Ensinando Técnicas de Inteligência Artifical (utilizando o Jogo da Velha)

Prof. Dr. Luciano Antonio Digiampietri

Escola de Artes, Ciências e Humanidades da USP

Roteiro

- Contexto Educativo
- Descrição do Jogo da Velha
- Técnicas de IA utilizadas
- Projeto *JogoDaVelhaII*
- Atividades

Contexto Educativo

- O objetivo desta apresentação é aplicar algumas técnicas de Inteligência Artificial na criação de *bots* (programas de computador que jogam de maneira automática) para o Jogo da Velha.
- As atividades presentes aqui podem ser usadas para exemplificar conceitos de Inteligência Artificial (busca por soluções, sistema especialista, representação de conhecimento, etc).

Descrição do Jogo

- Criaremos quatro *bots* para o Jogo da Velha.
- O Jogo da Velha contém um “tabuleiro” 3 x 3. Os dois jogadores alternam em colocar uma “peça” (tipicamente um X ou um O) no tabuleiro. Ganha quem conseguir formar uma sequência de três peças do seu tipo (três X ou três O). A sequência pode ser horizontal, vertical ou diagonal.

Técnicas de IA utilizadas

- Utilizaremos (e revisaremos brevemente) quatro técnicas de Inteligência Artificial:
  - Sistema Baseado em Regras (sistema especialista)
  - Busca Mini-Max
  - Busca Mini-Max com poda Alfa-Beta
  - Geração e uso de dicionário de jogadas

- A seguir descrevemos brevemente cada uma dessas técnicas.

Sistema Especialista

- “Um Sistema especialista é uma classe de programa de computador desenvolvido por pesquisadores de Inteligência artificial durante os anos 70 (1970) e aplicado comercialmente durante os anos 80 (1980). Em síntese, são programas constituídos por uma série de **regras** que analisam informações (normalmente fornecidas pelo usuário do sistema) sobre uma classe específica de problema (ou domínio de problema).” **[Wikipedia]**

Sistema Especialista

- Um sistema especialista tenta simular as ações de um especialista no domínio em que está sendo aplicado.
- No nosso caso, desenvolveremos regras para que o *bot* jogo como um bom jogador de Jogo da Velha.

Busca Mini-Max

- “Em teoria da decisão, minimax (ou minmax) é um método para minimizar a perda máxima possível. Pode ser considerado como a maximização do ganho mínimo (maximin). Começa-se com dois jogadores 0-0 da teoria dos jogos, cobrindo ambos os casos em que os jogadores tomam caminhos alternados (por rodadas) ou simultaneamente. Pode-se estender o conceito para jogos mais complexos e para tomada de decisão na presença de incertezas. Nesse caso não existe outro jogador, as consequências das decisões dependem de fatores desconhecidos.” **[Wikipedia]**

Busca Mini-Max

- O objetivo de um algoritmo MiniMax, no caso de jogos, é decidir qual a **melhor** jogada que o jogador poderá fazer de forma a **minimizar** a melhor jogada possível do adversário.
- Para poder maximizar a nossa jogada (minimizando a do adversário) é necessário dar notas para os estados do tabuleiro.
- Tipicamente este algoritmo recebe como parâmetro a quantidade de jogadas que serão avaliadas (por exemplo, 2 minhas e 2 do adversário).

Busca Mini-Max

- No Jogo da Velha, podemos nos dar ao luxo de darmos nota apenas nos tabuleiros que indicam fim de jogo (nota máxima para vitória, mínima para derrota e neutra para empate).
- Desta forma, não haverá necessidade de desenvolver uma fórmula mais complexa para o cálculo do valor de qualquer estado do tabuleiro e nem receber como parâmetro o número de níveis (jogadas) que deverão ser avaliadas.

Busca MiniMax com poda Alfa-Beta

- Algoritmos MiniMax exploram um espaço de busca muito grande e por isso podem não ser muito eficientes (em termos de tempo de execução).
- Por exemplo, para a primeira jogada do jogo da velha e fazendo uma busca em apenas dois níveis (duas jogadas minhas e duas do adversário) o algoritmo terá que verificar o seguinte número de combinações:

  9x8x7x6 = 3.024

- Para um jogo mais complexo, ou um número maior de níveis este número cresce exponencialmente.

Busca MiniMax com poda Alfa-Beta

- Uma maneira de deixar o algoritmo mais eficiente é não gastar tempo calculando estados irrelevantes.
- Já que o algoritmo é baseado em Maximizar e Minimizar valores, há duas situações que ele pode "podar" o espaço de buscas:
  - Quando estiver maximizando e encontrar o máximo valor possível;
  - Quando estiver minimizando e encontrar o mínimo valor possível.
- Além dos valores máximo e mínimo possíveis, podemos generalizar essa idéia para um limiar mínimo (alfa) e um limiar máximo (beta).

Dicionário de Jogadas

- Dicionários de Jogadas são tipicamente utilizados para os primeiros movimentos de jogos complexos (como Xadrez e Go).
- Para o Jogo da Velha, o dicionário pode conter todas as jogadas possíveis.
- O interessante aqui é desenvolver a aquisição do conhecimento (para montar o dicionário), a representação desse conhecimento e sua recuperação (para se executar uma jogada).

Dicionário de Jogadas

- A **aquisição de conhecimento** utilizará a busca MiniMax para todas as jogadas possíveis.
- A **representação de conhecimento** será feita via codificação do estado atual do tabuleiro (poderíamos simplesmente linearizar o tabuleiro, mas isto não seria muito eficiente).
- A **recuperação de conhecimento** será feita por meio da decodificação do conhecimento armazenado.

Projeto *JogoDaVelhaII*

- O Projeto JogoDaVelhaII contém os seguintes arquivos (além dos executáveis):
  - **dicionario.txt**
  - **geracaoDeDicionarioUtil.c**
  - **gerarDicionario.c**
  - **gerarDicionario_entrada.txt**
  - **jogadorDicionario.c**
  - **jogadorEspecialista.c**
  - **jogadorMiniMax.c**
  - **jogadorMiniMaxAlfaBeta.c**
  - **jogoDaVelhaDicionario.c**
  - **jogoDaVelhaEspecialista.c**
  - **jogoDaVelhaMiniMax.c**
  - **jogoDaVelhaMiniMaxAlfaBeta.c**
  - **jogoDaVelhaUtil.c**

JogoDaVelhaUtil.c

- Possui toda a logica e estruturas para um jogo da velha. Serve de base para o desenvolvimento de bots ou de um programa interativo para jogar jogo da velha.
- Segue a lista de alguns de seus métodos:
    - **void inicializarJogo(JOGO * j,jogador jog)**
    - **void imprimirTabuleiro(JOGO j)**
    - bool **fimDeJogo(JOGO j, jogador * vitorioso)**
    - bool **inserirJogada(JOGO * j, int posicao, jogador jog)**
    - LISTA_DE_JOGADAS **listaDeJogadas(JOGO * j)**

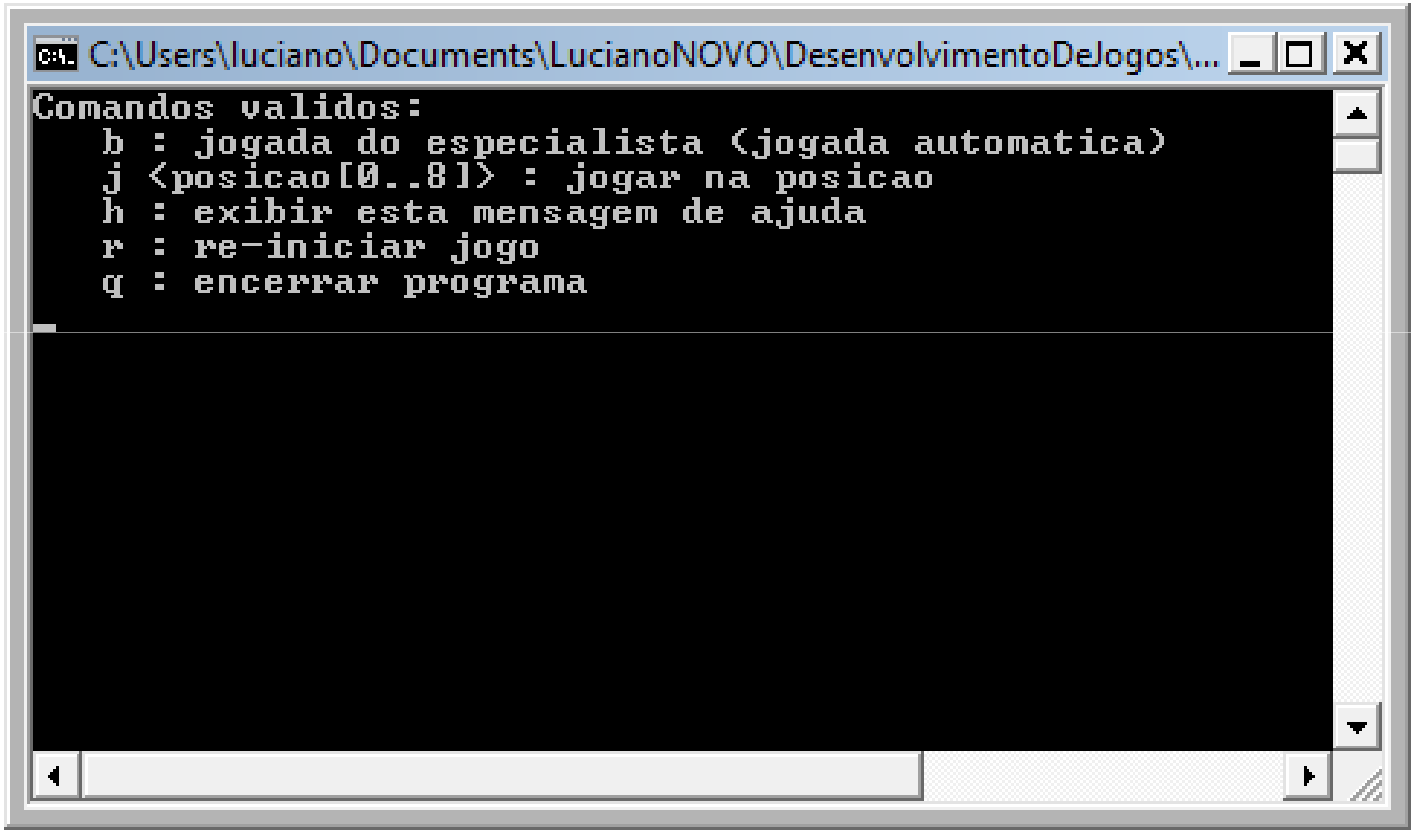
JogoDaVelhaEspecialista.c

- Dado o estado de um jogo (definido em JogoDaVelhaUtil.c), retorna a jogada que seria feita pelo especialista.
- O conjunto de regras é utiliza a seguinte lógica (na ordem apresentada):
  - Se possível faça a jogada com a qual você ganhe o jogo;
  - Se possível faça a jogada com a qual você não perca o jogo na próxima jogada do adversário;
  - Se possível jogue na posição central do tabuleiro ...
  - Se possível jogue nas diagonais ...

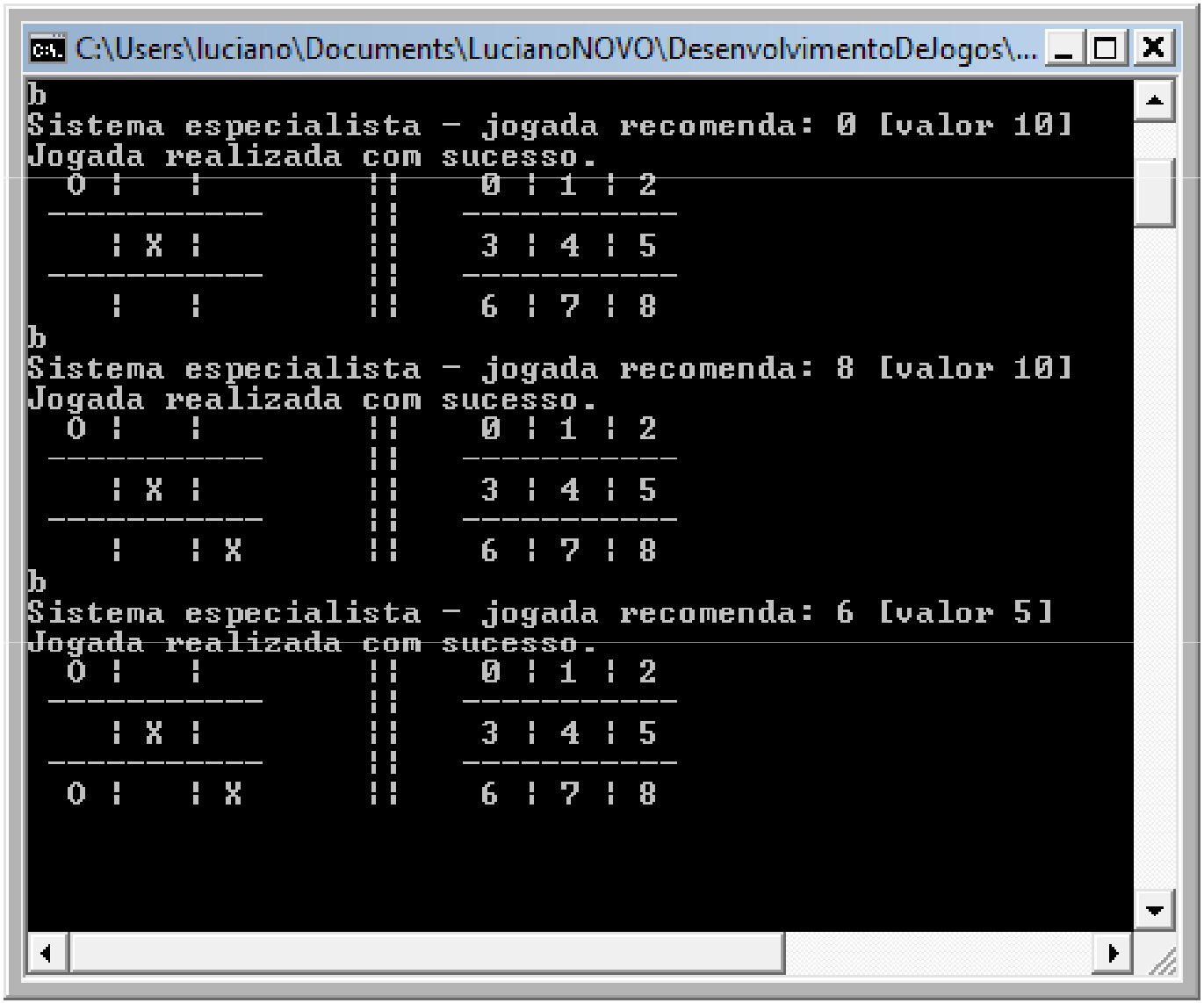
Veja o arquivo JogoDaVelhaEspecialista.c para detalhes e implementação de cada uma das regras.

JogadorEspecialista.c

- Sistema que interage com o usuário para se jogar Jogo da Velha. O *bot* utiliza o sistema especialista descrito anteriormente.



JogadorEspecialista.c



JogoDaVelhaMiniMax.c

- Dado o estado de um jogo (definido em JogoDaVelhaUtil.c), retorna a melhor jogada baseada na busca MiniMax.
- A idéia geral do algoritmo é a seguinte:
  - Para todas as jogadas possíveis
    - Verifique se a é fim de jogo
      - Se sim: atribua os valores:
        - 99999999 **– nível atual** para vitória
        - - 99999999 **+ nível atual** para derrota
        - 0 para empate
      - Se não, chame recursivamente o algoritmo alternando entre minimizar e maximizar.
  - Retorna a jogada que minimiza ou maximiza (dependendo da situação)

JogoDaVelhaMiniMax.c

- O valor **nível atual** significa a jogada que está sendo avaliada (terceira, quarta, etc).
- Esta variável é utilizada para:
  - dadas n jogadas vitoriosas, escolher aquela que levará a uma vitória mais rápida;
  - dadas n jogadas de derrota, escolher aquela que postergará a derrota;

JogadorMiniMax.c

- Sistema que interage com o usuário para se jogar Jogo da Velha. O *bot* utiliza a busca MiniMax descrita anteriormente.

JogoDaVelhaMiniMaxAlfaBeta.c

- Dado o estado de um jogo (definido em JogoDaVelhaUtil.c), retorna a melhor jogada baseada na busca MiniMax com poda Alfa-Beta.
- A idéia geral do algoritmo é a seguinte:
  - Para todas as jogadas possíveis
    - Verifique se a é fim de jogo
      - Se sim: atribua os valores:
        - 10 para vitória => **sair do laço se estiver maximizando**
        - - 10 para derrota => **sair do laço se estiver minimizando**
        - 0 para empate
      - Se não, chame recursivamente o algoritmo alternando entre minimizar e maximizar.
  - Retorna a jogada que minimiza ou maximiza (dependendo da situação)

JogoDaVelhaMiniMax.c

- Note que não foi necessário utilizar variáveis Alfa e Beta, pois o valor de Alfa seria -10 (valor da derrota) e Beta seria 10 (valor da vitória).
- As linhas em destaque do algoritmo mostrar exatamente o momento da poda Alfa ou Beta.
- Não utiliza uma variável que indica o nível da jogada, desta forma, não faz nenhum tipo de seleção entre, por exemplo, jogadas vitoriosas, simplesmente escolhe a primeira.

JogadorMiniMaxAlfaBeta.c

- Sistema que interage com o usuário para se jogar Jogo da Velha. O *bot* utiliza a busca MiniMax descrita anteriormente.

JogoDaVelhaDicionario.c

- Dado o estado de um jogo (definido em JogoDaVelhaUtil.c), retorna a jogada salva no dicionário de jogadas para o estado atual do jogo.
- *Bot* utiliza dicionário de jogadas (similar a um sistema especialista mas ao invés de regras todas as possibilidades já foram previamente mapeadas - 4520 jogadas/regras).
- Dicionário ocupa 38KB em disco (dicionario.txt) e provê a execução eficiente do jogo (que lê o dicionário num formato interno, para fácil acesso mas com consumo maior de memoria, gasta, ao todo, 216KB de RAM).

JogoDaVelhaDicionario.c

- As jogadas são geradas por uma versão modificada do algoritmo de busca MiniMax (sem poda):
  - geracaoDeDicionarioUtil.c
    - Contém a busca MiniMax propriamente dita e um método para codificar o estado do jogo.
  - gerarDicionario.c
    - Recebe alguns parâmetros de entrada para garantir a geração de todas as jogadas possíveis (é parecido com o sistema interativo jogadorMiniMax.c).
  - gerarDicionario_entrada.txt
    - Contém os parâmetros de entrada para o gerarDicionario.c

Codificação

- A codificação do estado atual é feita da seguinte maneira:
  - Já que o tabuleiro é formado por 9 “casas” e cada uma pode ter 3 valores (‘x’, ‘o’, ou ‘ ’[vazio], que são mapeados para +1,-1 e 0), simplesmente é feita uma operação com potências de 3 para converter este estado para um número de até 5 dígitos, conforme o método do próximo slide.
- Exemplo de códigos e jogada para o código (as jogadas estão após o código e variam de 0 a 8):
  - **1637 0**
  - **1643 8**
  - **1645 8**
  - **1691 0**
  - **1697 8**

Codificação

```
int indiceDeUmDadoTabuleiro(JOGO *j){
   int i, valor[3];
   if (j->jogadorAtual == j1){
     for (i=0;i<3;i++) {
        valor[i] = (j->A[3*i]+1)*9+(j->A[3*i+1]+1)*3+(j->A[3*i+2]+1)+1;
      }
  }else{
    for (i=0;i<3;i++) {
      valor[i] = (j->A[3*i]*-1+1)*9+(j->A[3*i+1]*-1+1)*3+(j->A[3*i+2]*-1+1)+1;
    }
  }
  return valor[0] +27*(valor[1]+27*valor[2])-1;
}
```

Decodificação

- O dicionário de jogadas é lido em um arranjo de variáveis do tipo *char* de tamanho 19683.
- Não existe uma decodificação propriamente dita, o programa JogoDaVelhaDicionario.c codifica o estado atual do jogo e verifica qual é a jogada equivalente para aquele estado (utilizando o método de codificação apresentado anteriormente).

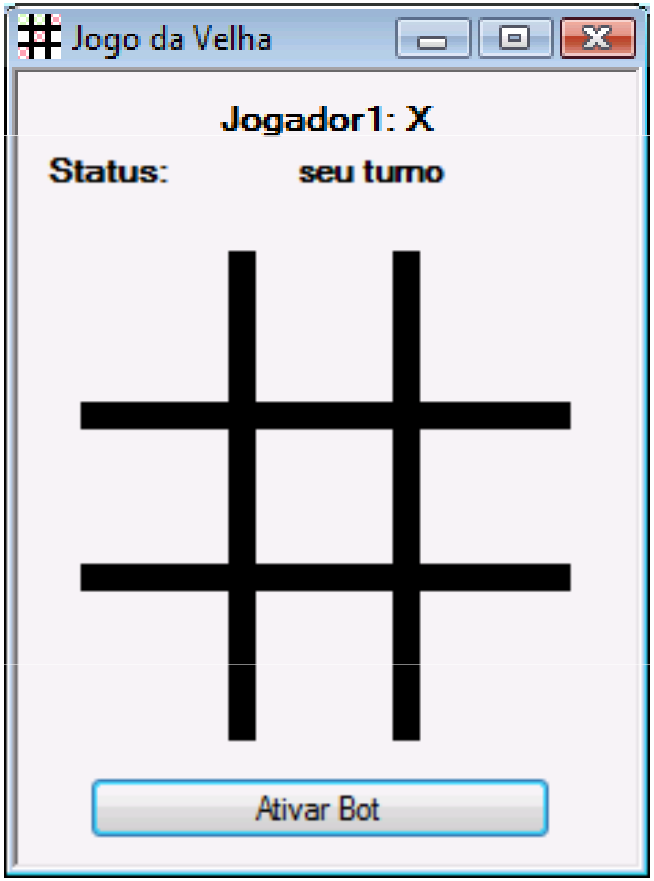
JogadorDicionario.c

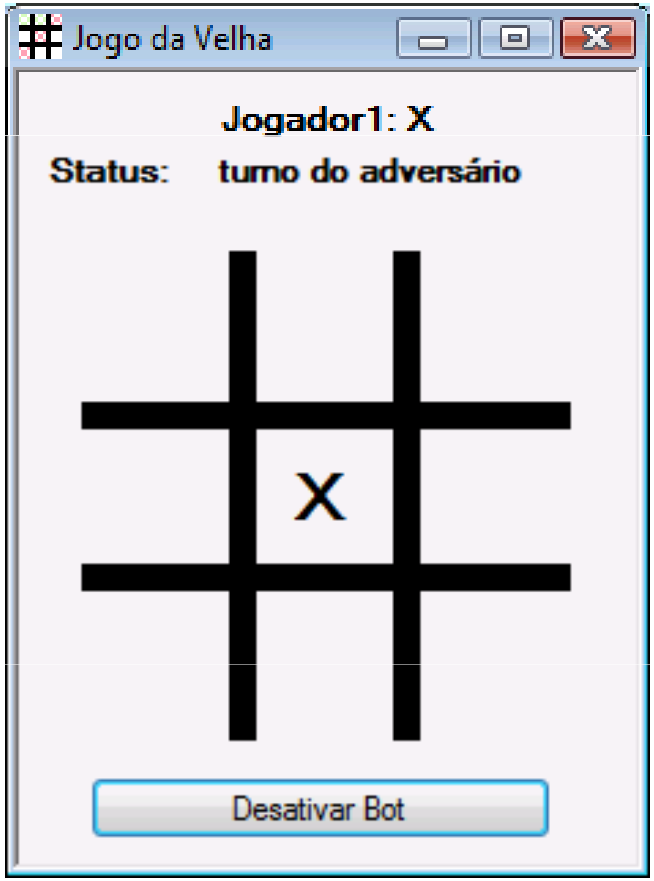
- Sistema que interage com o usuário para se jogar Jogo da Velha. O *bot* utiliza o dicionário de jogadas descrito anteriormente.
- Este sistema, ao ser executado, cria uma instância do *JogoDaVelhaDicionario.c* e copia para a memória todo o dicionário de dados (que está armazeando no dicionario.txt).

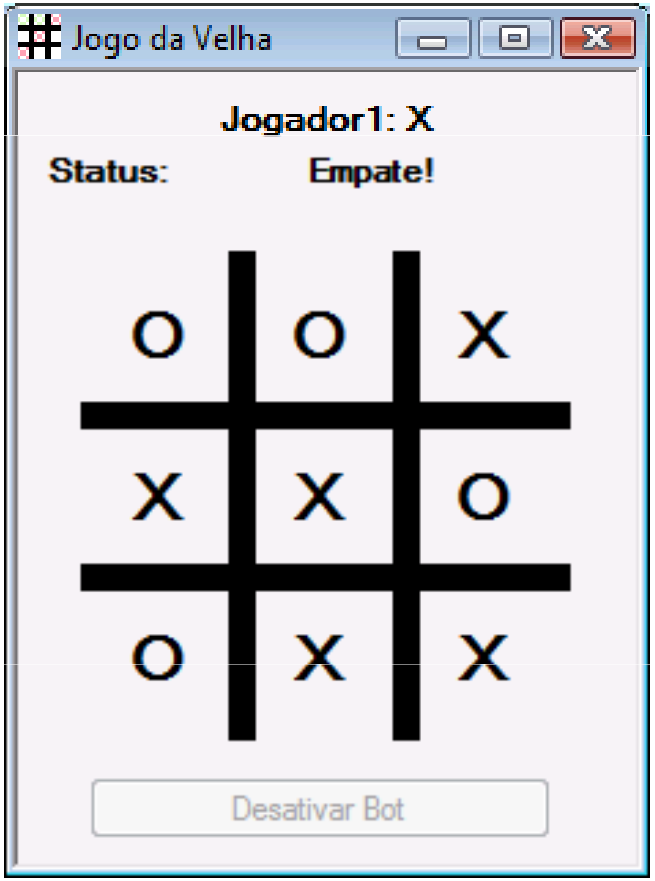
Atividade 1

- Como atividade para esta apresentação, sugere-se que o aluno insira no projeto gráfico *JogadorJogoDaVelha* os quatro métodos de inteligência artificial propostos e permita ao usuário selecionar qual método ele deseja utilizar ao ativar o *bot*.
- A versão atual deste projeto contém apenas um *bot* baseado num sistema de regras (Sistema Especialista).
- O próximo slide contém cópias de tela da versão atual desse projeto

Projeto JogadorJogoDaVelha







Atividade 2

- Mude a codificação do Dicionário de Jogadas para armazenar os estados do jogo como um conjunto de nove caracteres: ‘x’, ‘o’ ou ‘ ’ (espaço em branco).
- Utilize um hash ao invés de um arranjo para armazenar essas informações dentro do programa JogoDaVelhaDicionario.txt

http://www.uspleste.usp.br/digiampietri/jogos/