Relatório projeto CarSim

Aluno: Gabriel Moreno Frota Ra: 62961

Execução e funcionamento do programa

O programa encontra-se no github, através do endereço github.com/GabrielFrota/carsim. Foi escrito na linguagem Java, e requer as ferramentas Java corretamente instaladas no sistema para executá-lo. Requer Java versão 8 no mínimo. Utiliza a biblioteca jnativehook, que se encontra na pasta lib, para captura de eventos de teclado nativos. Eventos nativos foram usados para implementação dos atalhos globais do programa, pois o Java padrão não lida bem com captura de teclas globais. O Java padrão lança um evento de tecla no componente “focused”, que é útil para capturar teclas para inserção de texto em algum campo, mas para capturar atalhos globais do programa não funciona bem. Portanto para compilação/execução do programa, é preciso passar o caminho da biblioteca atraves da variável CLASSPATH ou flag -cp. Exemplo:

```
export CLASSPATH=.:./lib/jnativehook/jar/jnativehook-2.1.0.jar
javac *.java
java MainFrame
```

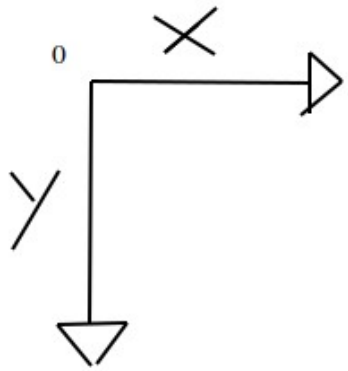
O programa ao iniciar busca por 2 arquivos de configuração. Caso os arquivos existam, irá carregar as configurações dos arquivos, e caso não existam, irá iniciar o programa com configurações padrão. Os arquivos tem um formato esperado, e caso estejam incorretos, o programa dará erro. Salvando os arquivos pelo programa CarSim, eles sempre serão salvos no formato correto, portanto só dará erro se alterar os arquivos por fora do programa. Os arquivos são “world.bmp” (representação do mundo) e “carsim.conf” (valores da classe Config). A ideia do programa é desenhar no painel o mundo desejado, marcar uma posição do carro, e iniciar a simulação do movimento.

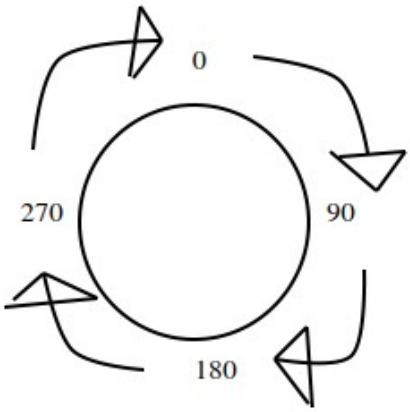
O menu “Options” seleciona o modo de execução do programa, e cada item do menu possui um atalho no teclado para melhorar a usabilidade. Os itens são os seguintes:
- Set Tiles (tecla Q): Seleciona modo de desenho do mundo. Um clique esquerdo no painel pinta um retângulo preto no local do clique, e um clique direito pinta um retângulo da cor de fundo no local do clique (por padrão é cinza). O retângulo preto é um obstáculo que o carro desviará, e o retângulo cinza tem a função de ser uma espécie de borracha, para retirar um desenho preto previamente ali.
- Set Car (tecla W): Seleciona a posição inicial do carro com um clique do mouse.
- Step Mode (tecla E): Seleciona modo de passo a passo na animação do carro. Esse modo tem a intenção de ser uma espécie de debugger do programa. Cada clique do mouse no painel, executa 1 ciclo do carro, e pinta o estado na tela. A execução será bastante lenta, e com isso é possível perceber com clareza o que acontece no programa, o que facilita a correção de erros.
- Start Simulation (tecla R): Seleciona modo de simulação do movimento do carro. Nesse modo, a cada 20ms, é executado um ciclo do carro, e pinta-se o estado do programa na tela.
- Configurations (tecla T): Abre janela de configurações. Nessa janela, é possível alterar vários parâmetros que influenciam no movimento do carro. As configurações serão detalhadas na seção da janela de configurações.

O menu “Tile Size”, altera o tamanho do retângulo que será desenhado na tela no modo “Set Tiles”. O tamanho pode ser selecionado com um clique, ou com atalhos do teclado. A tecla “= +” aumenta o tamanho, e a tecla “- _” diminui.

Sistema de Coordenadas

O programa possui um sistema de coordenadas implícito, que governa a posição dos objetos e o movimento do carro. É muito importante entender o sistema de coordenadas, para entender o movimento do carro. Existe a noção de uma posição x e y, e existe a noção de uma rotação de θ graus. O movimento para a frente do carro altera a posição x e y em função da velocidade (transformação geométrica de translação), e a rotação do carro faz um incremento circular na direção atual do carro, e gira o retângulo (transformação geométrica de rotação). Um valor positivo gira o carro no sentido horário, um valor negativo gira o carro no sentido anti-horário. A direção atual do carro indica a direção da translação no próximo ciclo, portanto atualizar a direção do carro e girar o retângulo são 2 operações distintas.





Movimento do Carro

O carro vai sempre para frente caso o caminho esteja livre, e caso algum sensor indique um obstáculo muito próximo, o carro faz um movimento diferente, na tentativa de evitar uma colisão. O carro possui 3 sensores, e seus alcances são representados pelas linhas verdes e vermelha.
- Sensor Infravermelho: possui um em cada quina frontal do carro. Um obstáculo muito próximo de um sensor infravermelho, causa o carro dar uma leve girada na direção oposta do sensor (o sensor esquerdo faz girar pra direita, o sensor direito faz girar pra esquerda). Esse sensor evita colisões leves, que são fáceis de desviar com um movimento pequeno.
- Sensor ultrassônico: possui 1 na frente do carro, encaixado em um motor giratório. Esse sensor tem a missão de evitar colisões frontais, e de identificar uma nova direção para o carro sair andando. Esse sensor causa um movimento mais complexo no carro, pois a colisão não é tao simples de ser evitada. O carro para e o motor giratório é acionado, posicionando o sensor nos limites esquerdo e direito do arco, sendo assim possível identificar o lado com mais espaço para o carro girar. O carro então gira no lugar, e resume o movimento para frente.

Janela de configurações

Possui vários campos para alterar os valores dos parâmetros do simulador. Cada opção corresponde a uma variável global da classe Config, que será usada no algoritmo da classe Car.

Opções do carro:
- width: Largura do retângulo do carro.
- height: Altura do retângulo do carro.

- speed: Velocidade de movimentação do carro em pixels. O carro irá andar speed pixels por ciclo.
- direction: Direção inicial do carro ao posicioná-lo no mundo.
- rotation step: Quantidade em graus de 1 passo da rotação do carro.
- collision rotation: Quantidade em graus de 1 passo da rotação do carro em uma colisão.
- collision check on: Liga/Desliga checagem de colisão.
- sensor check on: Liga/Desliga checagem dos valores dos sensores.

Opções do sensor infravermelho:
- maximum range: Alcance máximo.
- threshold value: Valor limite para reação do carro.
- left angle: Posição em graus do direcionamento do sensor esquerdo.
- right angle: Posição em graus do direcionamento do sensor direito.

Opções do sensor ultrassônico:
- maximum range: Alcance máximo.
- threshold value: Valor limite para reação do carro.
- rotation arc size: Tamanho do arco de rotação do sensor em graus.

Botões:
- save config: Salva valores da classe Config em um arquivo.
- load config: Carrega de um arquivo valores para a classe Config.
- save world: Salva representação do mundo em um arquivo.
- load world: Carrega de um arquivo a representação do mundo.

Código fonte

- State.java: Modo de execução do programa no instante atual. O valor dessa variável governa o programa em diferentes trechos de código, como em eventos de clique, teclas ou de desenho na tela. Cada item do menu “Options” seta o determinado valor dessa variável, que fará com que o programa execute diferente.

- TileSize.java: Mantém o tamanho do retângulo desenhado no modo “Set Tiles”

- World.java: Representação do mundo dentro do programa. O mundo é um array de pixels do tamanho da tela, aonde pixels pretos representam um obstáculo. O programa pinta esse array de pixels no DrawingPanel a cada ciclo da animação.

Config.java: Parâmetros de execução do programa. Podem ser alterados pela janela de configurações, lidos de um arquivo qualquer através do botão “load config”, ou carregados do arquivo “carsim.conf” na inicialização do programa.

MainFrame.java: Janela raiz do programa. Inicialização e configuração da interface gráfica. Mantém os Menus e seus MenuItems, e o painel DrawingPanel. Também inicializa a thread para captura de eventos de teclado nativos, que será usada para implementar os atalhos pelo teclado.

ConfigFrame.java: Janela de configurações. Inicialização e configuração dos campos e botões. Também valida os valores inseridos nos campos, e exibe uma mensagem de erro em caso de valor inválido.

DrawingPanel.java: Painel da animação. Esse painel está posicionado dentro do MainFrame, e a cada ciclo da animação é redesenhado (execução do método paintComponent()). A ordem das operações de desenho são a seguinte:

1- pintar array de pixels que representa o mundo
2- pintar um retângulo ou o carro na posição do ponteiro do mouse
3- pintar o retângulo do carro
4- pintar linhas que representam o alcance dos sensores
5- pintar texto com estatísticas

Car.java: Mantém tudo relacionado ao carro robô. Essa classe é complicada e eu vou escrever comentários no código sobre seu funcionamento e suas funções, pois escrever algo aqui vai ficar muito confuso. Entender o que acontece aqui requer noções de computação gráfica, e entender a biblioteca de gráficos da linguagem Java.

Conclusão

O projeto serviu para conhecer um pouco do microcontrolador Arduino e seus componentes, conhecer da montagem das peças e funcionamento de um carro robô, e principalmente para conhecer melhor a biblioteca de gráficos da linguagem Java, que foi muito importante no desenvolvimento do programa simulador.