Universidade da Beira Interior
Departamento de Informática

Nº 121 — 2022
***Visualização de funções implícitas por* ray marching**

Elaborado por:

Diogo Castanheira Simões

Orientador:

Professor Doutor Abel João Padrão Gomes

4 de julho de 2022

Agradecimentos

Agradeço primeiramente a Deus.
E a mais ninguém.

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

API	*Application Programming Interface*
CPU	*Central Processing Unit*
GLAD	*Multi-Language GL/GLES/EGL/GLX/WGL Loader-Generator*
GLM	*OpenGL Mathematics*
GLSL	*OpenGL Shader Language*

Capítulo

1

Introdução

1.1 Enquadramento

O estudo matemático de funções levou à descoberta de diferentes métodos para a sua representação e análise, nomeadamente os modelos explícitos, paramétricos e implícitos. Contudo, apenas as funções explícitas são trivialmente renderizadas por computadores; estas representam grandes desafios uma vez que nem sempre é possível obter expressões explícitas.

Neste sentido, o estudo de funções implícitas torna-se vital para inúmeros fins, pelo que a sua renderização computacional tem sido alvo de estudo por décadas. Os resultados dos enormes avanços feitos na área são atualmente desfrutados por milhões de pessoas, tanto a título pessoal como no mundo empresarial e de investigação.

1.2 Motivação

As versões modernas do *OpenGL*® (i.e. a partir da versão 3.3) permitem que o programador escreva os seus próprios *shaders* a serem utilizados na *pipeline* de renderização. Uma consequência imediata é a possibilidade de paralelizar inúmeros cálculos que normalmente seriam realizados pela *Central Processing Unit* (CPU).

Ainda assim, uma miríade de *software* de renderização não tira proveito de tais capacidades. Desta forma, é de interesse estudar um algoritmo de renderização por volume e analisar a sua potencial paralelização.

1.3 Objetivos

O presente projeto tem por **objetivo principal** implementar um sistema de visualização de funções implícitas.

Este tem ainda os seguintes **objetivos secundários**:

- Estudar funções implícitas e o cálculo das respetivas iso-superfícies;
- Analisar o modelo de renderização por volume;
- Implementar o algoritmo de *ray marching* em particular;
- Implementar aceleração por *hardware* através da programação de *shaders* em *OpenGL*®.

1.4 Organização do Documento

O presente relatório estrutura-se em seis capítulos:

1. No primeiro capítulo — **Introdução** — é apresentado o projeto, em particular o seu enquadramento e motivação, assim como os seus objetivos e a respetiva organização do relatório.
2. No segundo capítulo — **Estado da Arte** — .
3. No terceito capítulo — **Tecnologias e Ferramentas** — .
4. No quarto capítulo — **Implementação** — .
5. No quinto capítulo — **Testes e Resultados** — .
6. No sexto capítulo — **Conclusões e Trabalho Futuro** — .

Capítulo

2

Estado da Arte

2.1 Introdução

A visualização computacional de funções é a base de muitas aplicações práticas em computação gráfica, tais como em videojogos e visualização molecular. O estudo destas funções permite igualmente outro tipo de cálculos, tais como colisões.

Para alcançar os objetivos propostos no presente projeto, é imperativo estudar os seguintes tópicos:

- Funções implícitas;
- Técnicas de renderização em geral e algoritmos volumétricos em particular;
- Uso da *Application Programming Interface* (API) *OpenGL*®e programação em *OpenGL Shader Language* (GLSL).

2.2 Funções Implícitas

2.2.1 Definição e Aplicações

Às funções definidas em função de uma variável dá-se o nome de **funções explícitas**. Exemplos clássicos em $\mathbb{R}^2$ incluem equações de retas ($y = mx + b$) e parábolas ($y = ax^2 + bx + c$). Ora, nem todos os subconjuntos de pontos no espaço cartesiano podem ser definidos por funções explícitas. Um exemplo comum em $\mathbb{R}^2$ é a circunferência:

$$(x - x_0)^2 + (y - y_0)^2 = r^2 \tag{2.1}$$

onde (x_0, y_0) é o centro e r o raio.

Sendo uma equação de segundo grau, é possível representá-la através de duas funções explícitas:

$$y = y_0 + \sqrt{r^2 - (x - x_0)^2} \tag{2.2}$$

$$y = y_0 - \sqrt{r^2 - (x - x_0)^2} \tag{2.3}$$

Esta transformação só é possível até polinómios de grau 4, tornando-se impossível para graus superiores. Contudo, estas expressões polinomiais continuam a ser subconjuntos válidos de $\mathbb{R}^n$, necessitando então de formas alternativas de representação. Dois métodos e respetivas representações da circunferência são:

1. **Funções paramétricas**: cada eixo é definido em ordem a uma variável adicional t:
$$\begin{cases} x = r\cos(t) \\ y = r\sin(t) \end{cases} \tag{2.4}$$
2. **Funções implícitas**: a equação não é definida a ordem a uma variável em particular (equação (2.1)).

Uma função implícita é então definida por $f : \mathbb{R}^n \longrightarrow \mathbb{R}$, ou seja, para qualquer ponto em $\mathbb{R}^n$ é determinado um resultado em $\mathbb{R}$. Dependendo da função, o valor obtido pode ter significado, tal como uma grandeza física (*e.g.* densidade de um líquido ou sua temperatura a cada ponto do espaço). Esta função diz-se **algébrica** caso seja polinomial em cada variável.

Por seu turno, em $\mathbb{R}^3$, a **iso-superfície** de uma função implícita é a superfície que satisfaz a condição $f(\mathbf{x}) = 0$ (onde, doravante, $\mathbf{x} \equiv (x, y, z)$). Esta pode ser suavizada através de um parâmetro $s \in \mathbb{R}$ tal que $f(\mathbf{x}) - s = 0$.

[TODO] Mais aplicações.

2.2.2 Desafios Computacionais

[Hint] Renderização em computação gráfica. Que métodos existem? Que alternativas estão em aberto?

2.3 Técnicas de Renderização

[Hint] Breve introdução às categorias de técnicas/algoritmos de renderização.

2.3.1 Renderização por Volume

[Hint] O que é "volume rendering"? Que exemplos de algoritmos existem?

2.3.2 *Ray Marching*

[Hint] Como funciona o algoritmo? É paralelizável? Se sim, como e porquê?

2.4 *OpenGL®*

[Hint] Não recomendo um rip-off do meu relatório, mas ele pode servir de base para esta secção, tentando melhorá-lo e corrigir possíveis gafes.

2.5 Conclusões

...Whiskas Saquetas.

Capítulo

3

Tecnologias e Ferramentas

3.1 Introdução

Bla bla inicial...

3.2 Tecnologias

- **C++**: Linguagem compilada de uso geral multi-paradigma. Desenvolvida por Bjarne Stroustrup.
- ***FreeType*** **[1]**: biblioteca de desenvolvimento dedicada à renderização de fontes em *bitmaps* utilizáveis, por exemplo, pelo *OpenGL*®.
- **GLFW [2]**: API simplificada para o *OpenGL*®, igualmente multi-plataforma, permitindo a gestão de janelas, contextos, superfícies e comandos (rato, teclado e *joystick*);
- ***Multi-Language GL/GLES/EGL/GLX/WGL Loader-Generator*** **(GLAD) [3, 4]**: gerador automático de *loaders* para *OpenGL*®;
- ***OpenGL Mathematics*** **(GLM) [5]**: biblioteca matemática baseada na linguagem dos *shaders* do *OpenGL*®, GLSL.
- **GLSL**: principal linguagem de *shading* para *OpenGL*®.
- ***OpenGL***® **[6]**: API multi-plataforma e com suporte a múltiplas linguagens de programação para a renderização de gráficos vetoriais 2D e 3D com recurso à placa gráfica;

Software / Tecnologia	Versão
Aplicação *OpenGL*®	
OpenGL®	4.6
GLFW	3.3.5
GLAD	0.1.34
GLM	0.9.9.8
FreeType	2.10.4
Relatório	
XeTEX	3.141592653-2.6-0.999993
TeXstudio©	4.2.2
Controlo de versões	
git	2.36.1
GitKraken	8.6.1

Tabela 3.1: Ferramentas e tecnologias utilizadas, organizadas por categoria.

3.3 Código *Open Source*

- ***CParser***: biblioteca para *parsing* de uma sequência de caracteres como uma expressão usando o algoritmo *Dijkstra's Shunting-yard*;

3.4 Conclusões

...Whiskas Saquetas.

Capítulo

4

Implementação

4.1 Introdução

Bla bla inicial...

4.2 Requisitos

Ser operado por alguém que saiba que não deve fazer `sudo apt remove python`.
[Hint] Agora a sério, uma breve lista de requisitos funcionais.

4.3 Lógica e Estruturação

[Hint] Estrutura do código e respetivo fluxo.

4.4 Detalhes de Implementação

4.4.1 Injeção de Funções em *Shaders*

Work in progress...

4.4.2 *Ray Marching* em *Fragment Shaders*

[Hint] Aceleração por *hardware*.

4.5 Execução

Programa goes brrr.

4.6 Conclusões

...Whiskas Saquetas.

Capítulo

5

Testes e Resultados

5.1 Introdução

Bla bla inicial...

5.2 Secções?

A analisar...

5.3 Conclusões

...Whiskas Saquetas.

Capítulo

6

Conclusões e Trabalho Futuro

6.1 Conclusões

Concluí que sou muito *sexy* UwU

6.2 Trabalho Futuro

Pêssegos.

Bibliografia

[1] W. Lemberg, “The FreeType Project,” 2021, [Online] https://www.freetype.org/. Último acesso a 3 de janeiro de 2020.

[2] G. Project, “An OpenGL library | GLFW,” 2020, [Online] https://www.glfw.org/. Último acesso a 11 de julho de 2021.

[3] D. Herberth, “Dav1dde/glad: Multi-Language Vulkan/GL/GLES/EGL/-GLX/WGL Loader-Generator based on the official specs,” 2021, [Online] https://github.com/Dav1dde/glad. Último acesso a 24 de novembro de 2020.

[4] ——, “Glad,” 2021, [Online] https://glad.dav1d.de/. Último acesso a 24 de novembro de 2020.

[5] T. K. G. Inc, “GLM SDK contribution,” 2021, [Online] https://www.opengl.org/sdk/libs/GLM/. Último acesso a 21 de setembro de 2020.

[6] ——, “OpenGL,” 2021, [Online] https://www.opengl.org/. Último acesso a 21 de setembro de 2020.