# Representing traveling waves visually with vpython

In [1]:
```
import vpython as vp      # import all vpython functions including numpy incompatible sin, cos, exp, etc
import numpy as np
```

In [2]:
```
vp.canvas()

L=6.0                           # range of x is 6 units
x = np.linspace(-L/2, L/2, 20)    # from -3 to +3
k = 3*np.pi/L                     # set up the wave number
psi = np.exp(1j*k*x)              # set up the initial wave function

alist = []                      # an empty list for our arrow objects

def SetArrowFromCN( cn, a):
    """
    SetArrowFromCN takes a complex number  cn  and an arrow object  a .
    It sets the  y  and  z  components of the arrow s axis to the real
    and imaginary parts of the given complex number.

    Just like Computing Project 1, except y and z for real/imag.
    """
    a.axis.y = cn.real
    a.axis.z = cn.imag

for i in range(len(x)):
    a = vp.arrow(pos=vp.vec(x[i], 0, 0),  # on the y,z axis at location 'x'
                 axis=vp.vec(0,1,0),      # pointing in the 'real' direction
                 color=vp.color.red)      # make it red. ;->
    alist.append(a)                       # add to list
    SetArrowFromCN( psi[i], a)            # set up arrow from wave function
```

In [3]:
```
vp.canvas()  # open a new vpython window

omega = 2*np.pi             # 1 rev/sec
t=0.0                       # start t at zero
dt=0.01                     # 1/100 of a second per step

while t < 20:
    vp.rate(100)
    t+=dt
    for i in range(len(x)):
        psi1 = np.exp(1j*k*x) * (np.exp(-1j * omega * t))
        SetArrowFromCN( psi1[i], alist[i])
```

# Questions

## 1.

The wave appears to be propagating to the right. It moves this way because we can see from the wave function that the coefficient of x is a positive value of the wave number, k.

## 2.

The velocity of the crest is the phase velocity which is $\frac{\omega}{k}$.

$$v = \frac{2\pi}{\frac{3\pi}{L}} = 4m/s$$

## 3.

If we negate the wave number, the wave would move in the opposite direction. I understand that in general, changing a wavefunction would change the expectation value but changing the k number to negative, mathematically won't have any effect on the expectation value.

In [4]:
```
vp.canvas()  # open a new vpython window

omega = 2*np.pi             # 1 rev/sec
t=0.0                       # start t at zero
dt=0.01                     # 1/100 of a second per step

while t < 20:
    vp.rate(100)
    t+=dt
    for i in range(len(x)):
        psi1 = np.exp(1j*-k*x) * (np.exp(-1j * omega * t))
        SetArrowFromCN( psi1[i], alist[i])
```