Introduction

#1

Serdar ARITAN

Biomechanics Research Group,
Faculty of Sports Sciences, and
Department of Computer Graphics
Hacettepe University, Ankara, Turkey

Motion in One Dimension

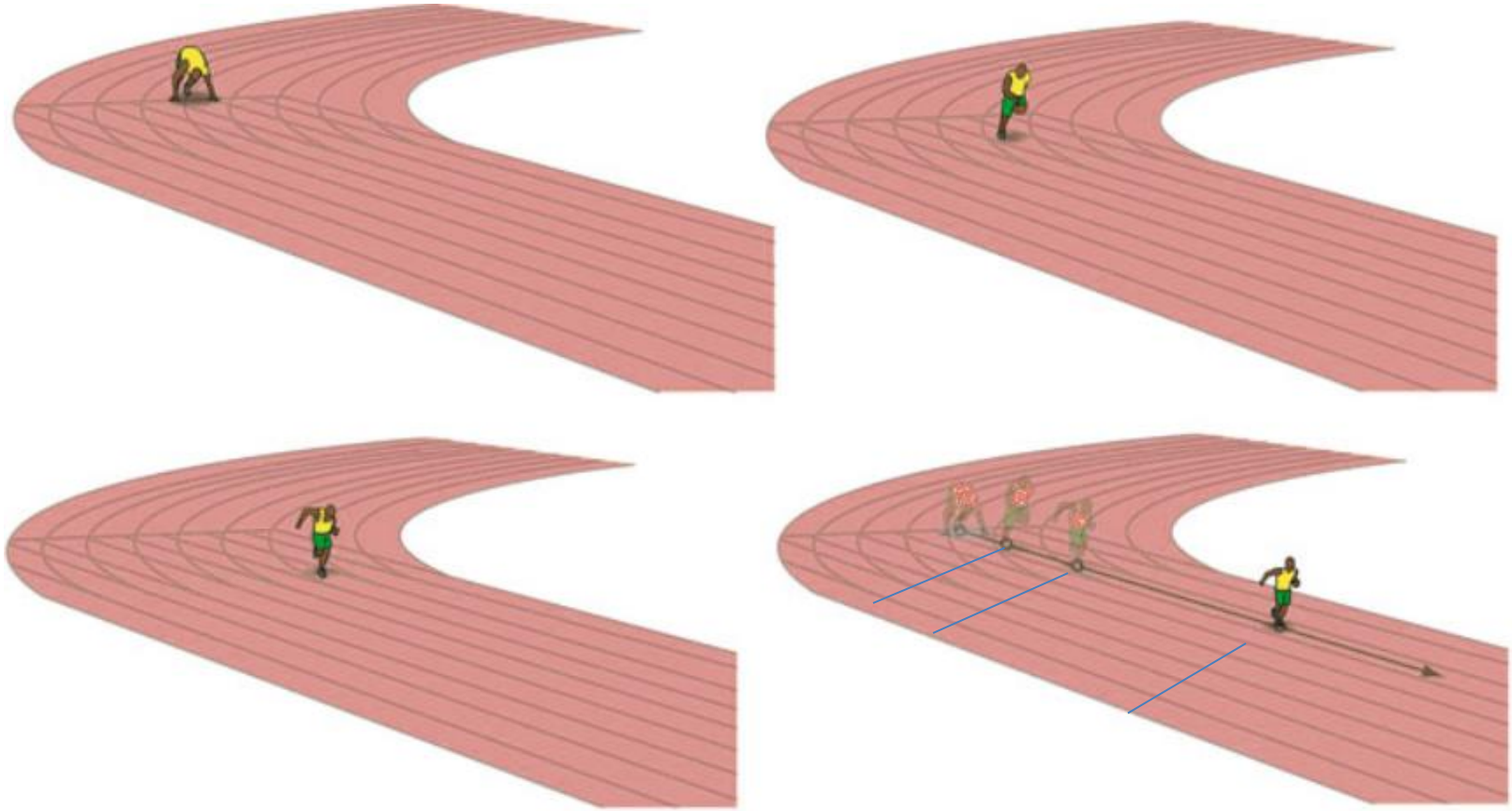
#2

Serdar ARITAN

Biomechanics Research Group,
Faculty of Sports Sciences, and
Department of Computer Graphics
Hacettepe University, Ankara, Turkey

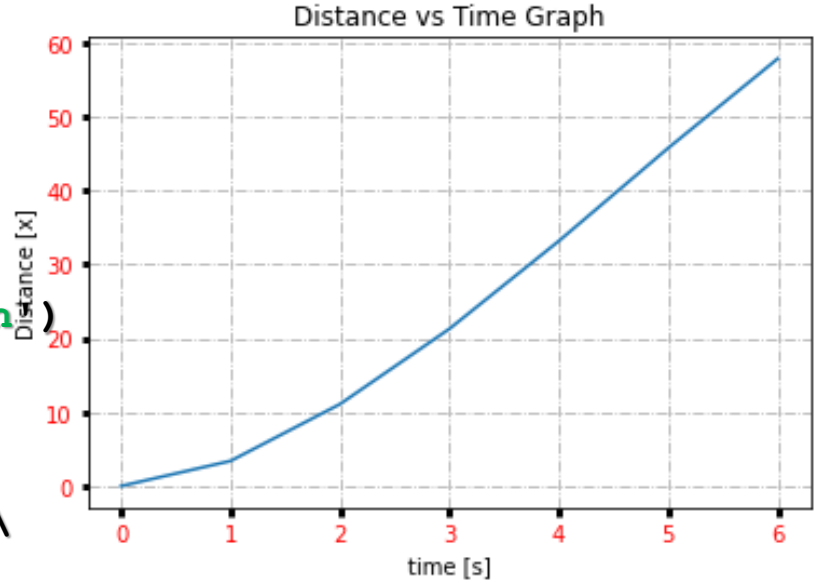
PHYSICS in COMPUTER ANIMATIONS and GAMES

PHYSICS in COMPUTER ANIMATIONS and GAMES



PHYSICS in COMPUTER ANIMATIONS and GAMES

```
import matplotlib.pyplot as plt
import numpy as np
# Data for plotting
t = np.array([0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0])
x = np.array([0.0, 3.4, 11.1, 21.3, 33.2, 45.8, 57.9])

fig, ax = plt.subplots()
ax.plot(t, x)

ax.set(xlabel='time [s]',\
       ylabel='Distance [x]',
       title='Distance vs Time Graph')

ax.grid(True, linestyle='-.')
ax.tick_params(labelcolor='r', \
                  labelsize='medium', \
                  width=3)

plt.show()
```



Average Velocity

Vectors and PyGame

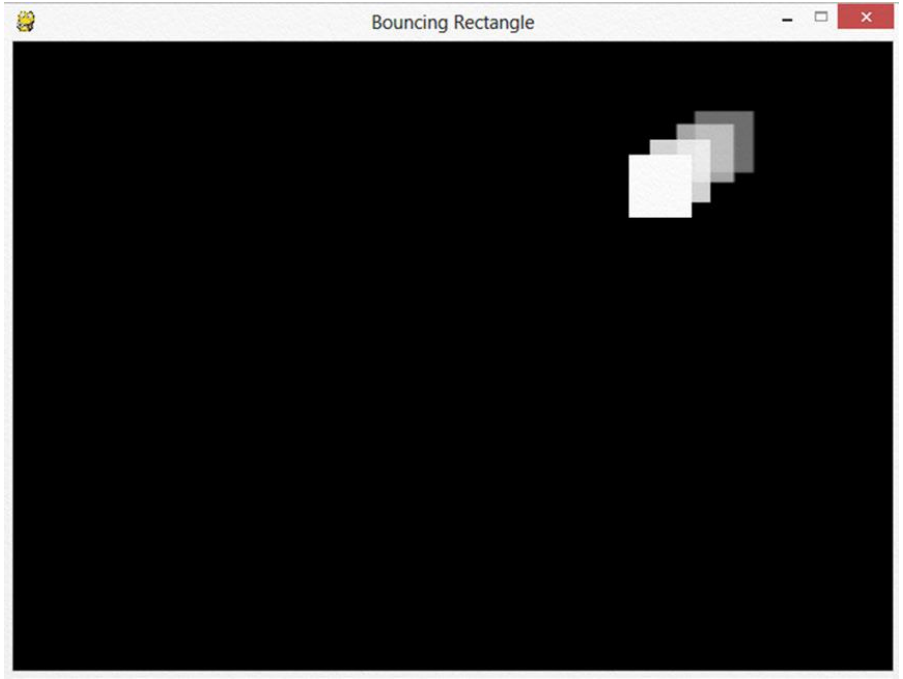
#3

Serdar ARITAN

Biomechanics Research Group,
Faculty of Sports Sciences, and
Department of Computer Graphics
Hacettepe University, Ankara, Turkey

Animation

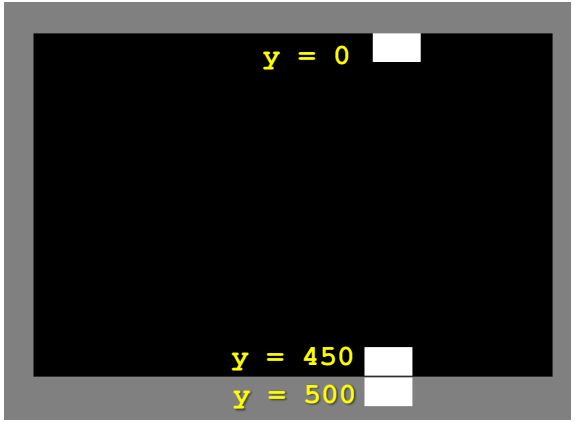
We will put together a program to bounce a white rectangle around a screen with a black background.



Animation

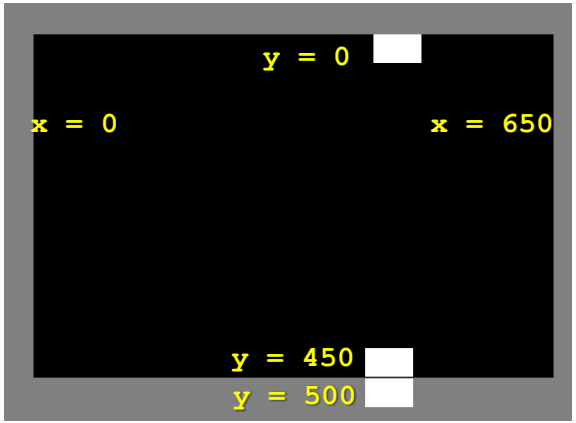
Once the box hits the edge of the screen it will keep going. Nothing makes the rectangle bounce off the edge of the screen. To reverse the direction so the rectangle travels towards the right, **rect_change_y** needs to change from 5 to -5 once the rectangle gets to the bottom side of the screen. The rectangle is at the bottom when **rect_y** is greater than the height of the screen. The code below can do the check and reverse the direction:

```
# Bounce the rectangle if needed
if rect_y > 450:
    rect_change_y = rect_change_y * -1
```

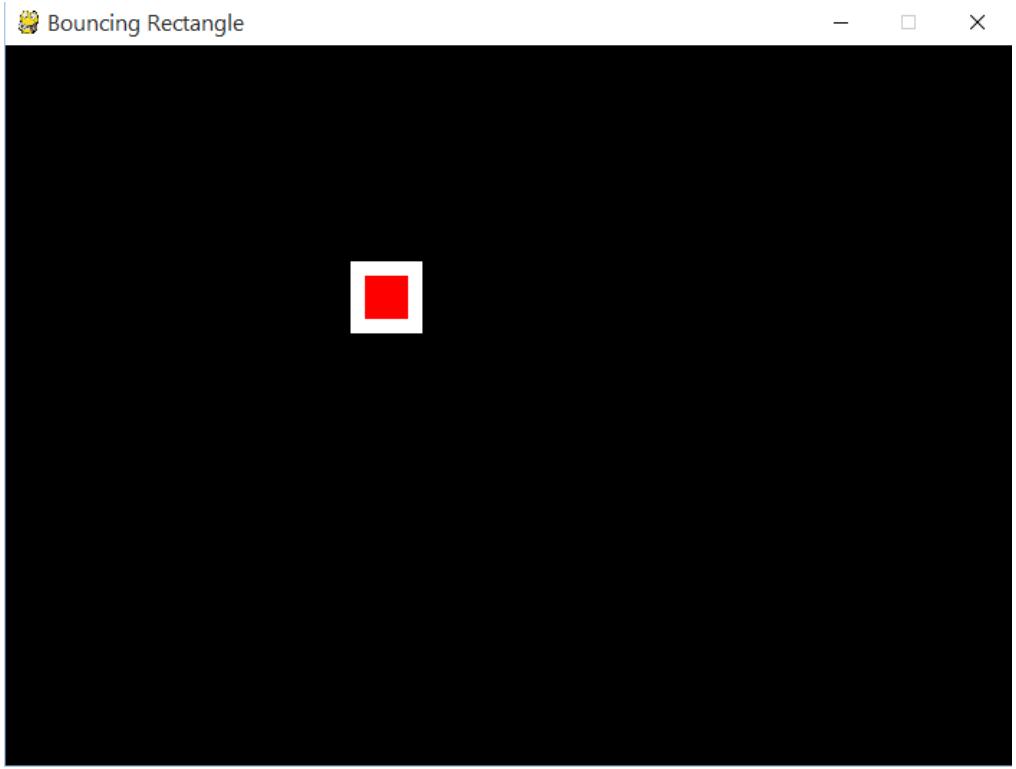


Animation

```
# Bounce the rectangle if needed
if rect_y > 450 or rect_y < 0:
    rect_change_y = rect_change_y * -1
if rect_x > 650 or rect_x < 0:
    rect_change_x = rect_change_x * -1
```




```
# Draw a red rectangle inside the white one
pygame.draw.rect(screen, WHITE, [rect_x, rect_y, 50, 50])
pygame.draw.rect(screen, RED, [rect_x + 10, rect_y + 10 ,30, 30])
```

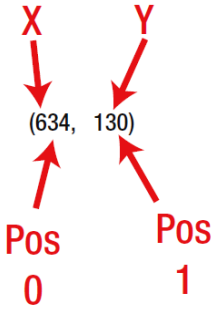


bouncing_rectangle.py

Mouse

It takes one line of code to get the mouse coordinates.

```
pos = pygame.mouse.get_pos()
```




```
# Game logic
pos = pygame.mouse.get_pos()
x = pos[0]
y = pos[1]
```

The mouse can be hidden by using the following code right before the main program loop

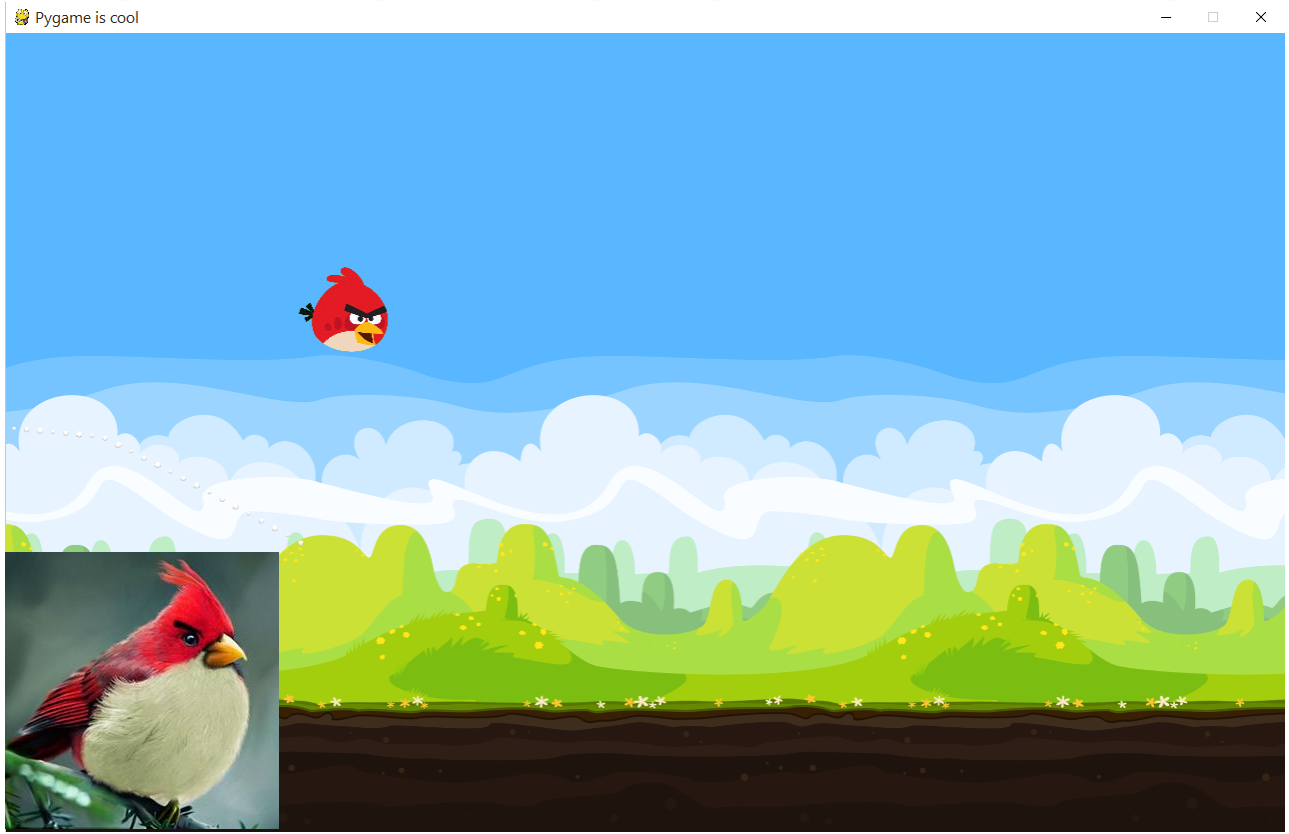
```
# Hide the mouse cursor
pygame.mouse.set_visible(False)
```

Game Controller

A joystick will return two floating-point values. If the joystick is perfectly centered it will return (0, 0).

Game Controller

Background Image, moving Image and Sounds in Pygame



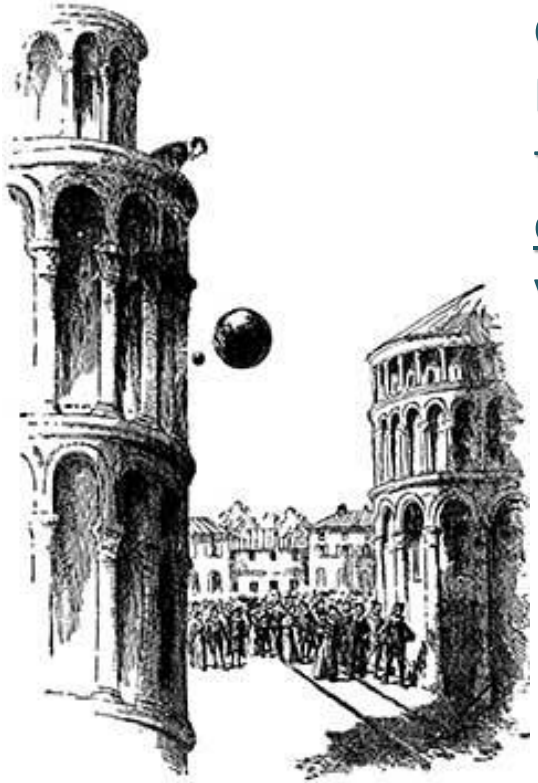
Motion in Two Dimensions and Gravity

#4

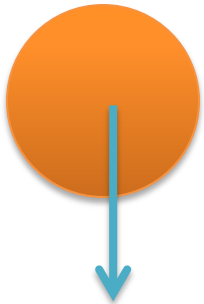
Serdar ARITAN

Biomechanics Research Group,
Faculty of Sports Sciences, and
Department of Computer Graphics
Hacettepe University, Ankara, Turkey

Free Fall of an Object: An Experiment by Galileo

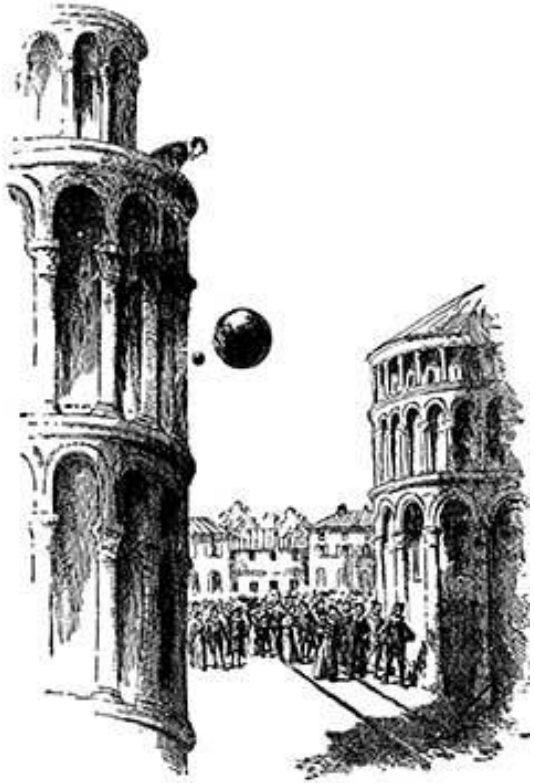


Galileo didn't know calculus (because Newton and Leibniz hadn't discovered it yet) so he couldn't derive the equation mathematically. Since we do know calculus (**SymPy**) we know that acceleration is the variation of velocity with time.



$$\vec{F}_{gravity} = m\vec{g}.$$

Free Fall of an Object: Analytical Solution



We assume no drag/air resistance

$$v_t = v_0 - \vec{g}t$$

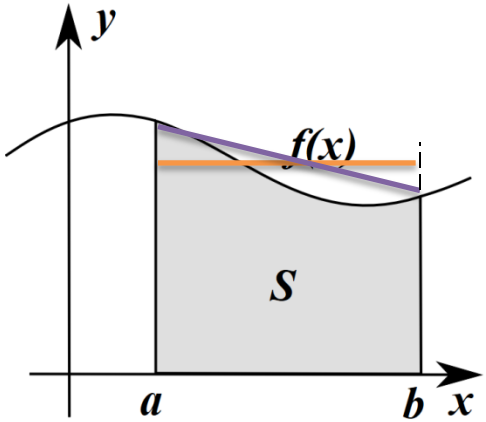
$$\frac{dx}{dt} = v_0 - \vec{g}t$$

$$\int \frac{dx}{dt} dt = \int [v_0 - \vec{g}t] dt$$

$$x_t - x_0 = v_0 t - \frac{1}{2}\vec{g}t^2$$

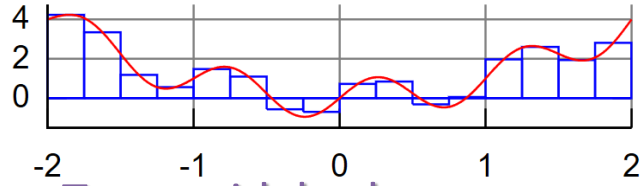
$$x_t = x_0 + v_0 t - \frac{1}{2}\vec{g}t^2$$

Numerical integration



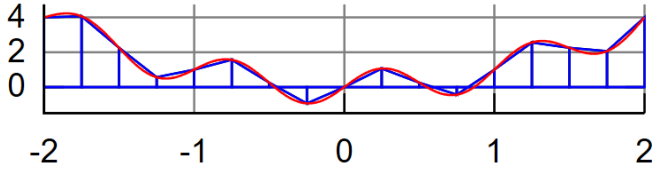
$$\int_a^b f(x)dx$$

Rectangle rule



$$\int_a^b f(x)dx \approx (b-a) f\left(\frac{a+b}{2}\right)$$

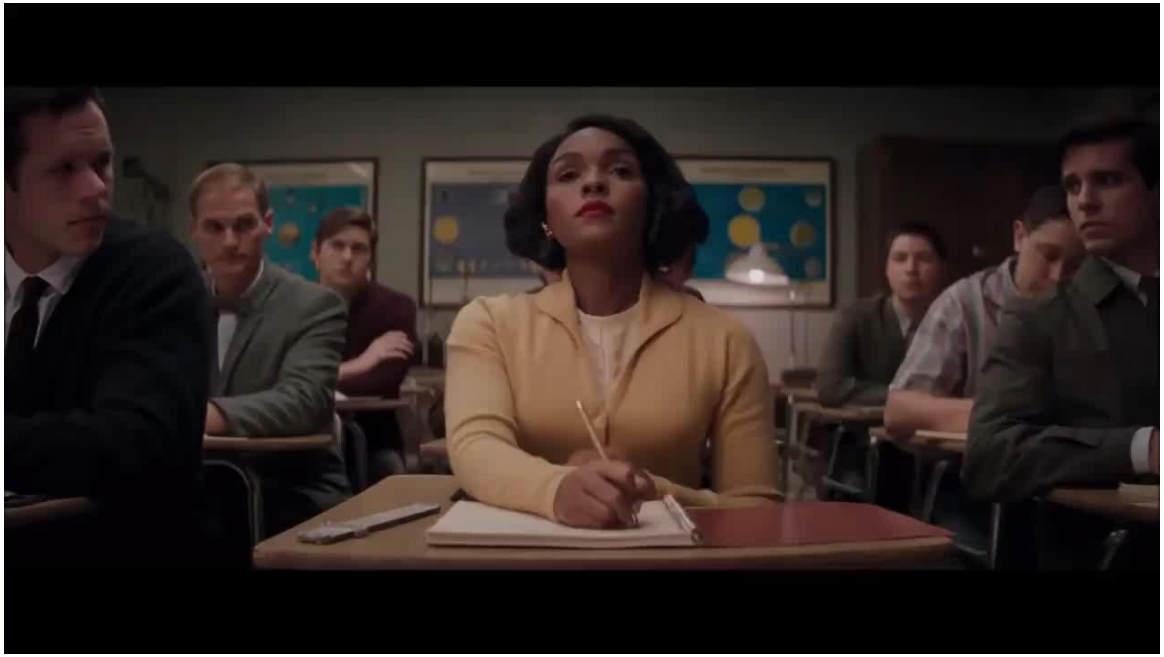
Trapezoidal rule



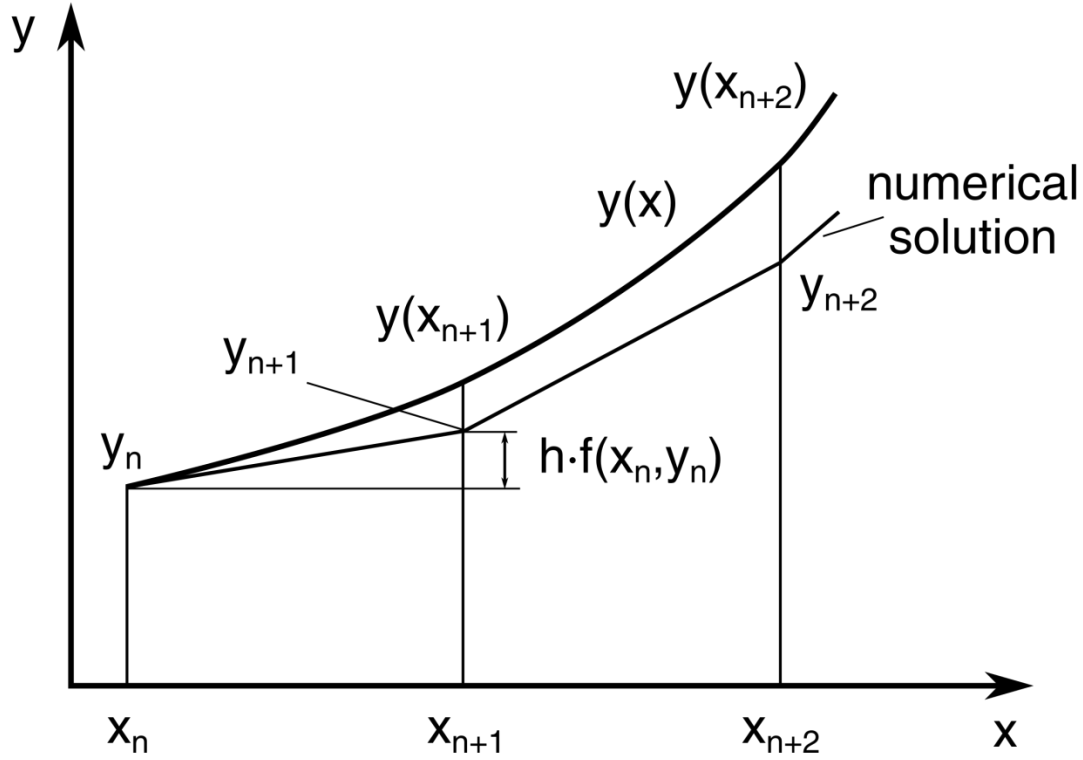
$$\int_a^b f(x)dx \approx (b-a)\frac{f(a)+f(b)}{2}$$

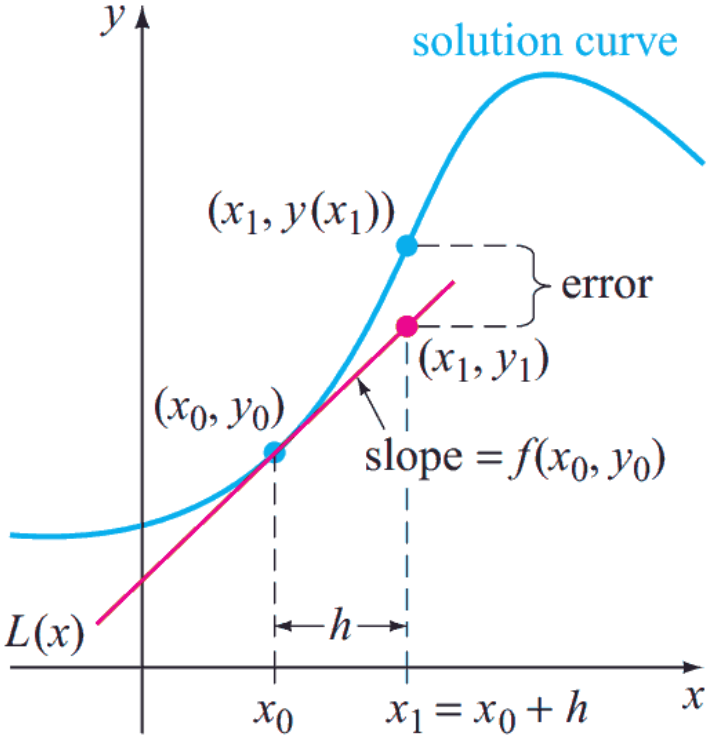
Euler method

Euler method : It’s ancient ….. but it's works



Euler method





Forces in One and Two Dimensions

#5

Serdar ARITAN

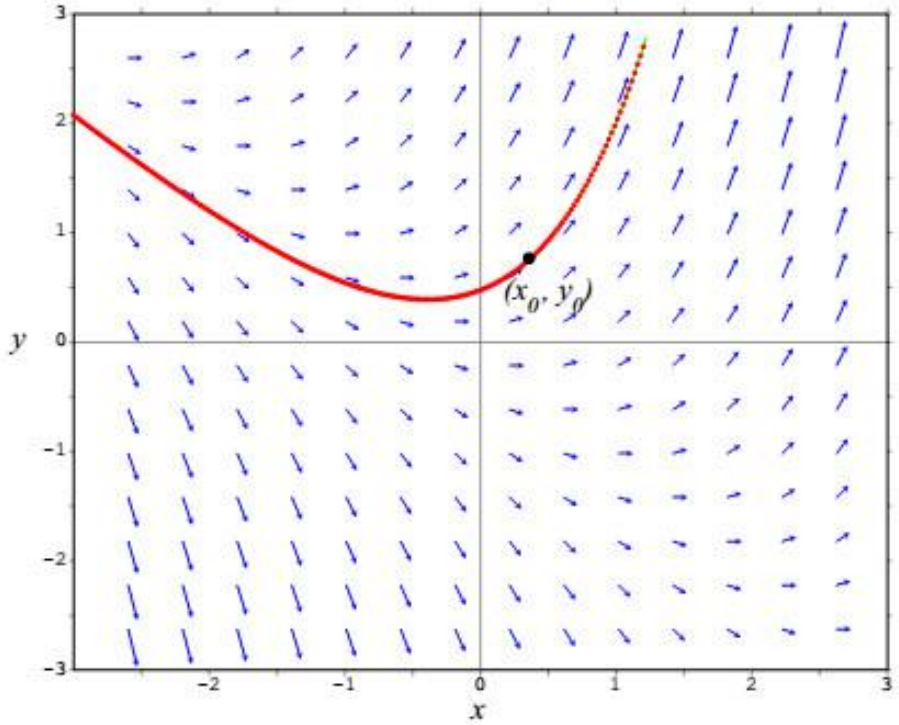
Biomechanics Research Group,
Faculty of Sports Sciences, and
Department of Computer Graphics
Hacettepe University, Ankara, Turkey

The vector field plot for the differential equation

$$\frac{dy}{dx} = y + x$$

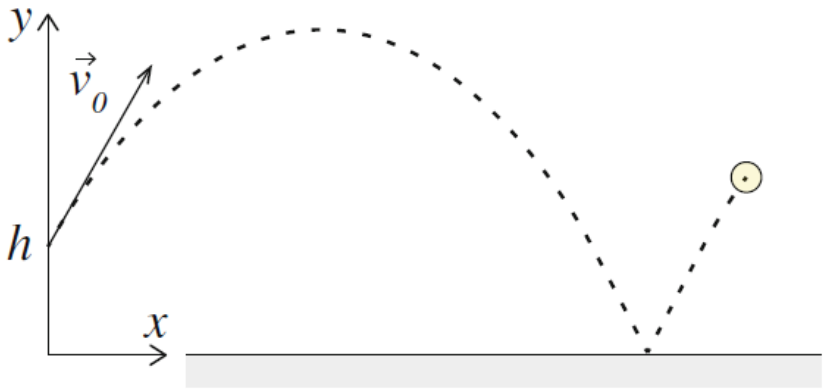
$$x_0 = -3$$

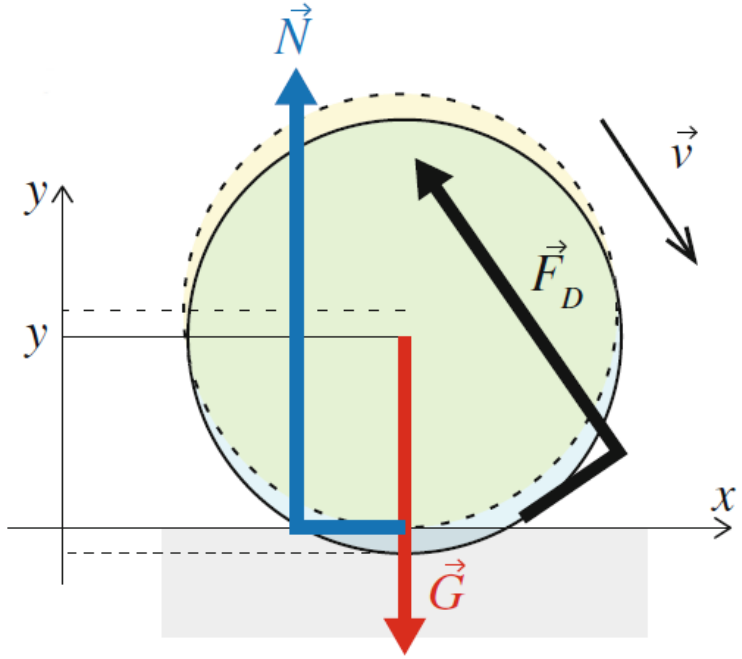
$$y_0 = 2$$



Initial-Value Problem :
An IVP is a differential equation together with a place for a solution to start.

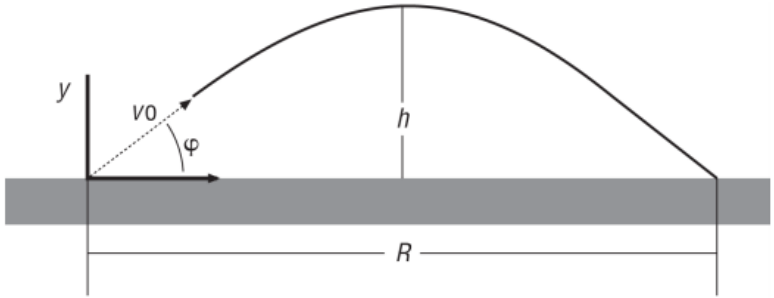
The normal force from the floor on the ball is represented by a spring force. This is a strong simplification of the actual deformation process occurring at the contact between the ball and the floor due to the deformation of both the ball and the floor.

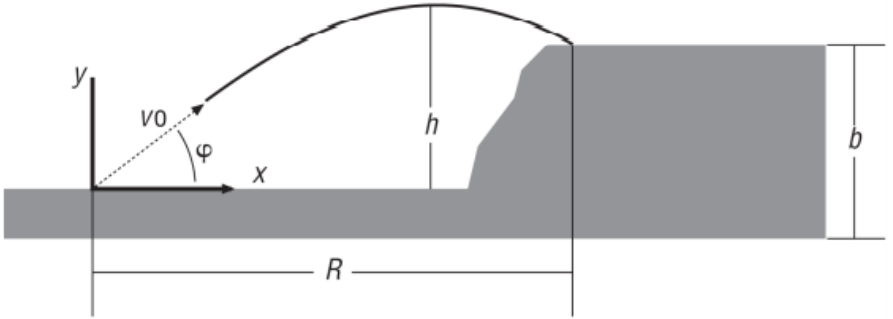


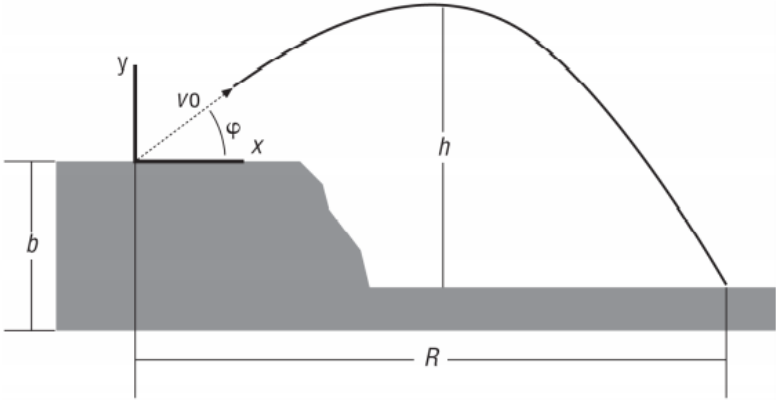


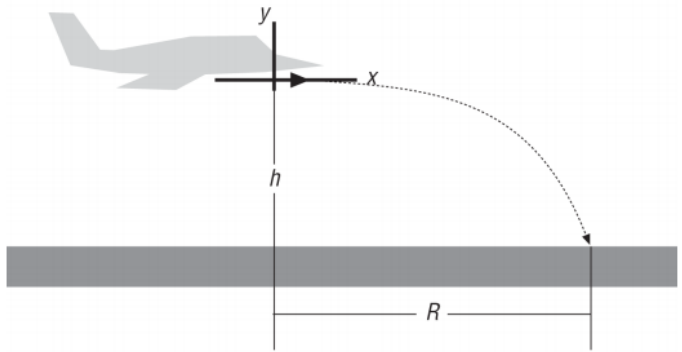
Values of Air Density As a Function of Altitude

Altitude (*m*)	Altitude (*ft*)	Density (kg/m^3)
0.0	0.0	1.225
305	1000	1.189
610	2000	1.154
914	3000	1.121
1219	4000	1.088
1524	5000	1.055
2134	7000	0.992
3048	10,000	0.905

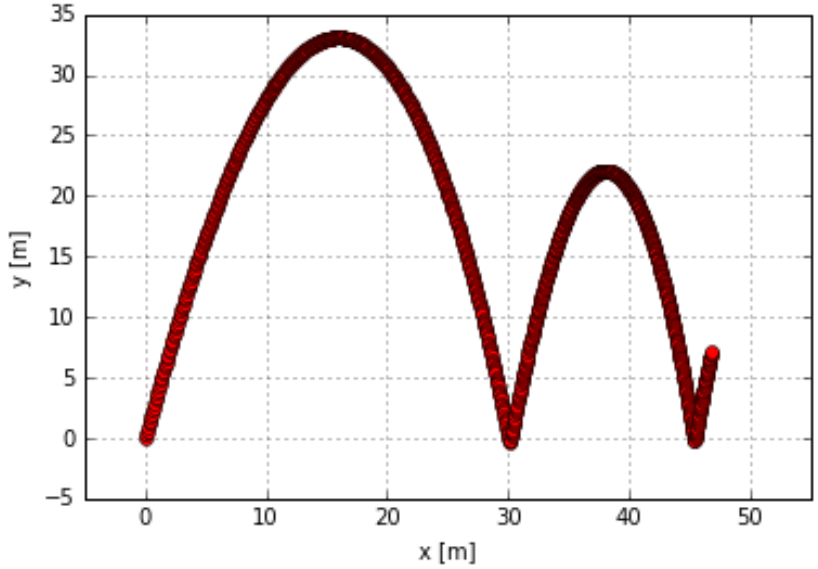

y
vo
φ
h
R


y
vo
φ
x
h
b
R


y
vo
φ
x
h
b
R


y
x
h
R

```
# Plotting the results,
fig, ax = plt.subplots()
ax.plot(r[:,0],r[:,1], 'ro')
ax.axis([0, 50, 0, 50]), ax.axis('equal')
ax.set_xlabel('x [m]'), ax.set_ylabel('y [m]'), ax.grid(True)
```



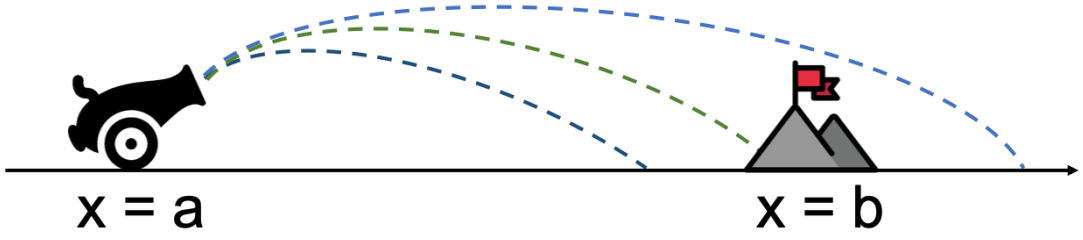
PHYSICS in COMPUTER ANIMATIONS and GAMES

Flock of Birds with the mass and the diameters

mass	12	8	7	5	5	4	4	2	1
size	Large	Large Medium	Large Medium	Medium	Medium	Medium	Medium Small	Small	Small

Boundary Value Problems - The Shooting Methods

The shooting methods are developed with the goal of transforming the ODE boundary value problems to an equivalent **initial value problems** (**IVP**), then we can solve it using the methods that we learned. In the IVP, we can start at **the initial value** and **march forward to get the solution**. But this method is not working for the boundary value problems, because there are not enough initial value conditions to solve the ODE to get a unique solution. Therefore, the shooting methods was developed to overcome this difficulty.



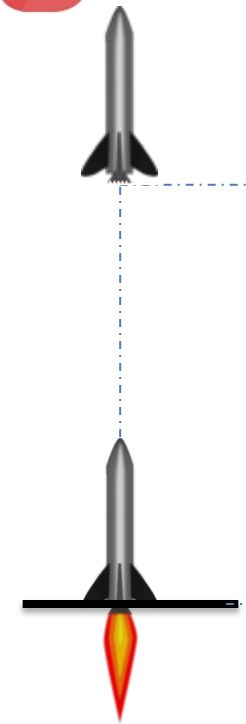
Forces in One and Two Dimensions

İnitial Value Problems and Boundary Value Problems

#6

Serdar ARITAN

Biomechanics Research Group,
Faculty of Sports Sciences, and
Department of Computer Graphics
Hacettepe University, Ankara, Turkey



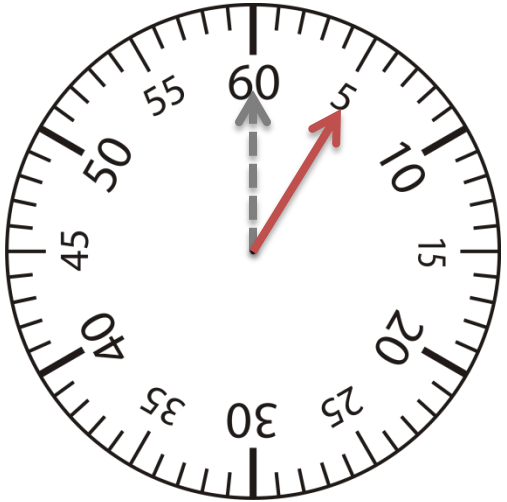
y(5)=50 m

y(0)=0 m

Equation of Motion

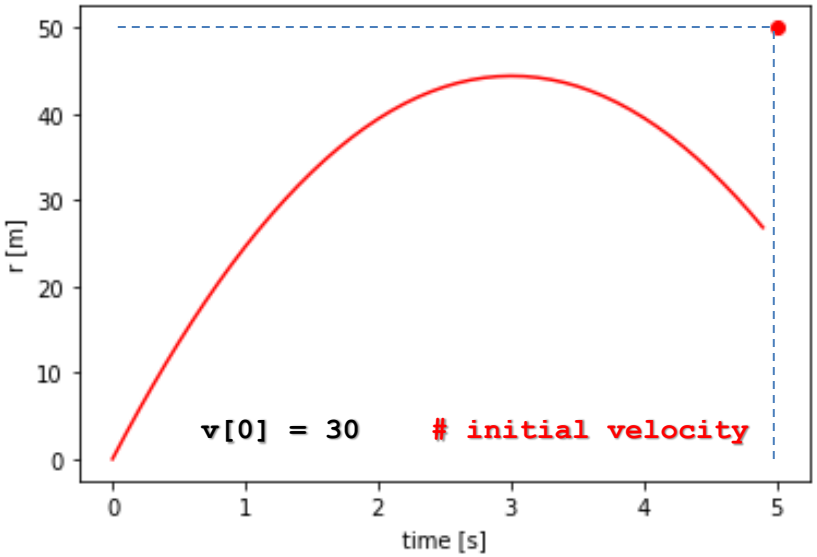
$$\frac{d^2y}{dt^2} = -g$$

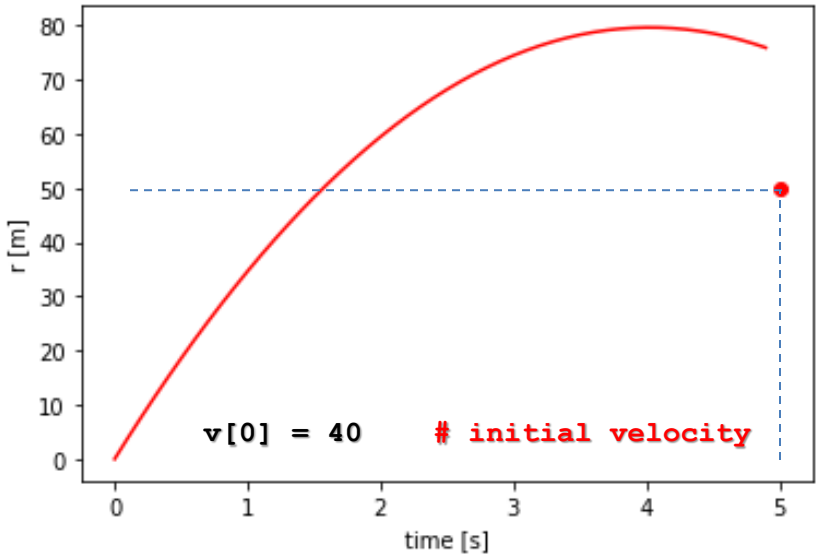
V_0= ?



boundary conditions are: y(0)=0 and y(5)=50

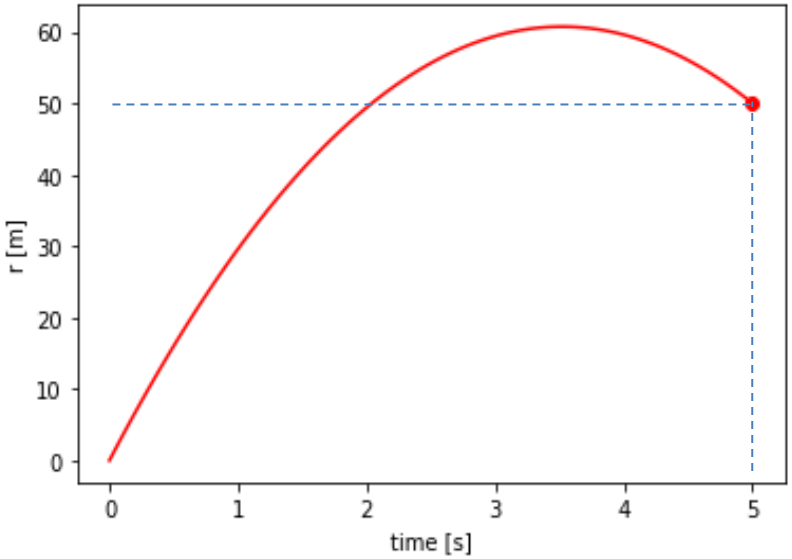
PHYSICS in COMPUTER ANIMATIONS and GAMES





PHYSICS in COMPUTER ANIMATIONS and GAMES

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

f = lambda t, s: np.dot(np.array([[0,1],[0,-9.8/s[1]]]), s)

t_span = np.linspace(0, 5, 100)
y0 = 0
v0 = 34.49999999999999
t_eval = np.linspace(0, 5, 100)
sol = solve_ivp(f, [0, 5], \
                [y0, v0], t_eval = t_eval)

# Position Plotting
fig, ax = plt.subplots()
ax.plot(sol.t, sol.y[0], '-r')
ax.plot(5, 50, 'ro')
ax.set_xlabel('time [s]')
ax.set_ylabel('r [m]')
plt.show()
```

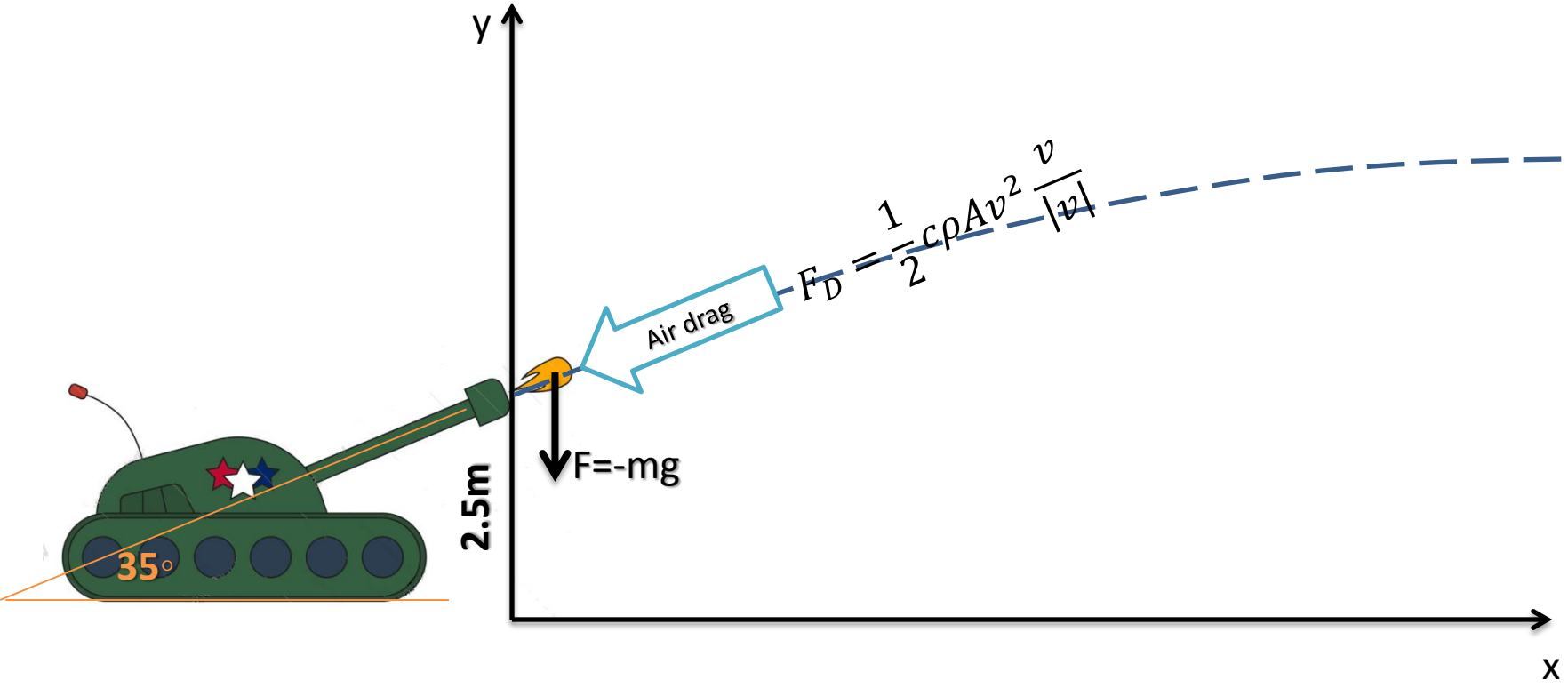


PHYSICS in COMPUTER ANIMATIONS and GAMES

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

f = lambda t, s: np.dot(np.array([[0,1],[0,-9.8/s[1]]]), s)

y0 = 0
t_eval = np.linspace(0, 5, 100)

def objective(v0):
    sol = solve_ivp(F, [0, 5], \
            [y0, v0], t_eval = t_eval)
    y = sol.y[0]
    return y[-1] - 50

for v0_guess in range(1, 100, 10):
    v0, = fsolve(objective, v0_guess)
    print(f"Init: {v0_guess}, Result: {v0}")
```

```
Init: 1, Result: 34.499999999999986
Init: 11, Result: 34.499999999999986
Init: 21, Result: 34.49999999999999
Init: 31, Result: 34.49999999999998
Init: 41, Result: 34.49999999999999
Init: 51, Result: 34.499999999999986
Init: 61, Result: 34.499999999999986
Init: 71, Result: 34.499999999999986
Init: 81, Result: 34.499999999999986
Init: 91, Result: 34.499999999999986
```

Note that changing the initial guesses does not change the result, which means that this method is stable

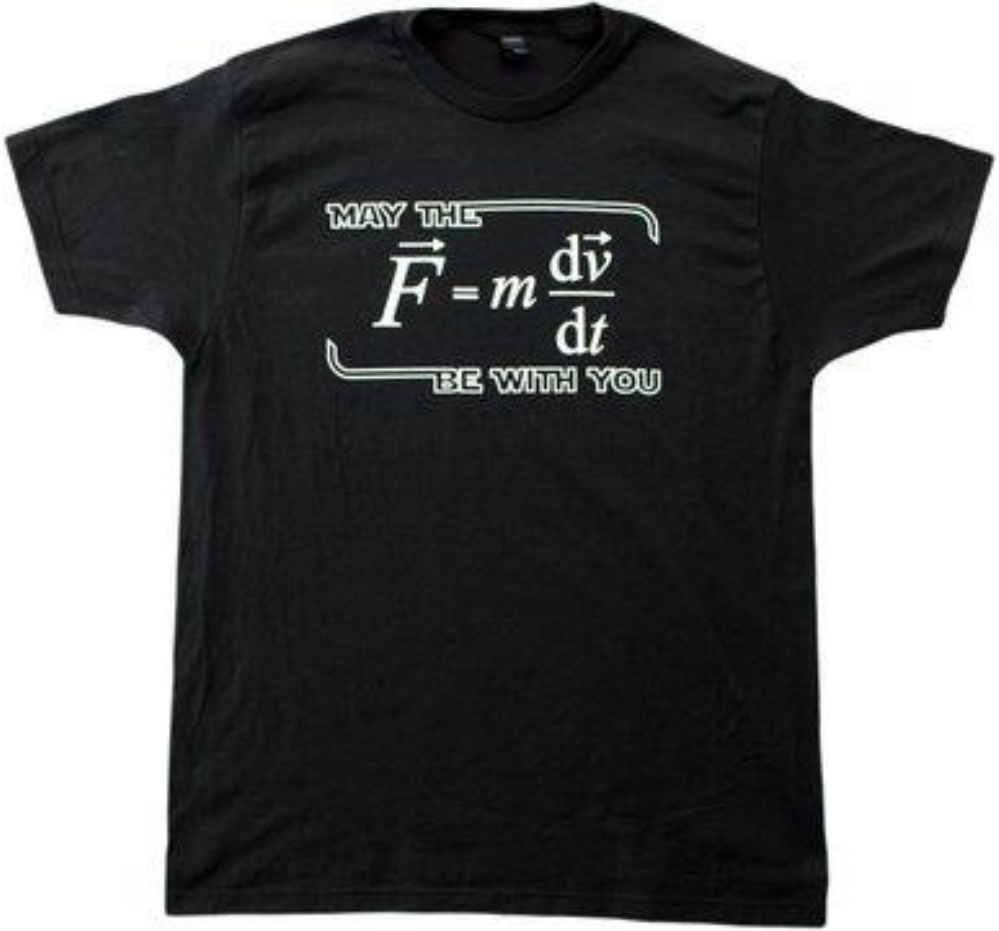

y
x
Air drag
$F_D = \frac{1}{2} c\rho A v^2 \frac{v}{|v|}$
F=-mg
2.5m
35°

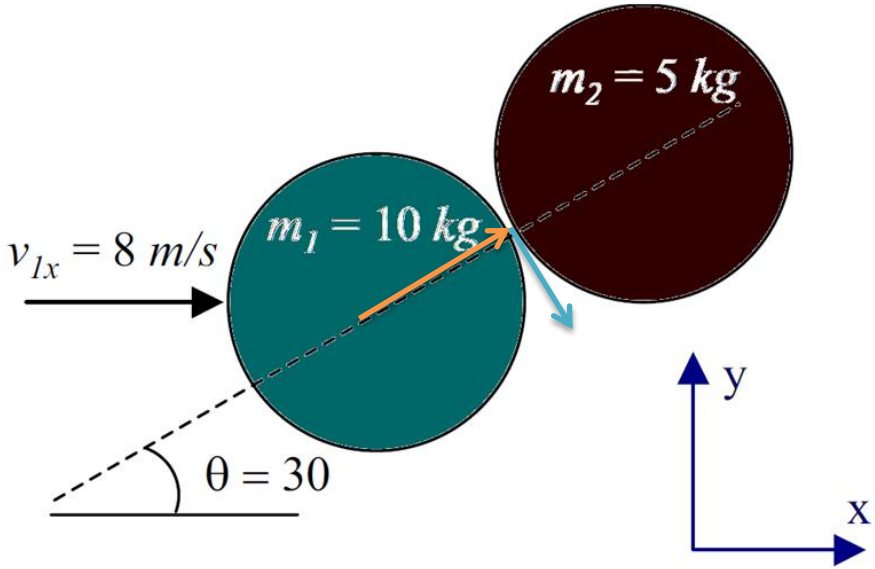
Linear Momentum, Impulse and Collision

#7

Serdar ARITAN

Biomechanics Research Group,
Faculty of Sports Sciences, and
Department of Computer Graphics
Hacettepe University, Ankara, Turkey


MAY THE
$\vec{F} = m\frac{d\vec{v}}{dt}$
BE WITH YOU



$$\begin{bmatrix} v_p \\ v_n \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix}$$

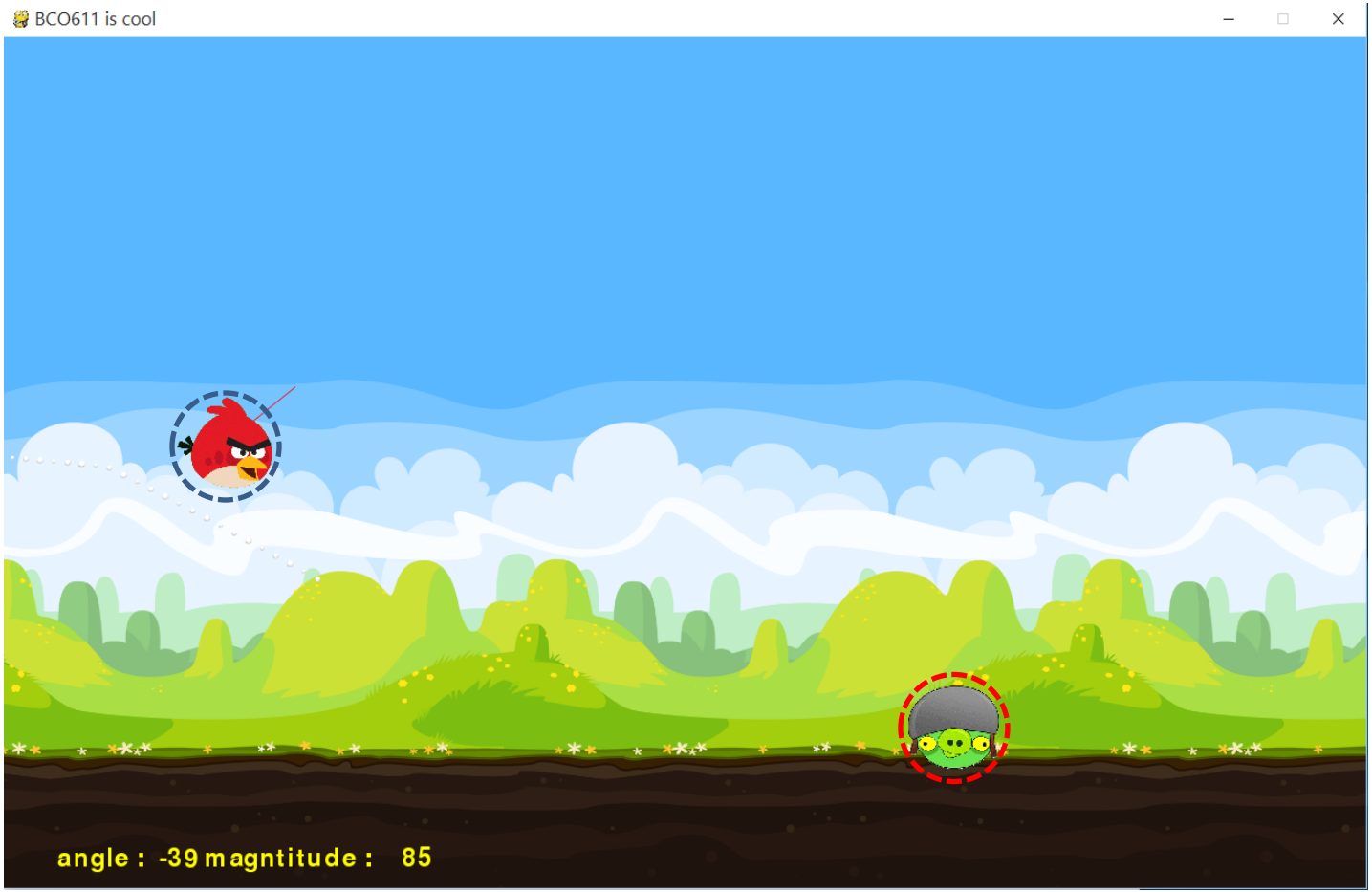
$$\begin{bmatrix} v_{1p} \\ v_{1n} \end{bmatrix} = \begin{bmatrix} \cos 30 & \sin 30 \\ -\sin 30 & \cos 30 \end{bmatrix} \begin{bmatrix} v_{1x} \\ v_{1y} \end{bmatrix}$$

$$\begin{bmatrix} v_{1p} \\ v_{1n} \end{bmatrix} = \begin{bmatrix} \cos 30 & \sin 30 \\ -\sin 30 & \cos 30 \end{bmatrix} \begin{bmatrix} 8 \\ 0 \end{bmatrix}$$

The velocity components parallel and normal to the line of action can be computed. Sphere 2 is initially not moving, so its velocity components will be zero.

$$v_{1p} = 8\cos 30 + 0\sin 30 = 6.93\ ms^{-1}$$

$$v_{1n} = -8\sin 30 + 0\cos 30 = -4.00\ ms^{-1}$$


BCO611 is cool
angle : -39 magntitude : 85

$$\theta = \operatorname{atan}\left(\frac{|y_2 - y_1|}{|x_2 - x_1|}\right)$$

$$a = r_2 \frac{|x_2 - x_1|}{r_1 + r_2} \qquad b = r_2 \frac{|y_2 - y_1|}{r_1 + r_2}$$

$$(x_c, y_c) = (x_2 + a\,, y_2 + b)$$

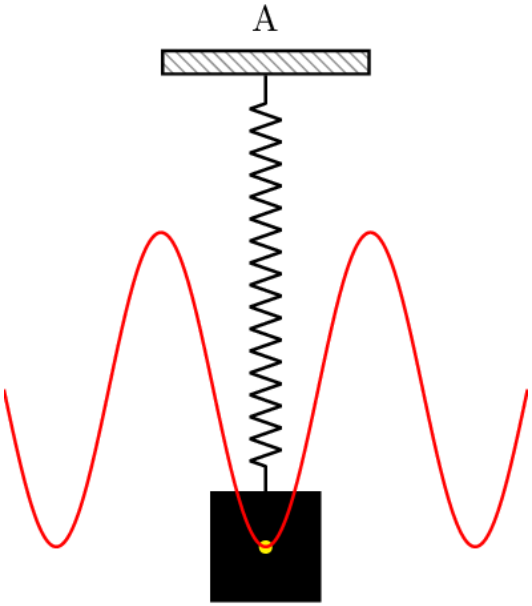
Spring and Damper in Modelling

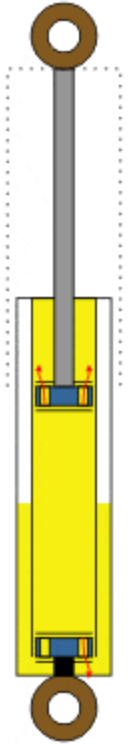
#8

Serdar ARITAN

Biomechanics Research Group,
Faculty of Sports Sciences, and
Department of Computer Graphics
Hacettepe University, Ankara, Turkey

Spring and Damper





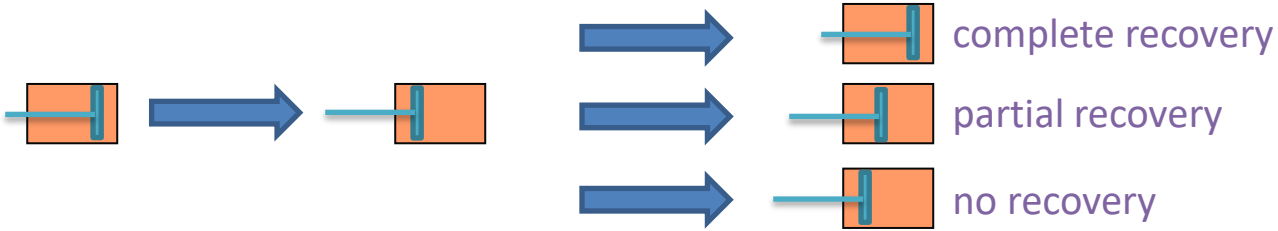
Spring and Damper

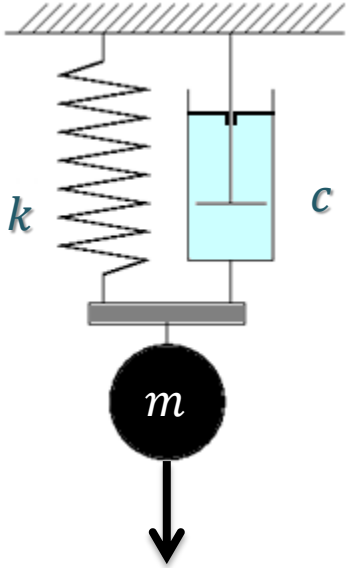
Spring: an ideal elastic element

Will immediately change and return to its original shape upon loading and unloading.

Damper: a viscous fluid

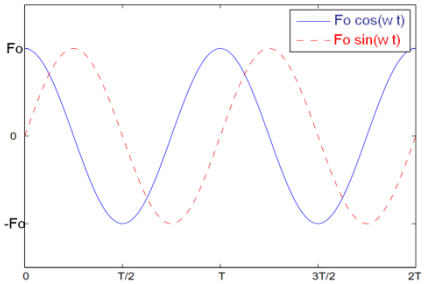
Will change its original shape upon loading, depending on time (and temperature).
May slowly return to or may not return to its original shape upon unloading.
partial or complete recovery





$$F(t) = F_0 \sin \omega t$$

External Forcing models the behavior of a system which has a time varying force acting on it. An example might be a suspended bridge subjected to wind loading.



External Forcing

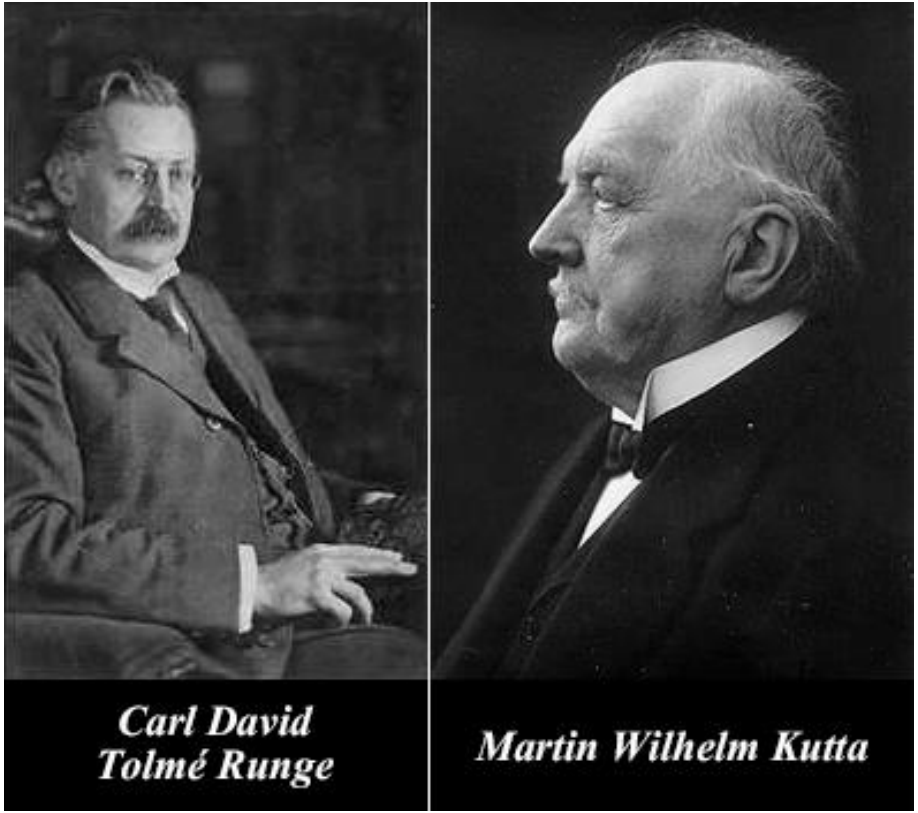
Viscoelastic Materials

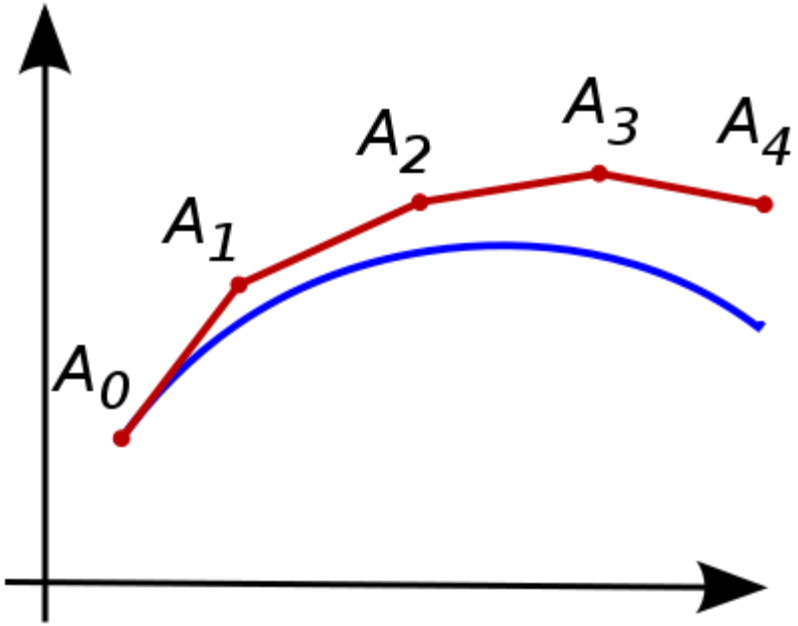
- Viscosity = The resistance to motion of a liquid
- Viscoelastic materials

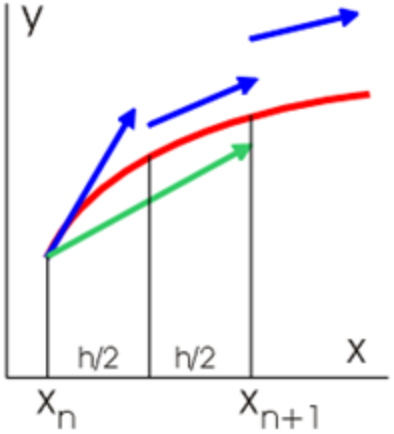
  Properties = Elastic solid + Fluid

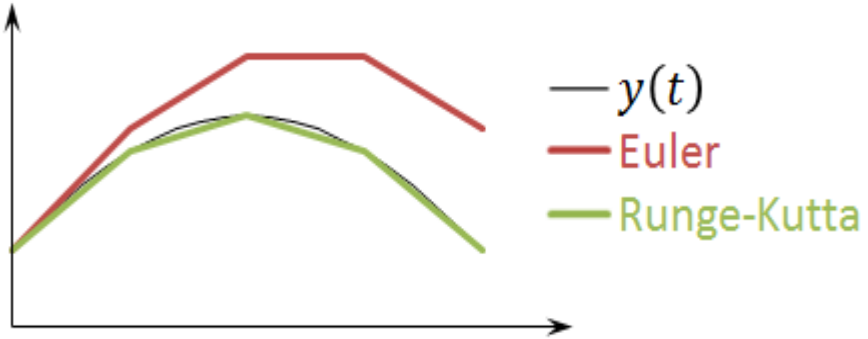
  = Materials that have mechanical properties dependent on time (loading rate or strain rate) and temperature

Carl David Tolme Runge – Martin Wilhelm Kutta




A_0
A_1
A_2
A_3
A_4


y
h/2
h/2
x
x_n
x_{n+1}


$y(t)$
Euler
Runge-Kutta

Multiparticle Systems

#9

Serdar ARITAN

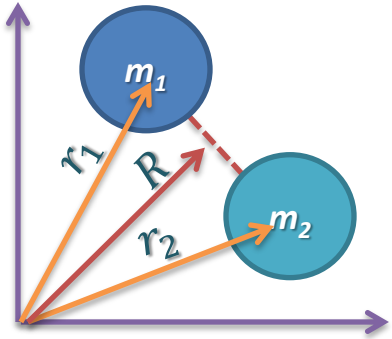
Biomechanics Research Group,
Faculty of Sports Sciences, and
Department of Computer Graphics
Hacettepe University, Ankara, Turkey

The Center of Mass

Why do we call the effective position R the center of mass? Let us start by address this in a two-particle system. The effective position, R, of the two-particle system is:

$$R = \frac{1}{M}\sum_i m_i r_i = \frac{1}{M}(m_1 r_1 + m_2 r_2)$$

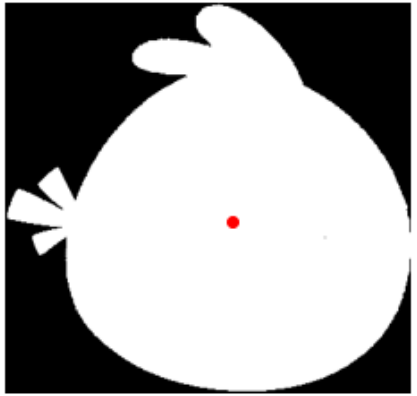
If the two masses are identical, we see that:



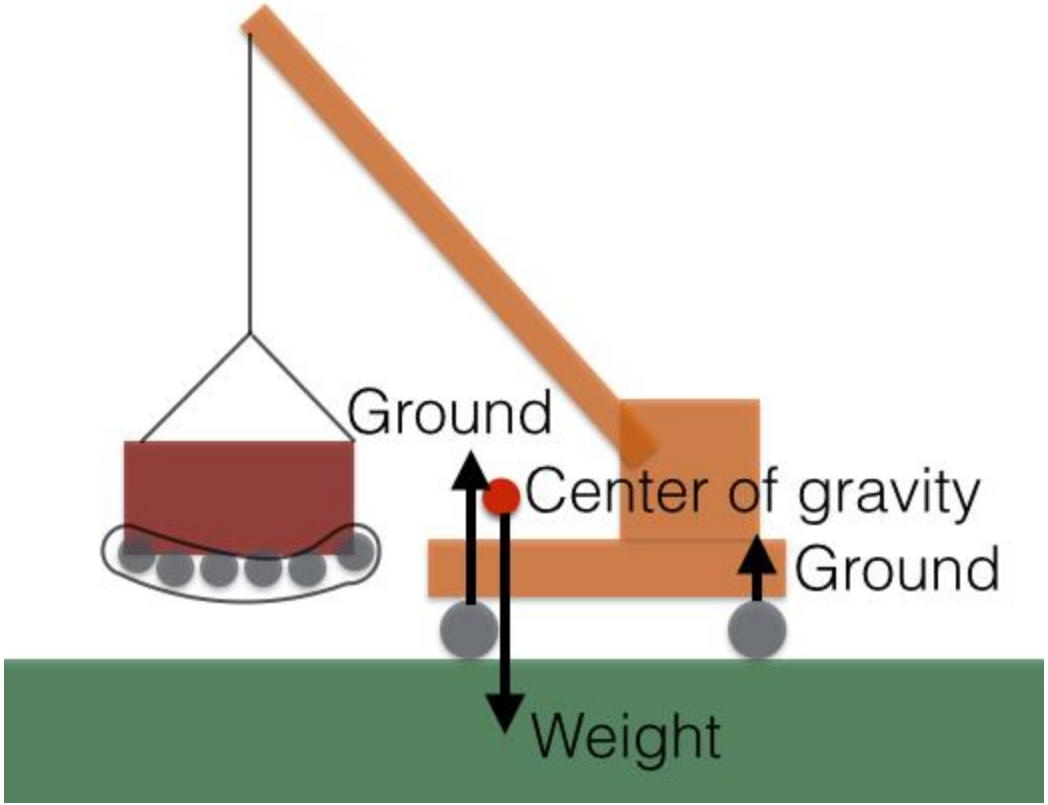
$$R = \frac{m_1 r_1 + m_2 r_2}{m_1 + m_2} = \frac{m r_1 + m r_2}{m + m} = \frac{1}{2}(r_1 + r_2)$$

Example: Center of Mass from Image Analysis

```
s = np.shape(Ibw)
x, y , m = 0, 0, 0

for iy in range(s[0]):
    for ix in range(s[1]):
        if Ibw[iy, ix]:
            x += ix
            y += iy
            m += 1

xcm, ycm = x/m, y/m
ax.plot(xcm, ycm,'ro')
plt.show()
```



Location of Center of Mass



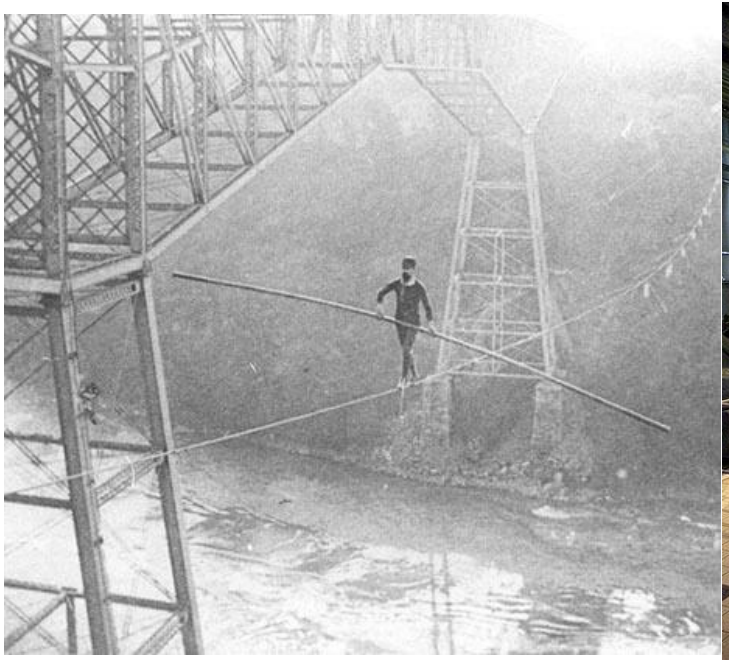
Rotational Motion

#10

Serdar ARITAN

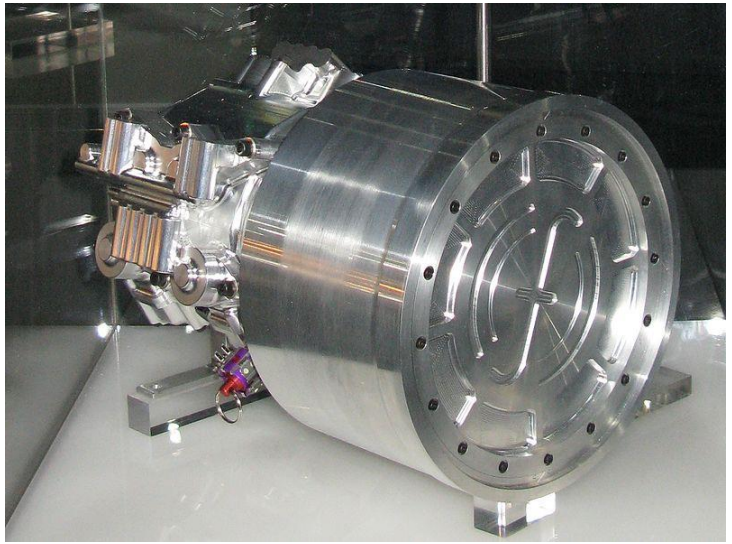
Biomechanics Research Group,
Faculty of Sports Sciences, and
Department of Computer Graphics
Hacettepe University, Ankara, Turkey

Moments of inertia



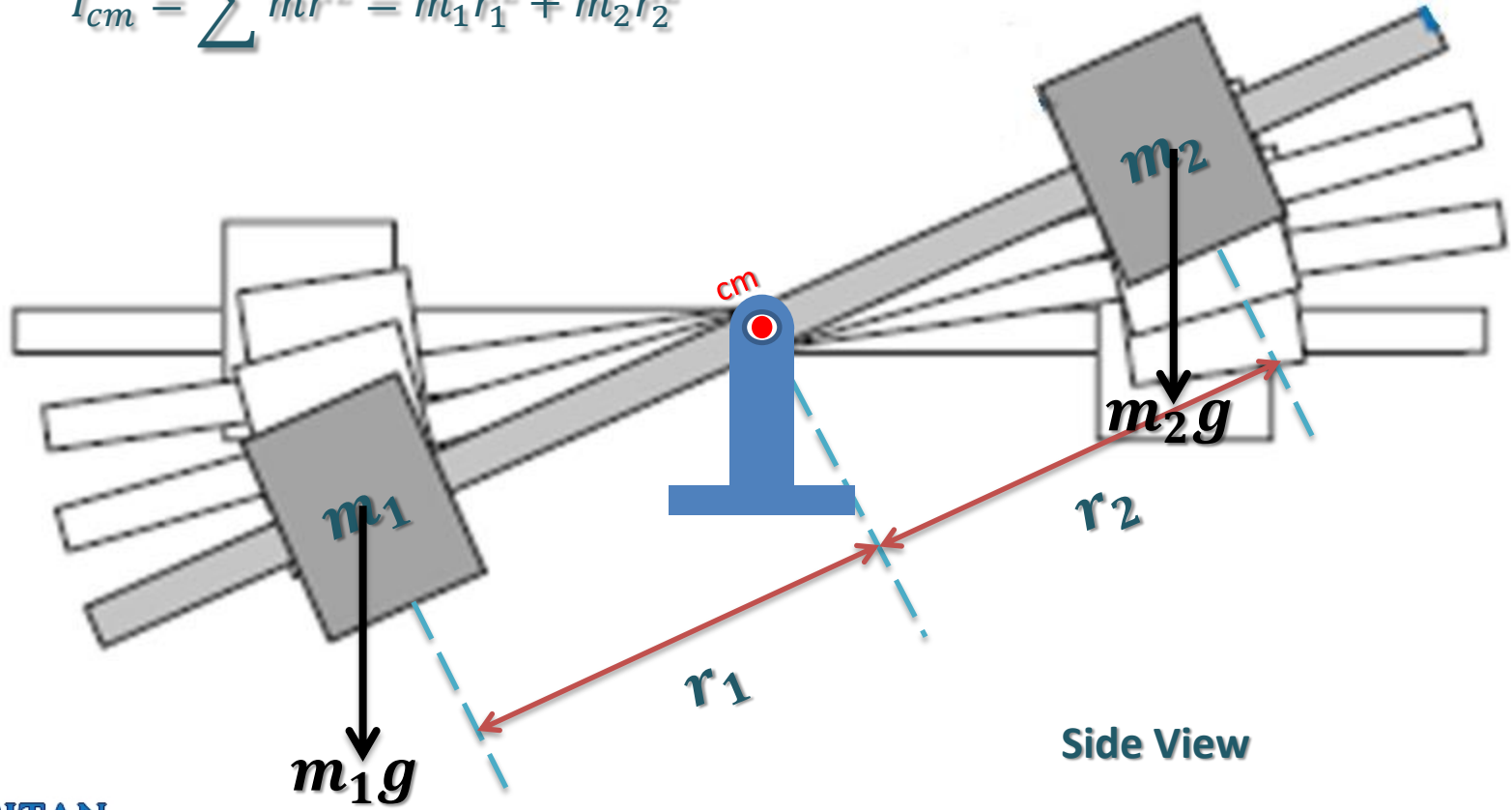
A flywheel is a mechanical device specifically designed to efficiently store rotational energy. Flywheels resist changes in rotational speed by their **moment of inertia.**

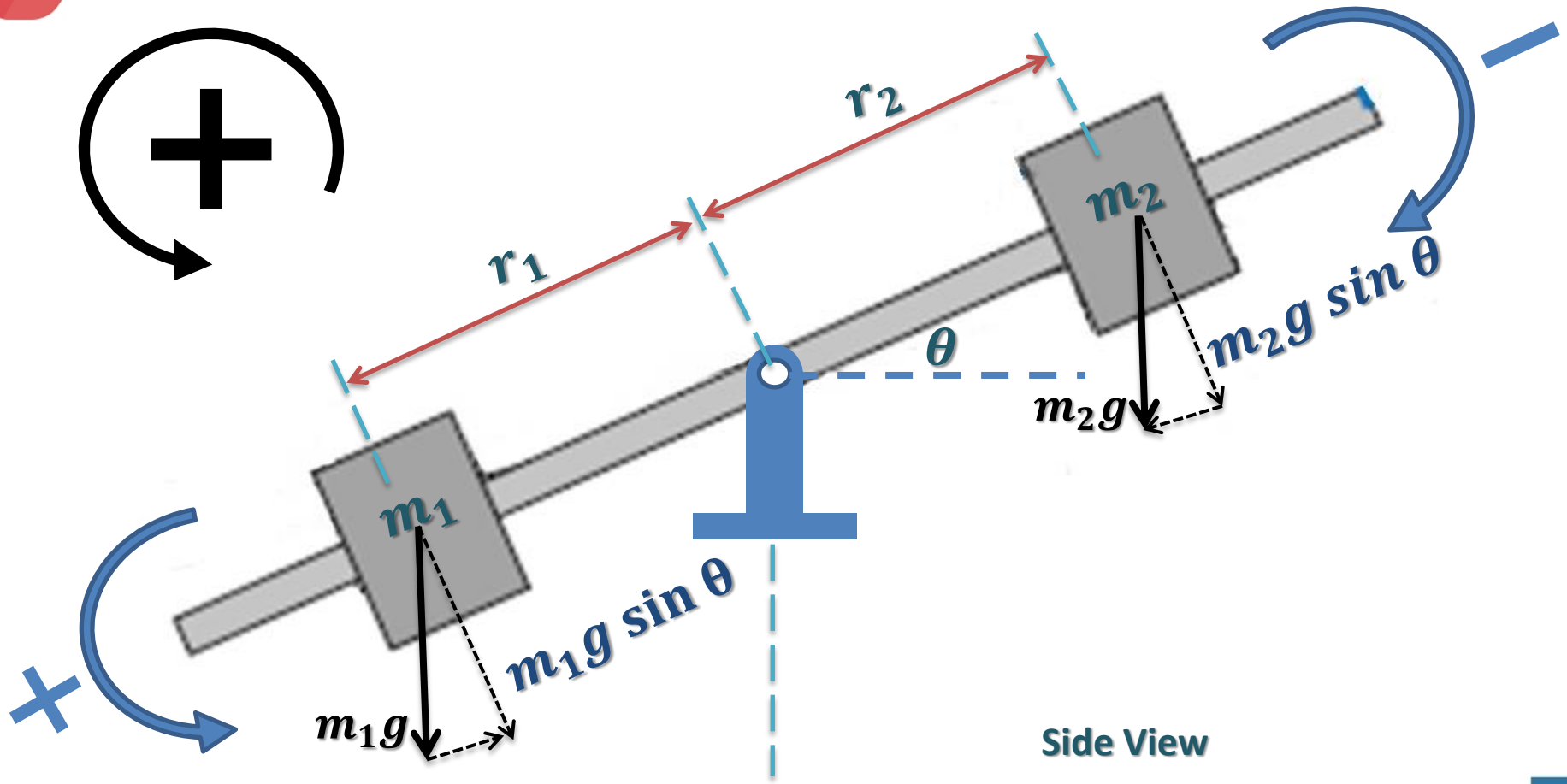
Moments of inertia

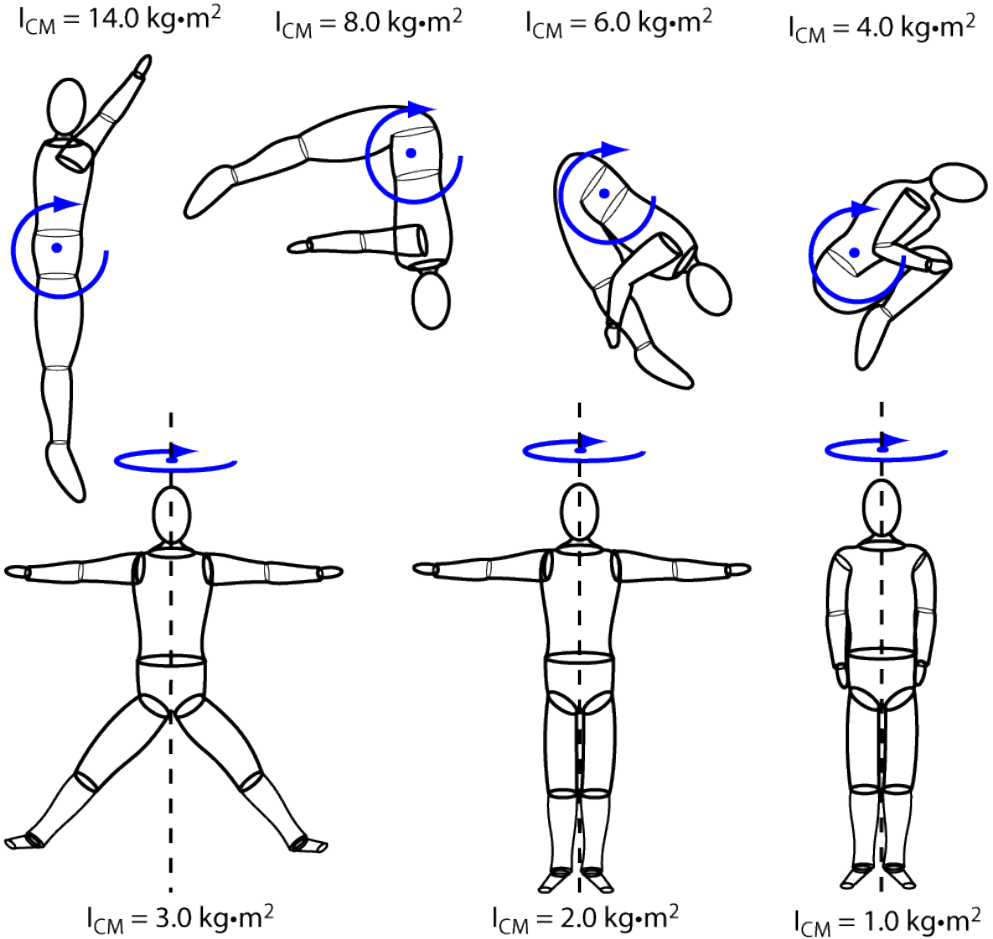


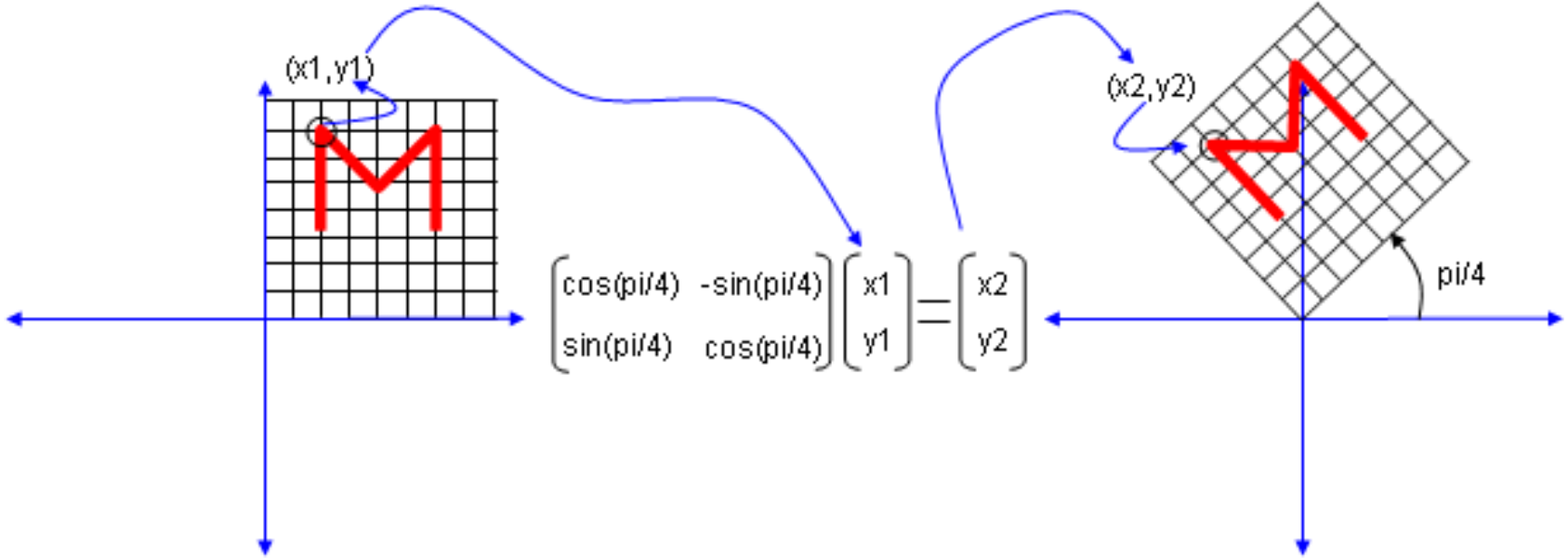
Using a continuously variable transmission (CVT), energy is recovered from the drive train during braking and stored in a flywheel. This stored energy is then used during acceleration by altering the ratio of the CVT.

$$I_{cm} = \sum mr^2 = m_1 r_1^2 + m_2 r_2^2$$




+
r2
r1
m2
θ
m2g sin θ
m2g
−
m1
m1g sin θ
m1g
+
Side View


I_{CM} = 14.0 kg•m²
I_{CM} = 8.0 kg•m²
I_{CM} = 6.0 kg•m²
I_{CM} = 4.0 kg•m²
I_{CM} = 3.0 kg•m²
I_{CM} = 2.0 kg•m²
I_{CM} = 1.0 kg•m²


(x1,y1)
(x2,y2)
pi/4
cos(pi/4) -sin(pi/4)
sin(pi/4) cos(pi/4)
x1
y1
x2
y2

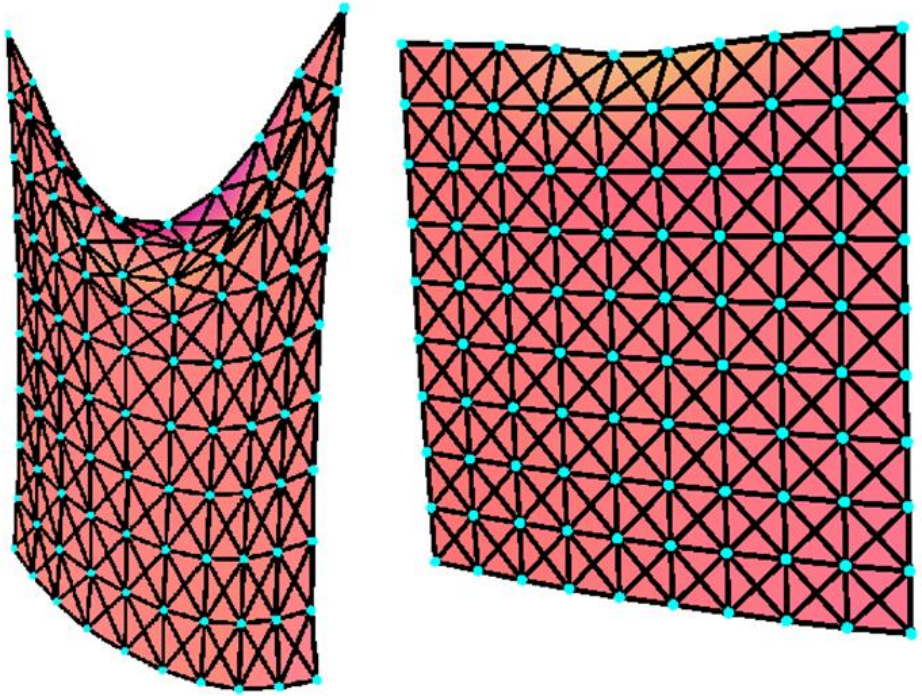
Cloth Simulation with Particle Mechanics

#11

Serdar ARITAN

Biomechanics Research Group,
Faculty of Sports Sciences, and
Department of Computer Graphics
Hacettepe University, Ankara, Turkey

Cloth Animation

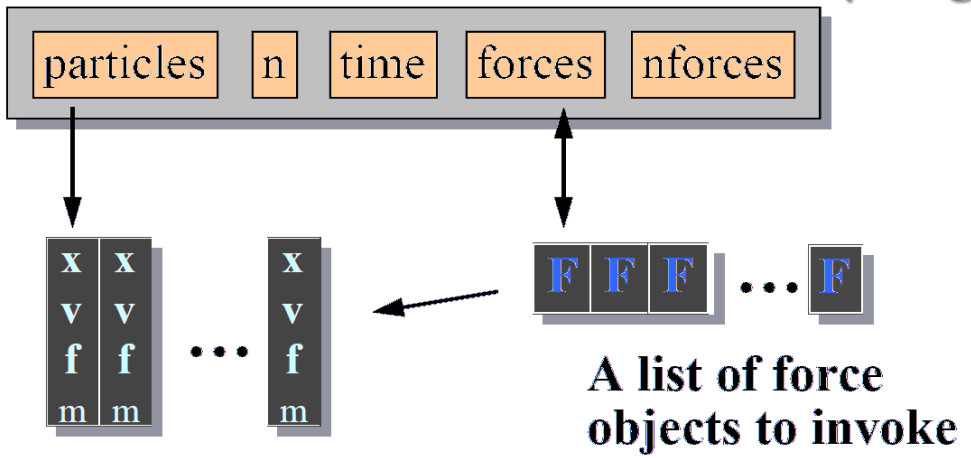


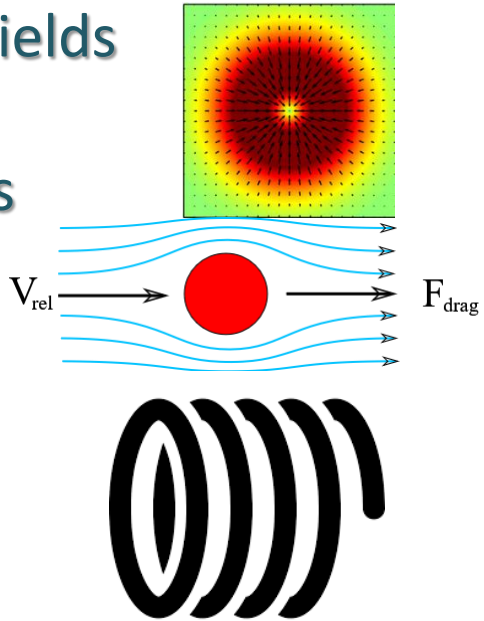
Low **k** – sagging

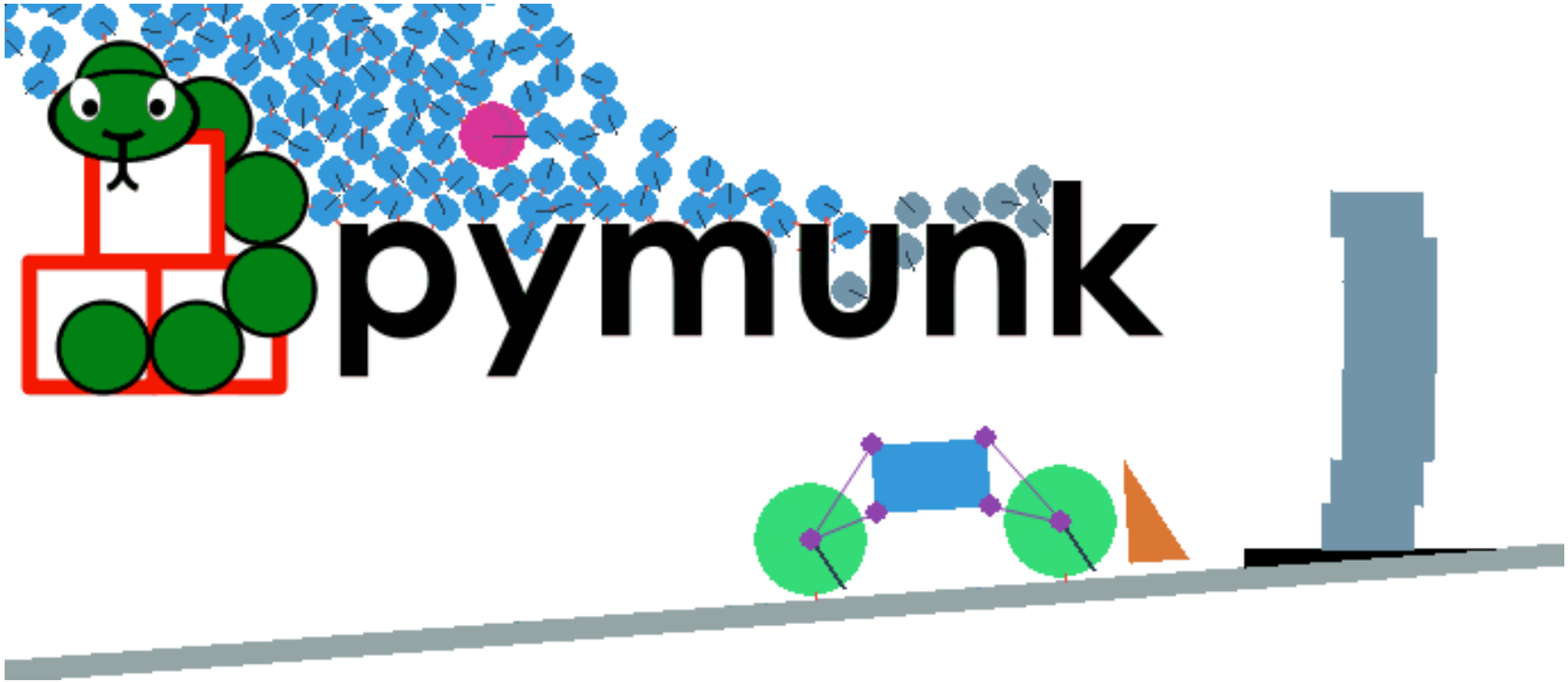
High **k** - stiff

Forces

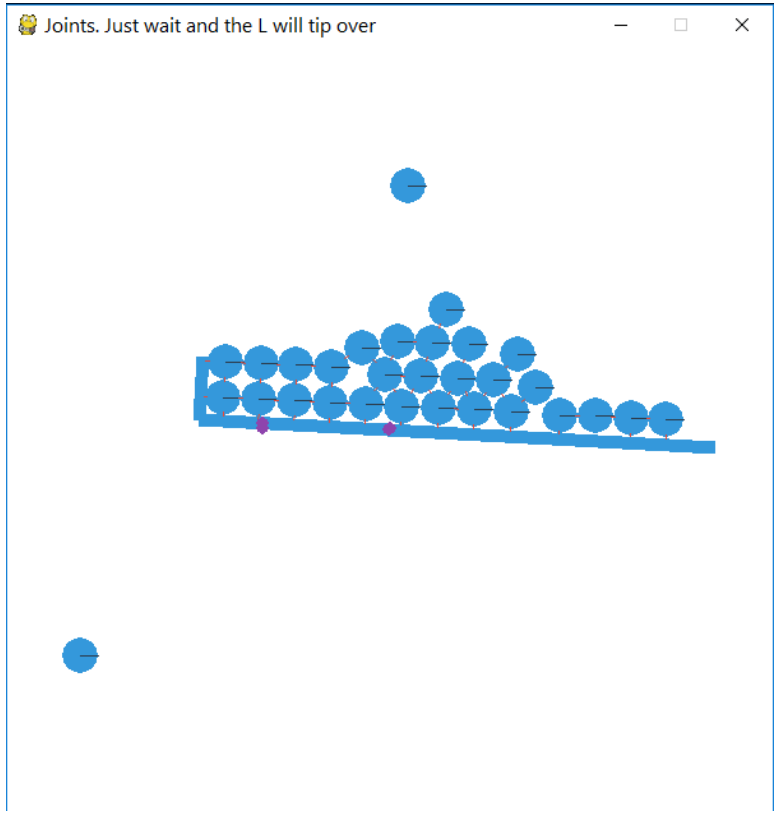
- **Constant** — gravity
- **Position/time dependent** — force fields
- **Velocity-Dependent** — drag
- **n-ary** — springs






pymunk

pymunk



Trajectory of a Spinning Object

Magnus Effect

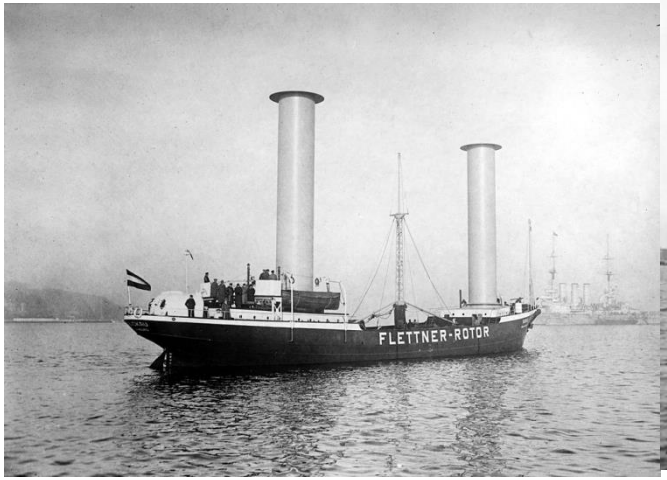
#12

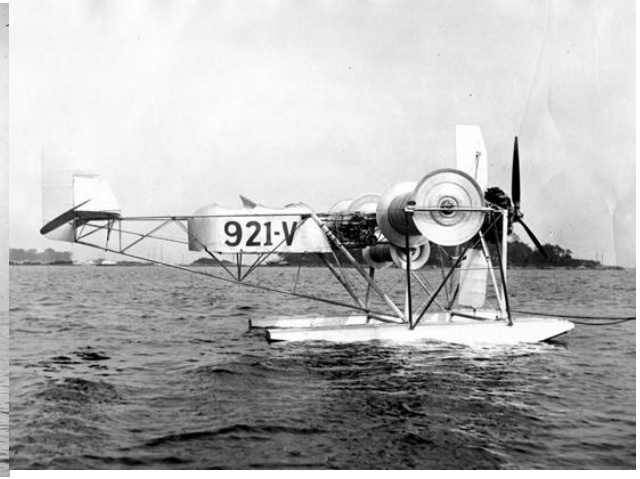
Serdar ARITAN

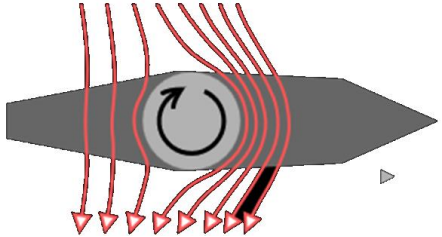
Biomechanics Research Group,
Faculty of Sports Sciences, and
Department of Computer Graphics
Hacettepe University, Ankara, Turkey

Magnus Effect

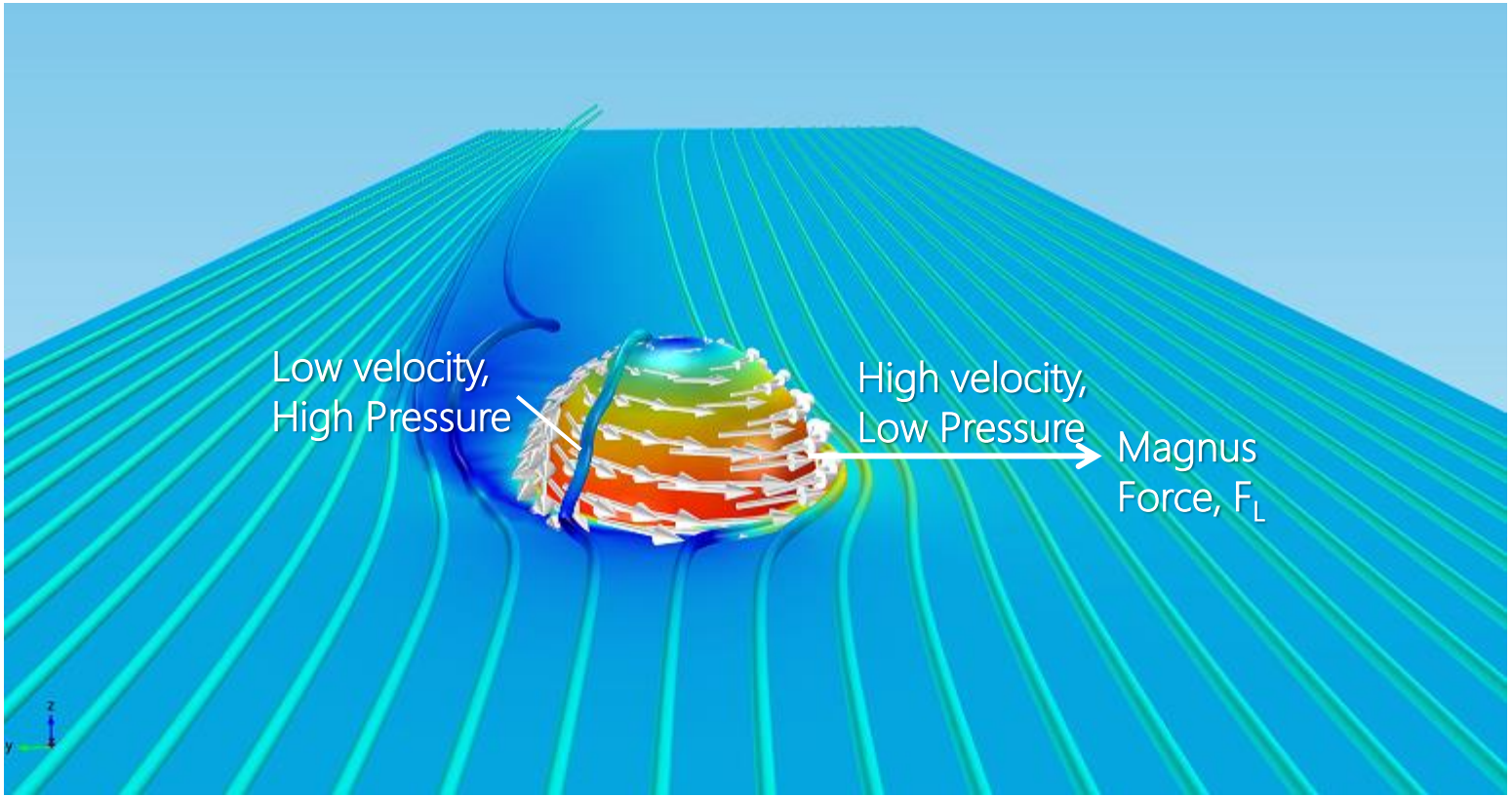
The Magnus effect is the commonly observed effect in which a spinning ball (or cylinder) curves away from its principal flight path. It is important in many ball sports. It affects spinning missiles, and has some engineering uses, for instance in the design of rotor ships and Flettner aeroplanes.



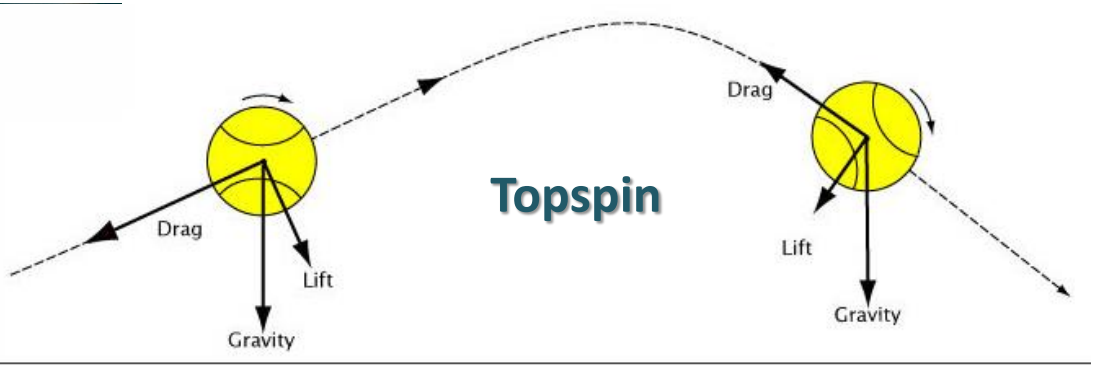


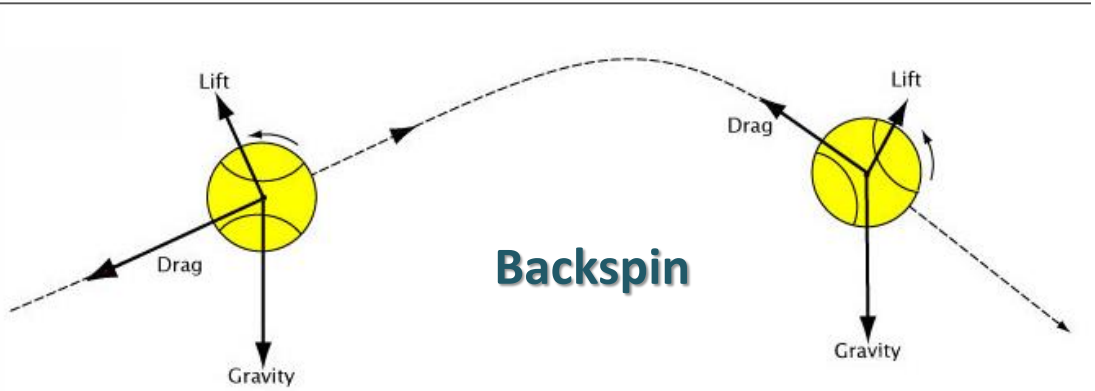


Magnus Effect



Magnus Effect





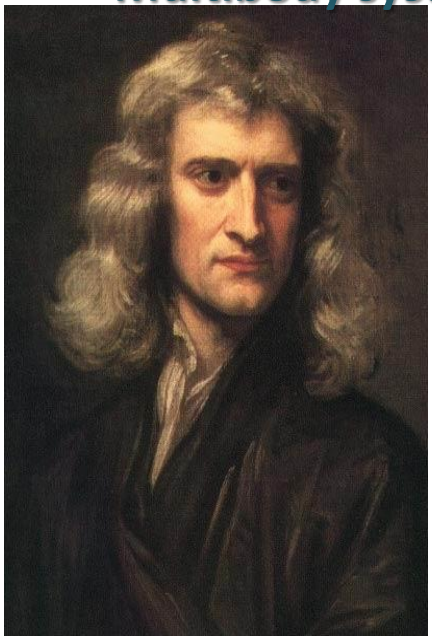
Multi Body Dynamics

#13

Serdar ARITAN

Biomechanics Research Group,
Faculty of Sports Sciences, and
Department of Computer Graphics
Hacettepe University, Ankara, Turkey

Multibody system

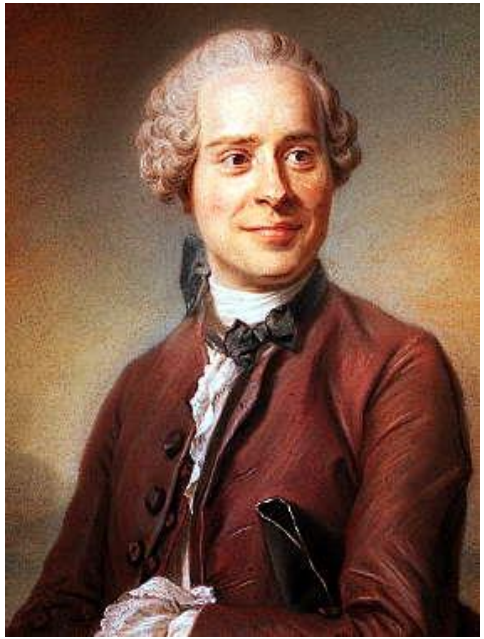


Newton

(1643 -1727)

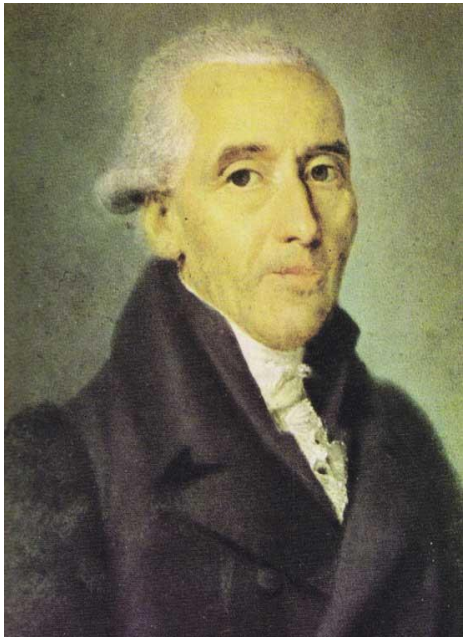
Euler

(1707 -1783)



D'Alembert

(1717 -1783)

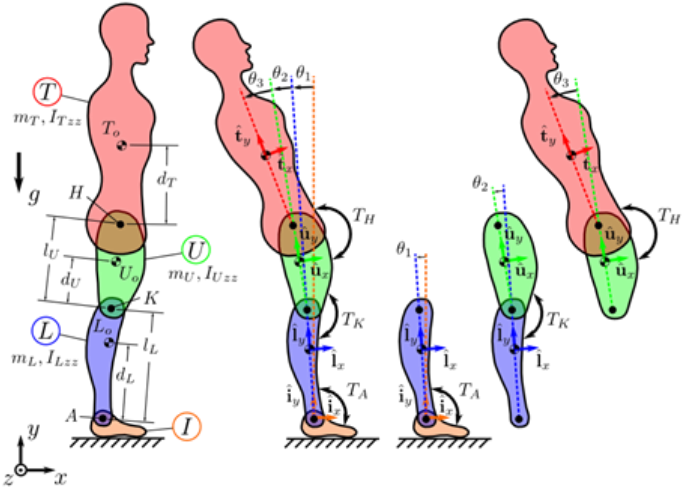


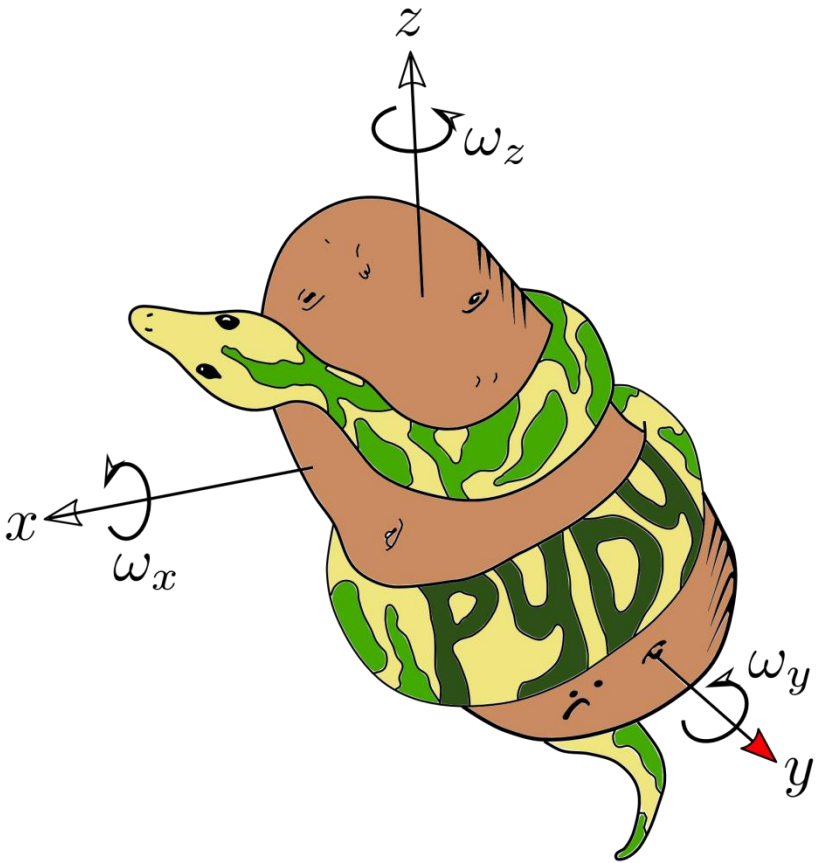
Lagrance

(1736 -1813)

PyDy

PyDy is a general tool for mutlibody dynamic analysis written in Python



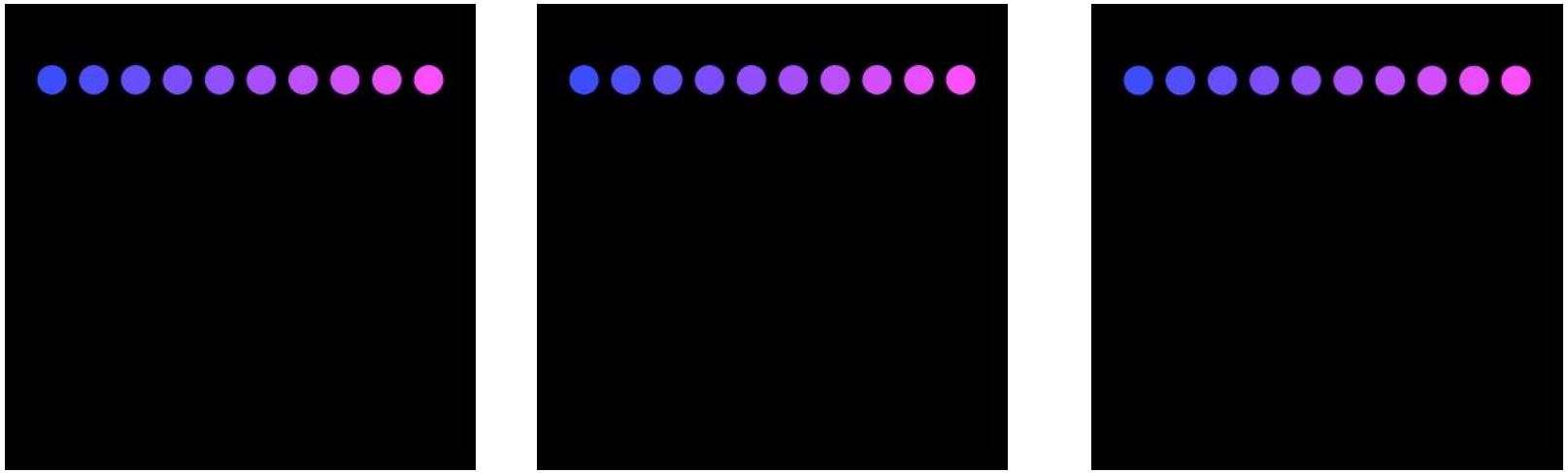


Multibody system Physics engines are able to model the motion of rigid bodies in a physical world

Box2D

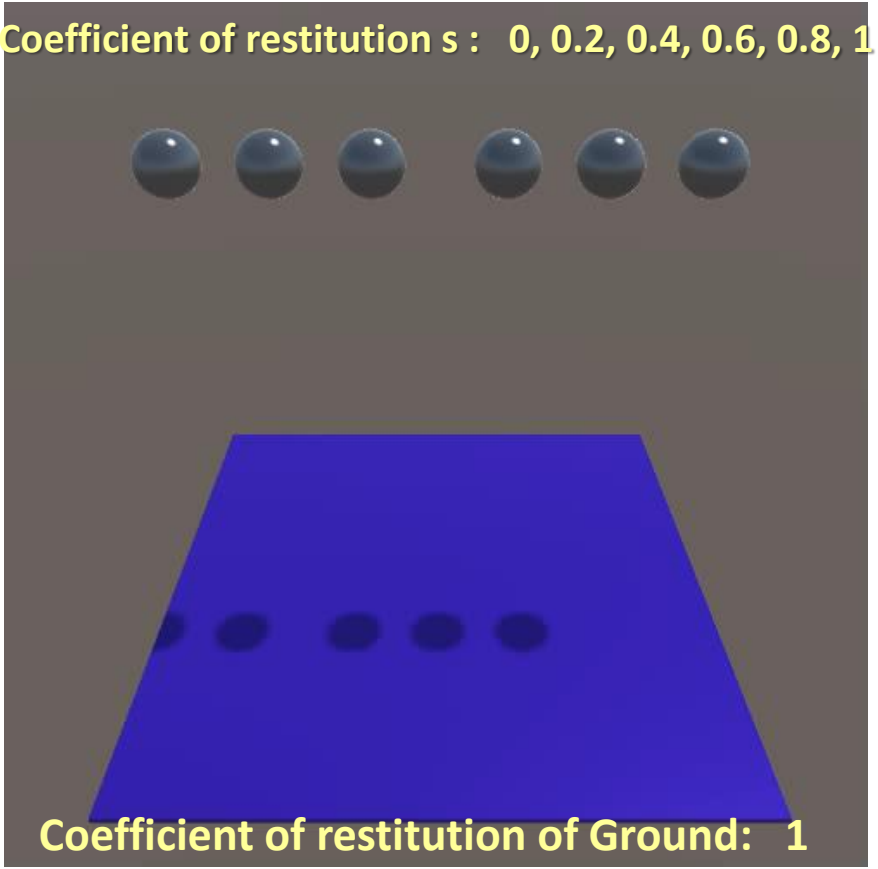
Coefficient of restitution s of balls

0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1



Coefficient of restitution of Ground

Multibody system Physics engines are able to model the motion of rigid bodies in a physical world



PhysX by NVIDIA

it is natural that most of a game company's efforts will be spent on 'how things look' rather than 'how things move'

Multi Body Dynamics with Control

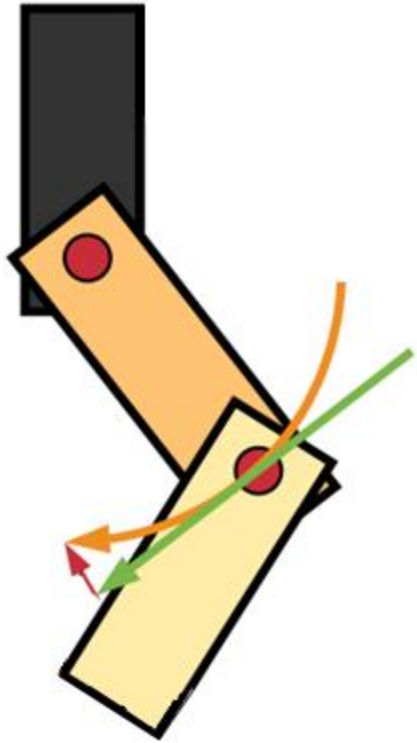
#14

Serdar ARITAN

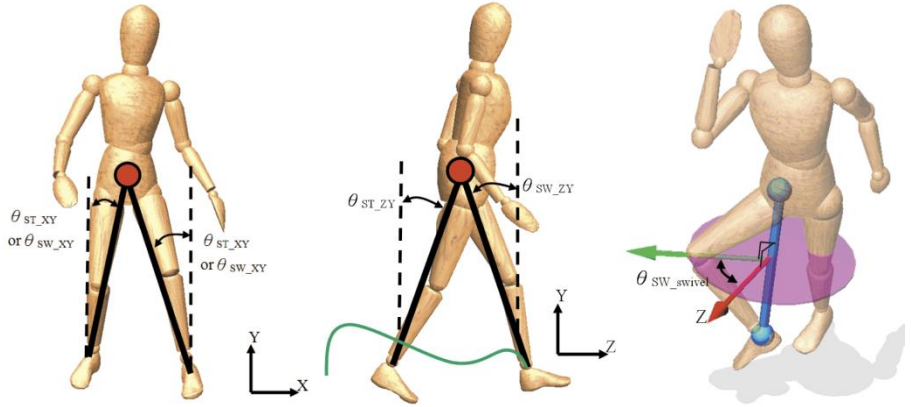
Biomechanics Research Group,
Faculty of Sports Sciences, and
Department of Computer Graphics
Hacettepe University, Ankara, Turkey

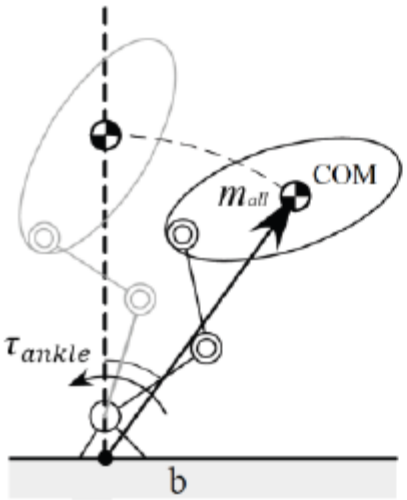
Equality Constraints

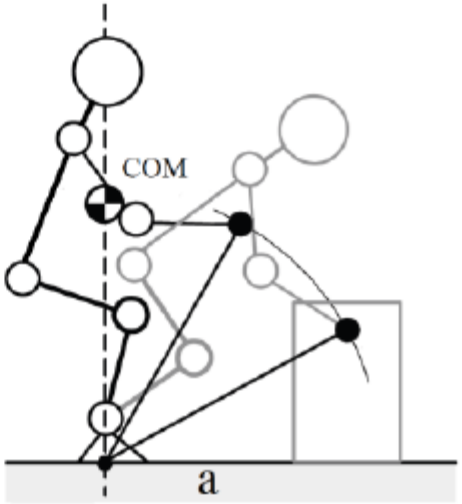
A common class of constraint is known as an equality constraint. An equality constraint is one in which the only acceptable value of **C** is **zero**. Thus, during each step of the simulation, we want to keep **C as close to zero** as possible. In other words, we want to **minimize C.** Equality constraints are used when the position of some point must always exactly match some predefined condition. A good example is a pin joint, where **two rigid bodies must always be connected at the location of the joint**.

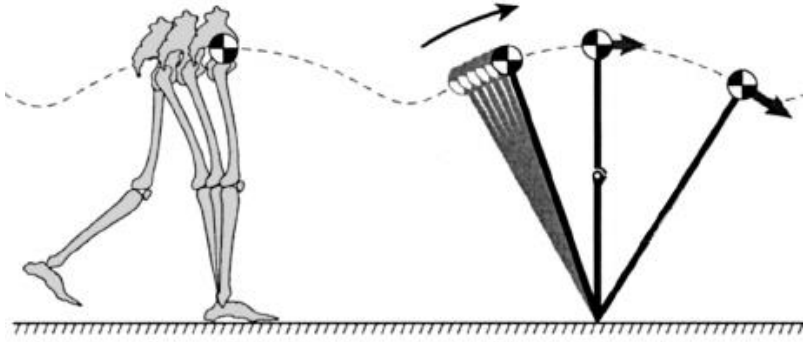


Why Pendulum is so important ?

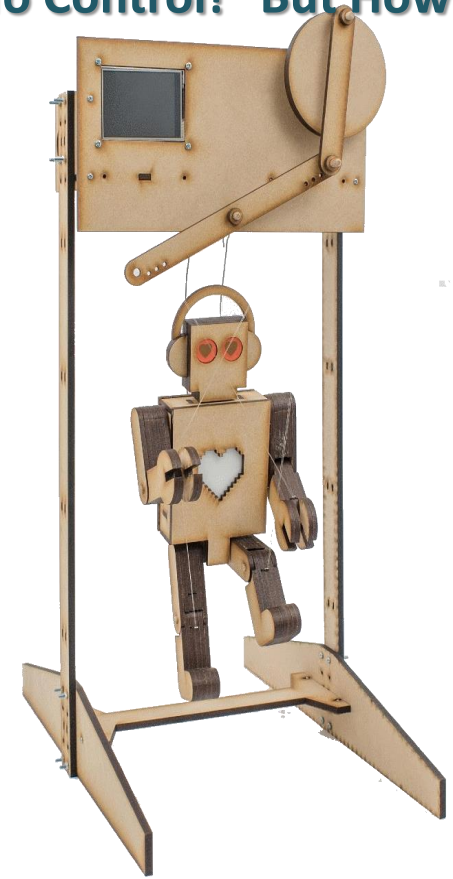


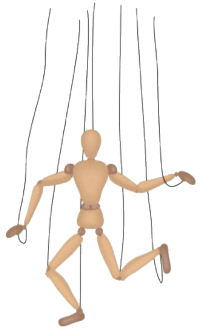




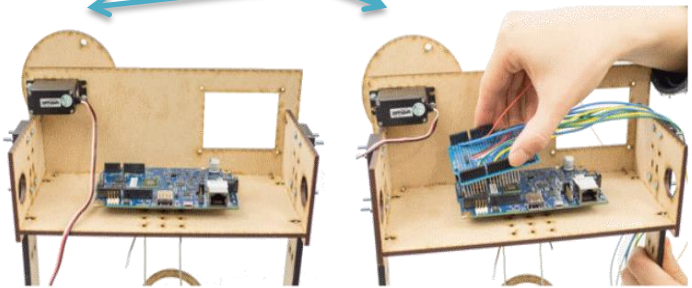


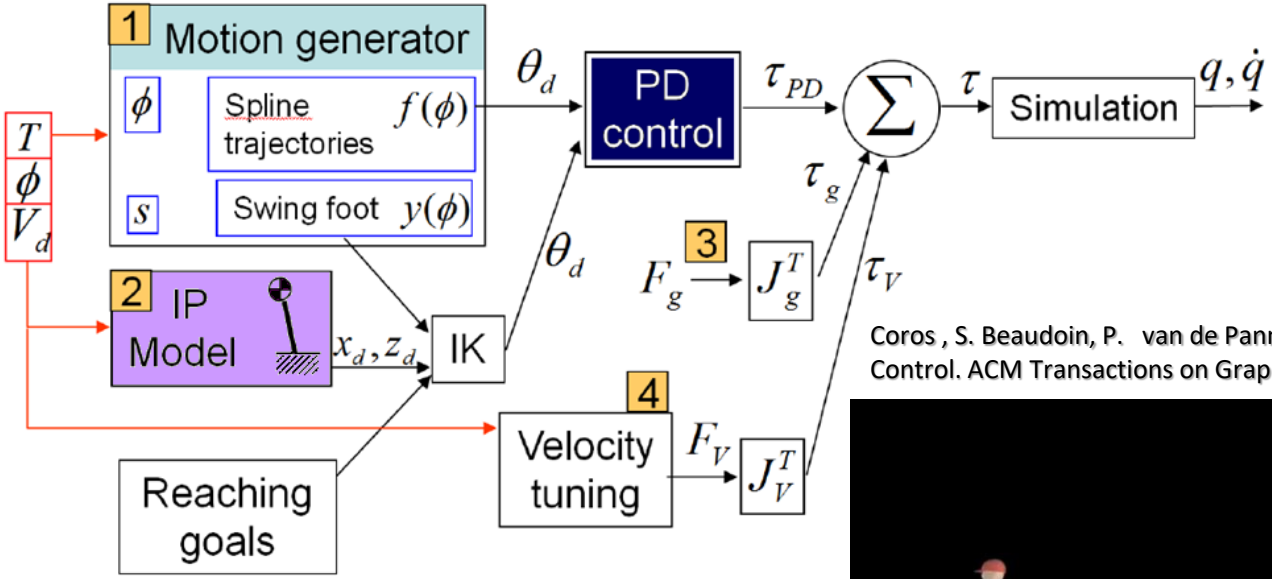
So We Need To Control! But How



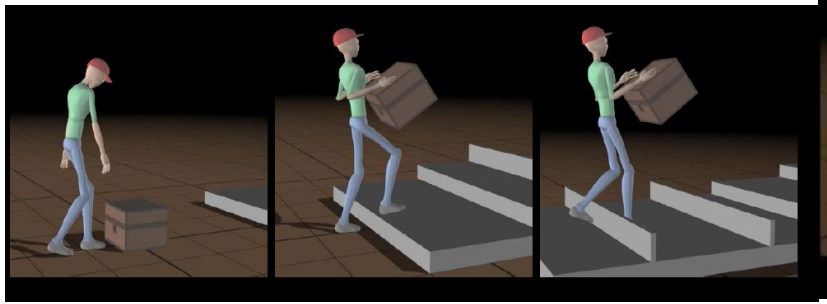


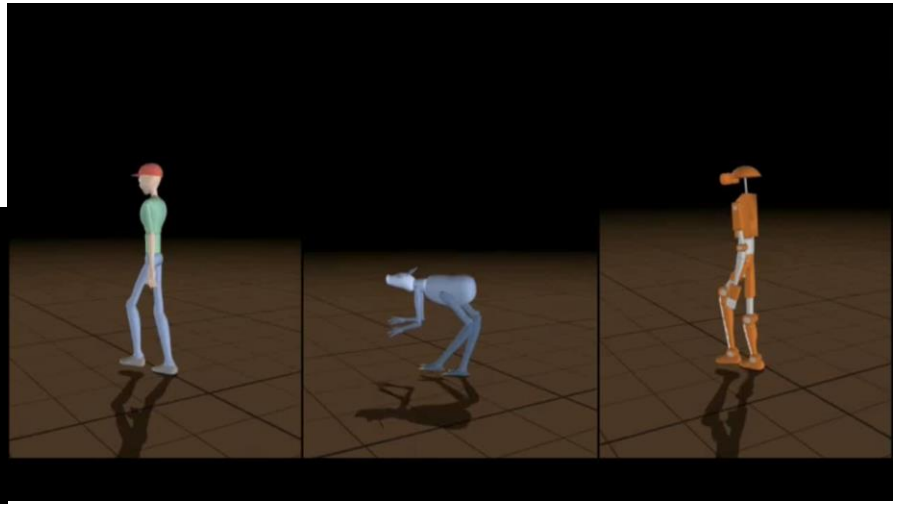
SERVO MOTOR



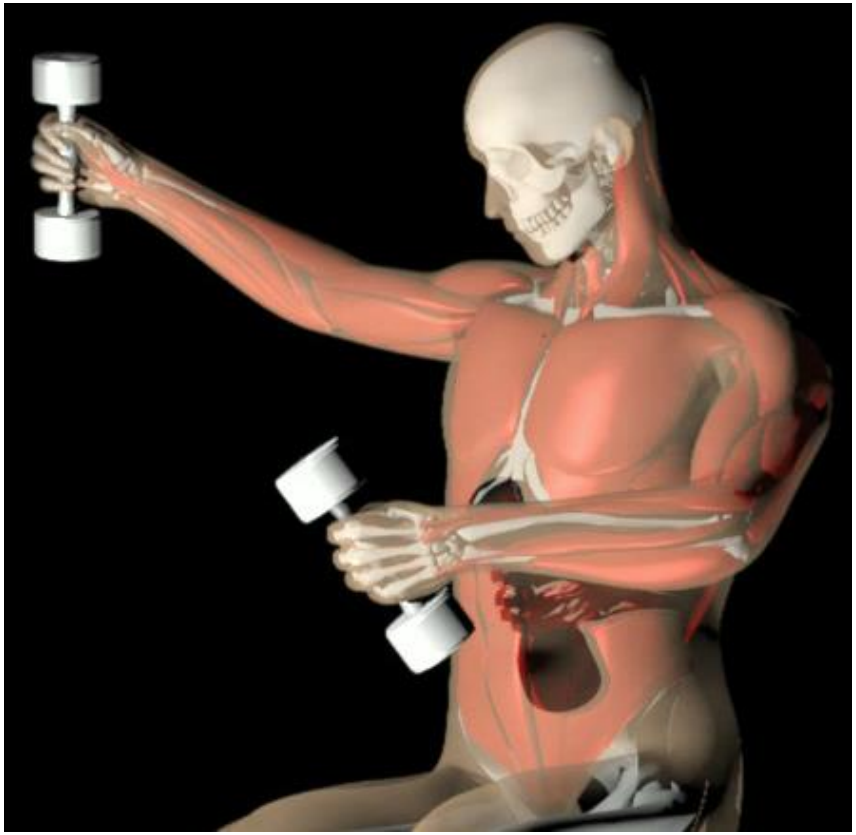


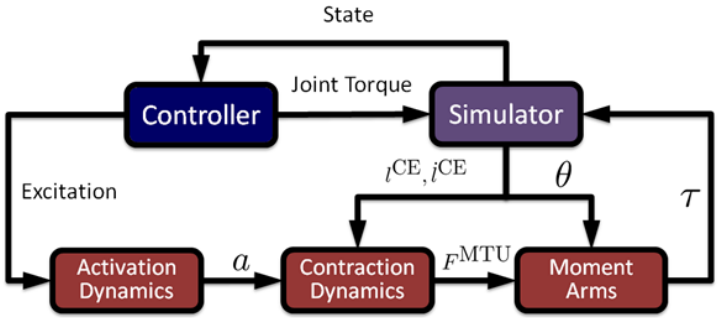
Coros , S. Beaudoin, P. van de Panne, M. (2010) Generalized Biped Walking Control. ACM Transactions on Graphics, Vol. 29, No. 4, Article 130,





HOW TO INCLUDE MUSCLES INTO THE MODEL?





Optimizing Locomotion Controllers Using Biologically-Based Actuators and Objectives