Reflection

CS500 – Project3

Rahil Momin

Contents:

Changes to Algorithm:

Overall, not much has been changed for the path tracing algorithm, a variable Wo has been added to the path tracing algorithm, which is passed in to `SampleBrdf`, `PdfBrdf` and `EvalScattering` functions. The Wo variable is the opposite direction of the ray that was casted into the scene and is updated every iteration of the algorithm.

```
Wi = P.object->mat->SampleBrdf(Wo, N).normalized();
Color f = P.object->mat->EvalScattering(Wo, N, Wi);
p = Q.object->mat->PdfBrdf(Wo, N, Wi);

Wo = -Wi;
```

Changes to the BRDF:

The following functions have been updated to receive Wo as a parameter and their functionality has been enhanced for reflection.

```
Vector3f SampleBrdf(Vector3f Wo, Vector3f N);
float PdfBrdf(Vector3f Wo, Vector3f N, Vector3f Wi);
Color EvalScattering(Vector3f Wo, Vector3f N, Vector3f Wi);
```

The probabilities for choosing between Diffuse and Reflection is as follows.

$$P_d = \frac{\|K_d\|}{S}, P_r = \frac{\|K_s\|}{S}$$

Where $S = \|K_d\| + \|K_s\|$

In addition, several functions were introduced to mediate the change in functionality.

characteristic function for D() and G():

```
bool CharacteristicX(float val);
```

Fresnal term:

```
Color F(float d);
```

Micro-facet distribution term:

```
float D(Vector3f m, Vector3f N);
```

G Factor:

```
float G(Vector3f Wi, Vector3f Wo, Vector3f m, Vector3f N);
float G1(Vector3f v, Vector3f m, Vector3f N);
```

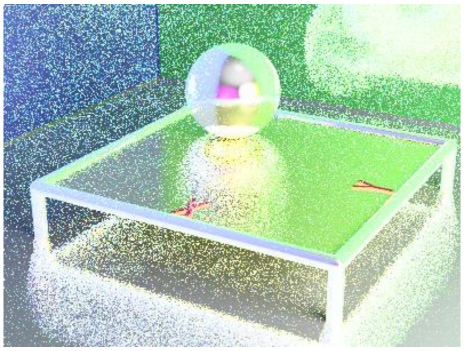
The surface rougness values used in the algorithm are as follows

```
#define ag 0.1
#define ap 2000
```

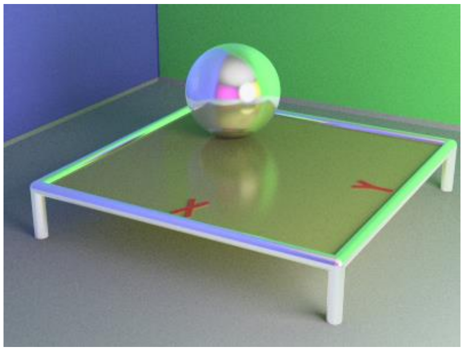
Fireflies' Reduction:

With the implementation of added functions and improved functionality, One more change was added to the pass, due to errors or such the number of fireflies were too high in the output.

The following image is at 4096 passes using Phong with $a_p = 2000$.



Which was then transformed to the following

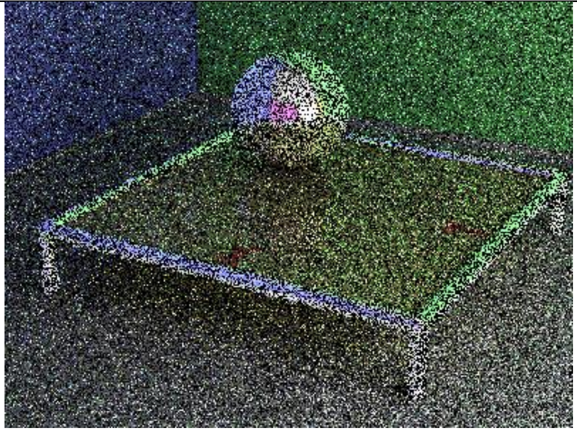
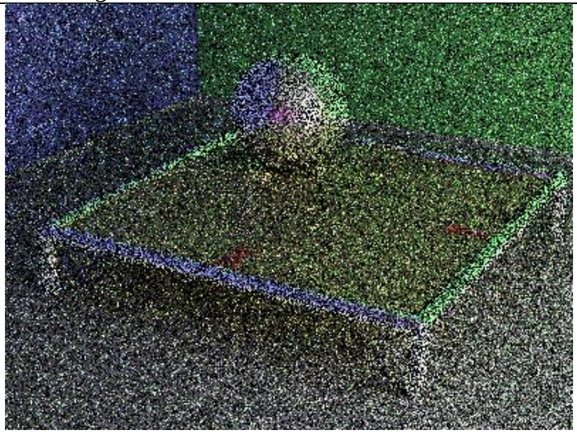
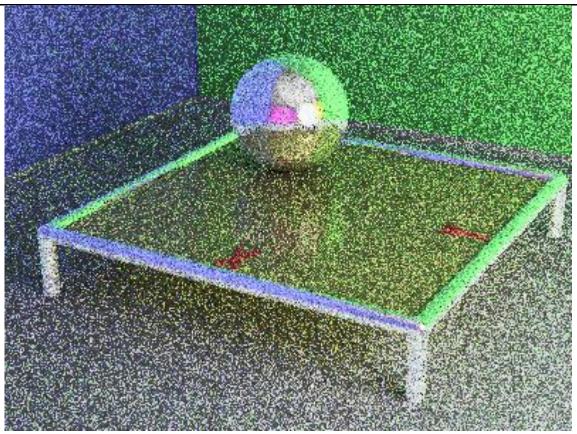
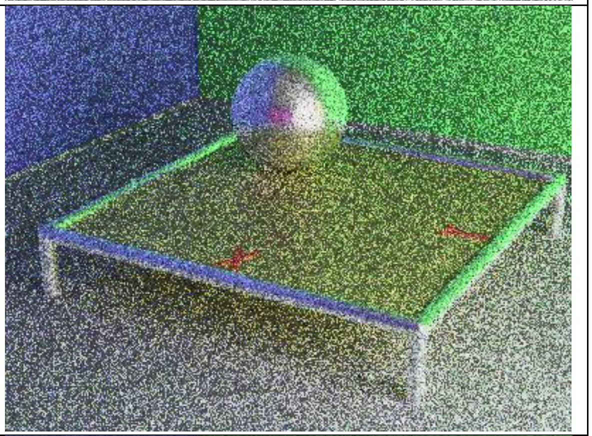
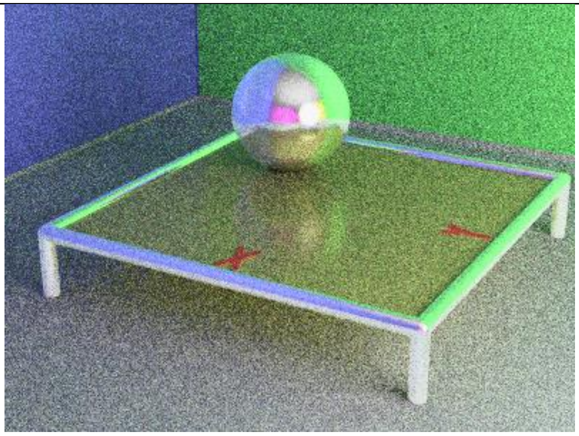
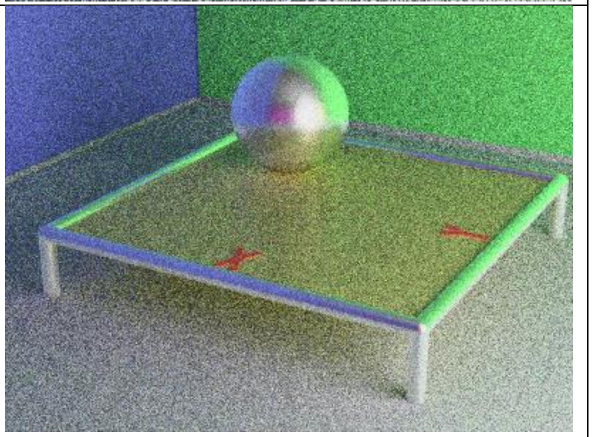


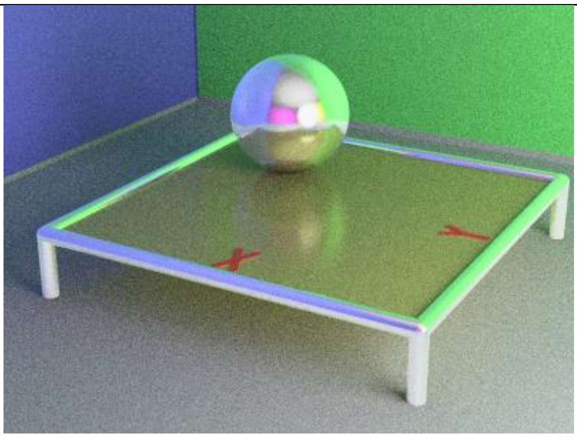
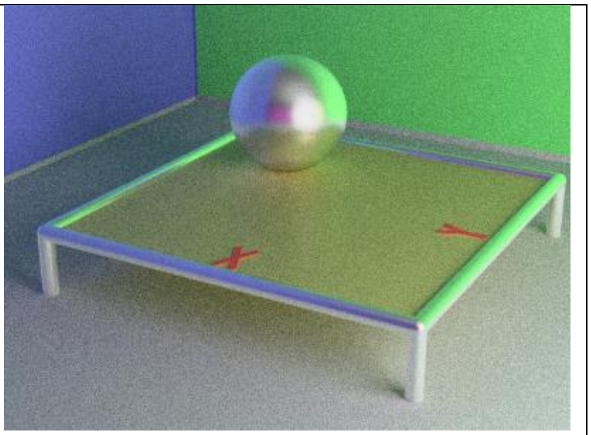
The number of fireflies were reduced by changing the pass as follows

```
Color C;
Vector3f c;

C = raytrace->trace(ray, Tree);
c = C;

if (isnan(C.x()) || isnan(C.y()) || isnan(C.z()) || c.norm() > sqrt(75))
{
  continue;
}

image[y * width + x] += C * 2.0f;
```

The check uses use sqrt(75) because of the intensity of light in the scene which is (5,5,5); therefore any returning output from path tracing algorithm that exceeds intensity of the light or is corrupted is dropped from the output.

Output:

Pass	Phong ($a_p = 2000$)	GGX ($a_g = 0.1$)
1		
8		
64		

512		
4096