Matemáticas Aplicadas

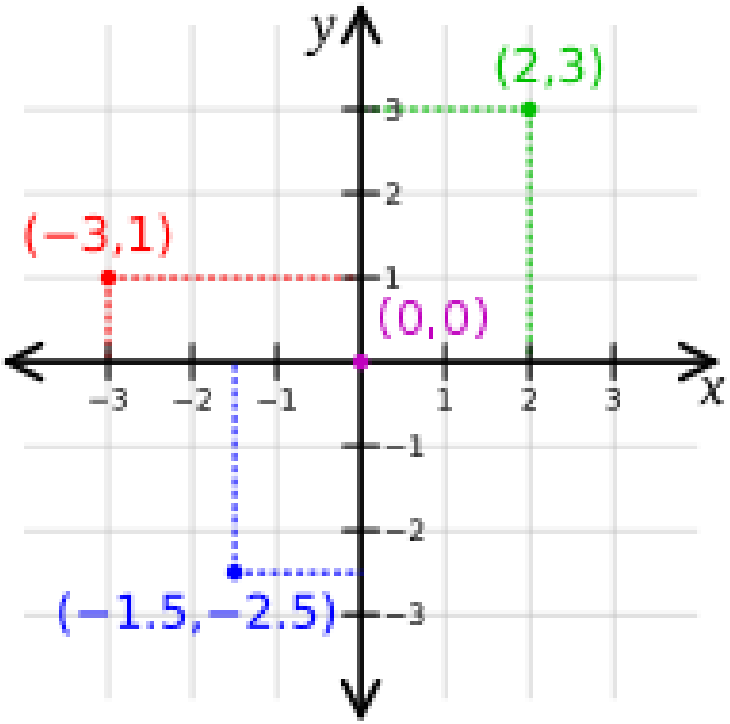
para Diseño de Videojuegos

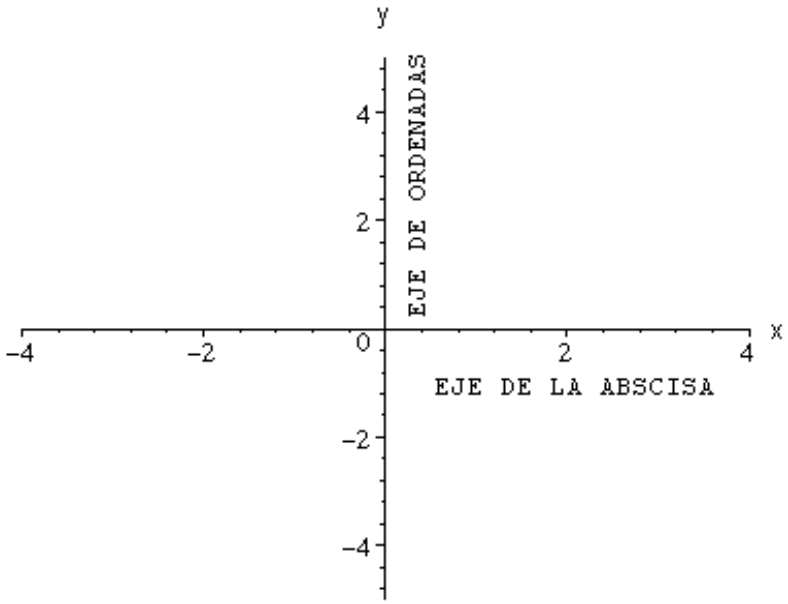
2. Geometría Analítica.

Contenidos

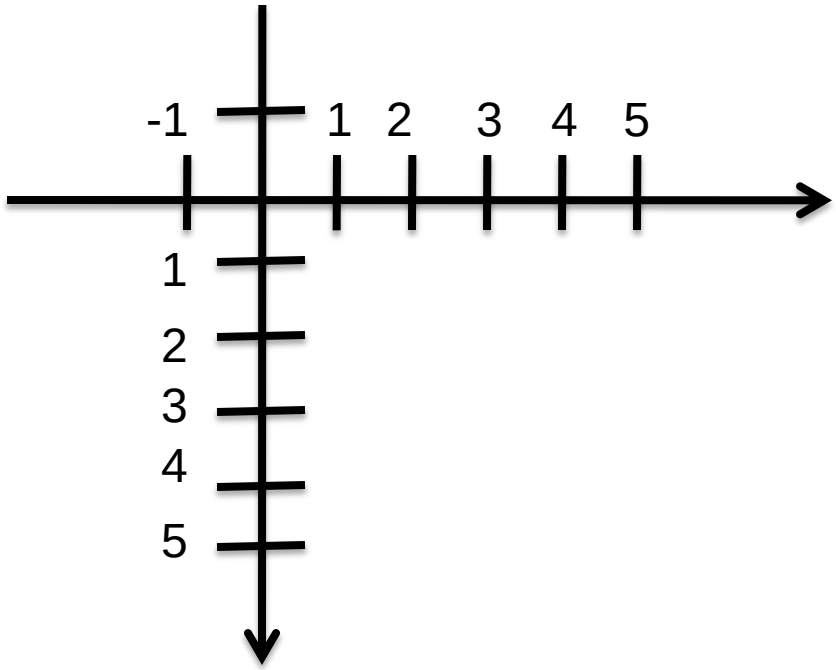
- Sistemas de coordenadas.
- Punto y traslación.
- Recta y movimiento rectilíneo.
- Teorema de Pitágoras y distancia.
- Sprites en la pantalla en un videojuego 2D.

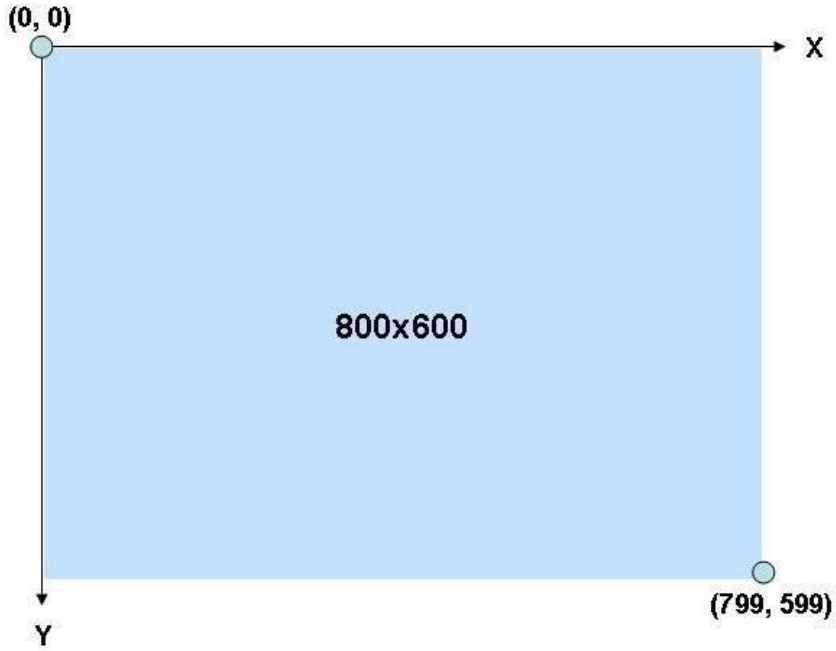
Coordenadas Cartesianas



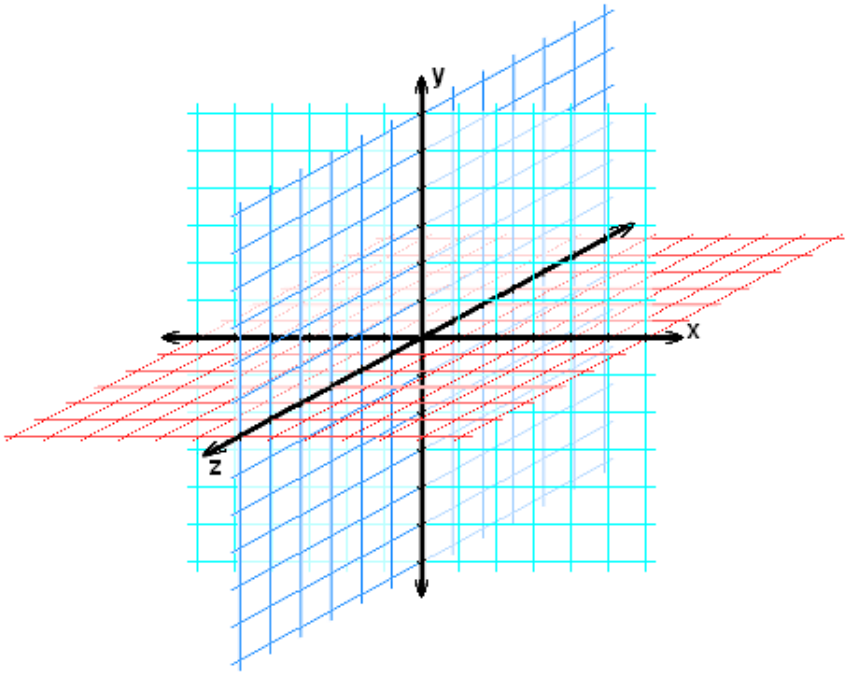


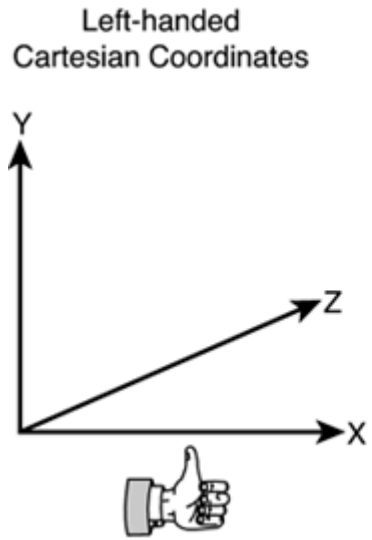
Sistema de coordenadas en videojuegos

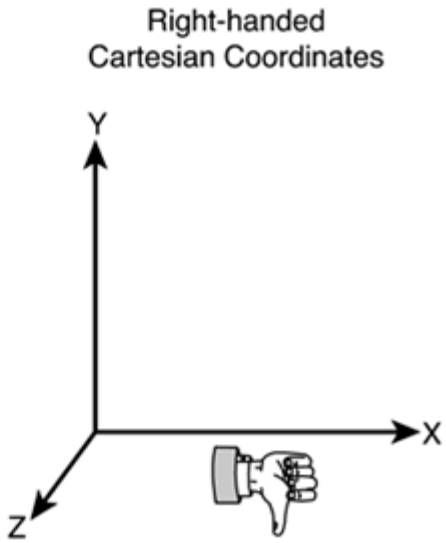




Sistemas de coordenadas 3D







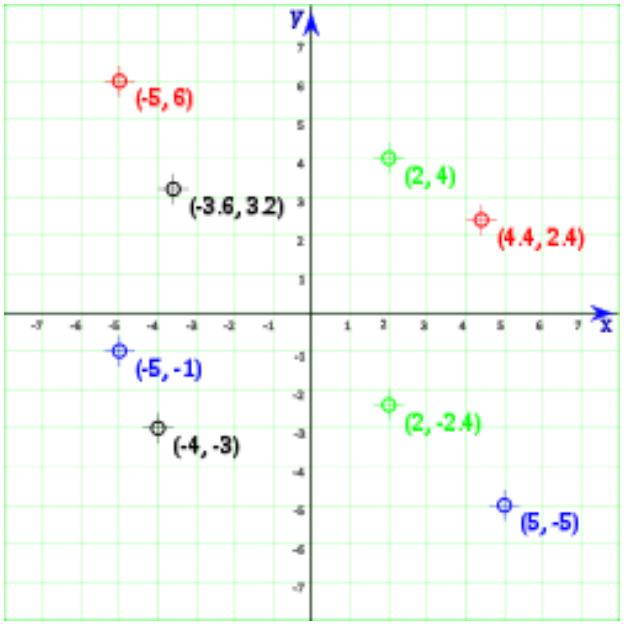
Geometría

- Rama de la matemática que se ocupa de las propiedades de las figuras geométricas en el plano o el espacio, como son: puntos, rectas, planos, polígonos, etc.

- Nociones básicas: punto, recta y plano.

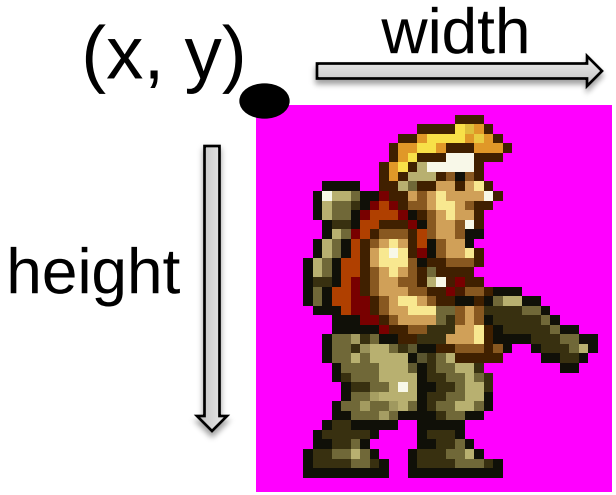
Punto

- Para localizarnos en el plano utilizamos el punto. (x, y)



Sprite

- Generalmente la representación de un Sprite en el videojuego está dada por:



Traslación

- Sabiendo que las posiciones de la nave son: (x, y)

- Si movemos el avión a la derecha a una velocidad Vx. ¿En qué posición queda al siguiente frame?

(x, y) -> (x + Vx, y)

Traslación

- Y si movemos el avión hacia abajo a una velocidad Vy. ¿?

(x, y) -> (x, y - Vy)

Traslación

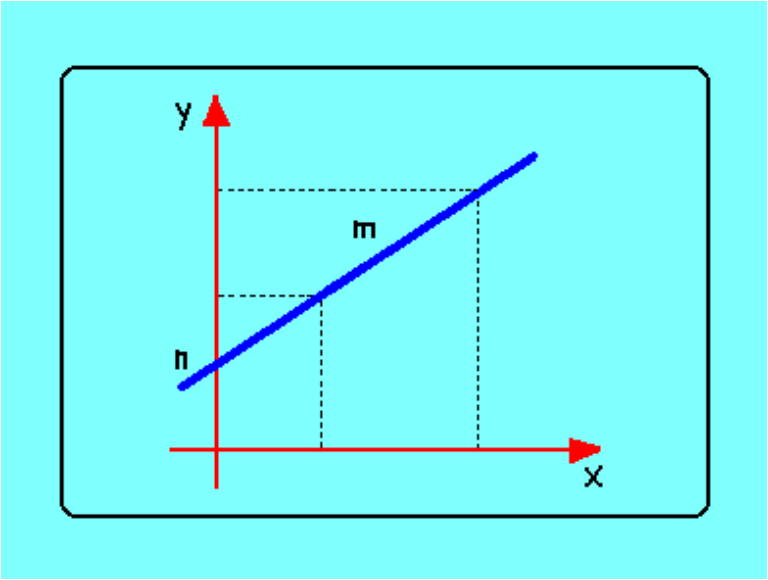
- Y si movemos el avión diagonalmente hacia arriba a la izquierda. ¿?

(x, y) -> (x - Vx, y + Vy)

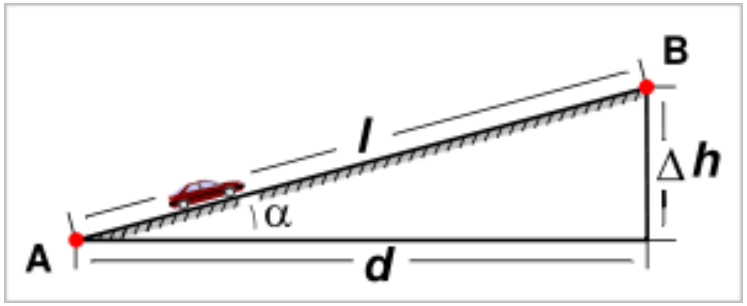
Recta

- Para representarla necesitamos la pendiente (m) y el coeficiente de posición (n), que es donde la recta corta el eje de coordenadas.

- y = m*x + n



Pendiente de la recta



$$m = \frac{\Delta y}{\Delta x}$$

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

- Es la inclinación con respecto a la horizontal.

- ¿Qué significa que la pendiente sea 0?
- ¿Qué significa que la pendiente sea negativa?

Recta

- La recta también puede ser representada de la siguiente forma:

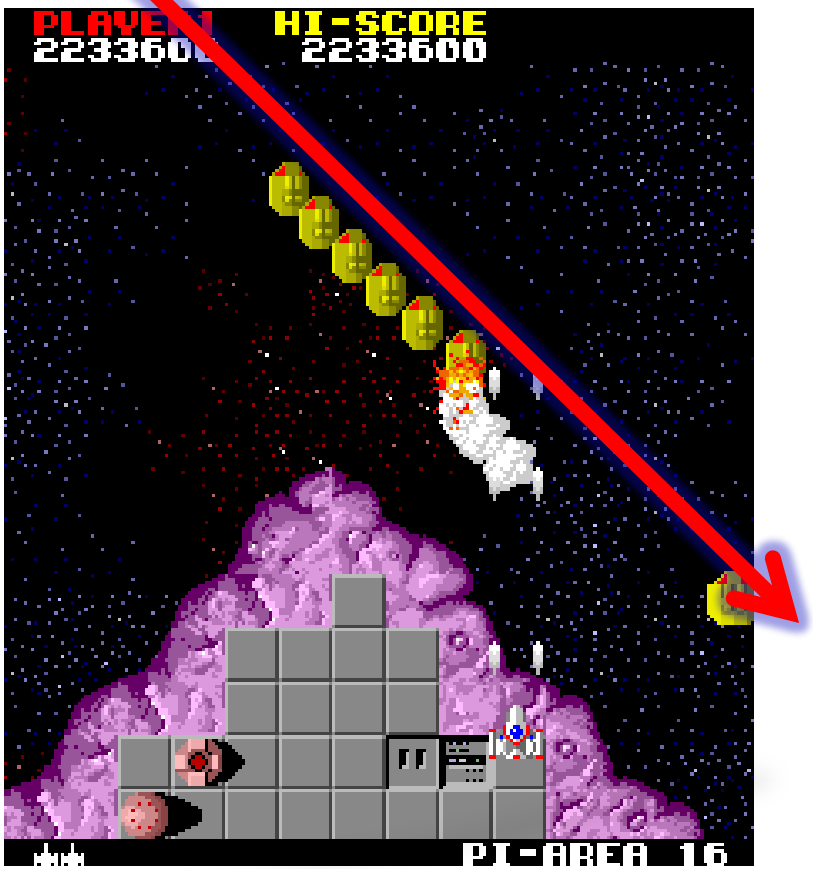
$$y - y_0 = m(x - x_0)$$

- Dónde n viene siendo $n = y_0 - m * x_0$

- Ahora bien, ¿Para qué utilizaremos la ecuación de la recta?

Movimiento rectilíneo

- Los enemigos pueden tener un movimiento rectilíneo. Para esto sólo aplicamos la fórmula de la recta.

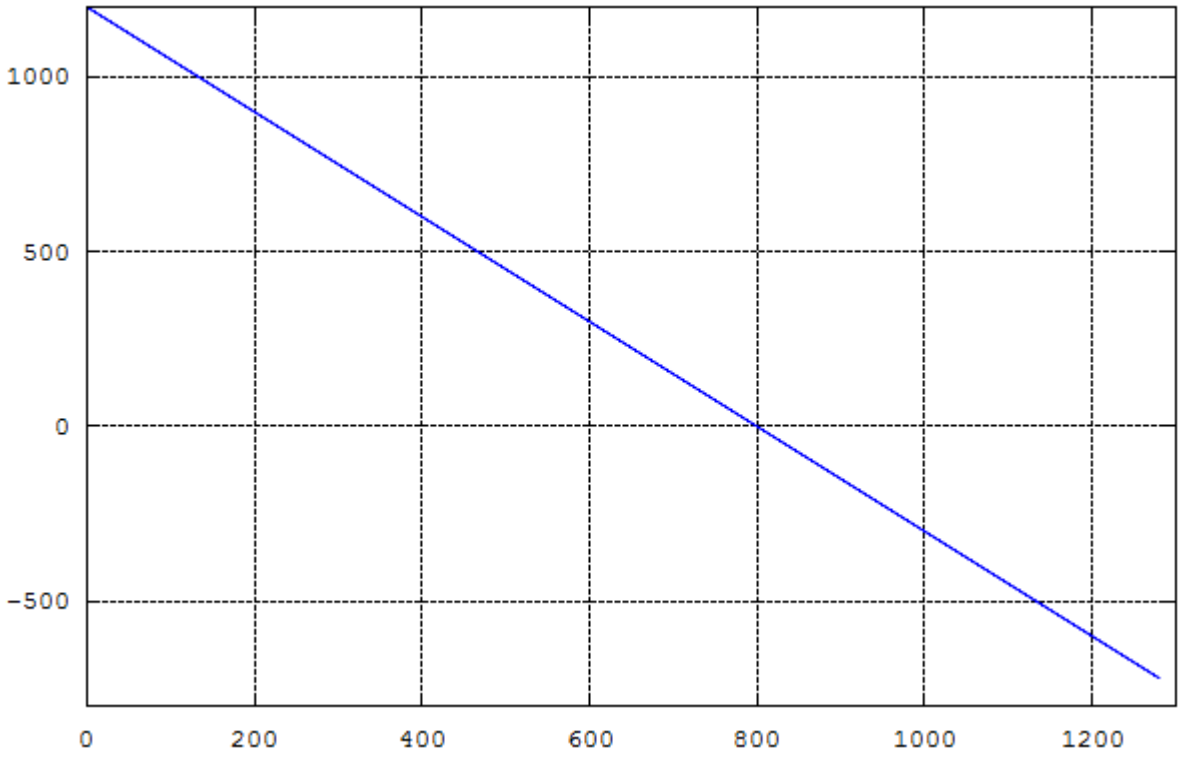


Movimiento rectilíneo

- Los enemigos pueden tener un movimiento rectilíneo. Para esto sólo aplicamos la fórmula de la recta.

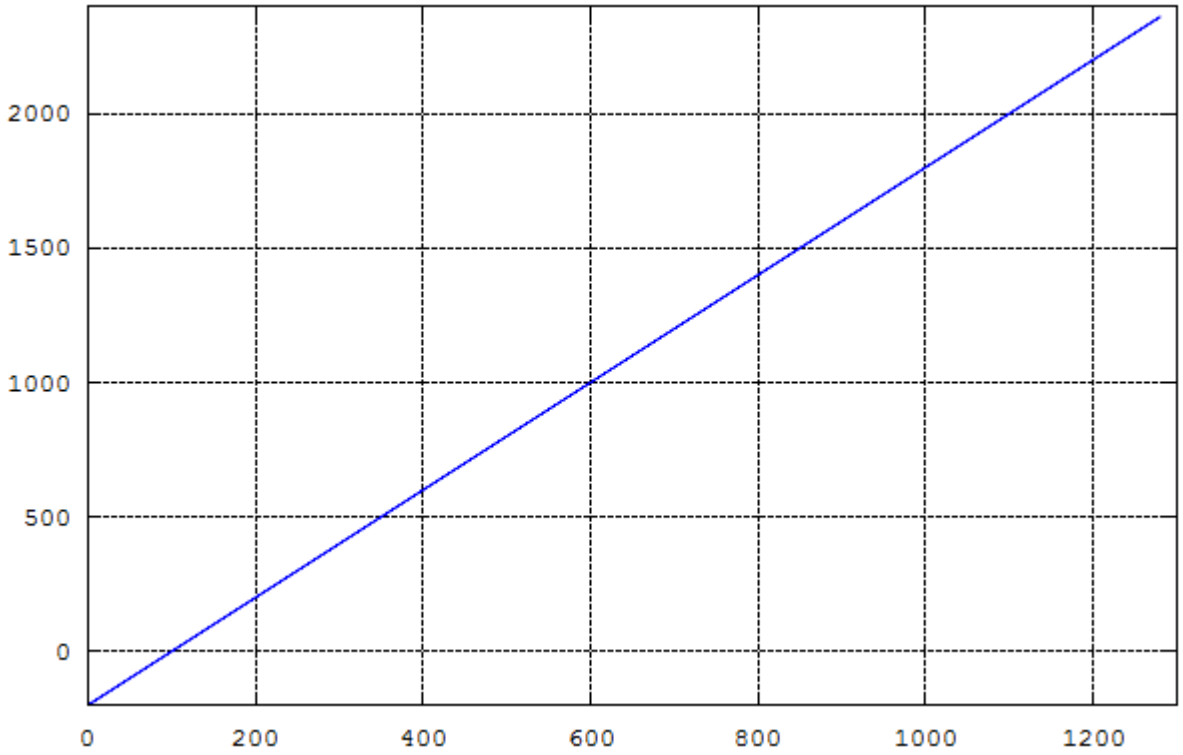
Movimiento rectilíneo

```
x = 0:.01:1280;
m=-1.5;
n=1200;
y=m.*x+n;
plot(x,y), grid on
```



Movimiento rectilíneo

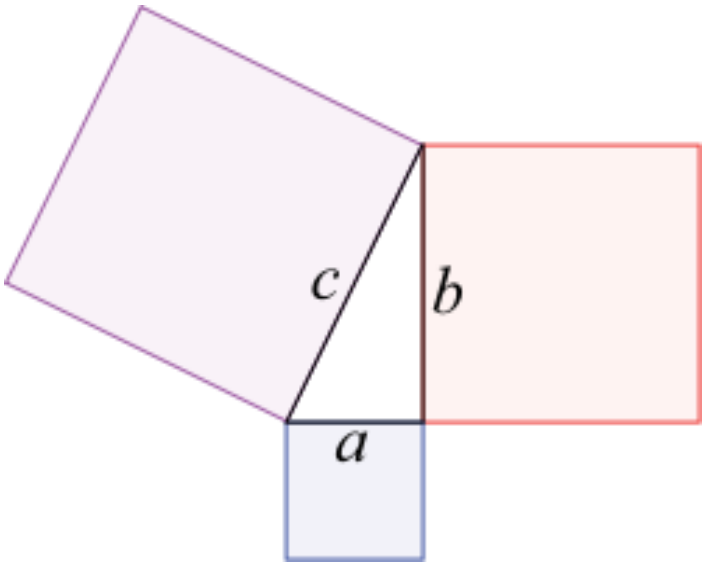
```
x = 0:.01:1280;
m=2;
n=-200;
y=m.*x+n;
plot(x,y), grid on
```



Teorema de Pitágoras

- En un triángulo rectángulo, el cuadrado de la hipotenusa es igual a la suma del cuadrado de sus lados.

$$a^2 + b^2 = c^2$$

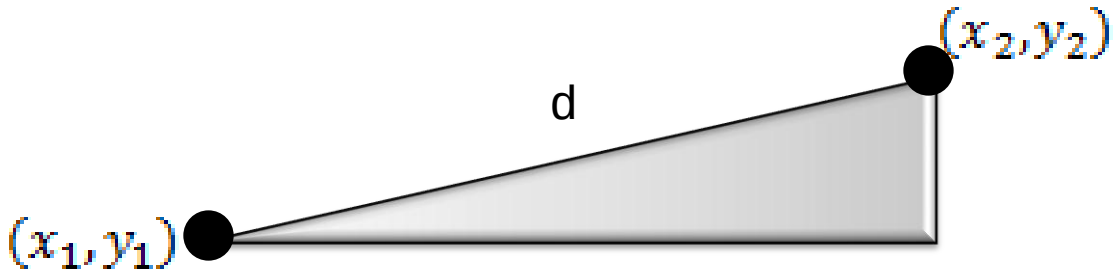


Teorema de Pitágoras

- ¿Y para qué nos sirve esto a nosotros?
- Principalmente para “la distancia”.
- En muchos videojuegos, en su inteligencia artificial se ocupa que si la distancia al enemigo es menor a una distancia mínima, entonces se procede a atacar.

Teorema de Pitágora

Teorema de Pitágoras: Distancia



- $$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

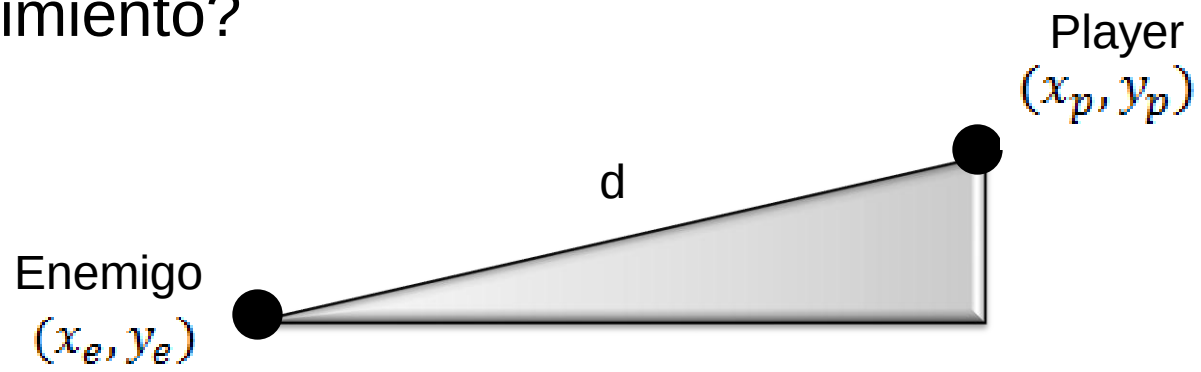
Teorema de Pitágoras: Distancia

- ¿Y en 3 dimensiones?

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Enemigo con seguimiento (2D)

- ¿Cómo modelaríamos el movimiento de un enemigo con seguimiento?



y = m*x + n

m = ? n = ?

$$m = \frac{y_p - y_e}{x_p - x_e}$$

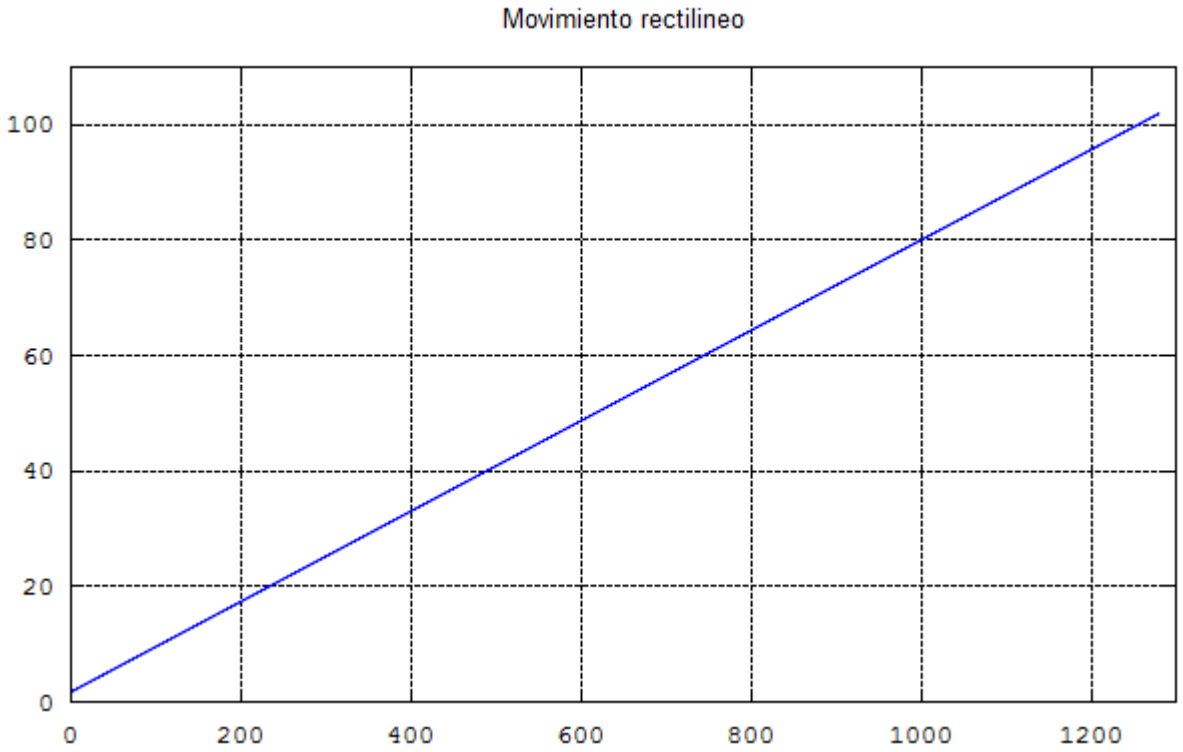
$$n = y_e - m * x_e$$

Enemigo con seguimiento (2D)

- Básicamente en cada iteración se debe volver a calcular la ecuación de la recta.

- Imaginémonos que en un momento determinado el player se encuentra en (1000, 80) y el enemigo en (3, 2).

Enemigo con seguimiento (2D)




```
x = 0:.01:1280;
m=(80-2)/(1000-3);
n=2-m*3;
y=m.*x+n;
plot(x,y), grid on, title('Movimiento rectilineo')
```

Enemigo con seguimiento (3D)

- ¿Cómo modelaríamos el movimiento de un enemigo con seguimiento en 3D?

- Generalmente tanto el enemigo como el player se encuentran en el mismo plano, por lo que un eje no se considera (eje y). Por lo que se aplica lo mismo que en 2D.

Pantalla

- Muchos de los videojuegos 2D, no tienen asociado el concepto de cámara.

- Por lo que si un sprite alcanza un extremo de la pantalla y sale de la visión de ésta, el sprite prácticamente muere! (desaparece, se elimina).

Pantalla

- Determinemos cuales son las condiciones que hacen que el sprite desaparezca.
- La pantalla tiene un ancho (pWidth) y un alto (pHeight).
- Un sprite tiene una posición (x,y) y un ancho (width) y alto (height)

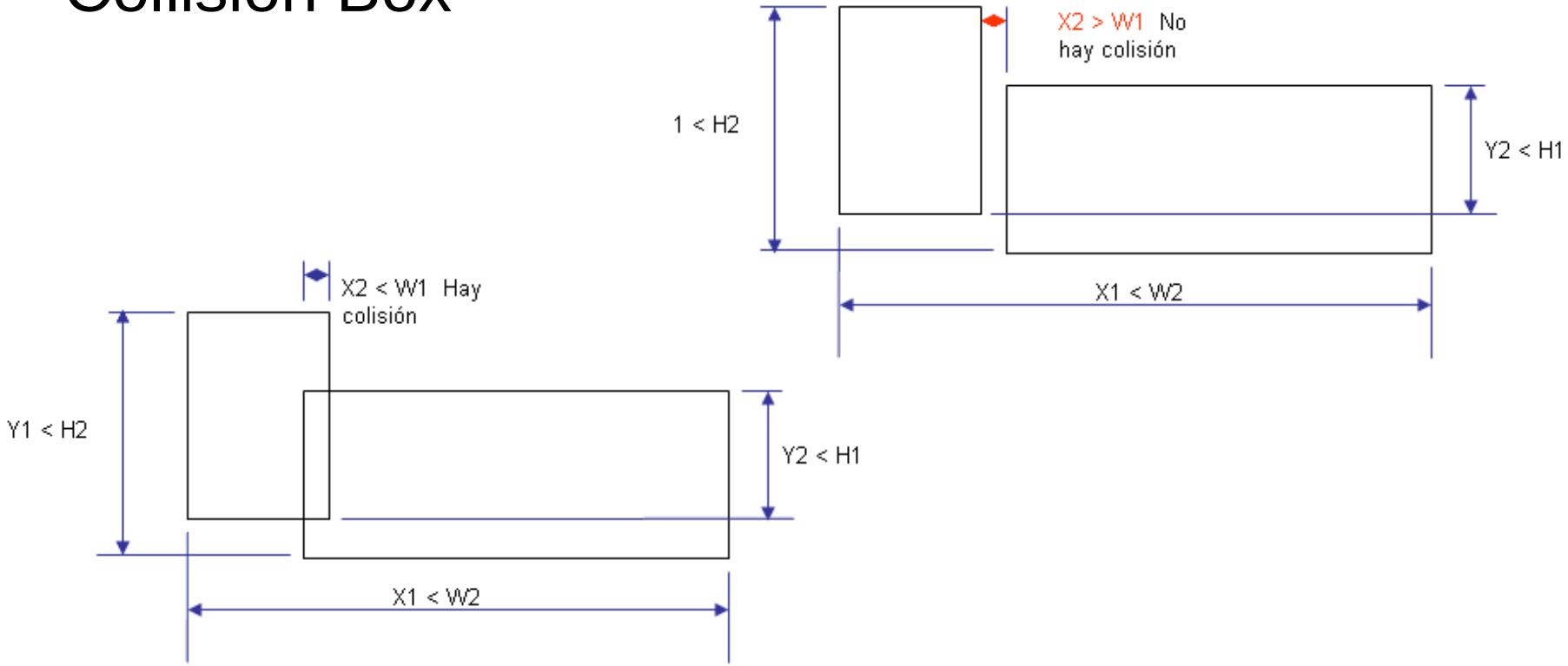
Pantalla

- Arriba:
  - y + height < 0
- Abajo:
  - y > pHeight

Pantalla

- Izquierda:
  - x + width < 0
- Derecha:
  - x > pWidth

Collision Box



Fuente: http://juank.black-byte.com/xna-colisiones-2d/

Frame de un Sprite

- Cada sprites está compuesto de frames.

Tarea

- Realizar un análisis matemático a un videojuego 2D clásico:
  - Mostrando lo que se vio en clases.
  - Adhiriendo datos técnicos del videojuego.
  - Explicando el concepto de collision box.
  - Mostrando las coordenadas (x,y) del frame de un sprite que se está dibujando.