Alumno: Gandía Jordà, Francisco Tutor: Alamán Garcera, Jorge

MEMORIA TRABAJO FIN DE GRADO:

Estudio de las nuevas **técnicas de renderizado en tiempo real** y sus posibilidades.

Escuela Técnica Superior de Ingeniería del Diseño

ÍNDICE

INTRODUCCIÓN

Visualización 3D realista en tiempo real, la motivación para realizar un proyecto de estas características empezó al ver de lo que eran capaces los nuevos motores gráficos. Pero uno no se decanta por realizar un estudio acerca de estas tecnologías, si no se dispone de medios, que por desgracia años atrás eran sumamente caros, inasumibles para un estudiante. Por suerte las empresas dedicadas al desarrollo de los motores gráficos en tiempo real cambiaron su modelo de negocio a pago de royalties e hicieron la tecnología accesible a todo el mundo.

Si buscamos una razón aplicada al diseño para la cual realizar un estudio sobre las nuevas posibilidades de los motores de renderizado en tiempo real, salta a la vista cúales son las ventajas o las desventajas de estos procedimientos frente a procedimientos más habituales cuando se habla de representación arquitectónica o de producto. El tiempo real siempre será mucho más dinámico permitiendo al cliente pasear virtualmente por un edificio, examinar un producto rotándolo y ampliándolo, o incluso ver cómo funciona de forma interactiva mediante animaciones o usando realidad virtual o aumentada. Cercanía con dicho producto o escenario es lo que sentirá el cliente, claro que siempre sacrificando algo de realismo y calidad gráfica inalcanzable a día de hoy por este tipo de software. Aunque es verdad que hoy por hoy, existen casos de uso de estas tecnologías por parte de infografistas, resulta poco común, lo que indica que estamos en una fase de adopción temprana. Por ese motivo se hace necesario realizar un estudio a nivel técnico acerca del uso de estas tecnologías, intentando maximizar las ventajas, haciendo experiencias lo mas dinámicas posibles, y minimizar las desventajas, intententando conseguir un acabado lo más realista posible.

Buscando facilitar la forma en la que se consiguen acabados gráficos realistas ingenieros de diversas ramas crearon los sistemas de renderizado físicamente correcto o basados en la física de los materiales conocido como PBR (Physically Based Rendering) o PBS (Physically Based Shading), en un intento por conseguir que los materiales tuvieran propiedades visuales físicas y que de esa manera respondieran correctamente fuera cual fuera el entorno en el que fueran implantados pudiendo reutilizar de forma más dinámica los materiales producidos con anterioridad. Rápidamente algunos sectores gráficos se dieron cuenta de la gran ventaja que esto suponía, invirtiendo así gran cantidad de recursos en la creación de estas herramientas. Pero PBR no es solo un conjunto de estándares en el shading gráfico, supone también una adopción de estándares físicos para los materiales en cuestión. Como explicaremos más adelante los materiales tienen ciertos canales con mapas, dichos canales tendrán (en el PBR) que cumplir unos estándares para según que material queramos crear. Con lo que algunos canales tendrán que cumplir estándares de forma forzosa y en otros se tendrá más libertad creativa.

Antes de entrar en materia vamos a ser específicos, para este trabajo se ha elegido trabajar con Unreal Engine 4 y Marmoset Toolbag 2. No son los únicos programas que soportan PBR y los motivos por lo cuales se han elegido estos programas serán explicados más adelante. Pero ya adelanta que la teoría de PBR expuesta a continuación es la propia de este Motor Gráfico. Y para explicar su modelo basado en el modelo PBS de Disney acudimos al material que mostraron el la conferencia del Siggraph de 2013 llamada “Real Shading in Unreal Engine 4” por Brian Karis de Epic Games.

Para la realización de este proyecto vamos a sentar ciertas bases que nos acompañaran a lo largo de toda su realización:

-Monográfico. Este trabajo fin de grado analizará con profundidad la oferta de nuevos medios para la visualización de productos, originalmente creados para el desarrollo de videojuegos, abordando el tema central, núcleo del proyecto, que es el motor de render. Bordeando aspectos intrínsecos del proyecto con el fin de rebajar la carga superflua, que aunque importante, no es materia de estudio en esta ocasión, y por lo tanto dicha materia gozará de mínima representación.

-Avanzado. Se parte de la base que los interesados en adoptar estas nuevas maneras de visualización 3D, son técnicos con experiencia previa en motores de render Biased o Unbiased, ya sean VRay, Mental Ray, Corona o Keyshot. Y por lo tanto son conocedores del flujo de trabajo previo a la iluminación 3D, a saber, modelado 3D y texturizado, del cual solo comentaremos algunos aspectos pero en ningún caso se profundizará al respecto. Por esta misma razón no será habitual explicar conceptos que debieran supuestamente ser conocidos por los técnicos, que en principio son los interesados en la lectura de este TFG.

-Actual. Para la realización de este proyecto se han consultado únicamente fuentes de documentación en línea, rechazando libros y revistas sobre el tema por una buena razón. Al tratarse de una tecnología extraordinariamente joven toda la documentación disponible se distribuye online a través de canales oficiales, de esta manera obtienes información nueva que sabes a ciencia cierta que aún no ha quedado obsoleta. En nuestro caso, el caso práctico propuesto fue en un par de ocasiones readaptado debido a las actualizaciones de los sistemas, demostrando la constante evolución del motor de render. De hecho muy probablemente la información contenida en este TFG quede obsoleta en menos de un año.

-Realista. Este trabajo pese a ser una investigación, se ha optado por ejecutar un caso práctico, el cual nos ha servido para discernir una metodología de trabajo que pudiera ser adoptada por un equipo de infografistas para la producción de espacios virtuales, bien para arquitectura o diseño, o incluso videojuegos o otros campos gráficos. Dicho caso práctico planteado podría ser vendido en tiendas virtuales ya que contiene ficheros útiles para la realización de otros proyectos. Y aunque la perspectiva de este TFG es la de difundir información deviene en un compendio muy valioso para quienes deseen iniciarse en estas técnicas de producción y además tengan una comprensión limitada del inglés, ya que prácticamente el 100% de las fuentes consultadas están en dicho idioma.

Dadas las bases con las que se decidió partir para este trabajo debíamos elegir un soporte tecnológico sobre el cual trabajar, para ver de primera mano el funcionamiento de dicha tecnología, y aunque la elección fue muy fácil a continuación pasamos a exponer el elenco de posibles plataformas.

Advertencia: Pese a estar escrito en un tono impersonal y buscar el contenido objetivo, advierto que la materia estudiada está tan sujeta al conocimiento tecnológico de la misma, que existen algunos asuntos con muchísimas maneras de abordarlos, muy diferentes entre sí, y que probablemente todos sean válidos, por ese motivo, cuando hay resquicio de valoración personal puede que el trabajo no se haya abordado de la manera más óptima, ofreciendo soluciones obsoletas.

TÉCNICAS ACTUALES.

NO ES UNA CUESTIÓN DE COMPETENCIA. POR AHORA.

Se pretende comprobar la validez de los motores de renderizado en tiempo real para el render de producto y la infoarquitectura, así que no se trata de ver qué método es mejor. Hasta ahora los motores de render Biased y Unbiased habían sido la única manera de obtener imágenes fotorrealistas de productos diseñados, pero aún no fabricados, o imágenes de lugares poblados de cosas existentes pero que aún no se habían construido.

Motores Biased, estos motores cuentan con parámetros para absolutamente todo, son muy difíciles de dominar, pero una vez se hace, no tienen rival cuando se trata de obtener resultados rápidos y de buena calidad.

VRAY

BRAZIL

MENTAL RAY

Motores Unbiased, en estos motores, configurar la iluminación resulta mucho más sencillo, aunque, por contra, configurar sus materiales presenta mayor complejidad. Su principal desventaja es que tardan muchísimo en renderizar, por eso algunos incluso tienen tiempo de render infinito, y se paran por el usuario cuando la cantidad de ruido presente en la imagen es menor y deseable.

INDIGO

FRYRENDER

MAXWELL RENDER

Existen otras alternativas como los motores ambivalentes que combinan CPU y GPU para renderizar, o solamente GPU. Introducido esto vamos a empezar con la nueva alternativa a estos motores.

PRESELECCIÓN DE UN CASO PRÁCTICO.

Con el objetivo de poder visualizar productos o interiores en tiempo real y con la mayor calidad posible se tuvo que escoger entre la pequeña oferta de desarrolladores que ofrecían su software. Originalmente pensados para el desarrollo de videojuegos la calidad es tal en comparación con la generación tecnológica anterior que esta vez sirve para nuestros propósitos.

POR JOSHUA KINNEY. DIGITAL TUTORS. MARMOSET TOOLBAG 1.

POR LOGITHX. POLYCOUNT. UNREAL ENGINE 3.

POR EA DICE. ELECTRONIC ARTS. FROSTBITE ENGINE 2.

En las imágenes anteriores podemos ver acabados punteros de la generación anterior de motores gráficos donde conseguirlos suponía un gran esfuerzo, no obstante ya se gestionaban con un excelente motor de iluminación.

Para entender qué supone exactamente un motor PBR, en el próximo apartado expondremos un estado del arte acerca de Unreal Engine 4 y su interpretación del PBR basado en el de Disney[1]. Pero antes listaremos los motores que fueron candidatos para el proyecto y explicaremos por qué finalmente se eligió el Unreal Engine 4 de Epic Games.

1: PDF explicativo del Shader PBR de Disney. Primer LINK: https://www.google.es/#q=disney+pbr

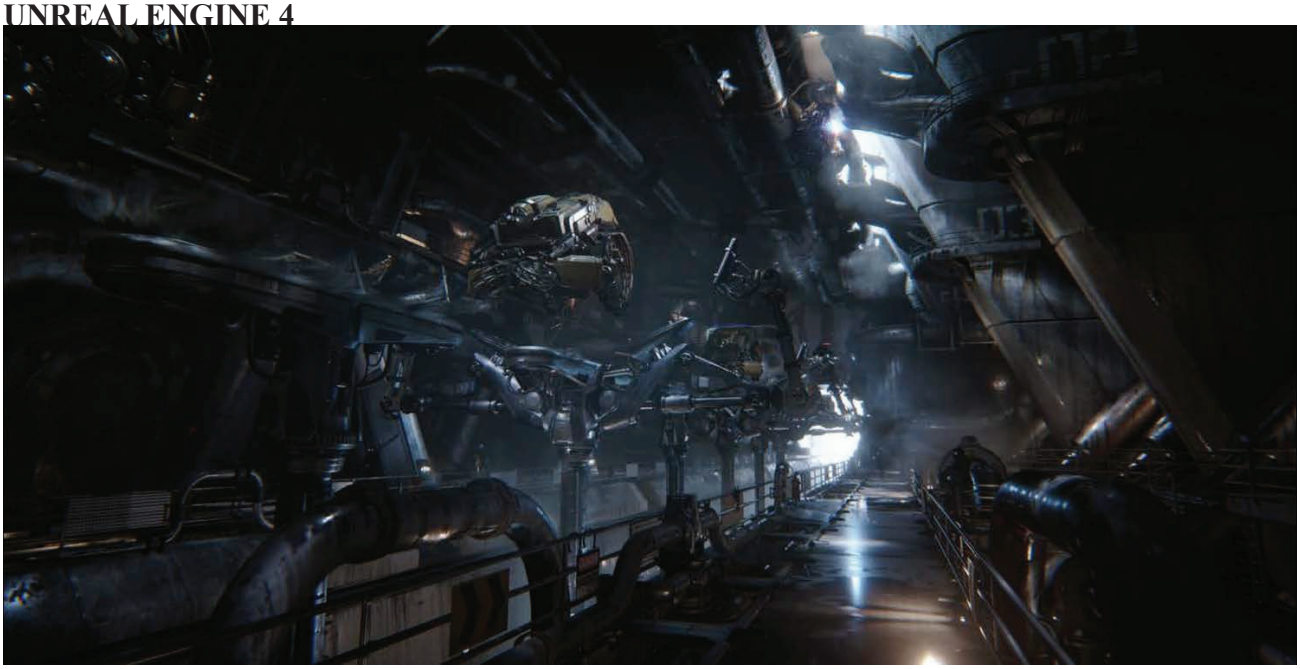
Motores utilizados en el caso práctico:

Cabe destacar que existen numerosos motores gráficos que nos valdrían como el Snowdrop[1], Frostbite[2], Luminous engine[3], Fox engine[4], etc... Pero todos estos motores son de uso privado para la explotación económica de sus respectivas compañías o bien para la explotación de terceros tras el pago de carísimas licencias. Este solía ser el modelo de negocio habitual en el sector de los videojuegos, pero hace años Epic Games propició el cambio con el lanzamiento del UDK, una versión recortada del Unreal Engine 3 totalmente gratuita[5] que ya anticiparía las tendencias del sector.

Durante el 2014 se anunció que el nuevo Unreal Engine 4, el CryEngine y el Unity 5, saldrían con modelos de negocio asumibles para desarrolladores independientes que irían desde la suscripción mensual, gratuito a cambio de royalties o el pago de una licencia PRO que ahorraba el pago de dichos royalties. Aunque a inicios del 2015 Unreal 4 y el Unity 5 se pasaron al modelo gratuito a cambio del pago de royalties para según que tipo de explotación comercial.

Así que eran estos tres motores los candidatos a ser puestos a prueba en este pequeño estudio.

UNREAL ENGINE 4



POR EPIC GAMES. INFILTRATOR DEMO. SIGGRAPH 2013.

El Unreal Engine 4, es el motor elegido para este trabajo. Se ha decidido no entrar en discusión sobre qué motor resulta superior a otro en potencia, que eso requeriría un análisis muy exhaustivo, pero el ser gratuito sumado a su interfaz intuitiva ha decantado finalmente la elección a este formidable motor.

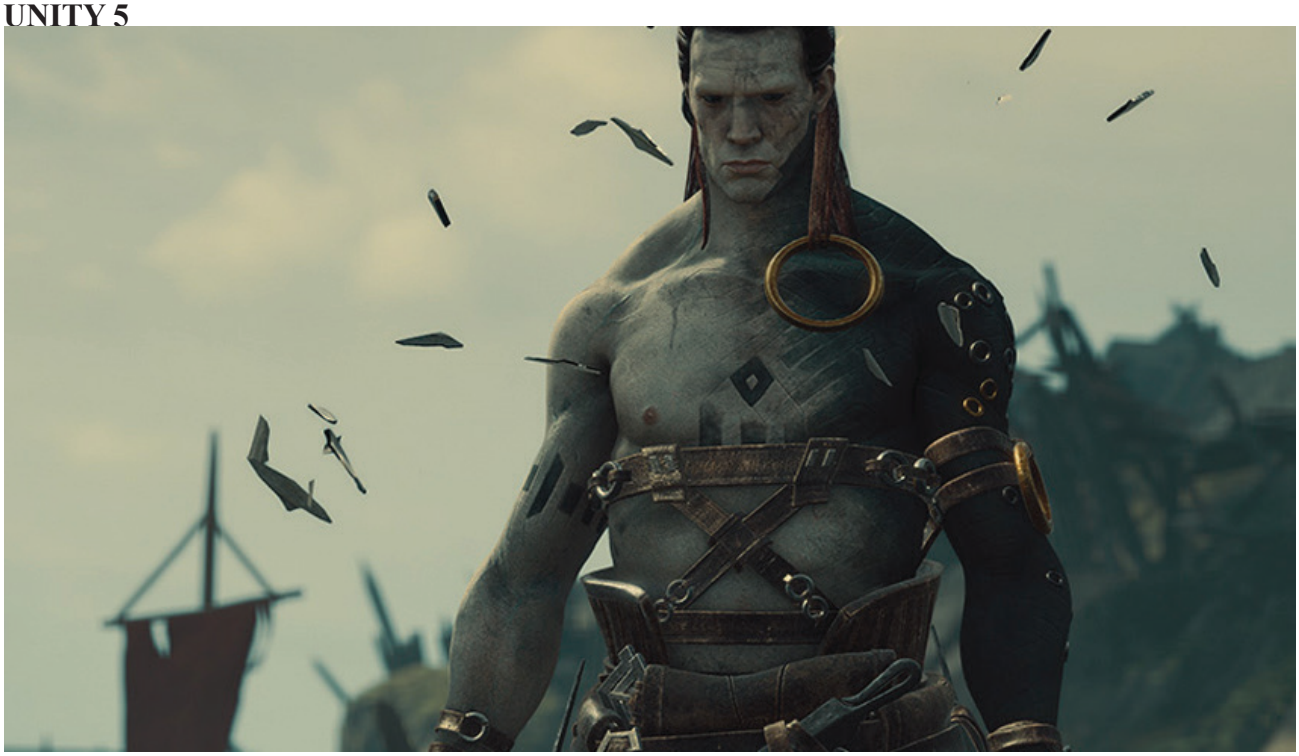
1: Motor gráfico de la compañia Franco-Canadiense Ubisoft. LINK: http://blog.ubi.com/the-division-getting-creative-with-snowdrop/
2: Motor gráfico del estudio Sueco EA Dice. LINK: http://www.frostbite.com/
3: Motor gráfico de la editora Japonesa Square-Enix. LINK: http://www.agnisphilosophy.com/en/
4: Motor gráfico de la editora Japonesa Konami. LINK: http://www.konami.jp/
5: UDK, motor previo al UE4. LINK: https://www.unrealengine.com/previous-versions

CRYENGINE 3

POR CRYTEK. RISE SON OF ROME. MICROSOFT.

El Cryengine, otro motor de gran musculo técnico, estandarte de la ingeniería gráfica europea. Este motor se ha rechazado principalmente porque la incertidumbre inicial del proyecto sobre su dificultad, priorizaba el uso de motores cuanto más baratos mejor. Este motor ya ha eliminado completamente los mapas de luz[1], usados en UE4 y usados en este trabajo, por suponer una desventaja en ambientes de iluminación cambiante[2]. Si el dinero no hubiese sido a un problema, este habría sido a priori la mejor elección.

UNITY 5



POR UNITY TECHNOLOGIES. THE BLACSMITH, GDC DEMO.

1: Lightmaps en UE4. LINK: http;//docs.unrealengine.com/latest/INT/Engine/Content/Types/StaticMeshes/LightmapUnwrapping/index.html
2: CryEngine. Dynamic VS Static Lighting. LINK: http://docs.cryengine.com/display/SDKDOC4/Static+vs.+Dynamic+Lighting

El Unity 5 y sus versiones anteriores componen un motor querido por desarrolladores independientes de todo el mundo para el desarrollo de videojuegos, sobre todo en móviles. No obstante a la hora de conseguir imágenes fotorrealistas el motor no se muestra amigable para ello y tras varias semanas de probarlo se habían logrado muchos más avances en UE4. Esto no quiere decir que no sea fotorrealista, simplemente no se le pudo exprimir todo lo rápido que se debería.

Ya se ha visto hasta dónde se puede llegar en términos gráficos con esta tecnología, ahora veamos cómo se llega. Siempre teniendo en cuenta que los profesionales forman parte de empresas titánicas con millones de dólares de inversión y en nuestro caso, estudiantes sin más inversión que nuestra tiempo y dedicación.

Estudiando las herramientas, ¿És posible usar estos programas para la visualización de producto? ¿Se usarán esta clase de herramientas en el futuro?

En Unity Technologies[1] ya han respondido a esta pregunta[2], pero para nosotros la única manera de responder a estas preguntas es adentrarse en el campo de los gráficos 3D y aprender a usar este conjunto de herramientas, porque usar Unreal Engine 4 requiere tener que aprender a usar ciertos programas entre un vasto número de alternativas. Y para conocer que programas usar antes se deben hacer ciertas averiguaciones. ¿Cuál es el proceso a seguir?

El proceso en materia de producción audiovisual es conocido como Flujo de Trabajo y es la manera de organizarse de un individuo o un colectivo para llegar a determinado acabado. Este Flujo de Trabajo variará dependiendo de la técnica y de los resultados deseados por eso antes de seguir adelante debemos hacer un inciso en el Estado del Arte acerca del PBR según Unreal Engine 4.

1: Unity 5. LINK: https://unity3d.com/es/5
2: Offtopic Showreel. LINK: https://youtu.be/CLPBFlA1DAw

ESTADO DEL ARTE. PBR EN UNREAL ENGINE 4. APROXIMACIÓN TEÓRICA.

Basándonos en la conferencia en el Siggraph que dieron los technical artists de Epic Games vamos a explicar en qué consiste su nueva tecnología PBR, intentando simplificar su presentación.

En la conferencia del 2013, Brian Karis de Epic Games explicó cómo funcionaba el modelo matemático usado en el shading[1] de su programa. En el PDF de la presentación se pueden encontrar las ecuaciones y código empleado en la creación de este shader pero eso ya es materia de ingenieros informáticos gráficos. Explicó que se basaba en el ya creado por Disney[2] para películas de animación como Rompe Ralph (Por extraño que parezca una película de animación estilizada introdujo un sistema de materiales realista) pero simplificándolo para cumplir con varias premisas sobre las que se apoyaba el proyecto. Ya que al tratarse de un motor en tiempo real debía ser mucho más óptimo en sus cálculos al tiempo que facilitaba la tarea a los artistas. De esta manera los materiales serian constituidos por pocos parámetros con el objetivo de que fuera más fácil optimizar el G-Buffer, que no es más que el espacio que utilizan las texturas activas en la memoria de la tarjeta gráfica. También constituirían las entrada de información que les permitiera crear un estándar para materiales físicos (Physically Based Shaded Material).

Estos parámetros de entrada son: BaseColor, Metallic, Roughness y Cavity. Los tres primeros son los mismos que en el modelo de Disney y este último en teoría se utilizaba para sombrear las cavidades que no eran geométricamente reales (Presentes en un Normal Map), esto lo explicaré mas adelante, pero a dia de hoy no se incluye entrada para este parámetro en la versión 4.9.1. Quizás se ofrecía en la versión 4.0.0 pero en la actual dicho efecto se logra de manera distinta. Como el Cavity ya no está procedo a definir cada parámetro por separado. Recordamos que una de las bases del trabajo es que se hace partiendo de que se conocen los flujos de trabajo de Vray y Mental Ray con lo que a menudo se tomarán ejemplos usados en esos flujos para comparar métodos.

BaseColor: Se le puede introducir información lineal (De un solo canal) o RGB (Tres canales cada uno escala de un color primario) y se le puede introducir tanto de forma vectorial (Numérica) o través de un Mapa o Textura. Dicho parámetro representa como su propio nombre indica el color base del material. Se diferencia del Diffuse o Difuso en que este lleva el color del brillo en el caso de materiales Conductores (Por lo general metálicos). En el caso de Materiales Dieléctricos (Como el plástico) el brillo especular será el físicamente correcto (Color del ambiente + Color de la fuente de luz).

Metallic: Solamente se le puede introducir información lineal de 0 a 1, es decir, blanco, negro o grises. Los desarrolladores no dicen que deba de ser un parámetro Binario pero en principio es la manera correcta de utilizarlo. Aunque recomiendan si en un objeto con zonas metálicas y no metálicas hay transiciones, en ese caso se podrán usar grises. Se diferencia de Specular (Mental Ray u otros flujos de trabajo PBR) y del Reflect Map (VRay) en que la cantidad de reflejo de un material está sujeta a si es metálico o no y no a si a la cantidad de reflejo que se crea por intuición en los Specular tradicionales induciendo a error. Lo malo se esto es que si se quieren lograr efectos irreales vamos a tenerlo complicado, por suerte los de Epic Games han conservado la entrada de parámetro Specular, por conveniencia propia, para el ya lanzado juego Fortnite[3].

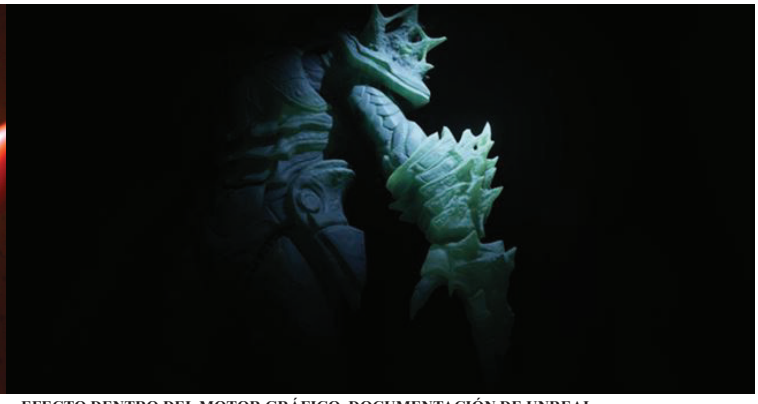
1: PDF 2013, UNREAL PBR Sigggraph conference. LINK:
https://de45xmedrsdbp.cloudfront.net/Resources/files/2013SiggraphPresentationsNotes-26915738.pdf
2: PDF Disney's Theory. LINK: http://blog.selfshadow.com/publications/s2012-shading-course/burley/s2012_pbs_disney_brdf_slides_v2.pdf
3: Fortnite a cartoon looking game inside Unreal. LINK: https://www.fortnite.com/

Roughness: Solamente se le puede introducir información lineal de 0 a 1, es decir, blanco, negro o grises. Su nombre viene a significar algo así como aspereza, así que cuando más áspero sea el material más cerca de 1 y menos reflejará la superficie y más absorberá la luz dispersándola sobre sí. En cambio cuando más cerca del 0 más pulido y más reflejará el entorno que el rodea. Este parámetro parece el inverso del Glossiness conocido por los técnicos de otros motores de render, aunque Roughness probablemente también les suene, ya que es aceptado por Vray. Este parámetro que también es conocido como Microsurface Detail, viene a indicar cuan irregular es la superficie a nivel microscópico lo que determina su capacidad de reflejar o absorber la luz. Al decir que parece el Glossiness Map es que solo se parece ya que el Roughness en realidad posee la capacidad de reflejar como el antiguo Specular Map.

Posteriormente siguiendo el modelo de Disney se incluyeron otros métodos de Shading pero no fueron incluidos como parámetros para el material base y en vez de eso se trataron como casos especiales. En este proyecto no han tenido un peso significativo debido a la ausencia de dichos efectos, al carecer de una necesidad real de esos materiales. Pero no está de más mentarlos. Aunque en la conferencia del Siggraph de 2013 estos casos de Shading eran: Subsurface, Anisotropy, Clear Coat y Sheen. A día de hoy, al parecer según la documentación Online del motor gráfico estos casos se han visto reestructurados como al parecer el Cavity del que hablábamos al principio. De esta manera Subsurface se fracciona en tres modelos de Shading: Subsurface, Preintegrated Skin y Subsurface profile. Clear Coat se mantiene, Anisotropy desaparece, aunque puede ser emulado mediante programación nodal. Y Sheen desaparece y al parecer nunca estuvo ya que el modelo fisico-matemático del shader nunca se definió bien[1]. Estos modelos de Shading responden a los siguientes casos:

-Subsurface: Simula el efecto de Subsurface Scattering, lo que viene a significar Dispersión Bajo la Superficie. O como se dispersa la luz por dentro de un cuerpo cuando este se ve alcanzada por un haz de luz. Ejemplo:

EFECTO EN EL MUNDO REAL. FOTO DE INTERNET[1].

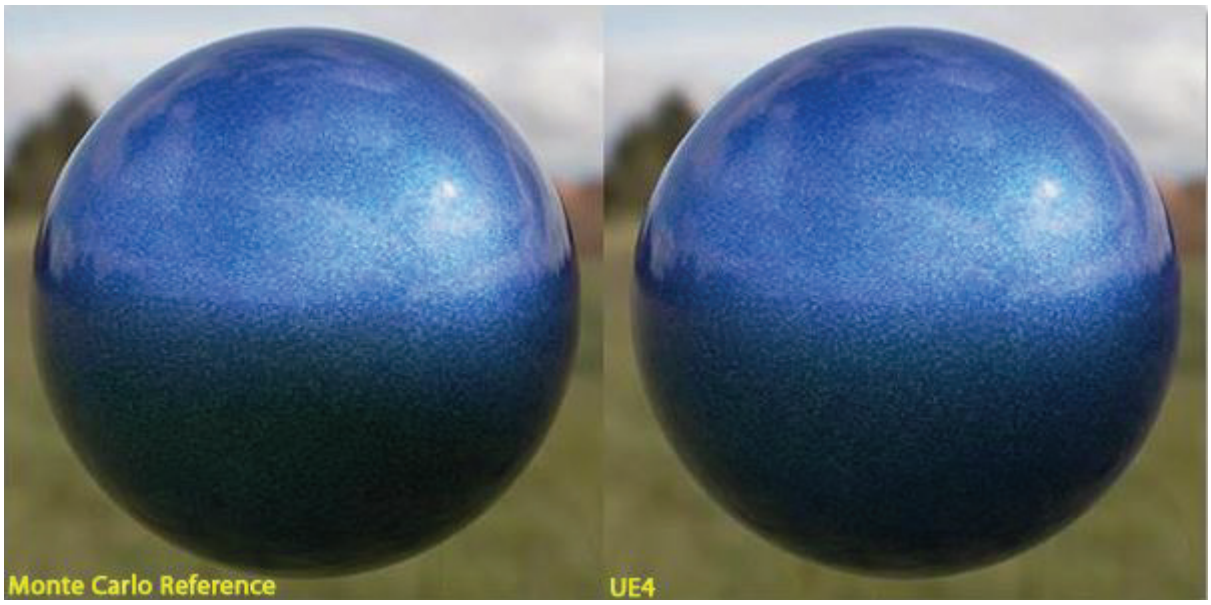


EFECTO DENTRO DEL MOTOR GRÁFICO. DOCUMENTACIÓN DE UNREAL.

-Preintegrated Skin y Subsuface Profile: Vienen a hacer lo mismo que el anterior Shader, solo que el Skin es físicamente inexacto, pero mucho más rápido de calcular, mientras que el Subsurface Profile está orientado a la piel humana, y este obtiene los mejores resultados pero cuesta más de calcular.

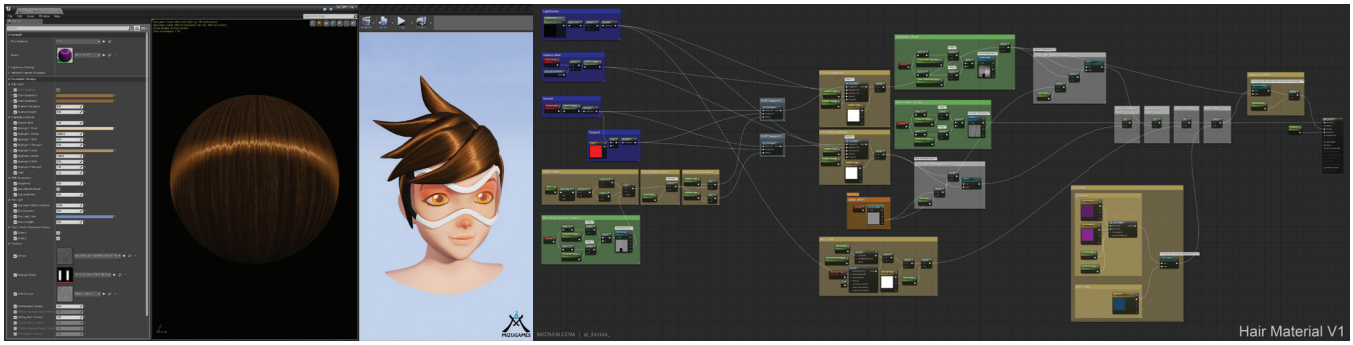
1: Foto de Internet. LINK:
http://www.neilblevins.com/cg_education/translucency/translucency.htm

-Clear Coat: Este Shader trata de simular un efecto de material laminado. Por ejemplo ciertas pinturas de coche que tienen un acabado metálico pero ofrecen cierta textura granulado por debajo. Ejemplo:



EFECTO DENTRO DEL MOTOR GRÁFICO. DOCUMENTACIÓN DE UNREAL.

-Anisotropy: No existe un Shader en Unreal que pueda calcular la anisotropia de un material, lo que viene a significar su distribución irregular a través de una superficie. No obstante se puede emular este efecto mediante programación nodal, de esta manera crear tu propio Shader en una capa de código superior. Un buen ejemplo es el reflejo del pelo, que se distribuye de forma irregular por todo el cabello y como resulta imposible calcularlo para cada pelo por su lado, se realiza un shader con parámetros propios. Algunas personas han logrado emular este efecto con éxito[1]:



IZQUIERDA: EFECTO DENTRO DEL MOTOR GRÁFICO. DERECHA: EL MATERIAL PROGRAMADO MEDIANTE NODOS EN EL EDITOR DE MATERIALES. POR MIDAEM.

El documento del Siggraph continúa explicando las bondades del editor de materiales, destacando su capacidad de adaptación. Esta adaptación se materializa en el Material Layering. Que abre la posibilidad de aplicar materiales a un objeto mediante máscaras y crear combinaciones de forma dinámica. Esto ahorra tiempo, ya que si se tiene un material de acero y un material de óxido, usando máscaras y capas puedes aplicarlo a distintos objetos en las zonas que interesen y luego cambiar alguno de los dos materiales.

1: Original Post:
https://forums.unrealengine.com/showthread.php?60501-WIP-Hair-Material-with-anisotropic-reflections

Aqui podemos observar el efecto del Material Layering. Mismo objeto, distintos materiales. Este efecto se ha utilizado en el caso práctico aunque de forma mucho mas limitada.

POR EPIC GAMES. SIGGRAPH 2013.

A continuación se abandona el tema de los Materiales y los Shaders para centrarse en el modelo de iluminación. Se explica todo lo que se ha mejorado en materia de iluminación desde versiones anteriores del Unreal a esta cuarta versión. Como se aprecia en la miniatura siguiente:

POR EPIC GAMES. SIGGRAPH 2013.

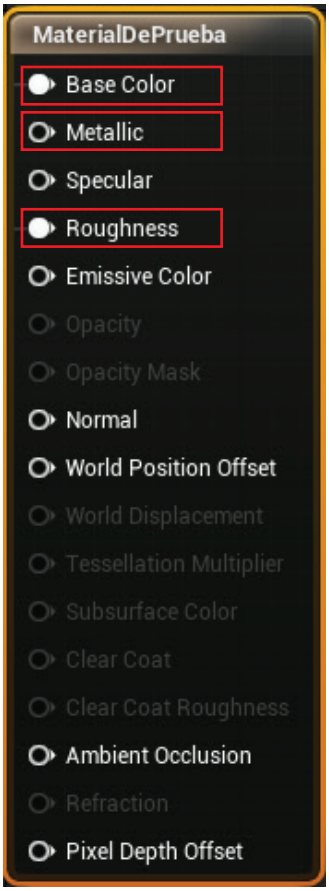
Lamentablemente la materia de iluminación es extraordinariamente compleja y dejamos por el camino el estudio de esta. Que por complejidad y falta de tiempo no podrá ser incluida en esta memoria. Por este mismo motivo el estudio del estado del arte acerca de iluminación en Unreal concluye. Más información en Bibliografía.

ESTADO DEL ARTE. PBR EN UNREAL ENGINE 4. APROXIMACIÓN PRÁCTICA.

Es fácil tras leer el apartado anterior seguir albergando muchas dudas acerca de las mecánicas que rigen el PBR. Para solucionar toda duda se explicara otra vez lo mismo pero desde un enfoque mucho más práctico. Y en esta ocasión nos apoyaremos sobre dos pilares. Por una parte explicaremos de forma más grafica el núcleo del proyecto, los materiales PBR, desde el interior del propio Unreal Engine 4. Y para apoyar esta explicación acudiremos a lo documentación online de Marmoset[1], un desarrollador de software cuyo visor 3D, llamado Marmoset Toolbag 2[2], desempeñará una función fundamental facilitando la comprensión de lo que se está haciendo.

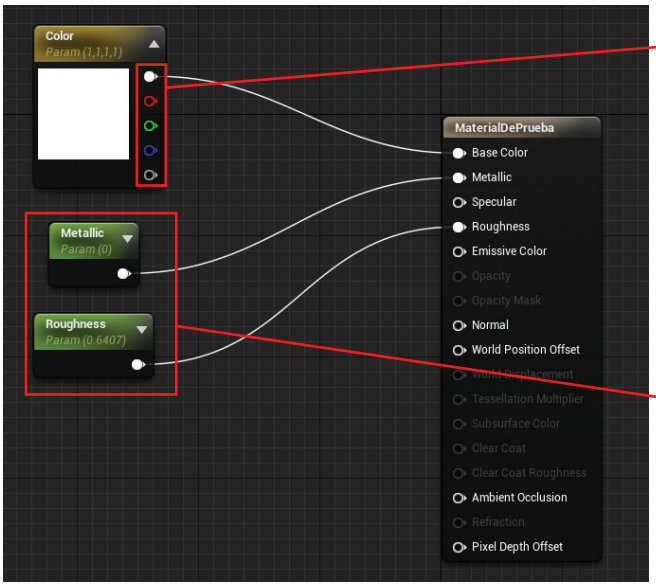
Dicho visor sirve de intermediario entre herramientas de producción y motores gráficos. Su versatilidad al apoyar múltiples métodos de renderizado PBR y no PBR. Metalness-Roughness (Unreal Engine 4) y Specular/Glossiness (CryEngine 3). Con diversos modos de Shading, lo convierten en un ayudante muy potente. Aunque como explicaremos más adelante, esto puede ser un arma de doble filo. Aun así tiene su lugar en el Pipeline (Diagrama de Flujo por el cual pasan los datos para llegar a tener valor y ser llamados Assets) sirviendo para verificar los niveles de los parámetros empleados, es decir, comprobar que todo está en su sitio y se ve como debería antes de seguir adelante.

Repasando la teoría, tenemos tres canales principales en los materiales de Unreal Engine 4. BaseColor, Roughness y Metallic. Ahora pasamos a ilustrar de un modo gráfico su funcionamiento:



CAPTURA EN EL EDITOR DE MATERIALES.

En la miniatura de la izquierda podemos apreciar encuadrados en rojo la entrada de los parámetros principales que definen un material PBR. Aunque existen más parámetros solo se explicaran cuando se usen y lo requieran. A continuación mostraremos un ejemplo de material muy sencillo:



En el parámetro de entrada del Base Color podemos apreciar 5 salidas. Blanco: Representación RGB-A de todos los componentes. Rojo, Verde y Azul seria cada componente del RGB por separado. Y por último en gris, el Alpha, la Máscara de opacidad. Un valor ligado a la transparencia de una imagen.

El parámetro Metallic y Roughness solo trabajan en una componente. Una componente lineal basta para definir su funcionamiento.

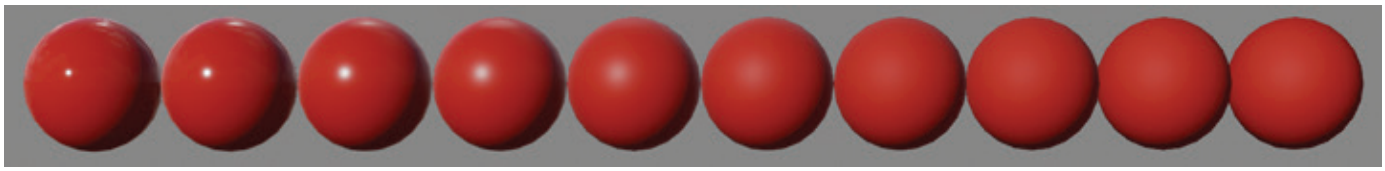
Vemos que el material se construye mediante programación nodal, como el Slate Material Editor de 3DMax. Y los valores de entrada son para los tres parámetros información vectorial: (1, 1 ,1 ,1), (0), (0,6407).

1: Marmoset Site. LINK: http://www.marmoset.co/toolbag/learn/pbr-practice
2: ToolBag2. LINK: http://www.marmoset.co/toolbag

Para diferentes BaseColor:

Para diferentes Roughness, con el Metallic a 0 (BaseColor: Rojo):

0 1

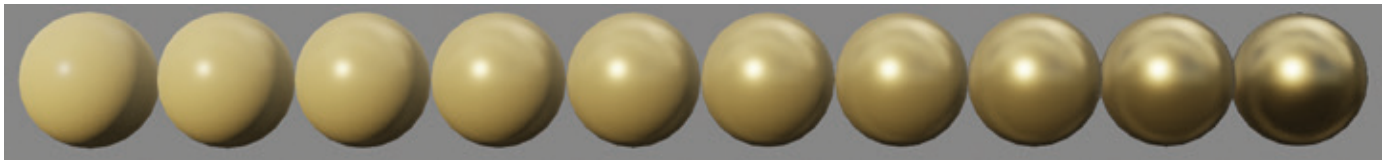


Para diferentes Roughness, con el Metallic a 1 (BaseColor: Gris Claro):

0 1

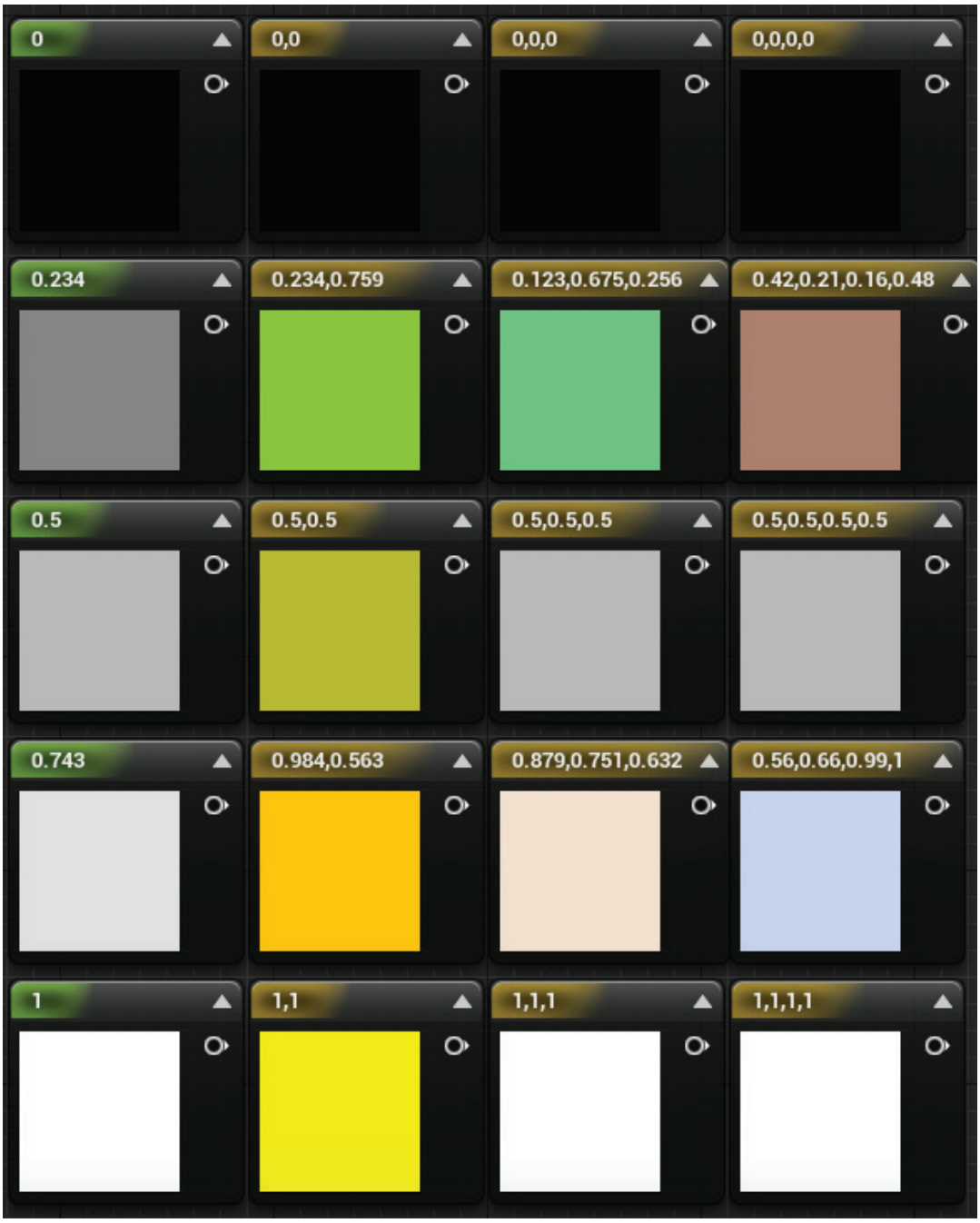
Para diferentes Metallic, (Roughness: 0.5)(BaseColor:Dorado):

0 1



Por lo general, a excepción de algunos casos, el Metallic de los Materiales será 1 o 0, consiguiendo la cantidad de reflejo deseada mediante el combinación “Binaria” de Metallic y “Continua” de Roughness.

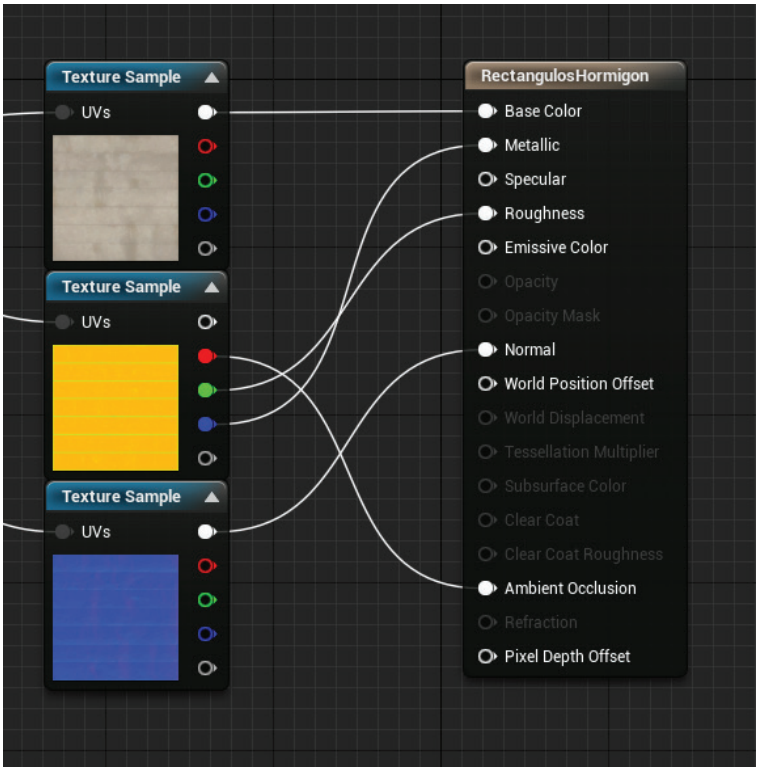
Cabe aclarar que en el Unreal las entradas de estos tres parámetros van a tener su equivalente en mapa de pixeles. Ya sea vectores de uno, dos, tres o hasta cuatro componentes. A continuación mostramos una tabla con los vectores en disposición horizontal de 1 componente a cuatro. De 0 a 1 en disposición vertical, intercalando con valores aleatorios. Y debajo de cada uno su representación L8 (Lineal), RG (2 Canales: RedGreen), RGB (3 Canales RGBlue) y RGB-A (3Canales mas Alpha)



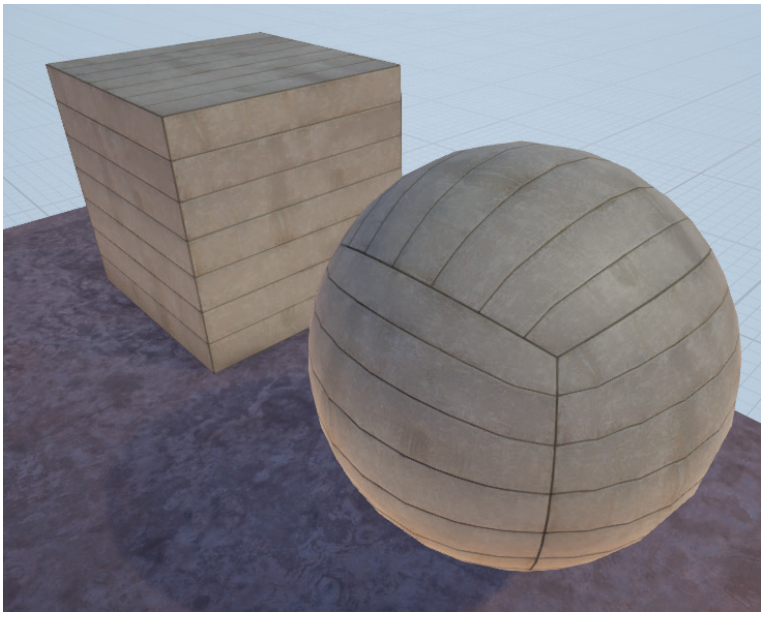
Este equivalente en mapa de pixeles es en realidad un único pixel color plano, pero adelanta la compatibilidad de los parámetros por mapas de pixeles cuadrados (relación de aspecto 1:1) traídos desde cualquier programa de edición de imágenes o otros medios digitales. Estos vectores, también sirven para realizar operaciones matemáticas dentro del editor de materiales, que nos permitirán crear efectos interesantes, como podremos ver más adelante.

Ahora vamos a exponer varios materiales bastante comunes cuyos parámetros de entrada serán mapas de pixeles. En la vida real nada es absolutamente plano, por eso las texturas necesitan llevar ruido, color, valor, contraste y en definitiva una serie de elementos que solo se puede lograr con mapas de pixeles. Los materiales elegidos serán hormigón, madera y por ultimo algo más concreto.

Hormigón:



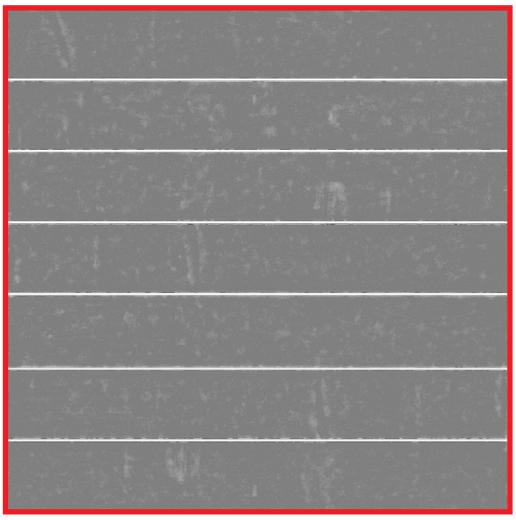
CAPTURA DEL MATERIAL



MATERIAL VISTO EN EL VISOR.

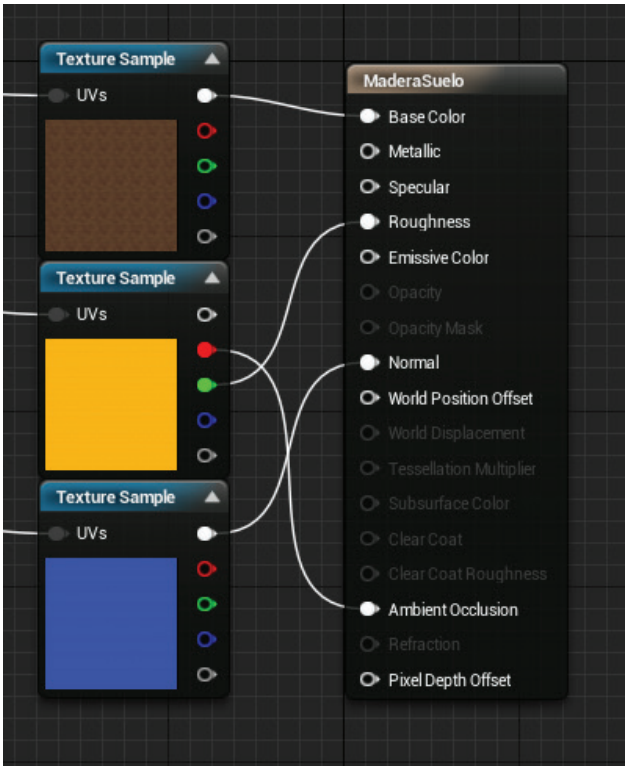
BASECOLOR TEXTURA .TGA, 2048X2048.

METALLIC TEXTURA .TGA, 2048X2048.

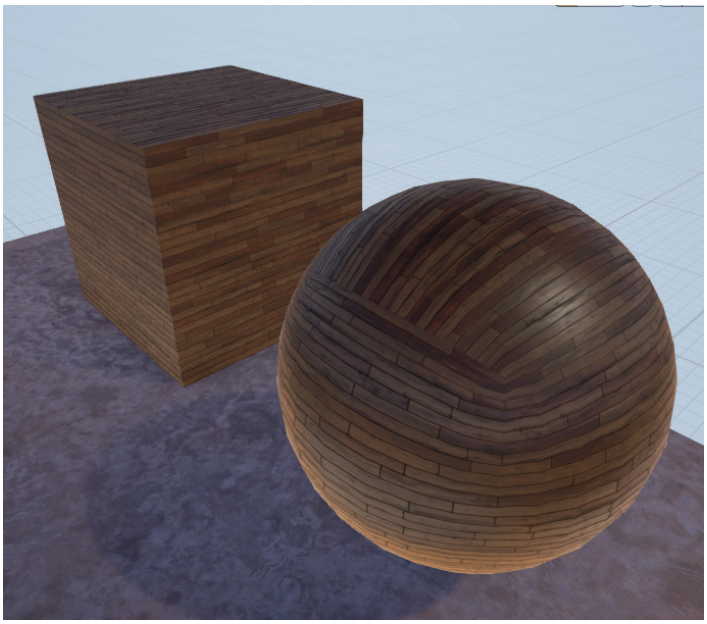


ROUGHNESS TEXTURA .TGA, 2048X2048.

Madera:



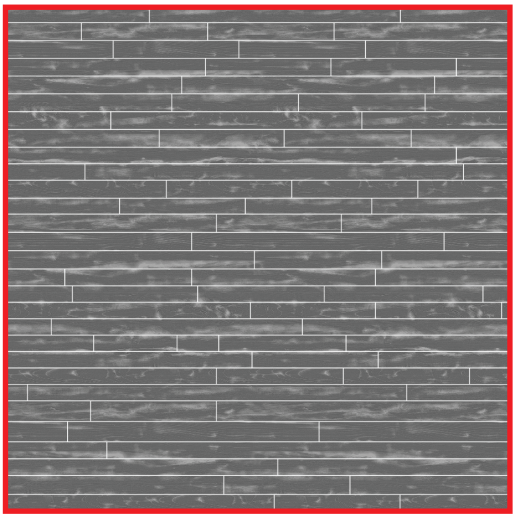
CAPTURA DEL MATERIAL



MATERIAL VISTO EN EL VISOR.

BASECOLOR TEXTURA .TGA, 2048X2048.

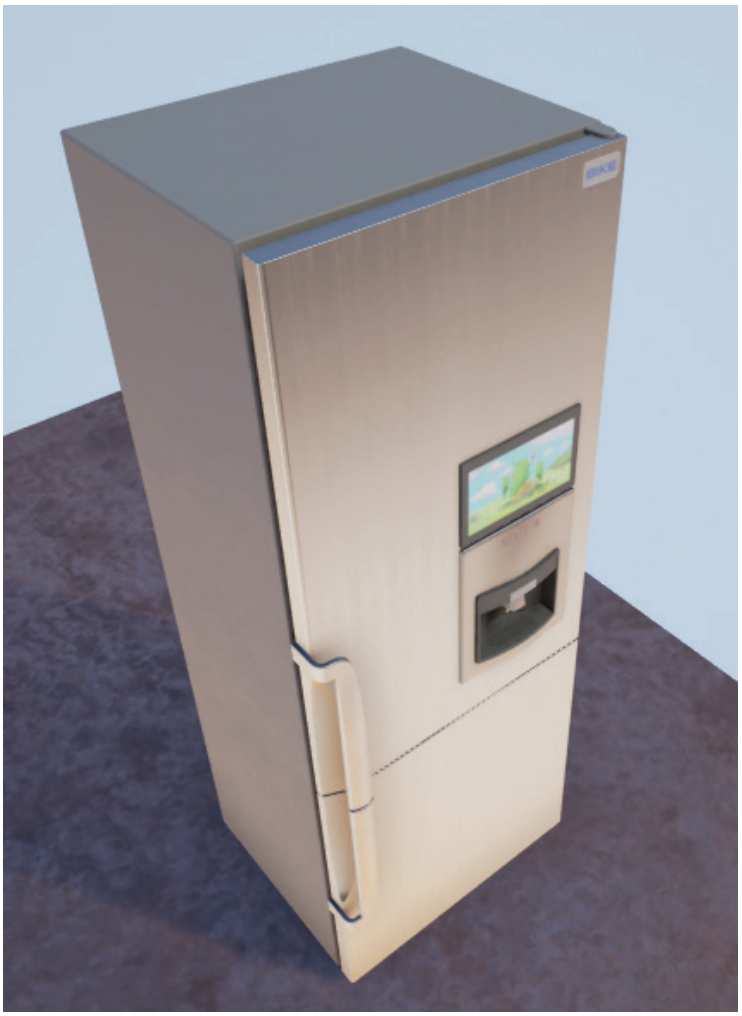
METALLIC TEXTURA .TGA, 2048X2048.



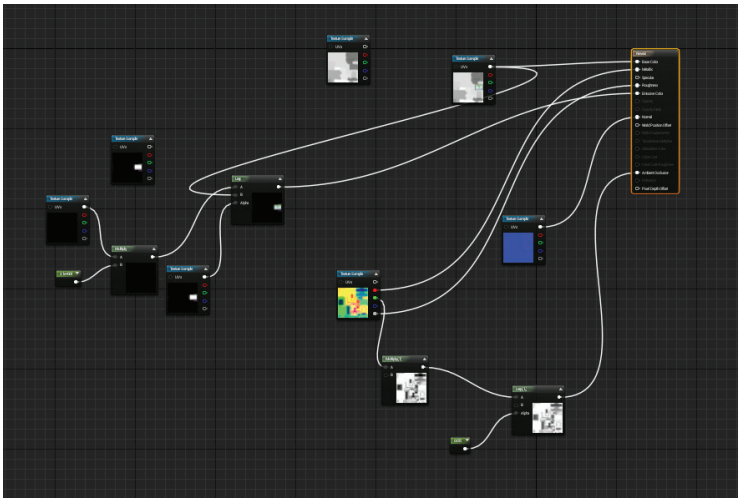
ROUGHNESS TEXTURA .TGA, 2048X2048.

Por ultimo un objeto más complejo que intercala material metálico con no-metálico. Un proceso de texturizado será explicado más adelante.

Diferentes materiales:

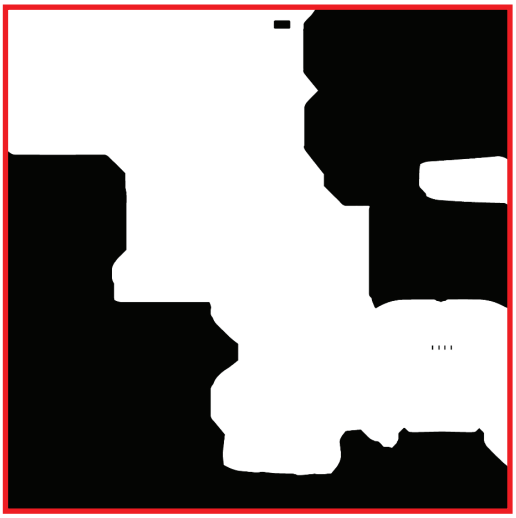


MATERIAL VISTO EN EL VISOR.

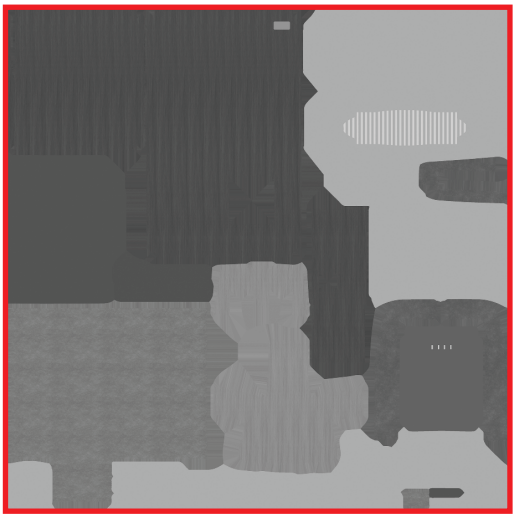


CAPTURA DEL MATERIAL

BASECOLOR TEXTURA .TGA, 2048X2048.



METALLIC TEXTURA .TGA, 2048X2048.



ROUGHNESS TEXTURA .TGA, 2048X2048.

A estas alturas del apartado llega el momento de introducir el Marmoset Toolbag 2, el visor 3D tiempo real más competente del mercado.

Hemos podido apreciar el funcionamiento de los materiales PBR, ahora podemos apreciar su gran capacidad de adaptación a cualquier entorno en el que se pongan. Para representar esto, resulta mucho más cómodo hacerlo usando un visor 3D basado en el mismo principio PBR que no un motor gráfico entero, no porque no se pueda hacer en este, pero sí que es mucho más laborioso.

Para ilustrarnos, se ha modelado una silla de diseño conocido. A la derecha se puede ver con un Shader estándar. La texturizamos, y la importamos a un motor de iluminación PBR.

Como se puede apreciar las texturas de la silla se adaptan a su entorno, reaccionan con la luz, con el color del ambiente, eso es una parte del PBR. Lamentablemente el cómo se vea en Marmoset, no resultará representativo de cómo se verá luego en el motor donde queramos disponer de ello, y es por eso que la segunda parte del estándar PBR entra en juego.

Un motor gráfico PBR o físicamente preciso es cuando por una parte reúne las condiciones en materia de renderizado, iluminación, gestión de los reflejos, etc... Pero por otra parte es necesario ejecutar un trabajo PBR, es decir, acudiendo a una tablas de valores correspondientes al material que se está trabajando para según qué parámetro. Esto es una parte muy importante del flujo de trabajo, y ambas partes son vinculantes y la una cosa sin la otra no serían PBR.

El fallo esta en pensar que por estar en un motor gráfico PBR vas a obtener dicho acabado, cuando en las bases del PBR se deja muy claro que fue creado para dejar menos cosas a la intuición o visión artística y más a la técnica y conocimiento físico de los materiales. Por eso destacamos que si algo funciona y parece realista, pero no cumple con los valores PBR, entonces sencillamente no lo es, y vendría a ser físicamente impreciso y lo más probable es que trabajando así, a la larga tus trabajos adquieran una disonancia técnica. No obstante es perfectamente válido saltarse los valores y dotar de materiales no PBR a un motor PBR, obteniendo un resultado diferenciador. Dicho esto ahora es necesario hacer especial hincapié en los dos estilos de PBR que existen. Por una parte tenemos el BaseColor/Roughness/Metallic o Metalness, que es el estilo que usa Unreal Engine 4. Y por la otra parte tenemos el Albedo/Glossiness/Specular, estilo usado por CryEngine3, como se ha indicado anteriormente.

El primer estilo calcula por ti los valores de reflectividad de los materiales, pero es más exigente a la hora de introducir los valores de BaseColor para Metales y exige más coherencia en el uso del Roughness para según que materiales, con valores habitualmente prefijados, libres, pero recomendados.

El segundo estilo te permite controlar la cantidad de reflectividad en un material, lo que aumenta considerablemente el error en manos inexpertas, pero dota el sistema de mucha mas libertad, no PBR, que permite llegar a resultado muy atractivos deseables para ciertos tipos de juegos No-Realistas.

Hacer un inciso en estos dos estilos nos viene bien para explicar la dualidad del sistema. De haber trabajado en un motor Glossiness/Specular, nos habríamos acogido a la explicación teórica de Marmoset[1], que tiene una gran cantidad de información, y pese al soporte que hace del Roughness/Metallic funciona nativamente con el otro estilo, aprovechando sinergias con Quixel Suite[2] y CryEngine. Que mientras que Epic Games solo nos ofrece valores de BaseColor respecto a Metales, Marmoset nos ofrece una lista más rica. Aunque en el fondo debido a las grandes comunidades de internet hablando sobre el tema, no es difícil hacernos con listas enormes de valores medidos para los parámetros de los materiales.

Material	BaseColor (R, G, B)
Iron	(0.560, 0.570, 0.580)
Silver	(0.972, 0.960, 0.915)
Aluminum	(0.913, 0.921, 0.925)
Gold	(1.000, 0.766, 0.336)
Copper	(0.955, 0.637, 0.538)
Chromium	(0.550, 0.556, 0.554)
Nickel	(0.660, 0.609, 0.526)
Titanium	(0.542, 0.497, 0.449)
Cobalt	(0.662, 0.655, 0.634)
Platinum	(0.672, 0.637, 0.585)

TABLA EPIC GAMES.

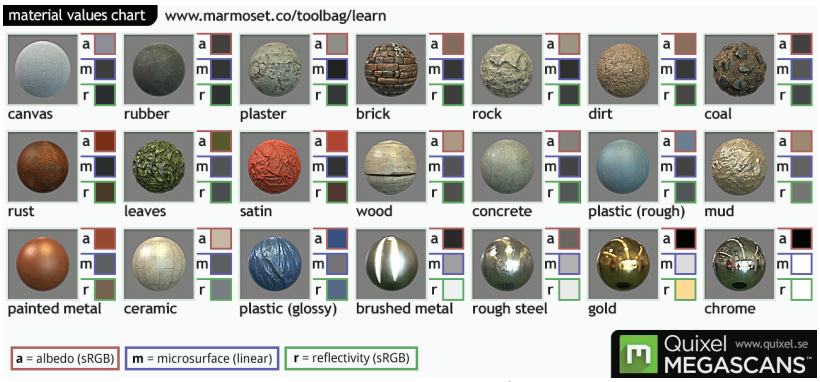


TABLA MARMOSET. DONDE M:GLOSSINESS Y R:SPECULAR SEGÚN LA NOMENCLATURA USADA.

1: Marmoset Site. LINK: http://www.marmoset.co/toolbag/learn/pbr-practice
2: Quixel Suite. LINK: http://quixel.se/

Conclusiones de la aproximacón práctica.

Aquellos que conozcan los materiales de Mental Ray o VRay comprenderán con más facilidad los materiales en Unreal Engine 4 ya que son relativamente parecidos. No obstante los de Epic Games hicieron un gran trabajo con la interfaz y casi todo resulta intuitivo. Sin embargo, si aún queda ápice de duda ahora empezamos con el verdadero núcleo del proyecto, los casos prácticos.

Todo lo que se ha quedado en el tintero específico de los materiales es más propio de los videojuegos que del PBR en sí y por eso no se explicó en la conferencia del Siggraph de 2013. Esto que ha quedado por definir es, en el editor de materiales en los tres materiales ejemplo, se puede apreciar que la hay un mapa de bits azulado conectado en el espacio de Normal, los que vengan de Vray no tienen necesariamente porque conocer este parámetro y por ello será extensamente explicado durante el caso práctico. También se puede apreciar una imagen de colores chillones que conecta con los parámetros de Metallic, Roughness y Ambient Oclusión. Aunque explicaremos esto con más detenimiento más adelante, en resumidas cuentas como eso tres parámetros son lineales, en una imagen RGB caben los tres, logrando optimizar espacio de texturas. En el caso de la nevera, la textura parece más compleja, pero en realidad no lo es tanto, simplemente lleva unas mascaras para el Emisivo, otro parámetro de antes del PBR que tiene la propiedad de emitir luz y que ya explicaremos con más detalle más adelante.

Otro gran pilar del proyecto es la iluminación, pero como comentaba en la explicación teórica, esta no se ha podido explorar con todo lujo de detalles. Pero pese a todo se ha invertido bastante tiempo, pero al ser un tema a caballo entre conocimientos técnicos informáticos y conocimientos técnicos de fotografía, se volvía extremadamente denso y complicado. Pero si no fuera poco para sumar complejidad al asunto, el sistema de Lightmass (Se explica en el caso práctico) de Unreal que es el motor de iluminación que se encarga de calcular la luz y de imprimirla sobre las texturas parece ser prácticamente Unbiased para que nos entendamos: Permite ciertos cambios y ajustes pero es el mismo al final el que se autogestiona. Eso al menos es así en la interfaz. Porque si que se pueden lograr efectos personalizados con acceso al código C++ del engine, pero eso ya es totalmente indigerible para un proyecto de estas características, ya que requerirían conocimientos opuestos a los que se tienen que tener para dominar otras partes del proceso productivo. Con lo que al final se ha optado dejar a la luz que haga lo que considere, y los resultados han sido no-satisfactorios. Tampoco han sido malos, pero el tema de la iluminación puede dar mucho más de sí, pero como va al final del proceso, es siempre a lo que menos horas se le pueden dedicar si se va corto de tiempo, produciendo artefactos; pequeños errores; fallos en el contraste y la gamma; rebotes de luz inexactos; dientes de sierra en haces proyectados, debido a la falta de resolución en el lightmap. Al final para obtener resultados satisfactorios se llegó a la conclusión de que había Tres opciones. La primera tirar a lo bruto y aumentar las calidades en extremo. Inviable para los medios de los que se dispone, pues 32GB de RAM se quedaban cortos. La segunda opción, había que romper el proyecto volver a empezar y hacerlo todo mucho más óptimo, pero obviamente no hay tiempo para eso, así que aceptaremos dichos problemas para no repetirlos en próximos proyectos. La tercera opción era migrar todo el proyecto a CryEngine, pero al tener un flujo de trabajo en las texturas, migrar era una opción muy laboriosa y que requería empezar a aprender otro Motor Gráfico. En definitiva, ninguna solución era viable para el tiempo del que se disponía y se han tenido que aceptar dichos resultados no satisfactorios. Se comenta esto en este apartado porque justifica la ausencia de un buen marco teórico práctico que explique la iluminación en el Unreal Engine 4.

SELECCIÓN DE UN CASO PRÁCTICO.

Después de un poco de teoría y práctica del PBR, para entrar en materia, uno se da cuenta de que no basta con estudiarse la teoría, esta es una materia eminentemente práctica y es necesario envolverse en el proceso productivo para poder hablar con propiedad y conocimiento de causa.

Este proyecto empezó en Febrero de 2015, se ha trabajado en él por casi ocho meses. Se empezó sin apenas conocimiento de 3D, iluminación o render. Y tal como se pudo comprobar tempranamente, para usar Unreal Engine 4 antes hay que saber usar una ingente cantidad de herramientas de entre un elenco aún más variado. Así que se plantearon dos casos prácticos y una adaptación del último.

1. Realizar una cocina en 3DMax y renderizarla con VRay para familiarizarse con modelado e iluminación.

2. Realizar una cocina más grande y a poder ser con espacios adyacentes hasta donde se llegara. Para familiarizarse con el motor, su iluminación, sus materiales, estimar un coste de producción, encontrar dificultades, solucionarlas y en definitiva averiguar si tenía potencial para ser usado en Diseño o Arquitectura como 'visor en Navegador'.

3. Llevar la segunda escena de vuelta al 3D Max, renderizar con VRay y comparar sus acabados.

De los tres casos solo vamos a centrarnos en el segundo. Los otros dos deben ser aclarados, pero en ningún caso nos centraremos en explicarlos en profundidad.

Para el primer caso, se trató de emular una situación real, que sirviera para autoexigirse una buena velocidad de aprendizaje, cuando oportunamente al tiempo de haber empezado, un amigo nos delegó un trabajo que le había sido encargado. El trabajo consistía en hacer varios Renders de una cocina y aunque al principio solo se trataba de aprender 3DMax y Vray este encargo dotó de un sentido profesional al caso práctico, y de un briefing que no vino nada mal. Tras un fin de semana de trabajo, con documentación online y las clases de 3DMax recibidas en la universidad para cuatro cosas de geometría, y unos cuantos modelos gratuitos de los que circulan por internet, pude acabar el trabajo satisfactoriamente.

El segundo caso fue un jarro de agua fría. Para producir para Unreal Engine hay que ser muchísimo más escrupuloso que produciendo para VRay para conseguir un efecto de igual calidad. Los modelos deben seguir ciertas pautas y las texturas son diferentes. Para este caso me propuse producirlo absolutamente todo, pero más que una proposición acabo siendo algo forzoso. Los modelos gratuitos que hay para VRay por ahí circulando están muy descuidadas a nivel técnico, cumplen para renderizarlos pero da más trabajo aprovecharlos para Unreal que hacerlos de cero. De tener todos los modelos listos para renderizar en el primer caso práctico a tener que crearlos todos para el segundo. Dio muchísimo más trabajo del estimado. Ahí fue cuando se tuvo que aprender todo lo necesario y a posteriori creo que no fue fácil dada la experiencia.

El tercer caso práctico ha sido imposible de realizar en el tiempo disponible. Aunque se llegó a disponer de la geometría de la escena, el hecho es que había que rehacer los materiales para que funcionaran en VRay, un trabajo muy repetitivo, lento y pesado. Que al final, gracias a la falta de tiempo, se pudo ahorrar.

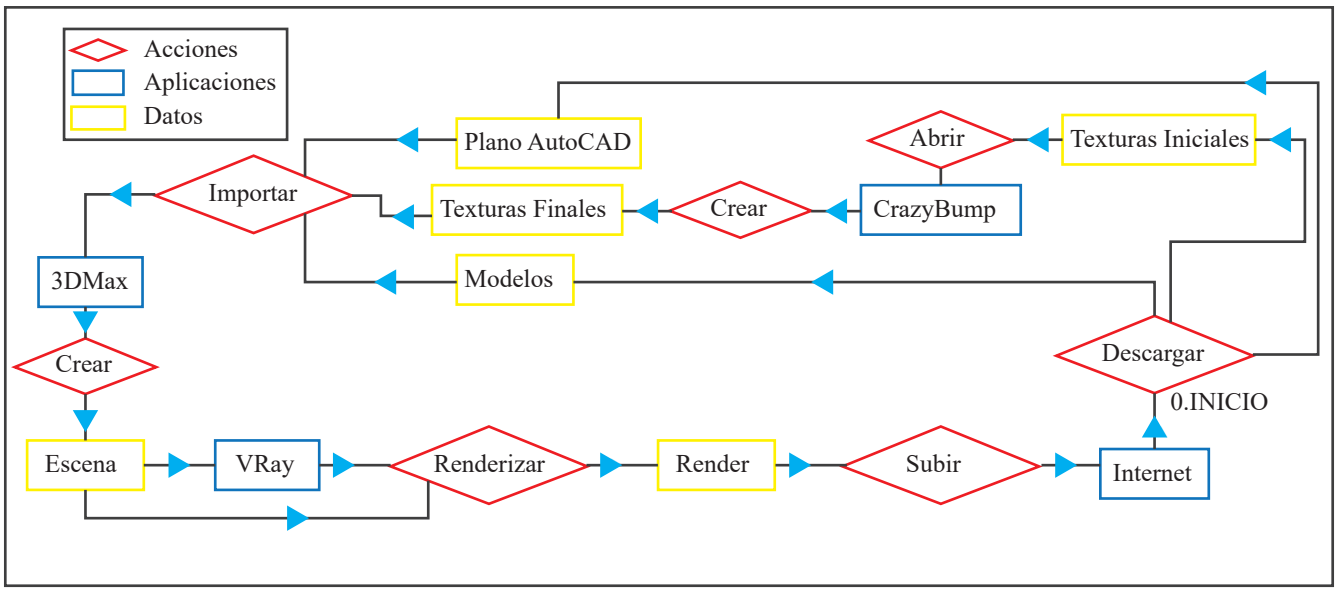
Ocho meses después de determinar los casos se finalizaron. Seguidamente la memoria de cada caso.

MEMORIA: CASOS I Y III.

Uno de los objetivos a la hora de realizar este trabajo fue que la memoria fuera corta y fácil de entender. Y se pretendía esto porque la otra opción sería escribir un documento súper extenso, el cual hubiese sido complicado de organizar y pesado de leer para los posibles interesados. Por este motivo en esta memoria no se explicará paso a paso el proceso realizado para producir el caso 1. Así que se opta por explicar que resultado se alcanzó o a que conclusión se llegó.

Caso 1.

Para este caso o trabajo fue dado un briefing, con un plano en autocad con la planta y el alzado de la cocina a modelar y las especificaciones visuales que debía cumplir. Para llegar a lo que se pedía solamente se necesitó del uso limitado de tres programas: 3D Max 2014, Crazybumb y VRay 3. El flujo de trabajo empleado fue el siguiente:



De las acciones necesarias para realizar el proceso, se tuvo que aprender el uso del CrazyBump y el uso del Vray. Que eran los programas con los cuales no se tenía experiencia previa. El 3DMax bastaron primitivas y Mappeado Box para realizar el trabajo. Las texturas fueron obtenidas del sitio web cgtextures.com y los modelos 3D de tf3dm.com y archive3d.net. Este fue el resultado:

Mas adelante se compararán con los Renders obtenidos con el Unreal Engine 4 a mayor tamaño.

Caso 3.

Como se ha explicado con anterioridad el caso 3 fue descartado por falta de tiempo. A continuación explicaremos lo que se quería hacer, que es: como llevar una escena de Unreal Engine 4 a 3DMax y renderizar con VRay. El proceso es sencillo pero pesado, aun así se incluye aquí por si quien este consultando este documento, se ve en la necesidad de hacerlo.

Partimos de una escena en Unreal Engine 4 ya terminada. El primer paso es montar la misma escena en 3DMax, disponiendo la geometría de la misma manera.

El segundo paso es el verdaderamente problemático. Para que VRay o Mental Ray puedan leer los materiales hay que convertirlos de BaseColor/Roughness/Metallic a Albedo/Glossiness/Specular. Los Normal Map y Ambient Oclusion no requieren conversión. Para hacer esto procedemos según la documentación de Marmoset[1]. No nos pararemos a representar gráficamente el proceso ya que es muy sencillo. Conversión:

-1-: Abrimos el BaseColor en Photoshop.
-2-: Creamos una capa nueva por encima de color negro HEX #000000.
-3-: Aplicamos el Metallic como mascara a la nueva capa anteriormente creada.
-4-: Guarda la imagen resultante como albedo.
-5-: Abrimos otra vez el BaseColor original.
-6-: Creamos una capa nueva por encima de color gris HEX #383838.
-7-: Aplicamos el Metallic como mascara a la nueva capa anteriormente creada.
-8-: Invertimos el valor de la máscara.
-9-: Guarda la imagen resultante como Specular.
-10-: Abrimos el Roughness ahora.
-11-: Invertimos los valores.
-12-: Guarda la imagen resultante como Glossiness.

No parece muy complicado pero si se tiene que hacer para quizás cientos de materiales se pueden tardar muchos días, en un proceso altamente repetitivo.

Tercer paso, renombramos texturas. Para evitar confusiones renombra el Albedo a Diffuse y el Specular si se va a usar VRay a Reflect. Lo demás se deja igual y con los nombres ya no hay posibilidad de confundirse en la entrada de parámetros.

El cuarto paso es respecto al Normal Map, un apunte para los de VRay y su asiduidad al Bump Map. Para que el motor de render lea el mapa es necesario incluir un nodo previamente al Bump Map llamado Normal Map de VRay, recordar subir la intensidad a 100% y la opción de “Flip green channel” activa, ya que Unreal usa el canal G inverso y si lo tenéis bien en Unreal lo tendréis mal en VRay o Mental.

Los últimos pasos ya son lo habitual, crear todos los materiales, asignarlos, poner un Dome con un buen HDR de un cielo que le de riqueza a la iluminación. y poner planos de luz para falsearla si es necesario y así zonear la intensidad de la iluminación global a interés. Render con la configuración óptima y paciencia. Todo está información está muy bien explicada aqui2, un canal que recomiendo.

1: Marmoset PBR Conversion Guideline. LINK: http://www.marmoset.co/toolbag/learn/pbr-conversion
2: Adán Martín, Youtuber Profesor. LINK: https://www.youtube.com/user/adanmq/videos

MEMORIA: CASO II.

En la selección de los casos comentábamos que este caso había sido particularmente complicado de llevar a cabo. Se podría realizar un diagrama de flujo del proceso de trabajo pero uno se perdería entre tanto nodo sin llegar a sacar nada en claro. Para tener una idea, el flujo de trabajo se ha adaptado hasta llegar a incluir el siguiente software:

Photoshop CS6[1], Illustrator CS6[2], Marvelous Designer[3], CrazyBump, Bitmap2Material[4], Substance Painter[5], Substance Designer[6], Quixel SUITE, Xnormal[7], 3Ds Max 2014, Marmoset Toolbag 2, Zbrush 4R7[8], 3D Coat[9] y Unreal Engine 4.

El usar tantos programas no complica necesariamente el trabajo, más bien lo simplifica, porque todo se podría hacer en menos programas, pero la manera de trabajar a la que uno se acostumbra le decanta más por unas opciones sobre otras, no porque sea mejor, es más bien cuestión de comodidad. Y muchos profesionales del sector verán esta forma de trabajar y probablemente suelten alguna carcajada, la experiencia hace al maestro.

Aclarado que el proceso fue un tanto complejo procederemos con el siguiente itinerario, con el fin de no explicar paso a paso lo que se hizo repitiéndose hasta el absurdo, porque es verdad que cada modelo tiene su propia identidad, pero comparten una gran parte del proceso y aunque algunos cuesten más que otros al final son casi siempre los mismos pasos. Itinerario:

1- Se explicará a grandes rasgos como fue la etapa de preproducción, pero no se profundizará en ella porque el proyecto no va sobre esto, y daría demasiada información.

2- Se tomarán los tipos de assets (datos de valor, como materiales o modelos 3D) más representativos del proceso de producción y se explicará paso a paso como se hicieron.

3- Se explicará cómo poner en marcha una escena vacía en Unreal Engine, como se distribuyen los assets por el escenario y que elementos se usaron para mejorar nuestra escena.

4- Se explicará cómo se preparó la escena para su iluminación y se mostrará el resultado final.

5- Valoración personal y conclusiones del caso práctico 2.

Los archivos del proyecto incluidos en el DVD expanden los datos del punto 2 al poderse ver todo el entramado interno del proyecto. Este mismo se optimizó en una ocasión para que tuviera una organización receptiva, invitando a explorar cada recoveco de la producción. Sin embargo la producción de cada asset individualmente se ha descartado incluirla en el DVD, más adelante se explicará el motivo.

1: Photshop CS6. LINK: http://www.adobe.com/es/products/photoshop.html
2: Illustrator CS6. LINK: http://www.adobe.com/es/products/illustrator.html
3: Marvelous Designer LINK: http://www.marvelousdesigner.com/
4: BitMap2Material LINK: https://www.allegorithmic.com/products/bitmap2material
5: Substance Painter LINK: https://www.allegorithmic.com/products/substance-painter
6: Substance Designer. LINK: https://www.allegorithmic.com/products/substance-designer
7: xNormal LINK: http://www.xnormal.net/
8: Zbrush4R7 LINK: http://pixologic.com/zbrush/features/overview/
9: 3DCoat LINK: http://3d-coat.com/

1- Preproducción.

En los proyectos de estas características es de vital importancia desarrollar la idea antes de ejecutarla, y dejarla totalmente definida antes de abordarla. Como este es un factor clave del diseño, no es materia de estudio en esta ocasión y bastará con repasarla por encima.

La preproducción del caso 2 consistió en la elaboración de una serie de documentos que intentaran recoger y acotar todas las dudas acerca del proyecto. Se empezó declarando que el punto de partida debía ser similar al del caso 1, pero expandido.

Se elaboró un briefing que complicó la idea de una cocina, inicialmente, al añadirle estancias adyacentes, la escena también debía dar la impresión que formaba parte de un todo más grande, porque en Unreal Engine 4 no hay fase de composición, y esto es prácticamente necesario.

Luego se procedió a la elaboración de una serie de bocetos que ayudaran a acotar un poco más la idea del proyecto. Posteriormente se realizaron tres tablas de moodboard que reflejaran el espíritu de la escena que se quería plasmar.

Seguidamente se procedió a una segunda tanda de bocetos y después se construyó un espacio 3D con casas intenten ando proporcionar los elementos que iban a aparecer en escena.

Con la escena básica de cajas en mente, se buscaron medidas ergonómicas y normativas en las medidas de puertas, ventanas, techos, distancias entre pasillos, encimeras, mesas, sillas, etc... Y se modeló el espacio que contendría la versión final de la cocina.

Por último se realizaron pruebas técnicas que alargaron mucho la etapa de preproducción. Dichas pruebas técnicas se concibieron para determinar cuál sería el mejor camino a seguir. Durante todo el tiempo lo único que hacía era buscar fallos y listarlos, corregirlos y listar más fallos. Hasta que la lista de fallos se redujo a unos pocos los cuales no se les encontró solución pero que no fueron lo suficientemente problemáticos como para detener el proyecto. En dichas pruebas técnicas de lo que se trataba era de reproducir los pasos de la escena final, pero a pequeña escala, seguir adelante y así destapar fallos que de esta manera no ocurrirían en etapas más avanzadas donde cambiar pequeñas cosas puede suponer prácticamente volver a empezar. Esta etapa fue la más larga, pero al terminarla se podía empezar finalmente con la producción.

2- Proceso de producción de los assets.

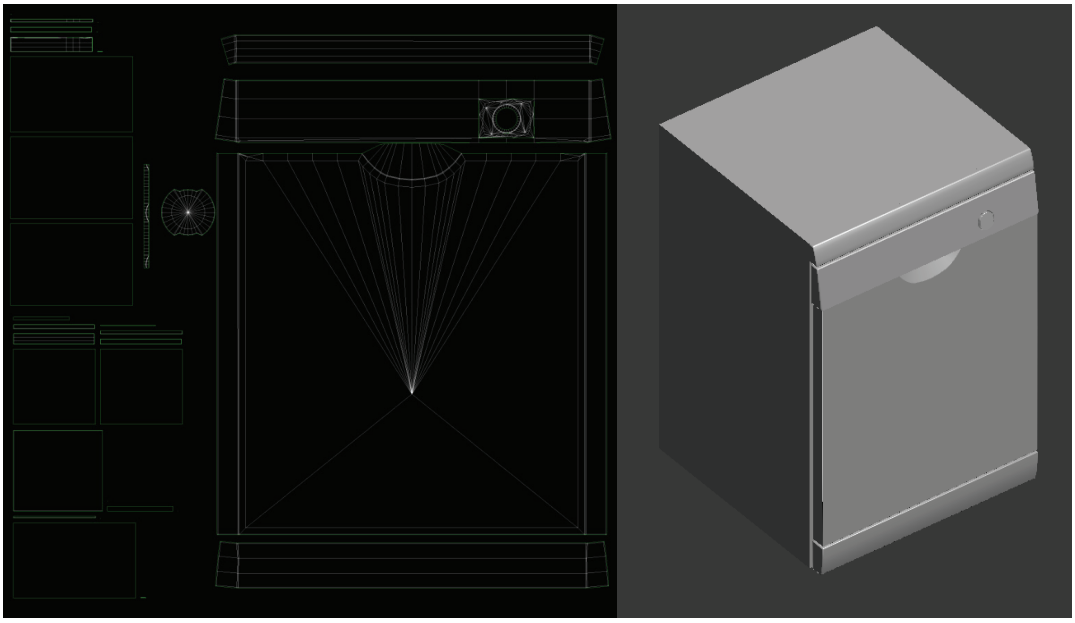
Los assets se pueden clasificar según diversos criterios, en este caso el criterio que se debe usar es “según flujo de trabajo”, es decir, según la manera en la se han hecho. Como muchos la comparten, una buena opción es seleccionar assets cuya elaboración sea lo más diferente posible. Los más completos en este aspecto, no por dificultad sino por cubrir más aspectos de la producción, son el Lavavajillas. Para explicar el proceso lo dividiremos en dos parte: Parte A y Parte B. En la parte A se explicarán por encima los procesos que ya se deberían conocer por el propio grado de diseño o por la experiencia con métodos regulares como el Render Biased (Vray). En la Parte B asumimos que forma parte de la materia que se está explicando en este estudio y será explicado con mayor detalle.

2.1- Asset: Lavavajillas.

Parte A:

Para diseñar el lavavajillas se recurren a referencias obtenidas en Google imágenes. Y cuando se tienen las suficientes para empezar a trabajar se montan en un moodboard y se dejan abiertas en un monitor secundario para ser consultadas en todo momento.

En 3D Max 2014 se modela la caja que ocupara el espacio del lavavajillas. Con el modificador Edit Poly vamos dando forma a la geometría y cuando la tenemos lista redondeamos las aristas con el modificador Quad Chamfer, el redondeo de aristas es algo fundamental, ya que en el mundo real rara vez se presentan aristas de 90 grados, este efecto se puede lograr de muchas maneras, Quad Chamfer es el elegido para nuestro proyecto. Recordemos que si el lavavajillas va ir empotrado, será innecesario modelar las partes ocultas.



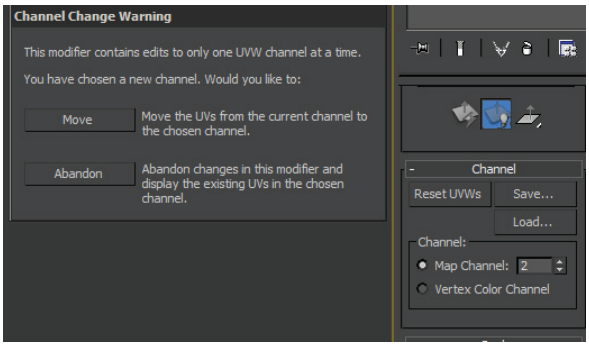
UV DEL OBJETO OBJETO MODELADO

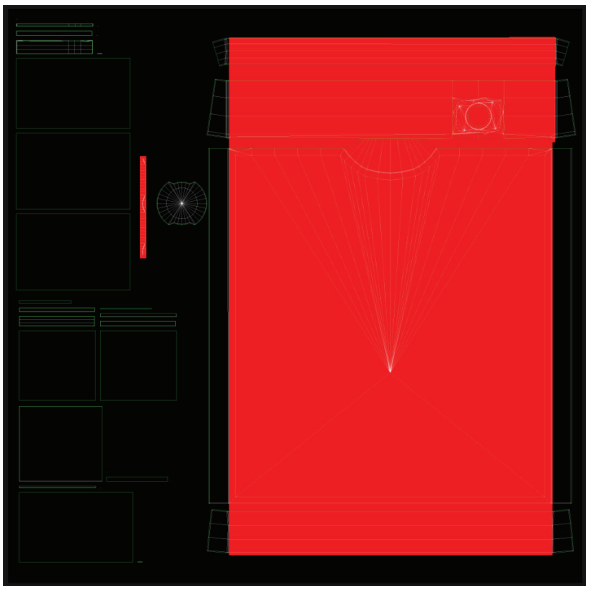
Después con el modificador "UVW Unwrap" sacamos las UVs principales del objeto donde ira la textura. Renderizamos un UVW Template a 2K para futuros procesos. Aplicamos un reset XForm para evitar problemas más adelante y convertimos a Editable Mesh y la devolvemos a Editable Poly para, también evitar algunos otros problemas más adelante.

Llegados a este punto la malla esta lista para empezar a texturizar. A excepción de que se necesitan dos canales de UVs en el Unreal Engine 4, para el lightmap.

Parte B:

Para el Unreal Engine debemos sacar un segundo canal de UVs para los lightmap, esto se hace de la siguiente forma.
Con el objeto limpio de modificadores se le añade el UVW Unwrap otra vez, esta vez se le cambia de canal:

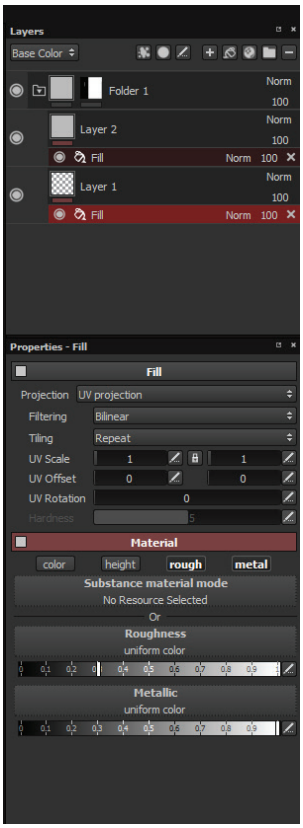




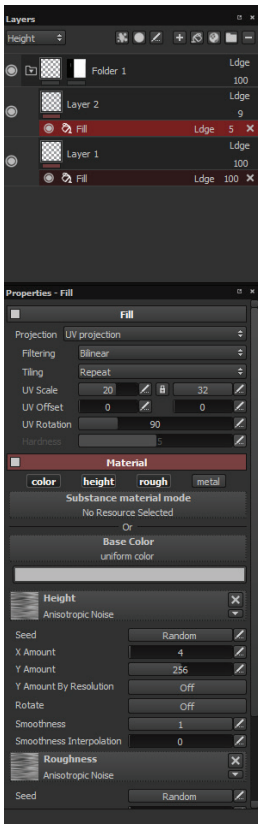
Click en abandonar y seleccionando todos los polígonos hacemos un Flatten Mapping. Botón situado en las opciones de mapeado del Visor de UVs. Con las opciones por defecto nos valdrá en esta ocasión. Colapsamos el modificador y exportamos el objeto como .FBX centrando su origen en 0.0.0. Esta Mesh nos valdrá para texturizar.
Nos llevamos el UVW Template a Photoshop y una vez allí enmascaramos las zonas que vayan a ser de distintos materiales, como se puede apreciar en la imagen de la izquierda las zonas en rojo representan las zonas metálicas.
Sacamos un Alpha de la zona roja y acto seguido nos vamos a Substance Painter.

Recomendamos que llegado a este punto se miren la documentación de Substance ya que esto no es un tutorial y si se desea saber lo que se está haciendo, la documentación en paralelo es la única manera de entenderlo.

Una vez en Substance con la Mesh y Alpha importadas creamos una carpeta y se la aplicado como Bitmap Mask y tal como se ve en SP#1 definimos un primer Fill con los valores base del Roughness y el Metallic. Y tal como se ve en SP#2 añadimos un segundo Fill para añadir un efecto isotrópico del material propio del metal en el Height y en el Roughness. Posteriormente exportamos BaseColor, Roughness, Metallic y Normal Map, generado a partir del Height. En TARGA a 2K. Estas texturas serán las que usaremos entonces como base para su posterior edición en Photoshop, y por eso las añadimos al documento de las máscaras donde tenemos el UVW Template que nos servirá de referencia para saber sobre qué zona estamos trabajando. Opcionalmente podríamos habernos generado un Ambient Oclusion, pero al ser un objeto metálico con el que nos genere el engine bastará.



SP #1



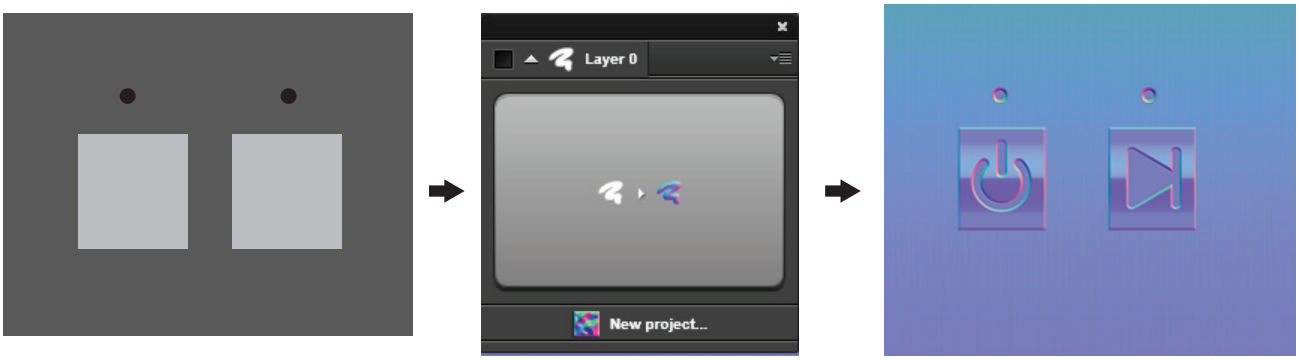
SP #2

A continuación usando el Quixel Suite creamos todos los elementos del panel de el lavavajillas que no hemos hecho junto con la geometría, que son las ranuras de los LEDs y los botones de Encendido y Pausa.

El Quixel suite es un plugin de Photoshop que nos permite obtener Normal Maps a partir de Height Maps.

Los Normal Maps son unas texturas similares al Bump que alteran las dirección normal de los polígonos simulando geometría que no existe. Gracias a su buen rendimiento son vitales en los videojuegos. El Height Map por su parte es una textura en escala de grises donde el blanco significa altura relativa 1 y el negro altura relativa 0, también se puede usar para la entrada de parámetro Displacement Map de VRay que no simula, sino que crea geometría real.

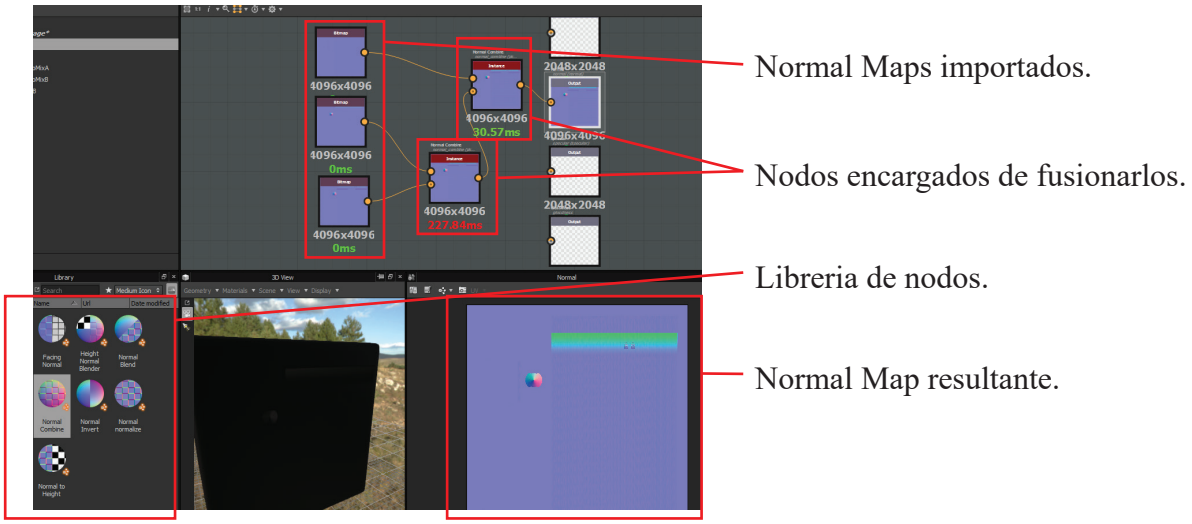
Concretamente del Quixel Suite vamos a utilizar el NDO, que nos permite obtener Normal Maps y editar sus características.



ORIGEN. HEIGHT MAP NDO. QUIXEL SUITE. NORMAL MAP GENERADO. NDO.

Para obtener los símbolos se ha utilizado el Illustrator. Para los textos, el símbolo de apagado y Play/Pause. Distribuyéndolo por los mapas para simular ser otro material.

Y finalmente los diferentes Normal Map obtenidos se han fusionado usando Substance Designer. Concretamente el nodo de Normal Combine en el apartado de Normals que funciona perfectamente y ahorra bastante tiempo en comparación con el Photoshop.

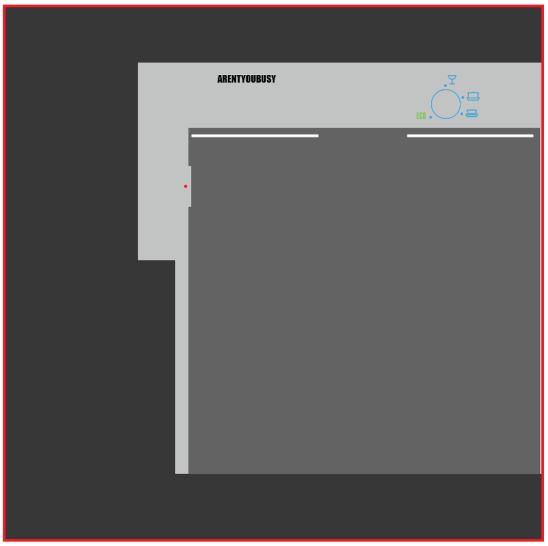


SUBSTANCE DESIGNER

Imagenes del lavavajillas en el Motor Gráfico. Iluminación realizada.

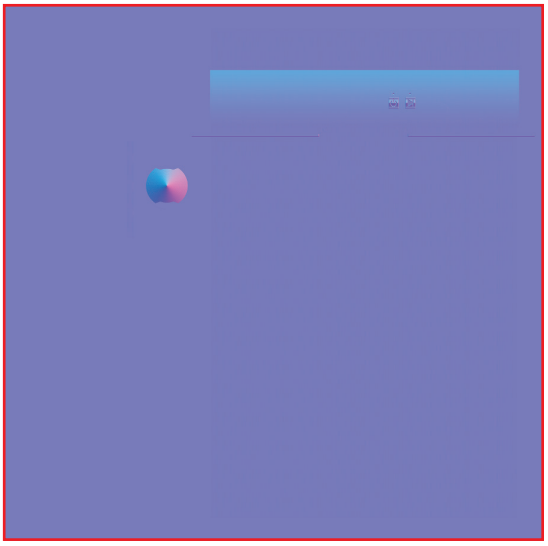
Se puede apreciar un punto brillante, para lograr este efecto basta con conectar un imagen en negro 100% con un punto mapeado brillante rojo encima del LED al parámetro Emisive.

Texturas finales:



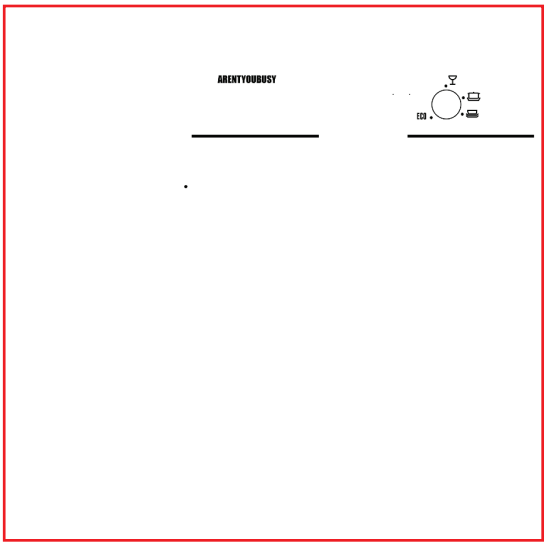
BASECOLOR 2K.

ROUGHNESS 2K.



NORMAL 2K.

EMISSIVE 2K.



METALLIC 2K.

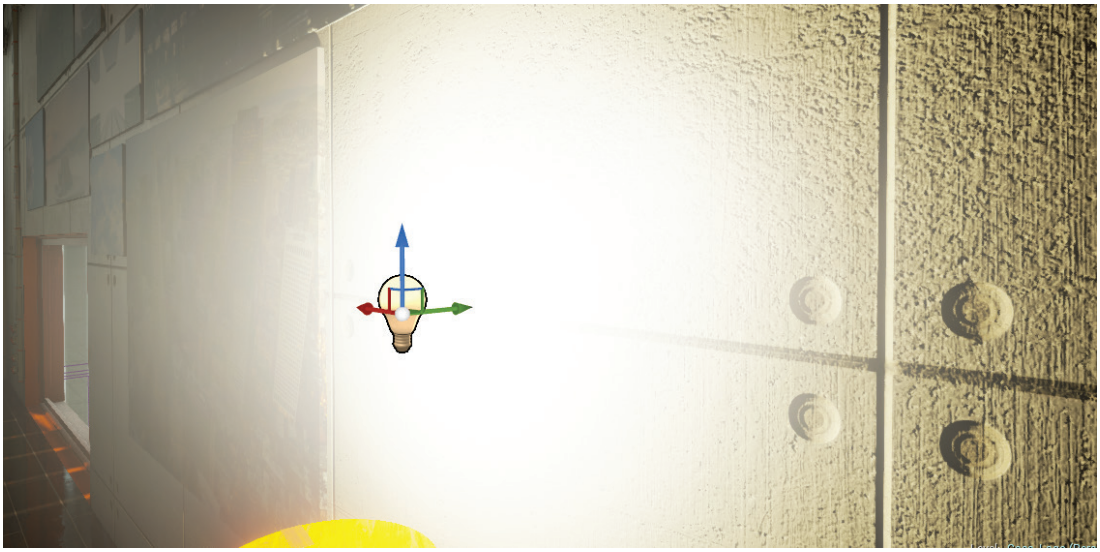
Queda un pequeño detalle por aclarar, el Normal Map es bastante sesudo de explicar, pero a groso modo podemos decir que cada canal representa el objeto recibiendo un haz de luz desde cada uno de los tres ejes de coordenadas.

En las imágenes de la derecha se ve que el selector del programa tiene un extraño efecto metálico parecido geométricamente a una lente Fresnel. Bien. Dicho efecto se consigue proyectando un cono dentado sobre una superficie plana. Lamentablemente los archivos del Baking se perdieron con lo que no se puede explicar con soporte visual.

2.2- Asset: Textura pared.
Aprovecharemos para explicar una cosa que se dejó en el camino. El Ambient Oclusion y el cavity.

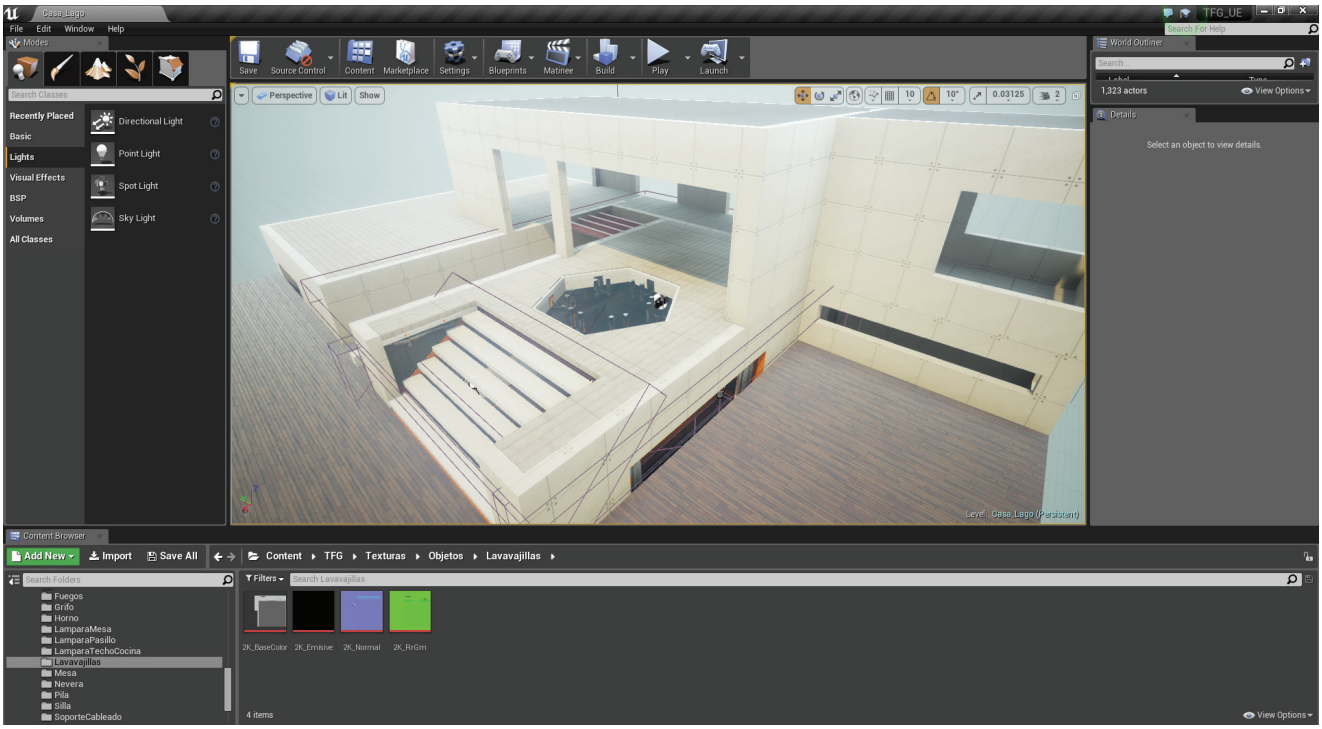
Se explicó en el Siggraph de 2013 que había un parámetro llamado cavity usado para sombrear las hendiduras que existen en los Normal Maps. Porque el motor al no ser geometría real no puede calcular la proyección de las sombras. Por este motivo añadieron cavity. En el flujo de trabajo actual el cavity se obtiene durante el Baking de un modelo en alta a otro en baja en XNormal o bien en Quixel Suite usando de fuente el propio Normal Map. Este cavity se multiplica en el Ambient Oclusion, y de esta manera tendremos sombras en las hendiduras no geométricas, como es el caso de la pared de Hormigo en el proyecto.

Entre placa y placa y en las hendiduras de puede observar las sombras producto del Sonido Ambiente, Oclusion Ambiental o Ambient Oclusion.



Al acercar una luz, estas sombras desaparecen. Para lograr esto se introdujo el Cavity.

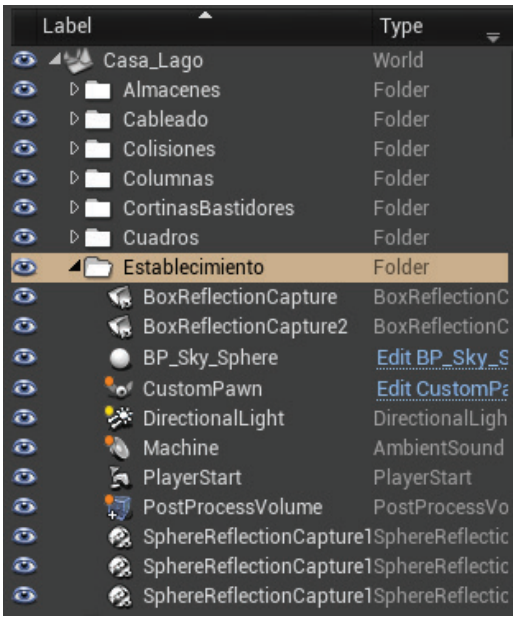
3- Preparando la escena en Unreal Engine 4.



INTERFAZ DE UNREAL

La interfaz es tan simple que no requiere demasiada explicación. A la izquierda arriba los actores predeterminados del motor. Abajo los archivos del proyecto en marcha. A la derecha arriba los actores presentes en la escena y abajo las opciones de la selección.

Para empezar nuestra escena se han arrastrado los siguientes actores predeterminados a un escenario en blanco.



Por una parte. Los capturadores del reflejo, puestos en su sitio indicado. Un SkyBox predeterminado, con el cielo, las nubes y el sol. DirectionalLight para controlar el sol, posición e intensidad. Un Volumen de PostProceso para retocar cosas del render final y un Custom Pawn que actúa como cámara. Para poder editar su velocidad.

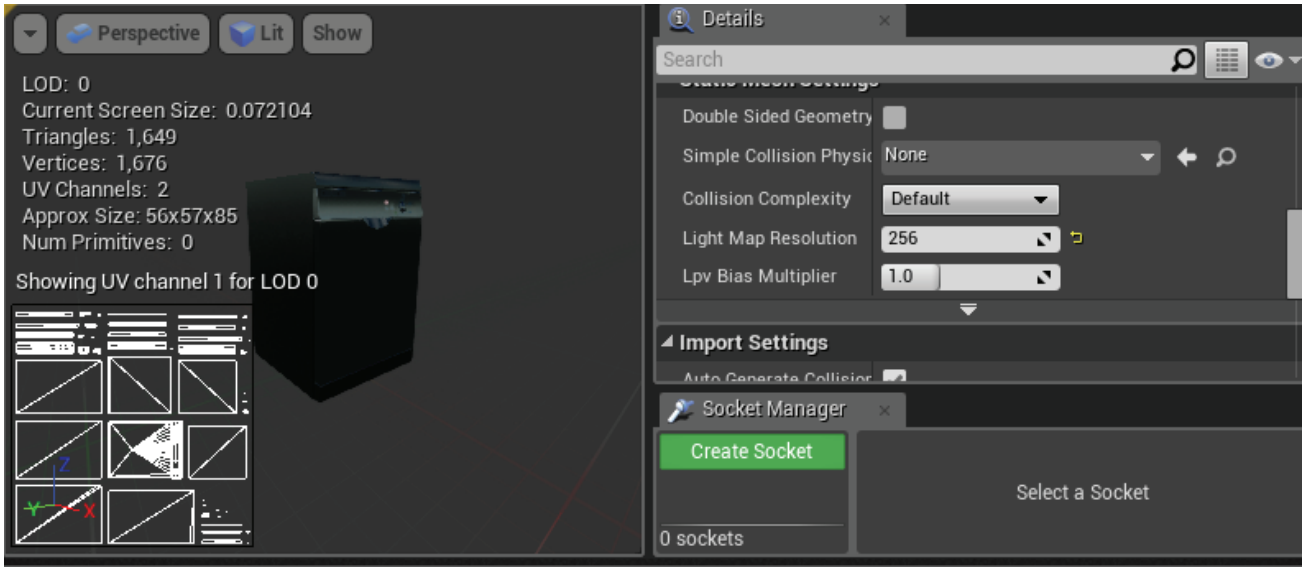
Y para poblar la escena de Assets tenemos este práctico control donde indicamos si rotamos, escalamos o movemos el objeto. Y luego un elenco de opciones clásicas, como el Gizmo (local o del mundo), el Grid o cuadricula (Muy útil sincronizado con 3DMax) e intervalos para las otras operaciones de escalado y rotado. Y por último el control de la velocidad de cámara.

4- Iluminación.

Como ya se ha comentado previamente el motor de iluminación del Unreal es Unbiased, es decir, no tiene capacidad de editarse con profundidad, así que aparte de la luz inicial del cielo del apartado anterior se colocaron tres luces estáticas en el sótano, provista de valores intuitivos. Funcionar funciona, pero no queda del todo bien.

La iluminación ha sido el apartado más descuidado del proyecto, y debería ser probablemente uno de los más importantes, ha cumplido porque el Unreal provee de una configuración por defecto excelente, pero queda como materia pendiente para proseguir este estudio. Cundo se trate de conseguir efectos muy realista, la iluminación es de vital importancia.

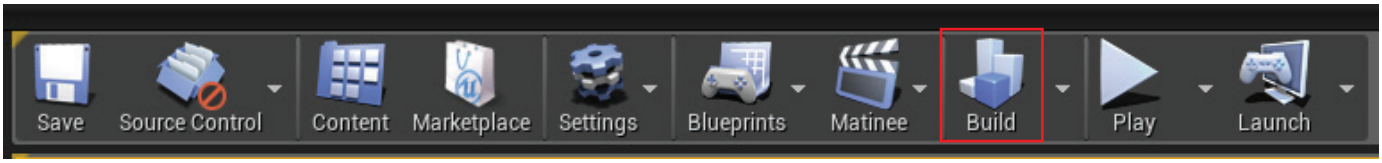
Como explicábamos en el apartado 2 los objetos debían tener un segundo canal en las UVs para el lightmap es en este apartado donde interviene.



En la imagen podemos observar a la izquierda abajo el segundo canal de UVs realizado, donde se guardan las luces y las sombras que reciben el objeto. A la derecha, la opción que dice “Light Map Resolution” indica que este objeto tiene un mapa de sombras y luces de 256x256 de textura. Lo que es en principio una calidad aceptable para las sombras.

Cuando hablamos de calidad de texturas en una escena hablamos de Texel. El texel es el indice de pixeles cuadrados por metro cuadrado. A mayor Texel mayor definición. En estos escenarios, el Texel de las texturas que colorean el objeto suele ser bastante mayor al texel de las sombras y las luces, que por lo general se deja bajo para que el ordenador no tenga problemas en calcularlo.

Se procede a darle al botón build, y el motor de iluminación hara el resto.

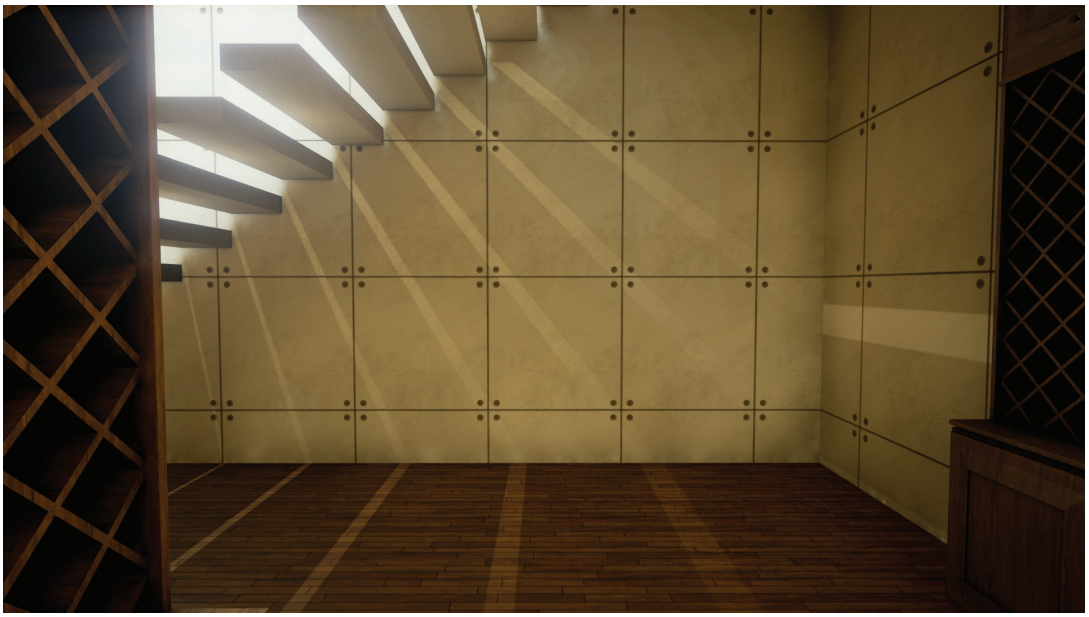


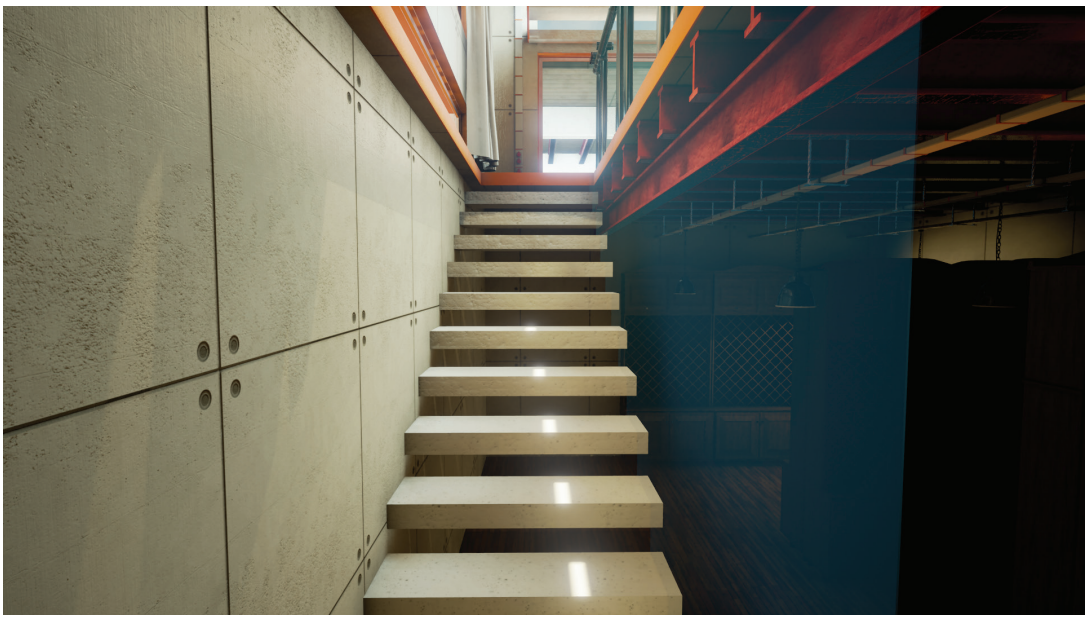
5- Conclusión del caso práctico y valoración personal.

Conclusión del caso práctico.

A continuación una captura de la escena ya finalizada.

El resultado final del caso práctico ha sido satisfactorio. Se podría haber realizado mejor trabajo pero se cumplió con la premisa inicial, intentar replicar el primer caso pero a mayor escala. Ha sido un éxito porque para ser un motor gráfico de videojuegos que renderiza 60 veces por segundo, su acabado final no dista tanto del primer caso y eso es lo que se quería comprobar. Es verdad que también ha habido un proceso de renderizado asíncrono en el Unreal (No-Tiempo Real), pero este solo se tiene que realizar una única vez, ya se conservará siempre en los LightMaps. Ignoramos si VRay es capaz de hacer esto mismo, al igual que ignoramos muchísimas cosas de la imaginería tridimensional, ya que es seguro que tanto el Unreal Engine como el VRay son capaces de dar mucho más de sí, el primer y segundo caso hacen flaco honor a su potencial técnico.





Tras ver las imágenes finales del trabajo, la auténtica conclusión, es que comparar la calidad gráfica de ambas imágenes es injusto. En este apartado, donde nuestro segundo caso siempre saldrá perdiendo, en otras muchas situaciones y necesidades del mundo real, saldrá ganando. Recientemente hemos descubierto que nuestro pequeño experimento tuvo lugar hace un año. El canal de Youtube UE4Architecture[1], lleva bastante tiempo trabajando con este software para lograr espacios virtuales realistas, profesionales, de diseño y de gran calidad. Este hecho demuestra que otros han invertido en proyectos de similares características porque también creen en su potencial. Es demasiado pronto para hablar de cambios profundos en el sector de la infografía, pero desde luego podemos confirmar que efectivamente, este método de trabajo se está adoptando y que solo es cuestión de tiempo que su mercado objetivo crezca y de trabajo directo e indirecto a una gran cantidad de profesionales.

Valoración personal.

Para mí ha sido bastante duro tener que aprender esta ingente cantidad de conceptos nuevos. Sobre 3D, sobre 2D, sobre iluminación, sobre fotografía, composición, diseño, etc... Pero la experiencia ha sido muy buena y me ha encantado trabajar en este proyecto, cuya produción ha durado unas 400 horas y me han dejado una carpeta de 71 GB en el ordenador. Estoy satisfecho con el trabajo y contento con las conclusiones que se han extraído de él, que aunque simples, necesarias para avanzar.

1: UE4 Architecture. LINK: https://www.youtube.com/channel/UCjpFOf-YWpois6WVG6hqXNQ

CONCLUSIONES DEL ESTUDIO

Tras finalizar este estudio hemos podido cuantificar ciertas cosas que pueden arrojar algo de luz sobre la temática que nos ocupa. Se decía al principio de la memoria, esto no es una competición entre VRay y Unreal Engine 4 pero se ha tratado como tal para intentar sacar ciertas conclusiones.

VRAY	UE4
Para una misma escena cada render cuesta entre 30 minutos y 2 horas, dependiendo de la complejidad de esta.	Tras un único render de 5-30 minutos, el motor renderizará 60 cuadros por segundo, permitiendo moverse por la escena en tiempo real.
Existen librerías de modelos de pago y gratuitas, de alta y baja calidad respectivamente.	La cantidad de modelos disponibles es limitada o cara en su defecto, así que se deberá pasar por una etapa de modelado.
Editor de materiales complejo con soporte para una gran cantidad de efectos de forma nativa. Programación cerrada.	Editor de materiales complejo pero con escaso soporte nativo de ciertos efectos, hay que trabajar mucho los nodos para lograr cosas que merezcan la pena.
Motor de iluminación de gran potencia, capaz de obtener imágenes prácticamente reales.	Motor de iluminación potente, su realismo no llega a ser perfecto, pero con pericia puede obtener resultados prodigiosos.
Motor de luz altamente configurable, con una gran cantidad de parámetros.	Menos parámetros, pero con posibilidad de tocar el código por debajo.
Obtienes imágenes de gran editabilidad para procesos posteriores.	Sistema cerrado, no se obtiene nada de él a menos que se usen chromas en el motor, ingenioso, pero limitado.
Interactividad nula. Lo que obtienes va a resultar en información unidireccional para el cliente.	Gran interactividad ofreciendo experiencias con posibilidades prácticamente infinitas.
Compatible con la realidad virtual.	Compatible con la realidad virtual, con el añadido de jugar con ella.
Precio de producción indeterminado, poco regulado, desfavorece al profesional.	Precio de producción indeterminado, poco regulado, desfavorece al profesional.

Tras comparar sus cualidades podemos ver que la elección siempre estará basada según la necesidad que se quiera cumplir, al principio erróneamente se pensó que se tenía que declarar un vencedor, pero eso sería totalmente simplista y además perjudicaría la imagen que se pudiera hacer el lector. Definitivamente, se ha analizado, se han mostrado datos suficientes, se han hecho hasta opiniones, ahora saquen sus propias conclusiones.

APLICACIONES REALES DEL ESTUDIO

-Aplicaciones docentes:

Como se ha demostrado, estas técnicas de renderizado en tiempo real pueden coexistir con las usadas actualmente e incluso ir reemplazándolas poco a poco. Tomando en cuenta los posibles riesgos y beneficios, se puede ir elaborando una oferta formativa de cursos y másteres en iluminación, modelado y animación con el uso de estos Engines.

Obviamente existen retos, porque para que estas tecnologías se adopten de forma masiva resulta complicado de predecir. Pero el experto en arte e iluminación con Unreal Engine 4, es un tipo de perfil que cada vez es mas buscado por empresas de videojuegos[1], cine y infoarquitectura. Cine por casos como ILM y su robot imperial en Star Wars Rogue One[2] que fue renderizado en UE4, o como Benoit Dereau, arquitecto francés que empezó a adoptarlo para realizar sus conceptos arquitectónicos y de interiorismo[3].

-Aplicaciones derivadas de la docencia.

Aparte de el uso en videojuegos cine y arquitectura, nos encontramos con que se esta empezando a usar en muchos otros campos.

Se está empezando a usar en medicina para la representación de gráficos biomédicos, anatómicos y forenses, sobretodo entralazandose con su poder docente en la propia carrera de medicina[4].

Tiene gran potencial para renovar los desfasados sistemas de gráficos interactivos en museos y parques temáticos.

Se empezado usar en diversas ferias y exposiciones para ampliar el poder interactivo de los expositores[5].

1: https://www.theverge.com/2015/3/4/8150057/unreal-engine-4-epic-games-tim-sweeney-gdc-2015
2: https://www.polygon.com/2017/3/1/14777806/gdc-epic-rogue-one-star-wars-k2so
3: https://www.idealista.com/news/inmobiliario/vivienda/2015/02/02/734379-realidad-o-ficcion-alucinaras-este-impresionante-apartamento-hasta-que-te-des-cuenta
4: http://www.baboonlab.com/blog/noticias-de-marketing-inmobiliario-y-tecnologia-1/post/realidad-virtual-y-medicina-usos-y-aplicaciones-27
5: https://www.youtube.com/watch?v=FI-QCgBe0rs

CARTA DEL AUTOR

Concretamente el “segundo caso” es el futuro profesional al que aspiro, la vida del artista 3D tampoco es que sea muy cómoda, gracias a mis andanzas en este mundillo he conocido a grandes amigos y compañeros que han ayudado en esta tarea, y en ellos he visto que es una vida de artista ante todo, te comes lo que produces y producir es durísimo por los altos estándares de la industria, aspirar a una vida acomodada es complicado dedicándose a esto, y viendo la cantidad de horas que se pasan frente a la pantalla, te acabas planteado si realmente compensa.

Si queda un poco de espacio para la sensibilidad en esta sociedad, aunque no sean ni el momento ni el lugar adecuados quiero añadir que aunque es normal desmotivarse un poco, siempre es apasionante pensar que se trabaja dentro de un campo construido sobre un mar de abstracción que es la informática, para mí el conocimiento, que aunque lógico, vendría a ser el más abstracto de imaginar posible. Cuando veo un videojuego en una pantalla, y pienso, son minerales, trozos de metal y de plástico y aun así son capaces de contener todo un mundo en su interior, maravillarme es lo mínimo que hago. Siempre he considerado que esto de los ordenadores era una magia e ir revelando incógnitas les resta un poco de gracia. Pero aun así, pese a que el listón este en órbita, me siento orgulloso de este proyecto y solo espero que sea algo realmente interesante, y que al salir de la defensa del trabajo, salga más convencido de esto y más motivado que nunca, preparado para la aventura del mundo laboral. Si estás leyendo esto porque realmente te interesa y quieres aprender te animo a que sigas adelante, en este sector hay grandes personas y por duro que sea todos acuden a prestarte su ayuda, los artistas son altruistas por naturaleza, no he visto nunca gente tan entregada, gente que valora la belleza que se ve tanto como la que no se ve, y que saben encontrarle a todo su punto especial. Si, te animo a que te subas a este barco donde todos somos camaradas.

Quiero agradecer la ayuda recibida por Elias Vergara Navarro que me ha ayudado muchísimo en esto y sin su ayuda el proyecto tendría la mitad de la calidad. También gracias a mi familia y amigos por el apoyo que me han dado y gracias también al tutor del proyecto Jorge Alamán Garcera, por atreverse a tutorizar semejante ambigüedad de proyecto.

FUTURAS APLICACIONES.

Quiero aprovechar el juego que da esto para hacer aproximaciones acerca del futuro de estas tecnologías. En un tono un tanto rimbombante y de poca seriedad, quizás no muy adecuado para un trabajo de estas características

No es nada nuevo que las gafas de realidad virtual se llevan desarrollando a un buen ritmo, pero que en realidad ni siquiera Michael Patcher sabe bien qué futuro les aguarda. Después de que cientos de compañías se subieran al carro, yo también mi quise subir al carro con este proyecto y al principio del mismo tuve la oportunidad de probar las Oculus Rift DK2, si bien me decepcionaron un tanto no se puede negar que esta tecnología es impresionante y que bien podría haber venido para quedarse. No como el estrepitoso fracaso de Virtual Boy de Nintendo.

Bien, estoy seguro de que cuando esta tecnología se mejore, aparecerán cientos de maravillosos usos, los cuales están sujetos a pura especulación, pero combinamos la realidad virtual con el Unreal Engine 4 y obtenemos un mundo de posibilidades al alcance de cualquier indie del sector y no tan indie. Divaguemos pues acerca de este dúo dinámico.

Amueblar tu casa con Ikea, Unreal Engine, Oculus Rift, Omni Virtuix y PS Move. Suena a alianza poco convencional, pero si estos artefactos se uniesen, proveyéndoles de unos pocos datos podríamos pasear por nuestra casa amueblándola sin dejarnos la cartera ni la espalda en el proceso, finalmente cuando estuviera a gusto, hacer el pedido y deslomarse para montarlo todo.

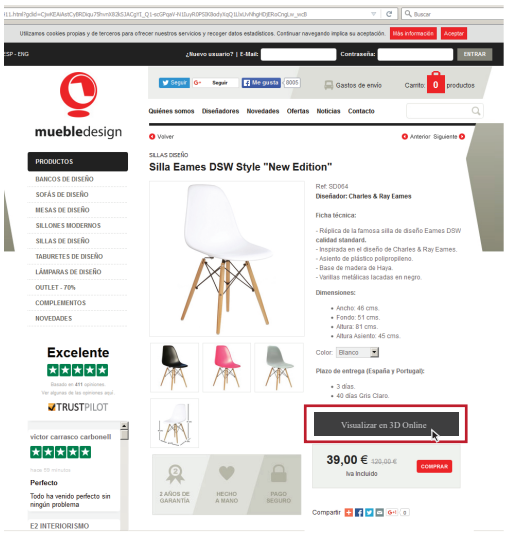
Pasear por otro periodo histórico con Unreal Engine, Oculus Rift y un sensor de Tracking, Imagina ver la antigua Roma o las ciudades Mayas en todo su esplendor sin tener que producir una brecha espacio temporal que acarreasen paradojas y el fin del universo. Desde luego suena a negocio que podria funcionaria, se pagaría lo que fuera, Google escúchanos.

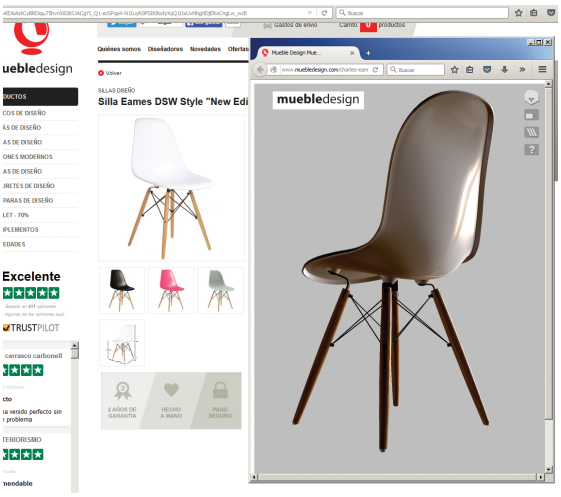
Entrenamiento militar, es difícil estar bien preparado para algo cuando un simulacro militar no es convincente, Unreal Engine 4 y Oculus Rift ofrecen la posibilidad de entregar una simulación tan real que puedas sentir el calor de la conflagración, y no solo por el tema gráfico a motores más sofisticados mejores capacidades para la inteligencia artificial y otros factores que cuestan procesador.

Se dice que hemos perdido la empatía, tal vez, combinar la realidad virtual con efectos sensitivos, pueda lograr a las personas vivir situaciones que logren engañar a la mente y hacernos ponernos en el lugar de los menos afortunados, refugiados de guerra, víctimas de violencia de género o racista, etc... Esto podría ser una manera de concienciar a la gente, una manera de reeducar presos o en última instancia un sórdido negocio para morbosos.

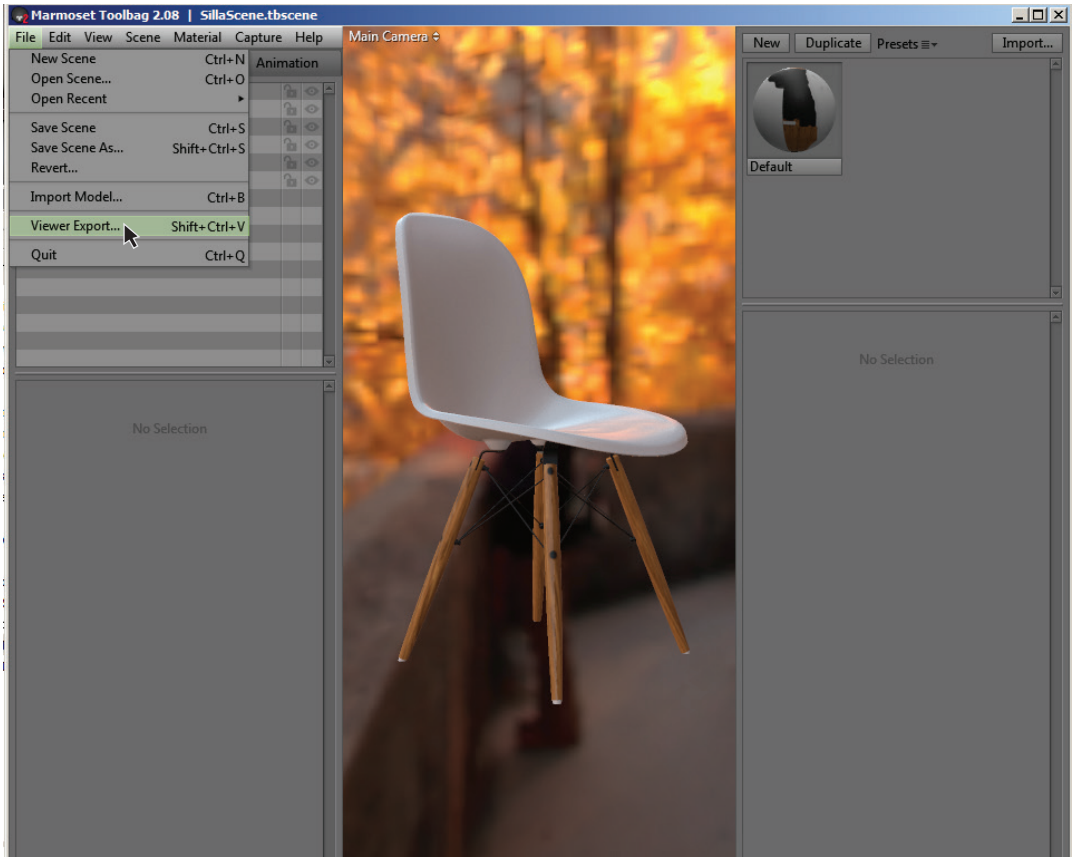
De estos ejemplos visualizar contenido tridimensional online ya es técnicamente posible aunque no es común. A continuación pasamos a illustrar estos ejemplos:

Utilizar la tecnologia provista por Marmoset Toolbag 2, un motor de render StandAlone PBR en tiempo real, que nos permite incrustar su tenologia en una página Web en el marco HTML5.





Tan fácil como, usar el Viewer Export incluido en el Visor, muy fácil de usar.



BIBLIOGRAFÍA

http://moviesincolor.com/

https://www.youtube.com/watch?v=b6KJOb-r16w

https://www.youtube.com/watch?v=hlg_O75jXkE

http://imgur.com/a/dEWXP

https://www.youtube.com/watch?v=sGN7Wby2thk

https://www.youtube.com/watch?v=0s0XLuzQHJ0&feature=youtu.be

https://www.youtube.com/watch?v=-_BFWjqNV-s

http://jamiesjewels.typepad.com/jamies_jewels/2011/11/the-full-100-3ds-max-tips.html

http://www.polycount.com/forum/showthread.php?t=136390&page=2

https://www.youtube.com/watch?v=FsFVtJd6qEw

https://www.youtube.com/watch?v=_MU3M6xqhe4

https://www.youtube.com/watch?v=URG3YNsUCQ8&index-
=9&list=PLZlv_N0_O1gak1_FoAJVrEGiLIploeF3F

http://www.marmoset.co/toolbag/learn/pbr-conversion

https://www.youtube.com/watch?v=UIt-nemQopE

http://www.rendertextures.com/category/effects/ies-lights/

https://forums.unrealengine.com/showthread.php?43784-Se-
rie-de-tutoriales-en-espa%C3%B1ol-sobre-Unreal-Engine-4

http://wiki.polycount.com/wiki/Texture_Coordinates

http://www.fxguide.com/featured/game-environments-parta-remember-me-rendering/

https://www.youtube.com/watch?v=V3FL8jvvFDw

https://www.youtube.com/watch?v=D3N9aK977Ew

http://cgpersia.com/2014/12/gumroad-ue4-materials-for-begin-
ners-vertex-painting-water-puddles-by-aaron-kaminer-58796.html

https://www.youtube.com/watch?v=PvL0W_sPZKI

http://www.world-machine.com/library/index.php?type=0&sort=best

https://www.youtube.com/watch?v=5dkP-XcLSG4

https://www.youtube.com/watch?v=K9WTKK9f1b8

https://forums.unrealengine.com/showthread.php?530-How-to-enable-Light-Propagation-Volumes-GI-WIP-AND-BETA

https://www.youtube.com/watch?v=DqqeeZya-J8

https://www.youtube.com/watch?v=p0o3bqoM0Qg&feature=player_embedded

https://www.youtube.com/watch?v=P-5Qzi-ljsc

https://www.youtube.com/watch?v=o3L-GlYWmpc

https://www.youtube.com/watch?v=D6zyUI33FqA

https://www.youtube.com/watch?v=OjOhQ1DeQ8s

http://wiki.polycount.com/wiki/Subdivision_Surface_Modeling

http://www.geekatplay.com/world-machine-tutorials.php

https://www.youtube.com/watch?v=sRQ5z8i_SV8

https://docs.unrealengine.com/latest/INT/Engine/Rendering/LightingAndShadows/Lightmass/index.html

https://www.youtube.com/watch?v=yeczjVAqznM

http://giancr.com/es/modelado-poligonal-esencial-pequenos-detalles-parte-02/

http://www.svartberg.com/tutorials/article_normalmaps/normalmaps.html

https://www.youtube.com/watch?v=gOdTrIB5te0

https://www.youtube.com/watch?v=CtyNXjN_HFk

https://forums.unrealengine.com/showthread.php?3124-Changing-resulution-of-exported-game

https://www.allegorithmic.com/substance-ue4

http://www.fxguide.com/featured/epics-unreal-engine-open-world-behind-the-scenes/

https://docs.unrealengine.com/latest/INT/Programming/Tutorials/index.html?utm_source=newsletter_may&utm_medium=email&utm_campaign=4dot8

https://www.youtube.com/watch?v=KPmRV1Z9ikY

https://www.youtube.com/watch?v=Ux9VFZL_DcM

https://docs.unrealengine.com/latest/INT/Resources/Showcases/Reflections/index.html

https://docs.unrealengine.com/latest/INT/Engine/Basics/Projects/Packaging/index.html#settingagamedefaultmap

https://www.youtube.com/watch?v=ZFSXtjFI9ig

https://docs.unrealengine.com/latest/INT/Engine/Rendering/Materials/HowTo/Fresnel/index.html

https://docs.unrealengine.com/latest/INT/Engine/Rendering/Materials/PhysicallyBased/index.html

https://es.wikipedia.org/wiki/Estado_del_arte
https://www.youtube.com/watch?v=-bLOi3mo9NE

https://www.vg247.com/2015/03/04/unity-5-available-for-free-with-no-fucking-around/

http://madewith.unity.com/

https://www.youtube.com/watch?t=19&v=CLPBFlA1DAw

http://blog.digitaltutors.com/unity-udk-cryengine-game-engine-choose/

http://cryengine.com/showcases/ryse-son-of-rome

http://www.polygon.com/2014/4/6/5588150/half-life-2-looks-great-in-unreal-engine-3

http://www.polycount.com/forum/showpost.php?p=2037585&postcount=3431

http://www.gamingdragons.com/images/game_img/screenshots/bf3/-Battlefield-3-screenshots-Staging-Area.jpg

https://docs.unrealengine.com/latest/INT/Engine/Rendering/Materials/MaterialProperties/LightingModels/index.html

https://forums.unrealengine.com/showthread.php?60501-WIP-Hair-Material-with-anisotropic-reflections

https://forums.unrealengine.com/showthread.php?63555-FREE-Hair-Material

https://dl.dropboxusercontent.com/u/4817133/UnrealForums/HairMaterialV1.png

https://docs.unrealengine.com/latest/INT/Engine/Paper2D/Sprites/index.html

http://www.polycount.com/forum/showthread.php?t=99335&page=54

http://www.neilblevins.com/cg_education/translucency/translucency.htm

https://docs.unrealengine.com/latest/INT/Engine/Rendering/Materials/PhysicallyBased/index.html#hable

http://www.marmoset.co/wp-content/uploads/materialref02.png

http://www.marmoset.co/viewer/gallery

http://www.marmoset.co/toolbag/learn/pbr-practice

https://docs.unrealengine.com/latest/INT/Engine/Rendering/PostProcessEffects/ColorGrading/index.html