

artisoc Cloudレシピブック

08. 複数階でポテンシャル法

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

Copyright© KOZO KEIKAKU ENGINEERING Inc.All Rights Reserved.

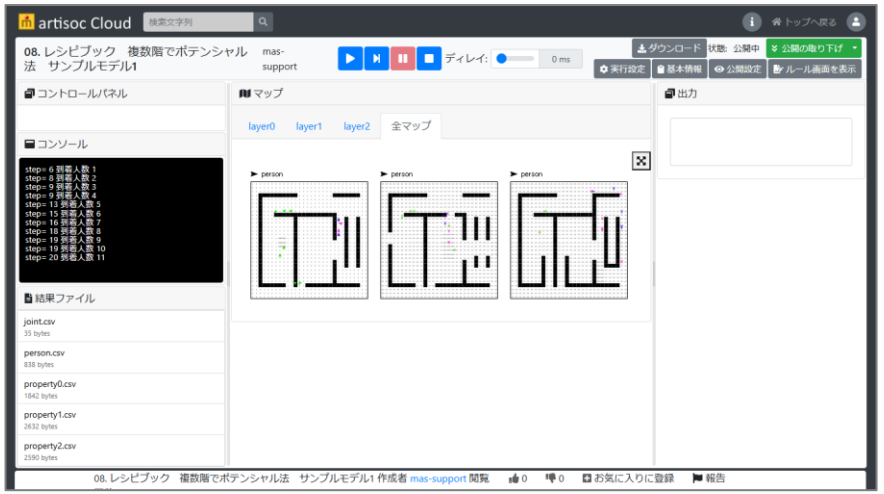
この資料には、弊社のノウハウ、営業機密等が含まれておりますので、お取り扱いには十分ご留意願います。この資料およびその内容を、弊社に無断で使用、複写、破壊、改ざんすること、ならびに第三者へ開示すること、漏洩すること、あるいは使用させることは、固くお断り申しあげます。

ポテンシャル法で移動しよう

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

05. レシピブック にて、ポテンシャル法を使った移動モデルについて解説しましたが、複数階をまたいだ移動の要望が多いため、複数階に対応しました。

1. ポテンシャル法とは？
2. 入力ファイルを作成する
3. モデルを定義する
4. Universeのルールを記述する
5. personのルールを記述する
6. 出力画面を定義する



08. レシピブック　サンプルモデル1

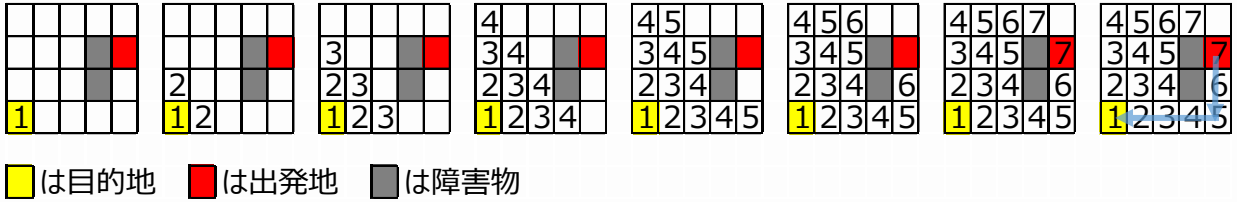
Copyright© KOZOKEIKAKU ENGINEERING Inc. All Rights Reserved.

1. ポテンシャル法とは？

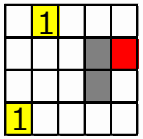
ボールが高いところから低いところへ転がっていく性質を利用した
探索方法です

ポテンシャル法で最短経路を求める手順

1. 空間をメッシュ状に切り、目的地、出発地、障害物のセルを決めます。
2. 目的地に「1」を代入し、その周り上下左右（４方向）に「＋１」した値を代入します。
3. 順番に値を決めていき、出発地の値が決まるまで繰り返します。
4. 出発地から値の小さいセルを順番に辿れば最短経路が見つかります。



※ 目的地が複数あるときはどうなるか試してみましょう。



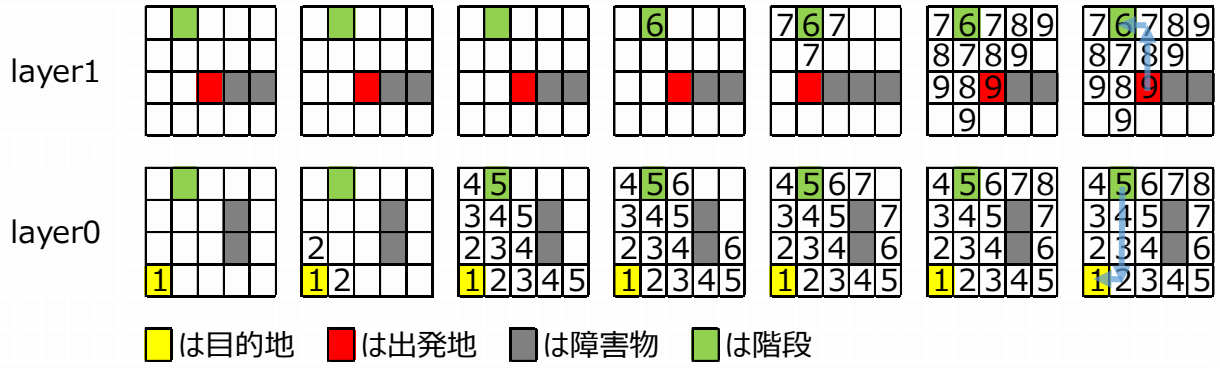
Copyright© KOZOKEIKAKU ENGINEERING Inc. All Rights Reserved.

ポテンシャル法を複数階で使うためには？

2つの階がつながっている場所（階段やエレベータ等）を定義して、
ポテンシャルを階をまたいで作成します。

複数階でポテンシャル法で最短経路を求める手順

1. 空間をメッシュ状に切り、目的地、出発地、障害物、階段のセルを決めます。
2. 目的地に「1」を代入し、その周り上下左右（４方向）に「＋１」した値を代入します。
3. 階段のセルのときは、つながっている階のセルにも「＋１」した値を代入します。
4. 順番に値を決めていき、出発地の値が決まるまで繰り返します。
5. 出発地から値の小さいセルを順番に辿れば最短経路が見つかります。



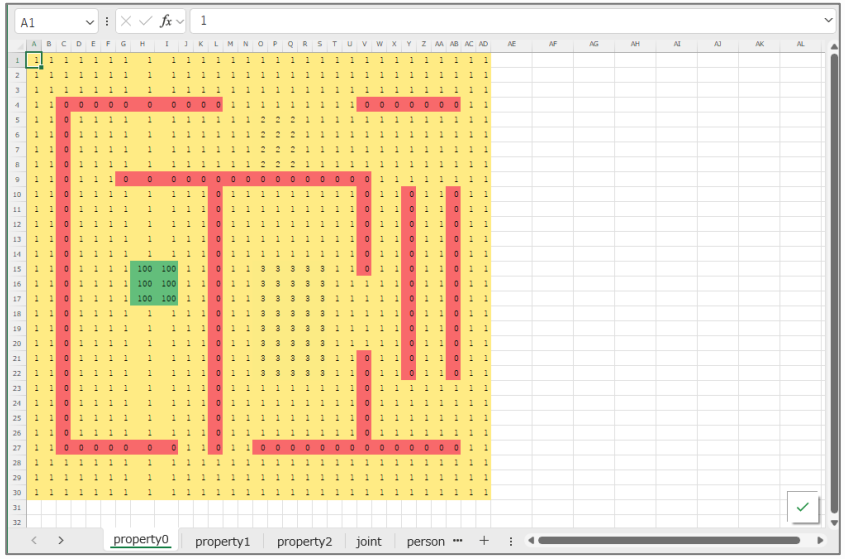
Copyright© KOZOKEIKAKU ENGINEERING Inc. All Rights Reserved.

2. 入力ファイルを作成する

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

Excelを使って入力ファイル（input.xlsx）を作成します。

- input.xlsx をExcelで開きます。
- input.xlsx には、3種類のデータがあります。
    - ① property：空間情報（障害物、階段、出発地、目的地）を定義
    - ② joint：　つながっている階のIDを定義
    - ③ person：　personエージェントの移動を定義



入力ファイル（input.xlsx）

Copyright© KOZOKEIKAKU ENGINEERING Inc. All Rights Reserved.

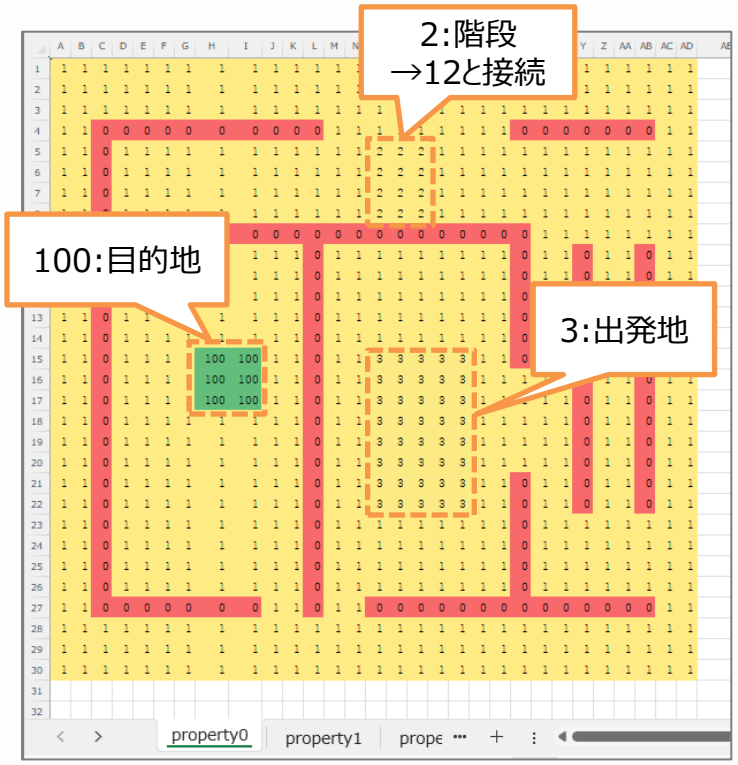
2-1. propertyシートを作成する（1）

- サンプルモデルでは、レイヤ数が3（3階層）なので、property0、1、2の3つのシートが定義されています。
- また、空間の大きさは30x30です。
- セルの色は、下記の指定で着色しています。
  ホームタブ > 条件付き書式 > カラースケール

- propertyの値（ID）は0が歩行不可のセル、1以上の値は歩行可能なセルです。

- property0シートでは、次の通りIDを定義しています。
  0：壁
  1：床
  2：階段（property1の「12」に接続）
  3: 出発地
  100：目的地

  ※ 0:壁、1:床 は全てのpropertyで共通



property0シート

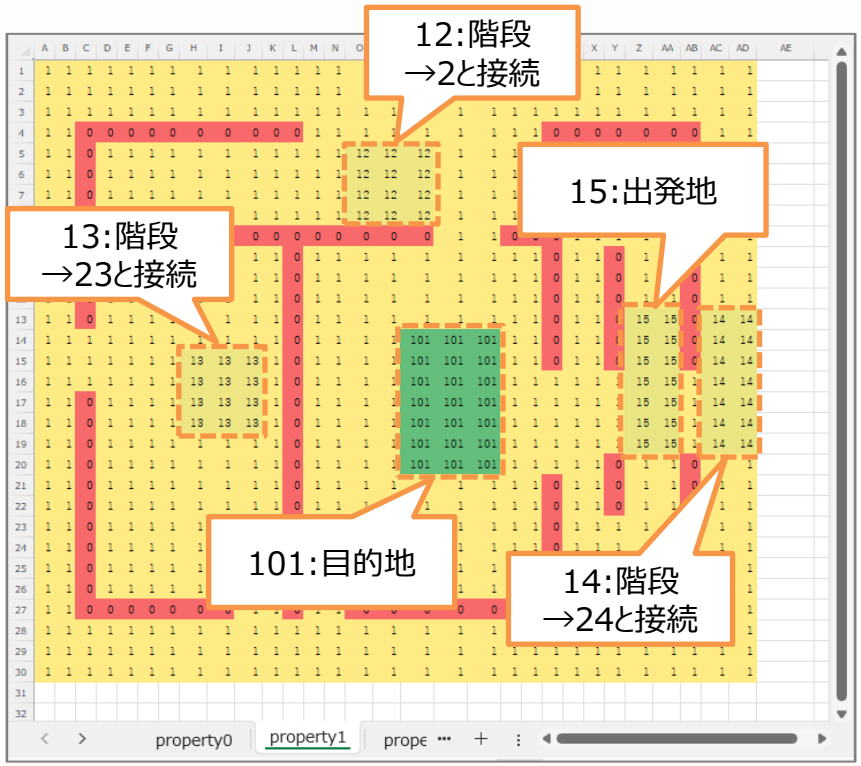
Copyright© KOZOKEIKAKU ENGINEERING Inc. All Rights Reserved.

2-1. propertyシートを作成する（2）

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

- property1シートでは、次の通りIDを定義しています。
  0：壁
  1：床
  12：階段（property0の「2」に接続）
  13：階段（property2の「23」に接続）
  14：階段（property2の「24」に接続）
  15: 出発地
  101：目的地

※ レイヤ間を移動する時に同じ(x,y)座標を取るため、同じセル範囲にIDを定義する必要があります。
例えば、property1の「12」とproperty0の「2」の場合は「O5:Q8」の範囲に値を定義します。

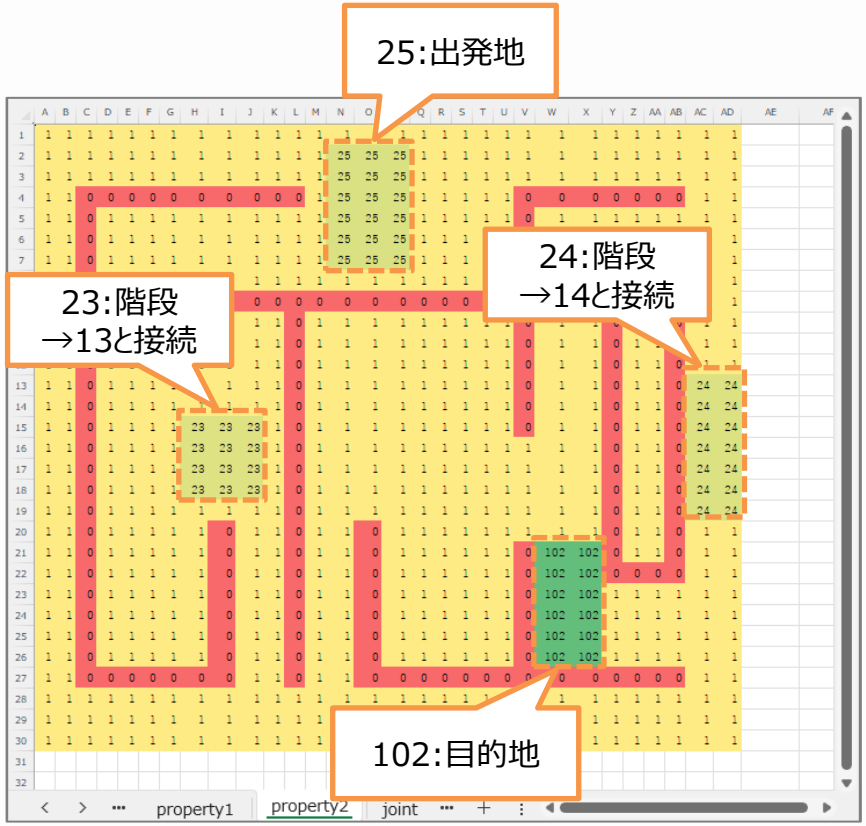


property1シート

Copyright© KOZOKEIKAKU ENGINEERING Inc. All Rights Reserved.

2-1. propertyシートを作成する（3）

- property2シートでは、次の通りIDを定義しています。
  0：壁
  1：床
  23：階段（property1の「13」に接続）
  24：階段（property1の「14」に接続）
  25: 出発地
  102：目的地



property2シート

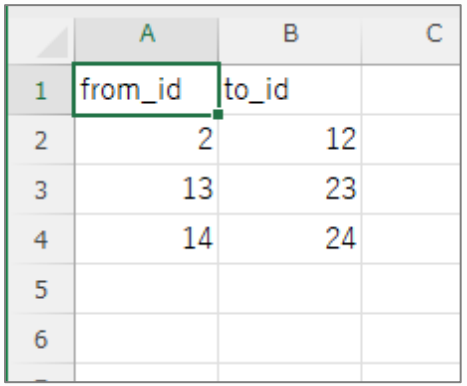
Copyright© KOZOKEIKAKU ENGINEERING Inc. All Rights Reserved.

2-2. jointシートを作成する

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

- jointシートでは、つながっている階のIDを定義します。
  2（property0） ー 12（property1）
  13（property1） ー 23（property2）
  14（property1） ー 24（property2）

- personエージェントは上下階へ移動ができます。



	A	B	C
1	from_id	to_id	
2	2	12	
3	13	23	
4	14	24	
5			
6			

jointシート

Copyright© KOZOKEIKAKU ENGINEERING Inc. All Rights Reserved.

2-2. personシートを作成する

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

- personシートでは、personエージェントの移動を定義します。

- 以下の値を定義します。

  person_id：　personの識別番号
  step：　発生するstep
  speed：　personの移動速度
  start_id：　出発地のID
  goal_id：　目的地のID

  ※ 同じstep、同じstart_idを定義することで
  1ステップに複数のpersonを生成できます

	A	B	C	D	E	F
1	person_id	step	speed	start_id	goal_id	
2	1	1	1.24	3	100	
3	2	1	1.33	15	100	
4	3	1	1.37	25	100	
5	4	1	1.28	3	101	
6	5	1	1.27	15	101	
7	6	1	1.3	25	101	
8	7	1	1.31	3	102	
9	8	1	1.22	15	102	
10	9	1	1.39	25	102	
11	10	2	1.24	3	100	
12	11	2	1.33	15	100	
13	12	2	1.37	25	100	
14	13	2	1.28	3	101	
15	14	2	1.27	15	101	
16	15	2	1.3	25	101	
17	16	2	1.31	3	102	
18	17	2	1.22	15	102	
19	18	2	1.39	25	102	
20	19	5	1.24	3	100	
21	20	5	1.33	15	100	
22	21	5	1.37	25	100	
23	22	5	1.28	3	101	

property2　joint　person

personシート

Copyright© KOZOKEIKAKU ENGINEERING Inc. All Rights Reserved.

2-3. csvファイルを作成する

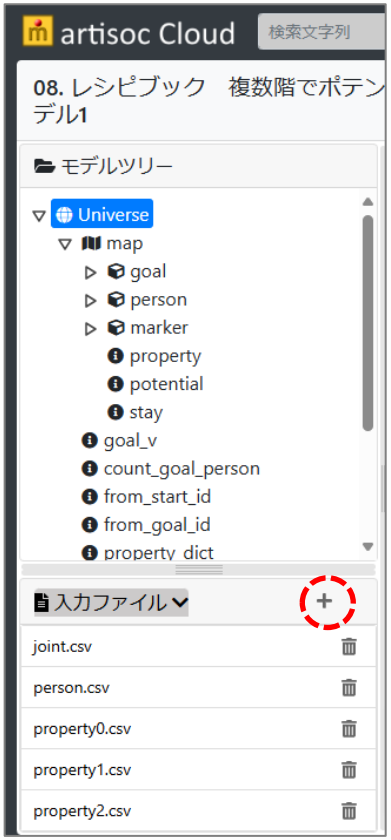
構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

- シート名をファイル名をに指定して、csvファイルを作成します。
  【操作】
  ファイルタブ > 名前をつけて保存 > csv（コンマ区切り）(*.csv)

- 以下、作成するファイルの一覧です。
  property0.csv
  property1.csv
  property2.csv
  joint.csv
  person.csv

- 入力ファイルには同一ファイル名のファイルをインポートできないため、削除アイコンをクリックして削除してください。

- 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てください。



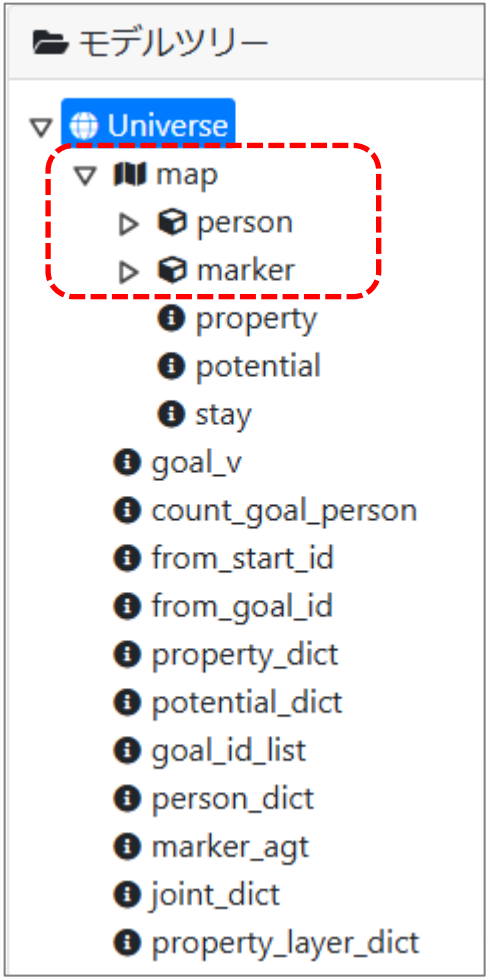
入力ファイルのインポート

Copyright© KOZOKEIKAKU ENGINEERING Inc. All Rights Reserved.

3. 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



- 空間（Universe.map）
  連続空間、大きさ30 x 30、レイヤ数3、ループしない

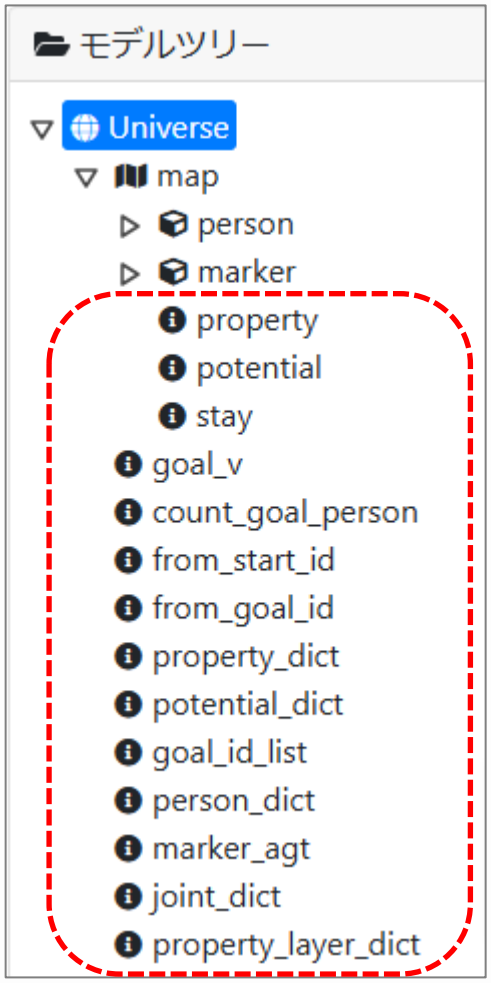
- エージェント（Universe.map）
  person：　歩行者エージェント
  - エージェント変数
    color：　マップ出力の表示色
    start_id：　出発地ID
    goal_id：　目的地ID
    person_id：　personのid
    step：　発生するstep
    speed：　移動速度

  marker　：　personを移動させるための移動マーカー

https://www.kke.co.jp

Copyright© KOZOKEIKAKU ENGINEERING Inc. All Rights Reserved.

3. モデルを定義する（2）



- 空間変数

property：	プロパティ値を格納　※ property.csvから読込
potential：	ポテンシャル値を格納　※モデル内で計算
stay　：	personの存在有無を格納

- Universe変数

goal_v：	目的地のポテンシャル値を格納
count_goal_person：	目的地に到着した人数を格納
from_start_id：	出発地IDの開始番号
from_goal_id：	目的地IDの開始番号
property_dict：	プロパティ値をキーに座標集合を格納
potential_dict：	プロパティ値をキーにポテンシャルを格納
goal_id_list：	目的地IDのリスト
person_dict：	stepをキーに行動計画のリストを格納 ※person.csvから読込
marker_agt：	移動マーカーのエージェントを格納
joint_dict：	つながっている階のIDを格納
property_layer_dict：	プロパティ値をキーにレイヤ番号を格納

Copyright© KOZOKEIKAKU ENGINEERING Inc. All Rights Reserved.

4. Universeのルールを記述する（1）

personが与えられた出発地から目的地へ移動し、
移動の途中に他のpersonと相互作用するモデルを作成します

- univ_initは、変数の初期化やファイル読込、ポテンシャルの計算等を行います。

```
def univ_init(self):

    # 変数の初期化
    Universe.from_start_id = 1      # 出発地IDの開始番号：0:歩行不可として定義しているため1以上とする
    Universe.from_goal_id = 100     # 目的地IDの開始番号：100以上を目的地とする
    Universe.count_goal_person = 0  # 移動完了人数のクリア
    Universe.property_dict = {}
    Universe.goal_id_list = []
    Universe.property_layer_dict = {}
    Universe.joint_dict = {}

    # personのファイル読込
    self.file_read_person('person.csv')

    # propertyのファイル読込
    self.file_read_property('property0.csv', 0)
    self.file_read_property('property1.csv', 1)
    self.file_read_property('property2.csv', 2)

    # jointのファイル読込
    self.file_read_joint('joint.csv')

    # potentialの計算
    self.set_potential()

    # 移動マーカーの生成
    Universe.marker_agt = create_agt(Universe.map.marker)

```


Copyright© KOZOKEIKAKU ENGINEERING Inc. All Rights Reserved.

4. Universeのルールを記述する（2）

- file_read_personは、person.csvを読み込み、step毎にまとめた上でperson_dictに格納します。

```
# personファイルの入力
def file_read_person(self, input_filename):

    import csv

    # ファイルの読込
    Universe.person_dict = {}
    try:
        with open(input_filename, encoding='shift_jis') as f:
            csvreader = csv.reader(f)
            row_ct = 0
            for row in csvreader:
                if row_ct > 0:
                    one_person_dict = {}
                    one_person_dict['person_id'] = row[0]
                    v_step = int(row[1])
                    one_person_dict['step'] = v_step
                    one_person_dict['speed'] = float(row[2])
                    one_person_dict['start_id'] = row[3]
                    one_person_dict['goal_id'] = row[4]

                    if str(v_step) in Universe.person_dict:
                        Universe.person_dict[str(v_step)].append(one_person_dict)
                    else:
                        Universe.person_dict[str(v_step)] = []
                        Universe.person_dict[str(v_step)].append(one_person_dict)
                row_ct += 1
    except:
        print('[Error] file_read_person :', input_filename)
        exit_simulation()
```


Copyright© KOZOKEIKAKU ENGINEERING Inc. All Rights Reserved.

4. Universeのルールを記述する（3）

- file_read_propertyは、空間変数を初期化し、goal_v（ポテンシャルの最も小さい値）を計算します。（道路が複雑な場合は、より小さい値をgoal_vに設定してください）
- 次にproperty.csvを読み込み、プロパティID毎にまとめて property_dict に格納します。このとき、目的地IDを判別して goal_id_list を作成します。

```
181  # propertyファイルの入力
182  def file_read_property(self, input_filename, one_layer):
183
184      import csv
185
186      w = get_width_space(Universe.map)
187      h = get_height_space(Universe.map)
188      layer = get_layer_space(Universe.map)
189
190      # 空間変数の初期化
191      for i in range(h):
192          for j in range(w):
193              Universe.map.property[j, i, one_layer] = 0
194              Universe.map.potential[j, i, one_layer] = 0
195              Universe.map.stay[j, i, one_layer] = 0
196
197      # 目的地のポテンシャル値を計算
198      Universe.goal_v = -1 * (w + h) * layer
199
200      # ファイルの読込
201      try:
202          with open(input_filename, encoding='shift_jis') as f:
203              csvreader = csv.reader(f)
204              row_ct = 0
205              for row in csvreader:
206                  for j in range(w):
207                      v = int(row[j])
208                      i = h - row_ct - 1
209                      key = str(j) + ',' + str(i) + ',' + str(one_layer)
210                      Universe.map.property[j, i, one_layer] = v
211                      # property_dictの作成
```


Copyright© KOZOKEIKAKU ENGINEERING Inc. All Rights Reserved.

4. Universeのルールを記述する（4）

- set_potentialは、goal_id_listに格納した目的地IDをcalc_potentialに渡して、目的地ID毎のポテンシャルを計算します。

```
150  # ポテンシャルの計算
151  def set_potential(self):
152
153      Universe.potential_dict = {}
154
155      # 目的地ID毎にポテンシャルを計算
156      for one_goal_id in Universe.goal_id_list:
157          self.calc_potential(one_goal_id)
```

- calc_potentialは、プロパティ値が0以外（歩行可能）のセルのポテンシャル値を求めます。計算結果はpotential_dictに格納します。

```
 93  # 任意の目的地IDのポテンシャルを計算
 94  def calc_potential(self, goal_id):
 95
 96      w = get_width_space(Universe.map)
 97      h = get_height_space(Universe.map)
 98      layer = get_layer_space(Universe.map)
 99
100      # 計算対象のセルを取得
101      target_list = []
102      cell_list = []
103      for one_layer in range(layer):
104          for i in range(h):
105              for j in range(w):
106                  v = Universe.map.property[j, i, one_layer]
107                  if v > 0:
108                      key = str(j) + ',' + str(i) + ',' + str(one_layer)
109                      if v == goal_id:
110                          target_list.append(key)
111                          Universe.map.potential[j, i, one_layer] = Universe.goal_v
112                      else:
113                          cell_list.append(key)
```


Copyright© KOZOKEIKAKU ENGINEERING Inc. All Rights Reserved.

4. Universeのルールを記述する（5）

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

- univ_step_beginでpersonの生成、univ_step_endで終了条件をチェックします。

```
20 def univ_step_begin(self):
21
22     # step毎にpersonを生成
23     self.create_person_step()
24
25
26 def univ_step_end(self):
27
28     # 終了条件のチェック
29     if count_agt(Universe.map.person) == 0:
30         exit_simulation()
31
32
33 def univ_finish(self):
34     pass
```

※ 終了条件は、personの数が0になった場合で判定しているため、step=1でpersonを生成しない場合は、終了条件を変更してください。

Copyright© KOZOKEIKAKU ENGINEERING Inc. All Rights Reserved.

4. Universeのルールを記述する（6）

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

- create_person_stepは、person_dictに格納したstep毎の行動計画を使って、任意のstepでpersonを生成します。このとき、personを生成するセルは出発地IDと同じ値かつ他のpersonが存在しないセルを選択します。

```
37  # step毎にpersonを生成
38  def create_person_step(self):
39
40      import random
41
42      ct_step = str(count_step())
43      if ct_step in Universe.person_dict: # 該当のstepでpersonを生成する場合
44          person_list = Universe.person_dict[ct_step]
45          for one_person_list in person_list:
46              one_person = create_agt(Universe.map.person)
47              one_person.person_id = one_person_list['person_id']
48              one_person.step = one_person_list['step']
49              one_person.speed = one_person_list['speed']
50              one_person.start_id = one_person_list['start_id']
51              one_person.goal_id = one_person_list['goal_id']
52
53              start_cell_list = Universe.property_dict[str(one_person.start_id)]
54              for i in range(10):
55                  one_cell = random.choice(start_cell_list)   # 出発地IDのセルをランダムで取得
56                  pos_list = one_cell.split(',')
57                  px = int(pos_list[0])
58                  py = int(pos_list[1])
59                  p_layer = int(pos_list[2])
60                  if Universe.map.stay[px, py, p_layer] == 0: # 該当セルにpersonが不在の場合
61                      break
62              one_person.x = px
63              one_person.y = py
64              one_person.layer = p_layer
65              one_person.color = self.get_color(one_person.goal_id)
66              Universe.map.stay[px, py, p_layer] += 1
```


Copyright© KOZOKEIKAKU ENGINEERING Inc. All Rights Reserved.

5. personのルールを記述する

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

- personのagt_stepでは、自らの目的地のポテンシャルに沿って、移動します。
  周囲8方向および上下階につながっている場合は該当の方向のうち、歩行可能で他のpersonが存在しないもしくは少ないセルの中からポテンシャル値が最も低いセルを選び、speedの距離だけ移動します。
  ゴールに到着すると到着人数を加算し、エージェントを消去します。

```
4  def agt_step(self):
5      w = get_width_space(Universe.map)
6      h = get_height_space(Universe.map)
7
8      # 目的地のポテンシャルを取得
9      potential_map = Universe.potential_dict[str(self.goal_id)]
10
11     # ポテンシャル値の取得関数
12     potential_dict = {}
13     def get_potential(cx, cy, c_layer):
14         if 0 <= cx and cx < w and 0 <= cy and cy < h:
15             cx = int(cx)
16             cy = int(cy)
17             if Universe.map.stay[cx, cy, c_layer] < rand()*3:       # 人混みを避ける
18                 key = str(cx) + ',' + str(cy) + ',' + str(c_layer)
19                 if potential_map[key] < 0:
20                     potential_dict[key] = potential_map[key]
21
22     distance = self.speed
23     goal_flag = False
24     while(distance > 0):
25         # 周囲8方向でポテンシャル値が低い方へ移動
26         px = self.x
27         py = self.y
28         p_layer = self.layer
29         get_potential(px-1, py, p_layer)   # 左
30         get_potential(px+1, py, p_layer)   # 右
31         get_potential(px, py+1, p_layer)   # 上
32         get_potential(px, py-1, p_layer)   # 下
33         get_potential(px-1, py+1, p_layer)   # 左上
```

※ ポテンシャルの計算は周囲4方向で求めましたが、personの移動は周囲8方向を対象としています。これは、移動方向を周囲4方向にすると、personの動きがカクカクした動きになるため、周囲8方向にしています。

Copyright© KOZOKEIKAKU ENGINEERING Inc. All Rights Reserved.

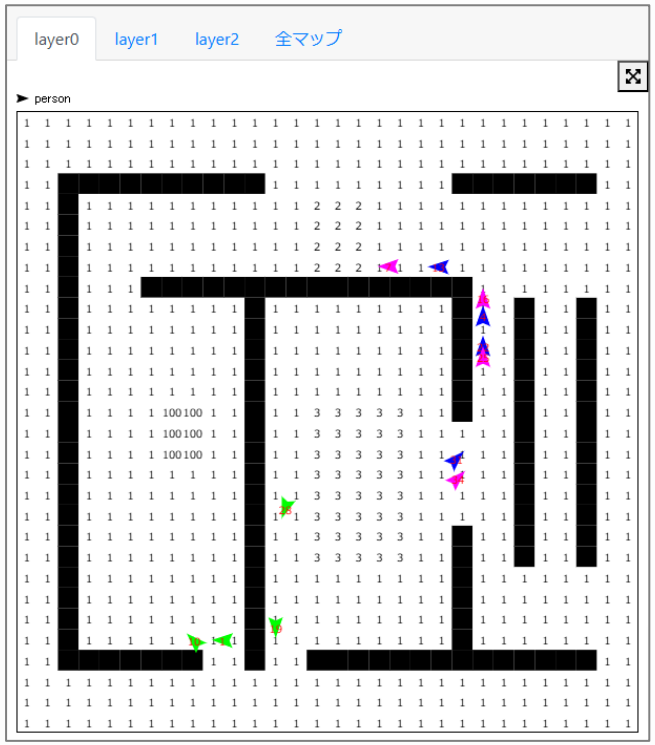
6. 出力画面を定義する

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.

マップ出力画面を定義します

- 出力設定 > layer0 を表示します。
  空間「map」、レイヤ番号「0」
  マップ要素リストは、以下の通り定義します。
  person：　マーカー「矢印」、表示色「color」
  　　エージェント情報の表示「person_id」
  property：グラデーション指定「0:黒」～「1:白」、
  　　変数表示「property」
  ※ layer1、layer2もレイヤ番号を変えて同様に
  　定義します

- 出力設定 > 値出力 を表示します。
  値出力要素リストは、以下の通り定義します。
  step：　出力値「count_step()」



マップ出力（layer0）

Copyright© KOZOKEIKAKU ENGINEERING Inc. All Rights Reserved.