artisoc Cloud

マニュアル

株式会社 構造計画研究所

mas-support@kke.co.jp

目次

artisoc Cloud 入門 1

サンプルモデルを実行する 1

口コミモデル・・・情報伝搬シミュレーション 3

口コミモデル・・・モデルの設定変更 4

簡単なモデルを構築する 11

トップページ 12

画面説明 12

情報表示ボタン 14

ユーザボタン 15

モデルリスト 16

ログインとユーザページ 17

ログイン 17

ユーザページ 18

モデルを閲覧する 20

モデルを編集する 25

基本情報設定 26

ルールの編集 27
エージェント種別の新規作成 32
変数の新規作成 34
出力設定 39
マップ出力設定 41
時系列グラフ出力設定 46
棒グラフ出力設定 48
値出力設定 59
コントロールパネル設定 61
ボタン 63
トグルボタン 64
スライドバー 65
直接入力 66
リスト 67
ルール 68
実行設定 69
シミュレーション設定 69
モデルを共有する 71
モデルのダウンロード・アップロード 71

モデルの保存・公開・削除 72
ルール文法 75
概要 75
Python の基礎文法 75
変数・代入・演算 75
コメントアウト 77
関数 78
if 文（条件分岐文） 78
for 文（繰り返し文） 79
変数の型 80
モデルツリーの構成 83
ルールの全体構成 84
変数の呼び出し 86
ツリー要素の呼び出し 86
エージェント変数の呼び出し 88
artisoc Cloud の関数種別 89
Python 組み込み関数 90
artisoc Cloud 組み込み関数 90
ユーザ定義関数 92

モジュール・ライブラリ 94

artisoc Cloud 入門

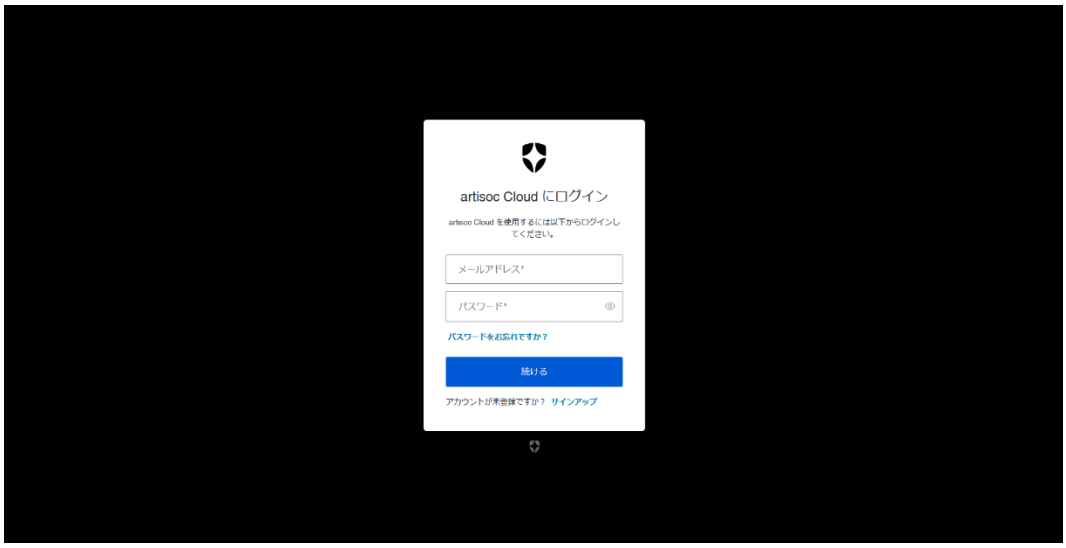
サンプルモデルを実行する

artisoc Cloud には、実際にどのように動くのか体感してもらうために、サンプルモデルが用意されています。artisoc Cloud のトップページにアクセスしてモデルを開けば、すぐにシミュレーションを開始することができます。

ログイン

モデルを実行、継承、編集するためには、artisoc Cloud ユーザアカウントでログインする必要があります。（まだ artisoc Cloud のユーザアカウントを作成していない場合、「新規登録」ボタンから、ユーザアカウントを作成してください）。 画面右上の「ログイン」をクリックすると、ログイン画面が表示されます。ユーザアカウントのメールアドレスとパスワードを入力して「ログイン」ボタンをクリックしてください。





モデルを検索

それでは、実際に試してみながら、artisoc Cloud の使い方に慣れていくことにしましょう。 artisoc Cloud のトップページ上の検索バーに「口コミモデル mas_support」と入力して Enter を押してください。



口コミモデルが表示されます。モデルのタイトルをクリックして開いてみましょう。

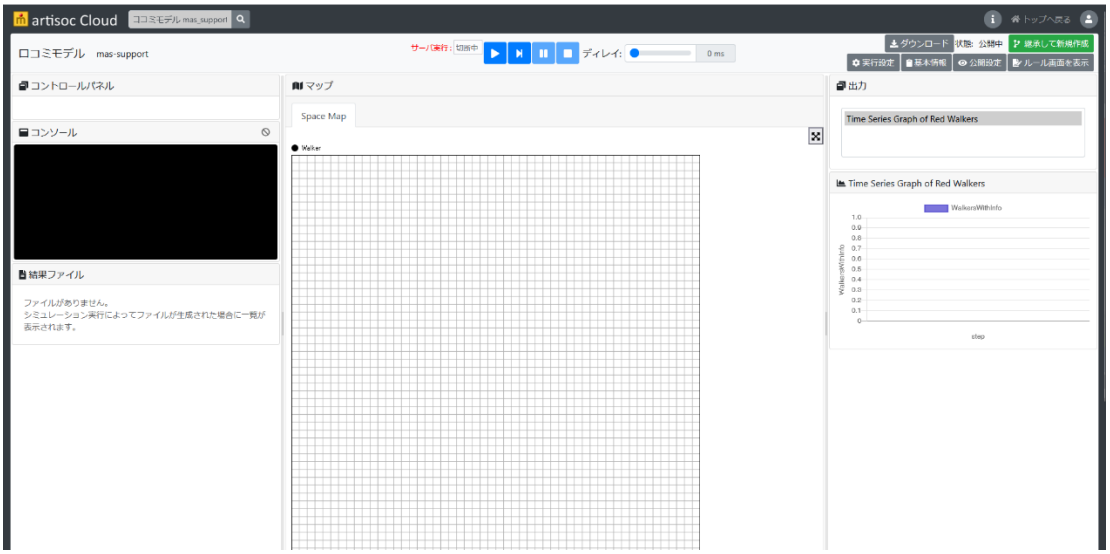


口コミモデル・・・情報伝搬シミュレーション

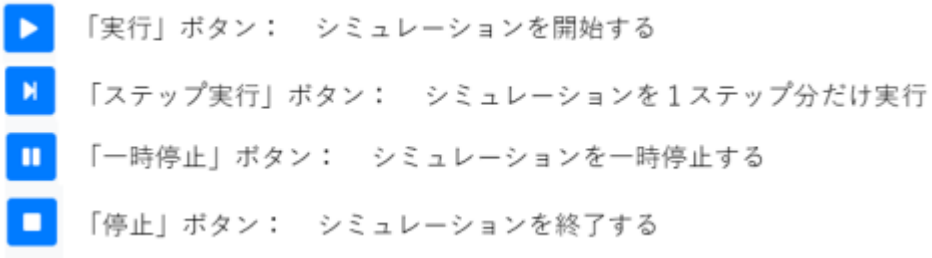
このシミュレーションについて

情報を持った少人数が、口コミで情報を流すことによって、うわさが広まっていく過程をシミュレートするモデルです。「二次元空間」上を、情報を持った「歩行者」エージェント（赤）がぶらついています。情報を持たないエージェント（緑）とぶつかると情報は伝達され、ぶつかられたエージェントの色は緑から赤へ変わります。次第に赤色のエージェントが増えていき、うわさは広まります。

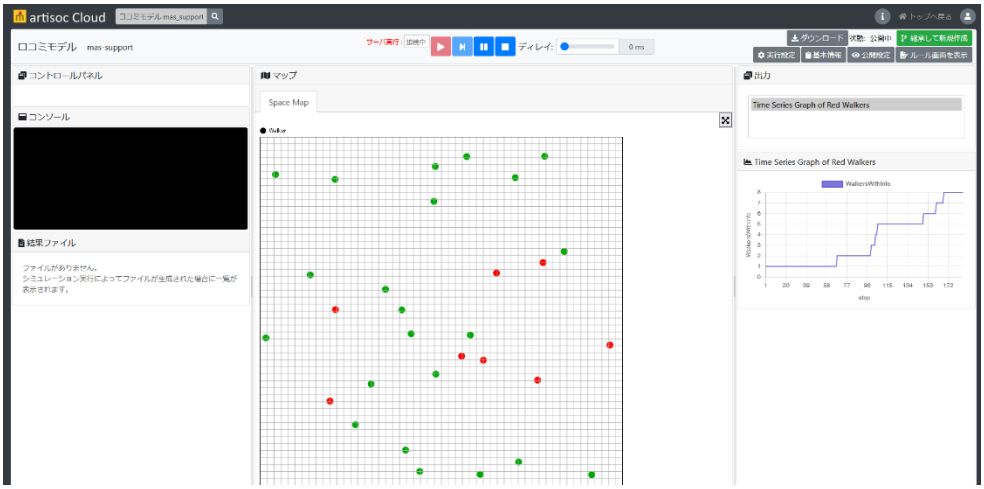
もちろん、これらの設定はユーザの皆様が自由に変更することもできますが、まずは設定はこのままで実行してみましょう。「口コミモデル」を開くと、ブラウザに次の図のようなモデル実行画面が表示されます。



それでは実際に実行してみましょう。実行させるには、画面上部にある実行パネルを使います。



さて、それでは実行してみましょう。「実行」ボタンをクリックしてみてください。次のような画面が表示されましたか？



実行結果

シミュレーションを実行すると「マップ（SpaceMap）」「出力（Time Series Graph of Red Walkers）」「コントロールパネル」「コンソール」「結果ファイル」などが表示されます。 シミュレーションは「停止」か「一時停止」を押すまで、または全てのエージェントに情報が伝わるまで実行され、リアルタイムに結果を表示します。

口コミモデル・・・モデルの設定変更

さて、前項において実際にモデルを実行してみましたが、このままでは一通りのシミュレーションしか実行することが出来ません。ここでは、設定を変更することにより、違う結果を導き出してみましょう。

artisoc Cloud では他のユーザが「公開」したモデルを「継承」して、自分の新しいモデルを作成することができます。口コミモデルを継承して、モデルの設定を変更することにより、違う結果を導き出してみましょう。

モデルの継承

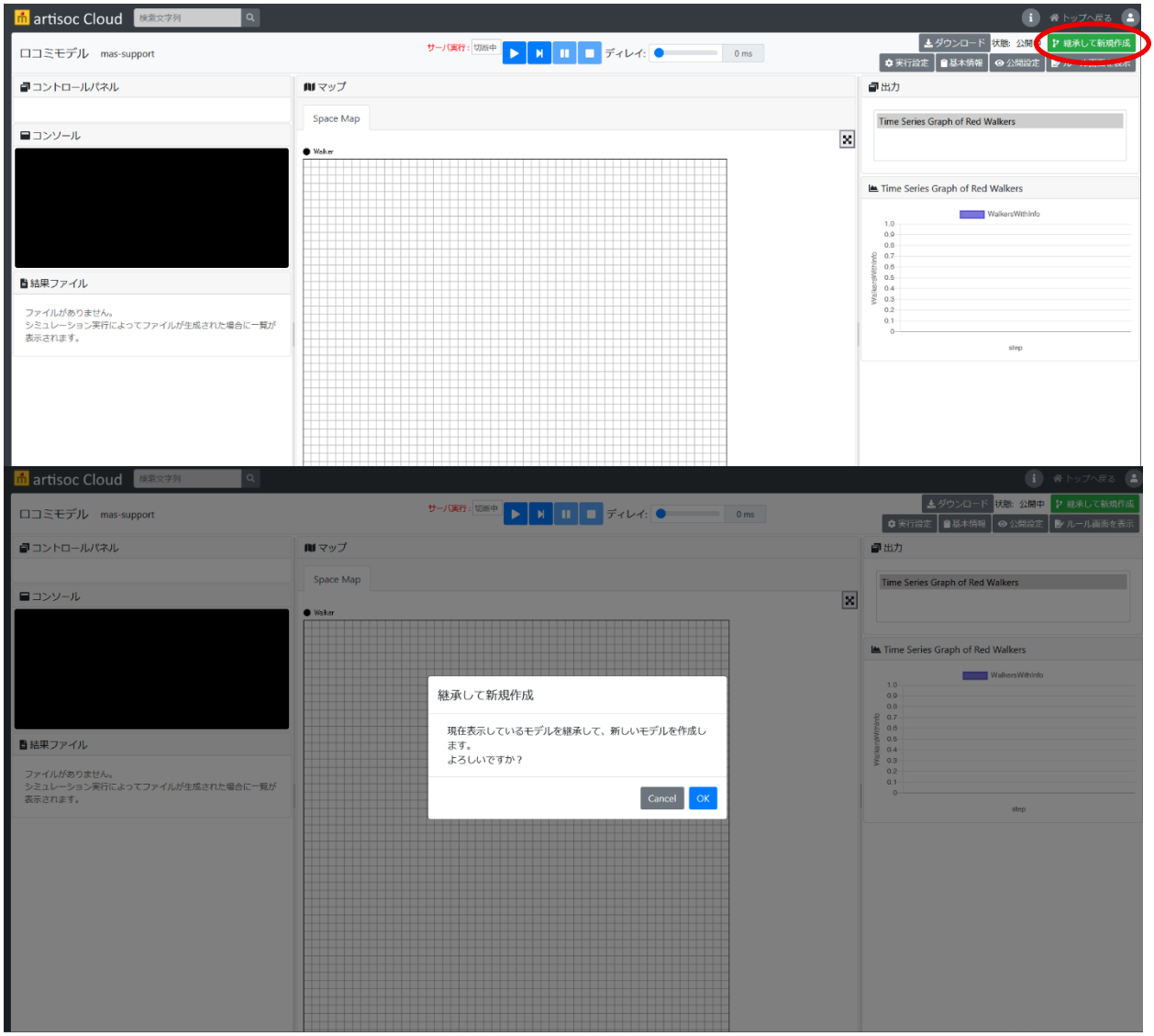
以下の手順で「口コミモデル」を継承して、自分の新しいモデルを作成します。

1.「口コミモデル」を開きます。

2.「口コミモデル」の画面右上に表示される「継承して新規作成」ボタンをクリックします。

3. 継承して新規作成　確認ダイアログが表示されます。「OK」をクリックします。

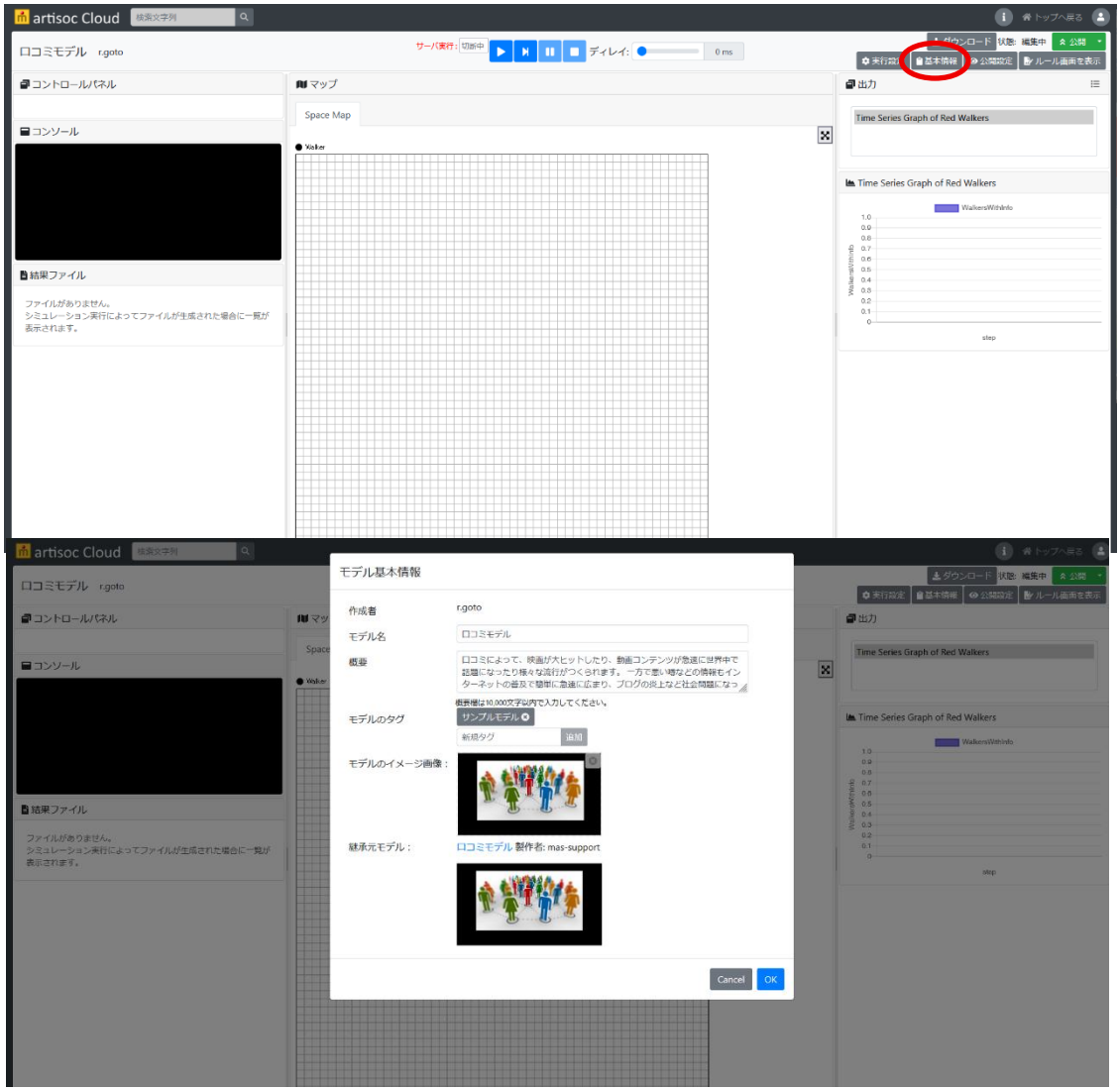
これで、「口コミモデル」が継承されて編集ができるようになりました。モデルの設定を変更してみましょう。



モデルの名前を変えてみよう

モデルを編集する前に、モデルの名前を変えてみましょう。手順は次の通りです。

1. 画面右上の「基本情報」ボタンをクリックします。⇒モデル基本情報ダイアログが表示されます。
2. モデル基本情報画面ダイアログで、「モデル名」を「口コミモデル　空間の大きさを変えた」に変更します。
3. モデル基本情報画面ダイアログで、「OK」ボタンを押してダイアログを閉じます。



空間の大きさを変えてみよう

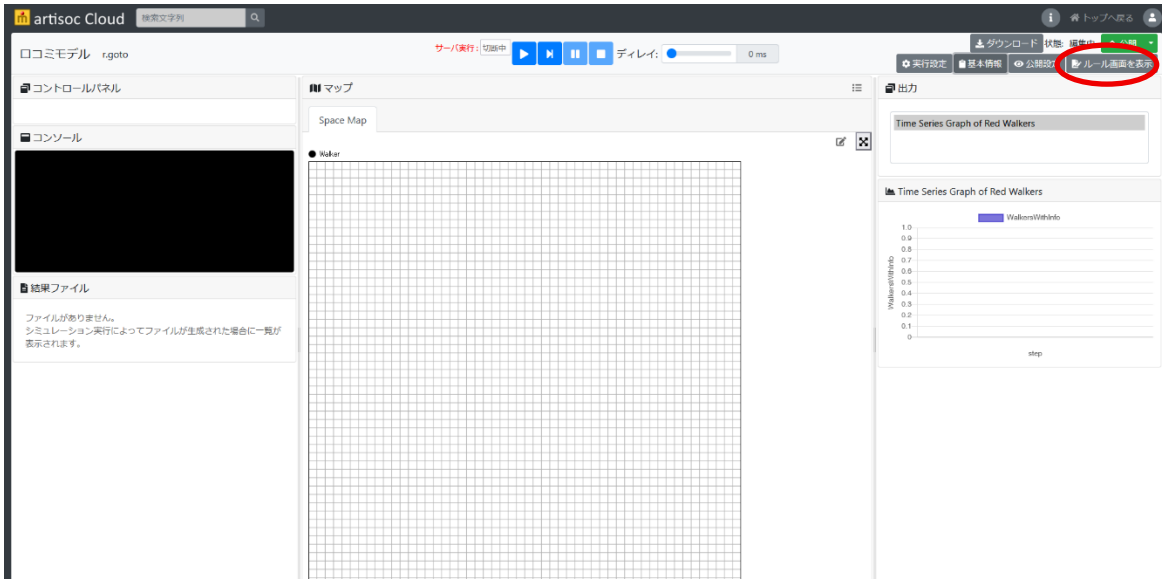
元のモデルでは 50×50 の二次元空間上を歩行者たちが移動していましたが、空間の大きさを 100×25 に変えてみましょう。手順は次の通りです。

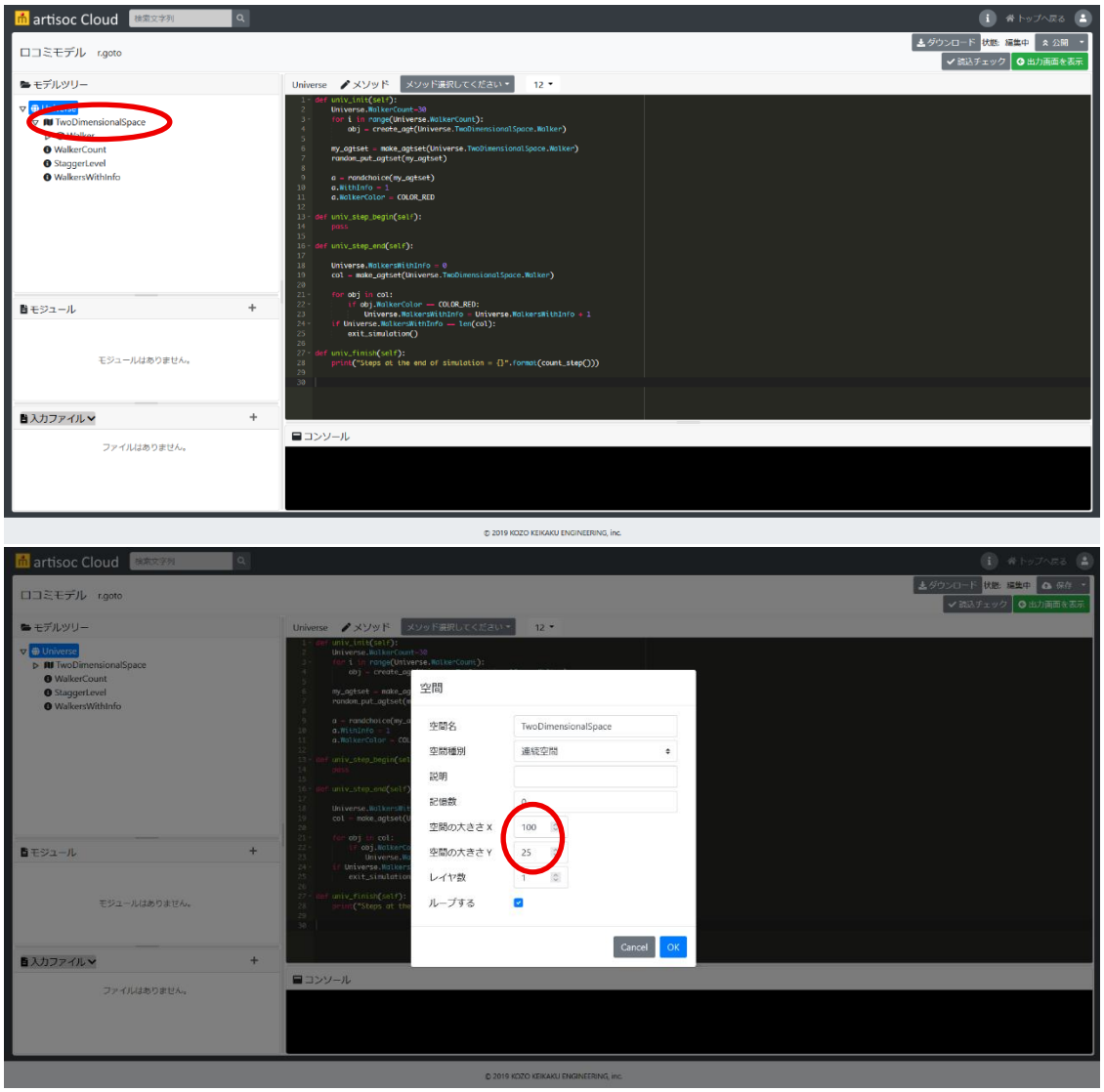
1.画面右上の「ルール画面を表示」ボタンをクリックします。⇒ルール編集画面が表示されます。

2.ルール編集画面 左のモデルツリーに表示されている TwoDimensionalSpace 空間にマウスを合わせます。右側に表示される編集ボタンをクリックします。⇒空間設定ダイアログが表示されます。

3.空間設定ダイアログで、「空間の大きさ X」を 100,「空間の大きさ Y」を 25 に変更します。

4.「OK」ボタンを押して空間設定ダイアログを閉じます。画面右上の「ルールの表示を終わる」をクリックします。⇒ルール編集画面が表示されます。



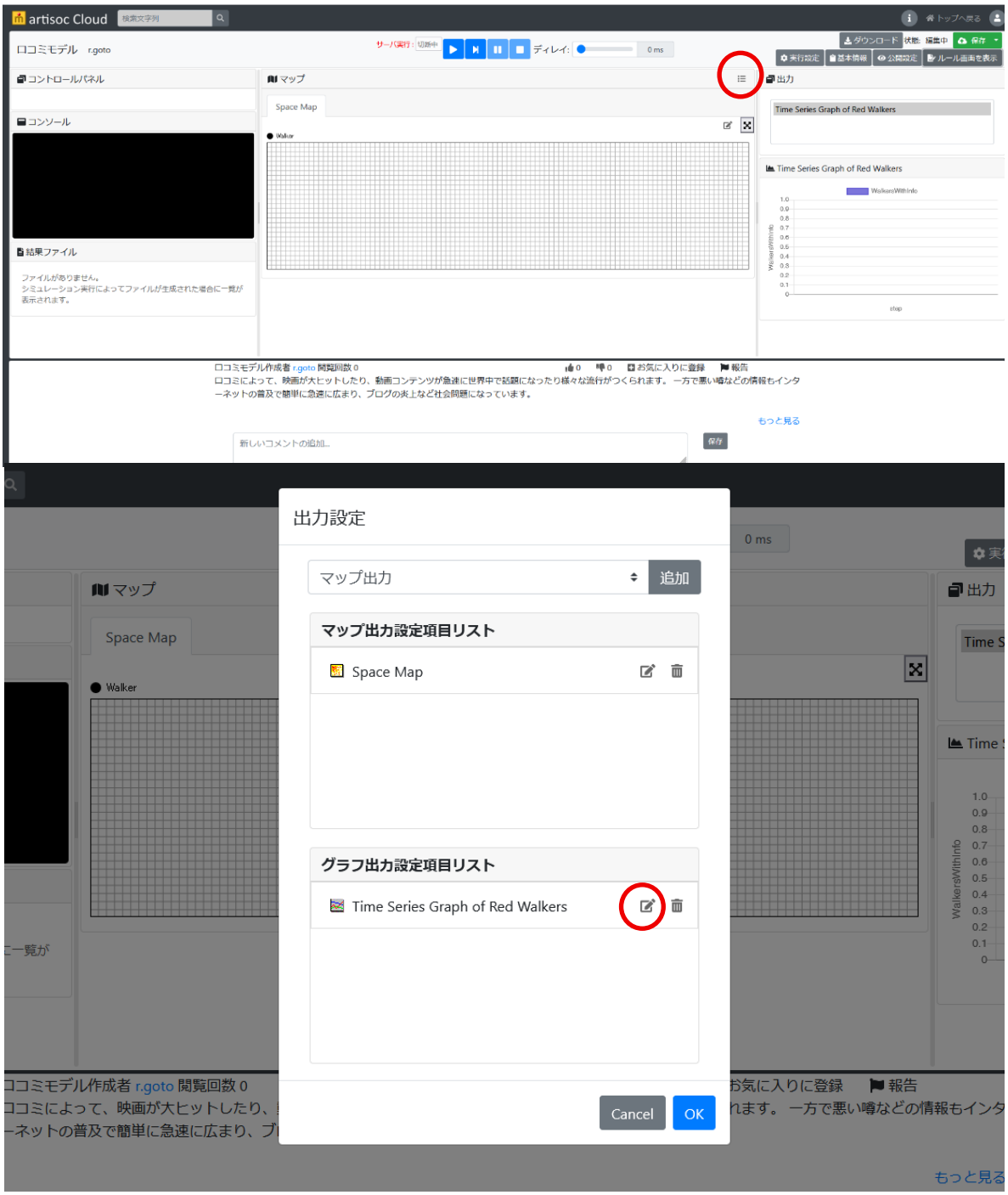


時系列グラフの出力を変えてみよう

次にここでは、時系列グラフ（人数）の折れ線の頂点部分にマーカーを追加し、線を太くして、グラフを見やすくしてみましょう。手順は次の通りです。

1.マップの右上にマウスカーソルを合わせると表示される設定ボタンをクリックします。出力設定ダイアログが表示されます。

2.リストの中の Time Series Graph of Red Walkers の右に表示されている編集ボタンをクリックします。⇒時系列グラフ設定ダイアログが表示されます。



1.「時系列グラフ要素リスト」の「WalkersWithInfo」の「編集」ボタンをクリックします。⇒時系列要素設定ダイアログが表示されます。

2.「線の太さ」を 2pt から 4pt に変更し、「印の大きさ」をなしから 5pt に変更し、「OK」ボタンをクリックして時系列要素設定ダイアログを閉じます。

3.時系列グラフ設定ダイアログで「OK」ボタンをクリックしてダイアログを閉じます。

4.出力設定ダイアログで「OK」ボタンをクリックしてダイアログを閉じます。

5.画面右上にある「保存」ボタンをクリックして、変更を保存します。



さて再び準備が出来ました。実行ボタンをクリックしてみましょう。



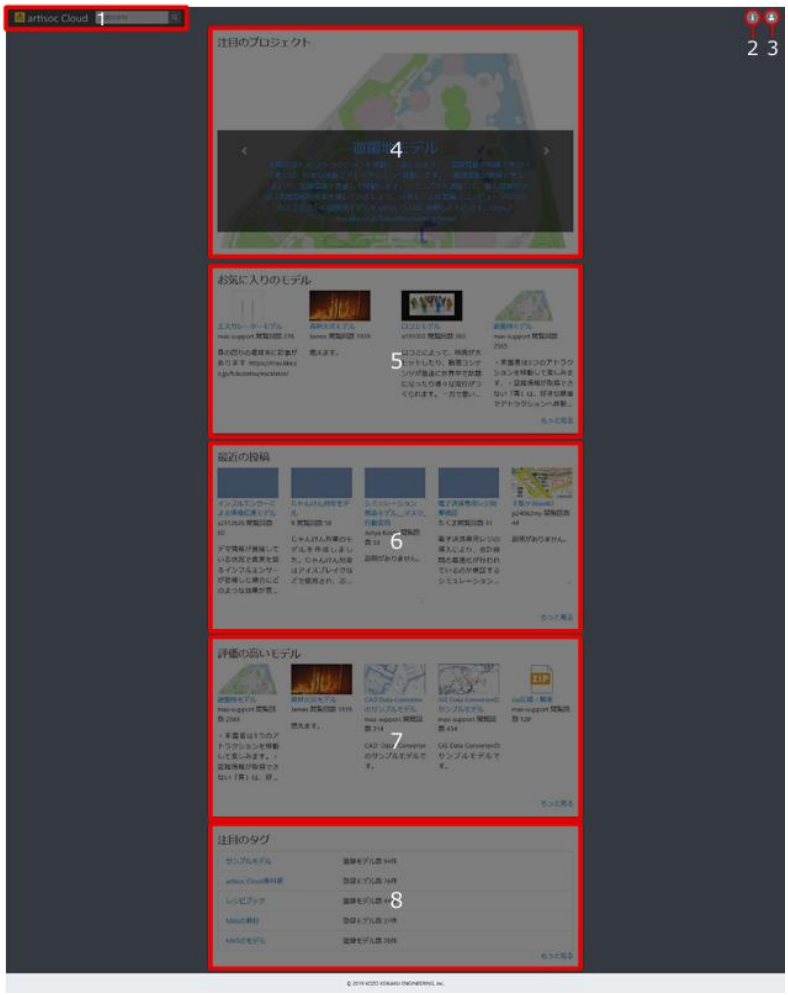
簡単なモデルを構築する

ここまでで、他のユーザが作成したモデルを実行する方法、継承して編集する方法を学びました。まったく白紙の状態からエージェントモデルを構築するための基本的な要素と手順について、「artisoc Cloud チュートリアル」にまとめてありますので、まずはこちらを参照しながら、一通りの手順を学んでみてください。

トップページ

トップページでは様々なモデルを探すことができます。

画面説明



1.検索バー

モデルの検索ができます。検索ワードを入力し検索ボタンをクリックすると、検索ワードをモデル名、概要欄、作成者のユーザ名のいずれかに含むモデルが抽出されます。

検索バーはすべてのページで表示されます。

2.情報表示ボタン

チュートリアルの閲覧や artisoc の仕様に関するフィードバックを送信することができます。詳細は 2.1.2 を参照してください。

3.ユーザボタン

ログイン時のみ表示されます。

ログインしていない状態では  が表示されます。

ユーザ情報ボタン、または  はすべてのページで表示されます。

詳細は 2.1.3 を参照してください。

4.注目のプロジェクト

公開されたモデルをループ表示します。モデルタイトルをクリックするとモデル実行画面に移動します。

5.お気に入りのモデル

お気に入り登録をしたモデルが表示されます。

お気に入りのモデル欄は、ログイン状態でのみ表示されます。

モデルタイトルをクリックするとモデル出力画面に移動します。

6.最近の投稿

最近投稿されたモデルが表示されます。

7.評価の高いモデル

評価の高いモデルから順に表示されます。

8.注目のタグ

モデルの登録数の多いタグが表示されます。

タグをクリックするとモデルリストページに移動し、選択したタグで抽出されたモデルがリスト表示されます。

「もっと見る」をクリックすると、タグ一覧ページに移動します。

情報表示ボタン

クリックするとポップアップが表示され、以下の操作を選択できます。



チュートリアル：チュートリアルに移動します。実際のモデルを用いた作成手順の学習ができます。
マニュアル：マニュアルに移動します。artisoc の仕様を確認することができます。
関数仕様：関数仕様に移動します。artisoc で用いる関数を検索することができます。
利用規約：利用規約を確認することができます。
質問掲示板：質問掲示板に移動します。artisoc に関するフィードバックがありましたら、ぜひお気軽にご送信ください。
MAS コミュニティ：MAS コミュニティに移動します。MAS 関連のイベントのご案内や、実際の研究に利用された artisoc モデルの公開も行われております。

ユーザボタン

クリックするとポップアップが表示され、以下の操作を選択できます。

ユーザ情報：新規登録時に登録したユーザ情報を確認、更新できます。



ユーザページ：ユーザページに移動します。
新規モデルの作成：新規モデルの作成を開始します。

モデルアップロード：ローカルに保存してあるモデルのアップロードをします。
ログアウト：ログアウトします。

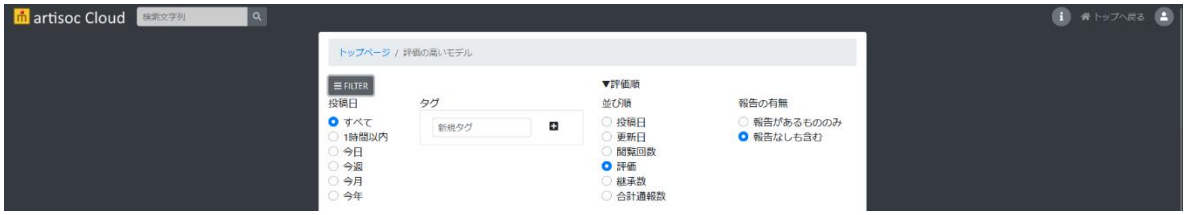
モデルリスト

指定された条件でモデルを抽出し、表示することができます。



1. FILTER ボタンをクリックすると下のような画面が表示されます。

タグ欄では検索したいタグ名を新規タグ欄に入力し、を押すことで検索対象のタグを追加できます。タグは複数追加することができ、AND 検索が可能です。



2.並び順で選んだ項目が表示されます。

▼をクリックすると、並び順の昇順、降順を切り替えられます。

ログインとユーザページ

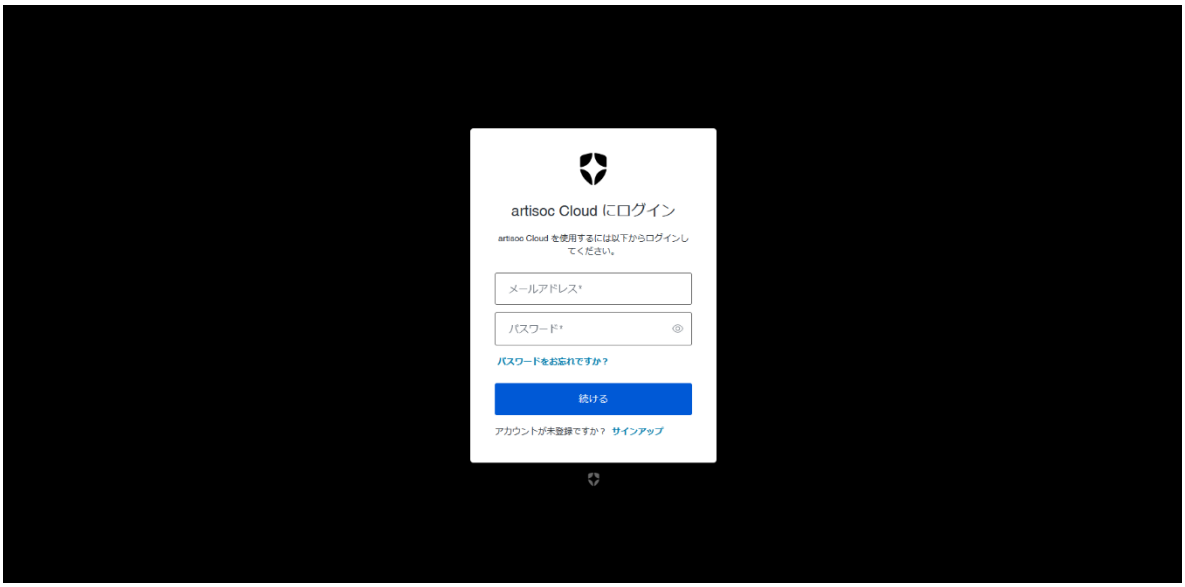
ログイン

ログイン方法について説明します。

まだ artisoc Cloud のユーザアカウントを作成していない場合、画面右上のをクリックしてユーザアカウントを作成してください。

1. 画面右上のをクリックします。

2. 下記のログイン画面が表示されるので、メールアドレスとパスワードを入力し、ログインボタンをクリックします。



ユーザ情報

ユーザ情報では、ユーザの基本情報とライセンスを確認することができます。

基本情報から、氏名やニックネーム等を更新することができます。

ライセンスから、ユーザに付与されているライセンスの期限や各機能の使用可否を確認することができます。

ユーザページ

ユーザページは、作成したモデルなどを閲覧できるユーザごとのページです。



1.作成したモデル

ユーザの作成したモデル一覧が表示されます。

2.お気に入りのモデル
ユーザがお気に入り登録したモデルが表示されます。

3.投稿したコメント
ユーザが投稿したコメントがモデルごとに表示されます。

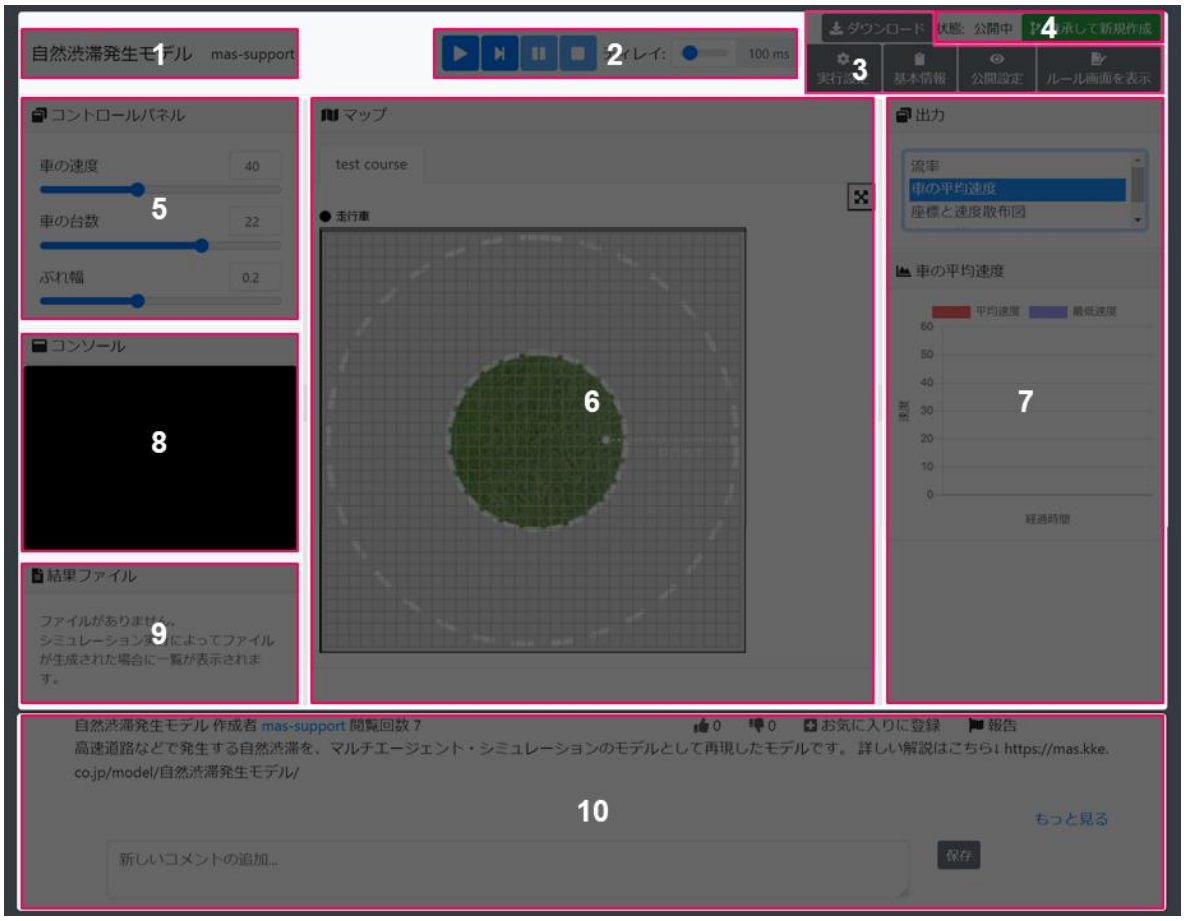
4.トップへ戻る
クリックするとトップページに遷移します。
シミュレーション実行画面、編集画面でも同様です。

モデルを閲覧する

出力画面説明

モデルが出力される画面です。

各領域の境界をドラック＆ドロップすることで表示サイズを調整することができます。



1.タイトル/作成者表示

モデルのタイトルとモデル作成者のユーザ名を表示します。

2.実行コントロールバー
2.3.2 を参照してください。

3.ツールボタン

：モデルを「.json」形式でダウンロードします。

：編集中のモデルでのみ操作できます。モデルの編集については 2.4 を参照してください。

：モデル基本情報（作成者、モデル名、概要、モデルのタグ、モデルのイメージ画像）を閲覧できます。

：編集のモデルでのみ操作できます。モデルを一般公開にするか限定公開(URL を知っている人のみがアクセス可能)にするか、選択できます。

：モデルのルール画面を表示します。

4.状態設定 / アクションボタン
モデルの状態や保存、公開などの操作ができます。状態の種類とアクションボタンの詳細については 2.4.3 を参照してください。
ボタンについては 2.4.1 を参照してください。

5.コントロールパネル
スライドバーなどを使って変数の値を変えることができます。
シミュレーション実行中にコントロールパネルを操作すると、実行中のシミュレーションに値が反映されます。

表示されるコントロールパネル要素については 2.4.10 を参照してください。

6.マップ表示

二次元の表示マップ上にエージェントや変数を配置し、それらの移動や増減を見る出力方法です。

マウスホイールの操作でマップの拡大・縮小表示ができます。また、マップ右上のをクリックすることでマップが全画面表示されます。

マップの出力がないモデルではこのエリアは表示されません。

マップが複数出力設定されている場合は、マップエリア上部のタブで表示するマップを切り替えることができます。

7.グラフ表示

変数の値をグラフに出力することができます。

グラフ表示エリア上部のグラフリストには、設定したすべてのグラフが表示されます。

デフォルトではすべてのグラフが出力されますが、リストをクリックして表示するグラフを選択することができます。Ctrl キー、Shift キーを押下しながらクリックすることで、複数選択、範囲選択を行うこともできます。

詳しくは 2.4.9.1－2.4.9.7 を参照してください。

8.コンソール画面

ユーザがルール中で標準出力（print）した結果やルールのエラーが表示されます。

コンソール画面にカーソルを合わせると表示されるをクリックすることで、コンソール画面をクリアすることができます。

9.結果ファイルリスト

シミュレーションの実行によって出力されたファイル一覧が表示されます。ファイル名をクリックすることでダウンロードできます。

をクリックすることでファイルを削除することができます。

をクリックすることで表示ファイルを更新することができます。

10.概要/評価・コメント欄



モデル名、作成者のユーザ名、閲覧回数、モデル概要等が表示されます。

なお、ユーザ名をクリックすると該当ユーザのユーザページへ移動します。

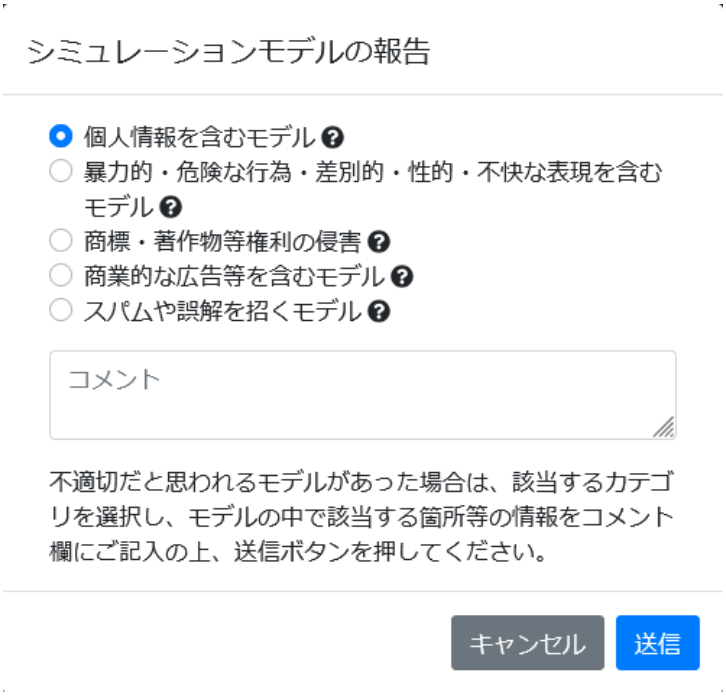
：モデルに高評価・低評価を付けることができます。クリックすると青くなり、もう一度クリックするとキャンセルできます。マークの右横には高評価・低評価の数が表示されます。

：クリックすることでモデルをお気に入りに登録することができます。お気に入りに登録したモデルはユーザページから閲覧できます。

：不適切なモデルを運営に報告することができます。

クリックすると下記のウィンドウが表示されます。

「？」にカーソルを合わせると詳しい説明が表示されます。



コメントの投稿：コメントを投稿するには、コメント欄にコメントを入力し、をクリックします。投稿したコメントはすべてのユーザが閲覧できます。コメントにカーソルを合わせると表示されるメニューから、コメントの編集と削除を行うことができます。

実行コントロールバー モデルを実行するには次の実行コントロールバーを使用します。



1.実行ボタン

シミュレーションを開始するときに、クリックします。

2.ステップ実行ボタン

シミュレーションを１ステップ分だけ実行します。

続けてクリックすれば、その都度１ステップずつ実行して、その後一時停止状態になります。

3.一時停止ボタン

シミュレーションを一時停止させる時に、クリックします。

4.停止ボタン

シミュレーションを終了させる時に、クリックします。

5.ディレイ設定

1 ステップ終了した段階で、この数値の時間だけ、処理をウェイトします。シミュレーション速度を調整したい場合などに、適当な数値を入力してください。

モデルを編集する

新規作成・継承して新規作成 ユーザボタンの「新規モデルの作成」から新規モデルまた、モデルの出力画面で 継承して新規作成 ボタンをクリックすると、開いているモデルをコピーして編集を開始することができます。

基本情報設定

基本情報 をクリックすることで下記のウィンドウが表示されます。



1.作成者

ユーザ名が表示されます。

2.モデル名

モデル名を編集できます。

3.概要

概要欄を編集できます。

4.モデルのタグ

タグを追加・削除することができます。

タグの追加は入力欄にタグ名を記入し、追加ボタンをクリックします。

タグの削除は各タグに表示されている「×」ボタンをクリックします。

5.モデルのイメージ画像

モデルのイメージ画像を設定できます（任意）。

設定可能なファイルは.png、.jpg で、画像サイズは最大で 1366×768 です。

画像部分をクリックすると表示されるファイル選択ダイアログでファイルを選択するか、ドラッグ＆ドロップによりファイルを指定できます。

選択した画像はトップページやモデルリストページに表示されます。

6.継承元モデル

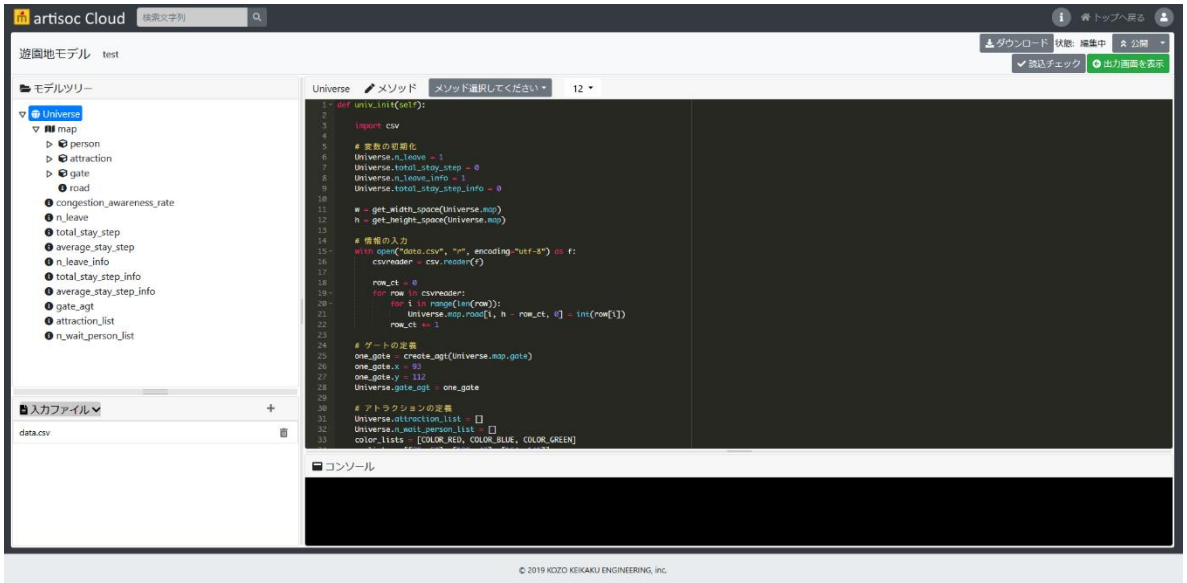
継承して新規作成した場合に、継承元のモデルが表示されます。

新規作成の場合は表示されません。

ルールの編集

　をクリックするとルール編集画面に移動します。

ルール編集画面では各領域の境界をドラックすることで表示サイズを調整することができます。



1.モデルツリー

各要素を階層構造で表示します。

ツリーの中には必ず一つの「Universe」が存在します。

そしてその下に「空間」や「エージェント」、「変数」などの要素を配置して、シミュレーションを設定します。

追加できる要素は次の表のとおりです。操作方法は 2.4.4 以降を参照してください。

モデル要素	持つことができるモデル要素	ルール定義	説明
Universe	空間 エージェント Universe変数	可	モデルに必ず一つ存在する要素。 ツリーのトップ要素としてデフォルトで表示される。
空間	（空間）エージェント 空間変数	不可	エージェントが動作する二次元座標を持った空間を表現するモデル要素。 Universeの下に配置される。
エージェント	エージェント変数	可	シミュレーションの行動主体。 Universe, 空間の下に配置される。
変数	なし	不可	モデル要素の状態を示す変数。

2.入力ファイル

モデルで入力するファイルをリストで表示します。

「ファイルの追加」、「ファイルの削除」の操作を行うことができます。

[ファイルの追加]

「＋」ボタンをクリックするとファイルインポートダイアログが表示されます。

網掛け領域をクリックすると表示されるファイル選択ダイアログでファイルを選択するか、ドラッグ＆ドロップによりファイルを指定できます。



[ファイルの削除]

ファイル名の横に表示されるごみ箱ボタンをクリックすることでインポートするファイルリストから削除することができます。

3.ルールエディタ

Universe やエージェントのルールを記述することができます。

操作方法については 2.4.8 を参照してください。

ルールの記述方法については 3.ルール文法を参照してください。

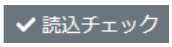
4.コンソール

ユーザがルール中で標準出力（print）した結果やルールのエラーが表示されます。

コンソール画面にカーソルを合わせると表示されるをクリックすることで、コンソー

ル画面をクリアすることができます。

5.ツールボタン



：ルールの読込チェックを行い、結果をコンソールに表示します。



：モデルの出力画面に移動します。

空間の新規作成 エージェントが平面上で移動する様を観察するために、空間を作成することが出来ます。

[手順]

1.モデルツリーの Universe にカーソルを合わせ、表示される「＋」をクリックします。

2.タイプの選択ダイアログが表示されるので「空間の追加」を選択します。



3.次のような「空間プロパティ」ダイアログが開きます。



■空間名： ユーザが任意の空間名を入力します。例えば、“map”と入力します。

■空間種別： 空間の種類を選択します。連続空間と四角格子空間のどちらかを選択します。

空間種別により使用できる組み込み関数が異なります。 連続空間：主にエージェントが連続的に動くときに用います。 四角格子空間：主にエージェントが離散的に動くときに用います。

■説明（任意）： 空間の説明を入力することができます。

■空間の大きさ： エージェントや変数一つをひとマスとした空間の広さを設定します。

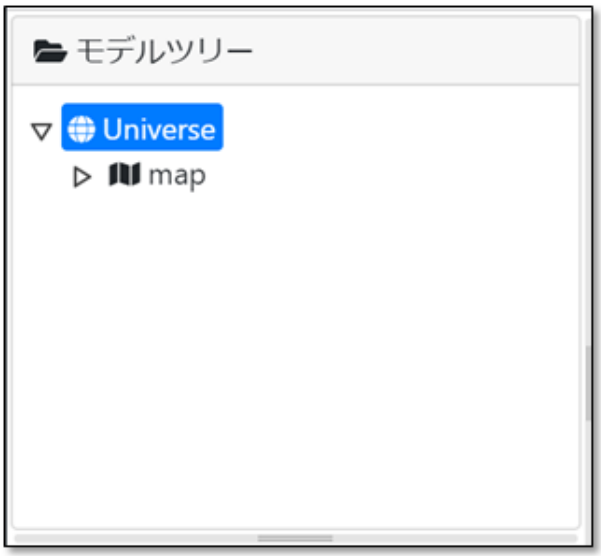
X 軸（横方向）、Y 軸（縦方向）を指定します。初期値では 50×50 です。

■レイヤ数： 空間のレイヤ数（階層）を指定します。初期値では 1 です。

■ループする： チェックを入れると、要素が空間の端まで来た時に反対側から現れます。

「ループしない」場合は空間の端に留まります。

4.「OK」ボタンをクリックするとダイアログが閉じ、モデルツリーに次の図のような空間が出現します。 （出ない場合は Universe 左の▷をクリックしてください）



Universe の左に表示されている▽をクリックすることで、Universe 以下の要素の表示、非表示が切り替えられます。

「空間プロパティ」は空間にカーソルを合わせ、表示されるをクリックするか空間名をダブルクリックすることでも表示できます。 また、ごみ箱ボタンをクリックすることで該当する空間を削除できます。

エージェント種別の新規作成

空間上にエージェント種別を新規作成します。

エージェント種別は空間とは違って Universe の直下でも作成することが可能ですが、ここでは空間上で作成する場合を例として示します。

[手順]

1. エージェントを作成する空間（ここでは“map”）にカーソルを合わせ、表示される「＋」をクリックします。

2.タイプの選択ダイアログが表示されるので「エージェントを追加する」を選択します。



3. 次のような「エージェントプロパティ」ダイアログが開きます。



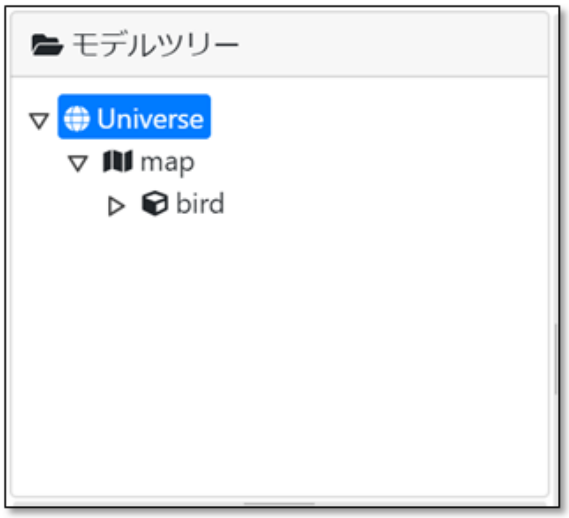
■エージェント名： 任意のエージェント名を入力します。例えば“bird”と入力します。

■説明（任意）： エージェントの説明を入力することができます。

■記憶数： artisoc では変数の過去の状態(値)を残しておくことができます。何ステップ分の状態を残すかを記憶数で指定します。

過去の状態は get_history 関数で取得することが可能です。

4.「OK」ボタンをクリックするとダイアログが閉じ、モデルツリーに次の図のようなエージェントが出現します。（出ない場合は Map 左の▷をクリックしてください）



Universe の直下に新規作成する場合も、同じ手順で作成します。

＜注意＞：「id、x、y、direction、layer」という名の５つの変数が自動的にエージェント種別の下に配置されます。

「ID」はエージェント種別毎にエージェントの１つずつを見分けるためにつけられる番号（0 以上）を表します。その他の変数は、空間上のエージェントの座標を表す変数で、空間が存在しない場合は作成されません。

なお、これらの変数は削除できません。

空間名の左に表示されている▽をクリックすることで、空間以下の要素の表示、非表示が切り替えられます。

「エージェントプロパティ」はエージェントにカーソルを合わせ、表示される▽をクリックするかエージェント名をダブルクリックすることでも表示できます。

また、ごみ箱ボタンをクリックすることで該当するエージェント種別を削除できます。

変数の新規作成

エージェントに変数を新規作成します。

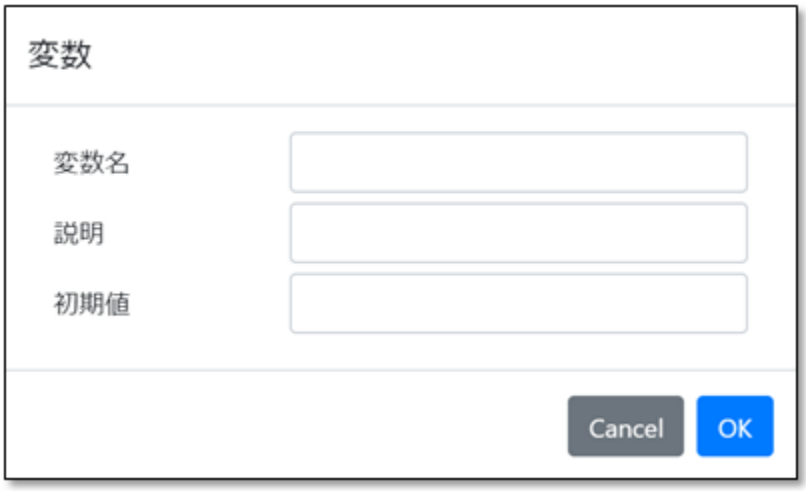
変数はエージェントの下なら何処にでも作成することが出来ますが、ここではエージェントの変数を作成する場合を例として示します。

［手順］

1. 変数を追加するエージェント（ここでは“bird”） にカーソルを合わせ、表示される「＋」をクリックします。

2.すると次のような「変数プロパティ」ダイアログが開きます。

（Universe、空間の下に変数を作成する場合は「タイプの選択」ダイアログで「変数を追加する」を選択してください。）



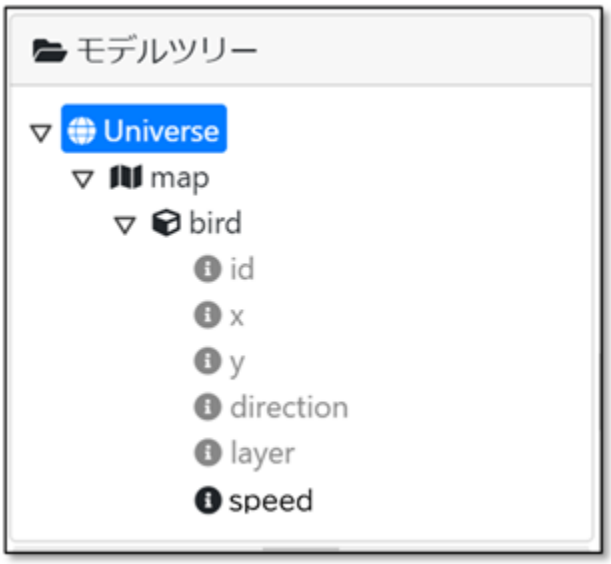
■変数名： 任意の変数名を入力します。例えば“speed”と入力します。

■説明（任意）： 変数の説明を入力することができます。

■初期値（任意）： 変数の初期値を設定することができます。

空欄の場合、変数の初期値は None になります。

3.「了解」ボタンをクリックするとダイアログが閉じ、ツリーに次の図のような変数が出現します。

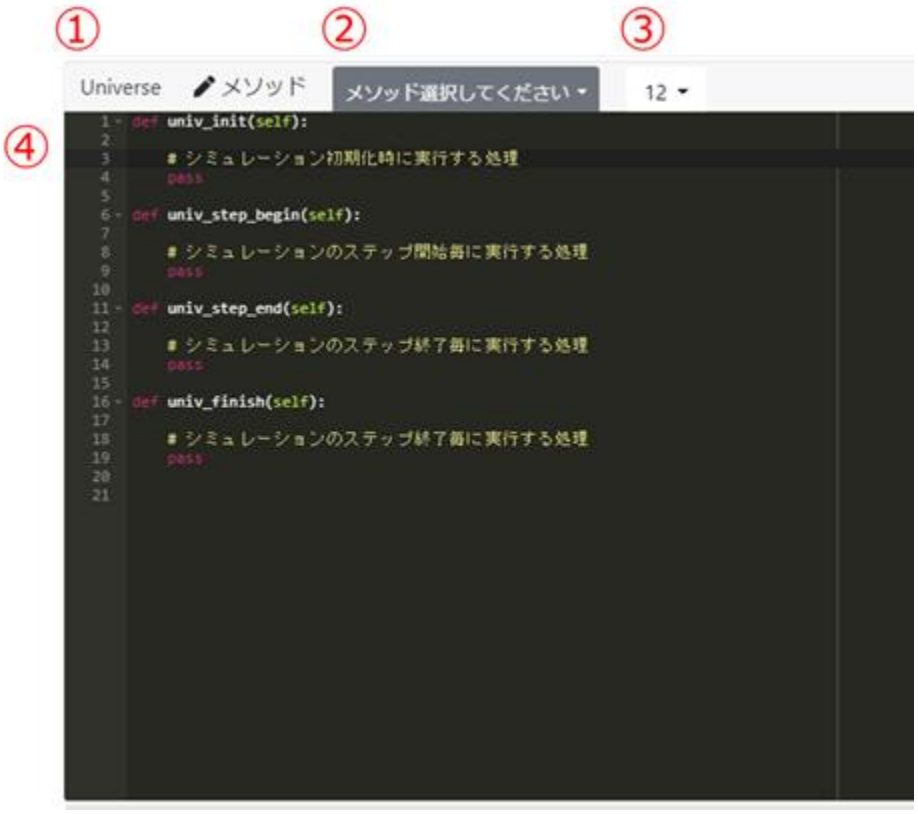


　エージェント名の左に表示されている▽をクリックすることで、エージェント以下の要素の表示、非表示が切り替えられます。
「変数プロパティ」はエージェントにカーソルを合わせ、表示されるをクリックするか変数名をダブルクリックすることでも表示できます。
また、ごみ箱ボタンをクリックすることで該当する変数を削除できます。

ルールの記述

　シミュレーションを動かすには Universe や各エージェントに対してルールを書きます。Universe とエージェント以外の要素にルールを書くことはできません。

本項ではルールエディタの使い方を説明します。ルール文法については 3.ルール文法を参照してください。



1. ルール表示中のモデル要素名

現在表示しているモデル要素名が表示されます。

ルールを表示するモデル要素を切り替える場合はモデルツリーの要素名をクリックします。

2. メソッド選択

クリックすると表示中ルール内のメソッド一覧が表示されます。

メソッド名をクリックすると該当メソッドまでカーソルが移動します。

3. フォントサイズ

ルール編集エリア内の文字サイズを変更できます。

4. ルール編集エリア

ここにルールを記述します。

あらかじめ必要な関数が表示されています。

文字を入力すると入力補完候補が表示され、↑↓キーと Enter キーで候補から選ぶことができます。

ルール編集エリアを選択した状態のとき、ctrl+F で検索、ctrl+H で置換ができます。

出力設定

出力画面の出力設定から、マップへの描画やグラフの出力が設定できます。
設定画面はグラフ表示エリアにカーソルを合わせると表示される　をクリックすると開きます。



出力設定画面が開いたら  をクリックします。出力可能な項目が表示されるので設定する項目を選択します。
出力可能な項目は以下の通りです。設定方法については各項を参照してください。

- マップ出力
- 時系列グラフ
- 棒グラフ
- 円グラフ
- 散布図
- 折れ線グラフ
- ヒストグラム
- 値出力

出力設定項目リストには出力設定済みのマップもしくはグラフ名が表示されます。
マップもしくはグラフ名はドラッグ&ドロップすると順番を並び替えることができ、このリストでの順番が出力画面での表示順に反映されます。
各行に表示されている編集ボタンを押すと設定画面を表示し、ごみ箱ボタンを押すと削除されます。

マップ出力設定

「マップ出力」を選択すると、次のダイアログが表示されます。



■マップ名：

このマップを呼び出す時に参照する名称です。

■空間：

ドロップダウンリストから表示対象の空間名を選択します。

■レイヤ番号：

表示する階層を指定します。空間設定で「レイヤ数」の値が 1 のときは 0 を、1 以上を設定した場合は表示したい階層（＝レイヤ数）を指定できます。

■背景画像：

表示される二次元マップの背景画像を指定することができます。

固定画像では任意の画像を選択し設定できます。

変数指定では Universe 変数を設定できます。ルール上で Universe 変数に画像ファイルを代入するとこで背景画像を指定できます。

■背景色：

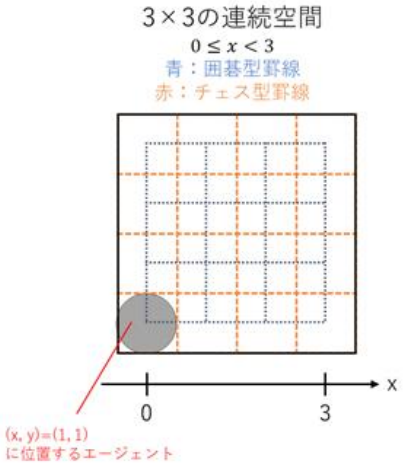
ここで指定した色が、表示される二次元マップの背景色として設定されます。

■原点位置：

原点の位置をコンピュータ等で一般的な左上にするか、数学等で一般的な左下にするか、選択することができます。

■罫線表示：

ここにチェックを入れておくと、表示される二次元マップに縦横の罫線が描画されます。チェス型と囲碁型では罫線の引き方が下図のように異なります。



■X 軸設定 Y 軸設定：

四角格子空間の場合[空間のサイズ]、連続空間モデルの場合[空間のサイズ+1] が最大値になります。

■マップ要素リスト：

マップ上に表示するエージェントや変数をマップ要素として登録します。

画面上に複数の要素を同時に表示したい場合は、それら要素毎に（ex.エージェント 1、エージェント 2、空間変数 1、etc.）マップ要素として追加する必要があります。（つまり要素の数だけ追加リストに並ぶことになります）

ドロップダウンリストからエージェントか空間変数を選択し、「＋」ボタンをクリックすると、任意の既存のマップ要素を設定することが出来ます。

マップ要素リストはドラッグ&ドロップで順番を入れ替えることができ、リストで上にある要素がマップ出力で前面に表示されます。

また、要素名の横に表示される編集ボタンによって設定画面を開き、ごみ箱ボタンによって削除することができます。

［エージェントの追加］



■要素名：

マップ要素に任意の名称をつけます。

■エージェント：

マップ上で動きを見たいエージェントを選択します。

■マーカー：

マップ上の要素にマーカーをどのように表示するかどうかを選択します。

「選択」では円、四角、三角、矢印、×の中から選択します。
「ファイル」では任意の画像を選択し設定できます。
「変数指定」ではエージェントの変数を選択し、ルール上で選択した変数に画像ファイルを指定することでマーカーの画像を設定できます。
■エージェント表示色：
任意の一色を指定する「固定色」と指定した変数の値によって動的に色が変化する「変数指定」のどちらかを選択できます。
■拡大率：
固定値が１のときには、１セルのサイズと同じサイズでマーカーが表示されます。変数指定では動的に表示サイズを変更できます。
■透明度：
固定値では 0～100 の間で透明度を変更できます。変数指定では動的に透明度を変更できます。
■エージェント情報の表示：
「表示する変数」で指定した変数の値が、マップ上のエージェントの上表示されます。少数の表示桁数と文字色も指定することができます。
■エージェント間に線を引く：
出力対象のエージェントがエージェント集合変数を持つ場合は、出力対象エージェントとエージェント集合の要素それぞれとを線で結ぶことができます。
「対象の変数」でエージェント集合を持つ変数を指定してください。「線の種類」「矢印の種類」「線の色」も設定することができます。
線引き対象、線種、矢印種別、色の設定を選択できるようになります。

[空間変数の追加]



■要素名：

このマップ要素に任意の名称をつけます。

■表示色：

空間の変数を出力対象とした場合、出力対象の変数の値によって色を表示することも可能です。

「グラデーション指定」では、指定された変数の範囲に該当する色が表示されます。表示する色に対応する最小の値と色、最大の値と色を指定します。この間にある値に対応する色は、最小値に近いほど、最小値に設定した色に近づき、最大値に近くなるほど最大値に設定した色に近づきます。

「固定色」では、任意の一色を選択します。

「変数指定」では、指定した変数の値によって色が変化します。

■透明度：

変数の表示透明度を指定します。

透明度は 0 から 100 まで指定可能で、値が大きいほど透明となります。

「固定値」では、透明度をテキストボックスで直接設定するか、スクロールバーで設定してください。

「変数指定」では、透明度を変数の値で指定することが可能です。

■変数表示：

「変数値を表示する」にチェックを入れると「対象変数」で指定した変数の値が、空間変数上に表示されます。少数の表示桁数と文字色も指定することができます。

時系列グラフ出力設定

「時系列グラフ」を選択すると、次のダイアログが表示されます。



■グラフ名：

このグラフを呼び出す時に参照する名称です。

■凡例表示：

ここにチェックを入れておくと、グラフの横に凡例が表示されます。

■線をなめらかにする：

ここにチェックを入れておくと、線がなめらかに表示されます。

■背景色：

グラフの背景色を設定します。

■軸ラベル：

グラフを表示する時にその軸のラベルとしてグラフとともに表示されます。

■自動目盛：

自動目盛にチェックを入れることで、表示されるグラフの最小値、最大値、目盛の間隔を自動的に変化させます。自動設定を解除する場合は最小値、最大値、目盛り間隔を入力してください。

■時系列グラフ要素リスト：

グラフに出力する要素が表示されます。

要素リストはドラッグ&ドロップで順番の入れ替えができ、リストの順番がグラフ出力の表示順に反映されます。

編集ボタンをクリックすると、任意の既存の要素を編集することが出来ます。またごみ箱ボタンによって、任意の既存の数値画面出力要素を削除することが出来ます。

「＋」ボタンをクリックすると次の画面が表示されます。



■要素名：

このグラフ要素に任意の名称をつけます。

■出力値：

そのグラフの経緯を見たい要素（変数等）を入力します。

■線の太さ：

グラフの太さを設定します。

■印の大きさ：

大きさを選択するとグラフ上にマーカーが表示されます。

■印の形：

グラフ上のマーカーの形を選択します。

■グラフの色 グラフの色を設定します。

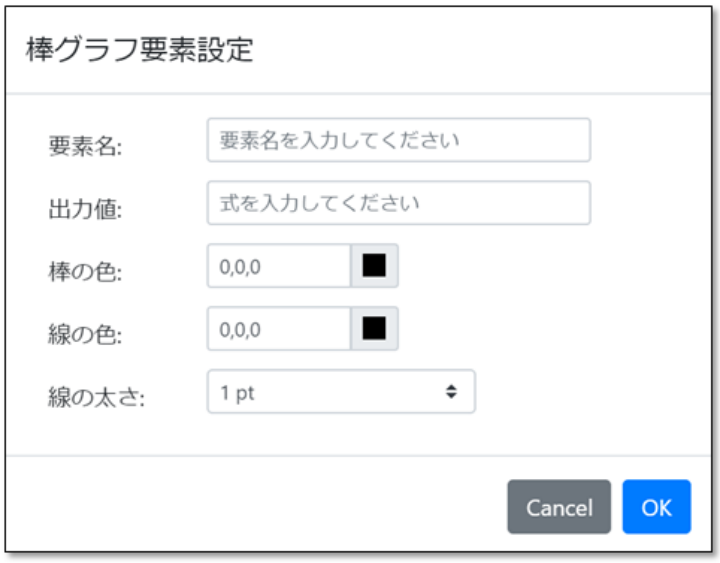
棒グラフ出力設定

「棒グラフ」を選択すると、次のダイアログが表示されます。

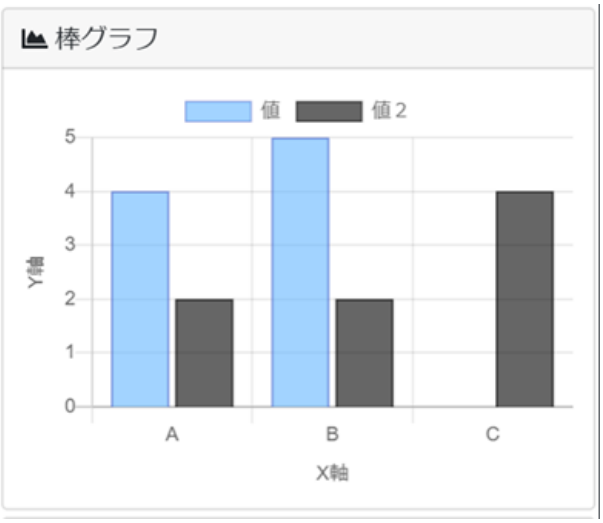


■グラフ名：
このグラフを呼び出す時に参照する名称です。
■凡例表示：
ここにチェックを入れておくと、グラフの横に凡例が表示されます。
■積み上げ表示：
ここにチェックを入れておくと、設定された要素が積み上げられた棒グラフが表示されます。
■棒グラフの方向：
グラフを横棒で表示するか縦棒で表示するかを選択できます。
■背景色：
グラフの背景の色を変更することができます。
■軸ラベル：
グラフを表示する時にその軸のラベルとしてグラフとともに表示されます。
■項目ラベル：
各グラフにつける項目ラベルを指定します。
後述する棒グラフ要素の出力値に合わせて、項目も配列値を指定することで左から（上から）順に表示します。
■自動目盛：
自動設定にチェックを入れることで、表示されるグラフの最小値、最大値、目盛の間隔を自動的に変化させます。自動設定を解除する場合は最小値、最大値、目盛り間隔を入力し、指定してください。
■棒グラフ要素リスト：
グラフに出力する要素が表示されます。
要素リストはドラッグ&ドロップで順番の入れ替えができ、リストの順番がグラフ出力の表示順に反映されます。
編集ボタンをクリックすると、任意の既存の要素を編集することが出来ます。またごみ箱

ボタンによって、任意の既存の数値画面出力要素を削除することが出来ます。
「＋」ボタンをクリックすると次の画面が表示されます。



■要素名：
このグラフ要素に任意の名称をつけます。
■出力値：
そのグラフの経緯を見たい要素（変数等）を入力します。
出力値にはカンマ区切りで配列を指定してください。
例えば、配列を持つ「値」と「値２」を棒グラフ要素リストに追加した場合、次のように表示されます。



■棒の色：

グラフの色を設定します。

■線の色：

グラフの枠線の色を設定します。

■線の太さ：

グラフの枠線の太さを設定します。

円グラフ出力設定

「円グラフ」を選択すると、次のダイアログが表示されます。



■グラフ名：

このグラフを呼び出すときに参照する名称です。

■凡例表示：

ここにチェックを入れておくと、グラフの横に凡例が表示されます。

■背景色：

背景の色を変更することができます。

■円グラフ要素リスト：

グラフに出力する要素が表示されます。

要素リストはドラッグ&ドロップで順番の入れ替えができ、リストの順番がグラフ出力の表示順に反映されます。
編集ボタンをクリックすると、任意の既存の要素を編集することが出来ます。またごみ箱ボタンによって、任意の既存の数値画面出力要素を削除することが出来ます。
「＋」ボタンをクリックすると次の画面が表示されます。



■要素名：
このグラフ要素に任意の名称をつけます。
■出力値：
そのグラフの経緯を見たい要素（変数等）を入力します。
■円の色：
その変数を表す領域の色を変更することができます。

散布図出力設定

「散布図」を選択すると、次のダイアログが表示されます。



■グラフ名：

このグラフを呼び出す時に参照する名称です。

■凡例表示：

ここにチェックを入れておくと、グラフの横に凡例が表示されます。

■背景色：

背景の色を変更することができます。

■軸ラベル：

グラフを表示する時にその軸のラベルとしてグラフとともに表示されます。

■自動目盛：

自動目盛にチェックを入れることで、表示されるグラフの最小値、最大値、目盛の間隔を自動的に変化させます。自動設定を解除する場合は最小値、最大値、目盛り間隔を入力し、指定してください。

■散布図要素リスト：

散布図に出力する要素が表示されます。

要素リストはドラッグ&ドロップで順番の入れ替えができ、リストの順番がグラフ出力の表示順に反映されます。

編集ボタンをクリックすると、任意の既存の要素を編集することが出来ます。またごみ箱

ボタンによって、任意の既存の数値画面出力要素を削除することが出来ます。
「追加」ボタンをクリックすると次の画面が表示されます。



■要素名：
このグラフ要素に任意の名称をつけます。
■プロット位置変数：
表示させるプロットの X の値と Y の値を決める変数を選択してください。ここには、配列変数を指定することで、上記指定されたプロット数だけの対象配列変数内の値をプロットします。
■印の大きさ：
プロットする印の大きさを変更することができます。
■印の色：
プロットする印の色を変更することができます。
■印の形：
プロットする形状を設定することができます。

折れ線グラフ出力設定

「折れ線グラフ」を選択すると、次のダイアログが表示されます。



■グラフ名：

このグラフを呼び出す時に参照する名称です。

■凡例表示：

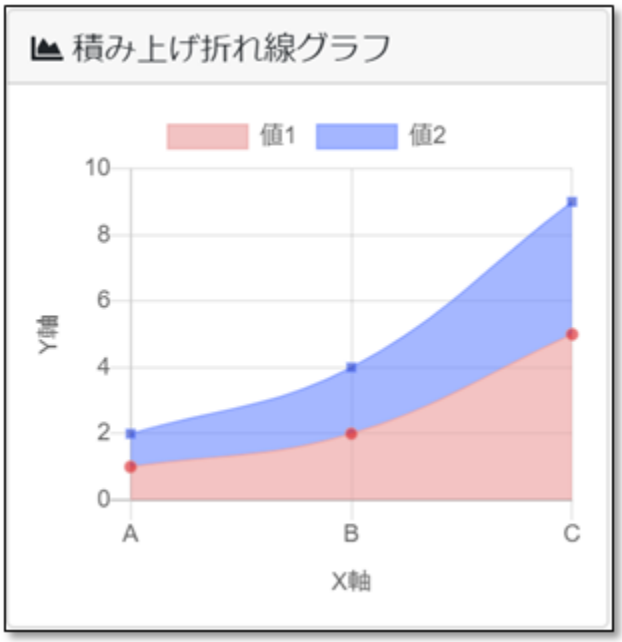
ここにチェックを入れておくと、グラフの横に凡例が表示されます。

■積み上げ表示：

ここにチェックを入れておくと、下図のようにグラフの間が塗りつぶされて表示されます。

■線をなめらかにする：

ここにチェックを入れておくと、線がなめらかに表示されます。



■背景色：

背景の色を変更することができます。

■軸ラベル：

グラフを表示する時にその軸のラベルとしてグラフとともに表示されます。

■自動目盛：

自動目盛にチェックを入れることで、表示されるグラフの最小値、最大値、目盛の間隔を自動的に変化させます。自動設定を解除する場合は最小値、最大値、目盛り間隔を入力してください。

■グラフ要素リスト：

グラフに出力する要素が表示されます。

要素リストはドラッグ&ドロップで順番の入れ替えができ、リストの順番がグラフ出力の表示順に反映されます。

編集ボタンをクリックすると、任意の既存の要素を編集することが出来ます。またごみ箱

ボタンによって、任意の既存の数値画面出力要素を削除することが出来ます。

「＋」ボタンをクリックすると次の画面が表示されます。



■要素名：

このグラフ要素に任意の名称をつけます。

■出力値：

表示させるプロットの X の値と Y の値を決める変数を選択してください。配列の変数を指定することで、複数のプロットを行うことができます。

■線の太さ：

線の太さを変更することができます。

■印の大きさ：

大きさを選択するとグラフ上にマーカーが表示されます。

■印の形：

グラフ上のマーカーの形を選択します。

■グラフの色：

グラフに表示される線の色を選択します。

■背景色：

積み上げグラフの場合のみ有効な設定で、塗りつぶしの色を指定できます。

ヒストグラム出力設定

「ヒストグラム」を選択すると、次のダイアログが表示されます。



■グラフ名：

このグラフを呼び出す時に参照する名称です。

■背景色：

背景の色を変更することができます。

■軸ラベル：

グラフを表示する時にその軸のラベルとしてグラフとともに表示されます。

■区間数：

一つの棒として表す区間の総数を入力してください。

■区間の自動設定：

区画に等分する範囲の最小値と最大値の値が自動で計算されます。指定する場合は区画の最小値と最大値を入力して下さい。

■区間配列：

区間を等分で設定しない場合は区間をカンマ区切りで指定してください。

■自動目盛：

自動目盛にチェックを入れることで、表示されるグラフの最小値、最大値、目盛の間隔を

自動的に変化させます。自動設定を解除する場合は最小値、最大値、目盛り間隔を入力し、指定してください。

■線の色：

線の色を変更することができます。

■出力値：

ヒストグラムに反映するデータとなる配列変数を選択してください。

※補足

ヒストグラムで表示する対象の変数は配列変数となります。例えば、Universe 以下に hist_data という変数を定義し、hist_data (0)～hist_data (99)までの変数にそれぞれ何らかの値を入れます。これらのデータをもとに度数分布を表示します。

設定方法は下記の通りです。

区間数：10 (hist_data (0)～hist_data (99)に格納してある値の最大値と最小値の区間を 10 分割します)

出力値：Universe.hist_data

10 分割したそれぞれの範囲内でのデータ数を棒グラフで表現します。

値出力設定

「値出力」を選択すると、次のダイアログが表示されます。



■出力名：

値出力エリアに表示されるタイトルです。

■値出力要素リスト：

値を表示する変数が表示されます。

編集ボタンをクリックすると、任意の既存の要素を編集することが出来ます。

要素リストはドラッグ&ドロップで順番の入れ替えができ、リストの順番が値出力の表示順に反映されます。

またごみ箱ボタンによって、任意の既存の数値画面出力要素を削除することが出来ます。

値出力要素リストの「＋」ボタンをクリックすると、次の画面が表示されます。



■要素名：

出力要素に任意の名称をつけます。

■出力値：

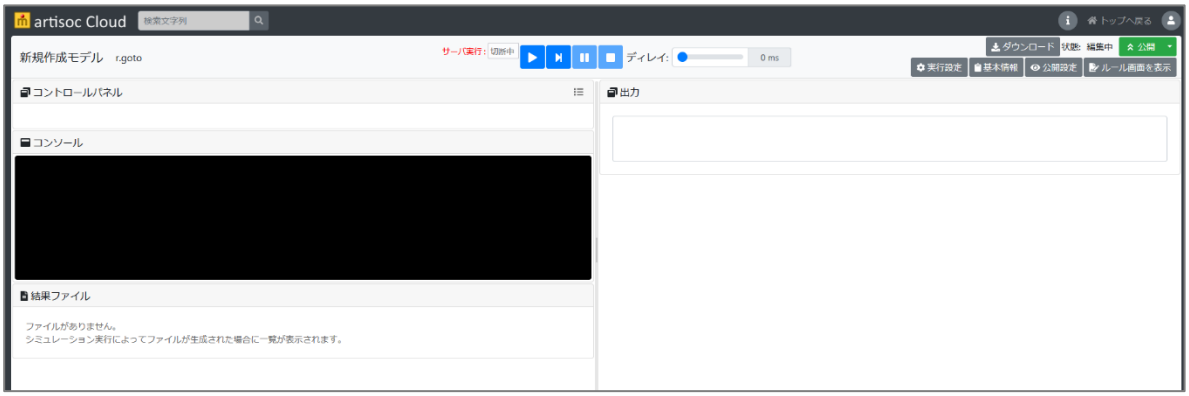
出力したい変数や計算式を入力します。

■小数表示：

小数点以下何桁で表示するかを設定します。

コントロールパネル設定

コントロールパネルを設定するにはコントロールパネルエリアにカーソルを合わせると表示される　を選択すると、コントロールパネル設定ダイアログが表示されます。





■入力設定項目リスト：

設定済みのマップもしくはグラフ名が表示されます。各行に表示されている編集ボタンを押すと設定画面を表示し、ごみ箱ボタンを押すと削除されます。

「＋」ボタンを選択するとコントロールパネルに表示させる設定項目を追加できます。

追加した項目はドラッグ&ドロップで順番の入れ替えができ、リストの順番がコントロー

ルパネルの表示順に反映されます。
追加や編集では次のような設定ダイアログが開きます。



設定ダイアログが開いたら ⇕ をクリックします。選択可能な項目が表示されるので設定する項目を選択します。

academic ライセンスでは、初期値の変更、シミュレーション実行中の変数値の変更が可能です。Standard ライセンス、オンプレミス版 artisoc Cloud では初期値のみ変更可能です。

ボタン

ボタンを押すことでボタンに設定した値を対象変数に代入します。

■コントロール名:

ボタンに表示される名前です。分かりやすい任意の名前を付けてください。

■設定対象：

コントロールパネルで操作したい変数を選択します。ただし、Universe の直下にある変数しか選択できません。

■値の型：

設定対象の変数に代入する値を整数、実数、文字列の中から選択してください。

■ON の値：

ボタンが ON になったとき変数に代入する値です。

トグルボタン

　ボタンを押すことで設定した値を対象変数に代入します。ON の時は ON の値を、OFF の時は OFF の値を代入するところが、通常のボタンと異なります。

■コントロール名:
ボタンに表示される名前です。分かりやすい任意の名前を付けてください。
■設定対象：
コントロールパネルで操作したい変数を選択します。ただし、Universe の直下にある変数しか選択できません。
■値の型：
設定対象の変数に代入する値を整数、実数、文字列の中から選択してください。
■ON の値：
ボタンが ON になったとき変数に代入する値です。　■OFF の値：
ボタンが OFF になったとき変数に代入する値です。

スライドバー

シミュレーション実行中にスライドバーを操作した際は、スライドバー操作を終えた瞬間のステップで設定値が対象変数に代入されます。

■コントロール名:
押すと設定内容を実行するボタンに表示される名前です。分かりやすい任意の名前を付けてください。
■設定対象：
コントロールパネルで操作したい変数を選択します。ただし、Universe の直下にある変数しか選択できません。
■値の型：
設定対象の変数に代入する値を整数、実数、文字列の中から選択してください。
■初期値：
モデルの出力画面を開いたときのスライドバーの位置を設定します。
■範囲：
左に変数に代入したい値の最小値を、右に最大値を入力してください。
■目盛り間隔：
ここに入力した値が断続的な変化の最小間隔になります。

直接入力

入力した値を対象変数に代入します。

■コントロール名:

押すと設定内容を実行するボタンに表示される名前です。分かりやすい任意の名前を付けてください。

■設定対象：

コントロールパネルで操作したい変数を選択します。ただし、Universe の直下にある変数しか選択できません。

■値の型：

設定対象の変数に代入する値を整数、実数、文字列の中から選択してください。

リスト

選択肢をリストとして設定することが出来ます。選択した値を対象変数に代入します。

■コントロール名:
押すと設定内容を実行するボタンに表示される名前です。分かりやすい任意の名前を付けてください。
■設定対象：
コントロールパネルで操作したい変数を選択します。ただし、Universe の直下にある変数しか選択できません。
■値の型：
設定対象の変数に代入する値を整数、実数、文字列の中から選択してください。
■リスト項目：
選択候補をカンマ区切りで入力してください。

ルール

ルールを設定し、ボタンを押すことで実行することが出来ます。

■コントロール名:
押すとルール記述を実行するボタンに表示される名前です。分かりやすい任意の名前を付けてください。
■ルール：
ボタンを押したときに実行したいルールを記述します。

実行設定

シミュレーション設定



■最大ステップ数：

途中でユーザがシミュレーションを停止しない限り、この数値の回数だけステップを実行します。

■最大実行時間：

途中でユーザがシミュレーションを停止しない限り、この数値の時間が経過するまでステップを実行します。

■終了条件式：

途中でユーザがシミュレーションを停止しない限り、ここに入力された条件式が成立するまでステップを実行します。

■実行ウェイト：

1 ステップ終了した段階で、この数値の時間だけ、処理をウェイトします。シミュレーション速度を調整したい場合に、適当な数値を入力してください。

なおこの項目はディレイ設定と同義です。

■実行中に変更可能な実行ウェイト最大値：

ツールバー上でディレイ設定スライダーバーにて変更可能な実行ウェイトの最大値を指定します。

モデルを共有する

モデルのダウンロード・アップロード

モデルは.json 形式でダウンロードし、また同様の形式でアップロードすることができます。

■ダウンロード

■アップロード

画面右上のユーザボタンを押すと、「モデルのアップロード」というメニューを選ぶことができます。



.json 形式のファイルをドラッグ・アンド・ドロップ、またはフォルダから選択するとモデルが新規モデルとして読み込まれます。

モデルの保存・公開・削除

■ モデルの状態

モデルには下記 3 つの状態が存在します。現在の状態は出力画面右上の表示で確認できます。

1.編集中

ユーザが編集中のモデルです。他のユーザは閲覧できません。

2.公開中

誰でも閲覧できるように公開されたモデルです。ただし、限定公開設定にしている場合は URL を知っている人のみが閲覧できます。また、公開中のモデルは他のユーザが継承して新規作成することができます。

3.公開停止中

利用規約に違反しているモデルは、運営によって公開停止状態になる場合があります。公開停止中のモデルは編集を加えることで再び公開することができます。

モデルによっては上記の状態が 2 つ以上存在することがあります。
例えばモデル「A」を作成・公開したとします。この時アクションボタンから「編集開始」を選択すると、「A」に変更を加えることができます（詳しくは後述）。
この状態で「A」を編集し「B」という内容に変わったとします。この時モデルは、公開中の「A」と編集中の「B」、2 つの状態が存在することになります。こういった 2 つ以上の状態が存在するモデルでは、状態右横の▼を選択すると、状態の違うモデルに表示が切り替わります。つまりモデルの「表」と「裏」を入れ替えることができる、というわけです。

■アクションボタン

　アクションボタンでは、公開、一時保存、保存して公開、などを選択できます。
ボタンを押すと表示中のアクションが実行され、▼を選択するとその他のアクションを選択できます。



アクション名	アクション
	「裏」にモデルが存在しないとき
公開	編集中モデルを公開する。
保存	編集中モデルを保存する。
保存して公開	編集中モデルを保存して公開する。
編集の取り消し(破棄)	編集中モデルを前回保存した状態まで戻す。
公開の取り下げ	モデルの状態を公開中から編集中に移行する。他のユーザはモデルを閲覧できなくなる。
編集開始	公開中モデルで表示される項目。公開中モデルをコピーして編集中モデルを作成し、画面遷移する。編集中モデルが改めて公開されるまで、公開中モデルは「裏」として存在し続ける。
継承して新規作成	モデルをコピーして編集中モデルを新規に作成し、画面遷移する。作成したモデルは継承元とは別のモデルとして存在する。
	「裏」に公開中モデルが存在するとき
公開	編集中モデルを公開する。モデル公開後、「裏」の公開中モデルは上書きされる。
編集中モデルの削除	編集中モデルを削除し、公開中モデルに画面遷移する。
	「裏」に編集中モデルが存在するとき
公開の取り下げ	公開中モデルを削除し、編集中モデルに画面遷移する。

アクションの種類は以下の表のとおりです。

アクション毎の状態遷移に関しては、以下の表を参照してください。

■公開設定

　のボタンから、モデルを一般公開にするか限定公開にするか、選択できます。
デフォルトでは一般公開設定になっており、公開するとモデルがトップページやモデルリストに表示され、誰でもモデルを検索できるようになります。
限定公開とは、URL を知っている人のみがアクセス可能なことを意味します。限定公開モデルはトップページやモデルリストに表示されず、artisoc Cloud の検索バーから検索することもできません。

アクション	状態番号	A	B	C	D
	表	編集中	編集中	公開中	公開停止
	裏		公開中		
公開		C	C	-	-
保存				-	A
保存して公開		C	C	-	C
編集の取り消し(破棄)				-	
編集中モデルの削除		-	C	-	-
公開の取り下げ		-	-	A	-
編集開始		-	-	B	-
継承して新規作成		A	A	A	A

ルール文法

概要

本章では、artisoc Cloud モデルのルール記述についての仕様を説明します。 モデルのルールは、基本的に Python 言語（ver.3 系）を用いて記述します。

Python の基礎文法

artisoc Cloud では、プログラミング言語 Python を用いてモデルのルールを記述します。本節では、artisoc Cloud のルール記述に最低限必要な Python の基礎的な文法を説明します。詳しくは Python の公式ドキュメントを参照してください。なお、一部に artisoc Cloud 特有の内容も含まれていますが、Python 使用経験のある方は読み飛ばしても問題ありません。

変数・代入・演算

変数とは、値を保存しておく箱のようなものです。たとえば以下のように記号「=」を用いて記述することで、変数 a を作成しそこに値 3 を保存することができます。これを代入と呼びます。

```
a = 3
```

ここで記号「=」は右辺の値を左辺に代入する操作を表します。数学とは異なり、両辺が等しいことを示す記号ではないことに注意してください。

変数名は基本的には自由につけることができますが、以下のようなルールがあります。

- 大文字と小文字は区別される。
- 変数名に空白を含んではいけない。

- 記号は一部を除き使えない。たとえば「-」（ハイフン）は後述するように引き算の演算子として解釈されるため変数名には使えない。一方で「_」（アンダースコア）は使用可能で、変数名内で単語を区切るためによく使われる。
- 変数の 1 文字目には数字を使えない。
- 以下の文字列は Python で特別な意味を持つため、変数名として使えない。これを Python のキーワード（あるいは予約語）という。
  - and, as, assert, break, class, continue, def, del, elif, else, except, false, finally, for, from, global, if, import, in, is, lambda, None, nonlocal, not, or, pass, raise
- 既に存在する関数名などを変数名に使ってしまうと、その関数が使えなくなってしまうため注意が必要である。

数値や変数を用いて、演算を行うことができます。たとえば以下のように記述することで、変数 b に数値 7 が代入されます。

```
a = 3
b = a + 4
```

Python の演算で用いることのできる代表的な演算子を以下に示します。

演算子	説明
=	右辺に左辺を代入する 例：変数 a に 3 を代入する a = 3
+	足し算 例:2 に 3 を足した値を a に代入する a = 2 + 3

-	引き算 例：3 から 2 を引いた値を a に代入する a = 3 - 2
*	掛け算 例：3 に 2 を掛けた値を a に代入する a = 3 * 2
/	割り算 例：3 を 2 で割った値(1.5)を a に代入する a = 3 / 2
**	累乗 例：3 の 2 乗の値を a に代入する a = 3 ** 2
%	割り算の余り 例：7 を 3 で割った余りを a に代入する a = 7 % 3
//	割り算の商 例：7 を 3 で割った商を a に代入する a = 7 // 3

コメントアウト

プログラムの行内で、記号「#」以降には何を記述してもプログラムの動作に影響を与えません。これをコメントアウトと呼び、主にプログラムの説明によく用いられます。

```
a = 2 + 3 # 2+3 の計算結果を変数 a に代入
```

```
# 変数 b に a+5 の計算結果を代入
b = a + 5
```

関数

関数とは一連の処理をまとめたものです。artisoc Cloud にはモデル作りに役立つ多くの関数が用意されています。

たとえば round()は実数型の数値を四捨五入する処理を表します。たとえば以下のように使用することで、4.7 を四捨五入した結果（5）を変数に代入することができます。

```
a = round(4.7) # 変数 a に整数 5 を代入
```

ここでの数値 4.7 を関数の引数、数値 5 を関数の返り値（または戻り値）と呼びます。

関数には返り値を持たないものもあります。たとえば print 関数は、引数の値をコンソールに出力する機能を持ちます。変数の値の確認などによく用いられます。

```
a = 2 + 3
print(a) # コンソールに「10」が出力される
```

if 文（条件分岐文）

条件に応じて処理を分岐させる文です。以下のように書きます。

```
if （条件式）:
    （条件式が真の場合の処理）
else:
    （条件式が偽の場合の処理）
```

処理を書く部分は、タブを用いて 1 段字下げをします。これをインデントといい、インデントによって処理の範囲が定義されます。else 以下は省略することもできます。

たとえば以下のように用います。

```
if a < 5:
   print(“a は 5 より小さい”)
else:
   print(“a は 5 以上”)
```

あるいは以下のように、複数の条件式を用いた条件分岐を記述することもできます。

```
if （条件式 1）:
     （条件式 1 が真の場合の処理）
elif （条件式 2）:
     （条件式 1 が偽で、条件式 2 が真の場合の処理）
else:
     （条件式 1 も条件式 2 も偽の場合の処理）
```

for 文（繰り返し文）

繰り返しの処理を表す文です。

たとえば以下のように書くと、0 から 9 までの数値を順にコンソールに出力します。

```
for i in range(10):
   print(i)
```

range(10)によって 0 から 9 までの数値の集合を生成し、それら 1 つ 1 つを順に i に代入し処理を行うようなイメージです。

artisoc Cloud においては、エージェント集合の 1 つ 1 つのエージェントに対し操作を行う際によく用います。たとえば以下のように使います。

```
people = make_agtset(type=Universe.hiroba.hito) # hito エージェントの集合 people を生成
for person in people:
    # people の要素 1 つ 1 つ（person）について、以下の処理を行う
    # もし x 座標が 25 未満なら、25 にする
    if person.x < 25:
        person.x = 25
```

ここで、make_agtset はエージェント集合を生成する関数です。エージェント集合 people の一つひとつの要素（person）を for 文を用いて取り出し、person の属性に応じた処理を if 文を用いて行っています。

このように、for 文の中で if 文を用いるなど、for 文や if 文は入れ子構造をとることができます。

変数の型

前節では変数について数値計算を例に説明しましたが、変数には数値以外にも様々な種類の値を代入することができます。これを変数の型といいます。以下では、artisoc Cloud を扱うにあたって最低限必要と思われる変数型を紹介します。

実数型

実数を扱う変数型です。

コンピュータの性質により計算に微妙な誤差を生じる場合がありますので、場合によっては注意が必要です。

下記の例では変数 a に 0.1+0.2 の計算結果を実数型として代入していますが、計算結果は 0.3 でなく 0.30000000000000004 となり、微妙な誤差を生じます。

```
a = 0.1 + 0.2
print(a) # --> 0.30000000000000004
```

整数型

整数を扱う変数型です。実数型とは異なり、計算に誤差は生じません。なお、整数型同士の割り算の結果は実数型になります。

```
a = 1 + 2
print(a) # --> 3 （整数型）
# a には整数型の数値 3 が代入される

b = 3 / 2
print(b) # --> 1.5 （実数型）
```

文字列型

数値ではなく文字列を扱う型です。多くの場合、「'」または「"」で囲まれた文字の並びで表現されます。

```
a = “artisoc Cloud”
print(a) # --> コンソールに「artisoc Cloud」と出力される
```

エージェント種別型

artisoc Cloud 特有の変数型で、エージェントの種別を表します。

主に create_agt 関数を用いてエージェントを生成するときに用います。下記の例ではエージェント種別型変数 Universe.hiroba.hito を用いて、その種別のエージェントを生成しています。

```
create_agt(Universe.hiroba.hito) # hito エージェントを生成
```

エージェント型

artisoc Cloud 特有の変数型で、具体的な 1 つのエージェントを表します。エージェントを操作する際などに使用します。

```
one = create_agt(Universe.hiroba.hito) # hito エージェントを生成し、変数 one に代入
one.x = 25 # エージェント one の x 座標に 25 を代入
```

セット型

複数の変数をひとまとめにした集合を表す変数型です。集合の要素となるそれぞれの変数の型は何でもかまいません。また、要素の順番は定義されません。

artisoc Cloud においては、関数を用いてエージェントの集合を生成するときによく使います。

```
# 全ての hito 種別エージェントの集合を変数 people に代入
people = make_agtset(agttype=Universe.hiroba.hito)
```

リスト型

セット型と同じく複数の変数をひとまとめにした集合を表す変数型ですが、要素に順序が定義されます。記号[]で位置を指定することで、その位置の要素を取得することができます。

```
# 全ての hito 種別エージェントの集合（セット型）を変数 people に代入
people = make_agtset(agttype=Universe.hiroba.hito)

# セット型変数 people をリストに変換し、属性 tall の昇順に並べる
people = make_agtlist(people, key="tall")

# 最も小さい人（リストの 0 番目）を取得する
smallest_person = people[0]
```

モデルツリーの構成

モデルツリーはモデルの概形を規定するもので、以下の要素からなります。

- **Universe**
  - モデルの全体を表す要素。モデル全体の特徴を規定するルールと変数を持つ。モデルにただ一つ存在する。
- **空間**
  - モデル内の空間を表す要素。Universe の直下に作成することができる。
- **エージェント種別**
  - モデル内で自立的に行動するエージェントを表す要素。エージェントの特徴を規定するルールと変数を持つ。Universe の直下または空間の直下に作成することができる。空間の直下に作成した場合は空間内で行動するエージェントを表し、位置座標など空間にまつわる変数を持つ。
- **Universe 変数**

- モデル全体の属性や状態を表す変数で、Universe の直下に作成することができる。

- **空間変数**
  - 空間の属性や状態を表す変数。配列型の変数であり、空間の座標やレイヤごとに値を持つ。空間の直下に作成することができる。
- **エージェント変数**
  - エージェントの属性や状態を表す変数。エージェント種別の直下に作成することができる。

ルールの全体構成

モデルのルールは、Universe のルールとエージェントのルールの 2 種類があります。

Universe のルールはモデル全体の挙動を決定するもので、以下のような構成となっています。

- **univ_init**
  - シミュレーション開始時に一度だけ実行されるルール
- **univ_step_begin**
  - シミュレーションの各ステップの最初に実行されるルール
- **univ_step_end**
  - シミュレーションの各ステップの最後に実行されるルール
- **univ_finish**
  - シミュレーション終了時に一度だけ実行されるルール

エージェントのルールはエージェント種別ごとに定義され、各エージェントの行動を決定するものです。以下のような構成となっています。

- **agt_init**
  - エージェント生成時に一度だけ実行されるルール
- **agt_step**

– シミュレーションの各ステップで実行されるルール

シミュレーション全体を通してのルールの実行順序は以下のようになります。

まずシミュレーション開始時に univ_init が実行されます。各タイムステップにおいて univ_step_begin →各エージェントの agt_step →univ_step_end の順でルールが実行されます。シミュレーション終了条件が満たされたとき、univ_finish が実行され、シミュレーションが終了します。

実際のルールエディタは以下のようになっています。ルールを記述する際は、タブを用いてインデント内に記述することに注意してください。また、何もルールを記述しない場合は「pass」と記述してください。

Universe のルールエディタ

```
def univ_init():
    pass

def univ_step_begin():
    pass

def univ_step_end():
    pass

def univ_finish():
    pass
```

エージェントのルールエディタ

```
def agt_init(self):
    pass
```

```
def agt_step(self):
    pass
```

変数の呼び出し

ツリー要素の呼び出し

ツリーの要素のうち、以下のものは変数として直接呼び出すことができます。

- Universe 変数
- 空間変数
- エージェント種別

■Universe 変数の呼び出し

たとえば、Universe 変数として hensuu を定義したとします。hensuu を呼び出すためには、ルール内に以下のように記述します。

```
Universe.hensuu
```

hensuu に数値 3 を代入するには、以下のように記述します。

```
Universe.hensuu = 3
```

■空間変数の呼び出し

Universe の下に空間 hiroba を定義し、hiroba の下に空間変数 takasa を定義したとします。takasa を呼び出すためには、ルール内に以下のように記述します。

```
Universe.hiroba.takasa
```

空間変数は 3 次元配列（リスト）となっており、0 番目の要素に x 座標、1 番目の要素に y 座標、2 番目の要素にレイヤ番号を持ちます。レイヤ番号 0 の座標(x,y)=(2,3)に数値 5 を代入するには、以下のように記述します。

```
Universe.hiroba.takasa[2, 3, 0] = 5
```

■エージェント種別の呼び出し

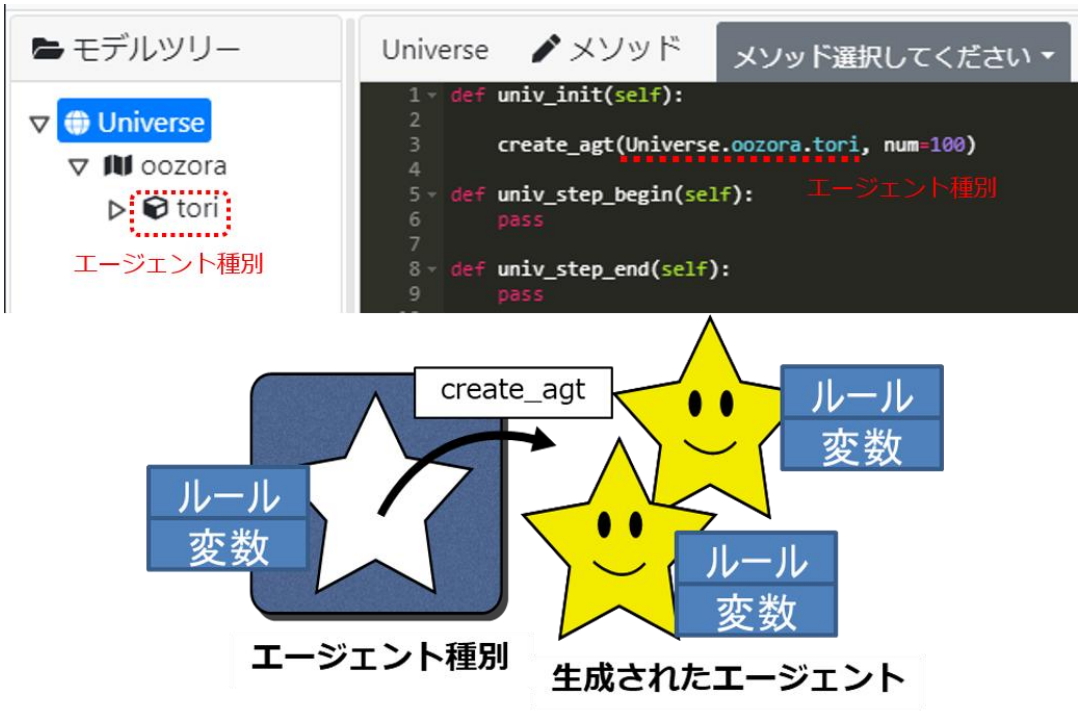
Universe の下に空間 hiroba を定義し、hiroba の下にエージェント種別 hito を定義したとします。hito を呼び出すためには、ルール内に以下のように記述します。

```
Universe.hiroba.hito
```

エージェント種別 hito からエージェントを 1 つ生成するためには、create_agt 関数を用いて以下のように記述します。

```
create_agt(Universe.hiroba.hito)
```

エージェント種別はエージェントを作成するためのひな形のようなものです。ルールや変数はエージェント種別に設定しますが、実際にルールや変数を持つのはエージェント種別から生成された個別のエージェントとなります。



エージェント種別に値を代入することは通常ありません。

エージェント変数の呼び出し

エージェントは変数を持ちます。全てのエージェントはデフォルトで属性 id を、さらに空間エージェントはデフォルトで属性 x, y, layer, direction を持ちます。さらに、属性は自由に追加することができます。

ルールからエージェント変数を呼び出すには 2 つ方法があります。

1 つ目の方法は、エージェントのルールエディタから self を用いてアクセスする方法です。たとえば属性 x を呼び出すためには、該当エージェントのルールエディタ内で以下のように記述します。

```
self.x
```

ここで self は自分自身のエージェントを意味するエージェント型の変数です。

2 つ目は、エージェント型の変数から呼び出す方法です。たとえば以下のように記述します。

```
taro = create_agt(Universe.hiroba.hito) # hito エージェントを生成し変数 taro に代入
taro.x = 3  # taro の x 属性に 3 を代入
```

ここで、1 行目では hito エージェントを 1 つ生成し、変数 taro に代入しています。つまりこれ以降、taro は具体的な 1 つの hito エージェントを表します。

2 行目で taro.x と記述することで、taro の持つ x 属性を呼び出すことができます。

artisoc Cloud の関数種別

関数とは一連の処理をまとめたものです。

artisoc Cloud で使える関数には大きく分けて以下の種類があります。

- **Python 組み込み関数**
  - Python にもともと組み込まれており、最初から使用可能な関数
- **artisoc Cloud 組み込み関数**
  - artisoc Cloud にもともと組み込まれており、最初から使用可能な関数
- **ユーザ定義関数**
  - ユーザが作成する関数

以降、このそれぞれについて説明します。

Python 組み込み関数

Python 組み込み関数は、Python の機能としてもともと組み込まれている関数です。ルールのどこからでも、関数名のみで呼び出すことができます。

たとえば round()は実数型の引数を四捨五入して整数を返す関数で、以下のように使用します。

```
a = round(4.7)  # 変数 a に整数 5 を代入
```

Python 組み込み関数は基本的に、後述する artisoc Cloud 組み込み関数のグローバル関数と同じように使用することができます。

Python 組み込み関数についての詳細は、Python の公式ドキュメントを参照してください。

artisoc Cloud 組み込み関数

artisoc Cloud 組み込み関数は、MAS モデルを作成するために便利な機能として、artisoc Cloud にもともと組み込まれており最初から使用可能な関数です。大まかな機能別に分類すると、以下のようなものがあります。

- 数値計算
- 文字列操作
- データ型変換
- エージェント操作（生成・削除）
- エージェント操作（その他）
- 空間エージェント関数（移動）
- 空間エージェント関数（空間操作）
- 空間エージェント関数（エージェント集合生成）
- エージェント集合操作（基本操作）

- エージェント集合操作（生成）
- エージェント集合操作（配置）
- エージェント集合操作（その他）
- 空間操作
- ファイル入出力
- その他

さらに、これらの関数は主に以下のように分類されます。

- **グローバル関数**
  - ルールのどこからでも関数名だけで呼び出せる関数
- **エージェント関数**
  - エージェントが持つ関数であり、エージェント経由で呼び出す関数

多くはグローバル関数ですが、以下の 3 分類の関数はエージェント関数です。これらは空間直下のエージェントのみについて使用することができます。

- 空間エージェント関数（移動）
- 空間エージェント関数（空間操作）
- 空間エージェント関数（エージェント集合生成）

以降、グローバル関数とエージェント関数それぞれの使い方について説明します。なお、各関数の詳細は関数仕様書をご参照ください。

■ グローバル関数の使い方

グローバル関数はルールのどこからでも関数名だけで呼び出すことができます。

たとえば、エージェントを生成する create_agt という関数は引数にエージェント種別をとり、ルールの好きなところに以下のように記述します。

```
create_agt(Universe.hiroba.hito)  # hito エージェントを生成
```

■エージェント関数の使い方

エージェント関数はエージェント自身の行動や操作を表す関数です。

この関数を使うためには、エージェントのルールエディタ内で self を用います。

たとえば、空間エージェントが持つエージェント関数 forward()を用いるためには、エージェントのルールエディタ内に以下のように記述します。

```
self.forward(1) # 前方に 1 進む
```

ユーザ定義関数

関数はユーザ自身で定義することもでき、これをユーザ定義関数と呼びます。

ユーザ定義関数には、Universe のユーザ定義関数とエージェントのユーザ定義関数があります。

基本的に、Universe のユーザ定義関数はモデル全体に関わる処理について定義し、エージェントのユーザ定義関数はエージェントの行動について定義します。

■Universe のユーザ定義関数

Universe のユーザ定義関数を作成するためには、Universe のルールエディタの末尾に記述します。

たとえば、エージェント hito を生成し指定の座標に置くという関数 put_hito を作るためには、以下のように記述します。

```
def univ_finish(self):
    ...
```

```
def put_hito(x, y):
    h = create_agt(Universe.hiroba.hito) # hito エージェントの生成
    h.x = x # x 座標の指定
    h.y = y # y 座標の指定
```

この関数を用いるためには、ルールの好きなところで以下のように記述します。

```
Universe.put_hito(2, 3) # hito エージェントを生成し、座標(2,3)に置く
```

■エージェントのユーザ定義関数

エージェントのユーザ定義関数を作成するためには、エージェントのルールエディタの末尾に記述します。

たとえば、エージェントの向いている正面前方（距離 dist）の一定範囲（range）にいる特定のエージェント種別（type）の数を調べる関数 watch_ahead を作るには、エージェントのルールエディタ末尾に以下のように記述します。

```
def agt_step(self):
   ...

def watch_ahead(self, dist, range, type):

   self.forward(dist) # 前方に進む

   # 周囲のエージェントを neighbor に格納し、数を数える
   neighbor = self.make_agtset_around_own(range, False, agttype=type)
   n = count_agtset(neighbor)
```

```
    # もとの位置に戻る
    self.forward(-dist)

    return n
```

この関数を用いるためには、エージェントのルールエディタ内で以下のように記述します。

```
# 前方距離 2, 範囲 1 にいる hito エージェントを数え、変数 n に格納
n = self.watch_ahead(2, 1, Universe.hiroba.hito)
```

モジュール・ライブラリ

モジュールとはプログラムを記述する際に有用となる関数などをまとめたものです。artisoc Cloud 内で使用できるライブラリに関しては、「artisoc Cloud で利用できる python ライブラリ」をご参照ください。

モジュールを使用するためには、ルールエディタ内で import 文を用いてモジュールを読み込みます。

たとえば math は様々な数学の計算を行うためのライブラリです。univ_init()内で math を利用するためには、以下のようにします。

```
def univ_init():
    import math # math モジュールをインポート
    a = math.asinh(1) # math モジュールの逆双曲線正弦関数を利用
    print(a) # --> 0.8813735870195429
```