



# Ejercicio del tema 2

# Mapas de normales

Entornos Virtuales

Pedro Manuel Gómez-Portillo López

gomezportillo@correo.ugr.es

10 de abril de 2018

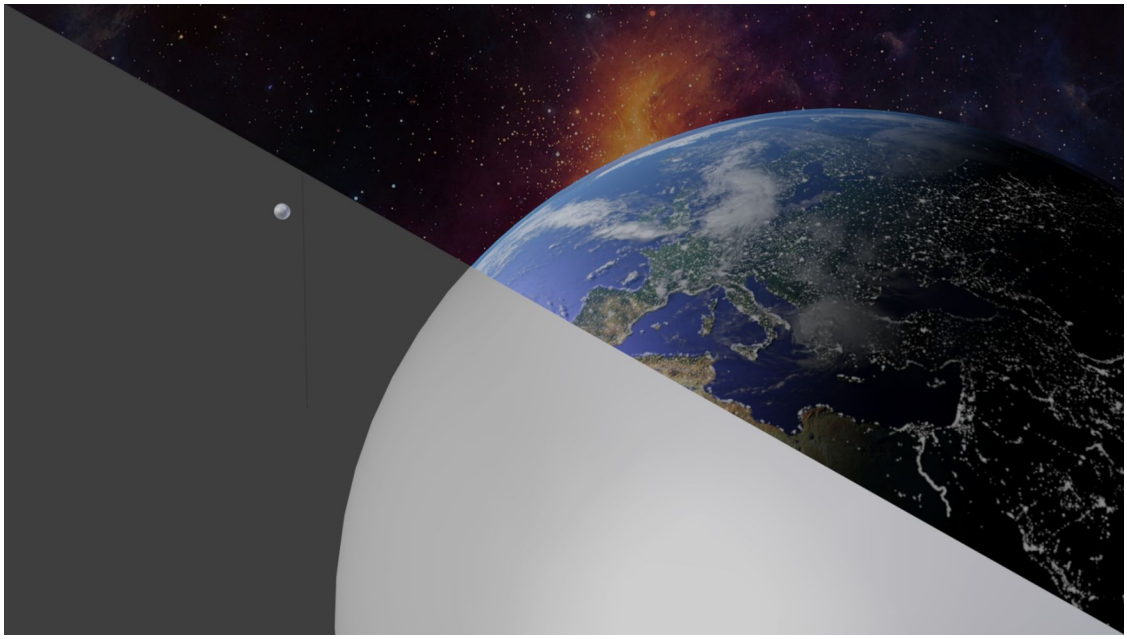
# Índice

# 1. Introducción

En esta práctica se explicará cómo se implementa en Blender alguno de los aspectos tratados en el tema 2 que no se está utilizando en la parte práctica. Concretamente, se explicará qué son los mapas de normales, cómo funcionan y cómo pueden utilizarse en Blender.

# 2. Práctica

Con el fin de aumentar su realismo, la mayoría de modelos 3D traen consigo varias capas de texturas sobrepuestas; texturas de color, mapas de desplazamiento o mapas especulares son solo algunos de estos ejemplos.



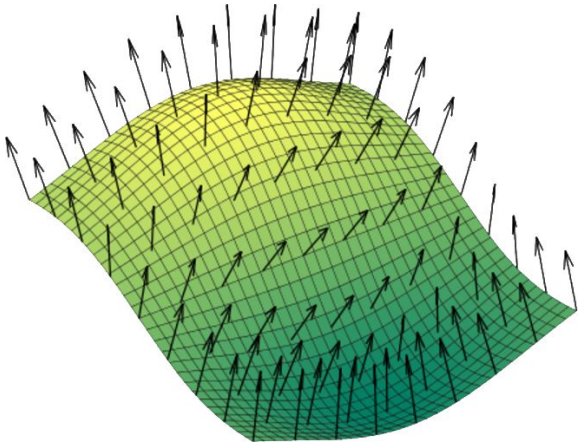
Comparativa de uno de los modelos de la práctica 4, con y sin todas las texturas

Uno de estos tipos de texturas son los mapas de normales, que son imágenes que permiten simular imperfecciones en la superficie de los modelos. Particularmente, son muy utilizados en el desarrollo de videojuegos, ya que debido a su naturaleza de sistemas en tiempo real es importante limitar el número de polígonos que se renderizan para aumentar su rendimiento; por ello, estos mapas son perfectos al permitir añadir detalles a modelos relativamente simples.

Comparativa entre tener o no un mapa de normales[1]

Para entender qué es un mapa de normales, primero debemos entender el concepto de un vector normal. En geometría, un vector normal es el vector perpendicular a la superficie de un plano[2].

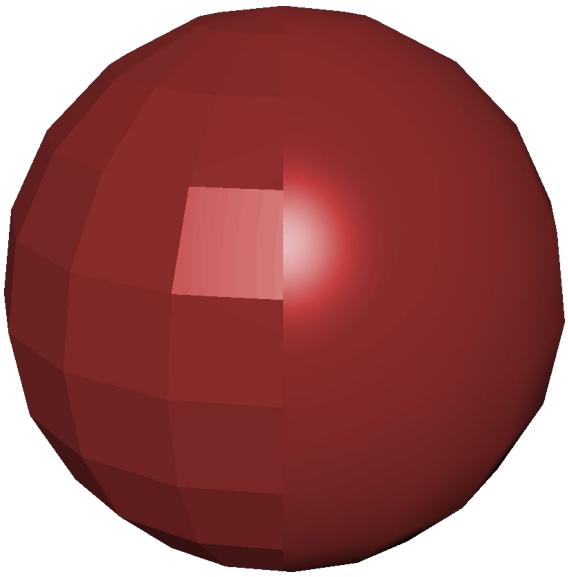
En un modelo 3D, todas sus caras tienen vectores normales, que son utilizados, entre otros, para calcular la iluminación de dicha cara a la hora del renderizado. En resumen, el ángulo entre la normal de cada cara y la fuente de luz puede usarse para calcular cómo de brillante será.



Ejemplo de las normales de las caras de un modelo 3D

1 Referencia - https://www.blenderguru.com/tutorials/basics-realistic-texturing
2 https://es.wikipedia.org/wiki/Vector_normal

Pero el método anterior para calcular la iluminación da lugar a caras planas y poco realistas (imagen inferior, izquierda), por lo que también se trabaja con las normales de los vértices. Así, al interpolar las normales de todos los vértices en cada polígono es posible crear una representación más suave de un objeto con un detalle geométrico relativamente bajo, lo que se conoce como *Phong shading*[3] (imagen inferior, derecha).



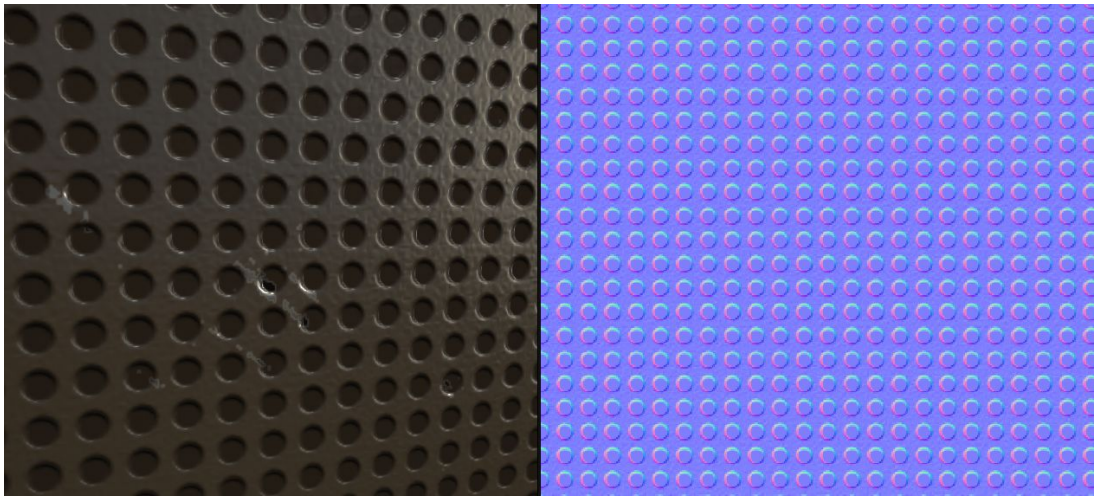
Comparativa entre ambos métodos de renderizado

Volviendo a los mapas de normales, son imágenes a color que permiten almacenar información tridimensional; cada pixel almacena un valor RGB tal que cada canal corresponde a un eje tridimensional distinto. Por convenio, el canal rojo corresponde al eje X, el canal verde al eje Y y el canal azul al eje Z.

Así, aunque los mapas de normales no permiten añadir geometría real a un modelo, sí permiten simular pequeños altibajos en las caras. De este modo se consigue silumar información de luces y sombras al modelo sin tener realmente que añadir geometría extra.

En la imagen inferior puede verse con cómo sobreponiendo un simple mapa de normales sobre un plano consigue simularse una rejilla.

3 https://en.wikipedia.org/wiki/Phong_shading

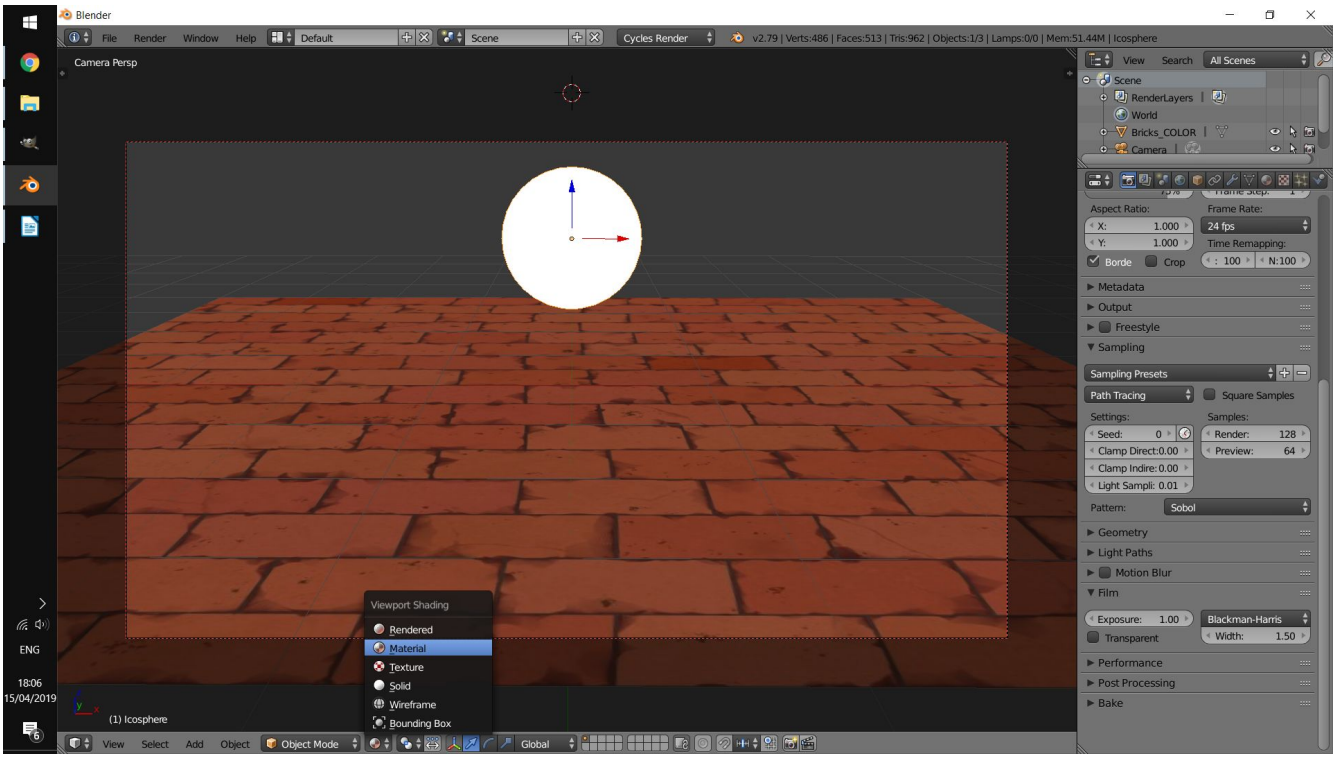


Fuente: https://blog.teamtreehouse.com/understanding-normal-maps

# 3. Ejemplo en Blender

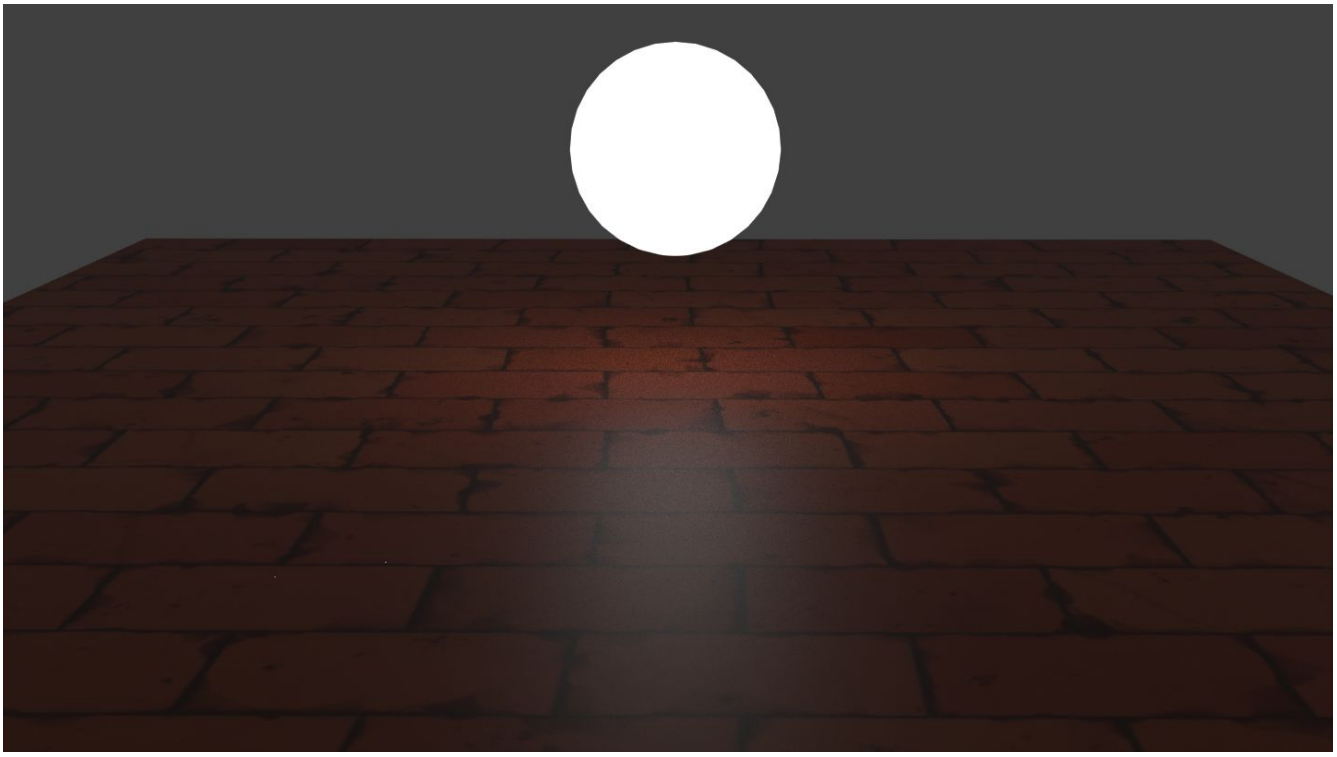
A continuación se presenta un ejemplo hecho con Blender para ver la diferencia entre usar o no un mapa de normales. Se usará el motor de renderizado Cycles, ya que es más gráfico ver qué está pasando con el uso de nodos.

Lo primero que haremos será descargar una textura con color y mapa de normales. Para ello, usaremos el sitio web *3DTextures,* en concreto la textura *Terracota Bricks*[4]*.* Tras ello, usaremos el addon *Import images as planes* para importar la textura de color. Una vez hecho, añadiremos una esfera UV que emita luz sobre ella.

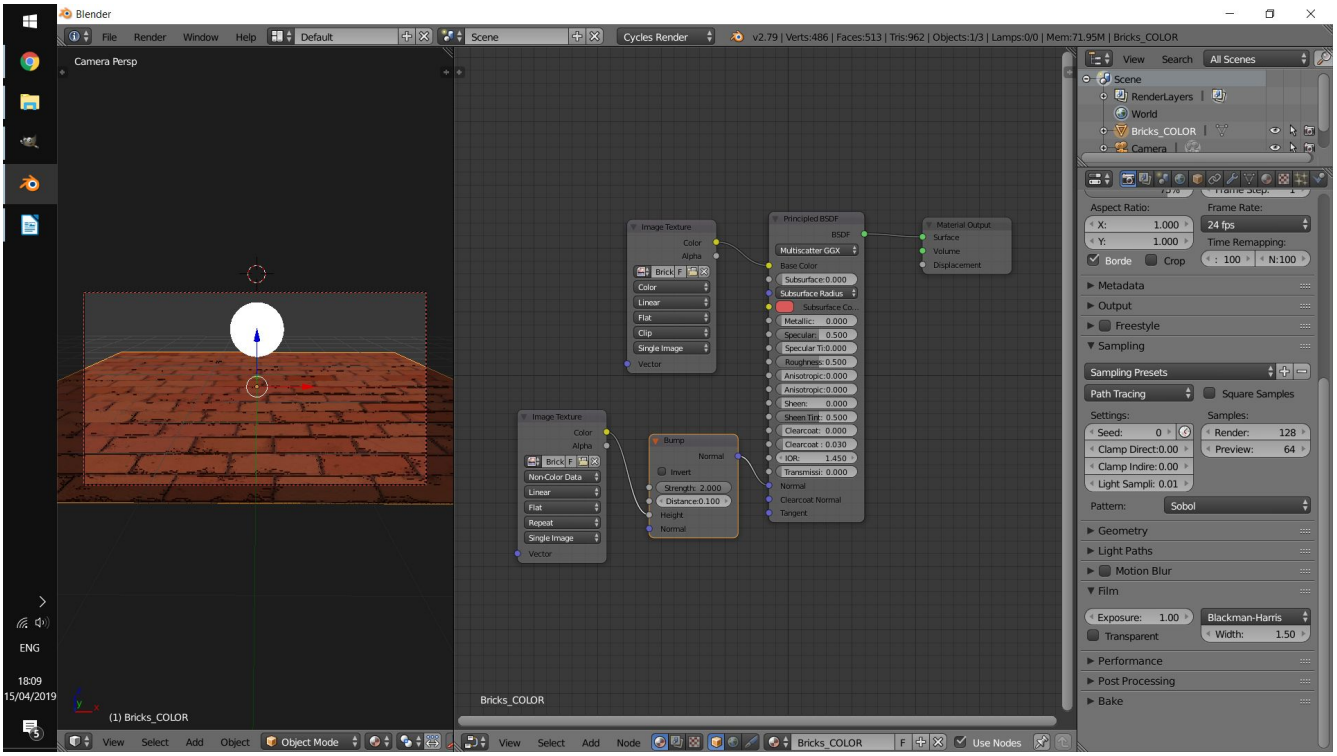



Ahora renderizaremos para ver el resultado.

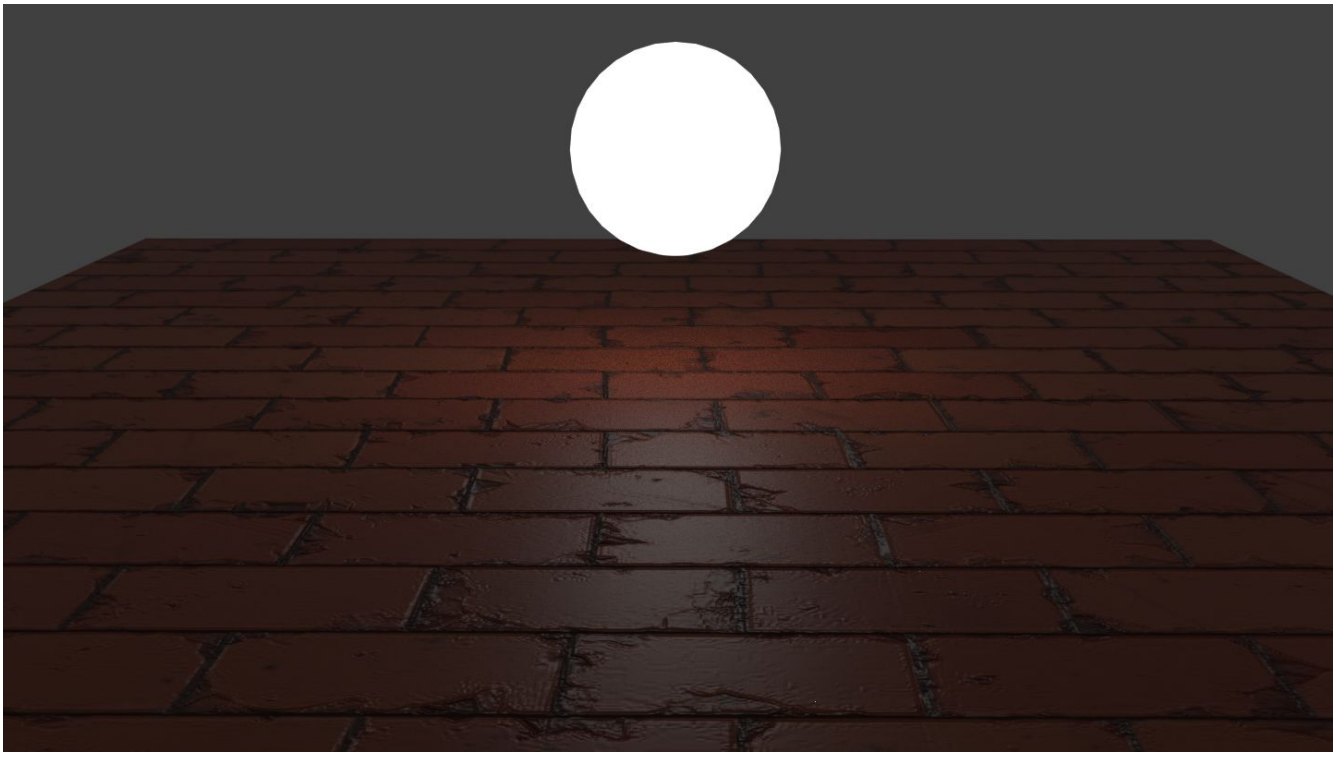
4 https://3dtextures.me/2019/02/19/terracotta-bricks-001/



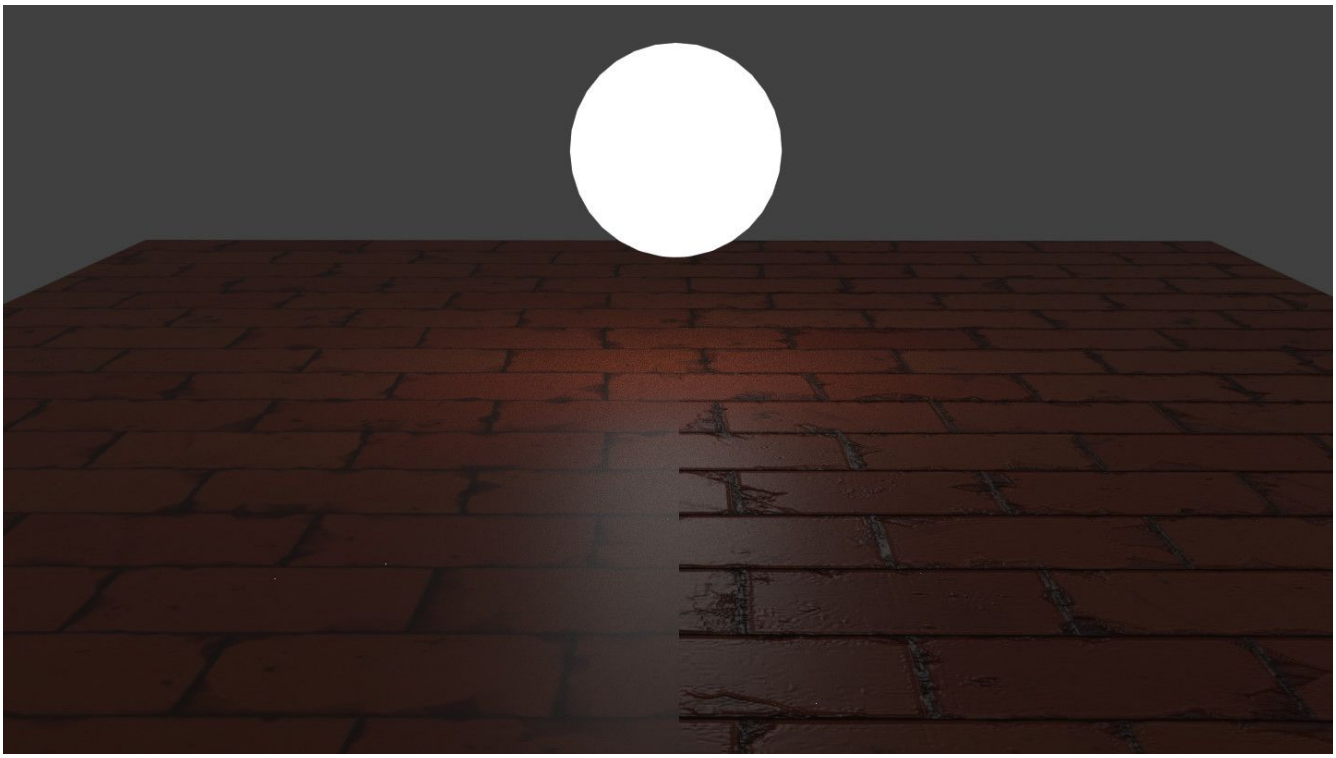
Ahora añadiremos el mapa de normales como una nueva textura, lo definimos como *Non-color data texture* y lo pasamos por un nodo *Bump* para que Blender lo reconozca como tal.




Y ahora renderizamos.



Como puede verse en la imagen inferior, la diferencia es muy notable.



Ambos modelos han sido renderizados con los mismos parámetros,

- Renderizado CUDA en una NVIDIA GEFORCE GTX
- Addon Auto-tile size activado
- Denoising desactivado
- Resolución 1920x1080 al 75% de tamaño
- Tamaño del sampling: 128

El resto de valores se han dejado por defecto.

Como era de esperar, los tiempos de renderizado obtenidos son prácticamente idénticos

| Mapa de texturas | Tiempo (s) |
|---|---|
| No | 41.8 |
| Sí | 42.4 |

# 5. Conclusiones

Como hemos visto, los mapas de normales son una herramienta muy potente que permite añadir detalles a modelos con baja poligonalización, lo que los hace perfectos para multitud de aplicaciones, especialmente el de los videojuegos, ya que apenas supone diferencia en tiempo de renderizado.

# 4. Webgrafía


- https://blog.teamtreehouse.com/understanding-normal-maps
- https://en.wikipedia.org/wiki/Normal_mapping
- https://cgcookie.com/articles/normal-vs-displacement-mapping-why-games-use-normals
- http://wiki.polycount.com/wiki/Normal_map
- https://www.youtube.com/watch?v=0r-cGjVKvGw
- https://www.pluralsight.com/blog/film-games/bump-normal-and-displacement-maps
- https://blenderartists.org/t/how-normal-maps-work/677099