UNIVERSIDADE DO MINDELO – SÃO VICENTE CABO VERDE

CURSO DE LICENCIATURA EM INFORMÁTICA DE GESTÃO

PROJECTO DE CONCLUSÃO DO CURSO

JOGO DE ORIL PARA TELEMÓVEL

JORGE COSTA

SAMUEL SANTOS DE LIMA

MINDELO, JULHO DE 2012

Monografia apresentada a Universidade do Mindelo como parte integrante do projecto final de Curso de Licenciatura em Informática de Gestão.

Mindelo, 18 de Julho de 2012.

O Orientador

Engº Joao Dias

Dedicatória Jorge Costa

Aquela que foi nossa mãe e nosso pai e que abdicou da sua juventude para que eu e meus irmãos tivéssemos esses pequenos momentos

Dedicatória Samuel Lima

Para Lídia Arianne, minha filha cujo teu nascimento me fortaleceu e me apoiou na concretização deste trabalho.

AGRADECIMENTO JORGE

Agradeço a todos aqueles que sempre estiveram do meu lado nos momentos bons e nos maus, e aqueles que directamente ou indirectamente ajudaram para e elaboração deste trabalho,

A todos o meu obrigado!

AGRADECIMENTOS SAMUEL

Os meus agradecimentos vão em primeiro lugar para minha namorada Ludmila Juff, pelo seu apoio e carinho.

Outro especial agradecimento dirige-se a minha mãe e ainda aos meus irmãos que me deram apoio e força para continuar.

Gostaria ainda de agradecer o meu colega Jorge Costa, o Reitor Dr. Albertino Graça, o Engº João Dias, e todos os funcionários da Universidade do Mindelo.

E por fim um agradecimento sincero para todos os meus amigos que de alguma forma ou de outra contribuíram para a realização deste trabalho.

A todos o meu obrigado!

RESUMO

A Inteligência Artificial hoje em dia tem várias aplicações e uma delas é em jogos de computadores. O objectivo deste trabalho é utilizar técnicas de Inteligência Artificial para desenvolver um agente capaz de jogar o jogo Oril e para tal, utiliza-se o algoritmo Minimax, que busca a árvore de jogadas da partida e tenta estimar a melhor jogada numa determinada profundidade avaliando os estados encontrados. Por outro lado pretende-se também fazer a implementação do jogo de oril num smartphone, tendo em conta a mobilidade e as capacidades de processamento desses equipamentos que nos permite utilizar elementos multimédia, para tornar a experiência de jogar contra o computador ou outro utilizador mais real.

Palavras-chaves: Inteligência Artificial, Teoria dos Jogos, Smartphone, Oril.

ABSTRACT

Artificial Intelligence today has many applications and one is computer games. The objective of this work is to use artificial intelligence techniques to develop an agent capable of playing the game "Oril" and such as , we use the Minimax algorithm that seeks the tree of moves of the game and tries to estimate the best move in a certain depth by evaluating the states found. On the other hand we intend to make the implementation of the game "Oril" on a smartphone, counting on the mobility and the processing capabilities of the equipment that allows us to use multimedia elements to make the experience of playing against the computer or another user more real .

Keywords: Artificial Intelligence, Game Theory, Smartphone, Oril.

ÍNDICEREMISSIVO

Conteúdo

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

SIGLAS E ABREVIATURAS UTILIZADAS

IA – Inteligência Artificial

ENIAC–Electrical Numerical Integrator and Computer

SDK - Software Development Kit

MODELO ER – Modelo Entidade Relacionamento

MACOSX - Macintosh Operation System Ten

2D – Duas Dimensões

NICKNAME – Nome de utilizador

HTTP – Hypertext Transfer Protocol

URL–Uniform Resource Locator

PHP–Hypertext Preprocessor

VPS – Virtual Private Server

SMS– Short Message Service

WI-FI – Wireless Fidelity

3G – Terceira Geração

iOS – iPhone Operating System

1. INTRODUÇÃO

Num país onde sete em cada dez habitantes possuem um telemóvel, e sendo a maioria jovem, há um enorme potencial para os jogos baseados na tecnologia móvel. Desde o aparecimento do Snake, para os Nokia 3210, que este fenómeno tem sido aproveitado pelos fabricantes e outros desenvolvedores da área. Os smartphones de hoje possuem processadores especializados para aplicações e jogos, dando aos criadores de jogos liberdade suficiente para criar jogos que correm em tempo real, com capacidades equivalentes as novas consolas de jogos portáteis.

Assim sendo, tendo em conta que o jogo de oril é um dos jogos de tabuleiro mais jogado pelos cabo-verdianos, a realização deste trabalho tem como meta o desenvolvimento de um programa que jogasse oril com a finalidade de imitar processos de tomada decisão com o uso da inteligência artificial e também aproveitando os recursos de rede disponíveis no telemóvel e dar a oportunidade de dois utilizadores disputarem uma partida via internet.

O jogo de oril é uma boa escolha para exploração da inteligência da máquina, pois o problema é prontamente definido tanto nas operações permitidas como na meta final e não é tão simples a ponto de ser trivial e nem tão difícil de obter uma solução satisfatória.

1.2. MOTIVAÇÃO

A realização deste projecto originou-se de uma proposta do coordenador do curso de criar algo inovador dentro da nossa realidade cabo-verdiana. E foi dentro desse contexto, que pensou-se no desenvolvimento do jogo de oril para telemóveis. Sendo a inteligência artificial uma área que não é estudada a nível do nosso curso, aceitamos esse desafio com intuito de aprendermos algo sobre o assunto e aplica-lo no jogo de oril que é de origem Africana e muito jogado aqui em cabo verde.

1.3. OBJECTIVOS

O principal objectivo deste projecto é de criar um programa para um telemóvel, mais especificamente um iPhone, capaz de jogar oril com um nível de jogo comparável a um jogador mediano. O programa utilizará técnicas de inteligência artificial, empregando uma variedade de métodos para à melhoria do jogo, com algoritmos de busca MiniMax, para a decisão da melhor jogada. A sua implementação dará ao utilizador a oportunidade de jogar não só contra o computador, mas assim como poderá faze-lo contra outro utilizador através da internet.

1.4. METODOLOGIA

Para a realização deste trabalho fizemos algumas pesquisas dos fundamentos teóricos, regras e estratégias do jogo oril e também de outros jogos de tabuleiro como xadrez, damas e tic-tac-toe que já possuem várias aplicações e estudos a nível de inteligência artificial. De igual modo foram feitos pesquisas a nível de

inteligência artificial, teoria dos jogo e a implementação do jogo utilizando a linguagem c.

Essas pesquisas foram feitas utilizando bibliografias existentes e algumas fontes na internet.

2. INTELIGÊNCIA ARTIFICIAL

A Inteligência Artificial (IA) é o ramo de ciência de computação que estuda formas de estabelecer comportamentos inteligentes a máquinas de forma a simular o comportamento humano[1].

Existente há décadas, e com o desenvolvimento acelerado da informática esta área da ciência é grandemente impulsionada, permitindo que novos elementos sejam rapidamente agregados à IA.

2.1. HISTÓRIA DA INTELIGÊNCIA ARTIFICIAL

Os estudos de IA iniciaram nos anos 1940, em torno desta iniciante ciência eram desenvolvidas apenas para procurar encontrar novas funcionalidades para o computador, ainda em projecto. Com a II Guerra Mundial este fato resultou na necessidade de desenvolver uma tecnologia voltada para a análise de balística, quebra de códigos e cálculos para projectar a bomba atómica. Surgia, assim, os primeiros grandes projectos de construção de computadores.

O desenvolvimento do computador, primeiramente impulsionado pela aplicabilidade militar e posteriormente comercial, mostrou-se viável. Seu rápido progresso, desde o surgimento dos primeiros computadores electrónicos (1943 - Collossus, na Inglaterra e 1946 - ENIAC, nos Estados Unidos) até o surgimento dos microcomputadores (na década de 70) demonstra que essa área recebeu grandes investimentos.

[1]Direne, Alexande. Universidade Federal de Panamá: Visão Geral Sobre Inteligência Artificial, http://www.nce.ufrj.br/GINAPE/VIDA/ia.htm, 2010-11-20, 18:10

O segundo grande passo foi dado nos Estados Unidos, em 1956, quando John McCarthy reuniu em uma conferência proferida ao Darmouth College, na Universidade de New Hampshire, vários pesquisadores de renome para estudar o que foi denominado por Minsky, McCarthy, Newell e Simon de Inteligência Artificial (IA), expressão utilizada para designar um tipo de inteligência construída pelo homem para dotar a máquina de comportamentos inteligentes.

A inteligência artificial é amplamente utilizada como um auxiliar que expande a capacidade de inteligência do homem e até mesmo o substitui em diversas funções. Isso se tornou possível em grande parte graças ao desenvolvimento dos sistemas especialistas, da Lógica Fuzzy e das Redes Neurais.

Actualmente, criar máquinas inteligentes não pode ser considerado uma ficção, a IA transformou essa ficção em um campo de estudo movido por uma meta que consome bilhões de dólares em projectos, os quais envolvem pesquisadores de instituições governamentais, militares, industriais e universitárias de todo o mundo[1].

3. TEORIA DOS JOGOS

A teoria dos jogos é um ramo da matemática aplicada que estuda situações de conflito de diversos tipos (sociais, económicos, políticos, militares, éticos, filosóficos, jornalísticos, etc.) de acordo com um modelo escolhido, cujas regras são mais ou

[1]Wikipedia: Inteligência Artificial, http://pt.wikipedia.org/wiki/Intelig%C3%AAncia_artificial, 2010-11-20, 18:38

menos rígidas, e mais ou menos conhecidas pelos jogadores. Assim, estes escolhem diferentes acções para tentarem melhorar o seu retorno.[1]

3.1. HISTÓRIA DA TEORIA DOS JOGOS

Uma primeira abordagem da Teoria dos jogos aplicando a matemática na resolução conflitos de interesse, foi feita por John Von Neumann em seus artigos de 1928 (" Zur Theorie der Gesellshaftsspiele") e 1937 ("A Modelof General Economic Equilibrium"), embora Emile Borel tenha publicado antes, entre 1921 e 1927, quatro notas introduzindo os conceitos de estratégias puras e mistas e a solução minimax, a qual é fundamental para a teoria dos jogos.

Entretanto, Borel considerou que o teorema minimax era em geral falso, apesar de tê-lo comprovado para casos especiais. Von Neumann provou o teorema para condições gerais e ainda criou a teoria dos jogos com mais de dois jogadores.

Em 1944, Von Neumann e o economista Oskar Morgenstern publicaram o livro clássico Theoryof Games and Economic Behavior, apresentando o teorema minimax como solução para jogos de soma zero com dois jogadores, além da fundamentação da teoria da utilidade, a qual é muito útil para situações de incerteza em economia.

Em 1950, baseado no trabalho de Melvin Dreshere Merrill Flood, Albert Tucker criou o Dilema do Prisioneiro, o mais conhecido problema na área de teoria dos jogos e aquele com maior influência nas ciências sociais.

[1]Wikipedia:Teoria dos Jogos, http://pt.wikipedia.org/wiki/Teoria_dos_jogos, 2010-11-21, 10:25

Entre 1950 e 1953, John Nash publicou quatro artigos importantes para a teoria de jogos não cooperativos e para a teoria de barganha. Em Equilibrium Pointsin N-Person Games (1950) e Non-cooperative Games (1951), Nash provou a existência de um equilíbrio estratégico para jogos não cooperativos - o equilíbrio de Nash - e sugeriu uma abordagem de estudo de jogos cooperativos a partir de sua redução para a forma não cooperativa. Nos artigos The Bargaining problem e Two-Person Cooperative Games, criou a teoria de barganha e provou a existência da solução de barganha de Nash.

3.2. CARACTERÍSTICAS DE UM JOGO

Para a realização de um jogo tem que haver pelo menos dois jogadores.

Jogada- é a maneira segundo a qual o jogo progride de um estágio a outro. Podem ser alternadas entre os jogadores de uma forma especificada ou ocorrer simultaneamente. Uma jogada consiste de uma decisão de um dos participantes ou de um resultado de um evento probabilístico.

Estratégia - é uma lista das escolhas óptimas para um jogador. Nesta lista já estão previstas todas as possíveis situações que o jogador poderá enfrentar. Assim, tendo uma estratégia, ele saberá o que fazer em qualquer situação, não importando o que seu oponente faça nem os resultados dos eventos probabilísticos.

No fim do jogo, cada jogador obtém um *payoff*. Podemos associar este número ao montante que foi ganho ou perdido, ou dizer, por exemplo, que o *payoff* é +1 para o ganhador, 0 se há um empate, e -1 para o perdedor.

3.3. REPRESENTAÇÃO DOS JOGOS

A aplicação da teoria dos jogos a um determinado caso de estudo, será objecto de modelos matemáticos bem definidos. E estes podem ser representados de duas formas:

FORMA NORMAL – A representação de jogos na forma normal é feita utilizando matrizes, que contem informações relativamente a jogadores, estratégias e pagamentos. Esta forma aplica-se quando não há alternância na tomada de decisão ou presume-se que os jogadores não têm conhecimento da acção dos outros.

O jogo dilema do prisioneiro, que retrata uma situação em que dois criminosos são presos por cometerem um crime, a polícia tem evidências para mantê-los presos por um ano, porém não para condená-los, os presos são colocados em celas separadas, para que não haja acordos prévios.

As decisões são simultâneas e um não sabe nada sobre a decisão do outro. O dilema do prisioneiro mostra que, em cada decisão, o prisioneiro pode satisfazer o seu próprio interesse, não confessar, ou atender ao interesse do grupo, confessar. A Tabela 1 demonstra as possibilidades de ganhos e perdas desse jogo.

	A Confessar	A Não confessar
B Confessar	3ano para A 3ano para B	4anos para A B fica livre
B Não confessar	A fica livre 4anos para B	2 anos para A 2 anos para B

Tabela 1: Matriz dos Payoffs do Dilema do Prisioneiro
Fonte: http://pt.wikipedia.org/wiki/Dilema_do_prisioneiro

Para qualquer um dos prisioneiros, o melhor resultado possível é confessar e seu parceiro ficar calado. E até mesmo se seu parceiro trair, o prisioneiro ainda lucra por também ambos falam só apanham 3 anos, já que ficando em silêncio pegará 4 anos de cadeia, caso o seu parceiro delata.

O único problema é que ambos chegarão a essa conclusão: a escolha racional é trair. Essa lógica vai, desta forma, proporcionar a ambos três anos de cadeia. Se os dois confessassem, haveria um ganho maior para todos, mas a optimização dos resultados não é o que acontece.

FORMA EXTENSIVA – A apresentação de jogos na forma extensiva é possível quando há alternância nos movimentos, todos os movimentos prévios são observados antes que o próximo movimento ocorra e os *payoffs* de cada combinação de movimentos viáveis são de conhecimento comum.

Na forma extensiva a representação é feita com árvores onde cada vértice (ou nodo) representa um ponto de decisão para um jogador. O jogador é especificado por um número listado no vértice. Os pagamentos são especificados na parte inferior da árvore.

Um exemplo do uso da representação da forma extensiva é o de um jogo entre uma empresa Inovadora e a empresa Líder de mercado, em que a Inovadora decide antes se vai ou não introduzir seu novo produto no mercado, e a partir daí a Líder toma sua decisão, já conhecendo a escolha da Inovadora.

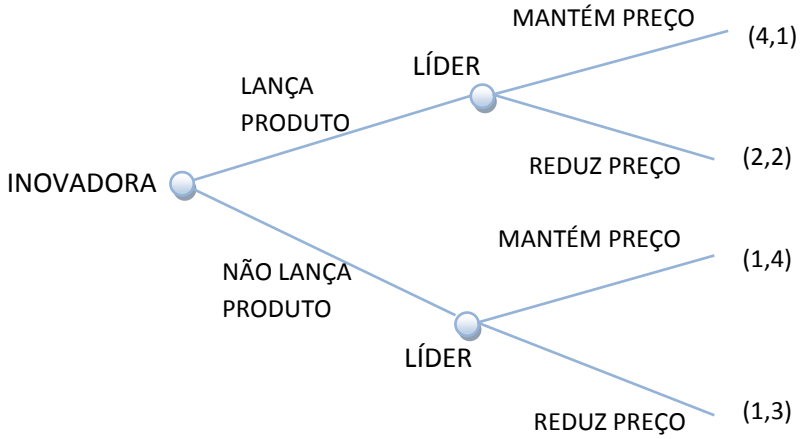


Figura 1: Árvore do jogo da empresa inovadora

3.4. CLASSIFICAÇÃO DE JOGOS

3.4.1. QUANTO Á SIMETRIA

SIMÉTRICOS- Um jogo simétrico é aquele no qual os pagamentos para os jogadores em uma estratégia particular dependem somente da estratégia escolhida, e não de quem está jogando. Se as identidades dos jogadores puderem ser trocadas sem alterar os pagamentos obtidos pela aplicação das suas estratégias, então este é um jogo simétrico. Muitos dos jogos 2×2 comummente estudados são simétricos. As representações padrões do Jogo da Galinha, do Dilema do prisioneiro, e da caça ao veado são todos jogos simétricos. Certos académicos estudam variações

assimétricas destes jogos, contudo, a maioria dos pagamentos destes jogos são simétricos.

ASSIMÉTRICOS - Os jogos assimétricos mais comuns são jogos onde existem grupos de estratégias diferentes para cada jogador. Por exemplo, o jogo do ultimato e seu similar, o jogo do ditador tem estratégias diferentes para ambos os jogadores. É possível, contudo, para jogos que tenham estratégicas idênticas para ambos os jogadores, que ainda assim sejam assimétricos. Por exemplo, o jogo representado na figura à direita é assimétrico, a despeito de possuir estratégias idênticas para ambos os jogadores

3.4.2. QUANTO AO PAYOFF

SOMA ZERO - São aqueles em que a soma dos payoffs dos jogadores é zero, ou seja, um jogador só pode ganhar se o outro perder, assim como no póquer, xadrez, entre outros. É a este tipo de jogo que se aplica o teorema minimax.

SOMA NÃO-ZERO - São os que não possuem a propriedade acima, como o Dilema do Prisioneiro, em que o payoff total é 2 anos de prisão se ambos ficam em silêncio e 1 ano se os dois prisioneiros confessam.

3.4.3. QUANTO À INFORMAÇÃO

INFORMAÇÃO PERFEITA - São aqueles em que todas as jogadas são conhecidas por todos os participantes envolvidos. Assim, o oril é um jogo com informação perfeita, enquanto o póquer simplificado não pode ser classificado como tal, já que

um jogador B não tem conhecimento sobre a carta que A escolhe, a primeira jogada, portanto estes são classificados de **informação imperfeita**.

3.4.4. QUANTO A SIMULTANEIDADE

SIMULTÂNEOS - Jogos simultâneos são jogos onde ambos os jogadores movem-se simultaneamente, ou se eles não se movem simultaneamente, ao menos os jogadores desconhecem previamente as acções de seus adversários (tornando-os *efectivamente* simultâneos).

SEQUENCIAIS - Jogos sequenciais são jogos onde o próximo jogador tem conhecimento da jogada de seu antecessor. Isto não necessita ser informação perfeita a cerca de cada acção do jogador antecessor; ele necessita de muito pouca informação. Por exemplo, um jogador deve saber que o jogador anterior não pode realizar uma acção em particular, enquanto ele não sabe quais das outras acções disponíveis o primeiro jogador ira realmente realizar.

3.5. O ALGORITMO MINIMAX

Inicialmente a teoria dos jogos não foi dado a sua devida importância, mas as coisa mudaram quando o matemático VonNeumann a partir da teoria de jogos de soma zero - onde o ganho de um jogador é igual a perda do outro, jogos perfeitamente equilibrados, ele desenvolveu o teorema MiniMax - que diz o seguinte:

"Para qualquer jogo para dois jogadores que respeite a teoria soma zero, existe uma estratégia mista para cada jogador tal que o resultado esperado para os dois é o mesmo valor V quando os jogadores usam esta estratégia. V é o melhor valor que

cada um pode esperar de uma jogada. Isto é, estas estratégias mistas são as estratégias óptimas para os dois jogadores."

Nesse algoritmo, o jogador denominado MAX tenta obter o maior resultado possível no jogo, enquanto MIN tenta minimizá-lo. O algoritmo Minimax desenvolve a árvore de jogadas da partida, atribuindo um estado para cada um de seus nós, a partir do estado inicial. Em cada nó, o jogador da vez tem um conjunto de jogadas que pode realizar, e cada uma delas leva a um nó diferente, formando a árvore de jogadas até os estados finais. Nos estados finais, ou seja, nas folhas da árvore, a utilidade dos estados é calculada, e são atribuídos pontos a cada um dos jogadores.

A função de MAX é escolher a melhor jogada, dado um ponto da árvore correspondente ao estado actual, supondo que MIN executará sempre sua melhor jogada. Esse cálculo é realizado desde as folhas até o nó da árvore que representa o estado actual do jogo, sempre admitindo que MIN escolherá a jogada que o leva a nós com função utilidade menor.

Um exemplo pode ser dado no jogo trivial de 2 jogadas descrito pela Tabela 2: o jogador A (MAX) deve escolher uma linha, e somente depois disso o jogador B (MIN) deve escolher uma coluna. O resultado do jogo é o elemento da matriz que corresponde à linha e à coluna escolhidas pelos jogadores. Prevendo que a melhor escolha de MIN seja a coluna que contém o menor elemento da linha escolhida, MAX tentará maximizar a jogada de MIN, escolhendo a linha 3. Dessa forma, o melhor resultado para MIN será a escolha da coluna 3, resultando o valor -1 no jogo.

	B escolhe 1	B escolhe 2	B escolhe 3
A escolhe 1	+3	-2	-1
A escolhe 2	-4	+1	0
A escolhe 3	+2	0	-1

Tabela 2: Matriz dos Payoffs

Em forma de árvore, o jogo de 2 movimentos acima seria descrito como na Figura 2:

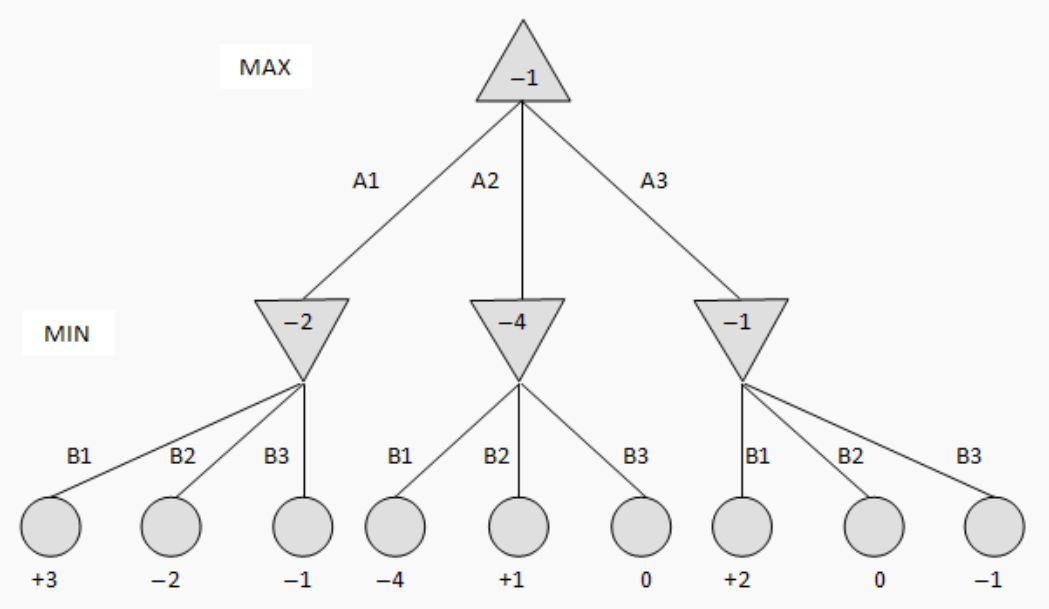


Figura 2: Árvore de Pesquisa do Min Max
Fonte: http://en.wikipedia.org/wiki/Minimax

Na árvore, pode-se perceber que o agente espera que MIN faça sempre a melhor jogada, indo para o estado de menor valor. Prevendo isso, MAX pode basear sua escolha aplicando valores -2, -4 e -1, respectivamente, às jogadas A1, A2 e A3. Escolhe, portanto, a de maior valor: A3.

No entanto, na maioria dos jogos (incluindo o Oril), não é possível descrever uma árvore de possibilidades em sua totalidade, dado que a característica exponencial do desenvolvimento dos nós produziria um número muito grande de estados, sendo admitido infinito na maioria dos jogos. Nesses casos, a função utilidade dos nós não pode ser calculada, e portanto deve ser estimada segundo uma heurística de avaliação.

3.6. OPTIMIZAÇÃO DA ÁRVORE DE BUSCA

Um método de optimizar a busca do algoritmo Minimax é a Poda α-β. Nela, o algoritmo elimina galhos da árvore que comprovadamente não podem trazer resultados melhores para o jogador, interrompendo a busca naquele trecho da árvore. Desta forma, o tempo de processamento é reduzido, mas o resultado da busca é o mesmo nas duas técnicas. Um exemplo pode ser dado na Figura 3.

Na árvore da Figura 3, o jogador MAX faz a avaliação dos três primeiros estados, determinando a A1 o valor -2. Ao analisar A2-B1, encontra o valor -4. Assim, assumindo que MIN fará sempre a melhor jogada, pode esperar que MIN escolha -4 (ou um número menor, dependendo da avaliação das outras 2 opções) se sua escolha for A2. Portanto, pode “podar” a busca nesse galho da árvore, pois não encontrará valores mais favoráveis do que encontrou em A1. Passando para o nó A3, faz normalmente a avaliação, encontrando o valor -1, não havendo poda. Há, portanto, uma economia no tempo de processamento da árvore, especialmente para buscas de profundidade maior.

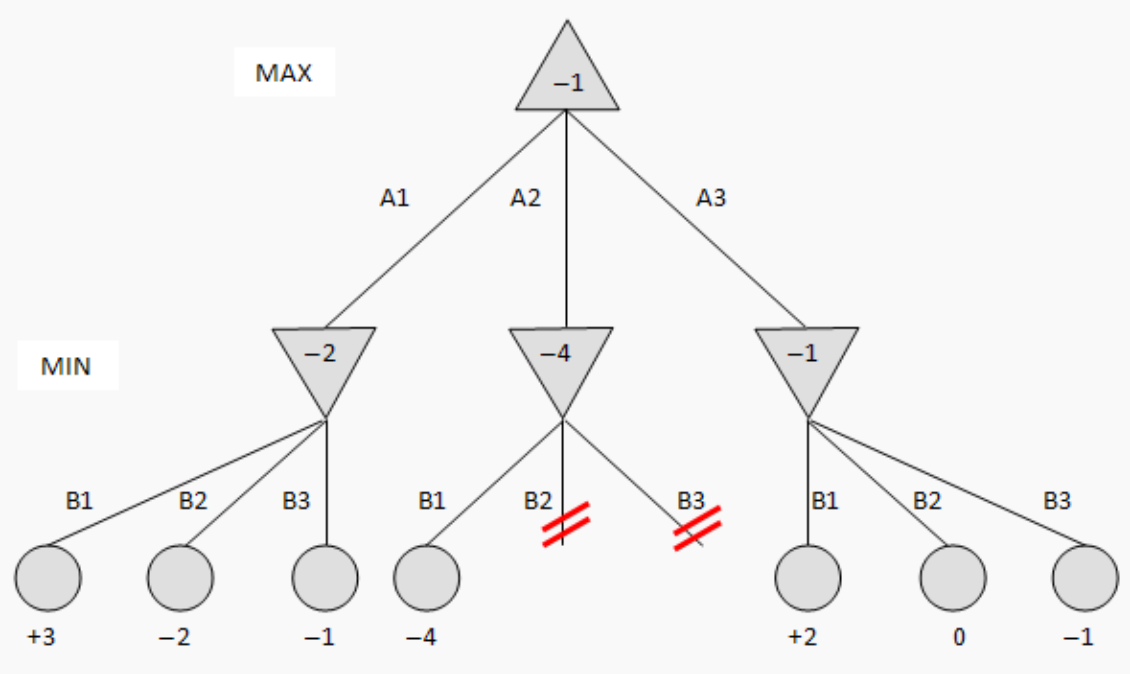


Figura 3: Árvore de busca Min Max com poda alfa-beta
http://en.wikipedia.org/wiki/Minimax

O algoritmo minimax possui o seguinte pseudocódigo:

```
FUNÇÃO MiniMax(nó, profundidade)
   SE nó é um nó terminal OU profundidade = 0 ENTÃO
      RETORNE o valor da heurística do nó
   SENÃO SE o nó representa a jogada de algum adversário ENTÃO
      α ← +∞
      PARA CADA filho DE nó
         α ← min(α, MiniMax (filho, profundidade-1))
      FIM PARA
      RETORNE α
   SENÃO
      α ← -∞
      PARA CADA filho DE nó
         α ← max(α, MiniMax (filho, profundidade-1))
      FIM PARA
      RETORNE α
   FIM SE
FIM FUNÇÃO
```

4. O JOGO DE ORIL

O Oril, também conhecido como A-i-u, Awalé, Wari, Adi, entre outros nomes, cada um da sua região, é um jogo de estratégia, jogado ao redor do mundo, principalmente em África,cuja sua origem exacta é desconhecida[1].O jogo consiste num tabuleiro com duas fileiras de seis buracos (chamados de casas) e quarenta e oito pedras.

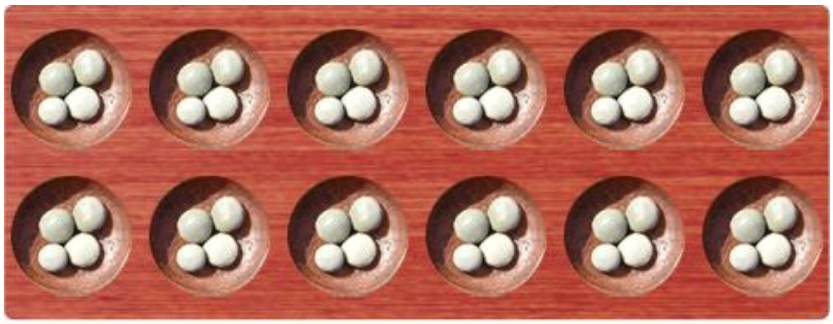


Figura 4: Tabuleiro de Oril

4.1. OBJECTIVO

Capturar o máximo número de pedras. E vence quem capturar pelo menos vinte e cinco pedras e no caso de ambos os jogadores capturarem vinte e quadro pedras cada um o resultado é um empate.

[1] Graça, Albertino. Jogo de Oril Regras, Estratégias e Teorias(1ª Edição). Mindelo, Edição ONDS, Novembro de 1998, Pag. 9

4.2. REGRAS

Para jogar o Oril, existem uma série de regras que devem ser obedecidas pelos jogadores.

4.2.1. JOGADORES E INICIO DO JOGO

Todos os buracos contêm quatro pedras no início do jogo. Cada um dos dois jogadores joga numa única fileira, a do seu lado do tabuleiro, um de cada vez, alternadamente, onde pode ser iniciado por qualquer um dos jogadores.

4.2.2. EXECUÇÃO DE JOGADAS

Para execução de jogadas o jogador pega todas as pedras de um dos buracos do seu lado e distribui-las no sentido anti-horário, sucessivamente, uma por cada buraco.

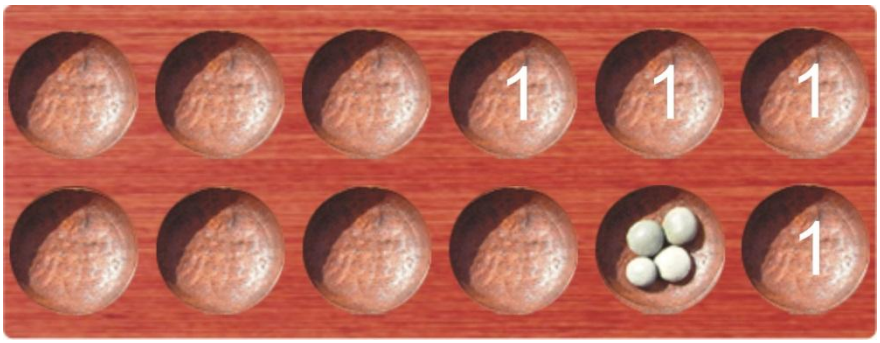


Figura 5: Execução de lance

Se o buraco contém mais de doze pedras, o buraco inicial será saltado durante a distribuição. No exemplo a seguir, o buraco inicialmente contém 14 pedras (à

esquerda) e a décima segunda pedra é colocada logo após o buraco inicial. A posição final é mostrada à direita. Três buracos receberam duas sementes.

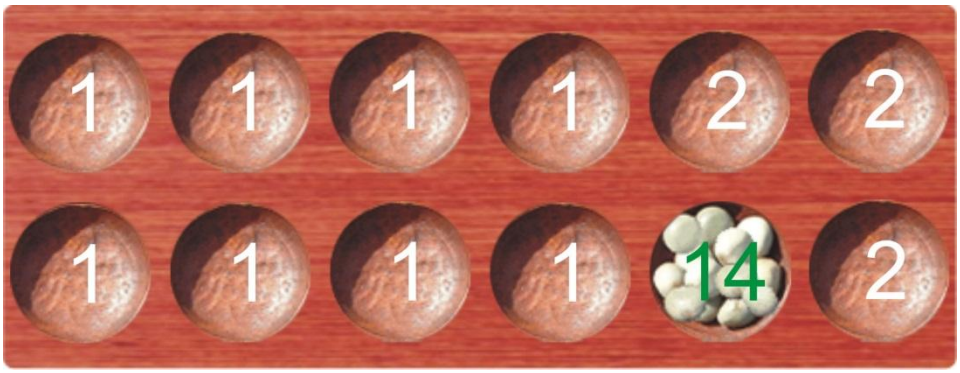


Figura 6: Execução de Lance

4.2.3. CAPTURA DE PEDRAS

Se após a distribuição de pedras o buraco final for do lado do campo adversário e este tiver duas ou três pedras, essas pedras serão capturadas pelo jogador.

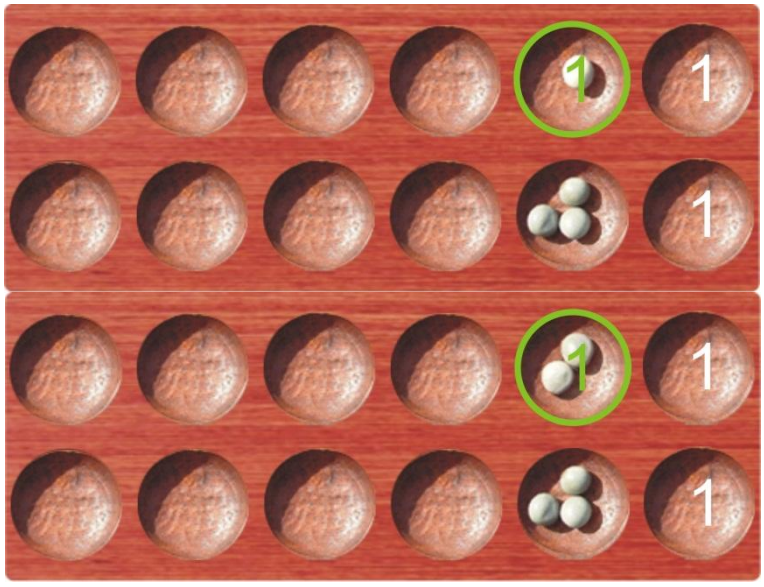


Figura 7: Captura de pedras

Também pode-se capturar vários buracos sucessivos, desde que todos eles têm duas ou três pedras.

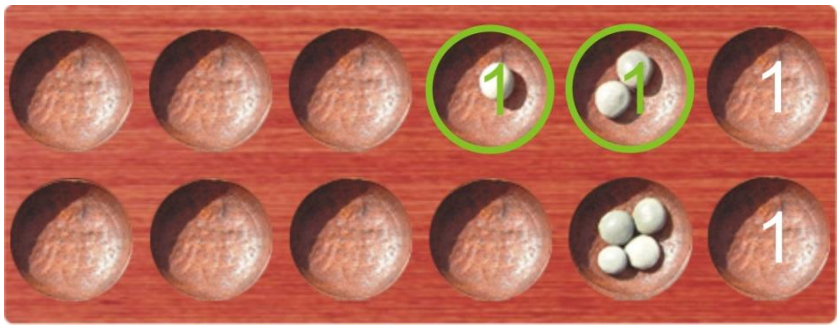


Figura 8: Captura de pedras em buracos sucessivos

Os buracos capturados devem ser contíguos, no exemplo a seguir, os três últimos buracos são capturados, mas não mais, devido ao quarto buraco que contém quatro pedras.

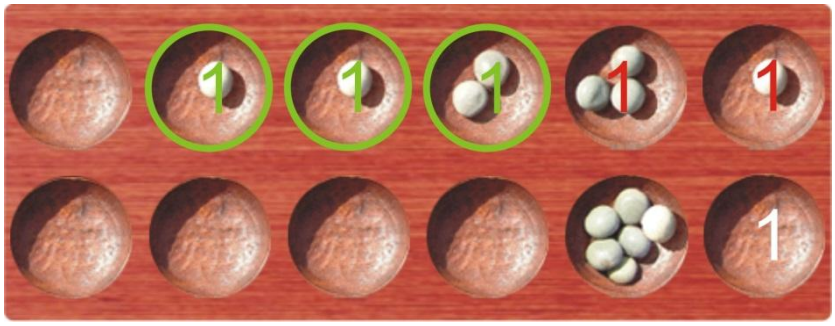


Figura 9: Captura de pedras em buracos sucessivos

O número máximo de casas que podem ser capturadas é cinco, e neste caso o jogador terá que executar uma outra jogada. Caso o jogador não ter a possibilidade de executar uma outra jogada ele executa a jogada e apanha todas as pedras do tabuleiro e o jogo termina.

Figura 10: Captura de pedras em buracos sucessivos

4.2.4. PASSAGEM DE PEDRAS

Se o adversário não tem pedras, é obrigatório jogar de forma a passa-lo pedras para o seu campo. Na situação seguinte, apenas pode-se jogar com o buraco com quatro pedras.

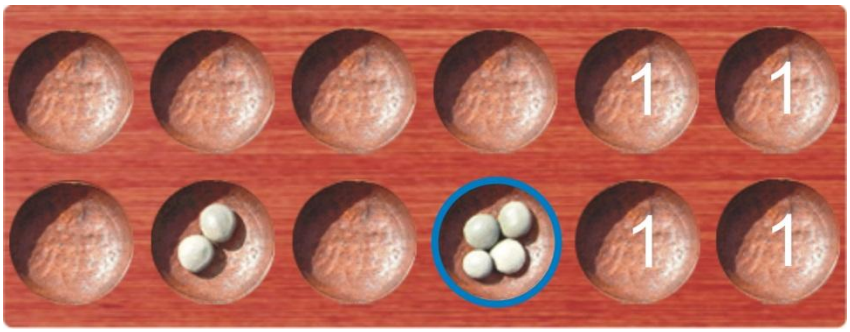


Figura 11: Passagem de pedras

4.2.5. FIM DE JOGO

O jogo termina quando um jogador não pode jogar, por falta de pedras no se campo. O adversário acrescenta suas pedras na pontuação final.

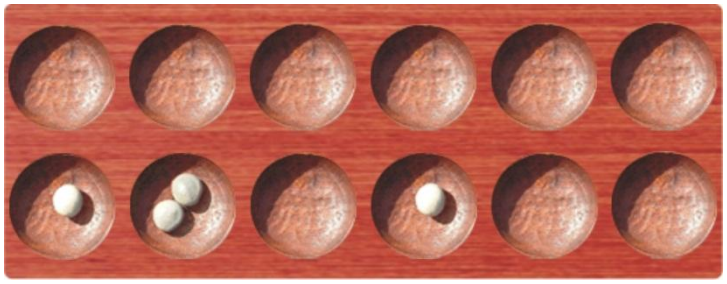


Figura 12: Fim do jogo

O jogo também termina quando cair num estado cíclico e que não houver oportunidade para qualquer dos jogadores fazerem mais capturas. Na seguinte posição, o jogo é cíclico se ambos os jogadores jogam de forma optimizada. Cada jogador apanha suas pedras para a pontuação final.

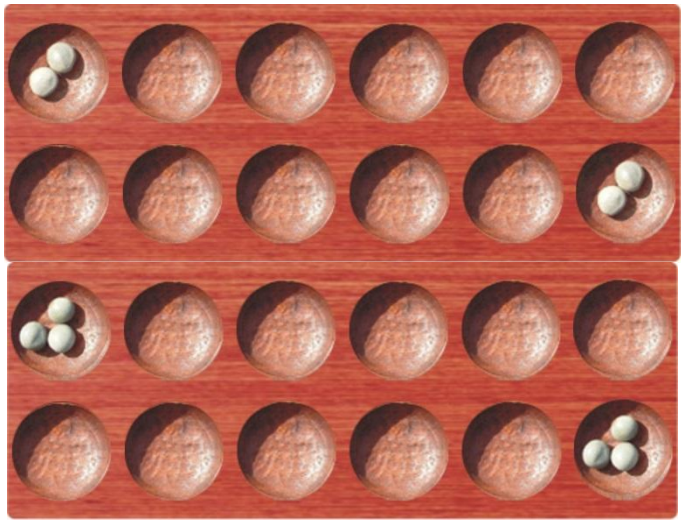


Figura 13: Estado cíclico do jogo

No final do jogo será feito a contagem das pedras e vence o jogador que conseguiu capturar o maior número de pedras.

4.3. FASES E ALGUMAS ESTRATÉGIAS DO JOGO

Durante o jogo podemos estabelecer três fases diferentes. A primeira é o inicio do jogo, que podemos chamar de abertura. A segunda é quando ainda não tem nenhuma pedra captura, ou algumas já estão capturadas, mas ainda a maior parte delas se encontram no tabuleiro. A ultima e quando já resta poucas pedras no tabuleiro e que a captura de parte delas por um jogador é decisivo para a sua vitória.

4.3.1. ABERTURA DO JOGO

Geralmente, não é uma boa ideia jogar um conjunto de casas consecutivas durante a primeira fase do jogo, pois coloca o jogador vulnerável, dando a possibilidade a várias capturas, muito difícil de evitar.

Por esta razão, as sequências 6-2-4, 2-4-6 ou 5-3-2 são melhores escolhas em vez de sequências de 1-2-3 ou 2-1-3.

4.3.2. MEIO DO JOGO

O objectivo do jogo é de capturar mais pedras do que o adversário. Para fazer isso, o jogador deve tentar preparar seus ataques, agrupando o número de pedras em seus buracos com os valores que atingem o território oposto e permitir uma captura.

O sistema de capturas múltiplas pode aumentar drasticamente a pontuação e, portanto, é para este tipo de captura que o sistema tem que ser orientada.

O adversário também deve ser impedido de fazer capturas múltiplas, por vezes sacrificando algumas das pedras do jogador.

Antes de um jogador decidir capturar algumas pedras do adversário, ele deve verificar se o buraco jogado não revela uma posição importante para o adversário em seu próprio território.

Ele também devera criar condições fazendo com que o adversário fica com mais casas vulneráreis ou diminuído as suas.

No jogo Oril, o jogador deve semear antes de colher. Uma série de buracos vazios no campo do adversário, judiciosamente criadas, pode-se levar a captaras múltiplas.

Uma estratégia ofensiva clássica é a criação de uma “casa”. Uma casa é a acumulação em um buraco de pedras suficiente para fazer uma revolução completa, ou seja, pelo menos doze pedras. Considera-se que uma casa é feita quando tiver pedras o suficiente para que na segunda volta consiga atingir pelo menos a primeira casa do campo do adversário.

A utilização da casa pode ser devastador quando o território do adversário está quase vazio, porque, durante o lance, a primeira volta preenche os buracos e a segunda termina com uma captura múltipla.

Um jogador pode usar uma série de estratégias defensivas para defender de uma casa. Segue-se algumas delas:

- Fazendo o fecho total da casa, que consiste em fazer com que a distribuição das pedras das casa termina num buraco que inicialmente tinha no mínimo duas pedras.

- Sobrecarregando a casa do adversário, que consiste na adição de pedras à casa do oponente, e fazê-lo perder o seu alvo.

- Não passando pedras para o adversário fazendo com que ele fica sem alternativas e ter que jogar com a casa antes que ela esteja completa.

- Ou contra-atacando, que consiste em agrupar pedras em um ou mais buracos de lhe permitira capturar várias casas logo após o adversário utilizar a casa.

4.3.3. FIM DO JOGO

No final do jogo quando não haver muitas pedras no tabuleiro, o jogador terá que construir “armadilhas”, onde o objectivo é deixar o adversário com apenas um buraco jogável e faze-lo jogar num buraco ameaçado.

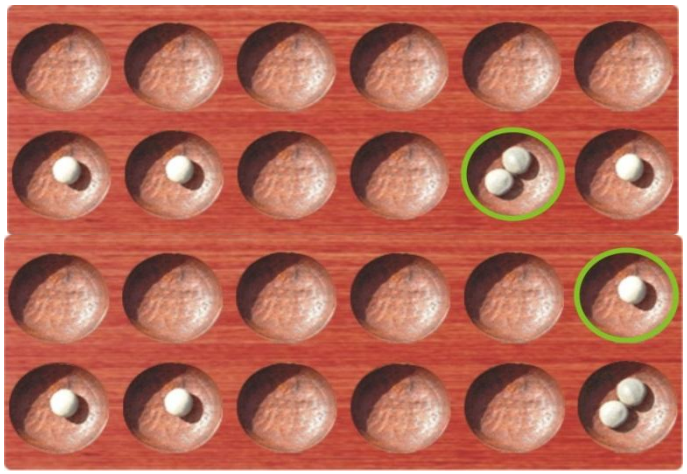


Figura 14: Estratégias de fim de jogo

Uma outra estratégia eficiente também e a do jogador mover lentamente as pedras em direcção do campo adversário fazendo com fica sem a possibilidade de passa-lo pedras para jogar, tendo em conta que um jogador não pode executar dois lances consecutivos.

Figura 15: Movendo lentamente

5. A ÁRVORE DE BUSCAS DO ORIL

Construir uma árvore de busca para todos os movimentos possíveis do jogo de oril é uma tarefa complicada. Em qualquer movimento de cada jogador tem um máximo de seis opções. Conforme o jogo avança, este número diminui, como buracos vazios não podem ser seleccionados para lances válidos. Isto dá-nos uma árvore com uma largura de 6 ramos em cada movimento (máximo). A duração de um jogo de oril pode variar enormemente de trinta a oitenta ou mais movimentos. Tomando por exemplo cinquenta como um valor médio, poderíamos ter uma profundidade de cinquenta níveis para a nossa árvore de busca.

Portanto, o número total de posições e, que cada posição de jogo que poderiam ser atingir pode ser facilmente calculado.

Número de posições: N = 6 * 6 * 6 * 6 ... ..* 6 (50 vezes) = 6^{50}

Este não é, naturalmente um resultadopreciso, mas isso mostra que o número de possíveis posições de jogo é muito alto.

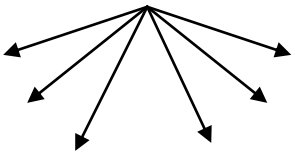
A figura abaixo mostra como uma árvore seria construída. A partir da primeira posição, a posição de início do jogo, a selecção em cada ramo um movimento diferente para cada jogador. Em alguns níveis, o número de ramos seria inferior a 6.

F E D C B A N

4 4 4 4 4 4 0

4 4 4 4 4 4 0

a b c d e f S

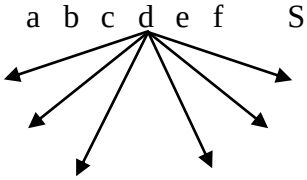


F E D C B A N

4 4 4 4 4 4 0

0 5 5 5 5 4 0

a b c d e f S



4b 4c ... 4d 4e

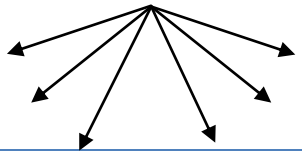
.....

F E D C B A N

4 4 5 5 5 5 0

4 4 4 4 4 0 0

a b c d e f S



É claro que para uma busca óptima como vimos anteriormente com a introdução de cortes alfa-beta no algoritmo de busca, eliminando os passos considerados desprezíveis o número de lances a analisarpoderá diminuir para a metade e neste caso seriam $6^{50/2}$ de passos.

6. IMPLEMENTAÇÃO DO JOGO ORIL

Após o estudo do teórico e do desenho do algoritmo da inteligência artificial do jogo fizemos também a implementação do jogo para funcionar num iPhone onde foram desenvolvidas as seguintes funcionalidades:

- O utilizador poderá jogar contra o computador, onde tem á sua escolha três níveis diferentes de dificuldade.

- Ou se pretender, poderá jogar online com um outro jogador via internet.

- Também o jogador tem a possibilidade de fazer o registo de pontuação online, activar ou desactivar os sons.

O diagrama ilustrado na Figura 16 mostra como o programa é dividido em dois módulos principais. O módulo que implementa a interface gráfica permite ao utilizador controlar, por meio de menus, um conjunto de opções de jogo e efectuar as suas jogadas por meio tabuleiro apresentado no ecrã. E o módulo do jogador artificial.

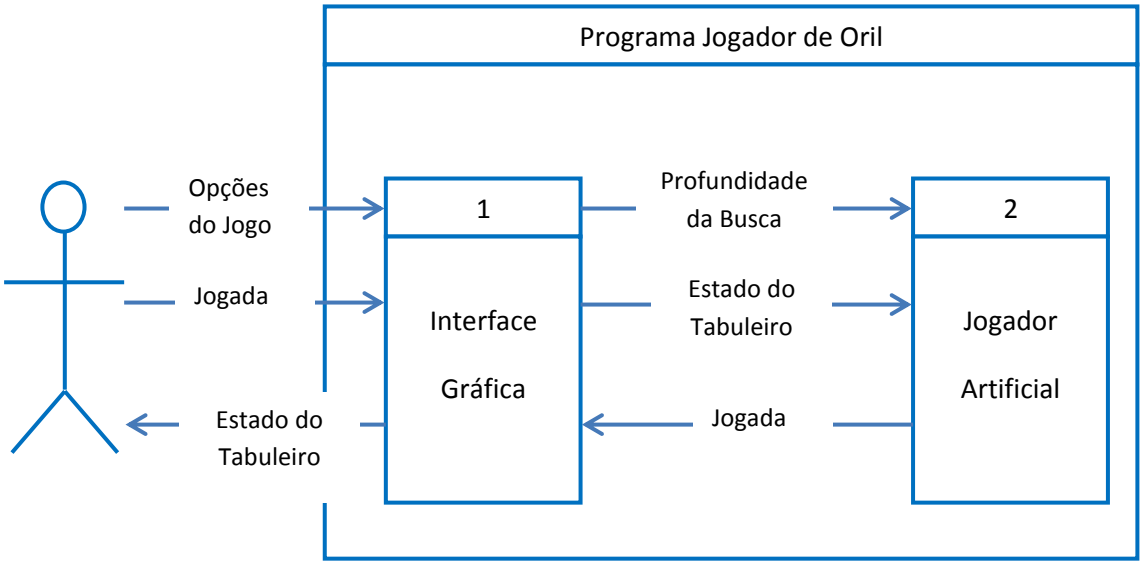


Figura 16: Programa jogador de oril

Após cada jogada efectuada pelo utilizador, a Interface gráfica traduz o nível de dificuldade escolhido num nível de profundidade para a pesquisa de jogadas e constrói uma descrição do estado corrente do tabuleiro de jogo. Estas informações são fornecidas ao módulo Jogador artificial, que devolve à Interface gráfica uma boa jogada por parte do computador. A Interface gráfica actualiza então a representação do tabuleiro no ecrã, por forma a reflectir a jogada do computador.

O diagrama apresentado na Figura 17 ilustra, num nível de abstracção inferior, a constituição do módulo Jogador artificial. O sub-módulo de Algoritmo minimax recebe as informações fornecidas pela Interface gráfica, utilizando-as como ponto de partida para a construção e pesquisa de uma árvore de jogadas com profundidade determinada (ou seja, uma árvore de pesquisa). Este sub-módulo implementa o algoritmo Minimax com cortes alfa-beta. Este é um algoritmo de pesquisa *genérico* para jogos com dois jogadores e informação completa, por isso tem que recorrer a

três outros sub-módulos para obter informações específicas do jogo de Oril implementado.

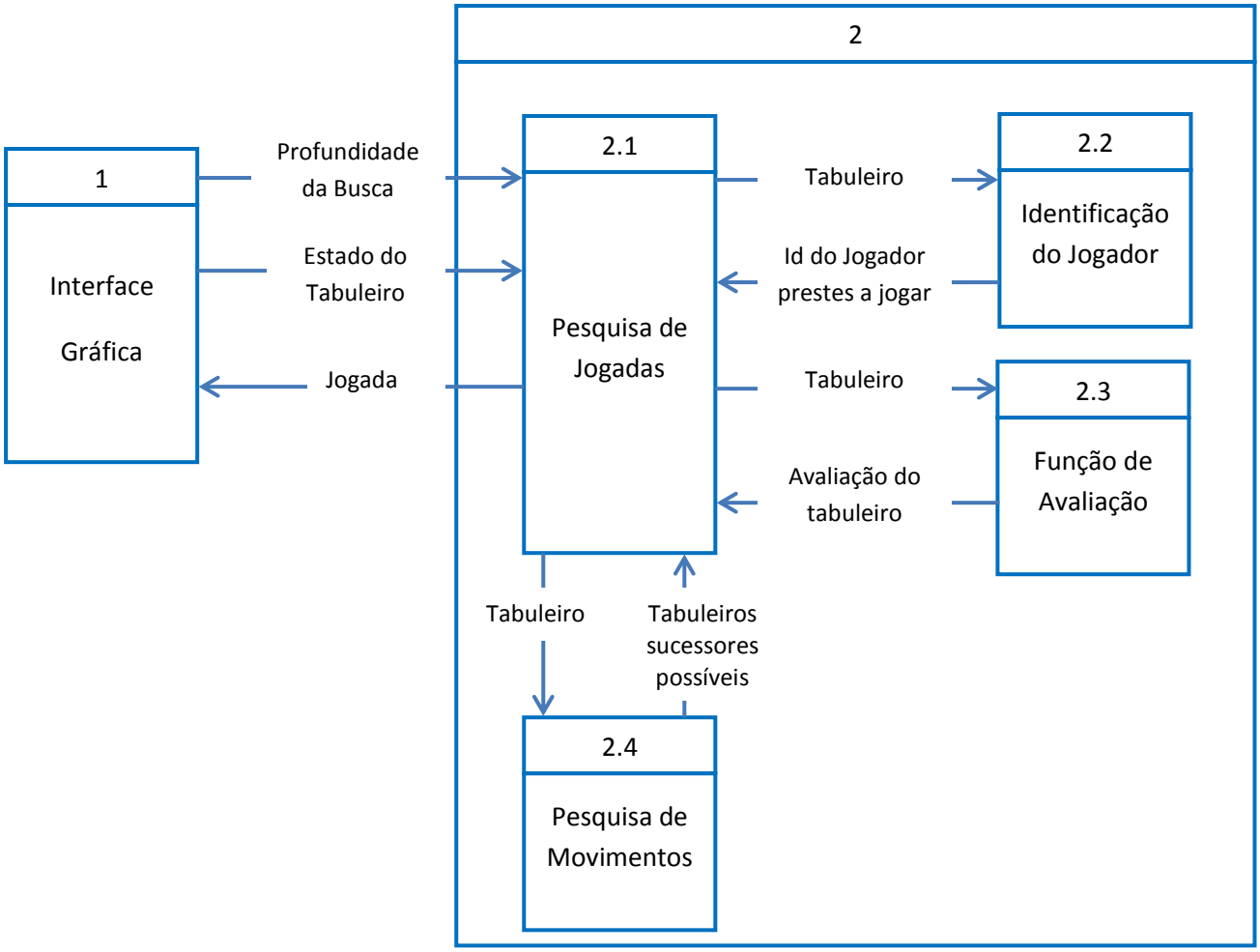


Figura 17: Jogador artificial

6.1. O JOGADOR ARTIFICIAL

Feito a o estudo teórico do jogo de oril e do algoritmo minimax que ajudará na implementação da busca da melhor jogada por um jogador, vamos fazer a sua devida implementação utilizando a linguagem c.

Para fazer a implementação da inteligência artificial do jogo de oril fazemos o uso do algoritmo minimax, que levou ao seguinte trecho de código:

```
/*     Procura a melhor jogada para uma dada PROFUNDIDADE     */

Intmelhor_jogada (inttabuleiro[], intpontuacao[])
{

int jogada;
intsupv = -100, supr = -100;
intval[12];
intdif[12];
int j, h;
intcomputer=1;

              /*
                     avalia numa determinada profundidade o valor de
                     cada buraco, guarda este na arrayval[]
                     e o maior valor será guardado na variável supv

              */
for (j = computer * 6; j <= computer * 6 + 5; j++)
              if (valido (tabuleiro, j))
                {

                     val[j] = valor (tabuleiro, pontuacao, j, 0);
                     supv = max (val[j], supv);

                }


              /*
                     baseando nos buracos com o maior valor calcula qual
                     destes poderá traduzir e maior ganho para o computador
                     e retorna a o buraco com maior diferença

              */
for (j = computer * 6; j <= computer * 6 + 5; j++)
              if (valido (tabuleiro, j) &&val[j] == supv)
                {

                     dif[j] = diferenca (tabuleiro, pontuacao, j);
                     if (dif[j] >supr)
                       {
                            supr = dif[j];
                            jogada = j;
                       }
                }


return jogada;

}
```

```
/*      Calcula e retorna a maior diferença depois da distribuição de pedras*/

Intdiferenca (inttabuleiro2[], int pontuacao2[], int i)
{

Inttabuleiro[12];
Intpontuacao[2];

Intdif, player, i0, tmp, j;

for (j = 0; j <= 11; j++)
tabuleiro[j] = tabuleiro2[j];
for (j = 0; j <= 1; j++)
pontuacao[j] = pontuacao2[j];

jogador = quem_joga (i);
i0 = jogador * 6;


if (i + tabuleiro[i] > i0 + 5)
       /*
       caso o nº de pedras depois de
       distribuir atingir o campo adversário
          */
dif = 0;
else
        {
       // verifica a diferença para as próximas jogadas
distribuir_pedras (tabuleiro, pontuacao, i);
dif = 0;
for (j = i0; j <= i0 + 5; j++)
              if (tabuleiro[j] != 0)
              dif = max (dif, diferenca (tabuleiro, pontuacao, j));

dif++;
    }

Returndif;

}
```

A função Melhor_Jogada() usa as funções valor() e diferenca(), onde a função valor() faz a avaliação de cada buraco numa determina profundidade e a função diferenca()baseando nesses valores, procura o buraco que levará a jogadas onde será passado o mínimo número de pedras para o adversário. Isso tendo em conta que quando menos pedras o adversário tem mais vulnerável ele ficará.

6.2. ESTRUTURA DE DADOS

Para modelar o problema da implementação do jogo levou a seguinte estrutura de dados ilustrado no diagrama da Figura 18, onde foram definidas as entidades assim como os seus atributos e métodos.

Onde a entidade jogo inicia a partida com dois jogadores e um tabuleiro. Cada jogador é inicializado com um nome e um número de pontos e o tabuleiro é composto por vários buracos e cada um com um determinado número de pedras. Para cada jogador executar e lances vai necessitar de uma mão. E cada mão tem as suas coordenadas x e y.

Também com o andar do jogo os atributos de cada entidade pode ser consultados e alterador e para isso, foi criado os métodos como por exemplo Movimentar da entidade Mão, AdicionarPedra da entidade Buraco e MostrarPontos da entidade Jogador, entre outros.

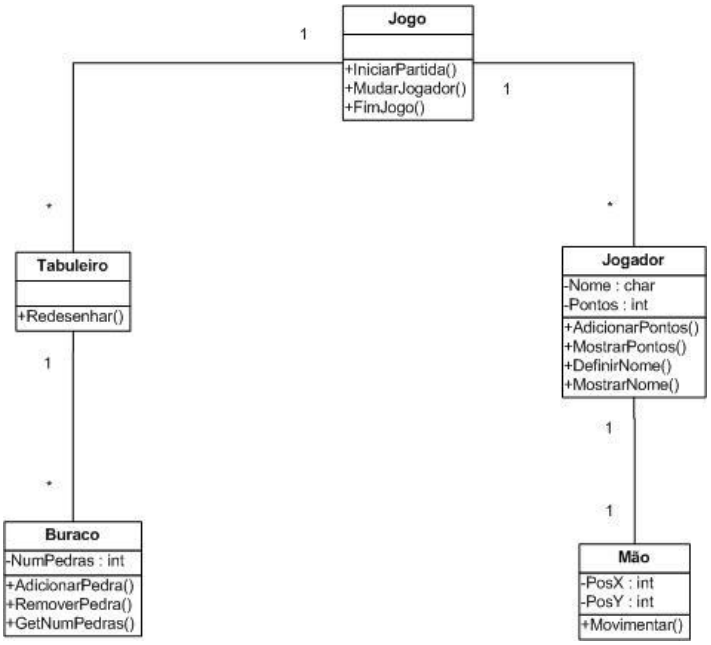


Figura 18: Diagrama de Classes

6.3. FERRAMENTAS DE DESENVOLVIMENTO

O desenvolvimento de aplicativos para iPhone requer o uso de umSDK (Software Development Kit). A Apple disponibiliza esse SDK gratuitamente mas só funciona no ambiente MACOSX. No nosso caso como não tínhamos um computador Macintosh disponível para execução do projecto optamos pelo uso do DragonfireSDK que é um SDK,shareware para o ambiente do Windows.

Apesar de algumas limitações em relação ao SDK da Apple, conseguimos implementar todas a funcionalidades pretendidas.

6.3.1. DRAGONFIRESDK

Concebido principalmente para desenvolvimento de pequenos jogos 2D, ele é um SDKshareware da Zimusoft, Inc. e foi feito para ser usado como livraria para o Microsoft Visual C++, portanto as aplicações tem que ser escritas em C/C++.

ODragonfireSDK requer essencialmente o uso de três funções:

AppMain()

OnTimer()

AppExit()

Assim como qualquer programa em C/C++ requer uma função main () como ponto de partida, aplicações com DragonFireSDK precisa de uma função chamada AppMain. Esta função é a primeira a correr - que é onde podemos inicializar variáveis e carregar os ficheiros externos como imagens e sons.

A função OnTimer é executada trinta vezes por segundo. É ai que temos a possibilidade de fazer o GameLoop, ou seja criar animações alterando sequências de imagens e assim como executar as rotinhas repetitivas.

E a função AppExit é chamado sempre que a aplicação e terminada, pressionando o botão home do iPhone ou houver alguma outra interrupção como uma chamada recebida.

Ainda o SDK inclui um simulador do iPhone, onde podemos testar as aplicações no computador antes de descarrega-lo para o telemóvel.

6.4. A INTERFACE GRÁFICA

No desenvolvimento de jogos a interface gráfica tem um papel tão importante como a inteligência artificial quer em termos de imagens assim como as animações que devem ser sempre acompanhados de sons para oferecer ao utilizador uma maior aproximação da realidade.

O primeiro passo na construção da interface foi de montar todas peças do jogo, desde os menus, tabuleiro, pedras num programa de tratamento e manipulação de imagens.

6.4.1. O MENU

O menu do jogo fornece ao utilizador a possibilidade de escolher que tipo de partida que ele pretende disputar, contra o computador ou online com outro jogador e ainda definir as suas opções de jogo ou consultar as pontuações dos jogadores online.

O menu encontra-se estruturado como mostra a figura a baixo.

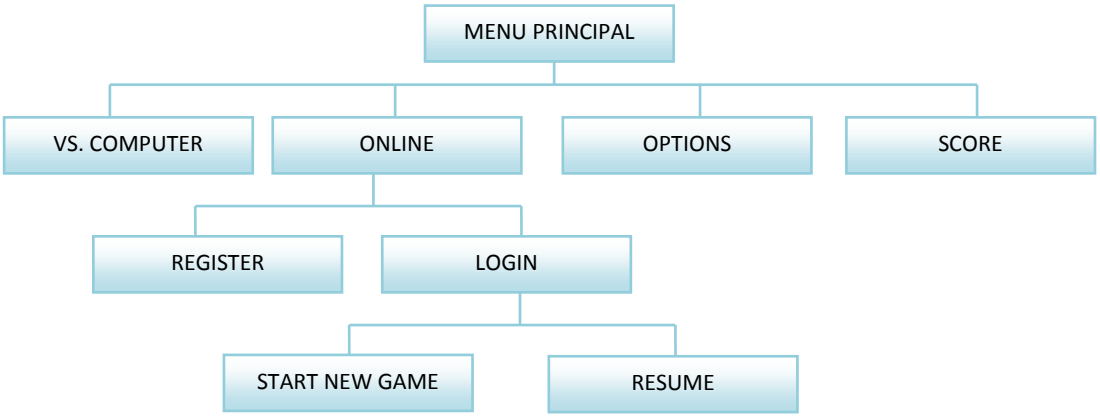


Figura 19: Estrutura de menus

A implementação dos botões do menu foi feita com recurso à função ButtonAdd do SDK, onde ela tem a seguinte sintaxe:

*intButtonAdd(char *filename, int x, int y, int (*callback)(int id), int id);*

Ela adiciona um botão que for tocado chama a função callback.

Parâmetros:

Filename - corresponde a parte comum do nome de dois ficheiros de imagem, <ficheiro>1.png e <ficheiro>2.png onde os índices 1 e 2 correspondem aos estados repouso e pressionado respectivamente.

X e Y – representam as coordenadas x e y onde o botão será posicionado no ecrã.

Callback – é o nome da função que será chamada aquando o botão for tocado.

Id – é um identificador do botão que será enviado para função callback. No nosso caso como temos vários botões e implementamos somente uma função callback, é com esse id que identificamos qual foi o botão tocado.

E o fundo do menu foi com uma chamada da função **ViewAdd**:

intViewAdd(intim, int x, int y);

Cria uma vista e adiciona uma imagem.

Parâmetros:

Im – identificador de uma imagem retornada pela função ImageAdd().

X e Y – representam as coordenadas onde a ela será mostrada e essas coordenadas corresponde ao canto superior esquerdo.

O trecho de código a seguir mostra a implementação menu principal utilizando as imagens da Figura 20:

Figura 20: Ficheiros utilizados na construção do menu.

```
Voidmostra_menu()
{
	bg_menu=ViewAdd("menu_bg.jpg",0,0);
	bt_vs_comp=ButtonAdd("vs_computer",15,95,OnMenuSelect,0);
	bt_online=ButtonAdd("online",137,95,OnMenuSelect,1);
	bt_options=ButtonAdd("options",220,155,OnMenuSelect,2);
	bt_scores=ButtonAdd("score",340,155,OnMenuSelect,3);

}
IntOnMenuSelect(intid)

{
switch(id)
  {
case 0:
	…
	break;
case 1:
	…
	break;

case 2:
	…
	break;

case 3:
	…
	break;

  }
Return0;

}
```

E o resultado da execução deste código está apresentado na Figura 21.

Figura 20: Menu principal

6.4.1.1. MENU VS. COMPUTER

Com o menu vs. Computer o utilizador inicia o jogo para defrontar uma partida com o computador com as configurações previamente seleccionadas no menu options, onde os lances executados pelo computador são feitos com base no algoritmo de inteligência artificial.

No ecrã do jogo o utilizador poderá visualizar o tabuleiro, assim como as pontuações e o indicador de turno de jogada.

Para o jogador executar lances basta fazer um toque sobe o buraco pretendido. Os lances executados ,quer pelo computador ou o utilizador são feitos com o uso de imagem de uma mão animada para dar mais realismo ao jogo.

6.4.1.2. MENU ONLINE

O utilizador ao tocar no menu online, ele terá duas opções:

1ª – Fazer o login, escolher um outro utilizador que está online e convida-lo para uma partida ou ainda continuar uma partida por concluir que tinha sido iniciado anteriormente, porque as vezes por um motivo ou outro, um dos jogadores pode ficar off-line e a partida fica registada no servidor para poder ser continuada no futuro.

2ª – Registar no servidor para poder jogar online, onde ele terá que informar um nickname e uma password.

6.4.1.3. MENU OPTIONS

O menu options é onde o utilizador poderá definir a suas configurações do jogo como a escolha de um dos três níveis de jogo contra o computador, activar/desactivar os sons e o lado do tabuleiro que inicia o jogo.

6.4.1.4. MENU SCORE

No final de cada partida será exibido a pontuação de cada jogador e informação de quem foi o vencedor ou de empate. E no caso dos jogos online a pontuação dos vencedores são registados numa base de dados para consultas futuras, e para tal basta estar online e entrar no menu score.

6.5. O JOGO ONLINE

O módulo do jogo online subdivide em três partes: o cadastro, o login e convite para início de jogo e o da execução do jogo.

Quando se faz o cadastro o utilizador digita um nicknamee uma password a sua escolha e quando e pressiona o botão OK para efectuar o registo primeiro será a avaliado localmente se as informações introduzidas obedecem os critérios. Se esses critérios não forem obedecidos será exibida uma mensagem de erro e o utilizador voltará a introduzi-las de novo, caso contrário essas informações serão enviadas para o servidor via internet, onde antes da introdução destes na base de dados será feita uma nova validação, que consiste em verificar a unicidade do nickname.

Figura 21: Ecrã de login

O diagrama da Figura 23 ilustra o processo de login.

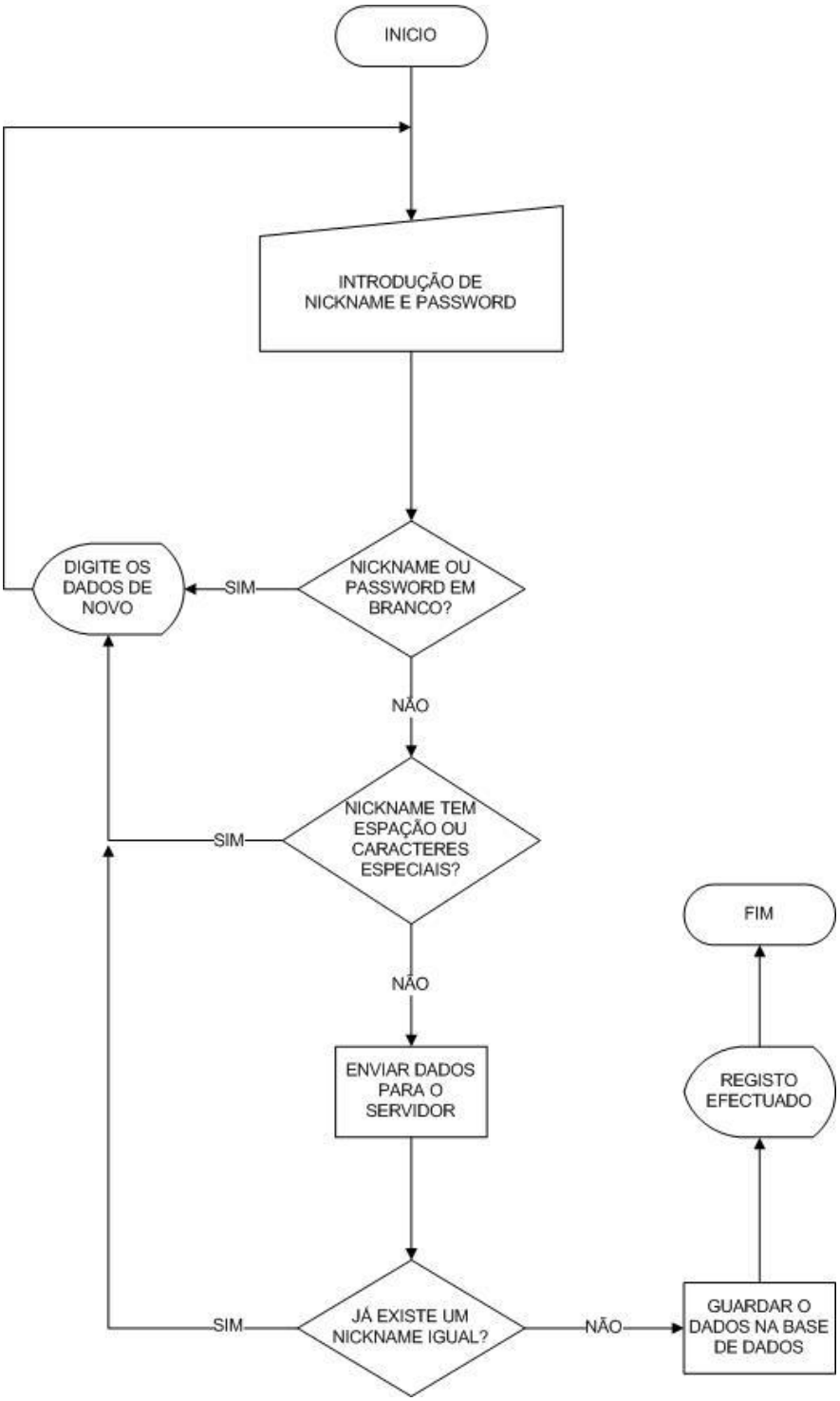


Figura 22: Diagrama do processo de login

Quando o utilizador pretender iniciar ou continuar uma partida online terá em primeiro lugar que efectuar o login com o seu nickname e password. O processo de login decorre de seguinte modo: após a introdução do nickname e da password será feita uma primeira validação dos dados localmente onde será averiguado somente se os campos foram preenchidos de forma correcta, de seguida estes são enviados para o servidor onde será verificada a sua autenticidade.

Com a autenticidade verificada a aplicação do servidor verifica se o utilizador tem partidas ainda com o estado em andamento, caso houver estes serão exibidos assim como a lista de todos os utilizadores online. Onde ele poderá continuar uma das partidas por terminar, fazer um dos jogadores da listaum convite ou ainda responder a um convite para iniciar uma partida.

Figura 23: Lista de jogadores online

Após o início da partida o utilizador que efectuou o convite será o primeiro a jogar e assim que ele efectuar o seu lance será enviado para o servidor as informações do estado do tabuleiro e enquanto isto acontece a aplicação do outro utilizador ficará periodicamente consultado o servidor para saber se é a sua vez de jogar e quando ele receber essa informação antes de colocar o tabuleiro no estado de aberto para o utilizador executar o seu lance, ele executa o lance do outro utilizador e o primeiro agora fica em estado de escuta assim sucessivamente.

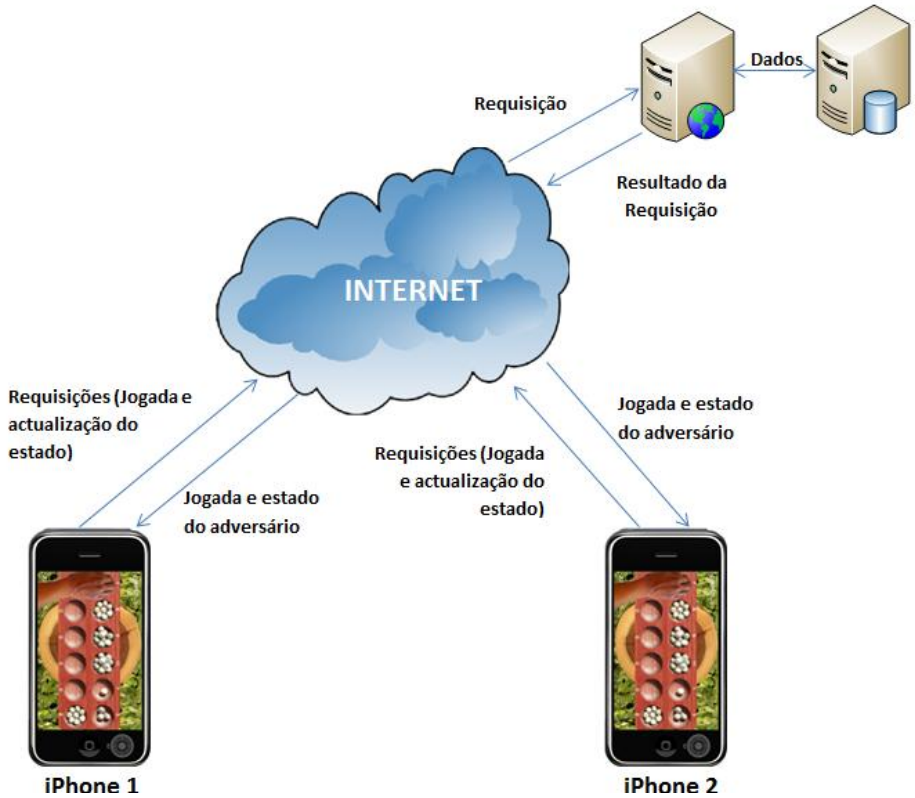


Figura 24: Jogo online

A comunicação entre a aplicação cliente e o servidor é feito com a função NetSend() do SDK, que envia requisições http do cliente para o servidor.

*intNetSend(char *url, int(*callback)(int event, char *response, int length));*

Parâmetros:

url – a URL pela qual envia o pedido.

Callback – a função que vai receber a reposta do pedido, e recebe os seguintes parâmetros:

Event – indica o resultado do pedido. Se for zero significa que o pedido falhou, e um significa que o pedido foi bem sucedido.

Response – contém o buffer de caracteres, que corresponde a resposta do servidor.

Length – indica o tamanho do buffer.

O trecho de código a seguir mostra um exemplo do uso da função:

```
void login(int opt)// opt : 1 - register , 2 - login
{
 char user[30];
 char pwd1[50];
 char Buffer[255];

 // user e pwd1 recebe o texto digitado nas caixas de texto txt_nickname e
txt_pwd1
 EditGetText(txt_nickname,user);
 EditGetText(txt_pwd1,pwd1);

 if(opt==1)
    sprintf(Buffer,"http://localhost/oril/login.php?user_name=%s&password=%s&opt=1
    ",user,pwd1);
 else
    sprintf(Buffer,"http://localhost/oril/login.php?user_name=%s&password=%s&opt=2
    ",user,pwd1);

 // envia a requisição da url guardado em Buffer
```

```
 NetSend(Buffer,OnLogin);
}

//=============================================

IntOnLogin(intevent, char *response, int length)
{
	if(event==0) // verifica se o pedido foi efectuado
		TextSetText(msg_login, "FALHA NA CONEXÃO COM O SERVIDOR");
	else
	{	// processa a resposta enviada pelo servidor
		if(response[0]=='0')
		{
			TextSetText(msg_login, "LOGIN EFECTUADO");
			mostra_lista_online();

		}
		elseif(response[0]=='1')
			TextSetText(msg_login, "LOGIN NÃO EFECTUADO");
		elseif(response[0]=='2')
			TextSetText(msg_login, "NICKNAME INDESPONIVEL");
		elseif(response[0]=='3')
			TextSetText(msg_login, "NICKNAME REGISTADO");
	}
	Returnevent;
}
```

6.6. ANIMAÇÕES

O movimento das imagens e sons, para muitos utilizadores são tão importante como a inteligência artificial porque quando mais próximo da realidade o jogo estiver mais divertido será. Por isso não poderíamos deixar de adicionar a nossa aplicação essa funcionalidade.

6.6.1. O MOVIMENTO DA MÃO

Após a escolha da jogada por parte de um dos jogadores este será executado com a animação de uma figura de uma mão, simulando a realidade e quando a mão passa por um buraco para apanhar ou soltar pedra(s), será executado um som para dar ao utilizador a percepção do apanhar ou soltar de pedras.

O movimento da mão é feito baseado nas coordenadas do ecrã do telemóvel onde o sistema de coordenadas está conforme a Figura 26.

Figura 25: Coordenadas do ecrã

A função que faz o movimento da mão funciona do seguinte modo: recebe inicialmente o buraco onde iniciar a jogada e o número de buracos por onde ele vai passar é determinado pelo número de pedras existentes no buraco inicial. E o deslocamento da mão é feito comprado as coordenadas onde a mão se encontra com as da sua próxima paragem, tendo em conta que cada vez que a mão chega a um buraco pára para fazer o efeito de soltar ou apanhar pedras.

O diagrama da Figura 27 ilustra o algoritmo que controla o movimento da mão, considerando o que a mão em questão a movimentar é a do jogador da linha de baixo.

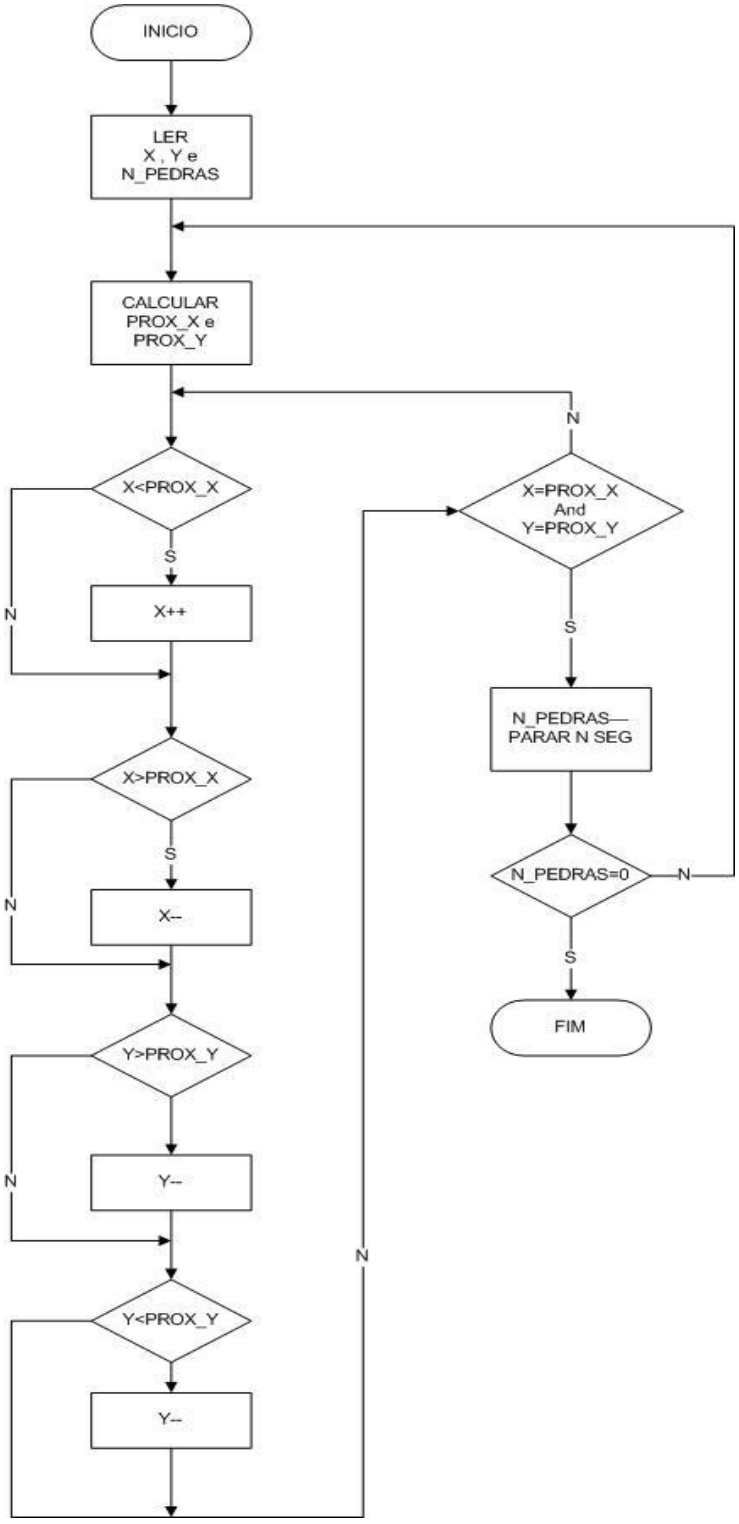


Figura 26: Diagrama de movimento da mão

A função que faz o movimento da mão e chamado dentro da função OnTimer, que nos permite fazer o GameLoop e a view da imagem da mão será actualizado a cada chamada e essa actualização é feita com a função ViewSetxy.

intViewSetxy(intvw, int x, int y);

Parâmetros:

VW – identificador retornado pela função ViewAdd();

X – nova coordenada x;

Y – nova coordenada y.

Exemplo do uso da função:

```
Voidiniciar_jogo_landscape()
{
	. . .

	user_hand=ViewAdd("user_hand.png",390,325); // inicialização da
viewuser_hand
	. . .

}
IntMovHand()
{
	. . .
	// atribuição de novas coordenadas a viewuser_hand
ViewSetxy(user_hand,posicao_mao_jogada.posicao_x,posicao_mao_jogada.posicao_y);

	. . .
}
```

6.7. O SERVIDOR

O servidor é composto por uma aplicação PHP e uma base dados MySQL. No nosso caso recorremos ao uso de um VPS (Virtual Private Server) que é um servidor em ambiente compartilhado que possui acesso root (administrador) e processos independentes para cada conta VPS criada, funciona assim como todo computador, cada conta VPS no servidor possui seu sistema independente, ou seja você pode configurá-lo de acordo com a sua real necessidade (instalar novos programas, etc...). A ideologia de um VPS é simples, temos por exemplo, um servidor extremamente robusto, dividido por várias máquinas virtuais através de técnica de virtualização (Ex. vmware, xen)[1].

Características do VPS utilizado.

CPU: 1000MHZ GUARANTEED
Memory: 512MB GUARANTEED
Disk Space: 20GB (RAID BASED CONFIG)
Bandwidth: 1000GB/MONTH
DNS Addresses:oril.dyndns.org
Sitema Operativo: Ubunto 8.03

[1]Wikipedia:VPS, http://pt.wikipedia.org/wiki/vps, 2011-02-20, 22:45

6.7.1. A APLICAÇÃO SERVIDOR

A aplicação do servidor, feito em PHP contém os seguintes módulos:

Login e registos de utilizadores – é o módulo responsável para processar os dados enviados pela aplicação cliente com a função NetSend() onde são enviados pela URL dados de login ou registo de novos utilizadores. O diagrama da Figura 28 mostra o algoritmo do módulo.

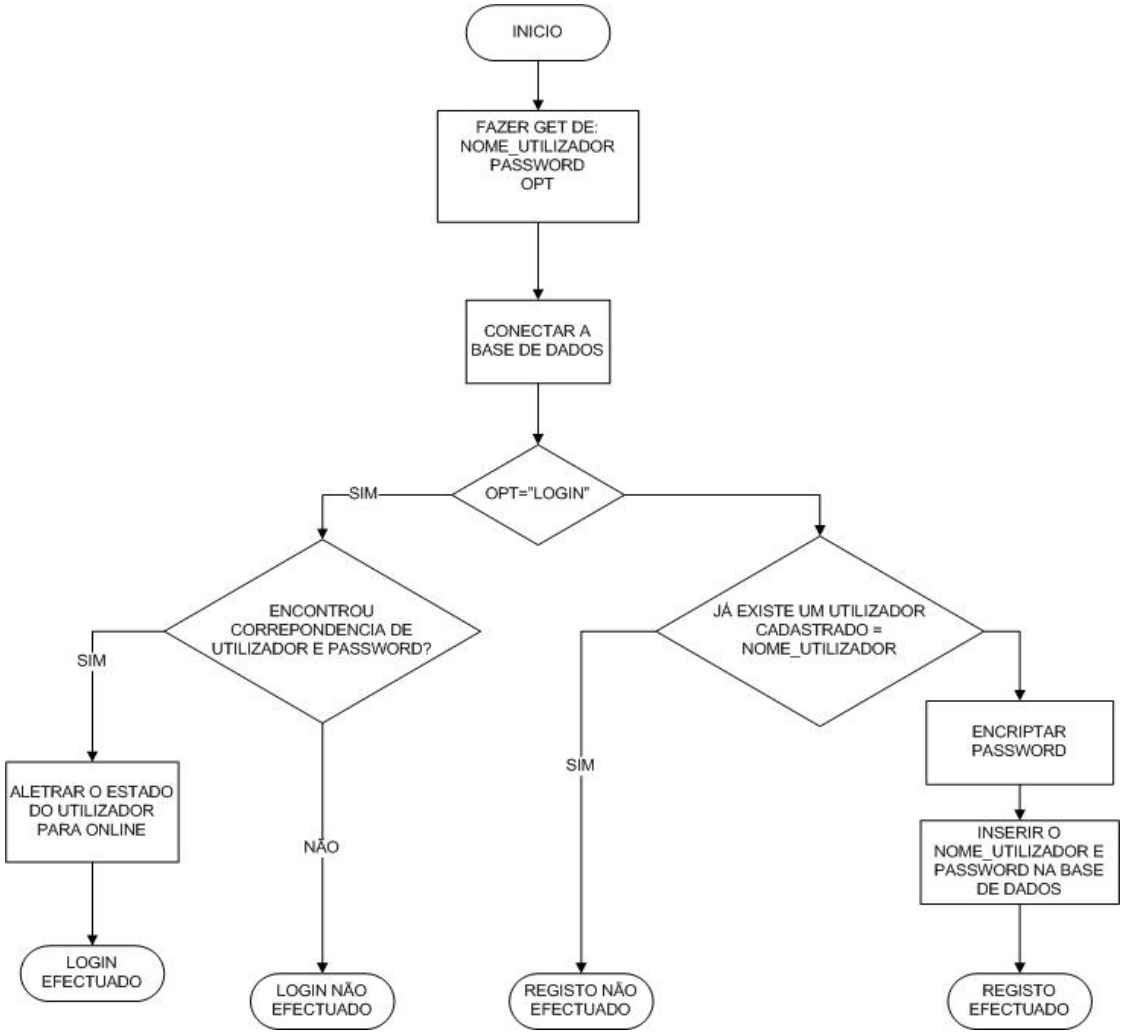


Figura 27: Diagrama de login e registo de jogadores

Registo de jogadas e controlo do estado dosutilizadores – com oinício de uma partida online a execução de cada lance no programa cliente enviará para o servidor, dados relativamente a configuração do tabuleiro e de pontuação dos jogadorese estes serão registados na base de dados.

A aplicação cliente saberá se será o turno do jogador enviando requisições periodicamente ao servidor, onde o resultado será o identificador do último jogador que efectuou um lance e caso esse identificador for do adversário a aplicação sai do modo de espera e desbloqueia o tabuleiro para o utilizador executar o seu lance.

Também quando é o turno do utilizador enquanto este não efectuar o lance será enviado periodicamente requisições para o servidor informando que o utilizador encontra-se online. Porque no servidor desencadeia um evento, num período de tempo mais longo que o das requisições enviadas pela aplicação cliente e se entre o tempo de o evento N e N+1 não foi registado nenhuma requisição por parte do cliente o estado do utilizador será alterado automaticamente de online para offline. E na próxima requisição da aplicação que ainda continua online será notificado e também alterará o estado do utilizador.

6.7.2. A BASE DE DADOS

Para guardar os dados relativamente aos jogadores e jogadas criamos uma base de dados para tal. Onde a base de dados tem o seguinte modelo ER da Figura 28.

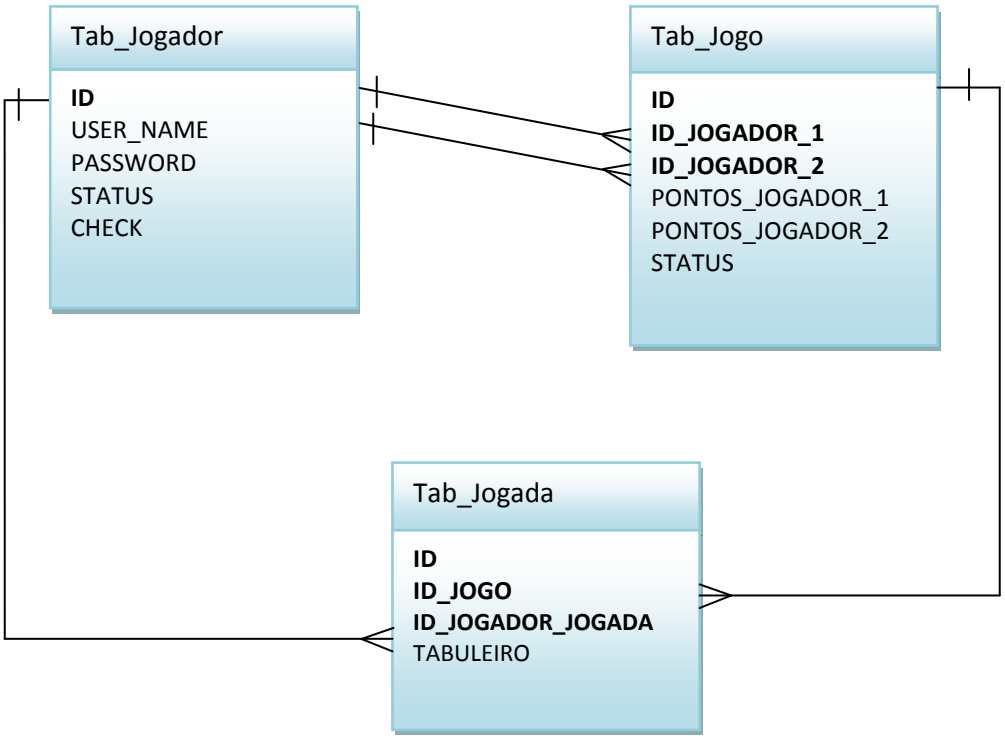


Figura 28: Modelo ER

Para melhor entendimento do modelo da figura a segue-se o dicionário de dados.

TABELA JOGADOR

Campo	Tipo de Dados	Tamanho	Descrição
ID	int		Id do jogador
USER_NAME	varchar	30	NickName do jogador
PASSWORD	varchar	50	Palavra passe encriptada com md5
STATUS	bool		Estado do jogador: 0-off-line 1-online
CHECK	DateTime		Hora da última requisição enviada para o servidor

Tabela 3: Tabela jogador

TABELA JOGO

Campo	Tipo de Dados	Tamanho	Descrição
ID	int		Id do jogo
ID_JOGADOR_1	int		Id do jogador que enviou o convite
ID_JOGADOR_2	int		Id do jogador que aceitou o convite
PONTOS_JOGADOR_1	int		Pontuação do jogador 1
PONTOS_JOGADOR_2	int		Pontuação do jogador 1
STATUS	bool		Estado do jogo: 0-fim 1-em andamento

Tabela 4: Tabela jogo

TABELA JOGADA

Campo	Tipo de Dados	Tamanho	Descrição
ID	int		Id da jogada
ID_JOGO	int		Id do jogo
ID_JOGADOR	int		Id do jogador que efectuou a jogada
TABULEIRO	int		Nº de pedras de cada casa do tabuleiro, separados por vírgula.

Tabela 5: Tabela jogada

EVENTO

O evento cujo código apresentado abaixo é para controlar os utilizadores que estão online, onde ele estará agendado para correr a cada quinze segundo e verifica se a aplicação cliente não fez uma actualização do campo check a pelo menos dez segundos atrás da hora actual e coloca o campo status a zero (offline). Isso tendo

em conta que aplicação cliente irá actualizar o campo check a intervalos de dez segundos.

```
DELIMITER |
CREATE EVENT onlinne
    ON SCHEDULE
      EVERY 15 SECOND
    DO
      BEGIN
        UPDATE jogador SET status=0 WHERE `check` <SUBTIME (NOW(),'00:00:10');
END
DELIMITER ;
```

7. iPHONE

O iPhone é um smartphone desenvolvido pela Apple Inc. com funções de iPod, câmera digital, internet, mensagens de texto (SMS), visual voicemail, conexão wi-fi local e, actualmente, suporte a vídeochamadas (FaceTime). A interacção com o usuário é feita através de uma tela sensível ao toque. A Apple registou mais de duzentas patentes relacionadas com a tecnologia que criou o iPhone.

Os modelos 2G e 3G saíram de linha. Nos Estados Unidos, o modelo 3GS de 8GB é vendido por US$99, e o iPhone 4 é vendido por 199 dólares (16 GB) e 299 dólares pelo modelo de 32 GB na Apple Store e pela AT&TMobility. Anunciado em 9 de Janeiro de 2007, o iPhone foi lançado no dia 29 de Junho de 2007 nos EUA, em 9 de Novembro de 2007 na Alemanha e no Reino Unido, e em 29 de Novembro na França. Em 2008 foi lançado no mercado asiático e resto da Europa. Em Portugal,

inicialmente vai ser vendido pela Vodafone. Foi lançado em 11 de Julho de 2008, e até Janeiro de 2008 foram vendidos quatro milhões de iPhones e somente durante o fim-de-semana de lançamento do iPhone 3G, a Apple afirma ter vendido 1 milhão de unidades do aparelho[1].

O iPhone 4 começou a ser comercializado no mercado cabo-verdiano em Fevereiro 2011.

[1]Wikipedia: iPhone, http://pt.wikipedia.org/wiki/IPhone, 2011-12-01, 22:15

8. INSTALAÇÃO PROGRAMA

A instalação do jogo pode ser feita através do computador pelo itunes como também directamente do iphone com uma ligação internet. Basta conectar no AppStore da Apple, depois fazer uma pesquisa de oril onde ira aparecer o ícone do jogo.

E por fim clicar em cima da palavra Free e automaticamente será feito o download do jogo e instalado no telemóvel, e um ícone do jogo será adicionado no ambiente de trabalho como mostra a fig 30

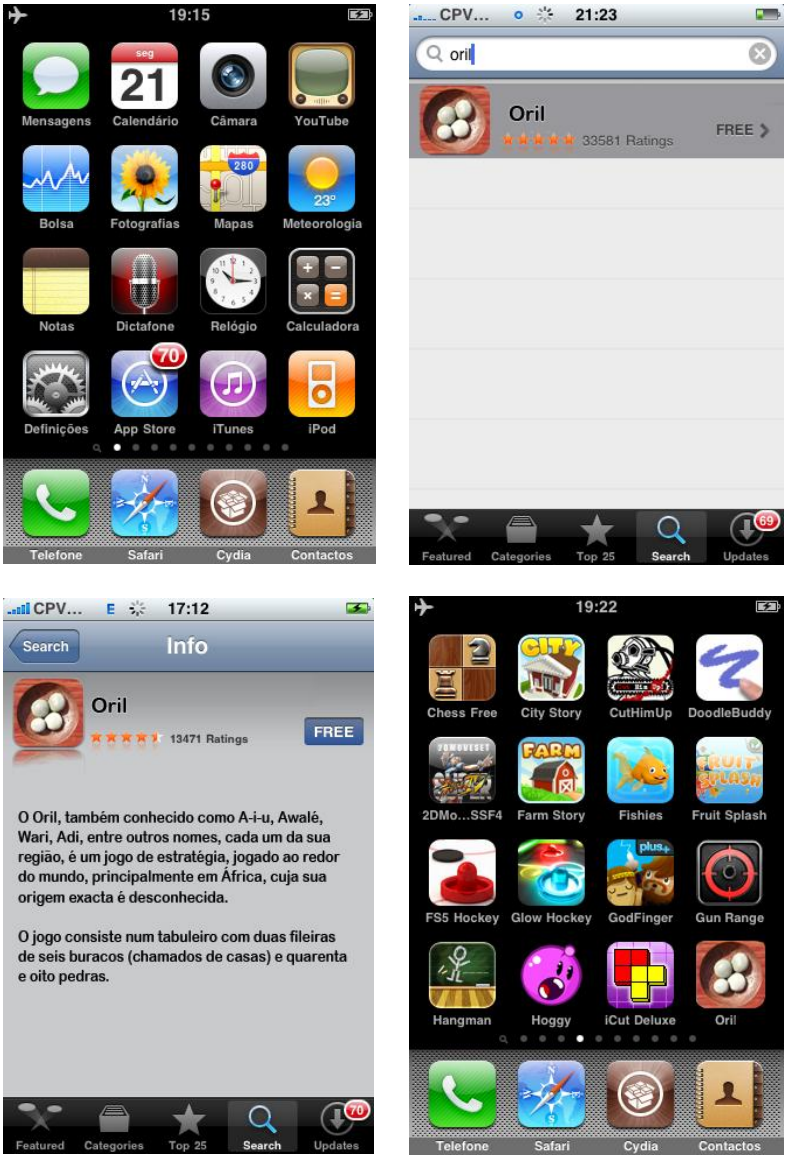


Figura 29 : Instalação

9. CONCLUSÃO

Neste trabalho tivemos a oportunidade de apresentar metodologias para criação de um software para um telemóvel, mais concretamente um iPhone capaz de jogar oril, onde foram empregadas técnicas de inteligência artificial conducentes para a sua concretização.

Apesar dos objectivos inicialmente estabelecidos foram todos alcançados, é importante realçar que há a necessidade de melhorar alguns aspectos como a base de conhecimento e métodos de aprendizagem da máquina, tais métodos que melhorarão a eficiência e velocidade do jogo da máquina.

O desenvolvimento deste projecto foi de suma importância para a nossa discência, uma vez que a teoria dos jogos é um assunto complexo no campo de estudos de situações de conflitos, tomada de decisões e desenvolvimento de estratégias, que surpreende a cada nova aplicação.

10. SUGESTÕES PARA TRABALHOS FUTUROS

O projecto desenvolvido, não foram efectuados testes exaustivos, pelo que aconselhamos o lançamento de uma versão beta para a recolha de subsídios para a melhoria da sua primeira versão final.

Para as próximas versões preconizamos a realização de um trabalho de campo para a recolha de dados acerca de técnicas e estratégias do jogo do oril para a construção da base de conhecimento do jogo e assim como de uso das jogadas online armazenadas no servidor.

Em relação a escolhaSDK recomendamos o uso do iOSSDK da Apple que é o mais completo, visto que com o DragonfireSDK tivemos que desenvolver muitas ferramentas que já estão disponíveis no iOSSDK.

11. BIBLIOGRAFIA

- Graça, Albertino. Regras, Estratégias e Teorias do Jogo de Oril (1ª Edição). Mindelo, Edição ONDS, Novembro de 1998.

- Damas, Luis Manuel Dias. Linguagem C (3ª Edição), FCA Editora.

- Wikipedia: iPhone, http://pt.wikipedia.org/wiki/IPhone, 2011-12-01, 22:15

- Wikipedia:VPS, http://pt.wikipedia.org/wiki/vps, 2011-02-20, 22:45

- Wikipedia:Teoria dos Jogos, http://pt.wikipedia.org/wiki/Teoria_dos_jogos, 2010-11-21, 10:25

- Wikipedia: Inteligência Artificial, http://pt.wikipedia.org/wiki/Intelig%C3%AAncia_artificial, 2010-11-20, 18:38

- Direne, Alexande. Universidade Federal de Panamá: Visão Geral Sobre Inteligência Artificial, http://www.nce.ufrj.br/GINAPE/VIDA/ia.htm, 2010-11-20, 18:10

- DragonfireSDK: DragonFireSDKHelp, http://www.dragonfiresdk.net/help/DragonFireSDKHelp.html

12. ANEXOS

Um CD contento:

- O código fonte da aplicação.

- Ficheiros de imagens e sons utilizados.

- Relatório do Projecto.