Obj_Game

```
rollback_define_input(
{
    left: ord("A"),
    right: ord("D"),
    up: ord("W"),
    down: ord("S"),
    r: ord("R"),
    fire: mb_left,
    mb_x: m_axisx,
    mb_y: m_axisy,
});

rollback_define_player(obj_player, "Instances");

if (!rollback_join_game())
{
        rollback_create_game(2, false);
}
```

Para iniciar um jogo online usaremos as funções rollback do gamemaker.

na função rollback_define-input estamos coletando os tipos de entrada do usuário e guardaremos em uma variável, exemplo w será guardada na variável up,

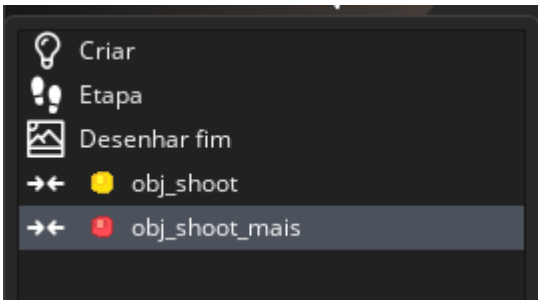
na funçaõ rollback_define_player definimos o nosso objeto player que no caso é nosso objeto jogável, e damos um id a ele que nesse caso é “instances”.

temos uma estrutura condicional que poderíamos interpretar como, “se a conexão for diferente de entrando, crie uma sala”, para isso usamos a rollback_join_player que é ima função para ver se tem outra pessoa se conectando, e a função rollback_create_game, para inicar um novo jogo, ela pede dois parâmetros, o número de jogadores, e um booleano que define

se o jogo está online ou não, onde false é online e true é offline
.

Obj_Player

Eventos necessários no objeto player



Criar, Etapa, Desenhar fim, Dois eventos de colisão.

Criando as variáveis para o nosso player(Evento criar)

```
shoot = 0;
direita = 0;
esquerda = 0;
cima = 0;
baixo = 0;
r = 0;


global.start = false;

spd = 3.5;
hspd = 0;
vspd = 0;
cooldown = 0;
life_mais = 2;
life_menos = -2;
dano = 1;
reiniciar = 0;
vencedor = "";


gravidade = 0.3;
```

fazendo nosso personagem nascer:

```
if (player_id == 0)
{
        x = 50;
        y = room_height / 2;
        sprite_index = spr_idle_menos
}
else if (player_id == 1)
{
        x = room_width - 50;
        y = room_height / 2;
        image_xscale = -1;

}
```

Quando iniciamos um jogo será criada duas instâncias automaticamente, que no caso seria nosso obj_player que definimos la no obj_game. O que diferencia essas duas instâncias são os id's, aqui verificamos os id's para um player nascer longe do outro.

Fazendo nosso player andar(Evento etapa)

```
//MOVIMENTAÇÃO
var _input = rollback_get_input();
direita = _input.right;
esquerda = _input.left;
cima = _input.up;
baixo = _input.down;
shoot = _input.fire;
r = _input.r

hspd = (direita - esquerda) * spd;

vspd = (baixo - cima) * spd;
```

usamos a função rollback_get_input para pegar as entradas dos usuários e guardamos em variáveis do gamemaker. a variável hsdp é a variáveis que vai fazer o X do nosso player andar e a vspd é a do Y.

Fazendo nosso personagem andar e colidir, também deixando a gravidade funcionando.

```
if !place_meeting(x, y + 1, obj_chao){
vspd += gravidade;
}

if place_meeting(x + hspd, y, obj_chao){
    while !place_meeting(x + sign(hspd), y, obj_chao){
          x += sign(hspd)
      }
hspd = 0;
}

x += hspd;

if place_meeting(x , y + vspd, obj_chao){
    while !place_meeting(x, y + sign(vspd), obj_chao){
          y += sign(vspd);
      }
vspd = 0;
}

y += vspd;
```

link do vídeo que explica melhor a movimentação e a gravidade:
Movimentação e Colisão | Fazendo um jogo Plataforma #1| GMS2

Controlando as sprites do player:

```
//CONTROLE DE SPRITES
if(hspd != 0){

if(player_id == 0){

sprite_index = spr_run_menos;
image_xscale = sign(hspd);

}else{
sprite_index = spr_run;
image_xscale = sign(hspd);
}
}else{
if(player_id == 0){

sprite_index = spr_idle_menos;

}else{
sprite_index = spr_idle;
}
}
```

Criando o projétil:

Para isso usamos instance_create_layer passando os parâmetros necessários.

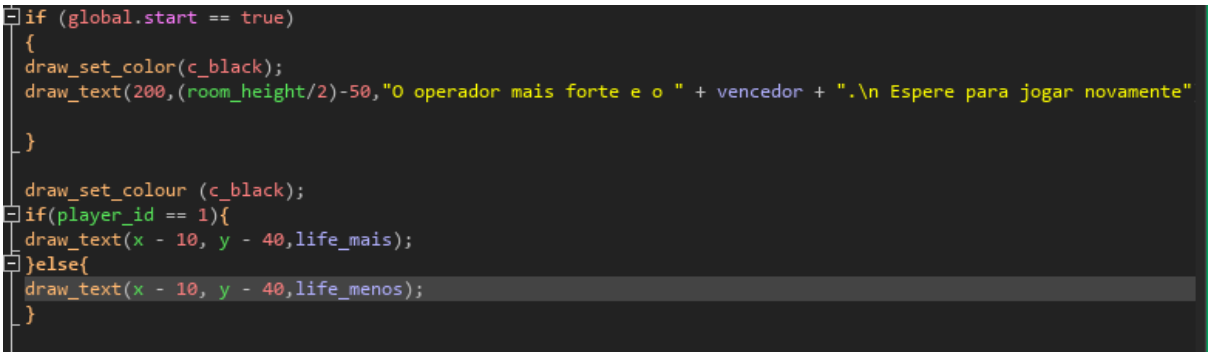
```
if (shoot)
 {

    if(cooldown <= 1){

        if(player_id == 1){
         var _proj = instance_create_layer(x+1, y+2, layer, obj_shoot_mais);
        _proj.image_angle = point_direction(x, y, _input.mb_x, _input.mb_y);
        _proj.speed = 4;
        _proj.direction = point_direction(x, y, _input.mb_x, _input.mb_y);
        _proj.player = self;
        cooldown = 40;
        }else{

        var _proj = instance_create_layer(x+1, y+2, layer, obj_shoot);
        _proj.image_angle = point_direction(x, y, _input.mb_x, _input.mb_y);
        _proj.speed = 4;
        _proj.direction = point_direction(x, y, _input.mb_x, _input.mb_y);
        _proj.player = self;
        cooldown = 40;
        }
    }
}

cooldown -= 1;
```

Dando um restart para que não seja necessário atualizar a página

```
if(global.start){
 instance_destroy(obj_shoot);
 instance_destroy(obj_shoot_mais);
 reiniciar += 1;
if(reiniciar == 400){
     global.start = false;
     vencedor = "";
 }
 }
```

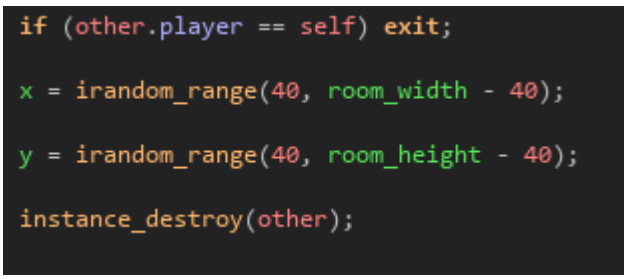
TXT(desenho fim)

Criando um texto em cima do nosso obj_player e fazendo uma estrutura condicional para mostrar um texto com o vencedor.



```
if (global.start == true)
 {
 draw_set_color(c_black);
 draw_text(200,(room_height/2)-50,"O operador mais forte e o " + vencedor + ".\n Espere para jogar novamente"

 }

 draw_set_colour (c_black);
if(player_id == 1){
 draw_text(x - 10, y - 40,life_mais);
}else{
 draw_text(x - 10, y - 40,life_menos);
 }
```

Tiro(Colisão obj_shoot)



```
if (other.player == self) exit;

x = irandom_range(40, room_width - 40);

y = irandom_range(40, room_height - 40);

instance_destroy(other);
```

destruímos a instância do outro objeto e destruída e ressurgida em um x e um um y aleatório.

Bloco de código que auxilia no restart do jogo, e que também define o vencedor.

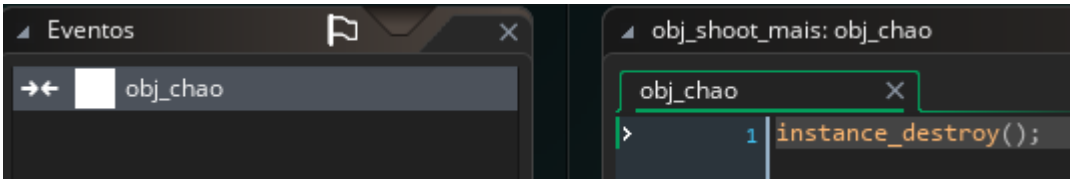
```
if(life_mais == 1){
 global.start = true;
 }


if(global.start){
 life_mais = 11;
 other.player.life_menos = -10;
 vencedor = "menos";
 }


if(player_id == 1){
 self.life_mais = life_mais - dano;
 reiniciar = 0;
 other.player.reiniciar = 0;
 other.player.vencedor = "";
}else{
 self.life_menos = life_menos + dano;
 reiniciar = 0;
 other.player.reiniciar = 0;
 other.player.vencedor = "";
 }
```

Mesmo código só que para o tiro do “Mais”

```
if(life_menos == -1){
 global.start = true;
 }

if(global.start){
 life_menos = -11;
 other.player.life_mais = 10;
 vencedor = "mais";
 }


if(player_id == 1){
 self.life_mais = life_mais - dano;
 reiniciar = 0;
 other.player.reiniciar = 0;
 other.player.vencedor = "";
}else{
 self.life_menos = life_menos + dano;
 reiniciar = 0;
 other.player.reiniciar = 0;
 other.player.vencedor = "";
 }
```

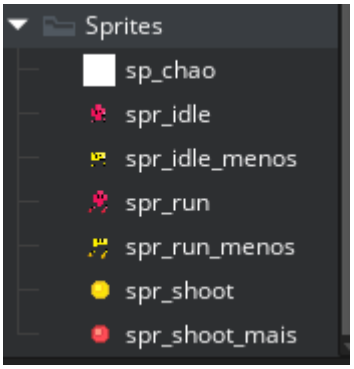
Obj_Shoot

O único evento que nosso tiro tem é uma colisão com a parede.
Também lembrando que temos dois objetos tiros um para o menos e um para o mais, os dois tem os mesmos eventos e mesmos códigos



Todos Objetos:

Todos Sprites:



link da documentação rollback do gamemaker:

https://beta-manual.yoyogames.com/GameMaker_Language/GML_Reference/Rollback/Rollback_Multiplayer.htm