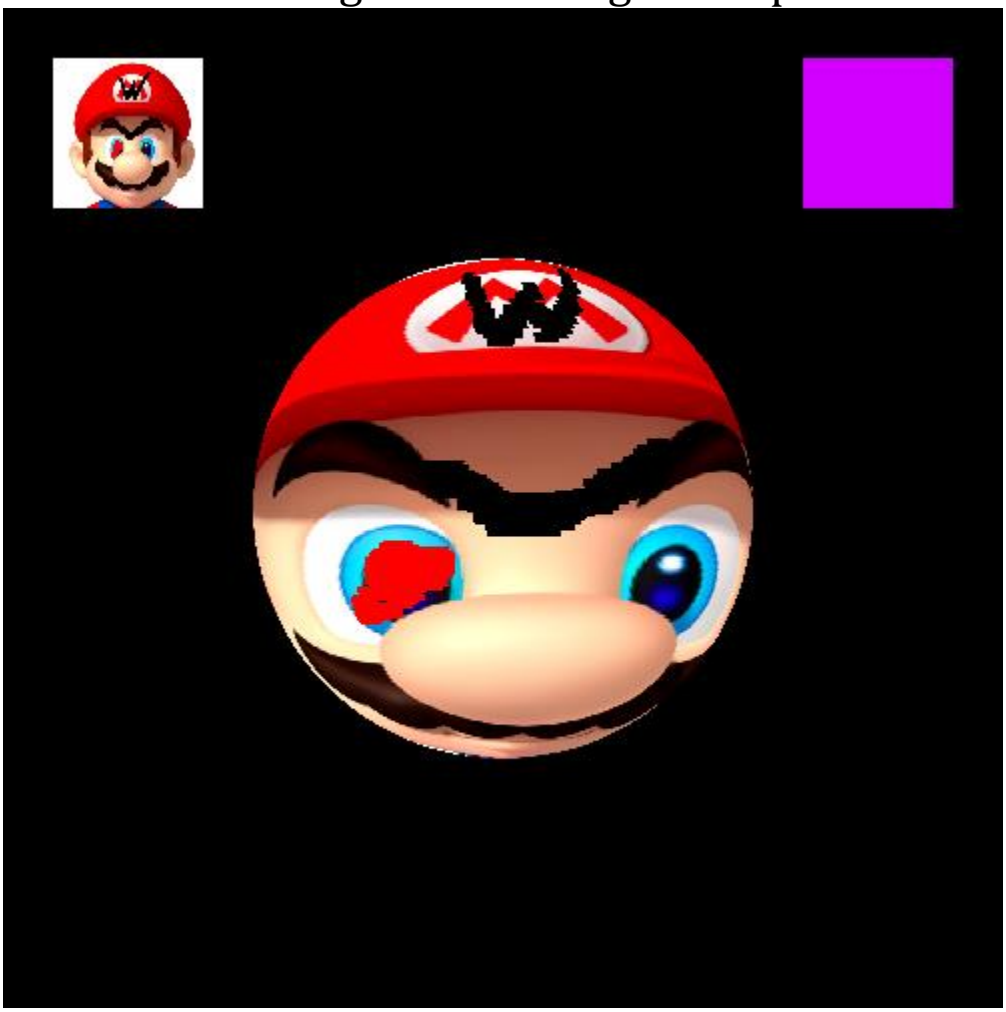
Intersecting and Painting a 3D Sphere



Description:

For this in class assignment we are going to be painting on the surface of a 3D sphere. You will learn about how to cast a ray from the mouse onto a 3D surface. You will then determine if the line has intersected with the surface, and if it has, be able to either show that intersection point or paint on the surface. This lab will also be the first time we use textures on 3D objects.

The purpose of this lab is to get you familiarized with the concepts of: (a) generating a ray, and (b) ray-object intersection (in this case, our object is a sphere that sits at the origin (0, 0, 0) and has a radius of 0.5).

Note that for this lab, we are using ***Ortho camera***. Recall that this is different from a perspective projection. The reason that we are using an Ortho camera is to simply the intersection code. Be careful in thinking about how the use of an Ortho camera changes where your eye point is and how you would need to generate the rays.

To be clear, the rays generated from an Ortho camera is different from the rays discussed in class. Rays from a perspective camera all start from a single eye point, whereas rays from an Ortho camera are parallel to each other.

Your Task:

- You will first build the intersect function to see if a ray intersects with an object
  - You can implement this in the "Intersect" function in main.
  - You will need to fill in the "getEyePoint" function
  - You will need to fill in the "generateRay" function
  - You will need to fill in the "getIsectPointWorldCoord" function
  - You can optionally then draw a ray so you can debug what is going on, by filling in the drawRayFunc();
- Once you know the co-ordinates of where your ray is cast, you can then determine how to paint onto an object based on the mouse position.
  - You can make use of the 'paintTexture' function to modify pixel values.
  - You should be able to see real time updates on both the flat 2D image, and the sphere.

Files Given:

main.cpp – Main where you will implement functions
object.cpp and object.h – Where you can understand how the sphere is rendered and modify the texture coordinates
ppm.cpp and ppm.h – These have been implemented for you
Algebra.h

FYI Reading -- PPM Image format?

We've built a simple PPM parser that will work with ppm files exported from GIMP and other 2D packages. Take a look at some of the notes below to learn more.

Based on: http://netpbm.sourceforge.net/doc/ppm.html

Sample ppm image

```
P3 # feep.ppm
4
4
15
0  0  0    0  0  0    0  0  0    15  0 15
0  0  0    0 15  7    0  0  0    0  0  0
0  0  0    0  0  0    0 15  7    0  0  0
15  0 15   0  0  0    0  0  0    0  0  0
```

OpenGL Textures – How do they work?

https://www.opengl.org/wiki/Texture

Going Further:
Did you enjoy this in class assignment?
Ray casting is a powerful technique that you will be using for your raytracer assignment.

- How would you paint on an arbitrary surface?
  - This could be an interesting final project!
- There are two textures loaded in this lab. See if you can implement a layering system, similar to how photoshop and other tools work.
  - Do layers necessarily need to store color information?
    - Could they store mesh deformation information?
    - Change the colors of the lighting on each vertex?
    - Be Creative!