Agente Autônomo com Busca

Filipe Calegario - Sistemas inteligentes (2021.2)

Relatório de Post-mortem

Eneri Melo Dangelis - emd
Fernando do Rego Pessoa de Macedo Neto - frpmn
Riei Joaquim Matos Rodrigues - rjmr
Victor Hugo Meirelles Silva - vhms
Victor Ximenes Carneiro de Oliveira - vxco

1. Introdução

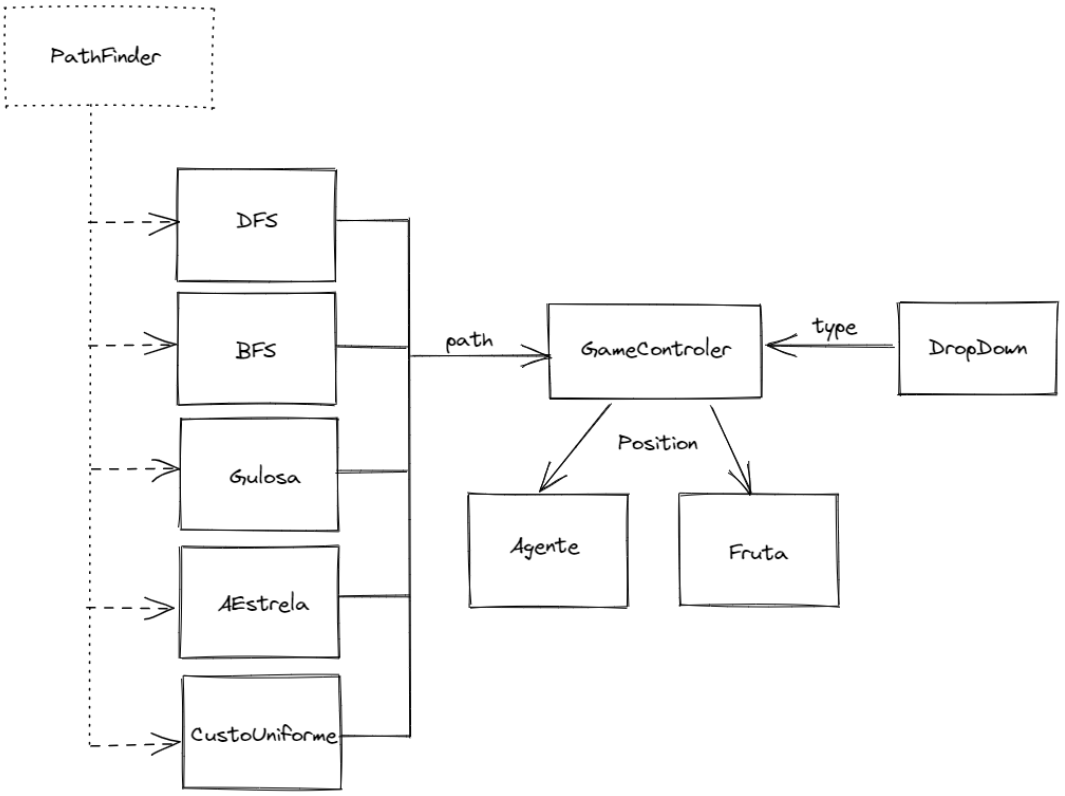
   Nesse projeto, temos um agente que nasce aleatoriamente em ambiente 2D criado também de forma aleatória com tipos variados de terrenos e que em algum lugar dele tem uma comida.
   Nosso objetivo é demonstrar como o agente pesquisa o ambiente para encontrar a comida através de algoritmos (Largura, Profundidade, Custo Uniforme, Gulosa e A*), buscando encontrar um path da posição do agente para o da comida.

2. Tecnologias usadas

   Como foco principal é demonstrar passos intermediários da busca, tendo um apelo visual importante, para facilitar a implementação de um GUI e o processo de animação da pesquisa, optamos por desenvolver o projeto usando o framework da Unity, Com isso passamos a ter um amplo suporte a construção, atualização e monitoramento do ambiente, além de facilidade de criar ciclos de animação.
   Com o uso da Unity, desenvolvemos o projeto na linguagem C# que assim como java é fortemente baseada em orientação a objetos, com isso usamos da herança e o polimorfismo para criar uma estrutura base que definiu o modelo de negócios entre os algoritmos de busca e a parte gráfica.

3. Arquitetura

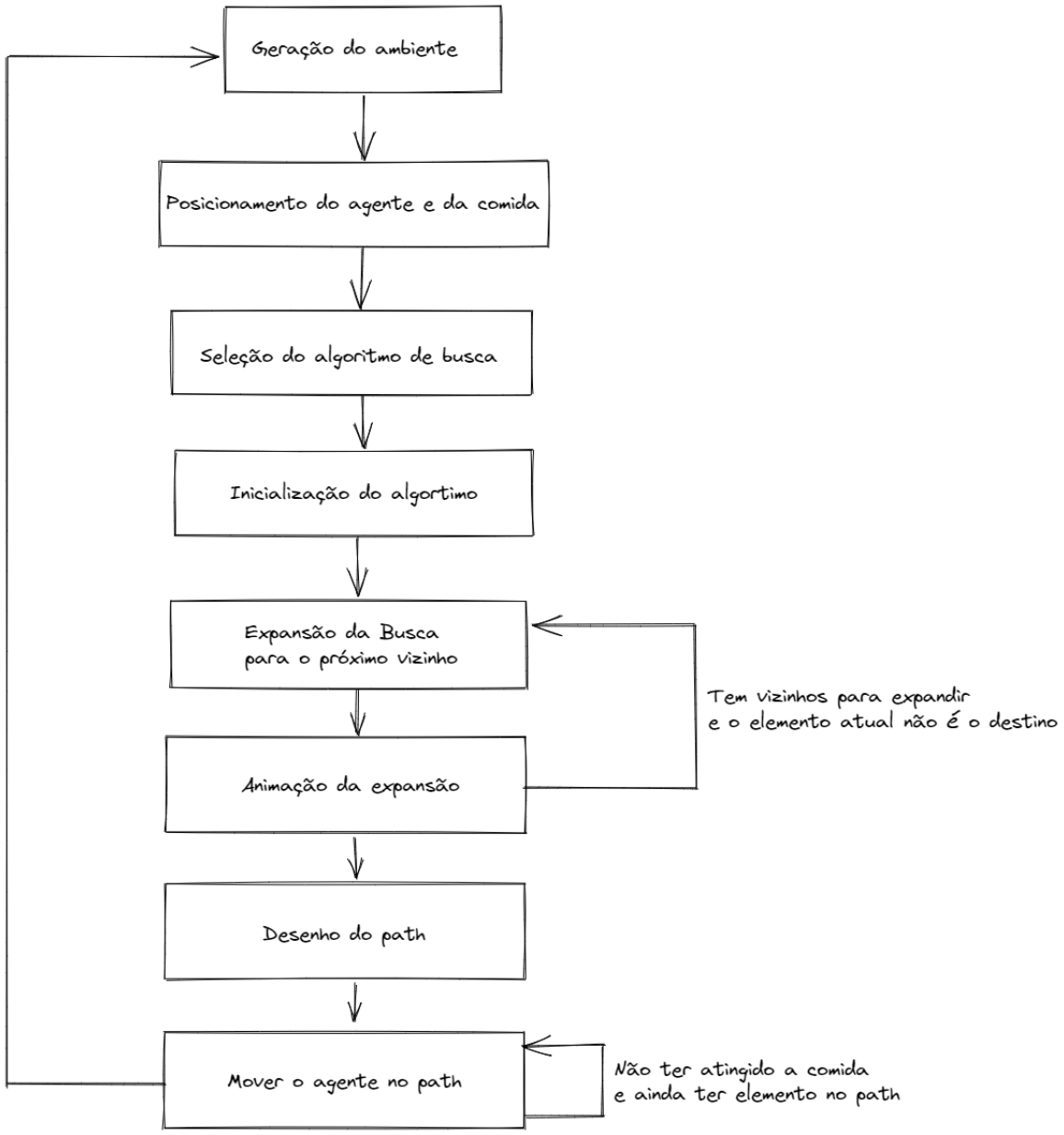
   Como foi utilizado o framework da unity em C#, para termos um desenvolvimento colaborativo com todos os membros podendo desenvolver de forma paralela, tentando reduzir a quantidade de conflitos foi criada a seguinte estrutura:



- pathFinder: Classe que fornece o método para dado uma posição no grid, retornar seus vizinhos e pesos. Além disso, ela possui os métodos virtuais necessários de serem implementados para seguir o modelo de negócios esperado pelo gameControler. As classes específicas de cada algoritmo que herdam de pathFinder tem que implementar:
    - init: Faz o setup de variáveis internas necessárias e define a origem da pesquisa.
    - iteration: Calcula a próxima iteração do algoritmo de pesquisa, a expansão para o próximo elemento.
    - getPath: a partir do contexto da execução da busca obtém uma lista de pontos que representa o caminho que o agente deve seguir para chegar ao destino.
- Dfs, Bfs, Gulosa, AEstrela, CustoUniforme: Classes filhas de pathFinder que implementam as suas funções virtuais seguindo seus respectivos algoritmos.
- Agente: Classe que representa o estado do agente no ambiente, tendo sua posição atualizada pelo GameControler.
- Fruta: Classe que representa o estado da fruta no ambiente, tendo sua posição atualizada pelo GameControler.
- DropDown: Classe que monitora e atualiza a comboBox que define qual algoritmo de busca deve ser usado.
- GameControler: Classe principal, que executa o loop de execução completo, possuindo objetos e links com das outras classes, para

chamar seus métodos como subrotinas, receber e enviar atualização de status.

Usando a ideia de animação frame a frame na unity para mostrar o processo de pesquisa passo a passo, criamos o seguinte fluxo de execução:



4. Processo de desenvolvimento

Uma limitação inicial foi que somente um membro do grupo, Victor Hugo, já tinha trabalhado com unity anteriormente, com isso, para familiarizar todos ao framework, nos juntamos todos em uma seção de pair-programming para desenvolver uma base comum, a estrutura já explicada na seção anterior, para que todos para que victor introduzisse os outros membros ao funcionamento da unity e entendesse um pouco para poder desenvolver suas partes.

No desenvolvimento dessa base comum, o grupo focou em desenvolver a estrutura backend do código, a geração do mundo com perlin noise, definindo

as matrizes e variáveis internas para servir de base para o projeto. Enquanto isso, Victor Hugo focou em desenvolver as funcionalidades de pintura na tela para mostrar os estados intermediários da pesquisa, mas sempre tirando as dúvidas de outros membros.
Tendo essa base comum desenvolvida com todos os membros tendo uma noção básica de onde mexer, dividimos as implementações de cada algoritmo por cada membro, sendo a responsabilidade de cada um implementar seus métodos e integrar com a base comum.

- Busca em largura e em profundidade: Riei e Victor Hugo ficaram o esses dois algoritmos, nós precisamos validar de a arquitetura e as funcionalidades de desenho na tela funcionaria corretamente, para esse algoritmos ao fazer esses dois algoritmos primeiro conseguimos antecipar complicações para os demais membros, de como era mais complicado executar corretamente os algoritmos mantendo o contexto, quando ele é executado step-by-step invés de com um loop até encontrar o objetivo.
- Custo-Uniforme ficou com Victor Ximenes, Gulosa ficou com Eneri e A-estrela ficou com Fernando, para esses algoritmos a falta de uma implementação nativa de uma fila de prioridades foi a principal dificuldade, após tentativas frustradas de incluir bibliotecas de terceiros, acabaram por implementar por conta própria essa estrutura de dados.

Os principais erros do desenvolvimento foi justamente o erro de recuperação do contexto e o uso incorreto de estrutura de dados necessárias, principalmente no caso da fila de prioridades, que demandou muito tempo de debug baseado na própria animação de expansão da busca e de estados intermediários, mas foi muito importante pois com isso conseguimos entender a fundo como os algoritmos funcionam.
O principal saldo positivo do projeto é que todos os membros agora conseguem desenvolver utilizando o framework da unity, e passaram a entender como um framework para desenvolvimento de agentes inteligentes.