REFATORANDO E ADICIONANDO NOVAS FUNCIONALIDADES NO VISEDU-CG COM MOTOR DE JOGOS UNITY

Douglas Eduardo Bauler, 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

dbauler@furb.br, dalton@furb.br

Resumo: *Este artigo apresenta o processo de desenvolvimento das novas funcionalidades e a refatoração do VisEdu-CG, ferramenta de ensino-aprendizagem para os alunos de computação Gráfica. Desenvolvido no motor de jogos Unity na linguagem C# através da ferramenta VisualStudio. Entre as novas funcionalidades, estão as peças Poligono, Spline e Iteração. Apresenta também uma documentação de ajuda, exportação e importação de um cenário construído com bloqueio de campos e realiza diversos ajustes de comportamentos das peças na ferramenta. Realiza testes de usabilidade da ferramenta nas plataformas WebGL e Windows, a fim de verificar o desempenho e o consumo de memória. Em sua aplicação apresentou alguns aumentos poucos significativos nos navegadores, porém demonstrou um ótimo desempenho no ambiente desktop.*

Palavras-chave*: Ensino aprendizagem. Unity. Computação Gráfica. Iteração. Refatoração.*

1 INTRODUÇÃO

A tecnologia está em constante evolução de maneiras muito significativas, melhorando o dia a dia, aumentando a produtividade e o entendimento de vários assuntos. Existem ferramentas interativas de ensino que dão interlúdio ao assunto a ser abordado, tornando-o de uma maneira mais lúdica na qual muitas vezes pode dificultar o aprendizado. Conforme descreve Monteiro e Nantes (2021, p.2), estes métodos de ensino-aprendizagem engajam os alunos nos seus estudos, contribuindo na qualidade de ensino. Segundo esses autores, "na era tecnológica, esse desafio se sobressai exigindo agilidade, principalmente, por parte de professores, para aprenderem a lidar com tanta inovação de uma só vez".

> A contribuição didática para uma pedagogia voltada para o sujeito requer assumir, entre outras coisas, o uso das mídias e das tecnologias da educação. O professor deve ser capaz de utilizar aparatos tecnológicos não apenas para seu uso próprio, mas trabalhar com esses recursos em sala de aula, em favor da aprendizagem dos alunos (SILVA, 2011, p.6).

Por meio dessas metodologias a ferramenta VisEdu-CG tem como objetivo trazer essas melhorias no aprendizado aos acadêmicos da matéria de Computação Gráfica. Conforme Reis (2011, apud BUTTENBERG, 2020, p. 1), "o VisEdu-CG é um projeto para desenvolver uma plataforma Web que permita os alunos da disciplina de Computação Gráfica do curso de Ciências da Computação praticarem os conceitos ministrados nesta disciplina", estando atualmente na versão 5.0 do projeto.

Essa aplicação contou com o desenvolvimento de vários módulos específicos, dentre eles pode-se citar o motor de jogos, matemática, estatística e simulação. Para que a ferramenta tenha uma evolução constante foi realizado um processo de migração de linguagem e refatoração do código. Uma das motivações para fazer a refatoração foi em relação à algumas funcionalidades do WebGL, o que torna o VisEdu-CG um sistema igualmente limitado (BUTTENBERG, 2020).

Um processo de migração de uma ferramenta já consolidada numa linguagem, não é um processo simples de realizar, devido à complexidade em diversas funcionalidades da ferramenta, tendo também o curto espaço de tempo para seu desenvolvimento e a falta de estrutura do código para melhor entendimento e manutenção, fazendo com que a migração não seja realizada completamente.

Em razão dessas dificuldades, este trabalho desenvolveu a continuação do processo de migração das funcionalidades, assim como a refatoração do código já migrado para melhor compreensão, manutenção e adição de novas funções como bloqueio de campos ao serem exportados e adicionar uma nova peça denominada como Iteração, utilizando a motor de jogos Unity. Os objetivos específicos são: permitir criar atividades em forma de exercícios práticos e disponibilizar novas peças do tipo Iteração, Polígonos e Spline.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção são destacados os principais assuntos do desenvolvimento da ferramenta. Primeiramente são apresentados os conceitos utilizados para o desenvolvimento como: Refatoração e Coroutines. Em seguida é apresentada a versão anterior do sistema, o VisEdu-CG 5.0. Por fim, são apresentados os trabalhos correlatos.

2.1 REFATORAÇÃO DE CÓDIGO

A refatoração é o processo de modificar um sistema de software de modo que não altere o comportamento externo do código, embora melhore a sua estrutura interna. É uma maneira disciplinada de reorganizar o código, minimizando as chances de introduzir *bug*. Em sua essência, ao refatorar, aperfeiçoará o design do código depois que ele foi escrito (FOWLER, 2020).

> Refactorings são modificações realizadas em um software preservando seu comportamento e visando exclusivamente a melhoria de seu código ou projeto. São exemplos de refactorings operações como renomeação de um método ou variável (para um nome mais intuitivo e fácil de lembrar), divisão de um método longo em dois métodos menores (para facilitar o entendimento) ou movimentação de um método para uma classe mais apropriada (VALENTE, 2020, c.9, p.12).

É arriscado ser feita uma refatoração, se exige mudanças que podem introduzir a *bugs* sutis em um código que está funcionando. A implementação se não for feita de forma adequada, pode-se fazer atrasar em dias ou até semanas. Ao começa a explorar o código, logo se descobre novas oportunidades para alterá-lo, à medida que o analisa com mais detalhes. Quanto mais se avalia, surgem novos detalhes da necessidade de mudanças a serem feitas (FOWLER, 2020).

Padrões de projetos e refatoração estão amplamente conectados, visando garantir uma melhor compreensão e fácil manutenção de código de um projeto específico. Conforme afirma Rapeli (2006), os padrões de projetos favorecessem a implementações mais eficientes tendo clareza e fácil entendimento do código, em casos de sistemas não projetados em seu uso, é possível aplicá-los sem alterar suas funcionalidades existentes.

2.2 COROUTINES

`Coroutines` é um recurso disponível no Unity permitindo dispersar rotinas em vários quadros, na maioria das rotinas quando for chamado um método ele é executado até a sua conclusão. Enquanto as `coroutines` pode-se pausar a sua execução utilizando o comando `yield` conforme suas validações, para retornar o método deixando pausado, executando a partir da linha na qual foi pausado até o fim da rotina. Em animações processuais ou sequência de eventos ao longo tempo, é recomendado utilizá-lo (UNITY, 2021c). O Quadro 1 demonstra sua utilização na ferramenta, onde é utilizado para realizar uma animação da peça deslocando-se até sua origem, por não ser válida para encaixe.

Quadro 1 – Método para remover uma peça não encaixada corretamente utilizando Coroutines

```
                                                    ...
IEnumerator RemovePeca()
{
    while ((transform.position.y != startPos.y && transform.position.x != startPos.x))
    {
        transform.position = Vector3.Lerp(transform.position, startPos, Time.deltaTime *
Consts.SPEED_DESLOC / Vector3.Distance(transform.position, startPos));

         yield return null;
    }
    Destroy(gameObject);
}
                                                    ...
```

Fonte: elaborado pelo autor.

Esse recurso foi utilizado para realizar animação das peças ao serem encaixadas ou desencaixadas dependendo da sua validação. Já na manipulação da peça `Iteracao` foi utilizado em paralelo as `coroutines` o método `WaitForSeconds` onde é passado como parâmetro a quantidade de segundos que a rotina deve esperar para terminar sua execução. O uso das `coroutines` permitiu melhorar a legibilidade do código e da própria animação da peça sendo iterada conforme a sua transformação na qual está encaixado. Porém deve-se destacar que a utilização indevida desse mecanismo pode gerar mais consumo de memória.

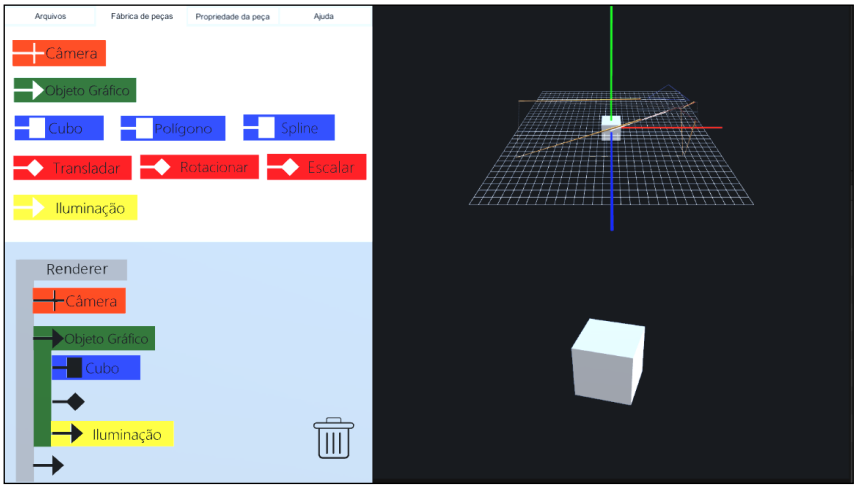
2.3 VERSÃO ANTERIOR DA FERRAMENTA

O objetivo principal da versão 5.0 da ferramenta VisEdu-CG foi realizar a migração para utilizar o motor gráfico Unity. Apesar de apresentar problemas de visualização em alguns objetos, como por exemplo, a iluminação *spot*, a plataforma teve resultado satisfatório na construção de cenários contendo conceitos básicos de computação gráfica, como as transformações geométricas e em conceitos com maior complexidade como é o caso das iluminações (BUTTENBERG, 2020).

O segundo objetivo, foi criar uma proposta de tutorial informativo de maneira simples, em que a criação de uma cena destacasse os conceitos essenciais. Buttenberg (2020) salienta que o tutorial pode ser melhorado, como por exemplo a forma de exibição dos passos no tutorial. Além disso, algumas peças ainda não foram migradas como: o Polígono, o Spline e as melhorias na implementação da peça iluminação.

O terceiro objetivo, de utilizar representações visuais a partir de peças de encaixe para gerar uma cena gráfica, foi atingido. As peças importadas de uma ferramenta de criação de modelos 3D se comportaram adequadamente no Unity e os encaixes das peças nos slots foram bem-sucedidos. Quase todas as peças tiveram suas representações gráficas efetuadas, com exceção das peças spline e polígono. Ele destaca essas peças a serem adicionadas nas próximas versões. Além das funções de `look at`, `near` e `far` da câmera (BUTTENBERG, 2020). A Figura 1 demonstra a tela principal da ferramenta na versão 5.0.

Figura 1 - Tela principal do VisEdu-CG 5.0



Fonte: Buttenberg (2020, p. 15).

Com relação a conclusão do resultado da ferramenta, Buttenberg (2018) destaca um nível no consumo de memória maior na maioria dos navegadores, com exceção do Google Chrome sendo estável. A melhor plataforma em desempenho foi na versão desktop Windows. Por este motivo, ele recomenda gerar executáveis não apenas para a versão web, e sim para as demais plataformas, visando a melhor resolução de desempenho da ferramenta e tornando-a multiplataforma. Sendo assim, também disponibilizou um tutorial com base nas funcionalidades disponíveis da versão, atendendo seus principais requisitos do projeto, como a migração, podendo manipular peças como: a câmera, o objeto gráfico, o cubo e iluminação, com exceção do Polígono, Spline e outras funcionalidades, como exportação/importação de projetos e a guia de ajuda. Além de que o tutorial inicialmente implementado não trata de todas as funções já desenvolvidas.

2.4 TRABALHOS CORRELATOS

São apresentados três trabalhos correlatos com características semelhantes aos objetivos do trabalho proposto. O primeiro trabalho é uma ferramenta chamada Duolingo (Quadro 2), aplicativo para auxílio de aprendizado de múltiplas linguagens e multiplataforma, sendo utilizado em forma m jogo com desafios diários, metas e recompensas com o objetivo de estimular o estudo de outras línguas. O segundo trabalho é o QuestMeter (Quadro 3), conforme descreve Vieira (2019), é uma ferramenta de quiz construída com elementos de gamificação juntamente com o conceito de Clickers. O terceiro trabalho é o Toweljs (Quadro 4), conforme descreve Zanluca (2018), é um motor gráfico que utiliza JavaScript e WebGL, com objetivo de facilitar a implementação e abstrair o uso dessas duas ferramentas.

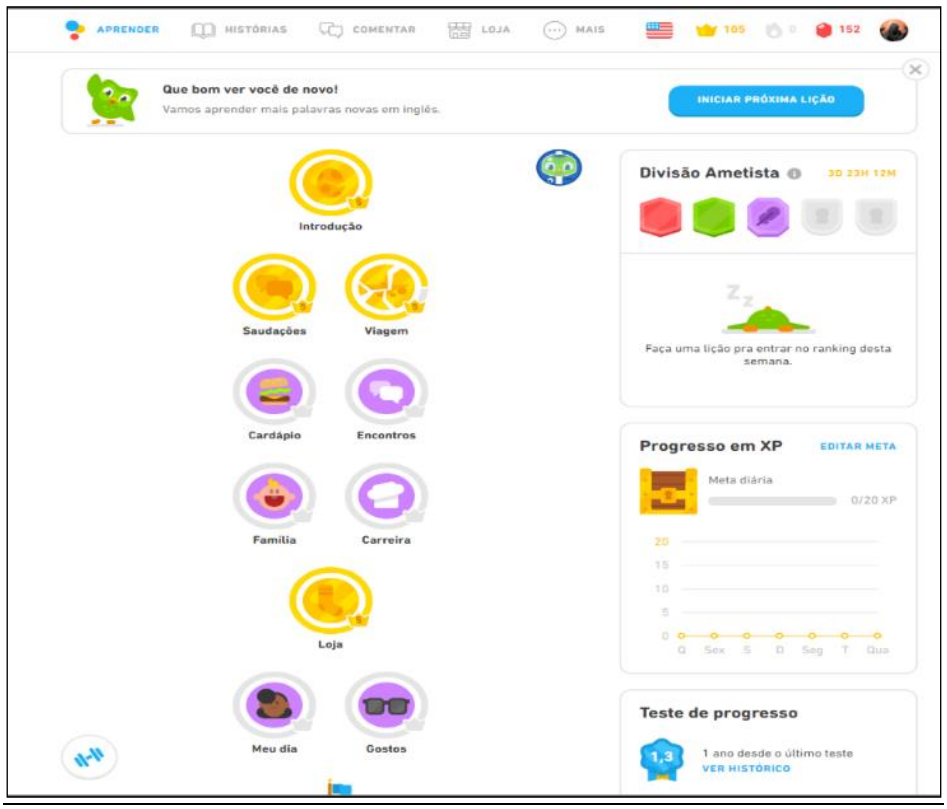
Quadro 2 – Duolingo

Referência	Melo (2021).
Objetivos	Aplicativo com objetivo na aprendizagem de idiomas em formato de jogo.
Principais funcionalidades	Dispondo de regras, pontuações, moedas e punições, estimulando atividades diárias de curta duração (em torno de vinte minutos podendo-se ser definido pelo usuário) premiando-o regularmente.
Ferramentas de desenvolvimento	Não consta.
Resultados e conclusões	Segundo MELO (2021), a experiência com o Duolingo no ensino formal de alemão durante um semestre letivo em um curso de Letras/Alemão com estudantes em níveis iniciais de aprendizagem mostrou-se relevante para ampliar o vocabulário do idioma.

Fonte: elaborado pelo autor.

Nas versões mais atuais, o Duolingo oferece um ambiente interativo tendo uma progressão de atividades realizadas. Cada exercício a ser realizado possui um tutorial explicando pronúncias e como serão abordadas as questões sobre o assunto em questão. A versão gratuita possui três vidas, ou seja, errando três atividades será obrigatório ter que aguardar elas serem recarregadas. Caso o usuário seja pagante, suas vidas e tentativas serão ilimitadas. Ao longo de todos os testes é sempre possível realizar uma tentativa de aptidão da linguagem, para verificar seu nível de desempenho no conhecimento de novas palavras, por exemplo. Na Figura 2 é possível visualizar a tela principal do Duolingo na versão web.

Figura 2 - Tela principal do aplicativo Duolingo na versão Web



Fonte: Duolingo (2021).

Quadro 3 – QuestMeter

Referência	Vieira (2019).
Objetivos	Auxiliar os professores na realização de atividades diversificadas para motivar e engajar os alunos em sala de aula.
Principais funcionalidades	O aplicativo possui dois papeis, cada um possuindo funcionalidades diferentes. O papel de professor na ferramenta é manter a funcionalidades como: atividades, questões dentro das atividades, respostas dentro das questões, gerar turmas dentro de atividades, apresentar a atividade criada para os alunos e controlar o andamento da apresentação. Enquanto o papel do aluno é ingressar em atividades disponibilizadas, escolher as opções oferecidas em cada questão da atividade, visualizar as respostas escolhidas, podendo ver as respostas corretas e verificar seu progresso na ferramenta na tela de perfil.
Ferramentas de desenvolvimento	Framework Ionic e a plataforma Firebase.
Resultados e conclusões	Segundo a autora foi concluído que, foram cumpridos os objetivos definidos, embora os resultados de usabilidade, engajamento e motivação, obtidos tenham sido razoáveis. Os *feedbacks* recebidos dos alunos e dos professores durante os testes e comentários disponíveis nos formulários, foram positivos em sua maioria.

Fonte: elaborado pelo autor.

O QuestMeter é uma ferramenta de quiz construída com elementos de gamificação com o conceito de Clickers, utilizando o framework Ionic e a plataforma Firebase. Com o objetivo de auxiliar os professores na realização de atividades diversificadas para motivar e engajar os alunos em sala de aula, além disso, outro propósito da ferramenta é testar a interação dos alunos com ferramentas diferenciadas em sala (VIEIRA, 2019).

Quadro 4 - Toweljs

Referência	Zanluca (2018).
Objetivos	Aumentar o nível de abstração na implementação entre o motor de jogos utilizando JavaScript e WebGL.
Principais funcionalidades	Disponibiliza a criação de objetos gráficos (cubos e esferas) e luzes permitido juntar tudo numa cena, permite também a criação de dois tipos diferentes de câmera sintética (perspectiva e ortogonal).
Ferramentas de desenvolvimento	JavaScript e WebGL.
Resultados e conclusões	Segundo o autor, descreve que dentre as dificuldades durante o desenvolvimento, se destaca a forma como foram implementadas as transformações geométricas e as luzes. Para evitar o processamento desnecessário do recalculo da matriz de transformação do objeto a cada vez que o desenhasse, optou-se por recalculá-la somente quando realmente houvesse alguma alteração nos seus valores.

Fonte: elaborado pelo autor.

Segundo Zanluca (2018), Toweljs é uma implementação de um motor de jogos utilizando JavaScript e WebGL, tendo como principal objetivo facilitar a implementação e aumentar o nível de abstração para aplicações desenvolvidas utilizando essas duas ferramentas. O motor por sua vez, disponibiliza a criação de objetos gráficos (cubos e esferas) e luzes permitido juntar tudo numa cena, permite também a criação de dois tipos diferentes de câmera sintética (perspectiva e ortogonal). Tudo isso utilizando uma arquitetura baseada em componentes, que ajudou na organização e facilitará futuras expansões do código.

3 DESCRIÇÃO DA FERRAMENTA

Nesta seção é apresentado como foi desenvolvido a ferramenta. Na primeira seção é realçado as especificações, atentando aos requisitos funcionais e diagrama de sequência da utilização da nova peça `Iteracao`. Em seguida, na segunda seção são apresentadas as principais técnicas utilizadas na implementação.

3.1 ESPEFICICAÇÃO

Os novos Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF) em relação ao sistema anterior são apresentados nos Quadro 5 e Quadro 6 respectivamente e foram utilizados como suporte a implementação da ferramenta.

Quadro 5 – Requisitos Funcionais

RF01	Permitir importar/exportar atividades em forma de exercícios com a opção de bloqueio de campos
RF02	Disponibilizar guia de Ajuda o qual documenta as funcionalidades disponíveis da ferramenta
RF03	Desenhar novos componentes dos tipos Iteração, Polígono e Spline

Fonte: elaborado pelo autor.

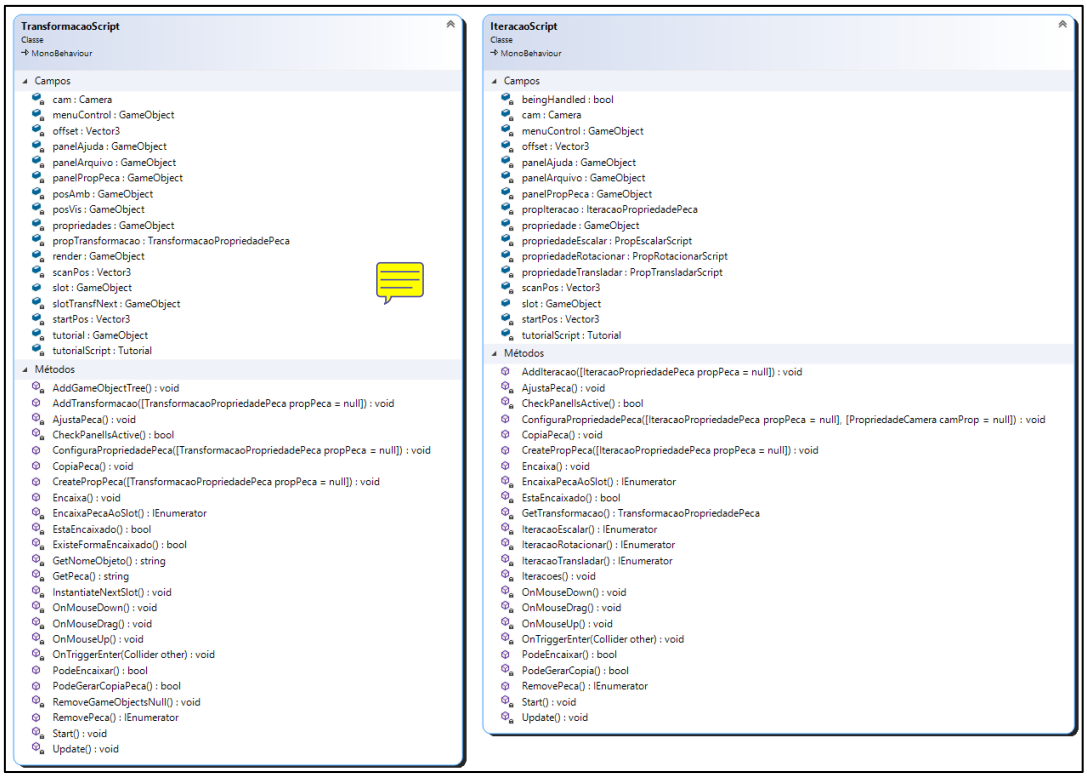
Quadro 6 – Requisitos Não Funcionais

RNF01	Ser desenvolvida na linguagem C#
RNF02	Utilizar motor de jogos Unity

Fonte: elaborado pelo autor.

A Figura 3 ilustra o diagrama de classes das peças do tipo Transformação (`Rotacionar`, `Transladar`, `Escalar`) e `Iteracao` nova peça adicionada nesta versão, na qual somente é possível encaixá-la em uma transformação encaixada. O APÊNDICE A representa os demais diagramas das classes da ferramenta.

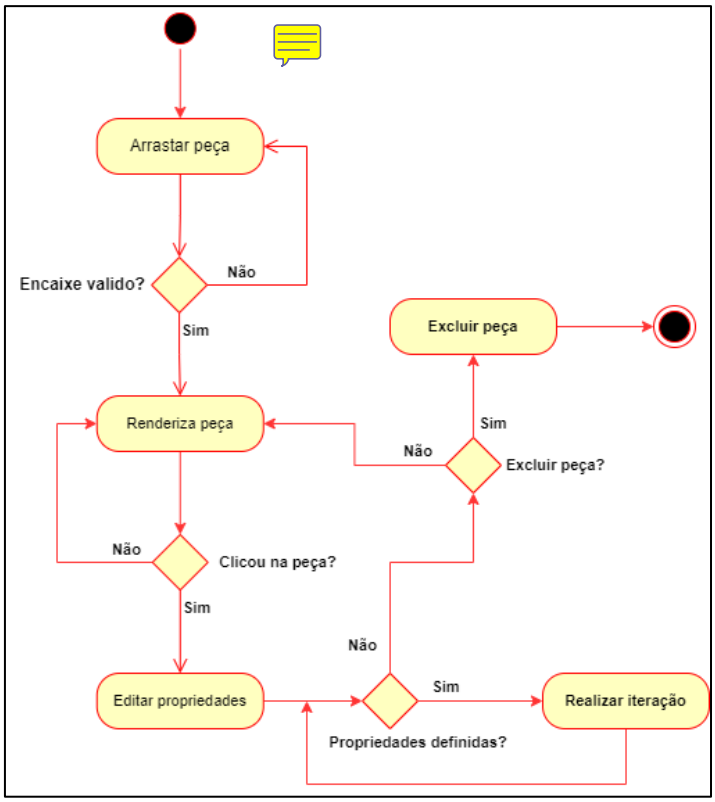
Figura 3 – Diagrama de classes das peças do tipo transformação e Iteração



Fonte: elaborado pelo autor.

A Figura 4 representa o diagrama de atividades da nova peça `Iteracao` implementada na ferramenta tendo como objetivo demonstrar a aplicação das peças de transformações sobre as peças do tipo formas disponíveis.

Figura 4 - Diagrama de atividades da peça Iteração

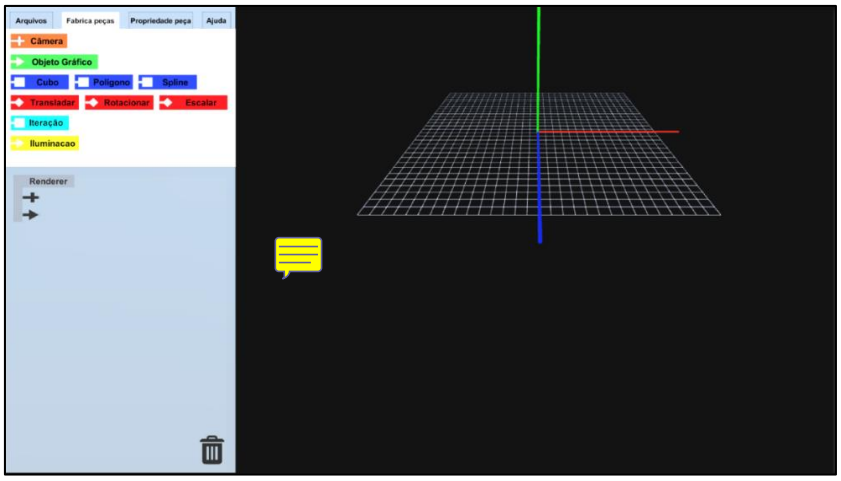


Fonte: elaborado pelo autor.

3.2 IMPLEMENTAÇÃO

A ferramenta implementada tem a tela dividida em quatro painéis, sendo o primeiro painel contendo os seguintes menus: `Arquivo`, `Fábrica de peças`, `Propriedades das peças` e `Ajuda`. O segundo painel é denominado como `Renderer`, que é utilizado para manter os encaixes das peças e exclusão. Enquanto o terceiro painel tem a função de representar as peças graficamente, tendo como base os encaixes definidos no segundo painel. Por fim, o painel `Visualizador` é responsável apenas em demonstrar apenas a forma geométrica das peças dispostas no terceiro painel, conforme suas configurações. A Figura 5 apresenta uma visão geral da tela principal da ferramenta.

Figura 5 – Tela principal da ferramenta



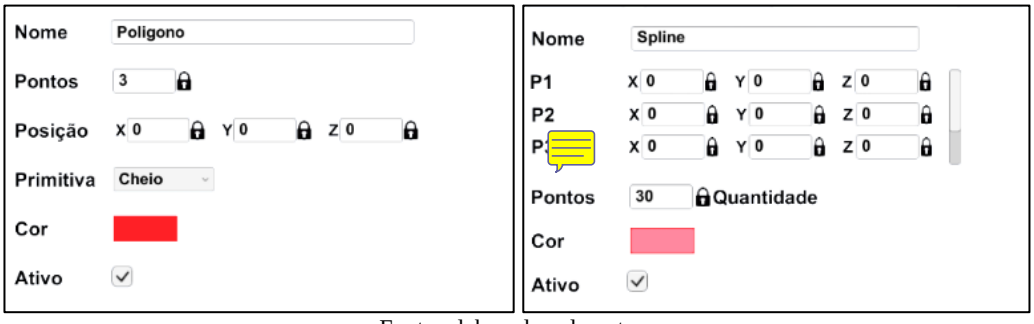
Fonte: elaborado pelo autor.

Conforme descreve Buttenberg (2020), ao abrir a ferramenta é exibida a seguinte mensagem: “Deseja realizar o tutorial para aprender os conceitos básicos da ferramenta?”, podendo ser respondido com os botões “Sim” e “Pular tutorial”, na qual o segundo botão é exibido a cada passo executado se deseja pular o tutorial. Se o botão “Sim” for selecionado em cada passo, ao totalizar dez, serão demonstradas algumas formas de encaixes e definições de algumas propriedades das peças disponíveis a serem encaixadas.

A `Fábrica de peças` dispõe um total de dez peças, na qual podem ser utilizadas no `Renderer`. A primeira é a `Câmera`, a segunda `Objeto Gráfico`, a terceira `Cubo`, a quarta `Poligono`, a quinta `Spline`, a sexta, sétima e oitava são as peças de transformações geométricas, são elas: `Transladar`, `Rotacionar`, `Escalar` respectivamente, a nona peça `Iteracao` responsável pelas iterações das transformações com as peças e a última peça `Iluminacao`.

Cada peça possui suas únicas propriedades, como pode ser observado na Figura 6. A peça `Poligono` possui o nome (propriedade em comum com todas as peças), pontos, a primitiva (tendo opções como: Vértices, Aberto, Fechado, Preenchido), cor, a posição (x, y, z) e ativo (funcionalidade para ativar ou não a peça, além da sua exibição no ambiente gráfico). Ao passo que a peça `Spline` possui `nome`, pontos de controle (enumerados como: `P1`, `P2`, `P3`, `P4`, `P5`), `cor` e `ativo`.

Figura 6 – Propriedades das peças Polígono e Spline



Fonte: elaborado pelo autor.

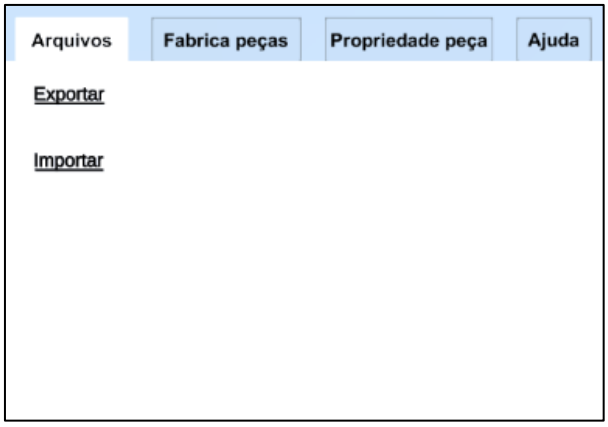
A Iteração compreende um conjunto de propriedades diferentes dependendo do tipo de transformação geométrica encaixado na peça, podendo ser do tipo `Transladar`, `Rotacionar` e `Escalar`. Suas propriedades são: `Nome`, posições `X`, `Y`, `Z` (`Intervalo`, `Min`, `Max`) e `Ativo`. Onde o intervalo é definido como a quantidade de valor a ser incrementado na posição a qual pertence. O `Min` é o valor mínimo a ser inicializado quando o valor de incremento chega ao valor máximo e o `Max` é o valor máximo que pode ser incrementado. A Figura 7 representa as propriedades da peça Iteração.

Figura 7 – Propriedades da peça Iteração

Fonte: elaborado pelo autor.

O painel `Arquivo` tem como principais funcionalidades: importação e exportação de um projeto. Caso seja selecionado a opção de exportação, será aberto uma tela de seleção de diretório com um nome sugestivo do arquivo definido como padrão (`ProjectVisEdu.json`). Ao selecionar o diretório será consistido o processo de exportação na estrutura em `json`, conforme as peças estejam dispostas nos encaixes do pa`Renderer`. Na hipótese de ser selecionar a opção de importação, será aberta a tela de seleção de diretório no qual se deve selecionar o projeto a ser importado. Ao selecionar e clicar em abrir, realizar-se-á o processo de importação conforme a estrutura do projeto, quando o mesmo já estiver definido nos encaixes, será descartado para importação completa do projeto. A Figura 8 demonstra a tela de exportação/importação de arquivos.

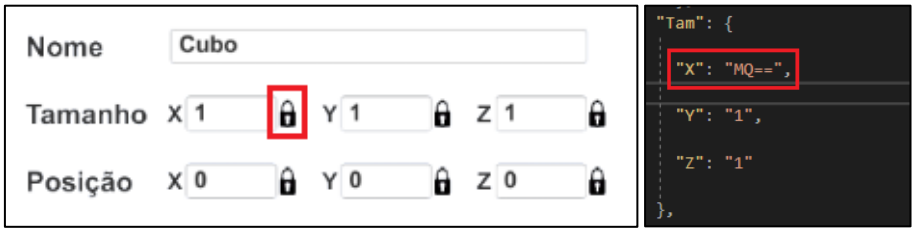
Figura 8 – Tela de arquivos



Fonte: elaborado pelo autor.

Na exportação será verificado uma propriedade chamada de bloqueio de campos, na qual tem como objetivo bloquear as propriedades das peças embaralhando seus valores, impedindo com que seja alterado ou visualizado pelo usuário. Em situação de importação do projeto, o valor dos campos, caso estejam bloqueados, é realizado o processo de desembaralhamento. A Figura 9 demonstra o comportamento do bloqueio dos campos, representado por um ícone de cadeado.

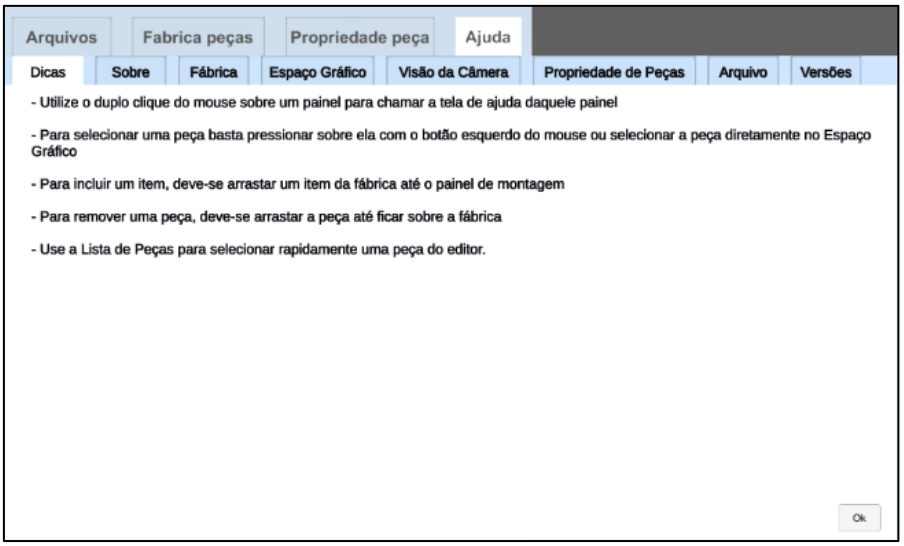
Figura 9 – Campo bloqueado e embaralhado no arquivo Json exportado



Fonte: elaborado pelo autor.

O painel de `Ajuda` contém as informações com o objetivo de documentação das principais funcionalidades da ferramenta. Sendo separada pelos menus: `Dicas`, `Sobre`, `Fábrica`, `Espaço Gráfico`, `Visão da Câmera`, `Propriedade de Peças`, `Arquivo` e `Versões`. Cada um deles documentando e auxiliando as respectivas funcionalidades. Na Figura 10 é demonstrado a documentação do menu `Dicas`.

Figura 10 – Tela de ajuda



Fonte: elaborado pelo autor.

No desenvolvimento da ferramenta foi utilizado o motor de jogos Unity na linguagem de programação C# no ambiente de desenvolvimento Visual Studio 2019 para implementação dos `scripts`. Sendo refatorado as peças gráficas, propriedades e comportamentos, foram adicionadas novas peças, como: `Iteracao`, `Poligono` e `Spline`. Também inseridos os mecanismos de exportação/importação do projeto desenvolvido além da funcionalidade de ajuda, uma documentação de todas as funcionalidades disponíveis na ferramenta.

Na versão anterior da ferramenta, todos os comportamentos das peças disponíveis eram implementados apenas no script `Controller`, realizando a verificação de qual peça estava sendo manipulada a cada vez que fosse executado alguma rotina, gerando impacto no desempenho com o Unity pelo fato de ser verificado em cada atualização da tela. Mas o principal ponto negativo está na centralização dos comportamentos e da dificuldade na compreensão, manutenção e rastreabilidade de problemas, caso sejam necessários ajustes na ferramenta.

Portanto, foi realizado a refatoração de todas as peças disponíveis, separando seus comportamentos em seus próprios scripts, como por exemplo as peças do tipo formas: `CuboScript`, `SplineScript` e `PoligonoScript`, para melhor manutenibilidade e compreensão do código da ferramenta, organizando seus comportamentos específicos e sendo utilizados apenas quando a peça estiver encaixada no `Renderer`. Todas as peças foram recriadas a partir de um objeto tipo painel dos componentes UI (User interface) não sendo mais necessário utilizar um objeto tipo `Cubo`, pelo fato das peças serem apenas para visualização e representação dos encaixes que são desenhados no `Visualizador` e `Ambiente Grafico`.

Em relação aos objetos encaixados, na versão anterior eram realizados cálculos das posições dos objetos das peças em relação aos `slots`, definindo constantes para encaixarem adequadamente aos seus respectivos `slots`. Estes por sua vez, como eram definidos de maneira estática, caso fossem redimensionados na tela exibida, as peças não se encaixariam adequadamente, sendo necessário recalcular as posições e suas constantes. Por esses motivos foram refatorados os comportamentos dos encaixes das peças, sendo todos removidos e utilizados nos componentes `VerticalLayoutGroup` e `HorizontalLayoutGroup`. O `VerticalLayoutGroup` coloca seus componentes filhos um embaixo do outro de maneira vertical, conforme suas respectivas alturas mínimas. Enquanto o `HorizontalLayoutGroup` possui o mesmo comportamento, porém sendo ordenado de maneira horizontal. Utilizando estes componentes, não se faz necessário o cálculo e nem o uso das constantes para as posições, entretanto, é obrigatório apenas definir a posição da peça com a mesma do seu `slot`, mas caso haja um redimensionamento da cena, os objetos irão se ajustar conforme definido.

Para implementação das funcionalidades de exportação e importação de um projeto, foram utilizados os recursos da biblioteca `JsonUtility` do Unity, na qual a função `ToJson` serializa o objeto passado como parâmetro, sendo a ferramenta uma classe do projeto definido como `ProjectVisEduClass`. Conforme a sua estrutura, será formatado para o arquivo do tipo Json, destacando que todas as classes devem estar marcadas com o atributo de serialização declarado como `Serializable`. Alguns atributos foram marcados como `NonSerialized` para não serem serializados, por não haver necessidade de comporem a estrutura. No Quadro 7 é demonstrado como resulta a estrutura do projeto em formato `json` após sua exportação.

Quadro 7 – Exemplo formato do arquivo Json

```
                                    ...
"ObjetosGraficos": [
    {
        "Propriedades": {
            "Nome": "ObjetoGraficoP",
            "Cor": { "r": 1.0, "g": 1.0, "b": 1.0, "a": 1.0 },
            "Ativo": true
        },
        "Cubo": {
            "Propriedades": {
                "Nome": "Cubo",
                "Cor": { "r": 1.0, "g": 1.0, "b": 1.0, "a": 1.0 },
                "Ativo": false,
                "NomeCuboAmbiente": "CuboAmb1",
                "NomeCuboVis": "CuboVis1",
                "Pos": { "X": "0", "Y": "0", "Z": "0" },
                "Tam": { "X": "0", "Y": "0", "Z": "0" },
                "Textura": { "instanceID": 0 },
        "Poligono": {
             "Propriedades": {
                 "Nome": "",
                 "Cor": { "r": 0.0, "g": 0.0, "b": 0.0, "a": 0.0 },
                 "Ativo": false,
                 "PoligonoAmb": "",
                 "PoligonoVis": "",
                 "Pos": { "X": "", "Y": "", "Z": "" },
                 "Pontos": "",
                 "Primitiva": 3
             },
                                    ...
```

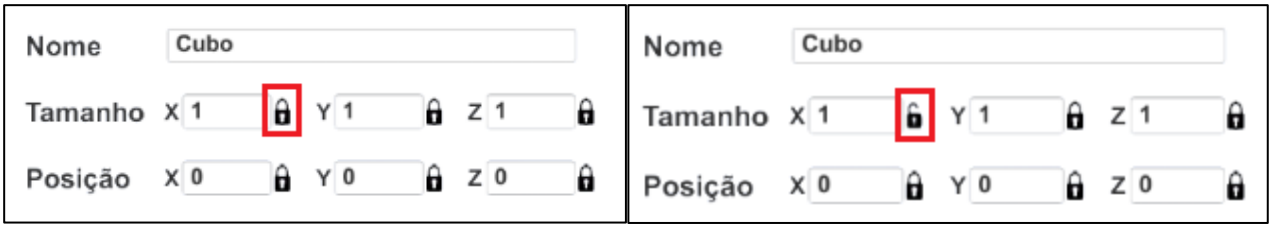
Fonte: elaborado pelo autor.

Como visualizado no Quadro 7 a estrutura do `ObjetoGrafico` dividiu as formas `Cubo`, `Spline` e `Poligono` em atributos específicos, mesmo permitido apenas um tipo de forma. O motivo dessa estrutura de classes, foi por conta da limitação da biblioteca `JsonUtility` que não trata adequadamente os conceitos de herança e polimorfismo. Já em uma situação que tenha uma classe do tipo das formas em foram herdados alguns tipos como: `Cubo`, `Spline` ou `Poligono`, a biblioteca apenas consideraria a classe pai e seus atributos, descartando as classes filhas. Dessa forma foi definida a estrutura da classe do projeto, mesmo sendo adicionados objetos vazios no arquivo.

Para a funcionalidade de importação de um projeto foi utilizado a mesma biblioteca da exportação, com a diferença da utilização de outra função definida como `FromJson`, na qual é passado como parâmetro o arquivo `json` extraído pelas funções disponíveis da linguagem C#. Com isso é retornado a classe `ProjectVisEduClass` com todas as definições necessárias para a importação do projeto, adicionando objetos gráficos, formas, transformações e iterações.

Conforme a Figura 11, foi implementado um novo recurso definido como bloqueio de campos, o qual quando clicado no cadeado, localizado ao lado do campo das propriedades das peças, será representado como fechado, concluindo que o campo deve ser bloqueado para exportação do projeto e caso esteja aberto, o campo não será bloqueado. Nesta função de bloqueio do campos é realizada uma verificação no momento da exportação, para que quando o campo estiver marcado como bloqueado, ele seja exportado e embaralhado utilizando a biblioteca da linguagem C# a função `Encoding.UTF8.GetBytes`. Dessa forma, ele retornará em formato `base64` com o objetivo de dificultar a visualização dos valores dos campos no arquivo exportado. Enquanto na importação é utilizado a função `System.Convert.FromBase64String` da mesma biblioteca, convertendo o valor do campo de `base64` para o formato padrão do atributo. Destacando que todos os atributos da classe `ProjectVisEduClass` são do tipo `string` para poder utilizar as funções do formato `base64`.

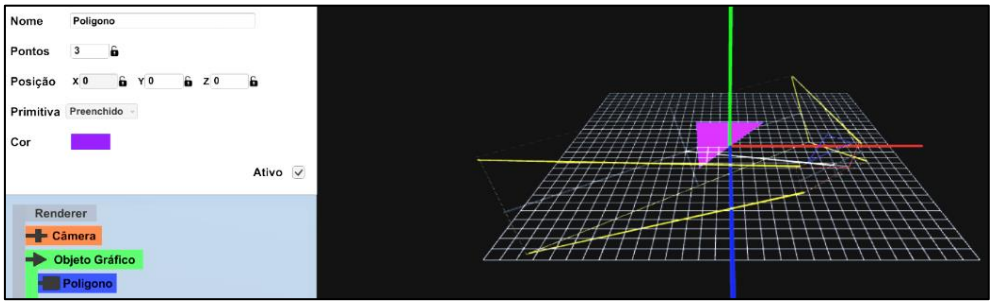
Figura 11 - Representação do cadeado bloqueado e desbloqueado nas propriedades de uma peça



Fonte: elaborado pelo autor.

O comportamento da peça `Poligono` foi adicionado nesta versão, porque como mencionado por Buttenberg (2020), a peça estava apenas desenhada no objeto `FabricaPecas`, porém não era possível ser manipulada. Foram utilizado os recursos do componente `MeshRenderer` para serem desenhados no polígono no painel `Ambiente Grafico`, na qual os pontos do polígono inicialmente constituídos com três tem a finalidade de utiliza-los na função `Triangulator`, melhorando a disposição dos pontos na malha do `Poligono`. Conforme Figura 12 é demostrada as propriedades disponíveis na peça e sua visualização no ambiente gráfico conforme definido.

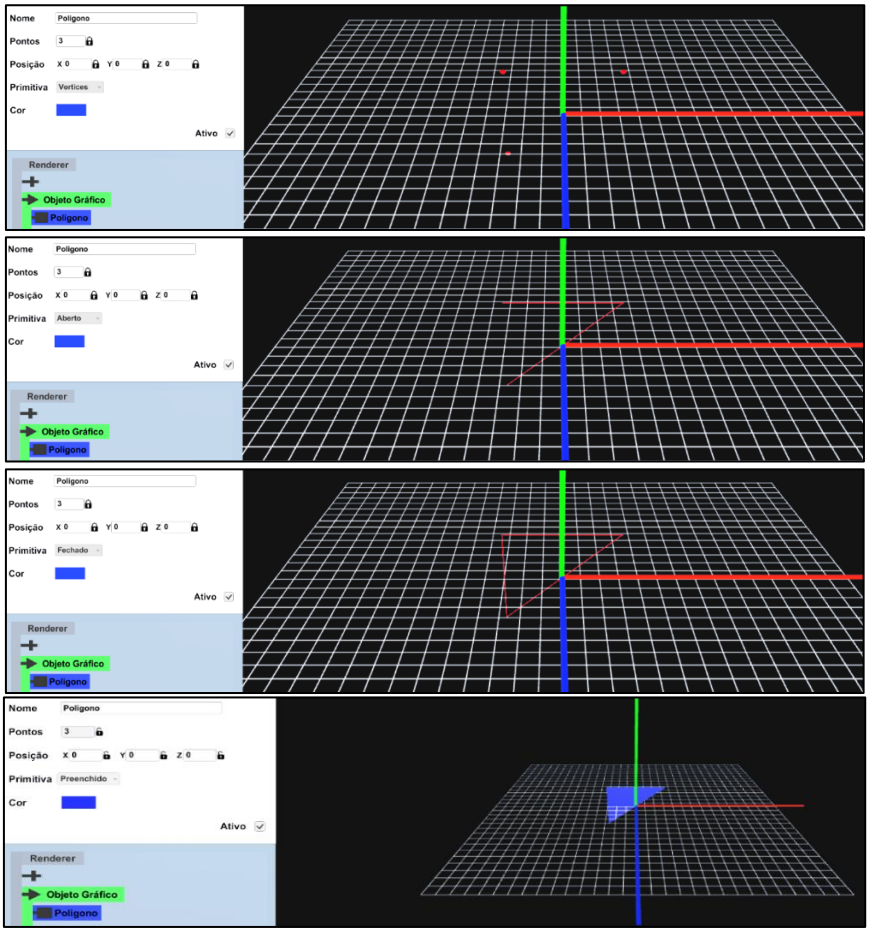
Figura 12 - Propriedades do polígono e sua relação ao ambiente gráfico



Fonte: elaborado pelo autor.

Destacam-se as propriedades `Pontos` e `Primitiva`, na qual os pontos são as quantidades de vértices desenhados na malha da peça `Poligono`, utilizando o componente `MeshRenderer`. Já a propriedade denominada `primitiva`, é a representação do polígono no ambiente gráfico, tendo as suas opções como: `Vértices`, `Aberto`, `Fechado` e `Preenchido`. Na opção `Vértices`, caso seja selecionado, o polígono será apenas desenhado os vértices. Com a opção `Aberto` são desenhados os contornos do polígono, com a exceção da última conexão entre os vértices. Entretanto no `Fechado`, apenas são desenhadas as conexões dos vértices sem o preenchimento da malha da peça. Por fim, na opção `Preenchido`, o polígono é representado com todo o preenchimento da malha e dos pontos. Na Figura 13 é demonstrada as configurações conforme o tipo de primitiva.

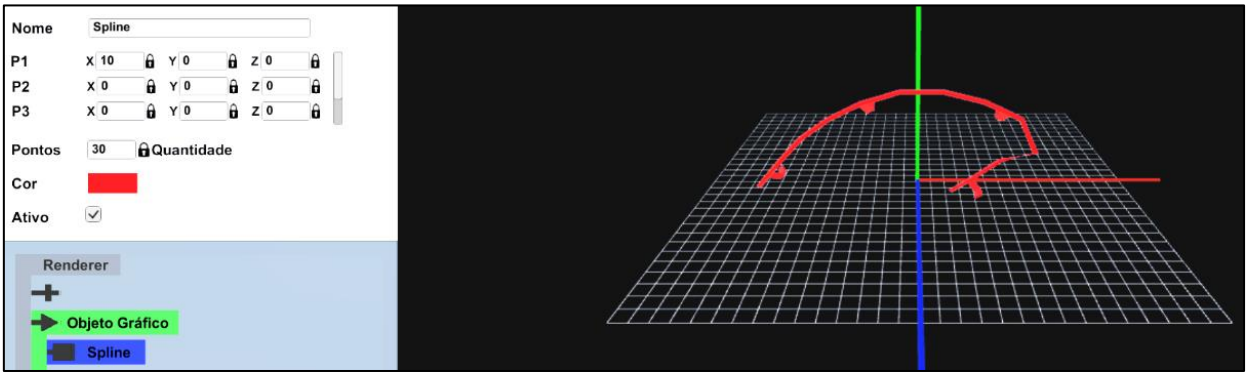
Figura 13 - Peça Polígono desenhado conforme as primitivas Vértices, Aberto, Fechado e Preenchido



Fonte: elaborado pelo autor.

Em seguida, foi realizada a implementação dos comportamentos da peça `Spline` utilizando o componente `SplineMesh` (UNITY, 2021d), obtido gratuitamente pela loja do Unity. As propriedades na qual a peça dispõe são: posição (`X`, `Y`, `Z`), pontos de controle com suas posições (`P1`, `P2`, `P3`, `P4`, `P5`), quantidade de pontos, cor e ativo. Onde destacam-se os pontos de controle que possibilitam a mudança de suas posições em relação ao ambiente gráfico. Enquanto a quantidade de pontos, revela o nível da curvatura entre os pontos de controle, ou seja, quanto maior a quantidade de pontos, menor será a percepção de que a `spline` é formada por segmentos de reta, mas a curva continuará passando pelos pontos de controle da `spline`. A Figura 14 representa uma peça `Spline` no ambiente gráfico conforme suas propriedades.

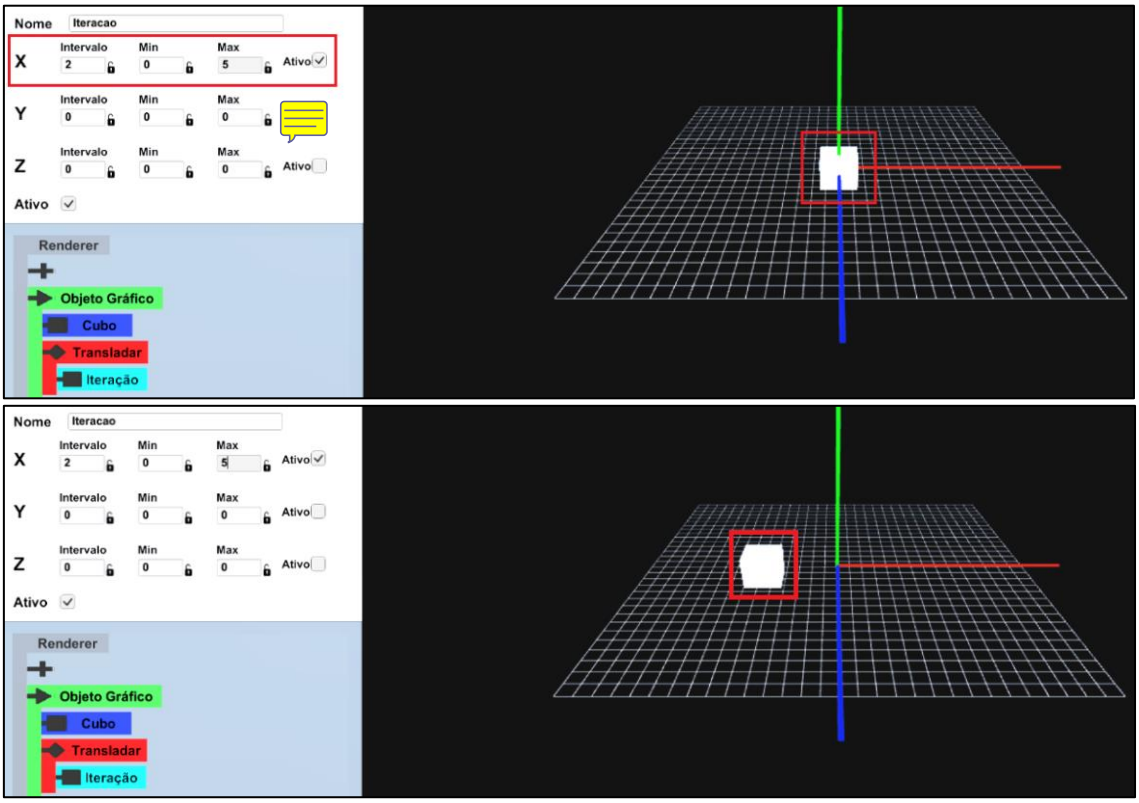
Figura 14 – Peça Spline no ambiente gráfico conforme suas propriedades



Fonte: elaborado pelo autor.

A última peça implementada na ferramenta foi a `Iteracao`, que possui as propriedades `Intervalo`, `Min` e `Max` sendo relativas as posições das transformações (`Rotacionar`, `Transladar`, `Escalar`) encaixadas. O `Intervalo` é responsável pelo incremento dos valores das posições em que são atualizadas a cada segundo. Em seguida, o `Min` é o valor mínimo a ser definido na posição, enquanto o `Max` é o valor máximo em que a posição possa atingir. Caso seja atingido o valor máximo da posição ele é redefinido ao valor mínimo. Esta peça tem como principal objetivo demonstrar de maneira iterativa a relação das transformações nas formas. Na Figura 15 é demonstrado o comportamento da peça `Iteração` conforme suas propriedades.

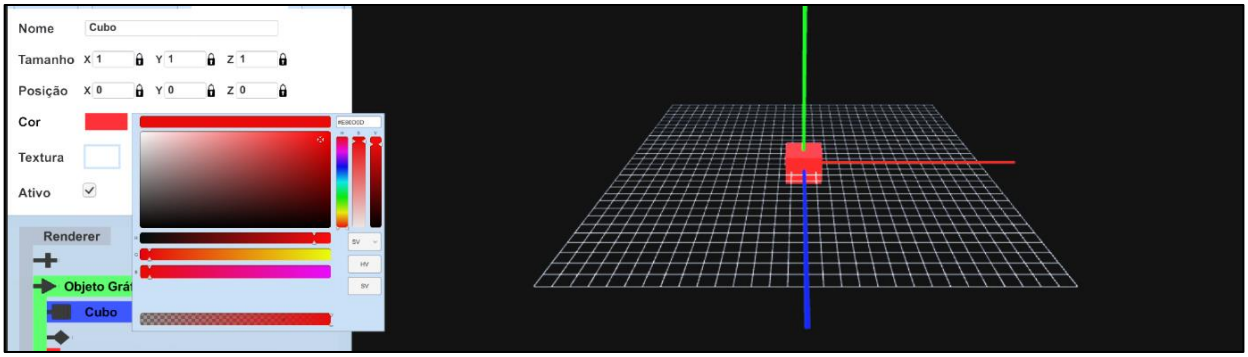
Figura 15 – Manipulação da peça Iteração sobre a peça Cubo



Fonte: elaborado pelo autor.

Ao longo da implementação da ferramenta, foram adicionados alguns componentes auxiliares gratuitos da loja do Unity, para melhorar a manipulação e os comportamentos dos objetos. Entre eles está o `FlexibleColorPicker` (UNITY, 2021a), um componente que possui o objetivo de representar um seletor `RGB` de cores, no qual o valor da cor definida no componente está localizado no atributo `color`. Ao selecionar a cor será definida a coloração do material da forma. A Figura 16 demonstra a utilização do componente em relação a peça `Cubo`.

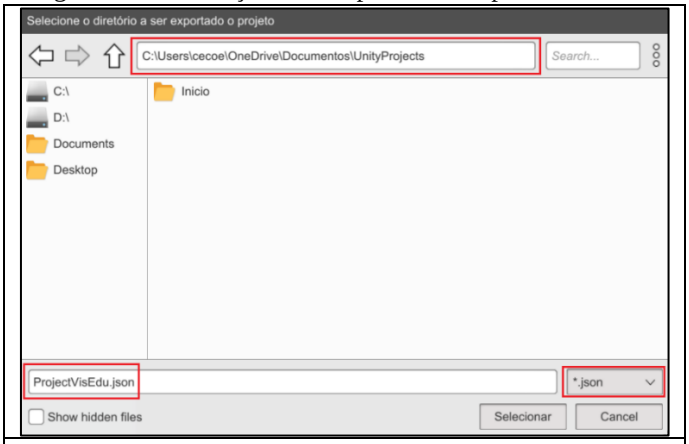
Figura 16 – Utilização do componente FlexibleColorPicker



Fonte: elaborado pelo autor.

Outro componente gratuito utilizado foi o `SimpleFileBrowser` (UNITY, 2021b), utilizado na ferramenta com a mesma função do explorador de arquivos de um sistema operacional, por exemplo, realizando a chamada do método estático `FileBrowser.ShowLoadDialog`. A Figura 17 detalha os recursos do componente com objetivo de manipular arquivos no formato conforme os seus parâmetros.

Figura 17 – Utilização do componente SimpleFileBrowser




```
...
void OnMouseDown()
{
    Util_VisEdu.EnableColliderFabricaPecas(false, false);
    Util_VisEdu.EnableGOAmbienteGrafico(false);
    Util_VisEdu.EnableGOVisualizador(false);
    try
    {
        FileBrowser.AllFilesFilterText = "*.json";
        FileBrowser.ShowLoadDialog(
           onSuccess: (path) => { GenerateProject(); },
           onCancel: nu
           pickMode: Fi          owser.PickMode.Files,
           allowMultiSelection: false,
           initialPath: null,
           initialFilename: Consts.INITIAL_FILE_PROJECT,
           title: Consts.TITLE_SELECT_EXPORT_DIRECTORY,
           loadButtonText: Consts.BTN_SELECT_PROJECT
        );
    }
    finally
    {
        Util_VisEdu.EnableColliderFabricaPecas(true);
        Util_VisEdu.EnableGOAmbienteGrafico(true);
        Util_VisEdu.EnableGOVisualizador(true);
    }
}
...
```

Fonte: elaborado pelo autor.

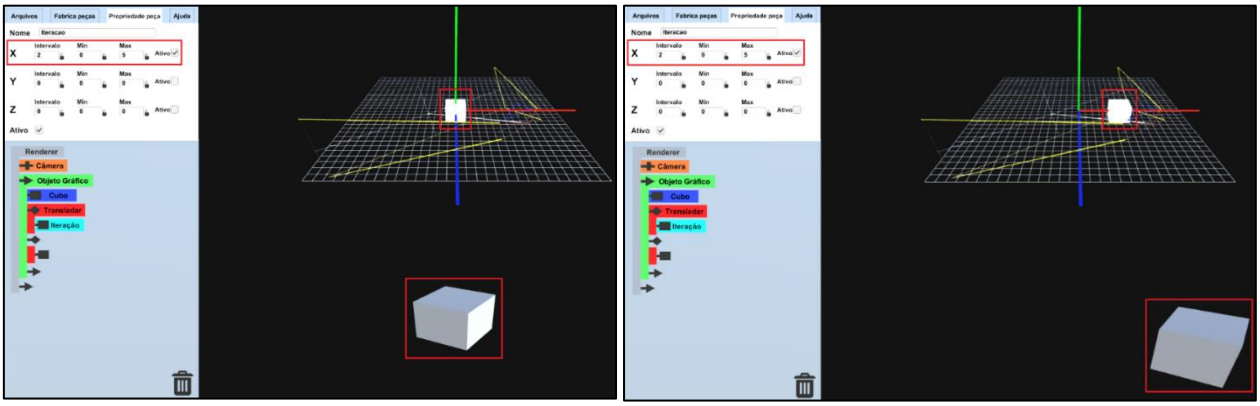
4 RESULTADOS

Esta seção apresenta os resultados obtidos nos testes na ferramenta e foi dividida em duas partes. A primeira traz o relato de uso, para detecção de falhas, na qual o usuário possa se deparar. Já na segunda documenta os testes de desempenho.

4.1 RELATOS DE USO

A Figura 18 demonstra a utilização da peça `Iteracao` em que está encaixada em uma transformação do tipo `Transladar` e por fim embu na peça `Cubo`. Utilizando o recurso de `Coroutines` disponível pelo Unity, a peça realiza a iteração incrementando a posição `X` na transformação. Por exemplo, caso a peça esteja no valor máximo definido nas propriedades a posição volta ao valor mínimo, realizando assim um `loop` na peça `Iteracao`.

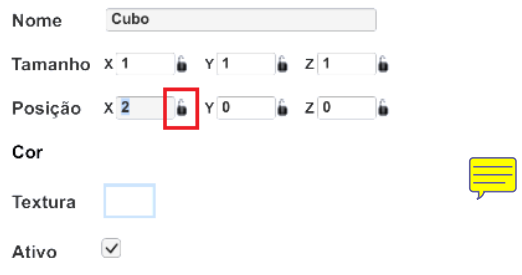
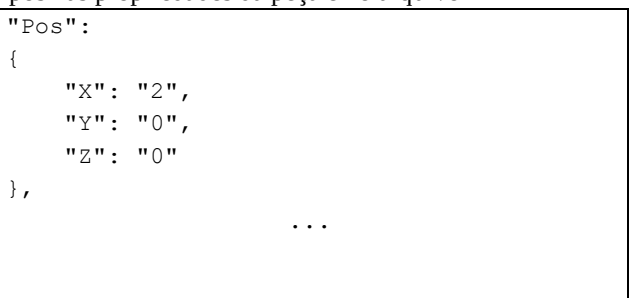
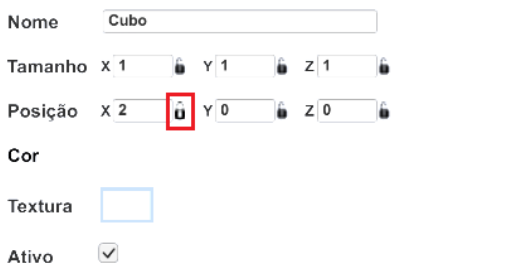
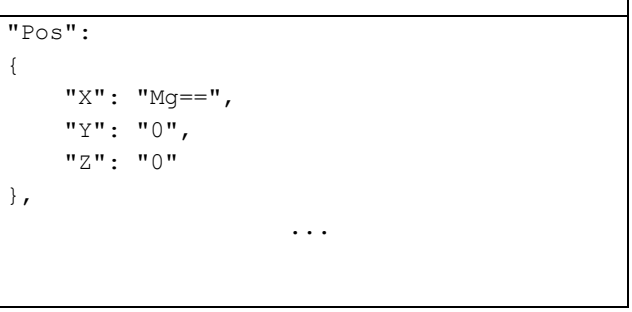
Figura 18 – Criação de um cenário de testes utilizando a peça Iteração na plataforma Windows



Fonte: elaborado pelo autor.

No Quadro 8 foi realizada comparação das propriedades das peças utilizando o recurso de bloqueio de campos, na qual corresponde ao embaralhamento do valor da posição `X` no arquivo `json`, utilizando o recurso de conversão em `base64` da própria biblioteca do C#.

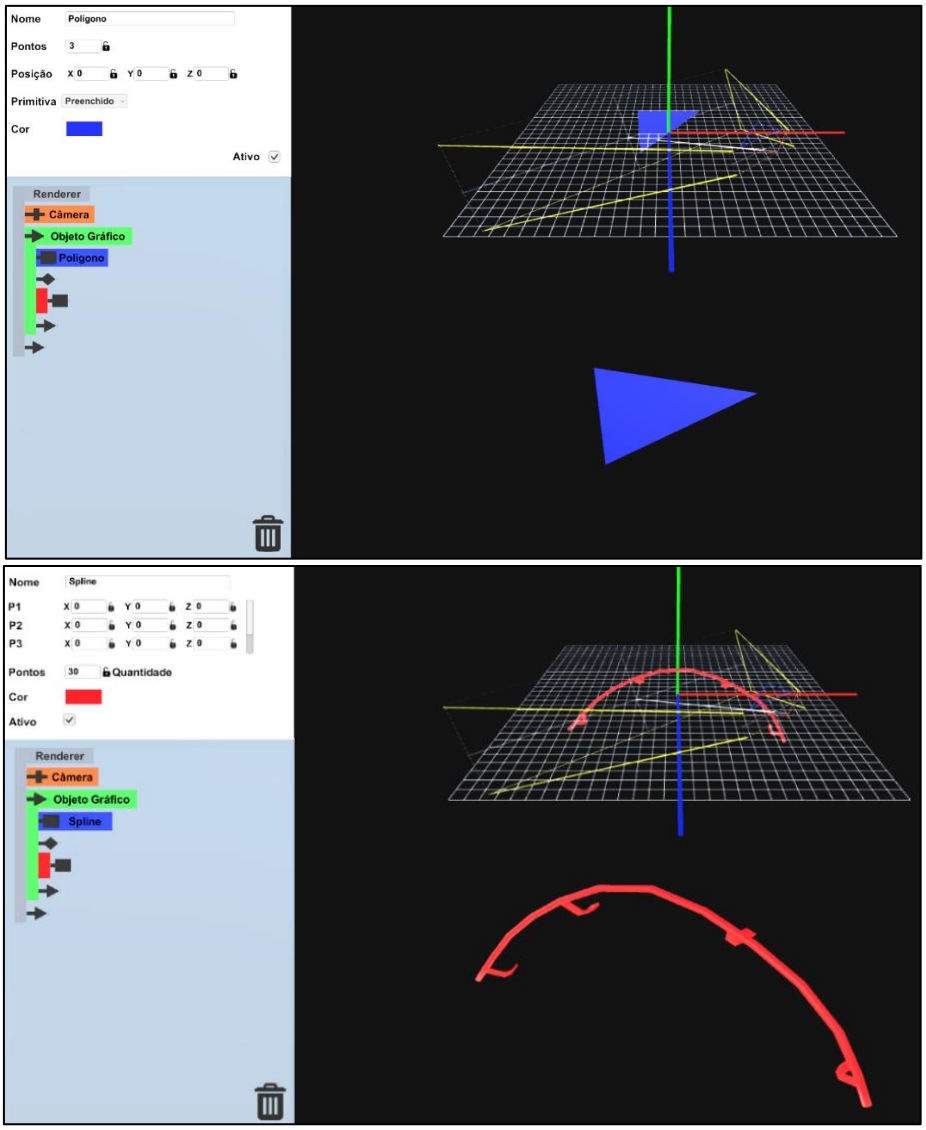
Quadro 8 – Representação do bloqueio de campos nas propriedades da peça e no arquivo

Fonte: elaborado pelo autor.

Na Figura 19 exibe a criação de dois cenários, utilizando as peças `Poligono` e `Spline` respectivamente, desde sua representação no ambiente gráfico, encaixes nos slots até a visualização das peças no painel `Visualizador`. A peça `Poligono` foi utilizado o recurso de `MeshRenderer` para seu desenho no ambiente. Enquanto a `Spline` foi utilizada com um componente gratuito da loja do Unity chamado `SplineMesh` (UNITY, 2021d), tendo recursos essenciais para manipulação de uma `Spline` como: controle dos pontos e sua curvatura.

Figura 19 – Cenários gerados para as peças Polígono e Spline respectivamente



Fonte: elaborado pelo autor.

Na versão anterior, a implementação dos encaixes das peças foi definida utilizando o eixo `y` das peças para validar como estava sendo executado seu encaixe. Neste caso foi realizada a refatoração, utilizando o recurso de gatilhos dos componentes chamados como `Colliders` validando se o objeto colidido na peça é o `slot` correto, evitando assim cálculos desnecessários e tornando mais acessível a realização de manutenções conforme a necessidade. Enquanto os comportamentos estavam encapsulados no `Controller`, foi realizada uma refatoração nos comportamentos das peças separando-os em `scripts` específicos para melhor encapsulamento, compreensão e separação dos componentes.

Destaca-se também a refatoração do deslocamento das peças, utilizando os componentes `VerticalLayoutGroup` e `HorizontalLayoutGroup` do Unity para melhor responsividade dos `slots`, descartando as implementações da versão anterior, na qual todas as peças eram recalculadas às suas posições e dependendo do tamanho das visualizações era preciso redefinir as constantes em que ajustavam as peças aos `slots`.

4.2 TESTE DE DESEMPENHO

Para verificar o desempenho e operacionalidade da ferramenta, foram montados e testados cenários gerando executáveis para web e desktop, tendo como critério de análise o consumo de memória utilizando gerenciador de tarefas do Windows em tempo real. Os testes de web foram feitos no mesmo sistema, utilizando os navegadores Google Chrome, Opera Gx e Microsoft Edge. Os cenários de testes são semelhantes aos usados na versão da ferramenta 5.0 desenvolvida por Buttenberg (2020). Foram criados cinco cenários de testes utilizando as peças já incorporadas da ferramenta, além da nova peça `Iteracao` implementada nesta versão, como representada na Tabela 1.

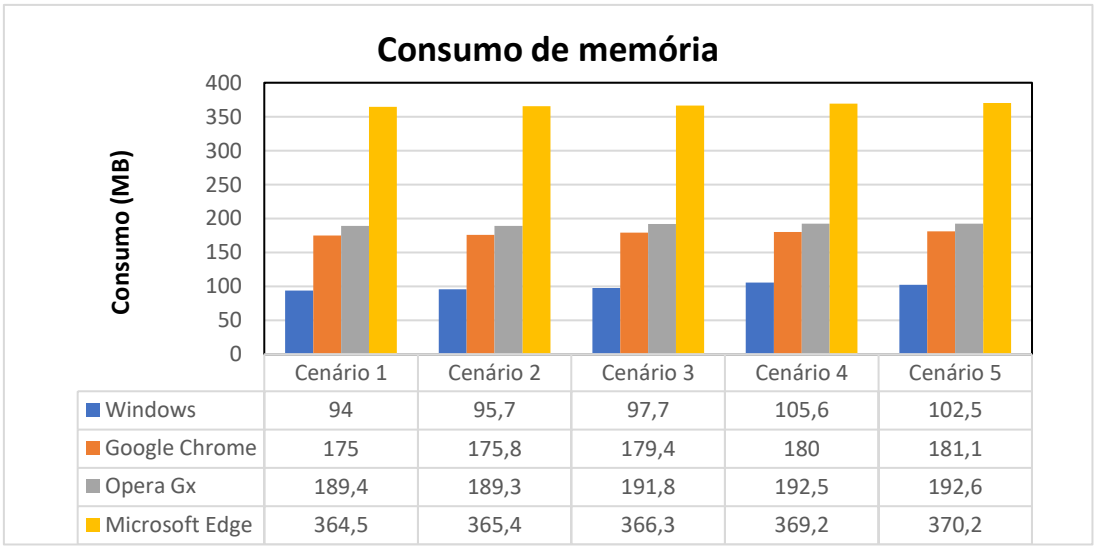
Tabela 1 – Cenários de teste

Peça	Cenário 1	Cenário 2	Cenário 3	Cenário 4	Cenário 5
Câmera	1	1	1	1	1
Objeto Gráfico	1	2	5	6	8
Cubo	1	2	5	6	8
Transladar	1	2	6	6	8
Rotacionar	1	2	6	6	8
Escalar	1	2	6	6	8
Iteração	2	4	4	10	12
Iluminação	1	2	4	6	8
	9	**17**	**37**	**47**	**61**

Fonte: elaborado pelo autor.

Em relação ao consumo de memória, a Figura 20 destaca que o uso da memória foi constante, sem nenhum aumento significativo entre os cenários. A aplicação desktop, assim como a versão anterior da ferramenta, teve o melhor desempenho, enquanto os navegadores consumiram mais memória. Entre os navegadores, o Google Chrome teve o menor consumo, o Opera Gx (sem a utilização do recurso de uso de memória), não teve uma grande diferença do Google Chrome. Por fim, o Microsoft Edge teve o maior consumo.

Figura 20 – Gráfico de consumo da memória



Fonte: elaborado pelo autor.

Os testes de desempenho feitos por Buttenberg (2020), tiveram resultados inferiores em relação a versão atual da ferramenta na web, porém são resultados que não afetam no desempenho da utilização. Comparando os resultados do quarto cenário da versão 5.0 com o terceiro cenário da versão 6.0 (atual) no ambiente Windows, à versão 5.0 teve um desempenho de consumo em 94.8 MB utilizando 37 peças e no Google Chrome 188.3 MB. Enquanto a versão 6.0 utilizando 37 peças, foram consumidos 97.7 MB e 179.4 MB, respectivamente.

5 CONCLUSÕES

Um dos principais objetivos deste trabalho foi adicionar a opção de bloqueio de campos nas propriedades das peças disponíveis na ferramenta. Para isso se utilizou o recurso da linguagem C# na conversão da propriedade para `base64` a fim de embaralhar os dados informados. Onde são exportados/importados em um projeto com formato de arquivo `json`. Este objetivo foi atingido sendo realizado o embaralhamento dos dados conforme definição nas propriedades tanto na exportação como desembaralhamento na importação.

Para exportação e importação dos projetos desenvolvidos na ferramenta, foi utilizado um componente terceiro gratuito chamado `SimpleFileBrowser` (UNITY, 2021b) e atendeu adequadamente para o objetivo. Na estrutura das classes para exportação foi utilizado a biblioteca `JsonUtility` do Unity pela facilidade na criação do arquivo `json` conforme a definição. Porém, algumas classes são carregadas vazias no arquivo pela limitação desnecessária da biblioteca a qual é um ponto a ser melhorado, utilizando terceiros componentes com recursos de orientação a objetos permitindo herança, polimorfismo e encapsulamento das classes.

A peça `Iteracao`, sendo um dos principais objetivos, foi realizado utilizando o conceito de `Coroutines` do Unity, o qual realiza iterações nas transformações das formas encaixadas no `Renderer`. A implementação em `Coroutines`, foi utilizada pela facilidade em realizar animações em tempo de execução, além da fácil manutenção do código. Os resultados foram satisfatórios na performance e o consumo não aumentou de maneira significativa. Por fim, a peça `Iteracao` demonstrou no ambiente gráfico de maneira iterativa, a relação das transformações e como afetam as formas onde estão encaixadas.

Outro objetivo, foi a implementação das peças `Poligono` e `Spline` pendentes da versão 4.0, destacando a implementação da peça `Spline` utilizando um componente terceiro gratuito do Unity chamado `SplineMesh`, tendo todas as propriedades necessárias para ser manipulada, desde os pontos de controle à curvatura definida como quantidade de pontos. Enquanto no `Poligono`, foi utilizado o componente `MeshRenderer` e para melhorar a dispersão dos pontos da malha, foi aplicado o algoritmo `Triangulator`. Ambas as peças atenderam os objetivos, contendo os conceitos básicos da computação gráfica em cada um.

A principal dificuldade ao longo da implementação do trabalho, foi a sua refatoração que se tornou obrigatoriamente necessária, pois não era possível utilizar rotinas especificas em novas peças, propriedades e recursos adicionados nesta versão. Muitos comportamentos estavam centralizados em apenas um script, chamado `Controller`, sendo desfragmentado em comportamentos específicos das peças, suas propriedades, utilizações de constantes globais, remoção de outras variáveis (não mais utilizadas) e rotinas funcionais dispersadas pelo projeto o que foi centralizado, o qual script já existente, chamado `Util_VisEdu`.

Outros aspectos refatorados da ferramenta, foram a disposição das peças nos `slots` do `Renderer`. Toda vez que era adicionada uma nova peça nos encaixes, as demais eram sempre recalculadas em suas posições. Portanto, foram utilizados os componentes `VerticalLayoutGroup` e `HorizontalLayoutGroup`, passando a responsabilidade da disposição das peças nos encaixes. Além destes componentes serem utilizados na tela das propriedades das peças para melhorar a responsividade dos campos. A refatoração do código da ferramenta, teve resultados positivos nas melhorias no entendimento do código e na separação de comportamentos, tornado mais fácil de realizar a manutenção e o desempenho da disposição das peças.

Por fim, os resultados foram satisfatórios, mesmo havendo pouco aumento significativo no consumo da memória em alguns navegadores. Um dos motivos desse pequeno aumento, foi a implementação das peças `Poligono`, `Spline` e `Iteracao`. Os ambientes mais estáveis foram o Google Chrome e Windows, nos testes aplicados. Como sugestão para trabalhos futuros na ferramenta, sugere-se: (i) adicionar novos passos no tutorial da ferramenta; (ii) utilizar outras bibliotecas ou componentes para exportação de um cenário construído com recursos de orientação a objeto; (iii) melhorias na iluminação já pendentes da versão anterior, aplicando nas novas peças adicionadas; (iv) melhorar documentação do painel de ajuda com os novos recursos adicionados na ferramenta; (v) desenvolver as propriedades já criadas nas cenas mas não implementadas `look at`, `near` e `far` da câmera pendentes da versão anterior.

REFERÊNCIAS

BUTTENBERG, Peterson Boni. **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, Fundação Universidade Regional de Blumenau, Blumenau.

FOWLER, Martin. **Refatoração: Aperfeiçoando o design de códigos existentes**. Novatec Editora, 2020.

KOEHLER, William Fernandes. **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, Fundação Universidade Regional de Blumenau, Blumenau.

MELO, Telma de Macedo. O emprego do aplicativo móvel Duolingo no processo de ensino e aprendizagem de alemão como língua estrangeira em contexto de ensino presencial universitário. **Pandaemonium Germanicum**, São Paulo, v. 24, no. 42, p. 78-107, jan./abr. 2021.

MONTEIRO, Edenar Souza; NANTES, Eliza Adriana Sheuer. O letramento digital como estratégia de ensino-aprendizagem no ensino superior, durante o ensino remoto emergencial. **Pesquisa, Sociedade e Desenvolvimento**, [S. l.], v. 10, n. 10, p. e03101018576, 2021. DOI: 10.33448/rsd-v10i10.18576. Disponível em: https://rsdjournal.org/index.php/rsd/article/view/18576. Acesso em: 06 dez. 2021.

RAPELI, Leide Rachel Chieusi. **Refatoração de sistema Java utilizando padrões de projeto: um estudo de caso**. 2006. 130 f. Dissertação de mestrado (Pós-Graduação em Ciência da Computação) – Centro de Ciências Exatas e de Tecnologia, Universidade Federal de São Carlos, São Carlos.

SILVA, Adriana Santos da. **A tecnologia como nova prática pedagógica.** 2011. Monografia apresentada ao curso de pós-graduação em Supervisão escolar, Vila Velha.

VALENTE, Marco Tulio. **Engenharia de Software Moderna:** Princípios e práticas para desenvolvimento de software com produtividade. Independente, 2020.

VIEIRA, Pâmela Carolina. **QuestMeter: Ferramenta de quiz com conceitos de clickers e gamificação**. 2019. 20 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro de Ciências Exatas e Naturais, Fundação Universidade Regional de Blumenau, Blumenau.

UNITY. **Unity Asset Store:** Flexible Color Picker. [S.1.], [2021a]. Disponível em: <https://assetstore.unity.com/packages/tools/gui/flexible-color-picker-150497>. Acesso em: 05 dez. 2021.

UNITY. **Unity Asset Store**: Runtime File Browser. [S.1.], [2021b]. Disponível em: <https://assetstore.unity.com/packages/tools/gui/runtime-file-browser-113006>. Acesso em: 05 dez. 2021.

UNITY. **Unity – Manual:** Coroutines. [S.1.], [2021c]. Disponível em: <https://docs.unity3d.com/2021.2/Documentation/Manual/Coroutines.html>. Acesso em: 01 dez. 2021.

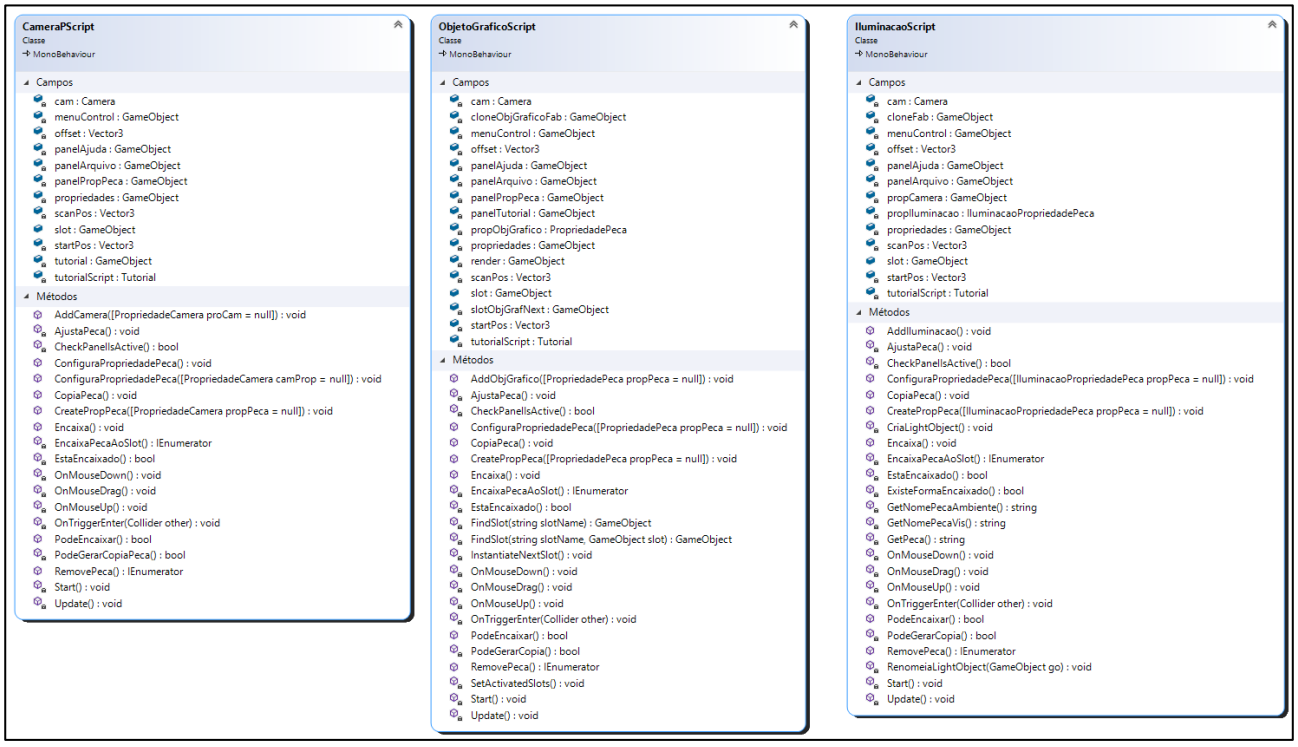
UNITY. **Unity Asset Store**: SplineMesh. Unity, [2021d]. Disponível em: https://assetstore.unity.com/packages/tools/modeling/splinemesh-104989. Acesso em: 05 dez. 2021.

ZANLUCA, Gabriel. **Toweljs: Engine 3D em JavaScript usando arquitetura baseada em componentes**. 2018. 80 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro de Ciências Exatas e Naturais, Fundação Universidade Regional de Blumenau, Blumenau.

APÊNDICE A – DIAGRAMAS DE CLASSES DA FERRAMENTA

Este apêndice ilustra os diagramas classes na qual compõe a ferramenta. A Figura 21 representa o diagrama de classes das peças `Camera`, `Objeto Grafico` e `Iluminacao` disponíveis na ferramenta, onde cada peça possui seu `script` com seus comportamentos específicos.

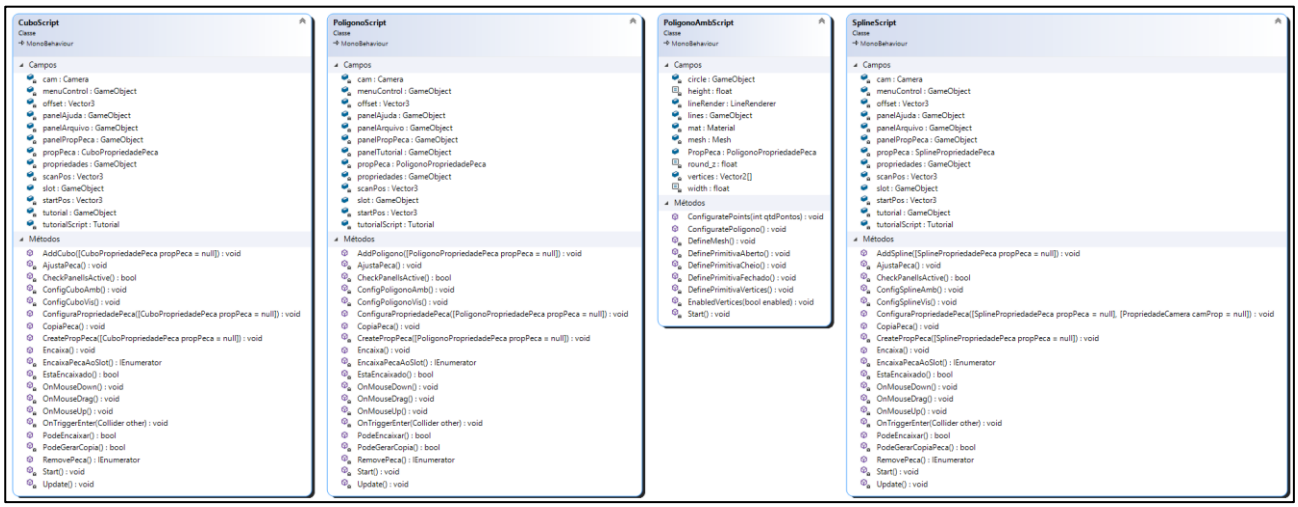
Figura 21 – Diagrama de classes das peças Câmera, Objeto Gráfico e Iluminação



Fonte: elaborado pelo autor.

A Figura 22 demonstra o diagrama de classes das peças `Cubo`, `Poligono` e `Spline`, onde a peça `Poligono` possui uma classe auxiliar para sua criação em tempo de execução.

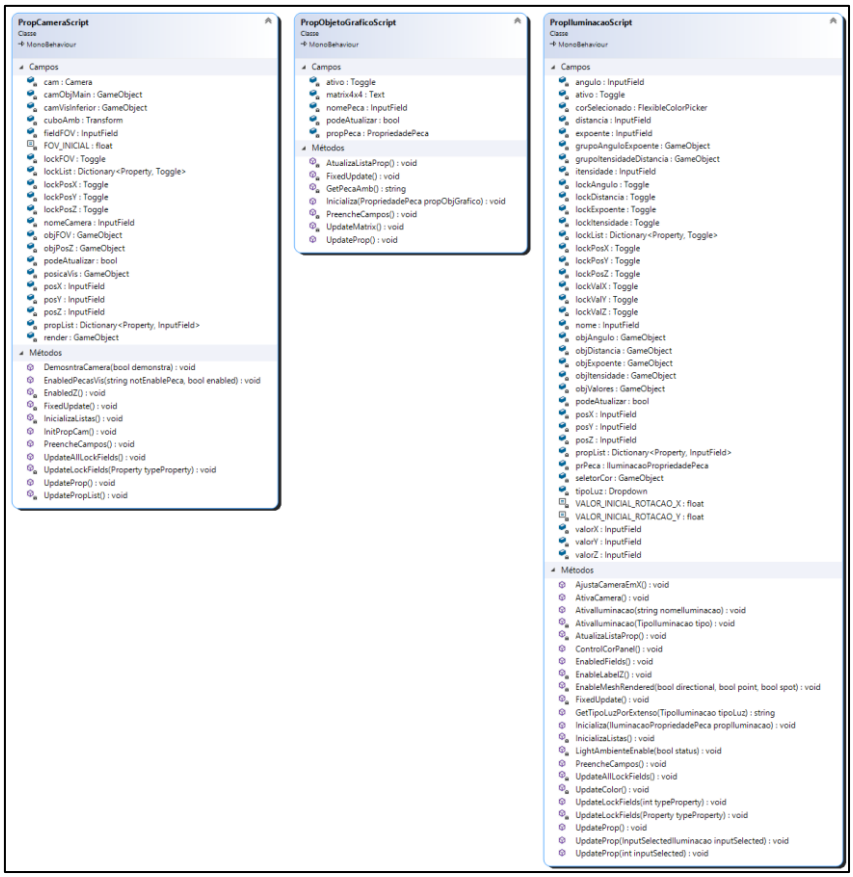
Figura 22 – Diagrama de classes das peças Cubo, Polígono e Spline



Fonte: elaborado pelo autor.

A Figura 23 demonstra o diagrama de classes dos comportamentos das propriedades das peças `Camera`, `Objeto Grafico` e `Iluminacao`.

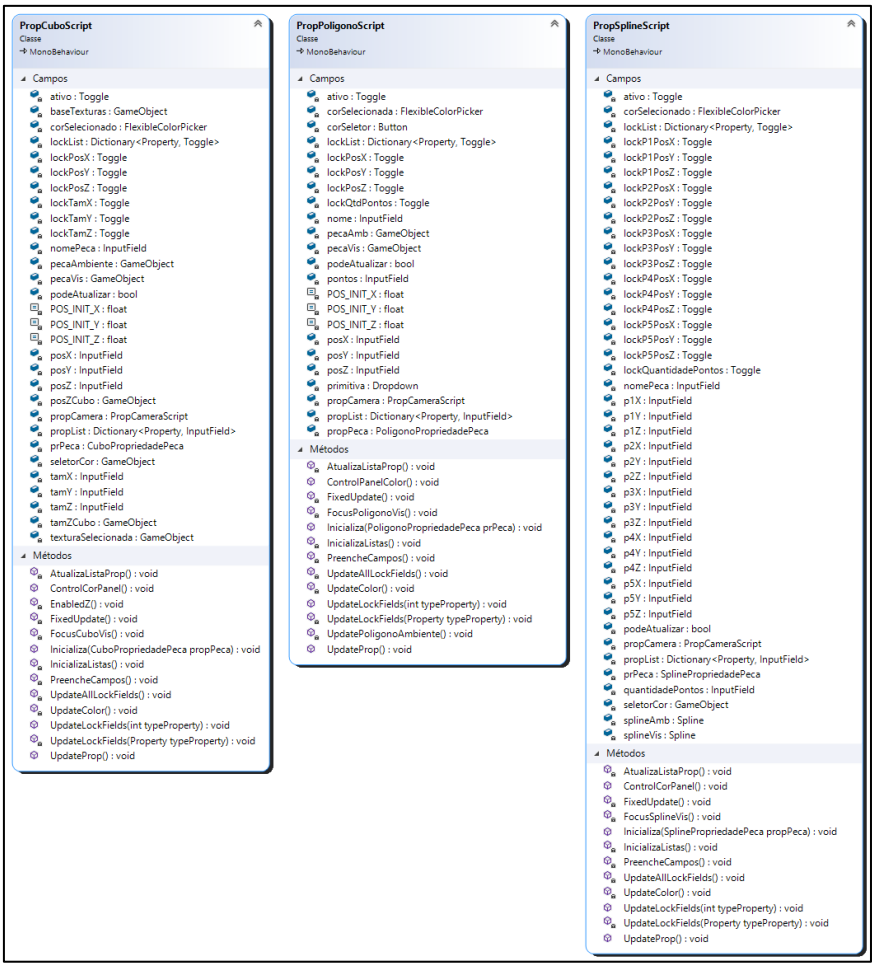
Figura 23 – Diagrama de classes dos comportamentos das propriedades das peças Câmera, Objeto Gráfico e Iluminação



Fonte: elaborado pelo autor.

A Figura 24 destaca o diagrama de classes dos comportamentos das propriedades das peças `Cubo`, `Poligono` e `Spline`.

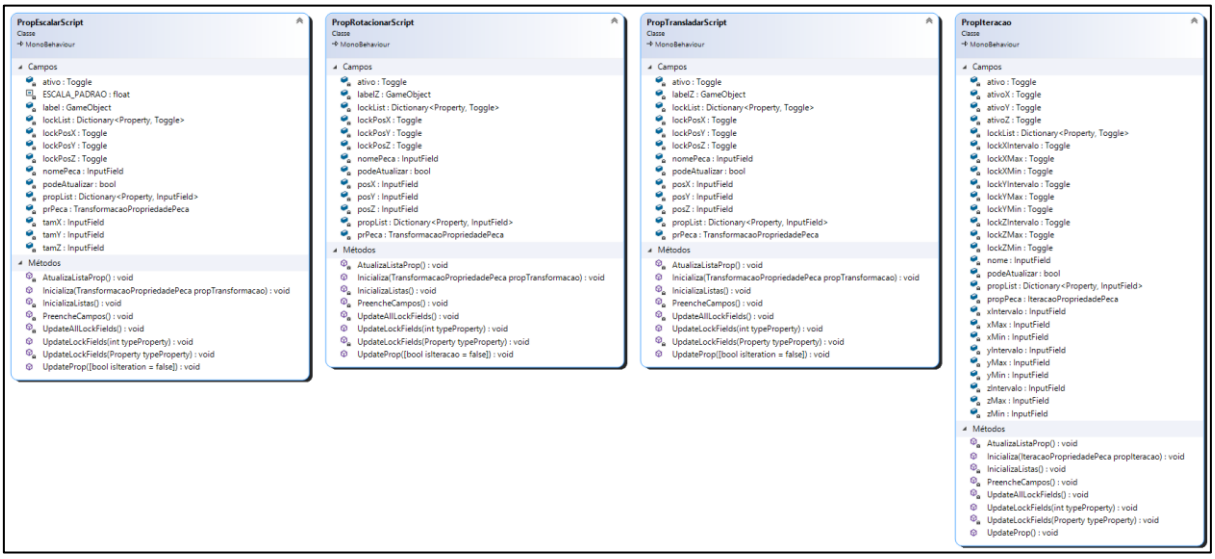
Figura 24 – Diagrama de classes dos comportamentos das propriedades das peças Cubo, Polígono e Spline



Fonte: elaborado pelo autor.

Na Figura 25 ilustra o digrama de classes dos comportamentos das propriedades das peças `Transladar`, `Rotacionar`, `Escalar` e `Iteracao`.

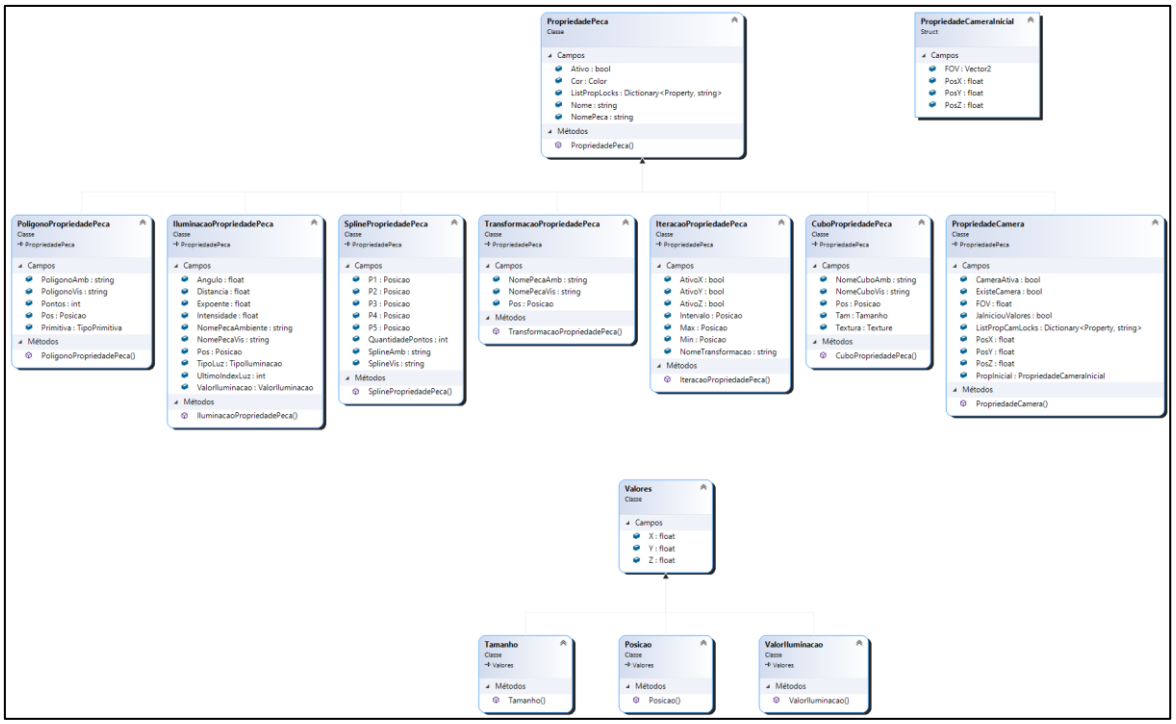
Figura 25 – Diagrama de classes dos comportamentos das propriedades das peças do tipo transformação e Iteração



Fonte: elaborado pelo autor.

A Figura 26 representa o diagrama de classes de todas as propriedades das peças disponíveis a serem manipulados na ferramenta.

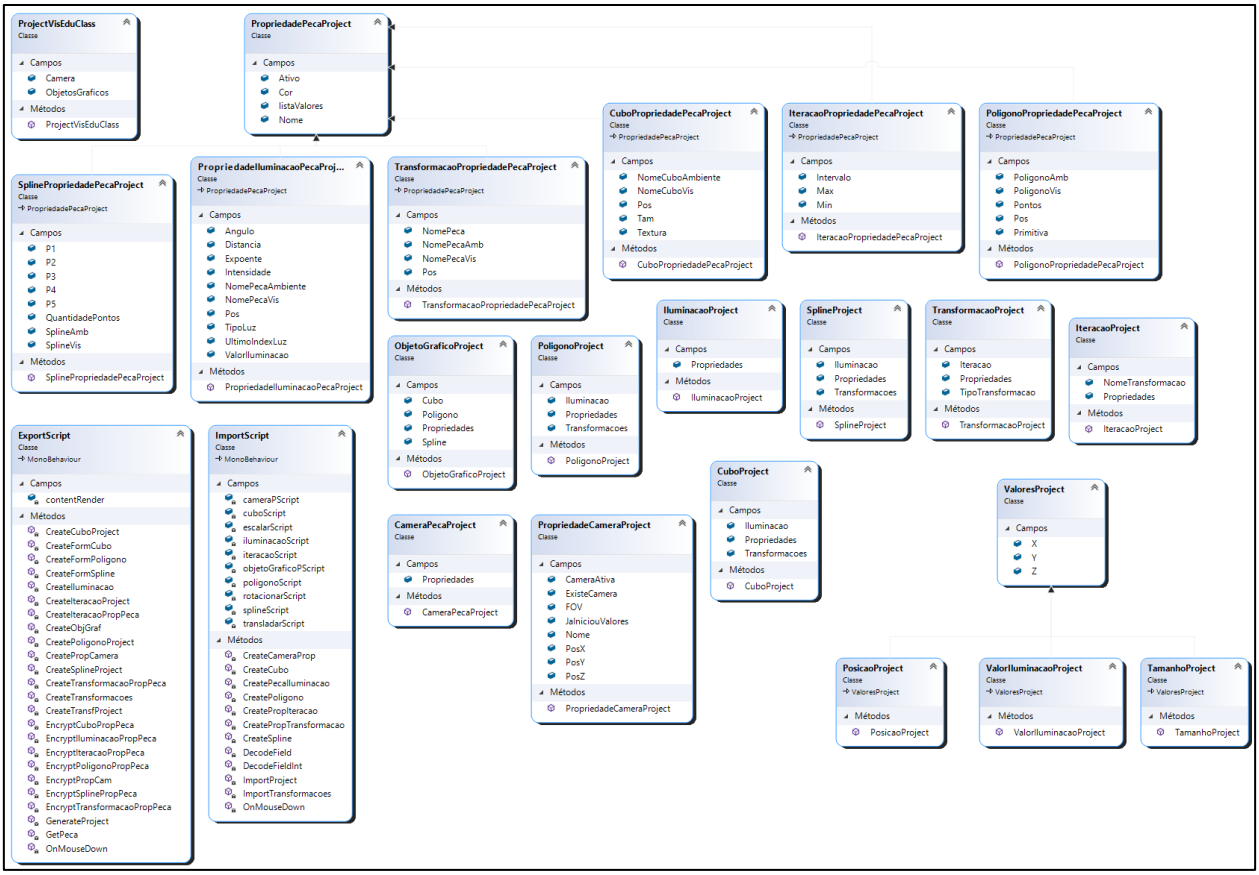
Figura 26 – Diagrama de classes de todas as propriedades das peças disponíveis



Fonte: elaborado pelo autor.

Na Figura 27 destaca-se o digrama de classes da exportação e importação de uma cena construída na ferramenta.

Figura 27 – Diagrama de classes da exportação e importação da ferramenta



Fonte: elaborado pelo autor.