TRABALHO PARA A DISCIPLINA DE PROGRAMAÇÃO AVANÇADA DO CURSO DE MESTRADO EM COMPUTAÇÃO APLICADA DA UTFPR: *CTA Simulator*

Edwins Leonardi Padilha
edwins.leonardi@gmail.com

Disciplina: **Programação Avançada** – Prof. Dr. Jean M. Simão
Departamento Acadêmico de Eletrônica – DAELN - Campus de Curitiba
Universidade Tecnológica Federal do Paraná - UTFPR
Avenida Sete de Setembro, 3165 - Curitiba/PR, Brasil - CEP 80230-901

Resumo – Este documento apresenta as técnicas, paradigmas de programação, elementos, ferramentas, conceitos, padrões de projeto e tecnólogias utilizados na implementação do simulador de Trânsito, chamado CTA Simulator. CTA Simulator é um simulador de trânsito, que simula o controle de fluxo de veículos de determinada região, através do uso de semáforos de trânsito. Os objetivos principais deste projeto são: o desenvolvimento de estratégias para o controle dos semáforos, a simulação destas estratégias, a avaliação de desempenho das diferentes estratégias utilizadas, a criação de uma interface gráfica, utilizada para ilustrar a simulação e por último a análise de desempenho e de complexidade do desenvolvimento do simulador utilizando a linguagem C++ orientada a objetos. O levantamento de requisitos do simulador foi feito textualmente através de um documento de CONOPS e o design e modelagem foi feito graficamente através da UML com a utilização de diagramas de componentes e de classe. A implementação do código do simulador em C++ possibilitou a prática de vários assuntos discutidos em aula como: codificação orientada a objetos, coesão e desacoplamento de classes, polimorfismo, sobrecarga de métodos, threads, mutex, Standard Template Library (STL), templates, padrões de projeto (Observer/Singleton), herança múltipla, manipulação de arquivos, relacionamento entre objetos e listas encadeadas.

Palavras-chave ou Expressões-chave: Simulador de Trânsito, estratégias de controle de semáforo, paradigmas de programação na construção de simuladores.

Abstract - This document presents the techniques, programming paradigms, elements, tools, concepts, design patterns and technologies that were used to implement de traffic simulator, called CTA. CTA Simulator is a traffic simulator that aims to simulate the vehicle flow control of a given region through the use of traffic lights. The main objectives of this project are: the development of traffic lights control strategies, the simulation of those strategies, the performance assessment of each chosen strategy, the development of a graphical interface to illustrate the simulation and lastly the performance and development complexity analysis of the simulator using the C++ language in an Object Oriented approach. The simulator requirements gathering were done textually in a CONOPS document and the design was done graphically using components and classes UML diagrams. The implementation code in C++ provided a practical practice of many subjects covered in class like: object oriented coding, cohesion and decoupling of classes, polymorphism, methods overloading, threads, mutex, Standard Template Library (STL), templates, design patterns(Observer/Singleton), multiple inheritance, files handling,, concurrent programming, relations between classes and chained lists.

Key-words or Key-expressions: Traffic simulation, traffic lights control strategy, programming paradigms in the development of simulators.

INTRODUÇÃO

O projeto do simulador de trânsito "CTA simulator", foi desenvolvido para a matéria de Programação Avançada do programa de graduação em computação aplicada da UTFPR.

O código desenvolvido neste trabalho contempla primeiramente o desenvolvimento do simulador em linguagem de programação C++ em um paradigma de programação orientado a objetos (POO). A escolha deste paradigma de programação serve como avaliação dos conceitos vistos na disciplina de programação avançada. Posteriormente o simulador também será desenvolvido utilizando o paradigma de programação orientado a notificações (PON), e então serão comparados os resultados das duas implementações.

O desenvolvimento do simulador se deu pela utilização do ciclo clássico da engenharia de software em cascata, onde cada ciclo do processo é feito separadamente. Iniciou-se pelo levantamento de requisitos do sistema através da elaboração do documento de CONOPS (Conceitos de Operações), este documento visa descrever as caracteristicas do sistema do ponto de vista dos operadores do sistema. Após o levantamento de requisitos foi feito o design da implementação do simulador utilizando os diagramas de componentes e de classes da UML. Seguindo o design foi feita a implementação do sistema utilizando a linguagem C++ na ferramenta de desenvolvimento Visual Studio 2010 Express. O último ciclo do desenvolvimento do simulador foi a fase de testes, feita através da análise dos arquivos de log gerados pelo simulador.

Nas seções a seguir, serão apresentadas as funcionalidades e requisitos esperados do simulador, o desenvolvimento do simulador orientado à objetos, a tabela de conceitos da disciplina que foram ou não utilizados no desenvolvimento do simulador, conclusões e referências bibliográficas utilizadas.

EXPLICAÇÃO DO SIMULADOR EM SI

O CTA Simulator simulará o trânsito de veículos em uma determinada região urbana e o controle de semáforos utilizado para organizar este trânsito.

A região simulada será composta por dez ruas horizontais e dez ruas verticais, ao total de vinte ruas. Cada uma das ruas na horizontal, faz interseção com outras dez ruas na vertical e vice-versa, formando assim uma matriz de 10x10 interseções, como mostra a figura 1. Assim cada rua tem dez interseções.

Cada seção da rua, presente entre duas interseções é chamada de quadra. Assim sendo, cada rua da simulação é composta por 10 quadras.

Em cada interseção da simulação existe um semáforo de trânsito, utilizado para controlar o fluxo de veículos. Cada semáforo é composto de 2 sinais de trânsito, um em cada rua da interseção. Cada sinal de trânsito possui três luzes de sinalização nas cores: vermelho, verde e amarelo. Estas cores correspondem ao estado atual do sinal de trânsito, então em dado momento, apenas uma das três cores está ativa(acesa). O estado representado pelas cores do sinal são as seguintes:

- Vermelho: os veículos devem permacer parados.
- Verde: os veículos podem seguir adiante.
- Amarelo: antecede a transição do estado verde para vermelho, serve como alerta.

O semáforo deve garantir que nenhum de seus dois sinais de trânsito estejam no estado verde simultaneamente, esta é a restrição fundamental de segurança dos semáforos e a razão dos mesmos existirem.

Cada rua possui de 1 a 4 pistas. Uma pista é um trecho da rua com largura suficiente para acomodar um veículo. Cada quadra possui o comprimento de 100 metros. Para facilitar a implementação do simulador, o tamanho dos veículos será fixado em 4 metros. Assim, cada quadra do simulador tem capacidade de conter 25 veículos por pista, ou seja, uma quadra que possui 2 pistas suportará no máximo 50 veículos simultaneamente.

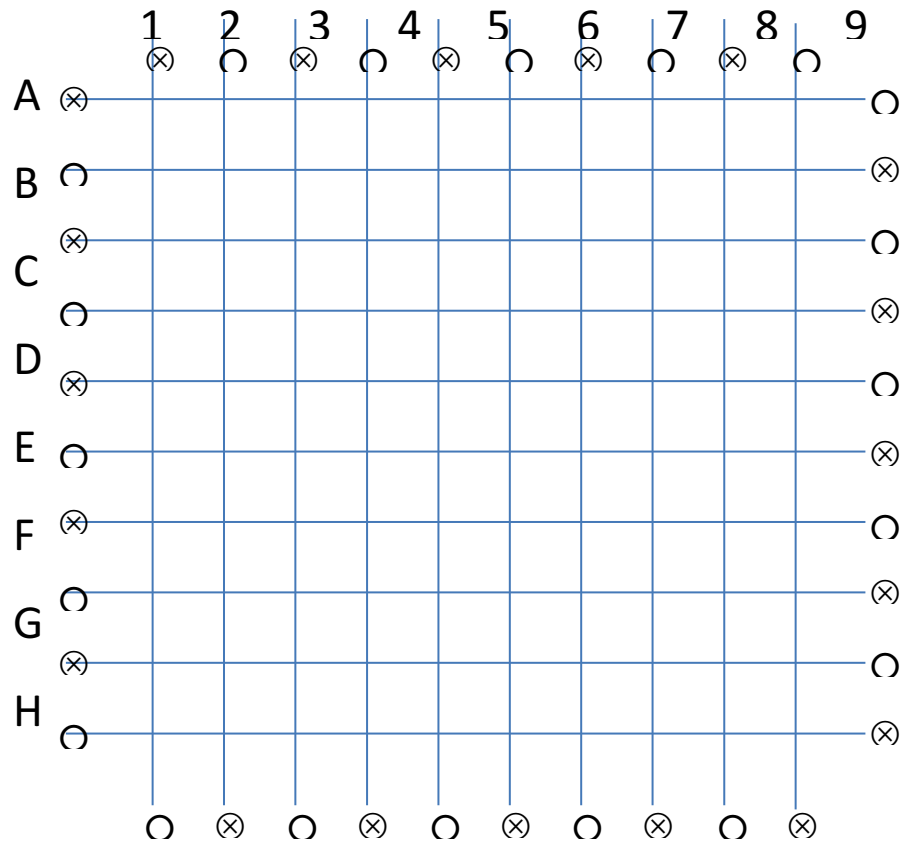

Figura 1. Região simulada, 10 x 10 interseções.

Cada rua possui ainda um ponto de entrada, representado na figura 1 pelo círculo e um ponto de saída, representado pelo círculo com um X dentro. Os veículos são criados no ponto de entrada, de forma constante, através de um número randômico no intervalo de 0.1, 0.2, 0.3, 0.4 ou 0.5 veículos por segundo.

Apartir do momento que os veículos são criados no ponto de entrada, eles podem se movimentar pelas quadras das ruas onde foram criados. Os veículos têm velocidade constante de 20 metros por segundo. Por consequência, um veículo leva no minimo 5 segundos para conseguir atravessar uma quadra, que tem o tamanho 100 metros.

Alguns veículos em uma quadra, acabam fazendo uma curva no fim de uma interseção, entrando na quadra da rua perpendicular a rua atual. Este número de veículos em cada quadra que acaba virando na interseção é uma porcentagem gerada randomicamente com os valores entre 5% à 35%. Digamos que uma quadra tenha essa porcentagem igual a 10% e no total tenha 20 veículos se movimentando, então 2 veículos (10% do total) irão virar na próxima interseção.

O controle dos semáforos é feito por uma entidade chamada *controlador.* O controlador tem conhecimento de todos os semáforos da simulação e os mantém sincronizados. É o controlador que coordenada a mudança de estado(cor do sinal) dos semáforos. Nesta simulação o controlador pode utilizar três estrátegias distintas de controle de semáforos:

- Controle independente: cada semáforo cumpre rigorosamente o seu ciclo de tempo para mundança de estado sem considerar estado de quadras e semáforos vizinhos
- Controle baseado em congestionamento: quando uma quadra atinge um limite determinado de veículos, o tempo do seu sinal vermelho é reduzido através de um ajuste com o semáforo vizinho da interseção.

- Controle baseado em tráfego facilitado: também chamado de “onda verde”, parte do principio de sincronizar o sinal verde de um sinal, de acordo com o sinal da quadra anterior na mesma rua, de uma forma que um veículo passando por um destes sinais abertos consiga também passar pelos sinais subsequentes na mesma rua também abertos, sem parar em nenhum semáforo.

Cada quadra possui sensores de veículo que monitoram a quantidade de veículos presentes. Quando o limite de 60% ou 100% é atingido, estes sensores notificam o controlador que então pode vir a mudar o estado dos semáforos afim de otimizar o tráfego.

O simulador terá o tempo de simulação igual a 2000 segundos, o que corresponde a algo em torno de 33 minutos. O resultado do processamento será disponibilizado em um arquivo de log.

O arquivo de log, capta um número limitado de eventos do simulador, e não toda a simulação em si. Esta captação baseada em eventos pré determinados é feita com o intuito de reduzir o tamanho do arquivo de log. Para uma simulação de 2000 segundos o tamanho do arquivo de log é de mais ou menos 10 MB. Os eventos captados pelo log são os seguintes: o veículo entra em uma quadra, o veículo entra no ponto de entrada, o veículo entra no ponto de saída, e o veículo se movimenta (muda sua posição).

A última parte do CTA simulator é a criação de uma interface gráfica, chamada de *player,* que irá ler o arquivo de log gerado pelo simulador e irá reproduzir os passos executados no simulador na tela gráfica, oferecendo assim uma melhor análise da performance oferecida por cada uma das estratégias de controle de semáforos simuladas.

DESENVOLVIMENTO DO SIMULADOR NA VERSÃO ORIENTADA A OBJETOS

O desenvolvimento do jogo pode ser dividido logicamente em três componentes distintos: simulador, controlador e player. A **figura 2** mostra estes três componentes, respectivamente como *Traffic Simulator, Controller e Simulator Player*.

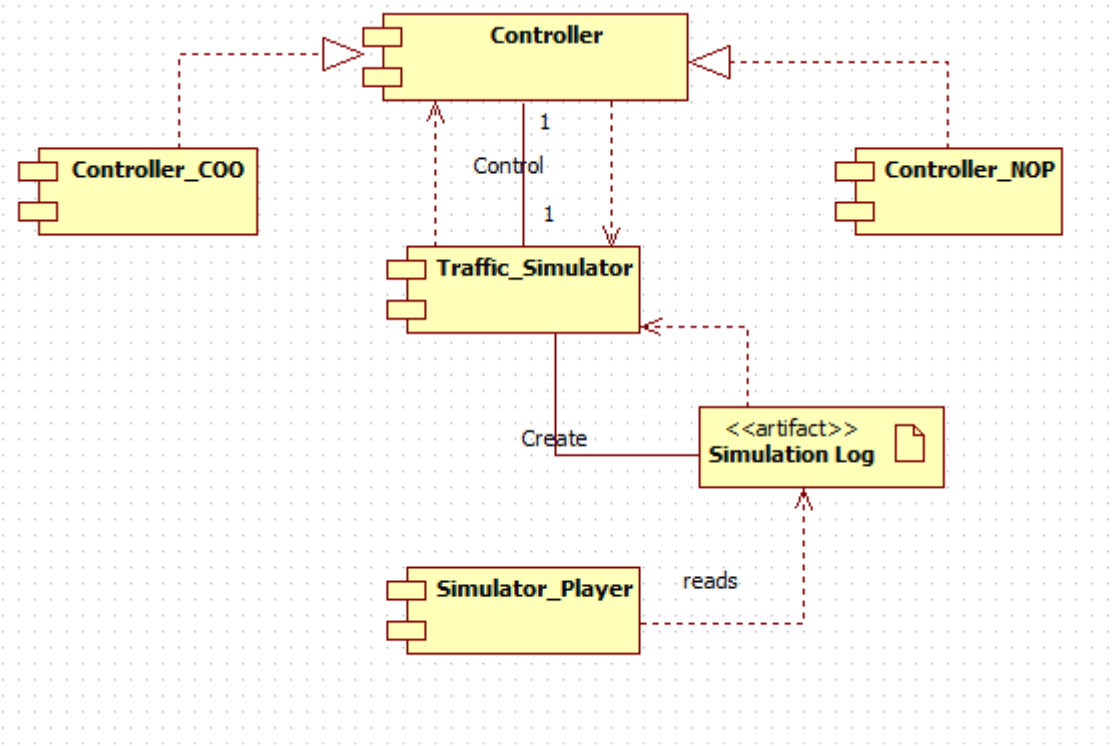

Figura 2. Diagrama de componentes da simulação

No componente *Traffic Simulator*, é onde ocorre a simulação das ruas, quadras, interseções, semáforos e veículos em si. Neste componente, as entidades descritas acima são criadas e se relacionam entre si afim de simular o tráfego de veículos em determinada região urbana e o impacto que os semáforos têm neste fluxo de veículos, estudo e motivação principal da criação deste simulador.

O controle dos semáforos é feito externamente ao componente *Traffic Simulator*, e é definido dentro do componente *Controller* ou controlador. O controlador tem conhecimento de todos os semáforos da simulação e é configurado para utilizar diferentes estratégias de controle para a sincronia dos semáforos. O requisito fundamental do controlador é prover a segurança do fluxo de veículos, não permitindo que em dado semáforo, ambos os sinais de trânsito estejam abertos (cor verde) ao mesmo tempo. Adicionalmente os controladores devem prover otimização do fluxo de veículos utilizando estratégias de controle que propiciem esta otimização.

O componente *Simulator_Player*, é responsável apenas por reproduzir graficamente os resultados de uma simulação previamente executada. O player deve reproduzir fielmente o resultado da simulação, lendo o arquivo log gerado pelo componente *Traffic Simulator.* Futuramente talvez seja necessário que o player simule também o estado dos semáforos de trânsito, o que demandaria a leitura de um arquivo de log a ser gerado pelo componente *Controller.*

Tabela 1. Lista de Requisitos e suas Situações .

Requisitos Funcionais	**Situação**
Simulador de Trânsito.	Requisito previsto inicialmente e realizado
Simular uso de sensores de veículos.	Requisito previsto inicialmente e realizado parcialmente – falta ainda a comunicação com o controlador.
Simulador de controlador de semáforos.	Requisito previsto inicialmente e realizado.
Simulador apto a receber diferentes estratégias de controle	Requisito previsto posteriormente e realizado
Simular tráfego de veículos	Requisito previsto inicialmente e realizado
Simular área de atuação, com ruas, quadras e interseções.	Requisito previsto inicialmente e realizado.
Simular a intensidade de tráfego de veículos.	Requisito previsto inicialmente e realizado.
Criar log com o resultado da simulação	Requisito previsto inicialmente e realizado.
Calcular velocidade média que os veículos atingiram na simulação	Requisito parcialmente implementado. Veículos individualmente já calculam a velocidade média, falta o cálculo geral da simulação, assim como definir onde estes dados serão mostrados.
Criar interface gráfica para reprodução gráfica da simulação.	Requisito não implementado
Estratégia de controle de semáforos independente.	Requisito previsto inicialmente e realizado.
Estratégia de controle de semáforos baseada em tráfego	Requisito não implementado.

Estratégia de controle baseado em tráfego facilitado.	Requisito não implementado.
Criar simulador e controlador de forma independente e desacoplado.	Requisito implementado.

A **figura 3** apresenta os diagramas de classe do componente *Controller*.

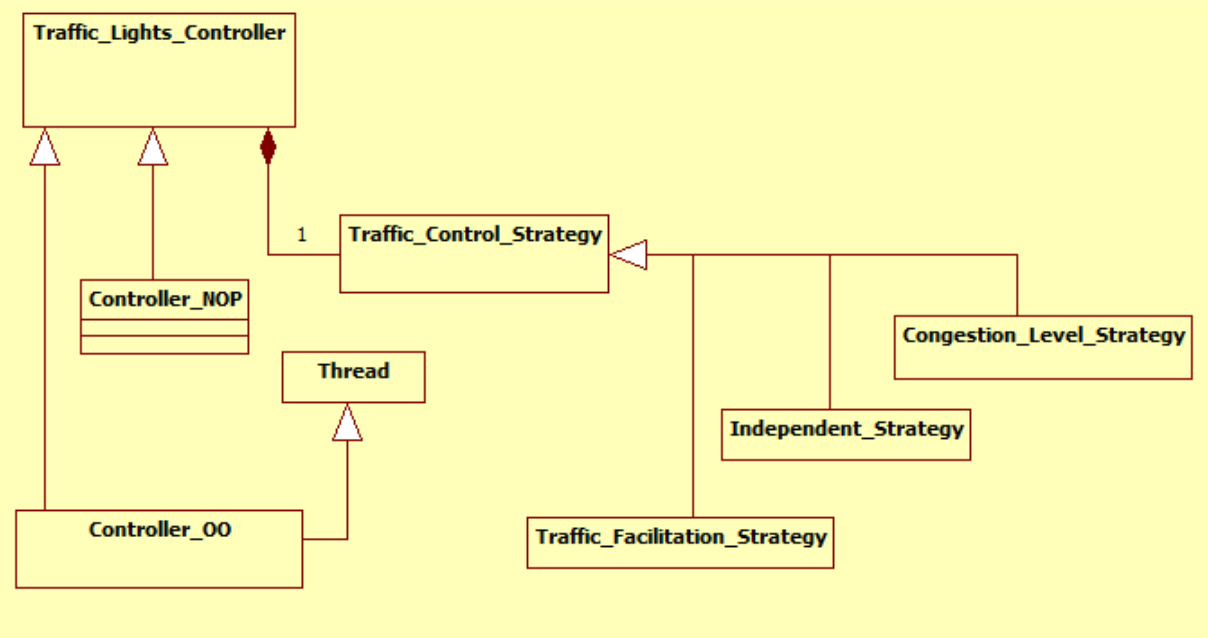

Figura 3. Diagrama de classes do componente Controller

A **figura 4** apresenta os diagramas de classe do componente *Traffic Simulator*.

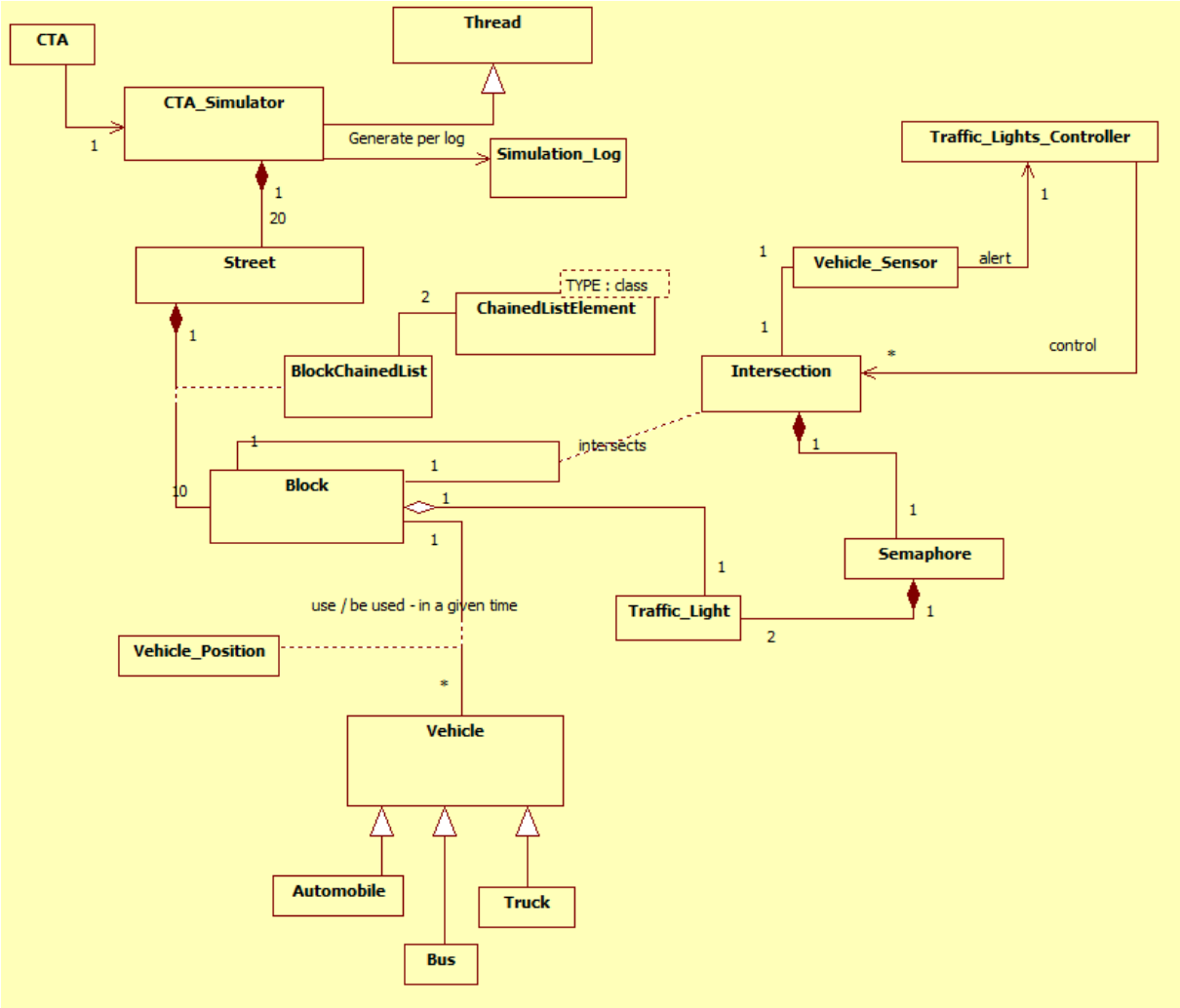

Figura 4. Diagrama de classes do componente Traffic Simulator.

É importante salientar como foi divido a implementação dos 3 diferentes componentes do simulador CTA.

O módulo player que contém a interface gráfica, depende apenas do arquivo de log gerado pelo simulador para funcionar, então é possível implementa-lo em um projeto(módulo) a parte dos demais componentes do sistema. Os pré-requisitos para seu funcionamento são a configuração do caminho de arquivo para o log gerado pelo simulador e a existência deste arquivo.

Os demais módulos, Controller e Traffic Simulator, foram criados inicialmente dentro do mesmo projeto C++. A divisão lógica da execução destes dois componentes ocorre através do uso de Threads. Cada um dos componentes, representados respectivamente pelas classes *Controller_OO* e *CTA_Simulator*, são duas threads que executam paralelamente e concorrentemente.

Vale destacar que o uso de threads para o controlador e para o simulador, foi importante para simular o tempo real da simulação. Estas threads executam cada passo da simulação a cada um segundo, e este controle de tempo é feito pelo método sleep das threads. Foi também necessário um sincronismo entre as duas threads para evitar que um thread avance na simulação antes que a outra thread tenha executado o passo (segundo) da simulação, ou seja, as duas threads devem simular o tempo de forma sincrona. Para o sincronismo da duas threads foi criado um objeto Mutex que garante turnos de execução para as threads, evitando que a mesma thread execute por duas vezes seguidas. Quando a mesma thread tenta executar por

duas vezes seguidas, ela acaba cedendo o processador a outras threads através da chamada de função yield.

A biblioteca de threads utilizadas no projeto foi o Posix.

TABELA DE CONCEITOS UTILIZADOS E NÃO UTILIZADOS

Tabela 2. Lista de Conceitos Utilizados e Não Utilizados no Trabalho.

N.	Conceitos	Uso	Onde
1	**Elementares:**		
	- Classes, objetos,	Sim	Todos .h e .cpp
	- Atributos (privados), variáveis e constantes	Sim	Todos .h e .cpp
	- Métodos (com e sem retorno).	Sim	Todos .h e .cpp
	- Métodos (com retorno *const* e parâmetro *const*).	Não	Nenhum
	- Métodos recursivos.	Não	Nenhum
	- Construtores (sem/com parâmetros) e destrutores	Sim	Todos .h e .cpp
	- Classe Principal.	Sim	CTA.cpp & CTA_Simulator.h/.cpp Controller_OO.h/.cpp
	- Divisão em .h e .cpp.	Sim	No projeto.
2	**Relações de:**		
	- Associação	Sim	CTA.cpp BlockChainedList.h Vehicle.h Block.h Vehicle_Sensor.h Intersection.h
	- Agregação via associação	Sim	Vehicle_Position.h Block.h Traffic_Light.h
	- Agregação propriamente dita.	Sim	CTA_Simulator.h Street.h Semaphore.h Traffic_Light.h Intersection.h Semaphore.h Traffic_Lights_Controller.h Traffic_Control_Strategy.h
	- Herança elementar.	Sim	CTA_Simulator.h Automobile.h Truck.h Bus.h Traffic_Facilitation_Strategy.h Independent_Strategy.h Congestion_Level_Strategy.h
	- Herança em diversos níveis.	Não	
	- Herança múltipla.	Sim	Controller_OO
3	**Ponteiros e generalizações:**		

	- Operador *this*	Sim	Street.cpp Automobile.cpp Block.cpp Vehicle.cpp
	- Alocação de memória (*new* & *delete*)	Sim	Block.cpp BlockChainedList.cpp CTA.cpp Street.cpp
	- Listas Encadeadas (bem Formadas)	Sim	BlockChainedList.cpp
	- Gabaritos/*Templates* criada/adaptados pelos autores	Sim	ChainedListElement.h
	Persistência de Objetos		
	- Texto via Arquivos de Fluxo	Sim	CTA_Simulator.cpp
	- Binário	Não	
4	**Sobrecarga de:**		
	- Construtoras e Métodos.	Sim	CTA_Simulator.h/.cpp Vehicle_Position.h/.cpp
	- Operadores (2 tipos de operadores pelo menos)	Sim	Street.h/.cpp
	Virtualidade:		
	- Métodos Virtuais.	Sim	Thread.h Traffic_Control_Strategy.h
	- Polimorfismo	Sim	Automobile.h Independent_Strategy.h
	- Métodos Virtuais Puros / Classes Abstratas	Sim	Traffic_Control_Strategy.h Vehicle.h
5	**Engenharia de Software**		
	- Levantamento de Requisitos Textualmente - E/ou via Diagrama de Casos de Uso em *UML*	Sim	CONOPS
	- Diagrama de Classes em *UML*	Sim	Todas as classes
	- Diagrama de Atividades em *UML* - E/ou Fluxograma.	Não	
	- Outros diagramas em *UML*, Diag. de Estados, Diag. de Seqüência, Diag. de Pacotes etc. - E/ou outros diagramas estabelecidos, como Diag. de Fluxo de Dados (DFD) ou Diag. em SysML (Diag. de Requisitos,. de Blocos etc).	Sim	Diagrama de componentes
6	**Biblioteca Gráfica**		
	- Funcionalidades Elementares.	Não	
	- Funcionalidades Avançadas como:	Não	
	Obs.: especificar quais funcionalidades.		
	Interdisciplinaridades por meio da utilização de Conceitos de Matemática, Física etc		
	- Ensino Médio*	Sim	Vehicle.h/.cpp
	- Ensino Superior*	Não	
	**Obs.: especificar quais Conceitos*		
7	**Organizadores:**		
	Espaço de Nomes (*Namespace*) criada pelos autores.	Não	
	Classes aninhadas.	Não	
	Estáticos e String:		
	Atributos estáticos e chamadas estáticas de métodos.	Sim	CTA_Simulator.h Block.h Helper.h Thread.h Vehicle.h

	A classe Pré-definida String.Conceito.	Sim	Block.h CTA_Simulator.h
8	**Standard Template Library *(STL)***		
	Vector da STL (p/ objetos ou ponteiros de objetos de classes definidos pelos autores).	Sim	Block.h/.cpp
	List da STL (p/ objetos ou ponteiros de objetos de classes definidos pelos autores).	Sim	CTA_Simulator.h/.cpp Independent_Strategy.h/.cpp Street.h/.cpp Traffic_Control_Strategy.h/.cpp
	*Pilhas, Filas, Bifilas, Filas de Prioridade, Conjuntos, Multi-Conjuntos, Mapas **ou** Multi-Mapas*.*	*Não*	
	*Obs.: Listar apenas os utilizados		
9	**Programação orientada a eventos e visual:**		
	*Objetos gráficos como formulários, botões etc**	*Não*	
	*Obs.: Listar apenas os utilizados		
10	**Programação concorrente:**		
	*Linhas de Execução (*Threads*) utilizando* Posix, C-Run-Time ***ou*** Win32API.	*Sim*	*Thread.h/.cpp* CTA_Simulator.h/.cpp Controler_OO.h/.cpp
	Mutex, Semáforos, **ou** Troca de mensagens.	Sim	Mutex.h/.cpp CTA_Simulator.h/.cpp Controler_OO.h/.cpp
	Threads *no âmbito da Orientação a Objetos*		*Thread.h/.cpp*

Tabela 3. Lista de Justificativas para Conceitos Utiliza e Não Utilizados no Trabalho.

No.	Situação
1	Os conceitos não utilizados no trabalho não se mostraram substanciais a implementação, como por exemplo o uso de métodos constantes, recursividade e classes aninhadas. Entretanto muito destes recursos poderiam ser inseridos na implementação sem maiores problemas
2	A interface gráfica do componente *player* será futuramente planejada e implementada

DISCUSSÃO E CONCLUSÕES

A implementação do simulador de trânsito CTA, em linguagem de programação C++ e utilizando o paradigma de programação orientado a objetos, propiciou um grande aprendizado, com a fixação dos conteúdos abordados em sala de aula através do uso prático da linguagem.

O levantamento de requisitos e design do simulador fez com fosse necessária um pesquisa referente a notação UML para a digramas de componentes e de classe, possibilitando um melhor entendimento dos relacionamentos entre objetos, associações, agregações e composições. Os design patterns abordados em classe também possibilitaram considerações importantes na hora da criação dos objetos e dois destes padrões foram utilizados: Observer e Singleton.

Foi utilizada uma rica gama de recursos da linguagem C++, o que para mim foi muito valioso, pois não tinha experiência com a linguagem.

A enfatização dos bons principios de codificação orientada a objetos, como coesão e desacoplamento causou a demanda de um tempo maior na elaboração do design do simulador. Em contrapartida, este tempo adicional na etapa de design, compensou em um tempo menor de desenvolvimento, pois mudanças feitas já na fase de desenvolvimento não causaram uma mudança tão impactante no código, pela forma desacoplada na qual o simulador foi projetado.

Por último, destaco também temas avançados vistos na disciplina, que pude utilizar no trabalho, como o uso de threads e de programação concorrente.

Fica como desafio a extensão do simulador de forma distribuída com o Traffic Simulator e o Controller executando em máquinas diferentes.

REFERÊNCIAS UTILIZADAS NO DESENVOLVIMENTO

[A] PAGE-JONES, MEILIR. Fundamentals of Object-Oriented Design in UML. Indianopolis, IN : v.1, 2000, p. 107-133.