Vector2D

```
from soccersimulator import Vector2D
v = Vector2D(x=1.,y=1.)  # creation a partir de coordonnees cartesiennes
v_bis = Vector2D(angle=3.14, norm = 2) # creation a partir de coordonnees polaires
v + v_bis, v - v_bis     # operation usuelle
v += v_bis               # pareil
print v.x, v.y, v.angle, v.norm # accesseur et modifieur
v.random(low = 0, high =1.) # vecteur aleatoire entre low et high (0 et 1) par defaut
v.distance(v_bis) # distance entre deux points
v.dot(v_bis)      #produit scalaire
v.normalisze()    #normalise le vecteur (norme = 1)
v.scale(3.)       # multiplie les coordonnees du vecteur
v_ter = v.norm_max(4) # renvoie le vecteur mis a l'echelle tel que sa norme soit inferieur
ou egale au parametre
v_quar = v.copy() # copie le vecteur
Vector2D.create_random(low=0.,high=1.) # cree un vecteur aleatoire
```

SoccerAction

```
from soccersimulator import Vector2D, SoccerAction
action = SoccerAction(acceleration = Vector2D(), shoot = Vector2D()) # creation
action_bis = action.copy()  # copier
action_ter = action + action_bis # additionne les vecteurs acceleration et shoot
print action_ter.acceleration, action_ter.shoot # acces aux vecteurs d'info
print action_ter.acceleration.x, action_ter.acceleration.y
print action_ter.acceleration.norm, action_ter.acceleration.angle
```

SoccerState

```
state.ball.position, state.ball.vitesse # recuperer la balle
p = state.player_state(id_team, id_player) # position et vitesse du joueur
p.position, p.vitesse
state.players # liste des (id_team,id_player) des joueurs du match
state.nb_player(id_team) # nombre de joueurs de l'equipe
state.score_team1, state.score_team2, state.get_score_team(id_team) # score des equipes
state.winning_team # 0 si pas de but a ce tour, 1 si equipe 1, 2 si equipe 2
state.step # numero du tour du jeu
```

settings : info sur le jeu en général

```
from soccersimulator import settings
settings.GAME_WIDTH = 150  # largeur du terrain
settings.GAME_HEIGHT = 90  # hauteur du terrain
settings.GAME_GOAL_HEIGHT = 10  # largeur des buts
settings.PLAYER_RADIUS = 1.     #rayon d'un joueur
settings.BALL_RADIUS = 0.65     # rayon de la balle
```

Créer une stratégie

```
from soccersimulator import AbstractStrategy, Vector2D

class MaStrategy(AbstractStrategy):
    def __init__(self):
        AbstractStrategy.__init__(self, "Random")
        # rajouter l'initialisation voulue

    def compute_strategy(self, state, id_team,  id_player):
        #### rajouter du code, renvoyer un SoccerAction
        return SoccerAction()
```

Lancer un match

```
import soccersimulator
from soccersimulator import SoccerTeam, SoccerMatch
from soccersimulator import Player, SoccerTournament

team1 = SoccerTeam("team1",[Player("t1j1",MaStrategy())])
team2 = SoccerTeam("team2",[Player("t2j1",MaStrategy())])
team3 = SoccerTeam("team3", [Player("t3j1", MaStrategy())])
match = SoccerMatch(team1, team2)
match.play()
tournoi = SoccerTournament(1) #nbre de joueurs par team
tournoi.add_team(team1)
tournoi.add_team(team2)
tournoi.add_team(team3)
soccersimulator.show(tournoi)
```