artisoc Cloudレシピブック

06. 描画ツール2を使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

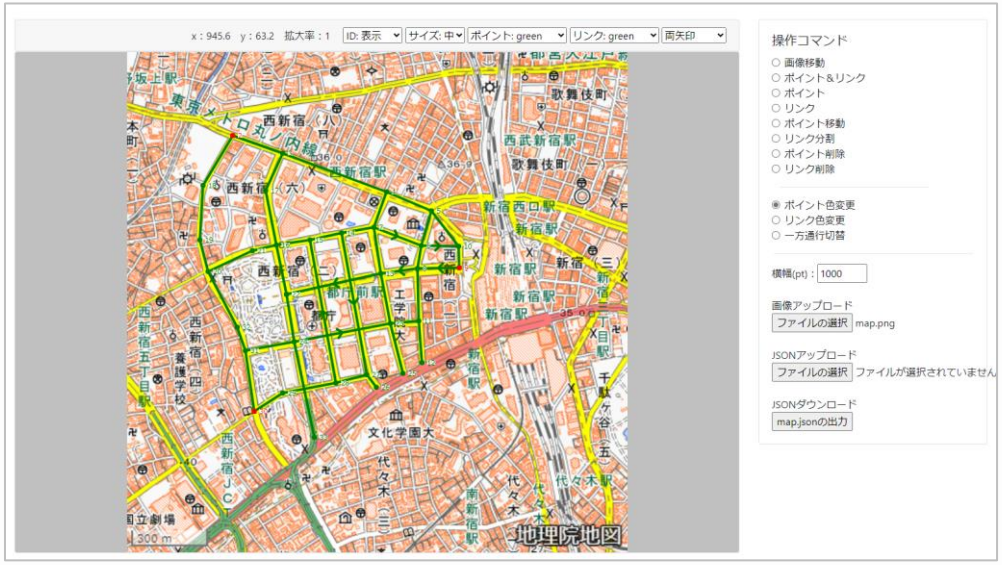
描画ツール2の提供をはじめました

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

描画ツール2は、マウスで簡単に道路が定義できるお絵描きアプリです

主な改良点

- 描画ツールの操作性を継承しながら、画面デザインを刷新して操作性を向上しました
- 「画像の拡大/縮小と移動」、「座標の表示」、「文字サイズの変更」を追加しました
- 操作コマンドに「リンクを分割」、「一方通行切替」を追加しました



描画ツール2

※ 旧バージョンはここからアクセスできます → 描画ツール

地理院地図から背景画像を取得する（1）

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

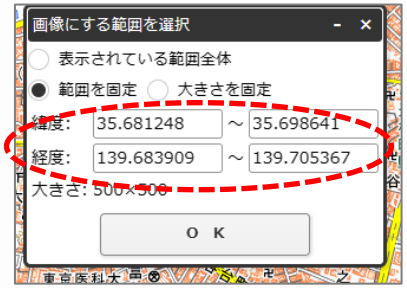
地理院地図から背景地図を取得します　※レシピブック03と同じ手順です

- 地理院地図にアクセスし、任意の場所（例えば、新宿都庁）を検索して表示します。

- 右上の 共有 > 画像保存 をクリックします。画像大きさを指定し、ファイルを出力します。
  大きさを固定
  大きさ：　500 x 500
  出力ファイル名：　map.png

地理院地図から背景画像を取得する（2）

地図の縮尺を取得します　※レシピブック03と同じ手順です



- 画像を出力するときに緯度・経度が表示されます。

- 距離と方位角の計算 を開き、入力単位選択を
「十進法度単位」に切り替えた後、緯度・経度を入力します。

  出発点　緯度：　35.681248
  　　経度：　139.683909
  到着点　緯度：　35.681248　←出発点と同じ値
  　　経度：　139.705367

- 「計算実行」をクリックすると距離が出力されます。
  測地線長　1,942.488(m)

- 地図の縮尺を求めます。
  地図の縮尺：　1942.488(m) / 500(セル) = 3.88（m/セル）←後述でルールに記述

描画ツール2を使って道路を定義する（1）

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

描画ツール2を使って道路を定義します

- 描画ツール2にアクセスします。
- 画像ファイル（map.png）の読み込みます。
- 操作コマンドを選択して、道路上をマウスでクリックしていきます。　（次ページ参照）
- すべての線を引き終わったら、JSONファイルをダウンロードします。
  ファイル名：　map.json

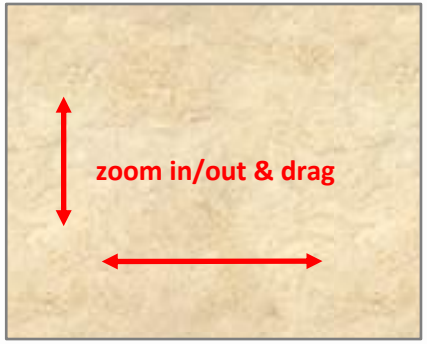
描画ツール2

描画ツール2を使って道路を定義する（2）

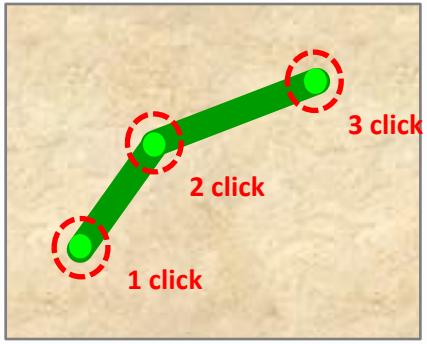
構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

操作コマンドを選択して、マウスをクリックします

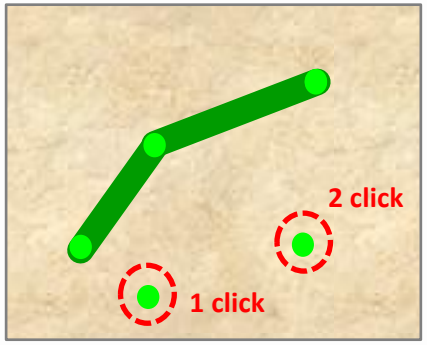
① 画像移動



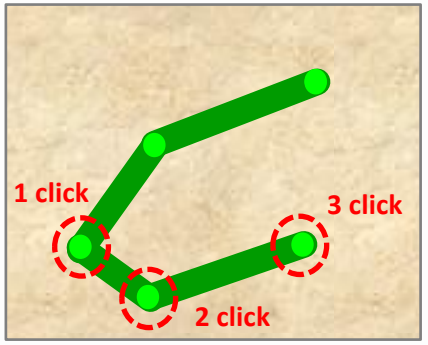
② ポイント＆リンク



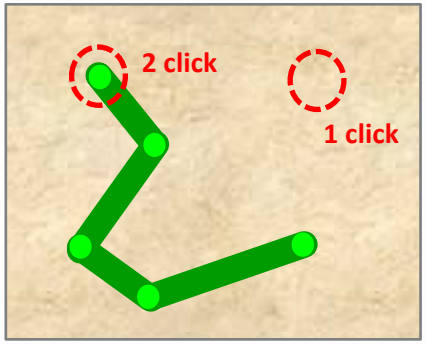
③ ポイント



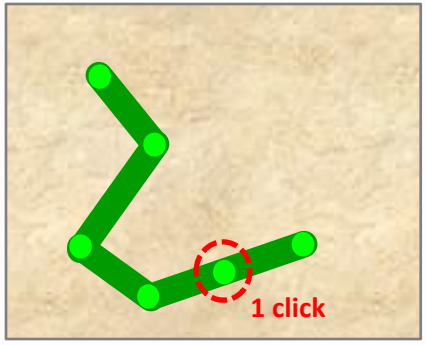
④ リンク



⑤ ポイント移動

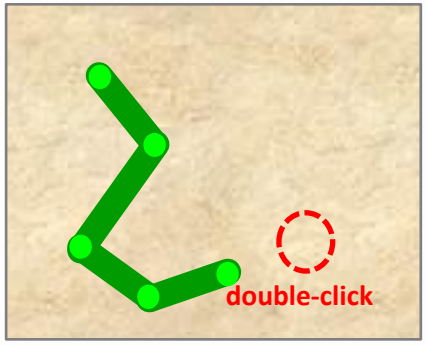


⑥ リンク分割



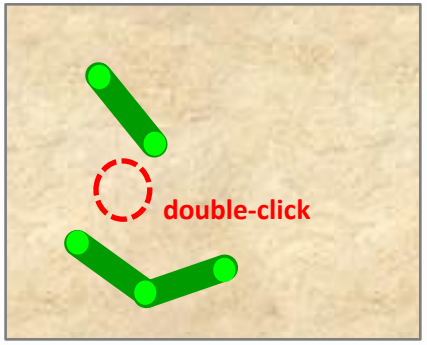
※ 分割するリンクの真ん中を
クリック

⑦ ポイント削除



※ ポイントにつながる
リンクも削除される

⑧ リンク削除



※ 削除するリンクの真ん中を
クリック

描画ツール2を使って道路を定義する（3）

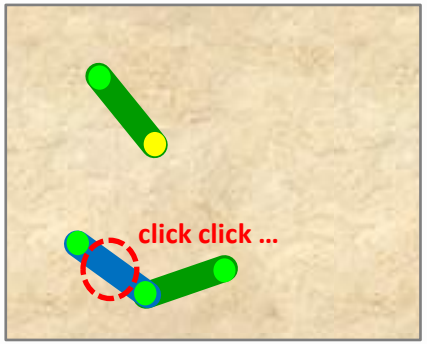
構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

操作コマンドを選択して、マウスをクリックします

⑨ ポイント色変更

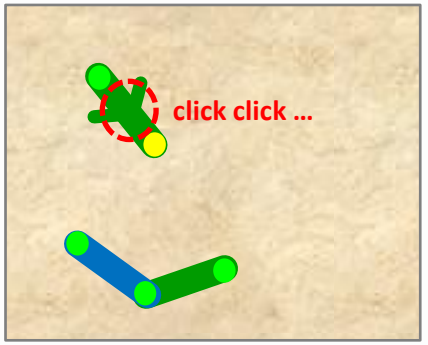
※ 緑→赤→青→黄→水色
→紫が順番に切り替わる

⑩ リンク色変更



※ 緑→赤→青→黄→水色
→紫が順番に切り替わる

⑪ 一方通行切替



※ 両方向→順方向→逆方向が
順番に切り替わる

横幅(pt)を変更すると、画面上部の座標の縮尺が変更されます

描画ツールで作成したjsonファイル

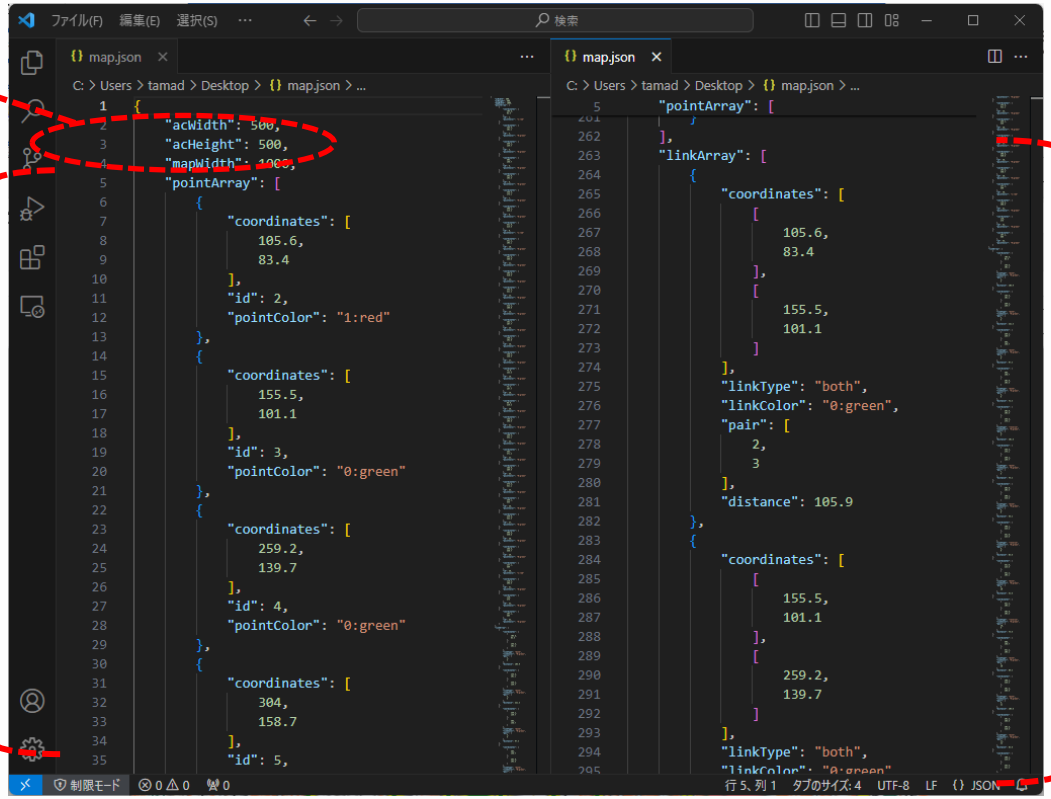
作成したjsonファイルは、テキストエディタで編集することができます

- acWidth, acHeight : マップの大きさ（横幅, 縦幅） ※画像ファイルのサイズ
- mapWidth : 横軸(pt)で指定した値
- pointArray : ポイントの配列（座標、id、表示色）
- LinkArray : リンクの配列（2点の座標、色、リンクの太さ）

マップの大きさ
横軸(pt)の値

pointArray

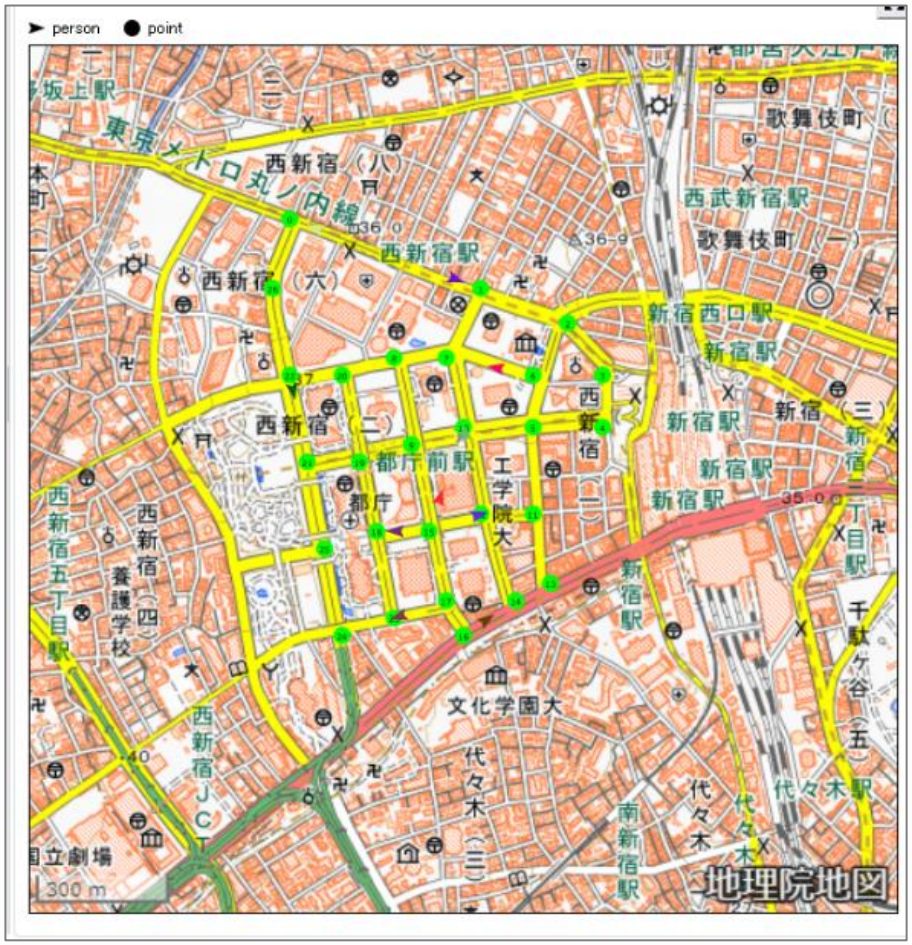
lnkArray



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 で実行することをおすすめします。



06. レシピブック　サンプルモデル1

① モデルを定義する（1）

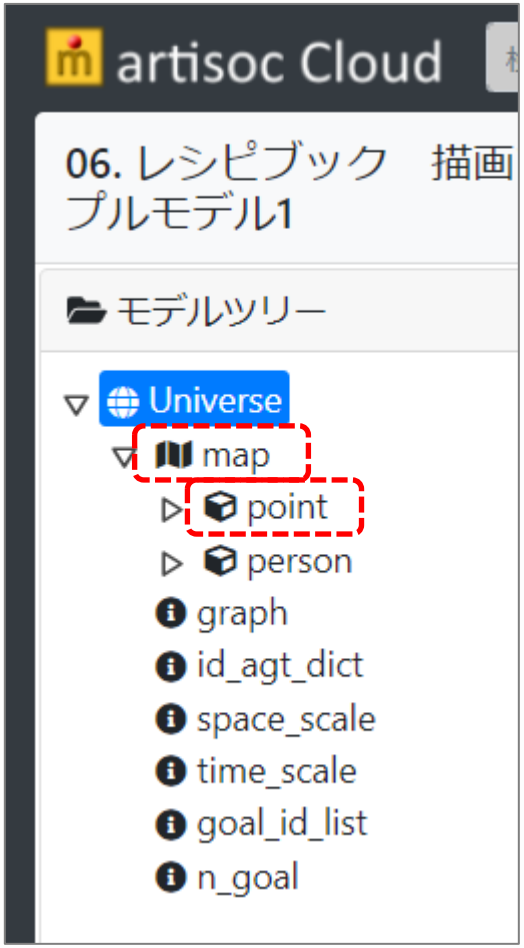
構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

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
  - 変数名：　link_color_list
  - 変数名：　link_type_list
  - 変数名：　view_size



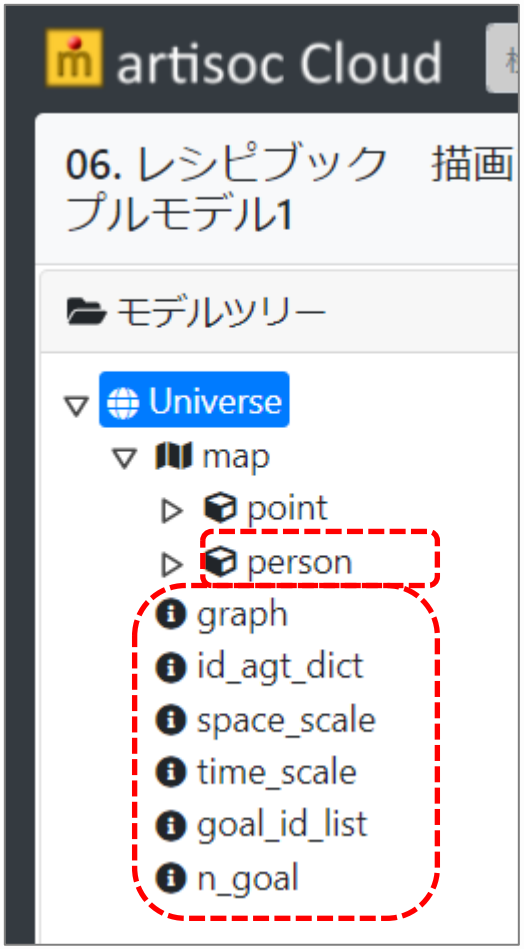
① モデルを定義する（2）

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

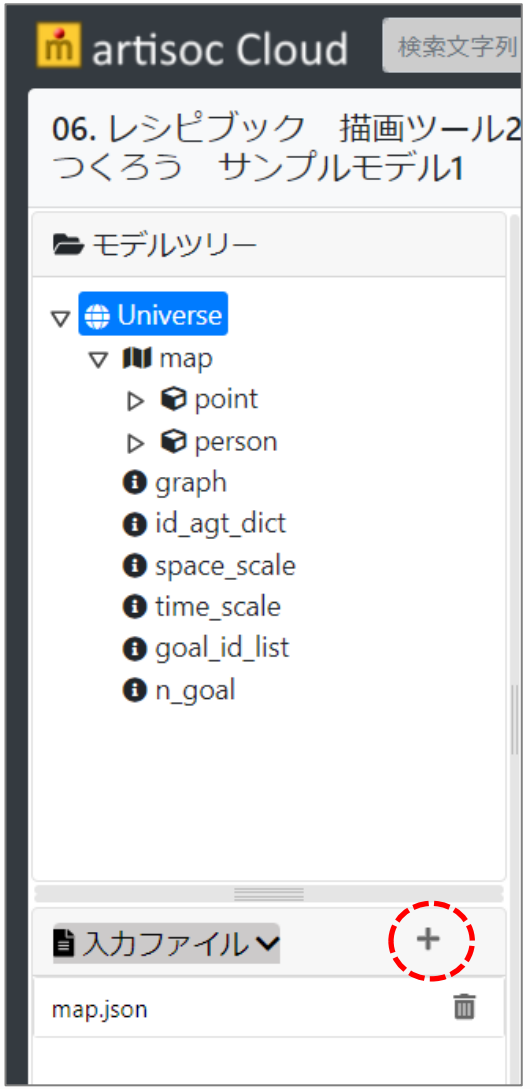
- モデルツリーの「Universe ＋」をクリックして、
「変数を追加」を選択します。
    変数名：　graph
    変数名：　id_agt_dict
    変数名：　space_scale
    変数名：　time_scale
    変数名：　goal_id_list
    変数名：　n_goal



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 ←描画ツール2で作成した
  jsonファイル



② エージェントのルールを記述する（1）

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

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）←地図の縮尺で求めた値
    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

    self.set_goal_id_list('red')            # ポイント色から目的地を格納
    create_agt(Universe.map.person, num=10) # personを生成

    Universe.n_goal = 0                     # 到着人数を初期化
```

```
# 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.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['pointColor'].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 = self.get_rgb(one_color[1])
        one_point.link_list = []
        one_point.link_color_list = []
        one_point.link_type_list = []
        if one_color[1] == 'green':
            one_point.view_size = 0
        else:
            one_point.view_size = 10

        # graphに追加
        Universe.id_agt_dict[str(one_id)] = one_point
        Universe.graph.add_node(str(one_id))
```

←描画ツール2に合わせて
ポイント属性を格納

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。

```
# リンクの初期化
def init_link(self, map_json_dict):

    import networkx

    for one_feature in map_json_dict['linkArray']:
        one_coordinates_array = one_feature['coordinates']
        one_link_type = one_feature['linkType']
        one_link_color = one_feature['linkColor'].split(':')
        one_color = self.get_rgb(one_link_color[1])
        one_pair = one_feature['pair']
        one_distance = one_feature['distance']

        one_id = one_pair[0]
        two_id = one_pair[1]
        one_point = Universe.id_agt_dict[str(one_id)]
        two_point = Universe.id_agt_dict[str(two_id)]

        # graphとlink_listに追加
        if one_link_type == 'both' or one_link_type == 'forward':
            Universe.graph.add_edge(str(one_id), str(two_id), weight=one_distance)
            one_point.link_list.append(two_point)
            one_point.link_color_list.append(one_color)
            one_point.link_type_list.append(one_link_type)

        if one_link_type == 'both' or one_link_type == 'reverse':
            Universe.graph.add_edge(str(two_id), str(one_id), weight=one_distance)
            two_point.link_list.append(one_point)
            two_point.link_color_list.append(one_color)
            two_point.link_type_list.append(one_link_type)
```

←両矢印／片矢印に対応

② エージェントのルールを記述する（4）

- 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。

```
104  # 色名からRGB取得
105  def get_rgb(self, one_color):
106
107      if one_color=='green':
108          one_rgb = rgb(0, 128, 0)
109      elif one_color=='red':
110          one_rgb = rgb(255, 0, 0)
111      elif one_color=='blue':
112          one_rgb = rgb(0, 0, 255)
113      elif one_color=='yellow':
114          one_rgb = rgb(255, 255, 0)
115      elif one_color=='cyan':
116          one_rgb = rgb(0, 255, 255)
117      elif one_color=='magenta':
118          one_rgb = rgb(255, 0, 255)
119
120      return(one_rgb)
121
122
123  # 色名からgoal_id_listを生成
124  def set_goal_id_list(self, one_color):
125
126      Universe.goal_id_list = []
127
128      point_set = make_agtset(agttype=Universe.map.point)
129      for one_point in point_set:
130          if one_point.color == self.get_rgb(one_color):
131              Universe.goal_id_list.append(one_point.point_id)
132
```

←ポイント属性の色名を使って色を塗る

←指定された色名でgoal_id_listを作成

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。

```
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

    # 最も近い目的地を決めて経路を設定
    self.route = self.get_route_from_goal_id_list(start_id)
    self.route_count = 0


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:
            # 目的地に到着したら終了
            Universe.n_goal += 1
            del_agt(self)
            return
        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)
```

←出発地はランダムに決め
目的地はgoal_id_listから
最短のポイントを選択

② エージェントのルールを記述する（6）

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。

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

    while(True):
        goal_id = self.get_random_id()
        if start_id != goal_id: # 出発地と目的地が同一でない場合
            break

    self.route = self.get_route(start_id, goal_id)   # 最短経路を取得
    self.route_count = 0

# 2点間の最短経路を取得
def get_route(self, start_id, goal_id):

    import networkx as nx

    try:
        # ダイクストラ法を使って最短経路を探索
        one_route = nx.dijkstra_path(Universe.graph, str(start_id), str(goal_id), weight='distance')

        return(one_route)
    except:
        print('[error] get_route: point_id=',start_id, 'と', goal_id, 'がつながっているか確認してください')
        exit_simulation()


# point_idをランダムに取得（但し、goal_id_list以外）
def get_random_id(self):

    r = random.choice(list(Universe.id_agt_dict.keys()))
    while r in Universe.goal_id_list:
        r = random.choice(list(Universe.id_agt_dict.keys()))

    return(r)
```

←目的地をランダムに設定
（但し、出発地を除く）

←ダイクストラ法を使って
２点間の最短経路を取得

←出発点をランダムに設定
（但し、goal_id_listの
目的地を除く）

② エージェントのルールを記述する（7）

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。

```
# 2点間の最短経路の距離を取得
def get_route_length(self, start_id, goal_id):

    import networkx as nx

    try:
        # ダイクストラ法を使って最短経路を探索
        one_length = nx.dijkstra_path_length(Universe.graph, str(start_id), str(goal_id), weight='distance')

        return(one_length)
    except:
        print('[error] get_route_length: point_id=',start_id, 'と', goal_id, 'がつながっているか確認してください')
        exit_simulation()

# 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
def get_route_from_goal_id_list(self, start_id):

    distance_dict = {}

    for one_goal_id in Universe.goal_id_list:
        distance_dict[one_goal_id] = self.get_route_length(start_id, one_goal_id)

    distance_dict2 = sorted(distance_dict.items(), key=lambda x:x[1])
    one_goal_id, one_length = distance_dict2[0]

    one_route = self.get_route(start_id, one_goal_id)

    return(one_route)
```

←ダイクストラ法を使って
2点間の最短距離を取得

←goal_id_listの中から
最も近い目的地を選択

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 ←地理院地図から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：
    - 変数指定：　view_size
  - 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）

- 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

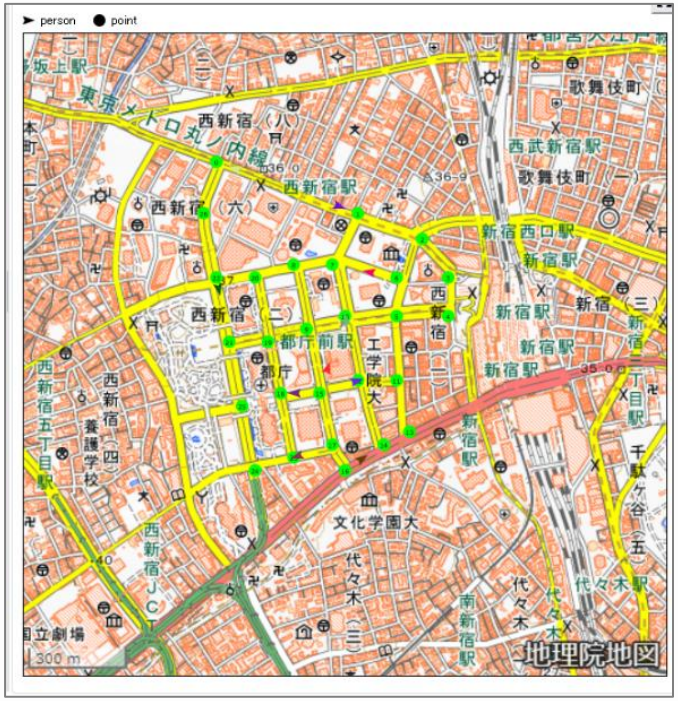
  要素名：　到着人数
  出力対象：Universe.n_goal
  小数表示：0 桁

- シミュレーションを実行します。

[Tips] 3D表示したいとき

第2回 artisoc Cloud勉強会にて、Cesiumを利用して3D表示する手順について解説しました。

- 勉強会資料
  第２回 artisoc Cloud勉強会



06. レシピブック　サンプルモデル2

Cesiumを利用した3D表示