C_Vida

October 29, 2016

0.1 Ejemplos de animaciones

In [1]:
```
#!/usr/bin/env python
"""
An animated image
"""
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation

fig = plt.figure()


def f(x, y):
    return np.sin(x) + np.cos(y)

x = np.linspace(0, 2 * np.pi, 120)
y = np.linspace(0, 2 * np.pi, 100).reshape(-1, 1)

im = plt.imshow(f(x, y), cmap=plt.get_cmap('coolwarm'), animated=True)


def updatefig(*args):
    global x, y
    x += np.pi / 15.
    y += np.pi / 20.
    im.set_array(f(x, y))
    return im,

ani = animation.FuncAnimation(fig, updatefig, interval=50, blit=True)
plt.show()
```

In [2]:
```
import numpy as np
from matplotlib import pyplot as plt
from matplotlib import animation

fig = plt.figure()
fig.set_dpi(100)
```

```
        fig.set_size_inches(7, 6.5)

        ax = plt.axes(xlim=(0, 10), ylim=(0, 10))
        patch = plt.Circle((5, -5), 0.75, fc='y')

        def init():
            patch.center = (5, 5)
            ax.add_patch(patch)
            return patch,

        def animate(i):
            x, y = patch.center
            x = 5 + 3 * np.sin(np.radians(i))
            y = 5 + 3 * np.cos(np.radians(i))
            patch.center = (x, y)
            return patch,

        anim = animation.FuncAnimation(fig, animate,
                                       init_func=init,
                                       frames=360,
                                       interval=20,
                                       blit=True)

        plt.show()
```

```
In [3]: import numpy as np
        import matplotlib.pyplot as plt
        import matplotlib.animation as animation

        def generate_data():
            a = np.arange(25).reshape(5, 5)
            b = 10 * np.random.rand(5, 5)
            return a - b

        def update(data):
            mat.set_data(data)
            return mat

        def data_gen():
            while True:
                yield generate_data()

        fig, ax = plt.subplots()
        mat = ax.matshow(generate_data())
        plt.colorbar(mat)
        ani = animation.FuncAnimation(fig, update, data_gen, interval=500,
                                      save_count=50)
        plt.show()
```

0.2 Conway Life

```
In [4]: import numpy as np
        import matplotlib.pyplot as plt
        import random
        import matplotlib.animation as animation



        def vecinos(N,(n,m)):
            return [(n-1%N,m%N),(n%N,(m-1)%N),((n+1)%N,m%N),(n%N,(m+1)%N),\
                     ((n-1)%N,(m-1)%N),((n-1)%N,(m+1)%N),\
                     ((n+1)%N,(m+1)%N),((n+1)%N,(m-1)%N)]



        M = np.array([[random.randint(0,1) for _ in range(128)] \
                        for _ in range(128)])
        ##M[8,8] = 1;M[8,9] = 1;M[9,9] = 1;M[9,8] = 1;M[10,8] = 1;M[8,10] = 1


        def siguiente(M):
            N = M.shape[0]
            M2 = np.array([[0]*N for _ in range(N)])
            for p in range(N):
                for q in range(N):
                    L = vecinos(N,(p,q))
                    vivos =len([1 for par in L if M[par[0],par[1]] == 1])
                    if (M[p,q] == 1) and (2 <= vivos <= 3):
                        M2[p,q] = 1
                    elif  (M[p,q] == 0) and (vivos == 3):
                        M2[p,q] = 1

            return M2


        def generate_data(M):
               return siguiente(M)

        def update(data):
            mat.set_data(data)
            return mat

        def data_gen():
            global M
            M = generate_data(M)
            yield M
```

```
fig, ax = plt.subplots()
mat = ax.matshow(generate_data(M))

ani = animation.FuncAnimation(fig, update, data_gen,interval=1)
plt.show()
```

0.3 C_life optimizado numpy+cython

```
In [5]: %load_ext autotime
        %load_ext cython

In [6]: %%cython -a
        import numpy as np
        cimport numpy as np

        import random

        cimport cython

        @cython.boundscheck(False)
        @cython.cdivision(True)
        cdef int vecinosVivos(np.uint16_t[:,:] M,int p,int q):
            cdef int v = 0
            cdef int i,j

            cdef int N = M.shape[0]

            for i in range(p - 1,p + 2):
                for j in range(q - 1,q + 2):
                    if i == j:
                        continue
                    v += M[i % N,j % N]

            return v

        @cython.boundscheck(False)
        def siguiente(np.ndarray[np.uint16_t,ndim=2] M):
            cdef int N = M.shape[0]
            cdef np.ndarray[np.uint16_t,ndim=2] M2 = np.zeros((N,N), \
                dtype=np.uint16)

            cdef np.uint16_t[:,:] MV = M
            cdef np.uint16_t[:,:] M2V = M2

            cdef int p
            cdef int q
            cdef int vivos
            for p in range(N):
                for q in range(N):
                    vivos = vecinosVivos(MV,p,q)

                    if (MV[p,q] == 1) and (2 <= vivos <= 3):
                        M2V[p,q] = 1
                    elif  (MV[p,q] == 0) and (vivos == 3):
                        M2V[p,q] = 1

            return M2
```

```
Out[6]: <IPython.core.display.HTML object>
```

```
time: 15.6 ms
```

```
In [7]: M = np.random.randint(0,2,size=4096**2).\
            reshape((4096,4096)).astype(np.uint16)
        %timeit MM = siguiente(M)
```

```
1 loop, best of 3: 1.4 s per loop
time: 5.97 s
```

```
In [8]: import numpy as np
        import matplotlib.pyplot as plt
        import random
        import matplotlib.animation as animation



        M = np.random.randint(0,2,size=128**2).\
            reshape((128,128)).astype(np.uint16)

        def generate_data(M):
               return siguiente(M)

        def update(data):
            mat.set_data(data)
            return mat

        def data_gen():
            global M
            M = generate_data(M)
            yield M

        fig, ax = plt.subplots()
        mat = ax.matshow(generate_data(M))
        ani = animation.FuncAnimation(fig, update, data_gen,interval=100)
        plt.show()
```

```
time: 2.48 s
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```