Ambiente para Desenvolvimento de Inteligência Artificial em Jogos Allegro

Autor: Arthur Phillip Silva [1] **Orientador: Pedro Olmo Stancioli Vaz De Melo** [1]

Abstract

Inteligência Artificial (IA) é um dos objetos de estudo mais pesquisados da atualidade por seu amplo espectro de utilização. Um ramo de estudo de IA é sua aplicação em jogos, na construção de agentes racionais capazes de aprender e jogar autonomamente como é o caso do *Alfa Go* da *DeepMind* que foi capaz de derrotar o campeão mundial de Go.
Para incentivar e viabilizar o estudo de IA em jogos foram desenvolvidos *frameworks* e bibliotecas que auxiliam na integração do agente e o jogo. Estas soluções se dedicam a criar uma conexão com o jogo através de emuladores dos consoles permitindo que o agente simule um humano jogando videogame.
Neste trabalho se propõe um ambiente em que o pesquisador não estará limitado a um jogo existente, mas poderá usar qualquer jogo que ele tenha acesso ao código fonte e feito em Allegro. O que traz uma possibilidade do pesquisador ter controle não só do agente como também do seu ambiente de atuação, que é o jogo. Acessando o código fonte do jogo e provendo um canal e interação com o agente, nossa proposta é aumentar a autonomia e flexibilidade da pesquisa e do pesquisador.

Palavras-chave: Allegro, Jogos Digitais, Inteligencia Artificial

1. Introdução

Uma grande busca das pesquisas em computação é o desenvolvimento de soluções adaptativas que não precisem de uma explicita definição de todas as possíveis situações para que se tenha uma ampla cobertura de domínio. Esta é uma das expectativas ao se estudar IA. O estudo de IA tem como objetivo o desenvolvimento de algoritmos capazes de competência geral em uma variedade de tarefas e domínios sem a necessidade de explicitar das características específicas do domínio. (Bellemare et al., 2013)

Segundo Yannakakis & Togelius, o campo da inteligência artificial e computacional em jogos não só tem visto grandes avanços como tem em seu currículo várias histórias de sucesso. Em seu livro (Yannakakis & Togelius, 2018) apresenta a longa história que jogos e inteligência artificial têm juntos. como também as pesquisas sobre IA para jogos e como estão relacionadas à construção de agentes que joguem, com ou sem aprendizado prévio. Em suas palavras.

> ”Mesmo antes de a inteligência artificial ser reconhecida como um campo, os primeiros pioneiros da ciência da computação escreveram programas de jogo porque queriam testar se os computadores poderiam resolver tarefas que pareciam exigir inteligência”.

Segundo Russell, o conceito de agência racional é considerado um dos principais candidatos a cumprir o papel de uma inteligência, os atributos para isso são ser precisa, permitir o desenvolvimento cumulativo de sistemas robustos e ser generalizável. O estudo de IA aplicado a jogos não só é antigo como se especializou em áreas específicas que abrangem grupos de jogos ou mesmo funções dentro do jogo. Outro trabalho de Yannakakis & Togelius mostra como essas áreas podem se beneficiar do conhecimento criado em outras áreas e fazer sua própria pesquisa mais relevante para as outras áreas.

Porém, as pesquisas de IA em jogos sofrem com o fato de além de se preocupar com o desenvolvimento da IA, devem obter uma solução a outro problema: como as decisões do agente racional serão aplicadas no jogo, ou seja, como a IA vai jogar de fato. Para viabilizar os estudos nessa área, algumas soluções foram propostas, *Arcade Learning Environment* (ALE) (Bellemare et al., 2013), por exemplo, utiliza um emulador de Atari para interfacear a interação entre os agentes racionais e os jogos desse console. *General Video Game Playing* (GVGP) (Perez-Liebana et al., 2018)

[1]Departmento de Ciência da Computação (DCC) da Universidade Federal de Minas Gerais (UFMG), Belo horizonte, Brasil. e-mail: Arthur Silva <artphil@dcc.ufmg.br >, Pedro Olmo <olmo@dcc.ufmg.br >.

expande a proposta anterior com uma plataforma de código aberto que visa criar controladores para reprodução geral de videogames, possibilitando e incentivando a pesquisa em qualquer jogo de videogame. Este, porém, sendo limitado aos jogos existentes e aos consoles que a plataforma consegue emular.

Neste trabalho se propõe um ambiente (*framework*) em que o pesquisador não estará limitado a um jogo existente, mas poderá usar qualquer jogo que ele tenha acesso ao código fonte e feito em Allegro, uma biblioteca de código aberto para desenvolvimento de jogos em C/C++, simples e robusta. Ela fornece rotinas de baixo nível normalmente necessárias na programação de jogos, como entrada, gráficos, midi, efeitos sonoros e temporização. (Leverton, acesso em 2019) O acesso ao código fonte proporciona ao pesquisador maior controle sobre o ambiente de aprendizado dos agentes.

(a) Frogger (b) Guittar hero (c) Osmos

Figure 1. Telas dos jogos desenvolvidos em Allegro.

O principal objetivo desse trabalho é propiciar um ambiente facilitador ao estudo de soluções de IA aplicada em jogos. Uma plataforma que permita que o pesquisador possa dedicar sua atenção e tempo as estratégias de decisão sem se preocupar na integração do sistema com o(s) jogo(s) escolhidos.

2. Referencial Teórico

Existem algumas soluções bem famosas de inteligência artificial que conseguem derrotar campeões mundiais de alguns jogos. Até então não se pensava que uma máquina conseguiria fazer estratégias tão sofisticadas em jogos de raciocínio lógico. *Alfa Go* (Silver et al., 2016) por exemplo, é um algoritmo que utiliza redes neurais para jogar Go, um jogo milenar, muito famoso na China, e derrotou o campeão mundial Lee Sedol em 2016. *Alfa zero* (Silver et al., 2018), o sucessor de *Alfa Go*, através de uma rede muito mais ampla consegue jogar Xadrez e Shogi além do original Go. *DeepMind*, a empresa criadora dos dois anteriores, desenvolveu recentemente o *Alfa Star* (Vinyals et al., 2019), lançado no inicio de 2019 num confronto com finalistas do mundial do jogo *Star Craft* e demonstrou uma grande performance vencendo a maioria das partidas.

ALE é uma estrutura de software para interface com ambientes de jogos Atari 2600 emulados. Ele permite que o usuário faça interface com o Atari 2600 recebendo movimentos do joystick, enviando informações de tela e/ou RAM e emulando a plataforma. O ALE também fornece uma camada de manipulação de jogos que transforma cada jogo em um problema padrão de aprendizado de reforço, identificando a pontuação acumulada e se o jogo terminou (Bellemare et al., 2013).

A competição da *General Video Game AI* (GVGAI) foi fundada na crença de que a melhor maneira de impedir que os pesquisadores de IA dependam de engenharia específica de jogo em seus agentes. Os pesquisadores desenvolveriam seus agentes sem saber que jogos iriam jogar e, depois de submeter seus agentes à competição, todos os agentes seriam avaliados usando um conjunto de jogos invisíveis (Perez-Liebana et al., 2018). GVGP é uma proposta, também chamada de GVGAI *framework*, que possibilita o controle dos jogos, fazendo a interface entre o agente e diversos jogos de vários consoles. Um *framework* que padroniza o controle dos jogos para a criação dos agentes racionais que participarão da competição GVGAI.

3. Metodologia

O objeto final desenvolvido aqui é uma ferramenta, algo que possa auxiliar outros desenvolvedores a realizarem seus projetos de aplicação de inteligência artificial em jogos.
A ferramenta a ser desenvolvida foi dividida em três atividades fundamentais que correspondem aos três módulos principais, Identificador, Atuador e Perceptor. O Identificador é a parte responsável por fornecer quais são as ações possíveis de atuação do agente sobre o jogo, o Atuador é aquele que transforma as decisões do agente em ações de fato no jogo e o Perceptor tem a função de captar o estado do jogo e fornecer para o agente informação necessária para sua tomada de decisão. Figura 2

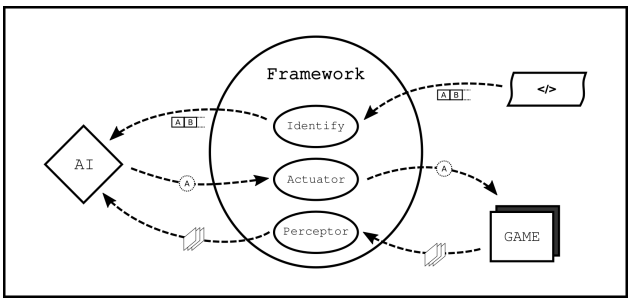

Figure 2. Fluxo de dados
1. O *framework* lê código e fornece informações ao agente.
2. O *framework* inicia o jogo.
3. O Agente envia comandos ao *framework*.
4. O *framework* aplica no jogo os comandos recebidos.
5. O resultado é registrado e enviado ao agente.
6. Enquanto o jogo não termina volta ao passo 3.

Identificador

O desafio nesta parte foi descobrir como retirar do código fonte os dados necessários para que o agente pudesse interferir no jogo. As funções e variáveis da biblioteca Allegro foram usadas como marcadores. Isto facilitou a obtenção dos comandos necessários para manipulação do jogo, pois foi a partir disso que se pensou em fazer uma busca por esses marcadores e retirar dali as informações necessárias.

Principais marcadores:
Allegro_KEY_X representa o uso do teclado no jogo onde X é a tecla a ser pressionada.
mouse indica a utilização do mouse no jogo.

Sobre o texto do código fonte, foram feitas buscas por reconhecimento de caracteres e são recuperados os comandos possíveis a serem executados durante o jogo.

Atuador

Deparou-se com um obstáculo ao tentar manipular os periféricos, os sistemas operacionais têm APIs muito complicadas de utilizar e há pouca documentação sobre isso. A solução foi a utilizar uma biblioteca de python chamada PyAutoGui, através dela podemos ter acesso as APIs dos principais sistemas operacionais e enviar sinais a ele simulando o sinal enviado por um periférico (teclado/mouse).

Para essa biblioteca funcionar a tela do jogo precisa ser a janela ativa, ou seja a primeira janela, a que está por cima de todas as outras. Para o controle da janela utilizou-se a biblioteca PyGetWindow, uma biblioteca cuja proposta é fornecer controle multiplataforma das janelas do sistema, porém no momento só tem suporte a' Windows.

Esta parte é responsável por traduzir as decisões da IA para ações no jogo, para isso foram usadas as bibliotecas Pyautogui e Pygetwindow. Pyautogui é uma biblioteca multiplataforma que acessa as APIs de sistema para enviar comandos de teclado e mouse, simulando os periféricos. Através da Pygetwindow é possível identificar e manipular as janelas de todos os programas abertos, porém, só funciona em Windows. O uso dessas bibliotecas permite identificar e manipular o jogo como se fosse um usuário humano, assim o Atuador recebe as decisões do agente e executa a ação através do mouse ou teclado dentro do escopo da tela pertencente ao jogo.

Perceptor

Para que o agente possa tomar uma boa decisão, esta precisa ter ciência do atual estado do jogo, assim como qual o impacto de sua ultima ação sobre ele. A solução mais prática, e a adotada neste trabalho, é fornecer para o agente uma imagem da tela do jogo.

O problema encontrado foi o ruido gerado pelo fundo da área de trabalho e as janelas abertas junto ao jogo, como também a variedade de formas e tamanhos de monitores e como a tela do jogo se adéqua a ela. Para sua atuação esse módulo utiliza a Pygetwindow para recuperar uma imagem da tela do jogo após cada comando ou um intervalo de tempo pré-definido.

A biblioteca utilizada, além de ativar, maximizar e mover a janela, fornece dados desta, como nome tamanho e posição. Estes dois últimos foram os utilizados para determinar a área da tela a ser registrada e fornecida a AI, para que possa ser tratada e utilizada no processo de tomada de decisão.

Jogos

Para testar o funcionamento da interface foram utilizados três jogos desenvolvidos de forma simplificada em Allegro, Frogger, Guittar Hero e Osmos.

Frogger é um jogo onde um sapo tem que atravessar uma rodovia de várias pistas. O sapo tem movimento em todas as direções enquanto os veículos caminham em linha reta da esquerda para a direita ou vice-versa. O objetivo é chegar do outro lado no menor tempo possível e sem ser atropelado, caso contrário ele pode morrer ou perder uma vida, dependendo da implementação. Na nossa simplificação o sapo é representado por um quadrado de vidas infinitas e os veículos por retângulos coloridos que andam apenas da esquerda para a direita. Figura 3

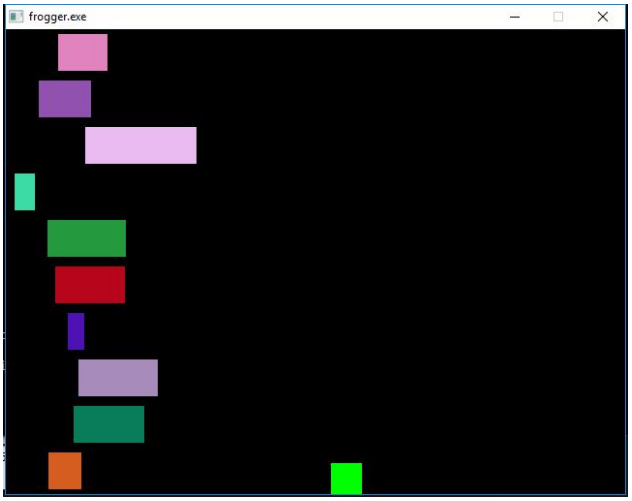

Figure 3. Frogger desenvolvido em Allegro.
Sapo: quadrado verde no centro inferior da tela
Veículos: demais retângulos coloridos

Guittar Hero é um jogo de coordenação motora, onde existem posições na tela relacionadas a botões controlados pelo jogador. Aparecem na tela símbolos que caminham

para uma dessas posições, normalmente a frequência de aparições está relacionada ao ritmo de uma música que é tocada durante o jogo. O modo do jogo é apertar o botão quando o símbolo estiver exatamente dentro da posição relativa a ele. Cada vez que isso acontece acrescenta-se certa pontuação ao jogador, caso o símbolo passe da posição sem que o botão seja acionado ou haja o acionamento do botão sem que haja algum símbolo na posição, o jogador perde pontos. O objetivo é fazer a maior pontuação dado um intervalo de tempo ou uma quantidade fixa de símbolos. Na nossa implementação foram usadas apenas três botões/posições e os símbolos aparecem aleatoriamente, sem relação com uma música específica. Figura 4

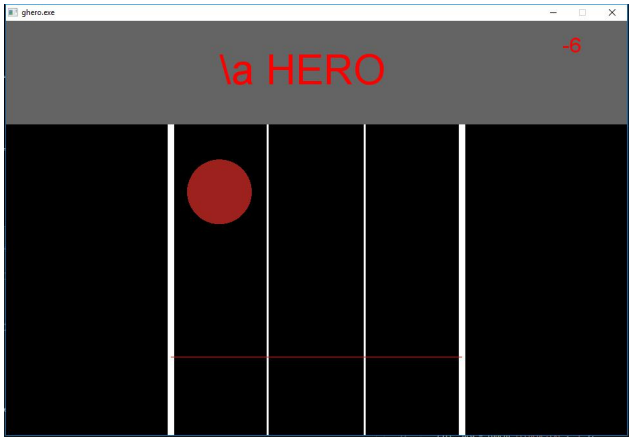

Figure 4. Guitar Hero desenvolvido em Allegro.
O jogador deve apertar os botões 1, 2 ou 3 quando os círculos vermelhos entrarem completamente no primeiro, segundo ou terceiro quadrado, respectivamente, que estão no centro inferior da tela.

Osmos, diferente dos anteriores, não utiliza botões do teclado, mas o jogador utiliza o mouse para interagir com o jogo. Na sua configuração inicial existem vários objetos dispersos se deslocando na tela, normalmente circulares e com tamanhos e velocidades diferentes. Quando dois desses objetos colidem, o maior incorpora a massa, representado pela área, e o momento do objeto menor. O jogador controla um desses objetos e tem como objetivo incorporar todos os objetos controlados por jogadores adversários. Sua única ação é clicar na tela, e ao fazer isso, seu objeto ejeta um pequeno pedaço em direção ao mouse ganhando momento na direção contrária. Para vencer o jogador deve jogar seu objeto em direção aos fragmentos espalhados pela tela até ter massa superior aos adversários para então colidir com eles. Para esse trabalho, o objeto adversário anda aleatoriamente pela tela, incorporando os fragmentos pelo caminho. Figura 5

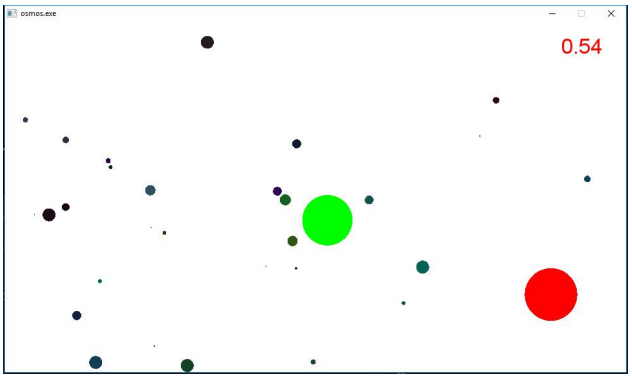

Figure 5. Osmos desenvolvido em Allegro. Jogador: círculo verde. Adversário: círculo vermelho. Os demais objetos são fragmentos.

O Agente

O agente racional escolhido foi o mais simples possível. Como o escopo deste trabalho não é a inteligência artificial em si, e sim o ambiente de interação entre ela e o jogo, o agente escolhido foi a aleatoriedade. O agente recebe as opções de ações possíveis no jogo e decide arbitrariamente quando e qual botão será apertado, ou onde no caso de usar o mouse, para cada intervalo de tempo previamente estabelecido.

4. Resultados

Foram feitos testes utilizando o nosso agente racional para cada um dos jogos citados acima. A primeira parte foi igual para todos, ao iniciar o Identificador é feita a leitura do código fonte e as ações possíveis foram armazenadas. A possibilidade do uso do mouse, no caso do Osmos e a lista de teclas acionáveis para os outros dois jogos.

Com as ações possíveis estabelecidas foi chamado o Atuador que iniciou o jogo e fez a mediação da interação agente-jogo até o final deste. Como as ações do agente eram aleatórias, era esperado um baixo rendimento deste, e foi o que aconteceu em todos os jogos.

Em Frogger, o sapo andou aleatoriamente pela tela sendo atropelado várias vezes até que chegou ao topo. Isso se deu em tempos variados e só foi possível quando foi concedida a imortalidade a ele. Quando o sapo só tinha uma vida o jogo só durava poucos segundos e ele raramente chegava ao meio da tela.

No Guttar Hero todas as partidas foram muito rápidas. Como apertar o botão com o seu respectivo espaço vazio gera penalidade na pontuação, o agente sempre atingia a pontuação mínima antes do círculo entrar no quadrado e o jogo acabava. Duas estratégias poderiam ser tomadas para permitir um jogo mais duradouro: retirar a restrição de

pontuação mínima, fazendo que o jogo não se encerre por o agente ter uma pontuação muito baixa ou aumentar o tempo entre as ações do agente, fazendo que ele demore mais para perder pontos. Contudo tirar a restrição iria apenas prolongar o tempo de jogo e aumentar a pontuação negativa do agente, sem contudo apresentar algum benefício ao resultado final, enquanto aumentar o tempo entre as ações não garantiria que ele iria apertar a tecla certa no momento em que o símbolo estivesse totalmente dentro do quadrado e, por consequência ele poderia perder pontos não só por apertar a tecla com o quadrado vazio como por deixar o símbolo passar. Assim essas ações não foram tomadas.

No Osmos o agente clicava aleatoriamente na tela fazendo com que seu círculo vagasse errante por ela, expelindo pequenos fragmentos no caminho. Na maioria das vezes a taxa em que o círculo do agente diminuía, por liberar fragmentos, era maior que a taxa em que ele crescia, por capturá-los, e assim quando colidia com seu adversário, esse era sempre superior em tamanho e o capturava, o que fazia que nosso agente perdesse o jogo. Em uma partida o círculo do agente diminuiu tanto que foi capturado por um fragmento.

Apesar do fracasso do agente a plataforma se mostrou uma boa ferramenta para fornecer os dados necessários e mediar a interação do agente e o jogo. A cada ação tomada o Perceptor capturava a imagem da tela e fornecia ao agente, este porém não tinha os recursos necessários para interpretação dos dados e usá-los em sua tomada de decisão. A figura 6 exemplifica uma partida e Osmos através das capturas de tela obtidas.

5. Conclusão

Foi desenvolvido um ambiente cujo o objetivo é promover interação entre um agente racional e um jogo criado com a biblioteca Allegro. As restrições para o uso dessa ferramenta são a necessidade de ter o código fonte do jogo e o agente ser capaz de interpretar a tela do jogo para a tomada de decisão. Porém o acesso ao código fonte é também uma vantagem, pois o pesquisador terá um meio de adequar o ambiente em que o agente está inserido ao propósito de sua pesquisa, não sendo obrigado a adequar a pesquisa a um jogo que consiga encontrar. Já a análise de dados visuais pode ser solucionada com ajuda de várias bibliotecas existentes e otimizadas para esse propósito, salientando que as demais soluções adotam a mesma estratégia de entregar ao agente uma imagem do jogo para que este a interprete em sua tomada de decisão.

No futuro, pretende-se desenvolver agentes mais inteligentes que possam aproveitar melhor os recursos da ferramenta desenvolvida, para que assim, sejam feitos testes de desempenho e as melhorias necessárias visando o um melhor suporte em pesquisas de IA em jogos.

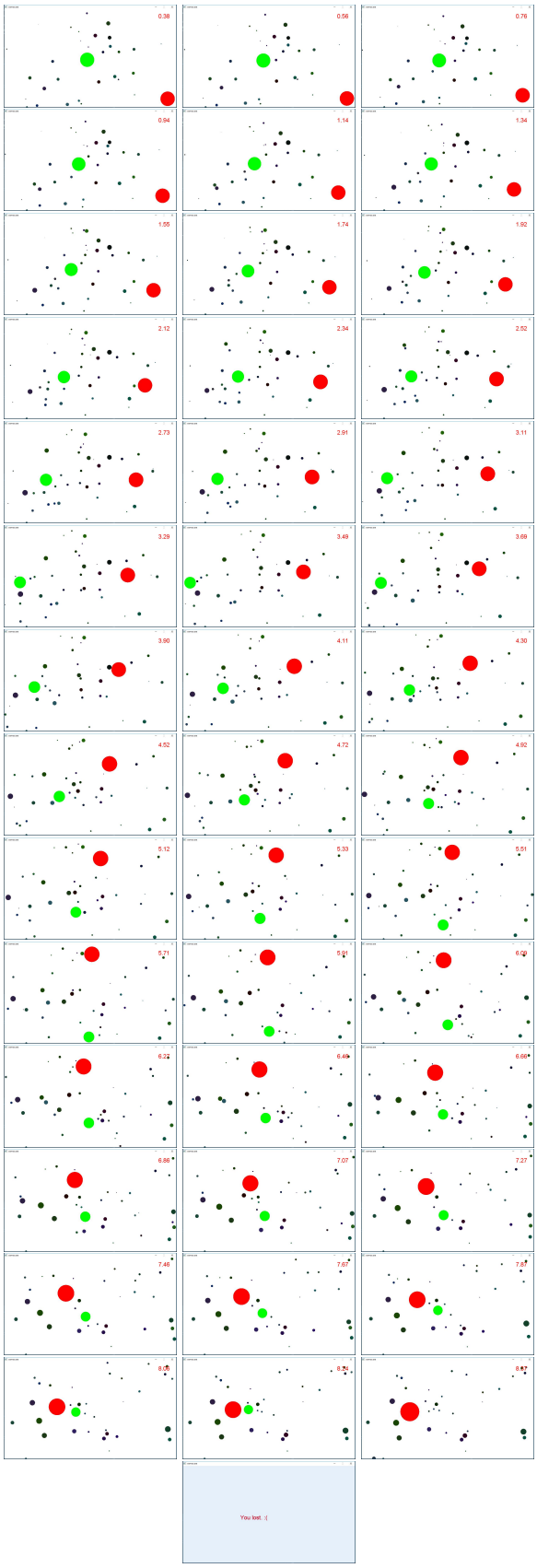

Figure 6. Sequencia de ações no jogo Osmos

Referências

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, (47):253–279, 2013.

Leverton, M. Site allegro. https://www.allegro.cc/about, acesso em 2019.

Perez-Liebana, D., Liu, J., Khalifa, A., Gaina, R. D., Togelius, J., and Lucas, S. M. General video game ai: a multi-track framework for evaluating agents, games and content generation algorithms. *CoRR*, abs/1802.10363, 2018.

Russell, S. J. Rationality and intelligence. *Artificial Intelligence - Elsevier Science B.V.*, (94):57–77, 1997.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362 (6419):1140–1144, 2018.

Vinyals, O., Babuschkin, I., Chung, J., Mathieu, M., Jaderberg, M., Czarnecki, W. M., Dudzik, A., Huang, A., Georgiev, P., Powell, R., Ewalds, T., Horgan, D., Kroiss, M., Danihelka, I., Agapiou, J., Oh, J., Dalibard, V., Choi, D., Sifre, L., Sulsky, Y., Vezhnevets, S., Molloy, J., Cai, T., Budden, D., Paine, T., Gulcehre, C., Wang, Z., Pfaff, T., Pohlen, T., Wu, Y., Yogatama, D., Cohen, J., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Apps, C., Kavukcuoglu, K., Hassabis, D., and Silver, D. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/, 2019.

Yannakakis, G. N. and Togelius, J. A panorama of artificial and computational intelligence in games. *IEEE Transacions on Computational Inteligence and AI in Games*, 7(4): 317–335, 2015.

Yannakakis, G. N. and Togelius, J. *Artificial Intelligence and Games*. Springer, 2018. http://gameaibook.org.