Universidad Complutense de Madrid
Facultad de Bellas Artes
Departamento de Diseño e Imagen
Máster Universitario en Diseño

Virtual Production y *Performance Capture*.

Estudio de técnicas en tiempo real para la producción de efectos visuales en la industria audiovisual.

David Guillermo Martínez Rodríguez

Director/a/es
Jaime Munárriz Ortiz

Madrid, convocatoria de octubre 2020

UNIVERSIDAD COMPLUTENSE MADRID

ANEXO I: DECLARACIÓN DE NO PLAGIO

D./Dña. David Guillermo Martínez Rodríguez

con NIF 05447327F, estudiante de Máster en la Facultad de Bellas Artes de la Universidad Complutense de Madrid en el curso 20 -20 , como autor/a del trabajo de fin de máster titulado Virtual Production y Performance Capture: estudio de técnicas en tiempo real para la producción de efectos visuales en la industria audiovisual.

y presentado para la obtención del título correspondiente, cuyo/s tutor/ es/son:

Jaime Munárriz Ortiz

DECLARO QUE:

El trabajo de fin de máster que presento está elaborado por mí y es original. No copio, ni utilizo ideas, formulaciones, citas integrales e ilustraciones de cualquier obra, artículo, memoria, o documento (en versión impresa o electrónica), sin mencionar de forma clara y estricta su origen, tanto en el cuerpo del texto como en la bibliografía. Así mismo declaro que los datos son veraces y que no he hecho uso de información no autorizada de cualquier fuente escrita de otra persona o de cualquier otra fuente.
De igual manera, soy plenamente consciente de que el hecho de no respetar estos extremos es objeto de sanciones universitarias y/o de otro orden.

En Madrid, a 03 de octubre de 20 20

Fdo.: 

Esta DECLARACIÓN debe ser insertada en primera página de todos los trabajos fin de máster conducentes a la obtención del Título.

Índice

RESUMEN

Los efectos visuales forman parte de la esencia del cine ya desde los primeros años tras su creación por los hermanos Lumière. Recurrir a trucos de cámara y a ilusiones se convirtió rápidamente en un recurso no solo recurrente, sino necesario, para el desarrollo de todo tipo de películas a lo largo de la historia del cine; existía una relación simbiótica entre el espectador, que con el transcurso de las décadas demandaba artificios y efectos cada vez más elaborados, y los cineastas, quienes tomaron ventaja de los avances tecnológicos de su tiempo para llevar a la gran pantalla impresionantes escenas e inimaginables historias.

Con este trabajo se pretende explorar los avances relacionados con las artes digitales dentro del mundo audiovisual, cómo estas han evolucionado a lo largo de la historia del cine hasta llegar al punto en el que hoy nos encontramos. Se estudiarán los aspectos fundamentales que caracterizan a los gráficos generados por ordenador (CGI) y se abordarán dos metodologías de trabajo que han revolucionado la manera de integrar los efectos especiales de una producción audiovisual, a saber: la *performance capture* y la *virtual production*.

Será también objeto de reflexión el papel que tanto artistas como diseñadores juegan dentro de la industria, y las posibilidades de desarrollo profesional que los nuevos avances supondrán para estas profesiones.

ABSTRACT

Visual effects have been part of the essence of cinema since its early years after Lumiere brothers created it. Camera tricks and visual illusions were not just a frequently used resource but a necessity in many film productions along cinema history. There was a symbiotic relationship between audience, who throughout decades had demanded more elaborated tricks and effects, and filmmakers who took advantage from the technological advances of their time to bring astonishing scenes and unbelievable stories to theater screen.

This essay is meant to explore the advances related to digital arts within the audio-visual field and in which way those have evolved during history until they reached nowadays state-of-art. On this document, the main features that characterize CGI will be explored, focusing on performance capture and virtual production as the most innovative way to integrate VFX in real time.

The role artists and designers play within the cinema industry will also be a relevant case of study, as well as the possibilities new technologies bring to all kind of audio-visual related professions.

PALABRAS CLAVE

CGI, VFX, Arte 3D, captura de movimiento, *performance capture*, *motion tracking*, producción virtual, *Unreal Engine*.

KEYWORDS

CGI, VFX, 3D Art, motion capture, performance capture, motion tracking, virtual production, Unreal Engine.

0. INTRODUCCIÓN

El objeto de esta investigación es analizar las diversas técnicas de captura de movimiento, así como la integración directa de CGI (del ingés *computer-generated imagery*) durante el proceso de producción en medios audiovisuales. Será también relevante el análisis de nuevas dinámicas surgidas dentro de la industria audiovisual y el uso de estrategias alternativas para el diseño de metodologías de trabajo ágil.

El perfeccionamiento de *hardware* orientado a la generación de entornos 3D, y los recientes avances en sistemas de renderizado en tiempo real, han permitido la integración de gráficos digitales durante la etapa de rodaje en una producción cinematográfica, algo que resultaba completamente imposible hace menos de cinco años. Esta nueva situación ha abierto un horizonte de posibilidades para los equipos de producción, que se han visto obligados a reevaluar la estructura tradicional de "preproducción-producción-posproducción" en pos de nuevas estrategias que involucren de una forma directa a los artistas digitales de VFX (abreviación de *visual effects*).

Estas nuevas tecnologías no solo suponen un cambio radical en la forma de hacer cine, poco a poco se están convirtiendo en un estándar dentro de la industria del entretenimiento audiovisual y pronto se consolidarán como motor de cambio para todos los profesionales que trabajen en este campo. Es, por tanto, necesario reevaluar el papel que los profesionales creativos (artistas, diseñadores, fotógrafos, escenógrafos, etc.) juegan en este entorno y cómo se integran dentro del sector profesional del entretenimiento audiovisual.

0.1. Objetivos

Como objetivo principal, se plantea el estudio y el análisis de técnicas digitales de diseño para VFX indagando en las diferentes vertientes que existen dentro del propio campo. Se propone también la evaluación situacional de los procesos de producción audiovisual y de cómo se han implementado nuevas estrategias y nuevos procedimientos para agilizar el flujo de trabajo entre rodaje, acción real y efectos digitales.

También se indagará sobre todos los medios donde se utiliza estas tecnologías y cómo procedimientos propios de un sector acaban siendo adoptados por profesionales de otros sectores que, en un principio, nada tenían que ver con la aplicación de dichas técnicas.

Existirá, por supuesto, un apartado técnico que presente el funcionamiento y las dinámicas de uso propias de la *performance capture* y de la *virtual production*, y cómo los profesionales involucrados tienen que operar estos sistemas y de qué manera el desarrollo tecnológico ha permitido perfeccionar la implementación de estas tecnologías.

No obstante, resulta esencial concluir este trabajo con un análisis crítico acerca del papel que el artista y/o el diseñador juegan en este campo profesional, así como la oportunidad de introducirse en nuevos entornos de trabajo que la educación y la enseñanza artística frecuentemente deja de lado.

0.2. Motivación

A modo de reflexión personal, existe una inquietud sobre las posibilidades de desarrollo profesional que la industria del entretenimiento ofrece actualmente a profesionales del arte. Resulta interesante investigar cómo el avance de determinadas tecnologías está reduciendo la distancia entre perfiles técnicos y perfiles artísticos, haciendo que cada vez sea más fácil para los artistas participar en proyectos cuyo planteamiento previo requería del dominio de aptitudes distantes del mundo del arte, propias de la ingeniería y/o de la informática.

Considero que el acercamiento de estas tecnologías a procesos artísticos puede ser una buena oportunidad para formar a creativos pluridisciplinares capaces de desempeñar su actividad en campos como el 3D, la animación, los efectos especiales o cualquier otro tipo de disciplina que todavía esté por llegar.

0.3. Metodología y planteamiento

El núcleo del TFM consiste en la investigación de las técnicas y tecnologías citadas, concretamente, *Performance Capture* y *Virtual Production*. Mediante el estudio de diferentes casos y situaciones en la industria audiovisual se pretende justificar la aparición de nuevos procesos que buscan romper con la estructura de producción en tres fases (preproducción, producción y posproducción).

La estructura de este ensayo consta de 3 partes diferentes. En cada una de ellas se abordan aspectos fundamentales que explican las características propias de los VFX.

En la primera parte se analizan las características y los procesos inherentes a la creación de elementos 3D para su posterior integración dentro del *pipeline* en productos audiovisuales. También se abordan conceptos esenciales para la construcción de elementos 3D, así como para el desarrollo de efectos especiales y de técnicas de integración CGI.

En la segunda parte se explica qué es el *performance capture* y cómo este concepto se ha vuelto imprescindible para múltiples proyectos. Se explican diferentes tecnologías dentro de este campo y se analiza el caso de *Hellblade: Senua's Sacrifice*, un proyecto que cambió el planteamiento de videojuego independiente y que, con un presupuesto limitado, definió los procesos de captura de movimiento como una parte fundamental en la narrativa de su historia.

Por último, en la tercera parte se exponen los problemas que las estructuras de producción tradicionales suponen para gran parte de los estudios hollywoodienses de VFX, y cómo, incorporando la visualización en tiempo real, se puede agilizar el proceso de integración de efectos visuales. En este apartado se analizará el concepto de *Virtual Production*, una metodología que utiliza *software* de renderizado en tiempo real para incorporar entornos CGI durante la fase de rodaje.

Todo el ensayo, en especial la tercera parte, se complementa con un aprendizaje práctico mediante la creación de una escena virtual en *Unreal Engine*. Con ello se pretende estudiar de forma práctica cuales son las ventajas y desventajas de incluir un motor de renderizado en tiempo real dentro de la línea de producción de un proyecto audiovisual, así como las limitaciones y las funcionalidades que se han de tener en cuenta cuando se construye una escena dentro de este motor.

1. Primera Parte: fundamentos del CGI

A medida que la computación gráfica ha ido evolucionando a lo largo de los años, los ordenadores se han convertido en una pieza fundamental dentro de la producción audiovisual. La integración de gráficos generados por ordenador (CGI) constituye gran parte del proceso de creación de los efectos visuales que vemos en la gran pantalla, incluso de aquellos que, como espectadores, no percibimos; el CGI ha alcanzado tal nivel de realismo que, muchas veces, nos resulta difícil distinguir, de entre todos los elementos que aparecen en escena, cuáles han sido grabados ante una cámara y cuales otros han sido añadidos en posproducción.

Referirse a gráficos generados por ordenador cómo concepto único sería un error, pues el término aúna multitud de disciplinas con diferentes propósitos y diferentes acabados estéticos. El CGI es un compendio de distintas técnicas y tecnologías que comparten un propósito común: crear elementos virtuales que puedan ser aplicados en el contexto audiovisual de la producción para la que hayan sido diseñados, ya sea cine, videojuego, un evento en vivo o cualquier otra experiencia multimedia.

Si bien el material gráfico producido puede ser bidimensional, es común que el término CGI se utilice para referirse únicamente a las imágenes generadas mediante entornos de desarrollo 3D. Por tanto, de aquí en adelante, cuando se empleen estas siglas, será para referirse a **gráficos 3D generados por ordenador**, independientemente del medio, software o propósito.

Si bien el desarrollo de la tecnología ha permitido dar pasos agigantados en el campo de la computación gráfica, todavía existen aspectos fundamentales que condicionan el flujo de trabajo de los artistas durante el proceso creativo. Valores técnicos como el número de polígonos que componen un modelo o la complejidad de los sombreadores (*shaders*) utilizados han de ser tenidos en cuenta si se quiere proceder de manera eficiente. Con el fin de optimizar el consumo de recursos por parte de la GPU y de agilizar la integración dentro del motor de renderizado, los artistas de CGI condicionan todo su proceso creativo a características propias del medio, que han de ser manejadas de un modo efectivo.

Así pues, en este apartado, se ahondará en las diferentes técnicas y disciplinas que caracterizan la producción de elementos digitales, se analizarán las características de cada uno de los apartados que componen el diseño de entornos 3D, cuál es su implementación dentro del ***pipeline*** (línea de producción), y cómo se articula todo el proceso creativo.

1.1. Modelado 3D

Del mismo modo que, a mediados del siglo pasado, se construían maquetas que recreaban los mundos imaginados por los escenógrafos y los artistas de concepto, la transición digital permitió aplicar los mismos principios a escenarios virtuales que posteriormente se utilizarían para enriquecer los efectos visuales de las películas en producción. Así como un artista tradicional se valía de la arcilla, la plastilina, la silicona o la escayola para construir las criaturas y la escenografía que más tarde cobraban vida en la gran pantalla, un artista 3D hace uso de los mismos fundamentos, pero aplicándolos a un medio digital para crear los elementos virtuales que posteriormente serán integrados en el metraje.

Veamos como ejemplo la primera trilogía de la saga *Star Wars* (Lucas, 1977). En los filmes de George Lucas se construyeron enormes modelos para recrear las escenas que acontecían en el espacio y en los escenarios de otros planetas, las maquetas de las naves espaciales son el ejemplo más evidente [fig. 03]. Tanto el maquillaje de las criaturas alienígenas como el atrezo eran **efectos prácticos** [1] que requerían de una mano artística que los diseñase, los crease y los operase. En las trilogías posteriores, tanto la estrenada en 2001 como la nueva producida por Disney, si bien es cierto que la escenografía sigue jugando un papel fundamental, gran parte de los efectos visuales son creaciones íntegramente generadas por ordenador. Las maquetas de las naves espaciales, los fondos pintados o las piezas de atrezo se han sustituido por modelos 3D.

La importancia que los gráficos por ordenador han ido adquiriendo ha hecho necesario el estudio y perfeccionamiento de disciplinas indispensables para su creación. Una de ellas es el **modelado 3D**, que basa gran parte de sus fundamentos en conocimientos directamente heredados de la escultura tradicional. El uso del término "modelado" no es casual, responde a la necesidad de manipular una materia prima para darle la forma deseada. A través de programas como Zbrush o Maya, los artistas 3D se valen de sólidos euclídeos como punto de partida para comenzar a esculpir y modelar. De esta manera, una esfera o un cubo se pueden convertir en personajes tan complejos cómo los primates que lideran la revolución en las últimas películas de la saga *El planeta de los simios* (Wyatt, 2011. Reeves, 2014-2017).

El modelado 3D es probablemente el proceso más elemental en el diseño de CGI, pues, supone la base de casi todos los objetos que aparecerán en escena, tal como apunta Kevin Hudson (Okun & Zwerman, 2010. pág. 591):

> *The pipeline for digital asset creation begins with digital modeling. It is the birthplace for a digital asset, and it is here that the basic aesthetic form and technology are laid. If they are not done well, this is the place where most problems for the future pipeline begin. Modeling is as important to the creation of an asset as the foundation is to the construction of a house. If the model is solid, all of the other crafts from rigging to painting to color and lighting can be done with far greater ease.*

Puesto que gran parte de los elementos dispuestos han de ser previamente modelados, y todos ellos compartirán las mismas propiedades fundamentales, es pertinente definir cuáles son los parámetros que caracterizan el proceso de modelar.

1.1.1. Tipos de modelos

En una producción audiovisual puede existir una gran variedad de elementos 3D necesarios para complementar la acción real, pero en cuestiones de modelado se podría limitar a una clasificación de cuatro tipos bien definidos (Hudson, en Okun y Zwerman, 2010 pág. 591):

1 . Efectos visuales recreados físicamente ante la cámara.

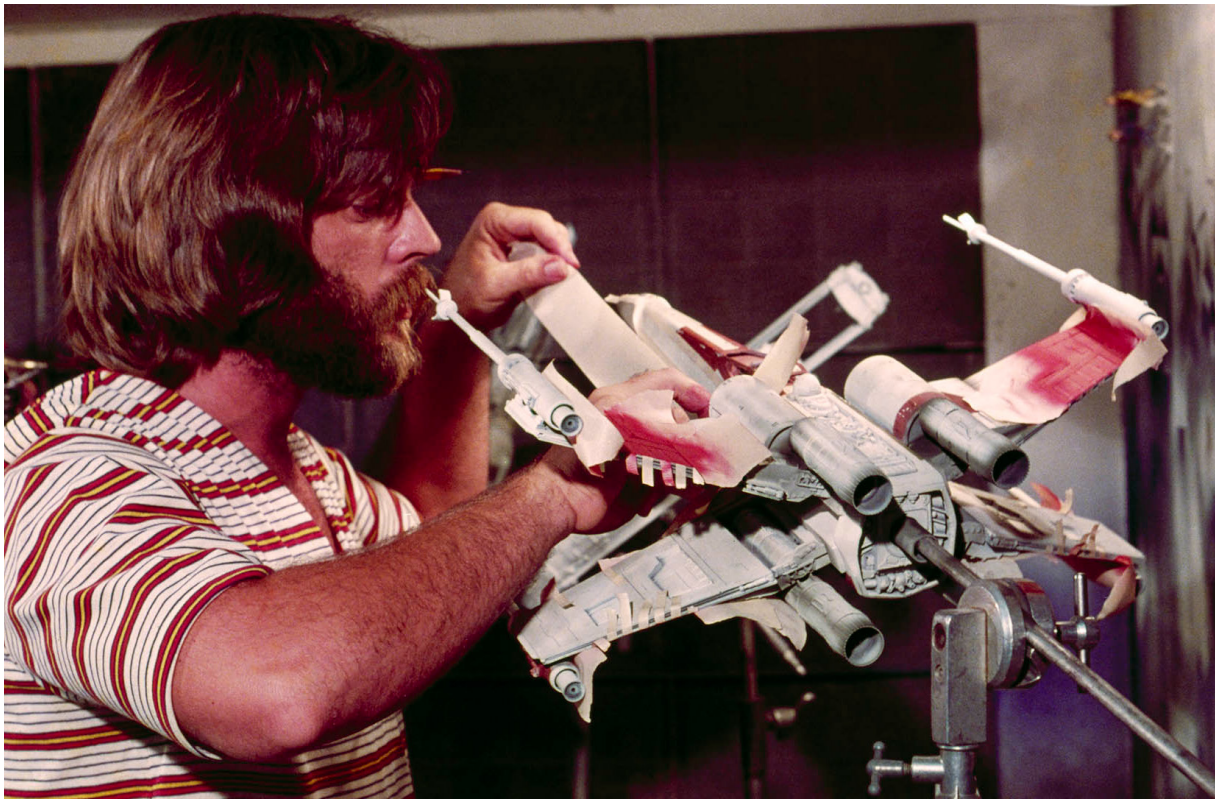


fig. 01 Horda de zombies en *The Walking Dead*.

fig. 02 Primer plano del modelo 3D de Thanos en *Avengers: Infinity War*.

fig. 03 John Dykstra (supervisor de efectos especiales en ILM) ajustando la maqueta de una nave utilizada en *Star Wars: Episode IV*. Fuente: ilm.com

Personajes: hace referencia a todo *asset* [2] que de un modo u otro tenga que realizar una acción o movimiento en la escena (Hudson, pág. 591). El abanico de posibilidades puede ser muy amplio: desde personajes que aparezcan en primer plano y que juegan un papel fundamental en la historia (ej. ***Thanos*** [fig. 02] en ***Vengadores: Infinity War*** (Russo, 2018)) hasta grandes grupos de individuos cuya única finalidad es rellenar el plano y dar sensación de multitud (ej. planos generales de ejércitos u hordas [fig. 01]).

Independientemente de cuál sea el uso final del personaje, este ha de ser modelado por un artista, por lo que el acabado recae por completo en la habilidad para esculpir el modelo 3D. Por supuesto, el nivel de detalle varía dependiendo del uso que se le quiera dar al personaje dentro de la escena. A personajes con gran peso narrativo, y con mucha presencia en plano, se les presupone un nivel de detalle mayor que a otros cuya función será simplemente aparecer en el fondo de una toma.

Props: estos modelos comparten el mismo concepto y significado que su homónimo físico en producciones cinematográficas tradicionales. Suelen ser objetos 3D que no realizan acciones o movimientos por sí mismos, pero son utilizados por los actores para interpretar una acción concreta (Hudson, cap.7 pg. 592).

En ocasiones son piezas sólidas que no necesitan ser animadas (una silla, una botella, un teléfono móvil, etc.), pero en otros casos han de ser ***rigueados*** [3] igual que un personaje para poder realizar una acción completa a través de la animación (una pistola cuando se dispara, un vehículo que se conduce, etc.)

Hard Surface: con el auge del videojuego y de los efectos especiales, este tipo de modelos se ha erigido como una categoría propia dentro del 3D debido a las particularidades en su fase de modelado. ***Hard Surface*** responde a todos los modelos que tienen relación con piezas mecánicas o superficies planas de ángulos marcados. Es frecuente encontrar coches, barcos, aviones, naves, robots, armas o herramientas dentro de esta categoría.

El modelado ***Hard Surface*** presenta sus propias particularidades. El uso de booleanas [4] es típico para definir las formas y los cortes en la figura. A veces es frecuente encontrar geometrías no equivalentes en la malla, con polígonos que no se corresponden ni en número de vértices ni en tamaño; además, el proceso de usar booleanas puede generar ***ngons*** [5] involuntariamente. Por todo ello, es frecuente que los modelos ***Hard Surface*** deban ser retopologizados [6].

Entornos y Escenarios: el título de esta categoría es bastante descriptivo por sí mismo. En ella se engloban todos los modelos destinados a construir un entorno en el que los personajes puedan actuar y en el que se desarrollará la acción. Es cierto que muchas veces la línea entre ***props*** y entorno se diluye, como apunta Hudson (Okun & Zwerman, 2010. pág. 593):

2 . Del inglés: bien, activo.

3 . Término coloquial para referirse a la acción de definir un ***rigging*** de personaje.

4 . Operaciones que siguen la lógica del álgebra de Bool. En modelado 3D se utilizan para ejecutar procesos de suma, resta o intersección entre dos o más modelos.

5 . En una malla 3D, los polígonos con caras de más de 4 vértices.

6 . Del concepto ***retopología de malla***: aplicar una nueva distribución de polígonos conservando la misma forma.

In the case of the train in The Polar Express (2004), it functions as a very complicated prop or hard surface model. But it is also a model that characters walk through and perform within. This is where it enters the realm of environmental modeling.

El modelado de entornos engloba multitud de objetos diferentes: desde vegetación (árboles, plantas, rocas, etc.) hasta elementos arquitectónicos, como casas y edificios. Dentro de esta categoría se encuentran también estructuras más pequeñas tales como elementos de mobiliario urbano. Por este motivo es frecuente que el diseño de *props* y el de entornos se haga conjuntamente.

Diseñar entornos requiere prestar atención a aspectos que no necesariamente son propios del 3D. Un artista de entornos ha de ser consecuente no solo con los detalles y la calidad de los modelos, también con cómo la disposición de los objetos afecta al flujo de la escena y a las sensaciones sobre la audiencia. El planteamiento tras la creación de escenarios está ligado a conceptos fundamentales de la fotografía, de la composición pictórica y del diseño escenográfico. En un medio pasivo, como películas y series, el espectador recibe simplemente la información sin ningún tipo de interacción, pero en un videojuego o una experiencia VR el usuario es una parte activa del entorno y, por tanto, se le ha de tener en consideración en el momento de plantear el diseño.

1.1.2. Características y tipos de datos

Antes de empezar a modelar una pieza existen ciertas cuestiones que se han de tomar en consideración. El diseño de un modelo 3D implica, ante todo, pensar en el medio en el que va a ser utilizado y la función que va a desempeñar. Los recursos gráficos dentro de un motor de *render* son limitados, por tanto, la optimización es una parte fundamental que condiciona todo el proceso de creación, desde la fase de conceptualización hasta el acabado final.

Como se avanzaba previamente, todos los modelos 3D comparten algunas características fundamentales, y estas son comunes independientemente del software que se esté utilizando, tanto para modelar como para renderizar la escena. Existen tres **tipos de datos** fundamentales con los cuales se puede modelar un objeto 3D:

Polígonos: son probablemente los más utilizados en la industria del entretenimiento audiovisual. Un polígono es la unidad básica de geometría definida por tres puntos o vértices, es decir, un plano con cara triangular (Hudson, en Okun y Zwerman, 2010. pág. 593). El modelado poligonal se basa en la consecución de muchos de estos planos para crear una malla que dé lugar a formas más complejas.

Lo más habitual cuando se hace referencia a los polígonos de una malla 3D es referirse a planos de tres vértices (o *tris*), pues muchos motores basan sus cálculos en el número de *tris* que han de renderizar; no obstante, en algunos programas de modelado rara vez se trabaja con *tris*, pues es más fácil operar con polígonos de cuatro vértices (*quads*) para construir cierto tipo de geometrías; solo en el momento de exportar el modelo se divide la geometría y todas las caras de cuatro vértices pasan a convertirse en dos planos de tres vértices [fig. 04].

Es muy importante tener en cuenta el número de *tris* que componen la malla de un objeto, pues el impacto en el rendimiento de la mayoría de los motores de renderizado se ve afectado por el proceso de calcular los polígonos en escena, a mayor número de *tris* en cámara mayor es el esfuerzo computacional que el motor ha de realizar para procesar la imagen. Por este motivo es frecuente que al diseñar un modelo existan dos o más versiones del mismo modelo con topologías diferentes (LOD, del inglés *Leve of Details*). Una versión *High Poly,* que conserva todos los detalles que el artista ha deseado plasmar con geometría real, y una versión *Low Poly,* menos densa, construida mediante *retopología* de malla. Los modelos *High Poly* sirven para *bakear* [7] los mapas de normales y de desplazamiento que luego se aplicarán sobre la versión *Low Poly*, siendo esta la que se utilizará en el motor de renderizado.

Como se introducía al comienzo de este apartado, este tipo de datos es el más utilizado en cualquier producción, no solo videográfica, también en videojuegos, eventos en vivo o experiencias VR (realidad virtual) y AR (realidad aumentada). Este tipo de datos, al ser el más prolífico de los tres, será el que se tendrá en mayor consideración durante el desarrollo de este TFM.

NURBs: de las siglas en inglés *Non-Uniform Rational B-Spline*. En contraposición con las mallas poligonales, son superficies generadas no mediante la unión de polígonos, sino mediante el cálculo matemático de vectores [fig. 05]. En palabras de Hudson (Okun & Zwerman, 2010. pág. 594):

> *A NURBS plane is essentially a mathematically defined surface that uses a B-spline algorithm to generate a patch. A model built-in NURBS is a series of these patches quilted together. An average character model may be made up of hundreds of these NURBS patches.*

La diferencia principal con una malla poligonal es el límite de resolución. Al ser superficies generadas mediante cálculos vectoriales, los modelos NURB carecen de unidad de geometría mínima, por lo que, siempre y cuando no sean convertidas en una malla, su resolución es infinita. Para aclarar un poco el concepto, se podría hacer una comparativa entre el modelado NURB y el diseño vectorial, pues ambos basan sus propiedades gráficas en cálculos vectoriales, mientras que el modelado poligonal podría ser equiparado al diseño por *bitmap*, ya que en ambos existe una unidad mínima de escala, aunque en una malla 3D el tamaño de esta sea variable.

Es frecuente encontrar este tipo de modelado en disciplinas de diseño industrial, pues con él se pueden representar formas complejas en un espacio cartesiano de forma no destructiva, y con un sencillo sistema de manejadores que permite modificar el volumen general en caso de ser necesario. Casi cualquier pieza mecánica suele estar diseñada con superficies NURBs en programas como Rhino o Solidworks.

Voxel: del inglés *volumetric pixel*, es la versión tridimensional de un pixel, la unidad cúbica mínima que compone un sólido 3D. De momento, su aplicación no está del todo implementada en la industria 3D para contenido audiovisual, pero sí es frecuente su uso en aplicaciones médicas que buscan representar gráficamente volúmenes recabados mediante resonancia o tomografía.

7 . Coloquialismo derivado del término inglés *baked maps*. Se utiliza para referirse a operaciones de precálculo. En este caso, define la acción de almacenar características de la geometría del modelos 3D sobre un mapa de textura 2D.

Su principal ventaja frente a los otros dos tipos de datos es que los ***voxels*** ofrecen un volumen efectivo, mientras que las mallas y las superficies NURBs son como un cascarón hueco por dentro. Otro punto a favor es su independencia como elemento geométrico, lo que podría suponer una experiencia más próxima a la de modelar con arcilla. Un bloque sólido de voxels podría modelarse con procedimientos próximos a la escultura tradicional, mientras que el modelado poligonal es como empujar y estirar una malla, donde los métodos aditivos y sustractivos no están definidos.

1.1.3. Proceso de creación

En la industria del entretenimiento audiovisual, las dinámicas de modelado más frecuentes son básicamente las dos siguientes:

Modelado poligonal: comúnmente conocido por las palabras inglesas ***poly modeling***, es el proceso por el cual se modifica una malla poligonal para crear formas mediante operaciones paramétricas. Busca construir un sólido 3D aproximando una malla de polígonos a la forma general del objeto que se quiere representar.

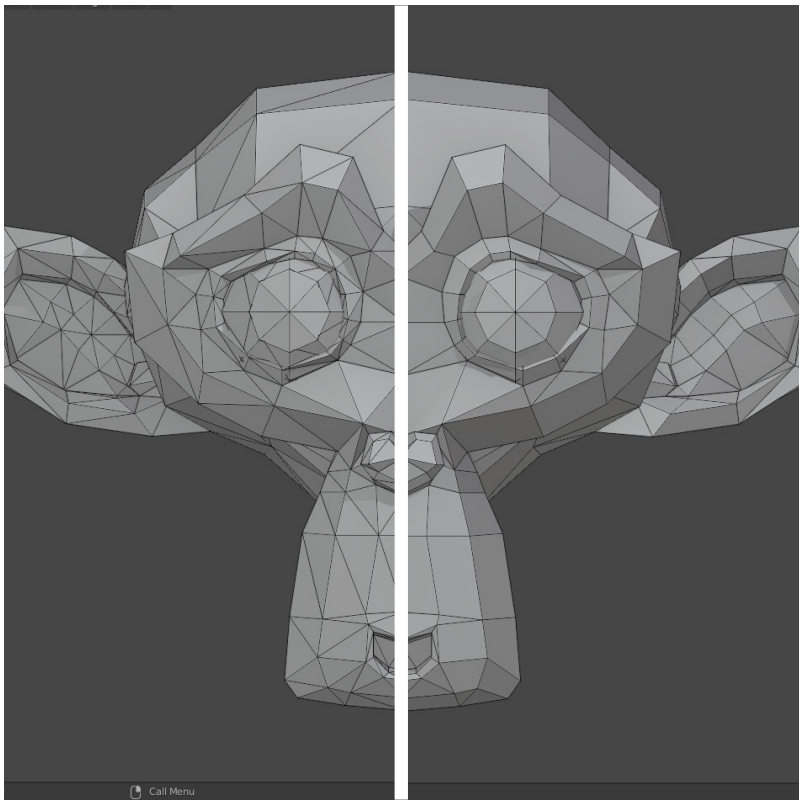
El control sobre los niveles de detalle que se puede alcanzar mediante modelado poligonal es bastante limitado, por lo tanto, esta estrategia de modelado 3D se utiliza sobre todo para recrear modelos con formas no demasiado complejas. El modelado de ***props*** y de elementos de entorno se suele realizar mediante modelado poligonal.

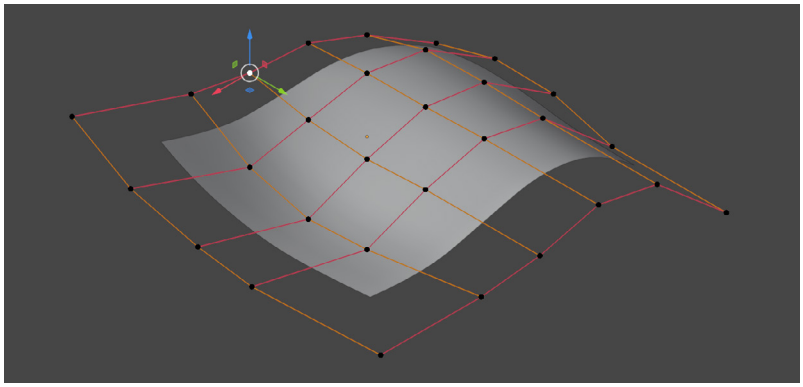
En resumen, el modelado poligonal es útil si se busca un enfoque constructivo, pero no es recomendable para modelar formas orgánicas que requieran un alto nivel de detalle e imperfecciones.

Escultura digital (modelado orgánico): este enfoque supuso un gran cambio en la forma de modelar piezas por su semejanza con el modelado tradicional en arcilla. Se basa en el uso de una tableta gráfica con niveles de presión para emular la experiencia de escultura mediante el uso de un lapiz, o ***stylus,*** y de su respuesta táctil en el ***software*** utilizado (Hudson, en Okun & Zwerman, 2010. pg. 596).

Aunque existen varios programas capaces de cumplir esta función, el software más extendido en la industria es Zbrush. Básicamente, permite manipular la geometría de un modelo con un alto nivel de precisión, mediante operaciones que generan nueva topología a medida que el artista esculpe sobre la malla; además, dispone de herramientas que permiten crear niveles de resolución para manipular la topología en función del nivel de detalle con el que se esté esculpiendo. En este tipo de software, no solo en Zbrush, es prácticamente imprescindible utilizar una tableta digitalizadora si se pretende modelar eficientemente.

La escultura digital es la técnica más utilizada cuando se busca diseñar modelos 3D con formas orgánicas. Prácticamente todos los modelos de personajes y criaturas se diseñan utilizando este procedimiento. También se recurre a estos programas para esculpir imperfecciones y detalles concretos sobre modelos previamente construidos por ***poly modeling***.





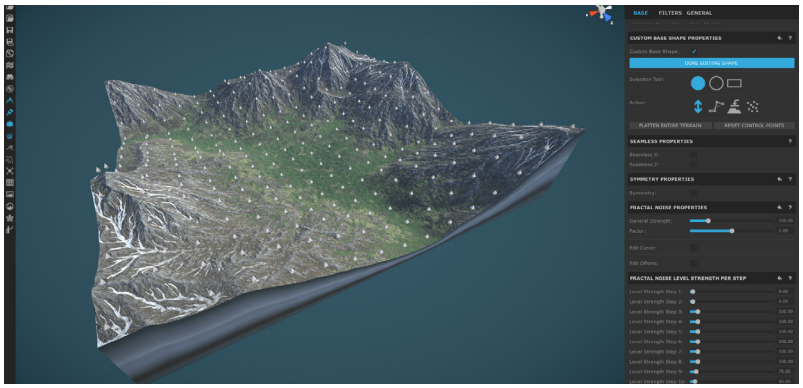


fig. 04 Ejemplo de malla Poligonal en Blender. A la Izquierda, malla en *tris*. A la derecha, mismo modelo en *quads*.

fig. 05 Superficie *NURB* en Blender.

fig. 06 Imagen del terreno diseñado para la escena de este TFM en World Creator. Modelado de Superficie Procedural.

Otros: existen otros procedimientos de creación 3D, pero su desarrollo está focalizado en resolver modelos muy concretos, mientras que la escultura digital y el ***poly modeling*** son metodologías que pueden aplicarse a cualquier tipo de objeto. No obstante, puesto que algunas de estas técnicas forman parte del ***pipeline*** en la creación de CGI para contenido audiovisual, me parece apropiado mencionar brevemente algunas de ellas.

Para la creación de mundos a gran escala (ej. los mapas de un videojuego o las ciudades de un plano general en una película de acción) se utilizan programas que **generan terreno de forma procedural**. Este tipo de software aprovecha un conjunto de generadores [8] para desplazar una superficie y crear formas que imitan formaciones geológicas como montañas, valles, colinas o cauces de ríos. El artista define el tipo de desplazamiento y las modificaciones que el programa debe generar, y el software desplaza una superficie recabando las formas en función de los operadores dados. Algunos de los programas utilizados por la industria son World Creator [fig. 06], World Machine y Gaia.

Otra técnica utilizada desde hace varias décadas es el **foto-escaneado** de objetos reales. Si bien no es una técnica de modelado ***per se***, este tipo de modelo 3D se ha vuelto muy popular ya que permite a los artistas recurrir a enormes librerías con millones de modelos 3D obtenidos a partir de objetos reales, lo cual agiliza el proceso creativo y proporciona ***assets*** con un acabado fotorrealista.

Para escanear un objeto físico se utilizan técnicas de fotogrametría. Mediante una serie de fotos tomadas desde diferentes ángulos se triangula la posición de puntos específicos en el objeto y se recaba una malla 3D.

1.2. Animación

Puesto que existen modelos cuya finalidad es realizar acciones y movimientos en escena, estos han de ser animados con el fin de poder cumplir dicho propósito. Es frecuente relacionar el proceso de animación con modelos de personajes, pero cualquier objeto 3D puede ser animado; no obstante, existe una diferencia notable entre desplazar o rotar un objeto y animar una malla, que se deforma ante los cambios. En 3D, se entiende por animación el proceso que permite añadir movimiento a una malla estática (Preeg, en Okun y Zwerman, 2010. pág. 601).

El concepto de animación responde a todo tipo de movimiento o cambio de estado que varía en un periodo concreto de tiempo. Aunque comúnmente se utilice el término para referirse al movimiento realizado por modelos humanoides o figuras zoomorfas que podemos identificar con seres y formas reales, cualquier modelo 3D es susceptible de ser animado, incluso aquellos que no guardan relación alguna con elementos conocidos. Por ejemplo, modelos mecánicos, vehículos o criaturas ficticias son ejemplos de mallas a las cuales se les puede aplicar animación mediante ***rigging***. Incluso modelos de vegetación que han de ser integrados en el entorno pasan por un proceso de ***rigging*** que simula el movimiento producido por el viento a través de ramas, yerba u hojas.

8 . Operaciones que el programa es capaz de hacer para generar desplazamiento procedural.

Es importante no confundir movimiento con animación. Modelos menos normativos realizan movimientos para cumplir una acción en escena, pero no por ello han sido previamente animados por *rigging*. Es el caso de la figura alienígena en el filme *Annihilation* (Garland, 2018) [fig. 07] cuyo movimiento errático y cambiante se creó mediante un proceso de dinámica de fluidos.

Por otra parte, parámetros como la intensidad de una luz o el cambio de temperatura de color pueden ser secuenciados para simular efectos concretos, como el parpadeo de un tubo fluorescente, aunque esto tampoco ha de ser considerado animación propiamente. El mismo principio se aplica al movimiento de cámara, los objetos desplazados en escena o los cambios atmosféricos del entorno.

1.2.1. Armazón, huesos y manejadores

Para poder someter un modelo 3D a un proceso de animación, se debe vincular este a una estructura con puntos de control que permita controlar el movimiento de la malla mediante fotogramas clave. Como si de una marioneta se tratase, esto manejadores permiten a los animadores modificar la pose de los modelos y establecer diferentes *keyframes* para definir la animación en la línea de tiempo. Es común referirse al armazón de un *rigging* como esqueleto [fig. 08], pues simula la misma función que los huesos de un esqueleto humano.

Los manejadores definidos en el *rigging* de un personaje pueden ser utilizados en procesos de *performance capture* para ser vinculados a los marcadores que capturan el movimiento de un actor.

1.2.2. Animación por Rigging

El Rigging es un concepto altamente extendido en el mundo de la animación, no solo 3D, pues comparte procesos que también se aplican en animación 2D. *Riggear* (termino coloquial adaptado del inglés para referirse a la acción de definir el *rigging* de un modelo) significa establecer una serie de manejadores digitales que permiten modificar el comportamiento de una malla en el tiempo.

Si bien es común relacionar el *rigging* con esqueletos digitales y huesos, este proceso puede ser realizado de muchas maneras diferentes, incluso mediante programación y *scripting,* u otros métodos externos a la aplicación que se haya utilizado para definirlo. Segun Steve Preeg (Okun & Zwerman, 2010. pág.601):

> *Rigging can be simply expressed as a process that allows motion to be added to static meshes or objects. [...] These rigs can be made up of many aspects such as bones, deformers, expressions, and processes external to the application in which the rig is built. [...] In general, rigging can be broken into two distinct types: animation rigging and deformation rigging. Each of those types may be further broken down into subcategories such as body rigging, facial rigging, and noncharacter rigging.*

En esencia, todo el proceso de animación 3D se basa en la asignación de una serie de manejadores. Estos esqueletos se diseñan en función de las necesidades del animador, y han de ser lo más esclarecedores posible. En palabras de Preeg (Okun & Zwerman, 2010. pág. 603):

Control structures for character rigging are as varied as the characters themselves. One may need bones to be able to squash and stretch cartoony characters, while another may need to be anatomically correct for realistic body motion.

Existen ciertos convencionalismos que se han establecido en el sector de la animación. El uso de colores y formas, la posición de ciertos manejadores o la topología específica requerida para ciertos movimientos, son reglas no escritas que se han mantenido invariables entre programas y estudios para poder facilitar el proceso de animación. Según sugiere Preeg (Okun y Zwerman, 2010. pág. 604):

Animators should feel that the rig is allowing them to use their talents to animate; anything that slows down that creative process should be avoided.

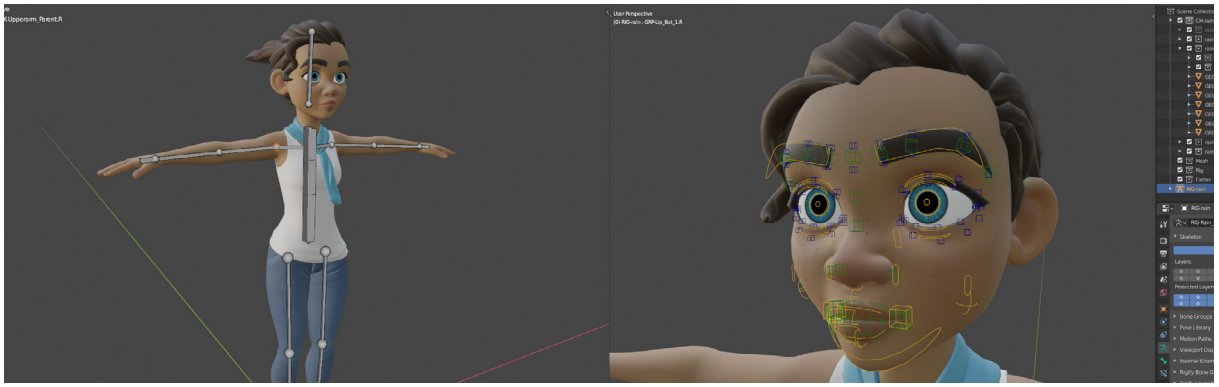
La animación por *rigging* está supeditada a características propias que permiten al animador ejecutar la acción del modelo. Un primer paso para poder animar de forma efectiva las poses es **pesar los vértices** de la malla. Esto significa añadir un grado de influencia a las partes de la malla que se van a deformar con la rotación y el reposicionamiento del esqueleto y de los manejadores asignados. Sin un correcto peso en los vértices de los polígonos deseados, la animación puede resultar en una deformación excesiva, o en la torsión de zonas no deseadas.

Otro concepto fundamental en animación es lo que en inglés se denomina ***Inverse Kinematic & Forward Kinematic*.** Una de las propiedades fundamentales de un *rigging* de personaje es la posibilidad de animar tanto por cinemática inversa (IK, sus siglas en inglés) como por cinemática directa (FK). En una cadena de articulaciones, el movimiento puede ser definido tanto por la articulación de origen como por la pieza final en la cadena de movimiento. Si, por ejemplo, se busca animar una pierna, se puede mover la posición del pie al punto de destino, y el movimiento del resto de partes de la pierna será interpretado por el *rigging* mediante cinemática inversa. El proceso contrario consistiría en desplazar la articulación del fémur con la cadena y, posteriormente, reposicionar el resto de las articulaciones para simular el movimiento, lo que constituiría una animación por cinemática directa. La posibilidad de decidir entre IK y FK en un *rigging* es una característica fundamental (Preeg, en Okun y Zwerman, 2010. pág. 602).

1.2.3. Deformación por Rigging

Este proceso toma los datos de animación del *rigging* para deformar la malla del personaje según los movimientos establecidos. La deformación por *rigging* proporciona cierta credibilidad a los movimientos del personaje, pues complementa la sensación de realidad al atribuirle a la malla 3D propiedades parecidas a las de la piel humana. Con un *rigging* por deformación se puede simular la contracción de músculos en función de movimientos concretos.

La deformación por *rigging* es un proceso esencial para animar expresiones faciales y para operar de manera efectiva los mecanismos de *performance capture*. Los manejadores del *rigging* en la cara de un personaje permiten mover y deformar la malla para reproducir expresiones propias de la morfología humana [fig. 08].



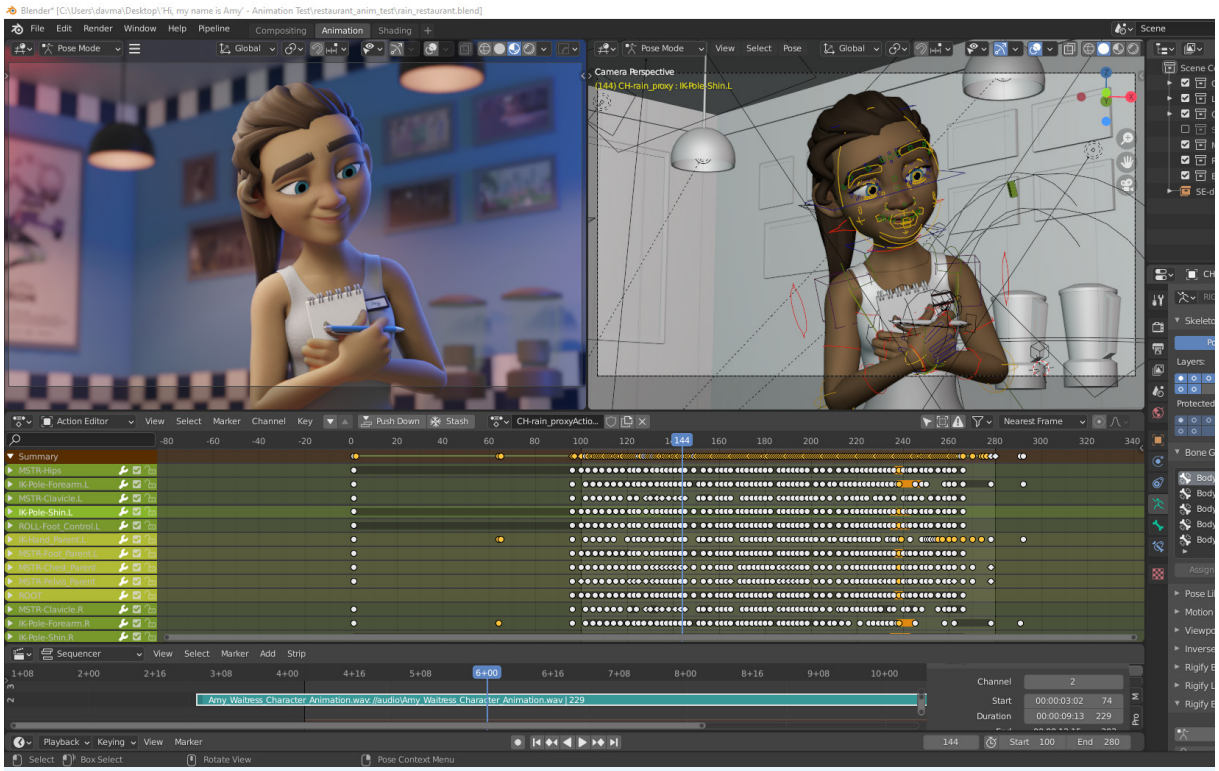


fig. 07 Fotograma del filme *Annihilation (Garland, 2018).*

fig. 08 Derecha, Esqueleto para personaje 3D. Izquierda, Facial Rigging. Fuente: Blender.org

fig. 09 Rigging completo e interfaz de animación en Blender. Fuente: Blender.org

1.3. Texturas y sombreadores

Tal como sucede con el modelado 3D, las **texturas** y los **sombreadores** (*shaders* en inglés) son una parte fundamental en la recreación de modelos 3D. Si el modelado permite construir la forma del modelo, el *shading* define el aspecto visual de las superficies que caracterizan cada objeto 3D. Según Spears (Okun & Zwerman, 2010. pág. 672):

> *Shading is the part of the rendering pipeline that gives each sample on a surface a value. In the design of a rendering system, the shading system, also called the shading machinery, is called into play once the renderer discovers visible surfaces at each pixel on the rendered image.*

El sombreado de modelos permite dotar a los objetos 3D de propiedades físicas que definen el modo en el que estos interactúan no solo con la luz en la escena, también con la que rebota de otros modelos adyacentes. En prácticamente todos los programas 3D existe un motor de materiales [fig. 10] [fig. 11] que permite definir los sombreadores de un modelo, pues **Material** y ***Shader*** son sinónimos de un mismo concepto, tal y como especifica Spears (Okun & Zwerman, 2010. pág. 672):

> *When applied to a CG object, a shader and all of its associated settings that define the look of an object are often referred to as a material.*

La principal característica de un material es establecer en qué modo los rayos de luz deben interactuar con la superficie de los modelos. A través del desarrollo y la evolución en 3D, se ha experimentado con diferentes marcos de iluminación que simulan comportamientos próximos a la realidad.

Pero cuando se trata de crear materiales, existe un concepto fundamental si se pretende un acabado fotorrealista. Los llamados **materiales PBR** [fig. 13] (***Physically Based Render***) son un marco de trabajo particular que busca dotar de todas las propiedades básicas que una superficie posee en el mundo físico. Mediante materiales PBR se puede definir el nivel de reflectividad, si el objeto es metálico o dieléctrico, el índice de especularidad, su nivel de transparencia o su índice de refracción.

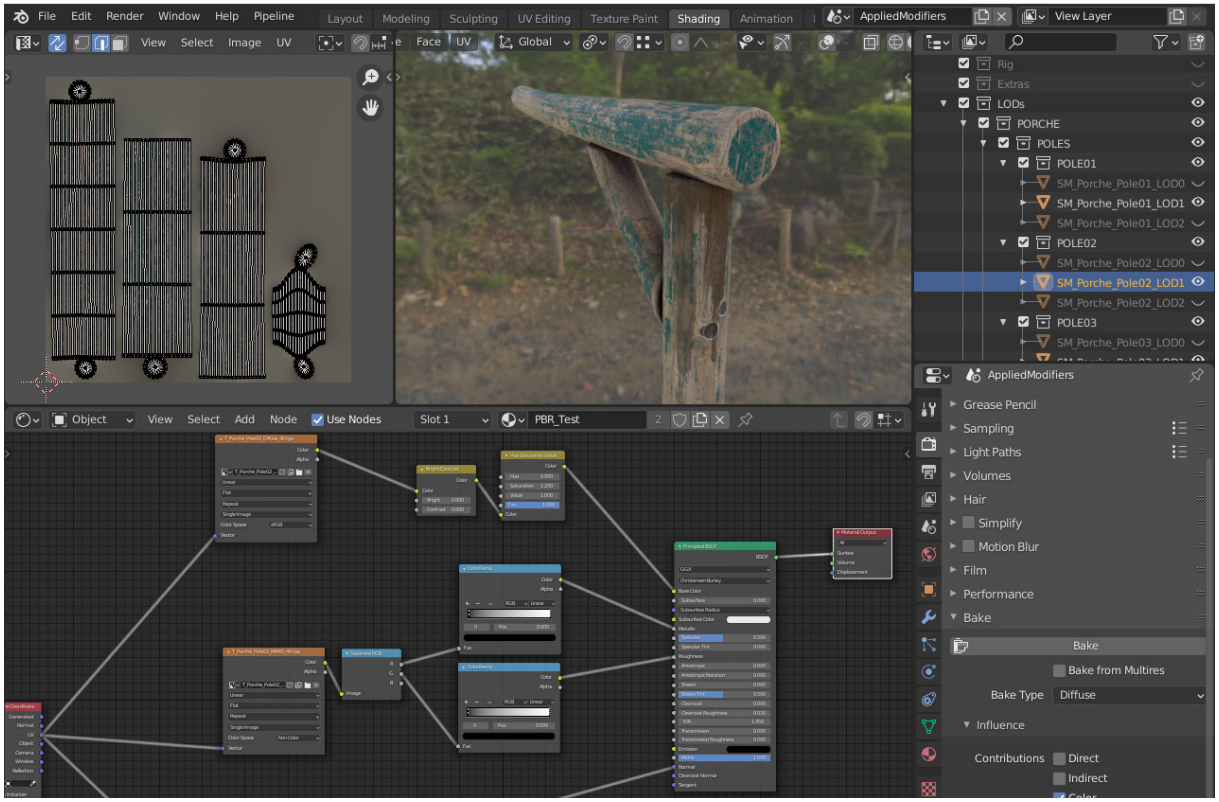
1.3.1. Tipos de mapa

El mapeado de texturas [fig. 12] es la técnica que permite proyectar sobre una superficie 3D las texturas 2D (***bitmap***) que servirán para construir la apariencia de un modelo.

En un material se definen diferentes tipos de mapa para ajustar cada una de las propiedades físicas del objeto cuando un rayo de luz alcanza su superficie.

Mapas representados en RGB:

Albedo (*Diffuse Maps*): este mapa confiere el aspecto visual, la imagen o el color de la superficie. Se compone de una imagen RGB en 8 bits y solo contiene la información de color del objeto. Los mapas de albedo no deben tener sombras pronunciadas ni brillos excesivos sobre la imagen.



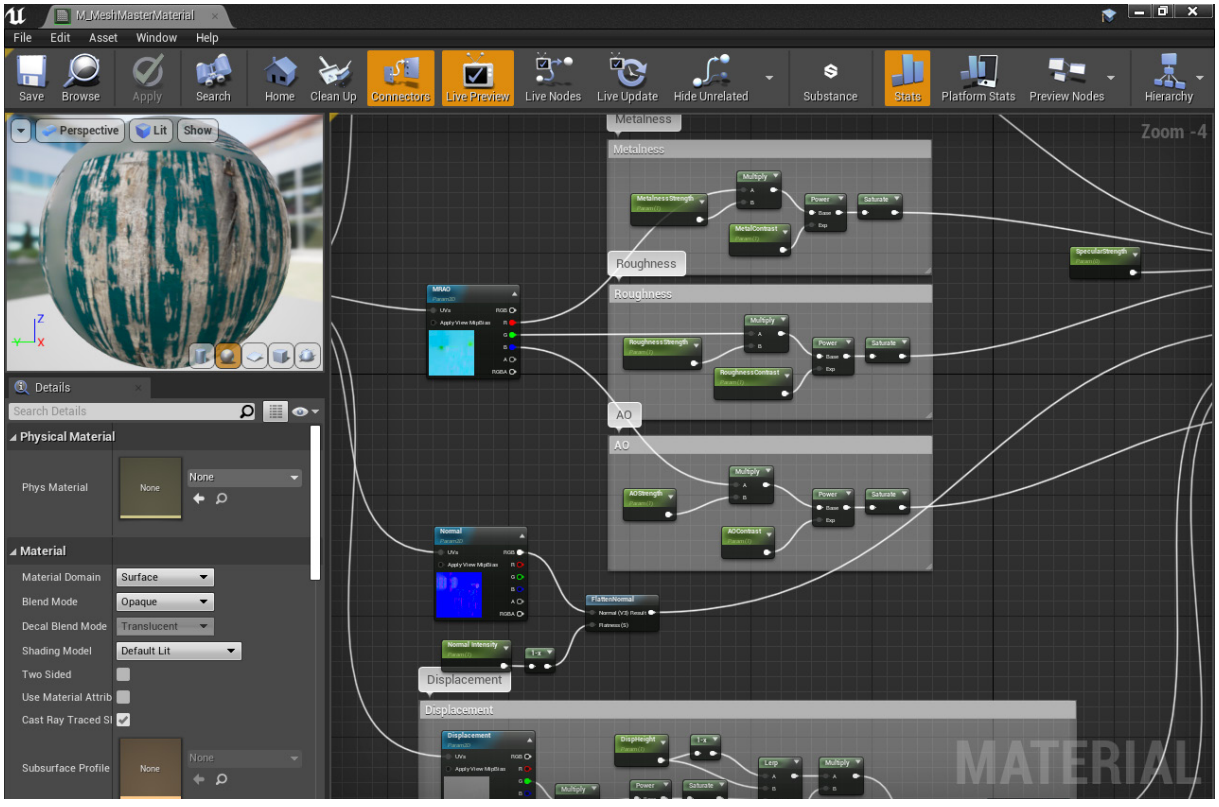


fig. 10 Motor de materiales en Blender.

fig. 11 Motor de Materiales en Unreal Engine.

Normales (*normal/bump*): Un mapa de normales reproduce la información del espacio tangencial, y le proporciona al motor la dirección en la que la luz ha de rebotar sobre cada polígono. Es una representación bidimensional del volumen 3D sobre cada uno de los puntos en la superficie del modelo. La información se almacena en una textura RGB donde las coordenadas X,Y, Z corresponden a los componentes RGB respectivamente.

El espacio tangencial define el espacio de coordenadas vectoriales tangente a la superficie del modelo. En un software 3D, se define mediante el cálculo matricial de la tangente (el vector que toca la superficie en un único punto), la normal (el vector perpendicular al punto) y la binormal (el vector perpendicular a la normal en el mismo plano que la tangente) de los vértices que componen cada polígono.

Los mapas de normales son apropiados para simular la sensación de volumen sobre una superficie plana sin generar geometría adicional, y permiten crear la ilusión de detalles sobre un modelo *low poly*.

Mapas representados en escala de grises:
Rugosidad (*Roughness Maps*): se utilizan para controlar el índice de irregularidades que presenta una superficie. Cuanto más rugosa sea una superficie, los rayos de luz incidente rebotarán con mayor aleatoriedad haciendo que el material se vea menos brillante.

Especularidad (*Specular Maps*): se utilizan para controlar la reflectividad especular de un modelo (Woodall, en Okun & Zwerman, 2010. pág. 608). Los valores de especularidad se representan en escala de grises, donde "negro" es nada reflectivo y "blanco" es totalmente reflectivo.

Un material especular no tiene porqué ser metálico, aunque su superficie refleje rayos de iluminación. Un ejemplo de ello es la superficie de un estanque de agua o de un espejo.

Oclusión ambiental (AO, siglas en inglés): para simular una iluminación global convincente se recurre a una técnica de sombreado conocida como oclusión ambiental. Mediante esta técnica se estima qué partes de la escena están más expuestas a la iluminación global. Esta estimación es calculada de forma global, por lo que la disposición de los objetos afectará al resultado final.

La función principal de la oclusión ambiental es la de simular sombras en las zonas más ocluidas de un modelo. Cuando dos superficies son adyacentes, o están próximas la una a la otra, se simula una sombra en las aristas de ambos objetos, así como en las superficies cóncavas donde la luz no puede llegar fácilmente.

Los mapas de oclusión ambiental plasman esos valores de exposición en una imagen *bitmap*, para poder aplicarlos a los modelos en condiciones de iluminación diferentes, conservando los detalles de oclusión ambiental.

Metálico (*Metalness*): definen los índices de reflectividad de un material. Mediante estos mapas se puede especificar cómo de metálico es un material y en que partes de su superficie lo es más o menos. Los mapas de metalicidad le indican al motor de render cómo de dieléctrica es una superficie, haciendo que los rayos reboten sin difracción o, por el contrario, que sean absorbidos y rebotados en varios ángulos.

Desplazamiento: estos mapas expresan en valores de gris el nivel de desplazamiento que el motor de sombreado ha de ejecutar sobre cada punto de la superficie del modelo. A diferencia de los mapas de normales, los mapas de desplazamiento se utilizan para generar geometría real sobre la malla del modelo. Su uso más frecuente es para transportar detalle real del modelo ***High Poly*** al modelo ***Low Poly***.

Los mapas de desplazamiento se expresan en escala de grises, pero para evitar efectos de escalonado en el desplazamiento de la malla se utilizan compresiones de 32 bit que aportan más información en los degradados.

1.3.2. Coordenadas UV

Antes de proceder con las tareas de texturizado es necesario definir las coordenadas UV [fig. 14] del modelo. Estas son la representación en dos dimensiones de una superficie 3D, siendo U el eje de coordenadas horizontales y V el de coordenadas verticales, en un plano cartesiano. Es el método mediante el cual se asigna un espacio efectivo a cada parte de la malla, con el fin de poder proyectar en 3D las texturas sobre la superficie del modelo.

En un principio, el proceso de texturizado se realizaba mediante proyecciones en coordenadas globales del mundo, pero posteriormente, la creación de UVs ha permitido definir un espacio de coordenadas específico para cada objeto, consiguiendo que cada modelo tenga su sistema de texturizado propio, independientemente del espacio de coordenadas globales que ocupe en la escena 3D.

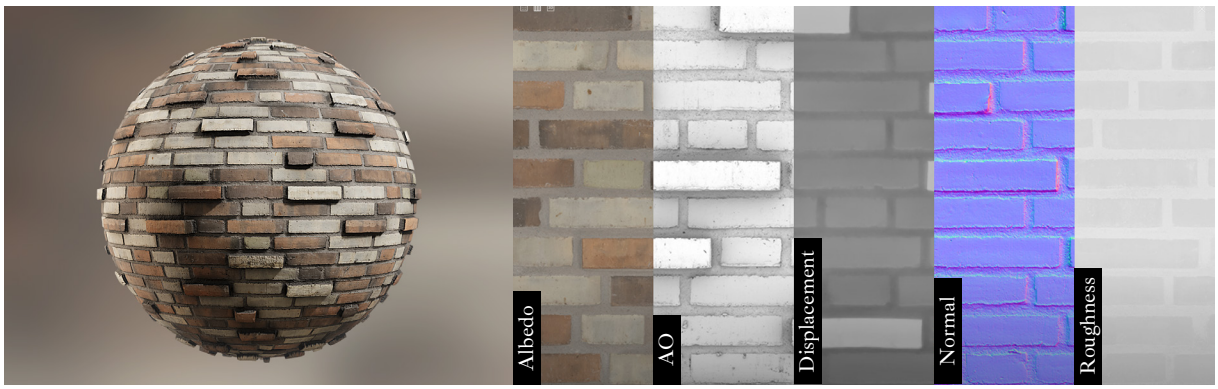
El **desempaquetado de UVs** (***UV unwrapping***) es un proceso fundamental por el que todo modelo ha de pasar si se pretender texturizar su superficie.

1.3.3. Proceso de texturizado

El **texturizado** es una parte fundamental de la creación de materiales, pero opera en un nivel diferente al del sombreado. Las texturas son imágenes en mapa de ***bits*** que se aplican sobre la malla del modelo para construir su apariencia visual. Estas pueden ser creadas de formas diferentes, y se componen de varios mapas que representan las diferentes propiedades de la superficie que se quiere sombrear.

Texturizado por imagen: para texturizar un modelo se puede recurrir a mapas de ***stock*** preparados para construir los materiales en un software 3D. Las librerías de materiales foto-escaneados permiten descargar el conjunto completo de mapas necesarios para reproducir una superficie PBR.

Pintura y procedural: este método es algo más tedioso, pero extremadamente versátil en ciertas ocasiones. Los mapas de texturas se pueden crear con programas específicos que permiten pintar directamente sobre un modelo 3D como si de una miniatura se tratase. Si, por ejemplo, se busca una estética estilizada (o ***cartoon)*** no tendría mucho sentido utilizar texturas obtenidas de imágenes tomadas de superficies reales; en cambio, se podría optar por pintar estas superficies como si fuesen ilustraciones.



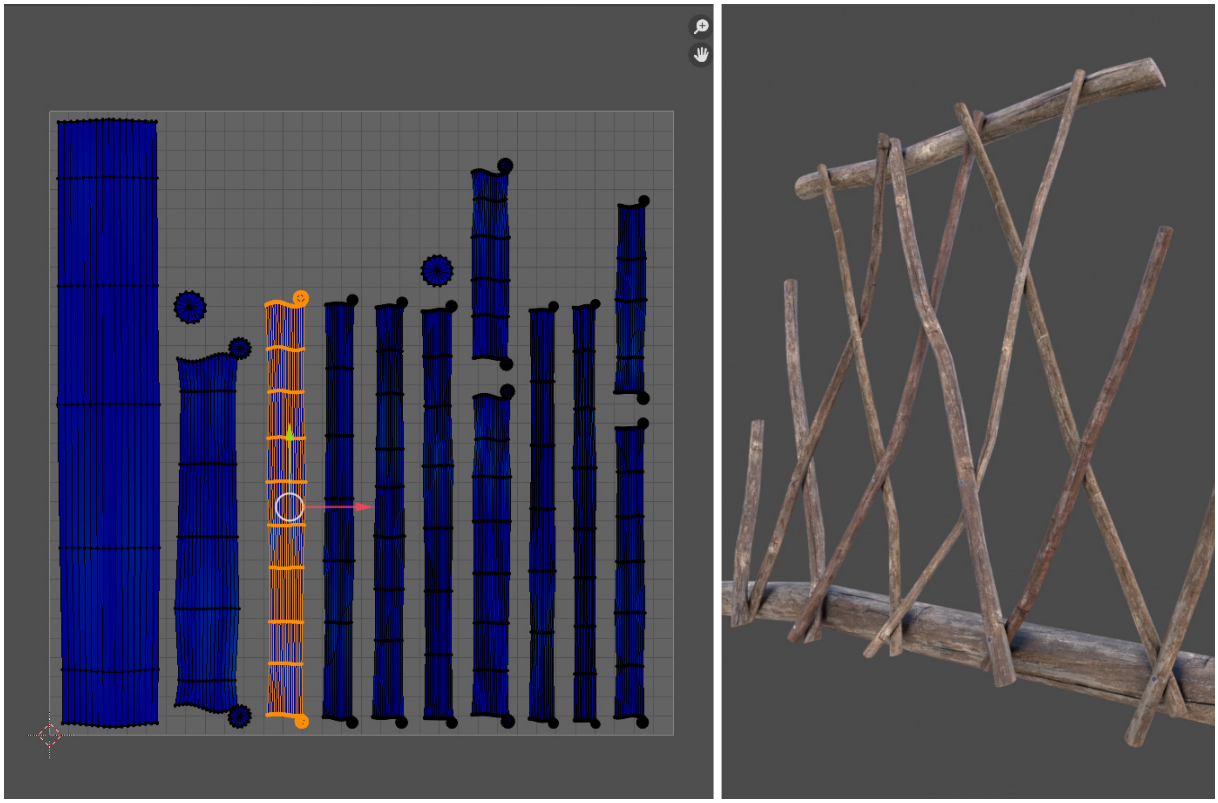


fig. 12 Imagen de material PBR. Mapas de izquierda a derecha: Albedo, AO, Desplazamiento, Normales. Fuente: Quixel.com

fig. 13 Fotografía de escena 3D renderizada con diferentes materiales PBR. Fuente: Artstation de Gavriil Klimov (director de arte senior en NVIDIA).

fig. 14 A la izquierda: UVs de uno de los modelos (derecha) diseñados en blender para este proyecto.

Los programas de texturizado, como *Substance Painter*, permiten no solo pintar el albedo, también el resto de los mapas necesarios para un flujo de trabajo PBR.

Por otra parte, el **texturizado procedural** trata de crear mapas de texturas mediante operadores procedurales.

1.4. Simulación y físicas

La simulación en entornos 3D es un área que engloba multitud de disciplinas, todas ellas demasiado extensas para poder describirlas con detalle en este documento. Su estudio detallado abarcaría toda una tesis, por ello, en este TFM, únicamente se mencionará en qué consisten estas técnicas y de qué manera influyen en la integración de una producción virtual.

Las técnicas de simulación resultan imprescindibles en la creación de efectos digitales para cine, pues permiten resolver situaciones que antes solo eran posibles mediante efectos prácticos. En esencia, las simulaciones permiten recrear fenómenos físicos del mundo real en entornos digitales, logrando comportamientos e interacciones realistas, pues permiten que el software calcule los resultados adecuados sin recurrir a una animación por fotogramas clave (Crow, en Okun & Zwerman, 2010. pág. 637).

Existen varias categorías de simulación, las más comunes son:

Simulación de fluidos: se basa en que todo elemento que no sea sólido, y cuyo proceso de modelado resulte demasiado complejo, es susceptible de ser generado mediante simulación de fluidos. Los elementos más típicos son el agua (o cualquier otro tipo de fluido), el fuego, humo, gases o efectos volumétricos en general.

Simulación de fuerzas: en un programa 3D, existen actores específicos que recrean fuerzas físicas fundamentales dentro del motor. Estas no están destinadas a aparecer en el render final, pero sirven para controlar el comportamiento de otras simulaciones. El uso de fuerzas se combina con otras simulaciones para emular, por ejemplo, las hojas de un árbol movidas por el viento o las olas de un mar embravecido.

Dinámicas de cuerpo rígido y cuerpo blando (*rigid and soft body dynamics*): se encargan de dotar a las mallas en escena de propiedades físicas, como el peso, el índice de fricción, la viscosidad o la resistencia. Son esenciales cuando se pretende recrear situaciones en las que parte del escenario es destruido (un edificio derrumbándose o un proyectil atravesando una pared).

Sistemas de partículas: se utilizan para simular el comportamiento de grupos de diminutas partículas. Habitualmente su función es complementar los fenómenos de otros sistemas. Por ejemplo, si se quiere recrear las cenizas suspendidas en el aire durante un incendio, se combinarán un sistema de partículas y una simulación de fluidos que recreen el humo y el fuego en la escena.

Las simulaciones son perfectas para recrear fenómenos naturales realistas. Estos fenómenos, que antes se recreaban mediante procedimientos prácticos, se han sustituido por simulaciones 3D. Trabajar este tipo de escenas en entornos digitales

permite un mayor control sobre cómo se desarrollan ante la cámara, y supone una reducción significativa del precio que costaría realizarlas con maquetas y escenarios.

Las técnicas de simulación requieren un gran consumo de recursos computacionales, y es probable que el proceso de ***bakear*** [9] una simulación sea incluso más lento que el de un render final, por tanto, hay que considerar cuándo es conveniente recurrir a estas técnicas y cuando evitarlas si no son del todo necesarias, con el fin de optimizar los tiempos de producción.

1.5. Iluminación

Una iluminación coherente es una elemento fundamental para la correcta representación de una escena. Una buena iluminación no solo es crítica para representar los modelos 3D, también es el elemento que condicionará gran parte del aspecto plástico y de las sensaciones que transmita la imagen.

Del mismo modo que en el plató existen focos y paneles, en una escena 3D existen actores específicos que cumplen la función de iluminar los objetos que se han dispuesto en plano.

El uso y posicionamiento de luces en una escena 3D comparte muchas de las características de la luz física. Propiedades como la intensidad, la temperatura de color o el ángulo de incidencia participan exactamente del mismo modo que en un foco o una lámpara real. De hecho, es frecuente que los programas 3D utilicen la misma terminología para ajustar y definir estos valores. Las propiedades físicas de las luces en un motor de render buscan simular de la manera más exacta posible el comportamiento natural de las luces reales.

Así pues, cuando se tiene que diseñar la iluminación de una escena 3D, se recurre a diferentes tipos de luces que empatan a sus homónimos en la realidad. Dentro de los diferentes programas se puede elegir entre **luces direccionales**, **focos**, **paneles** cuadrados o rectangulares, ***spots***, etc. También existen actores algo más específicos dependiendo del tipo de motor que se esté utilizando; en el caso de ***Unreal Engine***, al no ser originalmente un motor de ***ray tracing***, se pueden disponer actores concretos para suplir la luz ambiental rebotada, la oclusión ambiental y la difracción atmosférica.

A continuación, se enumeran los **tipos de luces** [fig. 15] comunmente utilizadas en programas 3D.

Luz direccional: este tipo de actor imita la iluminación natural producida por el sol. La luz direccional es típica cuando se recrean escenarios al aire libre o panorámicas de entornos naturales, ya sean reales o no. Aunque se puede ajustar su ángulo de incidencia, su intensidad y su temperatura de color, estos actúan con equidad sobre todos los objetos presentes en la escena.

9 . En este caso, el significado es el de precalcular la simulación y formatearla como animación.

Focos/spots: igual que los focos de estudio, este tipo de actor arroja una luz siguiendo un único vector de dirección, pero dibujando un patrón con forma de cono. A través de diferentes ajustes se pueden controlar parámetros como el ángulo de apertura del cono o el decaimiento. Al contrario que la luz direccional, los focos siguen la norma de la inversa del cuadrado, por lo que su intensidad sobre los objetos varía en función de la distancia que guarde con estos.

Paneles: actúan exactamente igual que los focos, pero la fuente de luz es un panel rectangular o cuadrado. Al abarcar más superficie, suelen utilizarse para arrojar luz difusa y cubrir grandes áreas de iluminación. Lo más parecido en un plató real sería un panel LED o un foco con un difusor.

Puntuales: son luces que irradian luz en todas direcciones desde su posición. Comúnmente se utilizan para simular bombillas o cualquier otra fuente de luz que no esté direccionada (hogueras, antorchas, etc.).

Emisores: no son actores de luz por sí mismos, pero contribuyen en la iluminación global de una escena. Un modelo 3D puede ser definido como emisor durante el proceso de creación de materiales, lo que hará que este emita fotones durante el proceso de renderizado.

1.6. Renderizado

En computación gráfica, se denomina render a la imagen final, fotorrealista o no, obtenida mediante la interpretación conforme a los cálculos realizados por un programa. Aunque muchas veces se utiliza para referirse a imágenes o secuencias producidas exclusivamente en entornos 3D, el término es común a otras disciplinas como la videografía, la animación digital, la ilustración o el retoque fotográfico.

Tomando como ejemplo un software más familiar, si construimos un montaje fotográfico en Photoshop, el resultado final que exportemos de dicha composición será nuestro *render*, mientras que el archivo *psd* donde hemos construido el montaje será nuestro archivo maestro.

Del mismo modo, un render 3D es la imagen, o serie de imágenes finales generadas a través de un motor de renderizado. Es irrelevante la herramienta de software 3D que se utilice para construir una escena, pues, en aras de poder ser utilizada en el montaje final de un vídeo, esta ha de ser renderizada con un motor que, en diversas ocasiones, es externo al programa de modelado.

Si bien el proceso de renderizado es algo que sucede de manera externa a la creación de los modelos y al diseño de la escena, muchos programas de modelado traen incorporados sus propios motores de renderizado para hacer que el flujo de trabajo sea más eficiente y sencillo para los artistas 3D.

1.6.1. ¿En qué consiste?

En resumen, la función de un motor de render es calcular todo lo dispuesto en la escena 3D y plasmarlo en una imagen 2D, o en una secuencia de estas. Dependiendo del tipo de motor utilizado, así como de los parámetros establecidos, ciertos factores influirán en la apariencia final del render.

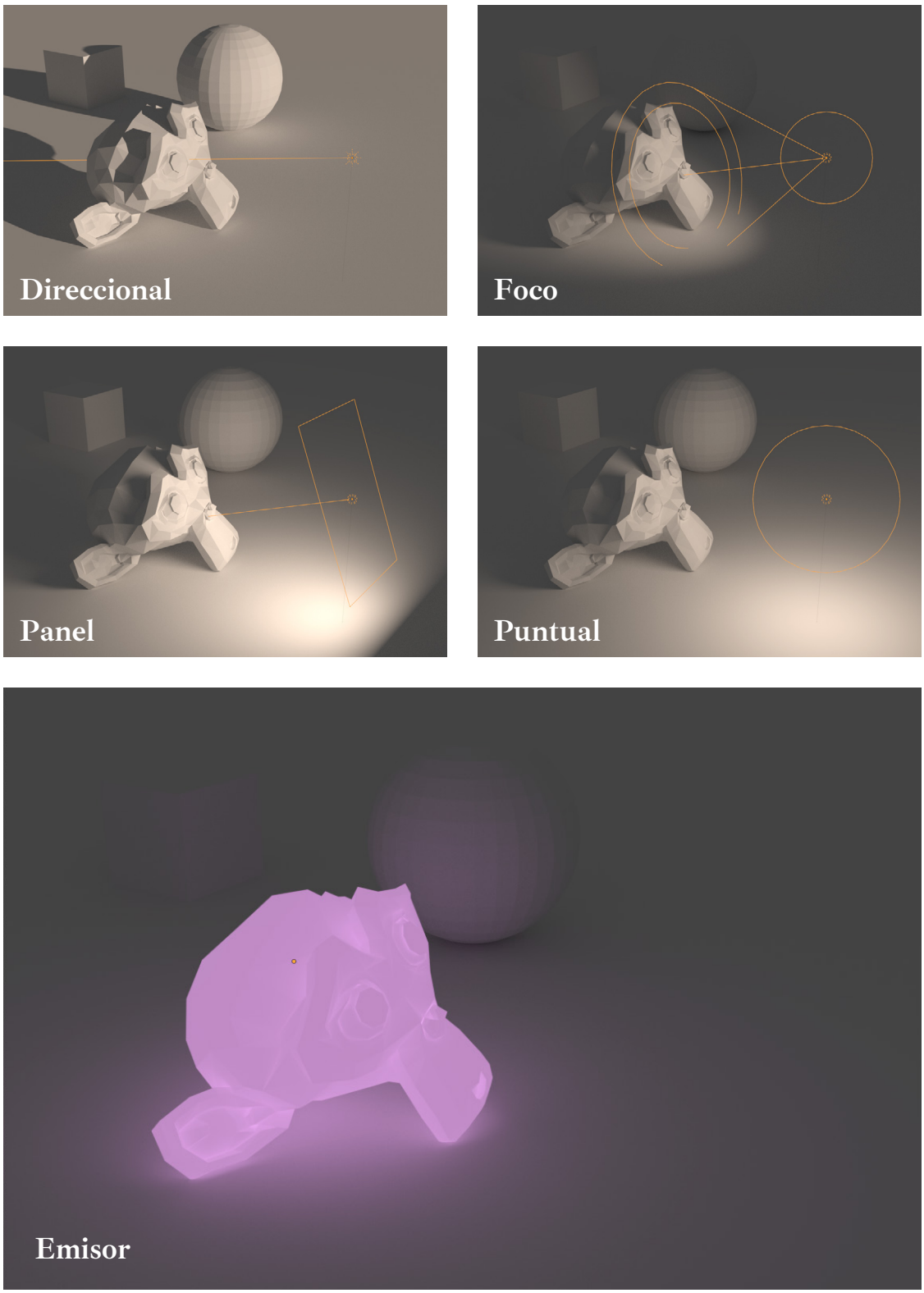


fig. 15 En orden de aparición de Izquierda a derecha y de arriba a abajo: Ejemplos de luz direccional, foco, panel rectangular, luz puntual y material emisor en Blender.

En la industria del diseño 3D existen multitud de motores, cada uno de los cuales se centra en potenciar características concretas a la hora de renderizar. Dependiendo de cuál sea la intención plástica del artista 3D, este elegirá el motor que mejor se adapte a sus necesidades. Incluso dentro de una misma categoría, cada motor resolverá aspectos específicos de manera diferente.

Se puede deducir que no hay un motor superior al resto, cada uno es competente en ciertas áreas, está en manos del artista decidir cuál es el que mejor se ajusta a las necesidades de su proyecto, o en su defecto, cómo suplir las carencias del motor que acostumbra a utilizar.

1.6.2. Diferencias y tipos de render

El desarrollo de los algoritmos de renderizado ha ido de la mano de los avances en computación gráfica, así como del aumento exponencial en la potencia gráfica de las GPU. Desde su implementación en 1970, se han buscado nuevos algoritmos que alcancen acabados de render cada vez más próximos al foto-realismo, sin sacrificar eficiencia en el proceso. Cada una de estas técnicas ha intentado mejorar aspectos en los que las otras fallaban, aproximándonos año tras año a motores de render cada vez más logrados y funcionales.

Como primera distinción, los motores de render se dividen en motores pre-render (***motores offline***) y motores en tiempo real (***motores online***). Los primeros son típicos de la industria cinematográfica y VFX (ej. V-Ray, Octane, Corona Render, Arnold, etc.), mientras que los segundos son utilizados habitualmente en videojuegos, aplicaciones VR y AR o en eventos en directo (***Unreal Engine, Unity, Cry Engine,*** etc.).

En la historia del 3D se han explorado multitud de enfoques, algunos de ellos descartados por la aparición de técnicas mejores, y otros que se han mantenido hasta nuestros días. Existen multitud de algoritmos que se encargan de solventar tareas concretas de renderizado, y los motores de render construyen su proceso ejecutando varios de estos algoritmos a la vez, pero, por convención y de forma más genérica, se podría reducir el catálogo a cuatro métodos esenciales de renderizado: **Rasterizado*, Ray casting***, ***Radiosity*** y ***Ray tracing***.

Rasterizado: al igual que sucede en un programa de diseño vectorial, los procesos de rasterizado toman las formas primitivas (curvas y vectores) y las convierten en formato de mapa de ***bits***.

En el caso del renderizado 3D, en vez de construir la imagen píxel por píxel, los algoritmos de rasterizado proyectan la geometría de los objetos 3D sobre un plano 2D, en este caso el monitor, y construyen dichas formas en base a los ***tris*** que componen la geometría de los objetos proyectados [fig. 16]. Trasformando los vértices 3D de cada triángulo en puntos bidimensionales sobre el plano, el motor es capaz de rellenar los espacios comprendidos entre cada punto mediante un proceso conocido como ***scanline rendering*** o escaneado lineal. Esta técnica "barre" la imagen de izquierda a derecha, y de abajo arriba, para identificar las primitivas proyectadas.

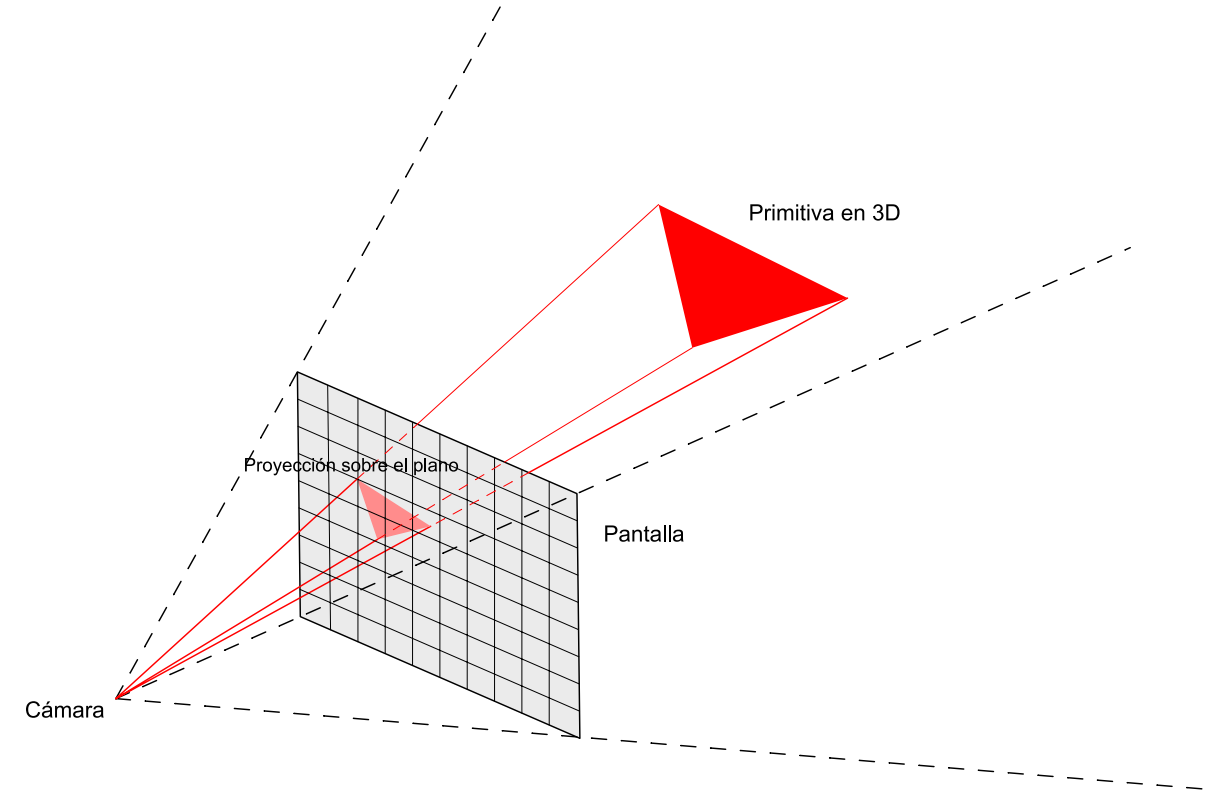


fig. 16 Esquema del funcionamiento de un algoritmo de rasterizado.

fig. 17 Modo de visualización del Z-buffer (o buffer de profundidad) en la escena creada en *Unreal Engine* para este TFM.

Cuando el motor ha interpretado la geometría 3D sobre el plano 2D, este necesita comprender la posición de profundidad de cada píxel, pues solo se pintarán los píxeles que sean visibles desde el punto de vista del espectador. Para ello se recurre a un **Z-Buffer** [fig. 17] que almacena la información de profundidad de los píxeles ya pintados por el render. Esta información se actualiza con cada pasada del escaneado lineal, proporcionándole al motor la información necesaria para saber qué píxeles están ocultos y cuáles han de ser pintados.

No obstante, el rasterizado no proporciona información de color, por lo que este debe ser obtenido mediante otros cálculos adicionales de sombreado y texturizado.

Por último, antes de recabar el color final de un píxel, el motor ha de realizar un cálculo de iluminación que proporcione la información de sombreado sobre los píxeles afectados por diversas condiciones lumínicas.

El rasterizado por escaneado lineal es la técnica más empleada por los motores en tiempo real, pues su velocidad de cálculo es considerablemente superior a la del resto de procesos de renderizado.

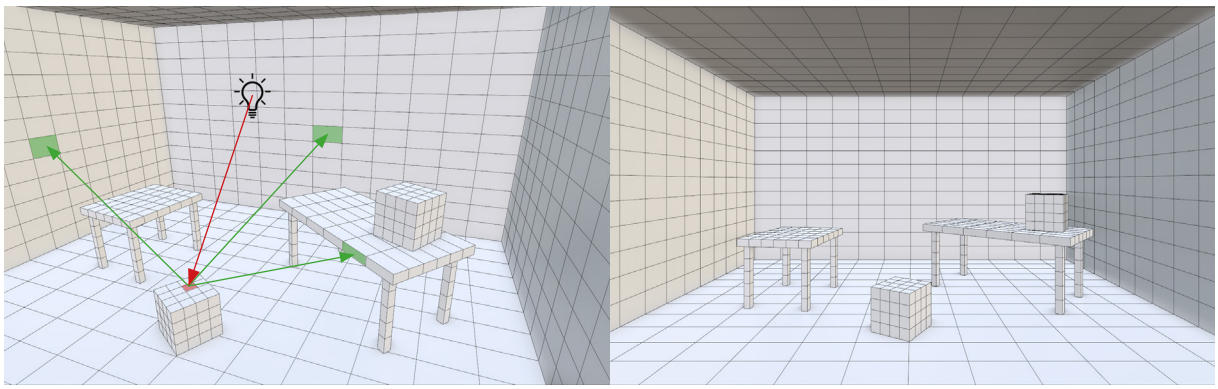
Radiosidad: es el método por el cual se intenta simular cómo la luz difusa reflejada de una superficie también ilumina las áreas a su alrededor. Originalmente, su desarrollo se ideó para resolver operaciones de transmisión de calor, mediante el cálculo de la cantidad de calor que una superficie irradia y cuánto de este es absorbido por otras superficies adyacentes.

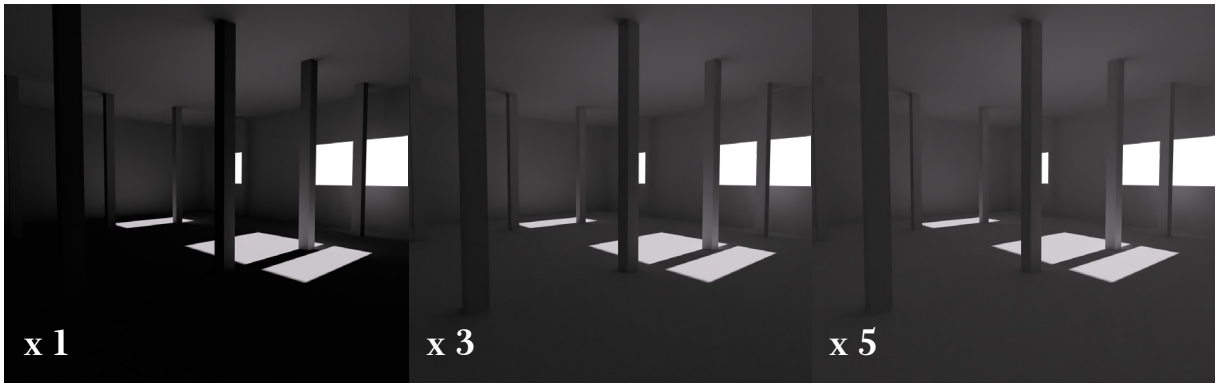
A diferencia del *ray tracing*, que busca calcular todo tipo de rayos, los algoritmos de radiosidad solamente calculan las trayectorias de los rayos de iluminación que parten de una fuente de luz y son reflejados de forma difusa un número finito de veces antes de alcanzar la cámara. Los cálculos de radiosidad producen sombreados realistas y permiten capturar la iluminación global, sobre todo en escenas *indoor* donde el rebote de luz se hace más evidente.

El funcionamiento de un algoritmo de radiosidad es bastante simple. El algoritmo subdivide las superficies a renderizar en porciones de superficies más pequeñas, llamadas parches (*patches* en inglés). Posteriormente, se deduce un **factor de forma** (*view factor o form factor*) por cada par de parches [fig. 18]; este determina la predisposición que cada parche tiene a recibir la luz rebotada del resto de parches. Este factor puede variar en función de la forma de los objetos y del ángulo de cada par de parches individualmente.

El algoritmo resuelve los cálculos de radiosidad con operaciones iterativas. A cada iteración, se calcula la iluminación radiada que cada parche absorbe de los demás. Una parte de la luz se considera absorbida, mientras que el resto es reflejado hacia la escena para la siguiente iteración [fig. 19].

Los algoritmos de radiosidad son extremadamente útiles si se combinan con otros métodos de renderizado. Se han vuelto muy populares en aplicaciones interactivas que requieran cómputo en tiempo real, ya que permite simular efectos de iluminación complejos en tiempos relativamente cortos. Además, en un proceso de pre-cálculo, los resultados de radiosidad se pueden almacenar en mapas de textura llamados *lightmaps* que el motor puede cargar como una textura tradicional sobre el modelo 3D.





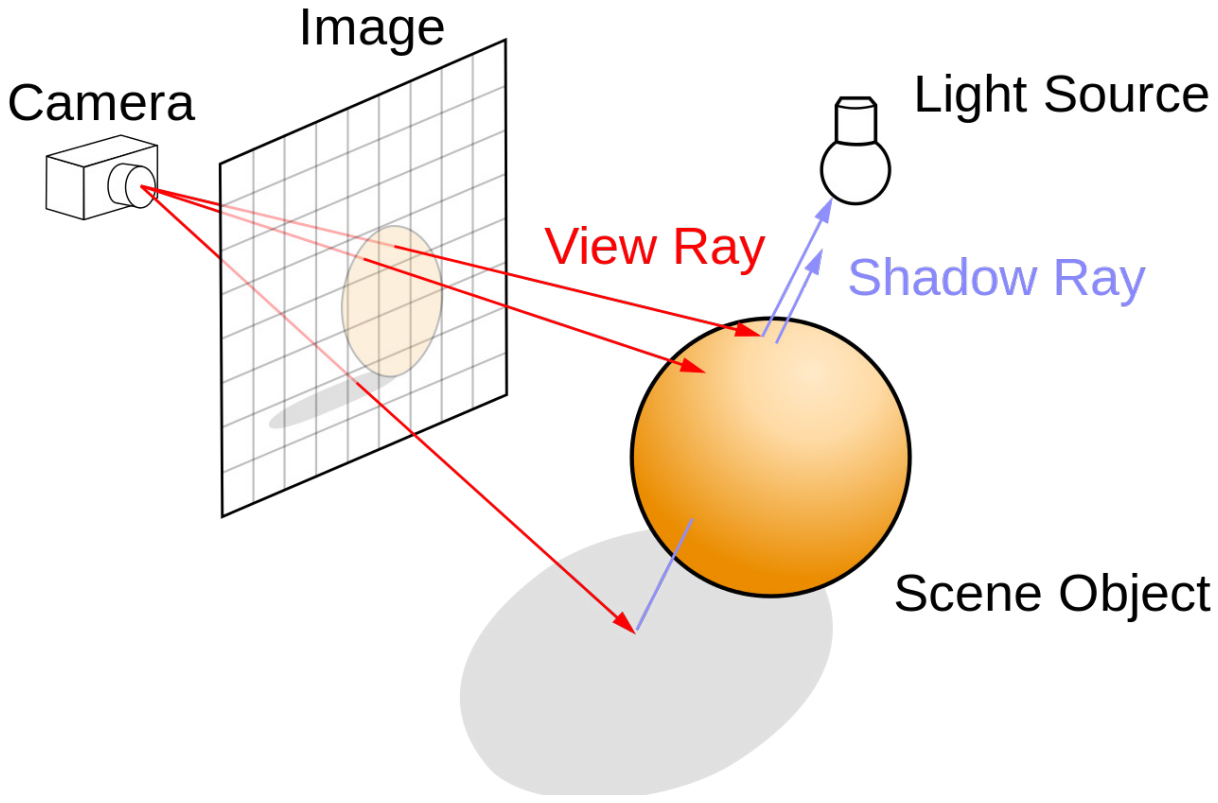


fig. 18 Representación en Blender de los parches y el comportamiento de un algoritmo de radiosidad.

fig. 19 Visualización de iteraciones en un algoritmo de radiosidad para calcular la iluminación difusa en Blender.

fig. 20 Gráfico general de cómo funciona un algoritmo de trazado de rayos. Fuente: Wikipedia.org

***Ray tracing*:** sin duda, uno de los nombres más populares en renderizado 3D. El motor construye la imagen trazando las trayectorias desde un punto de vista (la cámara virtual) hasta el primer objeto 3D que encuentre de la escena, entonces, otros rayos son proyectados hacia cada una de las fuentes de luz presentes en la escena. La intersección de uno de estos "rayos" con el plano de renderizado, situado entre la cámara y el objeto 3D, genera un píxel con la información de la superficie intercedida [fig. 20]. La iluminación se calcula midiendo los rayos de luz que inciden sobre el punto de intersección del rayo principal con el objeto encontrado.

Los motores de trazado de rayos buscan replicar las propiedades físicas de la luz natural siguiendo los mismos principios de la física, pero operando en el sentido contrario. En la naturaleza, los rayos parten de una fuente de luz, inciden sobre una superficie y posteriormente viaja hacia el ojo, que construye la imagen. En un motor por trazado de rayos el camino se invierte, el rayo viaja desde el ojo (la cámara virtual) hasta una superficie y rebota hacia las fuentes de luz. Este procedimiento responde a cuestiones de optimización, pues calcular todos y cada uno de los rayos que parten de una luz sería demasiado trabajo para la GPU. Es mucho más eficiente escoger únicamente las trayectorias que coinciden con cada uno de los píxeles del plano de renderizado, y trazar su camino inverso.

Un aspecto fundamental del *Ray tracing* es la posibilidad de realizar operaciones recursivas sobre una misma trayectoria. Esto permite calcular diferentes tipos de rebote de iluminación sobre un mismo punto de incidencia. Cuando un rayo alcanza una superficie, este puede generar tres nuevos tipos de rayo: **reflejado**, **refractado** o de **sombra**. Los rayos reflejados se trazan en la dirección del reflejo hasta el objeto más próximo, que será dibujado sobre la superficie del material reflectivo. Los rayos refractados viajan a través de superficies translúcidas y operan de manera parecida. Los rayos de sombra viajan hacia cada una de las luces en escena. Si estos encuentran objetos opacos entre la superficie de rebote y la luz, dicha superficie estará en sombra y la luz no la iluminará. El trazado de rayos recursivo añade un plus de realismo al renderizado de imágenes.

Existe una versión mejorada de *Ray tracing* llamada *Path tracing*. En este tipo de algoritmos, los rayos que parten de la cámara, al alcanzar una superficie, no trazan una trayectoria directa hacia las fuentes de luz; en vez de eso, rebotan sobre el resto de objetos de la escena hasta que encuentran en su camino una fuente de luz, o hasta que exceden el número máximo de rebotes definido por el motor. Si en *Ray tracing* un rayo traza una única trayectoria, desde la primera superficie encontrada hasta la luz, en *Path tracing* ese mismo rayo rebota múltiples veces en todas las superficies que encuentre a su paso. Esta técnica emula de manera todavía más efectiva el comportamiento natural de la luz, produciendo resultados mucho más fotorrealistas que los del *Ray tracing* a costa de aumentar los tiempos de render.

La popularidad del trazado de rayos reside en su capacidad para simular el comportamiento de la luz con resultados extremadamente fotorrealistas. Mientras que otras técnicas como el rasterizado centran sus esfuerzos en simular volúmenes y geometrías, los algoritmos de *Ray tracing* sacrifican velocidad de renderizado para mostrar imágenes que responden a los fenómenos de iluminación que acontecerían en el mundo real.

Ray casting*:** esta técnica puede confundirse con los algoritmos de ***Ray tracing por las similitudes en su proceso de renderizado, pero ambas difieren en características concretas. En esencia, el ***Ray casting*** utiliza el mismo principio que el trazado de rayos, pero únicamente calcula la geometría de los modelos 3D próximos de manera no recursiva. El motor traza trayectorias desde la cámara para muestrear la luz que viaja hacia el observador desde la dirección del rayo [fig. 21].

En esencia, la idea tras el ***ray casting*** es poder trazar rayos desde un punto de vista, uno por cada píxel del plano a renderizar, y discernir los objetos próximos que bloqueen la trayectoria de esos rayos. El algoritmo puede determinar el color de dicho píxel conforme al material aplicado sobre la superficie del objeto encontrado, así como a los efectos de iluminación en la escena.

A diferencia del ***Ray tracing***, el ***Ray casting*** no se ejecuta de forma recursiva y solo proyecta rayos primarios, por lo que el cálculo de reflexiones, refracciones o sombras de iluminación global es completamente imposible. No obstante, estos efectos se pueden falsear usando mapas de texturizado y técnicas de sombreado.

La ventaja del ***Ray casting*** frente al rasterizado es su capacidad para interpretar superficies de sólidos no planos o con geometrías complejas. Básicamente, si un rayo puede incidir sobre una superficie, esta puede ser renderizada usando R***ay casting***, mientras que con técnicas de rasterizado es necesario interpretar la posición 3D de los vértices que la componen.

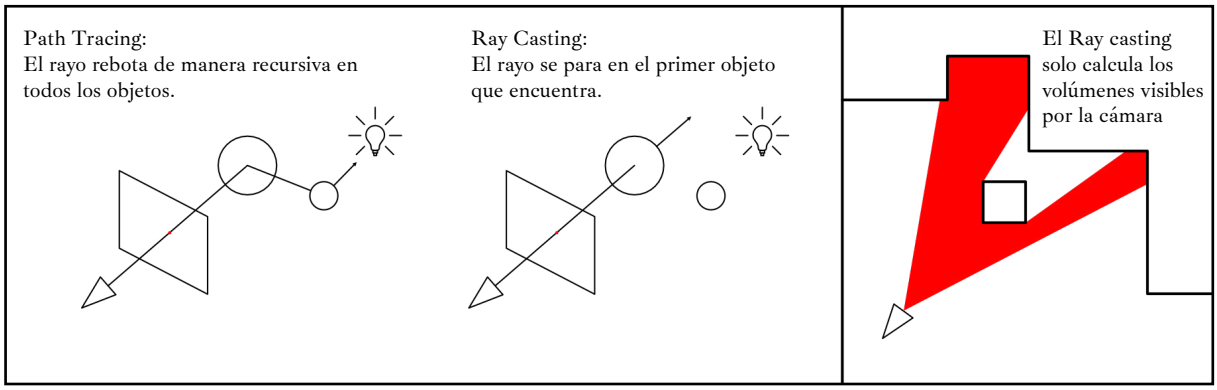
Este algoritmo se popularizó en los 90, pues era el método que empleaban muchos videojuegos [fig. 22] de aquella época para renderizar gráficos en 3D. En resumen, es una técnica que sacrifica los resultados fotorrealistas del R***ay tracing*** con el fin de poder renderizar la geometría de la escena a gran velocidad.

1.7. Integración de cámaras y *tracking*

El objetivo de haber creado una escena al completo es poder operar igual que si fuese un plató, por tanto, los programas 3D integran un actor específico que se comporta como una cámara de vídeo.

Desde las opciones de cámara se pueden manipular todos los parámetros típicos presentes en una cámara física. La idea es hacer coincidir los valores virtuales con los utilizados por el operador de cámara en el ***set*** de rodaje. Así pues, si por ejemplo se ha utilizado una lente 50 mm con una apertura f4 para grabar un plano, a la hora de construir la escena virtual se habrá de fijar una cámara que mantenga los mismos ajustes, de este modo el CGI y la imagen real conservarán las mismas propiedades ópticas.

Valorar todas estas consideraciones es trabajo de los artistas VFX, quienes habrán de manejar no solo los parámetros de la cámara, también deberán considerar efectos ópticos inherentes a las lentes utilizadas, a los artificios producidos por el sensor de la cámara y a las peculiaridades de cada equipo de grabación en particular. Por tanto, será necesario recrear efectos como cambios de foco, el ***bokeh*** específico de una lente, los ***lensflare*** sobre plano, la profundidad de campo o el nivel de desenfoque por movimiento.



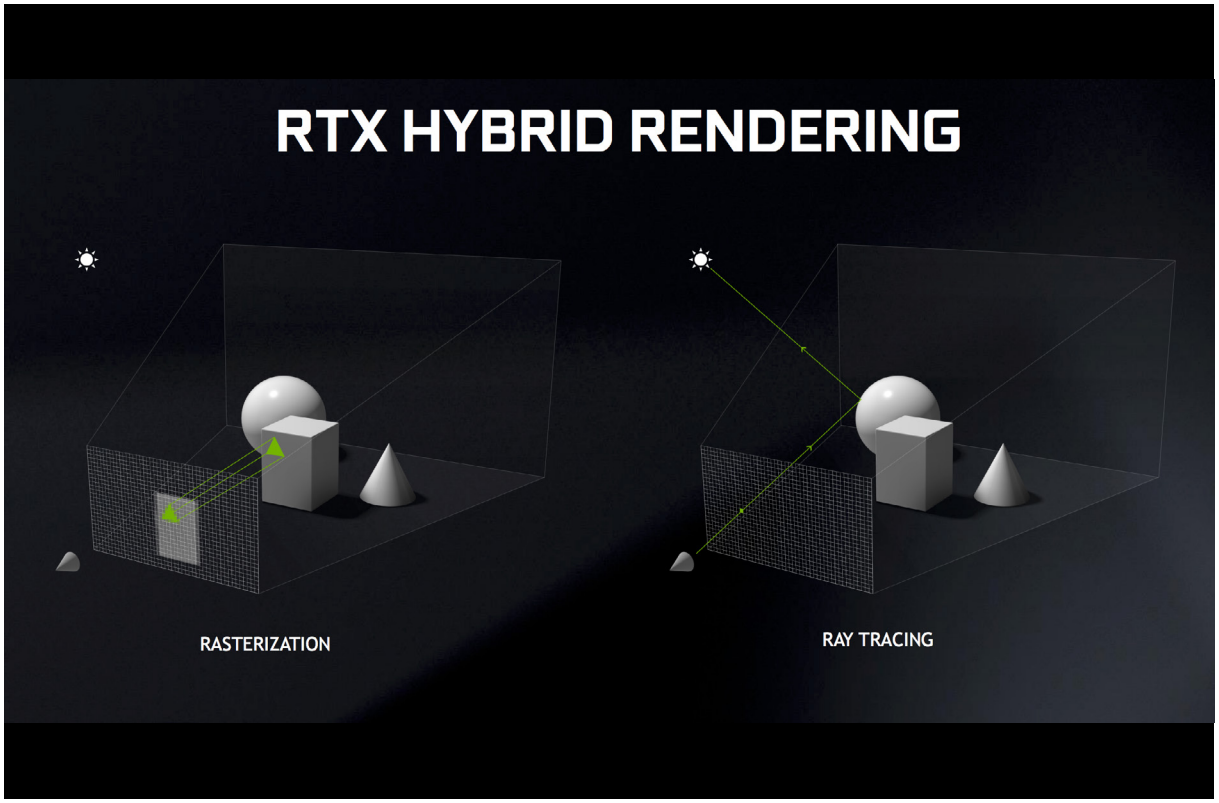


fig. 21 Gráfico esquematizado del funcionamiento de un algoritmo de *Ray casting*.

fig. 22 Imágenes de los videojuegos *Wolfenstain 3D* (ID Software, 1992) y *Doom* (ID Software, 1993), respectivamente.

fig. 23 Esquema de diferencias en el renderizado por rasterizado y por *Ray tracing*. Fuente: nvidia.com

Otro aspecto fundamental de una cámara virtual es el movimiento que realiza en el entorno 3D. Igual que sucede con las cámaras físicas, las cámaras virtuales han de ser operadas correctamente para recrear movimientos de cámara concretos.

Este concepto adquiere una dimensión completamente diferente cuando lo que se pretende es hacer coincidir milimétricamente los movimientos de cámara real empleados para rodar una escena con los de la cámara virtual, para que así los elementos CGI que aparezcan en escena reproduzcan las mismas situaciones de plano que los actores y el decorado práctico. Reproducir estos movimientos es relativamente fácil si la cámara física realiza una acción lineal de *traveling* o si, directamente, esta no se mueve y el plano es fijo. Pero la tarea se complica cuando la escena se ha rodado cámara en mano.

Para emular movimientos complejos se recurre a técnicas de ***tracking*** que permiten discernir el movimiento que ha ejecutado la cámara física en base al tipo de desplazamiento que una serie de marcadores, definidos por el artista VFX, realizan sobre el plano. Estos marcadores se pueden colocar físicamente sobre el decorado o, por el contrario, se pueden seleccionar en posproducción usando como referencia una parte de la imagen.

1.8. Proceso de composición

La etapa de composición digital se sitúa entre las últimas fases dentro de la cadena de posproducción; es el punto en el que el metraje original y el metraje generado por ordenador se combinan para crear una secuencia única que pueda ser incorporada como escena final dentro de la cinta.

La composición bebe de muchas disciplinas diferentes, y se caracteriza por combinarlas todas según las necesidades del plano. Así pues, es indiferente si el material que se quiere componer es un plano en *matte painting*, un *clean plate* o un personaje CGI, el proceso buscará el mismo objetivo para cualquiera de estos elementos.

Como si de un *collage* se tratase, la composición digital (*compositing* en inglés) se vale de varias técnicas para combinar en un mismo plano elementos de distinto origen. Como define García Barroso (pág. 21, parr. 3).

> Para que una composición funcione, todos los elementos aislados tienen que combinarse de una manera específica que facilite su visionado [...] Estos elementos deben actuar independientemente en el plano, y para ello, habrá que extraer, enmascarar y seleccionar áreas de la imagen.

Las técnicas más utilizadas para aislar elementos de una imagen ante la cámara son los procesos de extracción por clave de color, más conocidos como *chroma* o *green screen/blue screen*. Situando un objeto sobre un fondo con un único color, que no comparta gama cromática alguna con el objeto a recortar, se puede obtener un canal *matte* que aísle la silueta del objeto. Esto es posible gracias a la extracción del color de fondo, que normalmente se sitúa en un canal RGB diferente al de la figura que se va a recortar.

Por diferentes motivos, es posible que el *chroma keying* no pueda aislar correctamente las figuras del fondo, en este caso es probable que se necesite retocar la imagen por **rotoscopia**.

2. Segunda Parte: *Performance Capture*

Cuando todavía no existían efectos digitales y los medios analógicos dominaban el cine, la única manera de obtener el movimiento de una criatura ficticia ante la cámara era filmar una marioneta y animarla mediante ***stop-motion*** para posteriormente componer ambos metrajes. El ejemplo más representativo de esta técnica lo encontramos en ***Jasón y los argonautas*** (Don Chaffey,1963) [fig. 24], película que deslumbró por el planteamiento tras sus efectos especiales, a pesar de la escasa recaudación que obtuvo en la taquilla.

Las criaturas creadas por Ray Harryhausen para el filme, eran marionetas con un esqueleto articulable que permitía posados en diferentes posiciones. Gracias a la magia de la animación, se consiguió una integración fresca y llamativa con el metraje original, coreografiando las poses de los monstruos con el movimiento ejecutado por los actores ante la cámara.

Pero el proceso de animar por ***stop-motion*** es cuanto menos tedioso, pues para realizar una animación efectiva se ha de tomar una foto, con la marioneta en la pose deseada, por cada uno de los fotogramas presentes en la secuencia que se pretende rellenar. El resultado es muy plástico y con una estética ciertamente poética, pero está lejos de parecer fotorrealista; además, supone un enorme volumen de trabajo para los animadores por la cantidad de horas que se han de dedicar a una buena animación.

Entre otras posibles soluciones, la caracterización de criaturas y otros personajes podía lograrse mediante maquillaje, disfraces y/o animatrónica. Varios monstruos del cine, como los Critters (Stephen Herek y Scot Grimes, 1986) o los famosos Gremlins de Joe Dante (1984) se han resuelto con efectos prácticos ante cámara, sin necesidad de animar escenas externas. Actores caracterizados interpretaban estos personajes y les daban vida con su actuación o, en el caso de la animatrónica, un operario controlaba las marionetas y las movía de forma más o menos convincente. En ciertos aspectos, el acabado final puede considerarse bastante logrado; en el caso de la caracterización con maquillaje, no existía conflicto alguno en el movimiento y la actuación, pues dependía íntegramente de las dotes interpretativas del actor, y en cuanto a las marionetas por animatrónica, si bien es cierto que el movimiento era torpe y las expresiones limitadas, en términos fotográficos, estaban perfectamente integradas con el resto de elementos en la escena, y esa actuación artificial les daba cierto encanto que el espectador aceptaba con gusto.

Pero, a pesar de la magia tras la practicidad de estos efectos, existe un límite en lo que con ellos se podía llegar a recrear ante la cámara. Personajes como los alienígenas que alegremente beben y cantan en la escena de la cantina, en el cuarto episodio de *Star Wars*, son fáciles de caracterizar, pues su morfología es próxima a la de un ser humano; incluso criaturas menos antropomórficas como E.T. consistían en un disfraz con partes de animatrónica para simular las expresiones faciales. Pero ¿qué pasa cuando se ha de mostrar, por ejemplo, un T-Rex en pantalla? Bueno, la respuesta nos la ofreció Spielberg con el diseño de los efectos visuales de *Parque Jurásico* (1993).

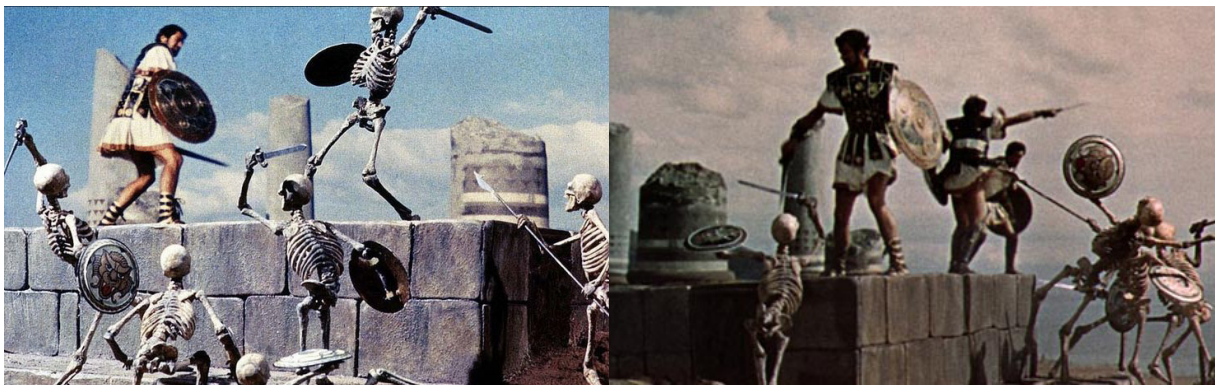


fig. 24 Fotogramas de la escena de los 7 esqueletos, en *Jasón y los argonautas*.

fig. 25 Izquierda: marioneta animatrónica a tamaño real (imagen del *set* de rodaje). Derecha: pata del tiranosaurio de animatrónica para solucionar un plano cerrado (fotograma de la película).

fig. 26 T-Rex CGI en la escena de *Parque Jurásico*. Fotograma de la película original de 1993.

El equipo de ILM (*Industries Light And Magic*) decidió combinar tanto la animatrónica como el maquillaje y las prótesis, además de integrar una nueva tecnología en ciernes, la animación por CGI, para dar vida a los dinosaurios del filme. En aquellos años, la computación gráfica todavía presentaba serias limitaciones, y el equipo de VFX era muy consciente de ello. Por tanto, diseñaron un plan de producción que aprovechase lo mejor de cada campo para resolver los efectos específicos de cada plano.

En un principio, los dinosaurios de *Parque Jurásico* iban a ser animados por *stop-motion*, como se había hecho en otras cintas años antes. Steven Spielberg contrató a Phil Tippett, a quien le abalaban sus trabajos en grandes producciones como *Star Wars* y *Robocop*, para que se encargase de las secuencias de *stop-motion*, pero Denis Muren de ILM convenció a Spielberg de que las animaciones por CGI podrían solventar los planos destinados al *stop-motion* con mayor credibilidad. Tippett, lejos de abandonar el proyecto, terminó supervisando al equipo de efectos especiales de ILM, lo que acabó valiéndole un Oscar por su trabajo recreando el movimiento de estos enormes animales prehistóricos para la gran pantalla.

Es precisamente esta buena sinergia entre efectos prácticos y CGI lo que caracteriza el desarrollo de los efectos especiales en *Parque Jurásico*. Lejos de limitarse a forzar una sola técnica para resolver una o varias escenas, la gente de ILM decidió aprovechar lo mejor de cada campo para solucionar planos concretos dentro del metraje. Cuando se debían filmar tomas generales en las que tanto actores como dinosaurios de gran tamaño aparecían en cuadro, se empleaba tecnología 3D para recrear a estas criaturas, manteniendo movimientos de cámara sutiles con ángulos muy abiertos [fig. 26]. Para encuadres más cerrados, planos detalle o escenas de acción, se utilizaban efectos prácticos como marionetas y disfraces. La animatrónica jugó un papel fundamental en la película, pues se construyeron enormes modelos a tamaño real que, en muchas ocasiones, valieron como referencia fotográfica para las recreaciones 3D.

Una de las escenas más icónicas del filme, cuando el T-Rex escapa del recinto y ataca los coches con los protagonistas dentro, es un ejemplo perfecto de cómo un buen *planning* puede evitar resultados desastrosos y la película acabar resultando una obra maestra.

La escena se desarrolla enteramente de noche, con luz puntual, en esta ocasión un foco sobre la alambrada, o los faros de los coches en otros planos. Esta decisión puede parecer una mera elección estética, pero mantener esa iluminación permitió disimular las carencias del CGI de la época, pues con destellos puntuales es más fácil disimular las animaciones por ordenador. Cuanto más exagerados y artificiales son los efectos de luz más fácil es camuflar el 3D.

Además, toda la acción se desarrolla bajo una intensa lluvia. El exceso de humedad sobre la piel del tiranosaurio permitió diseñar una textura con altos índices de especularidad, haciendo que toda la piel reflejase la misma intensidad lumínica. Con toda la escena literalmente empapada, los reflejos sobre las superficies del decorado práctico se integraban a la perfección con los del T-Rex, que por otro lado, solamente debía reflejar la luz proveniente de un único *spot*. Con esta estrategia, el equipo de ILM se ahorró una enorme cantidad de tiempo al poder

obviar los cálculos de iluminación rebotada que tendrían que haber realizado si se hubiese diseñado la escena con otras condiciones ambientales. Esta máxima se ha mantenido intacta durante los últimos 20 años y la veremos repetida en numerosas producciones de Hollywood (ej. ***Batman vs Superman*** (Snyder, 2018)).

Por otro lado, en esa misma escena, para encuadres más cerrados en donde se requería más detalle, se utilizaron enormes marionetas de animatrónica que simulaban partes del tiranosaurio (cabeza, patas, etc.) [fig. 25] que eran operadas al mismo tiempo que los actores actuaban. Esto proporcionó una referencia fotográfica directa de cómo debía verse el modelo 3D, haciendo que fuese más fácil calcular las zonas de máxima iluminación y sombra, así como el nivel de especularidad sobre la piel.

Los efectos especiales que le valieron un Oscar a ***Parque Jurásico*** no se consiguieron gracias a un CGI extremadamente fotorrealista, pues la tecnología de la época no lo permitía, sino gracias a una buena planificación en el diseño de producción. Hoy día las películas con unos VFX brillantes no suelen ser aquellas con el CGI de mejor calidad o con las animaciones más impresionantes, sino esas otras que integran 3D con efectos prácticos de la manera más eficiente, tal y como se hizo en Parque Jurásico, y que son capaces de engañar al espectador hasta el punto de hacerle dudar sobre lo que es CGI y lo que no lo es.

Sin duda, el filme de Spielberg sentó las bases de lo que estaba por venir en el mundo de los efectos especiales, preparando el camino para futuras producciones y demostrando que el 3D sería la tecnología más viable para los VFX de un futuro próximo, pero que estos deberían ser utilizados con coherencia.

A pesar de las implicaciones técnicas que encontramos detrás del CGI, la animación 3D se rige por los mismos principios y reglas que la animación tradicional, introducida en Hollywood por los estudios Disney y por los animadores de ***stop-motion***. Aunque el medio sea muy diferente, un animador 3D ha de respetar fundamentos básicos comunes a cualquier animación. Por ello, a pesar de los tecnicismos, un animador 3D no deja de ser, en esencia, un artista que pone su arte a disposición de una visión estética mayor.

2.1. Humanoides digitales

Hemos visto como la animación juega un papel fundamental dentro de la credibilidad de los efectos especiales. Mediante animación por ***rigging,*** un animador puede recrear movimientos muy similares a los de un ser humano, o dar vida a los de cualquier otra criatura ficticia. La animación por ***rigging*** se ha mantenido intacta durante mucho tiempo, y todavía hoy es necesario ajustar manualmente el movimiento de los manejadores en el esqueleto de un modelo 3D para lograr la acción deseada.

No obstante, aunque sea posible, esta estrategia no es siempre la más adecuada o la que mejores resultados ofrece. Muchas veces, la cantidad de datos de movimiento que el animador ha de interpretar es demasiado grande para poder reproducir una acción creíble. Si tenemos en cuenta la gestualidad de un ser humano, existen millones de sutiles movimientos que definen el carácter de una acción, la mayoría de ellos realizados de forma totalmente inconsciente por parte del propio actor.

Es precisamente en esta enorme cantidad de gestos donde reside el principal problema cuando se trata de animar modelos antropomorfos, ni que decir cuando se trata de seres humanos enteramente recreados por CGI [fig. 27].

Ante esta situación, surge la necesidad de encontrar un sistema capaz de recabar los datos de animación necesarios para posicionar los marcadores virtuales de un *rigging* de manera automática, sin que el animador tenga que partir de cero para posar la figura. Hasta hace no mucho, los animadores solían animar el *rigging* de los personajes 3D a mano, uno a uno, intentando aproximarse lo máximo posible a un movimiento creíble y semi-realista. Como era de esperar, esta tarea podía resultar increíblemente complicada, consumiendo horas y horas del tiempo de posproducción para ofrecer resultados no demasiado satisfactorios.

Algunos de los precedentes que la historia del cine nos ha legado son sin duda magníficos trabajos de animación a mano. Es el caso del primer Gollum que apareció en el segundo filme de la trilogía *El señor se los anillos* (Jackson, 2001). Weta Digital, empresa de efectos especiales fundada por Peter Jackson, decidió crear esta criatura completamente en 3D, pero por aquel entonces los medios necesarios para capturar el movimiento de un actor en tiempo real *on-set* no estaban tan desarrollados. Por tanto, primero se grabó la actuación del actor que interpretó a Gollum directamente en las localizaciones de rodaje junto al resto de actores, sin contar con ningún sistema de captura de movimiento. Meses después, tras finalizar el rodaje, se repitieron todas las escenas de Gollum en un estudio preparado con sistemas de captura de movimiento que registraban puntos esenciales del esqueleto (codos, rodillas, cadera, etc.). Los planos se animaron teniendo en cuenta los datos registrados, pero otros se tuvieron que animar enteramente a mano, incluyendo las expresiones faciales en las escenas del monólogo de la criatura con su *alter ego*.

Se contó con la participación de Andy Serkins, quien interpretó a Gollum ante la cámara, para que los animadores tuviesen una referencia real de la gestualidad interpretativa que debía alcanzar la animación. El actor vestía un traje completamente blanco, que solo mostraba la cara, y tomando como referencia sus gestos y sus movimientos, el equipo de Weta sustituyó en posproducción la imagen real del actor por el modelo 3D animado de la criatura Gollum [fig. 29].

Las escenas de monólogo entre las dos personalidades de Gollum fueron especialmente desafiantes. Se grabó a Serkins actuando ante la cámara sin ningún tipo de marcador facial, sin ningún punto de referencia que los animadores pudiesen usar, y a partir de las expresiones realizadas por el actor se adaptaron los movimientos faciales de Gollum para que imitase la misma gestualidad que Serkins mostró en su actuación.

Por otro lado, tenemos ejemplos nefastos de animación a mano. Es el caso de la escena de *El regreso de la momia* (Sommers, 2001) en donde la animación facial del modelo 3D de Dwayne Jhonson [fig. 28] genera una inquietante sensación de rechazo. La opinión sobre esta escena es unánime por parte del público, casi nadie considera que esta animación esté bien resuelta o pueda pasar los estándares de calidad de Hollywood, especialmente si tenemos en cuenta que *El regreso de la momia* se estrenó solo un año antes que *El señor de los anillos: las dos torres*. Pero ¿por qué motivo exactamente no convence la animación de esta película?

fig. 27 A la izquierda, comparación entre el actor real Peter Cushing (izquierda, fotograma de *Star Wars* (1977); derecha, su recreación CGI en un fotograma del filme *Star Wars: Rogue One).* A la derecha, fotograma de *Star Wars: Rogue One*, personaje original de Carrie Fisher recreado enteramente por CGI.

fig. 28 Fotogramas del rey escorpión en la película *El regreso de la momia.* Imagen CGI de Dwayne Jhonson.

fig. 29 Primer plano del monólogo de Gollum en *Las dos torres*. A la derecha, Andy Serkins actuando en el set de rodaje. A la izquierda, modelo 3D rudimentario de Gollum utilizado en *previs*. Fuente: Nature Video.

2.1.1. El valle inquietante

Los problemas en animaciones faciales y recreaciones digitales de seres humanos están vinculados a un concepto propio de la psicología humana. Para entender una de las motivaciones clave que impulsó la implementación de la captura de movimiento dentro del flujo de producción, es necesario aclarar un concepto que ha sido de vital importancia en el desarrollo del CGI en cualquier medio audiovisual, no solo en cine, conocido como valle inquietante (del inglés ***Uncanny Valley***).

A principios de los años 70, el profesor Masahiro Mori introdujo este término como hipótesis en el campo de la robótica, pero con el perfeccionamiento de los gráficos por ordenador rápidamente empezó a cobrar sentido dentro del mundo de la animación 3D.

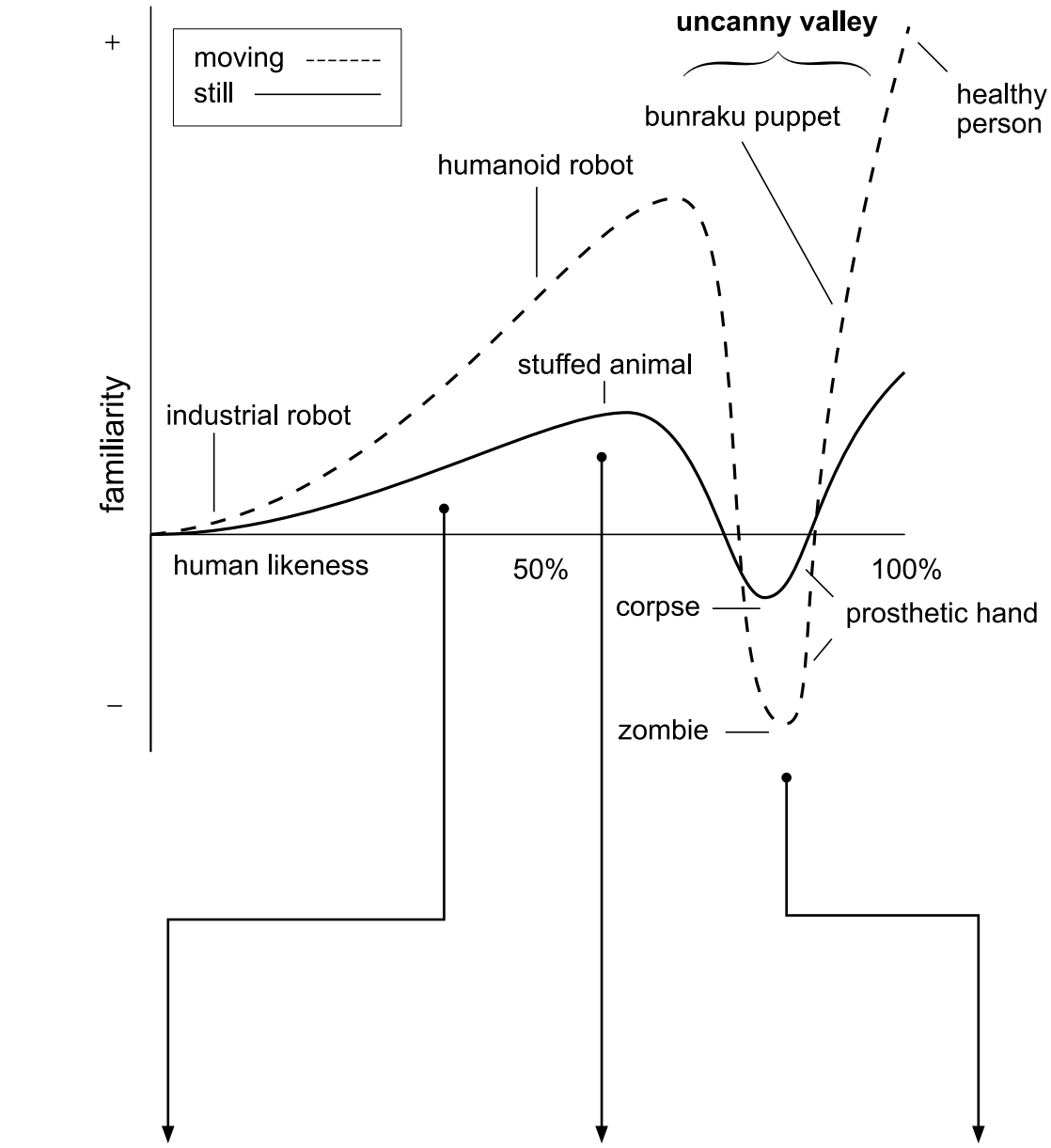
El concepto detrás del valle inquietante explica cómo los seres humanos experimentan un sentimiento de repulsión y rechazo hacia las réplicas humanoides que se asemejan en apariencia, pero difieren en comportamiento y gestualidad. Según el profesor Mori, un robot con apariencia demasiado humana puede virar hacia un territorio perturbador, activando las mismas alarmas psicológicas asociadas con un humano muerto o enfermo (Sofge, 2010).

Esta respuesta psicológica se puede visibilizar mediante un gráfico en el que la curva cae estrepitosamente tanto cuanto el aspecto de la recreación se aproxima más a los rasgos físicos de un humano [fig. 30], de ahí el término "valle" en el nombre de la hipótesis. La respuesta psicológica del espectador que interactúa de algún modo con el personaje artificial, ya sea robot o CGI, guarda una relación directa entre la empatía y el parecido humano; creaciones con menos apariencia humana son menos referenciales y, por tanto, será más fácil para el espectador mantener una distancia empática.

Se puede conseguir una respuesta positiva con modelos que actúen o tengan comportamientos humanos, pero cuya apariencia se aleje de la humana, mientras que, si se intenta recrear físicamente un ser humano 100% realista, muy probablemente se acabe cayendo en el valle inquietante, debido a la cantidad de gestualidad y comportamientos inconscientes que se han de reproducir. En palabras de Sofge (2010):

> *Stick to Roombas and blue-skinned aliens and you'll be fine. But build a realistic feminine android or render a CG version of Tom Hanks in a train conductor's outfit, and the uncanny valley will swallow you whole.*

Uno de los ejemplos más recientes es el de la película *Cats* (Hooper, 2019), estrenada las navidades del año pasado. Para el espectador, es evidente que los personajes de *Cats* son gatos antropomorfizados, pero algo en su morfología y, sobre todo, en las caras reales de los actores pegadas sobre los modelos 3D, hace que salten todas las alarmas del subconsciente. Los gestos, la inconsistencia del movimiento, la iluminación pseudo realista y la mala combinación entre CGI e imagen real provoca que el diseño VFX de esta película habite irremediablemente en el valle inquietante.



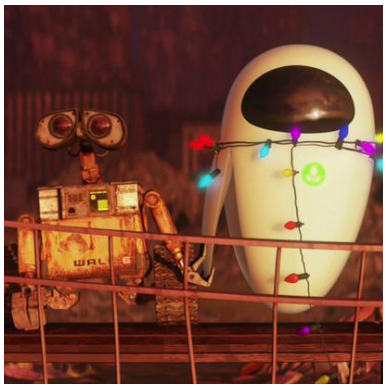


fig. 30 Arriba, Gráfico original del "valle inquietante" propuesto por de Masahiro Mori. Abajo, de izquierda a derecha, wall-e (personaje sin apariencia humana con comportamientos humanos), los personajes de Up (personajes humanos con comportamientos humanos, pero caricaturizados y alejados de una apariencia real) y Judi Dench en Cats (facciones humanas hiperrealistas, comportamientos y conductas humanas).

Hoy en día, todavía resulta muy difícil explicar el motivo exacto por el cual se produce esta respuesta del subconsciente. Entran en juego factores psicológicos inherentes al ser humano, que radican en impulsos primarios y que difícilmente pueden ser suprimidos. Se puede encontrar multitud de artículos e investigaciones que intenta explicar este fenómeno, desde respuestas reproductivas hasta la autoconsciencia de la propia mortalidad. El caso es que, por motivos diversos, este fenómeno existe y es algo que se ha de tener en cuenta cuando se diseña un personaje humanoide.

Los artistas 3D intentan por todos los medios evitar caer en el valle inquietante a la hora de animar personajes CGI. Con el transcurso de los años se han desarrollado varias estrategias para minimizar esta sensación de rechazo. Una de estas estrategias ha consistido en recabar los datos de animación directamente a partir de la actuación en tiempo real de un actor real. Si se pueden registrar todos los gestos de una persona, incluso aquellos hechos de manera inconsciente, probablemente se puedan trasladar a una animación que esté próxima a la gestualidad natural de un ser humano.

Este, junto con el ahorro de tiempo en producción, es uno de los motivos por los cuales se opta por capturar la actuación de un actor real y trasladarla al personaje 3D.

2.2. De *Motion Capture* a *Performance Capture*

Si bien el término *performance capture* ha tomado fuerza a lo largo de los años, las diferencias entre ambos conceptos son meramente conceptuales. Más que un conjunto de técnicas y tecnologías en sí, la *performance capture* es un concepto: la realización de una captura de movimiento de la acción interpretada por un actor en tiempo real; no solo de los movimientos de su cuerpo, también de todas sus expresiones faciales y de la gestualidad de su interpretación. Literalmente, se pretende **capturar la actuación** y transportarla al personaje 3D como se haría ante una cámara convencional.

Años atrás, cuando se debía realizar una captura de movimiento, no era un actor profesional quien desempeñaba necesariamente dicha tarea. Los datos que se pretendía recabar no eran demasiados y la captura de expresiones faciales todavía no se consideraba apta para ser implementada dentro de los flujos de producción, de modo que los sistemas de *motion capture* se limitaban a unos pocos marcadores en las articulaciones y en la cabeza, lo justo para registrar una idea aproximada del movimiento que debía realizar el *rigging* de un personaje.

Pero con el avance de la tecnología, los procesos de captura de movimiento comenzaron a ser fundamentales para la correcta animación de modelos humanos, e imprescindibles si se quería evitar al máximo el valle inquietante. Las mejoras en arquitectura de GPU permitieron gráficos cada vez más realistas que necesariamente debían ir acompañados de animaciones que empatasen en calidad y credibilidad.

El *motion capture* se comenzó a desarrollar con el fin de mejorar las animaciones y de agilizar los tiempos de producción, pero con los avances que año tras año la industria ha ido presentando, esta técnica ha adquirido una dimensión

completamente diferente, pudiendo materializar uno de los mayores deseos dentro del sector de la animación 3D: la actuación en tiempo real de un modelo completamente renderizado y con calidad final.

La posibilidad de incorporar actuaciones reales a modelos 3D ha permitido que surja toda una tendencia en el proceso de producir contenido audiovisual, que ha llevado a replantear los tiempos y los pasos a seguir en la producción de cada proyecto.

En la primera década del milenio se empezó a considerar la *performance capture* como un elemento esencial que debía formar parte del resto de procesos propios de la posproducción. Pero la captura de la acción todavía estaba separada de la actuación en el *set* de rodaje y cada toma debía repetirse en un plató especialmente preparado para ello (ej. *Señor de los anillos: las dos torres*]. Esto se debía a que la industria del cine todavía seguía estándares basados en la filmación de imágenes de acción real, y, por tanto, descartaba toda tecnología que no pudiese alcanzar un nivel de calidad fotorrealista. Si volvemos a tomar como ejemplo los planos de Gollum, casi todos los elementos en escena son parte del vídeo mientras que el CGI constituye una parte minoritaria.

Sin embargo, paralelamente a la animación 3D, estaba emergiendo otro sector que basaba todo su contenido únicamente en gráficos generados por ordenador, y que empezaba a tener una gran repercusión en la sociedad. Me refiero, por supuesto, al sector del videojuego.

Durante los años 80, los videojuegos se limitaban a planos bidimensionales de píxeles, pero a mitad de la década de los 90, se dio el siguiente paso en el entretenimiento doméstico al incorporar gráficos 3D en toda una generación de videoconsolas con procesador de 32 y 64 *bit*, como la primera Playstation y la Nintendo 64. El auge de las videoconsolas enfocadas al 3D y la aparición de las primeras tarjetas gráficas con aceleración 3D (ej. 3dfx Vodoo, S3 Virge), hizo que todos los desarrolladores fijasen la vista en esta tecnología como un medio viable para sus proyectos.

Desde entonces, el camino del cine y el de los videojuegos se ha retroalimentado mutuamente, incorporando estrategias y tecnologías propias de un medio al otro. No obstante, el enfoque particular de cada medio ha condicionado el modo en que se aborda el uso de gráficos CGI.

Mientras que en la producción de una película, al ser un medio no-interactivo, el CGI puede incorporarse durante la última fase de producción, en un videojuego este formaba parte de su desarrollo desde el primer minuto. Esta diferencia, que puede parecer trivial, ha definido el punto de inflexión para el desarrollo de gráficos orientados a motores de videojuego. Si en la producción de una película se pueden emplear tiempos de renderizado elevados para obtener renders de gran fotorealismo, en un videojuego se debe alcanzar la máxima calidad gráfica sin que el rendimiento del motor se vea afectado.

Esta condición ha motivado a los desarrolladores de videojuegos a investigar fórmulas que permitan mejoras de desarrollo relacionadas con la ejecución en tiempo real.

2.2.1. El caso de Hellblade: Senua's Sacrifice

En 2017, el estudio londinense Ninja Theory presentó un título que desafiaría el modo en el que hasta entonces se habían producido los videojuegos, y que abriría las puertas a una nueva tendencia de producción.

Según cuenta Tameem Antoniades, director creativo y cofundador de Ninja Theory, la motivación principal que impulsó el desarrollo de ***Hellblade: Senua's Sacrifice*** era poder crear un proyecto independiente con total libertad, pero con calidad de un videojuego triple A [10]. Se pretendía ocupar un espacio de mercado que hasta entonces no había sido explorado por ningún estudio, situado entre los juegos independientes de bajo presupuesto y los ***blockbuster***; el equipo de Ninja Theory llamó a esta nueva idea de producto "***videojuego triple I"*** o "***Independent triple A***". En declaraciones previas a la fase de desarrollo, Antoniades manifestó su deseo de poder emprender un proyecto complejo sin tener que sufrir las presiones y las exigencias de ventas a las que grandes estudios y compañías estaban sometidos. Existe en el manifiesto de ***Hellblade*** un deseo de volver a crear videojuegos, no solo por los beneficios, no por las expectativas de mercado, sino por la necesidad artística de comunicar y de contar una buena historia. Como dice el propio Antoniades (Matthews, 2014):

> ***With our Independent AAA game, we want to maintain our culture as a team striving for bleeding edge design, art and story but in a new format that doesn't compromise creativity and diversity.***

El equipo de desarrollo que trabajó en ***Hellblade*** se componía de 20 personas y su presupuesto era bastante ajustado, un poco mayor que el de un juego indie convencional, pero considerablemente menor que el de un proyecto triple A. Ninja Theory se comprometió con los fans a lanzar el juego por no más de 30 euros (en formato digital) y a publicar un seguimiento periódico del progreso del desarrollo del proyecto. Estos compromisos les obligaron a explorar soluciones poco ortodoxas, pero bastante creativas, con el fin de poder avanzar a través de las diferentes fases de producción.

Desde el principio el objetivo del videojuego era contar la historia de su única protagonista, quien estaría en plano el 90% del tiempo, por lo cual, todos los esfuerzos debían centrarse en conseguir al máximo posible que el personaje de Senua [fig. 31] fuese creíble a los ojos del jugador. Por tanto, era imprescindible diseñar una heroína que conectase con la audiencia, con una animación convincente y una actuación a la altura del guion que se pretendía narrar. En el transcurso de la historia aparecerían otros personajes, la mayoría enemigos y antagonistas, pero estos deberían ser más genéricos y consumir menos tiempo de desarrollo si se quería alcanzar una buena calidad en el diseño de Senua dentro de los plazos y el presupuesto marcados.

Otro deseo del equipo de desarrollo era conferir al videojuego un estilo visual acorde a la narrativa de otros títulos triple A. El lenguaje cinematográfico en la composición de las escenas sería fundamental para alcanzar estos estándares, por lo que se debían utilizar estrategias de filmado próximas a las del cine y la televisión. Así pues, se recurrió a la ***performance capture*** como proceso fundamental para

10 . En el sector del videojuego, un triple A es un proyecto de gran presupuesto y con un equipo de desarrolladores considerable. Es equiparable al término ***blockbuster*** para el cine.

dar vida a Senua, ya que la actuación jugaría un papel crucial en la credibilidad del personaje. Todos los esfuerzos de desarrollo debían centrarse en construir una heroína capaz de cautivar a los jugadores y de transmitir la historia.

En principio, se quiso contratar a una actriz profesional para interpretar el papel de Senua durante todo el videojuego. Mientras se buscaba a alguien apta para desempeñar el rol, el equipo se valió de uno de sus miembros, Melina Juergens [fig. 32], quien estaba a cargo de filmar y montar los vídeos para el diario de desarrollo, para hacer de ***stand-in*** [11] en los procesos de ***testeo*** y prueba durante la fase de prototipado del personaje (***test*** de iluminación, calibración de cámaras y sistemas de captura, digitalización, modelado, ***shading, etc.***). Al final, todos estaban tan acostumbrados a trabajar con Melina que se tomó la decisión de que ella interpretase el papel de Senua, y lo que podría parecer una elección avocada al fracaso acabó convirtiéndose en un éxito rotundo. Ya sea por su cercanía con el equipo, por su implicación con el proyecto, o por un talento innato, la actuación de Melina Juergens fue soberbia, valiéndole el TGA [12] de aquel año a la mejor interpretación.

Relegar todas las escenas cinemáticas de un proyecto a procesos de captura de movimiento es una decisión que puede resultar bastante cara, así que tuvieron que explorar algunas soluciones menos ortodoxas para abaratar los costes de producción. Con este objetivo en mente, surgió la idea de abordar la narrativa desde el concepto de ***performance capture***. En la animación de personajes de un videojuego, lo normal, en una producción tradicional, es tener a una persona que interprete los movimientos corporales, otra que se encargue de la gestualidad y las expresiones faciales, y, por último, la voz es grabada por un tercer actor de doblaje cuando la escena está finalizada. La conclusión a la que llegó el equipo a cargo de Hellblade fue intentar concentrar todas esas etapas en el propio set de rodaje, es decir, que tanto el movimiento como la actuación y el sonido se grabasen directamente con cámara en mano, igual que se haría en una película. De este modo una única persona realizaría a la vez tres tareas diferentes, ahorrando tiempo y recursos económicos, además de conseguir un movimiento de cámara más orgánico y próximo al lenguaje cinematográfico.

Para ello, no solo debían resolver cada una de estas cuestiones por separado, también debían encontrar una estrategia económica para aunarlas y tener un sistema de ***performance capture*** funcional y con calidad de producción. Así pues, se estudiaron los siguientes ámbitos:

Set* y sistema de captura:** una de las cuestiones que era importante solventar con mayor presteza fue cómo abordar el proceso de captura de movimiento. El ***Hardware necesario es relativamente caro, y preparar un ***set*** de captura de movimiento profesional implica una inversión considerable tanto de tiempo como de dinero. Una de las opciones a la que recurren la mayoría de los grandes estudios es que tanto actores como parte del equipo viajen a instalaciones especializadas

11 . En producción cinematográfica, ***stand-in*** es la persona que sustituye al actor para configurar aspectos técnicos como la iluminación o la calibración del equipo antes de grabar.

12 . ***The Game Awards*** son los premios anuales otorgados a los mejores videojuegos de cada categoría.

fig. 31 Personaje 3D de Senua. Fuente: hellblade.com

fig. 32 Referencia fotográfica de Melina Juergens (derecha) caracterizada como Senua en su modelo 3D (izquierda). Fuente: hellblade.com

que se encargan de realizar y supervisar todo el proceso. Pero Ninja Theory no podía optar por esta vía, pues debía ceñirse al presupuesto estipulado, por lo que su solución fue mucho más casera. Acondicionaron una sala de reuniones de su oficina a modo de *set* para grabar todas las escenas. Utilizaron mobiliario de IKEA y luces LED compradas en Amazon para crear las condiciones lumínicas que permitiesen capturar en tiempo real las expresiones faciales, y se cubrió el suelo con esterillas de gimnasio para grabar cómodamente las escenas. La sala estaba lejos de presentar un aspecto profesional, pero cumplía su función. Cómo afirmaba Dominic Mathews, director comercial en Ninja Theory, durante su conferencia en el GDC de 2017 (Reynolds, 2018):

> *Our attitude was - let's give it a go. If it doesn't pay off, we'll try out something else.*

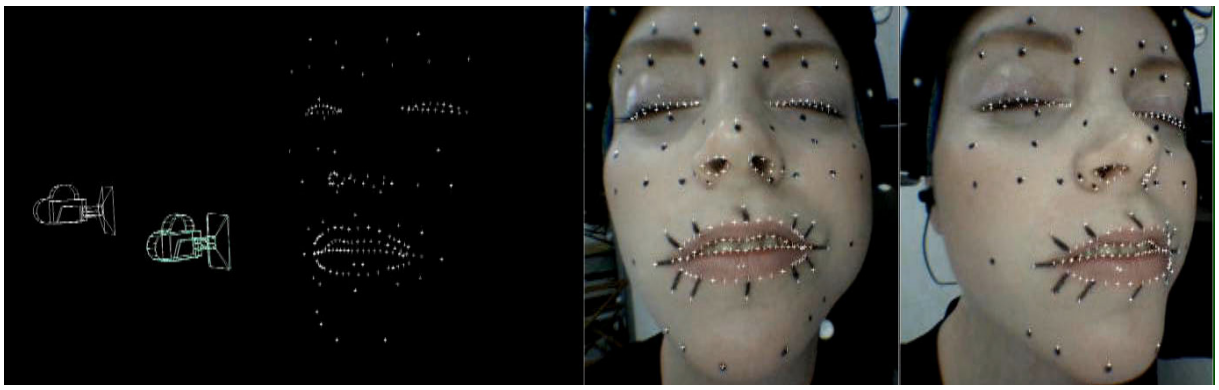
Esta táctica de prueba y error ha vertebrado el desarrollo de ***Hellblade: Senuas's Sacrifice*** de principio a fin. Por ejemplo, durante la fase de prototipado, se diseñó un casco de críquet con una ***webcam*** situada en la parte frontal del casco mediante un bastidor de metal para grabar las expresiones faciales, artilugio que finalmente no se utilizó debido a su escasa robustez. Para capturar la actuación de Melina no hubo más opción que utilizar un traje de captura de movimiento (***mocap suit***) y un sistema óptico de *tracking*. Se llegó a un acuerdo con la empresa Vicon, quien les alquiló 8 cámaras infrarrojas y uno de sus cascos (***Vicon Cara Head Rig***) [fig. 35] preparado para captura de movimiento facial.

Fotogrametría y escaneado: la manera más práctica de conseguir los modelos y el texturizado de una persona es recurrir a estas dos técnicas que permiten obtener fotografías con un alto nivel de resolución para ser utilizadas como texturas y un modelo 3D basado en el escaneo del cuerpo real del actor, respectivamente. Se realizaron varias pruebas con miembros del equipo, escaneando su cuerpo y su rostro para probar los resultados obtenidos con un escáner 3D de mano.

Como primera tentativa para escanear los detalles del rostro, se construyó un artilugio que el equipo bautizó como "prototipo maceta", porque consistía en una cámara, controlada con una ***raspberry Pi***, dentro de una maceta con varias luces LED en su interior para así capturar los detalles de la piel en diferentes condiciones de iluminación. Este sistema se ideó para recabar mapas de normales y mapas de relieve con una alta resolución y un gran nivel de detalle.

El prototipo fue fundamental para tener una idea aproximada de cómo proceder en el texturizado del modelo 3D, pero los resultados no fueron suficientemente convincentes para ser utilizados en producción. Por tanto, varios miembros de Ninja Theory viajaron junto a Melina Juergens rumbo a Serbia, donde la empresa 3Lateral elaboró un escaneado completo de la cara de la actriz.

Camera Rig*:** capturar la interpretación de Melina en tiempo real era un objetivo esencial, pero de nada serviría si su actuación no se acompañaba con un control de cámara cinematográfico. Para ello, se debía *trackear* la posición de la cámara virtual al mismo tiempo que se capturaba el movimiento y la interpretación de la actriz. El objetivo era poder operar una cámara física que emparejase los movimientos de la cámara virtual en ***Unreal.



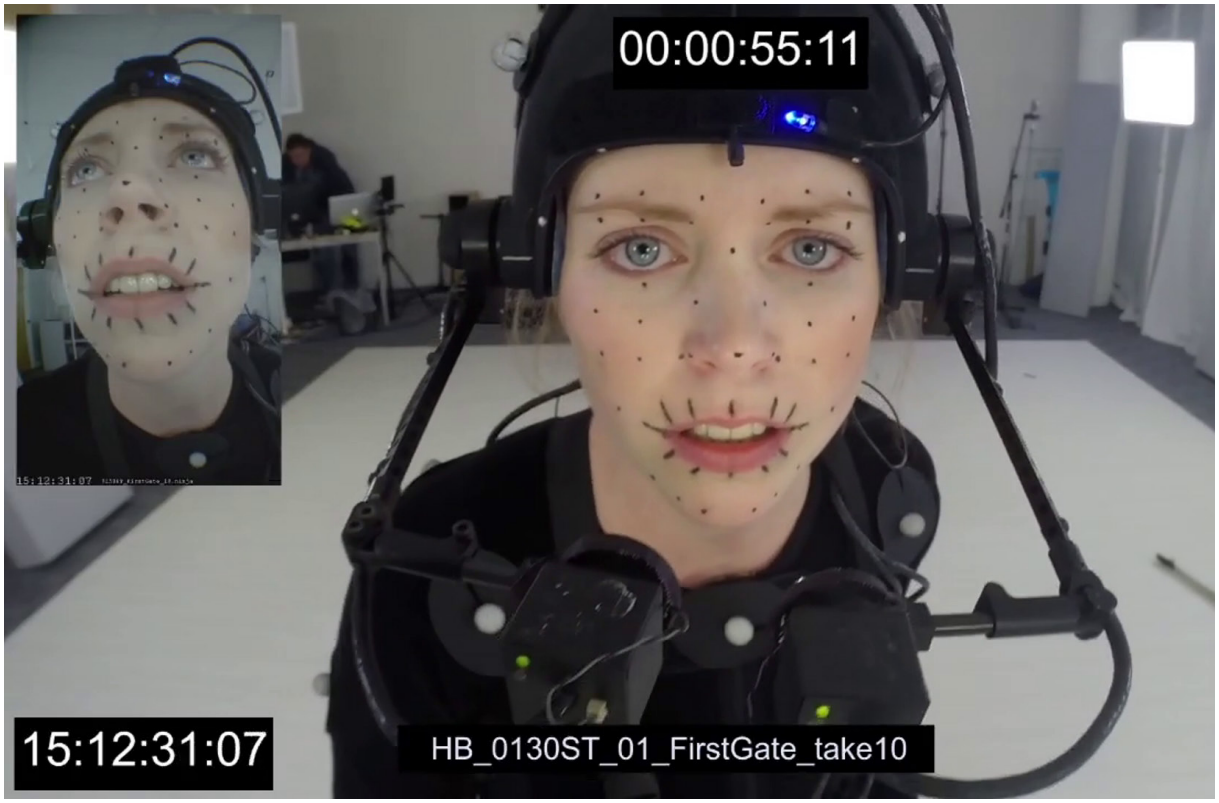


fig. 33 Imagen de marcadores dentro del *software* de captura facial. Fuente: fxguide.com

fig. 34 *Hydra prototype*. Imagen del primer prototipo de *virtual camera rig*. Fuente: hellblade.com

fig. 35 Primer plano del sistema de captura facial *Vicon Cara Head Rig*. Fuente: hellblade.com

Como primer prototipo, y con el fin de resolver todas las tareas con una misma solución, se diseñó un *rig* de cámara que consistía en varias *go-pro* con diferentes lentes [fig. 34]. Un grupo de estas capturaba el movimiento de la cara en 3D, otras el movimiento del cuerpo y, en conjunto, triangulaban la posición en el espacio 3D mediante el *tracking* de marcadores visuales situados en las paredes de la sala, igual que se haría con *After Effects*. El sistema dio sus frutos, pero para que fuese eficiente se dabían invertir demasiado tiempo y recursos, por tanto, se optó por recurrir al sistema óptico de Vicon, mencionado en el párrafo anterior.

El sistema de cámaras infrarrojas resolvía la captura de movimiento, pero aún era necesario registrar los datos de la cámara física operada durante el rodaje. La solución fue emplear un método, ampliamente extendido en los procesos de *previs* [13], que consistía en 3 o 4 marcadores ópticos adheridos al *rig* de la cámara utilizada para grabar a los actores en escena [fig. 36]. De este modo las cámaras de Vicon capturarían también el movimiento de un segundo objeto que *Unreal* podría interpretar como cámara virtual.

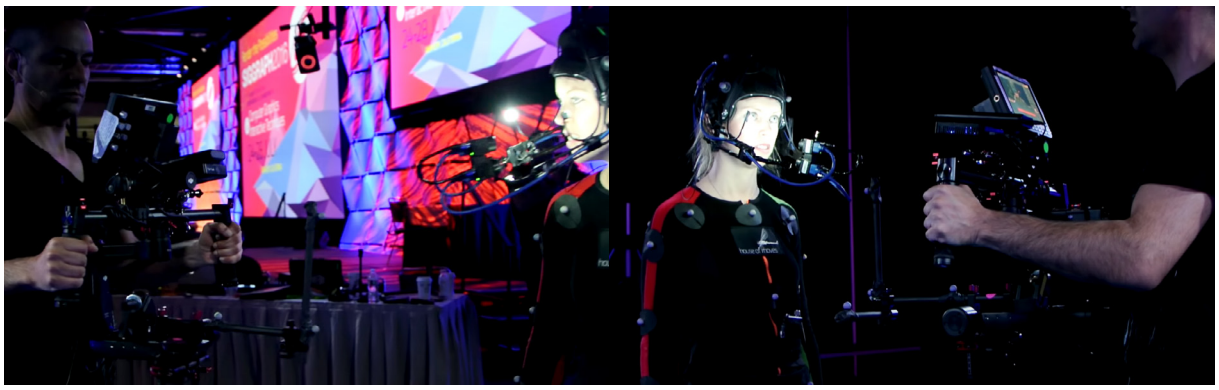
Audio: la experiencia auditiva desempeña una papel clave en la representación de la psicosis que sufre el personaje de Senua. El diseño de sonido debía estar muy cuidado si se pretendía conseguir una buena experiencia inmersiva. Esta sensación de espacialidad en el diseño de sonido se logró capturando la voz en estéro directamente desde la actuación en plató, a través de micrófonos binaurales [fig. 37]. Este tipo de micros emula las funciones que desempeña el oído humano, asignando los sonidos grabados en un espacio a los canales izquierdo o derecho en función de la dirección de la que procede la onda. Esta experiencia auditiva solo se puede experimentar si se utilizan auriculares estéro, pero con un buen diseño se pueden conseguir efectos muy logrados, tales como voces hablándote desde la lejanía mientras otras suenan dentro de tu cabeza simultáneamente.

Hellblade: Senua's sacrifice fue un éxito rotundo, recibió muy buenas críticas y fue considerado uno de los mejores videojuegos de 2017, no obstante, en mi opinión, su mayor logro no tiene que ver con el desarrollo técnico o con la tecnología utilizada, pues muchas de estas ya habían sido empleadas con anterioridad en el cine o en otros videojuegos. El verdadero hito de Ninja Theory fue demostrar que existía una posibilidad de alcanzar esos resultados con un presupuesto ajustado y un equipo pequeño, y poner de manifiesto que el avance tecnológico había democratizado el uso de técnicas que anteriormente estaban reservadas solo a las grandes compañías, las cuales únicamente centraban sus esfuerzos en producir *blockbusters*.

Además, el planteamiento de rodar un videojuego como si fuese una película, abrió un camino hacia enfoques más orgánicos, que acercaban el mundo del videojuego a artistas audiovisuales acostumbrados a trabajar con equipos de vídeo.

Año tras año vemos cómo la tecnología ha dejado de suponer una barrera técnica para muchos profesionales, haciendo que su trabajo sea más fácil y próximo a las herramientas que están acostumbrados a utilizar.

13 . Abreviación del término inglés *previsualization.* Se refiere al proceso de visualizar escenas y efectos complejos antes de ser rodadas. En VFX, es una manera rudimentaria de visualizar los efectos digitales sin pasar por todo el proceso de finalización y renderizado.



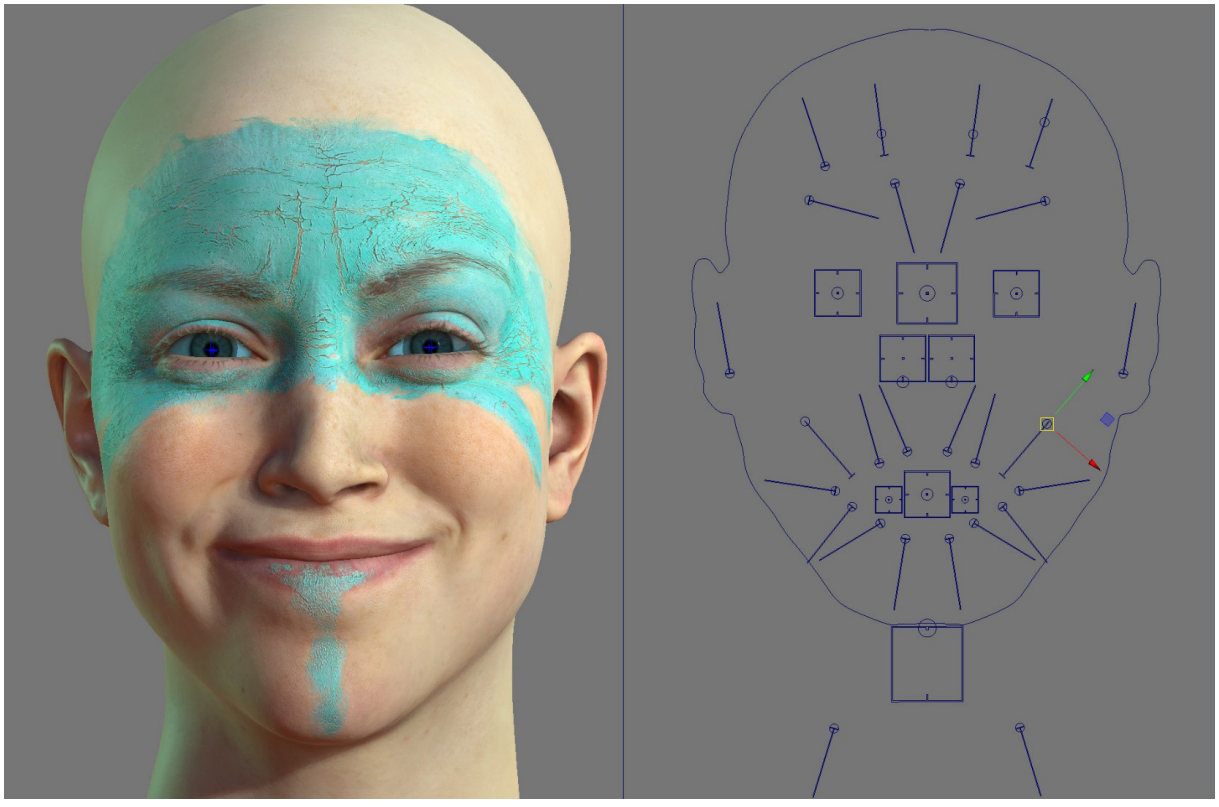


fig. 36 Manejo de cámara en tiempo real durante el SIGGRAPH 2016. Fuente: hellblade.com

fig. 37 Ejemplo de micrófono binaural. Fuente: hellblade.com

fig. 38 Fotoescaneado y *facial rigging*. Fuente: fxguide.com

2.3. Sistemas y tipos de captura

En una sesión de ***motion capture***, un sistema de cámaras y sensores muestrea, con una elevada tasa de fotogramas por segundo, el movimiento de los actores. Una cámara puede ser operada en torno al actor mientras la actuación se está desarrollando y el sistema de captura de movimiento podrá registrar los movimientos de la cámara, así como los ***props*** en la escena. Esto permite que los personajes, entornos y ***sets*** CGI tengan la misma perspectiva y movimiento que las imágenes capturadas por la cámara física, evitando discordancias de composición producidas por el efecto de paralaje.

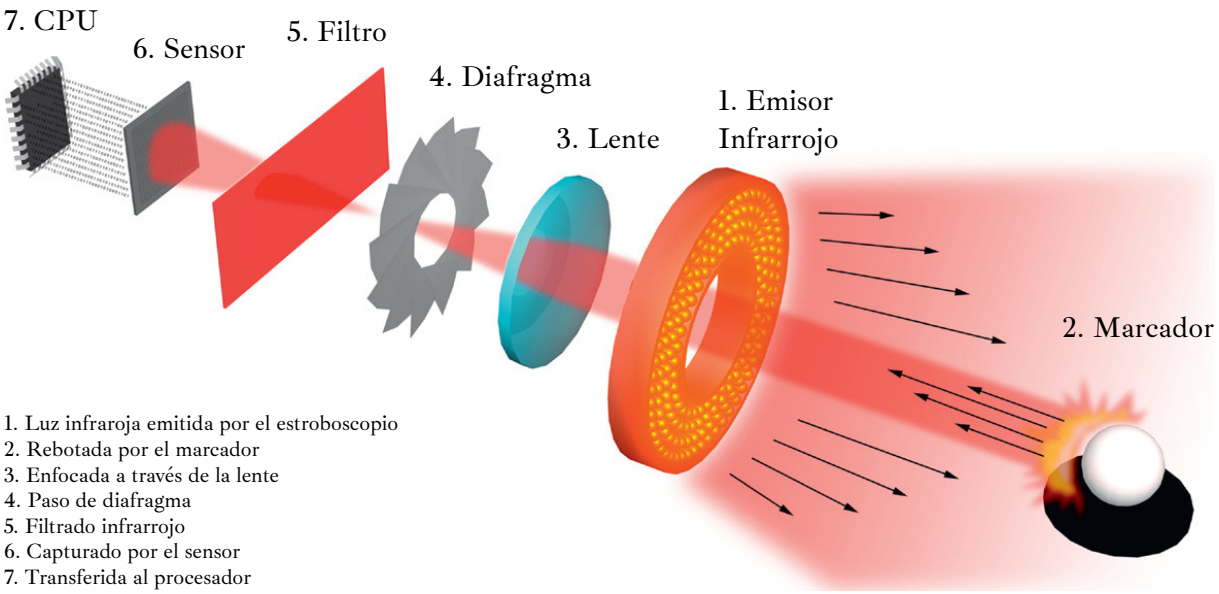
A su vez, todos estos datos son procesados por un ordenador que dispone la posición y el movimiento del actor en un espacio 3D, de acuerdo con la posición tridimensional inferida de cada marcador. No obstante, algunos de los sistemas más antiguos requieren de la acción de un animador para que "limpie" los datos erróneamente interpretados por el sistema, fenómeno conocido como ***marker swapping***, que intercambia la localización tridimensional de dos o más de los marcadores encargados de capturar la actuación del actor. La lectura errónea de un solo sensor puede provocar que el ordenador interprete libremente la posición de este en el espacio, dando lugar a poses no deseadas en un ***frame*** concreto. El ***marker swapping*** se puede reducir drásticamente con la utilización de ***hardware*** más moderno, basado en marcadores activos, pero aún con todo, es necesario que el ***software*** rellene los momentos en los que, por fracciones de segundo, el marcador desaparece de plano, ya que todos los sistemas ópticos sufren de oclusión.

Tras el procesado, los datos de animación se importan a un programa de animación 3D (Maya, Blender, 3DSmax, C4D etc.), donde los animadores pueden vincularlos al ***rigging*** de los personajes 3D. Si la captura de movimiento es lo suficientemente precisa, los tiempos en el proceso de animación 3D se reducen drásticamente, ya que se limita a integrar el modelo animado en la escena y a corregir las posibles interacciones con otros elementos 3D.

Existen multitud de sistemas diferentes para proceder en captura de movimiento, desde los más tradicionales, que se valen de marcadores ópticos para calcular la posición, hasta los sistemas basados en sensores inerciales de posición. A continuación, exploraremos algunos de ellos.

2.3.1. Sistemas ópticos

Estos sistemas basan su funcionamiento en los elementos ópticos propios de cámaras y sensores de imagen, triangulan la posición 3D de un sujeto en función de unos marcadores adheridos al cuerpo del actor [fig. 43] en zonas relevantes, como articulaciones, pies, manos o partes de la cabeza. El actor realiza la acción en un espacio específicamente preparado, delimitado por varias cámaras que están calibradas para proporcionar proyecciones superpuestas de la escena, y que registran el movimiento [fig. 39]. Al mismo tiempo, los datos de triangulación se registran en un equipo externo para, posteriormente, ser utilizados en los modelos 3D del personaje que se quiera animar. El tamaño del área delimitada varía en función del número de cámaras que se utilicen, así como de la cantidad de marcadores que se pueden triangular a la vez. Estos marcadores pueden ser pasivos (retro reflectantes) o activos (emiten luz).



1. Luz infraroja emitida por el estroboscopio
2. Rebotada por el marcador
3. Enfocada a través de la lente
4. Paso de diafragma
5. Filtrado infrarrojo
6. Capturado por el sensor
7. Transferida al procesador

fig. 39 *Set* completo de captura de movimiento con sistemas ópticos. Fuente: optitrack.com

fig. 40 Funcionamiento esquematizado de un sistema de captura óptico. Fuente: (Okun y Zwerman, 2010. pg. 353)

Marcadores Pasivos: Con marcadores **pasivos** se utilizan LEDs infrarrojos alrededor de las lentes de las cámaras de posición, junto con filtros de paso infrarrojo situados sobre los objetivos; de esta manera, las cámaras de posición miden la luz infrarroja rebotada por los marcadores colocados en el cuerpo del actor [fig. 40]. El umbral de sensibilidad de las cámaras se puede ajustar para capturar únicamente la luz reflejada por los marcadores y discriminar el resto de las luces rebotadas y reflejos presentes en la escena.

Para calibrar las cámaras y la posición dentro del espacio, se utiliza un objeto con marcadores colocados en posiciones conocidas, a modo de referencia. Si dos de estas cámaras calibradas registran uno de los marcadores, se puede obtener su posición tridimensional en un espacio virtual. Estos sistemas constan normalmente de entre 2 a 48 marcadores, aunque 33 marcadores suele ser el mínimo necesario para capturar el movimiento corporal de un ser humano. Añadir cámaras puede ayudar a corregir el *marker swapping*, pues cuanto mayor sea el número de cámaras mayor será el número de ángulos que el sistema será capaz de cubrir.

Marcadores activos: Por otra parte, los marcadores activos, en vez de reflejar la luz generada por fuentes externas, están equipados con LEDs que emiten su propia luz infrarroja pulsada. Esta es recogida por las cámaras, que cumplen la misma función que con los marcadores pasivos.

El proceso de captura en sistemas con marcadores activos se puede ajustar mucho más si se utilizan **marcadores activos modulados por tiempo.** Estos sistemas emiten luz estroboscópica en un marcador cada vez, o modulan la amplitud y frecuencia de onda de múltiples marcadores. El objetivo es dotar a cada marcador de una ID específica que permita al programa de rastreo identificarlos con mayor precisión, sin confundir unos con otros, reduciendo de este modo el *marker swapping* y proporcionando datos más precisos.

Marcadores semipasivos: Estos sistemas operan en sentido inverso a los mencionados previamente. En vez de registrar la luz infrarroja reflejada, o emitida, por los marcadores, el sistema utiliza un tipo de sensor fotosensible que recogen y decodifican las señales ópticas recibidas

2.3.2. Sistemas no ópticos

Son sistemas cuyo método de captura no se basa en el muestreo de luz a través de cámaras de posición, utilizan otro tipo de sensores basados en propiedades físicas como la aceleración y el ángulo de rotación para registrar los movimientos del actor.

A diferencia de los sistemas ópticos, ninguno de estos sistemas es capaz de capturar las expresiones faciales de un actor, para este tipo de tareas se utilizan las técnicas previamente mencionados.

Sensores de movimiento inercial: en vez de utilizar señales lumínicas para muestrear la posición de marcadores en cada *frame*, los sistemas inerciales capturan el movimiento del actor en función de una serie de propiedades físicas inherentes al movimiento realizado por el cuerpo.

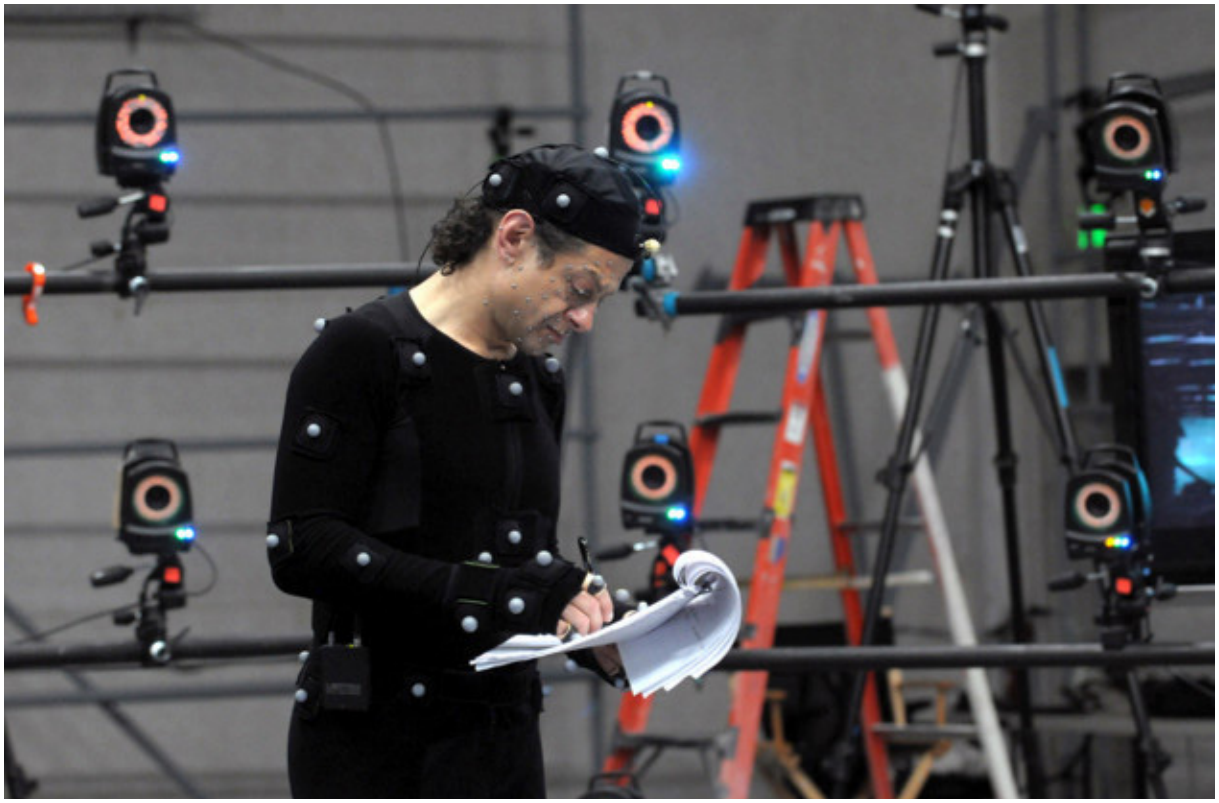


fig. 41 Cámara y marcadores de un sistema óptico. Fuente: optitrack.com

fig. 42 Montaje de cámaras sobre esctructura delimitadora. Fuente: optitrack.com

fig. 43 Andy Serkis vistiendo un traje de captura de movimiento. Fuente: cgicoffe.com

Igual que un móvil cuando se utiliza la aplicación de GPS, estos sistemas se valen de pequeños sensores, como giroscopios y acelerómetros, para medir las propiedades físicas de la acción realizada por el actor (Root y Gordon, en Okun y Zwerman, 2010. pág. 336).

Los giroscopios son capaces de calcular la velocidad angular y proporcionan el cambio de ángulo con respecto de una posición inicialmente conocida. Los acelerómetros miden la aceleración lineal, incluida la gravitacional. Con estos dos sensores se puede determinar tanto la velocidad como la posición de las partes del cuerpo a las que están adheridos.

Los datos se transmiten a un ordenador mediante comunicación inalámbrica, que determina no solo la posición del sensor, también su ángulo de inclinación.

Estos sensores proporcionan datos relacionados con el cambio de estado, pero no son capaces de transmitir su posición tridimensional, por lo que han de ser complementados con un sistema secundario adicional, óptico o magnético, para determinar la posición del actor.

Sensores de movimiento mecánico: los sistemas mecánicos rastrean y detectan directamente la flexión y el ángulo de las articulaciones. El actor viste un exoesqueleto con sensores en las articulaciones y, según el movimiento realizado, el exoesqueleto repite todos los movimientos que realiza el actor y los transmite a un ordenador que se encarga de registrar los datos mediante *software*.

Los sistemas mecánicos son muy aparatosos y pueden romperse fácilmente si no se operan con cuidado.

Sensores magnéticos: siguen un principio parecido al de los sistemas ópticos, pero en este caso los marcadores son imanes, y una serie de receptores magnéticos cumplen la función de las cámaras de posición en un sistema óptico. El sistema rastrea la posición en función de las distorsiones generadas por los imanes en el flujo del campo magnético. Al igual que los sistemas inerciales, los sistemas magnéticos utilizan sensores secundarios para determinar la posición y la rotación de las articulaciones correspondientes.

La zona de captura en sistemas magnéticos es significativamente menor que en aquellos con sensores ópticos.

2.3.3. Facial motion capture

Adicionalmente, para completar las rutinas de *performance capture*, existen técnicas para recoger el movimiento facial de los actores mientras estos actúan ante la cámara. Básicamente, consiste en colocar marcas sobre puntos estratégicos de la cara, (pliegues alrededor de la boca, pómulos, párpados, etc.), cualquier zona que sea susceptible de moverse mientras se habla o se gesticula. Un conjunto de cámaras, situadas a un palmo de distancia de la cara del actor, graba sus gestos, y un software interpreta cómo se ha de de desplazar la malla en función del movimiento que realizan los marcadores sobre la cara del actor.

Los procesos de captura facial son llevados a cabo necesariamente por sistemas ópticos, pues los marcadores inerciales y magnéticos no poseen la sensibilidad suficiente para registrar movimientos tan sutiles.

3. Tercera parte: *Virtual Production*

La técnica de producción virtual persigue los mismos objetivos que han determinado el desarrollo del resto de técnicas y tecnologías presentadas con anterioridad a lo largo de este TFM: poder agilizar y facilitar la integración de CGI al máximo posible, así como, reducir el tiempo invertido en posproducción. Por otra parte, uno de los mayores deseos de todo artista VFX es poder visualizar en tiempo real, y con un buen acabado final, las escenas y efectos creados en el estudio, por tanto la motivación para explorar soluciones en tiempo real está ligada también al objetivo de contribuir al desarrollo artístico a través de una visión holística que cohesione todas las fases de desarrollo. Así pues, la producción virtual da respuesta a ambos aspectos, procura reducir el tiempo en posproducción y facilita herramientas con las que tanto artistas como diseñadores y creativos audiovisuales están familiarizados.

Hasta hace no mucho tiempo, existía una desconexión habitual entre la fase de rodaje y la de integración de efectos especiales por CGI. Las herramientas utilizadas y el tipo de flujo de trabajo establecido hacían necesario dividir la producción de cualquier proyecto audiovisual en diferentes etapas, pues, mientras se siguiesen utilizando motores de renderizado ***offline***, sería imposible incorporar los elementos CGI al *set* de rodaje.

Este flujo de trabajo deriva de un planteamiento establecido durante años, ocasionado por las limitaciones técnicas del ***software*** y el ***hardware*** utilizado en posproducción. Era inevitable relegar la integración de CGI a las fases finales ya que todos los motores de renderizado utilizados en la industria se basaban en la iteración de cálculos para conseguir un único fotograma con calidad suficiente para ser integrado en el metraje final. Como es comprensible, esta metodología es bastante lenta, ya que la calidad del render final está condicionada por la cantidad de iteraciones que el motor realiza para renderizar la escena, cuanto mayor es el número de iteraciones más exactos son los cálculos realizados y mayor calidad presenta el render final; este mismo principio rige para cada uno de los fotogramas que componen una secuencia, por lo cual, el proceso completo de renderizar una toma de pocos segundos implica tiempos considerablemente altos.

Dependiendo de la potencia gráfica del equipo utilizado se puede reducir el tiempo de renderizado, pues cuanto mayor es la potencia gráfica menor es el tiempo de cálculo por iteración. Por tanto, es frecuente recurrir a máquinas diseñadas específicamente para cubrir única y exclusivamente el proceso de render. Una vez la escena está preparada y lista, se utilizan sistemas de ordenadores en ***cluster***, conocidos como granjas de render, que combinan la potencia de varias tarjetas gráficas para optimizar los tiempos de cálculo. Renderizar en granjas ayuda a reducir el tiempo que tarda el motor en producir una secuencia completa, pero, aun así, el planteamiento básico sigue siendo el mismo. Mientras se mantenga el uso de motores ***offline***, el CGI quedará irremediablemente relegado a la fase de posproducción.

Esta posición en la cola del ***pipeline*** ha propiciado uno de los males endémicos en la industria del VFX. Frecuentemente, cuando los efectos especiales de una película carecen de calidad o presentan problemas visuales, es debido, en la

mayoría de los casos, a la escasez de tiempo destinado. Muchos estudios de VFX, en Hollywood, se han quejado de la presión que las productoras ejercen sobre su trabajo exigiendo resultados cinematográficos en plazos de entrega muy ajustados y con presupuestos escuetos.

Uno de los casos más recientes es la escena de lucha al final del filme ***Black Panther*** (Coogler, 2018). En la totalidad del proyecto trabajaron 11 estudios diferentes, pero en esta escena, concretamente, los responsables fueron los miembros del equipo de DNEG. Este estudio londinense, con 8 sedes alrededor del mundo, posee una amplia experiencia en producción de efectos especiales para Hollywood y es ganador de cinco Oscar, cinco BAFTA y varios VES a la mejor producción de efectos especiales por proyectos tan destacables como ***Blade Runner*** 2049 (Villeneuve, 2017), ***Ex Machina*** (Garland, 2014) o ***Chernobyl*** (Mazin, 2019).

Lejos de querer ensalzar el trabajo de DNEG con este elenco de logros, lo que pretendo es remarcar su profesionalidad, que no son aficionados. Sus equipos están integrados por artistas profesionales capaces de producir VFX de calidad. Entonces ¿por qué el resultado de esta escena en particular es tan terrible?

Tras finalizar el rodaje de ***Black Panther***, por complicaciones en la producción, el equipo de DNEG acabó recibiendo el material grabado con dos meses de retraso, lo cual precipitó el plan de producción, ocasionando que los efectos especiales de toda la escena tuviesen que ser resueltos en tan solo 6 semanas. El equipo de artistas que trabajó en la secuencia tuvo que trabajar a marchas forzadas, y con el mismo presupuesto, para cumplir con las fechas marcadas por Disney.

El ejemplo de ***Black Panther*** no es un caso aislado, pues son muchos los estudios que han sufrido plazos ajustados, con poco tiempo para alcanzar la calidad visual que a ellos les hubiera gustado ofrecer. Todavía más drástica fue la situación del estudio ***Rhythm & Hues***, quienes se declararon en bancarrota solo dos semanas antes de ganar el Oscar a mejores efectos especiales por ***La vida de Pi*** (Lee, 2013). Durante la gala, Bill Westenhofer, quien fue responsable de coordinar los efectos especiales, intentaba denunciar la situación desde el podio, tras recoger el premio, cuando la organización del evento elevó el volumen de la música y silenció su micrófono, en directo, para acallar su discurso.

Es evidente que el afán recaudatorio y la maximización de beneficios por parte de las grandes productoras son los principales causantes de la situación que estudios como ***Rythm & Hues*** sufren dentro de la industria cinematográfica. No obstante, los problemas de plazos ajustados y escaso tiempo de producción también derivan, en parte, de esta estructura en tres fases, en la que el retraso de las dos primeras condiciona drásticamente el tiempo dedicado a posproducción.

Todavía hoy esta metodología de trabajo se mantiene intacta en muchos proyectos, pero con la aparición de nuevas tecnologías y los avances en computación gráfica, algunas de las estrategias clásicas de posproducción se han visto comprometidas, siendo necesaria su actualización a partir de nuevos planteamientos. Es en este punto donde el concepto de producción virtual empieza a cobrar fuerza.

En un principio, la idea de visualizar CGI en tiempo real empezó a investigarse para cubrir los procesos de *previs*. *Avatar* (James Cameron, 2009) fue una de las primeras cintas en utilizar entornos de producción virtual como una parte esencial de su metodología de producción. Se utilizaron técnicas de rastreo para operar cámaras virtuales, mediante interfaces físicas, en el plató de rodaje, mientras los actores actuaban con sistemas de *performance capture*. Pero en 2009 la tecnología no había avanzado lo suficiente para poder ofrecer renders realistas en tiempo real, por lo que el renderizado de las escenas seguía realizándose en posproducción, tras procesar el material obtenido en el *set*.

No obstante, poder plantear los efectos digitales directamente desde el plató fue un idea innovadora y bastante atractiva, a la que más tarde otras producciones decidirían adscribirse. Si bien la imagen final todavía quedaba desconectada de la fase de rodaje, poder operar físicamente en un entorno virtual, con actores interpretando sus personajes digitales en tiempo real, resultó ser un concepto muy potente que resolvía problemas básicos concernientes a la planificación del diseño 3D, o a la toma de decisiones sobre la dirección y la composición del plano. Poder visualizar los entornos digitales en el plató, a través de una cámara, eliminó el distanciamiento entre el mundo de los VFX y el de la acción real, y consiguió que tanto el equipo de rodaje (director, actores, operarios de cámara, etc.) como los artistas VFX colaborasen con mayor sinergia en el desempeño de sus funciones.

Por otro lado, el sector del videojuego llevaba mucha ventaja implementando la captura de movimiento en sus platós. Además, su medio se había desarrollado bajo el uso de motores que renderizaban en tiempo real, por lo que las previsualizaciones 3D en el plató ya estaban asimiladas dentro del *pipeline*.

Desde un punto de vista estrictamente gráfico, el único aspecto que separaba el videojuego del cine era la diferencia en la calidad del renderizado. Los motores de videojuego no eran capaces de alcanzar la calidad requerida para una película, pero esta limitación se ha diluido a lo largo de los años. Cada generación de tarjetas gráficas ha ido acortando cada vez más la distancia que separaba la calidad de un motor *offline* de la de un motor de videojuegos; las nuevas arquitecturas permiten renderizar escenas cada vez más complejas en tiempos más cortos. En 2018 NVIDIA presentó una serie de procesadores gráficos que han cambiado por completo el mercado del videojuego.

La serie RTX de NVIDIA introducía una característica fundamental, su arquitectura permitía aplicar algoritmos de *Ray Tracing* en tiempo real, ofreciendo una calidad próxima al fotorrealismo, pero con motores de videojuegos que renderizaban la escena en pocos milisegundos.

Aprovechando esta funcionalidad, el año pasado se estrenó una serie que puso la producción virtual en el punto de mira de muchos estudios, quienes empezaron a plantearse esta solución como un recurso viable en el desarrollo de sus producciones.

3.3.1. Producción virtual en The Mandalorian

En 2019, Epic Games publicó "The Virtual Production Field Guide", una guía que explica cómo utilizar los recursos de ***Unreal Engine*** para proyectar sobre un *array* de monitores LED entornos CGI renderizados en tiempo real, y cómo hacer un uso efectivo de las herramientas de producción virtual.

Durante el SIGGRAPH de 2019, se presentó un vídeo [14] que demostraba cómo esta técnica podía ser aplicada en cualquier producción cinematográfica. En él, un conjunto de muros formados por varios paneles LED proyectaban, detras de un actor, diferentes entornos digitales, con diferentes condiciones de iluminación, renderizados en tiempo real con ***Unreal Engine***.

Sinceramente, esta idea, como tal, ha estado presente como concepto en diversos medios audiovisuales durante muchos años, incluso décadas. Algunos vídeos musicales de principios de los 2000 ya incorporaban fondos de pantallas LED, y en el filme ***Oblivion*** (Kosinski, 2013) se grabaron algunas escenas haciendo uso de esta técnica. Podríamos incluso remontarnos a los años 50, en los que se empleaban proyectores ópticos para grabar las escenas de conducción. Pero su aplicación no se consideraba un elemento esencial de producción, era más bien un recurso ocasional para solventar ciertos planos. Además, la tecnología en que se basaba todavía no estaba suficientemente depurada y no se había implementado una estrategia de trabajo efectiva para incorporar su uso durante el rodaje. Pero actualmente, gracias a los avances presentados en estos últimos dos años, existe la posibilidad de proyectar entornos 3D y grabarlos, con una cámara, junto con los actores y el resto de los elementos en el *set* simultáneamente, sin necesidad de pasar por procesos de ***chroma keying*** [fig. 44].

Con esta idea en mente, John Favreau, quien ya había participado previamente en proyectos en los que se aplicaban estrategias de producción virtual como ***El rey león*** (Jon Favreau, 2019) o ***El libro de la selva*** (Jon Favreau, 2016), decidió utilizar un *set* con muros de pantallas LED para resolver gran parte de los VFX de la serie que estaba produciendo en esos momentos: ***The Mandalorian*** (Jon Favreau, 2019). Fue una decisión crucial que condicionó la producción desde el minuto uno, y que permitió demostrar cómo esta metodología era una opción más que viable para resolver gran parte de los VFX en una superproducción de Hollywood.

The Mandalorian, lejos de ser una producción tradicional, combinó diferentes técnicas para crear los efectos especiales ante la cámara. De hecho, lo realmente innovador de su planteamiento fue recurrir tanto a procesos prácticos como a esta metodología en ciernes, para conseguir una fotografía muy particular que recuerda a una mezcla entre las primeras películas de George Lucas y los ***westerns*** tradicionales.

¿En qué consistió exactamente la producción virtual de ***The Mandalorian***? Se construyó un *set* virtual [fig. 45] [fig. 46] [fig. 47], un plató específicamente pensado para proyectar entornos 3D en las paredes y el techo, con un sistema óptico de captura de movimiento que ***trackeaba*** unos marcadores situados sobre la cámara física (del mismo modo cómo se hizo en la producción de Hellblade: Senua's

14 . ***In-Camera VFX with Unreal***. Enlace al vídeo de presentación del SIGGRAPH 2019

fig. 44 Fotografía del plató de *Avengers: Endgame*. Fuente: imdb.com

fig. 45 Fotografía del *virtual set* en *The Mandalorian*. El muro circular proyecta el entorno renderizado en tiempo real. Fuente: dailybulletin.com

fig. 46 Escenas del *making of* en el *virtual set* con insertos del plano *in-camera* (recuadros abajo a la derecha). Fuente: fxguide.com, indiwire.com.

fig. 47 Escenas del *making of* en el *virtual set.*

Sacrifice). El set tenía forma de cúpula, cuyos muros y techo estaban construidos con paneles LED; el suelo se utilizó para distribuir los elementos escenográficos como se haría en un plató convencional, mientras que las paredes y el techo proyectaban en tiempo real, y sincronizados con el movimiento de la cámara, los entornos diseñados por el equipo de VFX en ***Unreal Engine***. De este modo, cuando la cámara grababa la acción de los actores, también capturaba los entornos virtuales proyectados por los muros y el techo.

Desde un punto de vista técnico, utilizar LEDs como fondos en la mayoría de los planos permitió perfeccionar aspectos que con un *green screen* habrían supuesto una mayor inversión de tiempo y que, en ciertos casos, no habrían dado el mismo resultado.

Una de las ventajas más evidentes que ofrece un *set* virtual es la **contribución a la iluminación ambiental**. Las pantallas, al emitir luz propia, construyen unas condiciones de iluminación realista sobre las superficies del *set* y sobre los actores, sin la necesidad de añadir fuentes de luz adicionales (focos, paneles, *spots*), al contrario de cómo se debería hacer en un *set* con *green screen*. Por otro lado, desde un punto de vista gráfico, cuando se tiene que integrar un fondo por *chroma* en posproducción, es posible que la iluminación sobre los elementos del metraje no sea del todo realista, pues, si no se ha iluminado correctamente a los actores, las condiciones de iluminación han de ser recreadas en posproducción.

Desde una perspectiva fotográfica, poder visualizar los entornos directamente sobre el *set* es una gran ventaja tanto para los directores de fotografía como para los actores. Antes, un director de fotografía debía imaginar cómo quedarían los efectos especiales tras finalizar la posproducción, pero con un *set* virtual se pueden tomar decisiones sobre el terreno de acuerdo con el aspecto real de la toma. Este principio se aplica también a la actuación de los actores, quienes pueden interactuar de un modo más natural en las escenas que están interpretando. Este flujo de trabajo se completa con la presencia de artistas y diseñadores 3D en el plató, quienes pueden modificar partes de la escena virtual de acuerdo con las sugerencias del equipo de fotografía [fig. 49]. En resumen, se establece un flujo de trabajo mucho más orgánico y natural.

Otra ventaja que proporciona la producción virtual es la facilidad para trabajar con **materiales reflectantes** y superficies metálicas, algo bastante tedioso si se usa *green screen*. Cuando se graba una escena en un *set* rodeado de muros verdes, es inevitable que las superficies reflectantes proyecten el verde del entorno; incluso aquellas superficies que no reflejan en absoluto adquieren un tinte verde, producto de la iluminación rebotada, conocido como *green spill* [fig. 48], que ha de ser corregido en posproducción. Esto es muy evidente en *The Mandalorian*, pues su protagonista viste una armadura con piezas de metal cromado que refleja todos los elementos de su alrededor. Si se hubiese utilizado un *green screen*, probablemente se hubiese optado por recrear partes del personaje con CGI (como ya se hizo con el traje de *Iron Man* en la saga *Vengadores*), ya que eliminar los reflejos verdes sobre la armadura habría sido demasiado laborioso, por no decir prácticamente imposible. Sin embargo, este problema desaparece en un *set* virtual porque los reflejos obtenidos en este tipo de superficie son los del propio entorno digital proyectado por los LEDs.

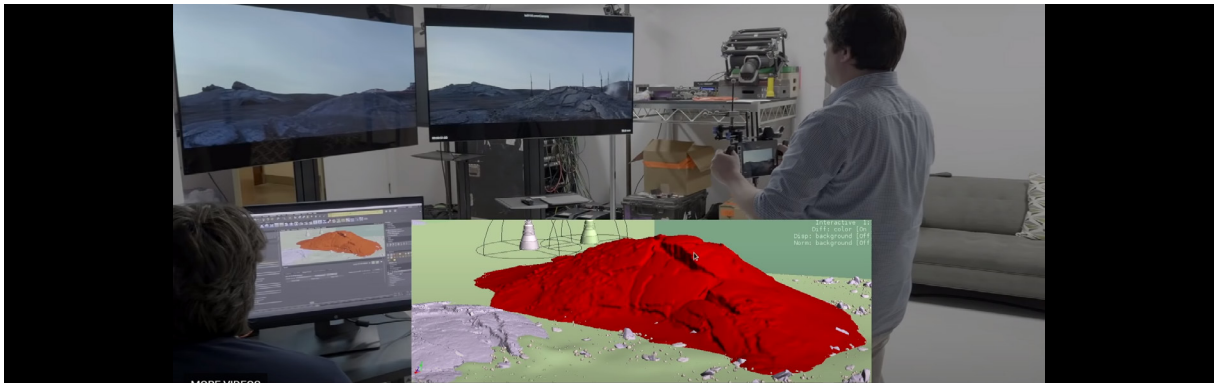


fig. 48 Ejemplos de *green spill*. Fuente (Barroso, 2016).

fig. 49 Modificación y visualización en tiempo real del entorno virtual dentro de *Unreal*. Fuente: ILMVFX

fig. 50 Uso de *green screen* en *The Mandalorian*. Fuente Pixomondo

No obstante, los procesos de *chroma* todavía son necesarios en algunos casos. En esta serie también se utilizó *green screen* para situaciones puntuales que requerían eliminar algún elemento concreto de la escena (como en esta secuencia donde los personajes montan unas criaturas recreadas por CGI [fig. 50]). Pero, la principal diferencia respecto de un plató tradicional es el modo en cómo se utiliza. En un *set* virtual se puede proyectar directamente una porción de fondo verde sobre las pantallas [fig. 50], sin necesidad de modificar la escenografía y conservando parte de la iluminación proyectada por el resto de los paneles LED. De esta manera, se reduce el *green spill* sobre las superficies afectadas por el *chroma*.

Sin duda, ***The Mandalorian*** ha marcado una nueva ruta en la creación de contenido audiovisual, tanto en proyectos para cine como para televisión. Esta nueva metodología de producción se ha vuelto extremadamente popular debido a las ventajas técnicas frente al *green screen*, y al ahorro de tiempo y de dinero.

Pero, en mi opinión, lo realmente novedoso de la producción virtual no son las ventajas técnicas que pueda ofrecer frente a otros métodos de trabajo tradicional. Lo verdaderamente puntero es la capacidad que posee para incorporar parte de las tareas de posproducción ya en la fase de rodaje. Desde un punto de vista creativo, es mucho más eficiente trabajar mano a mano con los artistas encargados de construir el CGI, en el mismo momento de rodar la acción. Si se realiza un buen plan de producción, todas las partes que intervienen en el rodaje se pueden ver beneficiadas por el trabajo de los otros artistas. El *feedback* que un director de fotografía pueda aportar es crucial para que los diseñadores de escenarios 3D puedan construir un entorno virtual consistente, y a los actores les resultará más fácil meterse en el papel de su personaje si pueden actuar frente a la imagen de un paisaje en vez de ante un muro de color verde.

De manera casi involuntaria, esta tecnología ha aportado un nivel de sinergia mayor entre dos etapas que tradicionalmente han estado desvinculadas la una de la otra.

La producción virtual no es solo un avance técnico, es un salto conceptual en la forma de hacer cine. Significa cambiar el punto de vista establecido durante décadas para poder cooperar de manera más eficiente en la creación de historias y relatos. Rompe la barrera entre los efectos prácticos y el CGI para incorporar una combinación de ambas técnicas que, lejos de resultar torpe o extraña, resuelve varios inconvenientes típicos de la integración por clave de color.

3.1. Unreal Engine

Durante años, la industria cinematográfica ha confiado en los motores de pre-renderizado para crear los efectos especiales de innumerables películas. Aplicaciones como Cinema 4D, Maya o 3DSMax han sido pilares indiscutibles durante mucho tiempo, lo cual tenía sentido considerando el tipo de flujo de trabajo que se había seguido hasta la fecha.

El procedimiento habitual era recibir el material grabado en plató sobre croma e integrar el CGI mediante un proceso de *chroma keying*, por tanto, era completamente irrelevante si en la renderización se invertían varias horas, o incluso días, pues la acción de los actores ya estaba grabada y el proceso de render se desarrollaba después de diseñar los modelos 3D, las animaciones y el *tracking* de cámara.

Sin embargo, a medida que la técnica de producción virtual se ha ido afianzando, ha comenzado a introducirse una nueva dinámica de producción que implementa la integración de CGI en tiempo real durante la fase de rodaje. Esta tendencia surgió a raíz de proyectos que creaban gran parte del material audiovisual basándose en los gráficos generados por ordenador. En 2009, durante la producción de ***Avatar***, James Cameron demostró que se podía orientar el filmado de una película hacia entornos virtuales que no presentasen imagen de acción real grabada.

Es posible utilizar motores de render ***offline*** para integrar el CGI a través de un proceso de composición mediante extracción de clave de color, ya que todos los efectos pueden ser retocados en posproducción, sin afectar al material grabado. Pero esta posibilidad no es factible si se busca una integración en tiempo real durante el rodaje en plató.

Sin embargo, la perspectiva de trabajo cambia drásticamente si se trabaja en un ***set*** donde se utilizan pantallas LED para proyectar escenarios virtuales y efectos de fondo, todos los VFX mostrados en los paneles deben tener calidad de producto final, pues estarán directamente mezclados con el metraje grabado por la cámara. Incluso en aquellos ***sets*** donde el planteamiento es con ***green screen*** o ***blue screen,*** a pesar de poder mantener la estructura tradicional de producción, se ha comenzado a utilizar motores de render en tiempo real para cumplir las tareas de ***previs*** y poder visualizar en tiempo real cual será el resultado final de los efectos digitales.

Así pues, ante esta nueva perspectiva, se debía encontrar una herramienta que permitiese construir escenarios virtuales, simular físicas e integrar animaciones, todo ello en tiempo real y con una calidad fotorrealista. Por suerte, la industria del cine siempre ha seguido muy de cerca los avances tecnológicos que el mundo del videojuego ha ofrecido en cada nueva generación. Y así, de un modo inequívocamente lógico, los videojuegos ofrecieron la respuesta a las demandas de la industria cinematográfica y de los efectos especiales.

Desde la aparición en los 90 del primer videojuego 3D, la computación gráfica se ha dividido en dos vertientes bien definidas: por un lado, se han desarrollado algoritmos de ***Ray tracing*** para dar salida a la demanda, cada vez mayor, de renders fotorrealistas para la industria del cine, y, por otra parte, en el mundo del videojuego, se han realizado grandes esfuerzos para ofrecer motores de rasterizado capaces de renderizar en tiempo real imágenes cada vez más próximas a la realidad. Las tecnologías no podían ser más diferentes, pero el objetivo era común para ambas, alcanzar una calidad gráfica próxima a la de una cámara de vídeo.

Con el paso de los años, ambos mundos han ido presentando avances, los motores ***offline*** se han sofisticado cada vez más, con algoritmos de ***Ray tracing*** capaces de simular casi por completo las propiedades físicas de la luz, y los motores ***online*** han alcanzado tasas de computación ***frame*** a ***frame*** que permiten cargar modelos casi fotorrealistas. Ambos caminos han ido convergiendo poco a poco hasta llegar al punto en el que hoy nos encontramos, y es que, en 2018, NVIDIA presentó su nueva gama de tarjetas gráficas RTX, las primeras capaces de aplicar ***Ray tracing*** en tiempo real mediante un proceso que combina el uso de diferentes algoritmos de renderizado (***Scanline, Ray tracing, Radiosity,*** etc.), llamado ***Hybrid rendering***.

A partir de este momento, el concepto de motor de videojuego cambió por completo, ya no tenía sentido separar ambos mundos. Los motores *offline* habían estado muy por detrás en cuestión de acabado y calidad final, por cuestiones obvias no eran capaces de alcanzar el fotorrealismo que otros motores de render, basados en *Ray tracing,* eran capaces de ofrecer. Pero con estas nuevas GPUs, era cuestión de tiempo que las empresas de desarrollo de videojuegos empezasen a trabajar en un motor que se beneficiase de esta tecnología, y así fue.

En 2019, durante la presentación de *Unreal* en el GDC [15], Epic Games expuso cómo su motor podía ser utilizado en metodologías de producción virtual, gracias a las operaciones de *Ray tracing* en tiempo real. Beneficiándose de las nuevas tarjetas de NVIDIA, el motor es capaz de ejecutar tareas específicas que resuelven, mediante algoritmos de *Ray tracing,* algunas de las funciones que realiza para renderizar la imagen.

Unreal Engine se ha ganado a pulso la categoría de herramienta esencial en todas las producciones que pretendan un enfoque de producción virtual. Desde que se introdujo la posibilidad de calcular iluminación realista en tiempo real, el objetivo de Epic Gamesha sido convertir su motor en la elección lógica para los productores que quieran emprender proyectos que cuenten con un *set* virtual.

3.1.1. Características y cuestiones de optimización

La principal ventaja de *Unreal* frente a otros motores de render es, por supuesto, poder operar en tiempo real un entorno completamente virtual. Pero, esto tiene serias limitaciones si se quiere alcanzar un apartado gráfico equiparable a los ofrecidos por otros motores *offline*.

Existe un factor fundamental que condiciona el uso de un motor *online*: el tiempo de procesado que la tarjeta gráfica emplea para renderizar la escena. Como ya se ha visto en párrafos anteriores, los motores de render tradicionalmente utilizados en el diseño de VFX (VRay, MantaRay, Octane, Arnold, etc.) confían la calidad y la exactitud de los cálculos al tiempo empleado para renderizar. Mediante la iteración de cálculos en cada uno de los *frames,* el programa es capaz de generar un render por cada fotograma, a partir de las condiciones de iluminación y de los modelos en la escena 3D. Por tanto, en todo este proceso existe un factor que determina rigurosamente la calidad del render final, el tiempo. Cuanto mayor es el número de iteraciones por fotograma, más tiempo emplea el motor en renderizar la escena. Pero este principio no es aplicable en *Unreal*.

Unreal Engine se diseñó para cubrir las necesidades propias del desarrollo de videojuegos, por tanto, sus funciones y características difieren de las de un motor *Ray tracing* convencional. Todo el proceso de renderizado transcurre en un margen de entre 15 y 20 milisegundos, y para una correcta visualización se debe alcanzar al menos una tasa de 25 fotogramas por segundo. Emplear más tiempo para obtener mejores resultados de renderizado es, simplemente, imposible si se busca un rendimiento en tiempo real que no afecte a la "jugabilidad". Por este motivo, la optimización es un concepto esencial que condiciona el uso de un motor en tiempo real. Puesto que la prioridad del motor es renderizar cada fotograma en

15 . Siglas en inglés de *Game Developers Conference*. Es una convención anual en la que se presentan los avances alcanzados en la industria del videojuego.

menos de 20 milisegundos (lo que se traduce en el estándar de 60 fotogramas por segundo), se ha de decidir meticulosamente qué procesos han de ser calculados con mayor prioridad y cuáles otros pueden ser relegados a los últimos puestos en cola.

Comparado con un motor *offline*, que simplemente depende de la fuerza bruta y del tiempo empleado para proporcionar imágenes realistas, un motor en tiempo real necesita decisiones conscientes del diseñador para poder rendir de un modo óptimo. Por tanto, si se pretende un uso eficiente, es necesario considerar algunas características específicas que no encontramos en otros motores *offline*.

En aras de poder operar correctamente un motor como *Unreal*, existen ciertas cuestiones técnicas que es necesario dominar con el fin de optimizar las funciones ejecutadas por el software.

Optimización de modelos

Una de ellas es la **complejidad de geometría.** Este aspecto es común a ambos tipos de motor, pero por la relevancia del factor tiempo, se ha de tener en consideración cuando se trabaja con motores en tiempo real.

Existe una relación directa entre el esfuerzo computacional y la densidad poligonal en la geometría de una malla. Cuanto mayor es el número de polígonos mayor es el esfuerzo que ha de realizar la GPU, pues mayor es también la complejidad del cálculo de rayos rebotados y del número de vértices que se han de rasterizar. Por tanto, es importante mantener un número de *tris* relativamente bajo en los modelos 3D que haya en la escena.

En ciertas ocasiones, la complejidad de geometría no está del todo relacionada con el número de polígonos. La densidad de malla es, sin duda, un factor que influye en el rendimiento, pero puede suceder que modelos con baja geometría presenten errores de iluminación debido a la forma básica del propio volumen. Si el modelo tiene zonas muy ocluidas o volúmenes convexos con formas complejas, esto puede derivar en una mala iluminación o en un rendimiento poco optimizado.

En estos casos, por ejemplo, para resolver detalles con mucha oclusión, como grietas o molduras, se recurre al mapeado de normales para que estas generen la sensación de volumen y detalle sobre una superficie plana.

Los **LODs,** de las siglas en ingles *Level of Detail*, son un sistema común en todos los motores de videojuegos que permiten seleccionar mallas con diferentes niveles de geometría para un mismo modelo. Es una estrategia que gestiona la densidad de malla en función de la cercanía de la cámara virtual al modelo 3D [fig. 51].

El procedimiento habitual, cuando se diseña una pieza para *Unreal*, es partir de un modelado en alta definición, es decir, con un elevado nivel de detalle y con una alta densidad poligonal. Este primer modelo representará el **LOD0**, es decir, el nivel de detalle con máxima definición. Después, se reduce la densidad poligonal de este nivel mediante un proceso llamado “decimado”, que conserva las formas generales, pero reduce la cantidad total de *tris* presentes en el modelo. La malla se “decima” varias veces en porcentajes diferentes para obtener topologías diferentes del mismo modelo, cada una menos densa que la anterior (LOD1, LOD2, LOD3, etc.).

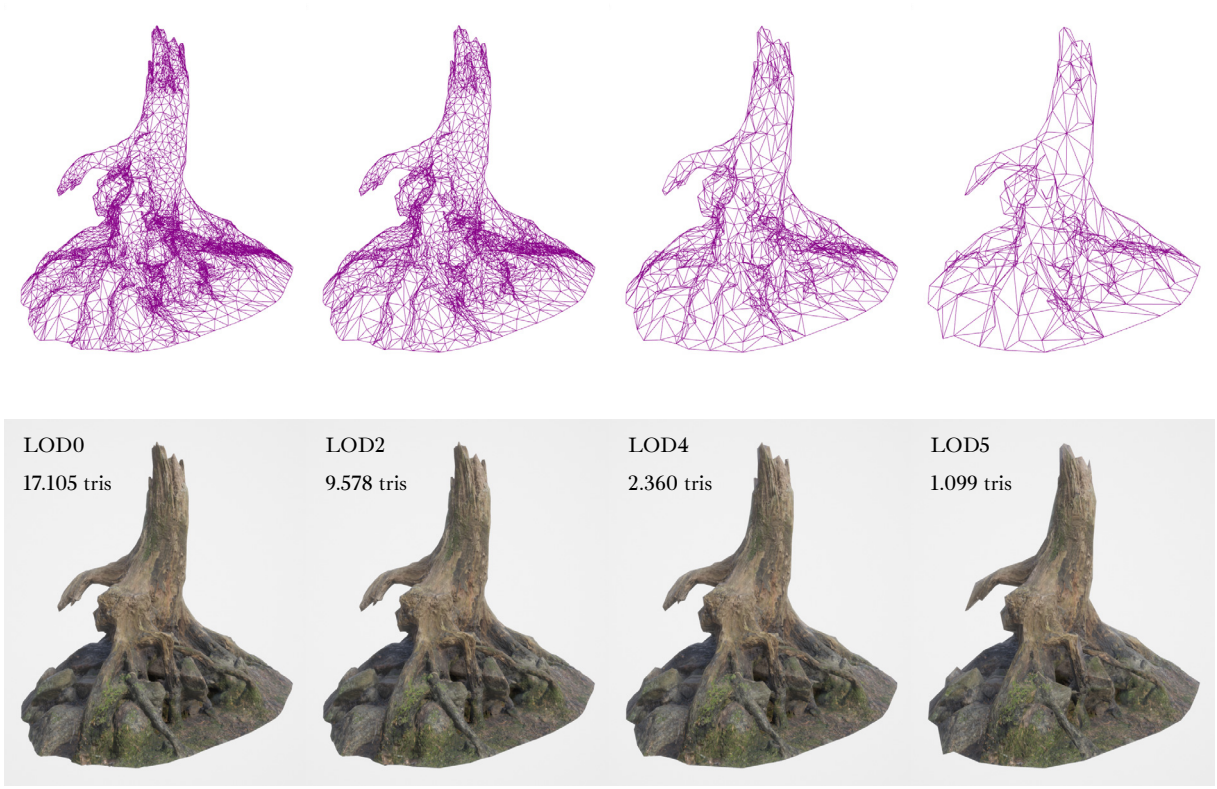


fig. 51 Arriba, visualización de la topología en los diferentes LODs de un mismo modelo. Debajo, Visualización de los LODs del mismo modelo. *Asset* utilizado en la escena construida para este TFM.

fig. 52 Escena diseñada en *Unreal*. En el recuadro abajo a la derecha, modo de visualización “complejidad de sombreadores”.

Cada una de estas mallas se utiliza para representar el mismo modelo a diferentes distancias de visualización. La distancia de visualización se expresa en el porcentaje de pantalla que ocupa el renderizado de dicho modelo, por tanto, cuanto mayor es el porcentaje de espacio ocupado, mayor es el nivel de detalle. En las opciones del modelo, ***Unreal*** permite definir los porcentajes de pantalla a los que es necesario "saltar" al siguiente nivel de detalle, de modo que, para modelos con diferentes funciones en la escena, se pueden asignar diferentes métodos de gestión para sus LODs.

Junto con los LODs, el **suavizado** mediante la modificación del **espacio tangencial *(Tangent space)*** [fig. 53] [fig. 54] [fig. 55] es otra característica a tener muy en cuenta si se está creando un modelo para un motor de videojuegos.
La manipulación de normales en el espacio tangencial de un polígono permite simular suavizados de superficies sobre la malla sin crear geometría adicional, valiéndose exclusivamente de las propiedades típicas del mapeado de normales. Es extremadamente útil en el modelado de ***props*** y de elementos de fondo o tercer plano (edificios, estructuras o piezas ***hard surface***), ya que permite simular superficies suavizadas sin necesidad de añadir polígonos a la malla.

Para conseguir estos resultados, se recurre a una estrategia de visualización que utiliza las propiedades de un concepto matemático conocido como espacio tangencial.

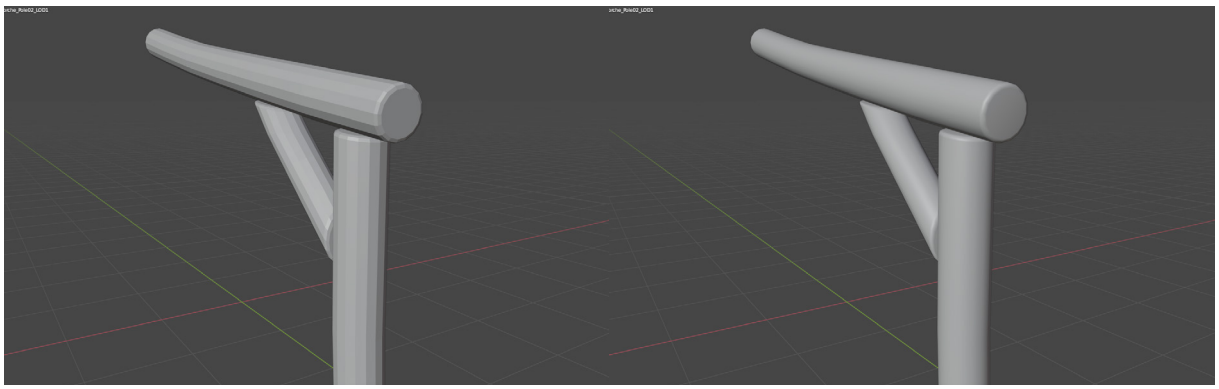
Al exportar los modelos desde Maya, 3DSMax o Blender, se deben salvar los archivos, en formato fbx, conservando el espacio tangencial asignado, para así poder importar en ***Unreal*** las mismas propiedades de suavizado que se han definido en el programa de origen.

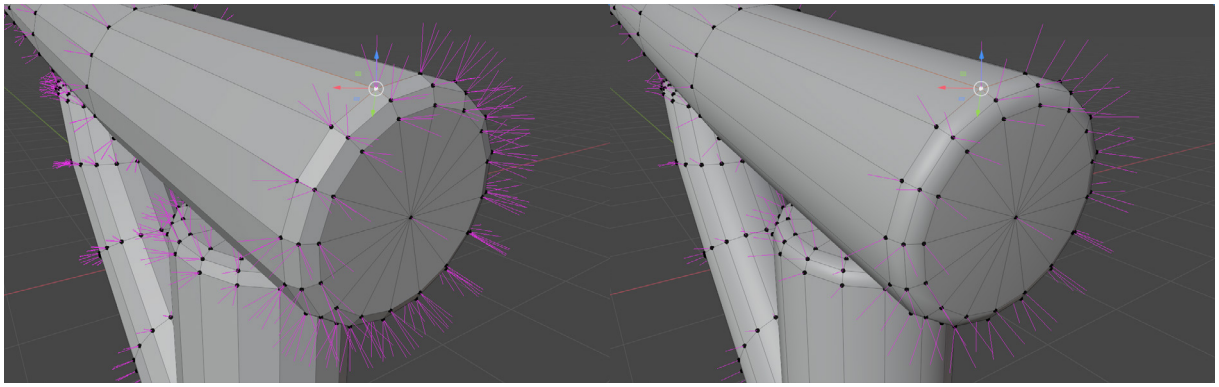
Esta técnica es prácticamente imprescindible en el uso de ***Unreal Engine***, pues permite ahorrar mucho esfuerzo de procesado sin apenas sacrificar detalle en la geometría de los modelos.

Optimización de *Shaders*
En la misma tónica se encuentran las opciones de **Teselado** y **Desplazamiento**. Estas dos propiedades permiten generar más geometría sobre la malla de forma procedural, y desplazarla para generar volumen de manera procedural, utilizando como base un mapa de desplazamiento que defina las formas con las cuales se ha de desplazar la malla.

El teselado es un proceso que permite subdividir cada uno de los ***tris*** que componen la malla en múltiplos regulares de la geometría original. Se puede definir el material para que el teselado suceda de manera proporcional a la cercanía de la cámara al modelo, lo que se conoce como **teselado adaptativo**. Este concepto es extremadamente importante cuando se texturizan superficies que presentan irregularidades cuyo volumen no es necesario en distancias de visualización alejadas del espectador.

Para generar volumen efectivo sobre la geometría creada mediante teselado se utilizan mapas de desplazamiento que "empujan" la malla del modelo en función de los valores en escala de gris asignados a la textura.





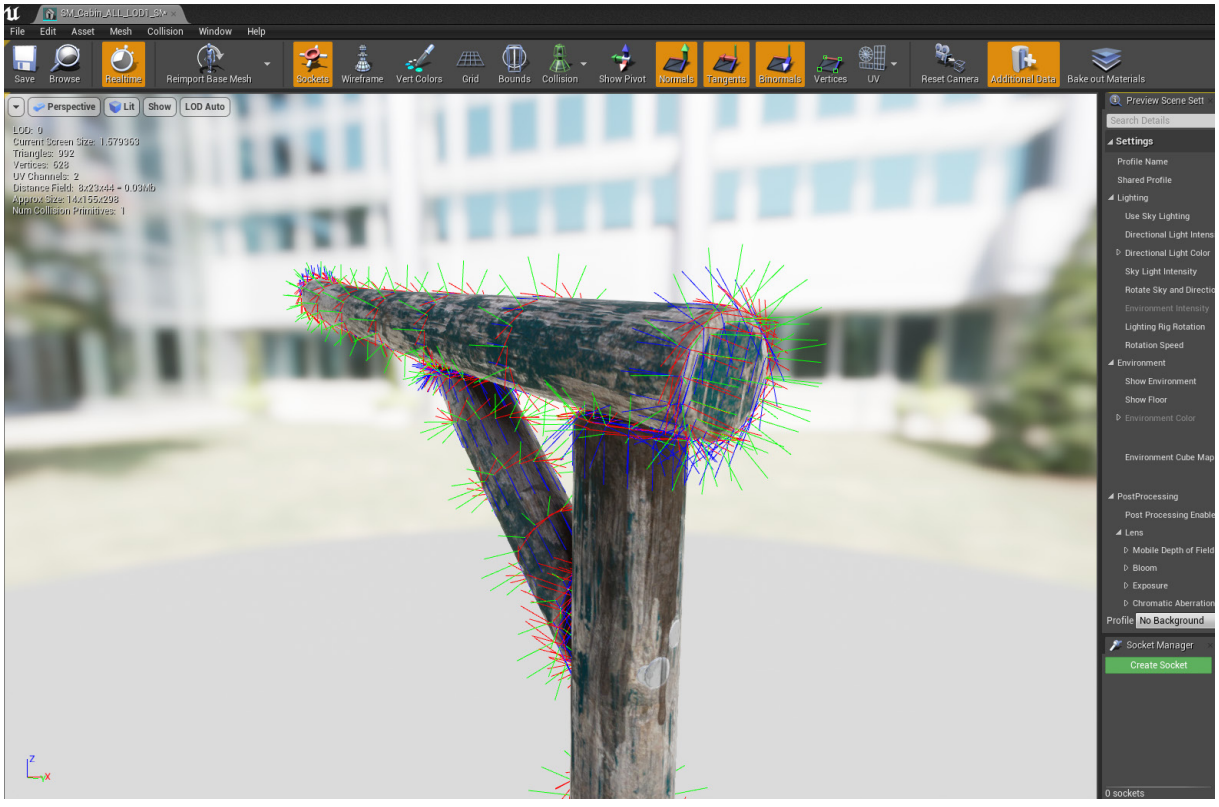


fig. 53 Ejemplo de aplicación de suavizado. A la izquierda de la imagen, modelo sin suavizar conservando el espacio tangencial por defecto; a la derecha, mismo modelo pero suavizado modificando el espacio tangencial. Ambos ejemplos tienen la misma topología de malla.

fig. 54 Visualización de Normales (*splited-normals*) en un espacio tangencial suavizado y sin suavizar.

fig. 55 Visualización del espacio tangencial exportado en el mismo modelo dentro de *Unreal Engine.*

Teselar un modelo consume recursos de la GPU, por tanto, establecer un proceso adaptativo en función de la distancia de visionado es fundamental para que ***Unreal*** se ejecute óptimamente. Por ejemplo, para construir una pared de ladrillo se puede utilizar un modelo 3D constituido por un plano rectangular bidimensional, sin ningún tipo de volumen que simule los ladrillos. La sensación de relieve se simula estableciendo un mapa de normales en el material PBR, pero cuando la cámara se aproxime será necesario ver como los ladrillos sobresalen si se quiere obtener un acabado fotorrealista; la combinación de teselado adaptativo y desplazamiento permite crear este volumen de manera procedural en tiempo real, y ajusta el nivel de subdivisiones en función de la distancia a la que se encuentra el espectador o la cámara.

Igual que en otros programas, ***Unreal Engine*** hace uso de un sistema de nodos para definir los **materiales** asignados a los modelos. No obstante, este sistema es sensiblemente diferente al de otros motores, pues presenta características propias para optimizar el flujo de trabajo. Por ejemplo, dentro de un material se pueden referenciar funciones de otros materiales [fig. 56] ya creados para simplificar la lectura en tiempo real.

Un concepto fundamental cuando se trabaja con materiales en un motor de videojuegos es la posibilidad de poder crear múltiples instancias de un mismo material maestro. Una **instancia de material** [fig. 57] permite modificar parámetros definidos en el material maestro sin tener que volver a compilar todo el árbol de nodos. Este proceso permite que el motor no tenga que calcular un material por cada uno de los objetos posicionados en la escena, pues toma la referencia del material maestro y aplica las modificaciones en cada una de las instancias.

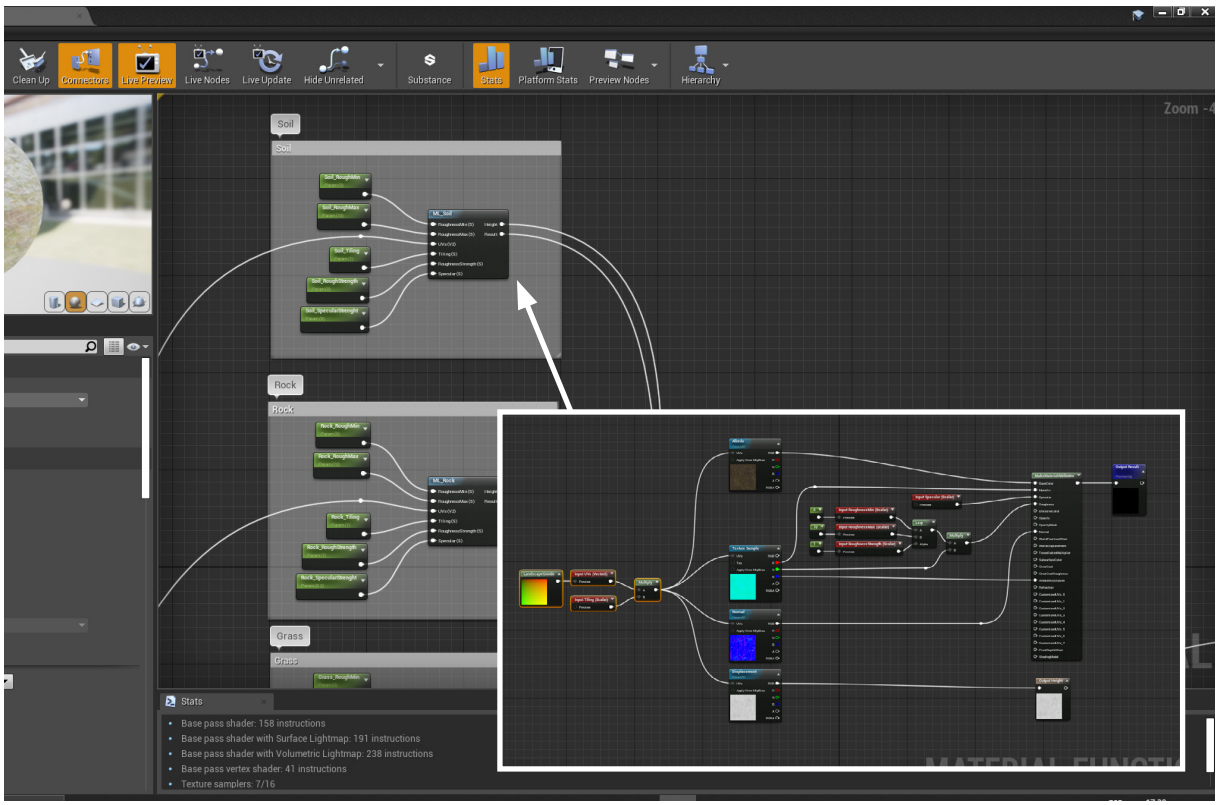
Por tanto, lo habitual cuando se texturiza en ***Unreal*** es crear un material común para todos los modelos con propiedades físicas similares, es decir, uno para materiales difusos, otro para translúcidos y otro para metálicos. Posteriormente, se crean distintas instancias de estos materiales, una por cada modelo 3D en escena, y se modifican los parámetros que controlan las características propias de cada objeto. Estas estrategias ayudan a mantener una baja complejidad de sombreadores en toda la escena [fig. 52].

3.1.2. Iluminación

La iluminación en ***Unreal*** merece una explicación aparte por sus peculiaridades respecto de otros motores. Originariamente, ***Unreal*** no ofrecía la posibilidad de renderizar con algoritmos de ***Ray tracing,*** aunque esta situación ha cambiado en el último año tras la aparición de la tecnología RTX. El proceso de render en ***Unreal*** es una actividad bastante compleja que basa su arquitectura en una ejecución ***multi-threating*** de diferentes tareas. En esencia, el render se construye combinando diferentes algoritmos con bajo impacto en los tiempos de cálculo para renderizar en tiempo real las escenas construidas dentro del motor.

Unreal, al igual que hacen otros motores de videojuego, utiliza iluminación ***bakeada*** [16] para simular los rebotes de luz difusa. Las sombras se calculan en un proceso de pre-renderizado con algoritmos de radiosidad e iluminación directa; tras

16 . Iluminación precalculada antes de ejecutar el render en tiempo real.



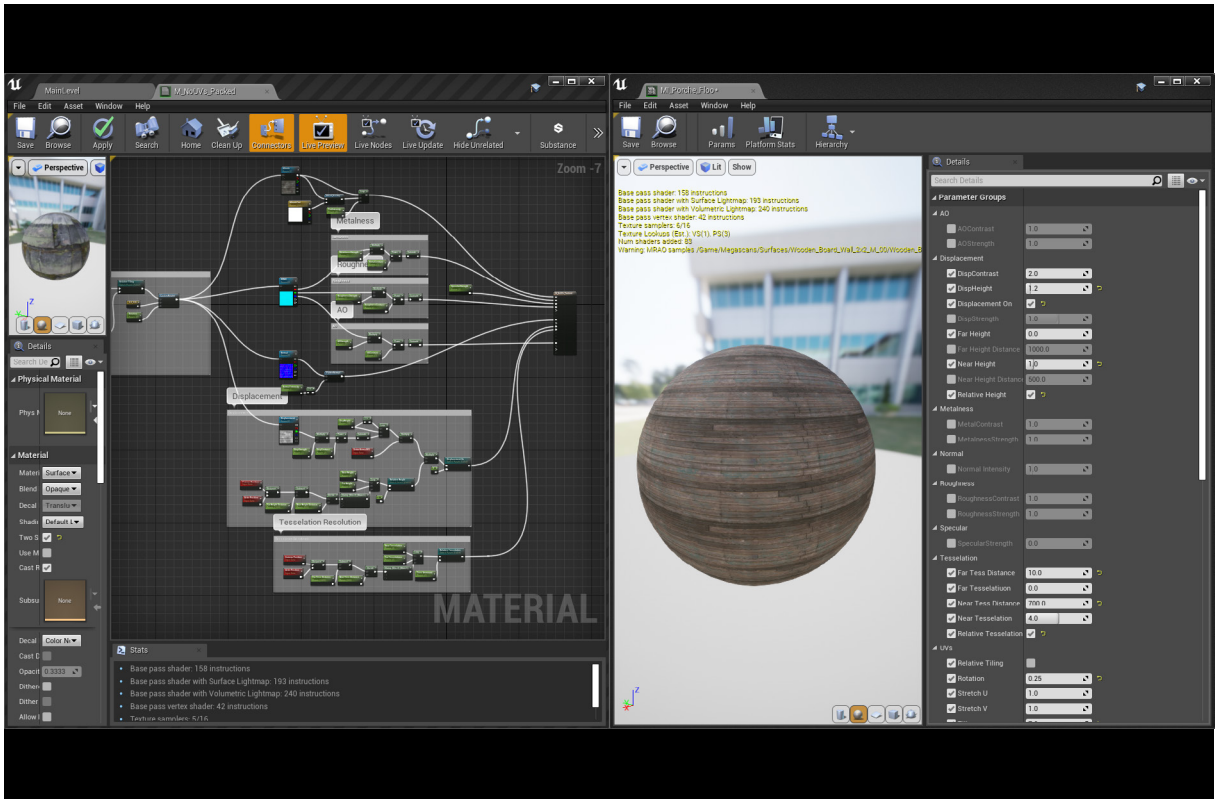


fig. 56 Visualización de “función de material” (recuadro inferior derecha) dentro del material maestro empleado para sombrear el terreno de la escena.

fig. 57 Editor de material maestro (izquierda) y de instancia de material (derecha).

el cálculo, se “pintan” sobre un tipo de textura específico que se aplicará sobre los modelos que se ven afectados por dichas sombras. Estas texturas se conocen como ***Lightmaps***, y son, en esencia, las sombras que los modelos en escena arrojan sobre otras superficies próximas. De este modo, cuando el juego se ejecute en tiempo real, el motor no necesitará calcular nuevamente la iluminación de cada *frame*, pues ya ha sido renderizada previamente, y simplemente ha de ser aplicada como una textura más.

Para poder texturizar los modelos 3D con los *lightmaps* correspondientes, cada modelo debe tener 2 IDs de coordenadas UV. El primero se utiliza para mapear las texturas de los materiales PBR y el segundo para los *lightmaps*. *Unreal* es capaz de crear por sí mismo este segundo conjunto de coordenadas UV; cada vez que se importa un modelo 3D dentro del motor se puede seleccionar la opción de crear UVs para *ligthmaps,* aunque para ciertos objetos será mejor que el artista las defina a mano en otro programa de modelado.

Este sistema de *lightmaps* es muy útil, pero presenta una desventaja crucial, ya que las sombras *bakeadas* son estáticas y no pueden variar con las interacciones del jugador. Si, por ejemplo, se mueve un objeto de la escena, o si se cambia la iluminación, los *lightmaps* correspondientes no se adaptan a las nuevas condiciones lumínicas y habrán de ser recalculados nuevamente [fig. 60]. Este fenómeno se ha de tener muy presente cuando se trabaja con *Unreal*, ya que condiciona el modo en el que se opera un entorno CGI. Si no se quiere cometer errores de iluminación, se habrá de recurrir a otras técnicas incorporadas en el motor para iluminar correctamente.

Debido a este sistema de iluminación precalculada, *Unreal* permite definir en todos sus actores [17] tres estados diferentes de movilidad: **estático**, **estacionario** y **móvil**. Dependiendo del modo elegido, se define la relación y el comportamiento de las luces en escena con el resto de los objetos.

Estático: si una luz se define como estática, esta no puede variar durante el juego. Son luces que no pueden ser movidas ni cambiadas durante el tiempo de ejecución, por lo cual tampoco pueden arrojar sombras dinámicas de objetos en movimiento; pero, si se combinan con objetos estáticos, son capaces de reproducir sombras de área en superficies de contacto.

Son calculadas mediante *lightmaps* y, una vez procesadas, no presentan mayor impacto sobre el rendimiento del motor. Los objetos definidos como móviles no pueden interactuar con este tipo de luces del mismo modo que lo harían los estáticos [fig. 60].

De los tres estados de movilidad, es el que presenta menor impacto de rendimiento, pero sacrificando calidad y versatilidad.

Estacionario: este estado permite que las luces presenten características de los otros dos (estático y móvil) en situaciones concretas. Las luces estacionarias están pensadas para ocupar una posición fija en la escena, pero se pueden modificar

17 . *Unreal* denomina a los elementos distribuidos en escena como “actores”. Existen actores de iluminación, mallas estáticas, reflexiones o volúmenes de posprocesado, entre otros.

algunas de sus propiedades como la intensidad, el brillo o el color, en tiempo real. Esta es su principal diferencia respecto de las luces estáticas, que no pueden ser modificadas de ningún modo durante el tiempo de ejecución.

No obstante, los cambios en tiempo real se representan mediante **iluminación directa**, de modo que, tanto los rebotes de iluminación difusa como las sombras de área no podrán sufrir cambios dinámicos con este tipo de luz. Por el contrario, la información de **iluminación indirecta (difusa o rebotada)** se almacena dentro de los ***lightmaps***, del mismo modo que se hace con las luces estáticas.

Unreal decide qué tipo de iluminación aplicar en función de los estados de movilidad definidos en cada uno de los modelos 3D. Así pues, una luz estacionaria aplicará iluminación ***bakeada*** sobre objetos estáticos, mientras que sobre objetos móviles aplicará iluminación dinámica.

El impacto de rendimiento de este tipo de luz es considerablemente alto, por tanto, y, aunque los resultados de iluminación son más realistas, ha de ser utilizado con cautela.

Móvil: este tipo de luz únicamente produce iluminación y sombras dinámicas. Se puede cambiar cualquiera de sus propiedades (posición, rotación, intensidad, color, radio, etc.) en tiempo real. No es posible obtener iluminación ***bakeada*** de estas fuentes de luz, por lo que los efectos de iluminación indirecta quedan completamente descartados [fig. 58].

Puesto que las sombras arrojadas por este tipo de luces son calculadas en tiempo real, su coste de rendimiento es el más elevado de los tres métodos.

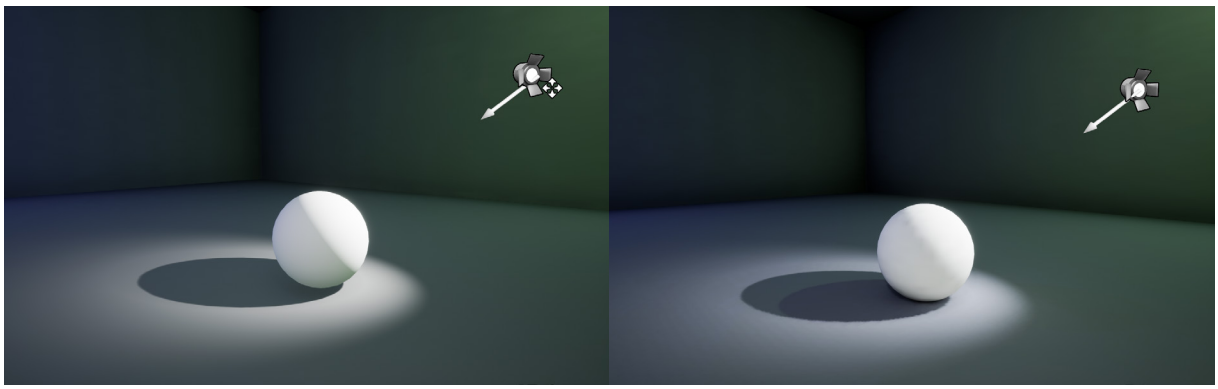
3.2. Unreal Engine V

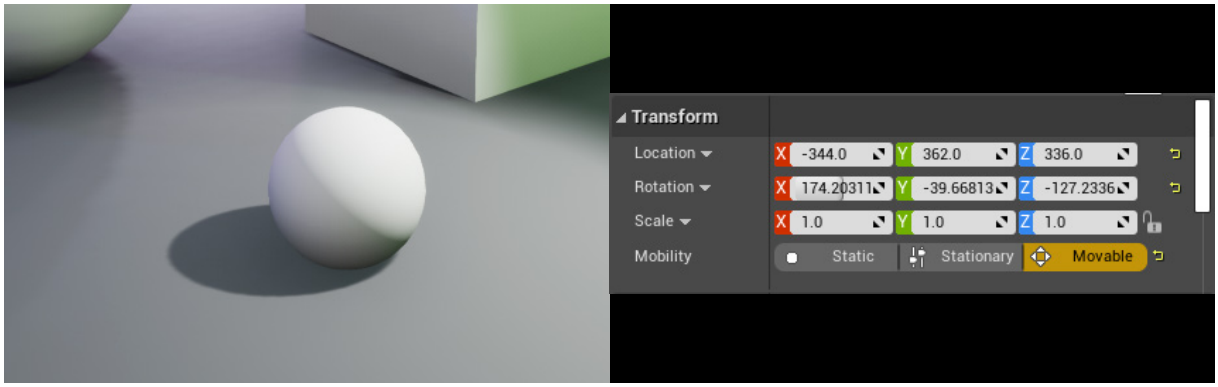
Al momento de plantear este TFM, ni siquiera había considerado la posibilidad de incorporar un apartado específico para introducir las versiones futuras de ***Unreal Engine***, pero los avances presentados durante el mes de mayo me impulsan a reconsiderar abiertamente las características que acabo de exponer previamente en este mismo capítulo.

Sin duda, las sucesivas actualizaciones de la versión actual del motor (***Unreal Engine 4***) han ido introduciendo importantes cambios en algunos aspectos, pero la tónica general de trabajo ha permanecido inalterada durante casi 10 años. Si esta última actualización hubiera continuado con la tendencia seguida hasta el momento actual por las versiones anteriores, no habría supuesto un gran cambio para la industria; pero, si lo que se presentó hace cuatro meses resulta ser cierto, esta versión cambiará por completo no solo el uso del motor en sí, también la manera en la que se concibe el diseño 3D para este *software*.

Si se cumplieran las expectativas de mejora presentadas en el último *teaser* de ***Unreal V***, todas las cuestiones de optimización tratadas en el apartado anterior dejarían de tener sentido.

El 10 de mayo de este mismo año, Epic Games publicó un vídeo que mostraba las dos nuevas tecnologías que serán el núcleo de la nueva versión de su motor.





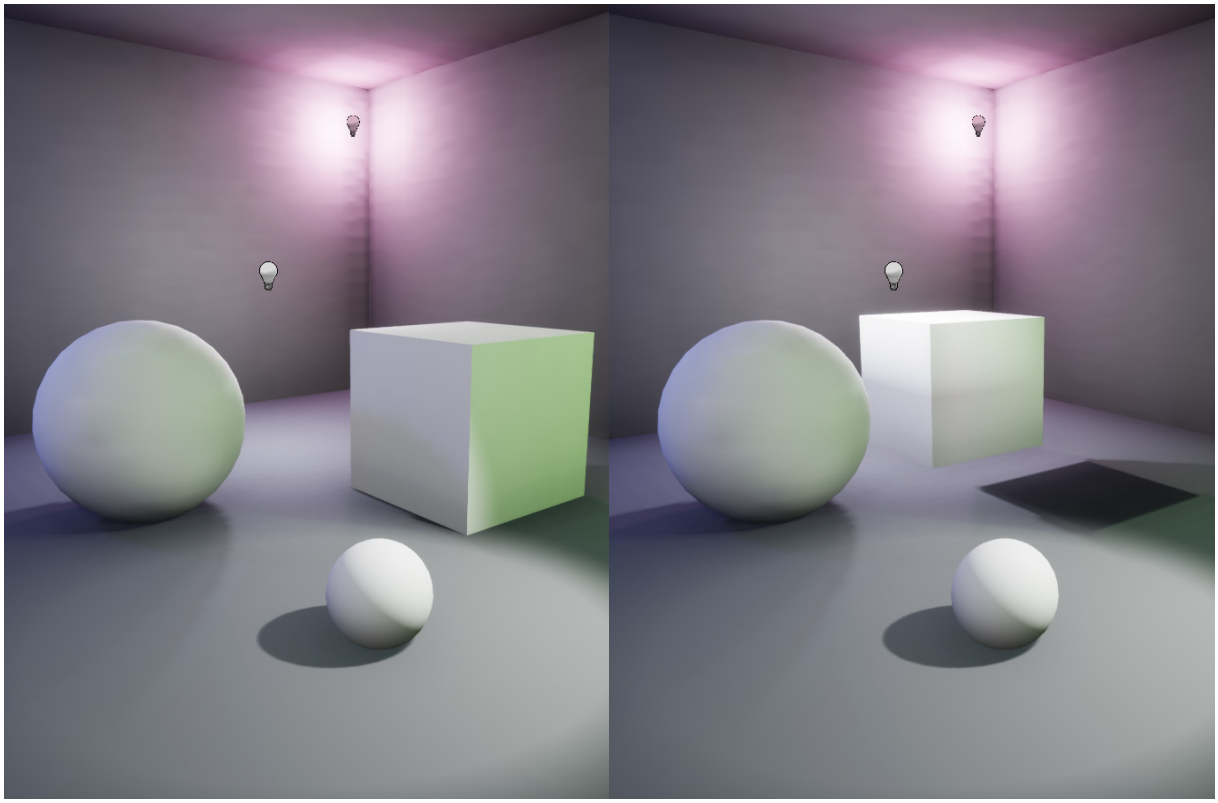


fig. 58 A la izquierda de la imagen, ejemplo de iluminación directa con luz móvil y objeto móvil. A la derecha, iluminación indirecta con luces estáticas y objeto estático.

fig. 59 A la izquierda, detalle de sombra de iluminación directa. A la derecha, opciones de movilidad en *Unreal Engine*.

fig. 60 Iluminación *bakeada*. La esfera pequeña se ha definido como movible, por lo que solo arroja sombras de iluminación directa. El resto de objetos se han definido como estáticos y presentan sombras de iluminación indirecta. A la derecha, error de iluminación en los *lightmaps* generados sobre el suelo al desplazar el cubo en el eje Y.

En el vídeo, se introduce un sistema de iluminación global completamente realista, capaz de gestionar los rayos de luz indirecta, de luz difusa, reflejada o rebotada, todo ello en tiempo real. Los miembros de Epic han llamado a esta tecnología ***Lumen***, y, como ellos mismos definen en su página web, es una solución de iluminación global que reacciona de forma inmediata a los cambios de luz y a las modificaciones en la escena («***A first look at Unreal Engine 5 - Unreal Engine***», 2020).

El sistema renderiza las reflexiones difusas con rebotes infinitos y reflejos especulares indirectos, todo ello en entornos enormes y con un gran nivel detalle, en escalas que van desde varios kilómetros hasta milímetros. Los artistas y diseñadores pueden crear escenas con mayor dinamismo si usan ***Lumen***, por ejemplo, pueden cambiar el ángulo del sol, apagar y encender luces o abrir un agujero en el techo, y la iluminación indirecta se adaptará acorde a los cambios efectuados en tiempo real («***A first look at Unreal Engine 5 - Unreal Engine***», 2020).

Esta solución elimina por completo tanto la necesidad de utilizar ***lightmaps*** para precalcular y simular la iluminación difusa, como la de crear UVs específicas para mapear dichas texturas. Sin duda, como afirma este párrafo en la web oficial de Epic Games («***A first look at Unreal Engine 5 - Unreal Engine***», 2020):

> ***a huge time savings when an artist can move a light inside the Unreal Editor and lighting looks the same as when the game is run on console.***

La otra pieza clave de esta versión es una tecnología que promete una gestión virtualizada de la geometría en escena. Este sistema, llamado ***Nanite***, permite importar en ***Unreal*** modelos de alto nivel poligonal, con millones o billones de polígonos, sin necesidad de "decimar" la malla ni de establecer niveles de detalle. El motor es capaz de escalar y "decimar" por sí mismo, y en tiempo real, la geometría de todos los modelos que aparecen en el plano para ajustarla a la distancia de visualización del espectador.

Para más inri, esta ***demo*** técnica ha sido ejecutada en una Playstation 5, una máquina que está muy lejos de tener la mayor aceleración gráfica del mercado, por lo que, en equipos especialmente preparados para VFX, esta versión de ***Unreal*** será todavía más eficiente.

Definitivamente, las nuevas características de ***Unreal V*** ponen de manifiesto la tendencia hasta ahora seguida en la industria de la computación gráfica: poder superar las limitaciones técnicas que entorpecen, o dificultan, el desarrollo creativo de los artistas. ***Lumen*** consigue eliminar la dificultad que existía en el planteamiento de la iluminación dinámica en el momento de diseñar un escenario, y ***Nanite*** permite evitar por completo procesos tediosos tales como la gestión de LODs, la retopología de malla o el ***bakeado*** de normales, permitiendo que los diseñadores y artistas centren sus esfuerzos en tareas de índole puramente creativa.

4. Conclusiones

No cabe duda, la tecnología 3D aplicada al medio audiovisual ha experimentado una evolución formidable en este último medio siglo. El desarrollo visual en CGI se ha manifestado de forma exponencial, presentando año tras año nuevas características que han permitido perfeccionar los resultados visuales en multitud de apartados. A medida que se han ido superando diferentes hitos en computación gráfica, el uso de CGI, y de otras técnicas relacionadas con el 3D se ha asentado en la comunidad artística hasta convertirse en una pieza esencial de muchos proyectos audiovisuales. Esta última hornada de procesadores gráficos presentados por Nvidia así lo demuestra; nuevamente, se abre una vía para los profesionales y artistas del medio, que, una vez más, comprueban cómo el desarrollo tecnológico ofrece varias posibilidades en relación con a la implementación de nuevas metodologías y procedimientos aplicables.

Pero, al igual que sucede en tantas otras áreas vinculadas al diseño, toda nueva tecnología siempre es sometida a debate dentro de la propia comunidad. Existe, de manera casi inherente, una duda que sobrevuela las disciplinas artísticas relacionadas con los medios digitales, una duda que lleva a los propios profesionales a cuestionarse hasta qué punto los resultados de su trabajo son el fruto de sus capacidades como artistas o si, por el contrario, son solo la consecuencia de la tecnología que utilizan.

Este debate se ha manifestado en incontables ocasiones; por ejemplo, cuando se presentó la primera cámara de fotos, los pintores y retratistas tradicionales temieron que su obra careciese de sentido ante esta nueva técnica de representación gráfica. Del mismo modo, cuando, en el cine, se introdujeron los primeros gráficos 3D, todos los profesionales que hasta entonces habían trabajado para dar vida a los efectos visuales de las películas (escenógrafos, artistas de marionetas, modeladores de maquetas, maquillaje, artistas de *matte-painting*, etc.) se hicieron la misma pregunta ¿Acaso sigue teniendo sentido mi trabajo si un ordenador es capaz de hacerlo mejor, más rápido y por menos dinero?

Es una preocupación lícita, que está totalmente justificada si consideramos los hechos desde un punto de vista puramente objetivo. Es lógico pensar que un profesional que ha dedicado gran parte de su vida a perfeccionar técnicas propias del medio en el que trabaja, vea peligrar su carrera cuando esas técnicas pueden ser sustituidas por una tecnología mejor, que él todavía no domina. Es un debate que genera gran inquietud en la comunidad y que afecta prácticamente a todos por igual, independientemente del campo profesional al que nos dediquemos. Pero, más allá de los cambios en las herramientas y procedimientos utilizados, lo que subyace a este debate es un problema de concepto.

La tecnología nos va a forzar siempre a reabrir este debate, que pone el foco sobre lo que debería ser considerado arte y si algo tiene menos valor artístico por el simple hecho de haber sido producido con tecnología que facilita su creación.

En parte, este debate deriva de la percepción que la propia sociedad tiene de los procesos artísticos en sí. La idea arcaica de que el artista debe ser un genio, artífice de su obra únicamente mediante el uso de sus manos desnudas, sigue presente en el

subconsciente de muchos de nosotros. Inconscientemente, atribuimos menos valor a un proyecto que haya sido creado con la ayuda de la tecnología, por el simple hecho de apreciarlo como falso, o al menos, no tan genuino, pues su ejecución es "más fácil" y requiere "menos esfuerzo".

Pero, en mi opinión, la pregunta debería ser si tiene sentido seguir considerando la tecnología como una característica intrínseca del propio proceso creativo. Si algo podemos afirmar, en relación con el desarrollo tecnológico experimentado en las últimas décadas, es la existencia de una lucha constante que los artistas han sufrido contra las limitaciones del ***hardware*** y el ***software*** utilizado. Las herramientas digitales, en ningún caso, deberían ser consideradas un fin creativo en sí mismas, deberían ser equiparadas a un utensilio que permite materializar un concepto o una idea.

El valor de un proyecto artístico no debería estar vinculado a las aptitudes relacionadas con el uso de un programa informático, debería estar asociado con los fundamentos que dotan de sentido artístico a dicho proyecto. Por poner un ejemplo, la principal función de un diseñador de entornos 3D no es poseer un extenso conocimiento en el manejo de ***Unreal*** o Maya, sino entender cómo funciona la composición de una escena, cómo distribuir los elementos que la componen y qué lenguaje visual utilizar para alcanzar los objetivos artísticos fijados, independientemente de si el uso de una herramienta digital facilita todo el proceso de ejecución.

Además, la democratización tecnológica trae como consecuencia una resignificación de los puestos especializados dentro de los equipos de producción, algo que, de hecho, ya estamos viendo en muchos casos. A medida que desaparezcan las barreras técnicas relacionadas con el desarrollo artístico (ej. Unreal V y sus nuevas características), será lógico pensar que la industria demandará profesionales con perfiles más generalistas, capaces de desempeñar diversas tareas y de desenvolverse en áreas diferentes. Es prudente pensar que desaparecerán los perfiles especializados en actividades concretas como el modelado 3D o el texturizado, del mismo modo que en el cine se reinterpretaron los puestos de los artistas de ***matte-painting***.

En conclusión, el uso de nuevas herramientas digitales en ningún caso debería ser visto como una amenaza, más bien debería ser considerado como una oportunidad para que cada vez más artistas de diferentes ámbitos colaboren en el desarrollo de proyectos multidisciplinares. Por ello, me gustaría finalizar este TFM con una cita que Andrew Maximov, artista 3D en Naughty Dog, pronunció durante su ponencia en el GDC de 2017 (***The Future of Art Production in Games***, 2017):

> En resumen, [...] se puede decir que crear arte es, esencialmente, un proceso de dotar de significado y, en ese caso, da igual cuanto optimices el proceso, el arte no va a desaparecer porque todavía habrás de pensar en qué estás comunicándole a tu audiencia. Si tu arte no tiene un significado suficiente detrás, no va a tener éxito, porque no habrá nada con lo que la gente pueda vibrar.

5. Bibliografía

A first look at Unreal Engine 5—Unreal Engine. (2020, junio 15). Recuperado 22 de septiembre de 2020, de https://www.unrealengine.com/en-US/blog/a-first-look-at-unreal-engine-5

Matthews, D. (s. f.). Hellblade | The Independent AAA Proposition. Recuperado 19 de enero de 2020, de https://www.hellblade.com/the-independent-aaa-proposition/

Okun, J. A., & Zwerman, S. (2010). The VES handbook of visual effects: Industry standard VFX practices and procedures (Third edition., Vols. 1–1 online resource). London ; Focal Press. Recuperado de https://ucm.on.worldcat.org/oclc/647764743

Reynolds, M. (2018, marzo 29). How Hellblade: Senua's Sacrifice was made as an «indie triple-A» game on a tight budget. Recuperado 22 de septiembre de 2020, de Eurogamer website: https://www.eurogamer.net/articles/2018-03-29-ninja-theory-hellblade-indie-triple-a

Sofge, E. (2010, febrero 12). The Truth About Robotic's Uncanny Valley [Blog]. Recuperado 26 de agosto de 2020, de Popular Mechanics website: https://www.popularmechanics.com/technology/robots/a5001/4343054/

The Future of Art Production in Games [Video]. (2017). Recuperado de https://www.youtube.com/watch?v=7Rt0wOyCCAI

Anexo

Fuentes utilizadas para elaborar el TFM, pero que no aparecen citadas en el cuerpo:

3D real-time rendering—How does it work? (s. f.). Recuperado 26 de agosto de 2020, de Unity website: https://unity3d.com/real-time-rendering-3d

Andy Serkis: Getting under his characters' skin. (s. f.). Recuperado 25 de septiembre de 2020, de https://www.cbsnews.com/pictures/andy-serkis-getting-under-his-characters-skin/

Art of LED Wall Virtual Production, Part One: 'Lessons from the Mandalorian'. (2020, marzo 4). Recuperado 30 de septiembre de 2020, de Fxguide website: https://www.fxguide.com/fxfeatured/art-of-led-wall-virtual-production-part-one-lessons-from-the-mandalorian/

Brejon, C. (2020, marzo 9). CG Cinematography [Blog]. Recuperado 23 de marzo de 2020, de The CG Cinematography Book website: https://chrisbrejon.com/cg-cinematography

Brinkmann, R. (2008). The Art and Science of Digital Compositing: Techniques for Visual Effects, Animation and Motion Graphics. Morgan Kaufmann.

Caulfield, B. (2018, marzo 19). What's the Difference Between Ray Tracing, Rasterization? | NVIDIA Blog. Recuperado 19 de enero de 2020, de The Official NVIDIA Blog website: https://blogs.nvidia.com/blog/2018/03/19/whats-difference-between-ray-tracing-rasterization/

CogSci-2005 Workshop: Toward Social Mechanisms of Android Science. (2007, marzo 2). Recuperado 25 de septiembre de 2020, de https://web.archive.org/web/20171101031655/http://androidscience.com/theuncannyvalley/proceedings2005/uncannyvalley.html

Creating Gollum. (2013). Recuperado de https://www.youtube.com/watch?time_continue=3&v=w_Z7YUyCEGE&feature=emb_title

Cubic Motion: The World's Definitive Solution for Facial Animation. (s. f.). Recuperado 19 de enero de 2020, de Cubic Motion website: https://cubicmotion.com

Dent, S. (2020, febrero 21). See how «The Mandalorian» used Unreal Engine for its real-time digital sets [Engadget]. Recuperado 1 de octubre de 2020, de Engadget website: https://www.engadget.com/2020-02-21-mandalorian-ilm-stagecraft-real-time-digital-sets.html

Desowitz, B., & Desowitz, B. (2020, mayo 8). 'The Mandalorian' Leads the Way: Real-Time Virtual Production Is Saving Hollywood During the Lockdown. Recuperado 30 de septiembre de 2020, de IndieWire website: https://www.indiewire.com/2020/05/the-mandalorian-real-time-virtual-production-saving-hollywood-lockdown-vfx-1202230120/

DNEG. (s. f.). Recuperado 4 de septiembre de 2020, de DNEG website: https://www.dneg.com/awards/

Dunlop, R. (2014). Production Pipeline Fundamentals for Film and Games. CRC Press.

Failes, I. (2018, noviembre 19). A Virtual Production Explainer: What It Is, And What It Could Mean For Your Project [Cartoon Brew]. Recuperado 19 de enero de 2020, de Cartoon Brew website: https://www.cartoonbrew.com/vfx/a-virtual-production-explainer-what-it-is-and-what-it-could-mean-for-your-project-166554.html

From Previs to Final in Five Minutes | Siggraph 2016 | Full Presentation. (s. f.). Recuperado de https://www.youtube.com/watch?v=OMENy0ptoyM

García Barroso, A. (2016). Posproducción de efectos especiales: Integración de imagen digital 2D y 3D (Memoria para optar al grado en doctor, Universidad Complutense de Madrid). Universidad Complutense de Madrid, Madrid, España. Recuperado de https://eprints.ucm.es/39136/

Gordon, W. (2019, octubre 10). PS5: What Is Ray Tracing, and Should You Care? - IGN. Recuperado 26 de septiembre de 2020, de https://www.ign.com/articles/2019/10/10/what-is-ray-tracing-and-should-you-care

Hardawar, D. (2018, febrero 24). «Black Panther» is amazing. Why are its CG models so terrible? Recuperado 5 de septiembre de 2020, de Engadget website: https://www.engadget.com/2018-02-24-black-panther-vfx-models.html

Harryhausen, R. (2003, diciembre 20). Ray Harryhausen on making Jason and the Argonauts. The Guardian. Recuperado de https://www.theguardian.com/books/2003/dec/20/featuresreviews.guardianreview16

Leberecht, S. (2014). Life After Pi [Documental]. Recuperado de https://www.youtube.com/watch?v=9lcB9u-9mVE

Letteri, J. (2013). Computer animation: Digital heroes and computer-generated worlds. Nature, 504(7479), 214-216. https://doi.org/10.1038/504214a

Martínez Sánchez, S. (2017). Evolución de los efectos visuales en la historia del cine y su influencia sobre la industria del video musical (Memoria para optar al grado en doctor). Universidad Complutense de Madrid, Madrid, España.

Marvel's Black Panther VFX Breakdown. (2018, abril 10). Recuperado 4 de septiembre de 2020, de Animation Boss website: http://www.animationboss.net/behind-scenes-marvels-black-panther-vfx/

Matthews, D. (s. f.). Hellblade | Development Diary 10: Capturing Performance. Recuperado 19 de enero de 2020, de https://www.hellblade.com/development-diary-10-capturing-performance/

Motion capture (mocap): What is it? (2017, mayo 13). Recuperado 30 de agosto de 2020, de TESLASUIT website: https://teslasuit.io/blog/motion-capture-what-it-is/

Movable Lights. (s. f.). Recuperado de https://docs.unrealengine.com/en-US/Engine/Rendering/LightingAndShadows/LightMobility/DynamicLights/index.html

NVIDIA TURING GPU ARCHITECTURE. (2018). Recuperado de https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf

Optitrack | VR and Filmmaking Collide in the Making of «The Lion King». (s. f.). Recuperado 19 de enero de 2020, de OptiTrack website: http://optitrack.com/about/press/20190910.html

Peddie, J. (2019, octubre 8). Ray Tracing Will be Everywhere in 2020. Recuperado 21 de febrero de 2020, de ACM SIGGRAPH Blog website: https://blog.siggraph.org/2019/10/ray-tracing-will-be-everywhere-in-2020.html/

Perkins, M. (2019, julio 25). The Virtual Production Field Guide: A new resource for filmmakers. Recuperado 22 de septiembre de 2020, de Unreal Engine—Blog website: https://www.unrealengine.com/en-US/blog/virtual-production-field-guide-a-new-resource-for-filmmakers

Pharr, M., Jakob, W., & Humphreys, G. (2016). Physically Based Rendering: From Theory to Implementation (3 edition). Cambridge, MA: Morgan Kaufmann.

Put your (digital) game face on. (2016, abril 24). Recuperado 24 de septiembre de 2020, de Fxguide website: https://www.fxguide.com/fxfeatured/put-your-digital-game-face-on/

State of Unreal | GDC 2019 | Unreal Engine [Conferencia]. (2019). Recuperado de https://www.youtube.com/watch?v=s55Uob494Do

Static Lights. (s. f.). Recuperado de https://docs.unrealengine.com/en-US/Engine/Rendering/LightingAndShadows/LightMobility/StaticLights/index.html

Stationary Lights. (s. f.). Recuperado de https://docs.unrealengine.com/en-US/Engine/Rendering/LightingAndShadows/LightMobility/StationaryLights/index.html

Strauss, B. (2020, marzo 14). How the worlds of The Mandalorian (and Baby Yoda) got made in Manhattan Beach. Recuperado 25 de septiembre de 2020, de Daily Bulletin website: https://www.dailynews.com/how-the-whole-world-and-worlds-of-the-mandalorian-got-made-in-manhattan-beach

The Virtual Production of The Mandalorian, Season One. (s. f.). Recuperado de https://www.youtube.com/watch?v=gUnxzVOs3rk&feature=youtu.be

Unreal Engine | Programs | Exploring the future of virtual production. (s. f.). Recuperado 19 de enero de 2020, de Unreal Engine website: https://www.unrealengine.com/en-US/programs/virtual-production

Unreal Engine in-camera VFX: a behind-the-scenes look. (2019, noviembre 12). Recuperado 20 de septiembre de 2020, de https://www.unrealengine.com/en-US/spotlights/unreal-engine-in-camera-vfx-a-behind-the-scenes-look

Unreal Engine User Group at SIGGRAPH 2019 [Conferencia]. (s. f.). Recuperado de https://www.youtube.com/watch?v=apLzZBqfqeU

Venkatasawmy, R. (2013). The digitization of cinematic visual effects: Hollywood's coming of age (Vols. 1–1 online resource (ix, 333 pages)). Lanham, Md.: Lexington Books. Recuperado de https://ucm.on.worldcat.org/oclc/827944887

Virtual production: Performance capture for everyone. (s. f.). Recuperado 26 de agosto de 2020, de https://www.unrealengine.com/en-US/blog/virtual-production-performance-capture-for-everyone

Weta Digital's Tissue System. (s. f.). Recuperado de https://www.youtube.com/watch?v=7VlthWa5pu8

What you need to know about 3D motion capture. (s. f.). Recuperado 28 de septiembre de 2020, de Engadget website: https://www.engadget.com/2014-07-14-motion-capture-explainer.html

Why «The Mandalorian» Uses Virtual Sets Over Green Screen | Movies Insider. (2020). Recuperado de https://www.youtube.com/watch?v=Ufp-8weYYDE8

Xsens. (s. f.-a). A history of motion capture. Recuperado 26 de agosto de 2020, de https://www.xsens.com/a-history-of-motion-capture

Xsens. (s. f.-b). Xsense | MVN Animate. Recuperado 19 de enero de 2020, de https://www.xsens.com/products/mvn-animate

Ye, M., Xianwang Wang, Yang, R., Liu Ren, & Pollefeys, M. (2011). Accurate 3D pose estimation from a single depth image. 2011 International Conference on Computer Vision, 731-738. Barcelona, Spain: IEEE. https://doi.org/10.1109/ICCV.2011.6126310

Mi nombre es David Martínez. He desarrollado parte de mi actividad profesional reciente como posproductor y editor de video en el equipo de marketing y publicidad de la firma BOSCH, con sede en Soleura, Suiza. Como responsable de edición, estaba a cargo de montar los videos destinados a publicarse en las redes sociales oficiales de la empresa. También participé en la ideación y desarrollo de los guiones e historias narradas en los vídeos publicados y, al mismo tiempo, trabajaba en el equipo de VFX (efectos especiales) y composición digital durante la fase de posproducción.

Comenzé mi formación en la Universidad Complutense de Madrid cursando el grado de Bellas Artes, pero ya durante los primeros años decidí orientar mi carrera profesional hacia el diseño multimedia y la creación audiovisual. Me interesé por el uso de entornos de diseño 3D y de posproducción de video, y aprendí de forma autónoma a utilizar las herramientas demandadas por este sector.

Mi trayectoria profesional empieza como diseñador web freelance para posteriormente realizar algunos proyectos como autónomo en el mundo de la fotografía.

Hoy, tras regresar a Madrid, mi intención es poder, en un futuro, formar parte de equipos que participen en proyectos audiovisuales, ya sea en vídeo, animación 3D o videojuegos.