File Edit View Selection Find Packages Help

ToolBox-AI
- .git
- images
- .gitignore
- ai-readme.md
- astar.py
- README.md

astar.py

```
136            self.h_cost = None
137
138        @property
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],
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,
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 adjacent to the argument, open, and not in the closed list."""
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 self.closed_list for c in adj_coords]
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
```

File 0   Project 0   ✓ No Issues   astar.py   147:10   LF   UTF-8   Python   master   +5, -5   1 update

Terminal - jwb@6RNPT32: ~/Documents/git/ToolBo
File Edit View Terminal Tabs Help

```
jwb@6RNPT32:~/Documents/git/Readi...   jwb@6RNPT32: ~/D
GeneFinder      ReadingJournal        toolbox
github-research   slaides            ToolBox-AI
jwb@6RNPT32:~/Documents/git$ cd ToolBox-AI
jwb@6RNPT32:~/Documents/git/ToolBox-AI$ ls
ai-readme.md  astar.py  images  README.md
jwb@6RNPT32:~/Documents/git/ToolBox-AI$ python astar.py
final cost is 24
jwb@6RNPT32:~/Documents/git/ToolBox-AI$ atom a
jwb@6RNPT32:~/Documents/git/ToolBox-AI$ atom astar.py
jwb@6RNPT32:~/Documents/git/ToolBox-AI$ python astar.py
final cost is 18
final cost is 18
final cost is 18
final cost is 18
final cost is 18
final cost is 22
final cost is 26
```

pygame window

File Edit View Selection Find Packages Help

ToolBox-AI
- .git
- images
- .gitignore
- ai-readme.md
- astar.py
- README.md

astar.py

```
136            self.h_cost = None
137
138        @property
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],
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,
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 adjacent to the argument, open, and not in the closed
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 self.closed_list for c in adj_coords]
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
```

File 0   Project 0   ✓ No Issues   astar.py   146:9   LF   UTF-8   Python   master   +5, -5   1 update

Terminal - jwb@6RNPT32: ~/Documents/git/ToolBox-AI
File Edit View Terminal Tabs Help

```
jwb@6RNPT32:~/Documents/git/Readi...   jwb@6RNPT32: ~/Documer
jwb@6RNPT32:~/Documents/git/ToolBox-AI$ atom astar.py
jwb@6RNPT32:~/Documents/git/ToolBox-AI$ python astar.py
ai-readme.md  astar.py
jwb@6RNPT32:~/Documents/git/ToolBox-AI$ python astar.py
final cost is 18
final cost is 18
final cost is 18
final cost is 18
final cost is 18
final cost is 22
final cost is 26
jwb@6RNPT32:~/Documents/git/ToolBox-AI$ python astar.py
final cost is 18
final cost is 26
final cost is 26
final cost is 26
final cost is 26
final cost is 30
final cost is 18
final cost is 24
final cost is 24
```
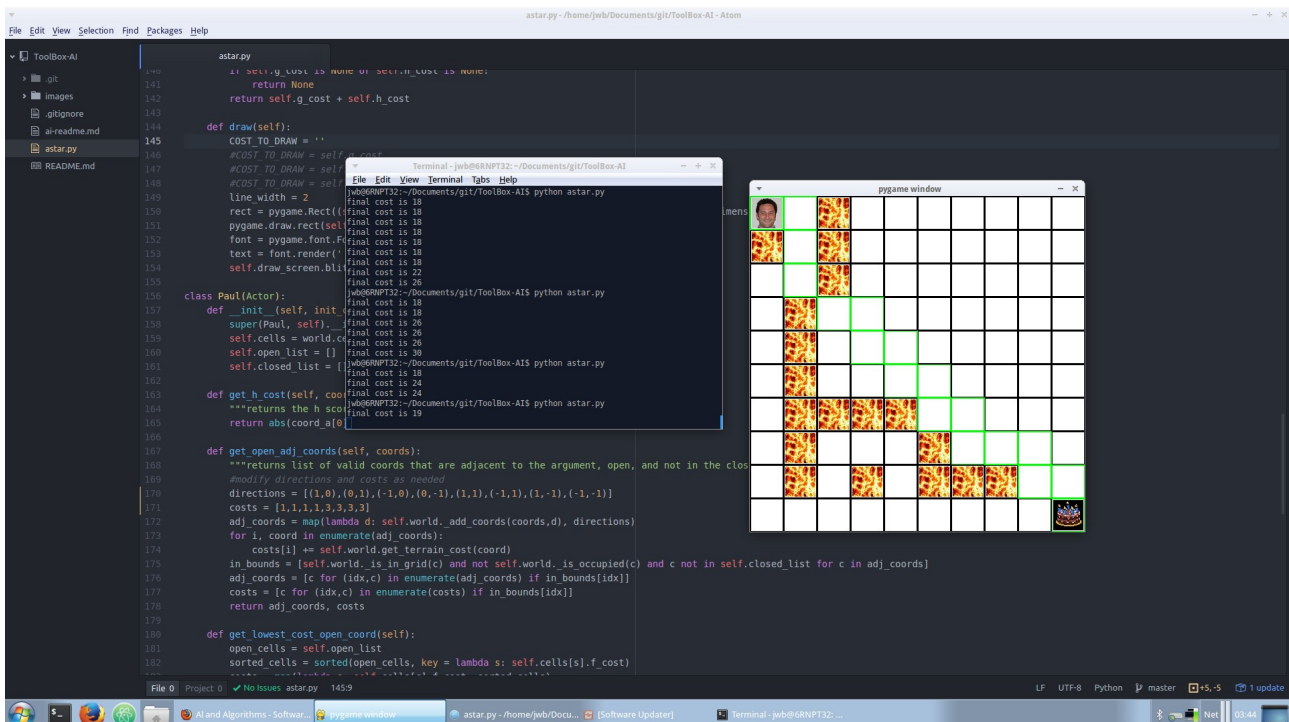
pygame window

Here we see three different costs visualized: f, g and h. g is the cost that it takes to reach that node on our abstract graph, or grid location in this case. H is a prediction of how costly it will be from that node to the goal, or that grid square. F is just their sum.

Here we see diagonals implemented. Instead of taking a more costly edge route, paul can cut through a diagonal.
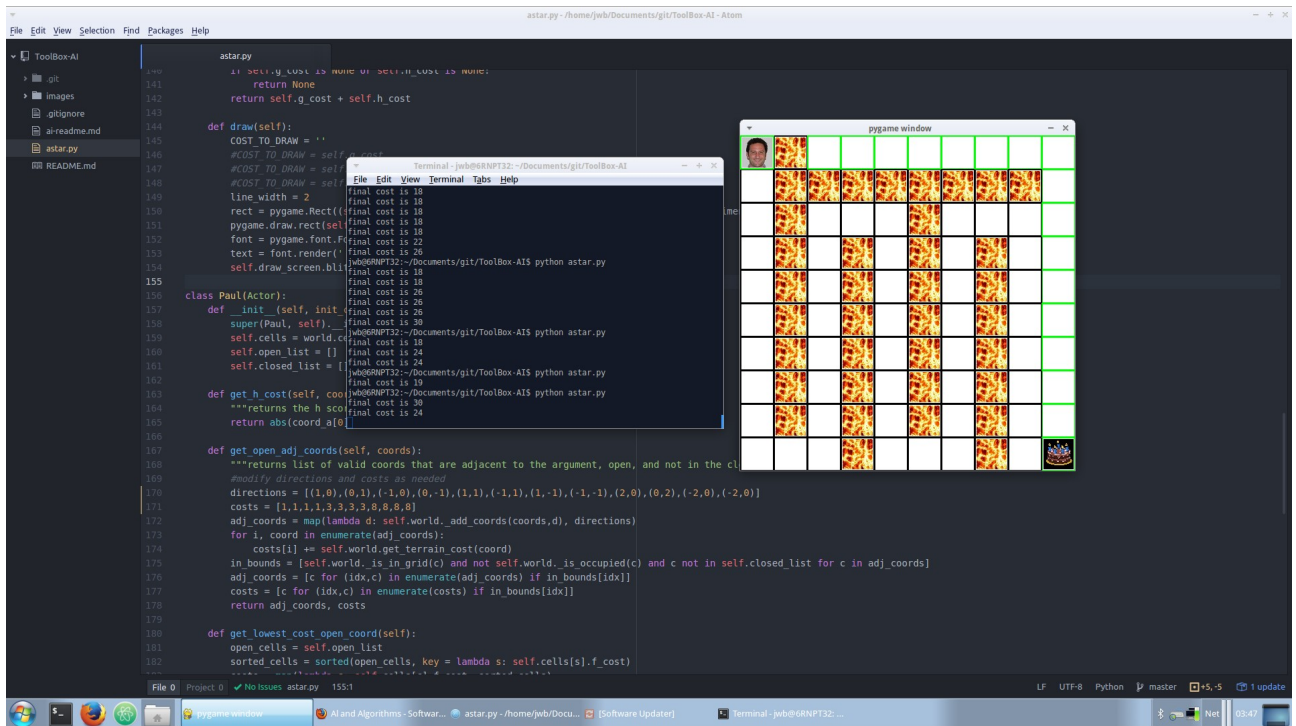
Here we see hops. Again, instead of a much more costly path in total, paul can just hop one square, even though this incurs a higher step cost.
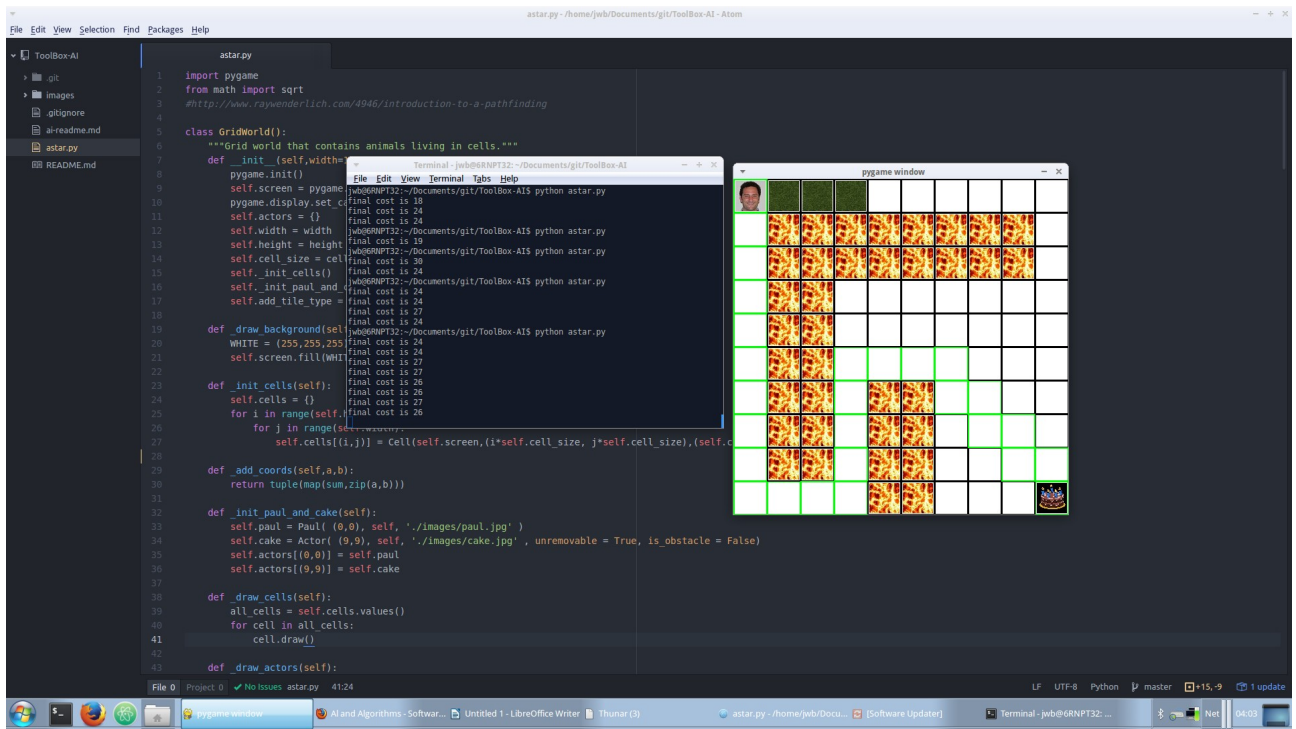
Here we see swamp tiles. In the first picture, you can see paul is very averse to moving through swamp tiles, taking a longer but cheaper route. Hover, with enough costs incurred on all other options, the moving through swamps becomes the most efficient.