Exercise 1

Kevin Serrano, Gianni Scarnera

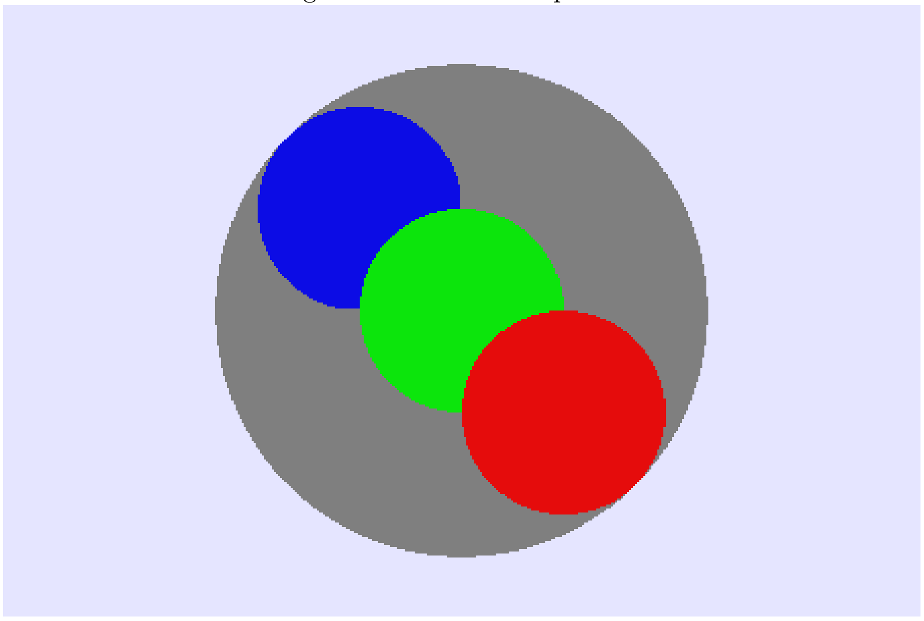
October 2, 2012

Part 1

Part 1.2 : Constant Shading

We saw that the parameters of the function *ConstantShader::shade* were a object *IntersectionData* and a object *Scene*. So in the constructor *IntersectionData* we found the properties of the colour of the intersection. So we just take it and return the colour on the function *ConstantShader::shade* to obtain the result display.

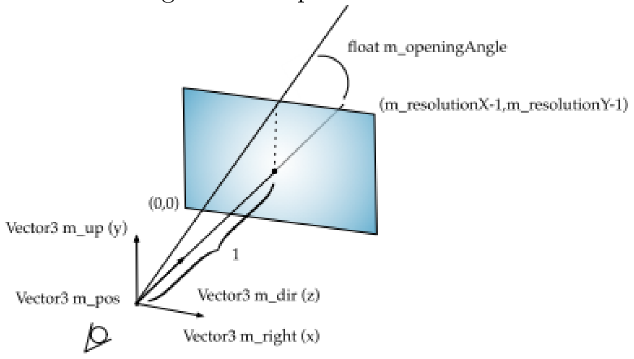
Figure 1: The result of part 1.2



Part 1.3

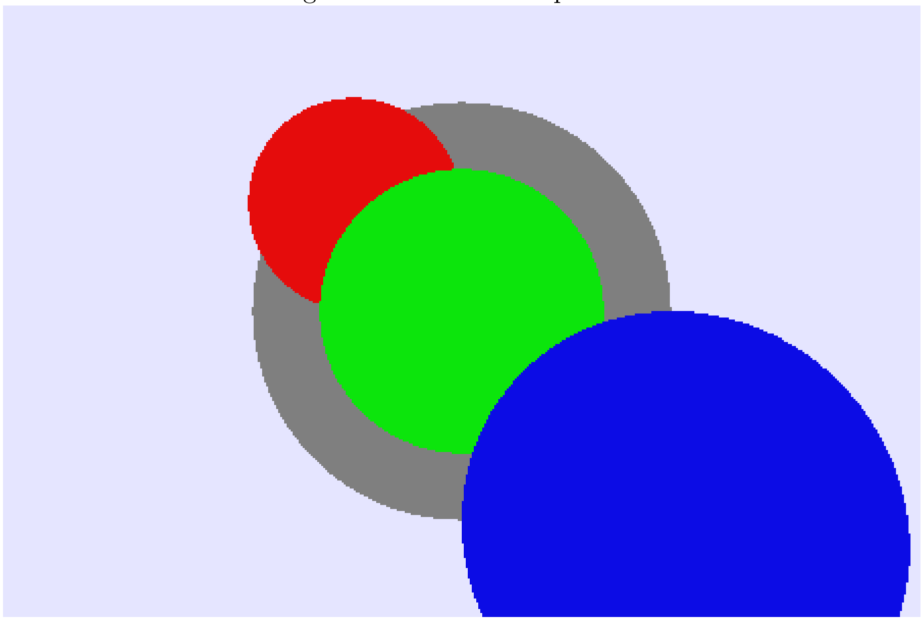
We have to replace the orthographic camera with a perspective one. To do so, we first have to calculate the correct pixel length. Pixel length is a function of the opening angle and the resolution, namely tan(opening angle in rad divided by half the resolution).

Figure 2: SimpleCamera model



Furthermore, the position of the camera is now a single point and the direction of the rays are from the camera to each of the pixels on the screen multiplied by the camToWorld matrix.

Figure 3: The result of part 1.3



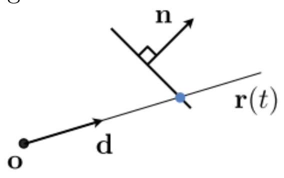
Part 1.4.1 : class Plane

To found the intersection for a plane, we have to resolve the equation

$$(\mathbf{x} - \mathbf{p}) \cdot \mathbf{n} = \mathbf{0}$$

where $\mathbf{x}$ is the point of interest, $\mathbf{p}$ the point on plane and $\mathbf{n}$ the normal vector of the plane. Then we can substitute the ray equation $\mathbf{o} + t\mathbf{d}$ (where $\mathbf{o}$ is the

Figure 4: Plane intersection



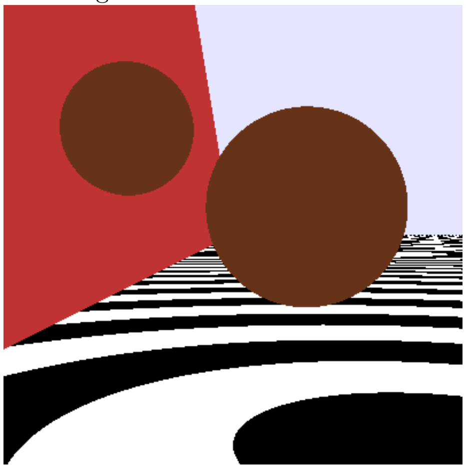
origin of the ray and $\mathbf{d}$ the direction of the ray) for $\mathbf{x}$ and solve for t. We get :

$$t = -\frac{(\mathbf{o} - \mathbf{p}) \cdot \mathbf{n}}{\mathbf{d} \cdot \mathbf{n}}$$

and if $\mathbf{d} \cdot \mathbf{n} = \mathbf{0}$ then that means that the direction of the ray is parallel to te plane and then there is no intersection. So we have to do a test in the code to avoid a crash in compilation.

Then now we can return informations in the intersection by the object of type *IntersectionData*, i.e it's position and it's color with the functions provided by the class *IntersectionData*.

Figure 5: Plane intersection



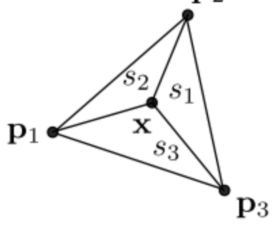
Part 1.4.2 : class Triangle

To do this part, we can use the same technique as before except that now it more complicated to verify if the intersection is in the triangle. We used the first approach to resolve this problem, i.e intersect ray with triangle's plane. First we use the barycentric coordinates

$$\mathbf{x} = s_1\mathbf{p_1} + s_2\mathbf{p_2} + s_3\mathbf{p_3}$$

where s_1, s_2, s_3 are the tree area and $\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}$ the three vertices of the triangles. Finally $\mathbf{x}$ is a point inside the triangle area and in our case it will be the ray intersection.

Figure 6: Barycentric coordinates of a triangle



So the ray intersection with the triangle is given by the following equation

$$(\mathbf{o} + t\mathbf{d} - \mathbf{p_1}) \cdot \mathbf{n} = 0$$

where $\mathbf{n}$ is the triangle's normal :

$$\mathbf{n} = (\mathbf{p_3} - \mathbf{p_1}) \times (\mathbf{p_2} - \mathbf{p_1})$$

and then we can resolve for t :

$$t = -\frac{(\mathbf{o} - \mathbf{p_1}) \cdot \mathbf{n}}{\mathbf{d} \cdot \mathbf{n}}$$

Again if $\mathbf{d} \cdot \mathbf{n} = \mathbf{0}$ then there are no solutions.

Now we can compute the three area s_1, s_2, s_3. To do that we created three vectors $\mathbf{n_{s_1}}, \mathbf{n_{s_2}}, \mathbf{n_{s_3}}$ wich are normal vector of the area s_1, s_2, s_3 respectively. So

- $\mathbf{n_{s_1}}$ is normal to the triangle of surface s_1. Then $\mathbf{n_{s_1}} = (\mathbf{p_2} - \mathbf{p_3}) \times (\mathbf{x} - \mathbf{p_3})$
- $\mathbf{n_{s_2}}$ is normal to the triangle of surface s_2. Then $\mathbf{n_{s_2}} = (\mathbf{p_1} - \mathbf{p_2}) \times (\mathbf{x} - \mathbf{p_2})$
- $\mathbf{n_{s_3}}$ is normal to the triangle of surface s_3. Then $\mathbf{n_{s_3}} = (\mathbf{p_3} - \mathbf{p_1}) \times (\mathbf{x} - \mathbf{p_1})$

where $\mathbf{x}$ is the intersection, given in the code by *ray.getPointOnray(t)* with t as above. Now we have to normalise s_1, s_2, s_3 and verify if

- $s_1 + s_2 + s_3 = 1$
- $0 \leq s_i \leq 1$

and if it is ok, then that means that the intersection $\mathbf{x}$ is in the triangle and we have ton return the value of the object *iData*. To normalise s_1, s_2, s_3, we have the following formula

$$s_i = \frac{\mathbf{n} \cdot \mathbf{n_{s_i}}}{||\mathbf{n}||^2}$$

Then we obtain the result display in the figure below.

Figure 7: Triangle intersection