Fourmi de Langton

Generated by Doxygen 1.9.3

1 Fourmi de Langton

Ceci est la documentation d'un projet de Prep'Isima 2: La fourmi de Langton.

Elle regroupe les packages et les classes correspondant à l'implémentation de fourmis de Langton.

L'implémentation étant basée sur le paradigme classes/objets, vous retrouverez principalement des classes contenant leurs méthodes ainsi que les diagrammes de classes et les diagrammes call&caller de chaque méthode.

1.1 Version

Dernière version stable : 1.0

Dernière version : 1.0

1.2 Auteurs

Responsable de projet:

- BARRA Vincent

Equipe de développement:

- MALLEPEYRE Nourrane
- DUREL Enzo

1.3 Licence

Le projet est open-source

2 Namespace Index

2.1 Packages

Here are the packages with brief descriptions (if available):

3 Hierarchical Index

3.1 Class Hierarchy

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

4 Class Index

4.1 Class List

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

5 File Index

5.1 File List

Here is a list of all files with brief descriptions:

6 Namespace Documentation

6.1 buttons Namespace Reference

Package corresponding to all the functions specific to buttons.

Namespaces

- namespace button
  Button package.
- namespace check_box
  CheckBox package.
- namespace input_box
  InputBox package.
- namespace menu
  Menu package.

6.1.1 Detailed Description

Package corresponding to all the functions specific to buttons.

Author

Durel Enzo
Mallepeyre Nourrane

Version

1.0

6.2 buttons.button Namespace Reference

Button package.

Classes

- class Button
  Represent a Button.

6.2.1 Detailed Description

Button package.

Author

Durel Enzo
Mallepeyre Nourrane

Version

1.0

6.3 buttons.check_box Namespace Reference

CheckBox package.

Classes

- class CheckBox
  Represent a CheckBox.

6.3.1 Detailed Description

CheckBox package.

Author

Durel Enzo
Mallepeyre Nourrane

Version

1.0

6.4 buttons.input_box Namespace Reference

InputBox package.

Classes

- class InputBox
  Represent an InputBox.

6.4.1 Detailed Description

InputBox package.

Author

Durel Enzo
Mallepeyre Nourrane

Version

1.0

6.5 buttons.menu Namespace Reference

Menu package.

Classes

- class Menu

  Represent a Menu.

6.5.1 Detailed Description

Menu package.

Author

Durel Enzo

Mallepeyre Nourrane

Version

1.0

6.6 langton Namespace Reference

Package corresponding to all the functions specific to Langton.

Namespaces

- namespace case

  langton.case package
- namespace fourmi

  langton.fourmi package
- namespace plateau

  langton.plateau
- namespace simulation

  Simulation package.

6.6.1 Detailed Description

Package corresponding to all the functions specific to Langton.

Author

Durel Enzo

Mallepeyre Nourrane

Version

1.0

6.7 langton.case Namespace Reference

langton.case package

Classes

- class Case

  Represent a Case.

6.7.1 Detailed Description

langton.case package

Author

Durel Enzo

Mallepeyre Nourrane

Version

1.0

6.8 langton.fourmi Namespace Reference

langton.fourmi package

Classes

- class Fourmi

  Represent a Fourmi de Langton.

6.8.1 Detailed Description

langton.fourmi package

Author

Durel Enzo

Mallepeyre Nourrane

Version

1.0

6.9 langton.plateau Namespace Reference

langton.plateau

Classes

- class Plateau

  Represent a Plateau.

6.9.1 Detailed Description

langton.plateau

Author

Durel Enzo

Mallepeyre Nourrane

Version

1.0

6.10 langton.simulation Namespace Reference

Simulation package.

Classes

- class Simulation

  Represent a Simulation.

6.10.1 Detailed Description

Simulation package.

Author

Durel Enzo

Mallepeyre Nourrane

Version

1.0

6.11 main Namespace Reference

first program to be execute

Variables

- simulation = Simulation(res=4)

6.11.1 Detailed Description

first program to be execute

Author

Durel Enzo
Mallepeyre Nourrane

Version

1.0

6.11.2 Variable Documentation

6.11.2.1 simulation `main.simulation = Simulation(res=4)`

Definition at line 16 of file main.py.

6.12 utils Namespace Reference

Package corresponding to all the const need for the program.

Namespaces

- namespace color
  Package of all colors used in the program.
- namespace const
  Package of all constants used in the program.

6.12.1 Detailed Description

Package corresponding to all the const need for the program.

Author

Durel Enzo

Mallepeyre Nourrane

Version

1.0

6.13 utils.color Namespace Reference

Package of all colors used in the program.

Variables

- dic = dict()
- tuple INACTIVE_BUTTON_COLOR = (46, 107, 81)
- tuple ACTIVE_BUTTON_COLOR = (69, 153, 125)
- tuple HOVER_BUTTON_COLOR = (49, 122, 110)
- tuple DISABLE_BUTTON_COLOR = (55, 64, 60)
- TEXT_BUTTON_COLOR = dic["white"]
- tuple INACTIVE_IB_COLOR = (46, 107, 81)
- tuple ACTIVE_IB_COLOR = (186, 191, 119)
- tuple DISABLE_IB_COLOR = (55, 64, 60)
- TEXT_IB_COLOR = dic["black"]
- tuple INACTIVE_CB_COLOR = (46, 107, 81)
- tuple ACTIVE_CB_COLOR = (186, 191, 119)
- tuple DISABLE_CB_COLOR = (55, 64, 60)
- tuple DISABLE_ACTIVE_CB_COLOR = (87, 56, 53)
- TEXT_CB_COLOR = dic["black"]
- tuple MENU_COLOR = (103, 168, 120)

6.13.1 Detailed Description

Package of all colors used in the program.

Author

Durel Enzo

Mallepeyre Nourrane

Version

1.0

6.13.2 Variable Documentation

6.13.2.1 ACTIVE_BUTTON_COLOR `tuple utils.color.ACTIVE_BUTTON_COLOR = (69, 153, 125)`

Definition at line 17 of file color.py.

6.13.2.2 ACTIVE_CB_COLOR `tuple utils.color.ACTIVE_CB_COLOR = (186, 191, 119)`

Definition at line 32 of file color.py.

6.13.2.3 ACTIVE_IB_COLOR `tuple utils.color.ACTIVE_IB_COLOR = (186, 191, 119)`

Definition at line 25 of file color.py.

6.13.2.4 dic `utils.color.dic = dict()`

Definition at line 9 of file color.py.

6.13.2.5 DISABLE_ACTIVE_CB_COLOR `tuple utils.color.DISABLE_ACTIVE_CB_COLOR = (87, 56, 53)`

Definition at line 34 of file color.py.

6.13.2.6 DISABLE_BUTTON_COLOR `tuple utils.color.DISABLE_BUTTON_COLOR = (55, 64, 60)`

Definition at line 19 of file color.py.

6.13.2.7 DISABLE_CB_COLOR `tuple utils.color.DISABLE_CB_COLOR = (55, 64, 60)`

Definition at line 33 of file color.py.

6.13.2.8 DISABLE_IB_COLOR

```
tuple utils.color.DISABLE_IB_COLOR = (55, 64, 60)
```

Definition at line 26 of file color.py.

6.13.2.9 HOVER_BUTTON_COLOR

```
tuple utils.color.HOVER_BUTTON_COLOR = (49, 122, 110)
```

Definition at line 18 of file color.py.

6.13.2.10 INACTIVE_BUTTON_COLOR

```
tuple utils.color.INACTIVE_BUTTON_COLOR = (46, 107, 81)
```

Definition at line 16 of file color.py.

6.13.2.11 INACTIVE_CB_COLOR

```
tuple utils.color.INACTIVE_CB_COLOR = (46, 107, 81)
```

Definition at line 31 of file color.py.

6.13.2.12 INACTIVE_IB_COLOR

```
tuple utils.color.INACTIVE_IB_COLOR = (46, 107, 81)
```

Definition at line 24 of file color.py.

6.13.2.13 MENU_COLOR

```
tuple utils.color.MENU_COLOR = (103, 168, 120)
```

Definition at line 39 of file color.py.

6.13.2.14 TEXT_BUTTON_COLOR

```
utils.color.TEXT_BUTTON_COLOR = dic["white"]
```

Definition at line 21 of file color.py.

6.13.2.15 TEXT_CB_COLOR

```
utils.color.TEXT_CB_COLOR = dic["black"]
```

Definition at line 36 of file color.py.

6.13.2.16 TEXT_IB_COLOR `utils.color.TEXT_IB_COLOR = dic["black"]`

Definition at line 28 of file color.py.

6.14 utils.const Namespace Reference

Package of all constants used in the program.

Variables

- tuple DEFAULT_SCREEN_SIZE = (1280, 720)
- tuple DEFAULT_PLATEAU_SIZE = (1000, 720)
- int DEFAULT_RESOLUTION = 4
- tuple BUTTON_SIZE = (100, 50)

6.14.1 Detailed Description

Package of all constants used in the program.

Author

Durel Enzo

Mallepeyre Nourrane

Version

1.0

6.14.2 Variable Documentation

6.14.2.1 BUTTON_SIZE `tuple utils.const.BUTTON_SIZE = (100, 50)`

Definition at line 14 of file const.py.

6.14.2.2 DEFAULT_PLATEAU_SIZE `tuple utils.const.DEFAULT_PLATEAU_SIZE = (1000, 720)`

Definition at line 10 of file const.py.

6.14.2.3 DEFAULT_RESOLUTION `int utils.const.DEFAULT_RESOLUTION = 4`

Definition at line 11 of file const.py.

6.14.2.4 DEFAULT_SCREEN_SIZE `tuple utils.const.DEFAULT_SCREEN_SIZE = (1280, 720)`

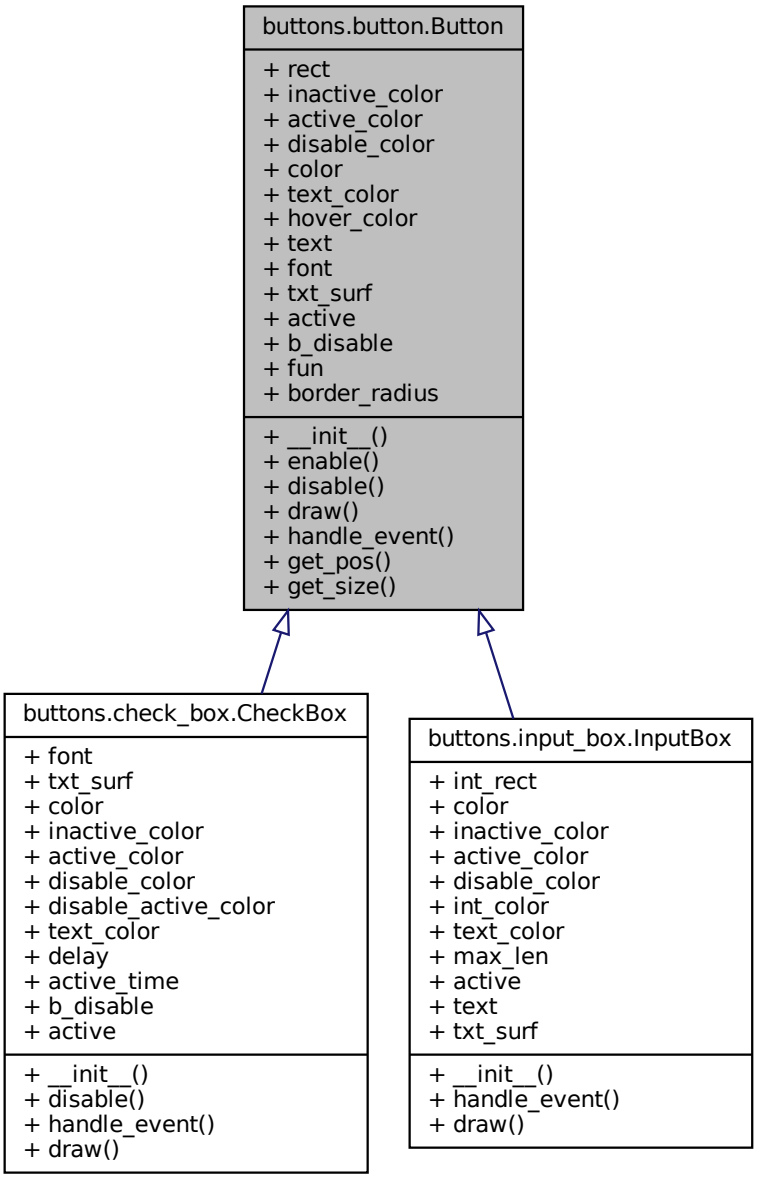
Definition at line 9 of file const.py.

7 Class Documentation

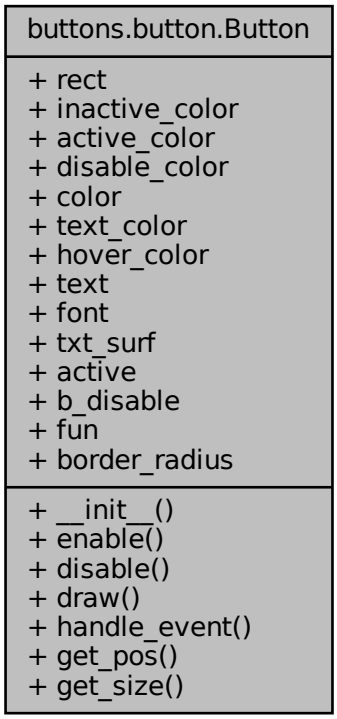
7.1 buttons.button.Button Class Reference

Represent a Button.

Inheritance diagram for buttons.button.Button:



Collaboration diagram for buttons.button.Button:



Public Member Functions

- def __init__ (self, pos, size, text='', fun=None)

  Construct Button object.
- def enable (self)

  Enable the button This method enable the button (the user can click on it).
- def disable (self)

  Disable the button This method disable the button (the user can't click on it).
- def draw (self, screen)

  Draw the button This method draw the button rectangle with pyGame draw.rect function and the text with screen.blit function.
- def handle_event (self, event)

  User input method This method operate users input with event.type pyGame attributs.
- def get_pos (self)

  Get the button top left position.
- def get_size (self)

  Get the button size.

Public Attributes

- rect
- inactive_color
- active_color
- disable_color
- color
- text_color
- hover_color
- text
- font
- txt_surf
- active
- b_disable
- fun
- border_radius

7.1.1 Detailed Description

Represent a Button.

Definition at line 11 of file button.py.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 __init__()

```
def buttons.button.Button.__init__ (
        self,
        pos,
        size,
        text = '',
        fun = None )
```

Construct Button object.

Parameters

pos	A tuple position of top left button corner
size	A tuple represent the size of button (width, height)
text	String affiliate to the button
fun	Function reference for button event

Reimplemented in buttons.check_box.CheckBox, and buttons.input_box.InputBox.

Definition at line 14 of file button.py.

7.1.3 Member Function Documentation

7.1.3.1 disable() def buttons.button.Button.disable (self)

Disable the button This method disable the button (the user can't click on it).

Reimplemented in buttons.check_box.CheckBox.

Definition at line 51 of file button.py.

7.1.3.2 draw() def buttons.button.Button.draw (self, screen)

Draw the button This method draw the button rectangle with pyGame draw.rect function and the text with screen.blit function.

Parameters

screen	Pygame screen object where the button with be draw

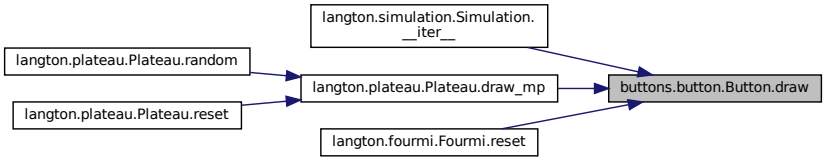
Exceptions

Exception	Used if pyGame is under update

Reimplemented in buttons.check_box.CheckBox, and buttons.input_box.InputBox.

Definition at line 59 of file button.py.

Here is the caller graph for this function:



7.1.3.3 enable() def buttons.button.Button.enable (self)

Enable the button This method enable the button (the user can click on it).

Definition at line 43 of file button.py.

7.1.3.4 get_pos() def buttons.button.Button.get_pos (self)

Get the button top left position.

Returns

A tuple of the position (x, y)

Definition at line 113 of file button.py.

7.1.3.5 get_size() def buttons.button.Button.get_size (self)

Get the button size.

Returns

A tuple of the size (w, h)

Definition at line 119 of file button.py.

7.1.3.6 handle_event() def buttons.button.Button.handle_event (self, event)

User input method This method operate users input with event.type pyGame attributs.

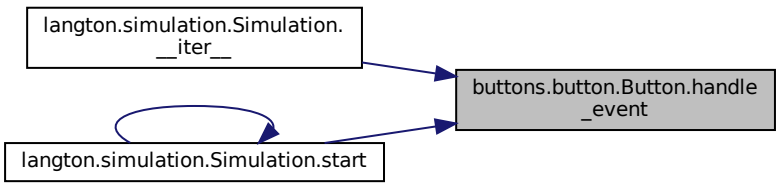
Parameters

event	Event user input

Reimplemented in buttons.check_box.CheckBox, and buttons.input_box.InputBox.

Definition at line 81 of file button.py.

Here is the caller graph for this function:



7.1.4 Member Data Documentation

7.1.4.1 active `buttons.button.Button.active`

Definition at line 36 of file button.py.

7.1.4.2 active_color `buttons.button.Button.active_color`

Definition at line 25 of file button.py.

7.1.4.3 b_disable `buttons.button.Button.b_disable`

Definition at line 37 of file button.py.

7.1.4.4 border_radius `buttons.button.Button.border_radius`

Definition at line 41 of file button.py.

7.1.4.5 color `buttons.button.Button.color`

Definition at line 28 of file button.py.

7.1.4.6 disable_color `buttons.button.Button.disable_color`

Definition at line 26 of file button.py.

7.1.4.7 font `buttons.button.Button.font`

Definition at line 33 of file button.py.

7.1.4.8 fun buttons.button.Button.fun

Definition at line 39 of file button.py.

7.1.4.9 hover_color buttons.button.Button.hover_color

Definition at line 30 of file button.py.

7.1.4.10 inactive_color buttons.button.Button.inactive_color

Definition at line 24 of file button.py.

7.1.4.11 rect buttons.button.Button.rect

Definition at line 22 of file button.py.

7.1.4.12 text buttons.button.Button.text

Definition at line 32 of file button.py.

7.1.4.13 text_color buttons.button.Button.text_color

Definition at line 29 of file button.py.

7.1.4.14 txt_surf buttons.button.Button.txt_surf

Definition at line 34 of file button.py.

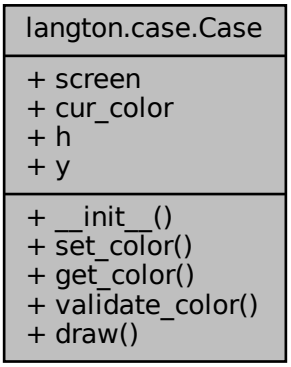
The documentation for this class was generated from the following file:

- button.py

7.2 langton.case.Case Class Reference

Represent a Case.

Collaboration diagram for langton.case.Case:



Public Member Functions

- def __init__ (self, size=(1, 1), pos=(0, 0))

  Construct Case object.
- def set_color (self, colour)

  Set a color the the Case.
- def get_color (self)

  get the Case color
- def validate_color (self, colour)

  Verify if it's a valid colour.
- def draw (self)

  draw the Case

Public Attributes

- screen
- cur_color
- h
- y

7.2.1 Detailed Description

Represent a Case.

Definition at line 11 of file case.py.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 __init__() `def langton.case.Case.__init__ ( self, size = (1, 1), pos = (0, 0) )`

Construct Case object.

Parameters

size	Pixel size of the case (pygame)
pos	Position of the case

Definition at line 14 of file case.py.

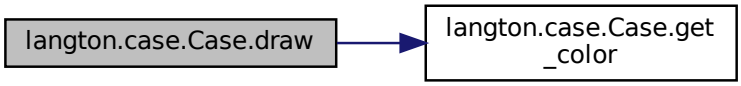
7.2.3 Member Function Documentation

7.2.3.1 draw() `def langton.case.Case.draw ( self )`

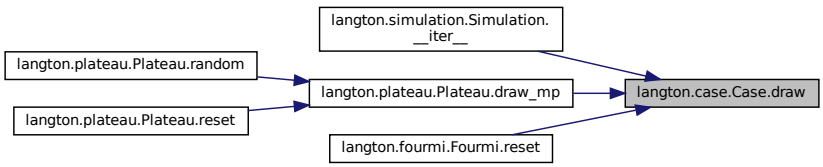
draw the Case

Definition at line 50 of file case.py.

Here is the call graph for this function:



Here is the caller graph for this function:



7.2.3.2 get_color() def langton.case.Case.get_color (self)

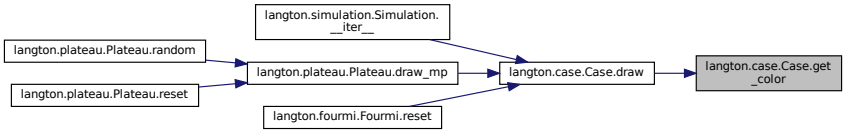
get the Case color

Returns

the Case color

Definition at line 30 of file case.py.

Here is the caller graph for this function:



7.2.3.3 set_color() def langton.case.Case.set_color (self, colour)

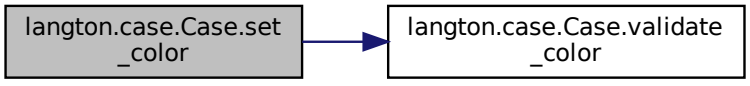
Set a color the the Case.

Parameters

colour	A tuple of int representing a rgb color

Definition at line 24 of file case.py.

Here is the call graph for this function:



7.2.3.4 validate_color() def langton.case.Case.validate_color (self, colour)

Verify if it's a valid colour.

Parameters

colour	A tuple of int representing a rgb colour

Returns

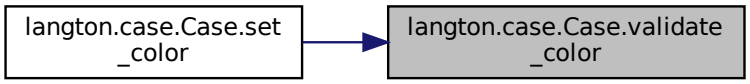
The valide colour

Exceptions

Exception	Not a valid colour

Definition at line 36 of file case.py.

Here is the caller graph for this function:



7.2.4 Member Data Documentation

7.2.4.1 cur_color `langton.case.Case.cur_color`

Definition at line 20 of file case.py.

7.2.4.2 h `langton.case.Case.h`

Definition at line 21 of file case.py.

7.2.4.3 screen `langton.case.Case.screen`

Definition at line 19 of file case.py.

7.2.4.4 y `langton.case.Case.y`

Definition at line 22 of file case.py.

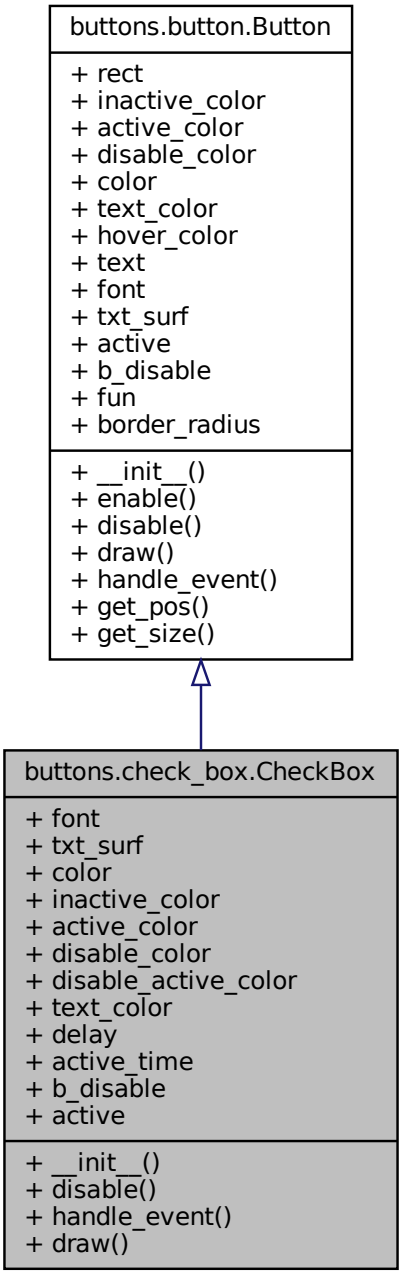
The documentation for this class was generated from the following file:

- case.py

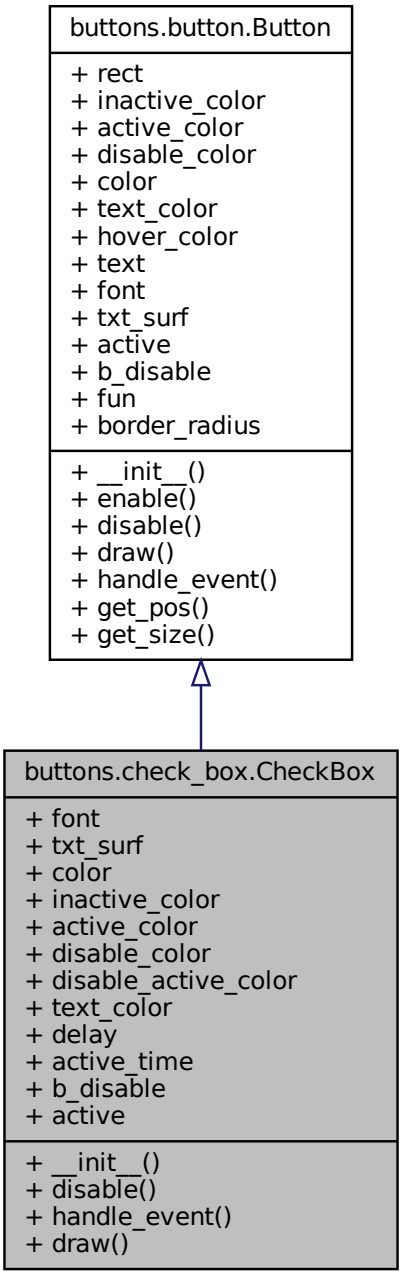
7.3 buttons.check_box.CheckBox Class Reference

Represent a CheckBox.

Inheritance diagram for buttons.check_box.CheckBox:



Collaboration diagram for buttons.check_box.CheckBox:



Public Member Functions

- def __init__ (self, pos, size, text='', fun=None)

  Construct Button object.
- def disable (self)

  Disable the button This method disable the button (the user can't click on it).
- def handle_event (self, event)

User input method This method operate users input with event.type pyGame attributs.

- def draw (self, screen)

  Draw the button This method draw the button rectangle with pyGame draw.rect function and the text with screen.blit function.

Public Attributes

- font
- txt_surf
- color
- inactive_color
- active_color
- disable_color
- disable_active_color
- text_color
- delay
- active_time
- b_disable
- active

7.3.1 Detailed Description

Represent a CheckBox.

Definition at line 13 of file check_box.py.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 __init__() `def buttons.check_box.CheckBox.__init__ (`

```
self,
pos,
size,
text = '',
fun = None )
```

Construct Button object.

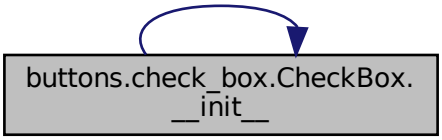
Parameters

pos	A tuple position of top left button corner
size	A tuple represent the size of button (width, height)
text	String affiliate to the button
fun	Function reference for button event

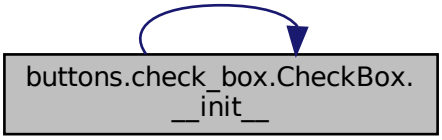
Reimplemented from buttons.button.Button.

Definition at line 16 of file check_box.py.

Here is the call graph for this function:



Here is the caller graph for this function:



7.3.3 Member Function Documentation

7.3.3.1 disable() `def buttons.check_box.CheckBox.disable ( self )`

Disable the button This method disable the button (the user can't click on it).

Reimplemented from buttons.button.Button.

Definition at line 34 of file check_box.py.

7.3.3.2 draw() `def buttons.check_box.CheckBox.draw ( self, screen )`

Draw the button This method draw the button rectangle with pyGame draw.rect function and the text with screen.blit function.

Parameters

screen	Pygame screen object where the button with be draw

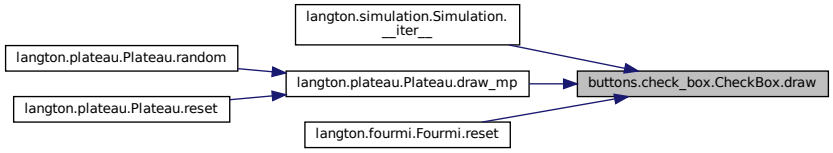
Exceptions

Exception	Used if pyGame is under update

Reimplemented from buttons.button.Button.

Definition at line 56 of file check_box.py.

Here is the caller graph for this function:



7.3.3.3 handle_event() `def buttons.check_box.CheckBox.handle_event ( self, event )`

User input method This method operate users input with event.type pyGame attributs.

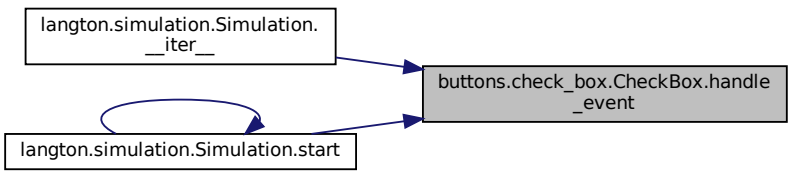
Parameters

event	Event user input

Reimplemented from buttons.button.Button.

Definition at line 42 of file check_box.py.

Here is the caller graph for this function:



7.3.4 Member Data Documentation

7.3.4.1 active `buttons.check_box.CheckBox.active`

Definition at line 48 of file check_box.py.

7.3.4.2 active_color `buttons.check_box.CheckBox.active_color`

Definition at line 26 of file check_box.py.

7.3.4.3 active_time `buttons.check_box.CheckBox.active_time`

Definition at line 32 of file check_box.py.

7.3.4.4 b_disable `buttons.check_box.CheckBox.b_disable`

Definition at line 36 of file check_box.py.

7.3.4.5 color `buttons.check_box.CheckBox.color`

Definition at line 24 of file check_box.py.

7.3.4.6 delay `buttons.check_box.CheckBox.delay`

Definition at line 31 of file check_box.py.

7.3.4.7 disable_active_color `buttons.check_box.CheckBox.disable_active_color`

Definition at line 28 of file check_box.py.

7.3.4.8 disable_color `buttons.check_box.CheckBox.disable_color`

Definition at line 27 of file check_box.py.

7.3.4.9 font `buttons.check_box.CheckBox.font`

Definition at line 21 of file check_box.py.

7.3.4.10 inactive_color `buttons.check_box.CheckBox.inactive_color`

Definition at line 25 of file check_box.py.

7.3.4.11 text_color `buttons.check_box.CheckBox.text_color`

Definition at line 29 of file check_box.py.

7.3.4.12 txt_surf `buttons.check_box.CheckBox.txt_surf`

Definition at line 22 of file check_box.py.

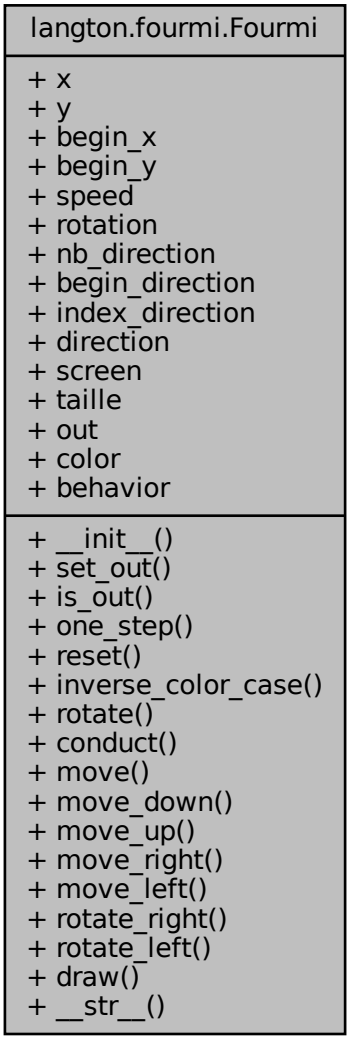
The documentation for this class was generated from the following file:

- check_box.py

7.4 langton.fourmi.Fourmi Class Reference

Represent a Fourmi de Langton.

Collaboration diagram for langton.fourmi.Fourmi:



Public Member Functions

- def __init__ (self, coords=(0, 0), taille=4, speed=1, direction=0, color=[(255, 255, 255),(0, 0, 0)], behavior="LR")

  Constuct Fourmi object This is the constructor of the Fourmi object.
- def set_out (self)

  Ant is out This method makes the ant out.
- def is_out (self)

  Ask if fourmi is out This method return the out state of the ant.
- def one_step (self, case)

  An ant complete movement This method make the ant follows a complete movement (rotate, change color, move).
- def reset (self)

  Reset the Ant This method hard reset the ant at its beginning direction, position.

- def inverse_color_case (self, case)

  Inverse Case color This method change the color of the case where the ant is.
- def rotate (self, case)

  Rotation the ant This method rotate the ant following the ant's behavior.
- def conduct (self)

  Move the ant following its conduct This method moves the ant compare to the conduct wanted.
- def move (self, coords=(0, 0))

  Vectorial ant movement This method reprensent primitive ant movement.
- def move_down (self)

  Ant move down This method calls move() with a down vector (0, y).
- def move_up (self)

  Ant move up This method calls move() with a up vector (0, -y).
- def move_right (self)

  Ant move right This method calls move() with a right vector (x, 0).
- def move_left (self)

  Ant move left This method calls move() with a left vector (-x, 0).
- def rotate_right (self)

  Ant rotate left This method rotate the ant in its right.
- def rotate_left (self)

  Ant rotate left This method rotate the ant in its left.
- def draw (self)

  Ant draw This method draw the ant with pyGame draw.rect function.
- def __str__ (self)

  Ant string representation This method redefine the ant's toString() representation.

Public Attributes

- x
- y
- begin_x
- begin_y
- speed
- rotation
- nb_direction
- begin_direction
- index_direction
- direction
- screen
- taille
- out
- color
- behavior

7.4.1 Detailed Description

Represent a Fourmi de Langton.

Definition at line 11 of file fourmi.py.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 __init__() def langton.fourmi.Fourmi.__init__ (

```
self,
coords = (0, 0),
taille = 4,
speed = 1,
direction = 0,
color = [(255, 255, 255), (0, 0, 0)],
behavior = "LR" )
```

Constuct Fourmi object This is the constructor of the Fourmi object.

Parameters

coords	coordinate where ant takes place (default (0, 0))
taille	number of pixels represent an ant (default 4)
speed	number of case ant moving (default 1)
direction	index of first direction (default 0 ("up"))
color	list of tuple represent the list of color used for behavior (default: (255, ...), (0, ...))
behavior	string representation of the ant behavior (default: 'LR')

Definition at line 14 of file fourmi.py.

7.4.3 Member Function Documentation

7.4.3.1 __str__() def langton.fourmi.Fourmi.__str__ (

```
self )
```

Ant string representation This method redefine the ant's toString() representation.

Definition at line 175 of file fourmi.py.

7.4.3.2 conduct() def langton.fourmi.Fourmi.conduct (

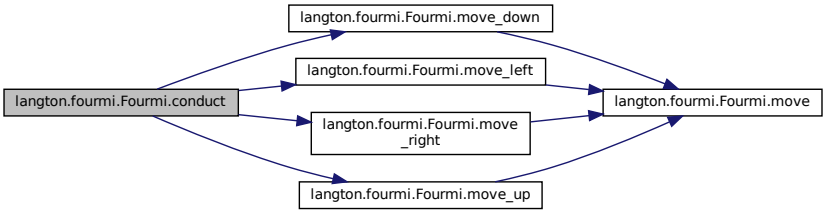
```
self )
```

Move the ant following its conduct This method moves the ant compare to the conduct wanted.

Here the ant move in the direction where it watches.

Definition at line 100 of file fourmi.py.

Here is the call graph for this function:



Here is the caller graph for this function:

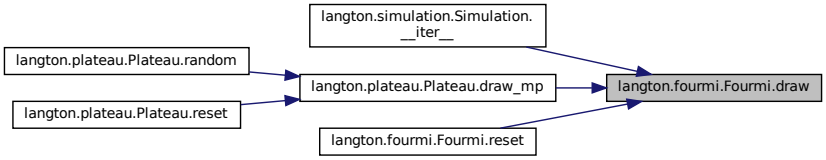
7.4.3.3 draw() def langton.fourmi.Fourmi.draw (self)

Ant draw This method draw the ant with pyGame draw.rect function.

Ant color is red.

Definition at line 165 of file fourmi.py.

Here is the caller graph for this function:



7.4.3.4 inverse_color_case() def langton.fourmi.Fourmi.inverse_color_case (self, case)

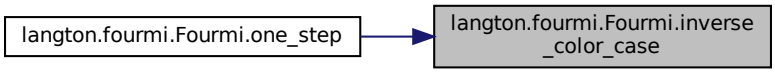
Inverse Case color This method change the color of the case where the ant is.

Parameters

case	Case where the ant is

Definition at line 80 of file fourmi.py.

Here is the caller graph for this function:



7.4.3.5 is_out() `def langton.fourmi.Fourmi.is_out ( self )`

Ask if fourmi is out This method return the out state of the ant.

Returns

boolean

Definition at line 53 of file fourmi.py.

7.4.3.6 move() `def langton.fourmi.Fourmi.move ( self, coords = (0, 0) )`

Vectorial ant movement This method reprensent primitive ant movement.

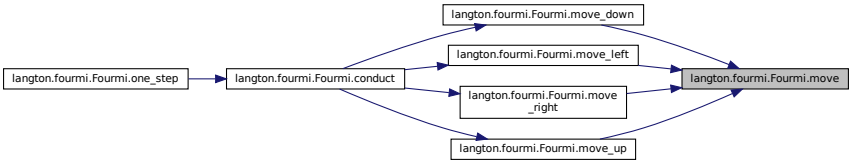
It's update the x and y of the ant.

Parameters

coords	A tuple represents a movement vector (default (0,0))

Definition at line 114 of file fourmi.py.

Here is the caller graph for this function:

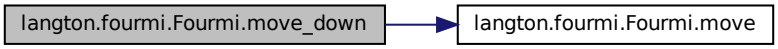


7.4.3.7 move_down() `def langton.fourmi.Fourmi.move_down ( self )`

Ant move down This method calls move() with a down vector (0, y).

Definition at line 123 of file fourmi.py.

Here is the call graph for this function:



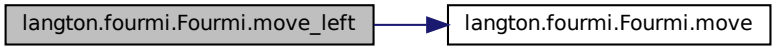
Here is the caller graph for this function:

7.4.3.8 move_left() `def langton.fourmi.Fourmi.move_left ( self )`

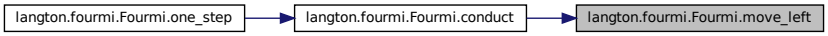
Ant move left This method calls move() with a left vector (-x, 0).

Definition at line 141 of file fourmi.py.

Here is the call graph for this function:



Here is the caller graph for this function:

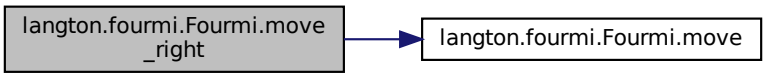


7.4.3.9 move_right() `def langton.fourmi.Fourmi.move_right ( self )`

Ant move right This method calls move() with a right vector (x, 0).

Definition at line 135 of file fourmi.py.

Here is the call graph for this function:



Here is the caller graph for this function:

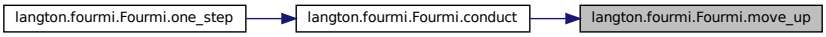
7.4.3.10 move_up() def langton.fourmi.Fourmi.move_up (self)

Ant move up This method calls move() with a up vector (0, -y).

Definition at line 129 of file fourmi.py.

Here is the call graph for this function:

Here is the caller graph for this function:



7.4.3.11 one_step() def langton.fourmi.Fourmi.one_step (self, case)

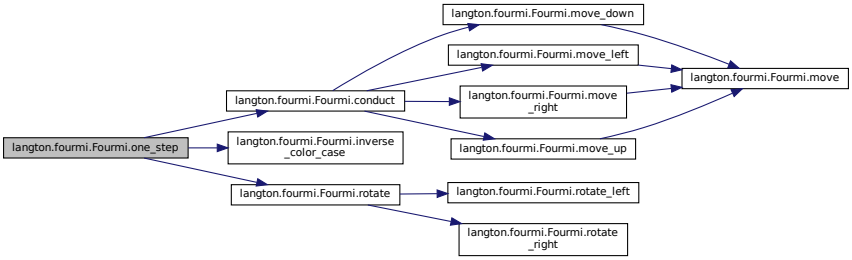
An ant complete movement This method make the ant follows a complete movement (rotate, change color, move).

Parameters

case	Case where the ant begin its step

Definition at line 60 of file fourmi.py.

Here is the call graph for this function:

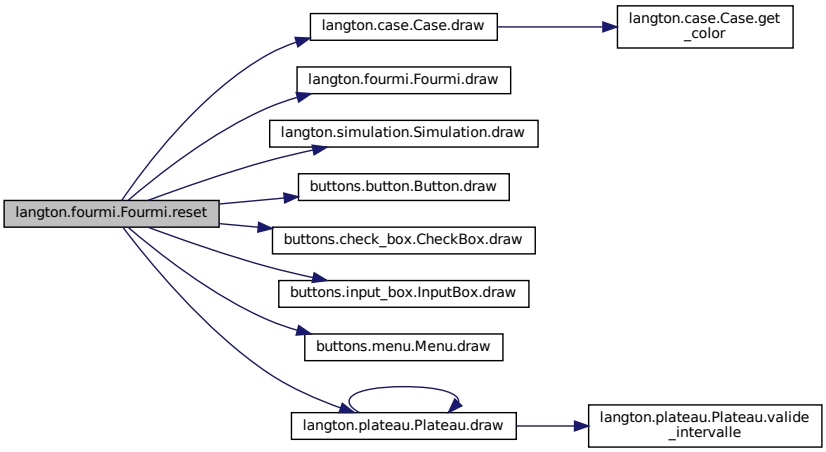


7.4.3.12 reset() `def langton.fourmi.Fourmi.reset ( self )`

Reset the Ant This method hard reset the ant at its beginning direction, position.

Definition at line 70 of file fourmi.py.

Here is the call graph for this function:



7.4.3.13 rotate() `def langton.fourmi.Fourmi.rotate ( self, case )`

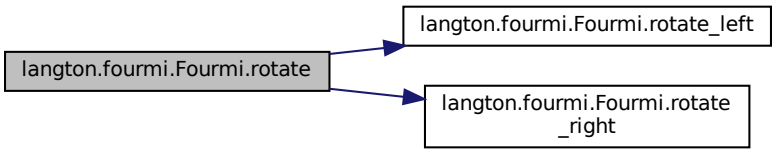
Rotation the ant This method rotate the ant following the ant's behavior.

Parameters

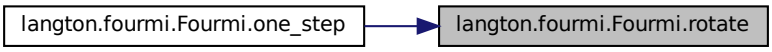
case	Case where the ant is.

Definition at line 88 of file fourmi.py.

Here is the call graph for this function:



Here is the caller graph for this function:



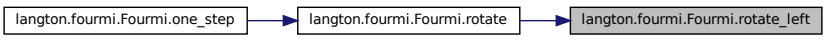
7.4.3.14 rotate_left() `def langton.fourmi.Fourmi.rotate_left ( self )`

Ant rotate left This method rotate the ant in its left.

It means the index of the current rotation is decrement by one in the list of rotation.

Definition at line 156 of file fourmi.py.

Here is the caller graph for this function:



7.4.3.15 rotate_right() `def langton.fourmi.Fourmi.rotate_right ( self )`

Ant rotate left This method rotate the ant in its right.

It means the index of the current rotation is increment by one in the list of rotation.

Definition at line 147 of file fourmi.py.

Here is the caller graph for this function:

7.4.3.16 set_out() `def langton.fourmi.Fourmi.set_out ( self )`

Ant is out This method makes the ant out.

She can't do anything anymore.

Definition at line 47 of file fourmi.py.

7.4.4 Member Data Documentation

7.4.4.1 begin_direction `langton.fourmi.Fourmi.begin_direction`

Definition at line 36 of file fourmi.py.

7.4.4.2 begin_x `langton.fourmi.Fourmi.begin_x`

Definition at line 30 of file fourmi.py.

7.4.4.3 begin_y `langton.fourmi.Fourmi.begin_y`

Definition at line 31 of file fourmi.py.

7.4.4.4 behavior `langton.fourmi.Fourmi.behavior`

Definition at line 45 of file fourmi.py.

7.4.4.5 color `langton.fourmi.Fourmi.color`

Definition at line 44 of file fourmi.py.

7.4.4.6 direction `langton.fourmi.Fourmi.direction`

Definition at line 38 of file fourmi.py.

7.4.4.7 index_direction `langton.fourmi.Fourmi.index_direction`

Definition at line 37 of file fourmi.py.

7.4.4.8 nb_direction `langton.fourmi.Fourmi.nb_direction`

Definition at line 35 of file fourmi.py.

7.4.4.9 out `langton.fourmi.Fourmi.out`

Definition at line 42 of file fourmi.py.

7.4.4.10 rotation `langton.fourmi.Fourmi.rotation`

Definition at line 34 of file fourmi.py.

7.4.4.11 screen `langton.fourmi.Fourmi.screen`

Definition at line 40 of file fourmi.py.

7.4.4.12 speed `langton.fourmi.Fourmi.speed`

Definition at line 33 of file fourmi.py.

7.4.4.13 taille `langton.fourmi.Fourmi.taille`

Definition at line 41 of file fourmi.py.

7.4.4.14 x `langton.fourmi.Fourmi.x`

Definition at line 28 of file fourmi.py.

7.4.4.15 y `langton.fourmi.Fourmi.y`

Definition at line 29 of file fourmi.py.

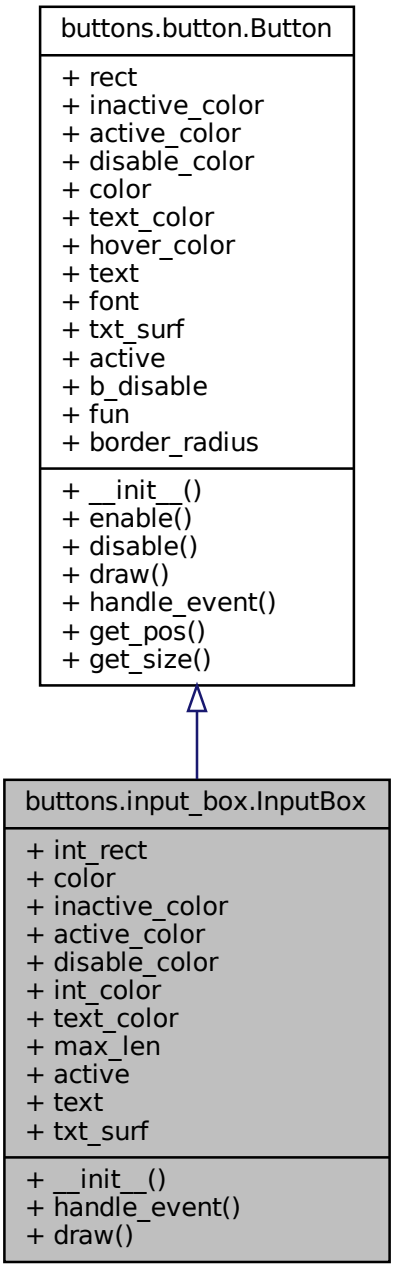
The documentation for this class was generated from the following file:

- fourmi.py

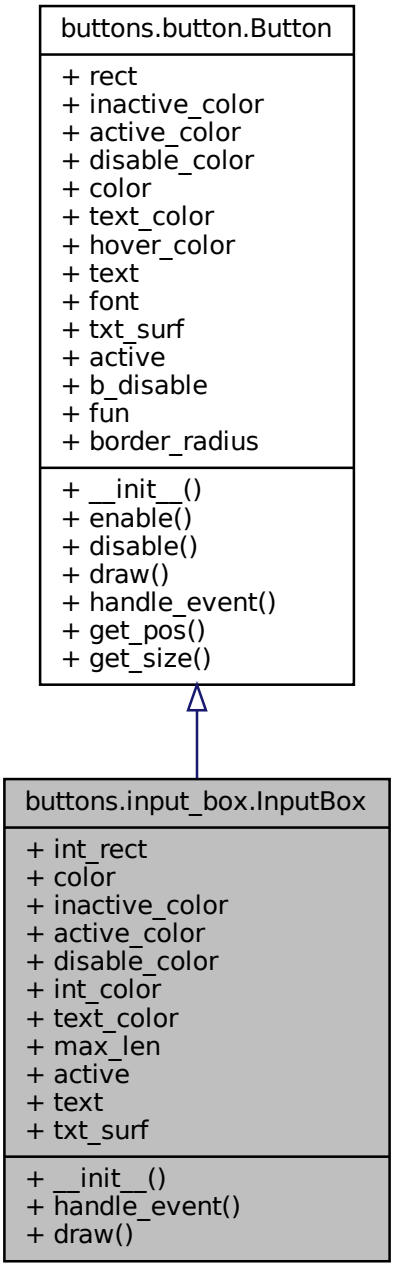
7.5 buttons.input_box.InputBox Class Reference

Represent an InputBox.

Inheritance diagram for buttons.input_box.InputBox:



Collaboration diagram for buttons.input_box.InputBox:



Public Member Functions

- def __init__ (self, pos, size, text='', fun=None, max_len=5)

  Construct Button object.
- def handle_event (self, event)

  User input method This method operate users input with event.type pyGame attributs.
- def draw (self, screen)

Draw the button This method draw the button rectangle with pyGame draw.rect function and the text with screen.blit function.

Public Attributes

- int_rect
- color
- inactive_color
- active_color
- disable_color
- int_color
- text_color
- max_len
- active
- text
- txt_surf

7.5.1 Detailed Description

Represent an InputBox.

Definition at line 13 of file input_box.py.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 __init__() `def buttons.input_box.InputBox.__init__ (`

```
self,
pos,
size,
text = '',
fun = None,
max_len = 5 )
```

Construct Button object.

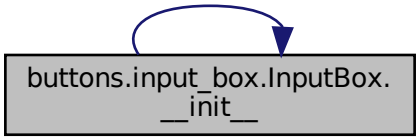
Parameters

pos	A tuple position of top left button corner
size	A tuple represent the size of button (width, height)
text	String affiliate to the button
fun	Function reference for button event

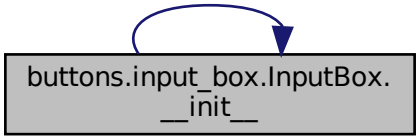
Reimplemented from buttons.button.Button.

Definition at line 16 of file input_box.py.

Here is the call graph for this function:



Here is the caller graph for this function:



7.5.3 Member Function Documentation

7.5.3.1 draw()

```
def buttons.input_box.InputBox.draw (
    self,
    screen )
```

Draw the button This method draw the button rectangle with pyGame draw.rect function and the text with screen.blit function.

Parameters

screen	Pygame screen object where the button with be draw

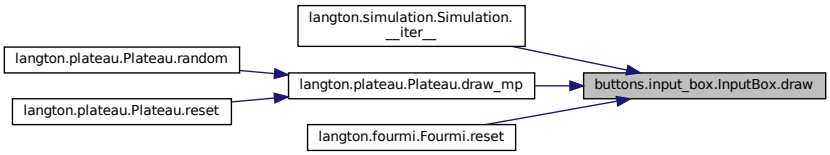
Exceptions

Exception	Used if pyGame is under update

Reimplemented from buttons.button.Button.

Definition at line 58 of file input_box.py.

Here is the caller graph for this function:



7.5.3.2 handle_event() `def buttons.input_box.InputBox.handle_event ( self, event )`

User input method This method operate users input with event.type pyGame attributs.

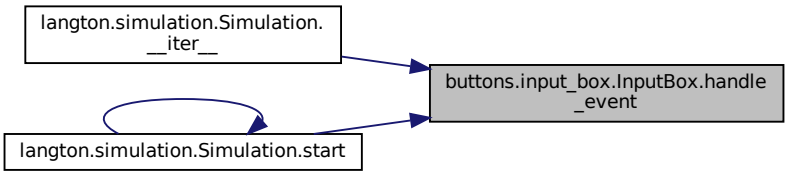
Parameters

event	Event user input

Reimplemented from buttons.button.Button.

Definition at line 30 of file input_box.py.

Here is the caller graph for this function:



7.5.4 Member Data Documentation

7.5.4.1 active `buttons.input_box.InputBox.active`

Definition at line 35 of file input_box.py.

7.5.4.2 active_color `buttons.input_box.InputBox.active_color`

Definition at line 23 of file input_box.py.

7.5.4.3 color `buttons.input_box.InputBox.color`

Definition at line 21 of file input_box.py.

7.5.4.4 disable_color `buttons.input_box.InputBox.disable_color`

Definition at line 24 of file input_box.py.

7.5.4.5 inactive_color `buttons.input_box.InputBox.inactive_color`

Definition at line 22 of file input_box.py.

7.5.4.6 int_color `buttons.input_box.InputBox.int_color`

Definition at line 25 of file input_box.py.

7.5.4.7 int_rect `buttons.input_box.InputBox.int_rect`

Definition at line 19 of file input_box.py.

7.5.4.8 max_len `buttons.input_box.InputBox.max_len`

Definition at line 28 of file input_box.py.

7.5.4.9 text `buttons.input_box.InputBox.text`

Definition at line 46 of file input_box.py.

7.5.4.10 text_color `buttons.input_box.InputBox.text_color`

Definition at line 26 of file input_box.py.

7.5.4.11 txt_surf `buttons.input_box.InputBox.txt_surf`

Definition at line 56 of file input_box.py.

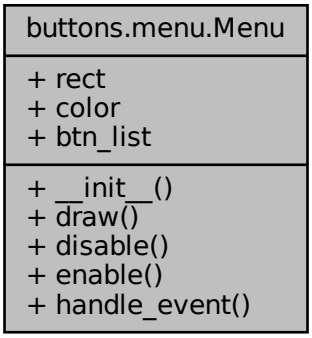
The documentation for this class was generated from the following file:

- input_box.py

7.6 buttons.menu.Menu Class Reference

Represent a Menu.

Collaboration diagram for buttons.menu.Menu:



Public Member Functions

- def __init__ (self, pos, size, btn_list=[ ], color=color.MENU_COLOR)

  Construct a Menu.
- def draw (self, screen)

  Draw the Menu.
- def disable (self, ∗args)

  Disable some button of Menu.
- def enable (self, ∗args)

  Enable some button of Menu.
- def handle_event (self, event, ∗args)

  Enable some button of Menu.

Public Attributes

- rect
- color
- btn_list

7.6.1 Detailed Description

Represent a Menu.

Definition at line 11 of file menu.py.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 __init__() `def buttons.menu.Menu.__init__ (`

```
self,
pos,
size,
btn_list = [],
color = color.MENU_COLOR )
```

Construct a Menu.

Parameters

pos	A tuple position of top left Menu corner
size	A tuple represent the size of Menu (width, height)
btn_list	A list of button in the Menu
color	The Color of the Menu

Definition at line 14 of file menu.py.

7.6.3 Member Function Documentation

7.6.3.1 disable() `def buttons.menu.Menu.disable (`

```
self,
* args )
```

Disable some button of Menu.

Parameters

args	variadic args which need to be disable

Definition at line 33 of file menu.py.

7.6.3.2 draw()

```
def buttons.menu.Menu.draw (
        self,
        screen )
```

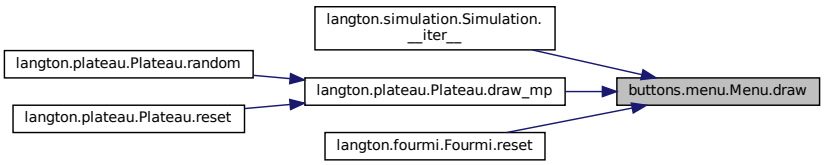
Draw the Menu.

Parameters

screen	Screen pyGame attributs

Definition at line 25 of file menu.py.

Here is the caller graph for this function:



7.6.3.3 enable()

```
def buttons.menu.Menu.enable (
        self,
      * args )
```

Enable some button of Menu.

Parameters

args	variadic args which need to be enable

Definition at line 42 of file menu.py.

7.6.3.4 handle_event()

```
def buttons.menu.Menu.handle_event (
        self,
        event,
      * args )
```

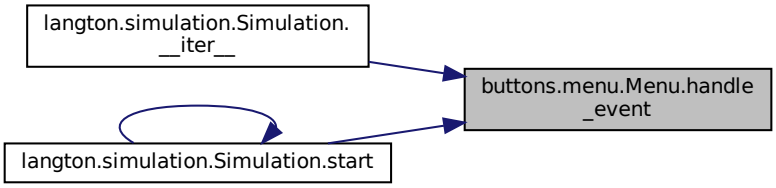
Enable some button of Menu.

Parameters

event	User event
args	variadic args which need to be handled

Definition at line 51 of file menu.py.

Here is the caller graph for this function:



7.6.4 Member Data Documentation

7.6.4.1 btn_list `buttons.menu.Menu.btn_list`

Definition at line 23 of file menu.py.

7.6.4.2 color `buttons.menu.Menu.color`

Definition at line 22 of file menu.py.

7.6.4.3 rect `buttons.menu.Menu.rect`

Definition at line 21 of file menu.py.

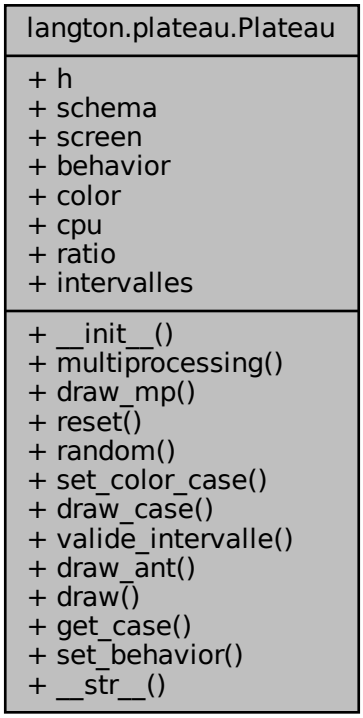
The documentation for this class was generated from the following file:

- menu.py

7.7 langton.plateau.Plateau Class Reference

Represent a Plateau.

Collaboration diagram for langton.plateau.Plateau:



Public Member Functions

- def __init__ (self, behavior, color, taille=(1, 1), res=(4, 4))

  Construct Plateau object.
- def multiprocessing (self, fun)

  create a multiprocess
- def draw_mp (self)

  draw multiprocessing
- def reset (self, start=[0, 0], end=[None, None])

  reset schema in white
- def random (self, start=[0, 0], end=[None, None])

  random color schema
- def set_color_case (self, pos, colour)

  set color in a Case
- def draw_case (self, pos)

  draw a case in specific schema position
- def valide_intervalle (self, start, end)

valide start & end in schema

- def draw_ant (self, pos)

  draw optimize cases near ant

- def draw (self, start=[0, 0], end=[None, None])

  default color schema

- def get_case (self, x, y, res)

  Get a Case in a position.

- def set_behavior (self, color, behavior)

  Modify grid color & behavior.

- def __str__ (self)

  Redefine toString() Plateau method.

Public Attributes

- h
- schema
- screen
- behavior
- color
- cpu
- ratio
- intervalles

7.7.1 Detailed Description

Represent a Plateau.

Definition at line 18 of file plateau.py.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 __init__() `def langton.plateau.Plateau.__init__ (`

```
self,
behavior,
color,
taille = (1, 1),
res = (4, 4) )
```

Construct Plateau object.

Parameters

behavior	String representation of the Plateau behavior
color	A tuple a int reprensent a rgb color
taille	tuple represent number of Case (default (1, 1))
res	Pixel representation of each Case (default (4, 4))

Definition at line 21 of file plateau.py.

7.7.3 Member Function Documentation

7.7.3.1 __str__() def langton.plateau.Plateau.__str__ (self)

Redefine toString() Plateau method.

Definition at line 159 of file plateau.py.

7.7.3.2 draw() def langton.plateau.Plateau.draw (self, start = [0,0], end = [None, None])

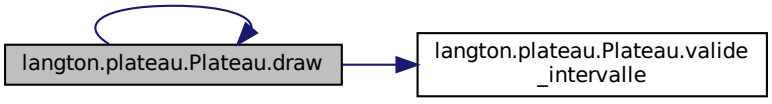
default color schema

Parameters

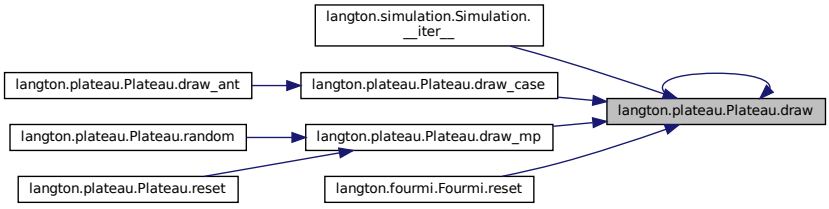
start	list of int index where start to draw (default (0, 0))
end	list of int index where end to draw (default (None, None))

Definition at line 131 of file plateau.py.

Here is the call graph for this function:



Here is the caller graph for this function:



7.7.3.3 draw_ant() `def langton.plateau.Plateau.draw_ant (`

```
self,
pos )
```

draw optimize cases near ant

Parameters

pos	postion of the ant

Definition at line 120 of file plateau.py.

Here is the call graph for this function:

7.7.3.4 draw_case() `def langton.plateau.Plateau.draw_case (`

```
self,
pos )
```

draw a case in specific schema position

Parameters

pos	tuple of position of the Case

Definition at line 92 of file plateau.py.

Here is the call graph for this function:

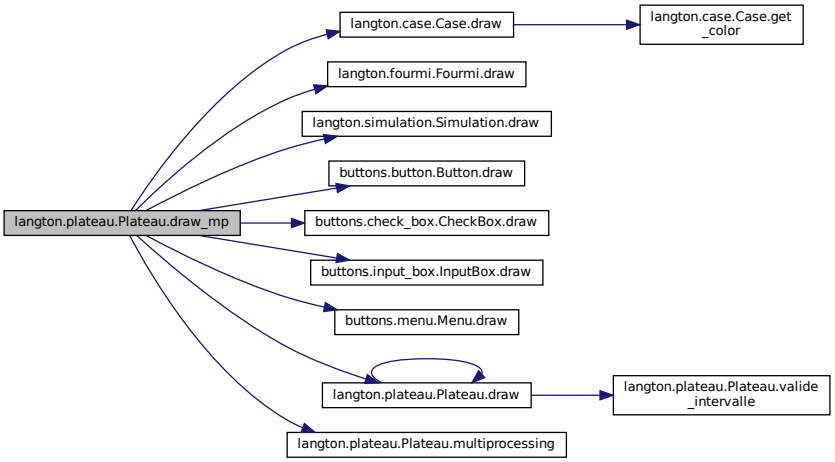
Here is the caller graph for this function:

7.7.3.5 draw_mp() def langton.plateau.Plateau.draw_mp (self)

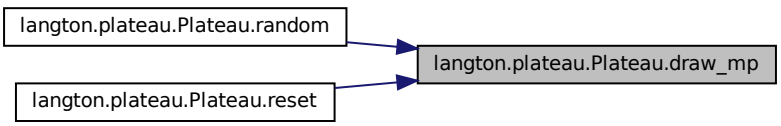
draw multiprocessing

Definition at line 56 of file plateau.py.

Here is the call graph for this function:



Here is the caller graph for this function:



7.7.3.6 get_case() def langton.plateau.Plateau.get_case (self, x, y, res)

Get a Case in a position.

Parameters

x	Horizontal position of the Case
y	Vertical position of the Case
res	Resolution of the Case

Definition at line 142 of file plateau.py.

7.7.3.7 multiprocessing() def langton.plateau.Plateau.multiprocessing (self, fun)

create a multiprocess

Parameters

fun	function reference which call in process

Definition at line 46 of file plateau.py.

Here is the caller graph for this function:

7.7.3.8 random() def langton.plateau.Plateau.random (self, start = [0, 0], end = [None, None])

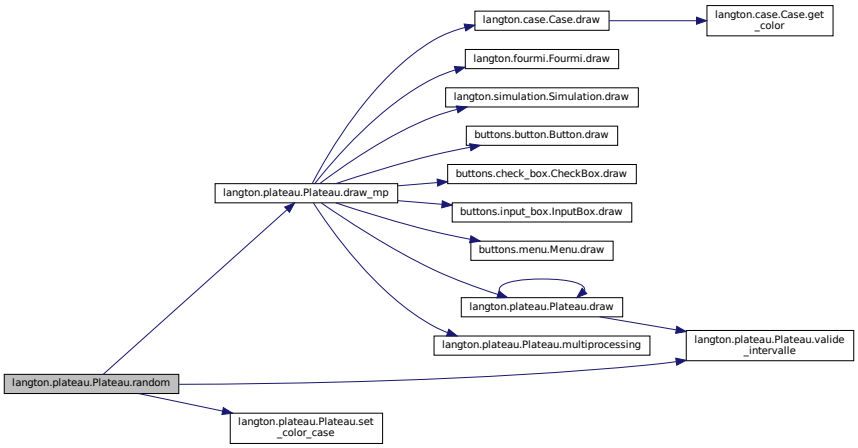
random color schema

Parameters

start	list of int index where start to draw (default (0, 0))
end	list of int index where end to draw (default (None, None))

Definition at line 72 of file plateau.py.

Here is the call graph for this function:



7.7.3.9 reset()

```
def langton.plateau.Plateau.reset (
        self,
        start = [0, 0],
        end = [None, None] )
```

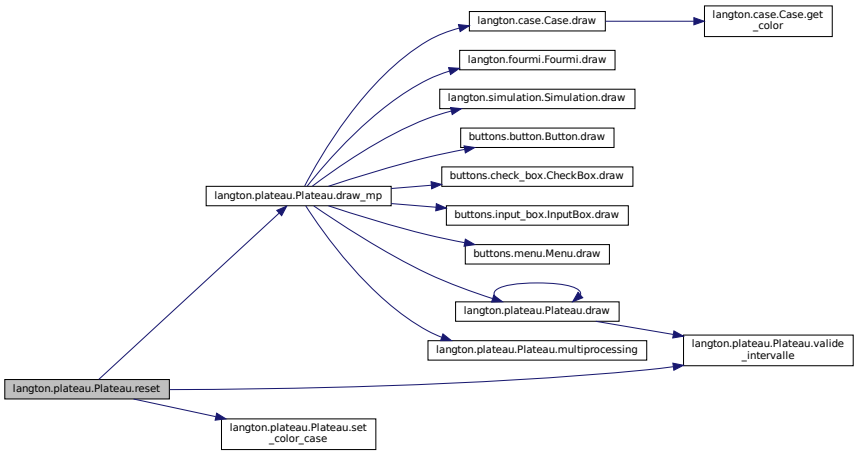
reset schema in white

Parameters

start	list of int index where start to draw (default (0, 0))
end	list of int index where end to draw (default (None, None))

Definition at line 61 of file plateau.py.

Here is the call graph for this function:



7.7.3.10 set_behavior() def langton.plateau.Plateau.set_behavior (self, color, behavior)

Modify grid color & behavior.

Parameters

color	list of color
behavior	string of the simulation behavior

Definition at line 151 of file plateau.py.

7.7.3.11 set_color_case() def langton.plateau.Plateau.set_color_case (self, pos, colour)

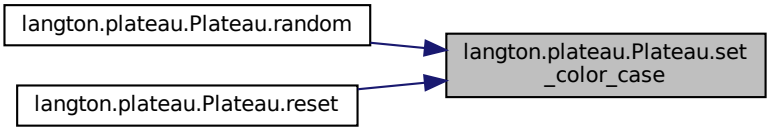
set color in a Case

Parameters

pos	tuple of position of the Case
colour	colour to set in the Case

Definition at line 84 of file plateau.py.

Here is the caller graph for this function:



7.7.3.12 valide_intervalle() `def langton.plateau.Plateau.valide_intervalle (`

self,
start,
end)

valide start & end in schema

Parameters

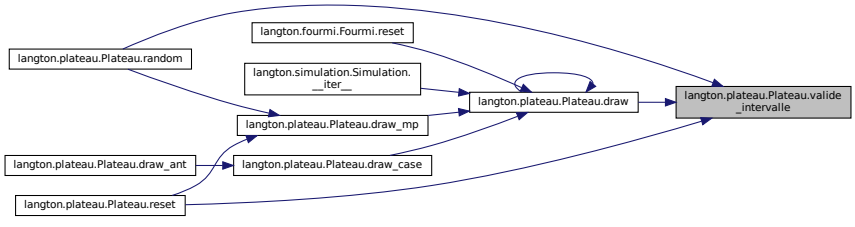
start	list of int index where start to draw (default (0, 0))
end	list of int index where end to draw (default (None, None))

Returns

a valid couple of start & end (valid index)

Definition at line 99 of file plateau.py.

Here is the caller graph for this function:



7.7.4 Member Data Documentation

7.7.4.1 behavior `langton.plateau.Plateau.behavior`

Definition at line 35 of file plateau.py.

7.7.4.2 color `langton.plateau.Plateau.color`

Definition at line 36 of file plateau.py.

7.7.4.3 cpu `langton.plateau.Plateau.cpu`

Definition at line 38 of file plateau.py.

7.7.4.4 h `langton.plateau.Plateau.h`

Definition at line 28 of file plateau.py.

7.7.4.5 intervalles `langton.plateau.Plateau.intervalles`

Definition at line 40 of file plateau.py.

7.7.4.6 ratio `langton.plateau.Plateau.ratio`

Definition at line 39 of file plateau.py.

7.7.4.7 schema `langton.plateau.Plateau.schema`

Definition at line 29 of file plateau.py.

7.7.4.8 screen `langton.plateau.Plateau.screen`

Definition at line 33 of file plateau.py.

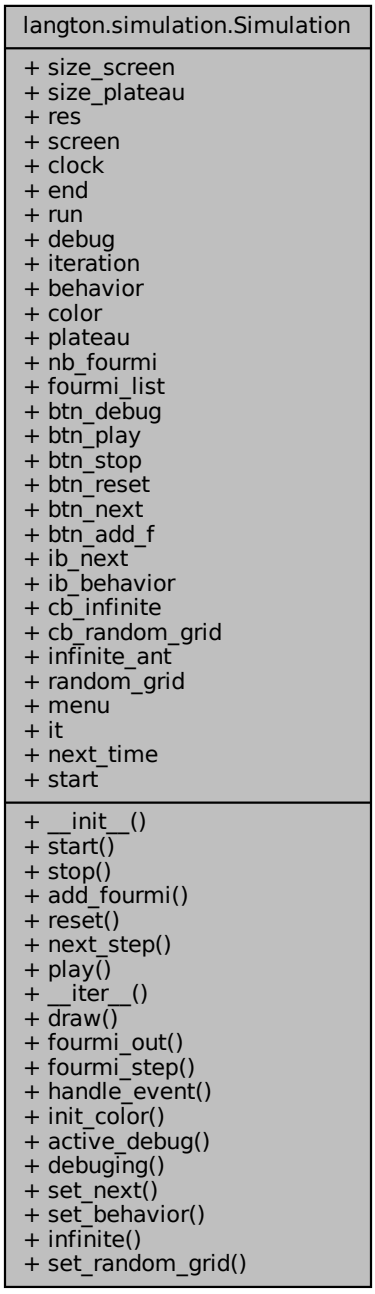
The documentation for this class was generated from the following file:

- plateau.py

7.8 langton.simulation.Simulation Class Reference

Represent a Simulation.

Collaboration diagram for langton.simulation.Simulation:



Public Member Functions

- def __init__ (self, size_screen=const .DEFAULT_SCREEN_SIZE, size_plateau=const .DEFAULT_↩ PLATEAU_SIZE, res=const .DEFAULT_RESOLUTION)

Construct Simulation object.

- def start (self)

  Global start, here when ants aren't running.
- def stop (self)

  Stop the game by set self.run to false.
- def add_fourmi (self)

  Add an ant on the grid.
- def reset (self)

  Reset the simulation.
- def next_step (self)

  Play simulation iterator next_number times, (default 1)
- def play (self)

  Play Simulation loop.
- def __iter__ (self)

  Simulation iterator.
- def draw (self)

  Draw the map and ants.
- def fourmi_out (self)

  Check if an ant is out of index of the grid.
- def fourmi_step (self)

  Move all ants once.
- def handle_event (self)

  Event listener user interactions: Button & Keyboard.
- def init_color (self)

  Init random color from simulation behavior.
- def active_debug (self)

  Change boolean value for debug option button.
- def debuging (self)

  Print the total of iteration since the simulation is running.
- def set_next (self, text)

  Set the step for next function.
- def set_behavior (self, text)

  Set the behavior before add ants and game start.
- def infinite (self)

  Define if ants are in an infinite place.
- def set_random_grid (self)

  Random Grid CheckBox function.

Public Attributes

- size_screen
- size_plateau
- res
- screen
- clock
- end
- run
- debug
- iteration
- behavior
- color

- plateau
- nb_fourmi
- fourmi_list
- btn_debug
- btn_play
- btn_stop
- btn_reset
- btn_next
- btn_add_f
- ib_next
- ib_behavior
- cb_infinite
- cb_random_grid
- infinite_ant
- random_grid
- menu
- it
- next_time
- start

7.8.1 Detailed Description

Represent a Simulation.

Definition at line 16 of file simulation.py.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 __init__() def langton.simulation.Simulation.__init__ (self, size_screen = const.DEFAULT_SCREEN_SIZE, size_plateau = const.DEFAULT_PLATEAU_SIZE, res = const.DEFAULT_RESOLUTION)

Construct Simulation object.

Parameters

size_screen	Size of the window
size_plateau	Size of the grid where stand ants
res	Number of pixel define size of Case and Fourmi

Definition at line 19 of file simulation.py.

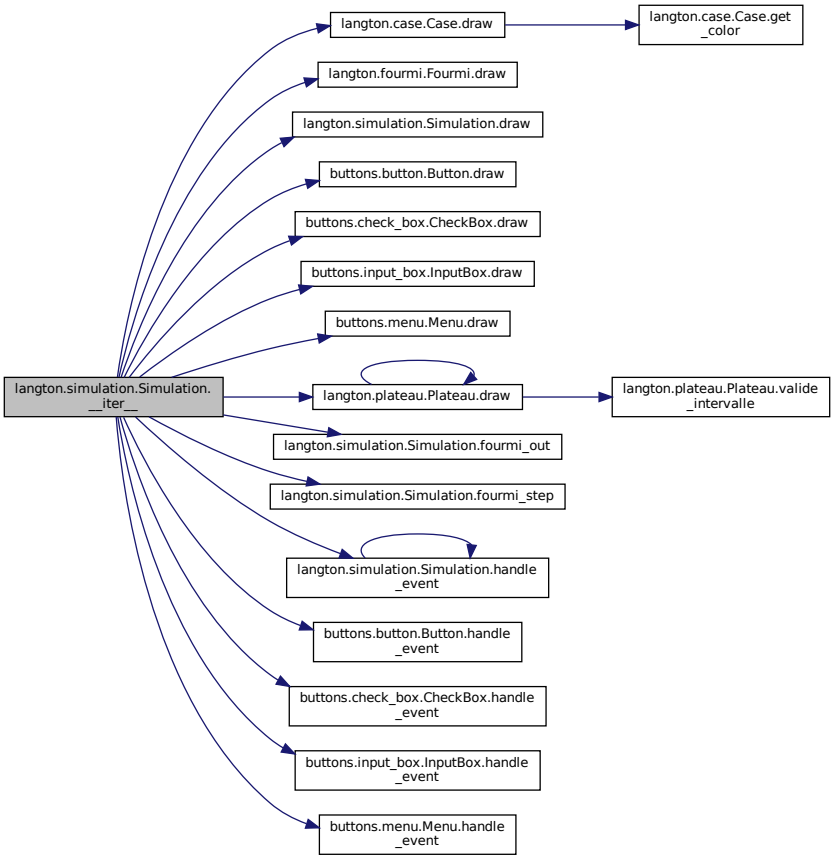
7.8.3 Member Function Documentation

7.8.3.1 __iter__() def langton.simulation.Simulation.__iter__ (self)

Simulation iterator.

Definition at line 233 of file simulation.py.

Here is the call graph for this function:



7.8.3.2 active_debug() def langton.simulation.Simulation.active_debug (self)

Change boolean value for debug option button.

Definition at line 321 of file simulation.py.

7.8.3.3 add_fourmi() def langton.simulation.Simulation.add_fourmi (self)

Add an ant on the grid.

Definition at line 157 of file simulation.py.

7.8.3.4 debuging() def langton.simulation.Simulation.debuging (self)

Print the total of iteration since the simulation is running.

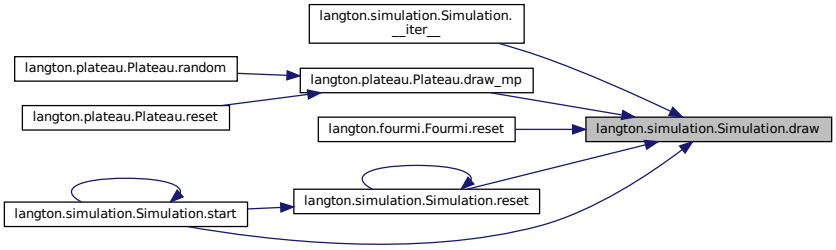
Definition at line 325 of file simulation.py.

7.8.3.5 draw() def langton.simulation.Simulation.draw (self)

Draw the map and ants.

Definition at line 249 of file simulation.py.

Here is the caller graph for this function:

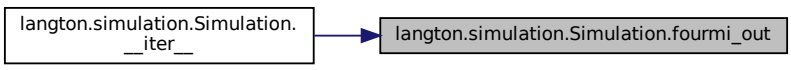


7.8.3.6 fourmi_out() def langton.simulation.Simulation.fourmi_out (self)

Check if an ant is out of index of the grid.

Definition at line 256 of file simulation.py.

Here is the caller graph for this function:

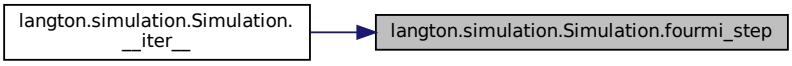


7.8.3.7 fourmi_step() `def langton.simulation.Simulation.fourmi_step ( self )`

Move all ants once.

Definition at line 269 of file simulation.py.

Here is the caller graph for this function:

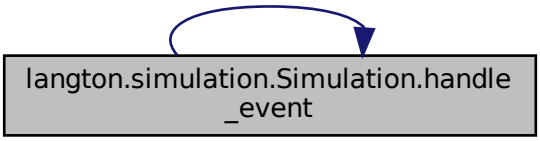


7.8.3.8 handle_event() `def langton.simulation.Simulation.handle_event ( self )`

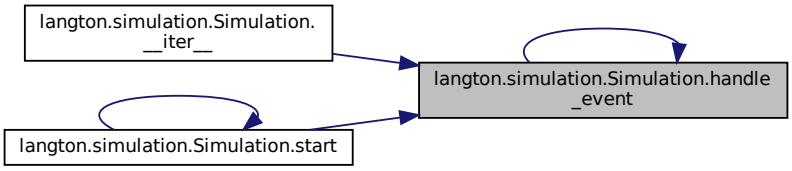
Event listener user interactions: Button & Keyboard.

Definition at line 275 of file simulation.py.

Here is the call graph for this function:



Here is the caller graph for this function:



7.8.3.9 infinite() `def langton.simulation.Simulation.infinite ( self )`

Define if ants are in an infinite place.

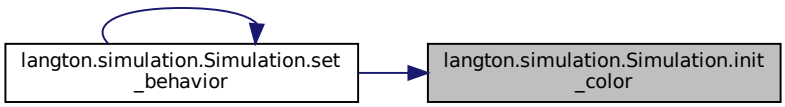
Definition at line 362 of file simulation.py.

7.8.3.10 init_color() `def langton.simulation.Simulation.init_color ( self )`

Init random color from simulation behavior.

Definition at line 309 of file simulation.py.

Here is the caller graph for this function:



7.8.3.11 next_step() `def langton.simulation.Simulation.next_step ( self )`

Play simulation iterator next_number times, (default 1)

Exceptions

StopIteration	If Simulation ended
Exception	If Simulation already run

Definition at line 205 of file simulation.py.

7.8.3.12 play() `def langton.simulation.Simulation.play ( self )`

Play Simulation loop.

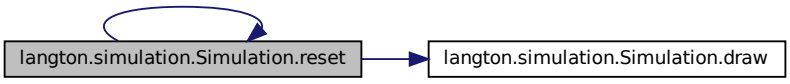
Definition at line 219 of file simulation.py.

7.8.3.13 reset() `def langton.simulation.Simulation.reset ( self )`

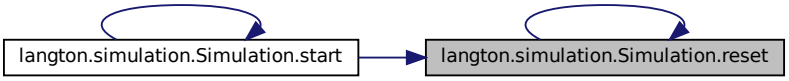
Reset the simulation.

Definition at line 188 of file simulation.py.

Here is the call graph for this function:



Here is the caller graph for this function:

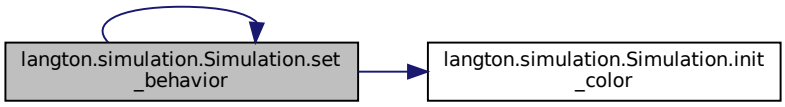


7.8.3.14 set_behavior() `def langton.simulation.Simulation.set_behavior ( self, text )`

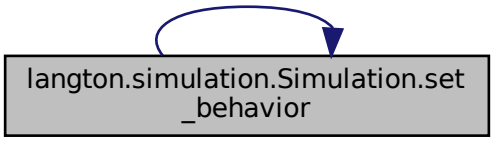
Set the behavior before add ants and game start.

Definition at line 351 of file simulation.py.

Here is the call graph for this function:



Here is the caller graph for this function:



7.8.3.15 set_next() def langton.simulation.Simulation.set_next (self, text)

Set the step for next function.

Definition at line 343 of file simulation.py.

7.8.3.16 set_random_grid() def langton.simulation.Simulation.set_random_grid (self)

Random Grid CheckBox function.

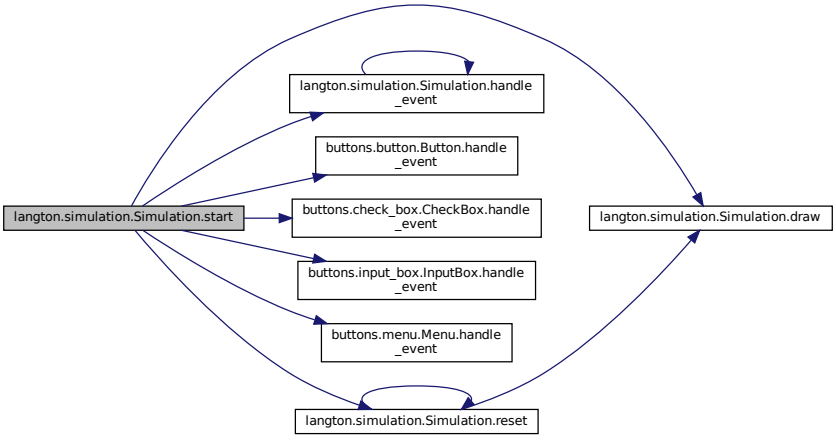
Definition at line 372 of file simulation.py.

7.8.3.17 start() def langton.simulation.Simulation.start (self)

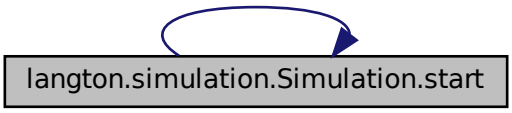
Global start, here when ants aren't running.

Definition at line 127 of file simulation.py.

Here is the call graph for this function:



Here is the caller graph for this function:



7.8.3.18 stop() `def langton.simulation.Simulation.stop ( self )`

Stop the game by set self.run to false.

Definition at line 153 of file simulation.py.

7.8.4 Member Data Documentation

7.8.4.1 behavior `langton.simulation.Simulation.behavior`

Definition at line 46 of file simulation.py.

7.8.4.2 btn_add_f `langton.simulation.Simulation.btn_add_f`

Definition at line 86 of file simulation.py.

7.8.4.3 btn_debug `langton.simulation.Simulation.btn_debug`

Definition at line 63 of file simulation.py.

7.8.4.4 btn_next `langton.simulation.Simulation.btn_next`

Definition at line 81 of file simulation.py.

7.8.4.5 btn_play `langton.simulation.Simulation.btn_play`

Definition at line 67 of file simulation.py.

7.8.4.6 btn_reset `langton.simulation.Simulation.btn_reset`

Definition at line 77 of file simulation.py.

7.8.4.7 btn_stop `langton.simulation.Simulation.btn_stop`

Definition at line 72 of file simulation.py.

7.8.4.8 cb_infinite `langton.simulation.Simulation.cb_infinite`

Definition at line 103 of file simulation.py.

7.8.4.9 cb_random_grid `langton.simulation.Simulation.cb_random_grid`

Definition at line 107 of file simulation.py.

7.8.4.10 clock `langton.simulation.Simulation.clock`

Definition at line 35 of file simulation.py.

7.8.4.11 color `langton.simulation.Simulation.color`

Definition at line 47 of file simulation.py.

7.8.4.12 debug `langton.simulation.Simulation.debug`

Definition at line 42 of file simulation.py.

7.8.4.13 end `langton.simulation.Simulation.end`

Definition at line 38 of file simulation.py.

7.8.4.14 fourmi_list `langton.simulation.Simulation.fourmi_list`

Definition at line 60 of file simulation.py.

7.8.4.15 ib_behavior `langton.simulation.Simulation.ib_behavior`

Definition at line 97 of file simulation.py.

7.8.4.16 ib_next `langton.simulation.Simulation.ib_next`

Definition at line 93 of file simulation.py.

7.8.4.17 infinite_ant `langton.simulation.Simulation.infinite_ant`

Definition at line 111 of file simulation.py.

7.8.4.18 it `langton.simulation.Simulation.it`

Definition at line 124 of file simulation.py.

7.8.4.19 iteration `langton.simulation.Simulation.iteration`

Definition at line 43 of file simulation.py.

7.8.4.20 menu `langton.simulation.Simulation.menu`

Definition at line 115 of file simulation.py.

7.8.4.21 nb_fourmi `langton.simulation.Simulation.nb_fourmi`

Definition at line 59 of file simulation.py.

7.8.4.22 next_time `langton.simulation.Simulation.next_time`

Definition at line 125 of file simulation.py.

7.8.4.23 plateau `langton.simulation.Simulation.plateau`

Definition at line 50 of file simulation.py.

7.8.4.24 random_grid `langton.simulation.Simulation.random_grid`

Definition at line 112 of file simulation.py.

7.8.4.25 res `langton.simulation.Simulation.res`

Definition at line 30 of file simulation.py.

7.8.4.26 run `langton.simulation.Simulation.run`

Definition at line 39 of file simulation.py.

7.8.4.27 screen `langton.simulation.Simulation.screen`

Definition at line 33 of file simulation.py.

7.8.4.28 size_plateau `langton.simulation.Simulation.size_plateau`

Definition at line 29 of file simulation.py.

7.8.4.29 size_screen `langton.simulation.Simulation.size_screen`

Definition at line 28 of file simulation.py.

7.8.4.30 start `langton.simulation.Simulation.start`

Definition at line 302 of file simulation.py.

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

- simulation.py

8 File Documentation

8.1 README.md File Reference

8.2 button.py File Reference

Classes

- class buttons.button.Button

  Represent a Button.

Namespaces

- namespace buttons

  Package corresponding to all the functions specific to buttons.
- namespace buttons.button

  Button package.

8.3 button.py

Go to the documentation of this file.

```
00001 """!
00002 @brief Button package
00003 @author Durel Enzo
00004 @author Mallepeyre Nourrane
00005 @version 1.0
00006 """
00007
00008 import pygame
00009 from utils import color
00010
00011 class Button:
00012     """!@brief Represent a Button"""
00013
00014     def __init__(self, pos, size, text=”, fun=None):
00015         """!@brief Construct Button object
00016         @param pos A tuple position of top left button corner
00017         @param size A tuple represent the size of button (width, height)
00018         @param text String affiliate to the button
00019         @param fun Function reference for button event
00020         """
00021         # RECTANGLE #
```

```
00022         self.rect = pygame.Rect(pos, size) # rect collison button
00023         # COLORS #
00024         self.inactive_color = color.INACTIVE_BUTTON_COLOR # when button inactive
00025         self.active_color = color.ACTIVE_BUTTON_COLOR # when button is clicked on
00026         self.disable_color = color.DISABLE_BUTTON_COLOR
00027         # self.disable_color = DISABLE_BUTTON_COLOR # when button disabled
00028         self.color = color.INACTIVE_BUTTON_COLOR # current color
00029         self.text_color = color.TEXT_BUTTON_COLOR # text inside color
00030         self.hover_color = color.HOVER_BUTTON_COLOR # when mouse on the button
00031         # TEXT #
00032         self.text = text # string text
00033         self.font = pygame.font.Font(None, 32) # font of the text
00034         self.txt_surf = self.font.render(text, True, self.text_color) # surface txt
00035         # STATE #
00036         self.active = False # if button is active
00037         self.b_disable = False
00038         # FUNCTION #
00039         self.fun = fun # fonction reference
00040         # STYLE #
00041         self.border_radius = 4 # round corners of the button
00042 
00043     def enable(self):
00044         """!@brief Enable the button
00045         This method enable the button (the user can click on it).
00046         """
00047         if self.b_disable:
00048             self.b_disable = False
00049             self.color = self.inactive_color
00050 
00051     def disable(self):
00052         """!@brief Disable the button
00053         This method disable the button (the user can't click on it).
00054         """
00055         if not self.b_disable:
00056             self.b_disable = True
00057             self.color = self.disable_color # disable color
00058 
00059     def draw(self, screen):
00060         """!@brief Draw the button
00061         This method draw the button rectangle with pyGame draw.rect function
00062         and the text with screen.blit function.
00063         @param screen Pygame screen object where the button with be draw
00064         @exception Exception Used if pyGame is under update
00065         """
00066         try:
00067             pygame.draw.rect(screen,
00068                              self.color,
00069                              self.rect,
00070                              border_radius=self.border_radius)
00071         except Exception:
00072             print("Mettez a jour PyGame")
00073             pygame.draw.rect(screen,
00074                              self.color,
00075                              self.rect)
00076         text_w = self.txt_surf.get_width()
00077         text_h = self.txt_surf.get_height()
00078         screen.blit(self.txt_surf,
00079                     (self.rect.x+self.rect.w/2-text_w/2, self.rect.y+self.rect.h/2-text_h/2))
00080 
00081     def handle_event(self, event):
00082         """!@brief User input method
00083         This method operate users input with event.type pyGame attributs
00084 
00085         @param event Event user input
00086         """
00087         if self.b_disable:
00088             return;
00089         if event.type == pygame.MOUSEBUTTONDOWN:
00090             """mouse click down"""
00091             if self.rect.collidepoint(event.pos):
00092                 self.active = True
00093                 self.color = self.active_color
00094                 if self.fun is not None:
00095                     self.fun()
00096         elif event.type == pygame.MOUSEBUTTONUP:
00097             """mouse click up"""
00098             self.active = False
00099             if self.rect.collidepoint(event.pos):
00100                 self.color = self.hover_color
00101             else :
00102                 self.color = self.inactive_color
00103         elif event.type == pygame.MOUSEMOTION:
00104             """mouse hover"""
00105             if self.rect.collidepoint(event.pos):
00106                 if self.active:
00107                     self.color = self.active_color
00108                 else:
```

```
00109                     self.color = self.hover_color
00110             else:
00111                 self.color = self.inactive_color
00112 
00113     def get_pos(self):
00114         """!@brief Get the button top left position
00115         @return A tuple of the position (x, y)
00116         """
00117         return (self.rect.x, self.rect.y)
00118 
00119     def get_size(self):
00120         """!@brief Get the button size
00121         @return A tuple of the size (w, h)
00122         """
00123         return (self.rect.w, self.rect.h)
```

8.4 check_box.py File Reference

Classes

- class buttons.check_box.CheckBox

  Represent a CheckBox.

Namespaces

- namespace buttons

  Package corresponding to all the functions specific to buttons.
- namespace buttons.check_box

  CheckBox package.

8.5 check_box.py

Go to the documentation of this file.

```
00001 """!
00002 @brief CheckBox package
00003 @author Durel Enzo
00004 @author Mallepeyre Nourrane
00005 @version 1.0
00006 """
00007 import pygame
00008 import time
00009 
00010 from utils import color
00011 from buttons import Button
00012 
00013 class CheckBox(Button):
00014     """!@brief Represent a CheckBox"""
00015 
00016     def __init__(self, pos, size, text=”, fun=None):
00017         super().__init__(pos, size, text=text, fun=fun)
00018         # TEXT #
00019         # self.check_rect = pygame.Rect(pos[0], pos[1], 20, 20)
00020         # FONT #
00021         self.fontfont = pygame.font.Font(None, size[1])
00022         self.txt_surftxt_surf = self.fontfont.render(text, True, self.text_colortext_color) # surface
      txt
00023         # COLORS #
00024         self.colorcolor = color.INACTIVE_CB_COLOR
00025         self.inactive_colorinactive_color = color.INACTIVE_CB_COLOR
00026         self.active_coloractive_color = color.ACTIVE_CB_COLOR
00027         self.disable_colordisable_color = color.DISABLE_CB_COLOR
00028         self.disable_active_color = color.DISABLE_ACTIVE_CB_COLOR
00029         self.text_colortext_color = color.TEXT_IB_COLOR
00030         # TIME #
00031         self.delay = 1_000_000 # en ns
00032         self.active_time = time.time_ns()
00033 
00034     def disable(self):
00035         if not self.b_disableb_disable:
```

```
00036             self.b_disableb_disable = True
00037             if self.activeactive:
00038                 self.colorcolor = self.disable_active_color
00039             else:
00040                 self.colorcolor = self.disable_colordisable_color
00041 
00042     def handle_event(self, event):
00043         if self.b_disableb_disable:
00044             return;
00045         if event.type == pygame.MOUSEBUTTONUP:
00046             if self.rect.collidepoint(event.pos):
00047 
00048                 self.activeactive = not self.activeactive
00049                 self.colorcolor = self.active_coloractive_color if self.activeactive else
      self.inactive_colorinactive_color
00050 
00051                 if self.fun is not None:
00052                     self.fun()
00053                 else:
00054                     print("CheckBox has no function.")
00055 
00056     def draw(self, screen):
00057         try:
00058             pygame.draw.rect(screen,
00059                              self.colorcolor,
00060                              self.rect,
00061                              border_radius=self.border_radius)
00062         except Exception:
00063             print("You need to update PyGame.")
00064             pygame.draw.rect(screen,
00065                              self.colorcolor,
00066                              self.rect)
00067         text_h = self.txt_surftxt_surf.get_height()
00068         screen.blit(self.txt_surftxt_surf,
00069                     (self.rect.x+self.rect.w+10, self.rect.y+self.rect.h/2-text_h/2))
```

8.6 input_box.py File Reference

Classes

- class buttons.input_box.InputBox

  Represent an InputBox.

Namespaces

- namespace buttons

  Package corresponding to all the functions specific to buttons.
- namespace buttons.input_box

  InputBox package.

8.7 input_box.py

Go to the documentation of this file.

```
00001 """!
00002 @brief InputBox package
00003 @author Durel Enzo
00004 @author Mallepeyre Nourrane
00005 @version 1.0
00006 """
00007 
00008 import pygame
00009 from utils import color
00010 
00011 from buttons import Button
00012 
00013 class InputBox(Button):
00014     """!@brief Represent an InputBox"""
00015 
00016     def __init__(self, pos, size, text='', fun=None, max_len=5):
```

```
00017         super().__init__(pos, size, text=text, fun=fun)
00018         # RECT #
00019         self.int_rect = pygame.Rect(pos[0]+5, pos[1]+5, size[0]-10, size[1]-10)
00020         # COLORS #
00021         self.colorcolor = color.INACTIVE_IB_COLOR
00022         self.inactive_colorinactive_color = color.INACTIVE_IB_COLOR
00023         self.active_coloractive_color = color.ACTIVE_IB_COLOR
00024         self.disable_colordisable_color = color.DISABLE_IB_COLOR
00025         self.int_color = color.dic["white"]
00026         self.text_colortext_color = color.TEXT_IB_COLOR
00027         # TEXT #
00028         self.max_len = max_len
00029 
00030     def handle_event(self, event):
00031         if self.b_disable:
00032             return;
00033         if event.type == pygame.MOUSEBUTTONUP:
00034             if self.rect.collidepoint(event.pos):
00035                 self.activeactive = not self.activeactive
00036             else:
00037                 self.activeactive = False
00038             self.colorcolor= self.active_coloractive_color if self.activeactive else
      self.inactive_colorinactive_color
00039         if event.type == pygame.KEYDOWN:
00040             if self.activeactive:
00041                 if event.key == pygame.K_RETURN:
00042                     if self.fun is not None:
00043                         self.fun(self.texttext)
00044                     else:
00045                         print("Button has no function.")
00046                     self.texttext = ”
00047                     self.activeactive = False
00048                     self.colorcolor = self.inactive_colorinactive_color
00049                 elif event.key == pygame.K_BACKSPACE:
00050                     self.texttext = self.texttext[:-1]
00051                 else:
00052                     if len(self.texttext) < self.max_len:
00053                         self.texttext += event.unicode
00054                     else:
00055                         print(f"Maximum of char : {self.max_len}")
00056                 self.txt_surftxt_surf = self.font.render(self.texttext, True,
      self.text_colortext_color)
00057 
00058     def draw(self, screen):
00059         try:
00060             pygame.draw.rect(screen,
00061                              self.colorcolor,
00062                              self.rect,
00063                              border_radius=self.border_radius)
00064             pygame.draw.rect(screen,
00065                              self.int_color,
00066                              self.int_rect,
00067                              border_radius=self.border_radius)
00068         except Exception:
00069             print("You need to update PyGame.")
00070             pygame.draw.rect(screen,
00071                              self.colorcolor,
00072                              self.rect)
00073             pygame.draw.rect(screen,
00074                              self.int_color,
00075                              self.int_rect)
00076         text_w = self.txt_surftxt_surf.get_width()
00077         text_h = self.txt_surftxt_surf.get_height()
00078         screen.blit(self.txt_surftxt_surf,
00079                     (self.rect.x+self.rect.w/2-text_w/2, self.rect.y+self.rect.h/2-text_h/2))
```

8.8 menu.py File Reference

Classes

- class buttons.menu.Menu

  Represent a Menu.

Namespaces

- namespace buttons

  Package corresponding to all the functions specific to buttons.
- namespace buttons.menu

  Menu package.

8.9 menu.py

Go to the documentation of this file.

```
00001 """!
00002 @brief Menu package
00003 @author Durel Enzo
00004 @author Mallepeyre Nourrane
00005 @version 1.0
00006 """
00007
00008 import pygame
00009 from utils import color
00010
00011 class Menu:
00012     """!@brief Represent a Menu"""
00013
00014     def __init__(self, pos, size, btn_list=[], color=color.MENU_COLOR):
00015         """!@brief Construct a Menu
00016         @param pos A tuple position of top left Menu corner
00017         @param size A tuple represent the size of Menu (width, height)
00018         @param btn_list A list of button in the Menu
00019         @param color The Color of the Menu
00020         """
00021         self.rect = pygame.Rect(pos, size)
00022         self.color = color
00023         self.btn_list = btn_list
00024
00025     def draw(self, screen):
00026         """!@brief Draw the Menu
00027         @param screen Screen pyGame attributs
00028         """
00029         pygame.draw.rect(screen, self.color, self.rect)
00030         for x in self.btn_list:
00031             x.draw(screen)
00032
00033     def disable(self, *args):
00034         """!@brief Disable some button of Menu
00035         @param args variadic args which need to be disable
00036         """
00037         if len(args) == 0:
00038             args = self.btn_list
00039         for button in args:
00040             button.disable()
00041
00042     def enable(self, *args):
00043         """!@brief Enable some button of Menu
00044         @param args variadic args which need to be enable
00045         """
00046         if len(args) == 0:
00047             args = self.btn_list
00048         for button in args:
00049             button.enable()
00050
00051     def handle_event(self, event, *args):
00052         """!@brief Enable some button of Menu
00053         @param event User event
00054         @param args variadic args which need to be handled
00055         """
00056         if len(args) == 0:
00057             args = self.btn_list
00058         for button in args:
00059             button.handle_event(event)
```

8.10 __init__.py File Reference

Namespaces

- namespace buttons

  Package corresponding to all the functions specific to buttons.

8.11 buttons/__init__.py

Go to the documentation of this file.

```
00001 """!
```

```
00002 @brief Package corresponding to all the functions specific to buttons
00003 @author Durel Enzo
00004 @author Mallepeyre Nourrane
00005 @version 1.0
00006 """
00007 from .button import Button
00008 from .check_box import CheckBox
00009 from .input_box import InputBox
00010 from .menu import Menu
```

8.12 __init__.py File Reference

Namespaces

- namespace langton

  Package corresponding to all the functions specific to Langton.

8.13 langton/__init__.py

Go to the documentation of this file.

```
00001 """!
00002 @brief Package corresponding to all the functions specific to Langton
00003 @author Durel Enzo
00004 @author Mallepeyre Nourrane
00005 @version 1.0
00006 """
00007 from .simulation import Simulation
00008 from .fourmi import Fourmi
00009 from .case import Case
00010 from .plateau import Plateau
```

8.14 __init__.py File Reference

Namespaces

- namespace utils

  Package corresponding to all the const need for the program.

8.15 utils/__init__.py

Go to the documentation of this file.

```
00001 """!
00002 @brief Package corresponding to all the const need for the program
00003 @author Durel Enzo
00004 @author Mallepeyre Nourrane
00005 @version 1.0
00006 """
00007 from .color import *
00008 from .const import *
```

8.16 case.py File Reference

Classes

- class langton.case.Case

  Represent a Case.

Namespaces

- namespace langton

  Package corresponding to all the functions specific to Langton.
- namespace langton.case

  langton.case package

8.17 case.py

Go to the documentation of this file.

```
00001 """!
00002 @brief langton.case package
00003 @author Durel Enzo
00004 @author Mallepeyre Nourrane
00005 @version 1.0
00006 """
00007
00008 import pygame
00009 from utils import color
00010
00011 class Case:
00012     """!@brief Represent a Case"""
00013
00014     def __init__(self, size=(1, 1), pos=(0, 0)):
00015         """!@brief Construct Case object
00016         @param size Pixel size of the case (pygame)
00017         @param pos Position of the case
00018         """
00019         self.screen = pygame.display.get_surface()
00020         self.cur_color = color.dic["white"]
00021         self.w, self.h = size
00022         self.x, self.y = pos
00023
00024     def set_color(self, colour):
00025         """!@brief Set a color the the Case
00026         @param colour A tuple of int representing a rgb color
00027         """
00028         self.cur_color = self.validate_color(colour)
00029
00030     def get_color(self):
00031         """!@brief get the Case color
00032         @return the Case color
00033         """
00034         return self.cur_color
00035
00036     def validate_color(self, colour):
00037         """!@brief Verify if it's a valid colour
00038         @param colour A tuple of int representing a rgb colour
00039         @return The valide colour
00040         @exception Exception Not a valid colour
00041         """
00042         if len(colour) == 3:
00043             for i in colour:
00044                 if i < 0 or i > 255:
00045                     raise(Exception("Invalide element in colour argument"))
00046         else:
00047             raise(Exception("Invalide length of colour argument"))
00048         return colour
00049
00050     def draw(self):
00051         """!@brief draw the Case"""
00052         pygame.draw.rect(
00053             self.screen,
00054             self.get_color(),
00055             pygame.Rect(self.x*self.w,
00056                         self.y*self.h,
00057                         self.w,
00058                         self.h))
```

8.18 fourmi.py File Reference

Classes

- class langton.fourmi.Fourmi

  Represent a Fourmi de Langton.

Namespaces

- namespace langton

  Package corresponding to all the functions specific to Langton.
- namespace langton.fourmi

  langton.fourmi package

8.19 fourmi.py

Go to the documentation of this file.

```
00001 """!
00002 @brief langton.fourmi package
00003 @author Durel Enzo
00004 @author Mallepeyre Nourrane
00005 @version 1.0
00006 """
00007
00008 import pygame
00009 from utils import color
00010
00011 class Fourmi :
00012     """!@brief Represent a Fourmi de Langton"""
00013
00014     def __init__(self, coords=(0, 0), taille=4, speed=1, direction=0,
00015                  color=[(255, 255, 255), (0, 0, 0)], behavior="LR"):
00016         """!@brief Constuct Fourmi object
00017         This is the constructor of the Fourmi object.
00018         @param coords coordinate where ant takes place (default (0, 0))
00019         @param taille number of pixels represent an ant (default 4)
00020         @param speed number of case ant moving (default 1)
00021         @param direction index of first direction (default 0 ("up"))
00022         @param color list of tuple represent the list of color used for
00023         behavior (default: (255, ...), (0, ...))
00024         @param behavior string representation of the ant behavior (default:
00025         'LR')
00026         """
00027         # Ant position #
00028         self.x = int(coords[0]) # current position
00029         self.y = int(coords[1])
00030         self.begin_x = int(coords[0]) # save begin position
00031         self.begin_y = int(coords[1])
00032         # Ant movement #
00033         self.speed = speed # ant speed (pixel per movement)
00034         self.rotation = ['up', 'right', 'down', 'left'] # ant list rotation
00035         self.nb_direction = len(self.rotation) # length of rotation available
00036         self.begin_direction = direction%self.nb_direction #save init direction
00037         self.index_direction = self.begin_direction #current index of direction
00038         self.direction = self.rotation[self.index_direction]
00039         # Graphics attributes #
00040         self.screen = pygame.display.get_surface()
00041         self.taille = taille
00042         self.out = False
00043         # Ant behavior #
00044         self.color = color
00045         self.behavior = behavior
00046
00047     def set_out(self):
00048         """!@brief Ant is out
00049         This method makes the ant out. She can't do anything anymore.
00050         """
00051         self.out = True
00052
00053     def is_out(self):
00054         """!@brief Ask if fourmi is out
00055         This method return the out state of the ant.
00056         @return boolean
00057         """
00058         return self.out
00059
00060     def one_step(self, case):
00061         """!@brief An ant complete movement
00062         This method make the ant follows a complete movement (rotate, change
00063         color, move).
00064         @param case Case where the ant begin its step
00065         """
00066         self.rotate(case)
00067         self.inverse_color_case(case)
00068         self.conduct()
00069
```

```
00070     def reset(self):
00071         """!@brief Reset the Ant
00072         This method hard reset the ant at its beginning direction, position.
00073         """
00074         self.index_direction = self.begin_direction
00075         self.x = self.begin_x
00076         self.y = self.begin_y
00077         self.direction = self.rotation[self.index_direction]
00078         self.draw()
00079
00080     def inverse_color_case(self, case):
00081         """!@brief Inverse Case color
00082         This method change the color of the case where the ant is.
00083         @param case Case where the ant is
00084         """
00085         index = self.color.index(case.get_color())
00086         case.set_color(self.color[(index+1)%len(self.color)])
00087
00088     def rotate(self, case):
00089         """!@brief Rotation the ant
00090         This method rotate the ant following the ant's behavior.
00091         @param case Case where the ant is.
00092         """
00093         index = self.color.index(case.get_color())
00094         index_behavior = self.behavior[index]
00095         if index_behavior == 'L':
00096             self.rotate_left()
00097         elif index_behavior == 'R':
00098             self.rotate_right()
00099
00100     def conduct(self) :
00101         """!@brief Move the ant following its conduct
00102         This method moves the ant compare to the conduct wanted. Here the ant
00103         move in the direction where it watches.
00104         """
00105         if self.direction == 'up' :
00106             self.move_up()
00107         elif self.direction == 'down' :
00108             self.move_down()
00109         elif self.direction == 'left' :
00110             self.move_left()
00111         elif self.direction == 'right' :
00112             self.move_right()
00113
00114     def move(self, coords=(0, 0)):
00115         """!@brief Vectorial ant movement
00116         This method reprensent primitive ant movement. It's update the x and y
00117         of the ant.
00118         @param coords A tuple represents a movement vector (default (0,0))
00119         """
00120         self.x += int(coords[0])
00121         self.y += int(coords[1])
00122
00123     def move_down(self):
00124         """!@brief Ant move down
00125         This method calls move() with a down vector (0, y).
00126         """
00127         self.move((0, self.speed))
00128
00129     def move_up(self):
00130         """!@brief Ant move up
00131         This method calls move() with a up vector (0, -y).
00132         """
00133         self.move((0, -self.speed))
00134
00135     def move_right(self):
00136         """!@brief Ant move right
00137         This method calls move() with a right vector (x, 0).
00138         """
00139         self.move((self.speed, 0))
00140
00141     def move_left(self):
00142         """!@brief Ant move left
00143         This method calls move() with a left vector (-x, 0).
00144         """
00145         self.move((-self.speed, 0))
00146
00147     def rotate_right(self):
00148         """!@brief Ant rotate left
00149         This method rotate the ant in its right. It means the index of the
00150         current rotation is increment by one in the list of rotation.
00151         """
00152         self.direction = \
00153             self.rotation[(self.index_direction+1)%self.nb_direction]
00154         self.index_direction += 1
00155
00156     def rotate_left(self):
```

```
00157         """!@brief Ant rotate left
00158         This method rotate the ant in its left. It means the index of the
00159         current rotation is decrement by one in the list of rotation.
00160         """
00161         self.direction = \
00162             self.rotation[(self.index_direction-1)%self.nb_direction]
00163         self.index_direction -= 1
00164 
00165     def draw(self):
00166         """!@brief Ant draw
00167         This method draw the ant with pyGame draw.rect function. Ant color is
00168         red.
00169         """
00170         pygame.draw.rect(self.screen, color.dic["red"], pygame.Rect(self.x,
00171                                                     self.y,
00172                                                     self.taille,
00173                                                     self.taille))
00174 
00175     def __str__(self):
00176         """!@brief Ant string representation
00177         This method redefine the ant's toString() representation
00178         """
00179         return f"Fourmi a coord: {self.x}, {self.y}" \
00180             + f" rotation {self.direction}"
```

8.20 plateau.py File Reference

Classes

- class langton.plateau.Plateau

  Represent a Plateau.

Namespaces

- namespace langton

  Package corresponding to all the functions specific to Langton.
- namespace langton.plateau

  langton.plateau

8.21 plateau.py

Go to the documentation of this file.

```
00001 """!
00002 @brief langton.plateau
00003 @author Durel Enzo
00004 @author Mallepeyre Nourrane
00005 @version 1.0
00006 """
00007 import pygame
00008 import random
00009 import numpy as np
00010 import time
00011 from multiprocessing import Process, Manager
00012 from multiprocessing import cpu_count
00013 
00014 from utils import color
00015 
00016 from langton import Case
00017 
00018 class Plateau :
00019     """!@brief Represent a Plateau"""
00020 
00021     def __init__(self, behavior, color, taille=(1, 1), res=(4, 4)):
00022         """!@brief Construct Plateau object
00023         @param behavior String representation of the Plateau behavior
00024         @param color A tuple a int reprensent a rgb color
00025         @param taille tuple represent number of Case (default (1, 1))
00026         @param res Pixel representation of each Case (default (4, 4))
00027         """
```

```
00028         self.w, self.h = taille
00029         self.schema = [[Case(res, (j, i))
00030                         for j in range(self.w)]
00031                        for i in range(self.h)]
00032         # SCREEN #
00033         self.screen = pygame.display.get_surface()
00034         # BEHAVIOR #
00035         self.behavior = behavior
00036         self.color = color
00037         # MULTIPROCESSING #
00038         self.cpu = cpu_count()
00039         self.ratio = int(self.w/self.cpu)
00040         self.intervalles = [([i*self.ratio, 0],
00041                              [(i+1)*self.ratio, self.h])
00042                             for i in range(self.cpu-1)]
00043         self.intervalles.append(([self.ratio*(self.cpu-1), 0],
00044                                  [self.w, self.h]))
00045
00046     def multiprocessing(self, fun):
00047         """!@brief create a multiprocess
00048         @param fun function reference which call in process
00049         """
00050         process = [Process(target=fun,
00051                            args=self.intervalles[i])
00052                    for i in range(self.cpu)]
00053         for x in process: x.start()
00054         for x in process: x.join()
00055
00056     def draw_mp(self):
00057         """!@brief draw multiprocessing
00058         """
00059         self.multiprocessing(self.draw)
00060
00061     def reset(self, start=[0, 0], end=[None, None]):
00062         """!@brief reset schema in white
00063         @param start list of int index where start to draw (default (0, 0))
00064         @param end list of int index where end to draw (default (None, None))
00065         """
00066         start, end = self.valide_intervalle(start, end)
00067         for i in range(start[1], end[1]):
00068             for j in range(start[0], end[0]):
00069                 self.set_color_case((i, j), color.dic["white"])
00070         self.draw_mp()
00071
00072     def random(self, start=[0, 0], end=[None, None]):
00073         """!@brief random color schema
00074         @param start list of int index where start to draw (default (0, 0))
00075         @param end list of int index where end to draw (default (None, None))
00076         """
00077         start, end = self.valide_intervalle(start, end)
00078         for i in range(start[1], end[1]):
00079             for j in range(start[0], end[0]):
00080                 r = random.choice(self.color)
00081                 self.set_color_case((i, j), r)
00082         self.draw_mp()
00083
00084     def set_color_case(self, pos, colour):
00085         """!@brief set color in a Case
00086         @param pos tuple of position of the Case
00087         @param colour colour to set in the Case
00088         """
00089         x, y = pos
00090         self.schema[x][y].set_color(colour)
00091
00092     def draw_case(self, pos):
00093         """!@brief draw a case in specific schema position
00094         @param pos tuple of position of the Case
00095         """
00096         x, y = pos
00097         self.schema[x][y].draw()
00098
00099     def valide_intervalle(self, start, end):
00100         """!@brief valide start & end in schema
00101         @param start list of int index where start to draw (default (0, 0))
00102         @param end list of int index where end to draw (default (None, None))
00103         @return a valid couple of start & end (valid index)
00104         """
00105         if end[0] == None:      # if end_x not setting in draw()
00106             end[0] = self.w
00107         if end[1] == None:      # if end_y not setting  in draw()
00108             end[1] = self.h
00109         else :
00110             if start[0] < 0:
00111                 start[0] = start[0]%self.w
00112             if start[1] < 0:
00113                 start[1] = start[1]%self.h
00114             if end[0] > self.w:
```

```
00115                 end[0] = end[0]%self.w
00116             if end[1] > self.h:
00117                 end[1] = end[1]%self.h
00118         return start, end
00119
00120     def draw_ant(self, pos):
00121         """!@brief draw optimize cases near ant
00122         @param pos postion of the ant
00123         """
00124         y, x = pos
00125         self.draw_case((x, y))
00126         self.draw_case((x, (y+1)%self.w))
00127         self.draw_case((x, (y-1)%self.w))
00128         self.draw_case(((x+1)%self.h, y))
00129         self.draw_case(((x-1)%self.h, y))
00130
00131     def draw(self, start=[0,0], end=[None, None]):
00132         """!@brief default color schema
00133         @param start list of int index where start to draw (default (0, 0))
00134         @param end list of int index where end to draw (default (None, None))
00135         """
00136         start, end = self.valide_intervalle(start, end)
00137         for i in range(start[1], end[1]):
00138             for j in range(start[0], end[0]):
00139                 # self.draw_case((i, j))
00140                 self.schema[i][j].draw()
00141
00142     def get_case(self, x, y, res):
00143         """!@brief Get a Case in a position
00144         @param x Horizontal position of the Case
00145         @param y Vertical position of the Case
00146         @param res Resolution of the Case
00147         """
00148         case = self.schema[y//res][x//res]
00149         return case
00150
00151     def set_behavior(self, color, behavior):
00152         """!@brief Modify grid color & behavior
00153         @param color list of color
00154         @param behavior string of the simulation behavior
00155         """
00156         self.color = color
00157         self.behavior = behavior
00158
00159     def __str__(self):
00160         """!@brief Redefine toString() Plateau method"""
00161         s = ""
00162         for i in self.schema:
00163             for j in i:
00164                 s += f"{j.cur_color} "
00165             s += f"\n"
00166         return s
```

8.22 simulation.py File Reference

Classes

- class langton.simulation.Simulation

  Represent a Simulation.

Namespaces

- namespace langton

  Package corresponding to all the functions specific to Langton.
- namespace langton.simulation

  Simulation package.

8.23 simulation.py

Go to the documentation of this file.

```
00001 """!
00002 @brief Simulation package
00003 @author Durel Enzo
00004 @author Mallepeyre Nourrane
00005 @version 1.0
00006 """
00007
00008 import pygame
00009 import time
00010 import random
00011
00012 import langton as lgt
00013 from buttons import Button, Menu, CheckBox, InputBox
00014 from utils import color, const
00015
00016 class Simulation:
00017     """!@brief Represent a Simulation"""
00018
00019     def __init__(self,
00020                  size_screen=const.DEFAULT_SCREEN_SIZE,
00021                  size_plateau=const.DEFAULT_PLATEAU_SIZE,
00022                  res=const.DEFAULT_RESOLUTION):
00023         """!@brief Construct Simulation object
00024         @param size_screen Size of the window
00025         @param size_plateau Size of the grid where stand ants
00026         @param res Number of pixel define size of Case and Fourmi
00027         """
00028         self.size_screen = size_screen # size of the screen
00029         self.size_plateau = size_plateau # size of the plateau
00030         self.res = res # resolution of a case / fourmi
00031
00032         # PYGAME #
00033         self.screen = pygame.display.set_mode(size_screen, pygame.DOUBLEBUF)
00034         pygame.display.set_caption("Fourmi de Langton") # title set
00035         self.clock = pygame.time.Clock() # clock to control the framerate
00036
00037         # STATES #
00038         self.end = False # if simulation end
00039         self.run = False # simulation loop
00040
00041         # DEBUG #
00042         self.debug = False
00043         self.iteration = 0
00044
00045         # BEHAVIOR #
00046         self.behavior = ""
00047         self.color = []
00048
00049         # PLATEAU INIT #
00050         self.plateau = lgt.Plateau(self.behavior,
00051                                    self.color,
00052                                    taille=(self.size_plateau[0]//self.res,
00053                                            self.size_plateau[1]//self.res),
00054                                    res=(self.res, self.res))
00055
00056         self.set_behavior("LR")
00057
00058         # Fourmi(s) Init #
00059         self.nb_fourmi = 0
00060         self.fourmi_list = []
00061
00062         # Button #
00063         self.btn_debug = Button((self.size_screen[0]-260, 20),
00064                                 const.BUTTON_SIZE,
00065                                 text="Debug",
00066                                 fun=self.active_debug)
00067         self.btn_play = Button((self.size_screen[0]-260,
00068                                 self.size_plateau[1]-70),
00069                                const.BUTTON_SIZE,
00070                                text="Play",
00071                                fun=self.play)
00072         self.btn_stop = Button((self.size_screen[0]-120,
00073                                 self.size_plateau[1]-70),
00074                                const.BUTTON_SIZE,
00075                                text="Stop",
00076                                fun=self.stop)
00077         self.btn_reset = Button((self.size_screen[0]-260, 90),
00078                                 const.BUTTON_SIZE,
00079                                 text="Reset",
00080                                 fun=self.reset)
00081         self.btn_next = Button((self.size_screen[0]-260,
00082                                 self.size_plateau[1]-140),
00083                                 const.BUTTON_SIZE,
```

```
00084                                 text="Next",
00085                                 fun=self.next_step)
00086         self.btn_add_f = Button((self.size_screen[0]-260,
00087                                  self.size_plateau[1]-210),
00088                                 const.BUTTON_SIZE,
00089                                 text="Add",
00090                                 fun=self.add_fourmi)
00091 
00092         # InputBox #
00093         self.ib_next = InputBox((self.size_screen[0]-120,
00094                                  self.size_plateau[1]-140),
00095                                 const.BUTTON_SIZE,
00096                                 fun=self.set_next)
00097         self.ib_behavior = InputBox((self.size_screen[0]-260, 160),
00098                                     (240, 50),
00099                                     fun=self.set_behavior,
00100                                     max_len=10)
00101 
00102         # CheckBox #
00103         self.cb_infinite = CheckBox((self.size_screen[0]-260, 230),
00104                                     (20, 20),
00105                                     text="Infinite",
00106                                     fun=self.infinite)
00107         self.cb_random_grid = CheckBox((self.size_screen[0]-260, 270),
00108                                        (20, 20),
00109                                        text="Random Grid",
00110                                        fun=self.set_random_grid)
00111         self.infinite_ant = False
00112         self.random_grid = False
00113 
00114         # Menu #
00115         self.menu = Menu((size_plateau[0], 0),
00116                          (size_screen[0]-size_plateau[0], size_screen[1]),
00117                          btn_list=[self.btn_debug, self.btn_play,
00118                                    self.btn_stop, self.btn_reset,
00119                                    self.btn_next, self.btn_add_f,
00120                                    self.ib_next, self.ib_behavior,
00121                                    self.cb_infinite, self.cb_random_grid])
00122 
00123         # Simulation #
00124         self.it = self.__iter__()
00125         self.next_time = 1
00126 
00127     def start(self):
00128         """!@brief Global start, here when ants aren't running"""
00129 
00130         self.plateau.reset()
00131         self.menu.draw(self.screen)
00132 
00133         # self.plateau.compare_fun(self.plateau.draw, self.plateau.draw_mp)
00134 
00135         while self.startstart:
00136 
00137             self.menu.enable() # enable all buttons in the menu
00138 
00139             if self.nb_fourmi <= 0:
00140                 self.menu.disable(self.btn_play,
00141                                   self.btn_next,
00142                                   self.ib_next,
00143                                   self.btn_stop,
00144                                   self.btn_debug)
00145 
00146             if self.nb_fourmi > 0:
00147                 self.menu.disable(self.ib_behavior)
00148 
00149             self.handle_event()
00150             pygame.display.update() # update the screen
00151             self.clock.tick(30) # control the max framerate
00152 
00153     def stop(self):
00154         """!@brief Stop the game by set self.run to false"""
00155         self.run = False
00156 
00157     def add_fourmi(self):
00158         """!@brief Add an ant on the grid"""
00159         if not self.run :
00160             not_click = True
00161             while not_click:
00162                 self.clock.tick(30) # control the max framerate
00163                 for event in pygame.event.get():
00164                     if event.type == pygame.MOUSEBUTTONDOWN:
00165                         not_click = False
00166                         if event.pos[0] >= 0 \
00167                            and event.pos[0] < self.size_plateau[0] \
00168                            and event.pos[1] >= 0 \
00169                            and event.pos[1] < self.size_plateau[1] :
00170                             e_x, e_y = event.pos
```

```
00171                             f_x = (e_x//self.res)*self.res
00172                             f_y = (e_y//self.res)*self.res
00173                             f_new = lgt.Fourmi(coords=(f_x, f_y),
00174                                                taille=self.res,
00175                                                speed=self.res,
00176                                                direction=0,
00177                                                behavior=self.behavior,
00178                                                color=self.color)
00179                             self.fourmi_list.append(f_new)
00180                             self.nb_fourmi += 1
00181                             f_new.draw()
00182                         else:
00183                             print("Can't create an ant here.")
00184
00185         else:
00186             print("Can't add an ant when simulation is running..")
00187
00188     def reset(self):
00189         """!@brief Reset the simulation"""
00190         if not self.run :
00191             if self.random_grid:
00192                 self.plateau.random()
00193             else :
00194                 self.plateau.reset()
00195
00196             self.fourmi_list = [].copy()
00197             self.nb_fourmi = 0
00198
00199             self.iteration = 0
00200             self.end = False
00201             self.menu.draw(self.screen)
00202         else :
00203             print("Can't reset when simulation is running")
00204
00205     def next_step(self):
00206         """!@brief Play simulation iterator next_number times, (default 1)
00207         @exception StopIteration If Simulation ended
00208         @exception Exception If Simulation already run
00209         """
00210         try:
00211             if not self.end and not self.run:
00212                 for _  in range(self.next_time):
00213                     next(self.it)
00214         except StopIteration:
00215             print("Simulation ended")
00216         except Exception:
00217             print("Already in function")
00218
00219     def play(self):
00220         """!@brief Play Simulation loop"""
00221         try:
00222             if not self.run:
00223                 if not self.end:
00224                     self.run = True
00225                 while self.run:
00226                     next(self.it)
00227                 if self.end :
00228                     print("Simulation ended.")
00229         except Exception:
00230             raise(Exception)
00231             print("Already playing")
00232
00233     def __iter__(self):
00234         """!@brief Simulation iterator"""
00235
00236         while True:
00237
00238             self.iteration += 1
00239             self.handle_event()
00240
00241             self.fourmi_step()
00242             self.fourmi_out()
00243             self.draw()
00244
00245             pygame.display.update() # update the screen
00246             # self.clock.tick(60) # control the max framerate
00247             yield;
00248
00249     def draw(self):
00250         """!@brief Draw the map and ants"""
00251         for f in self.fourmi_list:
00252             if not self.end:
00253                 self.plateau.draw_ant((f.x//self.res, f.y//self.res))
00254                 f.draw()
00255
00256     def fourmi_out(self):
00257         """!@brief Check if an ant is out of index of the grid"""
```

```
00258         for f in self.fourmi_list:
00259             if f.x == self.size_plateau[0] \
00260                or f.y == self.size_plateau[1] \
00261                or f.x < 0 or f.y < 0:
00262                 if self.infinite_ant:
00263                     f.x = f.x%self.size_plateau[0]
00264                     f.y = f.y%self.size_plateau[1]
00265                 else:
00266                     self.run = False
00267                     self.end = True
00268 
00269     def fourmi_step(self):
00270         """!@brief Move all ants once"""
00271         for f in self.fourmi_list:
00272             case = self.plateau.get_case(f.x, f.y, self.res)
00273             f.one_step(case)
00274 
00275     def handle_event(self):
00276         """!@brief Event listener user interactions: Button & Keyboard"""
00277 
00278         for event in pygame.event.get():
00279             self.menu.handle_event(event,
00280                                    self.btn_debug,
00281                                    self.btn_play,
00282                                    self.btn_stop,
00283                                    self.cb_infinite,
00284                                    self.cb_random_grid)
00285 
00286             if not self.run:
00287                 self.menu.handle_event(event,
00288                                        self.btn_next,
00289                                        self.ib_next,
00290                                        self.btn_reset,
00291                                        self.btn_add_f,
00292                                        self.ib_behavior)
00293 
00294             else:
00295                 self.menu.disable(self.btn_next,
00296                                   self.ib_next,
00297                                   self.btn_reset,
00298                                   self.btn_add_f)
00299 
00300             if event.type == pygame.QUIT:
00301                 self.run = False
00302                 self.startstart = False
00303 
00304         self.menu.draw(self.screen)
00305 
00306         if self.debug:
00307             self.debuging()
00308 
00309     def init_color(self):
00310         """!@brief Init random color from simulation behavior"""
00311         color_list = []
00312         color_tuple = []
00313         color_list.append(color.dic["white"])
00314         for _ in range(len(self.behavior)-1):
00315             for _ in range(3):
00316                 color_tuple.append(random.randint(0, 255))
00317             color_list.append(tuple(color_tuple.copy()))
00318             color_tuple = [].copy()
00319         self.color = color_list.copy()
00320 
00321     def active_debug(self):
00322         """!@brief Change boolean value for debug option button"""
00323         self.debug = not self.debug
00324 
00325     def debuging(self):
00326         """!@brief Print the total of iteration since the simulation is
00327         running"""
00328         # print(f"Iteration : {self.iteration}")
00329         txt_surf = self.btn_debug.font.render(f"{self.iteration}",
00330                                               True,
00331                                               color.dic["white"])
00332         pos = list(self.btn_debug.get_pos())
00333         size = list(self.btn_debug.get_size())
00334 
00335         text_w = txt_surf.get_width()
00336         text_h = txt_surf.get_height()
00337 
00338         pos = (self.size_screen[0]-70-text_w//2,
00339                20 + size[1]//2 - text_h//2),
00340 
00341         self.screen.blit(txt_surf, pos)
00342 
00343     def set_next(self, text):
00344         """!@brief Set the step for next function"""
```

```
00345         try:
00346             self.next_time = int(text)
00347             print(f"Set next step to {self.next_time}")
00348         except Exception:
00349             print("Invalide next value.")
00350 
00351     def set_behavior(self, text):
00352         """!@brief Set the behavior before add ants and game start"""
00353         for letter in text:
00354             if letter != "R" and letter != "L":
00355                 print("Invalide behavior, must be a text of 'R' and 'L'.")
00356                 return;
00357         self.behavior = text
00358         self.init_color()
00359         self.plateau.set_behavior(self.color, self.behavior)
00360         print(f"Set simulation behavior to {self.behavior}")
00361 
00362     def infinite(self):
00363         """!@brief Define if ants are in an infinite place"""
00364         self.infinite_ant = not self.infinite_ant
00365         if self.infinite_ant and self.end :
00366             self.end = False
00367             for f in self.fourmi_list:
00368                 f.x %= self.size_plateau[0]
00369                 f.y %= self.size_plateau[1]
00370         print(f"Set infinite board to {self.infinite_ant}")
00371 
00372     def set_random_grid(self):
00373         """!@brief Random Grid CheckBox function"""
00374         self.random_grid = not self.random_grid
00375         print(f"Set random grid to {self.random_grid}")
```

8.24 main.py File Reference

Namespaces

- namespace main

  first program to be execute

Variables

- main.simulation = Simulation(res=4)

8.25 main.py

Go to the documentation of this file.

```
00001 """!
00002 @brief first program to be execute
00003 @author Durel Enzo
00004 @author Mallepeyre Nourrane
00005 @version 1.0
00006 """
00007 
00008 import pygame
00009 
00010 from langton import Simulation
00011 
00012 if __name__ == '__main__' :
00013 
00014     pygame.init()
00015 
00016     simulation = Simulation(res=4)
00017 
00018     simulation.start()
00019 
00020     pygame.quit()
```

8.26 color.py File Reference

Namespaces

- namespace utils

  Package corresponding to all the const need for the program.
- namespace utils.color

  Package of all colors used in the program.

Variables

- utils.color.dic = dict()
- tuple utils.color.INACTIVE_BUTTON_COLOR = (46, 107, 81)
- tuple utils.color.ACTIVE_BUTTON_COLOR = (69, 153, 125)
- tuple utils.color.HOVER_BUTTON_COLOR = (49, 122, 110)
- tuple utils.color.DISABLE_BUTTON_COLOR = (55, 64, 60)
- utils.color.TEXT_BUTTON_COLOR = dic["white"]
- tuple utils.color.INACTIVE_IB_COLOR = (46, 107, 81)
- tuple utils.color.ACTIVE_IB_COLOR = (186, 191, 119)
- tuple utils.color.DISABLE_IB_COLOR = (55, 64, 60)
- utils.color.TEXT_IB_COLOR = dic["black"]
- tuple utils.color.INACTIVE_CB_COLOR = (46, 107, 81)
- tuple utils.color.ACTIVE_CB_COLOR = (186, 191, 119)
- tuple utils.color.DISABLE_CB_COLOR = (55, 64, 60)
- tuple utils.color.DISABLE_ACTIVE_CB_COLOR = (87, 56, 53)
- utils.color.TEXT_CB_COLOR = dic["black"]
- tuple utils.color.MENU_COLOR = (103, 168, 120)

8.27 color.py

Go to the documentation of this file.

```
00001 """!
00002 @brief Package of all colors used in the program
00003 @author Durel Enzo
00004 @author Mallepeyre Nourrane
00005 @version 1.0
00006 """
00007
00008 # color dictionnary (basic color) #
00009 dic = dict()
00010 dic["white"] = (255, 255, 255)
00011 dic["black"] = (0, 0, 0)
00012 dic["red"] = (255, 0, 0)
00013 dic["green"] = (0, 255, 0)
00014
00015 # Button special colors #
00016 INACTIVE_BUTTON_COLOR = (46, 107, 81)
00017 ACTIVE_BUTTON_COLOR = (69, 153, 125)
00018 HOVER_BUTTON_COLOR = (49, 122, 110)
00019 DISABLE_BUTTON_COLOR = (55, 64, 60)
00020
00021 TEXT_BUTTON_COLOR = dic["white"]
00022
00023 # Input Box special colors #
00024 INACTIVE_IB_COLOR = (46, 107, 81)
00025 ACTIVE_IB_COLOR = (186, 191, 119)
00026 DISABLE_IB_COLOR = (55, 64, 60)
00027
00028 TEXT_IB_COLOR = dic["black"]
00029
00030 # Check Box special colors #
00031 INACTIVE_CB_COLOR = (46, 107, 81)
00032 ACTIVE_CB_COLOR = (186, 191, 119)
00033 DISABLE_CB_COLOR = (55, 64, 60)
00034 DISABLE_ACTIVE_CB_COLOR = (87, 56, 53)
00035
00036 TEXT_CB_COLOR = dic["black"]
00037
00038 # Menu colors #
00039 MENU_COLOR = (103, 168, 120)
```

8.28 const.py File Reference

Namespaces

- namespace utils

  Package corresponding to all the const need for the program.
- namespace utils.const

  Package of all constants used in the program.

Variables

- tuple utils.const.DEFAULT_SCREEN_SIZE = (1280, 720)
- tuple utils.const.DEFAULT_PLATEAU_SIZE = (1000, 720)
- int utils.const.DEFAULT_RESOLUTION = 4
- tuple utils.const.BUTTON_SIZE = (100, 50)

8.29 const.py

Go to the documentation of this file.

```
00001 """!
00002 @brief Package of all constants used in the program
00003 @author Durel Enzo
00004 @author Mallepeyre Nourrane
00005 @version 1.0
00006 """
00007
00008 # Game constantes #
00009 DEFAULT_SCREEN_SIZE = (1280, 720)
00010 DEFAULT_PLATEAU_SIZE = (1000, 720)
00011 DEFAULT_RESOLUTION = 4
00012
00013 # Button constantes #
00014 BUTTON_SIZE = (100, 50)
```

Index