

gumonji/Q の使い方

1. gumonji/Q とは

1.1. gumonji/Q の紹介

gumonji/Q では、gumonji 上のキャラクターや動物にシナリオを割り当てることで、エージェントとして自動的に動かすことが可能になります。

エージェントは、与えられたシナリオに従って自動的に行動します。つまり、シナリオを与えるということは、キャラクターや動物に命を吹き込むということです。キャラクターに雑草を刈るシナリオを与えてあげれば、自動的に草刈をしてくれるようになるでしょう。また、動物に言葉に反応して動作するシナリオを与えてあげれば、自分の指示に答えて動いてくれるようになるでしょう。

オリジナルのシナリオを作って、あなたの世界そして友達の世界のキャラクターや動物に与えてあげることで、あなただけの世界の住人をつくってみてください。

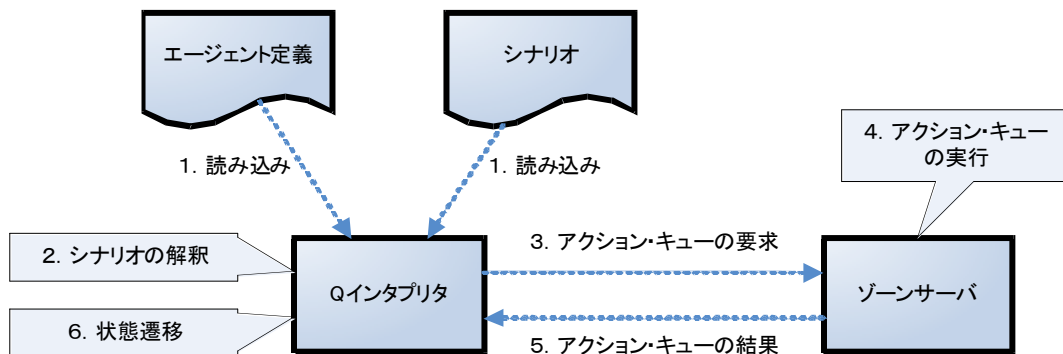
1.2. gumonji/Q の動作概要

シナリオは、ある事柄を観測するとそれに応じた動作を行って次の状態に移るという、状態遷移機械モデルで記述します。観測はキューとして、動作はアクションとして指定します。

エージェントは、状態に応じて1つ以上の事柄を観測し、その中のいずれかの条件が満たされるとその条件に応じた動作を行い、次の状態に移行するという行動を繰り返します。

シナリオの解釈は Q インタプリタで行い、観測や動作の実行はゾーンサーバ上で行います。ゾーンサーバと Q インタプリタの間での処理の流れを図 1 に示します。

1. Q インタプリタでエージェント定義とシナリオを読み込む
2. Q インタプリタでシナリオを解釈し、次の行動を決定する
3. Q インタプリタからゾーンサーバにアクション・キューの実行を依頼する
4. ゾーンサーバで、依頼されたアクション・キューを実行する
5. ゾーンサーバから Q インタプリタにアクション・キューの実行結果を送る
6. Q インタプリタでアクション・キューの実行結果に応じてエージェントの状態を更新する
7. 終了状態でなければ 2 に戻る



(図 1) シナリオ実行の流れ

2. gumonji/Q の実行方法

2.1. JAVA ランタイムのインストール

以下の手順で、JAVA ランタイムをインストールしてください。すでにインストールされている場合は、次に進んでください。

1. <http://java.com> にアクセスして、最新の JAVA ランタイムをダウンロード
2. JAVA ランタイムをインストール

2.1. Q の実行許可

以下の手順で、ゾーンサーバで Q シナリオの実行を許可するように設定してください。

1. gumonji ゾーンサーバの“設定”ボタンを押す
2. “Q の設定”タブを選択する
3. “Q シナリオの実行を許可する”をチェックする
4. 他のユーザーにも実行を許可する場合、“ログイン・エディット可能なユーザーにも実行を許可する”をチェックする
5. “OK”を押す

2.2. Q 処理系の起動

以下の手順で、Q 処理系を起動し、シナリオの実行に必要なライブラリとシナリオを読み込んでください。

1. **自分のゾーンで実行する場合**：ゾーンを起動して “runManager.bat” もしくは “runManager_auto.bat” を実行
友達のゾーンで実行する場合：ゾーンに 3D ブラウザでログインして “runManager_client.bat” もしくは “runManager_client_auto.bat” を実行

※ “runManager_auto.bat”, “runManager_client_auto.bat” を実行すると、2.3 で “(start)” を入力しなくても自動でシナリオの実行を開始します。

※ 友達のゾーンで実行する場合、そのゾーンの設定で他の人にもシナリオの実行が許可されている必要があります。(2.1.を参照)

コマンドプロンプトと Q ウィンドウの 2 つのウィンドウが開きますが、使用するのは Q ウィンドウのみです。コマンドプロンプトにはログやエラーメッセージなどが表示されます。

“scenario/main.q” に、シナリオとエージェントの定義を読み込んで実行を開始するまでのスクリプトが書かれています。自分で定義したファイルを追加で読み込みたいときは、適宜書き換えてください。

例えば、scenario フォルダ内に myscenario.q というファイルでシナリオを定義したときは、以下の記述を追加してください。

```
(load "../scenario/myscenario.q")
```

※カレントディレクトリは、“qv2-src” となっているので注意

2.3. シナリオの実行開始

以下の手順で、シナリオの実行を開始してください。

1. シナリオの実行開始 (Q ウィンドウに以下のコマンドを入力)

```
(start)
```

※2.2.Q 処理系の起動で、“runManager.bat”の代わりに“runManager_auto.bat”を実行すると、“(start)”を入力しなくても自動でシナリオの実行を開始します。

2.4. シナリオの実行停止

Q ウィンドウを閉じることで、シナリオの実行を終了します。

2.5. gumonji/Q の終了

ゾーンサーバを停止することで、gumonji/Q の実行を終了します。

エージェントの位置や所持アイテムなどの情報は、ゾーンのデータの中に保存されます。

※Q ウィンドウ上で実行可能なコマンドの詳細については、gumonji/Q アクション・キューリファレンスの、“4.Q のコマンド”を参照してください。

3. シナリオの書き方

サンプルシナリオのファイルに追加で記述するか、別のファイルにシナリオを記述して Q ウィンドウで読み込むことで、自由にシナリオを作ることができます。拡張子が“.q”や“.scm”のファイルは、テキスト形式のファイルです。適当なテキストエディタで閲覧・編集することが可能です。

定義したシナリオは、エージェントに割り当てることで実行が開始されます。また、複数のキュー・アクションから成る機能のまとまりを1つのシナリオとして記述して、別のシナリオから呼び出すこともできます。

詳しくは、“シナリオ記述言語 Q 言語仕様書”を参照してください。

3.1. シナリオの記述

以下の形式でシナリオを定義します。

```
(defscenario シナリオ名 (引数リスト) シナリオ実体)
```

3.2. エージェントの作成

以下の形式でエージェントを定義します。

```
(defagent エージェント名 パラメータリスト)
```

※詳しくは、アクション・キューリファレンスを参照してください。

3.3. シナリオの割り当て

以下の形式でシナリオをエージェントに割り当てます。

```
(assign-scenario 'シナリオ名' エージェント名 パラメータリスト)
```

3.4. シナリオの呼び出し

シナリオ内で以下のように記述することで、シナリオから別のシナリオを呼び出すことができます。

```
(シナリオ名 self パラメータリスト)
```

4. シナリオ自動割り当て

以下の方法で、エージェントを自動的に作成してシナリオを割り当てます。

“robot-scenario” もしくは “animal-scenario” のフォルダ内に、“エージェント名.q” というファイル名でシナリオを記述したファイルを置くことで、エージェントに自動的にシナリオが割り当てられます。

1. ファイルの作成
“エージェント名.q” という名前でファイルを作成してください。ファイル名をそのままエージェント名としてエージェントが作成されます。
2. シナリオの記述
エージェントに割り当てたいシナリオを、作成したファイル内に記述してください。ファイル内に記述したシナリオが自動的にエージェントに割り当てられます。
3. ファイルの配置
ロボットの場合は “robot-scenario”，動物の場合は “animal-scenario” というフォルダ内にファイルを置いてください。

※robot-scenario フォルダに記述したものだけを実行したいときは、“scenario.q” と “agent.q” を読み込まないようにしてください。“scenario/main.q” の、(load "../scenario/scenario.q")および (load "../scenario/agent.q") の行頭に; (セミコロン) を加えてコメントアウトすることで、読み込まないようにすることができます。

※robot-scenario 内のシナリオを書き換えて再割り当てをするためには、シナリオを開放した上で、Q ウィンドウに以下のコマンドを打ち込んで再読み込みを行ってください。

```
(load "../lib/scenarioloader.sem")
```

5. サンプルシナリオ

Q の使い方の例として、いくつかのサンプルシナリオを用意しました。サンプルシナリオの動作を簡単に説明します。

初期状態では、シナリオ 1~8 が実行されるようになっています。実行するシナリオを変えたいときは、“scenario.q” の最後のシナリオ割り当ての部分を書き換えてください。行頭に “;” を付けるとコメントアウトされますので、実行させたいときは “;” を消し、実行させたくないときは “;” を付けるようにしてください。

動物にシナリオを実行させるためには、指定した名前の動物が存在している必要があります。シナリオを実行させたい動物に名前を付けるか、エージェントの作成の際に存在している動物の名前を指定してください。

1. 同じ発言を繰り返すシナリオ (scenario-speak)

“こんにちは”、“お元気ですか?”、“天気がいいですね!”、“さようなら” という言葉を繰り返します。

2. ユーザからの発言に返答するシナリオ (scenario-hear)

ユーザまたは周囲のキャラクターの言葉を聞いて、それに対して返答します。近くに scenario-speak を与えたエージェントがいる場合は、そのエージェントと対話します。

3. 円を描いて走り回るシナリオ (scenario-move)

円を描いてキャラクターが走り回ります。

4. \$name に指定したキャラクターを追い掛け回すシナリオ (scenario-follow)

指定したキャラクターを追い掛け回します。\$name に別の名前を指定すると、そのキャラクターを追い掛け回すようにできます。

5. アクションを繰り返すシナリオ (scenario-action)

横たわる、ジャンプ、座る、手を振る、おじぎのアクションを繰り返します。

6. 表情の変化を繰り返すシナリオ (scenario-express)

ほほを赤くする、大粒涙をこぼす、ときめく、目からビームを出す、はてなマークの表情を繰り返す。

7. \$name に指定した動物の名前を付け替えるシナリオ (scenario-name)

動物の名前を付け替える。\$name に別の名前を指定すると、その動物の名前を付け替えるようにできます。

8. バッジの付け替えを繰り返すシナリオ (scenario-badge)

バッジを順番に付け替えます。

9. アイテムの装備を繰り返すシナリオ (scenario-equip)

缶、バケツ、シャベル、手斧の装備を繰り返します。

10. 持っているアイテムを置いて拾うシナリオ (scenario-put)

缶、インクローラー、シャベル、バケツを周囲に置いてそれを拾うという動作を繰り返します。

11. シャベルで地面を掘ったり埋めたりするシナリオ (scenario-shovel)

地面を掘って、移動して、土を埋めるという動作を繰り返します。

12. シャベルで地面を掘ったり埋めたり固めたり、バケツで水をすくったり撒いたりするシナリオ (scenario-shovel2)

地面を掘って、埋めて、平らにして、なめらかにして、固めて、水をすくって、撒くという動作を繰り返します。

13. 植物の種を収穫して刈るシナリオ (scenario-harvest)

指定した植物を探して刈ります。種がついている場合は種を収穫してから刈ります。近くに指定した植物がない場合にはランダムに歩き回ります。“雑草”を別の植物に替えると、その植物を刈るようになります。

14. 動物を捕獲するシナリオ (scenario-fish)

指定した動物を探して狩り、肉を拾います。1000 モルより小さい動物は刈りません。肉を拾った場合は、今までに拾った肉の合計のモル数を言います。近くに指定した動物がない場合はランダムに歩き回ります。“フナ”を別の動物に変えると、その動物を狩るようにできます。

15. ランダムに歩き回るシナリオ (scenario-randomwalk)

ランダムに歩き回ります。

16. アイテムの装備を繰り返すシナリオ 2 (scenario-equip2)

手斧、バケツ、シャベルの順に装備したあと缶を装備し、すべての装備を外すという動作を繰り返します。

17. ロボット全員で踊るシナリオ (scenario-dance)

全体アクションの動作例を示すシナリオです。ロボット全員で同じ動きをして踊ります。

6. 各ファイルの説明

- runManager.bat
Q 処理系の起動, lib/main.scn の読み込み
- runManager_auto.bat
Q 処理系の起動, lib/main_auto.scn の読み込み
- lib/main.scn
gumonji/Q の起動に必要なファイルの読み込み
- lib/main_auto.scn
gumonji/Q の起動に必要なファイルの読み込み (自動実行)
- lib/nodelib.scn
gumonji/Q ライブラリ
- lib/handler.scn
ハンドラの記述
- lib/cueadapter.scn
キュー (観測) の記述
- lib/actionadapter.scn
アクション (行動) の記述
- lib/definition.q
キュー・アクションの設定
- lib/scenarioloader.scn
シナリオの自動割り当て
- scenario/main.q
シナリオとエージェント定義の読み込み

- `scenario/scenario.q`
シナリオの記述
- `scenario/agent.q`
エージェントの設定