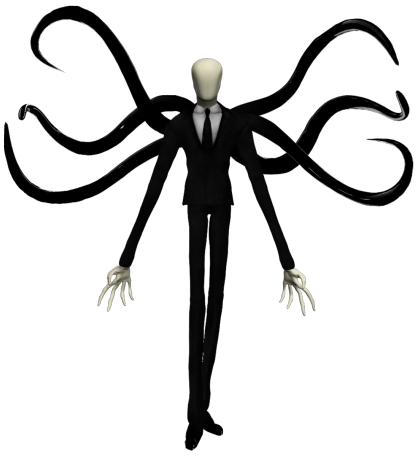
Slender: The Eight Devices

Grupo 9:
Anabela Costa e Silva, up201506034
Beatriz Souto de Sá Baldaia, up201505633

Problema:



O projeto baseia-se no jogo **“Slender: The Eight Pages”** onde o personagem principal, o jogador, é abandonado numa floresta, apenas com uma lanterna, tendo de recolher oito páginas espalhadas pelo ambiente, enquanto tenta sobreviver ao monstro, Slender Man.

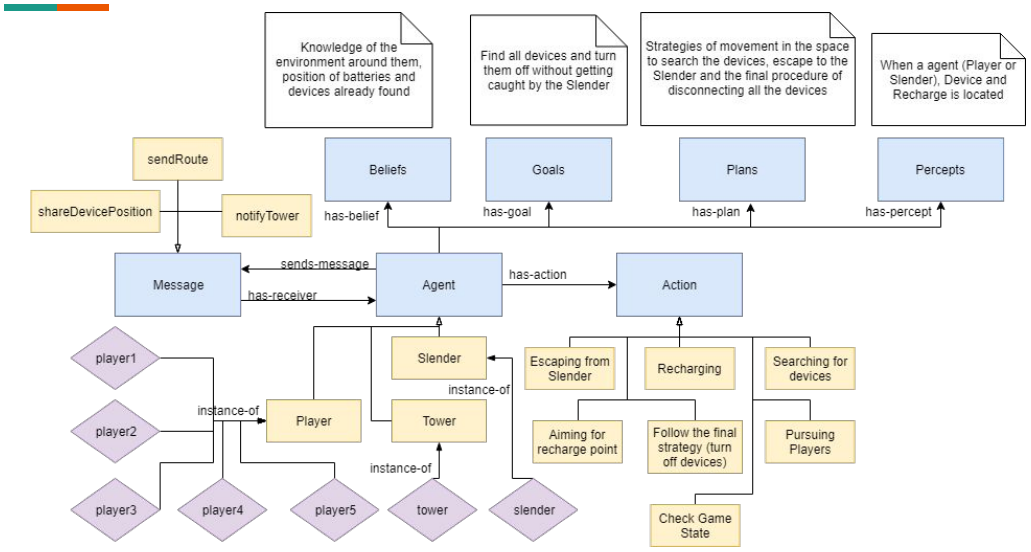
No nosso projeto seriam **oito dispositivos** que teriam de estar desativados ao mesmo tempo, para terminar o jogo. No entanto, cada dispositivo só fica **desativado durante um tempo limite**.

Problema:

Cada jogador possui também um **telemóvel** que lhe permite comunicar com os restantes jogadores assim como ver melhor na floresta. Mas o telemóvel ligado também atrai o Slender e tem bateria limitada, pelo que terá de ser **recarregado**.

Em contraste ao jogo original, aqui não temos apenas uma pessoa sozinha na floresta, mas sim **um grupo de amigos** que cooperam para que consigam escapar o mais rápido possível!

Esquema do tipo de agentes:

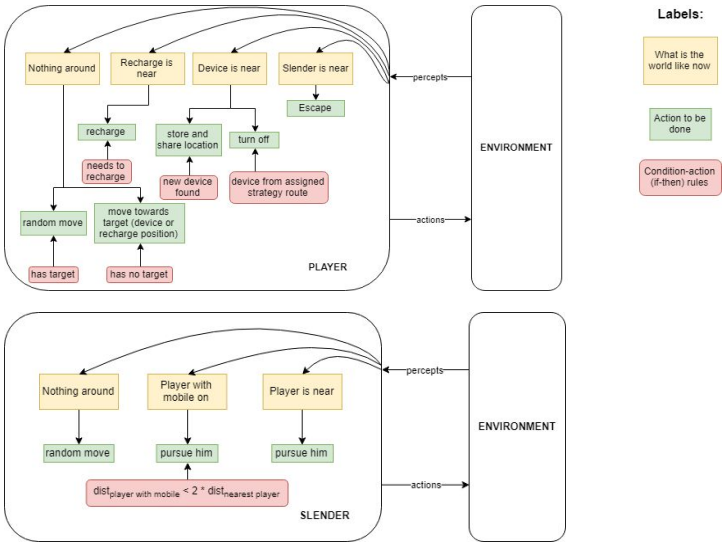


Os agentes envolvidos no nosso trabalho dividem-se de diferentes formas.

O Slender e os Players movem-se no espaço para realizar os seus objectivos, matar todos os Players e desligar todos os dispositivos respectivamente.

Enquanto a Tower, que não se encontra no espaço, realiza uma função mais administrativa, sendo a responsável por terminar o jogo e gerir os Players, dando-lhes rotas para desligar os dispositivos.

Arquitetura e estratégia



O Slender tem uma arquitectura reactiva. Se um jogador se encontrar na sua vizinhança, ele tenta apanhá-lo, se não existir nenhum jogador perto move-se aleatoriamente.

A arquitetura dos nossos agentes Players é híbrida e usa o mecanismo de subsunção. Primeiro preocupam-se em fugir do Slender. Caso ele não se encontre nas redondezas, age de acordo com o seu estado atual e perceções que extrai do ambiente.

Os Players cooperam entre si de diversas maneiras. Numa primeira instância, partilham a posição dos dispositivos que encontram. E no final cumprem rotas que conjuntas levam a desativação dos dispositivos todos dentro do período limite em que ficam desligados.

Uma rota é uma sequência de dispositivos a desligar, ciclicamente, sendo que o custo de um ciclo não excede o período limite de um dispositivo estar inativo.

Uma rota é atribuída ao Player mais próximo que não seja já responsável por uma outra rota.

Se o responsável por uma rota morrer, porque foi apanhado pelo Slender, a rota é atribuída a outro Player que não tenha uma rota. Caso não haja mais Players disponíveis, é fim de jogo e significa que o Slender ganhou, já que os atuais sobreviventes nunca conseguirão atingir o estado de todos os dispositivos estarem desligados ao mesmo tempo.

O agente Tower é quem verifica o estado do jogo, executa o algoritmo da estratégia final e distribui as rotas calculadas pelos Players.

Comunicação:

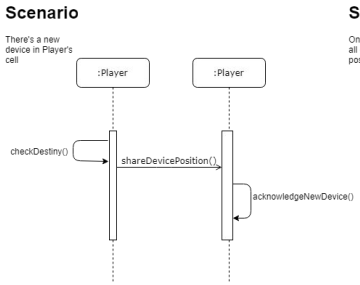
A comunicação dos agentes realiza-se entre:

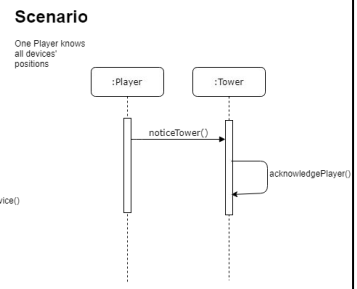
- a torre e os jogadores
- os diferentes jogadores

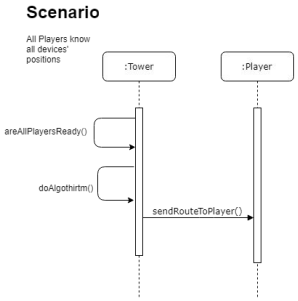
Cada jogador avisa os restantes quando encontra um dispositivo novo, aumentando a rapidez com que estes descobrem os dispositivos todos.

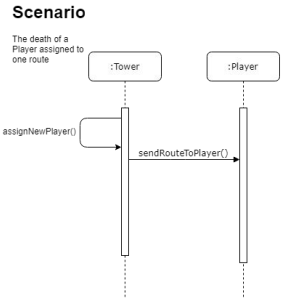
A torre é contactada pelos jogadores quando estes sabem a localização de todos os dispositivos.

A torre é quem envia as rotas calculadas aos jogadores para que estes executem a estratégia final.









Software:

Utilizamos as seguintes ferramentas:

- **Repast Simphony**, para a criação de uma interface gráfica, assim com para a realização das experiências executadas;
- **JADE**, para a descrição e implementação dos agentes e dos seus comportamentos;
- **SAJaS**, para a ligação entre os softwares anteriormente referidos.

Experiências:

Realizamos várias experiências usando o “batch run” do Repast, assim alteramos os parâmetros para ver como eles influenciam o resultado.

```
<?xml version="1.0" ?>
<sweep runs="1">
   <parameter name="player_count" type="list" value_type="int" values="5 8">
      <parameter name="slender_running_speed" type="list" value_type="double" values=".5 .8">
         <parameter name="slender_radius" type="list" value_type="int" values="1 5">
            <parameter name="player_big_radius" type="list" value_type="int" values="5 8">
               <parameter name="player_speed" type="list" value_type="int" values="1 3">
                  <parameter name="player_small_radius" type="list" value_type="int" values="1 3">
                     <parameter name="slender_walking_speed" type="list" value_type="double" values=".1 .3">
                     </parameter>
                  </parameter>
               </parameter>
            </parameter>
         </parameter>
      </parameter>
   </parameter>
</sweep>
```

Temos sete parâmetros para alterar e estamos interessados em dois resultados no final de uma simulação:

- o número de **jogadores vivos;**
- o **tempo** que demorou a acabar o jogo.

Análise:

O BatchRun do Repast Simphony devolve-nos um mapa de parâmetros, com a descrição de todas as simulações realizadas, e um ficheiro com os resultados obtidos.

Este ficheiro mostrar o resultados da simulação, dando-nos a informação sobre quantos Players estavam vivos e quantos dispositivos estavam ativos a cada tick do relógio.

No sentido de evitar ciclos, jogos em que o Slender não consegue matar todos os Players, mas estes também não são capazes de desligar os dispositivos, pelo facto de os Players terem de continuar a fugir quando o Slender se aproxima, o que influencia a realização das rotas, demos um tempo limite de 2000 ticks para os BatchRun.

Conclusões:

Neste trabalho implementamos o jogo Slender Man como um sistema multi-agente.

Cumprimos todos os objectivos a que nos propusemos, pondo em prática os conceitos lecionados nas aulas.

A contribuição dos elementos do grupo foi equitativa.

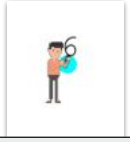
Na segunda parte deste projecto iremos aprofundar a análise das experiências realizadas, tendo como objectivo perceber o impacto de cada um dos parâmetros no sistema global.

Informação Adicional
Segunda parte

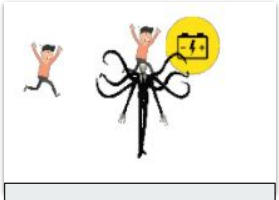
Exemplos detalhados de execução

Jogador com o telemóvel ligado à procura de um dispositivo

Jogador com o telemóvel desligado à procura de um dispositivo



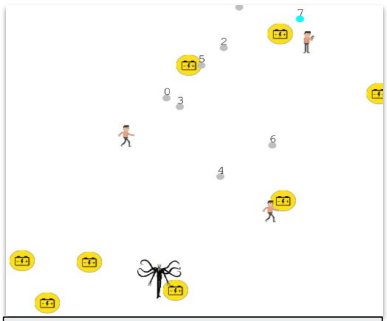
Jogador encontrou um dispositivo, desligou-o e está a enviar uma mensagem aos restantes jogadores.



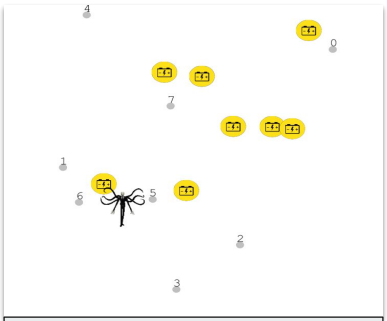
Jogadores a fugir do Slender

Jogador a carregar o telemóvel

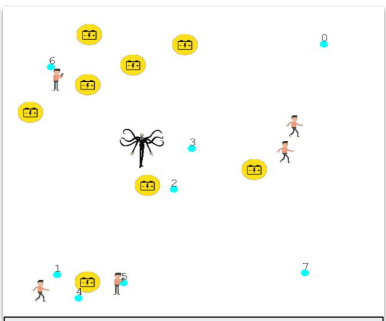
Exemplos detalhados de execução



Fim de jogo com o Slender a ganhar porque não há jogadores suficientes para se atingir o estado em que todos os dispositivos se encontram desligados.



Fim de jogo com o Slender a ganhar, matando todos os jogadores.



Fim de jogo com vitória dos jogadores.

Classes Implementadas

Foram implementadas 8 classes assim como uma enumeração “State”.

- **Player**, agente jogador;
- **Slender**, agente monstro;
- **Tower**, agente Torre;
- **Device**, dispositivo que pode ser desligado por um jogador que se encontre na mesma célula;
- **Recharge**, ponto onde o jogador pode carregar o telemóvel;
- **Node**, abstração de dispositivo usada para o algoritmo de procura de rotas;
- **SlenderManBuilder**, criador do jogo/modelo em Repast e dos agentes JADE;
- **Style**, classe que altera a apresentação do display dos agentes no espaço, para facilitar e tornar mais apelativa a análise do modelo.

Player

A classe **Player** deriva da classe **Agent** e representa um jogador. Uma instância desta classe tenta, colaborativamente com os outros jogadores, descobrir todos os dispositivos e desligá-los. Um agente Player tem dois Behaviours: "**Exploring**" e "**ListeningBehaviour**", ambos cíclicos.

O ListeningBehaviour é responsável por receber as mensagens, quer dos outros jogadores (mensagem de id "device_found") quer da torre de controlo, instância da classe Tower (mensagem de id "target_route").

O Exploring é responsável pela movimentação do agente no espaço. Dependendo da vizinhança e estado atual do Player, o comportamento do agente é alterado, mas mantém sempre alguns princípios, como o facto de se o Slender se aproximar do jogador ele ter de fugir e ser prioritário carregar o telemóvel em relação a procurar ou se aproximar de um dispositivo. O Player quando liga o telemóvel aumenta o seu raio de visão, mas também passa a ser avistado pelo Slender, independentemente da distância física entre eles.

Player

A bateria do telemóvel de um Player tem uma capacidade máxima de 100 e vai diminuindo a cada tick em que o agente o tem ligado.

Um Player tem um estado podendo este ser “**EXPLORING**” (estado em que o jogador ainda anda à procura dos dispositivos), “**WAITING**” (estado em que o jogador já tem conhecimento de todos os dispositivos, mas está à espera que os restantes jogadores também estejam prontos e que a torre - instância Tower - planeie uma estratégia e distribue as rotas a percorrer) e “**ACTING**” (estado em que já se conhece todos os dispositivos, havendo jogadores responsáveis por desligar uma determinada sequência de dispositivos).

Slender

A classe **Slender** deriva da classe **Agent** e representa um monstro, que tenta apanhar todos os jogadores e matá-los.

Este agente tem apenas um CyclicBehaviour chamado "**Hunting**" que orienta a ação do Slender. Primeiro ele tenta encontrar o jogador mais próximo e, dos jogadores com o telemóvel ligado, o mais próximo. Depois escolhe a sua vítima. Ela é o jogador mais próximo do Slender ou o com o telemóvel ligado, se a distância entre este e o Slender for menor que o dobro da distância entre o Slender e o Player mais próximo dele. O Slender tenta movimentar-se para essa vítima, mas se não existir nenhuma vítima, ou seja, nenhum jogador visível, ele movimenta-se aleatoriamente.

Assim, quando o Slender chega a uma célula e nela existir pelo menos um jogador ele mata-o.

Tower

A classe Tower refere-se ao controlador do jogo, que verifica se o jogo deve acabar e organiza os jogadores quando estes conhecem a localização de todos os dispositivos. Este agente tem dois Behaviours: “**WatchGame**” e“**ListeningBehaviour**”, ambos cíclicos.

O ListeningBehaviour é responsável por receber as mensagens dos jogadores, que lhe informam de quais dispositivos têm conhecimento e de quando eles têm conhecimento de todos os dispositivos.

O WatchGame é responsável pelo controlo do jogo. Ele mantém consistente o temporizador dos dispositivos, ligando-os se for caso disso, verifica se o jogo já acabou, executa o algoritmo da estratégia final se todos os jogadores estiverem prontos e resolve os casos em que a morte de um jogador compromete a finalização da estratégia.