

## Labyrinth - One Player mode

Generated by Doxygen 1.8.13



# Contents

<b>1</b>	<b>Hierarchical Index</b>	<b>1</b>
1.1	Class Hierarchy . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List . . . . .	3
<b>3</b>	<b>File Index</b>	<b>5</b>
3.1	File List . . . . .	5
<b>4</b>	<b>Class Documentation</b>	<b>7</b>
4.1	AutomateGraph Class Reference . . . . .	7
4.1.1	Constructor & Destructor Documentation . . . . .	8
4.1.1.1	AutomateGraph() . . . . .	8
4.1.2	Member Function Documentation . . . . .	8
4.1.2.1	accessibilityAutomate() . . . . .	8
4.1.2.2	adaptTourLangage() . . . . .	8
4.1.2.3	addWord2Langage() . . . . .	8
4.1.2.4	export2DESUMA() . . . . .	9
4.1.2.5	FSM2Automata() . . . . .	9
4.1.2.6	matrices2vector() . . . . .	9
4.1.2.7	PathResearche() . . . . .	9
4.1.2.8	structAutomata2vectorAutomata() . . . . .	9
4.1.2.9	vector2matrices() . . . . .	9
4.1.2.10	vector2structAutomata() . . . . .	10
4.1.3	Member Data Documentation . . . . .	10

4.1.3.1	langage	10
4.1.3.2	matrixTrans	10
4.1.3.3	state	10
4.1.3.4	transition	10
4.1.3.5	vector	10
4.2	Automaton Class Reference	11
4.2.1	Constructor & Destructor Documentation	11
4.2.1.1	Automaton()	11
4.2.2	Member Function Documentation	12
4.2.2.1	AjoutTransitionStable()	12
4.2.2.2	AutomateAccessible()	12
4.2.2.3	AutomateAccessibleIncertain()	12
4.2.2.4	CreationFonctionTransitionEnsembleDesParties()	12
4.2.2.5	Ensemble2Etats()	12
4.2.2.6	EnsemblesContenantEtat()	13
4.2.2.7	Etats2Ensemble()	13
4.2.2.8	Evolution()	13
4.2.2.9	PathRecherche()	13
4.2.3	Member Data Documentation	13
4.2.3.1	Fct	13
4.2.3.2	MatricesTransition	13
4.2.3.3	NomsEvenements	14
4.3	handle Class Reference	14
4.4	Labyrinthe Class Reference	14
4.4.1	Constructor & Destructor Documentation	15
4.4.1.1	Labyrinthe()	15
4.4.2	Member Function Documentation	15
4.4.2.1	affichage()	15
4.4.2.2	incertain()	15
4.4.2.3	jusqu_au_mur()	16

4.4.2.4	Pas_a_pas()	16
4.4.3	Member Data Documentation	16
4.4.3.1	Etats_Finaux	16
4.4.3.2	Etats_Initiaux	16
4.4.3.3	MursHorizontaux	16
4.4.3.4	MursVerticaux	16
4.5	ModelLaby Class Reference	17
4.5.1	Detailed Description	18
4.5.2	Constructor & Destructor Documentation	18
4.5.2.1	ModelLaby()	18
4.5.3	Member Function Documentation	19
4.5.3.1	f()	19
4.5.3.2	g()	19
4.5.3.3	m()	20
4.5.4	Member Data Documentation	20
4.5.4.1	initialState	20
4.5.4.2	presentState	20
4.6	ModelPacman Class Reference	21
4.6.1	Detailed Description	22
4.6.2	Constructor & Destructor Documentation	23
4.6.2.1	ModelPacman()	23
4.6.3	Member Function Documentation	23
4.6.3.1	f()	23
4.6.3.2	g()	24
4.6.3.3	m()	24
4.6.4	Member Data Documentation	25
4.6.4.1	initialState	25
4.6.4.2	presentState	25
4.7	ModelSED Class Reference	25
4.7.1	Detailed Description	26

4.7.2	Member Function Documentation . . . . .	26
4.7.2.1	f() . . . . .	26
4.7.2.2	g() . . . . .	27
4.7.2.3	m() . . . . .	27
4.7.3	Member Data Documentation . . . . .	28
4.7.3.1	initialState . . . . .	28
4.7.3.2	presentState . . . . .	28
4.8	ModelWalls Class Reference . . . . .	28
4.8.1	Detailed Description . . . . .	29
4.8.2	Constructor & Destructor Documentation . . . . .	30
4.8.2.1	ModelWalls() . . . . .	30
4.8.3	Member Function Documentation . . . . .	30
4.8.3.1	f() [1/2] . . . . .	30
4.8.3.2	f() [2/2] . . . . .	30
4.8.3.3	g() . . . . .	31
4.8.3.4	m() . . . . .	31
4.8.4	Member Data Documentation . . . . .	31
4.8.4.1	i . . . . .	32
4.8.4.2	initialState . . . . .	32
4.8.4.3	presentState . . . . .	32
4.8.4.4	val . . . . .	32
4.9	StopCondition Class Reference . . . . .	32
4.9.1	Constructor & Destructor Documentation . . . . .	33
4.9.1.1	StopCondition() . . . . .	33
4.9.2	Member Function Documentation . . . . .	34
4.9.2.1	f() [1/2] . . . . .	34
4.9.2.2	f() [2/2] . . . . .	34
4.9.2.3	g() . . . . .	34
4.9.2.4	m() . . . . .	35
4.9.3	Member Data Documentation . . . . .	35

4.9.3.1	<a href="#">initialState</a>	35
4.9.3.2	<a href="#">presentState</a>	35
4.10	<a href="#">Wrapper Class Reference</a>	36
4.10.1	<a href="#">Constructor &amp; Destructor Documentation</a>	36
4.10.1.1	<a href="#">Wrapper()</a>	37
4.10.2	<a href="#">Member Function Documentation</a>	37
4.10.2.1	<a href="#">get_out()</a>	37
4.10.2.2	<a href="#">get_stop()</a>	37
4.10.2.3	<a href="#">init()</a>	37
4.10.2.4	<a href="#">orderer()</a>	37
4.10.2.5	<a href="#">updateConnexion()</a>	37
4.10.3	<a href="#">Member Data Documentation</a>	38
4.10.3.1	<a href="#">commandPacman</a>	38
4.10.3.2	<a href="#">commandWalls</a>	38
4.10.3.3	<a href="#">in</a>	38
4.10.3.4	<a href="#">modellLaby</a>	38
4.10.3.5	<a href="#">out</a>	38
4.10.3.6	<a href="#">pacmanBit</a>	38
4.10.3.7	<a href="#">stop</a>	38
4.10.3.8	<a href="#">stopCondition</a>	39
4.10.3.9	<a href="#">wallsBit</a>	39
4.10.3.10	<a href="#">whoPlay</a>	39

<b>5</b>	<b>File Documentation</b>	<b>41</b>
5.1	automaton/AutomateGraph.m File Reference . . . . .	41
5.2	automaton/mainLaby.m File Reference . . . . .	41
5.3	automaton/modelGenerator/AutomatonSchedulingCreation.m File Reference . . . . .	41
5.3.1	Function Documentation . . . . .	41
5.3.1.1	function() . . . . .	41
5.4	automaton/modelGenerator/AutomatonStrutureLabyCreation.m File Reference . . . . .	41
5.4.1	Function Documentation . . . . .	42
5.4.1.1	AutomatonStrutureLabyCreation() . . . . .	42
5.5	automaton/modelGenerator/AutomatonWallsConstraintsCreation.m File Reference . . . . .	42
5.5.1	Function Documentation . . . . .	42
5.5.1.1	AutomatonWallsConstraintsCreation() . . . . .	42
5.6	automaton/modelGenerator/generer_lab.m File Reference . . . . .	42
5.6.1	Function Documentation . . . . .	42
5.6.1.1	generer_lab() . . . . .	42
5.7	automaton/modelGenerator/modelGenerator.m File Reference . . . . .	43
5.8	automaton/modelGenerator/Plan_desumaFunctions.m File Reference . . . . .	43
5.8.1	Function Documentation . . . . .	43
5.8.1.1	AutomatonStrutureLabyCreation() . . . . .	43
5.8.1.2	function() . . . . .	43
5.8.1.3	SaveDESUMAFile() . . . . .	43
5.8.1.4	writeStates() . . . . .	43
5.8.1.5	writeTransitions() . . . . .	44
5.9	automaton/modelGenerator/SaveDESUMAFile.m File Reference . . . . .	44
5.9.1	Function Documentation . . . . .	44
5.9.1.1	SaveDESUMAFile() . . . . .	44
5.10	automaton/modelGenerator/writeStates.m File Reference . . . . .	44
5.10.1	Function Documentation . . . . .	44
5.10.1.1	writeStates() . . . . .	44
5.11	automaton/modelGenerator/writeTransitions.m File Reference . . . . .	45



5.11.1	Function Documentation	45
5.11.1.1	writeTransitions()	45
5.12	automaton/optimalCommand/creationMatricetransition.m File Reference	45
5.12.1	Function Documentation	45
5.12.1.1	creationMatricetransition()	45
5.13	automaton/optimalCommand/getStateTransitionFSM.m File Reference	45
5.13.1	Function Documentation	45
5.13.1.1	getStateTransitionFSM()	46
5.14	automaton/optimalCommand/getStateTransitionTXT.m File Reference	46
5.14.1	Function Documentation	46
5.14.1.1	getStateTransitionTXT()	46
5.15	automaton/optimalCommand/main.m File Reference	46
5.16	main.m File Reference	46
5.17	automaton/optimalCommand/optimalCommand.m File Reference	46
5.17.1	Function Documentation	46
5.17.1.1	optimalCommand()	46
5.18	automaton/optimalCommand/ParcourirMatricesTransitions.m File Reference	47
5.18.1	Function Documentation	47
5.18.1.1	ParcourirMatricesTransitions()	47
5.19	automaton/optimalCommand/rafineAutomaton.m File Reference	47
5.20	automaton/optimalCommand/rafineAutomatonClass.m File Reference	47
5.20.1	Function Documentation	47
5.20.1.1	rafineAutomatonClass()	47
5.21	automaton/ParrallelComposition.m File Reference	47
5.21.1	Function Documentation	48
5.21.1.1	ParrallelComposition()	48
5.22	automaton_nd/Automaton.m File Reference	48
5.23	automaton_nd/Labyrinthe.m File Reference	48
5.24	automaton_nd/modi_main.m File Reference	48
5.25	CreatePituresAndVideo.m File Reference	48

5.25.1	Function Documentation	48
5.25.1.1	CreatePituresAndVideo()	48
5.26	CreatePituresAndVideo_textured.m File Reference	49
5.26.1	Function Documentation	49
5.26.1.1	CreatePituresAndVideo_textured()	49
5.27	figure_Laby.m File Reference	49
5.27.1	Function Documentation	50
5.27.1.1	connect_Callback()	50
5.27.1.2	createUIEscape()	50
5.27.1.3	createUIPacman()	52
5.27.1.4	createUIWalls()	52
5.27.1.5	figure_Laby()	53
5.27.1.6	figure_Laby_OpeningFcn()	53
5.27.1.7	figure_Laby_OutputFcn()	53
5.27.1.8	isOne()	54
5.27.1.9	resetUIConnection()	54
5.27.1.10	ui_Callback()	54
5.27.1.11	updatePresenceDetectorDisplay()	55
5.27.1.12	updateUI()	55
5.27.1.13	updateUIActiveCammand()	56
5.27.1.14	updateUIButton()	56
5.27.1.15	updateUIEscape()	56
5.27.1.16	updateUIPlayer()	57
5.27.1.17	updateUIWalls()	57
5.27.1.18	updateUIWallsAround()	57
5.28	ModelLaby.m File Reference	58
5.29	ModelPacman.m File Reference	58
5.29.1	Detailed Description	58
5.30	ModelSED.m File Reference	58
5.30.1	Detailed Description	59
5.31	ModelWalls.m File Reference	59
5.31.1	Detailed Description	59
5.32	setColor.m File Reference	59
5.32.1	Function Documentation	59
5.32.1.1	setColor()	59
5.33	Simulation.m File Reference	60
5.34	StopCondition.m File Reference	60
5.35	visupacman.m File Reference	60
5.36	visupacman2.m File Reference	60
5.37	wallsBorder.m File Reference	60
5.37.1	Function Documentation	60
5.37.1.1	wallsBorder()	60
5.38	Wrapper.m File Reference	60





# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

AutomateGraph . . . . .	7
Automaton . . . . .	11
handle . . . . .	14
ModelSED . . . . .	25
ModelLaby . . . . .	17
ModelPacman . . . . .	21
ModelWalls . . . . .	28
StopCondition . . . . .	32
Labyrinthe . . . . .	14
Wrapper . . . . .	36



## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">AutomateGraph</a>	7
<a href="#">Automaton</a>	11
<a href="#">handle</a>	14
<a href="#">Labyrinthe</a>	14
<a href="#">ModelLaby</a>	
Class which contains the "fmg" structure of the labyrinth for 1 player	17
<a href="#">ModelPacman</a>	
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 $P(D) > P(B) > P(H) > P(G)$	
21	
<a href="#">ModelSED</a>	
Input : necessary information for compute the next state of the model	
Output : output's action of the model	
25	
<a href="#">ModelWalls</a>	
Input : No need	
Output : [UPwalls , RIGHTwalls]	
State : contain the last move ( 0 = up ; 1 = right)	
This command do the sequence walls Right $\rightarrow$ walls down	
28	

<a href="#">StopCondition</a> . . . . .	<a href="#">32</a>
<a href="#">Wrapper</a> . . . . .	<a href="#">36</a>



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

CreatePituresAndVideo.m	48
CreatePituresAndVideo_textured.m	49
figure_Laby.m	49
main.m	46
ModelLaby.m	58
ModelPacman.m	
Contain ghost Pacman control	58
ModelSED.m	
Abstract Class who contain the structure of a "fmg" implementation	58
ModelWalls.m	
Contain wall movement command	59
setColor.m	59
Simulation.m	60
StopCondition.m	60
visupacman.m	60
visupacman2.m	60
wallsBorder.m	60
Wrapper.m	60
automaton/AutomateGraph.m	41
automaton/mainLaby.m	41
automaton/ParrallelComposition.m	47
automaton/modelGenerator/AutomatonSchedulingCreation.m	41
automaton/modelGenerator/AutomatonStrutureLabyCreation.m	41
automaton/modelGenerator/AutomatonWallsContraintsCreation.m	42
automaton/modelGenerator/generer_lab.m	42
automaton/modelGenerator/modelGenerator.m	43
automaton/modelGenerator/Plan_desumaFunctions.m	43
automaton/modelGenerator/SaveDESUMAFFile.m	44
automaton/modelGenerator/writeStates.m	44
automaton/modelGenerator/writeTransitions.m	45
automaton/optimalCommand/creationMatricetransition.m	45
automaton/optimalCommand/getStateTransitionFSM.m	45
automaton/optimalCommand/getStateTransitionTXT.m	46
automaton/optimalCommand/main.m	46
automaton/optimalCommand/optimalCommand.m	46

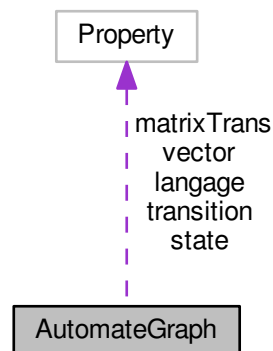
automaton/optimalCommand/ <a href="#">ParcourirMatricesTransitions.m</a> . . . . .	47
automaton/optimalCommand/ <a href="#">rafineAutomaton.m</a> . . . . .	47
automaton/optimalCommand/ <a href="#">rafineAutomatonClass.m</a> . . . . .	47
automaton_nd/ <a href="#">Automaton.m</a> . . . . .	48
automaton_nd/ <a href="#">Labyrinthe.m</a> . . . . .	48
automaton_nd/ <a href="#">modi_main.m</a> . . . . .	48

## Chapter 4

# Class Documentation

### 4.1 AutomateGraph Class Reference

Collaboration diagram for AutomateGraph:



#### Public Member Functions

- [function AutomateGraph \(\)](#)
- [function FSM2Automata](#) (in obj, in nameFileFSM)
- [function vector2matrices](#) (in obj)
- [function addWord2Langage](#) (in obj, in word)
- [function adaptTourLangage](#) (in obj)
- [function structAutomata2vectorAutomata](#) (in obj)
- [function matrices2vector](#) (in obj)
- [function PathResearche](#) (in obj, in initialState, in studiedState)
- [function vector2structAutomata](#) (in obj)
- [function export2DESUMA](#) (in obj, in file)
- [function accessibilityAutomate](#) (in obj)

## Public Attributes

- Property [state](#)
- Property [transition](#)
- Property [matrixTrans](#)
- Property [langage](#)
- Property [vector](#)

## 4.1.1 Constructor & Destructor Documentation

### 4.1.1.1 AutomateGraph()

```
function AutomateGraph ( )
```

## 4.1.2 Member Function Documentation

### 4.1.2.1 accessibilityAutomate()

```
function accessibilityAutomate (
    in obj )
```

### 4.1.2.2 adaptTourLangage()

```
function adaptTourLangage (
    in obj )
```

### 4.1.2.3 addWord2Langage()

```
function addWord2Langage (
    in obj,
    in word )
```

#### 4.1.2.4 export2DESUMA()

```
function export2DESUMA (
    in obj,
    in file )
```

#### 4.1.2.5 FSM2Automata()

```
function FSM2Automata (
    in obj,
    in nameFileFSM )
```

#### 4.1.2.6 matrices2vector()

```
function matrices2vector (
    in obj )
```

#### 4.1.2.7 PathResearche()

```
function PathResearche (
    in obj,
    in initialState,
    in studiedState )
```

#### 4.1.2.8 structAutomata2vectorAutomata()

```
function structAutomata2vectorAutomata (
    in obj )
```

#### 4.1.2.9 vector2matrices()

```
function vector2matrices (
    in obj )
```

#### 4.1.2.10 vector2structAutomata()

```
function vector2structAutomata (
    in obj )
```

### 4.1.3 Member Data Documentation

#### 4.1.3.1 langage

Property langage

#### 4.1.3.2 matrixTrans

Property matrixTrans

#### 4.1.3.3 state

Property state

#### 4.1.3.4 transition

Property transition

#### 4.1.3.5 vector

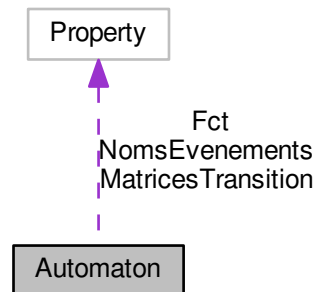
Property vector

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

- automaton/[AutomateGraph.m](#)

## 4.2 Automaton Class Reference

Collaboration diagram for Automaton:



### Public Member Functions

- [function Automaton](#) (in varargin)
- [function CreationFonctionTransitionEnsembleDesParties](#) (in obj)
- [function Evolution](#) (in obj, in Conditions, in Initial)
- [function AjoutTransitionStable](#) (in obj)
- [function PathRecherche](#) (in obj, in initialState, in studiedState)
- [function AutomateAccessible](#) (in obj, in initialState, in studiedState)
- [function AutomateAccessibleIncertain](#) (in obj, in Etat\_initial, in Etat)
- [function Ensemble2Etats](#) (in obj, in Ensemble)
- [function EnsemblesContenantEtat](#) (in obj, in Etats)
- [function Etats2Ensemble](#) (in obj, in Etats)

### Public Attributes

- Property [MatricesTransition](#)
- Property [Fct](#)
- Property [NomsEvenements](#)

### 4.2.1 Constructor & Destructor Documentation

#### 4.2.1.1 Automaton()

```
function Automaton (
    in varargin )
```

## 4.2.2 Member Function Documentation

### 4.2.2.1 AjoutTransitionStable()

```
function AjoutTransitionStable (  
    in obj )
```

### 4.2.2.2 AutomateAccessible()

```
function AutomateAccessible (  
    in obj,  
    in initialState,  
    in studiedState )
```

### 4.2.2.3 AutomateAccessibleIncertain()

```
function AutomateAccessibleIncertain (  
    in obj,  
    in Etat_initial,  
    in Etat )
```

### 4.2.2.4 CreationFonctionTransitionEnsembleDesParties()

```
function CreationFonctionTransitionEnsembleDesParties (  
    in obj )
```

### 4.2.2.5 Ensemble2Etats()

```
function Ensemble2Etats (  
    in obj,  
    in Ensemble )
```



#### 4.2.2.6 EnsemblesContenantEtat()

```
function EnsemblesContenantEtat (
    in obj,
    in Etats )
```

#### 4.2.2.7 Etats2Ensemble()

```
function Etats2Ensemble (
    in obj,
    in Etats )
```

#### 4.2.2.8 Evolution()

```
function Evolution (
    in obj,
    in Conditions,
    in Initial )
```

#### 4.2.2.9 PathRecherche()

```
function PathRecherche (
    in obj,
    in initialState,
    in studiedState )
```

### 4.2.3 Member Data Documentation

#### 4.2.3.1 Fct

Property Fct

#### 4.2.3.2 MatricesTransition

Property MatricesTransition

#### 4.2.3.3 NomsEvenements

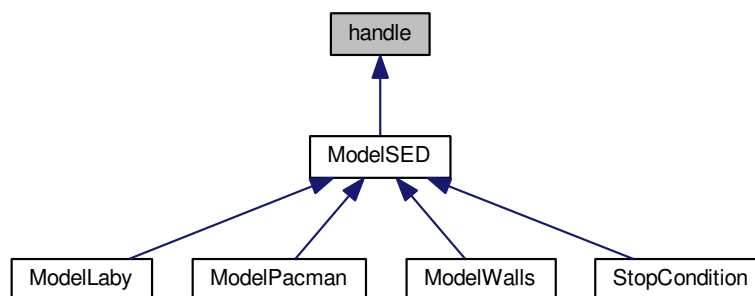
Property NomsEvenements

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

- automaton\_nd/[Automaton.m](#)

### 4.3 handle Class Reference

Inheritance diagram for handle:

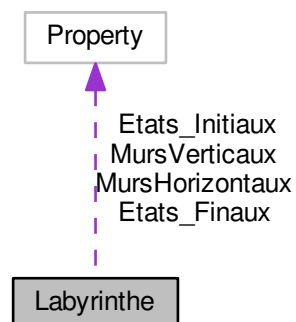


The documentation for this class was generated from the following file:

- [ModelSED.m](#)

### 4.4 Labyrinthe Class Reference

Collaboration diagram for Labyrinthe:



## Public Member Functions

- [function Labyrinthe](#) (in murs\_Verticaux, in murs\_Horizontaux, in etats\_Initiaux, in etats\_Finaux)
- [function Pas\\_a\\_pas](#) (in obj)
- [function jusqu\\_au\\_mur](#) (in obj)
- [function incertain](#) (in obj)
- [function affichage](#) (in obj)

## Public Attributes

- Property [MursVerticaux](#)
- Property [MursHorizontaux](#)
- Property [Etats\\_Initiaux](#)
- Property [Etats\\_Finaux](#)

### 4.4.1 Constructor & Destructor Documentation

#### 4.4.1.1 Labyrinthe()

```
function Labyrinthe (  
    in murs_Verticaux,  
    in murs_Horizontaux,  
    in etats_Initiaux,  
    in etats_Finaux )
```

### 4.4.2 Member Function Documentation

#### 4.4.2.1 affichage()

```
function affichage (  
    in obj )
```

#### 4.4.2.2 incertain()

```
function incertain (  
    in obj )
```

#### 4.4.2.3 jusqu\_au\_mur()

```
function jusqu_au_mur (  
    in obj )
```

#### 4.4.2.4 Pas\_a\_pas()

```
function Pas_a_pas (  
    in obj )
```

### 4.4.3 Member Data Documentation

#### 4.4.3.1 Etats\_Finaux

Property Etats\_Finaux

#### 4.4.3.2 Etats\_Initiaux

Property Etats\_Initiaux

#### 4.4.3.3 MursHorizontaux

Property MursHorizontaux

#### 4.4.3.4 MursVerticaux

Property MursVerticaux

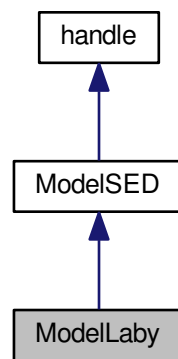
The documentation for this class was generated from the following file:

- automaton\_nd/[Labyrinthe.m](#)

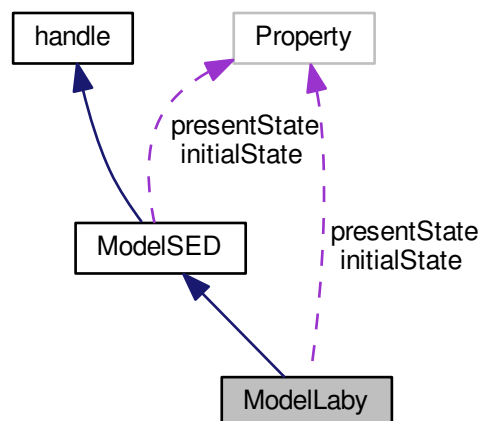
## 4.5 ModelLaby Class Reference

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

Inheritance diagram for ModelLaby:



Collaboration diagram for ModelLaby:



### Public Member Functions

- [function ModelLaby](#) (in wallsV\_init, in wallsH\_init, in pacman\_init, in escape\_init)  
Class constructor of Instance of [ModelLaby](#) Class.
- [function f](#) (in obj, in in)

- *Compute the evolution of the model.*
- **function m** (in obj, in nextState, in init)  
*Memory method update the state of the command.*
- **function g** (in obj)  
*Create the outputs in a 1x9 cell-array.*

## Public Attributes

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

### 4.5.1 Detailed Description

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

Input : necessary information for compute the next state of the model

Output : output's action of the model

State : minimal information necessary who evolve

### 4.5.2 Constructor & Destructor Documentation

#### 4.5.2.1 ModelLaby()

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

Class constructor of Instance of **ModelLaby** Class.

#### Parameters

<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>escape_init</i>	Contain a vector (x, y) of Escape 's Position.

**Returns**

instance of the [ModelLaby](#) class.

**4.5.3 Member Function Documentation****4.5.3.1 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.

**Returns**

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

Reimplemented from [ModelSED](#).

**4.5.3.2 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](#).

#### 4.5.3.3 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 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.

#### 4.5.4.2 presentState

Property `presentState`

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

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

- [Modellaby.m](#)



## 4.6 ModelPacman Class Reference

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

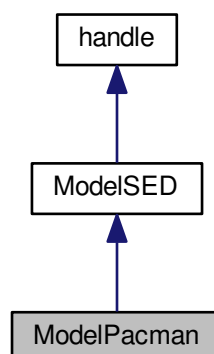
3 left

4 right

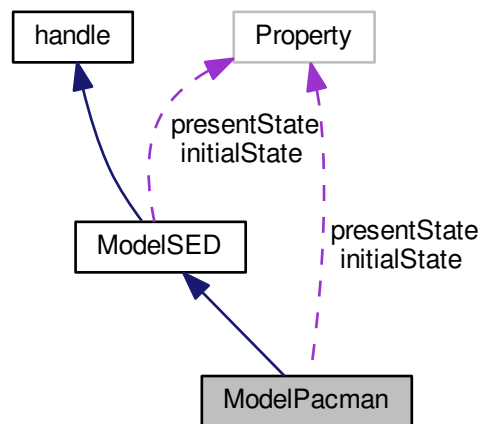
This command do the sequence  $P(D) > P(B) > P(H) > P(G)$

.

Inheritance diagram for ModelPacman:



Collaboration diagram for ModelPacman:



## Public Member Functions

- **function** **ModelPacman** (in initialValue)  
*Class constructor.*
- **function** **f** (in obj, in in)  
*Compute the evolution of the command.*
- **function** **m** (in obj, in nextState, in init)  
*Memory method update the state of the command.*
- **function** **g** (in obj)  
*Create the outputs.*

## Public Attributes

- Property **presentState**  
*This is the state of the command in the present moment.*
- Property **initialState**  
*This is the state of the command in the initialization and when it's reseted.*

### 4.6.1 Detailed Description

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

.

## 4.6.2 Constructor & Destructor Documentation

### 4.6.2.1 ModelPacman()

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

Class constructor.

#### Parameters

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

#### Returns

instance of the [ModelPacman](#) class.

## 4.6.3 Member Function Documentation

### 4.6.3.1 f()

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

Compute the evolution of the command.

#### Parameters

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

**Return values**

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

Reimplemented from [ModelSED](#).

**4.6.3.2 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](#).

**4.6.3.3 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](#).

#### 4.6.4 Member Data Documentation

##### 4.6.4.1 initialState

Property initialState

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

##### 4.6.4.2 presentState

Property presentState

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

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

- [ModelPacman.m](#)

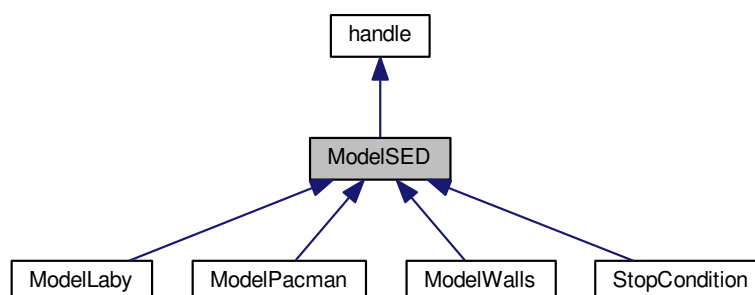
## 4.7 ModelSED Class Reference

Input : necessary information for compute the next state of the model

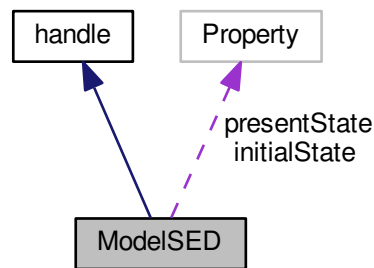
Output : output's action of the model

.

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.*

#### 4.7.1 Detailed Description

Input : necessary information for compute the next state of the model

Output : output's action of the model

.

State : minimal information necessary who evolve

#### 4.7.2 Member Function Documentation

##### 4.7.2.1 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](#).

## 4.7.2.2 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 [ModelPacman](#), [ModelLaby](#), [ModelWalls](#), and [StopCondition](#).

## 4.7.2.3 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 [ModelPacman](#), [ModelLaby](#), [ModelWalls](#), and [StopCondition](#).

### 4.7.3 Member Data Documentation

#### 4.7.3.1 initialState

Property `initialState`

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

#### 4.7.3.2 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](#)

## 4.8 ModelWalls Class Reference

Input : No need

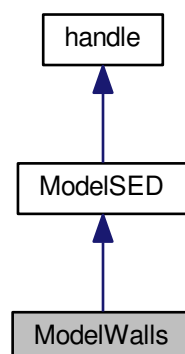
Output : [UPwalls , RIGHTwalls]

State : contain the last move ( 0 = up ; 1 = right)

This command do the sequence walls Right -> walls down

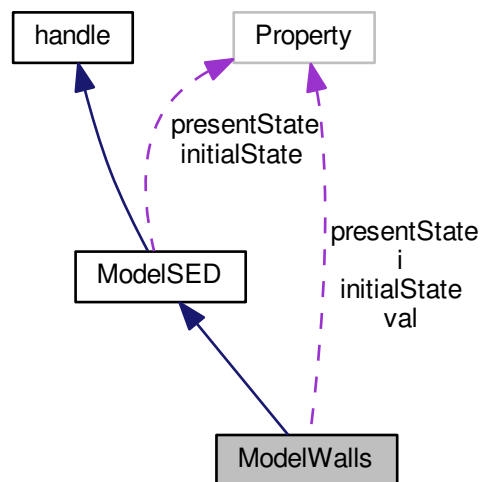
.

Inheritance diagram for ModelWalls:





Collaboration diagram for ModelWalls:



### Public Member Functions

- **function** `ModelWalls` (in `initValue`)  
*Class constructor.*
- **function** `f` (in `obj`)
- **function** `m` (in `obj`, in `nextState`, in `init`)  
*Memory method update the state of the command.*
- **function** `g` (in `obj`)  
*Create the outputs.*
- virtual **function** `f` (in `obj`, in `in`)  
*Compute the evolution of the model.*

### Public Attributes

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

#### 4.8.1 Detailed Description

Input : No need

Output : [UPwalls , RIGHTwalls]

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

## 4.8.2 Constructor & Destructor Documentation

### 4.8.2.1 ModelWalls()

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

Class constructor.

#### Parameters

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

#### Returns

instance of the [ModelWalls](#) 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 )
```

#### 4.8.3.3 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](#).

#### 4.8.3.4 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 i

Property i

#### 4.8.4.2 initialState

Property initialState

#### 4.8.4.3 presentState

Property presentState

#### 4.8.4.4 val

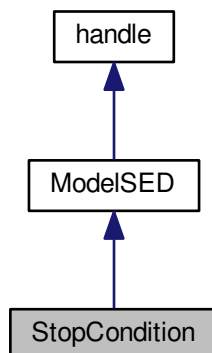
Property val

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

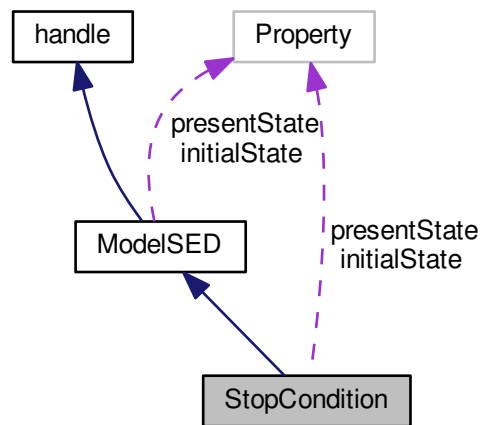
- [ModelWalls.m](#)

## 4.9 StopCondition Class Reference

Inheritance diagram for StopCondition:



Collaboration diagram for StopCondition:



### Public Member Functions

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

## 4.9.1 Constructor & Destructor Documentation

### 4.9.1.1 StopCondition()

```

function StopCondition (
    in initCondition )
  
```

## 4.9.2 Member Function Documentation

### 4.9.2.1 `f()` [1/2]

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

### 4.9.2.2 `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](#).

### 4.9.2.3 `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](#).

#### 4.9.2.4 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](#).

### 4.9.3 Member Data Documentation

#### 4.9.3.1 initialState

Property `initialState`

#### 4.9.3.2 presentState

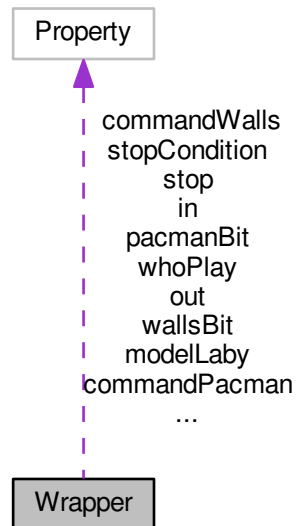
Property `presentState`

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

- [StopCondition.m](#)

## 4.10 Wrapper Class Reference

Collaboration diagram for Wrapper:



### Public Member Functions

- [function Wrapper](#) (in inSize, in outSize, in initLaby, in initWalls, in initPac, 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 [modelLaby](#)
- Property [commandWalls](#)
- Property [commandPacman](#)
- Property [stopCondition](#)
- Property [stop](#)
- Property [in](#)
- Property [out](#)
- Property [whoPlay](#)

#### 4.10.1 Constructor & Destructor Documentation



#### 4.10.1.1 Wrapper()

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

### 4.10.2 Member Function Documentation

#### 4.10.2.1 get\_out()

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

#### 4.10.2.2 get\_stop()

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

#### 4.10.2.3 init()

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

#### 4.10.2.4 orderer()

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

#### 4.10.2.5 updateConnexion()

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

### 4.10.3 Member Data Documentation

#### 4.10.3.1 `commandPacman`

Property `commandPacman`

#### 4.10.3.2 `commandWalls`

Property `commandWalls`

#### 4.10.3.3 `in`

Property `in`

#### 4.10.3.4 `modelLaby`

Property `modelLaby`

#### 4.10.3.5 `out`

Property `out`

#### 4.10.3.6 `pacmanBit`

Property `pacmanBit`

#### 4.10.3.7 `stop`

Property `stop`

#### 4.10.3.8 stopCondition

Property stopCondition

#### 4.10.3.9 wallsBit

Property wallsBit

#### 4.10.3.10 whoPlay

Property whoPlay

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

- [Wrapper.m](#)



## Chapter 5

# File Documentation

### 5.1 automaton/AutomateGraph.m File Reference

#### Classes

- class [AutomateGraph](#)

### 5.2 automaton/mainLaby.m File Reference

### 5.3 automaton/modelGenerator/AutomatonSchedulingCreation.m File Reference

#### Functions

- [function](#) ()

#### 5.3.1 Function Documentation

##### 5.3.1.1 [function](#)()

`function ( )`

### 5.4 automaton/modelGenerator/AutomatonStrutureLabyCreation.m File Reference

#### Functions

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

### 5.4.1 Function Documentation

#### 5.4.1.1 AutomatonStrutureLabyCreation()

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

## 5.5 automaton/modelGenerator/AutomatonWallsContraintsCreation.m File Reference

### Functions

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

### 5.5.1 Function Documentation

#### 5.5.1.1 AutomatonWallsContraintsCreation()

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

## 5.6 automaton/modelGenerator/generer\_lab.m File Reference

### Functions

- [function generer\\_lab](#) (in Matrice\_Horizontale, in Matrice\_Verticale)

### 5.6.1 Function Documentation

#### 5.6.1.1 generer\_lab()

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

## 5.7 automaton/modelGenerator/modelGenerator.m File Reference

## 5.8 automaton/modelGenerator/Plan\_desumaFunctions.m File Reference

### Functions

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

### 5.8.1 Function Documentation

#### 5.8.1.1 AutomatonStrutureLabyCreation()

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

#### 5.8.1.2 function()

```
function ( )
```

#### 5.8.1.3 SaveDESUMAFile()

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

#### 5.8.1.4 writeStates()

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

#### 5.8.1.5 writeTransitions()

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

### 5.9 automaton/modelGenerator/SaveDESUMAFFile.m File Reference

#### Functions

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

#### 5.9.1 Function Documentation

##### 5.9.1.1 SaveDESUMAFFile()

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

### 5.10 automaton/modelGenerator/writeStates.m File Reference

#### Functions

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

#### 5.10.1 Function Documentation

##### 5.10.1.1 writeStates()

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



## 5.11 automaton/modelGenerator/writeTransitions.m File Reference

### Functions

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

#### 5.11.1 Function Documentation

##### 5.11.1.1 writeTransitions()

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

## 5.12 automaton/optimalCommand/creationMatricetransition.m File Reference

### Functions

- [function creationMatricetransition](#) (in nameOfFileFSM, in cellOrder)

#### 5.12.1 Function Documentation

##### 5.12.1.1 creationMatricetransition()

```
function creationMatricetransition (
    in nameOfFileFSM,
    in cellOrder )
```

## 5.13 automaton/optimalCommand/getStateTransitionFSM.m File Reference

### Functions

- [function getStateTransitionFSM](#) (in nameOfFileFSM)

#### 5.13.1 Function Documentation

#### 5.13.1.1 `getStateTransitionFSM()`

```
function getStateTransitionFSM (
    in nameOfFileFSM )
```

### 5.14 `automaton/optimalCommand/getStateTransitionTXT.m` File Reference

#### Functions

- [function `getStateTransitionTXT`](#) (in `nameOfFileTXT`, in `ST`, in `SP`)

#### 5.14.1 Function Documentation

##### 5.14.1.1 `getStateTransitionTXT()`

```
function getStateTransitionTXT (
    in nameOfFileTXT,
    in ST,
    in SP )
```

### 5.15 `automaton/optimalCommand/main.m` File Reference

#### 5.16 `main.m` File Reference

### 5.17 `automaton/optimalCommand/optimalCommand.m` File Reference

#### Functions

- [function `optimalCommand`](#) (in `transitionsMatrix`, in `s_init`, in `s_final`)

#### 5.17.1 Function Documentation

##### 5.17.1.1 `optimalCommand()`

```
function optimalCommand (
    in transitionsMatrix,
    in s_init,
    in s_final )
```

## 5.18 automaton/optimalCommand/ParcourirMatricesTransitions.m File Reference

### Functions

- [function ParcourirMatricesTransitions](#) (in MatricesTransition, in Poids)

#### 5.18.1 Function Documentation

##### 5.18.1.1 ParcourirMatricesTransitions()

```
function ParcourirMatricesTransitions (
    in MatricesTransition,
    in Poids )
```

## 5.19 automaton/optimalCommand/rafineAutomaton.m File Reference

## 5.20 automaton/optimalCommand/rafineAutomatonClass.m File Reference

### Functions

- [function rafineAutomatonClass](#) (in A, in paternName)

#### 5.20.1 Function Documentation

##### 5.20.1.1 rafineAutomatonClass()

```
function rafineAutomatonClass (
    in A,
    in paternName )
```

## 5.21 automaton/ParrallelComposition.m File Reference

### Functions

- [function ParrallelComposition](#) (in A1, in A2)

### 5.21.1 Function Documentation

#### 5.21.1.1 ParrallelComposition()

```
function ParrallelComposition (
    in A1,
    in A2 )
```

## 5.22 automaton\_nd/Automaton.m File Reference

### Classes

- class [Automaton](#)

## 5.23 automaton\_nd/Labyrinthe.m File Reference

### Classes

- class [Labyrinthe](#)

## 5.24 automaton\_nd/modi\_main.m File Reference

## 5.25 CreatePituresAndVideo.m File Reference

### Functions

- [function CreatePituresAndVideo](#) (in n, in escape\_i, in labyState)

### 5.25.1 Function Documentation

#### 5.25.1.1 CreatePituresAndVideo()

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

## 5.26 CreatePituresAndVideo\_textured.m File Reference

### Functions

- [function CreatePituresAndVideo\\_textured](#) (in `n`, in `escape_i`, in `labyState`)

### 5.26.1 Function Documentation

#### 5.26.1.1 CreatePituresAndVideo\_textured()

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

## 5.27 figure\_Laby.m File Reference

### Functions

- [function figure\\_Laby](#) (in `varargin`)  
*figure\_Laby.m*
- [function figure\\_Laby\\_OpeningFcn](#) (in `hObject`, in `eventdata`, in `handles`, in `varargin`)  
*initialization function.*
- [function figure\\_Laby\\_OutputFcn](#) (in `hObject`, in `eventdata`, in `handles`)  
*Automatic generated function by GUI.*
- [function ui\\_Callback](#) (in `hObject`, in `eventdata`, in `handles`)  
*Callback for all the action's buttons (see detailed explications).*
- [function connect\\_Callback](#) (in `hObject`, in `eventdata`, in `handles`)  
*Callback for all the connection's buttons (see detailed explications).*
- [function createUIPacman](#) (in `handles`)  
*Creation of the graphical object "pacman".*
- [function createUIWalls](#) (in `handles`)  
*Creation of the graphical objects "walls".*
- [function createUIEscape](#) (in `handles`)  
*Creation of the graphical objects "escape".*
- [function updateUI](#) (in `handles`, in `out`)  
*This function update all graphicals element who can change.*
- [function updateUIActiveCammand](#) (in `handles`)  
*Update visibility of control panel, connection and step button.*
- [function updateUIButton](#) (in `handles`)  
*Show the needed moving buttons.*
- [function updateUIPlayer](#) (in `handles`, in `strPlayer`, in `position`)  
*Update graphical place of a player (only pacman in this case).*
- [function updateUIEscape](#) (in `elementToSet`, in `boolState`)
- [function updateUIWallsAround](#) (in `handles`, in `strElement`, in `wallsAround`)
- [function updateUIWalls](#) (in `wallsUI`, in `vertWalls`, in `horizWalls`)
- [function isOne](#) (in `boolCond`)
- [function updatePresenceDetectorDisplay](#) (in `elementToSet`, in `boolCondition`)
- [function resetUIConnection](#) (in `handles`)

## 5.27.1 Function Documentation

### 5.27.1.1 connect\_Callback()

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

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

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

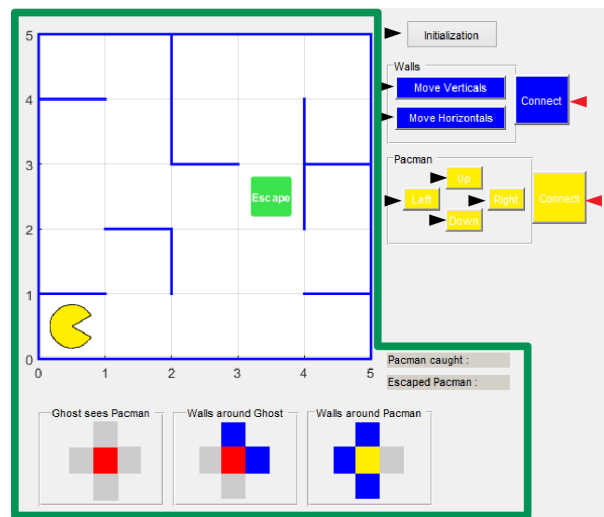


Figure 5.1 button's type of GUI

This callback lanch updateConnexion method of [Wrapper](#) class, which modify what command are automatic.

#### Parameters

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

### 5.27.1.2 createUIEscape()

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

Creation of the graphical objects "escape".

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

**Parameters**

<i>handles</i>	structure with handles and user data (see GUIDATA)
----------------	--

**Returns**

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

**5.27.1.3 createUIPacman()**

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

Creation of the graphical object "pacman".

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

**Parameters**

<i>handles</i>	structure with handles and user data (see GUIDATA)
----------------	--

**Returns**

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

**5.27.1.4 createUIWalls()**

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

Creation of the graphical objects "walls".

The walls are created as two line elements matrix. They are stored into handles in 'walls'.

The first matrix is for the verticals walls and named 'horizontals' and the second called 'verticals' for the verticals walls.

All possible walls are created and it is by making them visible or invisible that they appear or disappear.

**Parameters**

<i>handles</i>	structure with handles and user data (see GUIDATA)
----------------	--

**Returns**

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



## 5.27.1.5 figure\_Laby()

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

## figure\_Laby.m

Script linked to the graphical interface which contain all the graphical functions. This file contain also the instance of [Wrapper](#) class. All the handles of graphical elements and instance of class are stored into the "handles" structure. function call when figure\_Laby si open. It's initialize the UI.

## Parameters

<i>varargin</i>	Several inputs.
-----------------	-----------------

## Returns

varargout Several Outputs.

## 5.27.1.6 figure\_Laby\_OpeningFcn()

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

initialization function.

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

## Parameters

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

## 5.27.1.7 figure\_Laby\_OutputFcn()

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

Automatic generated function by GUI.

**Parameters**

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

**Returns**

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

**5.27.1.8 isOne()**

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

**5.27.1.9 resetUIConnection()**

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

**5.27.1.10 ui\_Callback()**

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

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

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

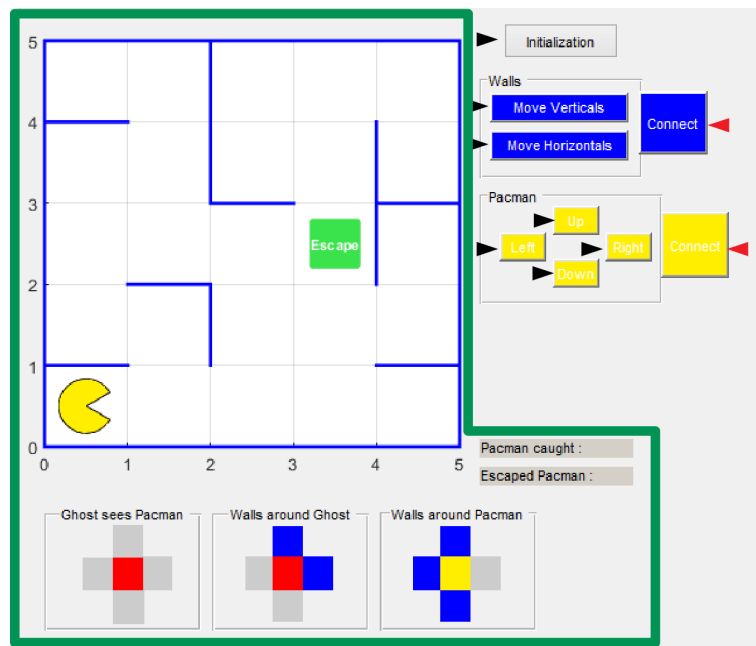


Figure 5.2 button's type of GUI

This callback launch orderer method of [Wrapper](#) class, which allows the simulation to evolve.

#### Parameters

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

#### 5.27.1.11 updatePresenceDetectorDisplay()

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

#### 5.27.1.12 updateUI()

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

This function update all graphicals element who can change.

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

## Parameters

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

5.27.1.13 `updateUIActiveCammand()`

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

Update visibility of control panel, connection and step button.

This function show or hide the control's panels and the connection's buttons according whit who will move. It also show step button if a command is connected.

Example : if is pacman time to move and command is not connected, this function hide walls and step element and show pacman one's.

## Parameters

<i>handles</i>	structure with handles and user data (see GUIDATA)
----------------	--

5.27.1.14 `updateUIButton()`

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

Show the needed moving buttons.

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

## Parameters

<i>handles</i>	structure with handles and user data (see GUIDATA)
----------------	--

5.27.1.15 `updateUIEscape()`

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

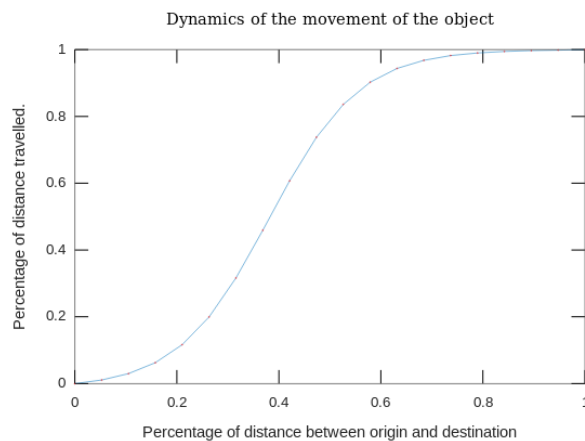
## 5.27.1.16 updateUIPlayer()

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

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

This function, with the actual position (present in the handles) and the new one as a input, move object.

The dynamics of movement is defined by this fonction  $out(t) = \frac{\frac{om+1}{om * e^{cv*t} + 1} - 1}{om}$  for  $t \in [0, 1]$ ,  $om = 72.89105$  and  $cv = -11.27357$ .



**Figure 5.3 Dynamics of movement**

## Parameters

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

## 5.27.1.17 updateUIWalls()

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

## 5.27.1.18 updateUIWallsAround()

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

## 5.28 ModelLaby.m File Reference

### Classes

- class [ModelLaby](#)  
*Class which contains the "fmg" structure of the labyrinth for 1 player.*

## 5.29 ModelPacman.m File Reference

Contain ghost Pacman control.

### Classes

- class [ModelPacman](#)  
*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  $P(D) > P(B) > P(H) > P(G)$   
.*

### 5.29.1 Detailed Description

Contain ghost Pacman control.

## 5.30 ModelSED.m File Reference

abstract Class who contain the structure of a "fmg" implementation

### Classes

- class [ModelSED](#)  
*Input : necessary information for compute the next state of the model  
  
Output : output's action of the model  
.*

### 5.30.1 Detailed Description

abstract Class who contain the structure of a "fmg" implementation

## 5.31 ModelWalls.m File Reference

Contain wall movement command.

### Classes

- class [ModelWalls](#)
  - Input : No need*
  - Output : [UPwalls , RIGHTwalls]*
  - State : contain the last move ( 0 = up ; 1 = right)*
  - This command do the sequence walls Right -> walls down*
  - .

### 5.31.1 Detailed Description

Contain wall movement command.

## 5.32 setColor.m File Reference

### Functions

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

### 5.32.1 Function Documentation

#### 5.32.1.1 setColor()

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

### 5.33 Simulation.m File Reference

### 5.34 StopCondition.m File Reference

#### Classes

- class [StopCondition](#)

### 5.35 visupacman.m File Reference

### 5.36 visupacman2.m File Reference

### 5.37 wallsBorder.m File Reference

#### Functions

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

#### 5.37.1 Function Documentation

##### 5.37.1.1 wallsBorder()

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

### 5.38 Wrapper.m File Reference

#### Classes

- class [Wrapper](#)



# Index

- accessibilityAutomate
  - AutomateGraph, 8
- adaptTourLangage
  - AutomateGraph, 8
- addWord2Langage
  - AutomateGraph, 8
- affichage
  - Labyrinthe, 15
- AjoutTransitionStable
  - Automaton, 12
- AutomateAccessible
  - Automaton, 12
- AutomateAccessibleIncertain
  - Automaton, 12
- AutomateGraph, 7
  - accessibilityAutomate, 8
  - adaptTourLangage, 8
  - addWord2Langage, 8
  - AutomateGraph, 8
  - export2DESUMA, 8
  - FSM2Automata, 9
  - langage, 10
  - matrices2vector, 9
  - matrixTrans, 10
  - PathRecherche, 9
  - state, 10
  - structAutomata2vectorAutomata, 9
  - transition, 10
  - vector, 10
  - vector2matrices, 9
  - vector2structAutomata, 9
- Automaton, 11
  - AjoutTransitionStable, 12
  - AutomateAccessible, 12
  - AutomateAccessibleIncertain, 12
  - Automaton, 11
  - CreationFonctionTransitionEnsembleDesParties, 12
  - Ensemble2Etats, 12
  - EnsemblesContenantEtat, 12
  - Etats2Ensemble, 13
  - Evolution, 13
  - Fct, 13
  - MatricesTransition, 13
  - NomsEvenements, 13
  - PathRecherche, 13
- automaton/AutomateGraph.m, 41
- automaton/ParrallelComposition.m, 47
- automaton/mainLaby.m, 41
- automaton/modelGenerator/AutomatonScheduling↔
  - Creation.m, 41
- automaton/modelGenerator/AutomatonStrutureLaby↔
  - Creation.m, 41
- automaton/modelGenerator/AutomatonWallsContraints↔
  - Creation.m, 42
- automaton/modelGenerator/Plan\_desumaFunctions.m, 43
- automaton/modelGenerator/SaveDESUMAFFile.m, 44
- automaton/modelGenerator/generer\_lab.m, 42
- automaton/modelGenerator/modelGenerator.m, 43
- automaton/modelGenerator/writeStates.m, 44
- automaton/modelGenerator/writeTransitions.m, 45
- automaton/optimalCommand/ParcourirMatrices↔
  - Transitions.m, 47
- automaton/optimalCommand/creationMatricetransition.↔
  - m, 45
- automaton/optimalCommand/getStateTransitionFSM.m, 45
- automaton/optimalCommand/getStateTransitionTXT.m, 46
- automaton/optimalCommand/main.m, 46
- automaton/optimalCommand/optimalCommand.m, 46
- automaton/optimalCommand/rafineAutomaton.m, 47
- automaton/optimalCommand/rafineAutomatonClass.m, 47
- automaton\_nd/Automaton.m, 48
- automaton\_nd/Labyrinthe.m, 48
- automaton\_nd/modi\_main.m, 48
- AutomatonSchedulingCreation.m
  - function, 41
- AutomatonStrutureLabyCreation
  - AutomatonStrutureLabyCreation.m, 42
  - Plan\_desumaFunctions.m, 43
- AutomatonStrutureLabyCreation.m
  - AutomatonStrutureLabyCreation, 42
- AutomatonWallsContraintsCreation
  - AutomatonWallsContraintsCreation.m, 42
- AutomatonWallsContraintsCreation.m
  - AutomatonWallsContraintsCreation, 42
- commandPacman
  - Wrapper, 38
- commandWalls
  - Wrapper, 38
- connect\_Callback
  - figure\_Laby.m, 50
- CreatePituresAndVideo
  - CreatePituresAndVideo.m, 48
- CreatePituresAndVideo.m, 48

- CreatePituresAndVideo, 48
- CreatePituresAndVideo\_textured
  - CreatePituresAndVideo\_textured.m, 49
- CreatePituresAndVideo\_textured.m, 49
  - CreatePituresAndVideo\_textured, 49
- createUIEscape
  - figure\_Laby.m, 50
- createUIPacman
  - figure\_Laby.m, 52
- createUIWalls
  - figure\_Laby.m, 52
- CreationFonctionTransitionEnsembleDesParties
  - Automaton, 12
- creationMatricetransition
  - creationMatricetransition.m, 45
- creationMatricetransition.m
  - creationMatricetransition, 45
- Ensemble2Etats
  - Automaton, 12
- EnsemblesContenantEtat
  - Automaton, 12
- Etats2Ensemble
  - Automaton, 13
- Etats\_Finaux
  - Labyrinthe, 16
- Etats\_Initiaux
  - Labyrinthe, 16
- Evolution
  - Automaton, 13
- export2DESUMA
  - AutomateGraph, 8
- f
  - ModelLaby, 19
  - ModelPacman, 23
  - ModelSED, 26
  - ModelWalls, 30
  - StopCondition, 34
- FSM2Automata
  - AutomateGraph, 9
- Fct
  - Automaton, 13
- figure\_Laby
  - figure\_Laby.m, 52
- figure\_Laby.m, 49
  - connect\_Callback, 50
  - createUIEscape, 50
  - createUIPacman, 52
  - createUIWalls, 52
  - figure\_Laby, 52
  - figure\_Laby\_OpeningFcn, 53
  - figure\_Laby\_OutputFcn, 53
  - isOne, 54
  - resetUIConnection, 54
  - ui\_Callback, 54
  - updatePresenceDetectorDisplay, 55
  - updateUIActiveCammand, 56
  - updateUIButton, 56
  - updateUIEscape, 56
  - updateUIPlayer, 56
  - updateUIWalls, 57
  - updateUIWallsAround, 57
  - updateUI, 55
- figure\_Laby\_OpeningFcn
  - figure\_Laby.m, 53
- figure\_Laby\_OutputFcn
  - figure\_Laby.m, 53
- function
  - AutomatonSchedulingCreation.m, 41
  - Plan\_desumaFunctions.m, 43
- g
  - ModelLaby, 19
  - ModelPacman, 24
  - ModelSED, 27
  - ModelWalls, 31
  - StopCondition, 34
- generer\_lab
  - generer\_lab.m, 42
- generer\_lab.m
  - generer\_lab, 42
- get\_out
  - Wrapper, 37
- get\_stop
  - Wrapper, 37
- getStateTransitionFSM.m
  - getStateTransitionFSM, 45
- getStateTransitionFSM
  - getStateTransitionFSM.m, 45
- getStateTransitionTXT.m
  - getStateTransitionTXT, 46
- getStateTransitionTXT
  - getStateTransitionTXT.m, 46
- handle, 14
- i
  - ModelWalls, 31
- in
  - Wrapper, 38
- incertain
  - Labyrinthe, 15
- init
  - Wrapper, 37
- initialState
  - ModelLaby, 20
  - ModelPacman, 25
  - ModelSED, 28
  - ModelWalls, 32
  - StopCondition, 35
- isOne
  - figure\_Laby.m, 54
- jusqu\_au\_mur
  - Labyrinthe, 15
- Labyrinthe, 14

- affichage, 15
- Etats\_Finaux, 16
- Etats\_Initiaux, 16
- incertain, 15
- jusqu\_au\_mur, 15
- Labyrinthe, 15
- MursHorizontaux, 16
- MursVerticaux, 16
- Pas\_a\_pas, 16
- langage
  - AutomateGraph, 10
- m
  - ModelLaby, 19
  - ModelPacman, 24
  - ModelSED, 27
  - ModelWalls, 31
  - StopCondition, 35
- main.m, 46
- matrices2vector
  - AutomateGraph, 9
- MatricesTransition
  - Automaton, 13
- matrixTrans
  - AutomateGraph, 10
- ModelLaby, 17
  - f, 19
  - g, 19
  - initialState, 20
  - m, 19
  - ModelLaby, 18
  - presentState, 20
- modelLaby
  - Wrapper, 38
- ModelLaby.m, 58
- ModelPacman, 21
  - f, 23
  - g, 24
  - initialState, 25
  - m, 24
  - ModelPacman, 23
  - presentState, 25
- ModelPacman.m, 58
- ModelSED.m, 58
- ModelSED, 25
  - f, 26
  - g, 27
  - initialState, 28
  - m, 27
  - presentState, 28
- ModelWalls, 28
  - f, 30
  - g, 31
  - i, 31
  - initialState, 32
  - m, 31
  - ModelWalls, 30
  - presentState, 32
  - val, 32
- ModelWalls.m, 59
- MursHorizontaux
  - Labyrinthe, 16
- MursVerticaux
  - Labyrinthe, 16
- NomsEvenements
  - Automaton, 13
- optimalCommand
  - optimalCommand.m, 46
- optimalCommand.m
  - optimalCommand, 46
- orderer
  - Wrapper, 37
- out
  - Wrapper, 38
- pacmanBit
  - Wrapper, 38
- ParcourirMatricesTransitions
  - ParcourirMatricesTransitions.m, 47
- ParcourirMatricesTransitions.m
  - ParcourirMatricesTransitions, 47
- ParrallelComposition
  - ParrallelComposition.m, 48
- ParrallelComposition.m
  - ParrallelComposition, 48
- Pas\_a\_pas
  - Labyrinthe, 16
- PathResearche
  - AutomateGraph, 9
  - Automaton, 13
- Plan\_desumaFunctions.m
  - AutomatonStrutureLabyCreation, 43
  - function, 43
  - SaveDESUMAFFile, 43
  - writeStates, 43
  - writeTransitions, 43
- presentState
  - ModelLaby, 20
  - ModelPacman, 25
  - ModelSED, 28
  - ModelWalls, 32
  - StopCondition, 35
- rafineAutomatonClass
  - rafineAutomatonClass.m, 47
- rafineAutomatonClass.m
  - rafineAutomatonClass, 47
- resetUIConnection
  - figure\_Laby.m, 54
- SaveDESUMAFFile
  - Plan\_desumaFunctions.m, 43
  - SaveDESUMAFFile.m, 44
- SaveDESUMAFFile.m
  - SaveDESUMAFFile, 44
- setColor

- setColor.m, 59
- setColor.m, 59
  - setColor, 59
- Simulation.m, 60
- state
  - AutomateGraph, 10
- stop
  - Wrapper, 38
- StopCondition, 32
  - f, 34
  - g, 34
  - initialState, 35
  - m, 35
  - presentState, 35
  - StopCondition, 33
- stopCondition
  - Wrapper, 38
- StopCondition.m, 60
- structAutomata2vectorAutomata
  - AutomateGraph, 9
- transition
  - AutomateGraph, 10
- ui\_Callback
  - figure\_Laby.m, 54
- updateConnexion
  - Wrapper, 37
- updatePresenceDetectorDisplay
  - figure\_Laby.m, 55
- updateUIActiveCammand
  - figure\_Laby.m, 56
- updateUIButton
  - figure\_Laby.m, 56
- updateUIEscape
  - figure\_Laby.m, 56
- updateUIPlayer
  - figure\_Laby.m, 56
- updateUIWalls
  - figure\_Laby.m, 57
- updateUIWallsAround
  - figure\_Laby.m, 57
- updateUI
  - figure\_Laby.m, 55
- val
  - ModelWalls, 32
- vector
  - AutomateGraph, 10
- vector2matrices
  - AutomateGraph, 9
- vector2structAutomata
  - AutomateGraph, 9
- visupacman.m, 60
- visupacman2.m, 60
- wallsBit
  - Wrapper, 39
- wallsBorder
  - wallsBorder.m, 60
- wallsBorder.m, 60
  - wallsBorder, 60
- whoPlay
  - Wrapper, 39
- Wrapper, 36
  - commandPacman, 38
  - commandWalls, 38
  - get\_out, 37
  - get\_stop, 37
  - in, 38
  - init, 37
  - modellaby, 38
  - orderer, 37
  - out, 38
  - pacmanBit, 38
  - stop, 38
  - stopCondition, 38
  - updateConnexion, 37
  - wallsBit, 39
  - whoPlay, 39
  - Wrapper, 36
- Wrapper.m, 60
- writeStates
  - Plan\_desumaFunctions.m, 43
  - writeStates.m, 44
- writeStates.m
  - writeStates, 44
- writeTransitions
  - Plan\_desumaFunctions.m, 43
  - writeTransitions.m, 45
- writeTransitions.m
  - writeTransitions, 45