IT University of Copenhagen, 2023

Foundations of Game AI

Exercise Session 1

Introduction & General Info

Welcome to all.

Focus of the Labs

The labs for this course will revolve around implementing in Python code the algorithms and techniques seen in the lectures. We will use Pac-man game. This is also the game that will be used for the Mandatory Hand-ins 1 and 2 (you can find more information about them on the Learn-it page for this course). By completing all the exercises in each lab, you will be well prepared to comfortably work on the Hand-ins.

Structure of the Labs

Except for a few labs in which the TA will explain the code live, most of the other Labs will be structured in a way that a PDF will be shared, and you will be free to do the exercises in class or at your own pace. The exercises are a mixture of actual exercises for you to do, and walkthroughs for you to understand the code, depending on what's appropriate for the topic.

Solutions are also shared with the code at the beginning of the Labs.

The code we will work with in our exercise sessions as well as for both hand-ins can be found here. Alternatively, you can find the same link and a donwloadable folder on LearnIT under "Mandatory Han-ind 1 - Basic Pac-Man Controller".

This site explains step by step how to create a fully playable Pac-Man game: for the scope of this course, we are only interested in the algorithms for controlling the characters (rather than the graphic aspects). Therefore, we will go through the explanations on the site but we will also ignore some less-relevant aspects (which will make it much simpler to navigate, explain and understand the code).

Today's topic

For this session, we want to create two simple controllers: one for Pac-Man and one for the ghost. This substitutes the key-press detection mechanism that is used in the code from the site i.e. instead of waiting for player's input, we will make Pac-Man move on its own.

These two controllers will have a fleeing and seeking behaviour respectively, as in Chapter 3.2 of the textbook.

We start by following the walk-through on the Pac-man website. As mentioned earlier, we will ignore some aspects. Thus, you can either implement the full Pac-man game by following the steps on the site (or downloading the code from LearnIT), or you can follow the simplified version that we will implement in this lab.

Thus, **if you want to skip to the controllers' implementation, go to Exercise 7.**

Exercise 1

Vectors are essential in game development. They offer a simple yet complete way of representing a character's movement in space. Head over to "Start: Vectors" page and quickly look at the various methods. You are not expected to memorize any of them, but understand what each of them does.

Implement the code as described on the web page.

Exercise 2

Let's implement the game screen. Head to "Start: Blank Screen" page and follow the explanation. Creating a game screen and rendering objects to it is not in the scope of our course, but every game needs such features to function. Thus, it's good knowledge to understand how it works.

Implement the code as described on the web page.

Exercise 3

Now, head to the next section "Start: Basic Movement". Make sure you understand this section well. In particular, pay close attention to the "update()" method in pacman.py

```
def update(self, dt):
        self.position += self.directions[self.direction]*self.speed*dt
        direction = self.getValidKey()
        self.direction = direction
```

Look at the second line.

- Why is "self.position" defined and updated using the above multiplication?
- What happens if we don't use the dt parameter?

Implement the code as described on the web page.

Exercise 4

To move our characters around the maze, we will create a system of nodes connected by edges and we will make Pac-Man and the ghost walk along these edges. This is a much quicker and easier to implement alternative to wall-collision detection.

Head to the "Level 1: Nodes" page and implement the code. Then, continue to the following Part 1, Part 2 and Part 3. Implement the code as described on the web page.

You are not expected to know how to replicate this nodes-system, but understanding this will help with creating the controllers for our characters.

After finishing the implementation, look again at the update() method in pacman.py.

- Why do we check that self.target is not self.node?
- How does this affect the movement?
- What happens if self.overshotTarget() is false? When could this happen?

Exercise 5

Go through and implement the code in “Level 2: Maze Basics” and “Level 2: PacMan Maze”.

This should be pretty straightforward: instead of manually hard-coding the mazes, we create a parses that does the job for us.

- Do both Pac-Man and the ghost start from the same node?
- How would you change it so that they start from different nodes?

DO NOT implement the code in Portals, Pellets or Eating Pellets pages if you want the simplified version of the Pac-man game.

Exercise 6

Head to “Level 3: Ghost intro” page and “Level 3: Ghost setup” page. Once again, follow through with the explanation and implement the code.

Before moving on with the next (the most important) section, make you understand the key differences between Pac-Man and ghost.

(Because we skipped some parts, there are superfluous methods that will cause errors. Delete them when flagged by the compiler).

Exercise 7

Finally, implement the code in the “Level 3: Ghost AI” page. This is the last page we will use from the site. Pay close attention to the goalDirection() method, as this is the key to defining the behaviour of each character.

Would you be able to explain in plain English what is happening in line 4?

```
vec = self.node.position  + self.directions[direction]*TILEWIDTH - self.goal
```

Exercise 8

After implementing the you'll see the ghost circle the top left corner of the screen continuously. This is because we are passing Vector2() to self.goal. This creates a vector with coordinates (0,0), meaning that the ghost is trying to go to the (0,0) position on the screen, which is the top-left corner.

Now, change the self.goalDirection() function so that the ghost chases PacMan.

Exercise 8

In the previous exercise you implemented the seek (or chase) behaviour for the ghost. You should now have a good understanding of how the code is structured and how the game works. Now, implement the fleeing behaviour for Pac-Man. This is the key point of this exercise session.