**Universidade Federal de Goiás**
Regional Catalão - RC
Departamento de Ciência da Computação
Disciplina: Estrutura de Dados 1
Professor: Thiago Jabur Bittar

**Paulo César de Morais Souza**
Genesis45.ppc@gmail.com
Matrícula: 201805041

**Matheus Rondon Fernandes Cardoso**
Matheusr220998@gmail.com
Matrícula: 201805037

**Daniel Elias Batista de Deus**
danielebdd500@outlook.com
Matrícula: 201805020

Trabalho Final
Estrutura de Dados 1

***CONTROLE BITTRÁFEGO***

**Catalão – GO**
**24/06/2019**

# SUMÁRIO

# 1.0 - INTRODUÇÃO

Este trabalho foi intitulado de “Controle Bittráfego” (Figura 1) e se trata de um sistema para simulação de um fluxo de trânsito entre duas ruas de sentido único controladas por dois semáforos, contendo as luzes vermelha e verde apenas (Excluindo-se a luz amarela tradicional), e um estacionamento que os participantes optaram por tornar independente das regras seguidas pelos objetos citados anteriormente, a fim de tornar a estrutura e a execução do projeto mais dinâmicas. Existem diversas condições que estarão ligadas ao simulador, como o atraso dos carros para dar a partida após o sinal ficar verde, a chegada aleatória de carros em cada rua e até mesmo entrada ou saída no estacionamento. Tudo isso feito de forma sincronizada para evitar erros que, embora inofensivos neste projeto, podem causar sérios problemas em uma situação cotidiana, além de indicadores visuais para facilitar a explicação dos apresentadores e o entendimento dos alunos.

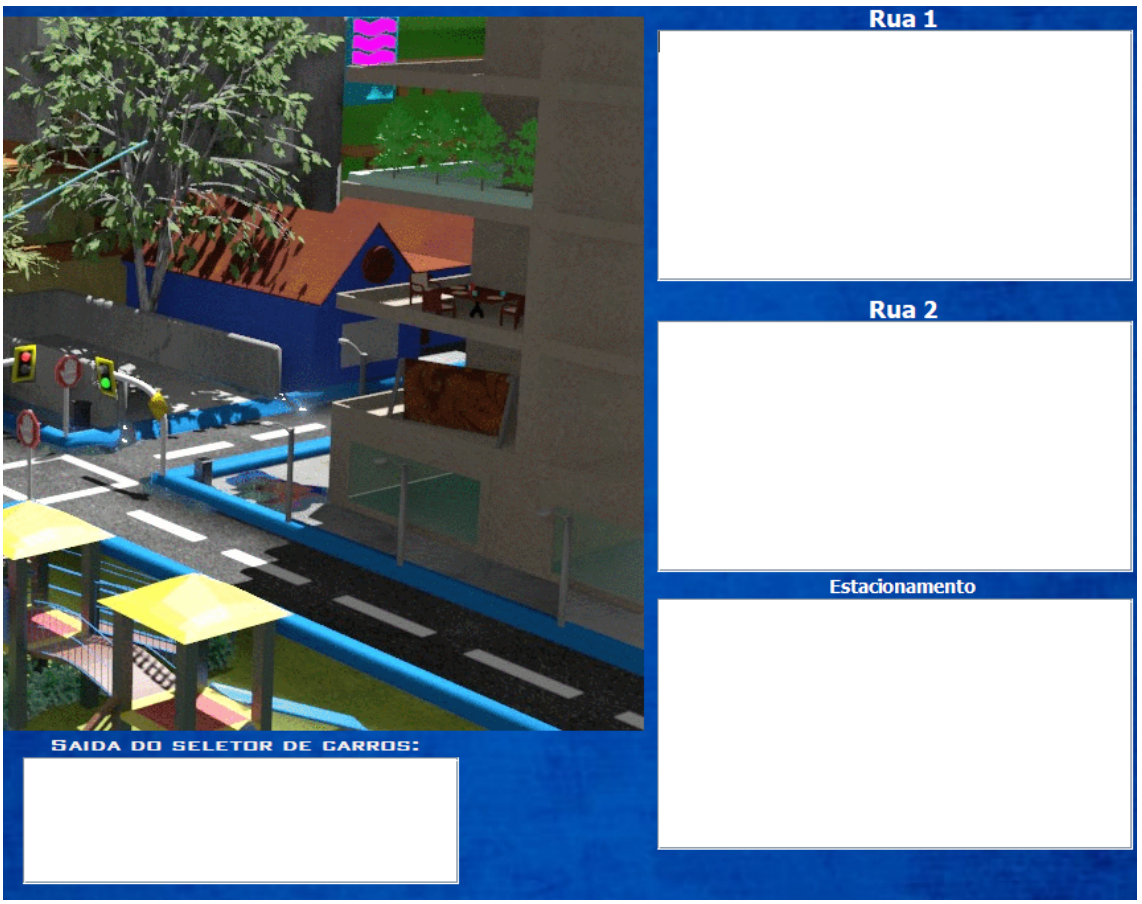
Figura 1 - Controle Bittráfego.


Fonte: Autor.

Todas as funções desempenhadas pelo código poderão ser acompanhadas visualmente no programa utilizando-se de uma imagem tridimensional pré-renderizada e de 4 sistemas de consoles (Figura 2) escritos em tempo real com a informação de entrada de cada carro na rua e a respectiva saída. Tudo isso será feito através da incorporação dos mais variados conceitos abordados

em estrutura de dados, como filas, onde o primeiro objeto a chegar será também o primeiro a sair(FIFO – *First In, First Out*), e pilhas, onde parte da lógica se inverte, tendo que o último a entrar é o primeiro a sair(LIFO – *Last In, First Out*), dessa forma simulando o comportamento de um sistema de trânsito realista, onde os carros que chegam primeiro à esquina precisam aguardar o semáforo indicar o sinal verde para seguir, seguido dos carros que os antecedem, e os carros entram e saem de um estacionamento obedecendo um conjunto de regras distinto.

Figura 2 – Consoles




Fonte: Autor.

O programa também conta com uma interface completa de login por usuário e senha (Figura 3), foi utilizado como background uma *jLabel* com fundo em azul, campos de texto para usuário e senha, respectivamente, e por fim, um botão de login, todos acoplados ao *jFrame* principal para efetuar o trabalho de montagem. Todo o programa foi completamente feito do zero, incluindo o logotipo e os *assets* da cena em 3D (que será explicada mais a frente). A interface de login foi projetada para evitar o acesso não autorizado, e emite um *pop-up* exemplificando o erro de login ao digitar incorretamente o usuário e a senha, utilizamos um sistema de comparação de *strings*, e não uma chave

*hash* para decodificar o usuário e senha pois isso exigiria tempo e conhecimento fora dos limites do prazo e dos níveis de conhecimento obtidos pelos integrantes desse grupo até o presente momento.

Figura 3 – Sistema de Login


Fonte: Autor.

# 2.0 – CONSTRUÇÃO

O programa para simulação de trânsito foi feito usando a linguagem Java como base de funcionamento. O Sistema Java foi criado em 1995 pela Sun Microsystems, hoje pertencente a Oracle, e tem um grande uso em diversos sistemas. Uma de suas grandes vantagens é a possibilidade de ser executado em qualquer sistema operacional, pois diferente de outras linguagens, esta é executada em uma máquina virtual, o que permite que o mesmo código seja executado em diferentes plataformas sem a necessidade de nenhuma mudança em sua escrita, além disso é possível instalar novas bibliotecas no Java para expandir suas características ou facilitar algum tipo de desenvolvimento, como integração à diferentes bancos de dados ou até mesmo com outras linguagens, como a C/C++, por exemplo.

Uma das funções que foram usadas no programa foram os threads. O conceito de threads se refere quando é necessário executar ao mesmo tempo duas ou mais operações no programa. Por exemplo: um sistema que durante o

carregamento de um arquivo exiba a atualização de uma barra de progresso ao mesmo tempo em que executa outra tarefa. Para isso, é necessário que duas linhas de código sejam executadas simultaneamente, no exemplo dado acima, é a que busca o arquivo e a que atualiza a barra conforme a leitura do arquivo avança, respectivamente. Esta função não deve ser interpretada como algo desenvolvido unicamente para acelerar o processamento (se levando em conta a ideia de que ao usar threads será possível realizar dois processos ao mesmo tempo), pois sua função não é restringida apenas a esses casos de utilização.

Quando houver a invocação deste método o que o programa fará é alternar o processamento entre cada função. Uma forma pratica de entender e pensar em um algoritmo que utilizará duas classes distintas que atuarão ao mesmo tempo uma imprimindo o número 1 na tela e outro imprimindo o número 2. Toda vez que o programa for executado veremos na saída uma sequência aleatória de 1 e 2 que a cada execução poderá mudar de ordem graças às diferenças no projeto de cada computador. O único momento que se pode pensar em múltiplas ações ocorrendo de forma paralela é quando se tem uma máquina com uma arquitetura onde o processador conta com vários núcleos. Nesse caso o processador controla onde cada dado é processado, para agilizar os processos.

O programa utiliza vários threads que são executados ao mesmo tempo, existe um thread para os sons do ambiente, um thread para a fila da rua 1, outro para fila da rua 2 e um para a pilha (estacionamento), todos são instanciados e iniciados pela interface principal. As duas ruas não possuem interação entre si, cada uma funciona totalmente independente da outra, o tempo de sincronização das duas ruas foi precisamente calculado, enquanto uma fica aberta a outra fica fechada, esse tempo de intervalo foi disponibilizado nas instruções do trabalho e é de 15 segundos.

O controle de cada um dos carros que estarão presentes no programa tem como base algumas estruturas de dados, que são maneiras de organizar e realizar operações entre diferentes dados. Existem vários tipos de organizações para cada tipo de necessidade, como pilha, fila, arvore e lista. Foi feito uso em maior quantidade do conceito de fila, onde o primeiro elemento a entrar será o primeiro a sair (O mesmo conceito se encontra em uma fila real onde a primeira pessoa a entrar em uma fila será a primeira a ser atendida e sairá primeiro). Outro conceito de estrutura de dados é o de pilha, onde o primeiro item a ser adicionado será o último a ser retirado, bem semelhante a uma alusão de uma pilha de roupas onde a primeira roupa colocada ficará por baixo das outras e só depois de todas as roupas colocadas depois dela serem retiradas, que a primeira irá sair. Existem várias outras estruturas de dados, cada uma com uma aplicação especifica, conceitos como os de uma agenda, uma lista de itens e outros itens diversos necessitam de uma correta organização para que possam ser eficientes e bem organizados de forma que o computador possa os encontrar de forma mais rápida.

Na parte visual temos uso de uma imagem 3D dinâmica de fundo com o layout de uma rua e um estacionamento. A imagem foi feita usando texturas para representar cada um dos objetos mostrados como a grama e a rua. O 3D tem aplicações em diversas áreas como filmes, jogos e outras aplicações profissionais. O processo é feito criando objetos digitais com 3 dimensões utilizando programas específicos para isso, existem diversos programas tanto pagos como gratuitos para este fim, por meio deles se pode criar diversas cenas, personagens, imagens, e muito mais. Porém, inicialmente é necessário modelar os objetos que serão usados. É preciso em seguida cuidar da parte da iluminação e posteriormente é adicionada uma cor solida aos modelos ou uma textura, pois durante a modelagem os objetos apresentam uma cor neutra e não utilizam texturas muito complexas, para facilitar a modelagem e para poupar processamento.

Os modelos tridimensionais são bastante usados hodiernamente e estão cada vez mais avançados desde os seus primeiros usos tanto no cinema quanto no meio militar. O 3D está ligado ao conceito de CGI (*Computer-Generated Imagery* ou imagens geradas por computador), e os primeiros usos de objetos tridimensionais se deram por volta da década de 60, tendo grande salto de tecnologia por volta dos anos 70 quando a IBM fez o lançamento um padrão de linguagem gráfica e técnicas de desenvolvimento 3D.

A modelagem 3D se baseia nos sólidos geométricos onde se usa um ou vários objetos que podem ser combinados entre si, moldados, reescalonados, além da possibilidade de se alterar outras propriedades relacionadas a eles. Eles são formados por triângulos e quanto mais deles houverem, mais processamento será necessário para executar a renderização do objeto. O conceito de renderização pode se referir a quando um filme em CGI é produzido e é necessário deixar os computadores trabalhando para transformar a animação feita no programa em um arquivo de vídeo. No caso de um jogo ou um programa, como o apresentado, é necessário que essa renderização seja feita antecipadamente e depois importada para o programa.

Neste caso há economia de recursos do computador que irá rodar o simulador com imagens 3D pré-renderizadas. Esta técnica foi usada extensivamente durante os anos 90 para permitir que hardwares de entrada, ou disponíveis aos consumidores, rodassem jogos e interfaces gráficas tecnicamente superiores. Isso era feito utilizando-se uma máquina que possuía funções e recursos de alta performance para renderizar o gráfico tridimensional em uma serie de objetos já prontos como se fossem um arquivo de imagem (que podem ser usados como um personagem em um jogo ou mesmo um cenário), dessa forma o sistema não teria a necessidade de criar o gráfico em tempo de execução, apenas ler o arquivo já pronto e o inserir no contexto a ser utilizado. Isso permite programas mais leves e que permitem gráficos “melhores”, mesmo que enganando o usuário de certa forma. Apesar de ter certas limitações como não permitir que um determinado cenário seja visto de muitos ângulos (pois de forma conceitual ele se comporta como uma imagem estática que teve origem em um objeto tridimensional que tinha liberdade de movimento, mas agora não

pode se mover livremente nos três eixos cartesianos XYZ por se tratar de um objeto 2D) ele tem seus usos até mesmo nos dias atuais, como na criação de filmes (como não há interação direta com o usuário, a limitação de não poder alterar a visão do cenário em mais de dois eixos não chega a ser um problema), e também neste projeto, que não necessita diretamente da necessidade de rotação. Além disso isso permite que mesmo maquinas menos robustas possam executar o programa sem grandes problemas, descartando a necessidade de um hardware dedicado com suporte a recursos 3D.

Outra técnica empregada no projeto é o Ray Tracing, que aqui possui uma implementação não exatamente em tempo real, mas sim de forma simulada. Esta técnica já usada a muito tempo pelo cinema começou a se expandir para o mundo dos games recentemente. Ela consiste em calcular a trajetória da luz por meio de não apenas um único raio de luz, mas sim de milhares ou até mais, buscando obter o melhor resultado possível na simulação gráfica de determinados alguns efeitos em formas mais realistas, como reflexos e sombras mais fisicamente precisos. Apesar de já serem bem usados em games os atuais efeitos de reflexos tratam se de imagens estáticas que simulam um possível reflexo dando a ilusão de um reflexo baseado nos raios de luz transmitidos pelos objetos próximos. Os filmes, como já citado anteriormente, não são renderizados em tempo de execução, então é possível usar esse efeito de forma mais abrangente e com melhor qualidade final, porém em games ou outras aplicações são necessárias grandes quantidades de processamento, pois é necessário que sejam rasterizados em tempo real e com bom desempenho, para permitir uma interação fluída com o jogo em si.

Na interface do programa há uma visão tridimensional onde se consegue ver as duas ruas por onde trafegarão os carros. Ademais, os carros poderão aleatoriamente entrar em um estacionamento, o que cria uma variação para melhorar o conceito de “simulador”, pois no mundo real existem diversas variáveis que podem causar mudanças no transito de uma rua, podendo ser tanto a entrada ou saída de um veículo no estacionamento ligado a este asfalto, quanto a trajetória de outros no sinal verde desta mesma rua. Ao lado da simulação em 3D é possível ter a visualização de alguns consoles (Figura 5) que imprimem na tela em qual posição cada carro está e sua respectiva saída, mudando de cor conforme a cor do sinal da respectiva rua é alterado. Como já havíamos falado anteriormente, os controles internos do programa fazem uso de conceitos da estrutura de dados, como filas que representam as ruas e o agendamento de saída dos semáforos, onde o primeiro veículo a entrar será o também o primeiro a sair(*FIFO - First In, First Out*) e pilha, incorporada ao estacionamento, que, por sua vez, utiliza outro conjunto de regras, onde o último a entrar é o primeiro a sair(*LIFO - Last In, First Out*).

Infelizmente nos foi impossível criar uma animação satisfatória dos carros em movimento, pois não apenas isso exigiria uma carga de tempo muito superior à estipulada para o tempo disponível para o projeto, como também aumentaria desproporcionalmente a complexidade da programação, para acomodar as camadas de forma mais inteligente, somado às problemáticas citadas

anteriormente, haveríamos de realizar mais e mais horas de testes de alinhamento, modelagem e animação 3d para criar as animações individuais dos carros. Diante disso, podemos afirmar que ainda estamos contentes com o resultado final da interface exibida pelo projeto, que possivelmente agradará aos olhos do professor e dos alunos, pois será uma empreitada que difere bem de algo geralmente visto nos trabalhos apresentados em classe.

# 3.0 – TESTES

De início, certamente foram encontrados inúmeros obstáculos para terminar, e algumas vezes ao menos começar, a implementação correta de cada uma das partes do trabalho, alguns imprevistos sempre ocorrem nesse tipo de empreitada, como bugs, dificuldades ao entender e/ou consertar alguma implementação inadequada e até mesmo erros do compilador, tudo isso pôde ser descoberto através de uma  testes. Não foi utilizado o conceito de *jUnit* para a realização dos testes, pois como é um trabalho extenso, isso demandaria uma quantidade massiva de horas que não estavam disponíveis para os integrantes do grupo, além de que para realizar esse teste é necessária uma saída fixa, o que não ocorre, pois o programa faz uso recorrente de *threading* e gera dados aleatórios, como o número de identificação dos carros, portanto todos os testes foram feitos de forma manual.

Após horas e horas de testes, consertos de bugs e problemas em geral, melhorias tanto no visual quanto na estrutura do código e refinamentos nos passos de execução, o projeto atingiu os seus estágios primordiais de funcionamento, com a base lógica toda incorporada, desde as filas e a pilha, até as representações visuais pela linha de comando e melhorias de qualidade de vida. Posteriormente o código recebeu sua camada de polimento, com melhor sincronização e agendamento nas estruturas, além de grandes avanços na parte visual do mesmo, que resultou no trabalho aqui presente, que dispõe de vários recursos novos se comparado ao momento em que este se tornou utilizável.

# 4.0 – CONCLUSÃO

Ao alcançarmos os estágios finais do código, os testes se provaram frutíferos, com todas as funções, elementos gráficos, sincronizações e saídas tanto dos consoles quanto das animações atingindo os padrões de qualidades esperados pelos integrantes, obtendo resultados precisos e reproduzíveis de forma pelo menos similar em diferentes sistemas operacionais e hardwares, atendendo às expectativas exigidas do mesmo, como flexibilidade e agilidade de execução, o

que nos permitiu construir o projeto de forma a exibir suas funções sublimemente em quaisquer dispositivos que tenham suporte à linguagem Java e às APIs básicas de suporte à desenhos gráficos em duas dimensões, atestando ambas a qualidade de escrita do código e a criatividade dos integrantes do grupo, pois nos foi possível acoplar um projeto com vários efeitos visuais tridimensionais e manter o desempenho do mesmo em margens aceitáveis aos usuários do mesmo.

Ademais, procuramos manter o número de threads utilizado no menor possível, a fim de reduzir possíveis erros de sincronização entre os diferentes elementos exibidos na tela, ao mesmo tempo em que reduz a complexidade geral do código, facilitando o entendimento do mesmo por terceiros sem que estes precisem ter um *background* sólido nos conceitos apresentados pelo mesmo, tornando-o um código relativamente '*beginner-friendly',*ou, para os menos instruídos em inglês, "amigável à iniciantes", podendo ser apresentado para diferentes tipos de pessoas sem que haja uma forte discrepância no conteúdo absorvido pelas mesmas.

# 5.0 - REFERÊNCIAS


CAELUM. O que é Java. Disponível em: < https://www.caelum.com.br/apostila-java-orientacao-objetos/o-que-e-java/#onde-usar-e-os-objetivos-do-java>. Acesso em: 15 jun. 2019.

CAELUM. PROGRAMAÇÃO Concorrente e Threads. Disponível em: < https://www.caelum.com.br/apostila-java-orientacao-objetos/apendice-programacao-concorrente-e-threads/>. Acesso em: 15 jun. 2019.

SAGA. O que é modelagem 3D?. 2016. Disponível em: < https://saga.art.br/o-que-e-modelagem-3d/>. Acesso em: 15 jun. 2019.

LINARES, Gustavo. O que é CGI e computação gráfica?. 12 out. 2016. Disponível em: < https://canaltech.com.br/software/O-que-e-CGI-e-computacao-grafica/>. Acesso em: 15 jun. 2019.

CAELUM. ALGORITMOS e Estruturas de Dados com Java. Disponível em: < https://www.caelum.com.br/apostila-java-estrutura-dados/>. Acesso em: 15 jun. 2019.

GARRET, Filipe. O que é ray tracing? Veja como funciona a tecnologia para placas de vídeo. Techtudo. 2018. Disponível em: < https://www.techtudo.com.br/noticias/2018/04/o-que-e-ray-tracing-veja-como-funciona-a-tecnologia-para-placas-de-video.ghtml>. Acesso em: 16 jun. 2019.

CAELUM. PRECISAMOS falar sobre Ray Tracing. Disponível em: < https://www.caelum.com.br/apostila-java-estrutura-dados/>. Acesso em: 15 jun. 2019.