A5 Project Proposal

Photorealistic Glass Rendering through Ray Tracing

Jacob Bush
20558637 · jbush

Final Project:

Purpose :

To use advanced algorithms and techniques to create a near photorealistic image.

Statement :

For this project, I will create a scene that achieves photorealistic elements. Primarily, I want to simulate glass; simulating something as simple as a wine glass combines a wide variety of graphical techniques, including reflection, refraction, and caustics, in addition to many that have been covered previously in the course. To supplement this core idea, texture and bump mapping will be used to create a more complicated background than has been used thus far. I will implement antialiasing and a depth of field in order to more closely match an actual photograph. In order to render an image such as this in a reasonable period of time, octrees will be used to accelerate the ray tracing process.

Technical Outline :

For new primitives, I will implement cylinders and toruses. I will use the equations provided in class, and also by referencing Watt & Watt [1] and Wolfram Mathworld [2]. This will require the creation of two new commands to the input language. The command `gr.cylinder` will create a cylinder, and the command `gr.torus` will create a torus. Non-hierarchical versions of these commands (`gr.nh_cylinder`, `gr.nh_torus`) may also be added for convenience.

For both texture and bump mapping, I will rely heavily on Watt & Watt [3], as well as referencing Blinn & Newell's 1976 paper [4] and Catmull's original dissertation on the topic [5]. Two new commands will be added to the input language to implement texture mapping. The command `gr.texturemap` will return a TextureMap object (by, for instance, reading an image file into a matrix of size equal to the image resolution). The object method `geonode:set_texturemap` will apply the given texture map to the GeometryNode. We will similarily define two new commands for bump mapping: `gr.bumpmap` to create a BumpMap object, and `geonode:set_bumpmap` to apply the bump map to a the GeometryNode. The essence of both texture and bump mapping is to split the geometry into patches, and then to apply a mapping to those patches. In texture mapping, the color of each patch is changed. In bump mapping, the normals of each patch are perturbed.

Reflection and refraction will be implemented by referencing the equations provided in class, and also by referencing Watt & Watt, Hall & Greenberg and Stanford's CS148 lecture slides [**?**][**?**][**?**]. Reflection will be implemented by tracing a new ray with angle of reflection equal to the angle of incidence. Refraction will be implemented by tracing a new ray with angle of refraction determined by Snell's law. Reflection and transmission coefficients, k_r and k_t, will need to specified for all materials in the scene.

Caustics will be implemented via photon mapping. For a point light source, photons are emitted uniformly (an easy calculation for this is given by Weisstein [**?**]) in all directions, with perhaps a modifier based on the relative density of the scene. The photons are traced with a randomized absorption strategy, and the absorption point of each photon is recorded. Jensen recommends using a kd-tree to store the photon map [**?**]. For every point in the scene, the density of the closest N photons to that point will provide an estimate of the illuminance at that point; call this $L_{caustics}$. Our rendering equation then becomes $L_r = L_{direct} + L_{specular} + L_{caustics}$ [**?**].

Acceleration will be achieved through the use of an octree. The space will be divided into octants, (eight regions) with each octant containing 0 or more objects. Octants with more than N objects in them will be further subdivided. Instead of testing a ray's intersection against all objects in the scene, we test it against only the objects in the regions that the ray will pass through, reducing computational overhead [6].

Antialiasing will be achieved via supersampling: specifically the jittering method described in class. Each pixel in the scene will be subdivided into an $N \times N$ grid, for some fixed N. For each region in

the subdivided grid, randomly select a point in that region, and trace a ray through that point. The resulting color of the pixel is the average of the colors returned by the N^2 rays.

The lack of depth of field in rendered images is a result of the camera being a pinhole. An area camera is simulated by randomly selecting N camera locations in a fixed area (jittering), and then averaging the resulting images produced at each camera location. Objects on or close to the focal plane of the camera will appear in approximately the same location in all of the images, and will thus be clear. Objects far away from the focal plane will appear in a different location in all the images, and will thus be blurred [**?**].

The final scene will not be technically challenging. It will be a culmination of the work done in the rest of the project. Textures will be procured from free online galleries. Polygon meshes will either be found online, or be made by myself in a program like Blender.

Bibliography :

[1] Watt, Alan H. & Watt, Mark (1992). Instersections: ray/quadratics. In Advanced Animation and Rendering Techniques: Theory and Practice. Addison-Wesley Professional. pp. 226-227.

[2] Weisstein, Eric W. Torus. From MathWorld–A Wolfram Web Resource. `http://mathworld.wolfram.com/Torus.html`

[3] Watt, Alan H. & Watt, Mark (1992). Mapping techniques: texture and environment mapping. In Advanced Animation and Rendering Techniques: Theory and Practice. Addison-Wesley Professional. pp. 178-201.

[4] Blinn, J. F., & Newell, M. E. (1976). Texture and reflection in computer generated images. Communications of the ACM, 19(10), 542-547. `https://doi.org/10.1145/360349.360353`

[5] Catmull, E. (1974). A subdivision algorithm for computer display of curved surfaces (PhD thesis). University of Utah.

[6] Watt, Alan H. & Watt, Mark (1992). The theory and practice of light/object interaction. In Advanced Animation and Rendering Techniques: Theory and Practice. Addison-Wesley Professional. pp. 33-64.

[7] Hall, R., & Greenberg, D. (1983). A Testbed for Realistic Image Synthesis. IEEE Computer Graphics and Applications, 3(8), 1020. https://doi.org/10.1109/mcg.1983.263292

[8] Fedkiw, R. (2017). Reflection and Transmission. Retrieved June 24, 2018, from `https://web.stanford.edu/class/cs148/pdf/class_11_reflection_and_transmission.pdf`

[9] Jensen, Henrik W. (2001). Realistic Image Synthesis Using Photon Mapping. A K Peters, Ltd.

[10] Waters, Z. (n.d.). Photon Mapping. Retrieved June 24, 2018, from `https://web.cs.wpi.edu/~emmanuel/courses/cs563/write_ups/zackw/photon_mapping/PhotonMapping.html`

[11] Weisstein, Eric W. Sphere Point Picking. From MathWorld–A Wolfram Web Resource. `http://mathworld.wolfram.com/SpherePointPicking.html`

[12] Watt, Alan H.& Watt, Mark (1992). Spatial coherence. In Advanced Animation and Rendering Techniques: Theory and Practice. Addison-Wesley Professional. pp. 241-248.

[13] Watt, Alan H.& Watt, Mark (1992). Depth of field. In Advanced Animation and Rendering Techniques: Theory and Practice. Addison-Wesley Professional. pp. 264-265.

Objectives:

Full UserID: jbush **Student ID:** 20558637

1. Primitives
2. Texture Mapping
3. Bump Mapping
4. Reflection
5. Refraction
6. Caustics via Photon Mapping
7. Acceleration via Octree
8. Antialiasing
9. Depth of Field
10. Final Scene

A4 extra objective: I did antialiasing as my extra objective for A4. I did not receive full credit for this objective since I did not end up demonstrating it in an image. Talking with Gladimir in class, he said that if I did not receive credit for this objective, then I could use it as an objective for the project. Since I did not receive full marks for this objective, I assume that I am allowed to implement antialiasing as one of my project objectives. If this is not the case, I will change the objective for another one.