UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE – UERN

FACULDADE DE CIÊNCIAS EXATAS E NATURAIS – FANAT

DEPARTAMENTO DE INFORMÁTICA – DI

KERGIONALDO PIRES FERNANDES SOARES

A BIBLIOTECA P5.JS COMO ALTERNATIVA DE APRENDIZAGEM EM COMPUTAÇÃO GRÁFICA

MOSSORÓ - RN

2019

KERGIONALDO PIRES FERNANDES SOARES

A BIBLIOTECA P5.JS COMO ALTERNATIVA DE APRENDIZAGEM EM COMPUTAÇÃO GRÁFICA

Monografia apresentada à Universidade do Estado do Rio Grande do Norte como um dos pré-requisitos para obtenção do grau de bacharel em Ciência da Computação, sob orientação do Prof. Dr. Carlos Heitor Pereira Liberalino.

MOSSORÓ - RN

2019

Catalogação da Publicação na Fonte.

Universidade do Estado do Rio Grande do Norte.

S676b Soares, Kergionaldo Pires Fernandes
A BIBLIOTECA P5.JS COMO ALTERNATIVA DE APRENDIZAGEM EM COMPUTAÇÃO GRÁFICA. / Kergionaldo Pires Fernandes Soares. - Mossoró, 2019.
68p.

Orientador(a): Prof. Dr. Carlos Heitor Pereira Liberalino.
Monografia (Graduação em Ciência da Computação). Universidade do Estado do Rio Grande do Norte.

1. Computação Gráfica. 2. OpenGL. 3. P5.js. 4. Aprendizagem. I. Liberalino, Carlos Heitor Pereira. II. Universidade do Estado do Rio Grande do Norte. III. Título.

O serviço de Geração Automática de Ficha CatalográNca para Trabalhos de Conclusão de Curso (TCC´s) foi desenvolvido pela Diretoria de Informatização (DINF), sob orientação dos bibliotecários do SIB-UERN, para ser adaptado às necessidades da comunidade acadêmica UERN.

KERGIONALDO PIRES FERNANDES SOARES

A biblioteca P5.js como proposta de aprendizagem em Computação Gráfica

Monografia apresentada como pré-requisito para a obtenção do título de Bacharel em Ciência da Computação da Universidade do Estado do Rio Grande do Norte – UERN, submetida à aprovação da banca examinadora composta pelos seguintes membros:

Aprovada em: 25/09/2019

Banca Examinadora

Prof. Dr. Carlos Heitor Pereira Liberalino
Universidade do Estado do Rio Grande do Norte - UERN

Prof. Dr. Francisco Chagas de Lima Júnior
Universidade do Estado do Rio Grande do Norte - UERN

Prof. Dr. Harold Ivan Angulo Bustos
Universidade do Estado do Rio Grande do Norte - UERN

Aos meus pais, Eneide e Kerginaldo,
meus irmãos, familiares e amigos

AGRADECIMENTOS

À inteligência suprema e criadora de todo o Cosmos, por me conceder a breve existência nesse espaço-tempo e a oportunidade de fazer coisas grandiosas.

Aos meus pais, Eneide e Kerginaldo, por serem meus dois pilares da vida e me darem todo o apoio e as condições de estar onde cheguei até esse ponto em minha jornada.

Aos meus irmãos, Kergivanaldo, Kergivaneide e Kéllya, por me acompanharem e apoiar, ajudando e incentivando meu progresso. Aos demais familiares pela boa convivência e apoio.

Aos meus grandes amigos, colegas e parceiros, por contribuírem com pequenas e grandes ações que foram decisivas até esse ponto de minha trajetória. Em especial, do antigo grupo Twelve Brothers dos tempos de IFRN (Alfredo, Aléx, Allan Kardec, Daniel, Israel, Ricardo, Isaias, Jório (in memorian), Judson, Hugo (in memorian), Paulo, Sadrak, Samuel), meus colegas de curso com quem tive boa convivência (Alfredo, Hítalo, Sadrak, Luiz, Victor, Exlley, Fernando, Yakamuri, Lígia, Elisa, Melissa, e os demais que tive grande afinidade durante a graduação), meus companheiros de ônibus e meus amigos de infância.

Aos meus professores, por me incentivaram e terem dado bons ensinamentos e visões de carreira, além de terem sido excelentes docentes e passarem conselhos e reflexões durante as conversas dentro e fora da sala de aula, em especial os professores do curso de Informática do IFRN Campus Apodi (Cleone, Carlos Fran, Ronnison, Galba, Rodrigo, “Chicão”, Igor e Felipe) e os professores do curso de Ciência da Computação da UERN Campus Central (Heitor [orientador do TCC], Sebastião [orientador da monitoria], Lima Jr., Rommel, Alysson, Cicília, Maximiliano, Isaac, André Pedro, Pedro Fernandes, Harold e os demais professores do Departamento de Informática do Campus Central da UERN.

Aos técnicos e secretários do Departamento de Informática, pelo apoio e ajuda no dia a dia na universidade: Mizael, Jetro, Elis e Neto.

“A imaginação é mais importante que o conhecimento”

– Albert Einstein

RESUMO

As tecnologias com foco em facilitar processos já existentes estão cada vez mais sendo novidades e grande parte vêm para facilitar os processos de aprendizagem em uma variedade de áreas do conhecimento, em especial nos campos científico e tecnológico. Uma das áreas que vem sendo de bastante importância nas últimas décadas é a Computação Gráfica, área de Ciência da Computação responsável pelas definições de imagens digitais e as técnicas e tecnologias envolvidas no contexto gráfico. Uma das principais ferramentas utilizadas na produção de imagens digitais em computadores é conhecida como OpenGL, ferramenta amplamente utilizada na indústria tecnológica e, por ser altamente eficiente e aberta, é o padrão na construção de aplicações gráficas de alta qualidade. Mas a ferramenta ainda é um grande desafio para iniciantes que são introduzidos na teoria e prática em Computação Gráfica, por motivos de sua complexidade e outros aspectos. Esse trabalho tem como objetivo apresentar a proposta da ferramenta Web P5.js como alternativa à OpenGL no contexto da disciplina de Computação Gráfica de maneira a facilitar aprendizagem com a aplicação prática simplificada da teoria geral.

Palavras-chave: Computação Gráfica, OpenGL, P5.js, Aprendizagem.

ABSTRACT

The technologies with focus on to facilitate the processes already existing are increasingly being news and most of them comes to facilitate the learning processes in a variety of knowledge areas, in special in the scientific and technological fields. One of the areas that has have been very importance in the last decades is the Computer Graphics, as Computer Science field that is responsible for the digital images definitions and the involved techniques and technologies in graphic context. One of the main technologies used in the digital images production in computers is known as OpenGL, widely used tool in the technology industry and, because of being highly efficient and open, is the standard in high quality graphic applications building. However the tool already is a big challenge for beginners who are introduced to theory and practice in Computer Graphics, for the reasons of its complexity and other aspects. This work aims to present the proposal of the P5.js tool as alternative to OpenGL in the introductory approach of the Computer Graphics discipline in a way to facilitate the learning with the simplified practical application of general theory.

Keywords: Computer Graphics, OpenGL, P5.js, Learning.

LISTA DE SIGLAS

API	–	*Application Programming Interface*
CG	–	Computação Gráfica
WWW	–	*World Wide Web*
HTTP	–	*HyperText Transfer Protocolo*
HTML	–	*HyperText Markup Language*
CERN	–	*European Center for Nuclear Physics*
CAD	–	*Computer Aided Design*
RGB	–	*Red, Green, Blue*
IDE	–	*Integrated Development Environment*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE CÓDIGOS

SUMÁRIO

1 INTRODUÇÃO

Computação Gráfica é uma área bastante importante em Ciência da Computação, e o seu desenvolvimento ao longo das últimas décadas fez essa área começar a ser aplicada em vários setores científicos, industriais etc. Seu desenvolvimento desde então trouxe à tona tecnologias de tamanha importância nas quais são atribuídas características de qualidade e desempenho na busca por resultados eficientes. As tecnologias dessa área vem sendo fortemente utilizadas, visto seu alto desempenho.

OpenGL é uma tecnologia da área da Computação Gráfica que permanece padrão até os dias atuais, estando disponível nas várias plataformas com o intuito de trabalhar a geração e a manipulação de gráficos digitais. Essa tecnologia é a base da construção de vários tipos de softwares e é adotada pelas grandes empresas da indústria de tecnologias de hardwares pela sua eficiência em permitir a comunicação com o sistema de hardware para gerar gráficos de alta performance.

A ferramenta OpenGL, ao permitir a construção de aplicações gráficas, impulsionou o desenvolvimento de vários sistemas de representação para projetos de engenharia, aperfeiçoamento de técnicas na medicina e a geração de modelos e simulações para o ramo científico. Também teve papel importante no entretenimento, possibilitando o surgimento de softwares para os efeitos especiais de produções do cinema e da TV, além de expandir a produção de videogames.

OpenGL é também a abordagem mais utilizada entre os mais importantes livros que introduzem a Computação Gráfica. OpenGL é utilizada nesses livros como ferramenta de implementação dos conceitos de CG na prática, na criação de programas que geram os gráficos.

Há várias alternativas com propostas similares ao OpenGL. Destacam-se: Microsoft DirectX, SDL (Simple DirectMedia Layer), Allegro etc.

Por motivos de OpenGL ser uma tecnologia complexa, na maioria das vezes, oferece certa dificuldade para iniciantes que começam a aplicar o estudo de

Computação Gráfica em implementações funcionais com a ferramenta. Dessa forma, esse trabalho tem como proposta a abordagem de uma tecnologia alternativa de maior simplicidade à ferramenta padrão para facilitar a aprendizagem na programação gráfica. A alternativa é a biblioteca P5.js, que é implementada com a linguagem de programação JavaScript, uma das linguagens mais utilizadas da atualidade.

Este trabalho não tem como proposta a substituição da ferramenta OpenGL como tecnologia padrão na programação de aplicações gráficas pela ferramenta P5.js, mas tende a demonstrar a utilização dessa última na introdução da disciplina como maneira mais simplificada, voltada para uma maior facilidade de aprendizagem de para os iniciantes em CG.

Como metodologia esse trabalho apresenta e contextualiza as tecnologias OpenGL e P5.js, com o objetivo fazer análises das suas funcionalidades e produzir comparações entre as duas ferramentas na construção de códigos. As análises e as comparações feitas entre as duas tecnologias representam os métodos que apoiam a proposta da biblioteca P5.js como alternativa na aprendizagem.

A estrutura desde trabalho segue a abordagem conceitual de Computação Gráfica, com as definições de conceitos importantes. Então o trabalho leva às definições das tecnologias que fornecem o plano de fundo para a funcionalidade das tecnologias em foco (OpenGL e P5.js). Em seguida são feitas análises da utilização de cada uma das ferramentas na geração de imagens e, na continuação, são feitas comparações entre as duas ferramentas com objetivo de demonstrar como a biblioteca P5.js é uma excelente alternativa para a iniciação na aprendizagem em Computação Gráfica.

2 COMPUTAÇÃO GRÁFICA

2.1 DEFINIÇÃO

O termo Computação Gráfica (CG) é utilizado para referenciar todas as operações de síntese, processamento e análise de gráficos e imagens na forma de dados digitais em sistemas computacionais.

“Computação Gráfica é a área da Ciência da Computação que estuda a geração, manipulação e análise de imagens, através do computador” (MANSSOUR e COHEN, 2006).

CG traz para o mundo computacional a representação de gráficos, resultada das aplicações matemáticas da geometria, trigonometria e outras disciplinas, que dispõem de modelos de representação visual, para a síntese de imagens digitais ou a representação digital de imagens de outras fontes, como a fotografia. “A Computação Gráfica é matemática e arte” (AZEVEDO e CONCI, 2003).

Desde que foi introduzida à Ciência da Computação veio sendo útil a diversas áreas, como as engenharias, a medicina, a arquitetura e às produções artísticas e de entretenimento.

2.2 HISTÓRICO

Segundo Azevedo e Conci (2003): “Parece existir um consenso entre os pesquisadores de que o primeiro computador a possuir recursos gráficos de visualização de dados numéricos foi o *Whirlwind I*, desenvolvido pelo MIT”. Os autores continuam afirmando que o computador foi desenvolvido em 1950 para finalidades acadêmicas e militares.

“Em 1959, surgiu o termo *Computer Graphics*, criado por Verne Hudson, enquanto ele coordenava um projeto para a Boing de simulação de fatores humanos em aviões” (AZEVEDO e CONCI, 2003).

O passar dos anos trouxe forte pesquisa e desenvolvimento de técnicas e tecnologias que evoluíram a Computação Gráfica, a qual seguiu acompanhando a evolução do poder de processamento dos computadores.

Os resultados da evolução dessa área trouxeram grandes inovações para vários ramos: na área da engenharia surgiu as ferramentas CAD (*Computer Aided Design*); na área artística surgiram ferramentas para desenho diretamente no computador; na área do entretenimento começaram a ser criados jogos de computador e consoles de videogames; no cinema os filmes começaram a utilizar gráficos digitais para integrar as cenas; e em diversas áreas da Ciência, pesquisadores começaram a utilizar visualização gráfica para simulações em experiências.

Nos dias atuais a CG tem aplicação indispensável em vários segmentos tecnológicos.

2.3 CONCEITOS MATEMÁTICOS

Os conceitos iniciais em Computação Gráfica utilizam as bases matemáticas da geometria, com as definições de plano e espaço, pontos, retas, e demais primitivas geométricas como representações gráficas. Também se utiliza das representações matriciais das operações com as matrizes.

Dentre os diversos conceitos matemáticos que são utilizados na CG, podem-se destacar três dos fundamentais, que são: Sistemas de coordenadas, primitivas e representação de imagens.

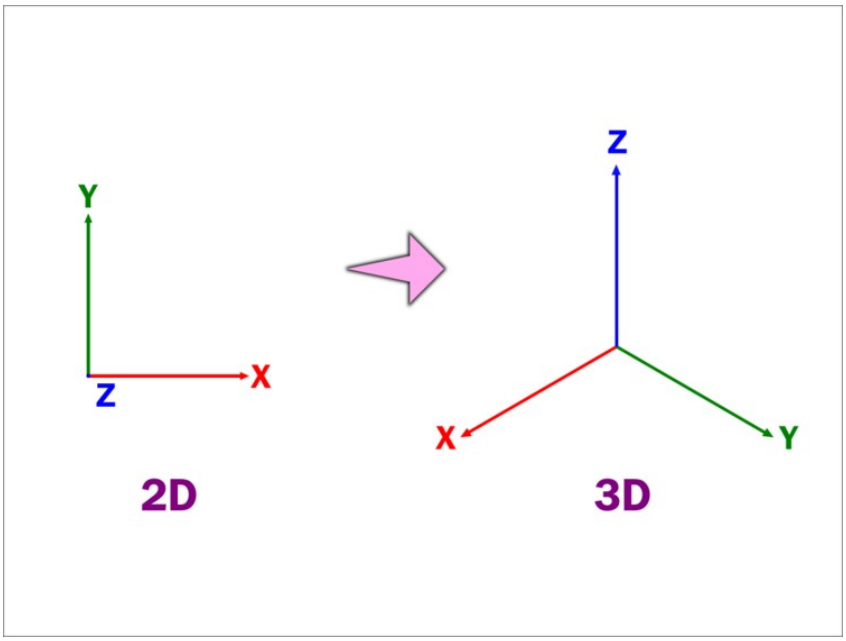
2.3.1 SISTEMAS DE COORDENADAS

São sistemas de representação de formas em suas propriedades e características visuais. Há dois sistemas de representação que são definidos pela geometria euclidiana para representar formas nas dimensões com representações possíveis:

- Plano: representa o sistema de coordenadas em duas dimensões (2D), onde as posições são definidas por dois valores nos eixos (X, Y).
- Espaço: representa o sistema de coordenadas em três dimensões (3D), onde as posições são definidas por três valores nos eixos (X, Y e Z).

A figura 1 fornece uma visualização dos dois sistemas:

Figura 1: Sistemas de coordenadas 2D e 3D



Fonte: https://www.hardware.com.br/tutoriais/rudimentos-3d/

2.3.2 PRIMITIVAS GEOMÉTRICAS

As primitivas geométricas são formas que apresentam medidas com extensões dimensionais, sendo as mais básicas o ponto e a linha. O ponto (ou vértice) é, basicamente a primitiva que dá origem às outras, sendo uma linha, como exemplo, formada por dois vértices.

Primitivas 0D e 1D:

- Ponto (ou vértice): é a primitiva geométrica definida pelo conjunto de duas coordenadas (X, Y) para o plano e três coordenadas (X, Y, Z) para o espaço, como não apresenta valor de medida, é considerado na dimensão 0.
- Reta: é formada por dois vértices e, pelo motivo de ter como valor apenas o comprimento, que é a distância entre os dois pontos, representa a dimensão 1.

Exemplos de primitivas no plano (2D – medidas de largura e altura):

- Triângulo: três vértices;

- Retângulo: quatro vértices;
- Polígonos: vários vértices.

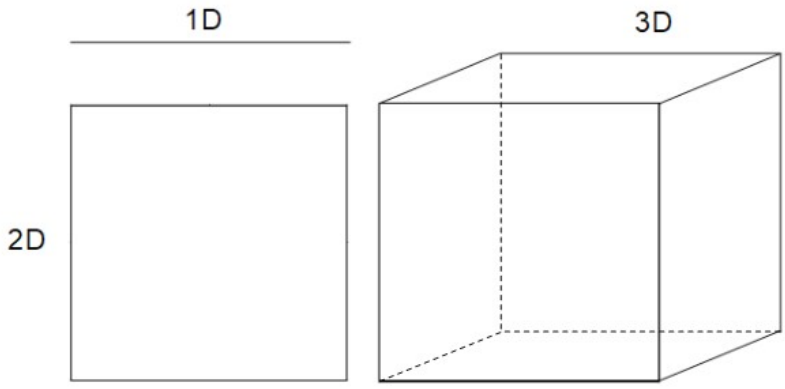
Exemplos de primitivas no espaço (3D – medidas de largura, altura e profundidade):

- Pirâmide: quatro vértices (base triangular) ou cinco vértices (base quadrada);
- Cubo e paralelepípedo: 8 vértices;

Outras primitivas tridimensionais podem utilizar número variado de vértices, como esfera, torus etc.

Na figura 2 são visualizadas três primitivas em suas dimensões:

Figura 2: Objetos de 1, 2 ou 3 dimensões



Fonte: autoria própria

2.3.3 REPRESENTAÇÃO DE IMAGENS

- Vetorial: é a representação ideal, mostrando a imagem baseada no sistema de coordenadas, como o plano cartesiano.
- Matricial: é a representação adaptada na tela, utilizando um conjunto limitado de posições para a visualização.

2.4 CONCEITOS COMPUTACIONAIS

Passando da Matemática para o computador, tais primitivas geométricas e qualquer tipo de imagem são representadas como dados matriciais e renderizadas em uma tela.

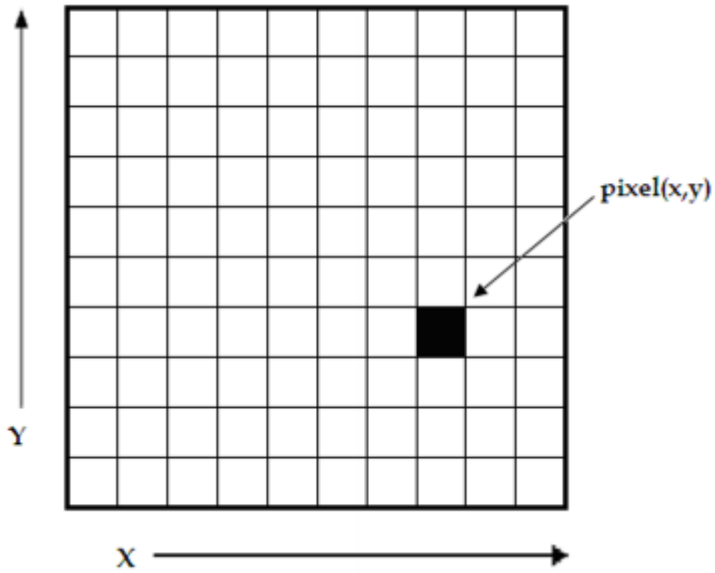
2.4.1 IMAGEM DIGITAL

Todas as imagens que são mostradas na tela de um computador, ou qualquer dispositivo digital, são representadas com esquemas de bits na memória. Esses esquemas de bits dão configurações específicas para que a tela possa apresentar a imagem.

Toda tela é formada por uma grande quantidade de dispositivos luminosos chamados pixels, dispostos horizontal e verticalmente, definindo a tela como uma matriz desses dispositivos. O total de colunas (pixels horizontais) da matriz definem a largura e o total de linhas (pixels verticais) definem a altura da tela. A quantidade de pixels para cada direção define a resolução da tela (largura x altura).

A figura 3 mostra como uma tela é representada na forma de uma matriz de pixels, estes assumindo posições (X, Y):

Figura 3: Matriz da tela e pixel



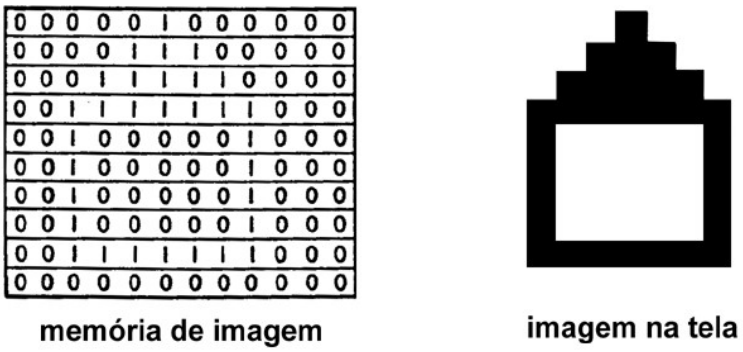
Fonte: https://www.researchgate.net/figure/A-digital-image-is-a-2D-array-of-pixels-Each-pixel-is-characterised-by-its-x-y_fig1_221918148

2.4.2 PIXEL

Pixel (Picture Element - “Elemento de Imagem”, em tradução livre) é a menor parte para a representação de uma imagem na tela. Os esquemas de bits das imagens digitais são utilizados para configurar cada pixel na tela.

Na figura 4 são mostradas a representação de uma imagem em dados digitais (bits) em matriz e sua apresentação na tela:

Figura 4: Dados matriciais e resultado de imagem na tela



Fonte: Azevedo e Conci (2003)

A figura 4 demonstra que os valores binários (bits) 0 e 1 configuram cada pixel para mostrar a cor branco e preto, respectivamente.

2.4.3 SISTEMA DE CORES

Para a coloração das imagens em uma tela cada pixel possui a configuração para as três cores primárias, que também são utilizadas para formar as outras cores. Assim os pixels representam o sistema RGB (*Red*, *Green*, *Blue* – *Vermelho, Verde, Azul*) como forma de gerar todas as cores vistas em uma tela.

Para cada uma das cores primárias são utilizados 8 bits, que configuram a intensidade da luz na faixa que vai da combinação binária de 00000000 a 11111111, que em decimal representam a faixa (0 – 255). Assim, para cada cor há 256 possíveis intensidades. A cor que se vê em cada pixel é resultante de combinações das intensidades das cores primárias.

Nas imagens em preto e branco, que segue a escala de cinza, cada pixel tem os três valores RGB configurados iguais e, assumindo a coloração da intensidade mínima (preto) à máxima (branco), estão na faixa (0 – 255), para as três cores.

As demais combinações dos valores RGB são utilizadas para dar uma cor resultante das misturas de vermelho, verde e azul a cada pixel.

Portanto, com todas as combinações de (0, 0, 0) a (255, 255, 255), um pixel um total de 16.777.216 de cores possíveis para representar. A Tabela 1 demonstra como são definidos os valores para cada cor:

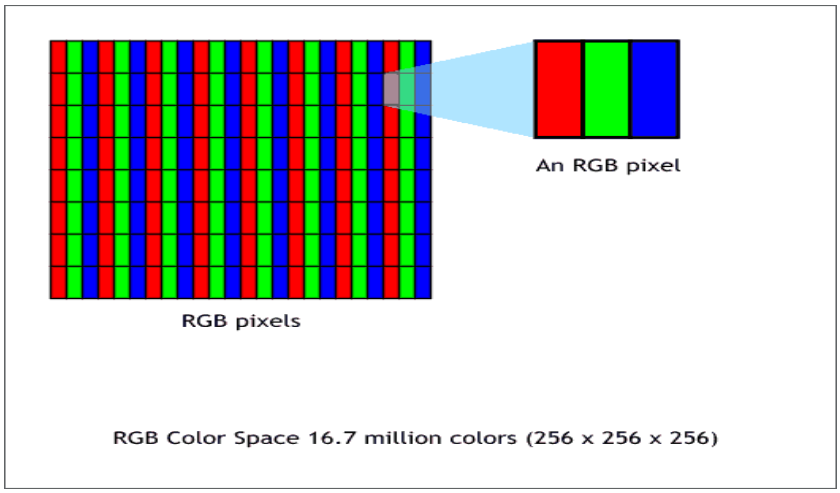
Tabela 1: Cores RGB

Cor	Configuração (R, G, B)	Total de configurações
VERMELHO	(0 – 255, 0, 0)	256 TONS
VERDE	(0, 0 – 255, 0)	256 TONS
AZUL	(0, 0, 0 - 255)	256 TONS

Fonte: autoria própria

A figura 5 mostra como é vista uma tela configurada na cor branca (255, 255, 255), que é o valor total em RGB. Assim, branco é a cor resultante observada da mistura entre as cores vermelho, verde e azul:

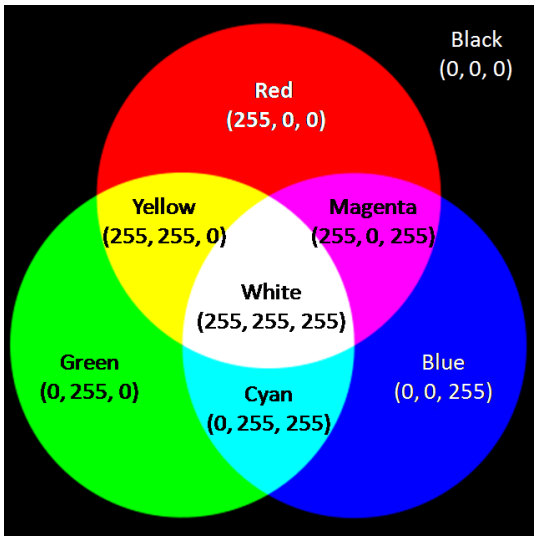
Figura 5: Pixels RGB compondo a tela



Fonte: http://archive.xaraxone.com/webxealot/workbook35/page_5.htm

As cores resultantes são alcançadas com a configuração dos valores RGB. A figura a seguir mostra o modelo de junções:

Figura 6: Cores RGB e o esquema de mistura

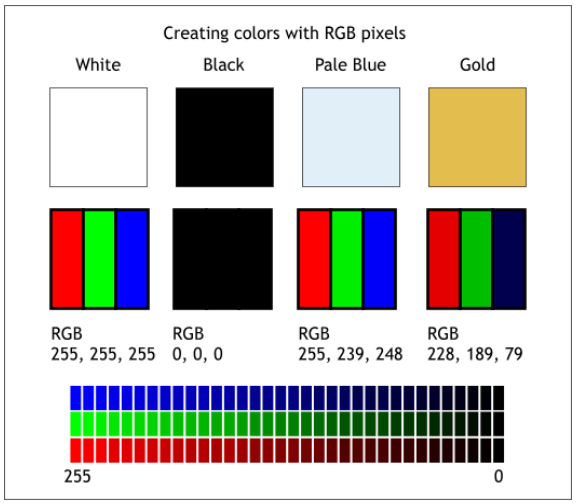


Fonte: https://cs.calvin.edu/activities/connect/CompRenew/03programming/01programming.html

A figura 6 mostra quais configurações de valores são definidas para mostrar uma cor a partir da mistura das três principais. Dessa forma, quando os três valores são (0, 0, 0) a cor será preto; quando são (255, 255, 255) a cor será branco; (255, 0, 0), vermelho; (0, 255, 0), verde; (0, 0, 255), azul; (255, 255, 0), amarelo; (0, 255, 255), ciano; (255, 0, 255), roxo.

A figura 7 mostra alguns exemplos de configurações e as cores resultantes:

Figura 7: Exemplos de configuração de cores



Fonte: http://archive.xaraxone.com/webxealot/workbook35/page_5.htm

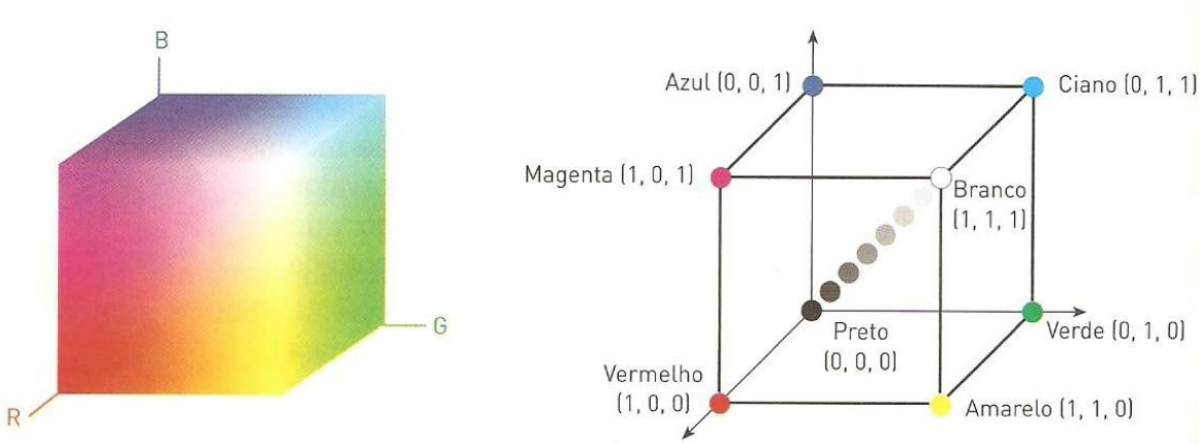
Modelos de cores

Configurar os valores para dar cor aos pixels pode apresentar certa dificuldade quando se precisa encontrar cores específicas apenas alterando os valores RGB, já que há mais de 16 milhões de possibilidades de configurações. Para facilitar a visualização na utilização de cores foram criados alguns modelos que simplificam a escolha de cores pretendidas. Alguns modelos de cor são apresentados a seguir.

Modelo RGB

Esse modelo determina um espaço com os três eixos (X, Y e Z), nos quais cada um representa uma cor RGB, formando um cubo com as possibilidades de mistura em que cada coordenada representa uma configuração, uma cor específica. A Figura 8 demonstra como é o modelo RGB:

Figura 8: Modelo RGB

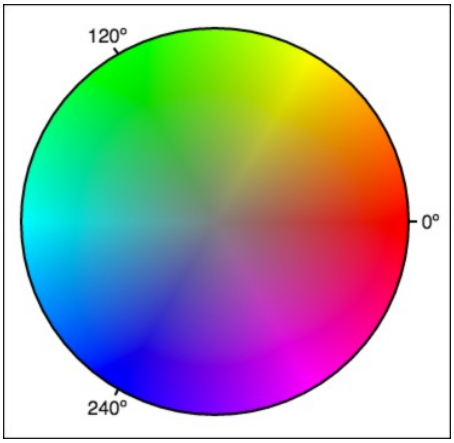


Fonte: https://sites.google.com/site/aimcjv/modelo-rgb

HSV

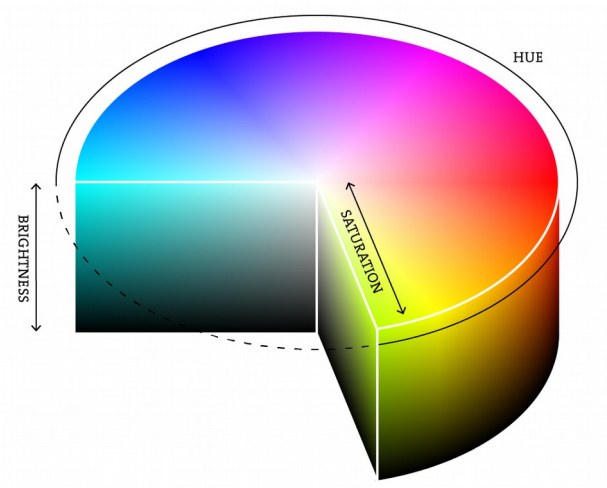
As siglas são das palavras Hue, Saturation e Value (Tonalidade, Saturação e Valor): tonalidade representada em ângulos (0º – 360); saturação em porcentagem (0% - 100%) e valor também em porcentagem. As figuras 9 e 10 dão visualização de como se dá o esse modelo:

Figura 9: Círculo de cores HSV



Fonte: https://www.khanacademy.org/partner-content/pixar/color/color-space/e/hsl-space

Figura 10: Cilindro HSV



Fonte: https://p5js.org/learn/color.html

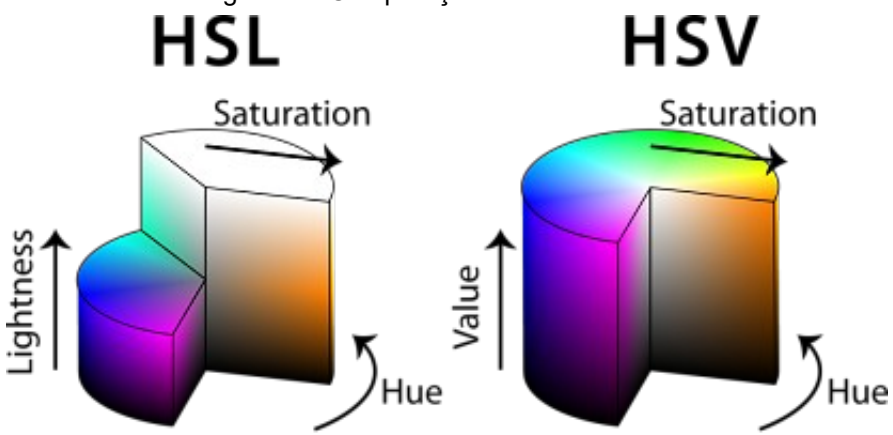
Na Figura 10: a cor (Hue) é dada em graus nas extremidades em volta de todo o cilindro; a saturação (Saturation) dá a pureza da cor e vai do centro do cilindro à extremidade; o valor é a altura do cilindro, indo das cores mais escuras às de maior intensidade; da parte central do cilindro, de baixo para cima, é a escala de cinza.

HSL

Modelo semelhante ao HSV, sendo as siglas referentes a Hue, Saturation e Lightness, trocando o termo Value do HSV por Lightness (Brilho). Esse modelo se diferencia na parte superior do cilindro onde o brilho é total (branco), sendo as cores

normais da parte superior do HSV na metade (50%) do HSL. A Figura 11 demonstra a diferença entre os dois modelos:

Figura 11: Comparação entre HSL e HSV



Fonte: https://meshlogic.github.io/posts/blender/materials/nodes-hsl-color-model/

3 TECNOLOGIAS ABORDADAS

As tecnologias que baseiam e são necessárias para as funcionalidades das ferramentas em foco desse trabalho representam as seguintes categorias:

- Plataforma: ambiente ou sistema que possibilita a execução das aplicações.
- Linguagens de programação: padrões de métodos para escrita formalizada de códigos para execução de instruções em uma plataforma.
- Ambientes e recursos de desenvolvimento: softwares e tecnologias que possibilitam a escrita de códigos e a execução das instruções nas plataformas.
- API/Biblioteca: tecnologias com instruções para possibilitar a adição de funcionalidades específicas nas linguagens de programação.

3.1 PLATAFORMAS (WEB E DESKTOP)

As plataformas são os ambientes que fornecem toda a infraestrutura tecnológica para que as ferramentas e demais aplicações para ser executadas. Esses ambientes podem ser sistemas de software ou hardware. As plataformas que dão suporte ao funcionamento das ferramentas P5.js e OpenGL são, respectivamente, Web e Desktop.

3.1.1 WEB

A World Wide Web (ou simplesmente WWW) é o sistema que funciona como uma plataforma para o acesso a diversos tipos de aplicações e conteúdos através da Internet. Foi criada no CERN (*European Center for Nuclear Physics* - Centro Europeu para Física Nuclear) por Tim Berners-Lee entre 1989 e 1991, de acordo com Kurose e Ross (2014). Esse sistema funciona com um conjunto de tecnologias que foram desenvolvidas por Berners-Lee e sua equipe para expandir o acesso a documentos pela rede mundial. “Berners-Lee e seus companheiros desenvolveram

versões iniciais de HTML, HTTP, um servidor Web e um navegador (browser) - os quatro componentes fundamentais da Web" (KUROSE e ROSS, 2014).

No início a Web tinha funcionamento basicamente para prover informações a partir de fácil acesso. Berners-Lee e Cailliau (1992) afirmam que o projeto World Wide Web (W3) permite o acesso ao universo de informação online. "O projeto W3 mescla recuperação de informações em rede e hipertexto para criar um sistema de informações global fácil, mas poderoso" (BERNERS-LEE e CALILLIAU, 1992).

Então a Web teve como proposta inicial a de dar acesso às páginas escritas em HTML (*HyperText Markup Language* – Linguagem de Marcação de Hipertexto) através de requisições de um software navegador seguindo o protocolo HTTP (*HyperText Transfer Protocol* – Protocolo de Transferência de Hipertexto). As páginas HTML, então, armazenadas nos servidores, eram encontradas pelo endereço do hospedeiro na rede. Pouco tempo depois começaram a ser disponibilizados vários outros tipos de formatos e mídias nos servidores mundo todo.

Com a evolução das tecnologias de rede e o alcance de velocidade dos meios de transmissão grande parte das atividades que eram apenas executadas em computadores sem a necessidade de conexão começou a migrar totalmente (ou em alta proporção) para a Web e ser acessada diretamente online. Hoje já é possível acessar imensa quantidade de serviços e mídias em toda a Web com o uso de vários dispositivos conectados à Internet. Além dos computadores, os telefones smartphones, tablets e até TVs possuem conexão e são considerados a maioria dos dispositivos conectados à rede.

3.1.2 MICROSOFT WINDOWS 10

O Microsoft (MS) Windows é o sistema operacional mais utilizado em computadores domésticos. Focado, basicamente na utilização de usuários comuns, fornece facilidade de acesso a recursos e permite boa experiência. A versão 10 é a mais recente lançada do sistema operacional pela Microsoft, importante organização da indústria tecnológica.

O sistema MS Windows fornece um ambiente com interface gráfica ao usuário com intuito de prover facilidade e praticidade para a realização das operações. O

sistema tem suporte para a execução da maioria dos softwares comerciais e livres tanto quanto para execução de jogos digitais.

O sistema da MS tem suporte para diversos tipos de aplicações e, para programadores que utilizam vários recursos de desenvolvimento, há um grande número de ferramentas que são compatíveis com o Windows.

3.2 LINGUAGENS DE PROGRAMAÇÃO

3.2.1 LINGUAGEM C

A linguagem C foi inventada por Dennis Ritchie no início da década de 1970, mais precisamente em 1972, no laboratório Bell (VICTORINE, VIVIANE, MIZRAHI, 1990). Schildt (1997) afirma que o criador, Dennis Ritchie, implementou a linguagem primeiramente em um computador DEC PDP-11 que utilizava o sistema UNIX.

C é resultado de um processo de desenvolvimento que começou em uma linguagem mais antiga, chamada BCPL, que ainda está em uso, em sua forma original, na Europa (SCHILDT, 1997). O autor também continua afirma que a linguagem BCPL foi desenvolvida por Martin Richards e influenciou a criação da linguagem B, por Ken Thompson. A linguagem B, na década de 1970, levou ao desenvolvimento da linguagem C.

Apesar de ser considerada uma linguagem de médio nível, C é bastante poderosa pois, classificada como uma linguagem estruturada, é utilizada para programar qualquer tipo de sistema. Isso tornou-a rapidamente uma das linguagens mais importantes e populares (VICTORINE, VIVIANE, MIZRAHI, 1990).

Atualmente, C ainda é bastante utilizada e maioria das linguagens de alto nível populares entre os programadores são baseadas nela, além de dar a programação base de vários sistemas e softwares eficientes do mercado.

3.2.2 LINGUAGEM JAVASCRIPT

JavaScript é a linguagem de programação da Web (FLANAGAN, 2011). Junto com HTML e CSS, compõe os pilares do desenvolvimento de páginas e sites na grande rede.

Atualmente está entre as linguagens de programação mais utilizadas no mundo. Flanagan (2011) explica esse fato afirmando que a maioria dos sites utilizam

a linguagem e todos os modernos navegadores, nos mais variados dispositivos, incluem interpretadores JavaScript, tornando-a a linguagem mais onipresente da história. Foi desenvolvida para integrar o software navegador Netscape (atual Mozilla) em 1995, nos primórdios da Web, como afirma Haverbeke (2018).

O nome JavaScript lembra a linguagem Java, porém a linguagem Web não foi baseada nessa e é completamente diferente. Além disso, a JavaScript - há muito tempo -, deixou de ser apenas uma linguagem em script e se tornou uma linguagem de propósito geral com robustez e eficiência. Outro ponto importante é esclarecer o nome padrão de versão da linguagem, ECMAScript, que foi adotado porque a Sun Microsystems (hoje Oracle) já havia registrado a marca JavaScript como definição de descrição da linguagem de implementação do Netscape. Mas, popularmente, é mesmo chamada JavaScript, o termo ECMAScript vem a ser mais frequentemente mencionado quando se refere à versão da linguagem.

O principal objetivo da linguagem JavaScript quando introduzida era trazer interatividade nas páginas Web. Isso porque, escritas em HTML e estilizadas com CSS (Cascading Style Sheet – Folha de estilo), essas páginas eram documentos estáticos. Com HTML descrevendo o conteúdo e a estrutura, CSS dando estilos visuais aos elementos, JavaScript deu dinamismo às páginas, oferecendo maior interatividade e conteúdos mais atrativos.

Flanagan também utiliza uma boa definição sobre papel da linguagem em seu funcionamento na Web:

> JavaScript é parte da tríade de tecnologias que todo desenvolvedor Web deve aprender: HTML para especificar o conteúdo das páginas Web, CSS para especificar a apresentação das páginas Web, e Javascript para especificar o comportamento das páginas Web (FLANAGAN, 2011)

Atualmente, como tem propósito geral, JavaScript é classificada como linguagem de alto nível, dinâmica, interpretada e não tipada. Abrange os estilos de orientação a objetos e funcional. Está sendo comumente utilizada por grandes organizações para ensinar lógica e programação para iniciantes por motivo da facilidade e simplicidade da linguagem.

3.4 API / BIBLIOTECAS PARA COMPUTAÇÃO GRÁFICA

3.4.1 OPENGL

OpenGL (*Open Graphics Library - "Biblioteca gráfica aberta", em tradução livre*) é um conjunto aberto de bibliotecas com funções para a geração de aplicações gráficas. Há vários autores que utilizam definições que apontam diferentes perspectivas sobre essa tecnologia, mas que seguem com fidelidade a conceituação geral da ferramenta.

Azevedo e Conci (2003) definem a tecnologia de forma bem convencional. De acordo com os autores, OpenGL é uma interface de software (API - *Application Program Interface*) para aceleração da programação de dispositivos gráficos, com vários comandos para especificação de objetos e operações necessárias para a produção de aplicações gráficas interativas 3D. Shreiner et. al (2013) também definem o seguinte: "OpenGL é uma Interface de Programação de Aplicação - "API" - na qual é meramente uma biblioteca de software para o acesso das funções dos dispositivos gráficos". "OpenGL é uma interface em que sua aplicação pode usar para acessar e controlar os subsistemas gráficos do dispositivo em que a executa" (Graham Sellers Richard, S. Wright, Jr. Nicholas Haemel, 2016).

Azevedo e Conci (2003) diferenciam OpenGL das linguagens de programação:

> OpenGL não é uma linguagem de programação como C, C++ ou Java, é uma poderosa e sofisticada biblioteca de códigos, para desenvolvimento de aplicações gráficas 3D em tempo real, seguindo a convenção de chamada de biblioteca da linguagem de programação C. (AZEVEDO e CONCI, 2003)

Desde sua primeira versão em 1992, desenvolvida pela Silicon Graphics Inc., OpenGL é o padrão mais amplamente utilizado da indústria para geração de gráficos computacionais. A explicação para isso vem do fato de ser uma tecnologia aberta de alta performance, o que possibilitou ser adotada por diversos fabricantes de hardware e maioria das aplicações gráficas desde então foram construídas com a API, tornando-a multiplataforma.

Os principais fabricantes de PC e supercomputadores como IBM, Microsoft, Samsung etc. adotaram OpenGL como estratégia padrão aberta para hardware gráfico, Segundo Azevedo e Conci (2003). São várias as aplicações que começaram a utilizar OpenGL como base:

“Suas aplicações variam de ferramentas CAD a jogos e imagens médicas ou programas de modelagem usados para criar efeitos especiais para televisão e cinema (como em Jurassic Park e Star Wars)”.

Os mesmos autores continuam destacando a utilização da tecnologia na indústria do entretenimento:

> Outro ponto importante é a vasta utilização dessa interface na indústria do entretenimento, sendo base das tecnologias de efeitos especiais em filmes famosos em Hollywood, bem como a base dos gráficos em diversos videogames e consoles. (AZEVEDO e CONCI, 2003)

Acrescentam ainda importantes softwares de modelagem e jogos famosos:

> Dentre os programas de modelagem, podemos citar 3D MAX, Bryce, Character Studio, Lightwave, Lightscape, Rhino 3D, TrueSpace e muitos outros. Dentre os jogos, temos Quake, Half-Life, MDK2, Baldur’s Gate, Descent, Maden NFL 2001 etc. (AZEVEDO e CONCI, 2003)

3.4.2 GLUT

GLUT (*OpenGL Utility Toolkit*) é uma biblioteca com funções e comandos que simplificam a criação e configuração das janelas gráficas que renderizam o programa OpenGL. “GLUT torna o processo de criação de uma aplicação OpenGL simples, desde sua forma mais básica, de forma que apenas quatro passos são necessários para fazer a aplicação funcionar” (SHREINER et. al, 2013). Essa interface foi desenvolvida com a proposta de atender, de forma bem simplificada, à criação de um sistema de janela independente para OpenGL que desse suporte e maior gerenciamento na geração e execução. “GLUT é projetado para preencher a necessidade de uma interface de programação independente para sistema de janela” (KILGARD, 1996).

Assim a biblioteca GLUT simplifica a implementação dos programas utilizando a renderização OpenGL e requere poucas rotinas para a renderização de gráficos na janela gerada (KILGARD, 1996).

3.4.3 P5.JS

P5.js é uma biblioteca gráfica para JavaScript que facilita a programação de desenhos e animações na Web. A biblioteca P5.js foi criada por Lauren McCarthy com primeira versão beta apresentada em 2014. Sua implementação firmou em se

basear na linguagem Processing. Assim, como está descrito no site oficial (p5js.org), trata-se de software livre sob a licença GNU Lesser General Public License e publicação pela Free Foundation Software, v. 2.1.

Em 2001, Casey Reas e Ben Fry, como relata McCarthy (2016), começaram a trabalhar em uma nova plataforma para tornar a programação de gráficos interativos algo fácil e simples. Isso se deu pela frustração dos dois sobre o quanto era difícil fazer programas do tipo em linguagens como C++ e Java. E, assim, inspirados na simplicidade de programar em linguagens que tiveram contato na infância, como Logo e BASIC, desenvolveram a ferramenta Processing. O intuito principal era tornar a programação gráfica simples para que pudessem testar ideias e fazer os esboços com facilidade, deixando o código acessível para artistas, designers e até educadores.

Segundo McCarthy (2016) com o passar dos anos, Processing foi ganhando uma grande comunidade de adeptos e passou a fazer parte de vários programas e matrizes curriculares em cursos de artes, design, Ciência da Computação entre outros. Em certo momento, os dois criadores do projeto, ao conversarem com o próprio McCarthy, fizeram o questionamento de como seria se a ferramenta Processing fosse uma tecnologia Web. Assim nasceu um projeto novo, chamado P5.js.

*"*P5.js inicia com o objetivo original de Processing, de tornar programação algo acessível para artistas, designers, educadores, e iniciantes, então reinterpretando o código para a atual Web utilizando JavaScript e HTML" (McCarthy, 2016).

A grande inovação que essa biblioteca traz é a facilidade e simplicidade de gerar desenhos e animações com poucas linhas de código. É utilizado o termo "sketch" para descrever os desenhos gerados no espaço chamado "canvas". Isso significa que a ferramenta tem a proposta de simular o desenho de esboços, rascunhos ou qualquer modelo inicial (sketch) em uma folha ou bloco de notas (canvas). Contudo, utilizando código em vez de lápis.

4 DESCRIÇÃO E ANÁLISE COMPARATIVA

Neste tópico serão feitas descrições dos requisitos de sistema para cada uma das ferramentas em foco e serão mostrados alguns passos para a instalação de ambas. Também será descrita a utilização das tecnologias OpenGL e P5.js na geração de um ambiente gráfico, na geração de primitivas geométricas 2D neste ambiente, além da coloração dessas primitivas. Toda a geração gráfica com OpenGL tem implementações na linguagem C e a geração com P5.js é baseada em JavaScript.

Então serão produzidas análises comparativas com relação à facilidade de se obter, configurar e utilizar as duas ferramentas na escrita de códigos para a geração de gráficos.

Inicialmente serão descritos os requisitos para se poder utilizar as ferramentas e os passos necessários para a instalação dessas a ponto de estarem prontas para uso. Assim, serão apresentadas as estruturas de programa (código-fonte) e a escrita dos códigos básicos necessários para que, então, se possa iniciar a programação das formas geométricas e as definições de coloração em cada uma das tecnologias em foco. As análises comparativas são produzidas a partir de imagens de captura de tela dos ambientes de codificação e das janelas geradas.

4.1 OPENGL

4.1.1 REQUISITOS E INSTALAÇÃO

No site oficial da companhia Kronos, que mantém OpenGL e demais tecnologias, há um conjunto de páginas disponíveis no tipo Wiki que dão informação geral sobre utilização da API. Essas páginas estão disponíveis em <https://www.khronos.org/opengl/wiki/>.

Em todas as três principais plataformas Desktop (Linux, McOS e Windows), OpenGL já vem, mais ou menos, junto com o sistema. O que é preciso é estar com o *driver* do hardware gráfico atualizado. Informações mais detalhadas e também os links para obter ou atualizar os *drivers* dos hardwares gráficos da AMD, Intel e

NVidia estão disponíveis na página <https://www.khronos.org/opengl/wiki/Getting_Started>.

Para programar utilizando OpenGL é preciso, além do *driver*, o pacote de desenvolvimento da API, o qual dependerá da plataforma e da linguagem de programação escolhida.

Como a plataforma escolhida para a utilização de OpenGL neste trabalho foi o Windows 10 e a implementação é feita com a linguagem C, são necessários um editor de texto para escrever os códigos e um compilador para transformar os arquivos com os códigos salvos na extensão da linhagem (.c) em programas compatíveis para serem executados no ambiente do sistema operacional. Foi escolhido software Dev C++, uma IDE (*Integrated Development Environment* - Ambiente de Desenvolvimento Integrado) que já integra editor e o compilador MinGW GCC. O software é livre e Dev C++ está disponível para download em <https://sourceforge.net/projects/orwelldevcpp/>.

Tendo o Dev C++ já instalado e pronto para uso, junto com o compilador MinGW GCC, o próximo passo é fazer o download dos arquivos necessários para poder integrá-los nos diretórios onde está instalado o compilador e, assim, começar a programar com a API. Os arquivos são referentes ao GLUT, que já integra as funcionalidades necessárias.

Os arquivos necessários para permitir o funcionamento da API no Dev C++ são: *glut.h*, *libglut.a*, *libglut32.a* e *glut32.dll*. O link para download destes arquivos e informações detalhadas de em quais diretórios eles deverão ser colocados e de como configurar o ambiente para encontrá-los estão disponíveis em <https://www.ft.unicamp.br/~magic/opengl/instala-windows.html>.

4.1.2 ESTRUTURA DE CÓDIGO

Um código-fonte na linguagem C, basicamente, segue a seguinte estrutura inicial definida pelas linhas de comentários iniciadas com “//” na Figura 12:

Figura 12: Estrutura de código-fonte em C

```c
//Arquivos de cabeçalho - Bibliotecas
#include <arquivo.h>

//Declaração de funções e variáveis
void Funcao(){
    //Instruções
}

int main(){
    //Instruções de execução
    return 0;
}
```

Fonte: Autoria própria

A estrutura básica de um programa OpenGL escrito em C contém as definições iniciais que permitem gerar uma janela gráfica na qual servirá de ambiente para qualquer elemento gráfico. É apresentada na Figura 13:

Figura 13: Estrutura de código-fonte OpenGL

```c
//Arquivos de cabeçalho - Bibliotecas
#include <GL/glut.h>

//Declarações globais e funções
void Display(){
    //Corpo da função
}

//Função principal
int main(int argc, char** argv){
//Inicialização e chamada de funções
    glutInit(&argc, argv);
    //...
    glutDisplayFunc(Tela);
    glutMainLoop();

    return 0;
}

```

Fonte: Autoria própria

4.1.3 BIBLIOTECAS DE FUNÇÕES

Segundo Azevedo e Conci (2003), OpenGL tem funcionalidade como outras APIs no Windows: “De forma semelhante a outras APIs do Windows, os comandos do OpenGL são disponibilizados através de bibliotecas dinâmicas, conhecidas como DLLs (*Dynamic Link Library*) e seus respectivos arquivos *header* e *library*”. Os autores continuam explicando que os arquivos *header* são incluídos no código-fonte e os *library* são incluídos no projeto onde se encontra o código.

As bibliotecas base da API OpenGL são OPENGL32.DLL, GLU32.DLL e GLUT.DLL. A Tabela 2 explica cada uma:

Tabela 2: Bibliotecas OpenGL

Biblioteca	**Arquivo header**	**Arquivo library**	**Funcionalidades**
OPENGL32.DLL	gl.h	opengl32.lib	Contém as funções padrão do OpenGL com prefixo **gl**
GLU32.DLL	glu.h	glu32.lib	Biblioteca de utilitários com funções para desenho de esferas, cubos, discos, cilindros, etc. com prefixo **glu**
GLUT.DLL	glut.h	glut32.lib	OpenGL Utility Toolkit (Conjunto de Ferramentas Utilitárias do OpenGL: é um sistema de gerenciamento de janelas, com prefixo **glut**

Fonte: Autoria própria

As funções utilizadas para o objetivo desse trabalho pertencem às bibliotecas OPENGL32 e GLUT. Todas as funções OpenGL, como facilidade, podem ser utilizadas simplesmente com a chamada do arquivo *header glut.h,* inserindo a linha do Código 1 no cabeçalho do código-fonte em C:

```
#include <GL/glut.h>
```
Código 1

OPENGL32

As funções da biblioteca OPENGL32 são utilizadas na configuração inicial da janela e na geração das primitivas geométricas, além das definições de cor, tamanho e outras configurações de como serão representados os desenhos na janela gráfica. Essas funções utilizam o prefixo gl. Geralmente as funções dessa biblioteca são inclusas em uma função de renderização declarada que será chamada na função principal do programa, a função main().

GLUT

A biblioteca GLUT (*OpenGL Utility Toolkit*) contém as funções que facilitam a interação com os recursos do sistema operacional, abrindo janelas e gerenciando entradas como mouse e teclado.

As suas funções são utilizadas dentro da função principal do programa, a função main(). Além de inicializar as demais funções OpenGL e chamar a função responsável por gerar a janela e os desenhos, a biblioteca traz as funções que também definem as configurações iniciais da janela gráfica. A Figura 14 mostra as funções com prefixo *glut* dentro da função main() e as explicações para cada linha estão na sequência, após o "//".

Figura 14: Definições GLUT para janela OpenGL

```
54  int main(int argc, char** argv){
55
56      glutInit(&argc, argv): //Inicializa a OpenGL
57      glutInitWindowSize(w, h): //Cria a tela com as dimensões(w, h)
58      glutInitWindowPosition(x, y); //Define a posição da tela
59      glutCreateWindow("Tela"); //Identifica a tela com o nome
60      glutDisplayFunc(Tela); //Faz a chamada da função Tela()
61      glutMainLoop(); //Responsável pela atualização (loop)
62
63      return 0;
64
65  }
```

Fonte: Autoria própria

Predefinições

Dentre as predefinições estão as funções glClearColor(), que determina a cor de fundo da janela, e glClear(), que limpa as cores atuais do buffer. Os parâmetros

que são inseridos em glClearColor() são os valores com ponto flutuante que definirão a cor em RGBA, com cada valor dentro do intervalo 0.0f – 1.0f. Já o parâmetro de glClear() será GL_COLOR_BUFFER_BIT. Assim deverão ser as linhas, no Código 2, para o início da função de renderização:

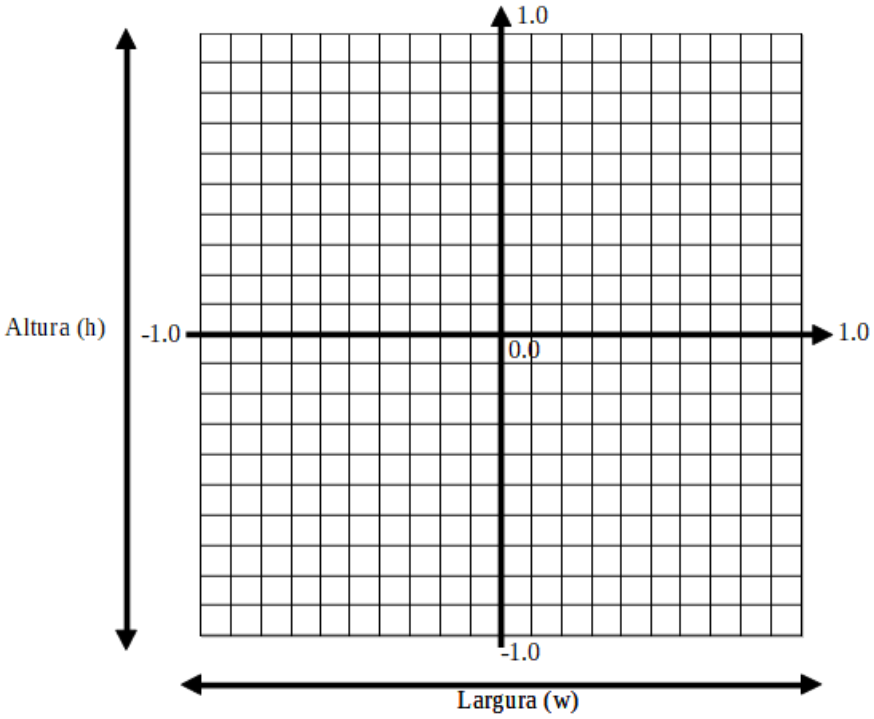
```
glClearColor(0.0f, 0.0f, 0.0f, 1.0f); //Cor preta e opaco
glClear(GL_COLOR_BUFFER_BIT); //Limpa o buffer
```

Código 2

4.1.4 SISTEMA DE COORDENADAS

O sistema de coordenadas OpenGL é, inicialmente, definido de forma que a origem (0.0, 0.0) seja no centro da janela gerada, como mostra a Figura 15:

Figura 15: Sistema de coordenadas de janela OpenGL



Fonte: Autoria própria

Como está definido na figura, os valores para a coordenada X vão crescendo da esquerda para a direita. À esquerda da origem (0.0) estão os valores negativos que vão até (-1.0) e à direita os positivos, indo até (1.0). Para os valores da coordenada Y o crescimento é de baixo para cima. Assim os valores abaixo da origem são negativos e vão até (-1.0) e acima os valores positivos, que vão até (1.0).

Logo, as coordenadas de qualquer posição na janela assumem a forma (X, Y). Como exemplo: (-0.5, 0.4) e (0.5, 1.0).

4.1.5 FUNÇÕES PARA PRIMITIVAS

Para o desenho na janela gráfica são utilizadas as funções glBegin() e glEnd() como delimitadores e as primitivas são basicamente conjuntos de um ou mais pontos, descritos pela função glVertex2d(x, y) – x e y como valores das coordenadas na janela. Para a descrição da primitiva geométrica é utilizado o prefixo GL_ seguido do nome das primitivas em inglês, dentro dos parênteses da função glBegin(). A Tabela 3 mostra os comandos e quantidades de pontos para cada primitiva:

Tabela 3: Primitivas e comandos OpenGL

Primitiva geométrica	Comando	Q. de glVertex2d(x, y)
Pontos	GL_POINTS	1 para cada ponto
Linhas	GL_LINES	2 para cada linha
Triângulos	GL_TRIANGLES	3 para cada triângulo
Quadrados/Retângulos	GL_QUADS	4 para cada retângulo
Polígonos	GL_POLYGON	3 ou mais

Fonte: Autoria própria

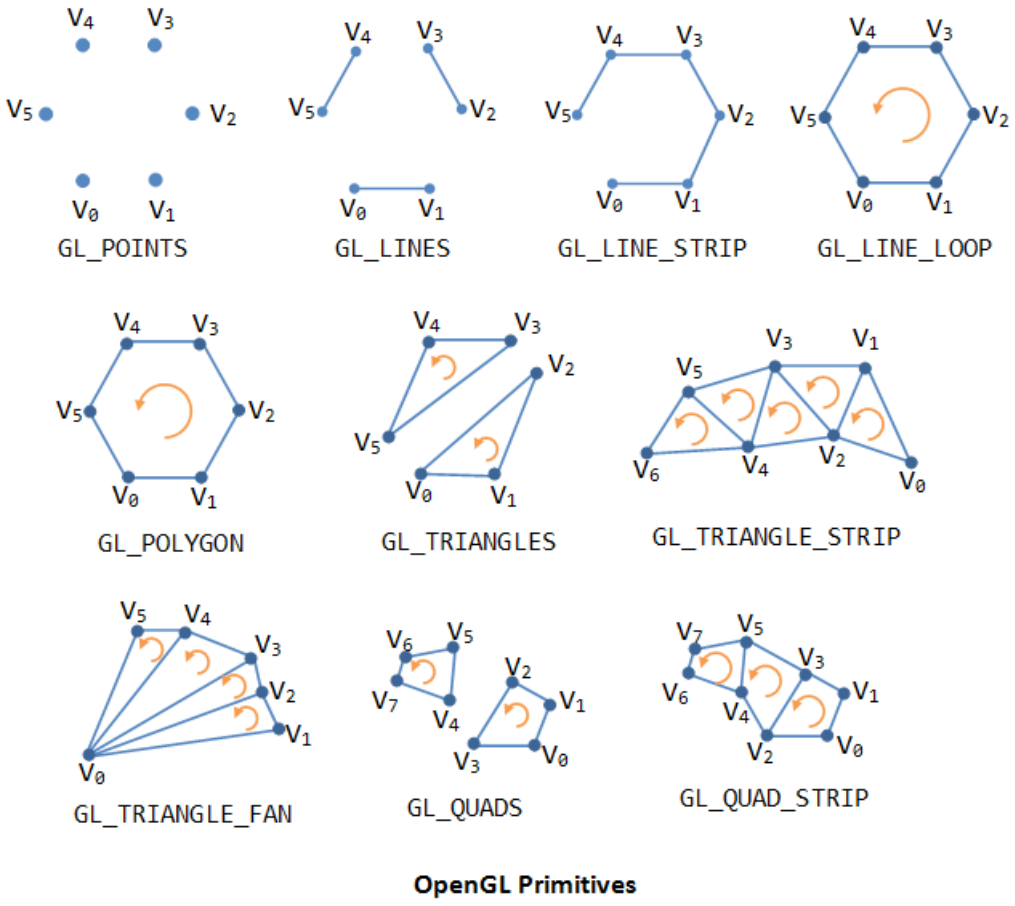
O Código 3 exemplifica as linhas que deverão ser escritas para a geração de um ponto e uma reta formada por dois pontos:

```
glBegin(GL_POINTS);
glVertex2d(0.0, 0.0);
glEnd();
glBegin(GL_LINES);
glVertex2d(0.0, 0.0);
glVertex2d(0.5, 0.5);
glEnd();
```

Código 3

Para se ter uma boa visualização dos resultados, ao gerar as primitivas é importante seguir uma certa ordem na colocação dos pontos que aparecerão na janela, ou seja, suas coordenadas. A Figura 16 serve como modelo:

Figura 16: Guia de vértices para formação de primitivas



Fonte: https://www.ntu.edu.sg/home/ehchua/programming/opengl/CG_Introduction.html

4.1.6 FUNÇÕES PARA COLORAÇÃO

A função glColor3f() é utilizada para definição de cores no formato RGB das primitivas a serem geradas. Os parâmetros da função são valores de ponto flutuante e deve estar dentro do intervalo 0.0f – 1.0f. As linhas com essa função devem ser escritas logo antes do código das primitivas, antes da função glBegin(). Alguns exemplos estão na Figura 17:

Figura 17: Funções de cores OpenGL

```
glColor3f(0.0f, 0.0f, 1.0f); //Define a cor azul no total
glBegin(GL_POINTS); //Define a primitiva ponto
glVertex2d(0.1, 0.1); //Define a coordenada (0.1, 0.1) na tela
glEnd(); //Finaliza o desenho

glColor3f(1.0f, 0.0f, 0.0f); //Define a cor vermelha no total
glBegin(GL_LINES); //Define a primitiva linha
glVertex2d(0.0, 0.0); //Coordenada do primeiro ponto na tela
glVertex2d(0.5, 0.5); //Coordenada do segundo ponto na tela
glEnd(); //Finaliza o desenho

glColor3f(0.0f, 1.0f, 0.0f); //Define a cor verde no total
glBegin(GL_TRIANGLES); //Define a primitiva triângulo
glVertex2d(0.0, 0.0); //Coordenada do primeiro ponto na tela
glVertex2d(0.5, 0.5); //Coordenada do segundo ponto na tela
glVertex2d(0.0, 0.5); //Coordenada do terceiro ponto na tela
glEnd(); //Finaliza o desenho
```

Fonte: Autoria própria

Renderização

Esta função responsável por fazer todo o código presente que irá ser gerado em uma janela na tela é glFlush(). A linha deve ser escrita depois de todo o código a ser desenhado, sendo a última função da função que gera a tela gráfica.

A Figura 18 mostra um exemplo da função gerando a janela gráfica com um triângulo sendo apresentado:

Figura 18: Função de renderização

```
 4  void Display(){
 5      glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
 6      glClear(GL_COLOR_BUFFER_BIT);
 7
 8      glColor3f(0.0f, 1.0f, 0.0f);
 9      glBegin(GL_TRIANGLES);
10      glVertex2d(-0.4, 0.0);
11      glVertex2d(0.4, 0.0);
12      glVertex2d(0.0, 0.7);
13      glEnd();
14
15      glFlush();
16  }
```

Fonte: Autoria própria

4.2 P5.JS

4.2.1 REQUISITOS E INSTALAÇÃO

Por motivos de P5.js ser uma biblioteca JavaScript, inicialmente é necessário ter um navegador Web (Browser) com suporte à linguagem JavaScript. Google Chrome, Mozilla Firefox e os demais navegadores modernos dão suporte à linguagem, com interpretadores que executam o código em tempo real.

Para a escrita dos códigos é necessário um editor de texto, livre de escolha. Há vários editores de texto especiais para códigos que oferecem funcionalidades e boa experiência na escrita, como Notepad++, Atom, Visual Studio Code etc.

O download da biblioteca pode ser feito na página específica do site oficial, no endereço <https://p5js.org/download/>. Feito o download da biblioteca e movendo o conteúdo para um diretório específico, é preciso apenas fazer o link do arquivo p5.min.js do pacote baixado dentro do código-fonte HTML, no qual deverá apresentar os gráficos, com a linha: <script src="../p5.min.js"></script>.

Alternativamente é possível fazer o link do arquivo P5.js hospedado online. Todas as versões P5.js estão disponíveis em <https://cdnjs.com/libraries/p5.js>. A linha, utilizando a última versão, será: <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.9.0/p5.min.js"></script>.

A partir da inclusão da linha que faz o link do arquivo da biblioteca P5.js, já é possível escrever códigos P5.js para gerar os gráficos, apenas salvando esses códigos em um arquivo com a extensão JavaScript (.js) no código-fonte HTML depois da linha que faz o link da biblioteca. Um exemplo é dado na seguinte linha de código: <script src="sketch.js"></script>.

Todo o resultado é visto abrindo o arquivo com o código-fonte HTML (.html) que contém os links para os arquivos JavaScript diretamente no navegador.

Um modo alternativo para utilizar a biblioteca, e que é de maior facilidade, é utilizando o editor online do site oficial da ferramenta P5.js. Esse editor permite a escrita de código e execução diretamente no ambiente do navegador, gerando um

espaço gráfico chamado Canvas para a visualização dos resultados, sem que seja preciso realizar qualquer download.

Para fins de simplificação as demonstrações e análises comparativas feitas no trabalho serão realizadas com o próprio editor online do site oficial da biblioteca, uma vez que ele disponibiliza o resultado de execução em uma tela ao lado do código. O editor está disponível em: <https://editor.p5js.org/ >.

4.2.2 ESTRUTURA DE CÓDIGO

A Figura 19 apresenta a estrutura básica para se começar a utilizar as funções da biblioteca gráfica:

Figura 19: Estrutura P5.js

Fonte: Autoria própria

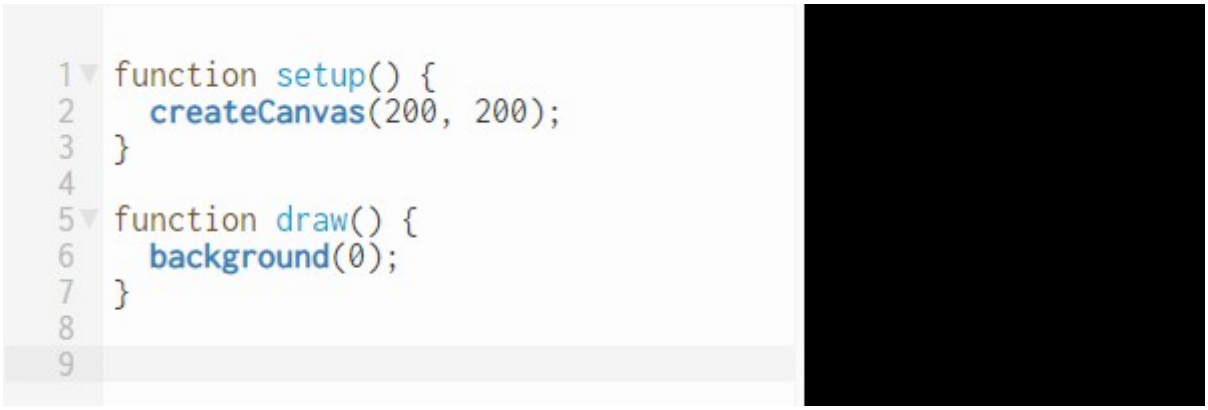
Na figura 19, a estrutura básica só contém as duas funções: setup() e draw(). Na função setup() está presente o código para o Canvas. Já a função draw() terá todo o código que fará gerar no Canvas o conteúdo gráfico esperado.

Para gerar o Canvas na função setup deve ser adicionada a função createCanvas(). Essa última terá como atributos os valores em quantidades de pixels do tamanho do Canvas. Por exemplo, a linha createCanvas(200, 200) gera o Canvas de largura com 200 pixels e altura também com 200 pixels.

Depois da linha que cria o Canvas deve ser adicionada a linha background(), ainda na função setup() ou já na função draw(). Essa função define a cor de fundo do Canvas e utiliza como parâmetros um valor no intervalo 0 – 255, representando a escala em cinza, ou três valores, cada um também dentro do intervalo 0 – 255, que define a coloração do sistema RGB.

Por padrão a função background() é definida dentro da função draw(), para possibilitar a geração de animações. A figura 20 mostra a definição das duas funções e seus parâmetros no código e o resultado gerado ao lado:

Figura 20: Gerando o Canvas



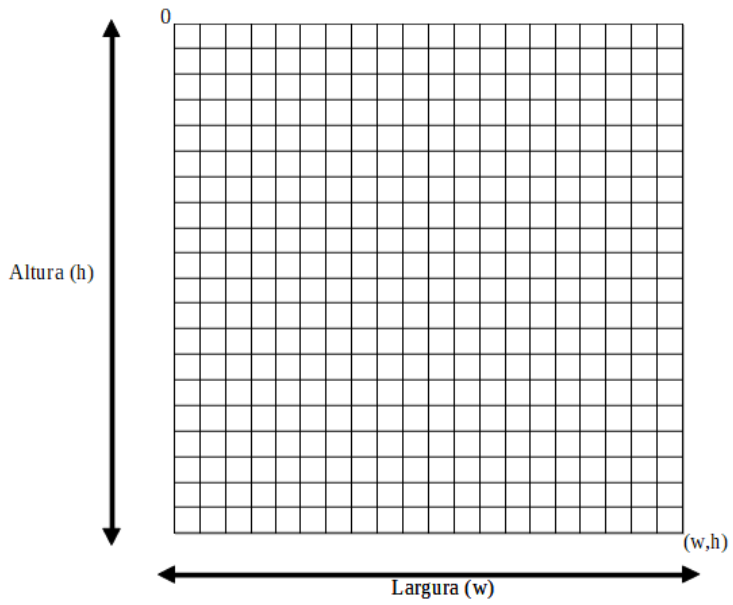
Fonte: Autoria própria

Assim, na figura 20, a função createCanvas() define o tamanho do Canvas em 200 x 200 pixels e a função background() define a cor de fundo com valor 0, que representa a cor preta na escala de cinza.

4.2.3 SISTEMA DE COORDENADAS

O Canvas gerado em P5.js terá o sistema de coordenadas, para o caso de duas dimensões 2D, de acordo com a Figura 21:

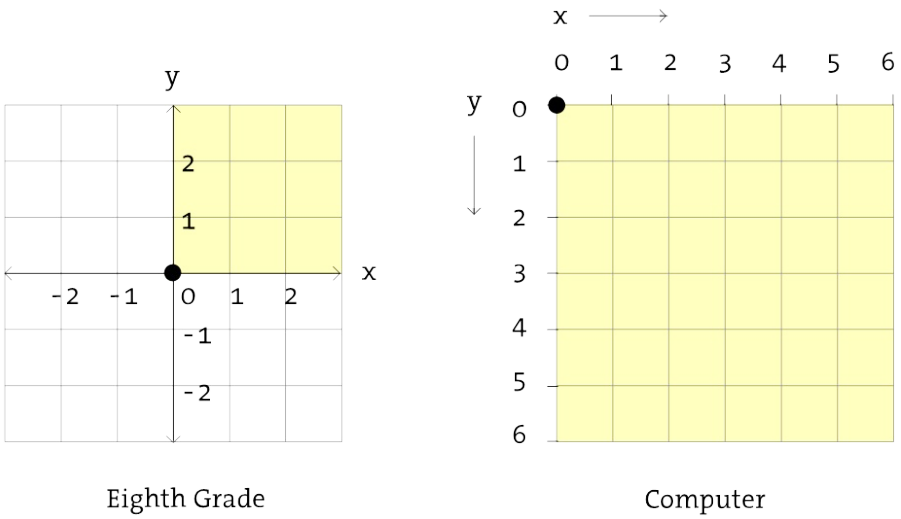
Figura 21: Sistema de coordenadas do Canvas em P5.js



Fonte: Autoria própria

Os valores (X, Y) que vão da origem (0, 0) ao valor máximo definido pelos parâmetros da função createCanvas(). Assim, a Figura 22 abaixo mostra o sistema de coordenadas normal e o utilizado para representar a tela no computador, que a P5.js utiliza predefinidamente. O valor de X cresce para a direita e o valor de Y cresce para baixo:

Figura 22: Sistema normal (esquerda) e Canvas P5.js (direita)



Fonte: https://p5js.org/learn/coordinate-system-and-shapes.html

4.2.4 FUNÇÕES PARA PRIMITIVAS

Assim como OpenGL, a biblioteca P5.js contém um grande número de funções, porém, seguindo o modelo de análises utilizado até aqui, as funções a serem apresentadas estão relacionadas a geração de primitivas geométricas, coloração e outras relevantes.

Do mesmo modo que as demais funções da biblioteca, as que definem as primitivas são, basicamente, escritas em uma linha cada. A Tabela 4 demonstra as linhas que descrevem as funções de algumas primitivas:

Tabela 4: Primitivas P5.js

Primitiva Geométrica	Função	Parâmetros
Ponto	point()	(x, y)
Linha	line()	(x1, y1, x2, y2)
Triângulo	triangle()	(x1, y1, x2, y2, x3, y3)
Quadrado/Retângulo	rect()	(x, y, w, h)
Circulo	circle()	(x, y, r)
Elipse	ellipse()	(x, y, w, h)

Fonte: Autoria própria

Obs.: as letras presentes nos parâmetros são referentes ao valor numérico no sistema de coordenadas do Canvas.

Análise da Tabela 4:

As funções das primitivas ponto, linha e triângulo utilizam os valores x e y como coordenadas no espaço bidimensional (2D) do Canvas. Dessa forma a função point() utiliza a coordenada x como valor na direção horizontal e y para o valor na direção vertical.

Para gerar um quadrado ou retângulo os parâmetros utilizados definem na função rect() os valores das coordenadas (x, y) que dão o pixel inicial (esquerdo superior) no Canvas e os valores de largura e altura, w e h, respectivamente.

Na geração de um círculo há uma diferença em relação à geração do quadrado e retângulo: as coordenadas (x, y) representam a posição do centro da primitiva no Canvas. Já o valor de r define o raio do círculo – distância do centro à extremidade. Na geração de uma elipse também são utilizadas as coordenadas (x, y) como posição do centro, porém, em lugar do raio por r se utiliza w para a largura e h para altura.

4.2.5 FUNÇÕES PARA COLORAÇÃO

Como já dito, onde foi explicada a função background(), o sistema RGB é utilizado pela biblioteca P5.js para dar coloração, não só para o fundo do Canvas, mas também para colorir os elementos gráficos gerados, sejam apenas pontos, linhas ou qualquer uma das primitivas apresentadas como também qualquer forma complexa ou abstrata.

Para colorir as formas gráficas é necessário inserir a função fill() antes da linha da função que gera a forma no Canvas. Os parâmetros são, novamente, os três valores correspondentes às cores R (vermelho), G (verde) e B (azul), no intervalo 0 – 255.

Deve-se notar que uma vez inserida a linha fill(), todas formas gráficas na sequência adotaram a cor definida pela função. Assim, caso se queira utilizar outra cor para uma forma, é necessário adicionar a função fill() com os valores para a cor desejada logo antes do código que gera a forma de interesse.

4.3 ANÁLISE COMPARATIVA - OPENGL VERSUS P5.JS

De acordo com as descrições anteriores das duas tecnologias em foco, são feitas, a seguir, análises comparativas entre as duas ferramentas em relação aos requisitos e instalação, na configuração e na escrita de códigos.

4.3.1 REQUISITOS E PASSOS PARA UTILIZAÇÃO

As tabelas comparam o que é preciso para se começar a escrever códigos utilizando as duas ferramentas:

Tabela 5: Requisitos e Download

	Requisitos	Download
OpenGL	Driver de hardware gráfico atualizado	Pacote de desenvolvimento (GLUT)
P5.js	Navegador Web com interpretador JavaScript	Arquivos da biblioteca

Fonte: autoria própria

Tabela 6: Instalação, Configuração e Codificação

	Instalação	Configuração	Codificação
OpenGL	Arquivos nos diretórios do compilador e do sistema	Configurar a IDE para fazer o link dos arquivos de compilação	Linguagem C
P5.js	Ter os arquivos em um diretório de escolha	Linha <script> com local da biblioteca em código HTML	Linguagem JavaScript

Fonte: autoria própria

As ferramentas OpenGL e P5.js seguem uma quantidade de passos parcialmente iguais, mas a biblioteca JavaScript se destaca por apresentar maior

facilidade, especialmente na instalação e configuração, nas quais só é necessário fazer a referência do caminho da biblioteca baixada. Não há uma especificação do local mas é de bom modo ter a biblioteca P5.js no mesmo diretório que o arquivo HTML ou diretório próximo, o que diminui a escrita a escrita da linha <script>.

A seguir são comparadas a quantidade de linhas escritas e o tamanho das funções junto aos seus parâmetros. Os resultados gerados são semelhantes, a ponto de possibilitar uma percepção de como cada uma das ferramentas é utilizada para gerar um mesmo resultado.

4.3.2 ESTRUTURA

As estruturas básicas de código-fonte em cada uma das ferramentas são apresentadas na Figura 23, sendo a estrutura OpenGL à esquerda e a P5.js à direita.

Figura 23: Comparação de estruturas

```
#include <GL/glut.h>

void Display(){

}

int main(int argc, char** argv){
    glutInit(&argc, argv);
    glutInitWindowSize();
    glutInitWindowPosition();
    glutCreateWindow();
    glutDisplayFunc(Display);
    glutMainLoop();

    return 0;
}
```

```
function setup() {

}

function draw() {

}
```

Fonte: Autoria própria

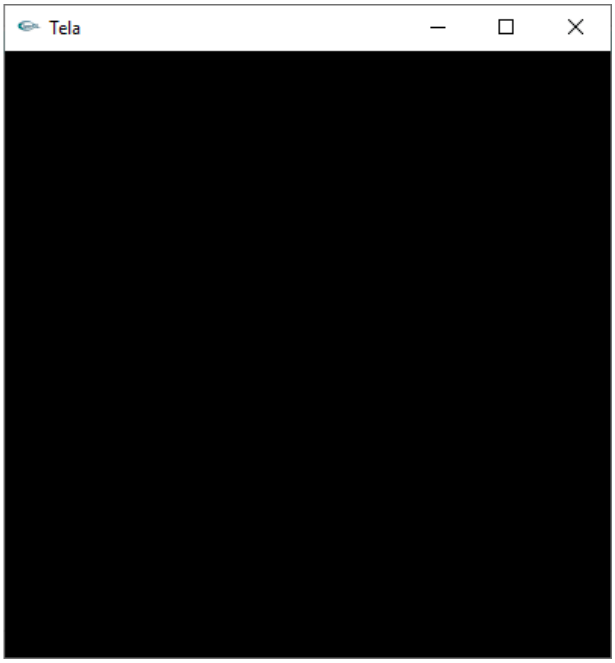
A estrutura OpenGL é, em sua base, composta pela função de renderização, `Display()`, e a função principal, `main()`, além da linha *header* (#include <GL/glut.h>). Nessa função principal são utilizadas outras funções para inicialização e definições do programa OpenGL. A estrutura P5.js também contém duas funções, porém com uma maior simplicidade, menos linhas de código.

4.3.3 JANELA – CANVAS

A execução dos códigos resultam em janelas gráficas para OpenGL e um Canvas para P5.js.

Figura 24: Janela OpenGL

```c
#include <windows.h>
#include <GL/glut.h>

void Tela(){
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT);
    glFlush();
}

int main(int argc, char** argv){

    glutInit(&argc, argv);
    glutInitWindowSize(400, 400);
    glutInitWindowPosition(50, 50);
    glutCreateWindow("Tela");
    glutDisplayFunc(Tela);
    glutMainLoop();

    return 0;

}

```



Fonte: Autoria própria

Figura 25: Canvas P5.js

```javascript
function setup() {
  createCanvas(400, 400);
}

function draw() {
  background(0);
}


```

Fonte: Autoria própria

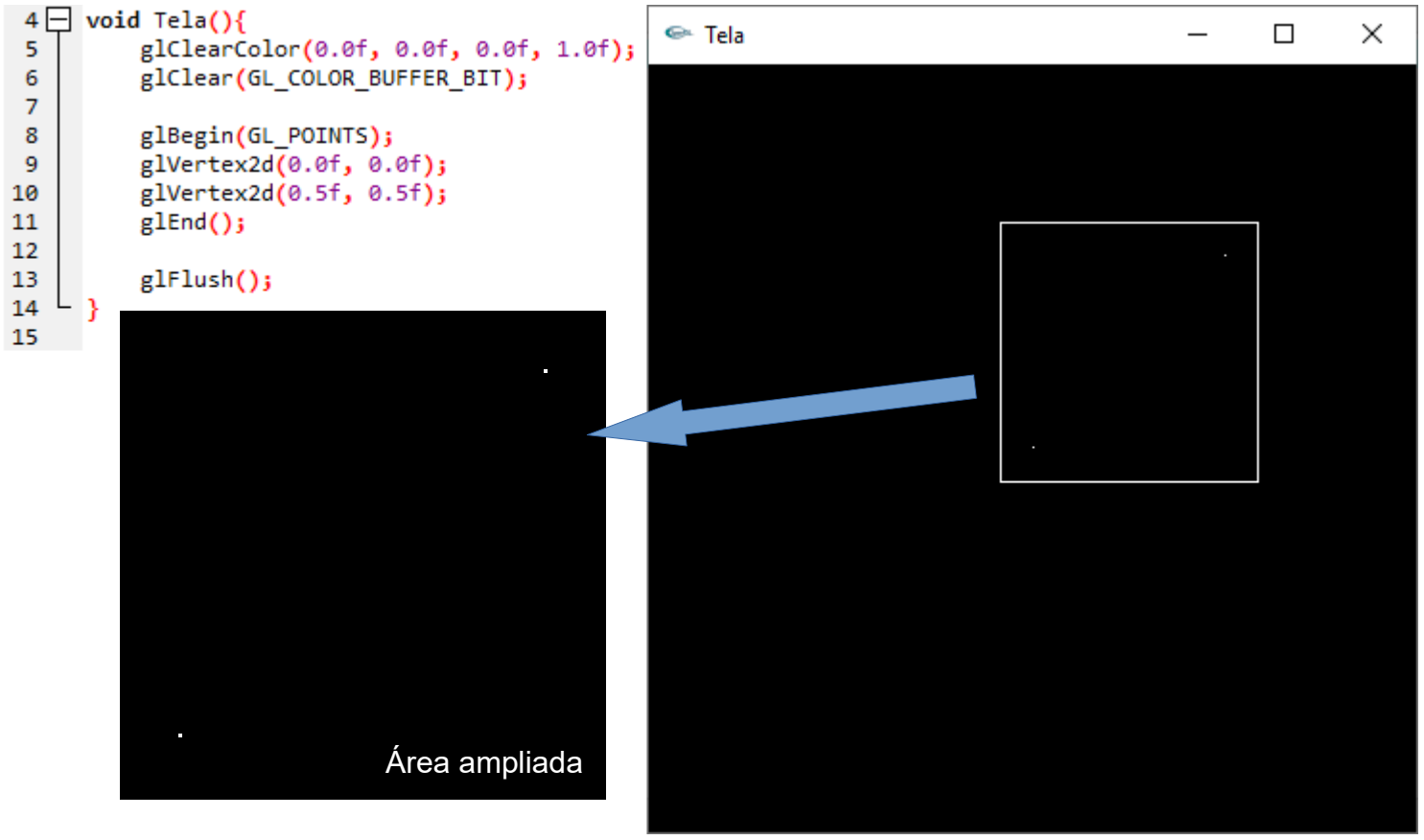
Na Figura 24, as funções para definição de tamanho, posição e nome da janela ocupam seis linhas de código dentro da função main(). A função Tela(), que gera a janela gráfica, contém três linhas de código além de seu escopo. No código P5.js, mostrado na Figura 25, são utilizadas apenas duas funções para gerar o Canvas e para definir a cor de fundo, o deixando com o mesmo tamanho (400, 400) e mesma cor da tela OpenGL. O código P5.js é notavelmente mais econômico e simples.

4.3.4 PRIMITIVAS

As figuras 26 e 27 utilizam os códigos nas duas ferramentas para gerar um mesmo resultado, o que faz focar na diferença entre as linhas escritas. Para melhor visualização as figuras referentes aos resultados dos pontos estão em tamanho original para poderem mostrar as formas sem dificuldade.

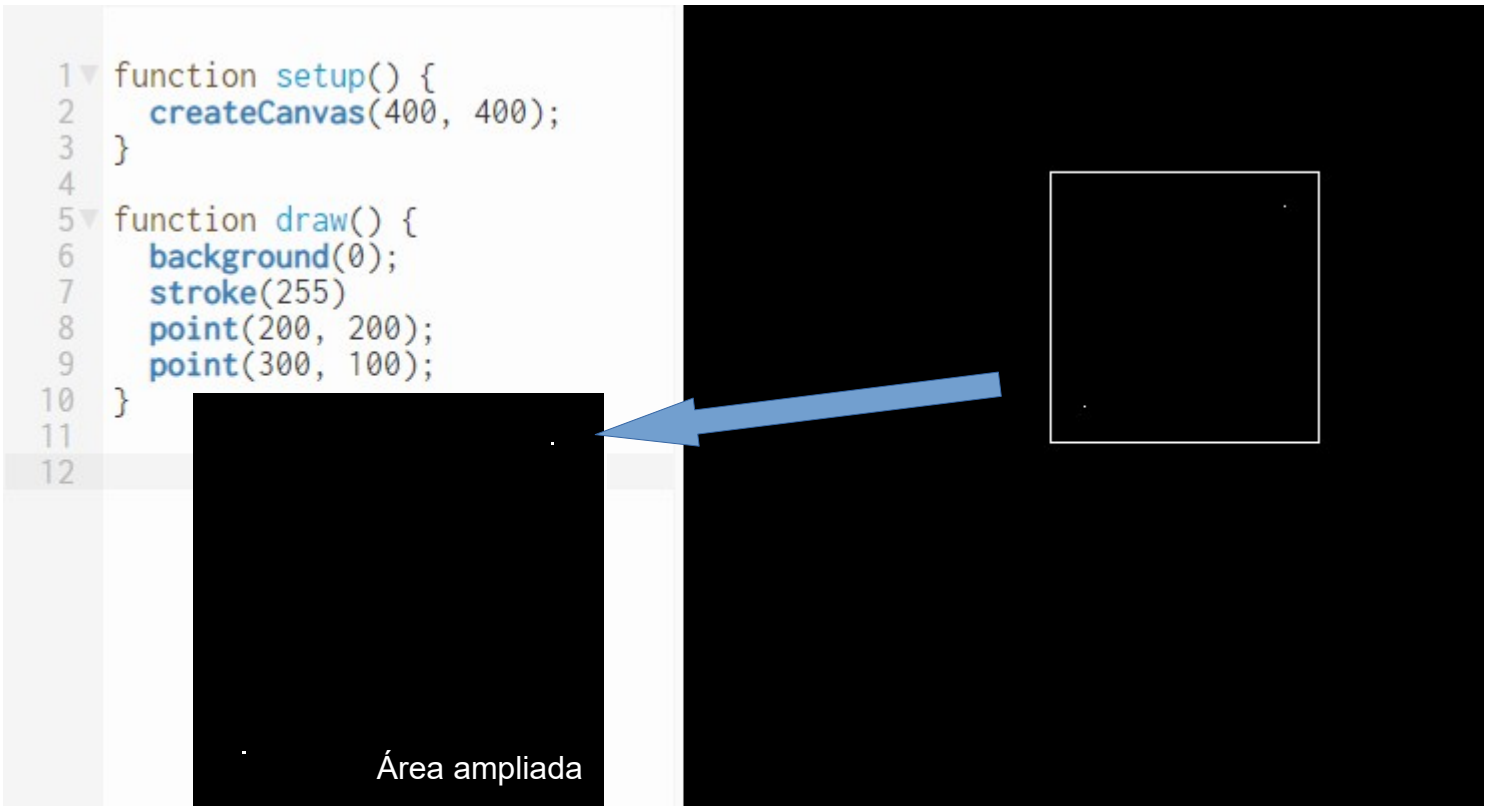
Pontos

Figura 26: Pontos OpenGL



Fonte: Autoria própria

Figura 27: Pontos P5.js



Fonte: Autoria própria

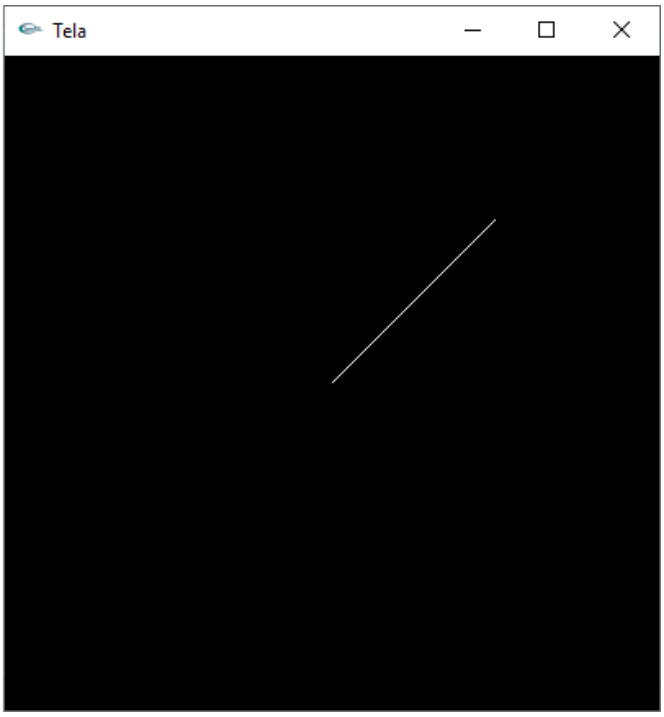
As figuras 26 e 27 mostram que, dentro da função Tela(), são utilizadas as funções com quatro linhas de código para gerar dois pontos – (0, 0) e (0.5, 0.5) - na janela OpenGL. Da mesma forma, o Canvas é gerado com P5.js, porém apenas com duas linhas e a função para definir os pontos com cor branca (255).

Reta

As figuras 28 e 29 apresentam os resultados das execuções que mostram uma reta em OpenGL e P5.js:

Figura: 28: Reta OpenGL

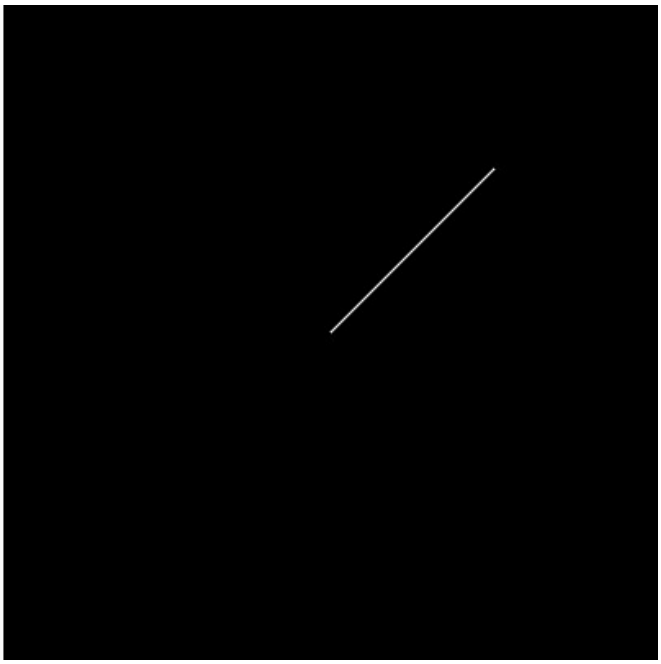
```
4  void Tela(){
5      glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
6      glClear(GL_COLOR_BUFFER_BIT);
7
8      glBegin(GL_LINES);
9      glVertex2d(0.0f, 0.0f);
10     glVertex2d(0.5f, 0.5f);
11     glEnd();
12
13     glFlush();
14 }
15
```



Fonte: Autoria própria

Figura 29: Reta P5.js

```
1  function setup() {
2    createCanvas(400, 400);
3  }
4
5  function draw() {
6    background(0);
7    stroke(255)
8    line(200, 200, 300, 100);
9  }
10
11
```



Fonte: Autoria própria

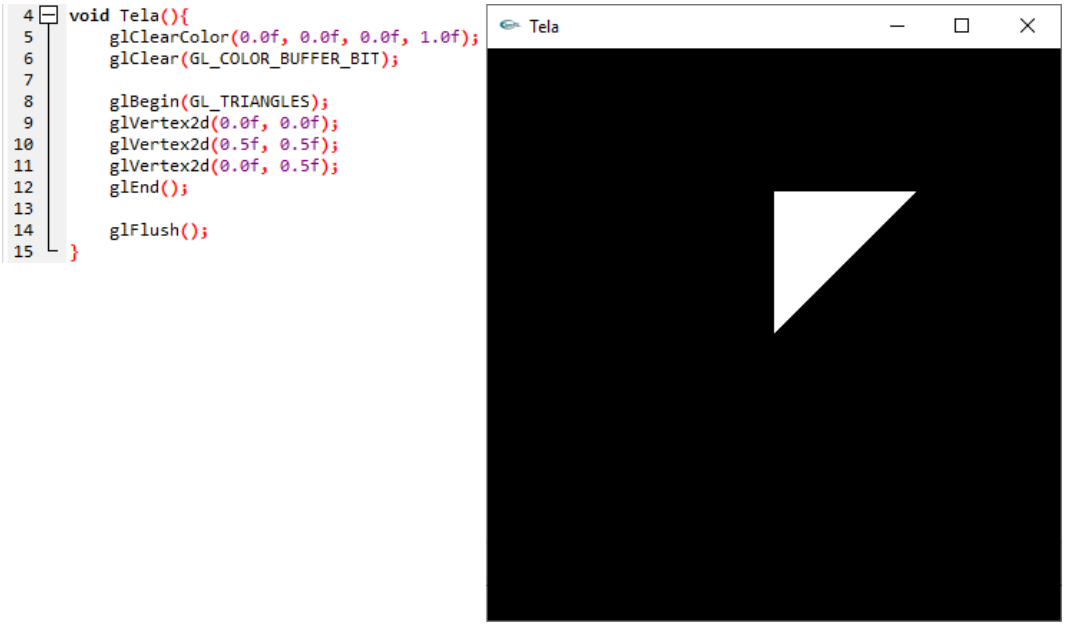
É gerada uma reta definida pelos mesmos dois pontos anteriores, das figuras 26 e 27. O resultado em OpenGL, na Figura 28, é gerado por meio da escrita de

quatro linhas de código. O mesmo é feito no Canvas P5.js com apenas uma linha e a função para coloração branca, como observado na Figura 29.

Triângulo

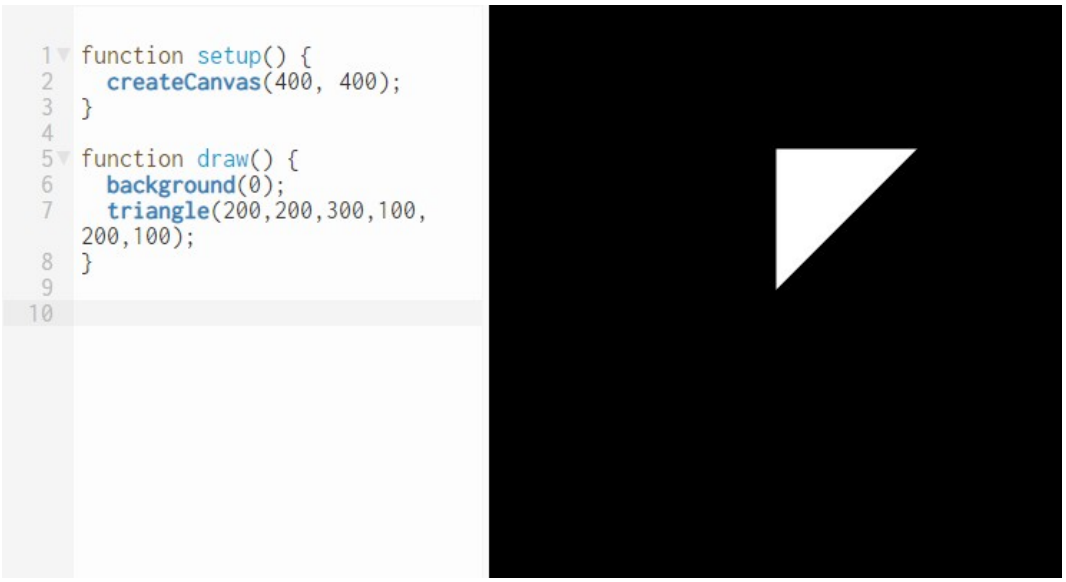
As figuras 30 e 31 apresentam a geração de um mesmo triângulo nas duas ferramentas.

Figura 30: Triângulo OpenGL



Fonte: Autoria própria

Figura 31: Triângulo P5.js



Fonte: Autoria própria

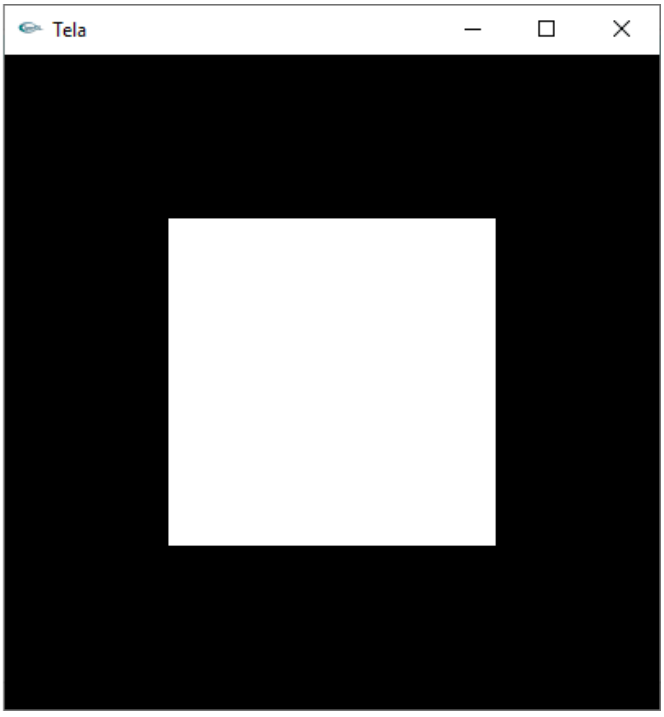
São utilizadas cinco linhas de código para gerar o triângulo na tela OpenGL, definido pelos pontos (0.0, 0.0), (0.5, 0.5) e (0.0, 0.5). Para alcançar o mesmo resultado no Canvas P5.js foi utilizada apenas uma linha de código contendo os três pontos como parâmetros. Obs.: os parâmetros ultrapassaram uma linha por conta do espaço limitado.

Quadrado

São mostradas nas figuras 32 e 33 a execução dos códigos em OpenGL e P5.js que geram um quadrado:

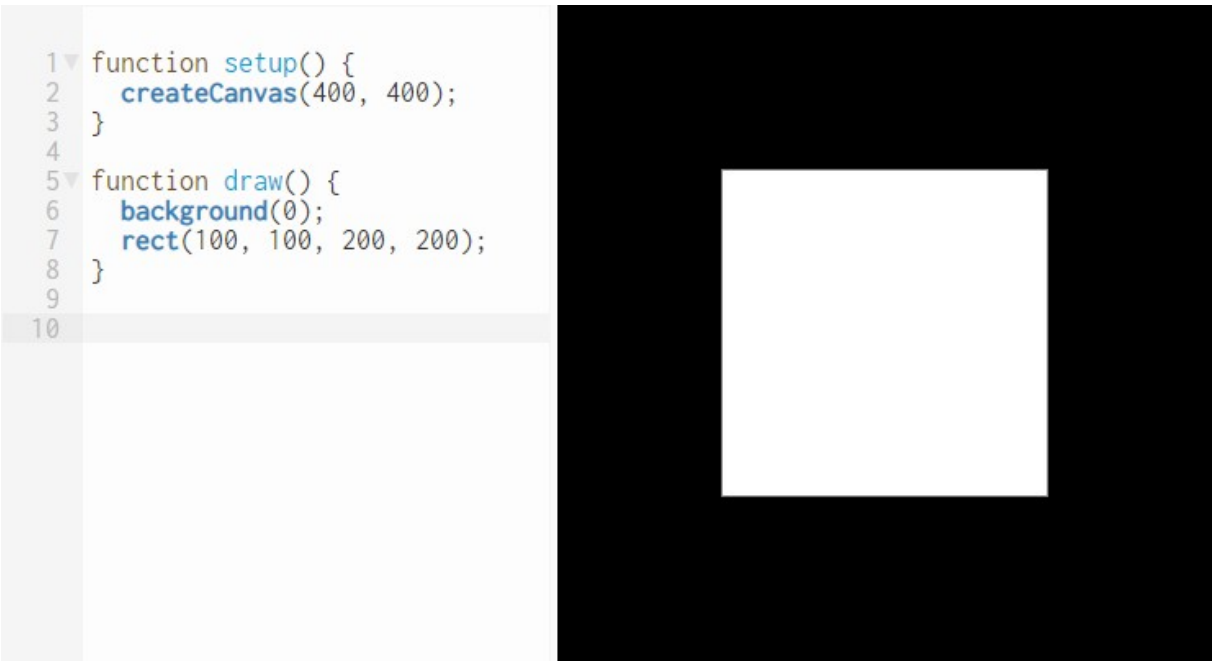
Figura 32: Quadrado OpenGL

```
4  void Tela(){
5      glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
6      glClear(GL_COLOR_BUFFER_BIT);
7
8      glMatrixMode(GL_MODELVIEW);
9      glLoadIdentity();
10
11     glBegin(GL_QUADS);
12     glVertex2d(-0.5f, -0.5f);
13     glVertex2d(0.5f, -0.5f);
14     glVertex2d(0.5f, 0.5f);
15     glVertex2d(-0.5f, 0.5f);
16     glEnd();
17
18     glFlush();
19 }
```



Fonte: Autoria própria

Figura 33: Quadrado P5.js



Fonte: Autoria própria

Na Figura 32 são utilizadas seis linhas de código para gerar o quadrado na tela OpenGL. O quadrado, na Figura 33, é gerado com mesma posição e tamanho no Canvas com a escrita de apenas uma linha de código P5.js.

Círculo

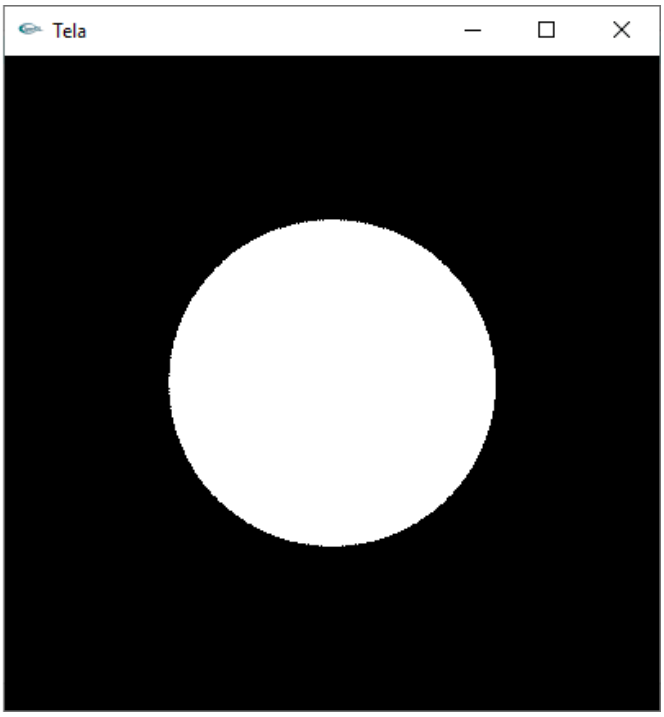
Não há uma função oficial em OpenGL que gere um círculo. Dessa forma o círculo terá de ser desenhado com funções matemáticas em C, fazendo chamada do arquivo *header math.h.* Assim, o cabeçalho do programa em C recebe a adição da seguinte linha:

```
#include <math.h>
```
Código 4

Com as funções matemáticas cos() e sin(), o código para gerar o círculo será o apresentado na Figura 34:

Figura 34: Círculo OpenGL

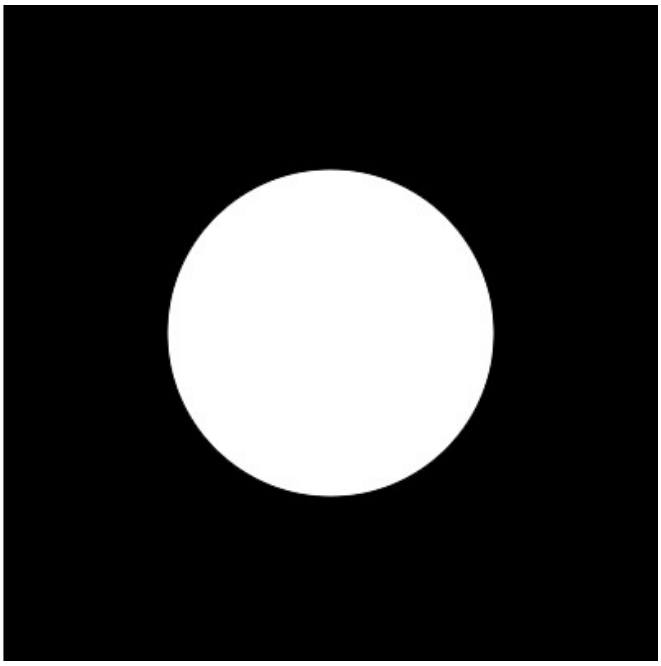
```
11      float x, y, r = 0.5;
12
13      glBegin(GL_POLYGON);
14      for(int i = 0; i <= 360; i++){
15          x = r * cos(i);
16          y = r * sin(i);
17          glVertex2d(x, y);
18      }
19      glEnd();
20
```



Fonte: Autoria própria

Figura 35: Círculo P5.js

```
1  function setup() {
2    createCanvas(400, 400);
3  }
4
5  function draw() {
6    background(0);
7    circle(200, 200, 200);
8  }
9
10
```



Fonte: Autoria própria

Pelo motivo de não haver a função desejada em OpenGL para gerar o círculo foram necessárias várias linhas de código, com utilização de funções matemáticas e um laço de repetição para gerar a forma geométrica na janela, como mostra a Figura 34. O mesmo resultado é alcançado no Canvas P5.js com apenas uma linha de código, visto na Figura 35.

4.3.5 CORES

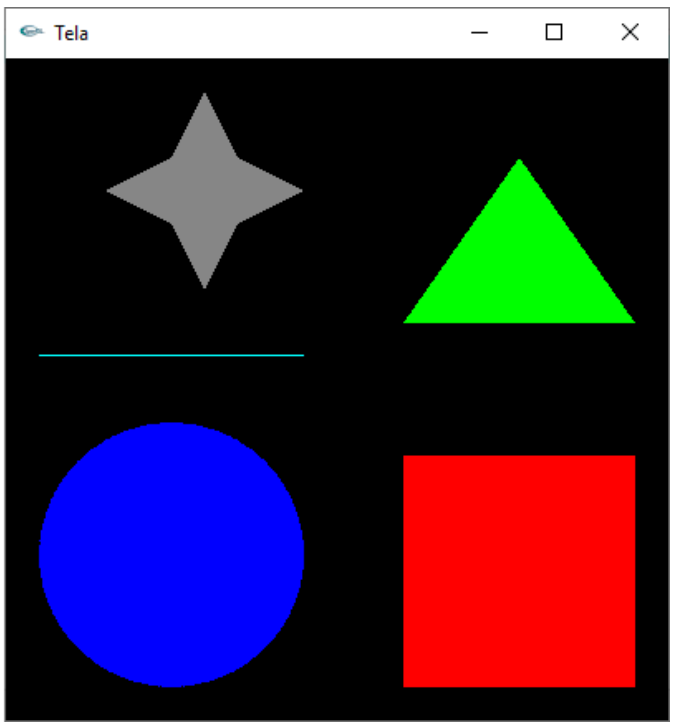
Para comparar as duas tecnologias na utilização de cores foram geradas cinco primitivas em uma mesma tela gráfica: linha, triângulo, quadrado, círculo e um polígono formado com oito pontos. Ambas as tecnologias realizam a coloração com apenas uma linha de código.

A função utilizada em OpenGL é a glColor3f(), com três valores tipo float como parâmetros para a definição da cor em RGB. A função em P5.js, também no sistema RGB com três valores, mas inteiros, é a fill().

As figuras 36 e 37 mostram o código para coloração e os resultados gerados com as aplicações das cores nas primitivas em cada tecnologia:

Figura 36: Cores OpenGL

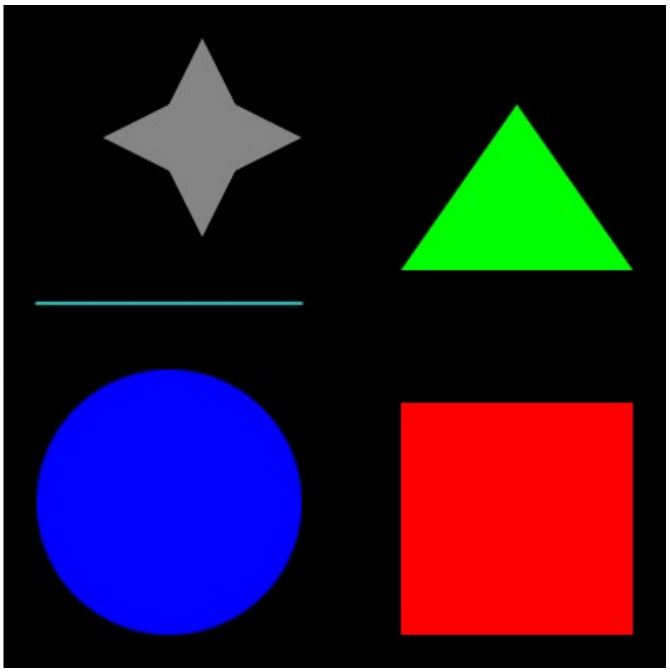
```
glColor3f(0.0f, 1.0f, 1.0f);
//linha
glColor3f(0.5f, 0.5f, 0.5f);
//polígono
glColor3f(0.0f, 1.0f, 0.0f);
//triângulo
glColor3f(1.0f, 0.0f, 0.0f);
//quadrado
glColor3f(0.0f, 0.0f, 1.0f);
//círculo
```



Fonte: Autoria própria

Figura 37: Cores P5.js

```
stroke(0, 255, 255);
//linha
fill(127);
//polígono
fill(0, 255, 0);
//triângulo
fill(255, 0, 0);
//quadrado
fill(0, 0, 255);
//círculo
```



Fonte: Autoria própria

Como as funções para definição de cor de cada primitiva são escritas apenas em uma linha, cada – que antecede a função que gera a primitiva geométrica -, todas as linhas das funções de cor foram extraídas e colocadas juntas para em uma mesma sequência de código para uma visualização mais simples. Dessa forma, cada função de coloração tem, logo abaixo, uma linha de comentário (iniciando com "//") que referencia a primitiva que será colorida.

Como são mostradas nas figuras 32 e 33, as funções de coloração em ambas as ferramentas são escritas com apenas uma linha. Contudo, a função fill(), em P5.js, se apresenta ainda menor, com nome identificador e parâmetros mais simples que a função glColor3f() em OpenGL. Obs.: a função fill() não podem definir a cor da linha. Dessa forma é utilizada a função stroke(). Outro ponto importante é que a função para colorir o polígono em P5.js utiliza apenas um valor como parâmetro, sendo assim, definindo apenas a escala de cinza.

5 CONSIDERAÇÕES FINAIS

Tendo como principal objetivo a abordagem comparativa entre as ferramentas OpenGL e P5.js este trabalho mostra como a esta última proposta é mais simples. Por meio dessa simplicidade de se utilizar a biblioteca P5.js como alternativa para o aprendizado inicial em Computação Gráfica se torna uma boa metodologia a ser aplicada, visto que demonstra as facilidades das funcionalidades da tecnologia, definindo P5.js como método prática mais interessante e fácil na aplicação dos conceitos da disciplina.

Iniciantes e leigos da área, ao estudarem a teoria de Computação Gráfica, terão maior facilidade utilizando as funções da biblioteca P5.js no lugar de OpenGL no momento de aplicar o conteúdo estudado para visualizarem os resultados. Assim a fase de aprendizagem terá um caminho de estudo com o mínimo de dificuldade.

A escolha da biblioteca P5.js como metodologia de aplicação inicial dos conceitos em Computação Gráfica faz guiar a aprendizagem ao facilitar as bases teóricas. Dessa forma, quando for o momento de iniciar o contato com as funcionalidades da ferramenta OpenGL o indivíduo que utilizou P5.js já terá o entendimento base sobre o funcionamento das técnicas de geração, imagens, processamento e análises de imagens no tipo digital. Isso fará com que o contato com OpenGL seja mais produtivo e de bom proveito.

REFERÊNCIAS

AZEVEDO, E. CONCI, A. Computação Gráfica: Geração de Imagens. Editora Elsevier Ltda. Rio de Janeiro, 2003.

BERNERS-LEE, T. CAILLIAU, R. The World-Wide Web. Computer Networks and ISDN Systems, 25(4-5), pp.454-459.

BERNERS-LEE, T et al. World-Wide Web: The Information Universe. Internet Research, 2010.

FLANAGAN, D. JavaScript: The Definitive Guide. 6th Edition. O'Reilly Media, Inc. 2011.

HAVERBEKE, M. Eloquent JavaScript: A Modern Introduction to Programming. 3rd Edition. 2018.

KILGARD, M. J. The OpenGL Utility Toolkit (GLUT) Programming Interface. Silicon Graphics, Inc. 1996.

KUROSE, J. K. ROSS, K. W. Redes de computadores e a internet: uma abordagem top-down. 6ª Ed. Pearson Education do Brasil, Ltda. 2014.

MANSSOUR, I. H. COHEN, M. Introdução à Computação Gráfica. RITA, v. 13, n. 2, p. 43-68, 2006.

MCCARTHY, Lauren; REAS, Casey; FRY, Ben. Getting Started with p5.js. Maker Media Inc. San Francisco, 2016.

MACCARTHY, L. Get Started. 2015. Disponível em: <https://p5js.org/get-started/> Acesso em 2019.

MACARTHY, L. Coordinate System and Shapes. Disponível em: <https://p5js.org/learn/coordinate-system-and-shapes.html> Acesso em 2019.

MIZRAHI, V. V. Treinamento em Linguagem C. Editora McGraw-Hill, Ltda. São Paulo, 1990.

SELLERS, G. WRIGHT JR, R. HAEMEL, N. OpenGL SuperBible: Comprehensive Tutorial and Reference. 7th Edition. Pearson Education, Inc. 2016.

SHILDT, H. C: Completo e Total. 3ª Edição. Pearson Education do Brasil, Ltda. São Paulo, 1997.

SHREINER, D. et. al. OpenGL Programming Guide. 8th Edition. Pearson Education, Inc. 2013.

APÊNDICES

APÊNDICE A – CÓDIGO-FONTE C PARA COLORAÇÃO DE PRIMITIVAS EM OPENGL

```
#include <windows.h>
#include <GL/glut.h>
#include <math.h>

void Tela(){
     glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
     glClear(GL_COLOR_BUFFER_BIT);

     //Reta
     glColor3f(0.0f, 1.0f, 1.0f);
     glBegin(GL_LINES);
     glVertex2d(-0.9, 0.1);
     glVertex2d(-0.1, 0.1);
     glEnd();

     //Polígono
     glColor3f(0.5f, 0.5f, 0.5f);
     glBegin(GL_POLYGON);
     glVertex2f(-0.5, 0.7);
     glVertex2f(-0.7, 0.6);
     glVertex2f(-0.5, 0.5);
     glVertex2f(-0.4, 0.3);
     glVertex2f(-0.3, 0.5);
     glVertex2f(-0.1, 0.6);
     glVertex2f(-0.3, 0.7);
     glVertex2f(-0.4, 0.9);
     glEnd();

     //Triângulo
     glColor3f(0.0f, 1.0f, 0.0f);
     glBegin(GL_TRIANGLES);
     glVertex2d(0.2, 0.2);
     glVertex2d(0.9, 0.2);
     glVertex2d(0.55, 0.7);
     glEnd();
```

```
	//Quadrado
	glColor3f(1.0f, 0.0f, 0.0f);
	glBegin(GL_QUADS);
	glVertex2d(0.2, -0.9);
	glVertex2d(0.9, -0.9);
	glVertex2d(0.9, -0.2);
	glVertex2d(0.2, -0.2);
	glEnd();

	float x, y, r = 0.4;
	glColor3f(0.0f, 0.0f, 1.0f);

	//Círculo
	glBegin(GL_POLYGON);
	for(int i = 0; i <= 360; i++){
		x = -0.5 + r * cos(i);
		y = -0.5 + r * sin(i);
		glVertex2d(x, y);
	}
	glEnd();

	glFlush();
}

int main(int argc, char** argv){
	glutInit(&argc, argv);
	glutInitWindowSize(400, 400);
	glutInitWindowPosition(50, 50);
	glutCreateWindow("Tela");
	glutDisplayFunc(Tela);
	glutMainLoop();

	return 0;
}
```

APÊNDICE B – CÓDIGO-FONTE JAVASCRIPT PARA COLORAÇÃO DE PRIMITIVAS EM P5.JS

```
function setup() {
  createCanvas(400, 400);
}

function draw() {
  background(0);

  //Reta
  stroke(0, 255, 255);
  line(20, 180, 180, 180);

  noStroke();

  //Poligono
  fill(127)
  beginShape();
  vertex(100, 60);
  vertex(60, 80);
  vertex(100, 100);
  vertex(120, 140);
  vertex(140, 100);
  vertex(180, 80);
  vertex(140, 60);
  vertex(120, 20);
  endShape();

  //Triângulo
  fill(0, 255, 0);
  triangle(240, 160, 380, 160, 310, 60);

  //Quadrado
  fill(255, 0, 0);
  rect(240, 240, 140, 140);

  //Círculo
  fill(0, 0, 255);
  circle(100, 300, 160, 160);

}
```