Ray Tracing

João Vitor Farias Silva (jvfsilva@inf.ufpel.edu.br)
Graziele Fagundes Martins (gfmartins@inf.ufpel.edu.br)

Aplicação escolhida

Para este trabalho escolhemos a aplicação gráfica Ray tracing. Ray tracing é uma técnica de renderização que simula a forma como a luz interage com objetos em um ambiente tridimensional para criar imagens realistas. Ela funciona traçando os caminhos que os raios de luz seguiriam ao sair de uma fonte de luz, passando por uma cena e atingindo a câmera.

Linguagens

Para a interface visual utilizamos Python. E para realizar os cálculos complexos necessários para a geração da imagem, recorremos à linguagem C.

Interface entre as linguagens

Para integrar as linguagens Python e C em nosso projeto, utilizamos a biblioteca “ctypes” do Python para facilitar a comunicação entre essas linguagens. O processo envolve compilar o código-fonte C como uma biblioteca compartilhada e depois acessá-la a partir do código Python.

1. Compilação do Código C

O código C é compilado para gerar uma biblioteca compartilhada no formato .so (shared object). O comando utilizado para essa tarefa é:

```
gcc -fPIC -shared -o raytracing.so raytracing.c utils.c
```

Neste comando:

- -fPIC instrui o compilador a gerar código independente de posição, necessário para criar bibliotecas compartilhadas.
- -shared indica ao compilador que o objetivo é criar uma biblioteca compartilhada em vez de um executável.
- -o raytracing.so especifica o nome do arquivo da biblioteca compartilhada gerada.

Após a execução deste comando, o resultado é um arquivo raytracing.so, que contém o código compilado de raytracing.c e utils.c.

2. **Utilização da Biblioteca Compartilhada em Python**

Para utilizar a biblioteca compartilhada no código Python, carregamos o arquivo .so usando a biblioteca ctypes:

```
PATH = './raytracing.so'
raytracing = ctypes.CDLL(PATH, winmode=0)
```

A função ctypes.CDLL(PATH) carrega a biblioteca compartilhada e retorna um objeto CDLL que permite interagir com as funções definidas na biblioteca C. Depois de carregar a biblioteca, é necessário definir os tipos dos argumentos e o tipo de retorno das funções que serão utilizadas. No nosso caso, a função PerPixel é configurada da seguinte forma:

```
raytracing.PerPixel.argtypes = [ctypes.c_float, ctypes.c_float]
raytracing.PerPixel.restype = ctypes.c_int
```

- argtypes define que PerPixel espera dois argumentos do tipo float (ctypes.c_float).
- restype especifica que PerPixel retorna um valor do tipo int (ctypes.c_int).

Com essas configurações, a função PerPixel, que realiza cálculos complexos de ray tracing no código C, pode ser chamada a partir do Python. Isso nos permite gerar imagens e manipular dados de forma eficiente, aproveitando a combinação das capacidades das duas linguagens.
s

Resultado

Após a conclusão, o resultado esperado é mostrado na imagem abaixo. É importante ressaltar que a qualidade da imagem gerada depende do valor da variável *numSamples* no código Python. Ao aumentar esse valor, uma imagem mais nítida será produzida, mas isso resultará em um maior tempo de execução.