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:** O resumo é uma apresentação concisa dos pontos relevantes de um texto. Informa suficientemente ao leitor, para que este possa decidir sobre a conveniência da leitura do texto inteiro. Deve conter OBRIGATORIAMENTE o OBJETIVO, METODOLOGIA, RESULTADOS e CONCLUSÕES. O resumo não deve ultrapassar 10 linhas e deve ser composto de uma sequência corrente de frases concisas e não de uma enumeração de tópicos. O resumo deve ser escrito em um único texto corrido (sem parágrafos). Deve-se usar a terceira pessoa do singular. As palavras-chave, a seguir, são separadas por ponto, com a primeira letra maiúscula. Caso uma palavra-chave seja composta por mais de uma palavra, somente a primeira deve ser escrita com letra maiúscula, sendo que as demais iniciam com letra minúscula, desde que não sejam nomes próprios.]

**Palavras-chave**: Ciência da computação. Sistemas de informação. Monografia. Resumo. Formato.

# 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.

# REVISÃO BIBLIOGRÁFICA

Nessa seção serão descritos os principais conceitos que servirão como base para esse projeto (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).

## CONCEITOS, TÉCNICAS e/ou FERRAMENTAS

Nessa seção 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).

### 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

### 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 o módulo necessário 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 mantem 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 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).

### 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).

## Versão anterior DO SOFTWARE

Ao longo dos anos, o VisEdu-CG já passou por diversas 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

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

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.

## TRABALHOS CORRELATOS

Essa seção 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 (Rodrigues; Gomes; Carneiro, 2022). |
| Principais funcionalidades | Encaixar blocos para atingir o objetivo de cada tarefa proposta (Rodrigues; Gomes; Carneiro, 2022). |
| Ferramentas de desenvolvimento | Blocos de materiais acessíveis e coloridos (exemplo: EVA) (Rodrigues; Gomes; Carneiro, 2022). |
| 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 (Rodrigues; Gomes; Carneiro, 2022). |

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 (Barros; Sousa; Viana, 2022). |
| 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 (Figura 2): 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 (Barros; Sousa; Viana, 2022). |
| Ferramentas de desenvolvimento | O jogo foi desenvolvido para web em Next.js e React,js (Barros; Sousa; Viana, 2022). |
| 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 obterem alto desempenho nas atividades do jogo. Os pesquisados também informaram que obtiveram sentimento de satisfação ao concluir as tarefas pré-estabelecidas com êxito (Barros; Sousa; Viana, 2022). |

Fonte: elaborado pela autora.

Quadro 3 - Trabalho Correlato 3

|  |  |
| --- | --- |
| Referência | Siedler *et al.* (2022) |
| Objetivos | Para obter o aprimoramento das técnicas de ensino sobre astronomia em sala de aula, Siedler *et al.* (2022) criaram uma plataforma com jogos para auxiliar os professores a ensinarem o tema de forma mais interessante aos alunos, promovendo engajamento (Siedler *et al.*, 2022). |
| 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 (Siedler *et al.*, 2022). 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 (Siedler *et al.*, 2022). 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) (Siedler *et al.*, 2022). |
| Ferramentas de desenvolvimento | Primeiro jogo: HyperText Markup Language 5 (HTML5), JavaScript, NodeJS e MongoDB (Siedler *et al.*, 2022).  Segundo jogo: Unity (Siedler *et al.*, 2022).  Terceiro jogo: Unity e Vuforia (Siedler *et al.*, 2022). |
| 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 (Siedler *et al.*, 2022). |

Fonte: elaborado pela autora.

# 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.

## 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).

### REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Os requisitos do projeto foram divididos em requisitos funcionais, que definem o que a ferramenta deve permitir que o usuário faça (Quadro 4), e requisitos não funcionais, 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 modo escuro) conforme melhor lhe agradar |
| RF04 | permitir que o usuário possa fazer uso da câmera com todas as suas propriedades para que funcione correta e completamente |
| 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 desktop (apenas para Windows) e web |

Fonte: elaborado pela autora.

### DIAGRAMAS

Para facilitar o processo de codificação, foram desenvolvidos diagramas de classe. A Figura 2 ilustra a relação das classes Exercício e Resposta, usadas para a funcionalidades de exercício da ferramenta. A classe Exercício 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 Exercício e Resposta

Diagrama

Descrição gerada automaticamente

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

Diagrama

Descrição gerada automaticamente

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, sugerida pelo professor orientador.

Figura 4 - Diagrama de classes Arquivo

Texto, Email

Descrição gerada automaticamente

Fonte: elaborado pela autora.

A Figura 5 represente 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

Tabela

Descrição gerada automaticamente

Fonte: elaborado pela autora.

## implementação

Para o desenvolvimento do projeto, 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#. 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, 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

Interface gráfica do usuário

Descrição gerada automaticamente

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 bem 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(); //troca o painel em visualização  break;  case 8:  StartCoroutine(apagarTela(passosTutorial[passo]));  StartCoroutine(apagarTela(painelTutorial)); //apaga o parabens  passo = 0;  grade.isOn = true; //devolve grade à visualização  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

Interface gráfica do usuário

Descrição gerada automaticamente

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 porcento acertou e, caso tenha errado algo, 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. Além disso, para contribuir com o aprendizado em aula, foram disponibilizados os gabaritos em formato JSON, tanto para correção por parte do professor quanto para revisão do exercício por parte dos alunos (o professor terá que repassar o gabarito quando achar necessário, para evitar cola durante os exercícios), 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

Interface gráfica do usuário

Descrição gerada automaticamente

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 5), 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.

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.

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).

Figura 9 - GRADE com o menu de informações aberto

Interface gráfica do usuário

Descrição gerada automaticamente

Fonte: elaborado pela autora.

Outra funcionalidade que não estava prevista, porém implícita nos requisitos (permitir que o usuário possa arrastar os blocos e editar suas informações conforme for desejado (RF)), foi a hierarquia de objeto pai e filho. Ela é responsável por permitir que qualquer propriedade do objeto pai seja herdada pelo filho (Figura 9). Entretanto, ela não foi adicionada no tutorial de criar cena, 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.

Figura 10 - Exemplo de cena com hierarquia pai e filho

Interface gráfica do usuário

Descrição gerada automaticamente

Fonte: elaborado pela autora.

# RESULTADOS

De modo a ampliar o seu caráter científico, todos os TCCs devem apresentar e discutir resultados não limitados à comparação com os trabalhos correlatos. Devem ser apresentados os casos de testes do software, destacando objetivo do teste, como foi realizada a coleta de dados e a apresentação dos resultados obtidos, preferencialmente em forma de gráficos ou tabelas, fazendo comentários sobre eles. Também é sugerida a comparação com os trabalhos correlatos apresentados na fundamentação teórica.

# CONCLUSÕES

As conclusões devem refletir os principais resultados alcançados, realizando uma avaliação em relação aos objetivos previamente formulados. Deve-se deixar claro se os objetivos foram atendidos, se as ferramentas utilizadas foram adequadas e quais as principais contribuições do trabalho sociais ou práticas para o seu grupo de usuários bem como para o desenvolvimento científico e ou tecnológico da área.

Deve-se incluir também as limitações e as possíveis extensões do TCC.

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, Karolina. 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 espacia**. 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

É fundamental que todo projeto apresente alguma forma de especificação do que foi desenvolvido. A descrição é opcional. Assim, **este apêndice deve conter os diagramas de especificação que não couberam ao longo do texto**. Os diagramas devem conter legendas numeradas na sequência do artigo.

Cada apêndice deve iniciar em uma nova página.

APÊNDICE B – XXX

Podem ser inseridos outros apêndices no artigo tais como códigos de implementação, telas de interface, instrumentos de coleta de dados, entre outros. **Apêndices são** **textos elaborados pelo autor** a fim de complementar sua argumentação. Os apêndices são identificados por letras maiúsculas consecutivas, seguidas de um travessão e pelos respectivos títulos. Deve haver no mínimo uma referência no texto anterior para cada apêndice. Colocar sempre um preâmbulo no apêndice. Caso existam tabelas ou ilustrações, identifique-as através da legenda, seguindo a numeração normal das legendas do artigo.

ANEXO A – DESCRIÇÃO

Elemento opcional, **anexos são documentos não elaborados pelo autor**, que servem de fundamentação, comprovação ou ilustração, como mapas, leis, estatutos, entre outros. Os anexos são identificados por letras maiúsculas consecutivas, seguidas de um travessão e pelos respectivos títulos. Deve haver no mínimo uma referência no texto anterior para cada anexo. Colocar sempre um preâmbulo no anexo. Caso existam tabelas ou ilustrações, identifique-as através da legenda, seguindo a numeração normal das legendas do artigo.

# DESCRIÇÃO DA FORMATAÇÃO

A seguir são apresentadas observações gerais sobre o texto do artigo do Trabalho de Conclusão de Curso (TCC). **Observa-se que esta descrição deve ser retirada do texto final.**

Na confecção do texto deve-se:

1. usar frases curtas. Segundo Teodorowitsch (2003, p. 3), “Frases com mais de duas linhas aumentam o risco de o leitor não compreender a ideia ou de entendê-la de forma equivocada.”;
2. usar linguagem impessoal (usar a terceira pessoa do singular) e verbo na voz ativa (a ação é praticada pelo sujeito), com conexão entre os parágrafos;
3. não usar palavras coloquiais;
4. não usar palavras repetidas em demasia;
5. usar verbos no presente quando for referir-se a partes do trabalho que já se encontram disponíveis no texto;
6. destacar palavras em língua estrangeira em itálico, conforme descrito abaixo:
   1. nome de software, ferramenta, aplicativo, linguagem de programação, plataforma, empresa: não deve ser escrito em itálico (exemplos: Delphi 7, Pascal, Object Pascal, Java, JavaScript, Java 2 Micro Edition, Basic, Microsoft Visual C++, C, Windows, Linux, MySQL, Oracle, Eclipse 3.0, Enterprise Architect, Rational Rose, Microsoft, Sun Microsystems),
   2. citações: o sobrenome do autor ou o nome da instituição responsável pela autoria do documento citado não deve ser escrito em itálico (exemplo: Segundo Sun Microsystems (2004), ...),
   3. palavras em língua estrangeira encontradas nos dicionários nacionais: não devem ser grafadas em itálico (exemplos: software, hardware, web, Internet),
   4. demais palavras em língua estrangeira: devem ser escritas em itálico (exemplos: *palmtop*, *classpath*, *play*, etc.). No entanto, Teodorowitsch (2003, p. 7), sugere que alguns termos em língua inglesa devem ser substituídos por termos em português (exemplos: núcleo em vez de *kernel*, aprendizagem de máquina em vez de *machine learning*, etc.);
7. observar as seguintes regras quanto ao uso de siglas:
   1. colocar as siglas entre parênteses precedidas pela forma completa do nome, quando aparecem pela primeira vez no texto (exemplos: Associação Brasileira de Normas Técnicas (ABNT), Trabalho de Conclusão de Curso (TCC)). Caso exista uma lista de siglas na parte pré-textual do volume final, pode-se usar somente a sigla, quando aparecer pela primeira vez no texto,
   2. usar apenas a sigla nas demais ocorrências no texto,
   3. escrever as siglas em letras maiúsculas e não usar itálico,
   4. escrever o plural das siglas sem apóstrofo (exemplos: PCs, APIs, PDAs) e determinar o gênero da sigla conforme o gênero do primeiro substantivo do seu nome (exemplo: o TCC – o Trabalho de Conclusão de Curso).

## formatação

A formatação geral para apresentação do documento, descrita na NBR 14724 (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2011), é a seguinte:

1. o texto divide-se em capítulos, seções e subseções (até cinco divisões);
2. a apresentação de citações em documentos deve seguir a NBR 10520 (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2002b);
3. a descrição das referências bibliográficas deve estar de acordo com a NBR 6023 (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2002a).

Observa-se ainda que todo capítulo, seção ou subseção deve ter no mínimo um texto relacionado.

O artigo deve ser digitado usando as fontes e formatação de parágrafos deste modelo, indicadas no Quadro 2.

Quadro 8– Estilos do modelo

|  |  |
| --- | --- |
| **USO** | **FORMATO** |
| título de capítulo ou seção primária (1) | TF-TÍTULO 1 (Times New Roman, 10pt, negrito, maiúsculas) |
| título de seção secundária (1.1) | TF-TÍTULO 2 (Times New Roman, 10pt, maiúsculas) |
| título de seção terciária (1.1.1) | TF-Título 3 (Times New Roman, 10pt, minúsculas, exceto a 1a letra da 1a palavra do título e de nomes próprios) |
| título de seção quaternária (1.1.1.1) | TF-Título 4 (mesma formatação seção ternária) |
| título de seção quinária (1.1.1.1.1) | TF-Título 5 (mesma formatação seção ternária) |
| texto | TF-TEXTO (Times New Roman, 10pt) |
| citação direta com mais de três linhas | TF-CITAÇÃO (Times New Roman, 9pt, com recuo de 4 cm) |
| itens (alíneas) | ver descrição abaixo (Times New Roman, 10pt) |
| referência bibliográfica | TF-REFERÊNCIA ITEM (Times New Roman, 10pt, alinhada à margem esquerda) |
| fonte, legenda, texto de quadro/tabela e figura | TF-FONTE (Times New Roman, 9pt, centralizada)  TF-LEGENDA, (Times New Roman, 10pt, centralizada)  TF-TEXTO- QUADRO (Times New Roman, 10pt)  TF-FIGURA (Times New Roman, 10pt, centralizada) |

Fonte: elaborado pelo autor.

O espaçamento, também definido no modelo, deve ser conforme indicado no Quadro 3.

Quadro 9 - Espaçamento

|  |  |
| --- | --- |
| **USO** | **ESPAÇAMENTO** |
| título de capítulo ou seção primária (1)  título de seção secundária (1.1)  título de seção terciária (1.1.1)  título de seção quaternária (1.1.1.1)  título da seção quinária (1.1.1.1.1) | espaço simples, com 12pt antes do parágrafo |
| texto | espaço simples, com 6 pt antes do parágrafo |
| citação direta com mais de três linhas | espaço simples com 6pt antes e depois do parágrafo |
| itens (alíneas) | espaço simples, com 6 pt antes do parágrafo |
| referência bibliográfica | espaço simples, com 6 pt antes do parágrafo |
| legenda e texto de ilustração/tabela | espaço simples, com 6 pt antes do parágrafo |
| fonte | espaço simples, com 0pt antes do parágrafo |

Fonte: elaborado pelo autor.

Na disposição gráfica de itens (alíneas) devem ser observados os seguintes quesitos:

1. o texto que antecede os itens termina com dois pontos;
2. cada item deve iniciar com uma letra minúscula seguida de fecha parênteses e terminar com um ponto e vírgula, sendo que o último item termina com ponto (FORMATO: TF-ALÍNEA);
3. o texto de cada item inicia com letra minúscula, exceto nomes próprios;
4. quando contiver subitens, os mesmos devem iniciar com hífen colocado sob a primeira letra do texto do item correspondente (FORMATO: TF-SUBALÍNEA nível 1 ou TF-SUBALÍNEA nível 2, conforme o caso). Nesse caso, cada subitem deve terminar com uma vírgula, exceto o último que termina com ponto ou com ponto e vírgula.

Segue um exemplo:

1. cada item inicia com letra minúscula, cada item inicia com letra minúscula, cada item inicia com letra minúscula, cada item inicia com letra minúscula (FORMATO: TF-ALÍNEA);
2. cada item inicia com letra minúscula, cada item inicia com letra minúscula, cada item inicia com letra minúscula, cada item inicia com letra minúscula (FORMATO: TF-ALÍNEA):
   1. cada subitem (nível 1) inicia com letra minúscula, cada subitem (nível 1) inicia com letra minúscula (FORMATO: TF-SUBALÍNEA nível 1):
      1. cada subitem (nível 2) inicia com letra minúscula, cada subitem (nível 2) inicia com letra minúscula (FORMATO: TF-SUBALÍNEA nível 2),
      2. cada subitem (nível 2) inicia com letra minúscula, cada subitem (nível 2) inicia com letra minúscula (FORMATO: TF-SUBALÍNEA nível 2),
   2. cada subitem (nível 1) inicia com letra minúscula, cada subitem (nível 1) inicia com letra minúscula (FORMATO: TF-SUBALÍNEA nível 1);
3. cada item inicia com letra minúscula, cada item inicia com letra minúscula, cada item inicia com letra minúscula, cada item inicia com letra minúscula (FORMATO: TF-ALÍNEA).

#### Exemplo de título de seção quaternária [FORMATO: TF-TÍTULO 4]

Formato: TF-TEXTO.

##### Exemplo de título de seção quinária [FORMATO: TF-TÍTULO 5]

Formato: TF-TEXTO.

### Formatação de quadros, figuras e tabelas

Um quadro contém apenas informações textuais, que podem ser agrupadas em colunas. Uma figura contém, além das informações textuais, pelo menos um elemento gráfico. Uma tabela é uma apresentação tabular de informações **numéricas** relacionadas.

Os quadros, figuras e tabelas são identificados na parte superior por uma legenda (a qual deve estar centralizada) composta pela palavra designativa (Figura, Quadro ou Tabela, conforme o caso), seguida de seu número em algarismo arábico (usar numeração progressiva, uma sequência para os quadros, outra para as figuras e outra para as tabelas), de hífen e do título. As ilustrações devem:

1. aparecer centralizadas no texto;
2. estar delimitadas por uma moldura simples (com exceção das tabelas não quais não devem ser usadas bordas (linhas) verticais em suas extremidades);
3. aparecer numa única página (quando o tamanho não exceder o da página), inclusive a legenda;
4. serem inseridas o mais próximo possível do trecho a que se referem pela primeira vez.

Toda ilustração deve ter fonte, centralizada. Quando foi o próprio autor que fez a ilustração, deve inserir o texto: “Fonte: elaborado pelo autor”.

Observa-se que quando um código fonte for descrito dentro de um quadro, deve-se utilizar letra do tipo courier new 9pt. (TF-CÓDIGO-FONTE)

Exemplos de como se deve referenciar uma figura, um quadro e uma tabela bem como descrevê-los são mostrados a seguir.

Um exemplo de uma rede de Petri pode ser visto na Figura 1.

Figura 11 – Exemplo de uma rede de Petri



Fonte: Schubert (2003, p. 18).

Um exemplo de código fonte gerado a partir de uma especificação pode ser visto no Quadro 4.

Quadro 10 – Funções que verificam se as transições estão sensibilizadas

|  |
| --- |
| function TestruturaMalha.T1Sensibilizada: Boolean;  begin  result := (Fp2 and Fp4);  end;  function TEstruturaMalha.T2Sensibilizada: boolean;  begin  result := (Fp1 and Fp3);  end;  function TEstruturaMalha.T3Sensibilizada: boolean;  begin  result := (Fp2 and Fp4);  end; |

Fonte: Schubert (2003, p. 63).

A quantidade de trabalhos finais realizados no Curso de Ciência da Computação (de 2010 até 2014) é apresentada na Tabela 1.

Tabela 1 – Trabalhos finais realizados no Curso de Ciência da Computação

|  |  |  |  |
| --- | --- | --- | --- |
| Ano | Estágios | TCC´s | Totais |
| 2010/1 | 0 | 16 | 16 |
| 2010/2 | 0 | 21 | 21 |
| 2011/1 | 0 | 25 | 25 |
| 2011/2 | 0 | 23 | 23 |
| 2012/1 | 0 | 23 | 23 |
| 2012/2 | 0 | 22 | 22 |
| 2013/1 | 0 | 25 | 25 |
| 2013/2 | 0 | 16 | 16 |
| 2014/1 | 0 | 18 | 18 |
| 2014/2 | 0 | 13 | 13 |
|  | **0** | **202** | **202** |

Fonte: elaborado pelo autor.

### Exemplos de citações retiradas de documentos ou de nomes constituintes de uma entidade

A apresentação de citações em documentos deve seguir a NBR 10520 (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2002b). O sistema a ser usado é o alfabético. Exemplos de citações são: “Numa publicação recente (SEBESTA, 2000) é exposto ...” e “Segundo Silva *et al.* (1987), execução controlada de programas é ...”.

Quando a citação referir-se a uma parte específica do documento consultado, especificar no texto do artigo a(s) página(s). Esta(s) deverá(ão) seguir a data, separada(s) por vírgula(s) e precedida(s) pelo designativo que a(s) caracteriza(m). Como exemplo, mostra-se: “(SCHIMT, 1999, p. 50)” ou “... visto que Schimt (1999, p. 50) implementou ...”.

As citações diretas (transcrição textual de parte da obra do autor consultado), no texto, com mais de três linhas, devem ser destacadas com recuo de 4 cm da margem esquerda, com letra menor que a do texto utilizado e sem as aspas (FORMATO: TF-CITAÇÃO), conforme o exemplo a seguir.

A Associação Brasileira de Normas Técnicas (ABNT) é o Fórum Nacional de Normalização. As Normas Brasileiras, cujo conteúdo é de responsabilidade dos Comitês Brasileiros (ABNT/CB) e dos Organismos de Normalização Setorial (ABNT/ONS), são elaboradas por Comissões de Estudo (CE), formadas por representantes dos setores envolvidos, delas fazendo parte: produtores, consumidores e neutros (universidades, laboratórios e outros). (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2002b, p. 1).

Quando da citação de um nome (identificador) constituinte de uma entidade ou de um elemento de interface em um texto, deve-se utilizar o tipo de letra *courier new*, com tamanho nove (9). Para facilitar a formatação, existe o estilo de palavra denominado TF-COURIER9. Como exemplo cita-se nome de classe, atributo ou método. A seguir são apresentados exemplos.

As classes TTabelaTransicao e TExpressaoRegular são classes de interface, porém estão sendo consideradas como classes de domínio da aplicação.

Ao clicar no botão Confirmar, o software abre uma nova tela.