CS420 - Project 02

Treasure Island

20125074 - Dương Hoàng Huy

20125107 - Hồ Văn Quân

20125123 - Hoàng Thanh Tùng

20125051 - Trần Nguyễn Đăng Tâm

Table of contents

Self-assessment

No	Criteria	Comments	Completion
1	Create 5 maps: - 3 maps with sizes: 16x16, 32x32, 64x64 - 2 maps larger than 64x64	Folder: Map, two larger size: 72 & 80	☑
1.1	Implement a map generator	File: map_generation	☑
2	Implement the game rule		☑
3	Implement the logical Agent		☑
4	Implement the visualization tools		☑
5	The Agent can handle a complex map with large size, many of regions, prison, mountains	Because the small screen, we only test with largest map size 80	☑
6	Report your implementation with some reflection or comments.		☑
7	Contest*		☐
	Total		100%

Note

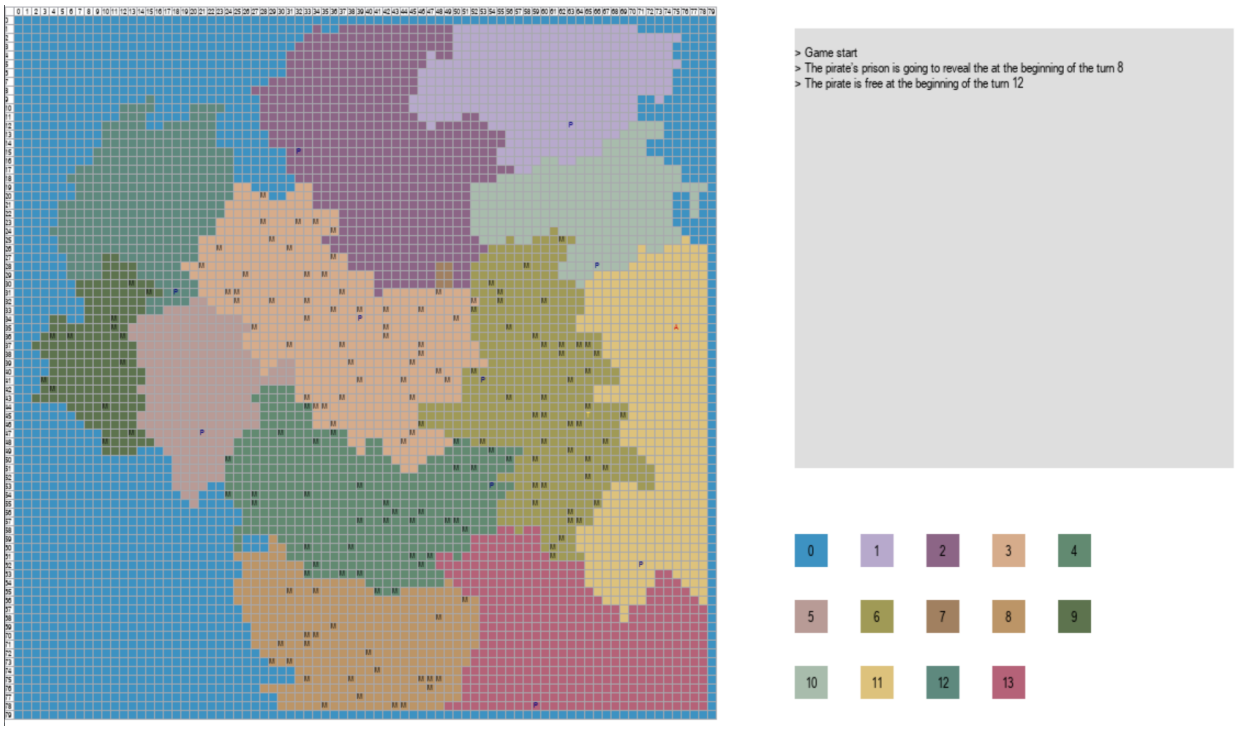
- All 15 hints are implemented successfully.
- We have an **HOW TO PLAY** section

I. Introduction

1. Overview

- The game strategy is entirely upon **random-based approach**, using heuristic logic processing on rules followed from the predefined rule set of the game
- random-based game hoàn toàn -> sử dụng chiến lược tự nghĩ dựa trên game rules (chung chung thôi)

2. Game design



3. How to play

- The game requires **Pygame** to run. Pygame is a set of cross-platform Python modules designed for building video games.
- How to play: **In main.py**, you can customize your map by choosing **input text file** for map information and **output text file** for log record. Run main.py and the game map will be displayed on the screen.
- **Press any key** to run each play turn, all events will be appeared on the log file
- **The log** process will be completed and saved whenever one of two players wins the game

II. Implementation

- Our game is built by many classes (objects) such as Grid, Map, Pirate ... which create the base of the game. We'll just basically discuss briefly about some features of each object above and focus more on some implementation or methods that support the logic of the game or the Agent
- Grid and Map are used to generate game maps, Pirate is the counterpart of our AI Agent who also tries to reach the treasure and mislead us, Agent is our logical agent, Hint is to keep track of each of all hints given by the pirate and Hint_Manager to store and manage all the hints
- We use A* algorithm to find the shortest path to the treasure location

1. Hint_Manager

Hint consists of 2 attributes:

- read_hint: generate needed components in each hint
- is_verified: automatically check validity of the hint, whether it is true or false
- **Hint_Manager**:
    - Manage all hints (and their implementation) in the whole game in the form of a list of Hints
    - Pick up a hint randomly in the hint list through method `get_random_hint`

III. Agent

- Here is some core variables and methods:

1. Variables

- **potential_locs**: the grid is the **center of the square area (size 5)** that is likely to contain the treasure **(has at least one non-masked grid)** and the pirate will move to those places as early as possible.
- **potential_loc:** the place is chosen to be the **current destination** of the Agent.
- **target**: the place where we think there is a pirate (before he is released from prison) or the current location of the pirate (after he is released from prison).

2. Methods

- **generate_potential_locs:** Generate all potential locations based on the current precepts of the Agent at that turn, including 2 processes::
- **Scan** the whole map with the **gap of value 2** to get all **potential_locs (For performance purposes).**
- **(Only used when the Pirate is free)** By tracking the Pirate's moves, we can find the directions that Pirate is trying to navigate to and **remove** all locations that are opposite that direction.
- After those 2 processes, if there is no potentials location left, we will do both processes again but with the gap of 1
- **update_percept:** continuously updates the current destination and the path to reach it
- **agent_ strategy**: handle the strategies as well as the logic of our Agent.

3. Agent strategies

- By analyzing the game rules, we realize that we cannot research any way to train the model (i.e. reinforcement learning) for the Agent because almost all the events are randomly happening.
- We decide to create some strategies based on the **phases**/**situations** of the game for the Agent to decide what it has to do.
- Base on the game rules, there are 3 important phases that we focus on:
    - *Before the revelation of the pirate's prison*
    - *After the revelation and of the pirate's prison and before the turn that pirate is free*
    - *After the pirate is free*
- In each turn, we will have a set of potential locations. We will evaluate those locations based on the heuristic function: the distance between each potential location and the target. The one that has the lowest heuristic (the nearest one) will be prioritized.

How Agent set the target

Before the revelation of the pirate's prison

- **Target**: all prisons.
- **All prisons** are the dangerous places in this phase because any of them can be the place of the Pirate. In the worst case scenario, the treasure is close to the Pirate's prison and wins the game immediately after freedom.

After the revelation and of the pirate's prison and before the turn that pirate is free

- **Target**: the prison that the Pirate is staying

After the pirate is free

- **Target**: the current location of the Pirate

How Agent decide what to do

- Our **move** in this section can be a **small** or a **large** move and the **large scan** action will be performed at the last move.
- The situation is dangerous when the pirate reaches potential_loc before Agent.
- We define **3 states** based on the distance between the Agent and the potential_loc:
    - **Close enough**: Agent can reach that place with a single move each turn without danger. In this phase, we can verify hints and make a move in each turn.
    - **Far enough**: Agent can reach that place with double moves each turn without danger. In this phase, we can only do the move actions.
    - **Too far**: Agent is impossible to reach that place in time. In this phase, we have to perform the teleport action (if available) to reach that place.

IV. Conclusion

- By using these strategies, our Agent can win almost games with our generated maps.