible causa | En la especificación diseñada no se tuvo claridad sobre cuales eran todas las mejoras para agregar a la herramienta |
| Impacto (I) | 8 semanas |
| Probabilidad de | 90% |

| | |
|---|---|
| ocurrencia (P) | |
| Exposición (I*P) | 7,2 semanas - Crítico |
| Estrategia de evasión | No se puede evadir, se acepta el riesgo |
| Estrategia de mitigación | Ir ajustando el desarrollo con las mejoras que se van identificando al momento y extender el período de desarrollo en lo posible. |
| Estrategia de contingencia | Se agregan las nuevas tareas y se implementan las nuevas funcionalidades de los requisitos, implica laborar más horas de desarrollo. |

* Como se mencionó anteriormente el cliente, es la empresa misma, por lo que los cambios y nuevas funcionalidades para el proyecto se presentan muy seguido.

**Tabla 3.2 Análisis del riesgo - 02**

| **Identificación** | **Riesgo-02** |
|---|---|
| Riesgo | Cambios en el software |
| Descripción | Cambios identificados durante los “test cases" para hacer el "tuning" de la herramienta. |
| Categoría | Técnico |
| Posible causa | En la especificación no se tuvo claridad sobre cuales eran todas las mejoras para agregar a la herramienta |
| Impacto (I) | 2 semanas |
| Probabilidad de ocurrencia (P) | 80% |
| Exposición (I*P) | 1,6 semanas - Moderado |

| | |
|---|---|
| Estrategia de evasión | No se puede evadir, se acepta el riesgo |
| Estrategia de mitigación | Ir haciendo “test cases” en paralelo con el desarrollo. |
| Estrategia de contingencia | Se realizan las mejoras identificadas y se implementan en la herramienta para hacer el “tuning”. |

**Tabla 3.3 Análisis del riesgo - 03**

| **Identificación** | **Riesgo-03** |
|---|---|
| Riesgo | Fallos en los builds generados por la herramienta para múltiples dispositivos |
| Descripción | Se encuentran errores en los builds generados para multiples dispositivos móviles, builds para iOS y Android OS. |
| Categoría | Técnico |
| Posible causa | Ocurren errores específicos con la plataforma del OS con los builds generados |
| Impacto (I) | 2 semanas |
| Probabilidad de ocurrencia (P) | 50% |
| Exposición (I*P) | 1 semana - Baja |
| Estrategia de evasión | No se puede evadir, se acepta el riesgo |
| Estrategia de mitigación | Ir haciendo builds de pruebas de integración durante el desarrollo. |

| | |
|---|---|
| Estrategia de contingencia | Se realizan las pruebas necesarias para identificar los posibles errores en los builds generados y se corrigen los errores encontrados. |

**Tabla 3.4 Análisis del riesgo - 04**

| **Identificación** | **Riesgo-04** |
|---|---|
| Riesgo | Enfermedades o incapacidades físicas |
| Descripción | Enfermedades o accidentes graves que obligan a los involucrados del proyecto a incapacitarse por cierto tiempo y no poder trabajar |
| Categoría | Persona |
| Posible causa | Contagio de alguna enfermedad o algún accidente |
| Impacto (I) | 1 semana |
| Probabilidad de ocurrencia (P) | 40% |
| Exposición (I*P) | 0,4 semana - Baja |
| Estrategia de evasión | No se puede evadir |
| Estrategia de mitigación | Estar en constante revisión médica y ponerse inyecciones contra las enfermedades contagiosas más comunes |
| Estrategia de contingencia | Extender el período de desarrollo |

**Tabla 3.5 Análisis del riesgo – 05**

| **Identificación** | **Riesgo-05** |
|---|---|
| Riesgo | Problemas con el entorno de desarrollo |
| Descripción | Problemas de compatibilidad de IDE u otras herramientas con el sistema operativo |
| Categoría | Técnico |
| Posible causa | Inestabilidad de operación de las herramientas en Windows |
| Impacto (I) | 1 semana |
| Probabilidad de ocurrencia (P) | 80% |
| Exposición (I*P) | 0,8 semana - Baja |
| Estrategia de evasión | Utilizar Mac o Linux |
| Estrategia de mitigación | Desarrollar en Mac o Linux |
| Estrategia de contingencia | Desarrollar en Mac o Linux |

## 4. Objetivos y Alcances del sistema

**El objetivo general del proyecto es:**

- Desarrollar las mejoras para la importación de las animaciones e integración del proceso de importación de Flash a Unity de la herramienta "Flash To Unity".

**Los objetivos específicos del proyecto son:**

- Identificar y documentar los requerimientos de las mejoras a la herramienta.
- Investigar opciones de tecnologías existentes que puedan incorporarse a la solución.
- Diseñar y desarrollar las mejoras identificadas.
- Probar integralmente la nueva herramienta.
- Optimización ("fine tuning") de la herramienta.

**Alcances del sistema:**

- Integración del exportador y generador de bitmaps.
- Integración e implementación de un importador de Fonts.
- Integración e implementación del uso de los componentes de Flash to Unity (SDK).
- Integración e implementación del importador de animaciones, este alcance se divide en:
    - Mejoraras en el Animation Pipeline.
    - Nuevas funcionalidades en las animaciones.
- Integración de la Documentación para los Desarrolladores y Artistas.

# 5. Solución implementada

En esta sección se muestra el diseño, clases e interfaces de la solución implementada, por cada subsección se presenta una descripción de cómo está implementada.

## 5.1 Modelo de Diseño

### 5.1.1 Arquitectura conceptual de la solución

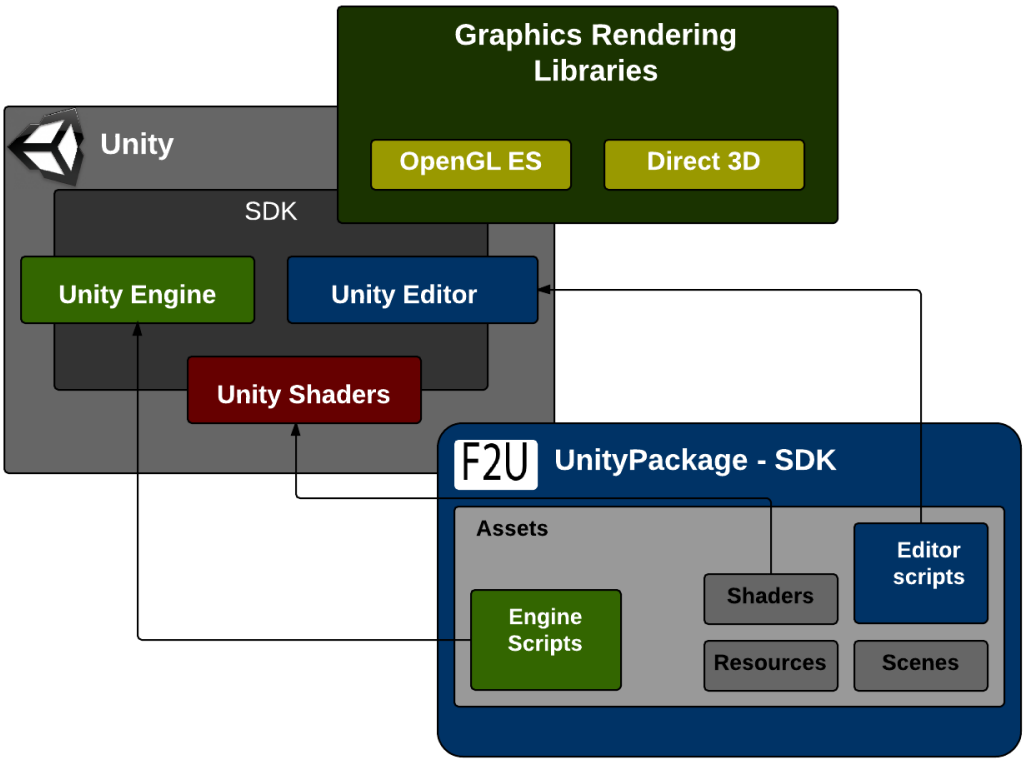


Figura 5.1 Arquitectura conceptual de Flash To Unity

**Editor Scripts:** Las clases del "Editor" son las clases que generan la interfaz de usuario y componente por medio del cual se interactúa con las clases de "Engine"; es decir, estas clases se ejecutan en tiempo de edición "EditTime". En esta parte

también se encuentran las clases que se encargan de convertir las animaciones de Flash y todos sus componentes en archivos binarios.

**Engine Scripts:** Las clases del “Engine” son las clases que proveen toda la funcionalidad necesaria para interpretar, brindar y renderizar los componentes de necesarios de las animaciones. Estas clases se ejecutan en tiempo de ejecución “RunTime”.

## 5.1.2 Diagramas de Clases

### Clases del “Engine”

El siguiente es un diagrama completo de las clases y su jerarquía:

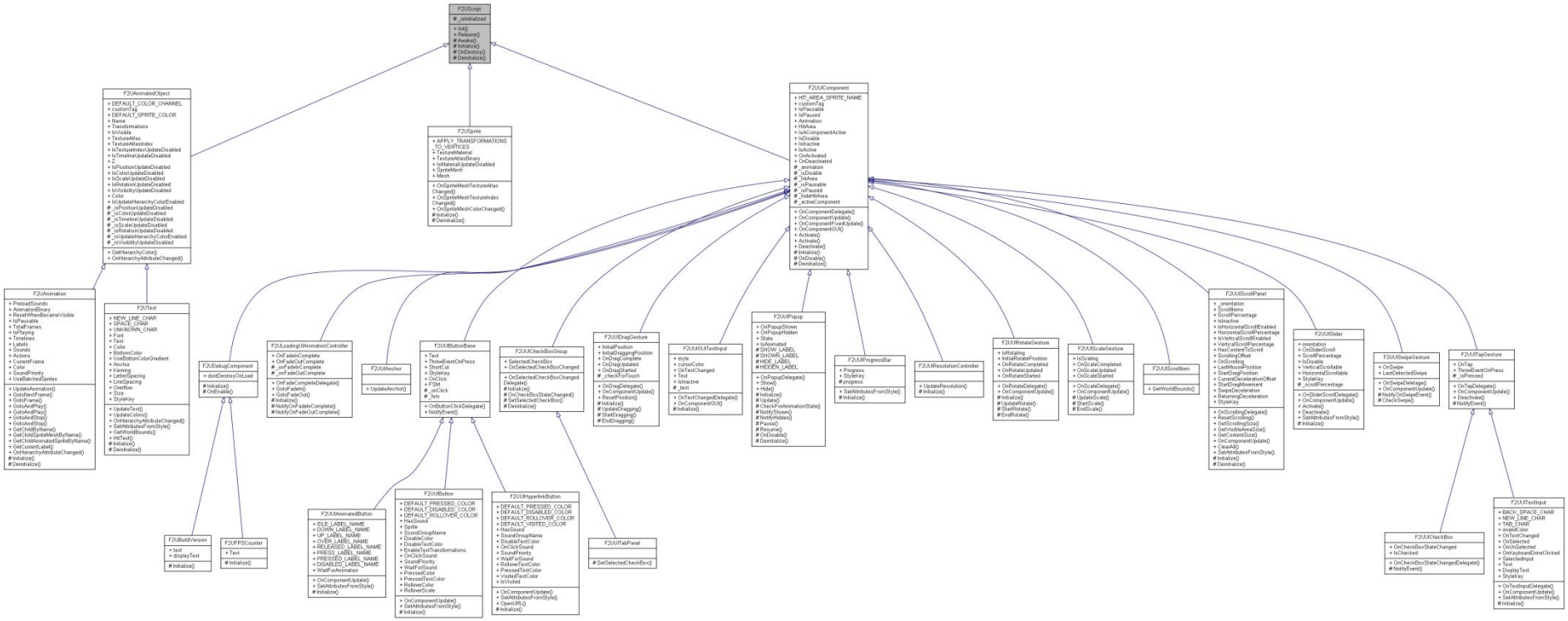

Figura 5.2 Diagrama de clases del “Engine”

“FU2Script” se llama la clase base de la cual todas las clases de F2U heredan propiedades y métodos. Esta clase contiene y expone las propiedades de la clase base “MonoBehaviour“ de UnityEngine, además contiene métodos propios para la inicialización y destrucción de sus componentes.

Las tres clases principales que heredan de la clase base del “Engine” son “F2UAnimatedObject”, “F2USprite” y “F2UUIComponent”, las cuales se mencionan y explican con más detalle seguidamente.

### F2UAnimatedObject

Esta clase representa la funcionalidad y el comportamiento necesario para poder crear objetos animados (animaciones y sprites). Las propiedades que expone principalmente son las transformaciones de los objetos.

#### Clases hijas

Las dos clases que heredan de “F2UAnimatedObject” son:

- **F2UAnimation**: Esta clase representa la funcionalidad proveída por la clase “Movieclip” de Flash.
- **F2UText**: Esta clase representa la funcionalidad para mostrar textos 3D en Unity.

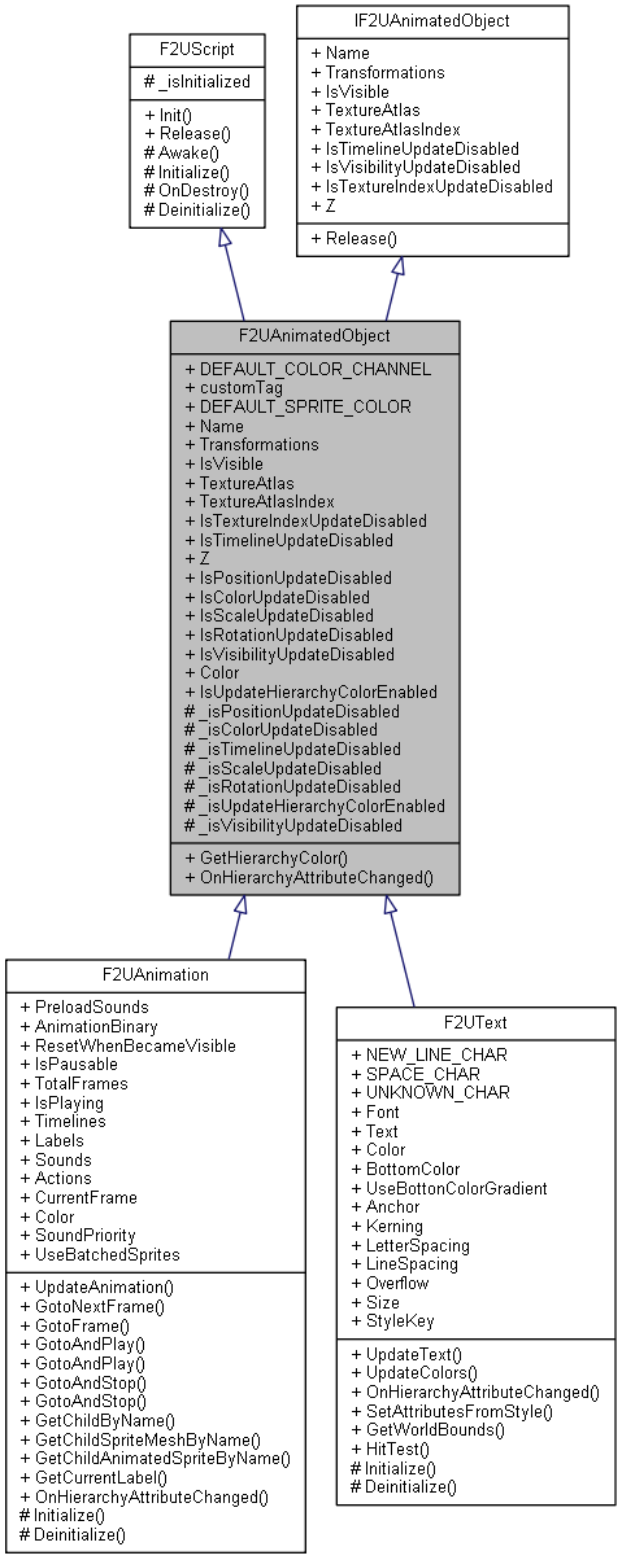


**Figura 5.3 Diagrama de Clases de F2UAnimatedObject**

## F2USprite

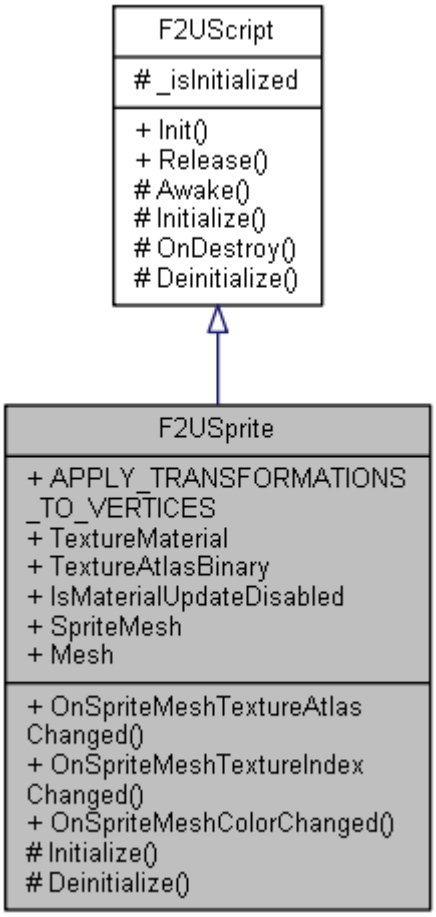


**Figura 5.4 Diagrama de clases de F2USprite**

## F2USpriteMesh


**Figura 5.5 Diagrama de clases de F2USpritMesh**

## F2USprite

Esta clase contiene la propiedad principal y base para representar los sprites como “Graphics” de Flash, “F2USpriteMesh”. Esta clase utiliza una Interface para exponer sus métodos en Interface al igual que lo hace “F2UAnimatedObject”. “F2USpriteMesh” es la clase que maneja el comportamiento de las texturas de los sprites, su jerarquía, transformaciones y posiciones dentro de la pantalla de juego.

## UI Components: F2UUIComponent

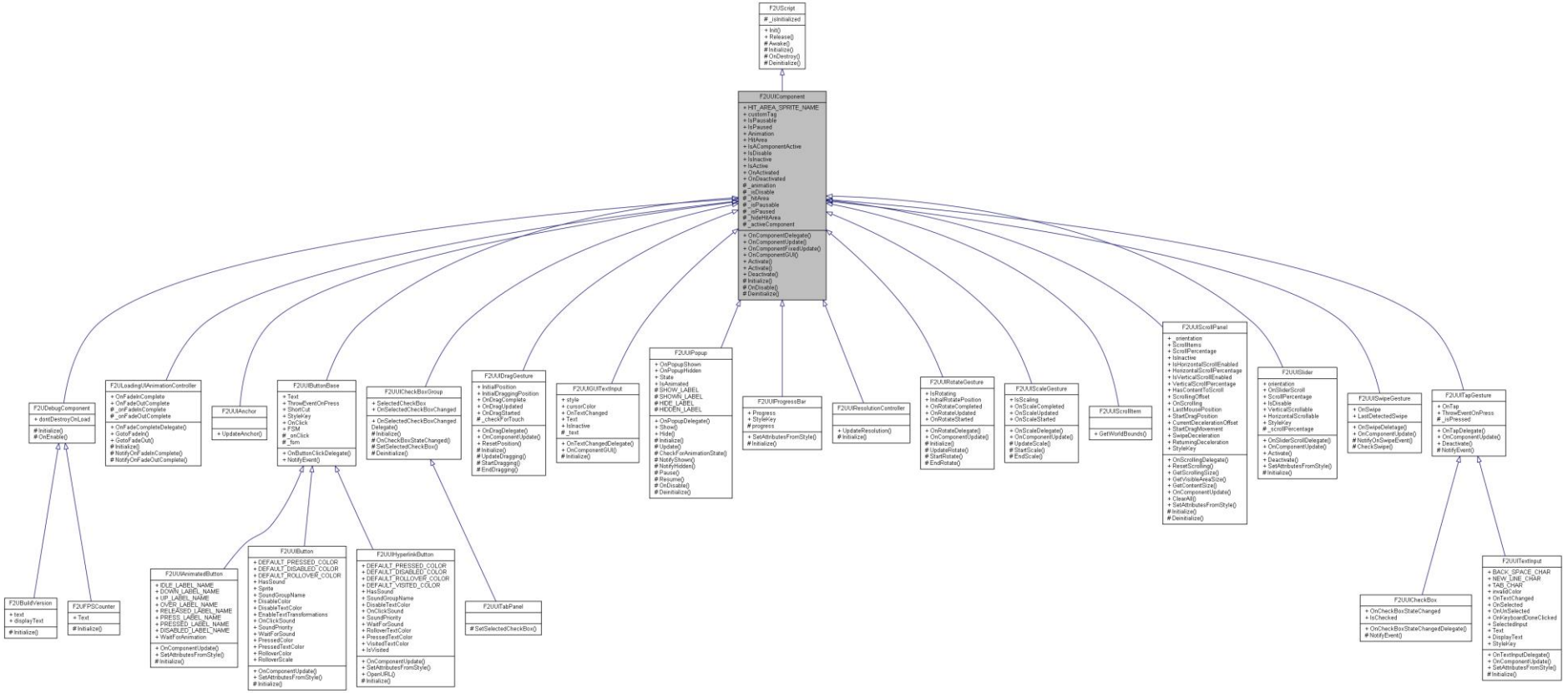


**Figura 5.6 Diagrama de clases los componentes F2UUIComponent**

Esta es la clase base para todos los elementos de Interface gráfica de usuario “UI” que provee F2U, contiene las propiedades necesarias para cada objeto gráfico “F2UAnimatedObject” o “F2USprite”. Por ejemplo, un botón “F2UButton” podría aplicarse a cualquiera de ambos objetos.

Las clases derivadas; específicas para cada componente de “UI” tienen sus propiedades y comportamientos para su respectiva su funcionalidad.

### Interfaces: IF2UAnimatedObject

Interfaz que expone las propiedades y métodos de las transformaciones y texturas de las clases “F2UAnimatedObject” y “F2USpriteMesh”.

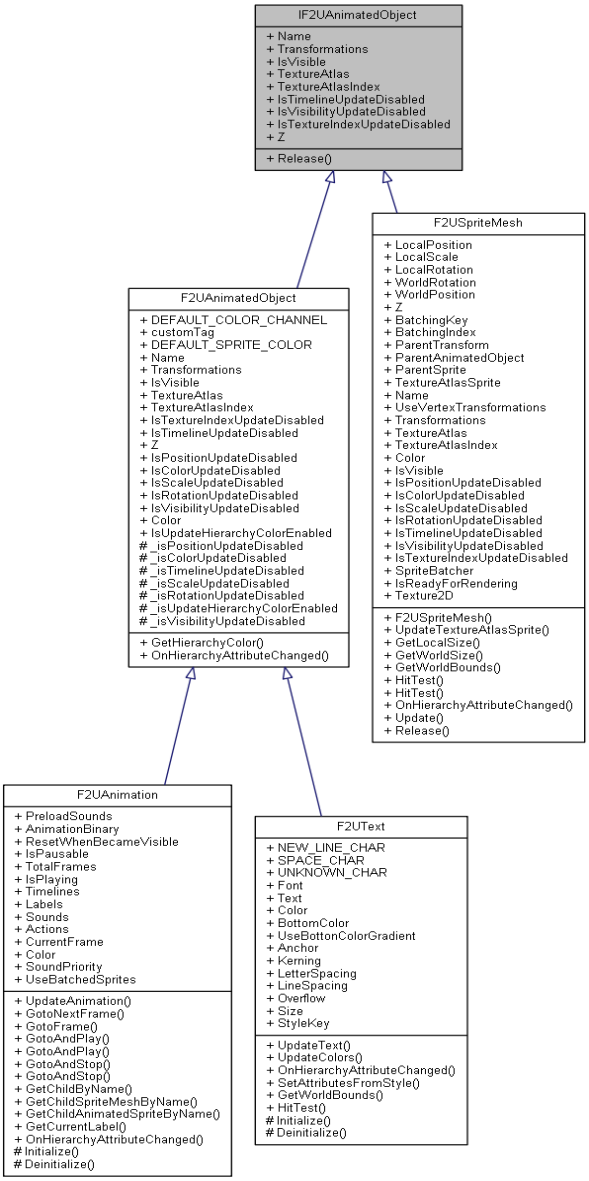


**Figura 5.7 Diagrama de clase de la interface IF2UAnimatedObject**

## Clases del “Editor”

### Binary writers

Son las clases que se encargan de convertir las animaciones de Flash y todos sus componentes a archivos en formato binario.

**Animation**


**Figura 5.8 Diagrama de F2UAnimationBinarywriter**

**Font**

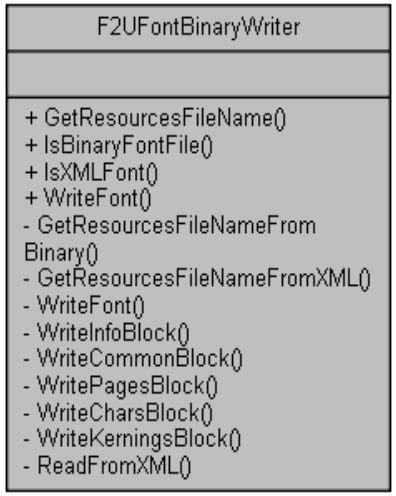


**Figura 5.9 Diagrama de F2UFontBinarywriter**

**Text**

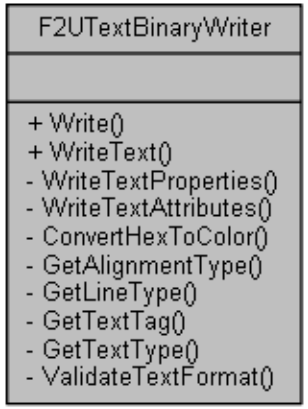


**Figura 5.10 Diagrama de F2UTextBinarywriter**

**Texture**

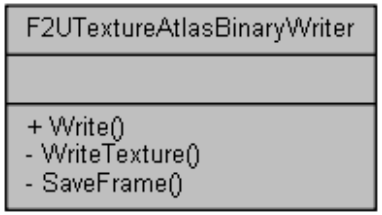


**Figura 5.11 Diagrama de F2UTextureAtlasBinarywriter**

### Interfaz de Usuario: Menú

Esta clase hace uso de UnityEditor, librería para poder crear componentes de UI de Unity. Por medio de esta clase se hace la opción de menú para hacer uso de la funcionalidad que provee F2U.


**Figura 5.12 Diagrama de F2UMenu**

## Clases Utilitarias

Estas clases se encargan de las configuraciones, validaciones y métodos utilitarios necesarios para hacer las importaciones de Flash a Unity.

### Binary Importer

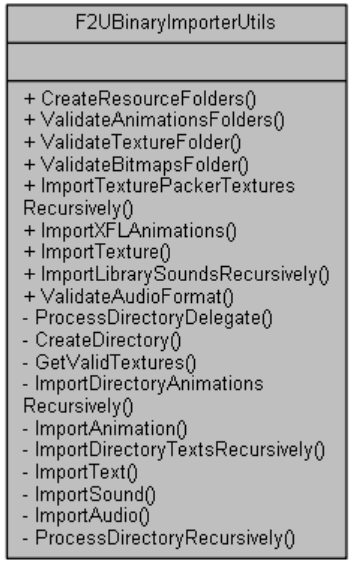


Figura 5.13 Diagrama de F2UBinaryImpoterUtils

### Editor


Figura 5.14 Diagrama de F2UEditorUtils

### Editor Command

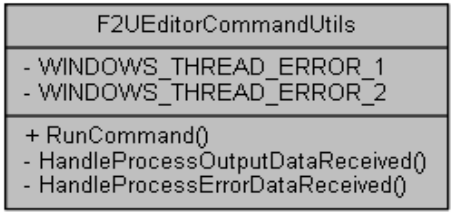


Figura 5.15 Diagrama de F2UEditorCommandUtils

### Editor Settings

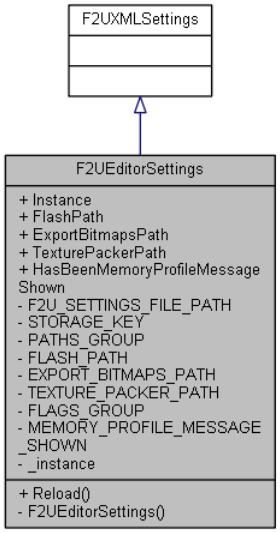


Figura 5.16 Diagrama de F2UEditorSettings

### 5.1.3 Interfaces de Usuario

Al ser un “plugin” para Unity3d la interfaz es muy sencilla, en este caso; es tan solo una opción que se integra al menú de Unity3d, el cual no requiere de diseño ya que se crea por medio de la Interfaz que Unity provee por medio de su librería de UnityEditor.

**Opción de Menú de “F2U”**

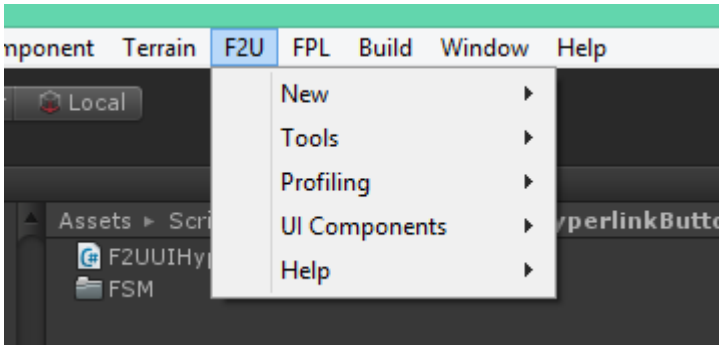


Figura 5.17 UI - Menu

**Opción Importar/Exportar de “F2U”**

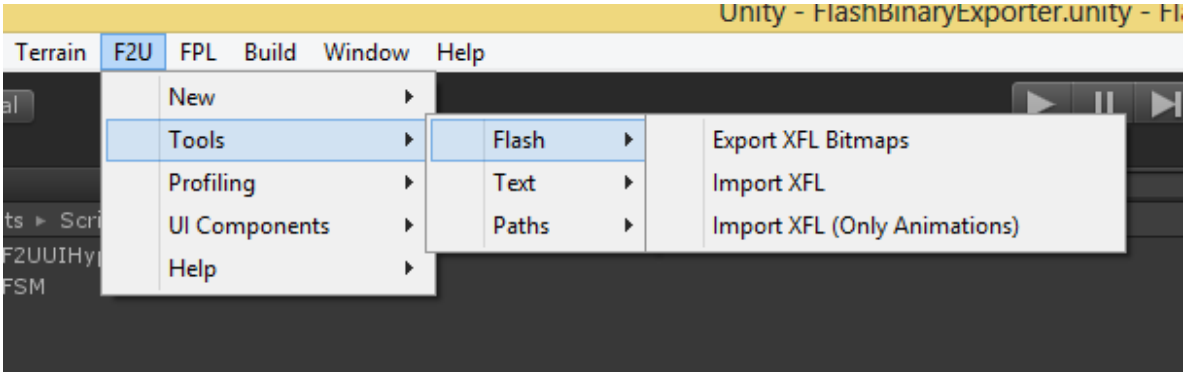


Figura 5.18 UI – Importer option

## 6. Conclusiones y comentarios

Se cumplieron los objetivos propuestos, hasta la fecha el proyecto se apegó lo más posible al cronograma propuesto inicialmente, se han generado dos lanzamientos de la versión 2 para uso interno dentro de la empresa.

El objetivo principal fue alcanzado. La integración de la herramienta como un “plugin” para el motor de videojuegos Unity3D se implementó, de esta manera brindando las mejoras planeadas de: integrar los importadores de las animaciones de Flash a Unity, importador de fonts, integración de documentación en línea del API integrada en menú y la integración de menú para la creación de los componentes del API de Flash to Unity, componentes tales como animaciones, sprites, textos y componentes de UI.

Los productos que se entregan son, la herramienta completamente funcional para su uso dentro de la empresa y para su comercialización en el “Asset Store” de Unity, la documentación de los requerimientos técnicos y funcionales y el manual de usuario.

Dentro de las experiencias adquiridas, las principales fueron el trabajo en equipo como una de las experiencias y de los valores de trabajo más importante, el tener que interactuar e intercambiar ideas, opiniones entre diferentes personas para llegar a una solución correcta y óptima.

También la aplicación de estándares de software, por medio de esta experiencia se aprende a cómo usar un sistema para el manejo correcto de versiones del software. Otro gran aprendizaje y experiencia ha sido el tener que trabajar bajo estándares de código establecidos y adecuados acordes a lo requerido por la empresa, de esta manera se escribe código muy entendible y se facilita el intercambio de desarrolladores entre diferentes proyectos.

Y como última experiencia, el trabajar con una metodología ágil por medio del uso de software para la administración de las tareas del proyecto, para controlar los sprints y tiempos de desarrollo, esto facilita y agiliza en gran manera el proceso de

desarrollo, además de brindar una organización eficiente y un seguimiento constante del tiempo de efectivo de trabajo del proyecto, ya que las tareas que se asignan son muy puntuales y cada tarea tiene su tiempo estimado para ser cumplida.

## 7. Plan de trabajo

**Tabla 7.1 Plan de trabajo del proyecto**

| Entregable | Fecha | Descripción |
|---|---|---|
| **Kick Off** | 22/07/13 | Inicio del proyecto y capacitación (2 semanas) |
| **Fase 1** | 05/08/13 | Requerimientos técnicos y funcionales (2 semanas) |
| **Fase 2** | 19/08/13 | Diseño y especificaciones finales de la herramienta (2 semanas) |
| **Fase 3** | | Desarrollo y pruebas de la herramienta (8 semanas) |
| | 02/09/13 | Sprint 1 – *First Delivery* (2 semanas) |
| | 16/09/13 | Sprint 2 – *Second Delivery* (2 semanas) |
| | 30/09/13 | Sprint 3 – *Refinements* 1 (2 semanas) |
| | 14/10/13 | Sprint 4 – *Refinements* 2 (2 semanas) |
| **Fase 4** | 28/10/13 | Pruebas, "*tuning*" y optimización (perfeccionamiento) de la herramienta (2 semanas) |
| **Gold** | 11/11/13 | Errores críticos corregidos y producto listo para entrega. |