Relatório de Implementação: Estacionamento de Caminhão com Lógica Fuzzy

Daniel Ricardo dos Santos
Diogo de Campos
Maurício Oliveira Haensch

2 de junho de 2010

Descrição

O trabalho consiste em implementar um programa onde um caminhão deve estacionar de ré utilizando lógica nebulosa, seguindo o modelo passado pelo professor, disponível em http://turing.gsi.dit.upm.es/~lssii/pfuzzy/truck/fztruck.html. O código foi desenvolvido na linguagem *Python* [2], utilizando a biblioteca gráfica *Qt*[3] através do framework *PyQt*[1]. As instruções de instalação das ferramentas necessárias para executar o programa podem ser encontradas no final deste relatório.

Decidimos preservar as características do programa de exemplo citado anteriormente, como a existência de 5 grupos de posição e 7 grupos de ângulos possíveis, que são combinados na *FAM (Fuzzy Associative Memory)* para gerar as diferentes possibilidades de atuação no caminhão. Também utilizamos a mesma terminologia para as possíveis respostas na *FAM*, sendo:

PS *Positive Small*

PM *Positive Medium*

PB *Positive Big*

NS *Negative Small*

NM *Negative Medium*

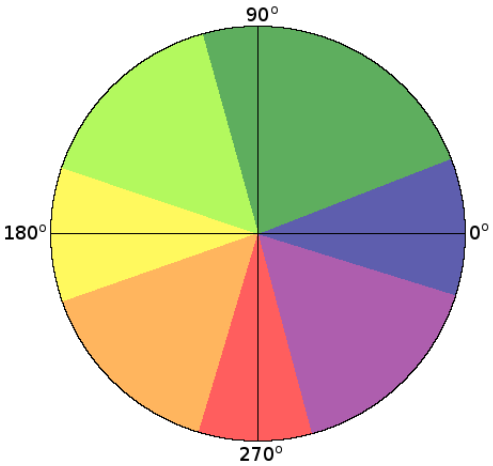
NB *Negative Big*

ZE *Zero*

Os valores iniciados por *Positive* significam que o caminhão deve virar no sentido anti-horário, os valores iniciados por *Negative* indicam o sentido horário. A segunda parte, *Small, Medium* e *Big* representam a acentuação da virada e a alteração desses valores tem uma grande influência no resultado final.

A seguir, uma imagem da interface do programa, onde é possível ver a matriz *FAM*, algumas barras para indicar alguns parâmetros para a simulação (velocidade da simulação, do caminhão e ângulo do caminhão), assim como os botões responsáveis por iniciar, pausar e resetar a simulação e resetar a matriz de possibilidades. O usuário pode também alterar cada uma das regras que deve ser seguida para cada uma das combinações presentes na *FAM* e a posição inicial do caminhão é escolhido através de um clique do mouse.

A *FAM* é uma matriz 7x5, onde cada linha representa um dos conjuntos de ângulos, seguindo uma divisão semelhante à mostrada na figura abaixo[1]:



As colunas da matriz representam os conjuntos relacionados à posição do caminhão no eixo X, divididos da seguinte forma:

1. Muito à esquerda;
2. À esquerda;
3. No meio;
4. À direita;
5. Muito à direita;

[1] a figura é apenas uma aproximação dos conjuntos utilizados de fato

Durante a execução do programa, o ângulo atual do caminhão é atualizado na interface, movimentando a barra deslizante *Ângulo do caminhão*, assim como o grau de pertinência dentro do caminhão nos conjuntos definidos pode ser visto na matriz durante a execução. Quanto maior o grau de pertinência em uma dada combinação, maior é a fonte utilizada no texto que exibe a função mapeada para aquela combinação na matriz.

Para chegar a uma inteligência artificial adequada ao programa, foram feitos diversos testes modificando os vários parâmetros, como os conjuntos de posição, conjuntos de ângulos e o valor das funções *PS, PM, PB, NS, NM, NB*. O programa foi executado diversas vezes com o caminhão iniciando em diferentes posições e com diferentes ângulos para testar a eficiência das funções e conjuntos definidos.

A modificação de um dos parâmetros presentes na simulação, como os conjuntos de posição e ângulo, pode alterar significativamente o resultado do comportamento do caminhão durante a execução do programa. A alteração dos conjuntos vai interferir no grau de pertinência do caminhão, de acordo com seu ângulo e posição com relação ao eixo X, modificando o valor final que será usado para rotacionar o caminhão durante um passo do algoritmo de movimentação. Por exemplo, ao aumentar o suporte de um dado conjunto, o caminhão pode acabar sendo afetado por mais combinações de posição/ângulo ao se executar o programa, tendo que aplicar mais regras para compor o resultado final.

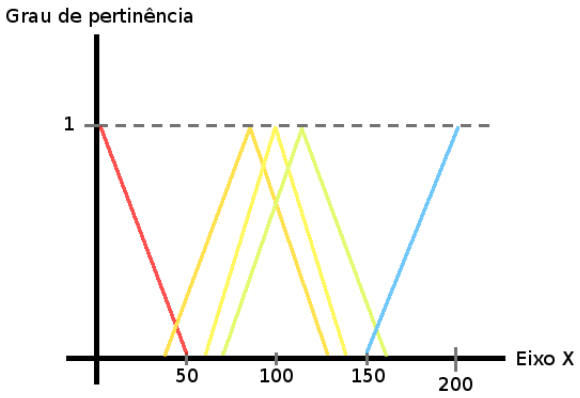
A alteração das regras a serem executadas dependendo da combinação das pertinências em diversos conjuntos também é refletida na trajetória adotada pelo caminhão. Em casos em que, para uma trajetória correta, seria necessário uma curva pouco acentuada em um dos sentidos, modificar a regra acentuando a curva pode levar o caminhão a seguir um trajeto não desejado. Numa situação real, seria o caso em que, por exemplo, o motorista deveria virar pouco o volante para a esquerda, mas acaba virando muito ou virando para o outro lado.

Modificar um conjunto ou uma regra isoladamente pode gerar um resultado indesejado. Não existe apenas uma combinação de conjuntos e regras que leve a um bom resultado, mas sim a harmonia e a interação entre estes elementos produz o efeito procurado. Para encontrar bons valores que produzissem um bom resultado, realizamos diversos testes, entre estes, dois que são apresentados a seguir.

Caso de testes 1

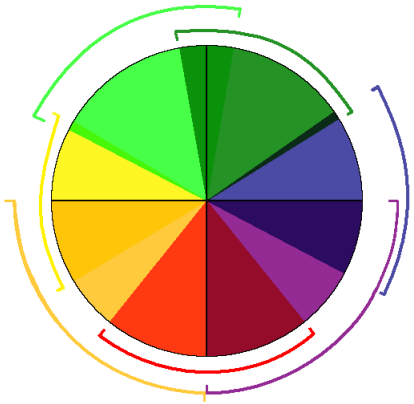
O primeiro conjunto de testes realizados utilizou a seguinte configuração:

Para os conjuntos correspondentes à posição do caminhão com relação ao eixo X, foram usadas formas triangulares dispostas da seguinte forma:



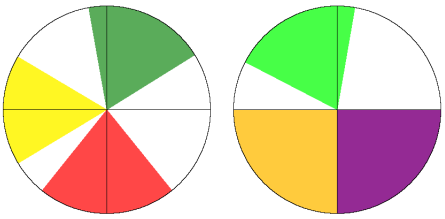
- Muito à esquerda: de 0 a 50, com pico em 0
- À esquerda: de 40 a 130, com pico em 85
- No meio: de 60 a 140, com pico em 100
- À direita: de 70 a 160, com pico em 115
- Muito à direita: de 150 a 200, com pico em 200

Para esse caso de teste, os conjuntos de ângulos utilizados foram os mostrados na figura, com os valores listados logo abaixo.



- Vermelho: de 230 a 310, com pico em 270
- Laranja: de 180 a 270, com pico em 225
- Amarelo: de 145 a 215, com pico em 180
- Verde claro: de 80 a 150, com pico em 115
- Verde escuro: de 30 a 100, com pico em 65
- Azul: de 150 a 200, com pico em 200
- Roxo: de 325 a 395 (congruente a 35 graus), com pico em 360

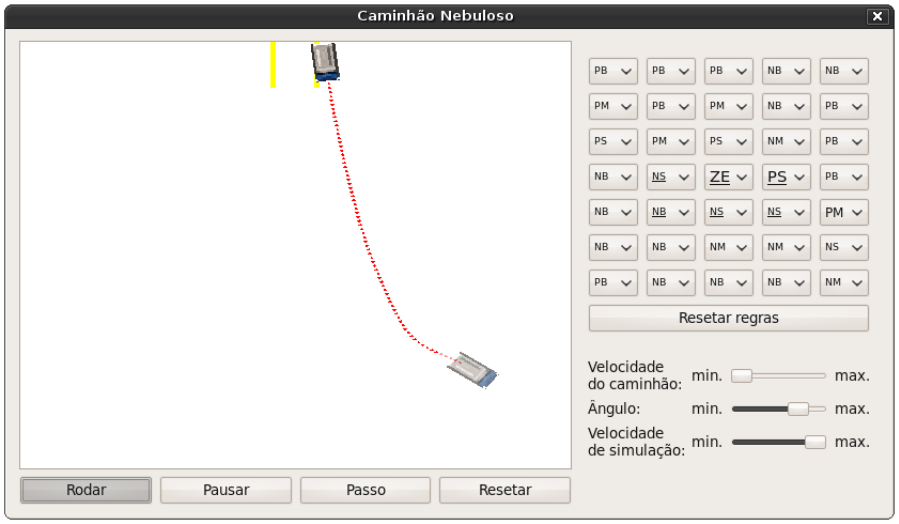
Abaixo, uma outra imagem com apenas alguns dos conjuntos de ângulos sem intersecção, para que fique mais claro algumas das divisões mostradas na figura acima.



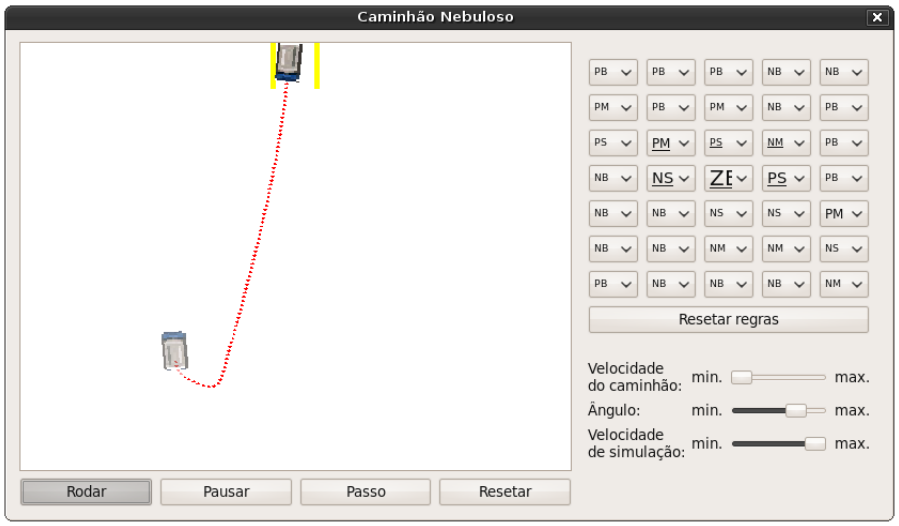
Os valores mapeados pelas funções *PS, PM, PB, NS, NM, NB* para esse caso foram 15, 30, 45, -15, -30, -45 respectivamente, e a FAM utilizada nesse caso pode ser vista abaixo:

PB	PB	PB	NB	NB
PM	PB	PM	NB	PB
PS	PM	PS	NM	PB
NB	NS	ZE	PS	PB
NB	NB	NS	NS	PM
NB	NB	NM	NM	NS
PB	NB	NB	NB	NM

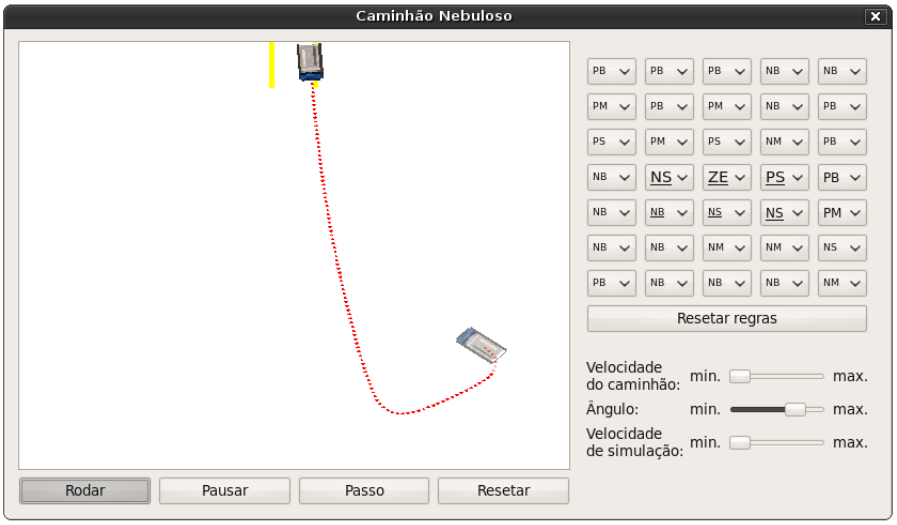
Para essa configuração adotada, os resultados foram suficientemente bons. O caminhão em geral chegava bem próximo à vaga, como pode ser observado nos exemplos abaixo:



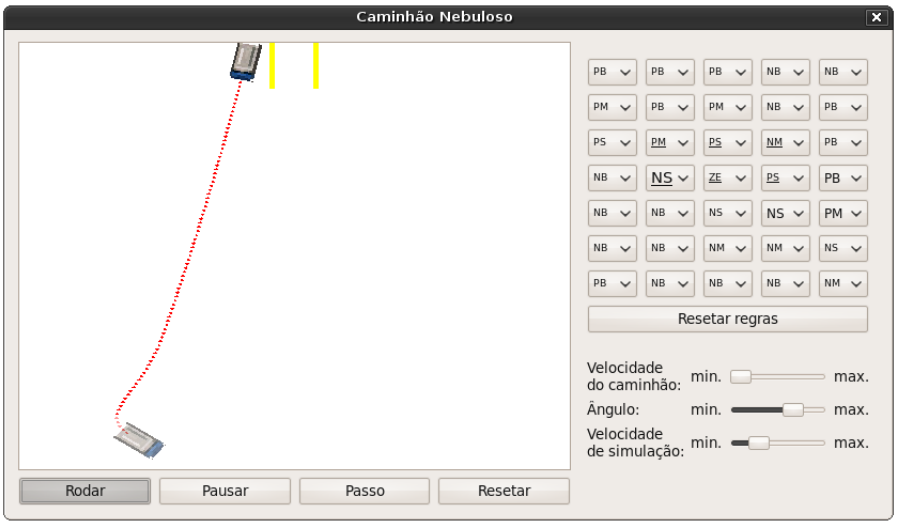
Exemplo 1



Exemplo 2



Exemplo 3

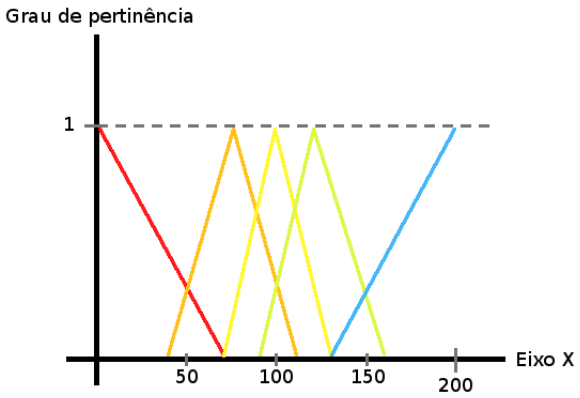


Exemplo 4

Após esses e outros testes terem sido realizados, modificamos alguns dos conjuntos e regras para chegar a um modelo mais eficiente baseado nos problemas encontrados durante a execução. A seguir, será mostrado o próximo caso de testes, que é a configuração que adotamos no programa final.

Caso de testes 2

Foram modificados os conjuntos de posição do caminhão com relação ao eixo X, ainda utilizando conjuntos triangulares. Os conjuntos ficaram da seguinte forma:

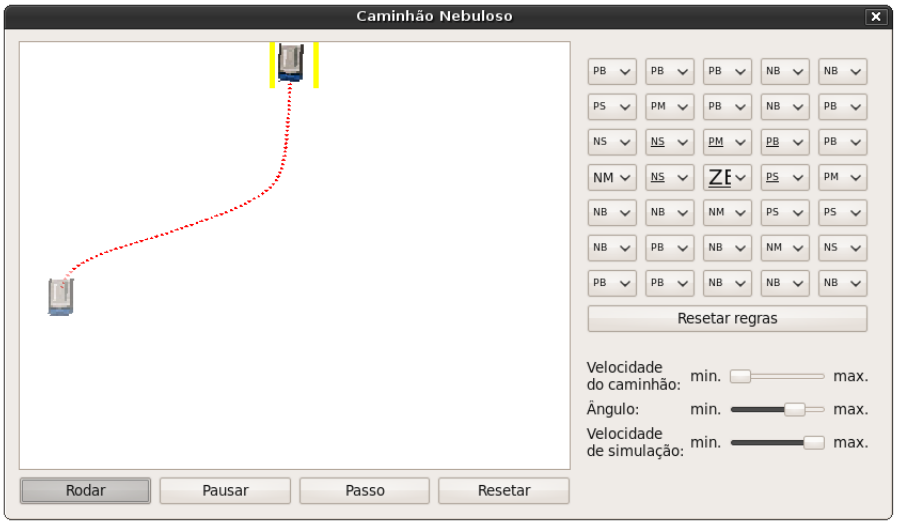


- Muito à esquerda: de 0 a 70, com pico em 0
- À esquerda: de 40 a 110, com pico em 75
- No meio: de 70 a 130, com pico em 100
- À direita: de 90 a 160, com pico em 125
- Muito à direita: de 130 a 200, com pico em 200

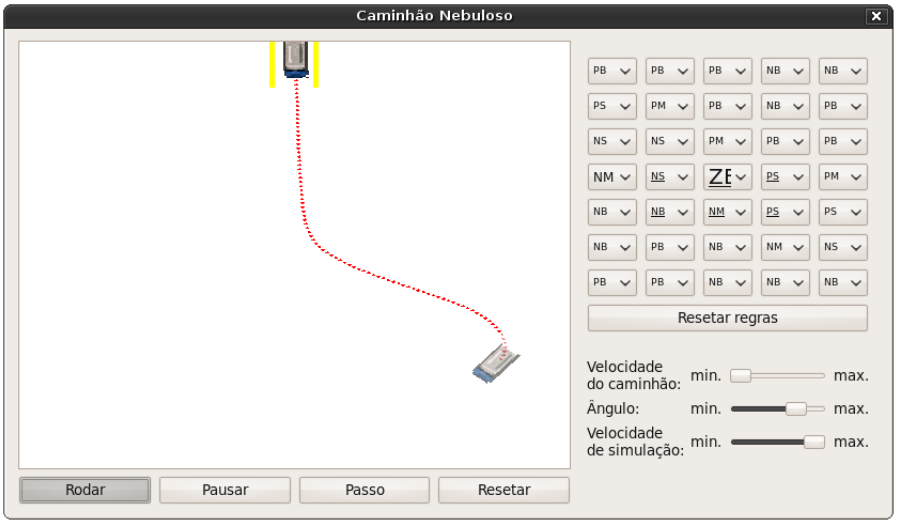
Os conjuntos de ângulos utilizados foram os mesmos definidos no caso anterior. Outra mudança realizada foi nos valores das funções *PS, PM, PB, NS, NM, NB*, que assumiram os valores 7, 14, 21, -7, -14, -21 respectivamente. A FAM inicial definida pode ser vista na imagem abaixo:

PB	PB	PB	NB	NB
PS	PM	PB	NB	PB
NS	NS	PM	PB	PB
NM	NS	ZE	PS	PM
NB	NB	NM	PS	PS
NB	PB	NB	NM	NS
PB	PB	NB	NB	NB

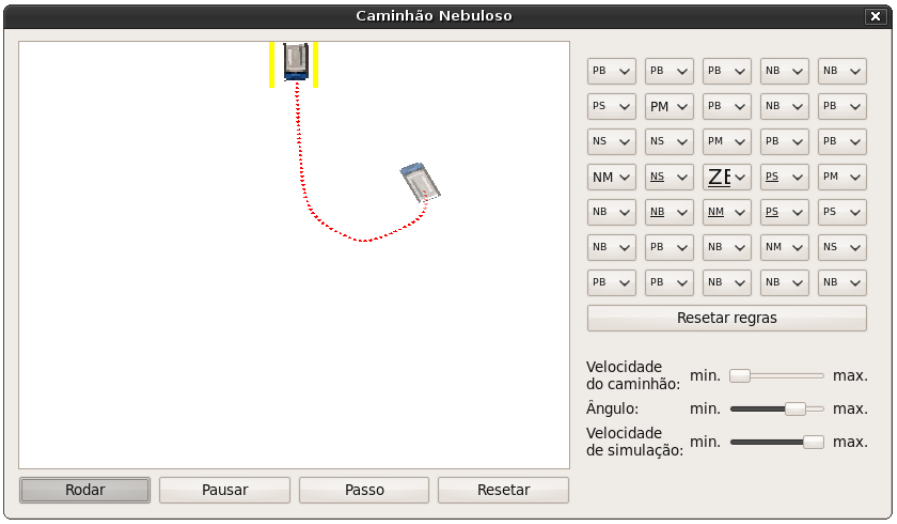
Com essa nova configuração, o caminhão passou a estacionar mais no centro da vaga. O aumento da precisão pode ser visto nos exemplos abaixo:



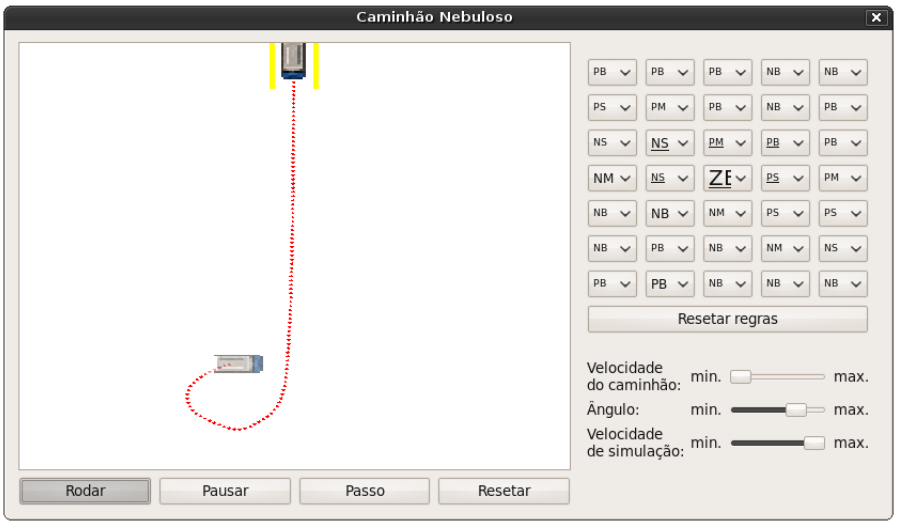
Exemplo 1



Exemplo 2



Exemplo 3



Exemplo 4

Essa foi a configuração deixada no código enviado, para que sejam realizados outros testes a critério do professor. É sempre possível modificar a FAM através da interface para se verificar outras situações, porém o único modo de editar os conjuntos de posição e de ângulo é através do código-fonte do programa.

Como rodar

Linux

Para rodar em alguma distribuição Linux, execute o arquivo Main.py por linha de comando:

```
>: python Main.py
```

Será necessário ter instalado o pacote de interface gráfica PyQt. Em distribuições baseadas em Debian, como o Ubuntu, pode-se utilizar o seguinte comando para instalar o pacote:

```
>: sudo aptitude install pyqt4-dev-tools
```

Windows

Para rodar o programa com o sistema operacional Windows, é necessário instalar os seguintes programas:

1. Python versão 2.6.5
    - Página da linguagem Python, na seção de Downloads: http://www.python.org/download/
    - Link direto para a versão requerida: http://www.python.org/ftp/python/2.6.5/python-2.6.5.msi
2. PyQt 4
    - Página do *framework* PyQt, na seção de Downloads: http://www.riverbankcomputing.co.uk/software/pyqt/download
    - Link direto para download: http://www.riverbankcomputing.co.uk/static/Downloads/PyQt4/PyQt-Py2.6-gpl-4.7.3-2.exe

Após a instalação, deve ser executado o arquivo Main.py contido na pasta "Código-fonte", presente junto ao relatório. Não foi criado um arquivo executável já com a biblioteca gráfica para ser rodado em Windows pois o programa excederia o tamanho limite para envio via Moodle.

Referências Bibliográficas

[1] Riverbank Computing. *Bindings* para a linguagem python do framework qt. http://www.riverbankcomputing.co.uk/software/pyqt.

[2] Python Software Foundation. Linguagem de programação python. http://python.org/.

[3] Nokia. Framework para interface. http://qt.nokia.com/.