CS488 A5 PROJECT DOCUMENTATION

YacRay

(*Yet Another CS488 Raytracer*)

Name: Lawson Fulton
Student ID: 20381453
User ID: ljfulton
April 3, 2015

Table of Contents

Summary . . . . **iii**

1.0 Manual **1**

1.1 Running Yacray 1

1.2 Creating Scenes 2

1.2.1 Modelling 2

1.2.2 Primitives 3

1.2.3 Materials 4

1.2.4 Lighting 5

1.2.5 Rendering 5

2.0 Implementation **6**

2.1 Source Code Considerations 6

2.2 Main Objectives 7

2.2.1 Reflection and Refraction 7

2.2.2 Soft Shadows and Area Lights 8

2.2.3 Glossy Reflections 9

2.2.4 Adaptive Anti-aliasing 10

2.2.5 Smooth Phong shading on triangle meshes 11

2.2.6 Texture and Bump Mapping 12

2.2.7 Tone Mapping 15

2.3 Additional Features 17

2.3.1 Kd-Trees 17

2.3.2 Depth of Field 19

2.3.3 Specular and Environment Mapping 19

2.3.4 Fractal Primitives - Menger Sponge 20

2.3.5 Multithreading 20

3.0 Results 21

4.0 Future Work and Potential Improvements 22

Asset Credits 24

References 25

Checksum 26

Summary

In this report I will present the documentation and implementation details for YacRay (Yet Another CS488 Raytracer). YacRay is my submission for the Winter 2015 CS488 final project. The goal of this project was to create a raytracer that implements most of the major foundational features for modern rendering. These objectives included the following:

- Reflective materials
- Transparent materials with refraction
- Soft Shadows
- Glossy Reflections
- Anti-aliasing using adaptive supersampling
- Smooth Phong shading on triangle meshes
- Texture Mapping
- Bump Mapping
- Tone Mapping
- Final scene that demonstrates the previous objectives

All objectives were successfully accomplished along with a number of additional features that will be detailed later. The first portion of this document will present instructions for using YacRay and creating accompanying scenes. The second part will present the software design considerations and technical details required when implementing each of the aforementioned objectives. The report will conclude by presenting results and considerations for future work.

I would like to thank Professor Baranoski for his valuable insights and encouragement, as well as my classmates for the many discussions during those long nights in the graphics lab.

1.0 Manual

This manual assumes that you already have a compiled YacRay executable for your platform. If you do not, please refer to the README included with the YacRay source code.

1.1 Running Yacray

Running YacRay is very straight forward. The executable may be run from any directory, and the only required argument is the path to a scene described in the lua language. Details for creating a scene will be covered in the following section.

```
$ ./rt <scene filename>
```

The only thing to keep in mind is that if your scene file refers to external paths then they will be relative to the directory that YacRay is run in. For example, if you would like to run the included final scene you would do the following:

```
$ cd A5/data/src
$ ../../rt final.lua
```

You should then scene some output corresponding to information about the scene being run, followed by a status bar indicating the progress of the render. When the render completes, the running time will be reported and the program should exit automatically. The output image can be found at the path specified in the scene file, typically in the same directory YacRay was run. In the previous example, you will find final.png in the current directory, which can then be viewed with your favourite image viewer supporting png images.

1.2 Creating Scenes

Creating scenes for YacRay is accomplished by writing lua scripts that are executed by the renderer to build up the scene in memory. The scene description language (hereafter referred to as SDL) is derived from the code we received in assignments 3 and 4. The following sections will present a description of all the commands in the SDL. Some of these descriptions have been sourced from the specifications presented in the assignment 3 and 4 outlines[1][2].

For an example of the SDL in action, please see the file final.lua in A5/data/scenes.

1.2.1 Modelling

These operations allow for hierarchical modelling in the scene. Transformations on any node will also be applied to it's children nodes.

- `gr.node(`*`name`*`)` - Return a node *`name`* that just contains a transformation matrix, which is initialized to the identity matrix. There must always be at least one root node in a scene which all other objects are added to.
- `gr.joint(`*`name`*`,{`*`xmin, xinit, xmax`*`},{`*`ymin, yinit, ymax`*`})` - Create a joint node with minimum rotation angles *`xmin`* and *`ymin`*, maximum rotation angles *`xmax`* and *`ymax`* and initial rotation angles *`xinit`* and *`yinit`* about the x and y axes.
- `pnode:add_child(`*`cnode`*`)` - Add *`cnode`* as a child of *`pnode`*.
- `node:rotate(`*`axis, angle`*`)` - Rotate node about axis (*`'x', 'y' or 'z'`*) by *`angle`* (in degrees).
- `node:translate(`*`dx, dy, dz`*`)` - Translate node by (*`dx, dy, dz`*).
- `node:scale(`*`sx, sy, sz`*`)` - Scale node by (*`sx, sy, sz`*).

1.2.2 Primitives

These operations allow for creation of the objects that can be rendered in the scene.

- `gr.plane(`*name*,*r*`)` - Return a plane (or more accurately a disk) with name *name* centered at the origin with radius *r* and a normal aligned with the positive *y* axis.
- `gr.sphere(`*name*`)` - Return a sphere with name *name*. The sphere will be centered at the origin with radius 1.
- `gr.nh_sphere(`*name*,(*x*,*y*,*z*),*r*`)` - Return a sphere with name *name*. The sphere will be centered at *x*,*y*,*z* with radius *r*.
- `gr.cube(`*name*`)` - Return a cube with name *name*. The cube will be centered at the origin with radius width and height 1.
- `gr.nh_box(`*name*,(*x*,*y*,*z*),*r*`)` - Return a box with name *name*. The box will have one corner at *x*,*y*,*z* and a diagonally opposite corner at *x+r*,*y+r*,*z+r*.
- `gr.mesh(`*name*,{*vertices*}, {*faces*}`)` - Create a polygonal mesh named name with the listed vertices and faces. The first list is a list of vertex coordinates, and the second list is a list of polygons. Each vertex is given as an (x,y,z) triple, and each polygon is a list of integer indices into the vertex list. Vertices are indexed starting at 0. It may be assumed that polygons are convex and planar. However, polygons may have an arbitrary number of vertices.
- `gr.obj_mesh(`*name*, *obj_path*`)` - Similar to `gr.mesh` but instead loads the mesh from a .obj stored on disk. The obj file must be in ASCII format, must have normals stored, and must be triangulated. This method for creating meshes is a much faster choice for rendering as it is able to use a simplified intersection calculation, as well as building a kd tree.
- `gr.menger_sponge(`*name*,*d*`)` - Return a fractal known as a menger sponge with name *name* centered at the origin with *d* iterations controlling the level of detail. An iteration

level of 0 will result in a cube, while each successive level will result in one additional subdivision. Read more about menger sponges here: `http://en.wikipedia.org/wiki/Menger_sponge`

1.2.3 Materials

Materials are applied to primitives to determine their colour, texture, and other visual properties.

- `gr.material({`*`dr,dg,db`*`},{`*`sr,sg,sb`*`},`*`p`*`)` - Return a material with diffuse reflection coefficients *`dr, dg, db`*, specular reflection coefficients *`sr, sg, sb`*, and Phong coefficient *`p`*.
- `gr.fancy_material({`*`dr,dg,db`*`},{`*`sr,sg,sb`*`},`*`p,r,ior,alpha,samples`*`)` - Similar to `gr.material` with the addition of reflectivity coefficient *`r`* which is currently unused as the specular coefficients control reflectivity. The parameter *`ior`* sets the index of refraction, *`alpha`* is a number from 0 to 1 controlling the transparency, and *`samples`* is an integer that sets the number of samples used when calculating the reflected colour (particularly important with low *`p`* values since they correspond to a more glossy surface).
- `node:set_material(`*`mat`*`)` - Give the node *`node`* material *`mat`*. Node materials can be changed at any time.
- `mat:set_texture_map(`*`image_path`*`)` - Give the material *`mat`* texture map corresponding to a png image at path *`image_path`*. The mapping of the texture depends on the uv mapping specific to each primitive.
- `mat:set_specular_map(`*`image_path`*`)` - Give the material *`mat`* a specular map corresponding to a png image at path *`image_path`*. Mapping is performed in the same manner as texture_map.

- `mat:set_bump_map(`*`image_path,scale`*`)` - Give the material *`mat`* bump map corresponding to a grayscale png image at path *`image_path`*. Mapping is performed in the same manner as texture_map. The intensity of the simulated displacement is controlled by a floating point number *`scale`*.
- `mat:set_fresnel(`*`amount`*`)` - Set the power of the fresnel effect for material *`mat`*.

1.2.4 Lighting

- `gr.light({`*`x,y,z`*`},{`*`r,g,b`*`},{`*`c0,c1,c2`*`})` - Create a point light source at (*`x,y,z`*) of intensity (*`r,g,b`*). The attenuation parameters *`c0, c1, c2`* specify the attenuation for the particular light source according to the formula $1/(c0 + c1 \cdot r + c2 \cdot r2)$.
- `gr.rect_light({`*`x,y,z`*`},`*`xlen,zlen`*`,{`*`r,g,b`*`},{`*`c0,c1,c2`*`},`*`samples`*`)` - Create a rectangular area light centered at *`x,y,z`* shining in the negative *`y`* direction with dimensions *`xlen`* and *`ylen`* in the *`x`* and *`z`* axis respectively. The other parameters mirror that of `gr.light` except for the addition of *`samples`* which is an integer specifying the number of samples used for soft shadow calculations. This light source is visible to the camera.

1.2.5 Rendering

- `gr.render(`*`node,filename,w,h,sslevel,`*
  `aperature,focallen,usetonemap,Lwhite,a,`
  `eye,view,up,fov,ambient,lights,envmap``)` - Render an image with dimensions *`w`*×*`h`* and save the result to *`filename`*. Set the supersampling subdivision level with *`sslevel`*. Control depth of field parameters with *`aperature`* and *`focallen`*. Tonemapping is turned on or off with a boolean passed to *`usetonemap`* and its parameters are controlled by *`Lwhite`* and *`a`* which will be explained in the implementation section on tonemapping. The camera is to be located at position *`eye`*, looking in direction *`view`*

with ***up*** pointing up (all of these quantities are three-vectors). A field-of-view of ***fov*** degrees is to be used. The ambient light should have an intensity of ***ambient*** (also a three-vector). All lights to be used in raytracing are listed in ***lights***. The final optional parameter ***envmap*** may contain the path to an image that will be spherically mapped to an infinitely large sphere enclosing the entire scene. If no environment map is supplied the background will default to black.

2.0 Implementation

In this section of the report, I will provide a brief description for each feature implemented in YacRay. This description will include details about the algorithm, data structure, and source.

2.1 Source Code Considerations

Throughout this portion of the report, I will often make references to several concepts defined as classes within YacRay's source code. The first being the `Ray` object, which is identical to the mathematical concept of a ray taht possesses both an origin point and a unit direction vector, along with additional functions that allow me to easily switch between parameterized and explicit representation of points along the ray.

The second common object I refer to is the `Intersection`. An intersection is simply a helpful container that holds all the information I may need later when calculating the resulting colour of a ray tracing calculation. For example, the intersection contains a pointer to the object hit, the point itself, the ray, the t value to determine the point of intersection along the ray, the original surface normal, etc...

2.2 Main Objectives

2.2.1 Reflection and Refraction

Reflection in its most basic form is fairly straightforward to implement, and as such, I did not require a source other than those notes presented in class. Originally, reflectivity was controlled by an additional material parameter. However, once glossy reflections were implemented (as described later) the model was generalized to have the specular coefficients control the intensity of the reflection, so that any object with non-zero specular coefficients will generate reflected rays.

If a reflected ray is to be generated, it shares the origin of the intersection point, and the direction is calculated by the following.

```
Vector3D reflDir = reflect(-i.ray->direction(), normal);
```

Reflect calculates a reflected vector according to $2(\hat{d} \cdot n)(n - \hat{d})$ where $\hat{d}$ is the first parameter and n is the second. The colour of the resulting ray is then calculated by passing it to a generic `traceRay()` function that is used to calculate the colour of all rays, including the primary rays.

To avoid infinite loops, a maximum recursion depth can be specified in `options.hpp`. The default is 3, since it seems to be the lowest number that can consistently produce realistic images except for degenerate cases such as a hallway of mirrors. If a ray is traced beyond the max recursion depth, the background colour in that direction is returned for the ray, since returning black alone may cause artifacts in environment mapped scenes.

Refraction is computed in a very similar manner to reflection. The only difference is that the direction of the refracted ray depends on the the index of refraction of the two mediums.

The formula for calculating the refracted ray is as follows:

$$s = \frac{n_1}{n_2}(\hat{d} - (\hat{d} \cdot n)n)$$

$$d_r = s - \sqrt{1.0 - ||s||^2}n$$

However, if $1 - |s|^2 < 0$ then the ray is totally reflected back internally, known as total internal reflection. In this case no further refraction calculation is carried out. Other than that, special care must be take to keep track of which medium the ray is leaving, and which medium the ray is entering. YacRay assumes that all rays start in a medium with an index of refraction equal to 1.

The source code pertaining to these features can be found in `material.cpp`.

2.2.2 Soft Shadows and Area Lights

The approach used for calculating soft shadows is based off of the seminal article by Cook, Porter, and Carpenter[3]. Essentially, a light source is defined over some area and when calculating the light contribution from that light, many samples are used from different points on the light source and the results are then average. The result of this is that many shadow checks will be performed on a slightly perturbed ray, causing a shadow only some of the time and hence soft edges.

In YacRay, soft shadows are achieved by defining a rectangular area light with the - `gr.rect_light()` function. In the code, `RectLight` provides a `getSample()` function which returns a random point uniformly distributed within the rectangle. This point is then used for the typical lighting calculation, including shadow test, and the result is averaged.

In addition to the soft shadows, another benefit of using an area light is that you can see the light in the rendered images. This comes in handy when doing glossy reflections, as the highlight will be more realistic than the Phong model for specular highlights. I achieved this effect by making the `Light` class inherit from `Primitive` and giving it a material as well as an intersection function.

A potential future improvement to this method for creating soft shadows would be a more complex sampling function. Using the current uniform distribution function, it is possible to get clumping analogous to random sampling for anti-aliasing. This could be remedied by uniformly subdividing the light source and using jittered sampling from within each region of the grid.

The code for this objective can be found in `computeColour()` in `material.cpp` and - `Light.cpp`.

2.2.3 Glossy Reflections

In reality, it is very rare to see an object that reflects light perfectly. Most reflective surfaces are somewhat blurry, or have "gloss". Glossy reflections is a technique used to simulate these surfaces that are reflective but scatter light slightly when reflected. The concept for doing glossy reflections is very similar to that described for soft shadows in the previous section, and is actually based on the same paper[3]. However, the formula used for calculating the distribution of the reflected rays was given in class and appears as follows:

$$\alpha = \cos^{-1}(1 - x_1)^{\frac{1}{p+1}}$$

$$\beta = 2\pi x_2$$

Where $x_1, x_2 \in [0, 1]$ are uniformly distributed random numbers, p is the shininess coefficient given in the material parameters, and α, β are spherical coordinates for a unit vector in the hemisphere around $[0, 1, 0]$, call it the pertubation vector v_p. Once this vector is calculated and converted to cartesian coordinates, it is necessary to perturb the original perfectly reflected vector described in 2.2.1 of this report. In YacRay, this is achieved by first recording the spherical coordinates of the original reflected vector, and then using those values to build a rotation matrix that will rotate the vector $[0, 1, 0]$ into the same direction as the reflected ray. When this matrix is applied to v_p, we are left with a reflected vector that has been perturbed according to a cosine weighted distribution and we can continue the reflection calculation as in 2.2.1.

The result of all this is a surface that is completely diffuse if $p = 0$ and perfectly reflective as $p \to \infty$. Of course, the lower the shininess factor you choose, the more samples you will require to achieve a smooth looking image. Hence, I have included a material parameter for setting the number of samples to use in this calculation. I have found that to achieve a perfectly reflective surface, you typically must choose a $p > 1000000$. One nice side effect of implementing glossy reflections is that it supersedes the Phong model for specular highlights previously used.

The code for this objective can be found in `computeReflectedContribution()` within `material.cpp`.

2.2.4 Adaptive Anti-aliasing

In assignment I implemented uniform grid supersampling to achieve anti-aliasing. In this project I extended the technique to use adaptive supersampling for antialiasing using the technique described by Whitted[4].

The approach is as follows. First, sample every pixel at all four corners. Note that this will necessarily produce duplicate rays, so I use a shared array to store the results of the calculations that I check before sending a new ray. Once the four corners are sampled, I check to see if they differ by a significant amount by checking to see if components of one corner is within ϵ of the every other corner, if not, then I subdivide the pixel and fire more rays. I chose $\epsilon = 0.02$ through trial and error. The final result is the average of all the rays fired so far.

In the original paper, the method is written to recursively subdivide the pixel. I limited the pixel to a single subdivision, as I found that in practice this looks almost as good as 4x4 uniform supersampling.

The code for this objective can be found in `renderer.cpp` at `computePixelColour()`.

2.2.5 Smooth Phong shading on triangle meshes

In assignment 4 we implemented meshes with flat shaded faces. This is because each face is a plane segment with a constant normal. However, if you would like to create the illusion of a smooth surface with this technique, you will have to use a mesh with many hundreds of thousands of faces which is obviously computationally undesirable. One solution to this problem is to calculate normals at every vertex of a mesh and then interpolate between them across a face of the mesh.

In YacRay, the `obj_mesh` primitive supports loading triangle based meshes with precomputed vertex normals from an obj file. Here I would like to give credit to Syoyo Fujita for the use of his obj loading library "tiny obj loader" Licensed under 2-clause BSD license which can be found at `https://github.com/syoyo/tinyobjloader`. Just to be clear, I used Fujita's obj loading code only to read the obj file data into memory which I then moved into my own data structures for rendering.

Once the face of intersection with a ray on the mesh is determined, the normal must be calculated. For this, barycentric interpolation between the 3 normals on the corners of the face is used. Essentially, the normals are combined in proportion to how close the intersection point is to each of the corners. Once this interpolated normal is calculated it can be used as usual in the following phong model lighting calculations, resulting in a smoothed appearance over the faces of the mesh.

There are two minor problems with this approach. Firstly, this is only a superficial effect and does not change the underlying geometry, therefore, if you look at the silhouette of a mesh, you will potentially see the sharp nature of the faces. Secondly, shadow calculations still take place using the underlying flat faceted mesh. This can result in sharp shadow lines on the mesh which may look unnatural with Phong shading, however, soft shadows largely ameliorate this issue.

Please see `getBarycentricCoordinates()` in `algebra.cpp` for the details of this computation.

2.2.6 Texture and Bump Mapping

Texture mapping is the process of replacing the diffuse colour at a point on the surface of a primitive with a colour sampled from an image. The notes presented in class were sufficient for me to determine how to implement this feature.

The first step in implementing texture mapping is to obtain a uv-coordinate for the point of intersection. The name comes from the fact that we parameterize the 2D image with $u, v \in [0, 1]$ where u and v correspond to the x and y coordinates respectively. Every primitive has its own unique uv-mapping that maps from a point $[x, y, z]$ on its surface to a point $[u, v]$ in the texture map. In the case of the sphere, the mapping is as follows:

$$u = \frac{1}{2} + \frac{\arctan(z, -x)}{2\pi}$$

$$v = \frac{1}{2} - \frac{\arcsin(y)}{\pi}$$

Every other primitive has a similar analytic mapping, except for the case of `obj_mesh` which interpolates the uv-coordinates defined per vertex in the same way as normal interpolation described in section 2.2.5.

Once the uv-coordinate is obtained, a colour to replace the diffuse component in the lighting calculation can be sampled from the texture image. The sampling method used can significantly affect the quality of the mapped image. In the case of YacRay, I use bilinear interpolation as described in class. Bilinear interpolation works by considering the 4 closest pixels to the uv-coordinate. The colours of the pixels are then combined proportionally to how close they are to the sample point.

A technique that shares many commonalities with texture mapping is known as bump mapping. Bump mapping allows for the simulation of 'bumpy' surfaces without actually changing surface geometry. My implementation of bump mapping draws heavily on the techniques first described by Blinn[5] where a grayscale image is used to specify a simulated offset from the surface of an object rather than a colour. In addition to that original paper, I used a textbook recommended by Prof. Baranoski for additional clarification[6].

Bump mapping is identical to texture mapping up until the point you have your uv-coordinate and corresponding sample from the mapped image. However, in addition to these, we must also calculate the partial derivatives with respect to u and v of both the bump map $B(u, v)$ and the 2D parameterization of the 3D surface of the primitive $O(u, v)$. Both B_v and B_u can be easily calculated numerically as follows:

$$B_u = \frac{B(u + \epsilon, v) - B(u - \epsilon, v)}{2\epsilon}$$

And B_V similarly. One important sticking point is how to determine an appropriate value for ϵ since choosing a constant value will yield variable results depending on the resolution of the mapped image. Through a process of trial and error I determined that $\epsilon = 2/\max(width, height)$ usually gives nice results. Now, calculating O_u and O_v is somewhat more difficult as it might not always be easy find and analytic expression for $O(u, v)$. My solution to this is to explicitly calculate tangent vectors in the direction of $[\partial u, \partial v]$. In the case of the sphere, I use cross products between the surface normal (equivalent to the surface point in object space) and the axis as follows.

```
if(normal.z() > 0) {
Ou = cross(normal, Vector3D(0,1,0));
Ov = cross(normal, Vector3D(1,0,0));
} else {
Ou = cross(normal, Vector3D(0,1,0));
Ov = cross(Vector3D(1,0,0), normal);
}
```

Although this is an approximation that will fail in certain cases, it seems to work quite well in practice. Once all of the aforementioned values are calculated, the final displacement vector can be calculated:

$$A = N \times O_v, B = N \times O_u$$

$$D = B_u A - B_v B$$

$$N' = N + D$$

Where N is the original surface normal and N' is the new perturbed normal used in all subsequent lighting calculations.

One important note about bump mapping is that using bilinear interpolation can cause a substantial loss of sharpness in the fine details of the bumps. Because of this, I use simple nearest integer sampling for obtaining the colour in the bump map image.

The code for this objective can be found in `material.cpp`.

2.2.7 Tone Mapping

When YacRay finishes determining the colour for all of the primary rays in an image, the resulting intensity values for each component of the pixel colours are in the range $[0, \infty)$. However, when these intensities are translated into values in $[0, 255]$ for saving to a png, the original range is simply truncated to $[0, 1]$. If careful attention is paid attention to the lighting levels in a scene, this may not be an issue. However, if for example your scene has many very bright lights, you could end up with 'hot spots' in your image that leave artifacts when truncated. The solution to this is known as tone mapping, which is the process of mapping $[0, \infty) \rightarrow [0, 1]$ using a function the preserves that natural appearance of the image.

YacRay implements a popular tone mapping operator described by Reinhard, Stark, Shirley, Ferwerda[7]. The operator described in the paper has two steps, the second of which is significantly more complex for little benefit in most scenes. The first step is much simpler and is often referred to as the Reinhard tone mapping operator. This first step is what YacRay implements and which I will describe here.

The first step is to convert all of the pixel rgb triples into scalar luminances with the following formula:

$$L_w(x, y) = 0.2126r(x, y) + 0.7152g(x, y) + 0.0722b(x, y)$$

The operator acts on luminance values and once the values have been mapped, they are translated back into rgb space. Then a value approximating the key of the image is calculated with the following:

$$\bar{L}_w = \exp\left(\frac{1}{N}\sum_{x,y}\log(\delta + L_w(x,y))\right)$$

In Reinhard's paper, the $\frac{1}{N}$ term is outside the exponent, however this results in absurd values, and other implementations seem to use the above form with the fraction inside the exponent. The user can then modify the key of the image by modifying the `a` parameter mentioned in the manual, giving a new luminance value of:

$$L(x,y) = \frac{a}{\bar{L}_w(x,y)} L_w(x,y)$$

Reinhard considers a good default value to be middle-grey for $a = 0.18$. Then the final operator is given by:

$$L_d(x,y) = \frac{L(x,y)\left(1 + \frac{L(x,y)}{L^2_{white}}\right)}{1 + L(x,y)}$$

Where L^2_{white} is the smallest luminance that will be mapped to pure white. This parameter can be tuned as a fraction of the maximum luminance in the scene and is controlled by `Lwhite` mentioned in the manual. Through experimentation I have found that a value between 0.5 and 1.0 is ideal. Once the mapping is finished, the fractional difference between the original and new luminance is used to scale intensity of the pixel colours.

Since this algorithm only requires a few simple calculations for each pixel of the image, it incurs very low cost when enabled in the renderer but often gives somewhat improved contrast to the final image.

The code for this objective can be found in `light.cpp`.

2.3 Additional Features

After completing the primary objectives of my project, I had some spare time to implement additional features to enhance my final scene. I will briefly describe those features in this section.

2.3.1 Kd-Trees

Some form of spatial subdivision is essential if you wish to render complex meshes in any reasonable time frame. I originally planned on implementing uniform spatial subdivision to accelerate the rendering of large meshes as I expected it to be the simplest spatial partitioning technique. However, after a failed attempt I decided to look into kd-trees and realized that it is fairly simple to do a straight forward implementation using my existing bounding box and triangle intersection code.

Due to the short timeframe I had available to finish my final scene, I based my implementation off of a blog post by Emma Carlson[8]. Though, I did significantly modify the implementation to work with my existing code.

When building the tree, the basic idea is to split all of your triangles into two groups, and place a bounding box around each of them. The process is then repeated on each of these new groups of triangles until some stopping criteria is reached. When testing for an intersection with the mesh, you first test for an intersection with the top level bounding box, if it hits, then you test with the left and right children, if it hits one or more of those, you test against their children, etc, until you reach a leaf node. Once at a leaf node, you test for intersection with each triangle contained in the leaf. This can be easily implemented in a recursive manner. As usual, the closest intersection wins.

Below you can see how using kd-trees stacks up against naive iteration for a mesh undergoing successive iterations of catmull-clark subdivision. Note that this includes the time required to build the kd-tree. In each iteration, the time to actually render the scene using the kd-tree stays nearly constant. This indicates that being more careful when building the tree could incur significant performance gains.

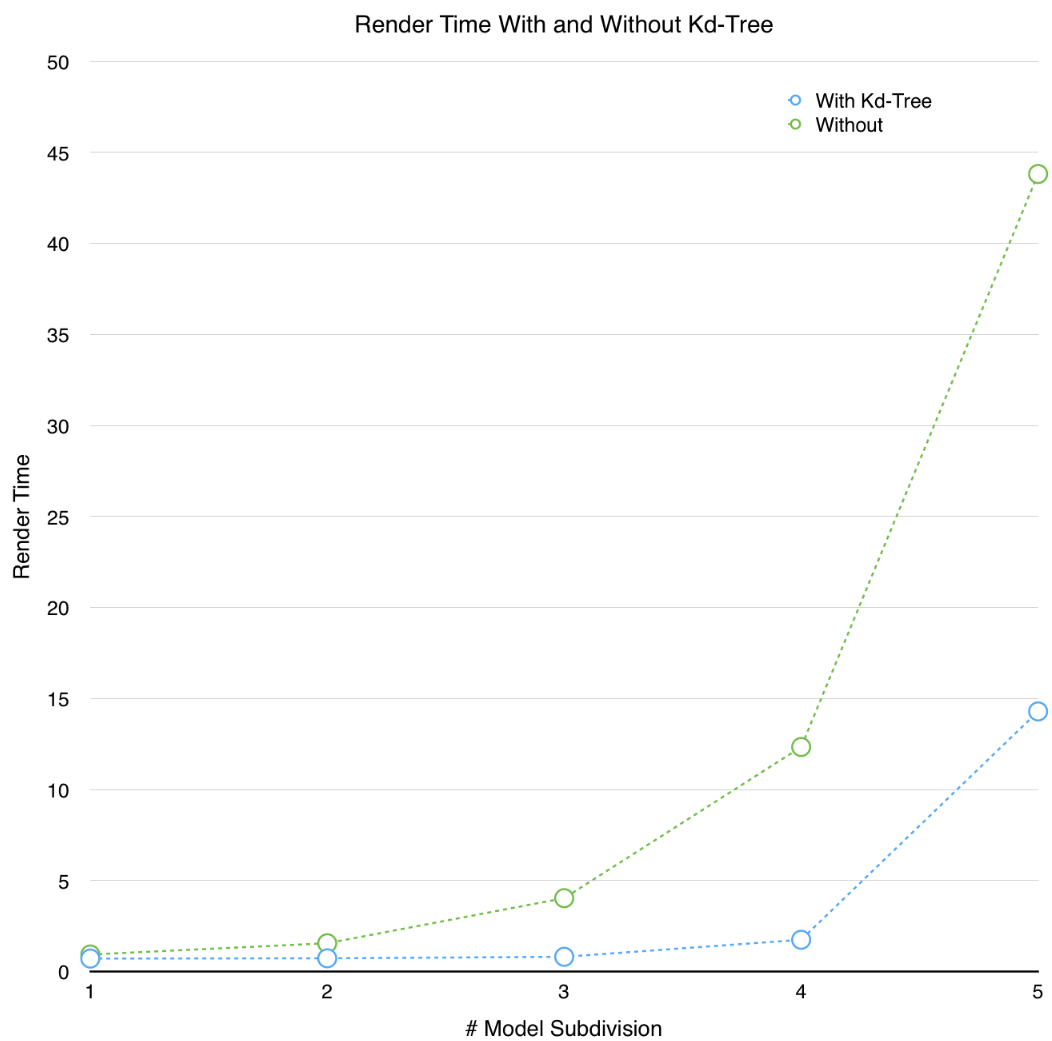

Figure 1: Kd Tree Performance

2.3.2 Depth of Field

Depth of field is a subtle effect that can greatly increase the photorealism of a scene. Again, pressed for time, I implemented depth of field based off of a high level explanation on stackoverflow[9] and continuing off of the concepts presented in the distributed raytracing paper[3].

The basic idea is that I generate primary rays as usual, but then use the `focalLen` to find a point on that ray called the focal point. Then, I generate a number of uniformly distributed random points in a disk of radius `aperture` around my original look_from, and perpendicular to the view direction. These points serve as origins for new rays that are generated aimed at the focal point. These new rays are traced as usual, and their results are averaged. The end result is an image in which objects get blurrier if they are closer or farther than the focal length. The effect can be increased by increasing the size of the aperture.

Like other effects generated using the principles of distributed raytracing, this feature could be improved by a better sampling scheme. The code for this objective can be found in `renderSlice()` of `renderer.cpp`.

2.3.3 Specular and Environment Mapping

These features are perhaps the most straight forward to describe since they are based off of previously implemented features.

Specular mapping is identical to texture mapping, except that instead of modifying a materials diffuse component, this map modifies a materials specular component, and hence reflectivity. The effect of this mapping can be seen in the varying reflectiveness of the wood surface in my final scene.

Environment mapping is achieved by modifying the background function which takes a vector direction and returns a colour. The function works by interpreting the vector direction as a point on a unit sphere, and the colour is then computed by doing the usual uv-mapping and texture lookup that is done for a normal texture-mapped sphere. The resulting impression is that of an infinitely large sphere surrounding the scene with an environment projected onto it. This feature is perhaps the most valuable enhancement in terms of increasing the photorealism of a scene.

2.3.4 Fractal Primitives - Menger Sponge

This feature was just for fun. I had been looking at pictures of 3D fractals and thinking about how I might raytrace one without ray-marching. The menger sponge is the result, since I realized I could easily create a recursive intersection function using only my preexisting bounding-box code. The intersection function works by first checking to see if the ray intersects the top level bounding box, a unit cube. If it does, then it checks for an intersection with a menger sponge defined at 20 other predefined locations relative to the current top level box. This process continues until a predefined depth has been reached and the closest intersection is returned, it is as simple as that.

The code for this objective can be found in in `primitive.cpp`.

2.3.5 Multithreading

Ray tracing is well known for its ability to be easily parallelized. YacRay uses multithreading to render strips of the image simultaneously in different threads. The naive implementation where the image is split into N slices delegated to N threads works, but is subject to bottleneck behaviour if the scene has 'hotspots' of complexity. YacRay's solution is to split the image into many more slices than the number of threads. These image slices are then placed in a queue to wait for an available thread. The results on performance on my final scene can be seen below:

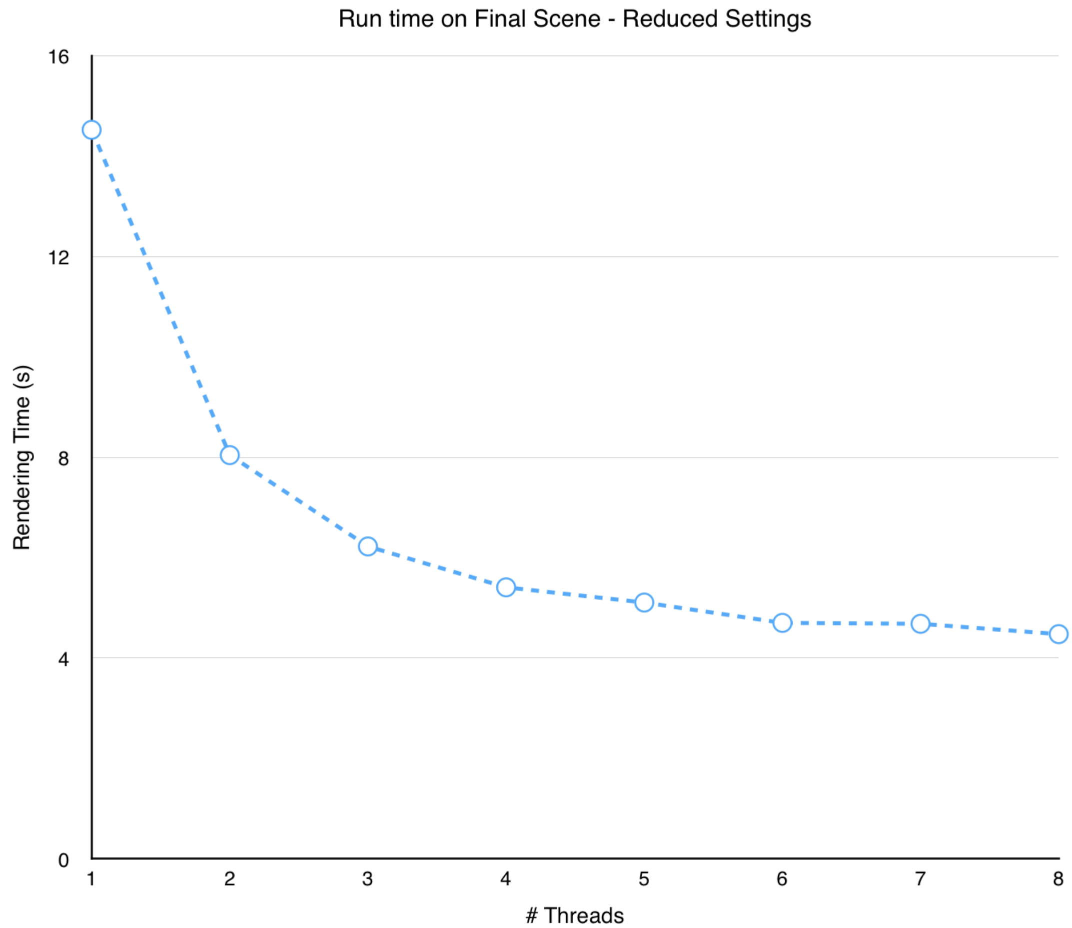

Figure 2: Multithreading Performance

3.0 Results

As has been shown, all objectives have been accomplished, along with a number of additional features. The final image is a rendering of four 'graphics primitives' that Prof. Baranoski often brought to class to demonstrate concepts, a golf ball, an apple, a water bottle, and 'the normal'. All of the features described in this report (save for the menger sponge) are present in this image, see if you can spot them all!

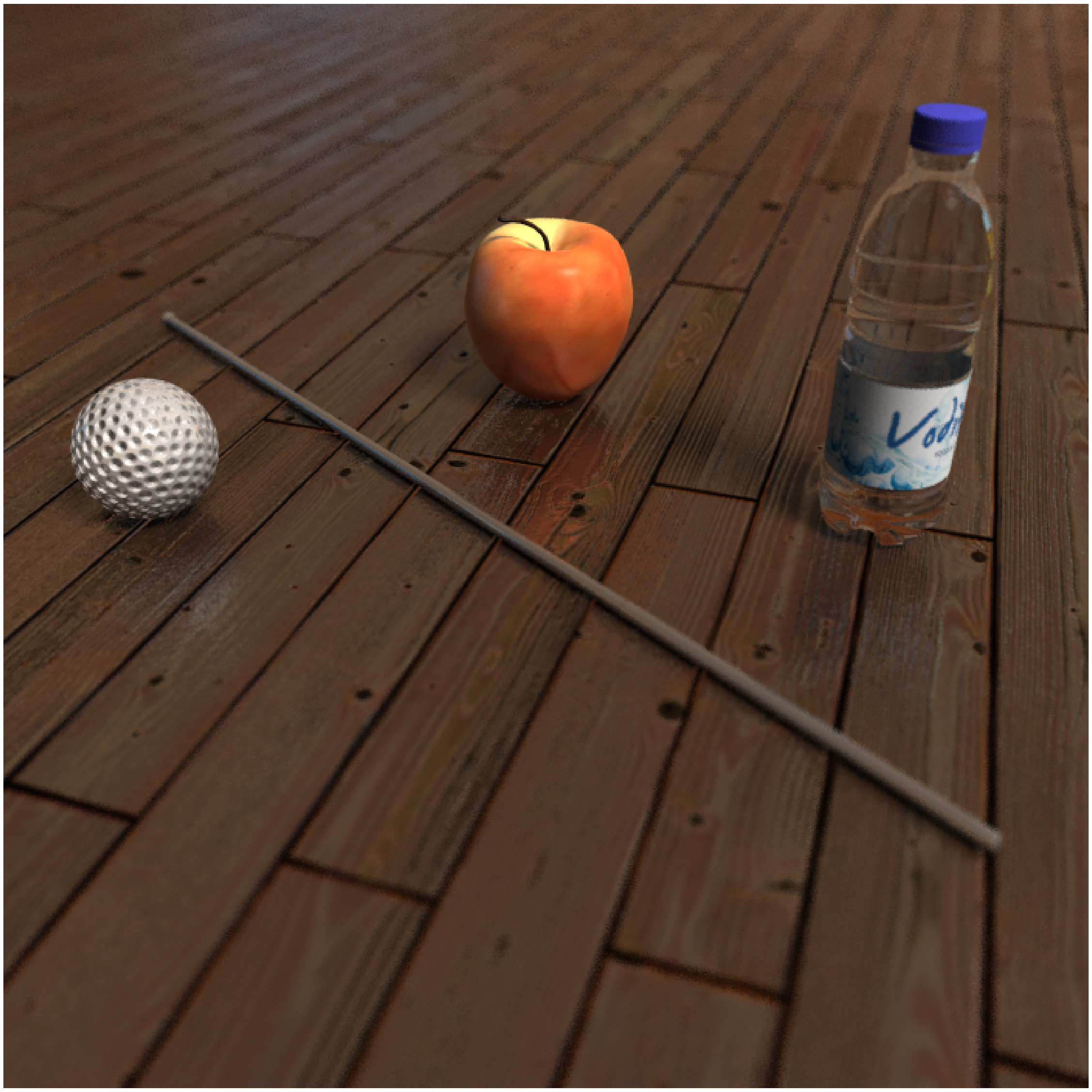

Figure 3: The Normal

4.0 Future Work and Potential Improvements

Although I am very pleased with the results I was able to produce in the allotted time, there are many, many features I would still like to implement, and many aspects of the code base I would like to improve in the future. Code improvements I would like to see in the future are the following:

- Refactor entire Material class. Factor out map (texture, bump, etc) into its own class.

- Factor out anti-aliasing code to enable easily switching between techniques.
- Make top level rendering code more modular. Shrink the horrible `gr.render` function.
- Calculate minimum intersection distances in world space rather than model space.

In terms of features, some ideas that should be easy to add to my existing work are as follows:

- Bump mapping for mesh primitives.
- Photon mapping for caustics.
- Procedural 2D and 3D textures.
- Level-set rendering through ray-marching.

The list literally goes on and on.

If you would like to see the final images produced for this project, you can view them on my personal website at `lawsonfulton.com/YacRay`.

Asset Credits

Here is a list of credits for models and images I used in the creation of my demonstration scenes for this project.

- Water Bottle - `http://www.turbosquid.com/FullPreview/Index.cfm/ID/582482`
- Apple - `http://tf3dm.com/3d-model/apple-51047.html`
- Environment Maps - `http://hdrmaps.com/freebies/`
- Wood Texture - `https://support.solidangle.com/display/mayatut/Part+1+-+Set+Up+The+Scene`
- Monkey - `www.blender.org`

References

[1] CS488/688 W15 A3: Introduction. Internet. University of Waterloo. [Online]. Available: https://www.student.cs.uwaterloo.ca/~cs488/a3.pdf

[2] CS488/688 W15 A4: Introduction. Internet. University of Waterloo. [Online]. Available: https://www.student.cs.uwaterloo.ca/~cs488/a4.pdf

[3] Robert L. Cook, Thomas Porter, and Loren Carpenter. 1984. Distributed ray tracing. SIGGRAPH Comput. Graph. 18, 3 (January 1984), 137-145.

[4] Turner Whitted. 1980. An improved illumination model for shaded display. Commun. ACM 23, 6 (June 1980), 343-349.

[5] James F. Blinn. 1978. Simulation of wrinkled surfaces. In Proceedings of the 5th annual conference on Computer graphics and interactive techniques (SIGGRAPH 78). ACM, New York, NY, USA, 286-292.

[6] Watt, Alan H., and Mark Watt. Advanced Animation and Rendering Techniques: Theory and Practice. New York, N.Y.: ACM ;, 1992. 199-201. Print.

[7] Erik Reinhard, Michael Stark, Peter Shirley, and James Ferwerda. 2002. Photographic tone reproduction for digital images. In Proceedings of the 29th annual conference on Computer graphics and interactive techniques (SIGGRAPH '02). ACM, New York, NY, USA, 267-276.

[8] KD Trees for Faster Ray Tracing . Internet. FrogSlayer. [Online]. Available: http://blog.frogslayer.com/kd-trees-for-faster-ray-tracing-with-triangles/

[9] References for depth of field implementation in a raytracer. Internet. Stack Overflow. [Online]. Available: http://stackoverflow.com/a/13686064

Checksum

```
sum is: /usr/bin/sum

2015-04-01 14:02       Checksum for A5 for ljfulton on gl02       Page 1


A5:

total 5264

55163664 drwxrwx--- 4 ljfulton cs488       4096 Apr  1 14:01 ./

40342050 -rw-r--r-- 1 ljfulton ljfulton    1143 Apr  1 14:01 README

89719120 drwxrwxr-x 5 ljfulton ljfulton    4096 Apr  1 13:56 data/

85595897 drwxrwx--- 8 ljfulton cs488       4096 Apr  1 13:54 ../

50463126 drwxrwx--- 2 ljfulton ljfulton    8192 Apr  1 13:54 src/

45338268 -rwxr-xr-x 1 ljfulton ljfulton 4610068 Apr  1 13:54 rt*

23573402 -rw-r--r-- 1 ljfulton ljfulton  728416 Apr  1 10:01 screenshot01.png


A5/data:

total 20

55163664 drwxrwx--- 4 ljfulton cs488    4096 Apr  1 14:01 ../

89719121 drwxrwx--- 4 ljfulton ljfulton 4096 Apr  1 13:57 scenes/

89719120 drwxrwxr-x 5 ljfulton ljfulton 4096 Apr  1 13:56 ./

30314200 drwxrwxr-x 2 ljfulton ljfulton 4096 Apr  1 13:56 project_information/

55163668 drwxrwxr-x 2 ljfulton ljfulton 4096 Apr  1 13:56 images/


A5/data/scenes:

total 248

89719121 drwxrwx--- 4 ljfulton ljfulton  4096 Apr  1 13:57 ./

89719120 drwxrwxr-x 5 ljfulton ljfulton  4096 Apr  1 13:56 ../

58548772 -rw-r--r-- 1 ljfulton ljfulton  4714 Apr  1 13:55 tone.lua

82424852 drwxrwxr-x 4 ljfulton ljfulton  4096 Apr  1 13:55 textures/

82424851 -rw-r--r-- 1 ljfulton ljfulton  4084 Apr  1 13:55 texture.lua

82424850 -rw-r--r-- 1 ljfulton ljfulton   828 Apr  1 13:55 suzy.lua

82424849 -rw-r--r-- 1 ljfulton ljfulton  4105 Apr  1 13:55 sponge.lua

82424848 -rw-r-x--- 1 ljfulton ljfulton  8859 Apr  1 13:55 smstdodeca.lua*
```

```
82424847 -rw-r-x--- 1 ljfulton ljfulton   997 Apr  1 13:55 simple.lua*
82424846 -rw-r-x--- 1 ljfulton ljfulton  2907 Apr  1 13:55 simple-cows.lua*
82424845 -rw-r--r-- 1 ljfulton ljfulton  3928 Apr  1 13:55 shadows.lua
82424844 -rw-r----- 1 ljfulton ljfulton 11770 Apr  1 13:55 sample.lua
82424843 -rw-r--r-- 1 ljfulton ljfulton  4389 Apr  1 13:55 refract.lua
82424842 -rw-r--r-- 1 ljfulton ljfulton  3605 Apr  1 13:55 reflect.lua
82424841 -rw-r-x--- 1 ljfulton ljfulton  1268 Apr  1 13:55 readobj.lua*
82424840 -rw-r--r-- 1 ljfulton ljfulton  4105 Apr  1 13:55 phong.lua
82424839 -rw-r--r-- 1 ljfulton ljfulton  3599 Apr  1 13:55 objective1.lua
82424837 -rw-r-x--- 1 ljfulton ljfulton  1470 Apr  1 13:55 nonhier2.lua*
35263136 -rw-r-x--- 1 ljfulton ljfulton  1281 Apr  1 13:55 nonhier.lua*
35263133 -rw-r--r-- 1 ljfulton ljfulton  3926 Apr  1 13:55 moon.lua
35263126 -rw-r-x--- 1 ljfulton ljfulton 38328 Apr  1 13:55 mickey.lua*
89719139 drwxrwxr-x 6 ljfulton ljfulton  8192 Apr  1 13:55 meshes/
89719138 -rw-r--r-- 1 ljfulton ljfulton  3965 Apr  1 13:55 mesh.lua
89719137 -rw-r-x--- 1 ljfulton ljfulton  2905 Apr  1 13:55 macho-cows.lua*
89719136 -rw-r--r-- 1 ljfulton ljfulton  1769 Apr  1 13:55 kd.lua
89719135 -rw-r-x--- 1 ljfulton ljfulton  1566 Apr  1 13:55 instance.lua*
89719134 -rw-r-x--- 1 ljfulton ljfulton   755 Apr  1 13:55 icosa.lua*
89719133 -rw-r-x--- 1 ljfulton ljfulton  2852 Apr  1 13:55 hier.lua*
89719132 -rw-r--r-- 1 ljfulton ljfulton  3013 Apr  1 13:55 glossy.lua
89719131 -rw-r--r-- 1 ljfulton ljfulton  3275 Apr  1 13:55 final.lua
89719130 -rw-r--r-- 1 ljfulton ljfulton  3328 Apr  1 13:55 final-good.lua
89719129 -rw-r-x--- 1 ljfulton ljfulton  1673 Apr  1 13:55 dodeca.lua*
89719128 -rw-r-x--- 1 ljfulton ljfulton  1011 Apr  1 13:55 cylinder.lua*
89719127 -rw-r--r-- 1 ljfulton ljfulton  4308 Apr  1 13:55 cup.lua
89719126 -rw-r--r-- 1 ljfulton ljfulton  4017 Apr  1 13:55 bump2.lua
89719125 -rw-r--r-- 1 ljfulton ljfulton  3314 Apr  1 13:55 bump.lua
```



```
89719124 -rw-r-x--- 1 ljfulton ljfulton  4322 Apr  1 13:55 buckyball.lua*
89719123 -rw-r--r-- 1 ljfulton ljfulton  3564 Apr  1 13:55 alias.lua
```

```
89719122 -rw-r--r-- 1 ljfulton ljfulton  8196 Apr  1 13:55 .DS_Store


A5/data/scenes/textures:
total 215516
89719121 drwxrwx--- 4 ljfulton ljfulton     4096 Apr  1 13:57 ../
79963466 -rw-r--r-- 1 ljfulton ljfulton  8718386 Apr  1 13:55 wood_floor.png
82424852 drwxrwxr-x 4 ljfulton ljfulton     4096 Apr  1 13:55 ./
79963465 -rw-r--r-- 1 ljfulton ljfulton   270825 Apr  1 13:55 weirdbump.png
79963464 -rw-r----- 1 ljfulton ljfulton   131072 Apr  1 13:55 weirdbump.jpg
41900901 -rw-r----- 1 ljfulton ljfulton   490265 Apr  1 13:55 uv_test.png
91359386 -rw-r--r-- 1 ljfulton ljfulton 27155708 Apr  1 13:55 sky4light.png
91359385 -rw-r--r-- 1 ljfulton ljfulton 21320675 Apr  1 13:55 sky4.png
91359384 -rw-r--r-- 1 ljfulton ljfulton 20757652 Apr  1 13:55 sky3.png
91359383 -rw-r--r-- 1 ljfulton ljfulton 45760585 Apr  1 13:55 sky2.png
91359382 -rw-r----- 1 ljfulton ljfulton 22631113 Apr  1 13:55 sky2.jpg
91359381 -rw-r--r-- 1 ljfulton ljfulton   726687 Apr  1 13:55 sky1.png
91359380 -rw-r----- 1 ljfulton ljfulton   130467 Apr  1 13:55 sky1.jpg
91359379 -rw-r----- 1 ljfulton ljfulton   930661 Apr  1 13:55 planksbump.png
91359367 drwxrwxr-x 2 ljfulton ljfulton     4096 Apr  1 13:55 planks/
91359366 -rw-r--r-- 1 ljfulton ljfulton   171733 Apr  1 13:55 newbump.png
17360068 -rw-r--r-- 1 ljfulton ljfulton   638510 Apr  1 13:55 moonbumpinv.png
17360067 -rw-r--r-- 1 ljfulton ljfulton   570650 Apr  1 13:55 moonbump2.png
17360066 -rw-r----- 1 ljfulton ljfulton   245713 Apr  1 13:55 moonbump2.jpg
17360065 -rw-r--r-- 1 ljfulton ljfulton   400086 Apr  1 13:55 moonbump.png
17360064 -rw-r----- 1 ljfulton ljfulton    47080 Apr  1 13:55 moonbump.jpg
17360063 -rw-r--r-- 1 ljfulton ljfulton  2208219 Apr  1 13:55 masonrytexture.png
17360062 -rw-r--r-- 1 ljfulton ljfulton  1377144 Apr  1 13:55 masonrybump.png
17360061 -rw-r----- 1 ljfulton ljfulton   487112 Apr  1 13:55 masonry-wall-texture.jpg
17360060 -rw-r----- 1 ljfulton ljfulton   204937 Apr  1 13:55 masonry-wall-bump-map.jpg
17360059 -rw-r--r-- 1 ljfulton ljfulton   398642 Apr  1 13:55 leafbump.png
17360051 drwxrwxr-x 2 ljfulton ljfulton     4096 Apr  1 13:55 grass/
89948585 -rw-r--r-- 1 ljfulton ljfulton 14279634 Apr  1 13:55 golfcourse.png
89948584 -rw-r--r-- 1 ljfulton ljfulton   113308 Apr  1 13:55 golfball_bump.png
```

```
89948583 -rw-r----- 1 ljfulton ljfulton   223898 Apr  1 13:55 golfball_bump.jpg
89948582 -rw-r----- 1 ljfulton ljfulton  2137763 Apr  1 13:55 earth.png
89948581 -rw-r----- 1 ljfulton ljfulton     9264 Apr  1 13:55 checker_low_res.png
89948580 -rw-r--r-- 1 ljfulton ljfulton    21427 Apr  1 13:55 checker.png
89948579 -rw-r--r-- 1 ljfulton ljfulton   565152 Apr  1 13:55 bumpx.png
89948578 -rw-r--r-- 1 ljfulton ljfulton   133604 Apr  1 13:55 bumptest.png
89948577 -rw-r--r-- 1 ljfulton ljfulton   714925 Apr  1 13:55 bumpdots.png
89948575 -rw-r--r-- 1 ljfulton ljfulton  1271801 Apr  1 13:55 brickbump2.png
89948574 -rw-r----- 1 ljfulton ljfulton   700481 Apr  1 13:55 brickbump2.jpg
89948573 -rw-r--r-- 1 ljfulton ljfulton   335767 Apr  1 13:55 brickbump.png
89948572 -rw-r----- 1 ljfulton ljfulton    97286 Apr  1 13:55 brickbump.jpg
89948571 -rw-r--r-- 1 ljfulton ljfulton    65912 Apr  1 13:55 big_checker.png
89948570 -rw-r--r-- 1 ljfulton ljfulton 12062940 Apr  1 13:55 apartment_env_map_sm.png
89948569 -rw-r--r-- 1 ljfulton ljfulton 31088236 Apr  1 13:55 apartment_env_map.png
89948568 -rw-r--r-- 1 ljfulton ljfulton     6148 Apr  1 13:55 .DS_Store

A5/data/scenes/textures/planks:
total 141700
82424852 drwxrwxr-x 4 ljfulton ljfulton     4096 Apr  1 13:55 ../
91359378 -rw-r--r-- 1 ljfulton ljfulton  7227769 Apr  1 13:55 wood_specular_sm.png
91359367 drwxrwxr-x 2 ljfulton ljfulton     4096 Apr  1 13:55 ./
```



```
91359377 -rw-r--r-- 1 ljfulton ljfulton 28897446 Apr  1 13:55 wood_specular.png
91359376 -rw-r--r-- 1 ljfulton ljfulton  9701457 Apr  1 13:55 wood_diffuse_sm_light.png
91359375 -rw-r--r-- 1 ljfulton ljfulton  8035781 Apr  1 13:55 wood_diffuse_sm.png
91359374 -rw-r--r-- 1 ljfulton ljfulton 44400766 Apr  1 13:55 wood_diffuse.png
91359373 -rw-r--r-- 1 ljfulton ljfulton  4972604 Apr  1 13:55 wood_bump_sm.png
91359372 -rw-r--r-- 1 ljfulton ljfulton 12585268 Apr  1 13:55 wood_bump.png
91359371 -rw-r----- 1 ljfulton ljfulton  9542551 Apr  1 13:55 wood-flooring-041_r.jpg
91359370 -rw-r----- 1 ljfulton ljfulton  8952284 Apr  1 13:55 wood-flooring-041_d.jpg
91359369 -rw-r----- 1 ljfulton ljfulton 10125291 Apr  1 13:55 wood-flooring-041_b.png
```

```
91359368 -rw-r--r-- 1 ljfulton ljfulton     6148 Apr  1 13:55 .DS_Store


A5/data/scenes/textures/grass:
total 11476
82424852 drwxrwxr-x 4 ljfulton ljfulton    4096 Apr  1 13:55 ../
17360058 -rw-r--r-- 1 ljfulton ljfulton 3758327 Apr  1 13:55 texture.png
17360051 drwxrwxr-x 2 ljfulton ljfulton    4096 Apr  1 13:55 ./
17360057 -rw-r----- 1 ljfulton ljfulton  966083 Apr  1 13:55 texture.jpg
17360054 -rw-r--r-- 1 ljfulton ljfulton 4121109 Apr  1 13:55 texture-light.png
17360053 -rw-r--r-- 1 ljfulton ljfulton 1862353 Apr  1 13:55 bump.png
17360052 -rw-r----- 1 ljfulton ljfulton  977949 Apr  1 13:55 bump.jpg


A5/data/scenes/meshes:
total 78248
89719121 drwxrwx--- 4 ljfulton ljfulton     4096 Apr  1 13:57 ../
91998169 -rw-r--r-- 1 ljfulton ljfulton    23288 Apr  1 13:55 uv_sphere.obj
89719139 drwxrwxr-x 6 ljfulton ljfulton     8192 Apr  1 13:55 ./
91998168 -rw-r--r-- 1 ljfulton ljfulton    75348 Apr  1 13:55 uv_mapped_sphere.obj
91998167 -rw-r--r-- 1 ljfulton ljfulton      263 Apr  1 13:55 uv_mapped_sphere.mtl
91998166 -rw-r--r-- 1 ljfulton ljfulton    83923 Apr  1 13:55 towers.obj
91998165 -rw-r--r-- 1 ljfulton ljfulton      137 Apr  1 13:55 towers.mtl
55359902 drwxrwx--- 2 ljfulton ljfulton     4096 Apr  1 13:55 teapot/
55359901 -rw-r--r-- 1 ljfulton ljfulton  2268618 Apr  1 13:55 teacup.obj
55359900 -rw-r--r-- 1 ljfulton ljfulton      137 Apr  1 13:55 teacup.mtl
55359899 -rw-r--r-- 1 ljfulton ljfulton   248538 Apr  1 13:55 suzy_smooth_tris.obj
55359898 -rw-r--r-- 1 ljfulton ljfulton      137 Apr  1 13:55 suzy_smooth_tris.mtl
55359897 -rw-r--r-- 1 ljfulton ljfulton   121193 Apr  1 13:55 suzy_smooth_no_normals.obj
55359896 -rw-r--r-- 1 ljfulton ljfulton      137 Apr  1 13:55 suzy_smooth_no_normals.mtl
55359895 -rw-r--r-- 1 ljfulton ljfulton   205577 Apr  1 13:55 suzy_smooth.obj
55359889 -rw-r--r-- 1 ljfulton ljfulton      137 Apr  1 13:55 suzy_smooth.mtl
55359888 -rw-r--r-- 1 ljfulton ljfulton    23838 Apr  1 13:55 suzy.obj
55359887 -rw-r--r-- 1 ljfulton ljfulton      137 Apr  1 13:55 suzy.mtl
55359885 -rw-r--r-- 1 ljfulton ljfulton   314614 Apr  1 13:55 stanford_bunny_smooth.obj
```

```
18537886 -rw-r--r-- 1 ljfulton ljfulton      137 Apr  1 13:55 stanford_bunny_smooth.mtl
18537885 -rw-r--r-- 1 ljfulton ljfulton   198442 Apr  1 13:55 stanford_bunny.obj
18537883 -rw-r--r-- 1 ljfulton ljfulton    56488 Apr  1 13:55 smoothed_uv_sphere.obj
18537882 -rw-r--r-- 1 ljfulton ljfulton      137 Apr  1 13:55 smoothed_uv_sphere.mtl
18537881 -rwxr-xr-x 1 ljfulton ljfulton   314564 Apr  1 13:55 run*
18537880 -rw-r--r-- 1 ljfulton ljfulton   421726 Apr  1 13:55 pointer.obj
18537879 -rw-r--r-- 1 ljfulton ljfulton      137 Apr  1 13:55 pointer.mtl
18537878 -rw-r--r-- 1 ljfulton ljfulton   884972 Apr  1 13:55 pointer.blend1
54759214 -rw-r--r-- 1 ljfulton ljfulton   884972 Apr  1 13:55 pointer.blend
54759213 -rw-r--r-- 1 ljfulton ljfulton   247514 Apr  1 13:55 monkey_sit_smooth.obj
54759201 -rw-r--r-- 1 ljfulton ljfulton      137 Apr  1 13:55 monkey_sit_smooth.mtl
25533031 -rw-r--r-- 1 ljfulton ljfulton   310350 Apr  1 13:55 monkey_sit.obj
88140942 -rw-r--r-- 1 ljfulton ljfulton      137 Apr  1 13:55 monkey_sit.mtl
88140941 -rw-r--r-- 1 ljfulton ljfulton   248932 Apr  1 13:55 med_poly_sphere.obj
```



```
88140938 -rw-r--r-- 1 ljfulton ljfulton      137 Apr  1 13:55 med_poly_sphere.mtl
88140937 -rw-r--r-- 1 ljfulton ljfulton   739384 Apr  1 13:55 kd.png
88140934 -rw-r--r-- 1 ljfulton ljfulton  1051559 Apr  1 13:55 high_poly_sphere.obj
88140933 -rw-r--r-- 1 ljfulton ljfulton      137 Apr  1 13:55 high_poly_sphere.mtl
10164877 drwxrwx--- 2 ljfulton ljfulton     4096 Apr  1 13:55 head/
10164876 -rw-r--r-- 1 ljfulton ljfulton     8963 Apr  1 13:55 grate.obj
10164875 -rw-r--r-- 1 ljfulton ljfulton      137 Apr  1 13:55 grate.mtl
10164874 -rw-r--r-- 1 ljfulton ljfulton   512781 Apr  1 13:55 golf_tee.obj
10164873 -rw-r--r-- 1 ljfulton ljfulton      144 Apr  1 13:55 golf_tee.mtl
10164872 -rw-r--r-- 1 ljfulton ljfulton 69034044 Apr  1 13:55 dragon_smooth.obj
10164871 -rw-r--r-- 1 ljfulton ljfulton     6230 Apr  1 13:55 cyl_smooth.obj
10164870 -rw-r--r-- 1 ljfulton ljfulton      137 Apr  1 13:55 cyl_smooth.mtl
10164869 -rw-r--r-- 1 ljfulton ljfulton      749 Apr  1 13:55 cube_with_normals.obj
10164868 -rw-r--r-- 1 ljfulton ljfulton      137 Apr  1 13:55 cube_with_normals.mtl
10164867 -rw-r--r-- 1 ljfulton ljfulton      746 Apr  1 13:55 cube_smooth.obj
10164866 -rw-r--r-- 1 ljfulton ljfulton      201 Apr  1 13:55 cube_smooth.mtl
```

```
10164865 -rw-r--r-- 1 ljfulton ljfulton      456 Apr  1 13:55 cube_no_normals.obj
10164864 -rw-r--r-- 1 ljfulton ljfulton      137 Apr  1 13:55 cube_no_normals.mtl
10164861 -rw-r-x--- 1 ljfulton ljfulton   180010 Apr  1 13:55 cow.obj*
92174249 drwxrwxr-x 2 ljfulton ljfulton     4096 Apr  1 13:55 bottle/
92174248 -rw-r--r-- 1 ljfulton ljfulton  1091220 Apr  1 13:55 apple.blend
92174230 drwxrwx--- 3 ljfulton ljfulton     4096 Apr  1 13:55 Apple/
92174229 -rw-r--r-- 1 ljfulton ljfulton     6148 Apr  1 13:55 .DS_Store

A5/data/scenes/meshes/teapot:
total 2272
89719139 drwxrwxr-x 6 ljfulton ljfulton    8192 Apr  1 13:55 ../
42145813 -rwxr-xr-x 1 ljfulton ljfulton  868772 Apr  1 13:55 teapot.obj*
55359902 drwxrwx--- 2 ljfulton ljfulton    4096 Apr  1 13:55 ./
42145812 -rwxr-xr-x 1 ljfulton ljfulton 1417216 Apr  1 13:55 teapot.max*
55359905 -rwxr-xr-x 1 ljfulton ljfulton    3071 Apr  1 13:55 default.png*
55359904 -rwxr-xr-x 1 ljfulton ljfulton     281 Apr  1 13:55 default.mtl*
55359903 -rwxr-xr-x 1 ljfulton ljfulton     652 Apr  1 13:55 copyright.txt*

A5/data/scenes/meshes/head:
total 119372
89719139 drwxrwxr-x 6 ljfulton ljfulton     8192 Apr  1 13:55 ../
 9411416 -rw-r--r-- 1 ljfulton ljfulton 24799648 Apr  1 13:55 skin.png
10164877 drwxrwx--- 2 ljfulton ljfulton     4096 Apr  1 13:55 ./
 9411415 -rw-r--r-- 1 ljfulton ljfulton 20835781 Apr  1 13:55 skin copy.png
 9411414 -rwxr-xr-x 1 ljfulton ljfulton    91931 Apr  1 13:55 rendered.jpg*
 9411413 -rwxr-xr-x 1 ljfulton ljfulton  8975275 Apr  1 13:55 lambertian.jpg*
 9411412 -rwxr-xr-x 1 ljfulton ljfulton     1206 Apr  1 13:55 Infinite-Scan_License.txt*
 9411411 -rw-r--r-- 1 ljfulton ljfulton  1592664 Apr  1 13:55 head_smooth.obj
 9411410 -rw-r--r-- 1 ljfulton ljfulton      256 Apr  1 13:55 head_smooth.mtl
 9411409 -rwxr-xr-x 1 ljfulton ljfulton  1436246 Apr  1 13:55 head.OBJ*
 9411408 -rwxr-xr-x 1 ljfulton ljfulton      194 Apr  1 13:55 head.mtl*
 9411407 -rwxr-xr-x 1 ljfulton ljfulton 29263143 Apr  1 13:55 bump.png*
 9411406 -rwxr-xr-x 1 ljfulton ljfulton   439109 Apr  1 13:55 bump-lowRes.png*
```

```
 9411405 -rwxr-xr-x 1 ljfulton ljfulton   377110 Apr  1 13:55 bump-lowRes copy.png*
 9411404 -rwxr-xr-x 1 ljfulton ljfulton 33859474 Apr  1 13:55 bump copy.png*

A5/data/scenes/meshes/bottle:
total 24888
89719139 drwxrwxr-x 6 ljfulton ljfulton    8192 Apr  1 13:55 ../
90502662 -rw-r--r-- 1 ljfulton ljfulton 6998960 Apr  1 13:55 sphere.obj

2015-04-01 14:02       Checksum for A5 for ljfulton on gl02       Page 5

92174249 drwxrwxr-x 2 ljfulton ljfulton    4096 Apr  1 13:55 ./
90502661 -rw-r--r-- 1 ljfulton ljfulton     737 Apr  1 13:55 sphere.mtl
90502660 -rw-r--r-- 1 ljfulton ljfulton 1329697 Apr  1 13:55 label.png
90502659 -rw-r--r-- 1 ljfulton ljfulton  733413 Apr  1 13:55 etiket.jpg
90502658 -rw-r--r-- 1 ljfulton ljfulton 1562392 Apr  1 13:55 bottle_water.obj
90502657 -rw-r--r-- 1 ljfulton ljfulton     195 Apr  1 13:55 bottle_water.mtl
90502656 -rw-r--r-- 1 ljfulton ljfulton  142824 Apr  1 13:55 bottle_label.obj
90502655 -rw-r--r-- 1 ljfulton ljfulton     195 Apr  1 13:55 bottle_label.mtl
90502654 -rw-r--r-- 1 ljfulton ljfulton  535265 Apr  1 13:55 bottle_cap.obj
90502653 -rw-r--r-- 1 ljfulton ljfulton     197 Apr  1 13:55 bottle_cap.mtl
90502652 -rw-r--r-- 1 ljfulton ljfulton 4033134 Apr  1 13:55 bottle_body.obj
90502651 -rw-r--r-- 1 ljfulton ljfulton     201 Apr  1 13:55 bottle_body.mtl
90502650 -rw-r----- 1 ljfulton ljfulton 9978946 Apr  1 13:55 Bottle.obj

A5/data/scenes/meshes/Apple:
total 1520
89719139 drwxrwxr-x 6 ljfulton ljfulton   8192 Apr  1 13:55 ../
92174247 -rw-r--r-- 1 ljfulton ljfulton 389794 Apr  1 13:55 apple_smooth.obj~
92174230 drwxrwx--- 3 ljfulton ljfulton   4096 Apr  1 13:55 ./
92174246 -rw-r--r-- 1 ljfulton ljfulton  37129 Apr  1 13:55 stem_smooth.obj
92174245 -rw-r--r-- 1 ljfulton ljfulton    228 Apr  1 13:55 stem_smooth.mtl
92174244 -rw-r--r-- 1 ljfulton ljfulton  20347 Apr  1 13:55 stem.png
92174243 -rw-r--r-- 1 ljfulton ljfulton 113458 Apr  1 13:55 skin.png
```

```
92174236 drwxrwx--- 2 ljfulton ljfulton   4096 Apr  1 13:55 Maps/
92174235 -rw-r--r-- 1 ljfulton ljfulton 389795 Apr  1 13:55 apple_smooth.obj
92174234 -rw-r--r-- 1 ljfulton ljfulton    403 Apr  1 13:55 apple_smooth.mtl
92174233 -rw-r--r-- 1 ljfulton ljfulton 429385 Apr  1 13:55 apple.obj
92174232 -rw-r--r-- 1 ljfulton ljfulton    582 Apr  1 13:55 apple.mtl
92174231 -rw-r--r-- 1 ljfulton ljfulton 109333 Apr  1 13:55 apple.3ds

A5/data/scenes/meshes/Apple/Maps:
total 264
92174230 drwxrwx--- 3 ljfulton ljfulton   4096 Apr  1 13:55 ../
92174242 -rw-r--r-- 1 ljfulton ljfulton  37584 Apr  1 13:55 stem_color.tif
92174236 drwxrwx--- 2 ljfulton ljfulton   4096 Apr  1 13:55 ./
92174241 -rw-r--r-- 1 ljfulton ljfulton   2468 Apr  1 13:55 stem_color.jpg
92174240 -rw-r--r-- 1 ljfulton ljfulton  37584 Apr  1 13:55 stem_bump.tif
92174239 -rw-r--r-- 1 ljfulton ljfulton 145221 Apr  1 13:55 skin.tif
92174238 -rw-r--r-- 1 ljfulton ljfulton   6732 Apr  1 13:55 skin.jpg
92174237 -rw-r--r-- 1 ljfulton ljfulton   6732 Apr  1 13:55 skin copy.jpg

A5/data/project_information:
total 396
30314200 drwxrwxr-x 2 ljfulton ljfulton   4096 Apr  1 13:56 ./
89719120 drwxrwxr-x 5 ljfulton ljfulton   4096 Apr  1 13:56 ../
94882730 -rw-r--r-- 1 ljfulton ljfulton   9710 Apr  1 10:00 CS488-Project-Proposal.tex
94882729 -rw-r--r-- 1 ljfulton ljfulton  25168 Apr  1 10:00 CS488-Project-Proposal.synctex.gz
94882728 -rw-r--r-- 1 ljfulton ljfulton 150112 Apr  1 10:00 CS488-Project-Proposal.pdf
94882727 -rw-r--r-- 1 ljfulton ljfulton   5375 Apr  1 10:00 CS488-Project-Proposal.log
94882726 -rw-r--r-- 1 ljfulton ljfulton      8 Apr  1 10:00 CS488-Project-Proposal.aux
94882725 -rw-r----- 1 ljfulton ljfulton 174935 Apr  1 10:00 a5.pdf
30314201 -rw-r--r-- 1 ljfulton ljfulton   6148 Apr  1 10:00 .DS_Store

A5/data/images:
total 12948
89719120 drwxrwxr-x 5 ljfulton ljfulton    4096 Apr  1 13:56 ../
```

```
55163668 drwxrwxr-x 2 ljfulton ljfulton    4096 Apr  1 13:56 ./
97122389 -rw-r--r-- 1 ljfulton ljfulton  254731 Apr  1 10:00 objective9-before.png
97122388 -rw-r--r-- 1 ljfulton ljfulton  342422 Apr  1 10:00 objective9-after.png
97122387 -rw-r--r-- 1 ljfulton ljfulton  852878 Apr  1 10:00 objective8-before.png
97122386 -rw-r--r-- 1 ljfulton ljfulton  901983 Apr  1 10:00 objective8-after.png
97122385 -rw-r--r-- 1 ljfulton ljfulton  262479 Apr  1 10:00 objective7-before.png
97122384 -rw-r--r-- 1 ljfulton ljfulton  176109 Apr  1 10:00 objective7-after.png
97122383 -rw-r--r-- 1 ljfulton ljfulton  795135 Apr  1 10:00 objective6-before.png
97122382 -rw-r--r-- 1 ljfulton ljfulton  818617 Apr  1 10:00 objective6-after.png
97122381 -rw-r--r-- 1 ljfulton ljfulton   44007 Apr  1 10:00 objective5-before2.png
97122380 -rw-r--r-- 1 ljfulton ljfulton   44299 Apr  1 10:00 objective5-before1.png
97122379 -rw-r--r-- 1 ljfulton ljfulton   54319 Apr  1 10:00 objective5-after.png
97122378 -rw-r--r-- 1 ljfulton ljfulton  792782 Apr  1 10:00 objective4-before.png
97122377 -rw-r--r-- 1 ljfulton ljfulton  717842 Apr  1 10:00 objective4-after.png
97122376 -rw-r--r-- 1 ljfulton ljfulton   77942 Apr  1 10:00 objective3-before.png
97122375 -rw-r--r-- 1 ljfulton ljfulton  210611 Apr  1 10:00 objective3-after.png
55163681 -rw-r--r-- 1 ljfulton ljfulton  553779 Apr  1 10:00 objective2-before.png
55163680 -rw-r--r-- 1 ljfulton ljfulton  591246 Apr  1 10:00 objective2-after1.png
55163679 -rw-r--r-- 1 ljfulton ljfulton  728416 Apr  1 10:00 objective10.png
55163678 -rw-r--r-- 1 ljfulton ljfulton  862206 Apr  1 10:00 objective10-nodof.png
55163677 -rw-r--r-- 1 ljfulton ljfulton  405074 Apr  1 10:00 objective10-2.png
55163676 -rw-r--r-- 1 ljfulton ljfulton  318221 Apr  1 10:00 objective1-before.png
55163675 -rw-r--r-- 1 ljfulton ljfulton  319832 Apr  1 10:00 objective1-after.png
55163674 -rw-r--r-- 1 ljfulton ljfulton  453335 Apr  1 10:00 bonus4.png
55163673 -rw-r--r-- 1 ljfulton ljfulton  751328 Apr  1 10:00 bonus3.png
55163672 -rw-r--r-- 1 ljfulton ljfulton  739384 Apr  1 10:00 bonus2.png
55163670 -rw-r--r-- 1 ljfulton ljfulton 1010364 Apr  1 10:00 bonus1.png
55163669 -rw-r--r-- 1 ljfulton ljfulton    6148 Apr  1 10:00 .DS_Store
```

A5/src:

```
total 316
55163664 drwxrwx--- 4 ljfulton cs488     4096 Apr  1 14:01 ../
50463126 drwxrwx--- 2 ljfulton ljfulton  8192 Apr  1 13:54 ./
 3716022 -rw-r-x--- 1 ljfulton ljfulton  1081 Apr  1 13:35 Makefile*
92663563 -rw-r-x--- 1 ljfulton ljfulton  2535 Apr  1 13:33 material.hpp*
99082100 -rw-r--r-- 1 ljfulton ljfulton  1976 Apr  1 10:01 TODO.txt
99082099 -rwxr-xr-x 1 ljfulton ljfulton  2829 Apr  1 10:01 tiny_obj_loader.h*
99082098 -rwxr-xr-x 1 ljfulton ljfulton 21860 Apr  1 10:01 tiny_obj_loader.cpp*
99082097 -rw-r-x--- 1 ljfulton ljfulton   136 Apr  1 10:01 scene_lua.hpp*
99082096 -rw-r-x--- 1 ljfulton ljfulton 21928 Apr  1 10:01 scene_lua.cpp*
99082095 -rw-r-x--- 1 ljfulton ljfulton  2547 Apr  1 10:01 scene.hpp*
24557059 -rw-r-x--- 1 ljfulton ljfulton  4477 Apr  1 10:01 scene.cpp*
24557058 -rwxr-x--- 1 ljfulton ljfulton    57 Apr  1 10:01 run*
24557057 -rw-r----- 1 ljfulton ljfulton  1730 Apr  1 10:01 renderer.hpp
24557048 -rw-r----- 1 ljfulton ljfulton 11538 Apr  1 10:01 renderer.cpp
24557047 -rw-r----- 1 ljfulton ljfulton  1491 Apr  1 10:01 ray.hpp
24557046 -rw-r----- 1 ljfulton ljfulton   227 Apr  1 10:01 ray.cpp
24557045 -rw-r-x--- 1 ljfulton ljfulton  2486 Apr  1 10:01 primitive.hpp*
88647733 -rw-r-x--- 1 ljfulton ljfulton 12908 Apr  1 10:01 primitive.cpp*
88647728 -rw-r-x--- 1 ljfulton ljfulton   966 Apr  1 10:01 polyroots.hpp*
88647726 -rw-r-x--- 1 ljfulton ljfulton 47780 Apr  1 10:01 polyroots.cpp*
88647723 -rw-r----- 1 ljfulton ljfulton   139 Apr  1 10:01 options.hpp
61207985 -rw-r-x--- 1 ljfulton ljfulton  2318 Apr  1 10:01 mesh.hpp*
61207984 -rw-r-x--- 1 ljfulton ljfulton 19478 Apr  1 10:01 mesh.cpp*
61207982 -rw-r-x--- 1 ljfulton ljfulton  9680 Apr  1 10:01 material.cpp*
61207980 -rw-r-x--- 1 ljfulton ljfulton   276 Apr  1 10:01 main.cpp*
```



```
61207979 -rw-r-x--- 1 ljfulton ljfulton   170 Apr  1 10:01 lua488.hpp*
61207978 -rw-r-x--- 1 ljfulton ljfulton  1172 Apr  1 10:01 light.hpp*
61207977 -rw-r-x--- 1 ljfulton ljfulton  2625 Apr  1 10:01 light.cpp*
61207976 -rw-r-x--- 1 ljfulton ljfulton  2289 Apr  1 10:01 image.hpp*
```

```
61207975 -rw-r-x--- 1 ljfulton ljfulton 10943 Apr  1 10:01 image.cpp*
61207974 -rw-r----- 1 ljfulton ljfulton   602 Apr  1 10:01 camera.hpp
61207973 -rw-r----- 1 ljfulton ljfulton  2569 Apr  1 10:01 camera.cpp
61207972 -rw-r-x--- 1 ljfulton ljfulton 14055 Apr  1 10:01 algebra.hpp*
61207971 -rw-r-x--- 1 ljfulton ljfulton  4034 Apr  1 10:01 algebra.cpp*
61207970 -rw-r-x--- 1 ljfulton ljfulton   831 Apr  1 10:01 a4.hpp*
50463128 -rw-r-x--- 1 ljfulton ljfulton  1585 Apr  1 10:01 a4.cpp*
50463127 -rw-r--r-- 1 ljfulton ljfulton  6148 Apr  1 10:01 .DS_Store


A5
A5/README                           37923     2
A5/data
A5/data/images
A5/data/images/.DS_Store            33880     7
A5/data/images/bonus1.png           54743   987
A5/data/images/bonus2.png           13473   723
A5/data/images/bonus3.png           28509   734
A5/data/images/bonus4.png           09493   443
A5/data/images/objective1-after.png     18680   313
A5/data/images/objective1-before.png    03323   311
A5/data/images/objective10-2.png        20078   396
A5/data/images/objective10-nodof.png    36511   842
A5/data/images/objective10.png          40438   712
A5/data/images/objective2-after1.png    38571   578
A5/data/images/objective2-before.png    01931   541
A5/data/images/objective3-after.png     63589   206
A5/data/images/objective3-before.png    27424    77
A5/data/images/objective4-after.png     37748   702
A5/data/images/objective4-before.png    50294   775
A5/data/images/objective5-after.png     17299    54
A5/data/images/objective5-before1.png   01259    44
A5/data/images/objective5-before2.png   61765    43
A5/data/images/objective6-after.png     38286   800
```

A5/data/images/objective6-before.png	36126	777
A5/data/images/objective7-after.png	29043	172
A5/data/images/objective7-before.png	00541	257
A5/data/images/objective8-after.png	34689	881
A5/data/images/objective8-before.png	51668	833
A5/data/images/objective9-after.png	40966	335
A5/data/images/objective9-before.png	47444	249
A5/data/project_information		
A5/data/project_information/.DS_Store	33880	7
A5/data/project_information/CS488-Project-Proposal.aux34896		1
A5/data/project_information/CS488-Project-Proposal.log10424		6
A5/data/project_information/CS488-Project-Proposal.pdf28890		147
A5/data/project_information/CS488-Project-Proposal.synctex.gz37047		25
A5/data/project_information/CS488-Project-Proposal.tex47410		10
A5/data/project_information/a5.pdf	35911	171
A5/data/scenes		
A5/data/scenes/.DS_Store	62014	9
A5/data/scenes/alias.lua	01053	4
A5/data/scenes/buckyball.lua	10689	5

2015-04-01 14:02 Checksum for A5 for ljfulton on gl02 Page 8

A5/data/scenes/bump.lua	50836	4
A5/data/scenes/bump2.lua	41619	4
A5/data/scenes/cup.lua	51564	5
A5/data/scenes/cylinder.lua	01173	1
A5/data/scenes/dodeca.lua	50755	2
A5/data/scenes/final-good.lua	40331	4
A5/data/scenes/final.lua	33496	4
A5/data/scenes/glossy.lua	47988	3
A5/data/scenes/hier.lua	44794	3
A5/data/scenes/icosa.lua	58132	1
A5/data/scenes/instance.lua	25944	2

```
A5/data/scenes/kd.lua                  59254     2
A5/data/scenes/macho-cows.lua          23029     3
A5/data/scenes/mesh.lua                14610     4
A5/data/scenes/meshes
A5/data/scenes/meshes/.DS_Store        34323     7
A5/data/scenes/meshes/Apple
A5/data/scenes/meshes/Apple/Maps
A5/data/scenes/meshes/Apple/Maps/skin copy.jpg17756     7
A5/data/scenes/meshes/Apple/Maps/skin.jpg17756     7
A5/data/scenes/meshes/Apple/Maps/skin.tif44873   142
A5/data/scenes/meshes/Apple/Maps/stem_bump.tif10131    37
A5/data/scenes/meshes/Apple/Maps/stem_color.jpg07177     3
A5/data/scenes/meshes/Apple/Maps/stem_color.tif64324    37
A5/data/scenes/meshes/Apple/apple.3ds   41188   107
A5/data/scenes/meshes/Apple/apple.mtl   28805     1
A5/data/scenes/meshes/Apple/apple.obj   19167   420
A5/data/scenes/meshes/Apple/apple_smooth.mtl54046     1
A5/data/scenes/meshes/Apple/apple_smooth.obj49192   381
A5/data/scenes/meshes/Apple/apple_smooth.obj~32829   381
A5/data/scenes/meshes/Apple/skin.png    38177   111
A5/data/scenes/meshes/Apple/stem.png    25248    20
A5/data/scenes/meshes/Apple/stem_smooth.mtl61665     1
A5/data/scenes/meshes/Apple/stem_smooth.obj32709    37
A5/data/scenes/meshes/apple.blend       35169  1066
A5/data/scenes/meshes/bottle
A5/data/scenes/meshes/bottle/Bottle.obj 59526  9746
A5/data/scenes/meshes/bottle/bottle_body.mtl53818     1
A5/data/scenes/meshes/bottle/bottle_body.obj53601  3939
A5/data/scenes/meshes/bottle/bottle_cap.mtl08358     1
A5/data/scenes/meshes/bottle/bottle_cap.obj43423   523
A5/data/scenes/meshes/bottle/bottle_label.mtl14818     1
A5/data/scenes/meshes/bottle/bottle_label.obj45348   140
A5/data/scenes/meshes/bottle/bottle_water.mtl43737     1
```

```
A5/data/scenes/meshes/bottle/bottle_water.obj59263  1526
A5/data/scenes/meshes/bottle/etiket.jpg 63723   717
A5/data/scenes/meshes/bottle/label.png  36486  1299
A5/data/scenes/meshes/bottle/sphere.mtl 09205     1
A5/data/scenes/meshes/bottle/sphere.obj 56731  6835
A5/data/scenes/meshes/cow.obj           57991   176
A5/data/scenes/meshes/cube_no_normals.mtl56550     1
A5/data/scenes/meshes/cube_no_normals.obj25381     1
A5/data/scenes/meshes/cube_smooth.mtl   11215     1
A5/data/scenes/meshes/cube_smooth.obj   52745     1
A5/data/scenes/meshes/cube_with_normals.mtl56550     1
A5/data/scenes/meshes/cube_with_normals.obj22196     1
```



```
A5/data/scenes/meshes/cyl_smooth.mtl    56550     1
A5/data/scenes/meshes/cyl_smooth.obj    49360     7
A5/data/scenes/meshes/dragon_smooth.obj 20208 67417
A5/data/scenes/meshes/golf_tee.mtl      64762     1
A5/data/scenes/meshes/golf_tee.obj      63982   501
A5/data/scenes/meshes/grate.mtl         56550     1
A5/data/scenes/meshes/grate.obj         03710     9
A5/data/scenes/meshes/head
A5/data/scenes/meshes/head/Infinite-Scan_License.txt10744     2
A5/data/scenes/meshes/head/bump copy.png44608 33066
A5/data/scenes/meshes/head/bump-lowRes copy.png53903   369
A5/data/scenes/meshes/head/bump-lowRes.png26237   429
A5/data/scenes/meshes/head/bump.png     34589 28578
A5/data/scenes/meshes/head/head.OBJ     48971  1403
A5/data/scenes/meshes/head/head.mtl     37109     1
A5/data/scenes/meshes/head/head_smooth.mtl55770     1
A5/data/scenes/meshes/head/head_smooth.obj00248  1556
A5/data/scenes/meshes/head/lambertian.jpg37841  8765
```

```
A5/data/scenes/meshes/head/rendered.jpg 08911    90
A5/data/scenes/meshes/head/skin copy.png45162 20348
A5/data/scenes/meshes/head/skin.png     38952 24219
A5/data/scenes/meshes/high_poly_sphere.mtl56550     1
A5/data/scenes/meshes/high_poly_sphere.obj53345  1027
A5/data/scenes/meshes/kd.png            13473   723
A5/data/scenes/meshes/med_poly_sphere.mtl56550     1
A5/data/scenes/meshes/med_poly_sphere.obj31838   244
A5/data/scenes/meshes/monkey_sit.mtl    56550     1
A5/data/scenes/meshes/monkey_sit.obj    05739   304
A5/data/scenes/meshes/monkey_sit_smooth.mtl56550     1
A5/data/scenes/meshes/monkey_sit_smooth.obj00041   242
A5/data/scenes/meshes/pointer.blend     58583   865
A5/data/scenes/meshes/pointer.blend1    08728   865
A5/data/scenes/meshes/pointer.mtl       56550     1
A5/data/scenes/meshes/pointer.obj       52812   412
A5/data/scenes/meshes/run               29983   308
A5/data/scenes/meshes/smoothed_uv_sphere.mtl56550     1
A5/data/scenes/meshes/smoothed_uv_sphere.obj45845    56
A5/data/scenes/meshes/stanford_bunny.obj55740   194
A5/data/scenes/meshes/stanford_bunny_smooth.mtl56550     1
A5/data/scenes/meshes/stanford_bunny_smooth.obj06150   308
A5/data/scenes/meshes/suzy.mtl          56550     1
A5/data/scenes/meshes/suzy.obj          56455    24
A5/data/scenes/meshes/suzy_smooth.mtl   56550     1
A5/data/scenes/meshes/suzy_smooth.obj   29948   201
A5/data/scenes/meshes/suzy_smooth_no_normals.mtl56550     1
A5/data/scenes/meshes/suzy_smooth_no_normals.obj18654   119
A5/data/scenes/meshes/suzy_smooth_tris.mtl56550     1
A5/data/scenes/meshes/suzy_smooth_tris.obj10533   243
A5/data/scenes/meshes/teacup.mtl        56550     1
A5/data/scenes/meshes/teacup.obj        03421  2216
A5/data/scenes/meshes/teapot
```

```
A5/data/scenes/meshes/teapot/copyright.txt28159     1
A5/data/scenes/meshes/teapot/default.mtl29301     1
A5/data/scenes/meshes/teapot/default.png10732     3
A5/data/scenes/meshes/teapot/teapot.max 30863  1384
A5/data/scenes/meshes/teapot/teapot.obj 64289   849
```



```
A5/data/scenes/meshes/towers.mtl        56550     1
A5/data/scenes/meshes/towers.obj        36379    82
A5/data/scenes/meshes/uv_mapped_sphere.mtl47944     1
A5/data/scenes/meshes/uv_mapped_sphere.obj03689    74
A5/data/scenes/meshes/uv_sphere.obj     24408    23
A5/data/scenes/mickey.lua               01039    38
A5/data/scenes/moon.lua                 33392     4
A5/data/scenes/nonhier.lua              07311     2
A5/data/scenes/nonhier2.lua             25636     2
A5/data/scenes/objective1.lua           65126     4
A5/data/scenes/phong.lua                56684     5
A5/data/scenes/readobj.lua              57040     2
A5/data/scenes/reflect.lua              17969     4
A5/data/scenes/refract.lua              27104     5
A5/data/scenes/sample.lua               29577    12
A5/data/scenes/shadows.lua              39268     4
A5/data/scenes/simple-cows.lua          09007     3
A5/data/scenes/simple.lua               02633     1
A5/data/scenes/smstdodeca.lua           02544     9
A5/data/scenes/sponge.lua               43578     5
A5/data/scenes/suzy.lua                 05383     1
A5/data/scenes/texture.lua              48359     4
A5/data/scenes/textures
A5/data/scenes/textures/.DS_Store       59001     7
A5/data/scenes/textures/apartment_env_map.png06654 30360
```

```
A5/data/scenes/textures/apartment_env_map_sm.png60021 11781
A5/data/scenes/textures/big_checker.png 63550    65
A5/data/scenes/textures/brickbump.jpg   26261    96
A5/data/scenes/textures/brickbump.png   30649   328
A5/data/scenes/textures/brickbump2.jpg  32245   685
A5/data/scenes/textures/brickbump2.png  45556  1242
A5/data/scenes/textures/bumpdots.png    39730   699
A5/data/scenes/textures/bumptest.png    08833   131
A5/data/scenes/textures/bumpx.png       30584   552
A5/data/scenes/textures/checker.png     19784    21
A5/data/scenes/textures/checker_low_res.png61723    10
A5/data/scenes/textures/earth.png       37424  2088
A5/data/scenes/textures/golfball_bump.jpg35746   219
A5/data/scenes/textures/golfball_bump.png31371   111
A5/data/scenes/textures/golfcourse.png  12170 13945
A5/data/scenes/textures/grass
A5/data/scenes/textures/grass/bump.jpg  42739   956
A5/data/scenes/textures/grass/bump.png  03611  1819
A5/data/scenes/textures/grass/texture-light.png64220  4025
A5/data/scenes/textures/grass/texture.jpg08022   944
A5/data/scenes/textures/grass/texture.png01010  3671
A5/data/scenes/textures/leafbump.png    36304   390
A5/data/scenes/textures/masonry-wall-bump-map.jpg42388   201
A5/data/scenes/textures/masonry-wall-texture.jpg07285   476
A5/data/scenes/textures/masonrybump.png 63101  1345
A5/data/scenes/textures/masonrytexture.png29861  2157
A5/data/scenes/textures/moonbump.jpg    52514    46
A5/data/scenes/textures/moonbump.png    59171   391
A5/data/scenes/textures/moonbump2.jpg   43180   240
A5/data/scenes/textures/moonbump2.png   63098   558
A5/data/scenes/textures/moonbumpinv.png 16855   624
```

A5/data/scenes/textures/newbump.png	53390	168
A5/data/scenes/textures/planks		
A5/data/scenes/textures/planks/.DS_Store	33880	7
A5/data/scenes/textures/planks/wood-flooring-041_b.png	10765	9888
A5/data/scenes/textures/planks/wood-flooring-041_d.jpg	33230	8743
A5/data/scenes/textures/planks/wood-flooring-041_r.jpg	26422	9319
A5/data/scenes/textures/planks/wood_bump.png	03693	12291
A5/data/scenes/textures/planks/wood_bump_sm.png	07864	4857
A5/data/scenes/textures/planks/wood_diffuse.png	31720	43361
A5/data/scenes/textures/planks/wood_diffuse_sm.png	21613	7848
A5/data/scenes/textures/planks/wood_diffuse_sm_light.png	17156	9475
A5/data/scenes/textures/planks/wood_specular.png	07415	28221
A5/data/scenes/textures/planks/wood_specular_sm.png	50450	7059
A5/data/scenes/textures/planksbump.png	51681	909
A5/data/scenes/textures/sky1.jpg	16119	128
A5/data/scenes/textures/sky1.png	06294	710
A5/data/scenes/textures/sky2.jpg	20445	22101
A5/data/scenes/textures/sky2.png	40154	44689
A5/data/scenes/textures/sky3.png	31869	20272
A5/data/scenes/textures/sky4.png	03545	20821
A5/data/scenes/textures/sky4light.png	39261	26520
A5/data/scenes/textures/uv_test.png	14815	479
A5/data/scenes/textures/weirdbump.jpg	63652	128
A5/data/scenes/textures/weirdbump.png	14538	265
A5/data/scenes/textures/wood_floor.png	07437	8515
A5/data/scenes/tone.lua	04828	5
A5/rt	29844	4503
A5/screenshot01.png	40438	712
A5/src		
A5/src/.DS_Store	33880	7
A5/src/Makefile	10473	2
A5/src/TODO.txt	30706	2

A5/src/a4.cpp	55338	2
A5/src/a4.hpp	27696	1
A5/src/algebra.cpp	53882	4
A5/src/algebra.hpp	29464	14
A5/src/camera.cpp	04933	3
A5/src/camera.hpp	17078	1
A5/src/image.cpp	26000	11
A5/src/image.hpp	12970	3
A5/src/light.cpp	40723	3
A5/src/light.hpp	00784	2
A5/src/lua488.hpp	04702	1
A5/src/main.cpp	38275	1
A5/src/material.cpp	41309	10
A5/src/material.hpp	41540	3
A5/src/mesh.cpp	08052	20
A5/src/mesh.hpp	34753	3
A5/src/options.hpp	41132	1
A5/src/polyroots.cpp	59889	47
A5/src/polyroots.hpp	52382	1
A5/src/primitive.cpp	55641	13
A5/src/primitive.hpp	38626	3
A5/src/ray.cpp	46222	1
A5/src/ray.hpp	45634	2
A5/src/renderer.cpp	53875	12

2015-04-01 14:02 Checksum for A5 for ljfulton on gl02 Page 12

A5/src/renderer.hpp	38288	2
A5/src/run	23820	1
A5/src/scene.cpp	35352	5
A5/src/scene.hpp	22658	3
A5/src/scene_lua.cpp	34310	22
A5/src/scene_lua.hpp	33354	1

A5/src/tiny_obj_loader.cpp	33663	22
A5/src/tiny_obj_loader.h	19203	3