A5 Project Proposal
Title: Ray Tracer
Name: Amr Abouelkhair
Student ID:20638554
User ID: aabouelk

Final Project:

Statement :

The final scene to be rendered is an animated scene. This requires both the animation feature, acceleration feature and adaptive anti-aliasing in order to render a big number of frames in a reasonable time frame. The scene starts with a bird eye view of the Pyramids of Giza and the Sphinx. This will be mostly done using texture maps. The view then descends and looks straight at the greatest pyramid, and then walks into it. Inside is a corridor with the walls rendered using texture mapping. The corridor leads to a room with a lot of artifacts. These artifacts will be using the reflection, refraction, bump mapping and Perlin Noise[9].

Technical Outline :

My main feature is animation. I am going to render a big number of frames and stitch them together in order to create my animated movie. In order to be able to do that in a reasonable time frame I am going to need to implement some algorithm of ray tracing acceleration. I am going to combine spatial subdivision with adaptive antialiasing.
The spatial subdivision will allow me to detect intersections faster by recursively dividing my scene space into voxels [1d] until each voxel intersects with only one object from my scene. This will then allow me to avoid checking for intersection between my ray and objects far away with it as I will follow the ray through the different voxels testing it with objects only in the voxels it goes through, ignoring all other objects. This will require a bit of preprocessing time to sort the objects into the voxels. However it will make rendering a lot faster, specially for higher resolution scenes as I need to check for way less intersections per ray.

This preprocessing will also make my adaptive antialiasing easier as I can store in my voxel whether or not it contains an edge of one of the objects. If so I will cast more rays per pixel specially around the edge to display a finer less jagged edge and average the colours detected by these rays. However if an intersection occurs within a voxel that's completely covered with one object then 1 intersection calculation along with storing the result of that intersection would suffice to calculate the colour for all the rays going through that voxel.
Referenced below multiple references on the interaction between a ray and an octree [6, 10, 13]. Even though this interaction is the core to accelerating this process, I find my first challenge in implementing this technique is building the tree [12, 14, 15]. My initial intuition is that I would start by flattening my hierarchy and build it top to bottom, trying to balance it as much as possible. In addition, octree is just my initial choice of data structure from previous experiences. However, I am not set on it and I might change it if I find a more efficient data structure, specially when it comes to traversal.

The rest of the features of my ray tracer are as important to my scene as my acceleration method or maybe even more important. However, I don't think they'll be as challenging. Most of them are basically depend on recursively casting a secondary ray or rays from the point of intersection to either reflect or refract the original ray [1a, 1c, 3, 4b, 4d]. These rays usually get solid colours and then use combine them together and/or change their shade depending on the light sources and normal direction [4a]. However, for the remaining features instead of using a solid colour I will either get the colour of a ray using a picture file (Texture mapping) or using an improved noise signal (Perlin Noise) [2b, 4a, 9].

Bibliography :

1. **An Introduction to Ray Tracing**, Glassner, et al., 1989,
   a. 13-17 Reflection Rays,
   b. 17-29 Anti-aliasing,
   c .126-138 Reflection and transmission rays,
   d. 217-227 Oct Tree acceleration.
2. **Realistic Ray Tracing, 2nd Edition**, Shirley et al., 2003,
   a. 71-72 Motion Blur,
   b. 79-85 Texture Mapping.
3. **Fundamentals of Computer Graphics, 2nd Edition**, Shirley et al., 2005,
   Glossy Refraction 230-236
4. **Fundamentals of Computer Graphics, 3rd Revised Edition**, Shirley et al., 2009,
   a. Texture and bump Mapping 242-256,
   b. Refraction 303-307
   c. Animation 413-443
   d. Glossy Reflection 645-650
5. **A new space subdivision method for ray tracing CSG modelled scenes**, Arnaldi, et al., The Visual Computer 1987, pp. 98-108
6. **Analysis of an algorithm for fast ray tracing using uniform space subdivision**, Cleary, and Wyvill, The Visual Computer 1988, pp. 65-83
7. **Selective and adaptive supersampling for real-time ray tracing**, Jin, Bongjun, et al., 2009, Proceedings of the HPG 2009: Conference on High-Performance Graphics 2009. 117-125.
8. **Two methods for improving the efficiency of ray casting in solid modelling** Bronsvoort, et al., 1984.
9. **Improving Noise** Perlin, Proceedings of SIGGRAPH '02 pp. 681-682
10. **The SMART navigation of a ray through an oct-tree**, Spackman, and Willis, Comput. & Graphics, Vol 15, No. 2, pp. 185-194, 1991.
11. **Reducing Render Time in Ray Tracing by Pixel Averaging** Behmanesh, et al., IJGCA Vol.2 No.4, 2012.
12. **On Building fast kd-Trees for Ray Tracing, and on doing that in O(N log N)**, Wald, and Havran, IEEE Symposium on Interactive Ray Tracing, pp. 61-69, 2006.
13. **Implementing Ray Tracing with octrees and neighbor finding**, Samet, Computer & Graphics, Vol13, Issue 4, pp. 445-460. 1989.
14. **A fast SAH-based construction of Octree** Yang, X., et al., 2009.
15. **On the Fast Construction of Spatial Hierarchies for Ray Tracing**, Havran, et al. IEEE Symposium on Interactive Ray Tracing, pp. 71-80, 2006.

Objectives:

Full UserID:aabouelk **Student ID:20638554**

___ 1: Final Scene.

___ 2: Glossy Reflection.

___ 3: Refraction.

___ 4: Glossy Refraction.

___ 5: Bump Mapping.

___ 6: Perlin Noise.

___ 7: Texture Mapping.

___ 8: Adaptive Antialiasing.

___ 9: Ray tracing Acceleration using spatial subdivision (OctTree).

___ 10: Animation.

A4 extra objective: Reflection

Extra Objectives:

___ 1: Extra primitives.

___ 2: PhotonMapping.

___ 3: Motion Blur.

___ 4: Soft Shadows.

___ 5: Radiosity.