Auxiliar Pre-Control 2

Profesora: Nancy Hitschfeld y Valentín Muñoz
Auxiliares: Julieta Coloma y Catalina Gajardo

Resumen

Iluminación

- Se resuelve en el fragment shader (Pixel Processing)
- Modelo Phong

Ambiental Difusa Especular

- Luz difusa: $I = \max(0, N \cdot L)$, donde N es la normal y L es la dirección de la luz.
- Luz especular: $I = \max(0, V \cdot R)^s$, donde V es el view vector, R es el vector reflejado y s es el shininess

Texturas

- **Magnificación**: Cuando la textura es mas pequeña que los pixeles que cubre.
- **Minificación**: Cuando la textura es muy grande para la superficie que cubre.

Hay dos formas en que podemos determinar el color que seleccionamos si la imagen no calza en resolución: `GL_NEAREST` o `GL_LINEAR`. Nearest selecciona el color del pixel más cercano. Vs Linear que hace un promedio de los colores cercanos.

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
```

- **Mip-Mapping**: Técnica donde se pre-generan texturas de distintos tamaños. Esto se utiliza para casos donde la textura es mucho más grande que el pixel. Esto es un problema ya que OpenGL tiene dificultades para escoger el color correcto ante tantas opciones en la gran textura.

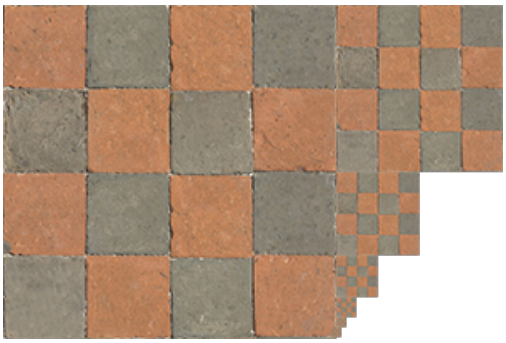


Figura 2: Ejemplo de Mipmap

También hay varias configuraciones:

- `GL_NEAREST_MIPMAP_NEAREST` escoge el mipmap más cercano y luego usa la interpolación nearest para escoger el color.
- `GL_LINEAR_MIPMAP_NEAREST` escoge el mipmap mas cercano pero usa la interpolación nearest.
- `GL_NEAREST_MIPMAP_LINEAR`
- `GL_LINEAR_MIPMAP_LINEAR`

Física

- Sistemas de partículas: Control sobre muchos objetos iguales con las mismas reglas.
- Animación basada en física: Aplicación de conceptos físicos y realistas sobre objetos en movimiento o sujetos a fuerzas.
- Simulación de cuerpos rígidos: Se reduce al análisis de objetos que no se deforman, se utilizan momentum, inercia y ecuaciones simples de movimiento para calcular posiciones en el tiempo.

Colisiones

Métodos de detección

- **A priori**:
  - Predice colisiones antes de que ocurran.
  - Requiere conocer velocidades y trayectorias.
  - Más precisa, pero más compleja.
- **A posteriori**:
  - Detecta colisiones después de que ocurren.
  - Más simple, pero puede generar problemas visuales si no se corrigen bien.

Primitivas geométricas usadas

- Cajas (AABB u OBB)
- Esferas
- Envoltura convexa (Convex Hull)

Mallas

Estructuras

- **Triangle Soup**: Se guarda una lista de vértices, sin conectividad. Es redundante y poco eficiente para recorrer vecindarios.

- **Representación por caras**: Se define cada vértice una sola vez, y las caras como índices. No hay conectividad entre caras ni vecindario.
- **Representación por aristas**: Cada arista conoce sus vértices, caras y aristas vecinas. Permite recorrer vecindarios linealmente.
- **Half-Edge (Sub-aristas)**:
  - Cada arista se divide en dos half-edges.
  - Cada half-edge conoce:
    - El vértice desde el que parte.
    - Su cara asociada.
    - Su twin (opuesto).
    - La half-edge siguiente y la anterior.

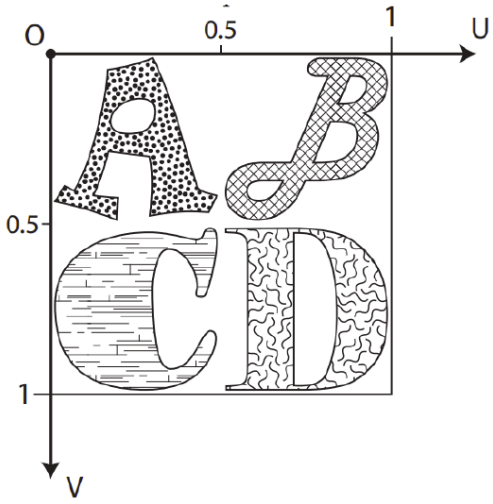
Operaciones sobre mallas

- **Subdivisión**: Agrega detalle dividiendo los triángulos.
- **Simplificación**: Reduce el número de triángulos la forma.
- **Regularización**: Mejora la calidad de la malla (triángulos mas equiláteros).

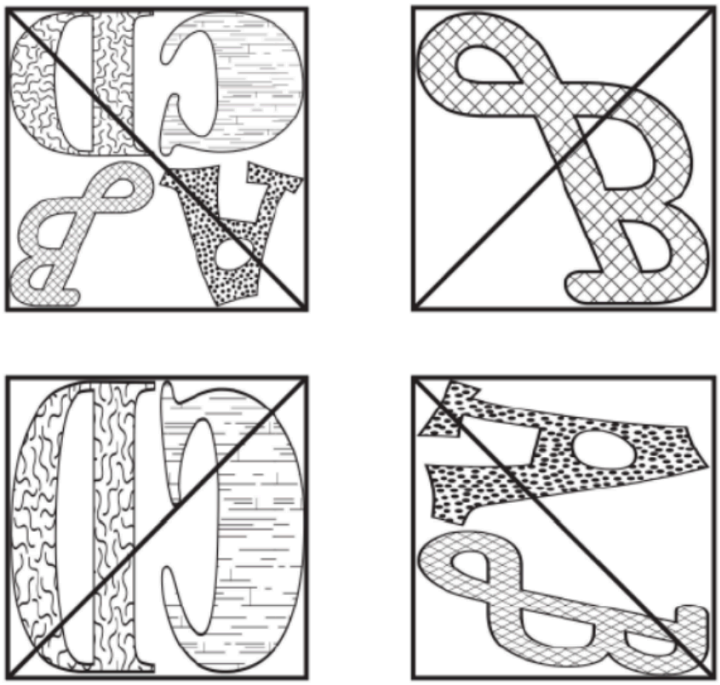
Problemas

P1 (P2 C2 2023-2) Textura

Considera la textura ilustrada abajo:



Se quiere aplicar esta textura a las cuatro mallas triangulares a continuación:



Para cada vértice de cada malla escribir las coordenadas de textura que se necesitan asociar para que se produzcan las imágenes mostradas. Escriba las coordenadas de textura como pares de valores al costado de cada vértice de la figura.

P2 (P1 C2 2022 Primavera) Iluminación

Cel-shading o *toon shading* es una técnica de sombreado que da una apariencia de dibujo animado a los objetos. La siguiente imagen ejemplifica la diferencia entre el sombreado de Phong y el sombreado toon. En resumen, el sombreado estilo dibujo animado tiene dos características: primero, utiliza dos tonos de color; segundo, se delinean los bordes del objeto dibujado con un contorno de color negro.

a. Defina una ecuación del modelo de *toon shading* (puede basarse en el modelo de iluminación de Phong) para las tonalidades de color.
b. Describa en pseudo-código el vertex program y el fragment program que implementa las tonalidades y el delineamiento (hint: el contorno es calculado en el fragment program).

P3 (P2 Examen 2022 Primavera) Mallas geométricas

Supongamos que tenemos un vértice v en una estructura *Half-Edge* y queremos dibujar el polígono indicado en la figura; es decir, un polígono cuyos vértices son los centros de los triángulos adyacentes a triángulos que contienen a v (en algún orden consistente, no importa si está a favor o en contra de las agujas del reloj). ¿Cuál de los siguientes códigos no extraen correctamente los vértices blancos marcados?

a.
```
P = []
he = v.he
P.append(he.next.twin.face.center)
he = he.twin.next

while (he != v.he):
  P.append(he.next.twin.face.center)
  he = he.twin.next
```

b.
```
P = []
he = v.he.next.next
P.append(he.next.next.twin.face.center)
he = he.twin.next.next

while (he != v.he.next.next):
  P.append(he.next.next.twin.face.center)
  he = he.twin.next.next
```

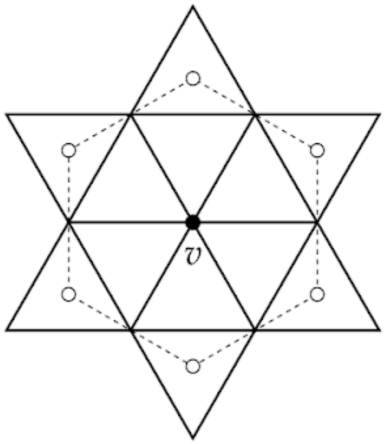
c.
```
P = []
he = v.he.next
P.append(he.twin.face.center)
he = he.next.twin.next

while (he != v.he):
  P.append(he.twin.face.center)
  he = he.next.twin.next
```

d.
```
P = []
he = v.he.next.twin
P.append(he.face.center)
he = he.twin.next.twin.next.twin

while (he != v.he.next.twin):
  P.append(he.face.center)
  he = he.twin.next.twin.next.twin
```

e. Todas las anteriores



P4 Colisiones

Tenemos la siguiente escena de un videojuego

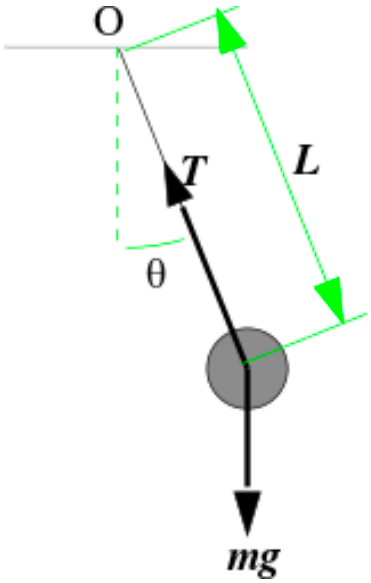
Para modelar correctamente las colisiones, se tienen disponibles las siguientes primitivas geométricas:

- AABB(name, min, max)
- Sphere(name, radius)
- ConvexHull(name, points)

a. Defina las colisiones necesarias para los distintos elementos de la escena anterior, utilizando las primitivas disponibles. Piense en que se quiere priorizar el rendimiento del videojuego en línea y mantener la dinámica típica de un shooter (como por ejemplo detectar impacto en distintas secciones del cuerpo)
b. Si el objetivo del juego es minimizar errores de colisión y evitar bugs visuales, ¿sería mejor implementar las colisiones a priori o a posteriori?Justifique su respuesta considerando la dificultad de implementación y el comportamiento esperado

P5 (P3 C2 2022 Primavera) Física

Tenemos un sistema de péndulo 2D como el siguiente:



Donde O es el origen de coordenadas y L es la longitud de la cuerda que sostiene a un cuerpo con masa. La fuerza que hace que el cuerpo quiera caer es mg donde m es la masa del objeto y g es la aceleración de la gravedad. Si uno conoce el ángulo del péndulo en algún momento dado, entonces la posición del objeto se calcula como

$$x = L\sin(\theta)$$

$$y = -L\cos(\theta)$$

El problema del péndulo se puede formular como una ecuación diferencial que depende del ángulo. La ecuación diferencial es la siguiente:

$$\theta'' = -\frac{g}{L}\theta$$

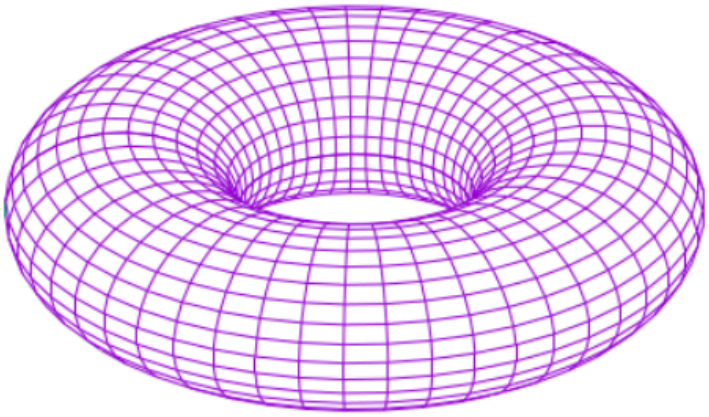
Escriba un algoritmo (o función de Python) que reciba como parámetros un ángulo inicial y genere la animación del péndulo. Use el método de Euler para resolver la ecuación y hacer su algoritmo.

Hint: la ecuación es de segundo grado, por lo que puede servir formular la ecuación como dos ecuaciones de primer grado. En su algoritmo puede asumir que existe una función que dibuja un objeto en una posición dada.

P6 Phong Faltante

Usted conoce el modelo clásico de iluminación llamado Phong✨. En esta ocasión se le pide dibujar y pintar la siguiente escena 3 veces:

1. Utilizando Phong pero sin el componente ambiental
2. Utilizando Phong pero sin el componente difuso
3. Utilizando Phong pero sin el componente especular



Propuestos

P? (P3-1 C3 2019 Otoño) Mallas geométricas

Dibuje 2 árboles de geometría sólido constructiva que generen la figura siguiente. Considere que usted dispone de las figuras básicas círculo y triángulo. Su árbol debe ignorar las transformaciones lineales básicas.