

## Labyrinth - Two Players 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	handle Class Reference . . . . .	7
4.2	ModelCommand Class Reference . . . . .	8
4.2.1	Member Function Documentation . . . . .	8
4.2.1.1	f() . . . . .	8
4.2.1.2	g() . . . . .	9
4.2.1.3	m() . . . . .	9
4.2.2	Member Data Documentation . . . . .	9
4.2.2.1	Down . . . . .	9
4.2.2.2	knowCompart . . . . .	9
4.2.2.3	Left . . . . .	9
4.2.2.4	presentState . . . . .	9
4.2.2.5	Right . . . . .	9
4.2.2.6	sizeTab . . . . .	10
4.2.2.7	Up . . . . .	10
4.3	ModelGhost Class Reference . . . . .	10

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

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

4.7.4.1	i	35
4.7.4.2	initialState	35
4.7.4.3	presentState	36
4.7.4.4	val	36
4.8	StopCondition Class Reference	36
4.8.1	Detailed Description	38
4.8.2	Constructor & Destructor Documentation	38
4.8.2.1	StopCondition()	38
4.8.3	Member Function Documentation	38
4.8.3.1	f() [1/2]	38
4.8.3.2	f() [2/2]	39
4.8.3.3	g()	39
4.8.3.4	m()	40
4.8.4	Member Data Documentation	40
4.8.4.1	initialState	40
4.8.4.2	presentState	41
4.9	Wrapper Class Reference	41
4.9.1	Detailed Description	42
4.9.2	Constructor & Destructor Documentation	42
4.9.2.1	Wrapper()	42
4.9.3	Member Function Documentation	43
4.9.3.1	get_out()	43
4.9.3.2	get_stop()	43
4.9.3.3	init()	44
4.9.3.4	orderer()	44
4.9.3.5	updateConnexion()	44
4.9.4	Member Data Documentation	45
4.9.4.1	commandPacman	45
4.9.4.2	commandWalls	45
4.9.4.3	ghostBit	45
4.9.4.4	in	45
4.9.4.5	modelLaby	46
4.9.4.6	out	46
4.9.4.7	pacmanBit	46
4.9.4.8	stop	46
4.9.4.9	stopCondition	46
4.9.4.10	wallsBit	46
4.9.4.11	whoPlay	46

<b>5 File Documentation</b>	<b>47</b>
5.1 CreatePituresAndVideo.m File Reference	47
5.1.1 Function Documentation	47
5.1.1.1 CreatePituresAndVideo()	47
5.2 CreatePituresAndVideo_textured.m File Reference	47
5.2.1 Function Documentation	47
5.2.1.1 CreatePituresAndVideo_textured()	48
5.3 figure_Laby.m File Reference	48
5.3.1 Function Documentation	49
5.3.1.1 connect_Callback()	49
5.3.1.2 createUIEscape()	49
5.3.1.3 createUIGhost()	51
5.3.1.4 createUIPacman()	51
5.3.1.5 createUIWalls()	52
5.3.1.6 figure_Laby()	52
5.3.1.7 figure_Laby_OpeningFcn()	52
5.3.1.8 figure_Laby_OutputFcn()	53
5.3.1.9 isOne()	53
5.3.1.10 resetUIConnection()	54
5.3.1.11 ui_Callback()	54
5.3.1.12 updatePresenceDetectorDisplay()	55
5.3.1.13 updateUI()	56
5.3.1.14 updateUIActiveCammand()	56
5.3.1.15 updateUIButton()	56
5.3.1.16 updateUICaught()	57
5.3.1.17 updateUIEscape()	57
5.3.1.18 updateUIPlayer()	58
5.3.1.19 updateUIWalls()	59
5.3.1.20 updateUIWallsAround()	59
5.4 main.m File Reference	59

5.5	<a href="#">matrixAllPossible.m File Reference</a>	60
5.6	<a href="#">ModelCommand.m File Reference</a>	60
5.7	<a href="#">ModelGhost.m File Reference</a>	60
5.8	<a href="#">ModelLaby.m File Reference</a>	60
5.9	<a href="#">ModelPacman.m File Reference</a>	60
5.10	<a href="#">ModelSED.m File Reference</a>	60
5.11	<a href="#">ModelWalls.m File Reference</a>	60
5.12	<a href="#">setColor.m File Reference</a>	61
5.12.1	<a href="#">Function Documentation</a>	61
5.12.1.1	<a href="#">setColor()</a>	61
5.13	<a href="#">Simulation.m File Reference</a>	61
5.14	<a href="#">Simulation2_allpossiblewalls.m File Reference</a>	61
5.15	<a href="#">StopCondition.m File Reference</a>	61
5.16	<a href="#">visupacman.m File Reference</a>	61
5.17	<a href="#">visupacman2.m File Reference</a>	61
5.18	<a href="#">wallsBorder.m File Reference</a>	61
5.18.1	<a href="#">Function Documentation</a>	62
5.18.1.1	<a href="#">wallsBorder()</a>	62
5.19	<a href="#">Wrapper.m File Reference</a>	63
	<b>Index</b>	<b>65</b>



# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

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



## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">handle</a> . . . . .	7
<a href="#">ModelCommand</a> . . . . .	8
<a href="#">ModelGhost</a> Contain ghost movement control . . . . .	10
<a href="#">ModelLaby</a> Class which contains the "fmg" structure of the labyrinth for 2 players . . . . .	16
<a href="#">ModelPacman</a> Contain Pacman movement control . . . . .	22
<a href="#">ModelSED</a> Abstract Class who contain the structure of a "fmg" implementation . . . . .	28
<a href="#">ModelWalls</a> Contain the wall movement command . . . . .	31
<a href="#">StopCondition</a> Class used to manage shutdown conditions . . . . .	36
<a href="#">Wrapper</a> Connects all models to the display function . . . . .	41



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

CreatePituresAndVideo.m	47
CreatePituresAndVideo_textured.m	47
figure_Laby.m	48
main.m	59
matrixAllPossible.m	60
ModelCommand.m	60
ModelGhost.m	60
ModelLaby.m	60
ModelPacman.m	60
ModelSED.m	60
ModelWalls.m	60
setColor.m	61
Simulation.m	61
Simulation2_allpossiblewalls.m	61
StopCondition.m	61
visupacman.m	61
visupacman2.m	61
wallsBorder.m	61
Wrapper.m	63

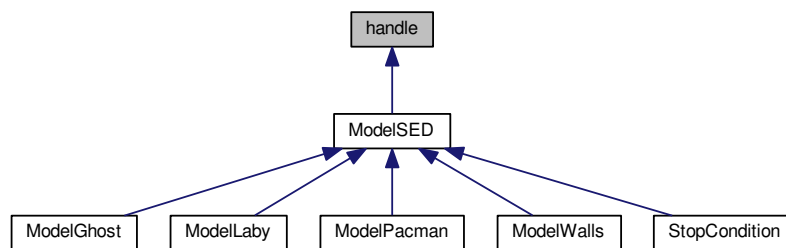


## Chapter 4

# Class Documentation

### 4.1 handle Class Reference

Inheritance diagram for handle:

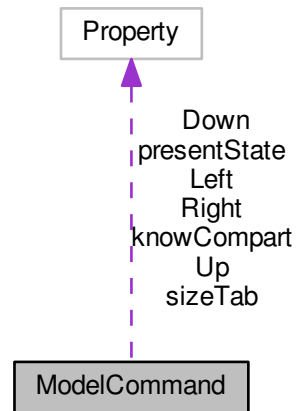


The documentation for this class was generated from the following file:

- [ModelSED.m](#)

## 4.2 ModelCommand Class Reference

Collaboration diagram for ModelCommand:



### Public Member Functions

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

### Public Attributes

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

### 4.2.1 Member Function Documentation

#### 4.2.1.1 f()

```

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



#### 4.2.1.2 g()

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

#### 4.2.1.3 m()

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

### 4.2.2 Member Data Documentation

#### 4.2.2.1 Down

Property Down

#### 4.2.2.2 knowCompart

Property knowCompart

#### 4.2.2.3 Left

Property Left

#### 4.2.2.4 presentState

Property presentState

#### 4.2.2.5 Right

Property Right

#### 4.2.2.6 sizeTab

Property sizeTab

#### 4.2.2.7 Up

Property Up

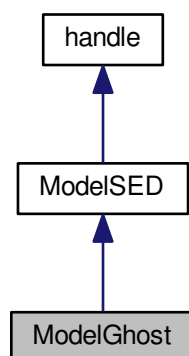
The documentation for this class was generated from the following file:

- [ModelCommand.m](#)

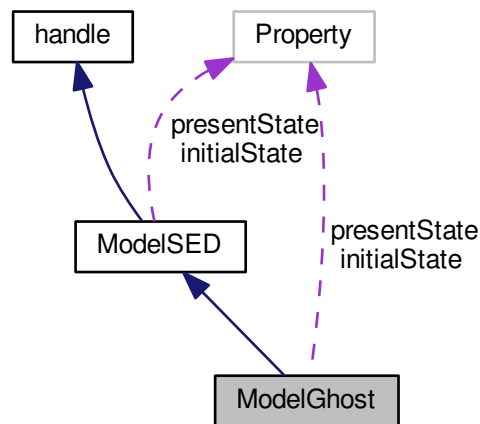
### 4.3 ModelGhost Class Reference

Contain ghost movement control.

Inheritance diagram for ModelGhost:



Collaboration diagram for ModelGhost:



## Public Member Functions

- function **ModelGhost** (in initialValue)  
*Class constructor.*
- function **f** (in obj, in in, in in\_view, in wallsV, in wallsH, in ghost\_position)  
*Compute the evolution of the command.*
- function **m** (in obj, in nextState, in init)  
*Memory method update the state of the command.*
- function **g** (in obj)  
*Create the outputs.*
- virtual **f** (in obj, in in)  
*Compute the evolution of the model.*

## Public Attributes

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

### 4.3.1 Detailed Description

Contain ghost movement control.

You can change here Pacman's command.

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

0 = move not possible ; 1 = move possible

( Wout{7} )

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

in: Walls around ghost  
 1 up  
 2 down  
 4 right

in\_view: Ghost sees Pacman  
 1 Up  
 2 Down  
 3 Left  
 4 Right  
 state :

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

## 4.3.2 Constructor & Destructor Documentation

### 4.3.2.1 ModelGhost()

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

Class constructor.

#### Parameters

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

#### Returns

instance of the [ModelGhost](#) class.

## 4.3.3 Member Function Documentation

### 4.3.3.1 f() [1/2]

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

Compute the evolution of the model.

## Parameters

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

## Return values

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

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

## 4.3.3.2 f() [2/2]

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

Compute the evolution of the command.

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

## Parameters

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

## Return values

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

## 4.3.3.3 g()

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

Create the outputs.

**Parameters**

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

**Return values**

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

Reimplemented from [ModelSED](#).

**4.3.3.4 m()**

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

Memory method update the state of the command.

**Parameters**

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

**Returns**

instance of the class updated

Reimplemented from [ModelSED](#).

**4.3.4 Member Data Documentation****4.3.4.1 initialState**

Property `initialState`

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

#### 4.3.4.2 presentState

Property `presentState`

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

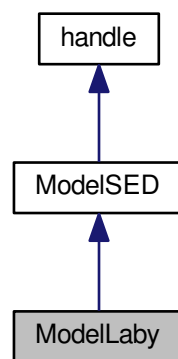
The documentation for this class was generated from the following file:

- [ModelGhost.m](#)

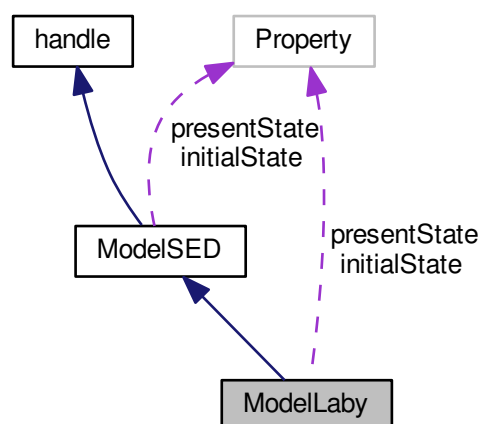
## 4.4 ModelLaby Class Reference

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

Inheritance diagram for ModelLaby:



Collaboration diagram for ModelLaby:





## Public Member Functions

- function [Modellaby](#) (in wallsV\_init, in wallsH\_init, in pacman\_init, in ghost\_init, in escape\_init, in caught\_init)  
*Class constructor of.*
- function [f](#) (in obj, in in)  
*Compute the evolution of the model.*
- function [m](#) (in obj, in nextState, in init)  
*Memory method update the state of the command.*
- function [g](#) (in obj)  
*Create the outputs in a 1x9 cell-array.*
- function [sameX\\_position](#) (in obj)  
*Method to analyze Ghost and Pacman Position.*
- function [sameY\\_position](#) (in obj)  
*Method to analyze Ghost and Pacman Position.*
- function [wallsVBetween](#) (in obj, in obj1, in obj2)  
*Method to analyze if a Vertical wall is between 2 objects.*
- function [wallsHBetween](#) (in obj, in obj1, in obj2)  
*Method to analyze if a Horizontal wall is between 2 objects.*
- function [wallsVBetweenOne](#) (in obj, in obj1, in obj2)  
*Method to analyze if a Horizontal wall is between 2 objects side by side.*
- function [wallsHBetweenOne](#) (in obj, in obj1, in obj2)  
*Method to analyze if a Horizontal wall is between 2 objects side by side.*

## Public Attributes

- Property [presentState](#)  
*Data Structure of the current state of Labyrinth.  
It contains "wallsV", "wallsH" (2 matrix for the walls), "ghost", "pacman" and "escape", a Cartesian position of current position of ghost, pacman and escape.  
There is also 3 vectors : 'wallsAroundPacman', 'wallsAroundGhost' and 'ghostSeesPacman' A vector indicating the presence of a wall around the Pacman and ghost for the 4 directions Up Down Left Right.*
- Property [initialState](#)  
*Data Structure of the initial state of Labyrinth. It contains "wallsV", "wallsH" (2 matrix for the walls), "escape" and "pacman", a Cartesian position of current position of escape and pacman and 'wallsAroundPacman' A vector indicating the presence of a wall around the Pacman for the 4 directions Up Down Left Right.*

### 4.4.1 Detailed Description

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

You can change here labyrinth's dynamic : how objects and walls are evolving in the labyrinth, not the command of then.

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

Output : output's action of the model

State : minimal information necessary who evolve

### 4.4.2 Constructor & Destructor Documentation

#### 4.4.2.1 ModelLaby()

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

Class constructor of.

##### Parameters

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

##### Returns

instance of the [ModelLaby](#) class.

#### 4.4.3 Member Function Documentation

##### 4.4.3.1 f()

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

Compute the evolution of the model.

##### Parameters

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

##### Return values

<i>nextState</i>	Next instance of the <a href="#">ModelLaby</a> class.
------------------	---

Reimplemented from [ModelISED](#).

## 4.4.3.2 g()

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

Create the outputs in a 1x9 cell-array.

## Parameters

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

## Return values

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

Reimplemented from [ModelSED](#).

## 4.4.3.3 m()

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

Memory method update the state of the command.

## Parameters

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

## Returns

instance of the class updated

Reimplemented from [ModelSED](#).

## 4.4.3.4 sameX\_position()

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

Method to analyze Ghost and Pacman Position.

**Parameters**

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

**4.4.3.5 sameY\_position()**

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

Method to analyze Ghost and Pacman Position.

**Parameters**

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

**4.4.3.6 wallsHBetween()**

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

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

**Parameters**

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

**4.4.3.7 wallsHBetweenOne()**

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

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

**Parameters**

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

**4.4.3.8 wallsVBetween()**

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

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

**Parameters**

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

**4.4.3.9 wallsVBetweenOne()**

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

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

**Parameters**

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

**4.4.4 Member Data Documentation****4.4.4.1 initialState**

Property `initialState`

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

#### 4.4.4.2 presentState

Property presentState

Data Structure of the current state of Labyrinth.

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

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

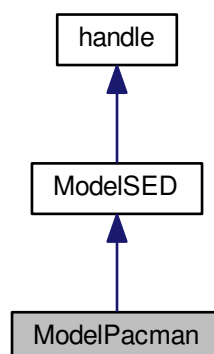
The documentation for this class was generated from the following file:

- [ModelLaby.m](#)

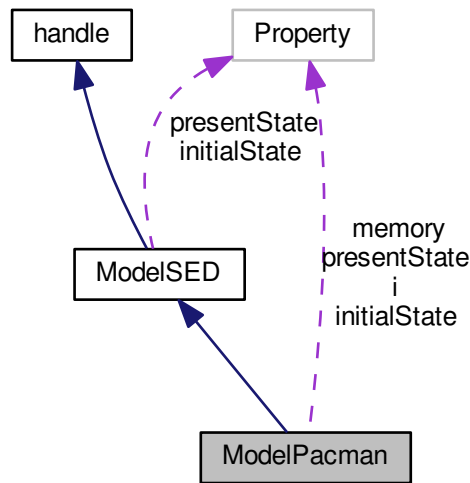
## 4.5 ModelPacman Class Reference

Contain Pacman movement control.

Inheritance diagram for ModelPacman:



Collaboration diagram for ModelPacman:



## Public Member Functions

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

## Public Attributes

- Property `presentState`  
*This is the state of the command in the present moment.*
- Property `initialState`  
*This is the state of the command in the initialization and when it's reseted.*
- Property `memory`  
*This is another state who deed to be include.*
- Property `i`

### 4.5.1 Detailed Description

Contain Pacman movement control.

You can change here Pacman's command.

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

→ 0 = move not possible ; 1 = move possible

( Wout{7} )

Output : Pacman's moves 1 : pacmanLeftBut, ( Wout(3) )

2 : pacmanUpBut, ( Wout(1) )

3 : pacmanRightBut, ( Wout(4) )

4 : pacmanDownBut, ( Wout(2) )

( Win( 4:7) of wrapper )

Input : Walls around Pacman

1 up

2 down

3 left

4 right

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

### 4.5.2 Constructor & Destructor Documentation

#### 4.5.2.1 ModelPacman()

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

Class constructor.

#### Parameters

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

#### Returns

instance of the [ModelPacman](#) class.

### 4.5.3 Member Function Documentation

#### 4.5.3.1 f()

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



Compute the evolution of the command.

**Parameters**

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

**Return values**

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

Reimplemented from [ModelSED](#).

**4.5.3.2 g()**

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

Create the outputs.

**Parameters**

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

**Return values**

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

Reimplemented from [ModelSED](#).

**4.5.3.3 m()**

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

Memory method, update the state of the command.

**Parameters**

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

### Returns

instance of the class updated

Reimplemented from [ModelSED](#).

## 4.5.4 Member Data Documentation

### 4.5.4.1 i

Property i

### 4.5.4.2 initialState

Property initialState

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

### 4.5.4.3 memory

Property memory

This is another state who deed to be include.

### 4.5.4.4 presentState

Property presentState

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

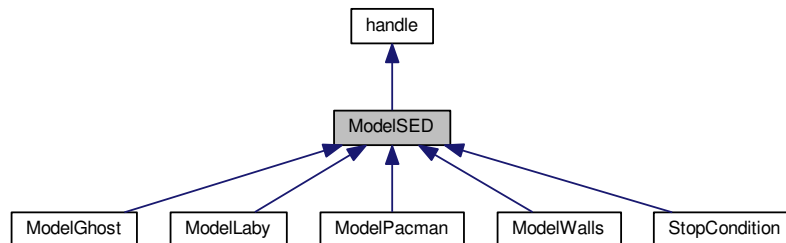
The documentation for this class was generated from the following file:

- [ModelPacman.m](#)

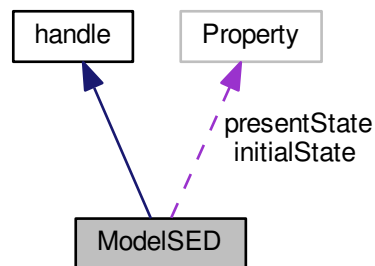
## 4.6 ModelSED Class Reference

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

Inheritance diagram for ModelSED:



Collaboration diagram for ModelSED:



### Public Member Functions

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

### Public Attributes

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

### 4.6.1 Detailed Description

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

This class is used for give a general definition of Model Class.

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

Output : output's action of the model

State : minimal information necessary who evolve

### 4.6.2 Member Function Documentation

#### 4.6.2.1 f()

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

Compute the evolution of the model.

##### Parameters

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

##### Return values

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

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

#### 4.6.2.2 g()

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

Create the outputs.

##### Parameters

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

**Return values**

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

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

**4.6.2.3 m()**

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

Memory method update the state of the command.

**Parameters**

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

**Returns**

instance of the class updated

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

**4.6.3 Member Data Documentation****4.6.3.1 initialState**

Property `initialState`

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

**4.6.3.2 presentState**

Property `presentState`

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

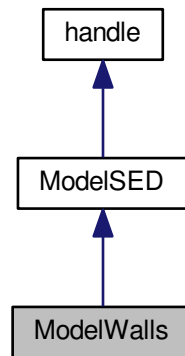
The documentation for this class was generated from the following file:

- [ModelSED.m](#)

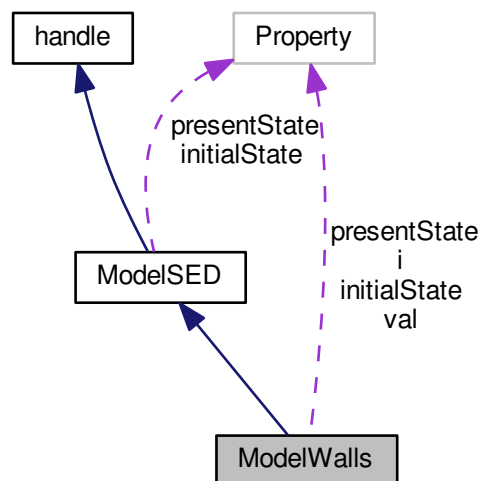
## 4.7 ModelWalls Class Reference

Contain the wall movement command.

Inheritance diagram for ModelWalls:



Collaboration diagram for ModelWalls:



### Public Member Functions

- function **ModelWalls** (in initValue)  
*Class constructor.*

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

## Public Attributes

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

### 4.7.1 Detailed Description

Contain the wall movement command.

You can change here the order in which the walls move

Input : No need

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

### 4.7.2 Constructor & Destructor Documentation

#### 4.7.2.1 ModelWalls()

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

Class constructor.

#### Parameters

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

#### Returns

instance of the `ModelWalls` class.



### 4.7.3 Member Function Documentation

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

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

Compute the evolution of the model.

##### Parameters

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

##### Return values

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

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

#### 4.7.3.2 `f()` [2/2]

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

Compute the evolution of the command.

##### Parameters

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

##### Return values

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

#### 4.7.3.3 `g()`

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

Create the outputs.

## Parameters

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

## Return values

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

Reimplemented from [ModelSED](#).

## 4.7.3.4 m()

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

Memory method update the state of the command.

## Parameters

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

## Returns

instance of the class updated

Reimplemented from [ModelSED](#).

## 4.7.4 Member Data Documentation

## 4.7.4.1 i

Property *i*

## 4.7.4.2 initialState

Property *initialState*

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

#### 4.7.4.3 presentState

Property presentState

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

#### 4.7.4.4 val

Property val

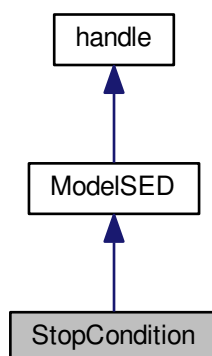
The documentation for this class was generated from the following file:

- [ModelWalls.m](#)

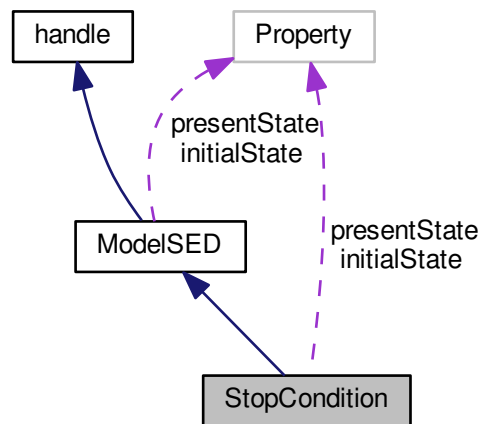
## 4.8 StopCondition Class Reference

Class used to manage shutdown conditions.

Inheritance diagram for StopCondition:



Collaboration diagram for StopCondition:



## Public Member Functions

- function [StopCondition](#) (in `initCondition`)  
*Class constructor of Instance of [StopCondition](#) Class.*
- function [f](#) (in `obj`, in `noEscape`, in `caught`, in `pacmanWallsBreak`, in `ghostWallsBreak`)  
*Compute the evolution of the model.*
- function [m](#) (in `obj`, in `nextState`, in `init`)  
*Memory method.*
- function [g](#) (in `obj`)  
*Create the outputs in a vector with 4 parameters.*
- virtual [f](#) (in `obj`, in `in`)  
*Compute the evolution of the model.*

## Public Attributes

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

### 4.8.1 Detailed Description

Class used to manage shutdown conditions.

Labyrinth shutdown conditions model.

You can modify the shutdown conditions here. It is developing in the same way as MODELSED, with "FMG" block. (MODELSED's legacy)

Input : walls of Pacman's  
walls of ghost's  
escape of Pacman  
CaughtBreak

Output : 1 Escape  
2 Caught  
3 pacmanWallsBreak  
4 ghostWallsBreak

### 4.8.2 Constructor & Destructor Documentation

#### 4.8.2.1 StopCondition()

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

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

#### Parameters

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

#### Returns

instance of the [StopCondition](#) class.

### 4.8.3 Member Function Documentation

#### 4.8.3.1 f() [1/2]

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

Compute the evolution of the model.

## Parameters

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

## Return values

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

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

4.8.3.2 `f()` [2/2]

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

Compute the evolution of the model.

## Parameters

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

## Returns

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

4.8.3.3 `g()`

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

Create the outputs in a vector with 4 parameters.

## Parameters

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

**Return values**

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

Reimplemented from [ModelSED](#).

**4.8.3.4 m()**

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

Memory method.

Update the state of the command.

**Parameters**

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

**Returns**

instance of the class updated

Reimplemented from [ModelSED](#).

**4.8.4 Member Data Documentation****4.8.4.1 initialState**

Property `initialState`

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



## 4.8.4.2 presentState

Property presentState

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

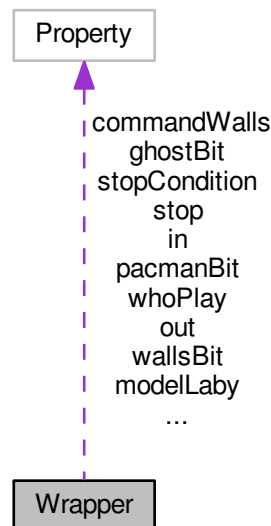
The documentation for this class was generated from the following file:

- [StopCondition.m](#)

## 4.9 Wrapper Class Reference

Connects all models to the display function.

Collaboration diagram for Wrapper:



### Public Member Functions

- function [Wrapper](#) (in inSize, in outSize, in initLaby, in initWalls, in initPac, in initGhost, in initStop)  
*Class constructor of Instance of [StopCondition](#) Class.*
- function [updateConnexion](#) (in obj, in indBit, in value)  
*Update the connection bit for connect automatic mode for Pacman and/or the walls.*
- function [init](#) (in obj)  
*Allows to completely reset the labyrinth.*
- function [orderer](#) (in obj, in vectIn)  
*Ordinate the global execution of Models.*
- function [get\\_stop](#) (in obj)  
*Return the current State of shutdown condition.*
- function [get\\_out](#) (in obj)  
*Return the current State of output cell.*

## Public Attributes

- Property [wallsBit](#)  
*Boolean connection for the walls.*
- Property [pacmanBit](#)  
*Boolean connection for the Pacman.*
- Property [ghostBit](#)  
*Boolean connection for the ghost.*
- Property [modelLaby](#)  
*contain the instance of the model of labyrinth.*
- Property [commandWalls](#)  
*contain the instance of wall's command.*
- Property [commandPacman](#)  
*contain the instance of Pacman's command.*
- Property [stopCondition](#)  
*contain the instance shutdown condition.*
- Property [in](#)  
*A integer vector who contain the state of input,.*
- Property [out](#)  
*A cell who contain the state of output,.*
- Property [stop](#)  
*A cell that contains the output of the stop conditions instance.*
- Property [whoPlay](#)  
*A increment integer that permit to know which object to play 0 = walls ; 1 = Pacman.*

### 4.9.1 Detailed Description

Connects all models to the display function.

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

### 4.9.2 Constructor & Destructor Documentation

#### 4.9.2.1 Wrapper()

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

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

**Parameters**

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

**Returns**

instance of the [Wrapper](#) class.

### 4.9.3 Member Function Documentation

#### 4.9.3.1 `get_out()`

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

Return the current State of output cell.

**Parameters**

<i>obj</i>	The instance of <a href="#">Wrapper</a> .
------------	---

**Returns**

Output cell.

#### 4.9.3.2 `get_stop()`

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

Return the current State of shutdown condition.

**Parameters**

<i>obj</i>	The instance of <a href="#">Wrapper</a> .
------------	---

**Returns**

instance of the Stop class.

**4.9.3.3 init()**

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

Allows to completely reset the labyrinth.

**Parameters**

<i>obj</i>	The instance of <a href="#">Wrapper</a> .
------------	---

**Returns**

instance of the [Wrapper](#) class.

**4.9.3.4 orderer()**

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

Ordinate the global execution of Models.

**Parameters**

<i>obj</i>	The instance of <a href="#">Wrapper</a> .
------------	---

**Returns**

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

Here a little graphic about the scheduling of the call of each model : walls > Laby > pacman > laby > ghost > laby

Every call of 'laby' implies a call of Stop Condition.

**4.9.3.5 updateConnexion()**

```
function updateConnexion (  
    in obj,
```

```
in indBit,  
in value )
```

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

#### Parameters

<i>obj</i>	The instance of <a href="#">Wrapper</a> .
<i>indBit</i>	Integer pointing the element to be connected : '1' for walls, '2' for ghost and '3' for Pacman.
<i>value</i>	Boolean indicating if the element is connected (True) or not.

#### Returns

instance of the [Wrapper](#) class.

### 4.9.4 Member Data Documentation

#### 4.9.4.1 commandPacman

Property `commandPacman`

contain the instance of Pacman's command.

#### 4.9.4.2 commandWalls

Property `commandWalls`

contain the instance of wall's command.

#### 4.9.4.3 ghostBit

Property `ghostBit`

Boolean connection for the ghost.

#### 4.9.4.4 in

Property `in`

A integer vector who contain the state of input,.

incremented by the callback or some action.

#### 4.9.4.5 modelLaby

Property modelLaby

contain the instance of the model of labyrinth.

#### 4.9.4.6 out

Property out

A cell who contain the state of output,.

incremented by the callback or some action.

#### 4.9.4.7 pacmanBit

Property pacmanBit

Boolean connection for the Pacman.

#### 4.9.4.8 stop

Property stop

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

incremented by the callback or some action.

#### 4.9.4.9 stopCondition

Property stopCondition

contain the instance shutdown condition.

#### 4.9.4.10 wallsBit

Property wallsBit

Boolean connection for the walls.

#### 4.9.4.11 whoPlay

Property whoPlay

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

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

- [Wrapper.m](#)

## Chapter 5

# File Documentation

### 5.1 CreatePituresAndVideo.m File Reference

#### Functions

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

#### 5.1.1 Function Documentation

##### 5.1.1.1 CreatePituresAndVideo()

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

### 5.2 CreatePituresAndVideo\_textured.m File Reference

#### Functions

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

#### 5.2.1 Function Documentation

### 5.2.1.1 CreatePituresAndVideo\_textured()

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

## 5.3 figure\_Laby.m File Reference

### Functions

- function [figure\\_Laby](#) (in varargin)  
*figure\_Laby.m*
- function [figure\\_Laby\\_OpeningFcn](#) (in hObject, in eventdata, in handles, in varargin)  
*initialization function.*
- function [figure\\_Laby\\_OutputFcn](#) (in hObject, in eventdata, in handles)  
*Automatic generated function by GUI.*
- function [ui\\_Callback](#) (in hObject, in eventdata, in handles)  
*Callback for all the action's buttons (see detailed explications).*
- function [connect\\_Callback](#) (in hObject, in eventdata, in handles)  
*Callback for all the connection's buttons (see detailed explications).*
- function [createUIPacman](#) (in handles)  
*Creation of the graphical object "pacman".*
- function [createUIGhost](#) (in handles)  
*Creation of the graphical object "Ghost".*
- 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 ( pacman or ghost).*
- function [updateUICaught](#) (in elementToSet, in caughtInt, in stp)  
*Update graphical static text block about how many times ghost caught pacman.*
- function [updateUIEscape](#) (in elementToSet, in boolState)  
*Update graphical static text block about escape status.*
- function [updateUIWallsAround](#) (in handles, in strElement, in wallsAround)  
*Update graphical elements for walls around pacman, ghost or ghost sees pacman.*
- function [updateUIWalls](#) (in wallsUI, in vertWalls, in horizWalls)  
*Update graphicals elements for the walls.*
- function [isOne](#) (in boolCond)  
*Return the string 'on' if the input is 1 else, return the string 'off'.*
- function [updatePresenceDetectorDisplay](#) (in elementToSet, in boolCondition)  
*Change the background color of the graphical Element according to the state of the binary condition.*
- function [resetUIConnection](#) (in handles)  
*Reset all commands connections to unconnected.*



### 5.3.1 Function Documentation

#### 5.3.1.1 connect\_Callback()

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

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

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

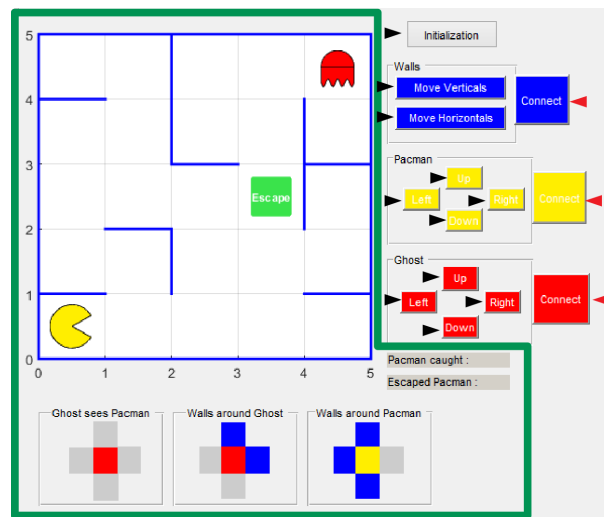


Figure 5.1 button's type of GUI

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

#### Parameters

<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)

#### 5.3.1.2 createUIEscape()

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

Creation of the graphical objects "escape".

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

**Parameters**

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

**Returns**

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

**5.3.1.3 createUIGhost()**

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

Creation of the graphical object "Ghost".

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

**Parameters**

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

**Returns**

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

**5.3.1.4 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)

### 5.3.1.5 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)

### 5.3.1.6 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.

### 5.3.1.7 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)

#### 5.3.1.8 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);

#### 5.3.1.9 isOne()

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

Return the string 'on' if the input is 1 else, return the string 'off'.

#### Parameters

<i>boolCond</i>	integer to convert
-----------------	--------------------

**Returns**

strOnOff Returned string. Can be worth "on" or "off".

**5.3.1.10 resetUIConnection()**

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

Reset all commands connections to unconnected.

This function reset wrapper's property and graphical element to unconnected state.

**Parameters**

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

**Returns**

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

**5.3.1.11 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.

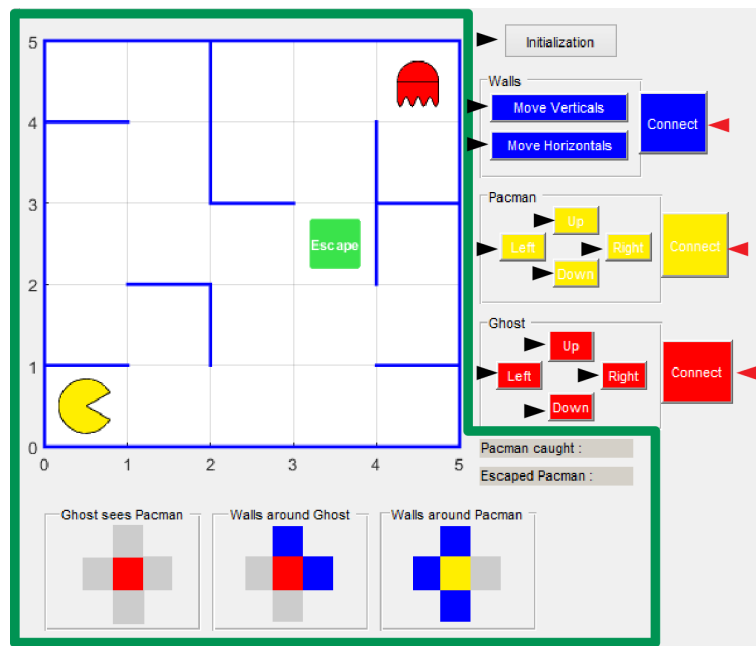


Figure 5.2 button's type of GUI

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

#### Parameters

<i>hObject</i>	handle to 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)

#### 5.3.1.12 updatePresenceDetectorDisplay()

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

Change the background color of the graphical Element according to the state of the binary condition.

This function, with a boolean condition (integer) and a handle to a graphical element update the background color. If the Boolean value is 1, the background color turns blue, otherwise it will be gray.

#### Parameters

<i>elementToSet</i>	handle to the graphical element.
<i>boolCondition</i>	input boolean condition store in a integer.

#### 5.3.1.13 updateUI()

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

This function update all graphicals element who can change.

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

##### Parameters

<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.

#### 5.3.1.14 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)
----------------	--

#### 5.3.1.15 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)
----------------	--



## 5.3.1.16 updateUICaught()

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

Update graphical static text block about how many times ghost caught pacman.

This function, with the output of the wrapper, show caughtInt into the block where is write "Pacman caught : ". In the green box in the following image :

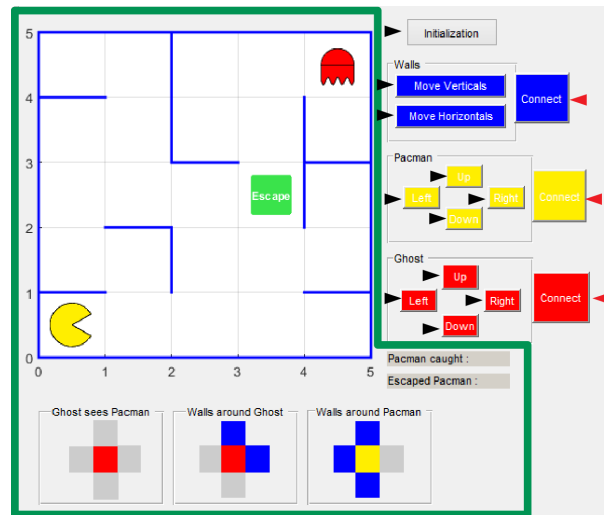


Figure 5.3 graphicals elements

If the pacman is caught enough times for stop the simulation, the background color become red.

## Parameters

<i>elementToSet</i>	handle to the graphical object for caught status.
<i>caughtInt</i>	How many times ghost caught pacman, from the output of the wrapper.
<i>stp</i>	Actual stop state. From the wrapper class.

## 5.3.1.17 updateUIEscape()

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

Update graphical static text block about escape status.

This function, with the output of the wrapper, show 'Yes' in the static text block where is write "Escaped Pacman : " if the pacman is on the escape case.  
In the green box in the following image.

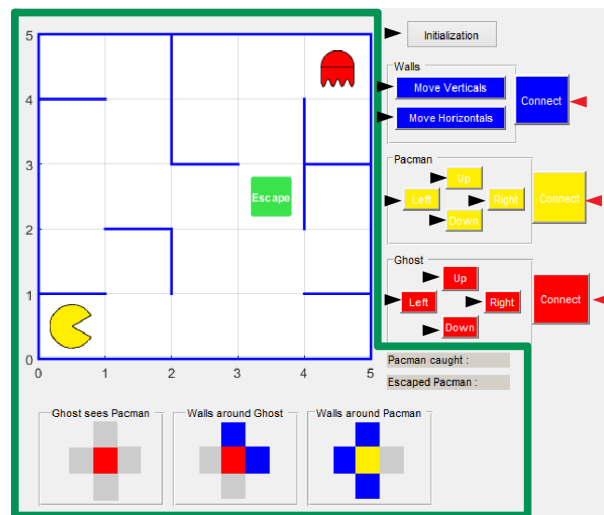


Figure 5.4 graphical elements

## Parameters

<i>elementToSet</i>	handle to the graphical object for escape status.
<i>boolState</i>	Value of the escape output of wrapper. (1 if pacman is escaped, else 0.)

## 5.3.1.18 updateUIPlayer()

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

Update graphical place of a player ( pacman or ghost).

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  $out(t) = \frac{om+1}{om*e^{cv*t}+1} - 1$  for  $t \in [0, 1]$ ,  $om = 72.89105$  and  $cv = -11.27357$ .

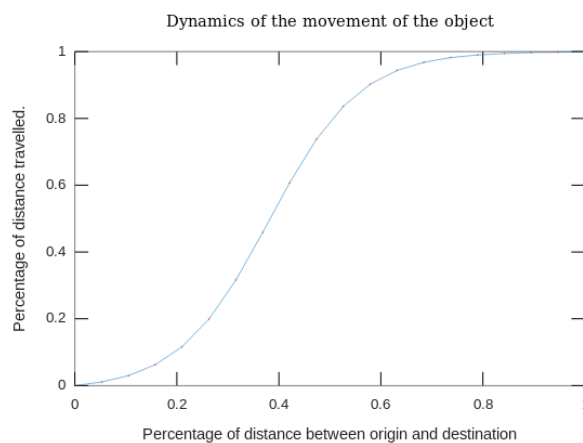


Figure 5.5 Dynamics of movement

## Parameters

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

## 5.3.1.19 updateUIWalls()

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

Update graphicals elements for the walls.

This function, with the output of the wrapper, update displayed part and hided part of the walls.

## Parameters

<i>wallsUI</i>	handle to the matrixs of graphicals elements for walls.
<i>vertWalls</i>	Corresponding part of output of wrapper for verticals walls..
<i>horizWalls</i>	Corresponding part of output of wrapper for horizontals walls.

## 5.3.1.20 updateUIWallsAround()

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

Update graphical elements for walls around pacman, ghost or ghost sees pacman.

This function, with the output of the wrapper and a handle to the graphical element, update color of gray squares that represents walls presence around the pacman, the ghost or when the ghost sees pacman.

## Parameters

<i>strElement</i>	common part of the name of the handle to the graphical object for walls around pacman.
<i>wallsAround</i>	Value of the corresponding part of output of wrapper.
<i>handles</i>	structure with handles and user data (see GUIDATA)

## 5.4 main.m File Reference

## 5.5 matrixAllPossible.m File Reference

## 5.6 ModelCommand.m File Reference

### Classes

- class [ModelCommand](#)

## 5.7 ModelGhost.m File Reference

### Classes

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

## 5.8 ModelLaby.m File Reference

### Classes

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

## 5.9 ModelPacman.m File Reference

### Classes

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

## 5.10 ModelSED.m File Reference

### Classes

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

## 5.11 ModelWalls.m File Reference

### Classes

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

## 5.12 setColor.m File Reference

### Functions

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

### 5.12.1 Function Documentation

#### 5.12.1.1 setColor()

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

## 5.13 Simulation.m File Reference

## 5.14 Simulation2\_allpossiblewalls.m File Reference

## 5.15 StopCondition.m File Reference

### Classes

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

## 5.16 visupacman.m File Reference

## 5.17 visupacman2.m File Reference

## 5.18 wallsBorder.m File Reference

### Functions

- function [wallsBorder](#) (in `walls`)  
[wallsBorder.m](#)

## 5.18.1 Function Documentation

### 5.18.1.1 wallsBorder()

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

[wallsBorder.m](#)

Function to add the outlines of the labyrinth to a wall matrix. Function.

## Parameters

<code>walls</code>	Wall's matrix.
--------------------	----------------

## Returns

walls Wall's matrix with border.

## 5.19 Wrapper.m File Reference

## Classes

- class [Wrapper](#)  
*Connects all models to the display function.*





# Index

- commandPacman
  - Wrapper, 45
- commandWalls
  - Wrapper, 45
- connect\_Callback
  - figure\_Laby.m, 49
- CreatePituresAndVideo
  - CreatePituresAndVideo.m, 47
- CreatePituresAndVideo.m, 47
  - CreatePituresAndVideo, 47
- CreatePituresAndVideo\_textured
  - CreatePituresAndVideo\_textured.m, 47
- CreatePituresAndVideo\_textured.m, 47
  - CreatePituresAndVideo\_textured, 47
- createUIEscape
  - figure\_Laby.m, 49
- createUIGhost
  - figure\_Laby.m, 51
- createUIPacman
  - figure\_Laby.m, 51
- createUIWalls
  - figure\_Laby.m, 51
- Down
  - ModelCommand, 9
- f
  - ModelCommand, 8
  - ModelGhost, 12, 14
  - ModelLaby, 18
  - ModelPacman, 24
  - ModelSED, 29
  - ModelWalls, 33
  - StopCondition, 38, 39
- figure\_Laby
  - figure\_Laby.m, 52
- figure\_Laby.m, 48
  - connect\_Callback, 49
  - createUIEscape, 49
  - createUIGhost, 51
  - createUIPacman, 51
  - createUIWalls, 51
  - figure\_Laby, 52
  - figure\_Laby\_OpeningFcn, 52
  - figure\_Laby\_OutputFcn, 53
  - isOne, 53
  - resetUIConnection, 54
  - ui\_Callback, 54
  - updatePresenceDetectorDisplay, 55
  - updateUIActiveCammand, 56
  - updateUIButton, 56
  - updateUICaught, 56
  - updateUIEscape, 57
  - updateUIPlayer, 58
  - updateUIWalls, 59
  - updateUIWallsAround, 59
  - updateUI, 55
- figure\_Laby\_OpeningFcn
  - figure\_Laby.m, 52
- figure\_Laby\_OutputFcn
  - figure\_Laby.m, 53
- g
  - ModelCommand, 8
  - ModelGhost, 14
  - ModelLaby, 18
  - ModelPacman, 26
  - ModelSED, 29
  - ModelWalls, 33
  - StopCondition, 39
- get\_out
  - Wrapper, 43
- get\_stop
  - Wrapper, 43
- ghostBit
  - Wrapper, 45
- handle, 7
- i
  - ModelPacman, 27
  - ModelWalls, 35
- in
  - Wrapper, 45
- init
  - Wrapper, 44
- initialState
  - ModelGhost, 15
  - ModelLaby, 21
  - ModelPacman, 27
  - ModelSED, 30
  - ModelWalls, 35
  - StopCondition, 40
- isOne
  - figure\_Laby.m, 53
- knowCompart
  - ModelCommand, 9
- Left
  - ModelCommand, 9

- m
  - ModelCommand, 9
  - ModelGhost, 15
  - ModelLaby, 19
  - ModelPacman, 26
  - ModelSED, 30
  - ModelWalls, 35
  - StopCondition, 40
- main.m, 59
- matrixAllPossible.m, 60
- memory
  - ModelPacman, 27
- ModelCommand, 8
  - Down, 9
  - f, 8
  - g, 8
  - knowCompart, 9
  - Left, 9
  - m, 9
  - presentState, 9
  - Right, 9
  - sizeTab, 9
  - Up, 10
- ModelCommand.m, 60
- ModelGhost, 10
  - f, 12, 14
  - g, 14
  - initialState, 15
  - m, 15
  - ModelGhost, 12
  - presentState, 15
- ModelGhost.m, 60
- ModelLaby, 16
  - f, 18
  - g, 18
  - initialState, 21
  - m, 19
  - ModelLaby, 17
  - presentState, 21
  - sameX\_position, 19
  - sameY\_position, 20
  - wallsHBetween, 20
  - wallsHBetweenOne, 20
  - wallsVBetween, 21
  - wallsVBetweenOne, 21
- modelLaby
  - Wrapper, 45
- ModelLaby.m, 60
- ModelPacman, 22
  - f, 24
  - g, 26
  - i, 27
  - initialState, 27
  - m, 26
  - memory, 27
  - ModelPacman, 24
  - presentState, 27
- ModelPacman.m, 60
- ModelSED.m, 60
- ModelSED, 28
  - f, 29
  - g, 29
  - initialState, 30
  - m, 30
  - presentState, 30
- ModelWalls, 31
  - f, 33
  - g, 33
  - i, 35
  - initialState, 35
  - m, 35
  - ModelWalls, 32
  - presentState, 35
  - val, 36
- ModelWalls.m, 60
- orderer
  - Wrapper, 44
- out
  - Wrapper, 46
- pacmanBit
  - Wrapper, 46
- presentState
  - ModelCommand, 9
  - ModelGhost, 15
  - ModelLaby, 21
  - ModelPacman, 27
  - ModelSED, 30
  - ModelWalls, 35
  - StopCondition, 40
- resetUIConnection
  - figure\_Laby.m, 54
- Right
  - ModelCommand, 9
- sameX\_position
  - ModelLaby, 19
- sameY\_position
  - ModelLaby, 20
- setColor
  - setColor.m, 61
- setColor.m, 61
  - setColor, 61
- Simulation.m, 61
- Simulation2\_allpossiblewalls.m, 61
- sizeTab
  - ModelCommand, 9
- stop
  - Wrapper, 46
- StopCondition, 36
  - f, 38, 39
  - g, 39
  - initialState, 40
  - m, 40
  - presentState, 40

- StopCondition, 38
- stopCondition
  - Wrapper, 46
- StopCondition.m, 61
- ui\_Callback
  - figure\_Laby.m, 54
- Up
  - ModelCommand, 10
- updateConnexion
  - Wrapper, 44
- updatePresenceDetectorDisplay
  - figure\_Laby.m, 55
- updateUIActiveCammand
  - figure\_Laby.m, 56
- updateUIButton
  - figure\_Laby.m, 56
- updateUICaught
  - figure\_Laby.m, 56
- updateUIEscape
  - figure\_Laby.m, 57
- updateUIPlayer
  - figure\_Laby.m, 58
- updateUIWalls
  - figure\_Laby.m, 59
- updateUIWallsAround
  - figure\_Laby.m, 59
- updateUI
  - figure\_Laby.m, 55
- val
  - ModelWalls, 36
- visupacman.m, 61
- visupacman2.m, 61
- wallsBit
  - Wrapper, 46
- wallsBorder
  - wallsBorder.m, 62
- wallsBorder.m, 61
  - wallsBorder, 62
- wallsHBetween
  - ModelLaby, 20
- wallsHBetweenOne
  - ModelLaby, 20
- wallsVBetween
  - ModelLaby, 21
- wallsVBetweenOne
  - ModelLaby, 21
- whoPlay
  - Wrapper, 46
- Wrapper, 41
  - commandPacman, 45
  - commandWalls, 45
  - get\_out, 43
  - get\_stop, 43
  - ghostBit, 45
  - in, 45
  - init, 44
  - modelLaby, 45
  - orderer, 44
  - out, 46
  - pacmanBit, 46
  - stop, 46
  - stopCondition, 46
  - updateConnexion, 44
  - wallsBit, 46
  - whoPlay, 46
  - Wrapper, 42
- Wrapper.m, 63