

Assignment 3 : Ray-based Rendering and Loop Subdivision

NAME: SU'AN XIA

STUDENT NUMBER: 18047482

EMAIL: XIASA@SHANGHAITECH.EDU.CN

1 INTRODUCTION

1.1 Phong lighting model and shaders

Requirement: A ray-based rendering program.

Main process:

- (1) Generating camera rays.
- (2) Ray intersection test with a single triangle.
- (3) Building a uniform grid for triangle meshes as the acceleration structure for ray-tracing.
- (4) Ray intersection test with triangle meshes using uniform grid.
- (5) Phong reflection model.

1.2 Texture and shaders

Requirement: A subdivision algorithm that smoothes meshes.

Main process:

- (1) Subdivide closed surface;
- (2) Subdivide non-closed surface;

optional:

- (1) Reduce the time complexity to $O(p^2 + f)$.
- (2) Reduce the space complexity to $O(p^2 + f)$.

2 IMPLEMENTATION DETAILS

2.1 Generating camera rays

In camera.hpp, we only need to complete the Class 'Camera' and the function 'Ray generateRay(int dx, int dy)'.

2.2 Ray intersection test with a single triangle

In triangleMesh.hpp, we need to complete the function 'bool raySingleTriangleIntersection(Interaction& interaction, const Ray& ray, int v0_idx, int v1_idx, int v2_idx)'

In this function, we need to calculate the value 't,v,w,n,e,d' following the formula. Then according the value of 'd' and some other limits, we choose to return false or true.

2.3 Building a uniform grid

In triangleMesh.hpp, we need to complete the function 'void buildUniformGrid()'

First we need to determine the appropriate dimensions of the grid by the formula.

Then we loop over each triangle's bounding box to see which grid

will overlap with the triangle by calculation 'x * gridDim.y() * gridDim.z() + y * gridDim.z() + z'

2.4 Ray intersection test

In triangleMesh.hpp, we need to complete the function 'bool rayIntersection(Interaction& interaction, const Ray& ray) override'

The key idea is how we maintain the 'tMaxX tMaxY XtMaxZ' when we travers the light. Simply, we only update the smallest one by adding to the corresponding tDelta.

Also we need to pay atteneion to the situations where tMax is out the grid.

In each non-empty grid, we do the raySingleTriangleIntersection one by one.

2.5 Phong reflection model

In directLightingIntegrator.hpp, we need to complete the function 'Eigen::Vector3f radiance(Interaction* interaction, Ray * ray) override'

It is similat to what we did in assignment 2.

2.6 build data struct

In refine.hpp, we need to complete the function 'void ref::build_data_struct()'. We keep 3 data,'P_list L_list F_list'.

First we loop the 'out_v_index' to fullfill the P_list.

Then wee loop the face to fullfill the L_list. If one line is repeated, then it is not a border line.

Last we loop the L_list to chech if it is a border line. If it is a border line, then we mark the corresponding points border.

2.7 loop subdivision

In refine.hpp, we need to complete the function 'void ref::loop_subdivision()' and 'void subdivision(int division_time)' in triangleMesh.hpp.

In additional to what we have, we need to keep 3 more data struct 'P_new_list L_new_list F_new_list'.

First we calculate the even points. It is easy by formula.

Then we calculate the odd points. It is also easy by formula.

Afterwards, we loop each face to complete the information of each new line and new face.

Last we loop the line to complete the 'border' information of certain points.

In the triangleMesh.hpp, what we should is to determine the division_time and coduct the specific alogrithm by cycle.

3 RESULTS

Fig. 4. `division_time = 3`