CSC 484/584, Spring 2020

Assignment 2: Pathfinding & Decision-making

***Released*:** Thursday, February 6, 2020
***Due*:** Thursday, February 27, before class

Overview

In this assignment, we will build on the skills you have learned in Assignment 1.

Your task for this assignment is to explore some of the pathfinding and decision-making algorithms discussed in class in order to give a virtual agent the appearance of behavior. Working alone and using *Processing* (http://www.processing.org/), you will implement a pathfinding and decision-making algorithm using the scenario described and analyse its performance. You will document your results in a 2–3 page writeup (not including images).

The assignment consists of 3 parts:

1. Drawing a world map according to specification
2. Implementing pathfinding for a virtual agent in the world map
3. Implementing a decision-making algorithm to control the virtual agent's behavior

Scenario

Your virtual agent is a knight in the kingdom of Leighra. They are travelling to Maugrim Castle to meet the king and pay their respects. Once there, they learn that the demonic ram Rameses is terrorizing the kingdom. Only the legendary sword, Fenrir can slay it. Fenrir is guarded by the mysterious Lady Lupa, who will only offer the sword to the one that can bring her the elusive Wolfsbane flower. The only place known to contain Wolfsbane is guarded by a ferocious Tree Spirit, which is withering away for want of water. Handing over a bag of coins, the king advises our knight to stop at the local inn to refresh themselves, before beginning their quest.

Key Locations

Name	Who lives here?
Maugrim Castle	King of Leighra
Tar Pit	The demonic ram Rameses
Ancient Cave	Lady Lupa
Supernatural Forest	Tree Spirit
Tavern	Innkeeper

Key Characters

Name	Located at	Wants	Offers
Knight	[start location]	-	-
King of Leighra	Maugrim Castle	To be greeted	Money
Rameses	Tar Pit	-	-
Lady Lupa	Ancient Cave	Wolfsbane flower	The legendary sword Fenrir
Tree Spirit	Supernatural Forest	Water	Wolfsbane flower
Innkeeper	Tavern	Money	Water

Part 1: Drawing the World (5 pts.)

The first task is to draw the world map according to the scenario described. This involves three parts:

1. Drawing the knight
2. Drawing the key locations
3. Drawing the obstacles

The map must be of size `640 x 480`.

For drawing the knight and key locations, we will provide their initial coordinates as a pair of integer values representing the x-y coordinates of that location. To represent the knight and key locations, you must use a sprite centred around the respective coordinate. The sprites may be an imported image or a custom-designed shape (like the shape used in Assignment 1). Points will not be awarded or deducted for the design of the sprite. However, all sprites must be visually distinct from each other. ***Please include a key in the final report to help us identify which sprite corresponds to which location.***

All obstacles in the game take the form of closed polygons. You will be provided with a list of x-y coordinates representing the corners of the polygon. Obstacles must be drawn by connecting their boundary points to form a closed polygon.

```
beginShape();
vertex(20, 20);
vertex(40, 20);
vertex(40, 40);
vertex(60, 40);
vertex(60, 60);
vertex(20, 60);
endShape(CLOSE);
```

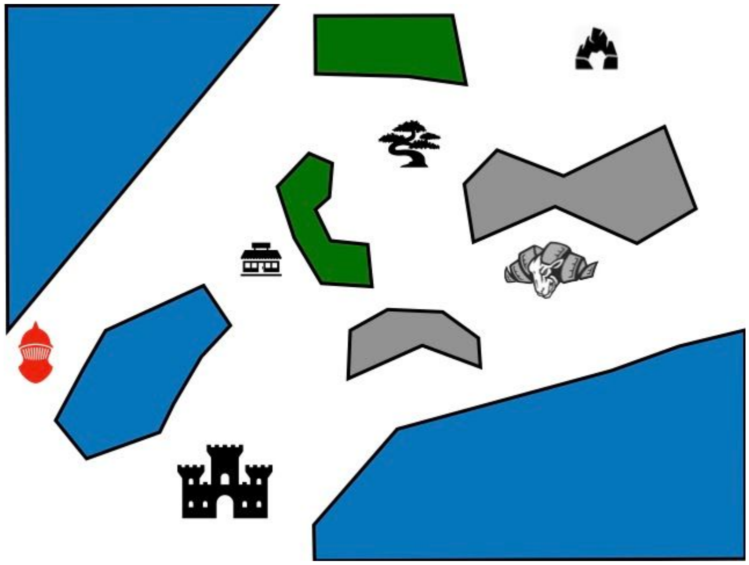
Code #1: Drawing a shape in Processing

Create a single Processing file named `1_drawing_[UnityID].pde` which implements the required functionality. When run, your file should draw all the sprites, locations and obstacles to the screen.

Input for Part 1 of the assignment is provided here. The input is formatted in JSON. You must create a separate JSON file containing the data shown below. Your algorithm will be run against this file and a second hidden file to calculate your final grade.

```
{
        "knight_start": [25, 290],
        "key_locations": {
                "castle": [181, 401],
                "tar_pit": [444, 218],
                "tavern": [211, 214],
                "tree": [334, 116],
                "cave": [487, 41]
        },
        "obstacles": {
                "lake1": [
                        [0,0], [222,0], [0,268]
                ],
                "lake2": [
                        [163,229], [80,266], [41,340], [67,372], [125,352], [163,285], [184,259]
                ],
                "lake3": [
                        [605,266], [322,345], [252,427], [252,452], [607,454]
                ],
                "forest1": [
                        [222,146], [245,121], [269,131], [268,158], [255,168], [268,194], [299,193],
[303,233], [259,231], [233,193]
                ],
                "forest2":[
                        [255,6], [370,6], [380,67], [329,55], [252,59]
                ],
                "mountain1": [
                        [377,147], [404,114], [460,139], [541,98], [567,166], [520,197], [450,167],
[383,197]
                ],
                "mountain2": [
                        [282,268], [315,248], [359,251], [387,275], [391,299], [341,283], [279,312]
                ]
        }
}
```

Sample output image:



Note: Filling colour into the obstacles drawn is optional

Part 2: Pathfinding (25 pts.)

In this part of the assignment, you must implement a pathfinding algorithm in order to have the virtual agent (knight) move to any (valid) target coordinate without passing through obstacles. You must allow for the target coordinate to be selected by mouse click. You may treat the knight as a point object when considering intersections with obstacles. You are free to use any algorithm of your choice for movement.

You will first have to select a world representation in order to obtain a graph representation of the world map. The simplest choice is a grid-based representation. More complex representations like navigation meshes will require greater effort, but result in more natural-looking paths. Describe your chosen implementation in the associated writeup and comment on the resulting output you found. What are the parameters or assumptions made in your algorithm? Are the paths taken by the agent optimal? Do they look natural? What are some of the other improvements that could be made to your algorithm?

Create a Processing file named `2_pathfinding_[UnityID].pde` which implements the required functionality.

Part 3: Decision-making (30 pts.)

In this part of the assignment, you must implement a decision-making algorithm to control the virtual agent's behavior. All other characters will remain static at their respective locations. This part of the assignment will not require any input beyond what is provided in Part 1.

Your program will take as input an initial world state and set of obstacles, given in a JSON file, as in Part 2. You are required to create a JSON file for testing and implement a reader that can parse the contents of this file.

Your decision-making algorithm must find and print (to standard output) a sequence of actions for the knight to take such that the knight attempts to defeat Rameses (successfully or otherwise). For example, at the end of the program, the console might read:

```
-  Knight moves to Maugrim Castle
-  Knight greets King of Leighra.
-  Knight moves to Tavern
-  Innkeeper wants Money
-  Knight wants Water
-  Knight exchanges Money for Water
-  Knight moves to Cave
-  … etc.
```

The knight must greet the king as part of these solutions.

Movement actions must be represented visually by having the knight sprite/shape move to the target location using the pathfinding algorithm you developed in Part 2 of the assignment. Actions other than movement need not be represented visually on the game screen, although they must still be output in the sequence of actions printed.

You may use any decision-making algorithm, but we recommend considering the ones we covered in class, such as behavior trees, goal-oriented action selection mechanism, and goal-oriented action planning. **In your writeup, describe your decision-making procedure in detail and explain why you selected it.** You may wind up trying multiple options, in which case, please document their differences and which you found worked the best. Implement your work for this part in a Processing file named **`3_planning_[UnityID].pde`**.

More details about the action model and world rules allowed are given below.

Possible actions for the knight:

- **Move:** The knight may move from one location to another
- **Exchange:** The knight may exchange an item they possess with another character, provided the other character wants the item and they both are in the same location
    - Example: The Knight can exchange 1 gold for an axe from the blacksmith. This is represented in the changes to the state of the world as follows:
      ```
      Has(Knight, 3Gold), Has(Blacksmith, Axe), Want(Blacksmith, Gold) ==> Has(Knight, Axe), Has(Knight, 2Gold)
      ```
    - Example: The knight can exchange wolfsbane with Lady Lupa for Fenrir:
      ```
      Has(Knight, Wolfsbane), Has(Lady Lupa, Fenrir), Want(Lady Lupa, Wolfsbane) ==> Has(Knight, Fenrir), !Has(Knight, Wolfsbane), !Has(Lady Lupa, Fenrir)
      ```
- **Greet:** The knight can greet the king if the knight and king are both in the same location.
- **Use:**
    - The knight can combine two items to make a third item (consuming the original items).
        - Example, **Use(Blade, Wood) = Cheap Sword,** implies the knight will lose (or consume) the Blade and the Wood from their inventory but obtain (or produce) the Cheap Sword. This is represented in the state of the world as follows:
      ```
      Has(Knight, Blade), Has(Knight, Wood) ⇒ Has(Knight, Cheap Sword), !Has(Knight, Blade), !Has(Knight, Wood)
      ```
    - The knight can also use an item on a person or place.
        - Example, Use(Axe, Tree Spirit) = Wood OR Use(Fenrir, Rameses)

- **Fight:** The knight can fight Rameses if the knight and Rameses are in the same location.

A note about "Wants:" The NPC wants do not expire. You can get unlimited items from the NPCs in exchange for what they want. However, there is ONLY ONE Fenrir.

The conditions under which the knight may use their items to defeat Rameses or be defeated have been listed in the **World Rules** below. The knight must be in the same location as the place, or person to exchange items. The knight must have both items in order to use items.

Rules for Use/Exchange:

- Greet + King = Get Gold (read JSON file to find out how much gold)
- Blacksmith + 1 gold = axe
- Axe + Tree Spirit = Wood
- Blacksmith + 1 gold = Blade
- Blade + Wood = Cheap Sword

- Sword + Wolfsbane = Poisoned Sword
- Fenrir + Wolfsbane = Poisoned Fenrir
- Poisoned Sword + Fight Rameses = Defeat Rameses
- Sword + Fight Rameses = Dead Knight
- Fenrir + Fight Rameses = Defeat Rameses
- Poisoned Fenrir + Fight Rameses = Defeat Rameses
- Innkeeper + 1 gold = water
- Innkeeper + 1 gold = ale
- Ale + wood = fire
- Fire + Rameses = defeat Rameses (burns the tar-pits)
- Lady Lupa + Wolfsbane = Fenrir
- Tree Spirit + Water = Wolfsbane
- Any other item + Fight Rameses = Dead Knight (eg. Gold + Fight Rameses = Dead Knight)

Note: The TAs will use a hidden JSON file that may have a different map, state of the world, or starting conditions for grading. Your implementation must be robust enough to handle an arbitrary file in this format. You are free to create your own JSON files and share them with your classmates on Piazza to test your code.

JSON File for testing Part 3:

```
{
        "knight_start": [25, 290],
        "key_locations": {
                "castle": [181, 401],
                "tar_pit": [444, 218],
                "tavern": [211, 214],
                "tree": [334, 116],
                "cave": [487, 41],
                "forge": [179, 107]
        },
        “greet_king”: 5,
        “state_of_world”: {
                “Has”: [
                        [“Blacksmith”, “Axe”],
                        [“Blacksmith”, “Blade”],
                        [“Lady Lupa”, “Fenrir”],
                        [“Innkeeper”, “Water”],
                        [“Innkeeper”, “Ale”],
                        [“Tree Spirit”, “Wolfsbane”],
                ],
                “Wants” : [
                        [“Blacksmith”, “1gold”],
                        [“Lady Lupa”, “Wolfsbane”],
                        [“Tree Spirit”, “Water”],
                        [“Innkeeper”, “1gold”]
                ]
        },
        "obstacles": {
                "lake1": [
                        [0,0], [222,0], [0,268]
                ],
                "lake2": [
                        [163,229], [80,266], [41,340], [67,372], [125,352], [163,285], [184,259]
                ],
                "lake3": [
                        [605,266], [322,345], [252,427], [252,452], [607,454]
                ],
                "forest1": [
                        [222,146], [245,121], [269,131], [268,158], [255,168], [268,194], [299,193],
[303,233], [259,231], [233,193]
                ],
                "forest2":[
                        [255,6], [370,6], [380,67], [329,55], [252,59]
                ],
                "mountain1": [
                        [377,147], [404,114], [460,139], [541,98], [567,166], [520,197], [450,167],
[383,197]
                ],
                "mountain2": [
                        [282,268], [315,248], [359,251], [387,275], [391,299], [341,283], [279,312]
                ]
        }
}
```

Writeup (40 pts.)

Now that you have implemented and evaluated a number of algorithms, write a 2–3 page single-spaced paper summarizing your findings. That means *at least* two full pages. It is strongly suggested that you do not limit yourself to only answering those questions posed in this assignment. Think creatively about what you have done. What other parameters can you tweak and what effect do they have on the results? The most successful writeups will contain evidence that you have thought deeply about these algorithms and what they can produce and have gone beyond the first thing that worked.

Your writeup should have at least two sections, one each for your analysis of Part 2 and Part 3. Please include all relevant screenshots to support your analysis. These screenshots do not count towards your 2–3 page requirement.

What to submit

Gradescope will have ***two submission portals*** for Assignment 2 as listed below. By the start of class on 02/27/2020, please upload the following to Gradescope.

1. **Assignment 2 - Writeup:** Upload your writeup (as a PDF) to Gradescope, and select the pages where you describe your efforts for each part of the assignment.
2. **Assignment 2 - Code:** Upload your code as a `.zip` archive to Gradescope. This archive should contain your 3 labeled `.pde` files from each part of the assignment. You

should also include a single README containing a description of your files, and any specific instructions for how to compile and run your code.