# Historia

La empresa Unity Technologies fue fundada en 2004 por David Helgason (CEO), Nicholas Francis (CCO), y Joachim Ante (CTO) en Copenhague, Dinamarca después de su primer juego, GooBall, que no obtuvo éxito. Los tres reconocieron el valor del motor y las herramientas de desarrollo y se dispuso a crear un motor que cualquiera pudiera usar a un precio accesible.

El éxito de Unity ha llegado en parte debido al enfoque en las necesidades de los desarrolladores independientes que no pueden crear ni su propio motor del juego ni las herramientas necesarias o adquirir licencias para utilizar plenamente las opciones que aparecen disponibles.

El enfoque de la compañía es "democratizar el desarrollo de juegos", y hacer el desarrollo de contenidos interactivos en 2D y 3D lo más accesible posible a tantas personas en todo el mundo como sea posible.

La primera versión de Unity se lanzó en la Conferencia Mundial de Desarrolladores de Apple en 2005. Fue construido exclusivamente para funcionar y generar proyectos en los equipos de la plataforma Mac y obtuvo el éxito suficiente como para continuar con el desarrollo del motor y herramientas.

Unity 3 fue lanzado en septiembre de 2010 y se centró en empezar a introducir más herramientas que los estudios de alta gama por lo general tienen a su disposición, con el fin de captar el interés de los desarrolladores más grandes, mientras que proporciona herramientas para equipos independientes y más pequeñas que normalmente serían difíciles de conseguir en un paquete asequible.

La última versión de Unity, Unity 5, lanzada a principios de 2015, se anunció en Game Developers e incluye añadidos como Mecanim animation, soporte para DirectX 11 y soporte para juegos en Linux y arreglo de bugs y texturas. Desarrollado por creadores de juegos para mayor expectativa.

# Gráficos

## Renderizado

La palabra renderización proviene del inglés render, y no existe un verbo con el mismo significado en español, por lo que es frecuente usar las expresiones renderizar o renderear.  El término **rendering** también es usado para describir el proceso del cálculo de los efectos en la **edición de archivos de videos (pre-renderizado)** para producir una salida final de video.

También es usado en los videojuegos solo que en este caso hablaremos de renderización en tiempo real, ya que los movimientos que pueda hacer el jugador no son predecibles y por esto se deberán generar en tiempo real con ayuda de una gpu para soportar la alta complejidad y cantidad de cálculos.

## Materiales y Shaders

El renderizado en Unity se hace con **Materials**, **Shaders** y **Textures**.

Hay una relación cercana entre Materiales, Shaders y las Texturas en Unity.

**Materials** son definiciones acerca de cómo la superficie debería ser renderizada, incluyendo referencias a texturas utilizadas, información del tiling (suelo de baldosas), tines de color y más. Las opciones disponibles para un material depende en qué shader del materia está utilizando.

**Shaders** son scripts pequeños que contienen los cálculos de matemáticas y algoritmos para calcular el color de cada pixel renderizado, basándose en el input de iluminación y la configuración del Material.

**Textures** son imágenes bitmap. Un material contiene referencias a texturas, para que el shader del Material puede utilizar las texturas mientras calcula el color de la superficie de un objeto. Adicionalmente al color básico (albedo) de la superficie de un objeto, las texturas pueden representar otros aspectos de la superficie de un material tal como su reflectividad o rugosidad.

Un material especifica un shader en ser utilizado, y el shader utilizado determina qué opciones están disponibles en el material. Un shader especifica una o más variables de textura que espera utilizar, y el inspector del Material de Unity le permite a usted asignar sus propios assets de textura a estas variables de textura.

Para la mayoría del renderizado normal - entendiendo por tal personajes, escenario, entorno, objetos sólidos y transparentes, superficies suaves y duras etc., el [Standard Shader](http://docs.unity3d.com/es/current/Manual/shader-StandardShader.html) por lo usual es la mejor decisión. Este es un shader altamente personalizado el cual es capaz de renderizar muchos tipos de superficie en una manera bastante real.

Hay otras situaciones dónde un diferente shader integrado, o incluso un shader escrito personalizado podría ser apropiado - tal como líquidos, follaje, vidrio de refracción, efectos de partícula, caricaturesco, ilustrativo u otros efectos artísticos, otros efectos especiales como visión nocturna, visión de calor, o visión de rayos x, etc.

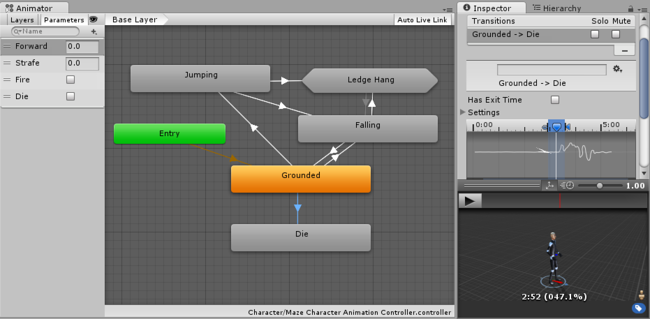
## Animación

En el caso de la animación haremos especial hincapié en la animación 3D ya que es la más compleja y en el caso de la animación 2D Unity nos proporciona la posibilidad de pasarle una hoja de scripts de un personaje 2D y automáticamente se crean las animaciones para los movimientos de este.

Unity tiene un sistema de animación excelente y sofisticado llamado Mecanim.

Mecanim proporciona:

* Un flujo de trabajo y configuraciones de animaciones fácil para todos los elementos de Unity incluyendo objetos, personajes, y propiedades.
* Soporta para [animation clips](http://docs.unity3d.com/es/current/Manual/class-AnimationClip.html) importados y animaciones creadas dentro de Unity
* Animación humanoide [retargeting](http://docs.unity3d.com/es/current/Manual/Retargeting.html) - habilidad para aplicar animaciones del modelo de un personaje a otro.
* Un flujo de trabajo simplificado para alinear clips de animación.
* Una pre-visualización conveniente de clips de animación, transiciones e interacciones entre estos. Esto le permite a los animadores funcionar de manera más independiente a los programados, dar prototipos y pre-visualizar sus animaciones antes de que el código del juego es conectado.
* El manejo de interacciones complejas entre animaciones con una herramienta visual de programación.
* Animar diferentes partes del cuerpo con diferente lógica.
* Características de capas (layering) y de masking



## Flujo de trabajo de la Animación

El sistema de animación de Unity está basado en el concepto de [Animation Clips](http://docs.unity3d.com/es/current/Manual/class-AnimationClip.html), los cuales contienen información acerca cómo ciertos objetos deberían cambiar su posición, rotación, u otras propiedades en el tiempo. Cada clip puede ser pensado como una sola grabación lineal. Los Animation clips de fuentes externas son creadas por artistas o animadores con herramientas de terceros como lo es Max o Maya, o proceden de estudios de captura de movimiento u otras fuentes.

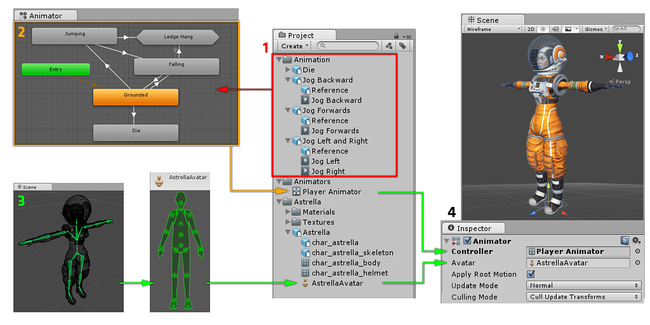
Los Animation Clips (Clips de animación) son organizados entorno a un sistema con una estructura similar a la del diagrama de flujo llamado Animator Controller. El Animator Controller funciona como un “[State Machine](http://docs.unity3d.com/es/current/Manual/AnimationStateMachines.html)” que mantiene un seguimiento de qué clip debería actualmente estar reproduciéndose, y cuando las animaciones deberían cambiar o mezclarse juntas.

Un simple Animator Controller podría solamente contener uno o dos clips, por ejemplo para controlar un powerup de girar y rebotar, o para animar una puerta que se abre y se cierra en el tiempo correcto.

Un más avanzado Animator Controller puede contener docenas de animaciones humanoides para todas las acciones de los personajes principales, y pueden mezclarse entre múltiples clips al mismo tiempo para proporcionar un movimiento fluido a medida que el jugador se mueve alrededor de la escena.

El sistema de animación de Unity también tiene un número de características especiales para manejar personajes humanoides los cuales le da a usted la habilidad de [retarget](http://docs.unity3d.com/es/current/Manual/Retargeting.html) una animación humanoide de cualquier fuente (Eg. captura de movimiento, el asset store, o alguna librería de animación de terceros) para su propio modelo del personaje, al igual que ajustar las [definiciones musculares](http://docs.unity3d.com/es/current/Manual/MuscleDefinitions.html). Estas características especiales están activadas por el sistema de [Avatar](http://docs.unity3d.com/es/current/Manual/class-Avatar.html) de Unity, dónde los personajes humanoide son mapeados a un formato interno común.

Cada una de estas piezas - los [Animation Clips](http://docs.unity3d.com/es/current/Manual/class-AnimationClip.html), el [Animator Controller](http://docs.unity3d.com/es/current/Manual/class-AnimatorController.html), y el [Avatar](http://docs.unity3d.com/es/current/Manual/class-Avatar.html), son traídos juntos a un GameObject vía el [Animator Component](http://docs.unity3d.com/es/current/Manual/class-Animator.html). Este componente tiene una referencia a un Animator Controller, y (si es requerido) el Avatar para este modelo. El Animator Controller, al contrario, contiene referencias a los [Animation Clips](http://docs.unity3d.com/es/current/Manual/class-AnimationClip.html) que utilizan.



El diagrama de arriba muestra lo siguiente:

1. Los animation clips (clips de animación) son importados de una fuente externa o creadas dentro de Unity. En este ejemplo, estas son de un movimiento capturado de animaciones humanoides.
2. Los clips de animación son colocados y arreglados en un Animator Controller. Esto muestra una vista de un Animator Controller en la ventana Animator. Los Estados (los cuales pueden representar animaciones o sub-estados de máquinas anidadas) aparecen como nodos conectados por lineas. Este Animator Controller existe como un asset en la ventana del Proyecto.
3. El modelo del personaje rigged (en este caso, el astronauta “Astrella”) tiene una configuración específica de huecos los cuales son mapeados en el formato Avatar común de Unity. Este mapeo es almacenado como un asset Avatar como parte del modelo del personaje importado, y también aparece en la ventana del Proyecto como es mostrado.
4. Cuando se anima un modelo del personaje, éste tiene un componente Animator adjunto. En la vista del Inspector mostrada arriba, usted puede ver el Animator Component que tiene ambos el Animator Controller y el Avatar asignados. El animator utiliza estos en conjunto para animar el modelo. La referencia del Avatar es solamente necesaria cuando se anime un personaje humanoide. Para otros tipos de animación, solamente un Animator Controller es requerido.

# Audio

## Recursos propios dentro del entorno

El sistema de audio de Unity es flexible y poderoso. Puede importar la mayoría de formatos estándares de audio y tiene características sofisticadas para reproducir sonidos en un espacio 3D, opcionalmente con efectos como echo y filtración aplicadas. Unity también puede grabar audio de cualquier micrófono disponible de la máquina del usuario para uso durante el modo de juego o para almacenamiento y transmisión.

En la vida real, los sonidos son emitidos por objetos y escuchados por listeners. La manera en la que el sonido es percibido depende de una cantidad de factores. Un listener puede decir en qué dirección un sonido viene y también puede tener algún sentido de su distancia por su intensidad y calidad



Para simular los efectos de la posición, Unity requiere que los sonidos sean originados de Audio Sources adjuntos a los objetos. Los sonidos emitidos luego son recogidos por un Audio Listener adjuntado a otro objeto, en la mayoría es la cámara principal. Unity puede simular los efectos de la distancia y la posición de una fuente del objeto listener y reproducirlos al usuario de acuerdo a esto.

Unity no puede calcular los echos solo con la geometría de la escena, pero usted puede simularlos al agregar Audio Filters a los objetos. Por ejemplo, usted puede aplicar el Echo Filter a un sonido que supuestamente viene dentro de una cueva. En situaciones dónde los objetos pueden moverse adentro y afuera de un lugar con un fuerte echo, usted puede agregar un Reverb Zone a la escena. Por ejemplo, su juego puede involucrar carros que se mueven a través de un túnel. Si usted coloca un reverb zone dentro del túnel, los sonidos de los motores de los carros va a comenzar hacer eco justo cuando entran y el echo morirá cuando ellos salgan al otro lado del túnel.

El Audio Mixer de Unity permite mezclar varias fuentes de audio, aplicarles efectos, y realizar una masterización

## Programación del sonido dentro del juego

El AudioMixer es un asset que puede ser referenciado por AudioSources para proporcionar un routing (direccionamiento) más complejo y mezcla de señales de audio generadas desde AudioSources.

Hace esta categoria basándose en la mezcla vía la jerarquía del AudioGroup que es construido por el usuario dentro del Asset.

Efectos DSP y otros conceptos de masterización de audio pueden ser aplicados a la señal del audio como si fuera direccionado desde el AudioSource al AudioListener.

Con objeto de facilitar la comprensión de la explicación recientemente expuesta en el trabajo añadimos la definición del concepto enroutamiento:

“El Enrutamiento de Audio es el proceso de tomar un número de señales de audio de input y output.”

El término señal aquí se refiere a un flujo continuo de datos de audio digitales, que se puede descomponer a canales de audio digitales (como estéreo o 5.1).

Internamente hay algo de trabajo en estas señales siendo hechas, tal como la mezcla, la aplicación de efectos, atenuación etc. Por varias razones que serán cubiertas, esto es un aspecto importante del procesamiento de audio y esto es lo que el AudioMixer está diseñado para permitirle a usted hacer.

# Recursos

## Formación externa

Cursos 1 trabajo

## Recursos multimedia externos

Unity posee un Marketplace de recursos multimedia desarrollados por la propia comunidad y profesionales del sector para poder comprar modelos 3D ya creados, escenarios, packs de texturas, etc…

# <https://www.assetstore.unity3d.com/en/#!/content/7065>

# Requisitos

En este apartado, expondremos los requisitos mínimos que debe cumplir nuestra Workstation para poder desarrollar usando como motor gráfico Unity y poder jugar o probar nuestro proyecto en tiempo real.

## Para desarrollo

**OS**: Windows 7 SP1+, 8, 10; Mac OS X 10.8+.

Windows XP y Vista no son compatibles y las versiones de servidor de Windows & OS X no se han probado.

**GPU**: Capacidades de tarjeta de vídeo con DX9 (modelo de shader 2.0). Se presupone que cualquier tarjeta de video posterior al 2004 cumple con los requisitos suficientes.

## Requisitos adicionales para el desarrollo de plataformas:

* iOS: Computadora Mac que ejecuta como mínimo la versión OS X 10.9.4 y Xcode 6.x.
* Android: Android SDK y Java Development Kit (JDK).
* Windows 8.1 Store Apps / Windows Phone 8.1: 64 bit Windows 8.1 Pro y Visual Studio 2013 Update 2+.
* WebGL: Mac OS X 10.8+ o Windows 7 SP1+ (solo editor de 64 bits)

## Para ejecutar juegos de Unity

Los requisitos necesarios variaran en función de la complejidad del proyecto pero podemos exponer como requisitos generales los siguientes.

* Escritorio:
  + OS: Windows XP SP2+, Mac OS X 10.8+, Ubuntu 12.04+, SteamOS+
  + Tarjeta de vídeo: capacidades DX9 (shader modelo 2.0)
  + CPU: compatible con el conjunto de instrucciones SSE2.
  + Web Player (obsoleto): Requiere un navegador que soporte plugins, como IE, Safari y algunas versiones de Firefox
* iOS: requiere iOS 6.0 o versiones posteriores.
* Android: OS 2.3.1 o posterior; CPU ARMv7 (Cortex) con tecnología NEON o CPU Atom; OpenGL ES 2.0 o posterior.
* WebGL: Cualquier versión de escritorio reciente de Firefox, Chrome, Edge o Safari
* Windows Phone: 8.1 o posterior
* Windows Store Apps: 8.1 o posterior

# Soporte Propio

## Documentación

Unity cuenta con mucha documentación gratuita y de calidad en internet pero haremos inciso en la proporcionada por ellos mismos:

<http://docs.unity3d.com/Manual/index.html>

Podemos destacar de este manual su completitud y frecuencia de actualización pues con cada nueva versión del software este es revisado y actualizado.

## Asistencia Técnica

Unity presenta 2 versiones de su producto, la gratuita (Si el fruto de nuestro desarrollo genera más de X beneficios deberemos pagar royalties a Unity de igual manera ) el servicio técnico está más centrado a cuestiones de mal funcionamiento de la aplicación, temas de compras de licencias y la de pago los cuales se llevan a cabo en mensualidades y es esta versión Premium la cual nos ofrece un servicio de asistencia técnica profesional.

Este programa de asistencia técnica es vendido con la premisa o estategia comercial de que tu como usuario no tengas que preocuparte por los problemas derivados de un mal uso o desconocimiento del software, que ellos de forma rápida y eficaz te ayudaran con tu problema para que te centres en lo realmente importante que es la creación del videojuego.

Cabe destacar que prometen un servicio 24/7 con diferentes canales de contacto y diferentes paquetes de asistencia técnica en función de tus necesidades.