Universidade Federal de Pernambuco

Centro de Informática

Mestrado em Ciência da Computação

ITRANS – SIMULADOR DE TRÂNSITO 3D

por

Paulo Gonçalves de Barros

Recife, Março de 2005.

Universidade Federal de Pernambuco

Centro de Informática

Mestrado em Ciência da Computação

ITRANS – SIMULADOR DE TRÂNSITO 3D

por

Paulo Gonçalves de Barros

Dissertação submetida ao Centro de Informática da Universidade Federal de Pernambuco para cumprimento parcial das exigências para obtenção do título de Mestre em Ciência da Computação.

Orientadora: Judith Kelner

Banca Examinadora:
Dr.Cláudio Kirner;
Dra. Verônica Teichrieb;
Dr. Fernando da Fonseca;
Dra. Judith Kelner.

Recife, Março de 2005.

Barros, Paulo Gonçalves de
ITRANS – Simulador de trânsito 3D / Paulo Gonçalves de Barros. – Recife : O Autor, 2005.
x, 83 folhas : il., fig., tab., quadros.
Dissertação (mestrado) – Universidade Federal de Pernambuco. Cin. Ciência da Computação, 2005.
Inclui bibliografia e apêndice.
1. Ciência da computação – Realidade virtual. 2. Tráfego urbano – Simulação de semáforos e veículos – Modelos matemáticos de microsimulação. 3. Interface tridimensional – Biblioteca gráfica OpenGL. 4. Vias de tráfego – Configuração de Dados – C++ (Linguagem de programação). I. Título.
004.94 CDU (2.ed.) UFPE
006.8 CDD (22.ed.) BC2005-106

RESUMO

O planejamento de tráfego urbano tem um papel fundamental na sociedade atual, pois permite um melhor aproveitamento das vias de tráfego e otimiza o fluxo de veículos e pedestres. Este planejamento também reduz engarrafamentos, períodos de viagem e poluição, ajudando não só a economia, mas a população, através da garantia de uma melhor qualidade de vida.

A fim de solucionar os problemas do tráfego, diversos modelos matemáticos foram criados. O advento do computador fez com que esses modelos passassem a ser não somente construídos, mas também simulados. Tais modelos são chamados de modelos de macro-simulação, pois apresentam o trânsito através dos fluxos em cada rua, ao invés da representação de cada veículo.

Com o aumento da capacidade de processamento dos computadores, novos modelos de simulação computacional foram desenvolvidos e surgiram as micro-simulações, que constroem um modelo para o trânsito baseado na simulação de comportamento de cada entidade de trânsito: os veículos e pedestres.

Dentre esses dois grandes grupos, pode-se dizer que a micro-simulação tem atualmente recebido uma ênfase crescente, e é neste contexto que está inserido o presente trabalho.

O objetivo primordial deste trabalho é o desenvolvimento de uma interface de realidade virtual (RV) para simuladores de trânsito. Para atingir este objetivo, um simulador básico foi desenvolvido com uma interface de realidade virtual tridimensional desktop.

A simulação também possui uma estrutura que servirá de base para o desenvolvimento de um simulador de trânsito com arquitetura distribuída semelhante às desenvolvidas na área de Sistemas Distribuídos.

Procura-se também definir uma interface para simulação de trânsito em ambientes virtuais 3D que busca explorar o potencial obtido de uma terceira dimensão. Essa tarefa demandou não só o estudo aprofundado de modelos de simulação, mas também a análise de interfaces de realidade virtual.

ABSTRACT

Urban traffic planning has a fundamental role in our society, for it improves the use of traffic roads and optimizes the flow of both vehicles and pedestrians. Traffic planning reduces traffic jams, travel times and pollution, not only helping the economy of a city or state, but also citizens by assuring them a better quality of life.

In order to solve the problem of urban traffic, different mathematical models were created. With the advent of computers, these models passed to be virtually built and simulated. Such models are called macro-simulation models, once they represent transit by flows in each street instead of representing the state of each vehicle separately.

With the increase of the processing capacity of computers, new computational models were developed and the micro-simulations appeared. These build a model for vehicle transit by simulating the behaviour of each transit entity, be it either a vehicle or a pedestrian.

Among these two above mentioned groups, it might be said that the micro-simulations are receiving more attention nowadays. It is in this group that the present work is inserted.

The primary goal of this work is to develop a Virtual Reality (VR) interface for transit simulators. To accomplish this objective, a basic simulator was developed with a 3D Virtual Reality desktop interface.

The simulation was structured in such a way to serve as a basis for the development of a transit simulator with a distributed architecture, similar to the ones currently being proposed in the area of Distributed Systems.

This work attempts to define an interface for transit simulation in three-dimensional virtual environments capable of exploring the potential obtained by the addition of a third dimension. It demanded not only the profound study of simulation models, but also the analysis of VR interfaces.

ÍNDICE ANALÍTICO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

AGRADECIMENTOS

Gostaria de agradecer à minha família, à qual tenho muito orgulho de pertencer. Agradeço por todo o apoio que sempre me deram, principalmente durante o mestrado.

Gostaria de agradecer também à minha amiga e orientadora, professora Judith Kelner que, desde 1999, tem me apoiado como profissional e me ensinado persistentemente a ser um pesquisador.

Agradeço aos amigos pelo apoio, paciência e compreensão em todas as vezes que não pude comparecer a sessões de RPG, de cinema e também a festas.

Em especial, gostaria de agradecer à minha amada Isabella, pessoa que nutre minha alma com carinho e amor mais que ninguém e a quem dedico essa simples, mas laboriosa obra.

Gostaria de agradecer aos professores Alejandro Frery, Geber Ramalho, Hermano Perrelli e Alex Sandro Gomes, que foram fontes de grande apoio, inspiração e sabedoria durante minha jornada tecnológico-científica nos últimos dois anos. Gostaria de agradecer, também, à Universidade Federal de Pernambuco, ao seu Centro de Informática e à CAPES por terem me possibilitado essa oportunidade de aperfeiçoamento acadêmico.

Por fim, gostaria de agradecer a *.*, a tudo e a todos que, mesmo com gestos impensados e, talvez, imperceptíveis como o bater de asas de uma borboleta, trazem brisas e ventanias para a realidade extraordinária em que eu vivo. Sem essa realidade, eu não seria quem sou nem estaria onde estou.

"Que o tempo que a vida me traz leve as mágoas e deixe a alegria.
Já se vê que esse tempo é a luz nos confins de uma aurora vazia".
(P.G.B.)

GLOSSÁRIO

A

Avatar: entidade responsável por representar o usuário no mundo virtual. Neste trabalho, um avatar será definido como a representação de qualquer objeto ou entidade, seja ele o usuário ou não.

C

CAVE: sala que permite a projeção de ambientes virtuais em suas paredes, além de sua manipulação por interfaces avançadas de Realidade Virtual.

CTTU: **C**ompanhia de **T**rânsito e **T**ransportes **U**rbanos.

D

Dead-reckoning: reconhecimento de estado comum para todos os jogadores durante uma partida envolvendo mais de um jogador. Esse termo se aplica somente a jogos em rede, onde um sincronismo entre diferentes PC se faz necessário.

DEVS: **S**istemas de descrição de **EV**entos **D**iscretos.

DIS: **S**imulação **I**nterativa **D**istribuída.

H

HLA: acrônimo de ***H****igh* ***L****evel* ***A****rchitecture*. É uma arquitetura de propósito geral para simulação com reuso e interoperabilidade.

I

ISA: acrônimo de ***I****ntelligent* ***S****peed* ***A****daptation System*. Sistemas autônomos que ajustam sua velocidade automaticamente de acordo com o limite de velocidade da pista.

ITS: ***I****nstitute for* ***T****ransport* ***S****tudies*. Instituto para estudo de meios de transportes localizado na cidade de Leeds, Reino Unido.

K

KQML: acrônimo de ***K****nowledge* ***Q****uery and* ***M****anipulation* ***L****anguage*. Linguagem utilizada no intercâmbio de conhecimento e informação entre entidades computacionais como agentes.

L

LOD: acrônimo de ***L****evel* ***O****f* ***D****etail*. Indica o nível de detalhe dentre as diversas representações que um objeto pode ter num mundo virtual.

O

Óculos estereoscópicos: óculos que provêem para cada um dos olhos imagens ligeiramente afastadas uma da outra de um ambiente virtual, fornecendo a impressão tridimensional do mundo sendo visualizado.

P

Pathfinding: busca de caminhos para entidades animadas num jogo ou mundo virtual.

R

Replay: rever simulação e voltar a pontos específicos da mesma, como num filme.

S

SACI: **I**nfra-estrutura **S**imples de **C**omunicação entre **A**gentes.

SLX: **L**inguagem de **S**imulação com e**X**tensibilidade.

U

UTC systems: ***U****rban* ***T****raffic* ***C****ontrol systems*. Sistemas eletrônicos que medem e controlam o fluxo de tráfego urbano em vias específicas.

W

World-Up: ferramenta de simulação que permite o uso de interfaces avançadas de Realidade Virtual imersiva em aplicações.

Capítulo 1 :

Introdução

O trânsito intenso e desordenado é um dos principais problemas das grandes cidades brasileiras. Além de promover o engarrafamento, polui o meio ambiente e gera acidentes. Em 2003 foram registrados no Recife 11.722 acidentes, que deixaram 57 mortos e 2.288 feridos, segundo a **CTTU** (Companhia de Trânsito e Transportes Urbanos). A solução para o problema do trânsito ainda é considerada um grande desafio, que tende a aumentar à medida que cresce a frota de carros nas cidades.

Tomando-se como exemplo a cidade do Recife, percebe-se que, durante o ano de 2004, cerca de mil veículos foram acrescidos à frota total a cada mês. Incluindo-se a região metropolitana, este número torna-se ainda maior, como seria de esperar. De acordo com a CTTU, na última década a cidade teve um aumento de 44% na sua frota. Além disto, recebe diariamente parte do volume veicular de toda a região metropolitana, devido à centralidade e polarização dos serviços. A união de todos estes fatores tem causado forte pressão no sistema viário, pressão essa que só tende a aumentar.

O planejamento de tráfego urbano permite um melhor aproveitamento das vias de tráfego e otimiza o fluxo de veículos e pedestres, promovendo a redução de engarrafamentos, tempos de viagens e poluição. Desse modo, o planejamento de tráfego ajuda não só a economia, através da provisão de meios de transportes mais eficientes aos bens de consumo e trabalhadores, como garante uma melhor qualidade de vida à população.

Esse capítulo abordará como o planejamento de tráfego evoluiu ao longo do tempo, além de apresentar as tecnologias utilizadas e características pertinentes a simuladores de trânsito atualmente, inserindo a pesquisa desenvolvida nesse contexto e, por fim, explicando como o conteúdo desse trabalho está organizado nos capítulos seguintes.

1.1. Histórico

Diversos modelos matemáticos foram criados visando minimizar o problema do tráfego. Tais modelos basicamente representam o fluxo de veículos através de setas e pontos num grafo. As setas ou arestas representam as ruas. Os pontos, também chamados de nós, são colocados no

início e fim de cada aresta e representam os cruzamentos ou junções. A cada aresta são adicionadas características, como fluxo, número de faixas, largura de faixas, limite de velocidade e distância. O mesmo ocorre com os nós. Assim, é possível construir um modelo em papel de qualquer estrutura de ruas.

O advento do computador fez com que os modelos de tráfego passassem a ser não somente construídos, mas também simulados no computador. Tais modelos foram chamados de modelos de macro-simulação, pois apresentavam o trânsito através dos fluxos em cada rua, ao invés da representação de cada veículo. Com a simulação computacional, foi possível determinar qual a melhor forma de se construir uma nova rua numa área urbana (Clark, 1997). O traçado da via, o número de faixas que ela deveria conter e o posicionamento de semáforos pôde ser repensado através de séries de simulações até se atingir as expectativas que motivaram sua construção (Macredie, 1996).

Com o aumento da capacidade de processamento dos computadores, novos modelos de simulação computacional foram desenvolvidos, surgindo então as micro-simulações, que constroem um modelo para o trânsito baseado na simulação de comportamento de cada entidade de trânsito como, por exemplo, veículos e pedestres. Os veículos percorrem caminhos distintos, dentro da rede de tráfego, definidos de acordo com o fluxo estimado em cada rua (Owen, 2000).

A simulação tem facilitado muito o trabalho de engenheiros de tráfego, pois permite a visualização e análise de situações críticas ou adversas em vias de tráfego importantes, que são projetadas sem a necessidade das mesmas ocorrerem na realidade.

1.2. Objetivo

O objetivo primordial desse trabalho é desenvolver uma interface de Realidade Virtual (RV) para simuladores de trânsito. Para tanto, um simulador básico será desenvolvido e, nele, uma interface de RV será inserida. Essa aplicação, composta por essa interface, pelo simulador e por um sistema flexível de configuração de malhas de tráfego, é denominada ITranS.

Em segundo lugar, pretende-se desenvolver uma estrutura de aplicação que servirá de base para o futuro desenvolvimento de simuladores de trânsito com arquiteturas distribuídas ou paralelas, semelhante às desenvolvidas nas referências (Cameron, 1994) e (Klein, 1998).

1.3. Características Comuns de Simuladores de Tráfego

Atualmente, inúmeros são os grupos de pesquisa e as empresas que estão desenvolvendo software na área de simulação de tráfego. Uma pesquisa realizada pelo **ITS** (*Institute for Transport Studies*), na Universidade de Leeds, Reino Unido (SMARTEST, 2000), revela 57 ferramentas produzidas para lidar com situações de tráfego das mais variadas formas.

De acordo com o ITS, essas ferramentas podem ser classificadas, basicamente, em 4 grupos:

- Modelos urbanos: simulam situações em áreas internas às cidades, onde o fluxo é mais lento e sofre mais interrupções;
- Modelos rodoviários: simulam vias de mais alta velocidade na periferia de áreas urbanas, onde os semáforos são mais esparsos e a velocidade dos veículos é mais alta;
- Modelos mistos: tentam mesclar os dois modelos acima, sendo úteis tanto para simulações urbanas quanto para simulações rodoviárias;
- Modelos de auto-estrada: modelos utilizados para teste de veículos e sistemas de transporte e viagem autônomos, onde há simulação de fluxo sem interrupção por semáforos.

De uma forma geral, os modelos possuem características comuns. A primeira delas é que quase todos têm seu tempo de atualização baseado em unidades discretas, ou seja, em intervalos de um segundo.

Uma segunda característica é a definição da rota dos veículos, especificando-se os pontos de origem e destino. No método tradicional, os pontos intermediários são escolhidos de acordo com o percentual que o fluxo de veículos contribui no fluxo de saída total de um determinado ponto anterior no percurso e ao qual estão ligados. Por exemplo, se 60% do tráfego de um ponto A se dirige para o ponto B, a probabilidade de um veículo, estando no ponto A, ir para o ponto B é também de 0,6. Como esse método não é flexível à mudança de rotas, um outro método, onde cada veículo define a própria rota à medida que se desloca, sabendo inicialmente apenas seu destino, está se tornando cada vez mais popular.

A operação de semáforos é também uma característica comum à maioria dos modelos. Sua implementação nos modelos encontrados é feita de várias formas: definido num módulo em

separado, com uma linguagem de descrição própria, internamente à simulação ou mesmo funcionando a partir de atualizações de dados extraídos da região em tempo real através de sistemas de controle de tráfego urbano (***UTC systems*** - *Urban Traffic Control Systems*).

Poucos são os modelos que consideram pedestres e ciclistas como parte do tráfego e influentes no fluxo das vias mapeadas. Do mesmo modo, na maioria dos modelos o comportamento de transportes públicos não é definido de maneira diferenciada (Liu, 2000).

Geralmente os modelos se preocupam com o fluxo contínuo dos veículos, sem levar em consideração a possibilidade de eles estacionarem, entrarem ou saírem de uma determinada loja ou local, alterando o fluxo em diversos pontos de uma única via de tráfego.

A maioria dos modelos possui como saída alguns indicadores de estado da simulação, como tempo de viagem de cada veículo, variação nesses tempos de viagem, velocidade dos automóveis, formação, localização e tamanho de engarrafamentos durante a simulação.

A simulação geralmente possui parâmetros configuráveis, o que dá flexibilidade aos modelos. Por outro lado, a integração com bancos de dados e outras ferramentas raramente ocorre. A fim de acelerar o processo de simulação, a execução dos eventos nos modelos é até cinco vezes mais rápida que na vida real. Os veículos geralmente se movimentam numa velocidade maior que a normal e os semáforos demoram menos tempo do que deveriam.

A calibração dos modelos – necessária para uma reprodução fidedigna da realidade – existe, mas os dados utilizados nesse processo variam de um modelo para outro. Entre os dados incluídos, estão estatísticas de consumo médio de combustível, grau de emissão de poluentes, limites de aceleração, desaceleração e velocidade dos veículos, além de dados sobre o fluxo de veículos em cada via.

Os modelos são validados comparando-se seus dados de saída com os dados reais obtidos na região. Atualmente, não é dada muita ênfase a essa característica, o que dificulta o aumento da confiança nessas ferramentas.

Com respeito à interface gráfica ainda há muito a se evoluir, apesar de algumas já utilizarem modelos de visualização tridimensionais (Seneviratne, 2001). Não foi encontrado nenhum modelo para esse tipo de simulação que utilizasse equipamentos de RV, além daqueles que simulam a

pilotagem de algum meio de transporte. Também existem modelos não-imersivos em que o usuário dirige um dos veículos no trânsito (Bayarri, 1996).

Na maioria dos modelos, o limite de velocidade dos veículos é fixo para todo o percurso, o que não ocorre na realidade, onde se pode ter uma pista com diferentes limites de velocidade em cada um de seus trechos. Alguns projetos têm tido sucesso em testar limites de velocidade simulando veículos que utilizam o sistema **ISA** (*Intelligent Speed Adaptation System*) (Liu, 2000).

A mudança de faixa já se encontra bem mapeada em alguns modelos, ocorrendo quando há necessidade de mudança de rota, de ultrapassagem, de parada, etc. Em outros, os veículos possuem capacidade de aprendizagem, mudando seu comportamento de uma simulação para a seguinte.

A definição do comportamento dos veículos tem se baseado em modelos psicofísicos (Schulze, 1997), que tentam modelar o movimento dos automóveis e também o comportamento sociológico do trânsito. Isso permite a existência de motoristas com diferentes níveis de agressividade ou passividade no trânsito. Conseqüentemente, obtêm-se veículos com diferentes velocidades, acelerações e disposições para ultrapassagem (Al-Shihabi, 2001).

Com relação à RV, os modelos têm evoluído e estão cada vez mais realistas, seja através do emprego de motores de jogos no processamento dos ambientes (DeLeon, 2000), seja pela definição de técnicas de nível de detalhe mais aprimoradas (De Floriani, 2002). Entretanto, o potencial da interface ainda não é inteiramente aproveitado, por conta, em parte, da utilização dos mesmos paradigmas originados da implementação de interfaces bidimensionais (Stappers, 2000). Assim, definir uma interface para simulação de trânsito em ambientes virtuais tridimensionais é uma tarefa que demanda não só o estudo aprofundado de modelos de simulação, mas também a análise de interfaces de RV.

A indústria de jogos tem contribuído enormemente para o desenvolvimento de aplicações que simulam trânsito de maneira realista. Problemas como obter um ***replay*** de simulações (Wagner, 2004), lidar com o atraso na comunicação entre jogadores (Pantel, 2002), permitir ***dead-reckoning*** e efetuar ***pathfinding*** rapidamente (Smith, 2002) têm sido solucionados de maneira satisfatória pela indústria de entretenimento digital. Embora essas soluções tenham complexidade limitada para não prejudicar o desempenho, elas podem ser parcialmente herdadas e adaptadas para simulações de trânsito (Adzima, 2001).

Outra semelhança entre esses dois tipos de aplicação é que ambientes virtuais extensos com grande número de entidades interativas também encontram problemas de processamento monolítico (Roehl, 1995; Brutzman, 1995). Para solucioná-los, ambientes virtuais utilizam tecnologias como o padrão IEEE **DIS** (Macedônia, 1995) e a arquitetura **HLA** (Fullford, 1996), que possibilitam a distribuição de processamento das entidades num mundo virtual e a transmissão de características visuais e comportamentais entre diferentes sistemas.

Características visuais informativas auxiliares na interface são fundamentais para que o usuário não se desoriente no mundo, mas deve-se saber quando e como utilizar cada uma em benefício do próprio usuário (Morar, 2002). O sentido de imersão não deve se restringir meramente ao sentido da visão, mas deve abranger os cinco sentidos. A influência de cada um desses sentidos na percepção do mundo virtual e da própria movimentação do usuário tem sido estudada na universidade de York (Harris, 2002). Quanto maior for o número de sentidos do usuário que a interface de um ambiente virtual conseguir envolver convincentemente, mais imersiva ela será.

1.4. Tecnologia Utilizada em Simuladores de Trânsito

As ferramentas de simulação de trânsito atualmente desenvolvidas utilizam macro-simulações, micro-simulações ou ambas. Modelos baseados em autômatos celulares (Blue, 2003) associados ou não com **DEVS** (Lo Tártaro, 2001; Díaz, 2001) também estão em desenvolvimento, mas com menos notoriedade. Dentre os dois grandes grupos citados acima, pode-se dizer que a micro-simulação tem tido ênfase crescente. Entretanto, mesmo com máquinas mais potentes, ainda não é possível construir micro-simulações de regiões muito extensas ou com uma quantidade grande de veículos sem que haja redução das taxas de atualização do modelo.

Arquiteturas para ferramentas executadas em uma única máquina têm sido implementadas, utilizando códigos com algoritmos otimizados para melhorar o desempenho, supercomputadores para auxiliar o processamento rápido das informações (Manouselis, 2001; Jayakrishnan, 1990) e linguagens com alta extensibilidade, como **SLX** (Schulze, 2001; Lemessi, 2001), para garantir a rápida extensão e adaptação da aplicação.

Sistemas multi-agentes (**SMA**), utilizando a linguagem de formalização Gaia (Manouselis, 2001), por exemplo, também têm sido propostos, onde cada agente representa um veículo ou semáforo. Uma arquitetura flexível baseada em agentes, onde cada agente permite o uso de um algoritmo diferente de definição de rotas de veículos, também foi proposta por Thangiah *et al.*

(Thangiah, 2001). Além disso, testes de SMA, utilizando as linguagens **SACI** e **KQML** (Schmitz, 2002) no controle de tráfego urbano com agentes simulando semáforos e veículos, também obtiveram resultados positivos (Schmitz, 2002; Schmitz, 2002-2). Ainda, um SMA desenvolvido por Paruchuri *et al.* (Paruchuri, 2002) utilizando C++ e a biblioteca gráfica Qt (Qt, 2005) simulou um exemplo de tráfego desordenado causado por motoristas com diferentes estados de humor, cada um deles dirigindo e reagindo ao trânsito de maneira diferente.

A fim de lidar com o problema da carga computacional no processador, abordagens distribuídas, como o uso de sistemas de computação paralela, têm sido utilizadas (Hsin, 1992). Um exemplo é o PARAMICS (Cameron, 1994), que é capaz de simular centenas de milhares de veículos trafegando por milhares de vias. O padrão IEEE de interoperação HLA, desenvolvido pelo Departamento de Defesa dos Estados Unidos, é outra abordagem no desenvolvimento de aplicações desse tipo. Ela permite o controle distribuído de entidades, chamadas de *federates*, através de uma interface padronizada. Resultados positivos têm sido apresentados por projetos utilizando HLA, como é o caso de Klein (Klein, 1998), que construiu uma simulação de veículos, pedestres e semáforos.

Algoritmos eficientes são essenciais para uma simulação com alto desempenho. Um grande problema enfrentado atualmente é a definição de melhores rotas para veículos de entrega de bens de consumo passando por pontos geográficos específicos. Como o problema do caixeiro viajante, esse problema é NP-completo, não permitindo métodos para alcance de resultados exatos. Algoritmos de computação evolutiva têm sido as melhores opções na solução de problemas desse tipo (Tavares, 2003; Li, 2002; Sun, 2002). O problema é relevante pela importância comercial que possui (Schulze, 2001), mas foge ao escopo do presente trabalho e não será discutido em profundidade.

Além de eficiência, flexibilidade é outro ponto importante na simulação de trânsito, uma vez que a re-configuração de vias e de seus fluxos tem que ser feita para cada nova região avaliada. Soluções para geração de via de tráfego baseadas em imagens bidimensionais têm sido desenvolvidas para construção de modelos urbanos de cidades (Marson, 2003) e de suas ruas (Sun, 2002), mas, ainda assim, uma preparação prévia dessas imagens é necessária na maioria dos casos.

A interação com ambientes virtuais é uma área de estudo polêmica. Apesar dos resultados inovadores apresentados em interfaces de jogos eletrônicos, cada aplicação requer uma interface diferente, de acordo com suas funcionalidades (Reisman, 2003). Daí a dificuldade em se definir regras de desenvolvimento para interfaces tridimensionais. Além disso, é necessário um profundo

entendimento de como as pessoas interagem no cotidiano, para que se saiba qual a melhor forma de se interagir num mundo virtual (Sheridan, 2000). A garantia de maior imersão, através do uso de interfaces avançadas como **CAVE** ou **óculos estereoscópicos**, ainda está sob discussão e análise, apesar de algumas metodologias já terem sido desenvolvidas para comprovar o benefício do seu uso (Raja, 2004, Raja, 2004-2). Pesquisas nessas áreas estão sendo mais bem formalizadas antes que seus resultados possam ser tomados como referência (Roberson, 1997). Mais importante que a interface de hardware é como o mundo virtual é apresentado e quão fácil é para o usuário se localizar no ambiente tridimensional.

1.5. Contextualização do Projeto

O modelo proposto no presente projeto se enquadra no tipo misto, onde podem ser simuladas situações de tráfego intenso e lento em regiões urbanas e também situações de tráfego mais esparso e veloz, como em rodovias e auto-estradas. Seu tempo de execução é limitado apenas pela taxa de quadros (*frame rate*) com que o computador é capaz de processar a simulação.

A aplicação desenvolvida nesse projeto é organizada em três partes, cada uma com um papel fundamental na aplicação:

- **Sistema de configuração de dados**, onde a parametrização e iniciação das estruturas básicas da aplicação.
- **Simulador**, onde todo funcionamento do sistema é implementado, incluindo veículos, semáforos e usuário.
- **Interface**, que é responsável por desenhar a cena gráfica e apresentá-la a ao usuário.

A interação entre esses componentes ocorre da seguinte forma. A malha viária é criada a partir de um conjunto de pontos de arestas que representam cruzamentos e trechos de pistas. Cada aresta possui um fluxo de veículos por minuto e valores de iniciação de um semáforo. A compilação dessas informações é feita na configuração dos dados. Em seguida, o simulador entra em ação executando as atividades de cada entidade no ambiente virtual. A cada atualização a interface apresentada ao usuário é devidamente atualizada. Esse funcionamento será explicado em mais detalhe nos capítulos Capítulo 2, Capítulo 3 e Capítulo 4.

As rotas do modelo são geradas pelo método tradicional e de uma única vez, antes do veículo começar a percorrê-la, o que as torna inflexíveis a incidentes de tráfego como, por exemplo, um engarrafamento. A manutenção desse modelo tradicional ocorreu em prol da redução

de complexidade dessa versão inicial. Um veículo não pára até encontrar seu ponto de destino. A velocidade máxima de cada veículo é fixa durante toda a simulação e a mudança de faixas se dará de forma aleatória e apenas entre trechos da pista. Medições do consumo de combustível e emissão de poluentes não fazem parte do modelo.

Existem parâmetros iniciais configuráveis através de diferentes arquivos. Alguns dados numéricos são extraídos da simulação e apresentados na tela, como taxa de quadros/segundo e posição e orientação do **avatar** do usuário. A validação possível no protótipo se dá pela medição do tempo de viagem dos veículos e não há, nesta versão, simulação de pedestres, ciclistas ou transportes públicos.

A calibração do modelo é feita a partir da velocidade e aceleração dos veículos. A interface é tridimensional, utilizando OpenGL (OpenGL, 2005), e os níveis de detalhe (LOD) são aplicados a elementos do modelo, como os veículos e o relevo. Além disso, a estrutura da simulação foi baseada em modelos distribuídos atuais, tendo em vista sua futura distribuição e expansão.

Por fim, o funcionamento da movimentação dos veículos se dá através de algoritmos de vida artificial modificados, utilizados na movimentação de grupos de personagens em jogos, como os de *pathfollowing* e *separation* (Reynolds, 1999).

Este projeto é a continuação de outros trabalhos tanto no nível de iniciação científica como de trabalho final de graduação, desenvolvidos pelo aluno no período de Agosto de 2001 a Março de 2003. Durante esse período, foram realizados testes iniciais com movimentação de veículos e desenvolvidas versões iniciais de simuladores de trânsito num modelo experimental do trânsito na região do Complexo de Salgadinho, em Recife, Pernambuco, utilizando a ferramenta de simulação ***World-Up*** (World-Up, 2000; Barros, 2003). Apesar as idéias dos algoritmos terem sido parcialmente aproveitadas, esse estudo de caso não foi considerado no trabalho atual.

1.6. Estrutura do Documento

O trabalho está organizado em quatro capítulos. O Capítulo 2 explica como ocorre a configuração inicial do sistema. O Capítulo 3 descreve as metodologias de desenvolvimento utilizadas durante a criação do simulador, desde a arquitetura até o funcionamento dos veículos e semáforos. O Capítulo 4 aborda a interface do usuário e suas diferentes formas de manipulação. O Capítulo 5 detalha experimentos realizados para testar o desempenho da aplicação e apresenta os resultados dos testes. Por fim, o Capítulo 6 apresenta as conclusões obtidas após a análise dos

resultados e mostra uma perspectiva sobre o que ainda deve ser feito e o que pode ser otimizado na simulação.

Capítulo 2 :

Configuração de Dados

Nesse capítulo a primeira parte que compõe o simulador, envolvendo o sistema de configuração de dados, é abordada. A fim de simular o tráfego de uma região, a aplicação necessita de três tipos de dados de entrada: das vias de tráfego, do relevo e dos veículos. O primeiro deles configura as ruas e cruzamentos, o segundo carrega os objetos que representam os terrenos, e o terceiro define o número de veículos na simulação. Esses dados são carregados durante a construção da classe Simulação e estão descritos a seguir.

2.1. Vias de Tráfego

As vias de tráfego podem ser especificadas uma a uma através de um arquivo ou geradas automaticamente numa grade homogeneamente distribuída. A primeira forma é a que deve ser utilizada na construção de modelos específicos e realistas. A segunda é utilizada para testes de desempenho do simulador. Abaixo, o processamento das informações para cada um desses casos é descrito.

2.1.1. Construção Manual das Vias de Tráfego

Na construção manual das vias de tráfego, as informações são extraídas a partir de um arquivo texto, contendo os dados de nós e arestas de um grafo direcionado, cíclico ou não, que é gerado manualmente pelo usuário a partir do entendimento da malha de tráfego de uma região. A tradução de dados contidos em orto-fotocartas para arestas e pontos envolve diretamente a utilização de algoritmos complexos de processamento de imagens e, mesmo assim, não são gerais ou necessitam de um pré-processamento da imagem a ser utilizada para funcionarem corretamente. O desenvolvimento de tal funcionalidade de tradução está fora do escopo desta primeira versão do trabalho.

O arquivo organiza, em cada linha de texto, informações sobre um ponto ou sobre uma aresta separadas por espaços. Os formatos de especificação no arquivo para pontos e arestas são ilustrados nas Tabela 1 e Tabela 2, respectivamente, com valores arbitrários para exemplificação de formato. Valores arbitrários serão definidos em todas as tabelas de formatos de especificação de

arquivo subseqüentes para exemplificar os valores possíveis em cada campo. Os campos da Tabela 2 referentes à configuração de semáforos serão esclarecidos na seção 3.3.

Tabela 1: Formato de arquivo para especificação de ponto do grafo de tráfego.

C	*Id*	*X*	*Y*	*Z*
p	0	-50.2	0.2	20.3

Os campos da Tabela 1 são os seguintes: *C* – código informando que um ponto está sendo descrito nessa linha de arquivo, *Id* – número de identificação do ponto, X – posição do ponto no eixo coordenado *x*, *Y* – posição do ponto no eixo coordenado *y* e *Z* – posição do ponto no eixo coordenado *z*.

Tabela 2: Formato de arquivo para especificação de aresta do grafo de tráfego.

C	*Id*	*F*	*Oid*	*Did*	*NF*	*LF*	*Tv*	*Ta*	*Tl*	*Fti*
e	0	12	0	1	3	2.5	10.3	13.5	23.3	2.5

Os campos da Tabela 2 são os seguintes: *C* – código informando que uma aresta está sendo descrita nessa linha de arquivo, *Id* – número de identificação da aresta, *F* – fluxo de veículos por minuto, *Oid* – número de identificação do ponto de origem, *Did* – número de identificação do ponto de destino, *NF* – número de faixas, *LF* – largura das faixas em metros, *Tv* – tempo final do sinal verde em segundos, *Ta* – tempo final do sinal amarelo em segundos, *Tl* – tempo final do sinal vermelho em segundos e *Fti* – fase temporal inicial do semáforo em segundos.

Uma estrutura de dados, contendo o conteúdo exato do arquivo, é criada na classe Simulação utilizando classes específicas. Essa estrutura é pública, podendo ser acessada através de seus métodos por qualquer classe. Tal estrutura se fez necessária devido ao alto grau de dependência das conexões entre si, bem como entre as junções. Ela consiste num modelo reduzido do tráfego da região que deve estar presente em cada junção, de forma que elas não dependam de dados contidos em outras junções e, assim, percam a autonomia necessária à sua distribuição.

2.1.2. Construção Automática das Vias de Tráfego

A fim de realizar testes em diversos níveis de complexidade de tráfego, optou-se por criar um gerador automático de vias de tráfego simples, mas eficaz. Sua função é gerar grafos de tráfego em forma de grades retangulares.

Esse gerador cria uma grade com $N \times M$ pontos homogeneamente distribuídos no espaço, separados por uma distância d e conectados entre si por duas arestas, uma em cada sentido possível. Todas as arestas desse grafo possuem o mesmo número de faixas nf de largura l e um mesmo fluxo f de veículos por minuto. Além disso, a configuração de tempos para os semáforos é calculada com base na distância d entre seus pontos. Isso inclui o tempo para o sinal verde, amarelo, vermelho e também o tempo de espera inicial, como descrito na seção 3.3.

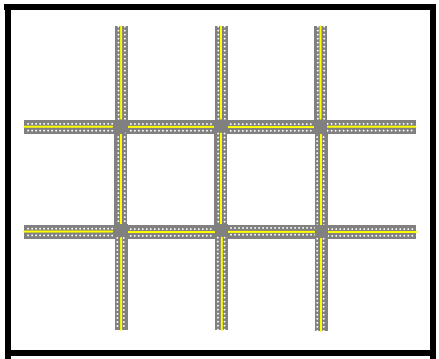


Figura 1: Exemplo de grafo de tráfego gerado automaticamente.

Os nós nas quinas da grade não são conectados a nenhuma aresta. Além disso, os demais nós localizados nas bordas não são conectados a outros nós também situados nas bordas.

O resultado é uma malha de tráfego semelhante à apresentada na Figura 1. Este exemplo possui 5×4 junções e conexões com 2 faixas cada. De cada via localizada na borda, partem f veículos por minuto de um lado da via enquanto, do outro lado, chegam outros veículos a seu ponto de destino.

Para requisitar a geração automática de um grafo de tráfego, uma linha extra de configuração é incluída no mesmo arquivo onde as arestas e os nós são definidos para a configuração manual já descrita. Essa linha especifica as variáveis utilizadas para geração automática do grafo e pode ser incluída em qualquer ponto do arquivo. Uma vez que ela esteja incluída, quaisquer outras configurações de pontos e arestas são ignoradas. O formato dessa linha pode ser visto na Tabela 3.

Tabela 3: Formato de linha de arquivo para especificação de geração automática do grafo de tráfego.

C	N	M	d	f	n_f	l
a	5	5	100	10	3	3

Os campos da Tabela 3 são os seguintes: C – código informando que o grafo deve ser gerado automaticamente, N – número de linhas contendo nós do grafo de tráfego, M – número de nós por linha do grafo de tráfego, d – distância entre nós adjacentes no grafo de tráfego, f – fluxo de veículos por minuto em cada conexão, nf – número de faixas em cada conexão e l – largura dessas faixas em metros.

2.1.3. Mapeamento de Estruturas de Tráfego

Após a carga das informações de vias de tráfego num grafo dentro da classe Simulação, essas informações são mapeadas nas estruturas de tráfego específicas dessa classe: as junções e as conexões. Como conseqüência do alto grau de dependência entre uma junção e as conexões nela contidas, a configuração inicial dessas entidades é feita em três estágios ao invés de apenas dois. Primeiramente, todas as conexões de uma junção têm as variáveis cujos valores são dependentes apenas dos valores do grafo de tráfego iniciadas. Elas são temporariamente armazenadas numa lista de conexões na classe Simulação. Em segundo lugar, as junções são criadas com suas listas de conexões. Por fim, o construtor da junção inicia, para cada uma de suas conexões de entrada, as variáveis que são dependentes da configuração da junção. Nesse momento, cada conexão termina de configurar suas variáveis internas e inicia seu semáforo, caso ele exista.

De acordo com a disposição da aresta que lhe é atribuída no grafo de tráfego, uma conexão pode ser classificada como:

- Conexão de entrada: representa um trecho de via que intercepta as bordas do mapa da malha de tráfego. Simula o tráfego proveniente de áreas vizinhas que chega na região da malha. Conexões desse tipo são os pontos de partida de qualquer veículo na simulação;

- Conexão de saída: também representa um trecho de via de tráfego que intercepta as bordas do mapa da malha de tráfego. Simula o tráfego que sai da região simulada e se dirige para áreas vizinhas. Conexões desse tipo definem o fim do caminho a ser percorrido por um veículo;

- Conexão intermediária: representa um trecho de via de tráfego que não intercepta as bordas da malha de tráfego. É, portanto, um trecho de pista completamente contido na região simulada e que compõe o caminho que leva um veículo de uma conexão de entrada a uma de saída.

A Figura 2 ilustra cada um dos três tipos de conexões.

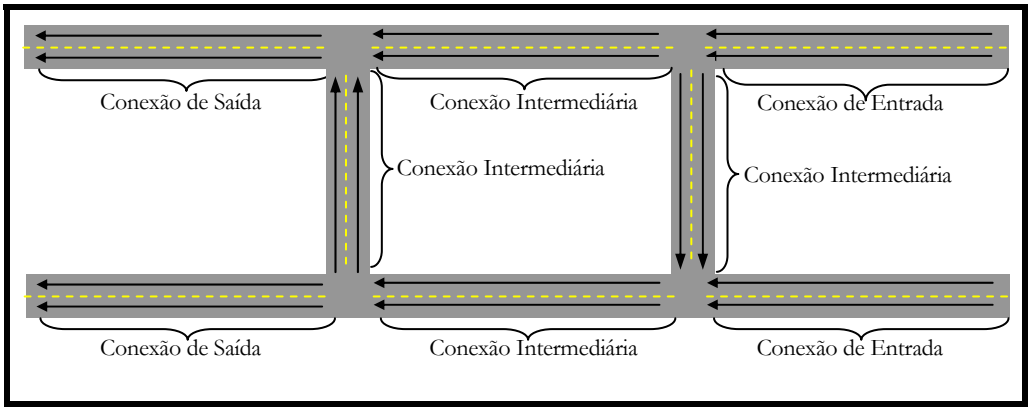


Figura 2: Tipos de conexões possíveis na malha de tráfego.

Cada conexão possui um certo número de faixas com uma mesma largura específica. Essas informações são obtidas diretamente das arestas do grafo de tráfego. No caso de um grafo gerado automaticamente, cada aresta possui a mesma configuração, exceto pelo tempo de espera inicial de seu semáforo.

Com base nas informações passadas para sua construção, uma conexão gera pontos de início e fim para cada uma de suas faixas. Os veículos perseguem estes pontos durante o percurso de seus caminhos. Os pontos iniciais das faixas de uma mesma conexão são alinhados numa reta perpendicular à direção da conexão. Entretanto, essa reta não passa pelo ponto inicial da conexão, pois sofre um afastamento de A. Sendo N o número máximo de faixas encontrado entre as conexões de entrada e saída da junção da qual a conexão atual parte e W a largura máxima encontrada dentre as faixas desse conjunto de conexões, tem-se que $A = W \times N$. Esse afastamento é necessário para que a distância entre o ponto final da faixa de uma conexão e o ponto inicial da faixa escolhida na próxima conexão seja suficiente para que os veículos possam fazer curvas suavemente. O cálculo e o resultado desse afastamento pode ser visto na Figura 3.

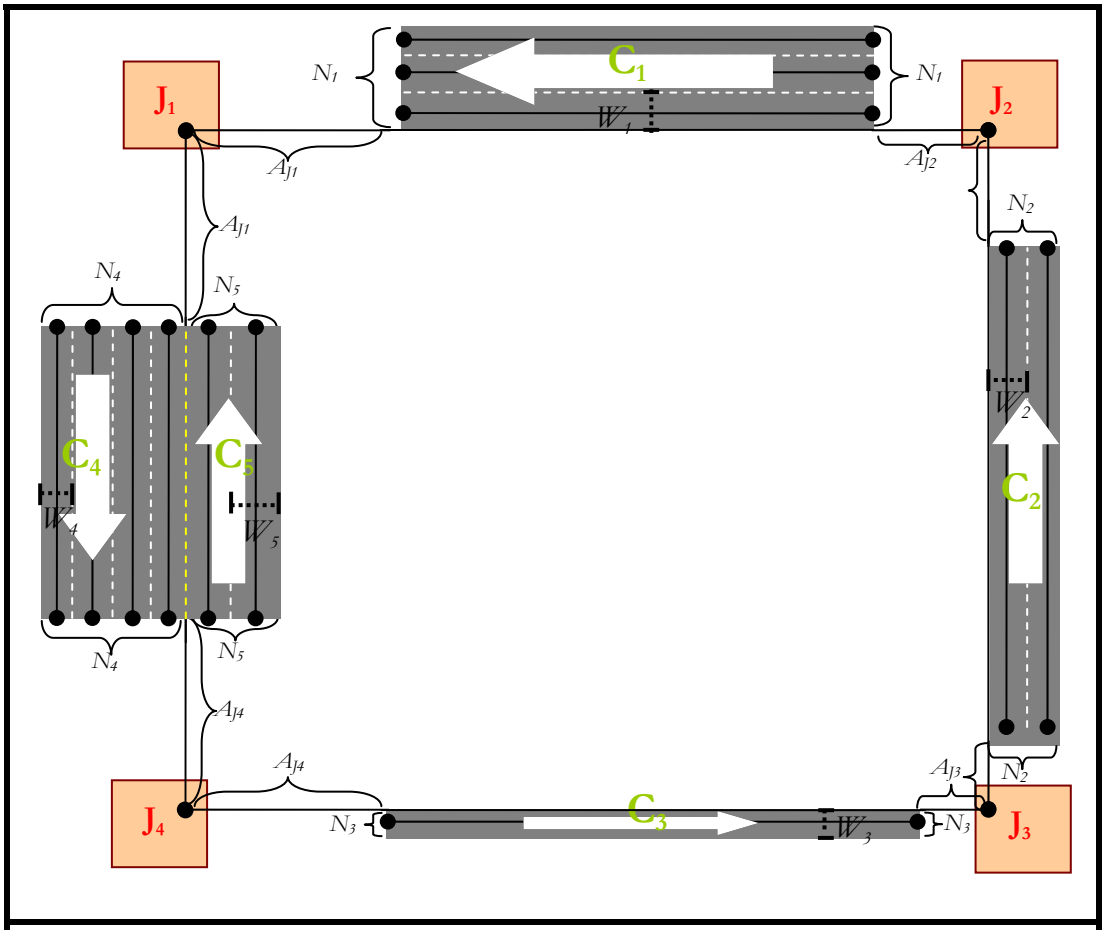


Figura 3: Cálculo e resultado do afastamento entre pontos iniciais e finais de faixas e de suas respectivas conexões.

Na Figura 3, as seguintes variáveis são apresentadas: C_i – conexão *i*, N_i – número de faixas da conexão *i*, W_i – largura de faixas da conexão *i*, J_i – junção i e A_{ji} – afastamento de pontos de faixa para J_i. O cálculo dos afastamentos para cada faixa nessa figura é mostrado abaixo:

- $W_5 \geq W_1 \geq W_4$ & $N_4 \geq N_1 \geq N_5$ → $\boldsymbol{A_{J1} = W_5 \times N_4}$
- $W_2 \geq W_1$ & $N_1 \geq N_2$ → $\boldsymbol{A_{J2} = W_2 \times N_1}$
- $W_2 \geq W_3$ & $N_2 \geq N_3$ → $\boldsymbol{A_{J3} = W_2 \times N_2}$
- $W_5 \geq W_4 \geq W_3$ & $N_4 \geq N_5 \geq N_3$ → $\boldsymbol{A_{J4} = W_5 \times N_4}$

Durante a construção da classe Simulação, além da configuração da malha viária através da leitura e tradução do grafo de tráfego, uma lista é criada contendo veículos inativos e disponíveis para inserção na aplicação pelas conexões de entrada. De acordo com seu fluxo de tráfego, cada conexão de entrada remove veículos dessa lista, gera-lhes caminho, ativa-os e os insere na

simulação. Em contrapartida, veículos que atingem o ponto final de uma conexão de saída são removidos, desativados e colocados de volta na lista de veículos inativos, ficando disponíveis para nova aparição na simulação quando ocorrerem outras requisições de veículo por conexões de entrada. A função da lista é limitar o número de veículos dentro da simulação. Além disso, ela agiliza o processamento durante a inserção e remoção de veículos da simulação, pois os veículos já se encontram parcialmente configurados.

2.2. Relevo

O relevo é configurado de duas maneiras diferentes, dependendo da forma como o grafo de tráfego é especificado. A especificação manual permite a definição de redes de tráfego quaisquer. Já a forma automática gera malhas de tráfego retangulares e foi criada para a geração rápida de malhas de tráfego com diferentes níveis de complexidade. Essas malhas foram utilizadas durante os experimentos apresentados no Capítulo 5. Abaixo, são descritas as formas de criação de um terreno para um grafo de tráfego especificado manualmente e para um outro gerado automaticamente.

2.2.1. Geração de Terreno para Grafo de Tráfego Definido Manualmente

Se o grafo de tráfego é definido manualmente num arquivo, através de arestas e nós, o terreno é também gerado através de um arquivo especial. Para cada trecho de terreno, esse arquivo especifica numa linha um número de identificação, uma posição, uma orientação e o número de níveis de detalhe. O formato dessa linha de configuração pode ser visto na Tabela 4.

Tabela 4: Formato de linha de arquivo para especificação de trecho de terreno.

C	*X*	*Y*	*Z*	*Ox*	*Oy*	*Oz*	*N*
t	10.0	0.0	20.0	0.0	90.0	0.0	3

Os campos da Tabela 4 são os seguintes: *C* – código informando que um terreno está sendo descrito, *X* – coordenada do terreno no eixo x, *Y* – coordenada do terreno no eixo y, *Z* – coordenada do terreno no eixo z, *Ox* – ângulo de rotação do terreno em graus ao redor do eixo x, *Oy* – ângulo de rotação do terreno em graus ao redor do eixo y, *Oz* – ângulo de rotação do terreno em graus ao redor do eixo z e *N* – número de níveis de detalhe do terreno.

Nas linhas seguintes do arquivo, são especificados os caminhos de diretórios para os avatares e para as texturas, além das distâncias de ativação dos níveis de detalhe do terreno que a

última linha de descrição de terreno definiu. Para cada nível de detalhe, uma nova linha é inserida e o formato dessa linha é ilustrado pelo Quadro 1.

Quadro 1: Formato de linha de arquivo para especificação de LOD do avatar de um trecho de terreno.

C	*Objeto 3D*	*Textura*	*D*
L	3dobjects\\terrain_default.3DS	textures\\terrain_defaultTexture_LOD0.bmp	100

Os campos do Quadro 1 são os seguintes: *C* – código informando que nível de detalhe do avatar do trecho de terreno está sendo descrito, *Objeto 3D* – caminho para objeto tridimensional representando o nível de detalhe do avatar do terreno, *Textura* – caminho para textura que será aplicada ao avatar do terreno nesse nível de detalhe e *D* – distância mínima entre o terreno e o usuário para ativação desse nível de detalhe.

É importante ressaltar o uso de duas barras invertidas para representar um arquivo de um subdiretório. Assim, a referência "3dobjects\\terrain_default.3DS" vista no Quadro 1, por exemplo, especifica um objeto tridimensional cujo caminho é "./3dobjects/terrain_default.3DS".

A especificação dos níveis de detalhe deve seguir uma ordem crescente de distâncias de ativação ou, visto de outra forma, uma ordem decrescente de detalhe do modelo tridimensional que representa o terreno. Com isso, o modelo do avatar com mais detalhes será descrito na linha situada logo após aquela que descreve o terreno ao qual pertence, seguido por um segundo modelo menos detalhado na próxima linha e assim por diante, até que se atinja o modelo com menor quantidade de detalhes.

2.2.2. Geração de Terreno para Grafo de Tráfego Definido Automaticamente

O terreno criado na geração automática de grafo de tráfego é representado por um avatar simples que consiste num quadrado de tamanho fixo. A textura aplicada, entretanto, é variável de acordo com o **LOD**. O número de LOD foi definido subjetivamente, analisando-se que quantidade mínima de LOD garantiria a percepção da mudança dos mesmos em diversas situações de visualização da cena. Percebeu-se que sete níveis de detalhe são mais que suficientes para garantir essa percepção na maioria das situações de visualização. Entretanto, um outro número poderia ter sido definido. As texturas aplicam cores distintas de acordo com a distância entre o usuário e o terreno. Elas servem para uma visualização do funcionamento do nível de detalhe dos

terrenos. As cores e as distâncias mínimas de ativação para cada nível de detalhe são apresentadas na Quadro 2.

Quadro 2: Níveis de detalhe com suas distâncias de ativação e cores das texturas que os representam.

Nível de detalhe	*0*	*1*	*2*	*3*	*4*	*5*	*6*
Distância	0	100	150	200	250	300	350
Cor da textura							

Os arquivos de modelos e texturas padrão para a geração automática de terreno podem ser modificados. O avatar, no formato *3ds*[1], pode ser substituído ou ter seu tamanho ajustado através de uma ferramenta de modelagem tridimensional (3D). Da mesma forma, as texturas podem ser substituídas ou modificadas através de um editor gráfico de imagens no formato *bitmap*. A visualização dos níveis de detalhe do terreno pode ser vista nas figuras Figura 24 e Figura 25 na seção 4.2 mais adiante.

2.2.3. Prédios e Urbanização

Uma vez que nenhuma interação entre veículos e terreno tenha sido definida ainda, outros objetos que não sejam terrenos podem ser incluídos na cena para melhor ilustrar uma região. Eles são tratados como se fossem um terreno e desenhados na cena com uma posição, uma orientação e um avatar com diversos níveis de detalhe.

Devido à pouca importância desses objetos na cena, não foi necessária a criação de uma nova classe para representá-los. Uma utilização mais aprimorada dos mesmos está fora do escopo deste trabalho.

2.3. Veículos

Além dos arquivos para especificação do grafo de tráfego e do terreno, um outro arquivo foi criado para configuração dos veículos na simulação. No momento, entretanto, este arquivo é utilizado somente para definir o número total máximo de veículos dentro da simulação. Isso é feito através da definição de um código de identificação, seguido de um espaço e do número de veículos desejado, como apresentado na Tabela 5.

[1] 3ds: formato de exportação da ferramenta 3D Studio Max©, produzida pela Discreet© (Discreet, 2005).

Tabela 5: Formato de linha de arquivo para especificação do número total máximo de veículos.

C	*N*
n	100

Os campos da Tabela 5 são os seguintes: *C* – código para definição desse número e *N* – valor para esse número.

Capítulo 3 :

Simulador

O simulador é o núcleo da aplicação. Ele é responsável por fazer com que as entidades do mundo virtual se movimentem e interajam entre si. Esse capítulo descreve sua arquitetura e seu funcionamento interno em termos do comportamento de suas entidades virtuais, que são os veículos e os semáforos.

3.1. Arquitetura do Simulador

O simulador ItranS foi desenvolvido com uma arquitetura cujo processamento é feito cruzamento a cruzamento, similar às sugeridas por Cameron e Klein (Cameron, 1994; Klein, 1998). Esta arquitetura permite estender o sistema para um modelo distribuído, uma vez que cada cruzamento tem o processamento dos seus veículos realizado de forma autônoma, permitindo a distribuição de cada um desses cruzamentos em máquinas diferentes. Teoricamente, essa distribuição de processamento também permite o aumento da abrangência geográfica do mundo virtual sem prejuízo do desempenho do simulador, estando limitado o aumento de dados a serem processados, apenas, ao número de máquinas disponíveis para essa distribuição.

O mapeamento da rede de tráfego de uma região é feito através de um grafo bi-direcional, como ilustrado na Figura 4.

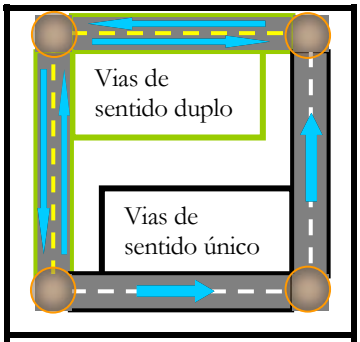


Figura 4: Representação da rede de tráfego usando um grafo bidirecional.

Os nós do grafo, que representam cruzamentos entre ruas e também pontos de entrada e de saída do tráfego da região, estão presentes em todo início e fim de cada trecho de rua contíguo, ou seja, num trecho de rua que não intercepta um cruzamento. Cada nó é posteriormente transformado em uma junção ou cruzamento de tráfego.

As arestas do grafo representam um conjunto de faixas de sentido único num trecho de rua contíguo. Um trecho de rua de sentido único é representado por uma única aresta. Já um trecho de rua de mão-dupla é representado por uma aresta para as faixas de um sentido e por outra para as faixas do sentido oposto.

Trechos de rua contíguos que não sejam retilíneos podem ser representados por um conjunto de arestas, formando uma polilinha que simula sua curvatura. Cada aresta será posteriormente transformada em uma conexão de tráfego, cuja descrição será feita mais adiante nesta seção, no mapeamento do grafo de tráfego nas classes da aplicação. Dessa forma, qualquer tipo de rua pode ser representado dentro do grafo por uma aresta e dois nós ou por um conjunto de arestas e de nós.

A arquitetura proposta divide o modelo de simulação em cruzamentos de vias de tráfego de acordo com os nós do grafo. Cada cruzamento controla o tráfego apenas contido em suas arestas de entrada de tráfego. Se um cruzamento contiver uma única aresta de entrada e de saída, ele representa o segmento de uma polilinha que define um trecho de rua curvo, o ponto de encontro entre o começo de uma rua e o fim de outra ou um trecho de rua de sentido duplo localizado no fim da malha de tráfego mapeada. Essas representações são ilustradas na Figura 5.

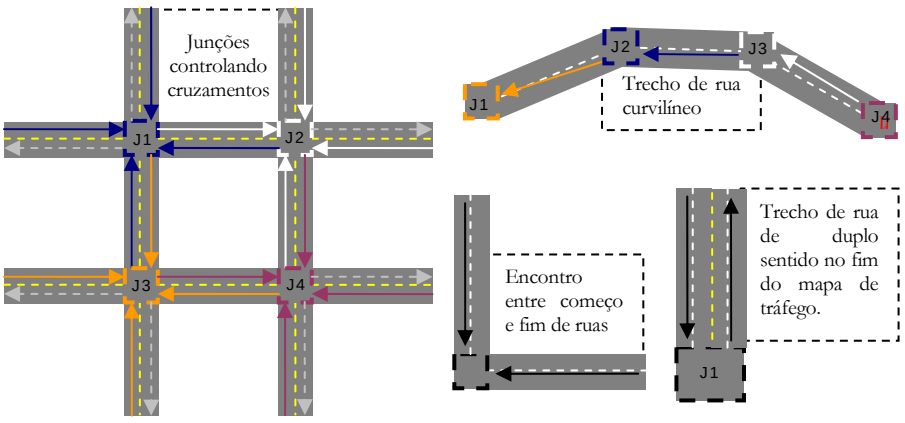


Figura 5: Mapeamento de arestas envolvendo cruzamentos em diversas situações na malha de tráfego.

Um cruzamento tem arestas de entrada, pelas quais o tráfego chega, e arestas de saída por onde o tráfego flui ou sai dele. O tráfego nas arestas de entrada é controlado pelo cruzamento que recebe o seu tráfego. As arestas de saída de um cruzamento são controladas por cruzamentos adjacentes que as considerarem arestas de entrada e receberem seu fluxo de tráfego. Assim, dentre as arestas às quais está conectado, cada cruzamento controla apenas as arestas de entrada.

Organizando-se o tráfego dessa forma, toda a malha tem seu controle distribuído de forma razoavelmente uniforme entre os cruzamentos. Como conseqüência dessa organização, espera-se que a distribuição do processamento de tráfego se torne possível, bem como a expansão do modelo sem mudanças em sua estrutura.

A construção da cena tridimensional é feita obtendo-se as informações gráficas dos objetos que estão sendo controlados pelos cruzamentos. Antes de enviar essas informações ao usuário, cada junção faz o ajuste do nível de detalhe poligonal de cada objeto de acordo com a distância de cada um destes objetos ao usuário, com o objetivo de reduzir ao máximo o processamento gráfico da cena visualizada.

As classes que compõem o sistema foram criadas tendo em vista esse modelo nodal de distribuição de processamento. A Figura 6 representa a arquitetura do simulador com a relação das principais entidades, representadas pelas seguintes classes:

1. **Usuário**: o avatar e o ponto de vista através do qual o usuário navega e observa a cena. Sua interface é descrita no Capítulo 4;
2. **Simulação**: contém todos os cruzamentos e o terreno. Controla o funcionamento do ambiente virtual como um todo;
3. **Junção** ou **Cruzamento**: a entidade que representa a maior parte das unidades de processamento autônomas;
4. **Relevo**: mapa topográfico tridimensional da região. Representa o resto das unidades de processamento autônomas;
5. **Conexão** de **entrada**: trechos de pistas com fluxo de veículos pré-definidos controlados por uma junção. Possuem uma ou mais faixas, além de controlarem os veículos que estão fluindo num determinado momento;
6. **Veículo**: cada entidade móvel que se desloca ao longo das faixas de cada trecho de pista.

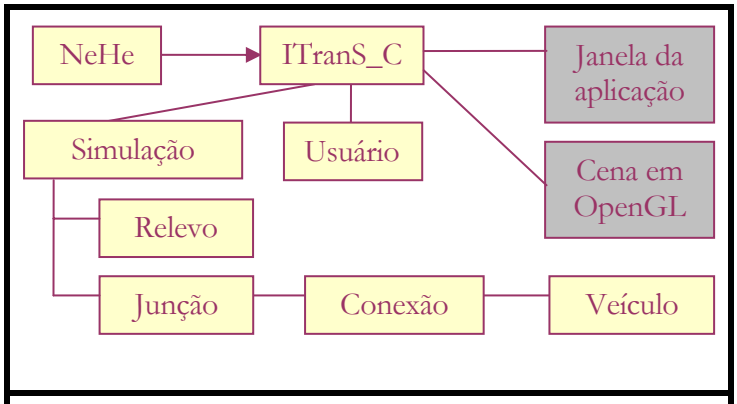


Figura 6: Arquitetura do simulador.

Da mesma forma que os nós do grafo são mapeados em junções, as arestas são mapeadas em conexões. Assim, todas as estruturas representadas no grafo de tráfego podem ser guardadas em classes na simulação.

Devido ao processamento centralizado nessa versão, criou-se uma entidade extra. A classe ITranS_C contém todo o ambiente de simulação, iniciando e controlando a cena em OpenGL, o terreno, o avatar e a interação do usuário, além de todas as junções, cada uma com uma lista de conexões de entrada contendo seus respectivos veículos e semáforos. Essa classe pode ser considerada a base da aplicação. A biblioteca gráfica de NeHe (Molofee, 2004) foi utilizada para simplificar a configuração de uma janela no sistema operacional para conter o ambiente em OpenGL, podendo ser entendida como um conjunto de métodos de apoio à aplicação.

Quatro métodos principais compõem a classe ITranS_C, baseados na interface fornecida pela biblioteca gráfica: *initialize*, *update*, *draw* e *deinitialize*. O primeiro deles, *initialize*, é utilizado para iniciar a aplicação. Nele, a janela, a cena gráfica, o usuário e a simulação são iniciados e configurados seqüencialmente. O funcionamento interno desse método pode ser visto na Figura 7.

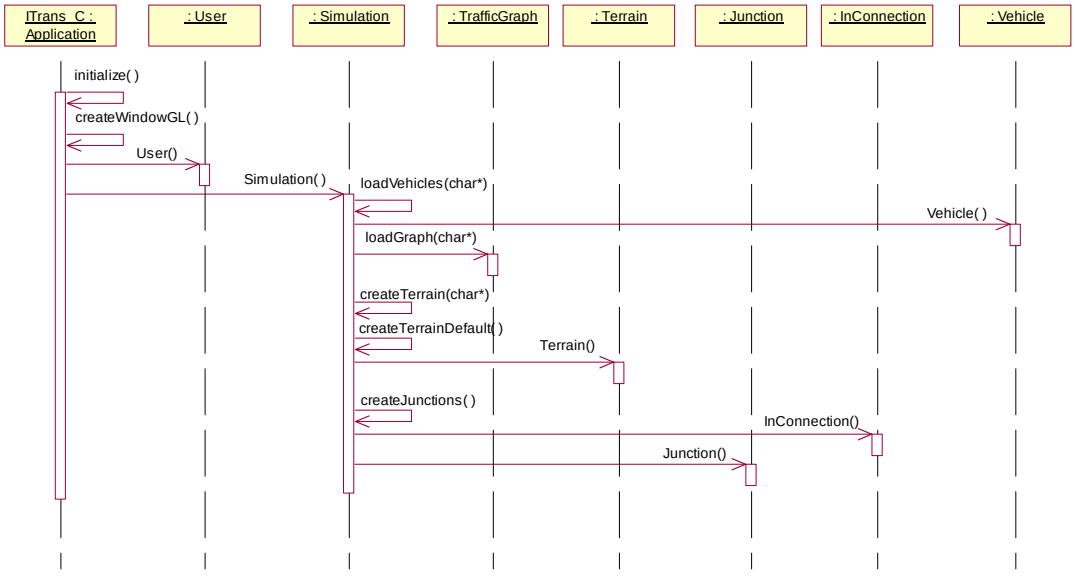


Figura 7: Diagrama de seqüência do método *initialize*.

O segundo é o método *update* que é responsável por processar os comandos de entrada e atualizar a cena. O seu funcionamento interno é descrito na Figura 8.

Esse método chama o método *simulate* na classe Simulação, que simula as junções e estas, por sua vez, simulam suas conexões de entrada. Essas conexões simulam os veículosque estão

ativos e se movem por suas faixas. Os veículos inativos são transferidos para a lista de veículos inativos da simulação, através do método *pushVehicle()*, se a conexão atual for uma conexão de fim de mapa. Se não, o veículo inativo é transferido para a próxima conexão que compõe o seu caminho, através do método *transferOutGoingVehicles()*. Numa versão distribuída da simulação, é nesse ponto em que a transmissão de informações de veículos entre conexões ocorrerá. Se a conexão for uma conexão de início, ela inserirá, perto de seu ponto inicial, novos veículos para percorrerem caminhos na simulação. Para tanto, essa conexão gera tempos de entrada num intervalo de um minuto e de acordo com o seu fluxo de tráfego, através do método *generateVechileEntranceTimes()*, explicado mais detalhadamente na seção 3.2.1. Quando o tempo de entrada de um novo veículo for atingido num determinado minuto, o veículo é retirado da lista de veículos inativos da simulação através do método *popVehicle()*, tem um caminho gerado para si através do método *generatePath()* e é ativado para ser simulado na próxima atualização através do método *activate()*.

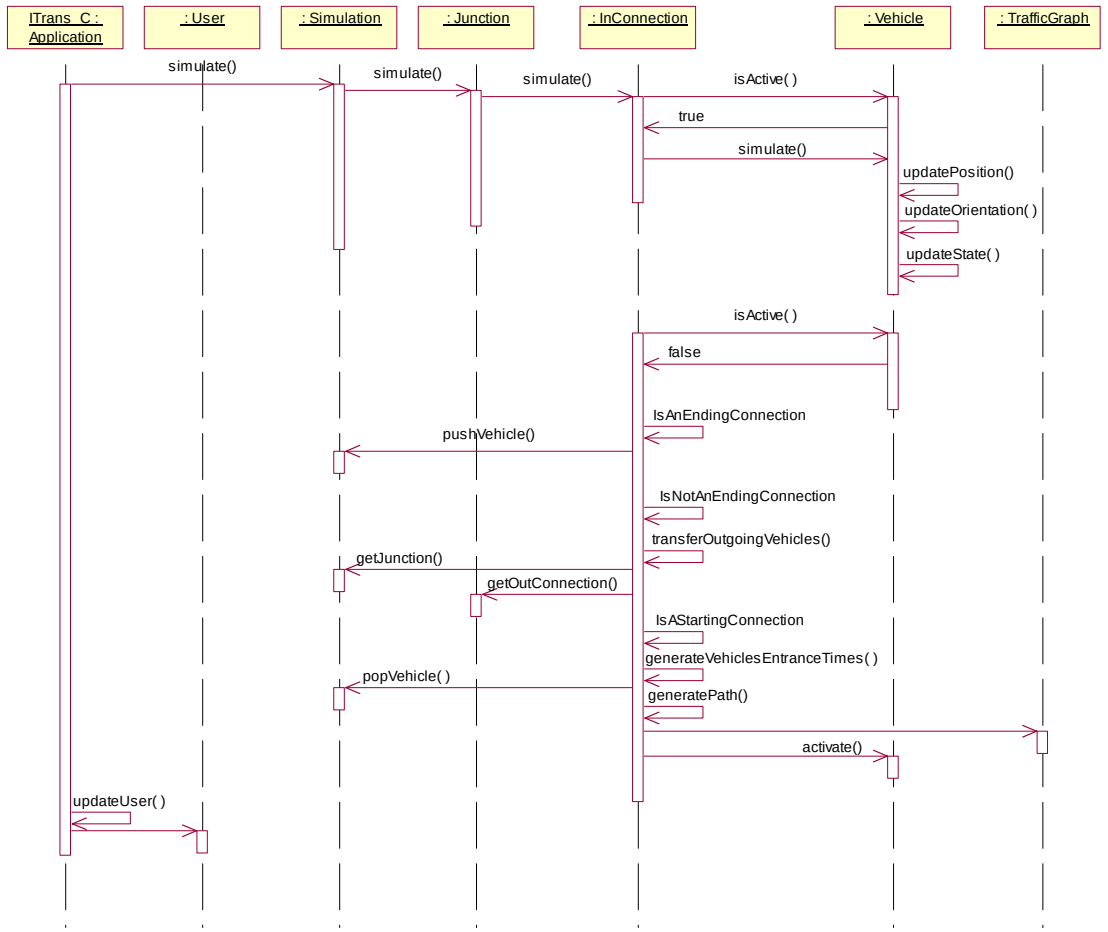


Figura 8: Diagrama de seqüência do método *update*.

O terceiro método, *draw*, constrói graficamente a cena já atualizada. O funcionamento interno desse método pode ser visto na Figura 9. Primeiramente, esse método configura o ponto de vista do usuário na cena de acordo com sua posição, através do método *updateUserView()*. Em seguida, o método *draw()* da classe Simulação é chamado, e requisita o desenho das junções e dos terrenos. Então, as junções desenham suas conexões de entrada. Essas conexões, por sua vez, desenham os veículos que contêm e controlam dentro de suas faixas e, após esta etapa, a interface do usuário é desenhada.

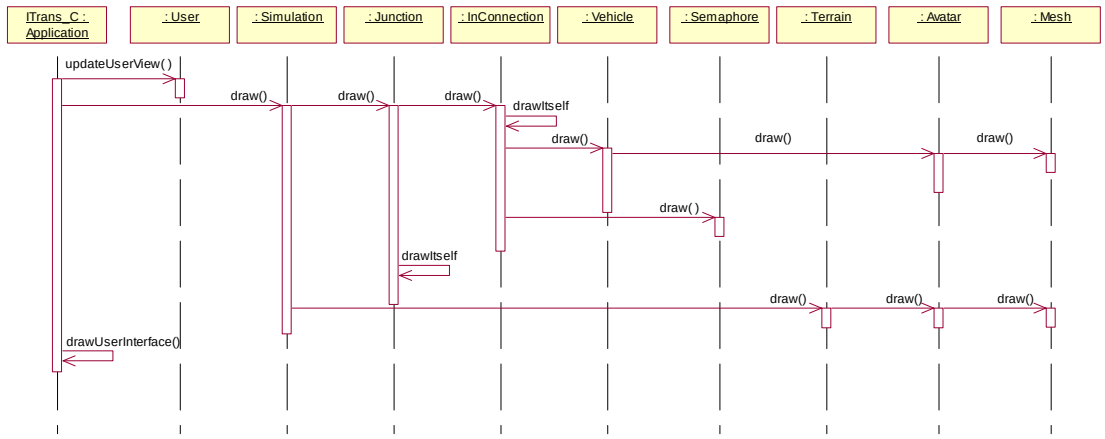


Figura 9: Diagrama de seqüência do método *draw*.

Por último, o método *deinitialize* garante que todas as estruturas sejam apagadas quando a aplicação terminar. O funcionamento interno desse método pode ser visto na Figura 10.

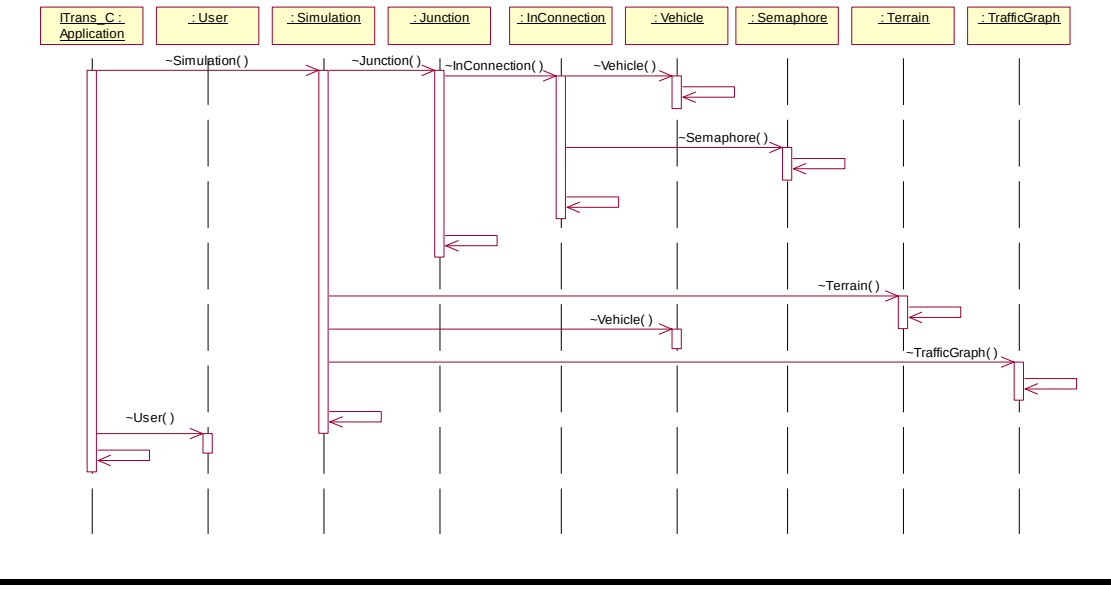


Figura 10: Diagrama de seqüência do método *deinitialize*.

Esse método chama o método destruidor da classe Simulação que, primeiramente, destrói as junções, que então destroem suas conexões de entradas. Essas conexões, por sua vez, destroem os veículos que contêm e controlam em suas faixas. Tendo destruído todas as junções, o método destruidor da classe Simulação destrói todos os terrenos, a lista de veículos e, por fim, o grafo de tráfego e as demais variáveis. Tendo o método destruidor da classe Simulação concluído sua execução, o método destruidor da classe Usuário é chamado pelo método *deinitialize* e a aplicação termina.

Pode-se entender a aplicação como tendo a estrutura procedimental ilustrada na Figura 11.

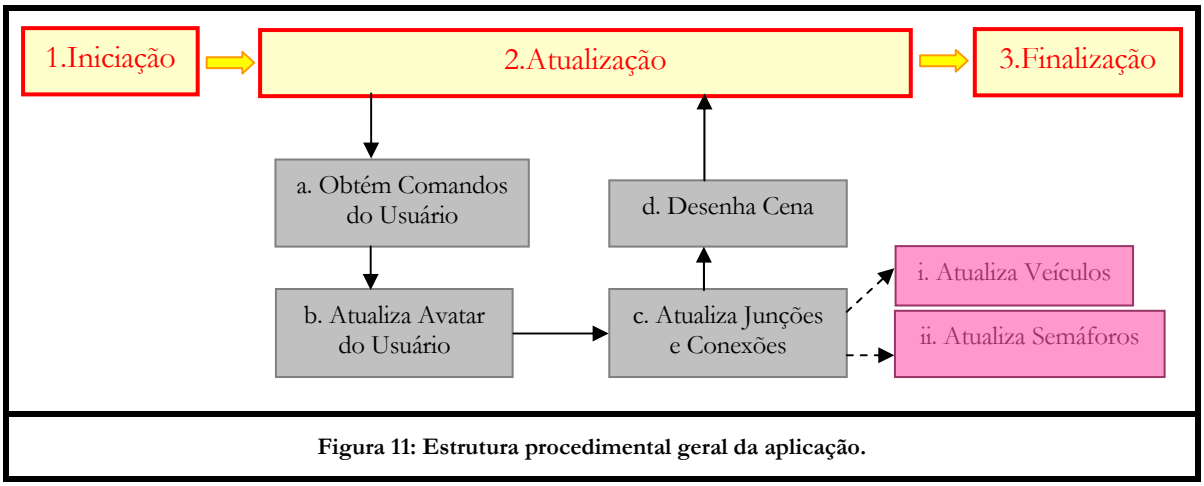


Figura 11: Estrutura procedimental geral da aplicação.

Como o terreno é representado por um avatar, a atualização dos níveis de detalhe ocorre quando a cena é construída graficamente. O mesmo ocorre com os avatares que representam os veículos. A descrição completa das classes pode ser vista no apêndice 0.

3.2. Caminho dos Veículos

O caminho dos veículos consiste em uma lista de nomes de conexões, dentro da rede de tráfego, que devem ser percorridas pelo veículo seqüencialmente. Esses nomes são, na verdade, números de identificação únicos distribuídos para as conexões durante sua configuração inicial.

A geração do caminho do veículo se dá através da escolha de conexões interligadas seqüencialmente por junções. A primeira conexão contida no caminho de um veículo é sempre a conexão de entrada pela qual esse veículo foi ativado. A partir daí, segue-se um ciclo de atividades até que se atinja uma conexão de saída, conforme a ordem descrita abaixo.

1. Primeiramente, escolhe-se um valor aleatório VA entre 0 e 1. Este valor representará a seleção do usuário.

2. Em seguida, calcula-se a probabilidade de entrada Pe_i de um veículo em cada uma das conexões de saída de uma junção. A probabilidade de escolha de uma conexão é calculada dividindo-se o seu fluxo pelo fluxo de saída total de uma junção. Um exemplo desse cálculo é mostrado abaixo.

   Sumponha que uma conexão C é conectada a duas outras conexões C1 e C2 através de uma junção J. Se C1 tem um fluxo de tráfego médio de A veículos/minuto, enquanto C2 tem um fluxo de tráfego médio de B veículos/minuto, então a probabilidade de um veículo, saindo de C, seguir para a conexão C1 é de A/(A+B), enquanto a probabilidade desse mesmo veículo seguir para a conexão C2 é de B/(A+B).

   O exemplo da Figura 13 apresenta uma junção com três conexões de saída. Sendo, C_i uma conexão *i*, $P(C_i)$ a probabilidade de uma conexão *i*, F_{Ci} o fluxo de uma conexão *i*, F_{total} o fluxo total de saída da junção, e I_{Ci} o intervalo de probabilidade de uma conexão *i*, o cálculo do fluxo total e das probabilidades de cada conexão ocorre da seguinte forma:

$$F_{total} = F_{C1} + F_{C2} + F_{C3}$$

e

$$P(C_1) = F_{C1}/F_{total} \qquad P(C_2) = F_{C2}/F_{total} \qquad P(C_3) = F_{C3}/F_{total}.$$

   Assim, definem-se as probabilidades para cada uma das três conexões de saída da junção.

3. Calculadas as probabilidades, esses valores são organizados num intervalo de 0 a 1, de forma a definir intervalos de valor que são atribuídos a cada uma das conexões, como ilustrado na Figura 12.

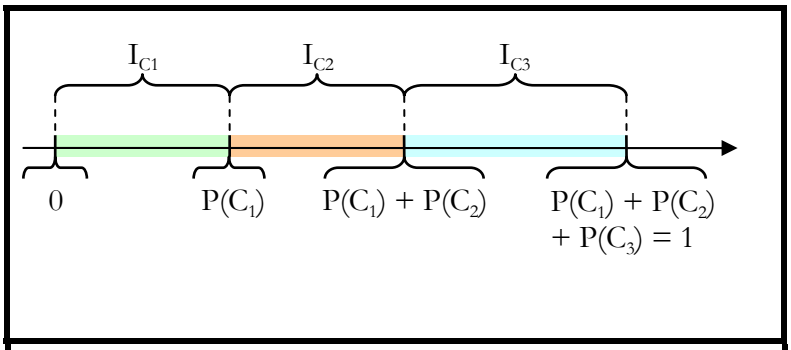


Figura 12: Cálculo de intervalos de probabilidade para conexões de saída numa mesma junção durante a geração de caminho de um veículo.

4. Por fim, a conexão escolhida será aquela cujo valor aleatório previamente sorteado se encaixe em seu intervalo. Caso ela já pertença ao caminho e haja outras opções de conexão, escolhe-se a próxima conexão que não pertença ao caminho. Se não houver outras opções ou se as opções existentes também pertencerem ao caminho, a conexão escolhida é inserida no caminho mesmo sendo duplicada.

   Uma vez escolhida a conexão, a faixa a ser seguida será, da mesma forma que na conexão de entrada, escolhida aleatoriamente. Porém as faixas são escolhidas durante o percurso do caminho – ou seja, em tempo real – e após esse caminho ter sido definido por completo. No caso de trechos de pistas bidirecionais, as faixas escolhidas estão, por razões óbvias, restritas ao sentido em questão. Além disso, para a realização da mudança de um trecho de pista para outro, não há transição para uma faixa mais próxima do próximo trecho de pista. O veículo muda para o próximo trecho de pista independentemente de em que faixa ele esteja.

5. Se a conexão for uma conexão de saída, depois de armazenada no caminho, esse caminho estará terminado e nenhuma outra atividade é necessária. Se não for, segue-se para a junção para a qual a conexão escolhida destina seu fluxo de tráfego e o ciclo acima descrito é reiniciado.

O ciclo básico desse processo pode ser visto na Figura 13. Uma explicação mais aprofundada do funcionamento desse e de outros algoritmos pode ser encontrada no apêndice A.3.

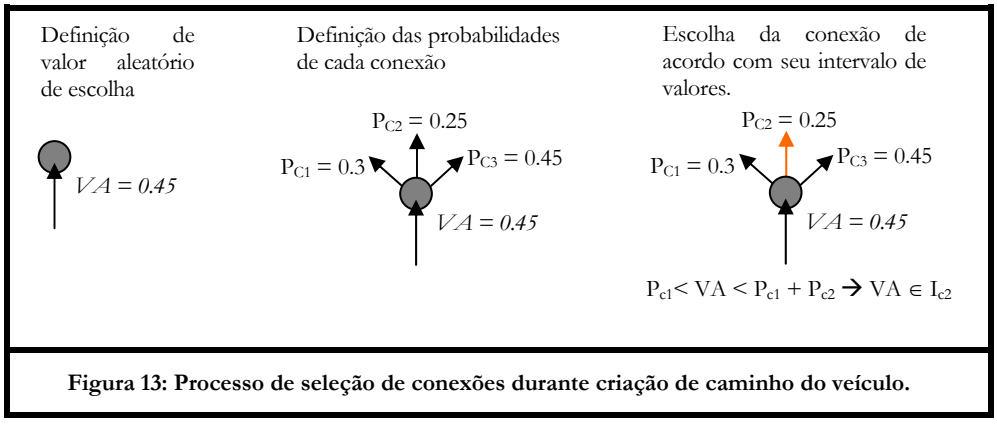


Figura 13: Processo de seleção de conexões durante criação de caminho do veículo.

3.2.1. Configurações Iniciais do Veículo com Nova Trajetória

As configurações iniciais dos veículos são feitas da seguinte forma: os veículos são inseridos na simulação somente mediante conexões classificadas como de entrada, situadas nas

bordas da malha de simulação. Ruas sem saída também servirão como pontos não só de entrada, mas também de saída para os veículos, pois essas também serão classificadas como conexões de entrada ou de saída.

Cada conexão de entrada possui um valor de fluxo de veículos por minuto F_{Ci}. A cada minuto, a conexão de entrada gerará uma lista interna com F_{Ci} valores aleatórios entre 0 e 60. Esses valores representarão os tempos de entrada de F_{Ci} novos veículos. A conexão conta o tempo passado em cada minuto de simulação. Quando um valor aleatório de tempo for igual ao valor apresentado no contador de minutos da conexão, esse valor é retirado da lista e um novo veículo é requisitado para inserção na simulação através da conexão de entrada.

Como mencionado anteriormente, inicialmente o veículo é criado sem possuir caminho algum, parcialmente configurado e colocado numa lista de veículos disponíveis dentro da classe Simulação para uso pelas conexões. Somente quando um novo veículo é requisitado por uma conexão de entrada é que um caminho é gerado para ele. Além disso, também nesse momento é que o veículo é ativado e re-inserido na simulação em uma faixa aleatória dentro da conexão que o ativou.

Como as faixas foram escolhidas aleatoriamente, o fluxo de tráfego numa conexão é distribuído homogeneamente entre as mesmas.

Vale ressaltar que as velocidades inicial e máxima, bem como a massa de cada veículo, são definidas aleatoriamente, baseadas no valor médio dessas características em veículos domésticos presentes atualmente no mercado. Esses valores podem ser vistos na Tabela 6.

Tabela 6: Variação de valores de velocidade e massa para os veículos da simulação.

Atributos	*Valores*
Velocidade inicial	0 m/s
Velocidade máxima	35m/s – 50m/s
Massa	800kg – 1300kg

3.2.2. Atualização do Próximo Ponto a que o Veículo Deve se Dirigir

Os pontos para onde o veículo se dirige seqüencialmente são os pontos de início e fim das faixas escolhidas aleatoriamente em cada conexão pertencente a seu caminho. Inicialmente, o veículo começa a 100 metros de distância do ponto inicial da faixa da primeira conexão contida em

seu caminho, localizando-se fora do mapa que define o ambiente virtual. Esse afastamento é necessário para estabilizar o movimento dos veículos antes que eles entrem nas vias. Nesse momento, o veículo ainda não aparece na simulação.

O primeiro ponto a ser seguido é exatamente o ponto inicial descrito acima. Ao atingir uma distância mínima d_m, definida quando a aplicação é iniciada, considera-se que o veículo chegou nesse ponto. Então o veículo passa a perseguir o ponto final dessa faixa e só então é representado graficamente.

Ao atingir a distância mínima também para esse ponto, o veículo é desativado e aguarda a transferência de seu controle da conexão atual para a próxima conexão que compõe o seu caminho, e que está contida em uma outra junção. O veículo sai da lista de controle da conexão atual para a lista de veículos da próxima conexão. Nesse momento, o veículo é reativado pela nova conexão, seleciona aleatoriamente uma faixa da mesma e, assim, o ciclo de perseguição aos pontos inicial e final da faixa se reinicia. Esse processo, ilustrado na Figura 14, prossegue até que se atinja um ponto final de uma faixa de uma conexão de saída.

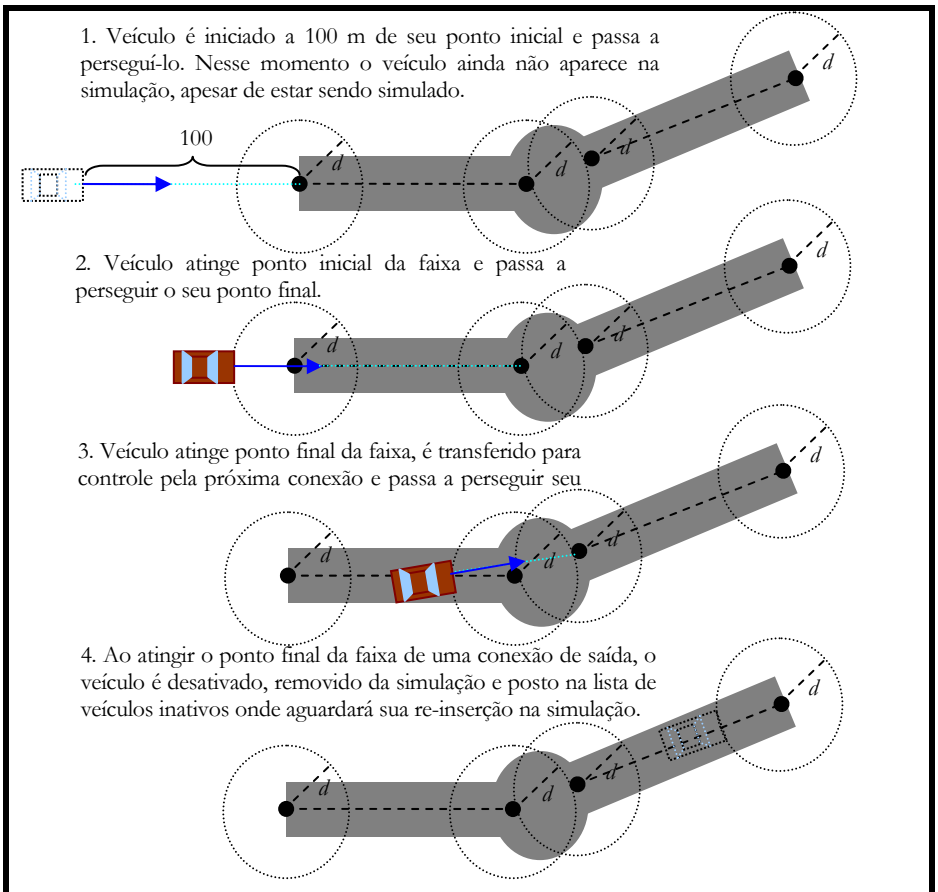


Figura 14: Atualização dos pontos de perseguição do veículo e de seu estado.

3.2.3. Verificação de Chegada ao Destino Final e Finalização do Veículo

Se o ponto atual for o ponto final de uma faixa da última conexão contida em seu caminho, o veículo tem suas configurações reiniciadas, é desativado e é colocado na lista de veículos inativos. Fica então disponível para chamada pelas conexões, esperando por sua vez para entrar novamente na simulação.

3.2.4. Possíveis Estados dos Veículos

Os veículos poderão estar ativos e se movimentando pelas pistas das conexões do modelo ou inativos e ocultos, aguardando serem reinseridos na simulação ou transferidos entre conexões de junções diferentes. A cena será composta por um número limitado de veículos, ativados e inseridos na simulação de acordo com os dados de fluxo de cada conexão. Os veículos inativos estão organizados numa fila de forma que uma ordem justa de ativação garante a participação de todos os veículos.

3.3. Simulação dos Semáforos

Os semáforos são entidades contidas em cada conexão e servem para controlar o seu fluxo. Cada semáforo controla apenas o fluxo da conexão na qual está contido e pode ser entendido como uma máquina de estados. Quatro são os estados percorridos pelo semáforo durante seu funcionamento. Eles são apresentados abaixo em sua ordem no ciclo de funcionamento. Entretanto, o primeiro deles só ocorre no primeiro ciclo.

- Estado de espera: o semáforo se encontra nesse estado antes de começar a funcionar com seu ciclo infinito de estados: luz verde → luz amarela → luz vermelha. Nele, o semáforo espera um intervalo de tempo T_e, cuja função é sincronizar semáforos pertencentes a conexões de entrada que despejam seu fluxo numa mesma junção ou a conexões de entrada que formam uma mesma avenida. Assim, é possível evitar colisões de fluxos perpendiculares num mesmo cruzamento e gerar o efeito "onda verde", quando blocos de veículos em uma rua ou avenida conseguem trafegar do seu começo ao seu fim ininterruptamente.

- Luz verde: estado que indica que o semáforo está aberto, permitindo que os veículos passem daquela conexão para a próxima.

- Luz amarela: estado intermediário que indica que o semáforo vai fechar. Tem o mesmo efeito do sinal verde, não afetando o fluxo de tráfego ou o comportamento dos veículos.

- Luz vermelha: os veículos que estão na conexão são obrigados a parar próximos ao ponto final da conexão e esperar que o sinal abra novamente.

Além desses quatro, um quinto estado também existe. Entretanto, esse estado não pertence ao ciclo básico de funcionamento do semáforo. Pelo contrário, o chamado "estado inativo" determina que o semáforo não está funcionando. Esse tipo de estado é utilizado quando se pretende simular uma falha interna ou a falta de energia na região onde o semáforo está localizado.

3.3.1. Configurações de Semáforos para Grafo Gerado Manualmente

A fim de que um semáforo seja iniciado corretamente, quando um grafo de tráfego é configurado manualmente, quatro valores de tempo são especificados para indicar a mudança entre seus estados e para definir o seu tempo de ciclo. Esses valores são obtidos a partir da leitura das arestas do grafo de tráfego, como previamente mencionado no Capítulo 2 e cujo formato pode ser visto na Tabela 2. Esses valores são do tipo ponto flutuante e a unidade de tempo que os define é a dos segundos. A função de cada um deles está descrita abaixo:

- *Tv*: tempo final do sinal verde. Esse valor indica o tempo em segundos, dentro do tempo de ciclo do semáforo em que o mesmo deve passar do estado de luz verde para o estado de luz amarela;

- *Ta*: tempo final do sinal amarelo. Esse valor indica o tempo em segundos, dentro do tempo de ciclo do semáforo em que o mesmo deve passar do estado de luz amarela para o estado de luz vermelha;

- *Tl*: tempo final do sinal vermelho. Esse valor indica o tempo total em segundos do ciclo do semáforo. Indica, conseqüentemente, o tempo em que o mesmo deve passar do estado de luz vermelha para o estado de luz verde e ter o seu contador de tempo de ciclo reiniciado;

- *Fti*: fase temporal inicial do semáforo. Esse valor indica por quantos segundos o semáforo deve permanecer estagnado no estado de luz vermelha antes de iniciar o contador de tempo de ciclo.

Com base nesses valores, o funcionamento do semáforo se dá seqüencialmente em quatro passos:

1. Um ciclo inicial de *Fti* segundos é iniciado. Durante esse período o semáforo se encontra em estado de luz vermelha. Como já mencionado, esse passo é utilizado para sincronizar

semáforos adjacentes, ocorre apenas uma vez, assim que o semáforo é iniciado, e pode ser entendido como um primeiro ciclo particular e inicial. Uma vez terminado, o ciclo normal de estados do semáforo começa;

2. O contador de tempo de ciclo do semáforo é ativado e iniciado, se necessário, e o estado do semáforo é posto para luz verde. Até que o contador atinja o tempo *Tv*, o semáforo permanece no estado de luz verde;

3. O estado do semáforo é posto para luz amarela. Até que o contador atinja o tempo *Ta*, o semáforo permanece no estado de luz amarela;

4. O estado do semáforo é posto para luz vermelha. Até que o contador atinja o tempo *Tl*, o semáforo permanece no estado de luz vermelha. Quando esse tempo é atingido, o contador de ciclo é reiniciado e volta-se ao passo 2.

Os veículos de uma conexão têm acesso direto ao estado do semáforo da conexão pela qual estão sendo controlados, variando seus comportamentos com base nessa informação.

Se um semáforo não tiver seus valores passados na linha do arquivo contendo sua aresta do grafo de tráfego, ele não será criado. Caso um dos quatro valores passados na criação do semáforo seja menor que zero, o semáforo é configurado com valores padrão de funcionamento, apresentados na Tabela 7.

Tabela 7: Tempos padrão para semáforos definidos pela aplicação.

Tv	*Ta*	*Tl*	*Fti*
26s	29s	60s	30s

Se os valores passados para *Tv*, *Ta* ou *Tl* forem iguais a zero, o semáforo será criado, mas não será ativado. Um semáforo nessa situação é colocado no estado inativo.

3.3.2. Configurações de Semáforos para Grafo Gerado Automaticamente

Se um grafo de tráfego é gerado automaticamente, o cálculo dos tempos para os semáforos é efetuado da seguinte forma: dado um tempo de espera inicial *t* padrão para os semáforos, um semáforo de uma conexão localizada na posição *i* de uma das linhas da matriz de junções criada possuirá um tempo de espera: $t_i = i \times t$. A mesma lógica será aplicada para os semáforos do sentido

inverso. Assim, os tempos de semáforos serão aproximadamente sincronizados para garantir o fluxo de tráfego contínuo, ocasionando as comumente chamadas "ondas verdes".

Para que esse efeito funcione para qualquer grafo de tráfego, *t* deve ter um valor igual ao tempo médio levado por um veículo para ir de uma junção à outra. Dado V_m como a velocidade média de um veículo na simulação e sendo D_j a distância entre duas junções adjacentes, tem-se:

$$t = D_j / V_m.$$

Os valores de tempo t_v, t_a e t_l para os estados de luz verde, amarela e vermelha respectivamente também são calculados com base em *t*, da seguinte forma:

$$t_v = t - 4; \qquad t_a = t - 1; \qquad t_l = 2 \times t;$$

Dessa forma, garante-se um tempo *t-1* segundos sem interrupção de fluxo e um tempo de *t+1* segundos com interrupção de fluxo. Dos *t-1* segundos sem interrupção de fluxo, *3* segundos são dados ao estado de sinal amarelo e *t-4* segundos ao estado de sinal verde. Esses valores foram definidos subjetivamente, com base na experiência pessoal com tempos de sinais de trânsito. Ainda, dos *t+1* segundos com interrupção de fluxo, o primeiro segundo é chamado de estado de "vermelho total ou completo". Nesse segundo, o sinal que controla o fluxo transversal ainda não entrou em estado de luz verde, permanecendo também em estado de luz vermelha. A função desse segundo de parada completa de fluxo, durante o qual todos os sinais do cruzamento se encontram vermelhos, é garantir um tempo para os veículos de um determinado fluxo que se encontram no meio do cruzamento saírem antes que o semáforo que controla o fluxo transversal a eles entre no estado de luz verde e libere mais veículos no cruzamento. Na geração automática de terreno, os valores mínimos para t_v, t_a e t_l são 3, 6 e 14 respectivamente. Esses valores foram definidos para evitar que o sinal se desative quando as junções estiverem muito próximas entre si, ou seja, quando $t = D_j / V_m \leq 4$, o que implicaria em $t_v = 0$ e desativaria o semáforo.

3.3.3. Representação Gráfica

Os semáforos são representados graficamente por um retângulo localizado 4 metros acima da faixa de mais alta velocidade da conexão, aquela localizada mais à esquerda, ou melhor, mais perto do centro da rua. Texturas diferentes são aplicadas para os estados de luz verde, luz amarela, luz vermelha e inativo, como visto na Figura 15.

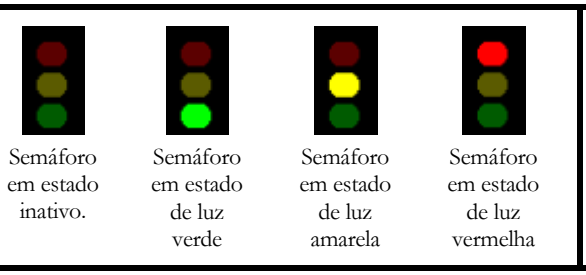


Figura 15: Texturas aplicadas ao semáforo em seus diferentes estados.

3.4. Comportamento dos Veículos

O veículo possui diversos comportamentos classificados em três tipos: os que alteram a aceleração do veículo, os que alteram sua velocidade e os que alteram sua posição.

Os comportamentos de separação e de *pathFollowing* aplicam uma força sobre a massa do veículo que gera uma aceleração e, por conseqüência, altera a sua velocidade. A soma das forças aplicadas por esses comportamentos ao veículo gera uma aceleração resultante que afeta a velocidade.

O comportamento de estacionamento, entretanto, afeta diretamente a velocidade, diminuindo seu valor gradativamente até que chegue a seu ponto de destino, desconsiderando as forças aplicadas ao veículo por outros comportamentos. Esse comportamento é utilizado quando o veículo deve parar num semáforo em estado de luz vermelha.

Por último, tem-se o comportamento de colisão, que altera diretamente a posição do veículo, transportando-o para sua posição anterior caso seus pontos de colisão interceptem a área de colisão de outro veículo.

Cada um desses comportamentos é apresentado em detalhes a seguir.

3.4.1. Atualização da Posição do Veículo Baseado no Ponto de Perseguição Atual

Sendo P a posição de um veículo Vc, V sua velocidade atual, V_{max} a sua velocidade máxima atingível, p_i o ponto inicial da sua faixa atual, p_f o ponto final da sua faixa atual, $D = p_f - p_i$ o vetor de direção da faixa e D_{norm} o vetor D normalizado, calcula-se a velocidade ideal V_i para o veículo como:

$$V_i = D_{norm} \times V_{max}.$$

Sendo d_p a distância máxima permitida entre V_i e V, L_v a largura do veículo e L_f a largura da faixa, tem-se que:

$$d_p = L_v/2 - L_f/2 = (L_f - L_v)/2.$$

Considerando a posição do veículo como estando situada no centro de seu avatar, se a distância d entre V_i e V for maior que d_p, isso indica que a velocidade V do veículo se encontra distante da borda da faixa a menos da metade de sua largura. Como conseqüência, no próximo segundo, no qual o veículo será deslocado de uma distância V, o veículo terá a extremidade de seu corpo, ou seja, sua parte mais distante que metade de sua largura, localizado fora da faixa. Para que isso não ocorra, calcula-se uma força de correção F cuja orientação é dada pelo vetor $V_i - V$. Essa força gera uma aceleração no veículo e faz com que ele retorne para dentro da faixa. O valor dessa aceleração será tão maior quanto menor for a massa do veículo e quanto maior for d. A Figura 16 apresenta um exemplo do funcionamento desse mecanismo.

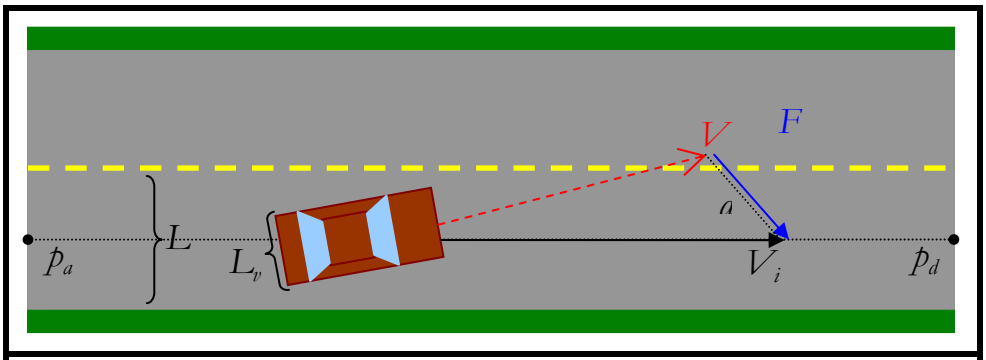


Figura 16: Ajuste da posição do móvel utilizando um algoritmo de pathfollow (Reinolds, 1999) modificado.

3.4.2. Estacionamento do Veículo no Sinal Vermelho

Um veículo não tem seu comportamento de estacionamento ativado pelo semáforo da conexão em que se encontra se o seu estado for o de luz verde, o de luz amarela ou o estado inativo. No estado de luz vermelha, entretanto, o veículo muda seu comportamento para estacionar o mais próximo possível do fim da sua faixa e, conseqüentemente, o mais próximo possível do semáforo. Para isso, as conexões e os veículos possuem estruturas de dados adicionais, seguindo modelo apresentado em (Cameron, 1994).

A conexão controla a ordem de entrada dos veículos em cada faixa, através de um sistema de bilhetes. Essa conexão armazena um valor de bilhete para cada faixa que indica o número do bilhete a ser entregue ao próximo veículo que entrar na faixa. Quando um veículo entra numa

determinada faixa, recebe o número de bilhete atual e a conexão incrementa de um o valor do bilhete a ser entregue ao próximo veículo. Observando-se o número de bilhete de cada veículo, tem-se o controle da ordem de entrada dos veículos em cada faixa.

Se um veículo Vc está numa faixa f de uma conexão C numa posição p e o semáforo S de C entra no estado vermelho, seu ponto de destino p_d, que pode ser o ponto inicial da faixa p_i ou o final p_f poderá ser substituído por um ponto de estacionamento p_e, de acordo com as condições abaixo.

Sendo D_{Pi} a distância entre a posição p de Vc e o ponto inicial da faixa p_i e sendo D_{Pe} a distância entre essa mesma posição p e o ponto de estacionamento p_e, tem-se que:

1. Se $p_d = p_f$, então $p_d = p_e$;
2. Se $p_d = p_i$ e $D_{Pi} \geq D_{Pe}$, então $p_d = p_e$;
3. Se $p_d = p_i$ e $D_{Pi} < D_{Pe}$, nada acontece.

Se o sinal está vermelho e se o veículo Vc atinge a distância mínima requerida para assumir que chegou ao seu ponto de destino p_d, e este é igual a p_e, o veículo entra num estado de espera e não se movimenta. Somente quando o semáforo sai do estado de luz vermelha é que o seu estado volta ao normal. Nessa volta, o seu ponto de destino p_d recebe o valor do ponto sendo perseguido anteriormente a p_e, tendo esse ponto sido p_i ou p_f, e Vc volta a percorrer seu caminho normalmente.

O ponto de estacionamento p_e é calculado com base no número de veículos numa faixa e no número do bilhete de cada veículo. Se existem n veículos na mesma faixa de um veículo Vc_1 e com números de bilhetes menores que o dele, seu ponto de estacionamento será:

$$p_e = p_f - n \times (C_v + 1) \times V_{norm},$$

onde V_{norm} é a normal indicando a direção oposta à da faixa, $V_{norm} = p_i - p_f$, e C_v é o comprimento do veículo. O valor de $1m$ adicionado a C_v representa a distância a que um veículo deve ficar do outro.

3.4.3. Cálculo da Colisão entre Veículos

Cada veículo tem quatro pontos de colisão, localizados nos vértices mais externos do avatar que o representa e formando um retângulo que delimita a sua área de colisão. São criados quatro pontos ao invés de apenas dois para evitar o seu cálculo durante a verificação de colisão.

O algoritmo de colisão é aplicado somente para veículos numa mesma faixa. Além disso, os números de bilhetes dados a cada veículo durante sua entrada numa faixa é utilizado para restringir ainda mais o número de veículos aos quais o algoritmo é aplicado. O mecanismo de verificação de colisão funciona da seguinte forma.

Dados dois veículos Vc_1 e Vc_2 localizados numa mesma faixa, com números de bilhete b_1 e b_2 respectivamente, seus respectivos pontos de colisão são o frontal direito Pfd_i, frontal esquerdo Pfe_i, traseiro direito Ptd_i e traseiro esquerdo Pte_i, para $1 \leq i \leq 2$. Dado, ainda, C_d como sendo o comprimento diagonal padrão de um veículo, tem-se que:

Se $b_1 < b_2$ e sendo $P_x \in \{Pfd_2, Pfe_2, Ptd_2, Pte_2\}$ de $Vc2$, tiver-se que:

- A distância entre P_x e Pfe_1 é menor que Cd e
- A distância entre P_x e Pfd_1 é menor que Cd e
- A distância entre P_x e Pte_1 é menor que Cd e
- A distância entre P_x e Ptd_1 é menor que Cd.

Então, para algum P_x, conclui-se que Vc_2 colidiu com Vc_1. A região de colisão definida por essas condições pode ser vista na Figura 17. Ela consiste na interseção das áreas dos quatro círculos de raio Cd, cada um centralizado em um dos pontos de colisão. Se um ponto de colisão de outro veículo entrar nessa região, admite-se que houve colisão.

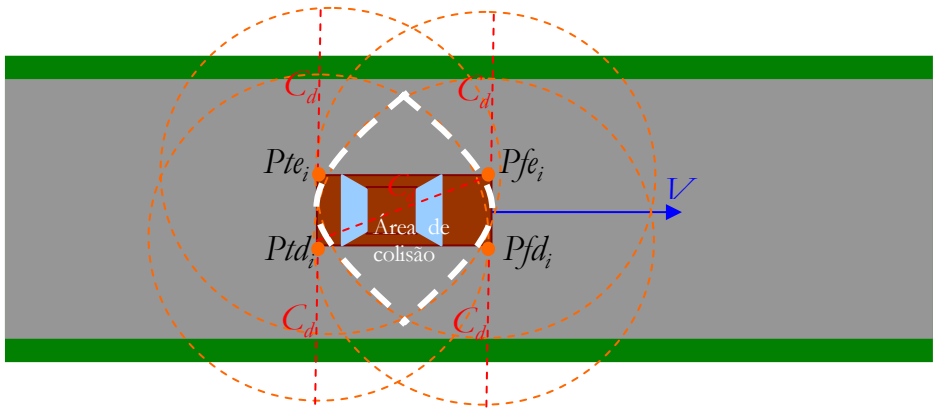


Figura 17: Definição da área de colisão de um veículo.

A posição anterior de um veículo é salva a cada atualização. Se, durante a atualização de sua posição, um veículo colide com outro, sua posição deixa de ser a atual e passa a ser a anterior armazenada na última atualização. Desse modo, o veículo que colidiu tem seu movimento interrompido, pois não se desloca para frente e, simultaneamente, é impedido de ocupar o mesmo lugar no espaço que o veículo no qual ele colidiu.

3.4.4. Cálculo da Força de Separação entre Veículos

Como forma de evitar a colisão, uma força de repulsão é aplicada a um veículo quando uma distância mínima d_m entre ele e um outro veículo é atingida. Sendo d a distância atual entre os veículos e sendo $D = d_m/d$, tem-se que, a cada atualização de posição de veículo, se $d < d_m$, uma força repulsiva F_s é adicionada à força total F de aceleração do veículo, onde $F_s \sim D$. A distância mínima d_m e a orientação de F variam entre dois valores, dependendo da situação em que os veículos se encontram:

- Caso os veículos estejam na mesma faixa, a distância mínima é:

$$d_m = CVc_1/2 + CVc_2/2 + 1,$$

sendo CVc_1 e CVc_2 os comprimentos de cada um dos veículos.

A orientação de F é inversa à da conexão, cujo vetor de orientação vai de seu ponto inicial ao seu ponto final. A Figura 18 apresenta um exemplo do funcionamento desse mecanismo.

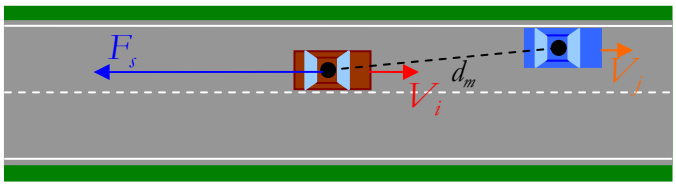


Figura 18: Ajuste da posição do móvel utilizando um algoritmo de separação (Reynolds, 1999) modificado para veículos numa mesma faixa.

- Caso os veículos estejam em faixas adjacentes, a distância mínima é calculada e o algoritmo de separação é efetuado somente se a condição abaixo for satisfeita:

Sendo $CDVc_1$ e $CDVc_2$ os comprimentos diagonais dos veículos, tem-se que a distância mínima d_C a partir da qual o algoritmo de separação deve ser aplicado é:

$$d_C = CDVc_1/2 + CDVc_2/2 + 1$$

A verificação de separação é efetivada e a distância mínima d_m é calculada e verificada somente se $d < d_C$. O valor de d_m é:

$$d_m = W_{vc1}/2 + W_{vc2}/2 + 1/8,$$

sendo W_{vc1} e W_{vc2} as larguras de cada um dois veículos.

A orientação de F é a mesma do vetor que vai do ponto final da faixa do veículo atual ao ponto final da faixa do veículo que entrou na área de repulsão. A Figura 19 apresenta um exemplo do funcionamento desse mecanismo.

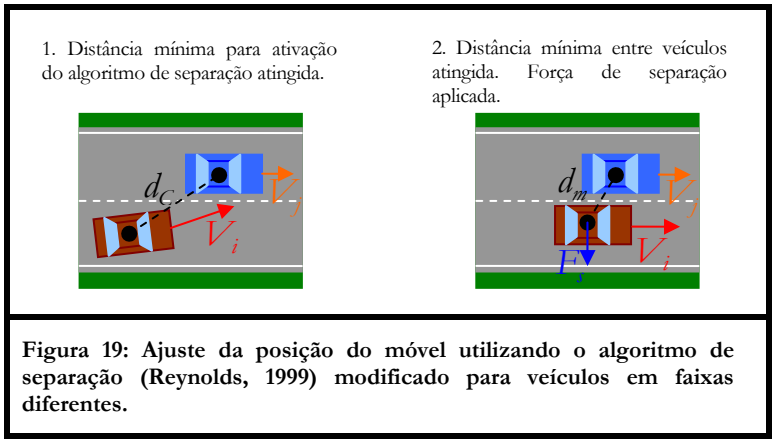


Figura 19: Ajuste da posição do móvel utilizando o algoritmo de separação (Reynolds, 1999) modificado para veículos em faixas diferentes.

3.5. Otimizações

A fim de melhorar o desempenho da simulação, otimizações foram realizadas. Algumas delas foram implementadas devido a problemas encontrados durante o desenvolvimento e realização de experimentos com a simulação, outras foram feitas para reduzir sua carga computacional. Houve ainda alterações para melhorar o processamento para a versão de arquitetura centralizada.

3.5.1. Alteração da Unidade Relativa de Tempo

A unidade relativa de tempo da simulação teve de ser alterada de acordo com as taxas de quadros por segundo da simulação, porque o cálculo das posições, velocidades e acelerações na

atualização da posição dos veículos foi feito com base no modelo de *Euler* (Mathworld, 2004). Nesse modelo, os valores de aceleração e velocidade são multiplicados por um intervalo de tempo Δt igual ao tempo decorrido entre uma atualização e outra da simulação. Considerando a taxa de quadros por segundo como QS, tem-se que $QS = 1/\Delta t$. A vantagem do modelo de *Euler* é que, mesmo que a taxa de quadros diminua, os veículos ainda assim se movem com a mesma velocidade, tornando o tempo na simulação síncrono ao tempo real, independentemente das taxas de atualização do simulador.

Entretanto, esse modelo apresenta alguns problemas para a simulação de trânsito. Quando as taxas de atualização do modelo ficam muito baixas, o Δt aumenta de valor e, com isso, os saltos entre posições consecutivas de um veículo se tornam maiores. O problema é que as curvas de movimento dos veículos se tornam mais abertas durante o algoritmo de *pathfollowing* (Reynolds, 1999). Como esse algoritmo requer que o veículo se aproxime de uma distância mínima de seu ponto alvo para que possa perseguir um outro ponto, quando Δt é muito grande isso simplesmente não se torna possível, principalmente quando o ângulo entre conexões adjacentes no caminho de um veículo é inferior a 90°.

A conseqüência disso é que os veículos andam em círculos ao redor do ponto de destino, sem nunca alcançá-los. Além disso, a dificuldade aparente que os veículos têm em seguir a pista é obviamente bem maior. Em resumo, o modelo de *Euler* não funciona bem para simulações com muitas entidades e baixas taxas de atualização.

Para sanar esse problema sem a implementação de outros métodos, como o de quarta ordem de Runge-Kutta (Mathworld, 2004), decidiu-se modificar os intervalos de tempo Δt passados para a simulação entre uma atualização e outra, dependendo de seu valor atual. Se $\Delta t \leq 0.04$, que representa uma $QS \geq 25$ quadros/segundo, o valor de tempo decorrido para a simulação é o próprio Δt e o modelo é estável. Entretanto, se $\Delta t > 0.04$, o valor de tempo decorrido para a simulação será sempre 0.04 ao invés do valor de Δt, de forma que, para um $QS < 25$, o modelo tenderá a diminuir a sua velocidade de simulação, tanto com relação ao deslocamento dos veículos como em relação ao tempo de mudança dos estados dos semáforos.

Fazendo-se isso, o tempo de simulação nem sempre será igual ao tempo real decorrido. Apesar dessa desvantagem, o modelo se torna estável e funciona independentemente da taxa de quadros por segundo com que a simulação está sendo executada.

3.5.2. Cena Gráfica

A fim de reduzir a complexidade da produção da cena gráfica da simulação, aboliu-se a iluminação. Nenhuma luz foi adicionada à cena de forma que a iluminação *default* de OpenGl foi a utilizada. Todos os objetos que aparecem na simulação têm texturas aplicadas em si para garantir sua coloração. A aparência final da cena gráfica pode ser vista na seção 4.2.

3.5.3. LOD

A fim de aumentar o desempenho da simulação, níveis de detalhe foram não só aplicados aos terrenos, como visto na seção 2.2, mas também aos veículos. A geração desses níveis de detalhe se deu da seguinte forma.

Primeiramente, um modelo razoavelmente detalhado de um veículo doméstico foi desenvolvido. Esse modelo, representando num veículo seus faróis, rodas, calotas, placas e vidros, ficou sendo a representação para o nível de maior detalhe do veículo. Outros modelos para representação de níveis de menor detalhe foram criados a partir da redução poligonal e de vértices desse primeiro modelo até que se atingiu um modelo com nível de detalhe mínimo. Este modelo mínimo representava o veículo por apenas três planos paralelos a cada um dos três eixos coordenados. Durante esse processo, conseguiu-se criar seis níveis de detalhes distintos. O processo envolveu um alto grau de subjetividade e criatividade para a redução correta dos detalhes sem que houvesse deformações perceptíveis entre modelos. Apesar dos seis níveis parecerem suficientes para representar o veículo em diversas situações de visualização, mais níveis de detalhe intermediários poderiam ter sido criados.

Também, da mesma forma que aos avatares dos terrenos, foram aplicadas aos avatares dos veículos texturas com colorações diferentes a cada um dos níveis de detalhe. Isso serviu para visualizar graficamente a mudança de níveis de detalhe do avatar do veículo durante os testes de seu funcionamento, pois apenas a mudança dos objetos tridimensionais que o representam era pouco perceptível, comprovando a eficácia da técnica de níveis de detalhe.

Os modelos utilizados para compor o avatar do veículo não foram rigorosamente otimizados no que diz respeito ao número de polígonos. Eles serviram apenas para testar o desempenho da aplicação em situações com números de veículos e níveis de detalhe variados. O número de faces e vértices, a distância de ativação e a textura para cada um deles podem ser vistos

no Quadro 3. A textura base foi obtida a partir da referência (Molofee, 2004-2). O funcionamento dos níveis de detalhe do veículo pode ser visto nas figuras Figura 24 e Figura 25.

Quadro 3: Informações sobre diferentes níveis de detalhe do avatar do veículo.

Nível de detalhe	LOD 0	LOD 1	LOD 2	LOD 3	LOD 4	LOD 5
Número de vértices	793	328	51	33	17	25
Número de faces	717	552	84	51	30	9
Distância de ativação	0	10	100	200	300	600
Textura						

3.5.4. Otimização de Estruturas de Dados

Pelo fato de se estar trabalhando, nessa versão, com um modelo centralizado, duas otimizações nas estruturas de dados foram realizadas com a finalidade de aumentar o desempenho do sistema. Essas otimizações não poderiam ser feitas para um sistema distribuído, mas foram consideradas aqui. As mesmas deverão ser reavaliadas no caso de a simulação ser distribuída em mais de uma máquina.

A primeira delas consistiu na centralização da lista de veículos a serem utilizados na simulação. Ao invés de cada junção contendo conexões de entrada possuir uma lista própria com um percentual do limite máximo de veículos da simulação para inserção na cena, foi criada uma lista única de veículos dentro da classe Simulação. Essa medida, além de diminuir a quantidade de variáveis em cada junção, auxiliou no controle mais eficiente do número total de veículos permitidos na simulação.

O mesmo ocorreu com o grafo de tráfego, onde cada junção deveria ter a sua própria cópia do mesmo. Entretanto, uma única versão foi deixada também na classe Simulação e compartilhada por todas as junções.

3.5.5. Aprimoramento de Algoritmos

Algoritmos que garantem o comportamento dos veículos, como os de *pathfollowing* e de colisão, foram otimizados para se tornarem menos onerosos.

O algoritmo de *pathfollowing* calcula se um veículo está saindo ou não de sua trajetória, baseado não mais na distância d_c entre sua próxima posição e o centro da faixa na qual está

localizado, mas na distância entre a velocidade desejada e a velocidade atual, cujos valores são mais rápidos de calcular que d_t.

O algoritmo de colisão foi significativamente simplificado, pois, para veículos numa mesma faixa, o veículo só calcula a colisão com os veículos que têm números de bilhetes de faixa inferiores ao seu. Além disso, o algoritmo de detecção de interseção entre pontos de colisão de veículo não utiliza o cálculo de distâncias entre pontos e retas. Assim, a definição perfeita de áreas de colisão dos veículos fica menos complexa e mais imprecisa, como pôde ser visto na seção 3.4.3.

Simplificações no algoritmo de separação também contribuíram para o desempenho. Entretanto, esse algoritmo foi simplificado não por necessidade de aprimoramento no desempenho, mas por ainda estar sendo aperfeiçoado no que diz respeito à sua estabilidade e à forma como o mesmo varia de acordo com a distância entre veículos.

Capítulo 4 :

Interface

Esse capítulo engloba a terceira e última grande parte que compõe a aplicação que é a interface. Ela foi criada utilizando-se a biblioteca gráfica de NeHe (Molofee, 2004) que permite a impressão de textos na cena em OpenGl, garantindo a apresentação de informações sobre a simulação. Essa interface permite a visualização tanto do ambiente tridimensional quanto de algumas informações da simulação.

Um eixo de orientação foi adicionado dentro da própria cena para guiar o usuário. A interface também foi configurada para acompanhar os movimentos do usuário, situando-se sempre à sua frente. Através de comandos do teclado, o usuário pode, além de navegar, salvar pontos de vista para posterior visualização.

4.1. Sistemas de Eixos

Antes que se entenda a movimentação do usuário, faz-se necessário o esclarecimento de em que direções e sentidos apontam os eixos coordenados do mundo. A Figura 20 esclarece concisamente essa questão. Esse é o sistema de eixos definido como padrão pela biblioteca gráfica utilizada (Molofee, 2004).

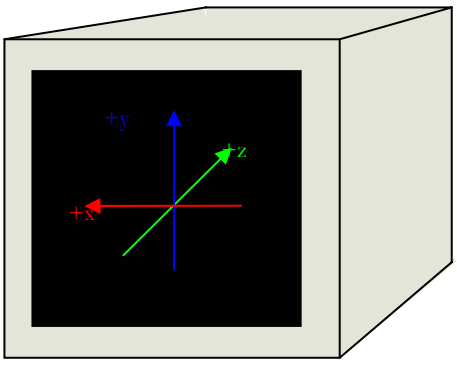


Figura 20: Direções e sentidos dos três eixos coordenados relativos à posição inicial do usuário e ao monitor.

4.2. Componentes da Interface

A interface do usuário é bastante simplificada. É composta por uma interface de apresentação tridimensional do mundo e de uma interface de entrada de comandos via teclado.

A interface introduz o usuário no mundo através de uma visão em primeira pessoa. Assim, o avatar do usuário não possui uma representação gráfica visível.

Informações textuais atualizadas em tempo real são apresentadas na região inferior da janela da simulação. Essas informações consistem em: posição e orientação do usuário, taxa de quadros por segundo em que a simulação está ocorrendo, número total de junções e número máximo permitido de veículos na simulação.

Um pequeno sistema de eixos tridimensional, semelhante ao apresentado na Figura 20, provê indicações dos sentidos positivo e negativo dos três eixos coordenados. Esses eixos servirão de orientação para o usuário, indicando-lhe o sentido correto dos eixos independentemente da direção para a qual o avatar de usuário esteja apontando. A interface está ilustrada na Figura 21.

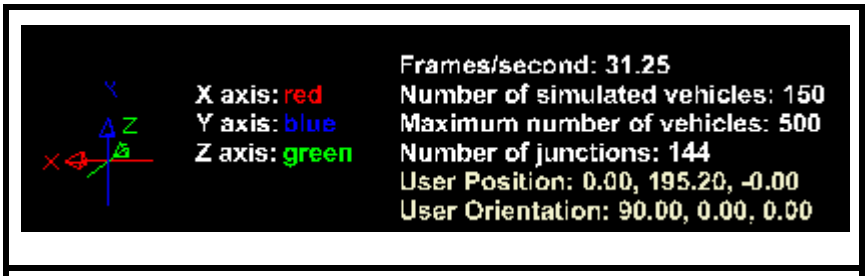


Figura 21: Interface fornecendo informações da simulação ao usuário.

A interface de entrada consiste num conjunto de comandos de navegação e de criação, seleção e remoção de pontos de vista. A lista de todos os comandos pode ser vista no Apêndice 0.

A interface do simulador pode ser vista em funcionamento nas figuras abaixo. A Figura 22 e a Figura 23 apresentam visões do usuário para um grafo gerado manualmente, onde é perceptível o funcionamento dos níveis de mais alto detalhe dos veículos, bem como o funcionamento dos semáforos. As figuras Figura 24 e Figura 25 apresentam visões do usuário para um grafo de tráfego gerado automaticamente. Nessas duas figuras é perceptível a variação dos níveis de detalhe no terreno e dos veículos, além do limite máximo e ótimo para o número de entidades simuladas.

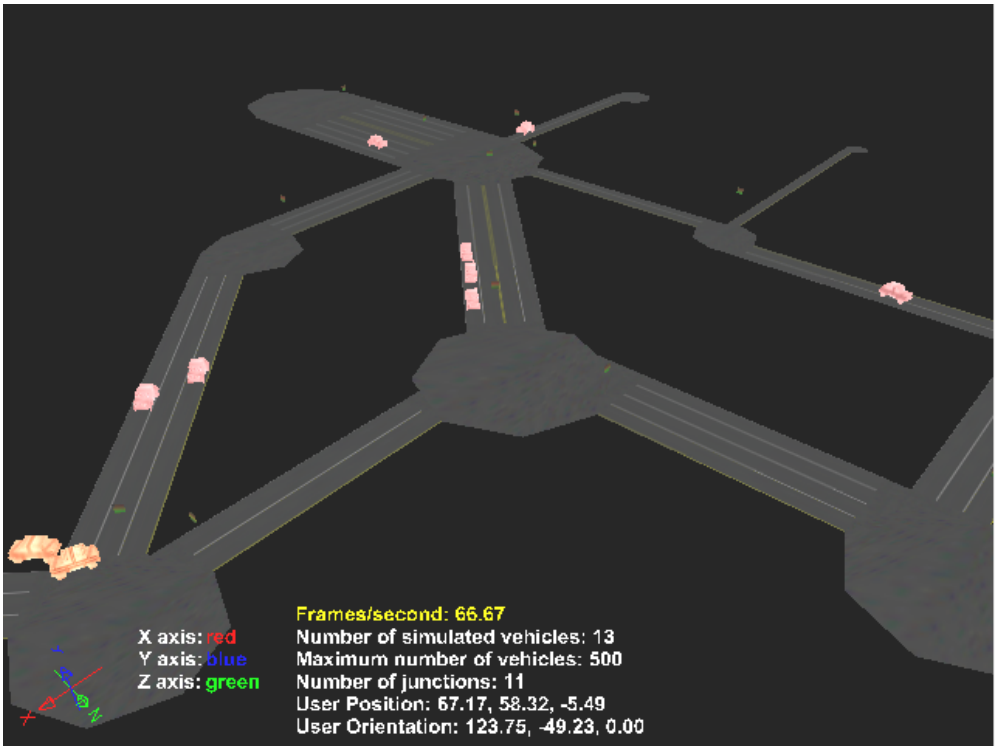


Figura 22: Visualização de vias de tráfego para um grafo gerado manualmente.

Figura 23: Funcionamento de semáforos e de níveis de alto detalhe dos veículos para um grafo gerado manualmente.

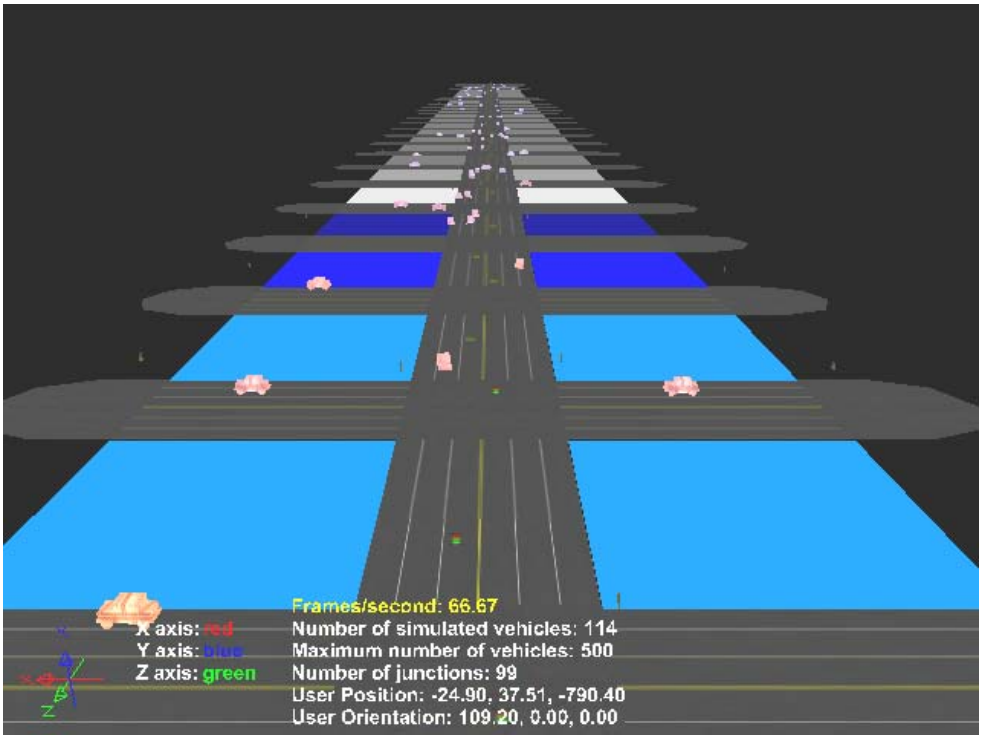


Figura 24: Visualização dos níveis de detalhe dos terrenos e dos veículos para um grafo gerado automaticamente.

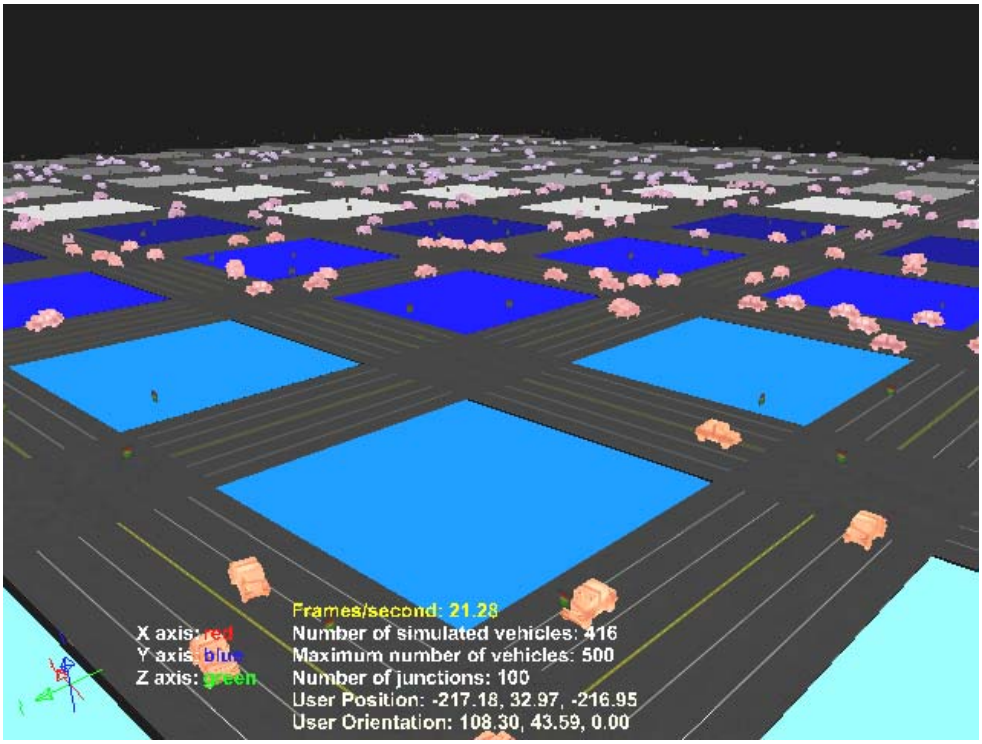


Figura 25: Visualização de níveis de detalhe dos terrenos e dos veículos para um grafo gerado automaticamente.

4.3. Pontos de Vista

O usuário tem o poder de armazenar pontos de vista para futura consulta durante a análise de tráfego. Esses pontos de vista consistem em orientações e posições específicas do avatar do usuário, e são armazenados em uma lista dentro do avatar do usuário. Toda vez que o usuário aciona o mecanismo de salvamento de seu ponto de vista atual, sua orientação e sua posição são armazenadas no final dessa lista.

Além da navegação seqüencial pelos pontos de vista armazenados, o usuário também pode remover ou voltar para o último ponto de vista visitado. Pode ainda remover ou ir diretamente para os doze primeiros pontos de vista armazenados na lista através das teclas de função. Essa lista e seu funcionamento estão ilustrados na Figura 26.

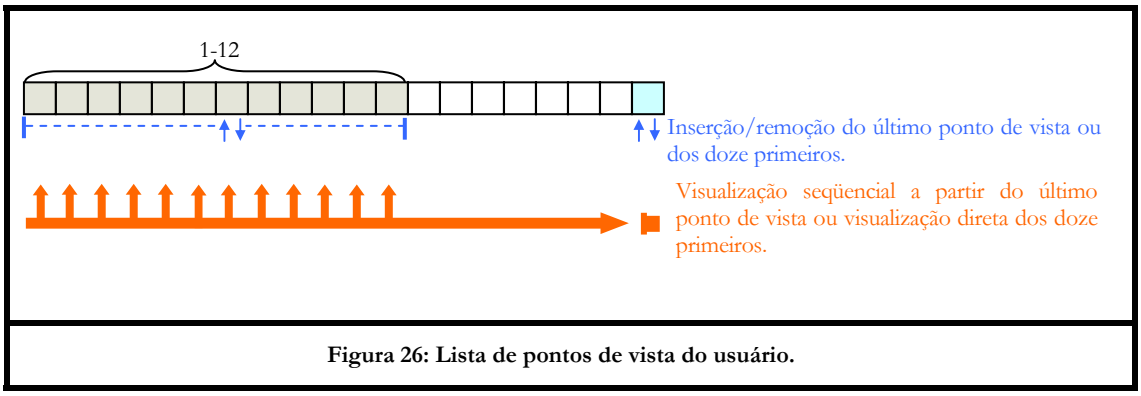


Figura 26: Lista de pontos de vista do usuário.

4.4. Comandos de Controle da Simulação

Além dos comandos para controle da lista de pontos de vista, existem também os comandos para controle do movimento do avatar na simulação. Esses movimentos se dividem em quatro grupos, apresentados abaixo:

1. Translação no sentido *s* em que seu avatar aponta, no sentido oposto ao de *s* e em qualquer direção no plano cujo vetor normal é *s*.

2. Translação numa reta paralela ao eixo y. Equivalente à modificação da altura do usuário relativa à superfície das pistas;

3. Translação no sentido *s* em que seu avatar aponta, mas mantendo-se num mesmo plano paralelo ao plano XZ. Consiste numa translação paralela à superfície das pistas.

4. Rotação em torno de seu eixo relativo x, movimento chamado de arfagem, ou em torno de seu eixo relativo z, movimento chamado de guinada.

A ilustração desses quatro tipos de movimento pode ser vista na Figura .

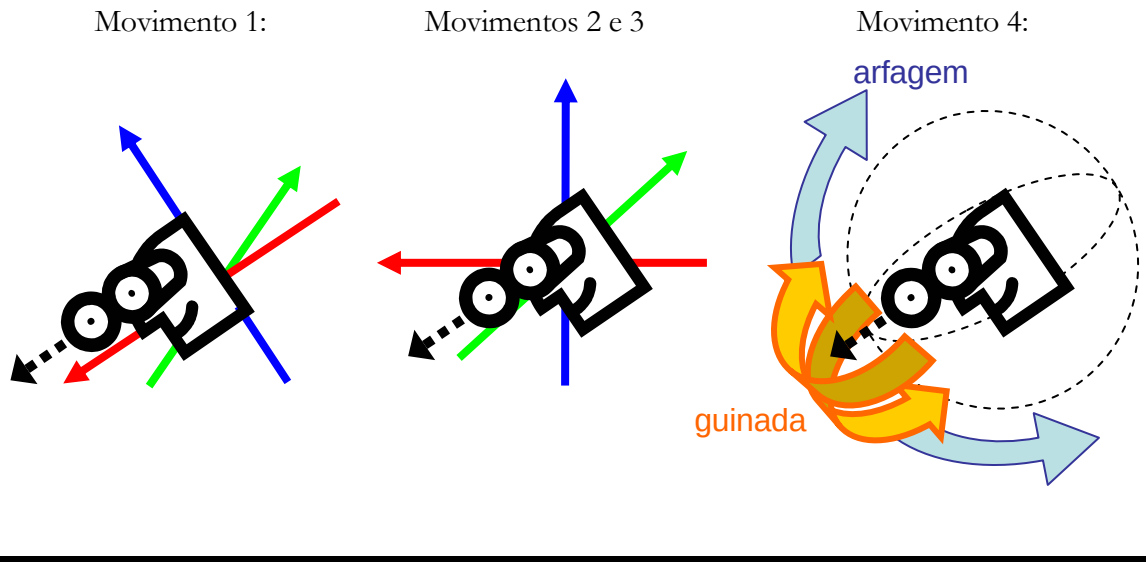


Figura 27: Ilustração dos movimentos possíveis ao usuário na simulação.

Capítulo 5 :

Análise de Desempenho

5.1. Experimentos

A fim de testar a influência na eficiência de diversos aspectos da simulação e o desempenho dos algoritmos implementados, alguns experimentos foram realizados. Os experimentos procuram comprovar o ganho de desempenho obtido com a aplicação de níveis de detalhe aos veículos, além de testar a degradação desse desempenho com a variação no número de junções e na forma do grafo de tráfego.

A máquina onde os experimentos foram realizados foi um PC com processador de 2.2 GHz, 1GB de memória e placa gráfica GeForce FX 5200© 128MB com sistema operacional Windows XP©. Os medidores de desempenho consistiram no consumo de memória e na taxa média de quadros por segundo em que a simulação funcionou.

Apenas uma variável da simulação teve seu valor modificado por vez em cada experimento, a fim de garantir a relação causa→efeito entre o valor da variável analisada e o desempenho da aplicação. Todos os experimentos realizados utilizaram a geração automática de grafo e terreno. Isso ajudou na manutenção da consistência dos dados e no teste mais rápido de diversas situações.

Antes de iniciar o processo de medição da taxa média da simulação, aguardou-se um tempo de aproximadamente 5 minutos. Fez-se isso para que a medida dessa taxa fosse feita quando o modelo estivesse numa situação estável. Essa situação ocorre quando todos os semáforos já entraram em seu ciclo de funcionamento normal, o fluxo de veículos já está distribuído uniformemente por toda a malha retangular de tráfego e o fluxo de entrada e saída de veículos se tornaram praticamente constantes. Em alguns experimentos, esse tempo de espera teve de ser de 10 minutos. Em todos os experimentos o número máximo de veículos na simulação foi estipulado em quinhentos.

Após os cinco minutos de espera, aproximadamente mais cinco minutos foram dedicados à medida da taxa média de quadros por segundo. A variação desse tempo entre 5 ou mais minutos é conseqüência da amostragem ter sido feita a cada 0,5 segundo passados no em relação ao tempo de simulaçãoe não do tempo real decorrido. Assim, o número de amostras realizadas é um total de

600, mas elas podem ter sido realizadas num intervalo real de mais de 5 minutos, caso a taxa de quadros esteja abaixo de 25 quadros/segundo. A taxa de quadros para cada teste dos experimentos foi, então, calculada a partir da média dessas 600 medidas.

Para facilitar a reprodução dos experimentos, uma tabela ou um quadro com as configurações utilizadas para o grafo de tráfego é apresentado ao começo de cada experimento. As especificações dos mesmos são apresentadas nas seções 5.1.1 e 5.1.2 a seguir. Os resultados e a análise dos experimentos aqui definidos estão descritos na seção 5.2.

5.1.1. Teste de Desempenho de LOD em Veículos

O primeiro experimento avalia o desempenho no uso de níveis de detalhe nos avatares dos veículos. A avaliação nesse experimento foi baseada apenas na taxa de quadros/segundo da simulação. As configurações do grafo nos quais o experimento se baseou podem ser vistas na Tabela 8.

Tabela 8: Configuração do grafo gerado automaticamente para o experimento de teste de desempenho dos LOD em veículos.

N	M	d	f	n_f	l
3	4	200	60	5	3

Os campos da Tabela 8 são os seguintes: N – número de linhas contendo nós do grafo de tráfego, M – número de nós por linha do grafo de tráfego, d – distância entre nós do grafo de tráfego, f – fluxo de veículos por minuto, nf – número de faixas em cada conexão e l – largura das faixas em metros.

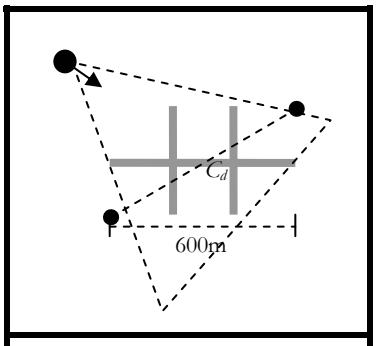


Figura 28: Posicionamento do usuário no experimento 1.

Inicialmente, foram realizados seis testes. Neles, os avatares dos veículos possuíam um nível de detalhe por vez, sendo representados para cada teste por um dos seis modelos disponíveis de veículos. Em todos esses testes o usuário foi posicionado afastado do grafo de forma que pudesse visualizá-lo por completo, como apresentado na Figura 28. A taxa média de quadros/segundo foi medida para cada um dos testes.

Em seguida, um sétimo teste foi realizado, onde os avatares dos veículos possuíam simultaneamente todos os seis níveis de detalhe possíveis. Devido à

mudança no nível de detalhe do veículo, percebeu-se durante os experimentos que, dependendo da posição do usuário, a simulação poderia apresentar taxas de quadros/segundo com valores bastante variados. Para melhor calcular o valor médio dessas taxas, identificou-se três possíveis situações básicas de posicionamento do usuário que poderiam ocorrer durante sua navegação e que afetariam de maneira diferente o modo de representação dos veículos por seus avatares. Esse sétimo teste consistiu de um teste para cada uma dessas situações. Sua taxa de quadros/segundo resultou da média dos três valores obtidos. As três situações detectadas estão descritas abaixo e podem ser vistas na Figura 29, juntamente com o formato do grafo de tráfego escolhido para o experimento.

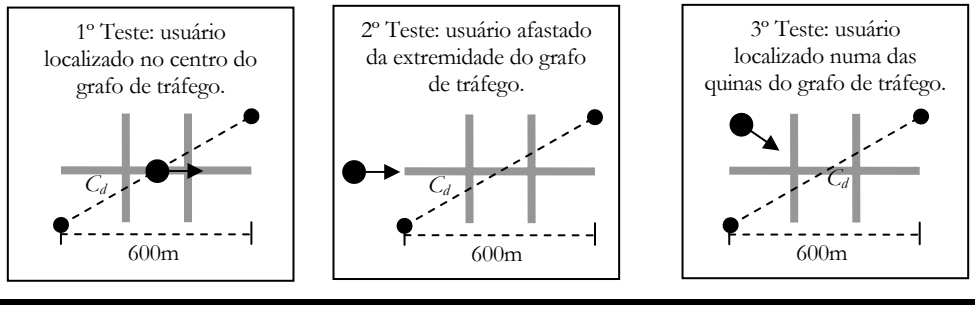


Figura 29: Posicionamentos do usuário no teste 7 do experimento 1 e formato do grafo utilizado.

Na primeira, o usuário é posto no centro do grafo de tráfego a uma altura de 1,5 m da superfície das pistas. Nessa situação, os níveis de detalhe dos avatares dos veículos variam de forma concêntrica ao redor do usuário. Os níveis com menor distância de ativação são os ativados nessa posição.

Na segunda situação, o usuário está situado afastado de uma das conexões de entrada mais distantes da grade de vias de tráfego a uma altura de 1,5 m da superfície das pistas. Nesse segundo caso, os níveis de detalhe dos avatares dos veículos também variam de acordo com limites concêntricos ao usuário. Entretanto, nessa posição é possível se ter veículos com avatares em todos os níveis de detalhe possíveis. Isso por que o limite de ativação do último nível de detalhe do veículo é igual a *600m,* que é também igual ao comprimento *D* de uma das dimensões do grafo sendo simulado, aliás a maior delas. Observe:

$$D = d \times (m-1) = 200 \times 3 = 600m.$$

O valor *m-1* representa o número de arestas entre *m* junções. Multiplicando-se pela distância *d* entre os pares de junções, tem-se o comprimento total de uma das linhas do grafo. Como o número de colunas M é maior que o número de linhas N no grafo especificado, essa é a

maior das distâncias entre conexões de entrada paralelas conseguida no grafo. Estando o usuário afastado a menos de *10m* da extremidade, valor esse dado para a distância máxima para apresentação do nível de maior detalhe, como visto na seção 3.5.3, todos os níveis de detalhe terão possibilidade de aparecer na simulação.

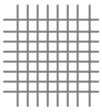
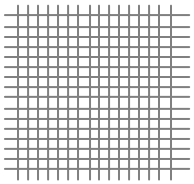
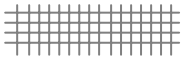
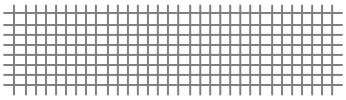
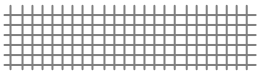
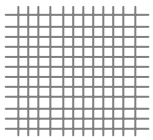
O terceiro cenário posiciona o usuário numa das quinas do grafo de simulação retangular e a 50 m de altura em relação à superfície das pistas. Se $D = 600m$, então o comprimento diagonal C_d do grafo de tráfego é maior que *600m* e isso garante a abrangência dos níveis de menor detalhe dos veículos. Entretanto, como o usuário está afastado de qualquer pista, essa situação beneficia níveis de detalhe com maior distância de ativação.

5.1.2. Teste de Desempenho de acordo com a Quantidade de Junções

O segundo experimento tenta analisar a influência da quantidade e posicionamento das junções no desempenho da simulação. O objetivo é testar o desempenho do algoritmo de criação de caminhos para os veículos e o desempenho quanto ao número de conexões de entrada criadas. A avaliação nesse experimento foi baseada tanto na taxa de quadros/segundo quanto no consumo de memória da simulação.

Para realizar o experimento, foram feitos sete testes, cada um com um grafo de tráfego com forma ou número de junções diferentes. Três deles são quadrangulares enquanto os outros quatro são retangulares. A fim de distingui-los durante a análise, foi dado um número de identificação a cada um deles. Além disso, devido à grande quantidade de junções presentes, aos grafos 1 e 3 foi dado um tempo de espera de 10 minutos antes de se começar a medição das suas taxas de quadros/segundo. As configurações para cada um desses grafos podem ser vistas no Quadro 4.

Quadro 4: Configuração dos grafos gerados automaticamente para o experimento de teste de desempenho de acordo com a quantidade de junções.

Id	*N*	*M*	*Representação do Grafo*	*Tempo de espera*	*d*	*f*	n_f	*l*
0	**8**	**8**		**5**	100	5	3	3
1	**16**	**16**		**10**	100	5	3	3
2	**4**	**16**		**5**	100	5	3	3
3	**8**	**32**		**10**	100	5	3	3
4	**3**	**48**		**5**	100	5	3	3
5	**6**	**24**		**5**	100	5	3	3
6	**12**	**12**		**5**	100	5	3	3

Os campos do Quadro 4 são os seguintes: *Id* – número de identificação do grafo de tráfego, *N* – número de linhas contendo nós do grafo de tráfego, *M* – número de nós por linha do grafo de tráfego, *Representação do grafo* – configuração espacial da malha de tráfego que representa o grafo, *Tempo de espera* – tempo inicial de espera anterior à medição da taxa de quadros/segundo necessário para a estabilização do fluxo de tráfego em todo o modelo, *d* – distância entre nós do grafo de tráfego, *f* – fluxo de veículos por minuto, n_f – número de faixas em cada conexão e *l* – largura das faixas em metros.

Os avatares dos veículos só possuíam um nível de detalhe, o LOD3, cujas configurações poligonais podem ser vistas no Quadro 3 da seção 3.5.3. Essa alteração foi feita para garantir que a simulação fosse executada com razoável eficiência para diferentes níveis de complexidade. Um nível mais baixo de detalhe não foi escolhido para garantir que as variações na quantidade de veículos pudessem causar impacto nas medidas de taxa de quadros/segundo em cada teste.

Durante esses testes, o usuário ficou posicionado numa pista próxima do ponto médio do grafo. Nenhuma outra variável sofreu alteração como pôde ser visto no Quadro 4 acima.

5.2. Resultados e Análise

Abaixo são apresentados os resultados dos dois testes propostos previamente. Uma análise dos resultados também é realizada.

5.2.1. Teste de Desempenho de LOD em Veículos

A seguir, no Quadro 5 e na Figura 30, são apresentadas as taxas de quadros por segundo para os sete testes realizados durante o primeiro experimento.

Quadro 5: Resultados das taxas de quadros/segundo para cada teste do experimento 1.

Número do teste	*Nível de detalhe do modelo do veículo*	*Taxa média de quadros por segundo*	*Desvio padrão*
0	LOD 0	1,84	0,14
1	LOD 1	4,41	0,37
2	LOD 2	20,96	3,18
3	LOD 3	29,39	5,13
4	LOD 4	40,70	14,35
5	LOD 5	51,55	15,81
6	Todos LOD juntos	(12,00+23,42+35,46)/3 = 23,63	(1,62+5,73+10,89)/3 = 6,08

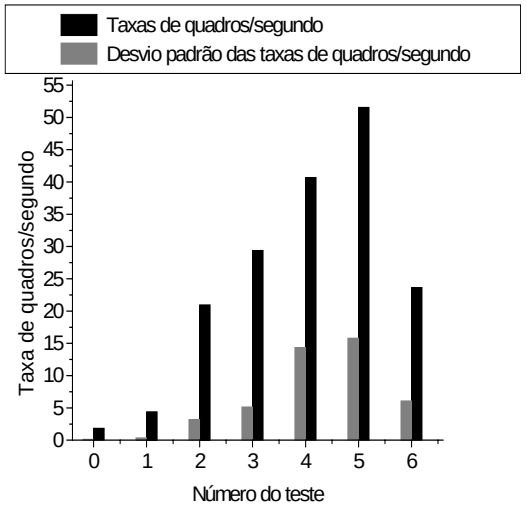


Figura 30: Gráfico com resultados das taxas de quadros/segundo para cada teste do experimento 1.

É perceptível a diminuição das taxas de quadros/segundo da simulação quando níveis de maior detalhe são aplicados. Ainda, percebe-se que essas taxas tiveram um aumento de desempenho menor à medida que seu nível de detalhe decaiu, considerando-se o aumento do desvio padrão para cada uma das médias.

A técnica de níveis de detalhe à simulação mostrou-se eficaz. Além da mudança de qualidade ser praticamente imperceptível, o desempenho da simulação ficou perto da média entre os valores medidos para os outros testes, onde apenas um nível detalhe por vez foi

utilizado. A alta variação poligonal entre modelos com mais detalhe fez com que esse resultado ficasse abaixo da média, como pode ser visto na Figura 30.

A seguir, da Figura 31 à Figura 36, são apresentados gráficos com as amostras de taxas de quadros/segundo coletadas e suas médias nos seis primeiros testes do experimento, onde foi considerado um único nível de detalhe por vez para os veículos.

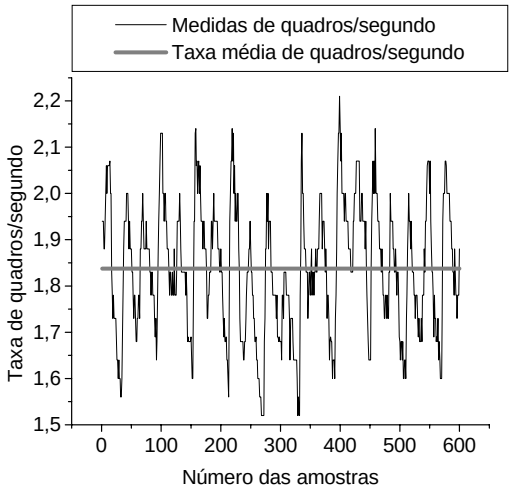


Figura 31: Taxas de quadros/segundo do experimento 1 com LOD 0.

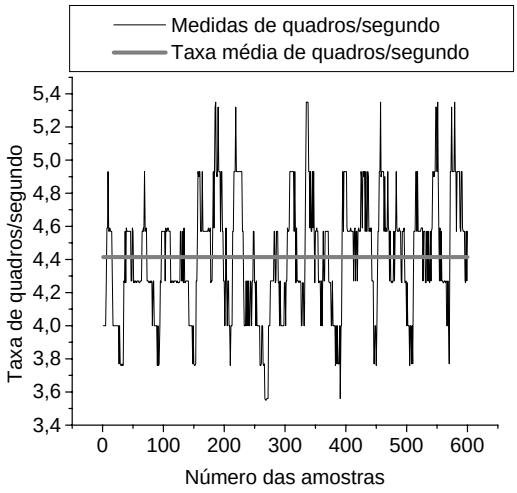


Figura 32: Taxas de quadros/segundo do experimento 1 com LOD 1.

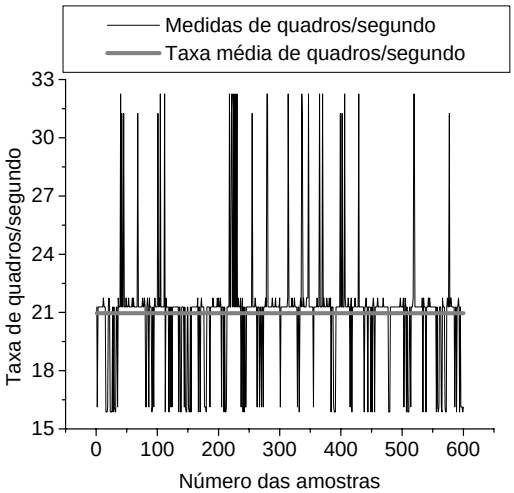


Figura 33: Taxas de quadros/segundo do experimento 1 com LOD 2.

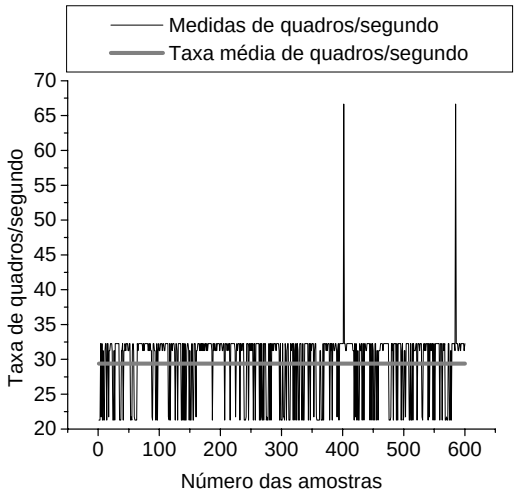


Figura 34: Taxas de quadros/segundo do experimento 1 com LOD 3.

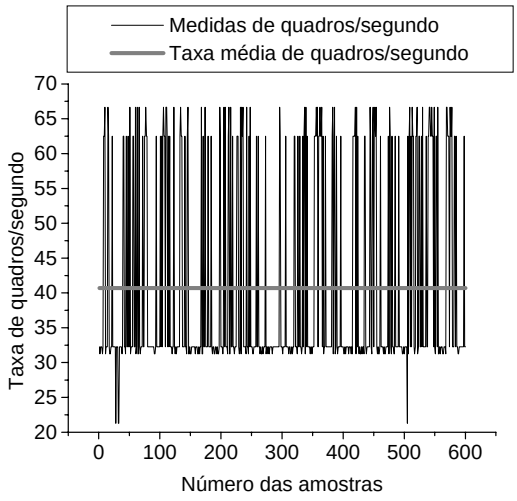


Figura 35: Taxas de quadros/segundo do experimento 1 com LOD 4.

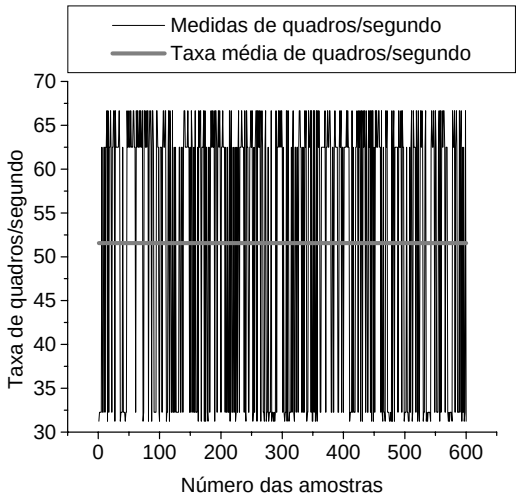


Figura 36: Taxas de quadros/segundo do experimento 1 com LOD 5.

A seguir, da Figura 37 à Figura 40, são apresentados gráficos com as amostras de taxas de quadros/segundo coletadas e suas médias para o sétimo teste com o usuário em três posições diferentes, além de um último gráfico mostrando a média das amostras em cada coleta e o valor médio dessas médias.

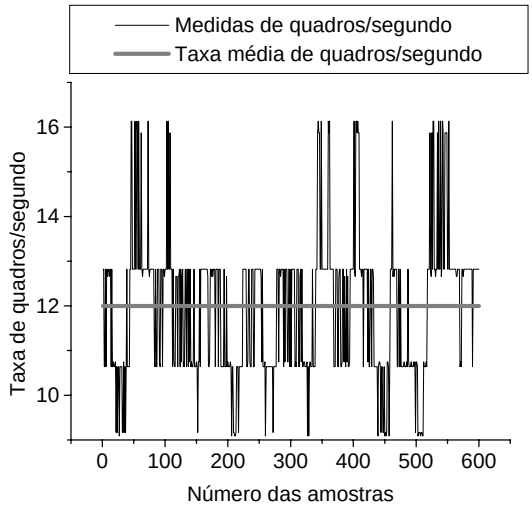


Figura 37: Taxas de quadros/segundo do experimento 1 com todos os LOD com o usuário no centro do grafo.

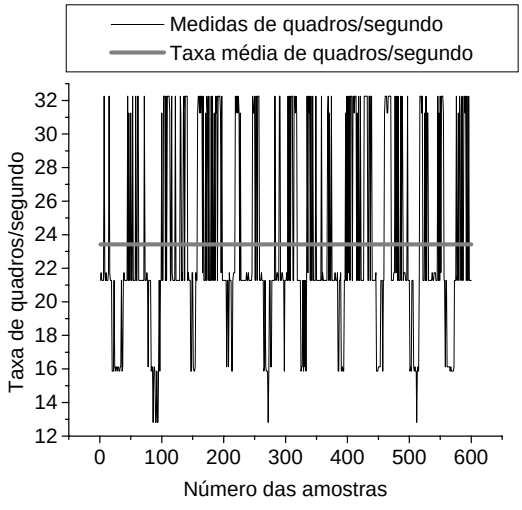


Figura 38: Taxas de quadros/segundo do experimento 1 com todos os LOD com o usuário na extremidade mais distante das pistas do grafo.

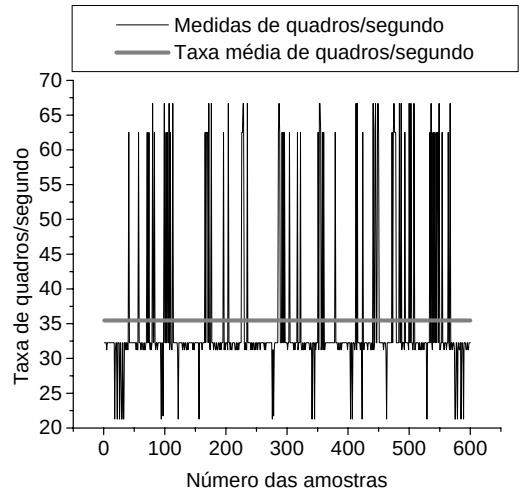


Figura 39: Taxas de quadros/segundo do experimento 1 com todos os LOD com o usuário na quina do retângulo que engloba o grafo.

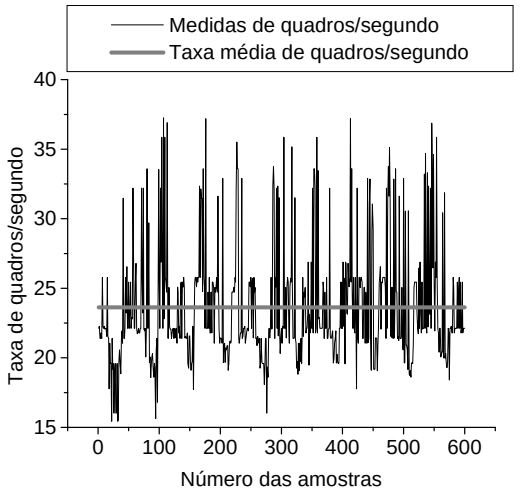


Figura 40: Taxas de quadros/segundo da média das amostras dos três casos para o sétimo teste do experimento 1.

A Figura 41 traça um histograma que foi definido com os dados do experimento 1 para o teste com veículos sendo representados por apenas o LOD3, dados esses que foram apresentados anteriormente na Figura 34. Escolheu-se os dados desse teste por possuírem a menor quantidade de diferentes valores dentre todos os testes realizados para esse experimento.

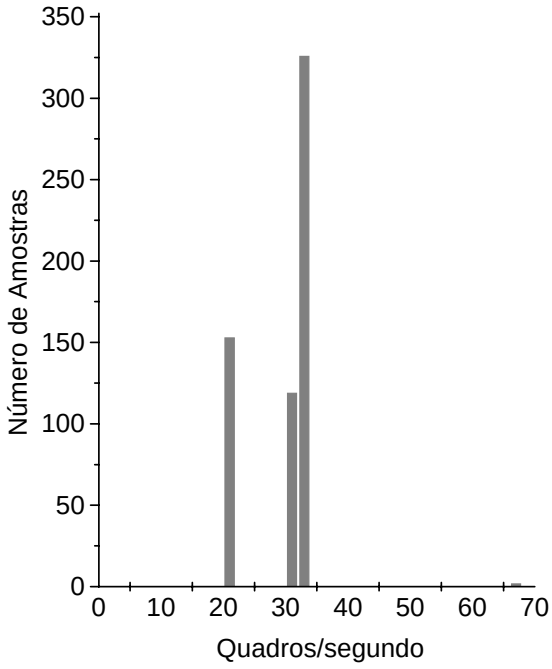


Figura 41: Histograma com os dados do teste do experimento 1 com veículos representados por apenas o LOD1.

Observa-se que os dados seguem uma distribuição multimodal, com a existência de vários picos separados. O maior desses picos está situado entre os valores de *30* e de *35* quadros/segundo, indicando que a maioria dos valores amostrados se encontram nesse intervalo. Acredita-se que o motivo para tal distribuição seja a pouca aleatoriedade dos tempos de entrada dos veículos na simulação, uma vez que essa falta de variação de valores não apareceu tão evidentemente nos resultados de outros testes. Entretanto, nenhuma evidência experimental foi ainda descoberta para comprovar essa afirmação.

5.2.2. Teste de Desempenho de acordo com a Quantidade de Junções

A seguir, na Tabela 9 e na Figura 42, são apresentadas as taxas de quadros por segundo e os níveis de consumo de memória para os sete grafos testados durante o segundo experimento.

Tabela 9: Taxas de quadros/segundo e níveis de consumo de memória para cada grafo testado no experimento 2.

Id do grafo	*Taxa de atualização (quadros por segundo)*	*Desvio padrão*	*Consumo de memória (MB)*
0	54,98	14,50	45,552
1	15,03	1,49	195,908
2	61,22	9,28	40,344
3	15,24	1,37	185,528
4	20,97	3,11	71,496
5	18,26	2,65	98,692
6	17,38	2,37	106,472

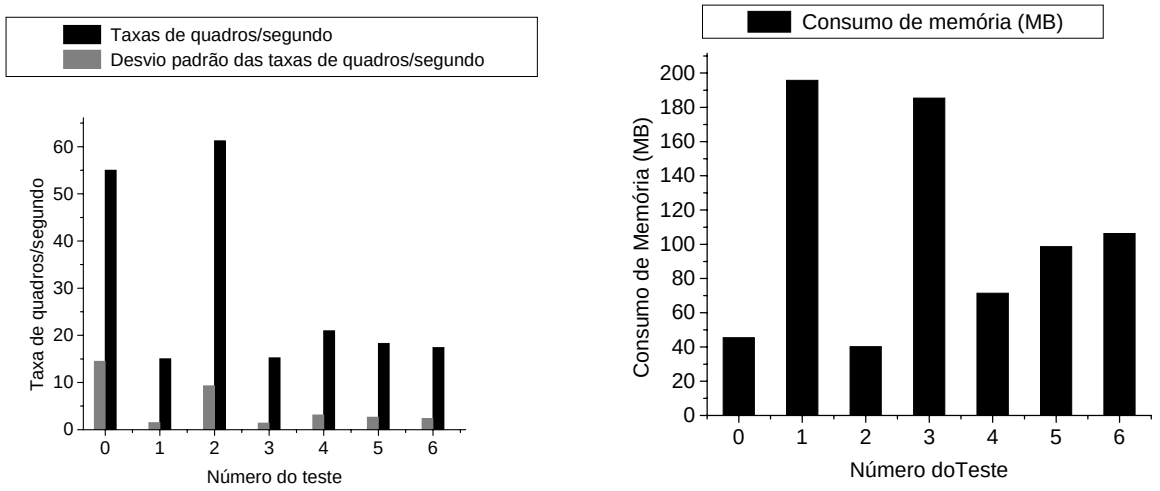


Figura 42: Gráficos com resultados de taxas de quadros/segundo e consumo de memória dos testes do experimento 2.

A seguir, da Figura 43 à Figura 49, são apresentados gráficos com as amostras de taxas de quadros por segundo coletadas e suas médias dos sete testes do experimento 2, onde foram considerados diferentes grafos como mostrado na seção 5.1.2.

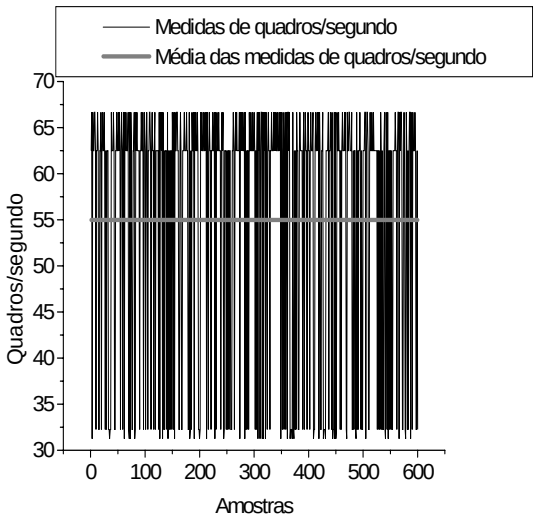


Figura 43: Taxas de quadros/segundo do experimento 2 para o grafo 0.

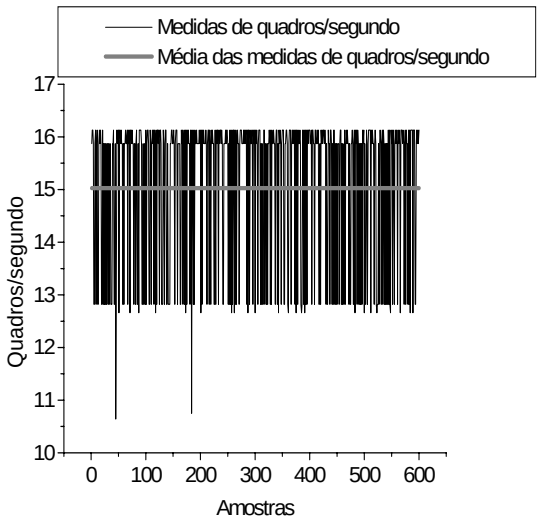


Figura 44: Taxas de quadros/segundo do experimento 2 para o grafo 1.

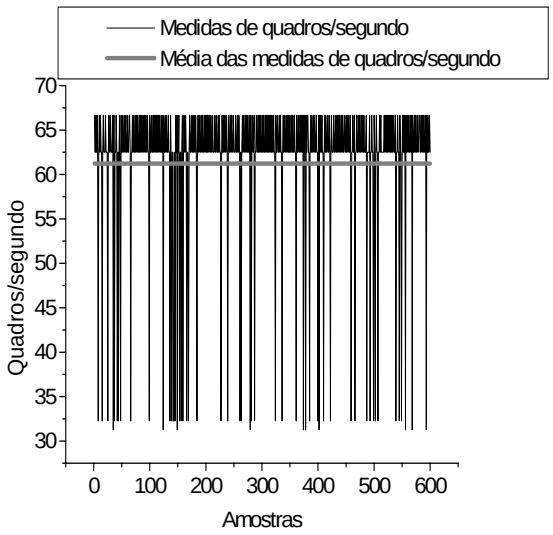


Figura 45: Taxas de quadros/segundo do experimento 2 para o grafo 2.

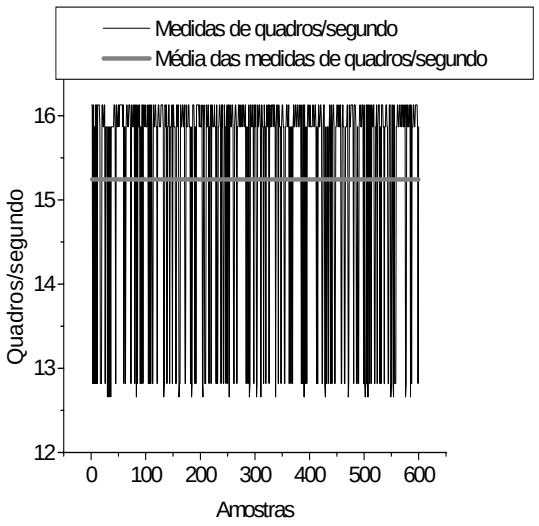


Figura 46: Taxas de quadros/segundo do experimento 2 para o grafo 3.

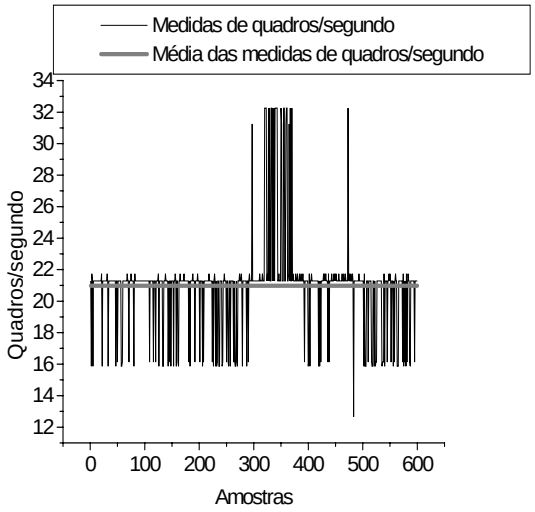


Figura 47: Taxas de quadros/segundo do experimento 2 para o grafo 4.

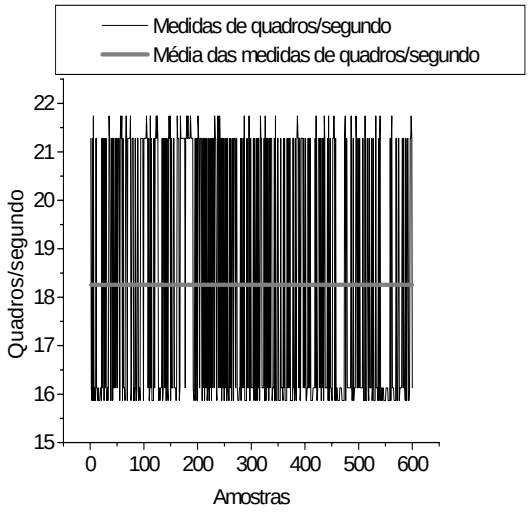


Figura 48: Taxas de quadros/segundo do experimento 2 para o grafo 5.

Comparando-se os pares de grafos: 0 e 2, 1 e 3 - duplas de grafos com mesmo número de junções - percebe-se pouca variação nas suas taxas em quadros/segundo. Isso indica que o formato do grafo da simulação tem pouca influência no desempenho da simulação. Esse resultado é comprovado também na análise dos grafos 4, 5 e 6, que também possuem o mesmo número de junções e cujas taxas de quadros/segundo são praticamente as mesmas.

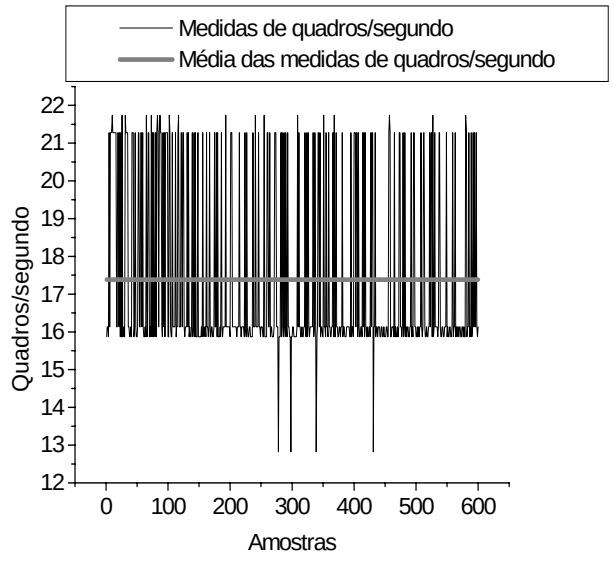


Figura 49: Taxas de quadros/segundo do experimento 2 para o grafo 6.

Entretanto, percebe-se um crescimento no consumo de memória entre os pares de grafos 4 e 5, 5 e 6, 2 e 0, 3 e 1, onde o primeiro grafo de cada uma dessas duplas consome menos memória que o segundo. Isso se deve ao fato desse primeiro grafo possuir menos arestas internas ao grafo que o segundo, como visto na Tabela 10. Assim, o número de conexões entre as junções é menor no primeiro que no segundo e, por conseqüência, a simulação envolvendo o primeiro consome menos memória que o segundo.

Tabela 10: Número de conexões de entrada para cada grafo do experimento 2.

Grafo	*Dimensões em número de junções*	*Número de conexões de entrada*
0	8×8	2×(8-2) + 2×(8-2) = 6+6 = **24**
1	16×16	2×(16-2) + 2×(16-2) = 14×14 = **56**
2	4×16	2×(4-2) + 2×(16-2) = 2+14 = **32**
3	8×32	2×(8-2) + 2×(32-2) = 6+30 = **72**
4	3×48	2×(3-2) + 2×(48-2) = 2+92 = **94**
5	6×24	2×(6-2) + 2×(24-2) = 8+44 = **52**
6	12×12	2×(12-2) + 2×(12-2) = 20+20= **40**

Em contrapartida, ainda considerando essas duplas de grafos, todos os primeiros grafos possuem mais arestas de entrada que o segundo. Isso acarreta um volume maior de veículos trafegando e conseqüentemente causa um aumento de processamento e redução nas taxas de atualização. Isso não foi percebido nos resultados obtidos dessas simulações, entretanto. A justificativa para isso está no fato de, em grafos retangulares, a probabilidade de um veículo cruzar o grafo de um lado a outro é maior que em grafos quadrangulares, o que leva a uma redução da média do tamanho dos caminhos dos veículos. O aumento de fluxo em mapas retangulares, com mais conexões de entrada é compensado pela redução do tempo de estada dos veículos no grafo de tráfego, não afetando assim a taxa de quadros/segundo da simulação.

A Figura 50 define um histograma com os dados do teste do experimento 2 para o grafo 4, dados esses apresentados anteriormente na Figura 47. Como no histograma do experimento 1, escolheu-se esse teste por que seu gráfico possui a maior quantidade de valores diferentes entre todos os testes realizados no experimento 2.

Observa-se que os dados também seguem uma distribuição multimodal, com a existência de vários picos separados. O maior desses picos está situado entre os valores de *20* e *22* quadros/segundo, indicando que a maioria dos valores amostrados se encontram nesse intervalo. A teoria de que a pouca aleatoriedade dos tempos de entrada dos veículos na simulação esteja causando esses resultados também se aplica nesse caso.

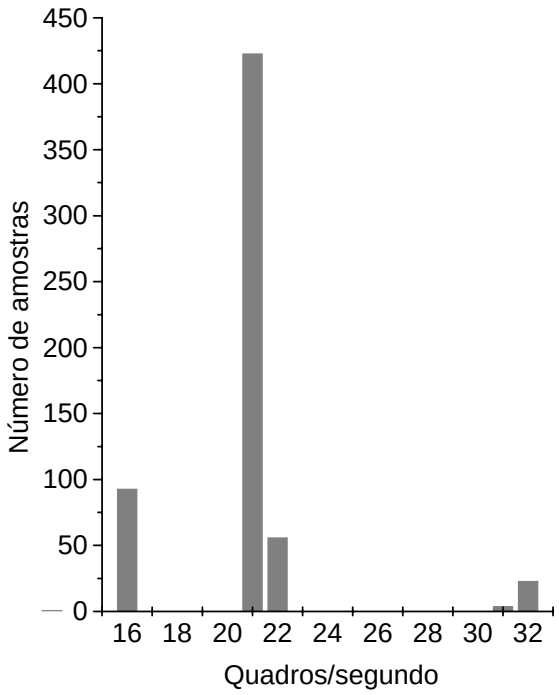


Figura 50: Histograma com os dados do teste do experimento 2 com grafo 4.

Capítulo 6 :

Conclusões

Com base nos experimentos realizados e na análise de seus resultados, fica evidente a eficiência em se simular tráfego em ambientes tridimensionais. A presença de uma terceira dimensão, apesar de sua eficácia ter sido apenas parcialmente comprovada nessa versão do trabalho, só tem a contribuir em aplicações desse tipo. Ela permite a superposição de mais informações simultaneamente, a navegação mais eficaz, a criação de ambientes mais realistas, além de análises mais precisas com base em dados de difícil visualização em interfaces bidimensionais como as de relevo e as envolvendo medidas de altura.

Apesar das interfaces 2D atenderem à grande parte das necessidades de simuladores de trânsito, elas não são completas. A visualização e navegação entre pontos distantes apresentados em perspectiva só são possíveis com o uso de uma interface tridimensional. Somente com essa terceira dimensão é que a interação entre o relevo, solo e a construção de pistas se torna implementável também. Essa interação permite a estimativa de custos baseada no tipo de construção que será feita para adaptar-se ao relevo, seja ela um viaduto, ponte ou túnel.

Porém, é importante ressaltar que se deve ter cautela na construção de interfaces tridimensionais. Ambientes de simulação de tráfego envolvem um número grande de agentes e uma quantidade de dados processados em tempo real maior que em interfaces bidimensionais. O uso de níveis de detalhes nos avatares, além da otimização de algoritmos envolvendo comportamento de veículos, é fundamental para o ganho de desempenho no processamento monolítico.

Entretanto, a distribuição dessa simulação é essencial para que a abrangência da região de tráfego simulada seja ampliada. A estrutura se mostrou eficiente e a idéia de divisão em junções parece ser a melhor forma de homogeneizar o processamento entre máquinas. Uma possibilidade que pode vir a aumentar o potencial de utilização dessas unidades é a criação de agrupamentos de junções cuja soma dos fluxos de tráfego seja semelhante. O grau de aglomeração seria regulado de acordo com a intensidade média do fluxo em cada junção e com a capacidade de processamento de cada unidade autônoma distribuída. Esses valores seriam otimizados para que cada unidade tivesse

seu processamento utilizado quase totalmente. O mesmo se aplicaria às unidades controladoras do terreno e outros objetos na cena virtual.

Acredita-se que a expansão deste modelo de simulação para conter mais tipos de veículos, com comportamentos mais complexos que os aqui apresentados, é possível. A inserção de elementos de controle de poluição e de consumo de combustível, de fluxo de ciclistas e de pedestres, de linhas de ônibus e de simulações de batidas entre veículos também parecem viáveis. Contudo, quanto maior a complexidade do modelo, maior o volume de dados a ser processado e, como conseqüência, maior a necessidade de sua distribuição.

Com a inserção de um protocolo de comunicação em rede e com a reformulação de algumas estruturas de dados, a arquitetura apresentada parece ser capaz de se adaptar facilmente a um modelo distribuído. Com essa adaptação, a construção de uma cena de trânsito envolvendo mais variáveis e conseqüentemente cada vez mais realista se torna um projeto cujos resultados certamente contribuirão significativamente para a análise e planejamento do tráfego urbano.

6.1. Dificuldades Encontradas

Apesar dos bons resultados quanto ao funcionamento da simulação, é fundamental mencionar os problemas encontrados no seu desenvolvimento. O principal deles consiste na regulagem dos algoritmos comportamentais dos veículos. A influência de um número muito grande de forças no veículo muitas vezes produz movimentos abruptos que alteram erraticamente sua trajetória durante o percurso.

Durante algumas das simulações acima, ocorreram problemas esparsos com algum dos veículos simulados, que estagnavam ao chegar a pontos de entrada de algumas das pistas. Os experimentos foram simulados novamente com avatares de veículos com menor qualidade gráfica e o problema desapareceu. Ainda não foi identificada a causa do problema, mas acredita-se que ele ocorra durante a utilização do algoritmo de separação.

O algoritmo de separação encontra-se ainda bastante instável. Quando aplicado em situação de tráfego intenso, ele gera movimentos aparentemente senoidais nos veículos, fazendo-os sair das faixas onde deveriam permanecer. A fim de minimizar este problema, a força de influência desse comportamento no veículo foi reduzida. Além disso, o veículo passou também a sofrer uma redução em sua velocidade ao se encontrar a uma distância intermediária dentro de uma faixa, com a força de separação entre veículos de uma mesma faixa ativada.

Um outro problema é a mudança excessiva de faixas pelos veículos, o que torna o trânsito um pouco mais caótico que o normal. Além disso, em vias de tráfego de mão dupla, não há controle da interação entre fluxos de tráfego de sentidos opostos que se cruzam. Isso adiciona um nível de complexidade bem mais alto ao comportamento dos veículos.

A criação automática de grafos de tráfego em muito ajudou na realização de experimentos. Entretanto, a sincronização dos semáforos ainda se encontra em fase experimental e foi causa de muitas colisões durante a simulação.

6.2. Contribuições

Esta dissertação contribui em diversas áreas da pesquisa, entre as quais pode-se citar a simulação de tráfego 3D, os jogos eletrônicos e realidade virtual. As principais contribuições deste trabalho estão descritas a seguir.

Apesar da grande quantidade de ferramentas atualmente disponíveis para simulação de trânsito, não se encontrou nenhuma com interface tridimensional que seja executada sem a ajuda de supercomputadores. Apesar de o trabalho aqui realizado ainda se encontrar num estágio inicial, deve-se ressaltar seu potencial para execução em computadores PCs domésticos.

O gerador automático de tráfego é bastante útil na elaboração inicial de malhas de trânsito urbano para jogos eletrônicos e ambientes virtuais de uma forma geral. O relacionamento dos usuários como pedestres e dos veículos como entidades automáticas poderia ser implementado utilizando-se mecanismos de comunicação em rede. A sua adaptação para gerar um arquivo contendo as informações do grafo construído automaticamente e a sua posterior manipulação através de um editor de grafos pode servir não só para a criação de cidades fictícias, mas também de cidades reais com uma malha de tráfego bem organizada.

O mapeamento de um grafo de tráfego, através da definição de pontos e arestas num arquivo, agiliza, mesmo que em parte, o processo de mapeamento de tráfego de uma determinada região. Contudo, a inserção de um editor gráfico para construção do grafo é uma etapa importante a ser implementada.

Finalmente, a modificação do algoritmo *pathfollowing*, que, ao invés de considerar o afastamento do veículo da faixa, passou a utilizar a distância entre vetores de velocidade e, no caso do algoritmo *separation*, teve sua abrangência reduzida a veículos mais próximos, contribuiu significativamente para a redução do processamento da simulação.

6.3. Trabalhos Futuros

O modelo de simulação atual já permite a previsão de situações de tráfego simples de maneira razoavelmente realista. Entretanto, muito ainda há que ser feito para que o corrente modelo atinja um nível suficientemente flexível para simular qualquer situação de tráfego. A seguir, são descritos os principais passos a serem realizados para que isso se concretize.

6.3.1. Aprimoramento dos Comportamentos dos Veículos

Primeiramente, o comportamento dos veículos precisa ser aprimorado em diversos aspectos. O algoritmo de separação precisa ser mais estável, para que a força exercida seja capaz de evitar a maioria das colisões entre veículos.

A área de colisão de cada veículo precisa ser mais precisa em relação ao modelo que o representa, para que os eventos de colisão ocorram somente quando uma interseção entre os avatares dos veículos ocorrerem. Além disto, esse algoritmo deve detectar melhor a posição relativa entre o veículo que colide e o veículo atingido, para que possa deslocar os carros de diferentes formas de acordo com suas posições. Uma detecção de colisão entre veículos numa mesma junção, mas de conexões diferentes, também precisa ser implementada.

O comportamento de *pathfollowing* necessita ser otimizado para o modelo de interpolação de *Runge-Kutta* (Mathworld, 2004), cuja precisão é bem melhor que o modelo de *Euler* atualmente utilizado. Isso tornará o movimento dos veículos menos suscetíveis às variações das taxas de atualização da simulação.

Os veículos necessitam ainda de comportamentos que lhes permitam cruzarem faixas de sentidos opostos sem colidirem com os veículos nessas outras faixas. Também devem tornar-se capazes de estacionar em pontos comerciais e em ruas onde o estacionamento seja permitido. Além disso, um mecanismo de ultrapassagem deve ser implementado de acordo com o comportamento psicológico dos motoristas. À medida que outras características forem inseridas no modelo, como pedestres, paradas de ônibus e variáveis climáticas, o comportamento de um veículo passará a ser mais realista.

6.3.2. Modificação para Distribuição da Simulação

Em segundo lugar, algumas alterações também serão necessárias para distribuir a simulação em unidades autônomas. Elas estão listadas a seguir.

Os métodos de transmissão de veículos entre conexões deverão ser feitos através de um protocolo de rede. O mesmo deverá ocorrer com os métodos de desenho de cada um dos objetos gráficos do modelo, pois as informações de cada modelo como referências aos níveis de detalhe, à posição e à orientação do avatar serão transmitidas da máquina que controla cada junção para a máquina do usuário. A máquina do usuário armazenará todos os avatares gráficos possíveis na simulação e os desenhará para cada objeto cujas informações estão sendo recebidas.

As estruturas de lista de veículos inativos e de grafo de tráfego, atualmente localizadas na classe Simulação, estarão contidas em cada junção, garantindo às últimas a autonomia necessária para serem executadas cada uma em máquinas diferentes.

Na distribuição do processamento, a classe ITranS_C servirá para iniciar, sincronizar e monitorar o funcionamento do sistema, sendo executada em uma máquina em separado junto à classe Simulação. O controle do usuário também não estará mais vinculado a essa classe, pois será executado autonomamente em um outro ponto da rede. Isso garantirá a entrada de mais de um usuário numa mesma simulação. Um controle de entrada de usuários deverá, então, ser criado. O modelo poderá prover, também, a possibilidade desses usuários dirigirem ou acompanharem o movimento de um veículo no trânsito.

A inserção de agrupamentos de junções servirá para reduzir o tráfego de rede quando existirem muitas junções com pouco fluxo de tráfego. Em contrapartida, um algoritmo que as agrupe de forma a distribuir o processamento homogeneamente entre máquinas terá de ser implementado. Ele deverá levar em consideração o fluxo de tráfego das junções e a capacidade de processamento de cada máquina durante a criação e distribuição desses agrupamentos de junções.

6.3.3. Aprimoramento das Informações Topográficas

Em terceiro lugar, a inserção de elementos topográficos ao modelo deverá ser feita de modo a garantir uma melhor orientação do usuário quanto à região sendo analisada. Algoritmos para posicionamento e ajuste de escala automática do relevo em relação ao grafo de tráfego devem ser implementados, a fim de reduzir o trabalho manual durante a construção do ambiente de simulação. Ainda, um editor de grafos deve ser criado sobre um mapa da região cuja malha de tráfego será analisada para que os pontos e arestas do grafo de tráfego sejam desenhados sobre o mapa, ao invés de terem suas coordenadas definidas em arquivo.

Uma outra possibilidade é a padronização da interface de configuração de dados, que passaria a receber dados reais, extraídos de mapas com diferentes informações geográficas. Isso faria com que todo o processo de obtenção de dados fosse feito automaticamente.

6.3.4. Sistema de Auto-ajuste dos Níveis de Detalhe

Por último, a distância de ativação e a utilização de diferentes níveis de detalhes nos objetos que compõem a cena poderiam variar de acordo com o desempenho da aplicação. Para uma simulação com muitos veículos ou abrangendo uma grande área geográfica, níveis de detalhe com qualidade gráfica inferior seriam utilizados para representar os objetos e veículos. Quando houvesse menos veículos sendo simulados ou se a simulação envolvesse uma área geográfica pequena, os níveis de detalhe dos objetos seriam maiores, o que proveria uma melhor qualidade gráfica à cena.

6.3.5. Considerações Finais

Com essas mudanças, a geração de simulações mais complexas e realistas se dará de forma mais eficiente e menos custosa. O desenvolvimento da maioria delas, porém, dá-se de modo não trivial e requer muito esforço para sua conclusão. A colaboração com grupos de pesquisa em outras áreas, como as de Redes, Sistemas Distribuídos e Teoria da Computação é importante para a aceleração na mudança de arquitetura e no aumento do desempenho algorítmico e da rede da simulação. Ainda, o contato com empresas responsáveis pelo controle de tráfego urbano deve ser levado em consideração para que a interface satisfaça ao usuário final, disponibilizando apenas as funcionalidades realmente necessárias, eliminando a execução de esforços em funcionalidades de utilidade duvidosa e aumentando, assim, o potencial da equipe de pesquisa e desenvolvimento.

Bibliografia

A bibliografia está organizada em duas seções. A primeira delas inclui as referências que foram citadas na dissertação e utilizadas na pesquisa. A segunda contém as referências que, apesar não mencionadas na dissertação, também foram utilizadas no desenvolvimento da pesquisa.

Referências

[1] Adzima, J., "AI Madness: Using AI to Bring Open-City Racing to Life". Gamasutra, publicado em 24 de Janeiro de 2001. Disponível em: www.gamasutra.com/features/20010124/adzima_01.htm. Acesso em: junho 2004.

[2] Al-Shihabi, T e Mourant, R.R. "A Framework for Modeling Human-like Driving Behaviors for autonomous Vehicles in Driving Simulations". Proceedings of the fifth international conference on Autonomous agents, International Conference on Autonomous Agents, 2001, Montreal, Quebec, Canadá, pgs.: 286 - 291, ISBN: 1-58113-326-X.

[3] Barros, P. G., Kelner, J., "Simulação de Tráfego: uma Experiência com Realidade Virtual". Proceedings of SVR 2003 - VI Symposium on Virtual Reality, editora COC, 2003. v.1. p.140 - 151, Ribeirão Preto, Brasil, outubro de 2003.

[4] Bayarri, S. et al. "Virtual Reality for Driving Simulation". Communications of the ACM, Maio de 1996, pgs.: 72-76, vol. 39, n. 5.

[5] Blue, M. e Bush, B. "Information Content in the Nagel-Schreckenberg Cellular Automaton Traffic Model". Physical Review E **67** (2003), p. 047103.

[6] Brutzman, D. P.; Macedonia, M. R. e Zyda, M. J. "Internetwork Infrastructure Requirements for Virtual Environments". Proceedings of the Virtual Reality Modeling Language (VRML) Symposium, San Diego Supercomputer Center (SDSC), San Diego, CA, 13 a 15 de dezembro, 1995.

[7] Cameron, G.; Wylie, B.J.N e McArthur D. "PARAMICS : Moving Vehicles on the Connection Machine". Proceedings of the 1994 ACM/IEEE conference on Supercomputing, Conference on High Performance Networking and Computing, 1994, Washington, D.C., pgs.: 291 – 300, ISBN ~ ISSN: 1063-9535, 0-8186-6605-6.

[8] Clark, J. e Daigle, G. "Importance of Simulation Techniques in Its Research and Analysis". Proceedings of the 29th Conference on Winter Simulation Conference, Winter Simulation Conference, Atlanta, Georgia, Estados Unidos, 1997, pgs.: 1236 - 1243, ISBN:0-7803-4278-X.

[9] De Floriani, L. e Magillo P., *Regular and Irregular Multi-resolution Terrain Models: a Comparison*, Proceedings - 10th ACM International Symposium on Advances in Geographic Information Systems, pgs.: 143 - 148 , 2002 , ISBN:1-58113-591-2. *World-up R5 User's Guide*, 1997-2000 por Engineering Animation, Inc.;

[10] DeLeon, V.; Berry, R.; *Bringing VR to the Desktop: Are You Game?*, IEEE Multimedia, edição de Abril-Junho de 2000 (Vol. 7, No. 2), pgs.: 68-72.

[11] Díaz, A.; Vázquez, V. e G. Wainer "Vehicle routing in Cell-DEVS models of urban traffic". Proceedings of European Simulation Symposium. Marseille, France. 2001.

[12] Discreet. Discreet website. Disponível em: http://www.discreet.com/. Acesso em: dezembro 2004.

[13] Fullford, D. "Distributed Interactive Simulation: It's Past, Present, and Future". Winter Simulation Conference Proceedings, 1996, pgs.: 179-185.

[14] Harris, L.R.; Jenkin, M.; Zikovitz, D.; Redlick, F.; Jaekl, P.; Jasiobedzka, U.; Jenkin, H. e Allison, R. "Simulating self motion I: cues for the perception of motion". Virtual Reality, 2002, volume 6, number 2, pgs.: 75 – 85.

[15] Hsin, V.J.K. e Wang, P.T.R "Modeling Concepts for Intelligent Vehicle Highway Systems (IVHS) Applications". Proceedings of the 24th conference on Winter simulation, Winter Simulation Conference, 1992, Arlington, Virginia, Estados Unidos, pgs.: 1201 – 1209, ISBN: 0-7803-0798-4.

[16] Jayakrishnan, R. e Mahmassani, H.S. "Dynamic Simulation-Assignment Methodology to Evaluate In-Vehicle Information Strategies in Urban Traffic Networks". Winter Simulation Conference Proceedings, Balci, O. Sadowski, R. P., e Nance, R. (eds.), pgs.: 763 - 769, 1990.

[17] Klein, U.; Schulze, Th.; Straßburger, S. e Menzler, H.P. "Traffic Simulation Based on the High Level Architecture". Proceedings of the 1998 Winter Simulation Conference, eds. Medeiros, D.J. e Ed Watson, SCS, Washington.

[18] Lemessi, M., "An SLX-based Microsimulation Model for A Two-lane Road Section" Proceedings of the 2001 Winter Simulation Conference. WSC 2001, Arlington, VA, USA, 9 a 12 de Dezembro de 2001, pgs.: 1064-1071, ISBN:0-7803-7309-X .

[19] Li, H. e Lim, A. "Local Search with Annealing-like restarts to Solve the vehicle routing Problem with Time Windows". Proceedings of the 2002 ACM symposium on Applied computing, Symposium on Applied Computing, 2002, Madrid, Spain, pgs.: 560 - 565 , ISBN:1-58113-445-2.

[20] Liu, R; Clark, S.D.; Montgomery, F.O. e Tate, J. (2000). *The Microscopic Modelling of Kerb Guided Bus Schemes*. Apresentado em Transport Research Board Annual Conference, Washington, 2000.

[21] Liu, R. e Tate, J. (2000). *MicroSimulation Modelling of Intelligent Speed Adaptation System*. Paper apresentado no European Transport Conference, Cambridge, Setembro de 2000.

[22] Lo Tártaro, M.; Torres, C. e Wainer, G. "Defining models of urban traffic using the TSC tool". Proceedings of the Winter Simulation Conference, Washington, DC. U.S.A. 2001.

[23] Macedonia, M. R. and Zyda, M. J. "A taxonomy for networked virtual environments". Proceedings of the 1995 Workshop on Networked Realities, 1995.

[24] Macredie, R.; J. E. Taylor, S.; Yu, X. e Keeble R. "Virtual Reality and Simulation: An Overview". Proceedings of the 28th Conference on Winter Simulation, pgs.: 669 - 674, Coronado, California, United States, 8 a 11 de Dezembro, 1996.

[25] Manouselis, N.; Karampiperis, P. e Kosmatopoulos, E. "A multi-agent, microscopic traffic simulation architecture incorporating entities with adaptive behaviors". Proceedings of the 1st Human Centered Transportation Simulation Conference, Iowa, Novembro de 2001.

[26] Marson, F., Jung, C. and Musse, S. "Modelagem Procedural de Cidades Virtuais". Simpósio Brasileiro de Realidade Virtual, SVR2003, Ribeirão Preto, Brazil, Outubro de 2003.

[27] Wolfram Research. Mathworld – The Web's Most Extensive Mathematics Resource. Disponível em: http://mathworld.wolfram.com/. Acesso em: outubro 2004.

[28] Molofee, J. NeHe Productions. Disponível em: http://nehe.gamedev.net/. Acesso em: 05 outrubro 2004.

[29] Molofee, J. NeHe OpenGL Tutorials. Disponível em: http://nehe.gamedev.net/lesson.asp?index=01. Acesso em: agosto 2004.

[30] Morar, S. S.; Macredie, R. e Cribbin, T. "An Investigation of Visual Cues used to Create and Support Frames of Reference and Visual Search Tasks in Desktop Virtual Environments", 2002, Virtual Reality, volume 6, fascicule 3, pgs.: 140 -150.

[31] OpenGL.org. OpenGL - The Industry's Foundation for High Performance Graphics. Disponível em: http://www.opengl.org/. Acesso em: março 2005.

[32] Owen, L.E.; Zhang, Y.; Rao, L. e McHale, G. "Traffic Flow Simulation Using Corsim". Proceedings of the 2000 Winter Simulation Conference, 2000. Disponível em: http://www.informs-cs.org/wsc00papers/152.PDF.

[33] Pantel, L. e Wolf, C. L. "On the Impact of Delay on Real-Time Multiplayer Games". International Workshop on Network and Operating System Support for Digital Audio and Video: Network Issues for Video and Games, New York, NY, USA, ACM Press, pgs.: 23-29, 2002.

[34] Paruchuri, P.; Pullalarevu, A.R. e Karlapalem, K. "Multi Agent Simulation of Unorganized traffic". Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1, International Conference on Autonomous Agents, 2002 Bologna, Italy, pgs.: 176 - 183 , ISBN:1-58113-480-0.

[35] Trolltech. Qt Overview. Disponível em: http://www.trolltech.com/products/qt/. Acesso em: março 2005.

[36] Raja, D; Bowman, D.A.; Lucas, J.; North, C. "Exploring the Benefits of Immersion in Abstract Information Visualization", *8th International Immersive Projection Technology Workshop*, 8 pages, Iowa State, maio de 2004.

[37] Raja, D; Bowman, D.A. "A Method for Quantifying the Benefits of Immersion Using the CAVE", Presence Connect, junho de 2004.

[38] Reisman, R. e Ellis, S. "Augmented Reality for Air Traffic Control Towers". Proceedings of the SIGGRAPH 2003 Conference on Sketches & Applications: in conjunction with the 30th Annual Conference on Computer Graphics and Interactive Techniques, International Conference on Computer Graphics and Interactive Techniques, San Diego, California, 2003, pg.: 1.

[39] Reynolds, C. W. "Steering Behaviors For Autonomous Characters". Proceedings of Game Developers Conference 1999, Miller Freeman Game Group, San Francisco, California, pgs.: 763-782.

[40] Roberson, G.; Czerwinski, M. e V. Dantzich "Immersion in Desktop Virtual Reality", Proceedings of the 10th annual ACM Symposium on User Interface Software and Technology, 1997, UIST'97, pgs.:11-19.

[41] Roehl, B "Distributed Virtual Reality – An Overview". Proceedings of the first symposium on Virtual reality modeling language, Virtual Reality Modeling Language Symposium, 1995, San Diego, California, United States, pgs.: 39 – 43, ISBN:0-89791-818-5 .

[42] Schulze, T.; Lemessi, M. e Filippi, F. "Simulation of a night taxi-bus service for the historical center of Rome". Proceedings of the 2001 Winter Simulation Conference, WSC 2001, Arlington, VA, USA, 9 a 12 de Dezembro de 2001, pgs.: 1072-1078.

[43] Schulze, T. e Fliess, T. "Urban traffic Simulation with Psycho-physical Vehicle-following Models". Proceedings of the 29th conference on Winter simulation, Winter Simulation Conference, 1997, Atlanta, Georgia, Estados Unidos, pgs.: 1222 - 1229, ISBN: 0-7803-4278-X.

[44] Stappers, P.J.; Gaver, W. e Overbeeke, K. "Beyond the limits of real-time realism: Moving from stimulation". L. Hettinger & M. Haas, Psychological Issues in the Design and Use of Virtual and Adaptive Environments. Lawrence Erlbaum Associates, Inc., Mahwah, NJ, 2000.

[45] Schmitz, M. "Sistema de controle de tráfego urbano utilizando sistemas multi – agentes". Blumenau, 2002. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) Universidade Regional de Blumenau.

[46] Schmitz, M. e Hübner, J. F. "Uso de SMA para avaliar estratégias de decisão no controle de tráfego urbano". Seminário de Compução, 2002, Blumenau. Anais do XI Seminário de Compução. Blumenau: FURB, 2002. pgs.: 243-254. Disponível em: http://www.inf.furb.br/~jomi/pubs/2002/Schmitz-seminco2002.pdf.

[47] Seneviratne, P., Access *Traffic Simulation Model (ACTSIM)*, relatório final para o Projeto 78 do ITS-IDEA, Intelligent Transportation Systems Program, Universidade do Estado de Utah, Logan, UT, Novembro de 2001.

[48] *SMARTEST - Final Report for Publication*, ITS, University of Leeds (GB), Projeto financiado pela Comissão Européia sobre o Projeto Transport RTD Programme do 4th Framework Programme, 13 de Janeiro de 2000, Número de contrato: RO-97-SC.1059.

[49] Smith, P., "GDC 2002: Polygon Soup Programmer's Soul: 3D Pathfinding". Gamasutra, publicado em 4 de abril de 2002. Disponível em: www.gamasutra.com/features/200220405/simth_01.htm.

[50] Sun, J.; Yu, X.; Baciu, G.; Green, M., "Template-based Generation of Road Networks for Virtual City Modeling". Proceedings of the ACM symposium on Virtual reality software and technology, Virtual Reality Software and Technology, 2002, Hong Kong, China, pgs.: 33 – 40, ISBN: 1-58113-530-0.

[51] Tavares, J.; Pereira, F. B.; Machado, P. e Costa, E. "On the Influence of GVR in Vehicle Routing", 2003.

[52] Thangiah, S. R.; Shmygelska, O. e Mennel, W. "An Agent Architecture for Vehicle Routing Problems". Departamento de Ciência da Computação, Universidade Slippery Rock. SAC'2001, ACM, 2001.

[53] Wagner, C. "Developing Your Own replay System". Gamasutra, publicado em 4 de fevereiro de 2004. Disponível em: www.gamasutra.com/features/20040204/wagner_01.shtml.

[54] *World-up R5 User's Guide*, 1997-2000 por Engineering Animation, Inc.;

Bibliografia Recomendada

Boehm-Davis, D.A.; Marcus, A.; Green, P.A., Hada, H. e Wheatley, D. "The Next Revolution: Vehicle Use-Interfaces and the Global Rider/Driver Experience", CHI '03 extended abstracts on Human factors in computing systems, Conference on Human Factors in Computing Systems, Ft. Lauderdale, Florida, USA, 2003, pgs.: 708 – 709, ISBN: 1-58113-637-4.

Ben-Moshe, B.; Katz, M.; Mitchell, J. e Nir, Y., *Visibility Preserving Terrain Simplification -- An Experimental Study*, Proceedings - 18th Annual ACM Symposium on Computational Geometry, pgs.: 303-311, Junho de 2002.

Companhia de Trânsito e Transporte Urbano do Recife. CTTU on-line, site na Web. Disponível em: http://www.recife.pe.gov.br/pr/servicospublicos/cttu/. Acesso em: 19 janeiro 2005.

Morar, S.S.; Macredie, R. D. "Special Issue on "Interacting with Desktops Virtual Environments: Perception and Navigation", Virtual Reality, 19 de Maio de 2004, volume 7, pgs.:129 -130.

Mamber, U. "Introduction to algorithms – A Creative Approach". Addison- Wesley, 1989, ISBN: 0-201-12037-3.

Microsoft Corporation. Microsoft Developer's Network Site. Disponível em: http://msdn.microsoft.com/. Acesso em: dezembro 2004.

Kohl, N. Nate Kohl's Cpp Reference Site. Disponível em: http://www.cppreference.com. Acesso em: janeiro 2004.

Schildt, H. "C Completo e Total – 3ª edição revista e atualizada". 1995 McGraw-Hill, 1997 Makron Books do Brasil Editora Ltda., ISBN: 85-346-0595-5.

Sheridan, T.B. "Interaction, Imagination and Immersion Some Research Needs". Proceedings of the ACM symposium on Virtual reality software and technology, Virtual Reality Software and Technology, Seoul, Korea, 2000, pgs.: 1 – 7, ISBN: 1-58113-316-2.

Apêndice

Este apêndice contém informações complementares que podem ajudar no entendimento mais aprofundado do assunto abordado na dissertação e do funcionamento do simulador implementado.

A.1. Diagramas de classes

Nessa seção, são apresentadas as classes que compõem o simulador ITranS. Primeiramente, a estrutura geral do simulador é apresentada. Em seguida, cada classe é detalhada com seus atributos e métodos.

A.1.1. Diagrama de relações entre classes

Na Figura 51, são apresentadas as classes do sistema e os relacionamentos entre elas:

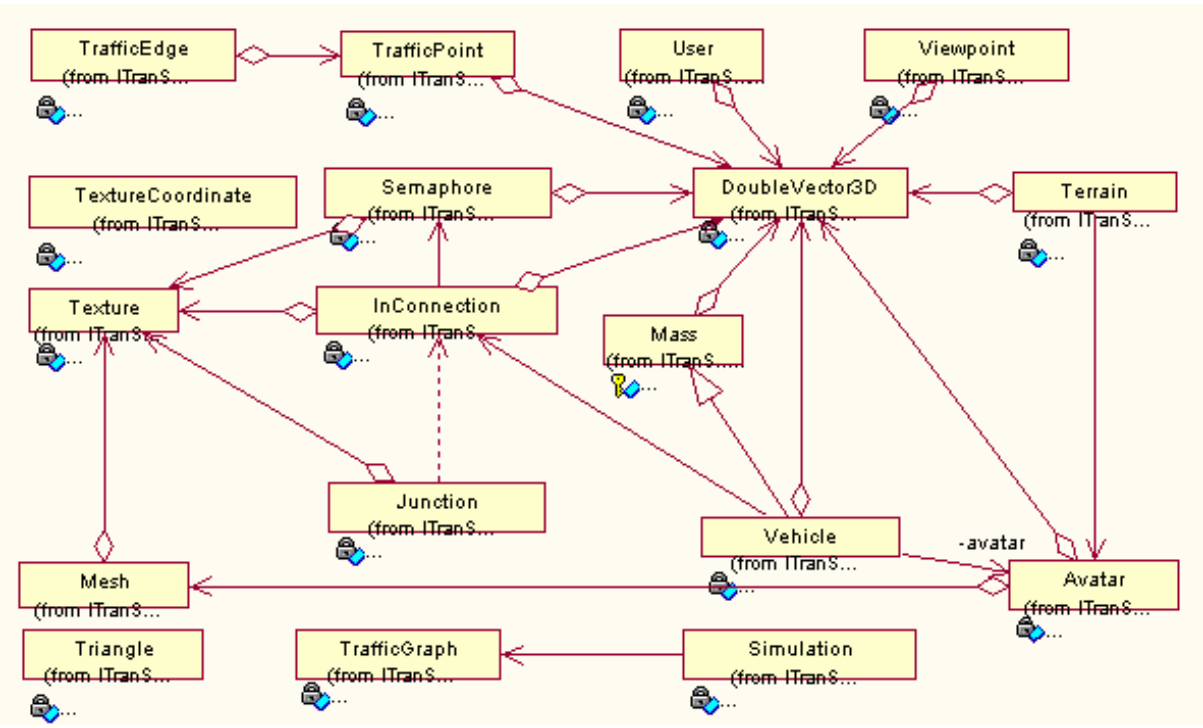


Figura 51: Principais classes do sistema e o relacionamento entre elas.

A.1.2. Diagrama detalhado de cada classe

Cada uma das classes apresentadas no diagrama de classes acima aparece com seus atributos e métodos detalhados, a seguir, na Figura 52.

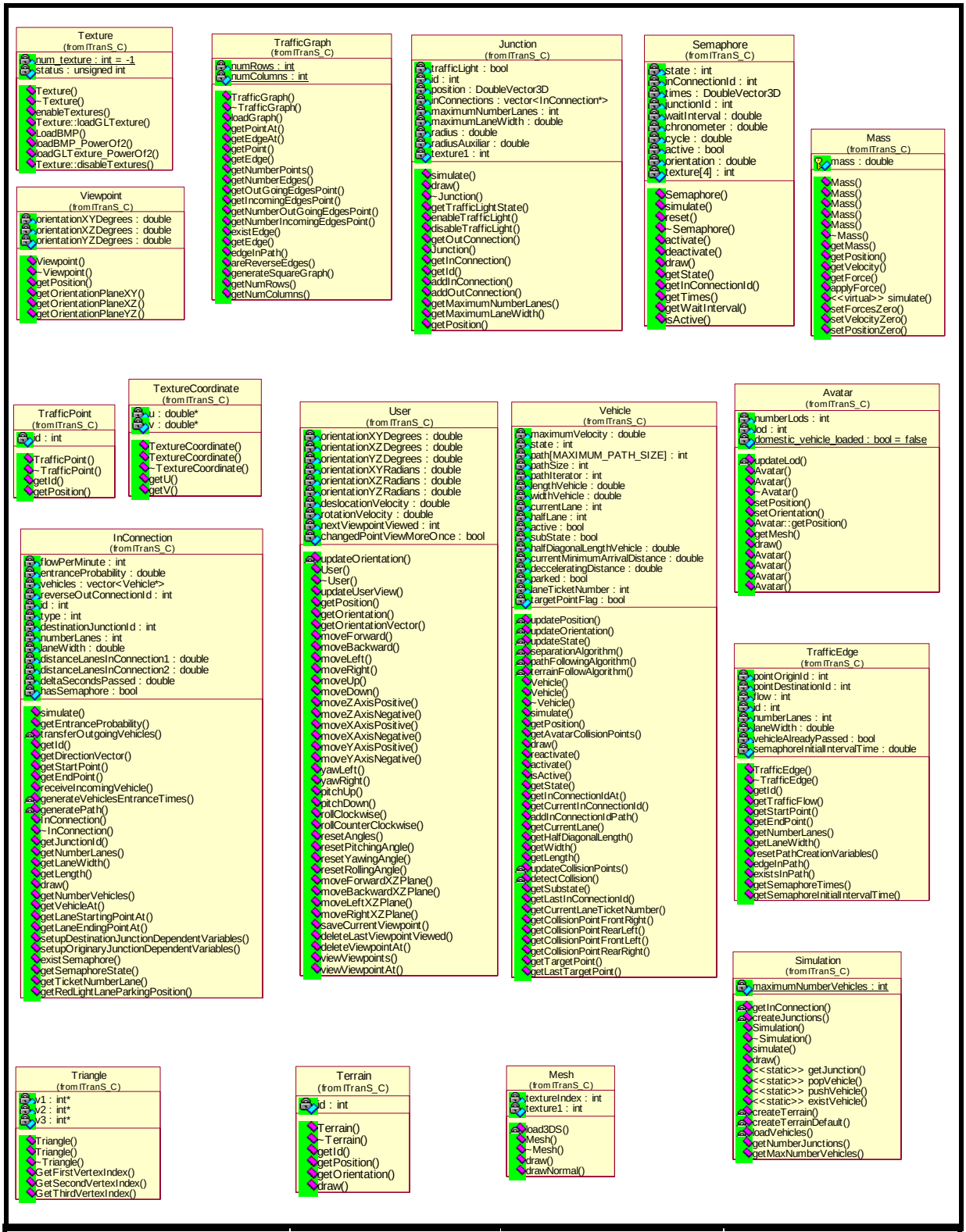


Figura 52: descrição de classes com seus métodos e atributos.

A.2. Teclas de Comando do Usuário

Abaixo, estão os quadros contendo as teclas de comando do usuário. O Quadro 6apresenta os comandos de controle de seu ponto de vista enquanto o Quadro 7 apresenta os comandos de navegação do usuário. Em ambos os quadros, a coluna da esquerda apresenta as teclas a serem pressionadas para efetuar cada comando de controle enquanto a coluna da direita descreve os seus efeitos.

A.2.1. Comandos de Controle de Pontos de Vista

Além dos comandos de navegação, o usuário também possui comandos para controle de pontos de vista, como já mencionado na seção 4.3. As teclas de ativação desses comandos estão listados no Quadro 6 abaixo.

Quadro 6: Comandos de controle de pontos de vista do usuário.

Tecla de comando	*Função de comando*
INSERT	Salva o ponto de vista atual do usário na lista de pontos de vista.
DELETE	Apaga o último ponto de vista visitado pelo usuário.
ENTER	Visualiza o último ponto de vista visitado. Se essa tecla for pressionada mais de uma vez consecutivamente sem que o usuário se movimente, o usuário visita seqüencialmente os próximos pontos de vista de forma cíclica.
Teclas de função F1 a F12	Atalhos para visualização do primeiro ao décimo segundo ponto de vista salvo respectivamente.
SHIFT + teclas de função F1 a F12	Apagam do primeiro ao décimo segundo ponto de vista salvo respectivamente.

A.2.2. Comandos de Navegação

Como mencionado na seção 4.4, o movimento do usuário está dividido em basicamente três grupos. As teclas para ativação dos comandos de controle desse três tipos de movimento podem ser vistas no Quadro 7 a seguir.

Quadro 7: Comandos de navegação do usuário.

Tecla de comando	*Função de comando*
Seta para cima (↑)	Move o avatar do usuário para a direção para a qual ele está apontando.
Seta para baixo (↓)	Move o avatar do usuário para a direção oposta a qual ele está apontando.
Seta para esquerda (←)	Move o avatar do usuário para a direção esquerda, perpendicularmente à posição para a qual ele está apontando.
Seta para direita (→)	Move o avatar do usuário para a direção direita, perpendicularmente à posição para a qual ele está apontando.
Home	Move o avatar do usuário para cima, ou seja, na direção do vetor normal de seu avatar.
End	Move o avatar do usuário para baixo, ou seja, na direção oposta do vetor normal de seu avatar.
Tecla de espaço	O ângulo de arfagem, responsável por girar o avatar do usuário para baixo ou para cima, é reiniciado para zero. Com isso o usuário aponta para uma direção paralela ao plano XZ.
CTRL + ↑	Gira o avatar do usuário para cima em torno de seu eixo.
CTRL + ↓	Gira o avatar do usuário para baixo em torno de seu eixo.
CTRL + ←	Gira o avatar do usuário no sentido anti-horário em torno de seu eixo.
CTRL + →	Gira o avatar do usuário no sentido horário em torno de seu eixo.
SHIFT + ↑	Move o avatar do usuário para a direção para a qual ele está apontando, mas mantendo-o num plano paralelo ao plano XZ na altura do seu avatar.
SHIFT + ↓	Move o avatar do usuário para a direção oposta à qual ele está apontando, mas mantendo-o num plano paralelo ao plano XZ na altura do seu avatar.
SHIFT + ←	Move o avatar do usuário para a direção esquerda, perpendicularmente à posição para a qual ele está apontando, mas mantendo-o num plano paralelo ao plano XZ na altura do seu avatar.
SHIFT + →	Move o avatar do usuário para a direção direita, perpendicularmente à posição para a qual ele está apontando, mas mantendo-o num plano paralelo ao plano XZ na altura do seu avatar.
SHIFT + tecla de espaço	O ângulo de guinada, responsável por girar o avatar do usuário para a esquerda e ou para a direita, é reiniciado para zero. Com isso o usuário aponta para uma direção paralela ao plano YZ.

A.3. Algoritmos

Abaixo estão descritos os principais algoritmos utilizados nessa dissertação. Eles estão descritos em um pseudocódigo sem necessidade de conhecimento da linguagem de programação.

A.3.1. Algoritmo de PathFollowing dos Veículos

```
- Calcula-se o vetor de velocidade desejada do veículo. Ela equivale
à normal do vetor que vai da posição do veículo ao seu ponto de
destino atual.
- Se a distância entre o vetor da velocidade atual e o vetor de
velocidade desejada for maior que metade da largura de uma faixa mais
metade da largura do veículo, então: {
  - Calcula-se uma força de atração para a faixa cuja normal é o
  vetor que vai da velocidade atual à velocidade desejada (ver Figura
  16).
  - Aplica-se essa força ao veículo.
}
```

A.3.2. Algoritmo de Detecção de Colisão entre Veículos

```
- Percorrem-se todos os veículos na conexão: {
  - Se o veículo estiver na mesma faixa que o veículo atual, então: {
    - Calcula-se o comprimento diagonal do veículo.
    - Se a distância entre um dos pontos de colisão do veículo e
    todos os outros pontos de colisão do veículo atual for menor que
    o comprimento diagonal do veículo, então, houve colisão (ver
    Figura 17).
  }
}
```

A.3.3. Algoritmo de Separação dos Veículos

```
- Percorrem-se todos os veículos na conexão do veículo atual: {
  - Se o veículo estiver na mesma faixa do veículo atual e seu número
  de ticket for menor que o do veículo atual, então: {
    - Calcula-se a distância entre os dois veículos.
    - Se a distância for menor que a distância mínima de separação
    para veículos numa mesma faixa, então: {
      - Se o veículo estiver perseguindo o ponto final da faixa e se
      estiver um pouco afastado (10m ou mais) dos pontos final e
      inicial da faixa, então, desacelera-se o veículo de forma
      proporcional à razão entre a sua distância e a distância mínima
      permitida.
      - Calcula-se uma força de repulsão na distância oposta à da
      conexão atual cujo valor cresce de forma inversamente
      proporcional à razão entre a sua distância e a distância mínima
      permitida(ver Figura 18).
      - Aplica-se essa força ao veículo.
    }
  }
  Se o veículo não estiver na mesma faixa do veículo atual, então: {
    - Calcula-se a distância entre os dois veículos.
    - Se os veículos estiverem quase paralelos um ao outro, apenas
    com uma diferença de 1m ou menos entre suas extremidades, então:
    {
      - Se a distância entre eles for menor que 1/8 da largura da
      faixa, então: {
        - Calcula-se uma força de repulsão ao veículo cuja normal é o
        vetor que vai do ponto final da faixa do veículo do qual se
        aproximou ao ponto final da faixa do veículo(ver Figura 19).
        - Aplica-se essa força ao veículo.
      }
    }
  }
}
```

A.3.4. Algoritmo para Geração de Caminhos dos Veículos

```
- Desmarca todas as arestas pelas quais o algoritmo passou durante a
última criação de caminho para veículo.

- Obtém o identificador da primeira conexão a ser inserida no
caminho. Ela é a própria conexão que requisitou a sua criação.

- Enquanto não se tiver inserido uma conexão de saída ao caminho: {

  - Defina um valor aleatório de escolha para o veículo entre 0 e 1.

  - Obtenha a lista de possíveis conexões a serem seguidas.

  - Se não existir nenhuma ou se só existir uma e ela for a conexão
  que dá o sentido inverso da conexão atual, então, chegou-se a uma
  conexão de saída e o caminho está completo.
  Senão: {

    - Calcula-se a probabilidade do veículo seguir cada uma das
    conexões, de acordo com seus valores de fluxo e o do fluxo total
    da junção.

    - Calcula-se intervalos de probabilidade complementares para cada
    uma delas (ver Figura 12).

    - Percorre-se a lista de conexões: {

      - Se o intervalo de probabilidade da conexão observada na lista
      abranger o valor escolhido no começo, então: {

        - Se a conexão não existe no caminho e não é a conexão inversa
        da última conexão adicionada ao caminho, então, adiciona-se
        ela ao caminho e marca-se ela como já estando no caminho.
        Senão: {

          - Enquanto todas as possibilidades de conexões não tiverem
          sido avaliadas: {

            - Obtenha a próxima conexão disponível.

            - Se ela não existe no caminho e não é a conexão inversa
            da última conexão adicionada ao caminho, então, adicione-a
            ao caminho e marque ela como já estando no caminho.
            Senão, obtenha a próxima conexão.

            }

          - Se todas as possíveis próximas conexões já pertenciam ao
          caminho, adicione ao caminho aquela inicialmente escolhida.

        }

      }

    }

  }

}
```

Dissertação de Mestrado apresentada por **Paulo Gonçalves de Barros** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título **"Itrans – Simulador de Trânsito 3D"**, orientada pela **Profa. Judtih Kelner** e aprovada pela Banca Examinadora formada pelos professores:

Prof. Fernando da Fonseca de Souza
Centro de Informática / UFPE

Prof. Claudio Kirner
Faculdade de Ciências e Tecnologia da Informação / UNIMEP

Prof. Verônica Teichrieb
Escola Politécnica / UPE

Profa. Judith Kelner
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 1 de março de 2005.

Prof. JAELSON FREIRE BRELAZ DE CASTRO
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.