test_modele_dynamique

Laurent Perrinet (INT, UMR7289)

February 14, 2014

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
from parametres import sliders, VPs, volume, p, kinects_network_config, d_x, d_y, d_z,
from modele_dynamique import Scenario, xyz2azel
events = [0, 0, 0, 0, 0, 0, 0, 0] # 8 types d'événements
print p['N']
```

```
DEBUG parametres , position croix:  [ 0.          3.76999998  1.37
]
32
```

In [3]:
```
def simulpos(s, t, players):
    positions = []
    for player in players:
        positions.append([player['center'][0] + player['amp'][0]*np.cos(2*np.pi*t/play
                          player['center'][1] + player['amp'][1]*np.cos(2*np.pi*t/play
                          player['center'][2] + player['amp'][2]*np.cos(2*np.pi*t/play
    return positions
```

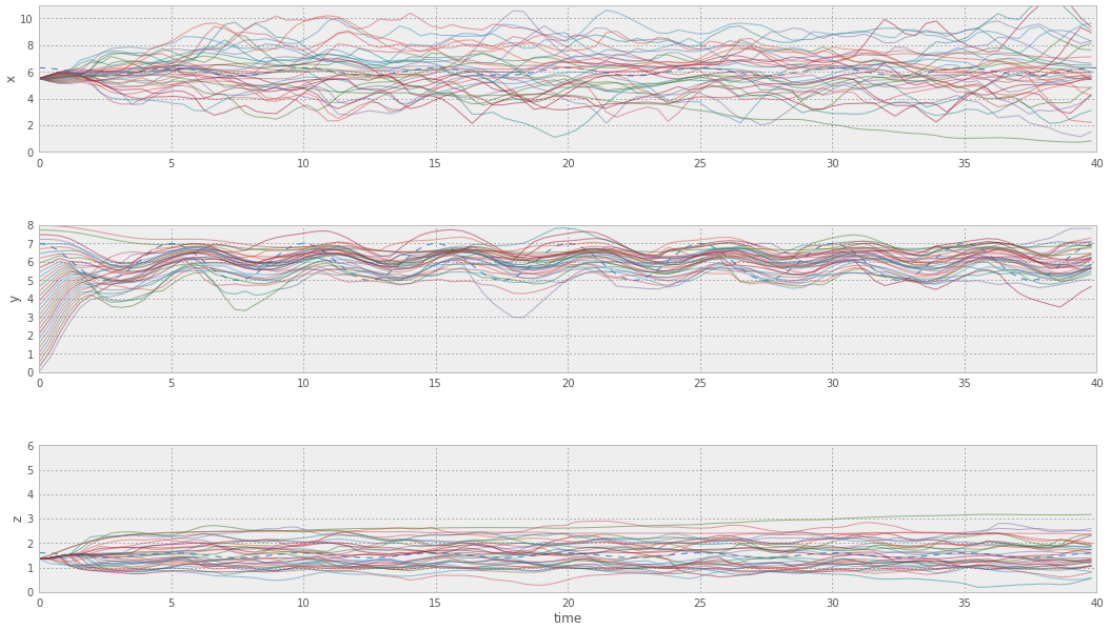
In [4]:
```
def simul(p, players, dt=.013, t_stop=40., s_VP = 0,
          display=True, polar=False, limit=True):
    time = np.arange(0., t_stop, dt)
    N_time = len(time)
    n_players = len(players)
    positions_ = np.zeros((3, n_players, N_time))
    particles = np.zeros((6, p['N'], N_time))
    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 = plt.figure(figsize=(18,10))
        T_step = max(N_time / 100, 1)
        if polar:
            rae_PC = xyz2azel(positions_, np.array([VPs[0]['x'],
                                        VPs[0]['y'],
                                        VPs[0]['z']]))
            rae_VC = xyz2azel(particles[:3, :, :], np.array([VPs[0]['x'],
                                        VPs[0]['y'],
                                        VPs[0]['z']]))
            for i_ax, axe in zip(range(3), ['r', 'a', 'e']):
                ax = fig.add_subplot(3, 1, 1+ i_ax)
                ax.plot(time, rae_PC[i_ax, :, :].T, '--')
                ax.plot(time[::T_step], rae_VC[i_ax, :, ::T_step].T, alpha=.5)
                #ax.errorbar(time[::T_step], particles[i_ax, :, ::T_step].mean(axis=0)
                #ax.errorbar(time[::T_step], particles[i_ax + 3, :, ::T_step].mean(axi
                ax.set_ylabel(axe)
        else:
```

```
            for i_ax, axe, d in zip(range(3), ['x', 'y', 'z'], [d_x, d_y, d_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)
                #ax.errorbar(time[::T_step], particles[i_ax + 3, :, ::T_step].mean(axi
                ax.set_ylabel(axe)
                ax.set_ylim([0., d])
        ax.set_xlabel('time')
    return positions_, particles
```

1 stabilité avec paramètres

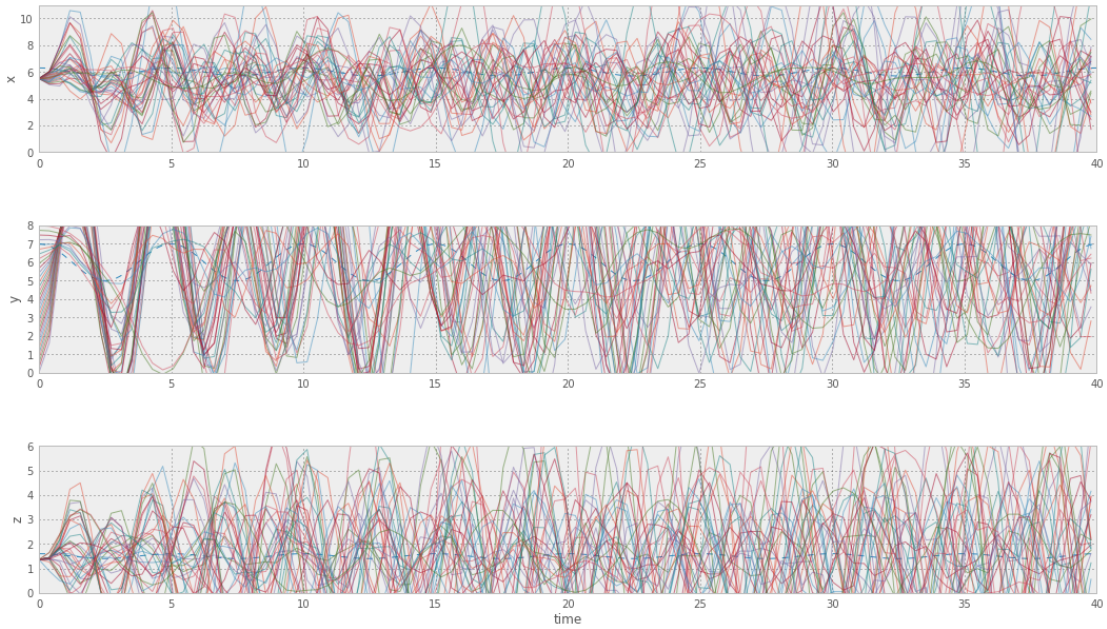
In [5]:
```
roger = [6., 6, 1.5]
players = [{'center': roger, 'amp': [.3,1.,.1] , 'T': 5.}]
print p['damp']
pos, particles = simul(p, players)
```

```
0.1
```

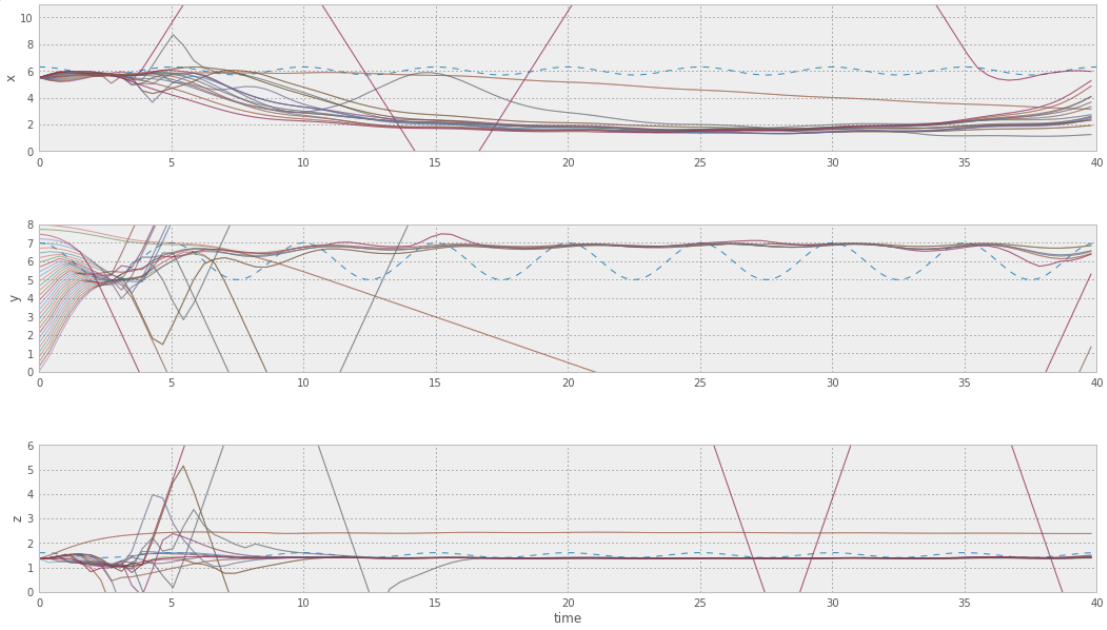


In [6]:
```
p_ = p.copy()
p_['damp'] = .0
print p_['damp']
pos, particles = simul(p_, players)
```

```
0.0
```



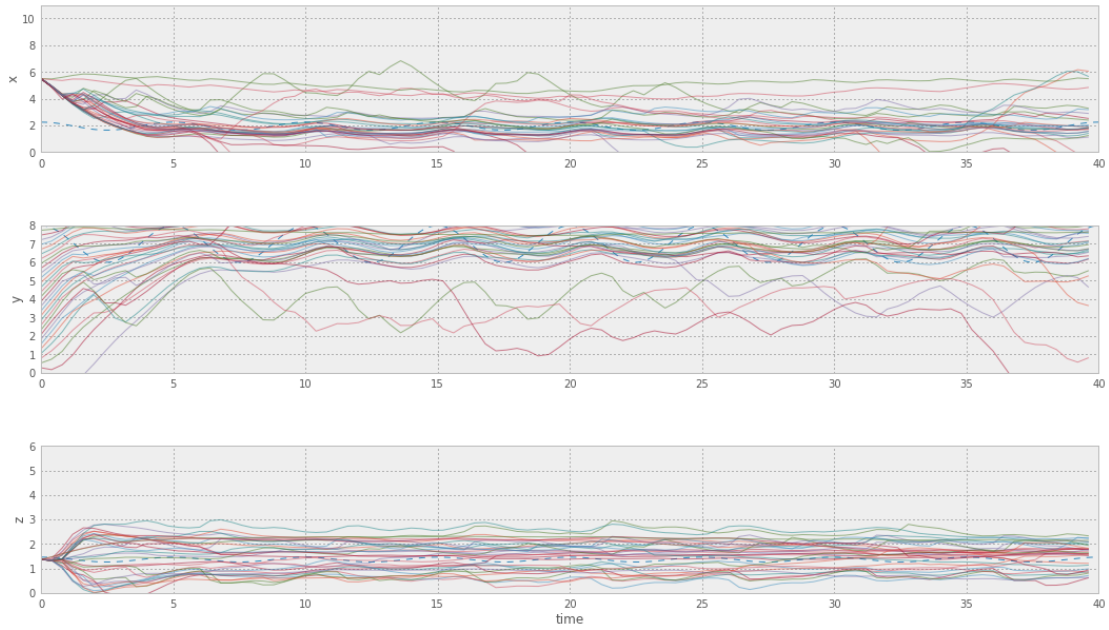
In [7]:
```
p_ = p.copy()
p_['G_repulsion'] = 0.
pos, particles = simul(p_, players)
```



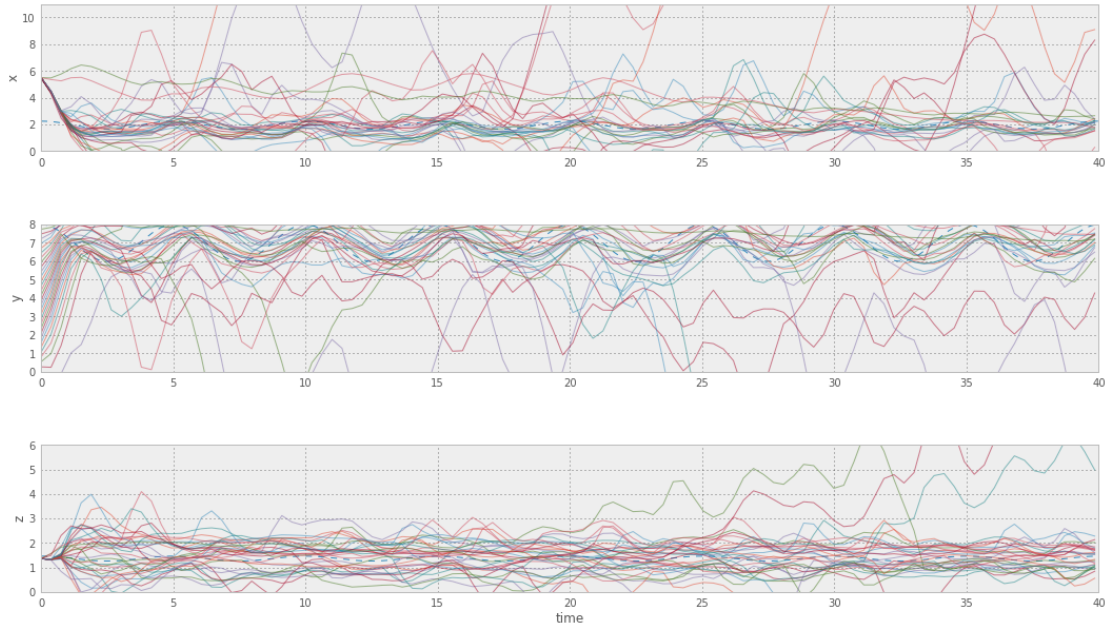
1.1 stabilité avec dt

In [8]:
```
for dt in np.logspace(-2., 0., 4, endpoint=False):
    print 'dt = ', dt
    players = [{'center': [VPs[0]['x'] + 1., VPs[0]['y'], VPs[0]['z']], 'amp': [.3, 1.
    pos, particles = simul(p, players, dt=dt)
    plt.show()
```

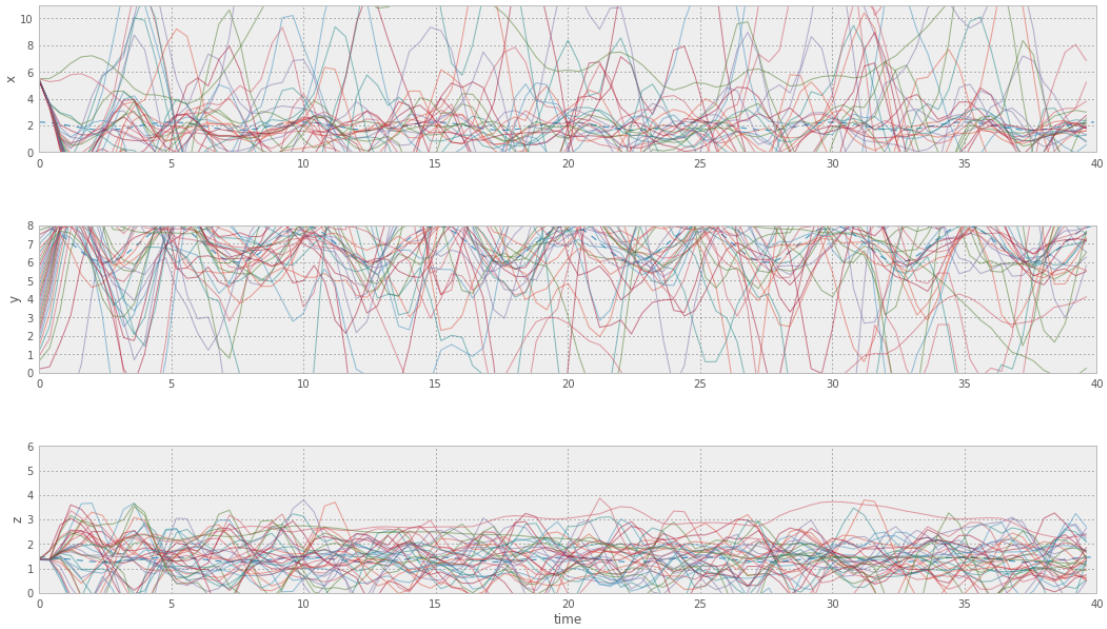
dt = 0.01



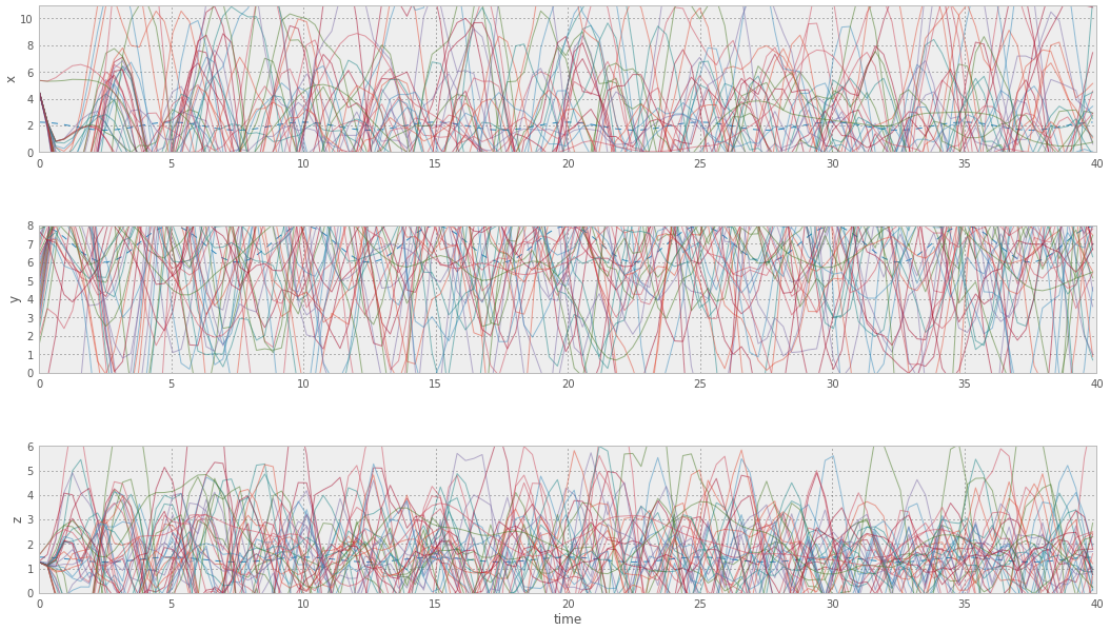
dt = 0.0316227766017



dt = 0.1




```
dt =  0.316227766017
```



2 players instables

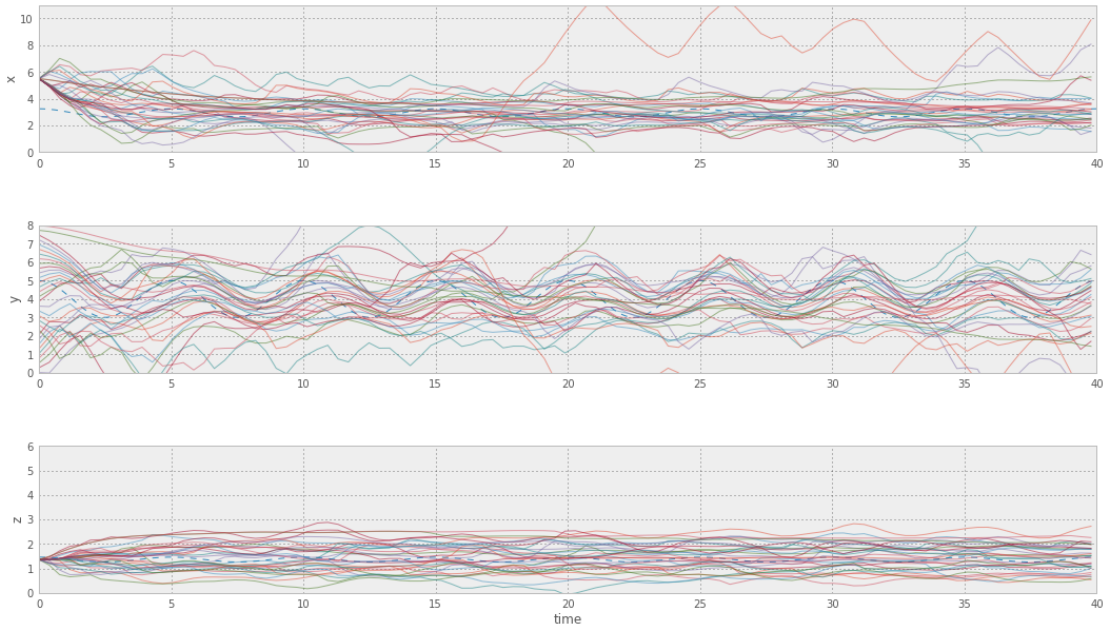
On définit un player seul qui se déplace devant le videoprojecteur, à une distance

In [9]:
```
i_vp = 1
print VPs[i_vp]
```

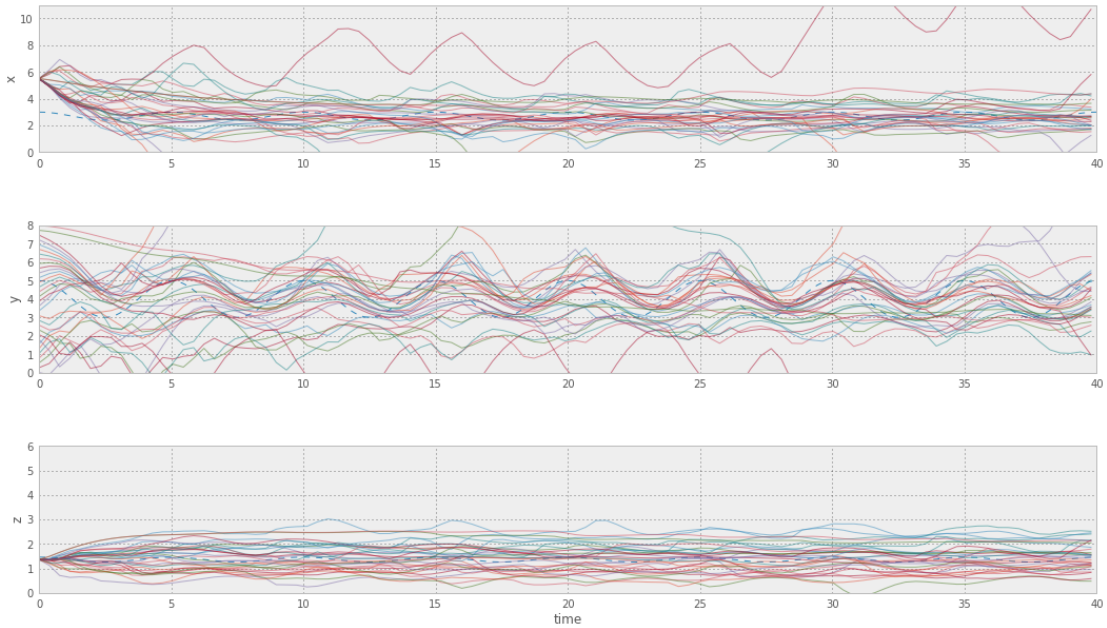
```
{'cz': 1.36, 'cy': 4.0, 'cx': 11.057115384615384, 'pc_min': 0.001,
'address': '10.42.0.52', 'y': 4.0, 'x': 0.9428846153846154, 'pc_max':
1000000.0, 'z': 1.36, 'foc': 21.802535306060136}
```

In [10]:
```
for distance in np.linspace(2., 0., 8, endpoint=False):
    print 'Distance du player au VP = ', distance
    players = [{'center': [VPs[i_vp]['x'] + distance, VPs[i_vp]['y'], VPs[i_vp]['z']],
    pos, particles = simul(p, players)
    plt.show()
```

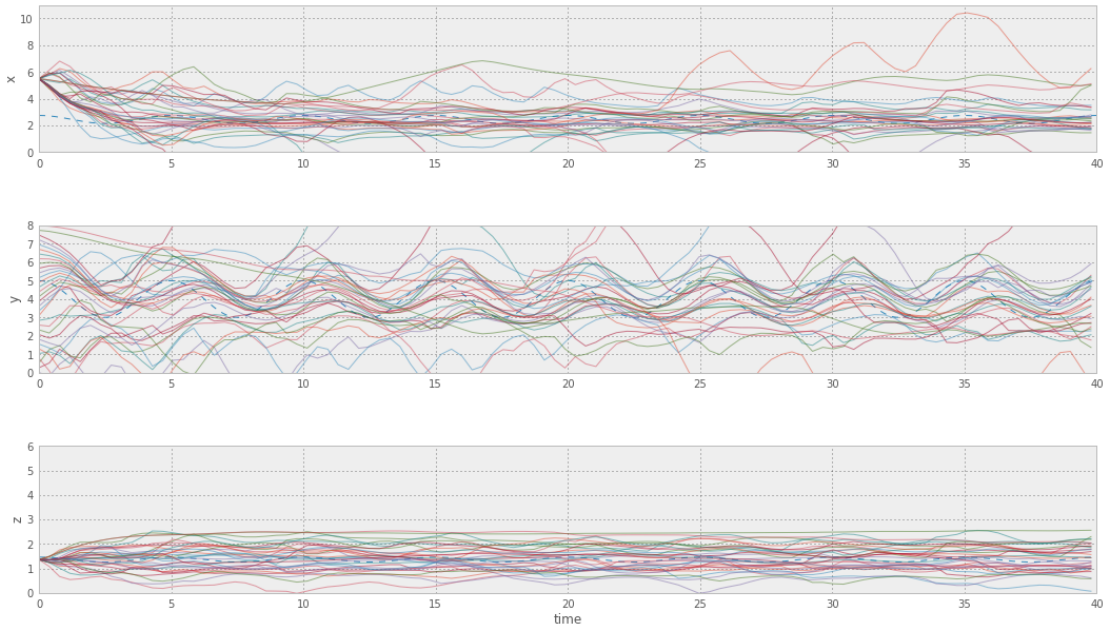
```
Distance du player au VP =  2.0
```



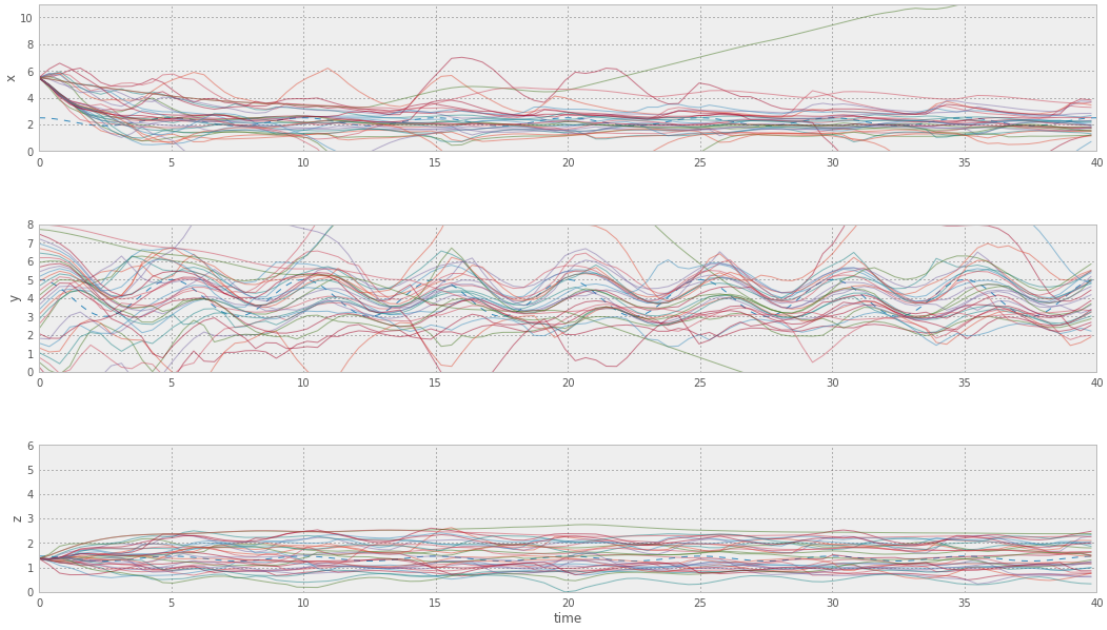

```
Distance du player au VP =  1.75
```



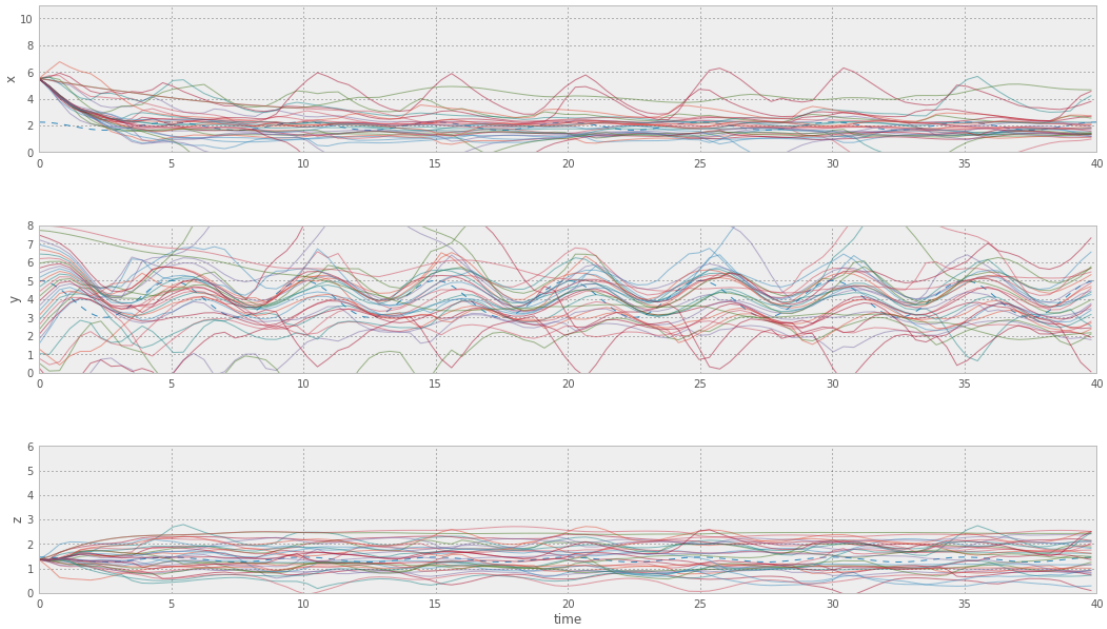

```
Distance du player au VP =  1.5
```



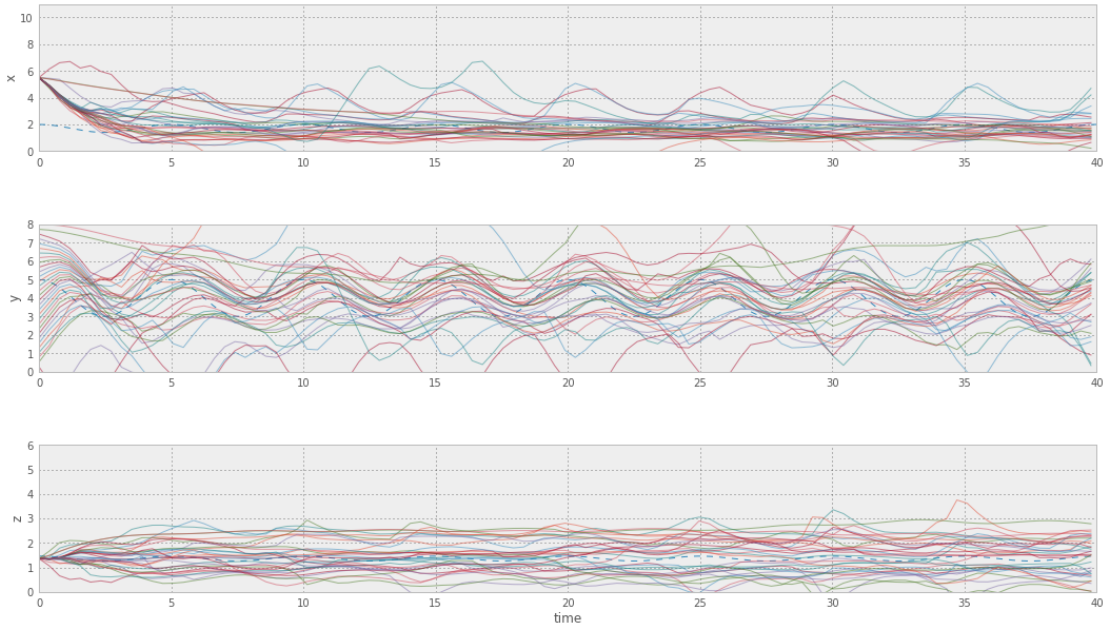
Distance du player au VP = 1.25



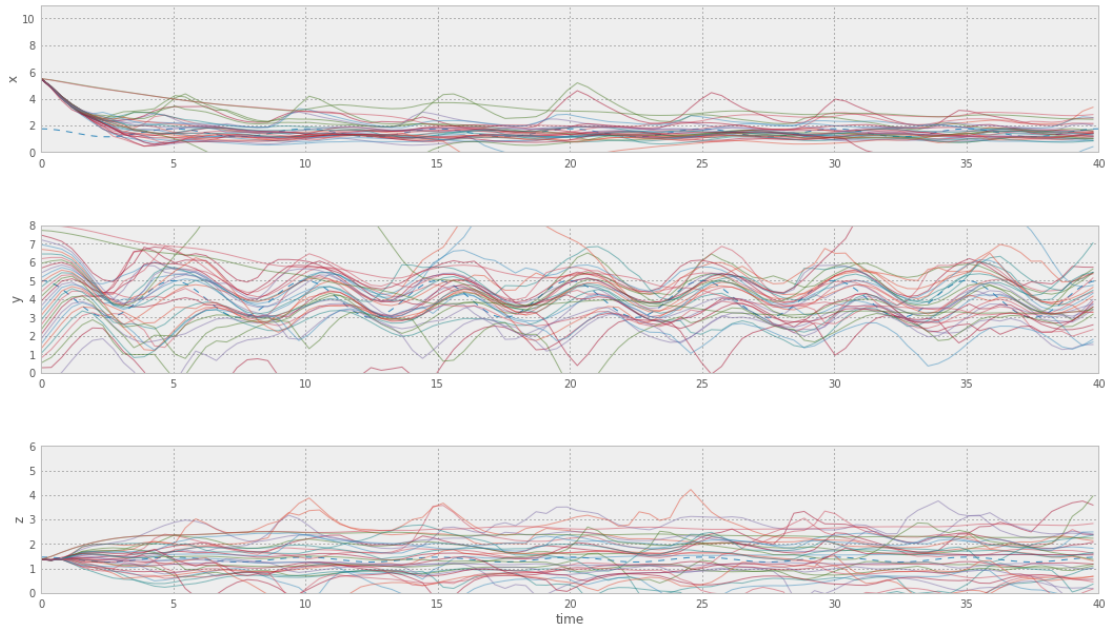
Distance du player au VP = 1.0



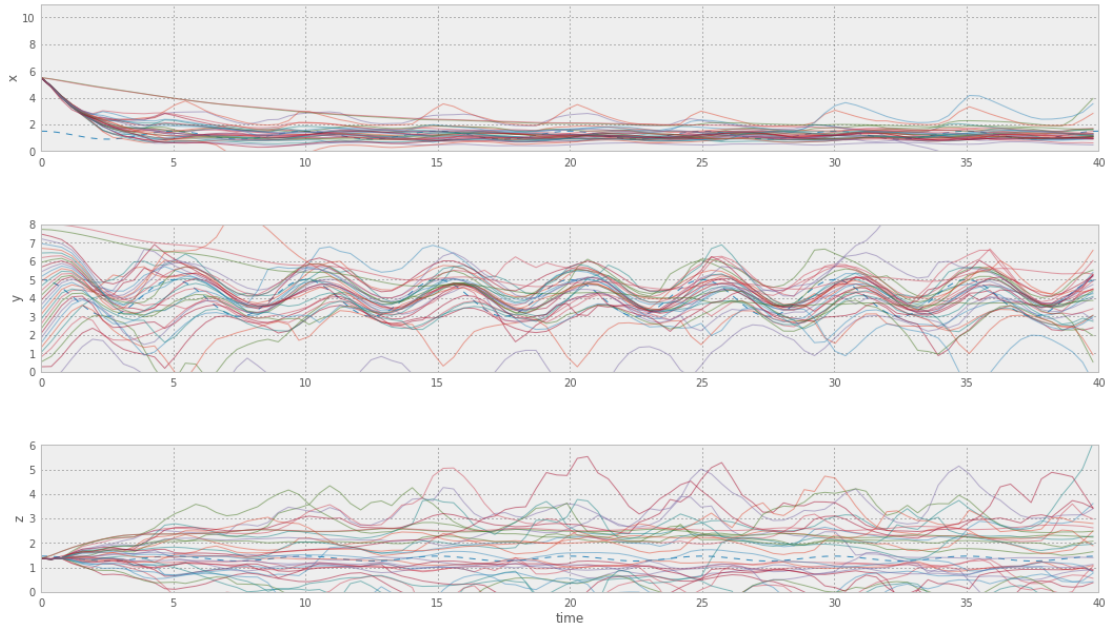
Distance du player au VP = 0.75



Distance du player au VP = 0.5



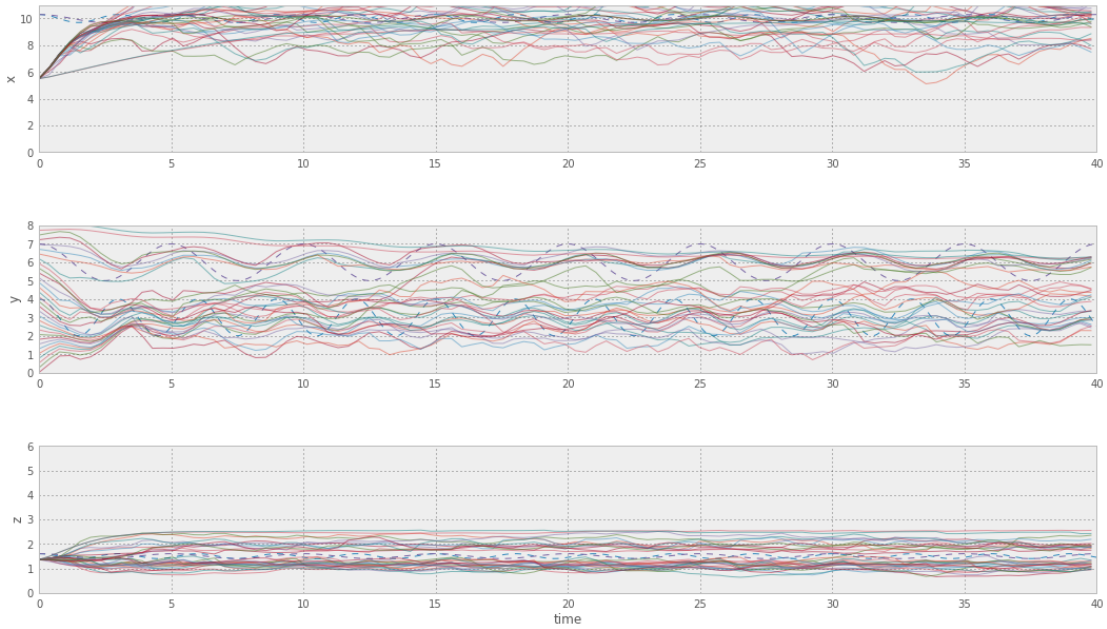

```
Distance du player au VP =  0.25
```



(note: il faut bien regarder l'amplitude des variations sur les axes)TODO : mes notes

3 multi players

In [11]:
```
players = [{'center': [10., 3, 1.5], 'amp': [.3,1.,.1] , 'T': 3.},
           {'center': [10., 6, 1.5], 'amp': [.3,1.,.1] , 'T': 5.}]
pos, particles = simul(p, players)
```

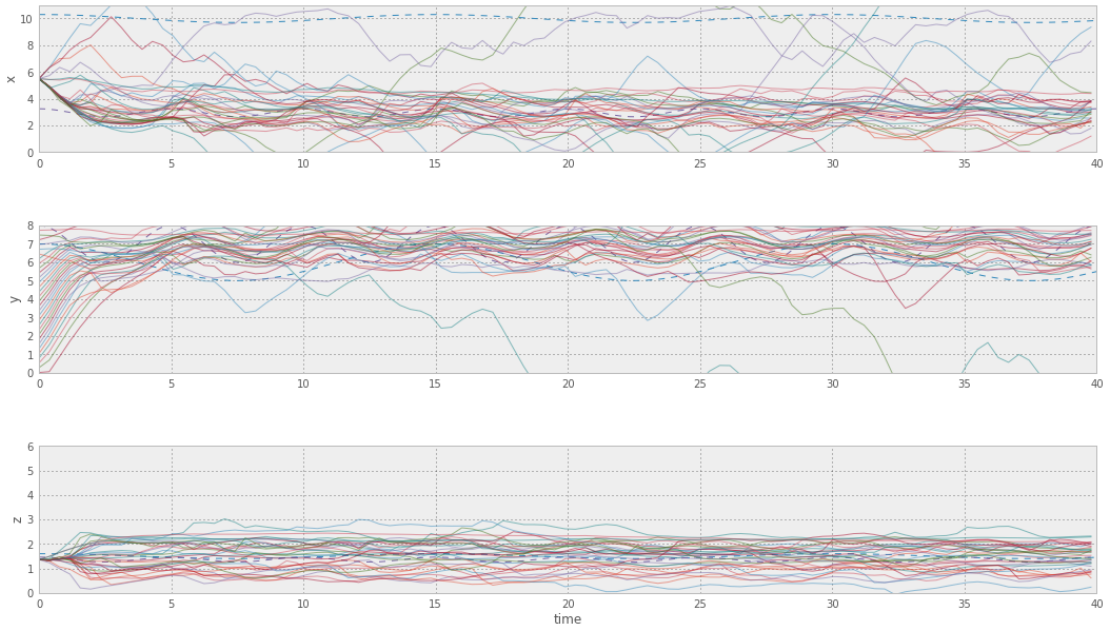


4 étude de la stabilité avec une autre personne

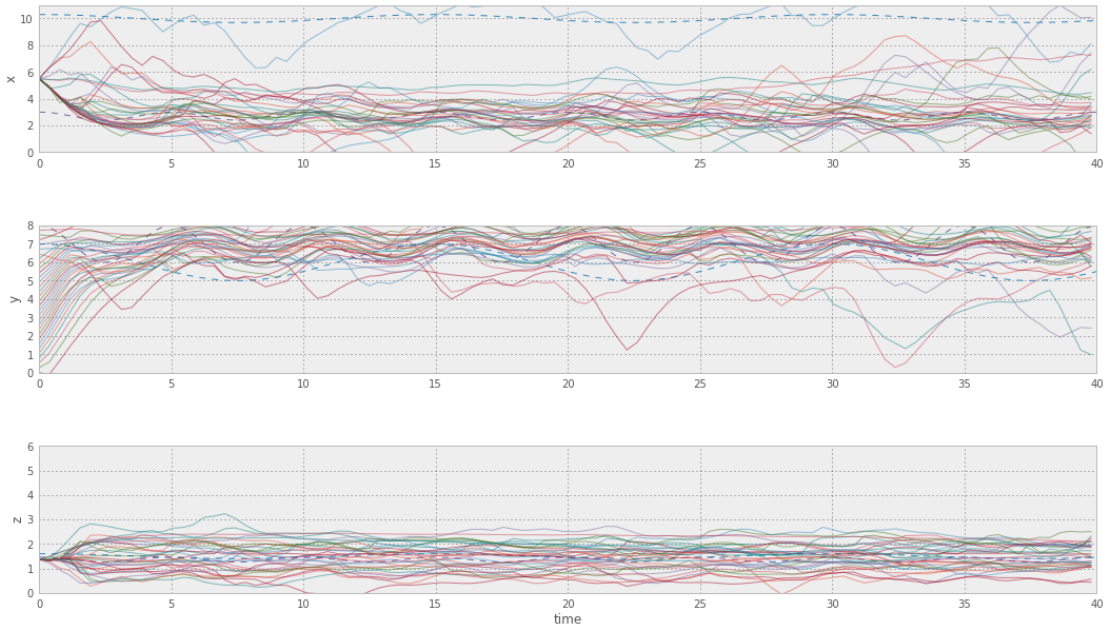
TODO: voir si on peut éviter de masquer des personnes dans le meme axe en annihilant cette force et en donnant ainsi plus de forces aux interactions venant des autres VPs / dynamique

In [12]:
```
for distance in np.linspace(2., 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)
    plt.show()
```

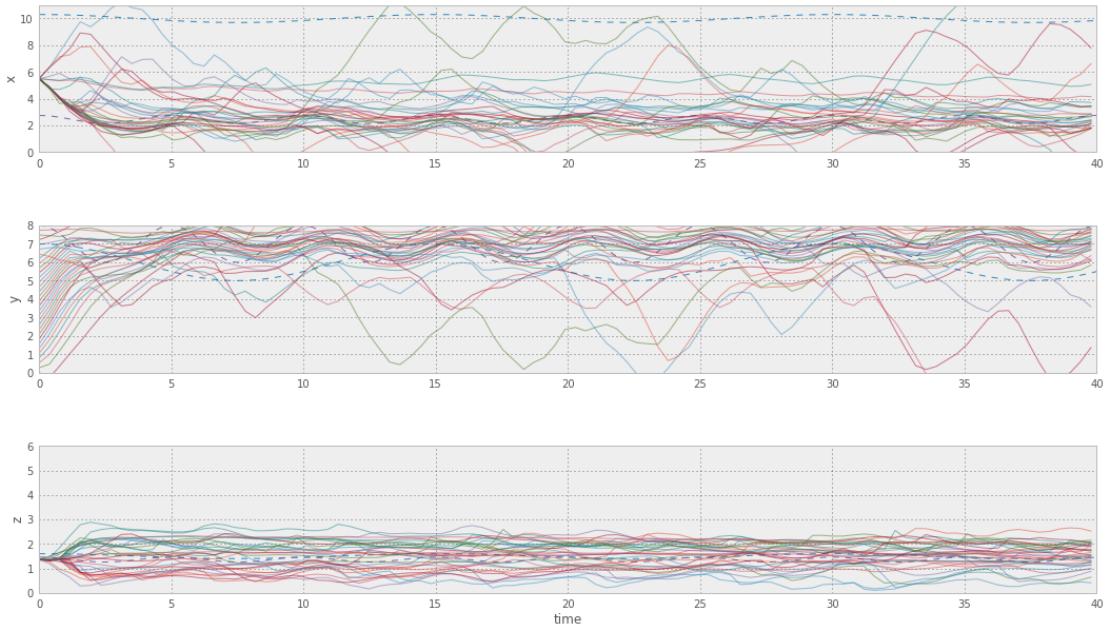
```
Distance du player au VP =  2.0
```



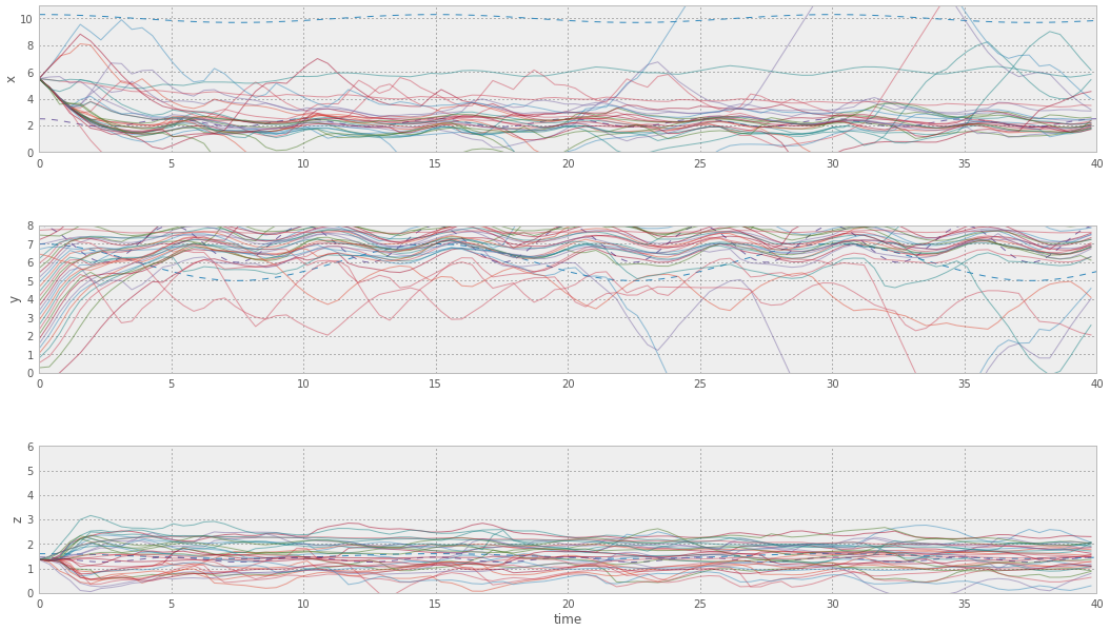
Distance du player au VP = 1.75



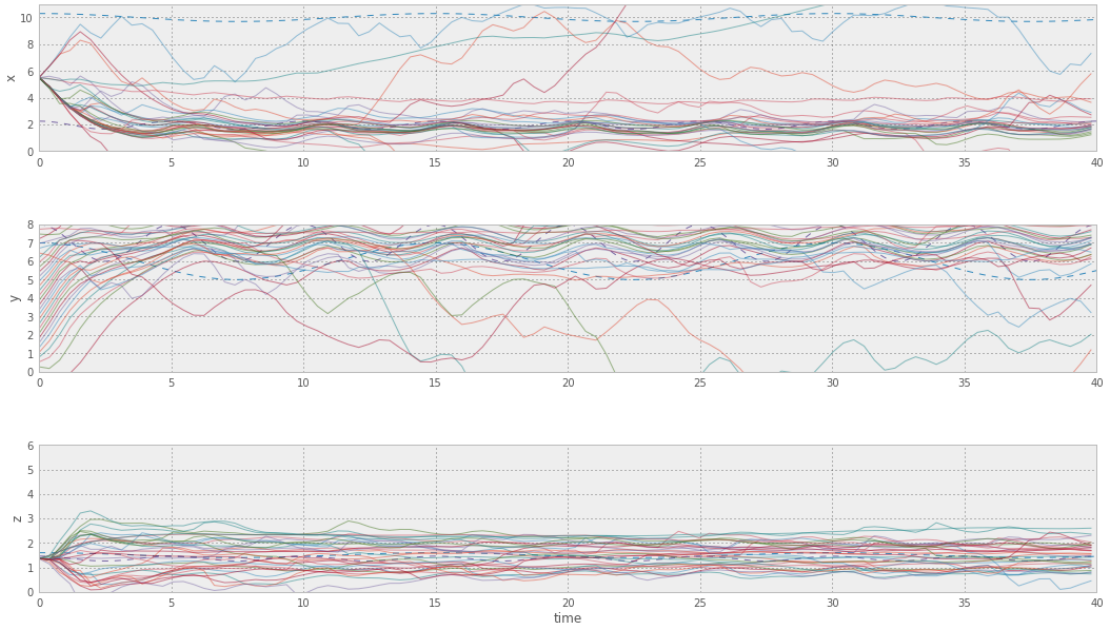
Distance du player au VP = 1.5



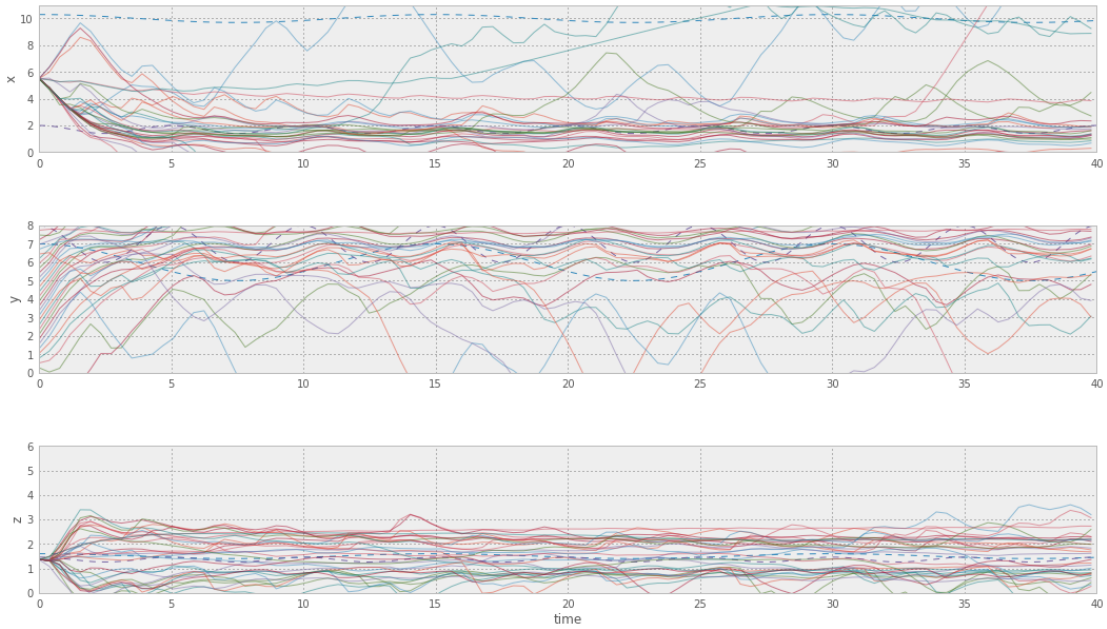
Distance du player au VP = 1.25



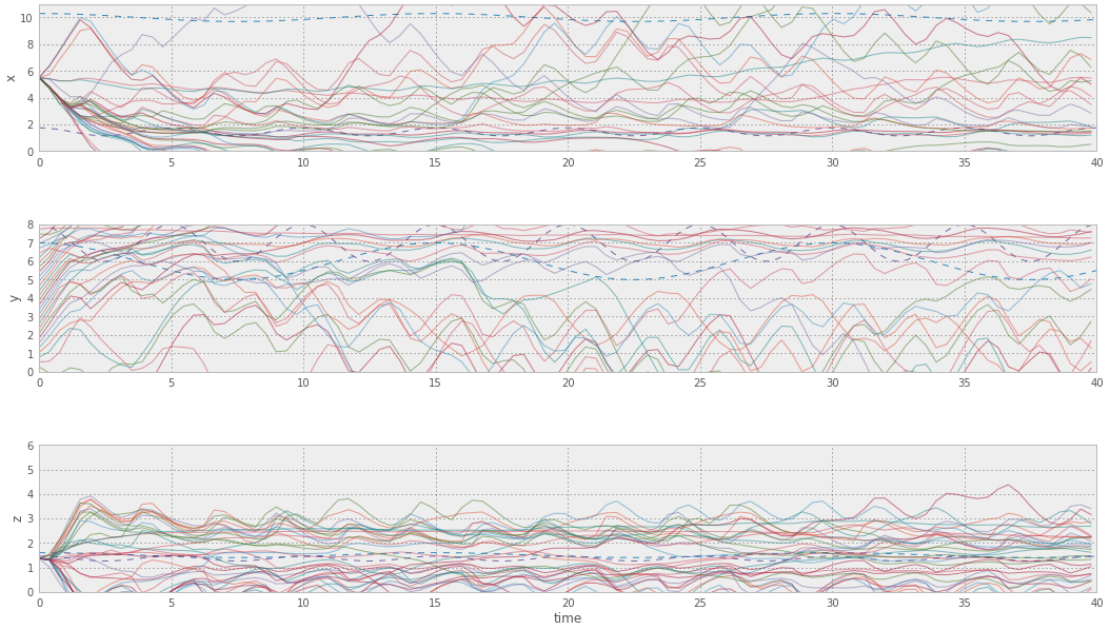
Distance du player au VP = 1.0



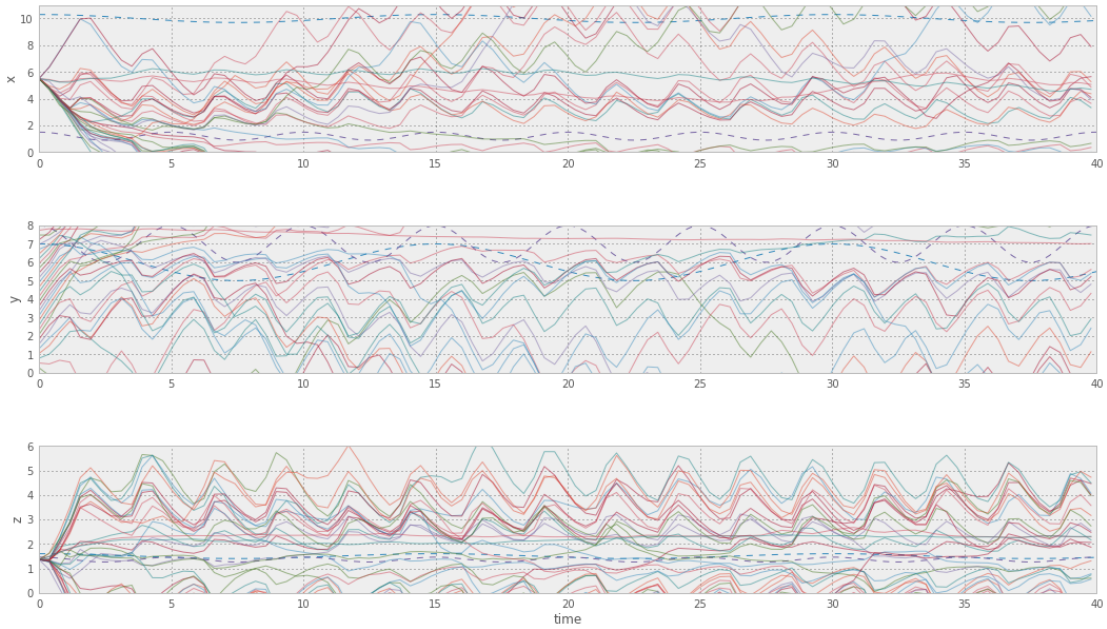
Distance du player au VP = 0.75



Distance du player au VP = 0.5



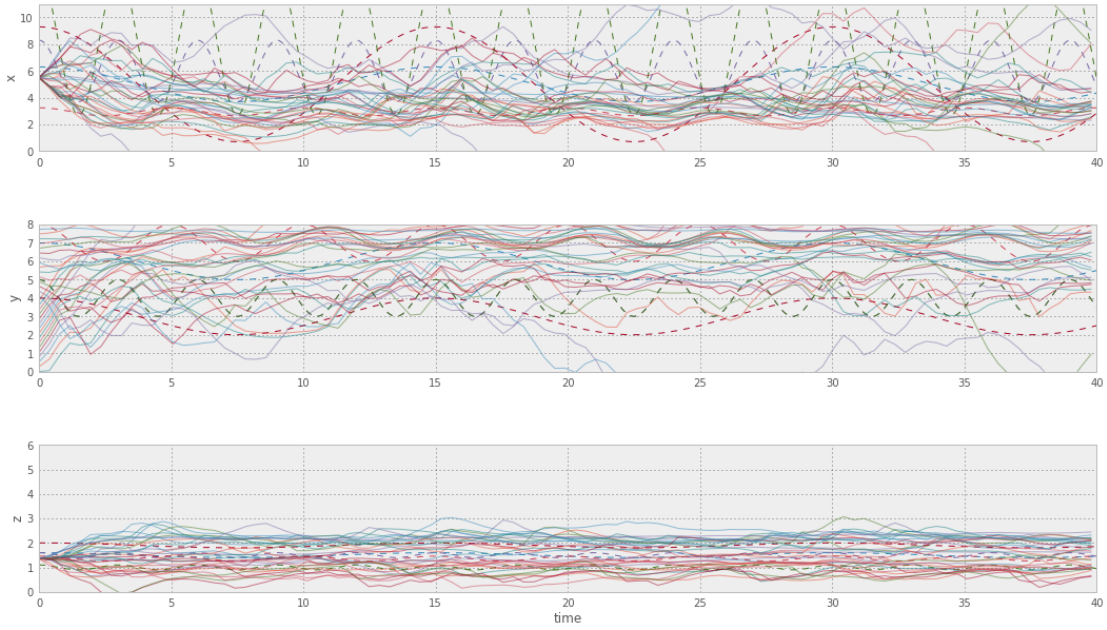
Distance du player au VP = 0.25



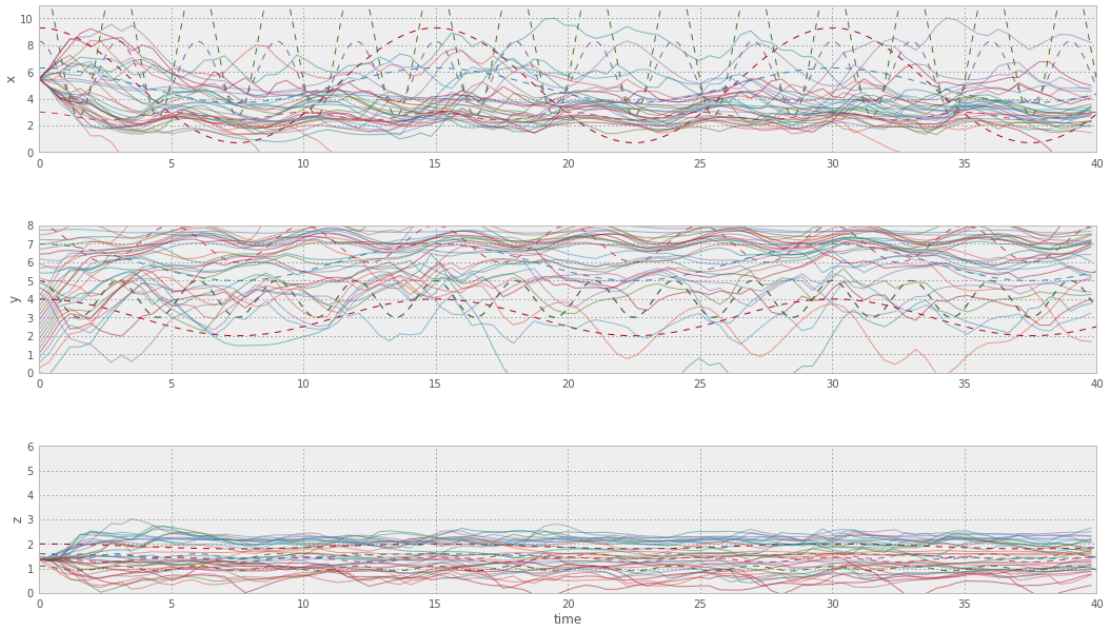
5 étude de la stabilité avec quatre autres personnes

In [13]:
```
for distance in np.linspace(2., 0., 8, endpoint=False):
    players = [{'center': [5., 6, 1.5], 'amp': [1.3,1.,.1] , 'T': 15.},
               {'center': [6., 4, 1.5], 'amp': [2.3,1.,.1] , 'T': 3.},
               {'center': [5., 3, 1.9], 'amp': [4.3,1.,.1] , 'T': 15.},
               {'center': [8., 4, 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)
    plt.show()
```

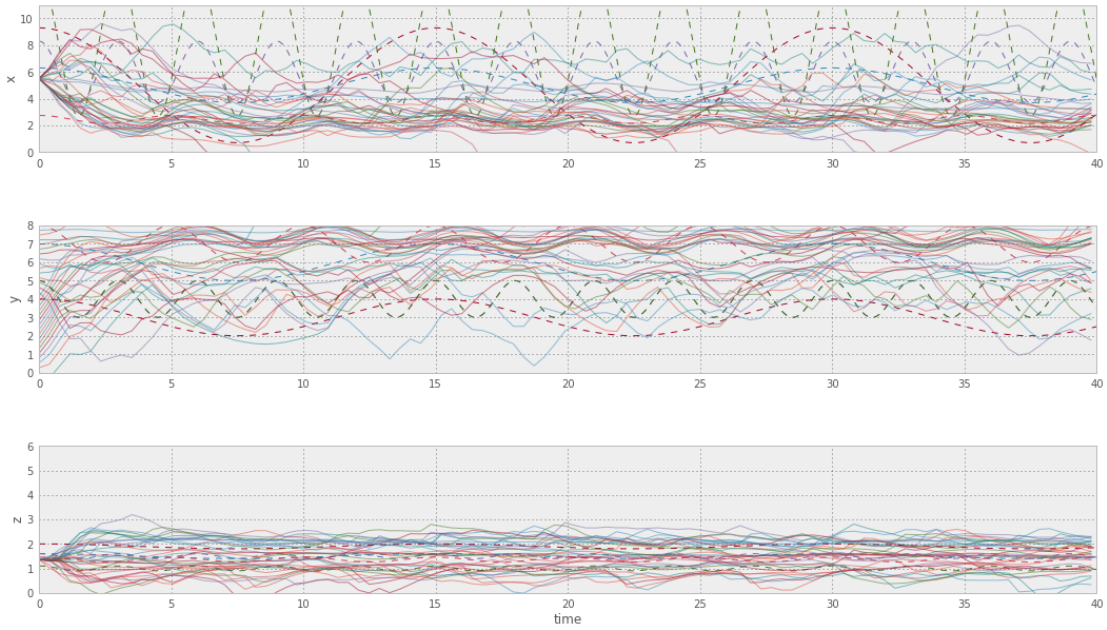
```
Distance du player au VP =  2.0
```



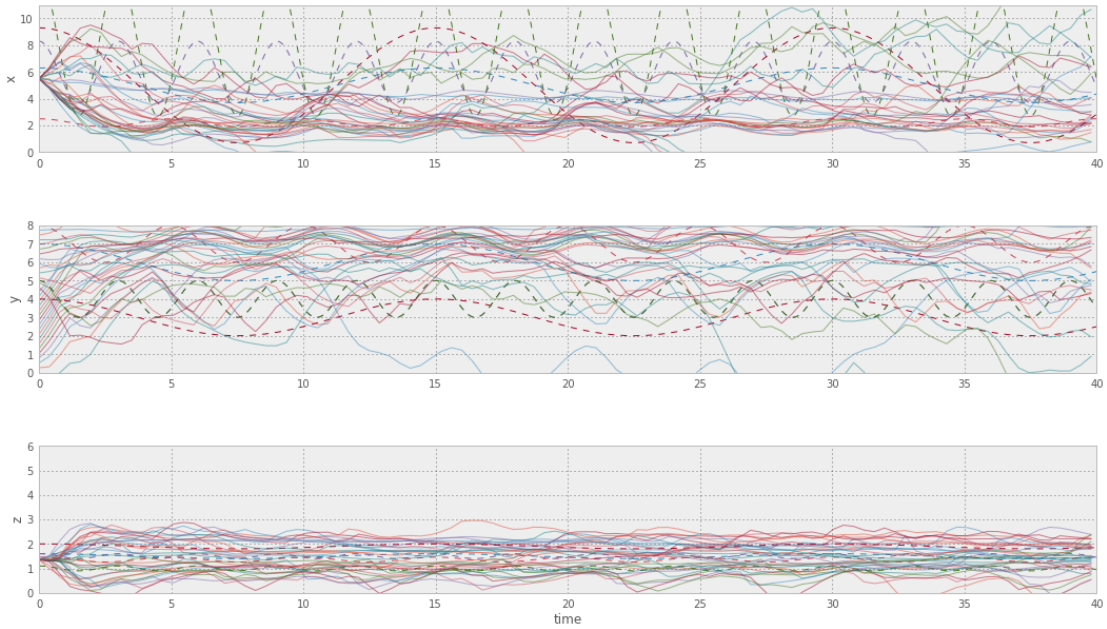
Distance du player au VP = 1.75



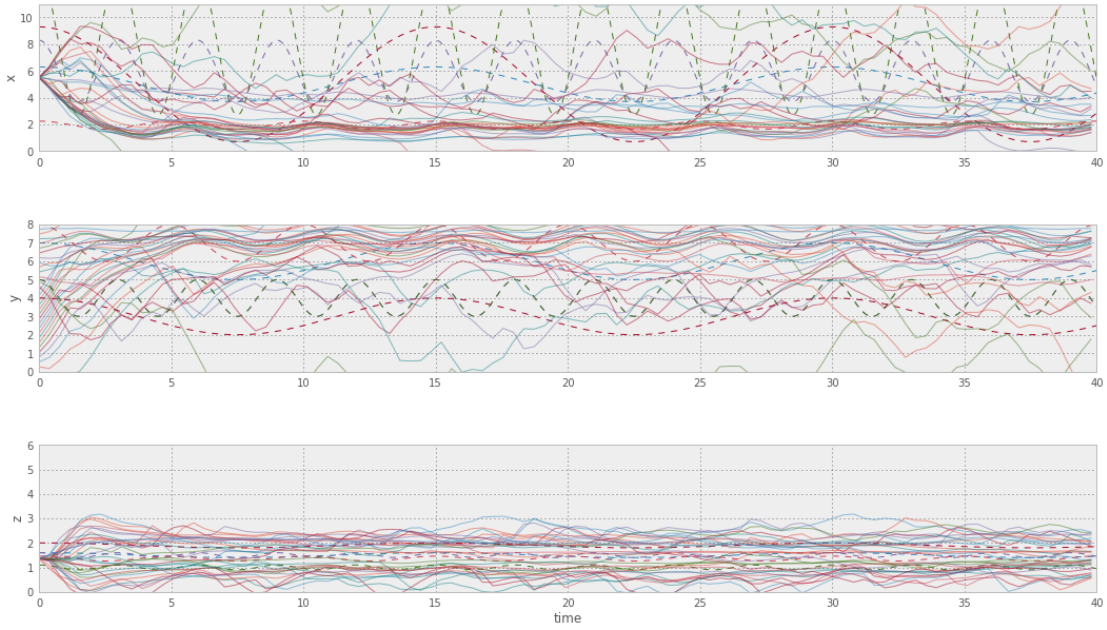
Distance du player au VP = 1.5



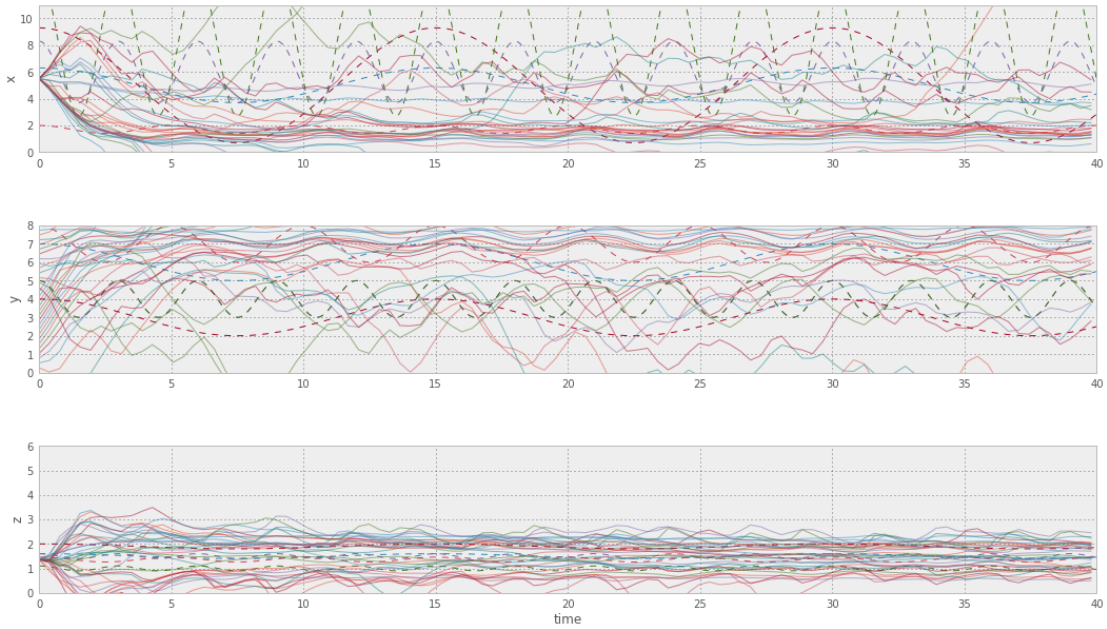
Distance du player au VP = 1.25



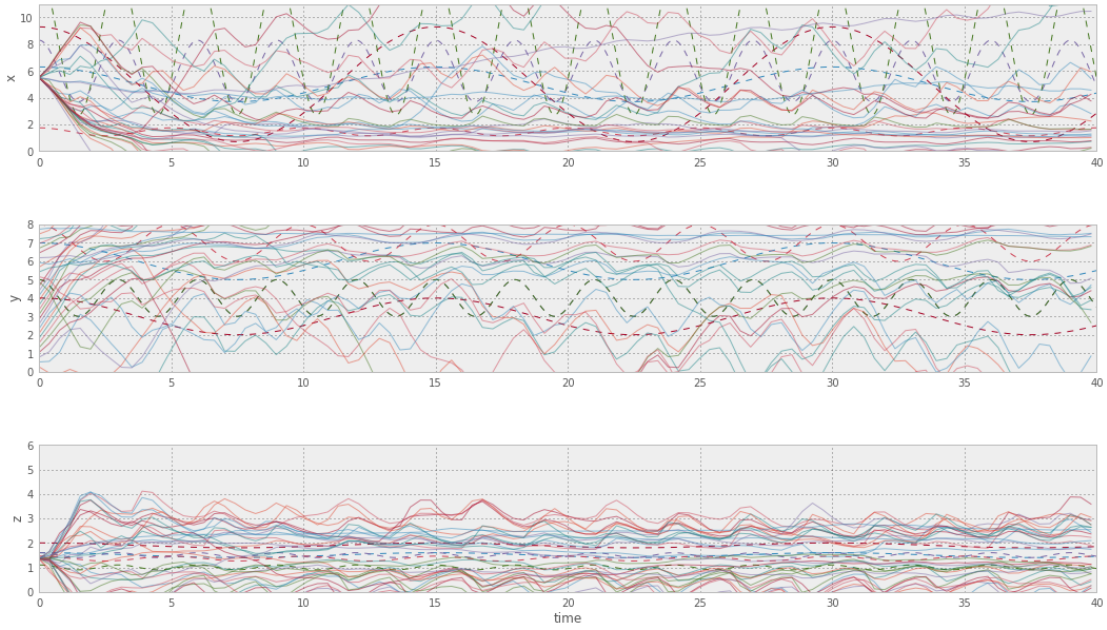
Distance du player au VP = 1.0



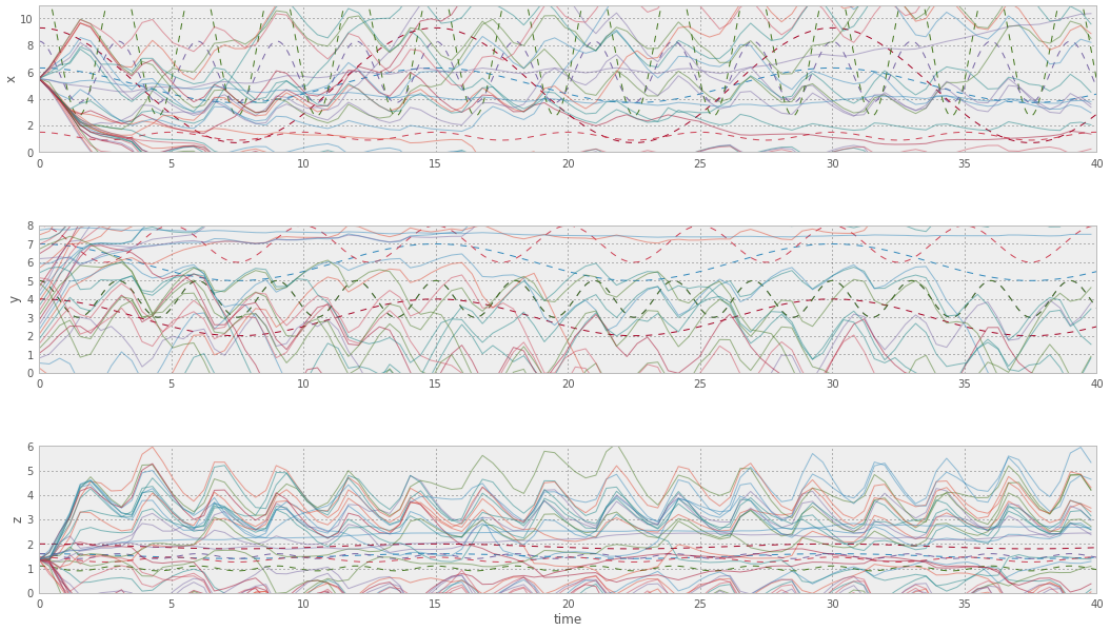
Distance du player au VP = 0.75



Distance du player au VP = 0.5



Distance du player au VP = 0.25



In [13]: