Animador CTA – Um aplicativo para demonstração gráfica de simuladores de controladores de trânsito

Fabio Negrini
Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial (CPGEI),
Universidade Tecnológica Federal do Paraná (UTFPR) - Av. Sete de Setembro, 3165. Curitiba-PR, Brasil
fabionegrini@alunos.utfpr.edu.br

Resumo— As disciplinas acadêmicas de mestrado relacionadas à tecnologia da informação estão fazendo uso de um simulador de controle de tráfego (CTA) como base para andamento das disciplinas. Visto isso, uma ferramenta que possa apresentar os resultados e apoiar o desenvolvimento se faz necessária. O animador CTA surgiu para atender a este tipo de demanda e o presente documento vem apresentar detalhes técnicos, funcionalidades, documentações e a experiência do desenvolvimento da solução durante sua maturação na disciplina de Estudos especiais em paradigmas de programação (EE2261) junto aos demais alunos que cursaram a disciplina.

I. INTRODUÇÃO

O convite feito pelor prof. Jean Simão para criar um simulador CTA para a disciplina EE2261 causou, num primeiro momento, um certo medo. Rapidamente, porém, este sentimento foi transformado em motivação e desafio. A solicitação inicial de retirar do projeto CTA[3] e transformá-lo em um produto auto suficiente também foi rapidamente convertida em um projeto mais audacioso tanto do ponto de vista arquitetônico quanto do seu resultado gráfico final. A proposta de construir uma solução multiplataforma que recebesse um arquivo de log como interface se tornou o objetivo principal do desenvolvimento. Como resultado final obtivemos um programa que permitiu a criação de um ambiente de simulação onde é possível variar o número de ruas verticais (1 a 10), o número de ruas horizontais (1 a 10), o tamanho de cada quadra (1 a 50 posições), o número de pistas por rua (1 a 5) e a velocidade do carro (1 a 5 posições por evento CLOCK). Além de permitir dois tipos de simulação: simplificada, onde a cada evento CLOCK os automóveis são automaticamente movimentados; e a detalhada onde cada movimento deve ser descrito no arquivo de log.

Neste artigo será falado então dos detalhes técnicos da solução que vão desde as especificações do arquivo de log, passando pela arquitetura utilizada no desenvolvimento, complementando com os dados de arquitetura como modelagem de dados e diagramas de threads. Por fim, mas não menos importante, relatos das experiências durante o desenvolvimento e maturação do produto no desenrolar da disciplina com apoio dos demais alunos que também cursaram.

II. ARQUIVO

Já definido em conversa informal, a alimentação do simulador seria feita por um arquivo de log com eventos dispostos de maneira sequencial. O animador deveria, então, reconhecer os eventos e os apresentar de maneira gráfica em uma janela própria. A partir da primeira versão, definida em conjunto com os professores desta disciplina, foram feitas ao todo nove revisões visando facilitar a interação do arquivo, reduzir informações desnecessárias ou recorrentes e corrigir interpretações incoerentes. O resultado final é apresentado a seguir.

Comportamento do simulador

O simulador inicializará as ruas e quadras conforme evento de inicialização. Irá fazer a leitura dos eventos do arquivo em lotes. Os lotes de movimentos serão definidos pelo evento CLOCK. Todos os eventos que estiverem entre um evento CLOCK e outro serão executados de maneira simultânea simulando que ocorreram no mesmo tempo. Os eventos serão lidos e executados, porém serão validados os erros de simulação abaixo:

- Automóvel entrando em quadra lotada
- Automóvel avança um sinal vermelho
- Conversão proibida
- Colisão
- O simulador terá dois modos de execução que mudarão cujo comportamento será detalhado a seguir.

Modo simplificado

No modo simplificado, os automóveis tentarão avançar uma posição a cada mudança de temo (evento CLOCK) de forma automática, simulando um evento CAR-MOVE para todos os automóveis inseridos na simulação sempre que possível. Neste modo, os automóveis respeitarão os semáforos, portanto não haverá avanço de sinal.

Modo completo

No modo completo, os movimentos de todos os automóveis deverão ser explicitados pelo arquivo de log. O animador respeitará todos os movimentos sempre que possível. Neste modo serão verificadas colisões (um automóvel avançar em uma posição preenchida) e avanços de sinal vermelho.

Definições gerais

O animador CTA tem o objetivo de apresentar graficamente os resultados dos trabalhos desobrigando o desenvolvimento de uma interface gráfica para cada aluno, desta forma cada aluno poderá focar na arquitetura de solução. A interface será através de um arquivo texto no padrão ANSII, onde os campos serão separados pelo caractere "|" (barra vertical). Serão aceitos três tipos de campo:

- Caractere: identificado pela letra C, aceitará os caracteres A-Z; a-z;0-9; " " (espaço); "-"-" (traço)
- Numérico Inteiro: identificado pela letra I, aceitará os caracteres 0-9
- Numérico decimal: identificado pela letra N, aceitará os caracteres 0-9 e terá como separador decimal o caractere "." (ponto)

O primeiro campo do log identificará o evento a ser disparado. Em seguida os demais campos serão os parâmetros aguardados pelo evento. Os parâmetros de cada evento serão descritos em capítulos próprios. Abaixo segue uma lista dos eventos aceitos pelo animador CTA, que serão detalhados um a um em seguida:

- INIT: Inicialização da animação com definições gerais
- CAR-NEW: Novo automóvel a ser adicionado no simulador
- CAR-MOVE: Movimento de automóvel no simulador
- CAR-EXIT: Retirada de um automóvel do simulador
- CLOCK: Uma unidade de tempo
- SEMAPHORE: Mudança do estado de um semáforo
- LOG: Inclusão de mensagem no Log do simulador

Eventos

INIT: Configurações iniciais da animação – É o evento de inicialização do animador. Deverá ocorrer apenas uma vez na primeira linha do arquivo.

Campo	Tipo	Descrição
EVENTO	C	Identificador do evento. Valor aceito: "INIT"
HOR_COUNT	I	Quantidade de ruas horizontais. Valores aceitos: 0-10
VER_COUNT	I	Quantidade de ruas verticais. Valores aceitos: 0-10
BLOCK_SIZE	I	Tamanho de uma rua em blocos
BLOCK_LANES	I	Número de pistas de uma quadra
HOR_DIR	I	Direção da primeira quadra horizontal identificada pelo caractere "A". Valores aceitos: 0 – Da esquerda para a direita 1 – Da direita para a esquerda A definição das demais quadras ocorrerá de forma alternada invertendo a direção a cada rua
VER_DIR	I	Direção da primeira quadra vertical identificada pelo caractere 1, valores aceitos: 0 – De cima para baixo 1 – De baixo para cima A definição das demais quadras ocorrerá de forma alternada invertendo a direção a cada rua
EXEC_TYPE	I	Tipo de execução da simulação, valores aceitos: 0 – Modo Simplificado 1 – Modo completo
CAR_SPEED	I	Velocidade do carro em blocos por intervalo de clock

Tabela 1. Detalhamento dos campos do evento INIT

CAR-NEW: Inclusão de automóvel na animação – O evento CAR-NEW implica na entrada de um automóvel no animador. A entrada do automóvel deverá sempre ocorrer pelas extremidades das ruas conforme seu sentido. Se uma quadra estiver ocupada, o

evento será adicionado a uma fila. Tão logo a quadra libere uma posição, o evento é aplicado e o automóvel é adicionado à simulação.

Campo	Tipo	Descrição
EVENTO	C	Identificador do evento. Valor aceito: "CAR-NEW"
ID	I	Identificador do automóvel (usado na identificação dos demais eventos)
DIRECTION	I	Direção da rua entrada. Valores válidos: 0 – Horizontal 1 – Vertical
STREET	I	Indicador da rua, conforme a direção. Valores válidos: Se CAR-NEW-DIRECTION = 0: 0 a INIT-HOR_COUNT Se CAR-NEW-DIRECTION = 1: 0 a INT-VER_COUNT

Tabela 2. Detalhamento dos campos do evento CAR-NEW

CAR-MOVE: Movimentação do automóvel na animação – O evento CAR-MOVE identifica o que o automóvel deverá se mover para a próxima posição disponível. Este evento será validado antes de ser executado conforme abaixo:

Campo	Tipo	Descrição
EVENTO	C	Identificador do evento. Valor aceito: "CAR-MOVE"
ID	I	Identificador do automóvel (criado no evento CAR-NEW)
CONVERT	I	Informa se o automóvel deverá fazer uma conversão. Valores válidos: 0 – O automóvel deverá seguir em frente 1 – O automóvel deverá fazer uma conversão
COUNT	I	Número de movimentos a percorrer. Campo opcional. Se suprimido, assume-se o valor 1

Tabela 3. Detalhamento dos campos do evento CAR-MOVE

- Se o automóvel está na última posição da última quadra, o automóvel sairá da animação e será descrito no log do animador informando também o tempo total de permanência na animação
- Se a próxima casa está ocupada por outro automóvel, o evento não ocorrerá e será descrito no log que houve uma colisão entre dois automóveis
- Se o automóvel estiver na última posição da quadra e o semáforo estiver vermelho, o movimento ocorrerá e será descrito no log que o automóvel invadiu o sinal vermelho
- Se o campo CONVERT = 1 e o automóvel não estiver em uma intersecção, o movimento não ocorrerá e será descrito no log que o automóvel tentou uma conversão proibida.

CAR-EXIT: Retirada do automóvel da animação – O evento CAR-EXIT implica na saída de um automóvel da simulação.

Campo	Tipo	Descrição
EVENTO	C	Identificador do evento. Valor aceito: "CAR-EXIT"
ID	I	Identificador do automóvel (criado no evento CAR-NEW)

Tabela 4. Detalhamento dos campos do evento CAR-EXIT

Se o carro identificado pelo ID já estiver saído da simulação, o será apresentada uma mensagem de erro no log informando que foi feita a tentativa de saída de um automóvel que já não (ou ainda não) está na simulação. Caso um evento CAR-EXIT seja identificado para um automóvel que ainda não está no final da rua, o evento será executado simulando a saída do automóvel da simulação como se entrasse em um estacionamento. Nestes casos, uma mensagem de aviso será apresentada no log informando que um automóvel saiu da simulação antes do final da rua.

CLOCK: Intervalo de tempo – O evento CLOCK identifica uma unidade de tempo do animador. Todos os eventos contidos entre um evento CLOCK e outro serão interpretados como simultâneos simulando ocorrências em paralelo

Campo	Tipo	Descrição
EVENTO	C	Identificador do evento. Valor aceito: "CLOCK"
COUNT	I	Identifica quantas unidades de tempo serão executadas

Tabela 5. Detalhamento dos campos do evento CLOCK

SEMAPHORE: Atualização de status de semáforo – O evento SEMAPHORE indica a mudança do estado de um semáforo.

Campo	Tipo	Descrição
EVENTO	C	Identificador do evento. Valor aceito: “SEMAPHORE”
HOR_BLOCK	I	Identifica a posição horizontal no semáforo. Valores válidos: 0 a INIT-HOR_COUNT
VER_BLOCK	I	Identifica a posição vertical do semáforo. Valores válidos: 0 a INT-VER_COUNT
POSITION	I	Indica a nova posição do semáforo. Valores válidos: 0 – Vermelho horizontal; vermelho vertical – Próximo Horizontal 1 – Verde horizontal; vermelho vertical 2 – Amarelo horizontal; vermelho vertical 3 – Vermelho horizontal; vermelho vertical – Próximo Vertical 4 – Vermelho horizontal; verde vertical 5 – Vermelho horizontal; amarelo vertical

Tabela 6. Detalhamento dos campos do evento SEMAPHORE

O animador validará se o novo estado é um estado válido, por exemplo: se o semáforo está na posição 1 e passar direto para a posição 3 sem passar pela posição 2. Um erro será descrito no Log informando a mudança incorreta de estado do semáforo.

LOG: Inclusão de mensagens no log do animador – Este evento não faz nenhuma interferência no animador, apenas transfere uma mensagem para o log.

Campo	Tipo	Descrição
EVENTO	C	Identificador do evento. Valor aceito: “LOG”
TYPE	C	Identifica o tipo de mensagem. Valores válidos: I – Informação W – Aviso E – Erro D – Depuração
MESSAGE	C	Mensagem

Tabela 7. Detalhamento dos campos do evento SEMAPHORE

Histórico de Mudanças

Desde sua versão inicial até a versão final entregue, algumas modificações foram necessárias para melhorar e simplificar a utilização do animador.

Versão	Data	Observação
1	03/04/2017	Versão inicial
2	05/04/2014	Alterados parâmetros do evento INIT: Incluído o campo EXEC_TYPE.
3	07/04/2017	Alteração nos status dos semáforos
4	07/04/2017	Alteração nos status dos semáforos
5	17/04/2017	Retirado parâmetro CONVERT_COUNT do evento CAR-NEW
6	20/04/2017	Tamanho de quadras e número de pistas definido dinamicamente pelo evento INIT
7	25/04/2017	Velocidade do carro (blocos por clock) definido no evento INIT.
8	02/05/2017	Evento CAR-NEW não dá erro quando a pista está cheia. Armazena o evento em fila para inclusão tão logo libere espaço
9	11/05/2017	Evento CAR-MOVE – novo parâmetro opcional: quantidade de movimentos

Tabela 8. Histórico de modificação dos documentos

III. ARQUITETURA

Visando atender a maior possibilidade de reutilização em projetos dentro do departamento, decidimos pela linguagem C++. Para atender à demanda multiplataforma, decidiu-se pela plataforma QT para C++ pois está bem madura e possui interface de desenvolvimento amigável nos principais sistemas operacionais[5]. Uma outra motivação para utilizar esta solução foi a possibilidade de migração do projeto CTA original para trabalhar com o animador CTA em tempo real, sem a necessidade de integração através de arquivo.

Uma vez decidida a ferramenta de desenvolvimento, passemos para a solução em si e seus detalhes de implementação que passa pela utilização das classes QGraphicsScene[6] e QgraphicsView[7]. A classe QGraphicsScene é a classe capaz de criar animações, que é exatamente o escopo desejado da solução. A classe QGraphicsView é o contêiner capaz de integrar as cenas criadas em uma janela.

Para a exibição do ambiente de simulação, foram usadas imagens do tamanho 20x20 pixels materializadas em objetos do tipo Qimage[8]. Baseado no evento INIT, uma matriz é gerada com o tamanho necessário para comportar todo o ambiente de simulação e as imagens são preenchidas conforme a tabela abaixo:

Imagem	Nome	Utilização
	Field	Preenchimento das quadras
	Horizontal street	Preenchimento das ruas horizontais
	Vertical street	Preenchimento das ruas verticais
	Corner	Preenchimento das esquinas
	Empty	Preenchimento inicial – bloco ainda não atribuído

Tabela 9. Lista de imagens básicas do animador

Uma vez inicializado o ambiente de simulação, este passará a sofrer mudanças conforme os demais eventos forem ocorrendo, seja na alteração da situação dos semáforos, seja na movimentação dos automóveis. Para esta integração, foram utilizadas imagens com fundo transparente de forma a permitir a união das imagens e a geração de uma imagem final como sendo a sobreposição de várias imagens para compor o bloco da matriz e representar o estado corretamente, conforme podemos ver na tabela abaixo:

Esquina	Automóvel	Semáforo	Resultado
			

Tabela 10. Demonstração da composição de uma imagem da matriz

Visando um resultado mais eficiente, as imagens somente são atualizadas sob demanda, desta forma os elementos da matriz que não sofreram coma mudança de estados do simulador não são atualizadas. Além disto, visando um produto mais independente, todas as imagens utilizadas no projeto foram adicionadas ao resource, ou seja, estas são compiladas e enviadas no programa executável de forma a minimizar as dependências de sua execução. Para um resultado mais fluído e independente, a interpretação dos eventos ocorre em uma thread separada da thread onde a janela é apresentada. Toda a comunicação entre estas threads é feita através de uma área de memória controlada por um semáforo. Veremos com detalhes a comunicação entre as threads logo abaixo neste documento.

Composição das imagens da animação

Para uma completa apresentação do simulador, além das imagens básicas já descritas, foram criadas imagens com fundos transparentes para todos os estados possíveis de semáforos e seis cores de automóveis como é possível visualizar abaixo.

	Verde	Amarelo	Vermelho
Baixo			

Cima			
Esquerda			
Direita			

Tabela 11. Lista de imagens de semáforos

A animação também contou com seis diferentes cores de automóveis onde cada cor era decidida através do módulo do identificador do automóvel por seis. Desta forma foi possível criar um ambiente mais agradável de visualização e com melhor percepção de movimentação dos automóveis. A imagem, então, é definida pelo identificador em conjunto com a direção que o automóvel está.

	Cima	Baixo	Esquerda	Direita
0 – Azul				
1 – Verde				
2 – Lilás				
3 – Vermelho				
4 – Amarelo				
5 – Preto				

Tabela 12. Lista de imagens de automóveis

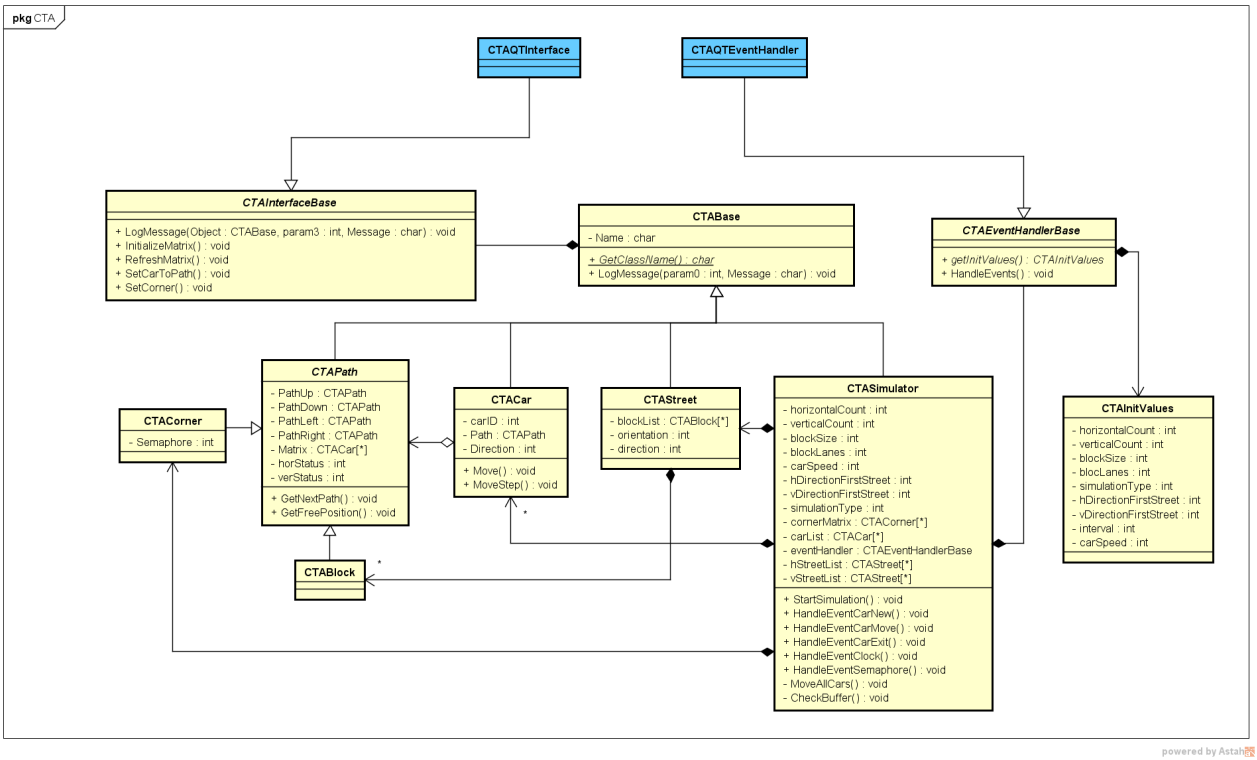
IV. MODELAGEM

Diagrama de classes

A modelagem de classes para esta solução focou em uma solução simples, porém completa e robusta visando o melhor reaproveitamento de código. O resultado desta modelagem é descrito a seguir:

- CTABase: Classe abstrata base para todos os objetos que podem interagir com o simulador. Nesta classe já estão implementada a geração de mensagens de log.
- CTAPath: Classe abstrata derivada de CTABase, nesta classe são implementados controles de movimentação dos automóveis. Através dos atributos de status de direção horizontal e vertical, ela interagem com a classe CTACar fazendo as movimentações destes no simulador. Esta classe também possui referências a outros elementos CTAPath para compor os caminhos.
- CTABlock: Especializada de CTAPath esta classe é a representação das ruas verticais e horizontais.
- CTAStreet: Especializada de CTABase, representa um conjunto de CTABlock para compor uma representação de uma rua no simulador
- CTACorner: Especializada de CTAPath esta classe é a representação das intersecções entre as CTABlock
- CTACar: Especializada de CTABase, a classe CTACar é a representação de um automóvel. Ela é capaz de saber em que posição de CTAPath e é capaz de interagir solicitando avanços e mudanças de direção na simulação.
- CTASimulator: É o simulador propriamente dito. A classe mais complexa que recebe os eventos através de uma especialização de CTAEventHandlerBase e os transforma em mudança de estados em todos os objetos que compõem a simulação.
- CTAEventHandlerBase: Classe abstrata criada para servir de interface para especializações que possam interagir com o simulador enviando eventos para interpretação.
- CTAInterfaceBase: Classe abstrata criada para servir de interface de abstração de eventos do simulador de forma a não ter objetos específicos da biblioteca QT.

Com a criação deste Core, todo o tratamento de mudança de estados do simulador ficou independente de qualquer biblioteca QT. Desta forma é possível reaproveitar todo o projeto para uma futura reimplementação em qualquer outro formato. No entanto,

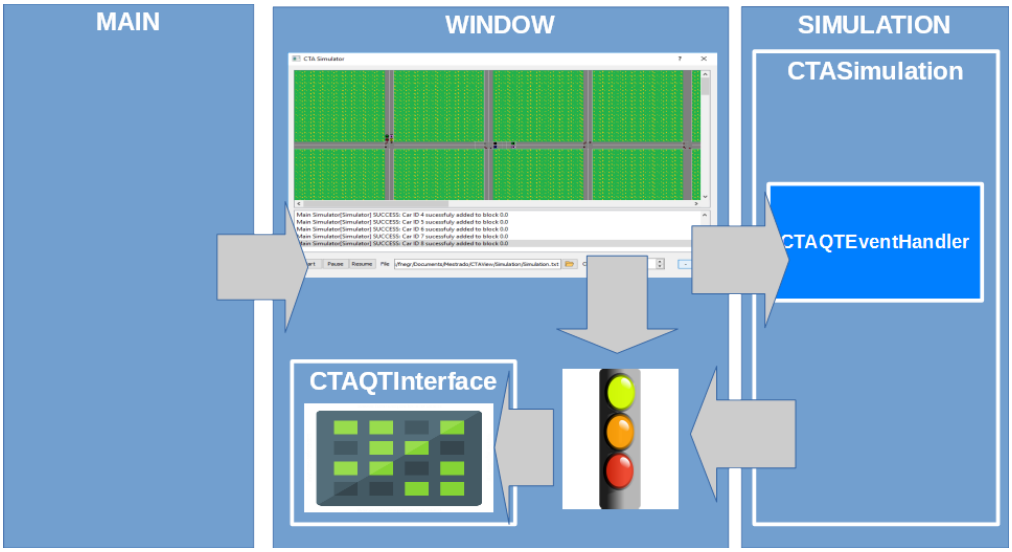


como este trabalho demandou a geração de interface, as seguintes classes foram criadas para interagir com os objetos QT já comentados no capítulo anterior:

- CTAQTInterface: Especializada de CTAInterfaceBase, esta classe interpreta as mudanças de estados e as transforma em mudanças de imagens dentro do animador.
- CTAQTEventHandler: Especializada de CTAeventHandlerBase, esta classe recebe um arquivo de log no formato já descrito anteriormente e transforma estas informações em eventos para serem interpretados pelo CTASimulator.

Diagrama de Threads

Para que a solução tivesse uma fluidez de execução, a execução em diferentes threads foi obrigatória. A solução final utilizada foi a criação de uma thread própria para a janela, thread esta criada automaticamente pela biblioteca QT, e uma thread criada para a



execução do simulador. A interação entre as duas threads ocorre então em memória compartilhada controlada por um semáforo Qsemaphore [9].

V. CONCLUSÕES E TRABALHOS FUTUROS

O desenvolvimento de uma ferramenta que apoie os experimentos para a universidade sempre traz um grande retorno tanto na experiência técnica e este trabalho não foi diferente. O aprimoramento da linguagem C++; conhecimento, ainda que muito limitado, de uma nova ferramenta; problemas, dificuldades e limitações de um desenvolvimento multiplataforma foram alguns dos conhecimentos que foram adquiridos ou expandidos durante esta atividade. Além da experiência técnica, também vale ressaltar a experiência adquirida dos demais alunos que cursaram a disciplina. A interação constante permitiu que o produto fosse testado e melhorado de forma a ter uma versão final estável e funcional, coisa que, com certeza, não conseguiria fazê-lo por minhas próprias forças.

Para trabalhos futuros, a migração do projeto CTA original para C++ multiplataforma (atualmente encontra-se apenas funcionando com C++ para Microsoft Visual Studio) e a integração em tempo real deste projeto com o CTA de forma a termos a simulação direta, sem a necessidade de geração de um arquivo intermediário.

VI. REFERÊNCIAS BIBLIOGRÁFICAS

[1] "CTA_CONOPS", disponível em: < http://www.dainf.ct.utfpr.edu.br/~douglas/CTA_CONOPS.pdf>, 2014, acessado em 15 de julho de 2015

[2] B. Stroustrup, The C++ Programming Language. 4th Edition. Addison-Welsey. 2013.

[3] L. F. Pordeus, "Comparação entre Paradigma Orientado a Notificações e Paradigma Imperativo sobre um Simulador de Tráfego". CPGEI/UTFPR, Curitiba, Brazil, 2015.

[4] P. Van Roy ans S. Haridi, "Concepts, Techniques, and Models of Computer Programming", 2004, ISBN 0-262-22069-5.

[5] "QT", disponível em: <http://www.qt.io>, 2017, acessado em 20 de março de 2017

[6] "QgraphicsScene", disponível em <http://doc.qt.io/qt-5/qgraphicsscene.html>, 2017, acessado em 20 de março de 2017

[7] "QgraphicsView", disponível em <http://doc.qt.io/qt-5/qgraphicsview.html>, 2017, acessado em 20 de março de 2017

[8] "Qimage", disponível em <http://doc.qt.io/qt-5/qimage.html>, 2017, acessado em 20 de março de 2017

[9] "Qsemaphore", disponível em <http://doc.qt.io/qt-5/qsemaphore.html>, 2017, acessado em 20 de março de 2017