

**Equations of motion**

- $V_1 = V_0 + at$ 
  - v is velocity (units m/s)
  - a is acceleration (unit  $m/s^2$ )
  - t is time (units seconds)
- the final velocity equals the initial velocity + time × acceleration
- if we integrate this equation with respect to time we get:
- $X_1 = X_0 + \frac{1}{2}at^2 + V_0t$
- the final position of the object equals the initial position +  $\frac{1}{2}at^2 + V_0t$

**Equations of motion**

- $V^2 = U^2 + 2a(X_1 - X_0)$
- final velocity  $V^2$  equals the initial velocity  $U^2 + 2 \times$  the acceleration × the difference in position
- $X_1 = X_0 + \frac{1}{2}(V + U)t$

**Equations of motion**

- $F = ma$
- force in Newtons (Kg  $m/s^2$ )
- the force required to accelerate one kilogram of mass at the rate of one metre per second squared
  - Saturn V rocket generated 34.5 million newtons (lift off) !
  - the translunar injection burn propelled the rocket to 10408 m/s (23286 miles per hour)

**Hooke's Law**

- $F = -k(l_1 - l_0)$
- the force is equal to  $-k \times$  the current position of the spring - the at rest position of the spring
  - $l_0$  at rest position of the spring
  - $l_1$  current position of the spring
  - k is the spring constant

## Hookes Law

- $F = ma$
- $a = \frac{F}{m}$
- $a = \frac{-k(l_1 - l_0)}{m}$

## PGE and springs

- examine the example code (<https://github.com/gaiusm/pge/blob/master/examples/springs/simple.py>)
- creates a two circles, one is fixed, one is moving and a spring between them
  - in function main
- <https://github.com/gaiusm/pge/blob/master/examples/springs>

```
first = placeBall (wood_light, 0.55, 0.95, 0.03).fix ()
second = placeBall (wood_dark, 0.55, 0.35, 0.03).mass (1.
s = pge.spring (first, second, 100.0, 3.0, 0.5).draw (yel
```

## PGE and springs

- the parameters to the `spring` method are:
  - `first` object
  - `second` object
  - spring constant ( $k = 100.0$ )
  - damping constant ( $d = 3.0$ )
  - at rest length ( $l = 0.5$ )
- if the at rest length is omitted then it is assumed that the distance between `first` and `second` is the at rest value

## Spring damping value

- $F_s = -k_s(l - r)$ 
  - $F_s$  is the force of the spring
  - $k$  is the Hookes constant of the spring
  - $l$  is the current position of the spring
  - $r$  is the at rest position of the spring
- if we model the spring using this equation, the spring will bounce an object forever
  - we need a method to extract energy out of the spring, to give it realism

## Spring damping value

- $F_d = -k_d(v_1 - v_2)$ 
  - $F_d$  is the damping force
  - $k$  is the hookes value of the damping value of the spring
  - $v_1$  is the velocity of object 1 at the end of the spring
  - $v_2$  is the velocity of object 2 at the other end of the spring

## Spring damping value

- finally these two equations are joined together
- $F_1 = -(k_s(l - r) + k_d((v_1 - v_2) \cdot L)/l)L/l$
- $F_1$  is a scalar overall force of the spring at the current position
  - $k_s$  is the Hookes constant for the spring
  - $k_d$  is the Hookes damping constant for the spring
  - $v_1$  and  $v_2$  are vectors of the velocity of the two objects connected by the spring
  - $L$  is a vector of the positional difference between the two objects
  - $l$  is the distance between the two objects

## Spring damping value

- notice the dot product, which for a vector of two items is equal to:
- $a \cdot b = a_0 \times b_0 + a_1 \times b_1$
- the input parameters to a dot product operator are vectors and the result is a scalar
- the force acting on object 1 is  $F_1$
- the force acting on object 2 is  $F_2 = -F_1$

## Springs in PGE

- take a look at the example [bridge.py](https://github.com/gaiusm/pge/blob/master/examples/springs/bridge.py) (<https://github.com/gaiusm/pge/blob/master/examples/springs/bridge.py>) which implements a simple bridge using 4 non fixed circles and 5 springs
- notice that PGE allows us to define a callback for a spring when it reaches a length
  - in the `bridge` example a spring will snap when it reaches `snap_length` (0.16)