Síntesis de Imagen por medio de *Ray Tracing*
Proyecto Final
Procesos Paralelos y Distribuidos

Maximiliano Monterrubio Gutiérrez
No.Cuenta: 405074855

E-mail: maxmg22@ciencias.unam.mx

9 de enero de 2009

Resumen

La síntesis de imagen 3D es un área de trabajo muy fresca en la computación moderna. Este proyecto tiene como objetivo el desarrollo de una aplicación de síntesis de imágenes empleando técnicas de procesamiento paralelo y distribuido.

1 Definición del problema

El proyecto desarrollado resuelve el problema de sintetizar una imagen tridimensional dado una especificación de *escena*. Dicha escena es un conjunto de objetos del espacio tridimensional y atributos tanto de estos últimos, como de la escena de manera global.

Aunque existen varias técnicas de síntesis de imagen, este proyecto se enfoca en la más moderna y utilizada actualmente en el ámbito de la visualización computacional: el algoritmo de *ray tracing*.

La parte central de este proyecto, además de ser la correcta implementación de este algoritmo, es el poder explotar el paralelismo de las nuevas máquinas multi-núcleo y además ofrecer la posibilidad de un sistema de síntesis distribuido; esto es, poder coordinar diversos equipos de cómputo conectados por medio de una red para realizar conjuntamente el trabajo de síntesis de la escena especificada.

2 Especificación del proyecto

2.1 Características Generales

1. **Plataforma de desarrollo:** El proyecto está desarrollado en `Java` con la versión `1.6` de la máquina virtual utilizando el entorno integrado de desarrollo `NetBeans 6.1`.
2. **Algoritmo de síntesis:** *Ray tracing* recursivo.
3. **Formato de la especificación de escena:** Documento `XML` con `DTD` embebido.
4. **Soporte multi-hilo:** Configurable y 100 % libre de espera y bloqueos.
5. **Subsistema de síntesis distribuido:** Montado sobre el stack `TCP/IP` utilizando el protocolo `TCP`. La aplicación integra tanto el módulo de cliente como el de servidor.
6. **Almacenamiento de imágenes:** Se soporta únicamente el formato `PNG` (*Portable Network Graphics*).
7. **Interfaz gráfica:** Biblioteca `Swing` de Java.

2.2 Sistema de síntesis

2.2.1 Soporte de primitivos

1. Plano infinito con la notación vector *punto/normal*.
2. Esferas definidas como vector origen y radio.
3. Paralelepípedos alineados a los ejes X, Y, Z definidos por un vector de esquina, y un vector para la longitud de cada lado en su eje respectivo.

2.2.2 Iluminación

1. Luz puntual con sombra sencilla.
2. Luz de superficie con multimuestreo bidimensional de monte carlo configurable en la escena.

2.2.3 Materiales

1. Color del material
2. Iluminación básica empleando el modelo de Lambert.
3. Soporte de iluminación especular con dureza e intensidad configurable.
4. Reflexión con coeficiente configurable.
5. Índice de refracción e intensidad configurable.

2.2.4 Algoritmo de *Ray Tracing*

1. Reflexión y refracción por *ray trace* recursivo con rebote configurable.
2. Reducción de artefactos de *aliasing* por medio de soporte para *oversampling* configurable en la escena.
3. Se puede configurar la ubicación y dirección de la cámara.

2.2.5 Sistema de síntesis multi-hilo

1. Soporte para *ray tracing* multi-hilo libre de bloqueo y espera.
2. Visualización progresiva del *framebuffer*.

2.2.6 Subsistema de síntesis distribuido

1. Arquitectura cliente/servidor bajo `TCP`.
2. Soporte para balanceo de carga basado en reloj de procesador y número de núcleos por nodo, soportando así síntesis distribuida y paralelizada en cada nodo.

3 Manejo de la aplicación

3.1 Entrada de datos

La escena 3D a sintetizar se debe especificar como un archivo `XML` con el siguiente esquema:

```
<!DOCTYPE scene [
    <!ELEMENT scene    ((light|box|plane|sphere)+)>
        <!ATTLIST scene       width     CDATA #REQUIRED>
        <!ATTLIST scene       height    CDATA #REQUIRED>
        <!ATTLIST scene       aa        CDATA #REQUIRED>
        <!ATTLIST scene       name      CDATA #REQUIRED>
        <!ATTLIST scene       campos    CDATA #REQUIRED>
        <!ATTLIST scene       camdir    CDATA #REQUIRED>
    <!ELEMENT light    EMPTY>
        <!ATTLIST light   pos       CDATA #REQUIRED>
        <!ATTLIST light   dir    CDATA #REQUIRED>
        <!ATTLIST light   intensity CDATA #REQUIRED>
        <!ATTLIST light   color     CDATA #REQUIRED>
        <!ATTLIST light   type      (point|surface) #REQUIRED>
        <!ATTLIST light   name      CDATA #REQUIRED>
        <!ATTLIST light   size      CDATA #REQUIRED>
        <!ATTLIST light   samples   CDATA #REQUIRED>
    <!ELEMENT sphere   EMPTY>
        <!ATTLIST sphere  radius    CDATA #REQUIRED>
        <!ATTLIST sphere  pos       CDATA #REQUIRED>
        <!ATTLIST sphere  specular  CDATA #REQUIRED>
        <!ATTLIST sphere  diffuse   CDATA #REQUIRED>
        <!ATTLIST sphere  reflect   CDATA #REQUIRED>
        <!ATTLIST sphere  spechard  CDATA #REQUIRED>
        <!ATTLIST sphere  name      CDATA #REQUIRED>
        <!ATTLIST sphere  color     CDATA #REQUIRED>
        <!ATTLIST sphere  refract   CDATA #REQUIRED>
        <!ATTLIST sphere  ior       CDATA #REQUIRED>
    <!ELEMENT box      EMPTY>
        <!ATTLIST box     width     CDATA #REQUIRED>
        <!ATTLIST box     height    CDATA #REQUIRED>
        <!ATTLIST box     depth     CDATA #REQUIRED>
        <!ATTLIST box     pos       CDATA #REQUIRED>
        <!ATTLIST box     specular  CDATA #REQUIRED>
        <!ATTLIST box     spechard  CDATA #REQUIRED>
        <!ATTLIST box     diffuse   CDATA #REQUIRED>
        <!ATTLIST box     reflect   CDATA #REQUIRED>
        <!ATTLIST box     name      CDATA #REQUIRED>
        <!ATTLIST box     color     CDATA #REQUIRED>
        <!ATTLIST box     refract   CDATA #REQUIRED>
        <!ATTLIST box     ior       CDATA #REQUIRED>
    <!ELEMENT plane    EMPTY>
        <!ATTLIST plane    normal    CDATA #REQUIRED>
        <!ATTLIST plane    pos       CDATA #REQUIRED>
        <!ATTLIST plane    specular  CDATA #REQUIRED>
        <!ATTLIST plane    spechard  CDATA #REQUIRED>
        <!ATTLIST plane    diffuse   CDATA #REQUIRED>
        <!ATTLIST plane    reflect   CDATA #REQUIRED>
        <!ATTLIST plane    name      CDATA #REQUIRED>
        <!ATTLIST plane    color     CDATA #REQUIRED>
```

```
        <!ATTLIST plane    refract   CDATA #REQUIRED>
        <!ATTLIST plane    ior       CDATA #REQUIRED>

]>
```

Lo más sencillo es ver un ejemplo:

```
<scene width="800" height="600" aa="1" name="TestScene" campos="0,-2,3"
       camdir="0,0.5,-1">
    <light color="1,1,0.7" pos="1,3,6" dir="0,0,-1" intensity="0.5"
           type="surface" name="Light2" samples="8" size="1" />
    <light color="0.6,0.6,1.0" pos="1,-3,6" dir="0,0,-1" intensity="0.5"
           type="surface" name="Light2" samples="8" size="1" />
    <plane color="1,1,1" diffuse="0.6" name="Plane1" specular="0.1"
           spechard="10" pos="0,0,1" normal="0,0,1" reflect="1"
           refract="0.0" ior="1.0" />
    <sphere color="0.0,0.7,0.0" diffuse="0.9" specular="0.9" spechard="30"
            reflect="0.5" radius="1" pos="-2,0,2" name="Sphere1"
            refract="0.0" ior="1.0" />
    <sphere color="0.8,0.1,0.1" diffuse="0.7" specular="0.7" spechard="30"
            reflect="1.0" radius="1" pos="2,0,2" name="Sphere2"
            refract="0.0" ior="1.0" />
    <box color="0.5,0,0.8" pos="-0.5,1,1" width="1" height="1" depth="1"
         specular="0.7" spechard="20" diffuse="0.9" reflect="0" name="Box1"
         refract="0.0" ior="1.0" />
    <box color="0.0,0.3,0.3" pos="-0.5,-1,1" width="1" height="1" depth="1"
         specular="0.7" spechard="20" diffuse="0.9" reflect="0" name="Box1"
         refract="0.0" ior="1.0" />
</scene>
```

4 Síntesis local

Para sintetizar una imagen en el equipo local, basta con seleccionar el menú *Render/Render Scene...* donde se abre un panel para seleccionar el documento XML que especifica la escena y luego se puede seleccionar el número de hilos de ejecución a crear.

5 Síntesis distribuida

Realizar un trabajo de síntesis distribuido requiere ejecutar diversas instancias de la aplicación, una en cada nodo cliente, y una en el nodo servidor. En caso de que el servidor también vaya a realizar trabajo es necesario ejecutar dos instancias de la aplicación, una actuando como servidor y la otra como cliente usando la dirección de *loopback* `127.0.0.1`.

5.1 Nodo servidor

En caso de querer establecer un equipo como servidor, seleccionamos la opción de menú *Render/Distributed Render/Host a scene...*, posteriormente aparece un cuadro de diálogo que pregunta el número de nodos a esperar y el puerto `TCP` de escucha.

NOTA: El servidor espera indefinidamente y no envía carga a ningún nodo hasta que se alcance el número de nodos especificado en este cuadro de diálogo. La razón de esto es porque necesita conocer información relevante de cada nodo para realizar el balanceo de cargas.

Una vez introducidos estos datos, aparece un cuadro de diálogo para introducir la escena a sintetizar. Al momento de seleccionar el archivo, el servidor inicia el proceso de escucha de los nodos cliente y abre una ventana para visualizar el *framebuffer* (i.e. la imagen que se va a ir sintetizando).

Cada que un nodo se conecta y empieza a realizar trabajo, se reflejará en el *framebuffer* servidor conforme su trabajo va avanzando.

5.2 Nodo cliente

Una vez levantado el nodo servidor, es posible conectarse a él levantando la aplicación en un cliente, y seleccionado la opción de menú *Render/Distributed Render/Connect to host....* Posteriormente aparece un cuadro de diálogo donde hay que especificar:

1. **Dirección del servidor:** Ya sea una dirección `IP` o un nombre de dominio.
2. **Puerto:** El puerto `TCP` donde el servidor espera conexiones.
3. **Velocidad de reloj del cliente:** Es importante proporcionar la velocidad de reloj de nuestro equipo con la finalidad de que el balanceador de cargas distribuya de manera más uniforme el trabajo en base al poder de cómputo de cada nodo.
4. **Hilos de ejecución:** El número de hilos de síntesis que el nodo cliente ejecutará al recibir los datos.

Al haber introducido estos datos, en caso de que la conexión sea satisfactoria y no hayan errores, aparecerá un diálogo de progreso del proceso de síntesis local. Al finalizar, el servidor habrá recibido todo el trabajo realizado por este nodo local y es posible cerrar la aplicación.

6 Ejecutar el proyecto

Para ejecutar el proyecto necesitamos compilar el código y crear el paquete *jar* con los archivos de clase y bibliotecas necesarias, para ellos usamos el target `jar` de `ant`, el `jar` ejecutable se encuentra en `dist`:

```
$ cd <ubicacion del proyecto>
$ ant jar
$ cd dist
$ java -jar SPDRender.jar
```

En el directorio `data` se encuentra una escena de muestra con 3 configuraciones diferentes para poder contrastar las características del *ray tracer*:

1. `sample1_noAA_pointShadows.xml` Escena de ejemplo con iluminación puntual sin sobremuestreo (*oversampling*).
2. `sample1_8AA_pointShadows.xml` Escena de ejemplo con iluminación puntual usando 64 muestras por píxel.
3. `sample1_8AA_softShadows.xml` Escena de ejemplo con iluminación de superficie empleando 64 muestras por píxel y 64 muestras por fuente luminosa. El trabajo de síntesis de este archivo es pesado, tomar en cuenta al probarlo.