

Simulation Laby

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	7
4.2.1	Member Function Documentation	8
4.2.1.1	f()	8
4.2.1.2	g()	8
4.2.1.3	m()	8
4.2.2	Member Data Documentation	8
4.2.2.1	Down	8
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	9
4.2.2.7	Up	9
4.3	ModelGhost Class Reference	10

4.3.1	Detailed Description	11
4.3.2	Constructor & Destructor Documentation	11
4.3.2.1	ModelGhost()	11
4.3.3	Member Function Documentation	12
4.3.3.1	f() [1/2]	12
4.3.3.2	f() [2/2]	12
4.3.3.3	g()	13
4.3.3.4	m()	13
4.3.4	Member Data Documentation	14
4.3.4.1	initialState	14
4.3.4.2	presentState	14
4.4	ModelLaby Class Reference	14
4.4.1	Detailed Description	15
4.4.2	Constructor & Destructor Documentation	15
4.4.2.1	ModelLaby()	16
4.4.3	Member Function Documentation	16
4.4.3.1	f()	16
4.4.3.2	g()	17
4.4.3.3	m()	17
4.4.3.4	sameX_position()	17
4.4.3.5	sameY_position()	18
4.4.3.6	wallsHBetween()	18
4.4.3.7	wallsHBetweenOne()	18
4.4.3.8	wallsVBetween()	19
4.4.3.9	wallsVBetweenOne()	19
4.4.4	Member Data Documentation	19
4.4.4.1	initialState	19
4.4.4.2	presentState	20
4.5	ModelPacman Class Reference	20
4.5.1	Detailed Description	21

4.5.2	Constructor & Destructor Documentation	21
4.5.2.1	ModelPacman()	21
4.5.3	Member Function Documentation	21
4.5.3.1	f()	21
4.5.3.2	g()	22
4.5.3.3	m()	22
4.5.4	Member Data Documentation	23
4.5.4.1	i	23
4.5.4.2	initialState	23
4.5.4.3	memory	23
4.5.4.4	presentState	23
4.6	ModelSED Class Reference	24
4.6.1	Detailed Description	24
4.6.2	Member Function Documentation	24
4.6.2.1	f()	24
4.6.2.2	g()	25
4.6.2.3	m()	25
4.6.3	Member Data Documentation	26
4.6.3.1	initialState	26
4.6.3.2	presentState	26
4.7	ModelWalls Class Reference	26
4.7.1	Detailed Description	27
4.7.2	Constructor & Destructor Documentation	27
4.7.2.1	ModelWalls()	27
4.7.3	Member Function Documentation	28
4.7.3.1	f() [1/2]	28
4.7.3.2	f() [2/2]	28
4.7.3.3	g()	28
4.7.3.4	m()	30
4.7.4	Member Data Documentation	30

4.7.4.1	i	30
4.7.4.2	initialState	30
4.7.4.3	presentState	31
4.7.4.4	val	31
4.8	StopCondition Class Reference	31
4.8.1	Constructor & Destructor Documentation	32
4.8.1.1	StopCondition()	32
4.8.2	Member Function Documentation	32
4.8.2.1	f() [1/2]	32
4.8.2.2	f() [2/2]	32
4.8.2.3	g()	33
4.8.2.4	m()	33
4.8.3	Member Data Documentation	33
4.8.3.1	initialState	34
4.8.3.2	presentState	34
4.9	Wrapper Class Reference	34
4.9.1	Constructor & Destructor Documentation	34
4.9.1.1	Wrapper()	35
4.9.2	Member Function Documentation	35
4.9.2.1	get_out()	35
4.9.2.2	get_stop()	35
4.9.2.3	init()	35
4.9.2.4	orderer()	35
4.9.2.5	updateConnexion()	35
4.9.3	Member Data Documentation	36
4.9.3.1	commandGhost	36
4.9.3.2	commandPacman	36
4.9.3.3	commandWalls	36
4.9.3.4	ghostBit	36
4.9.3.5	in	36
4.9.3.6	modelLaby	36
4.9.3.7	out	36
4.9.3.8	pacmanBit	37
4.9.3.9	stop	37
4.9.3.10	stopCondition	37
4.9.3.11	wallsBit	37
4.9.3.12	whoPlay	37

5 File Documentation	39
5.1 CreatePituresAndVideo.m File Reference	39
5.1.1 Function Documentation	39
5.1.1.1 CreatePituresAndVideo()	39
5.2 CreatePituresAndVideo_textured.m File Reference	39
5.2.1 Function Documentation	39
5.2.1.1 CreatePituresAndVideo_textured()	40
5.3 figure_Laby.m File Reference	40
5.3.1 Function Documentation	40
5.3.1.1 connect_Callback()	40
5.3.1.2 createUIEscape()	40
5.3.1.3 createUIGhost()	41
5.3.1.4 createUIPacman()	41
5.3.1.5 createUIWalls()	41
5.3.1.6 figure_Laby()	41
5.3.1.7 figure_Laby_OpeningFcn()	41
5.3.1.8 figure_Laby_OutputFcn()	41
5.3.1.9 isOne()	42
5.3.1.10 resetUIConnection()	42
5.3.1.11 ui_Callback()	42
5.3.1.12 updatePresenceDetectorDisplay()	42
5.3.1.13 updateUI()	42
5.3.1.14 updateUIActiveCammand()	42
5.3.1.15 updateUIButton()	43
5.3.1.16 updateUICaught()	43
5.3.1.17 updateUIEscape()	43
5.3.1.18 updateUIPlayer()	43
5.3.1.19 updateUIWalls()	43
5.3.1.20 updateUIWallsAround()	43
5.4 LabyMenu.m File Reference	44

5.4.1	Function Documentation	44
5.4.1.1	LabyMenu()	44
5.4.1.2	LabyMenu_OpeningFcn()	44
5.4.1.3	LabyMenu_OutputFcn()	44
5.4.1.4	OneEasy_Callback()	45
5.4.1.5	OneHard_Callback()	45
5.4.1.6	OneMedium_Callback()	45
5.4.1.7	slider1_Callback()	45
5.4.1.8	slider1_CreateFcn()	45
5.4.1.9	TwoEasy_Callback()	45
5.4.1.10	TwoHard_Callback()	46
5.4.1.11	TwoMedium_Callback()	46
5.5	main.m File Reference	46
5.6	matrixAllPossible.m File Reference	46
5.7	ModelCommand.m File Reference	46
5.8	ModelGenerator/AutomatonSchedulingCreation.m File Reference	46
5.8.1	Function Documentation	46
5.8.1.1	function()	46
5.9	ModelGenerator/AutomatonStrutureLabyCreation.m File Reference	47
5.9.1	Function Documentation	47
5.9.1.1	AutomatonStrutureLabyCreation()	47
5.10	ModelGenerator/AutomatonWallsConstraintsCreation.m File Reference	47
5.10.1	Function Documentation	47
5.10.1.1	AutomatonWallsConstraintsCreation()	47
5.11	ModelGenerator/generer_lab.m File Reference	47
5.11.1	Function Documentation	48
5.11.1.1	generer_lab()	48
5.12	ModelGenerator/modelGenerator.m File Reference	48
5.13	ModelGenerator/Plan_desumaFunctions_2Players.m File Reference	48
5.13.1	Function Documentation	48

5.13.1.1 AutomatonStrutureLabyCreation()	48
5.13.1.2 function()	48
5.13.1.3 SaveDESUMAFile()	49
5.13.1.4 writeStates()	49
5.13.1.5 writeTransitions()	49
5.14 ModelGenerator/SaveDESUMAFile.m File Reference	49
5.14.1 Function Documentation	49
5.14.1.1 SaveDESUMAFile()	49
5.15 ModelGenerator/writeStates.m File Reference	49
5.15.1 Function Documentation	50
5.15.1.1 writeStates()	50
5.16 ModelGenerator/writeTransitions.m File Reference	50
5.16.1 Function Documentation	50
5.16.1.1 writeTransitions()	50
5.17 ModelGhost.m File Reference	50
5.18 ModelLaby.m File Reference	51
5.19 ModelPacman.m File Reference	51
5.19.1 Detailed Description	51
5.20 ModelSED.m File Reference	51
5.20.1 Detailed Description	52
5.21 ModelWalls.m File Reference	52
5.21.1 Detailed Description	52
5.22 setColor.m File Reference	52
5.22.1 Function Documentation	52
5.22.1.1 setColor()	52
5.23 Simulation.m File Reference	53
5.24 Simulation2_allpossiblewalls.m File Reference	53
5.25 StopCondition.m File Reference	53
5.26 validation/Validation 2/Test1/validation2.m File Reference	53
5.27 validation/Validation 2/Test10/validation2.m File Reference	53

5.28 validation/Validation 2/Test11/validation2.m File Reference	53
5.29 validation/Validation 2/Test12/validation2.m File Reference	53
5.30 validation/Validation 2/Test13/validation2.m File Reference	53
5.31 validation/Validation 2/Test14/validation2.m File Reference	53
5.32 validation/Validation 2/Test15/validation2.m File Reference	53
5.33 validation/Validation 2/Test16/validation2.m File Reference	53
5.34 validation/Validation 2/Test17/validation2.m File Reference	53
5.35 validation/Validation 2/Test2/validation2.m File Reference	53
5.36 validation/Validation 2/Test3/validation2.m File Reference	53
5.37 validation/Validation 2/Test4/validation2.m File Reference	53
5.38 validation/Validation 2/Test5/validation2.m File Reference	53
5.39 validation/Validation 2/Test6/validation2.m File Reference	54
5.40 validation/Validation 2/Test7/validation2.m File Reference	54
5.41 validation/Validation 2/Test8/validation2.m File Reference	54
5.42 validation/Validation 2/Test9/validation2.m File Reference	54
5.43 validation/Validation 3/Test1/verification3.m File Reference	54
5.44 validation/Validation 3/verification3.m File Reference	54
5.45 validation/Validation 4/test.m File Reference	54
5.46 validation/Validation 4/Test1/test.m File Reference	54
5.47 validation/Validation 4/Test1/validation4.m File Reference	54
5.48 validation/Validation 4/validation4.m File Reference	54
5.49 validation/Validation 7/validation7.m File Reference	54
5.50 validation/Validation 8/Test1/validation8.m File Reference	54
5.51 visupacman.m File Reference	54
5.52 visupacman2.m File Reference	54
5.53 wallsBorder.m File Reference	54
5.53.1 Function Documentation	54
5.53.1.1 wallsBorder()	55
5.54 Wrapper.m File Reference	55

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

handle	7
ModelSED	24
ModelGhost	10
ModelLaby	14
ModelPacman	20
ModelWalls	26
StopCondition	31
ModelCommand	7
Wrapper	34

Chapter 2

Class Index

2.1 Class List

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

handle	7
ModelCommand	7
ModelGhost MODELGhost Summary of this class goes here 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 10	
ModelLaby Class which contains the "fmg" structure of the labyrinth for 2 players	14
ModelPacman 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) 20	
ModelSED State : minimal information necessary who evolve	24
ModelWalls This command do the sequence walls Right -> walls down 26	
StopCondition	31
Wrapper	34

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

CreatePituresAndVideo.m	39
CreatePituresAndVideo_textured.m	39
figure_Laby.m	40
LabyMenu.m	44
main.m	46
matrixAllPossible.m	46
ModelCommand.m	46
ModelGhost.m	50
ModelLaby.m	51
ModelPacman.m	
Command of the Pacman's moves 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)	
51	
ModelSED.m	
abstract Class who contain the structure of a "fmg" implementation Input : necessary information	
for compute the next state of the model	51
ModelWalls.m	
Command of the walls' move Input : No need	
Output : [UPwalls , RIGHTwalls]	52
setColor.m	52
Simulation.m	53
Simulation2_allpossiblewalls.m	53
StopCondition.m	53
visupacman.m	54
visupacman2.m	54
wallsBorder.m	54
Wrapper.m	55
ModelGenerator/AutomatonSchedulingCreation.m	46

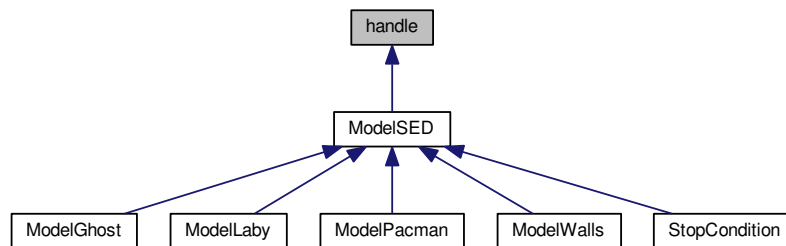
ModelGenerator/ AutomatonStrutureLabyCreation.m	47
ModelGenerator/ AutomatonWallsConstraintsCreation.m	47
ModelGenerator/ generer_lab.m	47
ModelGenerator/ modelGenerator.m	48
ModelGenerator/ Plan_desumaFunctions_2Players.m	48
ModelGenerator/ SaveDESUMAFFile.m	49
ModelGenerator/ writeStates.m	49
ModelGenerator/ writeTransitions.m	50
validation/Validation 2/Test1/ validation2.m	53
validation/Validation 2/Test10/ validation2.m	53
validation/Validation 2/Test11/ validation2.m	53
validation/Validation 2/Test12/ validation2.m	53
validation/Validation 2/Test13/ validation2.m	53
validation/Validation 2/Test14/ validation2.m	53
validation/Validation 2/Test15/ validation2.m	53
validation/Validation 2/Test16/ validation2.m	53
validation/Validation 2/Test17/ validation2.m	53
validation/Validation 2/Test2/ validation2.m	53
validation/Validation 2/Test3/ validation2.m	53
validation/Validation 2/Test4/ validation2.m	53
validation/Validation 2/Test5/ validation2.m	53
validation/Validation 2/Test6/ validation2.m	54
validation/Validation 2/Test7/ validation2.m	54
validation/Validation 2/Test8/ validation2.m	54
validation/Validation 2/Test9/ validation2.m	54
validation/Validation 3/ verification3.m	54
validation/Validation 3/Test1/ verification3.m	54
validation/Validation 4/ test.m	54
validation/Validation 4/ validation4.m	54
validation/Validation 4/Test1/ test.m	54
validation/Validation 4/Test1/ validation4.m	54
validation/Validation 7/ validation7.m	54
validation/Validation 8/Test1/ validation8.m	54

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

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

MODELGhost Summary of this class goes here

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)

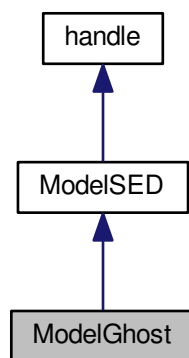
in: Walls around ghost

1 up

2 down

.

Inheritance 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

MODELGhost Summary of this class goes here

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.

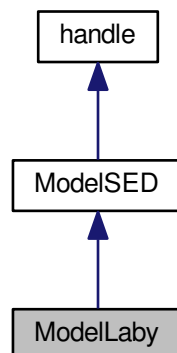
The documentation for this class was generated from the following file:

- [ModelGhost.m](#)

4.4 ModelLaby Class Reference

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

Inheritance diagram for ModelLaby:



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

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

Input : Walls around Pacman

1 up

2 down

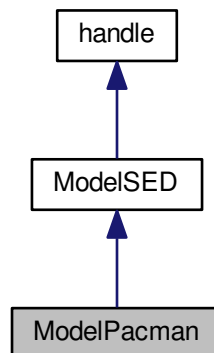
3 left

4 right

This command do the sequence $P(D) > P(B) > P(H) > P(G)$

.

Inheritance 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

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.

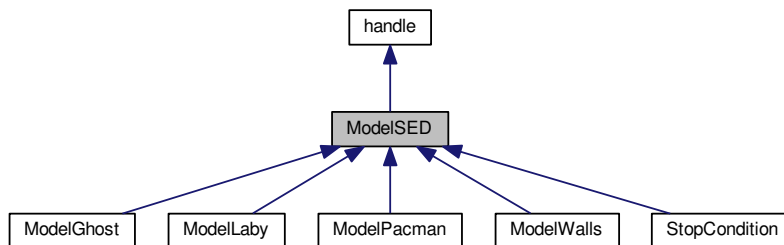
The documentation for this class was generated from the following file:

- [ModelPacman.m](#)

4.6 ModelSED Class Reference

State : minimal information necessary who evolve.

Inheritance 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

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](#), [ModelWalls](#), and [StopCondition](#).

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](#), [ModelWalls](#), and [StopCondition](#).

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.

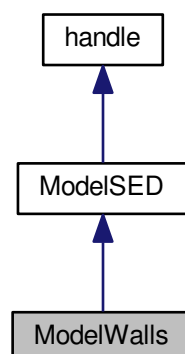
The documentation for this class was generated from the following file:

- [ModelSED.m](#)

4.7 ModelWalls Class Reference

This command do the sequence walls Right → walls down
.

Inheritance 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

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>initValue</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

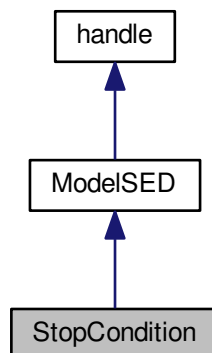
Property val

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

- [ModelWalls.m](#)

4.8 StopCondition Class Reference

Inheritance diagram for StopCondition:



Public Member Functions

- [function StopCondition](#) (in initCondition)
- [function f](#) (in obj, in noEscape, in caught, in pacmanWallsBreak, in ghostWallsBreak)
- [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](#)

4.8.1 Constructor & Destructor Documentation

4.8.1.1 StopCondition()

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

4.8.2 Member Function Documentation

4.8.2.1 f() [1/2]

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

4.8.2.2 f() [2/2]

```
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](#).

4.8.2.3 g()

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

Create the outputs.

Parameters

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

Return values

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

Reimplemented from [ModelSED](#).

4.8.2.4 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](#).

4.8.3 Member Data Documentation

4.8.3.1 initialState

Property `initialState`

4.8.3.2 presentState

Property `presentState`

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

- [StopCondition.m](#)

4.9 Wrapper Class Reference

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 LabyMenu.m File Reference

Functions

- [function LabyMenu](#) (in varargin)
- [function LabyMenu_OpeningFcn](#) (in hObject, in eventdata, in handles, in varargin)
- [function LabyMenu_OutputFcn](#) (in hObject, in eventdata, in handles)
- [function OneEasy_Callback](#) (in hObject, in eventdata, in handles)
- [function TwoHard_Callback](#) (in hObject, in eventdata, in handles)
- [function TwoMedium_Callback](#) (in hObject, in eventdata, in handles)
- [function TwoEasy_Callback](#) (in hObject, in eventdata, in handles)
- [function OneMedium_Callback](#) (in hObject, in eventdata, in handles)
- [function OneHard_Callback](#) (in hObject, in eventdata, in handles)
- [function slider1_Callback](#) (in hObject, in eventdata, in handles)
- [function slider1_CreateFcn](#) (in hObject, in eventdata, in handles)

5.4.1 Function Documentation

5.4.1.1 LabyMenu()

```
function LabyMenu (  
    in varargin )
```

5.4.1.2 LabyMenu_OpeningFcn()

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

5.4.1.3 LabyMenu_OutputFcn()

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

5.4.1.4 OneEasy_Callback()

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

5.4.1.5 OneHard_Callback()

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

5.4.1.6 OneMedium_Callback()

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

5.4.1.7 slider1_Callback()

```
function slider1_Callback (
    in hObject,
    in eventdata,
    in handles )
```

5.4.1.8 slider1_CreateFcn()

```
function slider1_CreateFcn (
    in hObject,
    in eventdata,
    in handles )
```

5.4.1.9 TwoEasy_Callback()

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

5.4.1.10 TwoHard_Callback()

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

5.4.1.11 TwoMedium_Callback()

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

5.5 main.m File Reference

5.6 matrixAllPossible.m File Reference

5.7 ModelCommand.m File Reference

Classes

- class [ModelCommand](#)

5.8 ModelGenerator/AutomatonSchedulingCreation.m File Reference

Functions

- [function](#) ()

5.8.1 Function Documentation

5.8.1.1 function()

```
function ( )
```


5.9 ModelGenerator/AutomatonStrutureLabyCreation.m File Reference

Functions

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

5.9.1 Function Documentation

5.9.1.1 AutomatonStrutureLabyCreation()

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

5.10 ModelGenerator/AutomatonWallsConstraintsCreation.m File Reference

Functions

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

5.10.1 Function Documentation

5.10.1.1 AutomatonWallsConstraintsCreation()

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

5.11 ModelGenerator/generer_lab.m File Reference

Functions

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

5.11.1 Function Documentation

5.11.1.1 generer_lab()

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

5.12 ModelGenerator/modelGenerator.m File Reference

5.13 ModelGenerator/Plan_desumaFunctions_2Players.m File Reference

Functions

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

5.13.1 Function Documentation

5.13.1.1 AutomatonStrutureLabyCreation()

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

5.13.1.2 function()

```
function ( )
```

5.13.1.3 SaveDESUMAFFile()

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

5.13.1.4 writeStates()

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

5.13.1.5 writeTransitions()

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

5.14 ModelGenerator/SaveDESUMAFFile.m File Reference

Functions

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

5.14.1 Function Documentation

5.14.1.1 SaveDESUMAFFile()

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

5.15 ModelGenerator/writeStates.m File Reference

Functions

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

5.15.1 Function Documentation

5.15.1.1 writeStates()

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

5.16 ModelGenerator/writeTransitions.m File Reference

Functions

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

5.16.1 Function Documentation

5.16.1.1 writeTransitions()

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

5.17 ModelGhost.m File Reference

Classes

- class [ModelGhost](#)

MODELGhost Summary of this class goes here

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

.

5.18 ModelLaby.m File Reference

Classes

- class [ModelLaby](#)

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

5.19 ModelPacman.m File Reference

Command of the Pacman's moves 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)

Classes

- class [ModelPacman](#)

Input : Walls around Pacman

1 up

2 down

3 left

4 right

This command do the sequence $P(D) > P(B) > P(H) > P(G)$

.

5.19.1 Detailed Description

Command of the Pacman's moves 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)

5.20 ModelSED.m File Reference

abstract Class who contain the structure of a "fmg" implementation Input : necessary information for compute the next state of the model

Classes

- class [ModelSED](#)

State : minimal information necessary who evolve.

5.20.1 Detailed Description

abstract Class who contain the structure of a "fmg" implementation Input : necessary information for compute the next state of the model

Output : output's action of the model

5.21 ModelWalls.m File Reference

Command of the walls' move Input : No need

Output : [UPwalls , RIGHTwalls]

Classes

- class [ModelWalls](#)

This command do the sequence walls Right -> walls down

5.21.1 Detailed Description

Command of the walls' move Input : No need

Output : [UPwalls , RIGHTwalls]

5.22 setColor.m File Reference

Functions

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

5.22.1 Function Documentation

5.22.1.1 setColor()

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

5.23 Simulation.m File Reference

5.24 Simulation2_allpossiblewalls.m File Reference

5.25 StopCondition.m File Reference

Classes

- class [StopCondition](#)

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

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

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

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

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

5.31 validation/Validation 2/Test14/validation2.m File Reference

5.32 validation/Validation 2/Test15/validation2.m File Reference

5.33 validation/Validation 2/Test16/validation2.m File Reference

5.34 validation/Validation 2/Test17/validation2.m File Reference

5.35 validation/Validation 2/Test2/validation2.m File Reference

5.36 validation/Validation 2/Test3/validation2.m File Reference

5.37 validation/Validation 2/Test4/validation2.m File Reference

5.38 validation/Validation 2/Test5/validation2.m File Reference

- 5.39 validation/Validation 2/Test6/validation2.m File Reference
- 5.40 validation/Validation 2/Test7/validation2.m File Reference
- 5.41 validation/Validation 2/Test8/validation2.m File Reference
- 5.42 validation/Validation 2/Test9/validation2.m File Reference
- 5.43 validation/Validation 3/Test1/verification3.m File Reference
- 5.44 validation/Validation 3/verification3.m File Reference
- 5.45 validation/Validation 4/test.m File Reference
- 5.46 validation/Validation 4/Test1/test.m File Reference
- 5.47 validation/Validation 4/Test1/validation4.m File Reference
- 5.48 validation/Validation 4/validation4.m File Reference
- 5.49 validation/Validation 7/validation7.m File Reference
- 5.50 validation/Validation 8/Test1/validation8.m File Reference
- 5.51 visupacman.m File Reference
- 5.52 visupacman2.m File Reference
- 5.53 wallsBorder.m File Reference

Functions

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

5.53.1 Function Documentation

5.53.1.1 wallsBorder()

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

5.54 Wrapper.m File Reference

Classes

- class [Wrapper](#)

Index

- AutomatonSchedulingCreation.m
 - function, [46](#)
- AutomatonStrutureLabyCreation
 - AutomatonStrutureLabyCreation.m, [47](#)
 - Plan_desumaFunctions_2Players.m, [48](#)
- AutomatonStrutureLabyCreation.m
 - AutomatonStrutureLabyCreation, [47](#)
- AutomatonWallsConstraintsCreation
 - AutomatonWallsConstraintsCreation.m, [47](#)
- AutomatonWallsConstraintsCreation.m
 - AutomatonWallsConstraintsCreation, [47](#)
- commandGhost
 - Wrapper, [36](#)
- commandPacman
 - Wrapper, [36](#)
- commandWalls
 - Wrapper, [36](#)
- connect_Callback
 - figure_Laby.m, [40](#)
- CreatePituresAndVideo
 - CreatePituresAndVideo.m, [39](#)
- CreatePituresAndVideo.m, [39](#)
 - CreatePituresAndVideo, [39](#)
- CreatePituresAndVideo_textured
 - CreatePituresAndVideo_textured.m, [39](#)
- CreatePituresAndVideo_textured.m, [39](#)
 - CreatePituresAndVideo_textured, [39](#)
- createUIEscape
 - figure_Laby.m, [40](#)
- createUIGhost
 - figure_Laby.m, [40](#)
- createUIPacman
 - figure_Laby.m, [41](#)
- createUIWalls
 - figure_Laby.m, [41](#)
- Down
 - ModelCommand, [8](#)
- f
 - ModelCommand, [8](#)
 - ModelGhost, [12](#)
 - ModelLaby, [16](#)
 - ModelPacman, [21](#)
 - ModelSED, [24](#)
 - ModelWalls, [28](#)
 - StopCondition, [32](#)
- figure_Laby
 - figure_Laby.m, [41](#)
- figure_Laby.m, [40](#)
 - connect_Callback, [40](#)
 - createUIEscape, [40](#)
 - createUIGhost, [40](#)
 - createUIPacman, [41](#)
 - createUIWalls, [41](#)
 - figure_Laby, [41](#)
 - figure_Laby_OpeningFcn, [41](#)
 - figure_Laby_OutputFcn, [41](#)
 - isOne, [41](#)
 - resetUIConnection, [42](#)
 - ui_Callback, [42](#)
 - updatePresenceDetectorDisplay, [42](#)
 - updateUIActiveCammand, [42](#)
 - updateUIButton, [42](#)
 - updateUICaught, [43](#)
 - updateUIEscape, [43](#)
 - updateUIPlayer, [43](#)
 - updateUIWalls, [43](#)
 - updateUIWallsAround, [43](#)
 - updateUI, [42](#)
- figure_Laby_OpeningFcn
 - figure_Laby.m, [41](#)
- figure_Laby_OutputFcn
 - figure_Laby.m, [41](#)
- function
 - AutomatonSchedulingCreation.m, [46](#)
 - Plan_desumaFunctions_2Players.m, [48](#)
- g
 - ModelCommand, [8](#)
 - ModelGhost, [13](#)
 - ModelLaby, [16](#)
 - ModelPacman, [22](#)
 - ModelSED, [25](#)
 - ModelWalls, [28](#)
 - StopCondition, [33](#)
- generer_lab
 - generer_lab.m, [48](#)
- generer_lab.m
 - generer_lab, [48](#)
- get_out
 - Wrapper, [35](#)
- get_stop
 - Wrapper, [35](#)
- ghostBit
 - Wrapper, [36](#)
- handle, [7](#)

- i
 - ModelPacman, 23
 - ModelWalls, 30
- in
 - Wrapper, 36
- init
 - Wrapper, 35
- initialState
 - ModelGhost, 14
 - ModelLaby, 19
 - ModelPacman, 23
 - ModelSED, 26
 - ModelWalls, 30
 - StopCondition, 33
- isOne
 - figure_Laby.m, 41
- knowCompart
 - ModelCommand, 8
- LabyMenu
 - LabyMenu.m, 44
- LabyMenu.m, 44
 - LabyMenu, 44
 - LabyMenu_OpeningFcn, 44
 - LabyMenu_OutputFcn, 44
 - OneEasy_Callback, 44
 - OneHard_Callback, 45
 - OneMedium_Callback, 45
 - slider1_Callback, 45
 - slider1_CreateFcn, 45
 - TwoEasy_Callback, 45
 - TwoHard_Callback, 45
 - TwoMedium_Callback, 46
- LabyMenu_OpeningFcn
 - LabyMenu.m, 44
- LabyMenu_OutputFcn
 - LabyMenu.m, 44
- Left
 - ModelCommand, 9
- m
 - ModelCommand, 8
 - ModelGhost, 13
 - ModelLaby, 17
 - ModelPacman, 22
 - ModelSED, 25
 - ModelWalls, 30
 - StopCondition, 33
- main.m, 46
- matrixAllPossible.m, 46
- memory
 - ModelPacman, 23
- ModelCommand, 7
 - Down, 8
 - f, 8
 - g, 8
 - knowCompart, 8
 - Left, 9
 - m, 8
 - presentState, 9
 - Right, 9
 - sizeTab, 9
 - Up, 9
- ModelCommand.m, 46
- ModelGenerator/AutomatonSchedulingCreation.m, 46
- ModelGenerator/AutomatonStrutureLabyCreation.m, 47
- ModelGenerator/AutomatonWallsConstraintsCreation.m, 47
- ModelGenerator/Plan_desumaFunctions_2Players.m, 48
- ModelGenerator/SaveDESUMAFFile.m, 49
- ModelGenerator/generer_lab.m, 47
- ModelGenerator/modelGenerator.m, 48
- ModelGenerator/writeStates.m, 49
- ModelGenerator/writeTransitions.m, 50
- ModelGhost, 10
 - f, 12
 - g, 13
 - initialState, 14
 - m, 13
 - ModelGhost, 11
 - presentState, 14
- ModelGhost.m, 50
- ModelLaby, 14
 - f, 16
 - g, 16
 - initialState, 19
 - m, 17
 - ModelLaby, 15
 - presentState, 19
 - sameX_position, 17
 - sameY_position, 18
 - wallsHBetween, 18
 - wallsHBetweenOne, 18
 - wallsVBetween, 19
 - wallsVBetweenOne, 19
- modelLaby
 - Wrapper, 36
- ModelLaby.m, 51
- ModelPacman, 20
 - f, 21
 - g, 22
 - i, 23
 - initialState, 23
 - m, 22
 - memory, 23
 - ModelPacman, 21
 - presentState, 23
- ModelPacman.m, 51
- ModelSED.m, 51
- ModelSED, 24
 - f, 24
 - g, 25
 - initialState, 26
 - m, 25
 - presentState, 26

- ModelWalls, 26
 - f, 28
 - g, 28
 - i, 30
 - initialState, 30
 - m, 30
 - ModelWalls, 27
 - presentState, 30
 - val, 31
- ModelWalls.m, 52
- OneEasy_Callback
 - LabyMenu.m, 44
- OneHard_Callback
 - LabyMenu.m, 45
- OneMedium_Callback
 - LabyMenu.m, 45
- orderer
 - Wrapper, 35
- out
 - Wrapper, 36
- pacmanBit
 - Wrapper, 36
- Plan_desumaFunctions_2Players.m
 - AutomatonStrutureLabyCreation, 48
 - function, 48
 - SaveDESUMAFFile, 48
 - writeStates, 49
 - writeTransitions, 49
- presentState
 - ModelCommand, 9
 - ModelGhost, 14
 - ModelLaby, 19
 - ModelPacman, 23
 - ModelSED, 26
 - ModelWalls, 30
 - StopCondition, 34
- resetUIConnection
 - figure_Laby.m, 42
- Right
 - ModelCommand, 9
- sameX_position
 - ModelLaby, 17
- sameY_position
 - ModelLaby, 18
- SaveDESUMAFFile
 - Plan_desumaFunctions_2Players.m, 48
 - SaveDESUMAFFile.m, 49
- SaveDESUMAFFile.m
 - SaveDESUMAFFile, 49
- setColor
 - setColor.m, 52
- setColor.m, 52
 - setColor, 52
- Simulation.m, 53
- Simulation2_allpossiblewalls.m, 53
- sizeTab
 - ModelCommand, 9
- slider1_Callback
 - LabyMenu.m, 45
- slider1_CreateFcn
 - LabyMenu.m, 45
- stop
 - Wrapper, 37
- StopCondition, 31
 - f, 32
 - g, 33
 - initialState, 33
 - m, 33
 - presentState, 34
 - StopCondition, 32
- stopCondition
 - Wrapper, 37
- StopCondition.m, 53
- TwoEasy_Callback
 - LabyMenu.m, 45
- TwoHard_Callback
 - LabyMenu.m, 45
- TwoMedium_Callback
 - LabyMenu.m, 46
- ui_Callback
 - figure_Laby.m, 42
- Up
 - ModelCommand, 9
- updateConnexion
 - Wrapper, 35
- updatePresenceDetectorDisplay
 - figure_Laby.m, 42
- updateUIActiveCammand
 - figure_Laby.m, 42
- updateUIButton
 - figure_Laby.m, 42
- updateUICaught
 - figure_Laby.m, 43
- updateUIEscape
 - figure_Laby.m, 43
- updateUIPlayer
 - figure_Laby.m, 43
- updateUIWalls
 - figure_Laby.m, 43
- updateUIWallsAround
 - figure_Laby.m, 43
- updateUI
 - figure_Laby.m, 42
- val
 - ModelWalls, 31
- validation/Validation 2/Test1/validation2.m, 53
- validation/Validation 2/Test10/validation2.m, 53
- validation/Validation 2/Test11/validation2.m, 53
- validation/Validation 2/Test12/validation2.m, 53
- validation/Validation 2/Test13/validation2.m, 53
- validation/Validation 2/Test14/validation2.m, 53

- validation/Validation 2/Test15/validation2.m, 53
- validation/Validation 2/Test16/validation2.m, 53
- validation/Validation 2/Test17/validation2.m, 53
- validation/Validation 2/Test2/validation2.m, 53
- validation/Validation 2/Test3/validation2.m, 53
- validation/Validation 2/Test4/validation2.m, 53
- validation/Validation 2/Test5/validation2.m, 53
- validation/Validation 2/Test6/validation2.m, 54
- validation/Validation 2/Test7/validation2.m, 54
- validation/Validation 2/Test8/validation2.m, 54
- validation/Validation 2/Test9/validation2.m, 54
- validation/Validation 3/Test1/verification3.m, 54
- validation/Validation 3/verification3.m, 54
- validation/Validation 4/Test1/test.m, 54
- validation/Validation 4/Test1/validation4.m, 54
- validation/Validation 4/test.m, 54
- validation/Validation 4/validation4.m, 54
- validation/Validation 7/validation7.m, 54
- validation/Validation 8/Test1/validation8.m, 54
- visupacman.m, 54
- visupacman2.m, 54
- wallsBit
 - Wrapper, 37
- wallsBorder
 - wallsBorder.m, 54
- wallsBorder.m, 54
 - wallsBorder, 54
- wallsHBetween
 - ModelLaby, 18
- wallsHBetweenOne
 - ModelLaby, 18
- wallsVBetween
 - ModelLaby, 19
- wallsVBetweenOne
 - ModelLaby, 19
- whoPlay
 - Wrapper, 37
- Wrapper, 34
 - commandGhost, 36
 - commandPacman, 36
 - commandWalls, 36
 - get_out, 35
 - get_stop, 35
 - ghostBit, 36
 - in, 36
 - init, 35
 - modelLaby, 36
 - orderer, 35
 - out, 36
 - pacmanBit, 36
 - stop, 37
 - stopCondition, 37
 - updateConnexion, 35
 - wallsBit, 37
 - whoPlay, 37
 - Wrapper, 34
- Wrapper.m, 55
- writeStates
 - Plan_desumaFunctions_2Players.m, 49
 - writeStates.m, 50
- writeStates.m
 - writeStates, 50
- writeTransitions
 - Plan_desumaFunctions_2Players.m, 49
 - writeTransitions.m, 50
- writeTransitions.m
 - writeTransitions, 50