

23 DE MARÇO DE 2024



A VIAGEM DE SLIMO

Como criar o jogo “A viagem de Slimo” na aplicação Game Maker

BORA TROCAR COMPANY

Sumário

1. Vocabulário	3
2. Instalação e Configuração	6
3. Criando Recursos.....	7
4. Criando os Objetos	11
5. Cenas	14
6. Eventos	16
7. Scripts e States.....	16
8. Personagem.....	17
9. Variáveis	18
10. Inimigo.....	24
11. Espada.....	28
12. Ataque espada (projétil).....	29
13. Hit box	31
14. Chave	32
15. Menu morte.....	32
16. Menu inicial.....	34
17. Pausa menu	35

1. Vocabulário

➤ Objetos e Estados:

- Personagem: Um elemento controlável dentro do jogo.
- Inimigo: Um elemento não controlável que geralmente oferece desafio ao jogador.
- Espada: Uma arma utilizada pelo personagem para atacar.
- Hit box: Uma área invisível ao redor de um objeto que define a área de colisão.
- Chave: Um objeto que permite o acesso a áreas bloqueadas ou desbloqueia novos recursos.
- Projétil: Um objeto em movimento que pode ser lançado ou disparado, como uma espada em movimento.
- Menu de morte: Uma tela exibida quando o personagem morre no jogo.
- Menu inicial: A tela exibida quando o jogo é iniciado.
- Menu de pausa: Uma tela exibida quando o jogo é pausado.
- Pausa: A ação de interromper temporariamente o jogo.

➤ Eventos:

- Criar: Um evento que ocorre quando um objeto é criado.
- Desenhar GUI: Um evento usado para desenhar elementos da interface gráfica.
- Etapa: Um evento que ocorre em cada frame do jogo.

➤ Ações e Funcionalidades:

- Colisão: O encontro de dois objetos no jogo.
- Gravidade: A força que atrai os objetos para baixo, simulando o efeito da gravidade na física do jogo.
- Movimentação: O deslocamento de um objeto no jogo.
- Ataque: A ação de causar dano a outro objeto ou personagem no jogo.
- Ativar: Mudar o estado de um objeto para permitir uma determinada funcionalidade.
- Destruir: Remover um objeto do jogo.
- Mostrar: Tornar visível um objeto ou elemento na tela.
- Esconder: Tornar um objeto ou elemento invisível na tela.
- Customizar: Ajustar o visual ou comportamento de um elemento do jogo.
- Abrir: Exibir um menu ou tela específica no jogo.
- Fechar: Ocultar um menu ou tela específica no jogo.

➤ **Variáveis:**

- Vida: A quantidade de saúde ou energia de um personagem ou objeto.
- Chaves: Um contador para rastrear quantas chaves foram coletadas.
- Morte: Uma variável que indica se o personagem morreu no jogo.
- Can_ataque: Uma variável que controla se o personagem pode ou não atacar.
- Arma_id: Uma variável que identifica a arma equipada pelo personagem.
- Arma_dir: Uma variável que armazena a direção para a qual a arma deve apontar.
- Arma_x e Arma_y: Variáveis que representam a posição da arma em relação ao personagem.

➤ **Controles e Comandos:**

- Teclado: Dispositivo de entrada usado para controlar o jogo.
- Botão: Um elemento do teclado ou controle do jogo que pode ser pressionado.
- ESC (Tecla de Escape): Uma tecla comumente usada para acessar menus ou sair de telas.

➤ **Listas:**

- Lista de objetos: Uma coleção de objetos no jogo.
- Lista de opções: Uma lista de itens ou ações disponíveis em um menu.

➤ **Condições e Verificações:**

- If: Uma instrução de controle de fluxo que executa um bloco de código se uma condição especificada for verdadeira.
- True: Um valor ou estado que indica que uma condição é verdadeira.
- False: Um valor ou estado que indica que uma condição é falsa.
- Condição: Uma expressão que pode ser avaliada como verdadeira ou falsa.

➤ **Iterações e Loops:**

- For: Uma estrutura de controle de fluxo usada para repetir um bloco de código um número específico de vezes.

Outros:

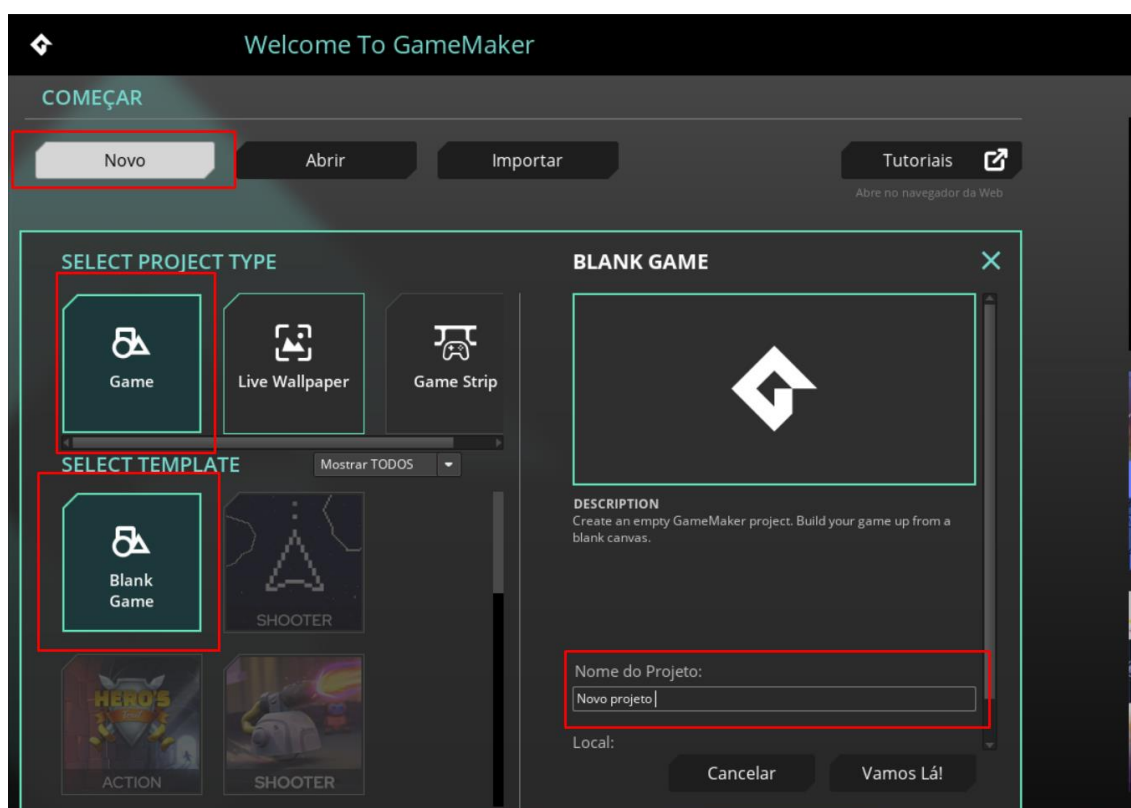
- Sprite: Uma imagem ou animação que representa um objeto no jogo.
- Camada: Uma camada de elementos gráficos em uma cena de jogo.

- Tela: A área visível do jogo na qual os elementos são exibidos.
- Mouse: Dispositivo de entrada usado para interagir com o jogo.
- Velocidade: A rapidez com que um objeto se move no jogo.
- Ângulo: A direção ou orientação de um objeto em relação a uma referência

2. Instalação e Configuração

Baixe e instale o Game Maker Studio, visite o site oficial do Game Maker Studio e siga as instruções para baixar e instalar o software em seu computador. Certifique-se de estar utilizando a versão compatível com o sistema operacional do seu computador.

Inicie o Game Maker Studio e crie um novo projeto, após a instalação, inicie o Game Maker Studio e clique em "Novo Projeto". Escolha o tipo de projeto que deseja criar (por exemplo, jogo 2D) e dê um nome ao seu projeto.



3. Criando Recursos

Os recursos incluem tudo o que será usado no jogo, como sprites, sons, fundos e scripts. Clique com o botão direito na pasta "Recursos" na barra lateral e selecione "Criar Recurso" para cada tipo de recurso que você precisar.

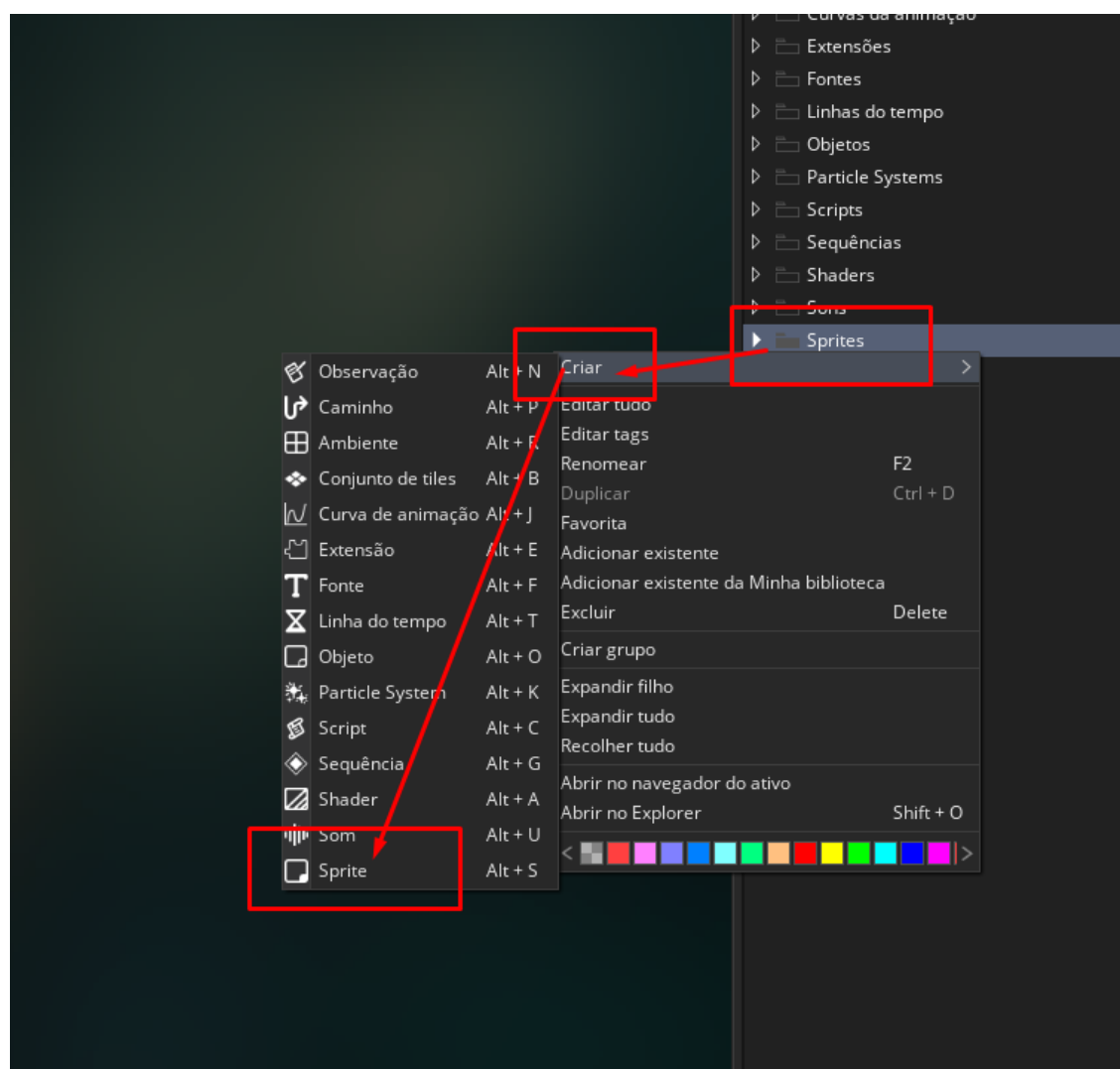
Criando o Payer

Descreve como criar e programar o personagem principal do jogo, incluindo movimento, animação, interação com o ambiente.

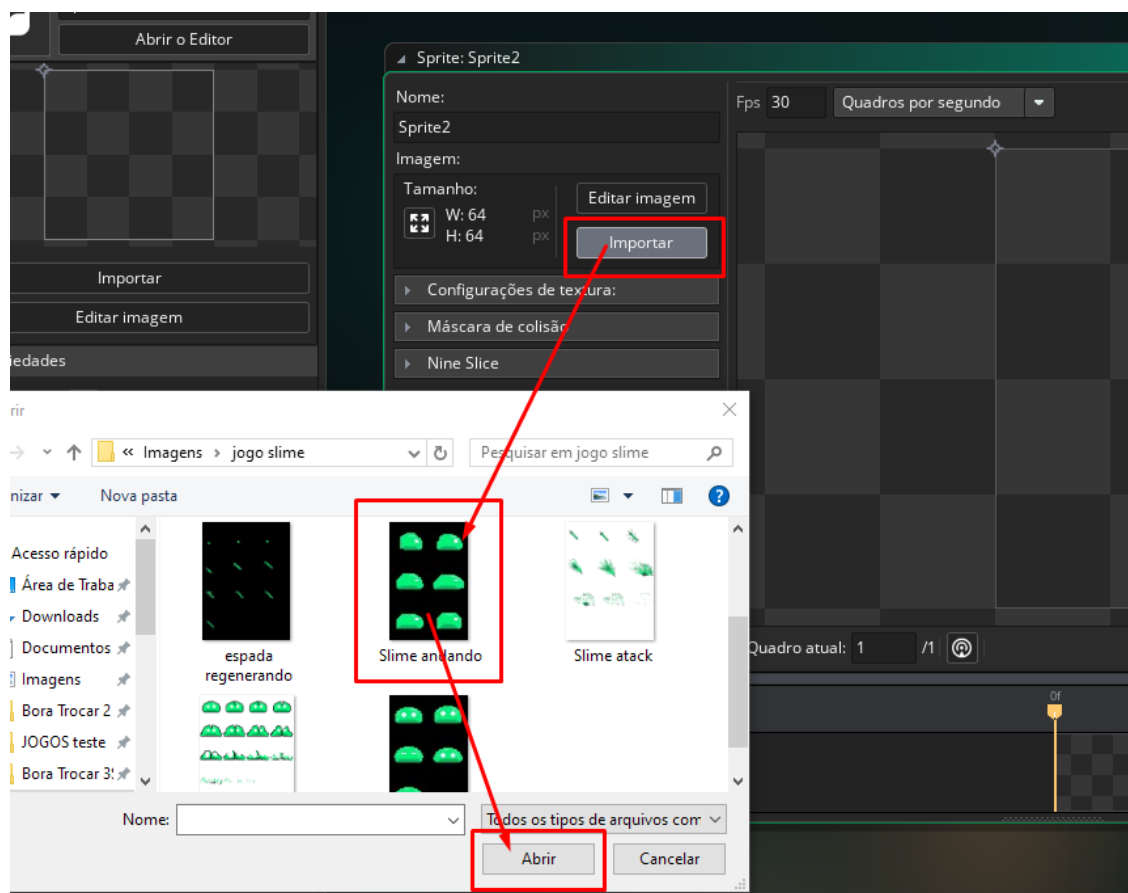
IMPORTANDO SPRITES

Para criar as sprites (desenhos) que serão usadas no jogo para o player, siga os passos abaixo:

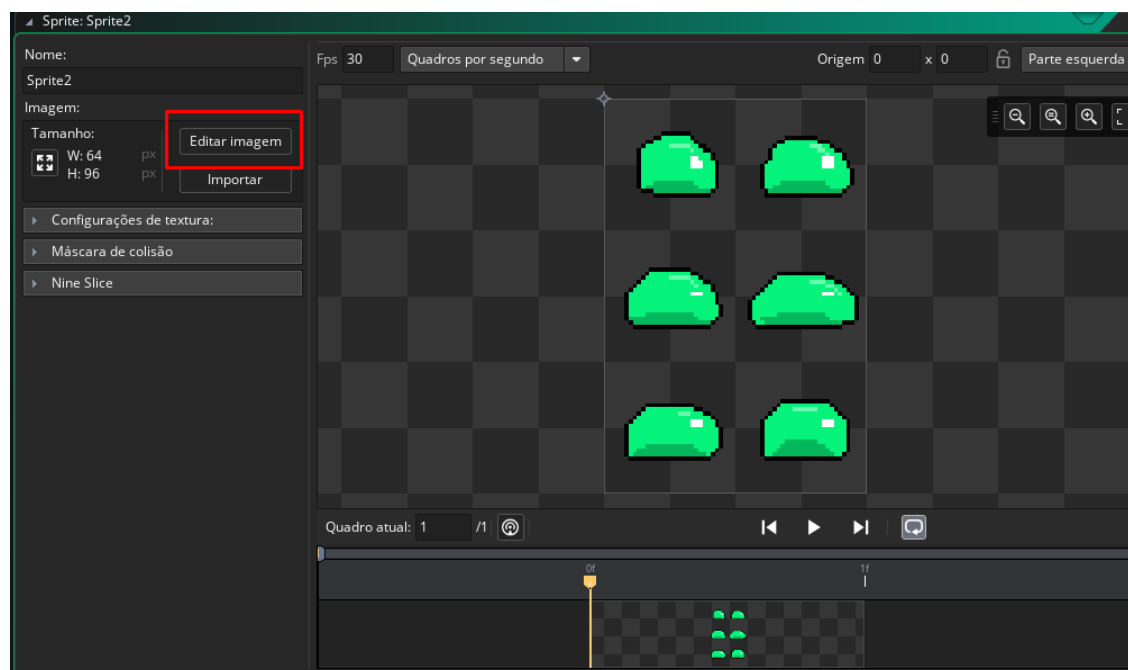
Crie a Sprite



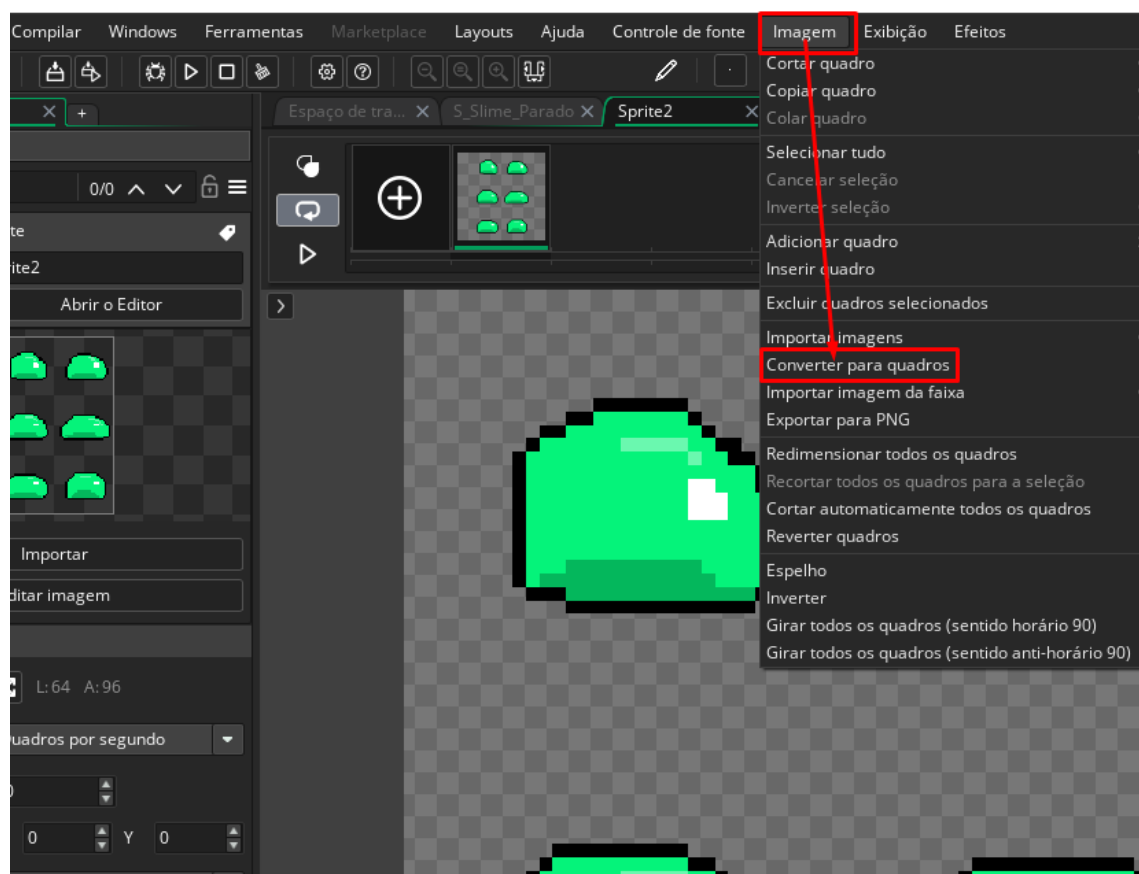
Importe o a imagem do personagem criado



Edite a imagem conforme sua preferência



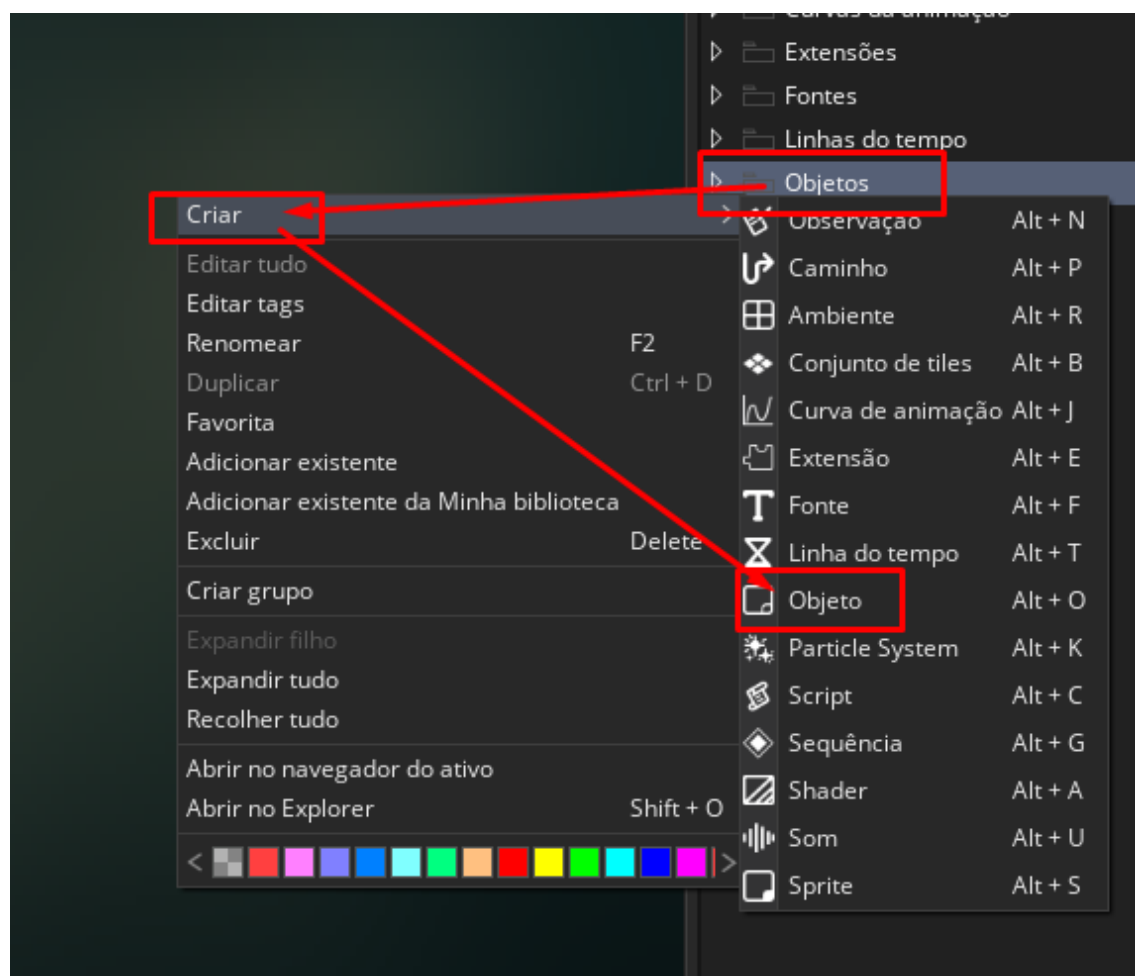
Converte a imagem em quadros



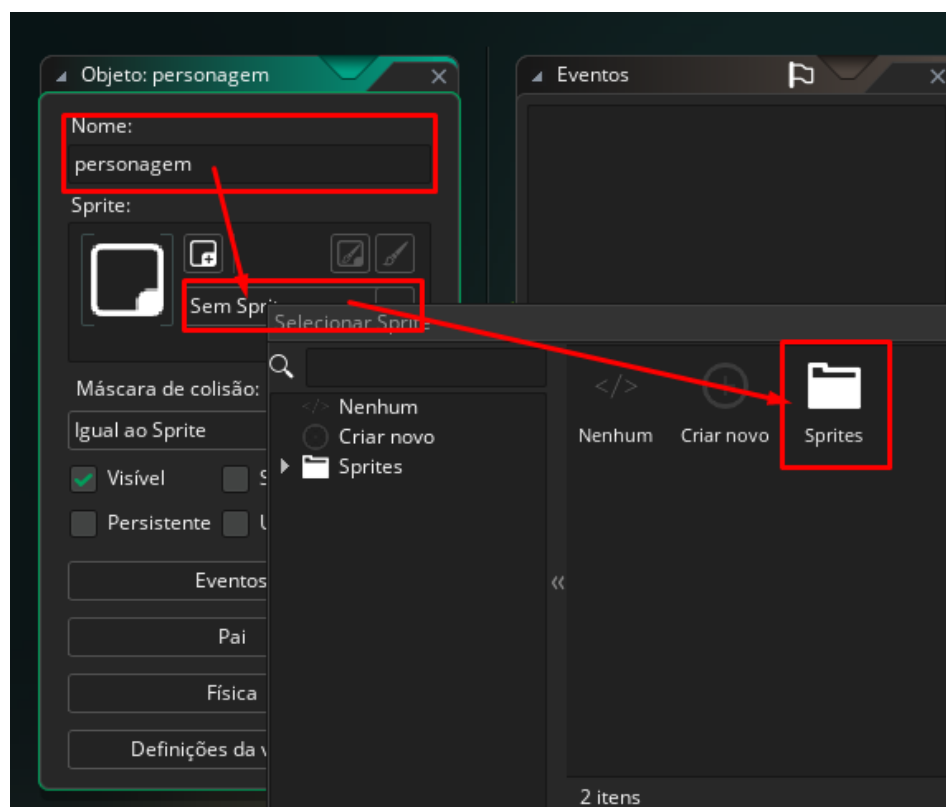
4. Criando os Objetos

No Game Maker, objetos são elementos fundamentais que representam entidades interativas no jogo. Eles desempenham um papel crucial na criação de jogos, pois são responsáveis por quase todas as interações visíveis e lógicas que ocorrem durante o jogo, sendo assim siga o passo para criação dos objetos

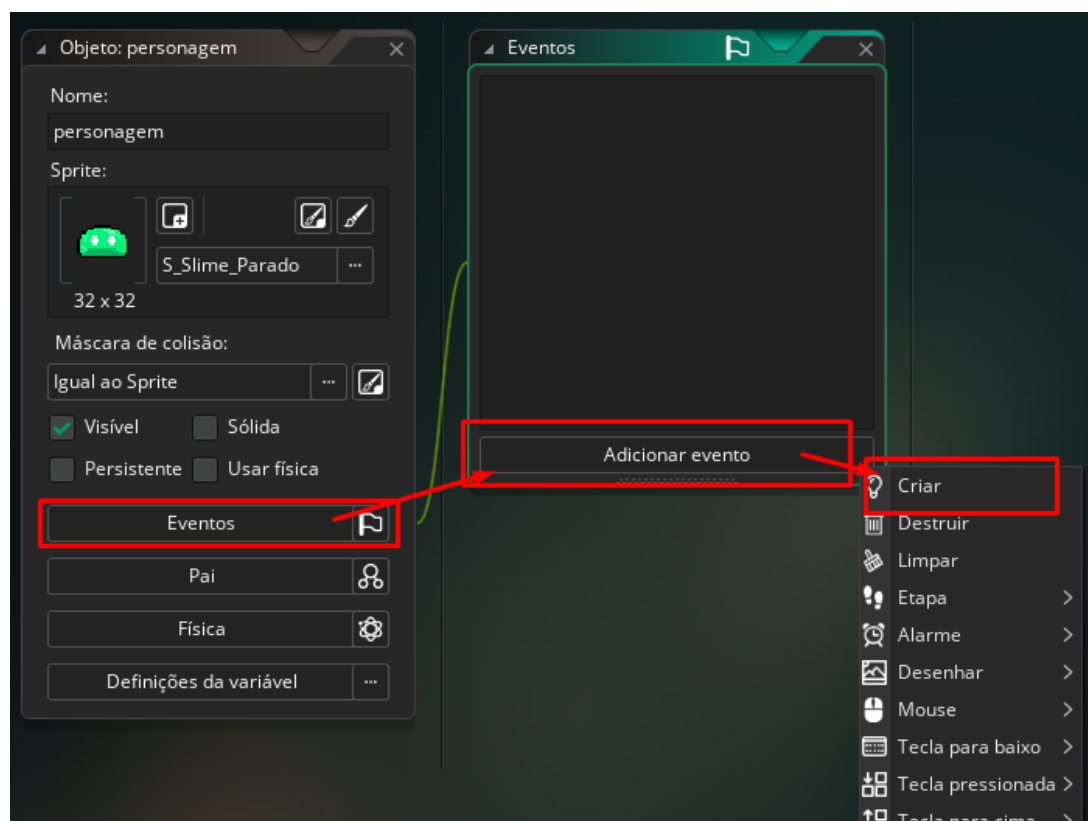
Crie o objeto



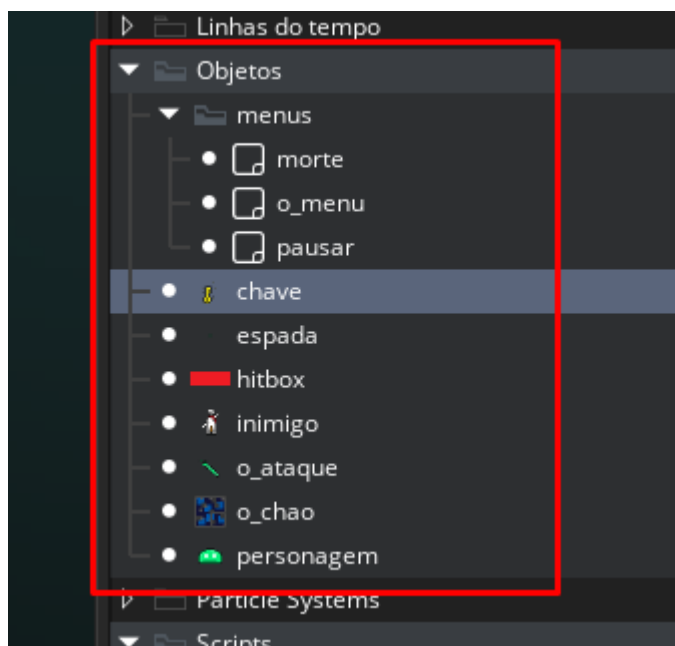
Vincule o objeto personagem a sprite player



Vincule ao evento – Criar os movimentos do personagem



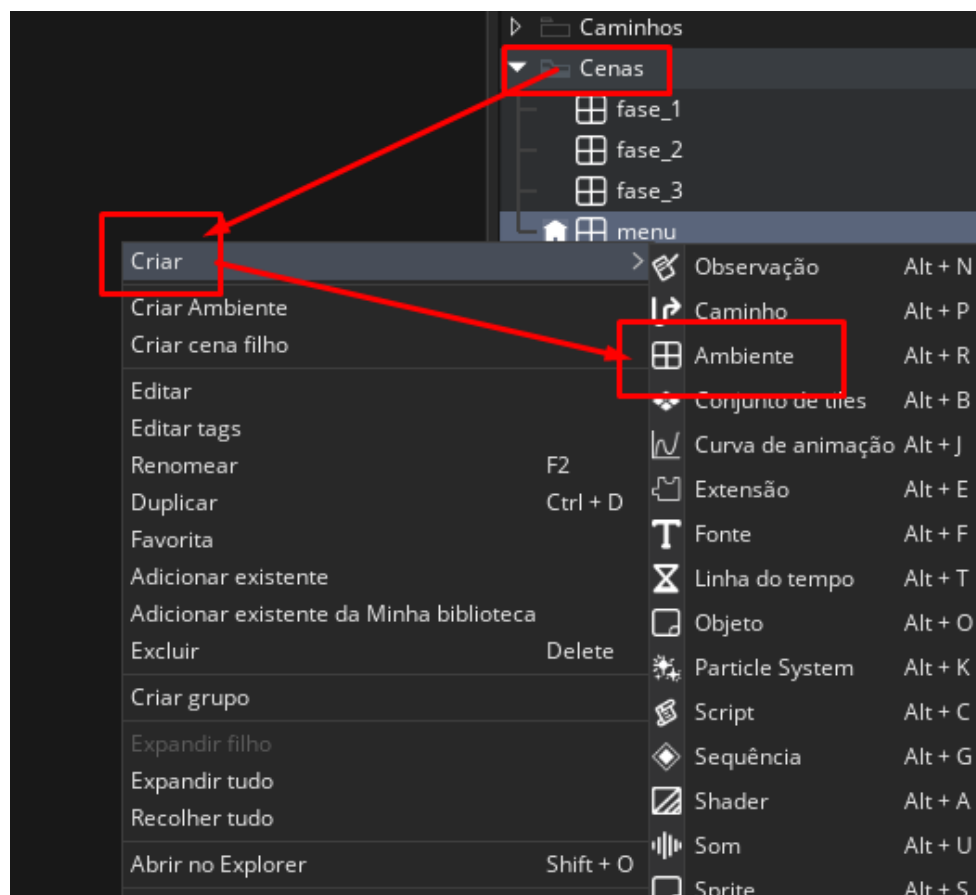
Para uma execução fiel do jogo e necessário a criação dos objetos abaixo



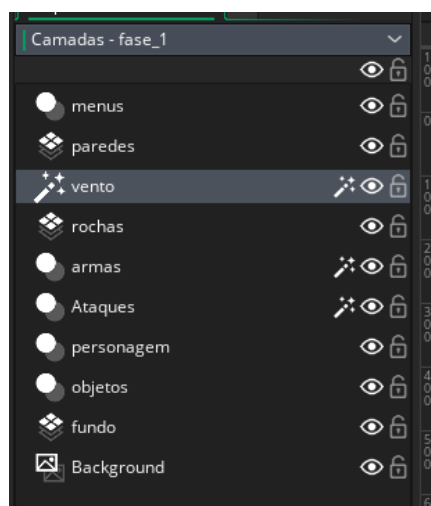
5. Cenas

Cenas ou salas no Game Maker Studio. As cenas são os ambientes onde o jogo ocorre e podem conter vários objetos, fundos e outros elementos

Crie cenas



É necessário criar todas essas camadas, sendo que a camada de efeitos (vento) é opcional, pois ela apenas adiciona um efeito ao jogo.



As camadas (armas, ataques) são utilizadas no código e não precisam conter nenhum conteúdo visual.

A camada "personagem" é destinada ao objeto do personagem principal e aos inimigos.

Na camada "objetos", estarão localizados objetos como o_chao (o chão), que possui colisão com o personagem, e também as chaves (objetivo) .

As camadas (fundo, rochas e paredes) são responsáveis pela organização dos elementos visuais do mapa. Ao posicioná-las acima ou abaixo umas das outras, é possível sobrepor objetos ou desenhos.

A camada "menus" deve conter apenas os objetos relacionados aos menus, como objetos de pausa e de morte.

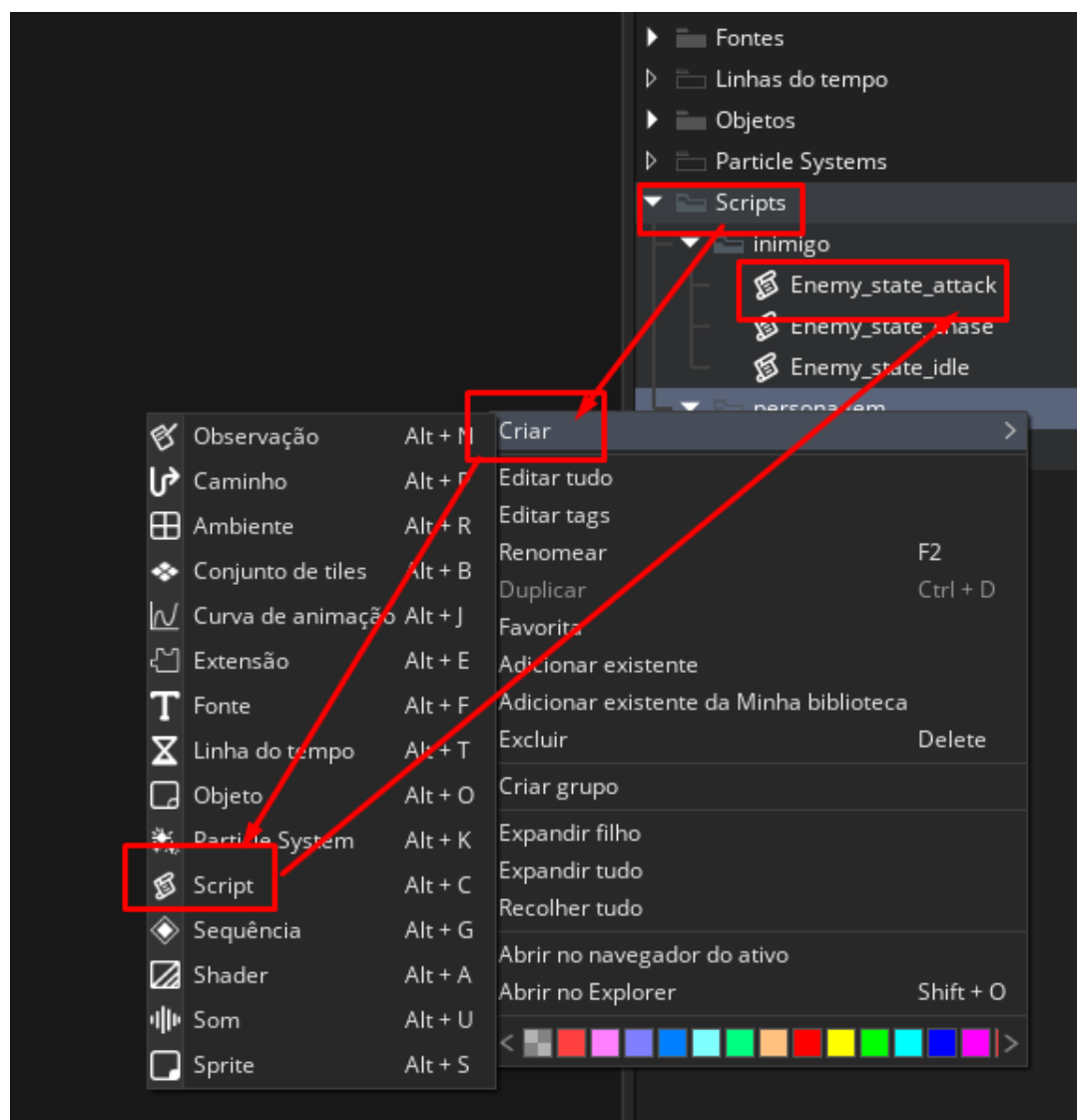
Crie uma fase separada e insira nela o objeto "o_menu". Esta fase será utilizada como o menu inicial do jogo.

6. Eventos

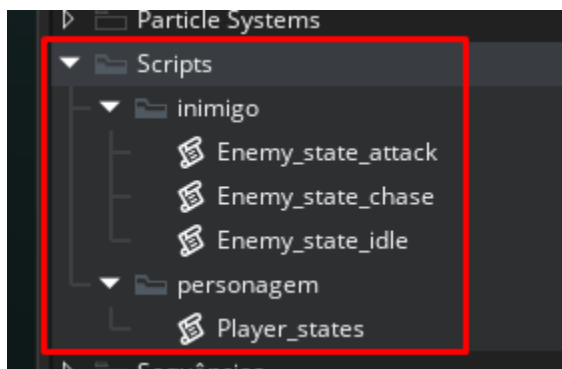
Por meio dos eventos, desenvolveremos as mecânicas do jogo. Os códigos apresentados neste documento são eventos que estão associados a objetos específicos. Ao escrever os códigos, certifique-se de selecionar o objeto correto para inserir o código e escolha o tipo de evento a ser criado com atenção.

7. Scripts e States

Alguns códigos são elaborados em scripts para organizar e evitar a repetição do mesmo código. Dentro desses scripts, criaremos os estados do personagem e inimigo, os quais podem ser invocados e utilizados por qualquer objeto.



Scripts que devem ser criados



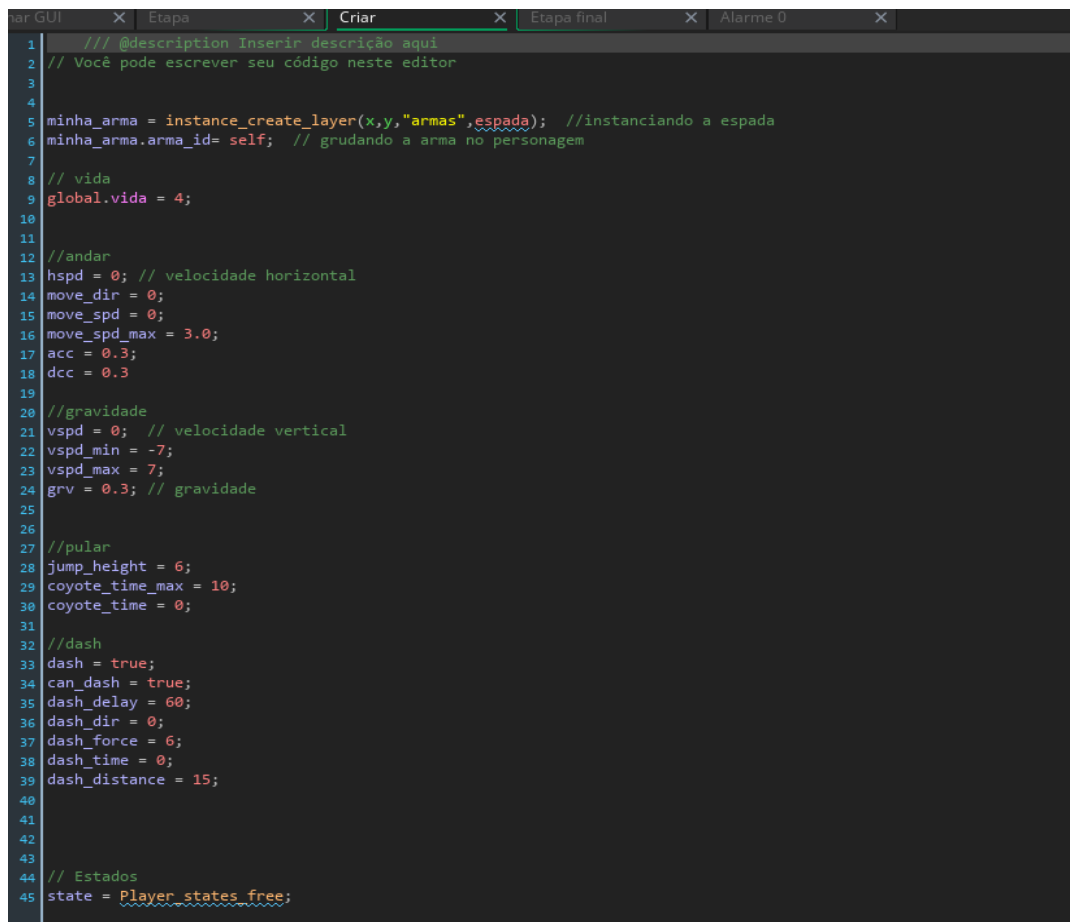
8. Personagem

Aqui é onde todo o código do personagem controlado pelo jogador está localizado. O código é dividido em dois lugares distintos: o objeto (personagem) e os estados (states). Ao dividir o código dessa maneira, conseguimos uma organização mais eficiente e facilitamos futuras manutenções e implementações.

9. Variáveis

Dentro do evento de criação (Create) do objeto personagem, estarão todas as variáveis necessárias para controlar as ações do personagem, juntamente com o estado (state) que será acessado, contendo as ações específicas do personagem.

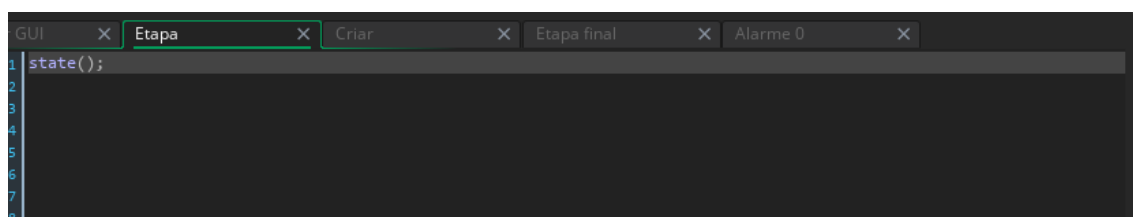
OBS: Crie uma variável chamada "chaves" e atribua o valor inicial de 0. Esta variável não está presente na imagem, mas é necessária para o funcionamento correto do código.



```

1  /// @description Inserir descrição aqui
2  // Você pode escrever seu código neste editor
3
4
5  minha_armas = instance_create_layer(x,y,"armas",espada); //instanciando a espada
6  minha_armas.armas_id= self; // grudando a arma no personagem
7
8  // vida
9  global.vida = 4;
10
11
12  //andar
13  hspd = 0; // velocidade horizontal
14  move_dir = 0;
15  move_spd = 0;
16  move_spd_max = 3.0;
17  acc = 0.3;
18  dcc = 0.3
19
20  //gravidade
21  vspd = 0; // velocidade vertical
22  vspd_min = -7;
23  vspd_max = 7;
24  grv = 0.3; // gravidade
25
26
27  //pular
28  jump_height = 6;
29  coyote_time_max = 10;
30  coyote_time = 0;
31
32  //dash
33  dash = true;
34  can_dash = true;
35  dash_delay = 60;
36  dash_dir = 0;
37  dash_force = 6;
38  dash_time = 0;
39  dash_distance = 15;
40
41
42
43
44  // Estados
45  state = Player_states_free;
  
```

Na parte destinada ao personagem, é onde residiriam as ações tradicionais do personagem. No entanto, como estamos empregando estados (states), faremos uso do método "state", o qual utilizará o estado selecionado nas variáveis.



```

1  state();
2
3
4
5
6
7
8
  
```

State 1

Todos os estados do personagem são centralizados no mesmo script, portanto é crucial separá-los com comentários. O estado "pausar" é responsável por interromper o movimento de todo o mapa e objetos quando o jogo é pausado. Abaixo, temos as seções para o movimento do jogador, animações, gravidade e ataque com a espada.

Durante o ataque com a espada, o código acessa diretamente o objeto "espada" usando o método "with". Utilizamos o método "point_direction()" para obter a direção do mouse e armazenamos essa informação na variável "arma_dir", indicando a direção em que a arma deve atacar. Para realizar um ataque, um dos requisitos é que "can_ataque" esteja habilitado, e quando atacamos, definimos "can_ataque" como falso para impedir ataques infinitos. Em seguida, chamamos o método "atacar()".

```

Player_states.gml
1 // aqui ficara todos estados do player
2
3 function Player_states_free(){
4
5     //PAUSAR
6     if (global.pause) or (global.morte) {
7         speed = 0;
8         image_speed = 0;
9         exit;
10    }else{
11        image_speed = 1;
12    }
13
14
15    // movimento do player
16
17
18    var key_left = keyboard_check(ord("A"));
19    var key_right = keyboard_check(ord("D"));
20    var key_jump = keyboard_check_pressed(ord("W"));
21
22    var move = key_right - key_left !=0;
23
24    if(move){
25        move_dir = point_direction(0,0,key_right - key_left,0);
26        move_spd = lerp(move_spd,move_spd_max,acc);
27
28    }else{
29        move_spd = lerp(move_spd,0,dcc);
30
31    }
32
33
34    hspd = lengthdir_x(move_spd,move_dir);
35
36
37    // animacao
38    if keyboard_check(vk_anykey){
39        sprite_index = S_Slime_Andando;
40        if (hspd !=0) image_xscale = sign(hspd); // checa a direção que o personagem esta andando, e direciona
41    }else{
42        sprite_index = S_Slime_Parado;
43    }
44
45
46
47
48    //Gravidade
49
50    vspd += grv;
51    vspd = clamp(vspd,vspd_min,vspd_max); // limita o valor mínimo e máximo da queda
52
53    // Atacar com a espada
54    with(minha_armas){
55        arma_dir = point_direction(x,y,mouse_x,mouse_y); //define a direção que a arma aponta, o mouse
56        if (mouse_check_button_released(mb_right) and ataque and can_ataque) {
57            can_ataque = false;
58            atacar();
59        }
60    }
61
62

```

State 2

A seguir, encontramos cada ação do personagem. Observe que a maioria dessas ações invoca um estado (state) no final, separando assim a lógica da função da execução da função. Todos os estados podem ser invocados em diferentes locais, proporcionando uma maior flexibilidade ao sistema. e organização

```

62 // atacar com o slime
63 if (mouse_check_button(mb_left)){
64     hspd = 0;
65     vspd = 0;
66     image_index = 0; // começa a animacao do primeiro sprite
67     if (mouse_x < x) image_xscale = -1; else image_xscale = 1; // direcionao o ataque para o lado do mouse
68     state = player_states_atk; // muda para o estado de ataque
69 }
70
71 //dash
72 if (keyboard_check_pressed(vk_space) and dash and can_dash){
73     can_dash = false;
74     alarm [0] = dash_delay;
75     hspd = 0;
76     vspd = 0;
77     dash_dir = point_direction(x,y,mouse_x,mouse_y);
78     state = player_states_dash;
79 }
80
81 //Ativar teleporte
82 if keyboard_check(ord("F")){
83     state = player_states_teleportar;
84 }
85
86 //pulo
87 var ground = place_meeting(x,y+1,o_chao);
88 if (ground){
89     coyote_time = coyote_time_max;
90 }else{
91     coyote_time --;
92 }
93
94 if (key_jump and coyote_time > 0){
95 }
96
97 // Dash
98 function player_states_dash(){
99     //PAUSAR
100     if (global.pause) or (global.morte) {
101         speed = 0;
102         image_speed = 0;
103         exit;
104     }else{
105         image_speed = 1;
106     }
107     vspd = lengthdir_y(dash_force,dash_dir);
108     hspd = lengthdir_x(dash_force,dash_dir);
109     dash_time++
110     if (dash_time >= dash_distance){
111         dash_time = 0;
112         vspd = 0;
113         hspd = 0;
114         state = Player_states_free;
115     }
116 }
117
118 }
119
120
121
122
123
124
125
126
127
128
129
130
131

```

State 3

Aqui temos o teleporte que pega a localização do objeto (o_ataque) e permiti que o objeto (personagem) vá para a localização x,y do objeto, e após isto volta para o (state) padrão do personagem.

O (state) abaixo cria uma caixa de colisão, para que os ataques do personagem sejam mais precisos.

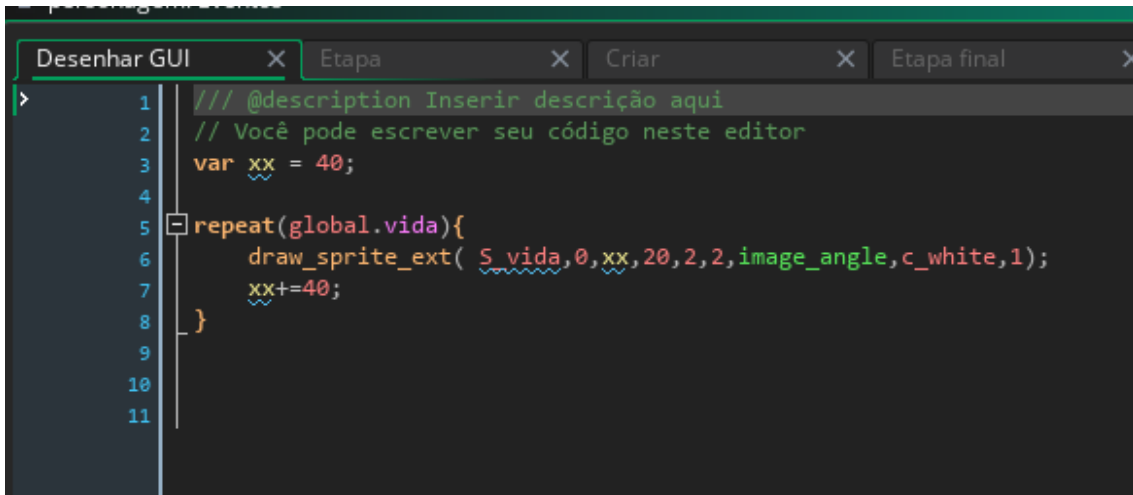
```

130
131
132 //teleporte
133 function player_states_teleportar() {
134     //PAUSAR
135     if (global.pause) or (global.morte) {
136         speed = 0;
137         image_speed = 0;
138         exit;
139     }else{
140         image_speed = 1;
141     }
142 }
143 if(instance_exists(o_ataque)){
144     var ex, ey;
145     ex = instance_nearest(x, y, o_ataque).x; //encontra o objeto solicitado mais próximo
146     ey = instance_nearest(x, y, o_ataque).y;
147
148     effect_create_below(ef_smoke, x, y, 1, c_gray); // efeito da fumaça ao teleportar
149     x = ex; // redefine a posição do personagem para o objeto ( o_ataque ) mais próximo
150     y = ey;
151     // destroi o objeto após o teleporte
152     state = Player_states_free;
153 }else{
154     state = Player_states_free;
155 }
156
157
158 }
159
160
161 //ataque
162 function player_states_atk(){
163     if(image_index >3){ // cria o hitbox apartir do segundo frame
164         if(!instance_exists(hitbox)){ // checa se existe a hitbox, se não, cria uma nova
165             instance_create_layer(x+(43 * image_xscale),y - 7,layer,hitbox); // criando a hitbox
166         }
167     }
168 }
169
170 sprite_index = S_Slime_Ataque;
171 if (image_index >= image_number -1){ // realiza a acao abaixo apenas enquanto esta rolando a animacao
172     if(instance_exists(hitbox)) instance_destroy(hitbox); // destroi a hitbox criada
173     state = Player_states_free; // saindo do estado de ataque
174 }
175 }
176
177

```


Vida do personagem

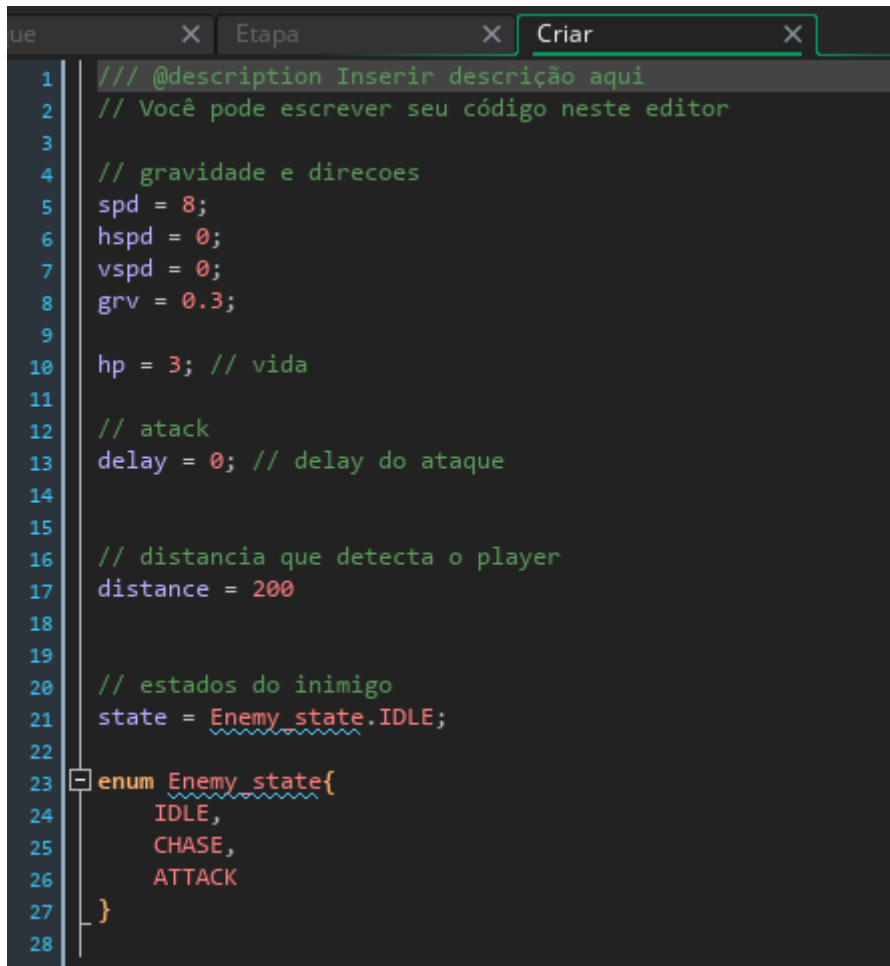
Quando criamos um evento "Desenhar GUI" no objeto (personagem) e usamos "REPEAT(global.vida)", estamos determinando que o código deve se repetir a mesma quantidade de vezes que o valor da variável global "vida". Esse código criará o sprite designado na posição especificada, representando assim a vida do personagem e ajustando-se conforme o valor da variável "vida" aumenta ou diminui.



```
1  /// @description Inserir descrição aqui
2  // Você pode escrever seu código neste editor
3  var xx = 40;
4
5  repeat(global.vida){
6      draw_sprite_ext( S_vida,0,xx,20,2,2,image_angle,c_white,1);
7      xx+=40;
8  }
9
10
11
```

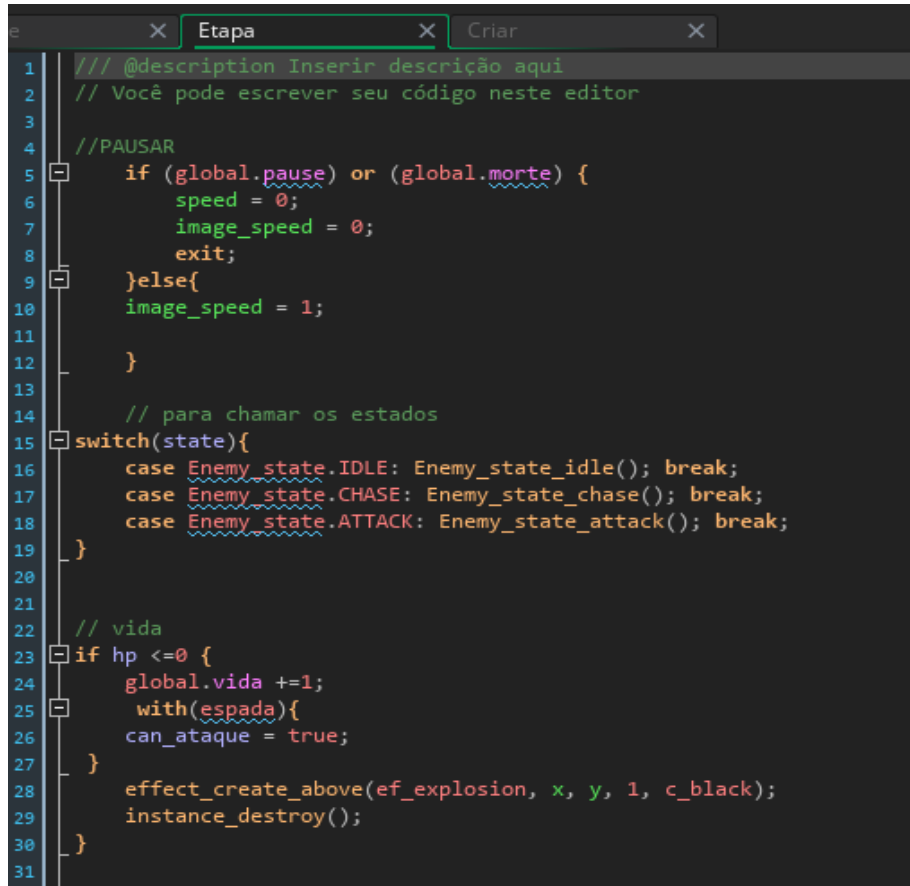
10. Inimigo

No evento "Criar" do objeto (inimigo), configure as variáveis do personagem e os estados que serão utilizados. Como serão empregados mais de um script para os estados, iremos criar um enum para facilitar a troca entre os scripts.



```
1  /// @description Inserir descrição aqui
2  // Você pode escrever seu código neste editor
3
4  // gravidade e direcoes
5  spd = 8;
6  hspd = 0;
7  vspd = 0;
8  grv = 0.3;
9
10 hp = 3; // vida
11
12 // attack
13 delay = 0; // delay do ataque
14
15
16 // distancia que detecta o player
17 distance = 200
18
19
20 // estados do inimigo
21 state = Enemy_state.IDLE;
22
23 enum Enemy_state{
24     IDLE,
25     CHASE,
26     ATTACK
27 }
28
```


Na etapa do inimigo, definiremos a pausa, um método para trocar entre os estados e o que ocorrerá se a vida dele chegar a zero. Nesse caso, o objeto (personagem) receberá 1 de vida e a espada terá "can_ataque" definido como verdadeiro, permitindo que o personagem utilize o ataque da espada novamente.



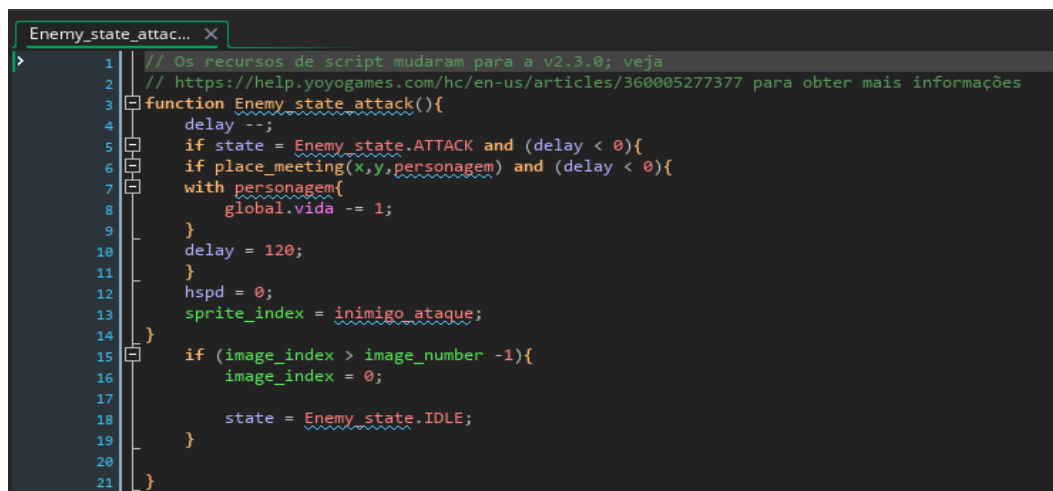
```

1  /// @description Inserir descrição aqui
2  // Você pode escrever seu código neste editor
3
4  //PAUSAR
5  if (global.pause) or (global.morte) {
6      speed = 0;
7      image_speed = 0;
8      exit;
9  }else{
10     image_speed = 1;
11
12 }
13
14 // para chamar os estados
15 switch(state){
16     case Enemy_state.IDLE: Enemy_state_idle(); break;
17     case Enemy_state.CHASE: Enemy_state_chase(); break;
18     case Enemy_state.ATTACK: Enemy_state_attack(); break;
19 }
20
21
22 // vida
23 if hp <=0 {
24     global.vida +=1;
25     with(espada){
26         can_ataque = true;
27     }
28     effect_create_above(ef_explosion, x, y, 1, c_black);
29     instance_destroy();
30 }
31

```

State de Ataque do Inimigo

O seguinte código determina que o inimigo realizará um ataque de acordo com o atraso especificado.



```

1  // Os recursos de script mudaram para a v2.3.0; veja
2  // https://help.yoyogames.com/hc/en-us/articles/360005277377 para obter mais informações
3  function Enemy_state_attack(){
4      delay --;
5      if state = Enemy_state.ATTACK and (delay < 0){
6          if place_meeting(x,y,personagem) and (delay < 0){
7              with personagem{
8                  global.vida -= 1;
9              }
10             delay = 120;
11         }
12         hspd = 0;
13         sprite_index = inimigo_ataque;
14     }
15     if (image_index > image_number -1){
16         image_index = 0;
17
18         state = Enemy_state.IDLE;
19     }
20
21 }

```

State de Perseguição do Inimigo

Neste estado, encontram-se a gravidade, movimentação e colisão do inimigo, todas elas foram copiadas do objeto (personagem), exceto pelas teclas que ativavam o movimento. Além disso, abaixo alteramos os estados do inimigo de acordo com a distância do personagem.

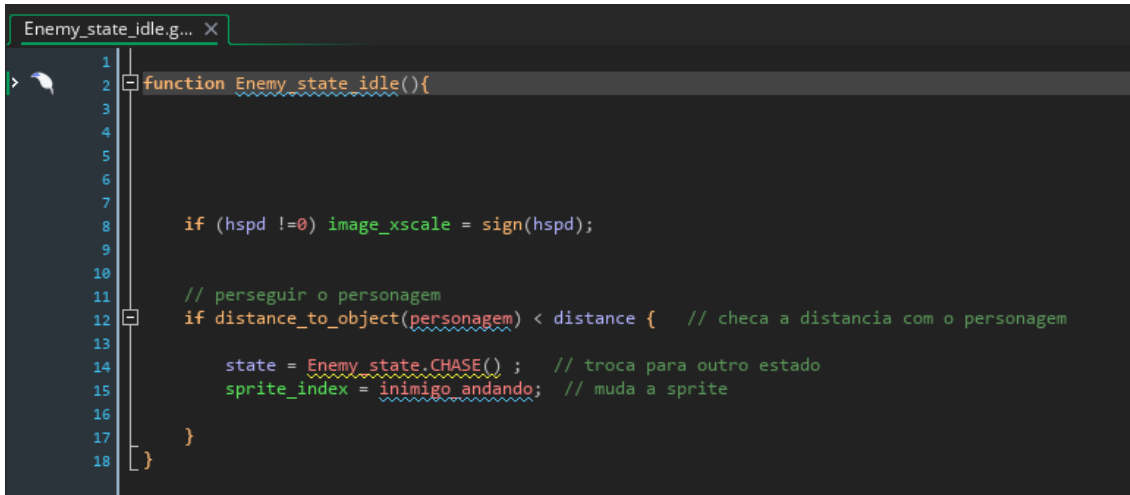
```

1
2 function Enemy_state_chase(){
3
4     // Gravidade
5     vspd = vspd+ grv;
6
7     // colisao horizontal
8     if(place_meeting(x+hspd,y,o_chao)){
9         while(!place_meeting(x+sign(hspd),y,o_chao)){
10             x = x + sign (hspd);
11         }
12         hspd = 0;
13     }
14     x = x + hspd;
15
16
17     // colisao vertical
18     if(place_meeting(x,y+vspd,o_chao)){
19         while(!place_meeting(x,y+sign(vspd),o_chao)){
20             y = y + sign (vspd);
21         }
22         vspd = 0;
23     }
24     y = y + vspd;
25
26
27
28     if (hspd !=0) image_xscale = sign(hspd);
29
30     // seguir
31     vir = sign(personagem.x - x); // seguir o personagem
32     hspd = vir * 2; // velocidade em que anda
33
34     // parar de seguir
35     if distance_to_object(personagem) > distance{
36         state = Enemy_state.IDLE();
37     }
38
39     // ataque
40     if distance_to_object(personagem) < 60{
41         state = Enemy_state.ATTACK();
42     }
43
44
45
46 }

```

State de Inatividade do Inimigo

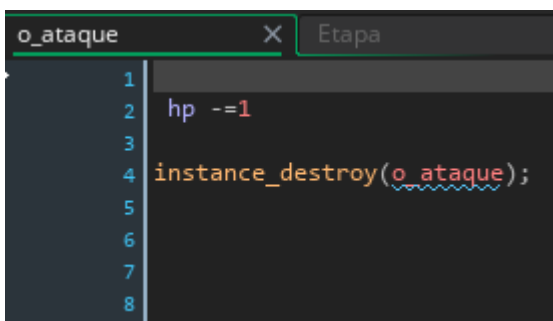
Ao usar este estado, o inimigo permanecerá parado, mas se estiver a uma certa distância do personagem, outros estados serão ativados.



```
1  
2 function Enemy_state_idle(){  
3  
4  
5  
6  
7  
8     if (hspd !=0) image_xscale = sign(hspd);  
9  
10  
11     // perseguir o personagem  
12     if distance_to_object(personagem) < distance { // checa a distancia com o personagem  
13  
14         state = Enemy_state_CHASE(); // troca para outro estado  
15         sprite_index = inimigo_andando; // muda a sprite  
16  
17     }  
18 }
```

Colisão do Inimigo com a Espada

Quando o inimigo colide com a espada, este código é ativado. Ele reduz um ponto de vida do inimigo e interrompe o ataque, exigindo que o objeto (personagem) reative "can_ataque = true" novamente. Isso pode resultar na eliminação do inimigo ou em interações com outro objeto que possam ter o mesmo efeito.

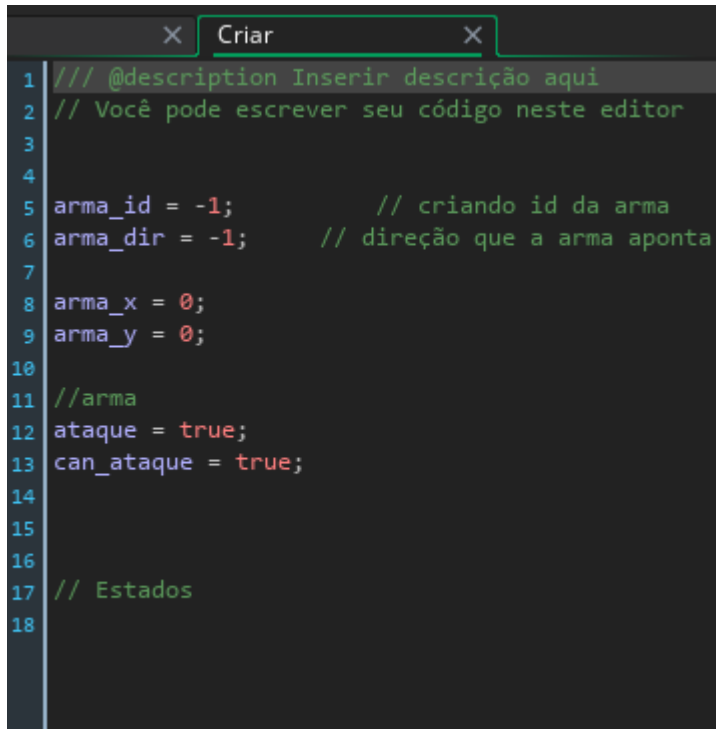


```
1  
2 hp -=1  
3  
4 instance_destroy(o_ataque);  
5  
6  
7  
8
```

11. Espada

Criação da Espada

No objeto (espada), vamos definir as variáveis da arma, como ataque, can_ataque, além das variáveis de direção e localização da arma, que estarão ao redor do personagem.



```

1  /// @description Inserir descrição aqui
2  // Você pode escrever seu código neste editor
3
4
5  arma_id = -1;           // criando id da arma
6  arma_dir = -1;         // direção que a arma aponta
7
8  arma_x = 0;
9  arma_y = 0;
10
11 //arma
12 ataque = true;
13 can_ataque = true;
14
15
16
17 // Estados
18

```

Espada etapa

Temos o evento de pausa, que pausa os sprites da espada quando o jogo é pausado ou o personagem morre. Abaixo, encontramos as funções da arma. Ao associarmos o "arma_id" ao objeto (personagem), podemos utilizá-lo aqui para definir o posicionamento da arma ao redor do personagem. O "arma_dir" retorna a posição do mouse em relação ao objeto (personagem), permitindo que definamos as variáveis "arma_x" e "arma_y" para indicar onde a arma deve apontar e a distância que deve ficar do jogador.

Na função "atacar()", criamos o projétil da espada. Antes disso, devemos criar uma nova camada na room com o nome selecionado para os projéteis. Neste caso, utilizaremos "Ataques". Definimos a sprite a ser utilizada e, usando as variáveis criadas anteriormente, determinamos o ângulo da sprite, a direção que ela deve seguir e, por último, a velocidade do projétil, que neste caso é 4.

```

1  /// @description Inserir descrição aqui
2  // Você pode escrever seu código neste editor
3  //PAUSAR
4  if (global.pause) or (global.morte) {
5      speed = 0;
6      image_speed = 0;
7      exit;
8  }else{
9      image_speed = 1;
10
11  }
12
13
14  if(instance_exists(arma_id)){ //checando se a arma existe
15
16      x = arma_id.x;           // definindo onde a arma ira se posicionar, e seguir o personagem
17      y = arma_id.y;
18
19      image_angle = arma_dir;   // para a arma apontar para o mouse
20
21      arma_x = x + lengthdir_x(30,arma_dir); // redefine a posição da arma para perto do personagem
22      arma_y = y + lengthdir_y(30,arma_dir);
23
24      function atacar(){
25
26
27
28      var ataque = instance_create_layer(arma_x ,arma_y,"Ataques",o_ataque);
29      ataque.sprite_index = S_Espada_Parada;
30      ataque.image_angle = arma_dir + 45;
31      ataque.direction = arma_dir;
32      ataque.speed = 4;
33
34
35      }
36
37  }
38

```

Desenhar GUI da Espada

Vamos adicionar um evento "Desenhar GUI" ao objeto (espada) e definir como o sprite da espada deve ser exibido.

```

1  /// @description Inserir descrição aqui
2  // Você pode escrever seu código neste editor
3
4  draw_sprite_ext(sprite_index,image_index,arma_x,arma_y,1,1,arma_dir + 45,c_white,1);
5  // gira a animação da arma para seguir o mouse
6

```

12. Ataque espada (projétil)

Este código está no objeto lançado pela espada, ou seja, o projétil. Aqui temos uma pausa e um condicional que verifica se a velocidade da espada é diferente de zero. Se for, ele cria um efeito ao redor do projétil, produzindo um rastro à medida que o projétil se move.

```

Etapa x o_chao x personagem x
1  /// @description Inserir descrição aqui
2  // Você pode escrever seu código neste editor
3
4  //PAUSAR
5  if (global.pause) or (global.morte) {
6      speed = 0;
7      image_speed = 0;
8      exit;
9  }else{
10     image_speed = 1;
11
12 }
13
14 if speed!= 0{
15     effect_create_above(ef_spark, x,y, 0.01, c_lime) // efeito da espada
16 }
17
18
19
20

```

Colisão com o Personagem

Quando o projétil colide com o personagem, ele altera "can_ataque" para verdadeiro novamente e destrói o projétil, permitindo que o ataque da espada seja possível novamente.

```

x o_chao x personagem x
1  /// @description Inserir descrição aqui
2  // Você pode escrever seu código neste editor
3
4  with(espada){
5      can_ataque = true;
6  }
7
8  instance_destroy();
9

```

Colisão com o Chão

Quando a espada colide com o chão, isso permite um novo ataque e destrói o projétil.

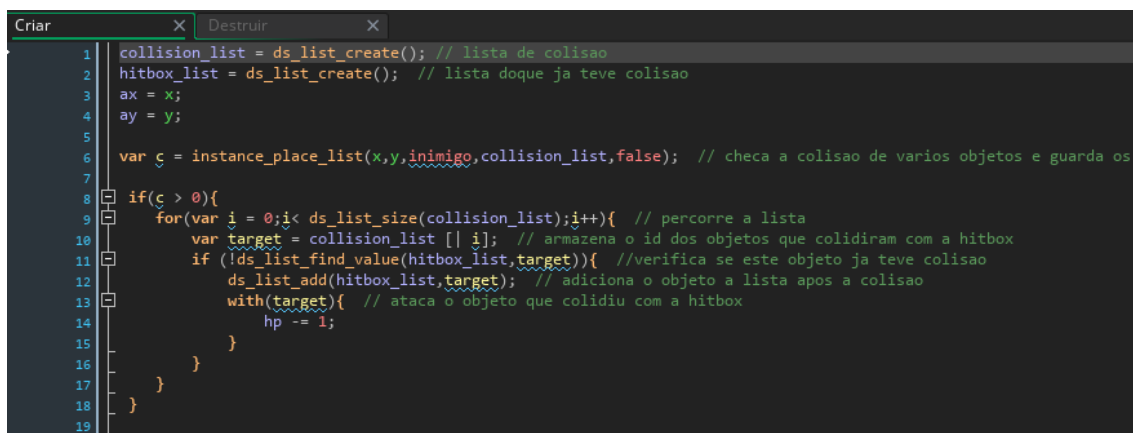
```

a x o_chao x per
1  /// @description Inserir descrição
2  // Você pode escrever seu código n
3
4  with(espada){
5      can_ataque = true;
6  }
7
8  instance_destroy();
9
10
11
12

```

13. Hit box

Crie um objeto com o tamanho e formato do ataque do personagem, que será um ataque diferente da espada, um ataque do próprio slime. Começamos criando duas listas: uma lista para armazenar os objetos que já colidiram e outra lista para os que estão em colisão. Armazene em uma variável os objetos que estão em colisão e percorra esta lista. Guarde o ID dos objetos em colisão e verifique se o ID já teve colisão antes. Se não teve, então aplique dano (hp -= 1) e adicione este objeto à lista de objetos que já colidiram. Este código é utilizado para causar dano em vários inimigos ao mesmo tempo, de forma precisa.



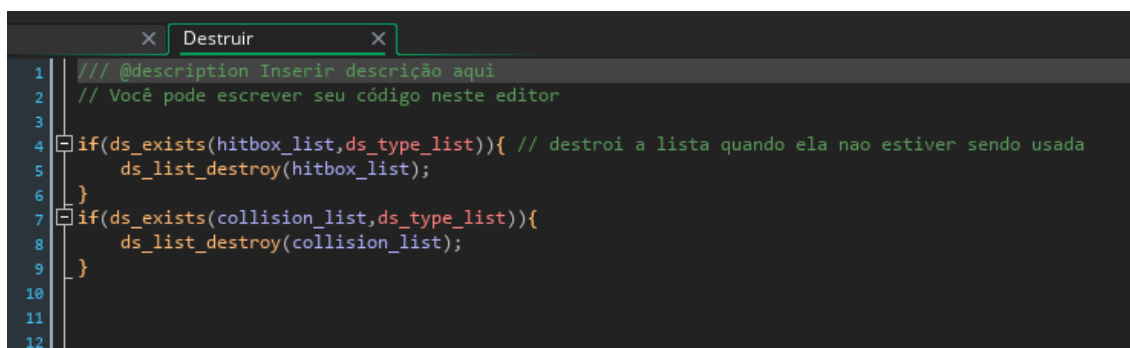
```

1 collision_list = ds_list_create(); // lista de colisao
2 hitbox_list = ds_list_create(); // lista doque ja teve colisao
3 ax = x;
4 ay = y;
5
6 var c = instance_place_list(x,y,inimigo,collision_list,false); // checa a colisao de varios objetos e guarda os
7
8 if(c > 0){
9     for(var i = 0; i < ds_list_size(collision_list); i++){ // percorre a lista
10         var target = collision_list[i]; // armazena o id dos objetos que colidiram com a hitbox
11         if(!ds_list_find_value(hitbox_list,target)){ //verifica se este objeto ja teve colisao
12             ds_list_add(hitbox_list,target); // adiciona o objeto a lista apos a colisao
13             with(target){ // ataca o objeto que colidiu com a hitbox
14                 hp -= 1;
15             }
16         }
17     }
18 }
19

```

Destruir Hitbox

Ao utilizar listas no Game Maker, é importante apagá-las posteriormente para evitar bugs e não sobrecarregar o sistema. Este código irá apagar as listas selecionadas quando elas não estiverem sendo usadas.



```

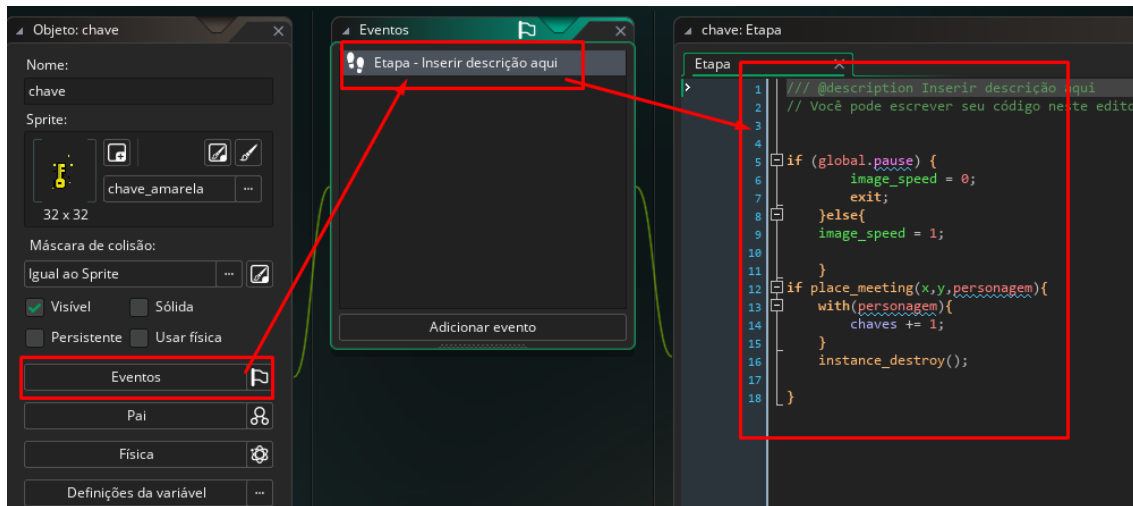
1 /// @description Inserir descrição aqui
2 // Você pode escrever seu código neste editor
3
4 if(ds_exists(hitbox_list,ds_type_list)){ // destroi a lista quando ela nao estiver sendo usada
5     ds_list_destroy(hitbox_list);
6 }
7 if(ds_exists(collision_list,ds_type_list)){
8     ds_list_destroy(collision_list);
9 }
10
11
12

```

14. Chave

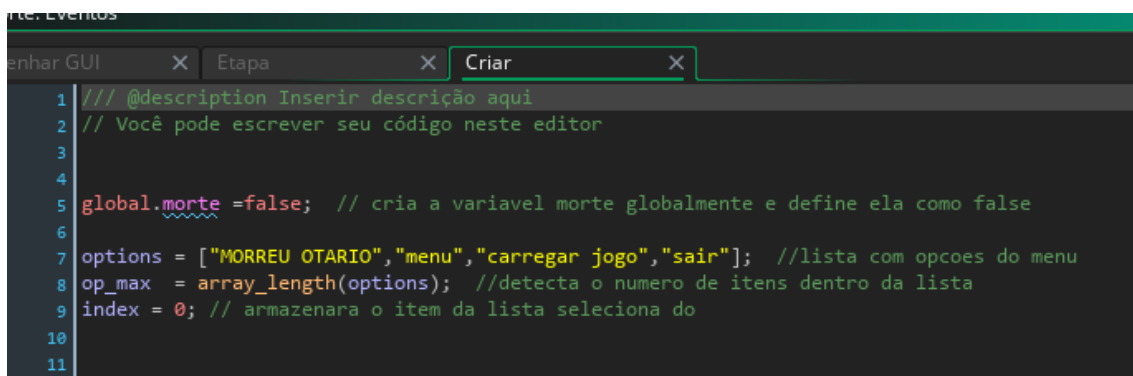
O objeto "chave" será utilizado para avançar para a próxima fase. Serão espalhadas três chaves pelo mapa, e ao serem encontradas, darão acesso a uma porta que leva ao próximo nível.

O código abaixo verifica se houve colisão com o personagem. Se sim, ele incrementa a variável "chaves" no personagem em +1 e destrói o objeto "chave".



15. Menu morte

Crie o objeto (menu_morte) e, dentro dele, no evento "Criar", defina a variável global "morte" que será usada para abrir a tela de morte. Em seguida, crie a lista de opções que devem aparecer na tela de morte e as variáveis que auxiliarão na manipulação da lista.



Desenhar Interface Gráfica do Menu de Morte

No objeto (menu_morte), adicione um evento "Desenhar GUI". Utilizaremos esse evento para criar a interface da tela de morte, utilizando métodos prontos do Game Maker para estilizá-la. Em seguida, criaremos uma maneira de exibir as opções da lista utilizando um laço "for()". Depois,

configuraremos a colisão com cada item da lista e definiremos ações ao clicar em cada um. Por fim, adicionaremos um sprite para ser exibido na tela de morte..

```

2  var gui_w = display_get_gui_width(); //tamanho da interface de pause
3  var gui_h = display_get_gui_height();
4
5  var x1 = gui_w / 2; // pega a medida do centro da tela, e armazena
6  var y1 = gui_h / 2;
7
8  var margin = 50; // distancia entre cada opcao
9
10 var m_x = device_mouse_x_to_gui(0); // encontra a posicao do mouse
11 var m_y = device_mouse_y_to_gui(0);
12 draw_set_halign(fa_center); // alinha o texto apartir do meio da palavra
13 draw_set_valign(fa_center);
14
15
16
17 if(global.morte = true) {
18
19     draw_set_color(c_black);
20     draw_rectangle(0,0,gui_w,gui_h,false);
21     draw_set_alpha(1);
22     draw_set_font(pause);
23
24     for(var i = 0; i < op_max; i++){ //percorre a lista e define elas no i
25         var y2 = y1 + (margin * i); // da a distancia entre cada opcao
26         var string_w = string_width(options[i]); // armazena o tamanho de cada palavra
27         var string_h = string_height(options[i]);
28
29         if(point_in_rectangle(m_x,m_y,x1 - string_w / 2, y2 - string_h / 2, x1 + string_w / 2, y2 + string_h / 2)){ //cria colisao em cada palavra da lista
30             draw_set_color(c_white); // muda a letra para verde ao colidir
31             index = i; // define o index para o item selecionado
32
33             if mouse_check_button_pressed(mb_left){
34                 if(index == 0){
35
36                 }
37                 }else if(index == 1){
38                     room_goto(menu) // vai para o menu
39                 }else if(index == 2){
40                     room_restart(); // volta para o inicio do nivel
41                     global.morte = false;
42                 }else if(index == 3){
43                     game_end(); // finaliza o jogo
44                 }
45             }
46
47         }else{
48             draw_set_color(c_red); // muda a letra para branco ao parar de colidir
49         }
50
51         draw_text(x1,y2,options[i]); // mostra os itens percorridos no i
52     }
53
54     draw_sprite_ext(5_Slime Morte,image_index,680,270,5,5,image_angle,c_white,1);
55 }
56
57
58
59
60

```

Etapa do Menu de Morte

Na etapa, vamos utilizar um condicional "if()" para identificar quando a vida do personagem chegar a zero. Quando isso ocorrer, a variável "morte" será definida como verdadeira, o que ativará o código feito no evento "Desenhar GUI" para mostrar a tela de morte.

te: Eventos

Desenhar GUI

```

1  /// @description Inserir descrição aqui
2  // Você pode escrever seu código nest
3
4
5
6  if global.vida <= 0 {
7
8      global.morte= true;
9  }else{
10     global.morte = false;
11 }
12
13
14
15

```

16. Menu inicial

O código utilizado aqui é o mesmo do objeto (menu_morte), com a diferença sendo apenas as opções da lista.

```

Criar  X  Etapa  X  *Desenhar GUI  X
1  /// @description Inserir descrição aqui
2  // Você pode escrever seu código neste editor
3
4
5  global.menu = true;
6
7  options = ["novo jogo", "sair"]; //lista com opcoes do menu
8  op_max = array_length(options); //detecta o numero de itens dentro da lista
9  index = 0; // armazenara o item da lista selecionado
10
11
12
13

```

Replicaremos o código do objeto (menu_morte), com algumas alterações nas opções de estilização e nas ações das opções da lista.

```

X  Etapa  X  *Desenhar GUI  X
1  // o pause
2
3  var gui_w = display_get_gui_width(); //tamanho da interface de pause
4  var gui_h = display_get_gui_height();
5
6  var x1 = gui_w / 2; // pega a medida do centro da tela, e armazena
7  var y1 = gui_h / 2;
8
9  var margin = 50; // distancia entre cada opcao
10
11  var m_x = device_mouse_x_to_gui(0); // encontra a posicao do mouse
12  var m_y = device_mouse_y_to_gui(0);
13  draw_set_halign(fa_center); // alinha o texto apartir do meio da palavra
14  draw_set_valign(va_center);
15
16
17
18  if(global.menu){
19      // desenhando a interface de pause
20      effect_create_above(ef_flare, m_x, m_y, 1, c_white)
21
22      for(var i = 0; i < op_max; i++){ //percorre a lista e define elas no i
23          var y2 = y1 + (margin * i); // da a distancia entre cada opcao
24          var string_w = string_width(options[i]); // armazena o tamanho de cada palavra
25          var string_h = string_height(options[i]);
26
27          if(point_in_rectangle(m_x, m_y, x1 - string_w / 2, y2 - string_h / 2, x1 + string_w / 2, y2 + string_h / 2)){ //cria colisao em cada palavra da lista
28              draw_set_color(c_black); // muda a letra para preto ao colidir
29              index = i; // define o index para o item selecionado
30
31              if mouse_check_button_pressed(mb_left){
32                  if(index == 0){
33                      room_goto(fase_1); // vai para fase 1
34                  }else if(index == 1){
35                      game_end(); // finaliza o jogo
36                  }
37              }
38
39              }else{
40                  draw_set_color(c_white); // muda a letra para branco ao parar de colidir
41              }
42
43              draw_text(x1, y2, options[i]); // mostra os itens percorridos no i
44          }
45
46          draw_set_alpha(.2)
47          draw_set_color(c_black);
48          draw_rectangle(0, 0, gui_w, gui_h, false);
49          draw_set_alpha(1);
50          draw_set_color(c_white);
51          draw_set_font(font_menu);
52
53
54
55  }
56
57

```

17. Pausa menu

Vamos repetir os códigos dos menus anteriores, fazendo ajustes nas opções da lista.

```

1 /// @description Inserir descrição aqui
2 // Você pode escrever seu código neste editor
3
4
5 global.pause = false;
6
7 options = ["menu", "carregar jogo", "sair"]; //lista com opcoes do menu
8 op_max = array_length(options); //detecta o numero de itens dentro da lista
9 index = 0; // armazenara o item da lista selecionado
10
11
12
13

```

Desenhar GUI da Pausa

Utilizaremos os mesmos códigos dos menus anteriores, personalizando o design e as opções conforme necessário.

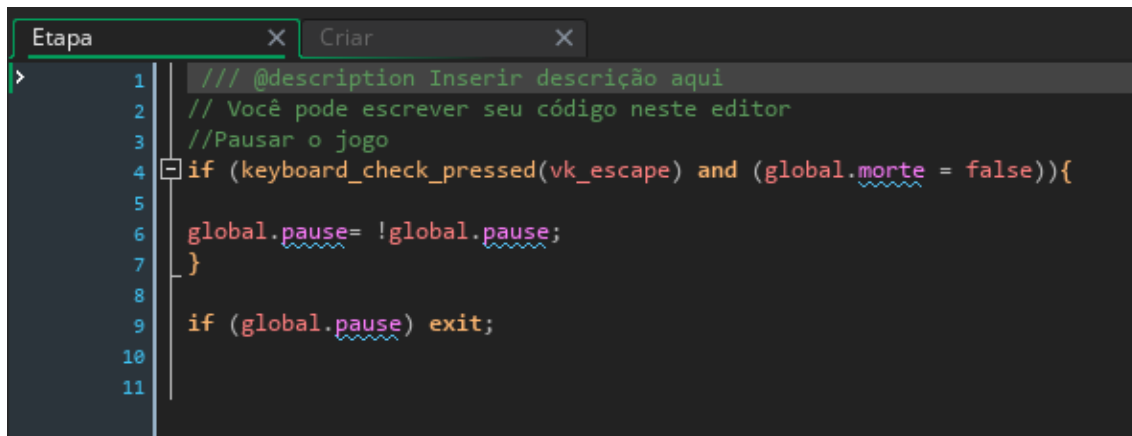
```

1 // o pause
2
3 var gui_w = display_get_gui_width(); //tamanho da interface de pause
4 var gui_h = display_get_gui_height();
5
6 var x1 = gui_w / 2; // pega a medida do centro da tela, e armazena
7 var y1 = gui_h / 2;
8
9 var margin = 50; // distancia entre cada opcao
10
11 var m_x = device_mouse_x_to_gui(0); // encontra a posicao do mouse
12 var m_y = device_mouse_y_to_gui(0);
13 draw_set_halign(fa_center); // alinha o texto apartir do meio da palavra
14 draw_set_valign(fa_center);
15
16
17
18 if(global.pause){
19     // desenhando a interface de pause
20
21     for(var i = 0; i < op_max; i++){ //percorre a lista e define elas no i
22         var y2 = y1 + (margin * i); // da a distancia entre cada opcao
23         var string_w = string_width(options[i]); // armazena o tamanho de cada palavra
24         var string_h = string_height(options[i]);
25
26         if(point_in_rectangle(m_x, m_y, x1 - string_w / 2, y2 - string_h / 2, x1 + string_w / 2, y2 + string_h / 2)){ //cria colisao em cada palavra da lista
27             draw_set_color(c_green); // muda a letra para verde ao colidir
28             index = i; // define o index para o item selecionado
29
30             if mouse_check_button_pressed(mb_left){
31                 if(index == 0){
32                     room_goto(menu) // vai para o menu
33                 }else if(index == 1){
34                     room_restart(); // volta para o inicio do nivel
35                 }else if(index == 2){
36                     game_end(); // finaliza o jogo
37                 }
38             }
39
40             }else{
41                 draw_set_color(c_white); // muda a letra para branco ao parar de colidir
42             }
43
44             draw_text(x1, y2, options[i]); // mostra os itens percorridos no i
45         }
46
47         draw_set_alpha(.2);
48         draw_set_color(c_black);
49         draw_rectangle(0, 0, gui_w, gui_h, false);
50         draw_set_alpha(1);
51         draw_set_color(c_white);
52         draw_set_font(pause);
53
54
55
56 }

```

Etapa de Pausa

Vamos usar o botão (ESC) do teclado e verificar se não estamos na tela de morte. Se as condições forem verdadeiras, o menu de pausa será aberto e ao pressionarmos (ESC) novamente, o menu será fechado.



```
Etapa X Criar X
> 1  /// @description Inserir descrição aqui
2  // Você pode escrever seu código neste editor
3  //Pausar o jogo
4  if (keyboard_check_pressed(vk_escape) and (global.morte = false)){
5
6      global.pause= !global.pause;
7  }
8
9  if (global.pause) exit;
10
11
```