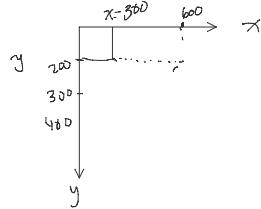


# Notas Juego rick and morty

Monday, December 4, 2023 6:43 PM

- Dimensiones del Morty 224x162 pixels.



## primera meta:

Mover a Morty. ✓

## Segunda meta:

Cambiar imagen de morty según dirección ↪

## Tercer meta:

insertar distintas imágenes de morty para cada dirección de movimiento.

## Cuarta meta:

Añadir obstáculos.

## Ocupar los sprites de un mismo movimiento.

izquierda ⌂, ⌂, ⌂, ⌂ y

derecha ⌂, ⌂, ⌂, ⌂ y

Arriba ↗ ↘

Abajo ↙ ↖

## Enemigos:

Animación de estadio de siempre.

Animación de disparo.

posición

dimensiones

## Estado de paso:

↳ public static:

movimiento (según la posición del personaje)

- move ( int x, int y, personaje \* p1 )

$$pos\_xp = p1->get\_posx$$

$$pos\_yp = p1->get\_posy;$$

$$\Delta x = pos\_enemigo.x - pos\_xp; \\ \Delta y = pos\_enemigo.y - pos\_yp;$$

if ( $\Delta x > 0$ ) {

    pos\_enemigo.x -= velocidad;

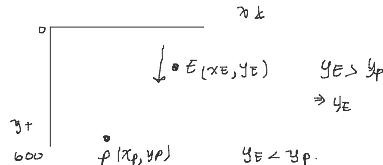
Else if ( $\Delta x < 0$ ) {

    pos\_enemigo.x += velocidad;

    if ( $\Delta y > 0$ )

        pos\_enemigo.y -= velocidad;

    Else (dy < 0) {



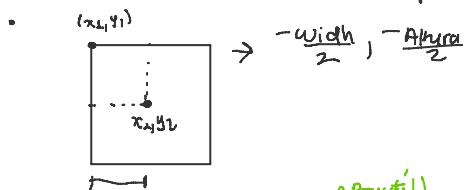
Abs Enemigo + = vc lo viola ;

Sut pos ( Posenemigo x, Posenemigo y )

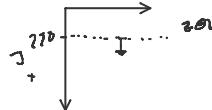
## Colisiones proyectil y enemigo:

$$\frac{175}{2} + 220$$

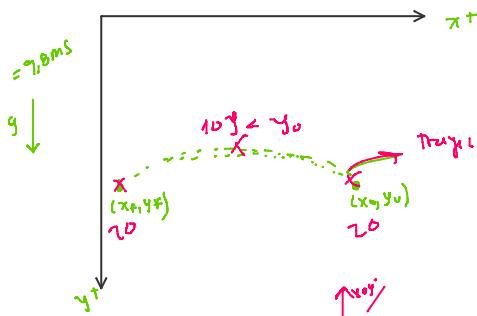
- primero redireccionar el centro del personaje, proyectil, enemigo.



$$\text{ancho} = \frac{\text{largo}}{2} = 62,5$$



Movimiento de Bala:



Nota: no tenemos en cuenta la fricción:

$$x(t) = x_0 - v_{ox}t + \frac{1}{2}v_{oy}t^2$$

$$x(t) = x_0 - v_{ox}t$$

$$y(t) = y_0 - v_{oy}t + \frac{1}{2}g \cdot t^2$$

proyectil se dispara hacia la izquierda.

(1)

(2)

$$x(t) = x_0 + v_{ox}t$$

$$y(t) = y_0 - v_{oy}t + \frac{1}{2}g \cdot t^2$$

① Proyectil se dispara a la derecha.

②

Como no se dispara con cursor

Ejemplo  $x_f = 800$  ó  $x_f = 0$

$$v_{ox} = ? \quad (v_{ox} > v_{oy}) \quad ; \quad v_{oy} = ?$$

$x_0$  = constante  
 $y_0$  = constante  
 $t_f$  = constante.

condiciones iniciales:

$$\rightarrow x_0 = x_0 \quad ; \quad x_f = 300 \text{ pixels} ; \quad v_{ox} = v_{oy} \Rightarrow t_f = ?$$

$$\rightarrow x_0 - 300 = x_0 - v_{ox}t \quad ; \quad y_0 = y_0 - v_{oy}t + \frac{1}{2}g \cdot t^2$$

$$t = \frac{300}{v_{ox}} \quad ; \quad 0 = -v_{oy}t + \frac{1}{2}g \cdot t^2$$

$$0 = -2v_{oy}t + \frac{1}{2}g \cdot t^2 \quad ; \quad 2v_{oy}t = \frac{1}{2}g \cdot t^2$$

$$v_{oy} = \frac{g \cdot t}{2}$$

$$\rightarrow v_{yt} = v_{oy} + 10t$$

$$v_{oy} = \frac{1500}{v_{ox}} \rightarrow 1500(\frac{1}{v_{ox}})$$

$$v_{oy} = \frac{10 \cdot 300}{v_{ox}} \quad ; \quad 5 \cdot 300 = \frac{1500}{v_{ox}}$$

$$10 = 30 - v_{ox}t + \frac{1}{2}gt^2$$

$$0 = -v_{ox} + gt$$

$$v_{ox} = gt$$

$$-20 = -9t^2 + \frac{1}{2}gt^2$$

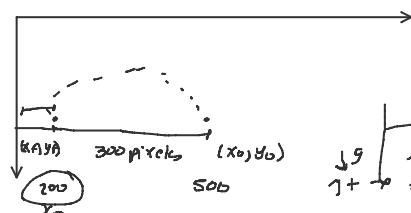
$$-20 = -\frac{9t^2}{2}$$

$$-\frac{40}{9} = t^2$$

$$t_f^2 = 2,070 \text{ seg.}$$

$$v(t_f) = -v_{ox} + g \cdot t_f = 2,070$$

$$v_{ox} \approx 14\sqrt{2} \approx 198$$



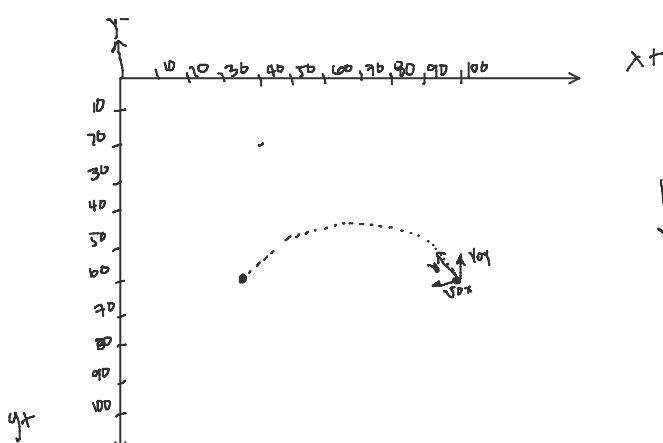
$$x_f = 30 - 20$$

$$y_f(t_f) = 10$$

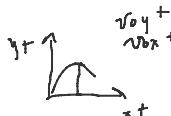
$$d = 10$$

$$30 - 20 = 10$$

$$10 = 10$$



↓ gravedad



$$y(t) = y_0 - v_{oy}t \pm \frac{1}{2}gt^2$$

$$y(t) = y_0 + v_{oy}t - \frac{1}{2}gt^2$$

$$v_{ox} = 12$$

$x_i = \text{conocido}$     $v_0x = 12$     $v_{oy} = ?$   
 $y_i = \text{conocido}$     $v_0y = ?$   
 $\Delta x = 200 \text{ pixels}$   
 $\cancel{x_i - dx = x_i - x_{0t}}$   
 $\boxed{\Delta t = \frac{dx}{v_0x}}$   
 $y_i = \text{conocido}$   
 $t \cdot v_y = 0$   
 $dy = 40 \text{ pix}$   
 $y(t_f) = y_0 - v_{oy} t v_y = 0 + \frac{g}{2} t^2 v_y^2$   
 $-dy = -v_{oy} t v_y = 0 + \frac{g}{2} t^2 v_y^2$   
 $\frac{g}{2} t^2 v_y^2 - v_{oy} t v_y = 0 + dy = 0$   
 $\frac{v_{oy}^2}{2g} - \frac{v_{oy} \cdot dy}{g} + dy = 0$   
 $\frac{v_{oy}^2}{2g} = -dy$   
 $\boxed{v_{oy} = \pm \sqrt{dy \cdot 2g}}$   
 $\sqrt{t} \propto v_{0x}$   
 $x_i = 675$   
 $y_i = 4138$   
 $y(t) = 438 - 2B \cdot t + \frac{g}{2} t^2$   
 $x(t) = 675 - 35t$   
 $473 = 675 - 35t$   
 $200 = 35t$   
 $\frac{200}{35} = t$   
 $t \approx 5,714$   
 $40/\%$   
 $\cancel{x_i - dx = x_i - y_{0t} \cdot 2 \sqrt{\frac{2dy}{g}}}$   
 $-dx = -v_{0x} \cdot 2 \sqrt{\frac{2dy}{g}}$   
 $\boxed{\cancel{v_{0x} = \frac{dx}{2} \sqrt{\frac{g}{2dy}}}}$   
 $v_{0x} = \frac{200}{2} \sqrt{\frac{9.18}{2 \cdot 40}} \approx 35 \text{ pix}$   
 $v_{0y} = \sqrt{40 \cdot 2 \cdot 9.18} = 78 \text{ pix}$   
 $\begin{array}{rcl} 1 & \longrightarrow & 1000 \\ 0,1 & \longrightarrow & x \end{array}$

### Cedición entre enemigos:

- Arreglo de 4 Enemigos; cada uno con coordenadas distintas.
- Seleccionar posiciones dependiendo de su posición en el arreglo
- Dependiendo del enemigo se cambian sus colisiones:

Usar QMap ; QMap<int, Enemigo> Mapa-enemigos

// inicializa enemigos.

```
for( int i=0 ; i < Mapa-enemigos ; i++ ) {
```

```
if( i==0 ) {
```

```
    Mapa-enemigos[ i ] = new Enemigo( x1, y1 );
```

```
    5 <--> addItem( Mapa-enemigos[i] );
```

```
    if( i==1 ) {
```

```
        Mapa-enemigos[ i ] = new Enemigo( x2, y1 );
```

```
        5 <--> addItem( Mapa-enemigos[i] );
```

```
}
```

```
if( i==2 ) {
```

```
    Mapa-enemigos[ i ] = new Enemigo( x1, y2 );
```

```
    5 <--> addItem( Mapa-enemigos[i] );
```

```

        S: unir > add_item( mapa_enemigos[i] );
    }
    if(i==z){
        Mapa_enemigos[i] = New_Enemigo( -x1,y1 );
        S: unir > add_item( Mapa_enemigos[i] );
    }
}

```

//Bajar si estan colisionando :

```

for(int i=0; i< Mapa_enemigos; i++){
    for(int j=0; j< Mapa_enemigos; j++){
        if(i!=j){
            if(collides){
                Move_colliding_enemigo[i];
                Move_colliding_enemigo[j];
            }
        }
    }
}

```

función desplazamiento después de colisión  
 Mov\_colidiendo\_enemigo (int Key) :

- if (Key == 0) {  
 Mapaspire → move\_left;  
 Mapaspire → move\_up;
 }
- if (Key == 1) {  
 Mapaspire [Key] → move\_right;  
 Mapaspire [Key] → move\_up;
 }
- if (Key == 2) {  
 Mapaspire [Key] → move\_left;  
 Mapaspire [Key] → move\_down;
 }
- if (Key == 3) {  
 Mapaspire [Key] → move\_right;  
 Mapaspire [Key] → move\_down;
 }

## Colisión Personaje - Enemigos:

- función principal (recorre lista de enemigos)
- función secundaria (desplaza al enemigo para evitar otra colisión)

función primaria:

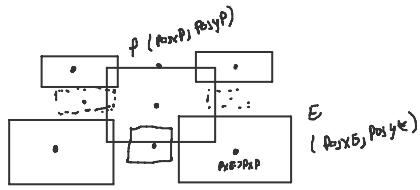
```

Colision_personaje_enemigos();
for(int i=0; i< Mapa_enemigosSize(); i++){
    if (!pL . collides_with_item( Mapa_enemigos[i] )) {
        //función secundaria:
    }
}

```

función secundaria:

• PosxP, PosyP ; PosxE, PosyE.



```

    si PosxE > PosxP
    Si PosxE < PosxP
    Si PosxE == PosxP

    if( PosxE > PosxP ) {
        Enemigo[i].move_right();
    } else if( PosxE < PosxP ) {
        Enemigo[i].move_left();
    } else {
        if( PosyE > PosyP ) {
            Enemigo[i].move_down();
        } else if( PosyE < PosyP ) {
            Enemigo[i].move_up();
        }
    }
}

```