CSE 461 Programming Assignment 1

DUE

April 15, 2024, 23:55

Description

- This is an individual assignment. Please do not collaborate
- If you think that this document does not clearly describes the assignment, ask questions before its too late.

This assignment is about implementing a simple ray tracer.

Scene Description File

This file is in "xml" format. You can use xml parsers.

Definition of the elements which can be found in a scene description file:

- `maxraytracedepth`: This is positive integer which defines the maximum recursive ray tracing depth. Rays starting from camera have depth of 0.
- `background`: `r`, `g`, `b` values for the pixel in a no-hit case. You are going to set the color of the pixel to this value in case the ray does not intersect with any objects in the scene.

```
<background>r g b</background>
```

- `camera`: This defines a camera in the scene. It has the following subfields:
  - `position`: `x`,`y`,`z` coordinates of the camera.
  - `gaze`: gaze direction of the camera.
  - `up`: up vector of the camera.
  - `nearplane`: `left`, `right`, `bottom`, `top` values of the image plane.
  - `neardistance`: distance between the image plane and the camera. `gaze` vector is perpendicular to the image plane.
  - `imageresolution`: `nx` and `ny` dimensions of the image.

```
<camera>
    <position>x y z</position>
    <gaze>x y z</gaze>
    <up>x y z</up>
    <nearplane>left right bottom top</nearplane>
    <neardistance> distance </neardistance>
    <imageresolution>nx ny</imageresolution>
</camera>
```

- `ambientlight`: `r`, `g`, `b` ambient light values. This is the amount of light received by the surface which in shadow.

```
<ambientlight>r g b</ambientlight>
```

- `pointlight`: point light source defined by a position vector and an intensity vector.

```
<pointlight id="someid">
    <position> x y z </position>
    <intesity> x y z </intensity>
</pointlight>
```

- `traingularlight`: planar directional light source defined by a triangle. The direction of the light follows the cross product `(vertex1-vertex2) x (vertex1-vertex3)`

```
 <triangularlight id="someotherid">
            <vertex1> x y z </vertex1>
            <vertex2> x y z </vertex2>
```

```
            <vertex3> x y z </vertex3>
            <intensity> x y z </intensity>
</traingularlight>
```

- `material`: This has the following subfields (the values are between `0.0` and `1.0`):
  - ambient
  - diffuse
  - specular
  - mirrorreflactance
  - phongexponent

```
<material id="someid">
    <ambient>x y z</ambinet>
    <diffuse>x y z</diffuse>
    <specular>x y z</specular>
    <phongexponent>e</phongexponent>
    <mirrorreflactance>x y z</mirrorreflanctance>
</material>
```

- `vertexdata`: lists the `x`, `y`, `z` coordinates of the all the vertices defined in the scene. The vertices are referred by `faces` field under `mesh`

```
<vertexdata>
x1 y1 z1
x2 y2 z2
x3 y3 z3
...
</vertexdata>
```

- `mesh`: Each mesh is a list of faces. Each face is a triangle. Triangles are defined by three vertex indices given in counter-clockwise direction.

```
<mesh id ="someid">
    <materialid>id</materialid>
    <faces>
        face1_vertex1_id face1_vertex2_id face1_vertex3_id
        face2_vertex1_id face2_vertex2_id face2_vertex3_id
        face3_vertex1_id face3_vertex2_id face3_vertex3_id
        ...
    </faces>
</mesh>
```

Here is the complete skeleton of the scene description file:

```
<scene>
    <maxraytracedepth>n</maxraytracedepth>
    <background>r g b</background>
    <camera>
        <position>x y z</position>
        <gaze>x y z</gaze>
        <up>x y z</up>
        <nearplane>left right bottom top</nearplane>
        <neardistance> distance </neardistance>
        <imageresolution>nx ny</imageresolution>
    </camera>
    <lights>
        <ambientlight>r g b</ambientlight>
        <pointlight id="someid">
            <position> x y z </position>
            <intesity> x y z </intensity>
```

```
        </pointlight>
        <triangularlight id="someotherid">
            <vertex1> x y z </vertex1>
            <vertex2> x y z </vertex2>
            <vertex3> x y z </vertex3>
            <intensity> x y z </intensity>
        </traingularlight>
        ...
    </lights>
    <materials>
        <material id="someid">
            <ambient>x y z</ambinet>
            <diffuse>x y z</diffuse>
            <specular>x y z</specular>
            <phongexponent>e</phongexponent>
            <mirrorreflactance>x y z</mirrorreflanctance>
        </material>
        ...
    </materials>

    <vertexdata>
        x1 y1 z1
        x2 y2 z2
        x3 y3 z3
        ...
    </vertexdata>

    <objects>
        <mesh id="someid">
            <materialid>id</materialid>
                <faces>
                    face1_vertex1_id face1_vertex2_id face1_vertex3_id
                    face2_vertex1_id face2_vertex2_id face2_vertex3_id
                    face3_vertex1_id face3_vertex2_id face3_vertex3_id
                    ...
                </faces>
        </mesh>
        ...
    </objects>
</scene>
```

Below is a simple example scene:

```
<scene>
    <maxraytracedepth>6</maxraytracedepth>
    <backgroundColor>0 0 0</backgroundColor>

    <camera>
        <position>0 0 0</position>
        <gaze>0 0 -1</gaze>
        <up>0 1 0</up>
        <nearPlane>-1 1 -1 1</nearPlane>
        <neardistance>1</neardistance>
        <imageresolution>800 800</imageresolution>
    </camera>

    <lights>
```

```
        <ambientlight>25 25 25</ambientlight>
        <pointlight id="1">
            <position>0 0 0 </position>
            <intensity>1000 1000 1000</intensity>
        </pointlight>
        <triangularlight id="2">
            <vertex1> 0 0 0 </vertex1>
            <vertex2> 1.2 0.5 0.5 </vertex2>
            <vertex3> 0.5 0.5 0.5 </vertex3>
            <intensity> 800 800 800 </intensity>
        </traingularlight>
    </lights>

    <materials>
        <material id="1">
            <ambient>1 1 1</ambient>
            <diffuse>1 1 1</diffuse>
            <specular>1 1 1</specular>
            <mirrorreflactance>0 0 0</mirrorreflactance>
            <phongexponent>1</phongexponent>
        </material>
    </materials>

    <vertexdata>
        -0.5 0.5 -2
        -0.5 -0.5 -2
        0.5 -0.5 -2
        0.5 0.5 -2
        0.75 0.75 -2
        1 0.75 -2
        0.875 1 -2
        -0.875 1 -2
    </vertexdata>

    <objects>
        <mesh id="1">
            <materialid>1</materialid>
            <faces>
                3 1 2
                1 3 4
            </faces>
        </mesh>
        ...
    </objects>
</scene>
```

Remarks

- There may be more than 1 material
- There may be more than 1 point light source
- There may be more than 1 mesh.
- It is a good idea to come with an object oriented solution.
- The performance of your solution is important. So, try to write efficient code. You should use threads.
- Discuss about the running time of your implementation. Test using different scenes and measure the render time. The efficiency of your implementation may affect your grade. (you may lose up to 30% if your implementation is not efficient.)
- In order to calculate the shadow properly and avoid numeric errors, you may need to create an offset on surface

points where the rays hit. For this, define a constant in your program.

Turn in

- You are going to submit your implementation in a zip file. You will include a documentation about hot to compile and/or run your program.
- You are going to demonstrate the run of your program. It is going to be either through a teams meeting or in a face-to-face meeting.