UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

EDITOR GRÁFICO DE RUAS PARA O SISTEMA DE CONTROLE DE TRÁFEGO DE AUTOMÓVEIS EM UMA MALHA RODOVIÁRIA URBANA

GEFERSON BERTHOLDI

BLUMENAU
2004

2004/2-18

GEFERSON BERTHOLDI

EDITOR GRÁFICO DE RUAS PARA O SISTEMA DE CONTROLE DE TRÁFEGO DE AUTOMÓVEIS EM UMA MALHA RODOVIÁRIA URBANA

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

Prof. José Roque Voltolini da Silva – Orientador

BLUMENAU
2004

2004/2-18

EDITOR GRÁFICO DE RUAS PARA O SISTEMA DE CONTROLE DE TRÁFEGO DE AUTOMÓVEIS EM UMA MALHA RODOVIÁRIA URBANA

Por

GEFERSON BERTHOLDI

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

Presidente:	Prof. José Roque Voltolini da Silva – Orientador, FURB
Membro:	Prof. Dr. Paulo César Rodacki Gomes, FURB
Membro:	Prof. Dr. Mauro Marcelo Mattos, FURB

Blumenau, 15 de Fevereiro de 2005

Dedico este trabalho a todos os amigos, especialmente aqueles que me ajudaram diretamente na realização deste.

“Quem quer fazer alguma coisa encontra um meio. Quem não quer fazer nada, encontra uma desculpa.”

Aforisma Árabe

AGRADECIMENTOS

À Deus, pelo seu imenso amor e graça.

À minha família, que mesmo longe, sempre esteve presente.

Aos meus amigos, pelos empurrões e cobranças.

Ao meu orientador, José Roque Voltolini da Silva, por ter acreditado na conclusão deste trabalho.

RESUMO

Este trabalho apresenta a especificação e implementação de um editor gráfico de malhas rodoviárias urbanas. A malha é desenhada através de retas, observando-se os sentidos e as intersecções. O resultado final é a visualização da malha rodoviária urbana em um monitor de vídeo. À partir da edição de uma malha rodoviária é gravado um arquivo texto com as informações da estrutura viária (ruas), o qual é usado pelo software desenvolvido por Josemar J. Freire (Freire, 2004). Para especificar o protótipo foram utilizadas definições da área matemática (geometria analítica e trigonometria) e de computação gráfica. O ambiente de programação Delphi e as bibliotecas gráficas Opengl foram utilizadas para confecção do protótipo.

Palavras chaves: Editor Gráfico; Tráfego de Veículos; OpenGl.

ABSTRACT

This work shows the specification and implementation of a graphic editor of urban road system. The system is designed through straight lines, observing the directions and the intersections. The final result is the visualization of the urban road system in a screen. From the edition of a road system will be recorded a text file with the information of the way (street) structure, which will be used by the software developed by Jocemar J. Freire (2004). To specify the prototype, it was used definitions of Math area (analytic geometry and trigonometry) and graphical computation. The Delphi programming surrounding and the Opengl graphic library were used to the prototype's making.

Key words: Graphic Editor; Vehicle's Traffic; OpenGl.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

As facilidades atuais para aquisição de um automóvel, a falta de planejamento dos sistemas viários da maioria das cidades brasileiras e a ineficiência de semáforos mal ajustados colaboraram para o surgimento dos sistemas de controle de tráfego, que têm como objetivo principal customizar o sistema viário existente, abolindo semáforos e utilizando sensores nos veículos, de forma que o trânsito possa se moldar dinamicamente, evitando assim os engarrafamentos (SINCMOBIL, 2003).

No primeiro semestre de 2004 foi desenvolvido um sistema que simula o tráfego de veículos em uma área urbana por Jocemar J. Freire (FREIRE, 2004). O software possibilita escolher uma malha rodoviária e a partir dela, definir-se a velocidade e a quantidade de veículos que trafegarão, auxiliando na identificação de possíveis problemas no tráfego, como por exemplo o congestionamento viário. No sistema, os veículos são conduzidos de forma automática (sem condutor). Na malha projetada foram abolidos semáforos.

Uma das principais dificuldades do software, que inclusive foi sugerida como continuação do trabalho desenvolvido por Freire, é a especificação da malha rodoviária. Esta especificação é feita digitando-se as informações através de um editor genérico para texto, sendo o arquivo armazenado em formato *txt.* Todas as informações inerentes são calculadas manualmente, tornando o processo trabalhoso e aumentando a probabilidade de inconsistências na formatação do *layout* previamente definido para o Sistema de Controle de Tráfego de Veículos.

Visto o acima descrito, este trabalho descreve o desenvolvimento de um editor gráfico para auxiliar na modelagem de uma malha viária para o sitema proposto em Freire (2004).

O software editor foi desenvolvido com o intuito de facilitar a modelagem da malha rodoviária urbana, fazendo uso do *mouse* como principal ferramenta de interação do usuário com o software. Quanto maior a similaridade do sistema com o ambiente Windows, menos tempo de treinamento será investido ao usuário para conhecer e utilizar o editor adequadamente, tornando-se simples e rápida qualquer manutenção em uma malha viária, possibilitando desta forma a simulação de trânsitos alternativos.

O Sistema faz uso da biblioteca gráfica OpenGl para visualização dos trechos e do mapa como um todo.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é a construção de um editor gráfico 2D que crie uma malha viária graficamente, e após, armazená-la em um arquivo texto compatível para o sistema de controle de trânsito desenvolvido por Freire.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está estruturado em quatro capítulos. O primeiro capítulo trata da introdução do trabalho. O segundo capítulo trata da fundamentação teórica. O terceiro capítulo trata do desenvolvimento do trabalho. O quarto e último finaliza o trabalho com a conclusão.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão abordados assuntos relacionados com o foco do trabalho proposto, enfatizando aspectos utilizados para definição e implementação. Os assuntos a serem abordados são: controle de tráfego viário, simulação do controle de tráfego de automóveis em uma malha rodoviária urbana no software (protótipo) desenvolvido por Jocemar J. Freire (FREIRE, 2004), editores gráficos, computação gráfica, biblioteca gráfica OpenGl e considerações sobre geometria analítica e trigonometria.

2.1 CONTROLE DE TRÁFEGO VIÁRIO

Segundo Abramcet (2000), pode-se definir Controle de Tráfego Viário como a supervisão do movimento de veículos e pessoas, com o objetivo de garantir eficiência e segurança. Uma rodovia pode ser considerada eficiente quando fornece aos seus usuários a possibilidade de se movimentar ao menor custo possível. É vista como segura ao reduzir ou eliminar os acidentes na sua totalidade.

Problemas de tráfego são tão antigos quanto o Império Romano. A causa básica, como nos dias atuais, era o pobre planejamento das cidades, com vias que convergiam aleatoriamente de diversos quarteirões para um ponto central. Na Roma antiga, Júlio César proibiu o tráfego sobre rodas durante o dia, medida que foi gradualmente estendida para outras províncias como uma forma de limitar o acesso e garantir um mínimo de circulação. Em 1500, Leonardo da Vinci, prevendo uma solução revolucionária para os problemas de tráfego, sugeriu separar o trânsito de veículos e pedestres pela criação de rotas em dois diferentes níveis. Na Europa do século 17, os congestionamentos levaram à proibição do estacionamento em certas áreas e à criação de vias de mão única. O advento dos trens causou problemas crescentes aos sistemas de controle de tráfego, visto que em certos momentos os fluxos eram subitamente maiores, particularmente nos terminais e nas entradas das cidades. O automóvel, que inicialmente ampliou sua velocidade e posteriormente sua quantidade, rapidamente criou uma nova situação que veio a se caracterizar como um dos principais problemas da sociedade industrial do século 20, o tráfego (ABRAMCET, 2000).

De acordo com o Departamento Nacional de Trânsito (DENATRAN, 1987 apud SCHREIDER JUNIOR, 2001, p. 6), aproximadamente 70% dos acidentes ocorrem em interseções. Portanto, é neste local que naturalmente o “conflito de interesses” apresenta-se

mais claramente. Desta forma, certos instrumentos de controle do tráfego tornam-se necessários para a regulamentação do direito de passagem de pedestre e veículos. Atualmente usa-se os tradicionais semáforos, cujo principal problema é a temporização dos mesmos. Diversos projetos têm sido criados para propor soluções para a temporização de semáforos. Entre eles, cita-se o projeto relatado em SincMobil (2003), desenvolvido pela Universidade Federal de Santa Catarina, que visa implementar o controle de tráfego de veículos automotores através de sensores em tempo real, solução esta que dinamizaria o fluxo de veículos nas interseções.

2.2 SIMULAÇÃO DO CONTROLE DE TRÁFEGO DE AUTOMÓVEIS EM UMA MALHA ROVIÁRIA URBANA

O Trabalho desenvolvido por Freire em 2004, propõe simular e identificar em uma determinada malha rodoviária situações de engarrafamento (Figura 1). Para tanto, o mapa viário é armazenado em um arquivo texto (quadro 1). Estas informações são digitadas manualmente através de um editor de texto. Todos os cálculos envolvidos são feitos manualmente, aumentando a possibilidade de erros na especificação e demonstração da malha viária no simulador do tráfego de automóveis.

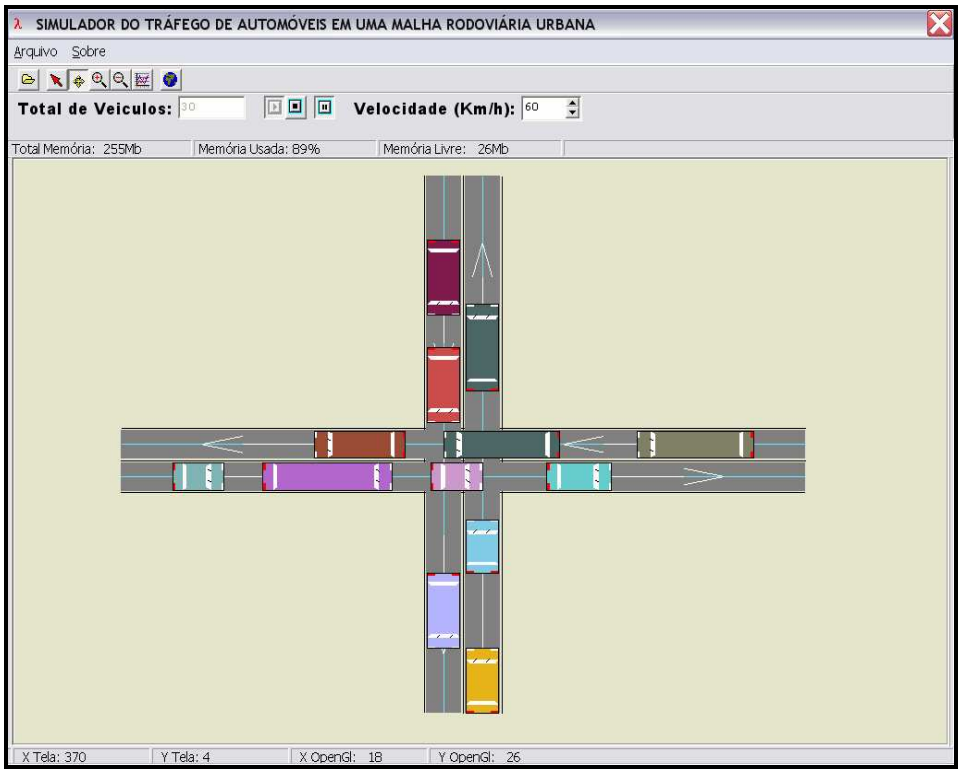


Figura 1- Protótipo Simulador do controle de tráfego de automóveis.

```
00 -03 25 -03 3 0 0 0 0 1 01
25 -03 25 -25 3 0 0 0 0 1 02
28 -25 28 -03 3 0 0 0 0 1 03
28 000 28 025 3 0 0 0 0 1 04
25 000 00 000 3 0 0 0 0 1 05
25 025 25 000 3 0 0 0 0 1 06
28 -03 53 -03 3 0 0 0 0 1 07
53 000 28 000 3 0 0 0 0 1 08
25 -03 28 -03 3 0 0 0 1 1 09
25 000 25 -03 3 0 0 0 1 1 10
28 -03 28 000 3 0 0 0 1 1 11
```

Quadro 1 – Representação textual do mapa rodoviário

Quando o arquivo é aberto, estas informações são armazenadas em uma estrutura do tipo lista e utilizadas nas funções matemáticas e de desenho durante a execução do programa desenvolvido por Jocemar Freire.

Os carros movimentam-se ordenadamente, utilizando o conceito de processos concorrentes.A comunicação entre os processos é feita através do uso de construções do tipo semáforos, existentes em linguagens de programação. É possível ainda determinar a velocidade e a quantidade de carros que se movimentarão em uma malha, simulando cenários do dia-a-dia, o que pode ser usado em tomadas de decisões, como por exemplo a construção de novas vias ou mudança do trânsito.

2.3 EDITORES GRÁFICOS

A ironia do termo “interface de usuário” é o fato de ser centrado na tecnologia: o “usuário” é especificado explicitamente, enquanto o computador é implícito (é claro que o termo não foi inventado pelos computadores, mas por pessoas que raciocinavam como se fossem um computador). Para analisar o surgimento e a evolução desse conceito é conveniente pensar na “interface do computador” com o seu usuário e o mundo. Esta perspectiva permite estender o termo “interface de usuário” para o período anterior à existência do termo e também para o futuro, no qual o computador terá uma interface capaz de abranger não apenas usuários individuais, mas grupos de usuários e até organizações como um todo (MAYER & BUNGE, 1997, p. 15). Técnicas para construção de interface de usuário são aplicadas exaustivamente em software do tipo editores de textos ou gráficos.

Os editores gráficos adotam como características fundamentais:

a) ser consistente: possuir consistência na geração e visualização dos dados;

b) fornecer *feedback*: em nível de hardware, emite sons em operações não corretas. em nível sequêncial, quando selecionado o objeto, menu ou ícone os mesmo são destacados;
c) minimizar possibilidades de erros: desabilita comandos que não podem ser usados para evitar o erro;
d) delimitar área de desenho: impede que o usuário utilize outra área, senão a pré-definida no software.

Nas “interfaces de usuário”, o dispositivo *mouse* é essencial, sendo sua principal função a de apontar, à qual serve na maioria das vezes como preparação para uma das seguintes operações: pressionar, clicar, clique duplo e arrastar.

À seguir serão descritas todas as ações do *mouse*.

2.3.1 Apontar

O ato de apontar (*point*, em inglês) com o *mouse* significa que o usuário deve mover o mesmo até que o ponteiro esteja na posição desejada. Na realidade o ponteiro do *mouse* (também chamado de cursor) é um desenho que ocupa uma certa região da tela.

O ponto exato que o ponteiro do *mouse* define é chamado de “*hot spot*”; por exemplo, o *hot spot* do ponteiro de *mouse* na forma de seta costuma ser a ponta da seta, enquanto o *hot spot* do ponteiro na forma de cruz costuma ser o seu ponto central.

2.3.1.1 Pressionar

O ato de pressionar (*press*, em inglês) o *mouse* significa que um dos seus botões é abaixado e mantido na sua posição inferior, sem que o *mouse* seja movimentado. Esta operação é usada quase sempre para identificar um objeto a ser selecionado.

2.3.1.2 Clicar

O ato de clicar o *mouse* (chamado também de dar um clique simples, ou de *click*, em inglês) consiste em apertar um botão do *mouse* e soltá-lo logo a seguir, sem que o *mouse* seja movimentado. Esta operação é a mais comum executada com o *mouse*, e por isso possui diversos significados: por exemplo, ela é usada para selecionar um item, operar um controle, ativar um controle, abrir um menu ou ativar uma janela, entre outros.

2.3.1.3 Clique Duplo

A operação clique duplo (*double click*, em inglês) consiste em dois cliques simples executados dentro de um intervalo pequeno de tempo[1]. O clique duplo com o botão direito é uma operação de uso incomum. Esta operação é usada na maioria das vezes como um atalho para operações comuns: enquanto o primeiro clique seleciona o objeto desejado, o clique duplo (confirmado pelo segundo clique) aplica a esse objeto uma ação padrão. Por exemplo, um clique duplo sobre um ícone pode ter as seguintes funções: se representa um diretório, este é aberto; se representa um programa, este é executado.

É importante notar que o clique duplo representa, do ponto de vista da aplicação, duas operações: o clique simples, correspondente ao primeiro pressionamento do botão do *mouse*, e o clique duplo (correspondente ao segundo pressionamento do botão do *mouse*) que aciona o botão de comando padrão. Esta operação é difícil de ser executada por alguns usuários. Por isso, ela nunca deve ser usada como a única maneira de acionar uma determinada parte da funcionalidade da aplicação.

2.3.1.4 Arrastar

A operação de arrastar (*drag*, em inglês) consiste em pressionar um botão do *mouse* e mantê-lo pressionado enquanto o *mouse* é movimentado. Esta operação é usada na maioria das vezes para identificar uma sequência de objetos ou para mover e/ou redimensionar objetos.

2.4 COMPUTAÇÃO GRÁFICA

> A Computação Gráfica é parte da Ciência da Computação e área de estudo de alguns aspectos da comunicação entre o homem e o computador. O aspecto principal abordado pela Computação Gráfica é o da comunicação visual no sentido máquina-homem, através da síntese de imagens em dispositivos de saída apropriados (BANON, 1989, p. 1).

Para a descrição e representação precisa dos atributos geométricos das primitivas gráficas, é necessário fixar-se um sistema de coordenadas. Diferentes aplicações podem requisitar diferentes sistemas de coordenadas por serem mais naturais na expressão do

[1] O tempo máximo disponível pode, em geral, ser configurado. No Windows, a opção *mouse* do Painel de Controle inclui esta possibilidade.

fenômeno em estudo. Um exemplo de sistema de uso corrente é o sistema de coordenadas cartesianas (PERSIANO; OLIVEIRA, 1989. p. 103). A fig. 2 ilustra a representação de um ponto no sitema cartesiano.

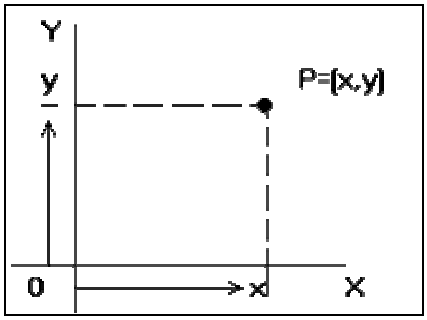
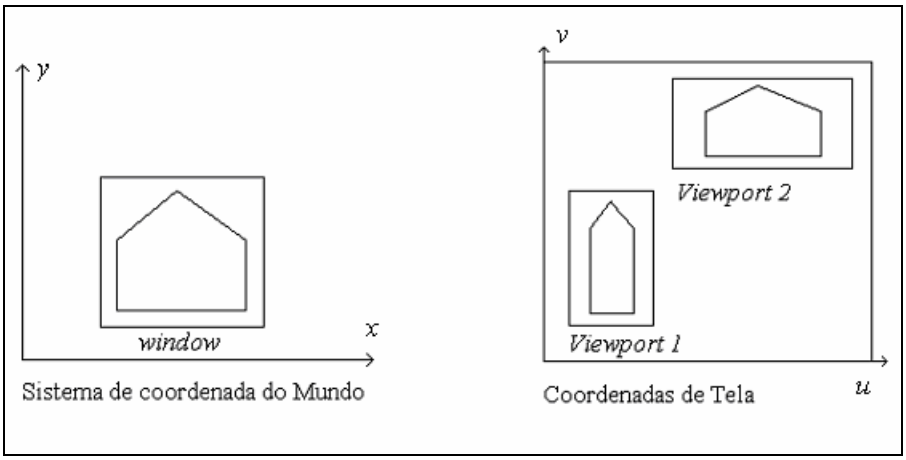


Figura 2 - Representação do sistema cartesiano

Para que as primitivas gráficas de saída com atributos geométricos expressos em coordenadas universais possam ser representadas na tela virtual, estabelece-se uma transformação linear entre os dois espaços. Uma maneira conveniente de caracterizar-se uma transformação com essa finalidade é definir-se uma área retangular no sistema de coordenadas universais e especificar por um segundo retângulo na tela virtual onde a primeira área é mapeada (PERSIANO; OLIVEIRA, 1989. p. 105).

A região retangular especificada no sistema de coordenadas do mundo é chamada de janela (*window*). A representação da janela no dispositivo de saída é chamada de *viewport*. Esta transformação representa na *viewport* a imagem gráfica das primitivas existentes dentro da janela do sistema de coordenadas do mundo (FOLEY et al, 1990, p. 210). A fig.3 representa o sistema de coordenadas e suas transformações.



Fonte: adaptado de Foley et al (1990, p. 211).

Figura 3 - Sistemas de coordenadas e suas transformações

2.5 BIBLIOTECA GRÁFICA OPENGL

OpenGL é uma biblioteca gráfica padrão de alto desempenho, independente do sistema de janelas e do hardware gráfico utilizado, voltada para manipulação e renderização de objetos tridimensionais, além de manipular também gráficos bidimensionais e imagens. A biblioteca OpenGL foi desenvolvida pela Silicon Graphics Inc., tendo sido padronizada em 1991 por um consórcio de empresas, a OpenGL *Architetural Review Board* (WOO; NEIDER; DAVIS, 1997, p. 5).

A biblioteca funciona como uma máquina de estados. Todos os estados ou modos habilitados nas aplicações têm efeito enquanto os mesmos estiverem ligados ou forem modificados. Todas as características do OpenGL são configuráveis através de variáveis, tais como cores, posições, característica de luzes e propriedades de materiais (WOO; NEIDER; DAVIS, 1997, p. 10). Por exemplo, a cor corrente é uma variável de estado que pode ser definida como branco. Todos os objetos, então, são desenhados com a cor branca, até o momento em que outra cor corrente é especificada. Cada variável de estado possui um valor inicial (*default*) que pode ser alterado.

Segundo Wangenheim (2004) os recursos gráficos disponíveis pelo OpenGl são:

a) modos de desenho de pontos;
b) ajustes de largura de linhas;
c) aplicação de transparência;
d) ativação e desativação de serrilhamento;
e) mapeamento de superfície com textura;
f) seleção de janela (*window*) de desenho;
g) manipulação de fontes, tipos de iluminação e sombreamento;
h) transformações de sistemas de coordenadas;
i) transformações em perspectivas;
j) combinações de imagens.

2.6 CONSIDERAÇÕES SOBRE GEOMETRIA ANALÍTICA E TRIGONOMETRIA

Aqui serão descritas algumas funções da geometria analítica e trigonometria para resolver problemas relacionados ao desenho de uma malha rodoviária urbana.

2.6.1 Geometria Analítica

Segundo Domingues (2003), a Geometria, como ciência dedutiva, foi criada pelos gregos. Mas, apesar do seu brilhantismo faltava operacionalidade à geometria grega. E isto só iria ser conseguido mediante a Álgebra como princípio unificador. Os gregos, porém, não eram muito bons em álgebra. Mais do que isso, somente no século XVII a álgebra estaria razoavelmente aparelhada para uma fusão criativa com a geometria.

A Geometria Analítica de Descartes apareceu em 1637 no pequeno texto chamado A Geometria como um dos três apêndices do Discurso do método, obra considerada o marco inicial da filosofia moderna. Nela, em resumo, Descartes defende o método matemático como modelo para a aquisição de conhecimentos em todos os campos (DOMINGUES, 2003).

Este capítulo abordará principalmente a geometria analítica plana e suas fórmulas.

2.6.1.1 Distância Entre Dois Pontos no Plano

A distância entre dois pontos, sendo estes pontos os pares ordenados A(Xa, Ya) e B(Xb, Yb), situados em um plano cartesiano, é dada pela fórmula: $d_{ab} = \sqrt{(X_b - X_a)^2 + (Y_b - Y_a)^2}$, onde d_{ab} é a distância entre o ponto 'A' até o ponto 'B' (SILVA; BARRETO FILHO, 1990, p. 268).

A fig. 4, mostra a dedução da fórmula para a distância entre dois pontos.

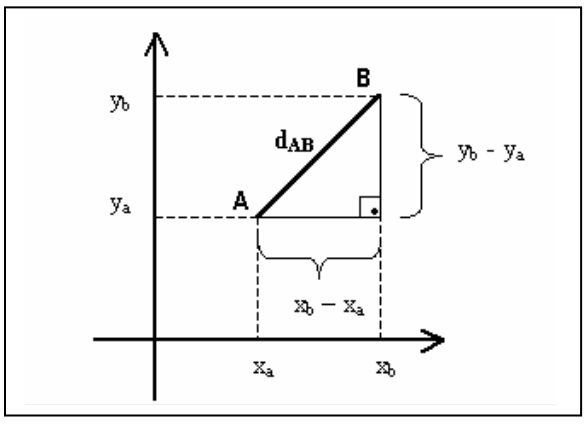


Figura 4 - Representação gráfica da distância entre dois pontos.

2.6.1.2 Ponto Médio de um Segmento

Sejam os pontos A, B e um ponto M que divide AB ao meio; pode-se dizer que as coordenadas x_M e y_M do ponto médio M são obtidos por meio da média aritmética das abscissas e ordenadas, respectivamente, dos pontos dos quais M é ponto médio (fig. 5) (SILVA; BARRETO FILHO, 1990, p. 270).

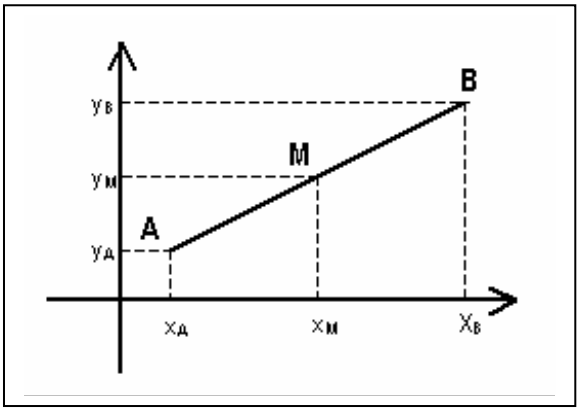


Figura 5 - Representação do Ponto Médio.

2.6.1.3 Retas Paralelas e Perpendiculares

A condição necessária e suficiente para que duas retas r e s não-verticais sejam paralelas entre si é que tenham o mesmo coeficiente angular **(r // s ⇔ mr = ms)** (SILVA; BARRETO FILHO, 1990, p. 271).

A condição necessária e suficiente para que duas retas r e s não-verticais sejam perpendiculares entre si é que o produto dos seus coeficientes angulares seja igual a –1 **(r ⊥ s $m_r . m_s$ = -1)** (SILVA; BARRETO FILHO, 1990, p. 271).

2.6.1.4 Coeficiente Angular de Uma Reta

Num sistema ortogonal, a reta r, não perpendicular ao eixo x, tem como coeficiente angular um número real m dado pela tangente do ângulo (fig. 6), onde a tangente do ângulo é dada pela formula: $m = tg\alpha = \frac{Yb - Ya}{Xb - Xa}$ (SILVA; BARRETO FILHO, 1990, p. 279).

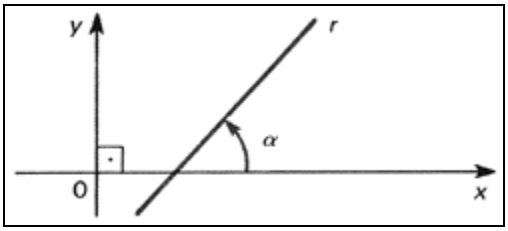


Figura 6 - Representação do coeficiente angular da reta.

2.6.1.5 Interseção de Duas Retas

Segundo Queiroz (2004, p. 24), dois segmentos r e s, formados pelos pontos p_1p_2 e p_3p_4 (fig. 7), respectivamente, verificar se eles se interceptam consiste em testar se os pontos p_1 e p_2 estão de lados opostos do segmento formado por p_3p_4 e também se p_3 e p_4 estão de lados opostos do segmento formado por p_1p_2. Este problema se conecta com o problema da área de triângulo, pois, determinar se p_3 está do lado oposto de p_4 em relação ao segmento r, consiste em avaliar o sinal da área dos triângulos formados por $p_1p_2p_3$ e $p_1p_2p_4$. Se os sinais forem contrários, significa que os pontos estão de lados opostos. Se o mesmo for verdadeiro para os triângulos $p_3p_4p_1$ e $p_3p_4p_2$, então, com certeza pode-se afirmar que as retas que passam pelos segmentos a e b se interceptam em algum ponto, embora não se possa afirmar ainda que os segmentos têm interseção. Obtêm-se $P(x_p,y_p)$, resolvendo o sistema {r,s} composto pelas equações: $r : a_1x + b_1y + c_1 = 0$ e $s : a_2x + b_2y + c_2 = 0$.

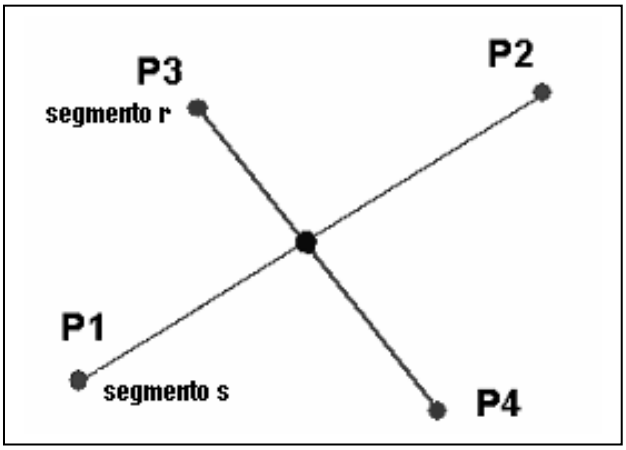


Figura 7 – Segmentos que se interceptam

2.6.1.6 Localização de Pontos em Relação a Polígonos

Segundo Queiroz (2004, p. 16), uma solução para este problema consiste em considerar uma semi-reta L partindo de c e determinar seus pontos de intersecção com a linha poligonal. Se c coincidir com um destes pontos de intersecção, concluí-se que ele está na fronteira de P, senão basta contar quantas vezes a semi-reta atravessa a poligonal. No infinito, L encontra-se na região exterior. Logo, se o número de cruzamentos for ímpar, o ponto c é interior; caso contrário, c é exterior.

Para obter um algoritmo baseado na descrição acima, é necessário ter cuidado com alguns detalhes. Já que L é arbitrária, opta-se por tomar a semi-reta horizontal $L = \{(x_0,y_0) + t\ (1,0) \mid t \geq 0\}$, para simplificar o cálculo dos pontos de interseção. Os pontos de interseção de L com a linha poligonal são obtidos calculando o ponto de intersecção de L com cada lado do polígono. No caso da fig. 8, por exemplo, o ponto a não deve ser contado como ponto de intersecção, enquanto b deve.

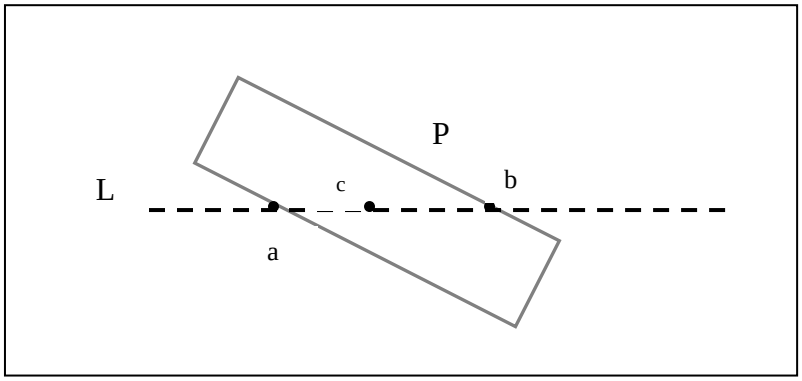


Figura 8 - Problema PONTO EM POLÍGONO.

3 DESENVOLVIMENTO DO PROTÓTIPO

Neste capítulo será detalhado o desenvolvimento do sistema proposto, sendo que as seções abaixo descrevem os requisitos do sistema, especificação do sistema, implementação do sistema, operacionalidade da implementação e resultados e discussão.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O sistema deverá satisfazer os seguintes requisitos funcionais (RF) e requisitos não funcionais (RNF):

a) permitir a modelagem da malha rodoviária (RF);
b) permitir a criação, modificação e exclusão de um trecho (RF);
c) permitir a junção de dois trechos (RF);
d) permitir a divisão de um trecho (RF);
e) permitir fazer trechos paralelos (RF);
f) permitir a opção de mudança de tamanho da malha (RF);
g) permitir a opção de mover mapa no mundo (RF);
h) apresentar as informações de forma perceptível (facilidade de identificação dos objetos) pelo usuário do sistema (RNF).

3.2 ESPECIFICAÇÃO

Neste capítulo será identificado o domínio da aplicação representando objetos e ações do domínio em questão, no caso, a modelagem de um editor gráfico de ruas para o sistema de controle de tráfego de automóveis em uma malha rodoviária urbana.

Foi utilizado um diagrama de atividades, confeccionado através da ferramenta *Enterprise Architect* (STRALEY, 2002), para representar de forma macro as funcionalidades do protótipo, tornando possível ao usuário visualizar todas as funções do software.

Para especificação do protótipo, foram utilizados diagramas de transição de estados (DTEs), confeccionado através da ferramenta *Enterprise Architect* (STRALEY, 2002) para as seguintes funcionalidades: criar trecho, alterar trecho, modificar trecho, partir trecho, juntar trecho e tornar trecho paralelo.

3.2.1 Diagramas de Atividades e de Estado do Protótipo

O diagrama de atividades, representado na fig. 9, mostra uma visão geral das funcionalidades do protótipo. Depois de abrir o editor, o usuário poderá decidir entre três atividades macros (manutenção do arquivo, manutenção da malha ou modos de visualização).

A função manutenção do arquivo é responsável pelas operações relacionadas à manipulação do arquivo como: carregar arquivo, quando já existir uma malha salva em um arquivo; gravar arquivo, quando o usuário desejar salvar o desenho; e novo arquivo.

A função manutenção da malha agrega as principais funcionalidades deste protótipo como: criar trecho (cria trecho simples); modificar trecho (altera as propriedades do trecho criado); juntar trecho (faz a ligação entre dois trechos); tornar trecho paralelo (torna paralelo o segundo trecho em relação ao primeiro); e partir trecho (divide o trecho conforme necessidade do usuário).

A função modos de visualização permite ao usuário aumentar ou diminuir o tamanho da malha visualizada.

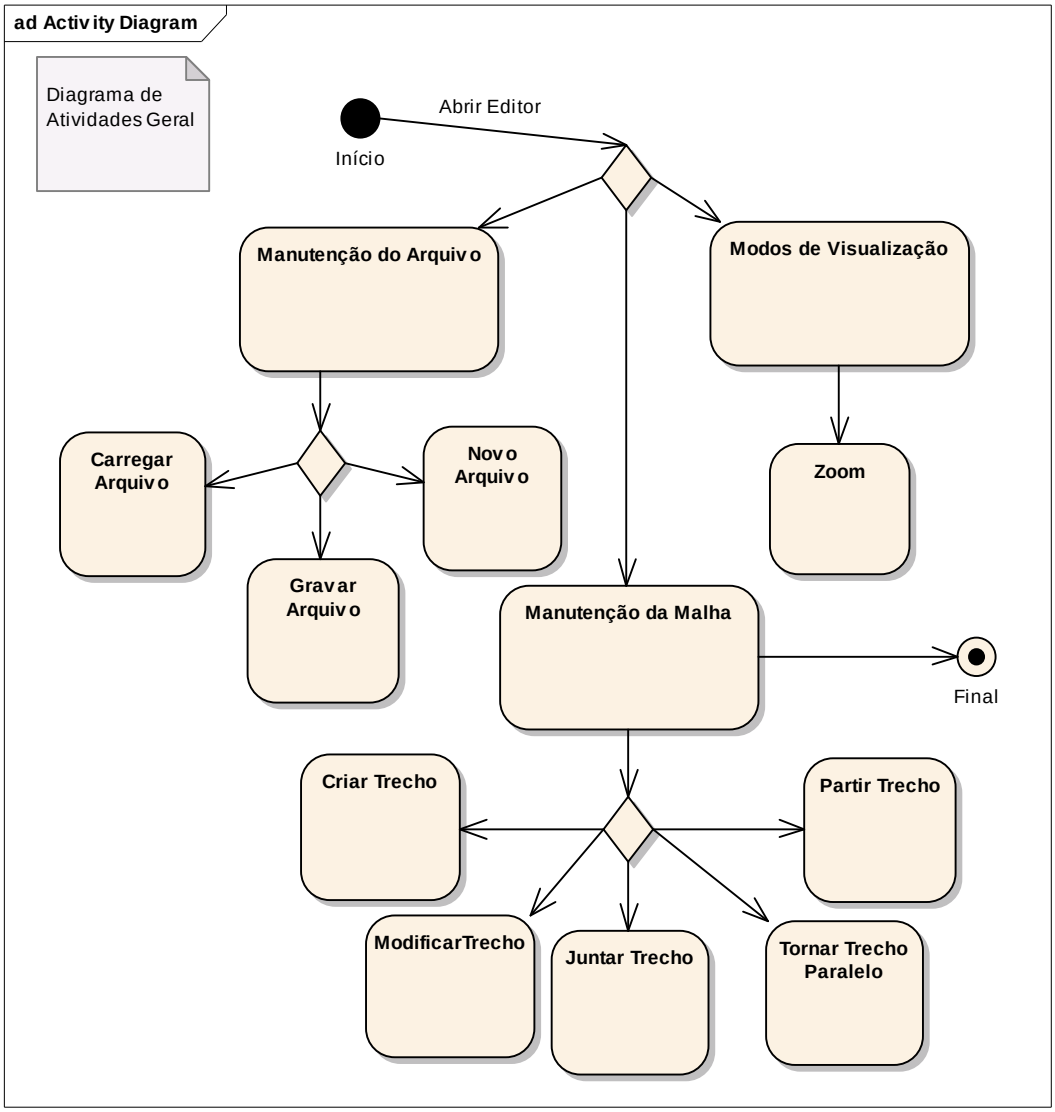


Figura 9 - Diagrama de atividades do sistema implementado

3.2.1.1 Processo “Criar Malha Viária”

O processo de criação da malha viária ocorre quando vários trechos são criados e posicionados na área de desenho com o objetivo de simular uma malha viária do mundo real. As funções implementadas no protótipo visam facilitar todo este processo de criação, para que se ganhe tempo na confecção da malha.

A qualquer momento durante a criação da malha viária o usuário pode salvar a malha rodoviária urbana.

3.2.1.2 Processo “Criar Trecho"

O diagrama de transição de estados representado através da fig. 10, demonstra a sequência executada pelo protótipo durante a criação de um trecho viário na malha urbana. Para criar um trecho, o usuário deve selecionar inicialmente uma posição inicial e final na área de desenho através da função arrastar/soltar do *mouse*.

No próximo estado, o sistema armazena estas posições em uma estrutura de dados. Antes do desenho ser mostrado em tela o sistema ainda verifica se os trechos se cruzam. Se a quantidade de interseções for igual à zero, então o sistema vai para a rotina de desenho do mapa. Se a quantidade de interseções for diferente de zero, o sistema divide os trechos em N segmentos, sempre no ponto de intersecção, seguindo logo após para a rotina de desenho do mapa.

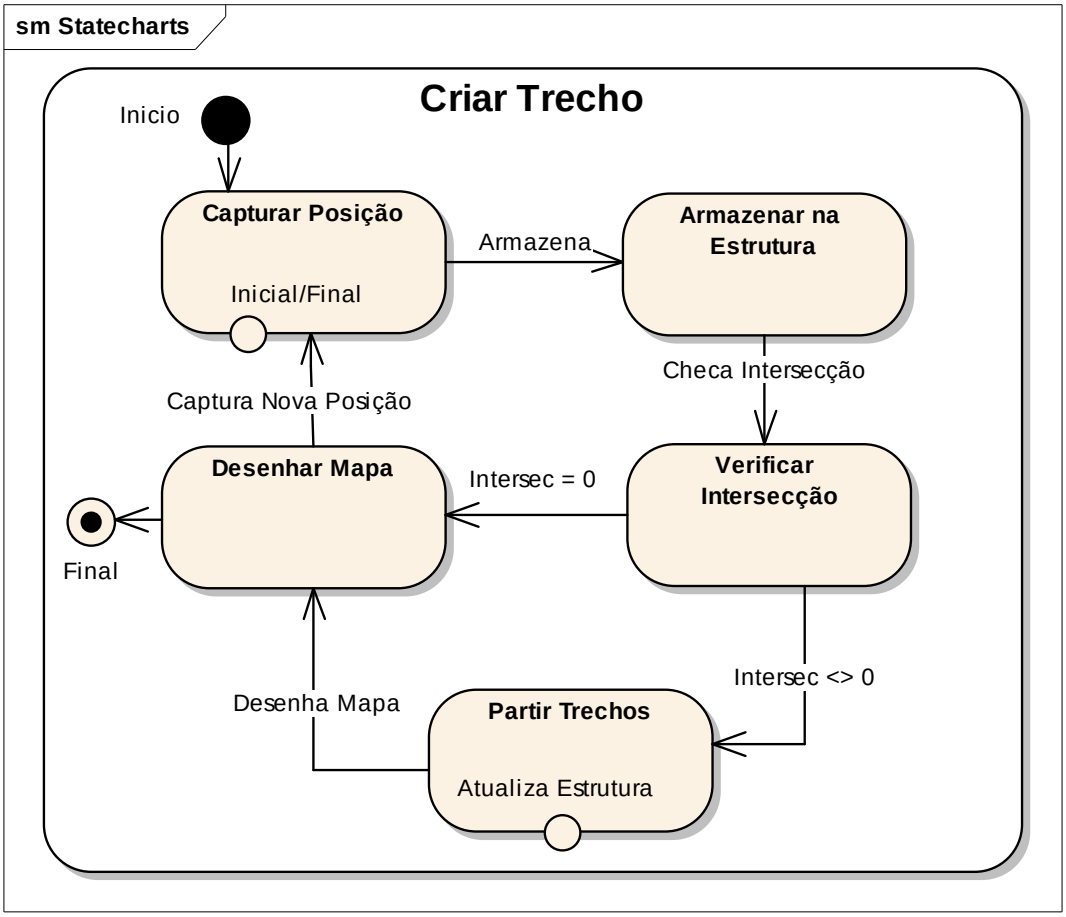


Figura 10 - Diagrama de estado criar trecho

3.2.1.3 Processo “Modificar Atributos do Trecho”

O diagrama de transição de estados representado através da fig. 11**,** demonstra a sequência executada pelo protótipo durante a alteração das propriedades de um trecho viário na malha viária.

Para modificar as propriedades de um trecho dentro da malha viária, o usuário seleciona o trecho desejado e executa as modificações. As alterações feitas pelo usuário são armazenadas na estrutura. No próximo estado, o sistema verifica se existe interseção e desenha o mapa na área de desenho.

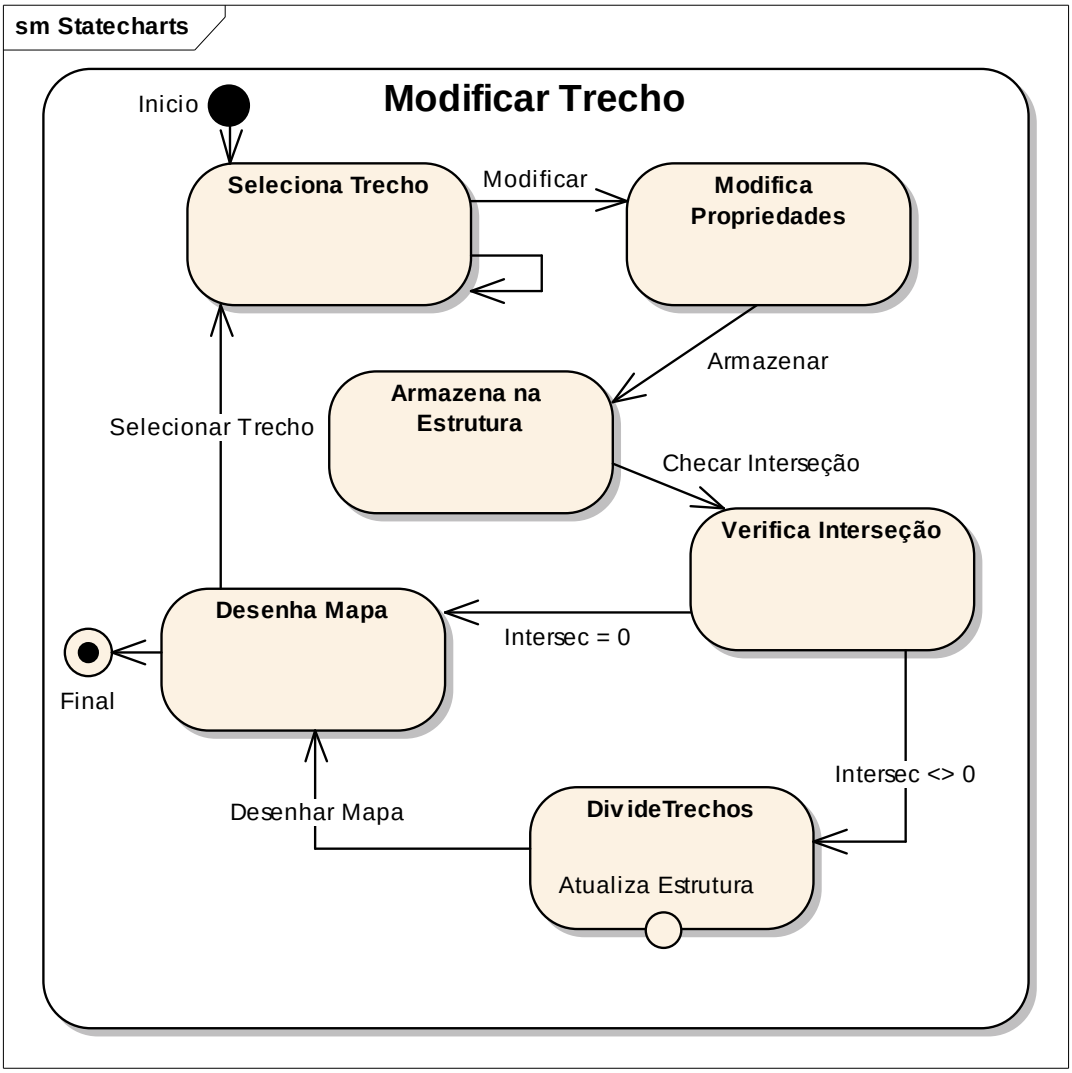


Figura 11 – Diagrama de estado modificar trecho

3.2.1.4 Processo “Partir Trecho”

O diagrama de transição de estados representado através da fig. 12**,** demonstra a sequência executada pelo protótipo durante a divisão de um trecho viário na malha viária.

Para dividir um trecho dentro da malha viária, o sistema verifica inicialmente se o ponto que o usuário está tentando selecionar através do clique do *mouse* encontra-se dentro ou fora do trecho. Se o resultado for verdadeiro, as coordenadas são capturadas e repassadas para a rotina divide trecho, que por sua vez atualiza a estrutura de dados e chama a rotina de desenho do mapa.

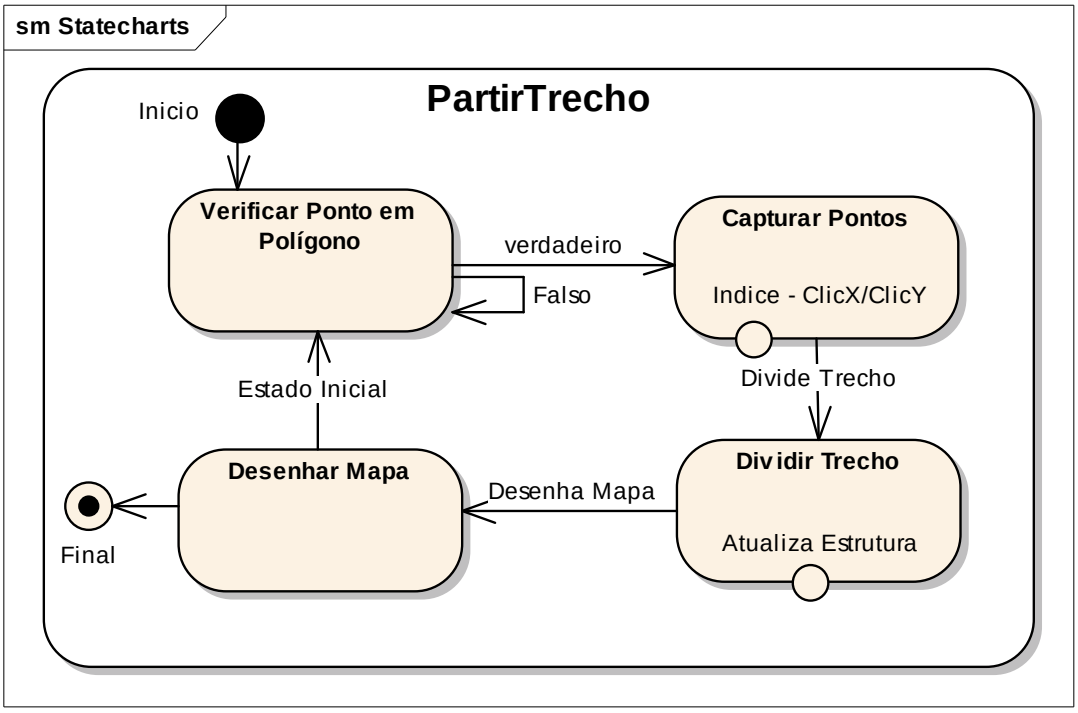


Figura 12 - Diagrama de estado partir trecho

3.2.1.5 Processo “Juntar Trecho”

O diagrama de transição de estados representado através da fig. 13, demonstra a sequência executada pelo protótipo durante a divisão de um trecho na malha viária.

Para unir dois trechos, o sistema verifica inicialmente se os pontos selecionados encontram-se dentro ou fora dos trechos. Caso estejam dentro dos trechos, o sistema captura os pontos e direciona o processamento para a rotina junta trecho. Logo após, a estrutura de dados é atualizada com as novas informações, sendo acionada na seqüência a rotina desenha mapa.

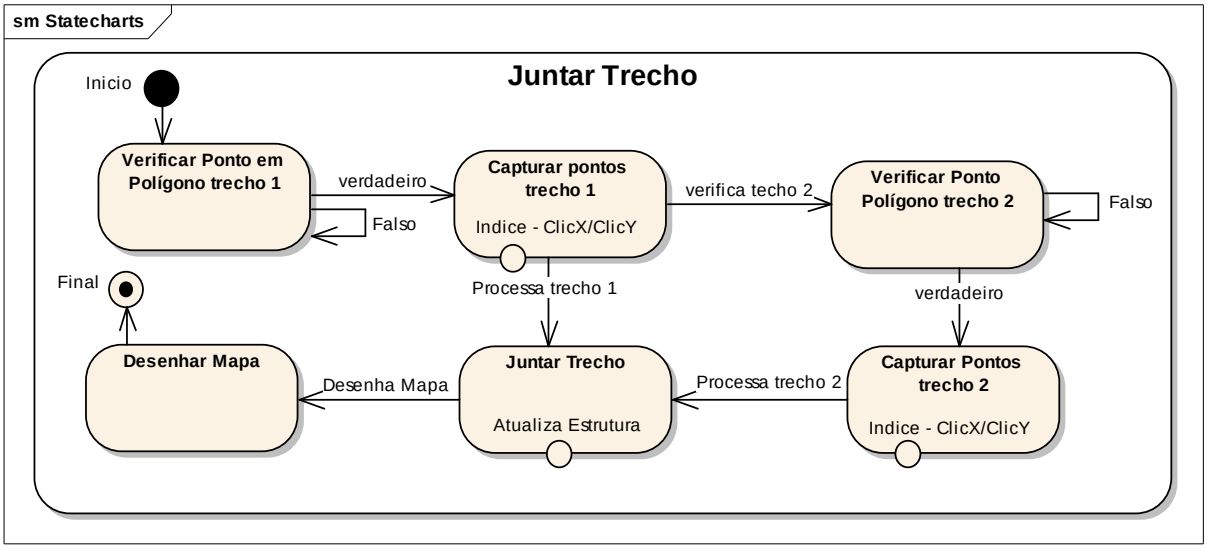


Figura 13 - Diagrama de estado juntar trecho

3.2.1.6 Processo “Tornar Trecho Paralelo”

O diagrama de transição de estados representado através da fig. 14, demonstra a sequência executada pelo protótipo durante a divisão de um trecho viário na malha viária.

Para tornar um trecho paralelo à outro já existente, o processo é semelhante ao processo juntar trecho. Os pontos selecionados pelo usuário são verificados pelo sistema se a localização destes estão dentro ou fora dos trechos. Se estiverem dentro dos trechos, os pontos são capturados e repassados para a rotina fazer trecho paralelo. Logo após, a estrutura de dados é atualizada e o mapa desenhado na área de desenho.

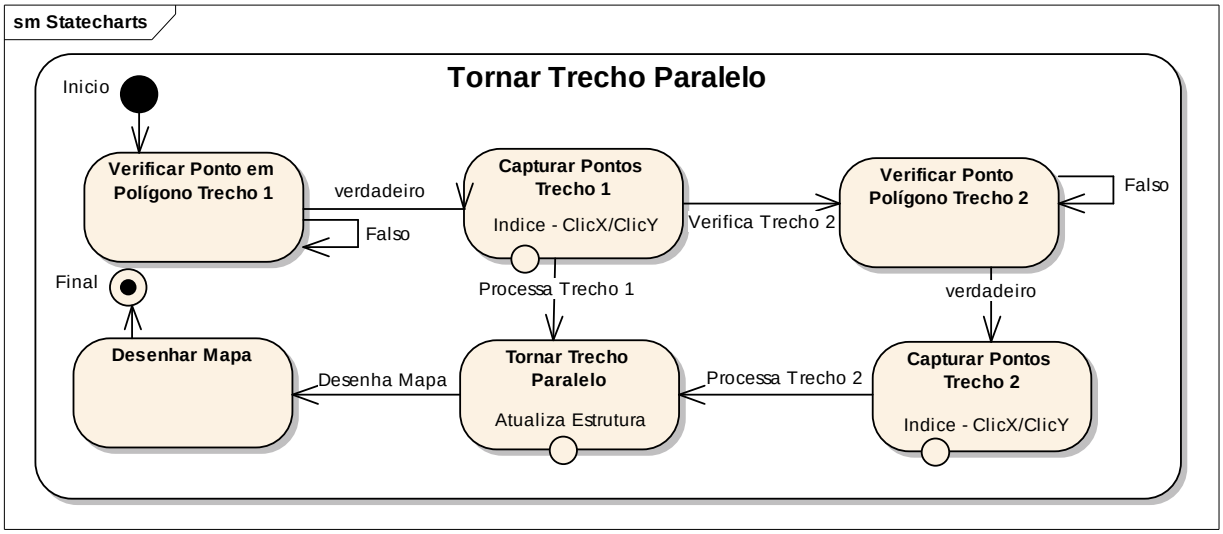


Figura 14 - Diagrama de estado tornar trecho paralelo

3.2.1.7 Processo “Desenhar Mapa”

O diagrama de transição de estados representado através da fig. 15, demonstra a sequência executada pelo protótipo durante o processo de desenho da malha viária.

O processo “Desenhar Mapa” é chamado pelo programa sempre que há alguma alteração nos trechos desenhados na tela. Desta forma, quando ocorre qualquer processo citado anteriormente, desde a criação de um novo trecho até tornar um trecho paralelo, a rotina “Desenhar Mapa” é ativada. Esta rotina é composta por três subrotinas: DesenhaLinhaLarg, DesenhaLinhaCentro, DesenhaSeta.

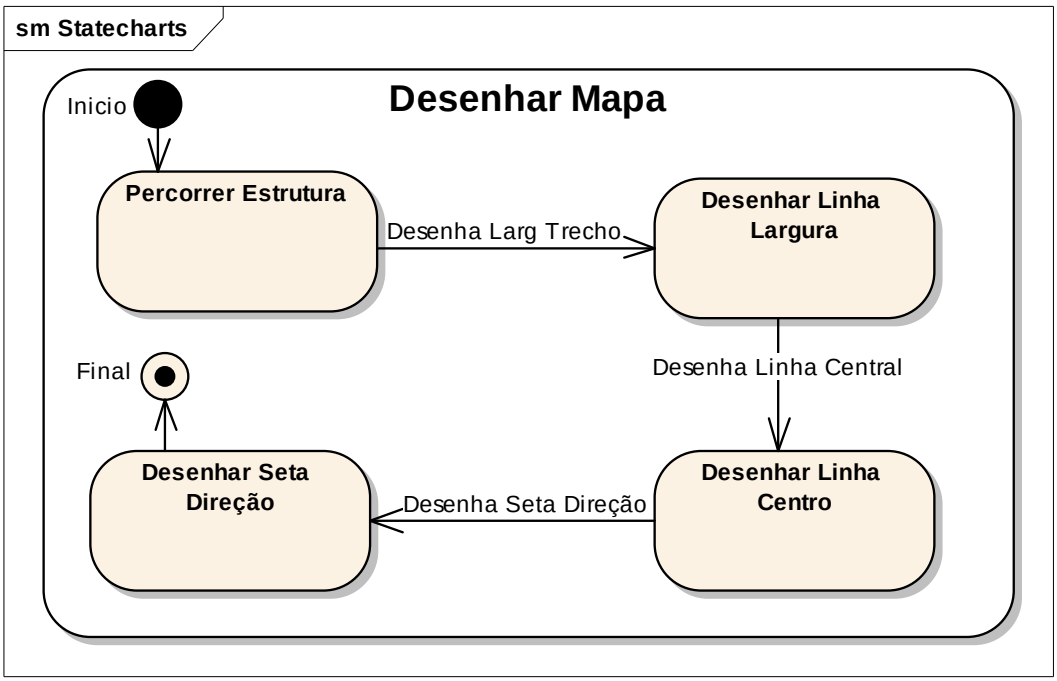


Figura 15 - Diagrama de estado Desenhar Mapa

3.3 IMPLEMENTAÇÃO

O protótipo foi implementado na ferramenta de programação Delphi, usando a biblioteca gráfica OpenGl.

Os trechos das ruas são armazenados em uma estrutura do tipo fila simples. A obtenção de informações sobre trechos é feita através de uma pesquisa seqüencial na estrutura (tipo fila).

O protótipo permite a construção de uma malha viária a partir de trechos individuais que podem ser conectados ou divididos. Em tempo de execução o protótipo trata também a interseção de vários trechos, quebrando em pedaços a partir do ponto de intersecção. A sobreposição horizontal e/ou vertical dos trechos não é tratada pelo software, sendo necessário a intervenção manual do usuário às propriedades do trecho, ou através da funcionalidade tornar trecho paralelo, localizada na barra de botões do software. A fig. 16 ilustra esta situação.

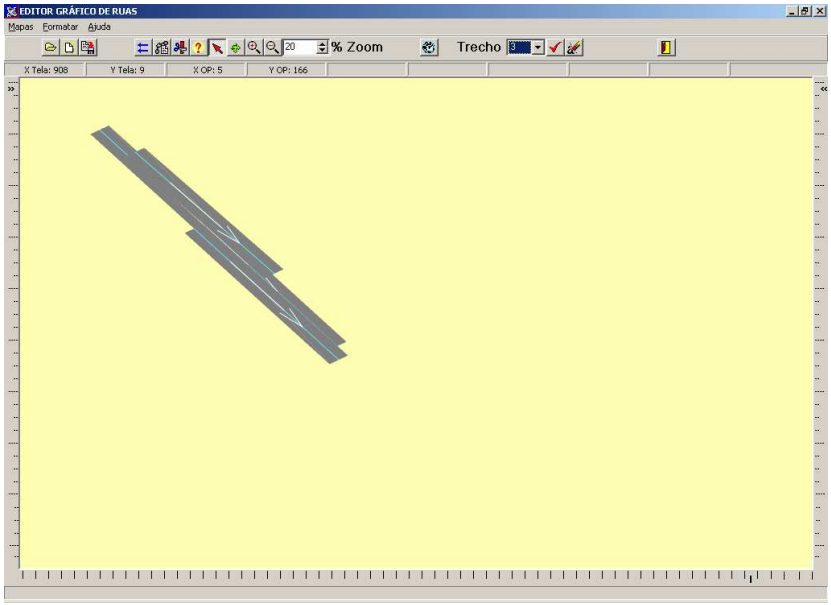


Figura 16 - Sobreposição de trechos

3.3.1 Técnicas e Ferramentas Utilizadas

Para a implementação da ferramenta foi explorado o uso da programação orientada a eventos.

Segundo Cantú (2003, p. 5), na programação orientada a eventos, as ações do usuário (denominadas eventos) é que determinam o fluxo de execução do programa, ou seja, qual procedimento/rotina será chamada. Isso significa que num programa desenvolvido com base nessa técnica, as chamadas das rotinas/procedimentos dependem dos eventos que ocorrem, em outras palavras, dependem do que o usuário fará ou não.

3.3.2 Descrição da Rotina Desenha Mapa

A rotina desenha mapa percorre toda a estrutura do tipo fila e utiliza as funções de desenho para desenhar a largura do trecho, a linha central e a linha de direção (Quadro 2).

```
                        //  ROTINA DESENHA MAPA
Begin
  //verifica se malha não está vazia
  if L_Malha <> nil then
  begin
     //percorre a malha
     for z:= 0 to L_Malha.Count-1 do
     begin
        //aloca estrutura para manipulação
        Ponteiro:= L_Malha.Items[z];
        //chama a função desenhalinhalarg
        DesenhaLinhaLarg(Ponteiro.Inicio,Ponteiro.Fim,Ponteiro.F_Largura);
        //chama a função desenhalinhacentro
        DesenhaLinhaCentro(Ponteiro.Inicio,Ponteiro.Fim,Ponteiro.F_Largura);
        //chama a função desenhaseta
        DesenhaSeta(Ponteiro.Inicio, Ponteiro.Fim, Ponteiro.F_Largura);
     end;
  end;
End;                    //FIM DA ROTINA
```

Fonte: adaptado de Freire (2004).

Quadro 2 - Rotina desenha mapa

A rotina que desenha a seta central (a qual indica o sentido da rua), utiliza os pontos inicial e final do trecho para calcular o ponto médio e chama a função desenhalinha para traçar a ponta da reta (Quadro 3).

```
   // DESENHA SETA
begin
  //Localiza ponto medio entre o Inicio e fim
  PtMedio:= CalcPtMedio(I, F);
  //Localiza ponto medio entre o Inicio e Ponto medio
  ISeta:= CalcPtMedio(I, Ptmedio);
  //Localiza ponto medio entre o Ponto medio e fim
  FSeta:= CalcptMedio(Ptmedio, F);
  //define novo ponto medio
  PtMedio:= CalcPtMedio(PtMedio, Fseta);
  //definir cor da linha
            glColor3f(1.0, 1.0, 1.0);
  //desenha linha central da seta
  DesenhaLinha(Iseta, Ptmedio);
  //Acha os dois pontos para tracar ponta reta
  CalcPt(PtMedio, FSeta,L/4, A, B, Lixo, Lixo);
  //cria ponta da seta
  DesenhaLinha(B, Fseta);
  DesenhaLinha(A,Fseta);
end;
```

Fonte: adaptado de Freire (2004).

Quadro 3 – Rotina desenha seta

A rotina desenha mapa é chamada sempre dentro da rotina redesenha mapa, para que todo o conteúdo da área de desenho seja redefinido, atualizando assim, o *Device Context* da biblioteca gráfica OpenGl (Quadro 4).

```
                              //   ROTINA REDESENHA MAPA
Begin
      // recebe como parâmetro a nova largura e altura da janela.
      glViewport (0, 0, Width, Height);
      // avisa o OpenGL das futuras alterações
     glMatrixMode (GL_PROJECTION);
     glLoadIdentity;
     // determinar que a projeção ortográfica (2D) será utilizada para exibir na tela a imagem 2D
     gluOrtho2D(Esquerda, Direita, Abaixo, Acima);
     // avisa o OpenGL das futuras alterações
     glMatrixMode(GL_MODELVIEW);
     glLoadIdentity;
     // limpa o buffer
     glClear(GL_COLOR_BUFFER_BIT or GL_DEPTH_BUFFER_BIT);
  if FrmPrincipal.BtnReta.Down and Reta then
    DesenhaLinhaLarg(Click,ClickF,6);
 //charma função desenhamapa
  DesenhaMapa;
 // redesenha a cena
  SwapBuffers (DC);

end;                        //     FIM DA ROTINA
```

Fonte: adaptado de Freire (2004).

Quadro 4 – Rotina redesenha mapa

3.3.3 Implementação da Rotina Ponto em Polígono

A rotina de ponto em polígono verifica se o ponto referenciado pelo usuário está à direita ou esquerda dos pontos que compõem a figura. Isto é feito desenhando-se uma linha imaginária deste ponto até o cruzamento de um dos lados da figura. Se o número de cruzamentos for ímpar, significa que o ponto selecionado pelo usuário está dentro da figura. Se estiver fora da figura, o número de cruzamentos será par e a rotina enviará ao sistema esta informação.

Nesta rotina também foi implementado o comando de seleção *case* para verificar a localização do ponto selecionado pelo usuário dentro da figura. Desta forma é possível saber se o ponto está antes ou depois do ponto médio. Este recurso é utilizado na função juntar trecho, para ligar os trechos de acordo com a extremidade escolhida pelo usuário.

No Quadro 5, está representada a rotina ponto em polígono. Nesta rotina é feita a verificação se o ponto está dentro de um polígono.

```
Begin                          // ROTINA PONTO EM POLÍGONO
 If L_Malha <> nil then
 begin
   For z:= 0 to L_Malha.Count-1 do
   begin
    Ponteiro:= L_Malha.Items[z];
     Case tipo of
     1: CalcPt(Ponteiro.Inicio,Ponteiro.Fim,Ponteiro.F_Largura, PtA, PtB, PtC, PtD);
     2: CalcPt(Ponteiro.Inicio,CalcPtMedio(Ponteiro.Inicio,Ponteiro.Fim),Ponteiro.F_Largura, PtA, PtB, PtC,
PtD);
     3: CalcPt(CalcPtMedio(Ponteiro.Inicio,Ponteiro.Fim),Ponteiro.Fim,Ponteiro.F_Largura, PtA, PtB, PtC, PtD);
     end;
    Fronteira[1]:= PtA;   Fronteira[2]:= PtB;    Fronteira[3]:= PtC;
    Fronteira[4]:= PtD;   Fronteira[5]:= PtA;
    n:=0;
    For i:=1 to 4 do    // teste cada lado do polígono
    begin
     PtE:= Fronteira[i];
     PtF:= Fronteira[i+1];
     If (abs(PtE.Y - PtF.Y) > Erro) and (abs(PtF.X - PtE.X) > Erro) then  // lado não é horizontal
     begin
      Xinter:= (PtY - PtE.Y + (PtF.Y - PtE.Y)/(PtF.X-PtE.X)*PtE.X)/
            ((PtF.Y - PtE.Y)/(PtF.X - PtE.X));
      If (abs(PtX - PtE.X)) < Erro then
      begin  //PtE é da Fronteira... PARE
        n:= 1;  break;
      end
      else // Não têm y mínimo
        If (Xinter > PtX) and (PtY > min(PtE.Y,PtF.Y)) and (PtY <= max(PtE.Y,PtF.Y)) then
          n:= n + 1;
     end
     else  // ponto é da fronteria horizontal... PARE
       If (abs(PtE.Y - PtF.Y) > Erro) then
         PtG:= Fronteira[3];
         if (PtY >= min(PtE.Y,PtG.Y)) and (PtY <= max(PtE.Y,PtG.Y)) and
           (PtX >= min(PtE.X,PtF.X)) and (PtX <= max(PtE.X,PtF.X)) then   // VERTICAL
         begin
           n:= 1;  break;
         end
       else
        if (abs(PtF.X - PtE.X) > Erro) then
          PtG:= Fronteira[3];
          if  (PtY >= min(PtE.Y,PtF.Y)) and (PtY <= max(PtE.Y,PtF.Y)) and
            (PtX >= min(PtE.X,PtG.X)) and (PtX <= max(PtE.X,PtG.X)) then      //HORIZONTAL
          begin
            n:= 1; break;
          end;
    end; //end For
    if ((n mod 2) <> 0) then
       break;
  end; // end For Malha
end;
 impar:= ((n mod 2) <> 0); // Verdadeiro se ímpar
 If impar then
  Result:=z

end; // FIM
```

Fonte: adaptado de Figueiredo (1991).

Quadro 5 - Rotina ponto em polígono.

3.3.4 Implementação da Rotina Partir Trecho

A rotina partir trecho divide um trecho em duas partes. Para que o novo trecho seja uma continuação do original, uma nova coordenada é calculada para o Ponto Y, utilizando-se a equação da reta. Em trechos verticais, é utilizada a coordenada do ponto Y do *click* do *mouse*, e adota-se qualquer coordenada de X por ambos serem iguais. No quadro 6 , está representada a rotina partir trecho.

```
Begin                          // ROTINA PARTIR TRECHO
   If L_Malha <> nil then
   begin
      For z:= 1 to L_Malha.Count do
      begin
         If z = indice then
         begin
           // Percorre Lista
            Ponteiroz:= L_Malha.Items[z-1];
           //  Calcula Coeficiente Angular
           CAng:= (Ponteiroz.Fim.Y-Ponteiroz.Inicio.Y)/Aux.
           // Calcula Coeficiente Linear
           CLin:= Ponteiroz.Inicio.Y-(CAng * Ponteiroz.Inicio.X);
           Aux:= Ponteiroz.Fim.X-Ponteiroz.Inicio.X;
          // Se o trecho é vertical, Xi = Xf , captura apenas Y para corte.
          If Aux = 0 then
          begin
             AuxFimX:= Ponteiroz.Fim.X;
             AuxFimY:= Ponteiroz.Fim.Y;
             Ponteiroz.Fim.Y:= PtY;
            Guarda_RetasEX(L_Malha.Count+1,0,0,0,0,0,AuxFimX,AuxFimY,AuxFimX,PtY,6,true);
            break;
          end;
           // Calcula novo Ponto Y
           NovoPtY:= (CAng * PtX) + CLin;
           // Guarda Pontos Finais
           AuxFimX:= Ponteiroz.Fim.X;
           AuxFimY:= Ponteiroz.Fim.Y;
           // Entrada de novos valores finais
           Ponteiroz.Fim.X:= PtX;
           Ponteiroz.Fim.Y:= NovoPtY;
           // Atualiza Estrutura
           Guarda_RetasEX(L_Malha.Count+1,0,0,0,0,0,PtX,NovoPtY,AuxFimX,AuxFimY,6,true);
         end;
      end;
   end;
end;                                 //   FIM DA ROTINA
```

Quadro 6 – Rotina partir trecho

3.3.5 Implementação da Rotina Tornar Trecho Paralelo

Na rotina tornar trecho paralelo, utiliza-se as informações do segundo trecho para deslocar o primeiro trecho para junto deste. Desta forma, as dimensões do trecho não são

alteradas, apenas a sua localização na área de desenho. No quadro 7 está representada a rotina tornar trecho paralelo.

```
Begin                          // ROTINA TORNAR TRECHO PARALELO
 If L_Malha <> nil then
 begin
   // Percorre Lista
   ponteirofig1:= L_Malha.Items[ind1];
    ponteirofig2:= L_Malha.Items[ind2];

   // Calcula Pontos Extremos
   CalcPt(Ponteirofig2.Inicio,Ponteirofig2.Fim,Ponteirofig2.F_Largura, PtA, PtB, PtC, PtD);

  // Calcula DX e DY
   dX:=  PtC.X - PtB.X;
   dY:=  PtC.Y - PtB.Y;
   dX1:= PtB.X - PtC.X;
   dY1:= PtB.Y - PtC.Y;

  // Calcula largura total
  larguratotal:= (ponteirofig1.F_Largura/2) + (ponteirofig2.F_Largura/2);

 // Calcula Delta X e Delta Y
  if ((ponteirofig2.Inicio.X - ponteirofig2.Fim.X) > Erro) then
  begin
    deltaX:= larguraTotal;
    deltaY:= 0;
  end else
   if ((PtB.Y - PtC.Y) < Erro) then
   begin
     deltaY:= larguraTotal;
     deltaX:= 0;
   end else
         If ((dX - dY) > Erro) then
          begin
           angulo:= ArcTan(dY/dX);
           deltaX:= larguraTotal * sin (angulo);
           deltaY:= larguraTotal * cos (angulo);
          end;
  end;
  //  Armazena
  auxFimX := PtC.X     + deltaX;
  auxFimY := PtC.Y     - deltaY;
  auxIniX:=  PtB.X  + deltaX;
  auxIniY:=  PtB.Y  - deltaY;

  // Atualiza Estrutura
  ponteiroz:= L_Malha.Items[ind1];
  ponteiroz.Inicio.X:= auxIniX;
  ponteiroz.Inicio.Y:= auxIniY;
  ponteiroz.Fim.X:= auxFimX;
  ponteiroz.Fim.Y:= auxFimY

end;           // FIM DA ROTINA
```

Quadro 7 – Rotina tornar trecho paralelo

3.3.6 Implementação da Rotina Juntar Trechos

A rotina juntar trechos recebe como parâmetro o índice do trecho na estrutura. Verifica-se então, em qual das metades do trecho o ponto está localizado. Logo após, a lista é percorrida para alterar-se os valores conforme seleção do usuário. O quadro 8 representa a rotina juntar trecho.

```
Begin              //ROTINA JUNTAR TRECHO
  If L_Malha <> nil then
  Begin
      // Verifica em qual das metades do trecho se encontra o ponto cliclado e
       armazena posição.
      If PontoEmPoligono(Pt1.X,Pt1.Y,2)<> 99 then
        fig1:= 'inicio'
       else
        fig1:= 'fim';
      If PontoEmPoligono(Pt2.X,Pt2.Y,2)<> 99 then
        fig2:= 'inicio'
       else
        fig2:= 'fim';

      // Percorre Lista e junta partes se condições forem atendidas
      Ponteiroz:= L_Malha.Items[ind2];
      If (fig1 = 'inicio') and (fig2 = 'inicio') then
      aux:= Ponteiroz.Inicio
       else If (fig1 = 'inicio') and (fig2 = 'fim') then
            aux:= Ponteiroz.Fim;

      Ponteiroz:= L_Malha.Items[ind2];
      If (fig1 = 'fim') and (fig2 = 'inicio') then
      aux:= Ponteiroz.Inicio
       else If (fig1 = 'fim') and (fig2 = 'fim') then
            aux:= Ponteiroz.Fim;

      Ponteiroz:= L_Malha.Items[ind1];
      If (fig1 = 'inicio') then
        Ponteiroz.Inicio:= aux
      else
        Ponteiroz.fim:= aux;
  end;

end;                    //FIM DA ROTINA
```

Quadro 8 – Rotina juntar trecho

3.3.7 Implementação da Rotina Criar Trecho

A rotina criar trecho recebe como parâmetro a posição inicial e final do *click* na tela. Logo após, armazena-se estas informações na estrutura do tipo lista. Verifica-se então, se existe interseção percorrendo toda a estrutura do tipo lista. Se existir, os dois trechos originais

são alterados, e outros dois novos trechos são criados. Em seguida todo o conteúdo da tela é redesenhado. O quadro 9 representa a rotina criar trecho.

```
// Se botão da reta estiver pressionado.   -    // ROTINA CRIAR  TRECHO
if FrmPrincipal.BtnReta.Down then
  begin
    // Guarda ponto final do click
    ClickF.X:= PtX;
    ClickF.Y:= PtY;
    // Armazena na estrutura fila.
    Guarda_Retas(indice,Click,ClickF,6,false);
    // Verifica se existe interseção.
    calcIntersec;
    // Incrementa contador
    indice:= L_Malha.Count+1;
    // Atualiza ComboBox
    frmCheck.AtualizarComboBox;
    reta:= false;                  // desabilita desenho da reta
    redesenha;                   // redesenha mapa
  end;                                        // FIM DA ROTINA
```

Quadro 9 - Rotina criar trecho

3.3.8 Implementação da Rotina Modificar Atributos do Trecho

Através desta rotina o usuário poderá alterar todos os atributos que compõem um trecho (x inicial e final, y inicial e final, largura e direção). O quadro 10 representa a rotina modificar atributos do trecho.

```
if inserir then              //ROTINA MODIFICAR ATRIBUTOS DO TRECHO
  begin
    // Armazena informações da Janela na Lista
    Guarda_RetasEx(indice,StrToInt(edtmd.Text), StrToInt(edtme.Text),StrToInt(edtni.Text),
                   StrToInt(edtqe.Text), StrToInt(edtvq.Text), StrToFloat(edtxi.Text), StrToFloat(edtyi.Text),
                   StrToFloat(edtxf.Text),StrToFloat(edtyf.Text),StrToFloat(edtlargura.Text),false);
    AtualizarComboBox;
    indice:=L_Malha.Count+1;
    inserir:=False;
  end else
  begin
     Ponteiro.I_CdTrecho:=StrToInt(edtReta.Text);
    Ponteiro.Inicio.X:=StrToFloat(edtXi.Text);
    Ponteiro.Inicio.Y:=StrToFloat(edtYi.Text);
    Ponteiro.Fim.X:=StrToFloat(edtXf.Text);
    Ponteiro.Fim.Y:=StrToFloat(edtYf.Text);
    Ponteiro.MDir:=StrToInt(edtmd.Text);
    Ponteiro.MEsq:=StrToInt(edtme.Text);
    Ponteiro.Nivel:=StrToInt(edtni.Text);
    Ponteiro.Quebra:=StrToInt(edtqe.Text);
    Ponteiro.Vl_Quebra:=StrToInt(edtvq.Text);
    Ponteiro.F_Largura:=StrToFloat(edtlargura.Text);
    Ponteiro.Cruza:= StrToBool(edtcruza.Text);
  end;
  // Redesenha conteúdo da tela.
  FrmGrafico.Redesenha;                        // FIM DA ROTINA
```

Quadro 10 - Rotina modificar atributos do trecho

3.3.9 Operacionalidade da Implementação

As principais funcionalidades do sistema que são criar trecho, modificar trecho, partir trecho, juntar trecho e tornar trecho paralelo estão representadas na fig. 17.

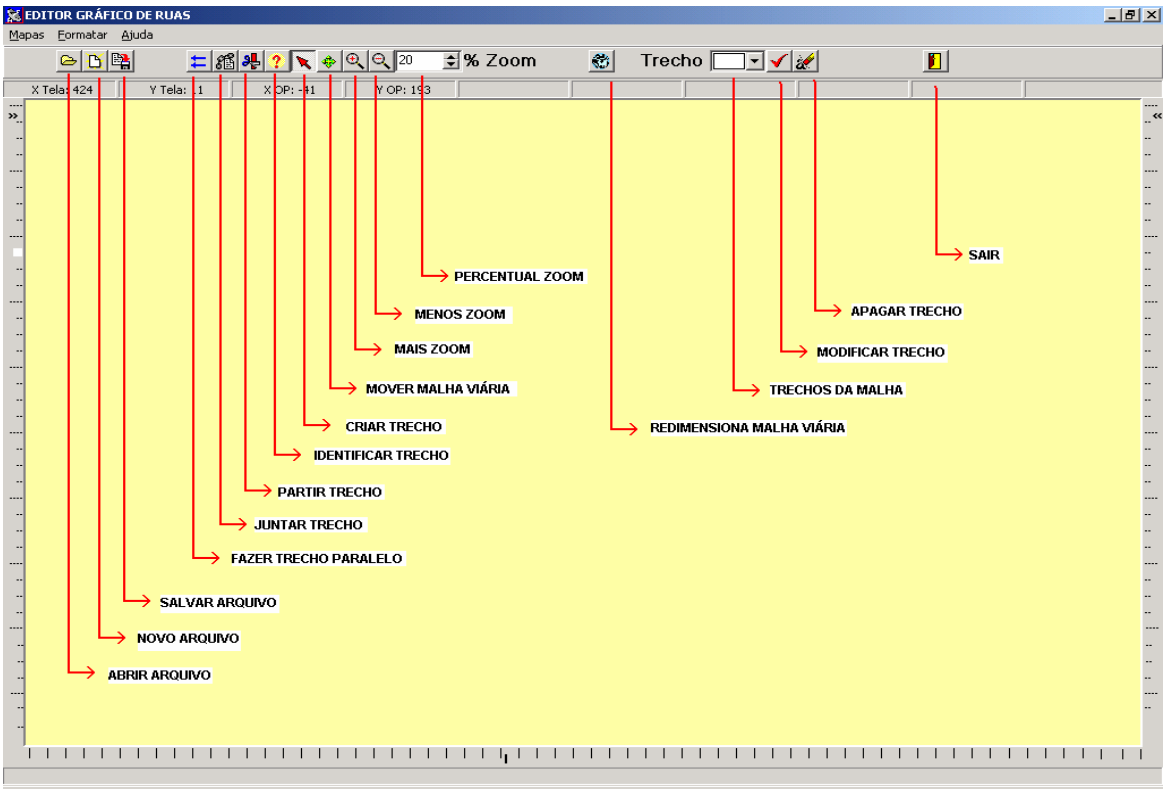


Figura 17 - Representação das funcionalidades do protótipo

3.3.9.1 Criar trecho

Para criar um novo trecho, o usuário deve selecionar o ponto inicial na área de desenho, manter o botão esquerdo pressionado e arrastar até o ponto final desejado. A direção do trecho é definida pelo sentido inicio/fim da criação, ou seja, o trecho representado na fig. 18 foi criado de baixo para cima. Logo, sua direção aponta para cima.

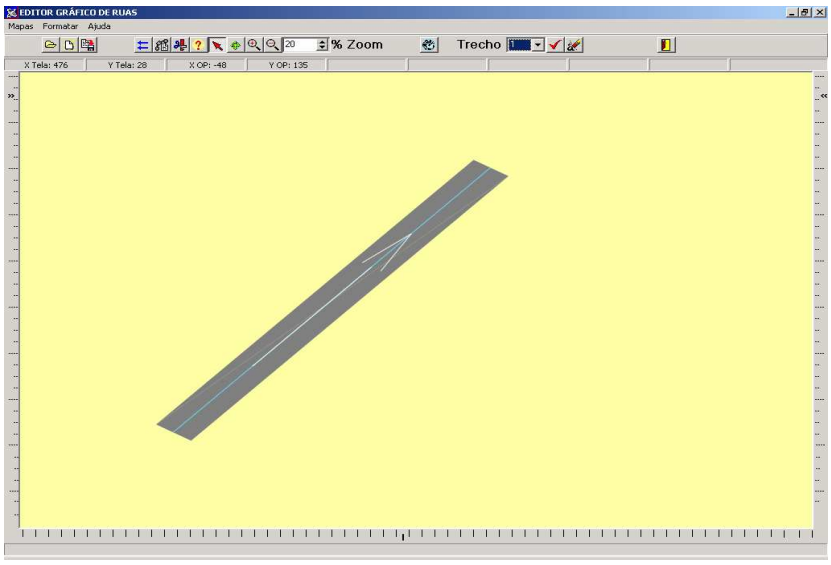


Figura 18 - Criação do primeiro trecho

3.3.9.2 Modificar Trecho

Para modificar as propriedades de um trecho, o usuário deverá selecionar o botão “Modificar”, localizado na barra de botões. Nesta tela é permitido também alterar a direção de um trecho específico da malha rodoviária (fig. 19) .

Figura 19 – Modificar atributos do trecho

3.3.9.3 Partir Trecho

Para partir um trecho em um determinado ponto, o usuário precisa setar o botão “Partir Trecho” e clicar com o *mouse* na altura do trecho de sua preferência. O sistema dividirá o trecho naquela posição, assumindo a mesma direção do trecho original. À partir da fig. 17, o usuário seleciona três pontos e o sistema divide o trecho principal em quatro partes, sendo o resultado apresentado na fig. 20.

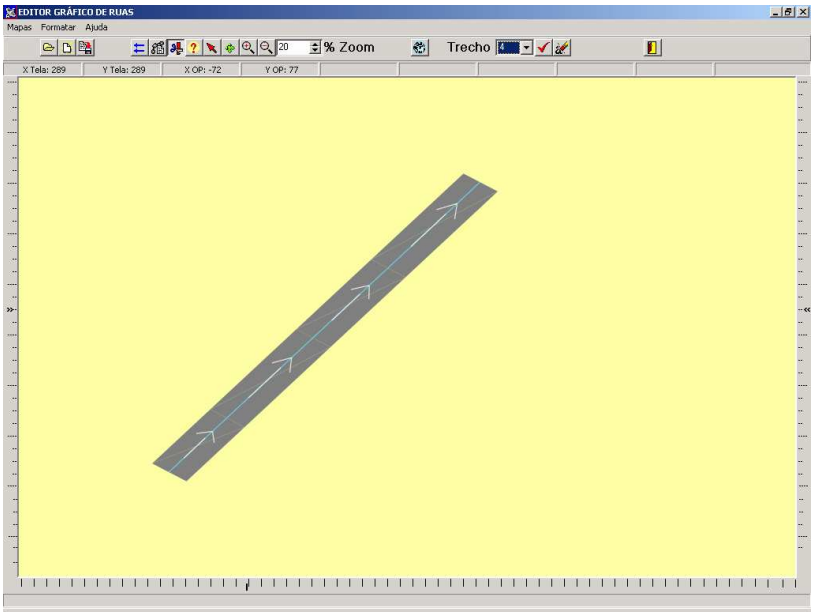


Figura 20 - Resultado final de trecho partido em três pontos

3.3.9.4 Juntar Trecho

Para juntar dois trechos, o usuário deve setar o botão “Juntar Trecho” localizado na barra de botões. Logo após, selecionar os trechos que se deseja juntar. Na figura 21, é apresentado o momento anterior à ligação. Na figura 22, o resultado final da operação. Note que o sistema alonga o segundo trecho para que este alcance o primeiro trecho no mesmo ponto.

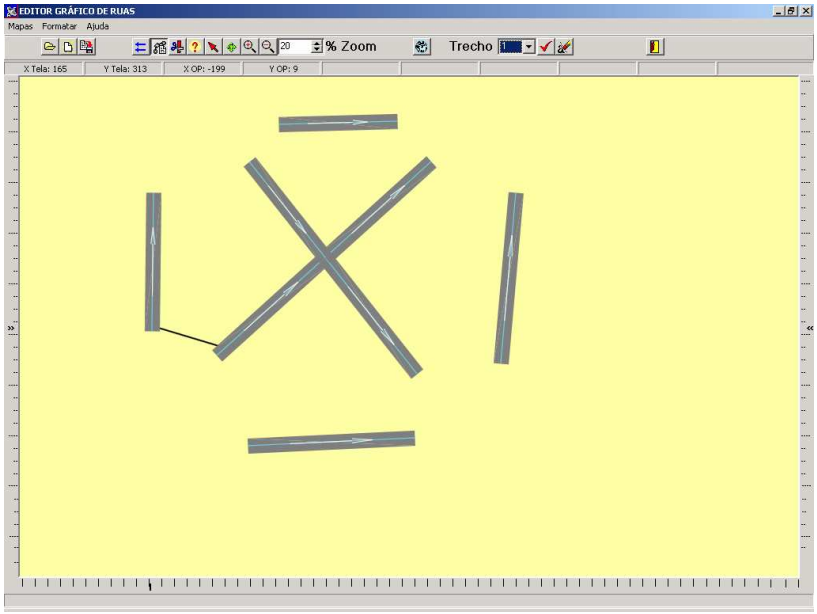


Figura 21 - Início da junção de um trecho

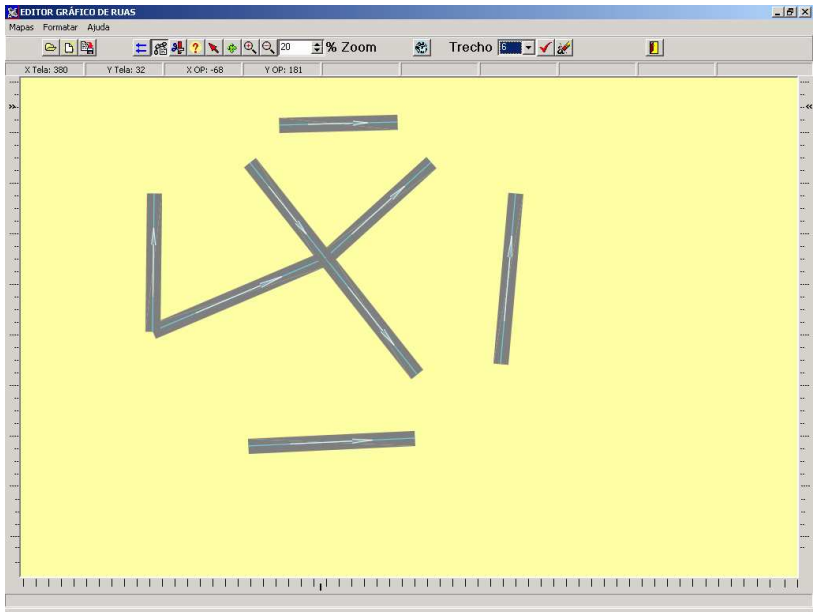


Figura 22 – Trecho juntado

3.3.9.5 Tornar Trecho Paralelo

Para tornar dois trechos paralelos, deve-se setar o botão “Fazer trecho paralelo” na barra de botões. Após, selecionar os dois trechos com o *mouse* na área de desenho do sistema. Na fig. 23, o usuário escolhe o trecho acima para tornar paralelo com o trecho central. Na fig. 24, é apresentado o resultado final da operação tornar trecho paralelo.

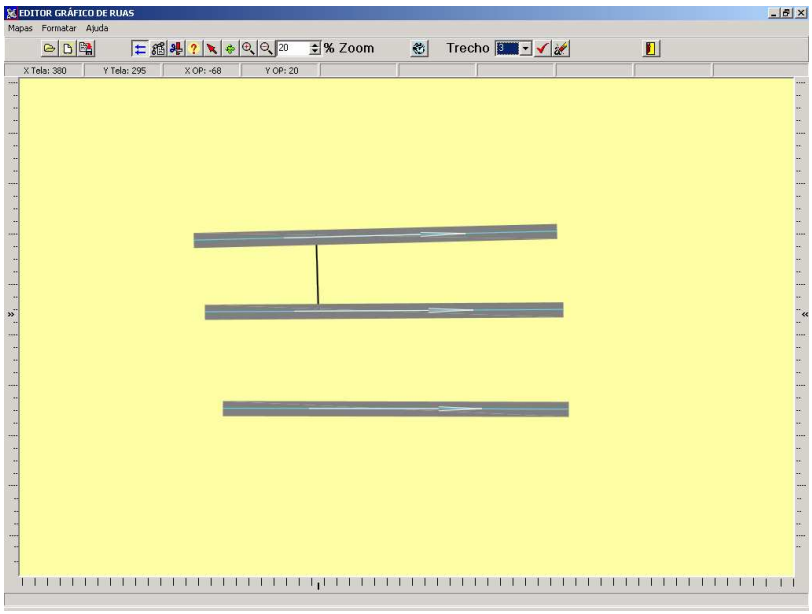


Figura 23 – Processo tornar trecho paralelo

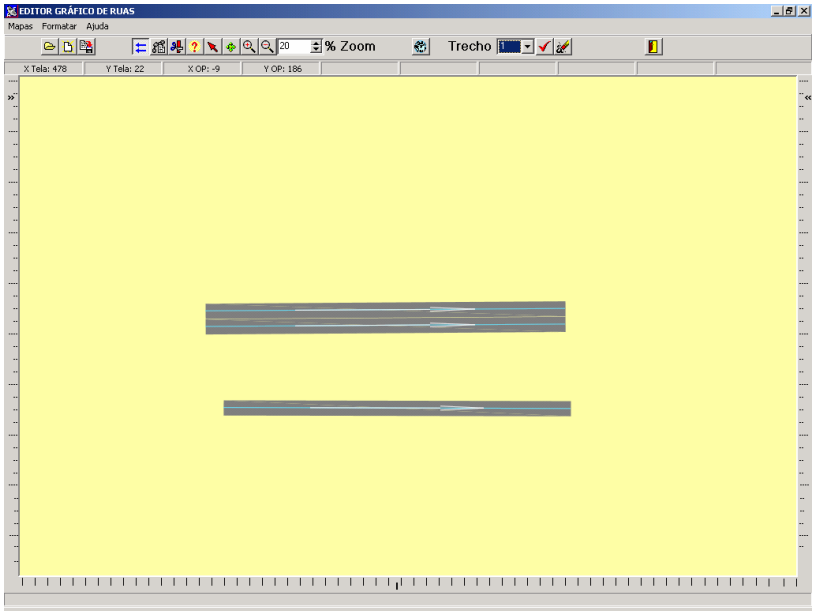


Figura 24 – Resultado do processo tornar trecho paralelo

3.3.9.6 Aumentar ou Diminuir Zoom

A fig. 25 destaca as opções de ajuste do desenho. O usuário pode selecionar o percentual de ajuste do desenho e em seguida pressionar o botão aumentar *zoom* ou diminuir *zoom*. A fig. 26 representa um mapa sem aplicação de zoom, enquanto a fig. 27 representa o resultado final da aplicação com 80% de zoom no mapa original.

Figura 25 – Ajustar percentual de zoom

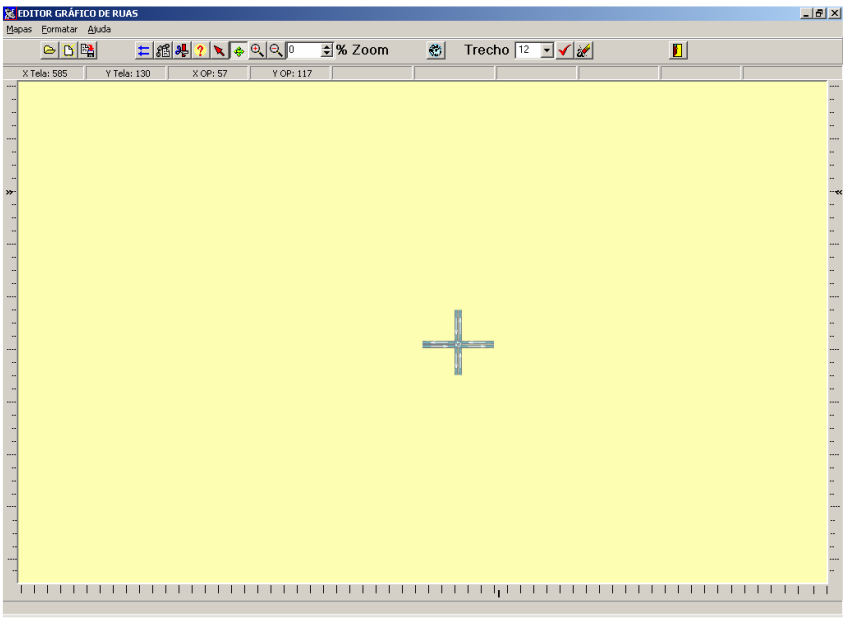


Figura 26 - Mapa antes do zoom

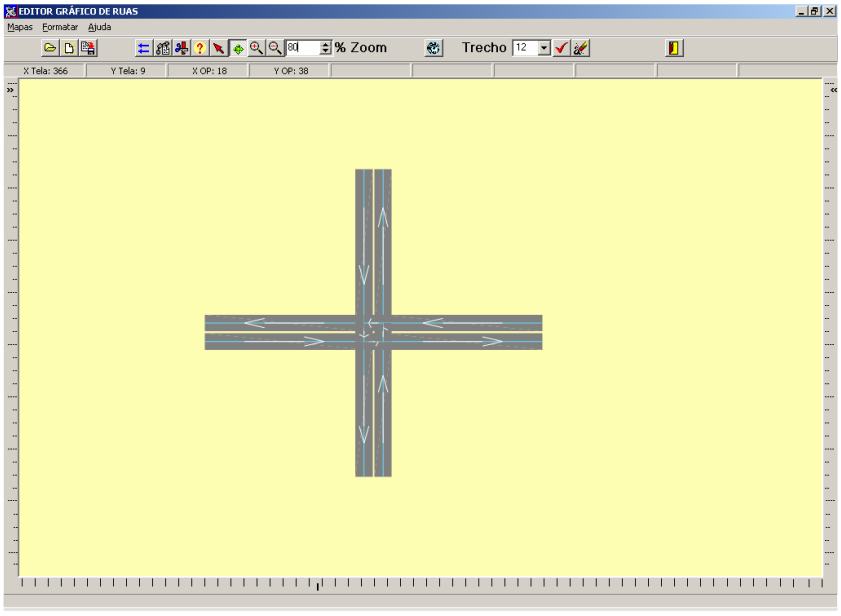


Figura 27 - Mapa depois de 80% de zoom

3.4 RESULTADOS E DISCUSSÃO

Para análise dos resultados obtidos foram confeccionadas cinco malhas hipotéticas, sendo que a quinta está representada na fig. 28. Todas as ferramentas de que o sistema oferece foram utilizadas na confecção da malha. Uma das principais limitações do sistema é a manipulação de trechos duplos ou triplos com trechos simples. A alternativa proposta para solucionar este problema, é a alteração das propriedades destes trechos através do botão “Modificar Trecho”.

A definição do mapa rodoviário referente à fig. 28 encontra-se descrita no quadro 11. Estas informações foram geradas conforme especificação detalhada em Freire (2004).

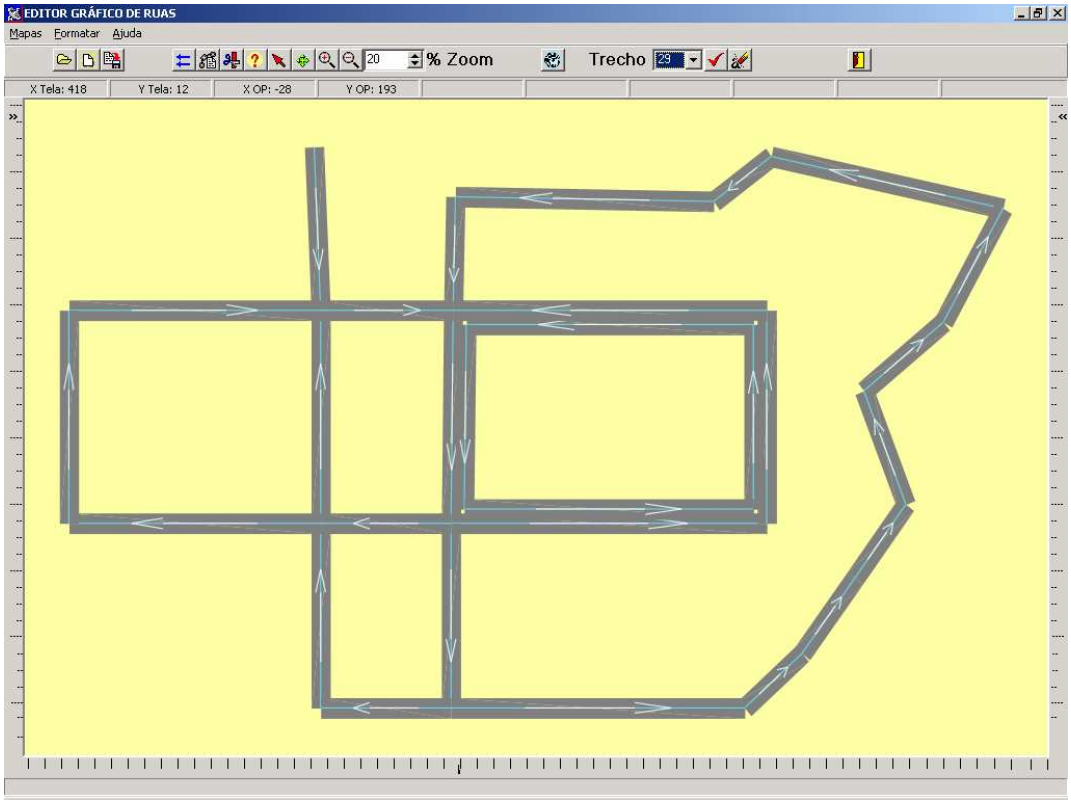


Figura 28 - Malha rodoviária hipotética

Xi	Yi	Xf	Yf	La	Me	Md	Ni	Qe	Vq	Cd
-94	179	91	110	6	0	0	0	0	0	1
-30	158	-31	110	6	0	0	0	0	0	2
105	104	-25	104	6	18	0	0	0	0	3
-91	20	-91	110	6	0	0	0	0	0	4
-205	110	-91	110	6	0	0	0	0	0	5
-31	110	-32	20	6	19	0	0	0	0	6
-91	110	-31	110	6	0	0	0	0	0	7
-32	20	111	20	6	21	0	0	0	0	8
-91	-58	-91	20	6	0	0	0	0	0	9
-91	20	-205	20	6	0	0	0	0	0	10
-32	20	-32	-58	6	0	0	0	0	0	11
-32	20	-91	20	6	0	0	0	0	0	12
-205	20	-205	110	6	0	19	0	0	0	13
-32	-58	-91	-58	6	0	0	0	0	0	14
218	153	113	175	6	0	0	0	0	0	15
191	105	218	153	6	0	0	0	0	0	16
111	20	111	110	6	20	0	0	0	0	17
111	110	-31	110	6	0	3	0	0	0	18
-25	104	-26	26	6	0	6	0	0	0	19
105	26	105	104	6	0	17	0	0	0	20
-26	26	105	26	6	0	8	0	0	0	21
174	28	155	76	6	0	0	0	0	0	22
155	76	191	105	6	0	0	0	0	0	23
101	-58	127	-35	6	0	0	0	0	0	24
151	-3	174	28	6	0	0	0	0	0	25
127	-35	151	-3	6	0	0	0	0	0	26
87	156	-30	158	6	0	0	0	0	0	27
113	175	87	156	6	0	0	0	0	0	28
-32	-58	101	-58	6	0	0	0	0	0	29

Quadro 11 – Representação textual do mapa rodoviário mostrado na figura 28

Os atributos especificados no quadro 11 são (FREIRE, 2004, p. 28-29):

a) Xi: ponto no plano cartesiano que define o inicio da reta pelo eixo X;
b) Yi: ponto no plano cartesiano que define o inicio da reta pelo eixo Y;
c) Xf: ponto no plano cartesiano que define o fim da reta pelo eixo X;
d) Yf: ponto no plano cartesiano que define o fim da reta pelo eixo Y;
e) La: define a largura da reta projetada. Cada reta é considerada um trecho. Um trecho pode ser definido como um pedaço de rua;
f) Me: código do trecho paralelo a esquerda da reta projetada. Se não existir, este código é zerado. A função “Me” é utilizada para mudar de faixa de rolamento. Um exemplo de mudança de faixa de rolamento são as ruas com duas ou mais faixas de rolamento;
g) Md: código do trecho paralelo a direita da reta projetada. Se não existir este código é zerado. A função “Md” é utilizada para mudar de faixa de rolamento, exemplo de mudança de faixa de rolamento são as ruas com duas ou mais faixas de rolamento;

h) Ni: define o nível em que o trecho encontra-se (tabela 1). Um exemplo de nível de trecho são os viadutos ou túneis que traz a sobre-posição de ruas;
i) Qe: define o manuseio do atributo quebra do carro. Se o valor for zero no atributo quebra da malha, o atributo quebra do carro é inicializado com zero, senão o atributo quebra do carro é incrementado com o valor do atributo quebra da malha (tabela 2);
j) Vq: define a permissão de entrada de um carro no trecho. Se o valor quebra do carro for menor ou igual ao valor de quebra da malha, o trecho para o carro é disponibilizado. Caso contrário, o trecho não é disponibilizado para o carro forçando a busca por novo trecho (tabela 2);
k) Cd: define o código do trecho.

Tabela 1 – Define atributo do nível de rua

...	Abaixo da superfície
-2	Abaixo da superfície
-1	Abaixo da superfície
0	superfície
1	Acima da superfície
2	Acima da superfície
...	Acima da superfície

Tabela 2 – Define direção carro no trecho

Quebra	**Valor quebra**	**Quebra <= Valor quebra**	**Direção do carro**	**Ação (Variaveis)**
0	0..X	sim	Qualquer (*random*)	Inicializa com 0
1..X	0..(X–1)	Sim	Qualquer (*random*)	Incrementa valor quebra
X	Y	Não	Trecho com quebra 0	Inicializa com 0

A fig. 29 mostra a simulação da malha rodoviária no simulador do tráfego de automóveis em uma malha rodoviária. Foram utilizados 300 carros à velocidade de 60 quilômetros por hora. Este teste certificou que o objetivo principal foi atendido, ou seja, todas as malhas construídas através deste protótipo, podem ser utilizadas no sistema proposto por Freire (2004).

Recomenda-se que na edição de malhas compostas por mais de 50 trechos, sejam utilizados computadores com no mínimo 512 *Megabytes* de memória e processador superior à um *Gigahertz*.

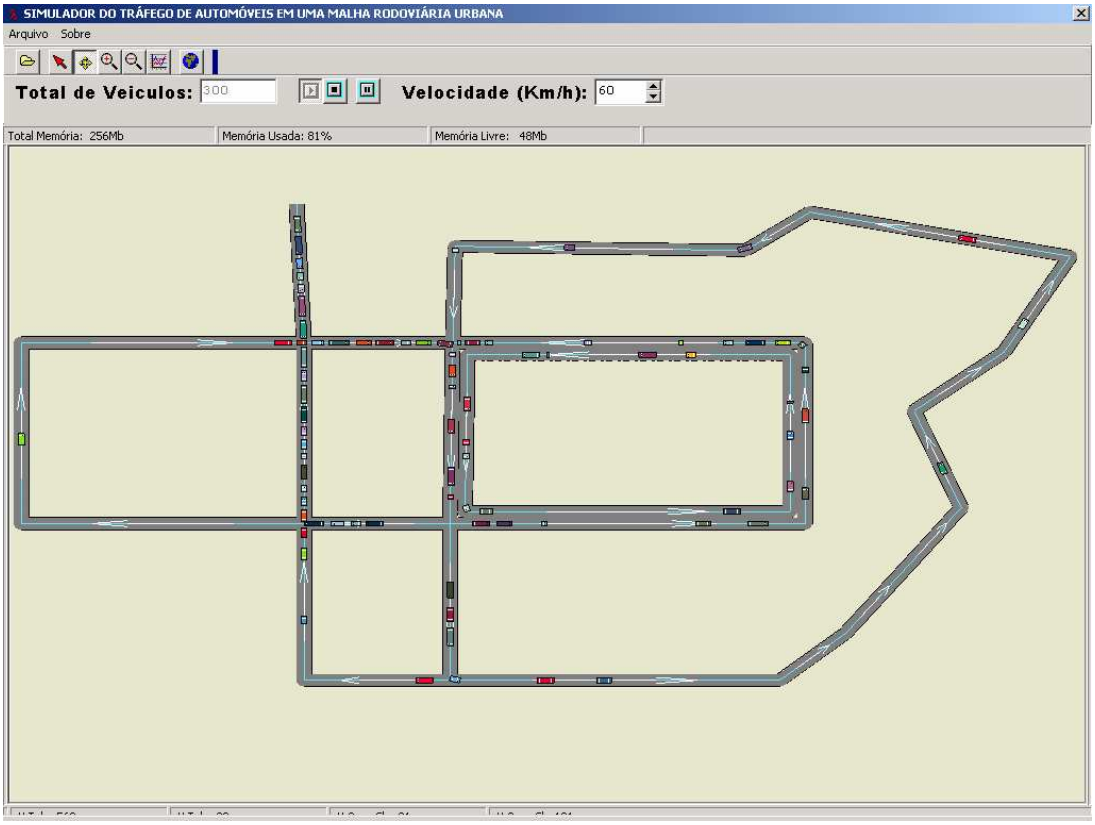


Figura 29 - Mapa da figura 28 no simulador do tráfego de automóveis em uma malha rodoviária

4 CONCLUSÕES

Este trabalho apresentou o desenvolvimento de um editor gráfico de ruas para o sistema de controle de tráfego de automóveis. O ambiente de programação Delphi e a biblioteca gráfica OpenGl foram adequados para o desenvolvimento do protótipo, facilitando principalmente o desenho das primitivas geométricas.

Uma das principais desvantagens do uso da biblioteca OpenGl é a performance da aplicação à partir de um determinado número de trechos criados na malha, necessitando alguns requisitos mínimos de *hardware*, já sugeridos para a utilização adequada da aplicação. Isto acontece porque as trechos são todos recalculados e redesenhados na tela à cada alteração feita pelo usuário.

O protótipo desenvolvido utiliza-se do *mouse* como principal ferramenta de trabalho, tornando-o de fácil entendimento e manuseio. O usuário pode confeccionar o mapa da malha viária através da criação de trechos individuais simples, informando uma série de configurações, como largura da via, mão de direção, direções que um carro pode seguir, entre outros.

As informações geradas pelo editor, quando salvas, são guardadas em arquivo texto compatível com o software desenvolvido por Jocemar J. Freire (FREIRE, 2004), visto que este trabalho trata-se de uma sugestão do trabalho desenvolvido pelo autor acima citado. A integração de ambos os softwares à partir do arquivo gerado funcionou.

O grande desafio do trabalho foi compreender inicialmente a biblioteca OpenGl para poder manipular as figuras matematicamente utilizando conceitos de computação gráfica. A função ponto em polígono tornou-se extremamente vital para o protótipo, pela necessidade de localização do trecho no plano, pois a maioria das operações realizadas na aplicação em questão necessitam desta função.

A principal vantagem do protótipo é a facilidade de manipulação da malha viária, sem que o usuário precise intervir manualmente no arquivo texto, gerando economia de tempo gasto para executar a confecção de qualquer malha.

A principal desvantagem do protótipo é que este não funciona adequadamente quando são inseridos trechos duplos ou maiores juntamente com trechos simples. Para solucionar este problema, uma alteração explícita das propriedades do trecho deve ser feita.

4.1 EXTENSÕES

Nesta seção são apresentadas sugestões de extensões para novos trabalhos a serem realizados a partir deste, os quais são:

a) desenvolvimento de função para geração de trechos duplos ou triplos à partir de uma malha com trechos simples, facilitando ao usuário simular ampliações na malha viária;
b) desenvolvimento de função que permita ao usuário o desenho da malha viária à partir de uma linha livre, ficando à cargo do sistema a transformação destas linhas em trechos retos (isto é possível através da utilização do algoritmo de Reumann-Witkam, segundo Queiroz (2004));
c) desenvolvimento de ferramenta para seleção de partes do desenho, objetivando a modificação das partes selecionadas ou a utilização da função copiar;
d) desenvolvimento da função copiar/colar, possibitando ao usuário simplificar a criação de malha viária à partir de trechos semelhantes;
e) desenvolvimento de função para criação de trechos à partir de informações como comprimento do trecho e angulação do trecho adjacente;
f) desenvolvimento de função para impressão de mapas ou apenas trechos selecionados;
g) desenvolvimento de função para “desfazer alteração”, para que o usuário retorne rapidamente ao cenário anterior do desenho sem precisar reeditar a malha viária;
h) desenvolvimento de função para tratar graficamente trechos curvilínineos, permitindo ao usuário viasualizar a malha como ela é no mundo real;
i) desenvolvimento de função para não permitir a sobreposição dos trechos, impedindo ao usuário sobrepor as figuras;
j) desenvolvimento de função para identificação dos trechos da malha viária, facilitando ao usuário a localização de determinadas ruas, como nome e número;
k) criar funções para a visualização em três dimensões (3D). A simulação atual do protótipo é em duas dimensões;

l) criar cenários virtuais para que o usuário possa andar pelas ruas, podendo desta forma ter a noção exata de distâncias, angulações etc.

m) integrar o sistema à tecnologia GPS, auxiliando o usuário a desenhar com precisão toda a malha viária.

REFERÊNCIAS BIBLIOGRÁFICAS

ABRAMCET. **Associação brasileira de monitoramento e controle eletrônico de trânsito.** São Paulo, [2000]. Disponível em: <http://www.abramcet.com.br/home/old/historico.shtml>. Acesso em: 18 out. 2004.

BANON, Gerald Jean Francis. **Bases da computação gráfica**. Rio de Janeiro: Campus, 1989.

CANTÚ, Marco. **Dominando o Delphi 7**: a bíblia. São Paulo: Makron Books, 2003. 896 p.

DOMINGUES, Hygino H. **Surgimento da geometria analítica.** Rio Grande do Sul, [2003]. Disponível em: <http://www.somatematica.com.br/historia/analitica.php>. Acesso em: 22 dez. 2004.

FIGUEIREDO, Luiz Henrique de; PINTO, Paulo César. **Introdução a geometria computacional.** Rio de Janeiro: IMPA, 1991. 111 p.

FOLEY, James D. et al. **Computer graphics:** principles and practice**.** Massachusetts: Addison-Wesley Publishing Company, 1990. 1174 p.

FREIRE, Jocemar José. **Simulação do controle de tráfego de automóveis em uma malha rodoviária urbana**. 2004. 47 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

MAYER & BUNGE Informática. **Projeto de interfaces gráficas de usuário na prática. [**São Paulo], 1997. 164 p.

PERSIANO, Ronaldo César Marinho; OLIVEIRA, Antonio Alberto Fernandes de. **Introdução a computação gráfica.** Rio de Janeiro: Livros Técnicos e Científicos Editora Ltda, 1989. 225 p.

QUEIROZ, Gilberto Ribeiro. **Algoritmos geométricos para bancos de dados geográficos:** da teoria à prática na terralib. 2004. 147 f. Dissertação (Mestrado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos.

SCHREIDER JUNIOR, Sideney A. **Sistema de apoio à decisão para o controle de tráfego em interseções isoladas.** 2001. 9 f. Projeto Final de Graduação (Engenharia Civil) – Centro de Formação de Recursos Humanos em Transportes, Universidade de Brasília, Brasília.DOMINGUES, Hygino H. **Surgimento da geometria analítica.** Rio Grande do Sul, [2003]. Disponível em: <http://www.somatematica.com.br/historia/analitica.php>. Acesso em: 22 dez. 2004.

SILVA, Cláudio Xavier da; BARRETO FILHO, Benigno. **Toda matemática.** São Paulo: Ática, 1990. 374 p.

SINCMOBIL. **Sistema de informação e controle para mobilidade urbana.** Florianópolis, [2003]. Disponível em: < http://www.das.ufsc.br/sincmobil/ >. Acesso em: 11 ago. 2004.

STRALEY, Steve. **Tutorial on using Enterprise Architect.** Atlanta, 2002. Disponível em: <http://www.sparxsystems.com.au/userdocs/Application20Development20with%20EA.pdf >. Acesso em: 02 fev. 2005.

WANGENHEIM, Aldo Von. **Tutorial de OpenGl.** Florianópolis, [2004]. Disponível em: <http://www.inf.ufsc.br/~awangenh/CG/opengl.html>. Acesso em: 11 ago. 2004.

WOO, Mason; NEIDER, Jackie; DAVIS, Tom. **OpenGL arquitectural review board:** OpenGL programming guide – the official guide to learning OpenGL. Boston: Addison-Wesley Publishing Company, 1997. 730 p.