## 1. Descrição do Problema

*Após matar o rei de Hyrule, o mago Agahnim está mantendo a princesa Zelda prisioneira e pretende romper o selo que mantem o malvado Ganon aprisionado no Dark World.*

*Link é o único guerreiro capaz de vencer o mago Agahnim, salvar a princesa Zelda e trazer a paz para o reino de Hyrule. Porem, a única arma forte o suficiente para derrotar o mago Agahnim é a legendaria Master Sword (**Figura 1**), que encontra-se presa em um pedestal em Lost Woods.*

*Para provar que é digno de empunhar a Master Sword, Link deve encontrar e reunir os três Pingentes da Virtude: coragem, poder e sabedoria (**Figura 2**). Os três pingentes encontram-se espalhados pelo reino de Hyrule.*

*O seu objetivo é encontrar os três pingentes da virtude e em seguida ir para Lost Woods até a legendaria Master Sword.*

**Figura 1. Master Sword.**

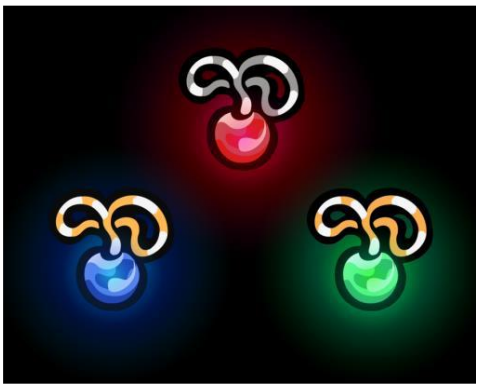

**Figura 2. Pingentes da Virtude.**

## 2. Implementação

O Trabalho consiste em implementar um agente capaz de locomover-se autonomamente pelo reino de Hyrule e reunir os três Pingentes da Virtude. Para isso, você deve utilizar o **algoritmo de busca heurística A***.

O agente deve ser capaz de calcular automaticamente a melhor rota para reunir os três pingentes da virtude e ir para *Lost Woods*, onde está localizada a *Master Sword*.

O mapa do reino de *Hyrule* é mostrado na Figura 3.

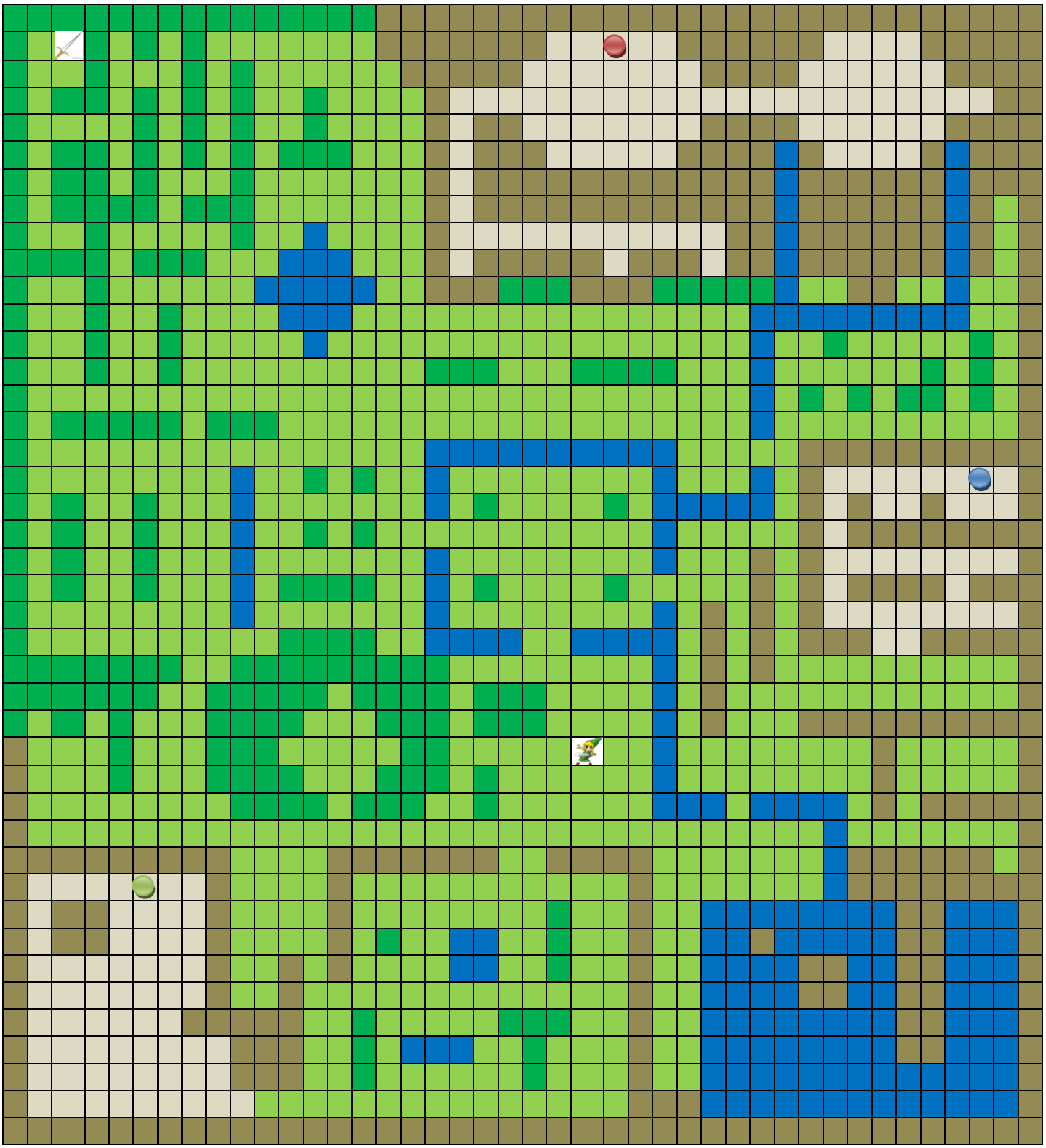

**Figura 3. Mapa do reino de Hyrule.**

O reino de *Hyrule* é formado por **5 tipos de terrenos**: grama (região verde claro), água (região azul), montanha (região marrom), areia (região marrom claro) e floresta (região verde escuro).

Os custos para passar por cada tipo de terreno são os seguintes:

- **Grama** – Custo: +10
- **Areia** – Custo: +20
- **Floresta** – Custo: +100
- **Montanha** – Custo: +150
- **Água** – Custo: +180

Os **três pingentes da virtude** estão identificadas no mapa pelos círculos coloridos.

Link inicia sua jornada na posição [24, 28] e termina após reunir os três pingentes da virtude e chegar até a Master Sword (posição [3, 2]). A melhor rota para cumprir essa missão é a rota de menor custo levando em consideração o terreno.

## 3. Informações Adicionais

- O mapa principal deve ser representado por uma matriz 42 x 42 (igual à mostrada na Figura 3).

- O agente sempre **inicia** a jornada na casa do Link (ponto onde está o Link no mapa [24, 28]).

- O agente sempre **termina** a sua jornada ao chegar até a Master Sword (posição [3, 2]).

- Os pingentes podem ser coletados **em qualquer ordem**. Porem, ordens diferentes vão resultaram em custos totais diferentes. Idealmente o seu algoritmo deve ser capaz de identificar qual é a melhor ordem para coletar os pingentes com o menor custo final.

- O agente não pode andar na diagonal, somente na **vertical** e na **horizontal**.

- Deve existir uma maneira de **visualizar os movimentos** do agente, mesmo que a interface seja bem simples. Podendo até mesmo ser uma matriz desenhada e atualizada no console.

- **Os mapas devem ser configuráveis**, ou seja, deve ser possível modificar o tipo de terreno em cada local. O mapa pode ser lido de um arquivo de texto ou deve ser facilmente editável no código.

- O programa deve exibir o **custo do caminho percorrido** pelo agente enquanto ele se movimenta pelo mapa e também o **custo final** ao terminar a execução.

- O programa pode ser implementado em qualquer linguagem.

## 4. Dicas

- Note que este problema é semelhante ao problema do **Caixeiro Viajante**. É necessário encontrar a melhor rota para capturar todas os pingentes da virtude e chegar até a *Master Sword*. No trabalho não é obrigatória a resolução deste problema, mas é única maneira de garantir o melhor custo.

- Implemente a função de busca de uma forma genérica, pois será necessário executa-la múltiplas vezes para diferentes destinos.

## 5. Extra

- A interface gráfica não é o objetivo desse trabalho, mas quem implementar uma "boa" interface gráfica (2D ou 3D) para representar o ambiente e o agente receberá até 3 pontos extras na nota.

## 6. Orientações

- Datas das Entregas:
  - Nomes dos Grupos: **11/09/2020**
  - Trabalho Completo: **09/10/2020**
  - Apresentação: **12/10 a 15/10/2020**

- O trabalho pode ser feito **individualmente** ou em **grupos** de no máximo **4 (quatro) alunos**. Os membros dos grupos podem ser de qualquer uma das turmas (Computação ou Sistemas) (não precisam ser obrigatoriamente da mesma turma).

- Na data especificada, cada grupo deverá entregar, via ***Moodle***, um arquivo comprimido (ZIP ou RAR) contendo todo o código.

- Cada grupo deverá apresentar o trabalho ao professor, na data especificada acima, em horário a ser marcado. **TODOS** os membros do grupo devem estar presentes para a apresentação, mostrando o funcionamento do algoritmo.