

FACULDADE DOCTUM DE CARATINGA  
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

MATEUS FREITAS DA COSTA  
PAULO RICARDO

**UMA ANÁLISE COMPARATIVA DE BIBLIOTECAS JAVASCRIPT PARA A  
RENDERIZAÇÃO DE MODELOS 3D EM TEMPO REAL EM AMBIENTES WEB**

CARATINGA  
2023

FACULDADE DOCTUM DE CARATINGA  
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

MATEUS FREITAS DA COSTA  
PAULO RICARDO

**UMA ANÁLISE COMPARATIVA DE BIBLIOTECAS JAVASCRIPT PARA A  
RENDERIZAÇÃO DE MODELOS 3D EM TEMPO REAL EM AMBIENTES WEB**

Trabalho de Conclusão de Curso apresentado à Faculdade  
de Ciência da Computação da Faculdade Doctum de  
Caratinga, como requisito para a aprovação na disciplina de  
TCC I.

Orientador: XXXX-XXXX-XXXX

CARATINGA  
2023

## SUMÁRIO

GLOSSÁRIO.....	3
LISTA DE FIGURAS.....	3
LISTA DE TABELAS.....	3
1. APRESENTAÇÃO.....	3
2. OBJETO DE ESTUDO.....	4
3. HIPÓTESES.....	4
4. OBJETIVOS.....	5
4.1 Objetivos Gerais.....	5
4.1 Objetivos Específicos.....	5
5. JUSTIFICATIVA.....	5
6. REFERENCIAL TEÓRICO.....	6
6.1. Computação Gráfica.....	6
6.2. OpenGL.....	8
6.3. Gráficos 3D.....	9
6.4. HTML.....	11
6.5. Javascript.....	11
6.7. WebGL.....	11
? METODOLOGIA.....	12
? PESQUISA.....	12
? CONCLUSÃO.....	12
? REFERÊNCIAS.....	12

## GLOSSÁRIO

## LISTA DE FIGURAS

## LISTA DE TABELAS

### 1. APRESENTAÇÃO

Uma das áreas mais proeminentes da computação dos dias de hoje refere-se a *World Wide Web*. Desenvolvida por Tim Berners-Lee em 1990 (WEBFOUNDATION, c2008-2022), as tecnologias fundamentais para o seu funcionamento consistem na linguagem de formatação *HTML – Hypertext Markup Language* - e no protocolo *HTTP – Hypertext Transfer Protocol* -, que permite o compartilhamento de recursos através da rede.

Além dos tradicionais arquivos de texto, o protocolo *HTTP* permite atualmente o compartilhamento de diversos outros tipos de mídia, como arquivos de estilo (*CSS*), *scripts (Javascript)*, imagens, áudios, modelos 3D, entre outros (MDN, c1998-2023), permitindo assim o desenvolvimento de aplicações muito mais complexas e enriquecendo a experiência dos usuários conectados a *internet*.

Dada a evolução dos equipamentos de *hardware* nos últimos anos, se tornou cada vez mais comum a utilização de aplicações que usam de modelagem 3D para suas necessidades. A constante evolução da computação gráfica para o desenvolvimento de jogos, fotografia digital, design gráfico, cartografia, visualização de dados, entre muitos outros, fez crescer a demanda para que estas tecnologias se tornassem disponíveis também nos navegadores *Web*. Assim, foi desenvolvida o padrão *WebGL*, versão da *API* gráfica *OpenGL*, funcionando nativamente no navegador, permitindo assim o desenvolvimento de gráficos 3D com aceleração de *hardware* em *websites*. (Gdad-s-River, 2017)

Na engenharia de *software*, Parnas (1972, p1053-1058, apud OLIVEIRA, 2017, p15) define a modularidade como a capacidade de dividir um sistema em submódulos, que podem ser modificados individualmente sem informações adicionais das outras.

Assim, é possível que o desenvolvedor de uma aplicação utilize de bibliotecas de códigos fonte fornecidas por terceiros, que ao abstrair tarefas complexas de baixo nível, podem facilitar o desenvolvimento de um programa.

Assim, este trabalho busca fazer um estudo comparativo de diversas bibliotecas feitas em *Javascript* construídas em cima das capacidades dos navegadores em renderizar objetos 3D, fornecendo assim resultados qualitativos e quantitativos acerca de suas capacidades funcionais, com as vantagens e desvantagens que um desenvolvedor terá ao optar por utilizar qualquer uma delas.

## **2. OBJETO DE ESTUDO**

O objeto de estudo deste projeto trata da utilização de bibliotecas, ferramentas e frameworks javascript selecionados que sejam capazes de fornecer uma camada de abstração do *WebGL*, e dessa forma, possam ser utilizadas em projetos *Web* que buscam renderizar objetos de três dimensões em tempo real.

Dessa forma, busca-se catalogar e analisar as diversas opções que existem no mercado de software a fim de procurar as melhores alternativas que um desenvolvedor *Web* possa utilizar no seu projeto.

## **3. HIPÓTESES**

Tendo feito as comparações entre as diferentes ferramentas utilizadas, é esperado obter grandes ou pequenas diferenças entre cada uma de suas métricas, e dessa forma, será possível obter um panorama geral das vantagens e desvantagens que um desenvolvedor terá ao utilizar em cada uma delas.

É esperado que algumas bibliotecas sejam melhores para uso geral, enquanto outras sejam superiores em propósitos específicos. Algumas bibliotecas podem performar melhor quando se trata de uma aplicação menor, com baixos requisitos de funcionalidades, além de ter um usabilidade mais simples, enquanto outras serão mais adaptadas para maiores projetos, de acordo com seu nível de complexidade.

## 4. OBJETIVOS

### 4.1 Objetivos Gerais

Investigar diversas bibliotecas para a linguagem de programação *Javascript* que possam ser utilizadas para o desenvolvimento de aplicações 3D em aplicações Web, e assim realizar uma análise comparativa entre elas utilizando diversas métricas de *software*, ressaltando os pontos positivos e negativos de cada uma.

### 4.1 Objetivos Específicos

A fim de realizar os propósitos definidos pelo objetivo geral, o seguinte trabalho apresenta as seguintes metas:

- Selecionar as bibliotecas *Javascript* que deverão ser utilizadas para a comparação, utilizando de critérios como: popularidade, disponibilidade da documentação e atualização do projeto.
- Fazer uma apresentação do uso de cada biblioteca escolhida, expondo sua origem, seu funcionamento em código fonte e suas funcionalidades.
- Buscar e entender as diversas métricas de software que deverão ser utilizadas para a análise de cada biblioteca analisada.
- Para cada biblioteca avaliada, fazer uma comparação de diversos de seus aspectos qualitativos, tais como: documentação e atualização do projeto.
- Realizar, para cada biblioteca, uma série de testes envolvendo suas funcionalidades, desenvolvendo código fonte com o objetivo de avaliar seus aspectos quantitativos, tais como: performance, uso de memória, funcionalidades presentes e quantidade de código necessária para implementação.
- Fazer uma conclusão geral, resumindo os dados levantados ao longo do projeto, e ressaltando os pontos positivos e negativos descobertos na pesquisa e no uso das bibliotecas analisadas.

## 5. JUSTIFICATIVA

Em todas as áreas da computação, a evolução dos paradigmas é um fenômeno inexorável. A medida que a tecnologia se desenvolve, novas possibilidades antes não

exploradas se tornam cada vez mais comuns no dia a dia dos desenvolvedores e usuários. O desenvolvimento dos navegadores e dos computadores *desktop* permitiu a criação *websites* cada vez mais complexos, com efeitos visuais detalhados, tais como: sombras, luzes, animações complexas, entre muitos outros.

Dessa forma, o uso do 3D, antes conhecido sobretudo nos vídeo games de computador ou consoles, nas programas de edição e modelagem, se tornaram possíveis também de serem executados no navegador, abrindo um novo leque de possibilidades, permitindo designs mais realistas, interações de usuário mais elaboradas, e proporcionando uma experiência mais rica ao usuário.

No entanto, no surgimento de muitas tecnologias, uma dificuldade no seu uso se manifesta na medida em que os desenvolvedores têm de adaptar-se a elas antes de poderem utilizá-la, e assim incorporarem-na no seu conjunto de habilidades. Dado que a computação gráfica é uma área notoriamente complexa, requerindo conhecimentos de matemática e álgebra linear, disciplinas comumente desnecessárias no desenvolvimento da maioria das aplicações Web, a disponibilidade de ferramentas de mais alto nível que auxiliem estes desenvolvedores a produzir gráficos 3D sem exigir deles essas competências técnicas se faz necessária.

Assim, a fim de facilitar aos desenvolvedores a pesquisa e o estudo por estas tecnologias, o seguinte trabalho busca analisá-las e fazer uma comparação detalhada entre elas, permitindo que os programadores que queiram utilizá-las adquiram um conhecimento prévio que os permitam fazer uma melhor decisão na escolha de sua ferramenta.

## 6. REFERENCIAL TEÓRICO

### 6.1. Computação Gráfica

A computação gráfica pode ser definida como:

A Computação Gráfica reúne um **conjunto de técnicas que permitem a geração de imagens** a partir de modelos computacionais de objetos reais, objetos imaginários ou de dados quaisquer coletados por equipamentos na natureza. (SILVEIRA, 2018)

Dessa forma, entendemos que a computação gráfica é uma forma de representar

objetos, que podem ser derivados do mundo real ou imaginários, num computador, assim gerando imagens que serão visualizadas por usuários, os quais podem ter objetivos educacionais, corporativos, científicos, lúdicos, entre outros.

A história da computação gráfica tem origem na década de 1950, época no qual pesquisadores do MIT foram capazes de criar um computador que podia processar imagens em três dimensões. Mais tarde, nos anos 1970, Ivan Sutherland e David Evans foram capazes de desenvolver um software que gerava e modelava gráficos. Posteriormente, foram criados os modelos 3D, utilizados amplamente na indústria cinematográfica, dando vida aos filmes da Disney e da Pixar. Outro grande marco a ser mencionado é o lançamento do computador da Apple, Macintosh, em 1980. (COUTINHO, 2021)

Devido ao alto custo que essa tecnologia demandava para ser utilizada, inicialmente a CG estava limitada às estações gráficas que possuíam recursos computacionais mais potentes, o que era caro. Foi nos anos 1990 que a evolução do hardware e dos dispositivos gráficos permitiu o barateamento desses recursos. Assim, acelerou-se o surgimento de ambientes que utilizavam da interface gráfica como o sistema operacional Windows. (SIQUEIRA, [s.d])

A computação gráfica se divide, principalmente, em três subcategorias (GONÇALVES, c2020-2021):

- Síntese de imagens, que se refere a produção de imagens sintéticas, em duas ou três dimensões;
- Análise de imagens, que busca obter dados provenientes de uma imagem, e traduzi-los em informações úteis para uma determinada função, como, por exemplo, reconhecimento facial.
- Processamento de imagens, o qual busca manipular uma determinada imagem com ajustes de cor, brilho, aplicações de filtros, etc. Exemplos incluem o Photoshop e os programas de edição de vídeo.

Atualmente, a computação gráfica é parte fundamental de diversos sistemas computacionais, e se tornou uma ferramenta essencial na vida e cotidiano de muitas pessoas. Entre os seus usos, incluem-se: interface de usuário de programas e sistemas



operacionais, visualização de gráficos, editoração eletrônica, CADs - Projeto assistido 6 por computador - (mecânico, civil, elétrico), simulações e animações, arte e comércio, cartografia, e muitos outros. (PINHO, [s.d])

## 6.2. OpenGL

Um dos grandes instrumentos utilizados na computação gráfica é o OpenGL, que:

OpenGL, assim como Direct3D ou Glide, é uma API (Application Programming Interface), termo usado para classificar uma biblioteca de funções específicas disponibilizadas para a criação e desenvolvimento de aplicativos em determinadas linguagens de programação. A OpenGL foi produzida com C e C++ em mente, mas pode ser utilizada para diversas outras com um alto nível de eficiência. (TECMUNDO, 2008)

Assim, entendemos que o OpenGL é uma API, uma especificação de funções escritas na linguagem de programação C, que deve ser implementada pelos fabricantes de placas de vídeo, criada para ser uma interface de programação para o desenvolvimento de aplicações gráficas tais como jogos, CADs, sistemas operacionais, programas de desenho e modelagem 3D, bibliotecas gráficas de alto nível, entre outros.

A história do OpenGL remonta a década de 1980, época no qual não havia uma padronização comum das interfaces gráficas, fazendo com que os desenvolvedores do período tivessem de produzir seus softwares para cada hardware diferente. Dessa forma, uma empresa de nome Silicon Graphics Inc. (SGI) criou o IRIS GL, um padrão gráfico que chamou a atenção da indústria. Mais tarde, devido ao interesse da empresa em manter parte do código fonte como proprietário, a ramificação de código totalmente aberto, o OpenGL, surgiu, estando em contínuo desenvolvimento até hoje, sendo suportado por diversas empresas, como a AMD, Dell, HP, Intel, Nvídia, Sun (Oracle) e outras. (SILVA, 2012)

O OpenGL é independente da implementação de uma placa de vídeo específica, assim, o programador apenas deve dizer apenas o que o computador deve fazer, e não como. Possui 250 funções, utilizadas para controlar os desenhos que serão realizados na tela do computador. Tem suporte a diversas primitivas, como pontos, linhas e

triângulos. A *rendering* pipeline é a palavra que se utiliza para descrever o processo que acontece quando os comandos vão sendo passados para a API no *buffer* de comandos, nos quais, são adicionadas coordenadas, cores, texturas, iluminação e outros efeitos, até chegar no frame buffer, do qual é enviado para a tela do computador. (MANSSOUR, 2003)

### 6.3. Gráficos 3D

Uma tela de computador é composta por milhares de pixels, pequenos quadrados constituídos de três feixes de luz, um de cada cor, vermelho, verde e azul, um padrão conhecido como RGB - Red Green Blue -. Assim, num processo complexo de transformações, é possível determinar a luminosidade de cada feixe de luz para cada pixel, o qual misturando as cores aditivamente, pode formar milhões de cores diferentes, permitindo a criação de imagens complexas e realistas. (MEIRELLES, 2002)

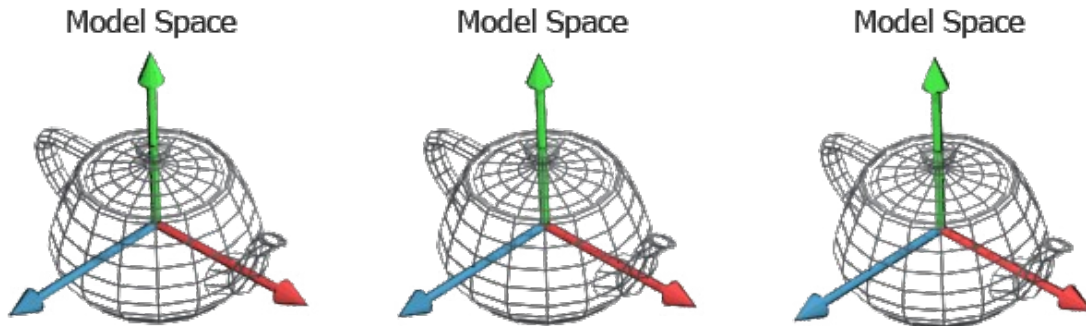
Para determinar o local nos quais os pixels serão desenhados, um sistema de coordenadas é necessário. No caso de gráficos 3D, o *OpenGL* um sistema que damos o nome de *right handed*, no qual as coordenadas começam no quanto inferior esquerdo, apontando para cima e para a direita, e a terceira dimensão, o Z, aponta para a frente da tela. (VRIES, 2020)

Os objetos tridimensionais são formados por polígonos – usualmente triângulos -, que, por sua vez, compõem os dados de vértices, os quais além de guardar coordenadas no espaço tridimensional, também podem conter texturas, cores e o modo de reflexão de luz. Esses vértices são então enviados a *GPU* para serem processados por programas chamados *shaders*, responsáveis por transformar a representação 3D de um objeto numa imagem que a tela seja capaz de exibir. (GREGORY, c2015).

O *vertex shader* é o programa da *GPU* responsável por realizar estas transformações. Primeiro, o objeto, chamado de *model*, é transformado no seu espaço local, no qual são aplicados a rotação e escalonamento. Depois o *view space* determina a posição do objeto em relação a cena e ao restante dos objetos. Por fim, a *projection matrix* da câmera é responsável por fornecer uma visão do mundo a partir de uma localização e direção, criando a imagem 2D que se vê na tela do computador.

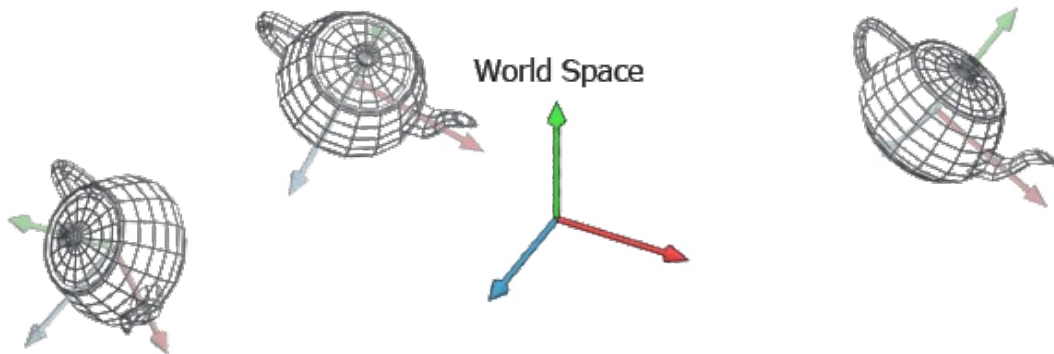
(MÖLLER et al, 2018)

Figura 1: Um modelo de um objeto 3D: visão local



Fonte: Codinglabs<sup>1</sup>.

Figura 2: Um modelo de um objeto 3D: visão na cena



Fonte: Codinglabs<sup>2</sup>.

Após o *vertex shader* lidar com as posições dos polígonos, inicia-se a montagem da forma e da geometria, e depois, o processo de rasterização, no qual os espaços entre os triângulos são determinados para serem coloridos, descartando os pixels que não estiverem entre os limites da tela. Ainda, o *fragment shader* é responsável por aplicar a cor ao objeto, utilizando-se de cores, texturas e fontes de luz. Por fim, é feito aos testes de profundidade, no qual pode-se descartar os objetos obstruídos no

1 Disponível em: [http://www.codinglabs.net/article\\_world\\_view\\_projection\\_matrix.aspx](http://www.codinglabs.net/article_world_view_projection_matrix.aspx). Acesso em 24 abr. 2023

2 Disponível em: [http://www.codinglabs.net/article\\_world\\_view\\_projection\\_matrix.aspx](http://www.codinglabs.net/article_world_view_projection_matrix.aspx). Acesso em 24 abr. 2023

plano de fundo, além da aplicação de efeitos de transparência. (VRIES, 2020)

#### 6.4. HTML

#### 6.5. Javascript

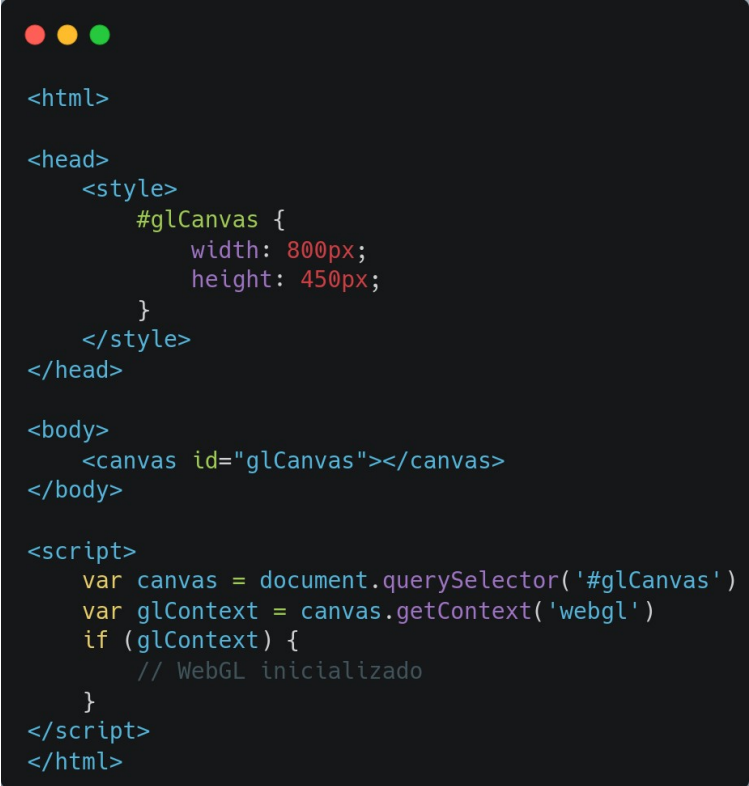
#### 6.7. WebGL

*WebGL* é uma *API* para *javascript* que tem o propósito de criar gráficos de duas ou três dimensões em alta performance em qualquer *browser* compatível sem a necessidade de utilizar *plug-ins* de terceiros. O *WebGL* realiza este feito ao implementar uma *API* quase totalmente compatível com o *OpenGL ES*, uma versão criada especificamente para rodar em dispositivos embarcados e *mobile*. (PETERBE et al, 2023)

Teve origem no ano de 2009, quando uma organização nomeada *Khronous group* iniciou uma equipe para o projeto, no qual estavam presentes representantes da *Apple*, *Google*, *Mozilla* e *Opera*. O seu propósito era suportar gráficos 3D na *Web*. Criada por Vladimir Vukićević, escolheu o *OpenGL* como *API* pela quantidade de desenvolvedores familiares com a tecnologia. Apesar do nome, a *WebGL* nem sempre utiliza o *OpenGL* como motor subjacente, podendo utilizar o *DirectX* no sistema operacional *Windows*. (MANOR, 2021)

A fim de desenvolver uma aplicação utilizando o *WebGL*, o desenvolvedor deverá utilizar a tag `<canvas>` no código *HTML*, e então, utilizar o *javascript* e criar um contexto *WebGL* neste elemento, habilitando assim o uso das funções comuns do *OpenGL* que serão renderizadas no espaço delimitado pelo *canvas*. Abaixo segue um exemplo de inicialização do *WebGL* num navegador.

Figura 3: Código HTML: inicializando WebGL no elemento canvas



```
<html>

<head>
  <style>
    #glCanvas {
      width: 800px;
      height: 450px;
    }
  </style>
</head>

<body>
  <canvas id="glCanvas"></canvas>
</body>

<script>
  var canvas = document.querySelector('#glCanvas')
  var glContext = canvas.getContext('webgl')
  if (glContext) {
    // WebGL inicializado
  }
</script>
</html>
```

Fonte: O Autor, 2023.

## ? . METODOLOGIA

## ? . PESQUISA

## ? . CONCLUSÃO

## ? . REFERÊNCIAS

Web Foundation. **History of the Web**. Web Foundation, c2008-2022. Disponível em: <https://webfoundation.org/about/vision/history-of-the-web/>. Acesso em: 16 mar. 2023.

Mfujio9 etc al. **MIME Types (IANA media types)**. MDN Web Docs, c1998-2023. Disponível em: [https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics\\_of\\_HTTP/MIME\\_types](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types). Acesso em: 16 mar. 2023.

Gdad-s-River. A **Brief History of Web Graphics**. Fossbytes, 2017, Disponível em: <https://fossbytes.com/history-web-graphics/>. Acesso em: 16 mar. 2023.

PARNAS, D. L. **On the criteria to be used in decomposing systems into modules**. Communications of the ACM, ACM, v. 15, 1972.

SILVEIRA, André. **O que é Computação Gráfica**. Ambiente DESIGN, 2018. Disponível em: <http://www.um.pro.br/index.php?c=/computacao/definicao>. Acesso em: 25 abr. 2023.

COUTINHO, Thiago. **O que é Computação Gráfica? Descubra as melhores oportunidades na área!**. Voitto, 2021. Disponível em: <https://www.voitto.com.br/blog/artigo/o-que-e-computacao-grafica>. Acesso em: 25 abr. 2023.

SIQUEIRA, Fernando. **Conceitos de Computação Gráfica**. Sites Google, [s.d]. Disponível em: <https://sites.google.com/site/profferdesiqueiracompgrafica/aulas/aula-1---conceitos-decomputacao-grafica>. Acesso em: 25 abr. 2023.

GONÇALVES, Júnior. **Introdução a computação gráfica**. HyperBytes, c2020-2021. Disponível em: <https://www.hiperbytes.com.br/introducao-a-computacao-grafica>. Acesso em: 25 abr. 2023.

PINHO, Márcio. **Origens da Computação Gráfica**. Pucrs, Escola Politécnica, [s.d]. Disponível em: <https://www.inf.pucrs.br/~pinho/CG/Aulas/Intro/introOLD.htm>. Acesso em: 25 abr. 2023.

Equipe NZN. **O que é OpenGL**. Tecmundo, 2008. Disponível em: <https://www.tecmundo.com.br/video-game-e-jogos/872-o-que-e-opengl-.htm>. Acesso em: 25 abr. 2023.

SILVA, Edson. **Opengl completa três décadas de constante evolução**. Diolinux, 2012. Disponível em: <https://diolinux.com.br/noticias/opengl-completa-tres-decadas.html>. Acesso em: 25 abr. 2023.

MANSSOUR, Isabel. **Introdução à OpenGL**. Pucrs, Escola Politécnica, 2003. Disponível em: <https://www.inf.pucrs.br/~manssour/OpenGL/Introducao.html>. Acesso em: 25 abr. 2023.

MEIRELLES, Adriano. **Como funciona o LCD**. Hardware, 2002. Disponível em: <https://www.hardware.com.br/livros/hardware-manual/como-funciona-lcd.html>. Acesso em: 26 abr. 2023.

VRIES, Joey. **Learn OPENGL – Graphics Programming**, 2020. Disponível em: [https://learnopengl.com/book/book\\_pdf.pdf](https://learnopengl.com/book/book_pdf.pdf). Acesso em: 26 abr. 2023.

GREGORY, Jason. **Game Engine Architecture**. 2. ed. Boca Raton: Taylor & Francis Group, c2015. Disponível em: <http://ce.eng.usc.ac.ir/files/1511334027376.pdf>. Acesso em: 26 abr. 2023

MÖLLER et all. **Real Time Rendering**. 4. ed. Boca Raton: Taylor & Francis Group, 2018. Disponível em: <http://cinslab.com/wp-content/uploads/2021/03/chenjiahao-Real-Time-Rendering-Fourth-Edition-2018-CRC-Press.pdf>. Acesso em: 26 abr. 2023.

PETERBE et all. **WebGL: 2D and 3D graphics for the web**, 2023. Disponível em: [https://developer.mozilla.org/en-US/docs/Web/API/WebGL\\_API](https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API). Acesso em 26 abr. 2023.

MANOR. **The story of WebGPU — The successor to WebGL**, 2021. Disponível em: <https://eytanmanor.medium.com/the-story-of-webgpu-the-successor-to-webgl-bf5f74bc036a>. Acesso em 26 abr. 2023.