CSC 484/584, Spring 2020

Assignment 2 – Pathfinding & Decision-Making

Unity id: mnagdev

1. Drawing the World

The world is drawn by reading the position of the objects from a json file. The map has a dimension of 640*480.

- Drawing the knight
  Every game has a knight and the position specified in the json file is the starting position of the knight. I used PImage to draw the sprites.
- Drawing the key locations
  These locations are represented by the images and a key is included at the end of the report to help in identifying which location corresponds to which sprite. All the sprites are drawn using sprite centered approach.
- Drawing the obstacles
  The position of the obstacles is defined in the form of JSON Array in the .json file. I implemented a loop to read the vertices of the obstacles and used PShape to draw them.

2. Pathfinding

I used A* algorithm to find the path between the source and destination. In my implementation of A*, I am using the knight's current position as the starting position and the destination is the location selected on mouse click. I used Priority Queue to insert the nodes based on the cost calculated. The total cost equals the actual cost form the current node to the next node and an estimated cost from the next location to the destination. The node with the minimum cost is inserted first. The next nodes from the current nodes are the pixel positions at 2 pixels from the current location. I used grid-based approach to navigate the map. The knight can move 2 pixels to either right, left, up, down, top-left, top-right, bottom-left or bottom-right.

Following are the data-structures that I am using in this algorithm-

- Front – This is a priority queue which contains the nodes to be explored next. The nodes are inserted in the increasing order of cost.
- Came_from – This is a HashMap which maps the next location to the current location. This is helpful in tracing the final path from source to destination.
- Cost_so_far – This is also a HashMap which maps the next node to the actual cost of getting there form the source.

I used the Euclidean distance between the next node and destination as the heuristics. I chose this heuristic as it is consistent and admissible.

The algorithm can find a path from source to destination but the transition from one location to another location has some glitch and only the drawing part of the path is not clear and because of this the notion of natural movement cannot be obtained. I tried multiple ways to solve to glitch but was not able to achieve a proper display of the motion. However, the pathfinding algorithm returns an optimal path from source to destination. One of the improvements can be that the motion can be made to look natural.

3. Decision Making

I used Recursive DFS to perform decision making. There are four things that are searched for in the game which are 'Fenrir', 'Fire', 'Poisoned Sword', and 'Poisoned Fenrir'. If any one of them is found by the knight then the knight goes to tar-pit, fight Rameses and wins. If none of them is found, then the knight loses when it fights with Rameses.
The data structures used are-

- Knight_has – This is an ArrayList of the things which knight has.
- Find_items – These are the items which the knight is looking for.
- Has – A HashMap which maps the characters to the items they possess.
- Wants – A HashMap which maps the characters to items which they want to have in exchange of the things they possess.

I made a recursive function 'find' which takes the String of the item name as the argument. The function returns true if the item is found by the knight else returns false.

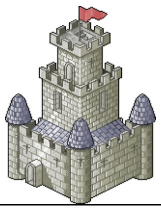
Following are the steps followed to find anything in the game world-

- Check the 'Has' map of the world and find the characters who possess the object needed.
- Check the wants of all the characters having that object.
- If the knight has the object which the character needs, then move to location of the character, exchange the object and return true else add the object to the list of the objects which the knight must find. Also, when the knight exchanges something, the 'knight_has' list is updated, and the new item is added to the list and the thing exchanged is subtracted from the list.
- Recursively call the 'find' function on the list of objects made in the previous step.

I chose this implementation because it is easy to implement, and I find it most appropriate for this situation. There are a smaller number of rules and therefore the DFS operation is not very costly. I tried to implement decision tree but failed because there are so many rules that need to be taken care off. It became tedious to code for every combination of has and want. It is time consuming and if one misses any case then the program will fail. Also, it is difficult to incorporate any changes to the decision tree. So, I found that Recursive DFS works the best. For the game world rules in which two objects can combine to form a new object like 'Blade + Wood = Sword', here if the knight is looking for sword then it won't check the 'Has' map for the sword but check for the basic things that it needs to collect from other characters that are blade and tree spirit (because wood can be acquired only if tree spirit is present). This is the strategy that I am using in my algorithm. Once the knight kills the Tree Spirit, I am removing the ‘wants’ and ‘has’ of Tree Spirit so that the knight doesn’t kill the Tree Spirit again. The knight greets the king only once.

The motion of the knight cannot be displayed correctly but the path is calculated correctly. All the actions are shown as the console output.

Sprites:

	Knight
	Maugrim Castle
	Cave
	Tar-Pit
	Tavern
	Tree

Screenshots

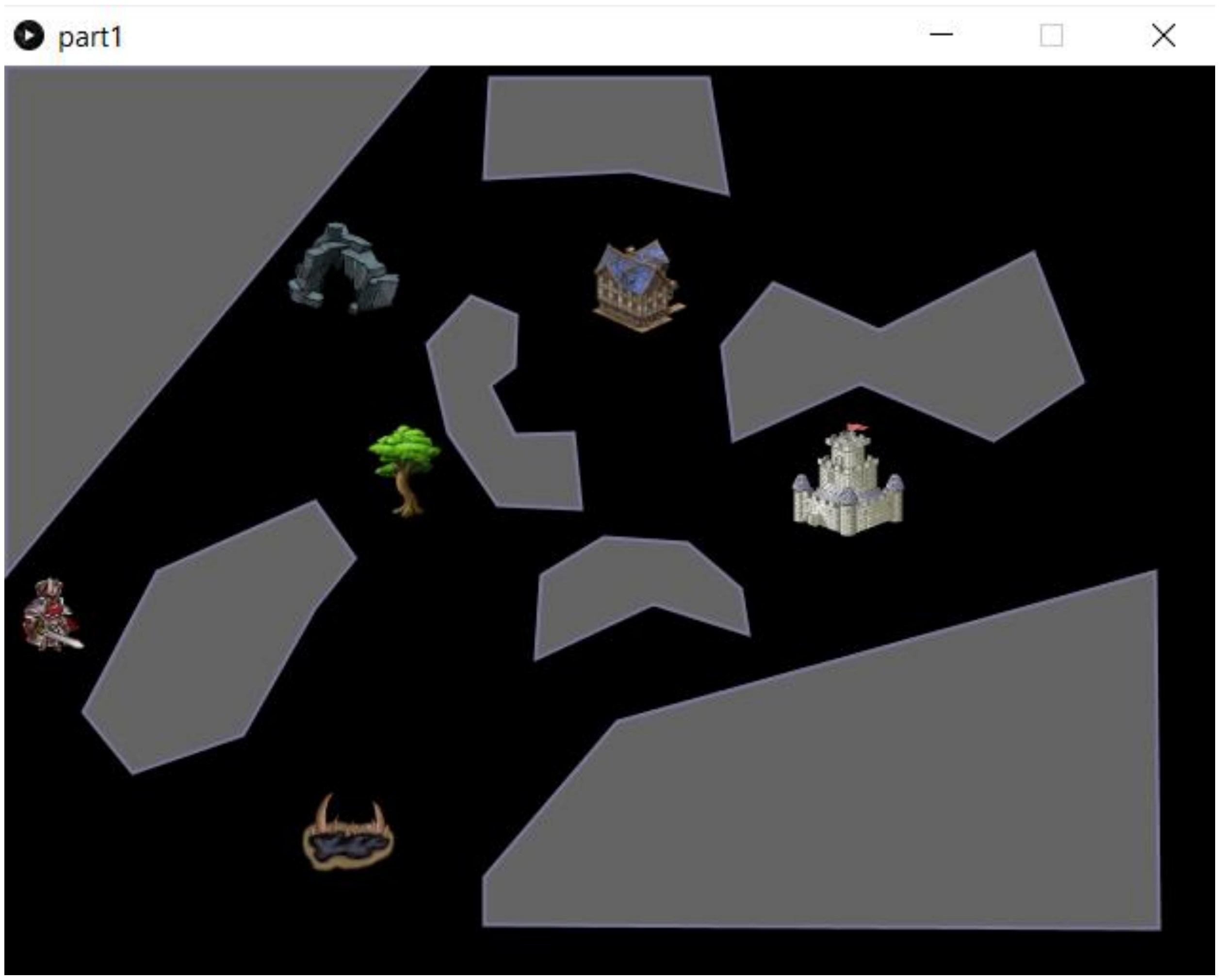


Figure 1: Drawing the World

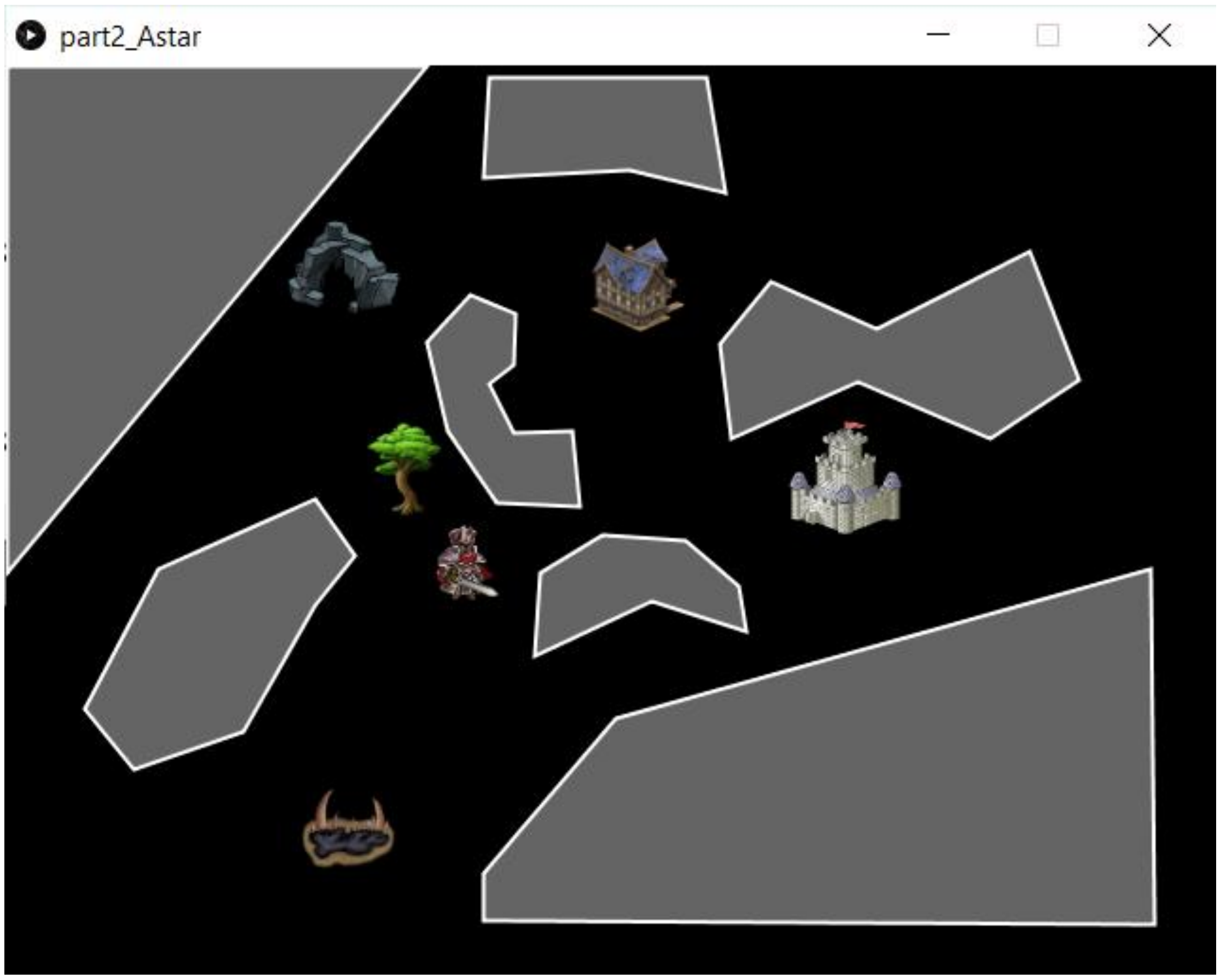
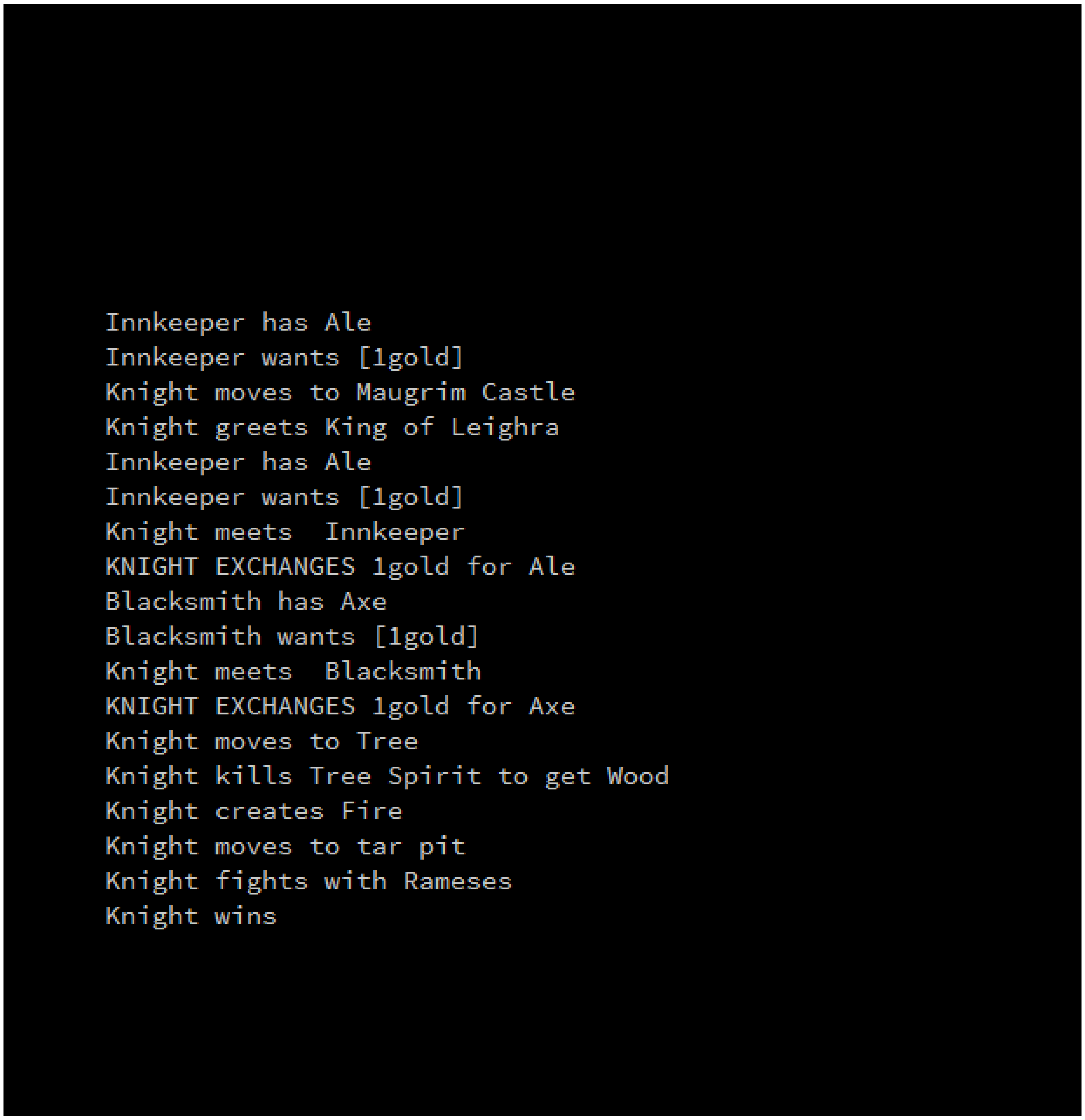


Figure 2: Knight moved to destination



```
Innkeeper has Ale
Innkeeper wants [1gold]
Knight moves to Maugrim Castle
Knight greets King of Leighra
Innkeeper has Ale
Innkeeper wants [1gold]
Knight meets  Innkeeper
KNIGHT EXCHANGES 1gold for Ale
Blacksmith has Axe
Blacksmith wants [1gold]
Knight meets  Blacksmith
KNIGHT EXCHANGES 1gold for Axe
Knight moves to Tree
Knight kills Tree Spirit to get Wood
Knight creates Fire
Knight moves to tar pit
Knight fights with Rameses
Knight wins
```

Figure 3: Actions taken by the knight