

TP Especial N°2: Ray Tracing

Bigio, Rafael Martín
Coffey, Santiago Andrés
Gregoire, Andrés Santiago

Introducción

En este trabajo se implementa un motor de *ray tracing* que permite representar escenas tridimensionales con un balance adecuado entre velocidad y calidad gráfica. El motor soporta luces puntuales, reflexión, refracción, *anti-aliasing* y penumbras. Además se utilizan *octrees* y el método de volúmenes envolventes de cajas para disponer los objetos en la escena eficientemente. Para la descripción de la escena se utiliza el formato X3D. Las primitivas de figuras implementadas son triángulos y esferas. El motor de *ray tracing* se implementó en Java, utilizando bibliotecas matemáticas de Java3D y bibliotecas de XJ3D para la carga de los archivos X3D.

Características implementadas

Cámara móvil

Los parámetros de la cámara se toman del (último) nodo Viewpoint del archivo X3D que describe la escena. Los atributos considerados son orientation (para la orientación de la cámara, o sea, vector del eje y rotación), position (para la ubicación en la escena) y fieldOfView (para el campo visual).

Anti-aliasing estocástico

Como estrategia de *anti-aliasing*, se utiliza la alternativa estocástica con un grillado de $N \times N$ a nivel de sub-píxel. Por cada celda de esta grilla se tira un rayo que se promedian para obtener el color del píxel final. N es el parámetro que se ingresa en la opción “as” de la línea de comandos. La ubicación del rayo dentro una celda de la grilla es aleatoria con distribución uniforme.

Luces puntuales

Se pueden ubicar múltiples luces en la escena a través del nodo PointLight del archivo X3D. Los atributos considerados en este nodo son location, color, intensity, on, radius y attenuation. El modelo de atenuación responde al polinomio de grado 2 de la especificación de X3D. Se utiliza el modelo de Phong para iluminación, tomando como nShiny el atributo shininess del material del objeto intersectado por el rayo multiplicado por 128 según el estándar de X3D.

Penumbra

Se puede generar el efecto de penumbras en las sombras con un parámetro que especifica la cantidad de rayos que se tiran desde cada luz puntual por cada píxel. Para esto las luces puntuales se modelan como esferas de un radio dado por el atributo radius del nodo PointLight.

Propiedades del material

Las propiedades del material para cualquier objeto son las tomadas del nodo Material (dentro de Appearance) de cada figura (nodo Shape) descripta en el X3D.

- Color difuso: Atributo diffuseColor de Material
- Color especular: Atributo specularColor de Material
- Índice de difusión: MetadataFloat de nombre “diffuse” dentro del nodo MetadataSet de Material
- Índice de especularidad: MetadataFloat de nombre “specular” dentro del nodo MetadataSet de Material
- Intensidad de ambiente: Atributo ambientIntensity de Material
- Transparencia: Atributo ambientIntensity de Material

- Índice de reflexión: MetadataFloat de nombre “reflection” dentro del nodo MetadataSet de Material
- Índice de refracción: MetadataFloat de nombre “refraction” dentro del nodo MetadataSet de Material
- Brillo: Atributo shininess de Material

Reflexión

Se utiliza el índice de reflexión de la metadata del material. Se calcula en el método computeReflection de la clase RayTracer.

Refracción

Se utiliza el índice de refracción de la metadata del material. Se calcula en el método computeRefraction de la clase RayTracer.

Transformaciones

Las transformaciones se realizan de acuerdo a los atributos scale, rotation y translation de los nodos Transform de X3D, y pueden estar anidadas. Las esferas sólo soportan traslación y escala en radio (tomando la coordenada de mínimo escalado). (Con esto, la rotación no influye.) Si se soportara un escalado no uniforme de las esferas, se generarían elipsoides que exceden el alcance de esta implementación.

Figuras

Las figuras soportadas son Sphere, TriangleSet, IndexedTriangleSet, IndexedTriangleStripSet e IndexedTriangleFanSet. En los conjuntos de triángulos indexados, se ignoran los parámetros del sentido de giro y los colores.

Método de volúmenes envolventes de cajas

Para determinar de manera más eficiente, en una primera aproximación, si hay una intersección en un objeto (ya sea una figura básica o un conjunto de figuras) se utiliza el método de cajas envolventes. El modelo de objetos utilizados fuerza a cada objeto a implementar (a través de la interfaz de SceneObject, con el método getBounds) un método que devuelva la mínima caja envolvente que lo contenga. Esto es utilizado en la construcción del octree de objetos de la escena y en la intersección con los rayos para ganar eficiencia.

Octrees

Para acceder más eficientemente a los objetos dispuestos en la escena se utiliza la clase OctreeScene que ubica a los objetos básicos (triángulos o esferas, no conjuntos de sub-figuras) en una estructura jerárquica que permite tener un conjunto más reducido de objetos candidatos a ser intersectados por un rayo. Dos parámetros importantes de esta estructura son las contantes (variables de clase en OctreeNode) que indican la cantidad máxima de objetos por nodo y la profundidad máxima del octree, ya que se utilizan en el criterio de corte para el crecimiento del árbol.

Otras consideraciones

Para mayor detalle sobre los datos cargados del archivo X3D se puede consultar el código de la clase SceneLoader. La mayoría de las fórmulas donde se utilizan las propiedades del material están en el método getColor de la clase RayTracer (o alguno de los métodos auxiliares que llama). La implementación del modelo de objetos utilizado está en las clases del paquete objects.

Forma de invocación

```
$ java -Djava.library.path=lib -jar raytracer.jar [options]
```

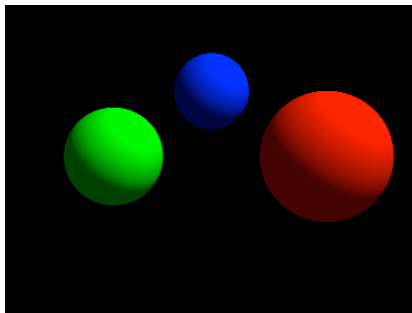
Las opciones soportadas por línea de comandos son todas las especificadas por consigna. Las opciones que corresponden a características implementadas (o sea, que no son ignoradas) son las siguientes:

- `-progress`: Muestra una barra de progreso por terminal de caracteres (80 caracteres), que avanza según la cantidad de píxeles graficados sobre la cantidad total (dada por las dimensiones de la imagen a generar).
- `-as <N>`: Procesa *anti-aliasing* estocástico siendo N la raíz cuadrada de la cantidad de rayos que se tiran por cada uno que se tiraría sin esta opción.
- `-o <archivo de salida>`: Especifica el nombre del archivo de salida. (Soporta cualquier extensión de los formatos de imagen soportados por las bibliotecas de imágenes de Java.)
- `-i <archivo de entrada>`: Especifica el nombre del archivo de entrada (descripción de la escena en formato X3D).
- `-s <ancho>x<alto>`: Genera la imagen de la escena en el tamaño especificado.
- `-show`: Muestra una ventana con la imagen generada.
- `-p <N>`: Procesa penumbras siendo N la cantidad de rayos que se tiran desde las luces puntuales por cada píxel.

Ejemplos

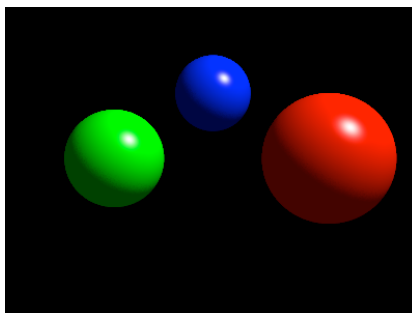
La primitiva más simple implementada es la esfera. Los objetos de la escena pueden ser configurados con los siguientes parámetros a través del X3D: color difuso, color especular, índice de difusión, índice especular, intensidad del ambiente, transparencia, índice de refracción, índice de reflexión y brillo.

En el siguiente ejemplo, se grafican tres esferas utilizando solamente la luz ambiental y el color difuso. Tanto la especularidad como la reflexión y la refracción están anuladas.



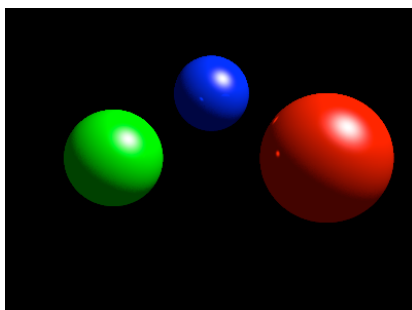
Test1.x3d

Si a este ejemplo se le agrega especularidad, se observa el reflejo de la luz en las esferas. El tamaño de este reflejo (dado por el parámetro nShiny del modelo de Phong) se obtiene a partir del atributo shininess del x3d, multiplicandolo por 128, según indica el estándar de X3D. El resultado es el siguiente:



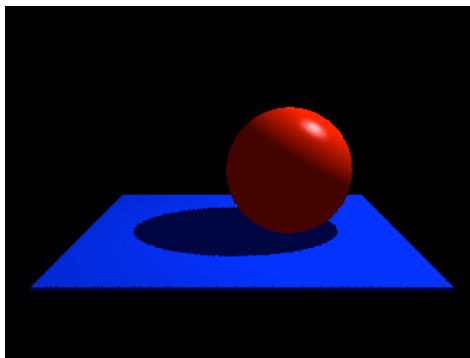
Test2.x3d

Si además se le agrega reflexión, se observa que los objetos se reflejan entre sí:



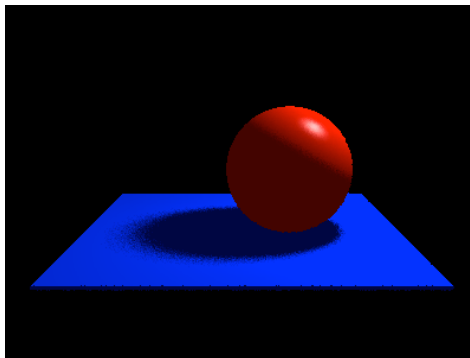
Test3.x3d

Si no se implementan penumbras, las sombras obtenidas son demasiado “marcadas”, y poco realistas, como se muestra en el siguiente ejemplo:



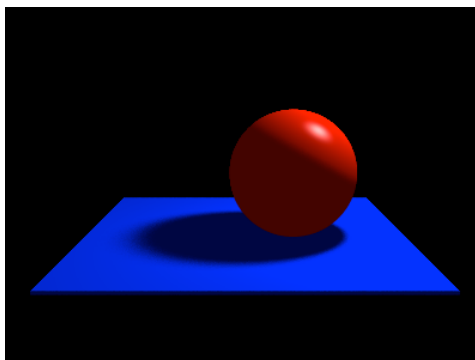
Test4.x3d (ejecutado sin parámetros de antialiasing ni penumbras)

Si la opción de penumbras está desactivada, entonces los rayos de luz salen del punto exacto que indica el atributo position de la luz puntual. Si se activa la opción de penumbras, entonces se considera el parámetro radius de la luz, y los rayos se originan de manera estocástica desde distintos puntos pertenecientes a la “esfera” de la luz. El resultado con la escena anterior es el siguiente:



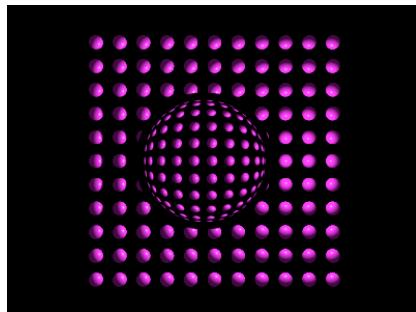
Test4.x3d (ejecutado con -p10, sin antialiasing)

Como la implementación de penumbras tiene una componente estocástica, se ve cierto “granulado” en la sombra, que tiene que ver con la distribución utilizada para tomar las muestras de la luz, y también para la cantidad de rayos. Sin embargo, habilitando antialiasing, para un mismo punto se computa varias veces la sombra, y por lo tanto se suavizan estos resultados. Habilitando antialiasing se obtiene el siguiente resultado:



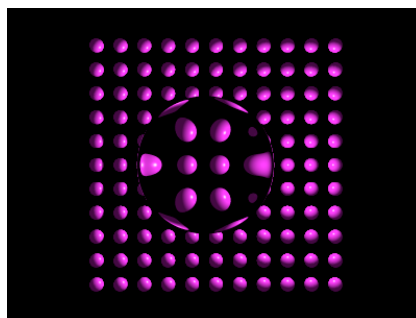
Test4.x3d (ejecutado con -p10 y -as5)

En cuanto a la refracción, se utilizan dos atributos de los objetos: su transparencia y su índice de refracción. La transparencia determina qué porcentaje del rayo refractado se suma al color acumulado para el punto en cuestión, y el índice de refracción para determinar cuánto se quiebra el rayo al atravesar el objeto. En el siguiente ejemplo, se tienen esferas distribuidas en forma de grilla, con una esfera más grande adelante, con transparencia 1 e índice de refracción 0.9. Como dicho índice es menor que 1, los rayos se “dispersan” y por lo tanto se obtiene una imagen como si tuviera un “zoom” hacia atrás.



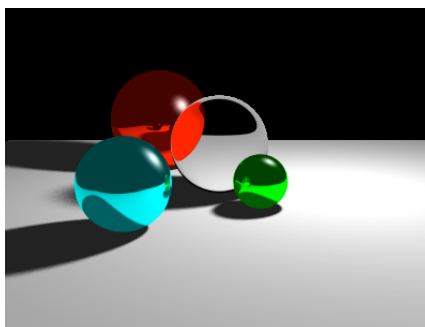
Test5.x3d

Si se modifica dicho índice de refracción a un valor mayor que 1, como por ejemplo 1.1, el resultado es el siguiente. Se puede observar que los rayos se concentran, y por lo tanto hace un efecto de “lupa” sobre el fondo.



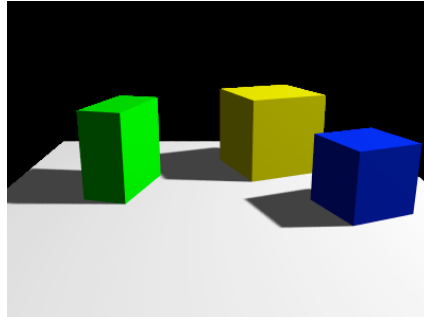
Test6.x3d

El siguiente es un ejemplo que combina reflexión y refracción. Las esferas de colores son totalmente reflectantes, mientras que la transparente refracta.



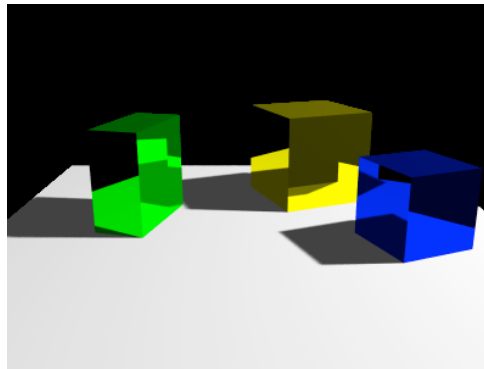
Test7.x3d

Además de esferas, otra primitiva soportada son los triángulos. El orden en el que se especifican los puntos del triángulo determinan la normal (es decir, el triángulo no es “sólido” de ambos lados, sino que solamente del de la normal). En el siguiente ejemplo se construyen cajas con triángulos con la normal hacia afuera, sin reflexión ni refracción.



Test8.x3d

Finalmente, en este ejemplo se hace que las cajas sean totalmente reflectantes.



Test9.x3d

Imagen para el concurso

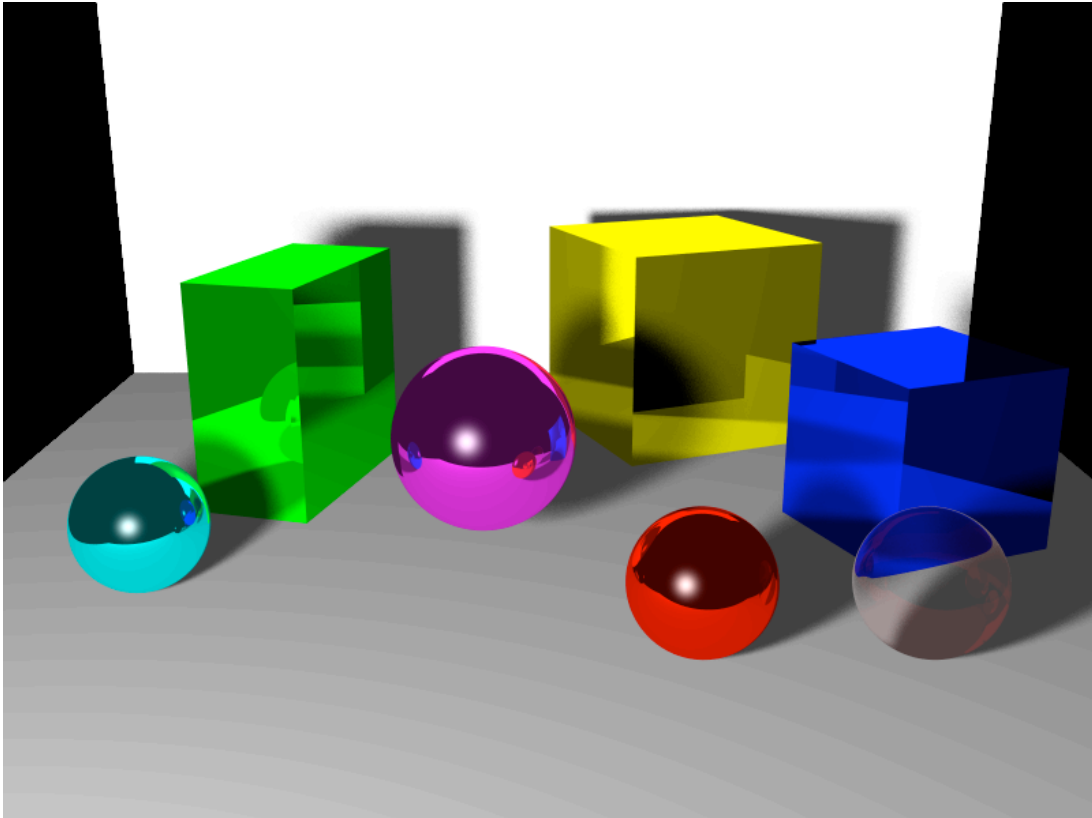


imagen.x3d