TPE Nº 1 – Ray Casting

El motor deberá ser desarrollado en el lenguaje **Java** y para tal fin se permitirá 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 el caso de necesitar funcionalidades no provistas por ImageIO)
- Librerías adicionales de Java para escenas.

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

Objetivo

Construir un motor de *Ray Casting* que determine intersecciones entre rayos y objetos permitiendo representar en pantalla un conjunto de escenas predefinidas.

Características a implementar

El motor de Ray Casting deberá implementar las siguientes funcionalidades:

- Soportar las siguientes primitivas: triángulos, cuadriláteros y esferas.
- Tener posicionada la cámara en el punto (0, 0, 10) mirando hacia el origen de coordenadas. Se asume un sistema de coordenadas de mano derecha.
- Poder renderizar las siguientes escenas, representadas en memoria:
  - **Escena 1 (scene1.sc):** deberá contener una pirámide rectangular, cuyo lado de la base y altura deberá tener una longitud de 5. La base de la misma deberá estar centrada en el origen. En su vértice superior deberá tener posicionada en perfecto equilibrio una esfera de radio 1. Adicionalmente a una distancia de 3, en cada una de los vértices de la base, deberá contener una esfera de radio 0.5.
  - **Escena 2 (scene2.sc):** deberá contener 64 esferas de radio 0.5 dispuestas en forma de cubo (4 x 4 x 4 esferas) equidistantes unas de otras por una longitud de 1.5. La distancia entre esferas se mide desde sus centros. El centro del cubo debe coincidir con el origen del sistema de coordenadas.
  - **Escena 3 (scene3.sc):** deberá contener de forma intercalada 3 cubos de lado 2 y dos esferas de radio 1 alineados de manera centrada sobre el eje x. Las bases de los cubos y las esferas deben estar "apoyadas" sobre el plano y = 0. La separación entre las figuras, esto es, el "hueco de aire" entre ellas es de 0.5. La escena debe

estar centrada a lo largo del eje x. Los centros de las figuras deben coincidir con el plano z = 0.

Detalles de la implementación

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 otro.

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).

Requerimientos de ejecución

La aplicación deberá recibir opciones por 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 comandos, pero sí deberán ser implementadas por la aplicación.

Se deberán implementar las siguientes opciones:

`-i [nombre de archivo]`
Nombre del archivo con la escena. Los valores posibles son de la forma: *sceneX.sc* dónde *X* deberá remplazarse por el número de la escena. La representación de la misma deberá estar en memoria y no deberá cargarse desde un archivo.

`-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 scene1.sc*, la salida será al archivo *scene1.png*.

`-size <ancho>x<alto>`
Indica la cantidad de pixeles de la imagen de salida. Valor por defecto: 640x480.
Ejemplo: -size 640x480

`-fov <ángulo>`
Indica el ángulo de apertura horizontal en grados. Valor por defecto: 60.
Ejemplo: -fov 60

`-cm [random | ordered]`
(Opcional) Indicador del modo de asignación de color a las primitivas de la escena. Puede tomar dos valores:

- `random`: la asignación del color es de forma aleatoria (valor por defecto).
- `ordered`: la asignación de color dependerá de la cercanía al punto donde se encuentre posicionada la cámara. El orden de asignación es: *violeta* (primitiva más cercana), *azul*, *verde*, *amarillo*, *naranja* y *rojo* (primitiva más lejana).

`-cv [linear | log]`

(Opcional) Indicador del modo de variación del color de los elementos de la escena. Puede tomar dos valores:

- `linear`: la variación del color es proporcional a la distancia a la cual la primitiva se encuentra de la posición de la cámara (valor por defecto).
- `log`: la variación del color es logarítmica.

`-time`

(Opcional) Mostrará el tiempo empleado en el renderizado.

Fecha de entrega

La fecha de entrega es el Miércoles 14 de Abril, al comienzo del horario de clase.

Material a entregar en el SVN

En el directorio TPE01 del SVN se deberán entregar:

- Códigos fuente, en el subdirectorio ***src***.
- Código binario, en el subdirectorio ***bin***.
- Un archivo de ejecución de comandos (*.bat* o *.cmd* en windows, *.sh* en Linux) que permita ejecutar la aplicación incluyendo todo lo necesario para el *classpath* y aceptando opciones de línea de comando.
- Imágenes obtenidas con ejecuciones de prueba en el subdirectorio ***img***.
- Documento del informe, en el subdirectorio ***doc***.

Antes cada entrega, **informar por e-mail a los docentes el número de revisión de la entrega**. El mail de la cátedra es: cg@it.itba.edu.ar.

En caso de usar *Maven* u otra herramienta que requiera una estructura de directorios distinta, notificar a los docentes al momento de entregar.

En caso de que el tamaño total de la entrega exceda los **50Mb**, notificar a los docentes antes de entregar.

La dirección de los repositorios de SVN les será informada vía IOL entre los días 31 de Marzo y 4 de Abril.