

Labyrinth \-- Two Players mode

Generated by Doxygen 1.8.13

Contents

1	Hierarchical Index	1
1.1	Class Hierarchy	1
2	Class Index	3
2.1	Class List	3
3	File Index	5
3.1	File List	5
4	Class Documentation	7
4.1	handle Class Reference	7
4.2	ModelCommand Class Reference	8
4.2.1	Member Function Documentation	8
4.2.1.1	f()	8
4.2.1.2	g()	9
4.2.1.3	m()	9
4.2.2	Member Data Documentation	9
4.2.2.1	Down	9
4.2.2.2	knowCompart	9
4.2.2.3	Left	9
4.2.2.4	presentState	9
4.2.2.5	Right	9
4.2.2.6	sizeTab	10
4.2.2.7	Up	10
4.3	ModelGhost Class Reference	10

4.3.1	Detailed Description	11
4.3.2	Constructor & Destructor Documentation	12
4.3.2.1	ModelGhost()	12
4.3.3	Member Function Documentation	12
4.3.3.1	f() [1/2]	12
4.3.3.2	f() [2/2]	14
4.3.3.3	g()	14
4.3.3.4	m()	15
4.3.4	Member Data Documentation	15
4.3.4.1	initialState	15
4.3.4.2	presentState	16
4.4	ModelLaby Class Reference	16
4.4.1	Detailed Description	17
4.4.2	Constructor & Destructor Documentation	17
4.4.2.1	ModelLaby()	18
4.4.3	Member Function Documentation	18
4.4.3.1	f()	18
4.4.3.2	g()	19
4.4.3.3	m()	19
4.4.3.4	sameX_position()	19
4.4.3.5	sameY_position()	20
4.4.3.6	wallsHBetween()	20
4.4.3.7	wallsHBetweenOne()	20
4.4.3.8	wallsVBetween()	21
4.4.3.9	wallsVBetweenOne()	21
4.4.4	Member Data Documentation	21
4.4.4.1	initialState	21
4.4.4.2	presentState	22
4.5	ModelPacman Class Reference	22
4.5.1	Detailed Description	24

4.5.2	Constructor & Destructor Documentation	24
4.5.2.1	ModelPacman()	24
4.5.3	Member Function Documentation	24
4.5.3.1	f()	24
4.5.3.2	g()	26
4.5.3.3	m()	26
4.5.4	Member Data Documentation	27
4.5.4.1	i	27
4.5.4.2	initialState	27
4.5.4.3	memory	27
4.5.4.4	presentState	27
4.6	ModelSED Class Reference	28
4.6.1	Detailed Description	29
4.6.2	Member Function Documentation	29
4.6.2.1	f()	29
4.6.2.2	g()	29
4.6.2.3	m()	30
4.6.3	Member Data Documentation	30
4.6.3.1	initialState	30
4.6.3.2	presentState	30
4.7	ModelWalls Class Reference	31
4.7.1	Detailed Description	32
4.7.2	Constructor & Destructor Documentation	32
4.7.2.1	ModelWalls()	32
4.7.3	Member Function Documentation	33
4.7.3.1	f() [1/2]	33
4.7.3.2	f() [2/2]	33
4.7.3.3	g()	33
4.7.3.4	m()	35
4.7.4	Member Data Documentation	35

4.7.4.1	i	35
4.7.4.2	initialState	35
4.7.4.3	presentState	36
4.7.4.4	val	36
4.8	StopCondition Class Reference	36
4.8.1	Detailed Description	38
4.8.2	Constructor & Destructor Documentation	38
4.8.2.1	StopCondition()	38
4.8.3	Member Function Documentation	38
4.8.3.1	f() [1/2]	38
4.8.3.2	f() [2/2]	39
4.8.3.3	g()	39
4.8.3.4	m()	40
4.8.4	Member Data Documentation	40
4.8.4.1	initialState	40
4.8.4.2	presentState	41
4.9	Wrapper Class Reference	41
4.9.1	Constructor & Destructor Documentation	42
4.9.1.1	Wrapper()	42
4.9.2	Member Function Documentation	42
4.9.2.1	get_out()	42
4.9.2.2	get_stop()	42
4.9.2.3	init()	43
4.9.2.4	orderer()	43
4.9.2.5	updateConnexion()	43
4.9.3	Member Data Documentation	43
4.9.3.1	commandGhost	43
4.9.3.2	commandPacman	43
4.9.3.3	commandWalls	43
4.9.3.4	ghostBit	44
4.9.3.5	in	44
4.9.3.6	modelLaby	44
4.9.3.7	out	44
4.9.3.8	pacmanBit	44
4.9.3.9	stop	44
4.9.3.10	stopCondition	44
4.9.3.11	wallsBit	44
4.9.3.12	whoPlay	44

5 File Documentation	45
5.1 CreatePituresAndVideo.m File Reference	45
5.1.1 Function Documentation	45
5.1.1.1 CreatePituresAndVideo()	45
5.2 CreatePituresAndVideo_textured.m File Reference	45
5.2.1 Function Documentation	45
5.2.1.1 CreatePituresAndVideo_textured()	46
5.3 figure_Laby.m File Reference	46
5.3.1 Function Documentation	46
5.3.1.1 connect_Callback()	46
5.3.1.2 createUIEscape()	46
5.3.1.3 createUIGhost()	47
5.3.1.4 createUIPacman()	47
5.3.1.5 createUIWalls()	47
5.3.1.6 figure_Laby()	47
5.3.1.7 figure_Laby_OpeningFcn()	47
5.3.1.8 figure_Laby_OutputFcn()	47
5.3.1.9 isOne()	48
5.3.1.10 resetUIConnection()	48
5.3.1.11 ui_Callback()	48
5.3.1.12 updatePresenceDetectorDisplay()	48
5.3.1.13 updateUI()	48
5.3.1.14 updateUIActiveCammand()	48
5.3.1.15 updateUIButton()	49
5.3.1.16 updateUICaught()	49
5.3.1.17 updateUIEscape()	49
5.3.1.18 updateUIPlayer()	49
5.3.1.19 updateUIWalls()	49
5.3.1.20 updateUIWallsAround()	49
5.4 main.m File Reference	50

5.5	matrixAllPossible.m File Reference	50
5.6	ModelCommand.m File Reference	50
5.7	ModelGenerator/AutomatonSchedulingCreation.m File Reference	50
5.7.1	Function Documentation	50
5.7.1.1	function()	50
5.8	ModelGenerator/AutomatonStrutureLabyCreation.m File Reference	50
5.8.1	Function Documentation	50
5.8.1.1	AutomatonStrutureLabyCreation()	50
5.9	ModelGenerator/AutomatonWallsConstraintsCreation.m File Reference	51
5.9.1	Function Documentation	51
5.9.1.1	AutomatonWallsConstraintsCreation()	51
5.10	ModelGenerator/generer_lab.m File Reference	51
5.10.1	Function Documentation	51
5.10.1.1	generer_lab()	51
5.11	ModelGenerator/modelGenerator.m File Reference	51
5.12	ModelGenerator/Plan_desumaFunctions_2Players.m File Reference	51
5.12.1	Function Documentation	52
5.12.1.1	AutomatonStrutureLabyCreation()	52
5.12.1.2	function()	52
5.12.1.3	SaveDESUMAFile()	52
5.12.1.4	writeStates()	52
5.12.1.5	writeTransitions()	52
5.13	ModelGenerator/SaveDESUMAFile.m File Reference	53
5.13.1	Function Documentation	53
5.13.1.1	SaveDESUMAFile()	53
5.14	ModelGenerator/writeStates.m File Reference	53
5.14.1	Function Documentation	53
5.14.1.1	writeStates()	53
5.15	ModelGenerator/writeTransitions.m File Reference	53
5.15.1	Function Documentation	54

5.15.1.1 writeTransitions()	54
5.16 ModelGhost.m File Reference	54
5.17 ModelLaby.m File Reference	54
5.18 ModelPacman.m File Reference	54
5.19 ModelSED.m File Reference	54
5.20 ModelWalls.m File Reference	54
5.21 setColor.m File Reference	55
5.21.1 Function Documentation	55
5.21.1.1 setColor()	55
5.22 Simulation.m File Reference	55
5.23 Simulation2_allpossiblewalls.m File Reference	55
5.24 StopCondition.m File Reference	55
5.25 validation/Validation 2/Test1/validation2.m File Reference	55
5.26 validation/Validation 2/Test10/validation2.m File Reference	55
5.27 validation/Validation 2/Test11/validation2.m File Reference	55
5.28 validation/Validation 2/Test12/validation2.m File Reference	55
5.29 validation/Validation 2/Test13/validation2.m File Reference	55
5.30 validation/Validation 2/Test14/validation2.m File Reference	56
5.31 validation/Validation 2/Test15/validation2.m File Reference	56
5.32 validation/Validation 2/Test16/validation2.m File Reference	56
5.33 validation/Validation 2/Test17/validation2.m File Reference	56
5.34 validation/Validation 2/Test2/validation2.m File Reference	56
5.35 validation/Validation 2/Test3/validation2.m File Reference	56
5.36 validation/Validation 2/Test4/validation2.m File Reference	56
5.37 validation/Validation 2/Test5/validation2.m File Reference	56
5.38 validation/Validation 2/Test6/validation2.m File Reference	56
5.39 validation/Validation 2/Test7/validation2.m File Reference	56
5.40 validation/Validation 2/Test8/validation2.m File Reference	56
5.41 validation/Validation 2/Test9/validation2.m File Reference	56
5.42 validation/Validation 3/Test1/verification3.m File Reference	56
5.43 validation/Validation 3/verification3.m File Reference	56
5.44 validation/Validation 4/test.m File Reference	56
5.45 validation/Validation 4/Test1/test.m File Reference	56
5.46 validation/Validation 4/Test1/validation4.m File Reference	56
5.47 validation/Validation 4/validation4.m File Reference	57
5.48 validation/Validation 7/validation7.m File Reference	57
5.49 validation/Validation 8/Test1/validation8.m File Reference	57
5.50 visupacman.m File Reference	57
5.51 visupacman2.m File Reference	57
5.52 wallsBorder.m File Reference	57
5.52.1 Function Documentation	57
5.52.1.1 wallsBorder()	57
5.53 Wrapper.m File Reference	57

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

handle	7
ModelSED	28
ModelGhost	10
ModelLaby	16
ModelPacman	22
ModelWalls	31
StopCondition	36
ModelCommand	8
Wrapper	41

Chapter 2

Class Index

2.1 Class List

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

handle	7
ModelCommand	8
ModelGhost Contain ghost movement control	10
ModelLaby Class which contains the "fmg" structure of the labyrinth for 2 players	16
ModelPacman Contain Pacman movement control	22
ModelSED Abstract Class who contain the structure of a "fmg" implementation	28
ModelWalls Contain the wall movement command	31
StopCondition Class used to manage shutdown conditions	36
Wrapper	41

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

CreatePituresAndVideo.m	45
CreatePituresAndVideo_textured.m	45
figure_Laby.m	46
main.m	50
matrixAllPossible.m	50
ModelCommand.m	50
ModelGhost.m	54
ModelLaby.m	54
ModelPacman.m	54
ModelSED.m	54
ModelWalls.m	54
setColor.m	55
Simulation.m	55
Simulation2_allpossiblewalls.m	55
StopCondition.m	55
visupacman.m	57
visupacman2.m	57
wallsBorder.m	57
Wrapper.m	57
ModelGenerator/AutomatonSchedulingCreation.m	50
ModelGenerator/AutomatonStrutureLabyCreation.m	50
ModelGenerator/AutomatonWallsConstraintsCreation.m	51
ModelGenerator/generer_lab.m	51
ModelGenerator/modelGenerator.m	51
ModelGenerator/Plan_desumaFunctions_2Players.m	51
ModelGenerator/SaveDESUMAFFile.m	53
ModelGenerator/writeStates.m	53
ModelGenerator/writeTransitions.m	53
validation/Validation 2/Test1/validation2.m	55
validation/Validation 2/Test10/validation2.m	55
validation/Validation 2/Test11/validation2.m	55
validation/Validation 2/Test12/validation2.m	55
validation/Validation 2/Test13/validation2.m	55
validation/Validation 2/Test14/validation2.m	56
validation/Validation 2/Test15/validation2.m	56

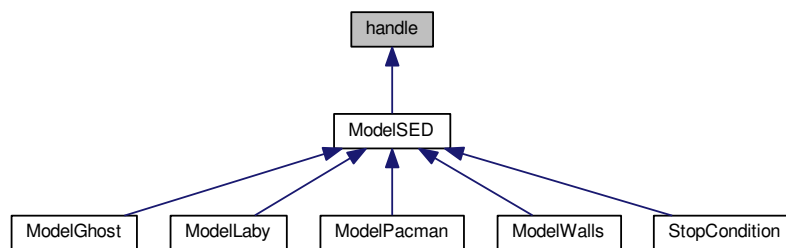
validation/Validation 2/Test16/ validation2.m	56
validation/Validation 2/Test17/ validation2.m	56
validation/Validation 2/Test2/ validation2.m	56
validation/Validation 2/Test3/ validation2.m	56
validation/Validation 2/Test4/ validation2.m	56
validation/Validation 2/Test5/ validation2.m	56
validation/Validation 2/Test6/ validation2.m	56
validation/Validation 2/Test7/ validation2.m	56
validation/Validation 2/Test8/ validation2.m	56
validation/Validation 2/Test9/ validation2.m	56
validation/Validation 3/ verification3.m	56
validation/Validation 3/Test1/ verification3.m	56
validation/Validation 4/ test.m	56
validation/Validation 4/ validation4.m	57
validation/Validation 4/Test1/ test.m	56
validation/Validation 4/Test1/ validation4.m	56
validation/Validation 7/ validation7.m	57
validation/Validation 8/Test1/ validation8.m	57

Chapter 4

Class Documentation

4.1 handle Class Reference

Inheritance diagram for handle:

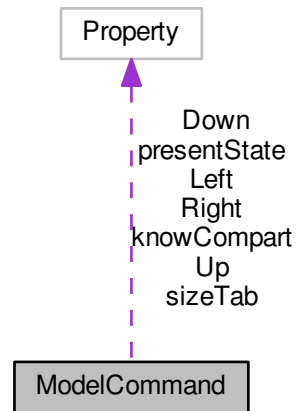


The documentation for this class was generated from the following file:

- [ModelSED.m](#)

4.2 ModelCommand Class Reference

Collaboration diagram for ModelCommand:



Public Member Functions

- [function f](#) (in obj, in [presentState](#))
- [function m](#) (in obj, in [presentState](#), in init)
- [function g](#) (in obj)

Public Attributes

- Property [sizeTab](#)
- Property [knowCompart](#)
- Property [presentState](#)
- Property [Down](#)
- Property [Left](#)
- Property [Up](#)
- Property [Right](#)

4.2.1 Member Function Documentation

4.2.1.1 f()

```

function f (
    in obj,
    in presentState )
  
```

4.2.1.2 g()

```
function g (  
    in obj )
```

4.2.1.3 m()

```
function m (  
    in obj,  
    in presentState,  
    in init )
```

4.2.2 Member Data Documentation

4.2.2.1 Down

Property Down

4.2.2.2 knowCompart

Property knowCompart

4.2.2.3 Left

Property Left

4.2.2.4 presentState

Property presentState

4.2.2.5 Right

Property Right

4.2.2.6 sizeTab

Property sizeTab

4.2.2.7 Up

Property Up

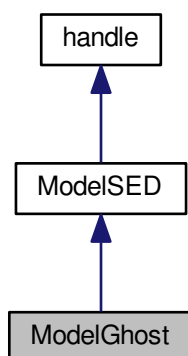
The documentation for this class was generated from the following file:

- [ModelCommand.m](#)

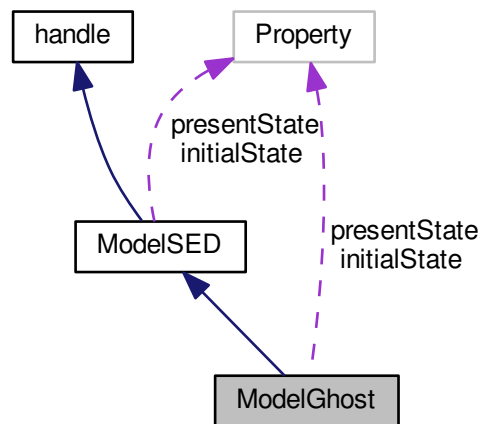
4.3 ModelGhost Class Reference

Contain ghost movement control.

Inheritance diagram for ModelGhost:



Collaboration diagram for ModelGhost:



Public Member Functions

- **function** `ModelGhost` (in `initialValue`)
Class constructor.
- **function** `f` (in `obj`, in `in`, in `in_view`, in `wallsV`, in `wallsH`, in `ghost_position`)
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.
- **virtual** `f` (in `obj`, in `in`)
Compute the evolution of the model.

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 movement control.

You can change here Pacman's command.

Input : Possible ghost's moves [Up Down Left Right]

0 = move not possible ; 1 = move possible

(`Wout{7}`)

Output : Ghost's moves 1 : ghostLeftBut, (Wout(3))
 2 : ghostUpBut, (Wout(1))
 3 : ghostRightBut, (Wout(4))
 4 : ghostDownBut , (Wout(2))
 (Win(4:7) of wrapper)

in: Walls around ghost
 1 up
 2 down
 4 right

in_view: Ghost sees Pacman
 1 Up
 2 Down
 3 Left
 4 Right
 state :

This command $P(D) > P(B) > P(H) > P(G)$

4.3.2 Constructor & Destructor Documentation

4.3.2.1 ModelGhost()

```
function ModelGhost (
    in initialValue )
```

Class constructor.

Parameters

<i>initialValue</i>	Contain the initial state
---------------------	---------------------------

Returns

instance of the [ModelGhost](#) class.

4.3.3 Member Function Documentation

4.3.3.1 f() [1/2]

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

Compute the evolution of the model.

Parameters

<i>obj</i>	The instance who evolve
<i>in</i>	Input needed for the computing

Return values

<i>nextState</i>	The future state of the model
------------------	-------------------------------

Reimplemented in [ModelPacman](#), and [ModelLaby](#).

4.3.3.2 f() [2/2]

```
function f (
    in obj,
    in in,
    in in_view,
    in wallsV,
    in wallsH,
    in ghost_position )
```

Compute the evolution of the command.

It takes more inputs than [ModelSED](#) because ghost can use more information from the laby

Parameters

<i>obj</i>	The instance who evolve
<i>in</i>	Input vector needed for the compute (walls around Ghost)
<i>in_view</i>	Vector of Information about ghost sees Pacman
<i>wallsV</i>	Matrix of vertical Walls
<i>wallsH</i>	Matrix of horizontal Walls
<i>ghost_position</i>	Cartesian vector of Ghost Position

Return values

<i>nextState</i>	The future state of the Ghost command
------------------	---------------------------------------

4.3.3.3 g()

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

Create the outputs.

Parameters

<i>obj</i>	the concerned instance of the class
------------	-------------------------------------

Return values

<i>out</i>	The output who is the command.
------------	--------------------------------

Reimplemented from [ModelSED](#).

4.3.3.4 m()

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

Memory method update the state of the command.

Parameters

<i>obj</i>	The selected instance of the class
<i>nextState</i>	The value of the state need to update
<i>init</i>	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.

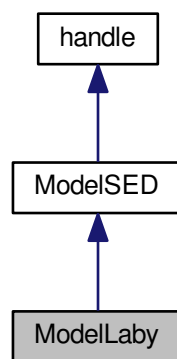
The documentation for this class was generated from the following file:

- [ModelGhost.m](#)

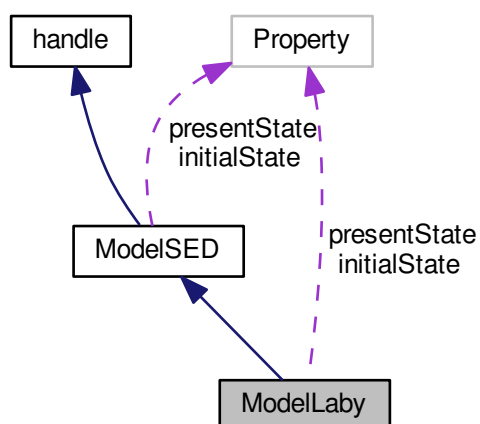
4.4 ModelLaby Class Reference

Class which contains the "fmg" structure of the labyrinth for 2 players.

Inheritance diagram for ModelLaby:



Collaboration diagram for ModelLaby:



Public Member Functions

- [function Modellaby](#) (in wallsV_init, in wallsH_init, in pacman_init, in ghost_init, in escape_init, in caught_init)
Class constructor of.
- [function f](#) (in obj, in in)
Compute the evolution of the model.
- [function m](#) (in obj, in nextState, in init)
Memory method update the state of the command.
- [function g](#) (in obj)
Create the outputs in a 1x9 cell-array.
- [function sameX_position](#) (in obj)
Method to analyze Ghost and Pacman Position.
- [function sameY_position](#) (in obj)
Method to analyze Ghost and Pacman Position.
- [function wallsVBetween](#) (in obj, in obj1, in obj2)
Method to analyze if a Vertical wall is between 2 objects.
- [function wallsHBetween](#) (in obj, in obj1, in obj2)
Method to analyze if a Horizontal wall is between 2 objects.
- [function wallsVBetweenOne](#) (in obj, in obj1, in obj2)
Method to analyze if a Horizontal wall is between 2 objects side by side.
- [function wallsHBetweenOne](#) (in obj, in obj1, in obj2)
Method to analyze if a Horizontal wall is between 2 objects side by side.

Public Attributes

- Property [presentState](#)
*Data Structure of the current state of Labyrinth.
It contains "wallsV", "wallsH" (2 matrix for the walls), "ghost", "pacman" and "escape", a Cartesian position of current position of ghost, pacman and escape.
There is also 3 vectors : 'wallsAroundPacman', 'wallsAroundGhost' and 'ghostSeesPacman' A vector indicating the presence of a wall around the Pacman and ghost 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.

4.4.1 Detailed Description

Class which contains the "fmg" structure of the labyrinth for 2 players.

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 evolve

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Modellaby()

```
function Modellaby (
    in wallsV_init,
    in wallsH_init,
    in pacman_init,
    in ghost_init,
    in escape_init,
    in caught_init )
```

Class constructor of.

Parameters

<i>wallsV_init</i>	Contain a matrix (N, N-1) of Initial Vertical Walls.
<i>wallsH_init</i>	Contain a matrix (N-1, N) of Initial Horizontal Walls.
<i>pacman_init</i>	Contain a vector (x, y) of Initial Position of Pacman.
<i>pacman_init</i>	Contain a vector (x, y) of Initial Position of Ghost.
<i>escape_init</i>	Contain a vector (x, y) of Escape 's Position.
<i>caught_init</i>	Contain a integer of the number of times the Pacman was caught by the ghost.

Returns

instance of the [Modellaby](#) class.

4.4.3 Member Function Documentation

4.4.3.1 f()

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

Compute the evolution of the model.

Parameters

<i>obj</i>	The instance which will evolve.
<i>in</i>	Input needed for the computing.

Return values

<i>nextState</i>	Next instance of the Modellaby class.
------------------	---

Reimplemented from [ModelISED](#).

4.4.3.2 `g()`

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

Create the outputs in a 1x9 cell-array.

Parameters

<i>obj</i>	the concerned instance of the class
------------	-------------------------------------

Return values

<i>out</i>	Constructed output 1x9 cell-array of the model
------------	--

Reimplemented from [ModelSED](#).

4.4.3.3 `m()`

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

Memory method update the state of the command.

Parameters

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

Returns

instance of the class updated

Reimplemented from [ModelSED](#).

4.4.3.4 `sameX_position()`

```
function sameX_position (
    in obj )
```

Method to analyze Ghost and Pacman Position.

Parameters

<i>obj</i>	Current Instance of the Labyrinth 1 if ghost and Pacman are on the same X colon
------------	---

4.4.3.5 sameY_position()

```
function sameY_position (
    in obj )
```

Method to analyze Ghost and Pacman Position.

Parameters

<i>obj</i>	Current Instance of the Labyrinth 1 if ghost and Pacman are on the same Y line
------------	--

4.4.3.6 wallsHBetween()

```
function wallsHBetween (
    in obj,
    in obj1,
    in obj2 )
```

Method to analyze if a Horizontal wall is between 2 objects.

Parameters

<i>obj</i>	Current Instance of the Labyrinth
<i>obj1</i>	Cartesian position of object 1
<i>obj2</i>	Cartesian position of object 2 1 if there No Horizontal wall Between Object 1 and Object 2

4.4.3.7 wallsHBetweenOne()

```
function wallsHBetweenOne (
    in obj,
    in obj1,
    in obj2 )
```

Method to analyze if a Horizontal wall is between 2 objects side by side.

Parameters

<i>obj</i>	Current Instance of the Labyrinth
<i>obj1</i>	Cartesian position of object 1
<i>obj2</i>	Cartesian position of object 2 1 if there No Horizontal wall Between Object 1 and Object 2

4.4.3.8 wallsVBetween()

```
function wallsVBetween (
    in obj,
    in obj1,
    in obj2 )
```

Method to analyze if a Vertical wall is between 2 objects.

Parameters

<i>obj</i>	Current Instance of the Labyrinth
<i>obj1</i>	Cartesian position of object 1
<i>obj2</i>	Cartesian position of object 2 1 if there No Vertical wall Between Object 1 and Object 2

4.4.3.9 wallsVBetweenOne()

```
function wallsVBetweenOne (
    in obj,
    in obj1,
    in obj2 )
```

Method to analyze if a Horizontal wall is between 2 objects side by side.

Parameters

<i>obj</i>	Current Instance of the Labyrinth
<i>obj1</i>	Cartesian position of object 1
<i>obj2</i>	Cartesian position of object 2 1 if there No Horizontal wall Between Object 1 and Object 2

4.4.4 Member Data Documentation

4.4.4.1 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.

4.4.4.2 presentState

Property presentState

Data Structure of the current state of Labyrinth.

It contains "wallsV", "wallsH" (2 matrix for the walls), "ghost", "pacman" and "escape" , a Cartesian position of current position of ghost, pacman and escape.

There is also 3 vectors : 'wallsAroundPacman', 'wallsAroundGhost' and 'ghostSeesPacman' A vector indicating the presence of a wall around the Pacman and ghost for the 4 directions Up Down Left Right.

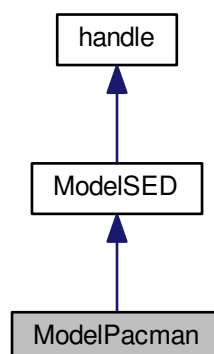
The documentation for this class was generated from the following file:

- [ModelLaby.m](#)

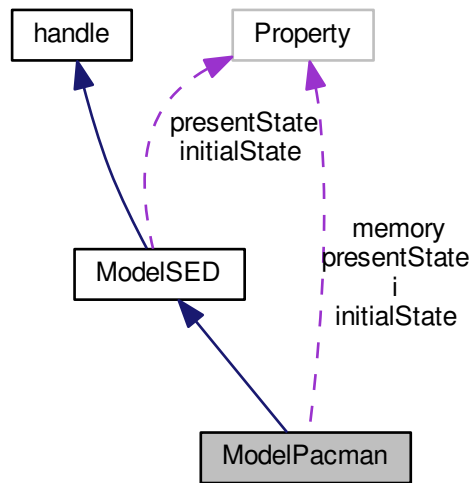
4.5 ModelPacman Class Reference

Contain Pacman movement 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.
- Property `memory`
This is another state who deed to be include.
- Property `i`

4.5.1 Detailed Description

Contain Pacman movement 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 $P(D) > P(B) > P(H) > P(G)$

4.5.2 Constructor & Destructor Documentation

4.5.2.1 ModelPacman()

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

Class constructor.

Parameters

<i>initialValue</i>	Contain the initial state
---------------------	---------------------------

Returns

instance of the [ModelPacman](#) class.

4.5.3 Member Function Documentation

4.5.3.1 f()

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

Compute the evolution of the command.

Parameters

<i>obj</i>	The instance who evolve
<i>in</i>	Input needed for the compute

Return values

<i>nextState</i>	The future state of the Pacman command
------------------	--

Reimplemented from [ModelSED](#).

4.5.3.2 g()

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

Create the outputs.

Parameters

<i>obj</i>	the concerned instance of the class
------------	-------------------------------------

Return values

<i>out</i>	The output who is the command.
------------	--------------------------------

Reimplemented from [ModelSED](#).

4.5.3.3 m()

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

Memory method, update the state of the command.

Parameters

<i>obj</i>	The selected instance of the class
<i>nextState</i>	The value of the state need to update
<i>init</i>	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

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

4.5.4.3 memory

Property memory

This is another state who deed to be include.

4.5.4.4 presentState

Property presentState

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

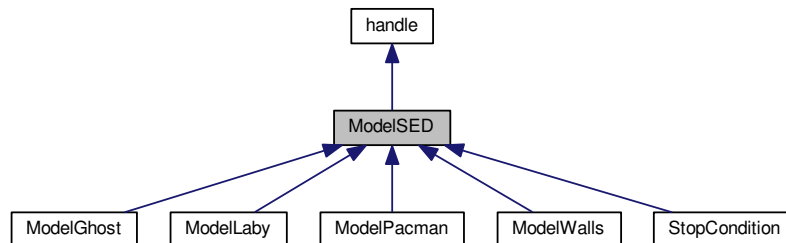
The documentation for this class was generated from the following file:

- [ModelPacman.m](#)

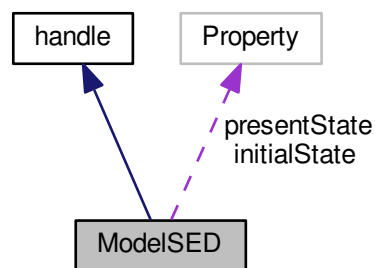
4.6 ModelSED Class Reference

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.

4.6.1 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 evolve

4.6.2 Member Function Documentation

4.6.2.1 f()

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

Compute the evolution of the model.

Parameters

<i>obj</i>	The instance who evolve
<i>in</i>	Input needed for the computing

Return values

<i>nextState</i>	The future state of the model
------------------	-------------------------------

Reimplemented in [ModelPacman](#), and [ModelLaby](#).

4.6.2.2 g()

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

Create the outputs.

Parameters

<i>obj</i>	the concerned instance of the class
------------	-------------------------------------

Return values

<i>out</i>	Constructed output of the model
------------	---------------------------------

Reimplemented in [ModelGhost](#), [ModelLaby](#), [ModelPacman](#), [StopCondition](#), and [ModelWalls](#).

4.6.2.3 m()

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

Memory method update the state of the command.

Parameters

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

Returns

instance of the class updated

Reimplemented in [ModelGhost](#), [ModelPacman](#), [ModelLaby](#), [StopCondition](#), and [ModelWalls](#).

4.6.3 Member Data Documentation**4.6.3.1 initialState**

Property `initialState`

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

4.6.3.2 presentState

Property `presentState`

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

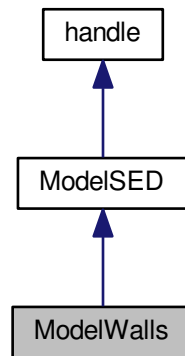
The documentation for this class was generated from the following file:

- [ModelSED.m](#)

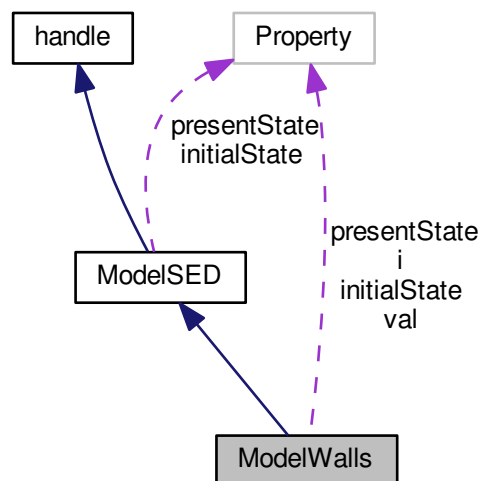
4.7 ModelWalls Class Reference

Contain the wall movement command.

Inheritance diagram for ModelWalls:



Collaboration diagram for ModelWalls:



Public Member Functions

- [function ModelWalls](#) (in `initValue`)
Class constructor.

- `function f` (in `obj`)
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.
- `virtual f` (in `obj`, in `in`)
Compute the evolution of the model.

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.
- Property `i`
- Property `val`

4.7.1 Detailed Description

Contain the wall movement command.

You can change here the order in which the walls move

Input : No need

Output : [UPwalls , RIGHTwalls] This command do the sequence walls Right → walls down

4.7.2 Constructor & Destructor Documentation

4.7.2.1 ModelWalls()

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

Class constructor.

Parameters

<code><i>initValue</i></code>	Contain the initial state
-------------------------------	---------------------------

Returns

instance of the `ModelWalls` class.

4.7.3 Member Function Documentation

4.7.3.1 `f()` [1/2]

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

Compute the evolution of the model.

Parameters

<i>obj</i>	The instance who evolve
<i>in</i>	Input needed for the computing

Return values

<i>nextState</i>	The future state of the model
------------------	-------------------------------

Reimplemented in [ModelPacman](#), and [ModelLaby](#).

4.7.3.2 `f()` [2/2]

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

Compute the evolution of the command.

Parameters

<i>obj</i>	The instance who evolve
<i>in</i>	Input needed for the compute

Return values

<i>nextState</i>	The future state of the walls command
------------------	---------------------------------------

4.7.3.3 `g()`

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

Create the outputs.

Parameters

<i>obj</i>	the concerned instance of the class
------------	-------------------------------------

Return values

<i>out</i>	The output who is the command.
------------	--------------------------------

Reimplemented from [ModelSED](#).

4.7.3.4 m()

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

Memory method update the state of the command.

Parameters

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

Returns

instance of the class updated

Reimplemented from [ModelSED](#).

4.7.4 Member Data Documentation

4.7.4.1 i

Property i

4.7.4.2 initialState

Property initialState

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

4.7.4.3 presentState

Property presentState

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

4.7.4.4 val

Property val

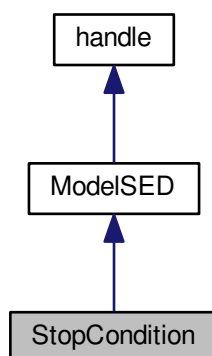
The documentation for this class was generated from the following file:

- [ModelWalls.m](#)

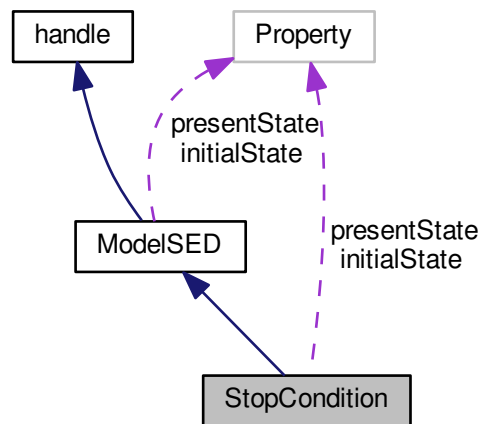
4.8 StopCondition Class Reference

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 caught, in pacmanWallsBreak, in ghostWallsBreak)
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.

4.8.1 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 Caught
3 pacmanWallsBreak
4 ghostWallsBreak

4.8.2 Constructor & Destructor Documentation

4.8.2.1 StopCondition()

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

Class constructor of Instance of [StopCondition](#) Class.

Parameters

<i>initCondition</i>	Structure for the InitialState. It have to contain : 'escape', 'caught', 'pacman', 'ghost and 'numberOfPossibleCaught'
----------------------	--

Returns

instance of the [StopCondition](#) class.

4.8.3 Member Function Documentation

4.8.3.1 f() [1/2]

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

Compute the evolution of the model.

Parameters

<i>obj</i>	The instance who evolve
<i>in</i>	Input needed for the computing

Return values

<i>nextState</i>	The future state of the model
------------------	-------------------------------

Reimplemented in [ModelPacman](#), and [ModelLaby](#).

4.8.3.2 f() [2/2]

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

Compute the evolution of the model.

Parameters

<i>obj</i>	The instance which will evolve.
<i>in</i>	Input needed for the computing.

Returns

Next instance of the [StopCondition](#) class.

4.8.3.3 g()

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

Create the outputs in a vector with 4 parameters.

Parameters

<i>obj</i>	the concerned instance of the class
------------	-------------------------------------

Return values

<i>out</i>	Constructed output vector with 4 parameters of the model
------------	--

Reimplemented from [ModelSED](#).

4.8.3.4 m()

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

Memory method.

Update the state of the command.

Parameters

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

Returns

instance of the class updated

Reimplemented from [ModelSED](#).

4.8.4 Member Data Documentation**4.8.4.1 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.

4.8.4.2 presentState

Property presentState

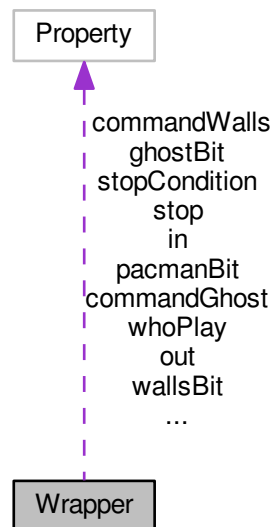
Data Structure of the current state of Labyrinth. It contains "wallsV", "wallsH" (2 matrix for the walls), "escape" and "pacman", a Cartesian position of current position of escape and pacman and 'wallsAroundPacman' A vector indicating the presence of a wall around the Pacman for the 4 directions Up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

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

- [StopCondition.m](#)

4.9 Wrapper Class Reference

Collaboration diagram for Wrapper:



Public Member Functions

- [function Wrapper](#) (in inSize, in outSize, in initLaby, in initWalls, in initPac, in initGhost, in initStop)
- [function updateConnexion](#) (in obj, in indBit, in value)
- [function init](#) (in obj)
- [function orderer](#) (in obj, in vectIn)
- [function get_stop](#) (in obj)
- [function get_out](#) (in obj)

Public Attributes

- Property [wallsBit](#)
- Property [pacmanBit](#)
- Property [ghostBit](#)
- Property [modelLaby](#)
- Property [commandWalls](#)
- Property [commandGhost](#)
- Property [commandPacman](#)
- Property [stopCondition](#)
- Property [in](#)
- Property [out](#)
- Property [stop](#)
- Property [whoPlay](#)

4.9.1 Constructor & Destructor Documentation

4.9.1.1 Wrapper()

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

4.9.2 Member Function Documentation

4.9.2.1 get_out()

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

4.9.2.2 get_stop()

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

4.9.2.3 init()

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

4.9.2.4 orderer()

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

4.9.2.5 updateConnexion()

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

4.9.3 Member Data Documentation

4.9.3.1 commandGhost

Property commandGhost

4.9.3.2 commandPacman

Property commandPacman

4.9.3.3 commandWalls

Property commandWalls

4.9.3.4 ghostBit

Property ghostBit

4.9.3.5 in

Property in

4.9.3.6 modelLaby

Property modelLaby

4.9.3.7 out

Property out

4.9.3.8 pacmanBit

Property pacmanBit

4.9.3.9 stop

Property stop

4.9.3.10 stopCondition

Property stopCondition

4.9.3.11 wallsBit

Property wallsBit

4.9.3.12 whoPlay

Property whoPlay

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)
- [function figure_Laby_OpeningFcn](#) (in hObject, in eventdata, in handles, in varargin)
- [function figure_Laby_OutputFcn](#) (in hObject, in eventdata, in handles)
- [function ui_Callback](#) (in hObject, in eventdata, in handles)
- [function connect_Callback](#) (in hObject, in eventdata, in handles)
- [function createUIPacman](#) (in handles)
- [function createUIGhost](#) (in handles)
- [function createUIWalls](#) (in handles)
- [function createUIEscape](#) (in handles)
- [function updateUI](#) (in handles, in out)
- [function updateUIActiveCammand](#) (in handles)
- [function updateUIButton](#) (in handles)
- [function updateUIPlayer](#) (in handles, in strPlayer, in position)
- [function updateUICaught](#) (in elementToSet, in caughtInt, in stp)
- [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 )
```

5.3.1.2 createUIEscape()

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


5.3.1.3 createUIGhost()

```
function createUIGhost (  
    in handles )
```

5.3.1.4 createUIPacman()

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

5.3.1.5 createUIWalls()

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

5.3.1.6 figure_Laby()

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

5.3.1.7 figure_Laby_OpeningFcn()

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

5.3.1.8 figure_Laby_OutputFcn()

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

5.3.1.9 isOne()

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

5.3.1.10 resetUIConnection()

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

5.3.1.11 ui_Callback()

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

5.3.1.12 updatePresenceDetectorDisplay()

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

5.3.1.13 updateUI()

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

5.3.1.14 updateUIActiveCammand()

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

5.3.1.15 updateUIButton()

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

5.3.1.16 updateUICaught()

```
function updateUICaught (
    in elementToSet,
    in caughtInt,
    in stp )
```

5.3.1.17 updateUIEscape()

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

5.3.1.18 updateUIPlayer()

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

5.3.1.19 updateUIWalls()

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

5.3.1.20 updateUIWallsAround()

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

5.4 main.m File Reference

5.5 matrixAllPossible.m File Reference

5.6 ModelCommand.m File Reference

Classes

- class [ModelCommand](#)

5.7 ModelGenerator/AutomatonSchedulingCreation.m File Reference

Functions

- [function](#) ()

5.7.1 Function Documentation

5.7.1.1 function()

```
function ( )
```

5.8 ModelGenerator/AutomatonStrutureLabyCreation.m File Reference

Functions

- [function AutomatonStrutureLabyCreation](#) (in labySize, in playerPosition, in escapePosition, in playerName)

5.8.1 Function Documentation

5.8.1.1 AutomatonStrutureLabyCreation()

```
function AutomatonStrutureLabyCreation (
    in labySize,
    in playerPosition,
    in escapePosition,
    in playerName )
```

5.9 ModelGenerator/AutomatonWallsConstraintsCreation.m File Reference

Functions

- [function AutomatonWallsConstraintsCreation](#) (in verticalsWalls, in horizontalsWalls, in FirstWallsMove)

5.9.1 Function Documentation

5.9.1.1 AutomatonWallsConstraintsCreation()

```
function AutomatonWallsConstraintsCreation (
    in verticalsWalls,
    in horizontalsWalls,
    in FirstWallsMove )
```

5.10 ModelGenerator/generer_lab.m File Reference

Functions

- [function generer_lab](#) (in Matrice_Horizontale, in Matrice_Verticale)

5.10.1 Function Documentation

5.10.1.1 generer_lab()

```
function generer_lab (
    in Matrice_Horizontale,
    in Matrice_Verticale )
```

5.11 ModelGenerator/modelGenerator.m File Reference

5.12 ModelGenerator/Plan_desumaFunctions_2Players.m File Reference

Functions

- [function writeStates](#) (in prefix, in nbrOfStates, in initialIndice, in markedStatesIndices)
- [function writeTransitions](#) (in prefix, in datas)
- [function SaveDESUMAFFile](#) (in transitionsString, in statesString, in fileName)
- [function AutomatonStrutureLabyCreation](#) (in labySize, in playerPosition, in escapePosition, in playerName)
- [function](#) ()

5.12.1 Function Documentation

5.12.1.1 AutomatonStrutureLabyCreation()

```
function AutomatonStrutureLabyCreation (
    in labySize,
    in playerPosition,
    in escapePosition,
    in playerName )
```

5.12.1.2 function()

```
function ( )
```

5.12.1.3 SaveDESUMAFile()

```
function SaveDESUMAFile (
    in transitionsString,
    in statesString,
    in fileName )
```

5.12.1.4 writeStates()

```
function writeStates (
    in prefix,
    in nbrOfStates,
    in initialIndice,
    in markedStatesIndices )
```

5.12.1.5 writeTransitions()

```
function writeTransitions (
    in prefix,
    in datas )
```

5.13 ModelGenerator/SaveDESUMAFFile.m File Reference

Functions

- [function SaveDESUMAFFile](#) (in transitionsString, in statesString, in fileName)

5.13.1 Function Documentation

5.13.1.1 SaveDESUMAFFile()

```
function SaveDESUMAFFile (
    in transitionsString,
    in statesString,
    in fileName )
```

5.14 ModelGenerator/writeStates.m File Reference

Functions

- [function writeStates](#) (in prefix, in nbrOfStates, in initialIndice, in markedStatesIndices)

5.14.1 Function Documentation

5.14.1.1 writeStates()

```
function writeStates (
    in prefix,
    in nbrOfStates,
    in initialIndice,
    in markedStatesIndices )
```

5.15 ModelGenerator/writeTransitions.m File Reference

Functions

- [function writeTransitions](#) (in prefix, in datas)

5.15.1 Function Documentation

5.15.1.1 writeTransitions()

```
function writeTransitions (
    in prefix,
    in datas )
```

5.16 ModelGhost.m File Reference

Classes

- class [ModelGhost](#)
Contain ghost movement control.

5.17 ModelLaby.m File Reference

Classes

- class [ModelLaby](#)
Class which contains the "fmg" structure of the labyrinth for 2 players.

5.18 ModelPacman.m File Reference

Classes

- class [ModelPacman](#)
Contain Pacman movement control.

5.19 ModelSED.m File Reference

Classes

- class [ModelSED](#)
Abstract Class who contain the structure of a "fmg" implementation.

5.20 ModelWalls.m File Reference

Classes

- class [ModelWalls](#)
Contain the wall movement command.

5.21 setColor.m File Reference

Functions

- [function setColor](#) (in *img*, in *imgRef*, in *colors*, in *indice*)

5.21.1 Function Documentation

5.21.1.1 setColor()

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

5.22 Simulation.m File Reference

5.23 Simulation2_allpossiblewalls.m File Reference

5.24 StopCondition.m File Reference

Classes

- class [StopCondition](#)
Class used to manage shutdown conditions.

5.25 validation/Validation 2/Test1/validation2.m File Reference

5.26 validation/Validation 2/Test10/validation2.m File Reference

5.27 validation/Validation 2/Test11/validation2.m File Reference

5.28 validation/Validation 2/Test12/validation2.m File Reference

5.29 validation/Validation 2/Test13/validation2.m File Reference

- 5.30 validation/Validation 2/Test14/validation2.m File Reference
- 5.31 validation/Validation 2/Test15/validation2.m File Reference
- 5.32 validation/Validation 2/Test16/validation2.m File Reference
- 5.33 validation/Validation 2/Test17/validation2.m File Reference
- 5.34 validation/Validation 2/Test2/validation2.m File Reference
- 5.35 validation/Validation 2/Test3/validation2.m File Reference
- 5.36 validation/Validation 2/Test4/validation2.m File Reference
- 5.37 validation/Validation 2/Test5/validation2.m File Reference
- 5.38 validation/Validation 2/Test6/validation2.m File Reference
- 5.39 validation/Validation 2/Test7/validation2.m File Reference
- 5.40 validation/Validation 2/Test8/validation2.m File Reference
- 5.41 validation/Validation 2/Test9/validation2.m File Reference
- 5.42 validation/Validation 3/Test1/verification3.m File Reference
- 5.43 validation/Validation 3/verification3.m File Reference
- 5.44 validation/Validation 4/test.m File Reference
- 5.45 validation/Validation 4/Test1/test.m File Reference
- 5.46 validation/Validation 4/Test1/validation4.m File Reference

5.47 validation/Validation 4/validation4.m File Reference

5.48 validation/Validation 7/validation7.m File Reference

5.49 validation/Validation 8/Test1/validation8.m File Reference

5.50 visupacman.m File Reference

5.51 visupacman2.m File Reference

5.52 wallsBorder.m File Reference

Functions

- [function wallsBorder](#) (in walls)

5.52.1 Function Documentation

5.52.1.1 wallsBorder()

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

5.53 Wrapper.m File Reference

Classes

- class [Wrapper](#)

Index

- AutomatonSchedulingCreation.m
 - function, [50](#)
- AutomatonStrutureLabyCreation
 - AutomatonStrutureLabyCreation.m, [50](#)
 - Plan_desumaFunctions_2Players.m, [52](#)
- AutomatonStrutureLabyCreation.m
 - AutomatonStrutureLabyCreation, [50](#)
- AutomatonWallsConstraintsCreation
 - AutomatonWallsConstraintsCreation.m, [51](#)
- AutomatonWallsConstraintsCreation.m
 - AutomatonWallsConstraintsCreation, [51](#)
- commandGhost
 - Wrapper, [43](#)
- commandPacman
 - Wrapper, [43](#)
- commandWalls
 - Wrapper, [43](#)
- connect_Callback
 - figure_Laby.m, [46](#)
- CreatePituresAndVideo
 - CreatePituresAndVideo.m, [45](#)
- CreatePituresAndVideo.m, [45](#)
 - CreatePituresAndVideo, [45](#)
- CreatePituresAndVideo_textured
 - CreatePituresAndVideo_textured.m, [45](#)
- CreatePituresAndVideo_textured.m, [45](#)
 - CreatePituresAndVideo_textured, [45](#)
- createUIEscape
 - figure_Laby.m, [46](#)
- createUIGhost
 - figure_Laby.m, [46](#)
- createUIPacman
 - figure_Laby.m, [47](#)
- createUIWalls
 - figure_Laby.m, [47](#)
- Down
 - ModelCommand, [9](#)
- f
 - ModelCommand, [8](#)
 - ModelGhost, [12](#), [14](#)
 - ModelLaby, [18](#)
 - ModelPacman, [24](#)
 - ModelSED, [29](#)
 - ModelWalls, [33](#)
 - StopCondition, [38](#), [39](#)
- figure_Laby
 - figure_Laby.m, [47](#)
- figure_Laby.m, [46](#)
 - connect_Callback, [46](#)
 - createUIEscape, [46](#)
 - createUIGhost, [46](#)
 - createUIPacman, [47](#)
 - createUIWalls, [47](#)
 - figure_Laby, [47](#)
 - figure_Laby_OpeningFcn, [47](#)
 - figure_Laby_OutputFcn, [47](#)
 - isOne, [47](#)
 - resetUIConnection, [48](#)
 - ui_Callback, [48](#)
 - updatePresenceDetectorDisplay, [48](#)
 - updateUIActiveCammand, [48](#)
 - updateUIButton, [48](#)
 - updateUICaught, [49](#)
 - updateUIEscape, [49](#)
 - updateUIPlayer, [49](#)
 - updateUIWalls, [49](#)
 - updateUIWallsAround, [49](#)
 - updateUI, [48](#)
- figure_Laby_OpeningFcn
 - figure_Laby.m, [47](#)
- figure_Laby_OutputFcn
 - figure_Laby.m, [47](#)
- function
 - AutomatonSchedulingCreation.m, [50](#)
 - Plan_desumaFunctions_2Players.m, [52](#)
- g
 - ModelCommand, [8](#)
 - ModelGhost, [14](#)
 - ModelLaby, [18](#)
 - ModelPacman, [26](#)
 - ModelSED, [29](#)
 - ModelWalls, [33](#)
 - StopCondition, [39](#)
- generer_lab
 - generer_lab.m, [51](#)
- generer_lab.m
 - generer_lab, [51](#)
- get_out
 - Wrapper, [42](#)
- get_stop
 - Wrapper, [42](#)
- ghostBit
 - Wrapper, [43](#)
- handle, [7](#)

- i
 - ModelPacman, 27
 - ModelWalls, 35
- in
 - Wrapper, 44
- init
 - Wrapper, 42
- initialState
 - ModelGhost, 15
 - ModelLaby, 21
 - ModelPacman, 27
 - ModelSED, 30
 - ModelWalls, 35
 - StopCondition, 40
- isOne
 - figure_Laby.m, 47
- knowCompart
 - ModelCommand, 9
- Left
 - ModelCommand, 9
- m
 - ModelCommand, 9
 - ModelGhost, 15
 - ModelLaby, 19
 - ModelPacman, 26
 - ModelSED, 30
 - ModelWalls, 35
 - StopCondition, 40
- main.m, 50
- matrixAllPossible.m, 50
- memory
 - ModelPacman, 27
- ModelCommand, 8
 - Down, 9
 - f, 8
 - g, 8
 - knowCompart, 9
 - Left, 9
 - m, 9
 - presentState, 9
 - Right, 9
 - sizeTab, 9
 - Up, 10
- ModelCommand.m, 50
- ModelGenerator/AutomatonSchedulingCreation.m, 50
- ModelGenerator/AutomatonStrutureLabyCreation.m, 50
- ModelGenerator/AutomatonWallsContraintsCreation.m, 51
- ModelGenerator/Plan_desumaFunctions_2Players.m, 51
- ModelGenerator/SaveDESUMAFFile.m, 53
- ModelGenerator/generer_lab.m, 51
- ModelGenerator/modelGenerator.m, 51
- ModelGenerator/writeStates.m, 53
- ModelGenerator/writeTransitions.m, 53
- ModelGhost, 10
 - f, 12, 14
 - g, 14
 - initialState, 15
 - m, 15
 - ModelGhost, 12
 - presentState, 15
- ModelGhost.m, 54
- ModelLaby, 16
 - f, 18
 - g, 18
 - initialState, 21
 - m, 19
 - ModelLaby, 17
 - presentState, 21
 - sameX_position, 19
 - sameY_position, 20
 - wallsHBetween, 20
 - wallsHBetweenOne, 20
 - wallsVBetween, 21
 - wallsVBetweenOne, 21
- modelLaby
 - Wrapper, 44
- ModelLaby.m, 54
- ModelPacman, 22
 - f, 24
 - g, 26
 - i, 27
 - initialState, 27
 - m, 26
 - memory, 27
 - ModelPacman, 24
 - presentState, 27
- ModelPacman.m, 54
- ModelSED.m, 54
- ModelSED, 28
 - f, 29
 - g, 29
 - initialState, 30
 - m, 30
 - presentState, 30
- ModelWalls, 31
 - f, 33
 - g, 33
 - i, 35
 - initialState, 35
 - m, 35
 - ModelWalls, 32
 - presentState, 35
 - val, 36
- ModelWalls.m, 54
- orderer
 - Wrapper, 43
- out
 - Wrapper, 44
- pacmanBit
 - Wrapper, 44
- Plan_desumaFunctions_2Players.m

- AutomatonStrutureLabyCreation, [52](#)
- function, [52](#)
- SaveDESUMAFFile, [52](#)
- writeStates, [52](#)
- writeTransitions, [52](#)
- presentState
 - ModelCommand, [9](#)
 - ModelGhost, [15](#)
 - ModelLaby, [21](#)
 - ModelPacman, [27](#)
 - ModelSED, [30](#)
 - ModelWalls, [35](#)
 - StopCondition, [40](#)
- resetUIConnection
 - figure_Laby.m, [48](#)
- Right
 - ModelCommand, [9](#)
- sameX_position
 - ModelLaby, [19](#)
- sameY_position
 - ModelLaby, [20](#)
- SaveDESUMAFFile
 - Plan_desumaFunctions_2Players.m, [52](#)
 - SaveDESUMAFFile.m, [53](#)
- SaveDESUMAFFile.m
 - SaveDESUMAFFile, [53](#)
- setColor
 - setColor.m, [55](#)
- setColor.m, [55](#)
 - setColor, [55](#)
- Simulation.m, [55](#)
- Simulation2_allpossiblewalls.m, [55](#)
- sizeTab
 - ModelCommand, [9](#)
- stop
 - Wrapper, [44](#)
- StopCondition, [36](#)
 - f, [38](#), [39](#)
 - g, [39](#)
 - initialState, [40](#)
 - m, [40](#)
 - presentState, [40](#)
 - StopCondition, [38](#)
- stopCondition
 - Wrapper, [44](#)
- StopCondition.m, [55](#)
- ui_Callback
 - figure_Laby.m, [48](#)
- Up
 - ModelCommand, [10](#)
- updateConnexion
 - Wrapper, [43](#)
- updatePresenceDetectorDisplay
 - figure_Laby.m, [48](#)
- updateUIActiveCammand
 - figure_Laby.m, [48](#)
- updateUIButton
 - figure_Laby.m, [48](#)
- updateUICaught
 - figure_Laby.m, [49](#)
- updateUIEscape
 - figure_Laby.m, [49](#)
- updateUIPlayer
 - figure_Laby.m, [49](#)
- updateUIWalls
 - figure_Laby.m, [49](#)
- updateUIWallsAround
 - figure_Laby.m, [49](#)
- updateUI
 - figure_Laby.m, [48](#)
- val
 - ModelWalls, [36](#)
- validation/Validation 2/Test1/validation2.m, [55](#)
- validation/Validation 2/Test10/validation2.m, [55](#)
- validation/Validation 2/Test11/validation2.m, [55](#)
- validation/Validation 2/Test12/validation2.m, [55](#)
- validation/Validation 2/Test13/validation2.m, [55](#)
- validation/Validation 2/Test14/validation2.m, [56](#)
- validation/Validation 2/Test15/validation2.m, [56](#)
- validation/Validation 2/Test16/validation2.m, [56](#)
- validation/Validation 2/Test17/validation2.m, [56](#)
- validation/Validation 2/Test2/validation2.m, [56](#)
- validation/Validation 2/Test3/validation2.m, [56](#)
- validation/Validation 2/Test4/validation2.m, [56](#)
- validation/Validation 2/Test5/validation2.m, [56](#)
- validation/Validation 2/Test6/validation2.m, [56](#)
- validation/Validation 2/Test7/validation2.m, [56](#)
- validation/Validation 2/Test8/validation2.m, [56](#)
- validation/Validation 2/Test9/validation2.m, [56](#)
- validation/Validation 3/Test1/verification3.m, [56](#)
- validation/Validation 3/verification3.m, [56](#)
- validation/Validation 4/Test1/test.m, [56](#)
- validation/Validation 4/Test1/validation4.m, [56](#)
- validation/Validation 4/test.m, [56](#)
- validation/Validation 4/validation4.m, [57](#)
- validation/Validation 7/validation7.m, [57](#)
- validation/Validation 8/Test1/validation8.m, [57](#)
- visupacman.m, [57](#)
- visupacman2.m, [57](#)
- wallsBit
 - Wrapper, [44](#)
- wallsBorder
 - wallsBorder.m, [57](#)
- wallsBorder.m, [57](#)
 - wallsBorder, [57](#)
- wallsHBetween
 - ModelLaby, [20](#)
- wallsHBetweenOne
 - ModelLaby, [20](#)
- wallsVBetween
 - ModelLaby, [21](#)
- wallsVBetweenOne
 - ModelLaby, [21](#)

- whoPlay
 - Wrapper, [44](#)
- Wrapper, [41](#)
 - commandGhost, [43](#)
 - commandPacman, [43](#)
 - commandWalls, [43](#)
 - get_out, [42](#)
 - get_stop, [42](#)
 - ghostBit, [43](#)
 - in, [44](#)
 - init, [42](#)
 - modelLaby, [44](#)
 - orderer, [43](#)
 - out, [44](#)
 - pacmanBit, [44](#)
 - stop, [44](#)
 - stopCondition, [44](#)
 - updateConnexion, [43](#)
 - wallsBit, [44](#)
 - whoPlay, [44](#)
 - Wrapper, [42](#)
- Wrapper.m, [57](#)
- writeStates
 - Plan_desumaFunctions_2Players.m, [52](#)
 - writeStates.m, [53](#)
- writeStates.m
 - writeStates, [53](#)
- writeTransitions
 - Plan_desumaFunctions_2Players.m, [52](#)
 - writeTransitions.m, [54](#)
- writeTransitions.m
 - writeTransitions, [54](#)