Linguagem SDL - Signed Distance Language

Aluno: André Corrêa

Professor: Raul Ikeda

Insper - Instituto de Ensino e Pesquisa

andrecs11@al.insper.edu.br

05/06/202

Overview

Introdução

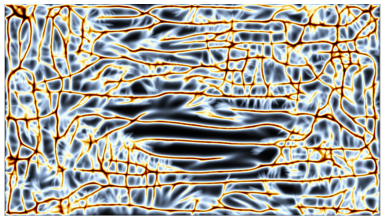
- SDL é inspirada em *"shading languages"*, linguagens nas quais a execução dos programas é inteiramente paralela e a saída de cada execução do programa normalmente é armazenada em buffers que compõem, de alguma forma, a imagem final apresentada na tela.
- Por limitações de tempo e capacidade, não foi possível compilar o código para SPIR-V de modo a executar o binário na placa de vídeo. Dessa forma a execução de programas na linguagem é bastante lenta quanto comparada à execução desses mesmos programas em outras linguagens como glsl que são passíveis de execução na gpu.

Motivação para criar a linguagem

- **Abstração de Overhead**: SDL surge da vontade de eliminar a barreira de ter que escrever dezenas de linhas para poder rodar o primeiro shader. Normalmente, é necessário já possuír um renderizador em Opengl ou vulkan (mais overhead ainda) para poder carregar qualquer shader já compilado para placa de vídeo e sincronizar a renderização e apresentação dos framebuffers.

- **Exemplos**: Sites como shadertoy e compute toys já permitem escrever e visualizar a saída de shaders (fragment shaders e compute shaders respectivamente) em navegadores web, entretanto, em ambos os casos não é possível injetar dados de cena como malhas de polígonos, por exemplo, de forma a obter uma forma concreta e definida:

(a) Imagem renderizada a partir de malha de polígonos. ()fonte



(b) Imagem renderizada com algoritmo bio-inspirado. (fonte

)

Solução

- **Gambiarra**: Para renderizar formas definidas e determinísticas mesmo sem input de objetos 3d, artistas técnicos utilizam de "campos de distância com sinal", que são uma forma de expressar um objeto tridimensional matemáticamente.
- **Signed Distance Fields**: SDFs mapeiam à todo ponto do espaço um valor que representa a distância do determinado ponto à superfície mais próxima e caso o ponto esteja dentro da superfície parametrizada o valor da SDF é negativo.

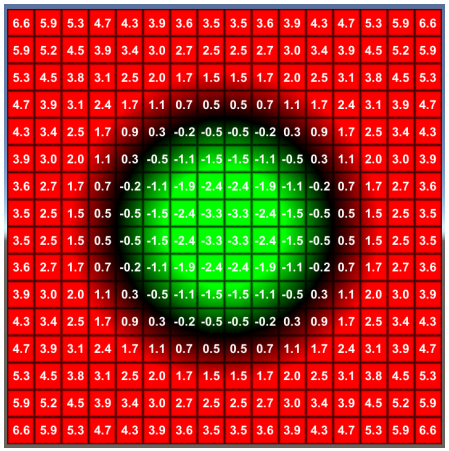


Figure: SDF bidimensional de um círculo fonte

- **Raios:** Para renderizar cenas parametrizadas por sdfs, é necessário partir da posição da câmera e lançar raios que atravessam o plano da imagem a ser renderizada. Cada raio deve atravessar um pixel correspondente no plano da imagem e marchar em direção à cena.

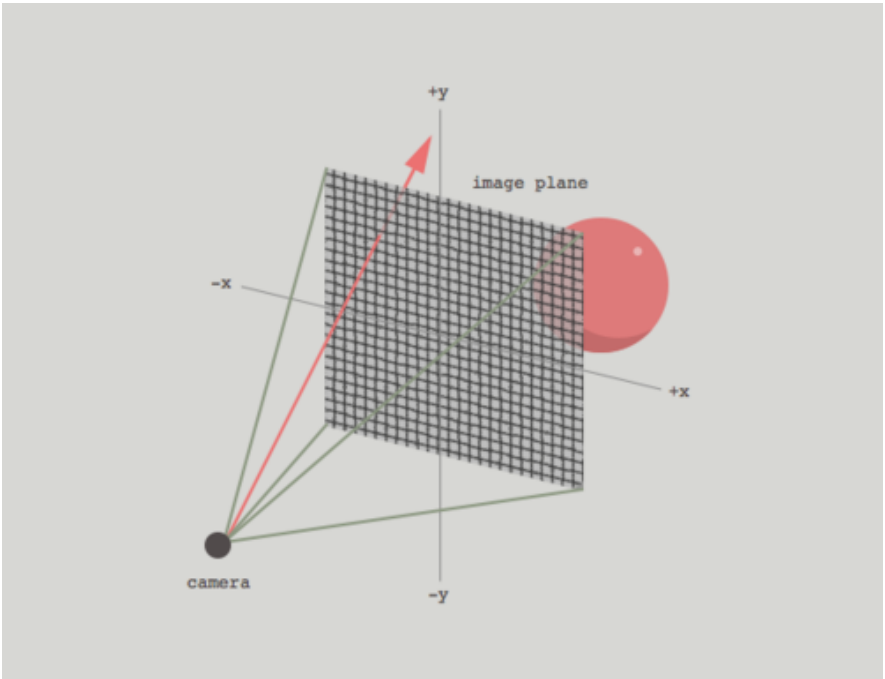


Figure: plano da imagem, fonte

- **Hit or Miss:** Os raios lançados são amostrados regularmente. Caso a distância em relação a alguma superfície calculada para uma dada amostragem de um raio seja menor que um *"threshold"*, considera-se que o raio atingiu a superfície em questão. A amostragem desse raio é interrompida e a cor da superfície atingida é desenhada no pixel atravessado pelo raio.

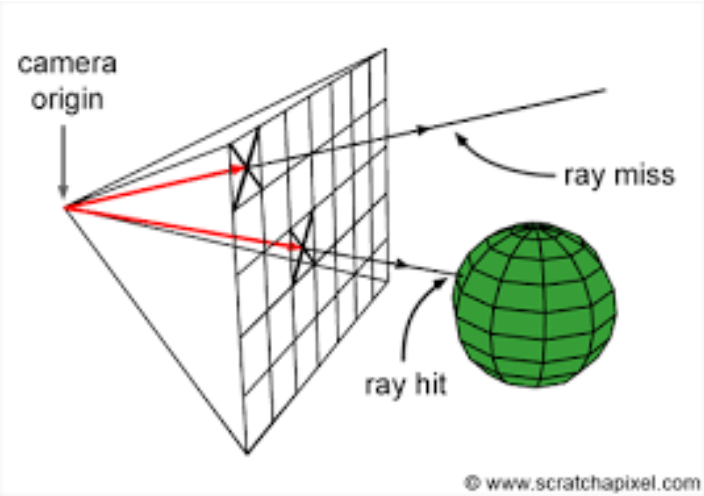


Figure: Amostragem de um raio que atingiu uma superfície, o pixel atravessado pelo raio que atingiu a esfera deverá ser colorido de verde. fonte

Exemplos

- As imagens finais renderizadas podem ser fantásticas e dependem apenas da capacidade do artista de escrever SDFs e funções de iluminação.

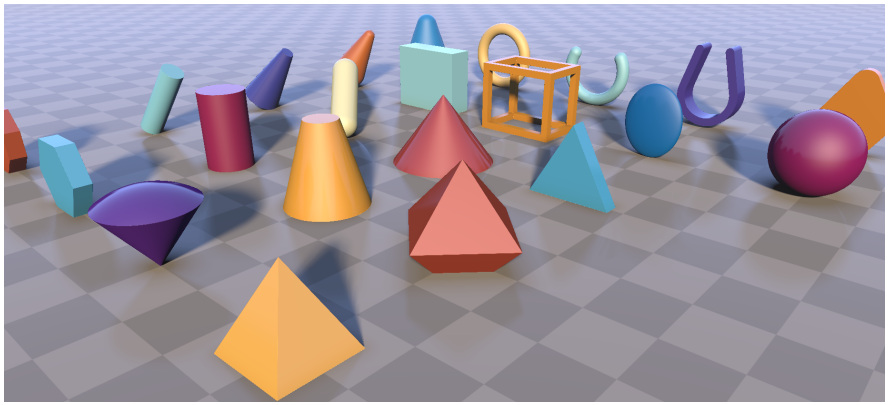


Figure: Artistas muito talentosos como Iñigo Quilez são capazes de criar cenas compostas por dezenas de SDFs diferentes. fonte

Linguagem

- **SDL:** O objetivo então da linguagem é mais do que abstrair a construção do renderizador, abstrair também o processo de amostragem volumétrica da cena que chamamos de raymarching. Dessa forma, objetiva-se poder escrever apenas as funções de distância, funções de cor da cena e a linguagem vale-se do algoritmo de raymarching para renderizar a cena descrita pelo usuário na linguagem.

Código 1 Exemplo

exemplo_codigo.png

Saída Exemplo 1

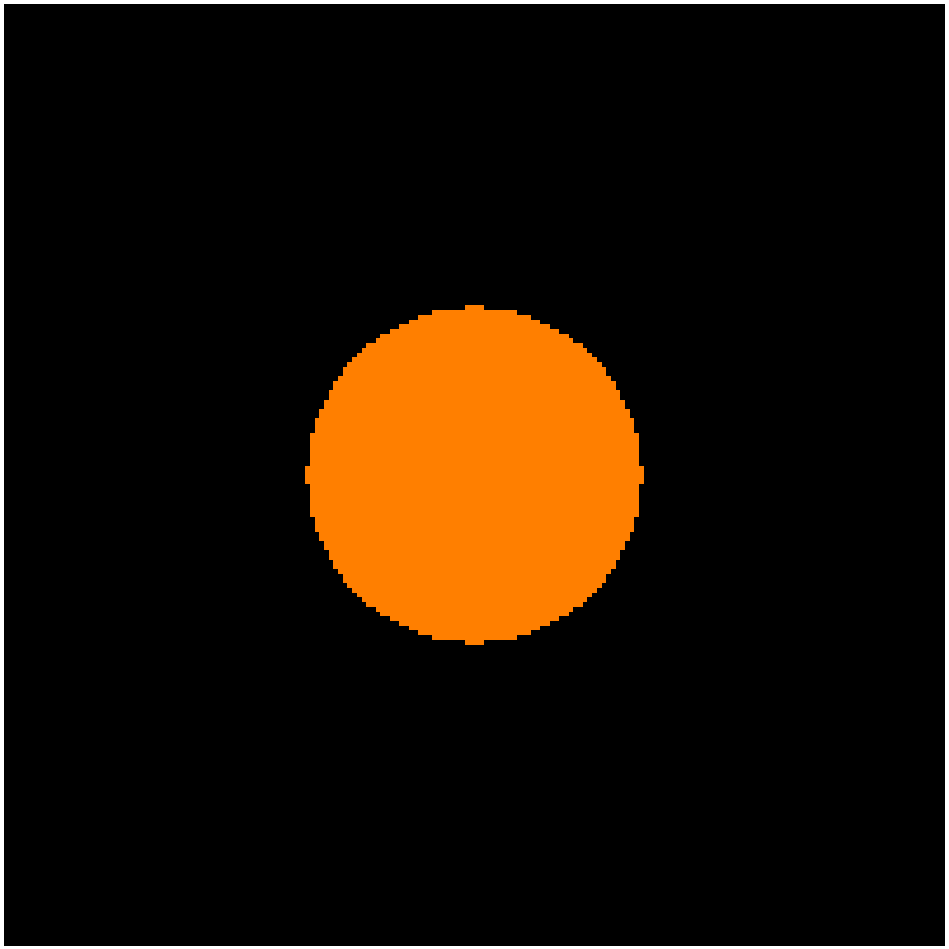


Figure: Saída obtida para o primeiro exemplo

Arquitetura (Output) 2 Exemplo

imgs/neural_network_2.png

Código 3 Exemplo

imgs/code_3.png

Arquitetura (Output) 3 Exemplo

imgs/neural_network_3.png