uc3m | Universidad Carlos III de Madrid

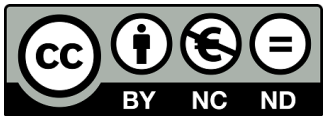
Grado en Ingeniería Informática

2021-2022

Apuntes

Informática Gráfica

Jorge Rodríguez Fraile[1]



[1]Universidad: 100405951@alumnos.uc3m.es | Personal: jrf1616@gmail.com

ÍNDICE GENERAL

1. INFORMACIÓN 3
1.1. Profesores 3
1.2. Sistema de evaluación 3
2. INTRODUCCIÓN A LA INFORMÁTICA GRÁFICA 5
2.1. Que es una Imagen 5
2.2. Áreas de Conocimiento 5
2.3. Aplicaciones 6
2.3.1. Diseño Asistido por Computador (CAD) 6
2.3.2. Gráficos de presentación 6
2.3.3. Creaciones artisticas 6
2.3.4. Entretenimiento 6
2.3.5. Simulación y entrenamiento 7
2.3.6. Visualización científica 7
2.4. Historia 7
2.4.1. Años 50 (Comienzos) 7
2.4.2. Años 60 7
2.4.3. Años 70 7
2.4.4. Anos 80 8
2.5. Campos de aplicación 8
2.6. Disciplinas relacionadas 9
2.7. Procesado de Imagen 2D 9
2.8. Procesado Gráfico 3D 10
2.8.1. Pipeline 3D 10
2.8.2. Framebuffer 11
2.8.3. Flujo de aplicación 12
2.8.4. Motor de Render 12
3. PERCEPCIÓN VISUAL 13
3.1. Ilusiones Ópticas 13

3.2. Percepción de la Luminosidad 13
3.3. Percepción de Tamaño y Profundidad 14
3.4. Características de la Visión 14
4. PROYECCIONES, OBJETOS 3D Y GSD 15
4.1. Proyecciones 15
4.1.1. Visión Estereográfica 16
4.1.2. Proyecciones no planas 16
4.2. Representación de Objetos 3D 17
4.2.1. Modelos de Superficie 18
4.2.2. Modelado Sólido 21
5. TRANSFORMACIONES GEOMÉTRICAS 23
5.1. Introducción 23
5.2. Aplicaciones en Informática Gráfica 23
5.3. Transformaciones lineales 23
5.4. Transformaciones 2D 24
5.4.1. Forma General 24
5.4.2. Transformación lineal 2D 24
5.4.3. Traslación 24
5.4.4. Escalado. 25
5.4.5. Rotación. 25
5.4.6. Cuidado con el orden de las transformaciones. 25
5.5. Sistema de Coordenadas Homogéneas 2D 26
5.5.1. Traslación en coordenadas homogéneas 26
5.5.2. Escalado en coordenadas homogéneas 27
5.5.3. Rotación en coordenadas homogéneas 27
5.5.4. Transformaciones inversas 27
5.5.5. Transformaciones generales afines 27
5.5.6. Estiramiento (Shear) 27
5.5.7. Composición de transformaciones 28
5.5.8. Operaciones respecto a un punto $P_1(x_1, y_1)$ 28

5.6. Sistema de Coordenadas Homogéneas 3D 28
5.6.1. Traslación 29
5.6.2. Escalado. 29
5.6.3. Rotación. 29
5.6.4. Rotación en coordenadas homogéneas 29
5.7. Rotación alrededor de cualquier eje 30
5.8. Rotación alrededor de cualquier eje y punto de origen 30
5.9. Matrices en PovRay 30
6. PRIMITIVAS 2D 31
6.1. Definiciones 31
6.2. Conversión de puntos 31
6.3. Conversión de rectas 31
6.3.1. Fuerza bruta 32
6.3.2. Algoritmo Analizador Diferencial Digital (DDA, Digital Diferenctial Analizer) 32
6.3.3. Algoritmo del punto medio de Bresenham (Bresenham midpoint algorithm) 33
7. FRACTALES 35
7.1. Definición 35
7.2. Caracterización 35
7.3. Aplicaciones 36
7.4. Procedimiento de Generación 36
7.5. Clasificación de Fractales 36
7.6. Dimensión Fractal 37
7.7. Fractales autosimilares. 38
7.8. Fractales autosimilares deterministas. 39
7.9. Fractales autoafines 39
7.10. Modelos Gramaticales 40
7.11. Fractales autocuadráticos 41
7.11.1. Conjunto de Julia 41
7.11.2. Conjunto de Mandelbrot 42

7.12. Fractales y caos 42
8. TEORÍA DEL COLOR 45
8.1. Introducción 45
8.2. La Luz 45
8.2.1. Historia 45
8.2.2. Radiación de cuerpo negro 46
8.3. Sistema visual humano 46
8.4. Caracterización del Color 47
8.5. Estandarización 48
8.6. Colores Primarios Estándar - Modelo CIE XYZ 48
8.7. CIE 1931: diagrama de cromaticidad (2D) 48
8.8. Diagrama de Cromaticidad 49
9. ILUMINACIÓN Y SOMBREADO 51
9.1. Rendering 51
9.2. Ray-tracing 51
9.2.1. Algoritmo 51
9.2.2. Ray-casting (primera iteración) 52
9.2.3. Reflexión 56
9.2.4. Refracción 57
9.2.5. Trazado de rayos recursivo (Whitted) 58
9.2.6. Modelos de rénder para iluminación global 58
9.3. Graphics pipeline 60
9.4. Aumentar el nivel de detalle de un objeto 62
9.5. Texturas 63
9.5.1. Cálculo de texel (texture filtering) 63

ÍNDICE DE FIGURAS

2.1 Disciplinas relacionadas 9
2.2 Pipeline Imagenes 2D 9

4.1 Proyección paralela 16
4.2 Proyección perspectiva 16

ÍNDICE DE TABLAS

7.1 Comparación Geometría Fractal y Euclídea 35

1. INFORMACIÓN

1.1. Profesores

Coordinador: Antonio Berlanga

Magistral: Yago Sáez, yago.saez@uc3m.es, 2.1.C.13

Magistral: José María Valls

Prácticas: David Quintana

1.2. Sistema de evaluación

- 50 % Teoría
  - 10 % Test 1. 4 temas
  - 10 % Test 2. 4 temas
  - 30 % Examen Final. Todos los contenidos teóricos de la asignatura
    Nota mínima: 3
- 50 % Práctica. La 1, 2 y 3 son por parejas y la última en grupo de 3
  - 12,5 % Práctica 1. Modelado 3D
  - 10 % Práctica 2. Fractales
  - 12,5 % Práctica 3. Paisajes y terreno
  - 15 % Práctica Final. Modelado 3D, Iluminación y sombreado y Animación

2. INTRODUCCIÓN A LA INFORMÁTICA GRÁFICA

2.1. Que es una Imagen

- Una vista en un monitor.
- Un fichero en un cámara.
- Números de una RAM

Definición

- Una distribución 2D de intensidad o color.
- Una función definida en un plano bidimensional.
- ! No hay mención a píxeles!

Necesidades

- Representar la imagen (codificarla en números)
- Mostrar la imagen (aplicar transformaciones, intensidades de corriente, cantidades de tinta).

2.2. Áreas de Conocimiento

Procesado de imágenes, se reciben solamente imágenes, se hacen modificaciones y análisis de imágenes.

- Análisis de escenas.
- Reconstrucción de modelos en 2D o 3D de las imágenes.
- Compresión de imágenes, mejora de nitidez, posterizado (convertir imágenes en dibujos o pósteres)...

Visión Artificial, dadas imágenes y datos sobre estas se obtiene información más completa.

- Construcción de sistemas que obtengan información automáticamente a partir de las imágenes.

- Clasificación de imágenes con base en conocimiento previo o información estadística.

Informática Gráfica, síntesis de objetos reales o imaginarios a partir de modelos basados en computación.

2.3. Aplicaciones

2.3.1. Diseño Asistido por Computador (CAD)

Herramientas gráficas para diseñar prototipos y evaluarlos antes de construirlos.

Áreas importantes: Diseño industrial, Arquitectura, Circuitos electrónicos y electrónicos.

Técnicas: Diseño basado en primitivas constructivas y Superficies curvas.

Posibilidades: Realidad virtual, Presentación realista, Análisis del diseño y Conexión con el sistema de fabricación.

2.3.2. Gráficos de presentación

Uso de los gráficos para producción de ilustraciones de soporte a informes y trabajos.

Áreas importantes: Economía, Estadística, Matemáticas y Administración y gestión.

Técnicas: Gráficos de líneas, Gráficos de barra, Gráficos de tarta y Superficies 3D.

2.3.3. Creaciones artisticas

Producción de imágenes con un fin artístico o comercial.

Áreas importantes: Diseño de logotipos, Bellas artes y Animaciones publicitarias.

Técnicas: Diseño vectorial, Soporte a la animación, Tratamiento de imagen y Rendering.

2.3.4. Entretenimiento

Producción de videos y videojuegos con efectos realistas.

Áreas importantes: Cine (películas y animación), Televisión (cortinillas, cabeceras, AR) y Videojuegos.

Técnicas: Animación, Visualización realista, Efectos especiales e Interactividad.

Se utilizan API y SDK: Truevision 3D, XNA, OGRE, IrrLicht, CryEngine 3, Unreal Engine y Unity.

2.3.5. Simulación y entrenamiento

Desarrollo de sistemas para simular tareas realistas y procesos.

Áreas importantes: Simulación de conducción, Simulación de procesos industriales, Entrenamiento y Educación.

Técnicas: Tiempo real, Interactividad y Realidad virtual.

2.3.6. Visualización científica

Visualización en dominios específicos y de grandes cantidades de datos.

Áreas importantes: Medicina (resonancias, tomografías), Ingeniería, Física (campos, dinámica de fuerza), Química (interacción molecular), Matemática y Topológico (terrenos y corrientes).

Técnicas: Codificación por color, Curvas de nivel, Visualización de volúmenes.

2.4. Historia

2.4.1. Años 50 (Comienzos)

Whirlwind y SAGE.

Se empezó utilizando la informática gráfica para detectar aviones o submarinos, con radares, en estos tiempos se utilizaban ordenadores del tamaño de ordenadores. En esta visualización se podía marcar puntos.

2.4.2. Años 60

- Sketchpad: el primer programa gráfico interactivo.
- Russell (MIT) desarrolla Spacewar en un PDP-1.
- El primer juego «Tennis for two» (Pong) se jugaba en un osciloscopio.
- Primer algoritmo de superficies ocultas (Catmull)
- Realismo mediante sombreado de superficies con color.

2.4.3. Años 70

- Suavizado de superficies poligonales (Gouraud)
- Comercialización del microprocesador.

- Fundación de Atari.
- Primeros intentos de informática gráfica en el cine.
- Diseños de la tetera (Newell, Utah, Univ.)
- Introducción de texturas y Z-buffer.
- Suavizado de superficies poligonales (Phong) que se utiliza en el 3D
- Fundación de Apple y Microsoft.
- Lucasfilm crea la división de gráficos por computador.
- Westworld: primera película en emplear gráficos por ordenador.
- La Guerra de las Galaxias.

2.4.4. Anos 80

- Popularización de SIGGRAPH como evento de referencia en el área.
- Publicación sobre el Raytracing (Bell Labs)
- Construcción del primer motor de rendering (REYES), precursor de Renderman (Carpenter)
- Ecuación de rendering (Kajiya)
- Película de TRON de Disney (Lisberger y Kushner)
- Venta masiva de terminales gráficas: IBM, Tektronix...
- GKS se constituye como estándar ISO y ANSI de construcción de librerías gráficas.
- IBM crea el ordenador personal (PC)
- Introducción de la radiosidad (Goral, Torrance, Cohen)
- Cornell Box: prueba para comprobar la adecuación de un sistema de renderizado a una fotografía real.

2.5. Campos de aplicación

- **Física, matemáticas y ciencias naturales:** Simulación, Modelado y Análisis numérico.
- **Ingeniería:** Sistemas, software y hardware, e Infraestructura e integración de entonos.
- **Arte y psicología:** Percepción y Diseño y composición.

2.6. Disciplinas relacionadas

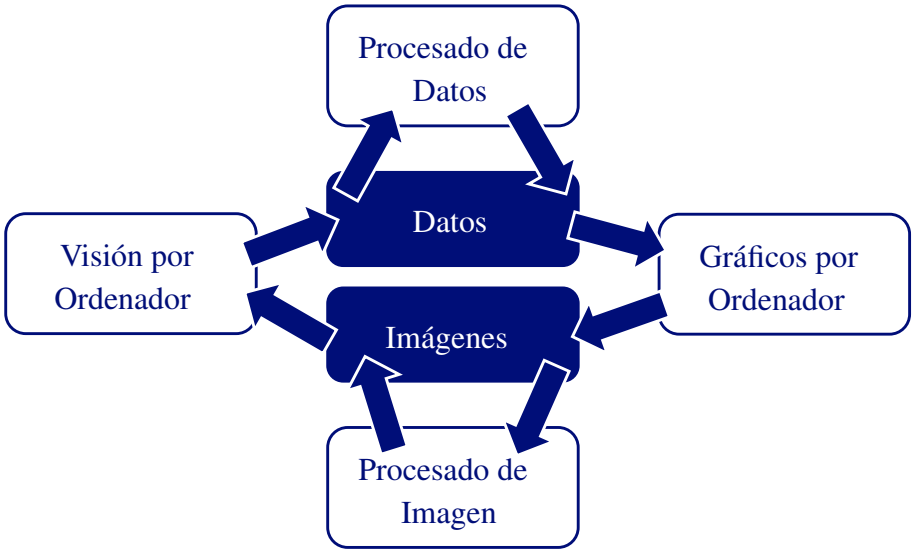


Fig. 2.1: Disciplinas relacionadas

2.7. Procesado de Imagen 2D

La secuencia de pasos por los que pasa el procesado de imagen en 2D puede representarse como un pipeline (no todos los pasos tienen que estar presentes).

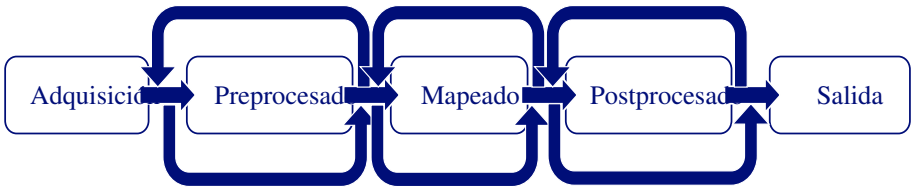


Fig. 2.2: Pipeline Imagenes 2D

1. **Adquisición:** Muchas técnicas, divisibles en dos categorías: Síntesis y Captura.
   - **Síntesis:** Imágenes creadas por un ordenador.
     Definición geométrica de los objetos (Renderman, Maya, PovRay).
     Imágenes «pintadas»(Photoshop, Illustrator, Fractal Painter).
   - **Captura:** Imágenes que provienen del mundo real.
     Capturadas o digitalizadas por hardware específico.
2. **Preprocesado:** Modificación de formas, tamaño y propiedades de color.
   Técnicas: Ajuste de color y niveles, Recorte, Escalado, Desenfoque, Mejora de bordes y Filtrado.
3. **Mapeado:** Varias imágenes se combinan con transformaciones.
   Transformaciones: Rotaciones, Escalados y Deformaciones.
   Composición: Efectos de transparencia / translucencia.

4. **Postprocesado:** Se utiliza para aplicar efectos globales a toda o parte de la imagen.
   Efectos artísticos: Posterizado, Envejecimiento, Desenfoque o Texturizado.
   Efectos técnicos: Mejora de contraste y Variaciones de color.
5. **Salida:** El dispositivo de salida puede afectar a su visualización.
   - El mapa de color de una impresora puede «falsear» o acentuar colores.
   - Se pueden mapear los colores entre la pantalla y la impresora.

   Dispositivos: Monitores, Impresoras, Discos y Mapas de texturas.

2.8. Procesado Gráfico 3D

Se encarga de generar las imágenes que se pueden ver en un dispositivo.

- Vista del sistema operativo.
- Imágenes (frames) de un videojuego.
- Pantalla de un teléfono.
- Imágenes de una película en el cine.

2.8.1. Pipeline 3D

Varios módulos van realizando diferentes tareas para la generación (render) de la imagen.

- **Transformación**
- **Iluminación**
- **Texturado**
- **Shaders:** Unidades de procesado que se compilan de forma independiente, modificando localmente las propiedades de los objetos de la escena.
  Vértices: Transformaciones de coordenadas y Colores y texturas.
  Geométricos: Generación de formas (primitivas).
  Fragmentos: Texturizado y sombreado a nivel de pixel.

Diagrama de flujo

- Entrada de los datos de la geometría de objetos y atributos de sus materiales.
- Definición de las condiciones de iluminación.
- Salida de las imágenes.

Historia

- Datos de entrada >Software de Render >Frame Buffer.
- Datos de entrada >Transformaciones e Iluminación >Transformaciones Raster >Procesado Pixel >Frame Buffer.
- Datos de entrada >Vertex Shading >Geometry Shading >Transformaciones Raster >Pixel Shading >Frame Buffer.

Entran vértices y salen píxeles.

2.8.2. Framebuffer

Memoria local para las imágenes que se mapean en un dispositivo. El hardware de video convierte el contenido en una señal para el dispositivo de salida.

Framebuffer sencillo:

- Dividido en planos.
- Cada plano representa 1 bit de valor del color del píxel.
- El plano tiene el tamaño del dispositivo (ejem. 1920x1080).
- Para RGB, profundidad de 24 bits.

Framebuffer complejo

- Doblar el número de planos (double buffering).
- Z-buffer para profundidad.
- Visión 3D, un conjunto de buffers para cada ojo.
- A-buffer para transparencia y sombras.

Z-buffer: es importante la precisión utilizada. Pueden aparecer problemas de «z-fighting», que dos superficies que ocupen el mismo espacio y se vea como una se funde con la otra.

El framebuffer necesita memorias muy rápidas

2.8.3. Flujo de aplicación

1. Software
    a) **Aplicación 3D:** CAD, Animaciones, Simulador o Juego 3D.
    b) **API:** Direct3D u OpenGL.
    c) **Drivers:** Preparación de datos, Mapeo hardware.
2. Hardware
    a) **Motor gráfico:** Transformaciones geométricas, Ilustración, Texturizado y Rasterización.
    b) **Memoria local:** Framebuffer, z-buffer y Stereo buffer.

2.8.4. Motor de Render

Software encargado de generar y visualizar información gráfica Abstrae los mecanismos de creación de gráficos.

Hay una gran variedad de motores gráficos en el mercado, de distintos tipos opensource, freeware y comerciales.

Técnicas: Sombreado, Mapeo de texturas, Bump-mapping, PBR, Efectos atmosféricos, Sombras, Reflexiones, Refracciones, Transparencias, Translucencias, Difracción, Iluminación global, Cáusticas, Profundidad de campo, Motion blur y NPR (cell-shading).

3. PERCEPCIÓN VISUAL

La percepción visual se divide en:

- **Percepción física:** Cuando los fotones llegan a nuestro nervio óptico y se transforman en impulsos electromagnéticos que estimulan el cerebro.
- **Procesado e interpretación:** Se transforman los estímulos/impulsos en imagen.

Por las características físicas, los humanos no podemos verlo todo.

El sistema de interpretación puede construir imágenes a partir de información incompleta.

El procesamiento compensa el Movimiento y Variaciones en la luminosidad, pero el contexto es fundamental. Las expectativas resuelven ambigüedades.

Movimiento: Mínimo de 16 fotogramas/segundo para percibirlo. Al recibir las imágenes tan rápido sentimos que se mueven.

- Cine mudo 16-18 Hz, Cine 24 Hz y TV 25-29 Hz.

3.1. Ilusiones Ópticas

El procesamiento visual puede crear ilusiones ópticas.

- Nombres de colores de diferente color.
- Tamaño de objetos iguales, pero en una posición diferente, escaleras, flechas o platos.
- Objetos que se mueven o cambian al mirarlos.

3.2. Percepción de la Luminosidad

Luminosidad: Cantidad de luz que recibimos (subjetiva) y depende de:

- **Luminancia:** Cantidad de luz emitida por un objeto.
- **Contraste:** Relación entre la luminancia de un objeto y la luminancia de su entorno.

3.3. Percepción de Tamaño y Profundidad

Ángulo de visión: Ángulo definido por los bordes del objeto percibido y el ojo. Los objetos familiares se perciben a tamaño constante.

Agudeza visual: Capacidad para percibir detalles. El límite está en un ángulo visual de 1/60 grados.

Pistas visuales: Estas ayudas permiten tener percepción del tamaño y profundidad.

- Tamaño y posición relativa de los objetos.
- Más/menos definidos parecen estar más cerca o más lejos.

Percepción de tamaño con el mismo ángulo visual.

Tamaños iguales percibimos como diferentes - diferente tamaño relativo.

Tamaño iguales percibimos como diferentes - diferente profundidad.

Tamaño diferentes percibidos como iguales - diferente profundidad.

3.4. Características de la Visión

Percepción del tamaño y la profundidad

- **Ángulo de visión:** Cantidad que un objeto ocupa el área visual. Permite determinar tamaño y distancia.
- **Agudeza visual:** Capacidad para percibir detalles.
- **Pistas:** Ayudas visuales que permiten tener percepción del tamaño y la profundidad.

Pistas visuales

- **Contraste, claridad y brillo:** Objetos más definidos parecen más cercanos.
- **Sombras:** Pistas adicionales de la posición relativa.
- **Texturas:** Menos definidas cuando más alejados del observador.

4. PROYECCIONES, OBJETOS 3D Y GSD

4.1. Proyecciones

Proyección: Transformación matemática que convierte un conjunto de puntos n-dimensional en un conjunto de puntos k-dimensional, siendo $k \leq n$. Reduce la dimensionalidad, normalmente 3D a 2D.

La proyección se define por unas **líneas de proyección** (proyectores) que, dirigidos hacia el **centro de proyección**, pasan a través del objeto e intersecan el **plano de proyección/-superficie de proyección** para formar la proyección del objeto en el mismo.

Proyecciones Geométricas Planas

- Paralela
  - Ortográfica
    - Elevaciones, las proyecciones ortográficas en tres planos.
    - Axonométrica, el objeto es rotado respecto a uno o más ejes del plano de proyección.
      - Isométrica
  - Oblicua
    - Cabinet, eje de profundidad (z) forma 45 grados con el eje x. Profundidad escalada (1/2, 2/3, etc.).
    - Caballera, eje de profundidad (z) forma 45 grados con el eje x. Profundidad real.
- perspectiva
  - 1 punto
  - 2 puntos
  - 3 puntos

Métodos de generar vistas de objetos:

- **Proyección paralela:** El centro de proyección en el infinito, los proyectores son paralelos hacia el centro.
  - Mantiene la forma y dimensión de los objetos.
  - Representación no realista de los objetos, se usan para el diseño.
  - Conservan las propiedades relativas.

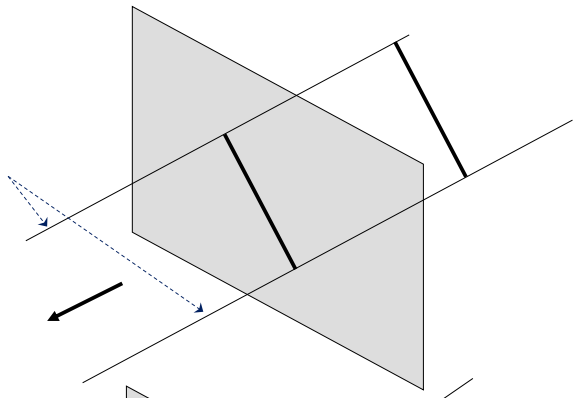


Fig. 4.1: P ón paralela

- **Proyección perspectiva:** royección cercano.
  - Altera la forma y dim os.
  - Representación realista de los imula la visión humana.
  - No conserva las proporciones

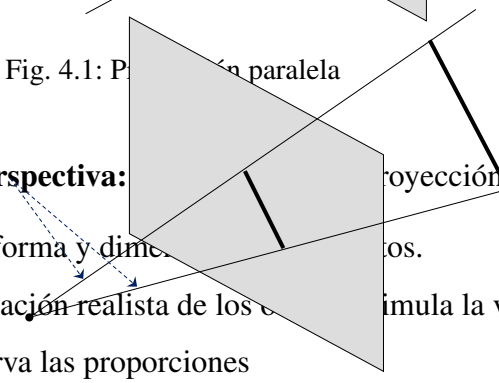


Fig. 4.2: Proyección perspectiva

4.1.1. Visión Estereográfica

Se obtiene realizando dos proyecciones en perspectiva, ambas imágenes se combinan para dar una tridimensional. Es el caso de los ojos.

4.1.2. Proyecciones no planas

- Proyección esférica: Imagen proyectada sobre una esfera.
- Proyección fisheye (ojo de pez, media esfera): Proyección esférica con un ángulo de 180 grados (estándar) o 360 grados (ultra). La proyección sobre un círculo.
- Proyección ultra wide angle: Fisheye en el que la proyección es un rectángulo.

- Proyección Omnimax: Fisheye con un ángulo de visión reducido en el eje vertical.
- Proyección cilíndrica: Imagen proyectada sobre un cilindro.
- Proyección panorámica: Tipo de proyección cilíndrica capaz de ver ángulos superiores a 180 grados.

4.2. Representación de Objetos 3D

Definir estructuras de datos 3D capaces de representar las propiedades geométricas y físicas.

Debemos ser capaces de representar el interior, el exterior y el comportamiento de la superficie.

Aproximaciones al problema:

- Definición de la superficie exterior: Modelos de superficie, solo interesa el contorno (exterior).
- Definición del espacio ocupado por el objeto: Modelado Sólido, para saber exterior, interior y propiedad de los objetos.

Requisitos:

- Facilidad en la generación de objetos.
- Calidad en el aspecto final de la representación, según las restricciones.
- Recursos computacionales de la aplicación.

Representación de Objetos 3D

- Modelos de Superficie.
  - Poligonales.
  - Matemáticos.
    - Superficies Implícitas.
    - Superficies Paramétricas.
- Modelo Sólido.
  - Modelos de Barrido.
  - Geometría Sólida Constructiva (GSC).
- Modelos de Partición Espacial.

4.2.1. Modelos de Superficie

Se emplea cuándo

- El objeto a representar es muy similar a una superficie.
- Solo interesa el aspecto externo del objeto.
- El tipo de biblioteca gráfica solo soporta este modelo (no hay más remedio).

Para modelar superficies debemos asegurarnos de que las superficies no se intersecan consigo mismas.

Cuando la superficie es cerrada se denominan a los objetos modelos de frontera → posibilidad de cambio con modelos sólidos.

Aproximaciones al modelado de superficies:

- Discreta y aproximada → modelo poliédrico
- Continua y exacta → modelo matemático

Modelos de Superficie: Poligonales

Describir una superficie a partir de un conjunto de polígonos conectados mediante vértices y aristas.

Los polígonos se intersecan únicamente en las artistas

- Una **arista** es compartida únicamente por dos polígonos.
- Un **vértice** pertenece a dos o más aristas

Cualquier forma 2D o superficie 3D puede aproximarse por polígonos, pero la calidad de la representación mejora a medida que aumentamos el número de polígonos.

Ventajas

- Simplicidad
- Fácil y rápido de renderizar
- Ocupa poca memoria

Limitaciones

- La naturaleza no es poligonal.

- Aliasing de polígonos, alisamiento para quitar bordes de sierra, debido a la discretización.
- No es sencillo modelar objetos complejos.

Normalmente, se trabaja a partir de primitivas poligonales optimizadas, por su reducción de la trasferencia de datos, el problema es la perdida de flexibilidad.

- Tira de cuadrados / Tira de triángulos
- Matriz de cuadrado / Abanico de triángulos

La primitiva poligonal más habitual es el **Triangulo**, es siempre convexo, matemáticamente simple y siempre coplanares. Además, cualquier polígono puede ser descompuesto en triángulos.

PovRay utiliza triángulos para definir superficies: Mesh almacena eficientemente un número de triángulos y Mesh2 utilizado para la conversión entre formatos gráficos.

Modelos de Superficie: Matemáticos

Ecuaciones matemáticas describen las superficies

- **Superficies Implícitas** $F(x, y, z) = 0$.
  - **Cuádricas**

    Primitivas matemáticas que responden a la ecuación, que dependiendo de los coeficientes será una forma u otra:

    $$a \cdot x^2 + b \cdot y^2 + c \cdot z^2 + 2d \cdot xy + 2e \cdot yz + 2f \cdot xz + 2g \cdot z + 2h \cdot y + 2j \cdot z + k = 0$$

    Limitada variedad de las formas (no es simple extenderlo a representación de sólidos) y no sirven para modelar elementos naturales.
  - **Superficies equipotenciales o Isosuperficies**

    Útiles para representación de formas suaves

    Cada superficie queda definida por el conjunto de puntos con un determinado campo, se modela según el equilibrio entre campos de potencia (equipotencial).

    Al interactuar objetos se crean superficies suaves. El modelado se basa en mover los puntos o elementos que generan el campo, habitualmente los campos son esféricos o cilíndricos.
  - **Blobs:** Globulares de formas abultadas.

    $$f(x, y, z) = \Sigma_k b_k e^{-r_k^2 a_k} - T = 0$$

    T = punto iniciar. a = abultamiento (positivo). b = hendidura (negativos).

- **Metaballs y Softobjects:** Bultos gaussianos

$$f(r) = \begin{cases} b(1 - 3r^2/d^2) & si\ r \in [0, d/3] \\ \frac{3}{2}b(1 - r/d)^2 & si\ r \in [d/3, d] \\ 0 & si\ r > d \end{cases}$$

- **Superficies Paramétricas** $(x, y, z) = f(u, v)$. Bezier, B-splines y NURBS.

  Características

  - Facilidad de modelado de formas libres: Control local de deformaciones y Posibilidades de cálculo de tangencias y curvatura
  - Buenas prestaciones en transformaciones a modelos poligonales
  - Buenas prestaciones en cuanto a almacenamiento

  Dos aproximaciones

  1. Con superficie que pasa por un conjunto de puntos → **Superficies de Interpolación**
  2. Un conjunto de puntos controlan la forma de la superficie, que no pasa necesariamente por ellos → **Superficies de aproximación**

  Podemos aproximar cualquier curva unidimensional

  - La más básica se basa en el uso de funciones polinómicas $C(u) = \Sigma_{i=0}^{n} a_i u^i$
  - La curva depende del grado del polinomio: para definir una curva con n puntos es necesario un polinomio de grado n (problema!!!)
  - Otra posibilidad consiste en definir la curva como combinación de trozos de curvas: splines. Cuidando los grados de continuidad.

  Tipos:

  - **Splines naturales**

    Normalmente se basan en polinomios de grado cúbico (n=3 en la ecuación), donde se exige continuidad y diferenciabilidad doble.

    Por tanto, la curva consiste en n-1 trozos de grado 3, donde n es el número de puntos por los que pasa la curva.

  - **Curvas de Bézier**

    Solamente pasan por los puntos extremos y cumplen la propiedad de cierre convexo del polígono de control.

    Se definen a partir de la combinación de una familia especial de polinomios: los polinomios de Berstein

    Dos aproximaciones: Curvas de Bezier basadas en polinomios de grado n y de las definidas a trozos.

Curva de Bezier generales:
$C(u) = \Sigma_{i=0}^{n} B_{i,n}(u) P_i \;\; 0 \leq u \leq 1$ con
$B_{i,n}(u) = \frac{n!}{i!(n-1)!} u^i (1-u)^{n-i}$ con $\Sigma_{i,n}^{n}(u) = 1 \forall u$

- **B-splines**

  Similar a las splines naturales, pero sin interpolación. Se une el espacio paramétrico de los diferentes trozos. Esto se realiza definiendo un vector de nudos.

  Ventajas

  1. Se puede seleccionar el grado de los polinomios base para controlar la suavidad de la curva (grado d)
  2. Permiten control local sobre la forma de la superficie

  $C(u) = \Sigma_{i=0}^{n} N_{i,d}(u) P_i \; u_{min} < u < u_{max}, 2 \leq d \leq n+1$
  $N_{i,1}(u) = 1$ si $u_i < u < u_{i+1}$ en caso contrario 0.
  $N_{i,d}(u) = \frac{u-u_i}{u_{i+d-1}-u_i} N_{i,d-1}(u) + \frac{u_{i+d}-u}{u_{i+d}-u_{u+1}} N_{i+1,d-1}(u)$
  Según la distribución de vectores de nudos:

  - **B-splines Racionales** Se definen a partir de la razón o promedio de los polinomios base. Se pueden representar cuádricas y otras superficies de manera exacta.
    $$C(u) = \frac{\Sigma_{i=0}^{n} w_i N_{i,d}(u) P_i}{\Sigma_{i=0}^{n} w_i N_{i,d}(u)}$$
  - **NURBS-Non Uniform Rational B-Splines:** Cuando podemos tener cualquier distribución del vector de nudos.

- **Superficies Explícitas** $Z = f(x, y)$, se tratan como superficies paramétricas

4.2.2. Modelado Sólido

Modelos de Barrido

1. Definición de una figura bidimensional (por sus bordes)
2. Desplazamiento de la figura bidimensional a lo largo de un camino y los bordes generan la superficie.
   - Rotación respecto de un eje "Superficie de revolución"
   - Desplazamiento a lo largo de una línea recta "Extrusión"

Geometría Sólida Constructiva (GSC)

La combinación de primitivas u objetos mediante operaciones booleanas, produciendo objetos más complejos.

Los objetos diseñados con CSG se representa mediante un árbol binario con nodos y operaciones

Las operaciones básicas son: Unión, Diferencia, Intersección y Fusión.

5. TRANSFORMACIONES GEOMÉTRICAS

5.1. Introducción

Transformación geométrica: Variación del tamaño, forma, posición y orientación de un objeto dentro de una escena.

- Transformaciones geométricas afines en 2D y 3D.
    - Transformaciones lineales y traslaciones:
        - Traslación
        - Escalamiento
        - Rotación
- Transformaciones no afines.

5.2. Aplicaciones en Informática Gráfica

Para modelar objetos.

- Ejem. Blender se puede partir de un cubo, escalarlo en una dirección y rotarlo para formar una columna.

Para crear objetos complejos.

Para animar objetos, traslaciones y rotaciones son los elementos básicos del movimiento.

5.3. Transformaciones lineales

Una transformación es una función: $v(x, y, z) \rightarrow v'(x', y', z')$.

Una transformación es lineal, si se cumple:

- $T(u + v) = T(u) + T(v)$ La transformación del vector suma es la suma de las transformaciones.
- $T(x \cdot u) = x \cdot T(u)$, siendo x un escalar.

Son lineales: Rotación, Escalado y Estiramiento (shear).

En las transformaciones lineales se cumple que las líneas paralelas siguen siendo paralelas (los ángulos no tienen por qué mantenerse).

El origen de coordenadas se mantiene.

No es lineal: Traslación.

En la traslación se mantiene el paralelismo, pero el origen de coordenadas no se mantiene.

La traslación no es una transformación lineal, pero sí es una transformación afín.

Transformación afín: transformación lineal y traslación.

5.4. Transformaciones 2D

5.4.1. Forma General

Las transformaciones lineales se pueden representar con matrices cuadradas. La traslación también, pero usando coordenadas homogéneas (una dimensión más).

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}$$

- a es eje x.
- d es eje y.
- b es deformación en x.
- c es deformación en y.

5.4.2. Transformación lineal 2D

Vector como combinación lineal de los vectores base i, j. Cuando se haga una transformación cambia el vector base.

- Ejem. $v(-1, 2) = -1i + 2j$

i es la primera columna de la matriz de transformación y j la segunda.

5.4.3. Traslación

Cambio de la posición de un objeto a lo largo de una línea recta.

Desplaza cada punto de una figura o espacio la misma cantidad en una determinada dirección, vector de traslación (d_x, d_y). d_x unidades paralelas al eje x y d_y unidades paralelas al eje y.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix} = \begin{bmatrix} x + d_x \\ y + d_y \end{bmatrix}$$

NO es una transformación lineal, aunque si afín. La transformación de la suma de dos vectores no es la suma de las transformaciones.

5.4.4. Escalado

Alteración de las dimensiones de un objeto:

- Segmentos paralelos al eje x se multiplican por s_x.
- Segmentos paralelos al eje y se multiplican por s_y.

Cuando $s_x = s_y$ es un escalado uniforme.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x \cdot s_x \\ y \cdot s_y \end{bmatrix}$$

5.4.5. Rotación

Cambiar la orientación de un objeto θ respecto del origen.

ϕ es la posición angular del punto $P(x, y)$ respecto al origen.

Los vectores base pasarán a ser:

- $i' = (1\cos\theta, 1\sin\theta)$
- $j' = (-1\sin\theta, 1\cos\theta)$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \end{bmatrix}$$

5.4.6. Cuidado con el orden de las transformaciones

Secuencia Rotación y Escalado no es equivalente a Escalado y Rotación.

Se multiplica de Derecha a Izquierda.

No se cumple la propiedad conmutativa, pero si la asociativa.

- $A2 = S \cdot R \cdot A$
- $A2 = (S \cdot R) \cdot A$

Aunque existen casos especiales donde al aplicar transformaciones de traslación, escalado y rotación, el producto de matrices es conmutativo, como:

- Traslación seguida de traslación.
- Escalado seguido de escalado
- Rotación seguida de rotación
- Rotación y escalado uniforme

Se busca ejecutar la secuencia de transformaciones más eficientemente, para esto necesitamos un sistema de referencia que sea homogéneo y entre la traslación.

5.5. Sistema de Coordenadas Homogéneas 2D

Un punto (x, y) en el sistema cartesiano se representa por (xW, yW, W) en el sistema homogéneo, donde $W \neq 0$ puede ser cualquier número real (geometría proyectiva).

Cuando $W \neq 1$, para simplificar cálculos, dividimos todas las coordenadas por W, obteniendo $(x, y, 1)$ (W suele utilizarse en el eje z profundidad).

Cuando $W = 1$, estamos realizando transformaciones del punto (x, y) sobre el plano $z = 1$.

Las matrices que representan puntos en 2D serán de 3x3.

5.5.1. Traslación en coordenadas homogéneas

Ahora la traslación podrá hacerse representarse como una matriz.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + d_x \\ y + d_y \\ 1 \end{bmatrix}$$

Es como el bias de las redes de neuronas, el 1 se multiplica por un desplazamiento.

Ahora se puede hacer composición de traslaciones.

$P' = T(d_{x1}, d_{y1}) \cdot P; \quad P'' = T(d_{x2}, d_{y2}) \cdot P'$

$P'' = T(d_{x2}, d_{y2}) \cdot (T(d_{x1}, d_{y1}) \cdot P) = (T(d_{x2}, d_{y2}) \cdot T(d_{x1}, d_{y1})) \cdot P$

$$\begin{bmatrix} 1 & 0 & d_{x1} \\ 0 & 1 & d_{y1} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & d_{x2} \\ 0 & 1 & d_{y2} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & d_{x1} + d_{x2} \\ 0 & 1 & d_{y1} + d_{y2} \\ 0 & 0 & 1 \end{bmatrix}$$

5.5.2. Escalado en coordenadas homogéneas

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x \cdot s_x \\ y \cdot s_y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} s_{x2} & 0 & 0 \\ 0 & s_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_{x1} & 0 & 0 \\ 0 & s_{y1} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_{x1} \cdot s_{x2} & 0 & 0 \\ 0 & s_{y1} \cdot s_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

5.5.3. Rotación en coordenadas homogéneas

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\beta & -\sin\beta & 0 \\ \sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos(\alpha+\beta) & -\sin(\alpha+\beta) & 0 \\ \sin(\alpha+\beta) & \cos(\alpha+\beta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

5.5.4. Transformaciones inversas

Traslación $T(d_x, d_y)$ es: $T^{-1}(d_x, d_y) = T(-d_x, -d_y)$

Escalado $S(s_x, s_y)$ es: $S^{-1}(s_x, s_y) = S(1/s_x, 1/s_y)$

Rotación $R(T)$ es: $R^{-1}(\theta) = R(-\theta)$ que equivale a cambiar los signos de los senos de la transformación normal.

5.5.5. Transformaciones generales afines

Producto de una secuencia arbitraria de matrices de rotación, traslación y escalamiento.

Propiedad de conservar el paralelismo de las líneas, pero no longitudes ni ángulos. Rotaciones, escalamientos y traslaciones subsiguientes no podrían hacer que las líneas dejen de ser paralelas (nótese que sí permiten deformar el objeto).

5.5.6. Estiramiento (Shear)

El estiramiento puede realizarse respecto de cualquier eje.

$$SH_x = \begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad SH_y = \begin{bmatrix} 1 & 0 & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

5.5.7. Composición de transformaciones

El objetivo es ganar eficiencia aplicando una sola transformación compuesta a un punto, como es la aplicación de una rotación a un objeto sobre un punto P, que no es el origen.

1. Traslación al origen $T(-x_1, -y_1)$
2. Rotación $R(\theta)$
3. Traslación al punto P $T(x_1, y_1)$

5.5.8. Operaciones respecto a un punto $P_1(x_1, y_1)$

Recordar que se hace las operaciones de derecha a izquierda.

Rotación

$$T(x_1, y_1) \cdot R(\theta) \cdot T(x_1, y_1) = \begin{bmatrix} \cos\theta & -\sin\theta & x_1(1-\cos\theta) + y_1 \sin\theta \\ \sin\theta & \cos\theta & y_1(1-\cos\theta) - x_1 \sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

Escalado

$$T(x_1, y_1) \cdot S(s_x, s_y) \cdot T(x_1, y_1) = \begin{bmatrix} s_x & 0 & x_1(1-s_x) \\ 0 & s_y & y_1(1-s_y) \\ 0 & 0 & 1 \end{bmatrix}$$

Escalado y Rotación

$M = T(x_1, y_1) \cdot R(\theta) \cdot S(s_x, x_y) \cdot T(-x_1, -y_1)$

5.6. Sistema de Coordenadas Homogéneas 3D

Un punto (x, y, z) en el sistema cartesiano se representa por (xW, yW, zW, W) en el sistema homogéneo, donde $W \neq 0$ puede ser cualquier número real.

Cuando $W \neq 1$, para simplificar cálculos, dividimos todas las coordenadas por W, obteniendo $(x, y, z, 1)$

5.6.1. Traslación

$$T(d_x, d_y, d_z) \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + d_x \\ y + d_y \\ z + d_z \\ 1 \end{bmatrix}$$

5.6.2. Escalado

$$S(s_x, s_y, s_z) \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \cdot s_x \\ y \cdot s_y \\ z \cdot s_z \\ 1 \end{bmatrix}$$

5.6.3. Rotación

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5.6.4. Rotación en coordenadas homogéneas

La composición de una secuencia arbitraria de rotaciones con respecto a los ejes x, y, z como submatriz 3x3 ortogonal. La inversa de una matriz ortogonal es su transpuesta.

$$R_z(\theta) = \begin{bmatrix} r11 & r12 & r13 & 0 \\ r21 & r22 & r23 & 0 \\ r31 & r32 & r33 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5.7. Rotación alrededor de cualquier eje

- Primero movemos el eje hasta el eje z.
    1. Rotamos alrededor de x R_x
    2. Rotamos alrededor de y R_y
- Después, aplicamos la rotación deseada seguida de las inversas usadas para colocarlo sobre el eje z.
    1. Rotamos lo deseado R_z
    2. Deshacemos la rotación $R_x^{-1}R_y^{-1}$

$$R_{final} = R_x^{-1}R_y^{-1}R_zR_yR_x$$

5.8. Rotación alrededor de cualquier eje y punto de origen

1. Traslación al origen deseado T_{xyz}
2. Rotamos $R_x^{-1}R_y^{-1}R_zR_yR_x$
3. Invertir traslación T_{xyz}^{-1}

$$R_{final} = T_{xyz}^{-1}R_x^{-1}R_y^{-1}R_zR_yR_xT_{xyz}$$

5.9. Matrices en PovRay

Las matrices se definen con matrix <v00, v01, v02, v10, ...>, en el que la última columna se asume que es 0, 0, 0, 1. De manera que la matriz está traspuesta, siendo la última fila la de traslación dx, dy, 0, 0 (está rotada).

6. PRIMITIVAS 2D

6.1. Definiciones

Pixel: Mínima unidad homogénea de color (intensidad) de una imagen digital. Área de la pantalla que tiene asociada una posición de memoria (pixel de 8 bits tiene 256 variaciones de color).

Primitivas de salida: Estructuras geométricas básicas continuas (puntos, rectas, curvas, áreas, ...) empleadas para generar imágenes

Conversión al «raster»: Aproximar primitivas mediante un conjunto de puntos discreto (mapa de píxeles)

- Localizar posiciones de pixel más próximas al objeto.
- Almacenados valores intensidad/color en búfer (PIXMAP)
- Trazado de imagen en pantallas (intensidad a píxeles)

Búfer de imagen: Área de memoria que almacena el conjunto de valores de intensidad para todos los puntos de la pantalla.

- B/N 1 bit por pixel, 2 intensidades. BITMAP.
- Color n bits por pixel, 2^n intensidades. PIXMAP.
- Sistemas alta calidad hasta 24 bits por pixel. Sistema color RGB de 24 bits.

Procesador graficó/ Controlador de gráficas: libera CPU de trabajos gráficos.

6.2. Conversión de puntos

Problema - convertir al raster un punto definido mediante una coordenada almacenando el color con el que se va a visualizar.

Algoritmo sencillo de conversión, sea (x, y) el punto real, la posición en el raster es $(x', y') = (round(x), round(y))$. $round()$ entero más próximo.

6.3. Conversión de rectas

Problema - calcular las coordenadas de los píxeles que representan una recta infinitamente delgada sobre una malla de un raster 2D.

Un buen rasterizador debe cumplir lo siguiente:

- Secuencia de píxeles lo más recta posible.
- Líneas con el mismo grosor independientemente de la pendiente.
- El algoritmo debe ser muy rápido.

6.3.1. Fuerza bruta

Ecuación de la recta entre puntos extremos (x_1, y_1) y (x_2, y_2), $y = mx + b$.

- $m = \frac{(y_2 - y_1)}{(x_2 - x_1)}$
- b se halla usando la pendiente, m, y los puntos conocidos. $b = y_1 - m * x_1$

Para cada punto x entre x_1 y x_2 se calcula el pixel más cercano a la curva definida por la recta. Se va aumentando progresivamente la x.

Restricción: m tiene que estar entre -1 y 1, si el grado que forma es mayor que 45 se forma de manera discontinua.

Solución: Intercambiar la x por la y en los extremos de la regla, hacer el proceso e intercambiarlas de nuevo.

Desventajas: Ineficiente, requiere multiplicaciones en coma flotante y función round().

6.3.2. Algoritmo Analizador Diferencial Digital (DDA, Digital Diferenctial Analizer)

Algoritmo que traza valores sucesivos de (x, y) incrementando simultáneamente x e y con pequeños pasos.

1. Calculamos $\Delta x = x_2 - x_1$ y $\Delta y = y_2 - y_1$.
2. Calculamos los incrementos que usaremos para dibujar, x_{inc} y y_{inc}. $x_{inc} = \frac{\Delta x}{\text{máx}(|\Delta x|,|\Delta y|)}$ y $y_{inc} = \frac{\Delta y}{\text{máx}(|\Delta x|,|\Delta y|)}$.
3. Para el eje vamos iterando: $x = x + x_{inc}$ y $y = y + y_{inc}$. Redondeando aquellos que sean decimales.

Ventaja: elimina el producto de coma flotante.

Desventaja: invoca la función round.

Seleccionamos un eje con pasos unitarios (según el valor de $|m|$)

Representamos la línea de izquierda a derecha si es positivo y derecha a izquierda si es negativo.

6.3.3. Algoritmo del punto medio de Bresenham (Bresenham midpoint algorithm)

Consiste en determinar si cuando se calcula el punto debe ir a superior o inferior según la distancia del punto real con estos. El factor de decisión, p, nos indicará cuál coger.

1. Calculamos $\Delta x = x_2 - x_1$ y $\Delta y = y_2 - y_1$. Además, para facilitar los cálculos podemos también calcular $2\Delta y - 2\Delta x$ y $2\Delta x$.
2. Calculamos el $p_0 = 2\Delta y - \Delta x$
3. Comenzando en $k = 0$, repetimos Δx veces:
   a) Si $p_k < 0$ entonces $x_{k+1} = x_k + 1$, $y_{k+1} = y_k$ y $p_{k+1} = p_k + 2\Delta y$
   b) Si $p_k > 0$ entonces $x_{k+1} = x_k + 1$, $y_{k+1} = y_k + 1$ y $p_{k+1} = p_k + 2\Delta y - 2\Delta x$
4. Dibujamos los puntos calculados

7. FRACTALES

Figuras que se autorrepiten, es decir, la figura está formada de sí misma.

Esta estructura se da en la naturaleza, en los vasos sanguíneos, cuenca del Amazonas o las vías romanas. Se pueden dar en forma de ramificación, en espiral o en conos.

- Ejem. Cálculo de la longitud de la costa de Gran Bretaña, que se trata de estimar con un segmento de x metros, cuanto más pequeño lo vayamos haciendo más grande sale la longitud. Por lo que siempre hay un segmento más pequeño.

7.1. Definición

El concepto de fractal aparece en 1975 con el libro Benoit Mandelbrot - «Los objetos Fractales: forma, azar y dimensión».

Aunque se habían descubierto antes y se llamaron «monstruos matemáticos».

Un objeto fractal tiene dos características básicas:

- **Infinito detalle** en cada punto
- **Auto similitud entre las partes** del objeto y su totalidad

Veremos **siempre la misma figura**, independientemente de lo que nos acerquemos al mismo (fractal auténtico).

Su geometría requiere **gráficos por ordenador** para visualizarse e investigarlos.

7.2. Caracterización

Geometría Euclídea	Geometría Fractal
Ecuaciones	Procedimientos (sin ecuaciones)
Objetos fabricados	Objetos naturales
Diferenciable, localmente suave	No diferenciable, localmente rugoso
Aumenta el detalle y definición a distancia corta	Mismo detalle desde todas las distancias

Tabla 7.1: Comparación Geometría Fractal y Euclídea

7.3. Aplicaciones

- Antenas fractales (ancho de banda)
- Compresión de imágenes
- Sistemas dinámicos (cotización de valores bursátiles)
- Modelado de formas naturales (helechos, montañas o relieve)
- Arte fractal.

7.4. Procedimiento de Generación

Aplicar recursivamente una función de transformación a los puntos de una región del espacio:

- Tenemos una función de transformación $F(X)$ y partimos de punto inicial seleccionado $P_0 = (x_0, y_0, z_0)$
- Cada vez que aplicamos F, generamos niveles sucesivos de detalle, como $P_1 = F(P_0), P_2 = F(P_1), ...P_{k+1} = F(P_k), ...$
- Sistema dinámico realimentado

$F(X)$ puede aplicarse a: Conjunto específico de puntos, o a un Conjunto inicial de primitivas (líneas rectas, áreas de color, superficies o sólidos).

$F(X)$ puede definirse en términos de transformaciones: Geométricas (escalado, traslación, rotación) o Coordenadas no lineales y parámetros de decisión.

No podemos desplegar variaciones de detalle inferiores a un pixel $\rightarrow F(X)$ se aplica un número finito de veces. Para ver más detalle acercamos la imagen y aplicamos $F(X)$ de nuevo.

Los procedimientos a utilizar pueden ser Deterministas o Aleatorios / estocásticos.

7.5. Clasificación de Fractales

- **Autosimilares:** Sus partes son versiones a escala de su totalidad. Se aplica un parámetro de escala (S) toda la forma inicial.

  Autosimilar estadísticamente si aplicamos variaciones aleatorias.

- **Autoafines:** Un factor de escala en cada eje, S_x, S_y, S_z.

  Autoafines estadísticamente si aplicamos variaciones aleatorias.

- **Invariantes:** Transformaciones no lineales. Fractales autocuadráticos (Mandelbrot), autoinversos...

7.6. Dimensión Fractal

Medida de la variación de detalle de un objeto fractal.

En la geometría euclídea los objetos tienen dimensión entera. Los objetos en la geometría fractal tiene dimensión fraccionaria (o dimensión fractal).

La dimensión fractal de un objeto siempre es mayor que su dimensión euclídea (o topológica).

Dada una dimensión, podemos generar objetos fractales mediante procedimientos recursivos.

Podríamos calcular la dimensión fractal de un objeto a partir de las propiedades del mismo, mediante otros procedimientos (tarea difícil de realizar).

Cálculo de la dimensión de un fractal autosimilar determinista (un solo factor de escala) por analogía a las subpartes de un objeto euclidiano.

$NS^{DE} = 1 \rightarrow NS^{D} = 1$

- DE = Dimensión euclídea
- D = Dimensión fractal
- N = n. piezas semejantes
- S = factor de escala

$$N = \frac{1}{S^D} = \left(\frac{1}{S}\right)^D ; \ln N = D \ln \frac{1}{S}; D = \frac{\ln N}{\ln \frac{1}{S}}$$

Dimensión de una **curva fractal en un plano 2-dim** (litorales)

- Habitualmente $1 < D \leq 2$ (más suave cuanto más cerca de 1).

  $D = 2$ cuando llena una región finita del plano
- $2 < D < 3$ cuando la curva se autointersecta y el área puede cubrirse infinitas veces

Dimensión de una **curva fractal espacial**

- En general $1 < D$ y puede ser $2 < D \leq 3$ sin autointersectarse.

  $D = 3$ cuando llena un volumen de espacio.
- $3 < D < 4$ cuando la curva se autointersecta y el volumen de espacio puede cubrirse infinitas veces.

Dimensión de una **superficie fractal** (tierra, nubes, agua)

- Habitualmente $2 < D \leq 3$ (más suave cuanto más cerca de 1).

  $D = 3$ cuando llena un volumen de espacio.

- $3 < D < 4$ cuando la superficie se autointersecta y el volumen de espacio puede y cubrirse infinitas veces.

Dimensión de un **solido fractal** (densidad de vapor de agua, temperatura en una región del espacio)

- En general $3 < D \leq 4$.
- $4 < D$ cuando el sólido se autointersecta.

7.7. Fractales autosimilares

Construir un fractal autosimilar:

- Iniciador. Forma geométrica determina.
- Generador. Patrón que siguen las subpartes del iniciador.
- Reglas básicas de construcción.
  - Infinito detalle en cada punto.
  - Auto similitud entre las partes del objeto y su totalidad.

Patrones muy diversos. Añadir/sustraer partes o características del generador.

Se puede añadir carácter aleatorio en la construcción de un fractal autosimilar.

- Seleccionando generador al azar.
  - Rotado aleatorio del generador o alguna de sus partes.
  - Escalado aleatorio del generador o alguna de sus partes.
  - Selección aleatoria del color.
  - Selección aleatoria de bultos.
- Desplazamiento de coordenadas al azar. Traslación aleatoria del generador

7.8. Fractales autosimilares deterministas

Conjunto de Cantor (1883) - subconjunto fractal de [0,1] que elimina, en cada paso, el segmento correspondiente al tercio central en cada intervalo. Línea que se parte en 3 y se elimina la del medio. $D = \frac{\ln 2}{\ln 3} = 0,63 = \frac{\ln(partes)}{\ln(escala)}$.

- Autosimilar
- Invariante respecto de la escala
- No puede ser descrito analíticamente

Curva de Koch (1904) - Línea que se divide en 4 partes, formando un pico, que ocupa el tercio central. $D = \frac{\ln 4}{\ln 3} = 1,26$.

- Continua en todos los puntos
- No derivable en ningún punto

Copo de Koch (1904) - Triángulo que se divide cada lado en 4 partes cada cara, formando un pico, que ocupa el tercio central. $D = \frac{\ln 4}{\ln 3} = 1,26$.

Triángulo de Sierpinski - Triángulo que divide su área en 4 trozos desechando el central, cada uno de estos triángulos tiene de lado la mitad. $D = \frac{\ln 3}{\ln 2} = 1,58$.

Alfombra de Sierpinski - Cuadrado que se divide su área en 9 trozos desechando el central, cada uno de estos cuadrados tiene un tercio de lado. $D = \frac{\ln 8}{\ln 3} = 1,89$.

Curva de Peano - S en la que cada lado se divide en 3, pero el espacio total se divide, por tanto, en 9 partes. $D = \frac{\ln 9}{\ln 3} = 2$.

- En el límite, recubre todo el plano
- Similar a la curva de Hilbert

7.9. Fractales autoafines

Construcción de fractales autoafines es igual que la de los fractales autosimilares pero con distintos factores de escala aplicados a cada una de las dimensiones del generador.

Podemos añadir aleatoriedad mediante el movimiento browniano

- Modelamos una curva fractal - desde posición inicial (x, y)
    - Generamos **dirección** aleatoria.
    - Generamos **longitud** aleatoria.

- Matriz bidimensional de saltos del browniano
    - Sobre una **cuadrícula de plano** (terreno).
    - Sobre una **esfera** (planeta).

Una trayectoria browniana se define como la sucesión de puntos (variables aleatorias) $s_0, s_1, \ldots, s_n$ definidas por $s_n = s_{n-1} + \textbf{salto}X_n$ en sus respectivos periodos de tiempo $0, \Delta t, 2\Delta t, \ldots, n\Delta t$ donde $salto = \sqrt{\Delta t}$ permite controlar la varianza de la trayectoria y las $X_1, \ldots, X_n$ con variables aleatorias con una distribución determinada. $X_j \sim U(0,1), N(0,1), \ldots$.

Ajustar dimensión fractal en cálculos del browniano para dar más realismo modelado.

Escalar elevaciones → incrementar/diminuir «saltos» browniano.

7.10. Modelos Gramaticales

Smith presenta un método para describir la estructura de ciertas plantas.

Se utiliza lenguajes gramaticales (gramáticas-L) de grafos paralelos. Lenguajes descritos con una gramática que consiste en una colección de **producciones que se aplican todas a la vez**.

El lenguaje generado por la gramática, interpretado de forma apropiada, nos da la sucesión de operaciones que debemos realizar para generar el objeto.

En general, más de una regla para un mismo símbolo no terminal. Podemos seleccionar la regla a aplicar de forma aleatoria

Lindenmayer extendió los lenguajes para que incluyeran corchetes (L-Systems). Podemos añadir a las gramáticas un operador de pila, de manera que lo que hay en los corchetes se meta en la pila. También podemos añadir paréntesis para enriquecer construcción

Especificación:

- Palabra en el lenguaje → secuencia de segmentos en una estructura gráfica.
- Porciones entre corchetes representan porciones que se ramifican desde el símbolo anterior.

Obtención de gramáticas que representen con precisión la biología de plantas durante el desarrollo

Variaciones:

- Las gramáticas se han enriquecido para permitir llevar un registro de la “edad” de la letra en una palabra. Las letras “viejas” y “jóvenes” se transforman de distinta manera.

- Se puede ajustar la longitud correspondiente al movimiento hacia delante de tal forma que en cada iteración las ramas se vayan haciendo cada vez más pequeñas.
- Aplicar de forma aleatoria el número de llamadas recursivas de la expansión, lo que hace que se generen árboles de diferente tipo.

7.11. Fractales autocuadráticos

Aplicación recursiva de función de transformación F(z) a puntos en el espacio complejo (función compleja).

Representamos un punto y su módulo en el plano complejo como: $z = x + iy$ $|Z| = (x^2 + y^2)^{\frac{1}{2}}$.

Definimos dicha función $F : Z \to Z$ como: $z_{k+1} = F(z_k) = z_k^2 + c$.

Otra familia de transformación rica en fractales (λ complejo) es: $F(z_k) = \lambda z_k^2(1 - z_k)$.

Al aplicar recursivamente cuadrados a un número complejo $F(x) = x^2 + c$, este:

- Tiende a infinito si $|z| > 1$
- Tiende a cero si $|z| < 1$
- Permanece en $|z| = 1$ si $|z| = 1$

Dependiendo del punto inicial, $z_0 = c$, la sucesión de transformaciones:

- Divergen a infinito si $|z| > 1$
- Convergen a un punto de atracción (atractor) si $|z| < 1$
- Son periódicas si $|z| = 1$

7.11.1. Conjunto de Julia

Para cada valor de $c = a + bi$, hay un conjunto de Julia diferente

$$F(z) = z^2 + c$$

Frontera fractal que separa:

- Los puntos que divergen a infinito.
- Los que convergen a un punto de atracción.

7.11.2. Conjunto de Mandelbrot

- El rey de los monstruos matemáticos.
- Es conexo.
- Contiene infinitas copias de sí mismo. Las copias están conectadas al cuerpo principal por "cadenas".
- Tiene dimensión fractal 2.

Calculamos el conjunto de Julia para cada valor de c y coloreamos:

- El punto de negro cuando el conjunto de Julia está conectado (continuo).
- El punto de blanco cuando el conjunto de Julia no está conectado (discontinuo).

Tenemos la función de transformación: $z_{k+1} = F(z_k) = z_k^2 + c$

Elegimos el punto inicial $z_0 = 0 + 0i$ y le aplicamos la función de transformación, para cada valor de c (de la forma $a + bi$.) Un punto c está en el conjunto de Mandelbrot si y solo si todos los puntos generados por esta función tienen como módulo un valor finito.

Fijamos un radio máximo n como límite de para las posiciones sucesivas.

En la práctica, si $|z_k| > n$, valores sucesivos de $|z_k|$ serán más y más grandes. Por tanto, podemos detener el cálculo para valores siguientes a z_k

Fijamos un número máximo de iteraciones k:

- Si $|z_k| > n$ antes de alcanzar k (la función diverge) $\rightarrow$ colorear el punto c de blanco.
- Si $|z_k| < n$ antes de alcanzar k (la función converge) $\rightarrow$ colorear el punto c de negro (c pertenece al conjunto).

Consideramos cuanto tarda la función en divergir para un cierto c representar con diferentes colores

7.12. Fractales y caos

El juego del caos:

- Dado tres puntos, A, B, C y un punto inicial arbitrario z_0.
- Seleccionar aleatoriamente A, B o C y calcular $z_1 = \frac{P+z_0}{2}, z_n = \frac{P+z_{n-1}}{2}$.
- Seleccionar.

Algoritmo de Midpoint Displacement:

1. Dividir la escena en regiones.
2. Cambiar la altura de las esquinas de cada región aleatoriamente.
3. Dividir cada región en regiones más pequeñas.
4. Cambiar la altura de las esquinas de cada región aleatoriamente, pero en menor cantidad.
5. Volver a 3.

8. TEORÍA DEL COLOR

8.1. Introducción

La percepción del color de los objetos depende de:

- Propiedades del objeto.
- Fuente de luz.
- Color del entorno.
- Sistema visual humano (no solo el ojo, también el cerebro).

La realidad física, son longitudes de onda, pero la percepción son los colores.

Existen diversas teorías, técnicas de medida, estándares..., pero no hay una teoría universalmente aceptada sobre la percepción del color que realizamos los humanos.

La ciencia del color es una mezcla de componentes subjetivos (percepción) con componentes físicos (longitudes de onda, espectros)

8.2. La Luz

La luz (gama de colores) es la banda de longitudes (o frecuencias) de onda del espectro electromagnético, que es visible para nosotros. Son de longitud 750-400 nm.

8.2.1. Historia

Siglo XIX aceptaban que la luz tenía naturaleza ondulatoria, aunque no se relacionaba con la electricidad ni con el magnetismo.

- Pero todas las ondas conocidas necesitaban un medio para propagarse y la luz viajaba en el vacío con más velocidad que en medios como el aire o el agua. Se proponía la existencia del éter luminífero, que se desecha en el s. XIX.

En 1845 Faraday (aproximación práctica) descubrió que el magnetismo altera el plano de polarización de la luz (relación entre la luz y el magnetismo). Y especuló que la luz podía ser resultado de la vibración de las líneas de fuerza eléctricas y magnéticas.

Nadie le hizo caso, hasta que Maxwell (aproximación teórica) publica las ecuaciones del electromagnetismo. Deduce que deben existir perturbaciones electromagnéticas que se autopropagan en el vacío. Mediante una ecuación consigue sin necesidad de punto de referencia aproximar la velocidad de la luz mediante campos.

Quedó aceptado que la luz es una radiación electromagnética.

A finales del siglo XIX se encontraron nuevos efectos que no se podían explicar con la naturaleza ondulatoria de la luz.

- Radiación del cuerpo negro
- Efecto fotoeléctrico

Solo se podían explicar si la luz consistía en partículas: cuantos de energía, fotones. Plank $E = hv$

La luz se puede manifestar como onda o como partícula (dualidad ondapartícula)

8.2.2. Radiación de cuerpo negro

La energía que emite un cuerpo depende de su temperatura. Esta temperatura les confiere un color que puede parecer contradictorio, son:

Temperatura más alta → Temperatura de color frío (azulado)

Temperatura más baja → Temperatura de color cálido (rojizo)

El sol actúa como una especie de cuerpo negro, cuya radiación solar en el camino se ve afectada, de manera que lo que se ve fuera de la tierra es diferente que en la superficie.

8.3. Sistema visual humano

Tres tipos de conos: S (corto), M (medio), L (largo). Cada uno responde a diferentes rangos de longitudes de onda. Estos se alinean bastante bien con el azul, verde y rojo.

Nuestro cerebro con 3 señales interpreta cualquier color que podamos percibir.

Las aves tienen cuatro tipos de conos, uno de ellos detecta la luz UV

Puede ocurrir que percibamos igual diferentes espectros, mezclas de colores.

- La luz espectral es monocromática, y no se puede descomponer en otros.
- Luz resultado de mezclar colores, se puede con un prisma descomponer en los colores que lo componen.

8.4. Caracterización del Color

Cada color del espectro puede describirse en función de su frecuencia de onda (f, medida en hercios) o su longitud de onda (λ, medida en nanómetros), que están relacionadas por $c = \lambda f$.

Luz monocromática o Color espectral: está formada por una sola longitud de onda (colores del arcoíris). Ejem. El rosa no existe como onda.

Fuente de luz blanca (sol) emite todas las frecuencias. Al incidir sobre un objeto, este absorbe algunas longitudes y refleja otras.

La combinación de longitudes de onda reflejadas determina el color (tono, matiz) del objeto

Desde el punto de vista de nuestra percepción, un color se caracteriza con 3 parámetros (**modelo HSB**/HSL/HSV):

- **Hue** (Tono, matiz, color): El color, longitud de onda predominante, se puede discriminar aproximadamente entre 150 tonos.
- **Saturation** (Saturación): Es el grado de pureza del color observado: va desde el color puro al gris. Los menos saturados contienen más luz blanca.
- **Brightness**, Lightness, Value (Brillo): Intensidad de la luz que se percibe. Cuanta energía emite.

  Intensidad - energía radiante que se emite por unidad de tiempo

Podemos distinguir del orden de 7 millones de colores

Colores primarios: Aquellos que no pueden obtenerse mediante la mezcla de ningún otro.

- Colores aditivos - al añadirse generan el blanco. Azul, verde y rojo.
- Colores substractivos - filtros que absorben la luz. Cian, amarillo y magenta.

Colores complementarios: Combinados producen luz blanca. Rojo-cian, verde-magenta, azul-amarillo

Se pueden combinar fuentes de luz de color con distinta intensidad para, seleccionando adecuadamente las intensidades, generar un determinado rango de colores.

No se puede combinar un conjunto finito o real de colores primarios para **generar todos los colores visibles** posibles

Modelos de color emplean **3 colores primarios para obtener la gama de colores (gamut)** asociada a ese modelo.

8.5. Estandarización

Necesidad de estandarizar los colores percibidos. En principio, con tres valores podríamos representar cualquier color.

Color matching experiment: Mezcla de tres colores primarios (ej. azul, verde y rojo del modelo RGB) para generar todos los colores del espectro visual.

- Combinar colores puros RGB para obtener un cierto color espectral.
- Se hace para todos los colores del espectro y con diferentes personas.
- Lo que si varía de los colores base es la intensidad de los colores (ya que el matiz es propio de la longitud de RGB)
- No todos los colores del espectro pueden ser igualados mediante la mezcla de R, G y B como el cian de 500 nm, pero si se le puede restar rojo.
- **Resultado:** Para cada color del espectro, obtenemos los valores RGB necesarios.

Estos experimentos son la base de la ciencia del color. Dieron lugar al espacio de color **CIE 1931** establecido por la **Comission Internationale de l'Eclairage (CIE)**.

8.6. Colores Primarios Estándar - Modelo CIE XYZ

La Comisión Internacional sobre Iluminación, CIE (1931) define un estándar de tres colores imaginarios aditivos que no necesiten valores negativos.

Se hace una transformación lineal del espacio R, G, B en el espacio X, Y, Z (vectores en espacio 3D de color aditivo) para que todos los valores de estos nuevos colores primarios sean positivos. $\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = M \begin{pmatrix} R \\ G \\ B \end{pmatrix}$. Combinando los colores imaginarios XYZ se puede generar todo el espectro.

Se eligen los primarios de forma que Y es igual a la luminosidad.

8.7. CIE 1931: diagrama de cromaticidad (2D)

Colores imaginarios XYZ: no son RGB, XYZ son combinaciones lineales de los valores de RGB para que todos los valores sean positivos.

Para hacerlo más manejable, se proyectará el espacio XYZ (que es 3D) en un plano xy → esto es el diagrama de cromaticidad (también llamado CIE xyY).

Esto se consigue normalizando los valores X, Y, Z, para que su suma sea 1 ($x + y + z = 1$). $x = \frac{X}{Z+Y+Z}$ $y = \frac{Y}{Z+Y+Z}$ $z = \frac{Z}{Z+Y+Z}$.

El color puede dividirse en dos partes: brillo y cromaticidad.

- Y es una medida del brillo o luminosidad de un color. Máxima en el amarillo verdoso.
- La cromaticidad se determina a través de los dos parámetros derivados x e y (dos de los valores de X, Y, Z).

8.8. Diagrama de Cromaticidad

Proyección del plano $x + y + z = 1$ en el plano XY.

- A partir de la cromaticidad x e y (fija relación entre colores) y la luminancia Y, se obtienen los valores mediante: $X = x\frac{Y}{y}$ y $Z = z\frac{Y}{y}$

El diagrama representa toda la cromaticidad percibida por el ojo humano

- Aplicaciones: Nombrar y normalizar colores, Mezcla de colores, Identificar colores complementarios, Determinar la frecuencia dominante y Comparar gamut de dispositivos.
- Propiedades
  - Los colores del borde son: **Colores espectrales puros (spectral locus)**
  - La línea recta es: **La línea del púrpura (purple boundary)**
  - La suma de dos colores se encuentra en la línea que los une
  - El blanco se encuentra en $x = 1/3$, $y = 1/3$. La línea que une dos colores complementarios pasa por ese punto.
- Se puede representar cualquier color con:
  - los distintos valores del **triestímulo XYZ**
  - o con los valores de **luminancia Y y de cromaticidad x, y**

Mezcla de colores

- Mezclando los **colores I, J** se pueden obtener toda la **gama** situada en el **segmento** $\overline{IJ}$.
- Con los colores **I, J, K se obtienen los colores del triángulo inscrito (gamut)**.

Color blanco

- El blanco es una mezcla de todos los colores.
- Se define un blanco estándar, blanco de igual energía.
- Pero hay muchos colores que pueden ser considerados blancos.
- CIE estandariza los diferentes blancos.
- El blanco D65 es el que emitiría un cuerpo negro a 6500 K.

9. ILUMINACIÓN Y SOMBREADO

9.1. Rendering

Generar una imagen 2D a partir de un modelo 3D,

- **Analíticas**, mediante ecuaciones.
- Aproximaciones mediante **mallas de triángulos**.

Dos maneras de convertir el modelo 3D en una imagen 2D:

- **Ray tracing (trazado de rayos):** simula el proceso físico de la propagación y reflejo de la luz desde la **fuente hasta la cámara** (aunque **lo hace al revés**, de la **cámara hasta la fuente de luz**)
  - Para cada pixel: Envía un rayo y Comprueba intersección.
  - Más realista, pero más lento.
- **Graphics pipeline:** proyecta vértices/triángulos «en bloque» (de manera paralela) sobre los píxeles de la imagen. Tarjetas gráficas.
  - Para cada objeto/triángulo: Proyectarlo a la pantalla.
  - Utiliza GPU, es más rápido, pero menos realista.

9.2. Ray-tracing

9.2.1. Algoritmo

Para cada pixel

1. Lanza un rayo a la escena
2. Encuentra el punto de intersección p más cercano que interseccione con el rayo.
3. Calcula lighting/sombra para p, $[A + D + S]$, colp.

$$I = R_a \cdot L_a + \Sigma_i s_i \left(\frac{R_d \cdot \text{máx}(n \cdot l, 0) \cdot L_d}{a + bd + cd^2} + \frac{R_s \cdot \text{máx}(cos(r, v)^n, 0) \cdot L_s}{a + bd + cd^2} \right)$$

4. Si la superficie es un espejo, se lanza un rayo de reflexión, $(1 - r)colp + r \cdot refCol$, refCol.
5. Si además la superficie es refractante, se lanza un rayo de refracción, $(1 - (r_1 + r_2)) \cdot colp + r_1 \cdot refCol + r_2 \cdot refrCol$, refrCol.

9.2.2. Ray-casting (primera iteración)

En el mundo real:

- Las fuentes luminosas emiten fotones
- Las superficies reflejan y absorben fotones
- Las cámaras registran fotones

Consideraremos que los fotones son rayos (líneas) de luz. Ignoraremos la naturaleza ondulatoria de la luz

Rayo

Se definirá a un rayo como un punto del que parte un vector.

- $Ray = (Point, Vector) = (S, V)$
- Point es el punto de vista (cámara), Vector es la dirección en la que mira la cámara.
- La forma paramétrica de un rayo lo expresa como una función escalar t. Para cada t se proporciona un punto por el que el rayo pasa: $P(t) = S + tV, 0 \leq t \leq \infty$

Rayos primarios: Aquellos que parten del observador (V).

Rayos secundarios: Aquellos rayos generadores para simular sombra, reflexión (E: espejos) y refracción (T: transmisión) en el punto de intersección del rayo con la superficie.

Fuentes de luz

Cada tipo emite fotones de manera diferente.

- **Fuentes de punto (Bombillas):** Emiten luz en todas las direcciones.
- **Fuentes direccionales (el Sol):** Están muy lejos y se puede suponer que los rayos llegan en paralelo.
- **Spot Lights (Focos):** Como las bombillas, pero la intensidad varía con la dirección. Hay zonas donde no se emite luz.
- **Luces de techo (o de área):** Emiten no desde un punto sino desde un área, conjunto de puntos de luz.

Superficies

- Reaccionan a los rayos: Absorbiéndolos, Reflejándolos, Refractándolos, Atenuándolos, Fluorescencia, etc.
- Tipos materiales: Mate, Brillante, Rugoso, Suave, Transparente, Translúcido, etc.

Cámaras: Transformación de una imagen 3D a una 2D. El punto por el que pasa la luz, como un ojo o una cámara fotográfica.

Tipos de Iluminación

Iluminación indirecta: Aquella que proviene de otro objeto que no emite luz. En trazado de rayos, nos ceñimos a la iluminación directa.

Iluminación directa: La que viene de una fuente que emite luz.

El problema de implementar el modelo directamente, es que el programa de rendering sería muy inestable, porque muchos de los rayos emitidos no alcanzarían la cámara. Es por esta razón por la que se hace al revés, en vez de emitir los rayos desde la fuente de luz, hagamos que la cámara los emita.

Para cada pixel

1. Lanza un rayo a la escena.
2. Encuentra el punto de intersección con el objeto más cercano que interseccione con el rayo.
3. Colorea/ilumina (lighting) el pixel con las propiedades de color del punto de intersección teniendo en cuenta la fuente de luz. Para mostrar espejo se necesitan varias iteraciones.

La esfera

Solo consideraremos un tipo de superficie, la esfera (hay otras: planos, conos, ...)

- Ecuación implícita de una esfera centrada en (x_0, y_0, z_0).
- $r^2 = (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2$
- Si está centrada en el origen: $r^2 = x^2 + y^2 + z^2$

Intersección rayo-esfera

Utilizando las ecuaciones de esfera ($r^2 = x^2 + y^2 + z^2$) y rayo ($P(t) = S + tV$) tenemos $r^2 = (S_x + tV_x)^2 + (S_y + tV_y)^2 + (S_z + tV_z)^2$, lo que queremos es encontrar el t que haga toda la expresión 0.

- $0 = t^2(V_x^2 + V_y^2 + V_z^2) + t(2S_xV_x + 2S_yV_y + 2S_zV_z) + (S_x^2 + S_y^2 + S_z^2 - r^2)$ y resolvemos con $t = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$.
- Tres casos: Sin intersección ($b^2 - 4ac < 0$), 1 intersección en t ($b^2 - 4ac = 0$) o 2 intersecciones t más cercanas ($b^2 - 4ac > 0$).

Este proceso se puede hacer para otros tipos de objetos si conocemos sus ecuaciones implícitas.

- En el caso del triángulo, calculamos la intersección entre el rayo y el plano que contiene el triángulo, comprobando si la intersección p está dentro del triángulo.
- Coordenadas baricéntricas: Cualquier triángulo de vértices A, B y C se puede expresar como $P = wA + uB + vC$. Si $0 \leq w, u, v \leq 1$, P está dentro del triángulo. Si no está fuera.

Iluminación (Lighting)

Interacción luz-superficie.

- **Reflexión difusa:** la luz se difunde en todas direcciones.
- **Reflexión especular:** la luz se difunde en una dirección preferente. El efecto es brillos.
- Se excluye de momento: espejos, refracción y translúcidos.

Espectrograma: La luz contiene energía en muchas frecuencias y la interacción luz-superficie podría ser distinta para cada frecuencia.

- **Simplificación:** usaremos un modelo de color RGB, cada color se puede definir como una mezcla de rojo, verde y azul. $L_m = R + G + B$

Simplificaciones, para hacer los cálculos tratables.

- Se utiliza el color RGB.
- Solo luz directa.
- En principio la fuente «de punto», aunque se puede extender a otros.
- Superficies con reflexión especular y difusa.

Modelo de reflexión de Phong

$I = I_a(R_a, L_a) + I_d(n, l, R_d, L_d, a, b, c, d) + I_s(r, v, R_s, L_s, n, a, b, c, d)$

$R_x = (R_R, R_G, R_B)$ es la reflectividad de la superficie y $L_x = (L_R, L_G, L_B)$ la intensidad de la luz.

Usa 4 vectores para computar el color de cada punto p

- Normal (n), Luz (l), Punto de vista (v) y Reflexión (r).

3 componentes.

- **Ambiente (uniforme):** es una manera de incluir una pseudo-iluminación indirecta. Misma luz de todas las partes, uniforme.

  $I_a(R_a, L_a) = R_a \cdot L_a$

- **Reflexión difusa:** Sigue la **Ley del coseno de Lambert**.

  El ángulo de incidencia del rayo de luz afecta la intensidad perceptible por el ojo. Si la **superficie es perpendicular al rayo, captura más cantidad de luz** que si es oblicua. La ley de Lambert afirma que es proporcional al coseno del vector de luz y la normal.

  La luz parece atenuarse con la distancia d entre la fuente de luz y la superficie. Se usan las constantes a, b y c, que son definidas por el usuario.

  $I_d(n, l, R_d, L_d, a, b, c, d) = \frac{R_d \cdot \text{máx}(n \cdot l, 0) \cdot L_d}{a + bd + cd^2}$, con $n \cdot l = |n| * |l| * \cos(n, l)$, **producto escalar** en ambas reflexiones.

- **Reflexión especular:** Se usa r y v, porque la reflexiones más fuerte en la dirección del vector de reflexión. Es máximo cuando coinciden ambos vectores.

  Se utiliza un factor n de intensidad de reflexión (hardness), cuanto mayor es menos intensidad de reflexión, puesto que el coseno es un valor entre 0 y 1.

  $I_s(r, v, R_s, L_s, n, a, b, c, d) = \frac{R_s \cdot \text{máx}(cos(r,v)^n, 0) \cdot L_s}{a + bd + cd^2}$

L_d suele ver igual a L_s. Y la atenuación suele ser la misma (a, b, c)

La suma de esos tres componentes define el color e intensidad en un punto de la superficie.

Múltiples fuentes de luz:

$$I = I_a(R_a, L_a) + \Sigma_i(I_d(n, l_i, R_d, L_{di}, a, b, c, d_i) + I_s(r_i, v, R_s, L_{Si}, n, a, b, c, d_i))$$

Sombras

Son causadas por un objeto interpuesto entre la superficie y la fuente de luz.

Hemos hecho ray-casting, y hemos visto que el **rayo intersecciona con la superficie**.

Ahora queremos iluminar el punto. Lo cual incluye **comprobar si está en la sombra**.

Podemos comprobarlo **emitiendo un rayo desde el punto hasta la fuente de luz**.

- Si se detecta intersección, puede que el punto esté en sombra.
- El punto está en sombra si el valor t de la intersección es menor que el valor t del punto de luz.
    - Luces direccionales, al estar en el infinito, cualquier intersección genera una sombra (t positivo).
    - Focos, hay que asegurarse que el rayo de sombra está dentro del cono de luz del foco, pero como la bombilla (fuente de punto)
    - Luces de área, las más complicadas. Hay que trazar, no uno, sino infinitos rayos de sombra, y sumar. Aparecen zonas de penumbra, sombra para una parte pero para otra no.

Modelo de Phong con sombra

$I = I_a(R_a, L_a) + \Sigma_i s_i \cdot (I_d(n, l_i, R_d, L_{di}, a, b, c, d_i) + I_s(r_i, v, R_s, L_{Si}, n, a, b, c, d_i))$

- s_i es 0 si luz bloqueada en ese punto y 1 si luz no bloqueada en ese punto. Si el objeto interpuesto es semitransparente y deja pasar algo de luz $0 \leq s_i \leq 1$.

9.2.3. Reflexión

El lighting de Phong ya tenía un componente para la luz especular (el cálculo de brillos/-reflejos), usando el vector de reflexión.

Para el reflejo en espejos, añadiremos un nuevo término a la fórmula de lighting, y usaremos la reflexión del vector del punto de vista para la nueva dirección del rayo.

e = vector reflexión del vector punto de vista v con respecto a la normal n. Mismo ángulo entre v y n que n con e.

Los puntos de un espejo no tienen propiedades de color intrínsecas, sino que son las del objeto reflejado.

$I = (1 - r) \cdot [Ambiente + Difusión + Especular] + r \cdot (refColor)$

- Se añade el término $r \cdot (refColor)$.
    - r es un valor en [0,1] que mide la reflectividad.
    - refColor es el color del rayo de reflexión e.
    - Se repite el cálculo para el rayo e, es recursivo el método, de manera que se calcula el color con los sucesivos cálculos.

El valor de refColor

- Se lo trata como cualquier rayo de la cámara.
- Comprueba si interseca con algún objeto
    - SI: iluminar normalmente y si es necesario, reflejarlo otra vez.
    - NO: devuelve el color de fondo.

Cuando solo se hace un paso de cálculo de color es Ray-casting, pero cuando sobre ese primer cálculo se repiten teniendo en cuenta los previos de manera recursiva es Ray-tracing.

Cuantas veces se debe hacer esta recursión o profundidad de algoritmo lo determina:

- Establecer un límite de profundidad (depth)
    - Los rayos desde la cámara tienen una profundidad de 0.
    - Cada rayo hijo incrementa la profundidad en 1, por lo que no se lanzan más rayo si se supera el límite.
- No lanzar más rayos si el coeficiente de reflexión es 0.

9.2.4. Refracción

La refracción es parecida a la reflexión

Cuando un rayo choca con una superficie, la ilumina según el algoritmo estándar y comprueba si hay que emitir un rayo de refracción.

- SI: calcula el nuevo rayo. Ilumínalo de la manera estándar y añade un término adicional a la ecuación de iluminación

Hay que almacenar índices de refracción para los materiales

Si el objeto es transparente, lanza un nuevo rayo usando la ley de Snell: $n_1 \sin \alpha_1 = n_2 \sin \alpha_2$

- Las n son los coeficientes del medio y α los ángulos de intersección con la normal.

9.2.5. Trazado de rayos recursivo (Whitted)

Ventajas:

- Gran realismo.
- Lighting por iluminación directa (iluminación local).
- Cálculo de Sombras.
- Incluye cierta iluminación indirecta (espejos y refracción), no en objetos difusos (paredes, techas).
- Incluso eliminación de superficies ocultas.

Limitaciones:

- Lento (1 rayo por pixel)
- Es susceptible a problemas de precisión numérica, provocando diferencia entre píxeles contiguos
- Rayos de luz L no se refractan en su trayectoria hacia la luz (pasan a través de los objetos sin modificar el rayo, solo se modifica al volverse al lanzar desde la superficie)
- No incluye iluminación indirecta (o iluminación global) aparte de espejos y refracción. No incluye la luz que rebota en objetos de reflexión difusa.

No tiene iluminación indirecta (luz reflejada por superficies), utiliza el componente de ambiente.

9.2.6. Modelos de rénder para iluminación global

Path tracing

Similar a Ray Tracing, pero trazando todos los caminos, no solo los de espejos y refracción, sino también cuando la luz choca y rebota en superficies difusas o especulares.

Radiosidad (Radiosity)

Radiosidad: Tasa con la que la energía (emitida + reflejada) parte de una superficie. $tasa = \frac{energía\ por\ unidad\ de\ tiempo}{unidad\ de\ area}$

Discretiza la **escena en patches (trozos)** y supone que **cada patch puede ser fuente de luz y/o reflejar luz de cualquier otro patch** en la escena. **Calcula la contribución de cada porción** a la iluminación de las restantes.

Tiene en cuenta **solo la reflexión difusa** (superficies mates). La cantidad de luz reflejada por un reflector difuso no depende de la dirección desde la que la observemos. Por tanto, el procesamiento que hace **radiosity es independiente del punto de vista** (a diferencia de Ray Tracing).

Shading por **interpolación para que no haya cambios demasiado bruscos** en patches adyacentes

Después habrá que generar la escena desde algún punto de vista y determinar cuáles son las superficies visibles

Ecuación de Radiosidad: $B_i = E_i + \rho_i \Sigma_{i \leq j \leq n} F_{i-j} B_j$

- B_i radiosidad del path B_i
- E_i tasa con que B_i emite luz (si B_i es una fuente de luz directa)
- ρ_i reflectividad del path B_i
- $F_{i-j} B_j$ radiosidad que alcanza B_j desde B_i. Fracción de energía que parte de j y alcanza i por completo.
- F_{i-j} factor de forma que conecta los patches i y j. Se calcula para cada pareja de patches, que depende de la orientación relativa entre ellos. Será 0 si no llega la luz.

Equivalente: $B_i - \rho_i \Sigma_{i \leq j \leq n} F_{i-j} B_j = E_i$.

Matricialmente: $(I - \rho F)B = E$, luego $B = (I - \rho F)^{-1} E$

Habiendo computado los factores de forma, se puede resolver el sistema. Es **muy lento cuando hay muchos patches**.

Para agilizar el proceso se puede usar otra estrategia, mediante soluciones parciales y aplicando un refinamiento progresivo.

1. Se colecta la energía del ambiente sobre el path i.
   $B_i^{k+1} = E_i + \rho_i \Sigma_{i \leq j \leq n} F_{i-j} B_j^k$
2. Se irradia la energía del patch i a todo el ambiente.
   $For(j = i : n) \;\; B_j^{k+1} = B_j^k + \rho_j F_{j-i} \Delta B_i$

9.3. Graphics pipeline

Se ejecuta en hardware GPU (Graphics Processing Unit), hace una ejecución paralela masiva de cada una de las etapas. Cada triángulo o vértices, se proyecta sobre los píxeles.

Se llama tubería porque, por un lado, **entra el modelo** como geométrica, y por el otro, **salen los píxeles** de la imagen.

El proceso que ocurre entre medias es:

1. **Procesamiento de vértices (Vertex shader)**

   - Transformaciones (translación, rotación, escalado, ...) de sus coordenadas originales a las que tenga que ocupar.
   - Lighting solo en vértices.
   - Cambio al sistema de coordenadas de la cámara (x, y), para que se pueda ver lo que queremos antes de proyectar.
   - Proyección a 2D (pantalla).

   Es un programa escrito por el usuario que toma un vértice y devuelve un vértice transformado (proyectado sobre la pantalla). Se ejecuta en paralelo para cada vértice en la geometría.

   Vector normal en el vértice, la malla de polígonos aproxima una superficie, se calcula el vector normal unitario promedio en cada vértice del polígono como el **promedio de las normales de polígonos con ese vértice**. $n = \frac{\Sigma_i n_i}{|\Sigma_i n_i|}$

2. **Rasterización**

   Convierte los vértices en la figura completa, en la pantalla, en forma de píxeles.

   Hay que **calcular que píxeles cubre un polígono**. Aunque en lugar de llamarlos **píxeles, les llamamos fragmento**, porque un pixel es un fragmento que se puede ver y pueden estar ocultos los fragmentos.

   Por fuerza bruta:

   - Para cada pixel:
     - Calcula las ecuaciones de la línea para el centro del pixel.
     - Si todas dan >0, el pixel está dentro (en sentido de estar dentro).
   - Si el triángulo es pequeño, se pierde tiempo.
   - Mejora:
     - Hacer los cálculos solo para los puntos dentro de la caja que contiene el triángulo. Aunque todavía pierde tiempo en píxeles externos.
     - Rasterizar los lados y rellenar.

3. **Procesado de fragmentos (Fragment shader)**

   Shading/Interpolado. Colorea/ilumina cada uno de los fragmentos de un polígono.

   Método de shading: Constante (rápido), Gouraud (medio) y Phong (lento).

   - **Constante (flat/facet shading):** Usa el lighting de un vértice (o centroide) para todos los fragmentos del triángulo.

     Es muy rápido, pero para conseguir escenas realistas, las caras de los objetos deben estar formadas por polígonos muy pequeños.

     Funciona razonablemente bien cuando: Fuente luminosa muy lejos, Observador muy lejos o Polígono representa la superficie real modelada (no aproximación a una superficie curva).

   - **Gouraud shading:** Interpolación de intensidades. El color de cada punto p se puede calcular como la interpolación de los colores de los vértices (I1, I2, I3). Cada vértice tiene una componente RGB.

     - **Interpolar bi-linealmente** las intensidades de los vértices para hallar intensidades sobre cada pixel. Hace la media ponderada según la cercanía a los vértices.
       $I_a = (1-\alpha)\cdot I_1 + \alpha \cdot I_2 = I_1 - (I_1 - I_2)\cdot\alpha$. α es la proporción de distancia entre nuestro punto a un vértice y la distancia entre los dos vértices sobre uno de los ejes.
       Si queremos un punto interior calculamos los I de los extremos en un eje y los usamos para el otro eje. Línea horizontal con I_a e I_b hallados con la y, y mediante estos dos en la x calculamos el punto entre I_a e I_b, I_p.

       $$I_a = I_1 - (I_1 - I_2)\cdot\frac{y_1 - y_s}{y_1 - y_2}; I_b = I_1 - (I_1 - I_3)\cdot\frac{y_1 - y_s}{y_1 - y_3}; I_p = I_b - (I_b - I_a)\cdot\frac{x_b - x_p}{x_b - x_a}$$

     - **Interpolación baricéntrica**, computa las coordenadas (α, β, γ) del punto p, y las usa para interpolar I_1, I_2 y I_3.
       Los valores indican la influencia de cada vértice para calcular nuestro punto.
       $0 \leq \alpha, \beta, \gamma \leq 1$

       $$I_p = \alpha I_1 + \beta I_2 + \gamma I_3$$

   - **Phong shading:** Para que los objetos con superficies curvas aproximadas por triángulos se visualicen bien, se pueden interpolar las normales de los vértices.

     De esta manera, las normales de las caras tienen influencia por su proximidad a los vértices, que son el resultado de interpolarse con las de las caras previamente.

     Computacionalmente, más costoso, ya que tiene que recalcular las normales y posteriormente normalizarlas.

Interpolar bi-linealmente las normales de los vértices en la superficie del polígono (también **interpolación baricéntrica**).

$$n_a = n_1 - (n_1 - n_2) \cdot \frac{y_1 - y_s}{y_1 - y_2}; n_b = n_1 - (n_1 - n_3) \cdot \frac{y_1 - y_s}{y_1 - y_3}; n_p = n_b - (n_b - n_a) \cdot \frac{x_b - x_p}{x_b - x_a}$$

$$n_p = \alpha n_1 + \beta n_2 + \gamma n_3$$

4. **Testing and blending**

   Eliminación de superficies ocultas.

   Blending: Combinar los diferentes fragmentos que hay para cada pixel. En principio, solo el fragmento más cercano (a la cámara) se va a ver en el pixel, pero se pueden tener en cuenta otros efectos como transparencia.

   Z-buffering: Se guarda el z para todos los fragmentos. s y t son los x e y proyectados. Además del buffer de color, se guarda un buffer de profundidad depth (del mismo tamaño y resolución)

   Si $fragment.z < depth[s,t]$, el fragmento está delante que el previo. Entonces se actualiza $depth[s,t] = frament.z$ y se pone el color $color[s,t] = fragment.color$.

Los shaders son las partes programables, las escritas por el usuario en lenguajes como OpenGL y ejecutados en la GPU. Son el Procesamiento de vértices y de fragmentos.

Las otras 2 partes, Rasterización y Test and blending, son fijas.

9.4. Aumentar el nivel de detalle de un objeto

Solución cara, dar más detalle al modelo.

Ventajas: Se incorpora como una parte de objeto.

Desventajas:

- Las herramientas de diseño no incorporan facilidades para añadir detalle.
- Los objetos tardan más en generarse en la escena.
- Los objetos ocupan más espacio en memoria.
- Los detalles muy complejos no pueden reusarse en otros objetos.

9.5. Texturas

Mapear una textura en el modelo, pegar una imagen «bitmap» sobre una superficie para dotarla de detalle. Una imagen no aumenta el número de polígonos.

Ventajas:

- Pueden reusarse en otros objetos.
- Ocupan memoria, pero pueden compartirse y comprimirse.
- Los algoritmos de mapeo son rápidos.
- Las texturas no afectan a la geometría de los objetos.

Tipos de texturas:

- Unidimensionales
- Bidimensionales, las más comunes, suelen ir en formato BMP, GIF o JPG. Llevan información RGB y transparencia opcional.
- Matemáticas, creadas mediante algoritmos.

Tenemos 3 espacios:

- Espacio de texturas, 2 dimensiones, la imagen.
- Espacio de objetos, 3 dimensiones, los polígonos sobre los que se pone.
- Espacio de imagen, 2 dimensiones, la proyección en pantalla.

Los **texels** son los píxeles de la textura.

Se definen mapeos que cubren el objeto en 3D con la textura, el cual, después, es proyectado al 2D de la imagen en la pantalla.

Mapeo: Normalmente, se va en sentido contrario, para cada fragmento, determinar que texels le corresponden.

9.5.1. Cálculo de texel (texture filtering)

Incluso aunque el pixel (fragmento) y el texel tengan tamaño parecido, no tienen por qué estar alineados.

Hay que elegir como calcular el pixel en función de los texels: Vecinos más cercanos o Interpolación (promediado, se pondera) de los texels vecinos. Funciona mejor la interpolación.

Para el cálculo se supone que píxeles y texels están en el centro del pixel y texel.

Magnification, un texel para varios píxeles, y **Minification**, varios texels para un píxel.

Mip-mapping

Guardar las texturas en diferentes resoluciones/tamaño. Usar la de tamaño más apropiado, o incluso interpolar entre los dos tamaños más cercanos.

- 512x512, 256x256, 128x128, 64x64, 32x32, 16x16, 8x8, 4x4, 2x2, 1x1

Tipos de mapeos

- **Mapeado plano:** Limitadas aplicaciones. Se ignora z.

  $(u, v) = (x, y)$

- **Mapeado cúbico:** Variación del anterior, pero con 6 texturas, 1 por cara. Mapea cada textura a un lado del objeto como si fueran planos.

- **Mapeado esférico:** Usado para mapeos de entorno. Se pone la textura como una esfera y en función del ángulo con cada eje tiene unas coordenadas.

  $(u, v) = (\phi/2\pi, (\pi - \theta)/\pi)$

- **Mapeado cilíndrico:** Se envuelve la textura alrededor del objeto. El mismo color para todos los puntos en el mismo ángulo.

  $(u, v) = (\phi/2\pi, y)$

- **Mapeado U, V:** Más realista. u, v determinan un punto en el espacio, se coloca ahí el centro de la textura y se estira o encoge para adecuarse al polígono

  Simplifiquemos para un triángulo en 3D.

  Cada vértice tiene una coordenada (x, y, z) y cada vértice se asocia unas coordenadas de la imagen/ textura (u, v).

  En los puntos interiores se rellena interpolando, por ejemplo con la baricéntrica. Aunque está la cuestión de asignar vértices a puntos en textura (unwrapping).

Texturas de Normales (Bump Mapping)

La textura contiene vectores normales en lugar de colores. Al mapear esta textura, sobre una superficie, el vector normal de esta se modifica según el valor encontrado en la textura.

Efecto de relieve detallado (aunque el objeto original sea plano), debido a los efectos de iluminación generados por la variación del vector normal.

A veces se define como textura 3D virtual, pues aporta información en el eje Z.

En cada punto de la superficie: Mira el punto correspondiente en la textura y perturba la normal usando el color del punto de la textura.

Desventajas

- Muy lento (sin aceleración hardware).
- Permite crear un efecto de iluminación, pero no hace variar la forma real de la superficie. Aunque parece deformarse, la superficie y sus polígonos no cambian, solo es visual.