DM 2231
GAMES DEVELOPMENT TECHNIQUES

2015/16 SEMESTER 1

Week 6 – Basic Game Physics

MODULE SCHEDULE

Week	Dates	Topic	Remarks	Public Holidays
1	20-Apr-2015 to 24-Apr-2015	Module Introduction / 3D Game Programming	Issue Assignment 1	
2	27-Apr-2015 to 1-May-2015	Game Application		1 May. Labour Day
3	4-May-2015 to 8-May-2015	User Input		
4	11-May-2015 to 15-May-2015	Camera and GUI #1		
5	18-May-2015 to 22-May-2015	Camera and GUI #2		
6	25-May-2015 to 29-May-2015	Basic Game Physics	Submit Assignment 1	
7	1-Jun-2015 to 5-Jun-2015	Implementing Game Audio (E-learning)		1 Jun. Vesak Day
8	8-Jun-2015 to 12-Jun-2015	Mid-Sem Break		
9	15-Jun-2015 to 19-Jun-2015	Mid-Sem Break		
10	22-Jun-2015 to 26-Jun-2015	2D Game Programming #1	Issue Assignment 2	
11	29-Jun-2015 to 3-Jul-2015	2D Game Programming #2		
12	6-Jul-2015 to 10-Jul-2015	2D Game Programming #3		
13	13-Jul-2015 to 17-Jul-2015	Game Data		17 Jul. Hari Raya Puasa
14	20-Jul-2015 to 24-Jul-2015	Design Pattern #1		
15	27-Jul-2015 to 31-Jul-2015	Design Pattern #2		
16	3-Aug-2015 to 7-Aug-2015	Basic Artificial Intelligence (E-learning)		7 Aug. SG50 Public Holiday
17	10-Aug-2015 to 14-Aug-2015	Good Programming Practices	Submit Assignment 2	10 Aug. National Day

RECAP ON LAST WEEK'S LECTURE

- We had discussed about the main issues with Camera and GUI #2
  - How to create Third-Person Cameras for games such as Socom
    - Use Inertia to reduce motion sickness
    - Techniques to handle camera occlusion with surrounding geometries
  - How to create heads up display using orthogonal projections
  - How to create minimaps to show an overview to the player

TABLE OF CONTENT

INTRODUCTION

- When creating physics in your game, think of the following…
  - Moving objects in game world
  - Position, Velocity, Acceleration
  - Static , Mobile collisions
  - Pass through ?
  - Realistic response
    - eg : bounce off, sound
  - Detect when is collision
  - Simple?
  - CPU intensive?
  - Reasonable accuracy?

NEWTONIAN PHYSICS

- Newton's Laws of Motion
  - A constant change in the location of a body
  - Change in motion is the result of applied force.

- Motion is typically described in terms of
  - Velocity
  - Acceleration
  - Displacement
  - Time

SPEED

- Speed is…
  - A measure of how fast something is moving.
  - It is a scalar quantity: Only has magnitude.
  - Two units of measurements to describe speed:
    - units of <u>**distance**</u> and <u>**time**</u>
    - Speed is defined as the distance covered per unit time: **speed = distance/time**
  - Various units for measuring speed
    - km/h, mi/h (mph), m/s

INSTANTANEOUS SPEED

- The speed at any instant is the instantaneous speed
    - The speed registered by an automobile’s speedometer.
- When we calculate the camera inertia, we find the instantaneous speed.
    - Then we calculate the distance travelled in *frameDeltaTime seconds*

AVERAGE SPEED

- Average speed is the <u>whole distance</u> covered divided by the <u>total time of travel</u>.
- General definition:

$$Average\ Speed = \frac{Total\ Distance\ Covered}{Time\ Interval}$$

- Difference between instantaneous speed and average speed:
  - When we take a car ride from one place to another,
    - We experience different instantaneous speeds.
    - The average speed has only 1 value.
  - Question: Is a fine for speeding based on ones average speed or instantaneous speed?

FINDING AVERAGE SPEED

- Example 1: If we travel 110 km in 4 hours, what is our average speed? If we drive at this average speed for 5 hours, how far will we go?
  - Answer:
    - v_{avg} = 120 km/4 h = 30 km/h.
    - d = v_{avg} x time = 30 km/h x 5 h = 150 km.

- Example 2: A plane flies 600 km away from its base at 300 km/h, then flies back to its base at 400 km/h. What is its average speed?
  - Answer:
    - total distance traveled, d = 2 x 600 km = 1200 km;
    - total time spent (for the round trip), t = (600 km/300 km/h) + (600 km/400 km/h) = 2 h + 1.5 h = 3.5 h.
    - Average speed, vavg = d/t = 1200 km/3.5 h = 342.9 km/h.
  - Tip: start from the general definition for average speed!

VELOCITY

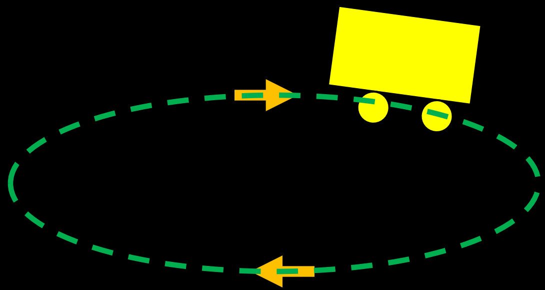
- Velocity is speed in a given direction; when we describe **speed and direction of motion**, we are describing <u>velocity</u>.

- Velocity = speed and direction; velocity is a <u>vector</u>.

VELOCITY

- Constant velocity = constant speed **and** no change in direction

Car moving 45km/h
in a straight line



Circle around the
race track at 45 km/h

<u>Question</u>: which car is moving with a constant velocity? Constant speed? Why?

ACCELERATION

- Acceleration tells you <u>how fast (the rate) velocity changes</u>:
  - Acceleration = change in velocity/time interval
  - Acceleration is **not** the total change in velocity; it is the **time rate** of change!
- Changing the velocity:
  - Changing its *speed*; *increase* or *decrease in speed*
  - Changing its *direction*
  - Or changing *both* its speed and direction
- Acceleration is the rate of change of the speed with respect to time

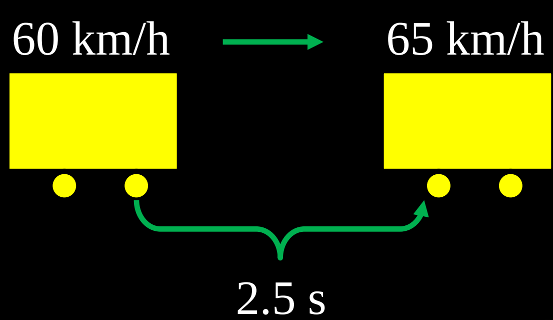
$$a = \frac{\Delta v}{\Delta t} = \frac{v_f - v_i}{t_f - t_i}$$

ACCELERATION

- Acceleration is a vector and is specified by both its magnitude and its direction.
  - When the direction of acceleration is the same as that of motion, it increases the speed;
  - When the direction of acceleration is opposite that of motion, it decreases the speed
    - This is called **deceleration**.

FINDING ACCELERATION

- Example 1: In 2.5 s a car increases its speed from 60 km/h to 65 km/h while a bicycle goes from rest to 5 km/h. Which undergoes the greater acceleration? What is the acceleration of each vehicle?



Acceleration of the car = (65 km/h - 60 km/h)/2.5 s = 2 km/h·s.

Acceleration of the bike= (5 km/h - 0 km/h)/2.5 s = 2 km/h·s.

3D MOTION

- All the formula we have discuss in this lecture can be easily extended to 3D.
  - $v_x = \frac{\Delta x}{\Delta t}$ - Velocity in direction of x-axis
  - $v_y = \frac{\Delta y}{\Delta t}$ - Velocity in direction of y-axis
  - $v_z = \frac{\Delta z}{\Delta t}$ - Velocity in direction of y-axis

- The motion along each axis can be calculated independently from other axis' motion.

3D MOTION

- To represent a 3D motion using the formulas which we had discussed,

$$\vec{v} = \frac{\Delta\vec{x}}{\Delta t}, \vec{a} = \frac{\Delta\vec{v}}{\Delta t}, \vec{F} = m\vec{a}$$

- Then, it will appear like this

$$\vec{x} \text{ is the displacement} \begin{bmatrix} x_x \\ x_y \\ x_z \end{bmatrix}, \vec{v} \text{ is the vector} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix},$$

$$\vec{a} \text{ is the vector} \begin{bmatrix} a_x \\ a_y \\ az \end{bmatrix}, \vec{F} \text{ is the vector} \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix}$$

APPLICATION

- Given the resultant force and mass, we can calculate the location of a point mass for a given change in time (Δt).

- When a force is applied, it causes an acceleration which can be calculated using Newton's second law,

$$\text{Force, } F = ma$$

$$\Rightarrow a = F/m$$

APPLICATION

- From the acceleration, we can find the change in velocity for a given change in time using:

$$\text{Acceleration, } a = \Delta v / \Delta t$$

$$\Delta v = a * \Delta t$$

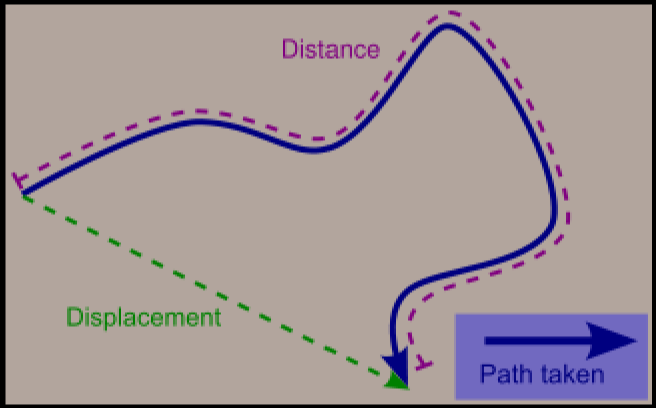
- From the velocity, we can find the change in distance for a given change in time using:

$$\text{Velocity, } v = \Delta x / \Delta t$$

$$\Rightarrow \Delta x = v * \Delta t$$

- Thus we can find the new location of the center of a point mass.

DISTANCE / DISPLACEMENT



- Displacement is the vector that specifies the position of a point or a particle in reference to a previous position, or to the origin of the chosen coordinate system.

- On the other hand, a distance is typically defined as a scalar quantity and can be used to indicate both the length of a displacement (minimum distance) and the length of a curved path (traveled distance), but not the direction of the motion.

SUVAT

- To calculate displacement all vectors and scalars must be taken into consideration. The following formulas can be used to calculate displacement, s

$$s = ut + \frac{1}{2}at^2$$

$$v = u + at$$

$$v^2 = u^2 + 2as$$

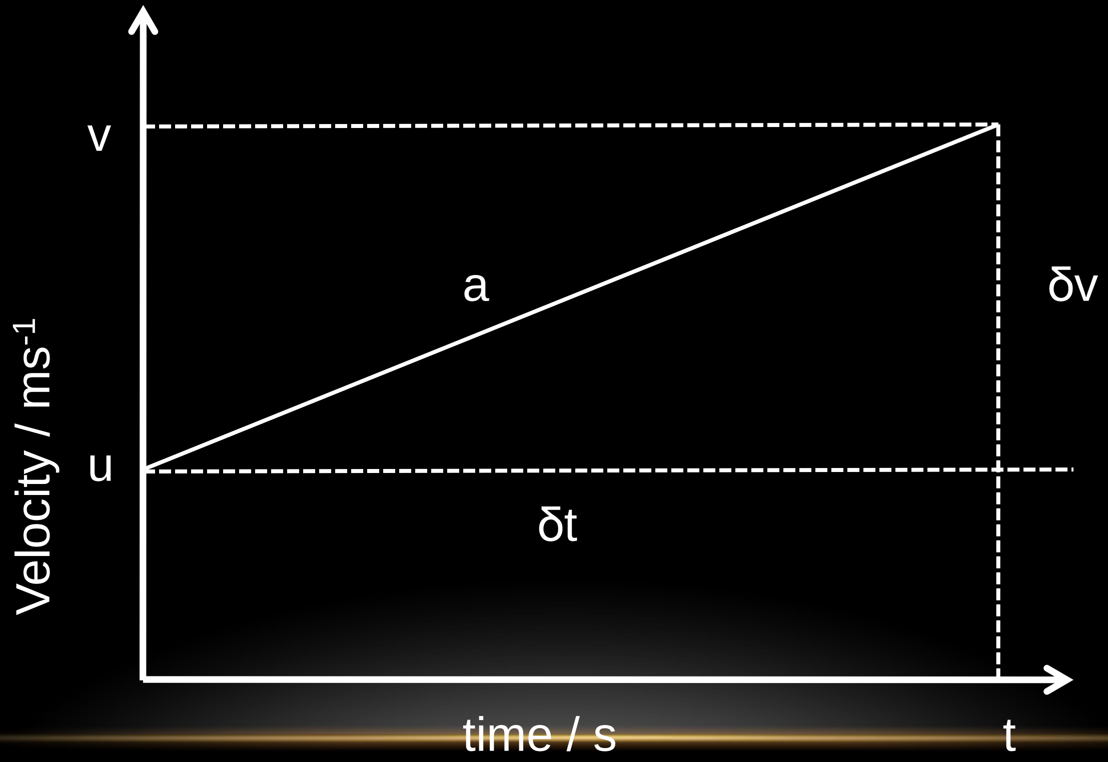
where u is initial velocity, v is final speed,

a is acceleration, t is time and s is distance

- They are often referred to as SUVAT, VUSAT, VUATS or UVATS equations, as the 5 variables they involve are represented by those letters (s = displacement, u = initial velocity, v = final velocity, a = acceleration, t = time)

REPRESENT SUVAT GRAPHICALLY

- Consider some generalised motion, starting with an initial velocity u, accelerating with acceleration a for a period t ending with a final velocity v



REPRESENT SUVAT GRAPHICALLY

- From our existing knowledge of velocity/time graphs we know that acceleration is given by the gradient of the line:

$$a = \frac{\Delta v}{\Delta t} = \frac{v_f - v_i}{t_f - t_i}$$

- So change in velocity, $\delta v = a\delta t$. So after acceleration a for t seconds

$$v = u + at$$

EXAMPLE ON SUVAT

- A driver of a vehicle travelling at a speed of 50 m/s on a motorway brakes to a standstill in a distance of 100m. Calculate the deceleration of the vehicle

- Solution
  - Initial Speed, u=50 m/s
  - Final Speed, u=0 m/s
  - Distance, s=100m
  - Deceleration, a=? m/s2
  - Time, t=? s

EXAMPLE ON SUVAT

- Choose one or more SUVAT equations which provide what we want using what we have

$$v^2 = u^2 + 2as$$

- Rearranging, we have this…

$$a = \frac{v^2 - u^2}{2s}$$

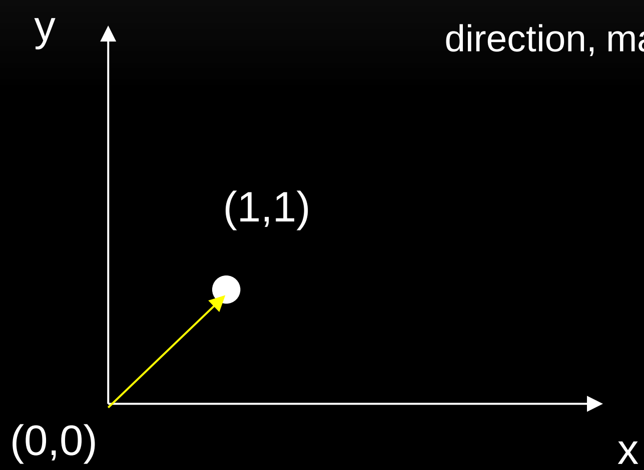
$$=\frac{0^2 - 50^2}{2 * 100}$$

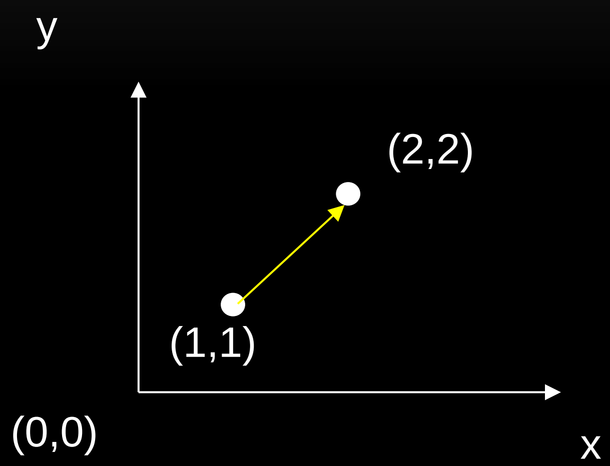
$$= -12.5 m/s^2 acceleration$$

VECTORS, POSITION, VELOCITY

position, displacement

direction, magnitude

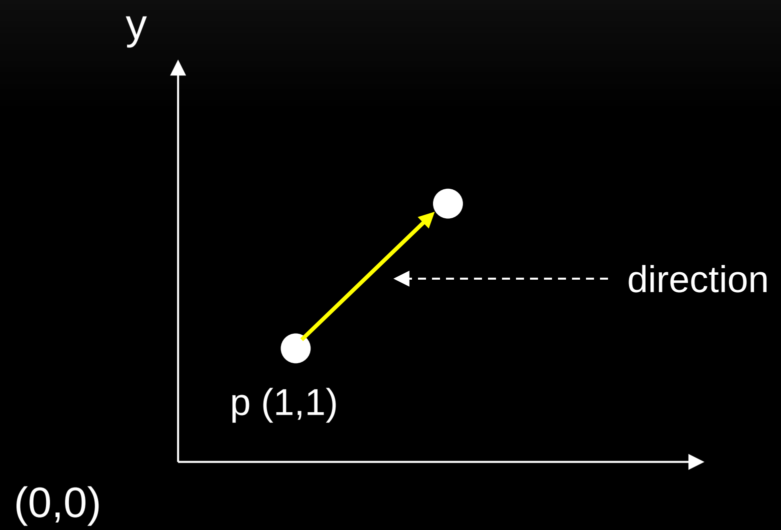




CVector vec (1,1);

```
class CVector
{
private:
        float x,y;
public:
        CVector( float sx, float sy ) {
                x = sx; y = sy;}
        ..
};
```

VECTORS, POSITION AND VELOCITY



new position = old position + velocity

vector form : p = p + v

compute : p.x = p.x + dirn.x*speed;

p.y = p.y + dirn.y*speed;

Defining it…

```
CVector p (1,1);
CVector dir (0.707, 0.707);
```

speed = 0.1 units per unit frame/ time

Magnitude of dir

m = sqrt(0.707*0.707 + 0.707*0.707)

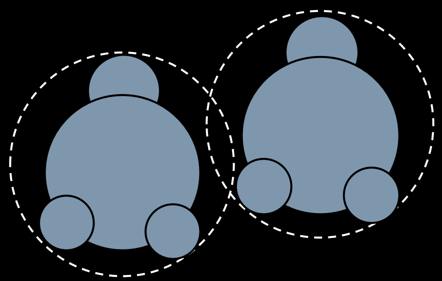
For unit movement dirn = dir/m (normalized)

COLLISION DETECTION

- Approach :
  - Collision detection may be computationally expensive
    - Avoid if possible
    - If not, choose detection method which is…
      - Cheap
      - Yet detect reasonably accurately

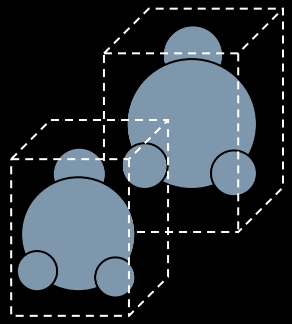
COLLISION DETECTION – CONT'D

- Various Methods - 3D

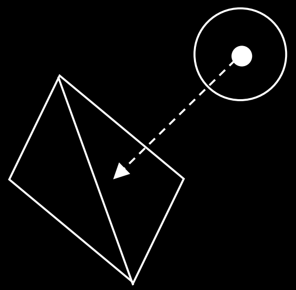


sphere – sphere

(eg : AI vs Player)



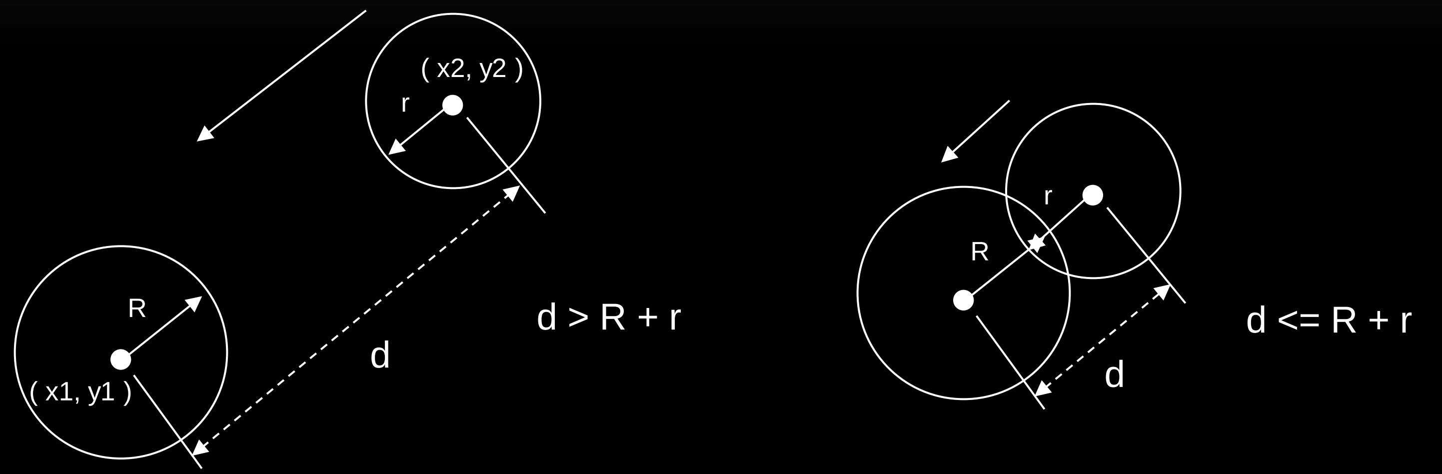
box - box



ray – plane

(eg :Player/AI vs Wall)

SPHERE – SPHERE



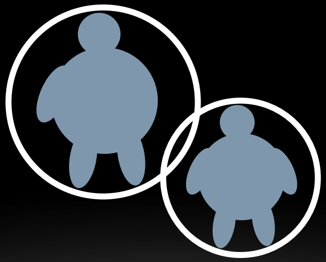
d = sqrt ((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1))

2 Dimensional

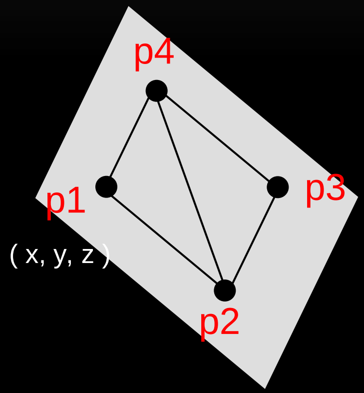
SPHERE-SPHERE – CONT'D

- Cheaper
- Simple
- Easy to implement
- No detail collision detection is needed

- Note :
    - False detection

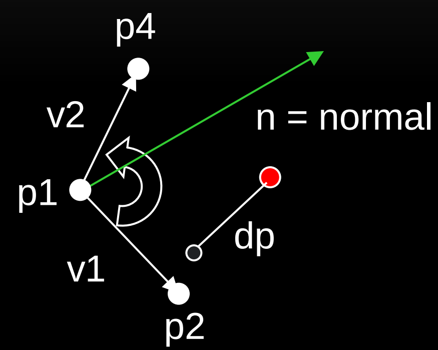


RAY – PLANE



Ax + By + Cz + d = 0

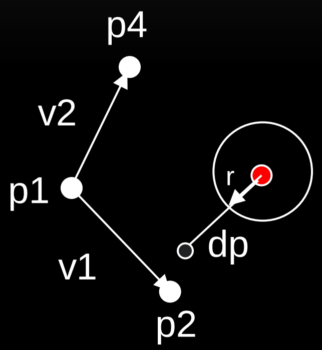
nx*x + ny*y + nz*z + d = 0



v1 = p2 - p1

v2 = p4 - p1

n = v1 x v2



d = - (nx*x + ny*y + nz*z)

Ax + By + Cz + d = dp

if (dp > 0) collision = false;

if (dp <= 0) collision = true;

for sphere :

if (dp <= r) collision = true;

SUMMARY

- We have discussed about the main issues with Basic Game Physics
  - Introduction to Physics
  - Newtonian Physics
  - Vectors, Position, Velocity
  - Movements
  - Collision Detection Methods
    - Sphere - Sphere
    - Ray – Plane
    - Sphere - Plane