artisoc Cloudレシピブック

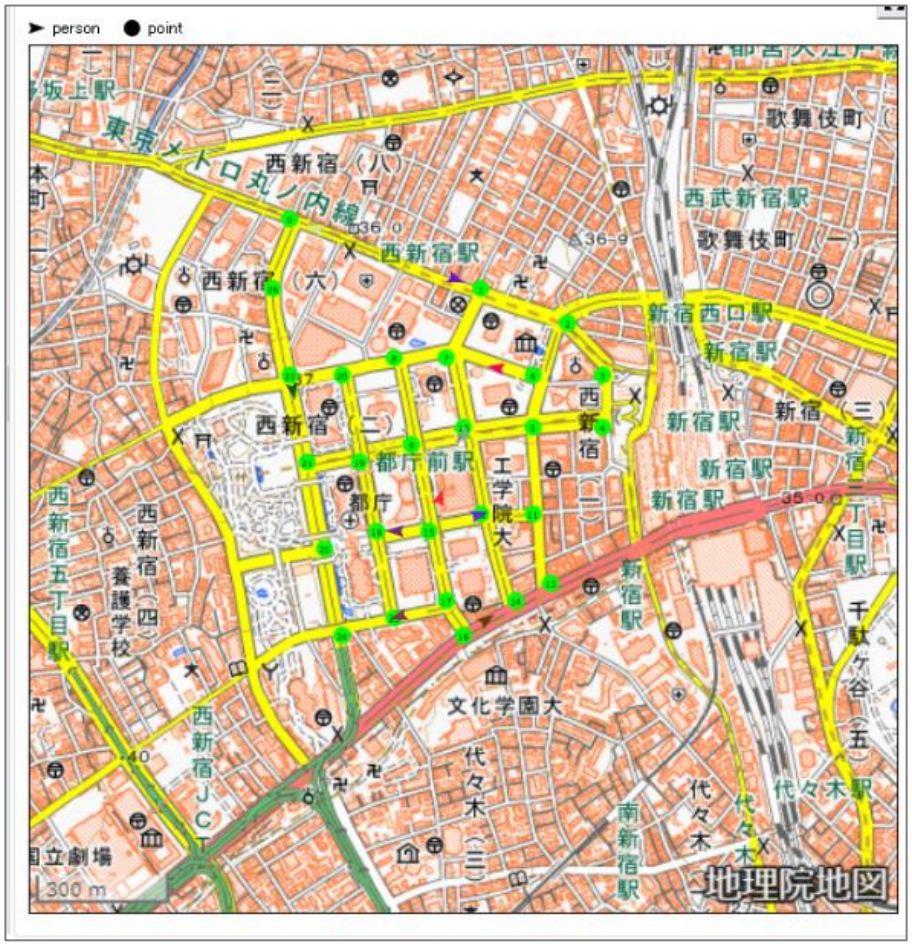
04. 道路に沿って歩くモデルをつくろう

（株）構造計画研究所
創造工学部
https://mas.kke.co.jp

道路に沿って歩くモデルをつくろう

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

マップ上を与えられた経路で歩くモデルを作成します。
下記のモデルは Firefox で実行してください。



04. レシピブック　サンプルモデル1

① モデルを定義する（1）

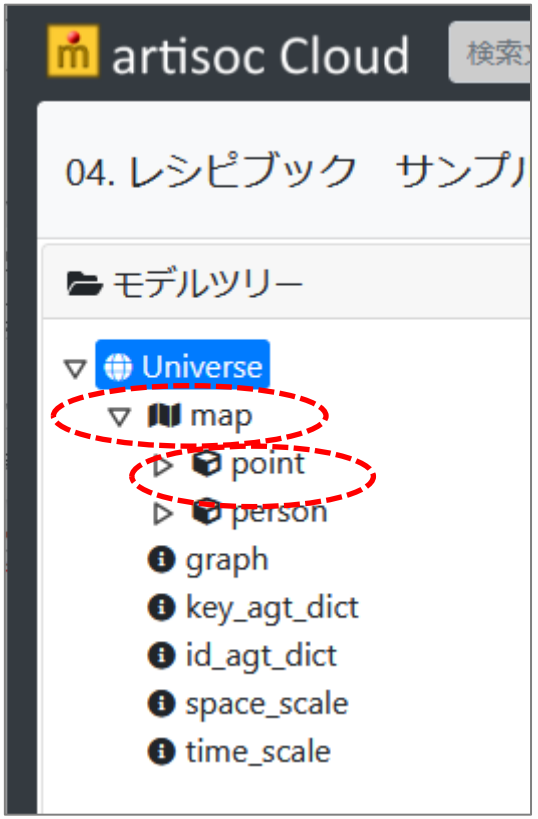
構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

モデルツリーで「空間」「エージェント」「変数」を定義します

- モデルツリーの「Universe ＋」をクリックして、
「空間を追加」を選択します。
  - 空間名：　map
  - 空間の大きさ X：　500
  - 空間の大きさ Y：　500
  - ループする：　チェックしない

- モデルツリーの「map ＋」をクリックして、
「エージェント種別を追加」を選択します。
  - エージェント種別名： point

- モデルツリーの「point ＋」をクリックします。
  - 変数名：　point_id
  - 変数名：　color
  - 変数名：　link_list



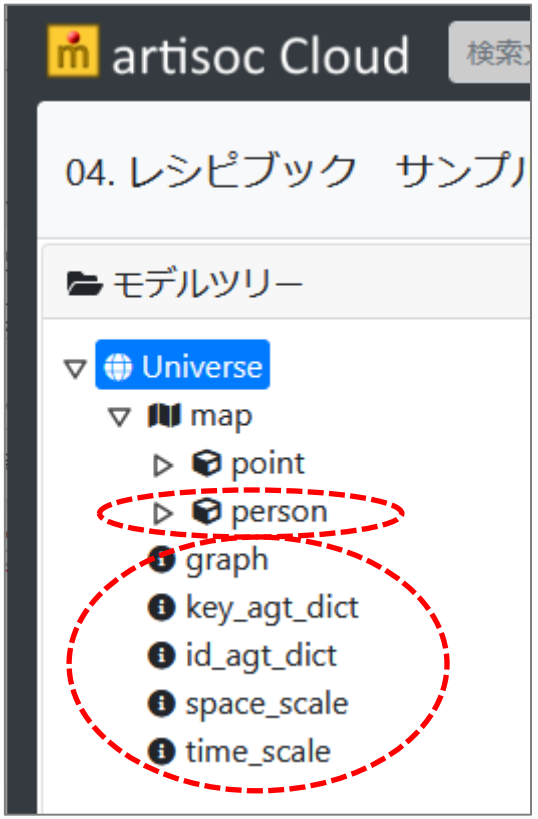
① モデルを定義する（2）

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

- モデルツリーの「map ＋」をクリックして、
「エージェント種別を追加」を選択します。
エージェント種別名： person

- モデルツリーの「person ＋」をクリックします。
変数名： color
変数名： speed
変数名： route
変数名： route_count

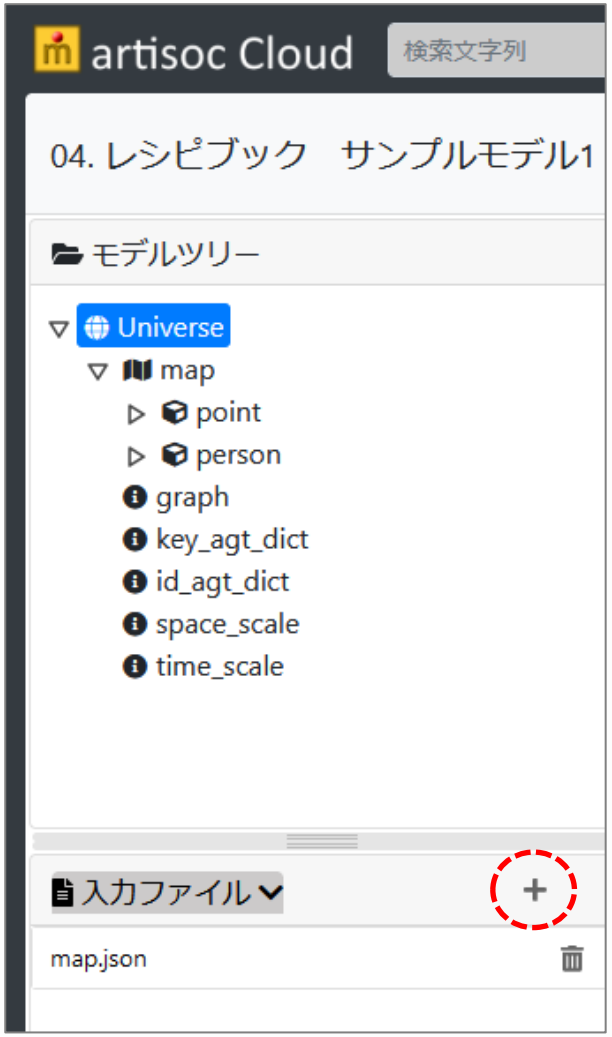
- モデルツリーの「Universe ＋」をクリックして、
「変数を追加」を選択します。
変数名： graph
変数名： key_agt_dict
変数名： id_agt_dict
変数名： space_scale
変数名： time_scale



① モデルを定義する（3）

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

- 入力ファイルの「＋」をクリックして、
  入力ファイルをインポートします。
  ファイル名：　map.json



② エージェントのルールを記述する（1）

ネットワークファイル（map.json）で定義した道路に沿って歩きます

- モデルツリーの「Universe」をクリックしてルールエディタを表示します。

```
def univ_init(self):

    Universe.space_scale = 3.88             # 地図の縮尺（m/セル）←03. レシピブックで求めた値
    Universe.time_scale = 1                 # 時間刻み（sec/step）
    self.file_read_network('map.json')      # ネットワークの初期化
    create_agt(Universe.map.person, num=10) # personを生成

```

```
# ネットワークの初期化
def file_read_network(self, one_filename):

    import json
    import networkx

    # 変数の初期化
    Universe.key_agt_dict = {}
    Universe.id_agt_dict = {}
    Universe.graph = networkx.DiGraph()

    # jsonファイル読込
    f = open(one_filename, 'r')
    map_json_dict = json.load(f)
    f.close()

    self.init_point(map_json_dict)      # ポイントの初期化
    self.init_link(map_json_dict)       # リンクの初期化
```

② エージェントのルールを記述する（2）

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

- モデルツリーの「Universe」をクリックしてルールエディタを表示します。

```
# ポイントの初期化
def init_point(self, map_json_dict):

    import networkx

    h = get_height_space(Universe.map)  # 高さの取得

    for one_feature in map_json_dict['pointArray']:
        one_coordinates = one_feature['coordinates']
        one_id = one_feature['id']
        one_color = one_feature['color'].split(',')

        # pointの生成
        one_point = create_agt(Universe.map.point)
        one_point.x = float(one_coordinates[0])
        one_point.y = h - float(one_coordinates[1])     # Y座標が逆向き
        one_point.point_id = one_id
        one_point.color = rgb(int(one_color[0]), int(one_color[1]), int(one_color[2]))
        one_point.link_list = []

        # graphに追加
        one_key = str(one_coordinates[0]) + str('_') + str(one_coordinates[1])
        Universe.key_agt_dict[one_key] = one_point
        Universe.id_agt_dict[str(one_id)] = one_point
        Universe.graph.add_node(str(one_id))
```

② エージェントのルールを記述する（3）

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

- モデルツリーの「Universe」をクリックしてルールエディタを表示します。

```
62  # リンクの初期化
63  def init_link(self, map_json_dict):
64
65      import networkx
66
67      for one_feature in map_json_dict['linkArray']:
68          one_coordinates_array = one_feature['coordinates']
69
70          # 1点目の情報取得
71          one_coordinates = one_coordinates_array[0]
72          one_key = str(one_coordinates[0]) + str('_') + str(one_coordinates[1])
73          one_point = Universe.key_agt_dict[one_key]
74          one_id = one_point.point_id
75
76          # 2点目の情報取得
77          two_coordinates = one_coordinates_array[1]
78          two_key = str(two_coordinates[0]) + str('_') + str(two_coordinates[1])
79          two_point = Universe.key_agt_dict[two_key]
80          two_id = two_point.point_id
81
82          # graphに追加
83          distance = measure_agt_distance(one_point, two_point)
84          Universe.graph.add_edge(str(one_id), str(two_id), weight=distance)
85          Universe.graph.add_edge(str(two_id), str(one_id), weight=distance)
86
87          # link_listに追加
88          one_point.link_list.append(two_point)
89          two_point.link_list.append(one_point)
```

② エージェントのルールを記述する（4）

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

- モデルツリーの「person」をクリックしてルールエディタを表示します。

```
def agt_init(self):

    self.color = rgb(randint(0, 255), randint(0, 255), randint(0, 255))     # 色をランダムに指定
    self.speed = 1                                                          # 移動速度（m/s）の指定
    self.speed = self.speed / Universe.space_scale * Universe.time_scale    # 速度変換
    self.route = []

    # 出発地をランダムに決める
    start_id = self.get_random_id()
    start_agt = Universe.id_agt_dict[str(start_id)]
    self.x = start_agt.x
    self.y = start_agt.y

    # 目的地をランダムに決めて経路を設定
    self.set_random_route(start_id)

def agt_step(self):

    # 経路に沿って移動
    target_id = int(self.route[self.route_count])
    target_agt = Universe.id_agt_dict[str(target_id)]
    distance = self.pursue(target_agt, self.speed)

    while(distance > 0):
        self.route_count += 1
        if len(self.route) <= self.route_count:
            # 新しい経路をランダムに設定
            self.set_random_route(target_id)
        else:
            # 経路に沿って移動
            target_id = int(self.route[self.route_count])
            target_agt = Universe.id_agt_dict[str(target_id)]
            distance = self.pursue(target_agt, distance)
```

② エージェントのルールを記述する（5）

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

- モデルツリーの「person」をクリックしてルールエディタを表示します。

```
34  # 目的地をランダムに決めて経路を設定
35  def set_random_route(self, start_id):
36
37      while(True):
38          goal_id = self.get_random_id()
39          if start_id != goal_id: # 出発地と目的地が同一でない場合
40              break
41
42      self.route = self.get_route(start_id, goal_id)   # 最短経路を取得
43      self.route_count = 0
44
45  # 2点間の最短経路を取得
46  def get_route(self, start_id, goal_id):
47
48      import networkx as nx
49
50      # ダイクストラ法を使って最短経路を探索
51      one_route = nx.dijkstra_path(Universe.graph, str(start_id), str(goal_id), weight='distance')
52
53      return(one_route)
54
55  # point_idをランダムに取得
56  def get_random_id(self):
57      return(random.choice(list(Universe.id_agt_dict.keys())))
```

③ 出力画面を定義する（1）

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

マップ出力画面を定義します

- 出力画面を表示します。

- 出力パネル > 出力設定 > マップ出力 を選択し、「追加」をクリックします。
  - マップ名： map
  - 空間： map
  - 固定画像： map.png ←03. レシピブックで取得した画像ファイル

- マップ要素リスト > エージェント「＋」をクリックします。
  - 要素名： person
  - 出力対象： person
  - マーカー
    - 選択： 矢印
  - エージェント表示色：
    - 変数指定： color
  - 拡大率：
    - 固定値： 10

③ 出力画面を定義する（2）

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

- マップ要素リスト > エージェント「+」をクリックします。

  要素名：　point
  出力対象：point
  エージェント表示色：
  　変数指定：color
  拡大率：
  　固定値：10
  エージェント情報の表示：
  　表示する変数：　point_id
  エージェント間に線を引く：
  　対象変数：　link_list
  　線の色：　0,255,0

③ 出力画面を定義する（3）

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

- 出力パネル > 出力設定 > 値出力 を選択し、「追加」をクリックします。
  - 出力名：　値出力

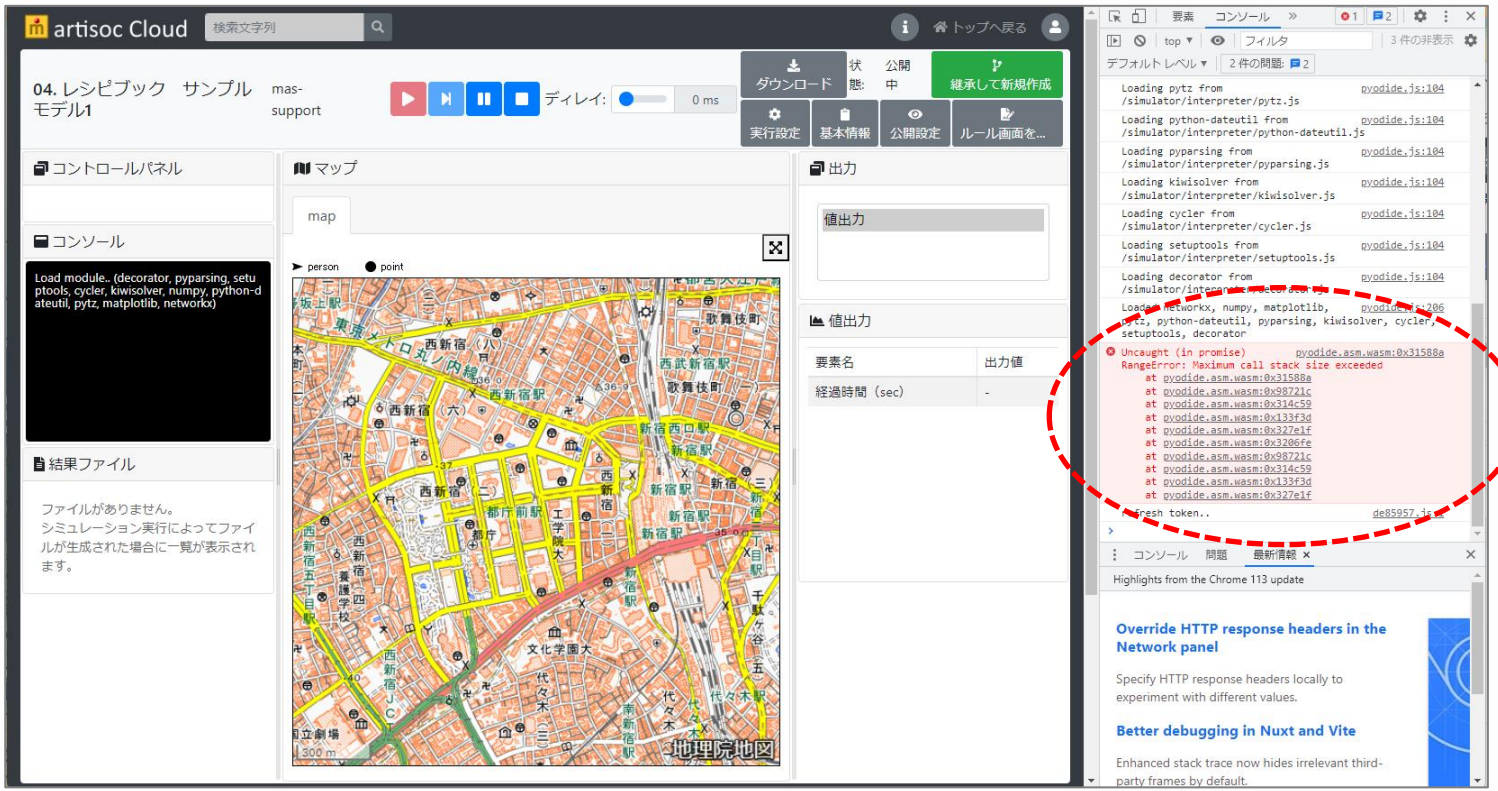
- 値出力要素リスト + をクリックします。
  - 要素名：　経過時間（sec）
  - 出力対象：count_step()
  - 小数表示：0 桁

- シミュレーションを実行します。

[Tips] サンプルモデルが実行できないとき

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

- Chromeでサンプルモデルが実行できない場合は、メモリエラーの可能性があります。
- [その他のツール]>[デベロッパーツール] を開き、下記のエラーが出ているときは Firefox で実行してください。

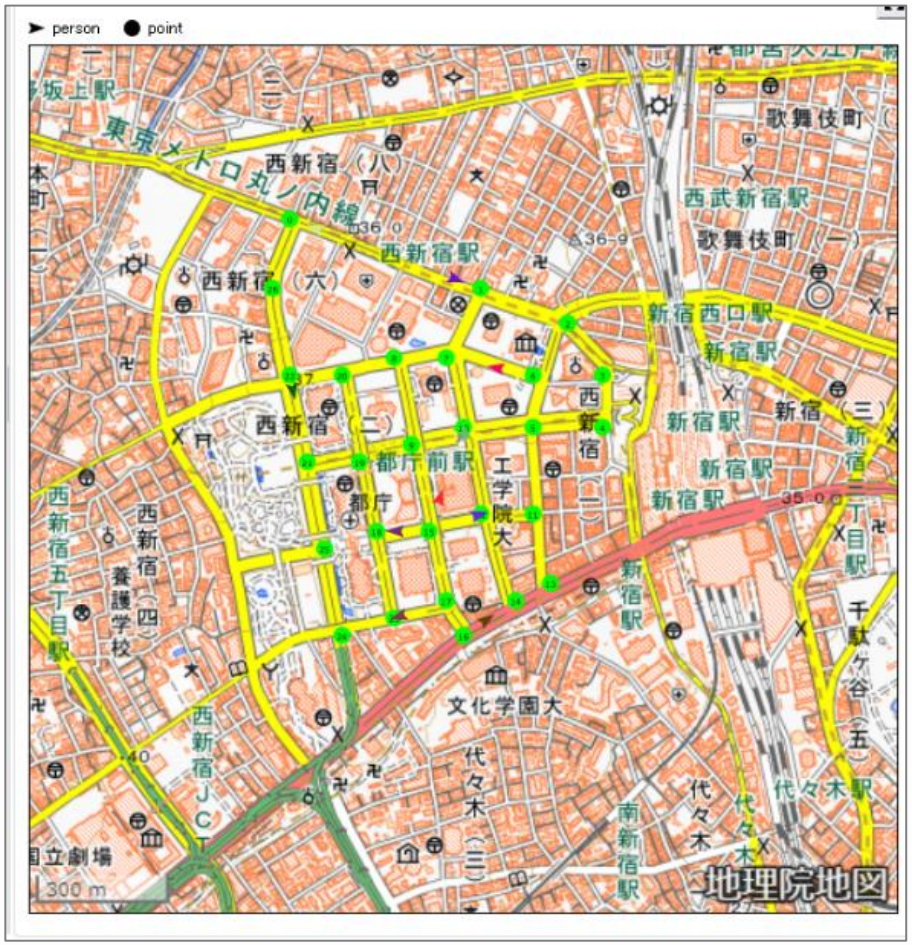


[Tips] 目的地を決めて移動したいとき

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

目的地を決めて移動したいときは、次の変更を加えることで実現できます。

- 目的地が 1 箇所の場合は、Universe.goal_id を追加して、univ_initで初期化します。
- 出発地と目的地が同じにならないようにperson のget_random_id を修正します。
- person の agt_init で出発地から目的地までの経路を設定します。
- person の agt_step で目的地に到着したら削除するようにします。



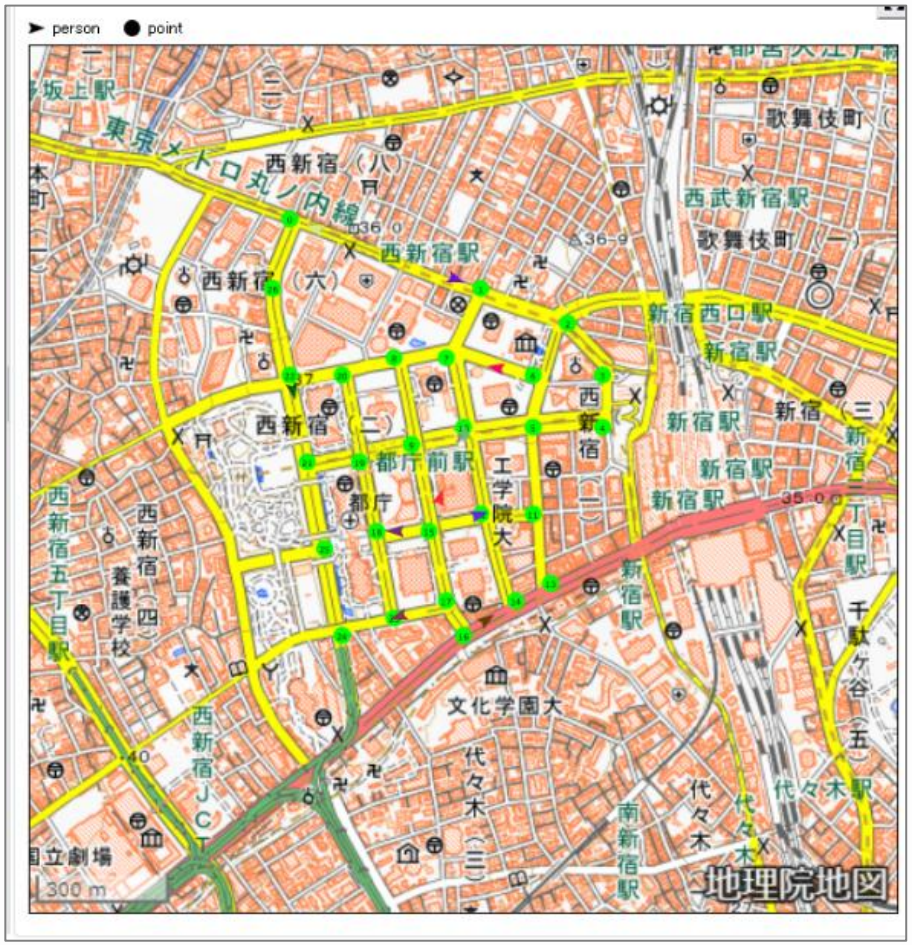
04. レシピブック　サンプルモデル2

[Tips] 最も近い目的地へ移動したいとき

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

複数の目的地の中から最も近い目的地へ移動したいときは、次の変更を加えることで実現できます。

- 目的地が複数の場合は、Universe.goal_id_list を追加して、univ_initで初期化します。
- 2点間の最短経路の距離を取得するためにperson に get_route_length を追加します。
- 目的地リストの中から最も近い目的地までの最短経路を取得するために person に get_route_from_goal_id_list を追加します。
- person の agt_init で出発地から目的地までの経路を設定します。



04. レシピブック　サンプルモデル3