

Práctica 2: Shader de Gaudí

Procesadores Gráficos,
Juegos y Realidad Virtual



© David Miraut



Universidad
Rey Juan Carlos

Máster Oficial
en Informática
Gráfica, Juegos y
Realidad Virtual
Escuela Técnica Superior de
Ingeniería Informática

<http://dac.etsii.urjc.es/rvmaster/>

"Las máquinas me sorprenden con mucha frecuencia".

-- Alan Mathison Turing

"All it takes is for the rendered image to look right."

-- Jim Blinn (inventor del bumpmapping)

Práctica 2: Shader de Gaudí

Objetivo de la práctica

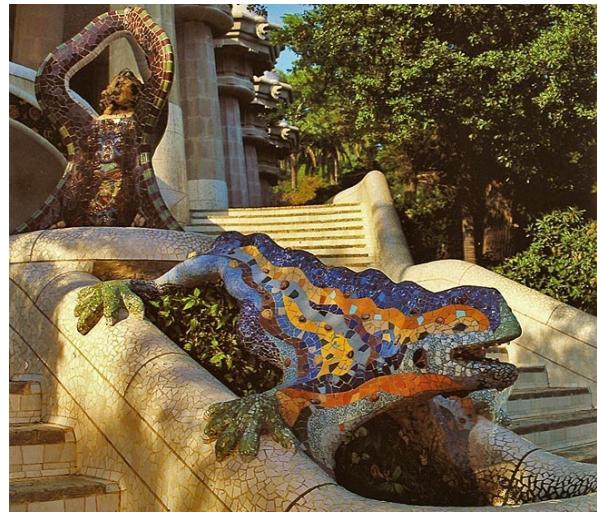
De nuevo, el objetivo de esta práctica consiste en profundizar en los conceptos de arquitectura que se han introducido en las asignaturas de Gráficos 3D y en la primera parte de Procesadores Gráficos y Aplicaciones en Tiempo Real, aplicando estos conocimientos y reforzándolos sobre la base práctica de la programación de procesadores gráficos de última generación para PC.

En esta ocasión, aprovechando el conocimiento adquirido en la práctica anterior (al explorar la sintaxis de GLSL), vamos a abordar una forma de hacer gráficos que no es directamente natural al cauce gráfico. En lugar de utilizar las distintas etapas del cauce gráfico en el orden clásico, volveremos a trabajar sobre una escena geométricamente muy simple (dos triángulos), y concentraremos nuestros esfuerzos en la programación de la etapa de sombreado.

Por un lado, nos familiarizaremos con un algoritmo extremadamente eficiente para la generación de mapas celulares, que se ha hecho muy popular en la comunidad gráfica, lo que nos ayudará a comenzar a entender las limitaciones que tenemos en el acceso a los datos en arquitecturas masivamente paralelas. Por otra parte, exploraremos la síntesis de imágenes con objetos 3D que no han sido descritos por listas de vértices, diseñaremos y desarrollaremos nuestros propios algoritmos para darles un aspecto más "gaudiano" mediante variantes del mapa celular.

Esta práctica va a tener un carácter artístico importante, que nos inspirará en el diseño de los algoritmos. El objetivo es que partiendo de cualquier tipo de geometría (procedural o explícita en forma de listas de triángulos), los objetos de la escena pasen a tener un aspecto que nos recuerde al tipo de mosaicos de vivos colores que recubren la superficie de esculturas tan emblemáticas como la Salamandra (o dragón de la escalinata) del Parque Güell.

No se va a hacer hincapié en el análisis y diseño en estas primeras tomas de contacto, dado lo pequeños y sencillos que van a ser los shaders, pero son dos aspectos muy a tener en cuenta en cualquier desarrollo software, y en especial en *shaders* en los que la arquitectura puede condicionar fuertemente la forma de enfrentarse al problema a solucionar¹.

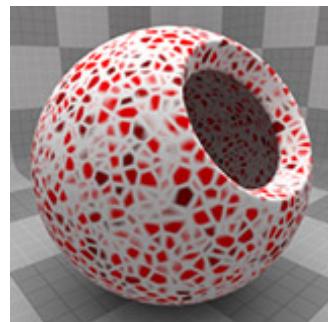


¹ Por supuesto, los alumnos son libres de hacerlo y serán bien valoradas este tipo de consideraciones en las memorias de las prácticas, pero en este momento es totalmente opcional dado el carácter inicial de esta práctica.

Planteamiento

La presente práctica consta de tres partes obligatorias, con elementos opcionales (sólo para aquellos alumnos que hayan escogido esta asignatura como completa).

La primera consiste en la lectura de un artículo y un capítulo de un libro, cuya explicación es fundamental para que podáis implementar el mapa celular. Aunque su generación se ha explicado durante la asignatura de Rendering Avanzado, estos dos textos explican en mayor detalle el fundamento matemática de las bases de ruido que vamos a aplicar, así como un conjunto de trucos que reducirán drásticamente las necesidades de comunicación de nuestros procesadores para generar de forma paralela el mapa celular. Ambos textos guardan fuertes parecidos (están escritos por la misma persona),



En la segunda parte exploraremos de forma práctica la implementación eficiente de mapas celulares y sus posibles variantes. El resultado de esta parte puede ser utilizado en la primera práctica de Rendering Avanzado como la textura que cubre las arterias y venas del recorrido del flujo sanguíneo, para dar una aspecto más realista a la animación. Dado que se os va a pedir que probéis combinaciones lineales y no lineales de mapas celulares, es posible que obtengáis ideas que os sirvan también para la fase de "mutación" en dicha práctica.

Finalmente, en la tercera parte introduciremos algunos conceptos de traza de rayos para generar escenas sencillas en cuyos objetos se han de aplicar los mapas celulares, no sólo como información de color difuso, sino como mapas de normales o desplazamiento que doten de un aspecto "gaudiano" a las superficies. El relieve en este caso es fundamental, ya que los mosaicos originales están hechos de vidrio y porcelana que sobresale de la superficie original. El algoritmo para dar "rugosidad" a las superficies tienen que diseñarlo los alumnos a partir de las propiedades del mapa celular estudiado en el artículo y el capítulo de la primera parte. son posibles muchas estrategias, por lo que se espera que cada grupo aporte una perspectiva original.

La siguiente práctica, que sólo han de realizar aquellos alumnos que hayan escogido Procesadores Gráficos como asignatura completa, consiste en utilizar shaders de geometría o teselación (a su elección) para dotar a una malla cerrada arbitraria de un aspecto semejante a las superficies procesadas en la tercera parte de esta práctica. La diferencia fundamental es que en la próxima práctica vamos a utilizar el cauce gráfico convencional.

IMPORTANTE:

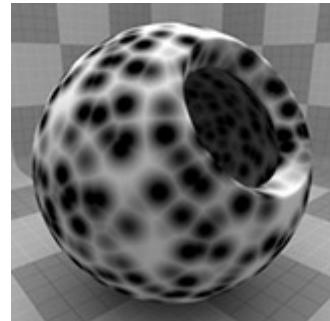
- En esta práctica habrá que entregar una memoria escrita en la que se respondan claramente a las preguntas y cuestiones propuestas en cada apartado.
- La entrega se realizará a través del Campus Virtual
- Es necesario incluir el código fuente o, al menos, indicar las modificaciones realizadas sobre el código existente.
- Es necesario incluir también una captura de pantalla (y un enlace a un video en YouTube en el que se visualice la animación en el caso de que las imágenes generadas cambien con el tiempo).
- Será absolutamente **obligatorio** comentar los resultados obtenidos.

Lecturas y tutoriales previos

Con el fin de adecuar la asignatura al nuevo espacio de europeo de enseñanza superior, y dado el reducido tiempo presencial que tenemos en la asignatura, parte de la explicación relativa a los detalles formales de la generación de los mapas celulares, las bases de ruido y los trucos para minimizar la comunicación entre procesadores, se hará fuera del horario de clase por parte del alumno, mediante la lectura de dos textos.

Así, la primera parte de la práctica consiste en la lectura del artículo titulado "A cellular texture basis function" de Steven Worley y el capítulo 4 "Cellular texturing" del libro "Texturing and Modelling: A procedural approach" de David S. Ebert. Ambos textos pueden encontrarse en la página web de la asignatura, dado que no tenemos suficientes ejemplares en la biblioteca. Os pedirá una clave para poder abrir el segundo documento: "gaudi" (sin tilde ni comillas).

El tiempo de clase lo aprovecharemos para concentrarnos en resolver las posibles dudas que hayan surgido tras la lectura y el "cacharreo" al realizar los pasos del tutorial.



Estos dos textos pueden ser objeto de preguntas en el examen, además es muy conveniente haberlos leído de cara a poder hacer con soltura la práctica, ya que en ellos se explican trucos muy interesantes para la generación procedural de mapas celulares en tiempo real.

Es importante destacar que el código adjunto en el capítulo 4 del libro de David S. Ebert no se adecúa a las restricciones arquitecturales de una GPU moderna por lo que, si bien podéis estudiarlo, es preferible que os centréis en la explicación teórica para deducir cómo construiríais el algoritmo. Tratar de "traducir" el código del capítulo sin más a un shader no es una estrategia que os facilite el trabajo en las siguientes etapas.

Normativa del laboratorio

- El criterio más importante es la funcionalidad: un programa que funciona siempre tiene más posibilidades de llevarse una buena puntuación; no se valorarán aquellos programas que no funcionen. Una práctica o proyecto modesto será evaluada mucho más favorablemente que un "proyecto" ambicioso que sólo da *core-dumps*. Los siguientes criterios que se tendrán en cuenta (y que hay que cuidar al realizar las prácticas) son:

- La manera de resolver el problema con el programa
- Estructuras de datos y diseño de los algoritmos
- Claridad y documentación en el código
- Eficiencia y elegancia en la implementación.
- ¡Por favor, no hagáis trampas! Se procura alentar el diálogo y el trabajo en equipo, pero por favor trabajad de forma independiente (a menos que el trabajo sea en grupos). Trabajos muy similares serán considerados como copias, a menos que la naturaleza lo pedido sea tan restrictiva que justifique las similitudes. Y una copia implica el suspeso automático. Simplemente piénsalo de esta manera: hacer trampas dificulta el aprendizaje y la diversión de conseguir hacerlo. Es vuestra responsabilidad proteger vuestro trabajo y aseguraros que no se convierte en el de otro.

- Si se utiliza (o mejora) código fuente u otro material obtenido a través de internet, la biblioteca... debe darse el crédito a los autores y pedir permiso de ser necesario (si tiene una licencia restrictiva). Tomar código de un libro o de internet sin tener en cuenta estas consideraciones será considerado copia.
- Está terminantemente prohibido la práctica de técnicas de *overclocking* en las tarjetas gráficas del laboratorio, así como desbloquear los procesadores de los chips gráficos. Este tipo de acciones pueden dañar físicamente el equipo del laboratorio y los alumnos responsables serán amonestados severamente.
- Las prácticas (código y memoria explicativa) deberán entregarse en los plazos indicados con las herramientas del portal de la asignatura en el Campus Virtual. La recepción de los trabajos a través del Campus Virtual restringe el periodo en el que se pueden enviar. Por favor, organizaros bien para evitar retrasos.

Recomendaciones

En principio la práctica se puede hacer en parejas, si alguien no encuentra pareja o por problemas de horario tiene complicado el poder equilibrar el trabajo también es posible hacerla de forma individual, esta práctica es sencilla y no debería suponer una dificultad añadida al no tener compañero/a para realizarla a aquellos que hayáis cursado carreras técnicas.

Índice del enunciado de la práctica

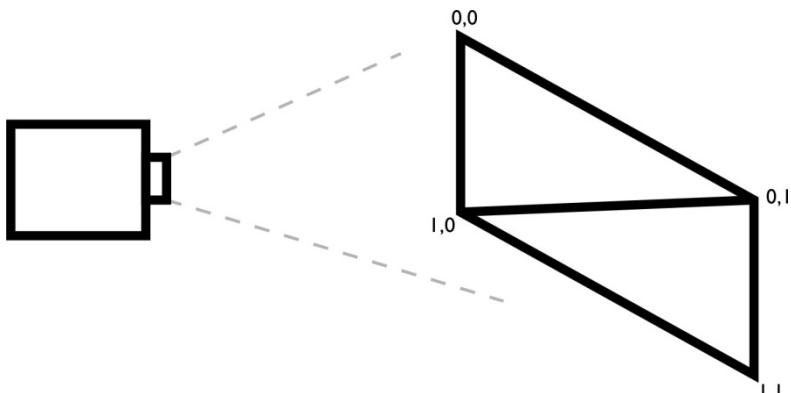
Objetivo de la práctica.....	1
Planteamiento	2
Lecturas y tutoriales previos	3
Normativa del laboratorio.....	3
Recomendaciones	4
Índice del enunciado de la práctica.....	4
1. Escena básica	5
2. Parte 1: Mapa celular simplificado y posibles variantes	5
2.1 Extensiones para los alumnos que han escogido PGATR como la asignatura completa:	8
3. Parte 2: Mapa celular caótico.....	9
4. Parte 3: El shader de Gaudi	9
Ejemplo sin colorear las teselas.....	12
4.1 Extensiones para los alumnos que han escogido PGATR como la asignatura completa (elegir dos de entre las propuestas):.....	12
Forma y fecha de entrega.....	13
Nota aclaratoria.....	13

1. Escena básica

En esta práctica vamos a volver a utilizar una herramienta especialmente diseñada para la codificación de shaders de fragmentos en la etapa de sombreado: ShadorZ. Las restricciones de la escena nos obligarán a diseñar estrategias y algoritmos nuevos para completar las partes de la práctica.

La escena de partida es muy sencilla, una cámara que apunta a un par de triángulos colocados a modo de *quad*, que cubren perfectamente el campo de visión. Las coordenadas de textura varían de cero a uno entre el vértice superior izquierdo y las coordenadas correspondientes de cada uno de los tres vértices en los extremos del campo de visión, de modo que tenemos una correspondencia perfecta entre los píxeles del framebuffer y los fragmentos que cubren la superficie del *quad*.

Podemos considerar que los colores calculados para cada uno de esos fragmentos son los téxeles de la textura que lo recubre en el caso más simple de proyección, ya que la superficie es ortogonal al eje óptico de la cámara.



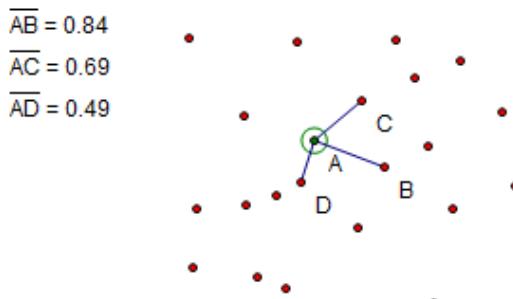
Este tipo de configuración se aprovecha en efectos gráficos en tiempo real de lo más variopinto (procesado de capas en postprocesado, filtrados, raymarching...). En el contexto de esta práctica es elegido por dos motivos:

1. Desde el punto de vista didáctico los shaders de fragmentos son los más sencillos de abordar, nos ayudan a entender la interfaz de memoria en modo gráfico y tienen la gran ventaja de tener una salida casi directa, lo que permite poder visualizar el resultado del *shader* con facilidad.
2. Además, su parecido con los shaders de cómputo nos permite experimentar con técnicas de Computational Graphics, traza de rayos y con la generación de texturas procedurales.

2. Parte 1: Mapa celular simplificado y posibles variantes

Tras haber leído las dos lecturas asociadas esta práctica, el alumno debe conocer los detalles de implementación del mapa celular en dos o más dimensiones (la explicación de Worley da por hecho que se trabaja en tres dimensiones, la reducción a dos dimensiones es trivial).

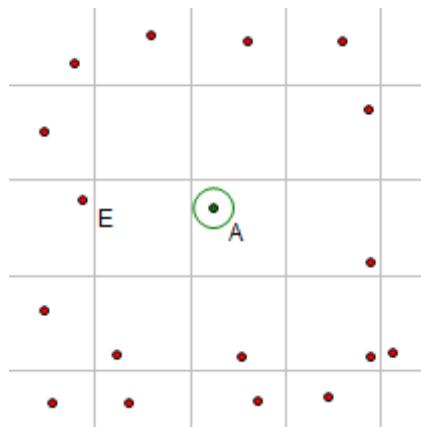
El color difuso aplicado a un pixel (o cualquier otra propiedad que nosotros deseemos describir con el mapa celular) depende de la distancia del centro de dicho pixel a los puntos característicos más cercanos. Estos puntos característicos se localizan aleatoriamente sobre el dominio que consideramos, uno se ve en la siguiente figura:



En este caso, el punto rodeado por el círculo verde (A) es la posición en la que se está evaluando la función base de ruido, y los puntos característicos son los coloreados en rojo. El punto característico D es el más cercano al punto de evaluación A. Según se ha indicado en el artículo de la bibliografía, $F_1(A) = 0.49$.

El cálculo del punto más cercano es un problema clásico en geometría computacional y destaca por su alto coste computacional. Aunque existen estrategias para el cálculo eficiente en GPU con un número finito de puntos, que aprovechan la jerarquía de memoria de la tarjeta gráfica para reducir la comunicación entre los procesadores, la propuesta de Steven Worley es especialmente interesante al eliminar los abultados requisitos de almacenamiento del algoritmo clásico a costa de incrementar la intensidad aritmética del algoritmo y reducir su aplicabilidad a casos concretos en los que un generador lineal congruente (LCG) proporcione el conjunto de puntos en un dominio que ha sido discretizado previamente.

La discretización en este caso tiene un impacto muy importante en la reducción del coste de este algoritmo. Ya que elimina la necesidad de comparar la posición de interés con todos los puntos posibles dentro del dominio. Si además se asegura que cada una de las celdas tiene al menos un punto, es suficiente con visitar la celda en la que se encuentra la posición de interés y el conjunto de celdas inmediatamente vecinas para localizar el punto más cercano (y potencialmente el segundo y el tercero más cercano también). Aunque con ello se está introduciendo un sesgo estadístico en la distribución de los puntos, a nivel visual apenas es perceptible, y dado que el objetivo es crear una textura con una cierta apariencia está justificada la aproximación.



En esta práctica no es necesario que el conjunto de puntos generados se ajuste a una distribución estadística concreta (aunque la más sencilla sea la distribución de Poisson). Es suficiente con que los puntos característicos estén localizados de manera aparentemente aleatoria e independientes unos de otros.

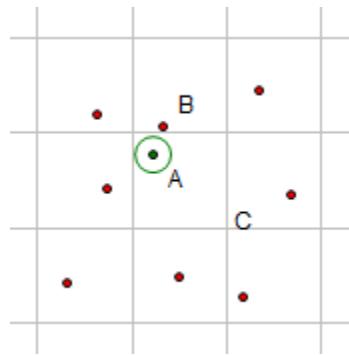
Para simplificar la generación del mapa celular, inicialmente sólo se va a pedir que generéis un punto por celda. De esta manera, no hace falta "calentar" el LCG hasta que los números generados tengan la distribución suficientemente aleatoria antes de que extraigamos uno que nos indique el índice de la tabla de Poisson o directamente el número de puntos a considerar en cada una de las celdas. Es más, es suficiente con utilizar una función *hash* que toma como entrada las coordenadas normalizadas del centro del píxel para obtener otras dos coordenadas lo suficientemente distorsionadas para asemejar un comportamiento lo suficientemente aleatorio para obtener los puntos característicos a partir de los que se calculan las funciones base de ruido.

Un ejemplo de función hash sencillo es el siguiente:

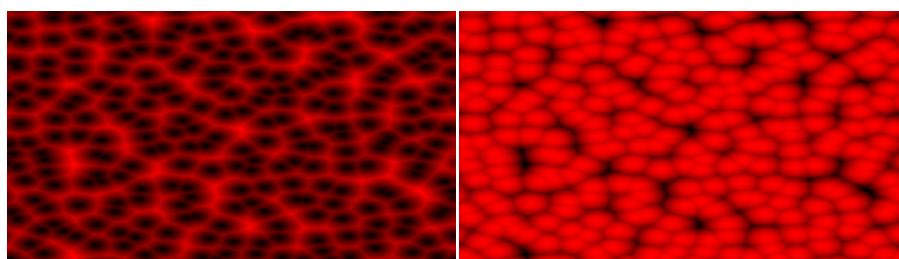
```
vec2 Hash2(vec2 p)
{
    float r = 523.0*sin(dot(p, vec2(53.3158, 43.6143)));
    return vec2(fract(15.32354 * r), fract(17.25865 * r));
}
```

La reducción a un único punto por celda implica que a cierta escala se está haciendo *jittering*. Por tanto, es posible que se aprecien artefactos a cierta distancia. En la implementación que se pide en la parte básica, no es necesario resolverlos.

Es importante destacar que la textura procedural se puede generar en una única pasada. Esto es posible gracias al mecanismo de generación de los puntos, resulta más barato consultar el valor de la función hash que almacenar los valores en memoria (por ejemplo en forma de textura) para luego leerlos. Además, este mecanismo evita los posibles conflictos debidos a la escritura y lectura en una estructura de datos por parte de un sistema masivamente paralelo, ya que sin dividir el kernel en varias subetapas no podría asegurarse que los datos se escriban en la textura antes de ser consultados (la división de los kernels tiene un coste que estudiaréis en los siguientes temas de la asignatura²).



El objetivo esta parte de la práctica consiste en que familiarices con el mapa celular y generéis una textura 2D que os pueda servir en la primera práctica de Rendering Avanzado.



Para ello es necesario que experimentéis con:

- distintas paletas de colores: bien con fórmulas o bien con tablas *lookup*
- variantes de la función Fn: combinaciones lineales y no lineales con las distancias de los puntos más cercanos (por ejemplo los 3 más cercanos)
- métricas: funciones de distancia como la Euclídea, Manhattan, Mahalanobis...

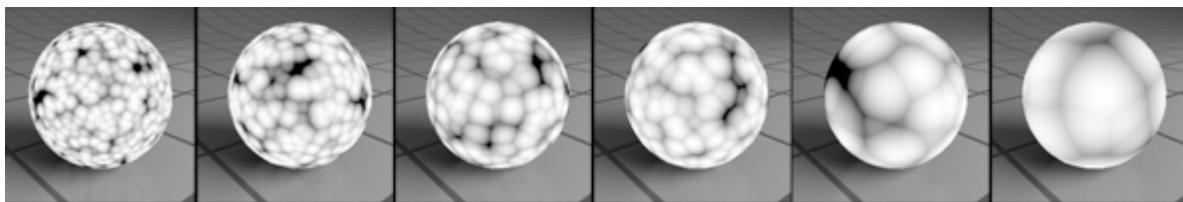
El entorno a utilizar será ShadorZ, al igual que en la práctica anterior. Donde podéis encontrar un esqueleto del código de esta parte con el nombre "practica2-1.frag".

En esta práctica hay tres restricciones importantes:

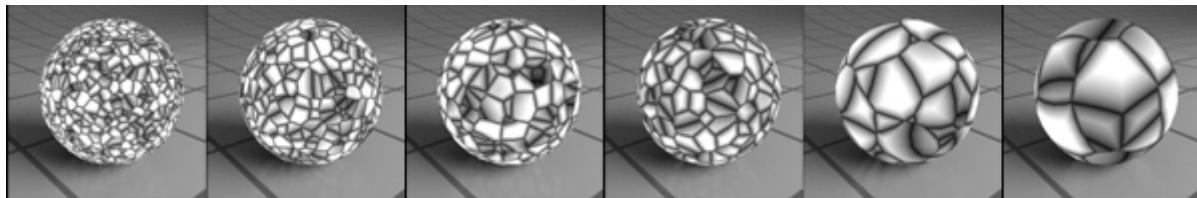
- **No es posible utilizar código de otros autores** (salvo en las extensiones)
- **En cada ejecución el patrón generado debe ser distinto.** Para ello os ayudar del uniform iGlobalTime.
- **No es posible utilizar texturas propias en esta práctica.**

Cuanto más experimentéis en esta parte, más sencillas os resultarán los siguientes bloques. En las siguientes figuras tenemos algunos ejemplos:

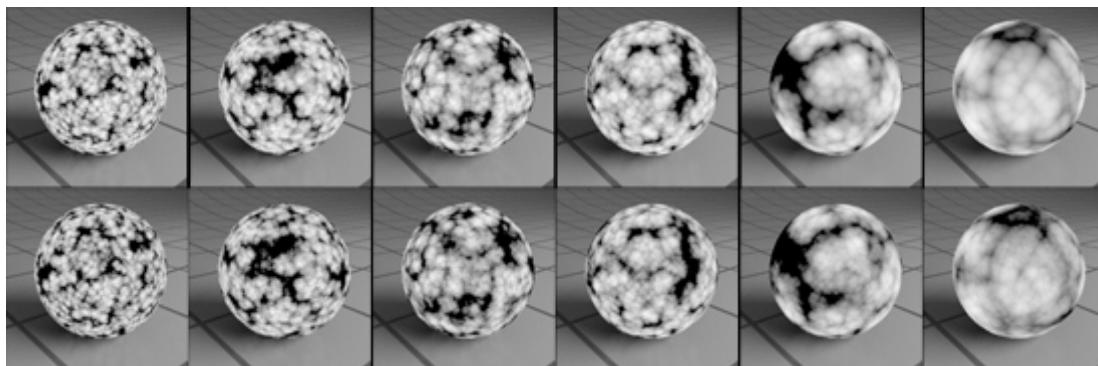
² En muchas ocasiones, es una estrategia interesante al reducir el uso de registros, pero nos obliga almacenar los datos en memoria de video para poder comunicar los distintos kernels y a esperar a que acabe el primer kernel antes de iniciar el segundo. En el cauce gráfico tiene un coste adicional, ya que tenemos que forzar una segunda pasada.



cambio en la densidad de las celdas



una fórmula distinta combinando distintos Fn, que recuerda a las teselas de los mosaicos de Gaudí.



variante fractal del mapa celular

2.1 Extensiones para los alumnos que han escogido PGATR como la asignatura completa:

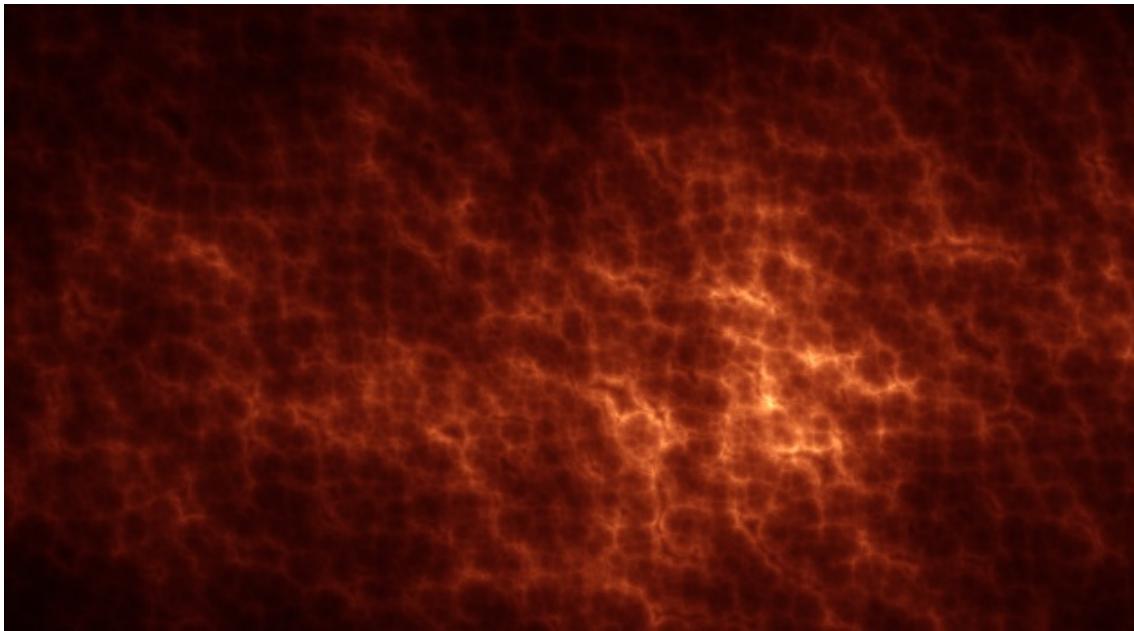
- Considerar un número variable de muestras en cada celda (mayor de uno, para evitar tener que hacer las búsquedas en un radio más amplio de celdas). Para ello es necesario diseñar y codificar un generador lineal congruente³ y dejarlo iterar suficientemente al inicio para que los números obtenidos no se pueda predecir a primera vista. La semilla con la que iniciamos el generador de números pseudo-aleatorios, debe estar relacionada con la posición central del píxel cuyos datos se han de "consultar". Además, sería interesante incorporar una tabla con una distribución como la que describe Steve Worley, cuyos valores se encuentran en el código al final del capítulo de la bibliografía.
- La textura debe ser "*seamless*" o *tileable*, esto es, se debe poder replicar sin que se note una transición brusca en los bordes. Una forma muy sencilla de lograrlo consiste en modificar la función *hash* para que actúe de forma periódica.

³ Es suficiente con una función del estilo de la descrita por Worley, pero podéis utilizar generadores de mayor calidad como los descritos en Efficient computational noise in GLSL de Ian McEwan [<http://ashimaresearch.com/wp-content/uploads/papers/McEwanEtAl2012.pdf>] y en el material suplementario del artículo [<http://www.itn.liu.se/~stegu/jgt2011/supplement.pdf>]

3. Parte 2: Mapa celular caótico

En la primera práctica de Rendering Avanzado, se pide a partir de cierto momento se produzca una mutación que dé lugar a un conjunto de efectos visuales interesantes.

Una posibilidad, es componer de formas originales las funciones base de ruido del mapa celular para conseguir una animación sencilla y efectista. También podéis jugar con los criterios de aplicación de los colores para darle un toque dramático.



En esta parte de la práctica, hay que entregar un shader basado en la idea del mapa celular experimentando más allá del algoritmo propuesto por Worley. Tenéis completa libertad creativa para ello.

Las únicas restricciones son las indicadas anteriormente:

- **No es posible utilizar código de otros autores** (salvo en las extensiones)
- **En cada ejecución el patrón generado debe ser distinto.** Para ello os podéis ayudar del uniform iGlobalTime.
- **No es posible utilizar texturas propias en esta práctica.**

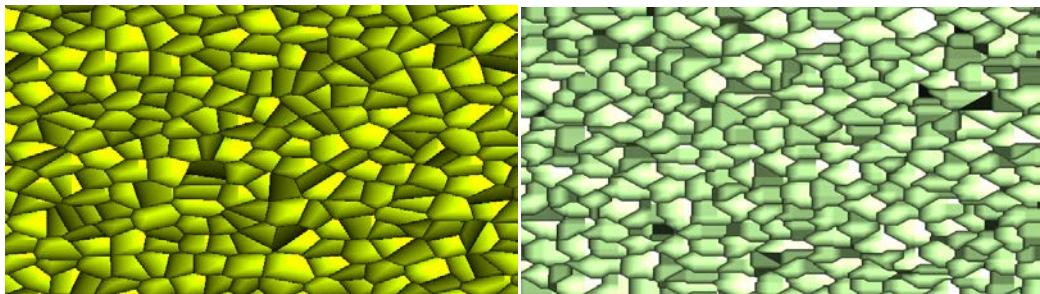
4. Parte 3: El shader de Gaudi

Dentro de las otras limitaciones que nos impone el entorno de trabajo que estamos utilizando para esta práctica, vamos experimentar con lo que se ha venido a llamar *Computational Graphics*, utilizando técnicas de síntesis de imagen 3D para las que el cauce gráfico no ha sido diseñado originalmente.



En la experimentación con las combinaciones lineales y no lineales de Fn, es posible que hayáis encontrado formas que recuerdan a las teselas de los mosaicos que cubren un buen número de las esculturas de Gaudí.

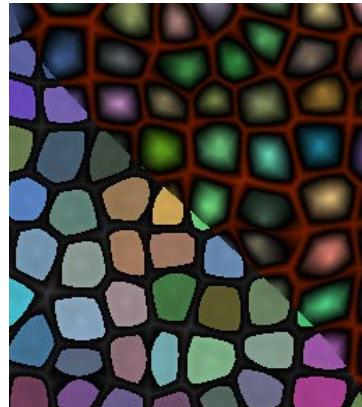
En el dibujo de la figura de la escalera se aprecian dos patrones en las teselas que pueden generarse de esta forma:



salamandra

base de la fuente

Además, cada una de las "escamas" o teselas se puede colorear de forma más o menos independiente, para crear patrones de color que las destaqueen de forma significativa. Para ello podemos escoger colorear con colores planos (constantes en el área interior de cada tesela) o con un fundido oscuro que destaque la parte interior:



En esta parte de la práctica vamos a trabajar con estos mapas, no sólo para colorear la superficie de objetos, sino para modificarla, de manera que se aprecien los bordes irregulares de las teselas y la zona de separación entre ellas.

La escena de partida se encuentra en el shader "practica2-3.frag" del entorno de trabajo. En ella se representa una vista del horizonte con el cielo casi al atardecer que se refleja en una esfera de aspecto metálico. El sol gira alrededor de la escena (o la cámara alrededor de la esfera) para destacar el reflejo.

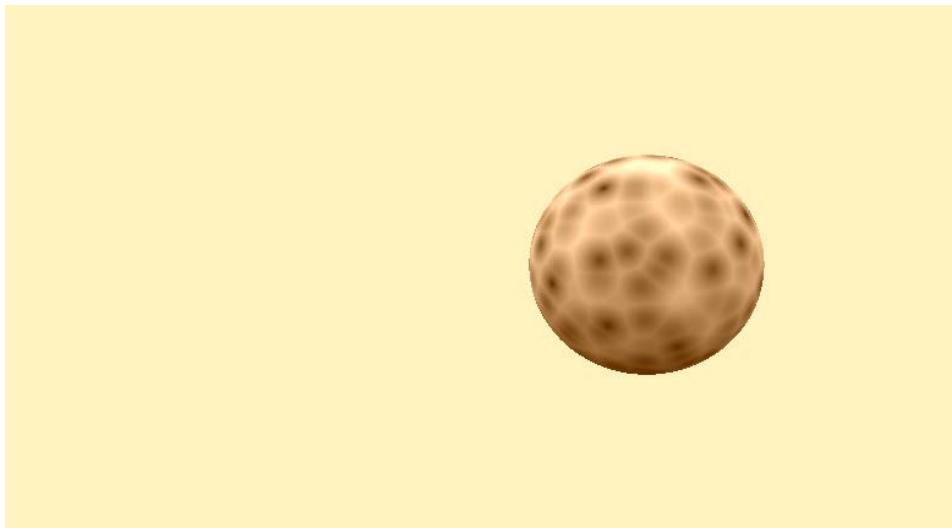


La escena se genera proceduralmente con una simplificación de la traza de rayos en la que se aplican varios trucos. Aunque todavía no se ha explicado esta técnica en Rendering Avanzado, la estrategia es muy sencilla en esta variante. En principio, se "lanza" un rayo desde la posición de cada uno de los píxeles y se comprueba si "interseca" con la esfera, de ser así se toma el color correspondiente en el mapa de entorno que tenemos con el vector cuyo ángulo viene dado por la reflexión calculada con vector incidente y la normal en el punto de intersección de la esfera (que se puede calcular como la diferencia entre el centro de la esfera y el punto de intersección), si no hay intersección simplemente se toma el color correspondiente al mapa de entorno.

Estudiad cuidadosamente el shader inicial. Hay simplificaciones demasiado drásticas en el modelo de cámara, la definición de la "esfera"... Es simplemente un código de partida, probablemente tendréis que modificarlo para hacer un test de intersección con la esfera más convencional.

El objetivo en esta parte de la práctica es doble:

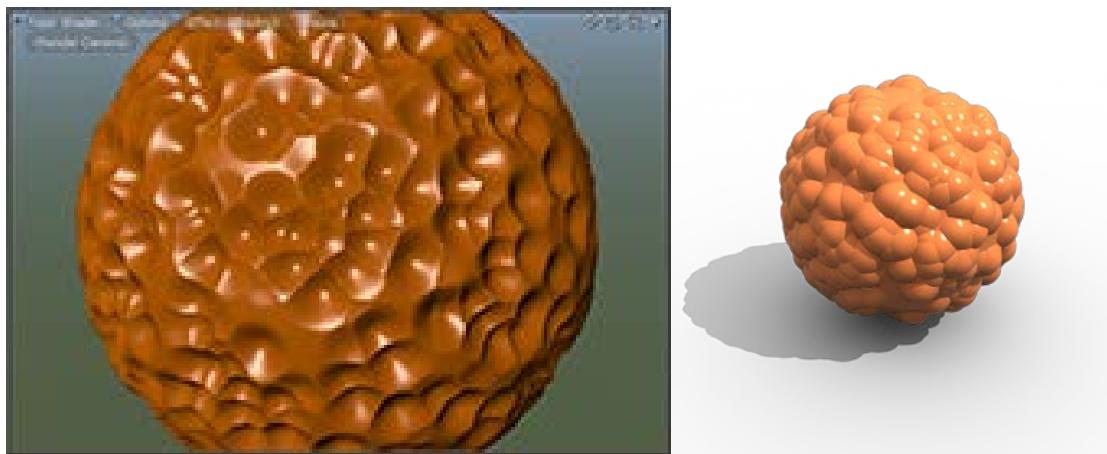
- Generar y aplicar un mapa celular sobre la superficie de la esfera para influir en el color difuso y la componente especular de la esfera. Cada tesela debe tener un color coherente.
- Utilizar dicho mapa como mapa de **desplazamiento** de modo que se aprecie la rugosidad de las teselas en la silueta y en la parte interior de la esfera.



Ejemplo de mapeado para color difuso (desactivando el mapa de entorno)

Tenéis que **diseñar nuevos algoritmos** para lograr los objetivos marcados.

No hay una única solución: Es posible trabajar con el mapa celular en 2 o 3 dimensiones para lograr el efecto deseado. Al no estar trabajando con gráficos rasterizados, no tenemos una descripción a nivel de primitivas triangulares de la esfera, sino que ésta se dibuja pixel a pixel en los puntos de intersección, por lo que no se pueden aplicar los algoritmos clásicos contados en clase. Es más, en un shader de fragmentos sólo se puede escribir en la posición del pixel asignada en el framebuffer; por lo que no es suficiente con calcular dónde debería estar la protuberancia, sino que hay que calcular qué grado de elevación corresponde al pixel en cuestión. Tened en cuenta que en el artículo y el capítulo de la bibliografía se dan pistas de cómo se puede aprovechar la información del mapa para calcular los vectores de un mapa de normales, no es necesario recurrir a diferencias finitas. Hay muchísimas posibilidades para abordar ambos objetivos.



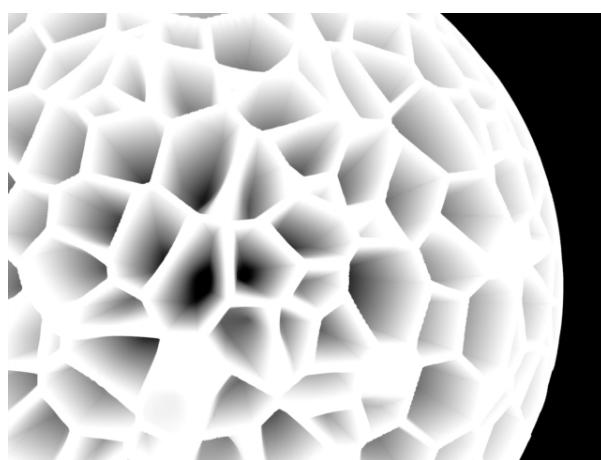
Ejemplo sin colorear las teselas

Las restricciones son las mismas en que en las otras dos partes de la práctica:

- **No es posible utilizar código de otros autores** (salvo en las extensiones)
- **En cada ejecución el patrón generado debe ser distinto.** Para ello os podéis ayudar del uniform iGlobalTime.
- **No es posible utilizar texturas propias en esta práctica.**

4.1 Extensiones para los alumnos que han escogido PGATR como la asignatura completa (elegir dos de entre las propuestas):

- Introducir más de un objeto con superficie reflectante y mapa de desplazamiento celular: lo que implica realizar múltiples rebotes.
- Modificar el mapa de entorno de forma creativa: generando otra textura procedural que sea accesible a través de un indexado a través de un vector, que pueda aplicarse en el fondo y en el reflejo de los objetos.
- Crear una textura de desplazamiento animada, de manera que el mosaico sobre la superficie de los objetos 3D cambie de forma suave con el tiempo.
- Introducir otras formas geométricas (tendréis que crear vuestros propios tests de intersección y/o modificar la forma de hacer el raycasting para añadir el mapa de desplazamiento).
- Modificar la forma en la se lanzan y proyectan los rayos para que parezca que la geometría se retuerce.



Ejemplo con el mapa de desplazamiento invertido y sin colorear las teselas

Forma y fecha de entrega

Se entregará la memoria en el formato de cualquier procesador de textos (ó PDF) debidamente identificada junto con el shader en GLSL en formato texto (y la textura que opcionalmente podéis añadir), tal y como los utilizaríais en el entorno de trabajo que se ha presentado en esta práctica.

La memoria debe estar acompañada por una captura de cada uno de los shaders (que puede volcarse desde el entorno de trabajo con la tecla F12) y un enlace a un pequeño video, si habéis realizado una animación.

Puede enviarse directamente mediante la aplicación correspondiente en el Campus Virtual dentro del plazo, e indicando “[PG Practica02]”. No olvidéis indicar vuestros nombres y apellidos en el cuerpo del mensaje.

La entrega de la práctica finalizará a las **15h del día 13 de Marzo**.

Un jurado independiente valorará el carácter artístico de vuestras creaciones, otorgando un premio a los tres shaders que más les gusten:

- 1er premio: 2 puntos extra
- 2o premio: 1'5 puntos extra
- 3r premio: 1 punto extra

Planificaros bien.

Nota aclaratoria

Todas las marcas y productos mencionados en este enunciado de prácticas están registradas por sus respectivas compañías, y su uso es de carácter descriptivo con fines docentes.