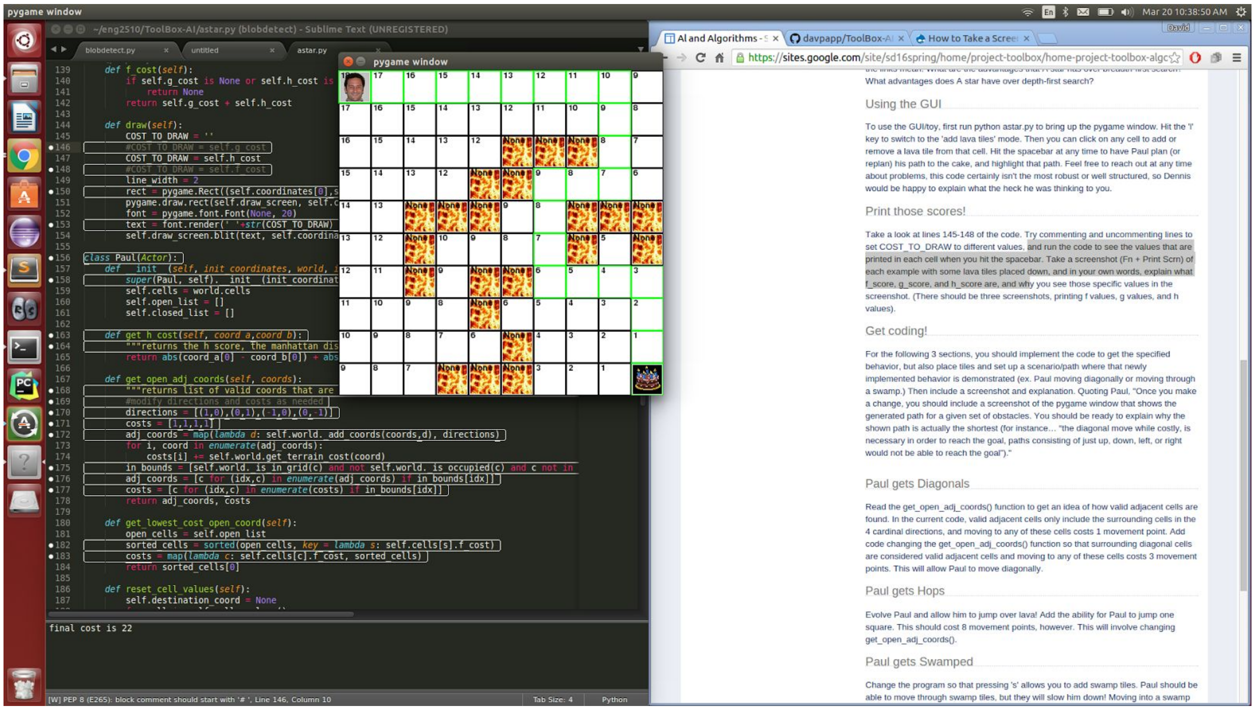
David Papp

H: J is the distance between our chosen position and the target. It is square root of the sum of the squares of the horizontal and vertical differences in distance. In our picture, H has a linear gradient that decreases in the direction towards the target.



pygame window

~/eng2510/ToolBox-AI/astar.py (blobdetect) - Sublime Text (UNREGISTERED)

```
139     def f_cost(self):
140         if self.g_cost is None or self.h_cost is
141             return None
142         return self.g_cost + self.h_cost
143
144     def draw(self):
145         COST_TO_DRAW = ''
146         #COST_TO_DRAW = self.g_cost
147         COST_TO_DRAW = self.h_cost
148         #COST_TO_DRAW = self.f_cost
149         line_width = 2
150         rect = pygame.Rect((self.coordinates[0],s
151         pygame.draw.rect(self.draw_screen, self.c
152         font = pygame.font.Font(None, 20)
153         text = font.render(' '+str(COST_TO_DRAW)
154         self.draw_screen.blit(text, self.coordina
155
156 class Paul(Actor):
157     def __init__(self, init_coordinates, world, i
158         super(Paul, self).__init__(init_coordinat
159         self.cells = world.cells
160         self.open_list = []
161         self.closed_list = []
162
163     def get_h_cost(self, coord_a,coord_b):
164         """returns the h score, the manhattan dis
165         return abs(coord_a[0] - coord_b[0]) + abs
166
167     def get_open_adj_coords(self, coords):
168         """returns list of valid coords that are
169         #modify directions and costs as needed
170         directions = [(1,0),(0,1),(-1,0),(0,-1)]
171         costs = [1,1,1,1]
172         adj_coords = map(lambda d: self.world._add_coords(coords,d), directions)
173         for i, coord in enumerate(adj_coords):
174             costs[i] += self.world.get_terrain_cost(coord)
175         in_bounds = [self.world._is_in_grid(c) and not self.world._is_occupied(c) and c not in
176         adj_coords = [c for (idx,c) in enumerate(adj_coords) if in_bounds[idx]]
177         costs = [c for (idx,c) in enumerate(costs) if in_bounds[idx]]
178         return adj_coords, costs
179
180     def get_lowest_cost_open_coord(self):
181         open_cells = self.open_list
182         sorted_cells = sorted(open_cells, key = lambda s: self.cells[s].f_cost)
183         costs = map(lambda c: self.cells[c].f_cost, sorted_cells)
184         return sorted_cells[0]
185
186     def reset_cell_values(self):
187         self.destination_coord = None
```

```
final cost is 22
```

[W] PEP 8 (E265): block comment should start with '# ', Line 146, Column 10

pygame window

18	17	16	15	14	13	12	11	10	9
17	16	15	14	13	12	11	10	9	8
16	15	14	13	12	None	None	None	8	7
15	14	13	12	None	None	9	8	7	6
14	13	None	None	None	9	8	None	None	None
13	12	None	10	9	8	7	None	5	None
12	11	None	9	None	None	6	5	4	3
11	10	9	8	None	6	5	4	3	2
10	9	8	7	6	None	4	3	2	1
9	8	7	None	None	None	3	2	1	

Mar 20 10:38:50 AM

AI and Algorithms - S | davpapp/ToolBox-AI | How to Take a Scree

https://sites.google.com/site/sd16spring/home/project-toolbox/home-project-toolbox-algo

the links mean. What are the advantages that A star has over breadth-first search? What advantages does A star have over depth-first search?

Using the GUI

To use the GUI/toy, first run python astar.py to bring up the pygame window. Hit the 'l' key to switch to the 'add lava tiles' mode. Then you can click on any cell to add or remove a lava tile from that cell. Hit the spacebar at any time to have Paul plan (or replan) his path to the cake, and highlight that path. Feel free to reach out at any time about problems, this code certainly isn't the most robust or well structured, so Dennis would be happy to explain what the heck he was thinking to you.

Print those scores!

Take a look at lines 145-148 of the code. Try commenting and uncommenting lines to set COST_TO_DRAW to different values, and run the code to see the values that are printed in each cell when you hit the spacebar. Take a screenshot (Fn + Print Scrn) of each example with some lava tiles placed down, and in your own words, explain what f_score, g_score, and h_score are, and why you see those specific values in the screenshot. (There should be three screenshots, printing f values, g values, and h values).

Get coding!

For the following 3 sections, you should implement the code to get the specified behavior, but also place tiles and set up a scenario/path where that newly implemented behavior is demonstrated (ex. Paul moving diagonally or moving through a swamp.) Then include a screenshot and explanation. Quoting Paul, "Once you make a change, you should include a screenshot of the pygame window that shows the generated path for a given set of obstacles. You should be ready to explain why the shown path is actually the shortest (for instance... "the diagonal move while costly, is necessary in order to reach the goal, paths consisting of just up, down, left, or right would not be able to reach the goal")."

Paul gets Diagonals

Read the get_open_adj_coords() function to get an idea of how valid adjacent cells are found. In the current code, valid adjacent cells only include the surrounding cells in the 4 cardinal directions, and moving to any of these cells costs 1 movement point. Add code changing the get_open_adj_coords() function so that surrounding diagonal cells are considered valid adjacent cells and moving to any of these cells costs 3 movement points. This will allow Paul to move diagonally.

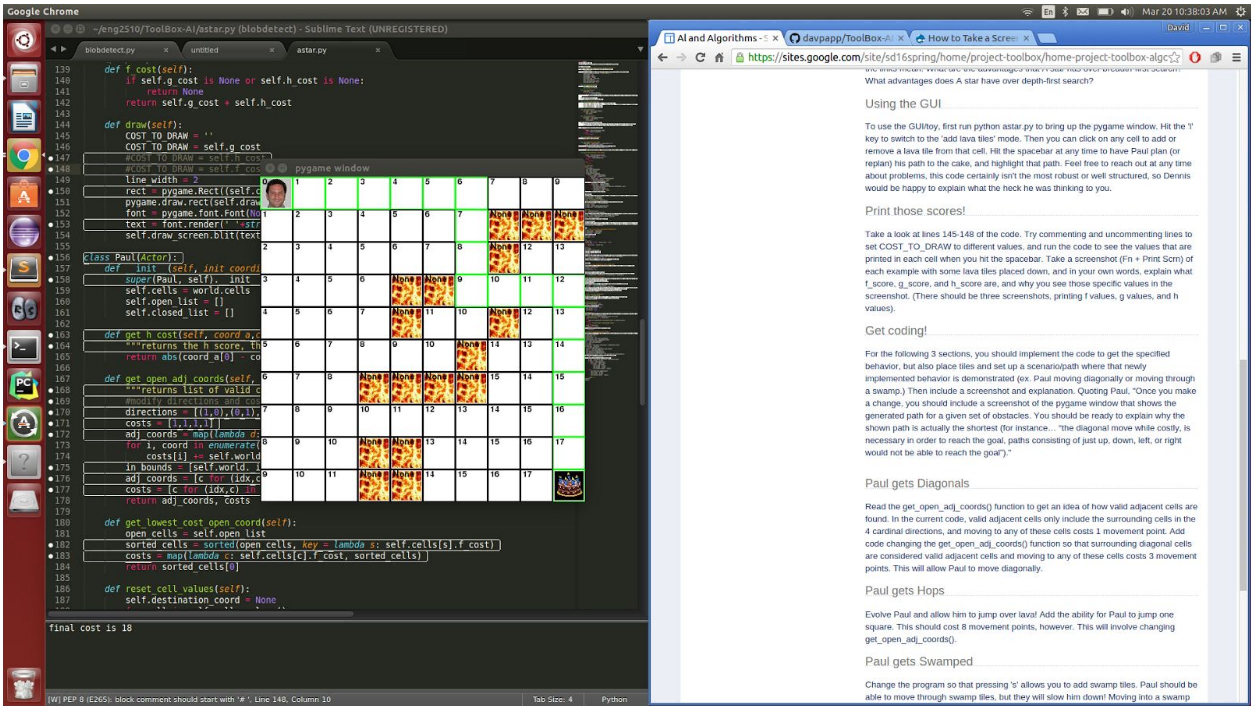
Paul gets Hops

Evolve Paul and allow him to jump over lava! Add the ability for Paul to jump one square. This should cost 8 movement points, however. This will involve changing get_open_adj_coords().

Paul gets Swamped

Change the program so that pressing 's' allows you to add swamp tiles. Paul should be able to move through swamp tiles, but they will slow him down! Moving into a swamp

G: G is the cost of moving between two positions. It is based on the minimum cost of its parent nodes. In the picture, it’s evident that the cost increases when taking the way around obstacles.



Google Chrome

~/eng2510/ToolBox-AI/astar.py (blobdetect) - Sublime Text (UNREGISTERED)

```
139     def f_cost(self):
140         if self.g_cost is None or self.h_cost is None:
141             return None
142         return self.g_cost + self.h_cost
143
144     def draw(self):
145         COST_TO_DRAW = ''
146         COST_TO_DRAW = self.g_cost
147         #COST_TO_DRAW = self.h_cost
148         #COST_TO_DRAW = self.f_cos
149         line_width = 2
150         rect = pygame.Rect((self.c
151         pygame.draw.rect(self.draw
152         font = pygame.font.Font(No
153         text = font.render(' '+str
154         self.draw_screen.blit(text
155
156 class Paul(Actor):
157     def __init__(self, init_coordi
158         super(Paul, self).__init__
159         self.cells = world.cells
160         self.open_list = []
161         self.closed_list = []
162
163     def get_h_cost(self, coord_a,c
164         """returns the h score, th
165         return abs(coord_a[0] - co
166
167     def get_open_adj_coords(self,
168         """returns list of valid c
169         #modify directions and cos
170         directions = [(1,0),(0,1),
171         costs = [1,1,1,1]
172         adj_coords = map(lambda d:
173         for i, coord in enumerate(
174             costs[i] += self.world
175         in_bounds = [self.world._i
176         adj_coords = [c for (idx,c
177         costs = [c for (idx,c) in
178         return adj_coords, costs
179
180     def get_lowest_cost_open_coord(self):
181         open_cells = self.open_list
182         sorted_cells = sorted(open_cells, key = lambda s: self.cells[s].f_cost)
183         costs = map(lambda c: self.cells[c].f_cost, sorted_cells)
184         return sorted_cells[0]
185
186     def reset_cell_values(self):
187         self.destination_coord = None
```

```
final cost is 18
```

[W] PEP 8 (E265): block comment should start with '# ', Line 148, Column 10

pygame window

0	1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	None	None	None
2	3	4	5	6	7	8	None	12	13
3	4	5	6	None	None	9	10	11	12
4	5	6	7	None	11	10	None	12	13
5	6	7	8	9	10	None	14	13	14
6	7	8	None	None	None	None	15	14	15
7	8	9	10	11	12	13	14	15	16
8	9	10	None	None	13	14	15	16	17
9	10	11	None	None	14	15	16	17	

Mar 20 10:38:03 AM

AI and Algorithms - S | davpapp/ToolBox-AI | How to Take a Scree

https://sites.google.com/site/sd16spring/home/project-toolbox/home-project-toolbox-algo

the links mean. What are the advantages that A star has over breadth-first search? What advantages does A star have over depth-first search?

Using the GUI

To use the GUI/toy, first run python astar.py to bring up the pygame window. Hit the 'l' key to switch to the 'add lava tiles' mode. Then you can click on any cell to add or remove a lava tile from that cell. Hit the spacebar at any time to have Paul plan (or replan) his path to the cake, and highlight that path. Feel free to reach out at any time about problems, this code certainly isn't the most robust or well structured, so Dennis would be happy to explain what the heck he was thinking to you.

Print those scores!

Take a look at lines 145-148 of the code. Try commenting and uncommenting lines to set COST_TO_DRAW to different values, and run the code to see the values that are printed in each cell when you hit the spacebar. Take a screenshot (Fn + Print Scrn) of each example with some lava tiles placed down, and in your own words, explain what f_score, g_score, and h_score are, and why you see those specific values in the screenshot. (There should be three screenshots, printing f values, g values, and h values).

Get coding!

For the following 3 sections, you should implement the code to get the specified behavior, but also place tiles and set up a scenario/path where that newly implemented behavior is demonstrated (ex. Paul moving diagonally or moving through a swamp.) Then include a screenshot and explanation. Quoting Paul, "Once you make a change, you should include a screenshot of the pygame window that shows the generated path for a given set of obstacles. You should be ready to explain why the shown path is actually the shortest (for instance... "the diagonal move while costly, is necessary in order to reach the goal, paths consisting of just up, down, left, or right would not be able to reach the goal")."

Paul gets Diagonals

Read the get_open_adj_coords() function to get an idea of how valid adjacent cells are found. In the current code, valid adjacent cells only include the surrounding cells in the 4 cardinal directions, and moving to any of these cells costs 1 movement point. Add code changing the get_open_adj_coords() function so that surrounding diagonal cells are considered valid adjacent cells and moving to any of these cells costs 3 movement points. This will allow Paul to move diagonally.

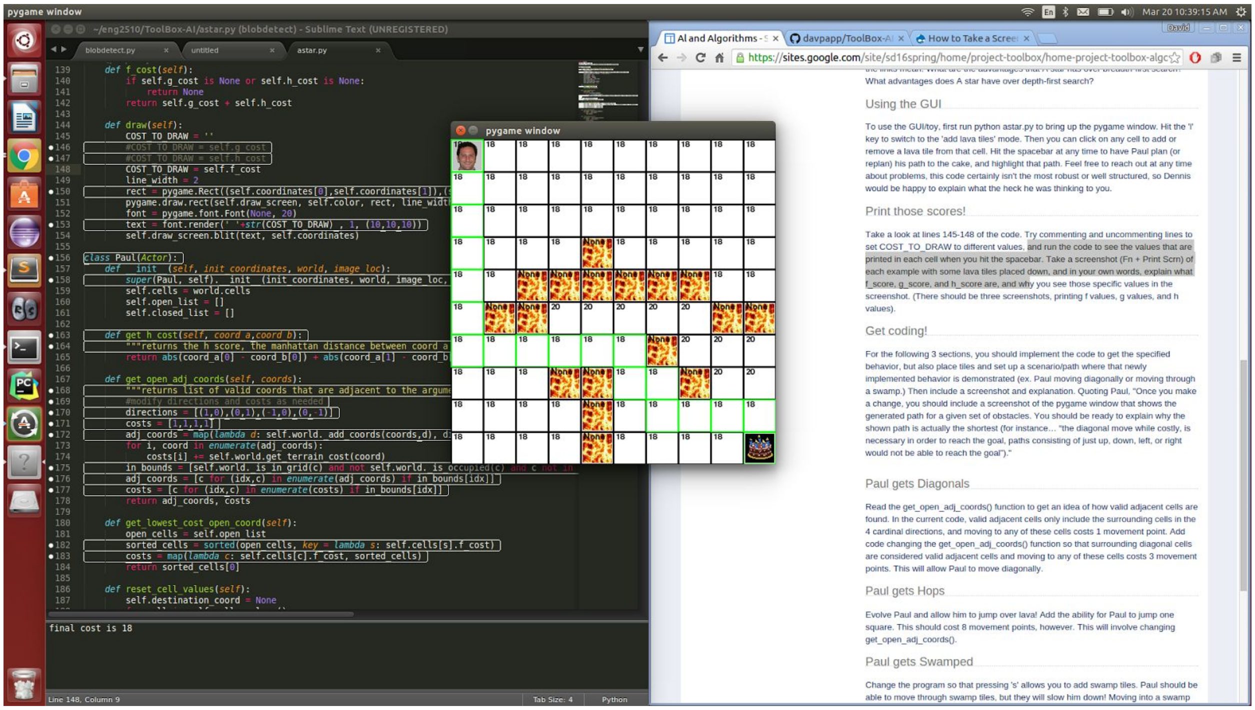
Paul gets Hops

Evolve Paul and allow him to jump over lava! Add the ability for Paul to jump one square. This should cost 8 movement points, however. This will involve changing get_open_adj_coords().

Paul gets Swamped

Change the program so that pressing 's' allows you to add swamp tiles. Paul should be able to move through swamp tiles, but they will slow him down! Moving into a swamp

F: F is the sum of the H and G. The minimum path lies where the F values are the least.


pygame window
~/eng2510/ToolBox-AI/astar.py (blobdetect) - Sublime Text (UNREGISTERED)
blobdetect.py
untitled
astar.py
Line 148, Column 9
Tab Size: 4
Python
AI and Algorithms - S
davpapp/ToolBox-AI
How to Take a Scree
https://sites.google.com/site/sd16spring/home/project-toolbox/home-project-toolbox-algc
Mar 20 10:39:15 AM
David

```python
139     def f_cost(self):
140         if self.g_cost is None or self.h_cost is None:
141             return None
142         return self.g_cost + self.h_cost
143 
144     def draw(self):
145         COST_TO_DRAW = ''
146         #COST_TO_DRAW = self.g_cost
147         #COST_TO_DRAW = self.h_cost
148         COST_TO_DRAW = self.f_cost
149         line_width = 2
150         rect = pygame.Rect((self.coordinates[0],self.coordinates[1]),(s
151         pygame.draw.rect(self.draw_screen, self.color, rect, line_widt
152         font = pygame.font.Font(None, 20)
153         text = font.render(' '+str(COST_TO_DRAW) , 1, (10,10,10))
154         self.draw_screen.blit(text, self.coordinates)
155 
156 class Paul(Actor):
157     def __init__(self, init_coordinates, world, image_loc):
158         super(Paul, self).__init__(init_coordinates, world, image_loc,
159         self.cells = world.cells
160         self.open_list = []
161         self.closed_list = []
162 
163     def get_h_cost(self, coord_a,coord_b):
164         """returns the h score, the manhattan distance between coord a
165         return abs(coord_a[0] - coord_b[0]) + abs(coord_a[1] - coord_b
166 
167     def get_open_adj_coords(self, coords):
168         """returns list of valid coords that are adjacent to the argume
169         #modify directions and costs as needed
170         directions = [(1,0),(0,1),(-1,0),(0,-1)]
171         costs = [1,1,1,1]
172         adj_coords = map(lambda d: self.world._add_coords(coords,d), di
173         for i, coord in enumerate(adj_coords):
174             costs[i] += self.world.get_terrain_cost(coord)
175         in_bounds = [self.world._is_in_grid(c) and not self.world._is_occupied(c) and c not in
176         adj_coords = [c for (idx,c) in enumerate(adj_coords) if in_bounds[idx]]
177         costs = [c for (idx,c) in enumerate(costs) if in_bounds[idx]]
178         return adj_coords, costs
179 
180     def get_lowest_cost_open_coord(self):
181         open_cells = self.open_list
182         sorted_cells = sorted(open_cells, key = lambda s: self.cells[s].f_cost)
183         costs = map(lambda c: self.cells[c].f_cost, sorted_cells)
184         return sorted_cells[0]
185 
186     def reset_cell_values(self):
187         self.destination_coord = None
```

```
final cost is 18
```

pygame window

18	18	18	18	18	18	18	18	18	18
18	18	18	18	18	18	18	18	18	18
18	18	18	18	18	18	18	18	18	18
18	18	18	18	None	18	18	18	18	18
18	18	None	None	None	None	None	None	18	18
18	None	None	20	20	20	20	20	None	None
18	18	18	18	18	18	None	20	20	20
18	18	18	None	None	18	18	None	20	20
18	18	18	18	None	18	18	18	18	18
18	18	18	18	None	18	18	18	18	

the links mean. What are the advantages that A star has over breadth-first search? What advantages does A star have over depth-first search?

Using the GUI

To use the GUI/toy, first run python astar.py to bring up the pygame window. Hit the 'l' key to switch to the 'add lava tiles' mode. Then you can click on any cell to add or remove a lava tile from that cell. Hit the spacebar at any time to have Paul plan (or replan) his path to the cake, and highlight that path. Feel free to reach out at any time about problems, this code certainly isn't the most robust or well structured, so Dennis would be happy to explain what the heck he was thinking to you.

Print those scores!

Take a look at lines 145-148 of the code. Try commenting and uncommenting lines to set COST_TO_DRAW to different values, and run the code to see the values that are printed in each cell when you hit the spacebar. Take a screenshot (Fn + Print Scrn) of each example with some lava tiles placed down, and in your own words, explain what f_score, g_score, and h_score are, and why you see those specific values in the screenshot. (There should be three screenshots, printing f values, g values, and h values).

Get coding!

For the following 3 sections, you should implement the code to get the specified behavior, but also place tiles and set up a scenario/path where that newly implemented behavior is demonstrated (ex. Paul moving diagonally or moving through a swamp.) Then include a screenshot and explanation. Quoting Paul, "Once you make a change, you should include a screenshot of the pygame window that shows the generated path for a given set of obstacles. You should be ready to explain why the shown path is actually the shortest (for instance… "the diagonal move while costly, is necessary in order to reach the goal, paths consisting of just up, down, left, or right would not be able to reach the goal")."

Paul gets Diagonals

Read the get_open_adj_coords() function to get an idea of how valid adjacent cells are found. In the current code, valid adjacent cells only include the surrounding cells in the 4 cardinal directions, and moving to any of these cells costs 1 movement point. Add code changing the get_open_adj_coords() function so that surrounding diagonal cells are considered valid adjacent cells and moving to any of these cells costs 3 movement points. This will allow Paul to move diagonally.

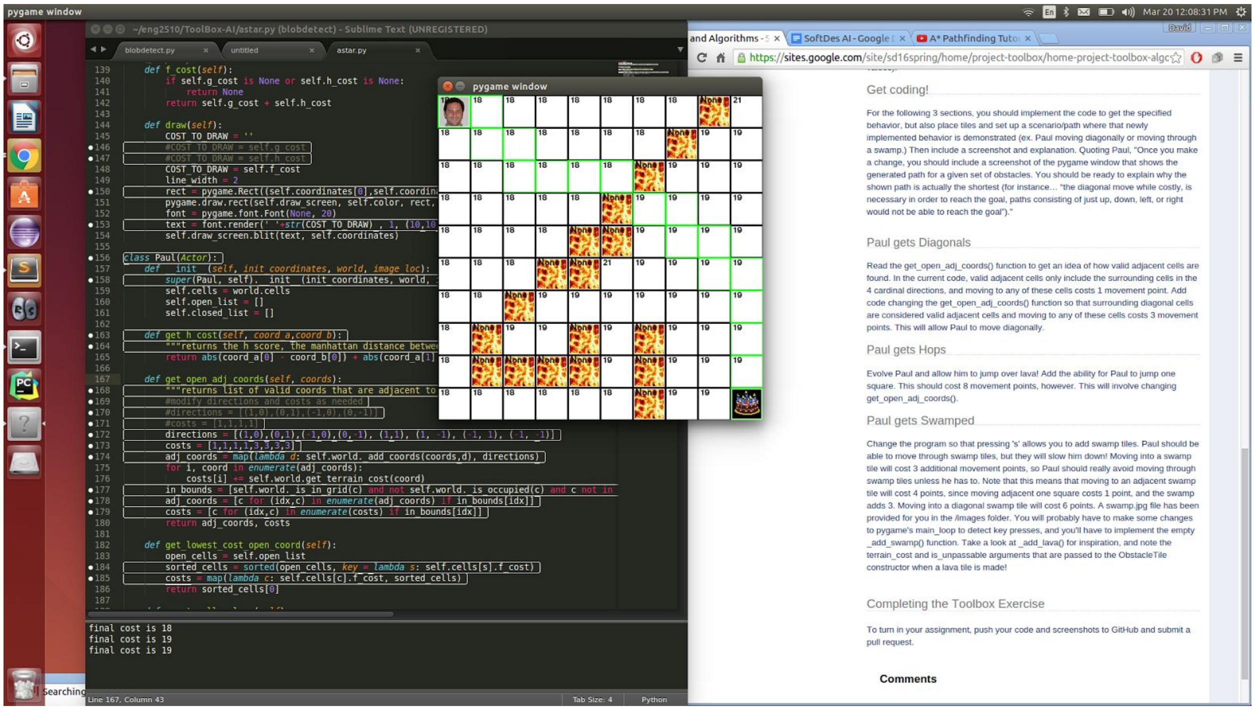
Paul gets Hops

Evolve Paul and allow him to jump over lava! Add the ability for Paul to jump one square. This should cost 8 movement points, however. This will involve changing get_open_adj_coords().

Paul gets Swamped

Change the program so that pressing 's' allows you to add swamp tiles. Paul should be able to move through swamp tiles, but they will slow him down! Moving into a swamp

Diagonal:
In this case, avoiding diagonals forces Paul to take the long excursion in the bottom-left corner.
It is quicker to jump over the diagonal.



```python
139     def f_cost(self):
140         if self.g_cost is None or self.h_cost is None:
141             return None
142         return self.g_cost + self.h_cost
143 
144     def draw(self):
145         COST_TO_DRAW = ''
146         #COST_TO_DRAW = self.g_cost
147         #COST_TO_DRAW = self.h_cost
148         COST_TO_DRAW = self.f_cost
149         line_width = 2
150         rect = pygame.Rect((self.coordinates[0],self.coordin
151         pygame.draw.rect(self.draw_screen, self.color, rect,
152         font = pygame.font.Font(None, 20)
153         text = font.render(' '+str(COST_TO_DRAW) , 1, (10,10
154         self.draw_screen.blit(text, self.coordinates)
155 
156 class Paul(Actor):
157     def __init__(self, init_coordinates, world, image_loc):
158         super(Paul, self).__init__(init_coordinates, world,
159         self.cells = world.cells
160         self.open_list = []
161         self.closed_list = []
162 
163     def get_h_cost(self, coord_a,coord_b):
164         """returns the h score, the manhattan distance betwe
165         return abs(coord_a[0] - coord_b[0]) + abs(coord_a[1]
166 
167     def get_open_adj_coords(self, coords):
168         """returns list of valid coords that are adjacent to
169         #modify directions and costs as needed
170         #directions = [(1,0),(0,1),(-1,0),(0,-1)]
171         #costs = [1,1,1,1]
172         directions = [(1,0),(0,1),(-1,0),(0,-1), (1,1), (1, -1), (-1, 1), (-1, -1)]
173         costs = [1,1,1,1,3,3,3,3]
174         adj_coords = map(lambda d: self.world._add_coords(coords,d), directions)
175         for i, coord in enumerate(adj_coords):
176             costs[i] += self.world.get_terrain_cost(coord)
177         in_bounds = [self.world._is_in_grid(c) and not self.world._is_occupied(c) and c not in
178         adj_coords = [c for (idx,c) in enumerate(adj_coords) if in_bounds[idx]]
179         costs = [c for (idx,c) in enumerate(costs) if in_bounds[idx]]
180         return adj_coords, costs
181 
182     def get_lowest_cost_open_coord(self):
183         open_cells = self.open_list
184         sorted_cells = sorted(open_cells, key = lambda s: self.cells[s].f_cost)
185         costs = map(lambda c: self.cells[c].f_cost, sorted_cells)
186         return sorted_cells[0]
187 
```

```
final cost is 18
final cost is 19
final cost is 19
```

pygame window

18	18	18	18	18	18	18	18	None	21
18	18	18	18	18	18	18	None	19	19
18	18	18	18	18	18	None	19	19	19
18	18	18	18	18	None	19	19	19	19
18	18	18	18	None	None	19	19	19	19
18	18	18	None	None	21	19	19	19	19
18	18	None	19	19	19	19	19	19	19
18	None	19	19	None	19	None	19	19	19
18	None	None	None	None	19	None	19	19	19
18	18	18	18	18	18	None	19	19	

Get coding!

For the following 3 sections, you should implement the code to get the specified behavior, but also place tiles and set up a scenario/path where that newly implemented behavior is demonstrated (ex. Paul moving diagonally or moving through a swamp.) Then include a screenshot and explanation. Quoting Paul, "Once you make a change, you should include a screenshot of the pygame window that shows the generated path for a given set of obstacles. You should be ready to explain why the shown path is actually the shortest (for instance… "the diagonal move while costly, is necessary in order to reach the goal, paths consisting of just up, down, left, or right would not be able to reach the goal")."

Paul gets Diagonals

Read the get_open_adj_coords() function to get an idea of how valid adjacent cells are found. In the current code, valid adjacent cells only include the surrounding cells in the 4 cardinal directions, and moving to any of these cells costs 1 movement point. Add code changing the get_open_adj_coords() function so that surrounding diagonal cells are considered valid adjacent cells and moving to any of these cells costs 3 movement points. This will allow Paul to move diagonally.

Paul gets Hops

Evolve Paul and allow him to jump over lava! Add the ability for Paul to jump one square. This should cost 8 movement points, however. This will involve changing get_open_adj_coords().

Paul gets Swamped

Change the program so that pressing 's' allows you to add swamp tiles. Paul should be able to move through swamp tiles, but they will slow him down! Moving into a swamp tile will cost 3 additional movement points, so Paul should really avoid moving through swamp tiles unless he has to. Note that this means that moving to an adjacent swamp tile will cost 4 points, since moving adjacent one square costs 1 point, and the swamp adds 3. Moving into a diagonal swamp tile will cost 6 points. A swamp.jpg file has been provided for you in the /images folder. You will probably have to make some changes to pygame's main_loop to detect key presses, and you'll have to implement the empty _add_swamp() function. Take a look at _add_lava() for inspiration, and note the terrain_cost and is_unpassable arguments that are passed to the ObstacleTile constructor when a lava tile is made!

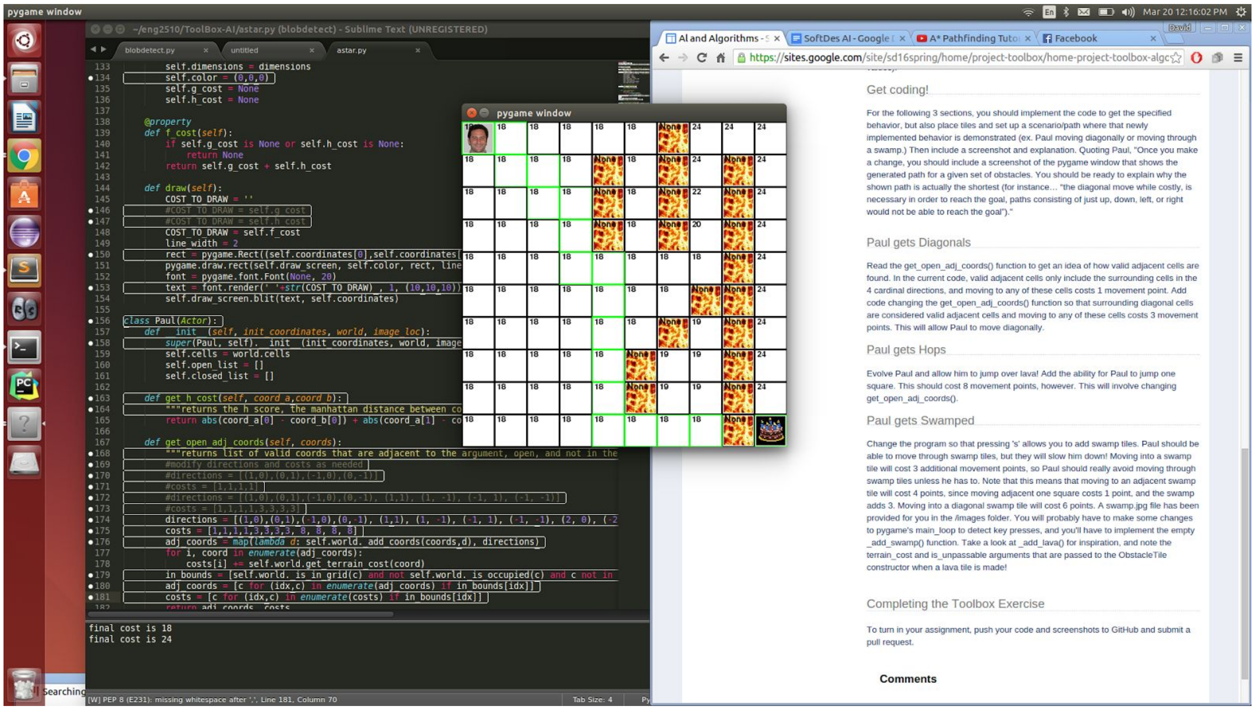
Completing the Toolbox Exercise

To turn in your assignment, push your code and screenshots to GitHub and submit a pull request.

Comments

Jump:

In this case, it is quicker to jump over the last lava than it is to take the long excursion in the top-right corner. You can tell Paul jumped successfully because the lava it jumped over is not highlighted in green.


pygame window
~/eng2510/ToolBox-AI/astar.py (blobdetect) - Sublime Text (UNREGISTERED)
blobdetect.py
untitled
astar.py
Searching
Mar 20 12:16:02 PM
En
AI and Algorithms - S
SoftDes AI - Google
A* Pathfinding Tuto
Facebook
David
https://sites.google.com/site/sd16spring/home/project-toolbox/home-project-toolbox-algo
pygame window

```
        self.dimensions = dimensions
        self.color = (0,0,0)
        self.g_cost = None
        self.h_cost = None

    @property
    def f_cost(self):
        if self.g_cost is None or self.h_cost is None:
            return None
        return self.g_cost + self.h_cost

    def draw(self):
        COST_TO_DRAW = ''
        #COST_TO_DRAW = self.g_cost
        #COST_TO_DRAW = self.h_cost
        COST_TO_DRAW = self.f_cost
        line_width = 2
        rect = pygame.Rect((self.coordinates[0],self.coordinates[
        pygame.draw.rect(self.draw_screen, self.color, rect, line
        font = pygame.font.Font(None, 20)
        text = font.render(' '+str(COST_TO_DRAW) , 1, (10,10,10))
        self.draw_screen.blit(text, self.coordinates)

class Paul(Actor):
    def __init__(self, init_coordinates, world, image_loc):
        super(Paul, self).__init__(init_coordinates, world, image
        self.cells = world.cells
        self.open_list = []
        self.closed_list = []

    def get_h_cost(self, coord_a,coord_b):
        """returns the h score, the manhattan distance between co
        return abs(coord_a[0] - coord_b[0]) + abs(coord_a[1] - co

    def get_open_adj_coords(self, coords):
        """returns list of valid coords that are adjacent to the argument, open, and not in the
        #modify directions and costs as needed
        #directions = [(1,0),(0,1),(-1,0),(0,-1)]
        #costs = [1,1,1,1]
        #directions = [(1,0),(0,1),(-1,0),(0,-1), (1,1), (1, -1), (-1, 1), (-1, -1)]
        #costs = [1,1,1,1,3,3,3,3]
        directions = [(1,0),(0,1),(-1,0),(0,-1), (1,1), (1, -1), (-1, 1), (-1, -1), (2, 0), (-2
        costs = [1,1,1,1,3,3,3,3, 8, 8, 8, 8]
        adj_coords = map(lambda d: self.world._add_coords(coords,d), directions)
        for i, coord in enumerate(adj_coords):
            costs[i] += self.world.get_terrain_cost(coord)
        in_bounds = [self.world._is_in_grid(c) and not self.world._is_occupied(c) and c not in
        adj_coords = [c for (idx,c) in enumerate(adj_coords) if in_bounds[idx]]
        costs = [c for (idx,c) in enumerate(costs) if in_bounds[idx]]
        return adj_coords, costs
```

```
final cost is 18
final cost is 24
```

```
[W] PEP 8 (E231): missing whitespace after ',', Line 181, Column 70          Tab Size: 4          Py
```

18	18	18	18	18	18	None	24	24	24
18	18	18	18	None	18	None	24	None	24
18	18	18	18	None	18	None	22	None	24
18	18	18	18	None	18	None	20	None	24
18	18	18	18	18	18	18	18	None	24
18	18	18	18	18	18	18	None	None	24
18	18	18	18	18	18	None	19	None	24
18	18	18	18	18	None	19	19	None	24
18	18	18	18	18	None	19	19	None	24
18	18	18	18	18	18	18	18	None	

Get coding!

For the following 3 sections, you should implement the code to get the specified behavior, but also place tiles and set up a scenario/path where that newly implemented behavior is demonstrated (ex. Paul moving diagonally or moving through a swamp.) Then include a screenshot and explanation. Quoting Paul, "Once you make a change, you should include a screenshot of the pygame window that shows the generated path for a given set of obstacles. You should be ready to explain why the shown path is actually the shortest (for instance... "the diagonal move while costly, is necessary in order to reach the goal, paths consisting of just up, down, left, or right would not be able to reach the goal")."

Paul gets Diagonals

Read the get_open_adj_coords() function to get an idea of how valid adjacent cells are found. In the current code, valid adjacent cells only include the surrounding cells in the 4 cardinal directions, and moving to any of these cells costs 1 movement point. Add code changing the get_open_adj_coords() function so that surrounding diagonal cells are considered valid adjacent cells and moving to any of these cells costs 3 movement points. This will allow Paul to move diagonally.

Paul gets Hops

Evolve Paul and allow him to jump over lava! Add the ability for Paul to jump one square. This should cost 8 movement points, however. This will involve changing get_open_adj_coords().

Paul gets Swamped

Change the program so that pressing 's' allows you to add swamp tiles. Paul should be able to move through swamp tiles, but they will slow him down! Moving into a swamp tile will cost 3 additional movement points, so Paul should really avoid moving through swamp tiles unless he has to. Note that this means that moving to an adjacent swamp tile will cost 4 points, since moving adjacent one square costs 1 point, and the swamp adds 3. Moving into a diagonal swamp tile will cost 6 points. A swamp.jpg file has been provided for you in the /images folder. You will probably have to make some changes to pygame's main_loop to detect key presses, and you'll have to implement the empty _add_swamp() function. Take a look at _add_lava() for inspiration, and note the terrain_cost and is_unpassable arguments that are passed to the ObstacleTile constructor when a lava tile is made!

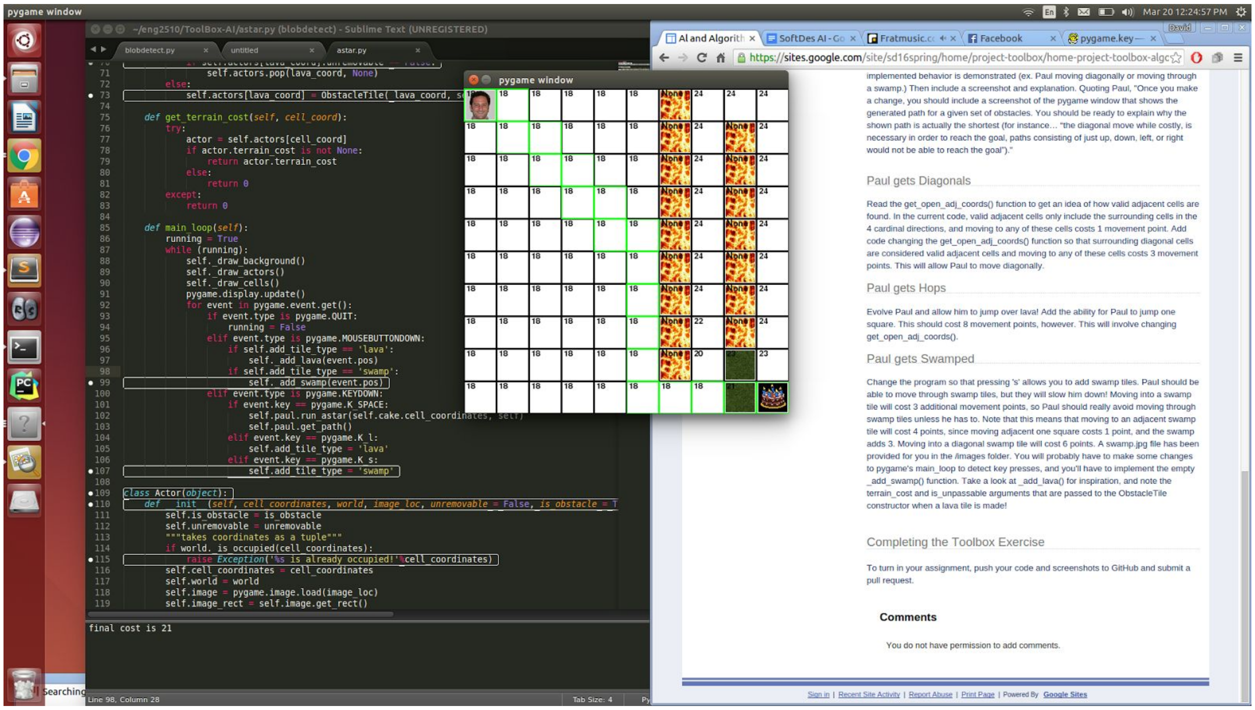
Completing the Toolbox Exercise

To turn in your assignment, push your code and screenshots to GitHub and submit a pull request.

Comments

Swamp:

In this case, it is quicker to move through the swamp than take the long excursion in the top-right corner. You can tell Paul crossed the swamp rather than jumping over it because the swamp is highlighted in green.



```
            if self.actors[lava_coord].unremovable == False:
                self.actors.pop(lava_coord, None)
        else:
            self.actors[lava_coord] = ObstacleTile( lava_coord, se

    def get_terrain_cost(self, cell_coord):
        try:
            actor = self.actors[cell_coord]
            if actor.terrain_cost is not None:
                return actor.terrain_cost
            else:
                return 0
        except:
            return 0

    def main_loop(self):
        running = True
        while (running):
            self._draw_background()
            self._draw_actors()
            self._draw_cells()
            pygame.display.update()
            for event in pygame.event.get():
                if event.type is pygame.QUIT:
                    running = False
                elif event.type is pygame.MOUSEBUTTONDOWN:
                    if self.add_tile_type == 'lava':
                        self._add_lava(event.pos)
                    if self.add_tile_type == 'swamp':
                        self._add_swamp(event.pos)
                elif event.type is pygame.KEYDOWN:
                    if event.key == pygame.K_SPACE:
                        self.paul.run_astar(self.cake.cell_coordinates, self)
                        self.paul.get_path()
                    elif event.key == pygame.K_l:
                        self.add_tile_type = 'lava'
                    elif event.key == pygame.K_s:
                        self.add_tile_type = 'swamp'

class Actor(object):
    def __init__(self, cell_coordinates, world, image_loc, unremovable = False, is_obstacle = T
        self.is_obstacle = is_obstacle
        self.unremovable = unremovable
        """takes coordinates as a tuple"""
        if world._is_occupied(cell_coordinates):
            raise Exception('%s is already occupied!'%cell_coordinates)
        self.cell_coordinates = cell_coordinates
        self.world = world
        self.image = pygame.image.load(image_loc)
        self.image_rect = self.image.get_rect()
```

```
final cost is 21
```

pygame window

18	18	18	18	18	18	None	24	24	24
18	18	18	18	18	18	None	24	None	24
18	18	18	18	18	18	None	24	None	24
18	18	18	18	18	18	None	24	None	24
18	18	18	18	18	18	None	24	None	24
18	18	18	18	18	18	None	24	None	24
18	18	18	18	18	18	None	24	None	24
18	18	18	18	18	18	None	22	None	24
18	18	18	18	18	18	None	20	23	23
18	18	18	18	18	18	18	18	21	

implemented behavior is demonstrated (ex. Paul moving diagonally or moving through a swamp.) Then include a screenshot and explanation. Quoting Paul, "Once you make a change, you should include a screenshot of the pygame window that shows the generated path for a given set of obstacles. You should be ready to explain why the shown path is actually the shortest (for instance... "the diagonal move while costly, is necessary in order to reach the goal, paths consisting of just up, down, left, or right would not be able to reach the goal")."

Paul gets Diagonals

Read the get_open_adj_coords() function to get an idea of how valid adjacent cells are found. In the current code, valid adjacent cells only include the surrounding cells in the 4 cardinal directions, and moving to any of these cells costs 1 movement point. Add code changing the get_open_adj_coords() function so that surrounding diagonal cells are considered valid adjacent cells and moving to any of these cells costs 3 movement points. This will allow Paul to move diagonally.

Paul gets Hops

Evolve Paul and allow him to jump over lava! Add the ability for Paul to jump one square. This should cost 8 movement points, however. This will involve changing get_open_adj_coords().

Paul gets Swamped

Change the program so that pressing 's' allows you to add swamp tiles. Paul should be able to move through swamp tiles, but they will slow him down! Moving into a swamp tile will cost 3 additional movement points, so Paul should really avoid moving through swamp tiles unless he has to. Note that this means that moving to an adjacent swamp tile will cost 4 points, since moving adjacent one square costs 1 point, and the swamp adds 3. Moving into a diagonal swamp tile will cost 6 points. A swamp.jpg file has been provided for you in the /images folder. You will probably have to make some changes to pygame's main_loop to detect key presses, and you'll have to implement the empty _add_swamp() function. Take a look at _add_lava() for inspiration, and note the terrain_cost and is_unpassable arguments that are passed to the ObstacleTile constructor when a lava tile is made!

Completing the Toolbox Exercise

To turn in your assignment, push your code and screenshots to GitHub and submit a pull request.

Comments

You do not have permission to add comments.

Sign in | Recent Site Activity | Report Abuse | Print Page | Powered By Google Sites