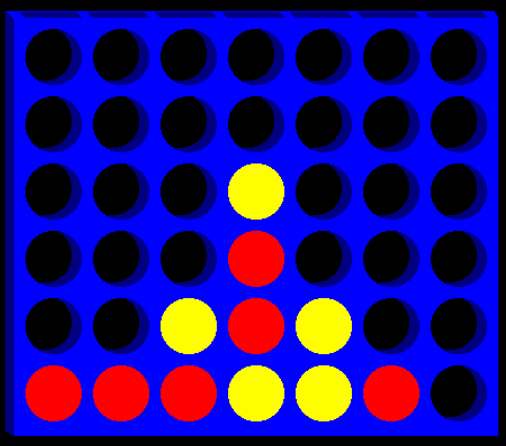
**Objetivos**

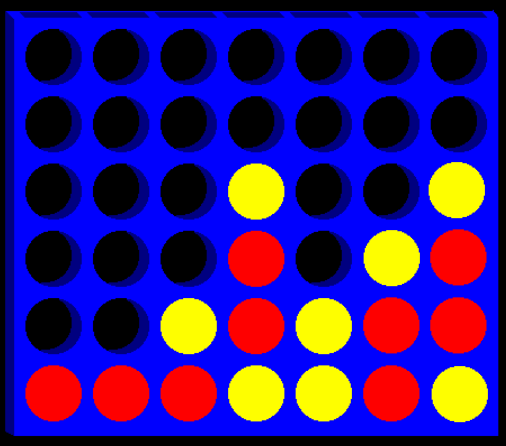
Este trabalho objetiva a prática dos conceitos vistos em aula na implementação da IA de um jogo simples.

**O Jogo**

O jogo escolhido para este semestre foi o Liga 4 (ou Connect 4, For-in-a-Row, ...), que consiste em um tabuleiro de dimensões 6x7 onde os jogadores irão colocar peças, intercaladamente em uma das 7 colunas. A peça colocada irá ocupar a última casa disponível nesta coluna.

  
situação de jogo

O vencedor será o jogador que conseguir ligar quatro peças consecutivas, seja na horizontal, vertical ou diagonais, como na figura abaixo:

  
situação de vitória da amarelas

**Implementação**

O jogo foi implementado para a web, rodando diretamente no navegador. A interface foi descrita em HTML e CSS, e a IA foi programada em Javascript.

O navegador utilizado para os testes, bem como para o campeonato e apresentação foi o Chrome Canary versão 14, mas o jogo deverá funcionar em qualquer navegador que preencha os requisitos que serão mencionados a frente.

A implementação foi dividida em duas partes:

* A interface, que receberá o input do jogador, computará um possível caso de vitória e repassará a jogada (no caso de jogo contra o computador) ao worker;
* O Worker, que é o trecho de código que implementará a inteligência artificial do computador.

**Interface**

A interface é descrita em HTML e CSS, com a interação sendo feita através de pequenas chamadas de eventos em Javascript. Foi utilizado jQuery para facilitar a manipulação do DOM e dos eventos associados a interação com o usuário.

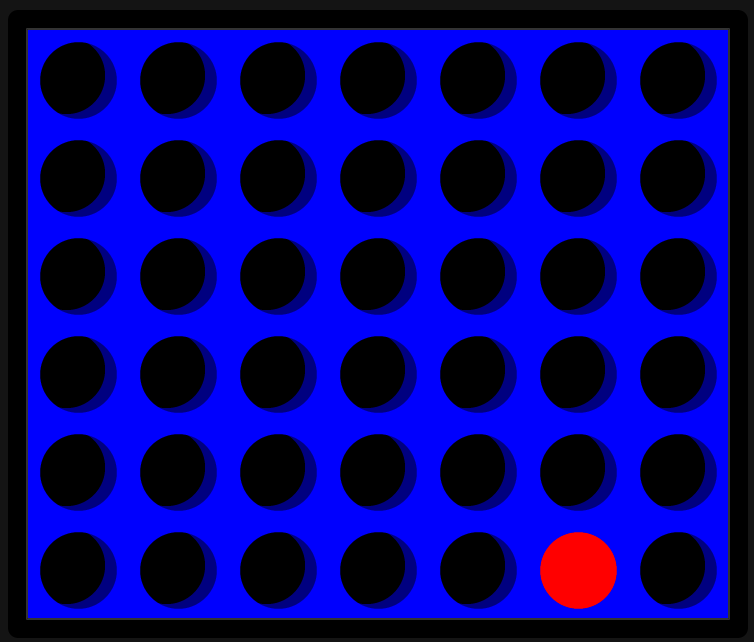
  
Tela inicial do programa

A tela inicial é composta de um logo simples, com opções para configuração do jogo conforme especificação do professor.

Uma das opções, utilizadas para o caso de uma partida Jogador vs Computador é a dificuldade. Ela irá configurar a profundidade alcançada pelo algoritmo MiniMax, bem como a “estupidez” do computador. Estes dados serão abordados com maior abrangência na sessão “Worker” deste relatório.

Uma vez iniciada, a interface aguardará uma ação do usuário, no caso do mesmo ser o primeiro a jogar, ou enviará uma solicitação de jogada ao Worker, no caso contrário.

O usuário deverá clicar na coluna onde deseja colocar a peça. Ao clicar, a interface irá pintar uma peça da cor definida na tela anterior na posição correta no tabuleiro, conforme a figura abaixo:

  
Jogada inicial do jogador humano

No caso de jogos humano vs humano, a interface permanecerá inativa até que o segundo jogador efetua a sua ação, e assim em diante, até uma situação de vitória ou empate.

No caso de jogo contra o computador, o Worker é notificado da jogada, e o mesmo retorna uma coluna para que seja executada a jogada do computador. A interface irá pintar uma peça de cor oposta a do jogador na posição indicada.

A interface notifica casos de vitória bloqueando a tela e apresentando a mensagem “Amarelas/Vermelhas venceram!”, conforme a figura abaixo:

  
Situação de vitória das peças vermelhas

Todas as casas do tabuleiro são implementadas através de um elemento “div”, que é marcado como classe “red” ou “yellow” dependendo da peça em cima. A estrutura de dados para controle do tabuleiro é a própria ordenação das divs. Na classe Interface, existe uma função que faz o “dump” do tabuleiro para uma string. O que ele faz é pegar todas as divs dentro do tabuleiro, e para cada div mapear o caracter “-“ para posição vazia, “y” para posição com uma peça amarela e “r” para posição com uma peça vermelha. Assim temos um string de 42 caracteres que é usado para verificar jogadas vitoriosas pela interface.

Saliento que a mesma estrutura de dados não é usada no Worker. O motivo é que ambas foram implementadas em separado, a interface primeiro. Este tipo de estrutura de dados para a interface é bastante conveniente, pois facilita a programação.

**Worker**

Worker (ou WebWorker) em Javascript é uma espécie de thread, que executa em paralelo com o fluxo principal do script da página. Ele pode se comunicar com este fluxo através de troca de mensagens.

Esta funcionalidade é nova, está sendo trabalhada para o HTML5 (que ainda é um draft), mas já está presente na maioria dos navegadores modernos, em suas últimas versões. Dos navegadores classe-A, o único que ainda não implementa WebWorkers, na modalidade utilizada neste trabalho é o Internet Explorer. Para todos os outros, deve funcionar sem problemas. Recomenda-se o uso do Chrome, em sua última versão estável.

Aqui usamos o Worker para implementar a IA do jogo, ou seja, o MiniMax que irá retornar a melhor jogada para o computador.

Neste caso, foi usada como estrutura de dados uma matriz n x m, onde n é o número de colunas e m o número de linhas desejadas. O padrão é 7 x 6.

**Minimax**

A implementação do MiniMax segue o exemplo simples visto em aula. A função minMove é chamada, repassando o id do jogador atual, e o nível da árvore que queremos alcançar. minMove chamará maxMove, e assim sucessivamente e intercaladamente, até que os nós folha (dada a altura selecionada) seja encontrado.

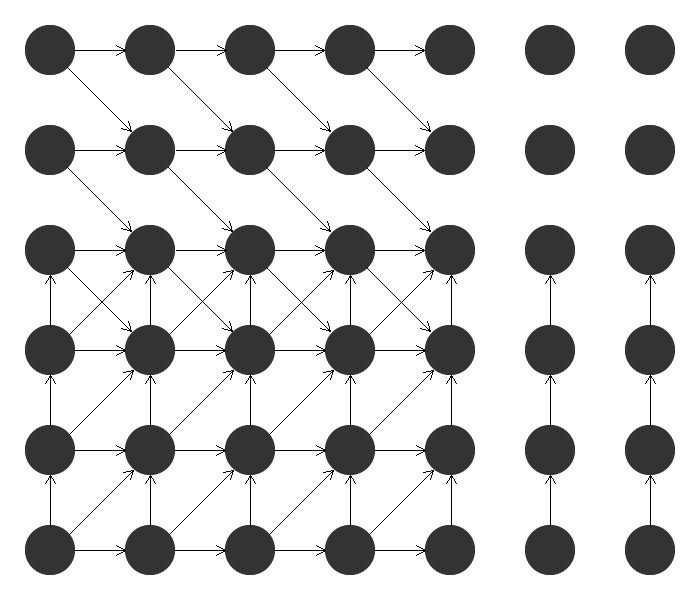
Ao encontrar nós folha, o minMove (ou maxMove) irá retornar a melhor avaliação dentre eles, e assim a melhor jogada irá emergir dos nós folha da árvore, até chegar na raiz, que retornara a coluna que o computador deverá jogar.

**Função de Avaliação**

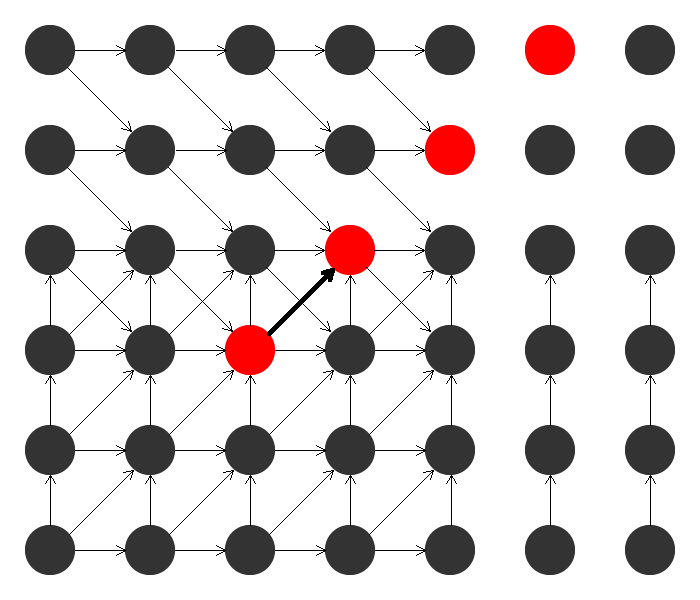
A função de avaliação deverá retornar um valor para uma determinada configuração de tabuleiro. Para isso, valoramos todas as posições do mesmo, levando em conta 4 posições subjacentes que configuram jogadas vitoriosas.

A função de avaliação é portanto um loop, que irá iterar sobre todas as possíveis jogadas vitoriosas e valorá-las.

Para iterar sobre estas jogadas, foi criada uma matriz bidimensional que indica quais são elas. Para cada posição, determina-se se deve-se procurar por quatro peças na horizontal, vertical ou diagonal. Como fazemos uma varredura da esquerda para a direita, e de baixo para cima, podemos excluir algumas jogadas repetidas. Ficamos no final com uma matriz que deverá representar as jogadas conforme o diagrama abaixo:

  
diagrama de possíveis jogadas

Note que para cada posição, uma seta indica um possível lugar para procurar por 4 em sequência. Usemos como exemplo o estado abaixo:

  
Possível jogada vitoriosa

A seta em negrito indica que iterando linha e coluna 4 vezes encontramos uma possível jogada.

A função de avaliação irá iterar sobre todas estas possíveis jogadas, calcular a força delas segundo o pseudo-algoritmo abaixo:

str(p1, p2) ->

x = 0

if (p1 == 1) x += 1

if (p1 == 2) x += 10

if (p1 == 3) x += 100 - p2 \* 20

if (p1 == 4) x = WIN

if (p2 == 1) x -= 1

if (p2 == 2) x -= 10

if (p2 == 3) x += 100 - p1 \* 20

if (p2 == 4) x = -WIN

return x

Algorítmo para valoração de jogada

Onde p1 é o número máximo de peças alinhadas do player 1, e p2 o número máximo de peças consecutivas do player 2.

No final da avaliação, teremos um valor para aquela configuração de tabuleiro. Essa é a avaliação daquela jogada.