

# Construção de modelos baseados em agentes utilizando técnicas de inteligência artificial para jogos de computadores

Victor Teixeira de Melo Mayrink

*MAC5784 - Inteligência Artificial em Jogos de Computador*

---

## Resumo

Este trabalho apresenta uma metodologia para construir modelos baseados em agentes, em combinação com algumas técnicas de inteligência artificial que normalmente são utilizadas no contexto de jogos de computador. A metodologia proposta é ilustrada através da implementação de dois modelos distintos: o primeiro simula o fluxo de pessoas atravessando uma estação de metrô em sentidos opostos; o segundo modelo simula um jogo clássico em que dois times adversários competem para capturar a bandeira do oponente. Os aspectos práticos de implementação são discutidos em detalhes, assim como os resultados observados e as conclusões realizadas.

---

## 1. Introdução

A modelagem baseada em agentes é um paradigma para construção de modelos computacionais a partir de entidades autônomas, chamadas *agentes*, capazes de interagir entre si e com o ambiente onde estão inseridas.

Um dos primeiros e mais conhecidos modelos baseados em agentes foi publicado em 1971 por Thomas Schelling, ao demonstrar como grupos étnicos diferentes tendem a se segregar ao longo do tempo. O resultado observado foi surpreendente: se cada indivíduo (ou família) possuir uma preferência em residir próximo de pessoas do mesmo grupo étnico, então os diferentes grupos tendem a se segregar de forma muito expressiva ao longo do tempo, mesmo que a preferência por vizinhos do mesmo grupo seja muito pequena.

Apesar de bastante simples, o modelo de Schelling permitiu extrair conclusões interessantes sobre o comportamento social dos indivíduos e explicar porque é tão difícil combater o problema de segregação. Na prática, ao modelar sistemas a partir dos níveis hierárquicos mais básicos (por exemplo, o comportamento de um indivíduo), em geral é possível observar um certo nível de auto-organização. Muitas vezes, padrões e comportamentos que não foram explicitamente programados emergem através da interação entre os agentes [1].

A utilização de modelos baseados em agentes têm ganhado bastante interesse nas duas últimas décadas. As aplicações estendem-se em uma grande diversidade de pesquisas; podemos destacar, por exemplo, trabalhos relevantes na área de ecologia [2] e ciências sociais [3].

O escopo deste trabalho, no entanto, têm como objetivo combinar o paradigma de modelagem baseada em agentes com técnicas de inteligência artificial comumente empregadas no contexto de jogos de computadores.

Conforme menciona Millington and Funge [4], a modelagem baseada em agentes no contexto de jogos de computadores está relacionada à criação de personagens que são capazes de: *i*) capturar informações sobre estado corrente do jogo (senso-reamento); *ii*) analisar quais ações podem ser tomadas com base nas informações observadas; e *iii*) executar a ação mais adequada tendo em vista um objetivo a ser alcançado.

Para ilustrar a combinação destas duas técnicas, este trabalho descreve a modelagem de dois problemas diferentes. O primeiro modelo é uma simulação do fluxo de passageiros cruzando uma estação de metrô em sentidos opostos. Este modelo têm como objetivo construir um cenário que seja visualmente realista; sem, contudo, levar em consideração questões técnicas muito detalhadas, como por exemplo: o espaço médio ocupado por uma pessoa, a concentração máxima de pessoas por metro quadrado, ou ainda a função distribuição de probabilidades da velocidade em que as pessoas se movimentam nessas estações. Portanto, esse modelo simplificado da realidade, possivelmente não seria adequado para simular, por exemplo, uma situação de evacuação de uma estação em casos emergência. No entanto, tal modelo certamente seria útil para construir uma animação visualmente realista deste tipo de situação.

O segundo modelo, inserido no contexto dos jogos de computador, simula uma competição entre duas equipes adversárias que disputam para capturar a bandeira do oponente e levá-la até uma região previamente delimitada. Neste contexto, o objetivo é construir uma estratégia suficientemente inteligente para competir com um jogador humano que esteja no comando de uma das equipes. O principal requisito, neste caso, é que o comportamento da equipe que está sendo simulada responda coerentemente às ações que seu adversário executa ao longo da partida.

O presente trabalho foi dividido da seguinte maneira: a próxima seção descreve os objetivos gerais deste estudo. Em seguida, na seção 3, discutimos os principais conceitos do paradigma de modelagem baseada em agentes, incluindo alguns detalhes práticos de implementação. As seções 4 e 5 apresentam os modelos da estação de metrô e do jogo da bandeira, respectivamente. Finalmente, a seção 6 apresenta os resultados observados, e a seção 7 discute as conclusões do trabalho.

## 2. Objetivos

Os objetivos desse trabalho são:

1. Discutir os princípios da modelagem baseada em agentes;
2. Ilustrar a utilização do paradigma de modelagem baseada em agentes no contexto dos jogos de computador;
3. Demonstrar o processo de modelagem do comportamento individual dos agentes utilizando técnicas de inteligência artificial;
4. Analisar os comportamentos coletivos que emergem da interação entre vários agentes inseridos em um mesmo ambiente;
5. Apresentar como é possível extrair conclusões úteis sobre problemas realistas a partir de uma análise da simulação de um modelo computacional relativamente simples.

### 3. Modelagem baseada em agentes

A modelagem baseada em agentes é um paradigma centrado no comportamento individual dos elementos que fazem parte de um sistema. Esta perspectiva contrasta com os métodos tradicionais de modelagem, como por exemplo simulações numéricas baseadas em equações diferenciais, que descrevem o comportamento de um sistema seguindo uma abordagem holística.

Conforme com Macal and North [5], a construção de um modelo baseado em agentes envolve as seguintes etapas:

1. identificar quais agentes do sistema e uma teoria sobre como esses agentes se comportam;
2. identificar o relacionamento entre os agentes e uma teoria sobre como e quando as interações acontecem;
3. coletar dados sobre o comportamento real dos agentes e do sistema que está sendo modelado;
4. validar o comportamento dos agentes e do modelo como um todo, utilizando os dados obtidos;
5. executar a simulação do modelo e analisar os resultados observados, relacionando os comportamentos em micro e macro escala.

Evidentemente, dependendo do tipo de aplicação, nem todos os requisitos acima são absolutamente necessários para um modelo baseado em agentes. Por exemplo, no contexto de jogos de computadores, as etapas de coleta e validação de dados, apesar de útil, pode não ser necessária; basta que o modelo atenda o objetivo final do problema, que é construir um cenário fictício em que os personagens atuem de forma coerente com seus objetivos no jogo.

Contudo, se o problema em questão for a simular a evacuação de um shopping center em caso de incêndio; onde o objetivo é projetar e dimensionar as saídas de emergências do prédio, então é muito importante validar a teoria sobre o comportamento das pessoas em situações de pânico, garantir que o modelo criado seja o mais semelhante possível da realidade.

#### 3.1. Implementação computacional dos modelos

O modelo computacional de um agente deve possuir variáveis de estado, como por exemplo sua posição geográfica ou a velocidade que ele se movimenta. Dependendo do tipo de problema, algumas desses atributos podem ser variáveis ou fixos durante a simulação. Muitas vezes é importante garantir que a população de agentes tenha características heterogêneas, por exemplo agentes que se movimentam com velocidades diferentes, ou que tenham comportamentos mais agressivos doo que outros.

Da mesma forma, o ambiente também tem um estado que pode ser influenciado pelas ações do agentes, ou simplesmente pela sua própria presença, conforme discutiremos no caso dos agentes que possuem um mapa de influência no ambiente.

O comportamento dos agentes pode ser implementado por funções que podem alterar o estado de um ou mais agentes, ou do próprio ambiente.

Diante do exposto, do ponto de vista técnico é bastante natural adotar o paradigma de programação orientada à objetos para construir modelos baseados em

agentes. Desta forma podemos pensar em cada agente da simulação como sendo uma instância de uma classe, com atributos e métodos próprios.

Nos modelos baseados em agentes a simulação se passa em instantes discretos de tempo, denominados *ticks*, que correspondem às iterações da simulação. A cada passo de tempo, um ou mais agentes são *ativados*. Ao ser ativado, o agente pode ter a oportunidade de tomar uma ação, o que lhe permite alterar o seu próprio estado e interagir com outro agente e/ou com o ambiente.

No entanto, do ponto de vista computacional, não é possível ativar todos os agentes simultaneamente. Mesmo utilizando técnicas de computação paralela, em geral a quantidade de agentes de um modelo é maior que o número de núcleos de processamento da máquina. Além disso, podem haver partes do código que não são paralelizáveis. A ordem em que os agentes são ativados dentro de um mesmo *tick* é controlada por um componente do modelo chamado *scheduler*. Por exemplo, os agentes podem ser ativados em uma ordem aleatória, ou seguir sempre uma determinada ordem pré-definida.

Finalmente, ainda de acordo com [5], podemos resumir o processo de implementação de um modelo baseado em agentes nas seguintes etapas:

1. Agentes: identificar os tipos de agentes e outros objetos (classes), juntamente com seus atributos;
2. Ambiente: definir o ambiente em que os agentes irão viver e interagir;
3. Métodos dos Agentes: especificar um conjunto de métodos em que o estado dos agentes é atualizado em resposta às interações agente-agente e agente-ambiente;
4. Interações: especificar os métodos que controlam quais agentes interagem, quando interagem e como interagem;
5. Implementação: implementar o modelo de agentes em algum software ou linguagem de programação.

Os modelos desenvolvidos neste trabalho foram desenvolvidos na linguagem *python* utilizando a biblioteca *mesa* [6], que implementa um *framework* para facilitar a criação de modelos baseados em agentes.

#### 4. Fluxo de passageiros em uma estação de metrô

Este experimento simula o fluxo de passageiros em uma estação de metrô. O cenário criado possui apenas dois acessos, e o objetivo dos passageiros é atravessar a estação até a saída localizada no lado oposto. A figura 1 ilustra esse cenário em um determinado instante da simulação.

Nesta simulação existem 3 tipos diferentes de agentes:

- roletas (*gate*): representam os acessos da estação, sua função é inserir e/ou remover os passageiros no ambiente da simulação.
- passageiros (*walker*): são criados continuamente nas roletas e movimentam-se em direção à saída do lado oposto. Ao alcançarem a saída, são removidos da simulação.
- barreira (*wall*): é um agente auxiliar, sua única função é obstruir a movimentação dos passageiros.

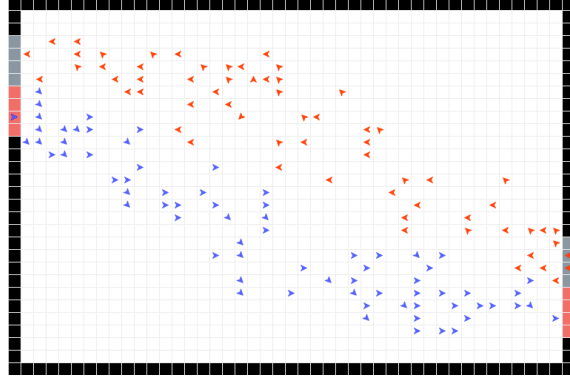


Figura 1: Simulação do fluxo de pessoas em uma estação de metrô

O cenário da estação de metrô foi construído com uma malha (*grid*) de  $45 \times 30$  células. Os *passageiros* podem movimentar-se somente sobre as *roletas* e células vazias que estejam dentro da vizinhança de sua posição atual (considerando uma vizinhança de *Moore* com raio unitário). Não é permitido que dois ou mais passageiros ocupem uma mesma célula simultaneamente.

A cada passo de tempo, os agentes do tipo *roleta* verificam se algum *passageiro* vindo da direção oposta chegou até a sua posição; em caso positivo, este agente é removido da simulação. Em seguida, a *roleta* tem uma probabilidade  $\rho$  de criar um novo passageiro que irá seguir em direção à saída oposta. Roletas localizadas no mesmo lado da estação possuem a mesma probabilidade de criar novos passageiros. Portanto, existem dois parâmetros,  $\rho_r$  e  $\rho_l$ , que controlam o fluxo de entrada de passageiros do lado direito e esquerdo da estação, respectivamente. Estes parâmetros podem ser ajustados durante a própria da simulação e variam dentro do intervalo  $[0; 0, 7]$ .

As roletas podem ser *bidirecionais* ou *unidirecionais*. Roletas bidirecionais podem inserir e remover os passageiros da simulação. Já as roletas unidirecionais podem ser do tipo *entrada* (apenas inserem novos passageiros), ou *saída* (apenas removem os passageiros existentes).

A cada passo da simulação, cada *passageiro*  $p$  tem uma probabilidade  $\sigma_p$  de lhe ser concedida a oportunidade de se mover. Conforme sugere a notação, esta probabilidade varia de indivíduo para indivíduo, o que permite criar *passageiros* que se movem com mais frequência que outros. Portanto, o atributo  $\sigma_p$ , associado ao passageiro  $p$  representa a velocidade com que esse passageiro se movimenta. Esse atributo é sorteado no momento em que o agente é criado pela roleta e permanece constante durante toda a simulação.

Quando um agente do tipo *passageiro* tem a oportunidade de se mover ele precisa decidir qual movimento deseja realizar. Esta decisão é implementada seguindo uma estratégia bastante simples: o passageiro verifica quais são todos os movimentos  $m_{xy}$  possíveis e então move-se para a célula que minimiza uma *função de avaliação estática* ([4], cap. 8) predefinida.

A função de avaliação utilizada é composta pela soma de duas componentes: *i*) o custo do movimento, e *ii*) a distância da nova posição até a roleta de saída mais próxima.

$$f(m_{xy}) = c(m_{xy}) + \min \left( \text{dist}((x, y), (px_i, py_i) |_{i=\text{gates}}) \right) \quad (1)$$

Onde  $f(m_{xy})$  é a função de avaliação estática para o movimento  $m_{x,y}$ ;  $c(m_{xy})$  é o custo do movimento  $m_{x,y}$ ; e  $dist((x, y), (px, py))$  representa a distância até do ponto  $(x, y)$  o portão localizado em  $(px, py)$ .

O custo do movimento é calculado usando a seguinte convenção: movimentos ortogonais têm custo unitário, enquanto que os deslocamentos na diagonal têm custo de  $\sqrt{2}$ . Esta estratégia evita que os passageiros fiquem fazendo movimentos diagonais e trocando de direção com muita frequência, o que causaria um comportamento anormal de zigue-zague.

$$c(m_{xy}) = \begin{cases} 1, & \text{para movimento ortogonal} \\ \sqrt{2}, & \text{para movimento diagonal} \end{cases} \quad (2)$$

A distância da nova posição até a saída mais próxima foi calculada segundo a *distância octal* [7], dada pela fórmula:

$$dist((x, y), (px, py)) = D_1 \times (dx + dy) + (D_2 - 2D_1) \times \min(dx, dy) \quad (3)$$

Onde  $dx = |x - px|$  e  $dy = |y - py|$ ; e  $D_1 = 1$  e  $D_2 = \sqrt{2}$ .

Portanto, função de avaliação estática expressa em 1 leva em consideração unicamente o objetivo individual de cada passageiro, o que tende a produzir agentes com comportamentos puramente egoístas. No entanto, conforme discutiremos nas seções 6 e 7, esta implementação foi suficiente para produzir um comportamento verossímil.

## 5. Competição de capturar a bandeira

Este experimento simula uma competição entre dois times adversários. O objetivo de cada equipe é capturar a bandeira do oponente e levá-la até a área de entrega, ao mesmo tempo em que deve defender-se dos ataques do adversário. A figura 2 ilustra a configuração inicial do jogo:

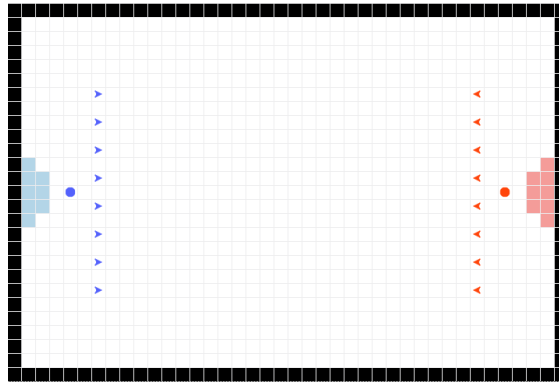


Figura 2: Configuração inicial do jogo

Para este jogo foram necessários 5 tipos diferentes de agentes:

- jogador (*player*): possui a capacidade de se mover; capturar e transportar a bandeira do time adversário; e disparar projéteis contra os oponentes.
- bandeira (*flag*): é um agente totalmente passivo. Uma vez capturada e entregue pelo time adversário, a bandeira reaparece na posição inicial.

- projétil (*fireshot*): é o agente criado quando um jogador realiza um disparo. O projétil movimenta-se em uma trajetória retilínea até atingir algum alvo ou percorrer uma distância máxima.
- zona de entrega (*delivery*): também foi modelada como um agente. Apesar de não poder se mover, a zona de entrega pode ser posicionada em qualquer ponto do *grid*. Sua única ação é recolher a bandeira do time adversário quando algum jogador a trazer até a sua localização.
- barreira (*wall*): assim como na da simulação anterior, a barreira é um agente totalmente inanimado. Sua única função é obstruir o movimento dos jogadores e funcionar como um anteparo para os projéteis.

O cenário da simulação representa um campo de batalha discretizado em uma malha de  $40 \times 27$  células. Cada time possui sua própria bandeira, oito jogadores e uma área de entrega que ocupa oito células (na prática, existe um agente do tipo *zona de entrega* em cada célula), conforme ilustrado na figura 2. A tabela abaixo apresenta a representação gráfica dos agentes:










Representação gráfica	Descrição
	Jogador
	Bandeira
	Zona de entrega
	Projétil
	Barreira
	Jogador transportando a bandeira adversária
	Jogador atingido por projétil
	Jogador imobilizado
	Colisão do projétil na barreira

Tabela 1: Representação gráfica dos agentes e seus estados.

Para manter o equilíbrio do jogo, foram estabelecidas as seguintes regras:

1. A posição inicial dos jogadores e das bandeiras é simétrica, de forma a não favorecer nenhuma das duas equipes
2. A cada passo da simulação cada jogador tem uma probabilidade de 50% de ser ativado, isto é, de ter a oportunidade de se mover ou fazer um disparo.
3. Ao ser ativado, o jogador deve escolher se deseja mover-se ou fazer um disparo (não é permitido fazer as duas ações ao mesmo tempo).
4. Os jogadores podem movimentar-se para qualquer uma das 8 células vizinhas (vizinhança de Moore), desde que elas não estejam ocupadas por algum outro jogador ou por uma barreira.
5. O jogador tem direito de mudar sua orientação antes de fazer um disparo. Um agente *projétil* será criado na célula em que sua orientação está apontando e irá seguir em linha reta de acordo com esta direção inicial.



6. O projétil se desloca uma célula por turno e, portanto, tem, em média, o dobro da velocidade de um jogador
7. O projétil é removido quando atinge um jogador, uma barreira, ou depois de percorrer uma distância máxima de 15 células.
8. O jogador que for atingido por um projétil fica imobilizado por 10 turnos.
9. Após o período de imobilização, o *jogador* atingido fica imune por 5 turnos.
10. Ao fazer um disparo o jogador fica 3 turnos sem poder disparar novamente (tempo de recarga). No entanto, durante esse período ele continua tendo oportunidade de se mover.

Os *jogadores* são os únicos agentes que utilizam técnicas de inteligência artificial mais elaboradas para a tomada de decisão. Os demais agentes, ou possuem um comportamento puramente determinístico, ou sequer são ativados durante a simulação.

As *bandeiras*, por exemplo, possuem uma lógica muito simples. Elas são ativadas em todos os turnos e sua única ação é verificar se a sua posição corrente está dentro da zona de entrega da equipe adversária; em caso positivo, ela reaparece na sua posição inicial e incrementa o contador de pontos do time oponente.

Os agentes do tipo *projétil* são criados quando um *jogador* realiza um disparo. Sua posição inicial será a célula vizinha que é apontada pela orientação do jogador que fez o disparo. Essa orientação é repassada para o projétil como um atributo que se mantém constante durante todo o seu ciclo de vida. Assim, a cada novo turno, os projéteis movem-se para a célula vizinha apontada pela orientação definida inicialmente (o que garante um movimento retilíneo). Ao atingir um agente do tipo *jogador*, o projétil altera seu estado para “imobilizado” (exceto se ele estiver em condição de imunidade); o que, segundo as regras do jogo, significa que este jogador não poderá ser ativado pelos próximos 10 turnos. Ao colidir com um agente do tipo *jogador* ou *barreira* o projétil modifica sua aparência gráfica (vide tabela 1) e é então removido no turno seguinte. Finalmente, toda vez que o projétil se movimenta ele incrementa uma de suas variáveis de estado que armazena a distância total percorrida; se esta variável for maior que o alcance máximo do projétil (definido em 15 células) ele é então removido da simulação.

As *zonas de entrega* e as *barreiras*, apesar de terem sido implementadas como agentes para facilitar a modelagem do problema, não possuem nenhum comportamento ativo durante a simulação. Portanto esses agentes sequer foram adicionados ao *scheduler* do modelo. A tabela 2 resume a como funciona a ativação de cada tipo de agente.

Agente	Ativação
Jogador	Probabilística (50%)
Bandeiras	Todos os turnos
Projétil	Todos os turnos
Zona de entrega	Não é ativado (não tem ação)
Barreira	Não é ativado (não tem ação)

Tabela 2: Ativação dos agentes no *scheduler* do modelo.

Cada equipe possui 8 jogadores que podem assumir uma postura ofensiva (atacante) ou defensiva (defensor). A quantidade de atacantes e defensores da equipe



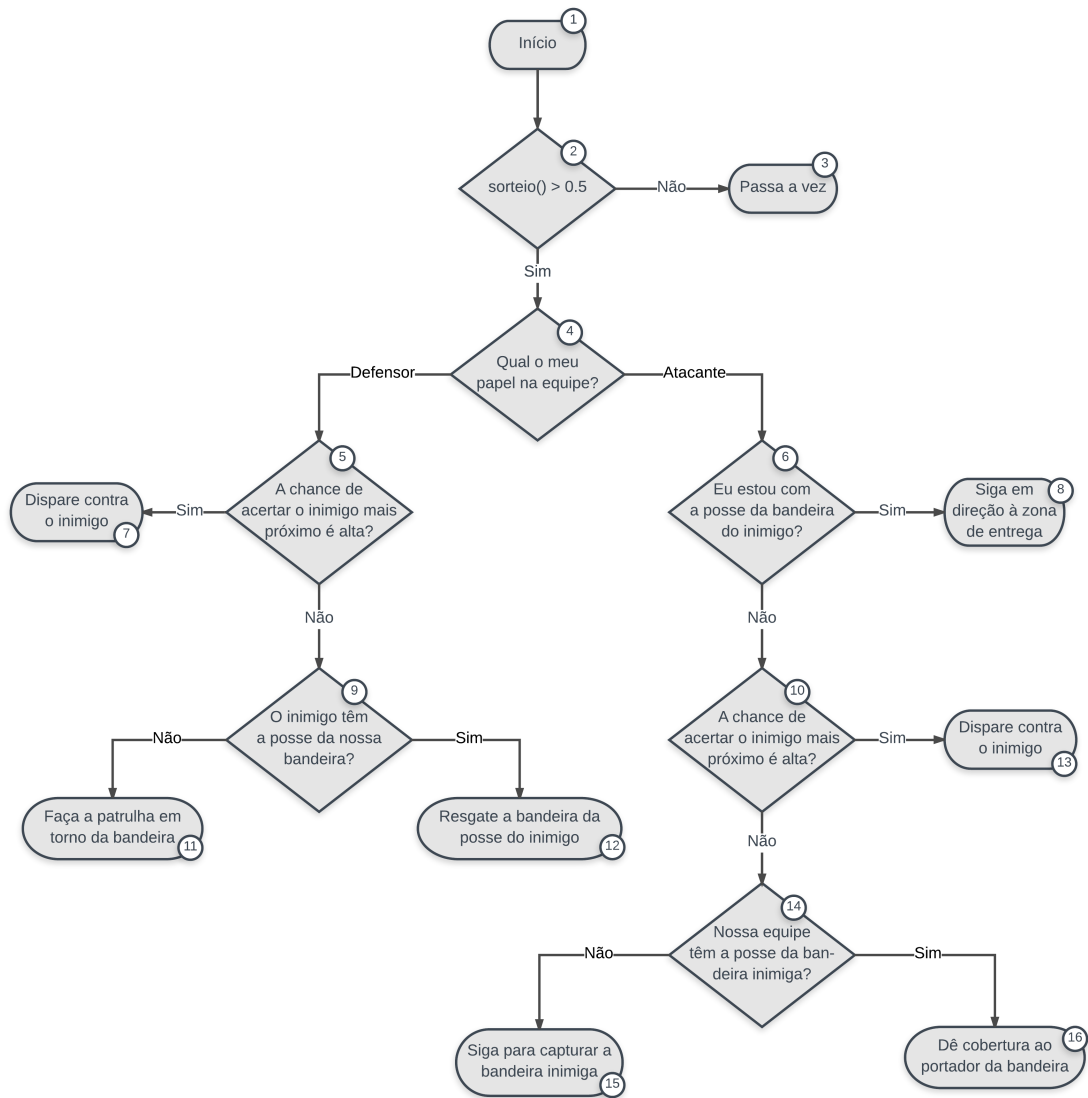


Figura 3: Árvore de decisão dos jogadores.

é um parâmetro do modelo que pode ser alterado durante a simulação da partida. Essa configuração define o *esquema tático* do time e o comportamento global de cada jogador. A figura 3 representa a árvore de decisões executada quando o jogador é ativado; os números em cada nó da árvore foram adicionados para facilitar a identificação dos comportamentos e dos processos de tomada de decisão.

Evidentemente, a velocidade dos jogadores deve ser inferior à dos projéteis, caso contrário seria muito improvável acertar um disparo sobre o oponente pois ele poderia esquivar-se com muita facilidade. Além disso, não faz sentido um jogador mover-se na mesma velocidade de um projétil. A condição de ativação definida no nó 2 da árvore cria uma condição probabilística para a ativação do jogador; isso possui dois efeitos principais: (1) reduz a velocidade dos jogadores, e (2) cada jogador não sabe quando será sua próxima oportunidade de se mover.

A decisão do nó 4 divide o comportamento dos jogadores de ataque e defesa; esta não é propriamente uma decisão do jogador, mas sim um parâmetro do modelo. Obviamente seria possível fazer com que os jogadores pudessem avaliar a situação do jogo e decidir se devem tomar uma postura ofensiva ou defensiva. Essa decisão po-

deria ser tomada com base, por exemplo, na distância do jogadores até as bandeiras e na postura dos demais jogadores. No entanto, optamos por manter essa decisão como um parâmetro da simulação; na próxima seção apresentamos o desempenho das equipes em relação ao esquema tático escolhido.

Os jogadores defensivos podem optar por três comportamentos distintos, conforme ilustra a árvore da figura 3. Primeiramente, o jogador observa se existe algum jogador inimigo nas proximidades, fato que representa uma ameaça à bandeira ou a sua própria integridade. Então, supondo que a arma do jogador está carregada, ele estima qual é chance de acertar oponente se fizer um disparo; se essa chance superar um determinado limite pré-definido então ele faz o disparo contra o oponente, caso contrário ele segue para o próximo nó de decisão.

Se a chance de sucesso do disparo for baixa, então o jogador defensivo irá observar se o time adversário está com a posse da bandeira de sua equipe. Em caso positivo ele irá seguir e em direção a bandeira para tentar resgatá-la do inimigo, caso contrário, o jogador irá fazer uma patrulha no entorno da bandeira.

Já comportamento dos jogadores de ataque funciona da seguinte maneira: primeiramente, se o jogador já está com a posse da bandeira adversária, então ele deve seguir para a zona de entrega tentando esquivar-se dos inimigos e projéteis. Caso contrário ele, segue a mesma lógica dos jogadores de defesa para verificar se existe algum inimigo próximo que represente uma ameaça e que possa ser atingido por um disparo. Se chance de sucesso do disparo for alta ele lança o projétil, caso contrário ele observa quem está com a posse da bandeira do adversário. Se for alguém da sua própria equipe, ele deve dar cobertura ao portador da bandeira; mas se a bandeira não está na posse da sua equipe ele deve tentar capturá-la.

A árvore de decisão da figura 3 descreve o comportamento dos jogadores em nível de abstração bastante elevado. Por exemplo ao decidir fazer um disparo, é necessário verificar se chance de acertar o adversário é suficientemente alta. Para isso é necessário saber em qual direção atirar e como calcular a chance de sucesso. Esta decisão foi implementada usando o conceito de *mapas de influência* (ver [4]). Cada jogador possui seu próprio campo de influência, que modifica o mapa de influência do ambiente; da mesma forma os projéteis também possuem um campo de influência dentro do ambiente. A decisão de fazer o disparo é implementada através da soma do produto ponto-a-ponto do campo de influência do projétil a ser criado sobre o mapa de influência dos jogadores (jogadores aliados têm uma influência negativa e jogadores inimigos têm influência positiva). Se este somatório for alto significa que o campo de influência do projétil cruza com o campo de influência dos jogadores adversários, e, portanto, o disparo tem uma chance razoável de acertar o alvo. Por outro lado, valores negativos indicam que o disparo pode acertar jogadores aliados.

O campo de influência dos jogadores e dos projéteis, assim como o valor limiar para decidir efetuar o disparo, foram definidos após alguns testes empíricos. Os campos de influência dos projéteis e jogadores podem ser consultados no arquivo `/projects/flag/influence_masks.py` do repositório deste projeto.

Além dos mapas de influência, utilizamos também o algoritmo  $A^*$  para traçar a rota dos jogadores ao tentar capturar e entregar a bandeira do adversário. O custo dos movimentos no algoritmo  $A^*$  também leva em consideração os mapas de influência dos jogadores e projéteis, para evitar rotas que: (1) cruzem com o trajeto de um projétil (evita o risco de ser atingido), (2) passem muito próximo de um

inimigo -evita que o inimigo faça um disparo à *queima roupa*), ou (3) passem muito próximo de outro jogador aliado (evita ataques concentrados, o que, em geral, tende a ser uma estratégia ruim).

## 6. Resultados

Nesta seção apresentamos alguns resultados interessantes sobre os modelos apresentados neste trabalho.

### 6.1. Regiões da estação com maior circulação de pessoas

Em uma situação real, pode ser interessante identificar quais são as regiões da estação que apresentam a maior circulação de pessoas, por exemplo, para evitar que essas áreas sejam obstruídas ou ainda para posicionar e precificar anúncios publicitários.

A figura 4 representa um mapa de calor da estação, indicando as regiões com maior circulação de passageiros. Cada cor indica pessoas caminhando em um determinado sentido, tons mais escuros representam as regiões com maior circulação de passageiros.

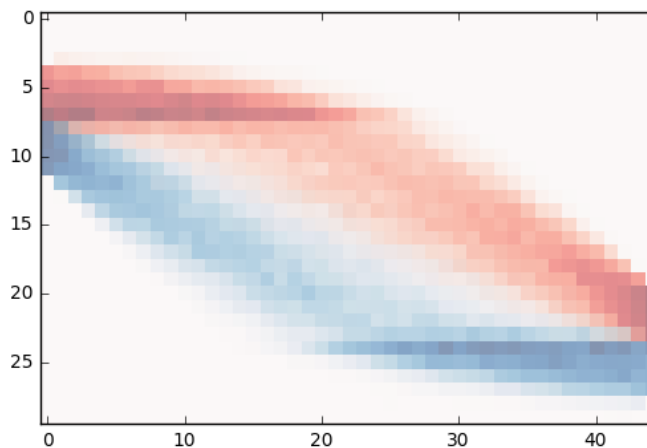


Figura 4: Mapa de calor das regiões com maior circulação de passageiros.

Observando a figura 4, é possível verificar que as pessoas começam a se dispersar após cruzarem a roleta de entrada e voltam a convergir ao se aproximarem das roletas de saída. Este é um comportamento bastante elementar, mas que não foi explicitamente programado na construção do modelo.

### 6.2. Tempo e distância percorrida para atravessar a estação

Outra característica importante para ser analisada no modelo de fluxo de passageiros é o tempo médio que as pessoas levam para atravessar a estação. Intuitivamente, podemos supor que quanto maior a circulação de pessoas na estação, a movimentação fica mais restrita e, portanto, maior deve ser o tempo de travessia.

Como os agentes possuem velocidades muito diferentes uns dos outros, o tempo médio de travessia pode ter uma variação muito grande. Portanto, ao invés de contabilizar o tempo total que o agente levou para atravessar a estação, optamos

por medir: (1) a distância total percorrida pelo agente, e (2) o número de vezes que o agente foi ativado. O primeiro indicador considera a distância euclidiana entre duas células vizinhas, penalizando os trajetos tortuosos; já a segunda medida, penaliza os casos em que o agente teve a oportunidade de se mover, mas dada as circunstâncias, preferiu continuar na mesma posição.

As figuras 5 e 6 demonstram como variam a distância percorrida pelos agentes e o número de vezes que eles são ativados em função do fluxo de pessoas nos acessos de entrada e saída. Estes gráficos foram gerados, mantendo o fluxo de passageiros constante por 1000 passos de tempo para cada simulação. Para facilitar a visualização limitamos o intervalo das ordenadas nas duas figuras, o que ocultou alguns *outliers* mais extremos.

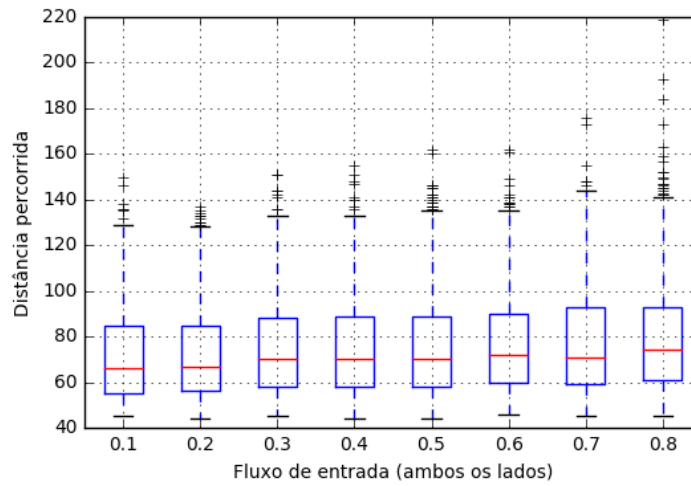


Figura 5: Distância percorrida em função do fluxo de passageiros.

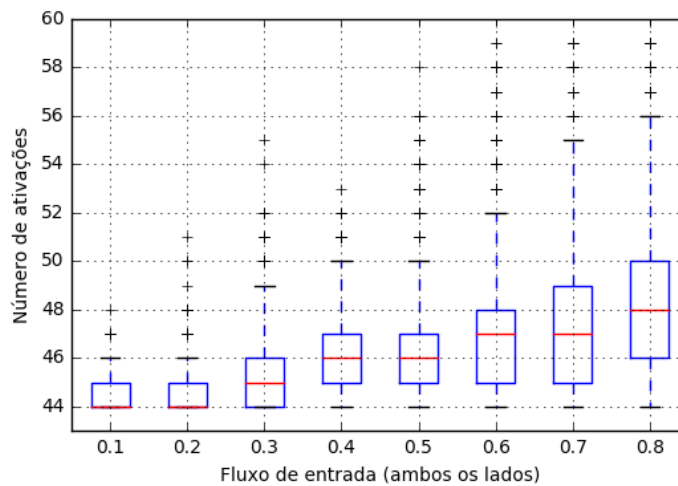


Figura 6: Número de ativações em função do fluxo de passageiros.

Conforme o esperado, os gráficos das figuras 5 e 6 confirmam que tanto a distância percorrida, quanto o tempo de travessia aumentam à medida que o fluxo de passageiros cresce. Um fato interessante é que o aumento das medianas é relativamente lento,

mas a variância da distribuição cresce bastante à medida que o fluxo de passageiros aumenta.

### 6.3. Esquema tático das equipes

O esquema tático de cada equipe, isto é, a proporção entre número de atacantes e defensores do time são parâmetros da simulação. A tabela 3 apresenta o resultado de uma partida de 5000 passos de tempo para todas combinações possíveis de esquemas táticos entre as duas equipes.

A primeira linha indica o esquema tático da equipe vermelha, enquanto que a primeira coluna indica o esquema tático da equipe azul. Utilizamos cores distintas para facilitar a visualização. A notação  $3A \times 5D$  indica que time azul joga com 3 atacantes ( $3A$ ) e 5 defensores ( $5D$ ). As células representam a pontuação de cada equipe ao final da partida.

Esquema tático	$1A \times 7D$	$2A \times 6D$	$3A \times 5D$	$4A \times 4D$	$5A \times 3D$	$6A \times 2D$	$7A \times 1D$
$1A \times 7D$	$10 \times 10$	$11 \times 18$	$09 \times 26$	$11 \times 27$	$15 \times 25$	$15 \times 29$	$09 \times 15$
$2A \times 6D$	$17 \times 11$	$16 \times 20$	$17 \times 22$	$20 \times 25$	$22 \times 26$	$24 \times 32$	$24 \times 30$
$3A \times 5D$	$18 \times 12$	$21 \times 17$	$22 \times 24$	$22 \times 29$	$24 \times 28$	$29 \times 31$	$32 \times 37$
$4A \times 4D$	$22 \times 13$	$24 \times 20$	$23 \times 22$	$26 \times 29$	$30 \times 34$	$28 \times 32$	$32 \times 35$
$5A \times 3D$	$24 \times 15$	$20 \times 22$	$27 \times 30$	$28 \times 30$	$29 \times 32$	$34 \times 36$	$34 \times 35$
$6A \times 2D$	$25 \times 19$	$26 \times 23$	$31 \times 30$	$31 \times 33$	$31 \times 35$	$33 \times 36$	$37 \times 37$
$7A \times 1D$	$13 \times 13$	$27 \times 28$	$29 \times 30$	$33 \times 35$	$35 \times 38$	$33 \times 39$	$35 \times 37$

Tabela 3: Representação gráfica dos agentes e seus estados.

Naturalmente podemos observar que as pontuações das equipes é relativamente baixa quando ambas adotam um esquema tático muito defensivo. Compare, por exemplo, a média das pontuações das equipes nas submatrizes  $3 \times 3$  no canto superior esquerdo (esquemas táticos defensivos) e no canto inferior direito (esquemas táticos ofensivos).

De modo geral, os pontos marcados por uma equipe tende a aumentar quando colocamos mais atacantes. No entanto, uma exceção interessante acontece quando uma equipe está com 7 atacantes e a outra com 7 defensores; nos dois casos simulados, a pontuação marcada pela equipe com 7 atacantes foi bem inferior ao esquema tático com 6 atacantes. Durante a simulação percebemos que isso ocorre pois o número de jogadores (aliados e adversários) cercando a bandeira é muito grande, obstruindo o caminho do jogador que está transportando a bandeira.

Conforme discutimos na seção 3 a modelagem baseada em agentes é essencialmente estocástica. Portanto, para obter resultados significativos é importante realizar várias simulações e observar o comportamento médio global do sistema à medida que os parâmetros são modificados. No nosso caso, haviam duas possibilidades de tentar neutralizar o efeito aleatório das simulações: (1) simular um número muito grande de partidas, ou (2) simular uma partida por um tempo muito prolongado. Obviamente, o ideal é combinar as duas abordagens para neutralizar ao máximo as influências aleatórias, entretanto, por restrições de tempo e custo computacional, optamos por seguir a segunda abordagem.

## 7. Conclusão

Neste trabalho apresentamos como o conceito de modelagem baseada em agentes pode ser aplicado no contexto de jogos de computador. O comportamento dos agentes foi modelado utilizando-se técnicas de inteligência artificial comumente empregadas em na construção de jogos: árvores de decisão, mapas de influência, função de avaliação estática, e o algoritmo  $A^*$ .

Para ilustrar a utilização desses conceitos, apresentamos dois modelos distintos: a simulação do fluxo de passageiros em uma estação de metrô e o jogo de competição pela bandeira adversária.

O primeiro caso envolve um modelo bastante simples em que os passageiros se comportam de forma puramente egoísta. A tomada de decisão dos passageiros resume-se à minimizar uma função de avaliação que considera apenas o custo do movimento e a distância até a saída mais próxima. Apesar da simplicidade do modelo, os resultados em escala macro levaram à um comportamento bastante verossímil; o suficiente, por exemplo, para produzir uma animação para um filme ou jogo de computador.

Por outro lado, o modelo de competição pela bandeira exige que os agentes se comportem seguindo uma estratégia bem mais elaborada. Os jogadores da mesma equipe precisam atuar de forma colaborativa entre si e competitiva com os times da outra equipe. Estes objetivos produzem uma diversidade de comportamentos, que inclui ações como: (1) guardar sua própria bandeira, (2) resgatar sua bandeira da posse do adversário, (3) atacar a bandeira adversária, (4) carregar a bandeira do adversário até a área de entrega, ou (5) dar cobertura ao jogador que está carregando a bandeira.

Uma vez que o jogador decide qual comportamento ele deve tomar, é necessário decidir também para qual direção ele deve se mover (ou em qual direção deve disparar), de forma a atingir os objetivos pretendidos com o comportamento escolhido. Essas decisões são feitas com base em uma análise sobre o estado atual do ambiente e dos agentes, tanto aliados como adversários, o que torna o processo de decisão bastante complexo.

No entanto, ainda é possível aprimorar os modelos propostos em vários aspectos. Algumas propostas de trabalhos futuros incluem: (1) associar grandezas físicas usando dados realistas ao modelo de fluxo de passageiros, considerando, por exemplo, a área média ocupada por uma pessoa, as dimensões da estação, e a velocidade das pessoas; (2) rever as regras gerais do jogo de capturar a bandeira, de tal forma que o resultado das equipes fique mais sensível à escolha de um esquema tático e da estratégia de jogo dos agentes; (3) implementar uma lógica para que o jogador possa alternar entre atacante e defensor dependendo de sua posição e da situação do jogo; (4) implementar comportamentos mais elaborados para os jogadores, levando em consideração previsões sobre as ações mais prováveis do adversário.

## Referências

- [1] Charles M Macal and Michael J North. Tutorial on agent-based modelling and simulation. *Journal of simulation*, 4(3):151–162, 2010.
- [2] Volker Grimm, Uta Berger, Finn Bastiansen, Sigrunn Eliassen, Vincent Ginot, Jarl Giske, John Goss-Custard, Tamara Grand, Simone K Heinz, Geir Huse,

- et al. A standard protocol for describing individual-based and agent-based models. *Ecological modelling*, 198(1):115–126, 2006.
- [3] Eric Bonabeau. Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences*, 99(suppl 3): 7280–7287, 2002.
  - [4] Ian Millington and John Funge. *Artificial intelligence for games*. CRC Press, 2016.
  - [5] Charles M Macal and Michael J North. Tutorial on agent-based modeling and simulation. In *Simulation Conference, 2005 Proceedings of the Winter*, pages 14–pp. IEEE, 2005.
  - [6] Mesa: Agent-based modeling in python 3+. URL <https://github.com/projectmesa/mesa>.
  - [7] Amit’s thoughts on pathfinding. URL <http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html>.