UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ – UTFPR
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

RENAN NOVELLI

MODELAGEM E INTERAÇÃO EM AMBIENTE 3D UTIZANDO BLENDER E UNITY

TRABALHO DE DIPLOMAÇÃO

MEDIANEIRA
2015

RENAN NOVELLI

MODELAGEM E INTERAÇÃO EM AMBIENTE 3D UTIZANDO BLENDER E UNITY

Trabalho de Diplomação apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas – COADS – da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Dr. Pedro Luiz de Paula Filho

MEDIANEIRA

2015

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria de Graduação e Educação Profissional
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

MODELAGEM E INTERAÇÃO EM AMBIENTE 3D UTIZANDO BLENDER E UNITY

Por

RENAN NOVELLI

Este Trabalho de Diplomação (TD) foi apresentado às 00:00 h do dia 00 de novembro de 2015 como requisito parcial para a obtenção do título de Tecnólogo no Curso Superior de Tecnologia em Manutenção Industrial, da Universidade Tecnológica Federal do Paraná, Câmpus Medianeira. O acadêmico foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado com louvor e mérito.

Prof. Pedro Luiz de Paula Filho
UTFPR – Câmpus Medianeira
(Orientador)

Prof. Me. Juliano Rodrigo Lamb
UTFPR – Câmpus Medianeira
(Convidado)

Prof. Dr. Evando Pessini
UTFPR – Câmpus Medianeira
(Convidado)

Prof. Me. Jorge Aikes Junior
UTFPR – Câmpus Medianeira
(Responsável pelas atividades de TCC)

Dedico este trabalho aos meus pais, que me incentivaram e me apoiaram sempre.

Ao meu irmão pelo carinho e apoio indispensáveis.

AGRADECIMENTOS

Agradeço a todos os professores do curso de Análise e Desenvolvimento de Sistemas, pela dedicação, comprometimento e ensinamentos passados durante as aulas, cada um contribuiu a sua maneira para que fosse possível minha formação acadêmica.

Especial agradecimento para o meu orientador Pedro Luiz de Paula Filho pela sua dedicação e orientação durante o trabalho.

Um agradecimento especial aos meus padrinhos que sempre me motivaram.

Por fim, gostaria de agradecer aos meus pais e irmão pelo carinho e apoio durante esta jornada.

RESUMO

NOVELLI, Renan. Modelagem e interação 3D utilizando Blender e Unity. 2015. 53 f. Trabalho de Diplomação (Tecnologia em Análise e Desenvolvimento de Sistemas), Universidade Tecnológica Federal do Paraná. Medianeira, 2015.

Este trabalho apresenta uma proposta de virtualização de ambientes, tornando possível caminhar, por exemplo, por uma instituição de ensino e aprender sobre fatos referentes a ela e informações sobre os cursos e professores entre outras possibilidades. Esse tipo de visitação virtual proporciona às pessoas experimentarem a instituição de forma mais próxima ao real. Para tanto foi criado um ambiente que permite a visitação à um bloco da UTFPR, câmpus Medianeira. Neste ambiente virtual, é possível a navegação e interação com os professores de Ciência da Computação apresentando informações referentes ao curso, disciplinas e pesquisas. Este projeto é baseado em um ambiente real, permitindo aos interessados no curso explorar o bloco sem a necessidade de estarem ali presencialmente. Focou-se a interação com os professores buscando as informações utilizando um banco de dados XML que possibilita um sistema dinâmico para inserção, alteração e remoção destes. Para a confecção deste projeto foram utilizadas as ferramentas Unity e Blender, que são ferramentas gratuitas, na qual a primeira é uma engine de desenvolvimento de jogos que unifica componentes, enquanto a segunda permite a modelagem de elementos 3D.Ambientes virtuais possibilitam uma maneira diferenciada de apresentar uma instituição permitindo criar um maior interesse de um possível futuro cliente ou aluno dependendo o contexto. Com o ambiente virtual possibilitou-se adquirir as informações com uma interação direta com os modelos dos professores simulando algo próximo do real.

Palavras Chave: Unity, Blender, Modelagem, XML, Ambiente Virtual.

ABSTRACT

NOVELLI, Renan. Modeling and Interaction in a 3D environment using Blender and Unity. 2015. 53 f. Trabalho de Diplomação (Tecnologia em Análise e Desenvolvimento de Sistemas), Universidade Tecnológica Federal do Paraná. Medianeira, 2015.

This paper present a proposal of virtualization environments, making possible walk, for example, through an educational institution to lean about facts related to it and information's about courses and teacher among others possibilities. This type of virtual visitation provides people experience the institution more closely to the real. Therefore was created an environment that allows a visitation to a block from UTFPR, campus Medianeira. In this virtual environment, it's possible a navigation and interaction with the teacher from Computer Science showing information's about the course, subjects and researches. This paper it's based in a real environment, allowing interested in the course explore the block without having to be there in person. It focuses on the interaction with teacher seeking information using a XML database which enables a dynamic system for adding, editing and removing. For making this project the tools Unity e Blender were used, which are free tools, where the first is a game development engine that unifies components, while the second allows the modeling of 3D elements.

Virtual Environments enable a differentiated way to introduce an institution allowing to create a greater interest in a possible future client or student depending on the context. With the virtual environment has enabled to acquire the information with a direct interaction with the teacher models simulating something close to real.

Key words: Unity, Blender, Modeling, XML, Virtual Environment.

LISTAS DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Devido ao avanço da tecnologia e as limitações referente a tempo e custo para fazer uma visitação presencial, futuros alunos e seus pais começaram a procurar mais do que imagens e vídeos dentro dos sites das instituições de ensino, ao invés procuravam uma experiência mais próxima do real.

Essa experiência pode ser apresentada utilizando um ambiente virtual, sendo esta uma ferramenta poderosa na busca de demonstrar à universidade e atrair estudantes para visitar uma instituição ou colocar ela no topo de sua lista de opções (ICEF 2012).

Nestes ambientes é possível caminhar pela instituição e aprender sobre fatos referentes ao câmpus, tamanho, informações sobre os cursos e professores entre outras possibilidades. Esse tipo de visitação virtual proporciona às pessoas experimentarem a instituição de forma mais próxima ao real enquanto aliviam custos que seriam destinados a uma visitação presencial (ICEF 2012).

Este trabalho apresenta uma proposta de virtualização de um bloco da UTFPR (L1), câmpus Medianeira. Neste ambiente virtual, é possível a navegação e interação com os professores do curso de Ciência da Computação apresentando informações referentes ao curso, disciplinas e pesquisas.

1.1 OBJETIVO GERAL

Desenvolver um ambiente para navegação e visitação do Bloco L1 da UTFPR utilizando Unity e Blender.

1.2 OBJETIVOS ESPECÍFICOS

- Realizar referencial teórico sobre ambientes virtuais.
- Realizar referencial teórico sobre game *engines*.
- Modelar equipamentos que constituem o Bloco L1.

- Modelar as salas que compõem o bloco.
- Programar interação com objetos do bloco.
- Programar Interação com professores.

1.3 JUSTIFICATIVA

Diferente de anos anteriores estudantes buscavam informações por meios tradicionais de comunicação como jornais, comerciais de televisão, entre outros., Atualmente eles estão buscando outros meios para explorar e obter informações sobre as universidades através da internet por meio do computador, tablets e celulares (CAPPEX 2012).

O ambiente virtual está apto a apresentar as informações de forma diferenciada para os usuários possibilitando assim uma experiência mais intuitiva ao pesquisar instituições de ensino. (ICEF 2012).

A escolha pela Unity se deve a sua grande quantidade de ferramentas disponíveis em sua versão gratuíta e principalmente devido a compatibilidade com a ferramenta Blender. As ferramentas foram escolhidas pois trabalham em conjunto de forma simples e eficiente.

Essa transparência na apresentação se torna possível com as novas ferramentas como Unity e Blender, tornando o ambiente virtual uma experiência única e interativa dentro deste meio.

2 REFERENCIAL TEÓRICO

Este Capítulo apresenta o referencial teórico do trabalho, que aborda ambientes virtuais e *game engines*.

2.1 AMBIENTE VIRTUAL

Ambiente Virtual pode ser definido como uma representação computacional de um mundo real ou imaginário nos quais usuários podem navegar ou interagir em tempo real com objetos (JANKOWSKI e HACHET 2014).

O ambiente interativo pode ser caracterizado em três tipos principais. Navegação - refere-se a capacidade de navegar pelo ambiente; Seleção ou Manipulação - refere-se a capacidade de escolher um objeto e especificar ações a ele; Controle do Sistema- refere-se a comunicação do usuário com o sistema que não faz parte do ambiente virtual em si (JANKOWSKI e HACHET 2014).

As aplicações dos ambientes virtuais 3D são inúmeras e podem ser usadas para diversas áreas como:

- Educação e Treinamento
- Visualização de Informações
- Computação Gráfica
- Jogos
- Turismo Virtual

Exemplificando um dos campos de aplicações de um ambiente virtual pode-se citar uma situação em que uma pessoa precisa aprender uma habilidade em específico. É muito aconselhável utilizar um ambiente 3D quando a situação necessita de um grande custo ou existe grande risco (DALGARNO e HEDBERG 2011).

2.2 GAME ENGINES

Game engines são conjuntos de componentes que proporcionam uma grande quantidade de recursos para fazer jogos. Esses componentes normalmente são integrados em um único ambiente de desenvolvimento para possibilitar um rápido desenvolvimento.

2.2.1 INPUTS

Um dos aspectos mais importantes de qualquer ambiente 3D é como será realizada a navegação, *game engines* normalmente suportam vários tipos de *inputs* a serem utilizados como: mouse, teclados, controles e aparelhos *touch*, entre outros mais incomuns como por exemplo a voz do usuário. Existem diversas maneiras de lidar com *inputs*, mas duas são bastante comuns: eventos e *polling* (ENGER 2013).

Os eventos funcionam com o computador esperando algum tipo de *input* (o apertar de um mouse, o apertar de uma tecla do teclado, pressionar o *touch*) acionando assim o seu código.

Polling normalmente serão utilizados quando se fala dos valores de cada posição, como coordenadas x/y do mouse. As *game engines* provém os meios para resgatar esses valores quando o desenvolvedor desejar e assim o desenvolvedor pode trabalhar com esses valores quando a pessoa se move no cenário ou quando o mouse muda de posição na tela (ENGER 2013).

2.2.2 GRÁFICOS

Um dos maiores atrativos de uma *game engine* é a capacidade que possuem para apresentar gráficos avançados diante da tecnologia disponível. Em geral o que limita a capacidade de um projeto é o meio em que ele será publicado, um aplicativo para celular ainda não consegue ter a mesma qualidade gráfica de um aplicativo lançado para computador. A Figura 1 apresenta uma imagem do jogo coreano Black Desert, um jogo de RPG para multijogadores que utiliza das técnicas mais avançadas para gerar gráficos de última geração no mercado. O jogo foi premiado como melhor do seu gênero em 2014 pela Gamescom (GAMESCOM 2014).

Figura 1 - Personagem do jogo Black Desert.

Fonte: BLACK DESERT (2015).

Produções 3D normalmente são criadas em torno de *assets* que são gerados em um renderizador 3D externo, como por exemplo, Blender e importados posteriormente para a *game engine*. *Assets* são os componentes presentes em um projeto como, por exemplo, arte, modelos 3D, efeitos de aúdio, plug-ins ou outros. Elas suportam uma grande quantidade de formatos para importação, possibilitando assim o desenvolvedor trabalhar com a ferramenta que mais lhe agrada.

É possível adicionar esse *asset* ao projeto que está sendo trabalhado e adicionar diversos efeitos de luz para dar vida ao objeto. Em geral as game *engines* proporcionam diversas ferramentas para deixar o trabalho o mais simples e intuitivo possível. Toda essa simplicidade combinada com a possibilidade de criar terrenos, efeitos de partículas faz ser possível criar um mundo completo utilizando uma *engine* como essa em conjunto com *assets* de várias fontes.

2.2.3 SOM

Os efeitos sonoros normalmente não saem diretamente dos dispositivos de áudio utilizados pelos usuários, a maioria das game *engines* possuem os meios para

colocar sons dentro do ambiente virtual que podem se alterar dependendo das condições do personagem diante deste ambiente.

Na vida real os sons são emitidos por objetos e escutados pelos ouvintes. O modo que o som será percebido depende de vários fatores. Um ouvinte pode dizer de qual direção um som está vindo e também ter uma noção da distância.

Para simular efeitos de posição, *as game engines* precisam de uma origem do som. O som será emitido e então recebidos pela *game engine* e atribuído para um outro objeto, como por exemplo o próprio jogador. *Game engines* podem simular os efeitos da origem do som, distância e posição de acordo com o ouvinte e alocar de acordo para simular o efeito desejado. Também é possível alterar a velocidade do som para adicionar maior realismo dependendo do tipo de origem (UNITY 2015).

As músicas e sons de interface são um caso a parte, pois não precisam respeitar uma posição do personagem diante do ambiente virtual, mas claro que ajustes são necessários para adequar esses sons ao clima do ambiente virtual.

2.2.4 REDE

Game engines mais modernas provêm uma alta quantidade de componentes previamente feitos e *scripts* que fazem boa parte do trabalho, possibilitando assim trabalhar com modos multijogadores para suas aplicações.

2.2.5 FÍSICA

A física e a modelagem 3D não são uma parte integral de um ambiente 3D. Para ter comportamento físico convincente, um objeto dentro de um ambiente virtual deve acelerar corretamente e ser afetado por colisões, gravidade e outros tipos de força. Game engines proporcionam elementos da física que irão providenciar e simular isso automaticamente para o desenvolvedor. Com alguns parâmetros, objetos podem ser criados de uma forma que se comportem de uma forma realista em relação à física (UNITY 2015).

O custo de adicionar física aos objetos é alto falando em termos de processamento, então para uma aplicação é crucial saber quais objetos realmente precisam da física adicionada a eles, assim deixando o projeto funcionando muito mais veloz.

2.2.6 INTERFACE GRÁFICA

As game engine possuem, ao menos as mais básicas, formas para se criar interfaces gráficas. O desenvolvedor possuiu a liberdade de criar janelas, botões, combinar cores ou texturas para criar sua própria interface.

A interface gráfica não se resume apenas a um menu principal da aplicação, podendo ser aplicada em muitos outros pontos como caixa de diálogos, inventários para armazenar objetos e outras possibilidades.

2.2.7 SCRIPTS

Um dos pontos mais importantes das game engines são os *scripts* que podem ser anexados a objetos dentro do ambiente virtual.

Scripts podem ser usados para iniciar o usuário em um determinado ponto do mapa, adicionar movimentos de câmera, manipular a luz do ambiente, ou acionar eventos de quando o usuário se move ou aciona algo dentro do ambiente virtual.

Em alguns casos adicionar um *script* pode ser bastante simples, como ao criar um personagem e designando-o como um personagem de 3° pessoa, a game engine então terá *scripts* pré-configurados para a animação do personagem e o ambiente virtual será capaz de reagir à posição ou ao ponto de visão do personagem.

2.3 UNITY

Unity é uma game engine com um grande conjunto de ferramentas que possibilita criar jogos com uma tecnologia que executa desde gráficos até a parte de programação. Mesmo com várias ferramentas a Unity possui um ambiente intuitivo para os usuários tornando-se mais agradável a transição do foco em programação para o foco no trabalho artístico (HELGASON, 2014).

Devido às plataformas evoluírem e ficarem mais complicadas (PC, Consoles, *smartphones*) automaticamente a tecnologia necessária para fazer jogos ou outro tipo de aplicativo utilizando modelagem 3D se tornaram mais complicadas. Em alguns momentos de um projeto existe a necessidade de tipos de sombras diferentes, efeitos especiais e estilos de fumaça que precisem de um realismo maior.

A Unity é uma plataforma que tem como aspecto principal unificar processos como animação, física, renderização, inteligência artificial, som e muitos outros aspectos em um único meio. Evita-se então a necessidade anterior de para cada tipo de efeito a necessidade de um software específico (HELGASON, 2014).

Ela possui ainda suporte para várias linguagens, além de ser multi-plataforma, sendo possível se desenvolver para Android, PC, Xbox, Nintendo Wii U, PS4 e entre muitas outras (UNITY, 2015).

2.3.1 AMBIENTE DE DESENVOLVIMENTO

O ambiente de desenvolvimento da Unity é dividido em cinco abas: *scene, game, hierarchy, project e inspector.*

a) Aba *Scene*: Mostra a cena que se está trabalhando no momento. Todos os objetos são colocados e manipulados dentro desta aba.
b) Aba *Game*: Apresenta a cena diante dos olhos da câmera. É a maneira que o usuário final irá ver a cena.
c) Aba *Hierarchy*: Essa janela mostra todos os objetos contidos na cena atual. Todos os objetos necessários são listados nela sejam eles visuais ou não.
d) Aba *Project*: Tudo pode ser acessado através dessa janela desde objetos, *scripts*, texturas, modelos, áudio, vídeo, entre outras.
e) Aba *Inspector*: Contém os diferentes atributos e propriedades de um objeto que esteja selecionado. Dependendo do objeto selecionado se torna possível alterar sua posição, rotação e informações mais complexas. A Figura 2 apresenta o ambiente de desenvolvimento da ferramenta Unity.

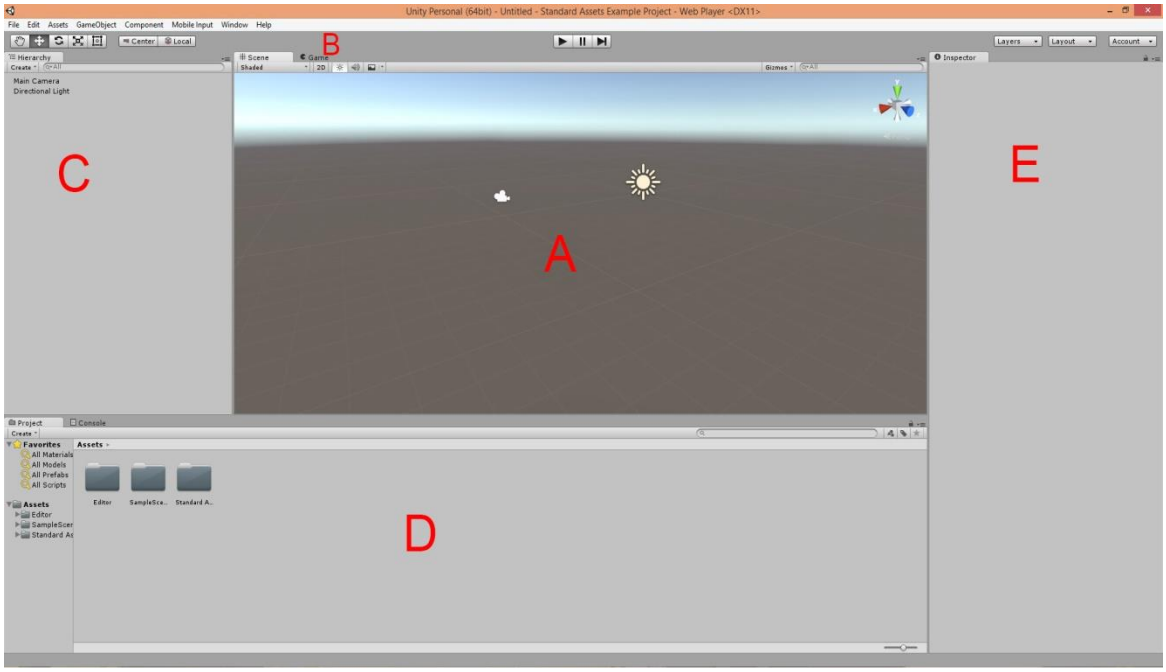

Figura 2 – Ambiente de Desenvolvimento Unity.

Fonte: Autoria Própria.

2.4 BLENDER

Blender é um software *open source* para modelagem em 3D. Suporta todos os níveis necessários para uma modelagem completa: modelagem, montagem, animação, simulação, renderização, composição e rastreamento de movimento, edição de vídeos e criação de jogos (BLENDER, 2015).

Blender apresenta multi-plataforma podendo rodar igualmente bem no Linux, Windows e Machintosh. Utiliza uma interface OpenGL que fornece uma experiência para todos os tipos de hardware e plataforma (BLENDER, 2015).

Utilizar o Blender exige uma grande dedicação do usuário, mas a plataforma recompensa esse esforço por ter infinitas possibilidades para se trabalhar em longo prazo. A Figura 3 apresenta uma escultura de um dragão modelada em Blender de nível avançado que utiliza características da ferramenta como topologia dinâmica.

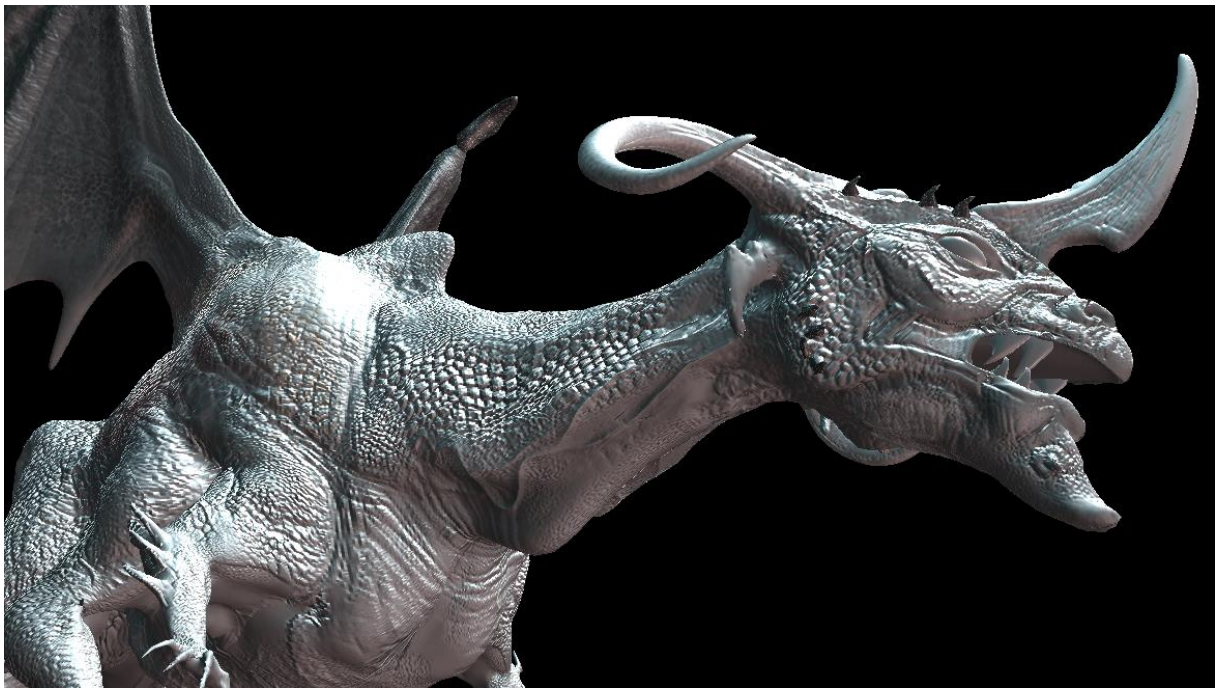

Figura 3 - Dragão modelado com Blender.

Fonte: BLENDER (2015).

Blender suporta vários tipos de render *engines*, mas três delas são a base da ferramenta: OpenGL, Blender Internal (BI) e Cycles.

OpenGL é uma engine de renderização mais técnica, no momento que se está utilizando Blender se está utilizando OpenGL pois a interface do software é gerada utilizando OpenGL. O grande benefício desse tipo de renderização é o fato de ser extremamente rápido. É possível modelar um trabalho no Blender sem atrasos devido a essa tecnologia. O lado ruim desse tipo de renderização é que para ter uma velocidade alta as modelagens são mais simplistas, com uma menor quantidade de detalhes (GUMSTER, 2011).

Blender Internal é a *engine* de renderização padrão do Blender. Desde o lançamento do Blender essa *engine* recebeu poucas mudanças, apenas pequenas adições e ajustem foram feitos. Em outras palavras a forma que ela funciona é a mesma. O motivo para se manter praticamente sem mudanças se deve pelo fato de priorizar a velocidade de renderização. Ela ignora efeitos gráficos como o reflexo de luz e utiliza atalhos da engine para simula-los de forma mais realística, mas com esse processo o resultado final da renderização é inferior caso comparado com a engine Cycles. Apesar de perder no produto final, em muitos casos ela é a ideal para se

trabalhar devido a sua velocidade e simplicidade, principalmente em ambientes de trabalho com computadores menos potente. (GUMSTER, 2011).

Cycles é focado na criação de modelagens realistas. É possível conseguir esse resultado com as *engines* anteriores, mas o custo em relação ao tempo seria muito maior (GUMSTER, 2011).

2.4.1 AMBIENTE DE DESENVOLVIMENTO NA PLATAFORMA BLENDER

O ambiente de desenvolvimento do Blender permite a visualização dos pontos mais importantes da ferramenta. Blender oferece a possibilidade para o usuário customizar da maneira que preferir este ambiente.

Blender também oferece uma grande variedade de comandos pelo teclado para facilitar no trabalho, estes, também oferecem possibilidades de customização.

Os elementos que compõem a tela são organizados em áreas, cada área possui um editor. Dentro dos editores é onde são localizados os painéis que possuem botões e controladores para se trabalhar.

Por padrão o Blender possui 5 editores, são eles: Info, 3D View, Outliner, Properties e TimeLine.

Os editores podem ser divididos em cabeçalhos ou regiões. Dentro de um editor sempre existirá ao menos uma região visível. Essa região que sempre existirá é chamado de região principal e é a parte mais importante de um editor.

Além dessa região principal existem outras disponíveis. Eles oferecem informações adicionais como as propriedades ou similares.

Cabeçalhos são pequenas partes do editor que podem estar tanto no topo ou na parte inferior. Possuem várias ferramentas comuns para serem utilizadas, assim como as regiões os cabeçalhos podem ser escondidos dentro do Blender.

A menor parte que compõem a interface são os paineis, cada um pode ser expandido para revelar diversas opções para serem trabalhadas dentro de um editor. Ao expandir um painel as informações são normalmente botões, menus, *checkboxes* e outros. A Figura 4 apresenta uma imagem do ambiente de trabalho da ferramenta Blender.

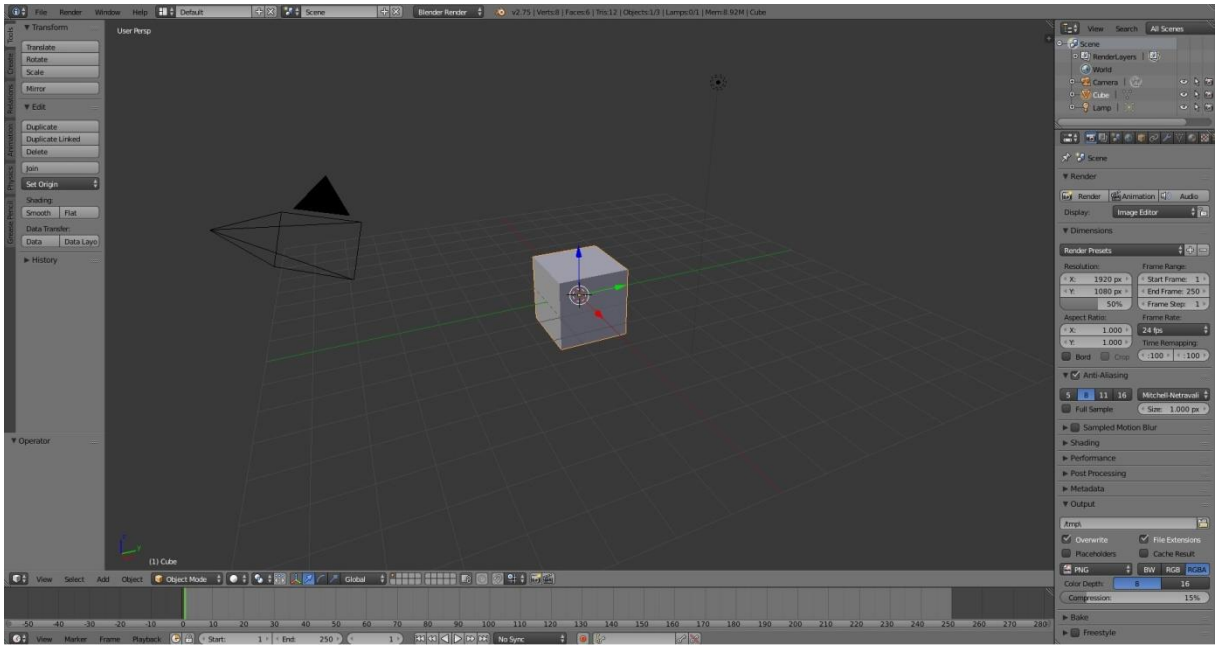

Figura 4 - Ambiente de desenvolvimento.

Fonte: Autoria própria.

2.4.2 MODELAGEM

Modelagem é um dos pontos mais importantes na criação de um ambiente. É uma arte que pode criar e replicar objetos da vida real ou dar existência a criatividade e imaginação dos seus criadores (BLENDER, 2015).

Os objetos básicos existem em várias formas e tamanhos e o Blender disponibiliza uma grande variedade para se trabalhar eficientemente com essa diversidade (BLENDER, 2015).

2.4.2.1 OBJECT MODES

Todas as scenes do projeto sempre terão no mínimo um objeto, não importa qual objeto esteja compondo o projeto todos eles podem ser movidos, rotacionados ou escalonados através *do Object Mode*. Todos os objetos podem ter alterações mais detalhadas dentro do modo de edição, *o Edit mode*.

Existem vários tipos de objetos: *Mesh*, Curva ou Câmera. Não importa o tipo de objeto, após sua criação não é mais possível alterar seu gênero (BLENDER, 2015).

2.4.2.2 MESHES

Todos os objetos compartilham algumas similaridades entre si. Embora, cada objeto possua seu tipo e suas especificações que não podem ser alteradas após sua criação (BLENDER, 2015).

Nos *meshes* mais básicos todos são construídos através de três estruturas básicas: vértices, arestas e faces.

Um vértice é basicamente um simples ponto ou posição dentro de um ambiente 3D. Esse ponto normalmente não aparece. É possível criar vértices e liga-los entre si.

Arestas ligam dois vértices. Normalmente apresentadas como linhas dentro do ambiente de desenvolvimento do Blender. São bastante importantes na modelagem de faces.

Faces são utilizadas para construir a superfície de um objeto. É a parte do objeto que o desenvolvedor estará visualizando enquanto modela (BLENDER, 2015).

2.4.2.3 CURVAS

O grande diferencial das curvas e superfícies é que estes são apresentados por funções matemáticas e não uma quantidade de pontos.

Um ponto importante das curvas é que utilizam menos dados, logo, sua produção resultando em um uso menor de memória enquanto se está trabalhando com Blender (BLENDER, 2015).

Algumas técnicas de Blender são exclusivas quando se trabalha com curvas da mesma forma que algumas técnicas utilizadas em mesh não são possíveis utilizando esse tipo de objeto (BLENDER, 2015).

2.4.2.4 SUPERFÍCIES

Superfícies são um tipo de objeto de duas dimensões. Apesar de dentro do Blender aparecer uma forma tridimensional ele não é considerado 3D pela ausência de volume. Por ser um objeto 2D ele apresenta apenas dois eixos, U e V (BLENDER, 2015).

São utilizadas principalmente para criar ondulações em terrenos, mas também podem ser aplicados em objetos de uma casa ou qualquer tipo de objeto que precise de alteração em sua superfície.

2.4.2.5 TEXTO

Um objeto que pode apresentar algum tipo de texto. Blender possuiu um sistema de fontes interno para criar os textos a serem apresentados, mas também suporta o uso de fontes externas como, por exemplo, OpenType e TrueType.

Os textos podem ser criados tanto em 2D como em 3D, e existe uma grande variedade de opções para customização das fontes (BLENDER, 2015).

Cada objeto do tipo texto pode ter no máximo 50000 caracteres, quanto mais caracteres um objeto tiver maior será o tempo para ele ser carregado (BLENDER, 2015).

2.4.2.6 OBJETOS META

São objetos que existem de forma processual dentro do Blender. A sua característica visual difere bastante dos outros tipos de objetos, lembrando algo pesado em formato esférico.

São usados principalmente na criação de efeitos como, por exemplo, utilizar vários objetos meta para criar um objeto de tipo diferente posteriormente no projeto (BLENDER, 2015).

2.4.2.7 DUPLICAÇÕES

Consiste na criação de um objeto idêntico, criando uma cópia perfeita de outro objeto. A cópia é considerada um novo objeto que recebe as informações do original. O diferencial de trabalhar com duplicações em Blender é que se pode escolher quais

tipos de informações do objeto original serão realmente transferidas para o objeto duplicado (BLENDER, 2015).

Podem ser utilizadas em estruturas onde uma parte da arquitetura é repetida inúmeras vezes, um exemplo comum seria um portão de uma casa ou instituição de ensino (BLENDER, 2015).

2.5 JAVASCRIPT

Javascript é uma linguagem de programação interpretada com a capacidade de ser orientada a objetos. Em alguns pontos da sintaxe lembra C, C++ e java, como o uso de if e while loop, essa similaridade acaba junto com a sintaxe. Em Javascript é permitido que variáveis não precisem necessariamente ter seus tipos especificados (FLANAGAN 2006).

A parte central da linguagem Javascript suporta numbers, strings e Boolean como tipos primitivos suportados (FLANAGAN 2006).

Javascript trabalhando em conjunto com características dos navegadores, é uma tecnologia para reforçar a experiência na web. Quando empregada por um computador cliente, javascript pode ajudar a tornar uma página estática simples em algo com conteúdo atraente, interativo e proporcionar uma experiência inteligente para os usuários (GOODMAN e MORRISON 2007).

Uma das teorias em codificação é que uma boa e forte escrita no código irá facilitar com que o compilador detecte uma grande quantidade de erros. Para o programador o quão mais cedo ele identificar esses erros, menor será o custo da aplicação. Javascript é uma linguagem que permite liberdade, a consequência disso é que os compiladores não estão aptos para detectar certos tipos de erros (CROCKFORD 2008).

2.6 XML

XML (*Extensible Markup Language*) é uma linguagem de marcação com o diferencial de ser possível criar as próprias *tags*. XML foi criado para suprir as limitações encontradas no HTML. Assim como o HTML, XML é baseado no SGML (*Standard Generalized Markup Language*) (TIDWELL 2002).

A essência do XML pode ser encontrada no seu próprio nome. O que define o extensível de XML é a possibilidade de definir as próprias tags, a ordem que essas tags irão ocorrer e como elas serão processadas ou apresentadas. Outra maneira de pensar sobre o fato extensível da linguagem é que ela possibilita para que todos que a utilizam extender a compreensão sobre o documento (MYER 2005).

Markup é onde o XML se destaca de outras linguagens, onde a característica mais marcante são as tags ou seus elementos. O diferencial que XML proporciona é a possibilidade do desenvolvedor criar e definir seu próprio conjunto de tags.

É importante lembrar que XML não é simplesmente uma linguagem, ela é uma metalinguagem. Metalinguagem possibilitada a criação ou definição de outras linguagens (MYER 2005).

3 ESTADO DA ARTE

De acordo com MELLO *et al*. (2013) a computação gráfica trabalha de forma que um universo tridimensional é criado e através de cálculos da CPU (unidade central de processamento) e GPU (unidade de processamento gráfico). O resultado desse processamento é a imagem convertida na tela do monitor. No caso de diferentes mídias como filmes, maquetes ou jogos não se trata apenas de uma imagem, e sim várias imagens sendo exibidas sequencialmente para causar a impressão de movimento.

Ao se trabalhar com Blender dificilmente haverá apenas uma técnica a ser utilizada na criação de um objeto. Para a criação do Bloco L1, MELLO *et al*. (2013), inicialmente adquiriram uma planta baixa para o resultado final da modelagem se aproximar ao máximo do modelo real. Essa planta dentro do Blender é conhecida como *Blueprints* e através dela é possível acelerar o processo de modelagem. A Figura 5 apresenta o modelo do Bloco L1.

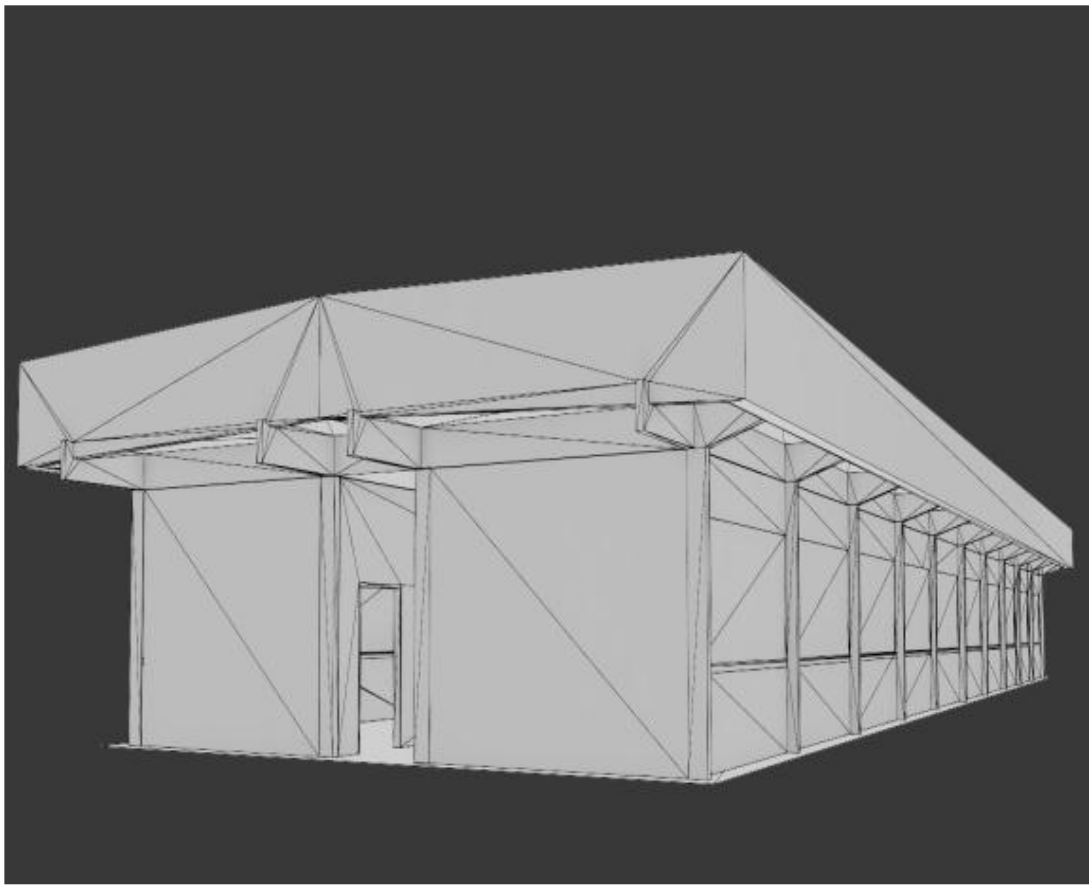

Figura 5 - Modelo do Bloco L1.
Fonte: MELLO *et al.* (2013)

De acordo com MELLO *et al*. (2013) após a modelagem estar concluída inicia-se o processo de texturização, que se trata de colorir o bloco utilizando imagens reais para chegar a um resultado mais próximo ao original. A Figura 6 apresenta o Bloco L1 finalizado com as devidas texturas aplicadas.

Figura 6 - Bloco L1 finalizado.

Fonte: Autoria Própria.

Por fim com o modelo finalizado no Blender o mesmo é transferido para a ferramenta Unity, recebendo as funções dentro do ambiente 3D proposto. Na Unity é realizada a disposição dos objetos em cena. Com o término da disposição dos objetos inicia-se então a adição de funções do ambiente, como abrir portas andar ou pular. A Unity oferece muitas dessas funções previamente configuradas facilitando o processo.

4 MATERIAIS E MÉTODOS

O ambiente virtual que foi desenvolvido nesse trabalho é baseado em um ambiente real localizado na UTFPR, o Bloco L1. Permitindo simular uma visita sem a necessidade de uma pessoa fazê-lo pessoalmente.

Utilizando Blender para modelagem e Unity para dar funções específicas dentro do projeto, foi desenvolvido um ambiente virtual onde é possível visitar o Bloco L1 assim como receber informações dos professores referentes ao curso e disciplinas de Ciências da Computação.

Para o desenvolvimento deste ambiente uma série de trabalhos foram desenvolvidas e são descritas a seguir.

4.1 MODELAGEM

Diferente do Bloco L1, modelado por Mello *et al*.(2013), onde existiu a necessidade de modelar o interior do objeto (o próprio bloco), os objetos decorativos que constituem o bloco (computadores, mesas, ...), apenas necessitam apresentar seu exterior para fazerem sua função. Devido a essa particularidade a modelagem dos objetos iniciou-se a partir de um cubo, e não de uma planta baixa como base.

A modelagem das mesas foi realizada utilizando o cubo como referência. Através do cubo busca-se alterar suas proporções formas e tamanho, a Figura 7 demonstra o estágio inicial da modelagem com o cubo alterado para representar a parte superior da mesa.

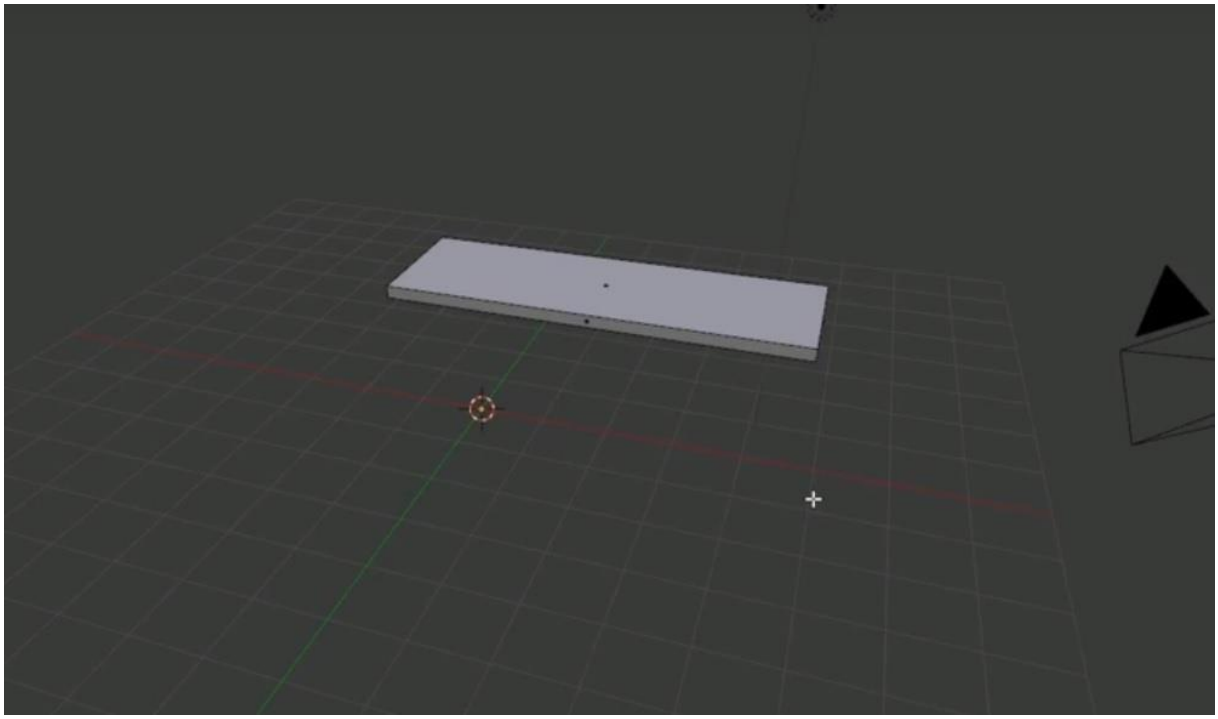

Figura 7 - Topo da mesa.

Fonte: Autoria Própria

Para servir de guia a mesa é dividida em várias partes utilizando a opção *Loop Cut and Slid*. *Loop Cut and Slide*, que cria um corte no local em que o objeto foi selecionado, gerando assim uma ou mais novas divisões. Essas divisões servirão para localizar e separar mais adequadamente a mesa e com isso tornar mais fácil a construção dos pés da mesa. Eles são construídos após selecionar uma área que foi dividida e então apenas esticar para cima ou para baixo. O formato dos pés da mesa inicialmente vem do tamanho das divisões realizadas com o *Loop Cut and Slide*. A Figura 8 demonstra a mesa devidamente dividida e já possuindo a estrutura inferior que foi construída a partir da expansão para baixo do topo da mesa.

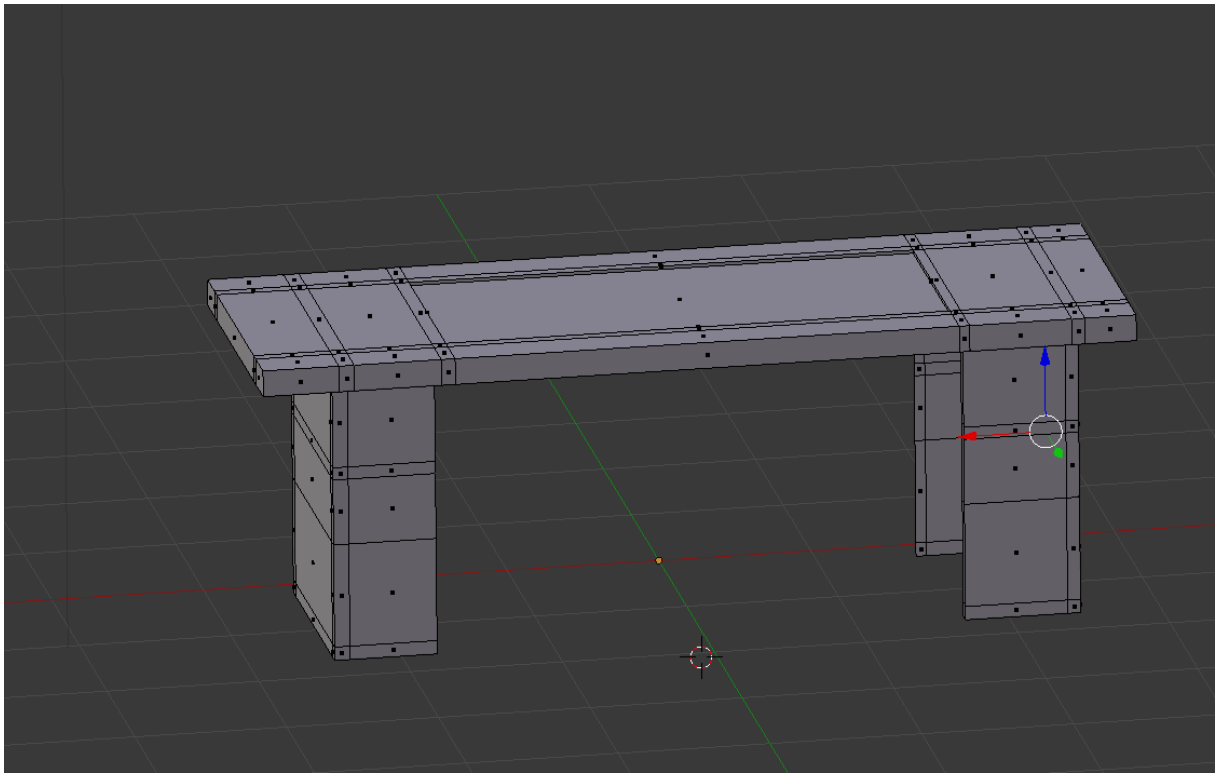

Figura 8 - Modelo 3D da mesa.

Fonte: Autoria Própria

Com todo o modelo pronto inicia-se a aplicação da textura. Isso é realizado selecionando o objeto e indo nas propriedades de textura, então se aplica a textura escolhida. O Blender proporciona várias texturas na própria ferramenta, mas também é possível adquirir outras texturas baixando-as da internet. Após a textura aplicada é possível editar o quão intenso ela será aplicada diante do objeto, essa alteração é realizada na propriedade material. A Figura 9 demonstra as propriedades de textura e material respectivamente.

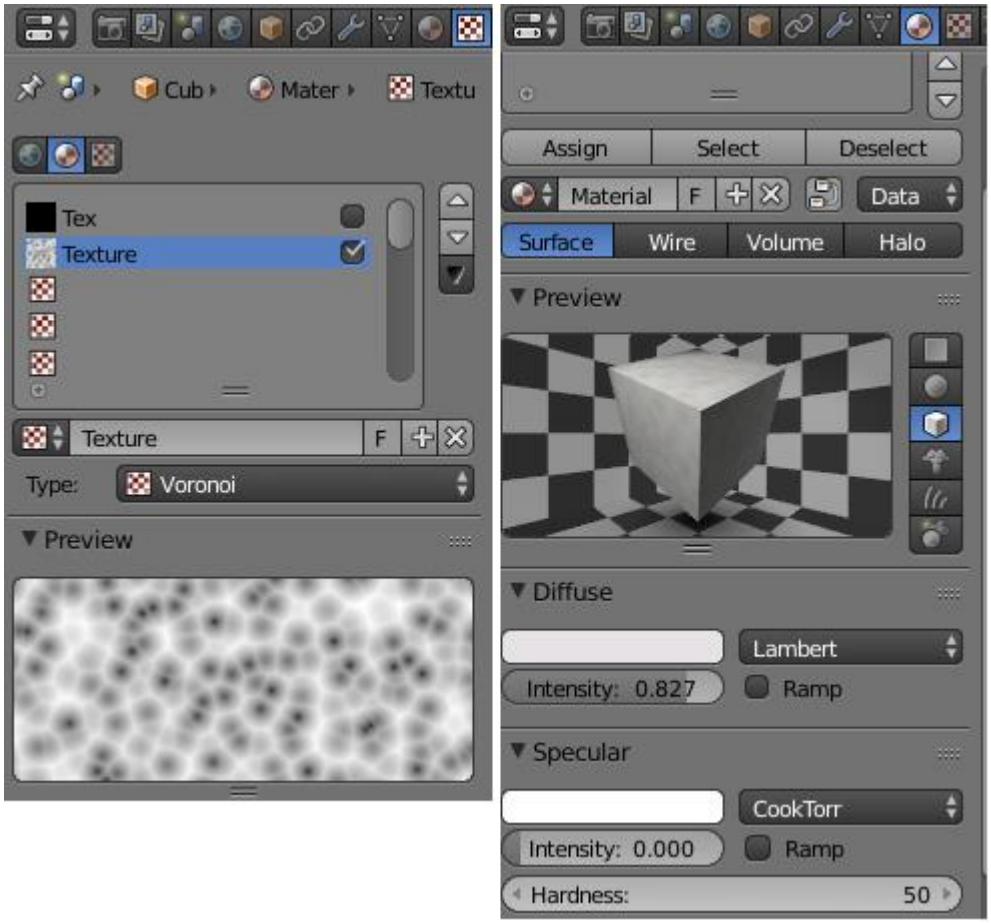

Figura 9 - Opções de textura e material.

Fonte: Autoria Própria

A modelagem das cortinas seguiu todas as técnicas descritas para as mesas, mas possuiu alguns pontos que diferem. Um desses pontos foi que as cortinas usaram um plano como referência, diferente das mesas que utilizaram um cubo.

A Figura 10 apresenta os estágios inicias de modelagem das cortinas.

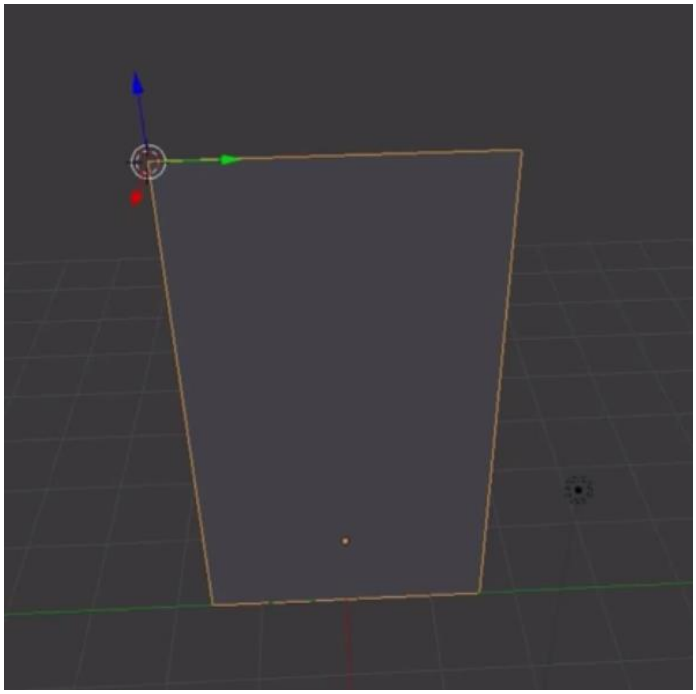

Figura 10 - Modelagem inicial das cortinas.

Fonte: Autoria Própria

Após o processo inicial de modelagem notou-se a necessidade de tornar a cortina mais realista, para isso algumas alterações foram feitas nas opções de tecido do objeto.

Utilizou-se os campos de colisão do tecido para criar ondulações características de cortinas, para isso foi adicionado a opção para o objeto colidir com si mesmo, gerando então ondulações no tecido. Para finalizar a modelagem da cortina a seda foi selecionada como tipo de tecido. A Figura 11 apresenta as opções referentes ao tecido da cortina:

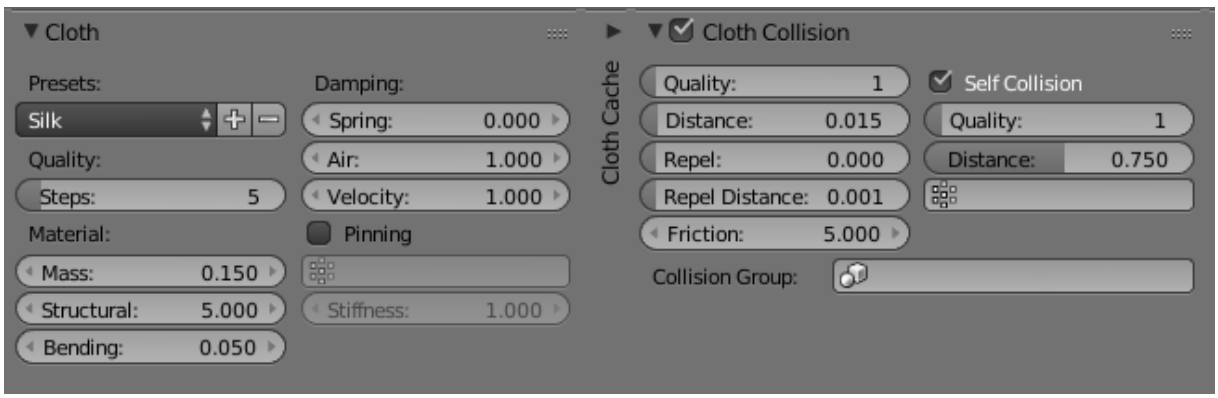

Figura 11 - Opções de tecido.

Fonte: Autoria Própria.

4.2 FUNÇÕES

Todo o ambiente virtual necessita receber algum tipo de informação para aplicar sua real função. Desde as mais básicas como, por exemplo, indicar que as paredes são sólidas e não podem ser atravessadas livremente como a interação com objetos ou pessoas dentro do ambiente virtual. Cada função adicionada difere em sua dificuldade e modo de ser aplicada.

4.3 BANCO/XML

O projeto do Bloco L1 e interação com professores na Unity foi programado utilizando a linguagem C#. Para realizar essa interação foi necessário importar os dados de vários documentos XML (*parsing*), para serem utilizados na interface do usuário, isso possibilita o usuário a receber informações dos professores virtuais localizados em suas respectivas salas dentro do Bloco L1.

Os professores irão compartilhar suas informações pessoas no ambiente virtual:

a) Disciplinas: Listadas quais disciplinas que o professor está trabalhando atualmente assim como uma sinopse do que cada disciplina trabalha.
b) Área de Pesquisa: Apresentar os campos ou temas de pesquisa em que um professor está atuando. Seja esse um projeto próprio ou em conjunto com alunos do Câmpus da UTFPR.
c) Formação: Informações referentes à formação acadêmica dos professores.

Iniciou-se o trabalho de desenvolvimento criando uma estrutura XML que possibilita para os administradores do sistema facilmente adicionar e editar qualquer tipo de informação referente aos professores. Com um sistema robusto de *data schema*, qualquer usuário do tipo administrador que tiver um mínimo entendimento da linguagem XML e os respectivos privilégios para tal tem a possibilidade de editar as informações que são passadas para os professores. Assim o administrador tem um melhor controle para manter os usuários atualizados diante das informações.

A estrutura do XML foi criada baseando-se nas informações que os professores terão previamente descritas, a Figura 12 demonstra a estrutura dos arquivos XML dos professores.

```
<?xml version="1.0" encoding="utf-8"?>

<professor>
  <nome>Mario</nome>
  <imagem>
    mario
  </imagem>
  <posição>
    <x>281.58</x>
    <y>0</y>
    <z>212.547</z>
  </posição>
  <rotação>
    <y>98</y>
  </rotação>
  <resumo>
    At vero eos et accusamus et iusto odio dignissimos du
  </resumo>
  <descrição>
    Et harum quidem rerum facilis est et expedita distinc
  </descrição>
  <disciplinas>
    Disciplina 1
    Disciplina 2
  </disciplinas>
  <pesquisa>
    Pesquisa 1
    Pesquisa 2
    Pesquisa 3
  </pesquisa>
</professor>
```

Figura 12 - Estrutura XML com informações dos professores.

Fonte: Autoria Própria

Cada elemento do arquivo XML tem seu objetivo específico, a seguir os elementos são descritos individualmente.

a) Nome: Elemento onde é inserida a informação referente ao nome do professor.
b) Imagem: Imagem que representa o professor, a Unity aceita todos os tipos de imagens (JPG, PNG, etc.).
c) Posição: Posição do professor dentro do ambiente da Unity com as coordenadas X, Y e Z.

d) Rotação: Informação que apresenta para qual lado o professor estará olhando, ou seja, sua rotação na coordenada em Y é alterada para girar o modelo 3D do professor.
e) Resumo: Um resumo sobre o professor.
f) Descrição: Neste elemento são apresentadas as informações referentes a graduação de um professor.
g) Disciplinas: Disciplinas que o professor está responsável em lecionar.
h) Pesquisa: Área de pesquisa que o professor participa.

Todas as informações podem ser editadas diretamente nos arquivos XML sem a necessidade de abrir a Unity para que o projeto se atualize. A exceção está nas informações referentes a posição dos professores, existe a possibilidade del editar mas não se tem como ter certeza exatamente onde o professor irá aparecer apenas editando os números referente a sua posição. Para essa informação da posição é aconselhável para uma maior precisão utilizar a Unity, mas para todas as outras informações o administrador do sistema pode atualizar as informações sem a necessidade de abrir a Unity.

Os documentos XML utilizam o *parsing* por um *script*. O único objetivo desse *script* é de carregar os documentos XML e dividi-los para as string de arrays que são facilmente acessíveis pela Unity. Esses *scripts* basicamente estão pegando a informação XML e transformando ela em string para que a Unity possa ler e usufruir dessa informação. A Figura 13 mostra uma parte do código do XMLParser.js que faz a leitura do XML e transforma para string para que seja possível popular as janelas de diálogo.

```
if(!collectNodeName && nodeName.length>0){
    if(nodeName[0]==SLASH){
        // close tag
        if(textValue.length>0){
            currentNode["_text"]+=textValue;
        }

        textValue="";
        nodeName="";
        currentNode=parents.Pop();
    }else{

        if(textValue.length>0){
            currentNode["_text"]+=textValue;
        }
        textValue="";
        var newNode:XMLNode=new XMLNode();
        newNode["_text"]="";
        newNode["_name"]=nodeName;

        if(!currentNode[nodeName]){
            currentNode[nodeName]=new XMLNodeList();
        }
        var a:Array=currentNode[nodeName];
        a.Push(newNode);
        parents.Push(currentNode);
        currentNode=newNode;
        nodeName="";
    }
```

Figura 13 - Trecho de código do XMLParser.js

Fonte: Autoria Própria

Em conjunto com o XMLParser.js estão os scripts XMLNode.js e XMLNodeList.js, estes três scripts são responsáveis por criar o array de professores e manterem ele de forma dinâmica para a Unity. Dessa forma é possível adicionar um número infinito de professores desde que se respeite a nomenclatura de professorN.xml, sendo N o número do professor. Todo esse sistema de *parsing* é feito no fundo enquanto o programa é carregado, e isso ocorre de uma forma que o usuário final não perceba o que está acontecendo.

Para eles, o programa está simplesmente rodando e pronto para eles andarem e explorarem as classes. Para o programa em si, uma quantidade está sendo analisada, linha por linha, e estruturada em um modo que será apresentada no ambiente virtual como um simples texto de um elemento da interface. Essa informação esta pronta para ser apresentada na tela com o requisito do usuário, como por exemplo, um clique do mouse.

4.4 MOVIMENTAÇÃO

Como os usuários podem controlar o ambiente virtual de uma visão de primeira pessoa, eles então podem caminhar pelo Bloco L1 utilizando as setas do teclado e o

mouse para se direcionar como olhar para cima, baixo, esquerda ou direita. Os controles são designados para serem fáceis de adaptar para até mesmo pessoas que não tem familiaridade com um computador, e isso prove uma ótima forma para o usuário ter controle da sua movimentação dentro do ambiente virtual. A Figura 14 apresenta a visão do usuário em primeira pessoa assim como o cursor que o guia dentro do ambiente.

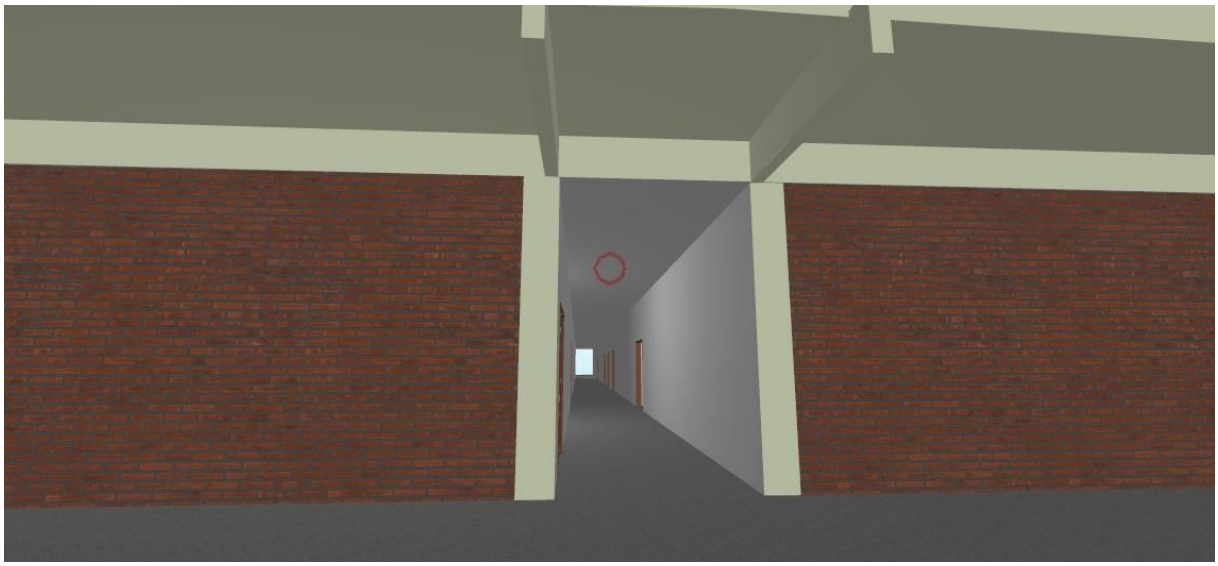

Figura 14 - Visão em primeira pessoa do usuário com o cursor.

Fonte: Autoria Própria

Quando um usuário entra em uma sala de aula, ele vê várias mobílias que foram pré-modeladas no Blender e organizadas no ambiente, assim como os professores para interagir. Essa interação é feita de um modo que até mesmo usuários com um computador antiquado consigam iniciar conversações e adquirir informações dos professores.

4.5 INTEGRAÇÃO

Para mobiliar os laboratórios com os móveis de cada sala utilizaram-se o processo de duplicação de cada objeto. Como em cada laboratório existem várias mesas e computadores, ao invés de adicionar um por um, basta adicionar o objeto e duplica-lo utilizando o comando CTRL+D. Esse tipo de característica que a Unity

proporciona facilita muito mobiliar ambientes com grande repetição de objetos sem gastar muito tempo fazendo-o. A Figura 15 apresenta um laboratório de aula com seus respectivos objetos.

Figura 15 - Laboratório L16 do Bloco L1.

Fonte: Autoria Própria

A mesma estratégia foi utilizada para mobiliar as salas dos professores. Um objeto de cada vez foi criado e então duplicado com o comando CTRL+D para serem alocados em seus devidos pontos dentro do ambiente. O diferencial nessas salas está nos professores, ao invés de simplesmente serem criados como objetos eles estão sendo localizados no ambiente diretamente pelos seus respectivos arquivos XML. A Figura 16 apresenta uma das salas dos professores com sua estrutura.

Figura 16 - Sala dos professores.

Fonte: Autoria Própria

Para fazer uma interação, um simples clique do mouse irá iniciar uma cadeia de reações dos *scripts*, e usando matrizes de dados que foram criadas pelo *parsing* do XML, uma interface será mostrada para o usuário final com as informações desejadas do professor. A seguir serão descritos os passos para a interação ser possível dentro da Unity.

As bordas que indicam que é possível interagir com um professor são geradas através de um *script* usando o cursor como referência, o *script* referente a isso é o PlayerInput.js. No momento que o usuário direciona o mouse para o professor o *script* irá verificar se o objeto é do tipo professor, caso positivo o *script* irá atualizar as informações adicionando uma borda em volta do professor para o mesmo brilhar. A Figura 17 apresenta a função do *script* que gera esse efeito dentro da Unity enquanto a Figura 18 apresenta um comparativo com o professor sem borda e com borda.

```
function Update () {

    // vai verificar se o usuário está "mirando" de fato no professor
    var _hit : RaycastHit;

    var _ray : Ray = Camera.main.ViewportPointToRay( new Vector2(0.5, 0.5) );

    // E se tiver um objeto/professor, então:
    if(Physics.Raycast(_ray, _hit, 3, interactionMask)) {

        // Se o objeto não for do tipo professor, vai ignorar
        if(!_hit.collider.CompareTag("Professor"))
            return;

        //se for professor irá pegar ele para usar com a interface
        currentProfessor = _hit.collider.GetComponent(Professor);

        if(currentProfessor == null)
            return;

        // aqui ele atualiza pro brilho (contorno)
        currentProfessor.Highlight(true);
    } else if(currentProfessor != null) {
        // e aqui ele retorna ao estado padrão
        currentProfessor.Highlight(false);
        currentProfessor = null;
    }

    // Irá verificar se o player clicou
    if(Input.GetMouseButtonDown(0) && Screen.lockCursor)
        CheckClick();
}
```

Figura 17 - Função que cria borda nos professores.

Fonte: Autoria Própria

Figura 18 - Comparativo professor sem borda e com borda.

Fonte: Autoria Própria

Dentro do mesmo *script* é onde se verifica se o usuário clicou no professor ou não. Caso o usuário clique no professor a Unity irá então alterar o cursor do mouse e gerar a caixa de diálogo para a interação com o professor desejado. Após o usuário verificar as informações desejadas e fechar a caixa de diálogo o *script* irá retornar o cursor ao seu estado padrão para exploração. A Figura 19 demonstra em código como a Unity alterna entre mostrando ou escondendo o cursor de exploração.

```
// Essa função fica alternando os controles do player ou mostrando/escondendo o cursor
// Ela é usado quando o usuário começa uma interação e quando ele fecha a caixa de diálogo
function EnableControls( _set : boolean ) {

    GetComponent(MouseLook).enabled = _set;
    transform.parent.GetComponent(MouseLook).enabled = _set;
    cursor.enabled = _set;
    Screen.lockCursor = _set;
}

// se clicou no professor, irá desabilitar os controles e abrir a caixa de diálogo
function CheckClick() {
    if(currentProfessor == null)
        return;

    // Controles sendo desativados~
    EnableControls(false);

    // E então irá mostrar a caixa de diálogo com as informações do professor clicado
    PanelsManager.inst.infoPanel.SetActive(true);
    InfoPanel.inst.UpdatePanel();
}~
```

Figura 19 - Função que alterna os modos do cursor.

Fonte: Autoria Própria

O *script* responsável por criar a caixa de diálogo e alimentar o mesmo com as informações dos professores é o InfoPanel.js. O *script* controla o tamanho da janela a ser apresentada, ao iniciar a interação a caixa de diálogo é apresentada em um tamanho padrão, mas o usuário tem a opção de diminuir essa janela com um clique do mouse. A Figura 20 apresenta um trecho do código onde o controle do tamanho da janela está sendo feito.

```
// Essa é a função que faz o controle de aumentar/diminuir a tela de diálogo
function TogglePanel() {


    fullscreen = !fullscreen;

    if(fullscreen) {
        // Se estiver fullscreen, update
        rect.anchoredPosition = Vector2.zero;
        rect.sizeDelta = Vector2.zero;
        rect.pivot = new Vector2(1, 0.5);
        rect.anchorMin = Vector2.zero;
        rect.anchorMax = new Vector2(1, 1);
        rect.offsetMin = Vector2.zero;
        rect.offsetMax = Vector2.zero;
    } else {
        // se não estiver fullscreen, então ele irá atuailizar para se ajustar
        rect.anchoredPosition = cachedRectSettings[0];
        rect.sizeDelta = cachedRectSettings[1];
        rect.pivot = cachedRectSettings[2];
        rect.anchorMin = cachedRectSettings[3];
        rect.anchorMax = cachedRectSettings[4];
        rect.offsetMin = cachedRectSettings[5];
        rect.offsetMax = cachedRectSettings[6];
    }
}
```

Figura 20 - Faz o controle do tamanho da caixa de diálogo.

Fonte: Autoria Própria

Este mesmo *script* tem a função de atualizar a caixa de diálogo com as informações do respectivo professor. Ele utiliza as informações que estão sendo fornecidas pelo *script* Professor.js e apresenta dentro da caixa de diálogo, a Figura 21 apresenta o código que faz essa atualização para as informações aparecerem.

```
/*
Função que vai alimentar as janelas dos professores~ No caso do professor atual
*/
function UpdatePanel () {

    // O professor
    var _prof : Professor = PlayerInput.inst.currentProfessor;

    // se o professor por algum motivo nulo, ignora o resto
    if(_prof == null)
        return;

    // o painel default então ficará on, e os restantes off
    descriptionLabel.transform.parent.gameObject.SetActive(true);
    disciplinesLabel.transform.parent.gameObject.SetActive(false);
    researchAreasLabel.transform.parent.gameObject.SetActive(false);

    // e aqui finalmente é onde o diálogo será atualizado com as informações do professor (imagens, formação, disciplina, etc)
    nameLabel.text = "Conversando com "+_prof.professorName;
    summaryLabel.text = _prof.summary;
    descriptionLabel.text = _prof.description;
    disciplinesLabel.text = _prof.disciplines;
    researchAreasLabel.text = _prof.researchAreas;
    image.texture = _prof.image;

}
```

Figura 21 - Função que atualiza os dados dos professores.

Fonte: Autoria Própria

Para ser possível o InfoPanel.js apresentar as informações do professor é necessário buscar as mesmas nos arquivos XML, dois *scripts* fazem essa função: Datasets.js e Professor.js.

O Datasets.js é o *script* principal que irá ler os arquivos XML que estão localizados na pasta Resources/XML dentro do projeto. Tudo que ele faz é ler todos os arquivos XML que estão dentro dessa pasta e deixar pronto para serem atualizados e utilizados por outros scripts.

Importante descrever que é aqui que é instanciado um objeto professor do tipo prefab. O prefab do professor nada mais é que um *template* para todos os professores se guiarem. É através dele que é possível pegar o modelo 3D e replicar para os diversos professores do ambiente junto com suas funções. Um prefab existe para facilitar a edição de todos os professores do ambiente caso necessário, se uma alteração de algum elemento for editada no prefab essa alteração automaticamente será replicada para todos os professores do Bloco L1.

A Figura 22 apresenta o trecho de código que faz a leitura dos arquivos XML.

```
function Start () {

    var _dirInfo = new DirectoryInfo(Application.dataPath+"/Resources/XML");
    var _files = _dirInfo.GetFiles("*.xml");

    for(var f : System.IO.FileInfo in _files) {

        var _xmlFile = Resources.Load("XML/"+f.Name.Replace(".xml", ""), typeof(TextAsset));
        var _xml = _xmlFile.text;

        var parser = new XMLParser();

        var node = parser.Parse(_xml);

        var _obj : GameObject = Instantiate(Resources.Load("Prefabs/Professor_PF"));
        var _professor = _obj.GetComponent(Professor);

        _professor.UpdateData(node);
    }
}
```

Figura 22 - Trecho de código responsável pela leitura dos arquivos XML.

Fonte: Autoria Própria

O Professor.js tem como função pegar as informações que foram tratadas pelo XMLNode.js e XMLParser.js e deixar elas disponíveis para outros *scripts*. É nesse ponto onde se atualizam os dados dos professores de acordo com o XML e os replica para todos os professores. O Professor.js está apenas atualizando os dados do

professor de acordo com os XML. A Figura 23 apresenta o trecho de código onde o *script* está recebendo os arquivos XML e transformando-os para string para serem trabalhados.

```
function UpdateData ( _xml : XMLNode ) {

    // usa o XML Node e retorna info em string
    professorName = ((((_xml["professor"] as Array)[0] as Hashtable)["nome"] as Array)[0] as Hashtable)["_text"].ToString().Trim();
    summary = ((((_xml["professor"] as Array)[0] as Hashtable)["resumo"] as Array)[0] as Hashtable)["_text"].ToString().Trim();
    description = ((((_xml["professor"] as Array)[0] as Hashtable)["descrição"] as Array)[0] as Hashtable)["_text"].ToString().Trim();
    disciplines = ((((_xml["professor"] as Array)[0] as Hashtable)["disciplinas"] as Array)[0] as Hashtable)["_text"].ToString().Trim();
    researchAreas = ((((_xml["professor"] as Array)[0] as Hashtable)["pesquisa"] as Array)[0] as Hashtable)["_text"].ToString().Trim();

    //mesma coisa do anterior mas irá pegar o nome da imagem aqui
    var _img = ((((_xml["professor"] as Array)[0] as Hashtable)["imagem"] as Array)[0] as Hashtable)["_text"].ToString().Trim();
    var _imgString : String = _img.ToString();

    // irá pegar a textura correta das imagens
    image = Resources.Load("Images/"+_imgString.Trim()) as Texture2D;

    //aqui ele está pegando a posição dos professores através do XML NODE
    var _x = ((((((_xml["professor"] as Array)[0] as Hashtable)["posição"] as Array)[0] as Hashtable)["x"] as Array)[0] as Hashtable)["_text"];
    var _y = ((((((_xml["professor"] as Array)[0] as Hashtable)["posição"] as Array)[0] as Hashtable)["y"] as Array)[0] as Hashtable)["_text"];
    var _z = ((((((_xml["professor"] as Array)[0] as Hashtable)["posição"] as Array)[0] as Hashtable)["z"] as Array)[0] as Hashtable)["_text"];
    var _rot = ((((((_xml["professor"] as Array)[0] as Hashtable)["rotação"] as Array)[0] as Hashtable)["y"] as Array)[0] as Hashtable)["_text"];

    // vai atualizar então a posição de acordo com o XML node
    transform.position = new Vector3( float.Parse(_x), float.Parse(_y), float.Parse(_z) );
    transform.localEulerAngles = new Vector3(0, float.Parse(_rot), 0);
}
```

Figura 23 - Função que retorna informações XML em string.

Fonte: Autoria Própria

5 RESULTADOS E DISCUSSÃO

Este capítulo apresentará os resultados obtidos durante o desenvolvimento do ambiente virtual que representa o Bloco L1 da UTFPR.

5.1 OBJETOS MODELADOS

Todos os objetos foram modelados utilizando a ferramenta Blender. A mesa e a cortina foram modeladas de acordo com a metodologia apresentadas neste trabalho. A Figura 24 e 25 apresentam os objetos finalizados e prontos para serem transferidos para a Unity e serem trabalhados e alocados de acordo com o ambiente virtual.

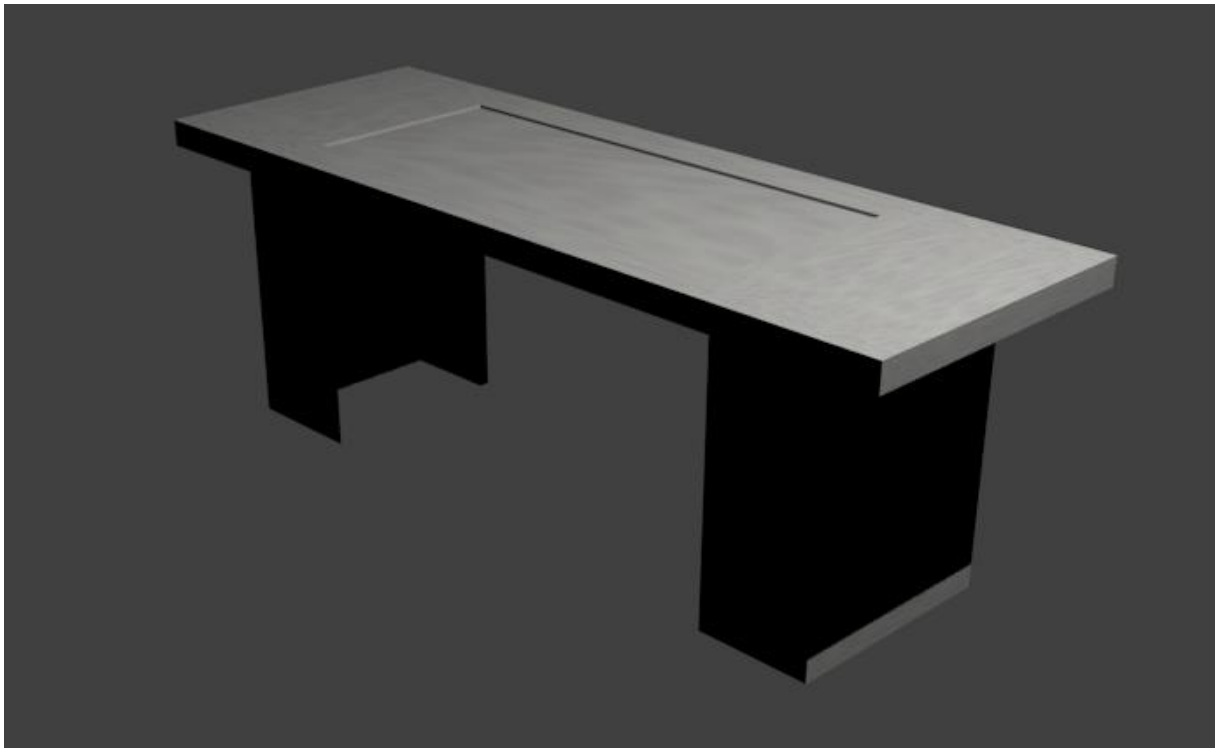

Figura 24 - Modelo da Mesa.

Fonte: Autoria Própria

Figura 25 - Modelo da Cortina.

Fonte: Autoria Própria

Outros objetos que compõem o ambiente do Bloco L1 foram obtidos através da *Unity Asset Store*[2]. Como o próprio nome diz é um local para armazenar todos os tipos de *assets* desenvolvidos pelos usuários da Unity como: scripts, interfaces, inteligência artificial, modelos 3D ou 2D, áudio, entre outros. Os desenvolvedores tem a opção de publicar seus *assets* de forma gratuita ou paga para o público.

Com o intuito de diminuir o tempo de desenvolvimento alguns objetos foram adquiridos através da Asset Store, são eles: computadores, monitores, mobília de escritório e as pessoas que representam os professores.

5.2 INTERAÇÃO COM PROFESSORES

Com a leitura do XML e os *scripts* direcionando essas informações corretamente por fim a caixa de diálogo estará preenchida e pronta para ser apresentada. A Figura 26 apresenta uma caixa de diálogo com informações de um professor fictício.

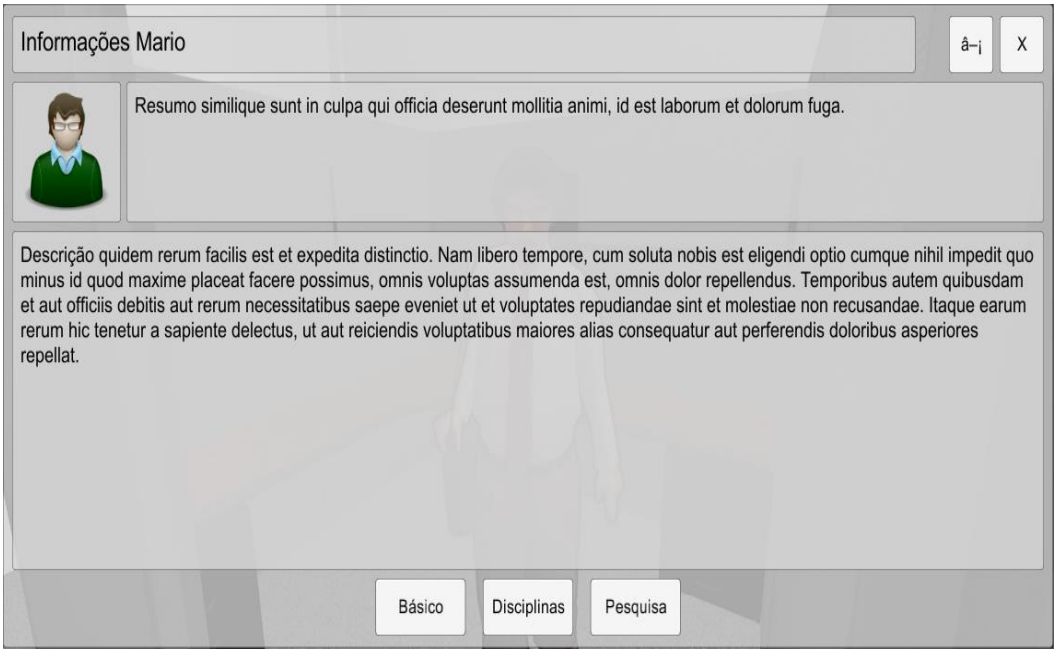

Figura 26 - Caixa de diálogo com as informações do professor.

[2] https://www.assetstore.unity3d.com/en/

Fonte: Autoria Própria

O usuário pode encerrar essa interação a qualquer momento, e estará livre para continuar explorando o Bloco L1 e interagindo com outros elementos como, por exemplo, o abrir e fechar portas do Bloco L1. Enquanto o usuário faz isso, em plano de fundo, a estrutura de *scripts* estará mantendo as informações de cada professor para o professor correto de fato, e quando essa interação é requisitada, o array apropriado de informações é obtido e usado como elementos textuais que são apresentados em uma tela de conversação.

Com o projeto finalizado foi então gerado o seu executável para a plataforma Windows na arquitetura x86_64. O executável é o projeto completo pronto para ser utilizado sem necessitar da Unity. Diferente de um projeto comum onde após pedir para gerar a versão final tudo estará pronto para executar, neste caso utilizando XML existe a necessidade de transferir os arquivos XML e imagens do projeto que são localizados na pasta Resources para a pasta de mesmo nome que fica localizada dentro da versão final do executável. Essa ação não irá interferir no executável que já foi gerado, eles são passados apenas para servirem como guia para encontrarem o caminho e lerem os arquivos dentro da pasta Resources.

O sistema desenvolvido além de possibilitar um ambiente de fácil uso, interfaces para adquirir informações dos professores e um ambiente do Bloco L1 buscando retratar o real, é também criado para deixar a manutenção e atualização das informações mais dinâmica. Esses dois diferentes sistemas estão reunidos graças a plataforma Unity.

6 CONSIDERAÇÕES FINAIS

6.1 CONCLUSÃO

Para a realização deste trabalho integrando Unity com Blender conclui-se que o desenvolvimento exige demanda de tempo e dedicação.

A modelagem de um ambiente virtual baseado no real é um processo que exige atenção aos detalhes, por mais simples que seja o objeto sempre existirá técnicas a serem estudas para a construção dos mesmos.

Com base no projeto desenvolvido nota-se que as ferramentas possuem muitas opções para facilitar o trabalho do desenvolvedor, ou seja, o desenvolvedor tem maior liberdade para focar no processo criativo da aplicação ao invés da codificação.

A integração utilizando XML permitiu um sistema dinâmico que possibilita inserção, alteração e remoção com facilidade para futuros administradores do projeto.

Os tours virtuais conseguem agregar valor para as instituições de uma maneira diferenciada chamando assim a atenção de possíveis clientes em casos de corporações ou estudantes em caso de instituições de ensino.

Os ambientes virtuais também podem ser utilizados para manter a história de uma instituição, criando-se assim museus virtuais em ambiente 3D para visitação de futuras gerações.

6.2 TRABALHOS FUTUROS/CONTINUAÇÃO DO TRABALHO

Como forma de continuidade ao trabalho propõe-se a modelagem de um ambiente 3D em maior escala. Este possuindo maior quantidade de objetos. A possibilidade de modelar os rostos dos professores para uma representação mais próxima do real.

Quanto ao funcionamento, na interação com os professores adicionar vídeo e áudio referente a cada professor para uma busca de informações mais dinâmica.

REFERÊNCIAS BIBLIOGRÁFICAS

BLENDER. **Introdução ao Blender**, 2015. Disponível em: <https://www.blender.org>. Acesso em 26 de maio de 2015.

CAPPEX. Disponível em: <https://www.cappex.com/media/digitalMobile2012.pdf> . Acesso em 18 de Agosto de 2015.

CROCKFORD Douglas. **JavaScript: The Good Part** - O'Reilly Media / Yahoo Press, 2008

DALGARNO, Barney, e John Hedberg. **3D Learning Environments in Tertiary Education**. Disponível em: <http://www.ascilite.org/conferences/melbourne01/pdf/papers/dalgarnob.pdf> Acesso em 15 de agosto de 2015.

ENGER, Michael. **Game Engines: How do they work?** Disponível em: <http://www.giantbomb.com/profile/michaelenger/blog/game-engines-how-do-they-work/101529/> Acesso em 05 de agosto de 2015.

FLANAGAN, David. **JavaScript The Definitive Guide** – 6. Ed. O'Reilly Media, 2011.

GOODMAN Danny e Michael MORRISON. **JavaScript Bible** – 7. Ed. Wiley, 2010.

GUMSTER, Van Jason. ***Blender for Dummies*** - For Dummies, 2011.

HELGASON, David. **How Unity3D Become a Game Development Beast.** Disponível em: <http://insights.dice.com/2013/06/03/how-unity3d-become-a-game-development-beast>. Acesso em 15 de maio de 2015.

ICEF. **The Power of Virtual Tours in Student Recruitment.** Disponível em: <http://monitor.icef.com/2012/09/the-power-of-virtual-tours-in-student-recruitment/>. Acesso em 20 de Agosto de 2015.

JANKOWSKI, Jacek, e Martin Hachet. **A survey of Interaction Techniques for Interactive 3D Environments**. Disponível em <https://hal.inria.fr/hal-00789413/document> Acesso em 08 de agosto de 2015.

MELLO, Vinicius B.; MORAES, Roger R.; METZ, Jean; PAULA FILHO, Pedro L. **Virtualização de Ambientes e Museu Virtual Interativo**. 4o. Meditec, 2013.

MYER, Tom. **A Really, Really, Really Good Introduction to XML**. Disponível em: <http://www.sitepoint.com/really-good-introduction-xml/> Acesso em 18 de Setembro de 2015.

TIDWELL, Doug. **Introduction to XML.** Disponível em <http://www.ibm.com/developerworks/xml/tutorials/xmlintro/xmlintro.html> Acesso em 18 de Setembro de 2015.

UNITY. Disponível em: <https://unity3d.com>. Acesso em 20 de maio de 2015.