test_modele_dynamique

Laurent Perrinet (INT, UMR7289)

February 12, 2014

In [1]:

```
import numpy as np
import pylab
%matplotlib inline
```

```
Populating the interactive namespace from numpy and matplotlib
DEBUG parametres , position croix:  [ 10.90999985   6.23999977   1.37
]
32
```

In [ ]:

```
from parametres import sliders, VPs, volume, p, kinects_network_config, d_x, d_y, d_z,
from modele_dynamique import Scenario
events = [0, 0, 0, 0, 0, 0, 0, 0] # 8 types d'événements
print p['N']
```

In [2]:

```
def simulpos(s, t, players):
    positions = []
    for player in players:
        positions.append([player['center'][0] + player['amp'][0]*cos(2*pi*t/player['T'
                          player['center'][1] + player['amp'][1]*cos(2*pi*t/player['T'
                          player['center'][2] + player['amp'][2]*cos(2*pi*t/player['T'
    return positions
```

In [ ]:

```
def simul(p, players, dt=.01, t_stop=60., s_VP = 0, display=True):
    time = np.arange(0., t_stop, dt)
    N_time = len(time)
    n_players = len(players)
    positions_ = np.zeros((3, n_players, N_time))
    particles = np.zeros((6, p['N'], N_time))
    s = Scenario(p['N'], scenario, volume, [VPs[0]], p, calibration)
    for i_t, t in enumerate(time):
        positions = simulpos(s, t, players)
        positions_[:, :, i_t] = np.array(positions).T
        s.do_scenario(positions=positions, events=events, dt=dt)
        particles[:, :, i_t] = s.particles[0:6, s_VP*s.N:(s_VP+1)*s.N]
    if display:
        fig = figure(figsize=(18,10))
        T_step = N_time / 100
        for i_ax, axe in zip(range(3), ['x', 'y', 'z']):
            ax = fig.add_subplot(3, 1, 1+ i_ax)
            ax.plot(time, positions_[i_ax, :, :].T)
            ax.plot(time[::T_step], particles[i_ax, :, ::T_step].T, alpha=.5)
            #ax.errorbar(time[::T_step], particles[i_ax, :, ::T_step].mean(axis=0), pa
            #ax.errorbar(time[::T_step], particles[i_ax + 3, :, ::T_step].mean(axis=0)
            ax.set_ylabel(axe)
        ax.set_xlabel('time')
    return positions_, particles
```

1 stabilité avec paramètres

In [ ]:

```
roger = [10., 6, 1.5]
players = [{'center': roger, 'amp': [.3,1.,.1] , 'T': 5.}]
print p['damp']
pos, particles = simul(p, players)
```

In [ ]:

```
p_ = p.copy()
p_['damp'] = .0
print p_['damp']
pos, particles = simul(p_, players)
```

In [ ]:

```
p_ = p.copy()
p_['G_repulsion'] = 0.
pos, particles = simul(p_, players)
```

2 players instables

On définit un player seul qui se déplace devant le videoprojecteur, à une distance

In [ ]:

```
print VPs[0]
```

In [ ]:

```
for distance in linspace(4., 0., 8, endpoint=False):
    print 'Distance du player au VP = ', distance
    players = [{'center': [VPs[0]['x'] + distance, VPs[0]['y'], VPs[0]['z']], 'amp': [
    pos, particles = simul(p, players)
    show()
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

3 multi players

In [ ]:

```
players = [{'center': [10., 6, 1.5], 'amp': [.3,1.,.1] , 'T': 5.}, {'center': [10., 9,
pos, particles = simul(p, players)
```

étude de la stabilité avec une autre personne

In [ ]:

```
for distance in linspace(4., 0., 8, endpoint=False):
    players = [{'center': [10., 6, 1.5], 'amp': [.3,1.,.1] , 'T': 15.}]
    print 'Distance du player au VP = ', distance
    players.append({'center': [VPs[0]['x'] + distance, VPs[0]['y'], VPs[0]['z']], 'amp
    pos, particles = simul(p, players)
    show()
```

étude de la stabilité avec quatre autres personnes

In [ ]:

```
for distance in linspace(4., 0., 8, endpoint=False):
    players = [{'center': [10., 6, 1.5], 'amp': [1.3,1.,.1] , 'T': 15.},
               {'center': [10., 9, 1.5], 'amp': [2.3,1.,.1] , 'T': 3.},
               {'center': [5., 3, 1.9], 'amp': [4.3,1.,.1] , 'T': 15.},
               {'center': [15., 9, 1.], 'amp': [5.3,1.,.1] , 'T': 3.}]
    print 'Distance du player au VP = ', distance
    players.append({'center': [VPs[0]['x'] + distance, VPs[0]['y'], VPs[0]['z']], 'amp
    pos, particles = simul(p, players)
    show()
```

In [ ]:

```
```