Simple Ray Tracer – Assignment 4

Source Code Description

The source code consists of the following modules:

1) **Parser** – Parses the input file into commands and arguments. This is a tokenizer.
2) **Command** – Ensures syntactical correctness of commands and sends them to the required handlers. Maintains an enumeration with a list of all available commands. A table maintians the command string, the associated enumeration identifiers and the number of arguments per command. This greatly simplifies command handling and management.
3) **Camera** – Stores camera viewing and perspective parameters. Calculates Rays given screen co-ordinates
4) **Image** – Stores and writes Traced Image
5) **SceneObjects** – Stores common properties and operations of each primitive. Since the following are properties and operations that are required and common to any primitives.

    a. **Material Properties**
    b. **Refraction** properties
    c. **General Lighting Calculations** – The contribution of each light to each primitive, using its properties has been calculated in `calculateIllumination`
    d. **Spotlight** contribution computations
    e. **General Transformation** housekeeping routines – Transforming Rays, homoogenous conversions etc
    f. **Transforming Normals**
    g. **Reflection Calculations** – Calculating reflection vectors and recursive computations
    h. **Refraction Computations**
    i. **Light Equation**: Net color and shading computations, from lighting equation using the parameters of
        i. Global Ambient
        ii. Emissive
        iii. Visibility: Shading computations are performed by firing rays to rach light in the scene
        iv. Each Light’s color interactions with each materials Diffuse and Specular components
        v. Recursive computations of Reflections
        vi. Recursive computations of refractions

6) **Sphere** (*inherits from SceneObjects*)–
   a. Geometric properties of center and radius.
   b. Net transforms active at the time of creation of the object in a `mat4` object.
   c. Per point normal computations are performed.
   d. Per point transformations are performed.
   e. Intersection calculations per object

7) **Triangle** (*inherits from SceneObjects*) –
   a. Geometric properties.
   b. Indices into global array of Vertexes.
   c. Indexes into global array of normals.
   d. Net transforms active at the time of creation of the object in a `mat4` object.
   e. Per point normal computations are performed.
   f. Per point transformations are performed.
   g. Intersection calculations per object

8) **Vertexes** –
   a. Global array of vertexes
   b. Global array of vertex normals
   c. Stores maxvertexes and maxvertnormals

9) **Transform** –
   a. Implements a stack of transforms. By default has the identity matrix on top of it.
   b. All transformations within a pair of pushTransform and popTransform directives are multiplied and stored in the same stack entry. The stack is pushed and poped as required.
   c. For efficieny, the current active matrix is always calculated to avoid repeated computations. `currTransform`

10) **Light** –
   a. Maintains properties of position/direction, attenuation, spotlight parameters, for a given light.
   b. The two types of light point and directional have been implemented.

11) **Shading** –
   a. Maintains current global state, in terms of information read from the configuration file, for material, ambient, and light color parameters.
   b. This object is reset each time a new set of properties are defined for a given primitive.

12) **Scene** – Brings everything together.
   a. Maintains a vector of all defined lights.
   b. Maintains a vector of all defined primitives.
   c. Routine for intersection calculations.
   d. Routine that is called recursively for ray tracing.
   e. Main routine calling ray computation, intersection calculation, color computations.
   f. Routine for visibilty computation, given a ray of light, and a given light, computes whether the ray reaches the light.

13) **RayTracer** – Instantiates al above classes. Opens files and initiates the ray tracing proccess.

Features

- Camera Computations
- Transformations of scale, rotations, translations
- Primitives: Trangles and spheres
- Lighting and Shading: A combination of the opengl model and the lighting equation provided.
- Spotlight effects
- Reflections: Using recursive ray tracing
- Refractions: Using Snells and refraction laws. Concepts and physics borrowed from Bramz paper: http://www.devmaster.net/articles/raytracing_series/Reflections%20and%20Refractions%20in%20Raytracing.pdf
  As mentioned in the paper, the method has trncates total internal reflections of lights at certain specific angles. This has been averted by doing a secondary intersection of the ray with the rest of the scene.

Scene Files

- **Scene1,2,3** given with openglviewer have been traced and are provided
- **Cornell Box Scene**:

  1) Shows complex interactions of different spheres and objects of other shapes.
     - The spheres and triangles have varying material peroperties, specular, diffuse and emissive.
     - There are different light sources, on all sides of the scene.
     - The interactions between the lights and objects are visible, diffuse and specular
     - The emissive properties of the lights on top are visilble
     - The shadows of the lights on the cube and spheres are visible

- **Cornell Box with Reflections:**

  A mirror is placed in place of the back face of the cornell box, this shows reflections
- **Refractions**

  A ball is placed on a plane on a blue background
- **Reflections**

  A ball is placed on a plane on a blue background

Detailed Cornell Box Scene Description

Scene: Cornell Box

File name: cornell_box.test

Total Number of walls (including floor): 4

Total Number of objects: 4

Total Number of light sources: 3

Lights

Light 1:

Type: Point Light

Location: Top Center

Properties: diffuse 0.7 0.7 0.7; specular 1 1 1; attenuation 0 1 0

Comments: The light attenuates linearly

Light 2:

Type: Spot Light

Location: Top Left

Properties: diffuse 1 1 1; specular 1 1 1; spot direction: 0 -1 0; spot cutoff: 10; spot exponent: 100

Comments: The light does not attenuates

Light 3:

Type: Spot Light

Location: Top Right

Properties: diffuse 1 1 1; specular 1 1 1; spot direction: 0 -1 0; spot cutoff: 10; spot exponent: 100

Comments: The light attenuation is quadratic

Objects

Sphere 1:

Location: Bottom Left Corner

Material Properties: ambient 0.5 0 0; diffuse 0 0.2 0.2; specular 0 0.5 0.5; shininess 50; emission 0 0 0

Sphere 2:

Location: Right Center

Material Properties: diffuse 1 0.0 0.2; specular 1 1 1; shininess 10; emission 0 0 0

Sphere 3:

Location: Half-immersed right bottom

Material Properties: diffuse 1 0.47 0.1; specular 1 1 1; shininess 10; emission 0 0 0

Cube:

Surface Type: Lambertian

Walls / Floors

Wall 1:

Location: Left

Type: Lambertian

Color: Red

Wall 2:

Location: Right

Type: Lambertian

Color: Green

Wall 3:

Location: Center

Type: Lambertian

Color: Dirty White

Floor:

Type: Tiled Lambertian

Color: Blue/Dirty White