UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

FELIPE LOPES ZEM
KLAUS METTEGANG DIENER

PROJETO E DESENVOLVIMENTO DE UM MÓDULO DE SOFTWARE PARA VIRTUALIZAÇÃO DE TRANSPORTE COLETIVO PARA A PLATAFORMA CURITIBA-VIEWPORT

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA

2017

FELIPE LOPES ZEM
KLAUS METTEGANG DIENER

PROJETO E DESENVOLVIMENTO DE UM MÓDULO DE SOFTWARE PARA VIRTUALIZAÇÃO DE TRANSPORTE COLETIVO PARA A PLATAFORMA CURITIBA-VIEWPORT

Trabalho de Conclusão de Curso apresentada ao curso de Bacharelado em Sistemas de Informação da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de "Bacharel em Sistemas de Informação".

Orientador: Paulo Cézar Stadzisz

CURITIBA

2017

Ministério da Educação
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
Câmpus Curitiba
Diretoria de Graduação e Educação Profissional
Departamento Acadêmico de Informática
Coordenação do Curso de Bacharelado em Sistemas de Informação

TERMO DE APROVAÇÃO

"PROJETO E DESENVOLVIMENTO DE UM MÓDULO DE SOFTWARE PARA VIRTUALIZAÇÃO DE TRANSPORTE COLETIVO PARA A PLATAFORMA CURITIBA-VIEWPORT"

por

"Felipe Lopes Zem e Klaus Mettegang Diener"

Este Trabalho de Conclusão de Curso foi apresentado como requisito parcial à obtenção do grau de Bacharel em Sistemas de Informação na Universidade Tecnológica Federal do Paraná - UTFPR - Câmpus Curitiba. O(a)(s) aluno(a)(s) foi(ram) arguido(a)(s) pelos membros da Banca de Avaliação abaixo assinados. Após deliberação a Banca de Avaliação considerou o trabalho _____.

_____ **Paulo Cezar Stadzisz** (Presidente - UTFPR/Curitiba)	_____ **Robson Ribeiro Linhares** (Avaliador 1 - UTFPR/Curitiba)
_____ **Alexandre Reis Graeml** (Avaliador 2 - I UTFPR/Curitiba)	_____ **Leyza Baldo Dorini** (Professor Responsável pelo TCC – UTFPR/Curitiba)
_____ **Leonelo Dell Anhol Almeida** (Coordenador(a) do curso de Bacharelado em Sistemas de Informação – UTFPR/Curitiba)	

"A Folha de Aprovação assinada encontra-se na Coordenação do Curso."

RESUMO

Zem, Felipe Lopes; Diener, Klaus Mettegang. PROJETO E DESENVOLVIMENTO DE UM MÓDULO DE SOFTWARE PARA VIRTUALIZAÇÃO DE TRANSPORTE COLETIVO PARA A PLATAFORMA CURITIBA-VIEWPORT. 50 f. Trabalho de Conclusão de Curso – Bacharelado em Sistemas de Informação, Universidade Tecnológica Federal do Paraná. Curitiba, 2017.

Gameficação pode ser utilizada como ferramenta de aprendizagem em ambientes virtualizados, ferramenta de visualização e gestão de dados e até mesmo como possibilitador de interações entre usuários e sistemas públicos. A UTFPR está desenvolvendo o primeiro modelo 3D da cidade de Curitiba voltado a utilidade pública – o Curitiva-Viewport. Este documento descreve o desenvolvimento de um módulo para a inclusão de veículos de transporte público da cidade, com especial ênfase nos ônibus biarticulados no Curitiba-Viewport.

Palavras-chave: Gameficação; Ambientes Virtualizados; Curitiba-ViewPort.

ABSTRACT

Zem, Felipe Lopes; Diener, Klaus Mettegang. . 50 f. Trabalho de Conclusão de Curso – Bacharelado em Sistemas de Informação, Universidade Tecnológica Federal do Paraná. Curitiba, 2017.

Gamefication can be used as a learning tool in virtualized environments, visualization and management tool of data or even as an enabler of interactions between users and public systems. UTFPR is developing the first 3D model of the city of Curitiba, which is aimed for public utility – the Curitiba-Viewport. This document describes the development of a module for the inclusion of public transportation vehicles in the virtual city, with special emphasis on bi-articulated buses in Curitiba-Viewport.

Keywords: Gamefication; Virtualized Environments; Curitiba-ViewPort.

LISTA DE FIGURAS

LISTA DE SIGLAS

BSI	Bacharelado em Sistemas de Informação
C#	C Sharp (Linguagem de Programação)
CPGEI	Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial
EUA	Estados Unidos da América
IBM	International Business Machines
GPS	Global Positioning System
UTFPR	Universidade Tecnológica Federal do Paraná

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTO DO TRABALHO

Com o crescimento das cidades, cresce também a necessidade de infraestrutura, o que, por consequência, gera a necessidade de um planejamento efetivo por parte dos gestores públicos. Planejar a distribuição de energia, água, esgoto, ruas e estradas, iluminação pública e transporte público não é uma tarefa simples. A criação do planejamento efetivo pode ser facilitada com a utilização de ferramentas computacionais de simulação, aliadas a sistemas que promovam a integração de informações da própria cidade, como no caso da integração das informações coletadas pelas forças policiais a partir de câmeras de segurança e controle de tráfego espalhadas por uma cidade, desde que exista uma alimentação dos dados coletados em tempo real.

O conceito de Cidade Inteligente (Smart City) não é algo simples e não existe um consenso mundial sobre ele (NEIROTTI et al., 2014), pois a realidade de cada cidade é única. Modelar e planejar Cidades Inteligentes é um grande desafio computacional pelo grande número de requisitos e dados que devem ser analisados de forma a conseguir uma boa aproximação da realidade. As cidades inteligentes visam facilitar o planejamento estratégico por parte dos governantes, favorecer e agilizar a comunicação da população com os principais serviços de emergência e fornecer dados precisos sobre condições do tempo, localização de ônibus e estabelecimentos próximos, entre outros.

Alguns países já possuem projetos de Smart Cities, citando como exemplo a cidade de Nova York nos Estados Unidos da América. O projeto da cidade inteligente de Nova York foi publicado em setembro de 2015 e, segundo o *site* oficial do governo (CITY, 2015), tem como objetivo a distribuição equitativa de recursos. Para isso, eles definiram alguns benefícios que o projeto da Smart City poderia trazer, como, por exemplo: a coleta de informações em tempo real aumenta a responsividade de governo, pois, com dados constantemente sendo coletados e atualizados, é possível planejar para diminuir custos e distribuir de maneira apropriada os principais serviços públicos (CITY, 2015). A utilização de ferramentas digitais e sensores

permite um uso eficiente da infraestrutura e dos recursos da cidade. Segundo o projeto que foi publicado, existe a pretensão de modificar vários aspectos da cidade, de maneira a torna-los inteligentes, como por exemplo o transporte público, que otimizaria a mobilidade urbana, diminuindo congestionamentos e a emissão de gás carbônico, por meio do gerenciamento remoto do tráfego e de semáforos.

Ambientes virtuais são ambientes reais que foram modelados computacionalmente, com o propósito de facilitar a interatividade do usuário com situações e objetos. Um exemplo clássico de ambiente virtual são os simuladores de pilotagem de aviões que, segundo a Agência Nacional de Aviação Civil, são uma ferramenta fundamental para o treino de pilotos, pois, o ambiente virtual permite a simulação de qualquer situação de emergência que pode ocorrer durante um voo, sem colocar vidas em risco. Pode-se também citar o exemplo de ambientes 3D para o ensino de física e outros tópicos de engenharia (TIBOLA et al., 2014), pois em um ambiente virtual no qual se tem controle das condições de um determinado experimento, também é possível visualizar e verificar os fenômenos sendo estudados, facilitando o aprendizado.

Com o advento de um Ambiente Virtual que simula as várias características de uma cidade, essa análise pode se tornar mais simples, pois se torna possível, simular o impacto das mudanças aplicáveis à sua infraestrutura. Para que a simulação seja eficiente é necessário que a cidade virtualizada contenha o máximo de informações possíveis sobre a cidade real, com altíssima precisão e disponibilidade.

O projeto Curitiba ViewPort, da Universidade Tecnológica Federal do Paraná, tem como objetivo principal a criação de uma plataforma digital para a emulação da cidade de Curitiba, representando desde o dia-a-dia da população até o andamento em tempo real dos serviços públicos, como transportes coletivos, operações de manutenção e contenção de catástrofes, entre outros, tudo isso através de virtualização tridimensional. Possuir um meio computacional que permita rapidamente visualizar o cotidiano de uma cidade de maneira remota, e utilizar este meio como ferramenta para tomada de decisões, é algo que potencialmente trará benefícios no trânsito e em outros setores de serviços públicos. Além disso, tanto a população quanto turistas poderiam fazer uso desta mesma ferramenta, para finalidades variadas.

Uma interação com outros usuários por meio da plataforma virtual traria, ainda, vários benefícios ao cidadão. Os usuários seriam representados por avatares controlados remotamente ou representados de acordo com o GPS contido nos aparelhos celulares e demais dispositivos portáteis. A participação remota em eventos, reais ou virtuais, seria viabilizada. A comunicação

com autoridades públicas seria agilizada e informações sobre o trânsito, horários de ônibus e suas rotas, estariam facilmente disponibilizadas e seriam melhor interpretadas com auxílio da representação digital em três dimensões. Muitos são os ganhos esperados decorrentes da utilização de tal software cujas pesquisas estão em andamento na UTFPR.

1.2 OBJETIVOS DO TRABALHO

O objetivo desta monografia de conclusão de curso é realizar o levantamento de requisitos e implementar um módulo de software para a plataforma Curitiba-Viewport, voltado à virtualização do transporte coletivo da cidade. Além da modelagem e animação, pretende-se interligar este módulo de software aos servidores de informação da URBS (Urbanização de Curitiba) para coletar informações em tempo real das posições dos ônibus, com a intenção de sincronizar os veículos reais com os virtuais.

Os objetivos específicos decorrentes do objetivo principal são:

- Especificar os requisitos funcionais para a virtualização do transporte coletivo em uma cidade virtual;
- Estudar e desenvolver técnicas de simulação;
- Especificar e construir modelos de dados para representação e armazenamento das informações relativas à virtualização do transporte coletivo;
- Desenvolver a interface de comunicação com os servidores da URBS através de *Web Services*;
- Realizar a verificação do módulo desenvolvido integrado à plataforma Curitiba-Viewport.

1.3 JUSTIFICATIVAS E MOTIVAÇÃO

A escolha do tema deste trabalho de conclusão de curso veio da oportunidade de ingresso em um projeto voltado à virtualização da cidade de Curitiba desenvolvido na UTFPR. Do produto de software gerado poderão ser derivados diversos usos benéficos tanto para as entidades governamentais quanto para os cidadãos. Alguns dos usos pretendidos desta ferramenta dizem respeito ao planejamento urbano, ao aprendizado e ao lazer. Este trabalho também é justificado pelo fato de não ser conhecida, até o momento em que a proposta deste trabalho foi apresentada, outra iniciativa similar no Brasil ou no mundo. O mais próximo que

se encontra são softwares puramente recreacionais, limitados à visualização tridimensional de cidades reais e fictícias, não incluindo alguns dos aspectos visados pelo Curitiba-ViewPort que são, por exemplo, a fiel representação da realidade dos serviços públicos e a rápida disseminação de informações de eventos. Além disso, para a elaboração do software proposto neste documento é necessário uma integração de diversas áreas de conhecimento ministradas no curso de Sistemas de Informação, ofertado pela UTFPR, efetivamente colocando em prática grande parte do conteúdo absorvido ao longo da formação. Algumas destas áreas são: programação, engenharia de software, banco de dados, redes de computadores e sistemas distribuídos e design de interação. Como motivação adicional, existe aquisição de novos conhecimentos relacionados à utilização de ferramentas de desenvolvimento aplicadas a plataformas gráficas, que normalmente são empregadas para jogos digitais. Ou seja, apesar deste projeto ser de âmbito acadêmico e voltado à obtenção de benefícios para a cidade, existe também um aspecto lúdico que envolve tanto desenvolvedores quanto possíveis usuários.

1.4 ESTRUTURA DO DOCUMENTO

O capítulo 2 descreve o referencial teórico, como artigos, dissertações e informações de *sites* oficiais, nos quais o desenvolvimento das atividades está baseado. O capítulo 3 descreve a metodologia utilizada para o desenvolvimento deste trabalho. O capítulo 4 discute os recursos de hardware, software bem como a viabilidade do projeto. O capítulo 5 apresenta como foram desenvolvidas todas as partes necessárias para o bom funcionamento do projeto. O Capítulo 6 explicita as dificuldades encontradas bem como as soluções utilizadas para resolver problemas decorrentes do desenvolvimento. O Capítulo 7 apresenta os detalhes da rota e do cenário construídos virtualmente para testes e demonstrações do módulo desenvolvido e, por fim, o capítulo 8 apresenta a conclusão do trabalho bem como possíveis maneiras de se utilizar os conhecimentos construídos para trabalhos futuros.

1.5 CONTRIBUIÇÕES DO TRABALHO

A plataforma Curitiba-Viewport é inovadora, complexa e com muitas funcionalidades. Esse documento descreve a metodologia utilizada para o desenvolvimento de um módulo independente mas passível de ser adicionado à plataforma Curitiba-Viewport. Esse módulo contém o gerenciamento e a animação dos veículos que representam parte do transporte público de Curitiba, utilizando uma conexão via *Web Services* com o servidor de dados da URBS com objetivo de coletar informações sobre o posicionamento destes veículos analisados em tempo

real. Quando integrado ao projeto, o módulo deve permitir que os personagens controlados pelos usuários “embarquem” e “desembarquem” dos veículos e façam virtualmente os trajetos de um ponto até outro, efetivamente simulando o funcionamento da frota de transporte coletivo em operação.

2 REFERENCIAL TEÓRICO

2.1 CIDADE INTELIGENTE

Uma cidade precisa ser atrativa tanto para turistas quanto para os residentes, pois uma cidade que possui uma população satisfeita tende a prosperar. O crescimento da população traz benefícios para os setores comercial e industrial, o que por sua vez aumenta a quantidade de impostos e recursos que o governo coleta. Esses impostos, ao serem utilizados para a manutenção e para o planejamento do futuro da cidade, são aplicados de modo que as necessidades da população sejam atendidas.

Em uma cidade de médio porte convivem centenas de milhares de pessoas que possuem necessidades comuns, como moradia, transporte, segurança, lazer, trabalho e estudo. É dever das autoridades públicas planejar a infraestrutura da cidade para que essas pessoas possam ter suas necessidades atendidas. Essa tarefa exige que, constantemente, sejam coletados e processados os dados relevantes de cada um destes tópicos. Segundo informações do *site* oficial da IBM uma cidade inteligente possui projetos e soluções que permitem gerar e coletar uma grande quantidade de dados, que podem ser utilizados para tornar o planejamento público mais coerente com a realidade.

Para gerenciar a enorme quantidade de dados que são gerados diariamente se torna necessária a utilização de softwares. Esses softwares atuam sobre estes dados e provêm informações relevantes sobre o funcionamento de determinada área da cidade, ou desta como um todo. Um exemplo é o que ocorre em diversas cidades dos Estados Unidos da América, onde existe um sistema que coleta dados sobre o funcionamento dos trens nas ferrovias, permitindo uma análise preditiva da posição individual de cada trem, melhorando o controle sobre toda a malha ferroviária, visando evitar acidentes e otimizar o tempo de viagem (MIRANDA, 2012).

Para a realização deste trabalho o conceito adotado para Cidade Inteligente foi o seguinte: uma cidade que utiliza meios computacionais para a realização de pelo menos um dos objetivos a seguir:

- Auxiliar o desenvolvimento da cidade como um todo;
- Melhorar a qualidade dos serviços de utilidade pública;
- Distribuir os recursos financeiros de forma estratégica;
- Planejar mudanças na infraestrutura.

2.2 CIDADE VIRTUAL

Uma cidade virtual nada mais é do que um ambiente virtual que simula o ambiente de uma cidade em seus vários aspectos como comércios, transporte público, pontos turísticos, entre outros. Existem vários exemplos no mercado de simuladores de cidades virtuais criados com o propósito de entreter os seus usuários, sendo o um dos mais famosos o *Second Life*. Porém, é totalmente possível de se aplicar o mesmo conceito para diversas outras finalidades, o que já é feito hoje em dia.

Um exemplo desta prática para fins não-recreacionais é a autointitulada Cidade Virtual desenvolvida por Maria João Pinto Luís Miranda, que possui como objetivo ajudar na reabilitação neurocognitiva de pessoas com disfunção executiva e outras funções cognitivas subsidiárias, como a atenção e a memória. Segundo a autora "O principal objetivo do jogo é proporcionar aos pacientes diversos desafios que se assemelhem a tarefas reais do dia a dia. Nesse sentido, pretende-se diariamente educar o paciente a primeiro planear as suas tarefas e posteriormente executá-las com sucesso pela ordem inicialmente agendada. A primeira fase é composta por um conjunto de tarefas que devem ser corretamente ordenadas, de acordo com a regra estipulada para o nível. Seguidamente, o jogador deve percorrer a cidade, usando os transportes que dispõe, no sentido de alcançar com sucesso todas as tarefas dadas". Os médicos e psicólogos podem utilizar esse jogo sério para avaliar a situação de um paciente e determinar quais os próximos passos de seu tratamento, pois os vários níveis de dificuldade incremental, permitem análise do real progresso que os pacientes estão tendo e quais pontos ainda precisam ser trabalhados com o paciente(MIRANDA, 2012).

2.3 AMBIENTE VIRTUAL CURITIBA-VIEWPORT

A plataforma Curitiba-ViewPort é um projeto desenvolvido na UTFPR, coordenado pelo professor Paulo Cézar Stadzisz, envolvendo estudantes de graduação dos cursos de Bacharelado em Sistemas de Informação, Engenharia de Computação e estudantes de pós-graduação do CPGEI/UTFPR.

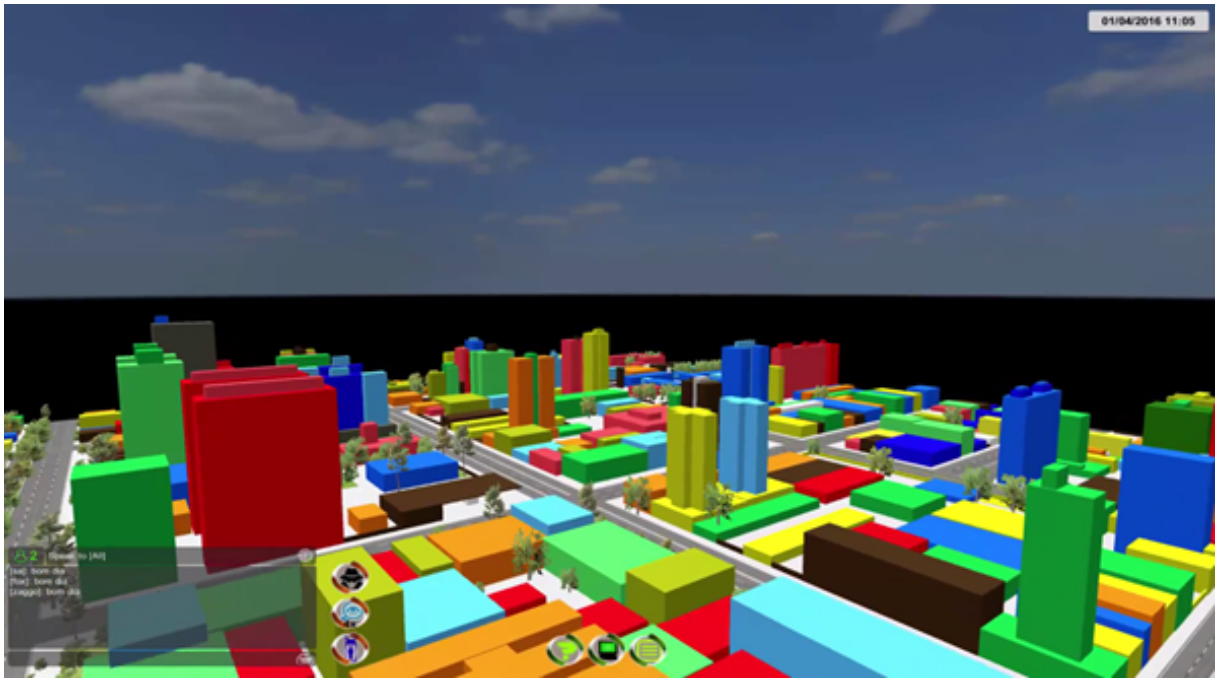

Figura 1: Visão aérea do Curitiba-ViewPort.

O objetivo do projeto é desenvolver uma plataforma que ofereça as funcionalidades básicas de uma cidade virtual em analogia com a cidade de Curitiba. Sobre esta plataforma, poderão ser agregadas camadas com funcionalidades que expandem as aplicações da cidade virtual. A criação desta plataforma envolve um projeto arquitetural e lógico intenso de forma a prever os recursos para dar dinâmica e garantir a infraestrutura semelhante a de uma cidade real. Isto envolve a modelagem de prédios, ruas, vegetação, além dos elementos móveis como pessoas e veículos e condições ambientais como as variações climáticas, condições de iluminação, entre outros. (LAZARINE GABRIEL JOSé; NASCIMENTO, 2015)

Atualmente a plataforma já possui recursos que permitem aos jogadores criarem seus personagens e explorarem aproximadamente vinte quadras em torno da UTFPR, incluindo prédios, vegetação e ruas. As coordenadas geográficas da plataforma correspondem as coordenadas reais com um grande grau de precisão, o que permite uma sincronização de locais, pessoas e veículos.

2.4 MOTOR DE JOGO

Softwares, denominados Motores de Jogo (em inglês, Game Engines), fornecem um conjunto de ferramentas diversas com o intuito de facilitar o trabalho necessário dos desenvolvedores de jogos durante o processo de criação, incorporando os elementos básicos para o seu desenvolvimento: motor de renderização gráfica, motor de física, interface gráfica

do usuário (GUI), inteligência artificial, som, comunicação em rede, entre outros (FRITSCH; KADA, 2004). Desta forma, é possível afirmar que os motores de jogo são o coração dos jogos (FRITSCH; KADA, 2004), um conjunto de módulos que permite que os programadores se foquem mais no conteúdo do produto do que no conteúdo técnico (FRITSCH; KADA, 2004). As comunidades de desenvolvedores que fazem uso destes motores de jogo normalmente os consideram como frameworks ou plataformas, por estas providenciarem uma camada de abstração entre o conteúdo do jogo e o ambiente de hardware e software em que é executado, pois como é impossível determinar os diferentes hardwares que serão utilizados pelos usuários, se torna necessária a abstração deles.

Os mais populares são: Unreal Engine, GameMaker: Studio, RPG Maker, CryEngine, Source Engine e Unity3D.

2.4.1 UNITY3D

O Unity3D é um Motor de Jogo proprietário com grande foco em portabilidade entre plataformas. Desde o seu lançamento em 2005, cinco versões foram distribuídas ao público e esta segue em desenvolvimento pela empresa Unity Technologies. Este software oferece suporte gráfico utilizando bibliotecas DirectX para os sistemas operacionais Windows, e bibliotecas OpenGL para Linux, Mac, e demais plataformas. As linguagens de programação que são suportadas (TECHNOLOGIES, 2017).

Optou-se pela utilização do *Unity3D* para o desenvolvimento do projeto Curitiba-ViewPort, por esta possuir uma opção de uso pessoal gratuito para fins não-comerciais ou cuja acumulação de lucros das vendas dos jogos criados não ultrapasse US$100.000. Além disso, por ser uma ferramenta que está há vários anos no mercado, cujos desenvolvedores patrocinam e disseminam o seu uso em eventos, é encontrada uma grande quantidade de documentos e videoaulas a respeito da utilização e das capacidades desta Game Engine. (TECHNOLOGIES, 2017)

2.5 MODELAGEM GEOMÉTRICA

Modelos computacionais tridimensionais são utilizados para fins de testes e verificações de situações diversas, tendo utilidade em vários campos do conhecimento. A indústria aeronáutica testa a aerodinâmica dos aviões que são fabricados por meio de modelos computacionais. A robótica utiliza de modelos para o teste de implementação de movimentações de um robô. Pode-se citar também os modelos computacionais utilizados para a

construção civil, onde se pode planejar cuidadosamente detalhes tanto internos quanto externos de obras utilizando-se destes softwares.

Ferramentas de modelagem geométrica existem em abundância no mercado, e como exemplos pode-se citar o AutoCad, o Google Sketchup e o Blender.

O AutoCad é uma ferramenta profissional de engenharia para diversas finalidades do ramo, como projetos elétricos, projetos de cabeamento estruturado, plantas e projeções 3D de construções, etc. A grande precisão e a enorme quantidade de funcionalidades que o AutoCad possui o torna um software de modelagem computacional muito utilizado por engenheiros, arquitetos e projetistas.

Já o Google Sketchup, por outro lado, é algo mais lúdico, voltado a modelagens que não necessariamente adotem normas de segurança e de desenho técnico, sendo muito utilizado para modelagens de casas, prédios, móveis, entre outros.

Por fim, o Blender também é uma ferramenta de modelagem 3D, que permite de maneira mais intuitiva a manipulação de texturas, animação e renderização dos objetos, e que é normalmente utilizada em conjunto com outras ferramentas para a criação de jogos digitais. Os modelos criados no Blender também não possuem a precisão necessária para serem considerados em projetos de engenharia.

2.5.1 BLENDER

O *Blender* é um software gratuito de código aberto desenvolvido pela *Blender Foundation*, cujos escritórios estão localizados em Amsterdã, na Holanda. A organização se mantém através de doações e vendas de diversos cursos e tutoriais para a utilização do *Blender*, além das centenas de contribuintes que estão envolvidos em algum grau com o desenvolvimento e melhoria do software (FOUNDATION,)

O software possui as principais ferramentas básicas para o desenvolvimento e a criação de modelos 3D e também permite utilizar outros softwares externos em conjunto, além de permitir o desenvolvimento de scripts em Python para criar ferramentas customizadas e add-ons e ter suporte a *Open Shading Language* (OSL) que pode ser utilizado para desenvolver shaders personalizados. O *Blender* possui diferentes ferramentas desenvolvidas especialmente para criar modelos mais complexos, como superfícies orgânicas e relevos irregulares, além de deixar a possibilidade de desenvolver um ambiente inteiro de uma só vez utilizando o full compositor. O programa conta ainda com uma vasta biblioteca de extensões adicionais criadas pela comunidade e um ambiente interno para o desenvolvimento de jogos, simulação,

renderização, composição, rastreamento de movimento assim como criação de animações (FOUNDATION,). Tem-se o *Blender* como a principal ferramenta de modelagem gráfica para os envolvidos com a plataforma Curitiba-ViewPort, conforme o comparativo feito por por Gabriel José Lazarine e Henrique Luiz Granato de Quadros do Nascimento em sua dissertação "Proposição, Especificação e Desenvolvimento de um Software Experimental de Cidade Virtual"(LAZARINE GABRIEL JOSé; NASCIMENTO, 2015).

2.6 REALIDADE AUMENTADA

A interação do mundo real com ferramentas computacionais está cada vez mais comum. Com apenas alguns toques em uma tela de celular é possível saber a sua exata localização em uma cidade e quais bares, restaurantes, pontos de ônibus, escolas, lojas e demais marcos da cidade estão próximos de sua posição. Os usuários de aplicativos de Realidade Aumentada conseguem projetar imagens e modelos tridimensionais em espaços vazios ou até mesmo sobre estruturas já existentes, fazendo uso de seus smartphones ou de dispositivos de imersão em realidade virtual,*Oculus Rift* ou o *Google Glass*.

Essa interação do mundo real com dispositivos computacionais pode ser chamada de Realidade Aumentada. A Realidade Aumentada é um campo de estudo ainda recente e que possui como característica a combinação de objetos reais e virtuais em um ambiente real (AZUMA *et al.*, 2001).Vários aplicativos estão disponíveis que fazem uso deste conceito. Um exemplo desta interação é o aplicativo *Coral Visualizer*, disponibilizado pela empresa Tintas Suvinil, que permite ao usuário manipular as cores de paredes das quais ele tirou fotos, possibilitando uma visualização de como certas cores iriam alterar o ambiente em questão antes mesmo do usuário comprar a cor de tinta que ele esteja testando desta maneira.

Um projeto da empresa *Augment* (AUGMENT,) permite ao usuário de um smartphone coletar informações históricas relevantes ao focalizar a câmera de seu celular em determinados objetos históricos, como por exemplo a Estátua da Liberdade que fica em Nova York, nos EUA.

A interatividade e a manipulação de ambientes reais por meio de dispositivos computacionais pode trazer diversos benefícios para uma Cidade Inteligente. Ao utilizar um aplicativo específico um cidadão qualquer poderia saber em tempo real várias informações importantes como as condições do trânsito em certas ruas que ele deseja transitar, quais pontos de ônibus se encontram próximos localização e qual o itinerário realizado por esses ônibus. Softwares poderiam ser integrados em viaturas policiais e em ambulâncias visando ao envio de informações para o controle de sinaleiros da cidade, que por sua vez poderiam tomar medidas

automatizadas para que o tempo de viagem desses veículos seja reduzido.

2.7 AMBIENTES VIRTUALIZADOS / SIMULADOS

Um ambiente simulado, é aquele em que o usuário consegue interagir com um ambiente virtualizado, e esse ambiente fornece ao usuário uma resposta que se aproxime da realidade. Um bom exemplo disso são os simuladores de direção de automóveis que são utilizados por algumas autoescolas. Esses simuladores oferecem ao usuário uma experiência que se aproxima muito da real experiência de dirigir de um carro e permite aos instrutores simular diversas situações diárias do trânsito, com o objetivo de ensinar ao aluno como reagir a determinadas situações, como neblina, chuva forte, desvios e acidentes na pista. O maior benefício desse ambiente de aprendizagem virtual, é a facilidade de representar situações e de interagir com essas situações sem a necessidade da preocupação com danos à vida humana e a outros veículos. Outro tipo de simulador muito utilizado, são os simuladores de física, onde alunos podem realizar diversos experimentos, observar e aprender sobre fenômenos físicos que muitas vezes seriam difíceis de se representar em laboratórios, por causa do grande custo que os equipamentos de laboratório possuem.

Outro tipo de ambiente virtualizado é o *Google Earth* que possibilita a uma pessoa visualizar imagens reais de qualquer continente, país, cidade e, até mesmo, das ruas próximas a sua casa. O modelo digital e 3D gerado pelo Google Earth permite que o usuário apenas visualize lugares sem que haja uma interação do usuário com o mundo digitalizado, isto é, o usuário apenas observa os modelos tridimensionais dos prédios, museus e monumentos. (GOOGLE,)

2.8 WEB SERVICES

Para realização da integração das informações de diferentes sistemas utilizados em diferentes ámbitos é necessária a sincronização de dados entre esses sistemas. Porém, na maioria dos casos, os sistemas foram desenvolvidos de maneiras diferentes e até em linguagens de programação diferentes. Para resolver o problema da diferença entre arquiteturas e linguagens de programação, é utilizada uma solução chamada Web Service. Um Web Service utiliza da linguagem de marcação XML para permitir as aplicações enviarem e receberem dados entre si. (W3SCHOOLS, 2017)

```
<?xml version="1.0" encoding="UTF-8"?>
<posicao_onibus>
<capao_raso>
  <longitude>49° 16' 23" W</longitude>
  <latitude>25° 25' 40" S</latitude>
  <numero_serie>111222333</numero_serie>
</capao_raso>
<capao_raso>
  <longitude>49° 16' 23" W</longitude>
  <latitude>25° 25' 40" S</latitude>
  <numero_serie>111222334</numero_serie>
</capao_raso>
</posicao_onibus>
```

Figura 2: Exemplo da organização de um arquivo XML.

2.8.1 XML

A linguagem XML é uma recomendação da W3C que foi criada com o objetivo de ser legível tanto para computadores quanto para humanos, transportar dados via web e facilitar a integração de bancos de dados diferentes, pois o XML se concentra na estrutura da informação e não em sua aparência. O XML não possui um padrão definido para a criação de tags, pois o usuário pode criar quantas tags desejar e da maneira que desejar, tornando a linguagem mais dinâmica. A organização de um arquivo XML é auto descritiva, ou seja, a própria organização de um documento XML fornece informações suficientes para descrever as estruturas de dados, os valores e os nomes dos campos. Essa organização facilita a leitura dos dados tanto por humanos, quanto por computadores. Essas características são essenciais quando desejamos coletar dados de aplicações sobre as quais nada conhecemos de maneira rápida e eficiente. (W3SCHOOLS,)

2.8.2 CONCEITO DE WEB SERVICE

Um *Web Service* fornece uma interface de serviço que permite aos clientes interagirem com servidores de uma maneira mais geral do que acontece com os navegadores web. Os clientes acessam as operações na interface de um serviço web por meio de requisições e respostas formatadas em XML e normalmente transmitidas por HTTP. Para os serviços web, informações adicionais precisam ser descritas, incluindo a codificação e os protocolos de

comunicação em uso e o local do serviço. Os Web Services fornecem uma infra-estrutura para manter uma forma mais rica e mais estruturada de interoperabilidade entre clientes e servidores. Eles fornecem uma base por meio da qual um programa cliente em uma organização pode interagir com um servidor em outra organização sem supervisão humana. A representação de dados externa e o empacotamento das mensagens trocadas entre clientes e serviços web são feitos em XML. Geralmente, um Web Service fornece uma descrição do serviço, a qual inclui uma definição de interface e outras informações, como URL do servidor. Isso é usado como base para um entendimento comum entre cliente e servidor quanto ao serviço oferecido. As operações de um Web Service podem ser fornecidas por uma variedade de recursos diferentes, por exemplo, programas, objetos ou bancos de dados. (W3SCHOOLS, 2017)

3 METODOLOGIA

Este capítulo apresenta a metodologia de desenvolvimento adotada para a realização dos estudos e desenvolvimento técnico do projeto proposto.

3.1 ESTUDO DOS FUNDAMENTOS TEÓRICOS PARA O TRABALHO

O entendimento das tecnologias utilizadas no desenvolvimento das tarefas deste trabalho é um ponto crucial. Conhecer as ferramentas, suas vantagens, desvantagens e limitações ajuda a nortear a realização das várias atividades necessárias para o sucesso do projeto. Como o principal foco deste trabalho é a representação da mobilidade urbana e o auxílio no planejamento necessário para a implementação de melhorias na cidade de Curitiba, também foram estudados os conceitos de cidades inteligentes e os benefícios que plataformas de software trazem para o planejamento da gestão pública. A plataforma Curitiba-Viewport é um exemplo de implementação de uma Cidade Virtual que almeja um grande grau de realismo. Portanto, foram pesquisados artigos e dissertações sobre a utilização e implementação de cidades virtuais, nos mais variados ramos do conhecimento.

3.2 ESTUDO DO AMBIENTE CURITIBA-VIEWPORT DESENVOLVIDA NA UTFPR

No presente momento, a plataforma Curitiba-ViewPort já possui funcionalidades que permitem ao usuário criar um avatar para si e com ele explorar as redondezas da UTFPR em um ambiente totalmente 3D. (LAZARINE GABRIEL JOSé; NASCIMENTO, 2015)

Para o desenvolvimento do módulo de transporte público foi necessário entender algumas das propriedades da plataforma, como por exemplo a escala que os objetos, prédios e pessoas possuem quando dentro do ambiente virtualizado. Outro ponto a ser entendido foi o sistema de coordenadas geográficas utilizado para a localização dos prédios, pessoas e objetos modelados.

Para isso, foi utilizado um computador Servidor da plataforma, que foi disponibilizado

pela UTFPR.

3.3 CONHECIMENTO SOBRE A PLATAFORMA DE DESENVOLVIMENTO DE SOFTWARE UTILIZADA

Uma vez que o projeto Curitiba-ViewPort já estava em andamento, sendo no momento desenvolvido por colaboradores e equipes que estão focando em módulos distintos a serem interligados, foi necessária a familiarização com as ferramentas e técnicas já empregadas, visando a um consenso entre os envolvidos.

O motor de jogo utilizado é o *Unity3D*, e o software de modelagem gráfica é o Blender. A linguagem de programação utilizada nos *scripts* que estão contidos nos módulos já existentes é o C#.

3.4 ESPECIFICAÇÃO DE REQUISITOS PARA O MÓDULO DE MOBILIDADE

Foram realizados estudos sobre a atual plataforma e suas propriedades para determinar os requisitos funcionais e não funcionais do módulo desenvolvido. Após determinar os requisitos, foram definidos os casos de uso e os casos de teste, posteriormente aplicados.

3.5 DESENVOLVIMENTO DOS MODELOS GRÁFICOS E DE ANIMAÇÃO DE VEÍCULOS DE TRANSPORTE COLETIVO

Para a realização da modelagem gráfica dos ônibus de Curitiba, foi utilizado o *Blender* como ferramenta de modelagem e imagens reais como base. A animação de elementos como as portas e o movimento das rodas dos ônibus serão realizados através das ferramentas disponibilizadas pelo *Unity3D*.

3.6 DESENVOLVIMENTO DA INTEGRAÇÃO DOS MODELOS DOS VEÍCULOS NO AMBIENTE DA CIDADE VIRTUAL

O desenvolvimento dos modelos dos veículos e suas animações foram realizados em um módulo independente. Inicialmente, para a integração do módulo com a plataforma Curitiba-Viewport, foram determinadas algumas localidades pelas quais os ônibus poderiam ou não transitar por meio de *Waypoints* determinados manualmente. Esses *Waypoints* representam a rota que os ônibus realizam, onde os ônibus deverão parar e que ponto será considerado o ponto final de uma viagem e inicial da próxima.

3.7 PROJETO E DESENVOLVIMENTO A SINCRONIZAÇÃO DOS VEÍCULOS VIRTUAIS COM O AMBIENTE REAL

Após a realização da integração do módulo desenvolvido com a plataforma, foi desenvolvido um *Web Service* que se comunica com os servidores de dados da URBS para coletar, em tempo real, as posições e a identificação dos ônibus na cidade de Curitiba. Tendo posse desses dados, foram retirados os *Waypoints* previamente determinados e a posição e movimentação individual dos ônibus foi regida pelos dados reais coletados e processados, fazendo com que os ônibus virtuais estejam sincronizados com a real posição que eles ocupam na cidade de Curitiba.

3.8 PLANEJAMENTO E APLICAÇÃO TESTES DE VERIFICAÇÃO

Para verificar o correto funcionamento dos ônibus modelados, foram realizados testes unitários como abrir e fechar porta e mover pneus, como também foram realizados testes de integração com o Curitiba-ViewPort, nos quais os modelos abriem e fecham suas portas para que os avatares dos usuários adentrem os veículos para que então entrem em movimento e, por fim, abram novamente as portas para que os avatares possam deixar o veículo.

3.9 EXECUÇÃO DE EXPERIMENTOS COM O AMBIENTE DESENVOLVIDO

Os experimentos realizados consistiram na utilização da plataforma Curitiba-ViewPort por vários usuários simultâneos, que puderam acompanhar o funcionamento das linhas de ônibus tanto por fora quanto por dentro dos veículos. Foram realizados dois tipos de experimentos: experimentos nos quais os ônibus seguem rotas pré-determinadas por *Waypoints* e experimentos nos quais as posições reais e virtuais estavam sincronizadas.

4 RECURSOS DE HARDWARE E SOFTWARE

Para a realização das atividades de modelagem, animação, programação e testes de integração, foi necessária a utilização de computadores com o sistema operacional *Windows* instalado, com processador I3 de 1.6 GHz, 4 GB de memória RAM, 1 GB de placa de vídeo e espaço em disco de 50 GBs. Essa configuração retrata o mínimo necessário para instalação e funcionamento dos seguintes softwares: *Windows* 10, *Unity 3D* versão 5.3.4, *Visual Studio 2015 Community*, *Blender* versão 2.7.7, servidor de banco de dados *SQL Server Express 2014*, navegador Web *Google Chrome* e o editor de textos *Microsoft Word 2015*. Já para o funcionamento da plataforma Curitiba-ViewPort foi necessário um computador Servidor com processador de I5 de 2,0 GHz, 4 GBs de memória RAM, 1 GB de placa de vídeo e espaço de disco de 50 GBs. Essa configuração suporta os softwares necessários para o correto funcionamento do servidor, que são: *Windows 10*, *Unity 3D* versão 5.3.4, *SQL Server Express 2014*, navegador *Web Google Chrome*.

5 DESENVOLVIMENTO

Este capítulo apresenta o desenvolvimento tanto do módulo de transporte urbano proposto, quanto das aplicações externas necessárias para o correto funcionamento do projeto.

5.1 VISÃO GERAL DO SISTEMA

O módulo de transporte coletivo é um sistema dividido em três partes:

- Módulo de transporte: ônibus modelado no Blender e animado no Unity 3D, que faz requisições da atual posição dos veículos de transporte coletivo na cidade de Curitiba, a fim de efetuar os cálculos de ajuste de posição na plataforma Curitiba-ViewPort.
- Aplicação web: essa aplicação coleta os dados de geo-posicionamento dos ônibus da cidade de Curitiba por meio de um Web Service para, então, processar esses dados de maneira que possam ser utilizados pelo módulo de transporte. Os dados obtidos são guardados em uma base de dados que é utilizada como consulta histórica, no caso de interrupção da comunicação com a URBS.
- Aplicação desktop: Aplicação que permite a configuração do banco de dados histórico, bem como a importação de novos dados, como a limpeza da base de dados existente, em caso de necessidade de atualização dos dados.

Ao obter-se os dados históricos de posição GPS, foi observado o fato de que os arquivos de texto estavam no formato JSON e que o mesmo arquivo possuía dados sobre várias linhas de ônibus, inclusive os dados da linha Santa Cândida - Capão Raso que foi a linha escolhida para a avaliação de adequação da aplicação. Processar o arquivo de texto foi um desafio computacional, pois o arquivo escolhido possuía quase 50 milhões de linhas de texto. Para processar o arquivo, separando apenas as informações da linha de ônibus desejada e inserir os dados em um banco de dados auxiliar, foi construída uma aplicação externa de apoio. A aplicação, feita na linguagem de programação C#, permitiu parametrizar os dados que deveriam

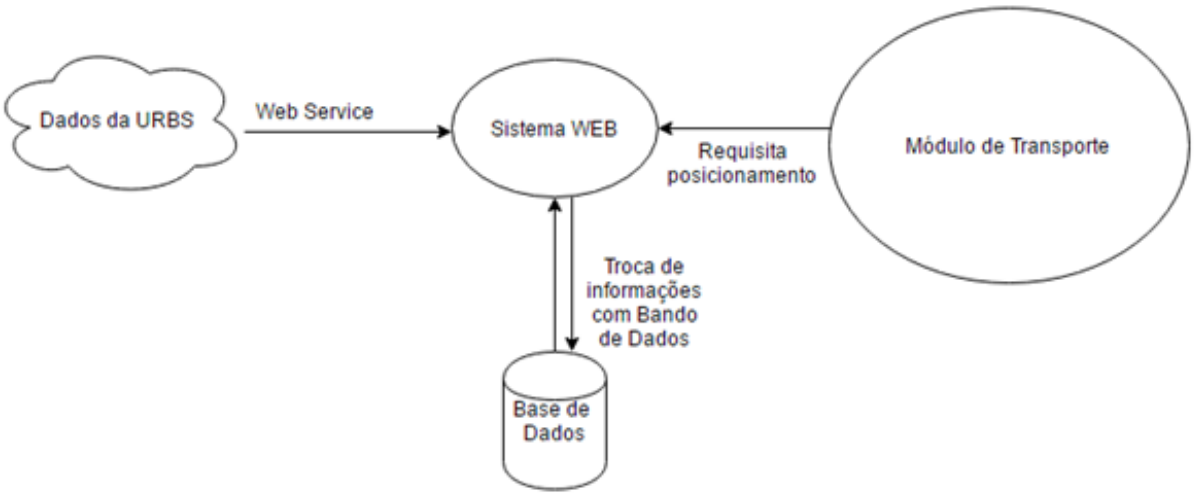

Figura 3: Visão Geral do Sistema - Dados são fornecidos em tempo real pela URBS.

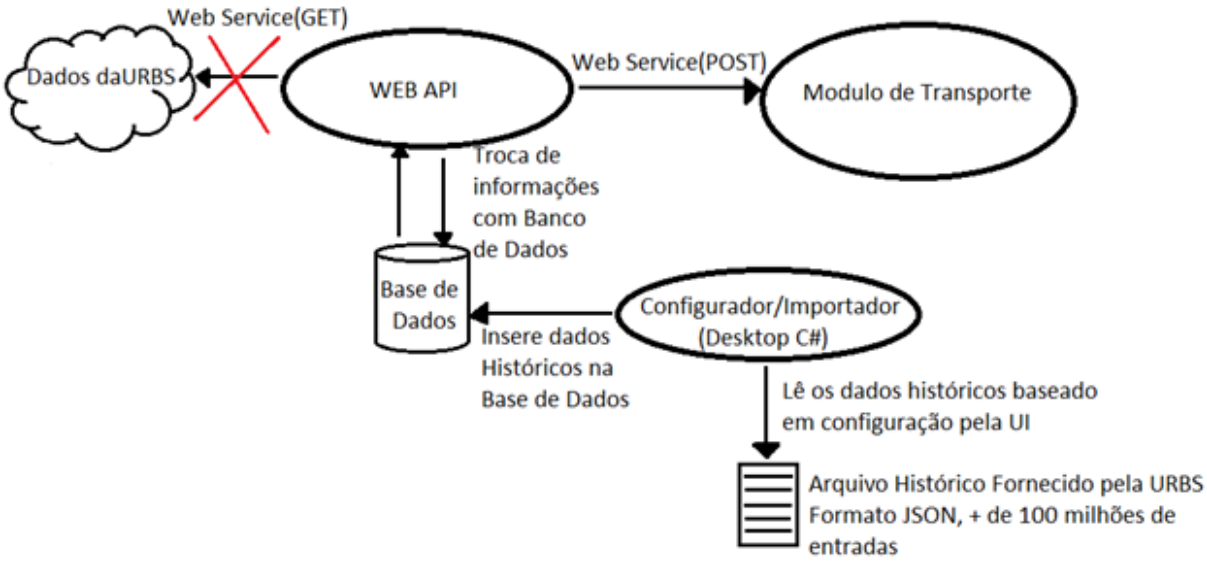

Figura 4: Visão do Sistema que utilizará dados históricos ao invés de dados em tempo real.

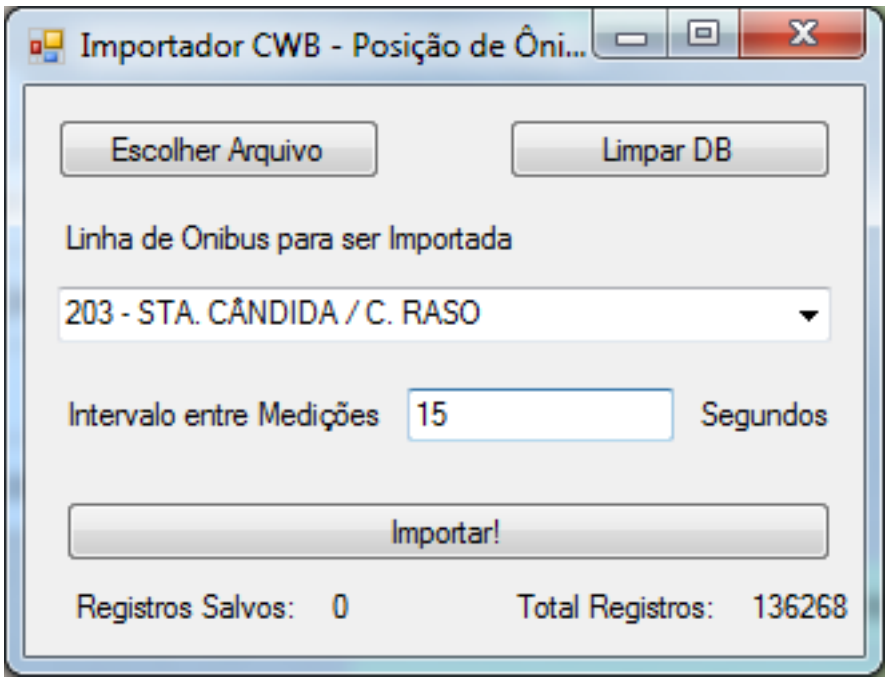

Figura 5: Aplicação parametrizável que realiza filtragem dos dados e insere no banco de dados auxiliar.

ser filtrados e inseridos no banco de dados. Dessa forma, a quantidade imensa de dados de um único arquivo pode ser processada e filtrada para 171 mil entradas no banco de dados.

Após a preparação do banco de dados, foi necessária a construção de uma WEB API com a única função de receber chamadas GET do UNITY 3D e retornar os dados de acordo com os parâmetros de Data e de identificação do ônibus desejado.

5.2 MODELAGEM DO SOFTWARE

5.2.1 DIAGRAMA DA BASE DE DADOS

A modelagem do Banco de Dados utilizado pela aplicação que obtém as posições de GPS dos ônibus levou em conta, principalmente, a nomenclatura de dados utilizada pela URBS. Essa modelagem buscou ser a mais simples possível, com o intuito de prezar pela eficiência e velocidade das buscas realizadas pelo módulo de transporte desenvolvido.

5.2.2 DIAGRAMA DE CLASSES

O diagrama de classes a seguir contempla todas as classes dos projetos complementares desenvolvidos com o objetivo de minimizar as funcionalidades no módulo de transporte.

Pela figura 7 pode-se perceber que tanto o projeto de *WEB API (Service)* quanto o

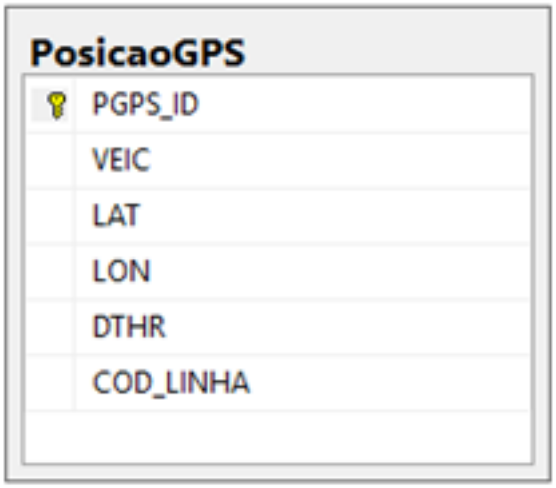

Figura 6: Diagrama de Entidade.

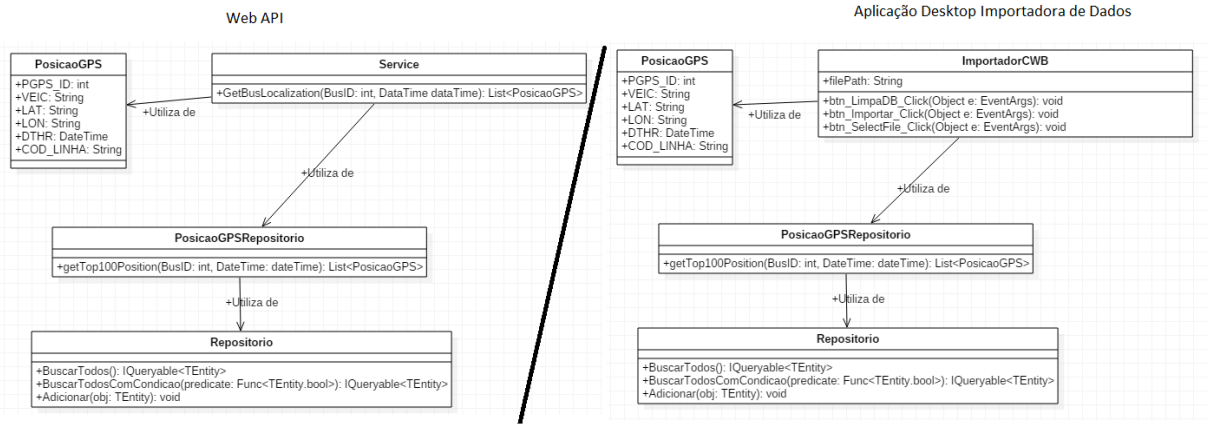

Figura 7: Diagrama de Classes.

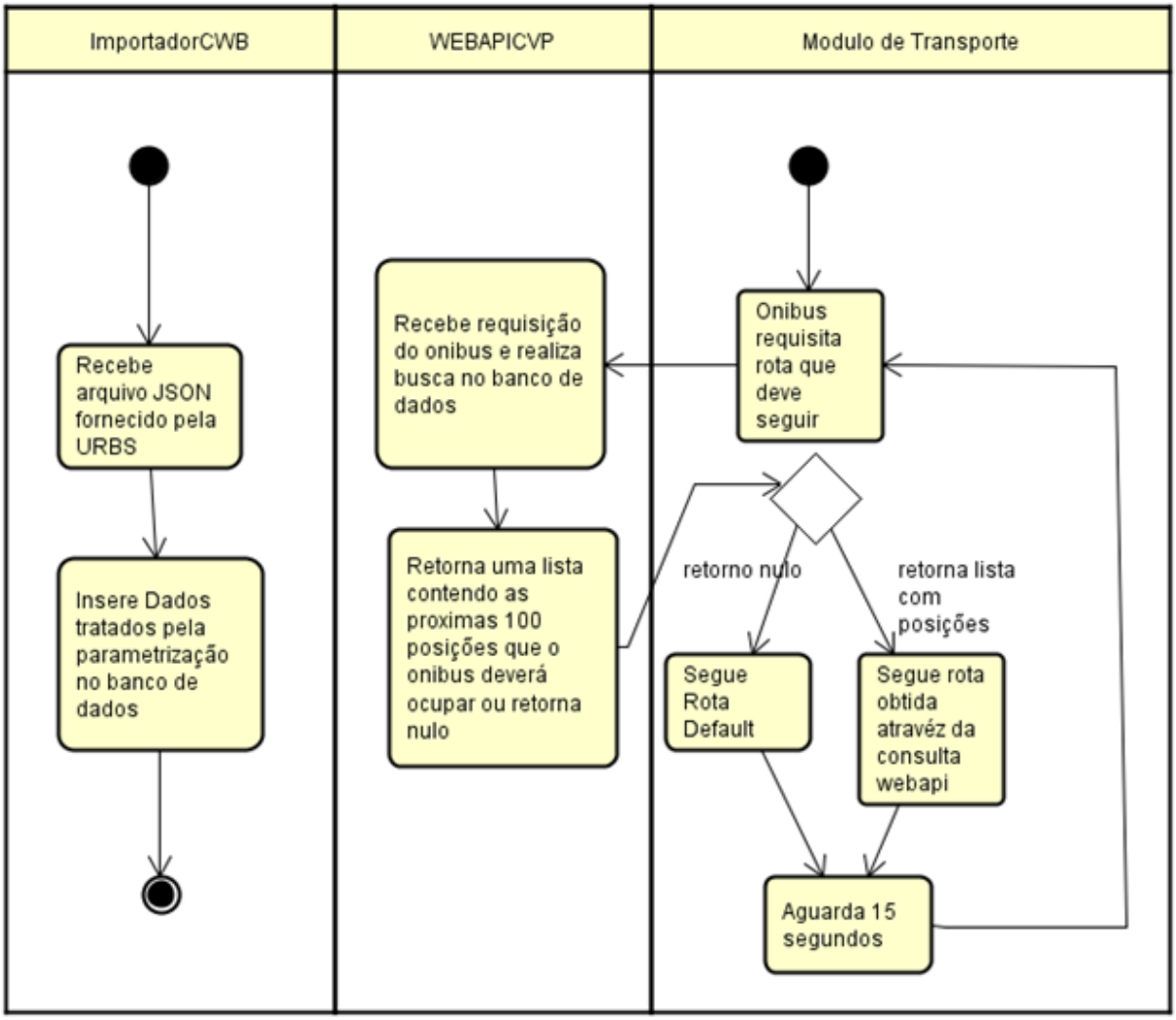

Figura 8: Diagrama de atividades do Módulo de transporte e seus componentes externos.

projeto *Windows Forms (ImportadorCWB)* utilizam a mesma estrutura básica de repositórios. Sendo assim, possuem uma classe derivada da Entidade PosicaoGPS (originada pelo banco de dados) e seu repositório de métodos de acesso aos dados. Dessa forma, a reutilização de código entre diferentes projetos, rodando em instâncias distintas, pode ser percebida.

5.2.3 DIAGRAMA DE ATIVIDADES

O funcionamento padrão do módulo de transporte é dado da seguinte maneira:

- A inserção dos dados no banco de dados ocorre por meio da aplicação externa ImportadorCWB, na qual o usuário deverá selecionar arquivos de texto no padrão da URBS, com os resultados desejados.
- A WEBAPI que consulta dados no banco de dados retorna ao módulo de transporte uma

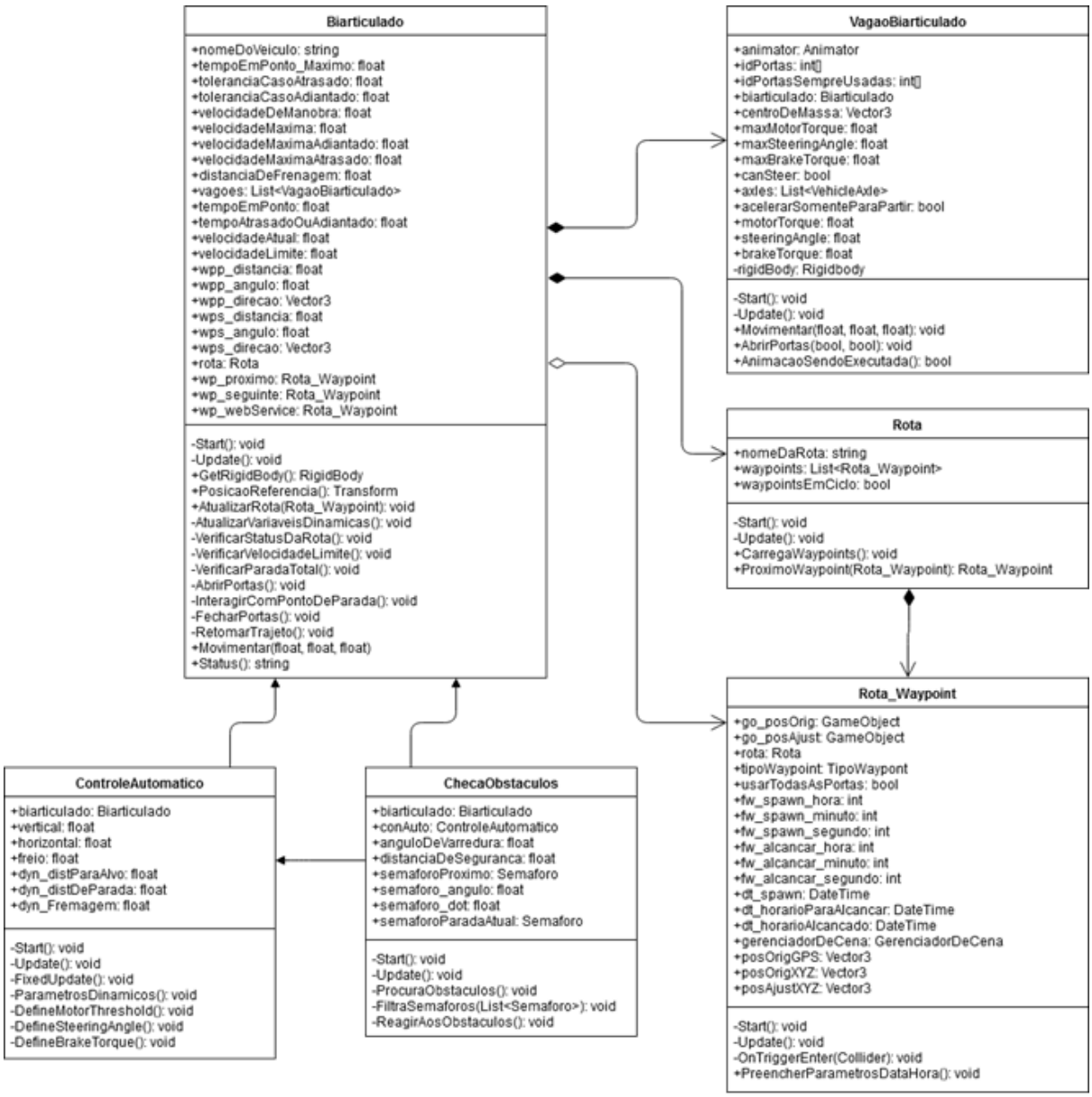

Figura 9: Diagrama de classes necessárias para o funcionamento básico do ônibus.

lista contendo posições de GPS relativas a Linha e ao horário requisitados. Caso os dados não existam no banco de dados, a lista retorna vazia.

- Caso não existam dados de posições GPS para o módulo de transporte se basear, os veículos seguirão a rota padrão pré-definida.

Na figura 9 estão ilustradas apenas algumas das classes implementadas, voltadas à movimentação do biarticulado. Abaixo está uma breve explicação da importância de cada uma neste projeto:

- **Biarticulado**: classe principal utilizada para identificar um objeto como um biarticulado,

vinculando a este os dados dos vagões que compõem fisicamente o veículo, da rota padrão adotada e dos *Waypoints* atuais sendo alvejados. É nesta classe que estão os métodos que ditam quanto e como o mesmo deve se deslocar, repassando os respectivos valores para as rodas informadas na classe VagaoBiarticulado.

- **VagaoBiarticulado**: usada para registrar os passageiros, realizar a animação da abertura e fechamento das portas e também para acionar as rodas conforme os parâmetros de movimentação recebidos.

- **Rota**: contém uma listagem dos *Waypoints* que compõem uma rota, destinados à representação da rota praticada por um veículo de transporte coletivo

- **Rota_Waypoint**: são os dados de um dos *Waypoints*, seja este pertencente a uma rota ou gerado dinamicamente através dos dados de posicionamento do veículo em questão. É neste que estão as datas e horários de quando o mesmo foi registrado pela URBS e, no caso, do horário em que o objeto foi gerado durante a execução da simulação.

- **ControleAutomatico**: nesta classe estão os métodos utilizados para interpretar a rota adotada, direcionando o veículo ao próximo *Waypoint* e regulando a velocidade e frenagem.

- **ChecaObstaculos**: quando esta classe é utilizada, o veículo dotado da mesma não somente irá trafegar pela sua rota mas também estará atento a obstáculos em seu caminho. É feita a verificação dos semáforos, incluindo o estado e se estes dizem respeito à rua sendo trafegada. Também é feita a verificação para veículos próximos, identificando a velocidade relativa entre os mesmos.

5.3 REQUISITOS FUNCIONAIS

Esta seção descreve os requisitos funcionais estabelecidos para o projeto do módulo de transporte urbano. Estes requisitos foram especificados a partir de reuniões com a equipe de desenvolvimento do sistema Curitiba ViewPort, que representam os stakeholders para este projeto. Os requisitos funcionais definidos são:

RF01 - O ônibus deverá seguir um caminho pré-definido

Os ônibus seguem um caminho pré-definido utilizando de *Waypoints* que são inicialmente e manualmente inseridos na base de dados dos mapas do Curitiba-ViewPort. Essa inserção é realizada de acordo com as rotas informadas pelo *site* oficial da URBS para as

linhas de ônibus modeladas, incluindo os pontos de embarque e desembarque de passageiros. Todas as unidades de transporte de uma mesma linha percorrem o mesmo caminho e param nos mesmos pontos de embarque e desembarque de passageiros. Este requisito se aproxima muito da real situação das linhas de ônibus na cidade pois, somente em raras situações de obras ou de acidentes no trânsito, ocorre alteração temporária no trajeto percorrido pelos veículos de transporte coletivo.O **RNF06** descreve as propriedades de um caminho de transporte.

RF02 - Identificar e parar nos pontos de ônibus

No momento em que a localização do ônibus for uma das localizações determinadas como "ponto de ônibus", o veículo deve desacelerar, parar e abrir as portas durante um intervalo de tempo de 10 segundos, fechar as portas e então acelerar e prosseguir com sua rota até o próximo ponto de referência.

RF03 - Respeitar a sinalização eletrônica (semáforo)

Quando o ônibus se aproximar de um cruzamento de ruas, ele deve ser capaz de verificar o estado do semáforo e decidir qual ação tomar. Caso o semáforo esteja no estado "fechado", o ônibus deverá desacelerar e parar próximo a ele até que o estado deste mude para "aberto". Caso o semáforo já esteja no estado "aberto", o ônibus continua com seu movimento normal. Essa verificação de estado é feita constantemente, enquanto o ônibus estiver a uma distância mínima e situado diante de um semáforo identificado como aquele que compartilha de mesma direção e sentido do veículo.

RF04 - Realizar curvas

O veículo deve ser capaz de realizar curvas da maneira suave. Considerando que duas das linhas escolhidas consistem no trajeto de veículos de transporte coletivo do tipo biarticulado, ou seja, que possuem sanfonas interligantes entre um carro e outro, ambos os carros devem ser capazes de realizar a curva da mesma forma.

RF05 - Movimentar as rodas

O ônibus deve movimentar suas rodas de maneira contínua e com velocidade compatível com a do veículo. Quando o movimento for em linha reta, as rodas devem apenas girar indicando que o movimento é para frente e devem rotacionar no sentido horizontal apenas quando o ônibus estiver realizando uma curva com angulação compatível.

RF06 - Surgir em um ponto e Desaparecer em outro

Considerando que a modelagem de Curitiba-Viewport ainda não contempla a totalidade do território da cidade de Curitiba, é necessário que os veículos apareçam de um

determinado ponto identificado como início, e que pertença à rota definida para o veículo seguir. Da mesma maneira, o veículo deve desaparecer no último ponto de sua rota definida, aparecendo novamente no local inicial após um certo intervalo de tempo para recomeçar o trajeto.

RF07 - Seguir o itinerário previsto

Os ônibus devem seguir o itinerário definido conforme o *site* oficial da URBS da forma mais aproximada possível, tanto em um sentido quanto no outro. Para isto, os ônibus irão, periodicamente, consultar um Web Service que fornece as coordenadas atuais destes. Caso esse Web Service esteja indisponível, o módulo poderá consultar também, uma base de dados histórica que foi importada previamente através da leitura dos arquivos JSON, fornecidos pela URBS, que contém as coordenadas de GPS da localização do ônibus e o horário em que ele estava em determinada posição, permitindo desta forma uma boa aproximação com a realidade.

RF08 - Modelagem realística

O veículo deve ser modelado graficamente em três dimensões com alto grau de realismo, porém, buscando otimizar o número de polígonos para não comprometer o desempenho, conforme os requisitos de hardware descritos anteriormente. As linhas de ônibus devem possuir os detalhes e particularidades de cada uma, de forma a facilitar a sua identificação. A cabine do motorista deve apresentar um modelo de condutor sentado em um banco e um volante. As movimentações do motorista e do volante não serão animadas. Já a parte interna do ônibus será modelada como uma superfície plana, sem a inclusão dos bancos e outros elementos comuns para, assim, maximizar o espaço interno e facilitar a movimentação no interior do ônibus.

RF09 - Comunicar com base de dados para obter informações sobre a sua posição atual

Os ônibus devem ser capazes de consultar qual deveria ser sua verdadeira localização em uma base de dados construída e populada anteriormente com dados fornecidos pela URBS. Considerando que a coleta de dados pela URBS é realizada a cada intervalo de 15 segundos, será realizada a consulta à base de dados também a cada 15 segundos. No caso da utilização de dados históricos, é possível configurar o intervalo de tempo entre as medições no momento da importação dos dados. A consulta será feita por um Web Service (conforme **RNF07**), que consultará a base de dados e retornará esses dados para o módulo. O ônibus então irá modificar a sua velocidade de acordo com cada situação a seguir:

- Posição real está mais adiantada que a posição atual: o ônibus irá calcular a velocidade necessária para alcançar esta posição em tempo hábil, e então irá acelerar o seu

movimento de acordo. Depois disso, o veículo seguirá o movimento com sua nova velocidade.

- Posição real está mais atrasada que posição atual: o ônibus desconsidera a posição real atual, e irá calcular a velocidade necessária para alcançar a próxima posição em tempo hábil. Na sequência, o veículo seguirá o movimento com sua nova velocidade.
- Posição real igual ou muito próxima a sua posição atual: caso o ônibus se encontre na mesma posição com até 10 metros de diferença para mais ou para menos, o movimento será mantido na mesma velocidade.
- Caso a posição atual esteja indisponível, o movimento do ônibus será realizado de acordo com as posições padrão fornecidas manualmente para a rota.

5.4 REQUISITOS NÃO FUNCIONAIS

RNF01 - Velocidades padronizadas

A velocidade máxima do ônibus será de 60 quilômetros por hora (km/h), e a velocidade a ser utilizada para curvas e demais manobras será de 15 quilômetros por hora.

RNF02 - Utilizar a Ferramenta de modelagem tridimensional Blender

A modelagem do ônibus deverá ser feita utilizando a ferramenta *Blender*, seguindo os padrões de modelagem utilizados na plataforma Curitiba-ViewPort, utilizando a escala padrão em que uma unidade de medição do *Blender* equivale a uma unidade de medição no *Unity 3D*. A modelagem deverá definir tanto o ônibus quanto suas partes constituintes, como por exemplo as rodas, os faróis, as portas e as articulações veículo, caso existam.

RNF03 - Utilizar a Ferramenta de desenvolvimento de jogos Unity 3D

Com o objetivo de facilitar a implantação do Módulo de Transporte desenvolvido, este será construído utilizando-se da mesma ferramenta de desenvolvimentos de jogos que a plataforma Curitiba-ViewPort: o *Unity 3D*.

RNF04 - Utilizar C# para programação dos scripts de animação e para o desenvolvimento das aplicações externas

Os scripts de animação dos objetos deverão ser programados em C#, seguindo os padrões utilizados anteriormente para animação dos objetos e jogadores que atualmente estão presentes na plataforma Curitiba-ViewPort. A aplicação externa de controle de Web

Services deverá também ser desenvolvida utilizando C# Web API. A aplicação de importação e configuração da base de dados histórica será desenvolvida com Windows Forms C#.

RNF05 - Utilizar o banco de dados SQL Server 2014

Para seguir os padrões atuais de desenvolvimento de aplicações externas a plataforma, utilizaremos o SQL Server 2014 para a modelagem, construção e implementação dos bancos de dados necessários para o funcionamento do módulo de transportes.

RNF06 - Formato para identificação das rotas e dos Waypoints

Cada rota representa uma linha de ônibus definida pela URBS, identificada pelos mesmos códigos e nomenclaturas utilizadas por estas. Compondo o trajeto de cada rota existe uma listagem de *Waypoints*, que fornecem dois a dois a direção e o sentido que cada veículo que trafega por esta rota deve adotar quando está se locomovendo por ela. Os *Waypoints* são diferenciados somente pelo fato de serem ou não pontos de parada, para o embarque e desembarque de passageiros.

RNF07 - Web Service e JSON

O Web Service será do tipo *REST* e utilizará o formato de comunicação de dados *JSON*. Este requisito foi definido com base no serviço de coleta de dados de geo-posicionamento da URBS, que é do tipo *REST* e fornece esses dados na forma de um arquivo *JSON*. Manter esse padrão, tanto para a entrada de dados quanto para a saída dos dados já processados para serem consumidos pelo módulo, é desejável.

5.5 COMUNICAÇÃO ENTRE OS MÓDULOS

O módulo de transporte recebe dados externos inerentes ao funcionamento das linhas de ônibus da cidade de Curitiba utilizando-se de um Web Service. Os dados históricos fornecidos pela UTFPR e cedidos pela URBS contêm dados sobre o posicionamento das linhas de ônibus modeladas, horário da coleta do dado e identificador único do ônibus em um intervalo de uma semana. Levando em conta estas restrições será considerado que os horários e posições representam uma média na qual será baseado o itinerário dos ônibus do módulo de transporte desenvolvido. Após a coleta e armazenamento destes dados no banco de dados, será desenvolvida uma aplicação web externa à plataforma Curitiba-ViewPort para fornecer os dados necessários ao correto funcionamento do módulo de transporte. Essa aplicação web será um *Web Service* do tipo *REST* e funcionará da seguinte forma:

- Método GET: fornece os dados de posicionamento geográfico previamente obtidos e

guardados no banco de dados para o módulo de transporte. Ele retorna um *JSON* contendo os dados de posição atual (longitude e latitude) e horário em que o ônibus se encontra nesta posição. Esse serviço será solicitado pelo módulo a cada 15 segundos, seguindo o padrão de coleta de dados realizado pela URBS. Após a realização de uma consulta, o sistema irá guardar em uma variável global o identificador primário da última consulta realizada facilitando e agilizando a próxima consulta pois os dados foram inseridos de forma sequencial.

5.6 MODELAGEM GRÁFICA DOS VEÍCULOS

O projeto desenvolvido neste trabalho de conclusão envolve veículos de transporte público em um ambiente de cidade virtual. A modelagem geométrica da cidade está sendo feita por uma equipe de pesquisa (alunos e professores) da UTFPR, envolvidos no projeto Curitiba – Viewport. Neste trabalho de conclusão de curso, incluiu-se a modelagem dos veículos de transporte. No escopo deste projeto, a modelagem geométrica estará limitada a um ônibus biarticulado que é um veículo comum na cidade de Curitiba.

A modelagem foi feita utilizando a ferramenta Blender versão 2.78. Essa é uma ferramenta de modelagem rica, utilizada por muitos designers e desenvolvedores, que disponibiliza uma versão gratuíta. Apesar das muitas possibilidades que este software oferece, a atividade de modelagem geométrica não é o foco da profissão de analista de sistemas, nem é tratada em profundidade no curso de BSI. Desta forma, a modelagem geométrica foi limitada a uma representação tridimensional básica do veículo com detalhes suficientes para criar um envolvimento com o usuário, mas sem conter todos os aspectos estéticos de um veículo real. A figura 10 mostra a modelagem em andamento do biarticulado utilizado.

O processo da modelagem foi realizado de maneira incremental, iniciado com o primeiro carro do biarticulado para que fosse possível ter um modelo para testes de comportamento físico e lógico do projeto o quanto antes. Esta modelagem inicial incluiu as portas, as rampas e as rodas. Posteriormente, foi realizada a modelagem dos demais carros, utilizando os componentes previamente gerados para manter uma uniformidade. A figura 11 ilustra as partes do modelo de veículo criado.

5.7 ANIMAÇÃO GRÁFICA DOS VEÍCULOS

Considerando os requisitos levantados para a integração com o Curitiba - Viewport e o fato do objeto modelado corresponder a um ônibus do tipo biarticulado, a única animação

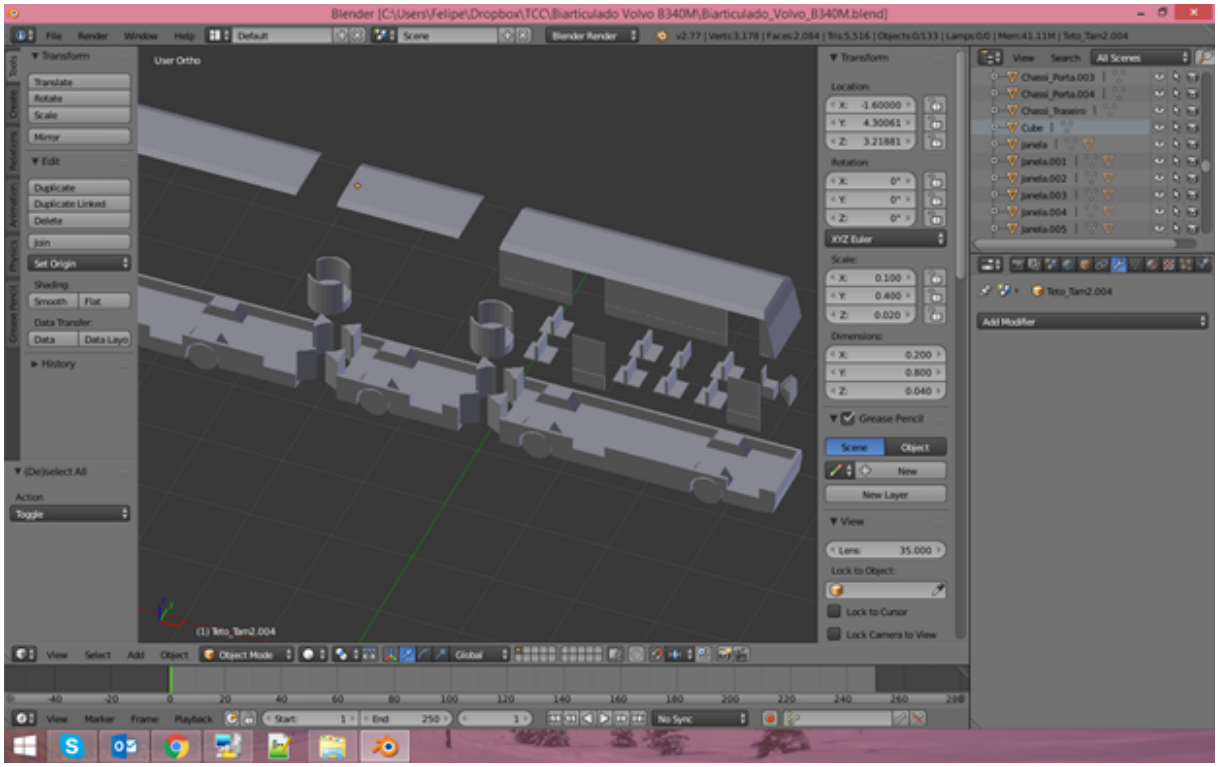

Figura 10: Modelagem em progresso de um biarticulado no software Blender.

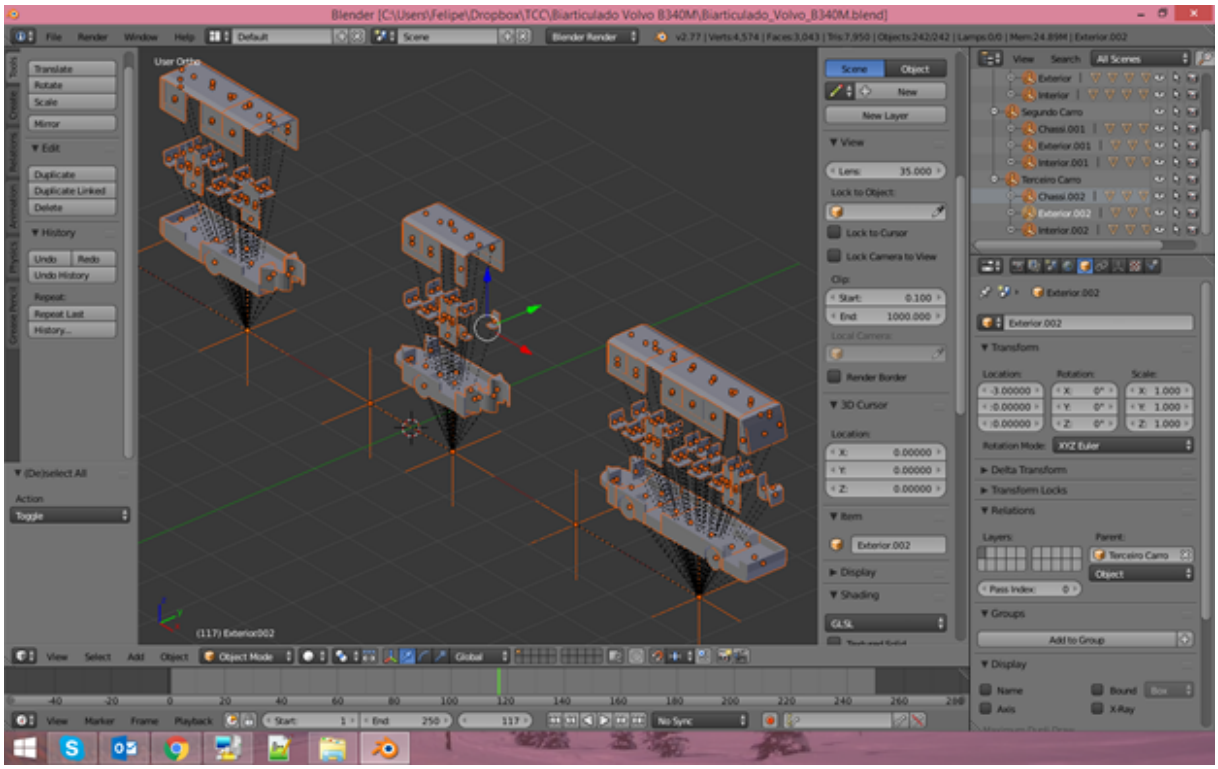

Figura 11: Biarticulado modelado, com componentes destacados espacialmente para fins de visualização.

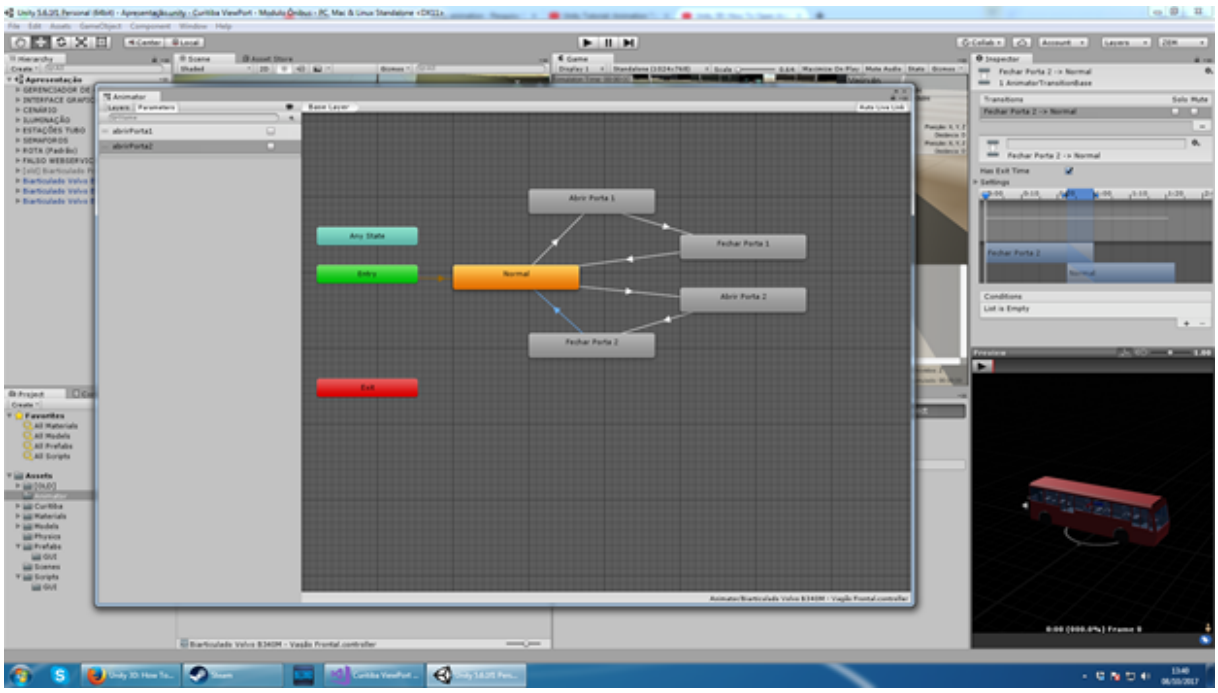

Figura 12: Utilização da funcionalidade Animator, para o gerenciamento dos estados possíveis das animações implementadas.

necessária é a abertura e fechamento das portas de embarque e desembarque de passageiros, bem como das rampas que auxiliam nestas atividades. A rotação das rodas, que poderia ser considerada como mais uma animação, na verdade é realizada pela própria engine física fornecida pelo Unity 3D, uma vez que é o torque das rodas que concede velocidade ao objeto sendo representado.

Sendo assim, a animação das portas e rampas se fez pelo recurso Animator, do próprio Unity 3D, uma vez que não houve necessidade de utilizar outros recursos para a geração de animações mais complexas.

5.8 DESENVOLVIMENTO DA MOVIMENTAÇÃO DO ÔNIBUS

A movimentação do ônibus por um trajeto pré-definido, baseado tanto na rota previamente estabelecida para cada linha de ônibus quanto nos dados de posicionamento registrados que foram fornecidos para este trabalho, também se deu de maneira incremental.

Inicialmente foi concebido um veículo movimentado por meio do controle de um usuário, já fazendo uso da engine física. Após testes iniciais para validação da simulação física deste, duas alterações foram feitas: a inclusão de *Waypoints* para a formulação de uma rota básica a ser seguida, e o controle automático de movimentação do ônibus de acordo com estes *Waypoints*. O ônibus, então, passou a acelerar quando de frente para o próximo *Waypoint* que

deveria alcançar, e a realizar curvas quando não estava alinhado com nenhum destes.

Tendo então um ônibus rudimentarmente automatizado, os esforços se voltaram para a implementação de controles da velocidade, visando representar um comportamento de acordo com a realidade. Foram adicionadas as seguintes considerações:

- Distâncias dinâmicas de parada, nas quais o veículo avalia a distância para o próximo *Waypoint* e a sua própria velocidade atual para determinar o momento correto em que deve acionar seus freios;
- A real necessidade de se aplicar os freios, mesmo que não totalmente, de acordo com o posicionamento dos *Waypoints* próximos:
  - *Waypoints* localizados em um trecho retilíneo não necessitam que haja frenagem do veículo;
  - *Waypoints* localizados em ângulos diferentes em relação à posição do ônibus determinam a frenagem a ser aplicada, diretamente proporcional ao ângulo entre o *Waypoint* atual e o próximo.
  - *Waypoints* especiais demarcados como pontos de parada, onde o veículo deve parar totalmente e efetuar as animações de abertura e fechamento das portas, representando os pontos de ônibus reais.

Com isto, já era possível uma representação, ainda que básica, de um ônibus trafegando por sua rota diária. O próximo passo foi a adição de verificações para obstáculos que pudessem influenciar nas decisões dos ônibus automatizados.

Na verificação de obstáculos, fez-se necessária a apuração de dois casos especiais de obstáculos: semáforos e outros veículos. Para os semáforos, é necessário confirmar que está fechado para a rua e para o sentido em que o ônibus está trafegando. Para os demais veículos, a velocidade relativa entre eles é que determina se é possível movimentar-se sem que haja colisão. Na figura 13 está sendo ilustrada a inclusão de semáforos no projeto e como o veículo reage diante destes.

5.9 DESENVOLVIMENTO DO INTERPRETADOR DE ROTA

Para o desenvolvimento do ônibus automatizado, foi necessário o desenvolvimento do seu interpretador de *Waypoints*, o qual solicita e interpreta as posições a serem alcançadas pelo veículo durante a simulação.

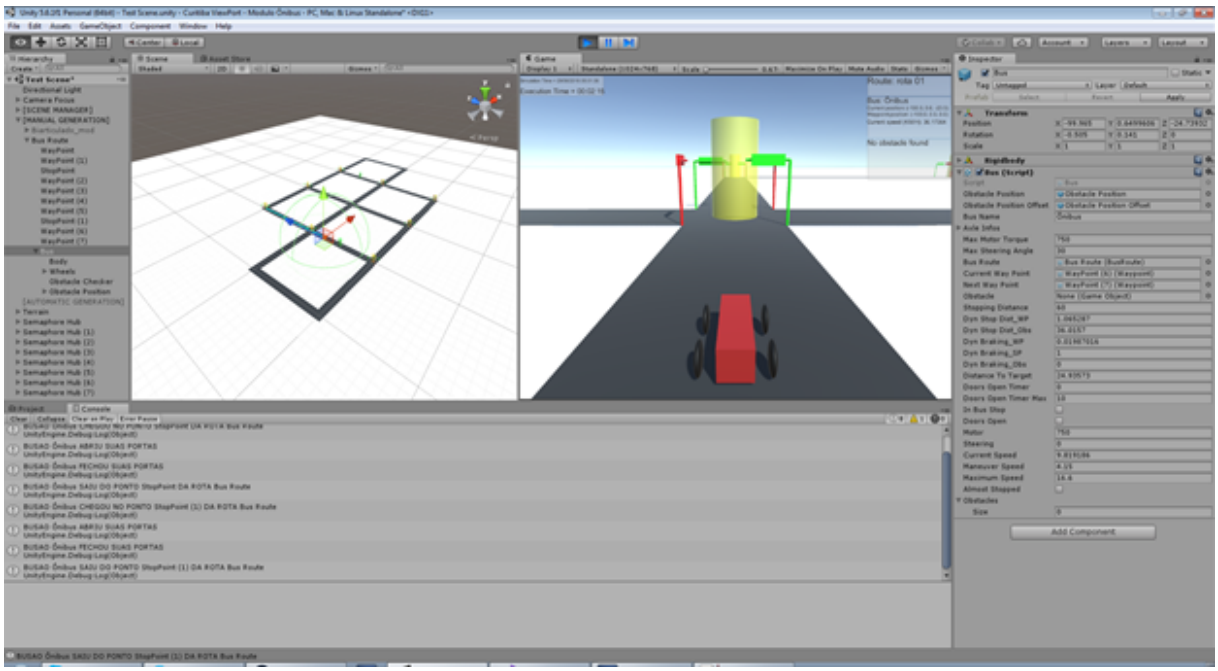

Figura 13: Projeto sendo executado dentro do Unity, retornando feedback visual e textual do comportamento de um veículo automatizado.

Partindo do princípio que todos os ônibus de qualquer rota devem, por padrão, seguir os *Waypoints* de suas respectivas rotas, inicialmente foi concebido um interpretador com dados inseridos manualmente. Com este, foram possíveis os testes iniciais do comportamento automatizado do ônibus. Mas logo que o veículo foi aperfeiçoado com detecção de obstáculos e maneiras de evitá-los, tornou-se necessária a utilização das posições fornecidas para este trabalho, obtidas de maneira dinâmica.

O interpretador passou a realizar consultas ao *Web Service* a cada 15 segundos, visando obter uma lista com as próximas 100 posições relativas ao momento atual da simulação. No caso de receber essa listagem, o interpretador adota estas posições como parte da rota a ser seguida individualmente para determinado veículo. Mas, para o caso de nada ser retornado, então retoma-se o uso das posições manuais, que agora formam rota padrão a ser adotada na falta do *Web Service*.

Porém, nem todas as posições retornadas na listagem recebida encontravam-se suficientemente próximas da rota esperada. Isso se deve não somente a eventuais perdas de precisão comuns aos dispositivos de posicionamento GPS, mas também ao fato de que estas coordenadas GPS, que representam uma posição em um espaço tridimensional, estão sendo convertidas para um plano bidimensional. Logo, é esperado que haja discrepâncias entre as posições e o que se está simulando. Portanto, para amenizar estas divergências, foi criada uma função para a aproximação destes pontos em posições condizentes com o esperado, de acordo

com a própria rota padrão. Assim foi possível garantir que, independente do tempo de medição entre um ponto e outro, o veículo possa alcançá-los sem realizar movimentos irreais.

6 DIFICULDADES ENCONTRADAS

Este capítulo apresenta os principais problemas encontrados ao longo do desenvolvimento deste trabalho, bem como as soluções encontradas para eliminá-los ou ao menos mitigá-los.

6.1 EXTRAÇÃO DOS DADOS HISTÓRICOS

A impossibilidade de utilização de dados em tempo real fornecidos diretamente pela URBS para o desenvolvimento do módulo de transporte causou mudanças tanto na arquitetura inicial quanto na forma de implementação da solução como um todo. A arquitetura final inclui uma aplicação externa escrita em C# que realiza a leitura dos arquivos históricos (em formato JSON) fornecidos pela URBS. Essa aplicação filtra apenas as informações escolhidas previamente e as insere no banco de dados. Uma segunda aplicação (WEBAPI) realiza as consultas de posição GPS baseadas na identificação única do ônibus e da data e horário desejado.

6.2 DOCUMENTAÇÃO INSUFICIENTE DOS RECURSOS DO UNITY

Conforme a movimentação do biarticulado foi se aprimorando, houve a necessidade de também aprimorar a forma como o veículo se deslocava, no sentido de prover maior realismo nas interações físicas entre os objetos em cena. Isto se deu em especial para dois aspectos deste trabalho: as rodas do veículo, criadas a partir de componentes *WheelCollider*, e as juntas utilizadas nas articulações, criadas a partir de componentes *HingeJoint*. Contudo, a documentação oficial do Unity para ambos os recursos se mostrou superficial demais.

Quanto às rodas, a documentação fornece um passo-a-passo básico de como preparar um veículo rudimentar controlado manualmente. Com base nesta documentação foram desenvolvidos os testes iniciais, e a partir daí houve as iterações seguintes que vieram a adicionar novas funcionalidades.

Porém, no momento em que foi gerado o biarticulado em si, composto de 3 objetos dotados de rodas e interconectados por juntas, todas as funcionalidades relacionadas à locomoção do veículo pararam de funcionar corretamente. Nenhuma relação entre o uso de *WheelCollider* com *HingeJoint* estava clara como a causadora das falhas geradas. Após diversas tentativas com reajustes de parâmetros e revisão das funcionalidades implementadas, foram identificados os dois casos em que as falhas na locomoção do veículo se davam:

- Um objeto com rodas é incapaz de se locomover se estiver preso a outros objetos, mesmo que estes também tenham rodas, a não ser que o peso dos demais objetos seja muito baixo em relação a ele;
- Um objeto com rodas é incapaz de se locomover se estiver preso a outros objetos, mesmo que estes também tenham rodas, a não ser que os demais objetos também estejam tentando se locomover.

Mesmo fazendo uso da Internet, buscando por problemas similares e as soluções que foram encontradas para cada caso, não foi possível identificar o motivo desta interação específica entre os objetos e seus componentes. Os usuários do Unity acabavam por relatar que adotaram outras soluções para o problema, que não resolvem o problema por completo mas que são suficientes para ocultar a sua ocorrência.

A solução adotada para este trabalho foi permitir que os carros intermediários e traseiros do biarticulado pudessem acelerar por conta própria, exercendo suficiente torque nas rodas para que o veículo como um todo pudesse sair do lugar. Com o veículo já em movimento, os problemas encontrados deixam de ocorrer.

7 PREPARO DA ROTA E DO CENÁRIO PARA TESTES E DEMONSTRAÇÃO

A linha de ônibus escolhida para a modelagem inicial foi a linha **Santa Cândida - Capão Raso**, por ser um dos trajetos de biarticulados que transitam em frente a UTFPR. Uma cena foi criada no Unity, fazendo uso de formas geométricas básicas, para representar um segmento do trajeto e as edificações encontradas ao longo dele. A rota padrão, adotada pelos biarticulados, foi gerada com *Waypoints* também posicionados ao longo deste trajeto.

De acordo com os dados de posição GPS que foram cedidos pela URBS foi possível reproduzir o caminho e os horários de veículos específicos, fazendo uso de ajustes para a correção de eventuais discrepâncias nos posicionamentos reais.

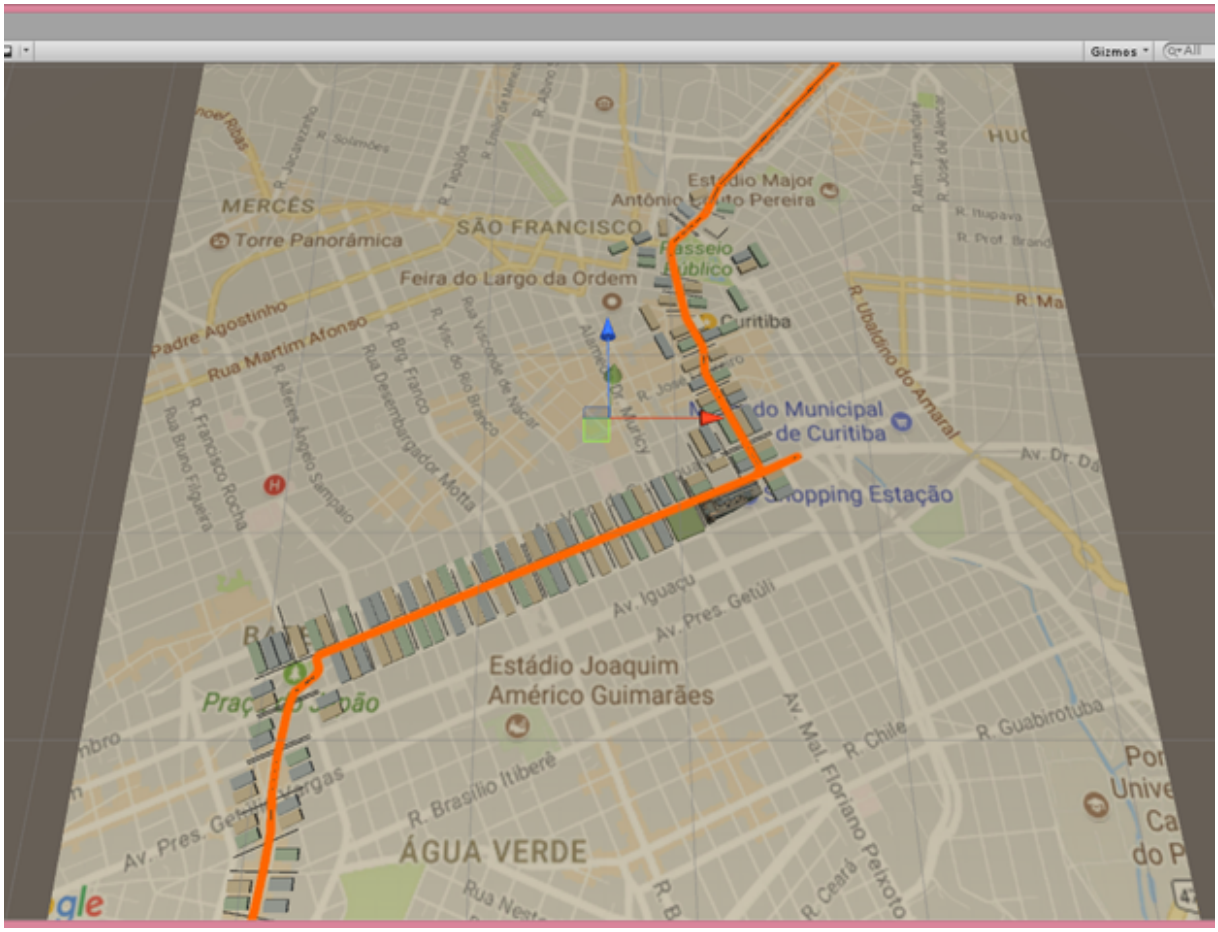

Figura 14: Segmento do trajeto da linha Santa Cândida – Capão Raso.

8 CONCLUSÃO

Com o término do trabalho e a partir de uma análise dos resultados, é possível concluir que o projeto obteve sucesso em seu principal objetivo: construir um módulo de transporte coletivo para o Curitiba-ViewPort, que permite a visualização e interação dos usuários com o ônibus biarticulado desenvolvido e que a posição deste esteja representando a real posição de um dos ônibus utilizados da cidade de Curitiba. Mesmo que tenham sido utilizados dados históricos, por questões de disponibilidade, é possível a conversão para dados de posicionamentos em tempo real, desde que estes possam ser adequadamente fornecidos pela URBS, senão por outras fontes.

Quanto à simulação do tráfego dos biarticulados, assim como dos demais veículos presentes na cidade, a fidelidade com que estes serão representados ao passar do tempo de determinada data depende de mais dados do que somente do posicionamento GPS. Isto foi observado durante os testes realizados e na construção das funcionalidades implementadas, onde em alguns casos como, por exemplo, identificar a real velocidade adotada por um veículo em determinado espaço de tempo, requer-se que sejam considerados outros fatores, como semáforos ao longo do caminho, demais veículos compartilhando as ruas e canaletas, alterações pontuais de trajeto e outros obstáculos existentes. Ainda assim, mesmo utilizando somente dos dados de posicionamento, é possível demonstrar o quão atrasado ou adiantado determinado veículo está em relação ao seu itinerário. Identificando estes casos ao longo da execução, é possível adotar alterações na velocidade de modo a compensar as diferenças entre o posicionamento real e o posicionamento simulado.

Desta forma, o projeto Curitiba-ViewPort se aproxima ainda mais da definição de cidade virtual utilizada neste trabalho. Futuramente poderá ocorrer o desenvolvimento de novos módulos para o Curitiba-ViewPort, como o módulo de serviços de emergência (bombeiros, ambulâncias e polícia) e de trânsito de veículos pessoais, fazendo uso deste módulo como base na implementação dos demais veículos necessários.

REFERÊNCIAS

APS, U. T. **Unity Terms of Service**. out 2017. Disponível em: <https://unity3d.com/pt/legal/terms-of-service>.

AUGMENT. **How augmented reality works**. Disponível em: <http://www.augment.com/how-augmented-reality-works/>.

AUMENTADA, P. R. **Portal Realidade Aumentada**. 2016. Disponível em: <http://realidadeaumentada.com.br/>. Acesso em: 29 de junho de 2016.

CITY, N. Y. **BUILDING A SMART + EQUITABLE CITY**. setembro 2015. Disponível em: <http://www1.nyc.gov/assets/forward/documents/NYC-Smart-Equitable-City-Final.pdf>.

COULOURIS, G. et al. **Sistemas Distribuídos - 5ed: Conceitos e Projeto**. [S.l.]: Bookman Editora, 2013. ISBN 9788582600542.

FOUNDATION, B. **Features – Blender**. Disponível em: <https://www.blender.org/features/>.

FRITSCH, D.; KADA, M. Visualisation using game engines. v. 35, p. B5, 06 2004.

GOOGLE. **GoogleEarth documentação oficial**. Disponível em: <https://www.google.com/earth/>.

IBM. **Smarter Cities: New cognitive approaches to long-standing challenges**. 2017. Disponível em: <https://www.ibm.com/smarterplanet/us/en/smarter_cities/overview/>.

JOY, K. **Geometric Modeling Lectures**. 10 2012. Disponível em: <http://graphics.cs.ucdavis.edu/ joy/GeometricModelingLectures/>.

LAZARINE GABRIEL JOSé; NASCIMENTO, H. L. G. d. Q. **Proposição, Especificação e Desenvolvimento de um Software Experimental de Cidade Virtual**. jun 2015.

MESSAOUDI, F.; SIMON, G.; KSENTINI, A. Dissecting games engines: The case of unity3d. In: **2015 International Workshop on Network and Systems Support for Games (NetGames)**. [S.l.: s.n.], 2015. p. 1–6.

MIRANDA, M. J. P. L. **Jogo sério para reabilitação neurocognitiva : cidade Virtual**. Dissertação (Mestrado) — FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO, Julho 2012. Disponível em: <http://hdl.handle.net/10216/65650>.

NEIROTTI, P. et al. Current trends in smart city initiatives: Some stylised facts. **Cities**, v. 38, n. Supplement C, p. 25 – 36, 2014. ISSN 0264-2751. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0264275113001935>.

TECHNOLOGIES, U. **Unity – Game Engine**. 2017. Disponível em: <https://unity3d.com>.

TIBOLA, L.; PEREIRA, C.; TAROUCO, L. Educational labs in 3d virtual worlds: Improve engineer's competencies through practice labs. 10 2014.

W3SCHOOLS. **XML Tutorial**. Disponível em: <https://www.w3schools.com/xml/>.

W3SCHOOLS. **Web Services Tutorial**. 2017. Disponível em: <http://w3schools.sinsixx.com/webservices/default.asp.htm>.