Jogo móvel BeeFish: Inteligência Artificial para movimentar personagens animais

Carolina Meireles José Soeiro Ana Paula Cláudio
BioISI – Biosystems & Integrative Sciences Institute, Faculdade de Ciências da Universidade de Lisboa
Campo Grande, Lisboa
apc@di.fc.ul.pt,{fc34517,fc40493}@alunos.fc.ul.pt

Abstract
This paper presents a mobile game created with the purpose of dissemination of the European Project ASSISIbf, which studies interactions between robots and animals. The game offers two different types of gameplay, with different characters, some of them (robots) controlled directly by the user and others (animals) by Artificial Intelligence. We describe the movement behaviours of the animals, which are influenced by the robots, with the inclusion of some excerpts of pseudocode.

Keywords
Game logic and design, mobile games, artificial intelligence, movement behaviours

1. INTRODUÇÃO

Este artigo descreve um videojogo desenvolvido no contexto do projeto europeu ASSISIbf – "Animal and robot Societies Self-organize and Integrate by Social Interaction (bees and fish)" –, cujo principal objetivo é estabelecer uma sociedade robótica capaz de desenvolver por conta própria canais de comunicação com sociedades animais (em particular, cardumes de peixes e enxames de abelhas jovens que ainda não voam) [ASSISIbf15].

Os robôs usados nas experiências deste projeto designam-se genericamente por CASU (do inglês "Combined Actuator-Sensor Units"). Em particular, bee-CASU é o nome dos robôs usados nas experiências com abelhas e fish-CASU o nome dos robôs usados nas experiências com peixes.

Os robôs das abelhas não são móveis, mas têm perceção e ação (sensores e atuadores). Os sensores são de proximidade, por infravermelhos ativos, de temperatura e de vibração. Os atuadores são de temperatura e de vibração.

Os robôs dos peixes são móveis, não têm sensores e são compostos por duas partes: a que está dentro do aquário, com a forma de um peixe, e a que está debaixo (e fora) do aquário e que, com o seu próprio movimento sobre rodas, faz mover a primeira através de um íman.

Os resultados deste projeto poderão vir a ter impacto na agricultura e na proteção ambiental. O projeto está a ser desenvolvido por seis instituições provenientes de diferentes países da União Europeia.

O videojogo apresentado, que denominámos por BeeFish, pode ser jogado em dispositivos móveis, é de carácter lúdico e destina-se à ampla disseminação do projeto. Para cumprir este objetivo os conceitos-base do jogo tiveram a sua inspiração no projeto. Foram criados dois tipos de jogabilidade diferentes: um com abelhas, outro com peixes e ambos com CASU inspirados nos robôs reais.

Recorreu-se ao *software* de criação de jogos Stencyl [Stencyl15] que permite o desenvolvimento e publicação de videojogos 2D para computadores, dispositivos móveis e para a Web. A lógica de jogo está estruturada em módulos, correspondendo a comportamentos, associados às personagens ou aos cenários do jogo. Estes comportamentos são construídos a partir de um conjunto de "blocos de código" de ações disponíveis, usando uma linguagem de programação visual.

Este artigo está organizado do seguinte modo: a secção 2 descreve o jogo BeeFish, a secção 3 detalha a Inteligência Artificial (IA) dos comportamentos de movimento implementados, com a inclusão de algum pseudo-código, a secção 4 discute os comportamentos implementados e a secção 5 apresenta conclusões sobre o trabalho realizado, mencionando a avaliação efetuada por testes de utilizador.

2. DESCRIÇÃO DO JOGO

O jogo BeeFish oferece dois tipos de jogabilidade diferentes: um conjunto de níveis com abelhas e bee-CASU (daqui em diante referido como o jogo Bee) e outro com peixes e fish-CASU (o jogo Fish). Na versão atual do jogo, existem seis níveis do jogo Bee e seis níveis do jogo Fish, com nível de dificuldade crescente, podendo ser escolhidos num menu inicial. De um modo geral, o propósito do jogador em cada nível é controlar os CASU de forma a influenciar o comportamento das abelhas ou peixes para que estes consigam alcançar uma meta predefinida.

Em ambos os jogos, a cor tem um papel fundamental na lógica do jogo. No jogo Fish, os CASU movimentam-se por percursos controlados no cenário, à semelhança dos seus congéneres das experiências reais. No jogo Bee, os CASU estão associados a posições no tabuleiro do jogo, refletindo a posição fixa que os robôs reais têm. Contudo esta posição no tabuleiro pode mudar, como a seguir se explica.

Nas subsecções seguintes detalhamos cada uma das jogabilidades.

2.1 Jogo Bee

No jogo Bee cada nível tem um tabuleiro com casas hexagonais que se assemelham a favos de mel. Os favos podem conter um CASU que tem associada uma cor que "irradia" para os favos vizinhos, i.e., os favos imediatamente adjacentes ao que contém o CASU obtêm a cor desse CASU. Os CASU podem ter três cores diferentes: vermelho, verde e azul. Todavia, favos que recebam cor de dois CASU de cores diferentes recebem a cor composta pelas cores dos dois CASU (resultando, por aplicação do modelo aditivo RGB, em favos com cores magenta, ciano e amarelo). Já os favos que recebam cor de três CASU de cores diferentes recebem a cor cinzento-clara (e não branca, como no modelo RGB, por motivos de visualização).

As abelhas jovens, que ainda não voam, caminham sobre os favos, descrevendo um percurso desde um favo inicial até um favo meta, ambos marcados no tabuleiro logo desde o início do nível. Cada abelha tem uma cor própria, que não muda ao longo do nível, e em cada instante apenas se pode mover para um favo adjacente que tenha a sua cor e que não esteja ocupado por um CASU ou por outra abelha. O propósito do jogador é alterar a cor dos favos de modo a criar um caminho para as abelhas através do tabuleiro, desde o favo inicial até ao favo meta.

Para atingir o seu objetivo, o jogador tem de usar os CASU, o que pode fazer de duas maneiras: mudando a sua cor ou movendo-os no tabuleiro. Todavia, existem algumas restrições. Os CASU são de dois tipos diferentes: CASU-M e CASU-C. Os CASU-M podem ser movidos de um favo para outro adjacente, um número limitado de vezes, mas têm cor fixa. Os CASU-C não podem ser movidos mas a sua cor pode ser alterada um número ilimitado de vezes. Os CASU-M são representados por círculos com um número que corresponde ao número de vezes que o jogador os pode mover para um favo adjacente arrastando-o. Por cada movimento este valor é decrementado uma unidade. Os CASU-C são também circulares e no seu centro têm dois semicírculos de cores distintas lado a lado. O jogador altera a cor do CASU-C usando um movimento de *swipe* para o lado do semicírculo com a cor pretendida.

No início do jogo, o tabuleiro pode já ter alguns CASU colocados para o jogador controlar, variando de nível para nível. No decurso do jogo mais CASU vão sendo criados: à esquerda do tabuleiro existe uma pilha de CASU, o CASU Pipe, mostrando os próximos CASU que podem ser colocados no tabuleiro. O CASU na base do Pipe pode ser retirado pelo jogador e colocado num favo do tabuleiro à sua escolha. Contudo, existe, um temporizador com um valor inicial (que pode variar em cada nível) que vai sendo decrementado a cada segundo e que quando chega a zero, desencadeia a colocação do CASU num favo aleatório, o que pode ser inconveniente para o jogador. Quando o CASU na base é colocado, o CASU acima no Pipe toma o seu lugar na base e o temporizador reinicia.

Em suma, há três tipos possíveis de movimentos que o jogador pode efetuar:

- mudar a cor de um CASU-C no tabuleiro;
- deslocar um CASU-M para um favo adjacente e não ocupado no tabuleiro;
- colocar num favo não ocupado do tabuleiro o CASU que se encontra na base do Pipe.

A dificuldade do jogo aumenta de nível para nível através da variação do valor inicial do temporizador do CASU Pipe, do número de abelhas no tabuleiro e das suas cores e da configuração inicial de CASU no tabuleiro. A Figura 1 mostra dois níveis diferentes do jogo Bee.

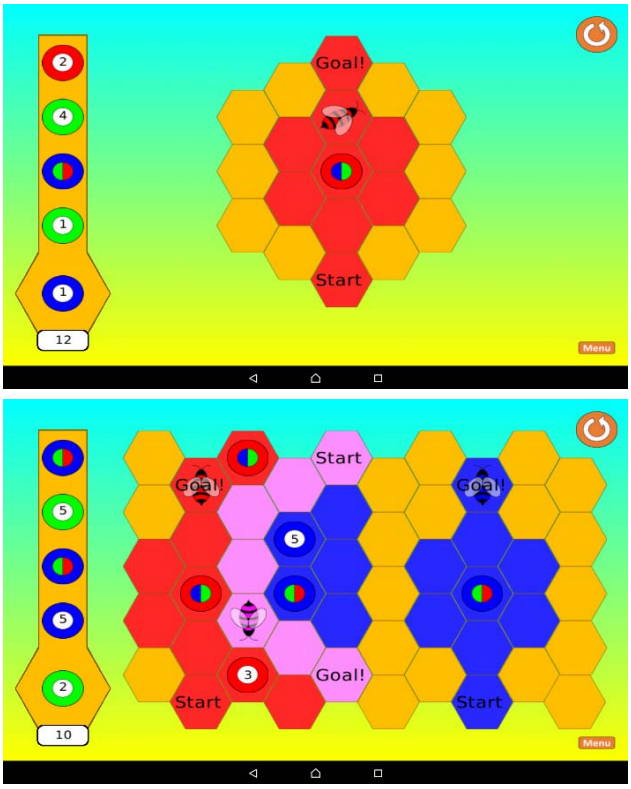

Figura 1: *Screenshots* de dois níveis do jogo Bee (nível 1 no topo e nível 4 em baixo), com diferentes graus de dificuldade. À esquerda encontra-se o CASU Pipe com dois tipos de CASU e na base deste pode ser visto o temporizador.

2.2 Jogo Fish

O jogo Fish tem lugar num aquário. Existem alguns robôs (fish-CASU) de diferentes cores, com a forma de peixes grandes, espalhados pelo aquário. Os peixes, muito mais pequenos que os robôs, também possuem cores e reagem à presença dos robôs da mesma cor, seguindo-os. Também existem algumas saídas circulares de diferentes cores no aquário. O objetivo do jogador é conduzir cada um dos peixes até uma saída da sua cor ou branca (uma saída branca serve para peixes de qualquer cor). Cada saída, porém, tem um número máximo de peixes que pode acolher. As saídas são representadas por círculos preenchidos a preto com um contorno indicativo da sua cor e dois valores numéricos que indicam: número de peixes que já usaram esta saída/número máximo de peixes suportados por esta saída. Por exemplo, a indicação 3/6 significa que 3 peixes já usaram esta saída e que o limite desta é de 6 peixes.

Cada CASU move-se continuamente por um caminho predefinido (descrevendo elipses ou retângulos, para a frente e para trás, etc.). O jogador pode trocar as cores dos robôs, para alterar os robôs que os peixes seguem e dessa maneira guiá-los até às saídas. Podem também, em alguns níveis, existir obstáculos no aquário, por exemplo simulando rochas para obstruir o percurso dos peixes.

Os fish-CASU podem ter uma de três cores: vermelho, verde e azul. Em cada nível, existe um número variável de CASU e de peixes, podendo existir vários CASU da mesma cor ou um CASU sem peixes da mesma cor. A Figura 2 ilustra dois níveis do jogo Fish.

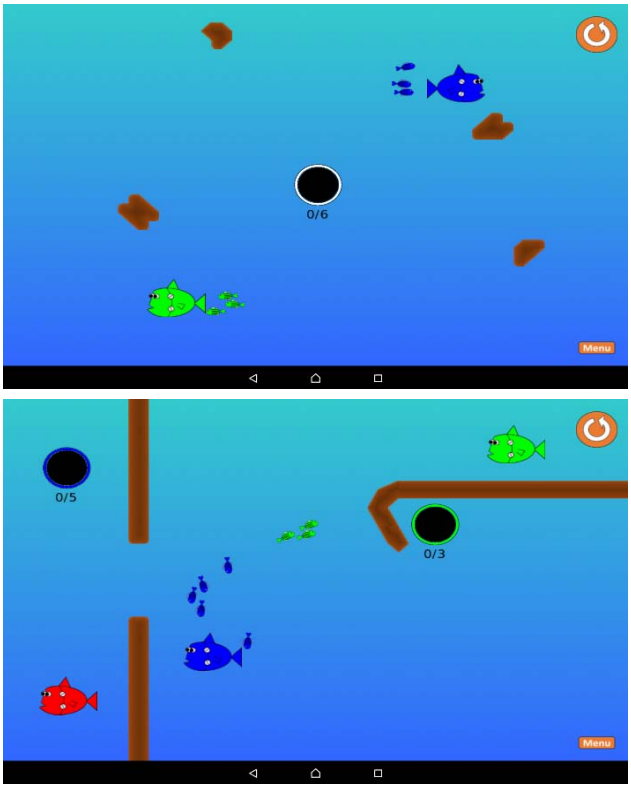

Figura 2: *Screenshots* de dois níveis do jogo Fish (nível 3 no topo e nível 5 em baixo), com diferentes graus de dificuldade.

3. COMPORTAMENTOS DE MOVIMENTO

No jogo BeeFish, enquanto os robôs são controlados pelo jogador, as abelhas e os peixes apenas podem ser controlados indiretamente, pela interação do jogador com os robôs. Isto faz um paralelismo com o objetivo principal do projeto de influenciar os animais através de ações de robôs controlados por humanos.

Por esta razão, foi necessário implementar no jogo IA nas personagens animais, para que estas pudessem responder ao meio onde se encontram inseridas. Em particular, tivemos de implementar o movimento das abelhas pelos favos e dos peixes pelo aquário, o que será detalhado nas subsecções seguintes.

3.1 Movimento das abelhas

No jogo Bee, o jogador precisa de usar os CASU para colorir os favos de forma a criar caminhos de uma certa cor para que as abelhas dessa cor possam passar. As abelhas movem-se sozinhas, de favo em favo da sua cor, tentando atingir um favo meta predeterminado. Implementámos IA num comportamento associado às abelhas, de modo a que uma abelha possa perscrutar a sua vizinhança (os favos adjacentes àquele em que se encontra) para decidir o próximo favo para onde se mover. De seguida detalhamos a lógica de movimento criada, com o auxílio de pseudo-código.

Cada abelha possui três atributos, armazenados em variáveis: o seu favo de partida (“StartingComb”), o seu favo de chegada (“GoalComb”) e a sua cor (“Colour”). Outras variáveis são inicializadas logo no início do nível:

ReachedGoal = false

CanMove = false

HousingComb = StartingComb

“ReachedGoal” é uma variável booleana que é inicializada a *false* e que se torna *true* quando a abelha alcança “GoalComb”. “CanMove” é uma variável booleana que armazena, em cada instante, se a abelha já decidiu para que favo se mover a seguir. “HousingComb” armazena o favo em que a abelha se encontra no momento.

O comportamento da abelha é constituído por três fases:

- a abelha tem de decidir o próximo favo para onde ir;
- a abelha já tem um alvo definido (“TargetComb”) e move-se nessa direção;
- a abelha chegou ao “TargetComb” e prepara-se para tomar uma nova decisão.

Em cada *frame* do jogo, verifica-se em que fase do comportamento a abelha se encontra e esta age em função disso.

Na fase de decisão, uma lista (“PossibleMoves”) é usada para guardar os favos para onde a abelha se pode mover. Estes favos consistem nos vizinhos (adjacentes) de “HousingComb” que se encontram desocupados (sem

outra abelha ou um CASU) e têm a mesma cor que a abelha. Se existirem favos para onde a abelha se pode mover, dois deles (não necessariamente diferentes) são selecionados: o mais próximo de "GoalComb" e um aleatoriamente escolhido da lista. De seguida, a lista é esvaziada e preenchida novamente com apenas estes dois favos, para que um deles possa ser aleatoriamente selecionado como o favo para onde a abelha se vai mover (e guardado em "TargetComb"). A variável "CanMove" é então alterada, passando a indicar que a decisão foi tomada.

Na fase de movimento em direção a "TargetComb", a velocidade da abelha é alterada, na direção do favo alvo. Na fase em que a abelha já atingiu "TargetComb", este passa a ser considerado como o novo "HousingComb", a velocidade da abelha passa a zero e a variável "CanMove" é novamente alterada para indicar que é preciso tomar uma nova decisão. Se "TargetComb" coincidir com "GoalComb", então a variável "ReachedGoal" é alterada para assinalar que a abelha chegou à meta.

Em seguida, apresenta-se em pseudo-código o raciocínio explicado acima:

```
If not ReachedGoal
  If not CanMove
    PossibleMoves = encontra os favos possíveis
    If PossibleMoves not empty
      ClosestToGoal = calcula o favo mais próximo da meta
      TargetComb = escolhe favo aleatório em PossibleMoves
      PossibleMoves = {ClosestToGoal, TargetComb}
      TargetComb = escolhe favo aleatório em PossibleMoves
      CanMove = true
  Else If Abelha não está em TargetComb
    Direciona Abelha para TargetComb
  Else
    HousingComb = TargetComb
    Pára Abelha
    CanMove = false
    If TargetComb=GoalComb
      ReachedGoal = true
```

No que diz respeito à orientação da abelha, à animação da personagem é aplicada uma rotação para corroborar a direção do movimento da abelha. O ângulo da rotação a aplicar é calculado em função do vetor velocidade da abelha em cada instante.

3.2 Movimento dos peixes

No jogo Fish, o jogador precisa de trocar as cores dos fish-CASU para guiar os peixes até às saídas das suas cores. Os peixes movem-se sozinhos, seguindo um dos robôs da sua cor. Implementámos IA num comportamento associado aos peixes, de modo a que um peixe possa escolher e seguir um robô, mantendo porém uma pequena distância deste. O tratamento de colisões entre os peixes e destes com as rochas é feito automaticamente pelo motor de jogo do Stencyl.

Cada peixe tem dois atributos, armazenados em variáveis: a sua cor ("Colour") e a margem de distância a que o peixe se deve manter do robô que está a seguir ("Margin"). Outra variável é inicializada logo no início do nível:

```
NearGoal = false
```

"NearGoal" é uma variável booleana que é *true* se o peixe encontrou alguma saída da sua cor (ou branca) perto de si e *false* caso contrário. Cada saída tem um determinado alcance de deteção, isto é, uma distância máxima a partir da qual os peixes deixam de perseguir os robôs e passam a dirigir-se para a saída.

O comportamento do peixe é constituído por duas fases:

- o peixe segue um dos robôs da sua cor e tenta detetar uma saída perto de si;
- o peixe encontrou uma saída e, caso esta ainda possa acolher mais peixes, dirige-se para ela.

Em cada *frame*, é avaliado em que fase do comportamento o peixe se encontra e este age em função disso.

Na primeira fase, é calculado o robô da cor do peixe que está mais próximo do mesmo e o ângulo do vetor que vai desse robô para o peixe ("AngleFromRobot"). Se a distância entre o peixe e o robô for maior que "Margin", então o vetor velocidade do peixe é alterado segundo "AngleFromRobot". Caso contrário, é chamado um método (detalhado abaixo) que altera a direção do peixe de modo a evitar que este colida com o robô.

Para além disto, entre as saídas espalhadas no nível, procura-se uma que não se encontre cheia (consiga acolher mais peixes), que seja branca ou da cor do peixe e ainda que se encontre a uma distância do peixe menor que o seu alcance de deteção. Caso tal saída seja encontrada, a velocidade do peixe passa a zero e, após um segundo, a saída é guardada numa variável "GoalFound", a velocidade do peixe é alterada, na direção de "GoalFound", e a variável "NearGoal" é então alterada, passando a indicar que o peixe encontrou uma saída para si.

Na segunda fase, em que o peixe encontrou uma saída ("GoalFound"), é analisado se essa saída continua a poder acolher mais peixes (pode acontecer a saída ter ficado cheia depois do momento inicial em que o peixe a detetou). Caso possa, a velocidade do peixe é alterada, na direção de "GoalFound" e, quando suficientemente perto, o peixe atinge a saída e desaparece da cena. A distância a que isto acontece está relacionada com o diâmetro da saída, de modo a que o peixe desapareça quando passa a circunferência delimitadora. Caso a saída já esteja cheia,

a variável “NearGoal” é novamente alterada para indicar que o peixe tem de voltar a seguir os robôs e procurar nova saída.

Em seguida, apresenta-se em pseudo-código o raciocínio explicado acima. Note-se que “distance(A, B)” denota o cálculo da distância entre as posições de A e B.

```
If not NearGoal
  ClosestRobot = calcula o robô mais próximo de Peixe que tenha a sua cor
  DistanceFromClosest = distance(ClosestRobot, Peixe)
  AngleFromRobot = calcula o ângulo do vetor que vai de ClosestRobot para Peixe
  If DistanceFromClosest > Margin
    Direciona Peixe para ClosestRobot
  Else
    avoidCollisionWithClosestRobot()
  For each Saída in level
    If Saída not full
      If Saída for branca ou Colour
        If distance(Saída, Fish) < alcance de deteção de Saída
          Pára Peixe
          After 1 second
            GoalFound = Saída
            Direciona Peixe para GoalFound
            NearGoal = true
          Exit For
Else
  If GoalFound not full
    Direciona Peixe para GoalFound e fá-lo entrar quando suficientemente perto
  Else
    NearGoal = false
```

O método “avoidCollisionWithClosestRobot()”altera a direção do peixe de modo a evitar que este colida com o robô que está a seguir, sendo apenas usado quando a distância do peixe ao robô é menor que “Margin”. Além disso, a alteração apenas ocorre se o robô estiver a mover-se na direção do peixe. Assim, é necessário fazer uma comparação entre dois ângulos: “AngleFromRobot”, o ângulo do vetor que vai do robô para o peixe, e “RobotDirection”, o ângulo do vetor velocidade do robô.

Na Figura 3 está esquematizada a lógica utilizada para alterar a direção do peixe. A traço ponteado representa-se o vetor que vai do robô para o peixe e o ângulo correspondente, “AngleFromRobot”. A tracejado representa-se o vetor velocidade do robô e o ângulo correspondente, “RobotDirection”. A traço contínuo representa-se o vetor velocidade que é então aplicado ao peixe em função da diferença entre estes dois ângulos.

Em seguida apresenta-se em pseudo-código o raciocínio utilizado:

```
avoidCollisionWithClosestRobot():
  RobotDirection = calcula o ângulo do vetor velocidade do robô
  If AngleFromRobot – 45 <= RobotDirection <= AngleFromRobot
    Direciona Peixe segundo AngleFromRobot + 45
  Else If AngleFromRobot <= RobotDirection <= AngleFromRobot + 45
    Direciona Peixe segundo AngleFromRobot - 45
```

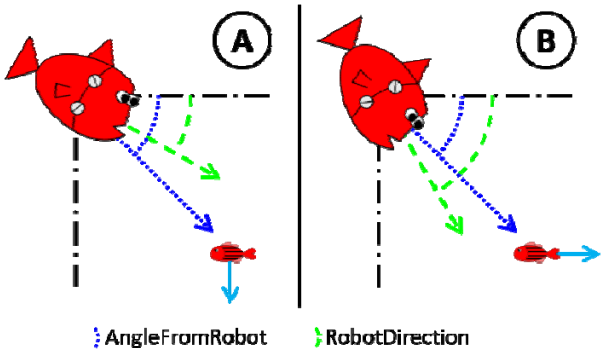

Figura 3: Esquema da lógica utilizada para alterar a direção do peixe em função da diferença entre “AngleFromRobot” e “RobotDirection”.

4. DISCUSSÃO DOS COMPORTAMENTOS

Na secção anterior foram descritos os comportamentos de IA desenvolvidos para o movimento das abelhas e peixes no jogo BeeFish. Estes comportamentos foram pensados de modo a assemelharem-se a comportamentos reais destes animais.

Para as abelhas, uma primeira abordagem seria que cada uma escolhesse sempre, entre os favos desocupados e da sua cor na sua vizinhança (os favos possíveis), aquele mais próximo da meta. Contudo, esta abordagem tem um problema: nem sempre é possível construir para a abelha um caminho que a aproxime progressivamente da meta; por vezes, devido ao bloqueio por CASU ou cores que já não se podem alterar, os únicos caminhos possíveis têm de levar a abelha a afastar-se inicialmente da meta, o que esta nunca faria se optasse sempre pelo favo mais próximo.

Assim, decidiu-se acrescentar aleatoriedade ao movimento da abelha, fazendo com que esta escolha, em cada movimento, entre o favo possível mais próximo da meta e um escolhido aleatoriamente entre os favos possíveis. Nesta escolha aleatória de um favo, cada um dos favos tem uma probabilidade de ser escolhido de 1/N, sendo N o número de favos para onde a abelha se pode mover. Uma vez escolhido o favo aleatório, volta a ser efetuada uma escolha entre este e o favo mais próximo, tendo cada um deles uma probabilidade de ½ de ser escolhido. O favo escolhido aleatoriamente pode coincidir com o favo mais próximo. Assim, as probabilidades finais de escolha são:

- $½ + ½_N$, para o favo mais próximo da meta;
- $½_N$, para cada um dos outros favos.

O comportamento resultante desta aproximação revela-se bastante natural, uma vez que na realidade as abelhas não fazem um percurso linear até ao seu objetivo, sendo esse percurso pontuado por pequenos desvios.

No caso dos peixes, o comportamento implementado, apesar de não criar explicitamente dinâmicas de grupo,

leva a que os peixes da mesma cor ajam em cardume, pois todos seguem, usualmente, um mesmo robô da sua cor. As colisões entre peixes são tratadas pelo motor de jogo do Stencyl, mas criam naturalidade no movimento em grupo, sendo que os peixes se vão mantendo a alguma distância uns dos outros, mas mantêm-se num grupo coeso e alinhado por seguirem o mesmo robô.

Assim, o comportamento criado aproxima-se do comportamento de *flocking* utilizado para representar as dinâmicas de animais que se deslocam em grupo, derivado do comportamento manifestado por bandos de pássaros, mas que, enquanto modelo matemático, se generalizou para representar os movimentos de outras espécies de animais. Este comportamento foi simulado pela primeira vez num computador por Craig Reynolds em 1987, com o seu programa de simulação "Boids" [Reynolds87]. Neste comportamento, o movimento de cada indivíduo é calculado em função dos movimentos dos indivíduos vizinhos, existindo três forças a atuar sobre cada um:

- Separação, para que o indivíduo mantenha uma distância mínima dos seus vizinhos;
- Coesão, para que o indivíduo se dirija para a posição média dos seus vizinhos;
- Alinhamento, para que o indivíduo alinhe a sua velocidade (magnitude, sentido e direção) com a velocidade média dos seus vizinhos.

Este comportamento é usado em simulações e também em alguns jogos que o justifiquem, como é o caso do jogo "The Digital Aquarist" [Schikarski15]. No jogo BeeFish, o comportamento criado resulta num movimento que parece adequado e natural, verificando-se uma dinâmica semelhante ao *flocking* sem que este tenha sido explicitamente implementado.

Numa versão inicial do comportamento aplicado aos peixes, estes aproximavam-se do robô até colidirem com este e mantinham-se colados ao robô a maior parte do tempo. Assim, optou-se por criar uma margem de distância entre os peixes e os robôs. Os peixes têm o comportamento normal de seguir os robôs até que fiquem demasiado próximos dos mesmos. Quando se aproximam demasiado, é avaliada a direção do robô para perceber se este está prestes a colidir com os peixes. Caso isso se verifique, a direção dos peixes é alterada para que estes se desviem, para um lado ou para o outro, do caminho do robô.

É importante ainda referir o que acontece com o movimento dos peixes em dois casos particulares: quando existe mais que um robô da cor dos peixes e quando não existe nenhum robô que o peixe consiga detetar. Esta segunda situação nunca se verifica com a jogabilidade atual do jogo Fish, mas foi considerada em protótipos anteriores, em que os peixes possuíam uma distância máxima à qual detetavam robôs da sua cor, e implementada uma solução que consideramos relevante mencionar.

Na primeira situação, é simplesmente escolhido o robô mais próximo dos peixes como o robô a seguir. Na segunda, era preciso criar uma alternativa ao movimento de seguir os robôs, normalmente realizado pelos peixes. Assim, optou-se por aplicar um comportamento de "*wander*", isto é, atribuir aos peixes movimentos curtos em direções aleatórias constantemente a serem alteradas, mas limitados a um certo raio. Com este comportamento, os peixes parecem ficar a vaguear por uma certa área do cenário, até encontrarem um robô para seguir.

Um último aspeto do comportamento dos peixes é a forma como detetam e se aproximam das saídas. Quando detetam uma saída por onde podem passar nas suas proximidades, os peixes param por completo o seu movimento durante um segundo e só depois se direcionam, mais lentamente, para a saída. Esta paragem de um segundo serve para dar a ideia de que o peixe está a reconhecer a saída e preparar-se para mudar a sua direção no sentido desta, tornando o movimento para a saída muito mais natural.

Com todas as características discutidas acima, o movimento dos peixes ganhou fluidez e, por conseguinte, naturalidade. O movimento das abelhas, por sua vez, foi pensado em função da jogabilidade necessária, uma vez que havia a necessidade das abelhas se moverem favo a favo e tenderem a dirigir-se para a meta. No entanto, a aleatoriedade introduzida conferiu mais naturalidade ao movimento que, apesar de poder não ser totalmente realista, acaba por ser verosímil.

5. CONCLUSÃO

No que diz respeito ao público-alvo do jogo, este está a ser pensado para crianças e jovens, que geralmente têm mais experiência em jogos e na utilização de *smartphones* e *tablets*, os equipamentos em que este jogo estará disponível. Contudo, considera-se que o jogo tem potencial para agradar a um grupo muito mais alargado de utilizadores, ou pela complexidade que o jogo Bee pode providenciar ou pela simplicidade das regras do jogo Fish, que ainda assim consegue fornecer um grau elevado de desafio. Além disso, os adultos cada vez mais têm também experiência no uso de dispositivos móveis e cada vez mais jogam videojogos, principalmente os jogos casuais típicos desses mesmos dispositivos, nos quais o jogo BeeFish se enquadra.

Um dos públicos mais importantes para o jogo é o dos estudantes universitários, por serem jovens adultos que costumam jogar mas que podem também ganhar, graças ao jogo, interesse em saber mais sobre o projeto, como pretendido. Um estudo publicado em 2014 confirma que os jogos casuais/*puzzle* são muito jogados por estudantes do Ensino Superior [Carvalho14], o que fornece boas perspetivas para a aceitação do jogo BeeFish por esse público. Além disso, Yong e Gates [Yong14] concluem que há um aumento exponencial no uso de dispositivos móveis (*smartphones* e *tablets*), o que valida a escolha feita sobre o tipo de equipamentos para onde desenvolver o jogo.

O jogo BeeFish foi testado por 31 utilizadores, nomeadamente jovens estudantes do Ensino Secundário e Superior, entre os 15 e os 22 anos de idade. Os resultados foram medidos numa escala de Likert de 1 a 5, em que 1 significa "péssimo" e 5 "excelente".

No geral, os utilizadores consideraram o menu simples e de fácil utilização. Em termos de grafismo e jogabilidade, os resultados foram bastante satisfatórios tanto para o jogo Bee como para o jogo Fish, destacando-se a jogabilidade. Os resultados da apreciação global feita pelos utilizadores podem ser observados na Figura 4.

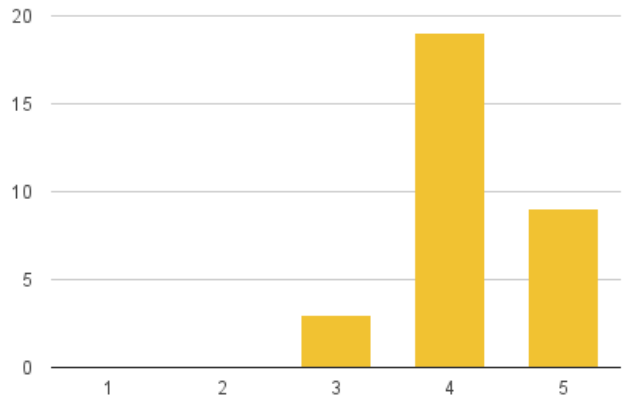

Figura 4: Apreciação global do jogo BeeFish.

Como trabalho futuro, pretende-se criar mais níveis no jogo, aproveitando os fatores de variação já existentes com a jogabilidade atual, mas também estender essa jogabilidade através da criação de novos obstáculos e dinâmicas de interação. Além disso, pretende-se criar uma ligação mais forte entre os níveis Bee e os níveis Fish, aproveitando o elemento que lhes é comum: a existência de CASU, que podem comunicar entre si.

Também se pretende, ao publicar o jogo, incorporar-lhe referências e informação sobre o projeto, para que o público tome conhecimento sobre os seus objetivos. Como visto acima, o jogo tem potencial para agradar a um público abrangente, por estar presente nas plataformas móveis usadas hoje em dia por grande parte da população, sendo uma forma lúdica de chamar a atenção para um projeto de investigação que pode ter aplicações reais num futuro não muito distante.

Concluindo, o jogo BeeFish cumpre o propósito de divulgação do projeto em que se insere, tendo obtido aprovação junto do público mais jovem. Os movimentos implementados para as abelhas e peixes são naturais e enquadram-se na jogabilidade definida, sendo que todo o jogo está pensado com base no projeto e nos comportamentos reais que os animais mostram nas experiências do mesmo.

6. AGRADECIMENTOS

Os autores agradecem o apoio do projeto ASSISIbf, EU-ICT nº 601074, e da unidade de I&D BioISI, UID/MULTI/04046/2013, financiada através da FCT/MCTES/PIDDAC.

7. REFERÊNCIAS

[ASSISIbf15] *Homepage – ASSISIbf*, http://assisi-project.eu/

[Carvalho14] A. A. Carvalho, I. Cardoso Araújo, *Jogos Digitais Que os Estudantes Portugueses Jogam: Diferenças de Género*, In IEEE 2014 9th Iberian Conference on Information Systems and Technologies (CISTI), 2014, pp. 1-6

[Reynolds87] C. W. Reynolds, *Flocks, herds, and schools: A distributed behavioral model*, Computer Graphics, 21(4), 1987, pp. 25-34

[Schikarski15] J. Schikarski, O. Meisch, S. Edenhofer, S. von Mammen, *The digital aquarist: An interactive ecology simulator*, in Proceedings of European Conference on Artificial Life, MIT Press, in press 2015

[Stencyl15] *Stencyl: Make iPhone, iPad, Android & Flash Games without code*, http://www.stencyl.com/

[Yong14] S. T. Yong, P. Gates, *Born Digital: Are They Really Digital Natives?*, International Journal of E-Education, E-Business, E-Management and E-Learning, vol 4, 2014, pp. 102-105