Labyrinth - One Player mode

Generated by Doxygen 1.8.13

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

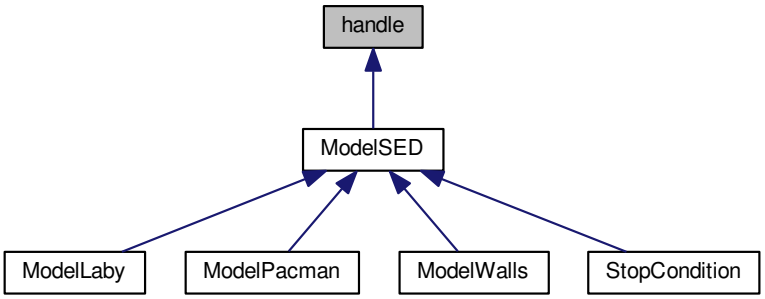
Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 handle Class Reference

Inheritance diagram for handle:



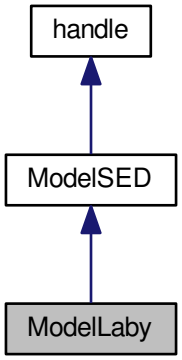
The documentation for this class was generated from the following file:

- ModelSED.m

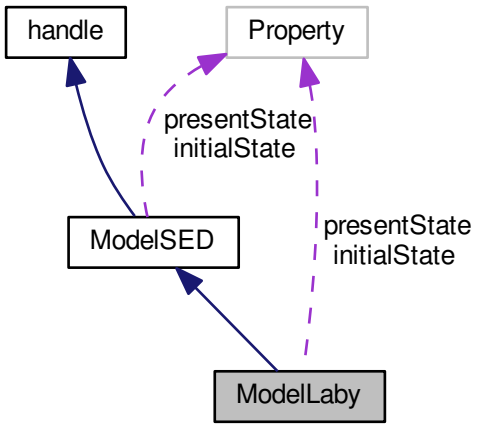
4.2 ModelLaby Class Reference

Class which contains the "fmg" structure of the labyrinth for 1 player.

Inheritance diagram for ModelLaby:



Collaboration diagram for ModelLaby:



Public Member Functions

- function ModelLaby (in wallsV_init, in wallsH_init, in pacman_init, in escape_init)

  Class constructor of Instance of ModelLaby Class.
- function f (in obj, in in)

  Compute the evolution of the model.
- function m (in obj, in nextState, in init)

  Memory method.
- function g (in obj)

  Create the outputs in a 1x9 cell-array.

Public Attributes

- Property presentState

  Data Structure of the current state of Labyrinth. It contains "wallsV", "wallsH" (2 matrix for the walls), "escape" and "pacman", a Cartesian position of current position of escape and pacman and 'wallsAroundPacman' A vector indicating the presence of a wall around the Pacman for the 4 directions Up Down Left Right.
- Property initialState

  Data Structure of the initial state of Labyrinth. It contains "wallsV", "wallsH" (2 matrix for the walls), "escape" and "pacman", a Cartesian position of current position of escape and pacman and 'wallsAroundPacman' A vector indicating the presence of a wall around the Pacman for the 4 directions Up Down Left Right.

4.2.1 Detailed Description

Class which contains the "fmg" structure of the labyrinth for 1 player.

You can change here labyrinth's dynamic : how objects and walls are evolving in the labyrinth, not the command of then.
Input : necessary information for compute the next state of the model

Output : output's action of the model

State : minimal information necessary who evolute

4.2.2 Constructor & Destructor Documentation

4.2.2.1 ModelLaby()

```
function ModelLaby (
            in wallsV_init,
            in wallsH_init,
            in pacman_init,
            in escape_init )
```

Class constructor of Instance of ModelLaby Class.

Parameters

wallsV_init	Contain a matrix (N, N-1) of Initial Vertical Walls.
wallsH_init	Contain a matrix (N-1, N) of Initial Horizontal Walls.
pacman_init	Contain a vector (x, y) of Initial Position of Pacman.
escape_init	Contain a vector (x, y) of Escape 's Position.

Returns

instance of the ModelLaby class.

4.2.3 Member Function Documentation

4.2.3.1 f()

```
function f (
             in obj,
             in in )  [virtual]
```

Compute the evolution of the model.

Parameters

obj	The instance which will evolve.
in	Input needed for the computing.

Returns

Next instance of the ModelLaby class.

Reimplemented from ModelSED.

4.2.3.2 g()

```
function g (
             in obj )  [virtual]
```

Create the outputs in a 1x9 cell-array.

Parameters

obj	the concerned instance of the class

Return values

out	Constructed output 1x9 cell-array of the model

Reimplemented from ModelSED.

4.2.3.3 m()

```
function m (
             in obj,
```

```
in nextState,
in init )  [virtual]
```

Memory method.

Update the state of the command.

Parameters

obj	The selected instance of the class
nextState	The value of the state need to update
init	Boolean condition for initialize or reset the command

Returns

instance of the class updated

Reimplemented from ModelSED.

4.2.4 Member Data Documentation

4.2.4.1 initialState

```
Property initialState
```

Data Structure of the initial state of Labyrinth. It contains "wallsV", "wallsH" (2 matrix for the walls), "escape" and "pacman", a Cartesian position of current position of escape and pacman and 'wallsAroundPacman' A vector indicating the presence of a wall around the Pacman for the 4 directions Up Down Left Right.

4.2.4.2 presentState

```
Property presentState
```

Data Structure of the current state of Labyrinth. It contains "wallsV", "wallsH" (2 matrix for the walls), "escape" and "pacman", a Cartesian position of current position of escape and pacman and 'wallsAroundPacman' A vector indicating the presence of a wall around the Pacman for the 4 directions Up Down Left Right.

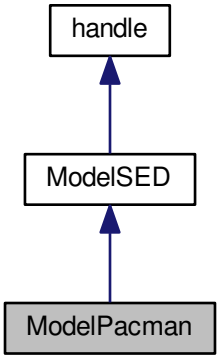
The documentation for this class was generated from the following file:

- ModelLaby.m

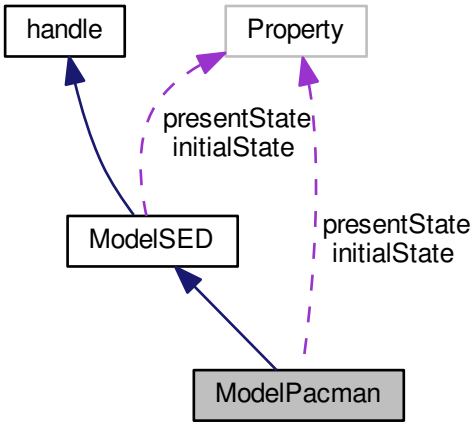
4.3 ModelPacman Class Reference

Contain ghost Pacman control.

Inheritance diagram for ModelPacman:



Collaboration diagram for ModelPacman:



Public Member Functions

- function ModelPacman (in initialValue)

  Class constructor.
- function f (in obj, in in)

Compute the evolution of the command.

- function m (in obj, in nextState, in init)

  Memory method update the state of the command.

- function g (in obj)

  Create the outputs.

Public Attributes

- Property presentState

  This is the state of the command in the present moment.

- Property initialState

  This is the state of the command in the initialization and when it's reseted.

4.3.1 Detailed Description

Contain ghost Pacman control.

You can change here Pacman's command.
Input : Possible Pacman's moves [Up Down Left Right]
0 = move not possible ; 1 = move possible
(Wout{7})

Output : Pacman's moves 1 : pacmanLeftBut, (Wout(3))
2 : pacmanUpBut, (Wout(1))
3 : pacmanRightBut, (Wout(4))
4 : pacmanDownBut , (Wout(2))
(Win(4:7) of wrapper)

Input : Walls around Pacman
1 up
2 down
3 left
4 right
This command do the sequence P(D) $>$ P(B) $>$ P(H) $>$ P(G)

4.3.2 Constructor & Destructor Documentation

4.3.2.1 ModelPacman()

```
function ModelPacman (
            in initialValue )
```

Class constructor.

Parameters

initialValue	Contain the initial state

Returns

instance of the ModelPacman class.

4.3.3 Member Function Documentation

4.3.3.1 f()

```
function f (
             in obj,
             in in )  [virtual]
```

Compute the evolution of the command.

Parameters

obj	The instance who evolute
in	Input needed for the compute

Return values

nextState	The future state of the Pacman command

Reimplemented from ModelSED.

4.3.3.2 g()

```
function g (
             in obj )  [virtual]
```

Create the outputs.

Parameters

obj	the concerned instance of the class

Return values

out	The output who is the command.

Reimplemented from ModelSED.

4.3.3.3 m()

```
function m (
             in obj,
             in nextState,
             in init )  [virtual]
```

Memory method update the state of the command.

Parameters

obj	The selected instance of the class
nextState	The value of the state need to update
init	Boolean condition for initialize or reset the command

Returns

instance of the class updated

Reimplemented from ModelSED.

4.3.4 Member Data Documentation

4.3.4.1 initialState

```
Property initialState
```

This is the state of the command in the initialization and when it's reseted.

4.3.4.2 presentState

```
Property presentState
```

This is the state of the command in the present moment.

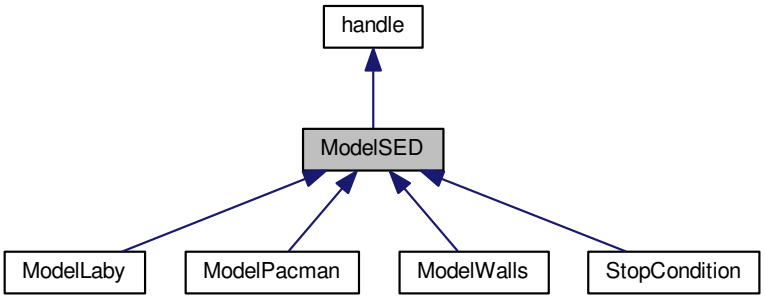
The documentation for this class was generated from the following file:

- ModelPacman.m

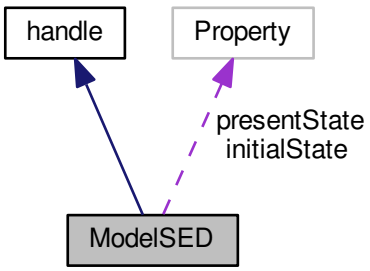
4.4 ModelSED Class Reference

Abstract Class who contain the structure of a "fmg" implementation.

Inheritance diagram for ModelSED:



Collaboration diagram for ModelSED:



Public Member Functions

- virtual f (in obj, in in)

  Compute the evolution of the model.
- virtual m (in obj, in nextState, in init)

  Memory method update the state of the command.
- virtual g (in obj)

  Create the outputs.

Public Attributes

- Property presentState

  This is the state of the command in the present moment.
- Property initialState

  This is the state of the command in the initialization and when it's reseted.

4.4.1 Detailed Description

Abstract Class who contain the structure of a "fmg" implementation.

This class is used for give a general definition of Model Class.
Input : necessary information for compute the next state of the model

Output : output's action of the model
State : minimal information necessary who evolute

4.4.2 Member Function Documentation

4.4.2.1 f()

```
virtual f (
            in obj,
            in in )  [virtual]
```

Compute the evolution of the model.

Parameters

obj	The instance who evolute
in	Input needed for the computing

Return values

nextState	The future state of the model

Reimplemented in ModelPacman, and ModelLaby.

4.4.2.2 g()

```
virtual g (
            in obj )  [virtual]
```

Create the outputs.

Parameters

obj	the concerned instance of the class

Return values

out	Constructed output of the model

Reimplemented in ModelPacman, ModelLaby, StopCondition, and ModelWalls.

4.4.2.3 m()

```
virtual m (
            in obj,
            in nextState,
            in init )  [virtual]
```

Memory method update the state of the command.

Parameters

obj	The selected instance of the class
nextState	The value of the state need to update
init	Boolean condition for initialize or reset the command

Returns

instance of the class updated

Reimplemented in ModelPacman, ModelLaby, StopCondition, and ModelWalls.

4.4.3 Member Data Documentation

4.4.3.1 initialState

```
Property initialState
```

This is the state of the command in the initialization and when it's reseted.

4.4.3.2 presentState

```
Property presentState
```

This is the state of the command in the present moment.

The documentation for this class was generated from the following file:

- ModelSED.m

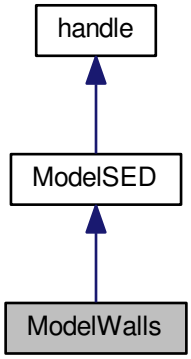
4.5 ModelWalls Class Reference

Input : No need

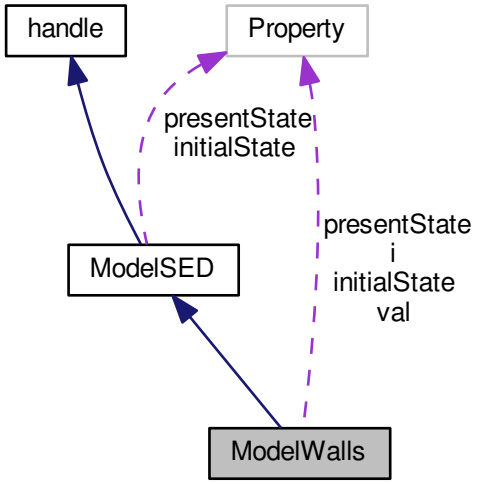
Output : [UPwalls , RIGHTwalls]

State : contain the last move (0 = up ; 1 = right)
This command do the sequence walls Right –> walls down
.

Inheritance diagram for ModelWalls:



Collaboration diagram for ModelWalls:



Public Member Functions

- function ModelWalls (in initValue)

  Class constructor.
- function f (in obj)
- function m (in obj, in nextState, in init)

  Memory method update the state of the command.
- function g (in obj)

  Create the outputs.
- virtual f (in obj, in in)

  Compute the evolution of the model.

Public Attributes

- Property presentState
- Property initialState
- Property i
- Property val

4.5.1 Detailed Description

Input : No need

Output : [UPwalls , RIGHTwalls]

State : contain the last move (0 = up ; 1 = right)
This command do the sequence walls Right –> walls down
.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 ModelWalls()

```
function ModelWalls (
            in initValue )
```

Class constructor.

Parameters

initialValue	Contain the initial state

Returns

instance of the ModelWalls class.

4.5.3 Member Function Documentation

4.5.3.1 f() [1/2]

```
virtual f (
            in obj,
            in in )  [virtual], [inherited]
```

Compute the evolution of the model.

Parameters

obj	The instance who evolute
in	Input needed for the computing

Return values

nextState	The future state of the model

Reimplemented in ModelPacman, and ModelLaby.

4.5.3.2 f() [2/2]

```
function f (
            in obj )
```

4.5.3.3 g()

```
function g (
            in obj )  [virtual]
```

Create the outputs.

Parameters

obj	the concerned instance of the class

Return values

out	The output who is the command.

Reimplemented from ModelSED.

4.5.3.4 m()

```
function m (
             in obj,
             in nextState,
             in init )  [virtual]
```

Memory method update the state of the command.

Parameters

obj	The selected instance of the class
nextState	The value of the state need to update
init	Boolean condition for initialize or reset the command

Returns

instance of the class updated

Reimplemented from ModelSED.

4.5.4 Member Data Documentation

4.5.4.1 i

```
Property i
```

4.5.4.2 initialState

```
Property initialState
```

4.5.4.3 presentState

```
Property presentState
```

4.5.4.4 val

`Property val`

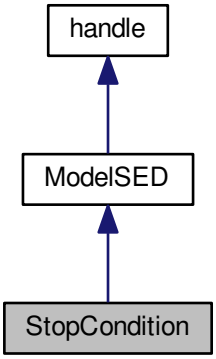
The documentation for this class was generated from the following file:

- ModelWalls.m

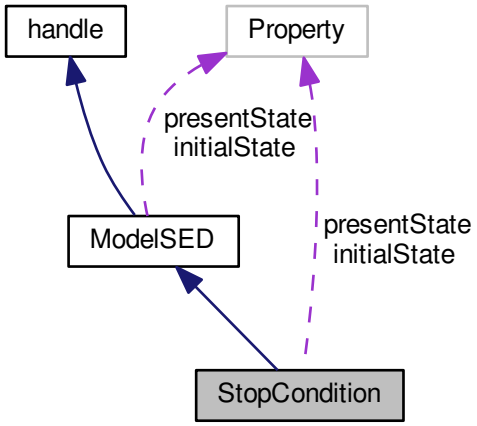
4.6 StopCondition Class Reference

Class used to manage shutdown conditions.

Inheritance diagram for StopCondition:



Collaboration diagram for StopCondition:



Public Member Functions

- function StopCondition (in initCondition)

  Class constructor of Instance of StopCondition Class.
- function f (in obj, in noEscape, in pacmanWallsBreak)

  Compute the evolution of the model.
- function m (in obj, in nextState, in init)

  Memory method.
- function g (in obj)

  Create the outputs in a vector with 4 parameters.
- virtual f (in obj, in in)

  Compute the evolution of the model.

Public Attributes

- Property presentState

  Data Structure of the current state of Labyrinth. It contains "wallsV", "wallsH" (2 matrix for the walls), "escape" and "pacman", a Cartesian position of current position of escape and pacman and 'wallsAroundPacman' A vector indicating the presence of a wall around the Pacman for the 4 directions Up Down Left Right.
- Property initialState

  Data Structure of the initial state of Labyrinth. It contains "wallsV", "wallsH" (2 matrix for the walls), "escape" and "pacman", a Cartesian position of current position of escape and pacman and 'wallsAroundPacman' A vector indicating the presence of a wall around the Pacman for the 4 directions Up Down Left Right.

4.6.1 Detailed Description

Class used to manage shutdown conditions.

Labyrinth shutdown conditions model.
You can modify the shutdown conditions here. It is developing in the same way as MODELSED, with "FMG" block. (MODELSED's legacy)
Input : walls of Pacman's
walls of ghost's
escape of Pacman
CaughtBreak

Output : 1 Escape
2 Caugnt
3 pacmanWallsBreak
4 ghostWallsBreak

4.6.2 Constructor & Destructor Documentation

4.6.2.1 StopCondition()

```
function StopCondition (
            in initCondition )
```

Class constructor of Instance of StopCondition Class.

Parameters

initCondition	Structure for the InitialState. It have to contain : 'escape', 'caught', 'pacman' and 'numberOfPossibleCaught'

Returns

instance of the ModelLaby class.

4.6.3 Member Function Documentation

4.6.3.1 f() [1/2]

```
virtual f (
            in obj,
            in in )  [virtual], [inherited]
```

Compute the evolution of the model.

Parameters

obj	The instance who evolute
in	Input needed for the computing

Return values

nextState	The future state of the model

Reimplemented in ModelPacman, and ModelLaby.

4.6.3.2 f() [2/2]

```
function f (
            in obj,
            in noEscape,
            in pacmanWallsBreak )
```

Compute the evolution of the model.

Parameters

obj	The instance which will evolve.
in	Input needed for the computing.

Returns

Next instance of the StopCondition class.

4.6.3.3 g()

```
function g (
             in obj )  [virtual]
```

Create the outputs in a vector with 4 parameters.

Parameters

obj	the concerned instance of the class

Return values

out	Constructed output vector with 4 parameters of the model. In this case, the output contain only escape information and Pacman block information.

Reimplemented from ModelSED.

4.6.3.4 m()

```
function m (
             in obj,
             in nextState,
             in init )  [virtual]
```

Memory method.

Update the state of the command.

Parameters

obj	The selected instance of the class
nextState	The value of the state need to update
init	Boolean condition for initialize or reset the command

Returns

instance of the class updated

Reimplemented from ModelSED.

4.6.4 Member Data Documentation

4.6.4.1 initialState

`Property initialState`

Data Structure of the initial state of Labyrinth. It contains "wallsV", "wallsH" (2 matrix for the walls), "escape" and "pacman", a Cartesian position of current position of escape and pacman and 'wallsAroundPacman' A vector indicating the presence of a wall around the Pacman for the 4 directions Up Down Left Right.

4.6.4.2 presentState

`Property presentState`

Data Structure of the current state of Labyrinth. It contains "wallsV", "wallsH" (2 matrix for the walls), "escape" and "pacman", a Cartesian position of current position of escape and pacman and 'wallsAroundPacman' A vector indicating the presence of a wall around the Pacman for the 4 directions Up Down Left Right.

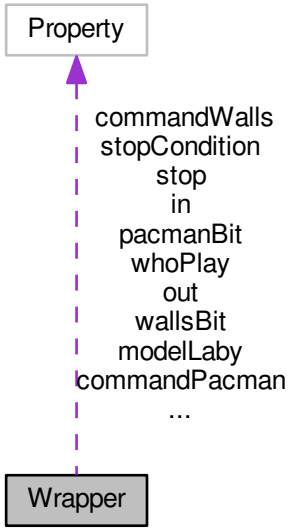
The documentation for this class was generated from the following file:

- StopCondition.m

4.7 Wrapper Class Reference

WRAPPER Global organization of the project.

Collaboration diagram for Wrapper:



Public Member Functions

- function Wrapper (in inSize, in outSize, in initLaby, in initWalls, in initPac, in initStop)

  Class constructor of Instance of StopCondition Class.
- function updateConnexion (in obj, in indBit, in value)

  Update the connection bit for connect automatic mode for Pacman and/or the walls.
- function init (in obj)

  Allows to completely reset the labyrinth.
- function orderer (in obj, in vectIn)

  Ordinate the global execution of Models.
- function get_stop (in obj)

  Return the current State of shutdown condition.
- function get_out (in obj)

  Return the current State of output cell.

Public Attributes

- Property wallsBit

  Boolean connection for the walls.
- Property pacmanBit

  Boolean connection for the Pacman.
- Property modelLaby

  contain the instance of the model of labyrinth.
- Property commandWalls

  contain the instance of wall's command.
- Property commandPacman

  contain the instance of Pacman's command.
- Property stopCondition

  contain the instance shutdown condition.
- Property in

  A integer vector who contain the state of input,.
- Property out

  A cell who contain the state of output,.
- Property stop

  A cell that contains the output of the stop conditions instance.
- Property whoPlay

  A increment integer that permit to know which object to play 0 = walls ; 1 = Pacman ;.

4.7.1 Detailed Description

WRAPPER Global organization of the project.

Contain the connection between the different elements and contain all the models (walls, labyrinth, Pacman, escape)

4.7.2 Constructor & Destructor Documentation

4.7.2.1 Wrapper()

```
function Wrapper (
            in inSize,
            in outSize,
            in initLaby,
            in initWalls,
            in initPac,
            in initStop )
```

Class constructor of Instance of StopCondition Class.

Parameters

inSize	Integer containing the size of the state of inputs.
outSize	Integer containing the size of the state of Outputs.
initLaby	Structure containing every fields need to initialize the labyrinth Model.
initWalls	Structure containing every fields need to initialize the Walls Model.
initPac	Structure containing every fields need to initialize the Pacman Model.
initStop	Structure containing every fields need to initialize the Stop Model.

Returns

instance of the Wrapper class.

4.7.3 Member Function Documentation

4.7.3.1 get_out()

```
function get_out (
            in obj )
```

Return the current State of output cell.

Parameters

obj	The instance of Wrapper.

Returns

Output cell.

4.7.3.2 get_stop()

```
function get_stop (
            in obj )
```

Return the current State of shutdown condition.

Parameters

obj	The instance of Wrapper.

Returns

instance of the Stop class.

4.7.3.3 init()

```
function init (
             in obj )
```

Allows to completely reset the labyrinth.

Parameters

obj	The instance of Wrapper.

Returns

instance of the Wrapper class.

4.7.3.4 orderer()

```
function orderer (
             in obj,
             in vectIn )
```

Ordinate the global execution of Models.

Parameters

obj	The instance of Wrapper.

Returns

instance of the Wrapper class. In a first case, it checks which models is connected via the interface. If a model is connected, we execute the 'fmg' structure of the model, else, we are waiting that a player push the desired button to move the labyrinth.

Here a little graphic about the scheduling of the call of each model : murs > Laby > pacman > laby
Every call of 'laby' implies a call of Stop Condition.

4.7.3.5 updateConnexion()

```
function updateConnexion (
            in obj,
            in indBit,
            in value )
```

Update the connection bit for connect automatic mode for Pacman and/or the walls.

Parameters

obj	The instance of Wrapper.
indBit	Integer pointing the element to be connected : '1' for walls and '3' for Pacman.
value	Boolean indicating if the element is connected (True) or not.

Returns

instance of the Wrapper class.

4.7.4 Member Data Documentation

4.7.4.1 commandPacman

```
Property commandPacman
```

contain the instance of Pacman's command.

4.7.4.2 commandWalls

```
Property commandWalls
```

contain the instance of wall's command.

4.7.4.3 in

```
Property in
```

A integer vector who contain the state of input,.

incremented by the callback or some action.

4.7.4.4 modelLaby

`Property modelLaby`

contain the instance of the model of labyrinth.

4.7.4.5 out

`Property out`

A cell who contain the state of output,.

incremented by the callback or some action.

4.7.4.6 pacmanBit

`Property pacmanBit`

Boolean connection for the Pacman.

4.7.4.7 stop

`Property stop`

A cell that contains the output of the stop conditions instance.

incremented by the callback or some action.

4.7.4.8 stopCondition

`Property stopCondition`

contain the instance shutdown condition.

4.7.4.9 wallsBit

`Property wallsBit`

Boolean connection for the walls.

4.7.4.10 whoPlay

`Property whoPlay`

A increment integer that permit to know which object to play 0 = walls ; 1 = Pacman ;.

The documentation for this class was generated from the following file:

- Wrapper.m

Chapter 5

File Documentation

5.1 CreatePituresAndVideo.m File Reference

Functions

- function CreatePituresAndVideo (in n, in escape_i, in labyState)

5.1.1 Function Documentation

5.1.1.1 CreatePituresAndVideo()

```
function CreatePituresAndVideo (
            in n,
            in escape_i,
            in labyState )
```

5.2 CreatePituresAndVideo_textured.m File Reference

Functions

- function CreatePituresAndVideo_textured (in n, in escape_i, in labyState)

5.2.1 Function Documentation

5.2.1.1 CreatePituresAndVideo_textured()

```
function CreatePituresAndVideo_textured (
             in n,
             in escape_i,
             in labyState )
```

5.3 figure_Laby.m File Reference

Functions

- function figure_Laby (in varargin)

  figure_Laby.m
- function figure_Laby_OpeningFcn (in hObject, in eventdata, in handles, in varargin)

  initialization function.
- function figure_Laby_OutputFcn (in hObject, in eventdata, in handles)

  Automatic generated function by GUI.
- function ui_Callback (in hObject, in eventdata, in handles)

  Callback for all the action's buttons (see detailed explications).
- function connect_Callback (in hObject, in eventdata, in handles)

  Callback for all the connection's buttons (see detailed explications).
- function createUIPacman (in handles)

  Creation of the graphical object "pacman".
- function createUIWalls (in handles)

  Creation of the graphical objects "walls".
- function createUIEscape (in handles)

  Creation of the graphical objects "escape".
- function updateUI (in handles, in out)

  This function update all graphicals element who can change.
- function updateUIActiveCammand (in handles)

  Update visibility of control panel, connection and step button.
- function updateUIButton (in handles)

  Show the needed moving buttons.
- function updateUIPlayer (in handles, in strPlayer, in position)

  Update graphical place of a player (only pacman in this case).
- function updateUIEscape (in elementToSet, in boolState)
- function updateUIWallsAround (in handles, in strElement, in wallsAround)
- function updateUIWalls (in wallsUI, in vertWalls, in horizWalls)
- function isOne (in boolCond)
- function updatePresenceDetectorDisplay (in elementToSet, in boolCondition)
- function resetUIConnection (in handles)

5.3.1 Function Documentation

5.3.1.1 connect_Callback()

```
function connect_Callback (
             in hObject,
             in eventdata,
             in handles )
```

Callback for all the connection's buttons (see detailed explications).

in the following image, buttons marked with a red arrow lanch this Callback.

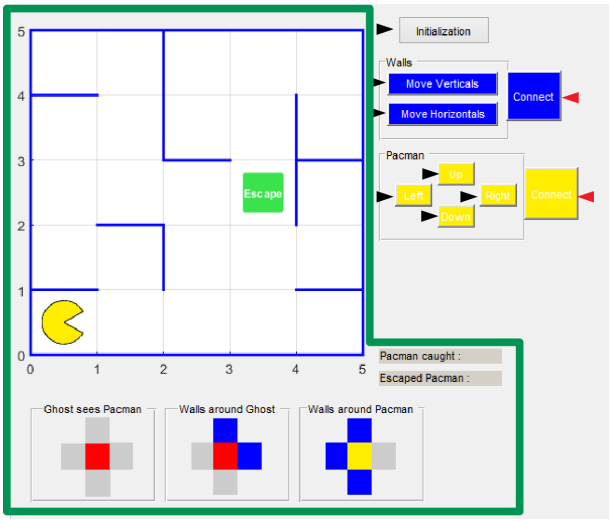


Figure 5.1 button's type of GUI

This callback lanch updateConnexion method of Wrapper class, which modify what command are automatic.

Parameters

hObject	handle to actived button
eventdata	reserved - to be defined in a future version of MATLAB
handles	structure with handles and user data (see GUIDATA)

5.3.1.2 createUIEscape()

```
function createUIEscape (
             in handles )
```

Creation of the graphical objects "escape".

The escape is created whit a rectangle and and text box. It's stored into handles in 'escape'.

Parameters

handles	structure with handles and user data (see GUIDATA)

Returns

h the updated structure with handles and user data (see GUIDATA)

5.3.1.3 createUIPacman()

```
function createUIPacman (
             in handles )
```

Creation of the graphical object "pacman".

The pacman is created by using the patch function and store into the handle in 'pacman'.

Parameters

handles	structure with handles and user data (see GUIDATA)

Returns

h the updated structure with handles and user data (see GUIDATA)

5.3.1.4 createUIWalls()

```
function createUIWalls (
             in handles )
```

Creation of the graphical objects "walls".

The walls are created as two line elmenents matrix. They are stored into handles in 'walls'.
The first matrix is for the verticals walls and named 'horizontals' and the second called 'verticals' for the verticals walls.
All possible walls are created and it is by making them visible or invisible that they appear or disappear.

Parameters

handles	structure with handles and user data (see GUIDATA)

Returns

h the updated structure with handles and user data (see GUIDATA)

5.3.1.5 figure_Laby()

```
function figure_Laby (
            in varargin )
```

figure_Laby.m

Script linked to the graphical interface whitch contain all the graphical functions. This file contain also the instance of Wrapper class. All the handles of graphical elements and instance of class are stored into the "handles" structure. function call when figure_Laby si open. It's initialize the UI.

Parameters

varargin	Several inputs.

Returns

varargout Several Outputs.

5.3.1.6 figure_Laby_OpeningFcn()

```
function figure_Laby_OpeningFcn (
            in hObject,
            in eventdata,
            in handles,
            in varargin )
```

initialization function.

It's where is initialize the parameters of the labyrinth and all the commands in the section "INITIAL PARAMETERS OF THE LABYRINTH AND THE COMMANDS".

Parameters

hObject	handle to figure
eventdata	reserved - to be defined in a future version of MATLAB
handles	structure with handles and user data (see GUIDATA)
varargin	command line arguments to figure_Laby (see VARARGIN)

5.3.1.7 figure_Laby_OutputFcn()

```
function figure_Laby_OutputFcn (
            in hObject,
            in eventdata,
            in handles )
```

Automatic generated function by GUI.

Parameters

hObject	handle to figure
eventdata	reserved - to be defined in a future version of MATLAB
handles	structure with handles and user data (see GUIDATA)

Returns

varargout cell array for returning output args (see VARARGOUT);

5.3.1.8 isOne()

```
function isOne (
             in boolCond )
```

5.3.1.9 resetUIConnection()

```
function resetUIConnection (
             in handles )
```

5.3.1.10 ui_Callback()

```
function ui_Callback (
             in hObject,
             in eventdata,
             in handles )
```

Callback for all the action's buttons (see detailed explications).

in the following image, buttons marked with a black arrow lanch this Callback.

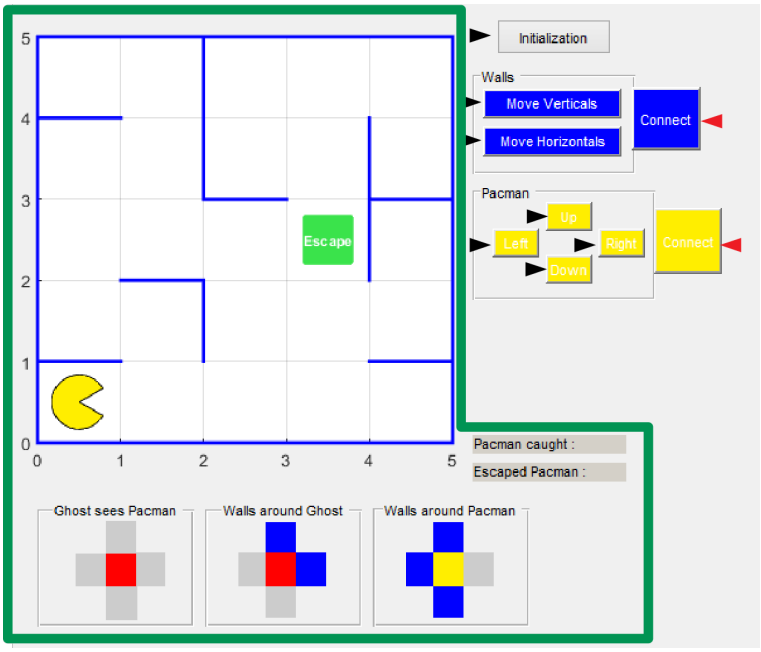


Figure 5.2 button's type of GUI

This callback lanch orderer method of Wrapper class, which allows the simulation to evolve.

Parameters

hObject	handle to actived button
eventdata	reserved - to be defined in a future version of MATLAB
handles	structure with handles and user data (see GUIDATA)

5.3.1.11 updatePresenceDetectorDisplay()

```
function updatePresenceDetectorDisplay (
             in elementToSet,
             in boolCondition )
```

5.3.1.12 updateUI()

```
function updateUI (
             in handles,
             in out )
```

This function update all graphicals element who can change.

With the input called 'out', this function lanch all the functions who update a specific graphical element.

Parameters

handles	Structure with handles and user data (see GUIDATA)
out	Cell who contain all informations needed from the wrapper for update the graphical interface.

5.3.1.13 updateUIActiveCammand()

```
function updateUIActiveCammand (
            in handles )
```

Update visibility of control panel, connection and step button.

This function show or hide the control's panels and the connection's buttons according whit who will move. It also show step button if a command is connected.
Example : if is pacman time to move and command is not connected, this function hide walls and step element and show pacman one's.

Parameters

handles	structure with handles and user data (see GUIDATA)

5.3.1.14 updateUIButton()

```
function updateUIButton (
            in handles )
```

Show the needed moving buttons.

This function show the direction's buttons allows by the output informations of modelLaby and hide the others one.

Parameters

handles	structure with handles and user data (see GUIDATA)

5.3.1.15 updateUIEscape()

```
function updateUIEscape (
            in elementToSet,
            in boolState )
```

5.3.1.16 updateUIPlayer()

```
function updateUIPlayer (
            in handles,
            in strPlayer,
            in position )
```

Update graphical place of a player (only pacman in this case).

This function, with the actual position (present in the handles) and the new one as a input, move object.
The dynamics of movement is defined by this foncion $out(t) = \frac{\frac{om+1}{om*e^{cv*t}+1}-1}{om}$ for $t \in [0,1]$, $om = 72.89105$ and $cv = -11.27357$.

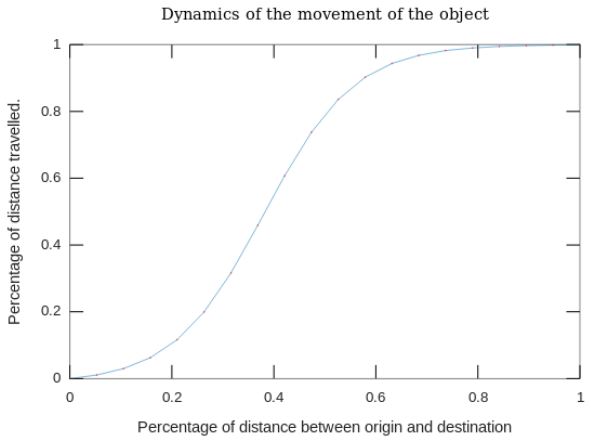


Figure 5.3 Dynamics of movement

Parameters

strPlayer	String contain the exact name of the object to move.
position	new position of the object. format : [x y]
handles	structure with handles and user data (see GUIDATA)

5.3.1.17 updateUIWalls()

```
function updateUIWalls (
            in wallsUI,
            in vertWalls,
            in horizWalls )
```

5.3.1.18 updateUIWallsAround()

```
function updateUIWallsAround (
            in handles,
            in strElement,
            in wallsAround )
```

5.4 main.m File Reference

5.5 ModelLaby.m File Reference

Classes

- class ModelLaby

  Class which contains the "fmg" structure of the labyrinth for 1 player.

5.6 ModelPacman.m File Reference

Classes

- class ModelPacman

  Contain ghost Pacman control.

5.7 ModelSED.m File Reference

Classes

- class ModelSED

  Abstract Class who contain the structure of a "fmg" implementation.

5.8 ModelWalls.m File Reference

Contain wall movement command.

Classes

- class ModelWalls

  Input : No need

  Output : [UPwalls , RIGHTwalls]

  State : contain the last move (0 = up ; 1 = right)
  This command do the sequence walls Right –> walls down
  .

5.8.1 Detailed Description

Contain wall movement command.

5.9 setColor.m File Reference

Functions

- function setColor (in img, in imgRef, in colors, in indice)

5.9.1 Function Documentation

5.9.1.1 setColor()

```
function setColor (
            in img,
            in imgRef,
            in colors,
            in indice )
```

5.10 Simulation.m File Reference

5.11 StopCondition.m File Reference

Classes

- class StopCondition

  Class used to manage shutdown conditions.

5.12 visupacman.m File Reference

5.13 visupacman2.m File Reference

5.14 wallsBorder.m File Reference

Functions

- function wallsBorder (in walls)

5.14.1 Function Documentation

5.14.1.1 wallsBorder()

```
function wallsBorder (
             in walls )
```

5.15 Wrapper.m File Reference

Classes

- class Wrapper

  WRAPPER Global organization of the project.

Index