Código do Player (Create):

```
idle_state = new state();
walk_state = new state();
atk_state = new state();
hurt_state = new state();
died_state = new state();
#region geral vars

keys = 0;

#endregion

#region stats
if (!variable_global_exists("level")) global.level = 1;
if (!variable_global_exists("xp")) global.xp = 0;

level = global.level;
xp = global.xp;


max_life = array_create(11, 0);
for (var i = 0; i <= 10; i++)
{
    max_life[i] = 15 + (i * 5);
}

max_xp = array_create(11, 0);
for (var i = 0; i <= 10; i++)
{
    max_xp[i] = 50 + (i * 50);
}

max_damage = array_create(11, 0);
for(var i = 0; i <= 10; i++)
{
    max_damage[i] = 1 + (i * 1);
}
#endregion

#region input
up = noone;
down = noone;
left = noone;
```

```
right = noone;
atk = noone;

global.pause = noone;
#endregion

#region movement
vlh = 0;
vlv = 0;
pVl = 2;
ldir = 0
depth = 0;
#endregion

#region life_system
i_frames = 0;
hurt_state = new state();
if(global.dificulade == 0 || global.dificulade == 1)
{
	if (!variable_global_exists("player_life") || global.reset_life)
	{
	    global.player_life = -1;
	    global.reset_life = false;
	}

	if (global.player_life == -1)
	{
	    if (level < array_length(max_life))
	    {
	        dano_num = max_damage[level];
	        plife = max_life[level];
	    }
	    else
	    {
	        var _last_index = array_length(max_life) - 1;
	        dano_num = max_damage[_last_index];
	        plife = max_life[_last_index];
	    }
	}
	else
	{
	    if (level < array_length(max_life))
	    {
	        dano_num = max_damage[level];
```

```
            plife = global.player_life;
        }
        else
        {
            var _last_index = array_length(max_life) - 1;
            dano_num = max_damage[_last_index];
            plife = global.player_life;
        }
    }

    if (is_array(plife))
    {
        plife = plife[0];
    }

    get_current_life = function()
    {
        if (is_array(plife))
        {
            return real(plife[0]);
        }
        return real(plife);
    }
}
else
{
    plife = 1;

    get_current_life = function()
    {
        if (is_array(plife))
        {
            return plife = 1
        }
        return real(plife);
    }
}
#endregion
#region save_system
if (!variable_global_exists("level")) global.level = 1;
if (!variable_global_exists("xp")) global.xp = 0;
if (!variable_global_exists("load_save")) global.load_save = false;
if (!variable_global_exists("load_x")) global.load_x = 0;
if (!variable_global_exists("load_y")) global.load_y = 0;
```

```
if (!variable_global_exists("load_vida")) global.load_vida = 1;
global.load_vida = real(global.load_vida);

if (global.load_save)
{
    x = global.load_x;
    y = global.load_y;
    level = global.level;
    xp = global.xp;
    if (is_array(global.load_vida))
    {
        var _idx = clamp(level, 0, array_length(global.load_vida) - 1);
        plife = real(global.load_vida[_idx]);
    }
    else
    {
        plife = real(global.load_vida);
    }
    global.load_save = false;
}

if (is_array(plife))
{
    plife = plife[0];
}
plife = real(plife);

global.player_life = plife;

#endregion

#region idle_state
idle_state.start = function()
{
    image_alpha = 1;
    var _sprite = define_sprite(ldir, sPlayer_Direita, sPlayer_Costas,
sPlayer_Esquerda, sPlayer_Frente);
    sprite_index = _sprite;
    image_index = 0;
}

idle_state.play = function()
{
    if (up xor down or left xor right)
```

```
    {
        trade_state(walk_state);
    }

    if (atk)
    {
        trade_state(atk_state);
    }

    if (get_current_life() <= 0)
    {
        trade_state(died_state);
    }
}
#endregion

#region walk_state
walk_state.start = function()
{
    image_alpha = 1;
    vlv = (down - up) * pVl;
    vlh = (right - left) * pVl;

    if (vlv != 0 or vlh != 0)
    {
        ldir = (point_direction(0, 0, vlh, vlv) div 90);
    }

    sprite_index = define_sprite(ldir, sPlayer_Andando_Direita,
sPlayer_Andando_Costas, sPlayer_Andando_Esquerda, sPlayer_Andando_Frente);
    image_index = 0;
}

walk_state.play = function()
{
    vlv = (down - up) * pVl;
    vlh = (right - left) * pVl;

    if (vlv != 0 or vlh != 0)
    {
        ldir = (point_direction(0, 0, vlh, vlv) div 90);
    }
```

```
    sprite_index = define_sprite(ldir, sPlayer_Andando_Direita,
sPlayer_Andando_Costas, sPlayer_Andando_Esquerda, sPlayer_Andando_Frente);

    if (!up and !down and !left and !right)
    {
        trade_state(idle_state);
    }

    if (atk)
    {
        vlv = 0;
        vlh = 0;
        trade_state(atk_state);
    }

    if (get_current_life() <= 0)
    {
        trade_state(died_state);
    }
}
#endregion

#region atk_state
atk_state.start = function()
{
    sprite_index = define_sprite(ldir, sPlayer_Atk_Direita, sPlayer_Atk_Costas,
sPlayer_Atk_Esquerda, sPlayer_Atk_Frente);
    image_index = 0;

    var _x = x + lengthdir_x(6,ldir*90)
    var _y = y + lengthdir_y(7,ldir*90)

    dano = instance_create_depth(_x,_y,depth, obj_player_dano)
}

atk_state.play = function()
{
    vlv = (down - up) * pVl;
    vlh = (right - left) * pVl;

    if (image_index >= image_number - 1)
    {
        trade_state(idle_state);
    }
```

```
    if (get_current_life() <= 0)
    {
        trade_state(died_state);
    }
}

atk_state.finish = function()
{
    if (instance_exists(dano))
    {
        instance_destroy(dano)
    }
}
#endregion

#region hurt_state
hurt_state.start = function()
{
    plife = get_current_life() - 10
    sprite_index = sPlayer_Hurt
    image_index = 0

    i_frames = 60;
    hurt_freeze = 10;

    sprite_index = define_sprite(ldir, sPlayer_Direita, sPlayer_Costas,
sPlayer_Esquerda, sPlayer_Frente);
    image_index = 0;

    vlv = 0;
    vlh = 0;
    atk = false;

    global.player_life = plife;
}

hurt_state.play = function()
{
    if (hurt_freeze > 0)
    {
        hurt_freeze--;
        vlv = 0;
        vlh = 0;
```

```
    }
    else
    {
        trade_state(walk_state);
    }

    if (i_frames > 0)
    {
        i_frames--;
        image_alpha = (i_frames mod 6 < 3) ? 0.5 : 1;
    }
    else
    {
        image_alpha = 1;
    }

    if (get_current_life() <= 0)
    {
        trade_state(died_state);
    }
}

hurt_state.finish = function()
{
    image_alpha = 1;
    i_frames = 0;
}
#endregion

#region died_state
died_state.start = function()
{
    if (get_current_life() <= 0)
    {
        Obj_Controller.gmov = 1;
        sprite_index = sPlayer_Died
        image_index = 0
        vlv = 0
        vlh = 0
    }
}

died_state.play = function()
{
```

```
    if(get_current_life() > 0)
    {
        trade_state(idle_state)
    }
}
#endregion
start(idle_state);
```

Código do Player (Step event):

```
global.player_life = get_current_life();
if(global.pause)
{
        image_speed=0;
        exit;
}
else{
        image_speed= 1;
}


depth = -y;

up = keyboard_check(vk_up) or keyboard_check(ord("W"));
down = keyboard_check(vk_down) or keyboard_check(ord("S"));
left = keyboard_check(vk_left) or keyboard_check(ord("A"));
right= keyboard_check(vk_right) or keyboard_check(ord("D"));
atk = keyboard_check_pressed(vk_space)


if xp >= max_xp[level]
{
        xp = xp - max_xp[level];
        level += 1;

        plife = max_life[level]
        dano_num = max_damage[level]
}

// player andando
// conectando velocidade com o eixo

repeat(abs(vlh))
```

```
{
if (!place_meeting(x + sign(vlh), y, obj_colisor))
{
        x = x + sign(vlh)
}
else
{
        vlh = 0
}
}

repeat(abs(vlv))
{
if (!place_meeting(x, y + sign(vlv), obj_colisor))
{
        y = y + sign(vlv)
}
else
{
        vlv = 0
}
}


// verificando se o player parou de colidir com recursos

if (!place_meeting(x, y, Obj_alavanca))
{
instance_destroy(ob_ebutton)
}

if (instance_number(ob_ebutton) > 1)
{
instance_destroy(ob_ebutton)
}

// rodando state machine

play();
```

Código de uma das portas:

```
instance_create_depth(x, y, depth, ob_ebutton)
obj = instance_nearest(x, y, obj_door2);
```

```
if obj.door_open == true
{instance_destroy(ob_ebutton)}

if !instance_exists(obj_enemie_goblin) and !instance_exists(obj_enemie_goblin1)
{


        if keys == 1 and keyboard_check(ord("E"))
        {
        obj.door_open = true;
        keys = 0
        }



                if obj_door2.image_index == 4 and room == DUNGEON_2
        {

        room_goto(DUNGEON_3)
        }
        else if obj_door2.image_index == 4 and room == DUNGEON_3
        {
        room_goto(DUNGEON_4)
        }

else if obj_door2.image_index == 4 and room == DUNGEON_5
        {
        room_goto(DUNGEON_6)
        }

}
```

Código do Karazor (Create):

```
#region var
var _fps = game_get_speed(gamespeed_fps);
spawn_cooldown = _fps;
hunt_min_time = _fps * 15;
hunt_max_time = _fps * 20;
summon_state = new state();
range = 300;
time_state = _fps*5
#endregion
```

```
event_inherited();

max_hp = irandom_range(20,25)
elife = max_hp

#region sprites
sprite =
{
        attack : spr_goblin_attack_1,
        death : spr_goblin_death_1,
        hurt : spr_goblin_hurt_1,
        idle : spr_goblin_idle_1,
        walk : spr_goblin_walk_1
}
#endregion

image_xscale = 1.5;
image_yscale = 1.5;

#region check_dist
check_player_range = function()
{
   if (instance_exists(Obj_Player))
   {
      var _dist = point_distance(x, y, Obj_Player.x, Obj_Player.y);
      var _raio_boss = 120;

      if (_dist <= _raio_boss)
      {
         target = Obj_Player.id;
         trade_state(summon_state);
         return true;
      }
   }
   return false;
}
#endregion

#region idle_state
idle_state.start = function()
{
        sprite_index = sprite.idle;
        image_index = 0;
```

```
	timer_state = irandom_range(game_get_speed(gamespeed_fps) *
4,time_state);
}

idle_state.play = function()
{
	timer_state--;
	check_player_range()
	if(timer_state <= 0)
	{
		new_state = choose(walk_state,idle_state,walk_state);
		trade_state(new_state);
	}
}
#endregion

#region walk_state
walk_state.start = function()
{
	sprite_index = sprite.walk;
	image_index = 0;
	timer_state = time_state;
	destiny_x = irandom(room_width);
	destiny_y = irandom(room_height);
	xscale = sign(destiny_x - x)
}

walk_state.play = function()
{
	timer_state--;
	check_player_range()
	if(timer_state <= 0)
	{
		new_state = choose(idle_state,walk_state,idle_state);
		trade_state(new_state);
	}
	mp_potential_step_object(destiny_x,destiny_y,1,obj_colisor);
}
#endregion

#region death_state
death_state.start = function()
{
	sprite_index = sprite.death;
```

```
	image_index = 0;
	vlv = 0
	vlh = 0
}

death_state.play = function()
{
	if(image_index >= image_number - .5)
	{
		if (instance_exists(Obj_Player)) {
			Obj_Player.xp += 60;
		}
		instance_destroy();
	}
}
#endregion

#region hunt_state
hunt_state.start = function()
{
	sprite_index = sprite.walk;
	image_index = 0
	create_warn();
	if(instance_exists(Obj_Player))
	{
		target = Obj_Player.id
	}
	if (alarm[1] == -1)
   {
      alarm[1] = irandom_range(hunt_min_time, hunt_max_time);
   }
}

hunt_state.play = function()
{
	if(!instance_exists(Obj_Player))
	{
		target = noone
		trade_state(summon_state);
	}
	mp_potential_step_object(target.x,target.y,1,obj_colisor);
	var _dist = point_distance(x,y,target.x,target.y);
	if(_dist <=range)
	{
```

```
                trade_state(attack_state)
        }
        xscale = sign(target.x - x);
}

hunt_state.finish = function()
{
        vlv = 0
        vlh = 0
}
#endregion

#region attack_state
attack_state.start = function()
{
        sprite_index = sprite.attack
        image_index = 0
        var _x = x + lengthdir_x(10,dir*90)
        var _y = y + lengthdir_y(10,dir*90)
        dano = instance_create_depth(_x,_y,depth, obj_goblin_damage_boss)
}

attack_state.play = function()
{
        if(image_index >= image_number - .5)
        {
                trade_state(hunt_state)
        }
}
#endregion

#region spawn_state
summon_state.start = function()
{
        sprite_index = sprite.idle
   image_index = 0;
   alarm[0] = spawn_cooldown;
   target_goblin_count = irandom_range(2, 4);
}

summon_state.play = function()
{

}
```

```
summon_state.finish = function()
{

}
#endregion

start(idle_state);
```

Código do Sistema de pausa, tela de gameover e sistema de save(create):

```
global.load_save = false;
global.load_x = 0;
global.load_y = 0;
global.load_vida = 1;
global.load_room = room;
global.reset_life = false;


// Menu de pausa
global.pause = false;
options = ["Continuar", "Salvar o Jogo", "Carregar Save", "Opções", "Sair"];
op_leng = array_length(options);
select = 0;

// Menu de Game Over
gmov = 0;
gmov_alpha = 0;
gmov_scale = 0.5;
gmov_timer = 0;
```

Código do Sistema de pausa, tela de gameover e sistema de save(Step event):

```
if (keyboard_check_pressed(vk_escape))
{
    global.pause = !global.pause;
}



if (gmov == 1)
{
    gmov_timer++;
```

```
    gmov_alpha = clamp(gmov_alpha + 0.02, 0, 1);
    gmov_scale = clamp(gmov_scale + 0.01, 0.5, 2);

    if (gmov_timer > game_get_speed(gamespeed_fps) * 5)
    {
                    global.reset_life = true;
        room_goto(MENU_INICIAL);

        gmov = 0;
        gmov_alpha = 0;
        gmov_scale = 0.5;
        gmov_timer = 0;



    }
}
```

Código do Sistema de pausa, tela de gameover e sistema de save(Draw GUI):

```
#region pause_menu
var gui_w = display_get_gui_width();
var gui_h = display_get_gui_height();
var margin = 30;
var m_x = device_mouse_x_to_gui(0);
var m_y = device_mouse_y_to_gui(0);
if (global.pause)
{

    // Fundo escurecido
    draw_set_alpha(.8);
    draw_set_color(c_black);
    draw_rectangle(0, 0, gui_w, gui_h, false);
    draw_set_alpha(1);
    draw_set_color(c_white);
    // Título
    draw_set_font(ft_menu_pause);
    draw_set_halign(fa_center);
    draw_text(gui_w / 2, gui_h / 8, "Jogo Pausado");
    draw_set_valign(fa_middle);
    // Botões centrais
    for (var i = 0; i < op_leng; i++)
    {
        var gui_h2 = gui_h / 2 + (margin * i);
```

```
        var string_w = string_width(options[i]);
        var string_h = string_height(options[i]);
        // Verifica hover
        if (point_in_rectangle(m_x, m_y, gui_w / 2 - string_w / 2, gui_h2 - string_h / 2,
        gui_w / 2 + string_w / 2, gui_h2 + string_h / 2))
        {
            draw_set_color(c_red);
            select = i;
            if (mouse_check_button_pressed(mb_left))
            {
                // Continuar
                if (select == 0)
                {
                    global.pause = false;
                }
                // SALVAR JOGO
                                    if (select == 1)
                                    {
                                        global.pause = false;
                                        if (file_exists("save.sav")) file_delete("save.sav");

                                        ini_open("save.sav");

                                        // Salva posição atual e status do player
                                        if (instance_exists(Obj_Player))
                                        {
                                            ini_write_real("Player", "x_atual", Obj_Player.x);
                                            ini_write_real("Player", "y_atual", Obj_Player.y);
                                            ini_write_real("Life", "vida_atual",
real(Obj_Player.plife));

ini_write_real("Level","Level_atual",Obj_Player.level);

ini_write_real("Level","xp_atual",Obj_Player.xp);

ini_write_real("Dificuldade","Dificuldade_atual",global.dificulade);
                                        }

                                        ini_write_real("Game", "current_room", room);
                                        ini_close();

                                        show_debug_message("✅ Jogo salvo com
sucesso!");
                                    }
```

```
                    // CARREGAR JOGO
                    if (select == 2)
                    {
                        global.pause = false;
                        if (file_exists("save.sav"))
                        {


                            ini_open("save.sav");
                            var _x = ini_read_real("Player", "x_atual", 0);
                            var _y = ini_read_real("Player", "y_atual", 0);
                            var _vida = ini_read_real("Life", "vida_atual", 1);
                            var _saved_room = ini_read_real("Game",
"current_room", room);
                                    var _level =
ini_read_real("Level","Level_atual",1);
                                    var _xp = ini_read_real("Level","xp_atual",0)
                                    var _dif =
ini_read_real("Dificuldade","Dificuldade_atual",0);
                            ini_close();
                            if (room_exists(_saved_room))
                            {
                                // guarda dados antes de mudar de room
                                global.load_x = _x;
                                global.load_y = _y;
                                global.load_vida = real(_vida);
                                global.load_room = _saved_room;
                                        global.level = _level;
                                        global.xp = _xp;
                                        global.dificulade = _dif;
                                global.load_save = true;
                                room_goto(_saved_room);
                            }
                        }
                    }
            // Sair do jogo
            if (select == 4)
            {
                game_end();
            }
        }
    }
```

```
        else
        {
            draw_set_color(c_white);
        }

        draw_text(gui_w / 2, gui_h2, options[i]);
    }
    draw_set_halign(-1);
    draw_set_valign(-1);
}
#endregion

#region gameover_menu

if (gmov == 1)
{
    // Fundo escurecido
    draw_set_colour(c_black);
    draw_set_alpha(gmov_alpha * 0.95);
    draw_rectangle(0, 0, gui_w, gui_h, false);

    // Texto "GAME OVER"
    draw_set_colour(c_red);
    draw_set_alpha(gmov_alpha);
    draw_set_font(ft_menu_pause);
    draw_set_halign(fa_center);
    draw_set_valign(fa_middle);

    draw_text_transformed(
        gui_w / 2,
        gui_h / 2,
        "GAME OVER",
        gmov_scale,
        gmov_scale,
        0
    );

    // Restaura configurações de desenho
    draw_set_halign(fa_left);
    draw_set_valign(fa_top);
    draw_set_colour(c_white);
    draw_set_alpha(1);
}
```

#endregion