

UNIVERSIDADE REGIONAL DE BLUMENAU
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

**VISEDU-CG: APLICAÇÃO DIDÁTICA PARA VISUALIZAR
MATERIAL EDUCACIONAL, MÓDULO DE COMPUTAÇÃO
GRÁFICA**

JAMES PERKISON MONTIBELER

**BLUMENAU
2014**

2014/1-10

JAMES PERKISON MONTIBELER

**VISEDU-CG: APLICAÇÃO DIDÁTICA PARA VISUALIZAR
MATERIAL EDUCACIONAL, MÓDULO DE COMPUTAÇÃO
GRÁFICA**

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Dalton Solano dos Reis, M. Sc. - Orientador

**BLUMENAU
2014**

2014/1-10

**VISEDU-CG: APLICAÇÃO DIDÁTICA PARA VISUALIZAR
MATERIAL EDUCACIONAL, MÓDULO DE COMPUTAÇÃO
GRÁFICA**

Por

JAMES PERKISON MONTIBELER

Trabalho aprovado para obtenção dos créditos
na disciplina de Trabalho de Conclusão de
Curso II, pela banca examinadora formada
por:

Presidente: Prof. Dalton Solano dos Reis, M. Sc. – Orientador, FURB

Membro: Prof. Alexander Roberto Valdameri, M. Sc. – FURB

Membro: Prof. Aurélio Faustino Hoppe, M. Sc. – FURB

Blumenau, 10 de julho de 2014

Dedico este trabalho a Deus, Jesus e o Espírito Santo, por terem me dado vida e sabedoria; e a meus pais, que sempre me concederam do seu amor sem esperar nada em troca.

AGRADECIMENTOS

A Deus, pelo seu infinito amor e graça, ao ponto de dar a vida de seu filho por mim.

A Jesus, por não ter me negado sua vida e assim criar um caminho para mim até Deus; pela sua interseção perante Deus que a cada instante salva minha alma.

Ao Espírito Santo pelo consolo, pela alegria, pelos ensinamentos, pelos milagres e por, até aqui, ter me sustentado.

Aos meus pais, Daniel e Marleci, por sempre sonharem com esse momento, mesmo quando eu desanimei em sonhá-lo, e terem me incentivado com todo amor, afeto e apoio financeiro que puderam; por cobrarem sempre o melhor de mim.

Aos meus irmãos, Jackeline e Junior, pela paciência, compreensão, incentivo e amizade; por serem verdadeiramente irmãos.

A minha namorada, Andressa, pelo carinho, incentivo incessante e por estar sempre ao meu lado.

Ao meu orientador, Dalton Solano dos Reis, pela grande paciência, pelos perdões e pelo conhecimento; por toda a ajuda, concelhos, incentivos e sua paixão pelo saber.

Aos meus professores por todo o conhecimento que me foi repassado, pela paciência ao me ensinar e por me incentivarem a não desistir.

A minha empresa, Senior, por todo incentivo, pelo apoio financeiro e pelas palavras de ânimo através de seus coordenadores Hammes, Paludo e Suzi.

Aos meus amigos, por me apoiarem com concelhos e conhecimento.

A todos os próximos a mim, por compreenderem e perdoarem minha ausência.

Há tempo de nascer e tempo de morrer; tempo
de plantar e tempo de arrancar o que se
plantou

Salomão

RESUMO

Este trabalho apresenta a implementação de uma aplicação web voltada ao aprendizado de computação gráfica com foco nos conceitos de câmera, grafo de cena e transformações geométricas, denominada VisEdu-CG. Sua construção é baseada em alguns conceitos de informática na educação, tornando o seu uso mais intuitivo e eficiente. A biblioteca Three.js é utilizada como ponto fundamental para montar o editor de exercícios da aplicação e visualizar o resultado desses exercícios no espaço 3D. A aplicação disponibiliza um jogo de encaixe de formas geométricas, onde é possível realizar exercícios através de peças que se encaixam coerentemente. Cada peça trabalha com um conceito específico de computação gráfica: câmera, transformações geométricas, grafo de cena e objeto gráfico (que na aplicação é representado por um cubo). Essas peças são apresentadas em um editor, onde é possível criar peças e encaixá-lasumas às outras. Conforme as peças são encaixadas, é possível visualizar o resultado gráfico obtido pelos encaixes em um espaço 3D. Ao selecionar uma peça, é possível visualizar o código em Java (utilizando a biblioteca JOGL) que é necessário para reproduzir o conceito representado pela peça, auxiliando o aluno a compreender a ordem lógica dos comandos utilizados para o desenvolvimento da cena. Na análise de desempenho o Firefox foi avaliado como o melhor navegador para uso da aplicação, por suportar a maior quantidade de peças em um mesmo exercício. A análise de usabilidade foi feita a partir de um questionário respondido por alunos da disciplina de computação gráfica, que demonstrou um desempenho muito bom da aplicação e trouxe várias contribuições de melhorias.

Palavras-chave: Informática na educação. Software educacional. WebGL. HTML5. Three.js.

ABSTRACT

This paper presents the implementation of a web application focused on the learning of computer graphics concepts with focus on camera, scene graph and geometric transformations, called CG-VisEdu. Its construction is based on some computer concepts in education, making its use more intuitive and efficient. The Three.js library is used as a key point to set up the exercises editor of the application and view their results in 3D space. The application provides a geometric shapes fitting game, where you can perform exercises through pieces that fit together coherently. Each piece works with a specific concept of computer graphics: camera, geometric transformations, scene graph and graphic objects (which is represented by a cube in the application). These pieces are displayed in an editor where it's possible create pieces and fit them together. As the pieces are fitted, it's possible view the graphic results obtained by the fittings in a 3D space. When a piece is selected, it's possible view the code in Java (using the JOGL library) that is needed to reproduce the concept represented by the piece, helping the student to understand the logical order of the commands used for the scene development. Firefox has been reported as the best browser to use the application on performance analysis, by supporting the largest number of pieces in one exercise. The usability analysis was taken from a questionnaire completed by students of computer graphics, which showed a very good performance of the application e and brought several contributions improvement.

Key-words: Computers in education. Educational software. WebGL. HTML5. Three.js.

LISTA DE ILUSTRAÇÕES

Figura 1 – Interface do AduboGL	22
Figura 2 – Principais painéis do StarLogo TNG	23
Figura 3 - Diagrama de casos de uso	27
Quadro 1 - Detalhamento do caso de uso UC01	28
Quadro 2 - Detalhamento do caso de uso UC02	29
Quadro 3 - Detalhamento do caso de uso UC03	30
Quadro 4 - Detalhamento do caso de uso UC04	30
Quadro 5 - Detalhamento do caso de uso UC05	30
Quadro 6 - Detalhamento do caso de uso UC06	31
Quadro 7 - Detalhamento do caso de uso UC07	32
Quadro 8 - Detalhamento do caso de uso UC08	32
Quadro 9 - Detalhamento do caso de uso UC09	32
Quadro 10 - Detalhamento do caso de uso UC10	33
Quadro 11 - Detalhamento do caso de uso UC11	33
Quadro 12 – Detalhamento do caso de uso UC12.....	33
Figura 4 - Diagrama de classes.....	35
Figura 5 – Diagrama com exemplos de classes do Three.js	36
Figura 6 – Pacote js.cg.core	38
Figura 7 – Pacote js.cg.extras	39
Figura 8 - Pacote js.cg.exporters	40
Figura 9 - Pacote js.cg.loaders.....	40
Figura 10 – Pacote js.cg.objects.core	41
Figura 11 - Pacote js.cg.objects.items.core	42
Figura 12 - Pacote js.cg.objects.items	44
Figura 13 – Pacote js.cg.objects	46
Figura 14 – Pacote js.cg.panels	48
Figura 15 – Pacote js.cg.....	54
Figura 16 – Diagrama de estado	55
Quadro 13 – Função de inicialização do ambiente gráfico da classe Editor	56
Quadro 14 – Trechos de código que renderizam as cenas da aplicação.....	58

Quadro 15 – Trechos de código que ajustam o tamanho do conteúdo dos painéis Espaço Gráfico e Visão da Câmera.....	59
Figura 17 – Relação entre alguns elementos HTML e classes da biblioteca UI	60
Quadro 16 – Trecho de código que utiliza o framework JSColor	61
Figura 18 – Exemplo de elemento imput associado ao framework JSColor.....	61
Figura 19– Setas que indicam se o nó está aberto ou fechado	62
Quadro 17 – Trecho de código da classe PainelListaItens	62
Quadro 18 – Métodos de construção das classes UI.TreeView e UI.ItemTreeView	62
Quadro 19 – Uso das bibliotecas Detector, THREE.CameraHelper e THREE.TrackballControls no VisEdu-CG	63
Quadro 20 – Trecho de código da classe AObjetoGrafico	64
Quadro 21 – Exemplo de texto gerado com os dados de um exercício	64
Quadro 22 – Função fabricarNovoItem() da classe FabricaDeItens	65
Quadro 23 – Exemplos de implementação do padrão de projeto Observer	65
Figura 20 – Tela de entrada do VisEdu-CG	66
Figura 21 – Painel Fábrica de Peças e subpainéis	67
Figura 22 – Demonstração do processo de criação de uma peça	69
Figura 23 – Demonstração do processo de exclusão de uma peça	69
Figura 24 - Demonstração do reencaixe de peças e encaixe de peças flutuando	70
Figura 25 - Demonstração do processo de seleção de uma peça.....	71
Figura 26 - Demonstração de interação com o painel Visão da Câmera.....	71
Figura 27 – Comparação da cena gerada no VisEdu-CG com a cena gerada pelo código JOGL	72
Quadro 24 – Análise de desempenho no navegador Chrome.....	74
Quadro 25 – Análise de desempenho no navegador Firefox.....	74
Quadro 26 – Análise de desempenho no navegador Internet Explorer	75
Quadro 27 – Questionário de usabilidade do VisEdu-CG.....	76
Figura 28 – Gráfico com as respostas da 1 ^a questão.....	77
Figura 29 – Gráfico com as respostas da 2 ^a questão.....	77
Figura 30 – Gráfico com as respostas da 3 ^a questão.....	77
Figura 31 – Gráfico com as respostas da 4 ^a questão.....	78
Figura 32 – Gráfico com as respostas da 5 ^a questão.....	78
Figura 33 – Gráfico com as respostas da 6 ^a questão.....	79
Figura 34 – Gráfico com as respostas da 7 ^a questão.....	79

Figura 35 – Gráfico com as respostas da 8 ^a questão.....	79
Figura 36 – Gráfico com as respostas da 9 ^a questão.....	80
Quadro 28 – Comparação com os trabalhos correlatos	82
Figura 38- Peça Câmera	90
Figura 39- Peça Objeto Gráfico.....	90
Figura 40- Peça Cubo	90
Figura 41- Peça Transladar.....	90
Figura 42- Peça Rotacionar	90
Figura 43- Peça Escalar	90
Figura 44- Peça Renderizador	90
Figura 45 – Exemplo de exercício construído no VisEdu-CG	91
Figura 46 – Exemplo de alteração das propriedades da peça Câmera.....	92
Figura 47 – Exemplo de alteração das propriedades da peça Objeto Gráfico	93
Figura 48 – Exemplo de alteração das propriedades da peça Transladar	93
Figura 49 – Exemplo de alteração das propriedades da peça Rotacionar	94
Figura 50 – Exemplo de alteração das propriedades da peça Escalar	94
Figura 51 – Exemplo de alteração das propriedades da peça Cubo	95
Quadro 29 – Comandos JOGL gerados para a peça Objeto Gráfico	96
Quadro 30 – Comandos JOGL gerados para a peça Transladar.....	96
Quadro 31 – Comandos JOGL gerados para a peça Rotacionar	96
Quadro 32 – Comandos JOGL gerados p ara a peça Escalar	97
Quadro 33 – Comandos JOGL gerados para a peça Câmera	97
Quadro 34 – Comandos JOGL gerados para a peça Cubo	98
Figura 52 – Primeira página do exercício criado para o VisEdu-CG	99
Figura 53 – Segunda página do exercício criado para o VisEdu-CG	100
Figura 54 – Terceira página do exercício criado para o VisEdu-CG	101
Quadro 35 – Respostas da 10 ^a questão	102
Quadro 36 – Respostas da 11 ^a questão	102
Quadro 37 – Respostas da 12 ^a questão	103

LISTA DE SIGLAS

API – *Aplication Programming Interface*

CSS – *Cascading Style Sheets*

DOM – *Document Object Model*

FOV – *Field Of View*

FPS – *Frames Por Segundo*

FURB – Universidade Regional de Blumenau

GLSL ES – *OpenGL ES Shading Language*

GT – Geometria da Tartaruga

HTML – *Hypertext Markup Language*

HTML5 – *Hypertext Markup Language* versão 5

JSON – *JavaScript Object Notation*

JOGL – Java OpenGL

MIT – *Massachusetts Institute of Technology*

OpenGL – *Open Graphics Library*

OpenGL ES – *OpenGL Embedded Systems*

STEP – *MIT's Scheller Teacher Education Program*

SVG – *Canvas e Scalable Vector Graphics*

UML – *Unified Modeling Language*

URL – *Uniform Resource Locator*

VBOs – *Vertex Buffer Objects*

W3C – *World Wide Web Consortium*

WebGL – *Web Graphics Library*

WHATWG – *Web Hypertext Application Technology Working Group*

SUMÁRIO

1 INTRODUÇÃO.....	14
1.1 OBJETIVOS DO TRABALHO	15
1.2 ESTRUTURA DO TRABALHO	15
2 FUNDAMENTAÇÃO TEÓRICA.....	16
2.1 INFORMÁTICA NA EDUCAÇÃO	16
2.1.1 Internet na educação.....	17
2.1.2 Softwares na educação	18
2.1.2.1 Ambientes interativos de aprendizagem	18
2.1.3 Jogos educacionais.....	20
2.2 WEBGL.....	20
2.3 ADUBOGL.....	21
2.4 TRABALHOS CORRELATOS	22
3 DESENVOLVIMENTO.....	25
3.1 PRINCIPAIS REQUISITOS	25
3.2 ESPECIFICAÇÃO	26
3.2.1 Diagrama de casos de uso	27
3.2.2 Diagrama de classes	33
3.2.3 Diagrama de estado	54
3.3 IMPLEMENTAÇÃO	55
3.3.1 Técnicas e ferramentas utilizadas.....	55
3.3.2 Operacionalidade da implementação	66
3.4 RESULTADOS E DISCUSSÃO	72
3.4.1 Análise de desempenho da aplicação.....	73
3.4.2 Análise de usabilidade da aplicação.....	75
3.4.3 Relação do presente trabalho com os trabalhos correlatos.....	81
4 CONCLUSÃO	83
4.1 EXTENSÕES	84
REFERÊNCIAS.....	87
APÊNDICE A – Peças disponíveis no VisEdu-CG	89
APÊNDICE B – Propriedades das peças disponíveis no VisEdu-CG e o impacto destas no conteúdo dos painéis Espaço Gráfico e Visão da Câmera	91

APÊNDICE C – Comandos JOGL gerados pelas peças do VisEdu-CG	96
APÊNDICE D – Exercício no VisEdu-CG aplicado em sala de aula.....	99
APÊNDICE E – Respostas descritivas do questionário de usabilidade do VisEdu-CG	102
ANEXO A – Requisitos do AduboGL.....	105

1 INTRODUÇÃO

A educação é um dos alicerces para o desenvolvimento e sustentação de um país. A evolução dos conceitos, técnicas e ferramentas utilizadas no processo de educação tem sido foco de intensa pesquisa nas últimas décadas, na tentativa de se obter um melhor resultado na transmissão de conhecimento (GONÇALVES; CANESIN, 2002).

No Brasil, de forma geral, pode-se observar que o processo de ensino tem ignorado um fator muito importante na aprendizagem: a motivação (BATTAIOLA et al., 2002). Ao avaliar-se o padrão de ensino em uma aula, é possível observar a existência do mesmo perfil de ensino que já era praticado na década de 80: professores cobrando a exatidão do que está escrito nos livros, sem incentivar o aluno a crítica e a experimentação do próprio conhecimento (COSTA et al., 2009).

Felizmente, visto que jogos eletrônicos são um dos principais atrativos na indústria de entretenimento e resultam numa movimentação de bilhões de dólares por ano, tem crescido a popularidade do uso de jogos educacionais. Neles busca-se encontrar um caminho ou incremento para este fator de motivação na aprendizagem (BATTAIOLA et al., 2002).

Cada vez mais a Internet tem ganhado importância na difusão de conhecimento, tanto no meio educacional quanto nas outras áreas de comunicação humana. A tecnologia tornou-se um fator de distância entre as pessoas ainda mais relevante que a distância física, e isso também reflete-se na educação. Universidades e escolas correm atrás da Internet para não ficarem para trás e alcançar visibilidade. Consequentemente, o ensino presencial também se modifica, onde os limites das paredes são quebrados e se alcança possibilidades quase ilimitadas de troca de informação, com a integração de vários tipos de mídia, tudo isso tanto em tempo real, como assincronicamente (MORAN, 1997).

Levando em conta a importância da Internet na transmissão e troca de informações (MORAN, 1997), a inexistência de um software educacional aplicável à disciplina de computação gráfica da Universidade Regional de Blumenau (FURB) e a necessidade deste software ser facilmente difundido no meio acadêmico, construiu-se um jogo educacional utilizando a biblioteca multiplataforma *Web Graphics Library* (WebGL). Esse jogo resume-se em um *framework* para estudo de computação gráfica que está disponível na Internet e irá possuir algumas funções prontas da WebGL, que poderão ser manipuladas livre e coerentemente pelo aluno. Esta interação do aluno com o conteúdo da disciplina através de um jogo visa trazer não só um incremento motivacional, como também uma possível melhoria no processo de ensino.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é estender a pesquisa intitulada "AduboGL – Aplicação Didática Usando a Biblioteca OpenGL", desenvolvida por Araújo (2012), onde foi realizada a implementação de uma aplicação voltada ao aprendizado de computação gráfica com foco nas transformações geométricas.

Os objetivos específicos do trabalho são:

- a) converter as funcionalidades já existentes do AduboGL para a biblioteca multiplataforma WebGL;
- b) representar visualmente a cena criada através de um grafo de cena;
- c) explorar outras funcionalidades gráficas, tais como câmeras e texturas.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está dividido em quatro capítulos. O segundo apresenta a fundamentação teórica necessária para o entendimento deste. Inicialmente é discutida uma visão geral do impacto da informática na educação. Após são apresentados alguns conceitos de softwares e jogos educacionais. Segue-se então com a definição das principais tecnologias abordadas. O capítulo é concluído com uma apresentação dos trabalhos correlatos.

O terceiro capítulo apresenta o desenvolvimento do aplicativo. Primeiramente são apresentados os principais requisitos. Segue-se após com a especificação, que é composta pelos diagramas de casos de uso, de classes e de estado. Na especificação também é discutido como o trabalho foi implementado a partir das técnicas e ferramentas utilizadas, onde se pode visualizar alguns trechos de código. O capítulo é finalizado com a exibição dos resultados e das discussões ocorridas durante o processo de desenvolvimento da aplicação.

Por fim, o quarto capítulo descreve as conclusões deste trabalho e apresenta sugestões para futuras extensões.

2 FUNDAMENTAÇÃO TEÓRICA

Na seção 2.1 é apresentada uma visão da informática na educação, onde também será discutida a influência da Internet sobre a educação e uma breve análise sobre softwares e jogos educacionais. Na seção 2.2 é dada uma definição das principais características da biblioteca multiplataforma WebGL. Na seção 2.3 é dada uma descrição do AduboGL, trabalho de pesquisa continuado por este. Por fim, na seção 2.4 são relacionados os trabalhos correlatos ao trabalho proposto.

2.1 INFORMÁTICA NA EDUCAÇÃO

Na adoção dos computadores para as salas de aula, encontra-se um caminho para transformar as aulas teóricas e o modelo passivo de ouvir e de reproduzir o que se ouviu, em um ambiente dinâmico de aprendizado. Nele aprendizes encontram liberdade para utilizar sua capacidade de descobrimento, realizar uma fácil troca de informações e dar lugar ao compreender em vez de decorar (COX, 2003, p. 70).

A informática no processo de educação cria um ambiente onde os professores e alunos podem caminhar lado a lado e juntos buscar, errar e aprender. Neste ambiente, o aluno é capaz de formar conhecimento através das próprias observações. Este conhecimento, obtido pelo próprio esforço e observação, tem muito mais relevância e importância para o aprendiz e se adapta melhor nas estruturas mentais do mesmo, visto que o processo de aprendizagem exige a existência de uma estrutura anterior, que faz com que as novas informações sejam melhor assimiladas (SILVA, 2000, p. 10).

Atualmente utiliza-se amplamente o computador para informatizar os processos pedagógicos existentes, o que facilita a implantação dos computadores nas escolas, pois não interfere na dinâmica já existente no processo de educação. Porém, utilizar o computador para construir um ambiente de aprendizagem focado na construção de conhecimento traz um desafio, visto que não é só necessário rever os conceitos já conhecidos, tornando o computador em uma nova forma de representar conhecimento, mas como também rever o papel do professor neste novo contexto de aprendizagem (VALENTE, 2002, p. 3).

Segundo Cox (2003, p. 73), vários requisitos são necessários para adotar o uso dos recursos de informática no ambiente acadêmico. Estes requisitos são fundamentais para o sucesso na renovação do processo de ensino. A seguir serão destacados alguns destes requisitos, os quais são:

- a) sensibilizar os agentes escolares: um fator humano comum diante de qualquer mudança é resistência a esta mudança. É necessário fazer com que professores,

- alunos e administradores compreendam a importância da reestruturação do processo de ensino e exerçam seu papel nesta mudança;
- b) preparar o professor: o professor é o ponto chave da reestruturação no processo de aprendizagem. É essencial que ele esteja apto a iniciar seus alunos nos ambientes informatizados e esteja preparado para lidar com as dúvidas e resistência dos alunos;
 - c) equipar a escola: disponibilizar os equipamentos e softwares necessários para execução do ambiente informatizado de educação.

2.1.1 Internet na educação

Quanto a Internet, Lucena e Fuks (2000, p. 13) afirmam que “[...] no ensino superior, sua utilização já é massiva como meio de comunicação (mensagens e circulação de documentos e textos), como instrumento de informação e pesquisa (*web*) e, num grau menor, como elemento de debate nos grupos de discussão [...].” A Internet possibilita a troca de informações em tempo real entre todas as regiões do mundo. Isso abre um grande leque, no que diz respeito à pesquisa e leitura inquiridora, pois de qualquer ambiente, o aprendiz tem acesso aos mais diversos documentos e autores, como se estivesse pessoalmente nas melhores bibliotecas do mundo, além de poder compartilhar e discutir conhecimentos com colegas (SILVA, 2000, p. 34).

Esta disponibilidade de troca de informações exercita o julgamento da veracidade e confiabilidade do que está sendo pesquisado; os aprendizes são estimulados a uma maior expressividade individual, desenvolvem novas habilidades de busca e troca de conhecimento, criam novas formas de selecionar e de compartilhar, para que se possa diferenciar mais facilmente o que “[..] é confiável, o que é de boa qualidade, o autêntico do lixo, o brilho do ouro verdadeiro daquele da imitação [...]” (SILVA, 2000, p. 34).

Com a difusão da Internet na educação, fica cada vez mais evidente a necessidade de modificar os modelos pedagógicos. Pode-se dizer que um novo modelo está em fase de gestação, que visa o “[..] desenvolvimento das competências e das habilidades, o respeito ao ritmo individual, a formação de comunidades de aprendizagem e as redes de convivência, entre outras [...]” (BEHAR, 2009, p. 16).

Em um ambiente *web*, “[..] o professor deve mudar o seu papel atual de provedor de conteúdo para o de facilitador – de solista para maestro [...]” (LUCENA; FUKS, 2000, p. 59). Observa-se então a necessidade do professor estar preparado para mediar o contato entre o aluno, a Internet e o conhecimento, fazendo-se necessário que o mesmo conheça a Internet,

suficientemente bem para aplicá-la e explorá-la na metodologia usada na sala de aula (LUCENA; FUKS, 2000, p. 59).

2.1.2 Softwares na educação

Analizando os softwares educacionais é possível observar que o aprendizado não fica limitado ao software, mas se expande pela interação que existe entre o software e o aluno. Este ambiente é ideal para a aprendizagem, pois “[...] como foi mostrado por Piaget, o nível de compreensão está relacionado com o nível de interação que o aprendiz tem com o objeto e não com o objeto em si [...]” (VALENTE, 2002, p. 90).

Valente (2002, p. 49) propõe diversas abordagens que podem ser utilizadas para a criação de sistemas de aprendizagem através de softwares. Uma delas é o aprendizado socialmente distribuído, onde identificam-se todas as iniciativas que buscam formar as denominadas redes de aprendizagem, que, de certa maneira, podem ser vistas como uma forma de disponibilizar ensino a distância utilizando a Internet. O objetivo dessas redes é criar um ambiente colaborativo de ensino, onde o aprendizado é alcançado através da interação e cooperação *on line*, tendo-se o monitoramento de um ou mais instrutores. A seguir, é apresentada outra dessas abordagens, os ambientes interativos de aprendizagem.

2.1.2.1 Ambientes interativos de aprendizagem

Segundo Valente (2002, p. 58), neste tipo de ambiente o conhecimento é construído através de “[...] atividades de exploração, investigação e descoberta [...]. Estes sistemas “[...] não ensinam nem instruem, apenas têm um determinado comportamento [...]”, eles levam o aprendiz, assim como um cientista, a explorar e adquirir princípios através da análise do comportamento destes sistemas (VALENTE, 2002, p. 58). Valente (2002, p. 58) descreve que estes ambientes, de maneira geral, são fundamentados nos seguintes princípios:

- a) construção e não instrução: o aprendizado é mais efetivo a partir da construção do próprio conhecimento, não por meio de leitura ou sequência de exercícios;
- b) controle do estudante e não controle do sistema: o aprendiz tem controle mais abrangente sobre a interação entre aluno e sistema;
- c) individualização é determinada pelo estudante e não pelo sistema: o *feedback* é formado a partir da interação do aprendiz com o sistema, e cada interação é única e individualizada;
- d) *feedback* rico, gerado a partir da interação do estudante com o ambiente de aprendizagem e não pelo sistema: o *feedback* é o resultado das escolhas e ações do

aprendiz durante o uso do sistema, e não um discurso criado pelo sistema tutor.

Valente (2002, p. 59) define que os ambientes interativos de aprendizagem englobam diversos tipos de sistemas, como por exemplo: sistemas de modelagem e simulação, micromundos, uso de linguagem de programação e sistemas de autoria. Muitos destes tipos de sistemas compartilham características em comum, o que torna nebulosa a distinção entre os mesmos. A seguir será apresentada uma breve descrição de três deles (VALENTE, 2002, p. 59), os quais são:

- a) sistemas de modelagem e simulação: caracterizam-se por sistemas que permitem modelar um fenômeno real ou fictício de forma que se possa observar e analisar o seu comportamento num determinado período de tempo. O sistema de modelagem permite ao usuário montar um cenário de simulação de um fenômeno e explorar as consequências deste cenário, podendo reavaliar, não só a construção do cenário, mas o próprio conhecimento a respeito do fenômeno. A diferença entre sistemas de modelagem e sistemas de simulação é: em sistemas de simulação a modelagem do cenário já está pronta; nos sistemas de modelagem é possível montar o cenário antes de dar início à simulação;
- b) ambientes de programação: permitem, não apenas criar um ambiente para criação de conceitos e ideias para resolução de problemas, mas como transmitem de forma clara o processo que o aluno utilizou para resolver os problemas. Isso torna visível o processo de aprendizado e facilita a reflexão e avaliação deste processo. Neste ponto de vista a programação torna-se uma janela para a mente, que quando associada a uma linguagem limpa (sintaxe e conceitos simples) colaboram grandemente para construção de um ambiente rico de aprendizado, onde facilmente obtém-se um *feedback* do real aprendizado do aluno;
- c) micromundos: são ambientes que possibilitam ao usuário criar um “[...] subconjunto da realidade ou uma realidade construída, cuja estrutura casa com a estrutura cognitiva [...]” (VALENTE, 2002, p. 65), de forma que se crie um ambiente onde se pode experimentar o conhecimento de forma efetiva. Nestes micromundos é possível interagir e controlar algum objeto do domínio ou conhecimento, onde os atributos deste objeto estão associados a um conceito fundamental e a um evento ou função de programação. Um exemplo clássico disso seria a linguagem de programação Logo, onde encontra-se o micromundo da Geometria da Tartaruga (GT), do processamento de listas, da animação, etc. Por

exemplo, no micromundo da GT tem-se como objeto a tartaruga, onde o aprendiz pode modificar seus atributos através de comandos e, observando como ela desenha os resultados, pode trabalhar os conceitos fundamentais (neste caso associado a desenhos gráficos): linha, forma, simetria, cor, etc. Usando o conceito programar de forma ampla, também pode-se definir como micromundos ambientes de modelagem e simulação, assim como diversas aplicações programáveis.

2.1.3 Jogos educacionais

Valente (2002, p. 104) define jogos como um dos diferentes tipos de softwares educacionais e diz que eles podem englobar características de mais de uma abordagem de criação de sistemas de aprendizagem, dependendo da maneira que o aprendiz interage com o software. Prensky (2007, p. 3) defende que existem três razões chaves para o uso de jogos educacionais:

- a) a aprendizagem baseada em jogos digitais não só atende às necessidades atuais de aprendizado, mas também das futuras gerações de alunos;
- b) a aprendizagem baseada em jogos digitais é motivadora, pois é divertida;
- c) a aprendizagem baseada em jogos digitais é extremamente versátil, adaptável a quase todos os assuntos, informações ou habilidades a serem desenvolvidas e, quando usada corretamente, é extremamente eficaz.

2.2 WEBGL

O WebGL é uma biblioteca multiplataforma *web*, baseada no *Open Graphics Library* (OpenGL) *Embedded Systems* (OpenGL ES) 2.0, que visa disponibilizar uma *Application Programming Interface* (API) para desenvolvimento de gráficos 3D de baixo nível. O WebGL está disponível no *Hypertext Markup Language* versão 5 (HTML5). Fabricantes dos navegadores mais utilizados pelo mercado fazem parte do grupo de trabalho do WebGL: Apple (Safari), Google (Chrome), Mozilla (Firefox) e Opera (Opera). Sua especificação é mantida pelo grupo Khronos, o mesmo grupo que mantém a especificação do OpenGL ES e a última versão disponibilizada atualmente é a 1.0.2 (KHRONOS, 2013).

O HTML5 será o próximo padrão para *Hypertext Markup Language* (HTML). Ele é um trabalho que ainda está em desenvolvimento através de uma cooperação entre a *World Wide Web Consortium* (W3C) e a *Web Hypertext Application Technology Working Group* (WHATWG). O HTML5 busca trazer melhorias, como reduzir a necessidade de plugins

externos (como o Flash) e novos recursos baseados no HTML, *Cascading Style Sheets* (CSS), *Document Object Model* (DOM) e JavaScript (HTML5, 2006).

O acesso para os recursos 3D do WebGL é feito através do objeto `WebGLRenderingContext` que é instanciado a partir da chamada do método `getContext()` do elemento *canvas* do HTML5. O WebGL gerencia vários tipos de recursos que são representados como objetos do tipo DOM. Cada objeto é manipulado a partir da interface `WebGLObject`. Os recursos atualmente suportados são: texturas, *buffers* (isso é, *Vertex Buffer Objects* - VBOs), *framebuffers*, *renderbuffers* e *shaders*. *Shaders* são programas escritos em OpenGL ES *Shading Language* (GLSL ES) utilizados para aplicar um comportamento personalizado na renderização de objetos (KHRONOS, 2013).

A especificação do WebGL é muito semelhante ao OpenGL ES, com algumas concessões que são feitas para os desenvolvedores acostumados a trabalhar em linguagens que não tratam gerenciamento de memória, como o JavaScript (KHRONOS, 2013).

2.3 ADUBOGL

O presente trabalho é uma extensão do trabalho intitulado "AduboGL – Aplicação Didática Usando a Biblioteca OpenGL", desenvolvido por Araújo (2012). O AduboGL é um software educacional desenvolvido para informatizar o estudo e aprendizado de alguns conteúdos da disciplina de computação gráfica na FURB. Ele é focado na manipulação de transformações geométricas e foi desenvolvido utilizando a biblioteca OpenGL e linguagem C++ (ARAÚJO, 2012).

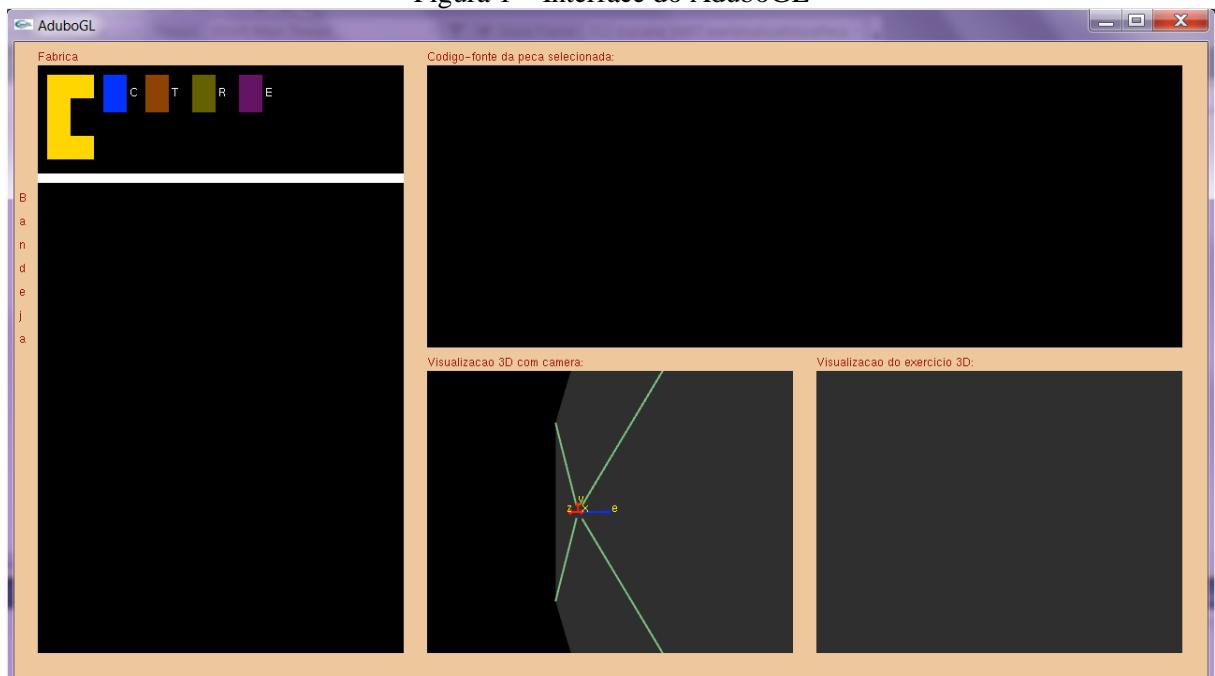
O AduboGL disponibiliza um ambiente 2D onde o usuário dispõe de diversos tipos de peças. Cada uma destas peças representa uma transformação geométrica ou empilhamento de transformações e manipulam uma forma geométrica simples: um cubo. Estas peças devem ser encaixadas para geração de uma cena, levando o usuário a uma montagem correta da lógica de construção das transformações. Em uma janela 3D, o usuário pode visualizar o resultado das transformações definidas pelos encaixes (ARAÚJO, 2012, p. 6).

O software também permite a visualização do código C++ e OpenGL das transformações definidas pelas peças encaixadas, o que permite ao usuário identificar a lógica de construção dos comandos necessários para o desenvolvimento da cena em 3D (ARAÚJO, 2012, p. 6).

A Figura 1 apresenta a interface do AduboGL. No painel superior esquerdo, pode-se observar a existência das peças disponibilizadas pela aplicação: a peça representando o empilhamento e desempilhamento das matrizes de transformação (no formato de um “C”); a

peça cubo (retângulo azul) e as peças de transformações geométricas de translação, rotação e escala (retângulos marrom, verde e roxo). As peças podem ser arrastadas para a bandeja (painel inferior esquerdo) onde o usuário pode montar exercícios envolvendo transformações geométricas sobre um cubo. O resultado das transformações pode ser visualizado nos espaços gráficos disponibilizados nos dois painéis do lado inferior direito. Quando uma peça é selecionada com o mouse, no painel superior direito, o usuário pode visualizar o código OpenGL necessário para gerar o resultado gráfico reproduzido pela peça no espaço gráfico. Também é possível alterar algumas propriedades da peça selecionada usando algumas teclas do teclado (ARAÚJO, 2012, p. 55).

Figura 1 – Interface do AduboGL



Fonte: Araújo (2012, p. 55).

2.4 TRABALHOS CORRELATOS

Atualmente existem diversos softwares voltados para o ensino na área da computação, como o StarLogo TNG (STEP, 2010), conforme será visto adiante. Também pode-se perceber o crescente interesse no desenvolvimento e pesquisa de aplicações utilizando a tecnologia WebGL, visto os diversos motores para renderização em WebGL que tem sido desenvolvidos, como o Three.js (THREEJS, 2013) que é um motor de grafos de cena 3D em JavaScript.

Financiado pelo *Massachusetts Institute of Technology* (MIT), o StarLogo TNG é um sistema de modelagem e simulação desenvolvido pelo MIT's *Scheller Teacher Education Program* (STEP). Ele é voltado para o estudo de sistemas descentralizados, ou seja, sistemas que são formados a partir da interação individual entre vários objetos distintos, em vez de

sistemas que são controlados centralmente. Um exemplo de um sistema descentralizado é a simulação do voo de um bando de pássaros, onde não existe um líder: o comportamento de cada indivíduo é único, porém influencia o comportamento dos demais (STEP, 2010).

O objetivo do software é criar uma experiência mais atrativa das pessoas com a programação, possibilitando o desenvolvimento de sistemas (como jogos 3D) através de uma interface gráfica simples, onde os elementos da linguagem são representados por blocos coloridos que se encaixam como peças de um quebra-cabeça (STEP, 2010).

Na Figura 2 são exibidos os dois painéis principais do StarLogo TNG. No painel 1 é disponibilizado um espaço onde o usuário criar peças, arrastá-las e encaixá-las, organizando e montando o exercício. Cada peça representa um objeto, função ou laço de repetição, que possui propriedades específicas e pode ou não ser complementada por outra peça. Assim o usuário deve escolher os objetos desejados para seu jogo e programar o comportamento esperado para cada objeto. Como o sistema é descentralizado, cada objeto possui seu próprio conjunto de comandos. Um dos objetos padrões que podem ser controlados é uma tartaruga. O resultado da programação feita através das peças pode ser visualizado no painel 2, onde o usuário pode iniciar a execução do exercício construído e visualizar o comportamento dos objetos sobre um plano, representando o chão onde os objetos irão agir (STEP, 2010).

Figura 2 – Principais painéis do StarLogo TNG



Window Name	Contains	Function
1 StarLogoBlocks	Palette of blocks Canvas	Blocks = programming commands Canvas = place to drag and drop blocks to build programs
2 Spaceland	Spaceland Runtime Box	Spaceland = 3-D virtual world that consists of a plane (terrain) and agents (characters, objects) that execute the blocks. Runtime Box = controls the speed of the running programs; programmable user-interface buttons

Fonte: STEP (2010).

O Three.js é um motor 3D JavaScript de código aberto em HTML5 licenciado pelo MIT, que é voltado para a renderização em WebGL, mas também permite a renderização em *Canvas* e *Scalable Vector Graphics* (SVG). Duas características fortes no Three.js são a leveza, a nível de processamento, e a facilidade de uso, a nível de programação (FHTR, 2013).

Existem 5 componentes básicos na estrutura do Three.js: o renderizador, a cena, a câmera, o ponto de luz e os objetos. Cada um desses componentes possui sua respectiva classe e é representado e acessível por uma instância dessas classes no programa a ser desenvolvido. Entre os recursos disponibilizados no Three.js, pode-se destacar o fato dele: funcionar em todos os *browsers* que suportam WebGL; prover o uso de *Shaders*; disponibilizar o uso de câmeras perspectivas e ortográficas assim como vários utilitários para manipulação das mesmas; permitir adicionar e remover objetos da cena em tempo de execução; fornecer o uso de mais de um tipo de luz; permitir o uso de diversos tipos de objetos (malhas, partículas, linhas, etc.), formas geométricas (plano, cubo, esfera, textos 3D, etc.) e materiais (Lambert, Phong, etc. - possuindo texturas, sombras, e muito mais), assim como o fato dele possuir uma boa documentação das principais classes, possuir centenas de exemplos de codificação, além de vídeos, artigos e diversos outros meios de suporte técnico (FHTR, 2013).

O Three.js também permite o carregamento de objetos nos formatos Blender, CTM, FBX, 3D Max e OBJ. Mais detalhes do Three.js serão apresentados adiante.

3 DESENVOLVIMENTO

Este capítulo apresenta todas as etapas de desenvolvimento do aplicativo. Nele são apresentados os principais requisitos, a especificação, a implementação e, por fim, os resultados e discussões.

3.1 PRINCIPAIS REQUISITOS

No ANEXO A são apresentados os principais requisitos do AduboGL - trabalho de pesquisa estendido pelo presente trabalho. Quanto aos requisitos funcionais, o VisEdu-CG deverá:

- a) disponibilizar as funcionalidades existentes no AduboGL (UC01, UC03, UC04, UC05 e UC06);
- b) disponibilizar peças para representar comandos gráficos básicos de CG (UC01);
- c) permitir o encaixe das peças em outras peças que disponibilizem o tipo de encaixe correspondente (UC02);
- d) permitir que o usuário resolva exercícios de computação gráfica que utilizem o conceito de câmera (UC01 a UC07);
- e) permitir que o usuário resolva exercícios de computação gráfica que utilizem um cubo e os conceitos de transformações de translação, rotação e escala (UC01 a UC07);
- f) permitir o agrupamento de transformações sobre um objeto gráfico e seus filhos (UC01 a UC07);
- g) permitir que o usuário resolva exercícios de computação gráfica que utilizem o cubo e o conceito de textura (UC01 a UC07);
- h) permitir que o usuário da aplicação visualize um cenário 3D montado a partir da estrutura das peças encaixadas (UC06);
- i) permitir que o usuário visualize um cenário 3D a partir do ponto de vista definido para a peça que representa uma câmera (UC07);
- j) permitir alterar a cor de limpeza do cenário 3D que representa o ponto de vista definido para a peça que representa uma câmera (UC05 e UC07);
- k) permitir definir um nome para cada peça inserida no editor (UC05);
- l) permitir alterar a posição, o *look at*, o *near*, o *far* e o *Field Of View* (FOV) da peça que representa uma câmera (UC05);
- m) permitir habilitar ou desabilitar a visualização de um objeto gráfico (cubo) no cenário 3D formado a partir das peças encaixadas (UC05);

- n) permitir visualizar a matriz resultante das transformações aplicadas sobre um determinado objeto gráfico (UC05);
- o) permitir que o usuário altere os valores das transformações existentes sobre um determinado objeto gráfico (UC05);
- p) permitir que o usuário habilite ou desabilite as transformações existentes (UC05);
- q) permitir que o usuário altere o tamanho, a posição, a cor e a textura dos cubos existentes no editor (UC05);
- r) disponibilizar uma lista de peças em forma de árvore com todas as peças encaixadas (UC12);
- s) permitir que o usuário selecione uma peça a partir da lista de peças (UC03);
- t) permitir que o usuário selecione uma peça do editor com o mouse (UC03);
- u) permitir a visualização de código em Java OpenGL (JOGL) dos comandos gráficos representados pela peça selecionada (UC04);
- v) permitir que o usuário salve o exercício realizado (UC08);
- w) permitir que o usuário carregue o exercício salvo para continuá-lo (UC09);
- x) permitir que o usuário carregue exercícios prontos de uma lista de exemplos (UC10);
- y) disponibilizar um tutorial de ajuda com informações a respeito do funcionamento da ferramenta (UC11).

Quanto aos requisitos não funcionais, o VisEdu-CG deverá:

- a) ser desenvolvido na linguagem HTML5;
- b) utilizar o WebGL para executar as funções de computação gráfica;
- c) utilizar o framework Three.js;
- d) ser de código aberto;
- e) executar em qualquer navegador que tenha suporte ao WebGL.

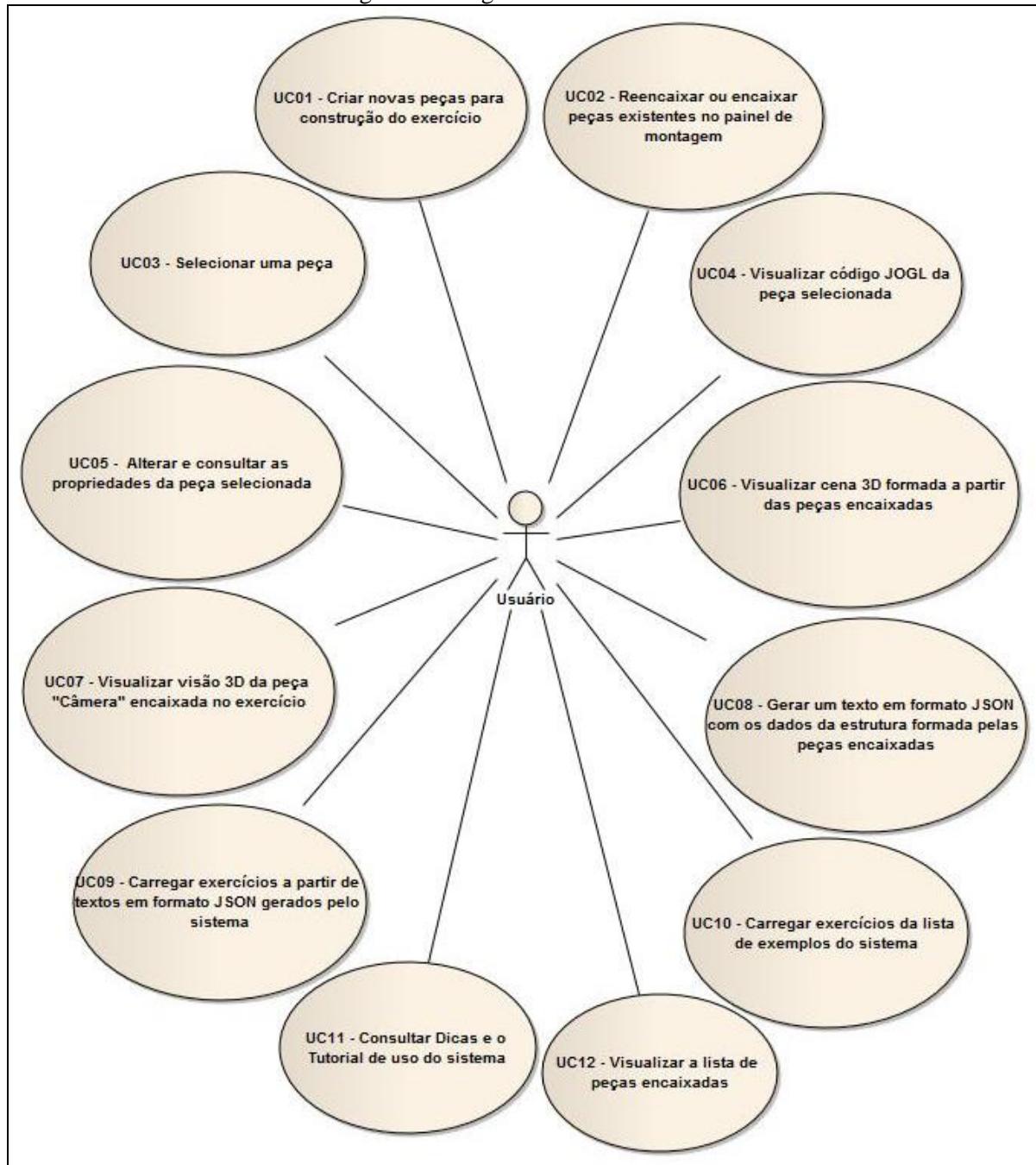
3.2 ESPECIFICAÇÃO

A especificação deste trabalho foi desenvolvida utilizando os diagramas da *Unified Modeling Language* (UML) em conjunto com a ferramenta Enterprise Architect 7.5.845. Para a modelagem da aplicação foram utilizados os diagramas de casos de uso, de classes e de estado, conforme apresentado nas seções seguintes.

3.2.1 Diagrama de casos de uso

Nessa seção são descritos os casos de uso do VisEdu-CG. Por se tratar de uma aplicação voltada à aprendizagem da matéria de computação gráfica, obteve-se apenas um ator que é o próprio usuário do sistema, sendo ele o aluno ou o professor da disciplina. A Figura 3 exibe o diagrama de casos de uso.

Figura 3 - Diagrama de casos de uso



Para compreensão dos casos de uso é necessário ter conhecimento das partes que formam a aplicação. A VisEdu-CG é formada por cinco painéis principais, sendo eles:

- Fábrica de Peças: local onde as peças são disponibilizadas para construção do

exercício. Ele é composto de dois subpainéis: o painel de montagem e a fábrica;

- b) Comandos em JOGL: painel onde o usuário visualiza o código fonte em Java que é necessário para reproduzir o resultado gráfico da peça selecionada usando a biblioteca JOGL;
- c) Lista de Peças: painel com uma lista das peças encaixadas no exercício que permite a visualização e seleção rápida das peças existentes;
- d) Espaço Gráfico: painel com a visualização da cena 3D formada pelo exercício;
- e) Visão da Câmera: é a visão da cena 3D exibida no painel Espaço Gráfico a partir da visão da peça Câmera. Esse painel permite visualizar a cena 3D formada pelo exercício de um ângulo diferente, utilizando as propriedades definidas para a peça Câmera.

O VisEdu-CG ainda disponibiliza outros três painéis que concorrem com a visualização do painel Fábrica de Peças: o painel Arquivo, que exibe opções para exportar ou abrir exercícios; o painel Propriedades da Peça, que exibe as propriedades pertinentes a peça selecionada e o painel Ajuda, que permite consultar informações e dicas a respeito do aplicativo e seu funcionamento. Mais detalhes referentes a estes painéis serão descritos na seção 3.3.2.

Em relação aos casos de uso, o UC01 mostra como o ator Usuário cria peças para interagir com um exercício. O Quadro 1 apresenta os detalhes desse caso de uso.

Quadro 1 - Detalhamento do caso de uso UC01

UC01 – Criar novas peças para construção do exercício	
Descrição	<p>As peças do exercício são aquelas que são encaixadas no painel de montagem para possibilitar a geração da cena 3D e do código fonte. Cada peça representa um comando ou conceito de computação gráfica, como por exemplo: o desenho de um cubo, um objeto gráfico (empilhamento e desempilhamento de matrizes de transformação), as transformações de escala, rotação e translação, ou ainda a visão de uma câmera em um cenário 3D.</p> <p>O painel Fábrica de Peças é composto por dois subpainéis: a fábrica e o painel de montagem. O painel fábrica possui todas as peças que podem ser fabricadas pelo usuário e o painel de montagem apresenta um ambiente para realizar a montagem do exercício através do encaixe das peças fabricadas. Ele possui uma peça principal (Renderizador) que disponibiliza encaixes para as peças Câmera e Objeto Gráfico, permitindo a inicialização do exercício.</p> <p>A peça Objeto Gráfico irá disponibilizar encaixes internos para peças filhas, com encaixes para as peças Cubo, Transladar, Rotacionar, Escalar e Objeto Gráfico.</p>

continua

continuação

Cenário Principal	1. O Usuário clica com o botão esquerdo do mouse em uma peça da fábrica e a arrasta para um dos encaixes livres compatíveis, dentro do painel de montagem, mantendo o botão do mouse pressionado.
Cenário Alternativo 1	1. Se no passo 1 do cenário principal o Usuário soltar a peça dentro da fábrica, a peça sendo arrastada será excluída.
Cenário Alternativo 2	1. Se no passo 1 do cenário principal o Usuário soltar a peça dentro do painel de montagem, fora de um encaixe compatível, a peça sendo arrastada ficará na posição que foi solta (conhecida como peça flutuante).
Pós-Condição	O Usuário deve poder visualizar a nova peça encaixada no painel de montagem. A peça deverá ter um nome automático, conforme um contador interno de inclusão de peças do mesmo tipo. Peças encaixadas alteram o resultado gráfico exibido na cena 3D do exercício. Peças encaixadas são incluídas na Lista de Peças. O Usuário poderá visualizar a peça conforme o cursor do mouse é movimentado e a peça deve acompanhar o cursor.

O UC02 mostra como o ator Usuário altera os encaixes das peças encaixadas ou realiza o encaixe das peças flutuando no painel de montagem. O Quadro 2 apresenta os detalhes desse caso de uso.

Quadro 2 - Detalhamento do caso de uso UC02

UC02 - Reencaixar ou encaixar peças existentes no painel de montagem	
Descrição	As peças já encaixadas no painel de montagem podem ser movidas para outros encaixes compatíveis com o encaixe da peça, alterando assim o resultado obtido pelo exercício. Também é possível encaixar peças que foram soltas no painel de montagem e que não estão encaixadas em nenhum encaixe. Essas peças não encaixadas não afetam o resultado do exercício até serem encaixadas em um encaixe.
Pré-Condição	UC01
Cenário Principal	1. O Usuário clica com o botão direito do mouse em uma peça do painel de montagem e a arrasta para um dos encaixes livres compatíveis, mantendo o botão do mouse pressionado.
Cenário Alternativo 1	1. Se no passo 1 do cenário principal o Usuário soltar a peça dentro do painel fábrica, a peça sendo arrastada será excluída.
Cenário Alternativo 2	1. Se no passo 1 do cenário principal o Usuário soltar a peça dentro do painel de montagem, fora de um encaixe compatível, e ela não estiver encaixada em outra peça (flutuando no painel), a peça sendo arrastada ficará na posição que foi solta.
Cenário Alternativo 3	1. Se no passo 1 do cenário principal o Usuário soltar a peça dentro do painel de montagem, fora de um encaixe compatível, e ela estiver encaixada em outra peça, a peça sendo arrastada continuará encaixada no encaixe de origem e o movimento da peça será cancelado.
Pós-Condição	O Usuário deve poder visualizar a peça posicionada no seu novo encaixe no painel de montagem. Peças encaixadas alteram o resultado gráfico exibido na cena 3D do exercício. A lista de peças será atualizada com o novo encaixe da peça, se existir. O usuário poderá visualizar a peça conforme o cursor do mouse é movimentado e a peça deve acompanhar o cursor.

O UC03 mostra como o ator **Usuário** seleciona uma peça no painel de montagem. O Quadro 3 apresenta os detalhes do caso de uso.

Quadro 3 - Detalhamento do caso de uso UC03

UC03 - Selecionar uma peça	
Descrição	As peças existentes no painel de montagem poderão ser selecionadas, possibilitando a edição de suas propriedades e a visualização do seu respectivo código JOGL.
Pré-Condição	UC01
Cenário Principal	1. O Usuário clica com o botão esquerdo do mouse em uma peça do painel de montagem ou sobre o nome da peça na Lista de Peças.
Pós-Condição	A peça terá uma coloração diferente das demais peças. O painel Propriedades da Peça será exibido sobre o painel fábrica, exibindo a(s) propriedade(s) da peça selecionada. O painel Comandos em JOGL será atualizado com o código fonte da peça.

O UC04 mostra como o ator **Usuário** visualiza o código-fonte JOGL de uma peça ou do exercício. O Quadro 4 apresenta os detalhes do caso de uso.

Quadro 4 - Detalhamento do caso de uso UC04

UC04 - Visualizar código JOGL da peça selecionada	
Descrição	A aplicação permite que o Usuário visualize o código fonte de: a) uma peça: selecionando a mesma; b) de um conjunto de peças: selecionando uma peça que tenha outras peças encaixadas a ela (peças filho); c) uma classe Java pronta para compilar o exercício em um projeto configurado com a biblioteca JOGL: selecionando a peça Renderizador do painel de montagem. Na classe será necessário ajustar o caminho das texturas, caso o exercício faça uso de texturas.
Pré-Condição	UC03
Cenário Principal	1. O Usuário seleciona uma peça.
Pós-Condição	O painel Comandos em JOGL será atualizado com o código fonte correspondente da peça.

O UC05 mostra como o ator **Usuário** altera as propriedades de uma peça. O Quadro 5 apresenta os detalhes do caso de uso.

Quadro 5 - Detalhamento do caso de uso UC05

UC05 - Alterar as propriedades da peça selecionada	
Descrição	É possível consultar e alterar as propriedades das peças existentes no exercício, alterando assim o resultado no cenário 3D e no código fonte. Com isto é possível visualizar o impacto de cada peça e de cada propriedade existente. A peça Renderizador permite modificar a cor de limpeza, que pode ser visualizada pelo painel Visão da Câmera. A peça Câmera permite a definir um nome para a peça, sua posição (x, y, z), o look at (x, y, z), o near, o far e o FOV. A peça Objeto Gráfico permite definir um nome para a peça e se ela será visível na cena. A peça também permite visualizar a matriz resultante das peças de transformações geométricas encaixadas nela. A peça Cubo permite a definir um nome para a peça, o seu tamanho (x, y, z), a sua posição (x, y, z), a sua cor ou a sua textura e se ela será visível na cena.

continua

continuação

	As peças Transladar, Rotacionar e Escalar permitem definir um nome para as peças, o valor da transformação (x, y, z) e se a transformação será visível na cena.
Pré-Condição	UC03
Cenário Principal	1. O Usuário seleciona uma peça. 2. O Usuário altera a propriedade desejada.
Pós-Condição	Os valores definidos para as propriedades deverão ser atualizados no código fonte e nas cenas do resultado 3D do exercício.

O UC06 mostra como o ator Usuário visualiza a cena 3D formada pelo exercício. O Quadro 6 apresenta os detalhes do caso de uso.

Quadro 6 - Detalhamento do caso de uso UC06

UC06 - Visualizar cena 3D formada a partir das peças encaixadas	
Descrição	No painel Espaço Gráfico, conforme o usuário encaixa as peças ou altera as propriedades das mesmas, é possível visualizar o resultado gráfico da peça ou alteração efetuada. Ele exibe um espaço tridimensional que possui 4 componentes principais: a) indicador dos eixos x, y e z: são três retas que indicam a direção dos eixos a partir do ponto [0,0,0] que são identificadas pelas cores vermelho (x), verde (y) e azul (z); b) grade: um plano quadrado em forma de grade, centralizado no ponto [0,0,0], que se estende pelos planos x e z, auxiliando na percepção da profundidade do ambiente; c) pirâmide da câmera: uma pirâmide tridimensional de 4 faces que indica a amplitude da visão da peça Câmera dentro do ambiente gráfico; d) cubos: são cubos que representam as peças cubos encaixadas no editor, exibindo as transformações e propriedade aplicadas sobre os mesmos. É possível ajustar o ponto de vista do Espaço Gráfico usando os botões do mouse: a) botão esquerdo: ao pressionar o botão e arrastar o mouse, o ponto de vista é rotacionado; b) botão direito: ao pressionar o botão e arrastar o mouse, o ponto de vista é transladado; c) botão de rolagem: ao rolar o botão de rolagem do mouse, é possível aumentar ou diminuir o <i>zoom</i> do ponto de vista.
Pré-Condição	UC01, UC02 ou UC05
Cenário Principal	1. Encaixar uma peça no painel de montagem ou editar a propriedade de alguma peça.
Pós-Condição	O conceito gráfico representado pela peça, ou propriedade, deverá ser visualizado ou modificar o ambiente 3D do painel Espaço Gráfico.

O UC07 mostra como o ator Usuário visualiza a cena 3D a partir do ponto de vista definido pela peça Câmera encaixada no painel de montagem. O Quadro 7 apresenta os detalhes do caso de uso.

Quadro 7 - Detalhamento do caso de uso UC07

UC07 - Visualizar visão 3D da peça Câmera encaixada no exercício	
Descrição	Através do painel Visão da Câmera é possível visualizar a cena 3D formada pelo exercício a partir de um segundo ponto de vista, que é determinado pelas propriedades da peça Câmera encaixada no painel de montagem. Os componentes de orientação dentro do painel Espaço Gráfico (os indicadores dos eixos x, y e z, a grade e a pirâmide da câmera) não serão exibidos neste painel. Assim, o único objeto que será reproduzido na cena será o Cubo. A cor de limpeza da cena exibida no painel Visão da Câmera usa a cor de limpeza definida nas propriedades da peça Renderizador do painel de montagem.
Pré-Condição	UC01, UC02 ou UC05
Cenário Principal	1. Encaixar na peça Renderizador a peça Câmera ou editar as propriedades da peça Câmera já encaixada.
Pós-Condição	O painel Visão da Câmera deverá exibir a visão da cena 3D do exercício (também visualizada no painel Espaço Gráfico) a partir do ponto de vista definido pela peça Câmera encaixada na peça Renderizador.

O UC08 mostra como o ator Usuário consegue salvar o exercício criado, gerando um texto em formato JavaScript *Object Notation* (JSON) contendo os dados da estrutura formada pelas peças encaixadas no exercício. O Quadro 8 apresenta os detalhes do caso de uso.

Quadro 8 - Detalhamento do caso de uso UC08

UC08 - Gerar um texto em formato JSON com os dados da estrutura formada pelas peças encaixadas	
Descrição	É possível criar um texto contendo os dados do exercício criado no aplicativo. Através do painel Arquivo é possível gerar um texto em formato JSON com toda a estrutura de peças encaixadas no exercício.
Cenário Principal	1. Selecionar o painel Arquivo; 2. Selecionar a opção Exportar.
Pós-Condição	Será aberta uma nova janela/guia no browser contendo o texto com os dados do exercício.

O UC09 mostra como o ator Usuário consegue abrir exercícios a partir de textos em formato JSON com a mesma estrutura dos textos gerados pela ferramenta. O Quadro 9 apresenta os detalhes do caso de uso.

Quadro 9 - Detalhamento do caso de uso UC09

UC09 - Carregar exercícios a partir de textos em formato JSON gerados pelo sistema	
Descrição	É possível carregar exercícios criados anteriormente no aplicativo. Através do painel Arquivo é possível abrir um texto com a mesma estrutura do formato JSON dos textos gerados pelo aplicativo.
Cenário Principal	1. Selecionar o painel Arquivo; 2. Selecionar a opção Abrir; 3. Selecionar um arquivo contendo os dados de um exercício usando o campo Arquivo ou colar um texto com os dados do exercício no campo Texto Exportado; 4. Selecionar o botão Abrir.
Cenário Alternativo 1	1. Se no passo 3 do cenário principal for informado um texto inválido será exibida uma mensagem de erro.
Pós-Condição	O aplicativo será focado no painel Fábrica de Peças com o exercício informado carregado.

O UC10 mostra como o ator Usuário consegue abrir exercícios existentes na lista de exemplos do aplicativo. O Quadro 10 apresenta os detalhes do caso de uso.

Quadro 10 - Detalhamento do caso de uso UC10

UC10 - Carregar exercícios da lista de exemplos do sistema	
Descrição	É possível carregar exercícios de exemplo do próprio aplicativo. Através do painel Arquivo é possível abrir textos de exemplo com o mesmo formato JSON dos textos gerados pelo aplicativo. Esses exemplos contêm exercícios criados para demonstrar o uso das diversas peças existentes no aplicativo.
Cenário Principal	1. Selecionar o painel Arquivo; 2. Selecionar a opção Abrir; 3. Selecionar um dos itens do campo Lista de Exemplos; 4. Selecionar o botão Abrir.
Cenário Alternativo 1	1. Se antes de se executar o passo 4 do cenário principal, o texto carregado no campo Texto Exportado pelo passo 3 do cenário principal for alterado de forma inválida, será exibida uma mensagem de erro.
Pós-Condição	O aplicativo será focado no painel Fábrica de Peças com o exercício selecionado carregado.

O UC11 mostra como o ator Usuário consegue consultar informações e dicas a respeito do aplicativo e seu funcionamento. O

Quadro 11 apresenta os detalhes do caso de uso.

Quadro 11 - Detalhamento do caso de uso UC11

UC11 - Consultar Dicas e o Tutorial de uso do sistema	
Descrição	Através do painel Ajuda, o aplicativo disponibiliza um tutorial de ajuda, com informações e dicas sobre o uso do sistema e seu desenvolvimento. O painel possui 3 opções: a) Ajuda: exibe um painel com os tutoriais de ajuda do aplicativo. b) Dicas: exibe um painel com o tutorial de dicas para uso do aplicativo; c) Sobre: exibe um painel com informações a respeito do aplicativo e sua autoria.
Cenário Principal	1. Selecionar o painel Ajuda; 2. Selecionar a opção Ajuda, Dicas ou Sobre.
Cenário Alternativo 1	1. Ao clicar duas vezes com o botão esquerdo do mouse sobre qualquer painel do sistema, automaticamente é aberta a janela de tutorial de ajuda, focado na seção de ajuda do painel selecionado.
Pós-Condição	O aplicativo abrirá uma janela com o tutorial de ajuda e de informações da ferramenta.

Por fim, o UC12 mostra como o ator Usuário consegue visualizar a lista de peças encaixadas no sistema. O Quadro 12 apresenta os detalhes do caso de uso.

Quadro 12 – Detalhamento do caso de uso UC12

UC12 - Visualizar a lista de peças encaixadas	
Descrição	Através do painel Lista de Peças, o aplicativo disponibiliza uma lista das peças encaixadas, que é montada conforme o usuário encaixa as peças no painel de montagem.

continua

continuação

Pré-Condição	UC01 ou UC02
Cenário Principal	<ol style="list-style-type: none"> 1. Visualizar o painel. 2. Esconder ou exibir as peças filhos.

3.2.2 Diagrama de classes

A Figura 4 apresenta o diagrama contendo o conjunto de pacotes que compõem o sistema e o relacionamento existente entre eles. Nela pode-se observar os *frameworks* e classes do sistema, dentro de seus respectivos pacotes. Neste diagrama foram omitidas as classes pertencentes aos *frameworks*, salvo os que são compostos de poucas classes. Adiante serão esclarecidos os recursos utilizados de cada *framework*, iniciando pelos menos complexos. Logo após será exposto um detalhamento de cada classe do sistema, partindo das classes de menor complexidade.

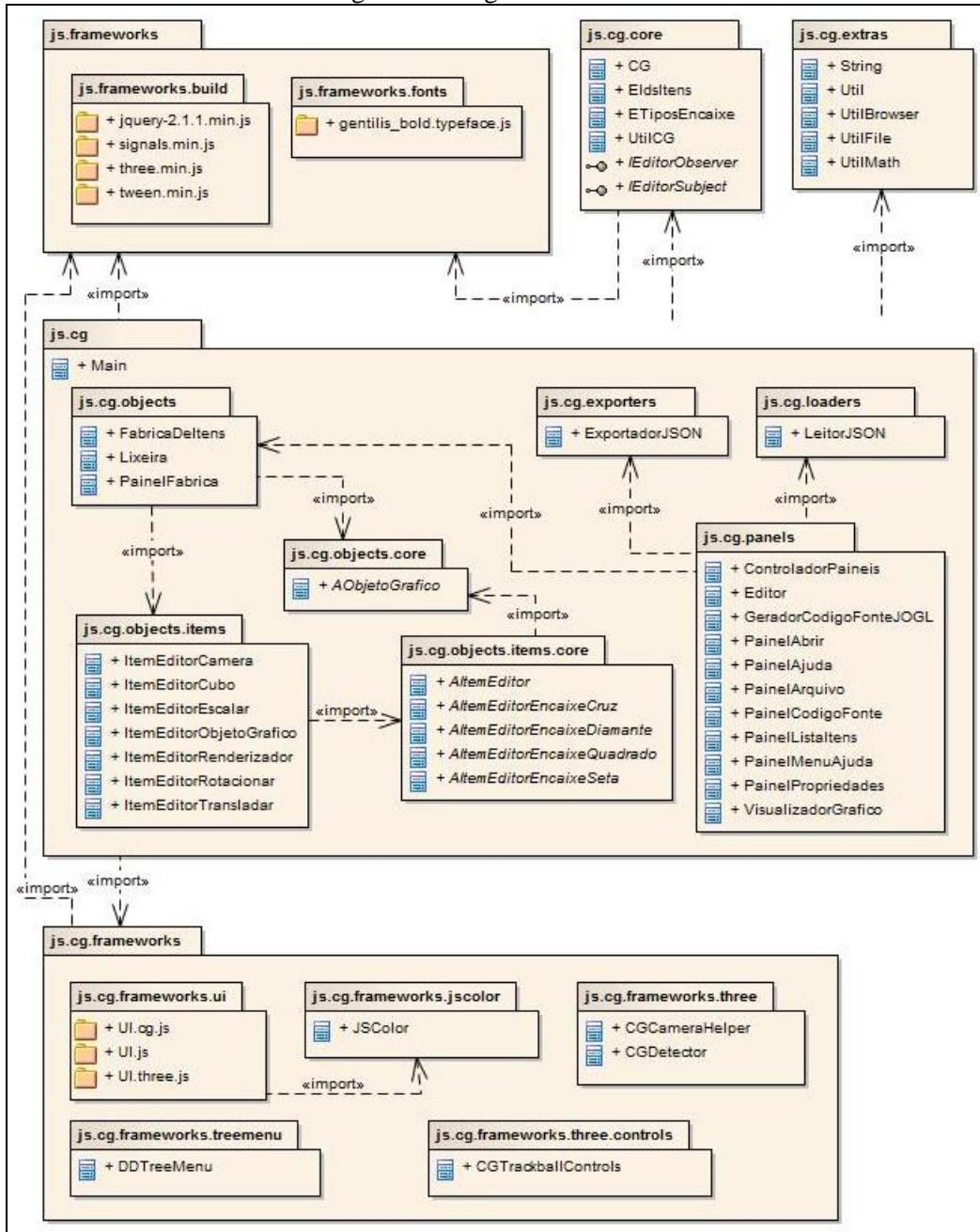
No diagrama é possível observar a existência de dois pacotes de *frameworks*: o pacote `js.cg.frameworks` onde estão incluídos os *frameworks* que precisaram ser ajustados para atender algumas necessidades específicas do sistema e o pacote `js.frameworks` onde estão incluídos os *frameworks* que não precisaram ser ajustados.

O pacote `js.cg.frameworks` é dividido em 5 subpacotes, cada um agrupando um *framework* específico: o pacote `js.cg.frameworks.treemenu` contendo o *framework* DDTreeMenu, que é responsável por construir e exibir a lista de itens contida no painel `Lista de Itens`; o pacote `js.cg.frameworks.jscolor` contendo o *framework* JSColor, que é responsável por exibir a paleta de cores na edição das propriedades do tipo cor das peças; o pacote `js.cg.frameworks.three` contendo classes do *framework* Three.js que precisam ser adaptadas para o sistema; o pacote `js.frameworks.three.controls` contendo uma biblioteca utilitária do ThreeJS usada para o controle da câmera de um ambiente 3D via mouse e o pacote `js.cg.frameworks.ui` contendo o biblioteca UI, que é responsável pela construção e manipulação dos elementos HTML do sistema. As classes a serem ajustadas do Three.js serão renomeadas para não criar conflito com as classes originais, assim o trecho "THREE." do nome original das classes será substituído por "CG", salvo a classe `CGDetector`, cujo nome original é `Detector`.

O pacote `js.frameworks` é dividido em três subpacotes: o pacote `js.frameworks.fonts` contendo uma biblioteca para renderização de textos 3D e o pacote `js.frameworks.build` contendo os *frameworks* JQuery, Signals, Tween e Three.js. O JQuery é uma biblioteca para manipulação do documento HTML via JavaScript. Signals, é uma biblioteca para auxiliar na criação de eventos e na chamada desses eventos e é usada no

sistema para controlar os eventos de redimensionamento do tamanho da página do navegador. Tween é uma biblioteca que realiza a movimentação elástica de objetos em um ambiente 3D usando as equações de Robert Penner. Quanto ao Three.js, a Figura 5 exibe as principais classes deste *framework* que são utilizadas pelo sistema. Visto que a maior parte das classes do sistema fazem uso desse *framework* ou foram construídas baseadas nele, é importante o conhecimento de sua estrutura a fim de facilitar a compreensão da estrutura do sistema.

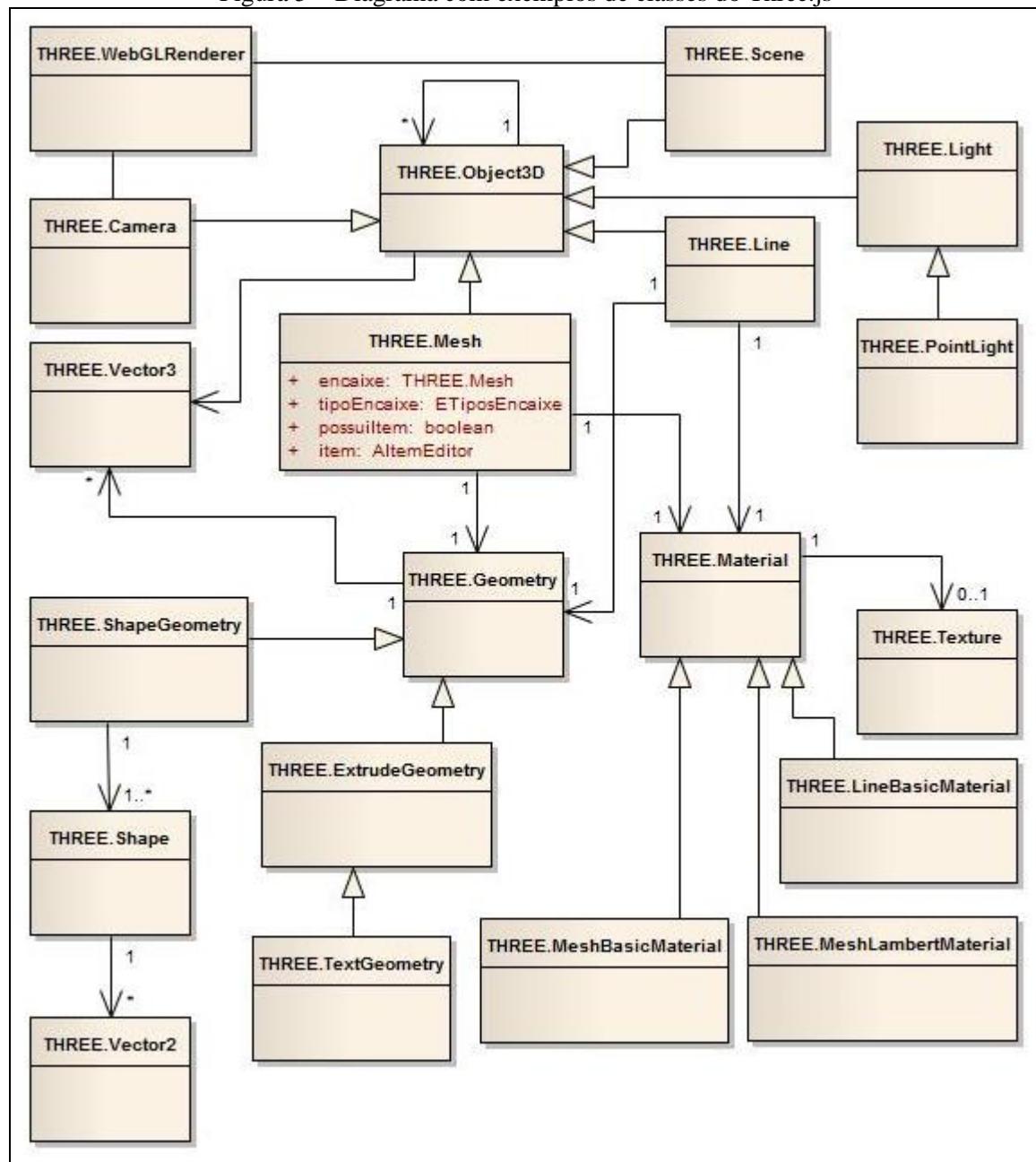
Figura 4 - Diagrama de classes



No diagrama a primeira classe exibida é a classe `THREE.WebGLRenderer`, que é responsável pela renderização do ambiente 3D na página. Quando instanciada ela cria internamente o elemento HTML onde o ambiente 3D será renderizado. Este elemento deve

ser incluído dinamicamente no documento HTML. Através das propriedades da classe é possível parametrizar o tamanho do elemento (largura e altura). Para renderizar um ambiente, usa-se a função `render` que tem 2 parâmetros: um objeto do tipo `THREE.Camera` e um do tipo `THREE.Scene`.

Figura 5 – Diagrama com exemplos de classes do Three.js



No Three.js, `THREE.Object3D` é a classe base para qualquer classe que irá representar um componente na cena, inclusive das classes `THREE.Camera` e `THREE.Scene`. A classe `THREE.Object3D` permite adição de filhos do mesmo tipo e possui sua própria matriz de transformação. `THREE.Scene` é a classe central de uma cena e todo objeto que precisa ser renderizado deve ser adicionado como filho de um objeto dessa classe. Assim, câmeras, luzes

e objetos devem ser incluídos como filhos de um objeto THREE.Scene. THREE.Camera é classe base abstrata para as classes THREE.OrthographicCamera e THREE.PerspectiveCamera, e através destas é possível criar o tipo de câmera desejado, editar suas propriedades e incluí-la na cena. THREE.Light é a classe base abstrata para as classes dos vários tipos de luzes disponíveis no Three.js, como a classe THREE.PointLight, que cria um ponto de luz onde é possível editar a intensidade e a distância de alcance da luz.

Todo objeto geométrico visível em uma cena é representado pela classe base abstrata THREE.Mesh. A classe THREE.Mesh agrupa o tipo de material de um objeto (THREE.Material) e sua informação geométrica (THREE.Geometry). Esse agrupamento define como o objeto será renderizado. As classes do tipo THREE.Geometry resumem-se na lista de pontos do objeto geométrico. As classes do tipo THREE.Material tem a informação do tipo de material que deve ser gerado usando a informação geométrica. Dependendo do tipo de material instanciado, uma mesma informação geométrica irá exibir resultados diferentes na renderização. O Three.js possui vários tipos de classes geométricas e de material.

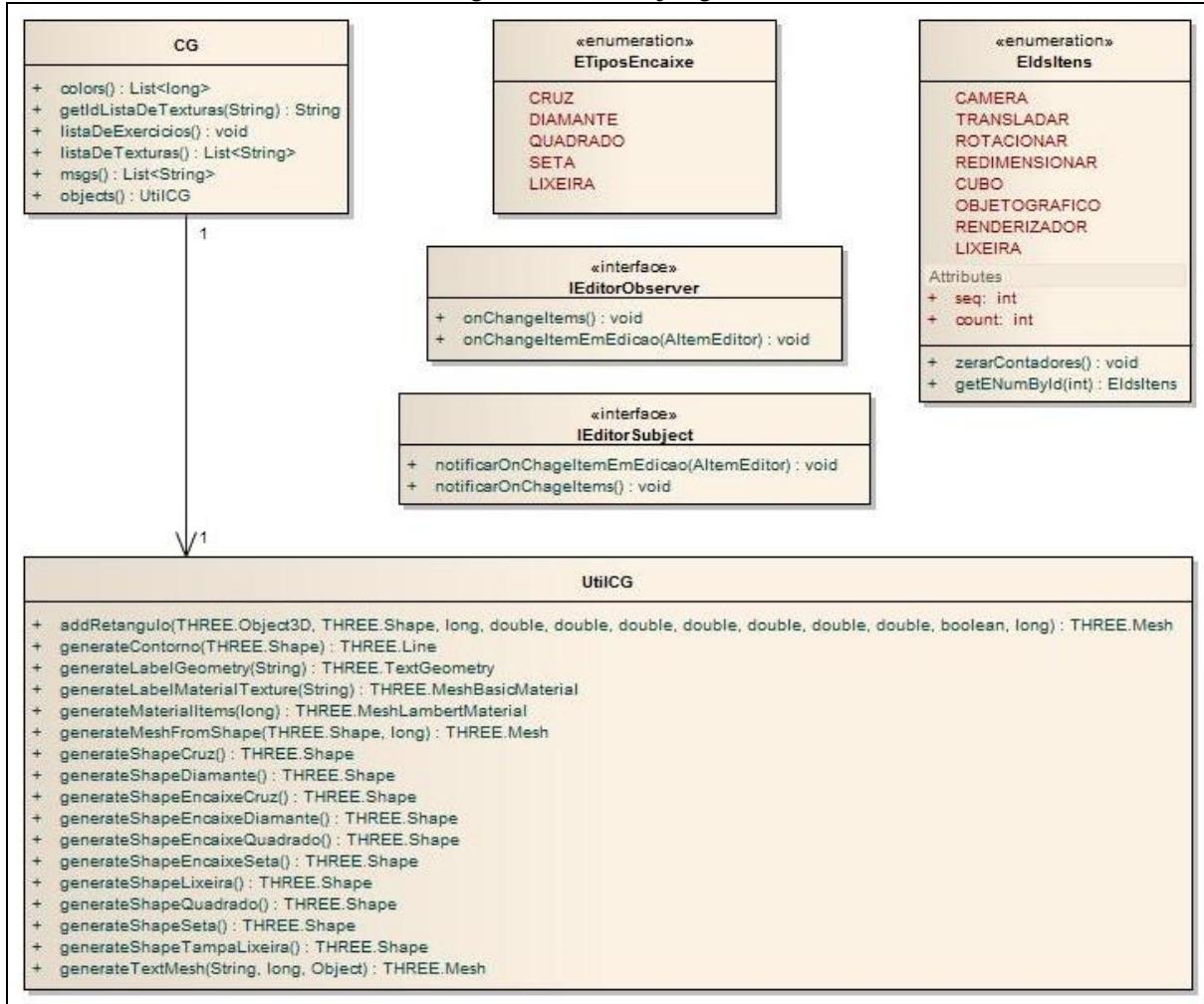
No diagrama são exibidas as classes THREE.Shape e THREE.ShapeGeometry que possuem funções auxiliares para a criação da informação geométrica de planos. Também são exibidas as classes THREE.TextGeometry e THREE.ExtrudeGeometry, que também criam informação geométrica, mas para textos 3D.

Depois de criar a informação geométrica, é necessário definir o material adequado para renderizar o objeto conforme o resultado que se espera. A classe THREE.LineBasicMaterial irá criar apenas uma linha ligando os pontos da informação geométrica. A classe THREE.MeshBasicMaterial irá desenhar formas geométricas com superfície simples, sem influência de luz. A classe THREE.MeshLambertMaterial cria uma superfície não brilhante. Para um material pode-se ainda definir uma textura associando ao material um objeto do tipo THREE.Texture.

Sendo o THREE.Mesh o tipo base de qualquer objeto geométrico visível no ambiente 3D, algumas propriedades são adicionadas a esta classe. Isso auxiliará na seleção e encaixe dos objetos do painel de montagem, visto que as funções utilitárias de seleção de objetos do Three.js retornam apenas esse tipo de objeto.

Todas as classes criadas ou adaptadas para o sistema estão contidas no pacote js.cg e em seus subpacotes. Na Figura 6 são exibidas as classes do pacote js.cg.core. Este pacote contém as classes bases utilizadas pelas demais classes do sistema.

Figura 6 – Pacote js.cg.core



A enumeração `EIdsItens` possui um identificador para cada item existente na Fábrica de Peças: Renderizador, Câmera, Objeto Gráfico, Transladar, Rotacionar, Escalar, Cubo e Lixeira. Esta enumeração é utilizada para identificar o tipo de item e efetuar o correto tratamento do mesmo. Cada identificador da enumeração possui um contador próprio, que é utilizado para armazenar quantos objetos daquele tipo já foram gerados, e um número sequencial único, que é usado para identificar o tipo de objeto no texto gerado com os dados do exercício.

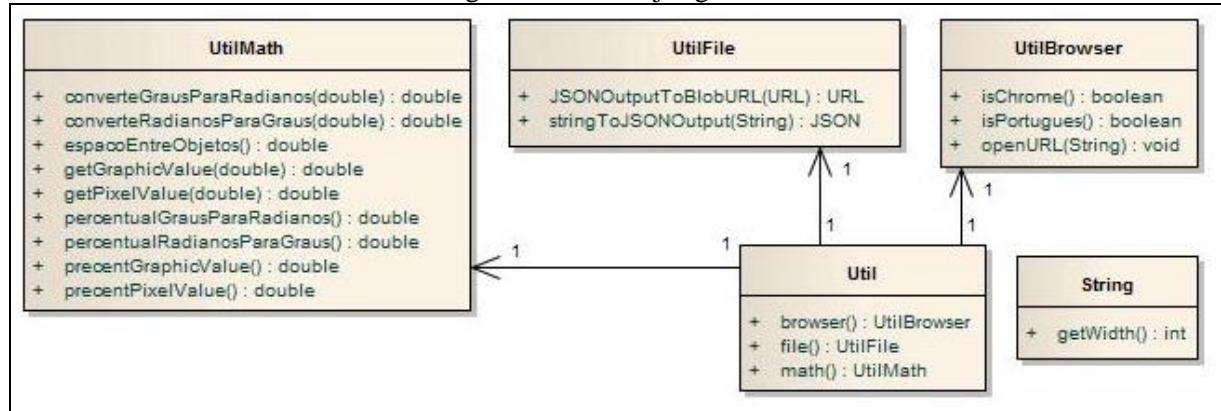
A enumeração `ETiposEncaixe` possui um identificador para cada tipo de peça existente no sistema: Cruz, Diamante, Quadrado, Seta e Lixeira. Essa enumeração permite o sistema comparar o tipo de encaixe de origem e destino das peças e efetuar o correto tratamento das operações de encaixe e exclusão. As interfaces `IEditorObserver` e `IEditorSubject` são utilizadas para implementar os eventos de atualização dos painéis do sistema quando é realizada alguma alteração nas propriedades dos itens ou no painel de

montagem. Essas interfaces garantem a implementação do padrão de projeto *Observer* na notificação dessas alterações.

A classe `UtilCG` possui funções utilitárias para geração dos objetos gráficos utilizados no sistema: peças, encaixes, retângulos e textos. Ela é disponibilizada no sistema através da classe `CG`. A classe `CG` é a classe que contém todas as funções utilitárias aplicáveis exclusivamente ao sistema e implementa o padrão de projeto *Singleton*. Além de acoplar os recursos da classe `UtilCG`, ela possibilita centralizar a configuração das cores utilizadas no sistema, das mensagens de aviso, da lista de texturas disponível nas propriedades dos cubos e da lista de exemplos disponível na tela Arquivo > Abrir.

Na Figura 7 são apresentadas as classes do pacote `js.cg.extras`. Neste pacote estão contidas classes utilitárias que não são aplicáveis somente ao sistema, ou seja, que poderiam ser reutilizadas por outros sistemas.

Figura 7 – Pacote `js.cg.extras`



Na classe `String` (nativa do JavaScript) foi adicionada uma função que permite verificar o tamanho aproximado em *pixels* que a *string* ocuparia na tela. A classe `UtilBrowser` possui recursos para verificar o tipo de navegador, a linguagem do mesmo, assim como outras funções relacionadas ao navegador de internet. A classe `UtilFile` permite manipular arquivos JSON, convertendo uma *string* para objetos no formato JSON e convertendo objetos JSON para uma *blob Uniform Resource Locator* (URL). A classe `UtilMath` auxilia com cálculos matemáticos, fazendo conversões entre graus e radianos ou pixels e coordenadas gráficas. A classe `Util` implementa o padrão de projeto *Singleton*. Ela acopla os recursos das classes `UtilBrowser`, `UtilFile` e `UtilMath` e disponibiliza esses recursos para o resto do sistema.

Na Figura 8 está a representação do pacote `js.cg.exporters`. Este pacote contém as classes utilitárias para disponibilizar o armazenamento e compartilhamento dos dados existentes no sistema. A classe `ExportadorJSON` realiza a conversão do exercício existente no

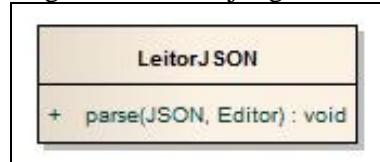
painel de montagem para estrutura de objetos no formato JSON, o que irá possibilitar a geração de uma *string* com os dados do exercício através do uso da classe `Util`.

Figura 8 - Pacote js.cg.exporters



Na Figura 9 está a representação do pacote `js.cg.loaders`. Este pacote contém as classes utilitárias para realizar a leitura de dados exportados do sistema. A classe `LeitorJSON` realiza a leitura de uma estrutura de objetos no formato JSON, e carrega essa estrutura no painel de montagem.

Figura 9 - Pacote js.cg.loaders



A Figura 10 apresenta as classes do pacote `js.cg.objects.core`. O pacote contém as classes bases para o controle dos objetos gráficos manipuláveis do sistema: as peças e a lixeira.

A classe abstrata `AObjetoGrafico` possui recursos que auxiliam no controle da seleção, atualização e hierarquia existente entre os objetos no painel Fábrica de Peças. As propriedades `objeto` e `objetoDetalhes` permitem definir dois conjuntos de objetos do Three.js que irão ser renderizados. A classe permite definir se os objetos da propriedade `objetoDetalhes` serão, ou não, renderizados junto com os objetos da propriedade `objeto`. Ela também é responsável pelo controle de hierarquia entre peças, onde a propriedade `pai` indica a peça em que o objeto está encaixado e a propriedade `filhos` indica as peças que estão encaixadas no objeto. As funções abstratas permitem a atualização do objeto quando seus filhos são alterados ou removidos. A propriedade `insertableMeshes` indica quais dos objetos existentes nas propriedades `objeto` e `objetoDetalhes` podem ser localizados pelas funções de seleção do editor.

Na Figura 11 pode-se visualizar as classes do pacote `js.cg.objects.items.core`. Este pacote contém as classes bases para a manipulação e renderização das peças do painel Fábrica de Peças.

A classe abstrata `AItemEditor` estende a classe `AObjetoGrafico` e complementa alguns controles de seleção. Ela armazena algumas propriedades comuns a todas as peças

(Nome e Visível/Ativo) e permite definir se essas propriedades podem ser exibidas e editadas no painel Propriedades da Peça. Esta classe é responsável pelo desenho do cabeçalho da peça (retângulo com o recorte do encaixe e o nome da peça). Ela possui uma função abstrata que permite as classes filhos definirem o tipo de encaixe que será desenhado. Os objetos do Three.js necessários para renderizar o cabeçalho da peça são incluídos na propriedade `objeto` (herdada de `AObjetoGrafico`), permitindo que o editor localize e renderize esses objetos.

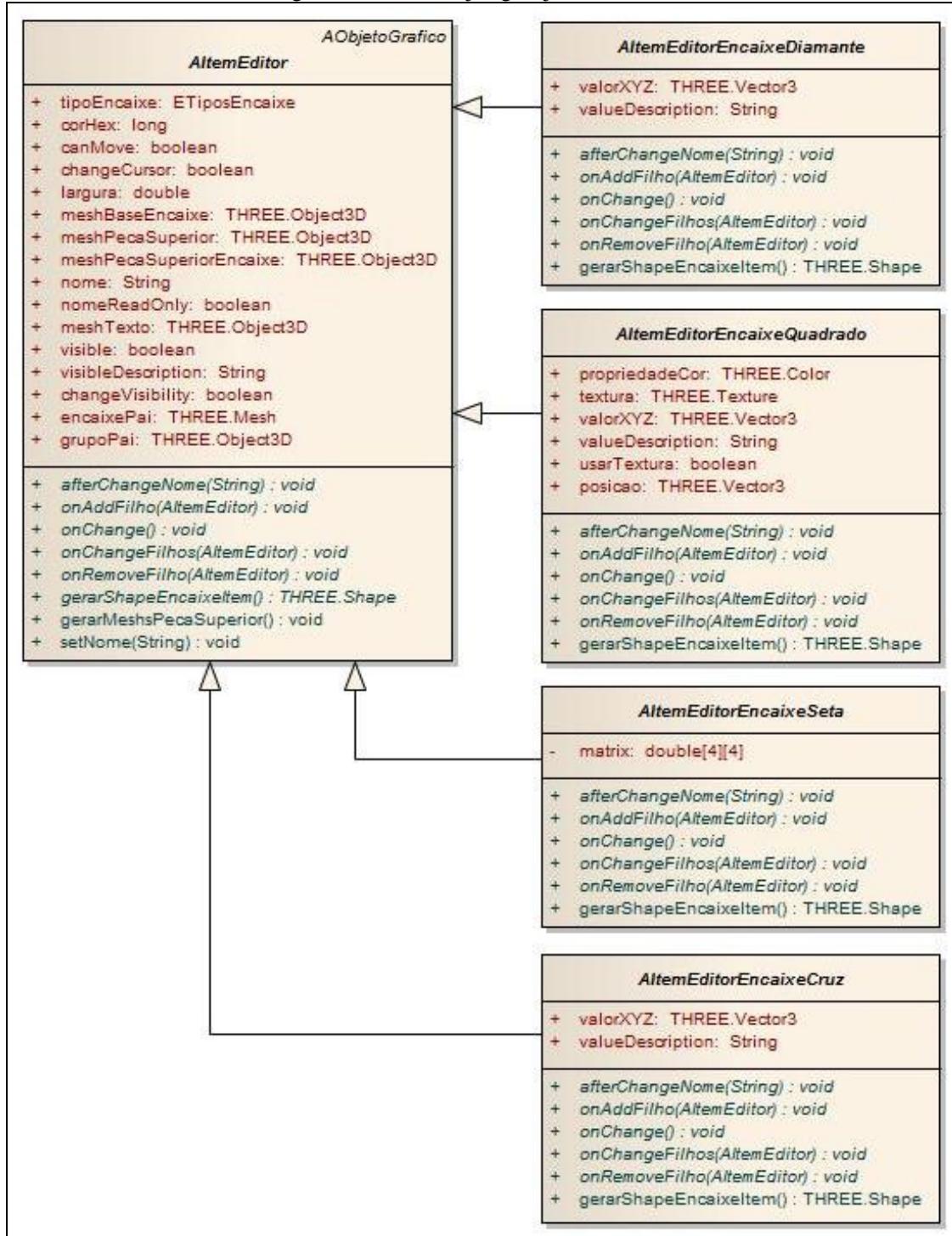
Figura 10 – Pacote js.cg.objects.core



As propriedades `meshbaseEncaixe`, `meshPecaSuperior`, e `meshPecaSuperiorEncaixe` armazenam os objetos do Three.js que desenham o retângulo e o recorte de encaixe da peça. Esses três objetos também são incluídos na propriedade `intersectableMeshs` (herdada de `AObjetoGrafico`), permitindo que o editor identifique a peça como um objeto selecionável. A propriedade `canMove` define se a peça poderá ser movimentada no editor. As propriedades `nome` e `nomeReadOnly` guardam no nome da peça e definem se o nome será exibido e editável no painel Propriedades da Peça. As propriedades `visible`, `visibleDescription` e `changeVisibility` definem se a peça causará efeito no painel Visualizador Gráfico, a descrição que a propriedade terá no painel Propriedades da Peça (“Visível” ou “Ativo”) e se essa propriedade será exibida. As

propriedades `encaixePai` e `grupoPai` armazenam *links* para os objetos em que a peça está encaixada, facilitando o desencaixe das peças.

Figura 11 - Pacote js.cg.objects.items.core



As classes abstratas `AltemEditorEncaixeDiamante`, `AltemEditorEncaixeQuadrado`, `AltemEditorEncaixeSeta` e `AltemEditorEncaixeCruz` estendem a classe `AltemEditor` e, além de serem responsáveis pela implementação da função `gerarShapeEncaixeItem()`, elas armazenam as propriedades específicas de cada peça. A função `gerarShapeEncaixeItem()`

irá retornar o desenho do recorte que será desenhado para a classe. As propriedades `valorXYZ` e `valueDescription` das classes `AItemEditorEncaixeDiamante`, `AItemEditorEncaixeQuadrado` e `AItemEditorEncaixeCruz` armazenam um valor para as coordenadas X, Y e Z e a descrição que a propriedade terá no painel Propriedades da Peça (“Tamanho”, “Posição” ou “Valor”).

A classe `AItemEditorEncaixeQuadrado` possui também outras propriedades: `propriedadeCor`, que permite definir uma cor para o objeto representado pela peça; `textura`, que permite definir uma textura para o objeto; `usarTextura`, que habilita ou desabilita a renderização da textura para o objeto, e `posicao`, que define a posição X, Y e Z do objeto. A classe `AItemEditorEncaixeSeta` possui também a propriedade `matrix`, que contém a matriz de transformação resultante das peças de transformação que serão encaixadas na peça.

Na Figura 12 são apresentadas as classes do pacote `js.cg.objects.items`. O pacote contém as classes que representam cada item existente no painel Fábrica de Peças.

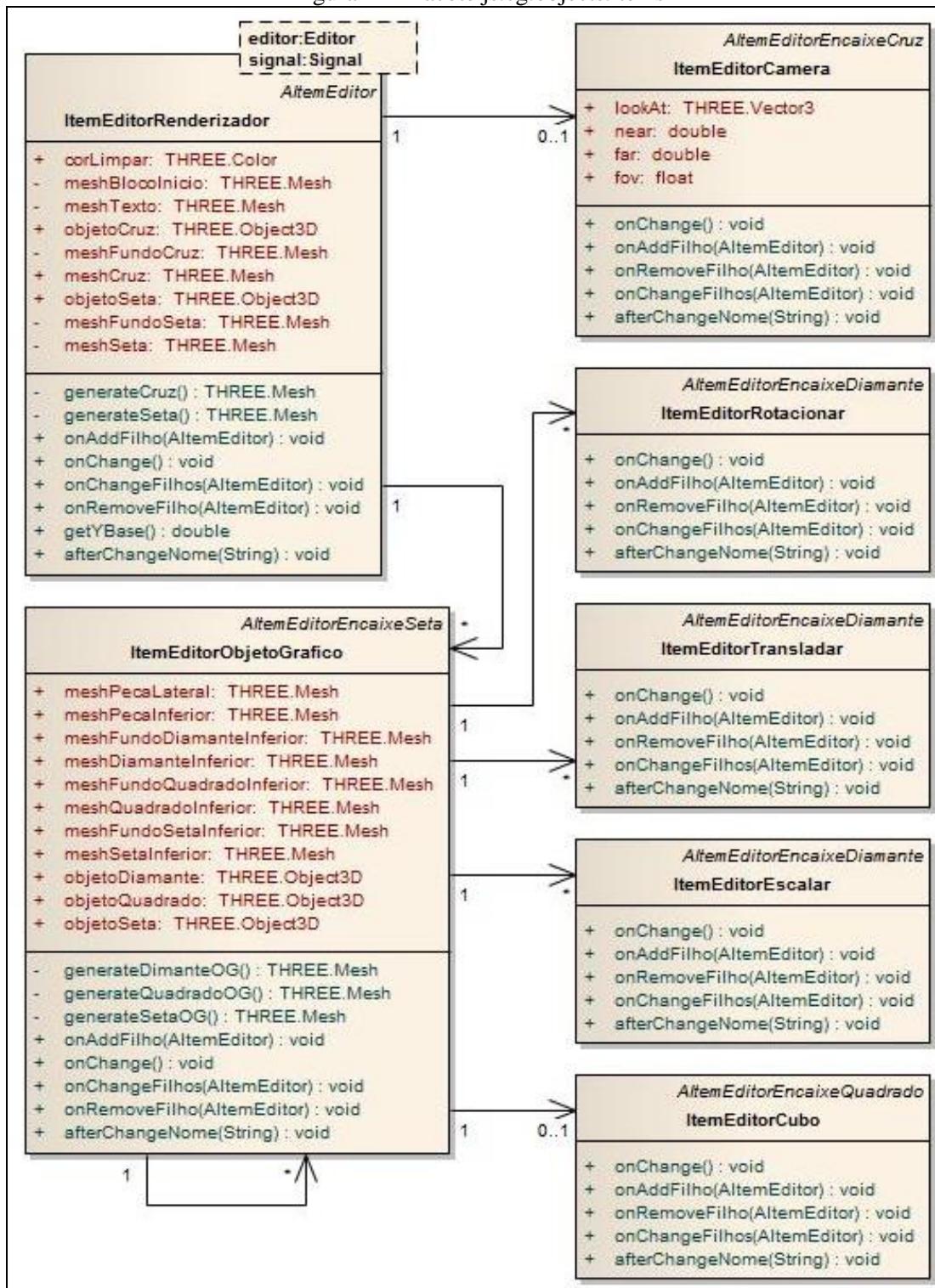
As classes `ItemEditorRenderizador`, `ItemEditorObjetoGrafico`, `ItemEditorCamera`, `ItemEditorRotacionar`, `ItemEditorTransladar`, `ItemEditorEscalar` e `ItemEditorCubo` implementam as funções abstratas herdadas das classes pais e definem algumas propriedades, também herdadas, permitindo assim que o sistema possa renderizar esses itens e manipulá-los corretamente. A classe `ItemEditorCamera` possui algumas propriedades específicas para a peça Câmera.

A classe `ItemEditorObjetoGrafico` controla os objetos que renderizam a parte inferior da peça e é responsável pelo reposicionamento dos itens encaixados nela. Os objetos que renderizam a parte inferior da peça são incluídos na propriedade `objetoDetalhes` (herdada de `AObjetoGrafico`), permitindo que o editor localize e renderize esses objetos. Os objetos existentes na propriedade `objetoDetalhes` são exibidos apenas quando a peça é arrastada para o painel de montagem. As propriedades `meshPecaLateral` e `meshPecaInferior` guardam os objetos do `Three.js` que renderizam o desenho em forma de "L", ligado ao cabeçalho, que delimita o espaço ocupado pela peça. Esses dois objetos também são incluídos na propriedade `objetoDetalhes`.

As propriedades `meshDiamanteInferior`, `meshQuadradoInferior` e `meshSetaInferior` guardam os objetos do `Three.js` que renderizam o desenho dos encaixes. As propriedades `meshFundoDiamanteInferior`, `meshFundoQuadradoInferior` e `meshFundoSetaInferior` armazenam os objetos do `Three.js` que renderizam um retângulo invisível a frente do desenho dos encaixes. Esses retângulos possibilitam que o editor

identifique quando a seta do mouse se aproxima do encaixe. Esses três pares de objetos são ligados um ao outro pelas propriedades adicionadas a classe `THREE.Mesh` (exibidas na figura 3), permitindo ao editor identificar a relação entre os objetos e validar o tipo de encaixe.

Figura 12 - Pacote js.cg.objects.items



Para serem renderizados, os objetos das propriedades `meshDiamanteInferior` e `meshFundoDiamanteInferior` são agrupados através da sua inclusão na propriedade

`objetoDiamante`. Os objetos das propriedades `meshQuadradoInferior` e `meshFundoQuadradoInferior` são agrupados através da sua inclusão na propriedade `objetoQuadrado`. Os objetos das propriedades `meshSetaInferior` e `meshFundoSetaInferior` são agrupados através da sua inclusão na propriedade `objetoSeta`. Os objetos das propriedades `objetoDiamante`, `objetoQuadrado` e `objetoSeta` são incluídos na propriedade `objetoDetalhes`, o que possibilita ao editor localizar esses objetos e renderizá-los.

A classe `ItemEditorRenderizador` renderiza a peça principal do exercício. Como a peça não possui um recorte de encaixe, a classe não faz uso dos recursos de desenho de cabeçalho disponíveis na classe `AItemEditor`. Contudo, ela utiliza as demais propriedades desta classe. Os parâmetros de instanciação da classe são um objeto da classe `Editor` e um da classe `Signals`. Quando as peças filhos são alteradas, ela chama os métodos de notificação do objeto editor, assinados pela classe `IEditorSubject`. O objeto `Signals` é usado para a criação do evento de redimensionamento do painel, que é chamado quando o tamanho da página do navegador é alterado. A propriedade `corLimpar` permite definir nas Propriedades da Peça a cor de limpeza da peça Câmera, que pode ser visualizada também no painel Visão da Câmera.

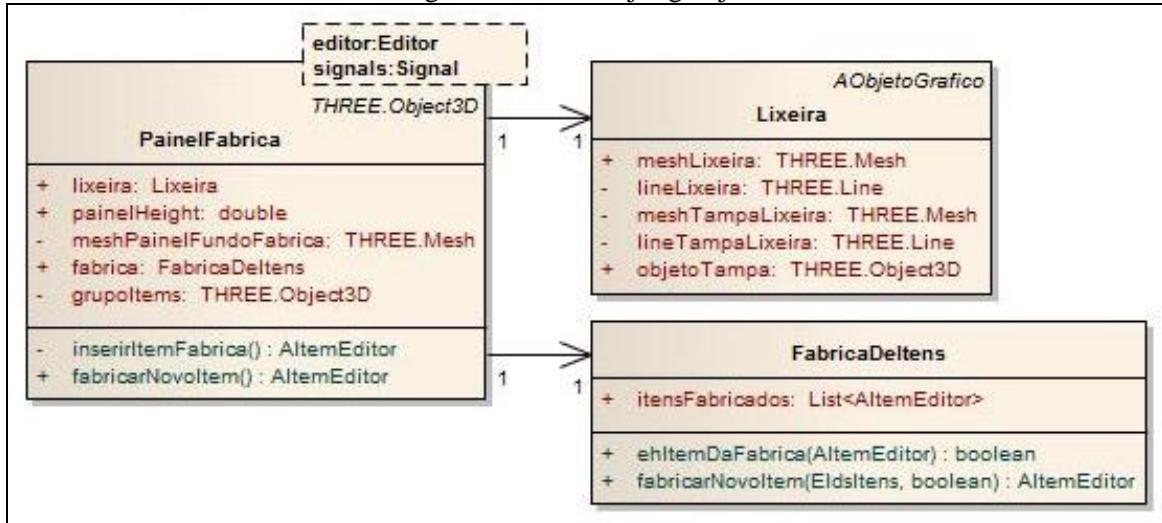
As propriedades `meshBlocoInicio` e `meshTexto` da classe `ItemEditorRenderizador` armazenam os objetos Three.js que renderizam o cabeçalho e o retângulo lateral da peça. Esses objetos também são incluídos na propriedade `objeto` (herdada de `AObjetoGrafico`), permitindo que o editor localize e renderize esses objetos. As propriedades `meshCruz` e `meshSeta` guardam os objetos do Three.js que renderizam o desenho dos encaixes. As propriedades `meshFundocruz` e `meshFundoseta` armazenam os objetos do Three.js que renderizam um retângulo invisível a frente do desenho dos encaixes, permitindo que seja possível identificar quando o mouse se aproxima do encaixe. Os objetos das propriedades `meshCruz` e `meshFundocruz` são agrupados através da sua inclusão na propriedade `objetoCruz`. Os objetos das propriedades `meshSeta` e `meshFundoseta` são agrupados através da sua inclusão na propriedade `objetoSeta`. `meshFundosetaInferior` são agrupados através da sua inclusão na propriedade `objetoSeta`. Os objetos das propriedades `objetoCruz` e `objetoSeta` são incluídos na propriedade `objetoDetalhes` (herdada de `AObjetoGrafico`), permitindo que o editor localize e renderize esses objetos.

No diagrama é possível observar a compatibilidade de encaixe existente entre as peças. Essa compatibilidade é demonstrada pelo relacionamento existente entre as classes. A peça

Renderizador (`ItemEditorRenderizador`) permite o encaixe de uma peça Câmera (`ItemEditorCamera`) e várias peças Objeto Gráfico (`ItemEditorObjetoGrafico`). A peça Objeto Gráfico permite o encaixe de várias peças Rotacionar (`ItemEditorRotacionar`), Transladar (`ItemEditorTransladar`), Escalar (`ItemEditorEscalar`) ou Objeto Gráfico e o encaixe de uma peça Cubo (`ItemEditorCubo`). As demais peças não permitem o encaixe de peças filhos.

Na Figura 13 são apresentadas as classes do pacote `js.cg.objects`. O pacote contém as classes que renderizam e auxiliam no controle do painel fábrica, subpaineis do painel Fábrica de Peças.

Figura 13 – Pacote `js.cg.objects`



A classe `Lixeira` é responsável por renderizar a lixeira no painel fábrica. As propriedades `meshLixeira` e `lineLixeira` guardam os objetos do Three.js que renderizam a parte inferior da lixeira e seu contorno. Esses objetos também são incluídos na propriedade `objeto` (herdada de `AObjetoGrafico`), permitindo que o editor localize e renderize esses objetos. As propriedades `meshTampaLixeira` e `lineTampaLixeira` guardam os objetos do Three.js que renderizam a tampa da lixeira e seu contorno. Esses dois objetos são agrupados através da sua inclusão na propriedade `objetoTampa`. O objeto da propriedade `objetoTampa` é incluído na propriedade `objeto` permitindo a renderização dos demais objetos agrupados nele.

A classe `FabricaDeItens` implementa o padrão de projeto *Factory*, centralizando e facilitando a criação das peças. A propriedade `itensFabricados` guarda os itens produzidos pela fábrica que compõem as peças que são listadas no painel Fábrica (itens que estão disponíveis para fabricação). A função `ehItemDaFabrica()` é usada pelo editor para verificar se a peça selecionada pertence ao painel fábrica.

A classe `PainelFabrica` é responsável por renderizar o painel fábrica. Ela renderiza os itens disponíveis para fabricação, dá suporte para o controle de fabricação de novas peças e de seleção das mesmas. A classe tem como parâmetros de instanciação um objeto da classe `Editor` e um da classe `Signals`. O objeto `Editor` é usado para inserir os itens existentes no painel fábrica (peças, lixeira e itens fabricados) na lista de objetos selecionáveis do editor. O objeto `Signals` controla o evento de redimensionamento do painel fábrica quando o tamanho da página do navegador é alterado. A propriedade `lixeira` guarda o objeto da classe `Lixeira` criado para renderização da lixeira. A propriedade `fabrica` armazena o objeto da classe `FabricaDeItens` instanciado para fabricação de novos itens. A propriedade `painelHeight` guarda a largura, em coordenadas gráficas, do painel fábrica. A propriedade `meshPainelFundoFabrica` guarda o objeto do `Three.js` que renderiza o fundo do painel fábrica. A propriedade `grupoItens` agrupa os objetos `Three.js` das peças existentes no painel e das peças fabricadas (até que elas sejam encaixadas em outra peça), permitindo que o editor localize e renderize essas peças.

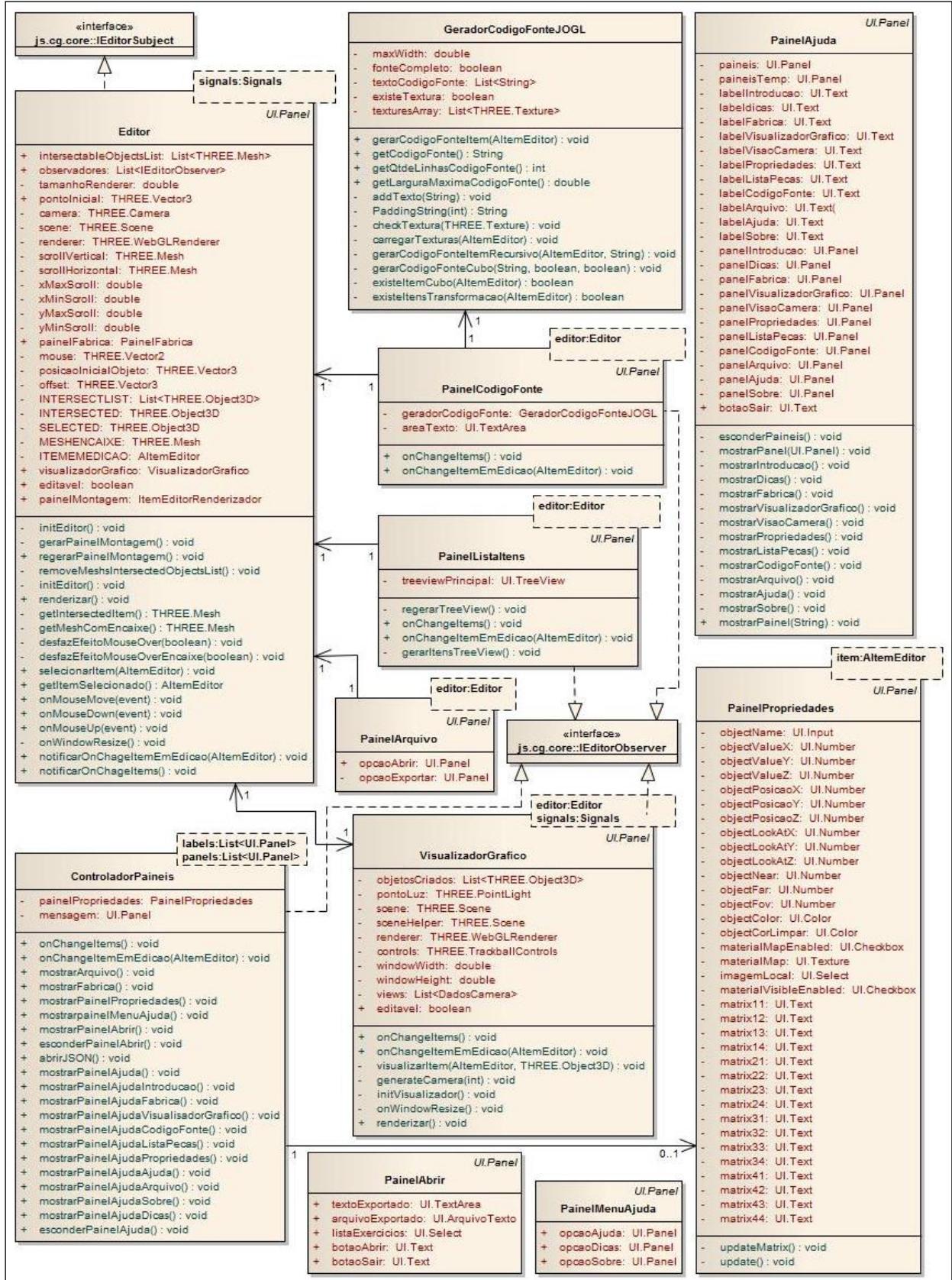
Na Figura 14 são apresentadas as classes do pacote `js.cg.panels`. Este pacote contém as classes que representam, controlam e organizam cada painel exibido no sistema.

A classe `PainelAjuda` cria o painel de ajuda do sistema, com as descrições de ajuda de cada painel. As propriedades `labelIntroducao`, `labelDicas`, `labelFabrica`, `labelVisualizadorGrafico`, `labelVisaoCamera`, `labelPropriedades`, `labelListaPecas`, `labelCodigoFonte`, `labelArquivo` e `labelAjuda` contém os objetos que exibem o título de ajuda de cada painel. As propriedades `panelIntroducao`, `panelDicas`, `panelFabrica`, `panelVisualizadorGrafico`, `panelVisaoCamera`, `panelPropriedades`, `panelListaPecas`, `panelCodigoFonte`, `panelArquivo`, `panelAjuda` e `panelSobre` guardam os objetos com o conteúdo de ajuda de cada título. As propriedades `paineis` e `paineisTemp` armazenam temporariamente os objetos com o conteúdo de ajuda que está sendo exibido, e são atualizadas conforme o usuário seleciona o título de ajuda desejada. A propriedade `botaoSair` guarda os objetos do botão de saída do painel.

A classe `PainelArquivo` cria o painel com o menu com as opções de manipulação de dados do sistema. Ela tem como parâmetro de instanciação um objeto da classe `Editor`, que é utilizado para buscar a estrutura de peças encaixadas ou carregar os exercícios. A propriedade `opcaoAbrir` armazena os objetos que exibem o *link* para a abertura do painel criado pela classe `PainelAbrir`. A propriedade `opcaoExportar` armazena os objetos que exibem o *link*

para a abertura de uma nova página no navegador contendo um texto com os dados do exercício.

Figura 14 – Pacote js.cg.panels



A classe `PainelAbrir` cria o painel que permite a abertura de algum dos exercícios da lista de exercícios do sistema ou a abertura de qualquer arquivo ou texto contendo exercícios exportados pelo sistema. A propriedade `arquivoExportado` contém o elemento HTML que permite ao usuário selecionar um arquivo texto com os dados do exercício a ser carregado no sistema. A propriedade `listaExercicios` contém o elemento HTML que exibe a lista de exercícios disponibilizada pelo sistema. A propriedade `textoExportado` contém o elemento HTML que exibe o texto do arquivo selecionado pelo objeto da propriedade `arquivoExportado` ou o texto do exercício selecionado no objeto da propriedade `listaExercicios`. A propriedade `botaoAbrir` contém os objetos do botão que confirma a abertura do texto. A propriedade `botaoSair` guarda os objetos do botão de saída do painel.

A classe `PainelMenuAjuda` cria o painel com o menu dos tipos de ajudas disponíveis. As propriedades `opcaoAjuda`, `opcaoDicas`, `opcaoSobre` guardam, respectivamente, os objetos que exibem *links* para os tipos de ajuda existentes: Ajuda, Dicas e Sobre.

A classe `PainelListaItens` cria o painel com a relação de peças encaixadas usando um *tree view*. Ela tem como parâmetro de instanciação um objeto da classe `Editor`, que é utilizado para buscar a estrutura de peças encaixadas e para se incluir na lista de observadores do objeto. A propriedade `treeviewPrincipal` guarda o objeto que internamente utiliza o framework `DDTreeMenu` para construir os elementos HTML que compõem o *tree view*.

A classe `VisualizadorGrafico` cria o painel que exibe o ambiente 3D com o resultado do exercício existente no editor (painel Espaço Gráfico) e com a visão que a peça Câmera tem deste mesmo ambiente (painel Visão da Câmera). O ambiente 3D é renderizado utilizando o WebGL. Ela tem como parâmetros de instanciação um objeto da classe `Editor` e um da classe `Signals`. A classe utiliza o objeto `Editor` para buscar as peças do exercício, gerando o resultado no ambiente 3D, e para se incluir na lista de observadores do objeto. O objeto `Signals` é usado para a criação do evento de redimensionamento do painel quando o tamanho da página do navegador é alterado. A propriedade `objetosCriados` guarda uma lista de todos os objetos do Three.js criados dentro do ambiente 3D.

A propriedade `renderer` da classe `VisualizadorGrafico` guarda o objeto do Three.js que renderiza a cena. A propriedade `controls` contém o objeto utilitário do Three.js que permite controlar a câmera do painel Espaço Gráfico via mouse. A propriedade `scene` contém a cena com os objetos do ambiente 3D que devem ser visíveis no resultado do exercício e na visão da peça Câmera. A propriedade `sceneHelper` contém a cena com os objetos que devem ser visíveis apenas no resultado do exercício e omitidos na visão da peça

Câmera. A propriedade `pontoLuz` guarda o objeto do Three.js que cria um ponto de luz no ambiente 3D. A propriedade `views` armazena uma lista com os dados de configuração da câmera que visualiza o resultado do exercício e da câmera que reproduz a visão da peça Câmera.

As propriedades `windowWidth` e `windowHeight` guardam a largura e altura, em *pixels*, do painel que é criado pela classe `VisualizadorGrafico`. A propriedade `editavel` é uma *flag* para identificar se o painel está acessível ao usuário. A propriedade `indica` se o sistema deve ou não atualizar a renderização do ambiente 3D do painel. O painel deixa de estar acessível quando ele é sobreposto por outro, tal como o formulário com os tutoriais de ajuda do sistema.

A classe `GeradorCodigoFonteJOGL` disponibiliza recursos para criar, a partir de uma peça do sistema, uma *string* contendo um código fonte na linguagem Java, utilizando a biblioteca JOGL, que irá reproduzir o conceito representado pela peça selecionada e seus filhos. A propriedade `maxWidth` guarda a largura, em *pixels*, da linha mais comprida do texto. A propriedade `existeTextura` indica se o item usado para geração do código fonte, ou um de seus filhos, possui uma textura habilitada para ele. A propriedade `texturesArray` contém a lista de texturas habilitadas encontradas no item selecionado e seus filhos. A propriedade `textoCodigoFonte` é uma lista de *strings* contendo o código fonte gerado. A propriedade `fonteCompleto` indica se o código JOGL foi gerado a partir do item `Renderizador`, o que resulta na geração do código fonte completo (contendo uma classe pronta para compilar no Java).

A classe `PainelCodigoFonte` cria o painel responsável por exibir o código fonte do item selecionado no editor. Ela tem como parâmetro de instanciação um objeto da classe `Editor`, que é utilizado para buscar o item selecionado e gerar o código fonte do mesmo, assim como para se incluir na lista de observadores do objeto. A propriedade `geradorCodigoFonte` guarda o objeto da classe `GeradorCodigoFonteJOGL` que é instanciado para gerar os códigos fonte e a propriedade `areaTexto` guarda o objeto HTML que exibe o código gerado.

A classe `PainelPropriedades` cria o painel que exibe as propriedades editáveis da peça selecionada do editor. Ela tem como parâmetro de instanciação um objeto da classe `AItemEditor`, que é utilizado para buscar as propriedades do item. Se o item passado nesse parâmetro for a peça `Renderizador`, a classe exibirá a propriedade `objectCorLimpar`, que guarda os objetos do painel que permitem editar a propriedade `corLimpar` do item.

Se o item passado no parâmetro de instanciação da classe PainelPropriedades for a peça Câmera, a classe exibirá as propriedades `objectName`, `objectPosicaoX`, `objectPosicaoY`, `objectPosicaoZ`, `objectLookAtX`, `objectLookAtY`, `objectLookAtZ`, `objectNear`, `objectFar` e `objectFov`. A propriedade `objectName` guarda os objetos do painel que permitem editar a propriedade nome do item. As propriedades `objectPosicaoX`, `objectPosicaoY` e `objectPosicaoZ` guardam os objetos do painel que permitem editar a propriedade posição do item. As propriedades `objectLookAtX`, `objectLookAtY` e `objectLookAtZ` guardam os objetos do painel que permitem editar a propriedade `lookAt` do item. A propriedade `objectNear` guarda os objetos do painel que permitem editar a propriedade `near` do item. A propriedade `objectFar` guarda os objetos do painel que permitem editar a propriedade `far` do item. A propriedade `objectFov` guarda os objetos do painel que permitem editar a propriedade `fov` do item.

Se o item passado no parâmetro de instanciação da classe PainelPropriedades for uma peça do tipo Objeto Gráfico, a classe exibirá as propriedades `objectName`, `materialVisibleEnabled`, `matrix11`, `matrix12`, `matrix13`, `matrix14`, `matrix21`, `matrix22`, `matrix23`, `matrix24`, `matrix31`, `matrix32`, `matrix33`, `matrix34`, `matrix41`, `matrix42`, `matrix43` e `matrix44`. A propriedade `objectName` guarda os objetos do painel que permitem editar a propriedade nome do item. A propriedade `materialVisibleEnabled` guarda os objetos do painel que permitem editar a propriedade `visible` do item. As propriedades `matrix11`, `matrix12`, `matrix13`, `matrix14`, `matrix21`, `matrix22`, `matrix23`, `matrix24`, `matrix31`, `matrix32`, `matrix33`, `matrix34`, `matrix41`, `matrix42`, `matrix43` e `matrix44` guardam os objetos do painel que permitem visualizar a propriedade `matrix` do item.

Se o item passado no parâmetro de instanciação da classe PainelPropriedades for uma peça do tipo Cubo, a classe exibirá as propriedades `objectName`, `objectValueX`, `objectValueY`, `objectValueZ`, `objectPosicaoX`, `objectPosicaoY`, `objectPosicaoZ`, `objectColor`, `materialMapEnabled`, `materialMap`, `imagemLocal` e `materialVisibleEnabled`. A propriedade `objectName` guarda os objetos do painel que permitem editar a propriedade nome do item. As propriedades `objectValueX`, `objectValueY` e `objectValueZ` guardam os objetos do painel que permitem editar a propriedade `valorXYZ` do item. As propriedades `objectPosicaoX`, `objectPosicaoY` e `objectPosicaoZ` guardam os objetos do painel que permitem editar a propriedade `posicao` do item. A propriedade `objectColor` guarda os objetos do painel que permitem editar a

propriedade `propriedadeCor` do item. A propriedade `materialMapEnabled` guarda os objetos do painel que permitem editar a propriedade `usarTextura` do item. A propriedade `materialMapEnabled` guarda os objetos do painel que permitem editar a propriedade `usarTextura` do item. As propriedades `materialMap` e `imagemLocal` guardam os objetos do painel que permitem editar a propriedade `textura` do item. A propriedade `materialVisibleEnabled` guarda os objetos do painel que permitem editar a propriedade `visible` do item.

Se o item passado no parâmetro de instanciação da classe `PainelPropriedades` for uma peça do tipo `Transladar`, `Rotacionar` ou `Escalar`, a classe exibirá as propriedades `objectName`, `objectValueX`, `objectValueY`, `objectValueZ` e `materialVisibleEnabled`. A propriedade `objectName` guarda os objetos do painel que permitem editar a propriedade `nome` do item. As propriedades `objectValueX`, `objectValueY` e `objectValueZ` guardam os objetos do painel que permitem editar a propriedade `valorXYZ` do item. A propriedade `materialVisibleEnabled` guarda os objetos do painel que permitem editar a propriedade `visible` do item.

A classe `Editor` é responsável por criar o painel Fábrica de Peças e seus subpainéis, que são renderizados usando o WebGL. Ela também é responsável pelo controle da renderização, criação, seleção, movimentação e exclusão dos itens do editor. Conforme foi apresentado nas classes anteriores, a classe também centraliza o controle de atualização dos outros painéis, notificando os mesmos quando um exercício é aberto ou alterado. Ela tem como parâmetro de instanciação um objeto da classe `Signals`, que é usado para a criação do evento de redimensionamento do painel quando o tamanho da página do navegador é alterado.

A propriedade `intersectableObjectsList` da classe `Editor` é uma lista de objetos Three.js que contém todos os objetos selecionáveis dentro da Fábrica de Peças. Ela é carregada com os objetos contidos na propriedade `insertectableMeshs` dos objetos do tipo `AObjetoGrafico` existentes no editor e com outros objetos auxiliares do editor, tais como os objetos Three.js que renderizam as barras de rolagem. Essa lista possibilita que a função de seleção localize e selecione esses objetos. A propriedade `observadores` contém a lista de todos os objetos que serão notificados quando um exercício for aberto ou alterado. A propriedade `visualizadorGrafico` mantém um vínculo do editor com o painel Espaço Gráfico, permitindo que o sistema exporte, junto com os dados do exercício, alguns dados do painel, como a posição da câmera.

Na classe `Editor` a propriedade `renderer` guarda o objeto do `Three.js` que renderiza a cena. A propriedade `scene` contém a cena com os objetos do `Three.js` que irão renderizar a Fábrica de Peças. A propriedade `camera` contém o objeto `Three.js` que configura a câmera que visualiza o ambiente renderizado. A propriedade `painelFabrica` guarda uma referência a instância do objeto `PainelFabrica`. Este objeto também é incluído no objeto da propriedade `scene`, habilitando a visualização do subpainele `fábrica`. A propriedade `painelMontagem` contém o objeto `ItemEditorRenderizador` que renderiza a peça `Renderizador`. As propriedades `scrollVertical` e `scrollHorizontal` guardam os objetos do `Three.js` que renderizam as barras de rolagem vertical e horizontal.

A propriedade `pontoInicial` da classe `Editor` contém a coordenada gráfica que determina o ponto inicial que é utilizado para renderizar os subpaineis `Fábrica` e `painel` de `montagem`. Essa coordenada é modificada conforme o tamanho da janela do navegador é alterado.

As propriedades `mouse`, `posicaoInicialObjeto` e `offset` da classe `Editor` são usadas para controlar o ponto de origem de *click* do mouse e da peça selecionada, e do ponto atual do movimento sendo realizado. Assim é possível calcular e converter a diferença entre esses pontos para coordenadas gráficas, possibilitando a seleção e movimentação das peças. As propriedades `xMinScroll`, `xMaxScroll`, `yMinScroll` e `yMaxScroll` definem as coordenadas gráficas limites para translação das barras de rolagem, impedindo que elas não fujam do campo de visão do usuário.

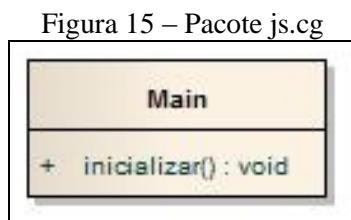
Para auxiliar no controle de seleção da classe `Editor`, a propriedade `INTERSECTLIST` guarda uma lista temporária de todos os objetos selecionáveis que estão abaixo do ponteiro do mouse. Essa lista é atualizada conforme o mouse se movimenta. A propriedade `SELECTED` guarda o objeto `Three.js` do item que está sendo arrastado. Quando nenhum objeto está sendo arrastado, a propriedade `INTERSECTED` é carregada com o primeiro item da lista disponível em `INTERSECTLIST`, permitindo os efeitos mouse *over*, como a mudança de cor das barras de rolagem e mudança do cursor sobre as peças. A propriedade `MESHENCAIXE` é carregada com o primeiro item da lista disponível em `INTERSECTLIST` quando existe um encaixe compatível sob a peça sendo arrastada, possibilitando ao editor encaixar e excluir peças. A propriedade `ITEMEMEDICAO` guarda o item selecionado para edição.

A propriedade `tamanhoRenderer` da classe `Editor` armazena o tamanho (largura e altura, que serão sempre iguais) do elemento HTML criado pelo objeto da propriedade

`renderer`. A propriedade `editavel` é uma *flag* para identificar se o painel está acessível ao usuário, indicando se o sistema deve ou não atualizar a renderização do painel.

A classe `ControladorPaineis` é responsável pelo controle de exibição e ocultação dos painéis existentes no sistema. Ela define todos os eventos *on click* envolvendo exibição e ocultação de painéis. Assim, qualquer botão existente no sistema que exiba ou esconda um painel, tem seu comportamento controlado e centralizado nessa classe. Ela tem como parâmetros de instânciação uma lista de *labels* e uma lista de painéis. A lista de *labels* deve conter o conjunto de objetos que exibem os títulos dos painéis existentes na abertura do sistema. A lista de painéis contém a lista de todos os painéis existentes no sistema. A propriedade `painelPropriedades` contém uma referência ao objeto `PainelPropriedades` que é novamente instanciado cada vez que um item é selecionado. A propriedade `mensagem` guarda o objeto que exibe uma mensagem de aviso no painel `Propriedades` da Peça indicando que nenhum item está selecionado.

Na Figura 15 pode-se visualizar a classe `Main` do pacote `js.cg`. Essa classe é a única classe do pacote e é responsável por gerar as instâncias das classes do pacote `js.cg.panels` e assim inicializar o sistema. A classe `Main` possui apenas a função `inicializar()` que cria as instâncias de todos os painéis, a instância da classe `ControladorPaineis`, a instância do objeto da biblioteca `Signals` e os eventos necessários para a execução do aplicativo. A classe implementa o padrão de projeto *Singleton*. No código fonte da página HTML, depois de carregar todas as classes do sistema, deve-se escrever apenas o código `Main.inicializar()` para carregar o sistema.



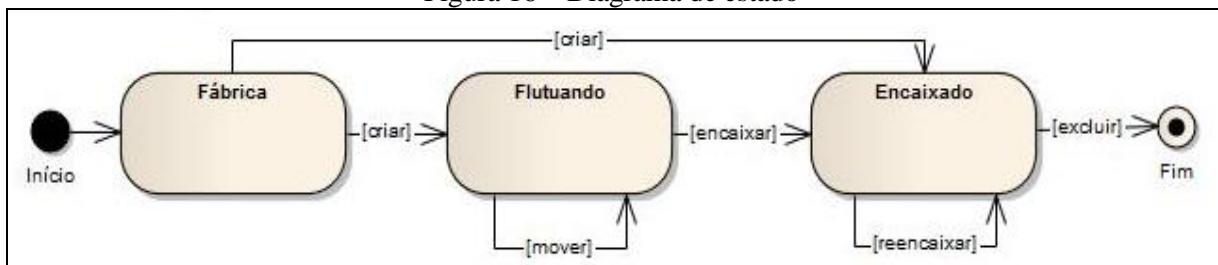
3.2.3 Diagrama de estado

O diagrama de estados da Figura 16 mostra os estados que representam o ciclo de vida das peças disponíveis no painel `Fábrica de Peças`.

Conforme diagrama, existem três estados para uma peça. O primeiro estado é a peça ainda fazer parte da fábrica e a partir dela ser criada outra peça e mover a mesma para uma posição qualquer no painel de montagem (sem encaixá-la), fazendo-a entrar no segundo estado, ou encaixá-la, que fará ela entrar no terceiro estado. Uma vez no segundo estado, é

possível mover a peça para outra posição qualquer dentro do painel (sem encaixá-la) e não alterar seu estado. Contudo, se uma peça no segundo estado for encaixada em alguma peça, ela irá permanecer no terceiro estado, visto que peças no terceiro estado só podem ser encaixadas em outras peças ou serem excluídas.

Figura 16 – Diagrama de estado



3.3 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas no desenvolvimento da aplicação. Logo após segue-se com uma descrição da operacionalidade da implementação com a exibição de algumas *interfaces*.

3.3.1 Técnicas e ferramentas utilizadas

As linguagens de programação HTML5, CSS 3.0 e JavaScript foram utilizadas para implementar a aplicação. Para o desenvolvimento no JavaScript, foram usadas as bibliotecas jQuery v2.1.1, Three.js v67, JSCOLOR 1.4.1 (JSCOLOR, 2008) e DDTreeMenu (DYNAMICDRIVE, 2012). Também foram utilizadas algumas bibliotecas auxiliares do Three.js que serão comentadas adiante. O Notepad++ v6.5.5 foi utilizado como ambiente de desenvolvimento e, como ambientes de testes, foram utilizados os navegadores Chrome versão 34.0.1847.131 m, Firefox versão 29.0.1 e Internet Explorer versão 11.0.9600.17107.

Para a construção do exercício, foi mantido o conceito de fabricação e encaixe de peças já existente no AduboGL. Contudo, era necessária uma mudança no comportamento dos encaixes que os tornasse mais intuitivos. Com base no jogo infantil de encaixe de formas geométricas, demonstrado no APÊNDICE A, criou-se uma estrutura mais lúdica para o encaixe das peças do exercício.

Para cada tipo de peça existente foi criado um recorte de uma forma geométrica simples, possibilitando que o usuário identifique o local correto do encaixe de cada peça e monte o exercício de forma coerente. O recorte de cada peça também pode ser observado no APÊNDICE A. Cada peça representa um conceito básico de computação gráfica e realiza uma interação com o resultado 3D que é gerado pelo encaixe e pelas propriedades da peça. Desta forma, o painel Fábrica de Peças disponibiliza um jogo de encaixe de peças que possibilita

ao usuário construir um ambiente 3D usando os conceitos de computação gráfica representados por cada peça.

O Quadro 13 apresenta o trecho de código fonte da classe `Editor` responsável por criar as instâncias necessárias para renderização da Fábrica de Peças.

Quadro 13 – Função de inicialização do ambiente gráfico da classe `Editor`

```

1  function initEditor() {
2      // SCENES
3      scene = new THREE.Scene();
4      // CAMERAS
5      camera = new THREE.PerspectiveCamera( 50, 1, 1, 5500 );
6      // LIGHTS
7      var skyColor = 0xFFFFFF;
8      var groundColor = 0xFFFFFF;
9      var intensity = 1;
10     var light = new THREE.HemisphereLight( skyColor, groundColor,
11         intensity );
12     light.position.set( 1, 1, 1 ).multiplyScalar( 200 );
13     scene.add( light );
14     // SCROLLS
15     //cria scroll Vertical
16     var points = [];
17     points.push( new THREE.Vector2 ( 0, 0 ) );
18     points.push( new THREE.Vector2 ( 5, 0 ) );
19     points.push( new THREE.Vector2 ( 5, 1 ) );
20     points.push( new THREE.Vector2 ( 0, 1 ) );
21     var squareShape = new THREE.Shape( points );
22     scrollVertical = CG.objects.generateMeshFromShape( squareShape,
23         CG.colors.corScroll );
24     scrollVertical.material.opacity = opacidadeScrolls;
25     scrollVertical.material.transparent = true;
26     scene.add( scrollVertical );
27     scope.intersectableObjectsList.push( scrollVertical );
28     //cria scroll Horizontal
29     var points = [];
30     points.push( new THREE.Vector2 ( 0, 0 ) );
31     points.push( new THREE.Vector2 ( 1, 0 ) );
32     points.push( new THREE.Vector2 ( 1, 5 ) );
33     points.push( new THREE.Vector2 ( 0, 5 ) );
34     var squareShape = new THREE.Shape( points );
35     scrollHorizontal = CG.objects.generateMeshFromShape( squareShape,
36         CG.colors.corScroll );
37     scrollHorizontal.material.opacity = opacidadeScrolls;
38     scrollHorizontal.material.transparent = true;
39     scene.add( scrollHorizontal );
40     scope.intersectableObjectsList.push( scrollHorizontal );
41     // PAINEL FÁBRICA
42     scene.add( scope.painelFabrica );
43     // RENDERER
44     renderer = new THREE.WebGLRenderer( { antialias: true,
45         clearAlpha: 1 } );
46     renderer.sortObjects = false;
47     renderer.shadowMapEnabled = true;
48     renderer.shadowMapType = THREE.PCFShadowMap;
49     renderer.setSize( tamanhoRender, tamanhoRender );
50     renderer.setClearColor( 0xFFFFFF );
51     scope.dom.appendChild( renderer.domElement );
52 }
```

O código apresenta os recursos básicos do Three.js utilizados para criar uma cena: a cena, uma câmera, um ponto de luz, os objetos gráficos e o renderizador. No código pode-se observar que os elementos mais simples de criar são a cena, na linha 3, e a câmera, na linha 5.

Existem vários tipos de ponto de luz disponibilizados pelo Three.js, cada um possuindo sua relação de atributos. A partir da linha 6 pode-se observar alguns dos atributos de uma luz hemisférica, que é utilizada na renderização do editor. Entre as linhas 14 e 40 está o código que cria os objetos que desenham as barras de rolagem do painel de montagem e na linha 42 é incluído a instância da classe PainelFabrica que contém todos os objetos responsáveis por renderizar o painel fábrica. Na linha 44 é criada a instância do renderizador, que gera o elemento HTML onde é renderizada a cena. Na linha 51 pode-se visualizar o momento em que o elemento HTML do renderizador é associado ao elemento HTML do editor.

No Quadro 14 são apresentadas as funções que renderizam todos os painéis com cenas gráficas. A função `renderizar()` da classe `Editor` é responsável por renderizar o conteúdo do painel Fábrica de Peças.

A função `renderizar()` da classe `VisualizadorGrafico` é responsável pela renderização do conteúdo dos painéis Espaço Gráfico e Visão da Câmera. Estes dois painéis são o mesmo painel (mesmo elemento HTML), que, conforme pode-se visualizar no quadro, tem sua renderização dividida em dois, uma para cada câmera existente no painel. Nele existe uma câmera fixa para o painel Espaço Gráfico e uma câmera personalizável, para o painel Visão da Câmera, que é configurada conforme as propriedades da peça Câmera encaixada no painel de montagem.

O trecho de código exibido da classe `Main` é responsável por atualizar continuamente a renderização das cenas do sistema. A função `requestAnimationFrame()` é uma função auxiliar no Three.js que cria uma *thread* que executa a função passada a ela por parâmetro. Essa função é chamada dentro da função `animador()`, para que sempre que a função `animador()` seja executada, seja criada uma nova *thread* que execute novamente a função `animador()`. Esse comportamento garante que sempre existirá uma *thread* renderizando as cenas.

Sempre que o usuário modifica o tamanho da janela, o tamanho dos elementos HTML criados pelas classes `THREE.WebGLRenderer` do Three.js precisam ser ajustados conforme o novo tamanho dos painéis. Para fazer o controle dos eventos que ajustam o tamanho desses elementos, foi utilizada a biblioteca `Signals`. No Quadro 15 pode-se visualizar exemplos de códigos que realizam o redimensionamento do elemento HTML onde são renderizados os painéis Espaço Gráfico e Visão da Câmera. A função `onWindowResize()` da classe `VisualizadorGrafico` informa o novo tamanho do painel para o objeto da propriedade

`renderer` dessa mesma classe. Esse objeto internamente atualiza o tamanho do elemento HTML onde a cena é renderizada.

Quadro 14 – Trechos de código que renderizam as cenas da aplicação

Classe	Código
Editor	<pre>scope.renderizar = function () { renderer.render(scene, camera); };</pre>
VisualizadorGrafico	<pre>scope.renderizar = function () { //percorre as câmeras do painél for (var i = 0; i < views.length; ++i) { var view = views[i]; var camera = view.camera; var left = Math.floor(windowWidth*view.left); var bottom = Math.floor(windowHeight*view.bottom); var width = Math.floor(windowWidth*view.width); var height = Math.floor(windowHeight*view.height); //seta a posição no painél onde será desenhada a //visão da câmera atual renderer.setViewport(left, bottom, width, height); renderer.setScissor(left, bottom, width, height); renderer.enableScissorTest (true); //set cor de limpeza definida para a câmera atual if (view.clearColorCamera !== undefined) { renderer.setClearColor(view.clearColorCamera, view.background.a); } else { renderer.setClearColor(view.background, view.background.a); } camera.aspect = width / height; camera.updateProjectionMatrix(); //se existe uma peça Câmera associada a esta //câmera (encaixada no painel de montagem) if (view.cameraHelper) { //remove objetos auxiliares da cena scene.remove(sceneHelper); //renderiza cena renderer.render(scene, camera); //recoloca objetos auxiliares da cena scene.add(sceneHelper); } else { renderer.render(scene, camera); } } controls.update(); };</pre>
Main	<pre>function animador() { //função do Three.js que cria uma thread que executa //novamente a função animador() requestAnimationFrame(animador); TWEEN.update(); if (paineis.editor.editavel) { paineis.editor.renderizar(); } if (paineis.visualizadorGrafico.editavel) { paineis.visualizadorGrafico.renderizar(); } } animador(); //faz chamada da função pela primeira vez</pre>

Na primeira linha do código exibido da classe `VisualizadorGrafico` pode-se visualizar o trecho em que a função `onWindowResize()` é associada ao objeto da biblioteca `Signals`. Existe apenas uma instância do objeto da biblioteca `Signals` no sistema, que é utilizada pelas demais classes. A função `onWindowResize()` da classe `Main` executa todos os eventos armazenados no objeto da biblioteca `Signals`, inclusive o evento `onWindowResize()` da classe `VisualizadorGrafico`.

Quadro 15 – Trechos de código que ajustam o tamanho do conteúdo dos painéis Espaço Gráfico e Visão da Câmera

Classe	Código
<code>VisualizadorGrafico</code>	<pre>signals>windowResize.add(function (object) { onWindowResize(); }); ... function onWindowResize() { if (windowHeight != scope.dom.offsetWidth windowHeight != scope.dom.offsetHeight) { windowHeight = scope.dom.offsetWidth; windowHeight = scope.dom.offsetHeight; renderer.setSize(windowWidth, windowHeight); } }</pre>
<code>Main</code>	<pre>function onWindowResize(event) { signals>windowResize.dispatch(); } onWindowResize(null); window.addEventListener('resize', onWindowResize, false);</pre>

A biblioteca UI, biblioteca auxiliar do Three.js, foi utilizada para facilitar o controle e criação dos componentes HTML da aplicação. Essa biblioteca disponibiliza uma classe específica para cada tipo de componente HTML manipulado dinamicamente no sistema, permitindo que as propriedades desses componentes possam ser modificadas usando as propriedades e funções dessas classes.

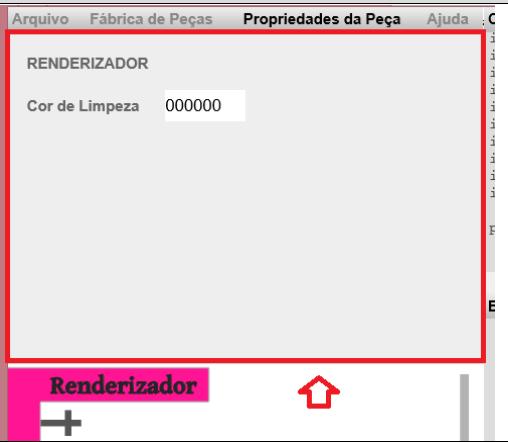
Na Figura 17 são apresentadas algumas classes da biblioteca UI e o tipo de componente HTML manipulados por elas. Como pode-se observar, algumas classes, como as classes `UI.Number` e `UI.Color`, criam o mesmo tipo de elemento HTML, `input`. Contudo, elas possuem definem configurações diferentes para esse elemento. Por exemplo, a classe `UI.Color` cria um elemento `input` associado a uma paleta de cores, já a classe `UI.Number` cria um elemento `input` que quando clicado, ao arrastar o mouse, seu valor é incrementado/decrementado.

Cada classe dessa biblioteca trata a informação de forma particular, adaptando o componente ao tipo de informação ou recurso que se deseja manipular. Na biblioteca foram incluídas as classes `UI.TreeView`, `UI.ItemTreeView`, `UI.Imagem` e `UI.ArquivoTexto`, que auxiliam no controle de elementos HTML exclusivos do sistema, como a lista em forma de

árvore do painel `Lista de Peças` e o elemento `input` que permite carregar um arquivo texto no painel `Abrir`. Várias classes nativas da biblioteca UI, como as classes `UI.Number`, `UI.Color`, `UI.Select` e `UI.Texture`, foram ajustadas para atender necessidades do sistema, inclusive a classe `UI.Element`, que é a classe base para todas as classes existentes na biblioteca UI.

Para facilitar a seleção de cores das propriedades de cor, foi utilizado o *framework* JSColor, que exibe uma palheta de cores para essas propriedades. O JSColor foi criado para que na abertura de uma página HTML, a instância da classe do *framework* fosse inicializada. Nessa inicialização o framework percorria todos os elementos `input` da página associados a classe CSS `color`, e relacionava esses elementos aos recursos do *framework*. Contudo, todos os elementos `input` existentes no VisEdu-CG, que editam as propriedades das peças, são criados dinamicamente. Isso impedia que o *framework* localizasse esses elementos quando era inicializado. Assim, foi criada a função `setColorElement()` dentro do *framework*, para ser possível definir manualmente os elementos que seriam associados a ele.

Figura 17 – Relação entre alguns elementos HTML e classes da biblioteca UI

Classe UI (em destaque) com ex. de cód.	Elemento	Exemplo visual
<pre>function PainelPropriedades(item) { UI.Panel.call(this); var scope = this; scope.item = item; scope.setPosition('absolute'); scope.setDisplay('block'); ... }</pre>	DIV	
<code>objectValueX = new UI.Number().setWidth('50px').onChange(update);</code>	INPUT	
<code>objectValueRow.add(new UI.Text('x:').setColor('#666'));</code>	SPAN	
<code>objectColor = new UI.Color().onChange(update);</code>	INPUT	

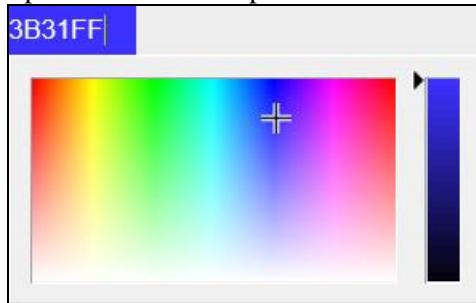
No Quadro 16 é exibido o trecho de código que faz uso do JSColor e associa o mesmo a um elemento `input`. Sempre que é necessário criar um campo para edição de uma propriedade de cor, o sistema usa a classe `UI.Color`, da biblioteca UI, que internamente utiliza o JSColor. Na Figura 18 pode-se visualizar um exemplo de um elemento `input`

associado ao *framework* JSColor e a palheta que é disponibilizada para o elemento quando o usuário edita o mesmo.

Quadro 16 – Trecho de código que utiliza o framework JSColor

```
UI.Color = function () {
    ...
    var dom = document.createElement( 'input' );
    ...
    dom.className = 'Color';
    JSColor.setColorElement( dom ); //habilita paleta de cores
    ...
}
```

Figura 18 – Exemplo de elemento input associado ao framework JSColor



Para a criação do painel *Lista de Peças*, optou-se por utilizar um *tree view*, devido a estrutura em forma de árvore que é produzida pelo encaixe das peças no painel de montagem. Para construir esse menu em forma de árvore, foi utilizada o *framework* DDTreeMenu.

O DDTreeMenu, semelhante ao *framework* JSColor, não estava apto para trabalhar com componentes dinâmicos. Ele possuía apenas uma função de inicialização que sempre percorria toda a estrutura HTML, procurando elementos `ul` associados a classe CSS `treeview`. Quando encontrava um elemento nessas condições, a biblioteca formatava esse elemento e seus filhos, aplicando os eventos de expansão e retração dos nós, assim como as imagens representativas, indicando se um nó está aberto ou fechado. Como a geração do menu do painel *Lista de Peças* é feita de forma dinâmica, foi criada nesse *framework* as funções `createTreeFromElement()` e `flattenFromElement()`, que permitem associar dinamicamente um elemento `ul` sem a necessidade de varrer toda a estrutura HTML da página.

As imagens utilizadas pelo *framework* que representavam se um nó estava aberto ou fechado também foram alteradas. Foram substituídos os ícones de uma pasta aberta e fechada por setas, semelhantes aos ícones utilizados no *tree view* de exploração de pastas do sistema operacional Windows 7. Na Figura 19 pode-se visualizar as setas que indicam se o nó está aberto ou fechado. No Quadro 17 é apresentado o trecho de código que faz uso do *framework*. As classes `UI.TreeView` e `UI.ItemTreeView`, da biblioteca UI, são utilizadas para auxiliar na

criação dos elementos `ul` e `li` que serão formatados pela biblioteca DDTreeMenu. No Quadro 18 são apresentados os métodos de construção dessas duas classes.

Figura 19– Setas que indicam se o nó está aberto ou fechado



Quadro 17 – Trecho de código da classe PainelListaItens

```
treeviewPrincipal = new UI.TreeView();
treeviewPrincipal.setId();

//exibe tree view
scope.add(treeviewPrincipal);

//preenche tree view
gerarItensTreeView( scope.editor.painelMontagem, treeviewPrincipal );

//formata tree view
DDTreeMenu.createTreeFromElement( treeviewPrincipal.dom, false );
```

Quadro 18 – Métodos de construção das classes UI.TreeView e UI.ItemTreeView

```
UI.TreeView = function () {
    UI.Element.call( this );

    var scope = this;
    var dom = document.createElement('ul');
    dom.className = 'treeview';
    dom.setAttribute('rel', 'open');
    this.dom = dom;
    return this;
};

...

UI.ItemTreeView = function (treeView) {
    UI.Element.call( this );

    var scope = this;
    if (!(treeView instanceof UI.TreeView)) {
        throw new Error('argumento deve ser da classe UI.TreeView !');
    }
    this.treeView = treeView;
    var dom = document.createElement('li');
    this.dom = dom;
    this.treeView.add( this );
    return this;
};
```

Além da biblioteca UI, o Three.js disponibiliza outras bibliotecas auxiliares que foram utilizadas na construção do VisEdu-CG. Entre elas, 3 precisaram ser adaptadas: Detector, THREE.CameraHelper e THREE.TrackballControls. Para não gerar conflito com as bibliotecas originais, elas tiveram seus nomes alterados. No Quadro 19 pode-se visualizar um exemplo de uso das 3 bibliotecas e o novo nome dado a cada uma.

Quadro 19 – Uso das bibliotecas Detector, THREE.CameraHelper e THREE.TrackballControls no VisEdu-CG

Nome original	Novo nome	Exemplo de código
Detector	CGDetector	// verifica se é possível utilizar o // HTML5 if (! CGDetector.webgl) { CGDetector.addGetWebGLMessage(); } else { ... }
THREE.CameraHelper	CGCameraHelper	if (!viewCamera.cameraHelper) { viewCamera.cameraHelper = new CGCameraHelper(viewCamera.camera); sceneHelper.add(viewCamera.cameraHelper); }
THREE.TrackballControls	CGTrackballControls	controls = new CGTrackballControls(views[0].camera, scope.dom); controls.rotateSpeed = 1.0; controls.zoomSpeed = 1.2; controls.panSpeed = 0.8; controls.noRotate = false ; controls.noZoom = false ; controls.noPan = false ; controls.staticMoving = false ; controls.dynamicDampingFactor = 0.3; controls.enabled = true ;

A biblioteca Detector é utilizada para verificar se o navegador utilizado suporta o uso de WebGL, se não suportar, ela exibe uma mensagem de aviso na página. As mensagens de aviso da biblioteca foram alteradas para a língua portuguesa.

A biblioteca THREE.CameraHelper desenha a pirâmide existente no painel Espaço Gráfico, que facilita a identificação dos limites enxergados pela peça Câmera. A biblioteca foi alterada para, além da pirâmide que já existia, desenhar também os planos transparentes que possibilitam identificar mais facilmente o *near* e o *far* definidos para esta peça.

Foi necessário ajustar a biblioteca THREE.TrackballControls devido a uma incompatibilidade dela com a versão da biblioteca UI utilizada. Quando a biblioteca UI foi adaptada para o sistema, era utilizada a versão 56 do Three.js. Na conversão do sistema para a versão 67 do Three.js, era inviável atualizar a biblioteca UI, devido à quantidade de modificações que foram efetuadas na biblioteca. Contudo, isso gerou um conflito com a biblioteca THREE.TrackballControls e optou-se por ajustar essa biblioteca, no lugar de readaptar a biblioteca UI novamente.

Duas das bibliotecas auxiliares utilizadas do Three.js que não precisaram ser adaptadas, foram as bibliotecas Tween e Gentilis_bold.typeface. No Quadro 20 pode-se visualizar um exemplo de uso da biblioteca Tween. Esta biblioteca foi utilizada para reproduzir o efeito de movimento elástico que é feito quando as peças são encaixadas. Conforme é apresentado no quadro, ao usar a biblioteca Tween é necessário apontar o objeto

que armazena a posição do objeto que será movimentado, a nova posição desse objeto, e o tempo, em milissegundos, que o movimento deve durar. A biblioteca Gentilis_bold.typeface disponibiliza ao Three.js os dados necessários para criar textos tridimensionais utilizando a fonte gentilis com estilo *bold*. Ao ser carregada na página, a biblioteca automaticamente se associa com as fontes disponíveis para o Three.js.

Quadro 20 – Trecho de código da classe AObjetoGrafico

```
//repositiona encaixe diamante
new TWEEN.Tween( scope.objetoDiamante.position ).to( {
    x: scope.objetoDiamante.position.x,
    y: novoY,
    z: scope.objetoDiamante.position.z },
    1000 ).easing( TWEEN.Easing.Elastic.Out ).start();
```

O sistema permite exportar os dados do exercício sendo manipulado gerando uma *string* em formato JSON. Esse texto pode ser salvo e posteriormente aberto, permitindo continuar a edição de exercícios. O formato JSON foi escolhido por ser uma estrutura amplamente difundida e com excelente legibilidade, possibilitando eventuais alterações do exercício diretamente na *string*. No Quadro 21 é apresentado um exemplo de *string* gerada para um exercício com apenas uma peça Câmera encaixada, sem nenhuma propriedade alterada.

Quadro 21 – Exemplo de texto gerado com os dados de um exercício

```
{
    "metadata": {
        "formatVersion": 1,
        "type": "VisEduCG",
        "generatedBy": "ExportadorJSON",
        "objects": 2,
        "textures": 0
    },
    "urlBaseType": "relativeToVisEduCG",
    "viewPos": [0, 600, 1800],
    "viewRot": [-0.3217505484695941, 0, 0],
    "itens": {
        "Item_7": {
            "nome": "Renderizador",
            "visible": true,
            "corLimp": 0,
            "tipo": 6,
            "filhos": {
                "Item_8": {
                    "nome": "Camera 1",
                    "visible": true,
                    "valorXYZ": [300, 300, 300],
                    "lookAt": [0, 0, 0],
                    "near": 100,
                    "far": 500,
                    "fov": 45,
                    "tipo": 0
                }
            }
        }
    },
    "textures": {}
}
```

Conforme foi descrito na seção 0, três padrões de projeto foram utilizados no sistema: *Factory*, *Observer* e *Singleton*. No Quadro 22 é exibida a função `fabricarNovoItem()` da classe `FabricaDeItens` que implementa o padrão *Factory* no sistema. Ela retorna o item desejado, de acordo com o tipo de item passado no parâmetro `idItem`.

Quadro 22 – Função `fabricarNovoItem()` da classe `FabricaDeItens`

```
scope.fabricarNovoItem = function (idItem, inserirNaLista) {
    var item = null;
    if      ( idItem == EIIdsItens.CAMERA ) {
        item = new ItemEditorCamera();
    } else if ( idItem == EIIdsItens.OBJETOGRAFICO ) {
        item = new ItemEditorObjetoGrafico();
    } else if ( idItem == EIIdsItens.CUBO ) {
        item = new ItemEditorCubo();
    } else if ( idItem == EIIdsItens.TRANSLADAR ) {
        item = new ItemEditorTransladar();
    } else if ( idItem == EIIdsItens.ROTACIONAR ) {
        item = new ItemEditorRotacionar();
    } else if ( idItem == EIIdsItens.REDIMENSIONAR ) {
        item = new ItemEditorEscalar();
    } else {
        throw new Error ("Nao foi possível fabricar o item. Id informado não existe!");
    }
    var nome = item.id.descricao;
    if      ( !inserirNaLista ) {
        item.id.count++;
        nome += " " + item.id.count;
    }
    item.setNome( nome );
    item.esconderDetalhes();
    if      ( ( inserirNaLista !== undefined ) && (inserirNaLista) ) {
        scope.itensFabricados.push(item);
    }
    return item;
};
```

No Quadro 23 é exibido um exemplo da implementação do padrão de projeto *Observer*. O padrão foi implementado para trabalhar com as atualizações das peças controladas pela classe `Editor`, sendo ela o objeto de interesse (*subject*). Várias classes, como a classe `VisualizadorGrafico`, observam a classe `Editor`, atualizando seu conteúdo quando notificadas de alguma alteração.

Quadro 23 – Exemplos de implementação do padrão de projeto *Observer*

Classe	Trecho de código
<code>Editor</code>	<pre>IEditorSubject.call(this); //interface //@Override scope.notificarOnChageItemEmEdicao = function (item) { for (var i = 0; i < scope.observadores.length; i++) { scope.observadores[i].onChangeItemEmEdicao(item); } }; //@Override scope.notificarOnChageItems = function () { for (var i = 0; i < scope.observadores.length; i++) { scope.observadores[i].onChangeItems(); } };</pre>

continua

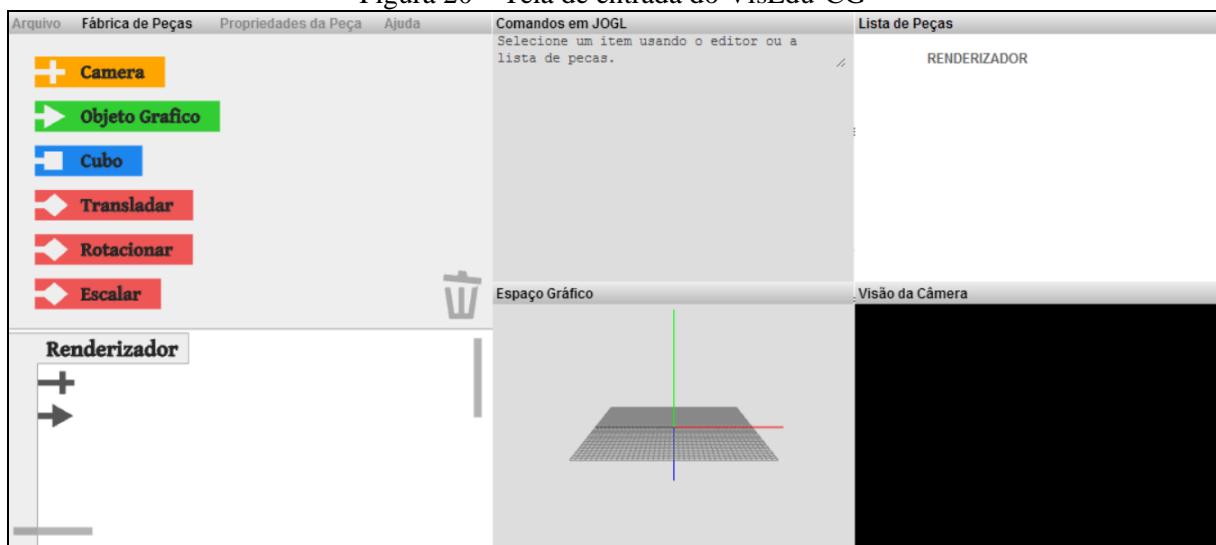
continuação

VisualizadorGrafico	<pre>IEditorObserver.call(this); //interface //@Override scope.onChangeItems = function () { //limpa objetos adicionados ao scene do visualizador for (var i=0; i < objetosCriados.length; i++) { scene.remove(objetosCriados[i]); } objetosCriados = []; //limpa objetos helpers for (var i=0; i < views.length; i++) { if (views[0].cameraHelper !== undefined && views[0].cameraHelper) { sceneHelper.remove(views[0].cameraHelper); views[0].cameraHelper = undefined; } } visualizarItem(scope.editor.painelMontagem, scene); }; //@Override scope.onChangeItemEmEdicao = function(item) { /* vazio */ };</pre>
---------------------	---

3.3.2 Operacionalidade da implementação

O VisEdu-CG possui dez painéis diferentes que são exibidos para o usuário. Os painéis Fábrica de Peças, Comandos em JOGL, Lista de Peças, Visualizador Gráfico e Visão da Câmera compõem os cinco painéis principais, que são exibidos para o usuário na abertura do aplicativo. Os painéis Arquivo, Propriedades da Peça e Ajuda são três painéis que concorrem com a exibição do painel Fábrica de Peças. Existem também os painéis Abrir e Tutoriais de Ajuda, que são exibidos sobre os demais painéis quando solicitados. Na Figura 20 é exibida a visão geral do aplicativo logo que é aberto.

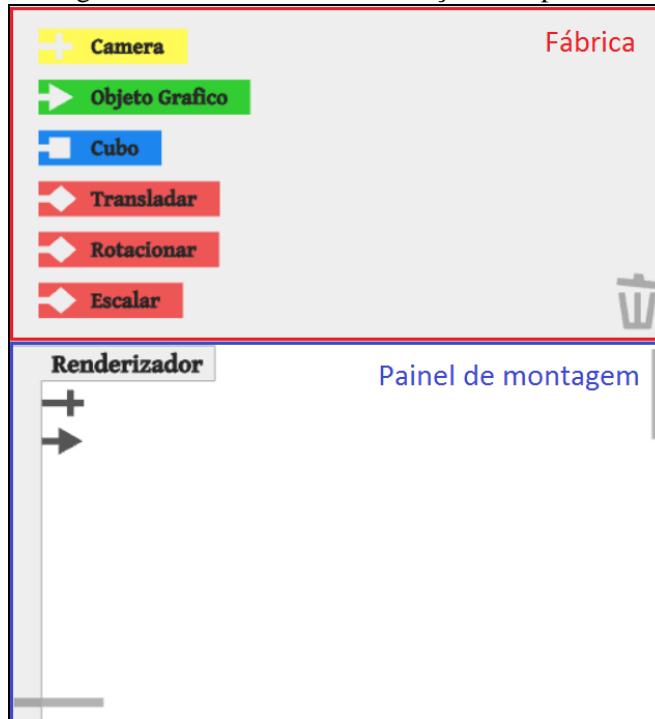
Figura 20 – Tela de entrada do VisEdu-CG



O painel Fábrica de Peças é o principal painel da aplicação e é dividido e dois subpainéis: fábrica e painel de montagem. Estes dois subpainéis compõem o editor onde o

usuário irá montar e manipular os exercícios. Na Figura 21 é apresentado o painel Fábrica de Peças e seus subpainéis.

Figura 21 – Painel Fábrica de Peças e subpainéis



O painel fábrica fica localizado na parte superior da Fábrica de Peças e possibilita a criação de novas peças ou a remoção das peças já existentes dentro do painel de montagem. Ele possibilita a criação de 6 peças diferentes, que são:

- Câmera: peça que representa a câmera de um ambiente gráfico. Através dessa peça é possível editar os atributos de uma câmera do tipo perspectiva e avaliar o impacto de cada um de seus atributos;
- Objeto Gráfico: peça que representa um objeto gráfico de um ambiente gráfico. Um objeto gráfico agrupa transformações, empilhando-as numa matriz que será utilizada para transformar os objetos filhos. A peça permite o encaixe de peças filhas do tipo transformações geométricas, objetos geométricos e outros objetos gráficos;
- Cubo: peça que representa um objeto geométrico de um ambiente gráfico. Através desta peça é possível editar os atributos de um objeto do tipo cubo e avaliar o impacto de cada um desses atributos. Este cubo é formado por seis faces utilizando o `gl_quads`;

- d) **Transladar:** peça que representa uma transformação geométrica translação de um objeto gráfico. Através desta peça é possível editar os atributos de uma translação e avaliar o impacto dos mesmos;
- e) **Rotacionar:** peça que representa uma transformação geométrica rotação de um objeto gráfico. Através desta peça é possível editar os atributos de uma rotação e avaliar o impacto dos mesmos;
- f) **Escalar:** peça que representa uma transformação geométrica escala de um objeto gráfico. Através desta peça é possível editar os atributos de uma escala e avaliar o impacto dos mesmos.

O painel de montagem fica localizado na parte inferior da Fábrica de Peças e possui uma peça principal denominada Renderizador. As demais peças deverão ser encaixadas nesta peça ou em peças encaixadas nela. No APÊNDICE A são exibidas todas as peças existentes na aplicação. Todos os outros painéis da aplicação irão tomar como base o resultado dos encaixes definidos nesse painel. Um exemplo disso é o painel Lista de Peças, que gera um *tree view* listando todas as peças encaixadas no painel de montagem, demonstrando a relação de pai e filho existente entre as peças.

Para criar uma peça é necessário clicar sobre uma das peças no painel fábrica e arrastá-la com o mouse até o painel de montagem. Esse processo é demonstrado na Figura 22. Se a peça for solta diante de um encaixe correspondente ao recorte da peça, a peça criada será encaixada no encaixe. Quando a peça fica diante de um encaixe correspondente, a cor do encaixe muda, indicando que a peça será encaixada naquele encaixe se for solta. Se a peça criada for solta em qualquer outro lugar do painel de montagem, irá ficar flutuando no lugar em que foi solta.

Se uma peça criada for solta dentro do painel fábrica, ela será automaticamente excluída. Para excluir uma peça do painel de montagem é preciso clicar sobre a peça e arrastá-la até ficar sobre o painel fábrica. Quando isso é feito a lixeira fica com uma cor diferente, indicando que se a peça for solta, ela será removida. Na Figura 23 pode-se visualizar as maneiras de ser excluir uma peça.

E possível definir um novo encaixe para peças já encaixadas ou definir um encaixe para peças flutuando no painel de montagem. Para isso, é necessário clicar e arrastar a peça até um encaixe correspondente ao recorte da peça. Quando a peça fica diante de um encaixe correspondente, a cor do encaixe muda, indicando que a peça será encaixada naquele encaixe se for solta. Se a peça for solta fora de um encaixe correspondente, caso já possua um encaixe,

o movimento da peça será anulado; caso não possua, irá permanecer na posição em que foi solta. Na Figura 24 é demonstrada a realização desse processo.

Figura 22 – Demonstração do processo de criação de uma peça

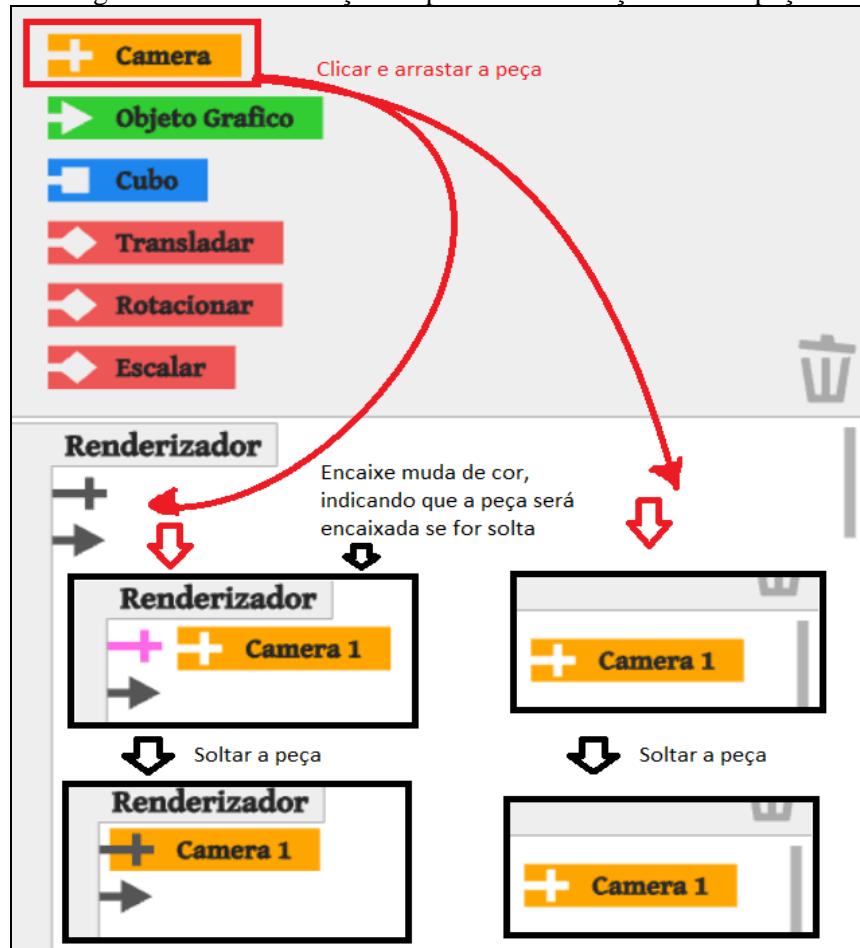
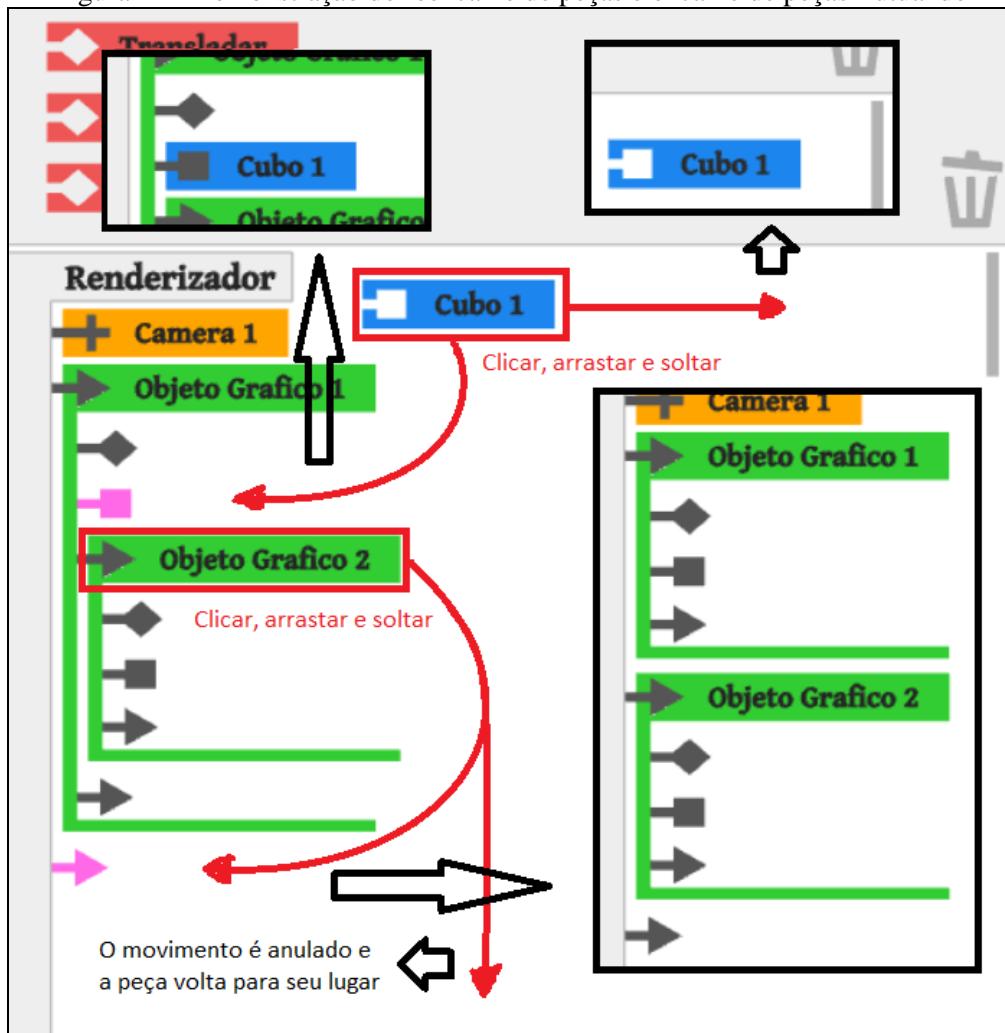


Figura 23 – Demonstração do processo de exclusão de uma peça



Figura 24 - Demonstração do reencaixe de peças e encaixe de peças flutuando

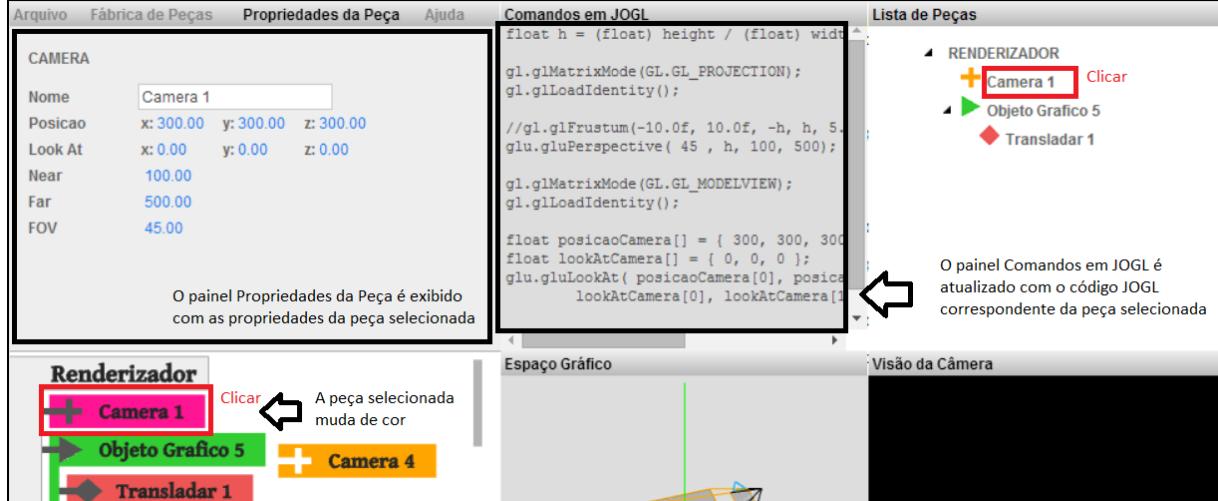


Para editar as propriedades de uma peça existente no painel de montagem é preciso selecionar a peça. A seleção de uma peça é feita através de um clique simples sobre a peça ou clicar sobre o nome da peça no painel `Lista de Peças`. Peças do painel que não estão encaixadas, podem ser selecionadas apenas pelo clique simples sobre a peça, visto que essas peças não são listadas no painel `Lista de Peças`. Quando a peça é selecionada o sistema abre o painel `Propriedades da Peça`, atualiza o conteúdo do painel `Comandos em JOGL` e muda a cor da peça, para ser possível identificá-la. Na Figura 25 são demonstradas as duas formas de selecionar uma peça.

Todas as peças são criadas com um valor padrão para suas propriedades, e através do painel `Propriedades da Peça` é possível editar essas propriedades. Os painéis `Espaço Gráfico` e `Visão da Câmera` exibem um ambiente gráfico que é gerado a partir do resultado produzido pelo encaixe das peças do painel de montagem e suas respectivas propriedades. Ao modificar o encaixe das peças ou suas propriedades é possível visualizar o impacto da alteração nesse ambiente gráfico. No apêndice B são exibidas as propriedades existentes para

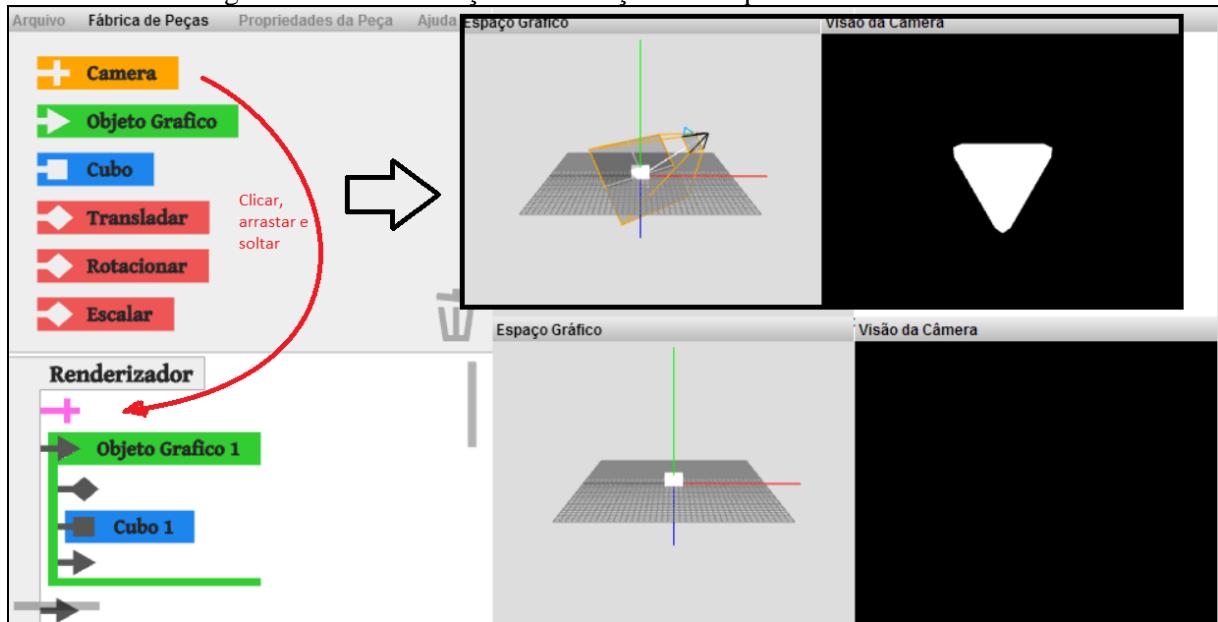
cada peça e exemplos de como essas propriedades interferem nos painéis Espaço Gráfico e Visão da Câmera.

Figura 25 - Demonstração do processo de seleção de uma peça



O painel Visão da Câmera exibe especificamente a visão desse ambiente, que é personalizada pelas propriedades da peça Câmera. Ao encaixar a peça Câmera na peça Renderizador, o painel Visão da Câmera é ativado. Esse painel irá utilizar as propriedades da peça encaixada para determinar o que ele irá enxergar dentro do ambiente gráfico gerado pelo encaixe das peças. Ele utiliza como cor de limpeza a cor de limpeza definida nas propriedades da peça Renderizador. Na Figura 26 é exibido um exemplo do comportamento desse painel.

Figura 26 - Demonstração de interação com o painel Visão da Câmera

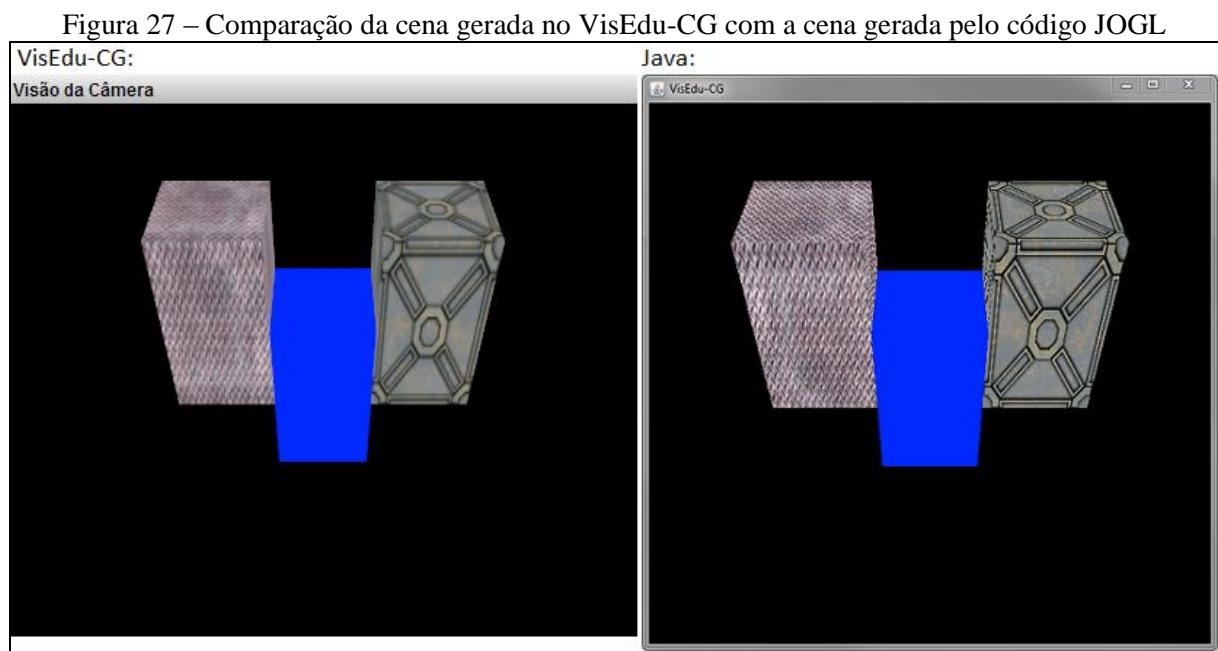


O painel Comandos em JOGL exibe o código fonte em Java que é necessário para reproduzir o resultado gráfico da peça selecionada no editor e seus filhos. O código Java

gerado usa a biblioteca JOGL versão 1.1.1a. No APÊNDICE C é possível visualizar os códigos JOGL gerados para cada tipo de peça. Diferente das demais peças, a peça Renderizador, ao ser selecionada, gera o código fonte de uma classe pronta para reproduzir o cenário visualizado pelo painel Visão da Câmera em um projeto Java. Para reproduzir o cenário, é necessário criar uma classe no Java com o código fonte exibido no painel Comandos em JOGL em um projeto configurado com a biblioteca JOGL e executar o código. Alguns ajustes no código fonte serão necessários, como informar o pacote da classe ou informar o caminho das texturas no diretório local do projeto.

Na

Figura 27 pode-se visualizar o exemplo de uma comparação entre a cena 3D gerada no painel Visão da Câmera e a cena gerada pelo Java, utilizando a classe gerada no painel Comandos em JOGL para a peça Renderizador. No exemplo foram utilizados três cubos, que foram modificados através do uso das peças Transladar e Escalar.



3.4 RESULTADOS E DISCUSSÃO

Este trabalho apresenta uma aplicação *web* para visualização de material educacional, onde é possível trabalhar com alguns conceitos básicos de computação gráfica através de um jogo de encaixe de formas geométricas. Assim, foi cumprido o objetivo principal do trabalho, que visava estender a pesquisa feita por Araújo (2012), e também foi cumprido um dos objetivos específicos, que era trazer os recursos disponíveis no AduboGL para um ambiente *web*. O VisEdu-CG encontra-se disponível para uso na Internet - ver VISEDU-CG (2013).

Através do uso do Three.js, foi possível agilizar o processo de desenvolvimento, visto que ele disponibiliza uma camada com uma interface simples para o uso do WebGL e várias bibliotecas auxiliares que facilitaram o desenvolvimento da aplicação. Isso permitiu o desenvolvimento dos outros objetivos específicos do trabalho, que eram a visualização da cena criada pelo exercício em um grafo de cena e o uso de conceitos básicos de computação gráfica não abordados no AduboGL: câmeras e texturas. Essa agilidade também permitiu uma melhoria no processo de encaixe das peças, que foi modificado para ser feito através de um jogo de encaixe de formas geométricas, inspirado a partir da análise de usabilidade do StarLogo TNG. A geração dos comandos de cada peça foi alterada para usar comandos JOGL no lugar dos comandos OpenGL em C++, visto que o Java é a linguagem padrão utilizada no ensino de programação de todas as disciplinas do curso de ciência da computação da FURB.

O presente trabalho pode ser considerado um jogo educacional (ver seção 2.1.3). Apesar de disponibilizar um jogo simples de encaixe de formas geométricas, a existência de um desafio, por menor que seja, pode despertar no aluno a motivação, ou curiosidade, de descobrir o que ocorre se as peças forem encaixadas nos lugares corretos. Ele também pode ser classificado como um software educacional do tipo micromundo, dentro da abordagem de ambientes interativos de aprendizagem (ver seção 2.1.2), visto que o usuário pode interagir com o sistema de forma livre, sem a obrigatoriedade de seguir um tutorial, e experimentar por si mesmo como o conceito representado por cada peça afeta um ambiente gráfico.

3.4.1 Análise de desempenho da aplicação

Na avaliação de desempenho da aplicação foi realizada somente a operação de inclusão de novas peças no painel de montagem. Os testes foram feitos em três navegadores diferentes que foram o Chrome, o Firefox e o Internet Explorer (descritos na seção 3.3.1). O computador utilizado para efetuar os testes foi um Intel® Core™ i7-3632QM CPU 2.20GHz, 8 GB de memória RAM DDR3, 2 GB de memória de vídeo dedicada e sistema operacional Windows 8.1 64 bits.

Os critérios utilizados para testar o desempenho da aplicação foram o processamento de Frames Por Segundo (FPS) em relação a quantidade de objetos adicionados na cena e o consumo de memória necessário para o sistema trabalhar com esses objetos. As ferramentas utilizadas para fazer o levantamento das informações foram o gerenciador de tarefas do Windows, para medir o consumo de memória, e a biblioteca Stats, que é uma biblioteca auxiliar do Three.js, para medir o processamento de FPS.

No Quadro 24 pode-se visualizar a análise de desempenho feita sobre o navegador Chrome. Nos testes feitos foi possível verificar que o sistema parou de funcionar corretamente após a inclusão da 42^a peça. Contudo, até a inclusão dessa peça, o processamento de FPS continuou estável em 60 FPS.

Quadro 24 – Análise de desempenho no navegador Chrome

Qtde. objetos	FPS	Memória (MB)	Observação
6	60	608	Sistema funciona normalmente. Foi utilizada 1 peça Câmera, 1 peça Objeto Gráfico, 1 peça Transladar, 1 peça Rotacionar, 1 peça Escalar e 1 peça Cubo.
31	60	1.131	Sistema funciona normalmente. Foram utilizadas 1 peça Câmera, 6 peças Objeto Gráfico, 6 peças Transladar, 6 peças Rotacionar, 6 peças Escalar e 6 peças Cubo.
41	60	1.283	Sistema funciona normalmente. Foram utilizadas 1 peça Câmera, 8 peças Objeto Gráfico, 8 peças Transladar, 8 peças Rotacionar, 8 peças Escalar e 8 peças Cubo.
42	22,12	1.368	Sistema funciona, mas fica lento para incluir novas peças. Foram utilizadas 1 peça Câmera, 9 peças Objeto Gráfico, 8 peças Transladar, 8 peças Rotacionar, 8 peças Escalar e 8 peças Cubo.

A análise de desempenho feita sobre o navegador Firefox é apresentada no Quadro 25. Nos testes feitos foi possível verificar que o sistema parou de funcionar após a inclusão da 106^a peça, sendo necessário reiniciá-lo. Contudo, até a inclusão dessa peça, o processamento de FPS continuou satisfatório em torno de 44 FPS.

Quadro 25 – Análise de desempenho no navegador Firefox

Qtde. objetos	FPS	Memória (MG)	Observação
6	57,75	528	Sistema funciona normalmente. Foi utilizada 1 peça Câmera, 1 peça Objeto Gráfico, 1 peça Transladar, 1 peça Rotacionar, 1 peça Escalar e 1 peça Cubo.
31	51,96	1.112	Sistema funciona normalmente. Foram utilizadas 1 peça Câmera, 6 peças Objeto Gráfico, 6 peças Transladar, 6 peças Rotacionar, 6 peças Escalar e 6 peças Cubo.
61	45,3	1.943	Sistema funciona normalmente. Foram utilizadas 1 peça Câmera, 12 peças Objeto Gráfico, 12 peças Transladar, 12 peças Rotacionar, 12 peças Escalar e 12 peças Cubo.
106	0	2.703	Sistema travou ao incluir tentar incluir a 106 ^a peça. Foram utilizadas 1 peça Câmera, 21 peças Objeto Gráfico, 21 peças Transladar, 21 peças Rotacionar, 20 peças Escalar e 21 peças Cubo.

No Quadro 26 pode-se visualizar a análise de desempenho feita sobre o navegador Internet Explorer. Nos testes feitos foi possível verificar que o sistema parou de funcionar após a inclusão da 51^a peça, sendo necessário reiniciá-lo. Contudo, até a inclusão dessa peça, o processamento de FPS continuou satisfatório em torno de 44 FPS. Mesmo tendo um bom desempenho no processamento de FPS e no consumo de memória, no Internet Explorer a

renderização do painel Espaço Gráfico não é feita corretamente. As linhas do painel são renderizadas muito grossas, dificultando a distinção dos objetos existentes no painel.

Quadro 26 – Análise de desempenho no navegador Internet Explorer

Qtde. objetos	FPS	Memória (MG)	Observação
6	67,39	403	Sistema funciona normalmente. Foi utilizada 1 peça Câmera, 1 peça Objeto Gráfico, 1 peça Transladar, 1 peça Rotacionar, 1 peça Escalar e 1 peça Cubo.
31	44,65	913	Sistema funciona normalmente. Foram utilizadas 1 peça Câmera, 6 peças Objeto Gráfico, 6 peças Transladar, 6 peças Rotacionar, 6 peças Escalar e 6 peças Cubo.
51	0	1.573	Sistema travou ao incluir tentar incluir a 52ª peça. Foram utilizadas 1 peça Câmera, 10 peças Objeto Gráfico, 10 peças Transladar, 10 peças Rotacionar, 10 peças Escalar e 10 peças Cubo.

Diante das avaliações efetuadas, percebe-se que o navegador Firefox é o mais indicado para o uso do sistema. Sua maior vantagem é suportar uma quantidade muito superior de objetos em relação aos demais navegadores (105 contra 50 do Internet Explorer e 41 do Chrome). O Internet Explorer, apesar de ter um consumo de memória menor, comparado aos outros navegadores, e um bom desempenho no processamento de FPS, não exibe corretamente o painel Espaço Gráfico.

O Chrome possui o maior consumo de memória entre os navegadores. Sua vantagem em relação aos demais é que ele não trava após a inclusão da peça limite (que nos testes foi a 42ª peça). Apesar do sistema ficar lento, praticamente impedindo a inclusão de peças, é possível ao usuário salvar ou continuar alterando o exercício, modificando propriedades e excluindo peças. Nos outros navegadores, ao chegar no número limite de peças (106 no Firefox e 51 no Internet Explorer), o sistema trava e precisa ser reiniciado, fazendo com que o usuário perca todas as informações do exercício não salvas até o momento.

3.4.2 Análise de usabilidade da aplicação

Com o apoio do professor da disciplina de Computação Gráfica da FURB, o aplicativo foi utilizado em sala de aula para que os alunos da disciplina pudessem avaliá-lo. Dos alunos presentes em sala, 28 alunos de duas turmas diferentes participaram da avaliação respondendo um questionário de usabilidade.

Com o objetivo de obter um alinhamento na experimentação do sistema e garantir que os alunos que responderiam o questionário tivessem um conhecimento mínimo dos recursos do VisEdu-CG, com o auxílio do professor da disciplina, foi criado um exercício que os alunos pudessem resolver usando o aplicativo. Esse exercício pode ser observado no

APÊNDICE D. O exercício foi aplicado aos alunos no fim do semestre e todos já possuíam o conhecimento dos conceitos de computação gráfica trabalhados pelo aplicativo. Assim, foi necessário apenas introduzir aos alunos como o sistema se comportava e explicar como o exercício deveria ser resolvido.

Após a realização do exercício, foi solicitado que os alunos respondessem um questionário avaliando a usabilidade do sistema e o quanto ele auxiliou na compreensão dos conceitos de computação gráfica de câmera, grafo de cena, transformações geométricas e textura. O questionário possui 12 questões e pode ser visualizado no Quadro 27.

Quadro 27 – Questionário de usabilidade do VisEdu-CG

Nº	Questão	Tipo de resposta
1	Qual o seu nível de conhecimento sobre conceitos de "Computação Gráfica"?	Múltipla escolha: Nenhum; Ruim; Bom ou Ótimo.
2	O que você achou do " <i>layout</i> " do aplicativo?	Múltipla escolha: Inadequado; Ultrapassado; Normal ou Moderno.
3	O que você achou da "usabilidade" do aplicativo?	Múltipla escolha: Difícil, nada intuitivo; Mediano, mas pouco intuitivo; Mediano, mas intuitivo ou Fácil e intuitivo.
4	O que você achou do tamanho e disponibilidade das "janelas"?	Múltipla escolha: Péssimo; Ruim; Bom ou Ótimo.
5	O código fonte visualizado no painel "Comandos JOGL" é legível e fácil de compreender?	Múltipla escolha: Ilegível e difícil de compreender; Legível, mas poderia ser mais sucinto; Legível, mas poderia ser melhor organizado ou Legível e comprehensível.
6	O aplicativo ajudou a entender o conceito de "Câmera Sintética" e suas propriedades (posição, <i>look at, near, far e fov</i>)?	Múltipla escolha: Não ajudou; Ajudou pouco; Ajudou ou Ajudou muito.
7	O aplicativo ajudou a entender o conceito de "Grafo de Cena" (uso do <code>glPushMatrix/glPopMatrix</code>)?	Múltipla escolha: Não ajudou; Ajudou pouco; Ajudou ou Ajudou muito.
8	O aplicativo ajudou a entender o conceito de "Transformações Geométricas" (translação, escala e rotação)?	Múltipla escolha: Não ajudou; Ajudou pouco; Ajudou ou Ajudou muito.
9	O aplicativo ajudou a entender a aplicação de "texturas" nos objetos gráficos?	Múltipla escolha: Não ajudou; Ajudou pouco; Ajudou ou Ajudou muito.
10	Com base nas respostas anteriores, quais seriam os principais "Pontos Positivos" no uso do aplicativo?	Descritiva.
11	Com base nas respostas anteriores, quais seriam os principais "Pontos Negativos" no uso do aplicativo?	Descritiva.
12	Com base nas respostas anteriores, quais seriam as suas "Sugestões" de melhoria para o aplicativo?	Descritiva.

Na primeira questão, conforme é demonstrado na Figura 28, a maior parte dos alunos que responderam o questionário se julgaram com um bom ou com um ótimo conhecimento dos conceitos de computação gráfica. Isso se deve pelo fato de todos os alunos estarem concluindo a disciplina de computação gráfica.

A maior parte dos alunos considerou o leiaute da aplicação normal e oito alunos o consideraram moderno. Na Figura 29 é apresentado um gráfico com as respostas da 2^a questão do questionário de usabilidade.

Figura 28 – Gráfico com as respostas da 1^a questão

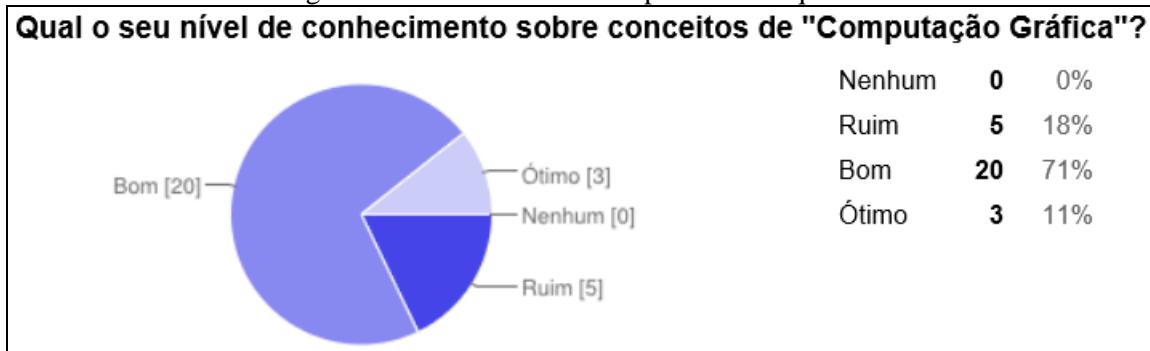
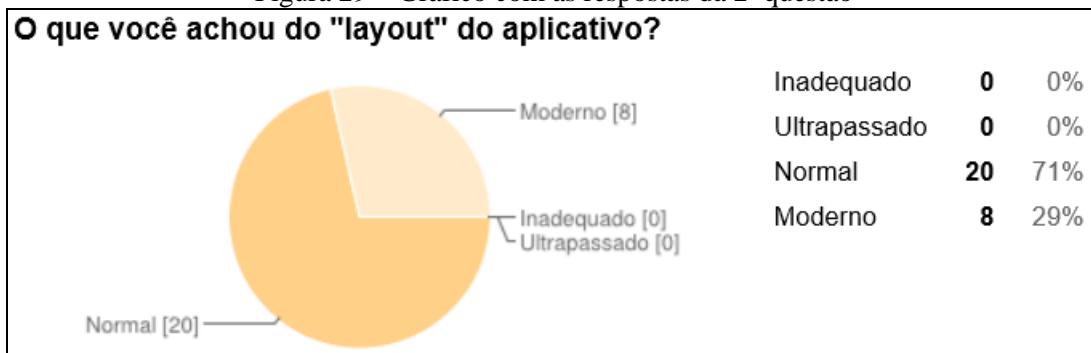
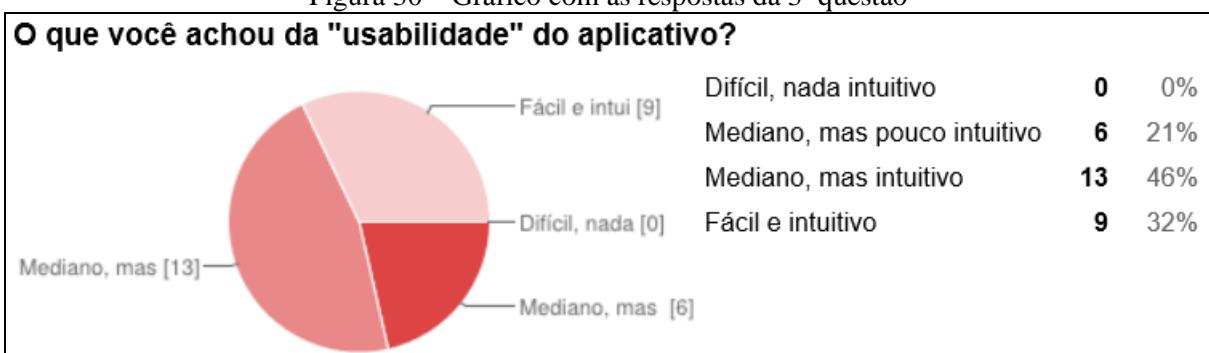


Figura 29 – Gráfico com as respostas da 2^a questão



Quanto a usabilidade da aplicação a maioria dos alunos considerou o sistema intuitivo ou bem intuitivo. Contudo, na Figura 30 pode-se observar que 21% dos alunos consideraram o aplicativo pouco intuitivo e este aspecto foi bem criticado nas questões descritivas, onde foi apontada uma necessidade de mudança no comportamento de vários aspectos, como adição de *hints*, remover a seleção automática do painel Propriedades da Peça, melhorias nas barras de rolagem (funcionar usando o mouse) e melhoria de performance.

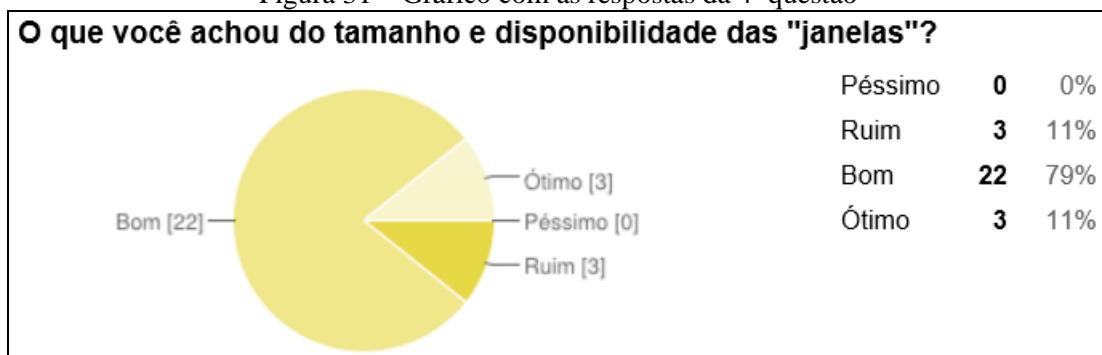
Figura 30 – Gráfico com as respostas da 3^a questão



Apenas três alunos classificaram como ruim a distribuição das janelas do sistema, os demais a classificaram como boa ou ótima. Na Figura 31 pode-se visualizar o gráfico da 4^a

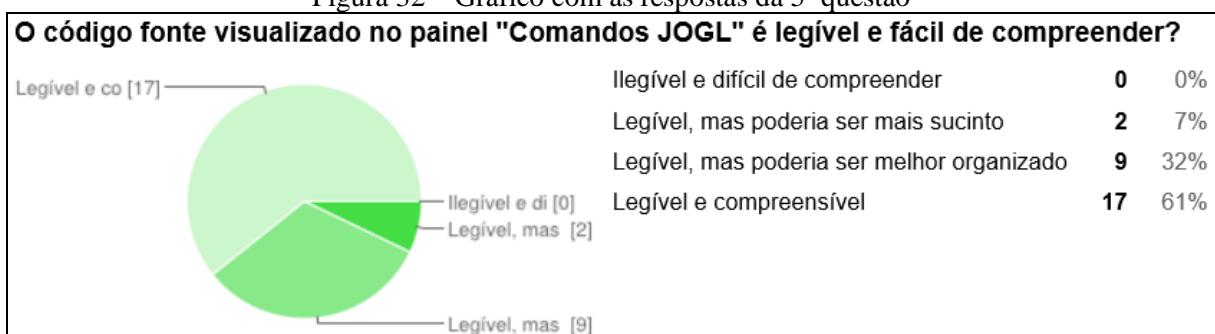
questão, que avaliou essa característica. Apesar disso, nas questões descritivas, a distribuição das janelas foi o aspecto mais criticado do sistema, onde foi fortemente sugerido redistribuí-las para destacar as mais importantes e permitir que o usuário pudesse minimizar ou maximizar os painéis exibidos. Também foi sugerido esconder ou remover o painel Lista de Peças, pois a hierarquia de peças já pode ser visualizada no painel de montagem. Outra sugestão foi o acoplamento dos recursos de abrir, exportar e ajuda, apenas como menu, ao lado do menu Fábrica de Peças, sem a necessidade de abrir um painel para exibir essas opções. Foi sugerido também exibir o painel Propriedades da Peça em um painel flutuante ao lado da peça selecionada.

Figura 31 – Gráfico com as respostas da 4^a questão



A questão 5, apresentada na Figura 32, demonstrou uma boa legibilidade do código JOGL exibido para cada peça. Mesmo assim, a legibilidade de código foi um dos aspectos criticados nas questões descritivas, onde foi sugerido a criação de mais recursos de ajuda ou de visualização (colorir o código) que permitissem compreender melhor o código e onde ele se encaixa com cada peça.

Figura 32 – Gráfico com as respostas da 5^a questão



De acordo com o questionário, o conceito de câmera foi o conceito mais bem trabalhado pelo sistema. A maioria dos alunos considerou que o sistema ajudou ou ajudou muito na compreensão desse conceito, conforme pode-se observar na Figura 33.

Ao avaliar as respostas da 7^a questão do questionário, apresentadas na Figura 34, observa-se que conceito de grafo de cena não foi tão bem trabalhado quando o conceito de

câmera, considerando que 40% dos alunos consideraram que o aplicativo não ajudou ou ajudou pouco na compreensão desse conceito. Apesar disso, a maior parte dos alunos julgou que o sistema ajudou ou ajudou muito a compreender esse conceito.

Figura 33 – Gráfico com as respostas da 6^a questão

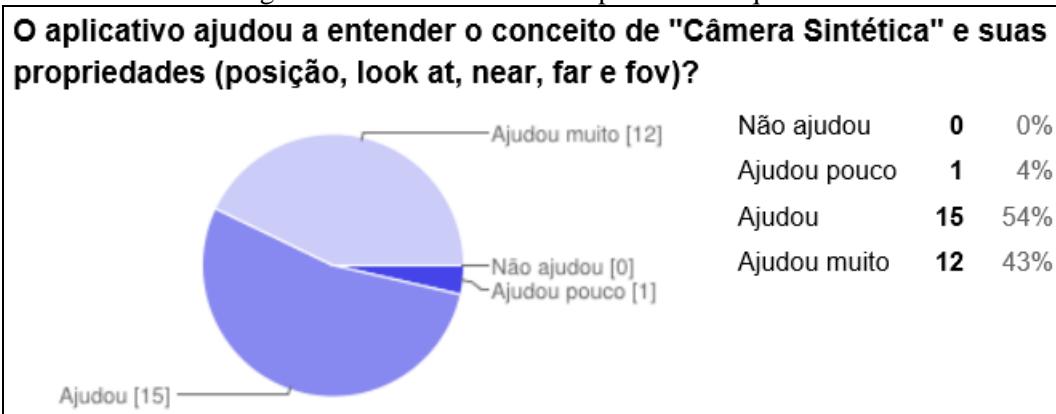
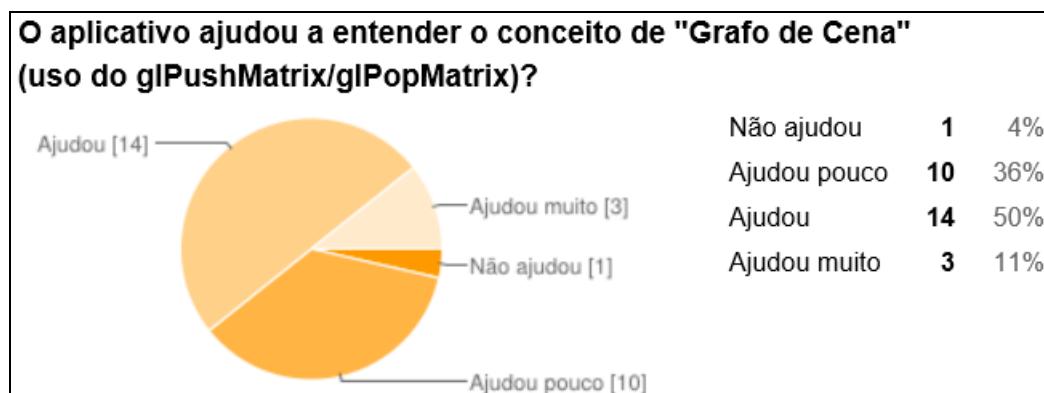
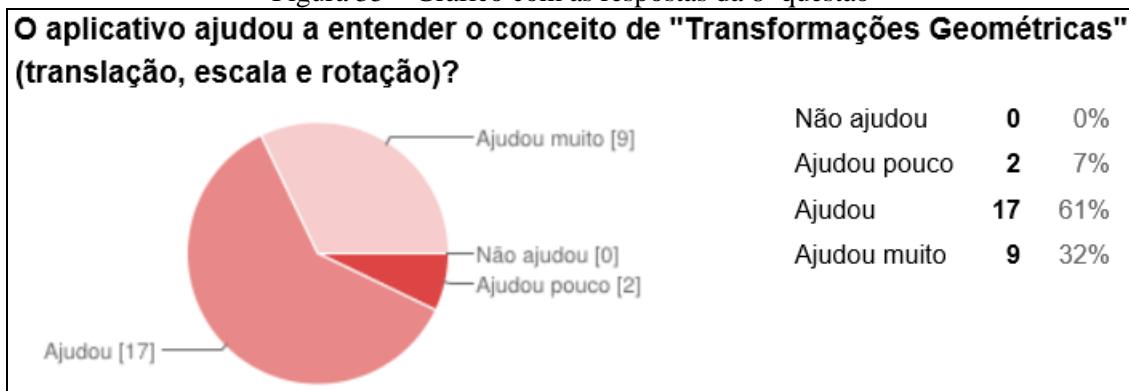


Figura 34 – Gráfico com as respostas da 7^a questão



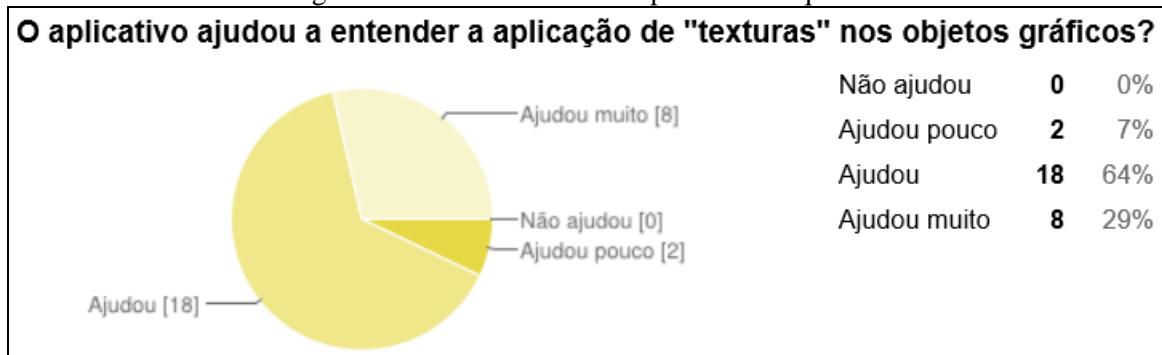
O conceito de transformações geométricas foi bem avaliado pelos alunos. Na Figura 35 é apresentado o gráfico com as respostas da 8^a questão, que averiguou a capacidade do sistema ajudar na compreensão desse conceito. Apenas 2 alunos consideraram que a aplicação ajudou pouco na compreensão desse conceito.

Figura 35 – Gráfico com as respostas da 8^a questão



A avaliação efetuada pela 9^a questão, apresentada na Figura 36, demonstrou uma boa aceitação no trabalho com o conceito de textura. Mesmo com dois dos alunos apontando que o sistema ajudou pouco na compreensão desse conceito, a maior parte dos alunos considerou que o sistema ajudou a compreendê-lo.

Figura 36 – Gráfico com as respostas da 9^a questão



As respostas das questões descritivas podem ser visualizadas no APÊNDICE E. Na questão 10 foram destacados os pontos positivos da aplicação. Os alunos destacaram a facilidade de uso, a simplicidade e a objetividade com que o sistema trata os conceitos de computação gráfica, o que acaba auxiliando no entendimento desses conceitos. O jogo de encaixe (arrastar e soltar das peças) foi elogiado por tornar o uso do aplicativo e o aprendizado, mais dinâmico e intuitivo. A geração de comandos JOGL em blocos (por peça selecionada) foi bem vista por tornar o código mais organizado e legível e ajudar a tirar dúvidas de implementação. A atualização dos resultados gráficos e comandos JOGL em tempo real foi apontada como uma ótima ajuda na formação da lógica de programação de computação gráfica.

Os pontos negativos da aplicação e sugestões de melhorias foram descritos na 11^a e 12^a questão. Além da necessidade de melhoria na disposição das janelas, na formatação dos comandos JOGL e na usabilidade, outras considerações foram feitas. Algumas dessas considerações se referem a erros que foram ajustados após a aplicação do questionário. Algumas respostas citam problemas na performance, como lentidão ou a tela ficar preta. Considerando que não foi possível reproduzir alguns desses problemas nos testes efetuados, eles foram classificados como problemas de performance, ocorridos pela falta de desempenho dos computadores do laboratório utilizado, visto que, por exemplo, eles não tinham placa de vídeo dedicada. O sistema mesmo sendo simples, exige um bom desempenho de máquina, conforme pode ser verificado na seção 3.4.1, onde o sistema trava após o uso de uma quantidade não muito grande de peças.

Um aluno descreveu que teve dificuldade ao manipular a visão do painel Espaço Gráfico com o mouse e outro teve dificuldade de remover objetos, mas isso foi considerado uma falta de familiaridade com os recursos, visto que nenhum dos outros alunos criticou esse recurso. Um aluno sugeriu a criação de um tutorial introdutório à ferramenta, para que não houvesse necessidade de um conhecimento inicial da ferramenta antes de usá-la. Foi sugerida uma alteração no painel Espaço Gráfico, para permitir selecionar a peça Câmera através do painel e movimentá-la, girá-la ou escalá-la, alterando suas propriedades sem a necessidade de informar seus valores no painel Propriedades da Peça.

No tratamento de erros e melhorias indicados pelos alunos na análise de usabilidade, optou-se por implementar neste trabalho apenas os erros e as sugestões simples. A maioria das sugestões estavam relacionadas a mudanças significativas na usabilidade das janelas, na exibição mais legível dos comandos JOGL ou em um controle mais complexo do painel Espaço Gráfico. Mesmo que a biblioteca UI, utilizada para controlar todos os elementos HTML da aplicação, tenha sido satisfatória para implementar os recursos disponibilizados pelo sistema, a usabilidade que ela disponibiliza dos elementos é completamente manual, ou seja, todos os elementos HTML são ajustados e posicionados manualmente. Sendo assim, qualquer ajuste e controle visual é manual e gera uma carga de trabalho considerável. Pela falta de tempo hábil para gerar um controle de todas essas melhorias, ou de alterar o sistema para utilizar uma outra biblioteca que facilitasse a criação delas, elas não foram implementadas e serão colocadas nas sugestões de extensão.

3.4.3 Relação do presente trabalho com os trabalhos correlatos

No Quadro 28 pode-se visualizar uma análise comparativa entre o presente trabalho e os trabalhos correlatos. Foram comparados com este trabalho o *framework* Three.js, o software StarLogo TNG e o trabalho de pesquisa AduboGL, estendido pelo presente trabalho.

O Three.js, por ser um *framework* para desenvolvimento de ambientes gráficos, não possui nenhum recurso visual que auxilia na compreensão do exercício, como o encaixe de peças, visualização do código nativo da biblioteca gráfica ou visualização do grafo de cena. Ele, ao contrário dos demais trabalhos, não é um software educacional, e assim como o StarLogo TNG e o AduboGL, não está disponível na *web*. Como o StarLogo TNG é usado para o ensino de programação, ele não permite a alteração do código em tempo real e também não possui os recursos específicos para a compreensão dos conceitos de computação gráfica, como a visualização do grafo de cena e a associação de pai e filho entre objetos gráficos. Quanto ao AduboGL, pode-se observar que todos os seus recursos foram repassados para o

VisEdu-CG, incluíndo algumas melhorias. Essas melhorias estavam previstas nos objetivos propostos, inclusive a associação de pai e filho entre objetos gráficos, que é necessária para criação do grafo de cena.

Quadro 28 – Comparaçāo com os trabalhos correlatos

Características	Three.js	StarLogo TNG	AduboGL	VisEdu-CG
Interação com o aluno a partir de peças	Não	Sim	Sim	Sim
Capacidade de resolver exercícios	Sim	Sim	Sim	Sim
Resolução de exercícios via código-fonte	Sim	Não	Não	Não
Alteração de parâmetros do código em tempo de execução	Não	Não	Sim	Sim
Exibe comandos da biblioteca gráfica	Não	Não	Sim	Sim
Permite alteração da posição da câmera	Sim	Sim	Não	Sim
Permite aplicação de texturas	Sim	Sim	Não	Sim
Exibe o grafo de cena do exercício	Não	Não	Não	Sim
Permite associação pai e filho entre objetos gráficos	Sim	Não	Não	Sim
Software educacional	Não	Sim	Sim	Sim
Tipo de integração	<i>Framework</i>	Software	Software	Software
Disponibilidade	<i>Desktop</i>	<i>Desktop</i>	<i>Desktop</i>	Web

4 CONCLUSÃO

Este trabalho apresentou o desenvolvimento de uma aplicação *web* para visualização de conceitos básicos de computação gráfica. Essa aplicação disponibiliza um jogo de encaixe de formas geométricas, que indica a ordem e a relação existente entre os conceitos que foram trabalhados.

O sistema desenvolvido permite o estudo dos conceitos câmera sintética, grafo de cena transformações geométricas, composição de transformações geométricas e textura. O aplicativo foi desenvolvido para os alunos que já tem um conhecimento de CG e irão usar o VisEdu para praticar e visualizar melhor os conceitos em um ambiente 3D.

Mesmo que o protótipo desenvolvido por esse trabalho seja limitado quanto a quantidade de conceitos trabalhados, através dele é possível realizar a transmissão de conhecimento apenas com a interação do aluno com o software. A partir do levantamento teórico, classificou-se a aplicação como um jogo educacional e como um software educacional do tipo micromundo, dentro da abordagem de ambientes interativos de aprendizagem. A aplicação é um instrumento que busca auxiliar o professor na metodologia da disciplina e permitir que os alunos pratiquem os conceitos ensinados em sala de aula antes de iniciar as atividades de programação.

A biblioteca Three.js, utilizada para abstrair a implementação WebGL dos ambientes gráficos da aplicação, se mostrou eficiente e fácil de usar. A simplicidade da biblioteca, juntamente com as várias bibliotecas auxiliares que ela fornece, permitiram que todos os objetivos do trabalho fossem alcançados. A grande quantidade de exemplos de implementação da biblioteca, juntamente com sua documentação, foi essencial para o seu uso, e consequentemente, para a implementação da aplicação.

Recursos importantes como seleção de objetos gráficos, texturas, controle da câmera do ambiente gráfico utilizando o mouse e criação elementos HTML, foram todos criados a partir do uso do Three.js. Outras bibliotecas utilizadas, como as bibliotecas JSColor e DDTreeMenu, também se mostraram simples de utilizar, mesmo sendo necessário adaptá-las para o sistema. Essas permitiram criar mais facilmente recursos visuais, como a paleta de cores e a lista de peças.

A partir das análises efetuadas, foi possível observar a existência de alguns problemas de performance, como o sistema travar após ultrapassar um determinado número de peças ou a exibição da tela preta e relatos de lentidão descritos pelos alunos através do questionário de usabilidade. Mesmo diante disso, o uso da aplicação não foi comprometido, visto que todos os

alunos conseguiram concluir os exercícios propostos e destacaram a facilidade e simplicidade de uso como um dos fatores positivos do sistema. Desta forma, considerou-se a que a aplicação obteve um desempenho muito bom.

Referente aos trabalhos correlatos, foi possível repassar para a aplicação os recursos existentes no AduboGL e incluir melhorias, conforme previa os objetivos do trabalho, permitindo que outros conceitos de computação gráfica fossem trabalhados pelo aplicativo. O Three.js foi utilizado como uma ferramenta essencial no desenvolvimento do trabalho e o StarLogo TNG foi analisado para trazer melhorias na usabilidade, como o jogo de encaixe de formas geométricas.

A aplicação trouxe para um ambiente *web* o trabalho de pesquisa iniciado por Araújo (2012). Dentro dessa perspectiva, é disponibilizada para o professor e alunos da disciplina de Computação Gráfica, com fácil acesso, uma nova ferramenta para transmissão e obtenção de conhecimento. Apesar de ser um protótipo, a existência de um software educacional *web* se mostrou efetiva, e cria a possibilidade não só de uma nova ferramenta de ensino, mas de uma nova maneira de ensinar. Maneira esta mais efetiva, capaz de utilizar a capacidade cognitiva do aluno e fazê-lo aprender através da sua própria interação com o objeto de estudo.

Com a linha de pesquisa, continuada por este trabalho, criou-se uma forma de disponibilizar os conceitos de computação gráfica para os alunos uma forma diferente. Não apenas utilizando os recursos de informática para compartilhar textos, imagens ou um ambiente de programação, mas fazer do aluno, seu próprio professor, possibilitando uma aula mais dinâmica e motivadora.

4.1 EXTENSÕES

São sugeridas as seguintes extensões para a continuidade do trabalho:

- a) incluir a aplicação em um sistema a ser criado dentro da abordagem de aprendizado socialmente distribuído, ou seja, fazer com que ele componha uma ferramenta de ensino a distância para disciplinas envolvendo conceitos de computação gráfica;
- b) incluir na aplicação um tutorial para ensinar quem não conhece computação gráfica e um tutorial para ensinar a utilizar a aplicação e os conceitos de computação gráficas existentes;
- c) melhorar a geração dos comandos JOGL, formatando os mesmos com cores diferentes e que os associem a suas respectivas peças, incluindo também comentários mouse *over* para os comandos;

- d) implementar outros conceitos de computação gráfica na aplicação, como iluminação, *splines*, polígonos, etc.;
- e) melhorar a experiência e usabilidade dos conceitos de computação gráfica já trabalhados pelo sistema, com base nas críticas do questionário de usabilidade;
- f) poder clicar com botão direito em cima de uma transformação para poder mudar o estado dela (ativada/desativada) e deixar num tom de cinza quando estiver desativada;
- g) alterar o painel Espaço Gráfico, para permitir selecionar a peça Câmera através do painel e movimentá-la, girá-la ou escalá-la, alterando suas propriedades sem a necessidade de informar seus valores no painel Propriedades da Peça;
- h) ter um ferramenta auxiliar (tipo uma calculadora) para demonstrar operações entre matrizes e criar um tutorial que ensine como elas funcionam;
- i) mostrar os valores das matrizes `GL_PROJECTION` e `GL_MODELVIEW` (matrizes do OpenGL). Assim o aluno poderia ver que os valores destas matrizes foram alterados. Também criar/similar uma pilha de matrizes para colocar as novas matrizes decorrentes do uso dos comandos `glPushMatrix` e `glPopMatrix` (objetos gráficos);
- j) incluir no exercício objetos alteráveis e não alteráveis (colocar uma marca de cadeado). Assim o professor faria um exercício para os alunos, protegendo os objetos desejados. Ex.: (colocar uma peça Cubo com propriedades bloqueadas, forçando o aluno a utilizar uma combinação de transformações para fazer uma rotação em torno do seu centro - trabalhando melhor com o conceito de transformações geométricas);
- k) rever usabilidade das janelas, permitir esconder ou expandir janelas (a lista de peças não precisa ser visualizada de início, devia ser opcional, alguns alunos sugeriram remove-la);
- l) colocar as opções dos painéis Arquivo e Ajuda em botões no topo do aplicativo e remover os dois painéis;
- m) tornar o painel Propriedade da Peça um painel flutuante;
- n) incluir novos jogos, ou alterar o jogo de encaixe de formas geométricas, para aumentar o desafio e motivação do uso da aplicação ou para trabalhar melhor com os conceitos computação gráfica da aplicação, como um quebra cabeças por exemplo;

- o) modificar o painel Fábrica de Peças para utilizar *canvas* no lugar de WebGL, a fim de melhorar a performance da aplicação;
- p) aplicar tecnologias e bibliotecas mais avançadas para controle dos elementos HTML da aplicação, como o React e o AngularJS.

REFERÊNCIAS

- ARAÚJO, Luciana P. de. **AduboGL**: aplicação didática usando a biblioteca OpenGL. 2012. 78 f. Trabalho de conclusão de curso (Bacharelado em Ciência da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.
- BATTAIOLA, André L. et al. Desenvolvimento de um software educacional com base em conceitos de jogos de computador. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 13., 2002, São Leopoldo. **Anais...** São Leopoldo: SBC, 2002. p. 282-290. Disponível em: <<http://www.br-ie.org/pub/index.php/sbie/article/view/189/175>>. Acesso em: 26 fev. 2014.
- BEHAR, Patricia A. **Modelos Pedagógicos em Educação a Distância**. Porto Alegre: Artmed, 2009.
- COSTA, Renata L. da. et al. Internet e laboratório de informática: dois importantes recursos metodológicos para surpreender os estudantes e beneficiar a interdisciplinaridade. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 20., 2009, Florianópolis. **Anais...** Florianópolis: SBC, 2009. Não paginado. Disponível em: <<http://www.br-ie.org/pub/index.php/sbie/article/view/1139/1042>>. Acesso em: 26 fev. 2014.
- COX, Kenia K. **Informática na educação escolar**. São Paulo: Autores Associados, 2003.
- DYNAMICCDRIVE. **Dinamic Drive DHTML Scripts**. [S.I.], 2012. Disponível em: <<http://www.dynamicdrive.com/dynamicindex1/navigate1.htm>>. Acesso em: 29 maio 2014.
- ELO7. **Jogo de Encaixe**. [S.I.], 2008. Disponível em: <<http://www.elo7.com.br/jogo-de-encaixe-formas-geometricas/dp/3C1100>>. Acesso em: 29 maio 2014.
- FHTR. **Introduction to Three.js**. [S.I.], 2012. Disponível em: <<http://fhtr.org/BasicsOfThreeJS/>>. Acesso em: 26 fev. 2014.
- GONÇALVES, Flávio A. S.; CANESIN, Carlos A. **Java applets para um software educacional distribuído em eletrônica de potência**. Sba Controle & Automação, Campinas, v. 13, n. 3, set. 2002. Não paginado. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0103-17592002000300010&lng=pt&nrm=iso>. Acesso em: 26 fev. 2014.
- HTML5. **HTML5 Introduction**. [S.I.], 2006. Disponível em: <http://www.w3schools.com/html/html5_intro.asp>. Acesso em: 26 fev. 2014.
- JSCOLOR. **JSColor – JavaScript / HTML Color Picker**. [S.I.], 2008. Disponível em: <<http://jscolor.com/>>. Acesso em: 16 jun 2014.
- KHRONOS. **WebGL Specification**. [S.I.], 2013. Disponível em: <<http://www.khronos.org/webgl>>. Acesso em: 26 fev. 2014.
- LUCENA, Carlos; FUKS, Hugo. **Professores e aprendizes na Web**: a educação na era da Internet. Rio de Janeiro: Clube do Futuro, 2000.
- MORAN, José M. **Como utilizar a Internet na educação**. Ciência da Informação, Brasília, v. 26, n. 2, maio 1997. Não paginado. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0100-19651997000200006&lng=pt&nrm=iso>. Acesso em: 26 fev. 2014.

PRENSKY, Marc. **Digital game-based learning:** practical ideas for the application of digital game-learning. St. Paul: Paragon House, 2007.

SILVA, Carlos A. da. **Informática na educação.** 2000. 45 f. Trabalho de conclusão de curso (Pós-Graduação em Tecnologias em Desenvolvimento de Sistemas), Universidade do Contestado, Canoinhas.

STEP. **StarLogo TNG.** [S.I.], 2011. Disponível em: <<http://education.mit.edu/projects/starlogo-tng>>. Acesso em: 26 fev. 2014.

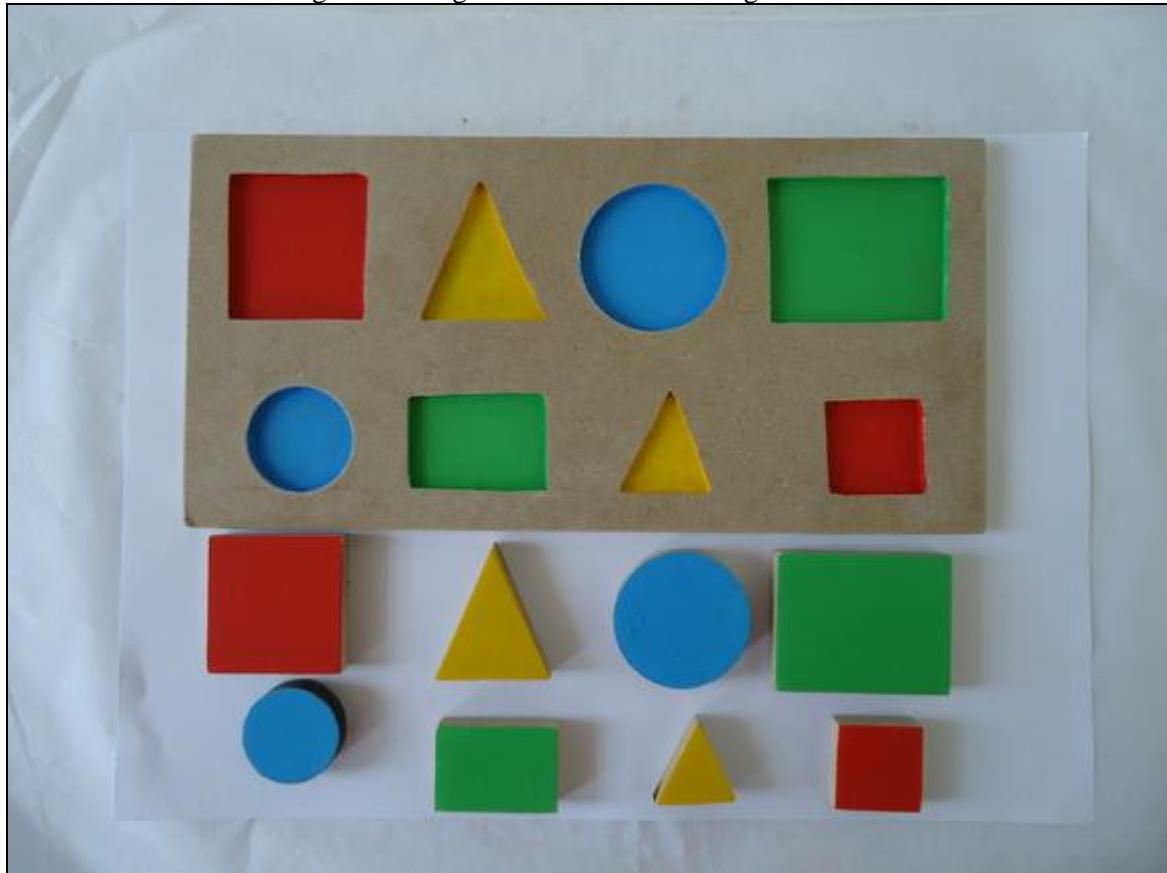
VALENTE, José A. **O computador na sociedade do conhecimento.** Campinas: Nied, 2002.

VISEDU-CG. **Visualizador de material educacional, módulo de computação gráfica.** Blumenau, 2013. Disponível em: <<http://gcb.inf.furb.br/visedu/cg/>>. Acesso em: 29 maio 2014.

APÊNDICE A – Peças disponíveis no VisEdu-CG

O encaixe das peças do sistema foi criado com base no jogo infantil de encaixe de formas geométricas, que pode ser visto na Figura 37.

Figura 37 - Jogo de encaixe de formas geométricas



Fonte: ELO7 (2008).

Cada tipo de peça encaixável possui um encaixe com uma forma geométrica distinta. O painel fábrica permite a criação de 6 peças diferentes:

- a) Câmera, que pode ser visualizada na Figura 38;
- b) Objeto Gráfico, que pode ser visualizada na Figura 39;
- c) Cubo, que pode ser visualizada na Figura 40;
- d) Transladar, que pode ser visualizada na Figura 41;
- e) Rotacionar, que pode ser visualizada na Figura 42;
- f) Escalar, que pode ser visualizada na Figura 43.

Figura 38- Peça Câmera



Figura 39- Peça Objeto Gráfico



Figura 40- Peça Cubo



Figura 41- Peça Transladar



Figura 42- Peça Rotacionar

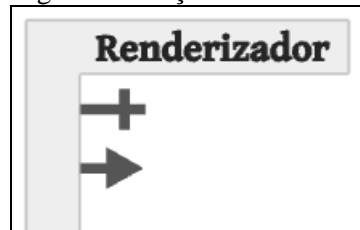


Figura 43- Peça Escalar



Todas as peças fabricadas deverão ser encaixadas na peça Renderizador (disponível no painel de montagem) ou em peças encaixadas nela. Na Figura 44 é exibida a peça Renderizador.

Figura 44- Peça Renderizador

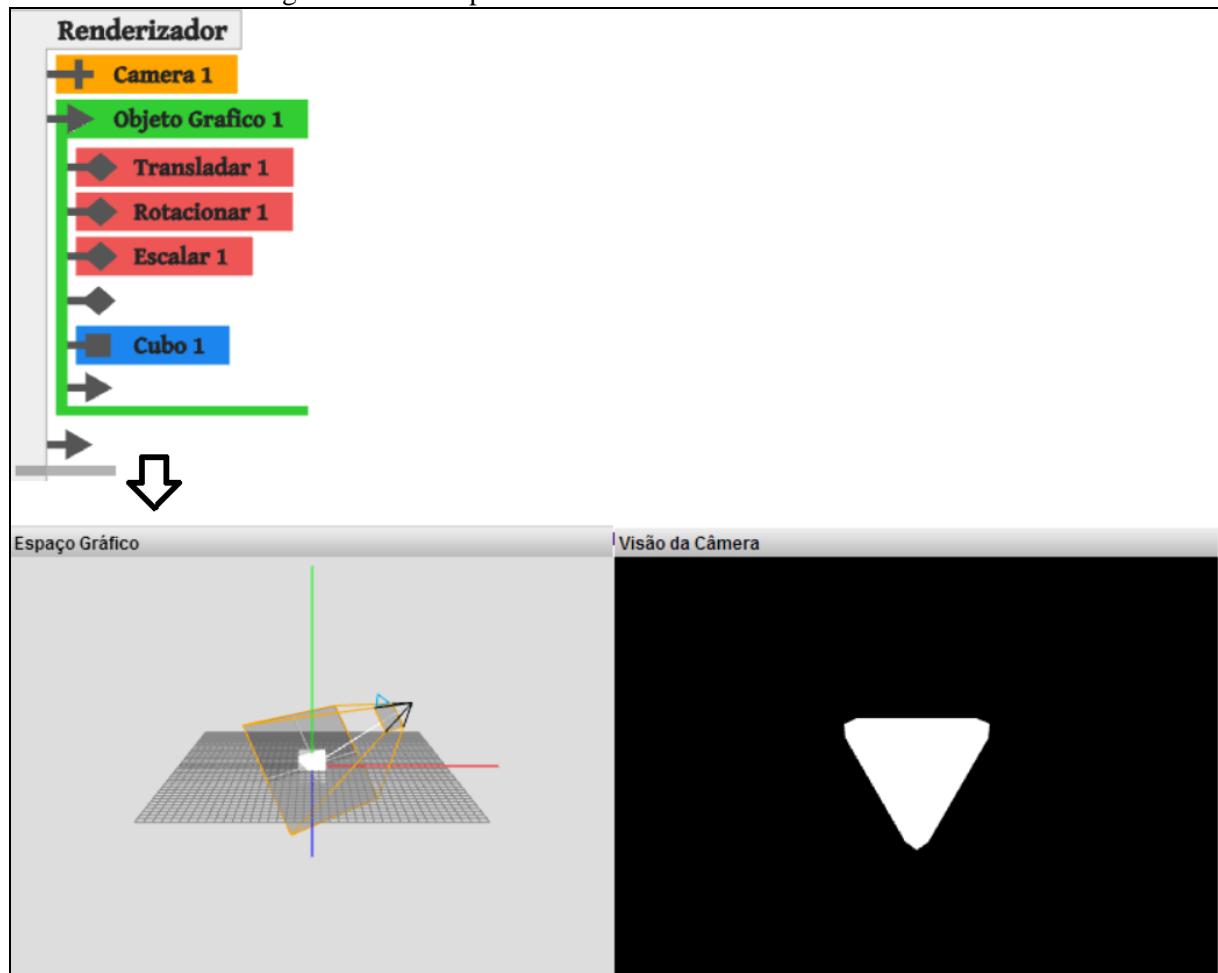


APÊNDICE B – Propriedades das peças disponíveis no VisEdu-CG e o impacto destas no conteúdo dos painéis Espaço Gráfico e Visão da Câmera

O painel Propriedades da Peça exibe todas as propriedades gráficas pertinentes a peça selecionada no editor. Ele permite que o usuário altere as propriedades de cada peça, afim de modificar o resultado gráfico do exercício exibido no painel Espaço Gráfico.

Na Figura 45 é apresentado um exercício construído no VisEdu-CG. Nele foram usadas todas as peças disponíveis na aplicação e cada peça possui os valores padrões definidos para cada propriedade. A seguir, essas propriedades serão alteradas para demonstrar o impacto de cada peça na criação do ambiente gráfico renderizado nos painéis Espaço Gráfico e Visão da Câmera.

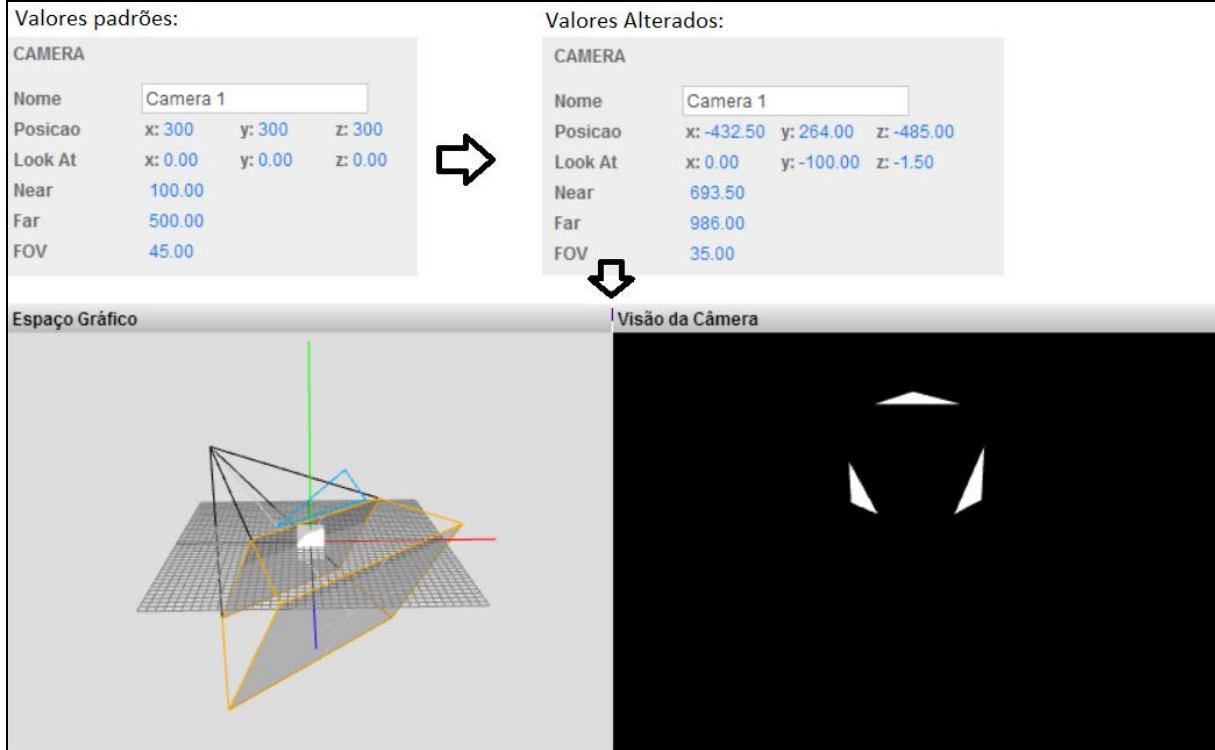
Figura 45 – Exemplo de exercício construído no VisEdu-CG



A Figura 46 demonstra como a peça Câmera influência no ambiente gráfico renderizado nos painéis Espaço Gráfico e Visão da Câmera. As suas propriedades padrões, demonstradas na Figura 45, foram alteradas, alterando também o espaço que é enxergado pelo

painel Visão da Câmera e a representação em forma de pirâmide que é feita desse espaço no painel Espaço Gráfico.

Figura 46 – Exemplo de alteração das propriedades da peça Câmera



A Figura 47 demonstra como a peça Objeto Gráfico influência no ambiente gráfico renderizado nos painéis Espaço Gráfico e Visão da Câmera. As suas propriedades padrões, demonstradas na Figura 45, foram alteradas, alterando também a visualização do cubo exibido nesses painéis.

A Figura 48 demonstra como a peça Transladar influência no ambiente gráfico renderizado nos painéis Espaço Gráfico e Visão da Câmera. As suas propriedades padrões, demonstradas na Figura 45, foram alteradas, alterando também a visualização do cubo exibido nesses painéis.

A Figura 49 demonstra como a peça Rotacionar influência no ambiente gráfico renderizado nos painéis Espaço Gráfico e Visão da Câmera. As suas propriedades padrões, demonstradas na Figura 45, foram alteradas, alterando também a visualização do cubo exibido nesses painéis.

A Figura 50 como a peça Escalar influência no ambiente gráfico renderizado nos painéis Espaço Gráfico e Visão da Câmera. As suas propriedades padrões, demonstradas na Figura 45, foram alteradas, alterando também a visualização do cubo exibido nesses painéis.

Figura 47 – Exemplo de alteração das propriedades da peça Objeto Gráfico

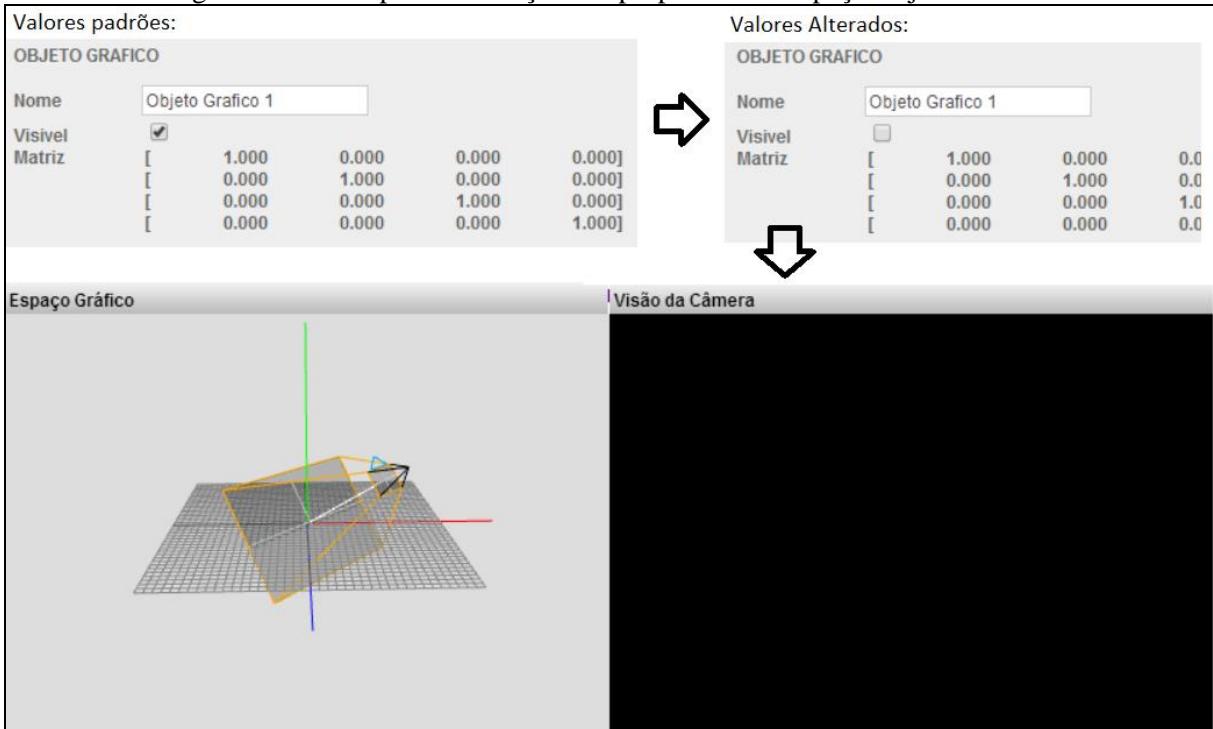


Figura 48 – Exemplo de alteração das propriedades da peça Transladar

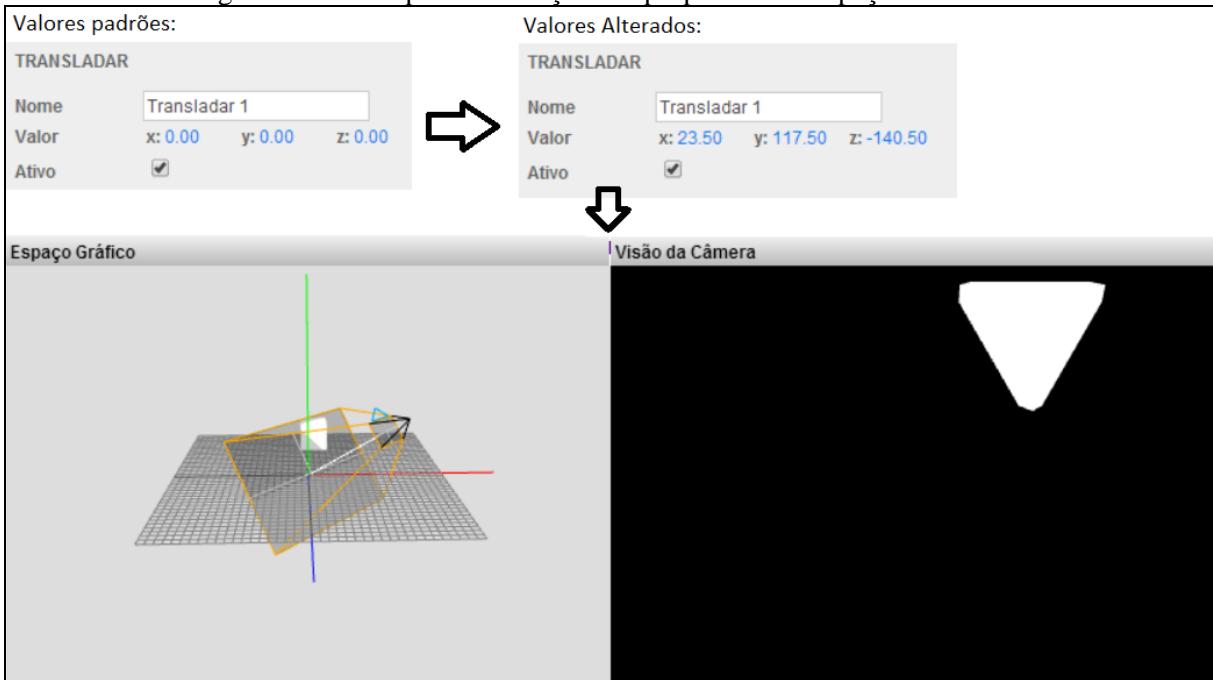


Figura 49 – Exemplo de alteração das propriedades da peça Rotacionar

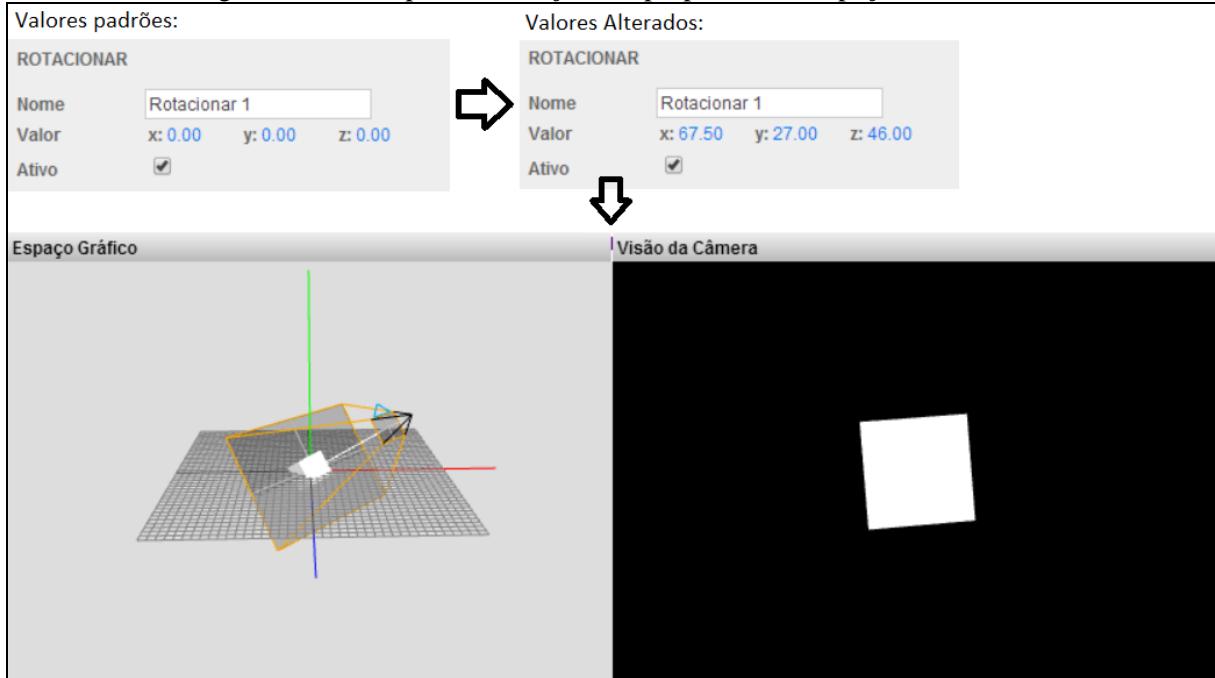
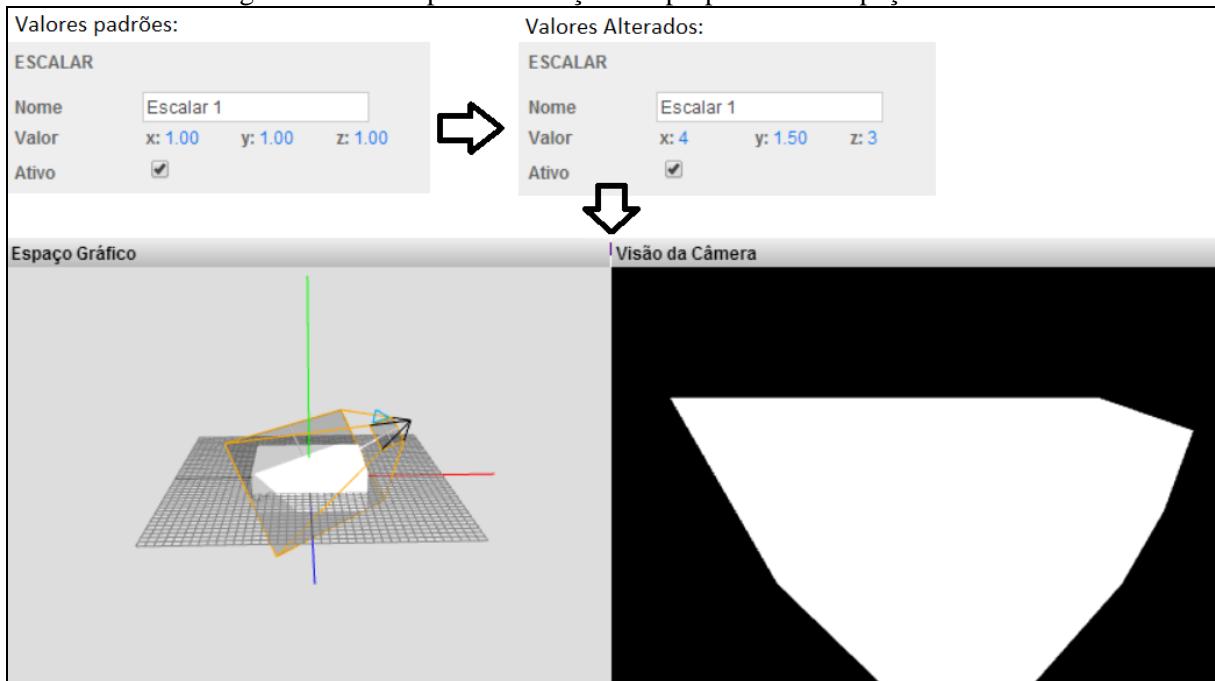
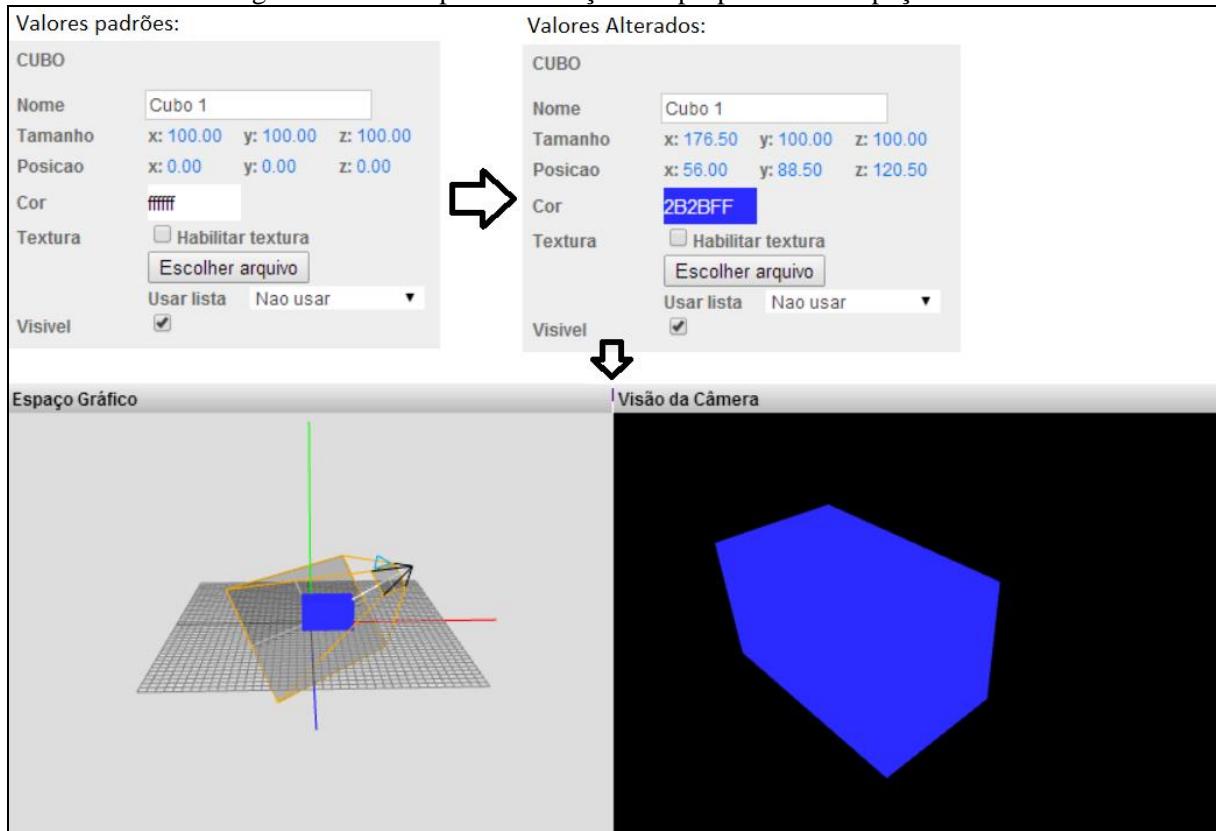


Figura 50 – Exemplo de alteração das propriedades da peça Escalar



A Figura 51 como a peça Cubo influencia no ambiente gráfico renderizado nos painéis Espaço Gráfico e Visão da Câmera. As suas propriedades padrões, demonstradas na Figura 45, foram alteradas, alterando também a visualização do cubo exibido nesses painéis.

Figura 51 – Exemplo de alteração das propriedades da peça Cubo



APÊNDICE C – Comandos JOGL gerados pelas peças do VisEdu-CG

O painel Comandos JOGL exibe o código fonte em Java que é necessário para reproduzir, no JOGL, o resultado gráfico ou conceito gráfico da peça selecionada. Adiante, serão apresentados os códigos gerados para cada tipo de peça.

No Quadro 29 pode-se visualizar os comandos JOGL que são gerados para a peça Objeto Gráfico. Essa peça, no JOGL, é representada pelo empilhamento e desempilhamento da matriz de transformação do ambiente gráfico.

Quadro 29 – Comandos JOGL gerados para a peça Objeto Gráfico

```
//Objeto Grafico 1
gl.glPushMatrix();
gl.glPopMatrix();
```

Dentro do empilhamento da matriz transformação (objeto gráfico) é que são inseridos as transformações geométricas e o desenho da forma geométrica do objeto gráfico. No Quadro 30 é apresentado o código que cria uma translação na matriz de transformação empilhada. Esse código, na aplicação, é gerado pela peça Transladar e foi gerado com os valores padrões de translação, ou seja, 0 (zero) para todos os eixos.

Quadro 30 – Comandos JOGL gerados para a peça Transladar

```
//Transladar 1
matrix1 = new Matrix4();
matrix1.makeTranslation( 0f, 0f, 0f );
gl.glMultMatrixd(matrix1.getDate(), 0);
```

O Quadro 31 apresenta o código que cria uma rotação na matriz de transformação empilhada. Esse código, na aplicação, é gerado pela peça Rotacionar e foi gerado com os valores padrões de rotação, ou seja, 0 (zero) para todos os eixos.

Quadro 31 – Comandos JOGL gerados para a peça Rotacionar

```
//Rotacionar 1
matrix1 = new Matrix4();
matrix2 = new Matrix4();
matrix3 = new Matrix4();
matrix1.makeXRotation( 0f );
matrix2.makeYRotation( 0f );
matrix3.makeZRotation( 0f );
gl.glMultMatrixd(matrix1.getDate(), 0);
gl.glMultMatrixd(matrix2.getDate(), 0);
gl.glMultMatrixd(matrix3.getDate(), 0);
```

O Quadro 32 apresenta o código que cria uma escala na matriz de transformação empilhada. Esse código, na aplicação, é gerado pela peça Escalar e foi gerado com os valores padrões de escala, ou seja, 1 (um) para todos os eixos.

Quadro 32 – Comandos JOGL gerados para a peça Escalar

```
//Escalar 1
matrix1 = new Matrix4();
matrix1.makeScale( 1f, 1f, 1f );
gl.glMultMatrixd(matrix1.getDate(), 0);
```

O código que constrói a forma geométrica do objeto gráfico pode ser visualizado no Quadro 34. Esse código, na aplicação, gera a forma geométrica de um cubo e é gerado pela peça `Cubo`. Ele foi gerado com os valores padrões das propriedades da peça. No quadro, foram destacados em vermelho os trechos de código que são gerados apenas para peças com a propriedade `Habilitar Textura` habilitada.

No Quadro 33 pode-se visualizar o código JOGL gerado para a peça `Câmera`. O código foi gerado com os valores padrões das propriedades da peça e exibe os comandos necessários para a inclusão da uma câmera do tipo perspectiva no JOGL.

Quadro 33 – Comandos JOGL gerados para a peça Câmera

```
float h = (float) height / (float) width; //altura/largura da tela

gl.glMatrixMode(GL.GL_PROJECTION);
gl.glLoadIdentity();

//gl.glFrustum(-10.0f, 10.0f, -h, h, 5.0f, 60.0f);
glu.gluPerspective( 45 , h, 100, 500);

gl.glMatrixMode(GL.GL_MODELVIEW);
gl.glLoadIdentity();

float posicaoCamera[] = { 300, 300, 300 };
float lookAtCamera[] = { 0, 0, 0 };
glu.gluLookAt( posicaoCamera[0], posicaoCamera[1], posicaoCamera[2],
    lookAtCamera[0], lookAtCamera[1], lookAtCamera[2], 0, 1, 0);
```

Ao selecionar a peça `Renderizador`, é gerado o código fonte necessário para reproduzir, no JOGL, o cenário visualizado pelo painel `Visão` da `Câmera`. Esse código gerado contém a estrutura completa de uma classe Java com o método `main`. Essa classe pode ser copiada para um projeto configurado com a biblioteca JOGL e ser executada. Caso exista no exercício alguma peça do tipo `Cubo` com a propriedade `Habilitar Textura` habilitada, a classe gerada precisará ser ajustada manualmente com o diretório e nome local das texturas dentro do projeto.

Quadro 34 – Comandos JOGL gerados para a peça Cubo

01	float xMax, xMin, yMax, yMin, zMax,	54	gl.glNormal3f(-1, 0, 0);
02	zMin;	55	gl.glTexCoord2f(0.0f, 0.0f);
03		56	gl glVertex3f(xMin, yMax, zMax);
04	private IntBuffer idsTextura;	57	gl.glTexCoord2f(1.0f, 0.0f);
05		58	gl glVertex3f(xMin, yMax, zMin);
06	//Cubo 1	59	gl.glTexCoord2f(1.0f, 1.0f);
07	gl.glShadeModel(GL.GL_FLAT);	60	gl glVertex3f(xMin, yMin, zMin);
08	gl.glNormal3f(0.0f, 0.0f, 1.0f);	61	gl.glTexCoord2f(0.0f, 1.0f);
09	gl.glMaterialfv(GL.GL_FRONT,	62	gl glVertex3f(xMin, yMin, zMax);
10	GL.GL_AMBIENT_AND_DIFFUSE, cor, 0);	63	gl glEnd();
11	gl glColor3f(1f, 1f, 1f);	64	
12		65	// Face lateral direita
13	xMax = 50f;	66	gl glBegin(GL.GL_QUADS);
14	xMin = -50f;	67	gl.glNormal3f(1, 0, 0);
15	yMax = 50f;	68	gl.glTexCoord2f(1.0f, 0.0f);
16	yMin = -50f;	69	gl glVertex3f(xMax, yMax, zMax);
17	zMax = 50f;	70	gl.glTexCoord2f(1.0f, 1.0f);
18	zMin = -50f;	71	gl glVertex3f(xMax, yMin, zMax);
19		72	gl.glTexCoord2f(0.0f, 1.0f);
20	gl glBindTexture(GL.GL_TEXTURE_2D,	73	gl glVertex3f(xMax, yMin, zMin);
21	idsTextura.get(-1)); //Posiciona	74	gl.glTexCoord2f(0.0f, 0.0f);
22	na Textura 3	75	gl glVertex3f(xMax, yMax, zMin);
23	gl glEnable(GL.GL_TEXTURE_2D); //	76	gl glEnd();
24	Habilita uso de textura	77	
25		78	// Face superior
26	// Face frontal	79	gl glBegin(GL.GL_QUADS);
27	gl glBegin(GL.GL_QUADS);	80	gl.glNormal3f(0, 1, 0);
28	gl.glNormal3f(0, 0, 1);	81	gl.glTexCoord2f(0.0f, 1.0f);
29	gl.glTexCoord2f(0.0f, 1.0f);	82	gl glVertex3f(xMin, yMax, zMin);
30	gl glVertex3f(xMax, yMax, zMax);	83	gl.glTexCoord2f(0.0f, 0.0f);
31	gl.glTexCoord2f(1.0f, 1.0f);	84	gl glVertex3f(xMin, yMax, zMax);
32	gl glVertex3f(xMin, yMax, zMax);	85	gl.glTexCoord2f(1.0f, 0.0f);
33	gl.glTexCoord2f(1.0f, 0.0f);	86	gl glVertex3f(xMax, yMax, zMax);
34	gl glVertex3f(xMin, yMin, zMax);	87	gl.glTexCoord2f(1.0f, 1.0f);
35	gl.glTexCoord2f(0.0f, 0.0f);	88	gl glVertex3f(xMax, yMax, zMin);
36	gl glVertex3f(xMax, yMin, zMax);	89	gl glEnd();
37	gl glEnd();	90	
38		91	// Face inferior
39	// Face posterior	92	gl glBegin(GL.GL_QUADS);
40	gl glBegin(GL.GL_QUADS);	93	gl.glNormal3f(0, -1, 0);
41	gl.glNormal3f(0, 0, -1);	94	gl.glTexCoord2f(1.0f, 1.0f);
42	gl.glTexCoord2f(1.0f, 0.0f);	95	gl glVertex3f(xMin, yMin, zMin);
43	gl glVertex3f(xMax, yMax, zMin);	96	gl.glTexCoord2f(0.0f, 1.0f);
44	gl.glTexCoord2f(1.0f, 1.0f);	97	gl glVertex3f(xMax, yMin, zMin);
45	gl glVertex3f(xMax, yMin, zMin);	98	gl.glTexCoord2f(0.0f, 0.0f);
46	gl.glTexCoord2f(0.0f, 1.0f);	99	gl glVertex3f(xMax, yMin, zMax);
47	gl glVertex3f(xMin, yMin, zMin);	100	gl.glTexCoord2f(1.0f, 0.0f);
48	gl.glTexCoord2f(0.0f, 0.0f);	101	gl glVertex3f(xMin, yMin, zMax);
49	gl glVertex3f(xMin, yMax, zMin);	102	gl glEnd();
50	gl glEnd();	103	
51		104	gl.glDisable(GL.GL_TEXTURE_2D); //
52	// Face lateral esquerda	105	Desabilita uso de textura
53	gl glBegin(GL.GL_QUADS);	106	

APÊNDICE D – Exercício no VisEdu-CG aplicado em sala de aula

Nas figuras Figura 52, Figura 53 e Figura 54 são exibidas as três páginas do exercício aplicado em sala de aula para que os alunos utilizassem e avaliasem a aplicação (ver seção 3.4.2).

Figura 52 – Primeira página do exercício criado para o VisEdu-CG

FURB – Universidade Regional de Blumenau
DSC – Departamento de Sistemas e Computação
Grupo de Pesquisa em Computação Gráfica, Processamento de Imagens e Entretenimento Digital
Disciplina: Computação Gráfica - prof. Dalton Solano dos Reis

Unidade 04 - Conceitos básicos de 3D

Informações utilizadas no exercício:

- aonde encontrar o VisEdu-CG: gcg.inf.furb.br/visedu ou www.inf.furb.br/gcg/visedu
- os arquivos das respostas devem ser zipados e salvos no AVA em VisEdu-CG
- cada arquivo de resposta deve ser nomeado seguindo este padrão cg4r_n.txt, aonde “n” corresponde ao número do exercício
- uma cópia dos arquivos utilizados como “entrada” nestes exercícios (se for o caso) estão em: <http://gcg.inf.furb.br/visedu/cg/exercicios/>
- clicando-se duas vez na interface é possível ter uma ajuda em relação ao contexto aonde se encontra a posição do mouse
- um vídeo com um exemplo simples de utilização pode ser visto em <http://www.youtube.com/watch?v=VIO0mYxtSys>
- Após terminar o exercício favor responder o questionário: <https://docs.google.com/forms/d/133mxQCax35kmpdXNdoH9SZ7zzMrmlPE7Y5hidDIX7a8/viewform?pli=1>

1) Crie uma cena utilizando o VisEdu-CG que **somente** contenha as peças “Camera”, “Objeto Gráfico” e “Cubo” (com textura “Logo Grupo CG”). Após altere as “Propriedades da Peça” da peça “Camera 1” no “Renderizador” para mudar o ponto de vista do observador na cena para visualizar o objeto “Cubo 1” deixando-o mais próximo possível ao exibido na imagem abaixo. Neste caso utilize os valores de “Near”, “Far” e “FOV” descritos nas figuras abaixo e mude os valores de “Posicao” e “Look At”.

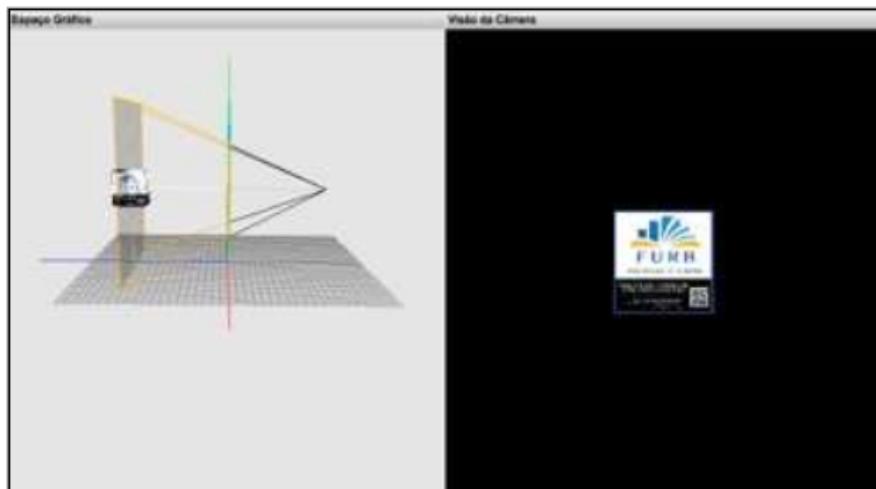
Arquivo	Fábrica de Peças	Propriedades da Peça
CAMERA		
Nome	Camera 1	
Posicao		
Look At		
Near	10	
Far	300	
FOV	45.00	

Figura 53 – Segunda página do exercício criado para o VisEdu-CG



FURB – Universidade Regional de Blumenau
 DSC – Departamento de Sistemas e Computação
 Grupo de Pesquisa em Computação Gráfica, Processamento de Imagens e Entretenimento Digital
 Disciplina: Computação Gráfica - prof. Dalton Solano dos Reis

- 2) Use o arquivo “CG-04_exer_02.txt” (entrada) disponível no VisEdu-CG em “Arquivo / Abrir / Lista de Exemplos” e modifique **somente** os valores do objeto “Camera 1” para que o “Near” tenha o seu plano coincidindo com o plano Z igual a zero, e “Far” para que o objeto “Cubo 1” fique visível dentro do Frustum só a metade conforme figura abaixo.



- 3) Crie uma cena utilizando o VisEdu-CG que **somente** contenha as peças descritas na figura abaixo para que tem as mesmas texturas e representação visual. No caso os tamanhos dos objetos “cubos” foram mantidos com 100 unidades e todos os “cubos” estão posicionados com a face de baixo com valor no plano “zero” do eixo Z.

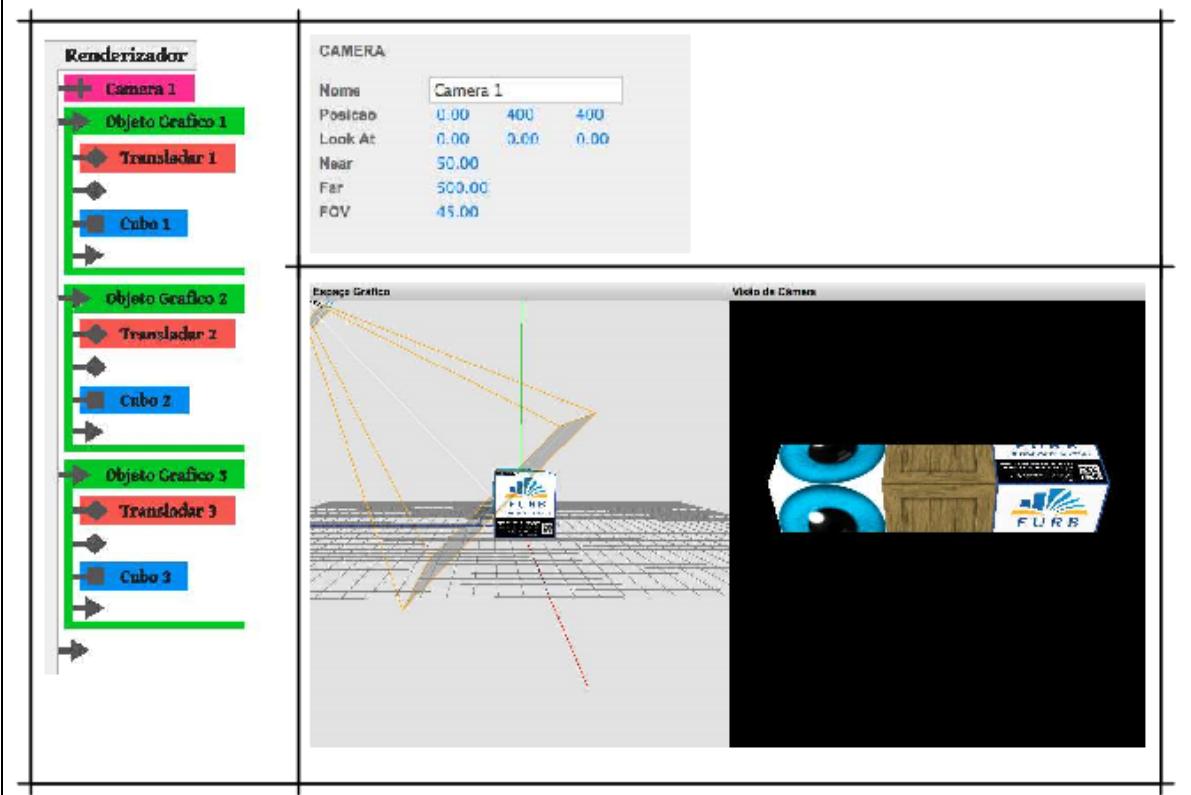
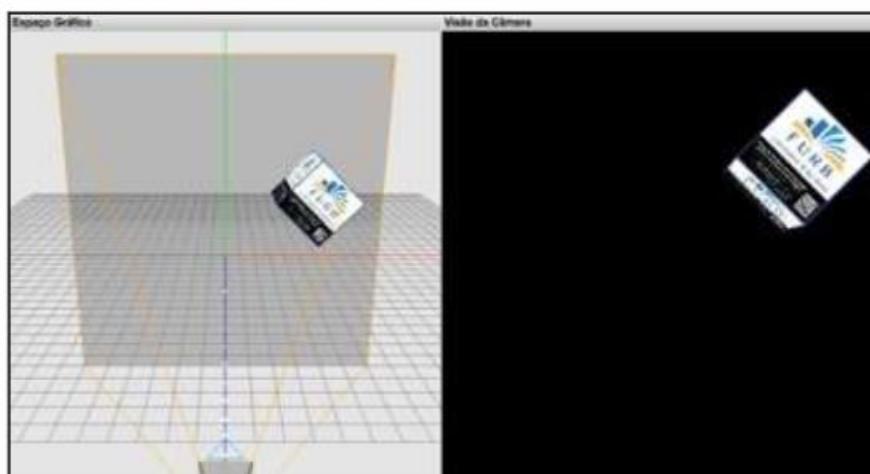


Figura 54 – Terceira página do exercício criado para o VisEdu-CG



FURB – Universidade Regional de Blumenau
 DSC – Departamento de Sistemas e Computação
 Grupo de Pesquisa em Computação Gráfica, Processamento de Imagens e Entretenimento Digital
 Disciplina: Computação Gráfica - prof. Dalton Solano dos Reis

- 4) Use o arquivo “CG-04_exer_04.txt” (entrada) disponível no VisEdu-CG em “Arquivo / Abrir / Lista de Exemplos” e **somente** usando as peças de transformação geométrica (Transladar / Rotacionar / Escalar) da “Fábrica de Peças” para que o “Objeto Grafico 1” tenha uma matriz de transformação global e rotacione o objeto “Cubo 1” 45 graus sentido horário em torno do seu centro conforme figura abaixo.



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



LEMBRE de responder o questionário: <https://docs.google.com/forms/d/133mxQCax35kmpdXNdoH9SZ7zzMrmIPE7Y5hidDIX7a8/viewform?pli=1>

APÊNDICE E – Respostas descritivas do questionário de usabilidade do VisEdu-CG

Nos quadros Quadro 35, Quadro 36 e Quadro 37 são apresentadas as respostas das questões 10, 11 e 12 do questionário de usabilidade do sistema (ver seção 3.4.2).

Quadro 35 – Respostas da 10^a questão

Com base nas respostas anteriores, quais seriam os principais "Pontos Positivos" no uso do aplicativo?	
1	Fácil de utilizar, e facil de enteder o que faz cada propriedade.
2	A organização e simplicidade dos temas trabalhados.
3	Usabilidade simples Facilidade no entendimento dos conceitos
4	Intuitivo e fácil utilização.
5	Renderizador: eu achei fantástica a estrutura do renderizador do renderizador.
6	
7	Simples de utilizar. O uso de drag/drop, a forma dos objetos, etc torna mais intuitivo e dinâmico o aprendizado e uso. A ferramenta atende bem o que se propõe a fazer e funciona bem dentro do seu escopo.
8	É fácil entender como cada objeto se relaciona com as transformações em função dos encaixes. o código é organizado e separado em bloco de ações.
9	O Aplicativo auxilia muito na formação da lógica de programação de Computação Gráfica. Possibilitando a visualização do resultado das alterações no mesmo instante.
10	Fácil de usar; Maior clareza para entender os conceitos de cg.
11	É fácil de entender, simples e objetivo.
12	Fácil utilização e simplicidade
13	Facilidade de uso
14	É simples de usar, permite obter a intuição necessária para os primeiros contatos com computação gráfica.
15	O aplicativo consegue, com uma interface intuitiva e prática, mostrar diversos elementos que podem ser usados para compor uma cena.
16	Intuitivo e agradável de utilizar
17	Um usuário que não é da área de computação tem capacidade para manipular o aplicativo, apenas conhecendo o plano cartesiano e conceitos matemáticos.
18	Facilidade no uso. Grande ajuda no entendimento do que é e para que serve cada objeto gráfico. Ótima ajuda no uso de JOGL.
19	
20	Usabilidade intuitiva Código gerado fácil de entender
21	Web, e com conceitos básicos e medianos de computação gráfica
22	bem tranquilo de entender, mesmo nunca tendo trabalhado com ele não preciso da janela de ajuda.
23	este aplicativo facilita a aplicação de conceitos de computação gráfica de forma simples e rápida, possui uma boa interface simples e intuitiva
24	rápido e prático no desenvolvimento de exemplos, atividades
25	Simples e fácil de utilizar
26	Ajuda a entender os conceitos de Computação Gráfica mais rápido e de forma simples.
27	A visualização do espaço gráfico e a forma visual de criação dos objetos da cena, que permite um melhor entendimento do papel de cada item na cena. Também vale comentar a possibilidade de visualizar o código JOGL que pode ajudar quando há dúvida em implementações (ocorreu comigo).
28	O aplicativo demonstra bem uma estrutura de grafo de cena, bem como permite a visualização em tempo real do que está sendo feito.

Quadro 36 – Respostas da 11^a questão

Com base nas respostas anteriores, quais seriam os principais "Pontos Negativos" no uso do aplicativo?	
1	Pouco intuitivo em algumas partes.

continua

continuação

2	A disposição das janelas poderiam ser modifica para que fosse dada enfase para as janelas mais utilizadas, como exemplo a Lista de peças onde não é muito utilizada e poderia ter seu tamanho reduzido.
3	Achei um pouco difícil manipular o ponto de vista na tela de Espaço Gráfico usando o botão esquerdo do mouse e arrastando. Os outros atalhos (botão direito e botão de rolagem) estão bons.
4	É muito lento a seleção dos objetos.
5	As vezes ocorrem erros inesperados, de acordo com as versões do Chrome.
6	
7	Não há pontos negativos.
8	Usabilidade pode ser melhorada.
9	Não vejo nenhum ponto negativo de GRANDE relevância.
10	A interface precisa ser melhorada. Ocorrem bugs na parte visual da ferramenta.
11	apenas o click no espaço gráfico, é muito sensível
12	apenas o tamanho da janela em que os comandos JOGL ficam
13	As janelas mais importantes possuem o mesmo tamanho de janelas pouco utilizadas. Espaço Gráfico e Visão da Câmera foram utilizadas durante todo o exercício. A janela de JOGL não pareceu muito relevante no momento. A Lista de Peças só reparei na hora de preencher este questionário.
14	Não encontrei pontos negativos
15	Talvez por ainda ser uma versão em desenvolvimento, ocorrem alguns erros de utilização que comprometem um pouco da experiência. Em geral, quem é da área de computação consegue desvair destes "problemas", mas podem existir complicações com pessoas mais leigas.
16	Não tem comentários explicativos de blocos. É pouco intuitivo saber por onde começar ou como começar a manipular os objetos. Somente após um conhecimento inicial é que sabemos o que devemos fazer.
17	Não encontrado
18	Alguns problemas com resolução e tamanho dos componentes.
19	
20	Para o que se propõe, não vi pontos negativos na ferramenta.
21	Na parte de scroll das telas não funciona com o mouse. A rotação da cena é muito precária.
22	O comportamento da aplicação é um tanto estranho, as vezes alterando entre a Fábrica e as Propriedades sem querer. Além de duplo click abrir a ajuda? Seria mais interessante o duplo click faz a alternância entre Fábrica e Propriedades.
23	A barra de rolagem do renderizador esta muito escondida. Pelo fato dela ser da mesma cor da interface prejudica um pouco no aprendizado.
24	Não seria necessário mostrar a todo momento o código gerado; Problemas com o navegador usado.
25	
26	Lento em alguns pontos, e a tela esta ficando preta se selecionamos um objeto e vamos para Fábrica de peças
27	Não encontrei uma função para remover os objetos, lentidão
28	

Quadro 37 – Respostas da 12ª questão

Com base nas respostas anteriores, quais seriam as suas "Sugestões" de melhoria para o aplicativo?

1	Ao criar um polígono é necessário selecionar no combo os pontos, o combo poderia estar acima dos valores.
2	Poderia haver um tutorial inicial. O código fonte poderia ter mais explicações de onde se encaixa com cada objeto. Poderia ter highlight o código Ter duas listas de peças talvez seja desnecessário.
3	JOGL: Acho que utilizar de cores (talvez cinza e preto) para destacar o que é da biblioteca OpenGL do código em java. Além disso por questão de didática talvez seja interessante os códigos que aparecem em java estarem em português. Não entendi a ideia da Janela "Lista de peças", pois já temos a listagem no renderizador. Talvez seja interessante passar o JOGL para a "Lista de peças" e fixar uma janela com as propriedades, e o local onde hoje é alternado entre propriedades e fábrica ficar só para fabrica. As outras opções fazer um menu como um context menu no canto superior esquerdo.
4	Poder omitir janelas que sejam relevantes para o que a pessoa está fazendo. Adição de um tutorial guiado no estilo do Facebook quando são adicionadas novas funcionalidades. Evitar disponibilizar a versão desenvolvimento, deixando sempre a versão mais estável para uso.

continua

continuação

5	Possibilidade de 'esconder' janelas que não são relevantes. Em alguns momentos, gostaria de poder ter mais espaço para janelas de Espaço e Visão. Outra sugestão é a remoção da Lista de Peças. O renderizador já possui essa informação de forma comprehensível.
6	
7	Apenas melhorar a interação do usuário com os componentes, principalmente no momento da edição das propriedades.
8	Poderia organizar melhor as janelas, como por exemplo deixar a cargo do usuário quais as telas que ele quer ver.
9	
10	Corrigir os bugs que ocorrem com diferentes versões do Chrome, como por exemplo a câmera que as vezes não é possível selecionar as propriedades.
11	talvez diminuir o espaço da janela lista de peças e aumentar o espaço da janela comando JOGL
12	Visualização do código JOGL em pop-up, ou outra forma em que seja possível ver mais código sem ter que ficar mexendo nas barras de rolagem.
13	apenas o espaço gráfico
14	
15	acabei, sem querer, selecionando e arrastando o objeto renderizador e ao tentar colocar no lugar novamente invadi a janela superior. que era a fabrica de objetos. Ele foi selecionado para a exclusão (a lixeira), mas ele não se exclui e após isso o objeto avançava para a janela de cima toda a vez que rolava a barra para ver todos os objetos que coloquei. sugiro também permitir o scroll na janela de objetos que estou usando na cena.
16	Acredito que poderia ser revisto, quando seleciona o Cubo por exemplo e vai direto para a aba de Fábrica de peças a tela fica preta, tem que tirar a seleção do cubo e aperta na aba para ai sim funcionar.
17	
18	Melhorar a performance da seleção dos objetos. Se eu deixar um valor em branco, fica com o valor NaN, podia ser 0.
19	Enfrentei alguns bugs envolvendo o 'drag and drop' dos componentes. O evento deveria ser revisado.
20	
21	Correção dos bugs; Melhoria na interface grafica; Disposição das janelas;
22	A disposição das janelas poderia ser um pouco diferente, permitindo o usuário exibir ou ocultar as que deseja (estilo eclipse). A janela de visualização do código poderia ser uma aba da janela do grafo de cena. Além disso, a janela de grafo de cena poderia iniciar oculta, já que é possível "visualizar o grafo" pela janela de renderizador - ou ser uma aba dessa janela. Quanto a fábrica/ Abrir/ Exportar/ Help, creio que poderiam ser "botões" no topo da página, o que abriria mais espaço na tela. E para a edição das propriedades que cada componente, poderia ser lançado uma popup com dois cliques no componente. No que diz respeito a câmera, poderia ser criado o recurso para selecioná-la e transformá-la com o uso do mouse, permitindo mexer com o X, Y, Z e posicionamento. Afinal, esse é o recurso mais "chato" de configurar na mão.
23	
24	Correção da parte visual da ferramenta.
25	Adição do recurso de dicas (hints) nas propriedades dos objetos gráficos.
26	O botão exportar não funcionou no Firefox. Na janela onde se arrastam os componentes para formar a cena, o scroll do mouse não funciona. É necessário clicar na barra lateral e arrastar.
27	Melhorar a performance do aplicativo
28	A janela de Comandos JOGL poderia ter o textos formatados para ser mais fácil a leitura do código.

ANEXO A – Requisitos do AduboGL

O AduboGL deve:

- a) possuir algumas funções que executem comandos e conceitos da OpenGL (Requisito Funcional – RF);
- b) permitir que o usuário resolva exercícios de computação gráfica que utilizem o cubo e transformações da OpenGL (RF);
- c) disponibilizar peças para representar os comandos da OpenGL (RF);
- d) permitir o encaixe das peças conforme os comandos que as representam (RF);
- e) permitir o agrupamento de transformações sobre um ou mais objetos gráficos (RF);
- f) permitir que o usuário altere a cor do cubo a ser desenhado em 3D (RF);
- g) permitir que o usuário altere os valores x, y e z a serem passados para a matriz de transformação (RF);
- h) permitir que o usuário da aplicação visualize um cenário 3D a partir das peças colocadas (RF);
- i) permitir a visualização do código gerado pelas peças do exercício, ao clicá-la (RF);
- j) permitir que o usuário salve o exercício realizado (RF);
- k) permitir que o usuário carregue o exercício salvo para continuá-lo (RF);
- l) ser implementado na linguagem C++ (Requisito Não Funcional – RNF);
- m) utilizar a biblioteca OpenGL para realizar as funções referentes a computação gráfica (RNF);
- n) utilizar classes da biblioteca V-ART (RNF);
- o) ter uma padronização nos nomes das funções (RNF);
- p) ser de código aberto (RNF);
- q) ser documentado em Doxygen (RNF);
- r) ser desenvolvida no Microsoft Visual C++ 2010 Express (RNF);
- s) executar sobre o sistema operacional Windows e Mac OS (RNF). (ARAÚJO, 2012).