# Relatório para a primeira entrega

1. Estrutura base do projeto

Temos as classes das esferas, dos planos, dos triângulos, das luzes, da camara, do raio e do cilindro. Existe também uma classe Object da qual as esferas, os planos, os triângulos e os cilindros herdam uma vez que os objetos tem coisas em comum e seria necessário durante o ray tracing guardar o objeto mais próximo, algo que não se conseguiria fazer sem esta classe.

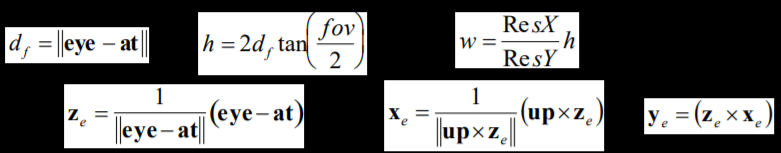
Existem dois vetores no programa principal. Um com objetos e outro com luzes que serão utilizados durante o algoritmo de ray tracing.

1. Parser NFF File

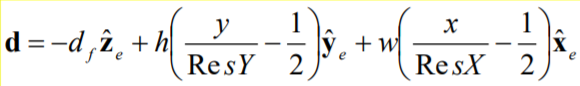
Para realizar o parser do NFF file foi utilizado a função getline() do c++. Basicamente, lê-se uma linha e depois essa linha é lida outra vez por outro getline() mas este tem um delimitador, neste caso o espaço vazio ‘ ‘, que nos permite separar a linha já lida em diferentes tokens( palavras delimitadas por espaços vazios).

Assim que o primeiro token é lido é então feita a verificação com diversos ifs de que comando é que se trata, por exemplo se é um “b” ou um “f”, etc… Depois de feita a verificação procede-se então a criação do objeto como por exemplo uma esfera ou um plano e colocam-se estes mesmos no respetivo vetor de objetos.

1. Raio primário

 Dado uma posição (x,y) é calculado o raio primário a partir dessa posição. Os cálculos efetuados para o cálculo do raio primário foram os seguintes:

E foi utilizado a formula do ray que passa no cante inferior esquerdo do pixel.



1. Ray tracing

O algoritmo de ray tracing funciona de uma maneira bastante fácil. Os raios primários calculados anteriormente são utilizados para realizar um verificação se este interseta algum objeto na cena ou não. Caso o raio intersete algum objeto então procede-se ao cálculo das contribuições de cada uma das luzes no local de interseção. É possível que o local de interseção esteja na sombra de algum objeto, ou seja, não tem contacto direto com a fonte de luz pelo que não haverá contribuição dessa fonte de luz para a cor do pixel.

A verificação se o ponto de interseção se encontra na sombra de um objeto é feito através do cálculo de um raio secundário que terá como origem o ponto de interseção e a direção no sentido da fonte de luz. Caso este raio intersete algum objeto então o ponto de interseção esta na sombra desse objeto.

No entanto, se nada anteriormente dito não acontecer então é devolvida a cor do fundo da cena.

Por fim, é verificado se o objeto mais próximo é refletor ou transmissor e em caso afirmativo são calculados os raios refletidos ou os raios refratados e a contribuição dos mesmo para a cor da interseção.

1. Interseções raio-objeto
   1. Esferas

Para o cálculo das interseções do raio com a esfera foi utilizado o algoritmo fornecido pelo professor.

* Primeiro, normaliza-se a direção do raio caso já não se tenha normalizado
* Segundo, calcula-se o quadrado da distância da origem do raio ao centro da esfera.
* Terceiro, calculamos o B(que representa o produto interno entre a direção do raio e o vetor OC) através da seguinte formula



É feita uma verificação, se o quadrado da distância calculado no segundo passo for maior que o raio^2, então verificar se o B é negativo e, em caso afirmativo, abandonar os cálculos porque isso significa que o raio aponta no sentido contrário a localização da esfera relativamente à origem do raio.

* Quarto, calcular o R que corresponde ao que esta no interior da raiz da fórmula resolvente e abandonar os cálculos caso o R seja negativo uma vez que não há raízes negativas.
* Quinto, determinar t e em seguida calcular o ponto de interseção. Também se deve calcular a normal à superfície da esfera no ponto de interseção trocando os sinais das componentes caso a origem do raio se encontra no interior da esfera.
  1. Planos

Com a equação implícita de um plano e a equação paramétrica do raio é possível deduzir a expressão para o valor do parâmetro t no ponto de interseção. Todos as variáveis em baixo são calculadas a partir dos 3 pontos que definem um plano ou são variáveis do raio. Portanto é realizado primeiro o produto interno entre N e R e se este for nulo, reta paralela ao plano, não existira interseção. Senão, calcula-se o valor de t e se este for menor que 0 então a interseção é atras da origem do raio.

* 1. Triângulos

Esta interseção foi realizada com o algoritmo de Tomas Moller onde são utilizadas as coordenadas baricêntricas. É possível então deduzir uma equação que quando transformada num sistema de equações é possível resolver uma vez que existem 3 incógnitas e 3 equações. Utilizou-se a regra de Cramer’s uma vez que é o método clássico mais rápido para resolver equações lineares 3 X 3.

A medida que vamos calculando as incógnitas são verificadas se algumas condições são compridas. Caso corra tudo bem é então calculado o ponto de interseção.

* 1. Cilindro

Utilizando a equação do círculo e do raio, substituindo, obtemos uma equação que pode ser expandida para uma equação quadrática.

São feitas as contas para a fórmula resolvente e os dois valores possíveis são guardados e ordenados. É então que se calculam os dois raios possíveis(y1,y2) baseados nos valores de t devolvidos pela formula resolvente.

É importante referir que as contas funcionam para cilindro que se encontram no eixo do y.

Se y1>base.y e y2>base.y então o raio passa ao lado.

Se y1>base.y e y2<base.y então o raio interseta a base positiva.

Se y1<base.y e y1>apex.y então o raio interseta a secção entre as duas bases.

Se y1<apex.y e y2>apex.y então o raio interseta a base negativa.

Se y1<apex.y e y2<apex.y então o raio passa ao lado.

1. Shadow Feeler

É calculado a direção da luz dando o ponto de interseção mais próximo. De seguida é criado um raio com essa direção. É então com a ajuda deste raio que é verificada a interseção com todos os objetos para verificar se o ponto de interseção se encontra ou não na sombra. Se não tiver são calculadas as contribuições da componente difusa e especular.

* 1. Componente difusa

Color = LN \* rgb \* kd \* light.rgb;

* 1. Componente especular

Calcular o half vetor com a direção da luz e o raio incidente. Normalizar. Calcular o produto interno e se for maior que 0 calcular a cor.

Color = pow(dotHN, shine) \* ks \* light.rgb;

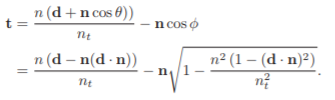
1. Reflecção

Se o objetivo for refletor então calcula -se o produto interno entre a normal e a direção do raio.

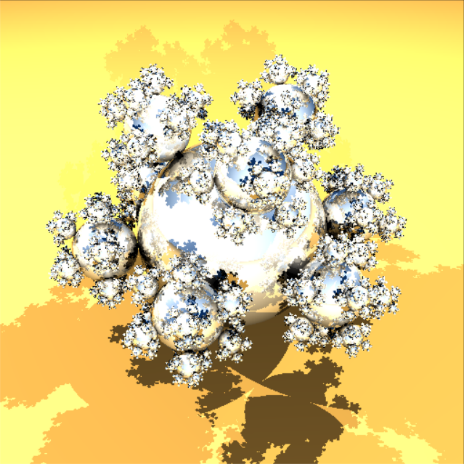
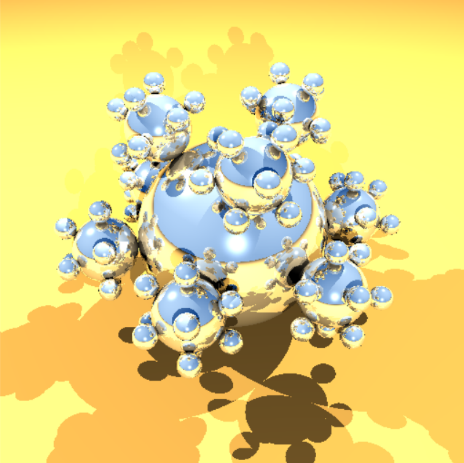
Calcula-se a direção do raio refletido: ray.d – (2 \* dotND \* normal), como nos slides.

A direção é normalizada e devolve-se o raio refletido.

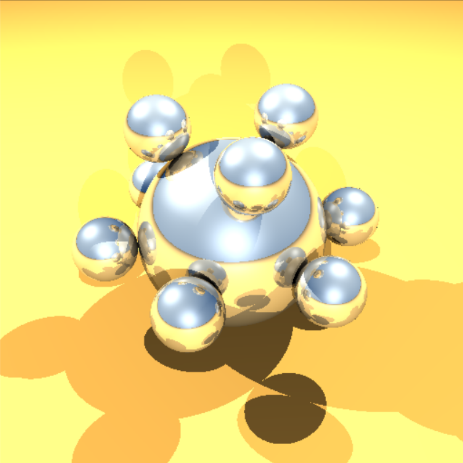
1. Transmissão

Para o cálculo do vetor transmissor foi utilizado a fórmula apresentada num dos pdf auxiliares disponibilizados na página da cadeira. Mais concretamente esta:

É importante referir que se o resultado dentro da raiz for negativo é devolvido um raio vazio, ou seja, não houve refração e toda a energia foi para o raio refletido sendo este fenómeno conhecido como reflexão total.

1. Tempos de execução e fotos

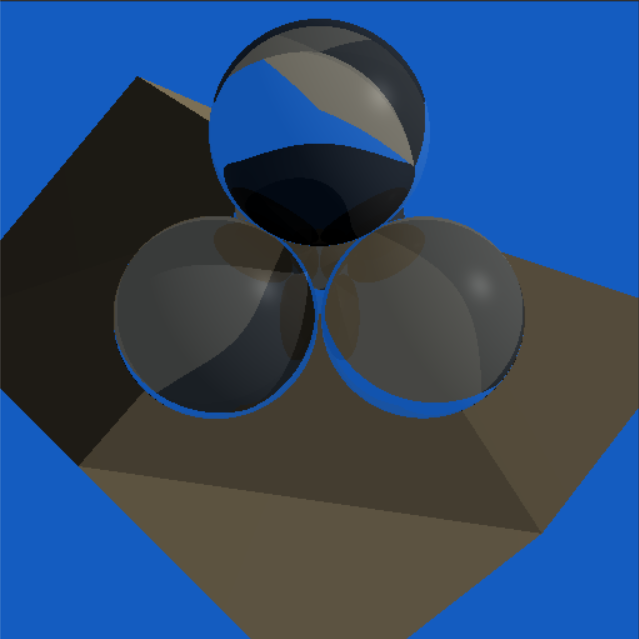
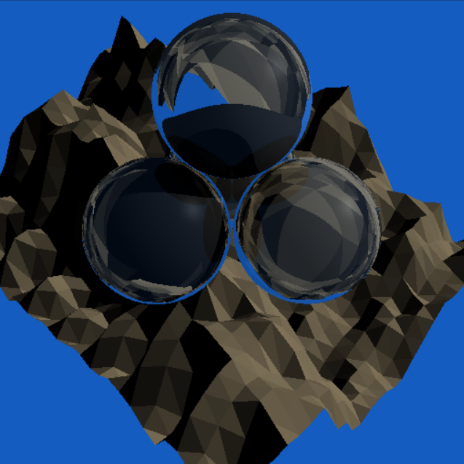
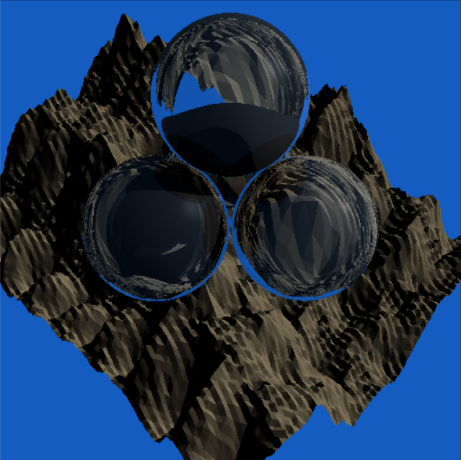
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nome | Balls\_low | Balls\_medium | Balls\_high | Mount\_low | Mount\_high | Mount\_very\_high |
| Tempo(s) | 6s | 22s | 1456s | 8s | 965s | 16801s |



Balls\_high

Balls\_medium

Balls\_low



Mount\_very\_high

Mount\_high

Mount\_low