UNIVERSIDAD DE MURCIA

ESCUELA INTERNACIONAL DE DOCTORADO

TESIS DOCTORAL

ESTUDIO DE LOS MATERIALES EN ENTORNOS 3D CON RENDER EN TIEMPO REAL PARA LA CREACIÓN EN ARTES DIGITALES.

D. José Javier Luis Tello

2023

UNIVERSIDAD DE MURCIA

ESCUELA INTERNACIONAL DE DOCTORADO

TESIS DOCTORAL

ESTUDIO DE LOS MATERIALES EN ENTORNOS 3D CON RENDER EN TIEMPO REAL PARA LA CREACIÓN EN ARTES DIGITALES.

DOCTORANDO:
José Javier Luis Tello

DIRECTORES:
Gerardo Diego Robles Reinaldos
Alfonso Burgos Risco

UNIVERSIDAD DE MURCIA

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD
DE LA TESIS PRESENTADA PARA OBTENER EL TÍTULO DE DOCTOR
Aprobado por la Comisión General de Doctorado el 19-10-2022

D./Dña. JOSË JAVIER LUIS TELLO

doctorando del Programa de Doctorado en

PROGRAMA DE DOCTORADO EN ARTES Y HUMANIDADES: BELLAS ARTES, LITERATURA, TEOLOGÍA, TRADUCCIÓN E INTERPRETACIÓN Y LINGÜÍSTICA GENERAL E INGLESA (PLAN 2013)

de la Escuela Internacional de Doctorado de la Universidad Murcia, como autor/a de la tesis presentada para la obtención del título de Doctor y titulada:

ESTUDIO DE LOS MATERIALES EN ENTORNOS 3D CON RENDER EN TIEMPO REAL PARA LA CREACIÓN EN ARTES DIGITALES.

y dirigida por,

D./Dña. GERARDO DIEGO ROBLES REINALDOS

D./Dña. ALFONSO BURGOS RISCO

D./Dña.

DECLARO QUE:

La tesis es una obra original que no infringe los derechos de propiedad intelectual ni los derechos de propiedad industrial u otros, de acuerdo con el ordenamiento jurídico vigente, en particular, la Ley de Propiedad Intelectual (R.D. legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, modificado por la Ley 2/2019, de 1 de marzo, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia), en particular, las disposiciones referidas al derecho de cita, cuando se han utilizado sus resultados o publicaciones.

Si la tesis hubiera sido autorizada como tesis por compendio de publicaciones o incluyese 1 o 2 publicaciones (como prevé el artículo 29.8 del reglamento), declarar que cuenta con:

- *La aceptación por escrito de los coautores de las publicaciones de que el doctorando las presente como parte de la tesis.*
- *En su caso, la renuncia por escrito de los coautores no doctores de dichos trabajos a presentarlos como parte de otras tesis doctorales en la Universidad de Murcia o en cualquier otra universidad.*

Del mismo modo, asumo ante la Universidad cualquier responsabilidad que pudiera derivarse de la autoría o falta de originalidad del contenido de la tesis presentada, en caso de plagio, de conformidad con el ordenamiento jurídico vigente.

En Murcia, a 5 de JUNIO de 2023

LUIS TELLO JOSE JAVIER - 76918057P

Firmado digitalmente por LUIS TELLO JOSE JAVIER - 76918057P
Fecha: 2023.07.07 10:21:09 +02'00'

Fdo.: JOSÉ JAVIER LUIS TELLO

Esta DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD debe ser insertada en la primera página de la tesis presentada para la obtención del título de Doctor.

Información básica sobre protección de sus datos personales aportados	
Responsable:	Universidad de Murcia. Avenida teniente Flomesta, 5. Edificio de la Convalecencia. 30003; Murcia. Delegado de Protección de Datos: dpd@um.es
Legitimación:	La Universidad de Murcia se encuentra legitimada para el tratamiento de sus datos por ser necesario para el cumplimiento de una obligación legal aplicable al responsable del tratamiento. art. 6.1.c) del Reglamento General de Protección de Datos
Finalidad:	Gestionar su declaración de autoría y originalidad
Destinatarios:	No se prevén comunicaciones de datos
Derechos:	Los interesados pueden ejercer sus derechos de acceso, rectificación, cancelación, oposición, limitación del tratamiento, olvido y portabilidad a través del procedimiento establecido a tal efecto en el Registro Electrónico o mediante la presentación de la correspondiente solicitud en las Oficinas de Asistencia en Materia de Registro de la Universidad de Murcia

RESUMEN

Esta tesis se enmarca en el Programa de Doctorado de Artes y Humanidades de la EIDUM, en concreto en el ámbito de las Artes Digitales y el desarrollo en entornos 3D con renderizado en tiempo real, que, como plataforma multimedia interactiva, tiene especial interés en el ámbito de las Artes.

Comenzando por el estudio y análisis de la literatura técnica, se contextualiza la evolución de las artes digitales hasta el desarrollo en entornos con *render real time*, se desglosa y define terminología específica, y gracias a los estándares alcanzados en el software que podemos encontrar en el mercado, se unifican y organizan los procesos más relevantes para la creación con dichas herramientas digitales, dejando patente la oportunidad de formación y estudio que supone este campo en la creación en artes.

Profundizaremos de forma práctica en los materiales PBR, traduciendo sus componentes a términos propios del ámbito de las artes y analizando el comportamiento visual de cada una de sus texturas por separado para generar una relación entre cada una de ellas y su impacto sobre el aspecto final de dichos materiales, aportando nuevas posibilidades para la creación de nuevo contenido y un mayor control sobre el resultado final en este ámbito.

En base al análisis previo del proceso y a la experimentación realizada, modificaremos el aspecto de un modelo 3D, obteniendo, a través de su renderizado en un entorno con *Render real-time*, diferentes valores y percepciones visuales, como ejemplo y muestra de las posibilidades de este tipo de materiales para la creación de contenido digital en el ámbito de las artes.

Esta metodología inductiva a partir de la usabilidad de materiales PBR demostrará la importancia y la necesidad de la aplicación del estudio, así como de la ampliación del conocimiento relacionado con la fenomenología de este tipo de materiales para entornos de

renderizado en tiempo real y las herramientas y disciplinas relacionadas en el campo de las artes digitales.

PALABRAS CLAVE

“Arte Digital”, “Renderizado *Real-time*”, “Material PBR”, “Multimedia”, “Interactivo”.

AGRADECIMIENTOS

A mis docentes, María del Carmen Castillo, Gerardo Diego Robles y Alfonso Burgos, por escucharme y embarcarse conmigo en este proyecto, por cederme su tiempo y tener una paciencia infinita, por su sabiduría y sus acertados consejos, y por guiarme y acompañarme hasta el final.

A la Universidad de Murcia y al personal de la EIDUM que me ha acompañado durante el viaje, por permitirme seguir avanzando en mi formación en el campo de las Artes.

A mi familia, Laura e Irene, por su apoyo en muchos sentidos, por aguantar las esperas, las noches en vela y mis nervios. Por aparcar sus deseos y sueños este tiempo para que yo pudiera completar esta tarea.

A mis padres, Ángel y María Pilar, porque siempre están ahí, escuchando mil veces el tema de mi tesis y mis enrevesadas explicaciones, mis dudas e inseguridades, por su consejo y sus ánimos.

Todos los hipervínculos de este documento han sido revisados a fecha:
11 de marzo de 2023.

INDICE

Página

1 SUMARIO 12

1.1 Índice de Abreviaturas 12

1.2 Índice de Figuras 13

1.3 Índice de Tablas 18

2 INTRODUCCIÓN 20

2.1 Hipótesis 27

2.2 Objetivos 30

2.3 Contexto histórico 32

2.3.1 Gráficos Digitales 32

2.3.2 Evolución de los gráficos 3D 35

2.3.3 Gráficos 3D y render *real-time* 41

2.3.4 Evolución tecnológica 48

3 METODOLOGÍA 56

3.1 Introducción 56

3.1.1 Análisis de los procesos para el desarrollo en entornos 3D con renderizado en tiempo real 57

3.2 Procesos del flujo de trabajo 62

3.2.1 Modelado 62

3.2.1.1 Malla 63

3.2.1.2 Normales de geometría 65

3.2.1.3 Tipologías de Modelos 3D 66

3.2.1.4 Selecciones y costuras 69

3.2.1.5 Mapeado UV 70

3.2.1.6 Modelos con animación 73

3.2.1.7 Densidad de píxeles 74

3.2.2 Entorno 3D 76

3.2.2.1 Cámara 77

3.2.2.1.1 Efectos de postprocesado 78

3.2.2.2 Iluminación 87

3.2.2.2.1 Iluminación Global (GI) 88

3.2.2.2.2 Tipos de emisores de luz virtual 89
3.2.2.3 Escena 92
3.2.3 Texturizado 94
3.2.3.1 Materiales PBR 96
3.2.3.2 *Shaders* 97
3.2.3.2.1 *Shader Lit* / *Unlit* 99
3.2.3.2.2 Modelos de simulación de la luz 100
3.2.3.2.3 *Shaders* y Transparencia 102
3.2.3.3 Texturas 102
3.2.3.3.1 Resolución 103
3.2.3.3.2 Creación de texturas 104
3.2.3.3.3 Encaje de texturas en el mapeado UV 106
3.3 Experimentación 109
3.3.1 Herramientas 109
3.3.1.1 *Unity* – Unity Technologies 109
3.3.1.2 *Microsoft Visual Studio* - Microsoft 110
3.3.1.3 *Photoshop* – Adobe System Incorporated 110
3.3.1.4 *Blender* – Fundación Blender 110
3.3.2 Configuración del proyecto 110
3.3.3 Componentes de un material PBR 116
3.3.4 Textura de muestreo 118
3.3.5 Ejecución de la aplicación 122
3.3.5.1 Color o Albedo 122
3.3.5.2 *Metallic* / *Smoothnes* 128
3.3.5.3 *Normal* 133
3.3.5.4 *Height* 137
3.3.5.5 *Occlusion* 142
3.3.5.6 *Emission* 146
3.4 Análisis de los Resultados 149
4 APLICACIÓN PRÁCTICA 154
5 CONCLUSIONES 164
6 REFERENCIAS 170
6.1 Bibliografía especializada 170
6.2 Webgrafía 174

1.- SUMARIO

1.1.- ÍNDICE DE ABREVIATURAS

2D - 2 dimensiones

2.5D - 2.5 dimensiones

3D - 3 dimensiones

AA - *Anti-aliasing* [Suavizado]

AGP - *Accelerated Graphics Port* [Puerto de Gráficos Acelerados]

AO - *Ambient Occlusion* [Oclusión Ambiental]

API - *Application Programming Interface* [Interfaz de Programación de Aplicaciones]

AR - *Augmented Reality* [Realidad aumentada]

CG - Cg - *C for Graphics* [C para gráficos]

CPU - *Central Processing Unit* [Unidad Central de Procesamiento]

FPS - *Frames Per Second* [Frames por segundo]

GI - *Global Illumination* [Iluminación global]

GLSL - OpenGL (*Graphic Library*) *Shading Language* [Lenguaje de Sombreado OpenGL]

GPU - *Graphic Processing Unit* [Unidad de Procesamiento Gráfico]

HDR - *High Dynamic Range* [Alta Gama Dinámica]

HDRP - *High Definition Render Pipeline* [Canal de Renderizado de Alta Definición

HLSL - *High-Level Shading Language* [Lenguaje de Sombreado de Alto Nivel]

HUD - *Heads-Up Display* (Visualizador Frontal)

LODs - LoDs - *Level of Detail* [Niveles de Detalle]

MOCAP - *Motion Capture* [Captura de Movimiento]

OS - *Operative System* [Sistema Operativo]

PBR - *Physically Based Rendering* [Renderizado basado en físicas]

RAM - *Random Access Memory* [Memoria de acceso aleatorio]

ROP - *Raster Output* [Salida de Rasterizado]

RP - *Render Pipeline* [Canal de Renderizado]

SSS - *Subsurface Scattering* [Dispersión del subsuelo/ Bajo la superficie]

TMU - *Texture Map Unit* [Unidad de Mapeo de Texturas]

URP - *Universal Render Pipeline* [Canal de Renderizado Universal]

VFK - *Visual effects* [Efectos Visuales]

VR - *Virtual Reality* [Realidad virtual]

VRAM - *video RAM* [Memoria de Video]

XR - *Extended Reality* [Realidad Extendida]

1.2.- INDICE DE FIGURAS

Fig. 1	K27 (1965) George Nees.	32
Fig. 2	*Rectangular Random Polygon* (1965) Frieder Nake.	32
Fig. 3	*Projection of a four-dimensional hypercube* (1965) Michael Noll.	32
Fig. 4	*Return to Square* (1968) Masao Komura, Makato Othake (CTG).	32
Fig. 5	Andy Warhol y Debbie Harry con una *Commodore Amiga 1000* (1985).	33
Fig. 6	*Legible city* (1991) Jeffrey Shaw.	33
Fig. 7	Equipo de VR para *Osmose* (1995) Char Davies.	33
Fig. 8	Imágenes de *A Computer Animated Hand* (1972) Edwin Catmull y Fred Parke.	34
Fig. 9	Uso de la animación *A Computer Animated Hand* (1972) en la película *Mundo Futuro* (1976) Richard T. Heffron.	35
Fig. 10	Integración de gráficos digitales en la película *TRON* (1982) Steven Lisberger.	35
Fig. 11	Captura de la película *Young Sherlock Holmes* (1985) Barry Levinson.	36
Fig. 12	Captura de la película *The Abyss* (1989) James Cameron.	37
Fig. 13	Detalle de la película *Terminator 2* (1991) James Cameron.	37
Fig. 14	Captura de la película *Jurassic Park* (1993) Steven Spielberg.	38
Fig. 15	Set del rodaje de la película *Oblivion* (2013) Joseph Kosinski.	39
Fig. 16	Detalle de un personaje del videojuego *Virtua Fighter* (1993).	41
Fig. 17	Detalle de un escenario del videojuego *Descent* (1995).	42
Fig. 18	Captura del Videojuego *Silent Hill* (2001) con sombras dinámicas.	43
Fig. 19	Captura del videojuego *Doom 3* (2004). Los modelos 3D tienen materiales que incluyen mapas de normales y mapas *specular*.	44
Fig. 20	Captura del videojuego *Halo 2* (2004). Los modelos 3D tienen materiales que incluyen mapas de normales y mapas *specular*.	44
Fig. 21	Captura del videojuego *Crysis* (2007) *Crytek*, donde se ve el efecto de la oclusión ambiental (AO) en los modelos 3D de la arquitectura.	45
Fig. 22	Captura para material promocional de *The Matrix Awakens* (2021) *Epic Games* / *Warner Bros.*	46
Fig. 23	Captura en una *Play Station 5* para material promocional de *Horizon Forbidden West* (2022) *Guerrilla Games*.	46
Fig. 24	Funciones fijas tradicionales del *Render Pipeline* 3D - RODRIGUES, P. (2021) *Field-configurable GPU.*	49
Fig. 25	Diferentes simulaciones de la luz en entornos con renderizado en tiempo real. (*NVIDIA*).	53
Fig. 26	Captura de la película *Monster House* (2006) Gil Kenan.	53
Fig. 27	Diagrama de flujo para la obtención de imágenes en entornos 3D. Elaboración propia.	59
Fig. 28	Tipos de polígonos de malla. Elaboración propia.	62

Fig. 29	Archivo de un cubo desarrollado en *Blender*: "v" son las coordenadas 3D de sus vértices. "vt" son las coordenadas 2D de su mapeado UV. "vn" son las normales de los polígonos que forman los vértices. "f" son los grupos y orden de los vértices de cada polígono que forma la malla. Elaboración propia.	63
Fig. 30	Visual de las Normales de los polígonos que configuran un cubo - http://courses.washington.edu/arch481/1.Tapestry%20Reader/1.3D%20Data/5.Surface%20Normals/0.default.html	64
Fig. 31	De izquierda a derecha, mismo modelo de Suzanne (*Blender*) con diferente recuento de polígonos: *low poly*, *mid poly* y *high poly* - Elaboración propia.	65
Fig. 32	Captura de *Lumen in the Land of Nanite* – https://www.youtube.com/watch?v=qC5KtatMcUw	67
Fig. 33	Detalle de las costuras (aristas resaltadas en azul) – https://download.blender.org/documentation/htmlI/ch11s06.html	69
Fig. 34	De izquierda a derecha - Mapeado UV de un guante, Textura realizada en *photoshop* de acuerdo con el mapeado UV, Modelo 3D con el material que contiene la textura ajustada al mapeado UV. Elaboración propia.	70
Fig. 35	Encaje correcto de textura en los lados de un cubo (izquierda). Diferentes errores de Mapeado UV que distorsionan la textura (derecha). Elaboración propia.	71
Fig. 36	Mapeado UV con proyección de la malla fuera de las dimensiones UV (arriba). Resultado de la aplicación en el modelo 3D donde el *software* completa el material rellenando con repeticiones de la textura (abajo). Elaboración propia.	72
Fig. 37	Detalle de la modificación de la malla en la animación de un guante que provoca la deformación de la textura. Elaboración propia.	73
Fig. 38	Diferencia de densidad de píxeles según el mapeado UV sobre un mismo modelo con la misma textura. Alta densidad de píxeles (izquierda). Baja densidad de píxeles (derecha). Elaboración propia.	74
Fig. 39	Imagen captura sin profundidad de campo (arriba) y con profundidad de campo (abajo) - https://docs.unity3d.com/es/2017.4/Manual/PostProcessing-DepthOfField.html	78
Fig. 40	Imagen capturada sin distorsión de lente (arriba) y con distorsión de lente (abajo) - https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@14.0/manual/Post-Processing-Lens-Distortion.html	79
Fig. 41	Imagen capturada sin aberración cromática (arriba) y con aberración cromática (abajo) - https://docs.unity3d.com/560/Documentation/Manual/PostProcessing-ChromaticAberration.html	80
Fig. 42	Imagen capturada sin ajustes de color - *color grading* (arriba) y son ajustes de color (abajo) – https://docs.unity3d.com/560/Documentation/Manual/PostProcessing-ColorGrading.html	80
Fig. 43	Efecto de destello de lente en *Unity* – https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@15.0/manual/shared/lens-flare/lens-flare-component.html	81
Fig. 44	Imagen capturada sin desenfoque de movimiento (arriba) y con desenfoque de movimiento (abajo) – https://docs.unity3d.com/es/2017.4/Manual/PostProcessing-MotionBlur.html	81
Fig. 45	Efecto del *anti-aliasing* – https://docs.unity3d.com/es/530/Manual/script-Antialiasing.html	83

Fig. 46	Efecto de *bloom* según variación de su parámetro de intensidad en *Unreal Engine* - https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/PostProcessEffects/Bloom/	84
Fig. 47	Imagen capturada sin AO (arriba) y con AO (abajo) – https://docs.unity3d.com/540/Documentation/Manual/script-ScreenSpaceAmbientOcclusion.html	84
Fig. 48	Misma escena con niebla generando el efecto de perspectiva aérea (arriba) y sin niebla (abajo) - https://docs.unity3d.com/2017.4/Documentation/Manual/PostProcessing-Fog.html	85
Fig. 49	Captura de *Lumen in the Land of Nanite.* Aplicación de Lumen para el cálculo de la GI en tiempo real. - https://docs.unrealengine.com/5.0/en-US/lumen-global-illumination-and-reflections-in-unreal-engine/	87
Fig. 50	Diagrama y ejemplo de un Punto de luz - Documentación de *Unity* - *Point Light* - https://docs.unity3d.com/Manual/Lighting.html	88
Fig. 51	Diagrama y ejemplo de una luz de foco - Documentación de *Unity* - *Spot Light* – https://docs.unity3d.com/Manual/Lighting.html	89
Fig. 52	Diagrama y ejemplo de una luz direccional - Documentación de *Unity* - *Directional Light* - https://docs.unity3d.com/Manual/Lighting.html	89
Fig. 53	Diagrama y ejemplo de una luz *softbox* o de pantalla- Documentación de *Unity* - *Area Light* - https://docs.unity3d.com/Manual/Lighting.html	90
Fig. 54	Efecto de luz usando un filtro (*cookie*) en *Unity* – https://docs.unity3d.com/Manual/Cookies.html	90
Fig. 55	Portada del libro *Unity 2021 - Shader and Effects* donde literalmente podemos leer *cookbook - Over 50 recipes …* (libro de cocina - Más de 50 recetas …). DORAN, J.	93
Fig. 56	Ejemplos de materiales PBR - *Adobe substance 3D*.	94
Fig. 57	Editor de Materiales *Shader Graph* de *Unity* – https://blog.unity.com/technology/introduction-to-shader-graph-build-your-shaders-with-a-visual-editor	94
Fig. 58	*Material editor UI* de *Unreal Engine* - https://docs.unrealengine.com/5.0/en-US/unreal-engine-material-editor-ui/	95
Fig. 59	Diferencia de aspecto entre un material Dieléctrico y uno Metálico - ArtStation https://cdnb.artstation.com/p/media_assets/images/images/000/706/215/large/5.jpg?1609633130	96
Fig. 60	Estructura del código de un *shader* en HLSL/CG – https://www.ighniz.com/2020/01/03/hlsl-unity3d-programacion-de-shaders-en-unity/	97
Fig. 61	Modelo de un cubo a la izquierda con un material configurado *Unlit*, y a la derecha con un material *Lit*, iluminados con luz ambiental para generar la AO - Elaboración propia.	98
Fig. 62	Diferentes efectos de una misma luz sobre la superficie de un modelo 3D según la fórmula de interpolación.	99
Fig. 63	Diferencia entre la reflexión de la luz sobre una superficie *specular* y una superficie difusa - PARK, S. y BAEK, N., (2021) "*A Shader-Based Ray Tracing Engine*"	100
Fig. 64	Detalle de Mapeado UV (líneas amarillas) sobre textura con *Shell* (espacios intermedios) - SPRITE UV	106
Fig. 65	Visual de la interfaz completa desplegada. Elaboración propia.	110
Fig. 66	Diferentes configuraciones de visualización mostrando 1, 2 o 4 objetos (de arriba a abajo: modelo 1, modelos 1 y 2, y modelos 1, 2, 3 y 4 respectivamente) con diferentes	111

	configuraciones en sus materiales bajo las mismas condiciones y al mismo tiempo. Elaboración propia.	
Fig. 67	Sección de la interfaz con las opciones para la selección o carga de diferentes mallas. Elaboración propia.	112
Fig. 68	Sección de la interfaz para el control de la iluminación. Elaboración propia.	112
Fig. 69	Sección de la interfaz que permite controlar la animación de los objetos. Elaboración propia.	113
Fig. 70	Sección de la interfaz para el control de la configuración de los materiales. Elaboración propia.	114
Fig. 71	Captura de TM01 en el entorno de *Photoshop* para diferenciar las secciones por colores (RGB) y con opacidad (A). Elaboración propia.	117
Fig. 72	Captura de TM02 en el entorno de *Photoshop* para diferenciar las secciones con opacidad (A). Elaboración propia.	120
Fig. 73	Diferentes estilos de transparencia permitidos por el motor gráfico de *Unity* (*blend modes*): Opaco, recorte, desvanecimiento y transparente. Elaboración propia.	121
Fig. 74	Transparencia de recorte (*cutout*) según valor de opacidad (*cutoff*) de izquierda a derecha y de arriba a abajo - 0.1, 0.26, 0.51 y 0.76. sobre valores de opacidad definidos en *photoshop* de 0, 0.25, 0.5 y 0.75. Elaboración propia.	122
Fig. 75	Diferencia entre modelos de transparencia “desvanecimiento” y “transparente”. Elaboración propia.	123
Fig. 76	De izquierda a derecha y de arriba a abajo - Textura TM01 modificado por color Rojo (RGBA = 255, 0, 0, 1), Verde (RGBA = 0, 255, 0, 1), Azul (RGBA = 0, 0, 255, 1) y gris neutro (RGBA = 128, 128, 128, 1) respectivamente. Elaboración propia.	124
Fig. 77	De izquierda a derecha y de arriba a abajo - Textura TM01 modificada por el valor de opacidad (Alfa - A) 0, 0.25, 0.5, 0.75 respectivamente. Elaboración propia.	125
Fig. 78	Diferencia entre el modificador de color configurado en Rojo y la aplicación de una luz Roja. Elaboración propia.	126
Fig. 79	Resaltado en rojo, los píxeles que tienen valores de R distintos a 0 sin importar el resto de los valores de RGB. Elaboración propia.	127
Fig. 80	Captura de TM03 en el entorno de Photoshop para diferenciar las secciones con opacidad (alfa). Elaboración propia.	128
Fig. 81	Efectos producidos por la textura MT03 sobre la superficie. Elaboración propia.	129
Fig. 82	Efectos producidos por la textura MT02 sobre la superficie. Elaboración propia.	129
Fig. 83	De izquierda a derecha y de arriba a abajo, aumento del valor de *smoothnes* (0 a 1) sobre la superficie R = 255 donde se puede observar las variaciones en el reflejo de la luz y de la escena que le rodea. Elaboración propia.	131
Fig. 84	Visual del cubo de muestra con la textura MT01 aplicada en su mapa de normal. Elaboración propia.	132
Fig. 85	Captura de la Textura MT04 tomada en el entorno de *photoshop* para visualizar los valores de opacidad. Elaboración propia.	134
Fig. 86	Captura de la textura TM04 aplicada al cubo por defecto de nuestro entorno con los bloques de colores de la textura enmarcados en su color correspondiente. Elaboración propia.	134
Fig. 87	Captura de la aplicación de MT01 en el mapa de relieve (izquierda), y sin ninguna textura aplicada en el Mapa de relieve (derecha). Elaboración propia.	136

Fig. 88	Captura de la aplicación de MT01 en el mapa de relieve y en el mapa de Albedo. Elaboración propia.	137
Fig. 89	De izquierda a derecha y de arriba a abajo - Modelos con HR, modelo con HG, modelo con HB y modelo sin textura aplicada en el *Height Map* y MT05 en el Albedo. Elaboración Propia.	138
Fig. 90	Textura TM02_01 creada a partir de TM02 (TM01 en escala de grises). Elaboración propia.	139
Fig. 91	Captura de la textura TM02_02 aplicada en el mapa de relieve y el Albedo. Elaboración propia.	140
Fig. 92	Modelo de cubo con la textura MT01 aplicada en su mapa de oclusión (izquierda) y cubo sin textura aplicada (derecha). Elaboración propia.	141
Fig. 93	Captura (niveles medios modificados) del modelo Suzanne con la textura MT01 aplicada en el mapa de oclusión (izquierda) y sin mapa de oclusión Derecha. Elaboración propia.	142
Fig. 94	De izquierda a derecha y de arriba a abajo - Modelo de cubo por defecto con la textura HR, HG, HB y TM01 respectivamente aplicadas en el mapa de oclusión. Elaboración Propia.	143
Fig. 95	Captura del modelo del cubo con la textura MT02 aplicada en su mapa de oclusión. Elaboración propia.	144
Fig. 96	Entorno sin iluminación. Modelo de cubo por defecto con la textura MT01 aplicada en su mapa de emisivo (izquierda) y sin ninguna textura aplicada (derecha). Elaboración propia.	145
Fig. 97	De izquierda a derecha y de arriba a abajo - Cubo por defecto con textura MT01 en el mapa de emisivo con variaciones de RGB: 255, 128, 0 respectivamente y modelo 4 de control sin mapa de *emission* (abajo derecha). Elaboración propia.	146
Fig. 98	De izquierda a derecha y de arriba a abajo - Cubo por defecto con textura MT01 en el mapa de emisivo con RGB: 255 y variaciones de I, 10, 0, -10 respectivamente, y modelo 4 de control sin mapa de *emission* (abajo derecha). Elaboración propia.	146
Fig. 99	Esquema de encapsulamiento del mapa de *Metallic*, *Oclussion*, *Detail Mask* y *Smoothness* en los canales de una misma textura. – https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@10.2/manual/Mask-Map-and-Detail-Map.html	149
Fig. 100	Esquema del encapsulamiento de los mapas de Albedo, *Normal* y *Smoothness* en una misma textura – https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@10.2/manual/Mask-Map-and-Detail-Map.html	149
Fig. 101	Modelo 3D seleccionado para la práctica. Elaboración propia.	154
Fig. 102	De arriba a abajo y de izquierda a derecha. Mismo modelo 3D con material de acero, porcelana, cristal, plástico, cuero y piedra. Elaboración Propia.	155
Fig. 103	Aplicación de un material de piedra escalado para simular un objeto de piedra de tamaño pequeño. Arriba, vista general, abajo, detalle del vientre del modelo.	156
Fig. 104	Aplicación de un material de piedra escalado para simular un objeto de piedra de tamaño colosal. Arriba, vista general, abajo, detalle del vientre del modelo.	157
Fig. 105	Detalle del modelo con la textura escalada para simular una estatua colosal (izquierda). Detalle de la modificación en el mapa de normales y mapa de oclusión ambiental para incluir detalles tallados en la roca (derecha). Elaboración propia.	158
Fig. 106	Modelo 3D con materiales diseñados en base a ambientación *Steampunk* (izquierda). Modelo 3D con materiales diseñados en base a ambientación *Cyberpunk* (derecha). Elaboración propia.	160
Fig. 107	Modelo 3D con material de creación propia. Elaboración propia.	161

1.3.- INDICE DE TABLAS

Tabla 1	Diferencia técnica en cifras entre tarjetas gráficas en 23 años. Elaboración propia.	49
Tabla 2	Relación de versiones del DirectX con su modelo de *shader*. Elaboración propia.	51
Tabla 3	Resoluciones (aspecto cuadrado) más utilizadas en la actualidad y su superficie en pixeles. Elaboración propia.	102
Tabla 4	Relación de los componentes que configuran un material PBR con el *shader Standard* de *Unity* y los parámetros de configuración. Elaboración propia.	116
Tabla 5	Desglose por secciones en sus valores RGBA de la textura TM01. Elaboración propia.	119
Tabla 6	Relación entre el valor de opacidad y el parámetro Alfa (A) de la textura. Elaboración propia.	123
Tabla 7	Color y valores de RGB para una textura de mapa de normales que no realice modificación alguna sobre la superficie del objeto 3D. Elaboración propia.	133
Tabla 8	Texturas de prueba para el mapa de *Height*. Elaboración propia.	138
Tabla 9	Relación del *shader Standard* de *Unity* y los parámetros de estudio - Verde si aplica, Naranja si aplica con condiciones, y negro si no aplica. Elaboración propia.	150
Tabla 10	Relación de los mapas de textura con el tipo de efecto que tienen sobre cada píxel y el efecto visual final sobre la superficie. Parámetros de color T - Tono, S - Saturación, L - Luminosidad, A - Alfa. Elaboración propia.	151

2.- INTRODUCCIÓN

Tras más de medio siglo utilizando equipos informáticos como herramienta para la creación de obra artística, se han producido obras y nacido movimientos de toda clase. Desde imágenes impresas con plóters, collages digitales, instalaciones interactivas, formas virtuales generativas, la creación y edición de gráficos en dos y tres dimensiones, entornos inmersivos y plataformas virtuales, *net art*, hacktivismo, cine experimental, fachadas media, arte interactivo, arte cibernético, arte electrónico o videojuegos, hasta obras generadas con inteligencia artificial, han sido objeto de experimentación y creación de obra artística en una clara expansión del Arte a través del campo tecnológico. Las concepciones sobre la tecnológica habitualmente se enfocan desde estudios técnicos y es común que las aportaciones y reflexiones procedentes de ámbitos más cercanos a las humanidades se aborden asumiendo los diferentes aspectos técnicos que se utilizan. En realidad, desde hace muchos años, autores como Winner (1979)[1] ya proponían que el concepto de tecnología implicaba características sociales, políticas y psicológicas que conllevaban una interacción en la vida cotidiana de las personas, y, por extensión, en las prácticas artísticas que poco a poco habían ido influenciándose por las nuevas posibilidades tecnológicas que han generado una constante retroalimentación en el ámbito artístico.

Los artistas utilizan todo tipo de dispositivos y soportes para mostrar nuevo contenido y esto los ha llevado a sumergirse en el ámbito tecnológico y mantener actualizadas sus habilidades dentro de este campo. Igual que un pintor conoce sus herramientas y sabe sobre óleos, pinceles y lienzos, el artista digital ha de conocer las suyas, sus opciones y posibilidades sin descartar, debido a sus constantes cambios y avances, la experimentación para alcanzar nuevos límites. En todos los espacios artísticos se ha dado la confluencia entre la práctica

[1] WINNER, L., CARDÍN, A., FOT, R., GILI, G. (1979) "La técnica incontrolada como objeto del pensamiento político", Barcelona, España.

artística y las diferentes tecnologías, y en los nuevos lenguajes los cambios y las posibilidades se centran en la simbiosis entre la tecnología y el arte (Gianett, 1998)[2].

Las herramientas y soportes tecnológicos han estado, y están actualmente, en constante evolución. Los avances exponenciales en hardware, las constantes actualizaciones y el desarrollo del software, que permiten nuevas posibilidades de expresión visual y técnica, resultan ser términos relacionados con la práctica artística, aunque parezcan más propios del área de informática. Estos conceptos se interconectan y están más presentes en diferentes aspectos de nuestras vidas, ya sea el profesional o el personal, en una sociedad cada vez más digitalizada, y lógicamente también ha dejado sentir su influencia en el campo de las artes.

La creación artística siempre ha visto posibilidades en las nuevas tecnologías explorando sus límites y nuevas formas de expresión, a la vez que la tecnología retroalimentaba al arte a través de nuevos soportes, dispositivos y posibilidades en constante crecimiento, un binomio que, en la célebre cita atribuida a John Lasseter, plantea el eje fundamental del motor creativo de la productora Pixar: *"el arte desafía a la tecnología, y la tecnología inspira el arte"*. Por ejemplo, la fotografía, que inicialmente era un trabajo desarrollado por químicos, en la actualidad nadie duda de que es un arte, aunque las cámaras digitales las desarrollan ingenieros. No obstante, también hay autores críticos con la tecnología como Andrew Keen, autor de "*The Cult of the Amateur: How Today's Internet is Killing Our Culture*", que defiende la depreciación del arte fotográfico reflexionando precisamente sobre el avance tecnológico, en tanto que la cámara digital ha destruido el arte de la fotografía debido a que cualquier persona se erige como fotógrafo, precisamente por la facilidad de la tecnología para suplir la formación y habilidad del usuario: *"la fotografía se ha convertido en algo tan fácil, que la gente no piensa que una foto tiene algún valor intrínseco".*

[2] GIANETT, C. (1998) "Algunos mitos del final del milenio. Contra la trivialización de la tecnocultura", Catàleg de la Mostra d'Arts Electròniques, Dept. de Cultura de la Generalitat Catalana, Centre d'Art Santa Mònica, Barcelona.
http://www.arteuna.com/talleres/lab/ediciones/libreria/Giannetti_AlgunosMitos.pdf

El debate tecnológico y las fronteras entre la técnica y el arte también están presentes en el ámbito de la producción audiovisual desarrollada en entornos 3D combinada con renderizado en tiempo real. Es habitual que, al pensar en este tipo de producciones, se imaginen programadores escribiendo código sentados frente a una pantalla de ordenador, pero las herramientas digitales para la producción de este tipo de contenido son cada vez más completas, intuitivas y visuales, dejando de lado los editores de texto y código. Eso permite al artista crear contenido sin tener conocimientos en otras áreas, y aunque la programación sigue siendo parte del desarrollo técnico, existen herramientas para simplificar su implementación durante la creación del contenido. Además, terminando el primer cuarto del siglo XXI, el artista digital ya está familiarizado (o debería estarlo) con la programación, como por ejemplo los lenguajes HTML, CSS y JavaScript para crear contenido web, o ActionScript para la creación de animaciones interactivas, pudiendo dar un paso más y profundizar en la programación relacionada con el desarrollo de este tipo de producciones.

Los artistas digitales que se adentran en la creación para este tipo de contenido han de conocer y manejarse en las distintas disciplinas que intervienen para la creación de dicho contenido, y, por lo tanto, los programas que intervienen en este tipo de producciones, ya que no todo el trabajo se desarrolla dentro del propio motor gráfico. Desde el modelado 3D, la animación de mallas virtuales, el texturizado a través de distintos editores en base a diferentes enfoques, hasta la preparación y ejecución del audio, forman parte en un largo proceso de creación en este tipo de proyectos artísticos y requieren del manejo de programas específicos y la práctica y habilidades con herramientas propias de cada uno. La formación actual en artes trata, en muchos casos, estas habilidades muy superficialmente, en ocasiones sin explorar más allá de las artes digitales bidimensionales, y es normal que no se lleguen a relacionar ni se desarrolle la práctica en este tipo de producciones.

La creación de contenido en base a entornos con renderizado en tiempo real ha evolucionado desde otros medios de producción como la animación 3D, el cine o los videojuegos, y ha heredado terminología propia de estos, pero también ha generado la suya propia. Uno de los elementos dentro de este campo que más ha evolucionado y modificado el aspecto visual final son los materiales aplicados a los modelos tridimensionales virtuales. En la actualidad, los materiales PBR (*Physically Based Rendering*) permiten alcanzar altos niveles de realismo o la

creación de diferentes estéticas según su configuración, y están compuestos de múltiples texturas que pueden ser modificadas para crear nuevo contenido. Estos materiales digitales permiten diversas posibilidades de renderizado en tiempo real. Es importante remarcar que esta investigación se aborda desde un punto de vista transdisciplinar que combina aspectos tecnológicos muy específicos en cuanto a motores gráficos, modelos y entornos 3D en tiempo real y creación artística para abordar un enfoque eminentemente práctico y aplicado a las posibilidades dentro de la creación artística digital tratando de traducir procesos, términos y componentes informáticos a un lenguaje más propio de las Artes.

Los materiales PBR se utilizan en diferentes disciplinas como la arquitectura o la animación digital, sin embargo, en los entornos con renderizado en tiempo real, su uso ha dependido de la evolución tecnológica, y juegan un papel fundamental en la creación de este tipo de contenido (como veremos más adelante) al permitir un gran detalle en los elementos de la escena reduciendo el consumo de recursos técnicos, proporcionando un flujo en ejecución más consistente y eficiente. Estos materiales pueden producir resultados más precisos y realistas que otros modelos de renderizado. De hecho, la mayoría de los motores gráficos de última generación como *Unreal Engine* o *Unity*, utilizan sistemas de renderizado PBR como su estándar de calidad. Por lo tanto, un artista digital que quiera crear contenido de alta calidad para estos motores debe conocer los materiales PBR y cómo utilizarlos.

Estos materiales virtuales tienen como objetivo final lograr un mayor realismo en la apariencia de los objetos renderizados. En lugar de utilizar modelos de iluminación predefinidos, los materiales PBR utilizan modelos físicos y matemáticos para simular la forma en que la luz se refleja, refracta y se absorbe en diferentes superficies. Esto significa que los materiales PBR tienen en cuenta la rugosidad, el índice de refracción, la metalicidad y otros aspectos físicos del material para producir un resultado más realista. Ese resultado se obtiene de la aplicación de múltiples imágenes, denominadas texturas, a un mismo material, y dichas texturas pueden ser modificadas y editadas por el artista para obtener diferentes resultados más allá de la simulación de la realidad en entornos virtuales.

Uno de los motivos para estudiar en profundidad este sistema de materiales es la necesidad de salir del estándar de simulación de la realidad y proporcionar soporte para la creación en

Artes. Es habitual que el desarrollador que está generando contenido 3D recurra a materiales pregenerados estándar del software que se está utilizando o acceda a librerías y repositorios online, que nos proporcionan materiales con aspectos predefinidos como madera, piedra o cuero, entre otros. También existen las "recetas" en las que podemos copiar los resultados obtenidos por otro autor, o podemos recurrir al "ensayo y error", modificando los parámetros que configuran un material PBR hasta obtener un resultado aproximado que se ajuste a nuestras necesidades. Un conocimiento más profundo de las diferentes texturas, su configuración, y su influencia y repercusión individual sobre el material, proporcionan al desarrollador un mayor control sobre el resultado final y la posibilidad de crear nuevo contenido.

El trabajo con motores gráficos y las habilidades artísticas de aplicación en los mismos no son sólo una línea de creación digital en el ámbito artístico. En los últimos años, con la posibilidad de acceder al software de forma gratuita con licencias específicas, el *open source*, o la expansión de las Realidades Extendidas, entre otros factores, se ha disparado la demanda de perfiles especializados a nivel profesional y laboral. Esto también ha generado la necesidad por parte del alumnado de recibir formación más específica en este tipo de creaciones, ya que no sólo se aplica en el ámbito de las artes audiovisuales interactivas: por ejemplo, el modelado 3D está relacionado con la impresión 3D, el escaneado 3D o la digitalización de patrimonio, el marketing y la publicidad, o la arquitectura.

Finalmente, existe un interés personal en este estudio. Ya en los años noventa, con libros como "Programación Gráfica en C" o "Realidad Virtual - Creaciones y desarrollo" entre otros, ya experimentaba con los gráficos por ordenador y los entornos 3D. Recuerdo las horas invertidas en copiar código para aprender, pero también las horas de experimentar con lo aprendido y los intentos por crear algo nuevo, una y otra vez, probando y probando, siempre intentando llegar más allá, o los disquetes, CDs y DVDs que venían de regalo con la revista mensual, y las horas de trabajo con la demo del programa de turno, revisando menús y probando los comandos hasta aprender a usarlos para sacarle el máximo partido a las versiones limitadas de los diferentes softwares que salían al mercado. Internet ha democratizado el acceso al conocimiento. Existen guías, blogs, tutoriales y videotutoriales de toda clase, pero también es cierto que, al menos desde mi punto de vista, se ha perdido en

parte la esencia investigadora y experimental de aquella época, como ya se ha comentado, al recurrir a las recetas y guías prediseñadas para obtener resultados estandarizados. Este trabajo pretende, en su desarrollo, retomar un poco de aquel espíritu, y, a través del análisis y la práctica con los materiales PBR y sus texturas, proporcionar un mayor conocimiento de estas y de esta parte tan relevante en el aspecto final de las obras desarrolladas en motores gráficos.

Esta investigación es un primer paso para la composición de una base transversal que unifique un contenido disperso en diferentes disciplinas de las artes, pero también un punto de partida desde el que continuar con la investigación ampliando en un futuro próximo el abanico más allá de los materiales PBR, y unificar, conectar, y mantener actualizado el conocimiento relacionado con la creación en entornos 3D con renderizado en tiempo real.

2.1.- HIPOTESIS

Actualmente la creación de entornos 3D con renderizado en tiempo real se ha vuelto cada vez más relevante en diferentes ámbitos, especialmente en el de las Artes Digitales, cuya aplicación abarca desde las producciones cinematográficas hasta el ocio interactivo, pasando por aplicaciones médicas y educativas, entre otras. No obstante, el conocimiento para el correcto desarrollo de obras dentro del ámbito artístico se encuentra muy disperso en las diferentes disciplinas que lo componen. Este hecho dificulta la formación para los artistas y diseñadores que desean incursionar en este campo de conocimiento precisamente por la fragmentación de conocimiento. Es por ello por lo que el tema de investigación de esta tesis se centra en la creación de entornos 3D con renderizado en tiempo real desde el ámbito artístico.

A través del estudio y análisis del proceso de creación o flujo de trabajo para la obtención de cada fotograma, y de los materiales dinámicos, siempre desde una perspectiva del conocimiento en Artes, se puede generar una base para la creación de una guía de iniciación que aborde los componentes y parámetros necesarios para la creación de entornos 3D con renderizado en tiempo real.

Esta hipótesis se sostiene en la necesidad de investigar y componer una base aplicada con un perfil artístico-tecnológico que facilite la comprensión y aplicación adecuada de materiales dinámicos en la creación de entornos 3D. Esta premisa es crucial para lograr un resultado satisfactorio en el proceso de renderizado en tiempo real, tanto visual como en consumo de recursos técnicos, aspectos de excelencia técnica y sostenibilidad de la práctica artística contemporánea, que son fundamentales. Asimismo, esta hipótesis de investigación parte de que el conocimiento en Artes aporta una perspectiva valiosa para el desarrollo de entornos 3D con un enfoque estético y creativo que complementa y enriquece cualquier producción, aunque no necesariamente tenga fines artísticos, como un valor añadido diferencial.

La hipótesis a través de la cual generar los conocimientos transversales y una base de perfil artístico y tecnológico tiene como fin que este estudio transversal sea de utilidad para artistas

y diseñadores que se sumerjan en la creación de contenido en entornos 3D con renderizado en tiempo real. Además, se espera que la exploración de los componentes y parámetros que intervienen en el proceso de creación de entornos virtuales, así como la experimentación con los mapas de texturas necesarios para la configuración de materiales PBR (*Physically Based Rendering*) y la interpretación de resultados en base a parámetros artísticos, permita un mayor entendimiento del proceso creativo en este campo para un público no necesariamente especializado y fomente, por tanto, la innovación en la creación de entornos 3D con renderizado en tiempo real.

2.2.- OBJETIVOS

OBJETIVO PRINCIPAL

Generar las bases para la creación de una guía de iniciación al desarrollo de entornos 3D con renderizado en tiempo real partiendo del estudio y análisis de materiales dinámicos desde una perspectiva del conocimiento en Artes, con la intención de mejorar y ampliar los procesos de creación de este tipo de contenido.

OBJETIVOS SECUNDARIOS

1.- A través del análisis de la literatura científica relacionada con la creación en entornos con *render real-time*, establecer un contexto histórico que justifique un punto de partida desde los entornos interactivos con renderizado en tiempo real.

2.- Explorar y analizar los componentes y parámetros que intervienen actualmente en el proceso de creación de imágenes en entornos virtuales.

3.- Experimentar con los mapas de texturas necesarios para la configuración de materiales PBR (*Physically Based Rendering*) y definir sus características en su aplicación.

4.- Analizar, ordenar e interpretar los resultados obtenidos en base a parámetros artísticos tras el estudio y la experimentación previa con texturas y materiales.

5.- Realizar una práctica en la que se apliquen materiales PBR sobre un objeto 3D con diferentes finalidades y efectos que evidencien su versatilidad y utilidad en su aplicación.

2.3.- CONTEXTO HISTÓTRICO

2.3.1.- GRÁFICOS DIGITALES

En 1965, el alemán George Nees (Fig. 1), estudiante de matemáticas, física y filosofía en Stuttgart, realizó la primera exposición de gráficos generados por ordenador en febrero de ese año en la misma ciudad, animado por Max Bense, profesor de filosofía y teoría científica en la Escuela Superior Técnica de Stuttgart. El autor reivindicaba que su realización era genuinamente robótica, matemática y digital.

Nees, junto a Frieder Nake (Fig. 2), estudiante de matemáticas en Stuttgart, y Michael Noll (Fig. 3), en Estados Unidos (Las tres N), y junto al CTG (*Computer Technique Group*) (Fig. 4), fundado en 1966 en Japón, sentaron las bases para un cambio de estética postmodernista que se centraba en las características y las posibilidades de las computadoras y su imaginario futurista.

En 1968 se celebra la primera exposición colectiva sobre arte digital que tuvo por título *Cybernetic Serendipity* en el *Institute of Contemporary Art* de Londres. En ese momento, los gráficos por ordenador permitían a los creativos y artistas trabajar en dos frentes que dieron lugar a la dualidad y confrontación entre la creación de imágenes a través de algoritmos, siguiendo la línea de la obra de Nees y Nake, y la que se basa en la manipulación de imágenes previamente capturadas, desarrollada principalmente en Japón y Estados Unidos.

Durante la década de 1970, los desarrolladores de gráficos por ordenador eran científicos del área de la informática o dependían de técnicos especializados que conocieran el lenguaje con el que funcionaban los ordenadores de la época. A finales de la década de los 70 y principios de los 80 aparece el ordenador personal, y posteriormente los sistemas operativos (OS) con interfaz de usuario, facilitando el acceso y el uso de estos dispositivos a través del teclado y el ratón.

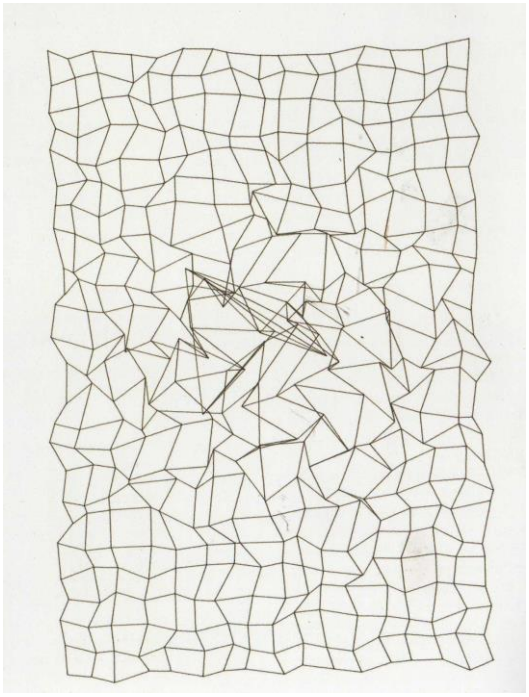


Fig. 1. K27 (1965) Georg Nees.

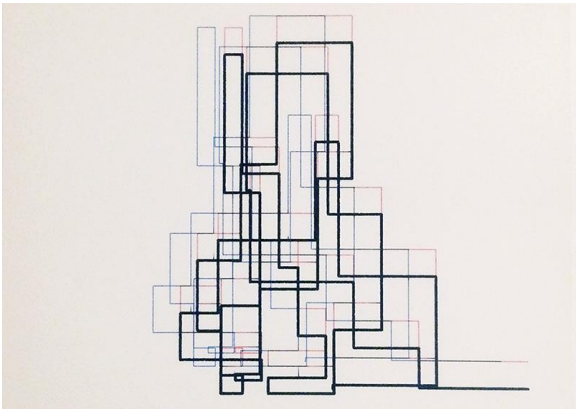


Fig. 2. *Rectangular Random Polygon* (1965) Frieder Nake.

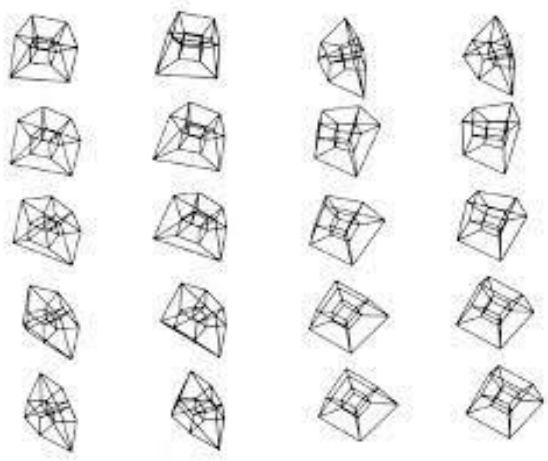


Fig. 3. *Projection of a four-dimensional hypercube* (1965) Michael Noll.

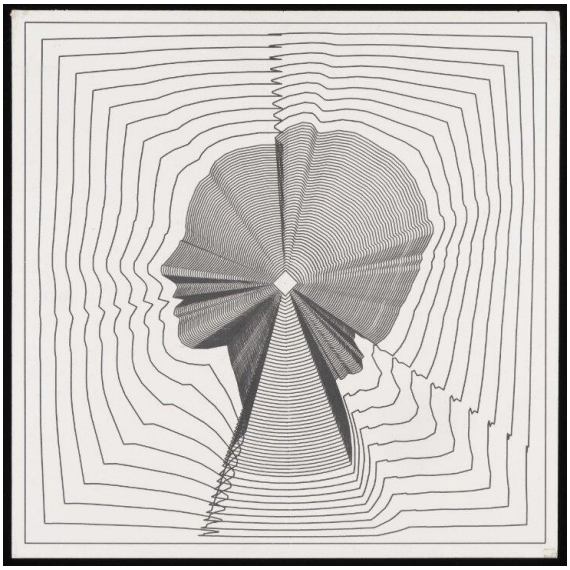


Fig. 4. *Return to Square* (1968) Masao Komura, Makato Othake (CTG).

En 1985, la empresa *Commodore* lanza al mercado el *Commodore Amiga 1000* que incluía un *software* de dibujo rudimentario con el que Andy Warhol (Fig. 5), artista reconocido y consagrado, realizó 28 obras, llegando a salir en televisión promocionando dicho ordenador[3],

[3] Andy Warhol pinta a Debbie Harry en un equipo Amiga. Recuperado de: https://youtu.be/wLvTG5hwa1A

lo que dejaba claro que las obras realizadas mediante un ordenador podían tener la misma categoría que las que se realizaban con otras herramientas o medios. Los primeros *softwares* para la edición y creación de gráficos 3D datan también de esta época, pero eran muy rudimentarios y su manejo seguía basándose en la programación.

La tecnología era accesible y estaba evolucionando a gran velocidad, y también lo hacían los programas para el desarrollo de gráficos digitales, tanto 2D como 3D. Estos últimos también tuvieron cabida en el mundo del arte e influyeron a artistas como Jeffrey Shaw, arquitecto y escultor que dio el salto a la videoinstalación. Generó uno de los primeros entornos interactivos multimedia *Legible City* (1991) (Fig. 6), en el que el usuario, montado en una bicicleta estática, podía recorrer una ciudad virtual[4].

En 1995, Char Davies, titulada en artes y pintora en sus comienzos, miembro fundador de la empresa *Softimage,* desarrolladora del *software Softimage 3D,* utilizado en muchas populares producciones cinematográficas para integrar gráficos 3D, presentó una de las primeras experiencias de Realidad Virtual, *Osmose* (Fig. 7) en el VI Simposio Internacional de Arte Electrónico (ISEA).

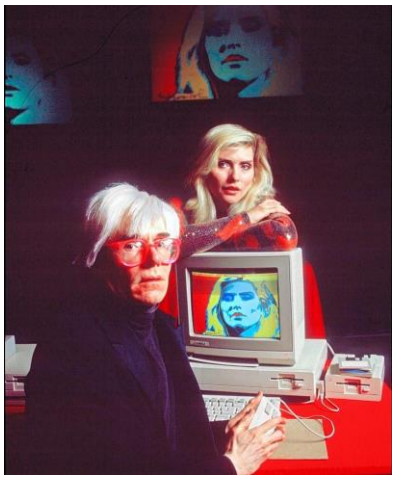


Fig. 5. Andy Warhol y Debbie Harry con una *Commodore Amiga 1000* (1985).

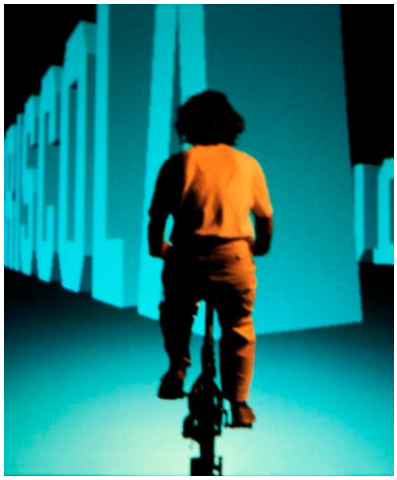


Fig. 6. *Legible city* (1991) Jeffrey Shaw.

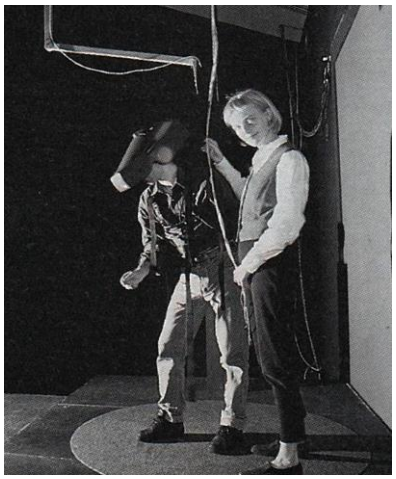


Fig. 7. Equipo de VR para *Osmose* (1995) Char Davies.

[4] *Legible City* - Jeffrey Shaw. Recuperado de: https://youtu.be/61l7Y4MS4aU

2.3.2.- EVOLUCIÓN DE LOS GRÁFICOS 3D

La evolución de los gráficos tridimensionales está ligada a una disciplina artística consolidada como es el cine, en la que se han integrado los recursos que proporciona la creación de gráficos 3D. Los efectos especiales son casi tan antiguos como esta disciplina artística: actores disfrazados, robots animatrónicos o maquetas, modelos a escala grabados en *stopmotion* o *go motion*, o la edición analógica de fotogramas, todos ellos efectos que en la actualidad han sido sustituidos por los efectos visuales realizados en la postproducción digital (VFX).

Edwin Catmull, que posteriormente sería uno de los co-fundadores de *Pixar*, realiza en 1972 la primera animación 3D por ordenador de la historia[5] (Fig. 8), pero esta representación animada en pantalla de un modelo tridimensional creado digitalmente y utilizado posteriormente en la película *Mundo Futuro*[6] (Fig. 9), quedaba aún muy lejos de la integración de un gráfico en 3D animado en la escena, como se puede ver en la película *TRON*[7] (Fig. 10), en la que los gráficos 3D son parte de la acción de los personajes y no un elemento decorativo.

Fig. 8. Imágenes de *A Computer Animated Hand* (1972) Edwin Catmull y Fred Parke.

[5] *A computer animated hand* (1972) Edwin Catmull y Frederic Parke - https://www.youtube.com/watch?v=lkHpoCveh4c
[6] Del original *Futureworld* (1976) Richard T. Heffron - https://www.youtube.com/watch?v=QfRAfsK5cvU
[7] *TRON* (1982) Steven Lisberger - https://www.youtube.com/watch?v=Tm4i6D3XXBQ

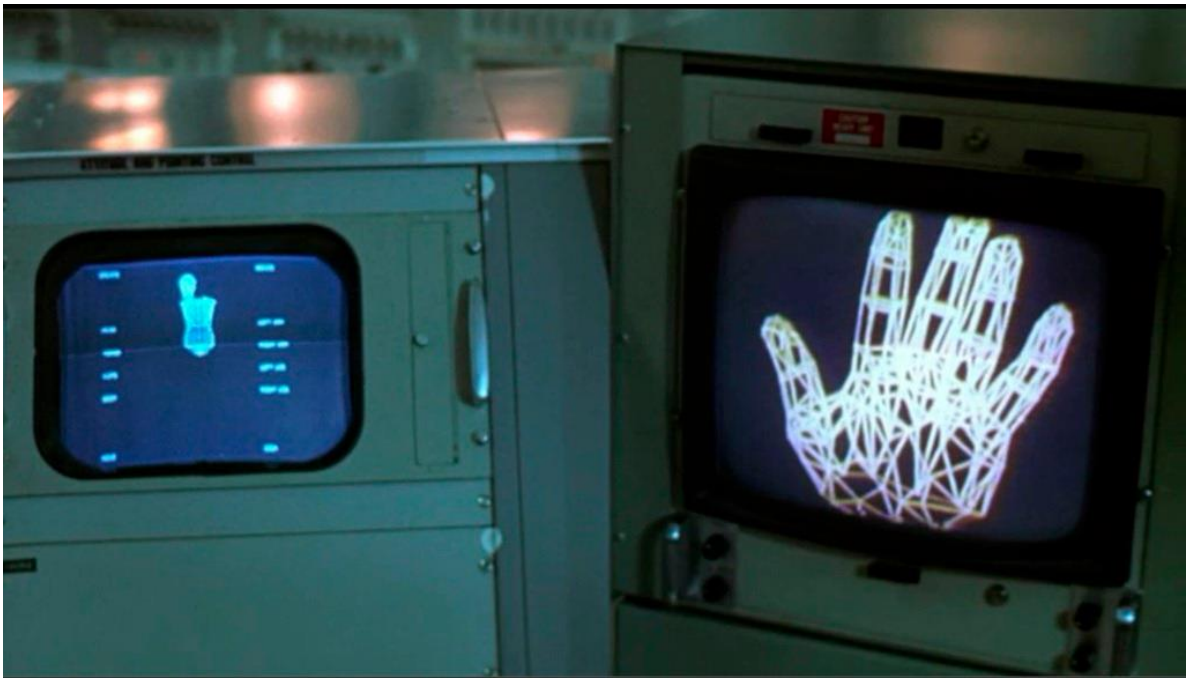


Fig. 9. Uso de la animación A Computer Animated Hand (1972) en la película Mundo Futuro (1976) Richard T. Heffron.

Fig. 10. Integración de gráficos digitales en la película TRON (1982) Steven Lisberger.ig. 9. Uso de la animación A Computer Animated Hand (1972) en la película Mundo Futuro (1976) Richard T. Heffron.

En 1985, en la película *El secreto de la pirámide*[8] vemos el primer personaje por gráficos 3D integrado en una escena con interacción con personajes reales y el entorno de la escena buscando un efecto fotorrealista (Fig. 11), efecto creado por los estudios de *Lucasfilm*, donde trabajaba John Lasseter, futuro fundador de *Pixar*.

Fig. 11. Captura de la película *Young Sherlock Holmes* (1985) Barry Levinson.

En *Abyss*[9] se integran gráficos 3D por ordenador como parte de la escena, en la que generan una simulación digital de físicas de fluidos junto a la recreación digital de un material, emulando las propiedades visuales del agua (Fig. 12), efecto digital que James Cameron utilizaría posteriormente para crear el famoso androide T-1000 en la película *Terminator 2*[10], sirviéndose de texturas dinámicas que componen la apariencia adaptable de un androide con capacidad de duplicar apariencias y que, por momentos, su aspecto, gracias al material, refleja la escena real que lo rodea (Fig. 13).

[8] Del original *Young Sherlock Holmes* (1985) Barry Levinson - https://www.youtube.com/watch?v=iB9KoJ79tGY
[9] Del original *The Abyss* (1989) James Cameron - https://www.youtube.com/watch?v=XSLQ_94R4sc
[10] *Terminator 2* (1991) James Cameron - https://www.youtube.com/watch?v=Aq5ydeWWr4A

Fig. 12. Captura de la película *The Abyss* (1989) James Cameron.

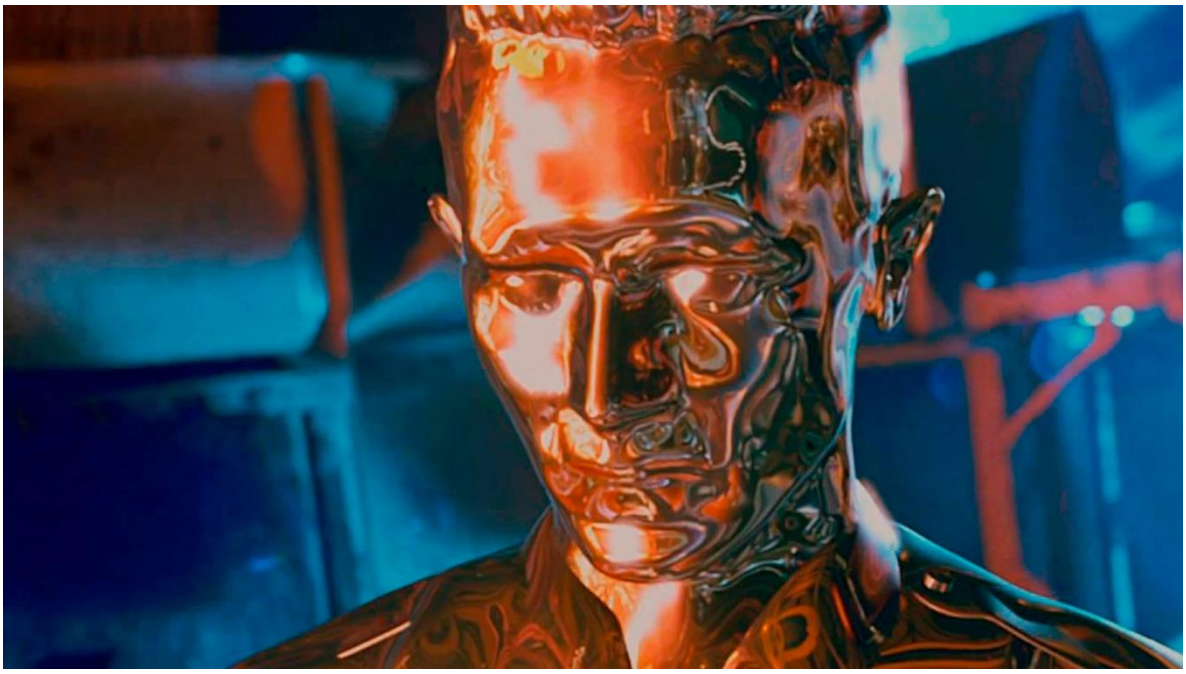


Fig. 13. Detalle de la película *Terminator 2* (1991) James Cameron.

Con *Jurassic Park*[11], los efectos visuales 3D, tanto en modelado como en texturizado, alcanzan niveles fotorrealistas (Fig. 14). A partir de este momento, los gráficos 3D en el cine

[11] *Jurassic Park* (1993) Steven Spielberg.

incrementan su complejidad, calidad y resolución, y aparecen programas y tecnologías cada vez más potentes. Sus resultados evolucionan junto a los avances tecnológicos en el sector de los gráficos por ordenador y su aplicación en el ámbito audiovisual.

Algunos hitos importantes en el ámbito del cine relacionado con los entornos con renderizado en tiempo real son, por ejemplo, el rodaje de la película *Avatar*[12]. Los actores pasaron dos años de rodaje en un estudio donde, a través de sistemas de MOCAP[13], se capturaron sus movimientos para todas y cada una de las secuencias de la película en las que intervienen personajes 3D. Posteriormente, se utilizaron entornos con renderizado en tiempo real donde podían verse una y otra vez los movimientos de los actores previamente capturados sobre modelos 3D hasta obtener la mejor toma, utilizando lo que se denominó como cámara virtual, dispositivo físico que es capaz de modificar la posición de la cámara en el espacio virtual en tiempo real, permitiendo visualizar una aproximación de la toma final y fijarla para su posterior postproducción y renderizado.

Fig. 14. Captura de la película *Jurassic Park* (1993) Steven Spielberg.

[12] *Avatar* (2009) James Cameron.
[13] Dispositivos de captura de movimiento a través de los cuales se digitalizan los movimientos de actores y objetos.

En 2013, durante el rodaje de la película *Oblivion*[14], se utilizaron pantallas en las que se proyectaban fondos tridimensionales renderizados, que, además de servir como fondo para la escena, generaban la iluminación sobre los actores y los elementos de la escena (Fig. 15).

El cine y los medios audiovisuales convencionales utilizan efectos en base a gráficos 3D previamente producidos para integrarlos en la postproducción de la obra. Esto se traduce en que el consumo de recursos es muy elevado, entendiéndose éstos como los equipos necesarios para dicho renderizado y el tiempo que tardan en producir la imagen. Un solo fotograma de una película puede tardar días en renderizarse, y posteriormente, una vez obtenida la imagen, se integra en la producción final en la fase de postproducción. Las grandes productoras de este tipo de efectos digitales, como *Weta Digital*, adquirida a principios de 2022 por *Unity*[15], utilizan múltiples equipos interconectados para repartir la carga de trabajo y los procesos a realizar para el renderizado de complejas escenas. Estos sistemas son lo que se conocen como Granjas de Renderizado (*Render Farms*).

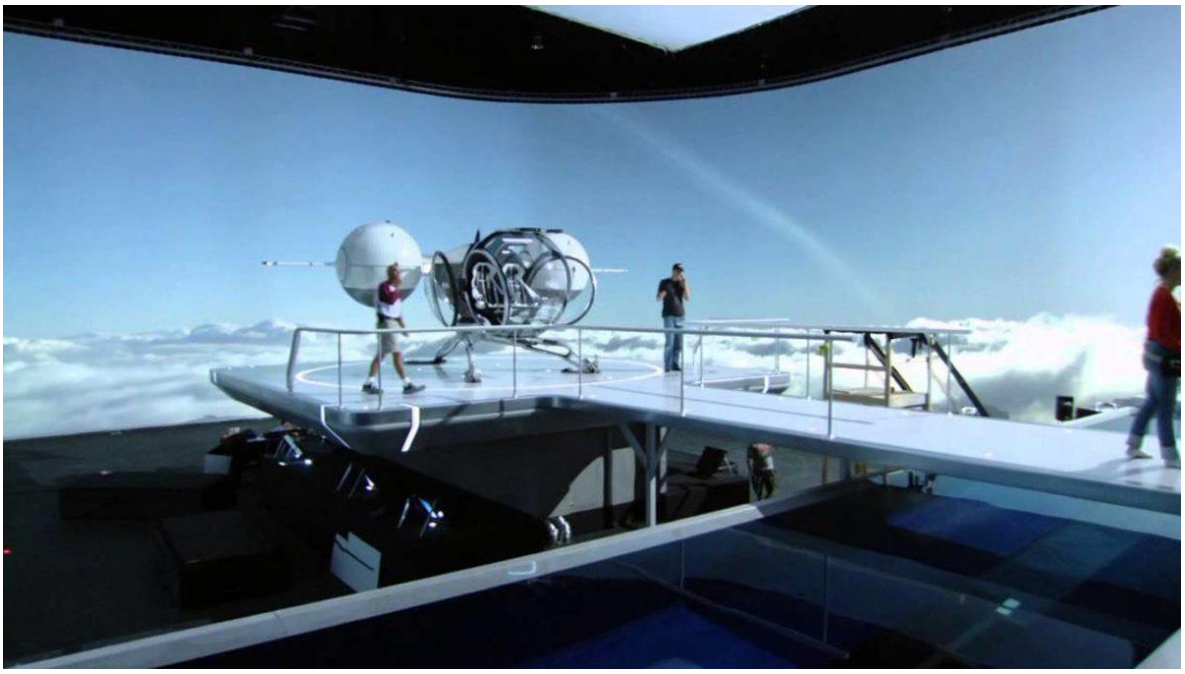


Fig. 15. Set del rodaje de la película *Oblivion* (2013) Joseph Kosinski.

[14] *Oblivion* (2013) Joseph Kosinski.
[15] Véase el artículo (27-02-2022) URL: https://www.europapress.es/economia/noticia-Unity-pagara-mas-1400-millones-weta-digital-estudio-fundado-peter-jacks/on-20211110090855.html

2.3.3.- GRÁFICOS 3D Y *RENDER REAL-TIME*

La evolución tecnológica alcanza también a los motores gráficos, que son la base para la generación de entornos virtuales con renderizado en tiempo real; motores centrados hasta ese momento en el desarrollo de espacios bidimensionales. En este punto es fundamental valorar el impacto e influencia del desarrollo de las aplicaciones interactivas más populares, los videojuegos, en la evolución de los gráficos 3D en entornos con renderizado *real-time*.

Los desarrolladores de estas aplicaciones interactivas trabajan con modelos 3D mostrados por pantalla a través del renderizado en tiempo real, consiguiendo una gran diversidad de aplicaciones, entre ellas, el desarrollo de videojuegos con gráficos 3D, una disciplina creativa y consolidada que ha crecido a partir del lenguaje del cine en muchos aspectos sensoriales, narrativos y técnicos. La tecnología que lo hace posible, tanto *hardware* como *software*, sigue evolucionando a fecha de hoy, permitiendo la creación de aplicaciones en realidad virtual (VR), aumentada (AR) o mixta (MR), las denominadas como realidades extendidas (XR).

A diferencia de la producción digital de fotogramas para Cine, sensiblemente más costosos a nivel de recursos y tiempo de producción, los fotogramas renderizados en tiempo real (*real-time*) deben generarse a una velocidad de entre 60 y 90 fotogramas por segundo, dependiendo del equipo y de la aplicación en ejecución (por eso suelen venir acompañadas de una serie de requisitos técnicos mínimos).

Se considera que los fotogramas por segundo (fps) mínimos para generar el efecto óptico de movimiento es de 15 cuadros por segundo. Lo habitual en las producciones audiovisuales convencionales es trabajar con entre 24 y 30 fps, pero en el campo de los entornos con renderizado en tiempo real se suele exigir una tasa de entre 60 y 90 fps, debido a la cantidad de modificaciones y cambios que hay entre cada imagen. La necesidad técnica de no sobrecargar los equipos y que las aplicaciones estén al alcance de los usuarios, obliga a los creadores de este tipo de contenido a ingeniárselas para apurar al máximo los recursos que ofrecen los diferentes dispositivos y herramientas para conseguir los resultados deseados.

Para avanzar en la evolución gráfica de los entornos 3D con renderizado en tiempo real, vamos a continuar el estudio tomando como referencia el campo que engloba el desarrollo de videojuegos, ya que se trata de un campo de aplicación de los entornos 3D. El estudio de estos entornos favorece el estudio de múltiples formatos y objetivos diferentes, aspecto vital en la hipótesis de partida, a la vez que son responsables de la democratización y evolución de las tecnologías relacionadas con la creación de este tipo de entornos virtuales.

Actualmente existe cierta discrepancia sobre cuándo aparece el primer videojuego con gráficos 3D, cuyo eje pivota sobre la aparición de los primeros vectores en animación, si se debe considerar a partir de la primera creación que integra polígonos, si sus elementos se integraban con imágenes planas en un entorno tridimensional, o si era 2.5D o 3D.

En el desarrollo de esta investigación, que se enfoca en la parte del texturizado de los elementos en la escena, comenzaremos a tratar el tema con el videojuego *Virtua Fighter*[16], en el que pueden verse claramente los polígonos que configuran los modelos 3D y cuyo texturizado, proceso por el cual dichos polígonos reciben un color o textura, se basa en dar un color plano a los polígonos del objeto tridimensional[17] (Fig. 16).

Fig. 16. Detalle de un personaje del videojuego *Virtua Fighter* (1993).

[16] YU SUZUKI (1993) Virtua Fighter (SEGA) [videojuego] SEGA AM2.
[17] Gameplay *Virtua Fighter* (27-02-2022) URL: https://www.youtube.com/watch?v=rCKKe25P4l4

En *Descent*[18], también en pugna por el título de primer videojuego en 3D, podemos ver cómo los desarrolladores gráficos aplicaron imágenes 2D a los polígonos[19], siempre dentro de los parámetros de la potencia gráfica del equipo medio de la década de los 90 (Fig. 17).

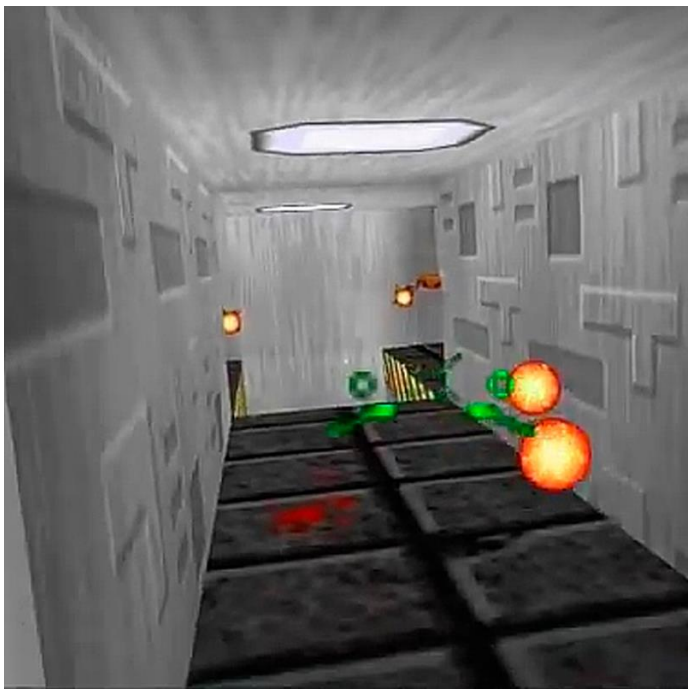


Fig. 17. Detalle de un escenario del videojuego *Descent* (1995).

El uso de imágenes 2D para dar textura a los polígonos implicaba la aparición de términos como mapeado UV (véase el apartado correspondiente en este mismo texto) y mapas de textura.

Inicialmente, la iluminación de la escena era homogénea y, o bien era general sin sombreado, o se introducía a través de texturas o procesos como el *lightmapping*[20], o sólo existía un punto de luz general para toda la escena, sombreando cada polígono independientemente.

El siguiente paso fue modificar el tono de los polígonos de las mallas individualmente, calculando la influencia de la luz a través de sus vértices (iluminación per-vertex), generando

[18] *Descent* (PC, Mac OS, Linux, PS, AA)[videojuego]. (1995) Parallax Software.
[19] *Gameplay Descent* (27-02-2022) URL: https://www.youtube.com/watch?v=2RAbnKUorkQ
[20] Proceso por el cual se calculan las luces durante la creación de la aplicación y se guardan en uno o varios mapas de texturas que se aplican en la escena durante la ejecución.

degradados de luces y sombras que mejoraban la simulación de iluminación, pasando de un *Flat shading* a un *Gouraud shading* (véase la sección correspondiente en este mismo texto).

Silent Hill 2[21], junto a otros juegos de la nueva generación de videoconsolas, marcan un nuevo hito en el desarrollo con la utilización de sombras dinámicas generadas en tiempo real según la posición del foco de luz en el espacio, como podemos ver en la captura, en la que el maniquí proyecta una sombra sobre la pared respecto del origen de la luz (la linterna) controlada por el jugador (Fig. 18).

Fig. 18. Captura del Videojuego *Silent Hill* (2001) con sombras dinámicas.

En 2004, al tomar la luz un mayor protagonismo por ser un elemento dinámico de la escena, los modelos, y sobre todo sus materiales, dieron un nuevo salto, apareciendo el mapa de Normales y el mapa *Specular*/*Highlights*, que simulan volumen y reflexión de la luz respectivamente sobre la superficie del objeto, utilizando como base la iluminación de la escena, tal y como se puede ver en los videojuegos *Doom 3*[22] (Fig. 19) o *Halo 2*[23] (Fig. 20). Estos mapas de texturas hacían que las superficies de los modelos 3D tuvieran una respuesta

[21] *Silent Hill* 2 (PC, PS2, Xbox)[videojuego]. (2001) *Tema Silent*.
[22] *Doom 3* (PC, Mac OS)[videojuego]. (2004) *id Software*.
[23] *Halo 2* (PC, Xbox) [videojuego]. (2004) *Bungie Studios*.

más realista a la luz de la escena y permitían, además, incluir más detalles en el modelo sin incrementar el número de polígonos de la malla.

Fig. 19. Captura del videojuego *Doom 3* (2004). Los modelos 3D tienen materiales que incluyen mapa de normales y mapa *specular*.

Fig. 20. Captura del videojuego *Halo 2* (2004). Los modelos 3D tienen materiales que incluyen mapa de normales y mapa *specular*.

En 2007, poco después del lanzamiento de la nueva generación de videoconsolas, *Xbox 360* (2005) y PS3 (2006), se aplica en los entornos con renderizado en *real-time* la oclusión ambiental (AO), efecto de iluminación que simula la atenuación de la luz en las zonas donde ésta estaría bloqueada u ocluida. Un claro ejemplo es el videojuego *Crysis*[24], que utiliza *Screen Space Ambient Occlusion*, que, a diferencia del mapa de textura de oclusión ambiental estático de un material, éste se calcula para toda la escena en cada fotograma durante la ejecución de la aplicación, tal y como muestra la captura (Fig. 21), en la que ciertas zonas de la arquitectura quedan ensombrecidas (resaltadas en azul).

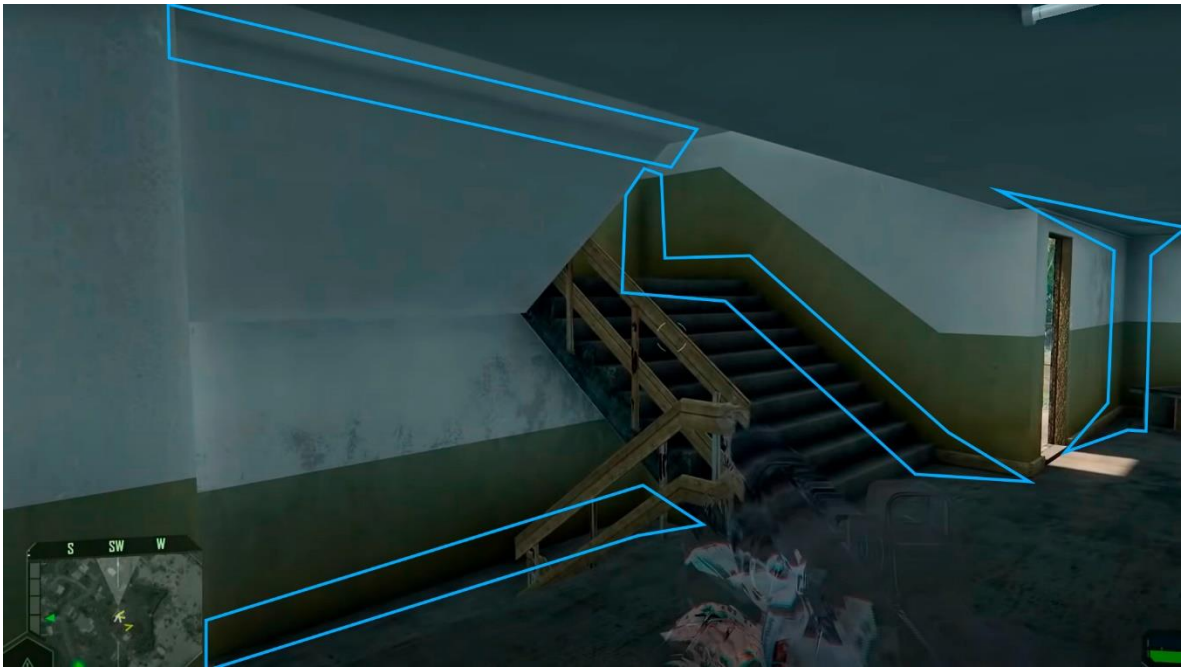


Fig. 21. Captura del videojuego *Crysis* (2007) *Crytek*, donde se ve el efecto de la oclusión ambiental (AO) en los modelos 3D de la arquitectura.

Junto a la aparición de otros mapas de texturas y las mejoras en los sistemas de render para la simulación de la iluminación, alcanzamos el hito tecnológico que revolucionará este contexto: se configuran lo que actualmente conocemos como los materiales PBR, siglas de *Physically Based Rendering*. Este sistema de materiales de aplicación en modelos 3D, creado por *Pixar* para la producción de animaciones 3D, resultó ser la mejor opción para el texturizado de modelos virtuales en entornos con renderizado en tiempo real, y con la

[24] *Crysis* (PC, PS3, Xbox)[videojuego] (2007) *Crytek*.

evolución tecnológica que permitía texturas con mayores resoluciones, logra generar escenas más complejas con un mayor número de polígonos, y sin pérdida de fotogramas.

Esto da lugar a la creación de entornos de aspecto cada vez más realista, como en la aplicación *The Matrix Awakens*[25] (Fig. 22), o, sin buscar tanto fotorrealismo, escenas espectaculares y más artísticas, como podemos ver en *Horizon Forbidden West*[26] (Fig. 23), repletas de detalles y una gama cromática espectacular.

Fig. 22. Captura para material promocional de *The Matrix Awakens* (2021) *Epic Games* / *Warner Bros*.

Fig. 23. Captura en una *Play Station* 5 para material promocional de *Horizon Forbidden West* (2022) *Guerrilla Games.*

[25] *The Matrix Awakens* (PS5, Xbox Serie X)[videojuego]. (2021) *Epic Games / Warner Bros*.
[26] *Horizon Forbidden West - Horizon 2*(PS4/PS5)[videojuego]. (2022) *Guerrilla Games*.

2.3.4.- EVOLUCIÓN TECNOLÓGICA

Con la aparición de los videojuegos en tres dimensiones, las videoconsolas y otros equipos también dieron el salto, pasando de los procesadores de 32 a los de 64 bits. Esto suponía poder desarrollar operaciones y procesos más complejos y trabajar con más datos, permitiendo a estos dispositivos domésticos realizar renderizados más elaborados. Esto supuso que las aplicaciones comenzaran a evolucionar al mismo ritmo que los dispositivos que salían al mercado, debido a que las empresas desarrolladoras debían generar aplicaciones que los usuarios pudieran ejecutar en sus equipos. Los videojuegos salen de los salones recreativos para entrar en los hogares, por lo que comienzan a depender, ya no sólo de la evolución tecnológica, sino de su comercialización y democratización para poder alcanzar nuevos hitos de desarrollo.

En este punto, el trabajo del artista 3D en la realización de entornos *Real-Time* ya no depende tanto de sus habilidades, ya que, si así lo desea, puede generar mallas 3D con miles de millones de polígonos, o texturas en resolución 12K. Uno de los principales problemas que debe abordar el desarrollador a la hora de realizar una producción que se ejecute en tiempo real, ya no es sólo la potencia de sus herramientas, sino que el usuario final tenga un dispositivo que pueda ejecutar la aplicación una vez finalizada y sin problemas de rendimiento, así que dentro de esta contextualización o marco histórico tenemos que empezar a hablar de los saltos tecnológicos relacionados con los gráficos 3D de aplicación en entornos con renderizado en tiempo real, tanto como herramienta del artista digital como elemento limitador a la hora de crear contenido.

Al hablar de evolución tecnológica debemos hablar tanto del *hardware* como del *software*. Respecto al *hardware,* es preciso reflexionar sobre la importancia y funcionalidad de las tarjetas gráficas y sus unidades de procesamiento gráfico (*GPUs*). Son procesadores que trabajan en paralelo al procesador del equipo para el cálculo y aceleración de los gráficos a mostrar. Estas *GPUs* tienen una arquitectura muy diferente a la de las unidades centrales de

procesamiento (*CPUs* de arquitectura Von Neumann[27]). Su principal diferencia está en la capacidad de las *GPUs* de segmentar las operaciones de cálculo, realizándose éstas simultáneamente en cada unidad de procesamiento para lograr una gran cantidad de datos de salida (imágenes) en muy poco tiempo. Resulta fundamental la diferenciación; mientras la *CPU* utiliza pocos y grandes núcleos muy potentes que realizan operaciones separadas, la *GPU* utiliza muchos y más sencillos núcleos que trabajan de forma simultánea para aumentar el rendimiento. La *GPU* funciona procesando vectores y píxeles para dar lugar a los gráficos que posteriormente se mostrarán en pantalla, realizando cálculos de preproducción, producción y postproducción en milésimas de segundo.

Se denomina como *Render Pipeline 3D* al proceso por el que la GPU recibe la información de los modelos 3D (input) y genera la imagen que se mostrará por pantalla en 2D (output) (Fig. 24). Este proceso se compone a su vez de diferentes subprocesos que se agrupan en dos (sub)*Pipelines* principales:

- ***World Space Pipeline***. Es el trabajo con vértices y polígonos que darán forma a los elementos tridimensionales dentro del espacio virtual. Este proceso era inicialmente realizado por la CPU, pero actualmente es también realizado por la tarjeta gráfica.
- ***Screen Space Pipeline*.** Se capturan los elementos del espacio tridimensional que se van a ver en la pantalla transformando toda la información a una imagen 2D (rasterizado), y se le aplica una postproducción consistente en la aplicación de los *shaders*, modificando el resultado final y la imagen mostrada por pantalla.

La evolución de los procesos de *Render Pipeline* 3D ha cambiado gracias al desarrollo tecnológico que continúa en nuestros días, pero que tiene un primer ítem destacable en el año 1999, cuando la compañía *NVIDIA* desarrolla la tarjeta gráfica *GeForce* 256, considerada la primera *GPU* completa (II Generación)[28], ya que era capaz de hacer todos los cálculos

[27] Se recomienda consultar el recurso en línea: estructuras de microprocesadores - https://bibdigital.epn.edu.ec/bitstream/15000/6172/1/CD-4827.pdf

[28] MCCLANAHAN, C. (2010) "*History and Evolution of GPU Architecture*" Georgia, Georgia Tech, *College of computing*.

necesarios para la creación de una imagen digital 3D dejando a la *CPU* sólo la labor de mostrar el resultado por pantalla.

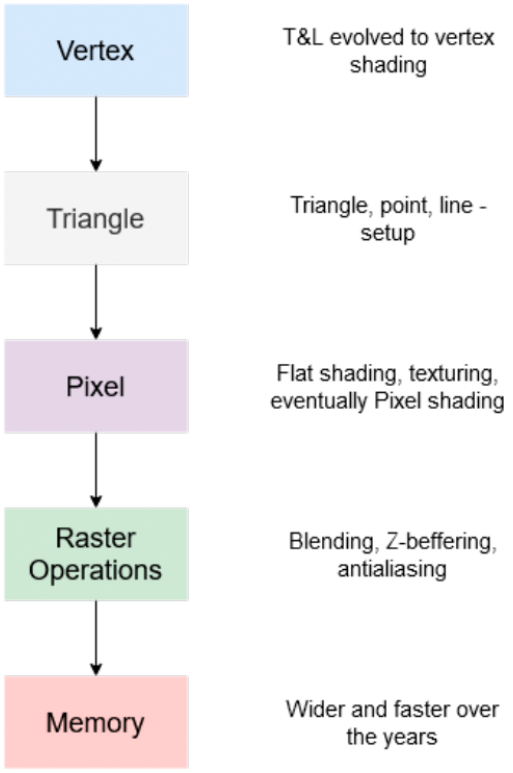


Fig. 24. Funciones fijas tradicionales del *Render Pipeline* 3D - RODRIGUES, P. (2021) *Field-configurable GPU*.

Tabla 1. Diferencia técnica en cifras entre tarjetas gráficas en 23 años. Elaboración propia.

	MODELO	
	NVIDIA GeForce 256	NVIDIA GeForce RTX 4090
Año de lanzamiento	1999	2022
ESPECIFICACIONES RELEVANTES		
Tasa de pixeles por segundo	480.0 Mpixels/s	443520.0 Mpixels/s
ROPs[29]	4	176
TMUs[30]	4	512
Memoria	32 MB	24000 MB

[29] ROP - sub-procesador responsable de cargar las texturas en la VRAM.
[30] TMU - sub-procesador responsable de aplicar las texturas a los pixeles

Esta tecnología ha evolucionado durante las últimas dos décadas, y no sólo en lo que a electrónica se refiere, sino también en que la programación relacionada se ha volcado en la creación y mejora del código y los procesos que permiten la creación de entornos cada vez más realistas y complejos.

Desde los años 90, con cada nueva generación de *GPUs* ha aumentado la capacidad de los equipos para la realización de gráficos a través de la gestión de los modelos 3D (vértices y polígonos), los mapas de texturas (color y rasterizado) y los *shaders* (iluminación y efectos). Esta evolución puede seguirse a través de los avances en *software*, en concreto a través de los avances de la API DirectX.

En 1995 aparece DirectX 1, la primera versión de este compendio de APIs centrado en la visualización de gráficos y vídeo desarrollado por *Microsoft*, y que manejaba gráficos en 2D y 3D. En caso de tener que trabajar con gráficos en 3D, el trabajo de proyección de los vértices y polígonos lo realizaba la *CPU* y no el procesador gráfico. Incluía *double buffering*, responsable de eliminar el lagrimeo o ruido entre imágenes al permitir mostrar una imagen finalizada por pantalla mientras se trabaja en la siguiente, y *Z-buffer*, responsable de la profundidad y la superposición de las geometrías en la escena, que se siguen usando y son esenciales a día de hoy para el proceso de renderizado.

En 1996, con el DirectX 2, aparece el rasterizado y la librería Direct3D, responsable del procesamiento y la programación de gráficos 3D dentro del DirectX, y en 1997 las tarjetas gráficas incluyen el Puerto de Aceleración Gráfica (*Accelerated Graphics Port* (AGP)), que aumentaba significativamente la velocidad de transferencia de los datos gráficos en el equipo. Además, incluía multi-texturizado, permitiendo el uso de múltiples mapas de texturas durante el rasterizado de una misma superficie. Esto supuso un gran avance en el código del DirectX 5, que en conjunto permitía el renderizado de escenas texturizadas más eficientemente y ampliaba el abanico de efectos visuales en la postproducción (*shaders*).

En 1999 las tarjetas gráficas incluyen un nuevo módulo que mejora el *Render Pipeline* 3D: *Transformation and Lighting module* (T&L), lo que liberaba totalmente a la *CPU* del cálculo de los modelos 3D (vértices y polígonos), y permitía los cálculos de iluminación per-vertex,

método de iluminación que calcula la influencia de la luz en cada vértice de los polígonos que conforman el modelo tridimensional.

En el año 2000, el DirectX 8 incluía: *multisampling* (proceso por el que se reduce el *aliasing* durante el rasterizado), *point sprites* (utilizado principalmente para la creación de emisores de partículas), 3D *volumetric textures* (permite el uso de texturas UVW), *higher-order surface primitives* (permitiendo el uso de polígonos complejos), o *vertex blending* (que hace posible el uso de múltiples texturas combinadas en el mismo polígono,) entre otros añadidos que mejoran el uso de la memoria y la fluidez de los datos. A partir de esta versión del DirectX se comienza a hablar de los modelos de *shader*, comenzando por el *Pixel Shader* y pasando al *Shader Model* en la Versión siguiente.

Tabla 2. Relación de versiones del DirectX con su modelo de *Shader*. Elaboración propia.

VERSIÓN DE DirectX	SHADER MODEL
DirectX 8.x	*Píxel Shader* 1.3 y *Pixel Shader* 1.4
DirectX 9.x	*Shader Model* 2.x y *Shader Model* 3.0
DirectX 10.x	*Shader Model* 4.x
DirectX 11	*Shader Model* 5.0
DirectX 11_0+ y DirectX 12	*Shader Model* 5.1 / 6.0 - 6.6

En 2002, El DirectX 9 pasa a formar parte del estándar del Sistema Operativo Windows y el Lenguaje de Sombreado de Alto Nivel, o HLSL, es actualizado a los *Shaders* Modelos 2 y 3. Durante los años siguientes, tanto el *hardware* como el DirectX, se desarrollaron en gran medida mejorando el rendimiento y aumentando los recursos admitidos.

En 2006, el DirectX 10 aparece con un nuevo *shader* para geometrías (Modelo 4,) entre otros cambios que fueron muy drásticos para el momento, lo que retrasó la adopción de la API por parte de los usuarios, ya que muchas aplicaciones que salían al mercado en esas fechas aún

utilizaban su versión anterior. Se dio una situación en la que el avance tecnológico se adelantó significativamente a los avances en el desarrollo de aplicaciones.

En 2009, llega el DirectX 11. El desarrollo de las tarjetas gráficas se centra en la mejora y la introducción de nuevos *shaders*, pasando la API al modelo de *Shader* 5.

En 2015, Llega el DirectX 12, versión actual, con una API mucho más ligera, lo que permite una mayor optimización del *hardware* en relación con las aplicaciones, y, como su propia documentación indica, ha sido diseñado para programadores gráficos avanzados, siendo su última versión a fecha de este escrito, la DirectX 12 Ultimate[31].

Parte integrante de la API DirectX, fundamental para la programación de los *shaders*, es el Direct3D, que utiliza el HLSL con el que se codifica o escribe el código de los *shaders* para dicha API, y es utilizada principalmente por los equipos configurados con OS de *Microsoft* (*Windows*). Existen otros lenguajes de codificación de *shaders* como el GLSL (*OpenGL* (*Graphic Library*) *Shader Language*) alternativa de código libre, y el Cg (*C for Graphic*s), desarrollado por NVIDIA.

Otras APIs en el mercado para el desarrollo de gráficos son *Vulkan*, alternativa a *OpenGL* y soportada por los OS *Windows*, *Android* y *Linux*; o *Metal*, para los equipos con OS de *Apple*.

A fecha de este escrito, uno de los últimos avances a nivel de *software* ha sido el paso del *Ray Tracing*, sistema por el cual se calculaba el trazado de la luz en la escena, al *Path Tracing* (Fig. 25), planteado y desarrollado por Jim Kajiya en 1986, se basa en las fórmulas para el cálculo de rayos de luz del *Ray Tracing* y permite representar con mayor precisión cómo actúa la luz en la escena y cómo se dispersa, pero su consumo en tiempo y recursos técnicos era inalcanzable hasta ahora para los motores con renderizado en tiempo real. Utilizado por primera vez en el cine en 2006 para el renderizado de la película *Monster house*[32] (Fig. 26), no ha sido adaptado hasta 2022 a los entornos con renderizado en tiempo real.

[31] Web oficial DirectX 12 con material informativo complementario sobre la capacidad tecnológica desarrollada. Recuperado de: https://www.nvidia.com/es-es/geforce/technologies/directx-12-ultimate/

[32] *Monster house* (2006) Gil Kenan.

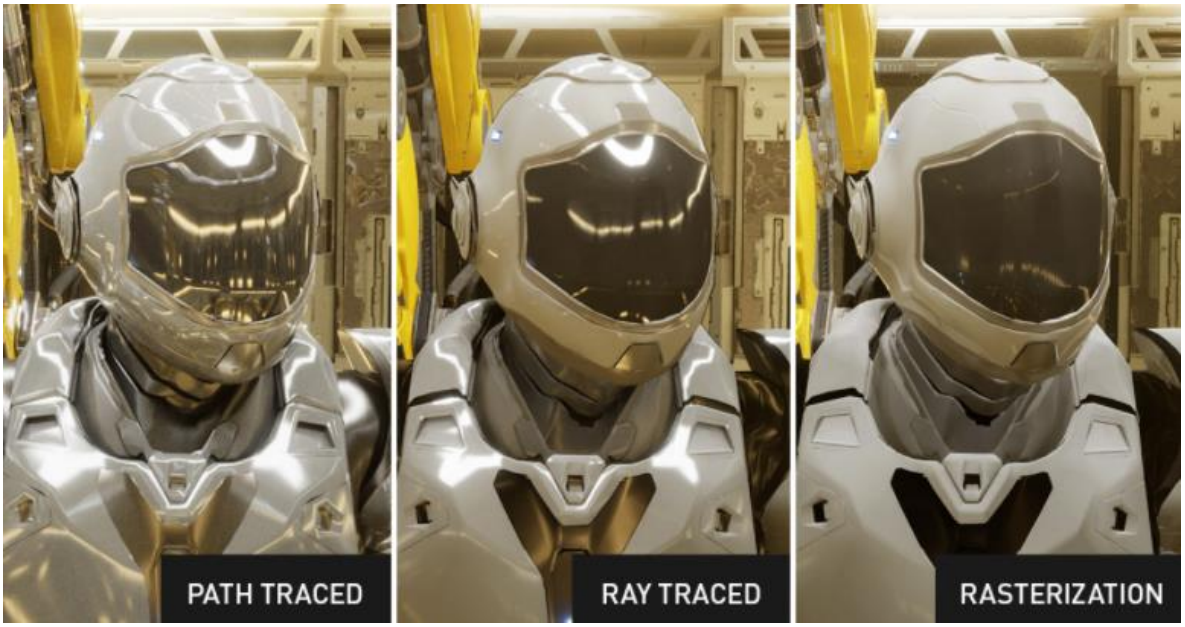


Fig. 25. Diferentes simulaciones de la luz en entornos con renderizado en tiempo real. (NVIDIA).

Resulta de especial relevancia la evolución tecnológica desarrollada hasta el momento, y gracias a la cual pueden comprenderse los vasos comunicantes que interactúan entre el desarrollo de los gráficos por ordenador, en concreto los gráficos 3D, y el comienzo del desarrollo de entornos 3D con renderizado en tiempo real. Además, hemos visto cómo ha evolucionado el *hardware* y el *software* en cuanto a términos gráficos se refiere, generando un marco completo de las herramientas con las que ha de trabajar el artista gráfico cuando se enfrenta al desarrollo en entornos con renderizado en tiempo real y materiales PBR.

Fig. 26. Captura de la película *Monster House* (2006) Gil Kenan.

3.- METODOLOGÍA

3.1.- INTRODUCCIÓN

Cuando tratamos con entornos con renderizado en tiempo real, estamos hablando de imágenes renderizadas y mostradas por pantalla. El contenido mostrado en esas imágenes es diseñado y creado por artistas. Desde el modelado del objeto 3D hasta la imagen final mostrada por pantalla, hay una serie de procesos que deben ser tratados para comprender este tipo de proyectos.

Procederemos a continuación a la recopilación, estudio y análisis de los diferentes elementos que intervienen en el proceso de creación de una imagen obtenida en base a un entorno con renderizado en tiempo real. Posteriormente, nos centraremos en los materiales PBR y realizaremos un proceso experimental en el que se estudiarán y analizarán los componentes y parámetros que definen los mapas de texturas que componen dichos materiales.

Los resultados, básicamente visuales, serán tratados desde una perspectiva técnica enmarcada en el campo de las artes. Esto servirá como base objetiva que permitirá reproducir los resultados por aquellos que se acerquen a este ámbito. Además, se aportarán experimentos y pruebas que profundicen en los aspectos estudiados para la obtención de resultados más concretos.

Para realizar dicho estudio se procederá utilizando una metodología cualitativa, ya que la información recopilada, a pesar de estar basada, en ciertos casos, en datos cuantificables, se traducen en resultados que facilitan la comprensión de la materia, la visualización de efectos y la comparación entre ellos.

Al afrontar la creación de volúmenes y espacios en un *software* de edición y creación 3D encontramos parámetros y características similares, si no idénticas, a todos ellos, lo que nos permite su estudio y análisis.

3.1.1.- ANÁLISIS DE LOS PROCESOS PARA EL DESARROLLO EN ENTORNOS 3D CON RENDERIZADO EN TIEMPO REAL

Como podemos ver en diferentes *softwares* existentes en el mercado, ya sean editores 3D como *Cinema 4D* (*Maxon*) o *Blender* (*Blender Foundation*), o motores gráficos como *Unity* (*Unity Technologies*) o *Unreal Engine* (*Epic Games*), para la obtención de una imagen digital a través del renderizado de una escena diseñada en un entorno 3D, ya sea con renderizado en tiempo real o no, el proceso es muy similar. Desde las imágenes que nos muestran los diferentes programas de edición 3D, a cada uno de los fotogramas renderizados de alta calidad para una producción audiovisual, todos ellos pasan por un proceso de creación o preproducción previa al renderizado final.

Las coincidencias y similitudes encontradas nos permiten convertir el proceso para la obtención de una sola imagen en dicho entorno en un diagrama de flujo (Fig. 27), más utilizado en otras disciplinas o áreas de conocimiento. Al observar el diagrama, nos encontramos con que todas las entradas de información o de datos que recibe el flujo para poder llegar al resultado óptimo y controlado, tienen base o relación directa con disciplinas artísticas como la escultura, el diseño gráfico, la ilustración o la fotografía entre otras, lo que hace necesario que un desarrollador 3D tenga conocimientos en áreas como el color, diseño o iluminación para la obtención de resultados óptimos.

Debemos tener en cuenta que cuando se introduce, por ejemplo, una textura en este flujo, un desarrollador ha debido producir dicha imagen y validarla para ser integrada. Muy probablemente, dicha imagen, habrá pasado por algún programa de edición de imágenes desarrollado por programadores, pero el desarrollador, el que realiza la validación profesional de dicha imagen para que pase a formar parte del flujo, es un artista, al igual que el diseñador que coloca los elementos 3D en el espacio dando forma al entorno que constituirá la escena, el iluminador que se encarga de la luz en la misma, el modelador 3D que da forma a las mallas o el texturizador que genera los mapas de textura. Los artistas digitales que trabajan en este campo utilizan herramientas desarrolladas por programadores, pero eso no implica que los programadores estén involucrados en el resultado final o en la aplicación artística de la

herramienta que han desarrollado, igual que pasa en otras disciplinas como el diseño web o el diseño gráfico 2D.

Tal vez, por este motivo, en ciertos ámbitos, la creación de entornos 3D interactivos o entornos inmersivos, o el desarrollo de aplicaciones multimedia están considerados trabajos de otras áreas de conocimiento más relacionadas con la ingeniería y la informática, pero en realidad, en su mano está el desarrollo de las herramientas y programas, y en cierta forma, la codificación de ciertos parámetros artísticos a los citados espacios virtuales como trataremos más adelante. Al final, el usuario de dichas herramientas muy probablemente tendrá un perfil artístico o tendrá que recurrir a conocimientos y habilidades dentro del área de las Artes.

Cuando la creación del espacio virtual se desarrolla en un entorno 3D diseñado y concebido para su renderizado en tiempo real, nos encontramos con que en el proceso intervienen dos líneas principales de desarrollo:

- **La programación**, que permitirá la ejecución de la aplicación y que el usuario final no suele ver (*Backend*).

- **El diseño**, que comprende la creación de todos los elementos gráficos y es la parte visible de la aplicación (*Frontend*).

Existen en el mercado programas para la creación y desarrollo de este tipo de entornos que afirman que no se requiere de conocimientos de programación, por lo tanto, si eliminamos del proceso de creación la parte en la que interviene el código, nos queda exclusivamente el desarrollo audiovisual de la aplicación.

Finalmente, tras desglosar los pasos del proceso de trabajo en entornos 3D, podemos diferenciar bloques de trabajo en el desarrollo. Centrándonos únicamente en la creación de una escena básica interactiva que puede o no necesitar animación, diferenciamos tres bloques o procesos:

- **El Entorno 3D**: Mínimo requisito. Un espacio tridimensional virtual en el que ubicar la cámara, una luz y el objeto virtual.

- **El Modelado**: En la actualidad existen múltiples formas de obtener un modelo 3D. Hará falta, al menos, un elemento tridimensional que colocar en la escena para no tener un espacio vacío infinito.

- **El Texturizado**: Los elementos en el entorno virtual requieren de un material para ser reconocidos por la cámara. Los objetos virtuales sin material no tienen datos sobre cómo el entorno y la luz ha de reaccionar con ellos y la cámara los capturará como errores.

Este esquema del flujo de trabajo puede ampliarse o completarse si abarcamos otras líneas de estudio como, por ejemplo, la animación o formas de interacción en entornos virtuales, entre otros muchos procesos que pueden intervenir en el desarrollo de un entorno virtual con renderizado en tiempo real.

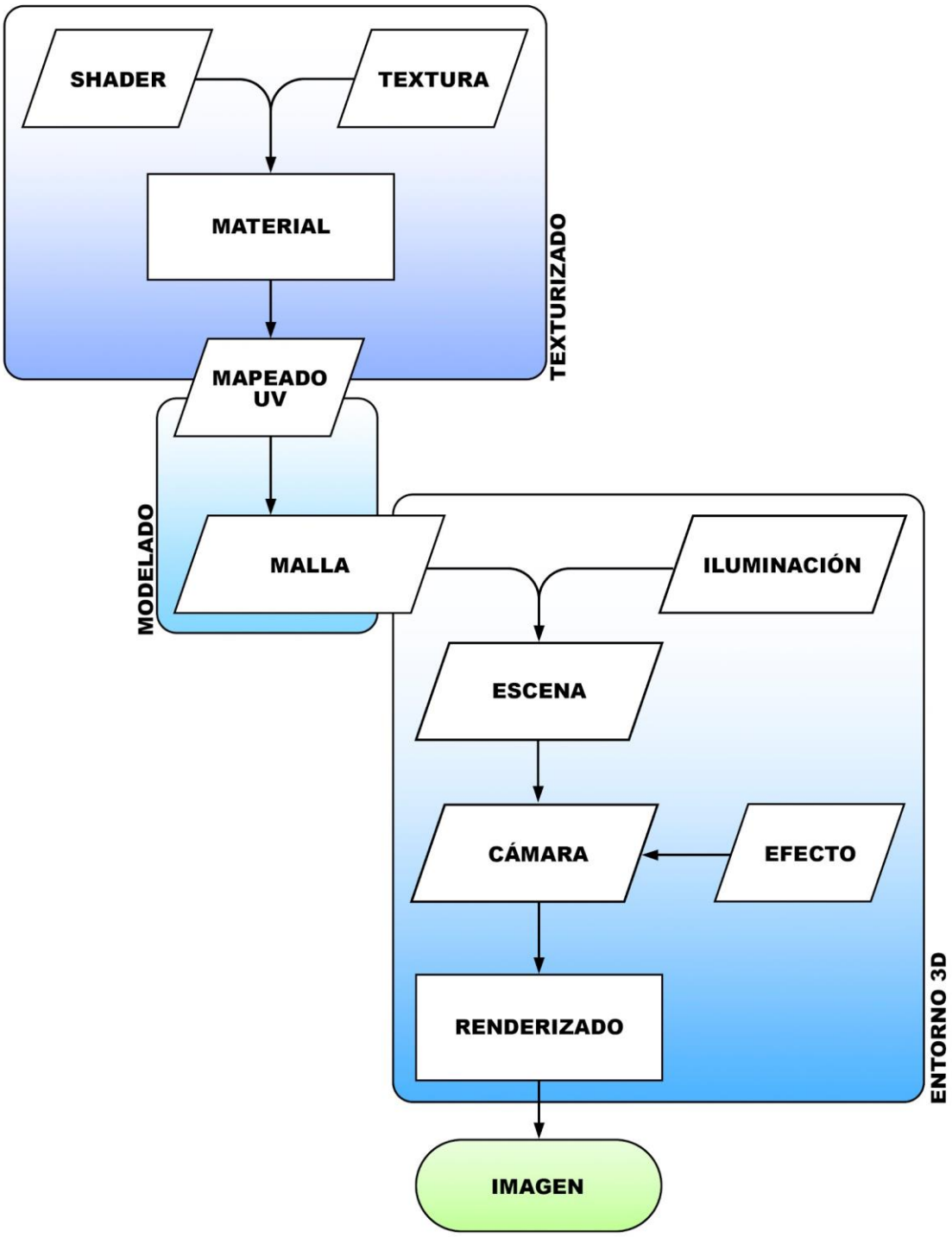


Fig. 27. Diagrama de flujo para la obtención de imágenes en entornos 3D. Elaboración propia.

3.2.- PROCESOS DEL FLUJO DE TRABAJO

Los procesos anteriormente expuestos en el flujo de trabajo para la obtención de una imagen renderizada son trabajos especializados, propios de las Artes, que el autor debe conocer en profundidad, incluyendo terminología técnica específica. A continuación, profundizaremos en cada uno de ellos analizando y definiendo los diferentes términos y cómo influyen en el resultado final.

3.2.1.- MODELADO

Término propio de las artes y en concreto del campo de la escultura, se caracteriza por el trabajo directo del artista sobre la materia blanda, maleable, que permite crear diferentes volúmenes retirando o incorporando materia al cuerpo inicial. En arte digital, el modelado es el proceso por el cual se genera y modifica una malla virtual compuesta de polígonos (modelo matemático), cuya complejidad varía con el trabajo del artista hasta obtener la forma deseada en el espacio tridimensional.

Actualmente existen diferentes formas de obtener un modelo tridimensional, ya sea a través del modelado en un editor 3D, usando escáneres tridimensionales, fotogrametría u otro medio de captura de objetos tridimensionales del mundo real, el modelado a través de algoritmos que generen modelos 3D aleatoriamente o en base a unos parámetros, u otros métodos.

Existen en el mercado diferentes programas para la creación y edición de modelos 3D conocidos como software CAD (*Computer-Aided Design*) o CADD (*Computer-Aided Design & Drafting*), pero suelen diferenciarse los editores para ingeniería o arquitectura (*Solid Works* o *AutoCAD* entre otros), y los editores artísticos (*Blender* o *Cinema 4D* entre otros).

Los diferentes programas para la visualización y edición de modelos 3D cuentan con herramientas para modificar la malla y realizar los diferentes procesos que requiere el trabajo con modelos virtuales como la animación, el mapeado UV, la creación de texturas o el renderizado de imágenes.

3.2.1.1.- MALLA

Un modelo, volumen o espacio delimitado en el entorno 3D está formado por una malla. La topología de la malla está configurada o definida por la posición de puntos (vértices) ubicados en el espacio tridimensional. Estos puntos ubicados en el espacio pueden estar relacionados o unidos por vectores (aristas). Cuando tres de estos vectores forman un triángulo cerrado (3 vértices y 3 aristas), se habrá formado un polígono de malla denominado *Tris*.

Un polígono de dicha malla puede estar compuesto de más vértices, denominados *Quad* (4 vértices y 4 aristas) o *N-Gons* (más de 4 vértices y más de 4 aristas) según el número de vértices y aristas, pero nunca menos de tres (Fig. 28).

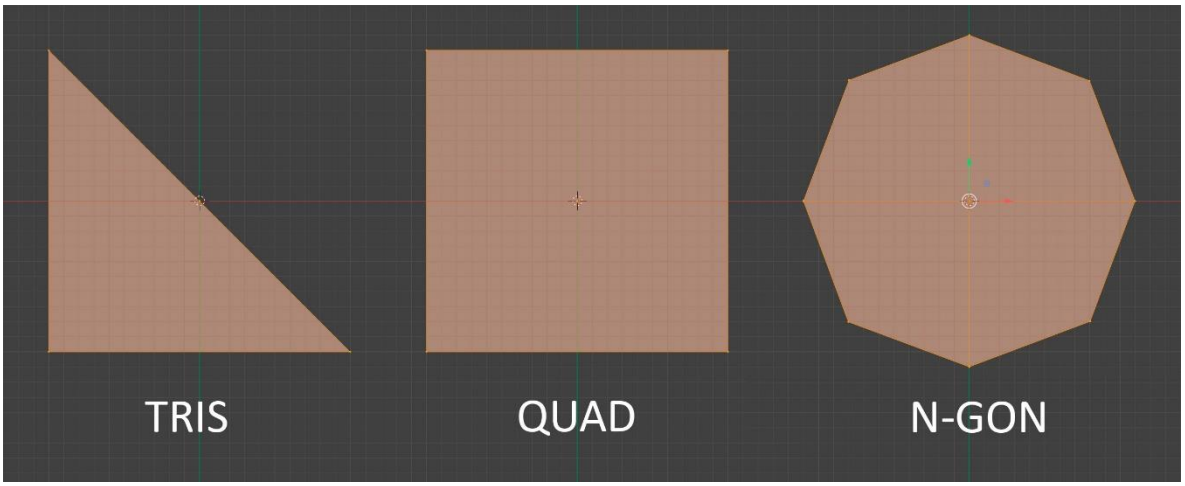


Fig. 28. Tipos de polígonos de malla. Elaboración propia.

Aunque el polígono pueda estar compuesto de más de tres vértices y aristas, el proceso del *Render Pipeline* tenderá a descomponer todas las geometrías en polígonos triangulares (*Tris*) durante el proceso de representación virtual de la geometría en la escena.

```
# Blender v3.0.1 OBJ File: ''
# www.blender.org
mtllib cubo.mtl
o Cube
v 1.000000 1.000000 -1.000000
v 1.000000 -1.000000 -1.000000
v 1.000000 1.000000 1.000000
v 1.000000 -1.000000 1.000000
v -1.000000 1.000000 -1.000000
v -1.000000 -1.000000 -1.000000
v -1.000000 1.000000 1.000000
v -1.000000 -1.000000 1.000000
vt 0.625000 0.500000
vt 0.875000 0.500000
vt 0.875000 0.750000
vt 0.625000 0.750000
vt 0.375000 0.750000
vt 0.625000 1.000000
vt 0.375000 1.000000
vt 0.375000 0.000000
vt 0.625000 0.000000
vt 0.625000 0.250000
vt 0.375000 0.250000
vt 0.125000 0.500000
vt 0.375000 0.500000
vt 0.125000 0.750000
vn 0.0000 1.0000 0.0000
vn 0.0000 0.0000 1.0000
vn -1.0000 0.0000 0.0000
vn 0.0000 -1.0000 0.0000
vn 1.0000 0.0000 0.0000
vn 0.0000 0.0000 -1.0000
usemtl Material
s off
f 1/1/1 5/2/1 7/3/1 3/4/1
f 4/5/2 3/4/2 7/6/2 8/7/2
f 8/8/3 7/9/3 5/10/3 6/11/3
f 6/12/4 2/13/4 4/5/4 8/14/4
f 2/13/5 1/1/5 3/4/5 4/5/5
f 6/11/6 5/10/6 1/1/6 2/13/6
```

Fig. 29. Archivo de un cubo desarrollado en *Blender*:
"v" son las coordenadas 3D de sus vértices.
"vt" son las coordenadas 2D de su mapeado UV.
"vn" son las normales de los polígonos que forman los vértices.
"f" son los grupos y orden de los vértices de cada polígono que forma la malla.
Elaboración propia.

Cuando un artista está modelando un objeto 3D, está editando la malla a través de la manipulación de los diferentes componentes del modelo matemático (vértices, aristas y polígonos), modificando las coordenadas espaciales (X, Y, Z) de los vértices (puntos en el espacio virtual) que forman la malla. Esto, al final, se traduce en un modelo matemático que queda plasmado en un archivo que guarda toda la información de la malla.

Si abrimos el archivo de un modelo 3D en un editor de código podremos ver una lista de coordenadas cartesianas de todos y cada uno de los puntos que forman la malla entre otros datos que definen aspectos relevantes del objeto virtual (Fig. 29).

3.2.1.2.- NORMALES DE GEOMETRÍA

Los polígonos de una malla, según el orden y el sentido en el que se ordenan sus vértices durante el proceso de creación del modelo dentro del entorno 3D, definen una propiedad de dichos polígonos que se denomina como normal del polígono (no confundir con el mapa de normales). La normal corresponde con un vector perpendicular a la superficie formada por dicho polígono, y cuando un polígono pasa por el Render Pipeline de un entorno con renderizado en tiempo real, éste interpreta los datos generando un polígono sólo visible por el lado en el que su normal es positiva, es decir, los polígonos de la malla sólo serán visibles (renderizados) para el resultado final por una sola de sus caras, aunque esto se puede modificar a través de los *shaders* que estudiaremos más adelante (Fig. 30).

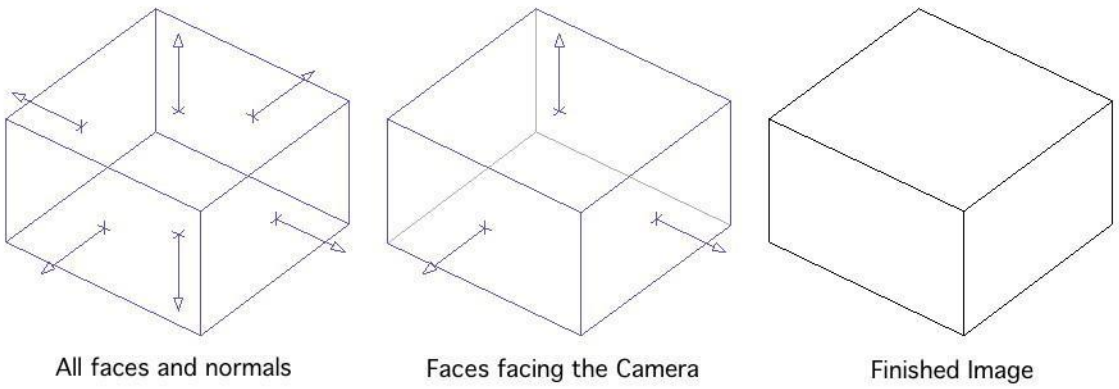


Fig. 30. Visual de las Normales de los polígonos que configuran un cubo - http://courses.washington.edu/arch481/1.Tapestry%20Reader/1.3D%20Data/5.Surface%20Normals/0.default.html

Los programas de edición de modelos 3D suelen tener un sistema para visualizar la orientación de la normal de los polígonos y herramientas para su modificación y reorientación.

3.2.1.3.- TIPOLOGÍAS DE MODELOS 3D

Los modelos 3D se diferencian en base a distintos factores, uno de ellos, de gran relevancia y relacionado con su aspecto visual final, es el número de polígonos del que está compuesta su malla respecto del tamaño total del modelo.

Aquellas mallas que tienen un gran detalle en su modelado, y por lo tanto cuentan con una gran cantidad de polígonos, se los define como modelos de alto poligonaje o *high poly*, y las que se realizan reduciendo los detalles al mínimo, y por lo tanto reduciendo su número de polígonos, se las denomina de bajo poligonaje o *low poly*.

Existe un término intermedio, el medio poligonaje o *mid poly,* que busca el equilibrio entre los detalles en la malla y los insertados a través de los materiales y sus texturas (Fig. 31).

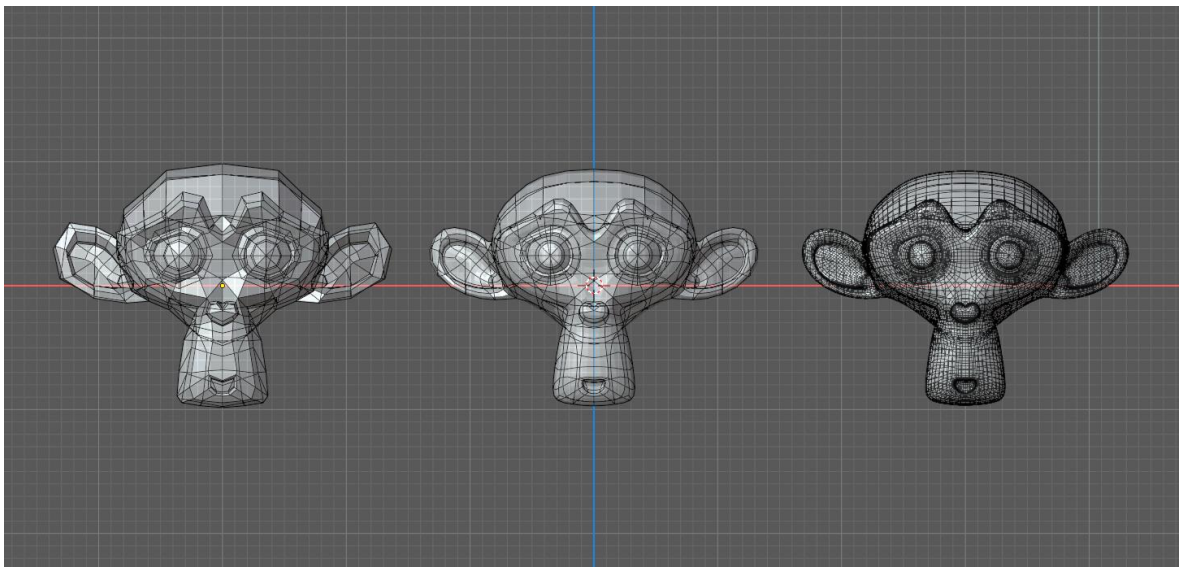


Fig. 31. De izquierda a derecha, mismo modelo de Suzanne (*Blender*) con diferente recuento de polígonos: *low poly*, *mid poly* y *high poly* - Elaboración propia.

Este aspecto es importante, ya que en los entornos con renderizado en tiempo real sobre los que versa esta tesis se suele recurrir a modelos *low poly* o *mid poly* optimizados, debido a que, en el proceso del Render Pipeline, como ya se ha explicado, se recreará toda la geometría

contenida en la escena. Una escena que tenga pocos polígonos tendrá un procesado de la geometría más rápido y permitirá un mayor recuento de polígonos por segundo.

Una forma de reducir los detalles en la malla sin perderlos es convertirlos en parte del material a través de sus mapas de texturas. Esto es posible gracias a los materiales PBR. Sus diferentes mapas de texturas y su funcionamiento en el entorno permiten al diseñador o artista complementar el modelo, al pasar detalles de una malla *high poly* a una con menos polígonos, sin renunciar a los detalles. El proceso por el que se pasa de una malla de alto poligonaje a una con un recuento menor de polígonos se conoce como retopología, lo que reduce de forma sustancial el número de polígonos de la malla.

Parte de los detalles de la malla original de alto recuento de polígonos son trasladados a la malla *low poly* a través de procesos como el "bakeado" (término utilizado coloquialmente, del inglés *to bake* - hornear), por el que esos detalles del modelo se reproducen en una textura que podremos insertar en un material, y que, a través del *shader,* modificará el aspecto final de la superficie del modelo simulando los detalles de la malla *high poly* sobre una de menor poligonaje.

En 2021, el motor gráfico *Unreal Engine* 5 (*Epic Games*) presentó la tecnología Nanite[33] (Fig. 32). Según la describen, "*Nanite is Unreal Engine 5's new virtualized geometry system which uses a new internal mesh format and rendering technology to render pixel scale detail and high object counts*" [Nanite es el nuevo sistema de geometría virtualizada de *Unreal Engine* 5 que utiliza un nuevo formato de malla interna y tecnología de renderizado para renderizar detalles a escala de píxeles y una gran cantidad de objetos]. Actualmente, y según podemos leer en la documentación, esta tecnología sólo puede usarse en los objetos estáticos de la escena, es decir, aquellos que no se mueven ni tienen animación.

Como podemos ver, esta nueva tecnología permite el uso de modelos tridimensionales muy pesados (*high poly*), lo que en cierto modo permite al diseñador o artista una mayor libertad

[33] Web del motor gráfico *Unreal Engine* 5 > Nanite Virtualized Geometry - https://docs.unrealengine.com/5.0/en-US/RenderingFeatures/Nanite/

creativa. El trabajo de crear o diseñar algunos de los modelos más pesados (en referencia a su recuento de polígonos) ya no está sujeto a la posterior retopología y al encaje de los detalles en las texturas.

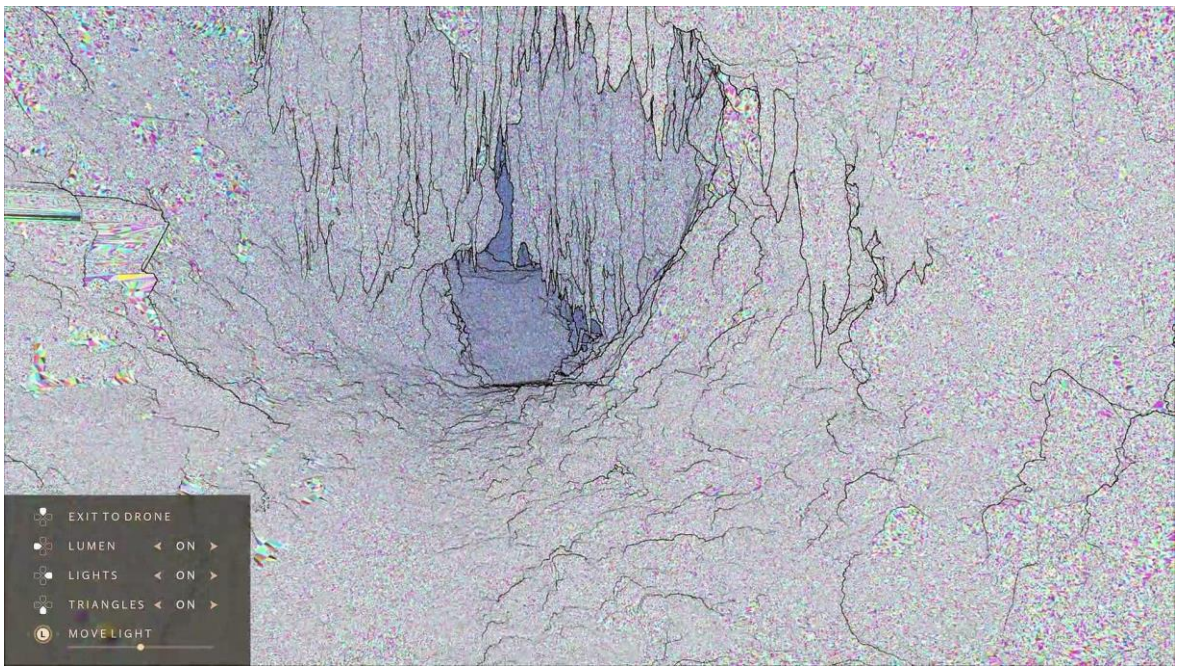


Fig. 32. Captura de *Lumen in the Land of Nanite* - https://www.youtube.com/watch?v=qC5KtatMcUw

Siguiendo con la clasificación de las mallas 3D, existe otro método de catalogación basado en las formas de su superficie, diferenciando modelos mecánicos o industriales, y modelos orgánicos.

Se denominan como modelos mecánicos o industriales aquellos cuyas mallas poseen superficies lisas y ángulos bien definidos, y suelen estar más relacionados con procesos industriales, como piezas mecanizadas, productos resultantes de un proceso industrial o relacionadas con la arquitectura o la construcción. Para trabajar u obtener modelos mecánicos existen gran cantidad de programas en el mercado, ya sea en el ámbito de las artes digitales o el de la ingeniería o la arquitectura, aunque estos últimos suelen carecer de herramientas propias de los *softwares* destinados al desarrollo artístico de dichas mallas (los desarrolladores de dichos programas ya están marcando una diferencia según el usuario al que van orientados sus programas). Cuando se trabaja en la creación de este tipo de modelos se lo denomina como modelado de *hard surface* o modelado inorgánico.

Se denominan como modelos orgánicos aquellos cuyas mallas están compuestas de superficies curvilíneas o abultadas, pliegues o formas más propias de la naturaleza, como figuras anatómicas, animales o vegetales. Este tipo de modelos están compuestos habitualmente de muchos más polígonos que los mecánicos, ya que sus detalles suelen requerir de una subdivisión mucho mayor de la malla. No es habitual que los programas para ingeniería o arquitectura cuenten con herramientas para este tipo de moldeado.

3.2.1.4.- SELECCIONES Y COSTURAS

Una misma malla puede tener aplicados múltiples materiales para obtener resultados visuales diferentes en el renderizado final. Pongamos como ejemplo un vehículo, donde habrá materiales metálicos para la chapa, materiales transparentes para los cristales, o materiales con efecto de goma para los neumáticos.

Para conseguir efectos visuales distintos en localizaciones diferentes de la superficie de un mismo modelo, se utilizan materiales con configuraciones y parámetros definidos para dicho efecto y aplicados en diferentes selecciones de polígonos. Durante el proceso de edición del modelo se pueden crear múltiples selecciones de geometría, ya sean vértices, aristas o polígonos con varias finalidades, como la aplicación de diferentes materiales.

Las selecciones de polígonos, además de permitir el uso de diferentes materiales en un mismo modelo, facilitan el proceso de mapeado UV junto a las “costuras” (*seams*) (Fig. 33), que son selecciones de aristas por donde se separará o “descoserá” la geometría, facilitando el despliegue (*unwrap*) de los polígonos sobre la superficie bidimensional del mapeado UV.

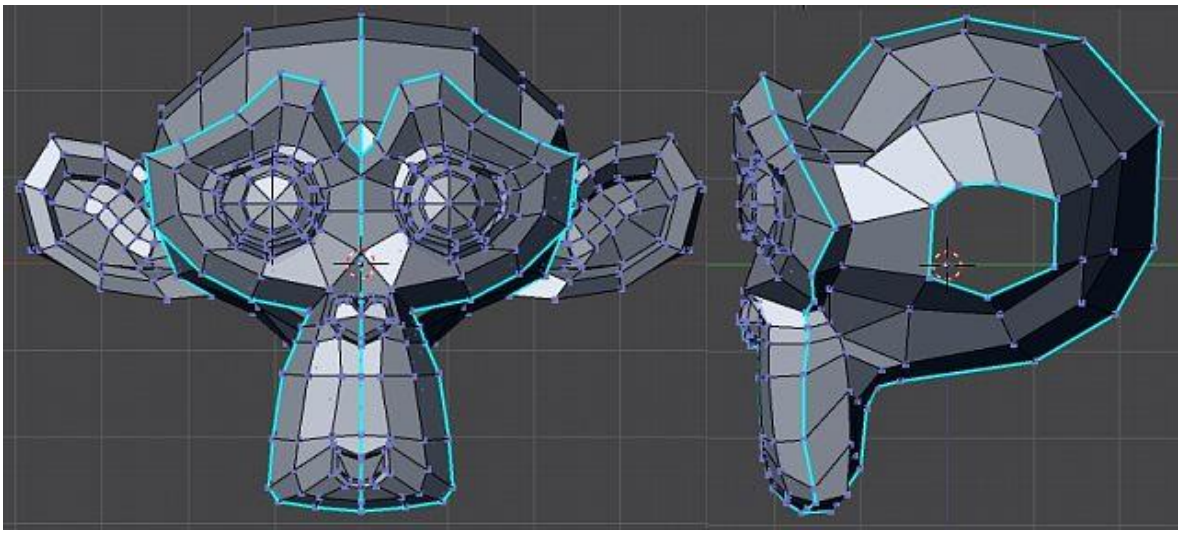


Fig. 33. Detalle de las costuras (aristas resaltadas en azul) –
https://download.blender.org/documentation/htmlI/ch11s06.html

3.2.1.5.- MAPEADO UV

Una vez obtenida la malla tridimensional final debemos aplicarle un material. Cuando ese material incluye texturas que deben ajustarse al modelo en sitios específicos de la malla, debemos realizar un mapeado UV. Este paso en el desarrollo se encuentra en un punto intermedio entre el modelado y el texturizado, ya que el autor estará trabajando sobre los polígonos de la malla al mismo tiempo que edita y modifica las texturas para ajustarlas al modelo.

El mapeado UV es el proceso a través del cual se proyecta la geometría (polígonos) de la malla de dicho modelo 3D sobre una superficie bidimensional llamada mapa UV, por sus coordenadas U para el eje X (horizontal) y V para el eje Y (vertical), con valores entre 0 y 1, estando el punto 0,0 abajo a la izquierda de la superficie de mapeo (Fig. 34).

Existe, también, el Mapeado UVW, de iguales características, en el que la coordenada W se utiliza para la profundidad de materiales 3D.

Fig. 34. De izquierda a derecha - Mapeado UV de un guante, Textura realizada en *photoshop* de acuerdo con el mapeado UV, Modelo 3D con el material que contiene la textura ajustada al mapeado UV. Elaboración propia.

Este proceso ha de realizarse de tal forma que los polígonos posicionados en el espacio tridimensional no sufran deformaciones al situarlos sobre el espacio bidimensional, evitando así errores de texturizado sobre el modelo al visualizarse o estamparse la textura contenida dentro de los polígonos que puedan haber sufrido algún tipo de deformación (Fig. 35).

Este posicionamiento puede hacerse a través de herramientas que facilitan los *softwares* de edición 3D, como el *unwrap* o desenvoltura, por el cual el programa calcula la descomposición y posición óptima de la malla para su mapeado sobre el espacio bidimensional, a través de las posibles y diferentes proyecciones y descomposiciones de la malla 3D sobre la superficie 2D, o bien de forma manual, moviendo y colocando cada polígono sobre la superficie de mapeado UV.

Contenidas dentro de la sección bidimensional destinada al mapa UV, se ubicarán y encajarán las texturas, sea cual sea su resolución, y a través de la relación establecida durante el mapeado UV los pixeles se ubicarán y visualizarán sobre el modelo 3D.

Contenidas dentro de la sección bidimensional destinada al mapa UV, se ubicarán y encajarán las texturas, sea cual sea su resolución, y a través de la relación establecida durante el mapeado UV los pixeles se ubicarán y visualizarán sobre el modelo 3D.

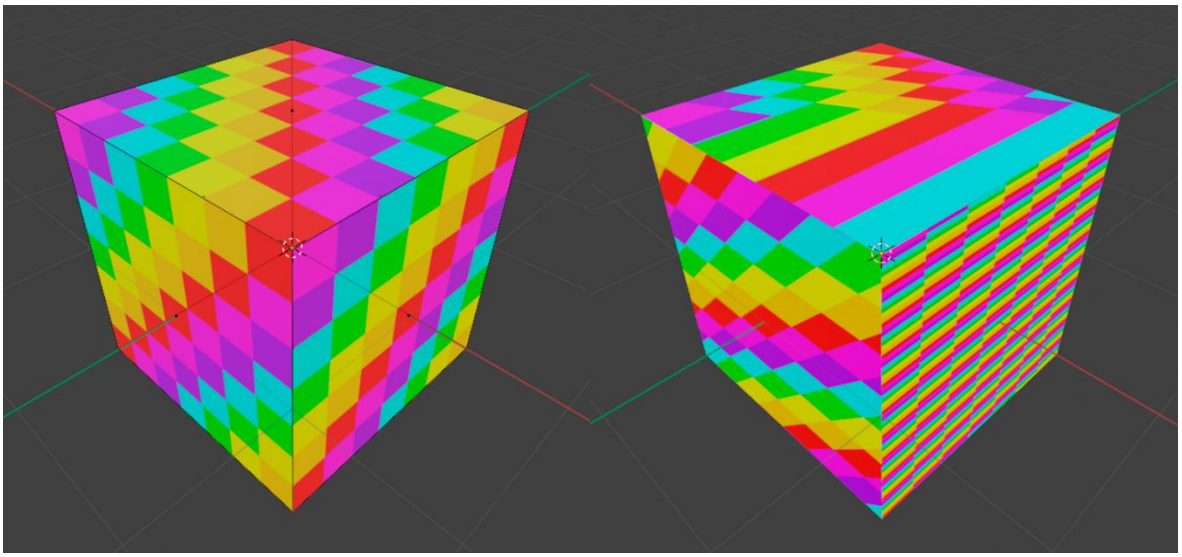


Fig. 35. Encaje correcto de una textura en los lados de un cubo (izquierda).
Diferentes errores de Mapeado UV que distorsionan la textura (derecha). Elaboración propia.

Para ello, el modelo tridimensional se descompone en los polígonos de los que está formada la malla y se subdivide en secciones o recortes de uno o más polígonos que se ubican y extienden sobre el mapa UV.

Una vez que se tiene una primera relación de píxeles/polígonos tras extender la malla sobre las coordenadas cartesianas UV, el diseñador modificará la posición y tamaño de los polígonos dentro de la sección bidimensional para ajustar lo máximo posible la relación de los píxeles de los mapas de texturas a la superficie de los polígonos en el espacio tridimensional.

En ocasiones, debido a las dimensiones de un modelo 3D, una textura debería tener unas dimensiones (calculadas en píxeles) muy grandes para que no se produzca una pixelación de la misma en la superficie del polígono por una baja densidad de píxeles, por ejemplo, una textura para una carretera. Esto supondría un gran consumo de memoria durante la ejecución final de la obra. Para evitar esto, el mapeado UV permite extender los polígonos más allá de las dimensiones de 1x1 (UxV) destinadas a encajar la textura, y el software completa la información repitiendo la misma textura infinitas veces utilizando la repetición como parte del texturizado. Este efecto se puede conseguir a través de parámetros como el *tiling* y el *offset,* que permiten realizar modificaciones en las coordenadas de las texturas sobre el encaje del mapeado UV (Fig. 36).

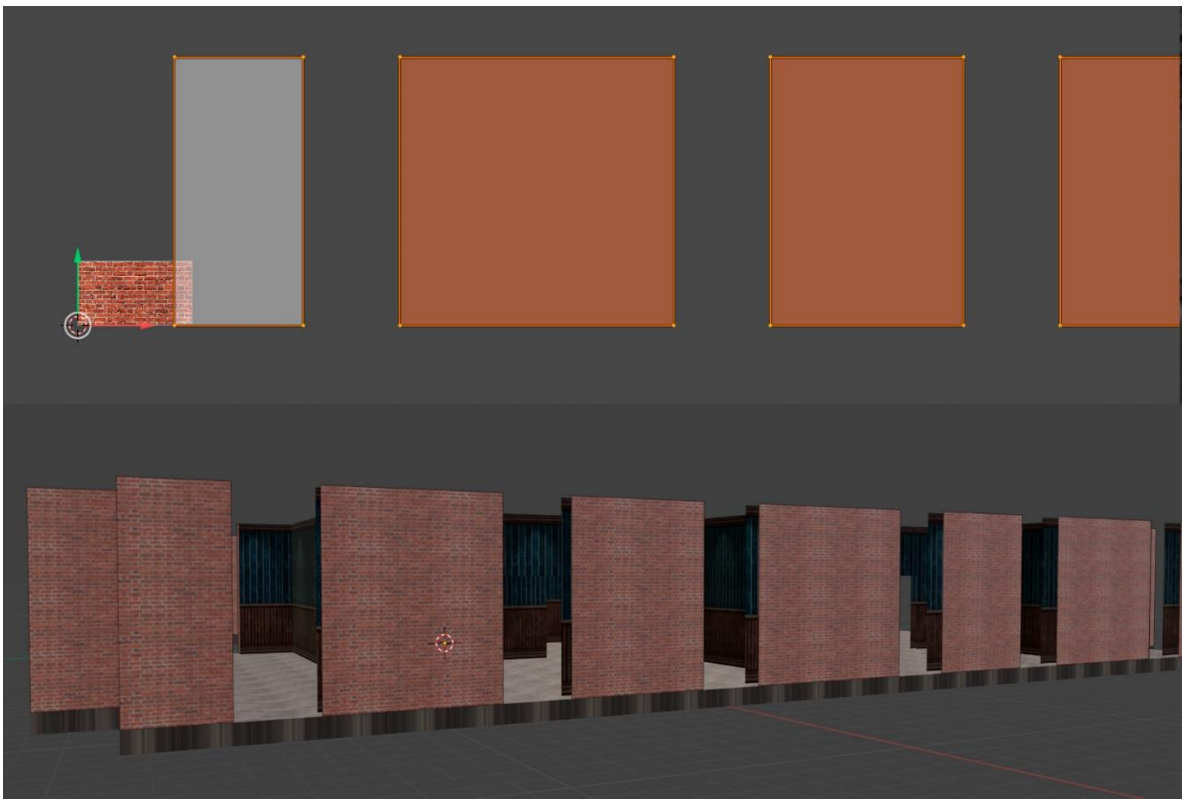


Fig. 36. Mapeado UV con proyección de la malla fuera de las dimensiones UV (arriba). Resultado de la aplicación en el modelo 3D donde el software completa el material rellenando con repeticiones de la textura. (abajo). Elaboración propia.

3.2.1.6.- MODELOS CON ANIMACIÓN

Los modelos 3D, en ocasiones, pueden incluir animaciones que modifican la malla. Estas animaciones suelen afectar a los vértices de los polígonos del modelo 3D, modificando su posición respecto de sus vértices más cercanos, produciendo cambios en la forma y dimensiones de los polígonos en el entorno tridimensional, pero cuando esto sucede, éste no modifica su posicionamiento dentro del espacio 2D del mapa UV ajustando el polígono en la imagen de la textura, por lo que es normal que se visualicen deformaciones de la texturas sobre el modelo 3D cuando la malla está animada debido a los cambios que sufre el modelo (Fig. 37). Es importante tenerlo en cuenta tanto a la hora del modelado, como del texturizado y la posterior animación del modelo.

Fig. 37. Detalle de la modificación de la malla en la animación de un guante que provoca la deformación de la textura. Elaboración propia.

3.2.1.7.- DENSIDAD DE PÍXELES

Cuando realizamos un mapeado UV, estamos asignando una sección de textura a cada polígono de la malla, es decir, estamos indicando cuántos píxeles están encajados dentro de la superficie de cada polígono. Existe un factor que el desarrollador ha de tener en cuenta: La densidad de píxeles. Esta relación hace referencia al número de píxeles contenidos dentro de un polígono respecto de su superficie bidimensional en el entorno 3D y de las dimensiones de la textura. Es decir, un mismo polígono de la malla puede ocupar una superficie mínima sobre el mapeado UV, pero contener una gran cantidad de píxeles debido a la alta resolución de la textura, u ocupar casi toda la superficie disponible para realizar el mapeado UV o más, y contener unos pocos píxeles debido a la baja resolución de la textura (Fig. 38).

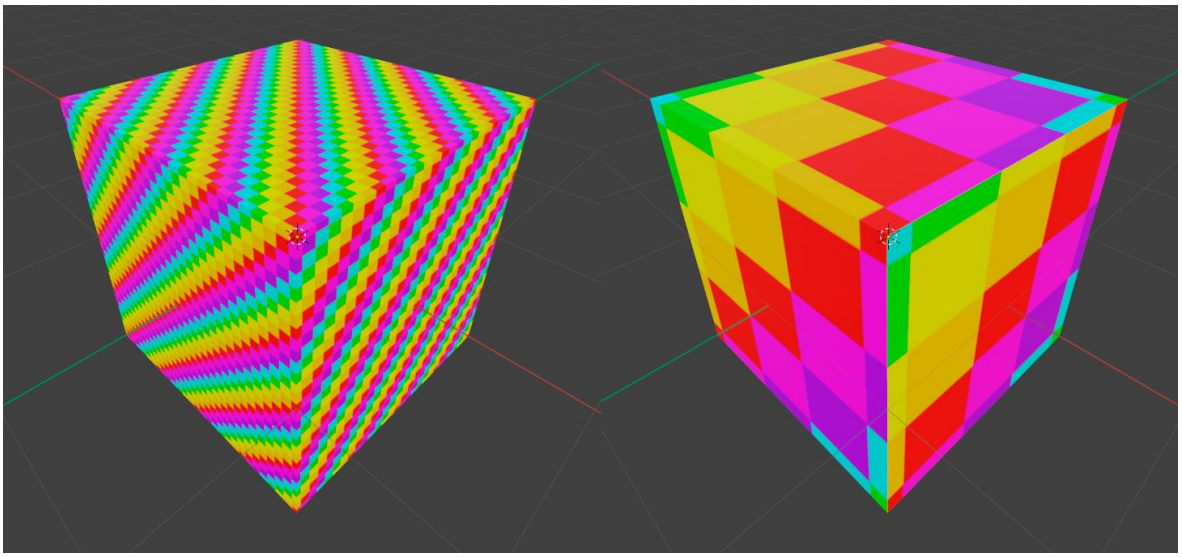


Fig. 38. Diferencia de densidad de píxeles según el mapeado UV sobre un mismo modelo con la misma textura. Alta densidad de píxeles (izquierda). Baja densidad de píxeles (derecha). Elaboración propia.

En una misma escena, puede haber variaciones en la densidad de píxeles en diferentes objetos dependiendo de su importancia en el entorno, pero en un mismo modelo deben evitarse los mapeados con diferentes densidades. Los objetos que pueden verse de cerca suelen requerir una mayor densidad de píxeles, mientras que los objetos que están lejos y nunca podrán verse de cerca pueden tener densidades de píxeles más bajas.

3.2.2.- ENTORNO 3D

Entendemos como entorno 3D el espacio infinito y tridimensional (alto, ancho y profundidad). en el que se diseñará la escena que se mostrará en la imagen renderizada. Dicho espacio es visualizado a través de capturas bidimensionales, ya sean desde un solo punto de vista, o a través de sistemas de estereoscopía o simulación del volumen determinados por las diferentes herramientas de *software* y *hardware*.

Si retomamos el diagrama de flujo para la obtención de una imagen renderizada, comprobamos que hay una serie de elementos indispensables y fundamentales para la creación de dicha escena: una cámara, un objeto e iluminación.

El proceso para la obtención de un modelo tridimensional que ubicar en la escena ya ha sido tratado y enmarcado dentro del contexto de las artes. Lo mismo ocurre con los procesos o componentes para la preparación de una escena en el entorno 3D. La configuración, tanto de la cámara como de la iluminación, corresponden a formación puramente artística dentro de campos como la fotografía, la producción audiovisual o el diseño de interiores, entre otras.

Los diferentes componentes que se van a tratar a continuación tienen funcionamientos y parámetros similares en los diferentes *softwares* existentes en el mercado, por ello podemos estudiarlos y definir conceptos generales sobre los que trabajar, pero es recomendable estudiar los manuales y la documentación propios de cada *software* para ajustar los resultados.

3.2.2.1.- CÁMARA

La cámara juega un papel fundamental en el proceso de obtención de imágenes generadas en entornos 3D, y más aún en experiencias con renderizado en tiempo real, ya no sólo por el hecho de que sea el punto principal desde el que se tomará la imagen a renderizar en cada cuadro devuelto por la aplicación, sino porque, además, pueden (y suelen ser) el principal elemento de interacción del usuario con la obra. Se debe a que éste, a través de los inputs como teclado, ratón, joysticks, dispositivos de RV u otros sistemas, puede controlar la posición y rotación de la cámara en el espacio virtual.

Este tipo de desarrollos permiten una mayor inmersión del usuario en el entorno y mejoran la experiencia, pero también se pueden desarrollar proyectos donde la cámara esté fija, siga un recorrido o trazado, o que se utilicen múltiples cámaras pudiendo cambiar el punto de vista sin desplazamiento. Lo que está claro es que sin cámara no hay imagen.

Todas las imágenes generadas por el proyecto dependerán de la posición y orientación en la que se encuentre la cámara dentro del espacio virtual.

El resultado final que obtendremos en el renderizado de la imagen utilizando una cámara virtual depende de la configuración de los parámetros de la misma. Estos están basados en las funcionalidades de las cámaras físicas reales y dichos parámetros se pueden modificar dependiendo del *software* utilizado.

LoDs

Cuando se trabaja con renderizado en tiempo real, el recuento de polígonos de la escena es fundamental durante la ejecución final del proyecto. Una herramienta muy útil para el desarrollador en escenas de grandes dimensiones y con gran cantidad de modelos 3D son los LoDs (*Levels of Detail* [niveles de detalle]), que permiten la sustitución de un modelo de alto detalle, por uno con menos polígonos, según la distancia a la que éste se encuentra de la

cámara, ya que, al encontrarse a gran distancia del punto de vista, no requieren de tanto detalle.

Esto permite que durante el proceso del *Render Pipeline* en el que se genera la geometría de la escena (*World Space Pipeline)*, la geometría se vea reducida y el proceso sea más rápido con una menor sobrecarga para el equipo y manteniendo un nivel estable de fps.

3.2.2.1.1.- EFECTOS DE POSTPROCESADO

Para lograr una estética concreta, modificar la imagen final resultante del render por pantalla de un parámetro incluido en un *shader* u otros modificadores, se puede recurrir a efectos que no se aplican sobre los propios materiales, sino sobre las propiedades de la cámara, y que afectan a todos los elementos de la escena. Estos pueden estar o no relacionados con la programación de alguno de los *shaders* y dependen del tipo de *Render Pipeline* que se ha configurado para el entorno.

De la aplicación de un efecto o post procesado de cámara se pueden obtener resultados muy distintos sobre una misma escena sin modificar ningún otro parámetro. Además, éstos pueden cambiar durante la ejecución de la obra para generar diferentes sensaciones o efectos según el momento.

Existen muchos y diferentes efectos y post procesados, algunos de ellos son la versión virtual de efectos propios de las cámaras físicas o del proceso de la postproducción audiovisual, codificados para ser simulados en el proceso de renderizado de cada fotograma.

Profundidad de campo

El efecto de profundidad de campo (*Depth of Field* o DoF) simula el enfoque de la cámara desenfocando los diferentes planos de la escena de acuerdo con la profundidad de la escena y a los parámetros configurados.

Este efecto digital simula la profundidad de campo propio de las cámaras físicas, es decir, el espacio anterior y posterior del plano enfocado, que será la zona de máxima nitidez, y que es definido por la distancia focal (distancia entre el centro óptico de la lente y el sensor de la cámara) y la distancia de enfoque (distancia al objeto o plano a capturar).

Modificando los parámetros de este efecto, podemos obtener imágenes digitales que reproducen el efecto de objetivos físicos como el ojo de pez, teleobjetivos o macros, entre otros (Fig. 39).

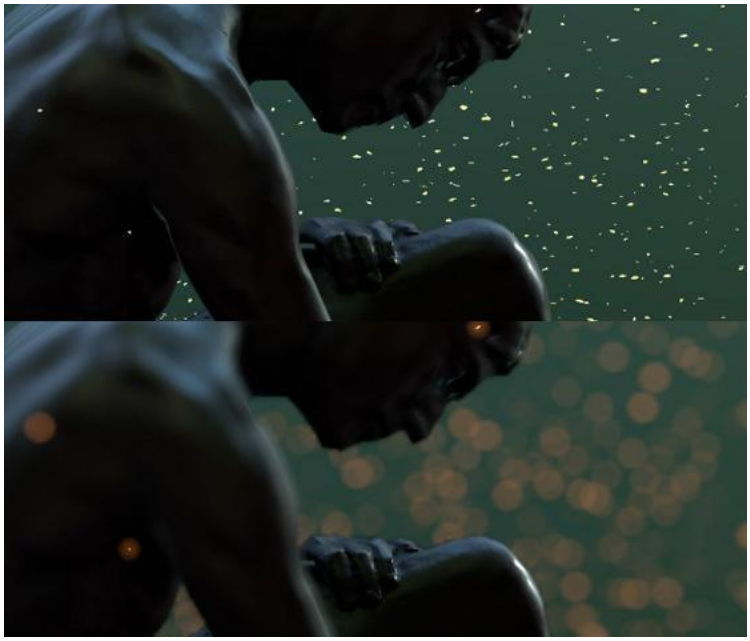


Fig. 39. Imagen captura sin profundidad de campo (arriba) y con profundidad de campo (abajo) - https://docs.unity3d.com/es/2017.4/Manual/PostProcessing-DepthOfField.html

Distorsión de lente

Igual que la profundidad de campo, este efecto recrea la distorsión de lente propia de las cámaras físicas, generando la deformación en la imagen final como si hubiera sido tomada con una lente (Fig. 40). El resultado final dependerá de los parámetros configurados, pudiendo generar diferentes tipos de distorsiones en la imagen.

Fig. 40. Imagen capturada sin distorsión de lente (arriba) y con distorsión de lente (abajo) - https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@14.0/manual/Post-Processing-Lens-Distortion.html

Aberración cromática

La aberración cromática es una distorsión óptica, producto de la limitación de las lentes de la cámara de fotografía o de vídeo cuando, bajo determinadas circunstancias, son incapaces de enfocar todo el rango cromático en mismo plano o punto de convergencia. De este modo, hay colores que aparentemente se plasman no alineados (Fig. 41).

En entornos 3D, las cámaras virtuales pueden imitar este tipo de distorsiones a través de su configuración.

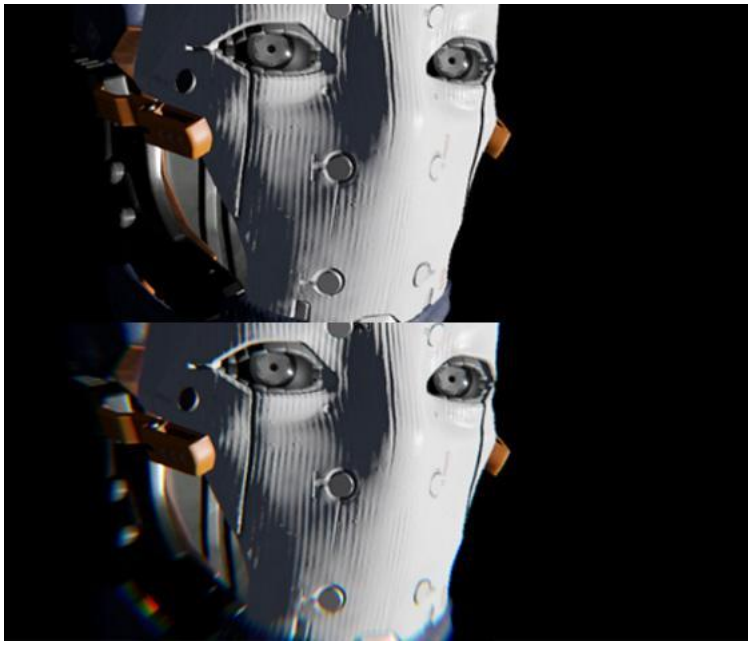


Fig. 41. Imagen capturada sin aberración cromática (arriba) y con aberración cromática (abajo) – https://docs.unity3d.com/560/Documentation/Manual/PostProcessing-ChromaticAberration.html

Ajustes de color

A través de este efecto podemos modificar el tono, el brillo y/o el contraste de la imagen final, o parte de ella. Este efecto es similar al etalonaje que se realiza durante la edición del video en las producciones audiovisuales y permite potenciar el color con fines narrativos o expresivos, modificando los valores iniciales del renderizado (Fig. 42).

Fig. 42. Imagen capturada sin ajustes de color - color grading (arriba) y son ajustes de color (abajo) – https://docs.unity3d.com/560/Documentation/Manual/PostProcessing-ColorGrading.html

Destellos de lente

Lens flare o destello de lente es un efecto digital que simula la dispersión de la luz generada por los objetos brillantes debido a las imperfecciones que puede tener una lente física (Fig. 43).

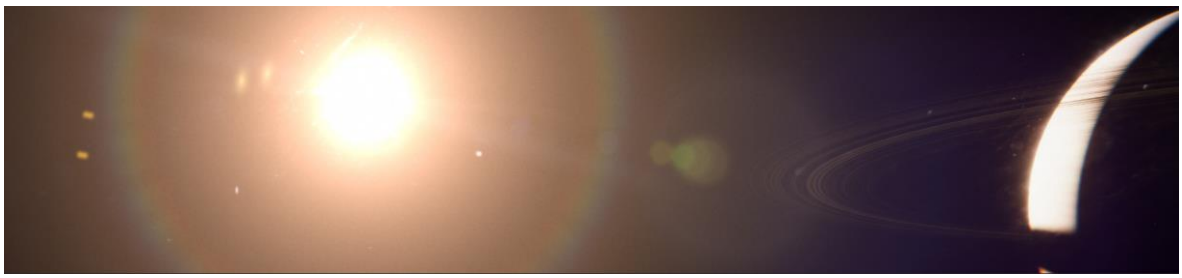


Fig. 43. Efecto de destello de lente en *Unity* - https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@15.0/manual/shared/lens-flare/lens-flare-component.html

Desenfoque de movimiento

Este efecto genera un desenfoque de la imagen en la dirección del movimiento de la cámara o de los objetos que se mueven a gran velocidad (Fig. 44).

Fig. 44. Imagen capturada sin desenfoque de movimiento (arriba) y con desenfoque de movimiento (abajo) - https://docs.unity3d.com/es/2017.4/Manual/PostProcessing-MotionBlur.html

Anti-aliasing (AA)

Es un post procesado que suaviza o difumina las líneas pixeladas de los bordes irregulares, conocidos como dientes de sierra, producidos por la mayor o menor definición de los píxeles de la imagen, y que dependen, en parte, del equipo en el que se vaya a ejecutar la aplicación (Fig. 45).

Existen diferentes tipos de AA:

- **SSAA (*SuperSampling Anti Aliasing*)**: Proceso que está actualmente fuera de uso debido al alto consumo de recursos, genera frames de mayor resolución que luego reajusta a la resolución de pantalla.
- **FXAA (*Fast aproXimate Anti-Aliasing*)**: Este proceso trabaja sobre la imagen ya renderizada buscando los bordes y la modifica aplicando colores aproximados (no exactos). Puede generar desenfoques y manchas, pero consume pocos recursos.
- **MLAA (*Morphological Anti-Aliasing*)**: Proceso similar al FXAA, pero con un mayor consumo de recursos.
- **SMAA (*Subpixel Morphological Anti-Aliasing*)**: Este proceso es una versión mejorada del MLAA, realiza un proceso similar al FXAA con un mejor resultado.
- **MSAA (*MultiSampling Anti-Aliasting*)**: Renderiza solo los bordes a una mayor resolución que la que se mostrará por pantalla, haciendo que, al adaptarse a la resolución final, los bordes queden disimulados.
- **TAA o TSAA (*Temporal Anti-Aliasing*)**: Proceso utilizado por algunas tarjetas gráficas de NVIDIA, analiza los bordes de los fotogramas renderizados en un *buffer* y posteriormente aplica la técnica de suavizado.

Los sistemas de AA que renderizan la imagen o parte de ella a una resolución mayor de la que se muestra por pantalla, suelen dar la opción de elegir el tamaño de la imagen prerrenderizada, por lo que vienen acompañados de un multiplicador (X2, X4, etc.) indicando hasta cuantas veces se quiere doblar el tamaño de dicho render.

Existen más sistemas de AA, como el CSAA o el EQAA, desarrollados por *NVIDIA* y *AMD* respectivamente para sus tarjetas gráficas, y existen programas gráficos y aplicaciones que utilizan combinaciones de diferentes tipos de AA para lograr mejores resultados como el TXAA (MSAA + TAAa 2X) o TSSAA (TAA + SSAA), entre otros.

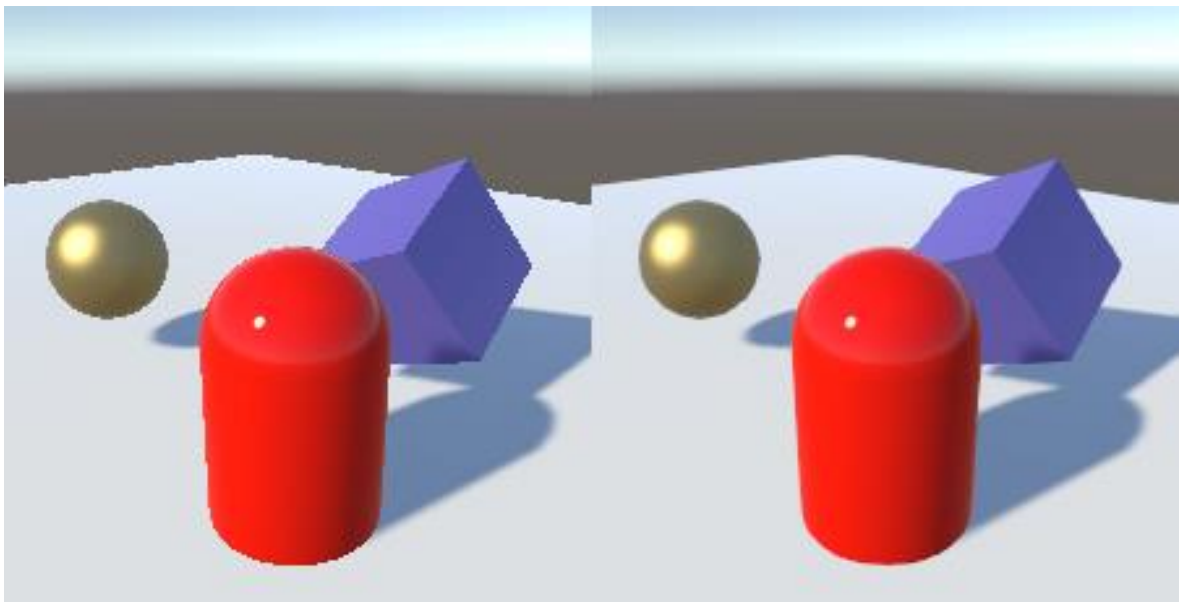


Fig. 45. Efecto del anti-aliasing - https://docs.unity3d.com/es/530/Manual/script-Antialiasing.html

Autoexposición

Es un efecto dinámico que ajusta automáticamente la exposición de la escena a la luz para alcanzar un tono medio simulando la adaptación del ojo a los cambios de la luz. [34]

Bloom

El efecto de *bloom* simula que las superficies u objetos con valores de brillo o emisivos en sus materiales resplandezcan generando un halo a su alrededor. Este efecto trata de dar una veracidad atmosférica a las iluminaciones para simular el efecto de cegarte si lo miras directamente (Fig. 46).

[34] Ejemplo de cómo funciona el efecto de autoexposición en *Unity* https://www.youtube.com/watch?v=6uZ1RUYmhgQ

Fig. 46. Efecto de *bloom* según variación de su parámetro de intensidad en *Unreal Engine* - https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/PostProcessEffects/Bloom/

Dependiendo del *software* en el que se trabaje dicho efecto, poseerá diferentes parámetros que permiten modificar la intensidad, el volumen, dimensiones o color, entre otros. Además, este post procesado tiene relación directa con los materiales PBR.

Oclusión ambiental (AO)

Es un efecto por el cual se oscurecen las áreas que no están directamente expuestas a la luz de la escena (Fig. 47). Hay que diferenciar esta AO de la que se aplica en los materiales a través de una textura. Está oclusión ambiental es generada por la iluminación de la escena y los volúmenes correspondientes a los modelos 3D que hay en ella, mientras que las texturas de AO han sido pregeneradas previamente a este proceso y puede contener detalles de luces y sombras del modelo anteriores a procesos como la retopología.

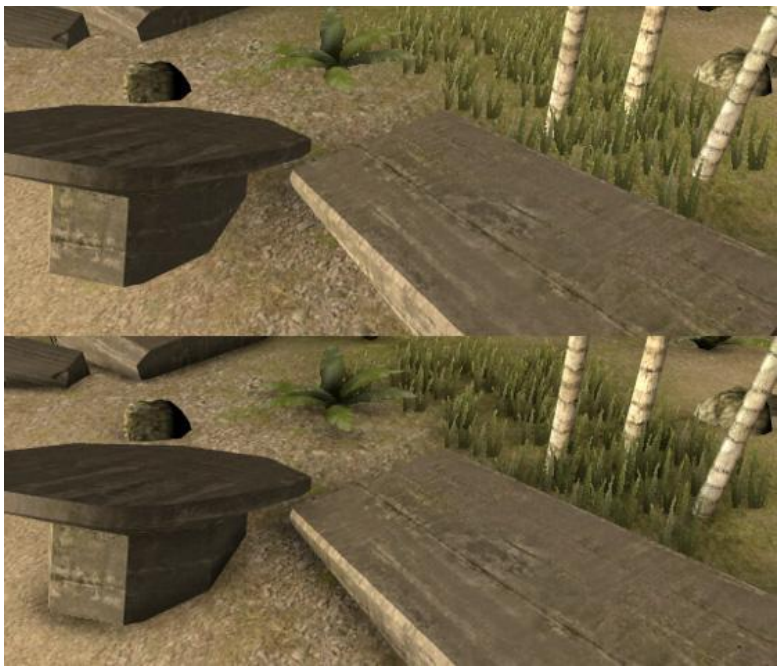


Fig. 47. Imagen capturada sin AO (arriba) y con AO (abajo) – https://docs.unity3d.com/540/Documentation/Manual/script-ScreenSpaceAmbientOcclusion.html

Niebla

La niebla generada de forma virtual se basa en la superposición de un color sobre los objetos dependiendo de la distancia de éstos desde la cámara. El efecto de niebla puede tener diferentes finalidades dentro de nuestro entorno, como generar efecto de perspectiva aérea sobre los elementos más alejados, densidad del aire, atmósferas cargadas o con partículas en suspensión, o inducir sensaciones en el espectador (Fig. 48).

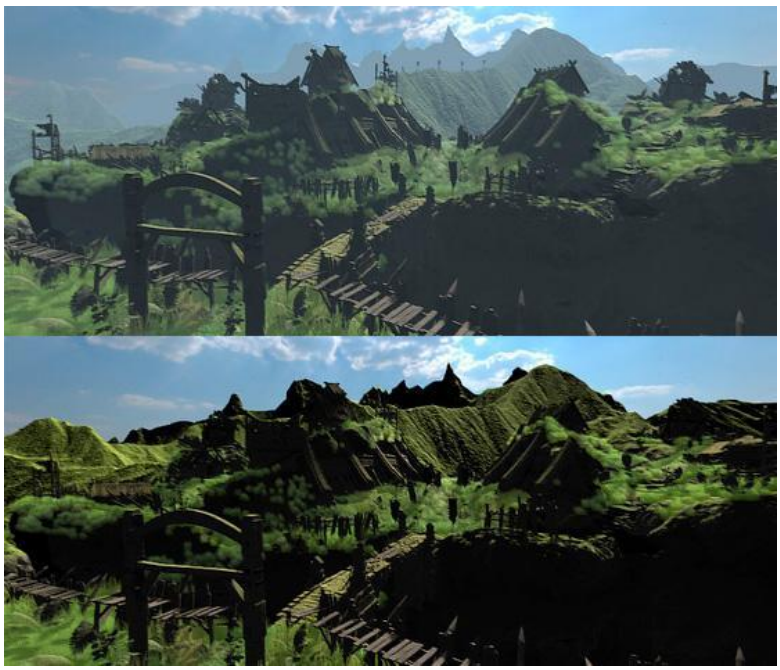


Fig. 48. Misma escena con niebla generando el efecto de perspectiva aérea (arriba) y sin niebla (abajo) - https://docs.unity3d.com/2017.4/Documentation/Manual/PostProcessing-Fog.html

Existen otros efectos y post procesados, como el efecto viñeta, que oscurece los bordes de la imagen desde el centro de esta, la reflexión, el grano o el balance de blancos. Estos, además, pueden variar en sus parámetros y configuración dependiendo del *software* que se esté utilizando, y aunque no es indispensable usarlas para la creación de una escena interactiva, sí que pueden mejorar en gran medida el resultado final dependiendo del objetivo de esta.

3.2.2.2.- ILUMINACIÓN

La luz es un componente fundamental en la creación de cualquier escena virtual para su correcto procesado en 2D. Si no existe ninguna luz en la escena, la cámara virtual capturará todos los objetos de la escena en color negro, salvo que dichos objetos sean los emisores de la luz (parámetro emisivo). Los materiales generados para un modelo 3D, definen cómo la luz va a reaccionar con su superficie, y si además hablamos de materiales PBR, muchos de sus mapas de textura y su efecto visual final dependerá de la configuración de la iluminación.

Durante el proceso de creación de la escena compleja dentro del entorno tridimensional el desarrollador utilizará múltiples elementos para la iluminación de la misma. Los diferentes programas en el mercado proporcionan una serie de generadores de luz virtual (*lights*) con parámetros de configuración muy similares, como el color, la intensidad o el alcance, entre otros.

Cuando se coloca una luz en la escena, ésta puede generar sombras proyectadas, aunque la propiedad para sombras arrojadas puede desactivarse para reducir el consumo de recursos. Las sombras, a su vez, pueden precalcularse (*baked*) durante el proceso de desarrollo y ser estáticas, lo que proporciona una mayor definición y menor consumo de recursos, o, por el contrario, pueden configurarse como luces dinámicas, las cuales son calculadas durante el renderizado de cada fotograma, con la característica de tener una menor definición, necesitar un mayor consumo de recursos y un procesado más lento.

Las escenas más complejas, con diferentes ambientes o que requieren del uso de múltiples generadores de luz, suelen configurarse con una combinación de ambos tipos de iluminación, dinámica y estática, utilizando luces estáticas para la iluminación de la escena, precalculando las luces y sombras, quedando éstas almacenadas en forma de texturas (*lightmaps)*, y luces dinámicas (calculadas en *real-time*) para los elementos no estáticos de la escena, además de recurrir al uso de componentes y complementos que ayudan a generar el efecto de iluminación dinámica, reduciendo el consumo de recursos en la simulación de la iluminación.

3.2.2.2.1.- ILUMINACIÓN GLOBAL (GI)

Gracias a la tecnología descrita en apartados anteriores, los motores gráficos han pasado de una iluminación directa, en la que sólo se calcula una vez cómo la luz incide en las superficies del entorno, a sistemas de iluminación indirecta que calculan cómo la luz rebota sobre los modelos dependiendo de las propiedades y la configuración del material aplicado, y cómo se influyen unos objetos a otros (radiosidad), modificando su aspecto final en el renderizado. A este proceso se le denomina Iluminación global, y suele ser precalculado durante el proceso de creación, antes de la exportación de la obra.

El motor gráfico *Unreal Engine* 5 (*Epic Games*), junto a su sistema Nanite, presentó un nuevo medio de iluminación llamado Lumen[35] (Fig. 49), un sistema de GI dinámico y de generación de reflejos sobre superficies reflectantes a través del cálculo de rebotes infinitos y reflejos especulares.

Fig. 49. Captura de *Lumen in the Land of Nanite.* Aplicación de Lumen para el cálculo de la GI en tiempo real. - https://docs.unrealengine.com/5.0/en-US/lumen-global-illumination-and-reflections-in-unreal-engine/

[35] Web del motor gráfico *Unreal engine 5 - Lumen Global Illumination and Reflections* - https://docs.unrealengine.com/5.0/en-US/lumen-global-illumination-and-reflections-in-unreal-engine/

3.2.2.2.2.- TIPOS DE EMISORES DE LUZ VIRTUAL

Los diferentes programas proporcionan componentes de escena para la generación de luz. Estos están diseñados para simular diferentes tipos de luz que podemos encontrar en el mundo real.

Luz General

La *Ambient light* o *Sky light* es el primer tipo de luz que debemos conocer. Es una luz general que se aplica a todo el entorno 3D sin atenuación y sin tener un punto concreto de origen. Este tipo de iluminación posee múltiples parámetros de configuración según el *software* utilizado, como color, intensidad o una textura para la simulación del cielo. Este tipo de luz aplica un tono constante a los objetos de la escena, y aunque no genera sombras, ya que no tiene un origen ni dirección concreta, sí influye en el cálculo de la AO.

Puntos de Luz

Los puntos de luz o *point lights* son elementos emisores en la escena 3D que generan rayos de luz omnidireccionales. Cabe destacar que cada punto emite luz desde su posición y ésta perderá intensidad a medida que aumenta la distancia respecto a su origen (Fig. 50).

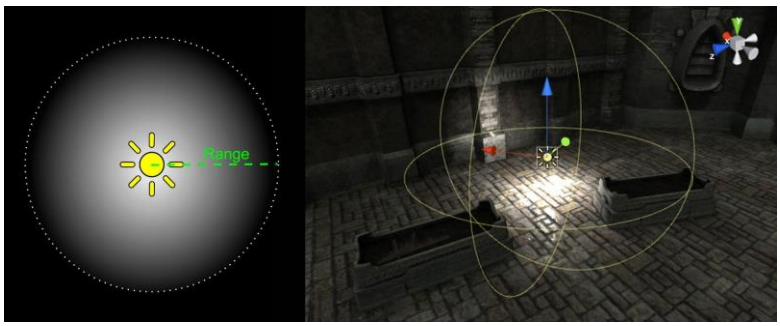


Fig. 50. Diagrama y ejemplo de un Punto de luz - Documentación de *Unity - Point Light* - https://docs.unity3d.com/Manual/Lighting.html

Focos

Estas luces, llamadas *spot light*, son similares a las *point light,* pero sólo emite luz dentro de un ángulo concreto, dando como resultado un cono de iluminación que perderá intensidad respecto de su origen. (Fig. 51).

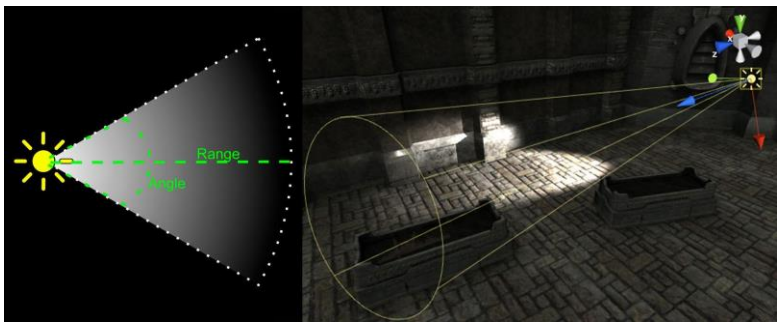


Fig. 51. Diagrama y ejemplo de una luz de foco - Documentación de *Unity - Spot Light* - https://docs.unity3d.com/Manual/Lighting.html

Luces Direccionales

A diferencia de los puntos de luz y los focos, no tienen un punto de origen en el espacio, y genera rayos de luz paralelos en una única dirección de forma infinita, sin atenuación por la distancia y de forma constante, por todo el entorno virtual. Al no tener un punto de origen, da igual su posición en el espacio, y sólo influirá en la iluminación su orientación, que indicará la dirección en la que se van a trazar los trazos de luz y las sombras. El efecto visual de esta iluminación es similar a la luz del sol (Fig. 52).

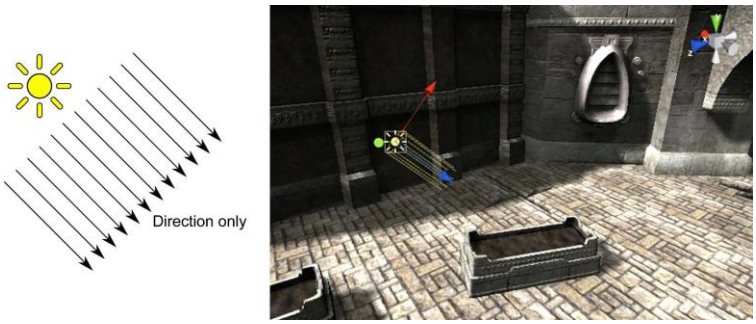


Fig. 52. Diagrama y ejemplo de una luz direccional - Documentación de *Unity - Directional Light* - https://docs.unity3d.com/Manual/Lighting.html

Softbox o Pantallas

Denominadas como *area lights* o *rect lights*, este tipo de iluminación o emisores de luz delimitan una superficie bidimensional (alto y ancho) en el espacio virtual, que emite luz desde uno de sus lados. El aspecto de este tipo de iluminación es similar al de una *softbox* de estudio, o a la luz que generaría un monitor o un cartel luminoso.

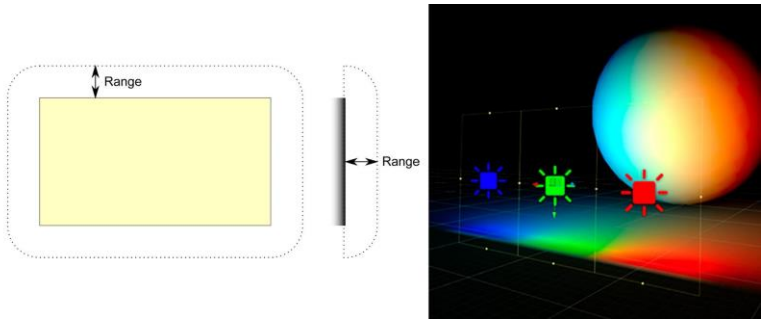


Fig. 53. Diagrama y ejemplo de una luz *softbox* o de pantalla- Documentación de *Unity - Area Light* - https://docs.unity3d.com/Manual/Lighting.html

Filtros de luz

Por defecto, los puntos de luz descritos anteriormente proyectan luz uniformemente, dentro de los parámetros propios de cada uno, proyectando sombras y radiosidad al colisionar sobre las superficies de los objetos en escena. Una forma de modificar el aspecto de la luz y hacerla más realista es a través del uso de filtros (*cookies*). Estos filtros son texturas que filtran la luz y se utilizan para simular diferentes efectos de iluminación (Fig. 54).

Fig. 54. Efecto de luz usando un filtro (*cookie*) en *Unity* - https://docs.unity3d.com/Manual/Cookies.html

3.2.2.3.- ESCENA

La escena dentro de un entorno 3D es la combinación de los elementos vistos previamente: objeto/s de escena e iluminación que serán capturados por la cámara.

Los elementos dentro de la escena pueden tener diferentes configuraciones y parámetros que modificarán el resultado final y el consumo de recursos a nivel técnico, y el autor ha de encontrar el equilibrio para alcanzar resultados óptimos de acuerdo con el tipo de resultado que quiere mostrar, configurando y ajustando la relación entre los diferentes componentes.

Uno de los factores que más influye en este equilibrio son los objetos tridimensionales en escena, o más bien, el número de polígonos del que están compuestos y su configuración. Ya hemos tratado los diferentes tipos de mallas según su número de polígonos (*low poly*, *mid poly*, *high poly*), pero insertar en la escena un número considerable de objetos de bajo poligonaje provocará igualmente un colapso en el *Render Pipeline* y una reducción en los fps, por lo que hay que tener en cuenta este factor a la hora de desarrollar una escena y apoyarnos en herramientas como los LoDs, o la reducción de polígonos a través de retopología, entre otras.

Cada elemento lumínico en escena 3D tiene su configuración individual, que está estrechamente ligada a la configuración de los objetos en escena, ya que las luces estáticas serán precalculadas, pero esto sólo importará si los objetos de la escena son también estáticos, es decir, la iluminación estática no tendrá en cuenta los elementos no configurados como estáticos y su iluminación dependerá de otros elementos. Además, el material aplicado al modelo tridimensional afectará al modo en que la luz interactúa en la escena, y estos mismos objetos, a su vez, pueden ser emisores de luz complementando la iluminación, como veremos más adelante en el apartado de materiales.

Aunque como hemos visto, este tipo de configuraciones y factores pueden ir desapareciendo o caer en desuso con la evolución tecnológica, hemos de tener en cuenta que afecta y afectará

a cierto tipo de desarrollos y estéticas, y seguirán existiendo proyectos o escenas que requieran de este tipo de configuraciones.

Existen muchos otros componentes que pueden ser insertados en la escena dependiendo del uso que se le dé a la aplicación, por ejemplo, una interfaz de usuario, sistemas de partículas, o sistemas de audio (emisores y receptores). De entre todas las opciones que pueden componer una escena 3D, estamos tratando los elementos fundamentales que intervienen en el aspecto visual de los objetos y los materiales implicados, y que intervienen directamente en la imagen final renderizada.

3.2.3.- TEXTURIZADO

Dentro del esquema de flujo de trabajo para la obtención de cada una de las imágenes producidas en un entorno con renderizado en tiempo real, procede continuar analizando el texturizado de los elementos tridimensionales de la escena.

En este apartado trataremos el proceso completo actual para la generación de materiales en entornos con renderizado en tiempo real. Los avances tecnológicos en este campo, orientados por el cine y los videojuegos fundamentalmente a la simulación visual fotorrealista, no implica que no se utilicen otros tipos de materiales para obtener otras estéticas, sino que el objetivo final de la evolución técnica es la simulación de las físicas del mundo real en el virtual, y en este momento ese objetivo se encuentra materializado en los materiales PBR.

La forma habitual de aprender a trabajar, crear y configurar materiales en el ámbito del diseño y el desarrollo de entornos 3D es a través de recetas (Fig. 55), es decir, utilizar parámetros preconfigurados para la obtención de resultados prediseñados, aprendiendo de lo que otros ya han hecho y copiando sus resultados, o bien por ensayo error, es decir, ir variando la configuración y los parámetros hasta alcanzar el objetivo deseado o una aproximación.

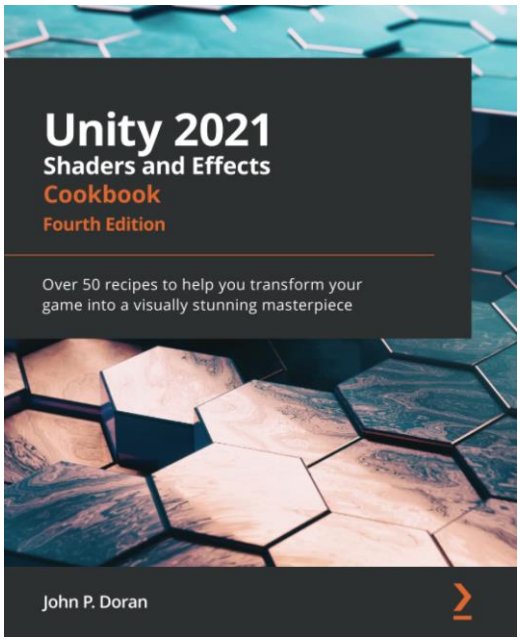


Fig. 55. Portada del libro *Unity* 2021 - *Shader and Effects* donde literalmente podemos leer "*cookbook - Over 50 recipes …"* [libro de cocina - Más de 50 recetas …]. DORAN, J.

Los motores gráficos más populares del mercado, como *Unity* y *Unreal Engine*, poseen editores de materiales, en los que se puede configurar un material a través de nodos, totalmente orientado para diseñadores (*Shader Graph* en *Unity* (Fig. 57) / *Material editor UI* en *Unreal Engine* (Fig. 58)), que permite llevar la creación de materiales un paso más allá, trabajando de forma mucho más profunda e intuitiva con los diferentes parámetros y valores implicados en la definición de los materiales.

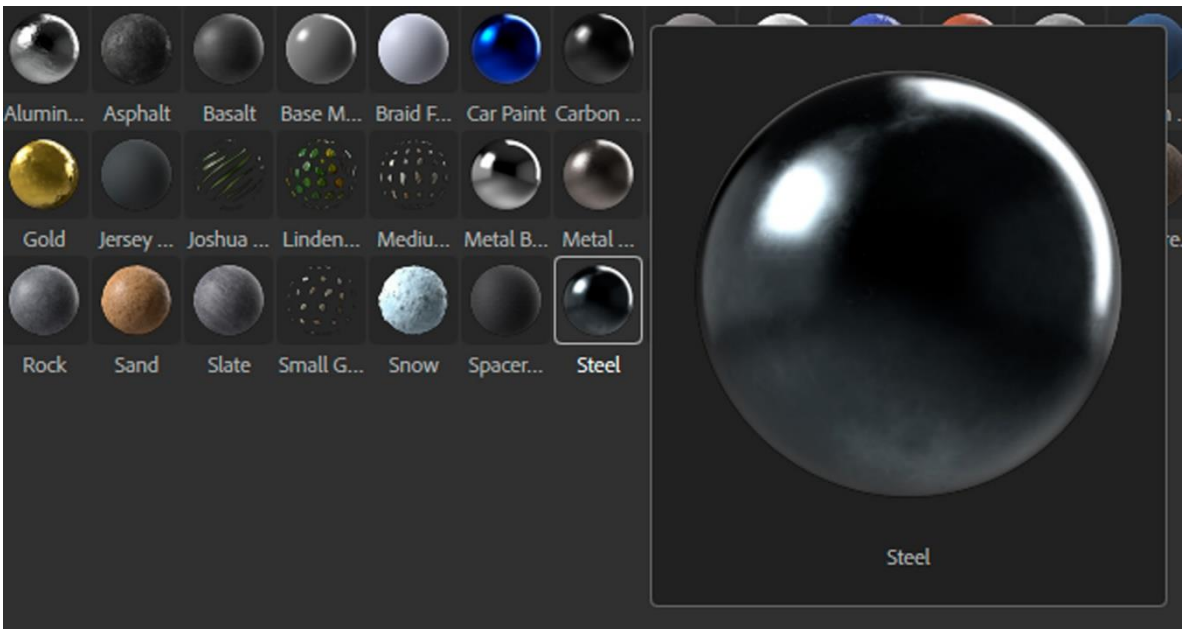


Fig. 56. Ejemplos de materiales PBR - *Adobe substance 3D*.

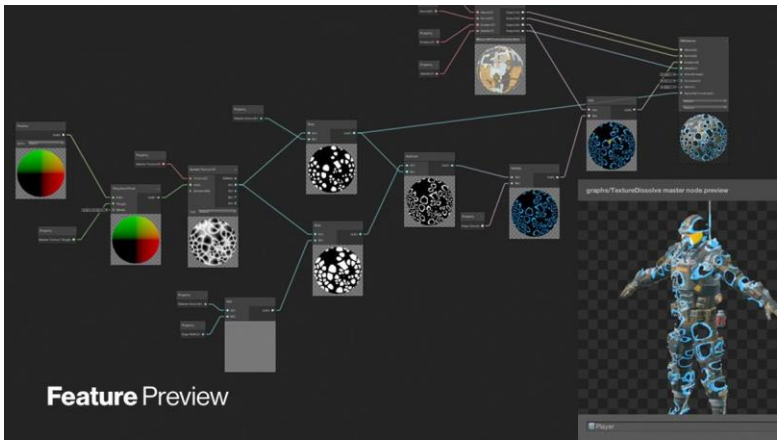


Fig. 57. Editor de Materiales *Shader Graph* de *Unity* – https://blog.unity.com/technology/introduction-to-shader-graph-build-your-shaders-with-a-visual-editor

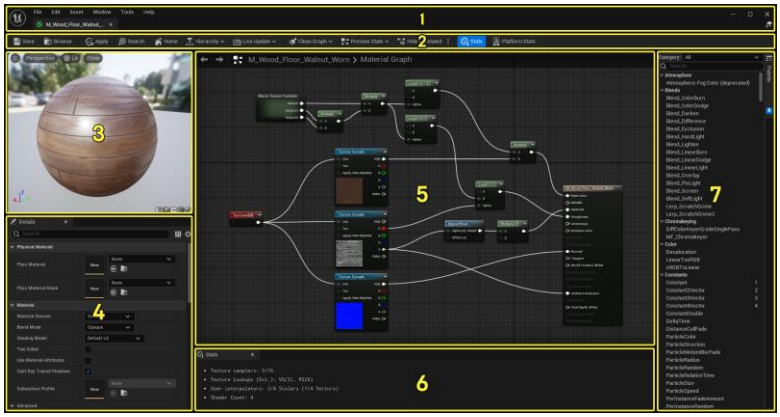


Fig. 58. *Material editor UI* de *Unreal Engine* - https://docs.unrealengine.com/5.0/en-US/unreal-engine-material-editor-ui/

3.2.3.1.- MATERIALES PBR

Como veremos a continuación, dependiendo del *shader* que apliquemos en un material podremos obtener diferentes resultados visuales al finalizar el proceso de renderizado.

Los materiales PBR (*Physically Based Rendering*) en los que se centra esta tesis, son materiales basados en la simulación de la física de la luz del mundo real en las superficies de los objetos 3D en los entornos virtuales, en concreto, el comportamiento con respecto a la luz dependiendo de sus propiedades físicas, lo que lo convierte en el método más eficiente actualmente para obtener simulaciones de materiales realistas.

Los desarrolladores de *software*, los últimos responsables de la codificación e implementación de las fórmulas necesarias para esa simulación, diferencian dos tipos de materiales (Fig. 59):

- **Materiales dieléctricos**: Tienen baja conductividad eléctrica o son aislantes.
- **Materiales metálicos**: son principalmente conductores de la electricidad.

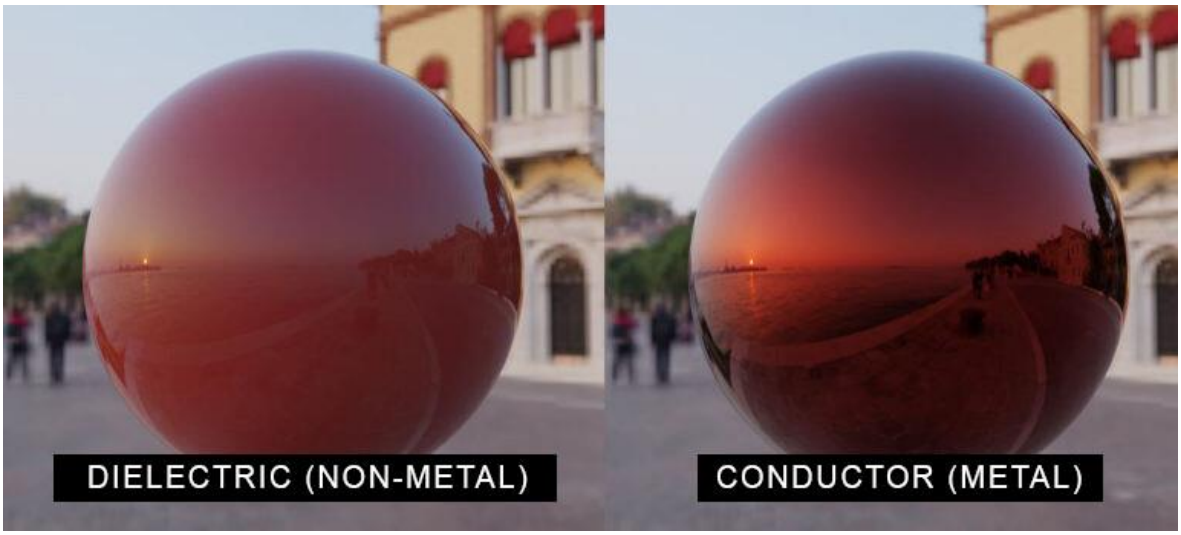


Fig. 59. Diferencia de aspecto entre un material Dieléctrico y uno Metálico - ArtStation
https://cdnb.artstation.com/p/media_assets/images/images/000/706/215/large/5.jpg?1609633130

El estudio de la reacción de los materiales en el mundo real a la electricidad sirve como referencia para el estudio de su reacción a la luz, y para desarrollar simulaciones virtuales de dichos materiales.

Esto ha permitido que los desarrolladores de *software* generen materiales capaces de simular diferentes tipos de superficies con propiedades totalmente diferentes a través de la codificación e implementación de fórmulas físicas a través de los lenguajes de sombreados (ya tratados anteriormente) y los *shaders*, con el añadido de tener que ser ejecutados en fracciones de segundo durante la ejecución de una obra realizada en un entorno con renderizado en tiempo real.

3.2.3.2.- *SHADERS*

Un material está definido por un *shader*, y un *shader* es la codificación de componentes y parámetros que configurarán el aspecto del material, y del modo en el que la luz interactúa con la superficie del modelo 3D sobre el que se aplique el material.

Dicha codificación, como ya se ha explicado en apartados anteriores, se puede realizar en diferentes lenguajes (HLSL, GLSL, CG, etc.), dependiendo de las herramientas a utilizar para el desarrollo y ejecución de la obra (Fig. 60).

```
Shader "MyShadersTutorials/01/MyFirstShader"
{
    Properties
    {

    }

    SubShader
    {
        Pass
        {
            CGPROGRAM

            void VertexFunction()

            {

            }

            void FragmentFunction()
            {

            }

            ENDCG
        }
    }
}
```

Fig. 60. Estructura del código de un *shader* en HLSL/CG- https://www.ighniz.com/2020/01/03/hlsl-unity3d-programacion-de-shaders-en-unity/

Los *shaders* realizan cálculos gráficos durante el *Screen Space Pipeline* modificando el aspecto final de la imagen, trabajando con los datos de la geometría que configuran los volúmenes de los objetos tridimensionales (*vertex shader*/*geometry shader*) y con cada píxel o fragmento de dichos modelos en la escena (*fragment shader* / *pixel shader*).

Un material puede estar definido por un *shader* que únicamente codifique un color para toda la geometría, y dependiendo de la configuración de la iluminación de la escena, cuando ésta incida y rebote (o no) sobre la superficie del modelo 3D, la cámara virtual captará la reacción de la luz sobre la superficie del modelo según la configuración del *shader* y de la escena para generar la imagen.

Existen muchos parámetros y propiedades a configurar en un *shader*. A continuación, procedemos a describir los principales factores que los definen y su funcionamiento.

3.2.3.2.1.- SHADER LIT / UNLIT

Lit, del pasado del verbo *to light* en inglés, implica que el material se verá afectado por la luz, es decir, su superficie será modificada por los diferentes efectos de luces y sombras en la escena, lo que significa que su superficie formará parte del proceso del cálculo de la iluminación de la escena.

En ocasiones, se insertan objetos que, por diferentes motivos, no deben o no importa si se ven afectados por la luz, lo que contribuye a reducir el consumo de recursos técnicos que el autor ha de seguir teniendo presente.

A estos materiales que poseen un *shader* que no interactúa con la luz se los denomina como *Unlit*, y su superficie no será modificada por los efectos de la iluminación, aunque el resto de los objetos de la escena sí se verán afectados por su presencia y sus correspondientes modificaciones sobre la iluminación, dependiendo de la configuración de la misma (Fig. 61).

Fig. 61. Modelo de un cubo a la izquierda con un material configurado *Unlit*, y a la derecha con un material *Lit*, iluminados con luz ambiental para generar la AO. Elaboración propia.

3.2.3.2.2.- MODELOS DE SIMULACIÓN DE LA LUZ

Los modelos de iluminación en entornos virtuales tienen en cuenta dos aspectos a la hora de simular la luz sobre las superficies de los objetos virtuales.

Por un lado, el modelo de interpolación es el modo en que se calcula el ángulo de reflexión de la luz sobre la superficie. Recordemos que los modelos 3D están compuestos de una malla formada por polígonos: cuando la luz incide sobre la superficie, podemos obtener diferentes efectos según cómo se calculan los ángulos de reflexión de la luz y su interpolación (Fig. 62).

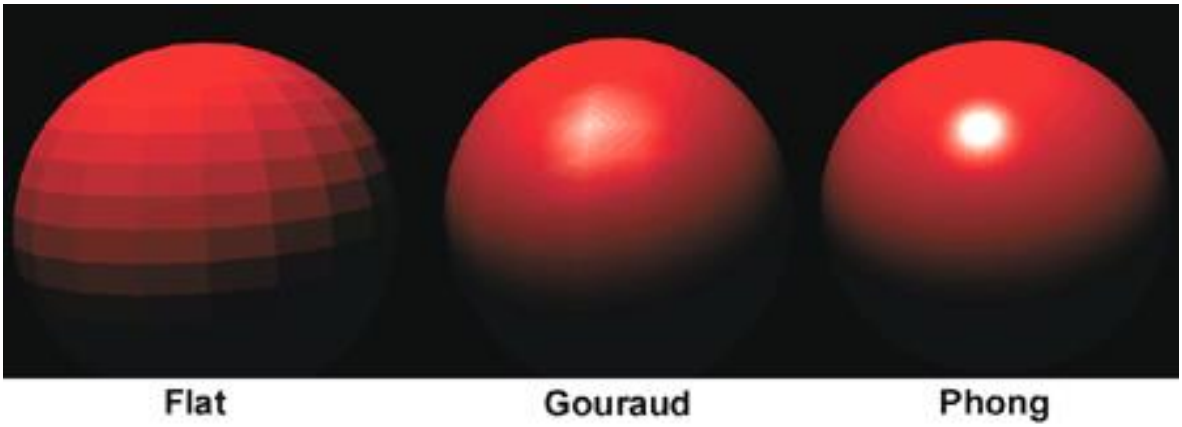


Fig. 62. Diferentes efectos de una misma luz sobre la superficie de un modelo 3D según la fórmula de interpolación. [36]

El *Flat shading* calcula un único ángulo de incidencia de la luz respecto de la normal de cada polígono (vector perpendicular a la superficie del polígono), y calcula la iluminación del modelo respecto a esos datos.

El *Gouraud shading* calcula la incidencia de la luz en cada vértice de cada polígono y realiza un renderizado *per-vertex* de la iluminación en cada vértice tras realizar una interpolación entre los ángulos del mismo polígono.

[36] Consultar referencia - https://opengl-notes.readthedocs.io/en/latest/topics/lighting/shading.html

El *Phong shading*, calcula la incidencia de la luz en cada vértice del polígono, igual que el *Gouraud*, pero posteriormente interpola los datos obtenidos y realiza una aproximación per-*pixel*, interpolando los datos de luz de cada uno con los datos de los píxeles cercanos.

Por otro lado, según la interacción de la luz con la superficie, definimos tres tipos de reflexiones: reflexión Difusa, reflexión *Specular* (Fig. 63) o SSS (*Subsurface Scattering*)

La reflexión Difusa es aquella que modifica los ángulos matemáticamente correctos de reflexión de los rayos de luz que inciden sobre la superficie del objeto y dispersa la luz sobre la superficie en diferentes direcciones. Para más datos, este sistema suele utilizar el modelo de *Lambert* para el cálculo de la luz, aunque existen otras fórmulas.

La reflexión *Specular* es aquella que genera la reflexión de los rayos de luz que inciden sobre la superficie sin apenas variación con respecto al ángulo matemáticamente correcto. Estos cálculos se realizaban en base al modelo de reflexión de *Blinn-Phong* (evolución del modelo *Phong*), aunque también existen otras fórmulas.

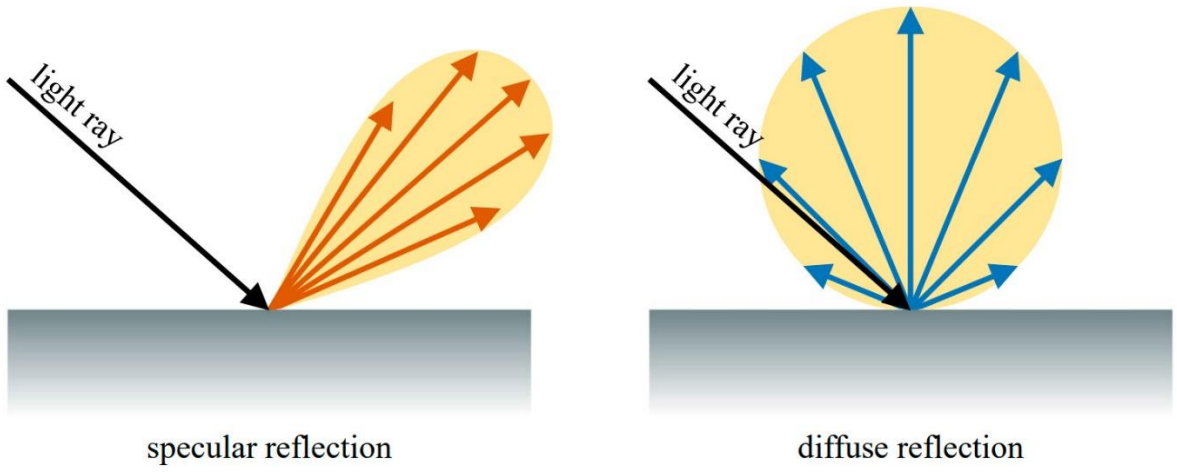


Fig. 63. Diferencia entre la reflexión de la luz sobre una superficie especular y una superficie difusa - PARK, S. y BAEK, N., (2021) *A Shader-Based Ray Tracing Engine*.

Las superficies SSS (*Subsurface Scattering*) permiten que la luz penetre bajo la superficie del modelo, y calcula el movimiento de los trazados de luz atravesando la malla, simulando la

dispersión de la luz, proporcionando información sobre el color interior del objeto. Este tipo de superficies se usa con un mapa de textura denominado como espesor (*thickness*).

A nivel estético, una reflexión *Specular* corresponderá a la de un espejo o una superficie pulida, la Difusa es más propia de superficies ásperas, mientras que las superficies SSS son propias de elementos orgánicos o con una superficie translúcida.

Existen variaciones de estos modelos de iluminación, como el cel (*celluloid*) *shading*, también llamado *toon shading*, ya que imita el estilo de los cómics o de los dibujos animados. Este sistema calcula la iluminación para cada píxel del objeto y posteriormente los remapea o distribuye en bloques con valores máximos y mínimos haciendo desaparecer el difuminado entre secciones según su iluminación.

3.2.3.2.3.- *SHADERS* Y TRANSPARENCIA

Hablamos de transparencia en entornos 3D cuando, desde la posición de la cámara, observando un polígono de malla con su lado visible orientado hacia la cámara (normal del polígono), podemos ver lo que hay detrás del polígono. Existen diferentes tipos de transparencia, dependiendo del efecto visual que se quiera obtener, y diferentes parámetros de configuración que se tratarán más adelante.

3.2.3.3.- TEXTURAS

Los *shaders* pueden requerir del uso de una o más texturas para completar la configuración de un material. Dichas texturas son mapas de bits bidimensionales que, según la configuración del *shader,* han de tener unos parámetros específicos para producir el efecto deseado sobre la superficie del modelo. Todos estos parámetros los trataremos más adelante.

3.2.3.3.1.- RESOLUCIÓN

La resolución de las texturas es el tamaño en píxeles de la imagen bidimensional que, a diferencia de las imágenes convencionales, cuya resolución se mide en píxeles por pulgada (ppp) según su uso, éstas se nombran por la longitud en píxeles de sus lados.

Lo habitual es trabajar con imágenes cuadradas. Cuando decimos que una textura es de 256 píxeles, estamos hablando de una imagen de aspecto cuadrado de 256 píxeles de ancho por 256 píxeles de alto (Tabla 3).

En la actualidad, ya no es necesario trabajar con resoluciones de imagen cuadradas, y tampoco es necesario que las dimensiones sean potencias de 2. Esta forma de trabajar se debe a que en las antiguas tarjetas gráficas estaba limitado el almacenamiento de imágenes en memoria, por ejemplo, de 256 X 256 píxeles, y el almacenamiento en dispositivos electrónicos funciona con este sistema de potencias de 2. Cuando hablamos 1 GB, en realidad estamos haciendo referencia 1024 MB de espacio, potencia de 2 (2^{10}).

Tabla 3. Resoluciones (aspecto cuadrado) más utilizadas en la actualidad y su superficie en pixeles. Elaboración propia.

RESOLUCIÓN	TAMAÑO
256	65.536 píxeles
512	262.144 pixeles
1024	1.048.576 píxeles
2048	4.194.304 píxeles
4096	16.777.216 píxeles

A pesar de la libertad actual que hay en la resolución a la hora de generar texturas, éstas siguen ocupando espacio, y una gran cantidad de texturas en la escena, si son de una gran resolución, pueden colapsar el equipo reduciendo el recuento de fps.

La resolución de las texturas debe generarse atendiendo al tamaño del objeto y a su importancia en la escena y en la imagen final, ya que el objetivo es conseguir una óptima densidad de píxeles en todos los elementos de la escena. En ocasiones, para optimizar al máximo el número de texturas de un proyecto, se utilizan imágenes complejas que se pueden aplicar en los materiales de varios objetos de la escena, de modo que se reduce el número de texturas en el proyecto y el uso de memoria durante el procesamiento en ejecución.

3.2.3.3.2.- CREACIÓN DE TEXTURAS

Las texturas son imágenes digitales que se utilizarán en los diferentes mapas que el *shader* pueda tener configurados para definir el aspecto final del material, y estos pueden tener orígenes muy diferentes.

Imágenes aleatorias

Una forma de obtener mapas de texturas puede ser a través de un generador de imágenes procedural, ya sea con parámetros que delimiten el resultado, o no. El autor no tiene control directo sobre el resultado y pueden tener diferentes usos. Los programas que poseen herramientas para crear y trabajar con materiales suelen incluir diferentes generadores de este tipo de imágenes, puede que, hasta animadas, para la simulación de diferentes efectos.

Fotografías e imágenes digitalizadas

Otra forma de obtener una textura, sobre todo para la realización de entornos fotorrealistas, es a través de fotografías del mundo real. Estas imágenes deben ser tomadas en unas condiciones específicas y con una configuración concreta y es habitual tener que retocarlas en algún programa de edición de imágenes. Existen repositorios online donde acceder a contenido de muy buena calidad y diferentes resoluciones.

Cabe mencionar, a pesar de su baja incidencia, que existe la posibilidad de obtener texturas a partir de imágenes digitalizadas a través de un escáner u otros sistemas de captura. En este sentido, actualmente en el mercado profesional y doméstico existen varios modelos de escáneres 3D, que además de proporcionar el modelo tridimensional, proporcionan capturas de la superficie del modelo para construir sus texturas.

Imágenes renderizadas y precalculadas (*baked*)

Los propios editores de modelos 3D y programas con capacidad de renderizado permiten obtener imágenes que luego podemos utilizar como texturas. Habitualmente, estos programas nos permiten realizar procesos de *bake* de un objeto sobre otro (diferente al *bake* de luces). Este proceso permite exportar detalles de un moldeado y sus materiales a mapas de textura. Por ejemplo, podemos exportar detalles del relieve de un modelo 3D a una textura de normales que podemos aplicar a otro modelo reduciendo el número de polígonos del modelo.

Imágenes editadas / diseñadas

Denominamos como imágenes editadas o diseñadas, o texturas en nuestro caso, a aquellas creadas, o que han sido modificadas, en un editor de imágenes 2D de acuerdo a las necesidades del artista, ajustando su aspecto hasta el último píxel. A diferencia de otras texturas que siempre pueden tener un cierto factor aleatorio, estas texturas se realizan de forma específica a través de un editor de imágenes.

Es habitual que los mapas de textura obtenidos por alguno de los otros medios mencionados anteriormente pasen por algún proceso de edición para ajustar alguno de sus parámetros o realizar modificaciones.

3.2.3.3.3.- ENCAJE DE TEXTURAS EN EL MAPEADO UV

Todos los modelos tridimensionales tienen un mapeado UV, esto quiere decir que las texturas se ubicarán o representarán sobre el modelo 3D de acuerdo con la disposición de la geometría del modelo en dicho mapeado UV, creando una relación entre la malla 3D y la textura.

En ciertos modelos no es fundamental para su aspecto el encaje de la textura en el mapeado UV, por ejemplo, a la hora de texturizar una piedra genérica de una escena, pero en otros casos sí es necesario que algunos elementos o píxeles concretos de la textura se dibujen en polígonos concretos de la malla.

Para conseguir este encaje, es habitual exportar una referencia del mapeado UV desde el *software* de edición 3D. Para ello, se colorean en una imagen o mapa de textura generado en el propio programa todos los contornos de los polígonos (aristas y vértices) de la malla sobre dicha textura, y se exporta como una imagen que puede ser abierta en un editor de imágenes 2D. Con esta imagen, el editor de las texturas tiene acceso a la malla desplegada que puede ser usada como referencia para realizar dicho encaje y la edición de las texturas.

Otro término importante que interviene en el encaje de las texturas con el mapeado UV es la concha o *shell* (Fig. 64). Más utilizado en texturas de baja resolución, se trata de un margen o borde que continúa la textura o mantiene el color y tono alrededor de los diferentes bloques de polígonos del mapeado UV para evitar que, al representarse la textura sobre el modelo 3D, queden espacios sin texturizar, estropeando el cosido del mapeado UV sobre el modelo. Es decir, al encajar durante el mapeado UV un polígono (vectorial) sobre una textura (rasterizado), éste puede estar en posición oblicua sobre los píxeles de las imágenes, por lo que el *software* puede reconocer como parte de la textura a representar píxeles que hayan quedado fuera del polígono de la malla y sin texturizar.

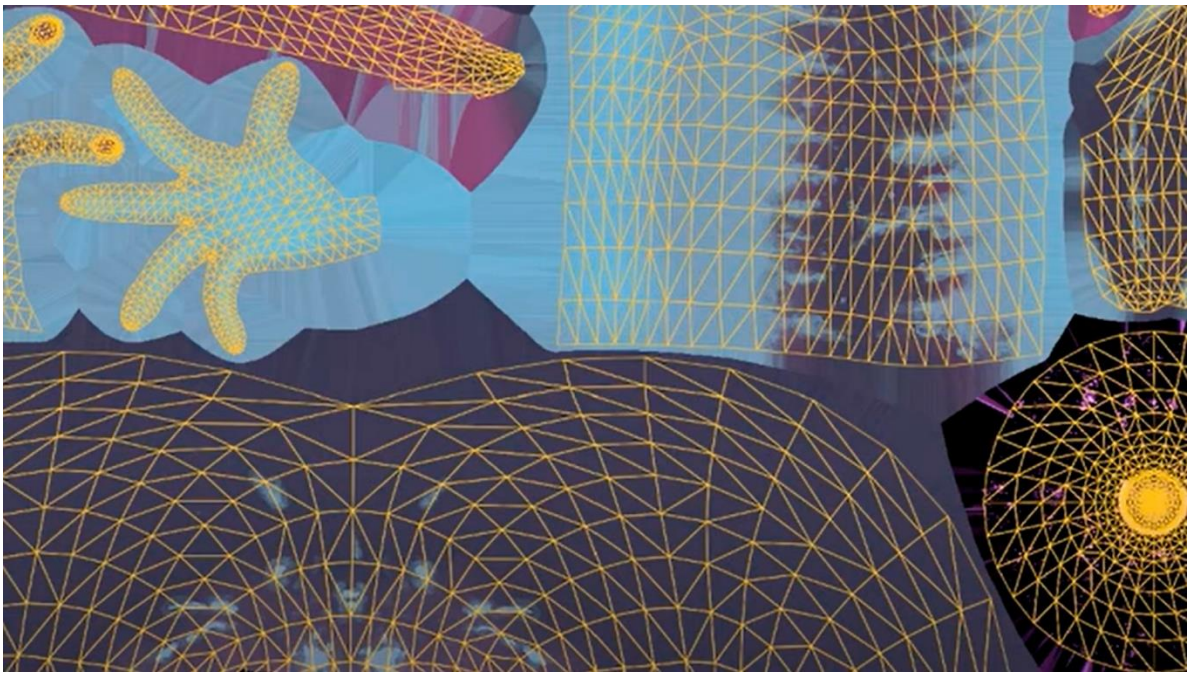


Fig. 64. Detalle de Mapeado UV (líneas amarillas) sobre textura con *Shell* (espacios intermedios) - SPRITE UV.

Texturas “sin costuras”

Como hemos visto en el bloque del mapeado UV de esta tesis, existen situaciones en que la textura debe repetirse para rellenar los huecos que pueda haber en uno o varios polígonos de la malla, cuyas dimensiones excedan el espacio destinado al mapeado UV, donde se ubicará la textura.

Cuando al aplicar una textura, los bordes de la imagen tienen continuidad al repetirse horizontal y/o verticalmente, se dice que es una textura sin costuras o *seamless texture*, ya que no se nota el paso de una imagen a otra. Estas texturas, normalmente, han sufrido algún tipo de edición o se trata de imágenes diseñadas para generar ese efecto y cubrir grandes superficies de polígonos sin sobrecargar el equipo con textura de gran tamaño. A este efecto visual se le denomina, de forma coloquial, como una textura “tilea”, término derivado del inglés *to tile* (embaldosar), debido al aspecto visual resultante de la repetición de la textura.

3.3.- EXPERIMENTACIÓN

Una vez expuesto y estudiado el proceso para la obtención de imágenes en un entorno con renderizado en tiempo real, conociendo la documentación y las herramientas, pasamos a realizar un proceso experimental para profundizar de forma directa en los materiales PBR.

3.3.1.- HERRAMIENTAS

3.3.1.2.- *UNITY – UNITY TECHNOLOGIES*

El motor gráfico elegido para el desarrollo de esta práctica es *Unity*, un motor gráfico muy completo y potente con un motor de renderizado que posee todas las herramientas que necesitamos.

Este motor gráfico utiliza Cg para la codificación de los *shaders*. Actualmente, este programa permite el uso o la creación con diferentes *Render Pipelines*. El programa incluye tres sistemas básicos:

- *Built-in Render Pipeline*.
- *Universal Render Pipeline* (URP).
- *High Definition Render Pipeline* (HDRP).

Tanto el *Built-in* RP como el URP, utilizan de base el *shader standard* para la generación de materiales PBR, que utiliza mapas de texturas en imágenes diferentes. El HDRP utiliza de base un *shader HDRP/Lit*, cuyos mapas de texturas están definidos en los diferentes canales de una imagen, técnica denominada como *channel packing*.

Durante esta experimentación queremos estudiar cada mapa de textura del *shader*, sus diferentes configuraciones y parámetros, y el efecto visual final de cada una por separado durante la ejecución del entorno. Para ello, utilizaremos el *shader standard* en un entorno configurado con el *Built-in* RP.

3.3.1.2.- *MICROSOFT VISUAL STUDIO - MICROSOFT*

Este *software* es un editor de código. Aunque no son imprescindibles los conocimientos de programación para el desarrollo de entornos en tiempo real, para la creación del espacio de trabajo que se va a utilizar durante la experimentación se ha hecho necesario crear algunos *scripts* que simplifiquen el trabajo, creando relaciones entre la interfaz de nuestro visualizador y los parámetros de configuración de los materiales, la iluminación, la cámara y el entorno.

3.3.1.3.- *PHOTOSHOP – ADOBE SYSTEM INCORPORATED*

El editor de imágenes *Photoshop* cumplirá en el desarrollo de la experimentación dos funciones, pre y post; mediante esta herramienta se van a generar y componer los mapas de textura del experimento, y, en segundo término, se utilizará el *software,* para el análisis de los resultados obtenidos.

3.3.1.4.- *BLENDER* – FUNDACIÓN BLENDER

Este *software* de edición 3D nos sirve para la edición de mallas, trabajar los mapeados UV y la preparación de objetos 3D para nuestro proyecto de *Unity*.

3.3.2.- CONFIGURACIÓN DEL PROYECTO

Unity es un *software* muy complejo con múltiples herramientas separadas en diferentes ventanas y menús. Para poder realizar esta práctica se ha generado un proyecto con una interfaz de usuario que da acceso a las principales herramientas y parámetros con los que se va a trabajar (Fig. 65).

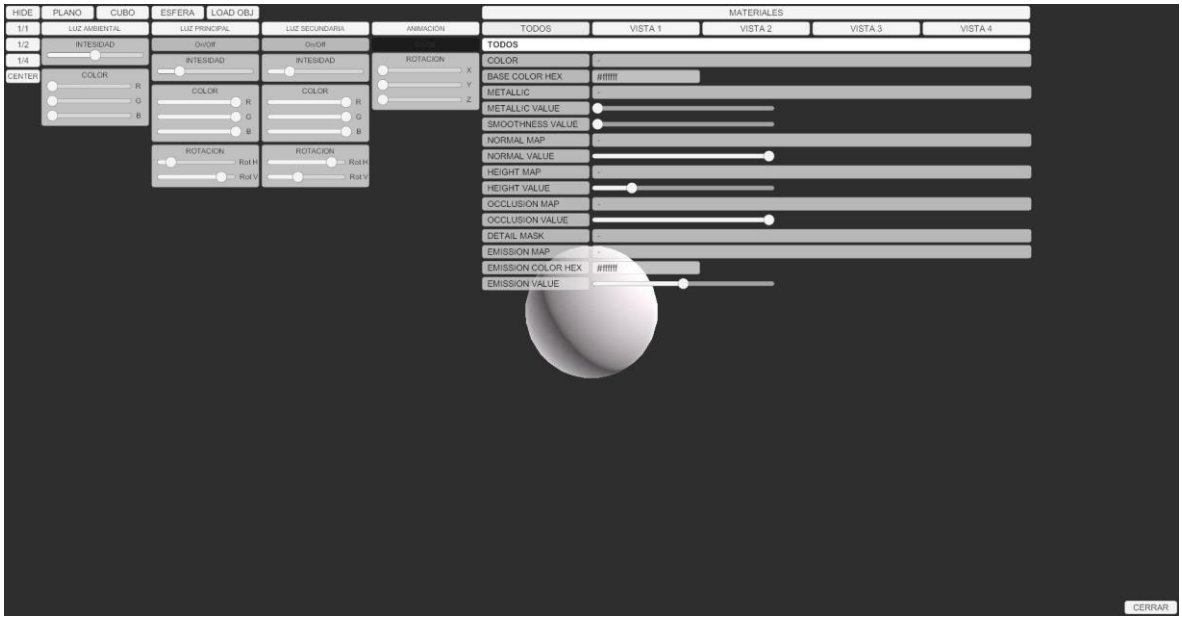


Fig. 65. Visual de la interfaz completa desplegada. Elaboración propia.

Múltiples objetos simultáneamente

Uno de los principales problemas a la hora de visualizar los cambios que se producen sobre una superficie virtual al modificar parámetros es no poder ver al mismo tiempo el mismo objeto antes y después de modificarlo bajo las mismas condiciones.

Para poder visualizar estos cambios al mismo tiempo sin tomar capturas de pantalla o exportar imágenes individuales de cada modificación, se ha desarrollado una interfaz que permite ver al mismo tiempo hasta 4 objetos (modelos del 1 al 4) con configuraciones diferentes y bajo los mismos parámetros de posición, rotación, escala, iluminación y punto de vista (Fig. 66).

Esto se ha conseguido creando cuatro grupos separados de cámara - objeto, configurando su visualización para que cada objeto sea sólo renderizado por su correspondiente cámara, y ubicando cada pareja cámara - objeto en exactamente la misma posición, siendo afectados por la misma iluminación en las mismas condiciones, y permitiéndonos trabajar con ellos de forma conjunta o separada.

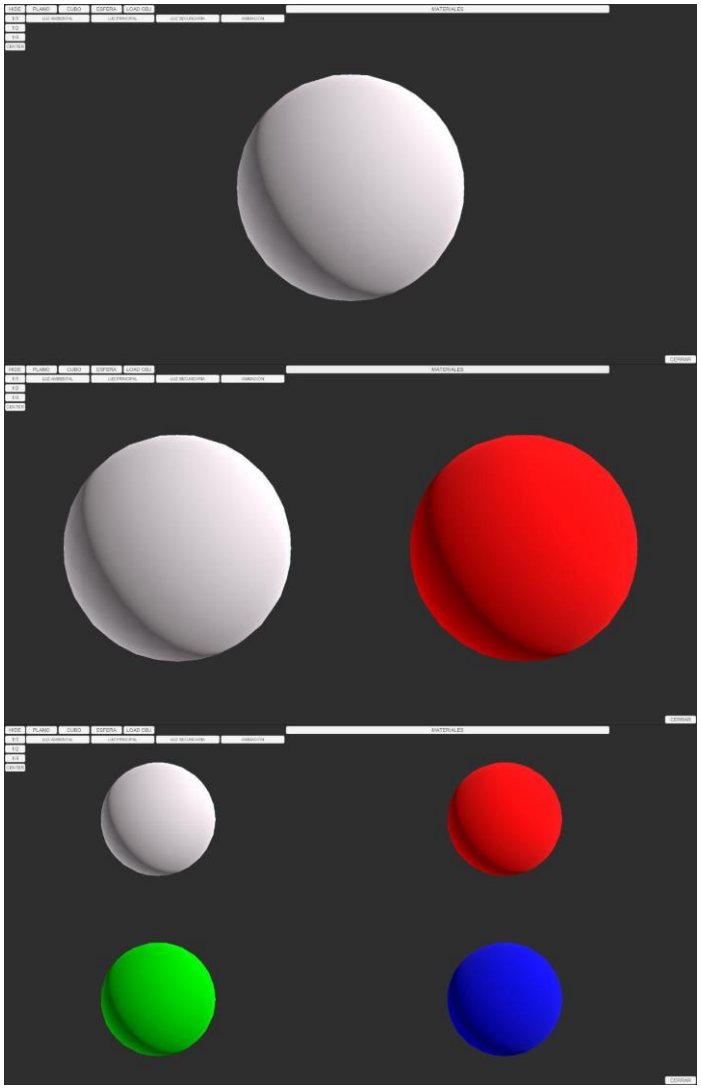


Fig. 66. Diferentes configuraciones de visualización mostrando 1, 2 o 4 objetos (de arriba a abajo: modelo 1, modelos 1 y 2, y modelos 1, 2, 3 y 4 respectivamente) con diferentes configuraciones en sus materiales bajo las mismas condiciones y al mismo tiempo. Elaboración propia.

Múltiples Mallas

Los diferentes mapas de texturas que se van a utilizar producen diferentes efectos y puede ser necesario cambiar el objeto sobre el que se van a visualizar para percibir mejor los cambios.

A través de la interfaz del programa podemos cambiar la malla sobre la que se aplica el material entre un plano, un cubo o una esfera, mallas proporcionadas por el programa *Unity* por defecto, y con su correspondiente mapeado UV, o podemos cargar mallas en formato OBJ si es necesario (Fig. 67).

Fig. 67. Sección de la interfaz con las opciones para la selección o carga de diferentes mallas. Elaboración propia.

Controles de iluminación

La interfaz permite controlar tres luces diferentes en el entorno: La iluminación ambiental y dos luces direccionales, ya que, como se ha visto, no sufren atenuación por la distancia al objeto, proporcionando una iluminación homogénea por toda la escena. La primera de ellas está configurada con sombras duras o de perfil definido, y la otra con sombras difusas o de contorno menos definido.

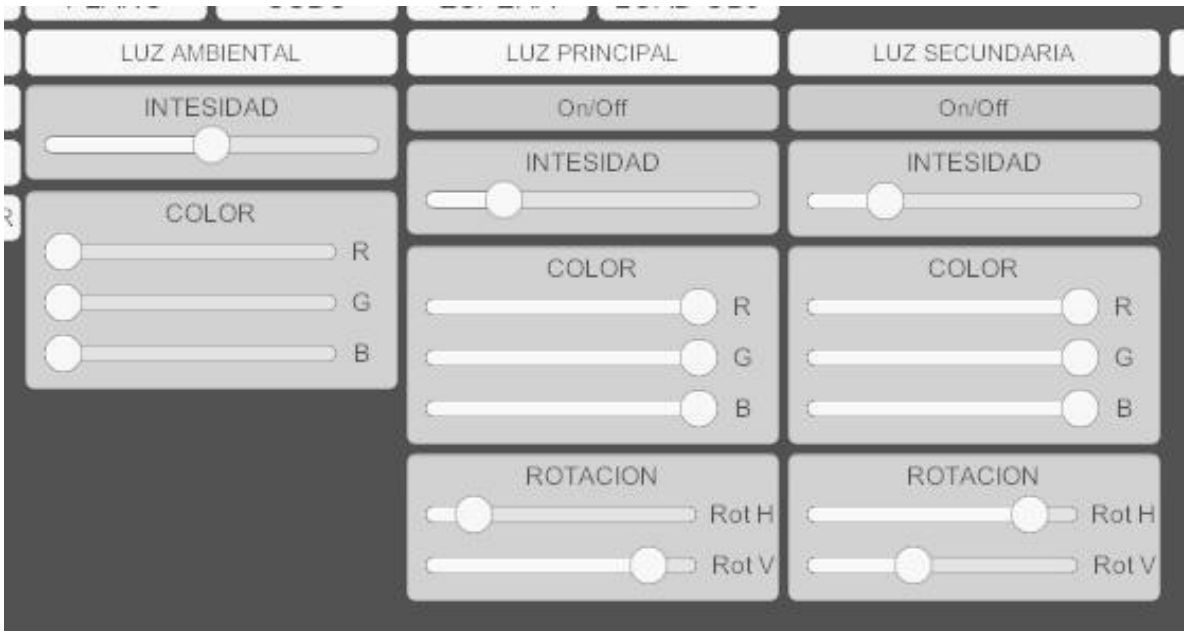


Fig. 68. Sección de la interfaz para el control de la iluminación. Elaboración propia.

A través de la interfaz, podemos activar y desactivar las luces, modificar su orientación, color o intensidad (Fig. 68).

Controles de movimiento

La interfaz incluye controles para generar sencillas animaciones en los objetos en base a su rotación en los diferentes ejes (Fig. 69), y, a través del ratón, podemos navegar, desplazar y rotar la cámara en el espacio tridimensional, además de hacer zoom sobre los modelos de la escena.

Fig. 69. Sección de la interfaz que permite controlar la animación de los objetos. Elaboración propia.

Configurador de materiales

A través de la interfaz podemos controlar los diferentes parámetros que configuran los materiales PBR de los diferentes modelos ya mencionados, ya sea en bloque, para aplicar una misma configuración a todos al mismo tiempo, o bien por separado, obteniendo y pudiendo visualizar al mismo tiempo las diferencias entre los diferentes parámetros (Fig. 70).

Esta sección de la interfaz permite aplicar texturas a los distintos mapas de textura del *shader standard* aplicado sobre los modelos de la escena, cargando los archivos de imagen desde una memoria externa a la aplicación. Esto permite generar nuevas texturas durante la experimentación, si es necesario, sin tener que guardarlas en el proyecto y volver a compilar toda la aplicación para su visualización.

Se han generado los *sliders,* necesarios para manipular los parámetros con punto flotante de la configuración de los Materiales, y cuadros para la entrada de texto (inputs), para poder definir colores en codificación hexadecimal (#xxxxxx).

Fig. 70. Sección de la interfaz para el control de la configuración de los materiales. Elaboración propia.

Finalmente, el visualizador se ha configurado con un fondo constante en gris oscuro (R: 50, G: 50, B: 50 / #323232), para que los objetos que estarán siempre iluminados se diferencien del fondo, delimitando claramente su contorno, pero evitando el negro de forma que al mismo tiempo podamos diferenciar el objeto del fondo si se produjeran efectos en la superficie que generen pixeles negros.

Un paso previo que se ha realizado a esta configuración del visualizador ha sido añadir a la escena un objeto de iluminación, *reflection probe*[37], en el que se ha dejado precalculado (*baked*) el *skybox* por defecto, de *Unity,* para poder visualizar los efectos de reflexión en las superficies de los modelos.

Una vez finalizada la configuración del proyecto, exportamos la aplicación y sólo necesitamos ejecutarla para empezar a trabajar.

[37] Documentación de *Unity* sobre *Reflection Probes* - https://docs.Unity3d.com//class-ReflectionProbe.html

3.3.3.- COMPONENTES DE UN MATERIAL PBR

Un *shader* puede estar compuesto de múltiples mapas de texturas y parámetros, y existen diferentes tipos de *shaders* que se incluyen dentro de la tipología de materiales PBR. Como ya hemos visto, dependen de su codificación y de la configuración del entorno, ya que cada *shader* tiene una finalidad distinta atendiendo a la estética final del proyecto.

Un material es considerado como PBR cuando incluye los siguientes mapas de texturas:

- **Color:** Determina el color base de la superficie del objeto y las zonas con transparencia (opacidad del material).
- **Metallico (*metallic*)**: Determina si la superficie responderá a la luz como un objeto metálico o no.
- **Aspereza (*roughness*)**: Similar al *smoothness* o suavizado, determina el nivel de reflexión y dispersión de la luz sobre la superficie.
- **Normal**: Modifica la forma en que la luz interactúa con la superficie del objeto simulando la modificación de la normal de los polígonos de la malla.
- ***Altura (height)*:** Varia el aspecto de la superficie de la malla modificando la posición de los píxeles de la superficie.
- **Oclusión ambiental (*Ambient occlusion*):** Al igual que el mapa de Normal, proporciona detalles sobre la malla aportando información de luces y sombras.
- **Emisivo (*Emissive*):** Convierte al objeto en un emisor de luz que afecta a la escena.

Un *shader* puede tener codificados otros parámetros, pero para esta práctica sólo hemos trabajado y estudiado aquellos que definen un material PBR.

Tabla 4. Relación de los componentes que configuran un material PBR con el *shader Standard* de *Unity* y los parámetros de configuración. Elaboración propia.

MAPA DE TEXTURA	*STANDARD SHADER* DE *Unity*	PARÁMETROS
COLOR	ALBEDO	Color (RGBA) Textura (RGBA)
OPACIDAD (*OPACITY*)	*RENDERING MODE*	Opaco / Recorte / Desvanecimiento / Transparente Textura (RGBA) = ALBEDO
METALLICO (*METALLIC*)	*METALLIC*	Valor (0/1) Textura (RGBA)
ASPEREZA (*ROUGHNESS*)	*SMOOTHNESS*	Valor (0/1) Textura (RGBA) = METALLIC
NORMAL	*NORMAL*	Valor (0/1) Textura (RGBA)
VOLUMEN (*HEIGHT*)	*HEIGHT*	Valor (0.005/0.08) Textura (RGBA)
OCLUSIÓN AMBIENTAL (*AMBIENT OCCLUSION*)	*OCCLUSION*	Valor (0/1) Textura (RGBA)
EMISIVO (*EMISSIVE*)	*EMISSION*	Color (HDR) Textura (RGBA) GI

3.3.4.- TEXTURA DE MUESTREO

Según la documentación, las texturas a utilizar en los diferentes parámetros del *shader* de un material han de tener unas características concretas según su finalidad, por ejemplo, usar tonos grises con sus correspondientes variaciones de saturación y luminosidad. A pesar de esta supuesta limitación, en el momento que un material nos permite insertar una textura, nos está permitiendo insertar cuatro parámetros propios de las imágenes digitales codificados como RGBA o el *linear space color* (espacio de color lineal), RGB (*Red*, *Geen*, *Blue*), más un canal A (*Alfa*).

Para poder estudiar el comportamiento de los diferentes mapas y su efecto sobre la superficie del modelo tridimensional, hemos generado una textura de muestreo (TM01) que nos servirá para comprobar las diferentes modificaciones de los cuatro valores (RGBA), que, a su vez, nos permite introducir una imagen en los diferentes mapas definidos en el *shader* (Fig. 71).

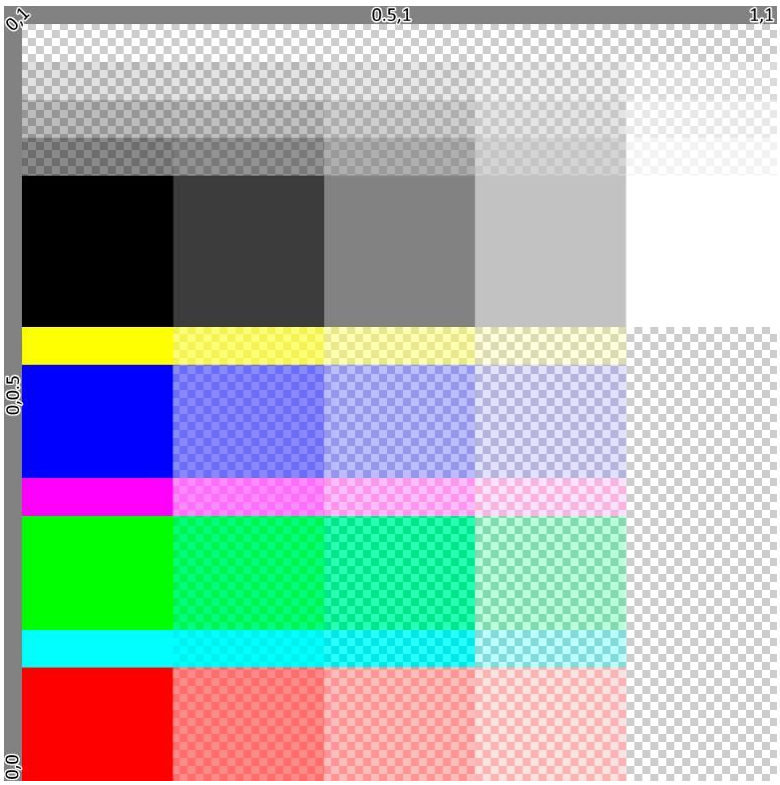


Fig. 71. Captura de TM01 en el entorno de *Photoshop* para diferenciar las secciones por colores (RGB) y con opacidad (A). Elaboración propia.

TM01 es una textura de 1024 píxeles de resolución (1024 X 1024 píxeles), que incluye diferentes secciones con diferentes configuraciones de RGBA, como podemos ver en la imagen.

Tanto en su borde izquierdo como superior, tenemos una “regla” con valores de 0 a 1, valores coincidentes con las dimensiones del mapeado UV.

MT01 está compuesta de una cuadrícula de 5 (filas) por 5 (columnas), divididas en 4 grupos, de arriba a abajo:

Escala de grises: Ocupando las dos primeras filas tenemos secciones con valores R=G=B. En la primera fila tenemos una progresión ascendente de valores de A desde 0 hasta llegar a la segunda fila con alfa 255 (opacidad total).

Azul B (blue): Ocupando la tercera fila tenemos una división horizontal. En la parte de arriba, el opuesto de azul R = G = 255, B = 0 y una progresión en horizontal de valores de A con valores de 255 a 0. Debajo tenemos el azul R = G = 0, B= 255 y su progresión horizontal de A de 255 a 0.

Verde G (Green): Ocupando la cuarta fila tenemos una división horizontal. En la parte de arriba, el opuesto de verde R = B = 255, G = 0 y una progresión en horizontal de valores de A de 255 a 0. Debajo tenemos el azul R = B = 0, G= 255 y su correspondiente progresión horizontal de A.

Rojo R (Red): Ocupando la quinta fila tenemos una división horizontal. En la parte de arriba, el opuesto de Rojo G = B = 255, R = 0 y una progresión en horizontal de valores de A de 255 a 0. Debajo tenemos el azul G = B = 0, R= 255 y su progresión horizontal de A.

Las secciones facilitan el visualizado y la delimitación de los efectos de la textura sobre la superficie de los modelos. Trabajar con degradados dificultará localizar el valor exacto que produce un resultado concreto.

Tabla 5. Desglose por secciones en sus valores RGBA de la textura TM01. Elaboración propia.

Referencia UV (Mapeado UV)					
	0,0,0,0	64,64,64,0	128,128,128,0	192,192,192,0	192,192,0
	0,0,0,64	64,64,64,64	128,128,128,64	192,192,192,64	192,192,64
	0,0,0,128	64,64,64,128	128,128,128,128	192,192,192,128	192,192,128
	0,0,0,192	64,64,64,192	128,128,128,192	192,192,192,192	192,192,192
	0,0,0,255	64,64,64,255	128,128,128,255	192,192,192,255	255,255,255,255
	255,255,0,255	255,255,0,192	255,255,0,128	255,255,0,64	255,255,0,0
	0,0,255,255	0,0,255,192	0,0,255,128	0,0,255,64	0,0,255,0
	255,0,255,255	255,0,255,192	255,0,255,128	255,0,255,64	255,0,255,0
	0,255,0,255	0,255,0,192	0,255,0,128	0,255,0,64	0,255,0,0
	0,255,255,255	0,255,255,192	0,255,255,128	0,255,255,64	0,255,255,0
	255,0,0,255	255,0,0,192	255,0,0,128	255,0,0,64	255,0,0,0

La textura MT01 integra un gran abanico de valores para los parámetros RGBA, como podemos ver en el siguiente esquema de distribución de valores (Tabla 5).

La documentación sobre el uso del *shader standard* de *Unity* indica que algunos mapas, aunque aceptan texturas RGBA, deben tener valores en escala de grises. No es lo mismo, a nivel visual, utilizar un píxel con valores distintos para cada parámetro de R, G, B, que usar un píxel que utiliza el mismo valor en los tres parámetros.

Debido a esta condición, realizamos una copia de la textura TM01 modificada, para que los diferentes valores de RGB se reajusten a escala de grises (TM02) (Fig. 72), y poder comprobar si el efecto es el mismo usando imágenes con píxeles con parámetros de R, G, B diferentes (colores), o iguales (escala de grises).

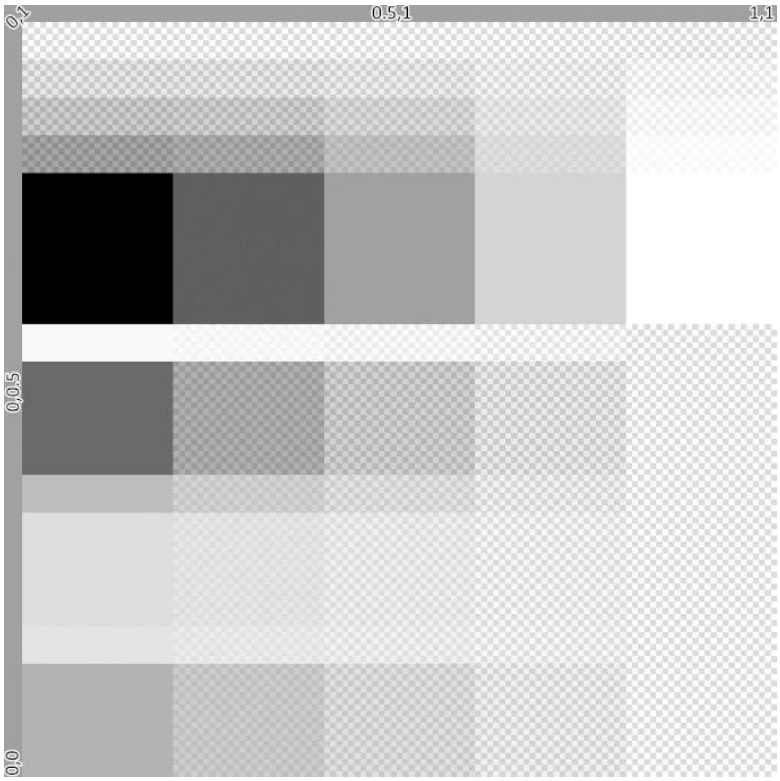


Fig. 72. Captura de TM02 en el entorno de *Photoshop* para diferenciar las secciones con opacidad (A). Elaboración propia.

Para TM02, los valores de A se mantienen igual que en TM01, pero podemos ver cómo los valores de RGB son totalmente variables, y nos ofrece un abanico más limitado de valores dentro de dichos parámetros.

3.3.5.- EJECUCIÓN DE LA APLICACIÓN

3.3.5.1.- COLOR O ALBEDO

Comenzamos aplicando nuestra textura MT01 en el mapa correspondiente al Color - Albedo, sin aplicar ningún tipo de transparencia (*opaque*) en el modelo 1, con desvanecimiento (*fade*) en el modelo 2, con configuración de recorte (*cutout*) al modelo 3, y transparencia (*transparency*) al modelo 4 (Fig. 73).

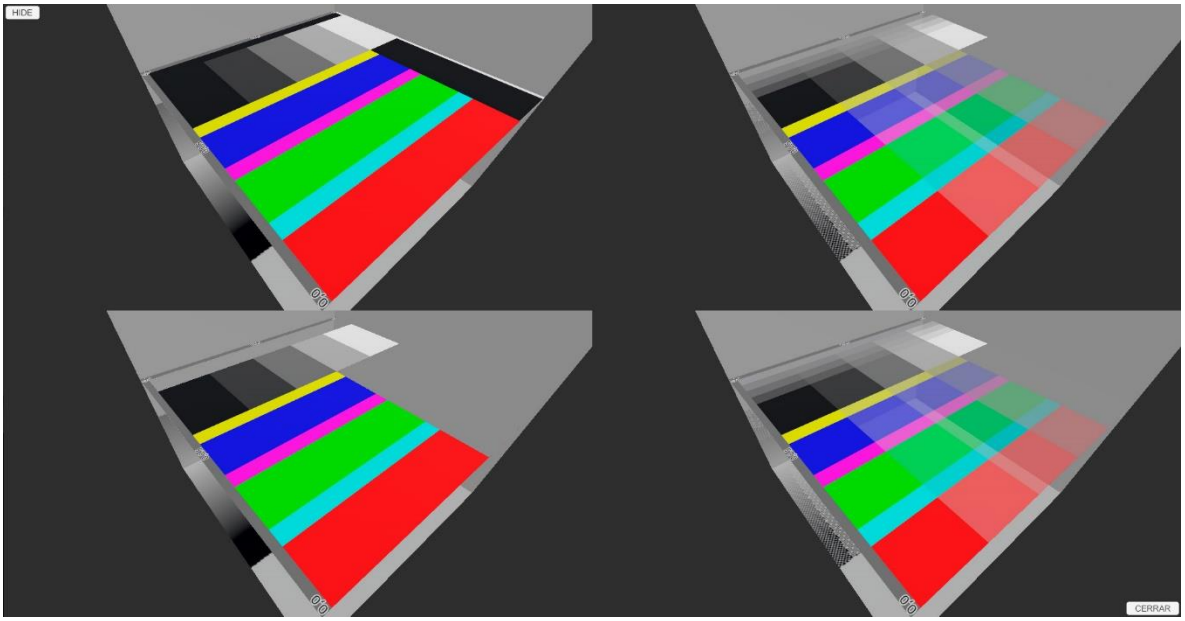


Fig. 73. Diferentes estilos de transparencia permitidos por el motor gráfico de *Unity* (*blend modes*): De arriba abajo y de izquierda a derecha - Opaco, recorte, desvanecimiento y transparente. Elaboración propia.

Si planteamos un valor de opacidad entre 0 y 1 (Tabla 6), donde 0 es transparente y 1 es opaco, vemos que en el modelo 1 (opaco - *opaque*) se pierden todos los valores de transparencia. Mantiene el color constante sobre la superficie excepto en el caso de que la opacidad sea 0, que lo convierte los valores R = G = B = 0 (negro).

En el modelo 2 (desvanecimiento - *fade*), genera una transparencia de acuerdo con los valores de opacidad de la textura.

En el modelo 3 (recorte - *cutout*), observamos cómo, según el valor de opacidad, las secciones con valores de alfa son totalmente transparentes u opacas, sin valores intermedios. Así, las secciones de TM01 con valores opacidad de 0, 0.25, 0.5 y 0.75, se vuelven transparentes respectivamente con valores (*cutoff*) de 0.1, 0.26, 0.51 y 0.76. Con un valor (*cutoff*) de 0 volvemos a un estado de opacidad con pérdida de los parámetros RGB, como en el modelo 1 (Fig. 74).

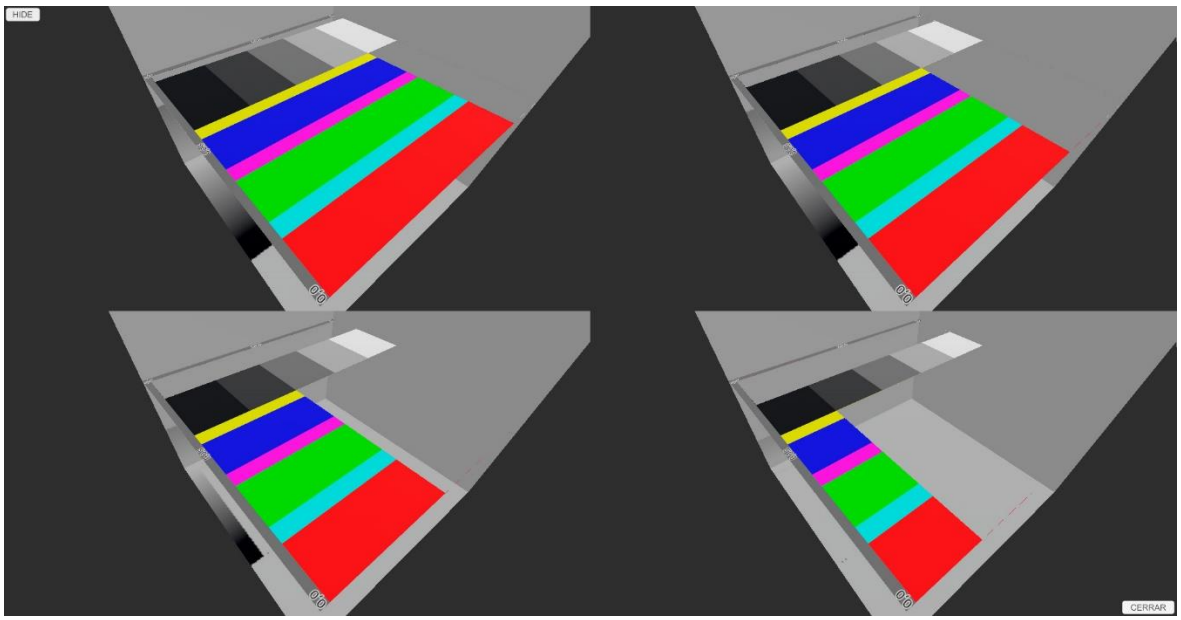


Fig. 74. Transparencia de recorte (*cutout*) según valor de opacidad (*cutoff*) de izquierda a derecha y de arriba a abajo -0.1, 0.26, 0.51 y 0.76. sobre valores de opacidad definidos en *photoshop* de 0, 0.25, 0.5 y 0.75. Elaboración propia.

En el modelo 4 (transparente - *transparent*), vemos a simple vista poca diferencia con el Modelo 2 (desvanecimiento - *fade*). La diferencia entre ambos es que el *fade* hace desaparecer la superficie por completo, eliminando su interacción con el entorno. El transparente hace que podamos ver a través de la superficie, pero mantiene su interacción con la luz. Al aplicar una textura gris neutro (R = 128, G = 128, B = 128) al mapa de *metallic*, aumentado la reflexión de la superficie (Fig. 75).

Tabla 6. Relación entre el valor de opacidad y el parámetro Alfa (A) de la textura. Elaboración propia.

OPACIDAD	VALOR DE A (RGBA)
0	0
0.25	64
0.50	128
0.75	192
1	255

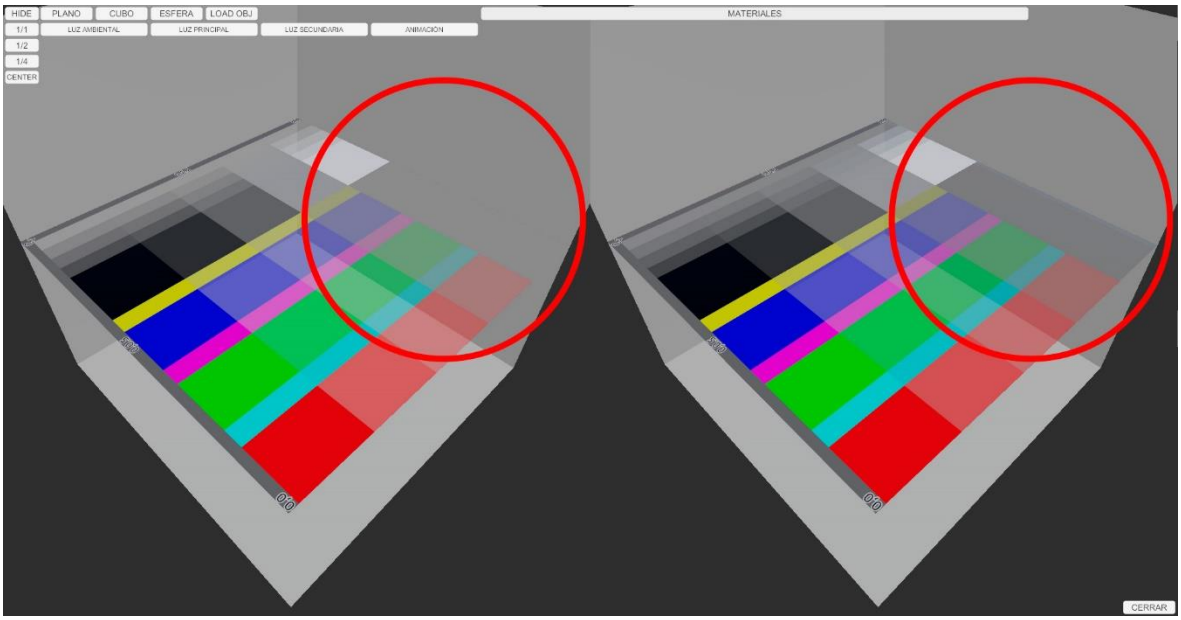


Fig. 75. Diferencia entre modelos de transparencia desvanecimiento y transparente. Elaboración propia.

Estéticamente, o a nivel de diseño, la transparencia desvanecimiento (*fade*) sería la de un holograma: podemos ver a través de él, pero no tiene reflejos ni interactúa con la luz, no es sólido, y la transparencia transparente (*transparent*) sería similar a la de un cristal, que nos permite ver a través de él pero tiene interacción con su entorno.

De ahora en adelante, los materiales que utilizaremos estarán configurados como transparentes - *Transparent* por defecto, para poder ver, a pesar de las secciones con variaciones de opacidad, cómo el entorno interactúa sus superficies.

Modificador de color

El albedo en *Unity* nos permite utilizar un color RGBA como modificador de la textura. Esto quiere decir que la textura se verá afectada por el color. Al aplicar diferentes colores en dicho parámetro a los diferentes modelos con nuestra textura TM01 aplicada, vemos que el efecto visual que produce es el de multiplicar el valor de RGBA de cada píxel, por los valores que le pasamos a través del parámetro de color (Fig. 76).

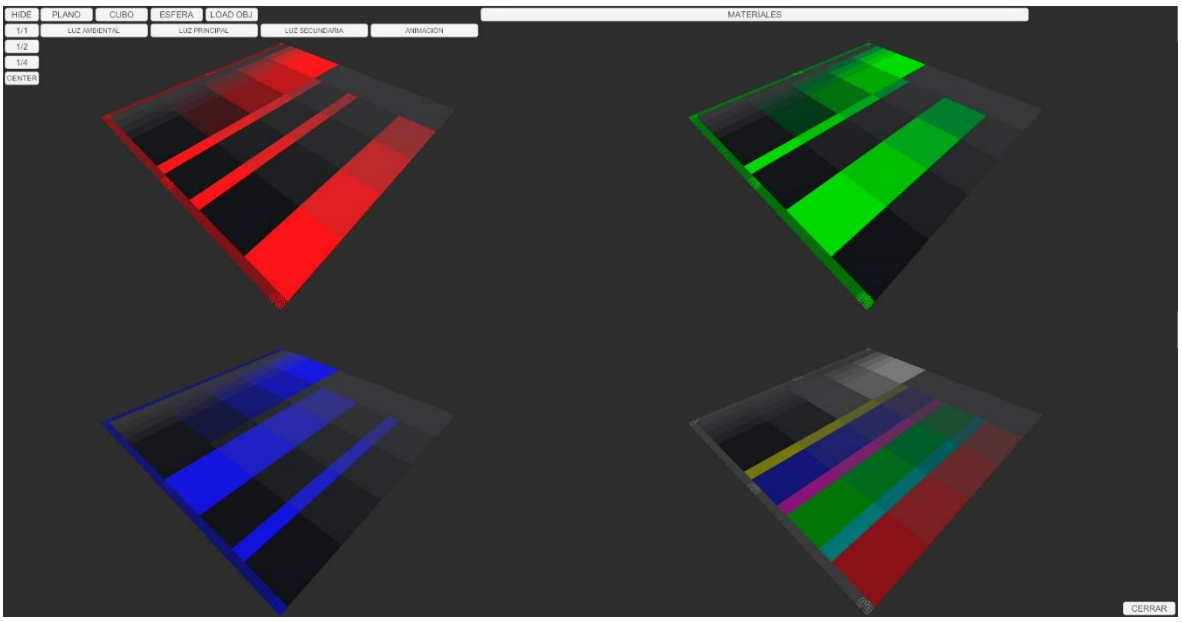


Fig. 76. De izquierda a derecha y de arriba a abajo - Textura TM01 modificado por color Rojo (RGBA: 255, 0, 0, 255), Verde (RGBA = 0, 255, 0, 255), Azul (RGBA = 0, 0, 255, 255) y gris neutro (RGBA = 128, 128, 128, 255) respectivamente. Elaboración propia.

Como podemos ver en el modelo 1, al que se le ha aplicado un color Rojo (R = 255, G = 0, B = 0, A = 1), todos los píxeles de la textura TM01 que tenían algún píxel con valores en "R" se ven coloreados, mientras los píxeles de verde G y azul B se quedan en negro al multiplicarse por 0, el valor introducido por el modificador de color. Lo mismo ocurre con cualquier otra combinación de valores en RGB.

El parámetro Alfa (A) del RGBA, se aplica por igual a toda la superficie del objeto, modificando el valor de A de todos los píxeles de la textura TM01 por igual (Fig. 77).

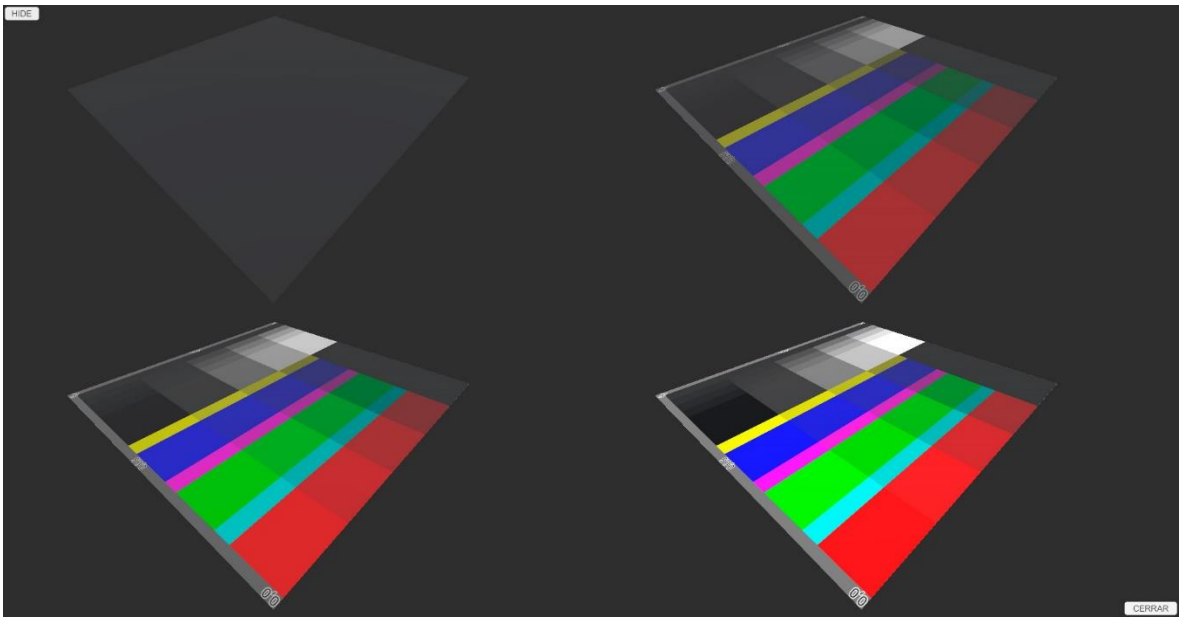


Fig. 77. De izquierda a derecha y de arriba a abajo - Textura TM01 modificada por el valor de opacidad (Alfa - A) 0, 0.25, 0.5, 0.75 respectivamente. Elaboración propia.

Luz

Cuando aplicamos luz sobre el modelo con la textura TM01 aplicada al albedo, ésta funciona de forma similar al modificador de color, aumentando o disminuyendo la intensidad de cada píxel de la superficie, dependiendo de si éste ya tiene un valor para los parámetros RGB. La luz blanca (R = G = B = 255) multiplica por igual todos los valores de RGB de la textura.

Cuando modificamos la luz y configuramos una luz Roja (R = 255, G = B = 0), genera un resultado similar al multiplicador de color, con una diferencia: tiñe ligeramente de rojo superficies de píxeles que tienen valor 0 en parámetro R del RGB (Fig. 78). Lo mismo ocurre con los parámetros de luz verde (G) y azul (B).

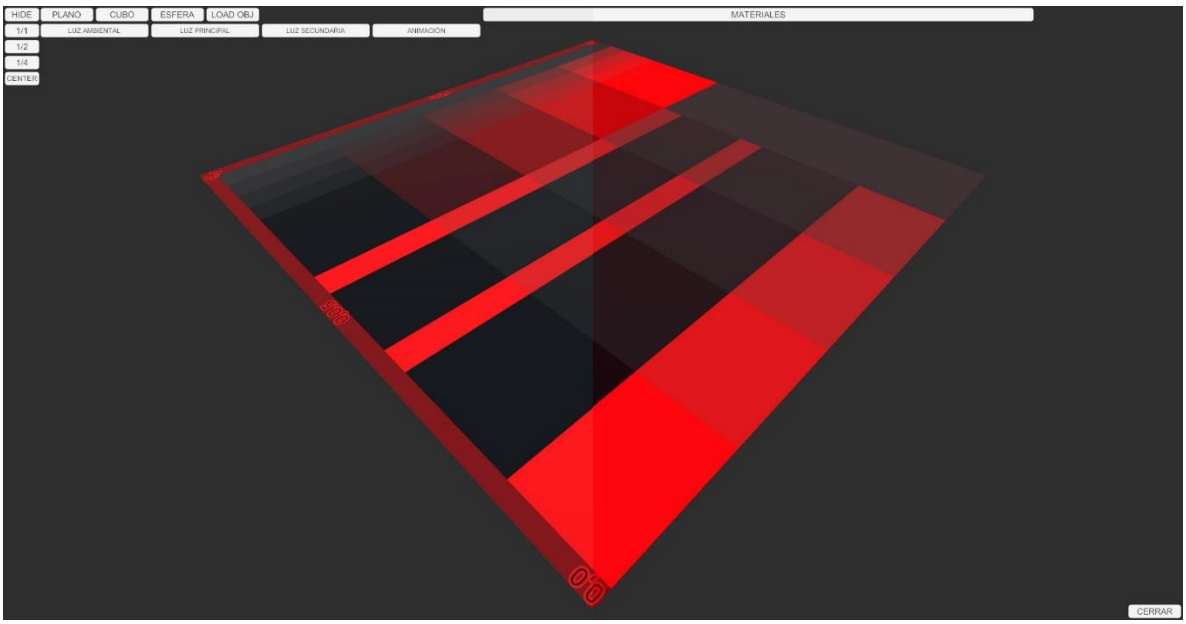


Fig. 78. Diferencia entre el modificador de color configurado en Rojo y la aplicación de una luz Roja. Elaboración propia.

3.3.5.2.- METALLIC / SMOOTHNESS

El parámetro *metallic* tiene un valor entre 0 y 1, donde 0 es nada metálico, y 1 es el metal perfecto, y este efecto se ve, además, afectado por el *smoothness* o suavizado, que también tiene un valor entre 0 y 1, donde 0 es una superficie totalmente mate, o la dispersión absoluta, y 1 es la reflexión perfecta.

Comenzamos aplicando nuestro material MT01 en el mapa de texturas correspondiente al *metallic* de nuestro material. Lo primero que se observa es que los píxeles que tienen valor de rojo en su RGBA reaccionan como píxeles metálicos, mientras que los que tienen otros valores en su RGBA, pero su valor de R es 0, generan una reacción igual a si el valor de *metallic* es 0 (Fig. 79).

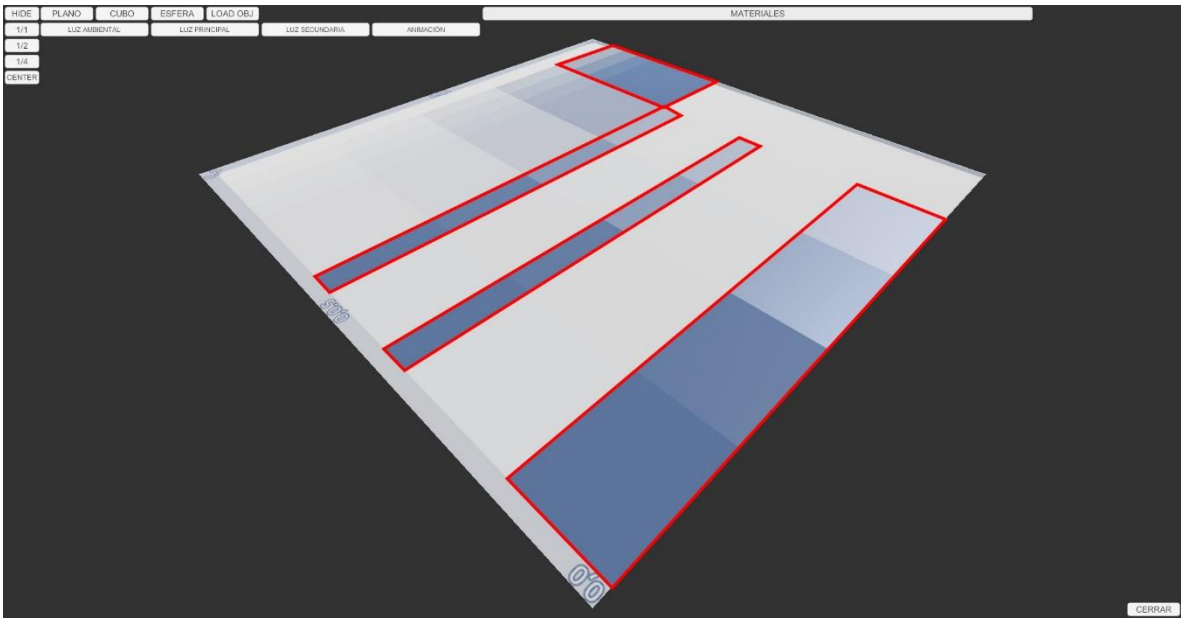


Fig. 79. Resaltado en rojo, los píxeles que tienen valores de R = 255 sin importar el resto de los valores de RGB. Elaboración propia.

Para ver con más detalle este efecto, generamos una textura con múltiples valores de RGBA (MT03) que aplicaremos a nuestro material (Fig. 80).

Observamos cómo la superficie cambia con los diferentes valores de R y A del RGBA de la textura (Fig. 81), sin importar el resto de los valores. El valor de R produce variaciones en el valor de *metallic* de la superficie. Cuanto menor es el valor de R en el píxel, menor es el valor de *metallic*. Un valor de R=255 será igual a un valor de *Metallic* = 1 (Metal perfecto) y un valor de R = 0 corresponderá con un *Metallic* = 0 (nada metálico).

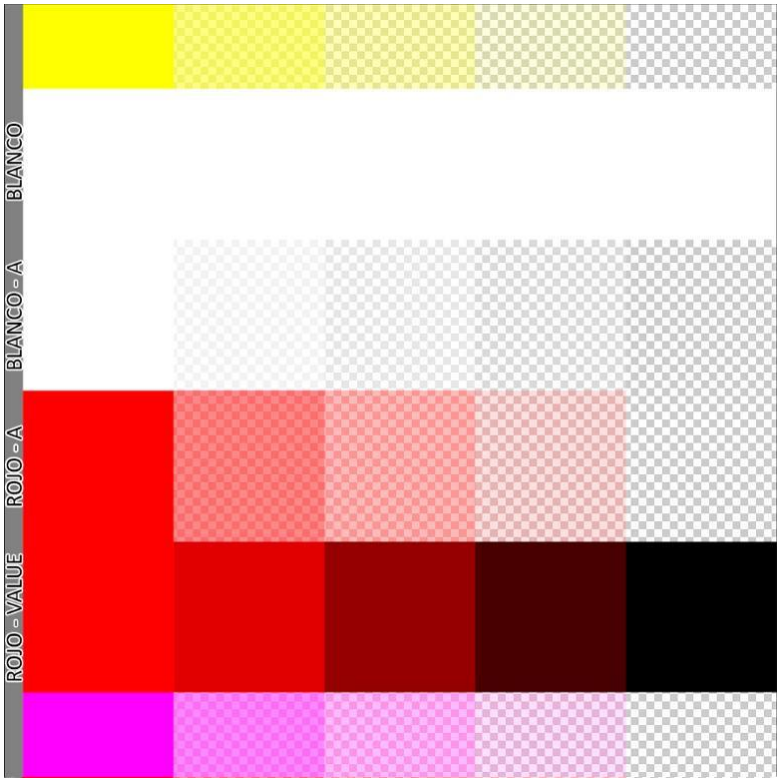


Fig. 80. Captura de TM03 en el entorno de Photoshop para diferenciar las secciones con opacidad (alfa). Elaboración propia.

Por otro lado, los valores de alfa producen modificaciones en el valor de *smoothness* de la superficie. Continuando con nuestro valor de opacidad descrito anteriormente, un valor de opacidad 0 (transparente) corresponderá con un valor de *smoothnes* de 0 (nada reflectante o dispersión máxima) y un valor de opacidad de 1 (opaco) corresponderá con un valor de *smoothness* de 1 (Reflexión perfecta o dispersión nula).

Para concluir la experimentación con el *metallic*, vemos que las texturas que se utilizan en este tipo de mapas suelen ser en escala de grises. así que aplicaremos nuestra textura TM02 al modelo para ver si aporta algún dato que no se haya tenido en cuenta (Fig. 82).

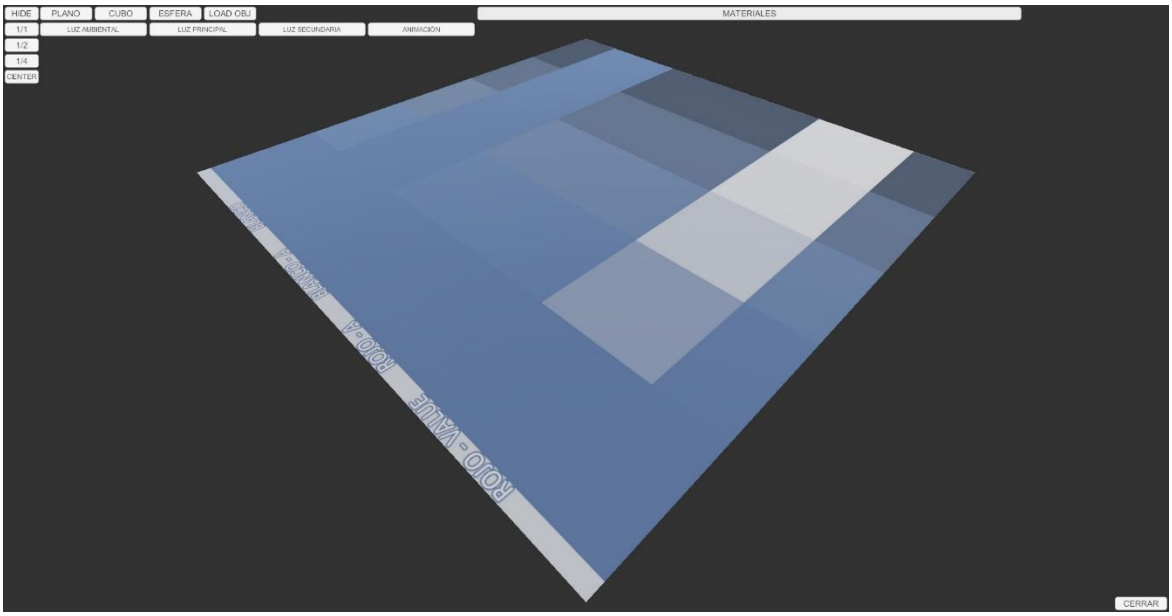


Fig. 81. Efectos producidos por la textura MT03 sobre la superficie. Elaboración propia.

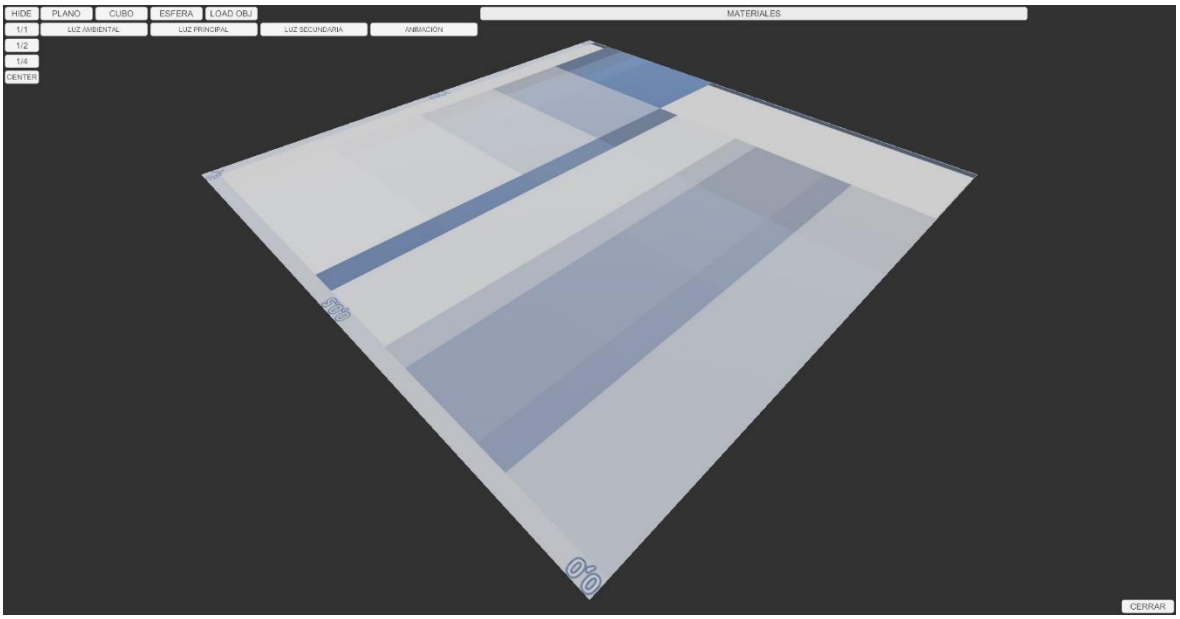


Fig. 82. Efectos producidos por la textura MT02 sobre la superficie. Elaboración propia.

El efecto es el mismo que en los casos anteriores, sólo el parámetro de R y A de cada píxel de la textura RGBA afecta al parámetro *metallic* y *smoothness* respectivamente.

Luz

Respecto a la interacción de la luz y la escena con la superficie, observamos que los valores de *metallic*, afectados por el valor de R de la textura, nos devuelven una superficie más o menos metálica, lo que produce que dicha superficie nos devuelva reflejos del entorno (de ahí los tonos azulados en la superficie), al reflejar el *skybox* precalculado en la *Reflection probe*, mientras que los valores de A (opacidad) que modifican la reflexión, modifican la refracción de la luz y del entorno, similar a volver la superficie más o menos porosa, focalizando o dispersando el reflejo de la luz y del entorno sobre la superficie (Fig. 83).

SMOOTHNESS - ROUGHNESS

Cuando hablamos de materiales PBR, y en concreto del mapa de metálico, salen a relucir dos términos: *smoothness* (suavidad) y *roughnees* (aspereza) y en ocasiones lleva a confusión. Ambos términos hacen referencia al mismo efecto visual, pero son, en sí mismos, opuestos. Un valor de *smoothness* de 1 corresponderá con un valor de 0 de *roughness* y viceversa. Eso es importante cuando utilizamos texturas diseñadas por otros autores, y éstas están declaradas como *roughnees*. O bien invertimos el canal de alfa, o invertimos el valor del parámetro para convertirlo en un mapa de *smoothnes*.

METALLIC - SPECULAR

En *Unity*, al trabajar con un *shader standard*, tenemos la posibilidad de elegir entre el perfíl "*standard*" u otro denominado como "*Standard* (*specular setup*)". La principal diferencia entre estos dos *shaders* está en que la configuración *standard* nos permite configurar un color diferente al de la superficie (Albedo) para la reflexión, mientras en el *metallic* podemos elegir que el color de la reflexión lo controle el mapa de *metallic* o el Albedo.

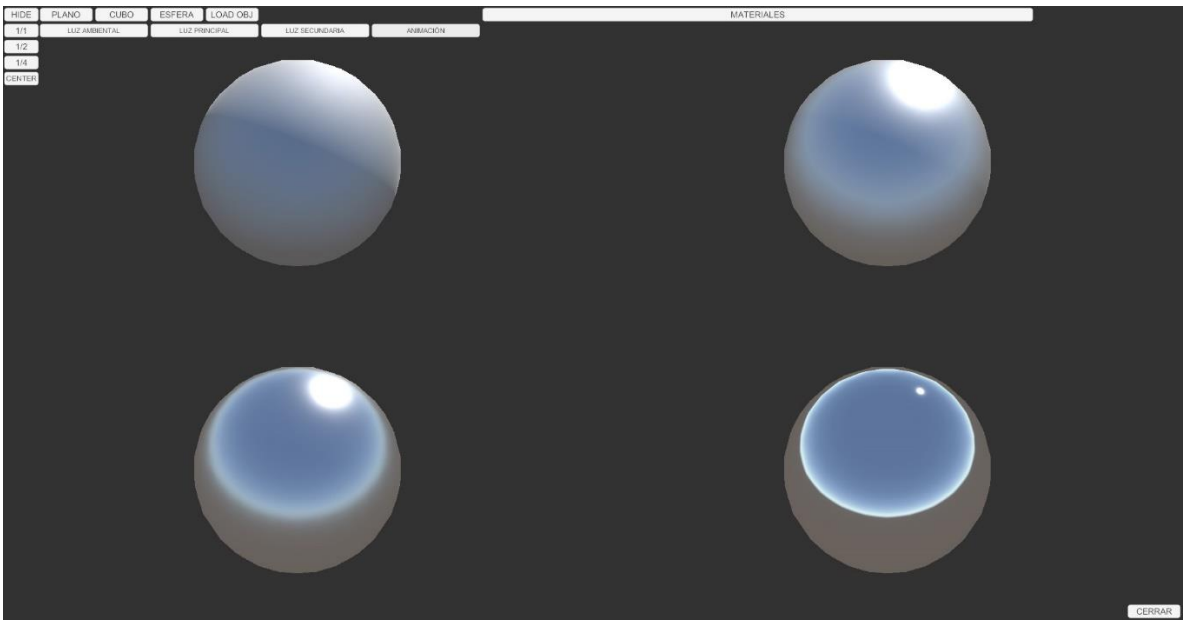


Fig. 83. De izquierda a derecha y de arriba a abajo, aumento del valor de *smoothnes* (0 a 1) sobre la superficie R = 255 donde se puede observar las variaciones en el reflejo de la luz y de la escena que le rodea. Elaboración propia.

3.3.5.3.- NORMAL

Al aplicar la textura MT01 a nuestro modelo (Fig. 84), vemos que se producen modificaciones en la tonalidad de la superficie, pero éstas son distintas en cada cara de nuestro cubo de muestra. Al aplicarle movimiento o cambiar la posición de las luces, vemos que los tonos cambian.

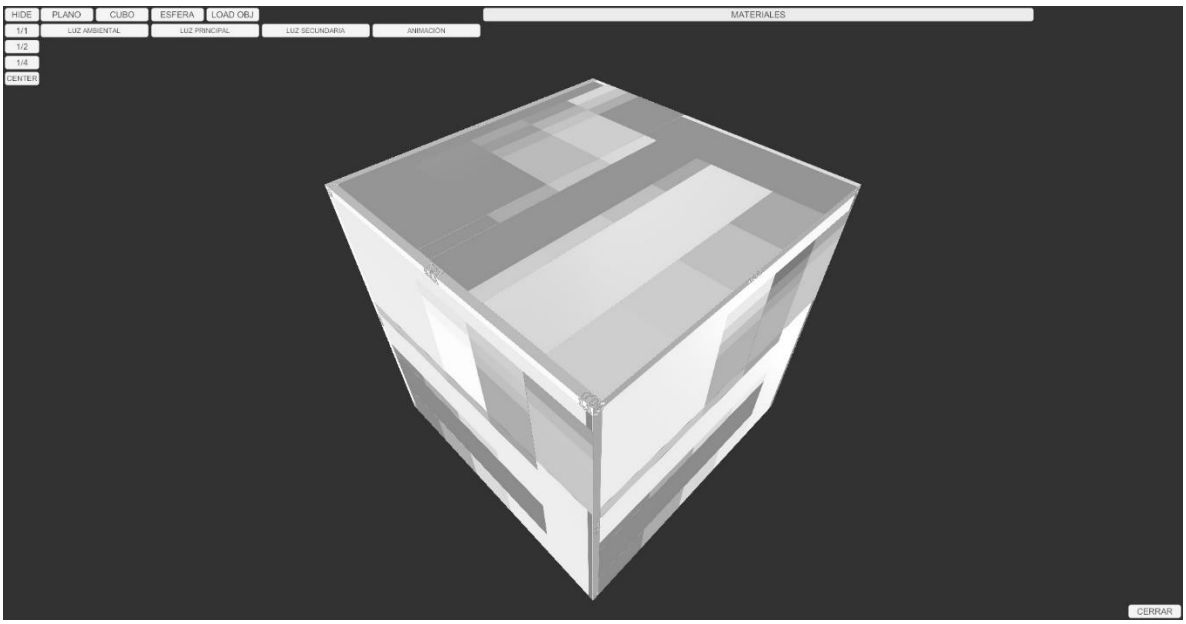


Fig. 84. Visual del cubo de muestra con la textura MT01 aplicada en su mapa de normal. Elaboración propia.

Este efecto se debe a que el mapa de normales modifica la interacción de la luz sobre la superficie, y la cámara, en lugar de captar toda la luz que incide sobre la superficie, muestra una modificación de esta producida por el mapa de normales. Gracias a este efecto, los mapas de normales pueden simular relieves, deformaciones de superficie u otros efectos.

Los mapas de normales utilizan todo el espectro de valores del RGBA. Al generar una imagen que servirá como mapa de normales, debemos tener en cuenta que:

- Los píxeles con valores de R producirán desviaciones en el eje X (derecha - izquierda de la superficie), siendo los píxeles con valores altos los que tendrán la dirección positiva (derecha)

- Los píxeles con valores de G producirán desviaciones en el eje Y (arriba - abajo de la superficie). Hay que tener en cuenta que los motores gráficos que utilizan HLSL o CG toman los píxeles de valores altos como dirección positiva hacia abajo, mientras que los motores gráficos que utilizan GLSL toman los píxeles con valores altos de G con dirección positiva hacia arriba. Como ya hemos indicado anteriormente, *Unity* utiliza CG.
- Los píxeles con valores de B producirán desviaciones en el eje Z (normal positiva - normal negativa de la superficie) siendo los píxeles con valores altos los que tendrán dirección positiva (normal positiva).

Por ejemplo, un píxel de la superficie que no realice ninguna modificación estará compuesto de los valores que se muestran a continuación. (Tabla 7).

Tabla 7. Color y valores de RGB para píxeles de la textura de mapa de normales que no realice modificación alguna sobre la superficie del objeto 3D. Elaboración propia.

PARÁMETRO	VALOR	COLOR
R (*Red*) Rojo	128 ½ de 255, no produce modificación ni a derecha ni a izquierda.	
G (*Green*) Verde	128 ½ de 255 no produce modificación ni a arriba ni a abajo.	
B (*Blue*) Azul	255 valor máximo, sigue la dirección de la superficie.	

Estos valores generan el color violeta que hace tan reconocibles los mapas de normales. A partir de este color base, cambiarán los valores de RGB según el ángulo y la dirección en la que se quiere modificar la reflexión de la luz sobre la superficie respecto de la normal del polígono sobre el que se aplica.

Estudiando en detalle los cambios en la superficie de nuestro modelo, vemos un efecto extraño. Los píxeles con valor de R distintos de 0, pero con diferentes valores de opacidad,

producen diferentes resultados ante la misma luz, mientras que los píxeles con valor G y B distintos de 0, pero R = 0, se mantienen constantes.

Generamos una nueva textura (MT04) (Fig. 85) muy similar a MT01, compuesta de bloques de colores rojo, verde, azul, negro, blanco y lila (R = 128, G = 128, B = 255) con diferentes valores de Alfa (0.25, 0.5, 0.75, 1).

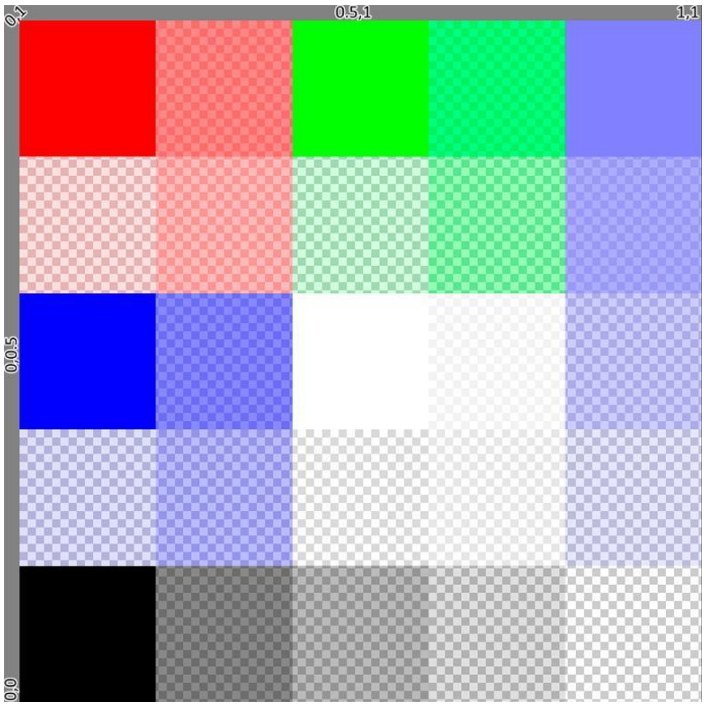


Fig. 85. Captura de la Textura MT04 tomada en el entorno de Photoshop para visualizar los valores de opacidad. Elaboración propia.

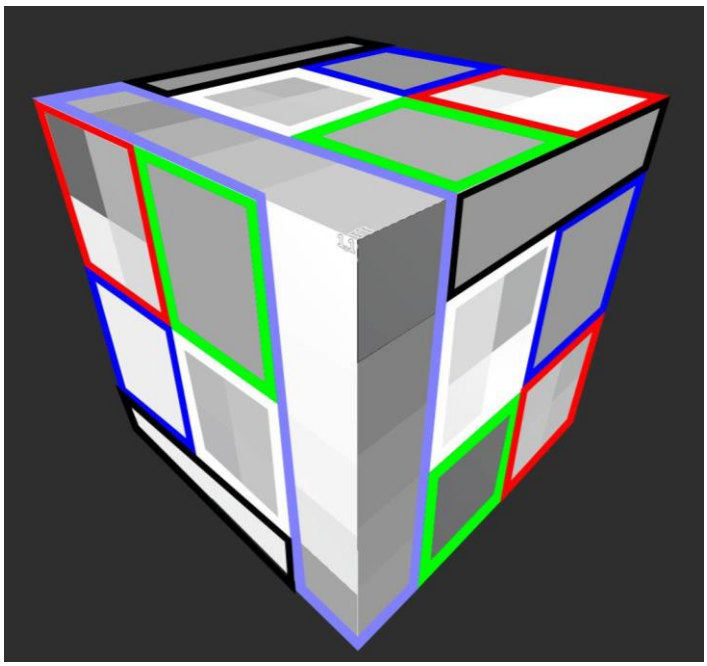


Fig. 86. Captura de la textura TM04 aplicada al cubo por defecto de nuestro entorno con los bloques de colores de la textura enmarcados en su color correspondiente. Elaboración propia.

Vemos cómo los bloques que tienen píxeles con valor R distinto de 0, el Rojo (R:255, G:0 B:0), el Blanco (R = 255, G = 255, B = 255) y el Lila (R = 128, G = 128, B = 255) muestran modificaciones en la superficie, no solo debidas al color, sino también a sus valores de opacidad A, mientras que el bloque Verde (R = 0, G = 255, B = 0), el Azul (R = 0, G = 0, B = 255), y el negro (B = 0, G = 0, B = 0) a pesar de tener valores de opacidad variados, como podemos ver en la textura TM04, no producen ningún tipo de modificación en la superficie, debido a los valores de opacidad A. Además, tanto los píxeles azules (R = 0, G = 0, B = 255), como los negros (R = 0, G = 0, B = 0) producen la misma modificación superficial mientras se encuentran en la misma superficie (misma normal) (Fig. 86).

3.3.5.4.- HEIGHT

Nuevamente, comenzamos aplicando nuestro material MT01 al mapa de texturas de relieve (*Height map*) de nuestro material configurado con el *shader standard*, y a primera vista, la superficie no sufre ningún tipo de modificación.

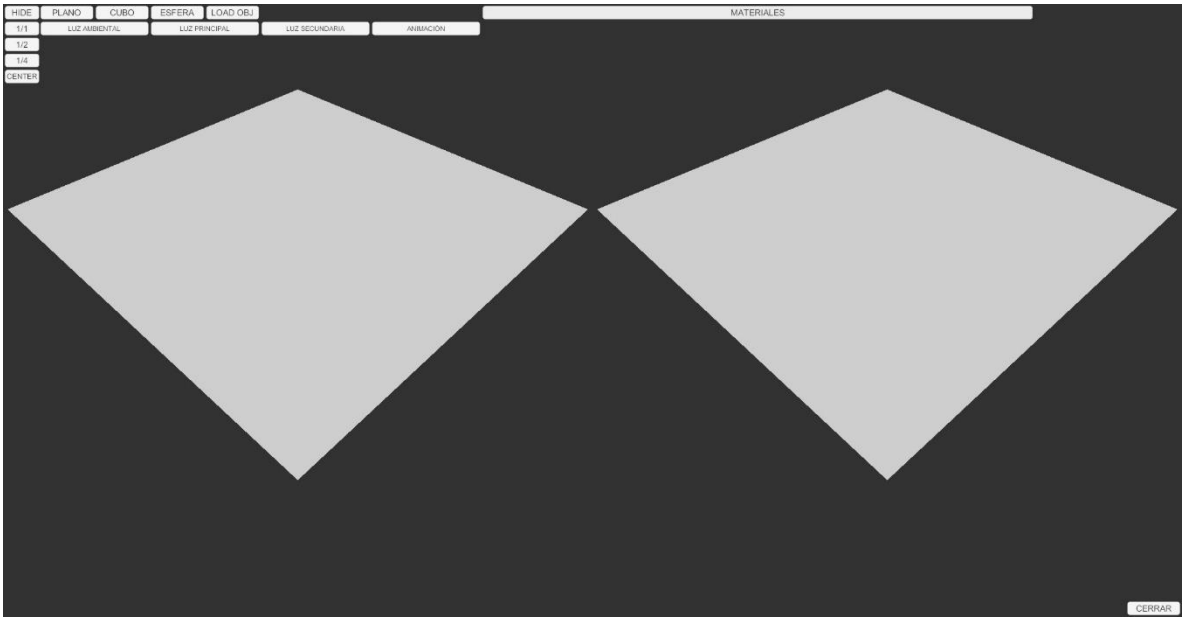


Fig. 87. Captura de la aplicación de MT01 en el mapa de relieve (izquierda), y sin ninguna textura aplicada en el Mapa de relieve (derecha). Elaboración propia.

El mapa de relieve, a diferencia del mapa de normales, genera una distorsión en la posición de los píxeles de acuerdo con los valores aportados por la de texturas respecto de su posición de origen, o posición por defecto, modificando así la imagen final y la percepción de la superficie. Si todos los píxeles tienen el mismo color y tono, resulta imposible visualmente detectar tal distorsión, por lo que aplicamos nuestro material MT01 también en el Mapa de texturas de color (albedo), para poder percibir el desplazamiento de los píxeles (Fig. 88).

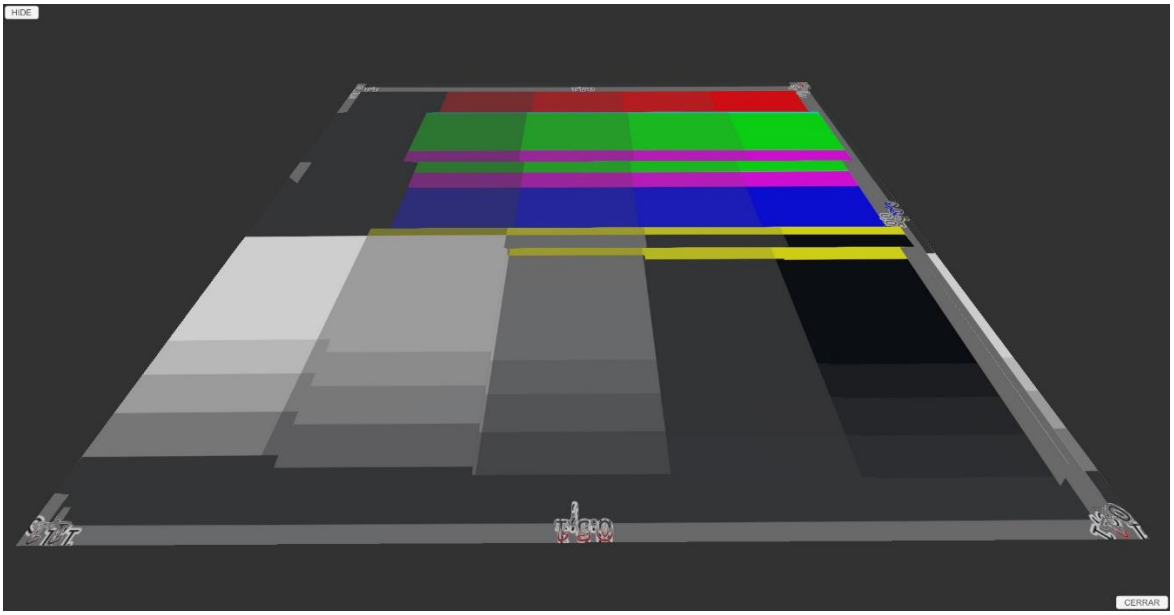


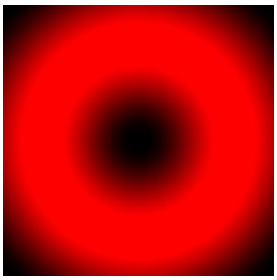
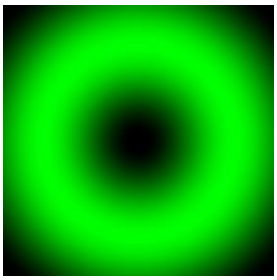
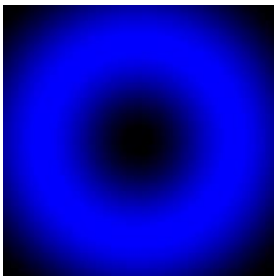
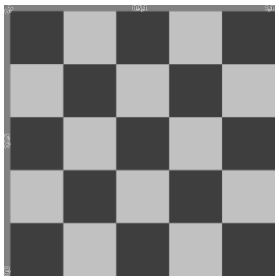
Fig. 88. Captura de la aplicación de MT01 en el mapa de relieve y en el mapa de Albedo. Elaboración propia.

Ahora podemos ver cómo los píxeles de la superficie están desplazados. Esto se debe al efecto de *parallax* generado por la textura, haciendo que los píxeles parezcan estar a diferentes alturas sobre la superficie del modelo.

Vemos cómo los píxeles que tienen tonos grises, verdes y amarillos se elevan sobre el resto de los píxeles sin ningún tipo de variación, debido a la opacidad. Todos estos píxeles tienen valores de G, por lo que el mapa de *Height* depende exclusivamente del parámetro de G de los píxeles de la textura.

Para comprobarlo, generamos 3 nuevas texturas, HR con valores de R de 0 a 255, HG con valores de G de 0 a 255, y HB con valores de B de 0 a 255, y para visualizar la deformación, generamos una textura sin valores de alfa (TM05) y procedemos a aplicarlas a los diferentes modelos (Fig. 89).

Tabla 8. Texturas de prueba para el mapa de *Height*. Elaboración propia.

			
HR (R:0-255, G:0, B:0, A:0)	HG (R:0, G:0-255, B:0, A:0)	HB (R:0, G:0, B:0-255, A:0)	TM05

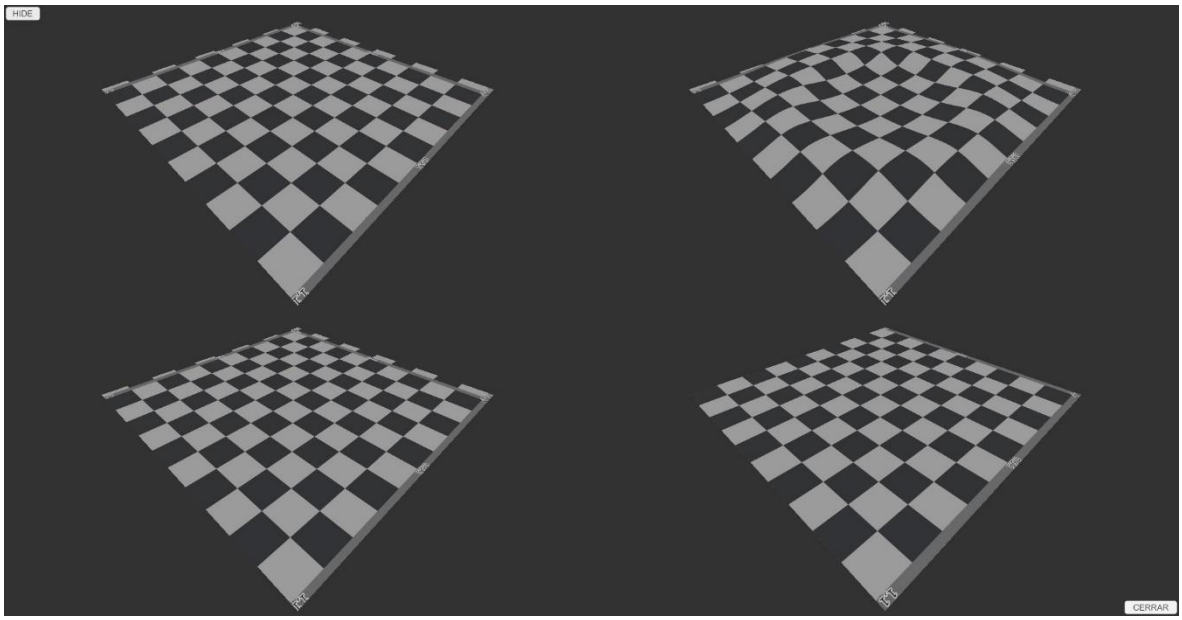


Fig. 89. De izquierda a derecha y de arriba a abajo - Modelos con HR, modelo con HG, modelo con HB y modelo sin textura aplicada en el mapa de *Height*, y MT05 en el albedo. Elaboración Propia.

Observamos que el modelo 2 (arriba a la derecha), con la textura HG aplicada, presenta una ligera deformación en su superficie con forma de cráter. Observamos también que los modelos 1 y 3, con las texturas HR y HB aplicadas respectivamente, están ligeramente desplazadas. Aunque sus valores de R y B no producen efecto, al aplicarle una textura a su mapa de relieve el valor G, sigue estando presente, aunque sea 0, lo que nos indica que los valores bajos simularán visualmente un desplazamiento hacia abajo, mientras que los valores altos de G en un píxel simularán un desplazamiento hacia arriba.

Tras varias pruebas con valores de G en una textura, descubrimos que el punto medio es el valor de G = 185. El *shader standard* de *Unity* tiene configurado un parámetro con valores entre 0.005 y 0.08 que modifican la deformación producida por el mapa de relieve ajustándose sobre el valor de 185, valor neutro de G para el mapa de relieve.

La documentación sobre este tipo de texturas indica la utilización de imágenes en escala de grises, por lo que procedemos a aplicar nuestra Textura TM02, tanto en el mapa de relieve como en el mapa de albedo, para poder percibir el efecto y tener presente las referencias de color. Pero el resultado es engañoso, ya que ciertos tonos de grises nos hacen entender que el valor de G es diferente a la de sus secciones cercanas, y esto se debe al efecto de la opacidad, por lo que modificamos la textura eliminando los valores de opacidad, añadiendo una capa blanca debajo que genere diferentes niveles de grises, y generando una nueva textura (TM02_01) (Fig. 90).

Fig. 90. Textura TM02_01 creada a partir de TM02 (TM01 en escala de grises). Elaboración propia.

El efecto visual da la sensación de desplazarse de acuerdo con los diferentes tonos de gris, pero esto únicamente se debe a los diferentes valores de G contenidos en cada tono de gris (Fig. 91).

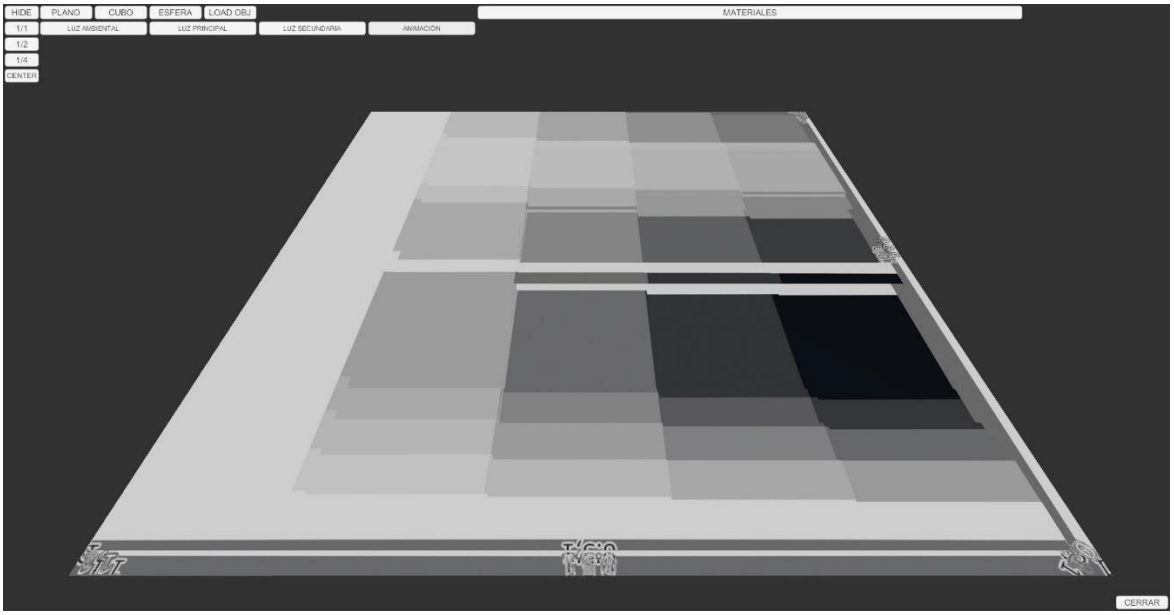


Fig. 91. Captura de la textura TM02_02 aplicada en el mapa de relieve y el Albedo. Elaboración propia.

Luz

El desplazamiento de los píxeles, provocado por el mapa de relieve y su efecto visual de superposición de objetos, no se aplica a la luz, como podemos ver en las imágenes del proceso de experimentación. La luz no ha generado ningún efecto de luces y sombras sobre la superficie.

RELLENO

Cuando el efecto del mapa de *Height* desplaza los píxeles, la textura se desencaja de la superficie del polígono, como si el mapeado UV se hubiera desplazado. Es importante, al utilizar este tipo de mapas, la utilización de texturas sin costuras, o el uso de *shell*, tratado anteriormente en esta misma tesis.

3.3.5.5.- OCCULSION

Aplicamos la textura TM01 al mapa de Oclusión de nuestro *shader standard* de Unity, quedando aplicado a través del material en nuestro modelo de cubo por defecto (Fig. 92).

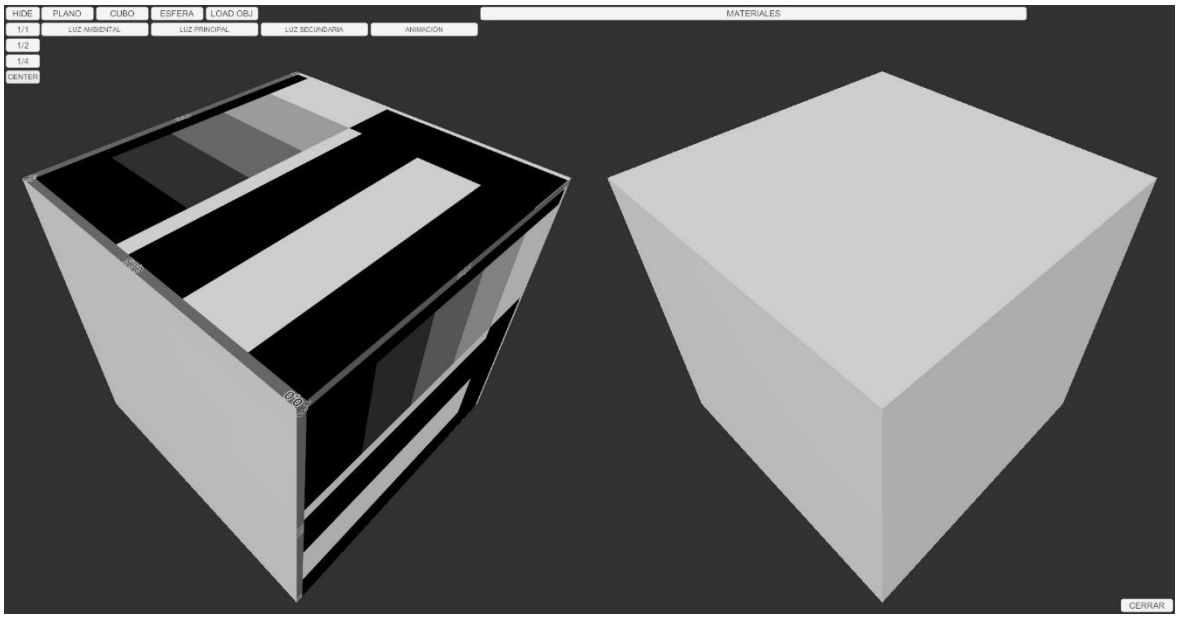


Fig. 92. Modelo de cubo con la textura MT01 aplicada en su mapa de oclusión (izquierda) y cubo sin textura aplicada (derecha). Elaboración propia.

El mapa de oclusión ambiental representa las zonas de mayor y menor incidencia de la luz en la superficie del modelo y éste se crea o precalcula para su posterior aplicación dentro del entorno de renderizado, y actúa como modificador de la luz de la escena.

En nuestro caso, la cara superior y derecha de nuestro cubo no reciben luz directa, por lo que el mapa de oclusión ambiental hace su trabajo de acuerdo a los parámetros aportados por la textura, mientras que, en la cara izquierda, al incidir la luz directamente sobre la superficie (recordemos que se trata de una luz direccional constante), no aplica los parámetros de oclusión ambiental aportados por la textura, ya que no encuentra zonas de sombra.

Preparamos un modelo más complejo que nuestros modelos por defecto para poder visualizar mejor los efectos de este mapa sobre la superficie. Recurrimos al modelo de suzanne que viene por defecto en *Blender*, con una subdivisión extra de malla y un sombreado suavizado

para evitar las superficies muy definidas. Lo cargamos en nuestro entorno de pruebas y le aplicamos la textura TM01 en el mapa de oclusión (Fig. 93).

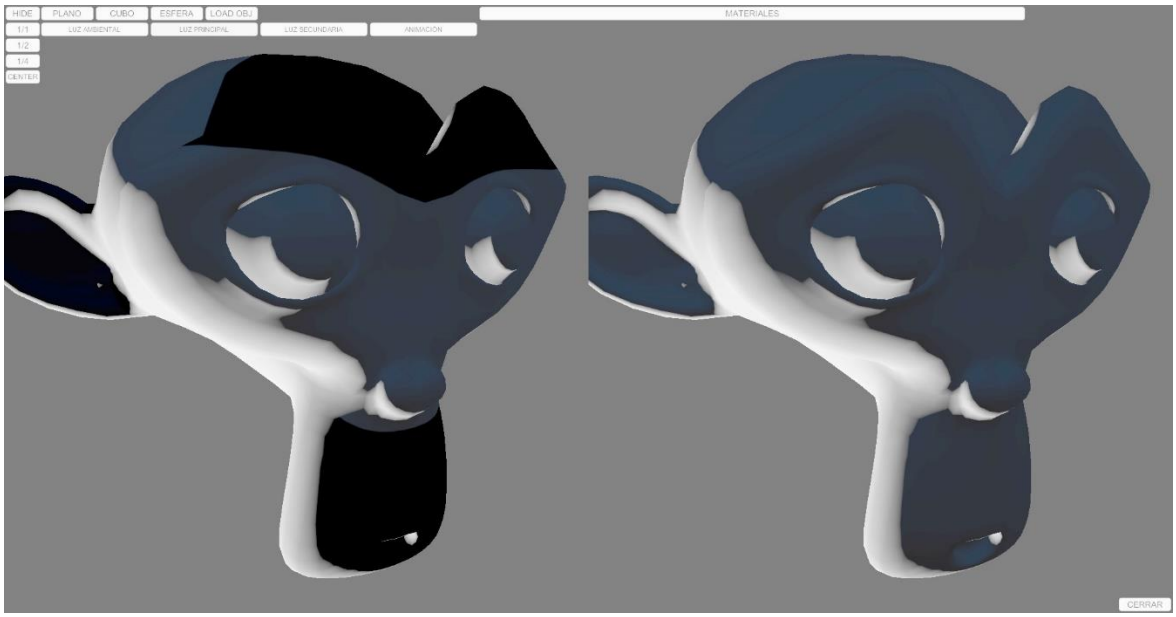


Fig. 93. Captura (niveles medios modificados) del modelo Suzanne con la textura MT01 aplicada en el mapa de oclusión (izquierda) y sin mapa de oclusión Derecha. Elaboración propia.

Muy sutilmente vemos cómo, en las superficies donde la luz incide directamente y no se daría motivo para una sombra al no existir oclusión a la luz, la textura de oclusión ambiental no se aplica. Según la luz, va perdiendo fuerza o incide de forma menos directa, pasando a zonas de sombra sobre la superficie de nuestro modelo. Vemos cómo el mapa de oclusión ambiental multiplica la fuerza de la sombra.

El *shader standard* de *Unity* también incluye un multiplicador de valores entre 0 y 1 para modificar el efecto de la textura de oclusión ambiental.

Para ver mejor cómo reacciona la superficie a los parámetros RGBA de la textura, volvemos a fijarnos en nuestro modelo de cubo por defecto. Los píxeles con valores de opacidad igual a 0 (completamente transparentes), son descartados y se pierden los parámetros de RGB, y si el valor de opacidad es distinto de 0 (tienen opacidad) transfieren a la superficie su valor de RGB completo.

Respecto a los parámetros RGB, vemos que el nivel de sombra depende del parámetro G, siendo R y B despreciado y tomados como sombra máxima (sin efecto de la luz). Para comprobarlo, aplicamos nuestras texturas HR, HG y HB, utilizadas para hacer comprobaciones en el mapa de relieve, en el mapa de oclusión.

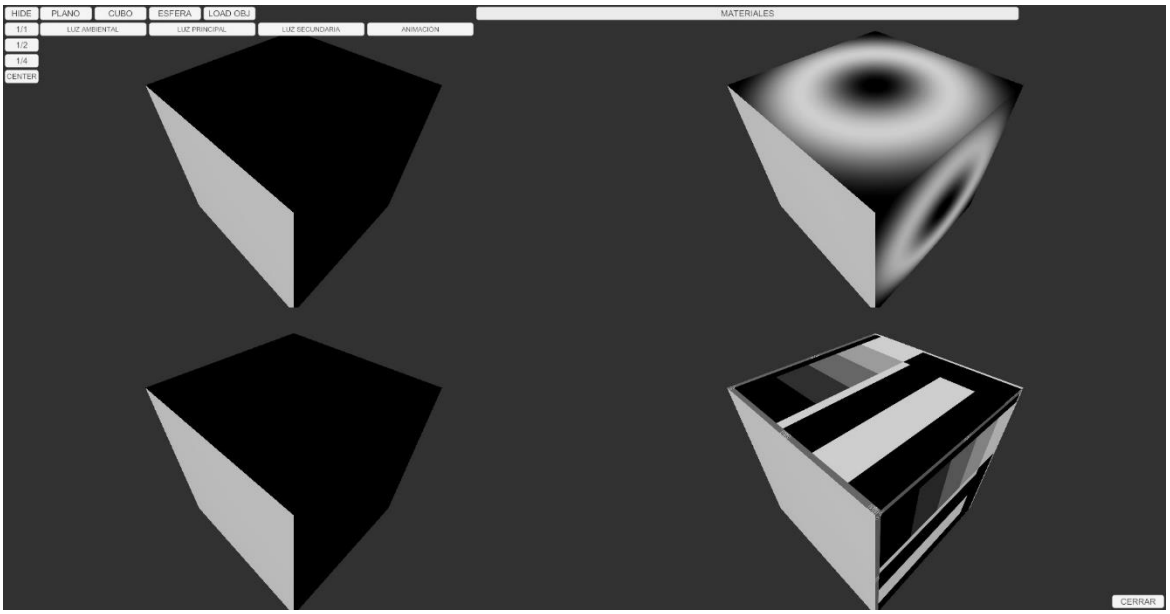


Fig. 94. De izquierda a derecha y de arriba a abajo - Modelo de cubo por defecto con la textura HR, HG, HB y TM01 respectivamente aplicadas en el mapa de oclusión. Elaboración Propia.

Como se preveía, sólo la textura HG produce modificaciones en la superficie del modelo, siendo G=0 la ausencia total de luz y G=1 la interacción máxima de la luz sobre la superficie (Fig. 94).

La documentación sobre este tipo de texturas indica que ha de trabajarse con imágenes en escala de grises, por lo que nuevamente procedemos a aplicar nuestra textura MT02, para ver si varían los resultados respecto al estudio realizado con la textura MT01, y vemos cómo se cumplen los mismos resultados, dependiendo el efecto de oclusión ambiental sólo del parámetro G, y cómo el valor de A es interpretado de acuerdo con su valor expuesto anteriormente (Fig. 95).

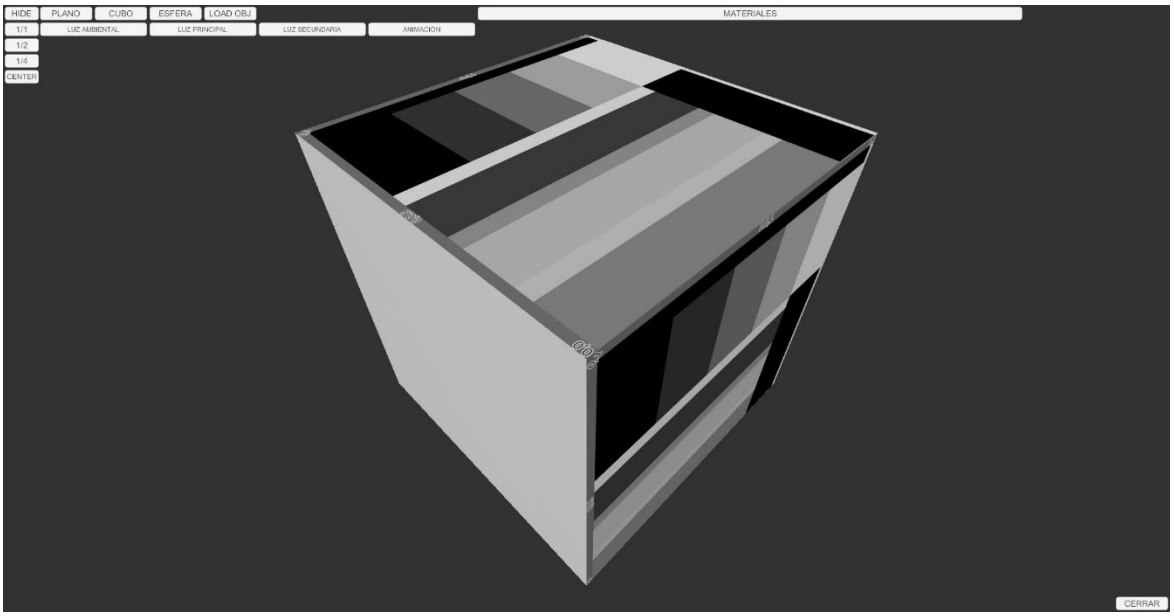


Fig. 95. Captura del modelo del cubo con la textura MT02 aplicada en su mapa de oclusión. Elaboración propia.

3.3.5.6.- EMISSION

Este material no es imprescindible para la generación de un material PBR per se, sino que, debido a los nuevos sistemas de GI, existe la posibilidad de que un objeto (y en concreto su superficie) puedan ser emisores de luz en la escena y afectar a otros objetos.

Para poder estudiar los efectos de este mapa de textura en la escena como objeto de luz, necesitamos configurar otro tipo de entorno denominado como caja de Cornell, en la que pudiéramos ver cómo la radiación o luminosidad generada por el objeto actúa sobre el resto de los elementos de la escena.

En esta tesis, nos centraremos en cómo el mapa de emisión modifica la superficie del objeto sobre el que se aplica a través de su material.

Procedemos entonces a aplicar a nuestro cubo por defecto en nuestro entorno la textura MT01 en el mapa de *Emission*, y procedemos a estudiar los resultados (Fig. 96).

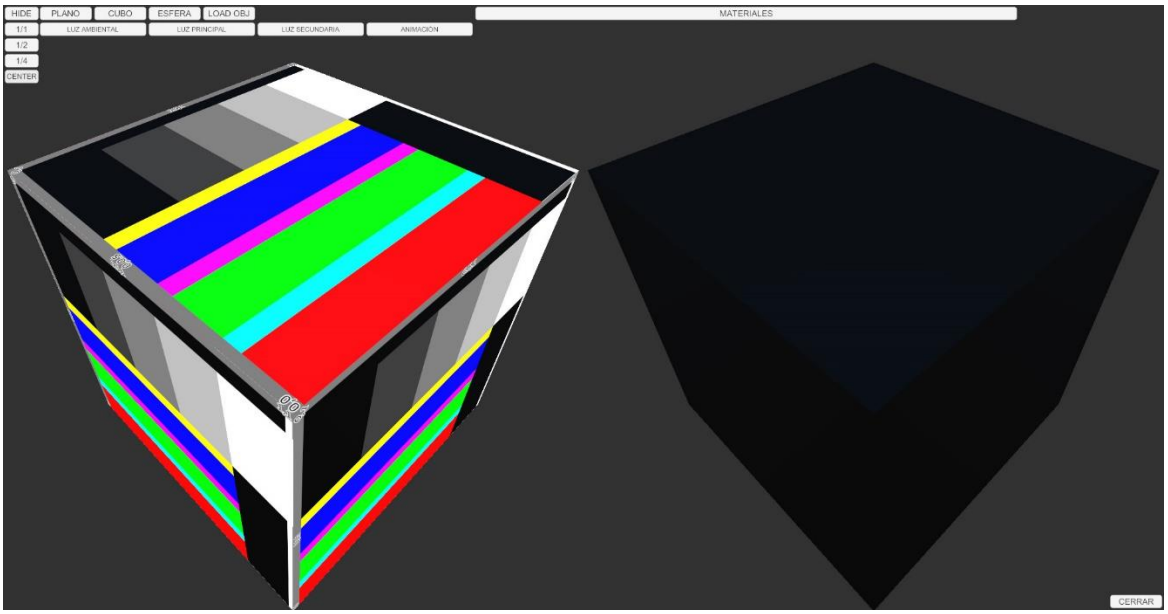


Fig. 96. Entorno sin iluminación. Modelo de cubo por defecto con la textura MT01 aplicada en su mapa de emisivo (izquierda) y sin ninguna textura aplicada (derecha). Elaboración propia.

Podemos ver cómo el objeto, a pesar de no tener ninguna textura en su mapa de Albedo, está cubierto de color de acuerdo con los valores aportados por la textura en sus parámetros RGB. El parámetro A, lo descarta en sus valores superiores a 0, leyendo el dato de RGB del píxel, y el Alfa 0 lo interpreta como negro (R=G=B=0) sin tener en cuenta los valores de RGB aportados por la textura para esos píxeles.

El *shader standard* de *Unity* tiene codificado, además, un multiplicador de color de tipo HDR (*High Dynamic Range*) o alta gama dinámica, que define la cantidad de luz que es capaz de proyectar en una superficie. Este tipo de color se compone de los parámetros R (*Red*), G (*Green*), B (*Blue*) e I (Intensidad). Al modificar estos parámetros (RGBI), vemos cómo varía la intensidad del color de la superficie (Fig. 97 y Fig. 98).

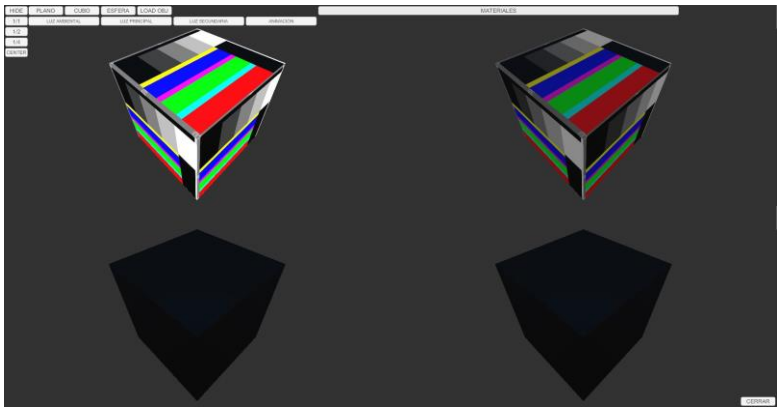


Fig. 97. De izquierda a derecha y de arriba a abajo - Cubo por defecto con textura MT01 en el mapa de emisivo con variaciones de RGB, 255, 128, 0 respectivamente y modelo 4 de control sin mapa de *emission* (abajo derecha). Elaboración propia.

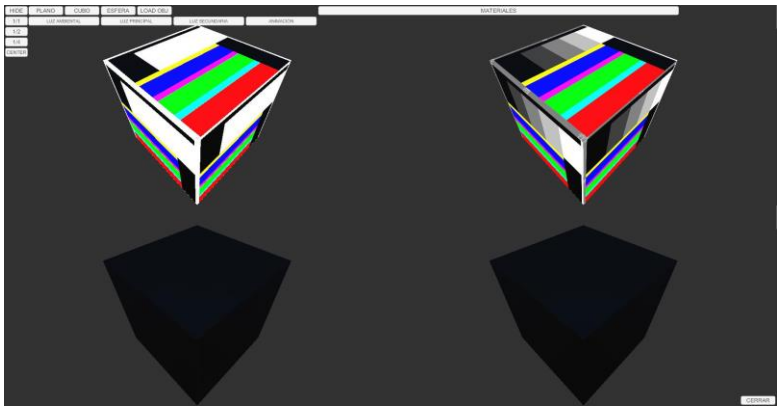


Fig. 98. De izquierda a derecha y de arriba a abajo - Cubo por defecto con textura MT01 en el mapa de emisivo con RGB 255 y variaciones de I, 10, 0, -10 respectivamente y modelo 4 de control sin mapa de *emission* (abajo derecha). Elaboración propia.

3.4.- ANÁLISIS DE LOS RESULTADOS

Tras una larga fase de investigación y estudio, y otra de experimentación y observación, en la que se ha trabajado con los diferentes parámetros que configuran las superficies a través de los materiales PBR en el entorno de *Unity*, procedemos a estudiar los resultados.

Herramienta técnica

En primer lugar, a nivel técnico, hemos obtenido una relación entre los parámetros RGBA de las texturas y su interacción con el *shader* (Tabla 9), y gracias a este cuadro, vemos algunas discrepancias entre la documentación y la práctica. Por ejemplo, *Unity* indica textualmente: "*An occlusion map is a greyscale image, with white indicating areas that should receive full indirect lighting, and black indicating no indirect lighting*" [Un mapa de oclusión es una imagen en escala de grises, con blanco que indica áreas que deben recibir iluminación indirecta completa y negro que indica que no hay iluminación indirecta].

Nuestra práctica ha demostrado que sólo el parámetro G del píxel es el que genera modificaciones en la superficie.

Interpretamos que el uso de imágenes en escala de grises en los diferentes mapas y programas puede deberse a la necesidad de visualizar las formas de la imagen para su asimilación a la hora de trabajar con ellas, ya que los píxeles en escala de grises tendrán igualmente valores de G entre 255 (blanco) y 0 (negro), pero obliga a usar una textura para cada mapa del *shader,* haciendo que los proyectos sean más pesados en sentido tecnológico.

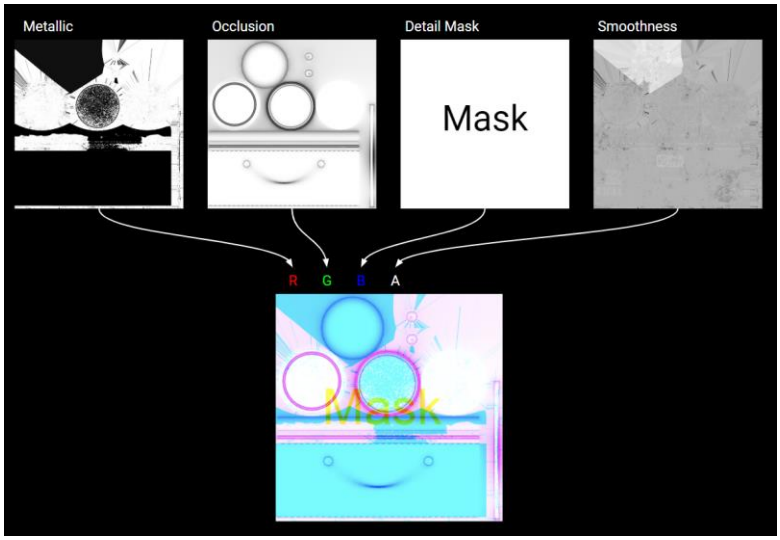


Fig. 99. Esquema de encapsulamiento del mapa de *Metallic*, *Oclussion*, *Detail Mask* y *Smoothness* en los canales de una misma textura - https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@10.2/manual/Mask-Map-and-Detail-Map.html

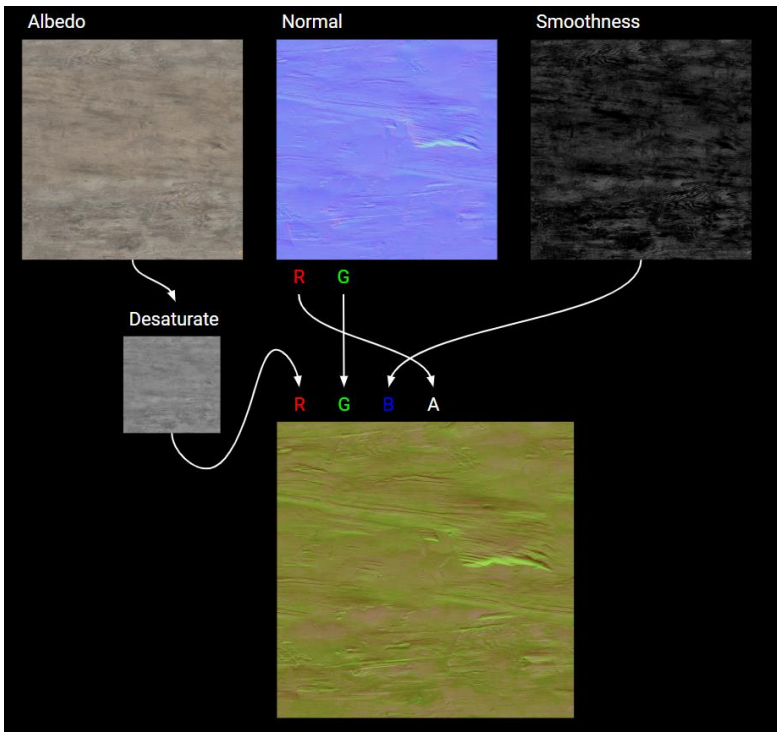


Fig. 100. Esquema del encapsulamiento de los mapas de Albedo, Normal y Smoothness en una misma textura - https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@10.2/manual/Mask-Map-and-Detail-Map.html

Sistemas más avanzados de *shader* para materiales PBR, como el usado en el propio motor de *Unity* para el HDRP, utiliza los canales de RGBA para encapsular la información, leyendo los datos correspondientes a cada mapa del *shader* en su canal correspondiente, reduciendo así el número de texturas necesarias para la creación de un material de idénticas características (Fig. 99 y Fig. 100).

Tabla 9. Relación de los componentes del *shader standard* de *Unity* y los parámetros de estudio - Verde si aplica, Naranja si aplica con condiciones, negro si no aplica. Elaboración propia.

	PARÁMETROS				
MAPA	**R - *Red***	**G - *Green***	**B - *Blue***	**A - *Alfa***	**I-*Intensity***
Albedo	🟢	🟢	🟢	⚫	⚫
Opacity	⚫	⚫	⚫	🟢	⚫
Metallic	🟢	⚫	⚫	⚫	⚫
Smoothness	⚫	⚫	⚫	🟢	⚫
Normal	🟢	🟢	🟢	🟠	⚫
Height	⚫	🟢	⚫	⚫	⚫
Occlusion	⚫	🟢	⚫	⚫	⚫
Emission	🟢	🟢	🟢	🟢	🟢

Creación artística

Por otro lado, desde un enfoque artístico, hemos podido obtener una relación entre los diferentes mapas y el tipo de modificación que realizan sobre los píxeles de la imagen. Estos resultados son válidos para cualquier tipo de *shader,* sin importar el tipo de *Render Pipeline*. Si un usuario define un mapa de oclusión, los resultados sobre el píxel siempre serán los mismos.

Tras visualizar el efecto de todos los píxeles modificados sobre la superficie de los objetos, hemos podido extrapolar una serie de efectos visuales específicos para cada mapa del *shader* de acuerdo con la siguiente tabla (Tabla 10).

Tabla 10. Relación de los mapas de textura con el tipo de efecto que tienen sobre cada píxel y el efecto visual final sobre la superficie. Parámetros de color T - Tono, S - Saturación, L - Luminosidad, A - Alfa

MAPA	MODIFICACIÓN POR PIXEL	EFECTO VISUAL SUPERFICIAL
Albedo	Color (T, S, L)	Color base de la superficie
Opacity	Color (A)	Transparencia - opacidad
Metallic	Color (T, S, L)	Metalicidad
Smoothness	Color (T, S, L)	Suavidad - Aspereza
Normal	Color (L)	Sombras (volumen)
Height	Desplazamiento	Volumen
Occlusion	Color (L)	Sombras (luz)
Emission	Color (T, S, L)	Luminosidad

Un material PBR está diseñado para simular materiales del mundo real en entornos virtuales, pero, trabajando con los diferentes valores y seleccionando los mapas de texturas por separado, se pueden obtener efectos visuales más allá de su diseño original o de la simulación virtual de una superficie real.

A través del estudio previo a la experimentación, hemos definido los parámetros y configuraciones necesarios para generar diferentes estéticas durante el proceso de trabajo en entornos con estas características, ya sea la creación de escenarios *cartoon* a través del Cel *shader*, renderizados retro al más puro estilo del videoclip del tema *Money for nothing*[38] configurando un flat *shader* y desactivando el anti-aliasing, escenarios con la estética de TRON, utilizando los parámetros de emisivo, o escenarios fotorrealistas, configurando los diferentes parámetros de un *shader* para un material PBR, y esto si nos centramos en entornos existentes, ya que siempre podemos ir un paso más allá de la mano del conocimiento.

[38] *Money for nothing* (1985) *Dire Straits*.

4.- APLICACIÓN PRÁCTICA

En base a los resultados obtenidos durante la experimentación, vamos a realizar una demostración práctica en la que veremos cómo un modelo 3D puede presentar múltiples aspectos o resultados visuales a través de los materiales aplicados al mismo, haciendo que el espectador perciba la figura y su entorno de forma completamente diferente.

El modelo elegido para la realización de esta práctica ha sido descargado de un market digital[39] por dos razones: por una parte, porque esta tesis no se centra en el modelado de los objetos digitales, que también es tema muy interesante de estudio, y por otra parte, para demostrar cómo un modelo no diseñado para esta finalidad puede ver modificada la percepción que el observador tiene del mismo sólo modificando sus texturas.

Se trata de una figura humanoide basada en el personaje *Exo Stranger*, o Elisabeth Bray, de la popular saga de videojuegos *Destiny* (Fig. 101). El modelado es un *mid poly* pensado para ser usado en motores gráficos, y el proyecto descargado incluye diferentes texturas obtenidas a partir de su modelo *high poly* a través de sistemas de *bake*.

Para desarrollar esta práctica, a parte de los programas citados durante la experimentación previa, utilizaremos el programa *Substance Painter* de *Adobe* para la edición de los materiales y sus texturas.

Lo primero que vamos a hacer es aplicar diferentes materiales básicos al modelo (Fig. 102). Al observar ahora el objeto 3D con los diferentes materiales aplicados, la percepción de las características físicas, como dureza, fragilidad o elasticidad del modelo, varían sólo a través de la modificación de parámetros y mapas de texturas de los materiales PBR aplicados. Con estos sencillos cambios, obtenemos aspectos tan dispares sobre un mismo modelo como una estatuilla de porcelana, de piedra o madera, o el aspecto de un muñeco de plástico.

[39] Descarga https://www.cgtrader.com/3d-models/character/woman/exo-stranger-character-rig-for-blender

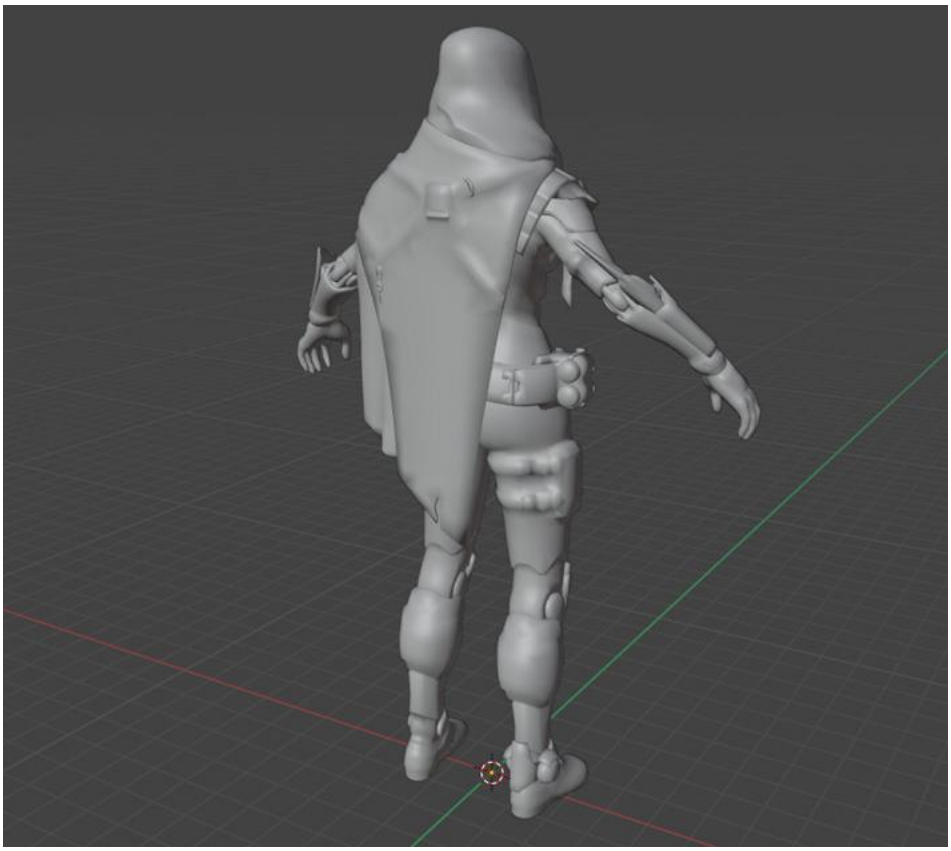


Fig. 101. Modelo 3D seleccionado para la práctica. Elaboración propia.

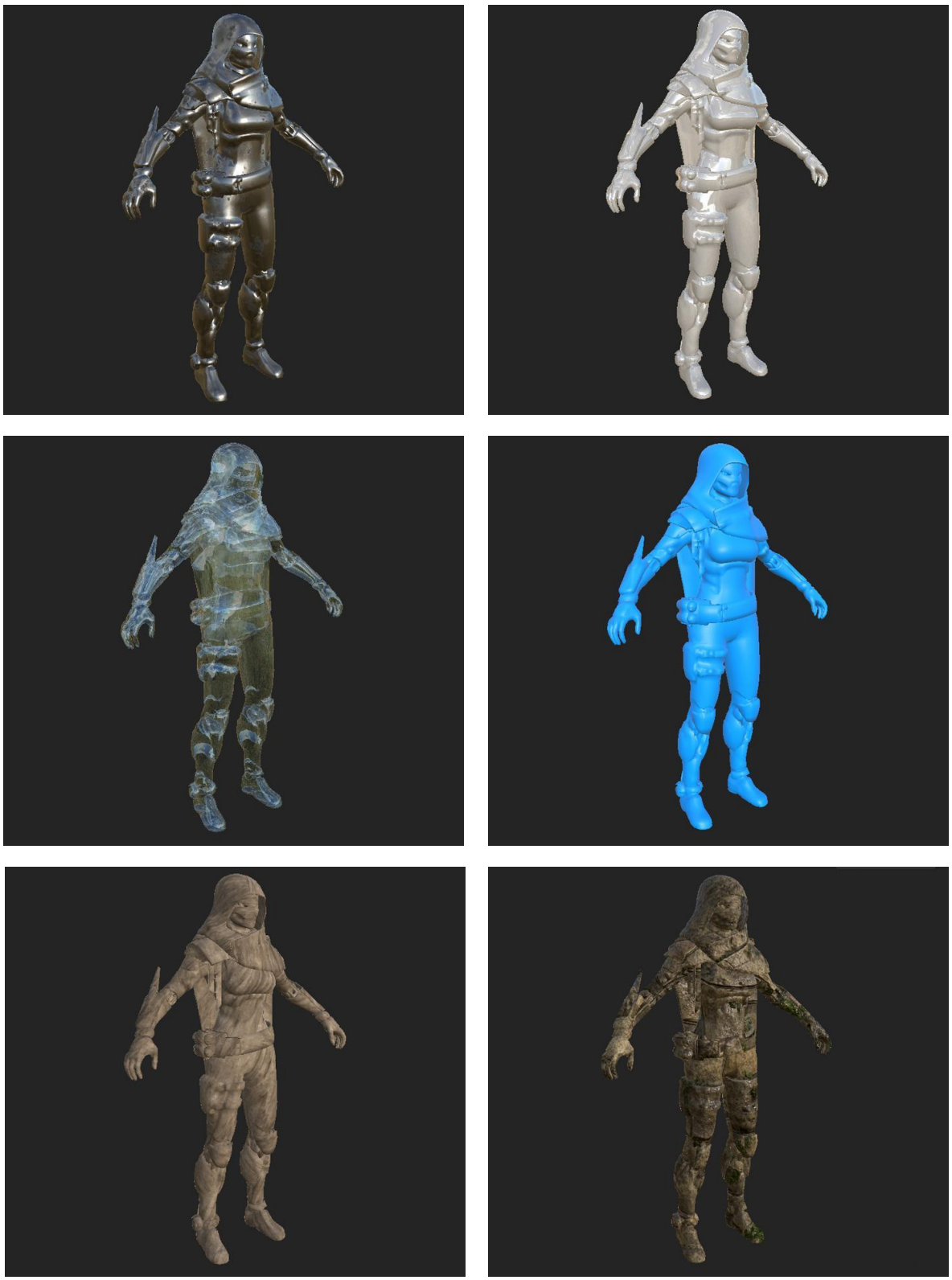


Fig. 102. De arriba a abajo y de izquierda a derecha. Mismo modelo 3D con material de acero, porcelana, cristal, plástico, cuero y piedra. Elaboración Propia.

Podemos, además, modificar la escala encajando más o menos parte del material y de sus texturas en los polígonos de la malla. Si tomamos, por ejemplo, el material de piedra, podemos aplicarlo simulando una pequeña talla, pero si reducimos la textura (o aumentamos el mapeado UV) aprovechando el "tileado" de las texturas, convertimos una pequeña talla de piedra en una escultura del tamaño de un coloso que pasa de cabernos en la mano (virtual) a ser un monumento que visitar y escalar (virtualmente) (Fig. 103 y 104).

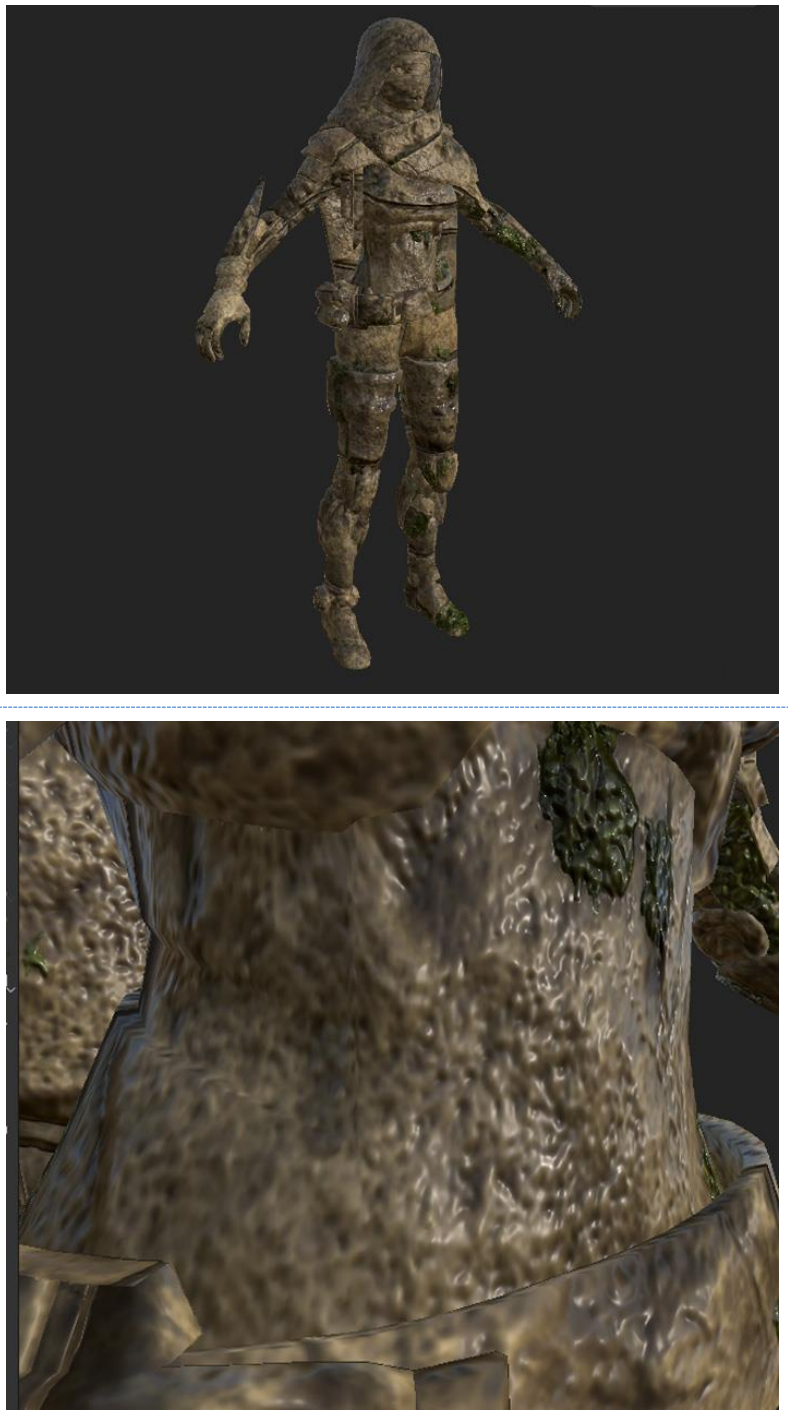


Fig. 103. Aplicación de un material de piedra escalado para simular un objeto de piedra de tamaño pequeño. Arriba, vista general, abajo, detalle del vientre del modelo.

En este ejemplo podemos ver cómo, con un mismo modelo y un mismo material, puede cambiar la percepción del objeto virtual utilizando las herramientas previamente analizadas en esta tesis. Aunque este efecto, en parte, también dependerá del tamaño del objeto en la escena, sí es verdad que una mala texturización del modelo haría que ese efecto se perdiera. A través de modificaciones en los materiales y sus texturas, ya no sólo modificamos cómo percibimos el objeto, sino su volumen en el espacio virtual.

Fig. 104. Aplicación de un material de piedra escalado para simular un objeto de piedra de tamaño colosal. Arriba, vista general, abajo, detalle del vientre del modelo.

Los materiales aplicados sobre un modelo 3D pueden incluir muchos detalles. Si tomamos el modelo con el material escalado para tener el aspecto de una estatua colosal, podemos editar el material de piedra en forma de sillares para que incluya más detalles, por ejemplo, relieves de una talla más compleja realizada sobre la propia piedra (Fig. 105). Estos detalles pueden estar incluidos en diferentes mapas de texturas. En este caso, hemos utilizado el mapa de normales y el de oclusión ambiental.

Fig. 105. Detalle del modelo con la textura escalada para simular una estatua colosal (arriba). Detalle de la modificación en el mapa de normales y mapa de oclusión ambiental para incluir detalles tallados en la roca (abajo). Elaboración propia.

Trabajando y modificando las texturas conseguimos diferentes efectos visuales que definen el aspecto de los objetos y la información que recibimos de ellos en el renderizado, y el resultado definirá al objeto u objetos mostrados en la imagen.

Avanzando en esta práctica, le aplicamos a nuestro modelo dos materiales específicos. Uno de inspiración *Steampunk*, y otro basado en un universo *Cyberpunk* (Fig. 106).

Un mismo modelo 3D al que se le aplican materiales distintos puede, ya no sólo percibirse de una manera diferente según los materiales, sino que puede, o no, incluir más o menos detalles, como ya hemos comentado. Está, además, definiendo el entorno que le rodea. En este ejemplo, siendo el mismo modelo, de acuerdo a los materiales, insertar uno en la ambientación del otro se consideraría un anacronismo, o se diría que está totalmente fuera de lugar y descontextualizado. De hecho, el modelo texturizado con ambientación *Steampunk* no invita a ser integrado en una ambientación de la Inglaterra victoriana, más bien nos transporta a un fantástico, salvaje y lejano oeste americano, mientras que los materiales *cyberpunk* nos presentan un universo de temática *Sci-fi* fantástica con un tono oscuro o postapocalíptico.

Hasta este punto de la práctica, hemos recurrido a la creación de materiales que simulan el mundo real, pero los materiales PBR pueden ir mucho más allá, gracias al amplio abanico de propiedades y mapas de texturas, y las múltiples configuraciones posibles.

Fig. 106. Modelo 3D con materiales diseñados en base a ambientación *Steampunk* (arriba). Modelo 3D con materiales diseñados en base a ambientación *Cyberpunk* (abajo). Elaboración propia.

Modificando los parámetros de transparencia del material hasta volver la figura invisible y aumentando la reflexión hasta conseguir la refracción máxima de la luz sobre la superficie del modelo, generamos un nuevo material que, a simple vista en el editor de nuestro motor gráfico, no es visible. Colocando una serie de luces en la escena con diferentes valores de color y configurándolos como luces calculadas en tiempo real, el resultado es el siguiente (Fig. 107).

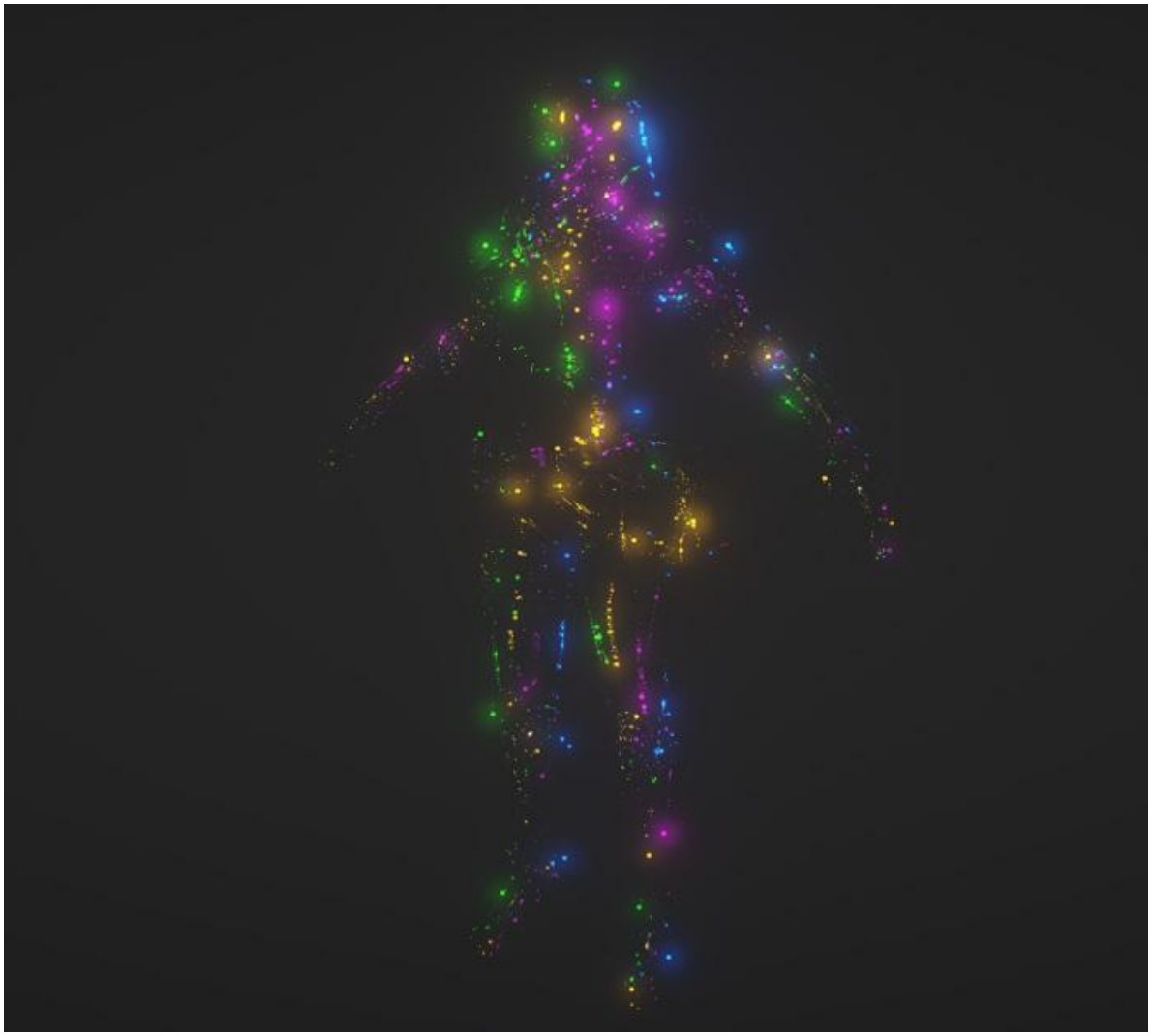


Fig. 107. Modelo 3D con material de creación propia. Elaboración propia.

Imposible de encontrar en el mundo real, se trata de un material con una configuración muy concreta, con un efecto visual único, y es sólo un ejemplo de las diferentes posibilidades que ofrecen este tipo de materiales.

5.- CONCLUSIONES

Como ha quedado demostrado, esta investigación se enmarca en un contexto de Artes y Humanidades con un enfoque técnico y centrado en la tecnología con la que deben trabajar actualmente los artistas 3D. Este trabajo indaga y explora las posibilidades creativas digitales en el contexto de las herramientas tecnológicas de aplicación en los entornos con renderizado en tiempo real.

Esta investigación facilita el uso y la formación sobre estas herramientas en el campo de las artes, y da un enfoque artístico a la terminología técnica que impera en este sector.

Ponemos de relieve la importancia y potencial de los materiales PBR y su aplicación en el ámbito artístico a través del estudio y análisis de sus predecesores y sus aplicaciones más frecuentes.

Durante la contextualización, hemos realizado un recorrido por diferentes ámbitos de conocimiento a través del análisis de la literatura relacionada con el tema, comenzando con las primeras imágenes generadas por ordenador en el ámbito de la informática, pasando por la producción audiovisual, llevando la cuestión hacia un ámbito artístico, afín a los entornos con renderizado en tiempo real, y finalmente, el ámbito de los videojuegos, que nace de la combinación de ambos mundos.

La revisión teórica explora las bases comunes de los principales y más potentes motores gráficos comercializados, como *Unity* o *Unreal Engine*, tienen una versión gratuita para desarrolladores *amateur* o estudiantes, permitiendo al desarrollador experimentar y trabajar con estas herramientas. A esto se añade que tenemos acceso a documentación, guías, ejemplos, tutoriales y videotutoriales en formatos digitales online, por lo que se ha democratizado aún más el acceso y uso de estos nuevos medios. Esto ha impulsado a muchos desarrolladores a embarcarse en proyectos basados en entornos virtuales con renderizado en tiempo real, pero, al mismo tiempo, también genera un cuello de botella que los encarrila o

direcciona hacia objetivos y resultados similares. Es decir, coarta la aventura de probar y experimentar con las nuevas herramientas digitales en este campo.

La utilización de estos materiales también se relaciona con los nuevos sistemas de visualización e interacción con el medio virtual que han llegado con la segunda ola de la VR. Los visores de Realidad Virtual y los controles espaciales, la Realidad Aumentada en la palma de la mano a través de cualquier dispositivo inteligente, o los visores de Realidad Mixta, son nuevos medios multimedia que permiten al espectador la visualización de objetos y mundos inexistentes, materializados digitalmente desde a creatividad del artista a través de estas herramientas.

Durante el proceso de elaboración de esta tesis, se ha estudiado, trabajado y analizado la situación actual de dichos avances, atendiendo los contextos del *hardware* como del *software*, utilizando una visión técnica desde una perspectiva artística, desglosando la terminología específica que el artista digital debe manejar en este campo.

Con la aplicación de los materiales PBR, utilizados en producción audiovisual, a los entornos 3D con render *real-time* junto a los nuevos sistemas de iluminación, entre otros avances tecnológicos, para el público consumidor no le pasan desapercibidos los últimos trabajos en este ámbito que a la fecha de esta tesis, han sido desarrollados en *Unity* o *Unreal Engine*.

Tenemos acceso a bibliotecas y colecciones, tanto de materiales como de mapas de texturas prediseñados, listos para integrarlos en proyectos realizados en entornos 3D, y existe *software* específico (como las herramientas *Substance* de Adobe) en los que podemos generar nuestros propios materiales y texturas. Estos son utilizados en diferentes ámbitos, como la producción audiovisual, arquitectura o diseño gráfico, pero es difícil encontrar información sobre cómo desarrollar este material tan necesario a la hora de configurar un entorno 3D.

En esta tesis se han expuesto de forma precisa los múltiples componentes de los materiales PBR y se ha estudiado su funcionamiento gráfico y visual, sin entrar en valoraciones fuera del ámbito de las artes, ateniéndonos a conocimientos propios del ámbito artístico.

En base a todo el trabajo previo, se han contextualizado los resultados obtenidos durante la experimentación, que ubica este trabajo en su contexto actual de las artes digitales, sirviendo como base para continuar trabajando en este campo, y proporcionando un esquema desglosado de fácil referencia para el uso de los diferentes mapas y su función en el *shader* para la generación de materiales PBR.

Con la práctica, queda demostrada la aplicación de los conocimientos desglosados a lo largo de este texto. Cómo un mismo objeto 3D virtual, a través de la modificación de los diferentes mapas de texturas que componen el material PBR aplicado al mismo, y los parámetros que lo definen, modifica radicalmente la percepción del espectador de ese mismo objeto, y cómo pueden estos mismos cambios, no sólo imitar el entorno real, sino crear conceptos totalmente nuevos imposibles de ver en el mundo real.

Anticipando el desarrollo de esta investigación, en 2018, durante las conferencias realizadas en Zaragoza para el congreso *Arte, patrimonio y tecnología en la era digital* se publicó el artículo *Nueva rama profesional: Artistas 3D*[40] (Luis Tello, 2018) en el que se planteaba una nueva ramificación dentro del arte digital orientada al arte en entornos 3D y al desarrollo de entornos inmersivos, resaltando en dicho artículo las tareas y disciplinas propias de las artes de dicho perfil artístico digital, directamente relacionadas con las áreas de conocimiento que posteriormente desglosamos en esta tesis y que pueden verse en el esquema del flujo de trabajo presentado en el apartado de Metodología[41].

En relación a este tema, en 2021, durante el *VIII Encuentro Internacional de Film de Danza y Música* se presenta la ponencia *Videojugos y el cuerpo el movimiento: Más allá de la silla*[42] (Luis Tello, 2021), en la que se exponía cómo el usuario de videojuegos de temática musical desarrollados para los medios interactivos y a su inmersión en los mismos debido, en mayor o menor medida, al nivel gráfico de la aplicación, se sentía más atraído hacia la idea de bailar delante de una pantalla, y cómo a través de este medio podía aprender coreografías e

[40] https://oaaep.unizar.es/wp-content/uploads/2019/09/ArteTecnologiaDigital.pdf pp. 227 - 229
[41] Esquema del fujo de trabajo desarrollado desde la página 61 en adelante.
[42] https://riuraufilmfestival.com/wp-content/uploads/2022/08/2021-Publicacion-VIII-Ponencia-FILM-DE-DANZA-web.pdf pp. 47 - 52

imitarlas fuera de su tiempo de juego como quien imita un baile aprendido en un videoclip musical o incluso en una película.

En el evento *La Noche Europea de los Investigadores e Investigadoras* del 2022, evento de difusión de la investigación de la Universidad de Zaragoza con actividades simultáneas en los Campus de Zaragoza, Huesca y Teruel, financiado por la Comunidad Europea (CE) bajo las Acciones Marie Skodowska-Curie (MSCA) del Programa Horizonte H2020, también en relación con el trabajo desarrollado en esta tesis, se presenta la ponencia *Entornos Virtuales – Inmersión e Interacción*. Desde el punto de vista de la evolución de los gráficos en el sector de los videojuegos se estudia la inmersión del usuario a diferentes niveles en la aplicación, resaltando la importancia de los materiales PBR en la creación de entornos inmersivos, aplicaciones interactivas y los videojuegos.

Durante la elaboración de esta tesis, se publica en la *Association for Computing Machinery* (ACM) durante el *XXII Congreso Internacional de Interacción Persona-Ordenador*, un artículo titulado *Interaction According to Immersion in Virtual Environments: Graphic Development and PBRs in Environments with Real-Time Rendering and Virtual Reality*[43](Luis Tello, Burgos Risco, Robles Reinaldos, 2022) en el que queda justificado, como extensión de los postulados de esta tesis, que el desarrollo de entornos inmersivos con renderizado en tiempo real es esencialmente un proceso relacionado con disciplinas artísticas, y no sólo eso, sino que la interacción del usuario con el dicho entorno dependerá de su inmersión, directamente relacionada con el nivel gráfico de la misma.

Hay otras disciplinas o competencias de sesgo exclusivamente tecnológico o psicológico entre otras, que se relacionadas con el desarrollo de entornos virtuales que no se han tratado en este documento, y que también se engloban dentro de las artes, como por ejemplo la animación y la captura de movimiento, el trabajo con físicas en los entornos con *render real-time*, o el uso de la programación como elemento de la creación artística porque así se concibe esta tesis desde su introducción e hipótesis, centrándonos en los aspectos relevantes de la creación artística digital que aluden a los materiales PBR.

[43] https://dl.acm.org/doi/10.1145/3549865.3549896

Se trata de un nuevo campo, como lo fueron la fotografía o el cine en su momento, en el que los artistas tienen mucho que decir y aportar, pero también hay mucho que analizar, adaptar y estudiar. Esta tesis doctoral pretende ofrecer su aportación en este ámbito abriendo posibilidades de ampliación y profundización a través de una línea de trabajo en constante evolución.

6.- REFERECIAS

6.1.- BIBLIOGRAFÍA ESPECIALIZADA

AMBROSE, G., SALMOND, M. (2014) “Los fundamentos del diseño Interactivo” Barcelona, España, Editorial Blume.

BREA, J., (2010), “Las tres eras de la imagen - Imagen-materia, film, e-imagen”, Móstoles, Madrid, Cofás, S.A.

CORBAL, JA. (2018) “Estética en Videojuegos” Madrid, España, Editorial Ra-Ma.

DEITEL H., Deitel, P., (2008) “Cómo programar en C++: Introducción a la programación de videojuegos y las bibliotecas Boost” 6ª Edición, Prentice Hall Ediciones

DORAN, J., (2021), “Unity 2021 Shaders and Effects Cookbook: Over 50 recipes to help you transform your game into a visually stunning masterpiece”, 4ª Edición revisada, Birmingham, United Kingdom, Packt Publishing Limited.

ESTEBAN, FJ. (2021). “El lienzo inmersivo. Diseño y desarrollo de una experiencia pictórica en una realidad virtual”, Valencia, Universitat Politècnica de València. https://riunet.upv.es/handle/10251/170222

FERNANDEZ, D. (2019), “Graphics Shaders For Scientific Visualization”, Madrid, Universidad complutense de Madrid.

FERRONE, H. (2020) “Learning C# by Developing Games with Unity 2020: An enjoyable and intuitive approach to getting started with C# programming and Unity, 5th Edition” Birmingham, Reino Unido, Packt Publishing Limited.

GALÁN, J., FELIP, F. (2020) “Realidad Virtual - Construyendo el presente del arte, el diseño, la arquitectura y el entretenimiento” Valencia, España, Tirant Humanidades.

GALÁN, M. (2021). “Producción de un escenario óptimo para videojuegos 3D” Valencia, España, Universitat Politècnica de València. https://riunet.upv.es/handle/10251/175507

GONZÁLEZ, B. CLARO, A., (2015) “El potencial educativo de la fotografía, Cuaderno Pedagógico” Santiago, Chile, Salesianos Impresores.

GORAL, C., TORRANCE, K., GREENBERG, D., BATTAILE, B., (1984), “Modeling the Interaction of Light Between Diffuse Surfaces” Ithaca, New York, USA, Cornell University. SIGGRAPG´84.

GRAU, O. (2004) “Virtual Art: From illusion to Immersion” Massachusetts, USA, Mit Press.

HAINES, E., HOFFMAN, N.; AKENINE-MÖLLER, T. (2018) "Real-Time Rendering" 4ª Edición, Boca Ratón, Florida, USA, CRC Press.

HERGAARDEN, M., (2011) “Graphics Shaders”, Amsterdam. Países Bajos. https://files.m2h.nl/LiteraturestudyShaders.pdf

HOFFMAN, N., (2015) “Physics and Math of Shading”, SIGGRAPH 2015.

HONG-YI Pai (2019), “Texture designs and workflows for physically based rendering using procedural texture generation” IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE).

IBORRA, AJ. (1993) “Integración de imágenes reales en entornos virtuales”, [Tesis de Doctorado, Universidad Politécnica de Madrid].

IGLESIAS, S. (2021) “El renderizado en tiempo real: integración y posibilidades en el ámbito académico” La Coruña, España, Universidade da Coruña. Escola Técnica Superior de Arquitectura.

KAJIYA, JT (1985), “Anisotropic reflection models” SIGGRAPH ’85.

KOK, B. (2021) “Beginning Unity Editor Scripting: Create and Publish Your Game Tools” Nueva York, USA, Apress.

KUMAR, A. (2020) “Beginning PBR Texturing - Learn Physically Based Rendering with Allegorithmic’s Substance Painter” Nueva York, USA, Apress.

LANZA, D. (2021) “Arte Digital. Historia, evolución y tendencias en el arte de los nuevos medios” Madrid, España, Universidad Complutense de Madrid.

LÉVY, P. (1998) “¿Qué es lo virtual?” Barcelona, España, Ediciones Paidós Ibérica.

LIDON, M. (2018) “Unity 3D” Barcelona, España, Marcombo Ediciones Técnicas.

LIESER, W (2010) “Arte digital - nuevos caminos en el arte, Königswinter, Alemania, Tandem Verlag GmbH / H. F. Ullmann.

LUIS-TELLO, JJ. (2022) “Interaction According to Immersion in Virtual Environments: Graphic Development and PBRS in Environments with Real-Time Rendering and Virtual Reality”, Interacción '22: Proceedings of the XXII International Conference on Human Computer Interaction. Teruel, Spain.

LUIS-TELLO, JJ. (2019) “Nueva rama profesional: Artistas 3D” en C. FORADADA, P. IRALA-HORTAL, Arte, patrimonio y tecnología en la era digital (pp. 227 - 229) IAACC Pablo Serrano.

LUIS-TELLO, JJ. (2021) ”Videojuegos y el cuerpo en movimiento: Más allá de la silla.” En J. BERNAT, R. ARNAL, Ponencias VIII Encuentro Internacional de Film de Danza y Música (pp. 47 - 52) L´Eixam Edicions.

MARTINEZ, DG., (2020) “Virtual Production y Performance Capture. Estudio de técnicas en tiempo real para la producción de efectos visuales en la industria audiovisual” Madrid, España, UCM. [Trabajo Fin de Máster]. https://eprints.ucm.es/id/eprint/63018/

MARTINEZ, J., NAVARRO, F., MARTINEZ, A. (2018) “Realidad Virtual y Realidad Aumentada” Madrid, España, Editorial Ra-Ma.

MAZOUKA, D., KRASNOPROSHIN, V. (2021), “Developmental Milestone of Graphics Technologies” Minsk, Bielorrusia, Belarusian State University.

MCCLANAHAN, C. (2010), “History and Evolution of GPU Architecture” Georgia, USA, Georgia Tech, College of computing.

McREYNOLDS, T., BLYTHE, D. (2005) “Advanced Graphics Programming Using OpenGL” San Francisco, California, USA, Elsevier.

MONIEM, M. (2015) “Unreal Engine Lighting And Rendering Essentials”, Birmingham, Reino Unido, Packt Publishing Limited.

NUÑEZ, F. (2022) "Custom Raytracing Material Library for Games" Zaragoza, España, Escuela de Arquitectura y Tecnología. Universidad de San Jorge. https://repositorio.usj.es/handle/123456789/860

PARK, S. y BAEK, N., (2021), “A Shader-Based Ray Tracing Engine”, Daegu, Korea.

PAUL, C., (2015), “Digital Art” 3ª Edición, Londres, Reino Unido, Thames and Hudson Ltd.

PEDDIE, J. (2019). “Ray Tracing: A tool for All” 1º Edición, Cham, Switzerland, Springer Nature Switzerland AG.

PEÑA, O., (2022), “Metaversos. La Gran Revolución Inmersiva”, Madrid, España, Anaya Multimedia.

PHARR, M., JAKOB, W., HUMPREYS, G. (2018) "Physically based Rendering. From Theory to implementation” 3ª Edición. Massachusetts, The MIT Press.

QUILES, A. (2019) “Nuevos formatos de Cine Digital - video interactivo, transmedia y Realidad Virtual” Barcelona, España, Redbook Ediciones

RODRIGUES, C. (2021), “Field-Configurable GPU”, Oporto, Portugal, Universidad de Oporto.

RIOJA, P., DE LA ROSA, R., (2021), “Curso de programación: videojuegos”, Madrid, España, Anaya Multimedia.

RUBIO, J. (2019) “Realidad extendida, interactividad y entornos 3D, Revisión de la literatura y proyecciones” Cartagena de Indias. Colombia. VII Congreso Internacional Ciudades Creativas.

RUSSELL, J., (2015) “Basic theory of physically-based rendering” Adv. autor. https://marmoset.co/posts/basic-theory-of-physically-based-rendering/

SHALAH, G., ALKHASAWNEH, M., KRASNOPROSHIN, V. y MAZOUKA, D. (2019) “Graphics pipeline evolution: Problems and solutions”, Science Publications.

STAMPE, D., ROEHL, B., EAGAN, J. (1994) “Realidad Virtual - Creaciones y desarrollos” Madrid, España, Anaya Multimedia.

TAVINOR, G. (2009) “The Art of videogames” Chichester, Reino Unidom, Wiley-Blackwell.

VAZ, B., MORLA, J. (2023) “El siglo de los videojuegos”, Barcelona, España, Arpa.

WANDS, B. (2007) “Art of the Digital Age”. Londres, Reino Unido, Thames and Hudson Ltd.

WOOP, S., SCHMITTLER, J., SLUSALLEK, P. (2005) “RPU: A programmable ray processing unit for real-time ray tracing” ACM Trans Graph 24 (3), p. 434-444.

YANG, Y., BARNES, C., FINKELSTEIN, A. (2022) “Learning from Shader Program Traces”, Computer Graphics Forum, 41(2), p. 41-56

6.2.- WEBGRAFÍA

Acute Art [Acute Art VR]. (8 de junio de 2017). Acute Art Virtual Reality - Jeff Koons, Marina Abramovic & Olafur Eliasson [Archivo de vídeo]. Recuperado de - https://www.youtube.com/watch?v=gVHiWqIw3J4

Adobe. *Documentation - substance 3D designer*. https://docs.substance3d.com/sddoc/substance-designer-102400008.html

Adobe. *The PBR Guide - Part 1*. https://substance3d.adobe.com/tutorials/courses/the-pbr-guide-part-1

Adobe. *The PBR Guide - Part 2*. https://substance3d.adobe.com/tutorials/courses/the-pbr-guide-part-2

ArtFutura. Realidad Virtual + Internet 3D. https://www.artfutura.org/v3/realidad-virtual-internet-3d/

Epic Games. *Unreal Engine - Materials, Controlling the appearance of surfaces in the world using shaders.* https://docs.unrealengine.com/4.26/en-US/RenderingAndGraphics/Materials/

Epic Games. *Unreal Engine - Nanite Virtualized GeometryOverview of Unreal Engine 5's virtualized geometry system to achieve pixel scale detail and high object counts.* https://docs.unrealengine.com/5.0/en-US/RenderingFeatures/Nanite/

Epic Games. *Unreal Engine - Post Process EffectsEffects applied to the whole rendered scene prior to being rendered.* https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/PostProcessEffects/

Filmsite. *Greatest Visual and Special Effects (F/X) - Milestones in Film.* https://www.filmsite.org/visualeffects.html

NVIDIA. *¿Qué es el Path Tracing?* https://la.blogs.nvidia.com/2022/05/10/que-es-el-path-tracing/

Programmerclick.com. *Comparación del lenguaje de coloración HLSL GLSL CG* https://programmerclick.com/article/291542992/

Real-time 3D Art Best Practices - Materials and shaders Guide https://developer.arm.com/documentation/102471/0100/Profile-and-compare-transparency-implementations

Referencias para programación y creación de scripts de Unity https://docs.unity3d.com/ScriptReference/

Repositorio de materiales PBR https://freepbr.com/

Repositorio de Texturas para PBR
https://www.textures.com/

The Rookies. *A Brief History of 3D Texturing in Video Games.*
https://discover.therookies.co/2019/05/09/a-brief-history-of-3d-texturing-in-video-games/

Unity Technologies. *Getting started with Shader Graph* .
https://docs.unity3d.com/Packages/com.unity.shadergraph@6.9//Getting-Started.html

Unity Technologies. *Substance PBR materials workflow for real-time rendering - Unite LA | AutoTech Summit.*
https://www.youtube.com/watch?v=aZE8YAVzXGk

Unity Technologies. *Unity Documentation - Materials*
https://docs.unity3d.com/Manual/Materials.html

Unity Technologies. *Unity Documentation - Post-processing and full-screen effects*.
https://docs.unity3d.com/Manual/PostProcessingOverview.html

Unity Technologies. *Unity Learn, Rendering and Shading*.
https://learn.unity.com/tutorial/rendering-and-shading

Wired shopper. *What is Ambient Occlusion?*
https://thewiredshopper.com/ambient-occlusion/?nonitro=1