TIPE Documentation

Version 1

Eric

mars 14, 2021

Contents:

CHAPITRE 1

Package *road_objects*

1.1 Module graphical_item

1.1.1 Module *graphical_item.py*

Created on 21/02/2021 *by* Eric Ollivier

Versioning : - 0.1 : Initial version

class road_objects.graphical_item.**GraphicalItem**(*axe : matplotlib.axes._axes.Axes*)

Représentation graphique des objets de la route

Paramètres **axe** – objet de type matplotlib.Axes (contexe graphique)

__class__
alias de type

__delattr__
Implement delattr(self, name).

__dict__ = mappingproxy({'__module__': 'road_objects.graphical_item', '__doc__': '\n

__dir__() → list
default dir() implementation

__eq__
Return self==value.

__format__()
default object formatter

__ge__
Return self>=value.

__getattribute__
Return getattr(self, name).

__getitem__(*name*)

__gt__
Return self>value.

__hash__
Return hash(self).

__init__(*axe : matplotlib.axes._axes.Axes*)

Paramètres **axe** – objet de type matplotlib.Axes (contexe graphique)

__init_subclass__()
This method is called when a class is subclassed.
The default implementation does nothing. It may be overridden to extend subclasses.

__le__
Return self<=value.

__lt__
Return self<value.

__module__ = 'road_objects.graphical_item'

__ne__
Return self!=value.

__new__()
Create and return a new object. See help(type) for accurate signature.

__reduce__()
helper for pickle

__reduce_ex__()
helper for pickle

__repr__
Return repr(self).

__setattr__
Implement setattr(self, name, value).

__sizeof__() → int
size of object in memory, in bytes

__str__
Return str(self).

__subclasshook__()
Abstract classes can override this to customize issubclass().
This is invoked early on by abc.ABCMeta.__subclasscheck__(). It should return True, False or NotImplemented. If it returns NotImplemented, the normal algorithm is used. Otherwise, it overrides the normal algorithm (and the outcome is cached).

__weakref__
list of weak references to the object (if defined)

add_component(*name : str*, *component*)
Ajout un élement graphique de base pour la représentation de l'Item

Paramètres
— **name** – Nom du composant
— **component** – Composant à ajouter de type matplotlib.Artist

add_plot(**args*, *name : str*, ***kwargs*)
Crée un objet grapghique avec la méthode *Axes.plot* et l'ajoute à la liste des composants graphiques

Paramètres
— **name** – Nom à donner au nouveau composant
— **args** – Liste des paramètre pour la méthode *Axes.plot*
— **kwargs** – Liste des paramètres nommés pour la méthode *Axes.plot*

ax

get_components() → list
Fournit la liste de composants graphique

Retourne objet *list* d'objets de type matplotlib.Artist

1.2 Module road_item

1.2.1 Module name : *road_item.py*

Created on 14/02/2021 *by* Eric Ollivier

Versionning :

— 0.1 : Initial version
— 0.2 ; Ajout de la notion de franchissable (attribut “passable”)

class road_objects.road_item.**RoadItem**(*axe*, *path=None*, *roads=None*, *name=None*, ***kwargs*)

Classe de base pour les objets de la route :
— Véhicule
— signalisation (ex : feux tricolores)
— obstable (à venir)
— ...
Contribue à calculer le changement de vitesse des véhicules

Paramètres
— **axe** – Context graphique (objet matplotlib.Axes)
— **path** – Itinéraire de l'objet *RoadItem* (objet de type *roadmaps.Path*)
— **roads** – Liste des routes si *path* n'est pas défini(objet *list* d'objet *roadmap.Road*)
— **name** – Nom de l'objet *RoadItem*
— **current_time** – Valeur initial du temps courant
— **init_time** – Valeur du temps de début de scénario
— **index** – Valeur initial de la position dans l'itinéraire
— **passable** – Valeur initial de propriété de franchissement

classmethod Add_item(*item*)
Ajoute un objet à l'inventaire des items de la route

classmethod Get_Items()
Retourne la liste des items

__eq__(*other*)
Définit l'égalité pour la comparaison entre les objets

__init__(*axe*, *path=None*, *roads=None*, *name=None*, ***kwargs*)

Paramètres
— **axe** – Context graphique (objet matplotlib.Axes)
— **path** – Itinéraire de l'objet *RoadItem* (objet de type *roadmaps.Path*)
— **roads** – Liste des routes si *path* n'est pas défini(objet *list* d'objet *roadmap.Road*)
— **name** – Nom de l'objet *RoadItem*
— **current_time** – Valeur initial du temps courant
— **init_time** – Valeur du temps de début de scénario
— **index** – Valeur initial de la position dans l'itinéraire
— **passable** – Valeur initial de propriété de franchissement

__le__(*other*)
Test si self a une position avant un autre objet dans le parcours de *self*.

Retourne
— True : si *other* est sur une position à venir de l'itinéraire de *self*
— **False** [si]
— *other* n'a pas de postion commune avec *self*
— *other* n'a pas démarré
— *self* n'a pas démarré

add_road(**road*)
Ajoute une ou plusieurs route(s) à l'itinairaire

delta_time
Durée entre deux mises à jour de *current_time*

distance(*position : roadmaps.position.Position*) → float
Calcul la distance entre <self> et l'objet <item>

Pré-requis de construction : la distance entre chaque position est constante et vaut parameters.DISTANCE_POSITION distance : := (nb positions entre self et position) * DISTANCE_POSITION

$distance = \sum_{self}^{position} nb\ positions.(parameters.DISTANCE\_POSITION)$

ancien mode de calcul : distance : := SUM(DISTANCE(postion_i, position_i+1)), avec position_i in[self.position, item.position [La fonction de calcul de distance entre 2 positions est défini par le fonction DISTANCE

Retourne
— distance(*self*, *other*) si *other* a au moins une position commune avec *self*
— *None* si pas de position commune

forward(*new_time*)
Methode par pdéfaut pour faire avancer un objet dans le temps

get_plot(*new_time=None*)
Retourne les éléménts graphique à afficher

get_position(*new_time*)
Retourne la position (x,y) du Vehicule

init_graphic()
Initialise la représentation graphique

is_ended
Retourne *True* si l'objet *RoadItem* a fini son chemin sinon'False'

is_started
Retourne *True* si l'objet *RoadItem* a débuté son chemin sinon *False*

length
Nombre de *Position* sur l'itinéraire

next_position
Position estimée de la prochaine frame

passable
Retour le statut de franchissement

path
Itinéraire de l'objet *RoadItem* (objet de type'Path')

position
Position courante de l'objet *RoadItem* Retourne *None* s'il n'est pas sur le chemin (pas actif)

remain_path(*end_index=None*)
Retourne la portion renatnt à parcourrir pour l'objet *RoadItem*

Paramètres **end_index** – Index correspondant à la borne max

Retourne
— Objet *list* contenant les objets *Position* à venir s'il en reste.
— Objet *list* vide s'il n'y a plus rien à parcourrir

set_impassable()
Rend RoadItem infranchissable

set_passable(*mode=True*)
Change le statut de franchissement : - True : rend RoadItem franchissable (par défaut) - False : rend RoadItem infranchissable

start(*init_time*)
Définit le moment du départ : permet un décalage dans la lecture des positions

CHAPITRE 2

Indices and tables

— genindex
— modindex
— search

2.1 Package *road_objects*

Ce module contient la définition de la classe Vehicule
— Vehicule.index : cast la valeur en “int”

class road_objects.vehicule.**Vehicule**(*axe*, *path=None*, *name=None*)

Regroupe les routes de son itinairaire et définit la position du véhicule dans le temps

classmethod List_vehicules()

Retourne la liste des véhicules

get_plot(*new_time=None*)

Retourne les éléménts graphique à afficher

update_speed(*distance : float = None*) → None

Calcul et met à jour la vitesse en fonction de la distance de l’objet routier suivant :param distance : distance entre <self> et l’objet suivant

2.2 Package *roadmaps*

Project name : TIPE_Yoann Module name : map.py

Classes list in this module : - Map : Classe de base pour les cartes ——— Author : Eric Ollivier Create date : 20/02/2021 ———
Versionning : 0.1 : Initial version

Project name : TIPE_Yoann Module name : path.py

Classes list in this module : - Path ———
Author : Eric Ollivier Create date : 14/02/2021 ———
Versionning : 0.1 : Initial version 0.2 : - Ajout de la méthode “append_position” permettant d’ajouter une position à un chemin

Ce module contient les routes permettant au véhicule de se déplacer - Class road - variable roadmap

class roadmaps.road.**Road**(*x_interval*, *y_interval*, *path_functions*, *step : int = None*)
Définit la fonction de calcul de la trajectoire entre deux positions.

Paramètres
- **x_interval** – Intervalle de calcul (x_start, x_end)
- **y_interval** – Intervalle de calcul (y_start, y_end)
- **path_functions** – Fonctions paramétriques (x_func, y_func) de calcul de la trajectoire entre les deux positions

init_step()
Calcul le nombre de pas pour avoir une distance de DISTANCE_POSTION

setup_path()
Calcule les coordonnées de la route

Project name : TIPE_Yoann Module name : position.py

Classes list in this module : - Position
Author : Eric Ollivier Create date : 14/02/2021
Versionning : 0.1 : Initial version

class roadmaps.position.**Position**(*x*, *y*)
Classe définissant une position pour un objet RoadItem

Carte d'un rond-point

Index des modules Python

r

Index