

GRADE: AMBIENTE GRÁFICO DE DESENVOLVIMENTO PARA ENSINO DE COMPUTAÇÃO GRÁFICA

Natália Sens Weise, Dalton Solano dos Reis – Orientador

Curso de Bacharel em Ciência da Computação
Departamento de Sistemas e Computação
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brasil
nweise@furb.br, dalton@furb.br

Resumo: Este artigo apresenta uma aplicação para uso na web, com o objetivo de ser disponibilizada como material de apoio nas aulas da disciplina de Computação Gráfica no curso de Ciência da Computação na universidade Fundação Universidade Regional de Blumenau. O projeto foi desenvolvido em Unity, juntamente com a IDE Visual Studio, fazendo uso da linguagem de programação C# para implementação. Para verificar se a ferramenta estava apta para ser usada como material didático, foi aplicado um questionário aos alunos da disciplina, o qual perguntava sobre como os alunos avaliavam seus conhecimentos dos assuntos abordados relacionados à aula antes de usar a aplicação e depois, questionando o quanto eles acham que o uso dela ajudaria no aprendizado dos assuntos abordados em aula. A partir das respostas obtidas, verificou-se que os alunos acreditam que a aplicação ajudará na fixação e revisão do conteúdo, em razão do fato de existir um retorno visual frente a cena em construção por parte do aluno, além de existir exercícios dentro da aplicação para validação do conteúdo. Com isso, verifica-se que todos os objetivos foram alcançados.

Palavras-chave: Material didático. Computação gráfica. Unity. Validação de conteúdo. **Visualização 3D.**

1 INTRODUÇÃO

Conforme dito por Manssour e Cohen (2006, p. 1), a Computação Gráfica (CG) “é uma área da Ciência da Computação que se dedica ao estudo e desenvolvimento de técnicas e algoritmos para a geração (síntese) de imagens através do computador.”. Para realizar as devidas transformações nas imagens, é preciso fazer uso da matriz de transformação, que é responsável por proporcionar escala, rotação e translação aos objetos gráficos da cena. Também é necessário o conhecimento de outros assuntos dentro dessa temática, sendo eles: grafo de cena, objetos gráficos, transformações geométricas homogêneas (matriz de transformação), câmera sintética e iluminação. Contudo, ainda é preciso que se tenha uma boa fundamentação teórica em geometria, visto que os conceitos de CG se baseiam nessa área da matemática (Azevedo; Conci; Vasconcelos, 2022).

Como Settimy e Bairral (2020) observaram, os alunos possuem dificuldade na abstração do espaço 3D em razão do ensino pouco aprofundado na área da Geometria. Segundo Settimy e Bairral (2020, p. 3), “a Geometria é um campo fértil para perceber e entender as formas geométricas presentes em nosso cotidiano, sendo possível desenvolver habilidades importantes como a experimentação, representação, descrição e argumentação [...]”, sendo fundamental para o entendimento de CG. Dentre as diversas ferramentas de apoio existentes, uma que se destaca no âmbito da Geometria é o Geogebra 3D, que permite criar objetos 3D e manipular os valores de suas propriedades, o que contribui muito para o aprendizado da matéria como visto por Fassarella e Rocha (2018).

Outro material de apoio que se sobressai é o VisEdu-CG, construído por Buttenberg (2020) com o objetivo de auxiliar os alunos da Fundação Universidade Regional de Blumenau (FURB) no entendimento dos assuntos abordados na disciplina de CG do curso de Ciência da Computação. O projeto apresenta uma tela dividida em quatro seções: Fábrica de Peças, na qual o usuário pega as peças para programar; Renderer, em que o usuário deposita as peças que coletou na Fábrica de Peças; Ambiente Gráfico, em que é possível visualizar os eixos, grade e peças colocados em cena; e Visualizador, que mostra o resultado da execução do que foi projetado pelo usuário. Todavia, nem todos os objetivos propostos por Buttenberg (2020) foram concluídos. Algumas funcionalidades propostas, como os objetos Polígono e Spline e algumas propriedades da câmera não foram implementadas. Além disso, o tutorial desenvolvido ficou limitado a poucas funções básicas.

Sendo assim, esse projeto visa auxiliar os alunos de CG a entenderem os assuntos abordados em aula continuando com o desenvolvimento do antigo VisEdu-CG (Buttenberg, 2020), implementando as funcionalidades faltantes e trazendo novas, como a interface com mudança de tema (claro e escuro) para o usuário escolher o que mais lhe agrada à vista, além de exercícios para fixação do conteúdo e um tutorial mais completo. Portanto, o objetivo principal deste trabalho é disponibilizar uma nova versão do VisEdu-CG, agora chamado de ambiente GRÁFICO de Desenvolvimento para Ensino de computação gráfica (GRADE), para ser utilizado na disciplina de Computação Gráfica na forma de material de apoio. E seus objetivos específicos são: validar se o ambiente desenvolvido consegue representar objetos gráficos 3D definidos em um Grafo de Cena, validar se estes objetos gráficos 3D podem ser manipulados por Transformações Geométricas e

avaliar se a utilização de exercícios, usando o ambiente desenvolvido, pode auxiliar no entendimento dos assuntos abordados em aula.

2 REVISÃO BIBLIOGRÁFICA

Nessa seção serão descritos os principais conceitos que servirão como base para esse trabalho (abstração do espaço 3D, computação gráfica e fundamentos para criar um tutorial - subseção 2.1), além de uma subseção sobre o projeto anterior (subseção 2.2) e sobre os trabalhos correlatos (subseção 2.3).

2.1 CONCEITOS, TÉCNICAS E/OU FERRAMENTAS

Nessa subseção serão descritos os principais conceitos que servirão como base para esse projeto: abstração do espaço 3D (subseção 2.1.1), Computação Gráfica (subseção 2.1.2) e fundamentos para criar um tutorial (subseção 2.1.3).

2.1.1 ABSTRAÇÃO DO ESPAÇO 3D

Segundo Azevedo, Conci e Vasconcelos (2022, p. 35), “[...] a abstração matemática dita Sistema de Coordenadas é explorada pela Computação Gráfica como ferramenta que permite escolher e alterar a representação de objetos gráficos de maneira que for mais conveniente a cada operação de processamento visual.”. Como Settimy e Bairral (2020) observaram, os alunos possuem dificuldade na abstração do espaço 3D pelo fato do ensino básico não abordar a geometria de forma mais clara e aprofundada. Segundo Settimy e Bairral (2020, p. 3), “[...] a Geometria é um campo fértil para perceber e entender as formas geométricas presentes em nosso cotidiano, sendo possível desenvolver habilidades importantes como a experimentação, representação, descrição e argumentação.”.

Entre as ferramentas de auxílio para aprendizado de Geometria se tem o Geogebra 3D. Com o Geogebra 3D, os usuários podem construir qualquer forma geométrica 3D e manipular seus valores de escala, rotação e translação. Assim, utilizando esse espaço de visualização 3D, contribui mais com o aprendizado, conforme observado por Fassarella e Rocha (2018). Ao considerar que a Computação Gráfica se baseia fundamentalmente em Geometria, esta se torna indispensável para o aprendizado de CG.

Conforme observado por Settimy e Bairral (2020) e Azevedo, Conci e Vasconcelos (2022) isso se dá pelo fato de que, além de abstrair o espaço 3D, também é necessário entender o conceito de matriz de transformação homogênea e aplicá-la a objetos gráficos da cena, sendo necessário conhecimentos da área da geometria

2.1.2 COMPUTAÇÃO GRÁFICA

Conforme dito por Manssour e Cohen (2006, p. 1), computação gráfica “[...] é uma área da Ciência da Computação que se dedica ao estudo e desenvolvimento de técnicas e algoritmos para a geração (síntese) de imagens através do computador.”. E, como visto por Azevedo, Conci e Vasconcelos (2022, p. 183), “isso se faz a partir da descrição da geometria dos objetos [...], dos materiais associados às suas superfícies [...], das fontes de luz e do modelo de representação da iluminação adotado, da definição de uma câmera virtual que estabelece a posição de observação de cena, [...]”. Sendo assim, para maior entendimento do assunto, é necessário o conhecimento de outros conceitos dentro dessa temática, sendo eles: grafo de cena, objetos gráficos, transformações geométricas homogêneas, câmera sintética e iluminação.

Conforme Silva, Raposo e Gattas (2004, p. 3), “[...] grafos de cena são ferramentas conceituais para representação de ambientes virtuais tridimensionais nas aplicações de computação gráfica.”. Isso significa que o grafo é uma espécie de mapa para a cena construída, mostrando quais objetos gráficos fazem parte dela, quais objetos possuem filhos e quais suas características (cor, textura, posicionamento etc.). Azevedo, Conci e Vasconcelos (2022, p. 183) também afirmam que “[...] é comum que os objetos sejam descritos como malhas poligonais, compostas por conjuntos de vértices e arestas.”. Sendo assim, objetos gráficos são formas compostas por coordenadas que são mapeadas e representadas no mundo gráfico.

Para entender o conceito de transformações geométricas homogêneas, primeiro é preciso conceituar transformação em si. Conforme dito por Azevedo, Conci e Vasconcelos (2022, p. 52), transformação “[...] é qualquer função f que realiza um mapeamento de um conjunto de entrada, dito domínio, em um conjunto de saída, dito contradomínio.”. Dito isso, transformações geométricas homogêneas são funções que alteram o valor inicial das coordenadas e são aplicadas igualmente a todos os pontos de um objeto gráfico. Dentre as transformações existentes, destacam-se: rotação, escalamento e translação. A função de rotação é responsável por rotacionar os pontos, podendo ser no sentido horário ou anti-horário, para uma nova posição a partir da sua origem. Escalamento seria a transformação usada para alterar o tamanho de um objeto gráfico, podendo tanto aumentar quanto diminuir sua escala. Já a translação, é a transformação necessária para mudar a posição de um objeto a partir de sua origem. Esses três tipos de transformações são comumente usados em conjunto para se obter o resultado desejado e, por isso, acabam sendo complementares umas das outras (Azevedo; Conci; Vasconcelos, 2022).

Uma câmera sintética, também conhecida como câmera virtual, “[...] define um ponto de vista sob o qual a cena será visualizada e com isso cria uma representação no sistema de Computação Gráfica para o observador da cena.” (Azevedo; Conci; Vasconcelos, 2022, p. 38). Dessa forma, ela é necessária para a visualização dos objetos gráficos na cena. Vale ressaltar que apenas serão vistos em cena os objetos gráficos alinhados com o volume de visão da câmera, que seria toda a área visível a partir da sua localização. Para dispor devidamente a câmera, é preciso ter sua localização e orientação no espaço, o tipo de projeção que realizará e como ela interpretará os dados das imagens que serão visualizadas (Azevedo; Conci; Vasconcelos, 2022).

Como a câmera fica na cena junto com os outros objetos, ela também é um objeto gráfico, sendo preciso definir suas coordenadas e sua orientação (para onde ela está olhando). A projeção trata sobre como o objeto gráfico será visto em cena, podendo ser do tipo paralela, que mantém a linha de projeção seguindo os pontos de forma paralela entre si (muito usado em projeção 2D). Ou sob perspectiva, fazendo com que objetos mais próximos apareçam maiores do que os mais distantes do ponto de visualização (projeção mais usada no 3D). A forma em que a câmera interpretará se relaciona com os outros dois aspectos anteriores. Para projetar a imagem, é preciso saber seu centro (para inserir no lugar correto) e sua escala (para ficar do tamanho desejado), para então mostrá-la de forma adequada (Azevedo; Conci; Vasconcelos, 2022).

Para que os objetos gráficos sejam percebidos em cena, é preciso a presença de iluminação para a percepção de suas cores e texturas. Existem, por exemplo, quatro tipos de luz: a ambiente, a direcional, a pontual e a holofote. A luz ambiente é a mais comum e simples de se utilizar. Ela funciona como uma luz global, iluminando a cena de forma igualitária, permitindo que todos os objetos sejam visualizados, mas sem produzir grande efeitos de reflexão e sombreamento. A luz direcional é a utilizada para simular a luz solar: ela vem de um ponto em específico e segue a angulação, traçando raios paralelos de luz entre si. É importante lembrar que esse tipo de iluminação considera que todos os raios emitem quantidade equivalente de luz. A luz pontual é um ponto no espaço que ilumina em todas as direções e apresenta intensidades de luz diferentes conforme afastamento da origem. Por se tratar de um ponto, é usada para representar lâmpadas, explosões, entre outros tipos de objetos com pontos luminosos. Por fim, a luz holofote, como o próprio nome diz, é a luz proveniente de uma lâmpada do tipo holofote, iluminando apenas a região abrangente pelo seu ângulo de abertura, reduzindo de intensidade conforme afastamento (Azevedo; Conci; Vasconcelos, 2022).

2.1.3 FUNDAMENTOS PARA CRIAR UM TUTORIAL

Como observado por Cieślak (2021), um tutorial é o primeiro contato do usuário com a aplicação, onde terá uma demonstração do que se trata e de como é seu funcionamento. A partir disso, o usuário aprende regras e comportamentos da ferramenta, ganhando certa familiaridade com ela. Sendo assim, com base em relatos de outros desenvolvedores e em seus próprios, Cieślak (2021) desenvolveu uma lista de dicas para montar o tutorial perfeito e garantir que usuário absorva o máximo de informação possível sem perder o interesse.

A primeira dica se refere a fazer com que o tutorial seja parte do jogo/aplicação, para que o usuário tenha a impressão de que já começou a jogar e que se dedique e fique entusiasmado com isso. A segunda dica é sobre introduzir as funções aos poucos. Assim, o usuário absorverá melhor como cada função age e diminuirá suas chances de esquecer algo que o tutorial tenha mostrado (Cieślak, 2021).

A terceira dica trata sobre ações: um tutorial deve ser feito de ações, e não texto. Isso quer dizer que, para garantir que o usuário aprenda o que se está ensinando, é preciso que ele faça aquilo que se pede. Tutoriais somente textuais acabam sendo ignorados e esquecidos mais facilmente. Além disso, como dito por Cieślak (2021), dar ao usuário o que fazer dá a sensação de controle para ele, o que é muito bom para prender ainda mais o interesse. A dica quatro acaba se relacionando com a anterior: os tutoriais devem ter entre cinco e nove etapas. Um tutorial com muitas etapas ou muito texto acaba ficando cansativo, o que desmotiva o usuário e faz com que ele não lembre direito do que lhe foi apresentado (Cieślak, 2021).

A dica cinco é: seja breve. Além da quantidade de texto já mencionada anteriormente, um tutorial deve conter apenas as funções necessárias para que o usuário possa se virar. Exagerar nas informações apresentadas só o deixará confuso. A sexta dica trata sobre recursos visuais, indicando que, caso seja necessário, “aponte o dedo”. Isso significa que, no tutorial, o desenvolvedor pode fazer uso de setas e mapas para ajudar o usuário a entender o que ele tem que fazer para seguir na direção correta (Cieślak, 2021).

A sétima dica fala sobre armazenar o que foi apresentado, isto é, permitir que o usuário possa rever aquilo que já lhe foi mostrado, como por exemplo ter um painel de informações para serem acessadas quando o usuário desejar. Por fim, a última dica trata sobre senso de significado. Isso quer dizer que tudo o que foi apresentado ao usuário deve fazer sentido (ter uma utilidade) e despertar o interesse do usuário, para que ele decida continuar com o jogo (Cieślak, 2021).

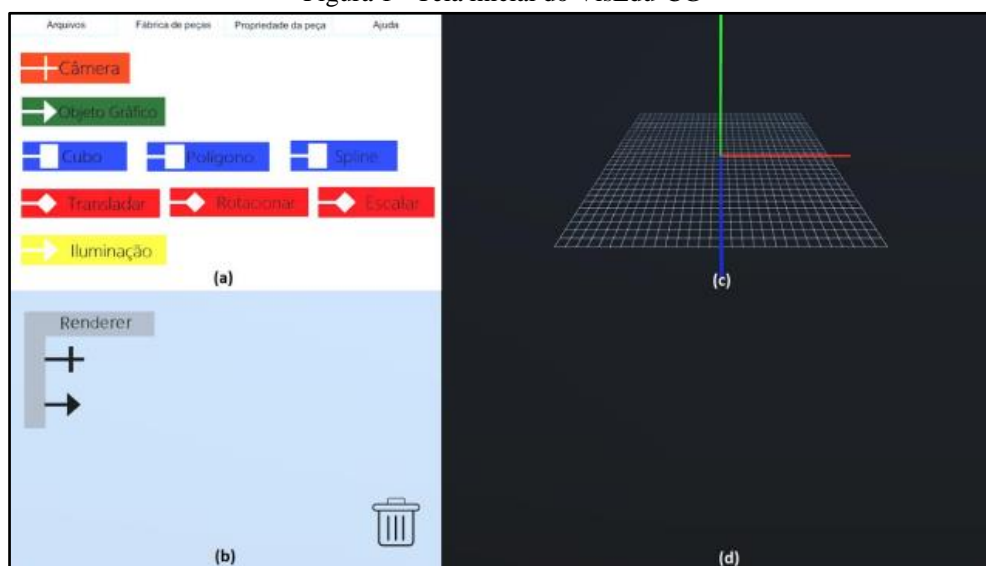
2.2 VERSÃO ANTERIOR DO SOFTWARE

Ao longo dos anos, o VisEdu-CG já passou por algumas versões: tendo as duas primeiras em C++ (Araújo, 2012; Schramm, 2012), as três seguintes em Three.js (Nunes, 2014; Montibeler, 2014; Koehler, 2015) e a atual em Unity

(Buttenberg, 2020), cuja tecnologia se manteve nessa nova versão desenvolvida neste projeto. Inicialmente chamado de Adubo e posteriormente de VisEdu-CG, a ferramenta surgiu com o objetivo de auxiliar os alunos da disciplina de Computação Gráfica do curso de Ciência da Computação da FURB a compreender melhor os temas abordados em aula, os principais sendo: grafo de cena, objetos gráficos, transformações geométricas homogêneas, câmera sintética e iluminação.

Buttenberg (2020) projetou a última versão do antigo nomeado VisEdu-CG em Unity, na versão 2018.2.6f1, a fim de aprimorar para uma ferramenta mais popular. Ao inicializá-la, o usuário pode optar por um tutorial de sete passos para aprender a usar a ferramenta. Nesta ferramenta são apresentadas quatro seções de tela distintas: Fábrica de Peças (Figura 1 (a)), na qual o usuário pega os blocos para programar; Renderer (Figura 1 (b)), em que o usuário deposita as peças que coletou na fábrica; Ambiente Gráfico (Figura 1 (c)), no qual é possível visualizar os eixos, grade e objetos colocados em cena; e Visualizador (Figura 1 (d)), que mostra o resultado da execução do que foi programado pelo usuário sem a presença da grade e dos eixos de orientação presentes na tela de Ambiente Gráfico.

Figura 1 - Tela inicial do VisEdu-CG



Fonte: Buttenberg (2020).

A Fábrica de Peças (Figura 1 (a)) apresenta nove diferentes tipos de peças, sendo eles: Câmera, Objeto Gráfico, Cubo, Polígono, Spline, Transladar, Rotacionar, Escalar e Iluminação. O Objeto Gráfico, Cubo, Polígono e Spline são formas geométricas para dispor no espaço gráfico. As peças Transladar, Rotacionar e Escalar são responsáveis pela matriz geométrica (transformações geométricas homogêneas), podendo mudar a posição, orientação e o tamanho no espaço do objeto em que forem aplicados. A Câmera e Iluminação são fundamentais para o funcionamento da aplicação, visto que a Câmera possibilitará a visualização do resultado e a Iluminação permitirá que os objetos sejam vistos em cena (Buttenberg, 2020).

Ao selecionar a peça desejada, o usuário deve arrastá-la até o Renderer (Figura 1 (b)), encaixando conforme formato da peça. Ao inserir um objeto geométrico, é possível adicionar tanto a iluminação quanto os objetos da matriz geométrica. Ao selecioná-los, é possível excluir o objeto ou editar suas propriedades, que aparecerem no canto superior esquerdo. Enquanto o aluno vai adicionando as peças, é possível pré-visualizar o resultado na tela de Ambiente Gráfico (Figura 1 (c)), podendo fazer alterações nos valores de Transladar, Rotacionar e Escalar para obter o resultado desejado, que é apresentado na tela Visualizador (Figura 1 (d)) (Buttenberg, 2020).

Ao concluir o projeto, Buttenberg (2020) demonstra que os objetivos específicos foram parcialmente cumpridos, visto que algumas funcionalidades propostas, os objetos Polígono e Spline e algumas propriedades da câmera, não foram implementados. Além disso, o tutorial desenvolvido ficou limitado a poucas funções básicas. Ademais, o projeto não possui a funcionalidade de se fazer exercícios de treinamento do conteúdo relacionado a Computação Gráfica.

2.3 TRABALHOS CORRELATOS

Essa subseção expõe três trabalhos selecionados com características em comum ao que se desenvolveu, os quais são apresentados em quadros. O Quadro 1 traz um jogo desplugado para ensinar pensamento computacional às crianças, proposto por Rodrigues, Gomes e Carneiro (2022). O Quadro 2 aborda o jogo GeNiAl desenvolvido por Barros, Sousa e Viana (2022), que busca ensinar a tabela periódica para estudantes do ensino superior. O Quadro 3 apresenta uma plataforma com jogos que ensinam astronomia projetada por Siedler *et al.* (2022).

Quadro 1 – Trabalho Correlato 1

Referência	Rodrigues, Gomes e Carneiro (2022)
Objetivos	Trazer o Scratch para meio físico, a fim de ajudar os alunos de escolas sem acesso à tecnologia e internet a desenvolverem o pensamento computacional durante cenário pandêmico.
Principais funcionalidades	Encaixar blocos para atingir o objetivo de cada tarefa proposta.
Ferramentas de desenvolvimento	Blocos de materiais acessíveis e coloridos (exemplo: EVA).
Resultados e conclusões	Os alunos conseguiram concluir as atividades e adquiriram o conhecimento desejado. Contudo, os alunos levaram mais tempo por não terem apoio presencial dos professores para tirar dúvidas.

Fonte: elaborado pela autora.

Quadro 2 - Trabalho Correlato 2

Referência	Barros, Sousa e Viana (2022)
Objetivos	Ajudar estudantes de ensino superior, que estejam na área das ciências ou que apenas tenham interesse no assunto, a aprender sobre a tabela periódica.
Principais funcionalidades	Nele, existe um quiz e mais três trilhas para treinar diferentes conhecimentos da área, sendo que cada uma delas está ligada a um objetivo proposto (Erro! Fonte de referência não encontrada.): Germânio (Ge), com exercícios de agilidade para memorizar nome, símbolo e número atômico do elemento; Níquel (Ni), um minijogo da memória com o objetivo de relacionar elementos químicos com artigos do cotidiano; e Alumínio (Al), com atividades de lógica que buscam relacionar a posição do elemento na tabela com suas características.
Ferramentas de desenvolvimento	O jogo foi desenvolvido para web em Next.js e React.js.
Resultados e conclusões	Com base nas respostas obtidas, notou-se que a aplicação contribuiu com o fortalecimento e aprimoramento dos saberes dos alunos, visto que obtiveram alto desempenho nas atividades do jogo. Os pesquisados também informaram que obtiveram sentimento de satisfação ao concluir as tarefas pré-estabelecidas com êxito.

Fonte: elaborado pela autora.

Quadro 3 - Trabalho Correlato 3

Referência	Siedler <i>et al.</i> (2022)
Objetivos	Para obter o aprimoramento das técnicas de ensino sobre astronomia em sala de aula, Siedler <i>et al.</i> (2022) criaram uma plataforma com jogos para auxiliar os professores a ensinarem o tema de forma mais interessante aos alunos, promovendo engajamento.
Principais funcionalidades	Primeiro jogo: apresenta dois módulos, Professor e Aluno. Em Professor, o docente pode inserir mais informações sobre o tema, aplicar questionários, coletar dados de desempenho dos discentes, entre outras funcionalidades. No modo Aluno, o estudante pode visualizar as informações postadas clicando em cada um dos planetas alinhados na tela, além de realizar questionários e salvar em arquivo no formato PDF tanto o conteúdo sobre planetas quanto as questões com suas respostas registradas. Segundo jogo: apresenta dinâmica de fases. Cada fase é um planeta e, para ganhar o jogo, o usuário deve viajar de planeta em planeta, começando pelo Sol e terminando o trajeto em Netuno. Para alcançar ao próximo astro, o aluno deve completar tarefas e ao chegar no destino pode acessar informações sobre aquele planeta. Terceiro jogo: o usuário lê com a câmera do smartphone com sistema Android cartas que funcionam como marcadores. Ao ler a imagem, o aplicativo projeta o respectivo astro em 3D na tela. Caso o usuário não possua os cartões, pode visualizar as imagens em 2D (sem a experiência de Realidade Aumentada).
Ferramentas de desenvolvimento	Primeiro jogo: HyperText Markup Language 5 (HTML5), JavaScript, NodeJS e MongoDB. Segundo jogo: Unity. Terceiro jogo: Unity e Vuforia.
Resultados e conclusões	Ao testar a plataforma com alunos do quinto ano, notou-se maior interesse e aprendizado do conteúdo. Além disso, as crianças fizeram uso de trabalho em equipe no segundo jogo, como estratégia para passar de fase.

Fonte: elaborado pela autora.

3 DESCRIÇÃO DO SOFTWARE

Nesta seção serão abordados a especificação, a qual apresentará a lista de requisitos e os diagramas, e a implementação, que mostrará de forma aprofundada o desenvolvimento do projeto em questão.

3.1 ESPECIFICAÇÃO

Esta subseção irá abordar sobre os diagramas de classes, apresentando-os e ilustrando-os, e sobre os requisitos, listando-os. É importante frisar que nem todas as funcionalidades e atributos das classes foram ilustrados, visando uma visualização mais clara do diagrama. Sendo assim, apenas as funções e atributos principais foram descritas (em caso de códigos fonte muito longos).

3.1.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Os requisitos do projeto foram divididos em Requisitos Funcionais (RFs), que definem o que a ferramenta deve permitir que o usuário faça (Quadro 4), e Requisitos Não Funcionais (RNFs), que definem como a ferramenta deve realizar tais funcionalidades (Quadro 5).

Quadro 4– Requisitos funcionais da ferramenta

RF01	permitir que o usuário possa seguir um tutorial para auxiliar o entendimento da ferramenta
RF02	permitir que o usuário possa arrastar as peças e editar suas informações conforme for desejado
RF03	permitir que o usuário possa mexer no tema da aplicação (modo claro ou escuro) conforme melhor lhe agradar
RF04	permitir que o usuário possa fazer uso da câmera com todas as suas propriedades.
RF05	permitir que o usuário possa realizar exercícios pré-definidos, a fim de treinar seus conhecimentos adquiridos
RF06	permitir que o usuário saiba se acertou o exercício de treinamento ou não e, caso tenha acertado, qual foi a porcentagem desse acerto
RF07	permitir que o usuário possa importar e exportar a cena criada com as peças em formato JavaScript Object Notation (JSON)

Fonte: elaborado pela autora.

Quadro 5– Requisitos não funcionais da ferramenta

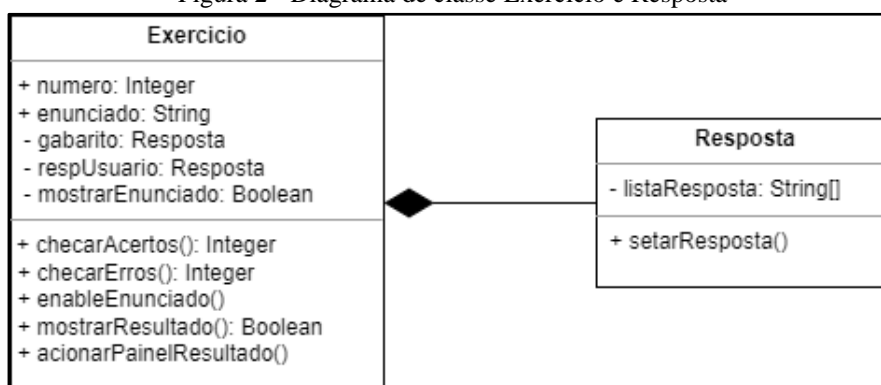
RNF01	utilizar o motor de jogos Unity em conjunto com a Integrated Development Environment (IDE) Visual Studio
RNF02	utilizar a linguagem de programação C# para implementação
RNF03	ser desenvolvido para plataforma web

Fonte: elaborado pela autora.

3.1.2 DIAGRAMAS

Para facilitar o processo de codificação, foram desenvolvidos diagramas de classe e diagramas de atividade, presentes no APÊNDICE A. A Figura 2 ilustra a relação das classes `Exercicio` e `Resposta`, usadas para a funcionalidades de exercício da ferramenta. A classe `Exercicio` terá acesso à `Resposta`, visto que é a responsável por checar quanto o usuário acertou do exercício. Uma terceira classe `Resultado` poderia ter sido construída, mas como a ordem dos blocos e suas propriedades já ficam salvas em variáveis globais, sua criação seria redundante.

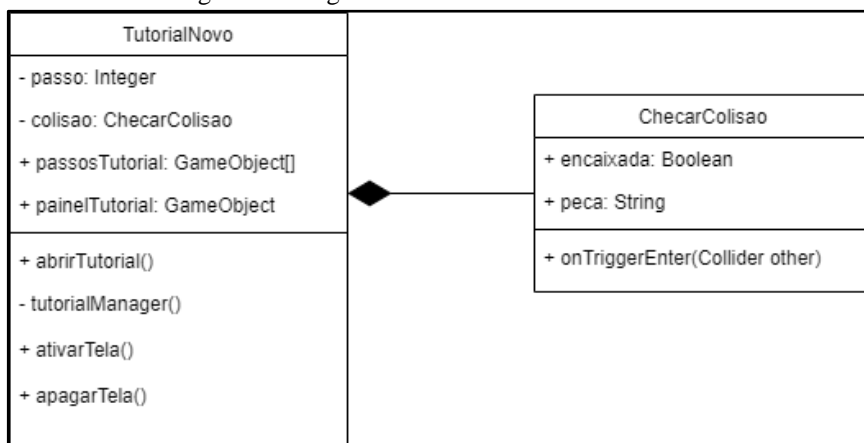
Figura 2 - Diagrama de classe `Exercicio` e `Resposta`



Fonte: elaborado pela autora.

A Figura 3 ilustra a relação da classe `TutorialNovo` e `ChecarColisao`. A `TutorialNovo` é responsável pela troca de imagens do passo a passo e, por conseguinte, por checar se a etapa foi concluída ou não. Para isso, ela precisa do auxílio da `ChecarColisao`, que verifica se a peça que foi encaixada é "a da vez". Para facilitar esse processo de verificar a peça, foram colocadas *tags* com nomes representando-as (exemplo: peça `Cubo` possui a *tag* `Cubo`); assim, a `TutorialNovo` ficou com um código mais legível: verificando se está encaixada e se o nome da peça condiz com o correto (isso na maioria das etapas).

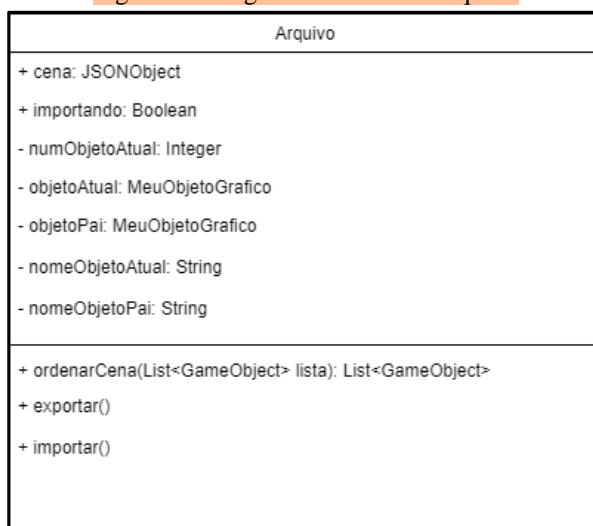
Figura 3 - Diagrama de classes do Novo Tutorial



Fonte: elaborado pela autora.

A Figura 4 representa a classe Arquivo, responsável tanto pela importação quanto pela exportação da cena criada. Para ser possível exportar, primeiro é preciso chamar pela função de `ordenarCena`, para então percorrer ela e, conforme os nomes das peças, criar objetos correspondentes em JSON, para então no final exportar o objeto cena. Já na importação, o objeto de cena é percorrido e, conforme o nome na chave do objeto, é criada uma peça correspondente, para então colocar os valores passados pelo usuário. Para a criação desses objetos foi utilizada a biblioteca SimpleJSON.

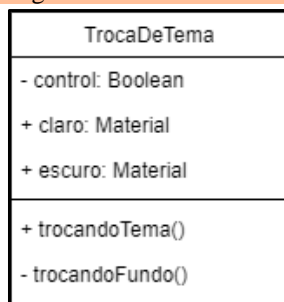
Figura 4 - Diagrama de classes Arquivo



Fonte: elaborado pela autora.

A Figura 5 representa a classe TrocaDeTema, responsável por mudar a cor do fundo da tela na parte das abas. Por ser uma classe bem simples, apenas apresenta as variáveis que contém as cores de fundo e a função para realizar tal tarefa. Como será explicado mais à frente, inicialmente imaginava-se trocar a cor das letras também; contudo, isso não foi possível.

Figura 5 - Diagrama de classes da Troca de Tema



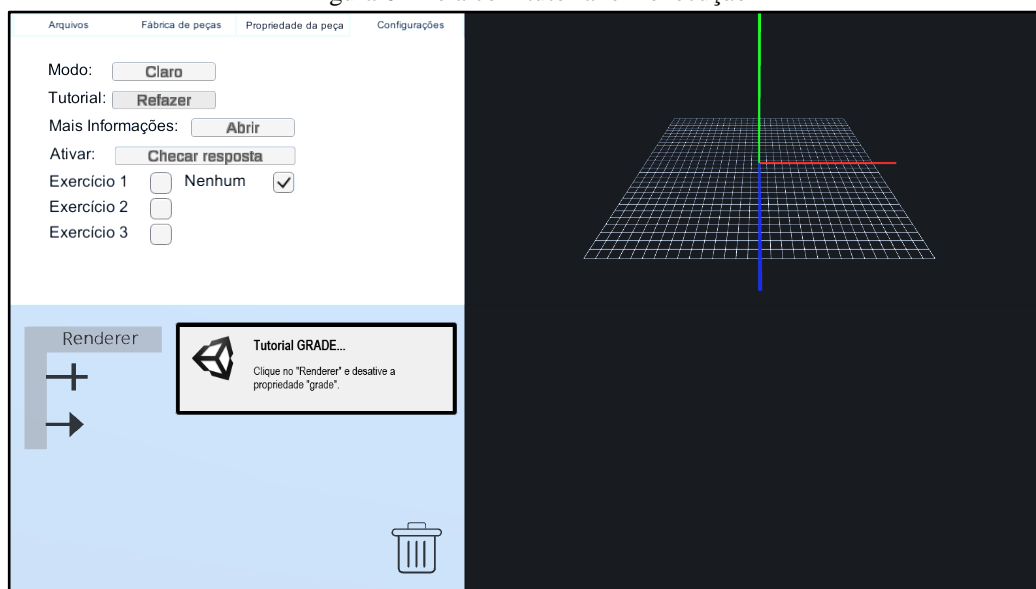
Fonte: elaborado pela autora.

3.2 IMPLEMENTAÇÃO

Para o desenvolvimento deste trabalho, foi preciso primeiramente converter a última versão do VisEdu (Buttenberg, 2020) para uma versão mais recente, sendo a 2022.1.13f da Unity escolhida para esse propósito. Em conjunto, foi utilizada a IDE Visual Studio 2019 na versão 16.11.34 para implementar os códigos em C#. Durante o desenvolvimento, foram usadas as bibliotecas SimpleJSON, usada na funcionalidade de importar/exportar cena, e WebGL Copy And Paste, para permitir que o campo de texto aceitasse ações de copiar e colar. Para auxiliar na implementação, foram usados diagramas, desenvolvidos no software Draw.io na versão 1.0. Na etapa da construção do novo tutorial, que será abordada mais à frente, o Paint 3D (disponibilizado pelo próprio sistema operacional Windows) foi usado para criar telas (APÊNDICE B), a partir de imagens já existentes dentro do projeto, para auxiliar o usuário no entendimento do uso da ferramenta. Todos os softwares utilizados eram de acesso gratuito.

Além das imagens, a criação do novo tutorial foi baseada no artigo “How to design a perfect game tutorial?” (Cieślak, 2021). Nele, Cieślak (2021) aborda oito dicas para serem seguidas na hora de criar um tutorial para jogos, tais como “não fazer um tutorial com menos de cinco etapas nem com mais de nove etapas” e “não se alongar nos textos”. A partir disso, o tutorial do GRADE ficou com nove telas: oito com instruções para seguir e uma parabenizando o usuário. O tutorial pode ser acionado assim que o programa for aberto. As telas ficam visíveis até o momento em que a instrução solicitada é realizada, indo para próxima até acabar e mostrar a mensagem de parabenização. Caso for desejado, o tutorial pode ser refeito. Basta acessar a aba Configurações e clicar no botão Refazer (Figura 6).

Figura 6 - Tela com tutorial em execução



Fonte: elaborado pela autora.

Inicialmente, se pretendia apenas aprimorar o código de tutorial que já existia no projeto até então, não sendo necessário construir outros diagramas. Contudo, complicações na hora da implementação inviabilizaram tal possibilidade e foi decidido que seria melhor criar um código novo do zero (Quadro 6). Com isso, o projeto parou de apresentar problemas e ficou com um tutorial em funcionamento e mais detalhado.

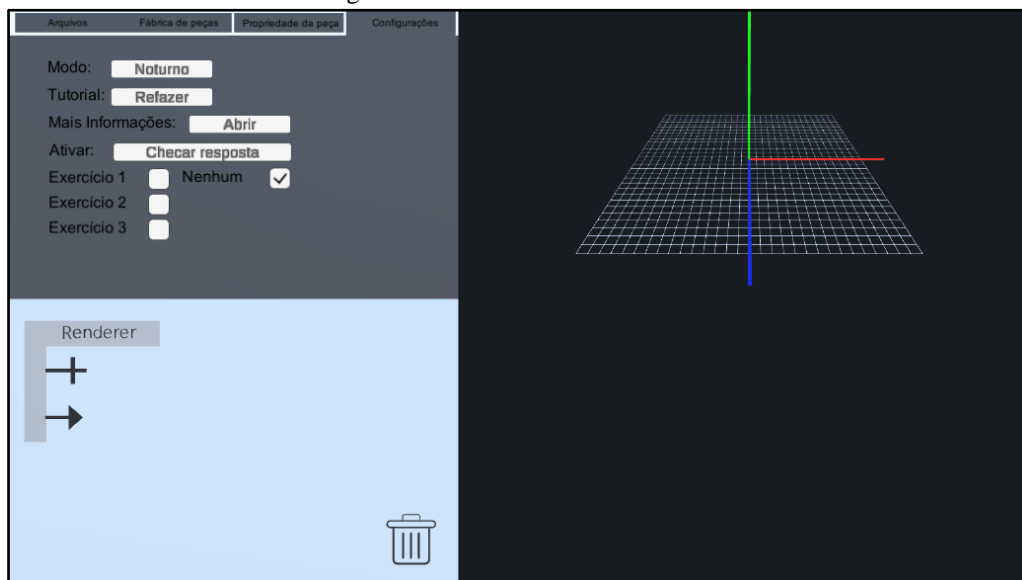
Quadro 6 - Função principal da classe TutorialNovo

```
public void abrirTutorial()
{
    switch (passo)
    {
        case 0 when render.activeSelf && !grade.isOn:
        case 1 when colisao.encaixada && colisao.peca == "Camera":
        case 2 when colisao.encaixada && colisao.peca == "Objeto":
        case 3 when colisao.encaixada && colisao.peca == "Iluminacao":
        case 4 when colisao.encaixada && colisao.peca == "Cubo":
        case 5 when colisao.encaixada && colisao.peca == "Escala":
        case 6 when escala.activeSelf && escalarTexto.text == "3":
        case 7 when Global.listaEncaixes.Count == 0: tutorialManager(passosTutorial);break;
        case 8:
            StartCoroutine(apagarTela(passosTutorial[passo]));
            StartCoroutine(apagarTela(painelTutorial));
            passo = 0; grade.isOn = true; break;
    }
}
```

Fonte: elaborado pela autora.

Outra funcionalidade apontada na lista de requisitos que passou por problemas foi a troca de tema. A ideia inicial era fazer com que tanto o fundo da tela quanto as letras mudassem de cor. Infelizmente, com as novas atualizações da Unity na parte de componentes e devido à forma como o projeto já havia sido programado por Buttenberg (2020), não foi possível mudar a cor das letras como se imaginava sem que prejudicasse outros componentes de texto do projeto. Sendo assim, a funcionalidade troca apenas a cor de fundo, mas ainda é possível ler o texto mesmo ele estando em preto (Figura 7).

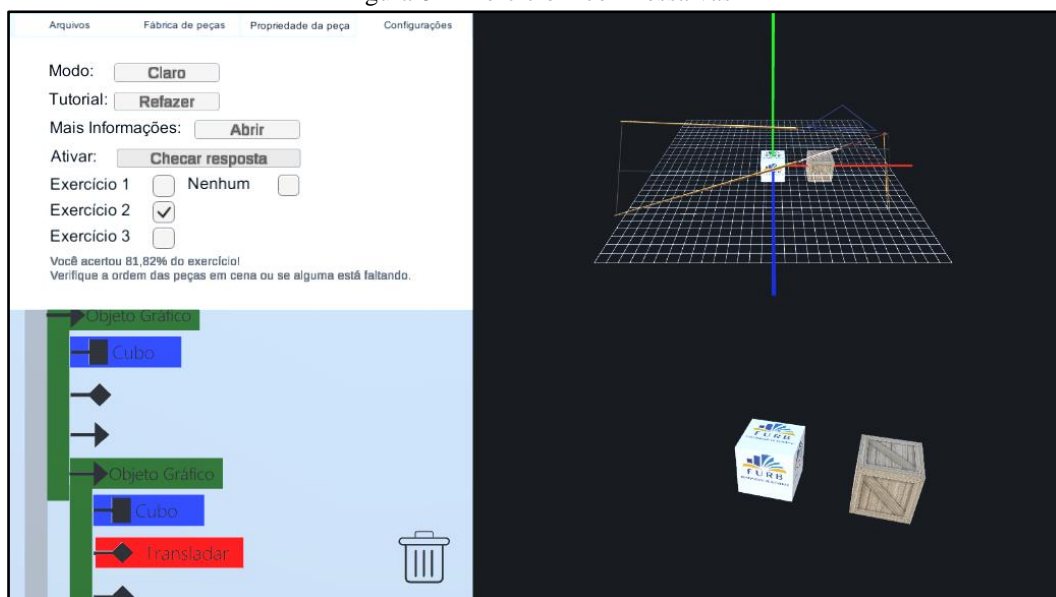
Figura 7 - GRADE no modo noturno



Fonte: elaborado pela autora.

A funcionalidade de exercícios foi desenvolvida conforme esperado. A ideia inicial (que se manteve) era o usuário ter como escolher entre três exercícios, para então ler o enunciado e fazer a atividade. Quando acabasse, deveria clicar no botão **Checar resposta** para ter um retorno de quantos por cento acertou. Caso a resposta apresentasse equívocos, seria devolvida uma mensagem avisando onde o usuário deveria rever o que foi feito para acertar. Para criar esses três exercícios, foi utilizado como base uma lista do professor da disciplina de Computação Gráfica (ANEXO A). Além disso, para contribuir com o aprendizado em aula, foram disponibilizados os gabaritos em formato JSON (APÊNDICE C), tanto para correção por parte do professor quanto para revisão do exercício por parte dos alunos (o professor repassará o gabarito quando achar necessário), podendo importar a cena (essa funcionalidade será abordada mais à frente) e checar o que está diferente da resposta dada. Na Figura 8, tem-se um exemplo de exercício com erros na execução. Nesse caso, o usuário esqueceu de colocar a peça **Escalar** no objeto pai, então a mensagem informa para que o usuário “Verifique a ordem das peças em cena ou se alguma está faltando”.

Figura 8 - Exercício 2 com ressalvas



Fonte: elaborado pela autora.

Em relação a finalizar o que faltou na última versão do projeto (Buttenberg, 2020), a propriedade de `LookAt` da câmera foi feita, além das propriedades `Near` e `Far`. Contudo, o `Near` e `Far` dão retorno visual apenas na tela do Visualizador, o `LookAt` dá retorno apenas na tela de Ambiente Gráfico, e as formas Polígono e Spline não foram construídas, em razão do fato de que houve problemas para entender o código desenvolvido por Buttenberg (2020) e foi preciso priorizar outras funcionalidades dentro do projeto, que não estavam previstas no projeto inicialmente.

A primeira funcionalidade não prevista foi a de importar e exportar a cena no formato JSON, permitindo a persistência de dados, para então o aluno poder continuar com a cena em outro momento. Para construir o arquivo (Quadro 7), foi preciso ordenar a lista dos objetos em cena pela ordem em que apareciam em tela para então começar a escrever o JSON. Para isso, foi utilizada a biblioteca `SimpleJSON`, que permite criar um `JSON Object` para então adicionar `JSON Arrays` conforme for necessário, assim como outros `JSON Objects`. Nele, cada nome de peça era a chave de uma lista de propriedades visíveis no painel de propriedades da ferramenta, além de informações relevantes para encaixar a peça em seu devido lugar posteriormente na importação da cena em JSON.

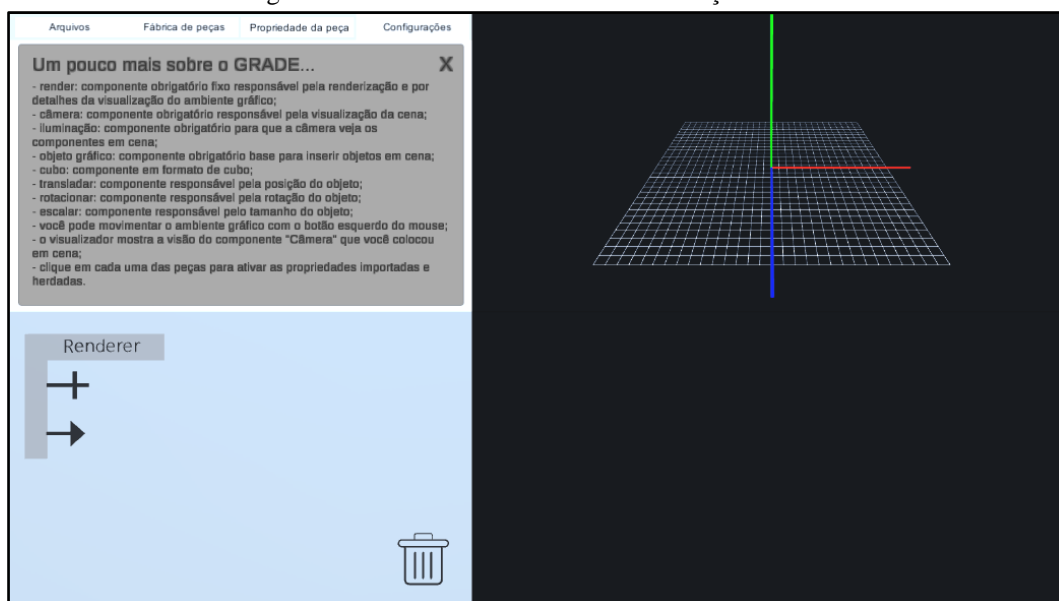
Na importação, o arquivo JSON é percorrido e cada peça é criada com as propriedades informadas e com seus respectivos filhos. Contudo, devido ao fato de que algumas propriedades não são retornadas no Visualizador e no Ambiente Gráfico de imediato, o usuário deve clicar em cada peça para “ativar” as propriedades e fazer com que elas sejam visíveis nas janelas citadas anteriormente. Isso ocorre em razão do fato de, na versão de Buttenberg (2020), não existir a função de importar e de herdar propriedades. Então a única forma que o usuário poderia mudar uma propriedade seria com o painel dela em aberto, sendo necessário clicar nas peças para isso. Para evitar maiores complicações mexendo nessa parte mais complexa do código, optou-se apenas por informar ao usuário sobre a necessidade de clicar nas peças quando importadas na aba Configurações em Mais Informações. Ao clicar no botão Abrir, aparecerá um painel com informações adicionais para ajudar o usuário (Figura 9).

Quadro 7 - Exemplo de cena exportada em JSON

```
{
  "CameraP": {
    "nome": "Câmera",
    "posicao": ["100", "300", "300"],
    "lookAt": ["0", "0", "0"],
    "fov": "45",
    "near": "100",
    "far": "600",
    "posPeca": [696.351135253906, 624.93212890625, -870.424987792969]
  },
  "ObjetoGraficoP": {
    "nome": "ObjetoGraficoP",
    "ativo": true,
    "children": [
      {
        "Cubo": {
          "nome": "Cubo",
          "tamanho": ["2", "2", "2"],
          "posicao": ["0", "0", "0"],
          "cor": "RGBA(1.000, 0.000, 0.000, 1.000)",
          "textura": "FURB",
          "ativo": true,
          "posPeca": [697.420593261719, 618.979125976563, -870.424987792969]
        }
      },
      {
        "Iluminacao": {
          "nome": "Iluminacao",
          "tipoLuz": "Ambiente",
          "posicao": ["100", "300", "0"],
          "cor": "RGBA(1.000, 1.000, 1.000, 1.000)",
          "ativo": true,
          "posPeca": [698.855163574219, 613.408264160156, -870.403076171875]
        }
      }
    ],
    "posPeca": [697.301147460938, 621.630798339844, -870.403076171875]
  }
}
```

Fonte: elaborado pela autora.

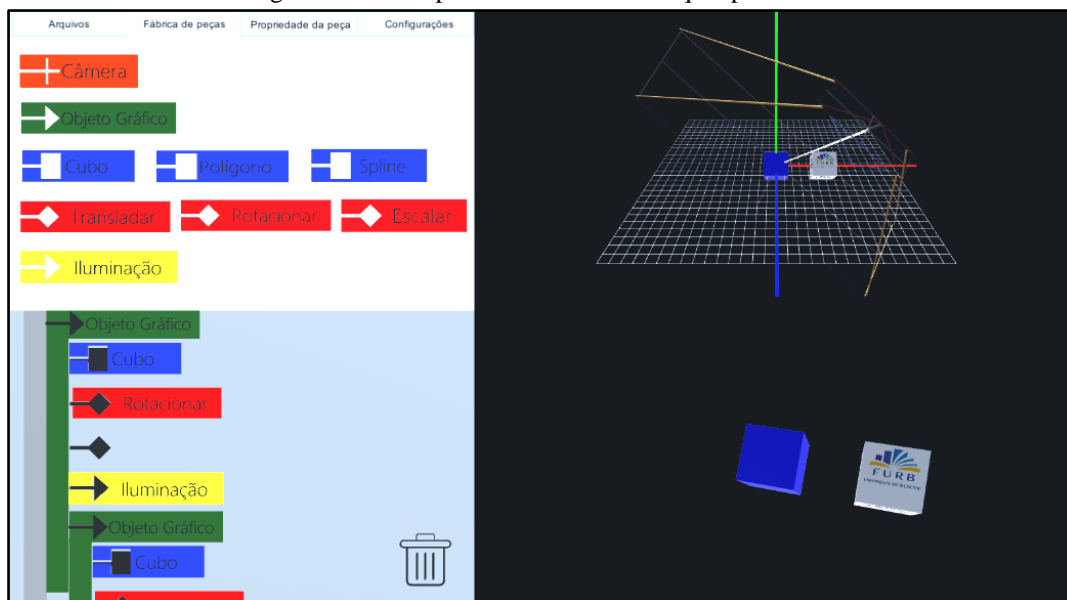
Figura 9 - GRADE com o menu de informações aberto



Fonte: elaborado pela autora.

Outra funcionalidade que não estava prevista, porém implícita nos requisitos funcionais (Quadro 4), foi a hierarquia de objeto pai e filho. Ela é responsável por permitir que qualquer propriedade do objeto pai seja herdada pelo filho (Figura 10). Entretanto, ela não coube no tutorial que aparece no início da aplicação, visto que o limite de telas já tinha sido atingido e, caso fosse adicionada, faria o tutorial ficar muito longo, podendo fazer com que o usuário perdesse o interesse, prejudicando no aprendizado do aluno. Sendo assim, um tutorial extra foi construído para explicar tal funcionalidade de forma completa, podendo ser acessada pelo caminho Configurações > Tutorial Pai-Filho > Abrir.

Figura 10 - Exemplo de cena com hierarquia pai e filho



Fonte: elaborado pela autora.

4 RESULTADOS

Nesta seção serão apresentados os testes de funcionalidade, testes com os alunos de Computação Gráfica e comparação com os trabalhos correlatos.

4.1 TESTES DE FUNCIONALIDADE

Para testar as funcionalidades da aplicação, foi feito o *build* da aplicação para WebGL. Após concluído o *build*, que foi mandado para seu respectivo repositório no GitHub, o qual foi feito um link para o GitHub Pages, para então

utilizar a aplicação construída. Assim que o GitHub terminou de verificar o *commit*, foi possível usufruir a ferramenta. Ao clicar no link, a aplicação carregou e abriu conforme esperado, mostrando o menu de pular ou não o tutorial. Ao clicar em *Sim* para realizar o tutorial, ele segue conforme esperado: mostrando as telas na ordem correta e esperando pela ação do usuário que corresponde ao que foi solicitado, até encerrar e parabenizar o usuário por ter completado o tutorial. Quando a opção *Pular tutorial* é escolhida, o menu fecha conforme esperado.

Ao navegar pelas abas *Arquivo*, *Fábrica de peças*, *Propriedade da peça* e *Configurações*, elas mostram seus respectivos conteúdos com sucesso. Na aba de *Arquivo*, a cena montada é exportada em formato de JSON assim que o botão *Exportar* é acionado, conforme foi programado. Ao importar uma cena, contudo, é preciso que a aplicação seja reiniciada para não gerar complicações, o que é informado ao usuário conforme esperado. Após isso, ao colar o conteúdo JSON na caixa de texto e clicar em *Importar*, a cena é carregada com sucesso, sendo apenas preciso clicar nas peças (em especial, as peças de ações) para garantir que todas as propriedades tenham retorno visual.

Ao clicar na aba da *Fábrica de peças*, as peças são apresentadas e podem ser arrastadas até a janela de *Renderer* para montar a cena, conforme esperado. Contudo, as peças *Spline* e *Polígono* continuam apenas como elementos visuais da *Fábrica de Peças*, não podendo ser adicionadas no *Renderer*. Após posicionar as peças nos respectivos *slots*, ainda se pode clicar nelas, o que abrirá a aba de propriedades daquela peça, onde é possível editar os valores das propriedades ali presentes. Ainda é permitido que as peças sejam arrastadas até a lixeira, para então serem removidas da cena. Todas as ações da janela de *Renderer* funcionaram com sucesso.

Ao selecionar a aba de *Configurações*, algumas funcionalidades extras são apresentadas. A primeira delas é a troca de tema, a qual permite trocar a cor do fundo das abas entre branco e cinza-escuro, conforme esperado. Nesta aba, também é possível refazer o tutorial que é disponibilizado no início da aplicação, o qual funciona da mesma maneira de quando é ativado logo que o usuário entra na aplicação. O painel de *Mais Informações* presente na aba *Mostra Informações* e dicas da aplicação com sucesso. Por fim, existe a funcionalidade de *exercícios*. Nela, um dos exercícios deve ter seu *checkbox* selecionado e mostrar seu respectivo enunciado, conforme ocorrido durante os testes. Após construir a cena solicitada, deve-se clicar no botão *Checar resposta*, que dará um retorno frente aos acertos e possíveis correções da atividade. Os três exercícios foram testados e todos se comportaram da maneira esperada: mostrando 100% quando tudo está correto e mostrando uma variação de porcentagem acompanhada de uma dica sobre onde o erro está.

Ao clicar nas janelas *Ambiente Gráfico* e *Visualizador* mostraram a cena em tempo real. A primeira permitia zoom e navegação dentro dela, podendo posicionar a grade e os eixos conforme desejado. A segunda respeitava as propriedades da *Câmera*, sendo então necessário mexer nessa peça para mudar a forma como era a visão dessa tela.

4.2 TESTES COM OS ALUNOS DE COMPUTAÇÃO GRÁFICA


Para verificar se a aplicação desenvolvida alcançaria a satisfação dos usuários, foi desenvolvido um questionário no *Google Forms* para os alunos da disciplina de Computação Gráfica do primeiro semestre de 2024, com um passo a passo de uso da aplicação acompanhado de perguntas sobre o entendimento do usuário e das funcionalidades do GRADE. O formulário foi aplicado pelo professor da disciplina durante o período de aulas valendo pontos extras.

Primeiramente, o aluno é contextualizado sobre a aplicação ser um projeto de TCC e do que ela se trata. Ao continuar, o usuário deve inserir seu e-mail institucional (para receber pontos extras), classificar seu conhecimento tanto em Geometria quanto em CG de uma escala de um a cinco, sendo que um representa Ruim e cinco representa Excelente, e responder se já teve algum contato com aplicações com visualização de objetos em 3D, tais como Unity. Nesta primeira etapa, dentre as 43 respostas dadas, mais da metade dos alunos observados definiu seus conhecimentos, tanto em Geometria quanto em CG, como três, tendo o valor quatro como segunda resposta mais selecionada em ambas as perguntas. Já na questão de aplicações 3D, 58,1% dos alunos disseram já ter usado aplicações desse tipo.

Na segunda etapa de respostas, os alunos testaram a aplicação e responderam as perguntas apresentadas na ordem que foram formuladas. Após abrir o link para o projeto, o aluno deveria realizar o tutorial e concluí-lo. Todos os alunos conseguiram chegar ao final do tutorial. Após isso, foram solicitados a clicar na aba *Configurações* e mudar o tema da aplicação. Todos os alunos conseguiram realizar a tarefa. Nesta mesma aba, ainda tiveram que abrir o painel de *Mais Informações* e ler o seu conteúdo, para então responder se entenderam tudo, parcialmente ou nada do que estava escrito. No qual 90,7% dos alunos disseram ter entendido tudo o que estava escrito no painel, e o restante afirmou ter entendido parcialmente o texto, ficando com dúvidas em alguns tópicos.

Em seguida, tiveram que realizar os exercícios cadastrados na aplicação, seguindo sua respectiva ordem (exercício 1, exercício 2 e exercício 3). No primeiro exercício apresentado, dez alunos ficaram abaixo dos 75% de acerto e, desses dez, cinco estavam abaixo de 60% de acerto, tendo como menor porcentagem de acerto 37,5%. A maior porcentagem de acerto dentre os alunos foi 96%. Tanto a maior nota quanto a menor foram alcançadas apenas por uma ocorrência cada. No segundo exercício apresentado, nove alunos ficaram abaixo dos 75% de acerto, tendo apenas um aluno abaixo dos 60%, com 16,67% de acerto (a menor pontuação do exercício). Neste exercício, cinco alunos atingiram

a pontuação máxima. No último exercício, não houve alunos com menos de 60% de acerto. Contudo, sete alunos ainda tiveram pontuação menor que 75%, tendo como menor nota 60%, alcançada por quatro alunos. Já no quesito nota máxima, oito alunos conseguiram realizar com sucesso o exercício.

Para encerrar essa etapa,  apresentada a funcionalidade de exportar e de importar uma cena em formato JSON. Na questão de exportar, os alunos tiveram que exportar a cena do último exercício desenvolvido. Após exportar, tiveram que copiar o conteúdo JSON da caixa de texto e guardar para usar na próxima atividade. Foi requisitado que colassem o conteúdo em um formatador de arquivos JSON, para poderem visualizar melhor os valores armazenados, para então responderem se o arquivo veio com as informações corretas. 86% dos alunos afirmaram que o arquivo JSON estava com os valores corretos, 11,6% disseram que o arquivo JSON estava com alguma informação errada e 2,3% afirmaram que não conseguiram exportar a cena. Na questão de importar, 73,8% disseram ter conseguido com sucesso, 23,8% importaram, mas alguns valores não foram importados, e 2,4% não conseguiram importar. Infelizmente, essa questão acabou não sendo colocada como obrigatória, o que acarretou numa quantidade diferente de alunos (42 de 43 responderam).

Por fim, na última etapa, os alunos tiveram que avaliar se a **ferramenta** ajuda no conhecimento de Geometria e CG após o uso da aplicação, fazendo uso da mesma escala apresentada anteriormente. Em ambas, boa parte dos alunos ficou dividida entre as respostas quatro e cinco, e alguns poucos responderam três e dois. Por fim, as últimas três perguntas eram optativas e referentes a pontos positivos e negativos da aplicação e sugestões para a aplicação. Em relação a quantidade de respostas, foram respondidas por, respectivamente, 34, 32 e 16 alunos. Visto que os alunos divagaram demais sobre a aplicação e fizeram uso de gírias, a maior parte das respostas foi reformulada para melhorar a leitura e interpretação do texto; contudo, mantendo a essência das respostas. Em relação aos pontos positivos (Quadro 8), os alunos usaram termos muito semelhantes para descrever a aplicação, além de fazer mais de um apontamento ao longo do texto, fazendo com que muitas das descrições dadas se repetissem.

Quadro 8 - Pontos positivos da aplicação apontados pelos alunos de CG

Palavras-chave/termos usados para descrever a aplicação	Quantidade de ocorrências
Fácil/simples/prática de usar	10
Facilitou no entendimento/aprendizado dos assuntos abordados em aula	10
Intuitiva/fácil de entender	9
Visualização 3d/ apoio visual ajuda a entender/visualização da cena	8
Scrath/sketch/drag-drop das peças/interação com a cena	6
Educativa/didática	4
Exercícios bons para fixação de conteúdo/exercícios interessantes e correspondem ao assunto de cg	2
Web	2
Tutorial ajuda muito	2
Divertido	2
Rápida	2
Interessante	2
Funcional	1
Leve	1
Legal	1
Permite autoavaliação e revisão de conteúdo	1

Fonte: **questionário** elaborado pela autora.

Em relação aos pontos negativos (Quadro 9), notou-se que alguns se repetiram muito e que outros não eram *bugs*, como foram apontados por alguns alunos, e sim um comportamento esperado pela aplicação que, se fosse alterado, poderia prejudicar o aprendizado e experiência dos alunos (exemplo: “tutorial não permite pular etapas”). Outros pontos foram apontados como negativos (como não poder mexer na angulação dos objetos), mesmo existindo dentro da aplicação, o que mostra certa falta de atenção dos alunos ao explorar ela. Outros pontos negativos acabaram aparecendo pelo fato de, inicialmente, ter sido disponibilizado o link errado para a aplicação (exemplo: “problemas na acentuação de palavras”). Ainda é visível que alguns alunos utilizaram o espaço de pontos negativos para dar sugestões, tais como utilizar o localStorage para armazenar informações de sessão do usuário. E, alguns outros, não foram específicos o suficiente sobre seus apontamentos, dando frases confusas como resposta (exemplo: “visualização não renderiza”).

Quadro 9 - Pontos negativos da aplicação apontados pelos alunos de CG

Palavras-chave/termos usados para descrever a aplicação	Quantidade de ocorrências
Texto fica invisível no campo de texto das propriedades após o uso da tecla <i>backspace</i>	19
Visual pouco elaborado/baixa resolução/problemas na responsividade	4
Problemas na acentuação de palavras	3

Bug na porcentagem de acerto do exercício quando clica mais de uma vez no botão	3
Bugs	3
Quando há muitos objetos em cena, demora para fazer a rolagem de tela	2
Aplicação um pouco travada	2
Visualização dos ângulos dos objetos está presa	2
Funcionamento da câmera é estranho/complicado de entender	2
Textos pequenos	1
Poderia utilizar o localStorage para armazenar informações de sessão do usuário.	1
Não é mostrado ao usuário qual o intervalo de números que podem ser inseridos nas propriedades	1
Deveria ter uma explicação mais detalhada sobre as outras funcionalidades da aba Configurações	1
Modo escuro restrito a apenas uma das quatro janelas	1
Na hierarquia pai-filho, quando o filho já herdou algo do pai e é clicado no filho para editar, ele perde na visualização o que herdou do pai e é preciso clicar novamente no pai para a propriedade reaparecer na visualização	1
Tutorial não permite pular etapas	1
Métricas de avaliação dos exercícios são confusas	1
O check sai do checkbox do exercício selecionado quando o botão de checar é clicado	1
Os enunciados dos exercícios podem ser confusos para alguns, deveria ter dicas de como resolver	1
Visualização não renderiza	1
Confusa	1
Talvez mais exercícios em sala de aula	1
Importação não aplica propriedades corretamente, mesmo estando certo no JSON	1
Bug na exclusão de objetos	1
Tela de visualização dos objetos é pequena	1
A aplicação não permite o acesso ao gabarito	1

Fonte: questionário elaborado pela autora.

Já em relação às sugestões (Quadro 10), houve algumas repetições, mas não de forma tão evidente quanto nos quadros anteriores. Esse espaço também foi utilizado pelos alunos para parabenizar pelo trabalho desenvolvido. É evidente também que, alguns dos alunos que usaram o espaço dos pontos negativos para sugestões, não as repetiram nesta pergunta (exemplo: “poderia utilizar o localStorage para armazenar informações de sessão do usuário” e “Os enunciados dos exercícios podem ser confusos para alguns, deveria ter dicas de como resolver”).

Quadro 10- Sugestões para a aplicação apontadas pelos alunos de CG

Termos e elogios usados/dados para descrever a aplicação	Quantidade de ocorrências
Ficará ótimo se arrumar bugs anteriormente apontados	4
Melhorar modo noturno	3
Parabéns pelo trabalho!	2
Corrigir bug dos inputs das propriedades	2
Fazer uso de setas no tutorial para indicar onde encaixar as peças, onde pegá-las, fazendo com que fique mais fácil e interativo	2
Remover Spline e Polígono da Fábrica de Peças, visto que não funcionam	1
Permitir acesso ao gabarito dentro da aplicação	1
Poder mover objetos utilizando o mouse como na Unity	1
Arrumar bugs	1
Aumentar tela de visualização	1
Arrumar acentuação	1

Fonte: questionário elaborado pela autora.

4.3 COMPARAÇÃO COM OS CORRELATOS

Retomando sobre os correlatos citados anteriormente, todos os presentes são aplicações educacionais, visando contribuir com o aprendizado em alguma área do conhecimento. O primeiro, chamado Scratchim, busca ensinar o pensamento computacional através da programação por blocos desplugada (não faz uso do digital, são peças físicas). O segundo, nomeado GeNiAl, traz trilhas de conhecimento sobre química através de brincadeiras, como jogo da memória. O terceiro, intitulado OrbitAndo, é uma plataforma com três jogos sobre astronomia, indo de questionários e indo até o uso de realidade aumentada. Veja no Quadro 11 a comparação entre estes trabalhos e a aplicação desenvolvida, GRADE.

Quadro 11 - Comparação entre os correlatos e aplicação desenvolvida

Trabalhos Observados Características	Scratchim (Rodrigues; Gomes; Carneiro, 2022)	GeNiAI (Barros; Sousa; Viana, 2022)	OrbitAndo (Siedler et al., 2022)	GRADE
Existe interação por meio de peças	X			X
É um software educacional		X	X	X
Apresenta exercícios para validação	X	X	X	X
Apresenta tutorial explicando o seu uso	X			X
Apresenta conteúdos teóricos	X		X	X
Possui acesso off-line	X		X	Apenas após carregamento completo da página
Foi desenvolvido em Unity			Dois dos três jogos	X
Disponibilidade	Físico	Web	Multiplataforma	Web

Fonte: elaborado pela autora.

Frente às características analisadas no quadro anterior, é evidente que a aplicação desenvolvida buscou trazer o melhor de cada um deles, mesmo que ainda não seja completamente de uso off-line. A aplicação faz uso da programação por blocos, preenchendo a primeira característica. É um software educacional, visto que tem como público-alvo os alunos da disciplina de CG. Seguindo com a lista, apresenta três exercícios para validação, acompanhados da porcentagem de acerto. Apresenta um tutorial que ensina o básico para usufruir da ferramenta, podendo retomá-lo a qualquer momento. Também apresenta conteúdos teóricos no painel de Mais Informações. Além disso, foi desenvolvida em Unity para plataforma Web, o que facilita muito o acesso, pois não depende do sistema operacional do aparelho em que está sendo acessado.

5 CONCLUSÕES

Através dos resultados obtidos na avaliação aplicada aos alunos de CG, nota-se que o uso da ferramenta por parte dos alunos foi satisfatório. Grande parte do deles elogiou a ferramenta, dizendo ser intuitiva e até divertida, o que contribui muito no aprendizado. Muitos alunos também afirmaram que a visualização 3D de objetos durante a montagem da cena ajudou muito a entender melhor o conteúdo da disciplina de CG. Também foi possível observar a evolução dos alunos ao longo dos exercícios, visto que, no primeiro, havia alguns alunos com pontuações baixas e, no último, não havia alunos com acertos menores do que 60% (média da FURB). Sendo assim, verifica-se que a aplicação alcançou tanto o objetivo principal quanto os objetivos específicos.

Em relação ao uso de Unity com Visual Studio como principais ferramentas para desenvolver o projeto, estas cumpriram perfeitamente bem sua função, facilitando a implementação com sua interface de elementos arrastáveis em conjunto com a IDE e suas dicas e notificações sobre erros no código. As bibliotecas usadas (SimpleJSON e WebGL Copy And Paste) tiveram papel importante no desenvolvimento da funcionalidade de importar e exportar cena. O Paint 3D também se mostrou muito eficaz na produção das telas do tutorial, visto que era possível apagar o texto da etapa e escrever outro no lugar, mantendo então uma padronização nas imagens. O Draw.io se mostrou uma ferramenta de diagramação muito prática, trazendo modelos prontos, o que ajudou a reduzir o tempo gasto na montagem de diagramas.

Infelizmente, o projeto acabou tendo algumas limitações em razão de ter sido construído a partir do código da versão anterior (Buttenberg, 2020) e pelo tempo limitado para sua produção. Alguns requisitos também não foram feitos completamente. As peças Spline e Polígono não foram construídas e as propriedades da Câmera ainda são insuficientes, visto que não dão retorno visual em ambas as telas (Ambiente Gráfico e Visualizador). Frente ao que não foi desenvolvido e às respostas dadas pelos alunos, como possíveis extensões para o projeto, tem-se:

- construir as peças Spline e Polígono;
- fazer com que as propriedades LookAt, Near e Far da Câmera funcionem de forma satisfatória;
- corrigir validação dos campos de propriedades para que o texto não perca a visibilidade;
- corrigir cálculo da porcentagem de acertos dos exercícios;
- arrumar as propriedades das peças, para que não seja necessário clicar nelas quando forem importadas nem herdadas;
- fazer um esquema de temporização ao longo do exercício e, quando o aluno estiver levando muito tempo para fazer, disponibilizar acesso a dicas de como resolver;
- trazer o modo noturno para a tela de Render;
- trazer setas indicando o caminho durante o tutorial.

REFERÊNCIAS

- ARAÚJO, Luciana P. de. **Adubogl**: Aplicação Didática usando a Biblioteca Open GL. 2012. 76f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2012. Disponível em: https://bu.furb.br/docs/MO/2012/350348_1_1.pdf. Acesso em: 04 maio 2024.
- AZEVEDO, Eduardo; CONCI, Aura; VASCONCELOS, Cristina. **Computação Gráfica: Teoria e Prática: Geração de Imagens**. 1. ed. rev. Rio de Janeiro: Alta Books, 2022.
- BARROS, Gabriel C.; SOUSA, Janyeid K. C.; VIANA, Davi. Jornada Química GeNiAl: um jogo sério para o ensino da tabela periódica e seus elementos. *In: CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO*, 11., 2022, Manaus. **Anais [...]**. Manaus: Publication chair, 2022. p. 1-12. Disponível em: <https://sol.sbc.org.br/index.php/sbie/article/view/22432/22256>. Acesso em: 27 nov. 2023.
- BUTTENBERG, Peterson B. **VisEdu-CG 5.0: Visualizador de material educacional**. 2020. 19 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2020. Disponível em: https://www.furb.br/dsc/arquivos/tccs/monografias/2020_1_peterson-boni-buttenberg_monografia.pdf. Acesso em: 27 nov. 2023.
- CIEŚLAK, K. How to design a perfect game tutorial?. **Try_evidence**, 19 maio 2021. Disponível em: <https://tryevidence.com/blog/how-to-design-a-perfect-game-tutorial/>. Acesso em: 19 jun. 2024.
- FASSARELLA, Lucio S.; ROCHA, Rosângelo J. da. Geogebra 3D: Relato de uma experiência na superação de dificuldades de aprendizagem em geometria espacial. **Kiri-kerê**, São Mateus, v. 3, n. 5, p. 261-275, nov. 2018. Disponível em: <https://periodicos.ufes.br/kirikere/article/view/20347/14547>. Acesso em: 28 nov. 2023.
- KOEHLER, William F. **VisEdu-CG 4.0: Visualizador de Material Educacional**. 2015. 90 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2015. Disponível em: https://www.furb.br/dsc/arquivos/tccs/monografias/2015_1_william-fernandes-koehler_monografia.pdf. Acesso em: 28 nov. 2023.
- MANSSOUR, Isabel H.; COHEN, Marcelo. Introdução à computação gráfica. **Revista de Informática Teórica e Aplicada**, Rio Grande do Sul, v. 13, n. 2, p. 1-25, 2006. Disponível em: <https://www.inf.pucrs.br/manssour/Publicacoes/TutorialSib2006.pdf>. Acesso em: 1 out. 2023.
- MONTIBELER, James P. **VisEdu-CG: Aplicação Didática para Visualizar Material Didático, Módulo de Computação Gráfica**. 2014. 106 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2014. Disponível em: https://www.furb.br/dsc/arquivos/tccs/monografias/2014_1_james-perkison-montibeler_monografia.pdf. Acesso em: 28 nov. 2023.
- NUNES, Samuel A. **VisEdu-CG 3.0: Aplicação Didática para Visualizar Material Didático: Módulo de Computação Gráfica**. 2014. 89 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2014. Disponível em: https://www.furb.br/dsc/arquivos/tccs/monografias/2014_1_samuel-anderson-nunes_monografia.pdf. Acesso em: 28 nov. 2023.
- REIS, Dalton S. **Entrevista sobre aulas de Computação Gráfica**. Entrevistador: Natália Sens Weise. Blumenau. 2018. Entrevista feita através de conversação – não publicada.
- RODRIGUES, Amanda K. M.; GOMES, Kamily C. O.; CARNEIRO, Murillo G. Scratchim: uma abordagem para o ensino do Pensamento Computacional para crianças de forma remota e desplugada. *In: CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO*, 11., 2022, Manaus. **Anais [...]**. Manaus: Publication chair, 2022. p. 1-12. Disponível em: <https://sol.sbc.org.br/index.php/sbie/article/view/22515/22339>. Acesso em: 27 nov. 2023.
- SCHRAMM, Elizandro J. **Adubogl ES 2.0: Aplicação Didática usando a Biblioteca OpenGL EE 2.0 no iOS**. 2012. 64f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2012. Disponível em: https://bu.furb.br/docs/MO/2012/350319_1_1.pdf. Acesso em: 04 maio 2024.
- SETTIMY, Thaís F. de O.; BAIRRAL, Marcelo A. Dificuldades envolvendo a visualização em geometria espacial. **VIDYA**, Santa Maria, v. 40, n. 1, p. 177-195, jan./jun. 2020. Disponível em: https://www.researchgate.net/publication/343556166_DIFICULDADES_ENVOLVENDO_A_VISUALIZACAO_EM_GEOMETRIA_ESPACIAL. Acesso em: 28 nov. 2023.

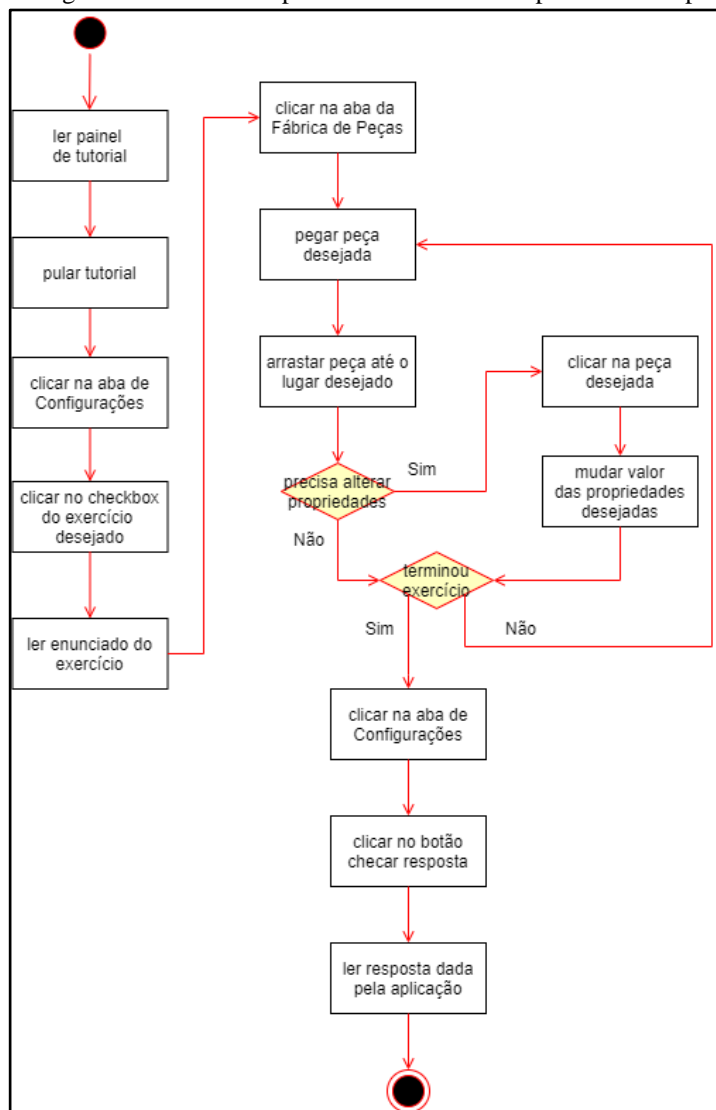
SIEDLER, Marcelo S. *et al.* OrbitAndo: uma plataforma para ensino de Astronomia de outro mundo. *In: CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO*, 11., 2022, Manaus. **Anais** [...]. Manaus: Publication chair, 2022. p. 1-11. Disponível em: <https://sol.sbc.org.br/index.php/sbie/article/view/22434/22258>. Acesso em: 27 nov. 2023.

SILVA, Romano J. M. da; RAPOSO, Alberto B.; GATTAS, Marcelo. **Grafo de Cena e Realidade Virtual**. Rio de Janeiro: PUC, 2004. Disponível em: https://web.tecgraf.puc-rio.br/~abraposo/INF1366/2007/02_GrafoDeCena_texto.pdf. Acesso em: 27 nov. 2023.

APÊNDICE A – DIAGRAMAS DE ESPECIFICAÇÃO

Neste apêndice são apresentados os diagramas de atividades das principais funções desenvolvidas, sendo elas: realização de exercícios, tutorial, exportar e importar a cena montada em conteúdo de formato JSON. A Figura 11 demonstra o diagrama das atividades que serão feitas pelo usuário após a aplicação abrir para conseguir realizar os exercícios.

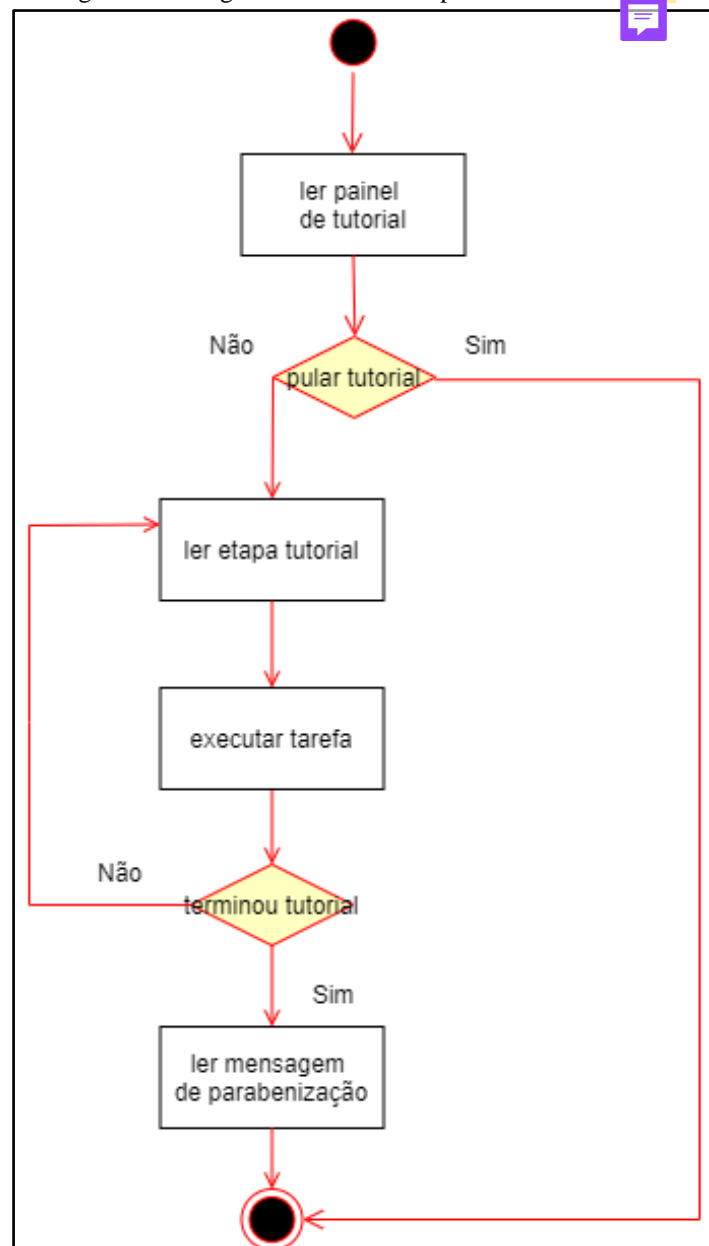
Figura 11 - Diagrama de atividades para realizar exercícios pré-definidos pela aplicação



Fonte: elaborado pela autora.

A Figura 12 representa o diagrama de atividades referente a realização do tutorial. O diagrama considera a primeira aparição do painel de tutorial, não o botão de Refazer presente na aba Configurações.

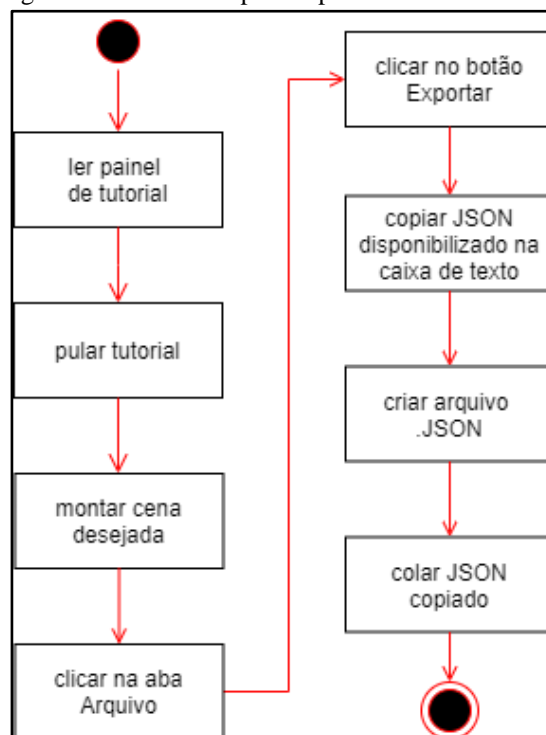
Figura 12 - Diagrama de atividades para realizar o tutorial



Fonte: elaborado pela autora.

A Figura 13 apresenta o diagrama de atividades da função de exportar a cena criada pelo usuário em formato JSON. O cenário montado considera que o usuário pule o tutorial e monte a cena e, assim que acabar, vá direto para a exportação para, então, copiar o conteúdo em formato JSON.

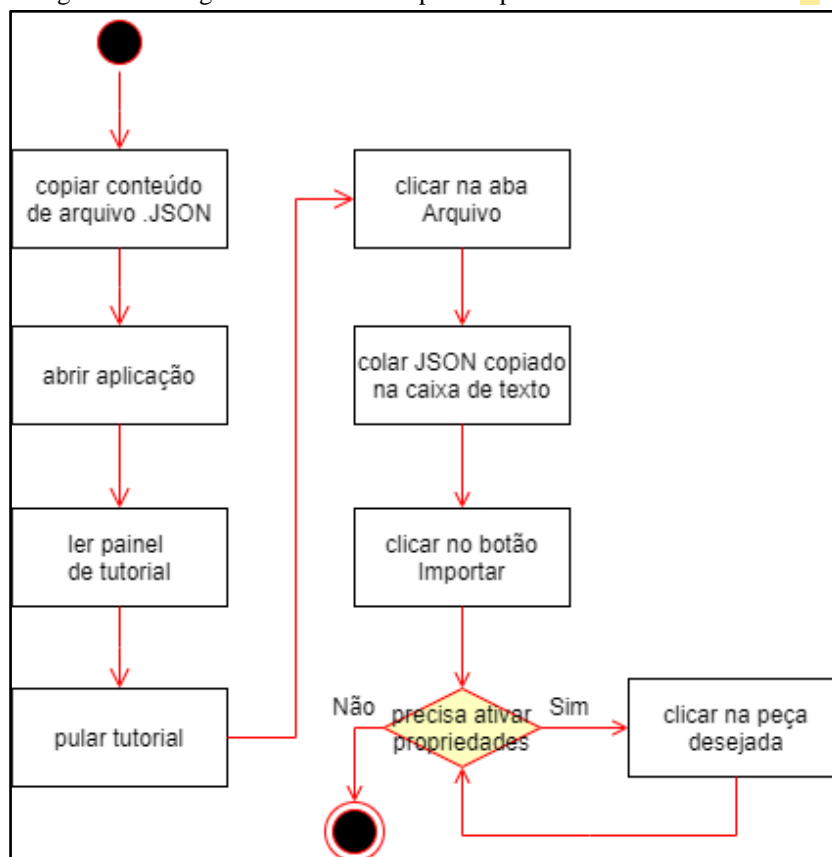
Figura 13 - Diagrama de atividades para exportar cena criada em formato JSON



Fonte: elaborado pela autora.

A Figura 14 mostra o diagrama de atividades da importação de uma cena em formato JSON, e considera que o usuário deva copiar o conteúdo e abrir a aplicação, para então pular o tutorial e importar a cena desejada.

Figura 14 - Diagrama de atividades para importar cena em formato JSON












Fonte: elaborado pela autora.

APÊNDICE B – TELAS DESENVOLVIDAS PARA O TUTORIAL











Neste apêndice são apresentadas as telas desenhadas no software Paint 3D para orientar o usuário durante o tutorial, onde cada janela foi desenhada para uma etapa. Foram feitas 14 telas de etapas que foram utilizadas no projeto, mais outras cinco, que foram descartadas durante a implementação do tutorial. A Figura 15 mostra as etapas do tutorial principal, que é apresentado ao usuário assim que ele abre a aplicação. Já a Figura 16 mostra tanto as etapas desenvolvidas para o tutorial extra, referente a funcionalidade de hierarquia pai-filho, quanto etapas que foram descartadas ao longo do desenvolvimento da aplicação. As etapas descartadas não apresentam o título em negrito.

Figura 15 - Etapas do tutorial principal

 Tutorial GRADE... Clique no "Renderer" e desative a propriedade "grade".	 Tutorial GRADE... Arraste a peça "Câmera" até o encaixe de cruz no "Renderer".
 Tutorial GRADE... Arraste a peça "Objeto Gráfico" até o encaixe de seta no "Renderer".	 Tutorial GRADE... Arraste a peça "Iluminação" até o encaixe de seta no "Objeto Gráfico".
 Tutorial GRADE... Arraste a peça "Cubo" até o encaixe de quadrado no "Objeto Gráfico".	 Tutorial GRADE... Arraste a peça "Escalar" até o encaixe de losango no "Objeto Gráfico".
 Tutorial GRADE... Clique no "Escalar" posicionado e mude o valor de "x" para 3.	 Tutorial GRADE... Agora, arraste a peça "Câmera", que está encaixada, até o ícone de "Lixeira", assim como a peça "Objeto Gráfico".
 Tutorial GRADE... Parabéns! Você concluiu o tutorial! Agora você já pode criar suas próprias cenas!	

Fonte: elaborado pela autora.

Figura 16 - Etapas do tutorial extra e etapas descartadas da aplicação

 <p>Tutorial GRADE...</p> <p>Arraste a peça "Objeto Gráfico" até o encaixe de seta no "Objeto Gráfico" em cena.</p>	 <p>Tutorial GRADE...</p> <p>Arraste a peça "Transladar" até o encaixe de losango no "Objeto Gráfico" filho e mude o valor de "x" para 2.</p>
 <p>Tutorial GRADE...</p> <p>Arraste a peça "Cubo" até o encaixe de quadrado no "Objeto Gráfico" filho.</p>	 <p>Tutorial GRADE...</p> <p>Arraste a peça "Cubo" até o encaixe de quadrado no "Objeto Gráfico" pai.</p>
 <p>Tutorial GRADE...</p> <p>Arraste a peça "Transladar" até o encaixe de losango do "Objeto Gráfico" pai e mude o valor de "y" para 2.</p>	 <p>Tutorial GRADE...</p> <p>Clique no 'Cubo' posicionado e mude a cor.</p>
 <p>Tutorial GRADE...</p> <p>Arraste a peça 'Transladar' até o encaixe de losango no Objeto Gráfico.</p>	 <p>Tutorial GRADE...</p> <p>Arraste a peça 'Rotacionar' até o encaixe de losango no Objeto Gráfico.</p>
 <p>Tutorial GRADE...</p> <p>Clique no 'Transladar' posicionado e mude o valor de x para 3.</p>	 <p>Tutorial GRADE...</p> <p>Clique no 'Rotacionar' posicionado e mude o valor de y para 3.</p>

Fonte: elaborado pela autora.

APÊNDICE C – GABARITOS DOS EXERCÍCIOS DESENVOLVIDOS

Para auxiliar o aprendizado, foram desenvolvidos arquivos JSON com as respostas de cada um dos exercícios. Assim, o professor poderá aplicar os exercícios da ferramenta como uma atividade avaliativa e depois aplicar a correção, além de disponibilizá-los para revisão por parte dos alunos. Os respectivos gabaritos dos exercícios 1, 2 e 3 estão presentes em Quadro 12, Quadro 13 e Quadro 14.

Quadro 12 - Gabarito do exercício 1

```
{
  "CameraP": {
    "nome": "Câmera",
    "posicao": ["0", "0", "300"],
    "lookAt": ["0", "0", "0"],
    "fov": "14",
    "near": "100",
    "far": "600",
    "posPeca": [696.351135253906, 624.93212890625, -870.424987792969]
  },
  "ObjetoGraficoP": {
    "nome": "ObjetoGraficoP",
    "ativo": true,
    "children": [
      {
        "Cubo": {
          "nome": "Cubo",
          "tamanho": ["1", "1", "1"],
          "posicao": ["0", "0", "0"],
          "cor": "RGBA(1.000, 1.000, 1.000, 1.000)",
          "textura": "BarCode",
          "ativo": true,
          "posPeca": [697.420593261719, 618.979125976563, -870.425048828125]
        }
      },
      {
        "Iluminacao": {
          "nome": "Iluminacao",
          "tipoLuz": "Ambiente",
          "posicao": ["100", "300", "0"],
          "ativo": true,
          "posPeca": [698.855163574219, 613.408264160156, -870.403137207031],
          "cor": "RGBA(1.000, 1.000, 1.000, 1.000)"
        }
      }
    ],
    "posPeca": [697.301147460938, 621.630798339844, -870.403137207031]
  }
}
```

Fonte: elaborado pela autora.

Quadro 13 - Gabarito do exercício 2

```
{
  "CameraP": {
    "nome": "Câmera",
    "posicao": ["100", "300", "300"],
    "lookAt": ["0", "0", "0"],
    "fov": "45",
    "near": "100",
    "far": "600",
    "posPeca": [696.351135253906, 628.43212890625, -870.424987792969]
  },
  "ObjetoGraficoP": {
    "nome": "ObjetoGraficoP",
    "ativo": true,
    "children": [
      {

```



```

    "Cubo1": {
      "nome": "Cubo1",
      "tamanho": ["1","1","1"],
      "posicao": ["0","0","0"],
      "cor": "RGBA(1.000, 1.000, 1.000, 1.000)",
      "textura": "FURB",
      "ativo": true,
      "posPeca": [697.420593261719, 622.479125976563, -870.425048828125]
    }
  },
  {
    "Escalar": {
      "nome": "Escalar",
      "ativo": true,
      "valores": ["1","2","1"],
      "posPeca": [698.104431152344, 616.340454101563, -870.588928222656]
    }
  },
  {
    "Iluminacao": {
      "nome": "Iluminacao",
      "tipoLuz": "Ambiente",
      "posicao": ["100","300","0"],
      "ativo": true,
      "posPeca": [698.855163574219, 613.908264160156, -870.403137207031],
      "cor": "RGBA(1.000, 1.000, 1.000, 1.000)"
    }
  },
  {
    "ObjetoGraficoP1": {
      "nome": "ObjetoGraficoP1",
      "ativo": true,
      "children": [
        {
          "Cubo": {
            "nome": "Cubo",
            "tamanho": ["1","1","1"],
            "posicao": ["0","0","0"],
            "cor": "RGBA(1.000, 1.000, 1.000, 1.000)",
            "textura": "Madeira",
            "ativo": true,
            "posPeca": [699.0029296875, 608.582580566406, -870.425048828125]
          }
        },
        {
          "Transladar": {
            "nome": "Transladar",
            "ativo": true,
            "valores": ["2","0","0"],
            "posPeca": [700.536743164063, 605.943908691406, -870.588928222656]
          }
        }
      ],
      "posPeca": [698.883483886719, 611.234252929688, -870.403137207031]
    }
  },
  {
    "posPeca": [697.301147460938, 625.130798339844, -870.403137207031]
  }
}

```

Fonte: elaborado pela autora.

Quadro 14 - Gabarito do exercício 3

```
{
  "CameraP": {
    "nome": "Câmera",
    "posicao": ["100", "300", "300"],
    "lookAt": ["0", "0", "0"],
    "fov": "45",
    "near": "100",
    "far": "600",
    "posPeca": [696.351135253906, 624.93212890625, -870.424987792969]
  },
  "ObjetoGraficoP": {
    "nome": "ObjetoGraficoP",
    "ativo": true,
    "children": [
      {
        "Cubo": {
          "nome": "Cubo",
          "tamanho": ["1", "1", "1"],
          "posicao": ["0", "0", "0"],
          "cor": "RGBA(1.000, 1.000, 1.000, 1.000)",
          "textura": "FURB",
          "ativo": true,
          "posPeca": [697.420593261719, 618.979125976563, -870.425048828125]
        }
      },
      {
        "Rotacionar": {
          "nome": "Rotacionar",
          "ativo": true,
          "valores": ["0", "60", "0"],
          "posPeca": [698.954406738281, 619.840454101563, -870.588928222656]
        }
      },
      {
        "Iluminacao": {
          "nome": "Iluminacao",
          "tipoLuz": "Spot",
          "posicao": ["0", "300", "0"],
          "cor": "RGBA(1.000, 1.000, 1.000, 1.000)",
          "ativo": true,
          "intensidade": "1,5",
          "distancia": "1000",
          "angulo": "15",
          "expoente": "10",
          "valores": ["0", "0", "0"],
          "posPeca": [698.855163574219, 610.408264160156, -870.403137207031]
        }
      }
    ],
    "posPeca": [697.301147460938, 621.630798339844, -870.403137207031]
  }
}
```

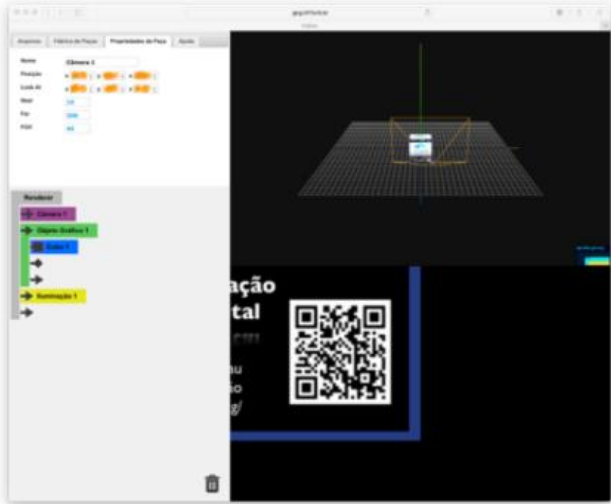
Fonte: elaborado pela autora.

ANEXO A – DESCRIÇÃO

Neste anexo, são apresentados os enunciados da lista de exercícios do professor da disciplina de CG que foi usada como base para criar os exercícios presentes na aplicação. A Figura 17 mostra o enunciado que serviu de base para criar o enunciado do Exercício 1, presente na aplicação desenvolvida.

Figura 17 - Exercício usado como base para criar o Exercício 1

1) (peso 2,5) Crie uma cena utilizando o VisEdu-CG que **somente** contenha as peças "Camera", "Objeto Gráfico", "Cubo" (com textura "Logo Grupo CG") e "Iluminação". Após altere as "Propriedades" da peça "Camera 1" no "Renderer" para mudar o ponto de vista do observador na cena. A visualização do objeto "Cubo 1" deve ser a mais próxima possível ao exibido na imagem abaixo. Neste caso utilize os valores de "Near 10", "Far 300" e "FOV 45" também descrito na figura abaixo, mudando os valores de "Posicao" e "Look At". Os valores de "Tamanho" e "Posição" do "Cubo 1" não devem ser alterados.

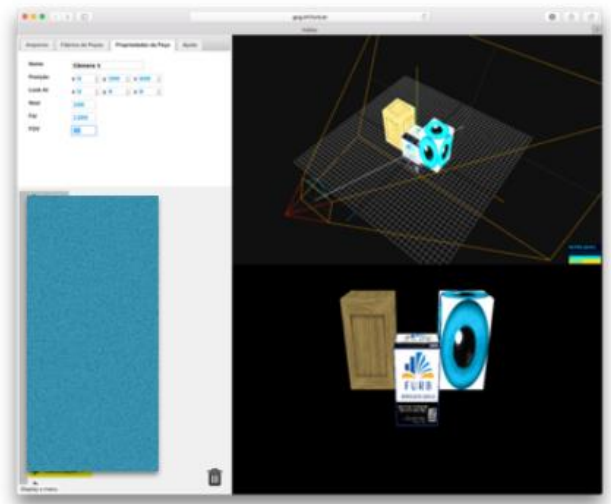


Fonte: Reis (2018).

A Figura 18 mostra o enunciado usado para criar o enunciado do Exercício 2. Este enunciado trouxe a ideia de solicitar que o usuário usasse diferentes texturas na cena.

Figura 18 - Exercício usado como base para criar o Exercício 2

3) (peso 2,5) Crie uma cena utilizando o VisEdu-CG para ter o resultado visual conforme figura abaixo. Não é permitido mudar os parâmetros dos objetos "Cubo", somente o valor da sua textura. O objeto "Cubo 1" tem o seu centro com coordenadas (0,0,0) coincidindo com a origem, os outros dois "Cubos" sofreram transformação geométrica e se encontram adjacentes ao "Cubo 1". Todos os objetos "Cubo" são ampliados somente na altura em duas vezes ao seu tamanho original usando uma transformação geométrica. Neste caso esta cena **somente** pode conter as peças: uma "Câmera", uma "Iluminação", três "Objetos Gráficos", três "Cubo", uma "Escalar" e dois "Transladar".

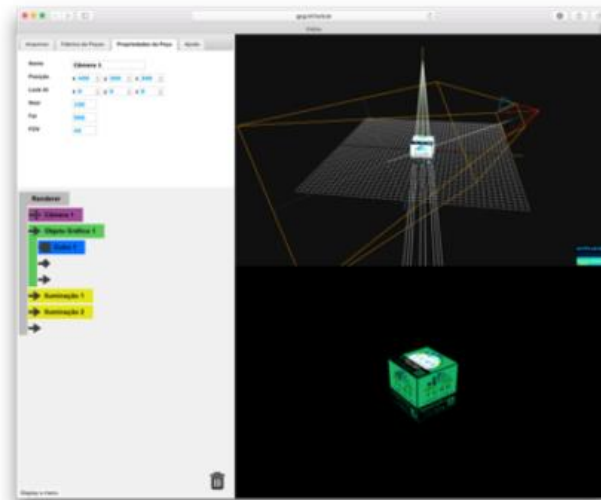


Fonte: Reis (2018).

Por fim, a Figura 19 apresenta o enunciado que foi utilizado como base para criar o enunciado do Exercício 3.

Figura 19 - Exercício usado como base para criar o Exercício 3

4) (peso 2,5) Use o arquivo "CG-04_exer_04.json" (entrada) disponível em http://gcg.inf.furb.br/visedu/cg/exercicios/CG-04_exer_04.json e com **somente** as peças descritas na figura abaixo para reproduzir o efeito visual gerado pela iluminação.



Fonte: Reis (2018).