Assignment 1 report

zainana salma

May 2023

1 "classes.cpp"

Firstly, a Vector class is defined. The class has various functions that perform operations on vectors such as addition, subtraction, multiplication, division, dot product, cross product, normalizing, and squared norm of the vector.

Following that, there are two functions, random_cos and boxMuller. random_cos is a helper function that returns a random cosine-weighted vector around the given input vector and is used to model indirect lighting in ray tracing. The boxMuller function generates two random numbers following a normal distribution with a given standard deviation.

The next two classes defined in the code are the Ray class and the Camera class. The normalized_ray_direction function generates rays for each pixel in the image plane based on the camera parameters, uses boxMuller if antialiasing and returns a Ray object.

The fifth class is the Intersection class which stores information about the intersection of a ray with a geometry object. It has boolean variable is_intersection, point and normal of intersection, distance from the origin of the ray to the intersection point, rho that stores the color of the geometry at the intersection point, and a pointer geometry that points to the geometry object that was intersected.

The sixth class is the Geometry class which is an abstract class used as a base for other geometric classes. It defines a virtual function intersect that takes a Ray object as input and returns an Intersection object.

The seventh class is the Sphere class which inherits from the Geometry class and represents a sphere. It has variables center and radius of the sphere, albedo, and a vector details that stores additional details about the sphere such as whether it has a mirror effect, whether it is transparent and its refractive index. It also defines the intersect function to calculate the intersection between a ray and the sphere.

The eighth class is the Boundingbox class which represents a bounding box around a geometry object. It has B_min and B_max that store the minimum and maximum coordinates of the box respectively. It defines is_intersect that takes a Ray object and the distance to the intersection by reference to returns a boolean indicating whether the ray intersects the bounding box and also updates the reference variable with the updated distance.

The last class is the node class which represents a node in a tree structure used for efficient ray tracing. It has variables starting_triangle and ending_triangle, Bbox of the node, child_left and child_right, and a boolean is_leaf that indicates whether the node has children.

2 "scenes.cpp"

This is C++ code for a scene class that represents a 3D scene consisting of geometrical objects such as spheres and triangle meshes, a light source, and camera position. It provides methods to add objects to the scene, compute the intersection between a given ray and the objects in the scene, and calculate the color of a pixel on the screen.

The class has a constructor that takes a Vector object representing the position of the light source. It also has a vector of pointers to Geometry objects that represents the objects in the scene. The add_geometry method adds a new Geometry object to the scene.

The intersection method takes a Ray object and returns an Intersection object that represents the closest intersection of the ray with the objects in the scene.

The get_color method takes a Ray object representing the ray from the camera to a pixel on the screen, the current ray depth, and a boolean flag that indicates whether to include indirect light or not. It calculates the color of the pixel on the screen by performing lighting computations using the reflection and refraction. It first checks if the ray intersects with any objects in the scene. If not, it returns black. If it intersects with an object, it checks whether the object is a mirror or a transparent object. If it is a mirror, it computes the reflected ray and recursively calls itself with the reflected ray as input. If it is a transparent object, it computes the refracted ray and recursively calls itself with the refracted ray as input. If it is a diffuse object, it calculates the direct and indirect lighting components using the Lambertian model. It computes the direct component by checking if the point is in shadow or not. If it is not in shadow, it calculates the light intensity using the reflection model and adds it to the color. It computes the indirect component by shooting a new random ray around the normal at the point of intersection and recursively calls itself with the new ray as input. Finally, it returns the sum of the direct and indirect components.

3 "TriangleMesh.cpp"

The TriangleMesh class is the main class that represents the entire mesh object. It inherits from the Geometry class and contains several member variables, including the mesh color, scaling factor, translation vector, and details such as whether the mesh has mirror effect, is transparent, and its refractive index.

The TriangleMesh class has several member functions, including compute_bbox,

which computes the bounding box of a subset of the mesh defined by a range of triangle indices. This function takes two integer arguments that specify the starting and ending indices of the triangles to consider, and returns a Boundingbox object that contains the minimum and maximum coordinates of the bounding box.

The TriangleMesh class also has a function called BVH that builds a binary tree structure that is used to accelerate ray tracing computations. This function takes a node pointer argument that represents the current node being processed, as well as two integer arguments that specify the starting and ending indices of the triangles to consider. The function recursively divides the triangles into two groups based on the longest axis of the bounding box and builds child nodes until each leaf node contains a small enough number of triangles.

Finally, the TriangleMesh class has a function called intersect that performs a ray-BVH intersection test to determine if a given ray intersects the mesh. This function takes a Ray object argument that represents the ray to test, and returns an Intersection object that contains information about the intersection point if one exists. The function uses the BVH structure built by the BVH function to efficiently search for intersecting triangles in the mesh.

4 Images

4.1 Lab 1

Sphere with mirror effect, transparent one, and one with none. Fresnel reflection taken into account by default.

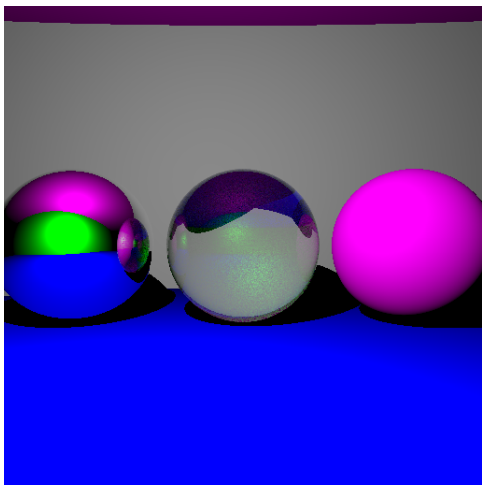

Figure 1: Scene=1; I=2e10; n_rays = 100; ray_depth = 5; indirect_light = false; anti-aliasing=false; time_taken=71.038ms

4.2 Lab 2

Application of indirect lights anti-aliasing on the above image.

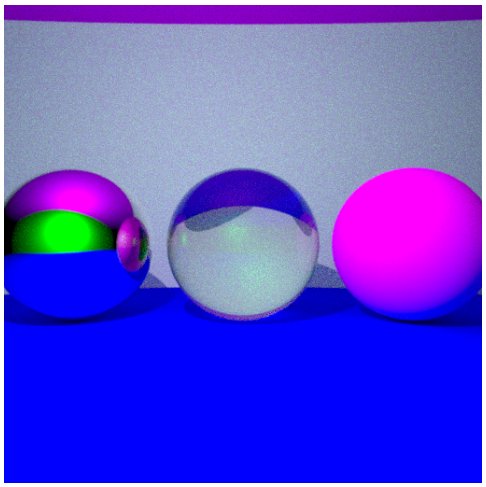

Figure 2: Scene=1; I=2e10; n_rays = 100; ray_depth = 5; indirect_light = true; anti-aliasing=true; time_taken=393.685ms

4.3 Lab 3 and Lab 4

These are two picture for the cat object both with BVH on. The first has has no additional features. The second is an adapted version so it fits the expected picture version with some activated optional features.

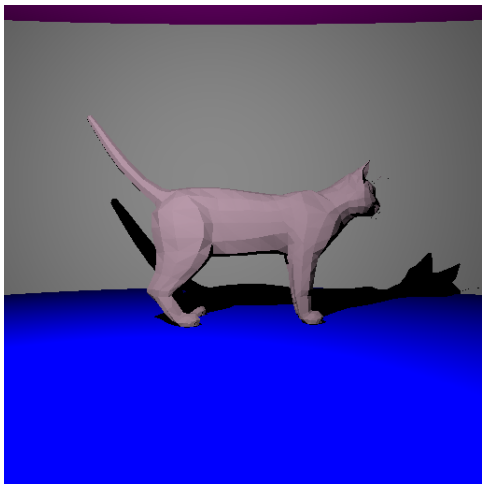

Figure 3: Scene=2; I=2e10; n_rays = 100; ray_depth = 5; indirect_light = false; anti-aliasing=false; BVH=true; time_taken=172.639ms

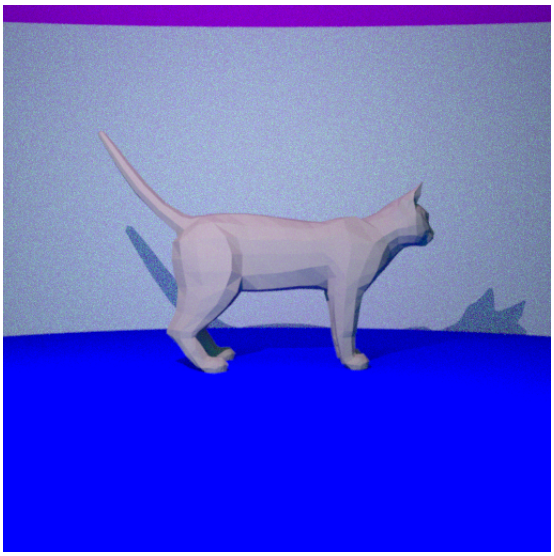

Figure 4: Scene=2; color=(0.3, 0.2, 0.25); I=3E10; n_rays = 100; ray_depth = 5; indirect_light = true; anti-aliasing=true; BVH=true; time_taken=3.248.245ms

5 Resources

The development of the code for this project was largely inspired by various resources. Firstly, the tutorial provided a basis for the code which was later

adapted to fit the classes introduced and the parameters chosen. Additionally, code was copied from the lecture notes. When faced with challenges, I turned to Google to find similar functions and compare them to what I had. I also referred to public Github repositories such as "https://github.com/faraheleuch" for ideas and guidance. ChatGPT played a significant role in helping me understand and debugg many errors that were encountered during development. While I couldn't trace back all the resources I used, some of them date back to the beginning of the second semester. Furthermore, I utilized ChatGPT to correct spelling, improve the use of professional language and to make the project report clearer.