

Editor de materiales para modelos en 3D basado en nodos.

Grado en Ingeniería Multimedia



Trabajo Fin de Grado

Autor:

Santiago Palacio Caro

Tutor/es:

Rafael Molina Carmona

Septiembre 2015



**Justificación y Objetivos.**

**Motivación.**

Diferentes son los motivos por los cuales el autor de este trabajo decidió realizarlo. Cabe volver mencionar que se trata de un trabajo de fin de grado, por lo tanto, es un requisito indispensable para todos los estudiantes del régimen estudiantil, el presentar y superar dicha prueba para obtener el título de ingenieros que este conlleva. No obstante, el enfoque que se tiene de éste no es la de un obstáculo, sino por el contrario, la de una grandísima oportunidad para demostrar las habilidades conseguidas durante todo el periodo estudiantil y que mejor forma de hacerlo que con un trabajo de tema a elegir por el estudiante. Un desafío nunca mejor recibido y que se afrontó con toda la pasión y esfuerzos posibles.

Dicho lo anterior, también he de mencionar motivaciones más concretas fuera de lo que representa el trabajo de fin de grado como requisito para la obtención del título.

El estado actual del arte y de la industria involucrada en mantener sistemas en 3D es cada vez más complejo, aportando soluciones y métodos únicos que permiten obtener resultados cada vez más interesantes y logrados. La realización del trabajo aquí descrito es también motivada por el gran interés que despiertan las mencionadas tecnologías en el autor del mismo. Así pues, uno de los principales focos de atención dentro de un entorno de gráficos en 3D es la posibilidad que dichos sistemas tienen de aportar a cada modelo en 3D un aspecto único, un material. Un material que le dota de un aspecto en ocasiones realista, estilizado o abstracto, según la finalidad.

Es en este punto donde el arte y la tecnología se unen en una simbiosis perfecta y es la que le permite a artistas e ingenieros gráficos el lograr grandes creaciones a través de su trabajo en combinación. Es aquí donde este trabajo busca conseguir unificar de una forma nueva y más sencilla esa combinación entre los artistas y los ingenieros. Es por eso que la decisión tomada fue la de crear una herramienta para crear materiales para objetos en 3D, sencilla y fácil de usar, que permita al usuario que lo use, entender el proceso por el cual pasan los modelos en 3D hasta ser dibujados de una forma más intuitiva y más aún, entender el estado de Shading en donde se le asignan los colores a las superficies de los modelos teniendo en cuenta tanto parámetros del material como parámetros de la luz puntual dada.

Una vez tomada la decisión de desarrollar una herramienta para crear materiales y por consiguiente **Shaders**\*, se detectó un modelo frecuente de este tipo de aplicaciones en los principales productos de la industria actual – motores gráficos, herramientas de modelado, etc.-. Dichas aplicaciones tienen en común que todas son capaces de generar materiales usando una interfaz basada en nodos. Esto simplifica de una manera considerada la forma en que se generan dichos materiales, entre otros motivos, porque se sustenta de las virtudes de un VPL (visual programming language) como el fácil entendimiento del funcionamiento del programa debido a que está basado en gráficos y grafos intuitivos, entre otros.

Una aplicación basada en nodos y que genere las operaciones necesarias para un determinado material en función del grafo de nodos que se haya diseñado de antemano. Es aquí donde el artista sin conocimientos de programación, gráficos en profundidad, y HLSL/GLSL puede aportar todo su talento sin problema alguno.

En definitiva, éste trabajo es considerado como un buen – y desafiante- proyecto a desarrollar como parte del TFG ya que implica tener los suficientes conocimientos que conlleva este tipo de aplicaciones. Conocimientos que ha sido adquiridos durante los últimos años de carrera universitaria y que serán puestos a prueba en dicho proyecto

**Agradecimientos.**

**Dedicatoria.**

**Citas.**

**Índices.**

**Cuerpo del documento.**

**Conclusiones.**

**Bibliografia y Referencias**

**Estado del arte.**

El término “editor de materiales “es un concepto bastante extendido en la actualidad (2015) en contextos de gráficos 3d por computador. Si bien existen gran variedad de herramientas que permiten de alguna forma proveer, modificar y administrar y los parámetros que caracterizan un determinado material sobre una superficie o una malla geométrica, cabe destacar algunas características que son diferenciadoras entre unos y otros editores de materiales.

**Altamente integrado con el motor gráfico VS Independientes.**

En determinados entornos, los editores de materiales son parte de un sistema mucho más complejo y funcionan como sub-sistema dentro de un motor gráfico o sistema más generales. Dichos editores funcionan de forma complementaria al sistema de renderizado de sus motores, estableciendo integridad en la parametrización de valores de cara a su ecuación de renderizado a varios niveles de complejidad. En otras palabras, existe una alta cohesión entre lo ofrecido por el editor de materiales y su puesta en escena por parte del motor, donde otros procesos entran en juego como la iluminación global, el sombreado de la escena y otros efectos más avanzados como el desenfoque de movimiento basado en mapeado de movimientos o la iluminación volumétrica etc. Todos ellos hacen parte del dibujado final de la escena pero no así de las propiedades físicas del material.

Hacer referencia a la rendering equation como función de dibujado final, y a la BDRF como intento por definir la reflectividad y color del objeto 3d ¿Aquí o en otro sitio anterior mejor?

Un ejemplo claro de este tipo de herramientas son las ya incorporadas en motores comerciales de alto desempeño como el motor gráfico Unreal Engine 4 (UE4) de Epic Games Inc. o el motor gráfico CryEngine de Crytek.

Por otra parte, si bien hemos dicho que existen editores pensados para un determinado motor de renderizado en tiempo real, también se da el caso de editores de materiales que por el contrario no están diseñados para un sistema de renderizado especifico, manejando conceptos y parámetros más genéricos y que pertenecen mas al “estándar” de gráficos por computador permitiendo dibujar los materiales producidos bajo distintos motores de renderizado. Un caso claro de esto son los editores de materiales incluidos en las aplicaciones de edición 3d. En estas es común realizar el trabajo de edición de los materiales y luego elegir el motor de renderizado. Un buen ejemplo se da en el programa 3ds Max, donde podemos elegir por defecto entre el motor de renderizado Scanline o el motor de renderizado MentalRay. Además existen motores de terceros como el motor V-Ray que pueden ser añadidos y usarse para el renderizado dentro de 3ds Max.

Llegados a este punto cabe puntualizar que aunque estos editores manejen parámetros más generales, también pueden disponer de otros más avanzados en función del motor de renderizado que se va a utilizar, pero nunca abandonando los valores estándar. En el caso de 3ds Max, es frecuente ver como el editor de materiales se ve modificado añadiéndole nuevas interfaces que permiten el poder trabajar con dicho motor gráfico y sus parámetros específicos.

**Para renderizado en tiempo real VS renderizado offline.**

Una importante característica más a destacar, es también el hecho de que el destino de algunos de estos editores sean sistemas de renderizado en tiempo real o bien de renderizado offline. Esto puede ser importante ya que ambos sistemas difieren sustancialmente en aspectos clave como la calidad del resultado o el tiempo de respuesta final, aun así, aunque ambos manejen aspectos claves en común, se debe más a una herencia de modelos de iluminación clásicos ligados a las condiciones físicas de la naturaleza, más que a su forma de procesar el dibujado en sí.

Un claro ejemplo de esto se da en los parámetros básicos como de factor de especularidad de una superficie, o su valor de componente difusa. Si bien la forma en que un editor para renderizado en tiempo real los manejan respecto a otro offline puede resultar similar, a la hora de realizar el dibujado final, sus proceso internos son radicalmente distinto, ya que en la mayoría de los casos, un sistema offline usa métodos basados en Raytracing, los cuales son inviables para renderizado en tiempo real debido a su gran costo computacional.

Tipo de interfaz: Basada en Nodos VS Clásica.

Cons y pros de una y de otra

A continuación se hará un análisis de los editores de materiales que más han influido este trabajo y sus principales características.

**Editor de materiales de UE4.**

Este editor permite la creación de Shaders para aplicarlas sobre alguna superficie geométrica determinada. Dicho editor viene incorporado en el motor gráfico UE4 y hace parte de uno de sus subsistemas.

Su interfaz gráfica basada en nodos, es una de sus características más destacables. Es frecuente en UE4 ver dicho tipo de interfaces basada en nodos. Si bien sirve como interfaz de usuario para el desarrollo de materiales, también entra en la categoría lenguaje de programación visual (VPL) el cual permite programar y generar Shaders mediante la manipulación de elementos visuales, en este caso entre nodos y sus conexiones. El editor de materiales de UE4 permite la creación de una variedad considerable de nodos que bien representan una operación la cual encapsula una función determinada, o bien nodos que representan algún tipo de información como valores de color, o de mapas de texturas. Cada nodo está provisto de entradas y salidas a las cuales se les puede conectar otros nodos ya bien sea como entradas o como salidas. El número de entradas depende del tipo de nodo.

Este tipo de interfaz es común en otros subsistemas del UE4 como lo son los ***Blueprints***, en donde la finalidad no es crear materiales sino modificar o crear características del gameplay, cámaras y otros tipos de elementos de una forma más intuitiva y rápida, más aun para gente no programadora.

**Physically-based shading**

UE4 utiliza un modelo de shading basado en físicas (physically-based shading model). Los resultados finales son más precisos y con un aspecto típicamente más natural. El salto gráfico respecto a su predecesor Unreal Engine 3 (UE3) es considerable.

Los modelos basados en físicas (physically-based), son la característica de moda en cuanto a gráficos en tiempo real de nueva generación.

Su condición de physically-based le permite establecer propiedades de los materiales más intuitivas, más cercanas a la realidad y al cómo funcionan los fenómenos físicos de iluminación en la naturaleza. UE4 define un material basándose en 4 propiedades básicas: BaseColor, Metallic, Roughness y Specular.



*Nodo principal del panel de grafo en UE4. Por defecto este nodo está siempre presente. A él son conectados los inputs creados en el editor para generar el resultado final*.

La propiedad ***BaseColor*** define un color total para el material del objeto. Recibe como parámetro de entrada un vector de tres dimensiones donde cada componente representa los valores de intensidad para el estándar de color RGB. Estos valores son restringidos al rango [0-1].

Por su parte, la propiedad ***Roughness*** controla el nivel es aspereza del material. Un material con un alto nivel Roughnessrefleja luz a mas direcciones que uno con poco nivel dejando una luz especular más difusa y borrosa, alternativamente un material con un nivel de Roughness más bajo refleja luz en parcialmente una misma dirección y tiene una luz especular mucho más definida tendiendo a imitar el comportamiento de un espejo.

La propiedad ***Metallic*** controla el nivel de metalizado de la superficie. Fenómenos como la cantidad de luz que es absorbida por el material son controladas por este parámetro contribuyendo a definir el nivel de metalizado del material. Un material con un valor de 1 se comporta como totalmente como un metal mientras que un material con un valor de 0 no poseería ninguna característica metálica (rocas, madera, etc.).

La propiedad ***Specular*** recibe un valor entre 0 y 1 y se utiliza para escalar la cantidad actual de especularidad en superficies no metálicas. Es decir, no tiene ningún efecto sobre los metales. Por defecto, si no se le pasa un valor, el valor inicial es de 0.5.

Si se modifica el valor de especularidad, se hace para añadir micro colusiones o sombreado a pequeña escala, representadas en el mapa de normales. Esto en ocasiones es referido como cavidades.

Geometría a pequeña escala, especialmente detalles solo presentes en el modelos high poly y luego generados (baked) sobre el mapa de normales, no son recogidas por las sombras del renderizador en tiempo real. Para capturar estas sombras, el motor UE4, genera un mapa de cavidades (Cavity), el cual es típicamente un mapa AO (Ambient Oclusion) con poca distancia de trazado. Este mapa es multiplicado por el *BaseColor* y posteriormente multiplicado por 0.5 (especularidad por defecto) como la salida final de la propiedad ***Specular***. Es decir:

BaseColor = Cavity \* OldBaseColor, Specular = Cavity \*0.5

Interfaz gráfica.

Aquí se describe la interfaz grafica

**Editor de materiales 3ds Max**

El editor de materiales de 3ds Max ofrece dos versiones en su interfaz de usuario: La versión compacta y la versión de pizarra (slate editor). Ambas opciones producen la misma calidad de resultados en términos técnicos. No obstante la diferencia entre ambos es considerable.

**Versión compacta:**

La versión compacta es representada en una pequeña ventana modal, invocada desde el la interfaz principal. Los materiales generados se ven en la parte superior de la ventana en forma de grid. Por defecto 3ds Max ofrece una serie de materiales a partir de los cuales se pueden empezar a alterar propiedades. Sin embargo el más típicamente usado en términos generales es el llamado “Standard”. Dicho perfil de material posee las propiedades más típicamente usadas y entendidas por gráficos por computador.

***Análisis de la interfaz gráfica***

La interfaz gráfica de 3ds Max, en su mayoría, está estructurada por rollouts\*. El modelo estándar tiene como principales rollouts a:

* Shader basic Parameters: Aquí se puede definir el mediante que función de reflexión se ha de interpretar el material. Entre sus opciones más destacadas se encuentran: Blinn, Phong, Metal y etc.
* Basic Parameters: Sus opciones cambian según el modelo de reflexión elegido en el rollout anterior. Por lo general presenta los valores más típicos del material:

**Ambient**: Constante de color que intenta simular la contribución lumínica de las reflexiones globales de la escena. Recibe un color rgb como parámetro de entrada.

**Diffuse**: Se trata del color base del material. La cantidad de luz que es reflejada en direcciones uniformes. Recibe un color rgb como parámetro de entrada.

**Specular** (Color): Define el color con el que las luces especulares son interpretadas. Recibe un color rgb como parámetro de entrada.

**Specular Level:** Indica el nivel de especularidad del material. Un material con un valor especular igual a 0 no reflejara luces especulares hacia la cámara, mientras que un material con un valor especular mucho más alto tendrá mucha riqueza de dichas luces en su superficie.

**Glossines:** Define el la fuerza de las luces especulares. De ello depende cuán grande son las luces especulares que son reflejadas. Si el valor es muy bajo, se producen luces más grandes con gradientes mayores. Si el valor es alto, la luz especular se decrementa al igual que lo hace el gradiente, dejando luces mucho más definidas y puntuales.

* Maps: En este rollout están presentes los slots a los cuales se pueden introducir distintos tipos de mapas para dotar al material de diferentes características. Por lo general, a cada mapa se le es asignado un componente de tipo Bitmap, pues es una acción típica el importar mapas para diferentes acabados en ficheros de tipo imagen y que son generados, generalmente, en otro tipo de programas como Photoshop. Sin embargo a dichos slots se les puede asignar otro tipo de componente. 3ds Max ofrece gran variedad de mapas, algunos por ejemplo, más procedurales capaces de generar mapas de ruido aleatoriamente como el componente “Noise”, u otros capaces de generar gradientes como el componente “Gradient”. Por lo tanto, estos valores no tiene por qué estar enteramente ligados a un mapa imagen, aunque suele ser lo más frecuente en el flujo de trabajo de la industria de gráficos en tiempo real como los videojuegos.

Entre los canales típicamente más reconocibles se encuentran:

**Diffuse Color:** Recibe un mapa (por lo general una textura), que aporta el color base a la superficie del material. Los valores de coordenadas UW también están a disposición del usuario para su manipulación.

**Specular Level:** Recibe un mapa de valores especulares.Este tipo de mapa, típicamente está en escala de grises y controla el comportamiento del índice especular a lo largo de todo el modelo.

**Opacity:** Recibe un mapa de opacidad donde el valor de los pixeles define la visibilidad de la superficie.

**Bump:** Recibe un mapa que, típicamente está en escala de grises y que define micro detalles y rugosidades a lo largo del material. Para ello, el mapa es capaz de modificar la dirección de las normales encapsulando valores numéricos sobre cada pixel, normalmente, en espacio de coordenadas tangencial.

**Versión Slate.**

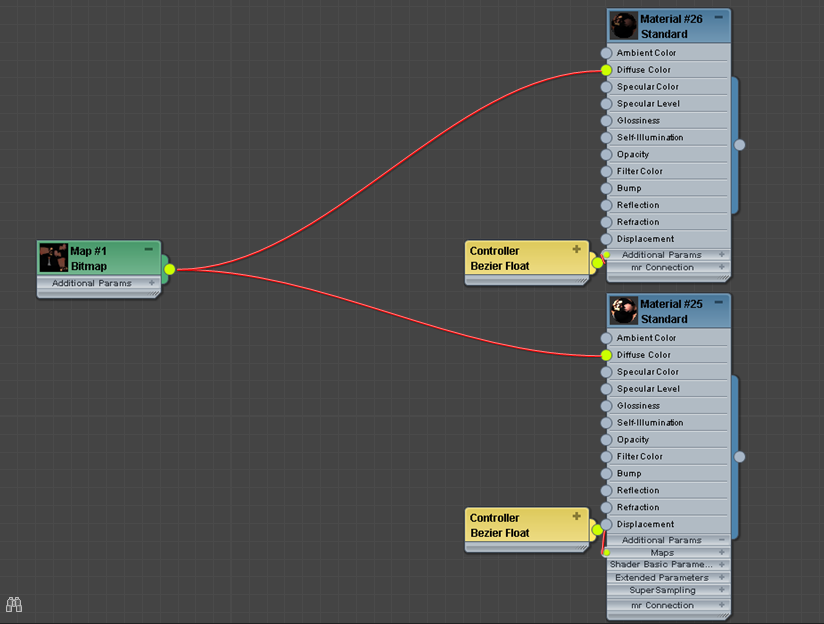
Esta versión es una aproximación más cercana a la propuesta en este proyecto en su interfaz gráfica.

Su interfaz de usuario deja de ser la convencional interfaz compacta para pasar a ser una más actualizada respecto a las tendencias actuales de edición de materiales. Dicha interfaz está basada en nodos, lo cual la hace mucho más adecuada a las demandas y flujos de trabajo actuales de este tipo de edición.

El espacio de trabajo tiene como componente principal un área donde se pueden crear nuevos materiales bien sea arrastrándolos de una lista de perfiles o haciendo clic derecho sobre dicha área y eligiendo el perfil del material a iniciar. En ese momento se crea un nodo, este nodo principal contiene una lista de inputs a los cuales se les pueden conectar nuevos nodos. Los inputs representan las propiedades expuestas para dicho material en dicho perfil.

*Nodo principal correspondiente a un material de tipo* ***Standard.***

En el mismo espacio de trabajo se pueden generar más materiales permitiendo al usuario reutilizar nodos o resultados de nodos entre distintos materiales que conviven en el espacio de trabajo. Lo cual ofrece mucha más potencia a la herramienta al igual que escalabilidad.



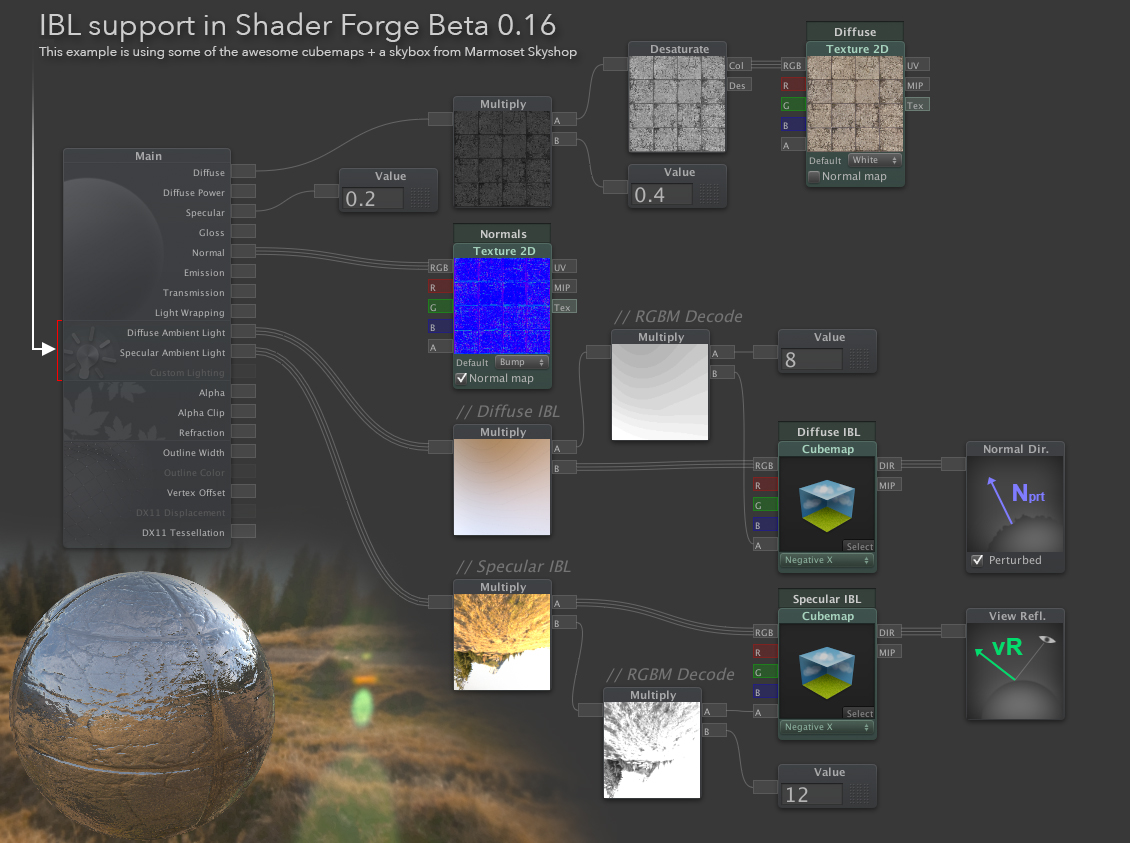
*Dos materiales compartiendo y reutilizando una textura representada por un nodo de tipo Bitmap*

**Shader Forge**

Se trata de un editor de Shaders basado en nodos. Está diseñado para que funcione bajo el motor gráfico Unity, de tal forma se presenta como una extensión de pago del motor. La licencia es adquirida a través de la Unity Asset Store – tienda de contenidos y extensiones de Unity-.

Shader Forge trabaja con modelos de iluminación basado en físicas (PBR) y shading de conservación de la energía haciendo uso de los modelos Blinn-Phong o Phong.

Además Shader Forge tiene la capacidad de poder trabajar con *Image Based Lighting* lo que le permite trabajar con técnicas como la de *Cube Maps* para visualizar la forma en que se iluminan los objetos, y generar valores para *Diffuse Ambient Light* y *Specular Ambient Light*. Gracias a esto Shader Forge es compatible con shaders y herramientas de la familia Marmoset (Marmoset Skyshop).



*(Ejemplo de un Shader creado por Shader Forge usando Skyboxes de Marmoset Skyshop)*

Su flujo de trabajo intuitivo es perfecto para artistas quienes no está muy habituados a programar ni a generar shaders mediante programación de texto.

El nodo principal de Shader Forge está compuesto por 21 inputs a los que se les puede pasar un valor que modificara los parámetros del material y de la iluminación del objeto al que se le aplique el Shader. Entre los 21 inputs se encuentran algunos clásicos como Diffuse, Specular y Normal, así como también algunos más personalizados como Custom Lighting el cual le permite al usuario definir un comportamiento de iluminación personalizado sin atender a las físicas de iluminación preestablecidas por el modelo de Shader Forge.



*(Conjunto de inputs de Shader Forge)*

Shader Forge dispone de una número de operaciones considerable que permiten la creación de nuevos valores resultado de operaciones realizadas sobre otros mismos. Operaciones como suma, resta, interpolación, mínimo, máximo, valor absoluto, clamp, blend (fundido), etc. son típicas en el desarrollo mediante Shader Forge.



*(Ejemplo de una operación de suma entre un nodo de tipo textura y un vector representando un color del tipo RGB)*

**Marmoset Toolbag**

Marmoset Toolbag es una herramienta principalmente de renderizado de alta calidad basado en físicas (PBR).

Marmoset Toolbag permite a artistas visualizar su trabajo en un render de alta calidad sin necesidad de utilizar un motor gráfico para lograrlo. Esto proporciona un estado de estandarización más amplio ya que muchos artistas pasan antes por Marmoset Toolbag que por el motor y empieza a convertirse en un referente de testeo para materiales en PBR.

La calidad de su render es la principal característica de Marmoset Toolbag. Está pensado para visualizar escenas generalmente pequeñas, con iluminación basada en mapas (Cubemaps y Skyboxes) para modelar el comportamiento en el que la luz refleja determinado modelo. Además de poseer una gran gama de características de renderizado que le dotan de gran vistosidad y realismo en sus escenas.



*(Modelo 3D renderizado con Marmoset Toolbag en tiempo real.*

*Imagen por: Colt Python by Benjamin Turner | turnededges.com)*

Entre los parámetros que define un material en Marmoset Toolbag se encuentran: **Albedo,** (define el color base del material), **Specular, Gloss y Normales.** Otras características de materiales que se pueden encontrar en Marmoset Toolbag son:

* Dynamic Tessellation
* PN Triangle mesh smoothing
* Height & Vector Displacement modes
* Detail Normal Maps
* Parallax Occlusion Maps
* "Metalness" Maps
* GGX, Blinn-Phong, & Anisotropic Reflections
* Secondary Reflections
* Skin Diffusion
* Microfiber ("Fuzz") Diffusion
* Occlusion and Cavity Maps
* Emissive Maps
* [**Allegorithmic**](http://www.allegorithmic.com/) Substances
* Dithered Supersampled Transparency

***Fuentes***

[***http://acegikmo.com/shaderforge/***](http://acegikmo.com/shaderforge/)

[***http://www.marmoset.co/toolbag/learn/pbr-theory***](http://www.marmoset.co/toolbag/learn/pbr-theory)

[***https://docs.unrealengine.com/latest/INT/Engine/Rendering/Materials/PhysicallyBased/index.html***](https://docs.unrealengine.com/latest/INT/Engine/Rendering/Materials/PhysicallyBased/index.html)