artisoc Cloud初級チュートリアル

（株）構造計画研究所
創造工学部

構造計画研究所
KOZO KEIKAKU ENGINEERING Inc.
株式会社　構造計画研究所
〒164-0011 東京都中野区中央 4-5-3

https://www.kke.co.jp

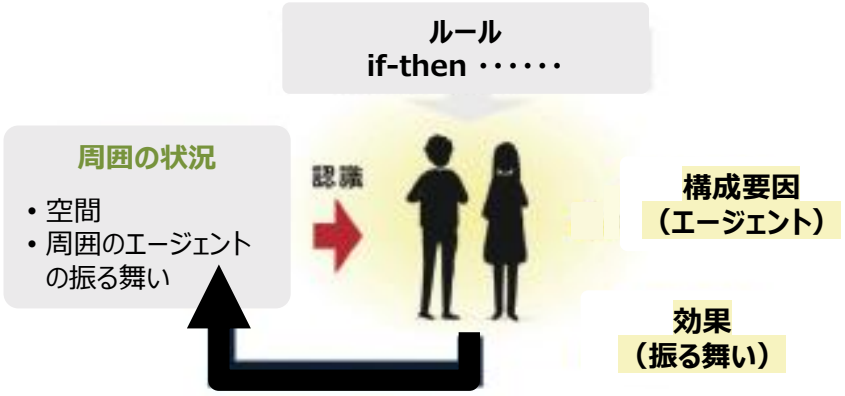
□本日のチュートリアル

- 0．イントロダクション
- 1．モデルの作成手順を学ぶ
  - 空間・エージェントを作成
  - 出力形式を決定
  - エージェントの行動ルールを作成
- 2．基本的なテクニックを学ぶ
  - 変数設定の変更
  - 関数の使用
  - エージェントの生成の仕方の変更
  - コントロールパネルの設定
- 3．相互作用を含むモデルを作成する
  - エージェント集合型の変数の活用
  - 条件分岐文の活用
  - エージェント間の相互作用
- おわりに

0. イントロダクション

❑マルチエージェントシミュレーション（MAS）とは

個々の「エージェント」の振る舞いを定義

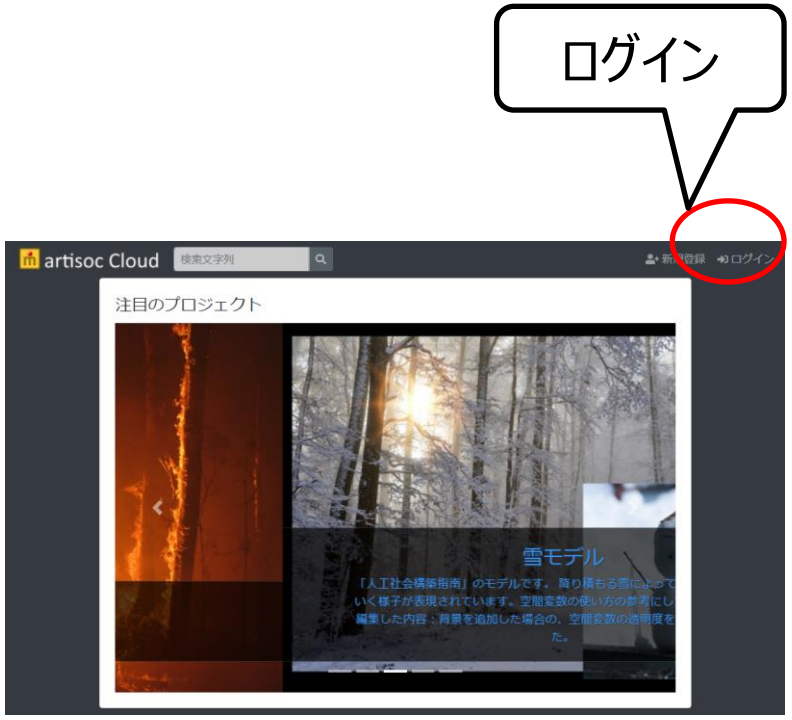


エージェント同士の相互作用による
複雑な社会現象を再現

コンピューターの中に人工社会を構築し、社会現象のメカニズムを理解する

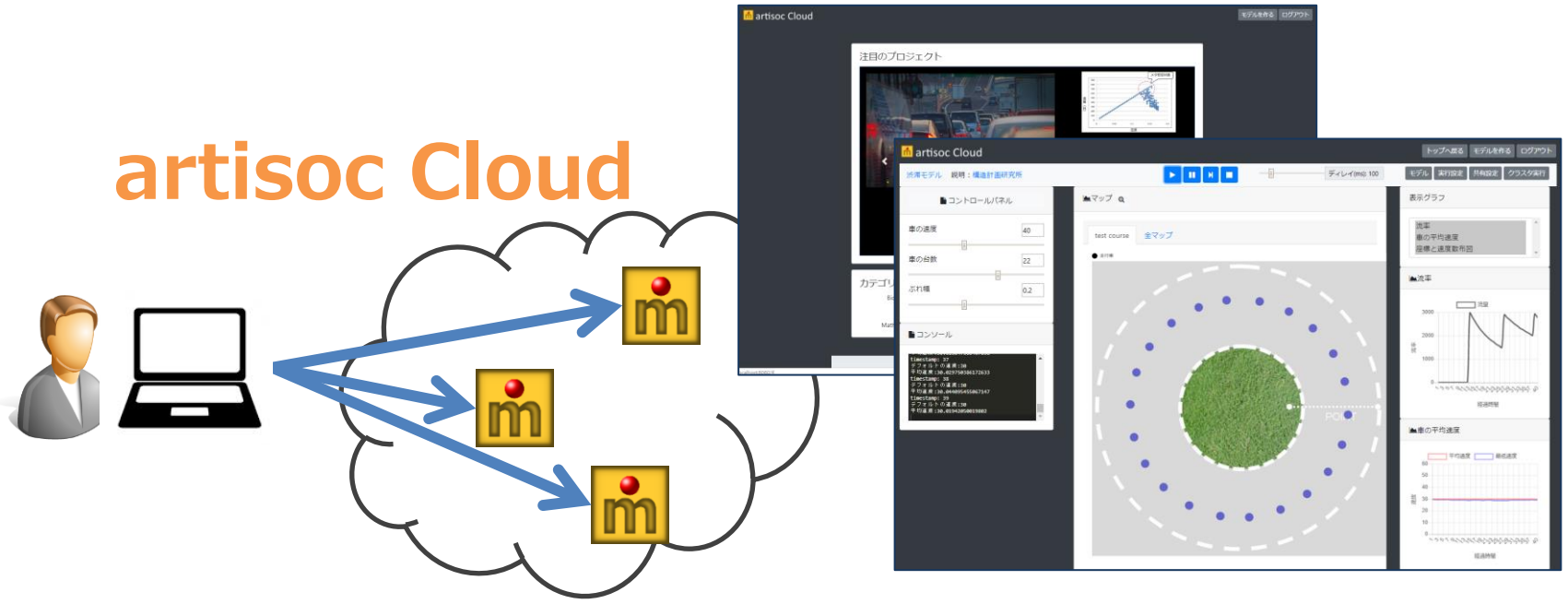
❑受講に向けた準備

- artisoc Cloudというソフトウェアを使います。
  - https://artisoc-cloud.kke.co.jp/ から「artisoc Cloud」に接続してください。
  - 動作環境は、Google ChromeもしくはFirefoxブラウザを使ってください。



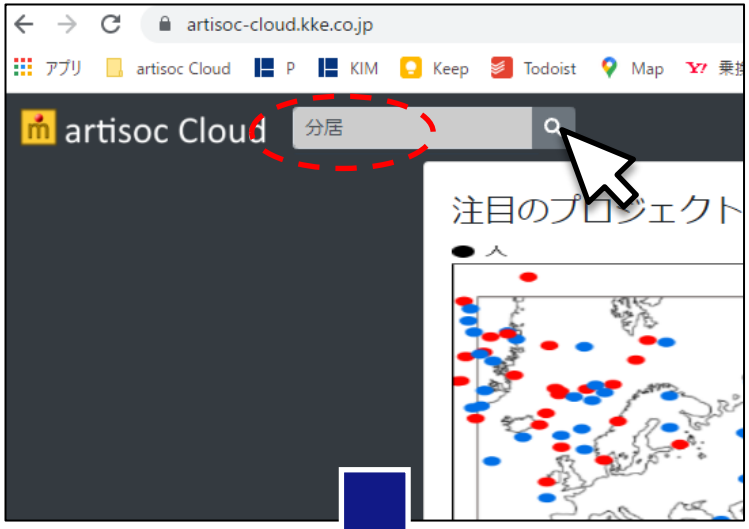
❑MASプラットフォーム　artisoc Cloud

- 社会シミュレーションのプラットフォーム
  - 利用者がWeb上で簡単に社会シミュレーションを使うことができる。その結果を共有・議論することができる。
  - クラウドのパワーを使ったより高度な解析を簡単に実行できる。
  - モデルをクラウドでテンプレートとして共有し、派生したモデルを簡単に構築できる。

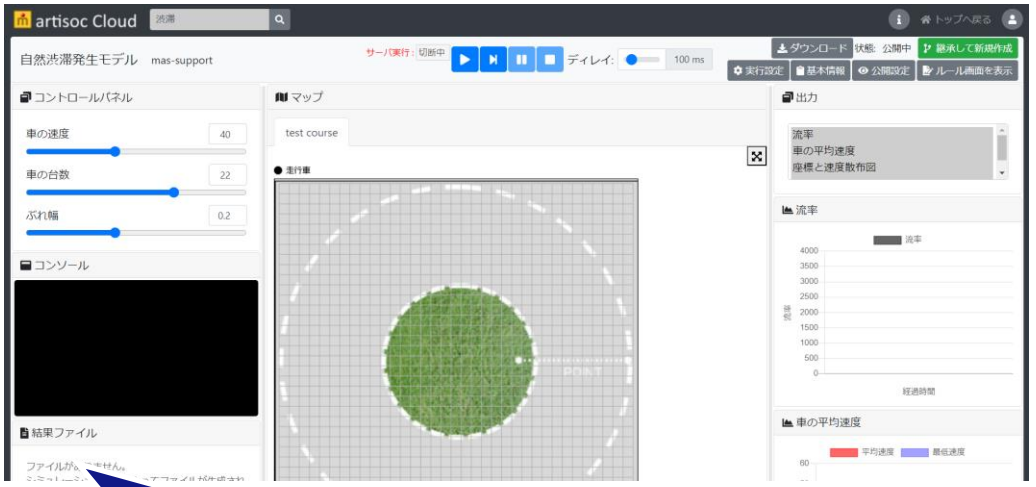


❒artisoc Cloudのモデルを動かす

■ トップページから「渋滞」で検索、「自然渋滞発生モデル」をクリックします。



自然渋滞発生モデルが開きます



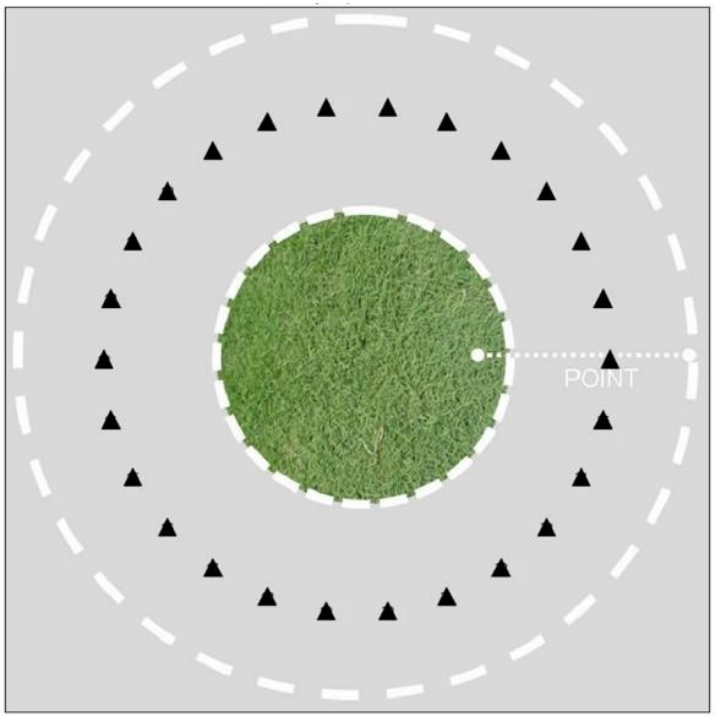
自然渋滞発生モデル

mas-support 閲覧回数 4 更新日 2021-08-11

高速道路などで発生する自然渋滞を、マルチエージェント・シミュレーションのモデルとして再現したモデルです。 詳しい解説はこちら↓

https://mas.kke.co.jp/model/自然渋滞発生モデル/

❑自然渋滞発生モデル



渋滞の写真（C)ヌンヌン“渋滞",
http://www.flickr.com/photos/nunnun/2152291102/

ルール

1. 前の車を追従　(ただし速度にはぶれ幅があり)
2. 車間距離が短くなるとブレーキ
3. 車間距離が長くなると加速

シンプルなルールで渋滞の本質を表現

シチュエーションやパラメータは東京大学西成教授の実証実験を参考
参考文献　渋滞学, 西成 活裕著, 新潮選書, 2006

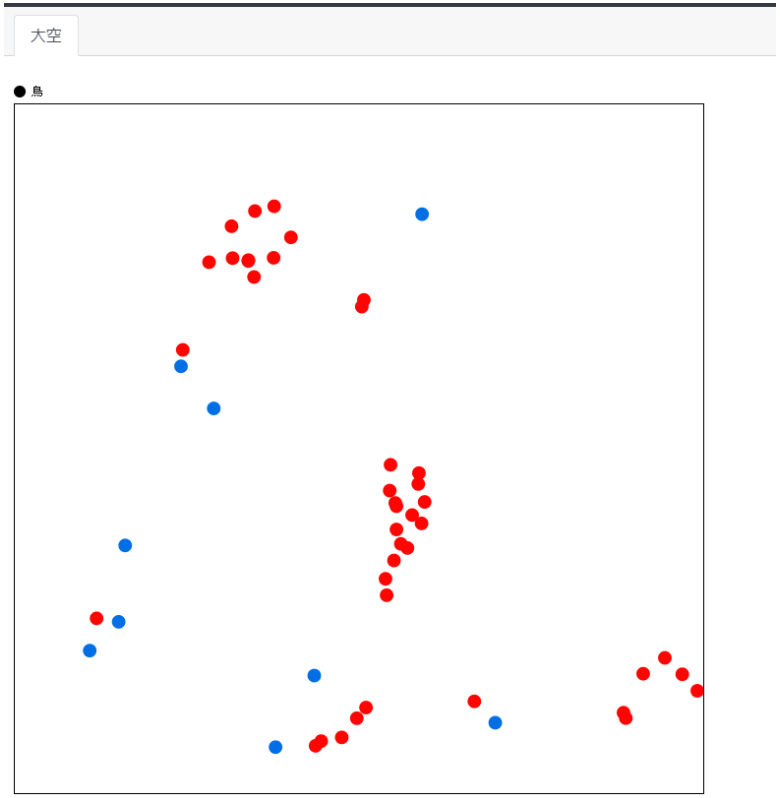
❑本日のチュートリアルについて

■ 概要

- ❑『飛ぶ鳥モデル』の作成を通じ、artisoc Cloudの基本的テクニックを習得する。
  - ➡ 飛ぶ鳥モデル
    ：「大空(oozora)」という空間上に「鳥(tori)」というエージェントが存在し、様々な向きに飛んでいく。

■ 講習の流れ

1. モデルの作成手順を学ぶ
2. 基本的なテクニックを学ぶ
3. 相互作用を含むモデルを作成する



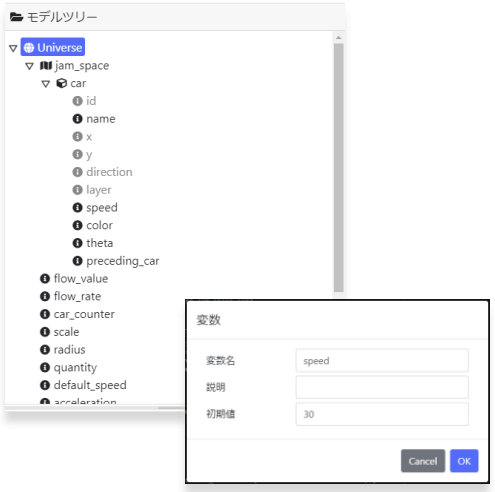
Ⅰ．モデルの作成手順を学ぶ

❑まずは動かしてみましょう

- 以下のような簡単なモデルを作成し、artisoc Cloudのモデル作成手順を学びます。
  - 「大空(oozora)」という空間上に「鳥(tori)」というエージェントが1体だけ存在する。
  - 「鳥」は空間の中央から出発して前方にまっすぐ飛んでいく。

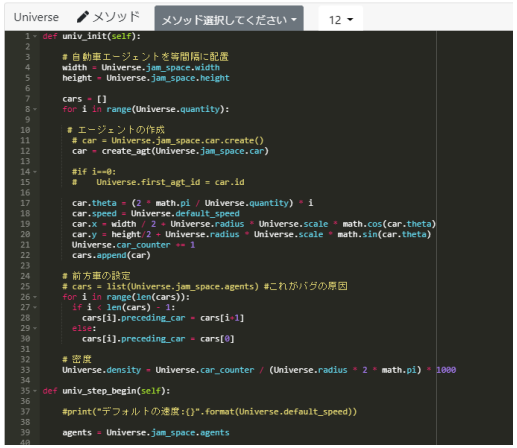
□モデル構築の流れ

① 空間／エージェントの **種類・属性**を作成



② シミュレーション結果の **出力形式**を決定

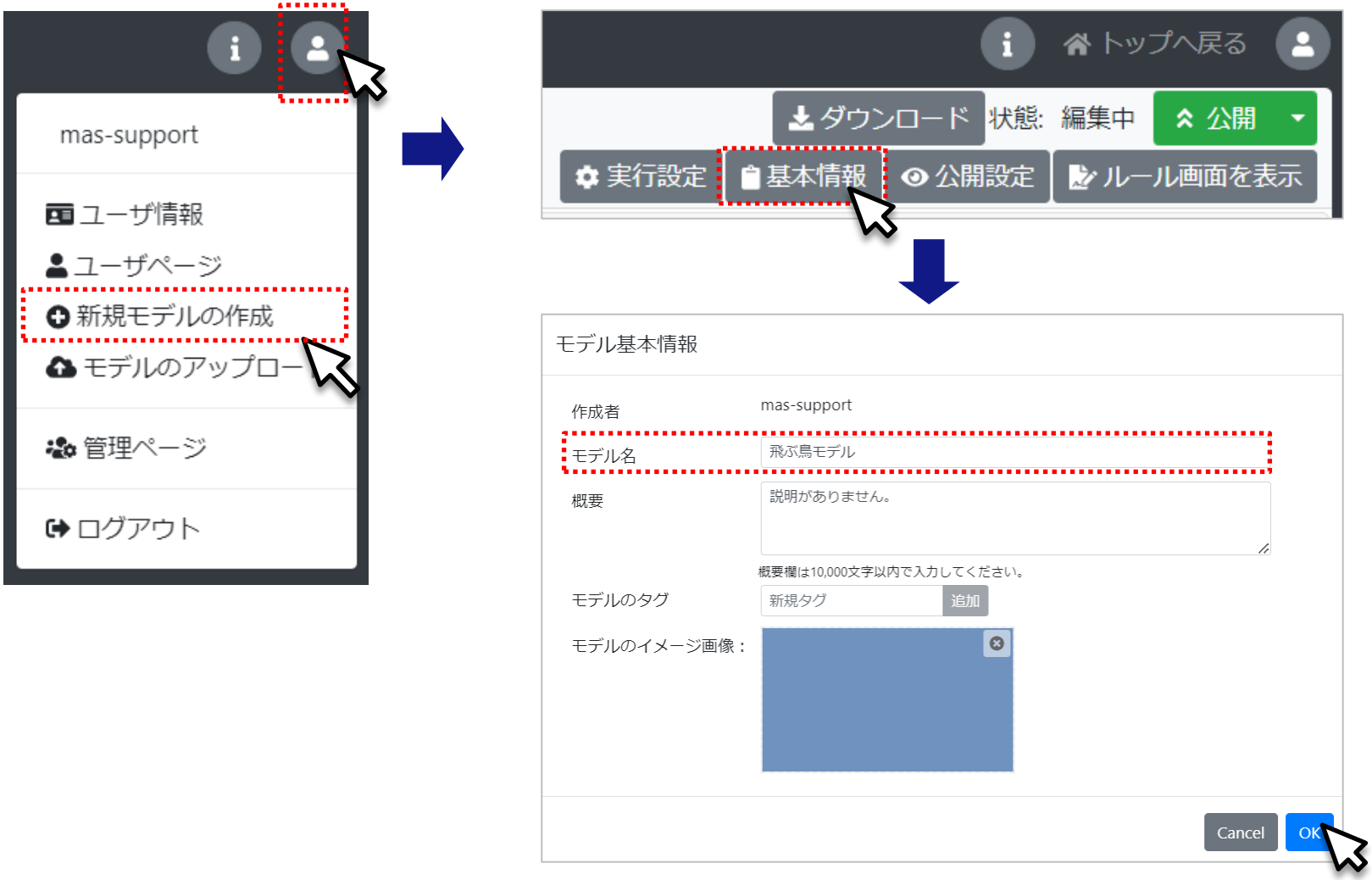
③ エージェントの **行動ルール**を作成



実行ボタンをクリック！

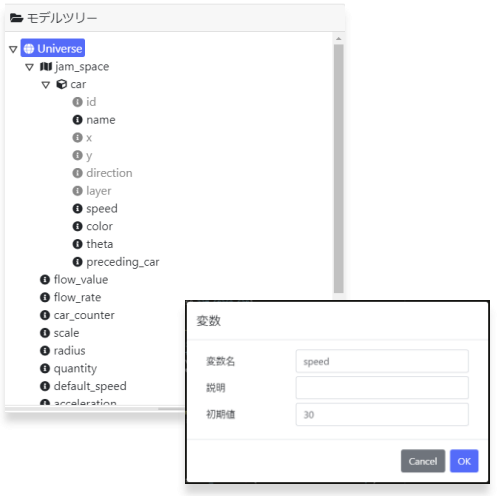
❒準備

- 右上のアイコンマークから「新規モデルの作成」をクリックし、モデル名を入力します。
  - 「飛ぶ鳥モデル」と名付けましょう。



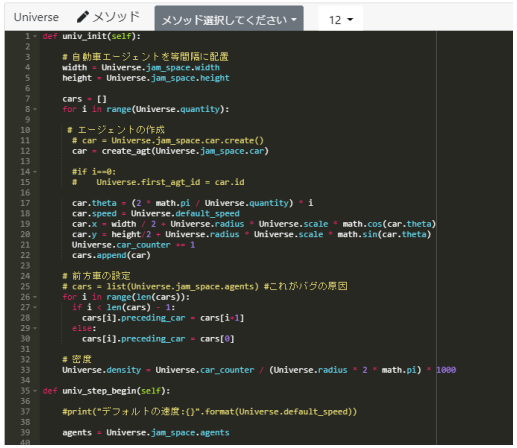
❑空間／エージェントの種類・属性を作成

① 空間／エージェントの
種類・属性を作成



② シミュレーション結果の
出力形式を決定

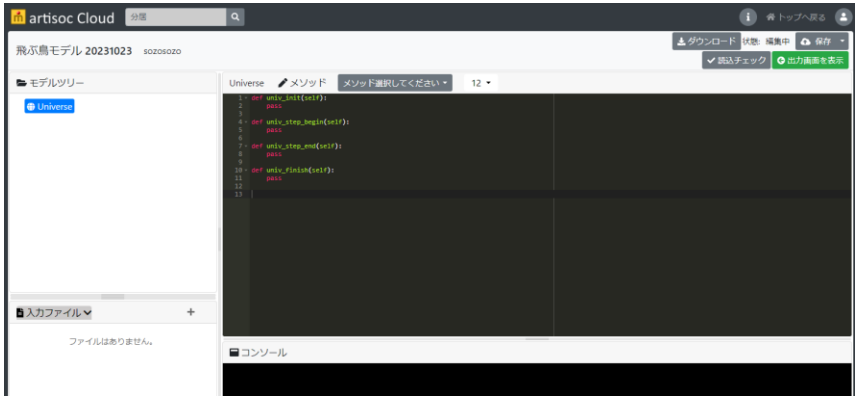
③ エージェントの
行動ルールを作成



ルール画面の表示

■「ルール画面を表示」をクリックしてモデルのルール画面に遷移します。

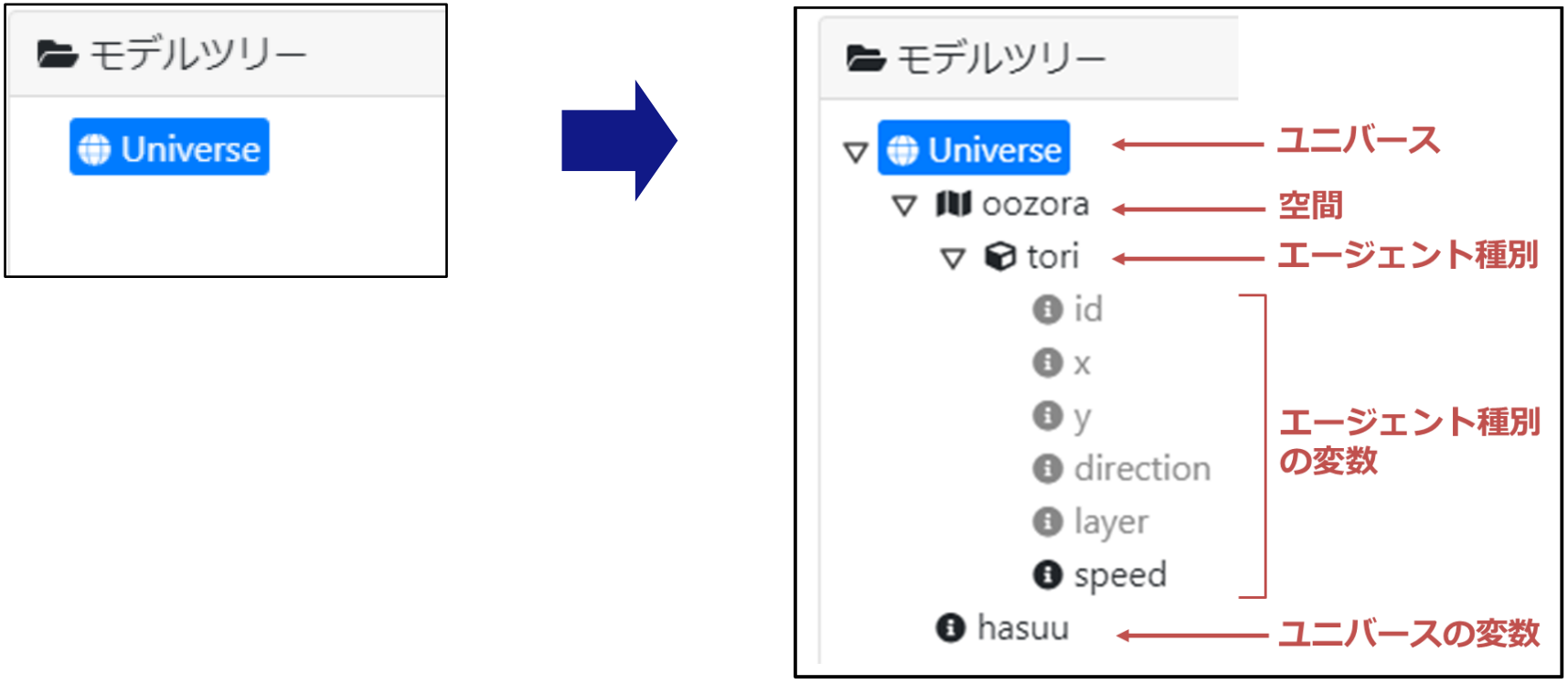
出力画面



ルール画面

❑空間／エージェントの種類・属性を作成

- 「モデルツリー」を編集することでモデルの枠組みを構築します。
  - 最初はモデル全体を表す「Universe」だけがあります。
  - ここに「空間」「エージェント種別」「変数」などを追加していきます。

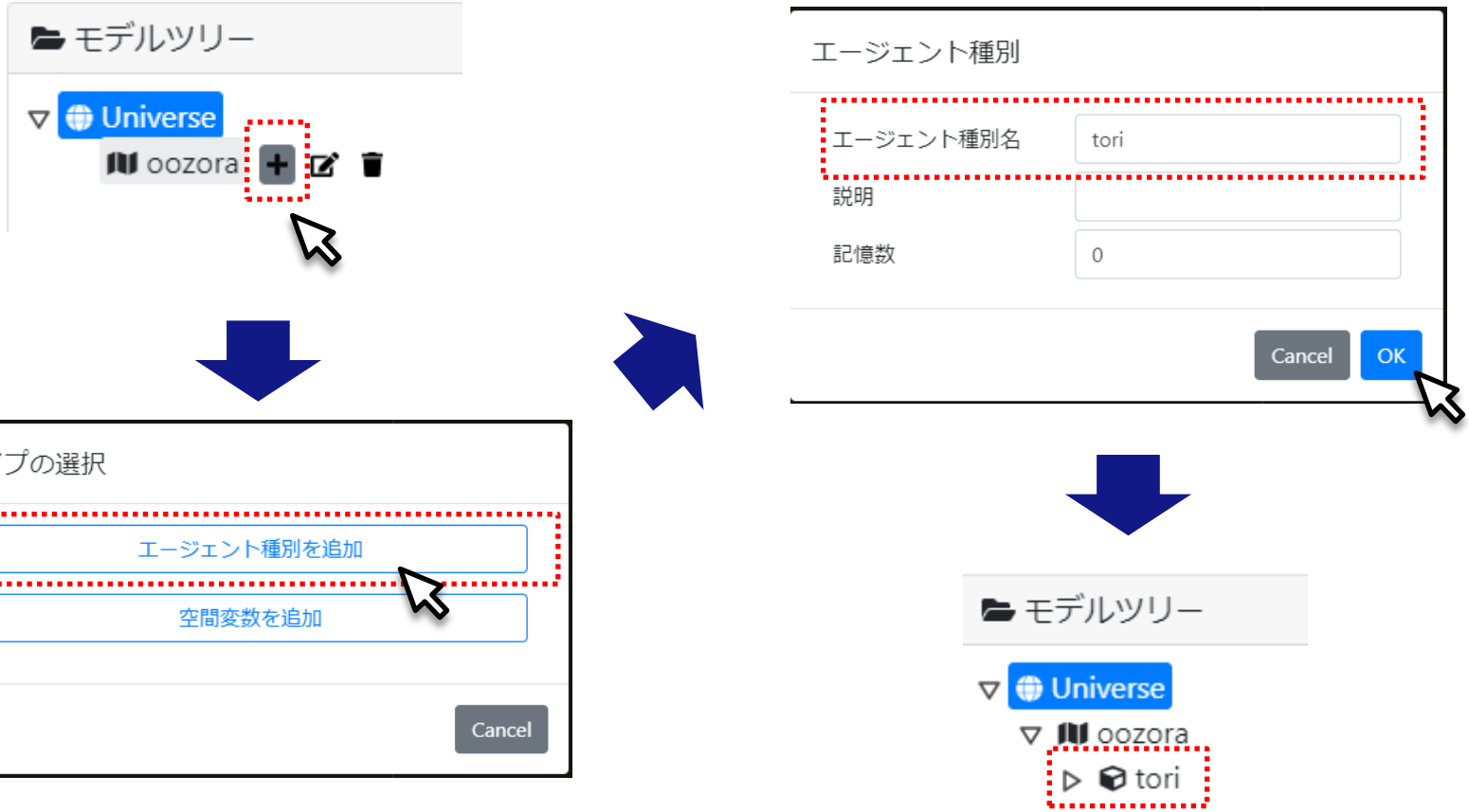


❒空間の定義

- Universe右に表示される「＋」をクリックして空間を作成します。
  - 空間名「oozora」・・・・大空
  - その他はデフォルト値
    - 説明は自由に入力

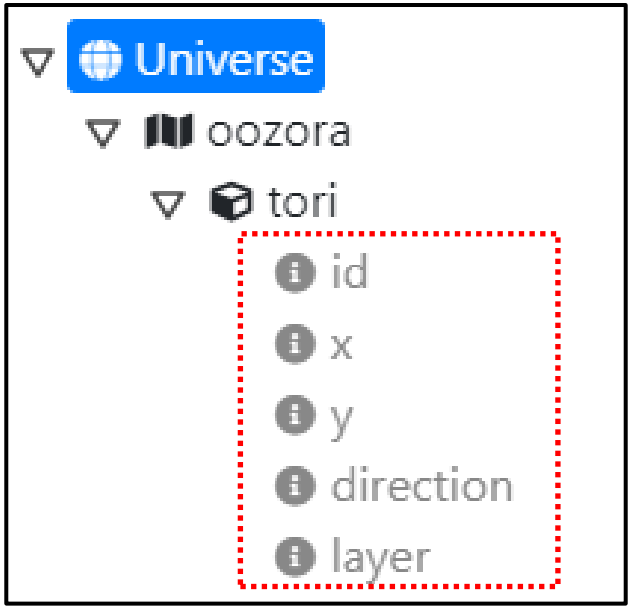
❒エージェント種別の作成

- 空間（oozora）右に表示される「＋」をクリックしてエージェント種別を作成します。
  - エージェント種別名「tori」・・・鳥エージェント
    - 「説明」は自由に、「記憶数」は0のままでOK。
      ※この定義はエージェント種別でエージェントそのものではありません。



❐エージェントの変数

- エージェントは以下の変数を持ちます。
  - id … 識別番号
  - x … x座標
  - y … y座標
  - direction … 向き
  - layer … 空間レイヤー（今回は使いません）
- 変数はエージェントの性質を表します。

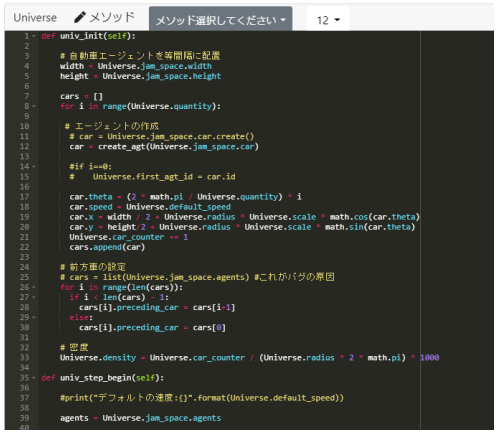


❑シミュレーション結果の出力形式を決定

① 空間／エージェントの
種類・属性を作成

② シミュレーション結果の
出力形式を決定

③ エージェントの
行動ルールを作成



❒TIPS：シミュレーションモデルの保存

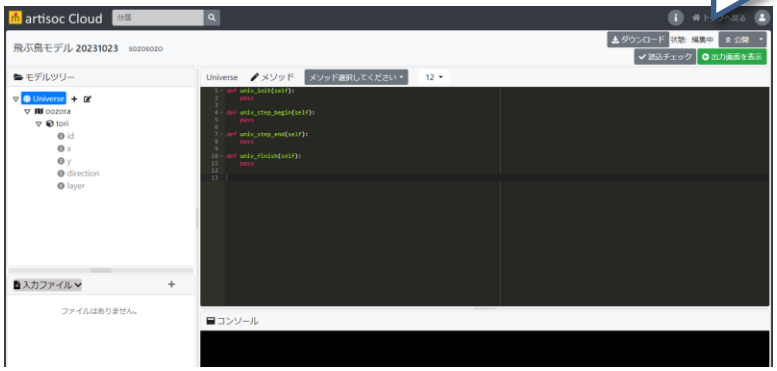
シミュレーションモデルに
不具合があって、artisoc　Cloudが
停止してしまった‥

ルールを変更したら
動かなくて、元に戻せなく
なった‥

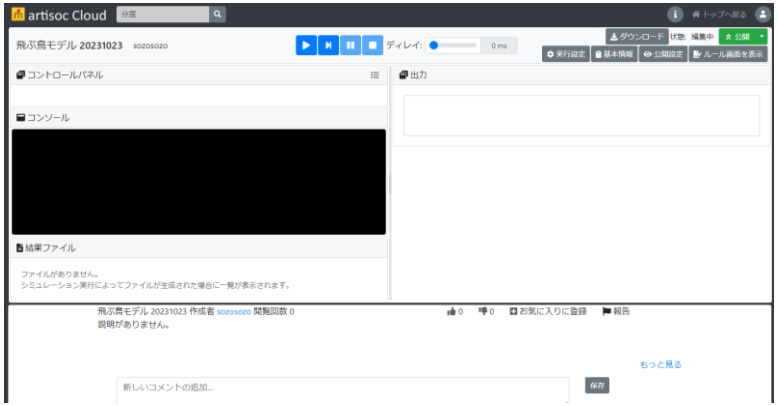
こまめに保存

□出力設定

■「出力画面を表示」をクリックして出力画面へ移動します。



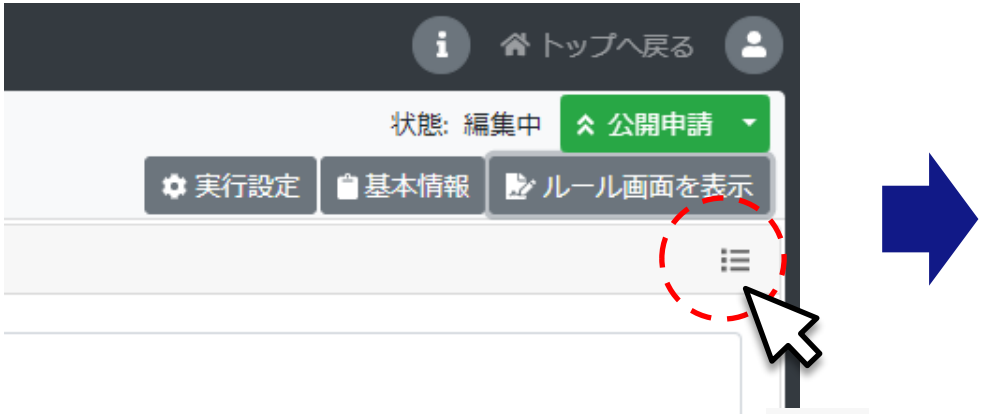
ルール画面



出力画面

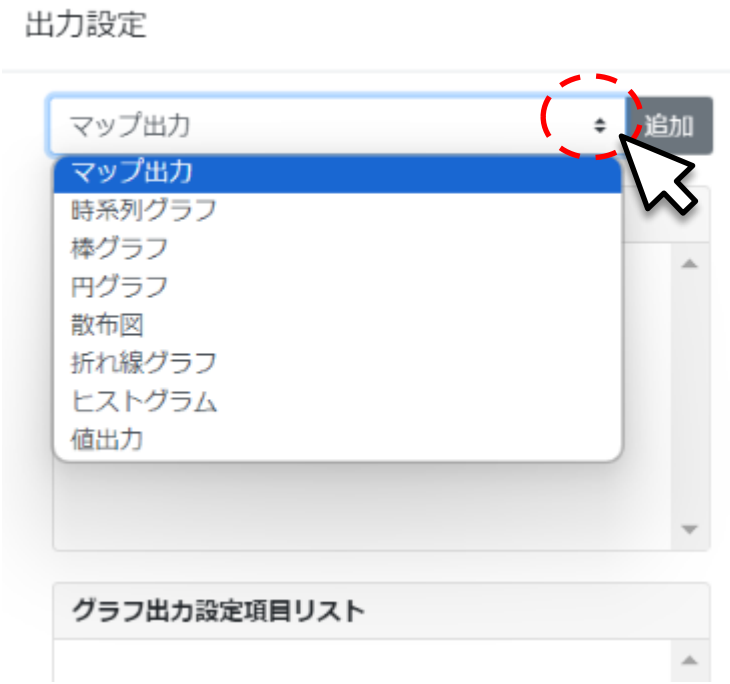
❑出力設定

- 空間とエージェントを「マップ」として出力します。
  - 下図のようにマップ出力を選択します。



カーソルをここに合わせると が表示される

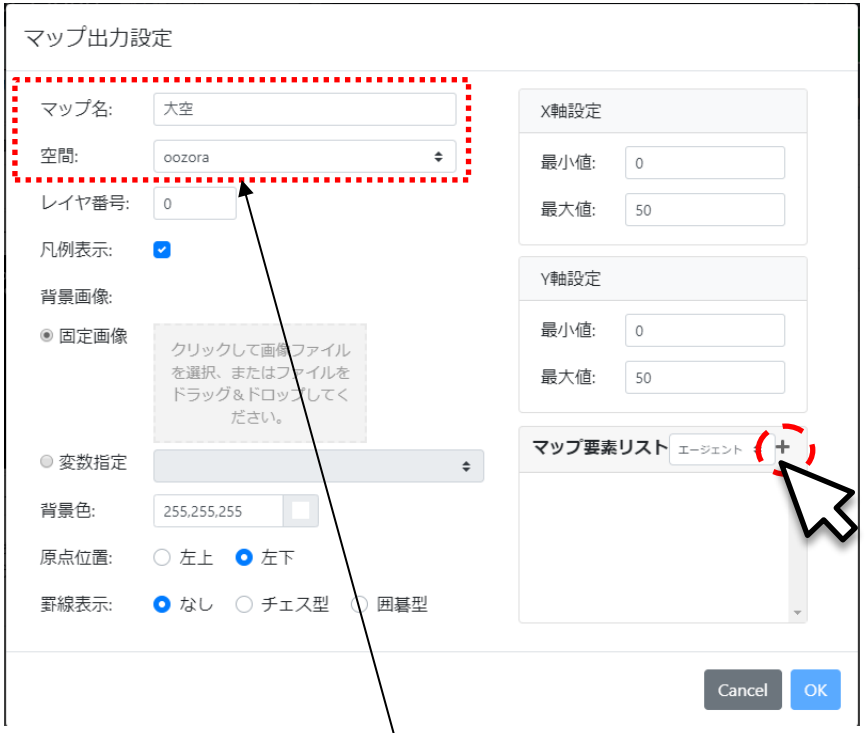
結果の出力の仕方を設定する画面を表示



結果を「マップ」とする出力項目を追加

□出力設定

■ 空間とエージェントを「マップ」として出力します。



マップ名：大空
空間：oozora
→空間oozoraを「大空」という名前で出力

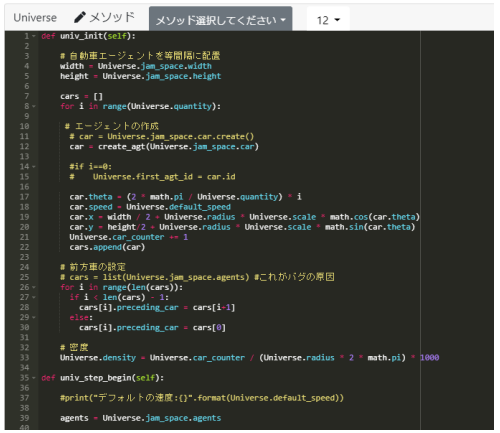
要素名：鳥
エージェント：tori
→エージェントtoriを「鳥」という名前で出力
※エージェントを表示する形状、色なども設定できるが、デフォルトでOK

❑エージェントの行動ルールを作成

① 空間／エージェントの
種類・属性を作成

② シミュレーション結果の
出力形式を決定

③ エージェントの
行動ルールを作成

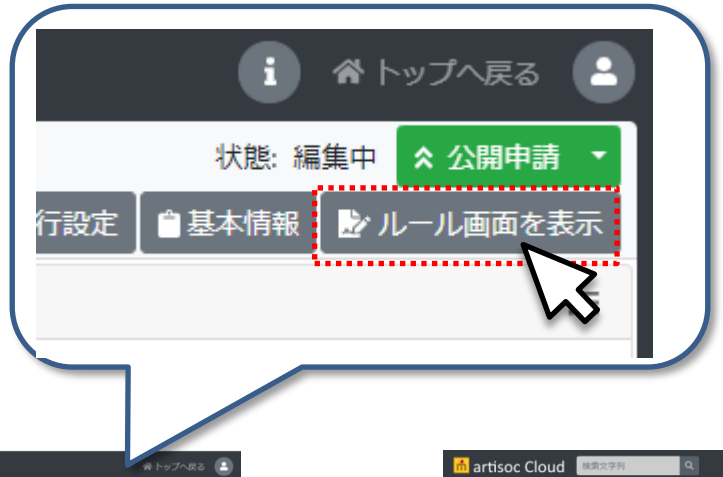


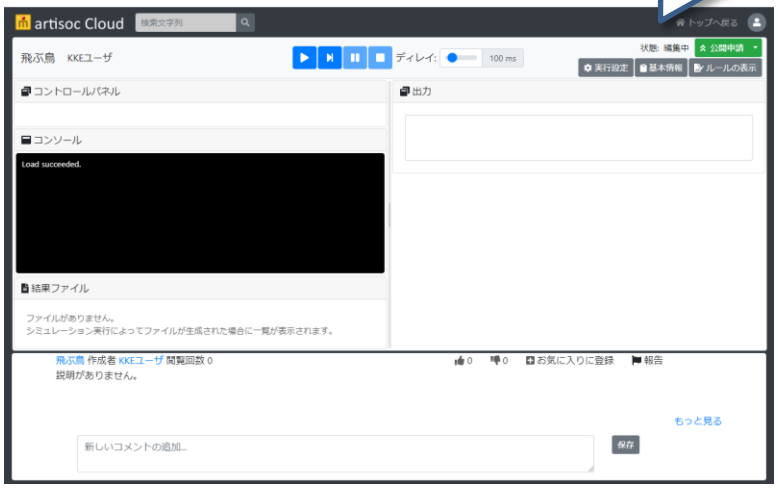
❑ルールを作成

- 以下のようなモデルのルールを作成します。
  - シミュレーション開始時に鳥エージェントを1体だけ作成
  - 鳥エージェントの初期位置は空間の中央
  - 鳥エージェントは毎ステップ、前方に1進む

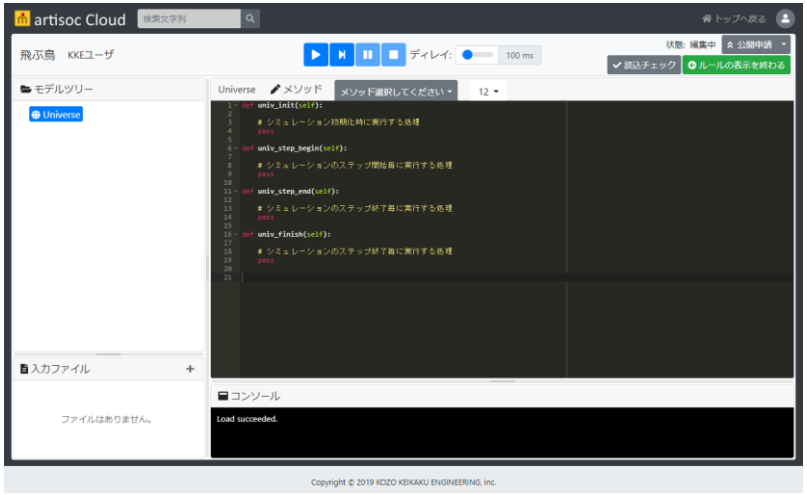
❑ルールを作成

■「ルール画面を表示」をクリックしてモデルのルール画面に遷移します。





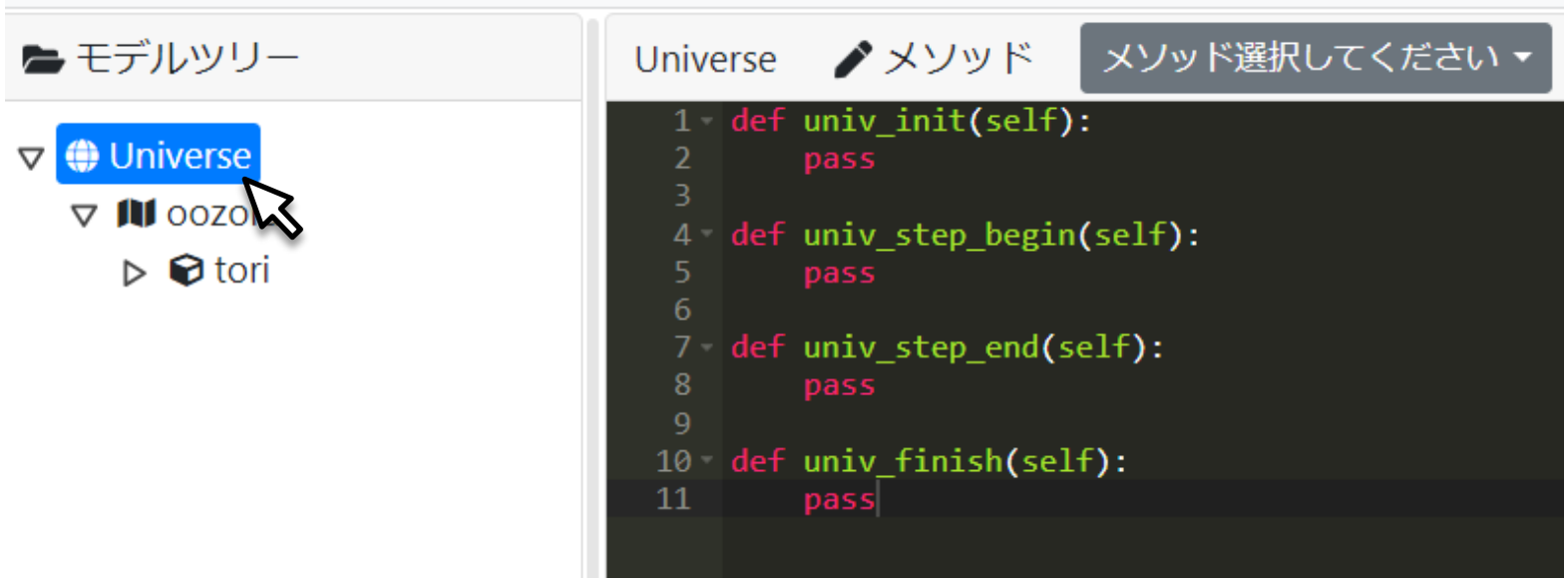
出力画面



ルール画面

❒エージェントの生成

- シミュレーション開始時に鳥エージェントを1体生成するルールを作成します。
  - モデル全体に関わるルール→Universeのルールエディタに記述します。
  - モデルツリーのUniverseをクリックするとUniverseのルールエディタが開きます。
  - Universeのルールは以下の構成です。
    - univ_init … シミュレーション開始時に一度だけ実行するルール
    - univ_step_begin … シミュレーションの各ステップ開始時に実行するルール
    - univ_step_end … シミュレーションの各ステップ終了時に実行するルール
    - univ_finish … シミュレーション終了時に一度だけ実行するルール



❑エージェントの生成

- シミュレーション開始時に鳥エージェントを1体生成します。
  - univ_initに鳥エージェントを生成するルールを記述します。
  - エージェントの生成→**create_agt**：エージェントを生成する**関数**
    - 関数：ルールをひとまとめにしたもの
    - 「create_agt(Universe.oozora.tori)」でtoriエージェントを生成します。
  - 「pass」を消し、下図のように記述します。
    - インデントに注意→次ページ

❒TIPS：ルール作成時の注意：インデント

- ルールは、tabキーを使って1段下げた部分に記述することに注意しましょう。
  - これをインデントといいます。
  - インデントをするにはtabキーを使うと便利です。
    - 「shift + tab」でインデントが戻ります。
  - エンターキーで改行してもインデントは維持されます。

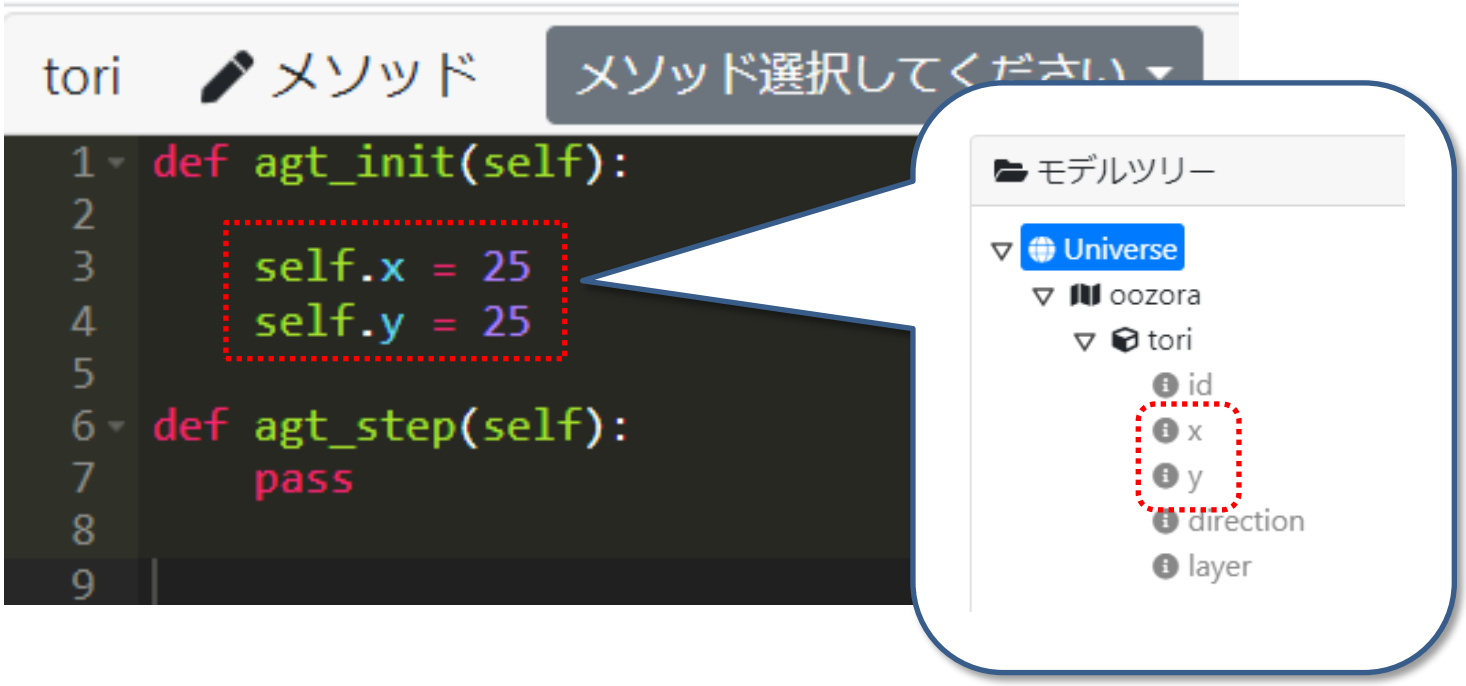
※空白行は自由に挿入してかまいません。

❒エージェントのルールを作成

- エージェントの行動ルールを作成します。
  - エージェントの行動ルール→エージェントのルールエディタに記述します。
  - ツリー上のエージェント種別名をクリックするとエージェントのルールエディタが開きます。
  - エージェントのルールは以下の構成です。
    - agt_init　… エージェントが生成されたとき一度だけ実行されるルール
    - agt_step … エージェントが毎ステップ実行するルール

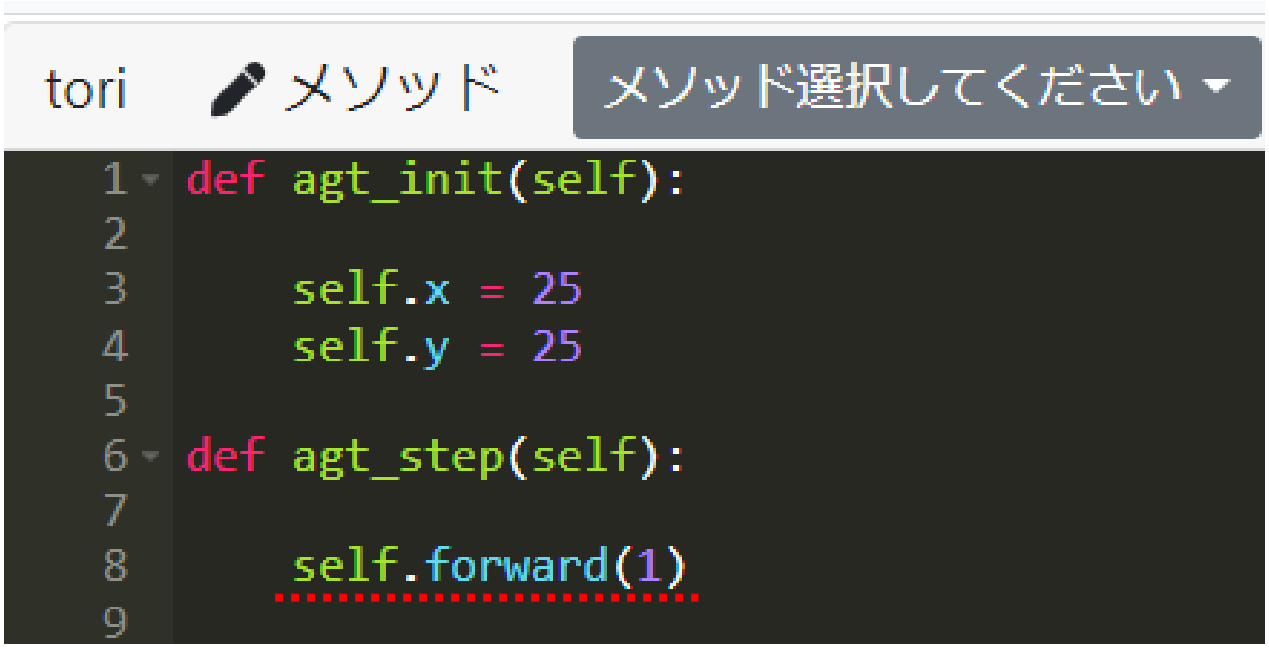
❑エージェントのルールを作成

- 鳥エージェントを初期位置として空間の中央に配置します。
  - agt_initにルールを書きます。
  - 自分自身のx座標とy座標に25を代入します。
    - 自分自身の変数を呼び出すときには「self.[変数名]」と記述します。
    - 「変数名 = 数値」は、「変数に数値を代入する」という意味です。
  - 「pass」を消し、下図のように記述してください
    - pass（パス）は何もしないという意味



❑エージェントのルールを作成

- 鳥エージェントは毎ステップ前に進みます。
  - agt_stepにルールを書きます。
  - **forward**：前に進む関数
    - 「self.forward([数値])」と書くことで数値だけ前に進みます。
      - ☑「self」は自分自身なので、自分自身に「進め」と命令するイメージです。



ルールの構成

- Universeのルールエディタ→モデル全体のルール
  - univ_init ・・・ シミュレーション開始時に一度だけ実行
  - univ_step_begin ・・・ 各ステップの最初に実行
  - univ_step_end ・・・ 各ステップの最後に実行
  - univ_finish ・・・ シミュレーション終了時に一度だけ実行

toriエージェントの生成

- エージェントのルールエディタ→エージェントの行動ルール
  - agt_init ・・・ エージェント生成時に一度だけ実行
  - agt_step ・・・ 各ステップで実行

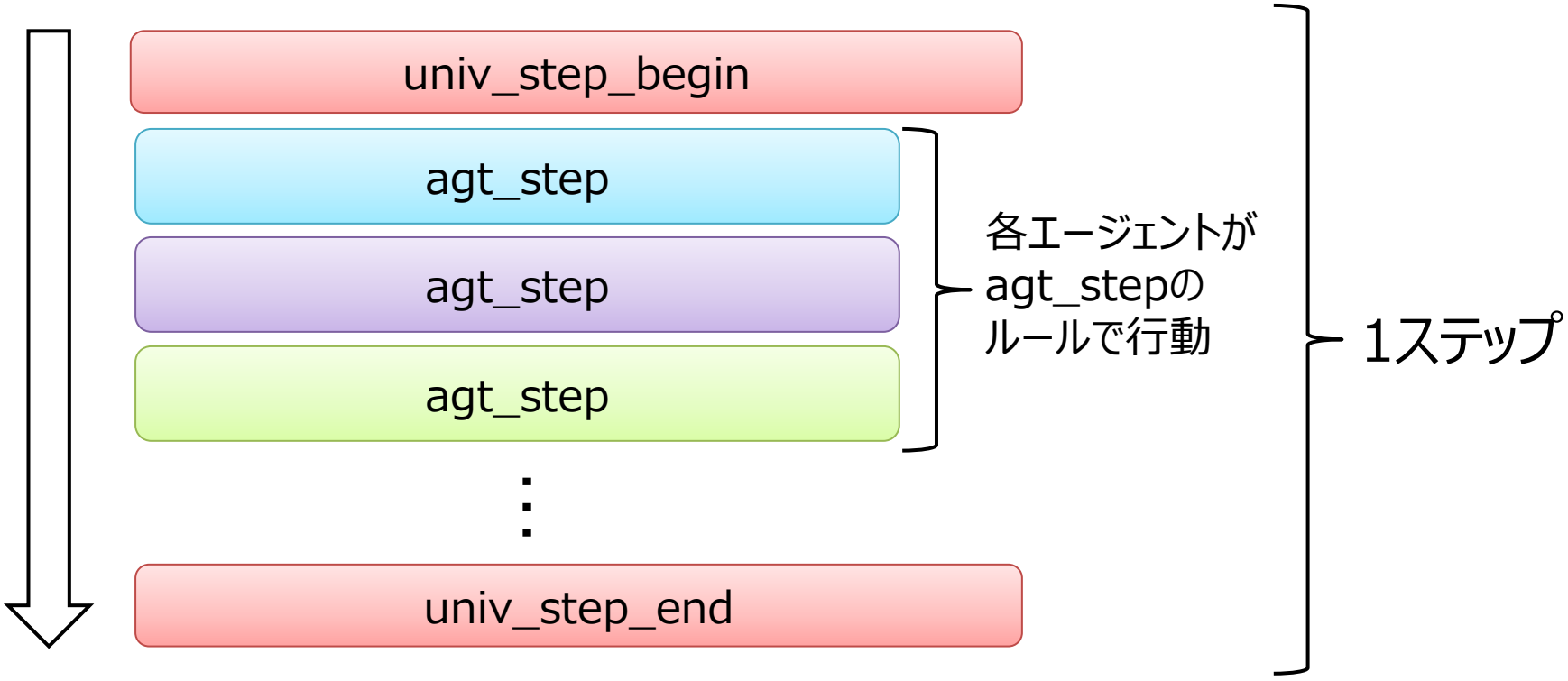
toriエージェントが
X=25,y=25に配置

toriエージェントが「1」前進

□ステップについて

- ステップ
  - artisoc Cloudの時刻単位
  - 1ステップの間に、全てのエージェントがルールにしたがって行動する。

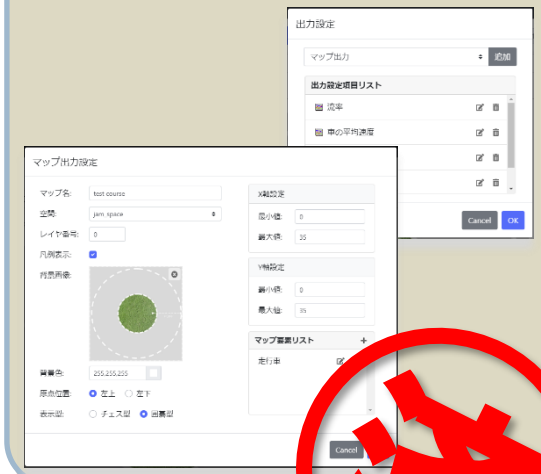
- 1ステップの流れ



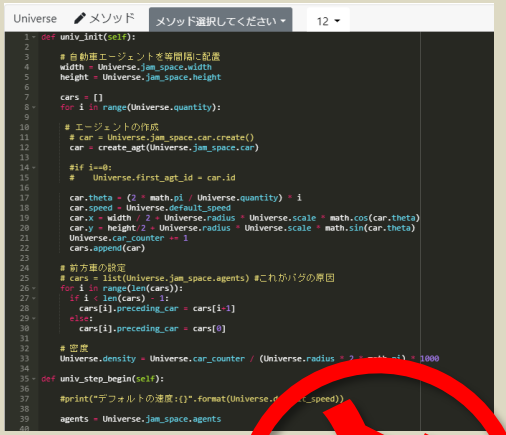
□モデル構築完了

① 空間／エージェントの
種類・属性を作成

② シミュレーション結果の
出力形式を決定



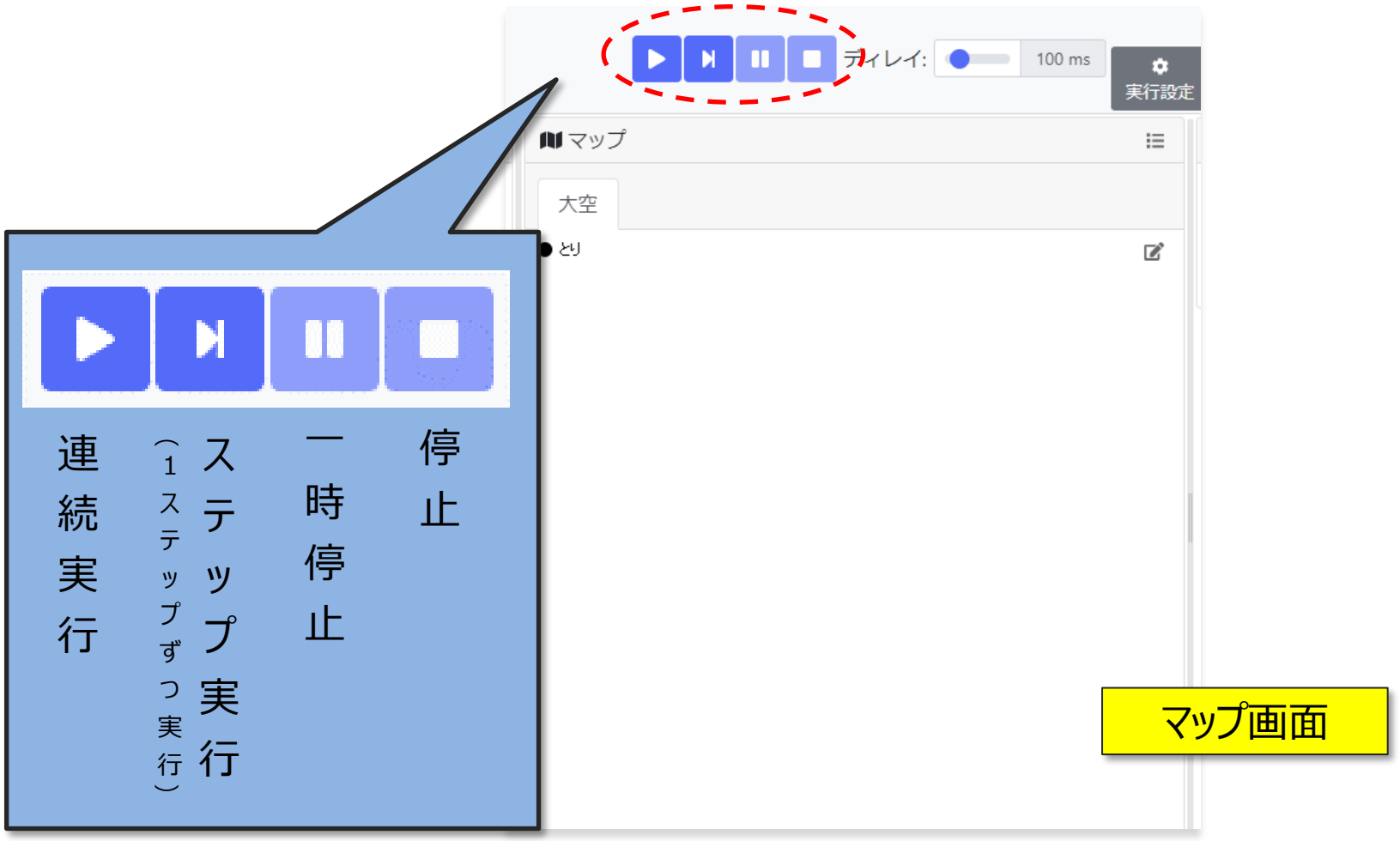
③ エージェントの
行動ルールを作成



❑ここまでのモデル

- 途中から来た人や途中でついていけなくなった人は、下記のモデルにアクセスしモデルを継承してください。
  - https://artisoc-cloud.kke.co.jp/models/GyXulUvDRbG-mZA4hkwF0g
    - チャットでURLを送ります
  - 継承：他のユーザが作ったモデルをコピーして編集

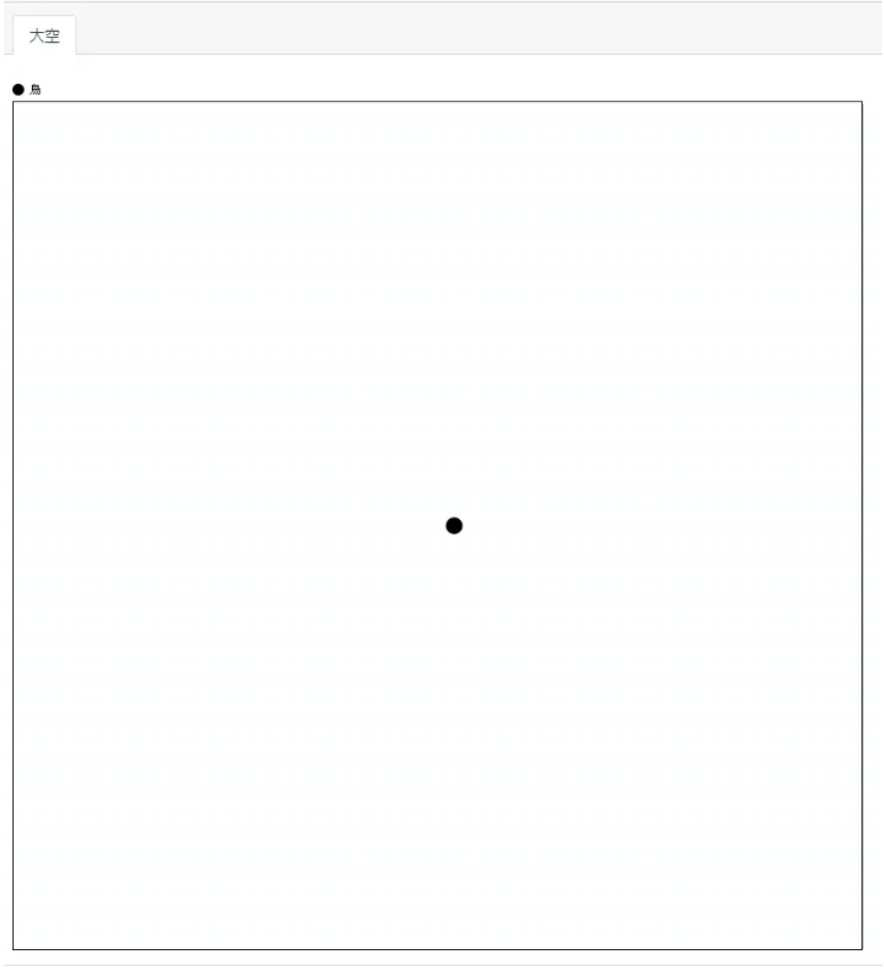
❑シミュレーションを実行



ルールの編集、出力の設定を行うときは「停止」状態である必要があります。

❑モデルの動作を確認する

- マップ上を鳥が左から右に動いていれば成功です。
  - 10000ステップで自動終了するようになっています。



❒TIPS：シミュレーション速度を調整する

■ シミュレーション速度を変更したい・・・

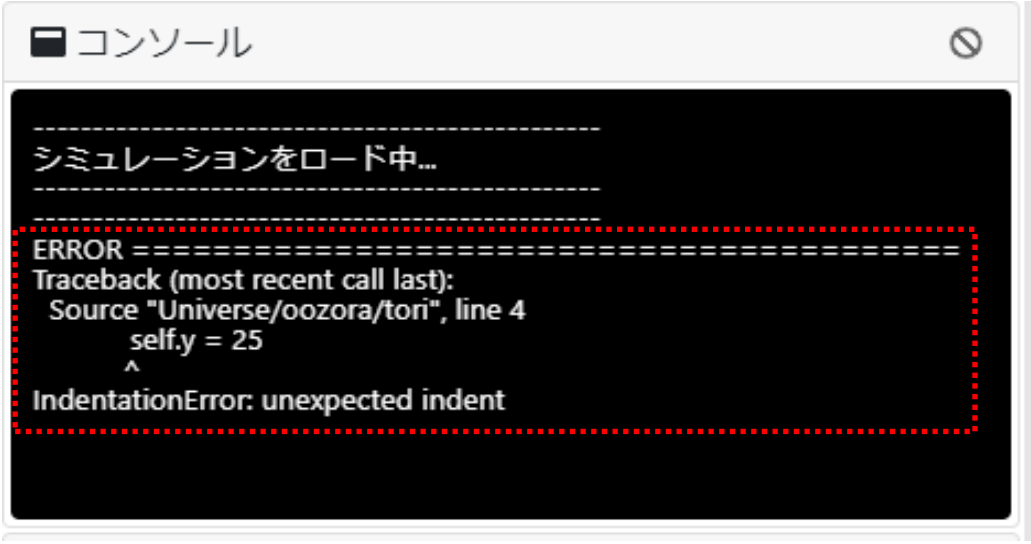
スライドバーで実行速度を調整する

右に動かすと実行速度が遅くなり、左に動かすと速くなる

❒TIPS：シミュレーションが実行できないとき

- うまく動かなければ、「読込チェック」をクリックしてください。
  - コンソール画面にエラーメッセージが表示されます。

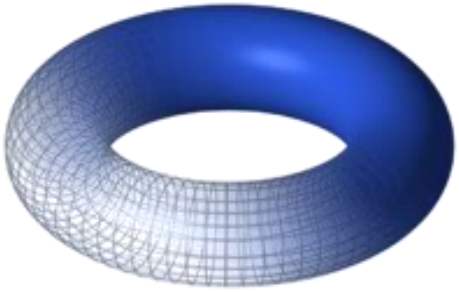
- エラーメッセージの意味が分からないときは、質問してください。
  - 遠隔参加の方は、エラーメッセージをQ&Aにコピーしてお知らせください。



❐TIPS：空間のループを変更する

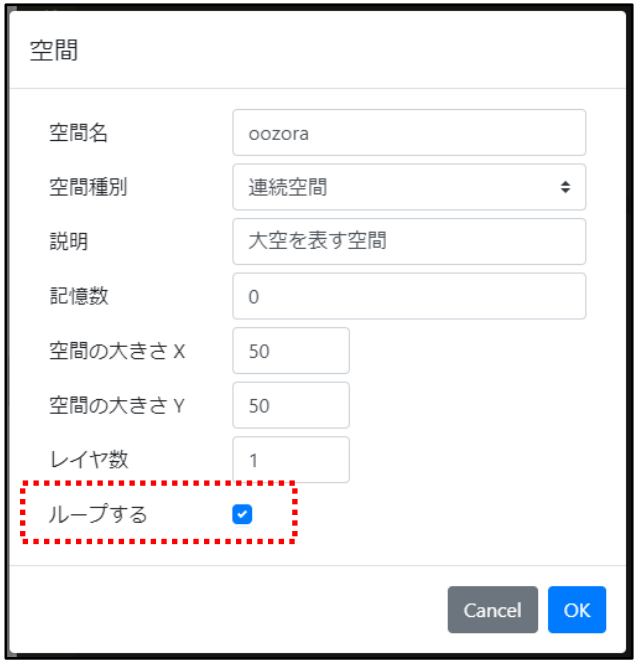
■ 空間がループ ＝上端-下端・左端-右端が繋がった空間

ループした空間のイメージ



「ルール画面を表示」をクリック

「Edit」をクリック



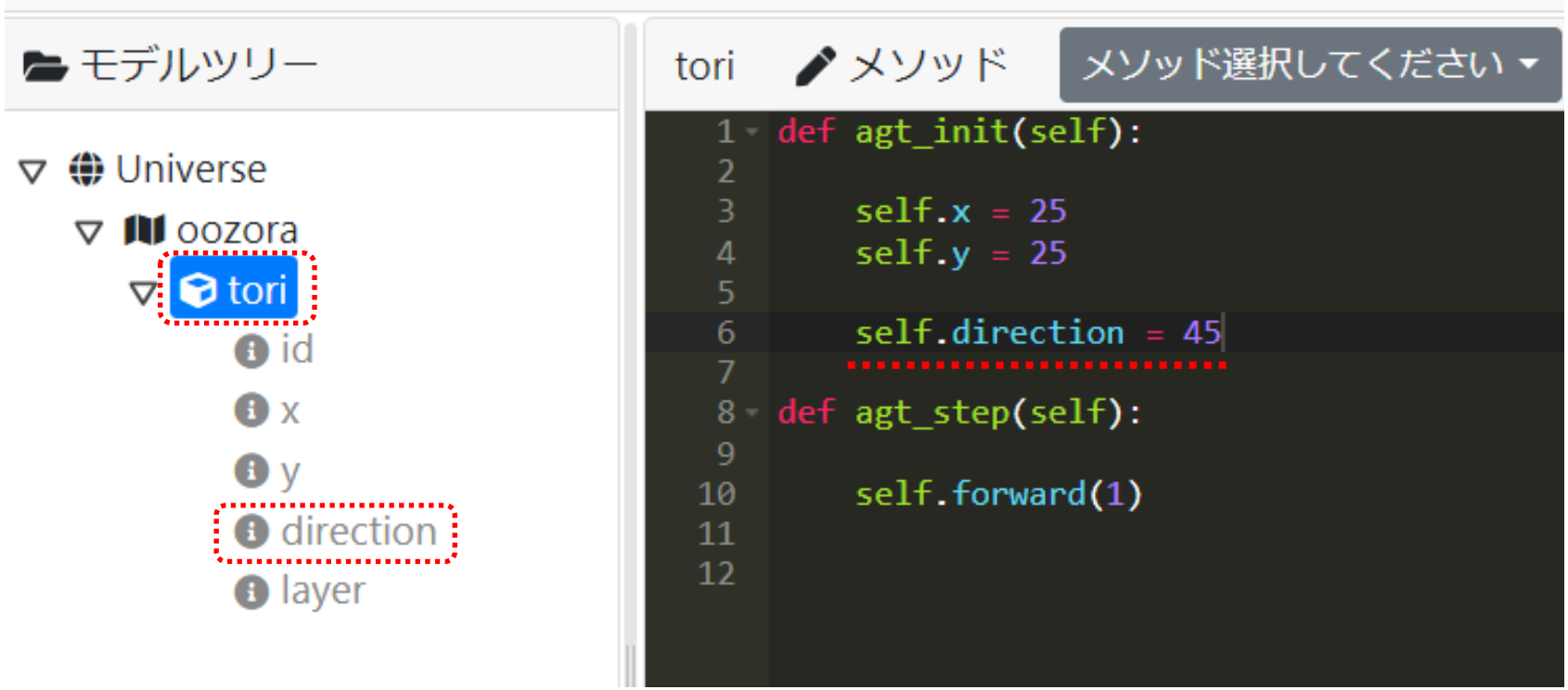
2. 基本的なテクニックを学ぶ

❑基本的なテクニックを覚える

- 途中から来た人は、下記のモデルにアクセスしモデルを継承してください。
  - https://artisoc-cloud.kke.co.jp/models/GyXulUvDRbG-mZA4hkwF0g
    - チャットでURLを送ります
  - 継承：他のユーザが作ったモデルをコピーして編集

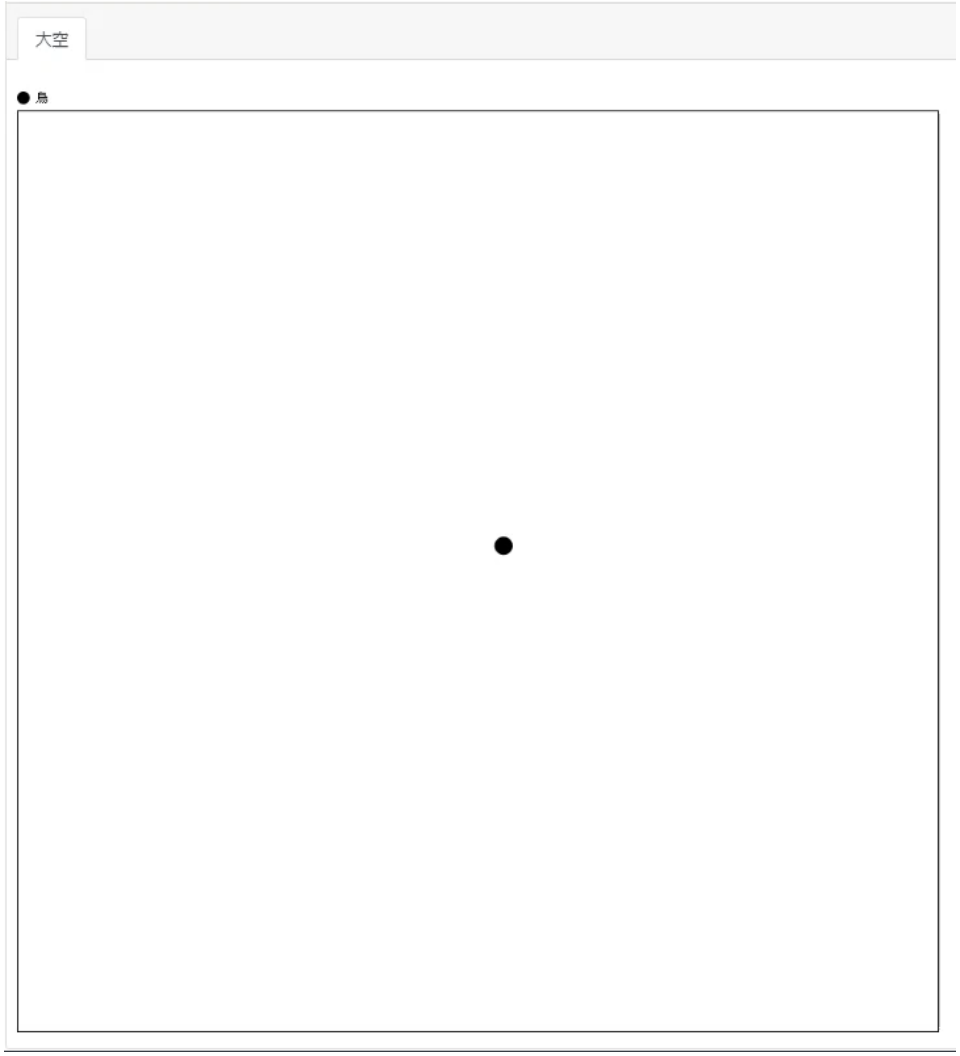
❒変数設定の変更（進む方向を変える）

- ルール画面に戻り、ツリー上のtoriをクリックしてルールエディタを開きます。
- agt_initで変数「direction」に45を代入して実行します。
  - self.direction = 45

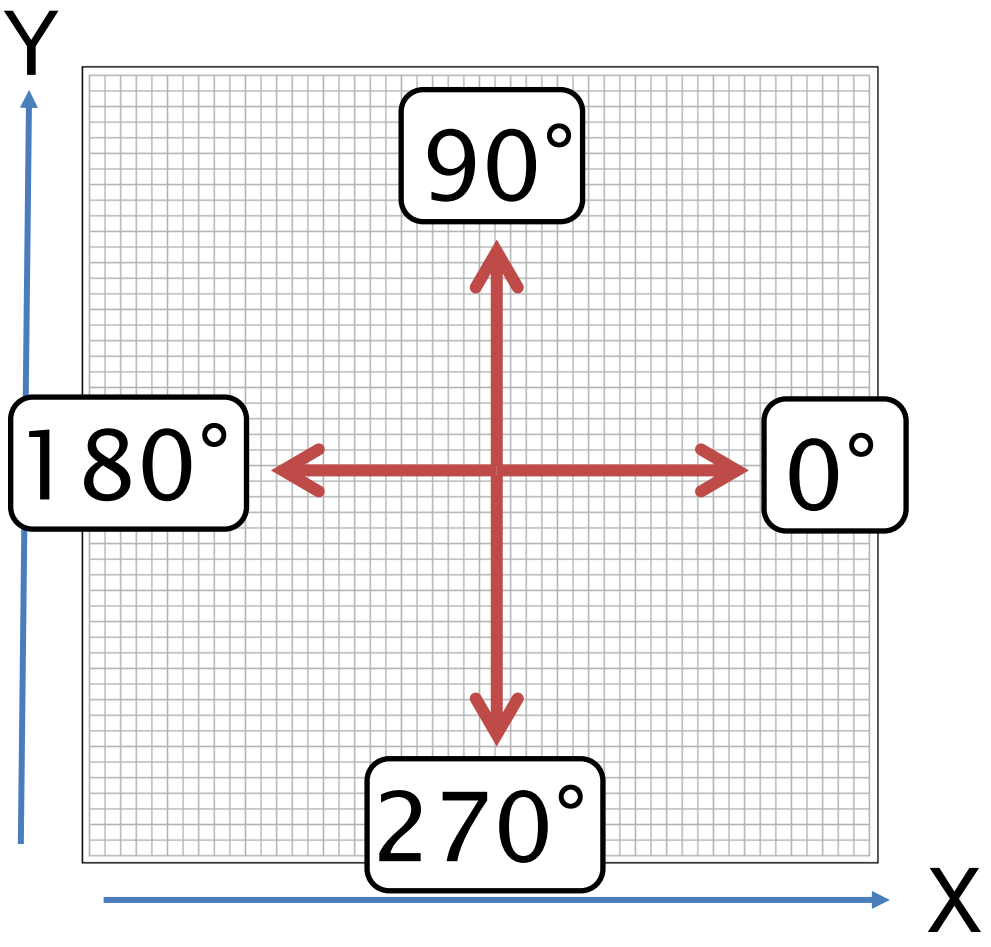


□モデルの動作を確認する

- 鳥が右上に飛んでいきます。



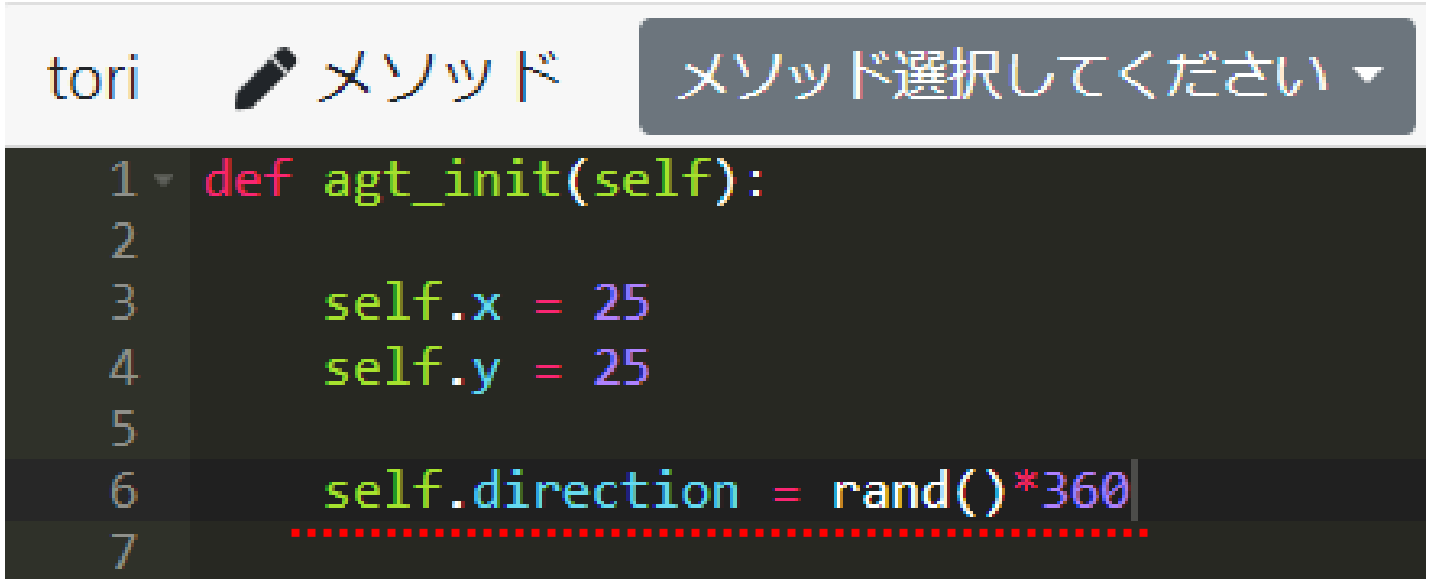
❒artisoc Cloudにおける方向



※左下原点の場合

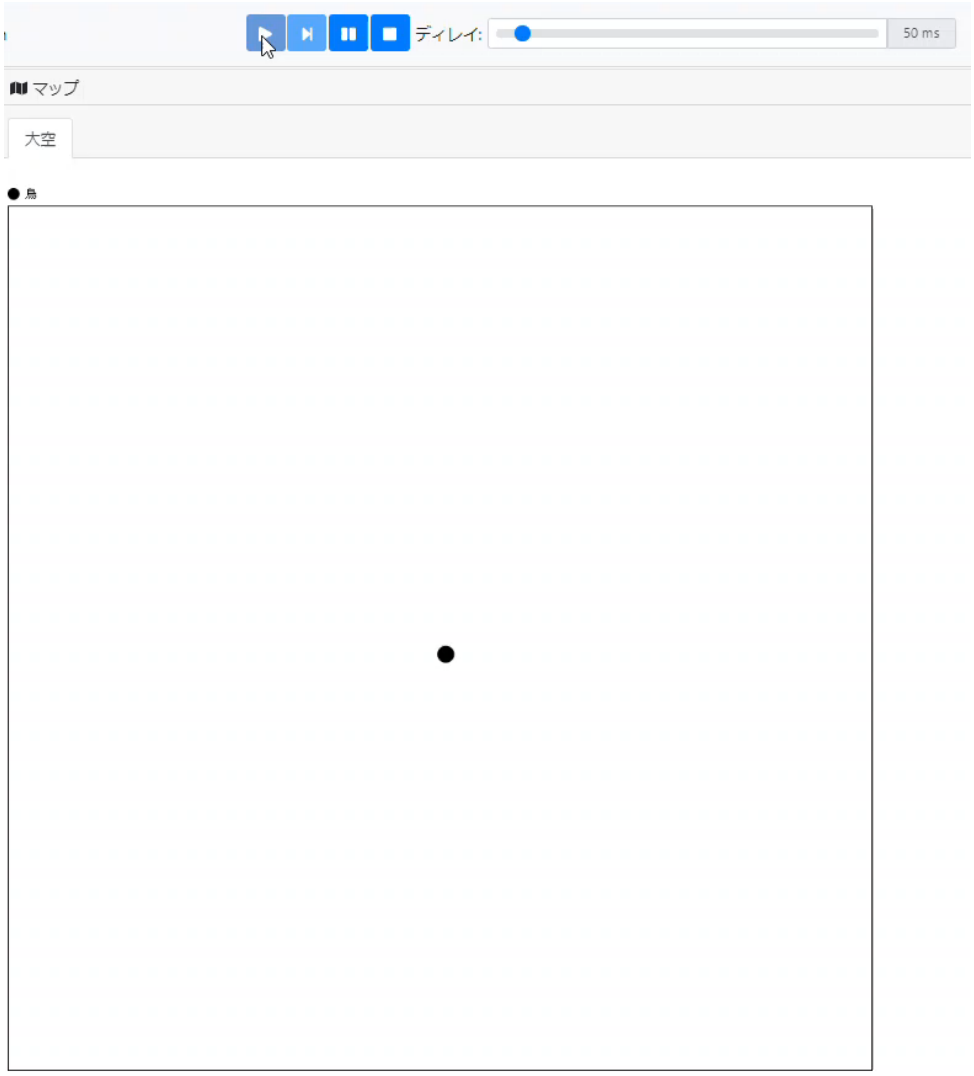
❒関数の使用（進む方向をランダムに変える）

- 属性「direction」に進む方向の値をランダムに設定（0~360度）して実行します。
  - **rand** ：0~1のランダムな値（一様乱数）を取得する関数
  - 「rand() * 360」で0~360のランダムな値を意味します。
    - 「*」は掛け算の記号です。
    - 関数には必ず()がつきます。
    - 「=」は右の値を左の変数の値に代入するという操作を示す。←要注意



□モデルの動作を確認する

■ 実行と停止を繰り返すと、実行するたびに鳥の飛ぶ向きが変わります。



❒関数とは

- 関数：処理をひとまとめにしたもの
  - 引数：関数へ渡す値
  - 処理：引数をもとに関数が行う動作
  - 返り値（戻り値）：処理をもとに関数が返す値

  ※引数は複数あったり、なかったりします

  ※返り値がない（使わない）関数もあります

- 関数の例：forward
  - 引数：進む距離
  - 処理：引数の値だけ前に進む
  - 返り値：正常終了時は-1（今は使っていません）

```
self.forward(10)　# 引数の数10だけ前に進む
```

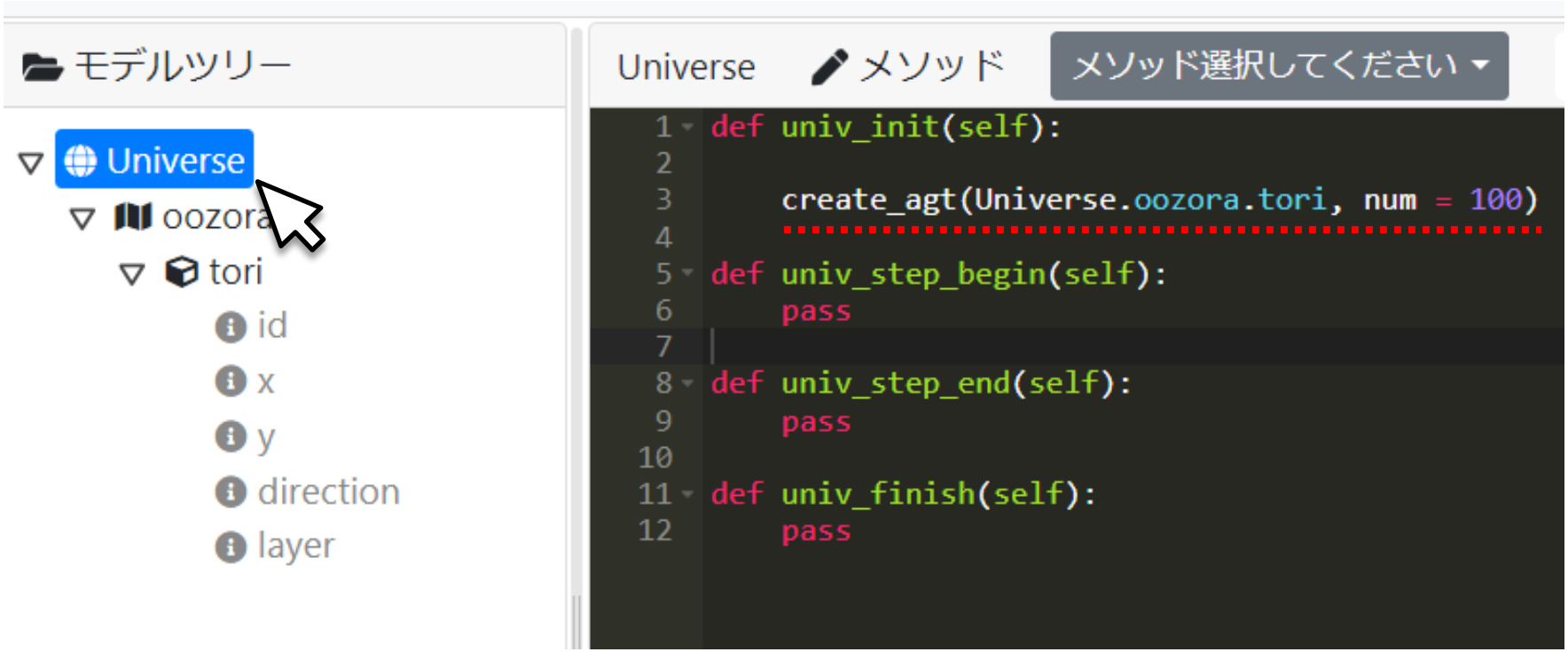
- 関数の例：rand
  - 引数：なし
  - 処理：0~1の一様乱数を発生させる
  - 返り値：0~1の一様乱数

```
r = rand()　# rand関数の返り値（0~1の一様乱数）を変数rに代入する
```

※引数がないときも()が必要

❑エージェントの生成の仕方の変更

- エージェントを一度に複数生成する。
  - 「create_agt([エージェント種別], num=[エージェント数])」で指定した数のエージェントを生成します。
  - 100個生成します。

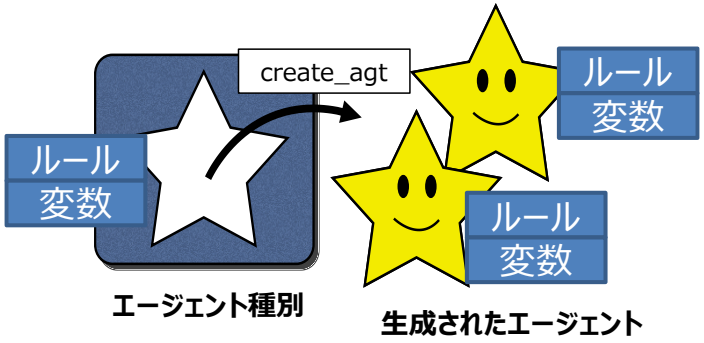


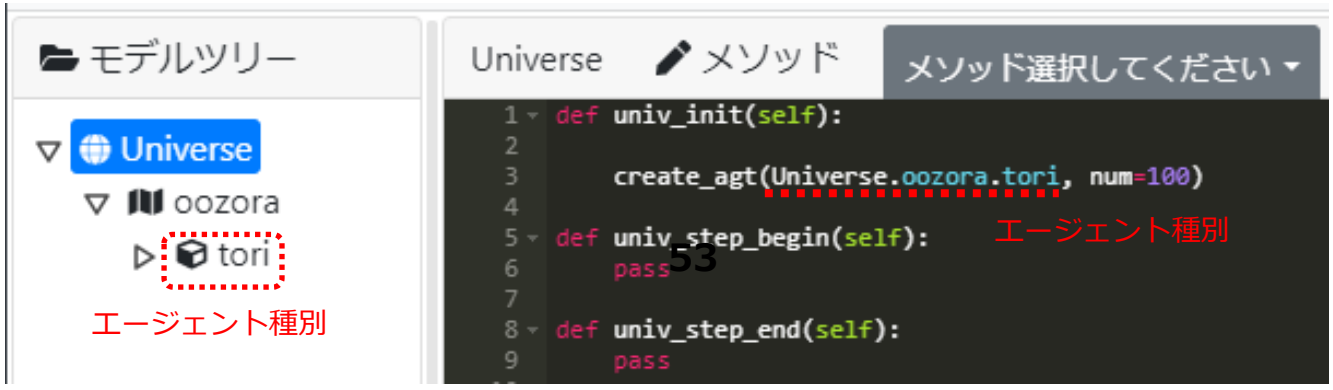
❑モデルの動作を確認する

- 100羽の鳥が円形に飛んでいきます。

❑エージェント種別とエージェント

- エージェント種別：エージェントの「ひながた」のようなものです。
  - ツリー上に存在するのはエージェントでなくエージェント種別です。
- create_agtはエージェント種別を指定して、エージェントを生成します。
- 生成されたエージェントは共通のルールと変数を持ちますが、変数の値はそれぞれのエージェントで異なります。
  - 共通のルール：
    「directionをランダムに設定する」→direction変数の値は個々のエージェントで異なる。

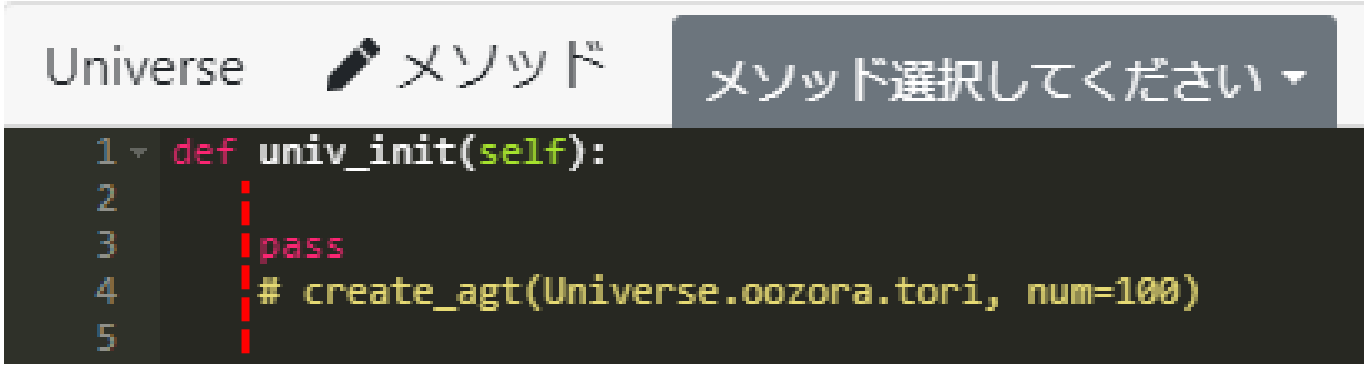




❐TIPS：コメントアウトとpass

- コメントアウトは、コードを無効化する機能です。
  - 行の先頭に「＃」
    - 行にカーソルを合わせたり選択した状態で「ctrl ＋ /」でもコメントアウトできます。
  - コードを一時的に変更したり、説明を加えるのに使います。

- passは、「何もしない」を意味するルールです。
  - univ_initなどのルールエディタの要素内では、インデントの中に何かを記述する必要がある。
    - でないと、「インデントがない」というエラーになる。
  - 何もしないときは、「何もしない」ということをartisoc Cloudに教えるためにpassを記述します。

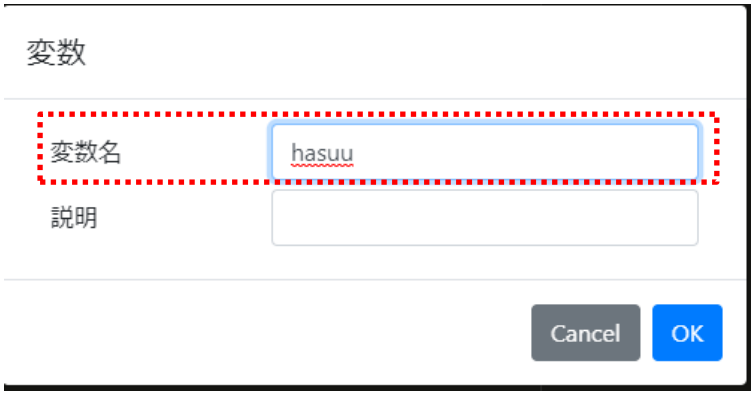


□コントロールパネルの設定

- 鳥の数をスライドバーの操作で設定できるようにする。

□コントロールパネルの設定

- 鳥の数を表す変数としてUniverseの下に変数「hasuu」を作成する。



□コントロールパネルの設定

- Universe変数「hasuu」をコントロールパネルの操作対象に設定する。
  - 出力画面に移動し、実行画面左上の「コントロールパネル」で設定する。
  - 「種類」で「スライドバー」を選択する。

□コントロールパネルの設定

- 鳥の数をスライドバーの操作で設定できるようにする。
  - Universe変数「hasuu」をコントロールパネルの操作対象に設定する。
    - ➡ 下図のように項目を設定します。

コントロールパネルで鳥の数を設定できます。

□コントロールパネルの設定

- 変数「hasuu」の数だけ、鳥を生成する。
  - for i in range(Universe.hasuu):
    one = create_agt(Universe.oozora.tori)

- 生成した順番に応じて、進む方向を決める。
  - （上記の続き）
    one.direction = i

```
1  def univ_init(self):
2
3      for i in range(Universe.hasuu):
4          one = create_agt(Universe.oozora.tori)
5          one.direction = i
6
```

❑ここまでのモデル

- 途中から来た人や途中でついていけなくなった人は、下記のモデルにアクセスしモデルを継承してください。
  - https://artisoc-cloud.kke.co.jp/models/oOf0rCNOTU-38dfhIYHfzg
    - チャットでURLを送ります
  - 継承：他のユーザが作ったモデルをコピーして編集する

3. 相互作用を含むモデルを作成する

❑相互作用を含むモデルを作成する

- 鳥が群れながら飛べるのはなぜ？
  - ×リーダーの鳥が指示している。
  - ○個別の鳥が周囲の鳥に飛び方を合わせている。

  →ボイドモデル（Craig Raynolds, 1987）

- ごく単純なボイドモデルを作り、群れながら飛ぶ鳥を表現しましょう。

❒準備

- Universeのルールを以下のように修正します。
  - 向きを指定しない（コメントアウトする）。
    - 向きはagt_initでランダムに指定される。

（復習）コメントアウトの方法
- 行の先頭に「＃」
- 行にカーソルを選択した状態で「ctrl + /」

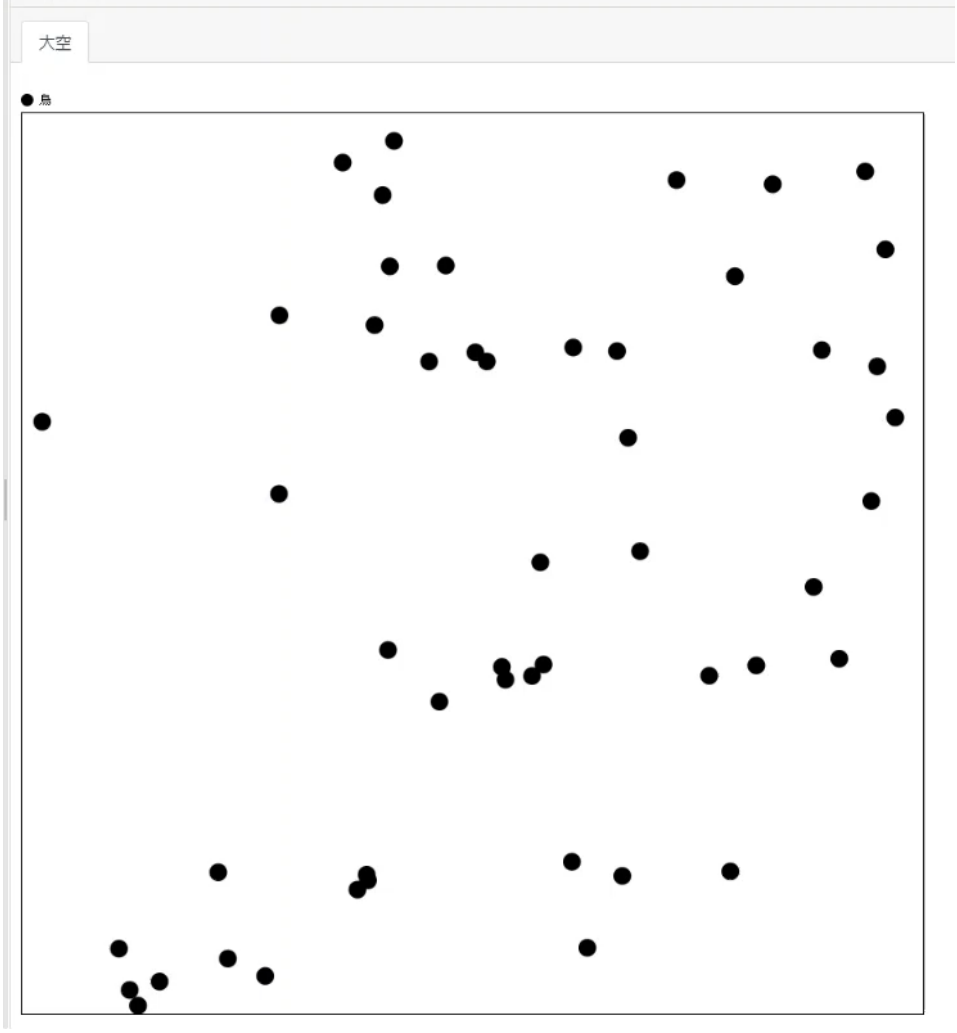
```
def univ_init(self):

    for i in range(Universe.hasuu):
        one = create_agt(Universe.oozora.tori)
        # one.direction = i

```

- 鳥のルールを以下のように修正します。
  - 初期位置をランダムに。
    - 「rand() * 50」で0~50のランダムな値

```
def agt_init(self):

    self.x = rand() * 50    # ランダムな初期位置
    self.y = rand() * 50    # ランダムな初期位置

    self.direction = rand()*360

def agt_step(self):

    self.forward(1)

```

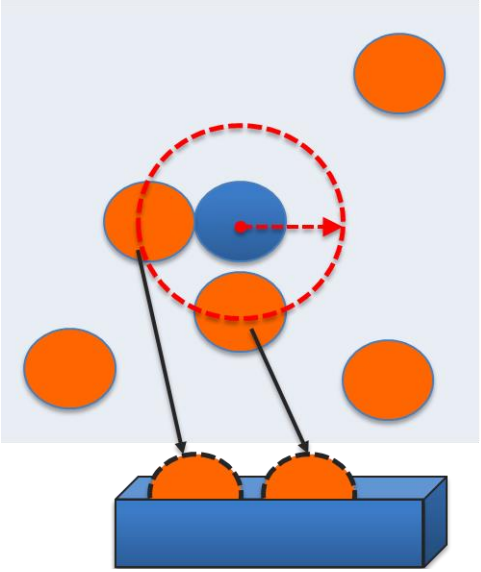
❑モデルの動きを確認する

- 鳥がランダムに飛んでいきます（鳥の数＝50）。



❑エージェント集合型の変数の活用（周囲の鳥に方向を合わせる）

- 鳥エージェントのルールに、周囲の鳥を認識し飛び方を合わせるルールを追記します。
  - 周囲のエージェントを探し、エージェント集合型の変数に格納する。
    - ➡ エージェント集合型変数：エージェントの集合を値として持つ変数
  - エージェント集合型の変数に入っているエージェント数を数える。
    - ➡ 1以上の場合、周囲にエージェントがいる。
  - 周囲にエージェントがいる場合、そのうち1つのエージェントを選び、自分の飛ぶ向きをそのエージェントに合わせる。



エージェント集合型変数
（複数のエージェントを格納するための変数）

❑エージェント集合型の変数の活用（周囲の鳥に方向を合わせる）

- 周囲のエージェントを取得し、エージェント集合を変数に格納します。
  - agt_stepに記述します。
  - **make_agtset_around_own**：視野の範囲内のエージェント集合を取得する。
    - 第1引数：視野
    - 第2引数：自分を含むかどうか（TrueかFalse）

```
8  def agt_step(self):
9
10     # 自分から距離2以内のエージェントをtori_setに格納（自分自身を含まない）
11     tori_set = self.make_agtset_around_own(2, False)
12
13     self.forward(1)
14
```

❑条件分岐文の活用（周囲の鳥に方向を合わせる）

- もし周囲に鳥がいれば、そのうち1羽を選び、自分の方向をその鳥に合わせます。
  - **「条件分岐文」はif文**で記述します。
    - 条件分岐分：もし ～ ならば … という処理

```
if 条件分岐文 :
    処理
```

※処理はインデント内に記述

条件分岐文の例

1 < 5 [1は5未満である] ⇒ 真(True)
4 >= 8 [4は8以上である] ⇒ 偽(False)
3 == 4 [3と4は等しい] ⇒ 偽(False)

- **count_agtset**: エージェント集合の要素数を数える関数
- **randchoice**: エージェント集合からランダムにエージェントを1つ取得する関数

```
8  def agt_step(self):
9
10     # 自分から距離2以内のエージェントをtori_setに格納（自分自身を含まない）
11     tori_set = self.make_agtset_around_own(2, False)
12
13     if count_agtset(tori_set) > 0: # もし周囲に鳥がいれば
14         one = randchoice(tori_set) # ランダムに一を選ぶ
15         self.direction = one.direction # 自分の向きをその鳥に合わせる
16
17     self.forward(1)
18
```

※ここで、tori_setは順序を持たない集合型。artisoc4でのagtsetと異なり、位置を指定してエージェントを取り出すことができないことに注意。

❑条件分岐文の活用（周囲の鳥に方向を合わせる）

- if count_agtset(tori_set) > 0:　# もし周囲に鳥がいれば
  one = randchoice(tori_set)　# ランダムに一羽を選ぶ
  self.direction = one.direction　# 自分の向きをその鳥に合わせる

```
8  def agt_step(self):
9
10     # 自分から距離2以内のエージェントをtori_setに格納（自分自身を含まない）
11     tori_set = self.make_agtset_around_own(2, False)
12
13     if count_agtset(tori_set) > 0: # もし周囲に鳥がいれば
14         one = randchoice(tori_set) # ランダムに一を選ぶ
15         self.direction = one.direction # 自分の向きをその鳥に合わせる
16
17     self.forward(1)
18
```

❑モデルの動作を確認する

- 鳥が群れを作ります。

❑本チュートリアル終了時点のモデル

- 途中から来た人や途中でついていけなくなった人は、下記のモデルにアクセスしモデルを継承してください。
  - https://artisoc-cloud.kke.co.jp/models/GM0tA8hfTwKvcfp9M3TWJQ
    - チャットでURLを送ります。
  - 継承：他のユーザが作ったモデルをコピーして編集

おわりに

❑初級チュートリアル、おつかれさまでした！

■ 初級チュートリアルで学んだこと

1. artisoc Cloudの起動方法
2. 新しくモデルを作る方法
3. エージェントの作り方

・・・

これから好きなモデルを好きなだけ作成できます！

→次は何をすればよいのだろう？

❒Step1: artisoc Cloudをはじめる

- 初級チュートリアルに参加する
  - まずはartisoc Cloudを動かしてみよう。
  - コンピュータのなかの人工社会を体感しよう。

- いろんなモデルを見てみる
  - 身の回りの複雑系には、読み物として身近な事例を紹介しています。
  - 渋滞、人のつながり、流行、エスカレーター問題など

- 手を動かしてみる
  - MASの教材にいろんなコンテンツがアップされています。
  - 興味のあるコンテンツをダウンロードしてお試しください。

❒Step2: artisoc Cloudの使い方を学ぶ

- artisoc Cloud教科書を読んで勉強する
  - 書籍『人工社会構築指南』をベースにartisoc Cloud向けに加筆
    - ✓ **講義や研究室ゼミのテキストとしてご利用ください**

- もっと本格的なモデルをつくりたいとき
  - モデル作成のレシピブックには、ポテンシャル法やネットワークモデルによる経路探索など、モデル作成で参考になる手法が学べます。

- 分からないことがあったら質問掲示板
  - モデルを作成していてつまづいたとき
  - モデル作成のヒントが欲しいとき

❒Step3: 勉強会に参加して仲間をつくろう

■ artisoc Cloud勉強会が目指すべきところ

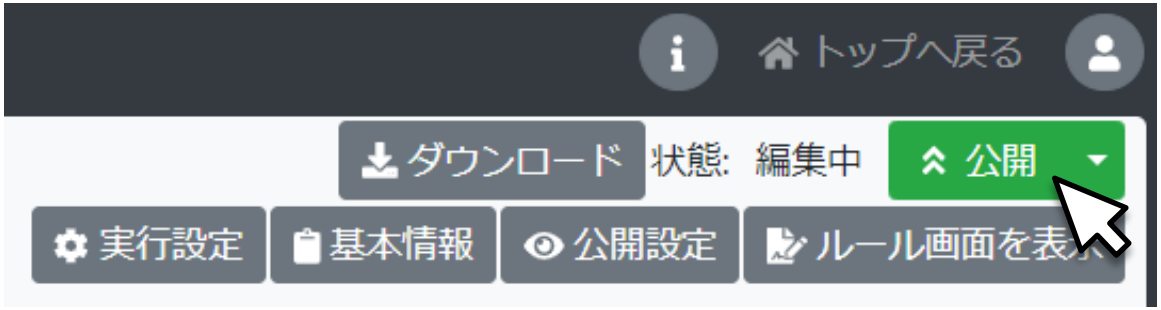
1. 複雑系やMAS(マルチエージェント・シミュレーション)を
学ぶための仲間づくりを行う場を提供したい。

2. 経験や分野の異なる多様な方々が集まり、相互作用することで、
新しい発見や問題解決の場に育てたい。

3. 誰もが簡単に artisoc Cloud を利用して、社会課題について
自分事として考え、行動する社会をつくりたい。

❑モデルの公開

- オリジナルのモデルを作成したら、公開しましょう。
  - 公開すると、ユーザ登録者なら誰でもモデルを見ることができます。
  - コメントなどをつけることも可能です。
  - URLを共有すれば、モデルの共有ができます。



❑参考情報

■ 最新情報はMASコミュニティのほか、SNSでも発信します。

MASコミュニティ
https://mas.kke.co.jp

@artisoc_

@kkeartisoc