Relatório Final - IA para Jogos
Implementação do algoritmo GOAP

Alunos: Ezequiel Morais da Rocha e Silva e Fernanda Gonçalves Barros
Professores: André Mauricio Cunha Campos e Charles Andrye Galvão Madeira

1. Introdução.....1

1. Introdução

O Planejamento Orientado a Objetivos, GOAP (Goal-Oriented Action Planning), é um algoritmo utilizado na área de inteligência artificial para jogos, que visa aprimorar a tomada de decisões dos personagens não jogáveis (NPCs) dentro do ambiente do jogo. Desenvolvido para lidar com situações dinâmicas e complexas, o GOAP proporciona uma abordagem flexível e eficiente para a criação de comportamentos autônomos e adaptativos.

A base do GOAP reside na definição de objetivos que o NPC deve alcançar e na construção de um plano de ação para atingir esses objetivos. Cada ação é associada a um conjunto de pré-condições e efeitos, permitindo que o algoritmo avalie quais ações são mais adequadas para alcançar um determinado objetivo em um determinado contexto.

A abordagem GOAP permite uma maior flexibilidade que outras técnicas como máquinas de estado-finito, pois todas as ações são separadas umas das outras e mudanças em uma ação não vão influenciar outras ações diretamente. Em vez disso, as ações da abordagem GOAP dependem apenas de suas condições e estados (do mundo e do personagem), assim, um mesmo objetivo pode precisar de dois planos de ação diferentes para personagens com estados diferentes ou estados de mundo diferentes.

2. Modelo

Para este experimento foi fornecido um código base implementado em typescript, o algoritmo inicial está disponível neste link.

Foi criado então um modelo de mundo e npcs para testar o algoritmo fornecido. Cada ação do modelo possui um custo, uma lista de requisitos e uma lista de efeitos. Neste cenário o npc é um caçador que tem como objetivo se alimentar, para isso tem as seguintes ações:

O estado inicial de mundo e npc é o seguinte:

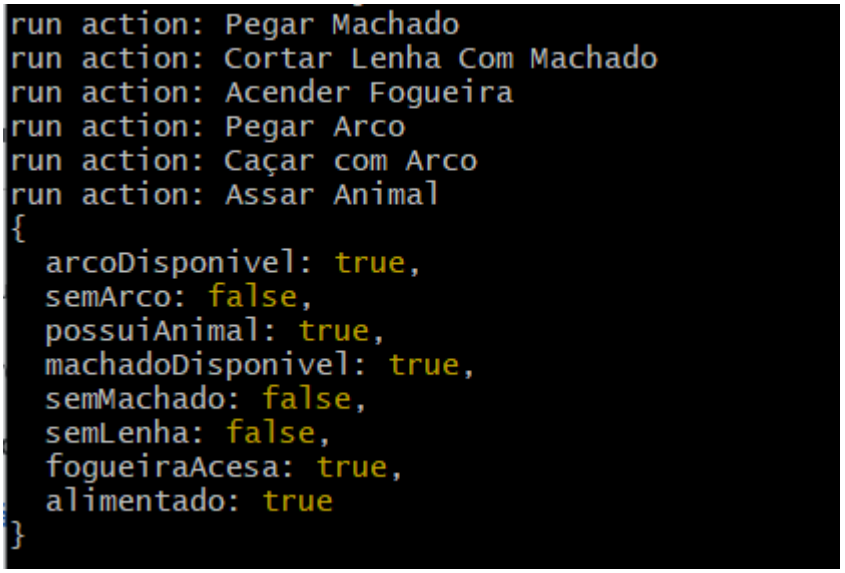
- arcoDisponível: true
- semArco: true
- machadoDisponivel: true
- semMachado: true
- semLenha: true
- fogueiraAcesa: false
- alimentado: false

Isso significa que o npc inicial sem arco, mas um arco está disponível, sem lenha e sem machado mas um machado está disponível, ele está sem fogueira e está com fome.

Para se alimentar o npc precisa assar um animal, para isso ele precisa caçar um animal e acender uma fogueira. O npc pode caçar um animal com ou sem um arco, dependendo se existem arcos disponíveis e pode acender uma fogueira com gravetos ou com lenha, dependendo se existem machados disponíveis.

3. Análise

O resultado obtido ao rodar o programa foi o seguinte:



O npc primeiro pegou um machado (custo 3), pois um machado estava disponível no início, depois cortou a lenha com o machado (custo 4) pois ele tinha um machado após a última ação, em seguida acendeu uma fogueira (custo 5) pois ele conseguiu lenha após a última ação, depois ele pegou um arco (custo 2) pois um arco estava disponível, e caçou um animal com o arco (custo 5) pois ele tem um arco e então assou o animal (custo 4) e se alimentou. O custo total desse caminho foi de 22.

Vamos agora analisar qual será o resultado caso o npc inicie sem um arco disponível e comum machado disponível porém com já com lenha.. Sem o arco ele terá que caçar sem arco (custo 10), e não vai precisar acender uma fogueira, com isso o novo custo será de 14.

O novo estado inicial de mundo e npc é o seguinte:

- arcoDisponível: false
- semArco: true
- machadoDisponivel: true
- semMachado: true
- semLenha: false
- fogueiraAcesa: false
- alimentado: false

O novo resultado é este:

```
$ node build/index.js
run action: Caçar Sem Arco
run action: Acender Fogueira
run action: Assar Animal
{
  arcoDisponivel: false,
  semArco: true,
  possuiAnimal: true,
  machadoDisponivel: true,
  semMachado: true,
  semLenha: false,
  fogueiraAcesa: true,
  alimentado: true
}
```

Como podemos ver, o resultado foi condizente com as previsões feitas.