CS100433
Computer Graphics

Assignment 4

Programming

- 1. Write a pseudo code to implement a mesh query, i.e. given a vertex *v*0, find all its surrounding vertices, by using the half-edge data structure (according to the following definition).

```
struct Halfedge {
         Halfedge* next;
         Halfedge* opposite;
         Face* incident_face;
         Vertex* incident_vertex;
}

struct Face{
         Halfedge * halfedge;
}


struct Vertex{
         Halfedge* incident_halfedge;
}
```

Programming

- 2. Write a pseudo code to draw a Bezier curve give four points P_0, P_1, P_2, P_3.

Programming

- 3. Write a pseudo code to implement the ray-tracing with BVH optimization (one ray per pixel and no sampling for diffuse).

Answer

- 1. Write a pseudo code to implement a mesh query, i.e. given a vertex *v*0, find all its surrounding vertices, by using the half-edge data structure (according to the following definition).

```
struct Halfedge {
         Halfedge* next;
         Halfedge* opposite;
         Face* incident_face;
         Vertex* incident_vertex;
}

struct Face{
         Halfedge * halfedge;
}


struct Vertex{
         Halfedge* incident_halfedge;
}
```

```
Halfedge*  hl = v0->incident_halfedge;
Halfedge*  nexHl = hl->next;
vector<Vertex*> vertArray;
vertArray.push_back(nexHl->incident_vertex);
while{nexHl->opposite != hl}
{
   nexHl = nexHl->opposite->next;
   vertArray.push_back(nexHl-
>incident_vertex);
}
return vertArray ;
```

Answer

- 2. Write a pseudo code to draw a Bezier curve give four points P_0, P_1, P_2, P_3.
- de Casteljau
- Berstein
- Or BGT

Programming

- 2. Write a pseudo code to implement the ray-tracing with BVH optimization (one ray per pixel and no sampling for diffuse).

```
given the BVH tree bvh
def traverse_recursive(r, node, nodeList):
    if r intersect node:
        if bvh.isLeaf(node):
            nodeList.add(node, distance)
        else:
            childNodeL = bvh.LeftChild(node)
            childNodeR = bvh.RightChild(node)
            traverseRecursive(r, childNodeL, nodeList)
            traverseRecursive(r, childNodeR, nodeList)
```

```
given a ray r
def ray_tracing(r):
    set an empty nodelList
    traverse_recursive(r, bvh.root, nodeList)
    sort nodes in nodeList by its distance
    while(nodelList is not empty):
        if r intersect with the geometry in the closet node in the nodeList:
            shoot the reflection ray rr, the shadow ray sr
            return ray_tracing(rr) + shadow_ray(sr)
        else:
            nodeList pop out the first node
```