Trabajo repartido:

Pareja 1: Lili y Javi UNITY

Pareja 2: Kevin y Laia UNREAL

Apartado 1: descripción 🡪 historia del motor y descripción general

Apartado 2: categorías:

1. DISEÑO
   1. Interfaz

Análisis de la interfaz del programa

* 1. Recursos predefinidos

Material, assets

1. CARACTERÍSTICAS GENERALES
   1. Multiplataforma

Capacidad de adaptar el mismo desarrollo a distintos entornos

Consolas 🡪 kit de desarrollo

* 1. Licencia

1. RENDIMIENTO
   1. Optimización

Herramientas del entorno para optimizar

* 1. Escalabilidad

Capacidad de cambiar los gráficos

1. UTILIDAD
   1. Compatibilidad con programas externos
2. PROGRAMACIÓN
   1. Lenguaje
   2. API

Librerías, scripts

1. GRAFICOS
   1. Renderizado
   2. materiales y shaders
   3. Animación 2D y 3D
2. SONIDO
   1. Recursos propios dentro del entorno
   2. Programación del sonido dentro del juego
3. RECURSOS
   1. Formación externa
   2. Recursos multimedia externos

Tiendas etc

1. HARDWARE
   1. Requisitos mínimos para ejecutar el entorno
   2. Requisitos mínimos para ejecutar el juego
2. SOPORTE PROPIO
   1. Documentación
   2. Asistencia técnica

5. Programación

Lenguaje

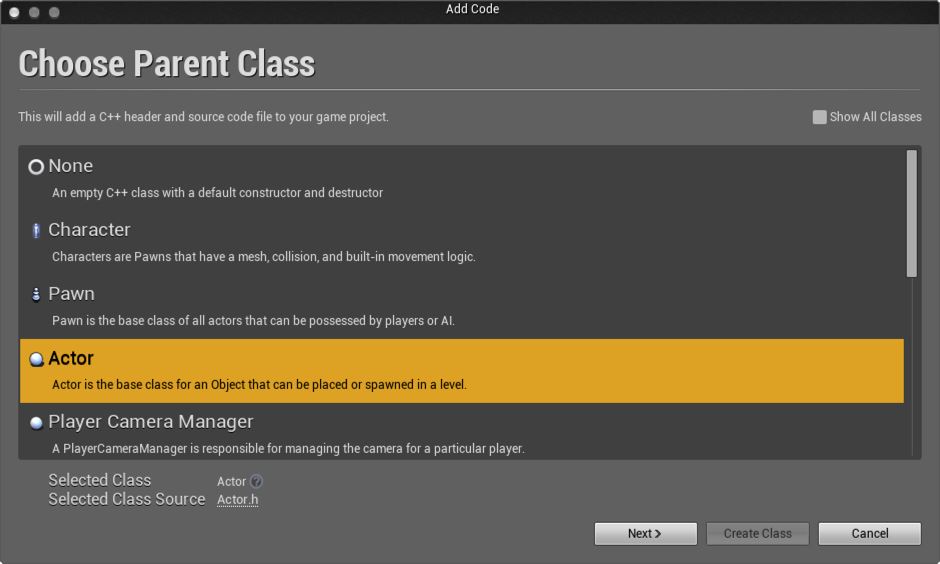


El lenguaje de programación utilizado para escribir el código del juego es C++, dicho lenguaje nos permite controlar de forma muy minuciosa la gestión de memoria dentro del desarrollo de nuestro juego. No obstante, es un lenguaje mucho más complicado que otros lenguajes orientados a objetos como pueden ser C# o Java.

Unreal Engine nos permite utilizar código C++ plano, sin depender de las librerías que nos proporciona el motor, no obstante en caso de que decidamos utilizar estas, la programación se simplificará enormemente debido a la cantidad de métodos que ya trae incorporados. Dichas librerías se explicarán en un punto posterior.

En adición al lenguaje de programación C++, Unreal Engine, nos trae un sistema de scripting visual llamado Blueprint Scripting, el cual nos permite realizar tareas de programación desde un entorno visual con solo clickar y arrastrar nodos. Aunque este no sea un lenguaje de programación al uso, debemos hacerle mención ya que suele ser usado ampliamente para facilitar las tareas de los programadores al realizar tareas más sencillas o bien los propios programadores pueden programar scripts para ser usados en este lenguaje y facilitar que el resto del equipo pueda “programar” ciertas partes de la aplicación de forma autónoma.

API:



El API de Unreal Engine nos proporciona una gran variedad de clases predefinidas las cuales podemos extender, además de poder crear una clase en C++ vacía, también tenemos la posibilidad de crear clases con distinta funcionalidad base pre implementada para que después nosotros podamos modificarlas o extenderlas.

Entre las clases predefinidas podemos destacar las siguientes:

* Actor: es la clase base de objetos, se usa para cualquier objeto que pueda ser colocado en un nivel, es decir, objetos con un cuerpo físico no son clases lógicas.
* Pawn/Peón: es la clase básica de objetos que van a usarse, ya sean por medio del jugador o por una IA.
* Character/Personaje: es una clase que desciende de la clase peón pero que tiene preconstruido una malla de renderizado, un detector de colisiones y un sistema de movimiento.
* Player Camera Manager: es una clase que es la responsable de gestionar la cámara a la que está asociada.
* Game Mode: es una clase que nos permite programar y gestionar la lógica del juego.
* HUD: esta clase nos permite programar la interfaz que se va a mostrar al usuario durante el juego.

Además de las distintas clases precreadas podemos destacar una serie de métodos que son transversales a todas las clases:

* AMyActor(); (sirve para poner valores por defecto al actor en cuestión)
* Virtual void BeginPlay() override; (se usa para establecer lo que queramos que haga cuando el objeto entra el juego o cuando se carga la partida)
* Virtual void Tick (float DeltaSeconds) override; (la función se llama a cada frame del juego)

6. Graficos.

Renderizado:



El sistema de renderizado de Unreal Engine está basado en DirectX 11 lo que incluye Sombreado diferido, Iluminación global, translucidad ligera y simulación de partículas mediante campos vectoriales postprocesados por la GPU.

Sombreado diferido:

Todas las luces son aplicadas en diferido en Unreal Engine 4. Los materiales tienen escritos sus atributos en el código lo que influye como la luz pasa a través de ellos e iluminan el escenario.

Iluminación global:

El sistema de Iluminado de Unreal Engine 4 se basa en varias herramientas que podemos utilizar para iluminar el mundo que varían en función de la calidad y el rendimiento que busque el desarrollador. Dichas luces pueden ser totalmente dinámicas, parcialmente estáticas y totalmente estáticas.

Translucidad ligera:

La translucidad emite sombras provenientes de su propio efecto y de otras iluminaciones además de garantizar el correcto fusionado de las distintas luces.

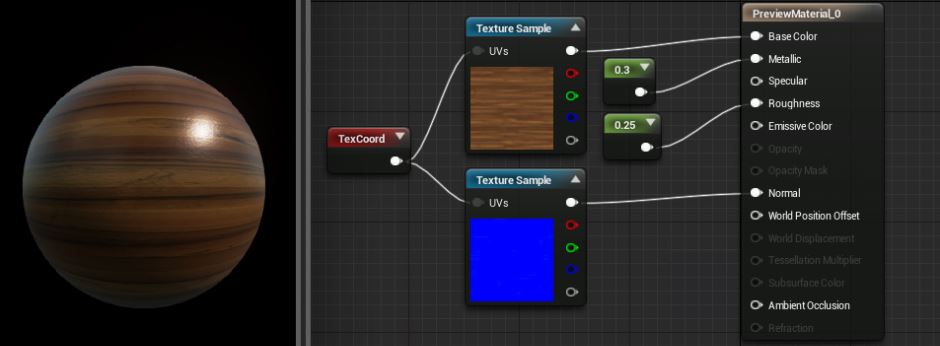
Partículas simuladas por GPU:

Unreal Engine 4 simula partículas en la GPU, comparado a la simulación tradicional que se carga sobre la CPU permite una simulación de cientos de miles de partículas comparadas con simplemente unas pocas miles de la simulación tradicional.

Efectos de PostProcesado:

Unreal Engine cuenta con varios efectos de post procesado que permiten a los diseñadores alterar el ambiente de la escena. Entre estos efectos se incluye: Oclusión Ambiental, Luces de Ambiente, Adaptación Ocular, Flashes, etc.

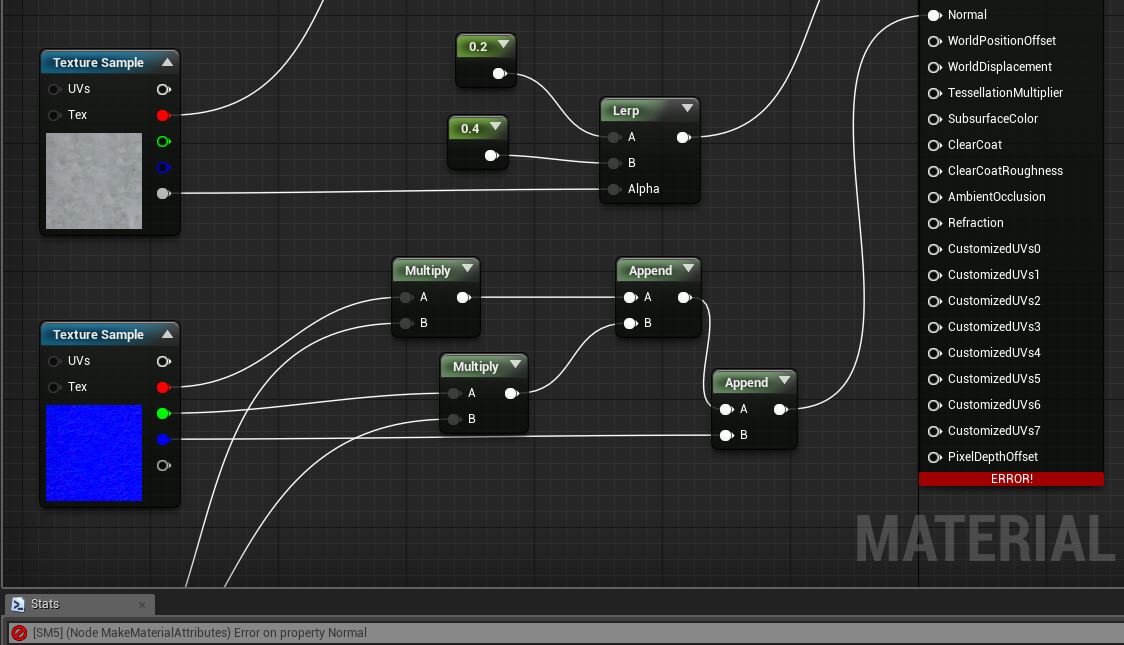
Materiales:



Un material es un objeto que puede aplicarse a un modelo para controlar como se ve en la escena. A grandes rasgos podemos decir que un Material es como la pintura que se aplica a un objeto. Un material define el tipo de superficie de la cual nuestro objeto va a aparentar estar hecho. Entre sus propiedades puedes definir su color, su brillo, su transparencia, etc.

Cuando la luz de la escena impacta en la superficie del objeto, es el material lo que se usa para realizar los cálculos de cómo reaccionará la luz con la superficie. Esto se hace mediante expresiones matemáticas e imágenes.

Unreal Engine nos permite construir materiales mediante un Sistema de scripting visual en el que, a través de una red de nodos, dentro del editor de materiales, seleccionamos sus propiedades. Este sistema de programación visual se denomina HLSL, el cual nos permite seleccionar cada parámetro del material individualmente para construirlo justo como lo queremos. La única desventaja de este editor de materiales es su alta complejidad ya que para aprender a usarlo deberemos mirar el manual y algunos cursos acerca de esta funcionalidad.



Animación:

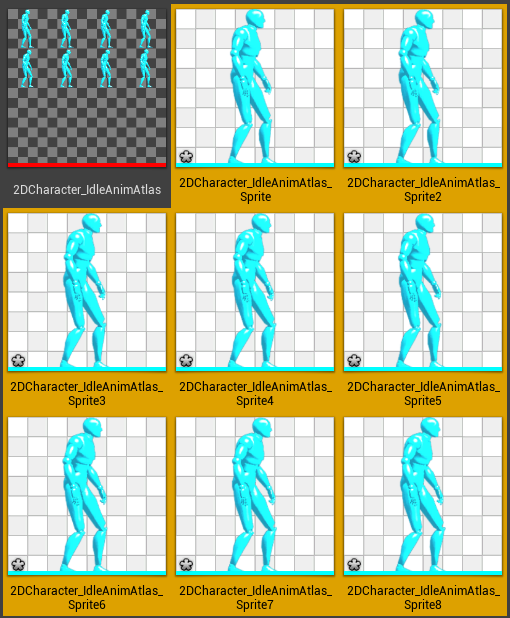


El Sistema de Animación de Unreal Engine, mezcla deformaciones basadas en esqueleto con deformaciones basadas en vértices para crear complejas animaciones. Este sistema puede hacer que movimientos básicos del jugador, se vean mucho más realistas reproduciendo y fusionando secuencias de animación, creando movimientos especiales personalizados como movimientos de escalada, aplicar efectos de daño e impacto e incluso crear expresiones faciales o controlar directamente la transformación de los huesos del modelo.

El motor nos permite controlar y configurar las animaciones de los personajes dentro del mismo, estableciendo los patrones de animaciones mediante programación con BluePrints o mediante un completo editor de animaciones. Dicho editor nos da las herramientas para crear nuestras propias animaciones a partir del modelo del personaje. Una vez que tenemos el modelo dentro del editor vamos moviéndolo y guardando las poses que queremos que compongan la animación, esto nos permite crear las animaciones de forma eficiente.



En cuanto a la animación 2D, Unreal nos permite crear animaciones mediante hojas de sprites. Dichas hojas contendrán todos los dibujos del personaje en las distintas posiciones, los cuales se unirán todos mediante el editor de sprites que nos proporciona unreal para crear la animación.



7. Sonido

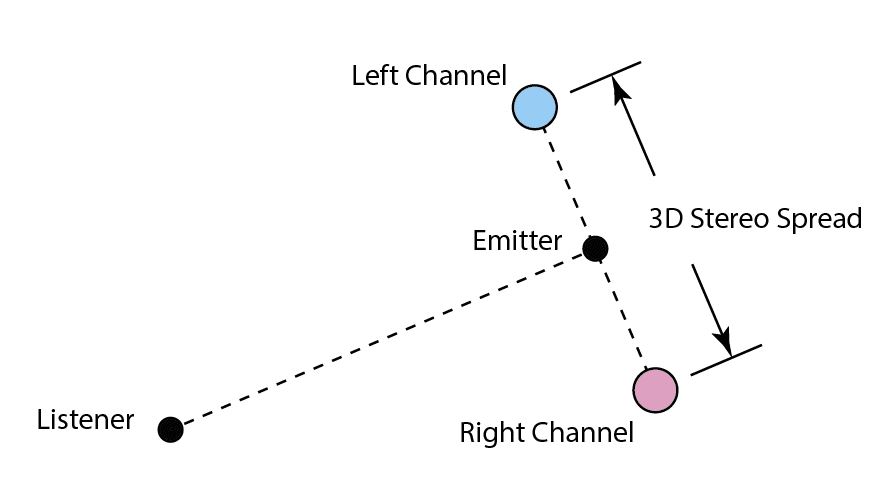
Recursos propios dentro del entorno:



El Sistema de audio de Unreal Engine 4 está compuesto de varios componentes, cada uno de ellos trabajando en conjunto con todos los demás para construir la experiencia de audio que experimentará el jugador. Cuando importamos un archivo de audio en el motor y lo colocamos en el nivel, tenemos varias opciones para personalizar, desde las básicas como ajustar el volumen o el tono hasta algunas más complejas como la atenuación del sonido.

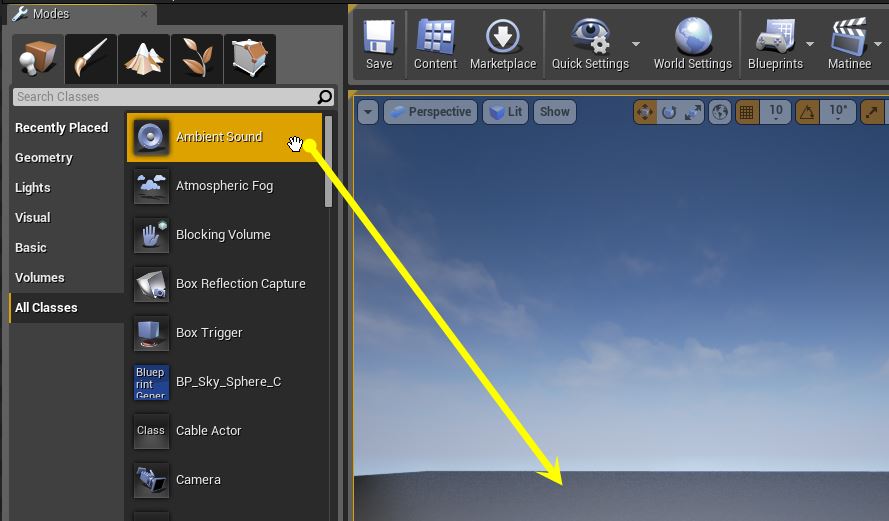
Además en Unreal Engine 4, podemos crear sonidos compuestos en forma de señales de audio mediante un editor de señales de audio que trae incorporado el propio motor. En el podemos combinar sonidos mediante nodos igual que hacíamos con los materiales para conseguir el resultado deseado.

En las últimas versiones de Unreal Engine, podemos encontrar que se ha añadido una mejora al sistema de audio. Dicha mejora consiste en la compatibilidad de sonido estéreo en los objetos del juego, antes cada objeto dentro del juego podía emitir solo un sonido. Con el nuevo sistema, un mismo objeto podrá emitir dos sonidos de forma simultánea los cuales se percibirán en estéreo por el jugador.



Programación del sonido:

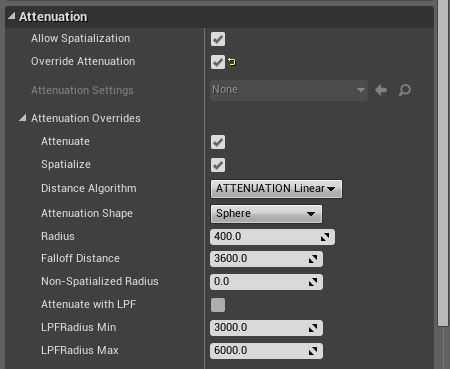
Unreal Engine 4 maneja el proceso por cual podemos producir o modificar sonidos a traves del uso del Actor de Sonido. Cuando una pista de sonido o una señal de sonido es colocada en el nivel, un Actor de Sonido con ese sonido es creado.



El actor de sonido toma un gran papel a la hora de transmitir al jugador ya que reproducirá el sonido con más o menos volumen en función de la distancia a la que nos encontremos de la fuente del sonido.

Una vez que colocamos el sonido podemos configurarlo de múltiples formas en función de nuestras necesidades:

* Si necesitamos que se empiece a reproducir nada más comience el juego, dejaremos la función de AutoPlay activada.
* También podemos hacer que el sonido se reproduzca en bucle en función de lo que hayamos establecido en la opción looping.
* Por último, podemos configurar una gran cantidad más de parámetros para que el sonido se adapte a nuestras necesidades.



8. Recursos:

Formación Externa:







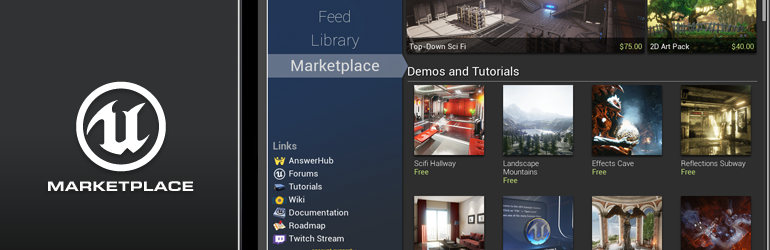


Como hemos indicado ya en otros trabajos, hay mucha formación en internet acerca de Unreal Engine, no obstante, cabe destacar que la mayor parte de esa formación es formación de pago, es decir, cursos de pago. Entre los proveedores que ofrecen cursos de pago para Unreal Engine, podemos destacar famosas empresas especializadas en formación como Digital Tutors, Udemy o Lynda.com.

Aunque haya mucha formación no gratuita acerca de Unreal Engine en Internet, este motor cuenta con una gran comunidad detrás y si buscamos un poco podemos encontrar foros, cursos en plataformas de videos como Youtube, blogs especializados, etc.

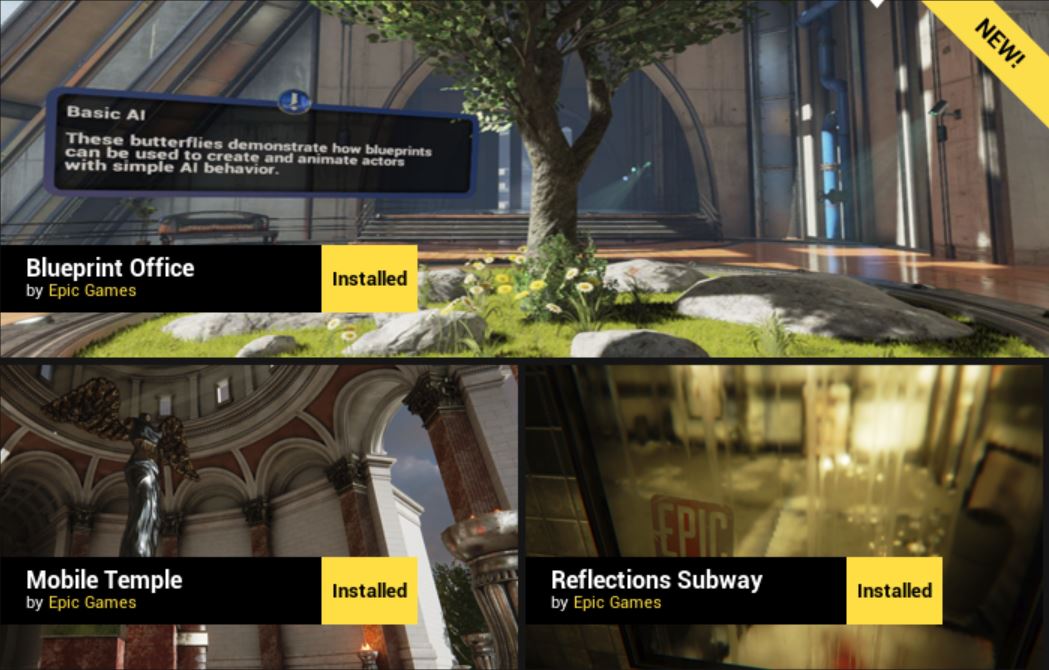
Aparte de la formación que podamos recibir online, en casi todos los másteres universitarios dedicados al desarrollo de videojuegos se suele aprender a manejar este motor debido a su alta popularidad en el mercado. Esto no solo se aplica a los másteres sino también a los grados especializados en el desarrollo de videojuegos siendo más notable su enseñanza en las academias privadas.

Recursos Multimedia Externos:



Unreal Engine cuenta con su propio Marketplace donde la comunidad sube sus materiales para que cualquiera que esté interesado pueda comprar dicho material y añadirlo a su proyecto. En el Marketplace podemos encontrar material de todo tipo, desde objetos individuales hasta packs temáticos con una gran variedad de objetos agrupados. Los precios van desde artículos gratis hechos por la propia unreal o por la comunidad que los pone gratuitamente bajo licencias creative commons, hasta paquetes con muchos objetos a un precio moderado.

No obstante, no solo podemos encontrar recursos en el Marketplace que nos ofrece unreal, también podemos encontrar música de uso libre en sitios como youtube o personajes, fondos y demás piezas de arte en webs como opengameart.org.



9. Hardware:

Requisitos del Motor.

Unreal Engine tiene algunos requisitos específicos de hardware y software para ejecutar el juego el editor para el desarrollo.

## Hardware Recomendado

|  |
| --- |
| **Sistema Operativo** | Windows 7/8 64-bit |
| **Procesador** | Quad-core Intel or AMD, 2.5 GHz or faster |
| **Memoria** | 8 GB RAM |
| **Tarjeta de video/DirectX Version** | DirectX 11 compatible |

## Requisitos Mínimos de Software

A continuación, están los requisitos mínimos para ejecutar el motor o el editor.

| **Ejecutar el Motor** | |
| --- | --- |
| **Sistema Operativo** | Windows 7/8 64-bit |
| **DirectX Runtime** | [DirectX End-User Runtimes (June 2010)](http://www.microsoft.com/en-us/download/details.aspx?id=8109) |

Los requisitos para los programadores que vayan a usar el motor para desarrollar están listados a continuación.

| **Desarrollo con el motor** | |
| --- | --- |
| **Requisitos necesarios para 'Ejecutar el motor' (Instalados automáticamente)** | |
| **Visual Studio Versión** | Visual Studio 2015 Professional or Visual Studio 2015 Community:   * [Pro](https://www.visualstudio.com/downloads/download-visual-studio-vs) * [Community](https://www.visualstudio.com/en-us/products/visual-studio-community-vs.aspx) |
| Desarrollo iOS | |
| **iTunes Versión** | [iTunes 11 or higher](http://www.apple.com/itunes/) |

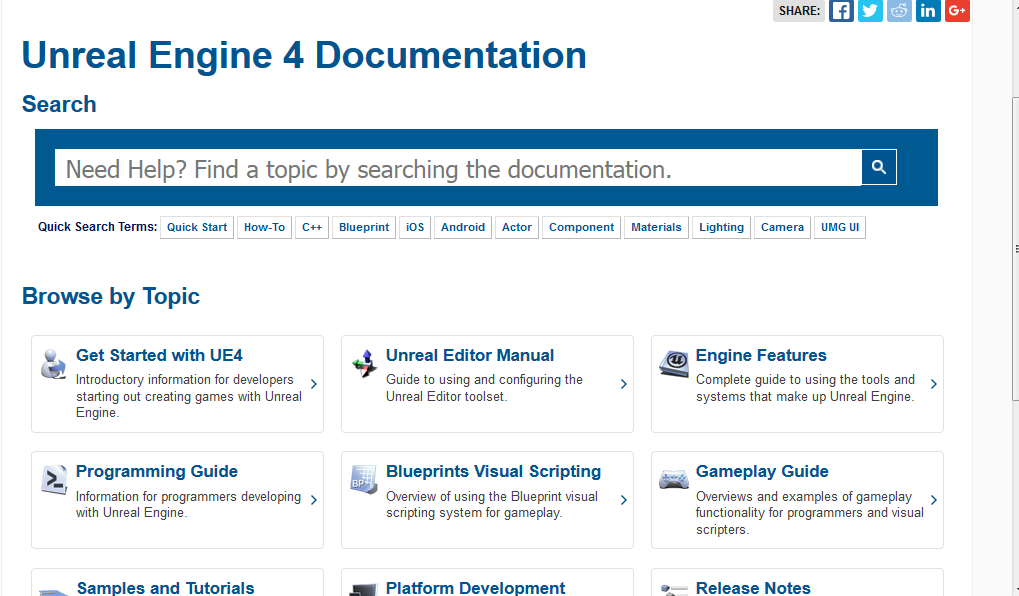
Requisitos de las aplicaciones construidas:

Para ambos motores 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

10 Soporte Propio:

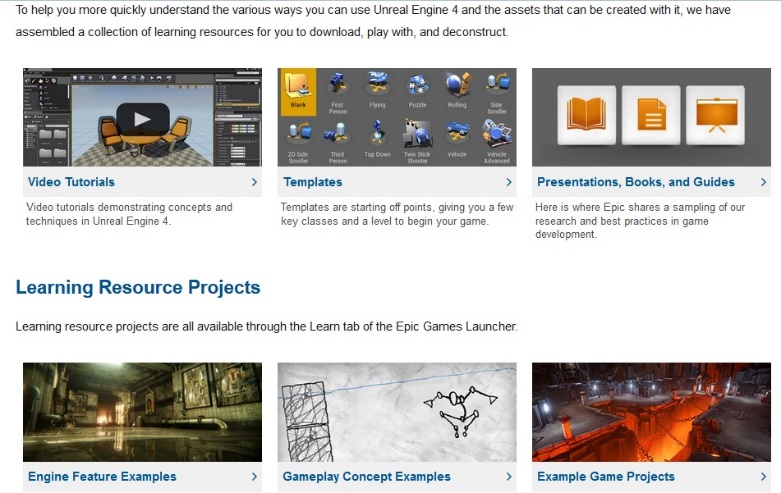
Documentación:



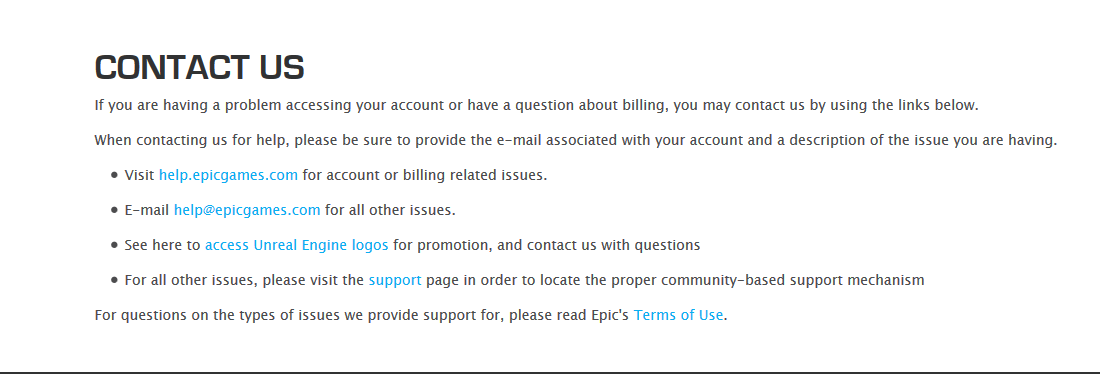
La documentación que nos proporciona Epic Games sobre Unreal Engine es amplia y completa. Con solo leernos la documentación podemos aprender lo suficiente para utilizar el motor con eficacia. Dicha documentación es muy extensa ya que abarca en profundidad todas las características que posee el motor.

Además de toda la documentación detallada, podemos acceder a unas guías rápidas las cuales sirven para que si ya sabemos manejarnos un poco con versiones anteriores del motor o si por ejemplo venimos de haber trabajado con otro motor distinto (como Unity por ejemplo) podamos ponernos al día de forma rápida y sencilla sin complicarnos demasiado.

Por último, en la documentación podemos encontrar una serie de tutoriales guiados que nos indicarán como movernos por el motor y nos enseñarán a desenvolvernos por el de forma satisfactoria. Aparte de estos tutoriales también tenemos bastantes ejemplos funcionales de cómo podemos hacer proyectos para que los veamos y aprendamos de ellos.



Asistencia Técnica:



Además de todo lo mencionado en el tema de la documentación, Epic Games, cuenta con un equipo de soporte para Unreal Engine 4 que ofrece servicio a estudios que necesiten modificar el motor para algún proyecto específico o que necesiten algún tipo de licencia personalizada.

Para los usuarios normales, el soporte, se encarga de tareas relacionadas con la seguridad y el acceso a las cuentas de los usuarios, problemas de facturación, promoción y marketing, etc.

En caso de que tu consulta no esté entre lo que se menciona en la página de soporte te proporcionan un enlace hacia otra página para tratar tu solicitud de forma más especializada.

Por último, especifican que los tipos de incidencias a los que ellos proporcionan soporte se encuentran especificadas en los términos de uso.