



TP Especial N° 2

Ray Tracing

Objetivo

El objetivo de este TP es el desarrollo de un motor de Ray Tracing que permita representar escenas con un balance adecuado entre velocidad y calidad gráfica.

Requerimientos Mínimos para Aprobación

Deberá ser desarrollado en Java, y se permitirán el uso de:

- Toda la biblioteca estándar de Java
- Librerías auxiliares para operaciones con vectores y matrices (ej: vecmath)
- Librerías para el manejo de imágenes (lectura y escritura, en caso de necesitar algo adicional a ImageIO)
- Librerías adicionales de Java3D para escenas.

NO podrán emplearse librerías externas para las pruebas de intersección de los rayos con los objetos de la escena.

Características a Implementar

El programa deberá implementar las siguientes funciones:

- Cámara móvil: Deberá poder indicarse una ubicación y orientación arbitraria para la cámara
- Antialiasing: Podrá ser estocástico o adaptable
- Múltiples Luces: Para cada una deberá poder indicarse su color y alcance (que podrá ser infinito). El modelo empleado para el *falloff* (lineal, exponencial, etc.) queda librado a la implementación. Deberán implementarse por lo menos luces puntuales (iluminan en todas direcciones). Podrán implementarse otros tipos (spot, etc.)
- Sombras
- Penumbra: Deberá implementarse penumbra mediante alguno de los métodos vistos en clase u otros.
- Reflexión
- Al menos una de las siguientes opciones:
 - Refracción
 - Texturas
 - Profundidad de Campo

Las escenas deberán permitir elementos de los siguientes tipos:

- Esferas
- Triángulos (TriangleSet, TriangleStrip, TriangleFan, etc.)



Para el material de objetos se deberán implementar las siguientes características:

- Intensidad ambiente
- Color (emisivo)
- Color difuso
- Color especular
- Índice de especularidad
- Índice de transparencia

Se deberá implementar algún método de volúmenes envolventes. Podrá ser uno de los siguientes:

- Cajas
- AABB
- Esferas
- Otro (consultar a los docentes antes de elegir otro)

Se deberá implementar algún método de subdivisión espacial, para acelerar los cálculos, que podrá ser uno de los siguientes:

- Octrees
- Subdivisión regular del espacio
- KD-Trees
- BSP
- Otro (consultar a los docentes antes de elegir otro)

Requerimientos de Ejecución

El programa deberá poder correr en máquinas virtuales de Sun Java 5 o 6. Deberán entregarse tanto el código fuente como los binarios. Deberá acompañarse de un archivo de ejecución de comandos (.bat o .cmd en windows, .sh en Linux) que permita ejecutarlo incluyendo todo lo necesario para el *classpath* y aceptando opciones de línea de comando.

El programa deberá recibir opciones en la línea de comandos de la forma **-<opción> [parámetro]**, o sea, guión seguido del indicador de opción, y de haber un parámetro deberá estar separado por un espacio. Las opciones indicadas como “(Opcional)” podrán no estar presentes en toda línea de comando, pero SI deberán ser implementadas por el programa.

Se deberán implementar las siguientes opciones:

-progress

(Opcional) El programa deberá mostrar el progreso en el *rendering* mediante una salida periódica en pantalla (ej: impresión de un carácter cada x líneas, o de un porcentaje de completado). En caso de no especificarla el comportamiento *default* será no mostrar nada.

-gui

(Opcional) En el caso de que el programa implemente una interfaz gráfica en la que se puedan especificar los parámetros de ejecución, esta opción permitirá mostrarla e ignorar todas las otras opciones de la línea de comandos.

**-a[a|s] <N>**

(Opcional) Indica que se desea realizar antialiasing de la escena. La segunda letra ('a' o 's') indicará si es adaptable o estocástico, en caso de que se hayan implementado ambos métodos. El parámetro N indicará el umbral a usar para el método, siendo la cantidad de rayos por pixel para estocástico y la cantidad de niveles máximos de recursión para el adaptable. Si se implementan dos métodos y no se aclara cuál se usará (ej: usando solo -a N) se tomará el estocástico por *default*.

Ejemplos: “-as8” indica antialiasing estocástico con 8 rayos por pixel, y “-aa4” indica antialiasing adaptable con hasta 4 niveles de recursión.

-o <nombre de archivo>

(Opcional) Nombre del archivo de salida, incluyendo su extensión. En caso de no indicarlo usará el nombre del archivo de input reemplazando la extensión y usando el formato PNG.

Ejemplo: si se especifica **-i casa.x3d**, la salida será al archivo **casa.png**.

-i <nombre de archivo>

Nombre del archivo de entrada (definición de escena en X3D)

-size <ancho>x<alto>

(Opcional) Tamaño de la imagen a generar, en pixeles. En caso de no especificarlo, el valor *default* será de 400x300.

Ejemplo: -s 640x480

-show

(Opcional) Al finalizar el raytracing mostrará una ventana con la imagen. En caso de no especificarlo, el comportamiento *default* será no mostrar la ventana.

-p <N>

(Opcional) Usar penumbra. La N indica el límite de procesamiento impuesto y significará lo mismo que para el antialiasing (dependiendo del método usado). Si se implementan dos o más métodos distintos para las penumbras, se podrán especificar explícitamente mediante una segunda letra, de la misma forma que en la opción de antialiasing.

-pdc <T>

(Opcional) Usar profundidad de campo. El parámetro <T> indica el tamaño (en medidas de la escena) del lente desde el que se proyectan los rayos para simular profundidad de campo.

Representación de la Escena en Memoria

Quedará librado a cada implementación la forma de representar la escena en memoria (*scenegraph*). Podrá usarse el scenegraph de Java 3D u otros.



Formatos de Archivos Gráficos

- Los formatos de salida de la imagen deberán ser sin pérdida de datos (*lossless*), y se deberán implementar por lo menos PNG y BMP. Podrán implementarse formatos adicionales (ej: TGA).
- La escena a representar se leerá desde un archivo con formato X3D.
- Se aceptarán como formatos para texturas todos los siguientes: PNG, GIF, BMP, JPG. Podrán implementarse formatos adicionales.

Sugerencias Importantes

- Comenzar donde termina el trabajo de RayCasting (TP4).
- Ante una pantalla negra, probar las escenas con un visualizador de X3D.
- Implementar de a un *feature* a la vez y luego hacer *testing regresivo* sobre los anteriormente implementados.
- No buscar performance inmediatamente: experimentar con escenas con pocos objetos para probar cada feature nuevo.
- Aprovechar los procesadores modernos multi-core: subdividir el rendering para distribuirlo entre varios threads.

Material a Entregar en el SVN

En el directorio TPE02 del SVN correspondiente, colocar:

- Códigos fuente, en el subdirectorio “source”
- Código binario, en el subdirectorio “bin”
- Un archivo de ejecución de comandos (*.bat* o *.cmd* en windows, *.sh* en Linux) que permita ejecutarlo incluyendo todo lo necesario para el *classpath* y aceptando opciones de línea de comando, en el subdirectorio “bin”
- Imágenes obtenidas con ejecuciones de prueba en el subdirectorio “img”
- Documento del informe, en el subdirectorio “doc”
- Imagen elegida para participar en el concurso interno de RayTracing, en el subdirectorio “concurso”

Fecha de Entrega

- Miércoles 20 de Junio
- Entrega tarde hasta el 2 de Julio (cada semana de retraso se penaliza en un 20%)