

GML Language

Introdutório

Fonte: Ponto e Vírgula

07 de Outubro de 2022

GameMaker Language (GML) é a linguagem de programação aplicada a jogos, da engine GameMaker Studio 2. Relato aqui minha experiência pessoal de aprendizado da linguagem utilizando, em primeiro momento, os vídeos introdutórios do canal “Ponto e Vírgula”, presente no youtube.

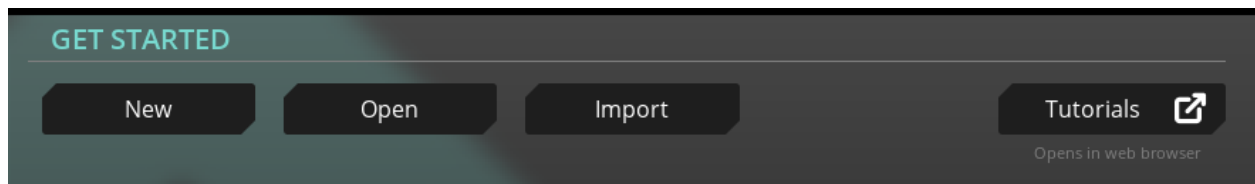
Iniciei às 12 horas e 20 minutos e irei utilizar o método pomodoro para me auxiliar. Inicio adicionando ao projeto os arquivos que serão utilizados nas video-aulas.

Aula #0

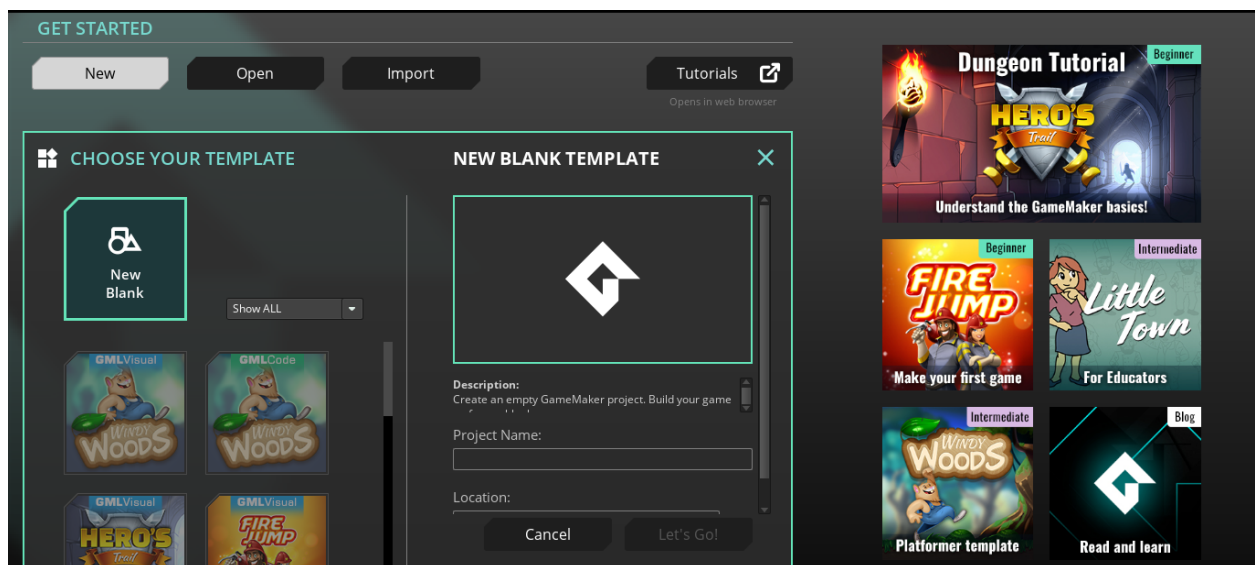
- Introdução à engine GameMaker Studio 2
 - O GameMaker é mais utilizado para a produção de jogos 2D (Duas Dimensões) .

Criando o primeiro projeto:

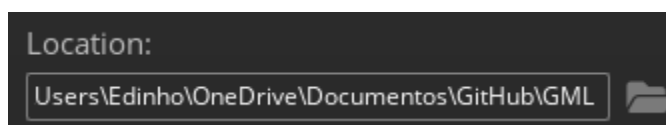
Para criar seu primeiro projeto, basta ao entrar no gamemaker clicar na opção “new”:



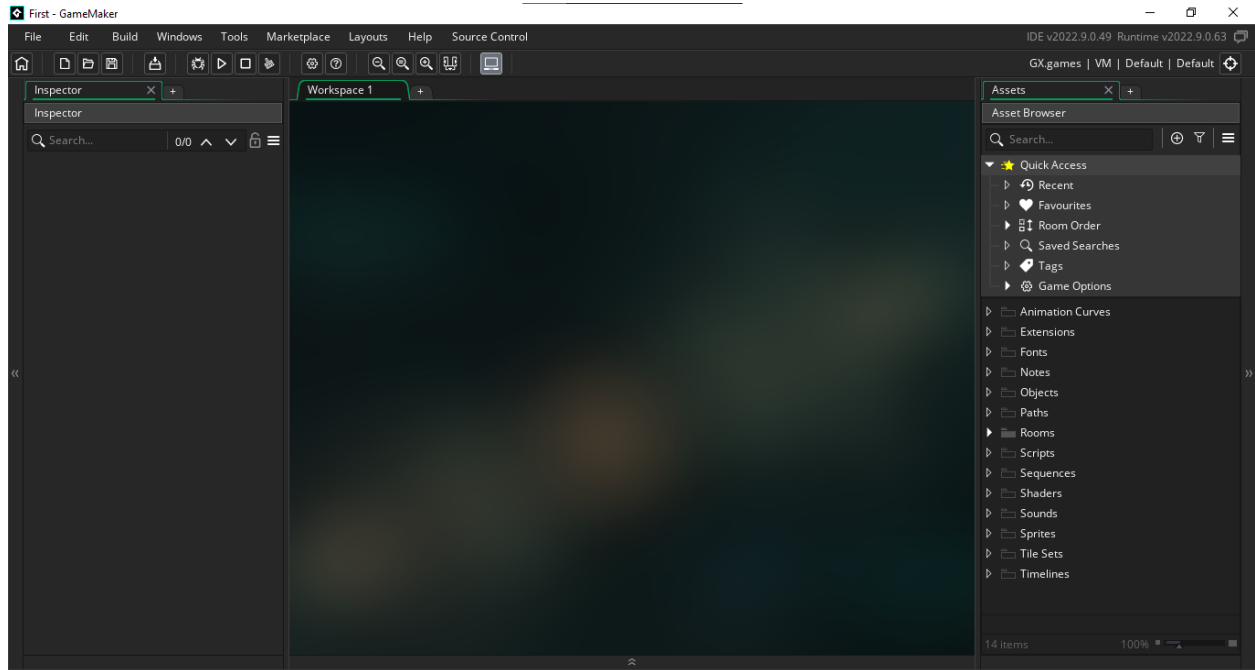
Você pode criar esse projeto por meio de um template já existente, ou começar do zero:



Ao adicionar um nome e possivelmente uma descrição, escolha um local para salvar o arquivo de seu projeto, no meu caso utilizarei a pasta do repositório do GitHub:

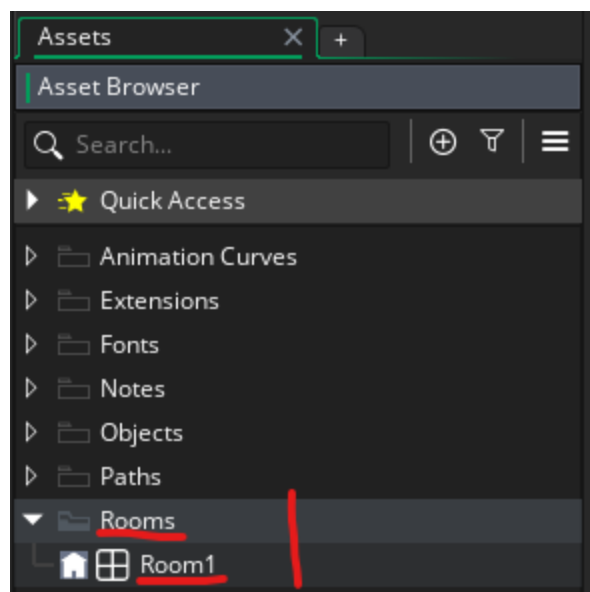


Agora basta clicar em “Let’s Go” e seu projeto será criado:

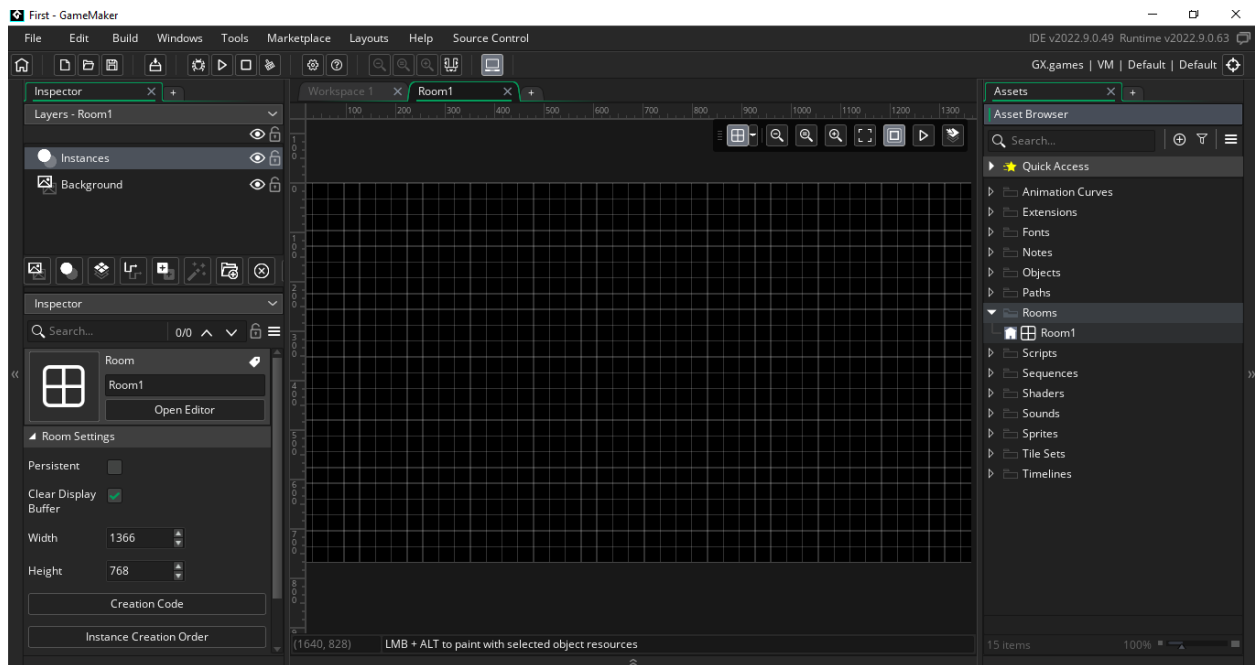


Explicando conceitos básicos:

As rooms será o espaço do jogo, onde terão os mapas, menus e outras partes visíveis e não visíveis ao usuário.

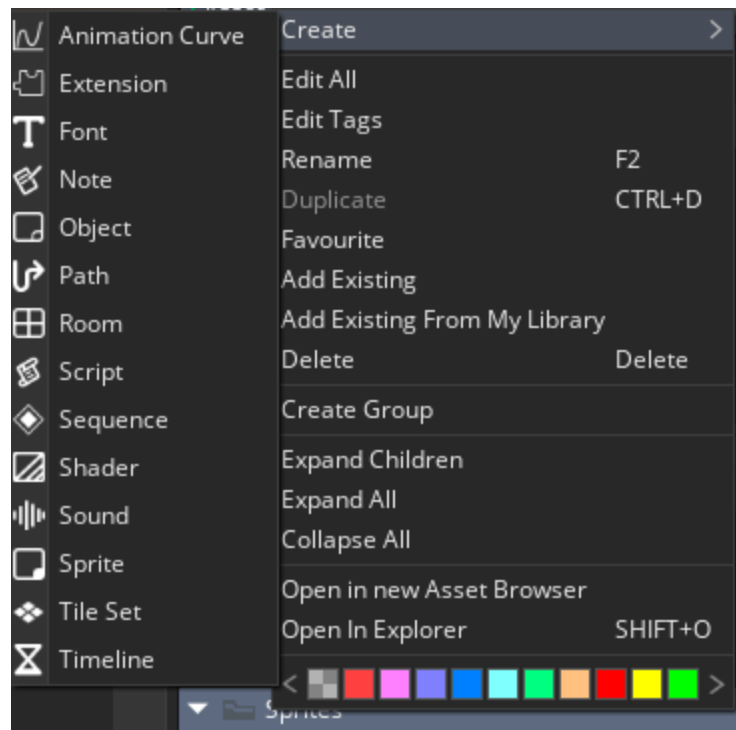


Essa é a visão que temos quando clicamos na Room1:

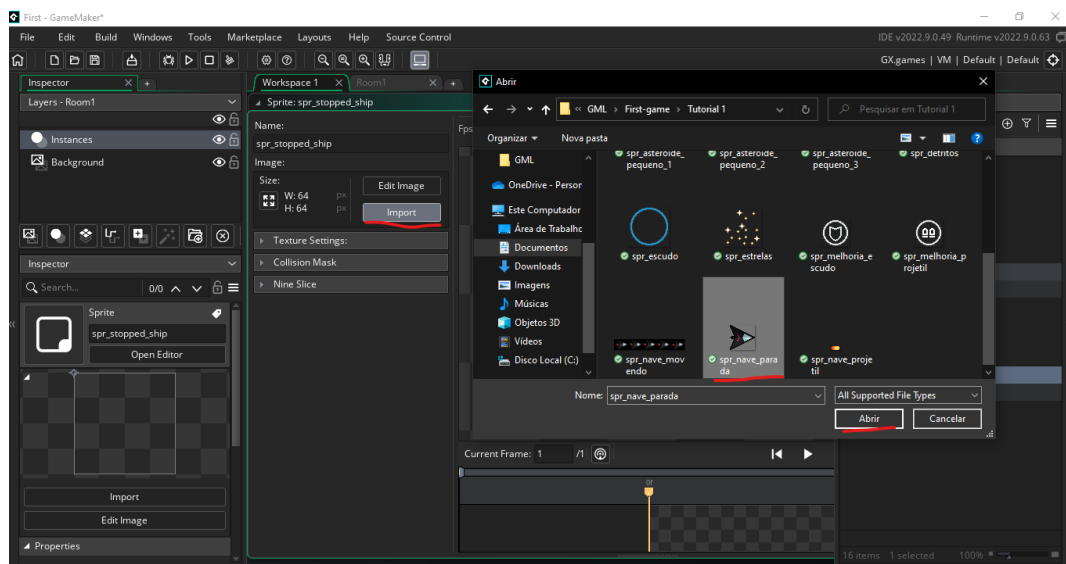


Aula #1

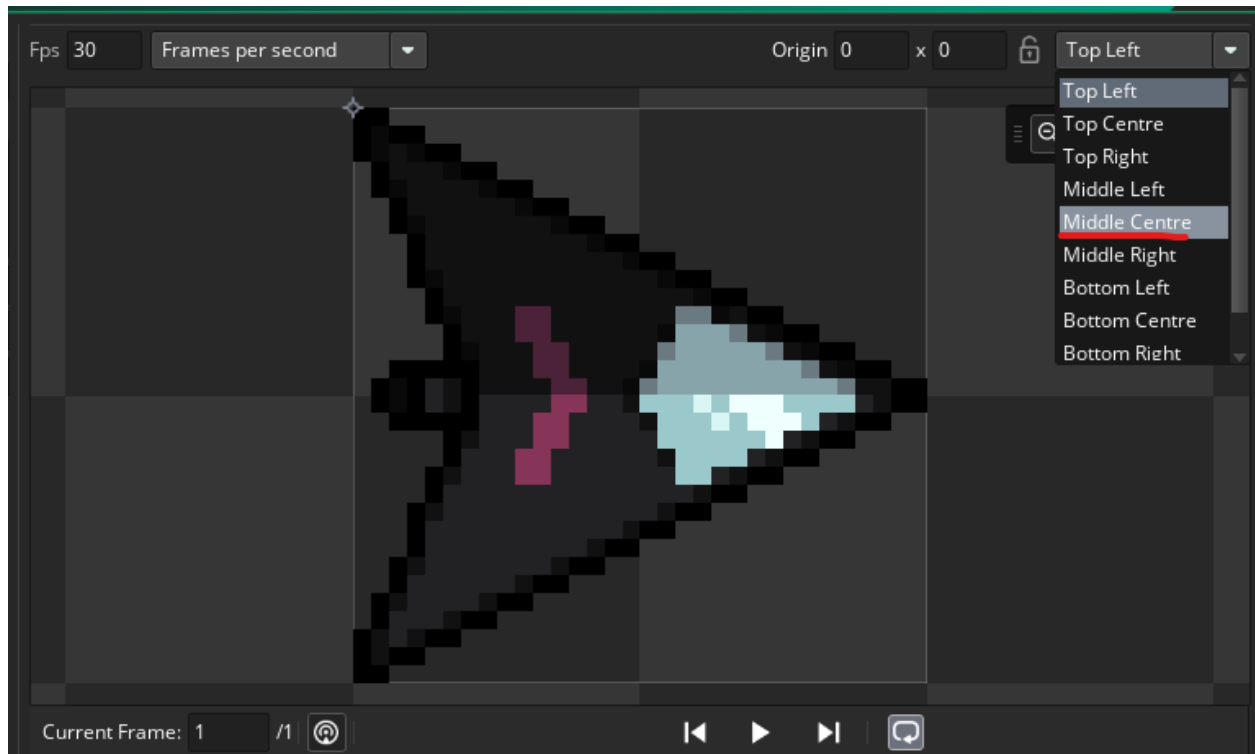
No conteúdo da aula 1 iniciamos adicionando alguns arquivos da nossa pasta ao jogo, para isso criamos um sprite, que é um recurso visual do jogo, muitas vezes atribuído a um objeto:



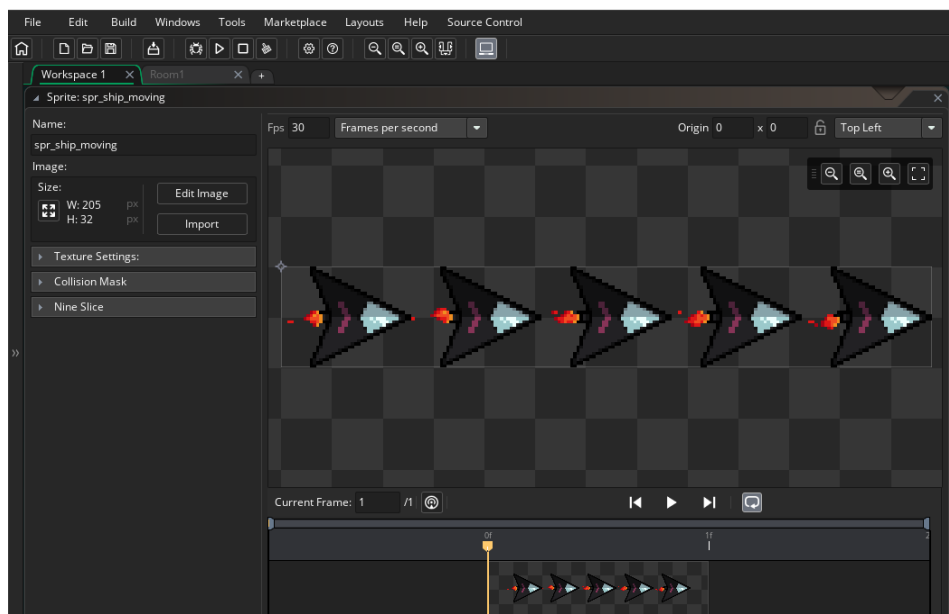
Comumente na criação de um sprite utilizamos um prefixo para classificá-lo, esse seria “spr_”, no nosso caso usamos spr_stopped_ship e adicionamos a nossa imagem para a nave parada:



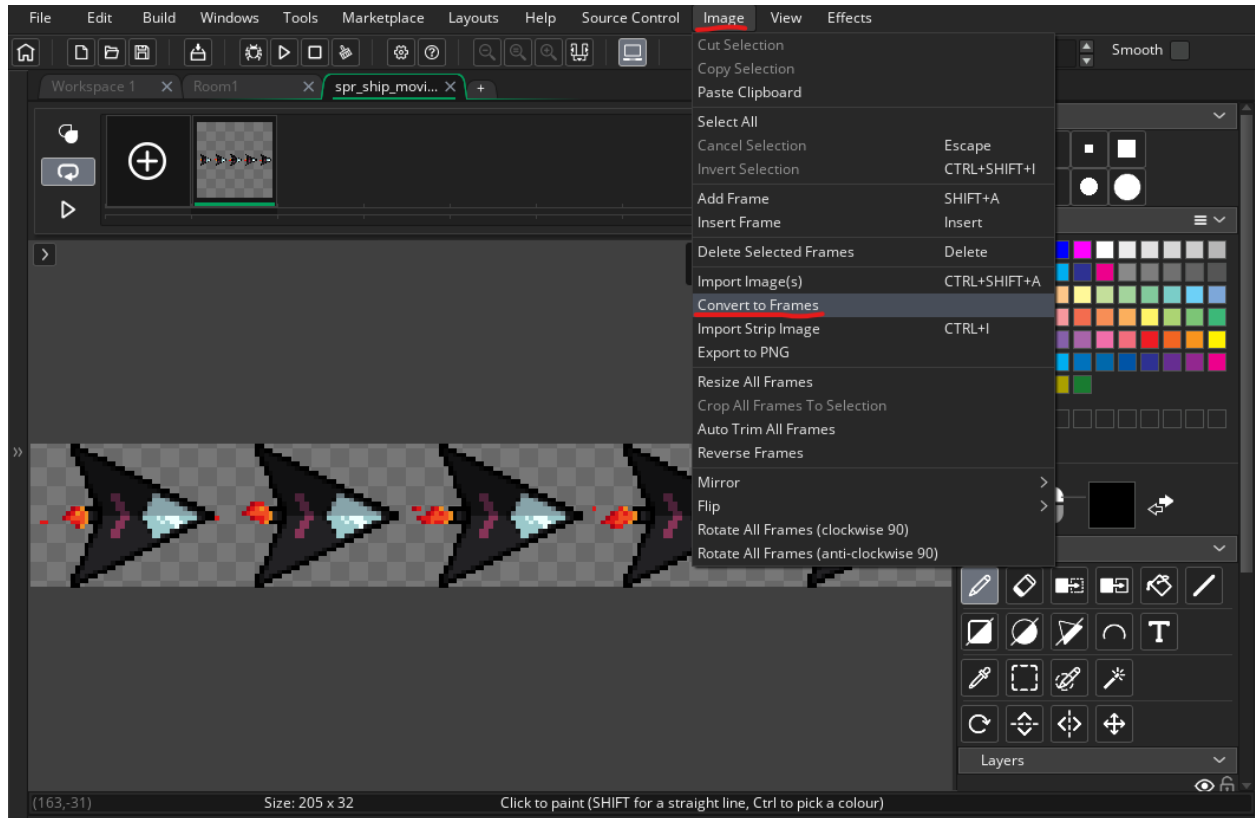
É importante também modificar o ponto do eixo de rotação do sprite para uma rotação coerente ao nosso jogo:



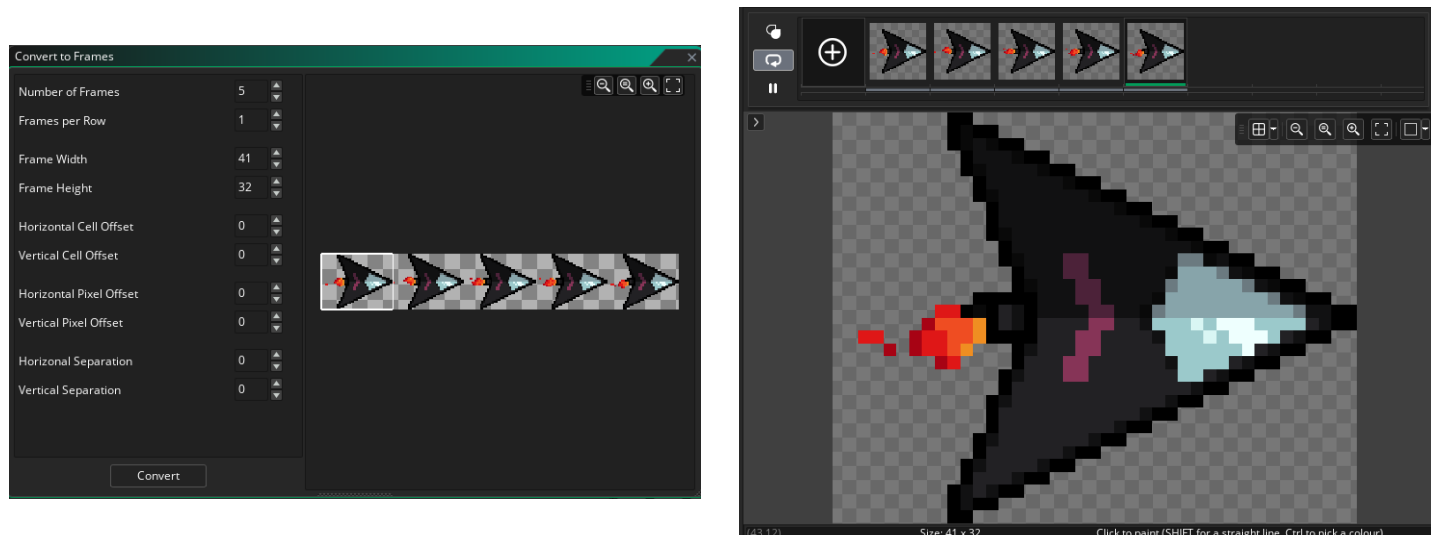
Após isso criamos o sprite da nave se movendo, porém há um problema, o que nós temos é um spritesheet da nave movendo, ou seja, toda a animação em uma única imagem:



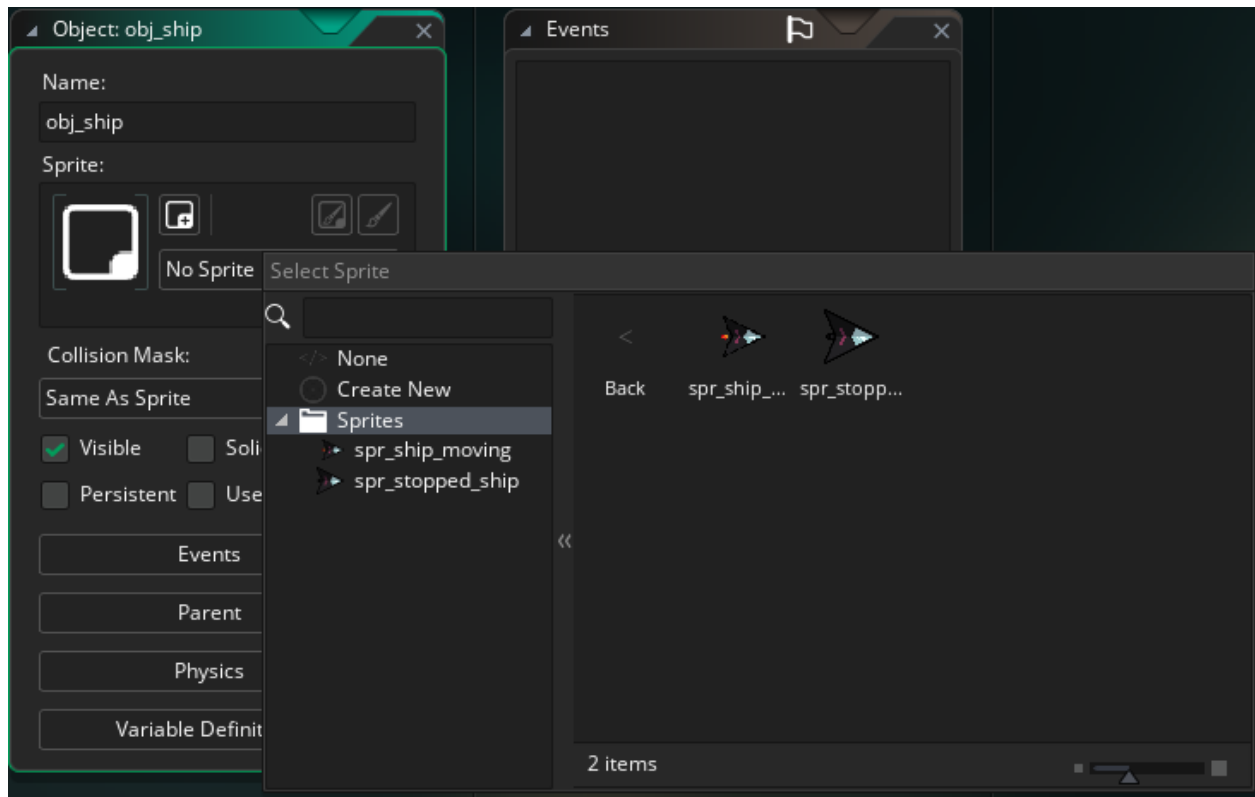
Para resolver isso, basta clicar em Edit Image > Image > Convert to Frames:



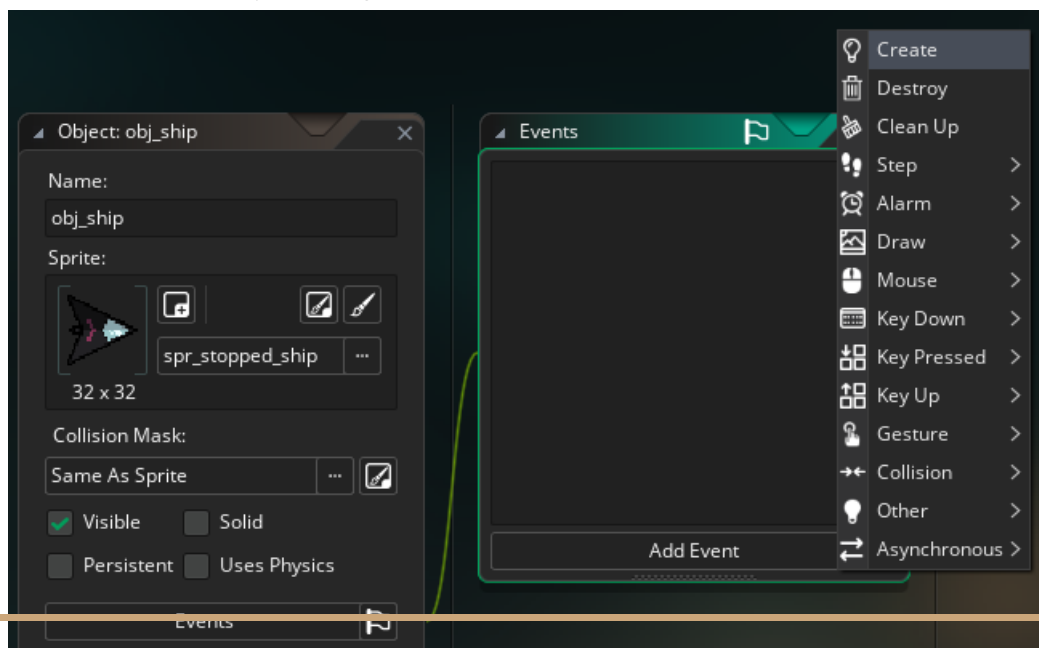
Após este passo, basta ajustar:



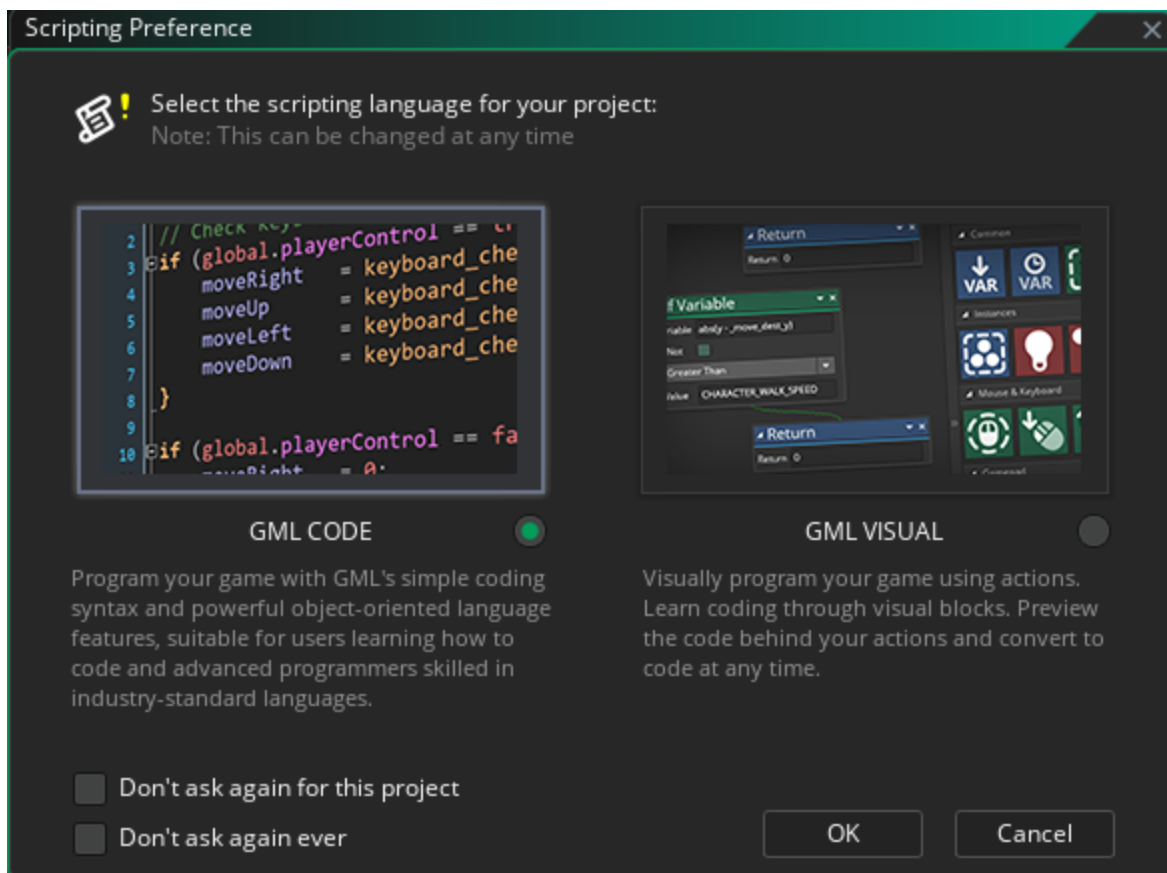
Com os sprites criados, criamos agora os objetos nos quais esses sprites serão atribuídos:



Com isso já podemos começar a programar, adicionando eventos a esse objeto:



Iremos focar nossa atenção em dois tipos de eventos por enquanto, são eles create e step. Começamos criando um step que basicamente executa seus comando a cada frame (Se o jogo tem 60 FPS, será executado 60 vezes por segundo). Quando criamos teremos duas opções de código, “GML Code” e “GML Visual”, usaremos o code:



Nesse evento step iremos começar a digitar os primeiros códigos para fazer a movimentação da nave:

```
1 /// @description Insert description here
2 if keyboard_check()
3 // if é um código condicional
4 // keyboard_check é um código que checa se a tecla está pressionada
```

Após isso podemos criar o bloco de código que representa cada tecla das setas do teclado, com o vk, que vamos usar para controlar a nave:

```
*Step
1  /// @description Insert description here
2  if keyboard_check(vk_up){ // if é um código condicional
3
4  }else keyboard_check(vk_down){ // keyboard_check é um código que checa se a tecla vk está pressionada
5
6  }
7  if keyboard_check(vk_left){
8
9  }else keyboard_check(vk_right){ // O código else também faz parte do código if condicional
10
11 }
```

O GameMaker sabe o que é uma velocidade então podemos utilizar uma variável speed sem preocupações:

```
1  /// @description Insert description here
2  if keyboard_check(vk_up){ // if é um código condicional
3      speed = 4;
4  }else if keyboard_check(vk_down){ // keyboard_check é um código que checa se a tecla vk está pressionada
5      speed = -4;
6  } else{ // O código else também faz parte do código if condicional
7      speed = 0;
8  } // speed = 0; para a nave parar de se mover caso nenhuma das duas teclas esteja sendo pressionada
9  if keyboard_check(vk_left){
10
11  }else if keyboard_check(vk_right){
12
13  }
```

Com este código já podemos nos mover para frente e para trás, precisamos agora adicionar a rotação da nave:

```
1  /// @description Insert description here
2  if keyboard_check(vk_up){ // if é um código condicional
3      speed = 4;
4  }else if keyboard_check(vk_down){ // keyboard_check é um código que checa se a tecla vk está pressionada
5      speed = -4;
6  } else{ // O código else também faz parte do código if condicional
7      speed = 0;
8  } // speed = 0; para a nave parar de se mover caso nenhuma das duas teclas esteja sendo pressionada
9  if keyboard_check(vk_left){
10      direction += 4; // O GameMaker reconhece o termo direction (Quanto maior a direção rotaciona à esquerda)
11  }else if keyboard_check(vk_right){
12      direction -= 4;
13  }
```

Esse código nos permite andar em todas as direções, porém o sprite permanece parado e não sabemos para qual direção está apontando, para resolver adicionamos:

```
14 | image_angle = direction // image_angle modifica o angulo do sprite
```

O nosso código de movimentação completo seria:

```
1 | /// @description Insert description here
2 | if keyboard_check(vk_up){ // if é um código condicional
3 |     sprite_index = spr_ship_moving; // sprite_index seleciona o sprite que o objeto estará durante a execução
4 |     speed = 4;
5 | }else if keyboard_check(vk_down){ // keyboard_check é um código que checa se a tecla vk está pressionada
6 |     sprite_index = spr_ship_moving;
7 |     speed = -4;
8 | } else{ // O código else também faz parte do código if condicional
9 |     sprite_index = spr_stopped_ship;
10 |    speed = 0;
11 | } // speed = 0; para a nave parar de se mover caso nenhuma das duas teclas esteja sendo pressionada
12 | if keyboard_check(vk_left){
13 |     direction += 4; // O GameMaker reconhece o termo direction (Quanto maior a direção rotaciona à esquerda)
14 | }else if keyboard_check(vk_right){
15 |     direction -= 4;
16 | }
17 | image_angle = direction // image_angle modifica o angulo do sprite
```

Para finalizar queremos apenas que a nossa nave ultrapasse por uma borda do mapa e volte no lado contrário, por sorte o gamemaker tem uma função para isso:

```
19 | move_wrap(true, true, 0); // move_wrap(hor, vert, margin);
```

08 de Outubro de 2022

Aula #2

Assim como já tínhamos feito, vamos criar um sprite e um objeto para o projétil.

Instância X Objeto:

Caso exista um objeto que se repete dentro do jogo, ele é diferenciado pelo seu ID (Cada objeto repetido terá um ID diferente) , ou seja, a instância é o objeto interligado ao ID.

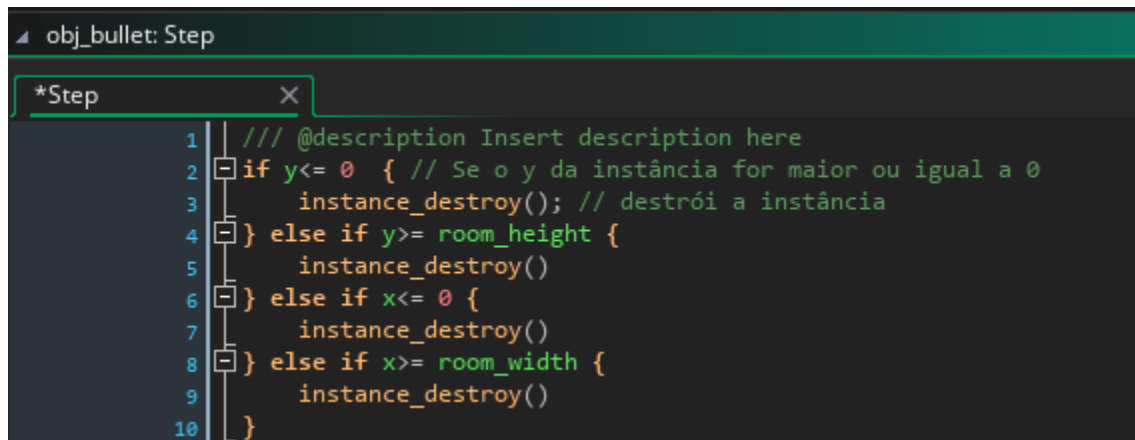
Voltando ao código:

```
21 if keyboard_check_pressed(vk_escape){ // Checa se a tecla foi pressionada
22     instance_create_layer(x, y, "Instances", obj_bullet);
23 } // Cria a instância de algum objeto, selecionando o ponto x e y
24 // Instances entre aspas indica a layer onde será criada a instância
```

Com este código a instância do projétil já está sendo criada, porém ele ainda não tem uma velocidade.

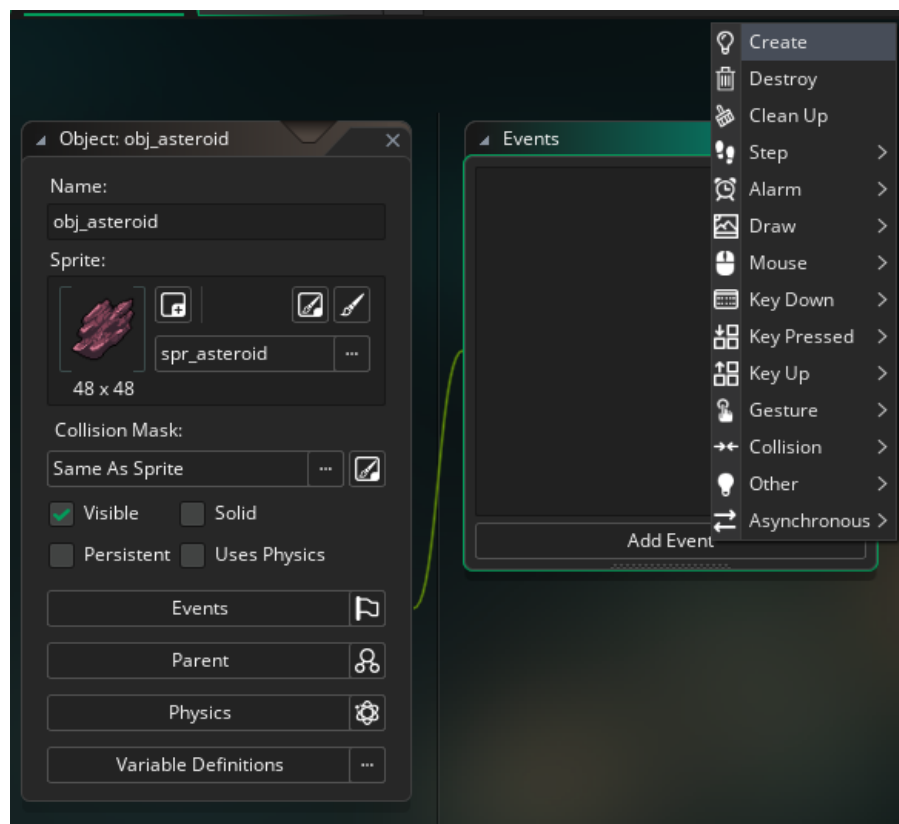
```
21 if keyboard_check_pressed(vk_space){ // Checa se a tecla foi pressionada
22     var inst = instance_create_layer(x, y, "Instances", obj_bullet);
23     // Cria a instância de algum objeto, selecionando o ponto x e y
24     // Instances entre aspas indica a layer onde será criada a instância
25     // var cria a variável local inst que recebe o valor da instância criada
26     inst.speed = 7;
27     inst.direction = direction; // igual a direção da var inst à direção da nave
28     inst.image_angle = direction; // rotaciona o sprite do projétil
29 }
```

A partir daqui o nosso tiro já está funcional, porém há um pequeno problema. Nesse caso, ao atirar a instância do projétil vai ficar andando infinitamente, consumindo uma memória muito grande, para resolver isso, basta fazer com que o objeto seja excluído assim que ultrapassar a borda do mapa:

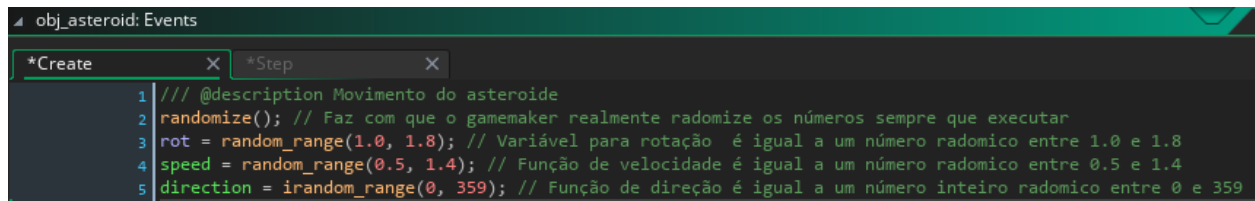


```
obj_bullet: Step
*Step
1  /// @description Insert description here
2  if y <= 0 { // Se o y da instância for maior ou igual a 0
3      instance_destroy(); // destrói a instância
4  } else if y >= room_height {
5      instance_destroy()
6  } else if x <= 0 {
7      instance_destroy()
8  } else if x >= room_width {
9      instance_destroy()
10 }
```

Finalizando isto, temos tiros funcionais, porém não temos em que ou quem fazê-lo, para isso adicionamos o asteroide ao projeto:

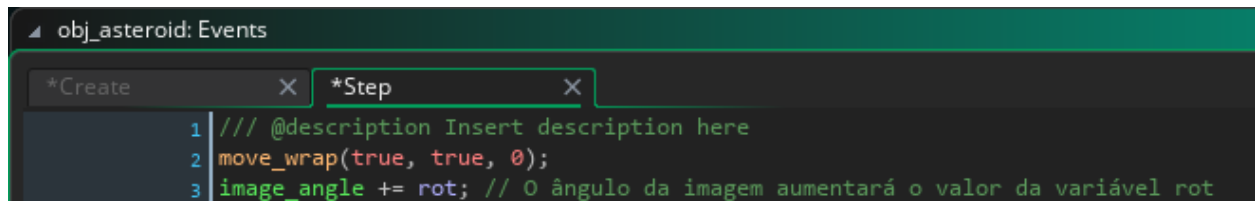


No objeto do asteroide criamos o evento criar e adicionaremos o bloco de código:



```
obj_asteroid: Events
*Create
1 /// @description Movimento do asteroide
2 randomize(); // Faz com que o gamemaker realmente randomize os números sempre que executar
3 rot = random_range(1.0, 1.8); // Variável para rotação é igual a um número randomico entre 1.0 e 1.8
4 speed = random_range(0.5, 1.4); // Função de velocidade é igual a um número randomico entre 0.5 e 1.4
5 direction = irandom_range(0, 359); // Função de direção é igual a um número inteiro randomico entre 0 e 359
```

Depois criamos um step para os asteroides rotacionarem:



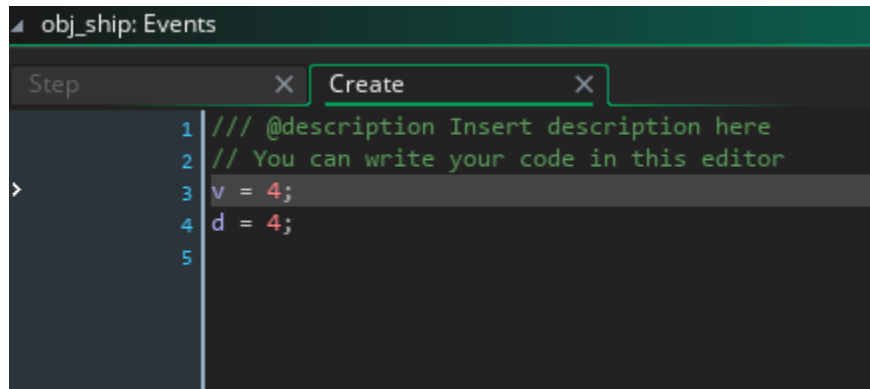
```
obj_asteroid: Events
*Create
*Step
1 /// @description Insert description here
2 move_wrap(true, true, 0);
3 image_angle += rot; // O ângulo da imagem aumentará o valor da variável rot
```

14 de Outubro de 2022

Aula #3

Melhorando a movimentação:

- Adicionar o evento criar e dentro dele colocar as variáveis v e d (velocidade e direção).



```
1 /// @description Insert description here
2 // You can write your code in this editor
3 v = 4;
4 d = 4;
5
```

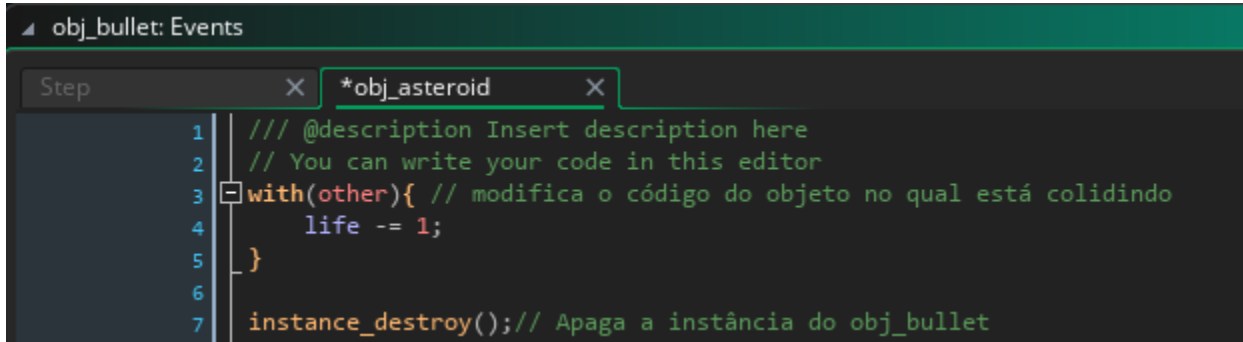
```
2 if keyboard_check(vk_up){ // if é um código condicional
3     sprite_index = spr_ship_moving; // sprite_index seleciona o sprite que o objeto estará durante a execução
4     speed = v;
5 }else if keyboard_check(vk_down){ // keyboard_check é um código que checa se a tecla vk está pressionada
6     sprite_index = spr_ship_moving;
7     speed = -v;
8 } else{ // O código else também faz parte do código if condicional
9     sprite_index = spr_stopped_ship;
10    speed = lerp(speed, 0, 0.1);
11 } // Speed diminui 0.1% a cada frame (desaceleração)
```

Melhorando a rotação:

```
13 if keyboard_check(vk_left){
14     d = 4;
15 }else if keyboard_check(vk_right){
16     d = -4;
17 }else {
18     d = lerp(d, 0, 0.09); // Direção vai desacelerando
19 }
20
21 direction += d; // Modifica a direção da rotação
```

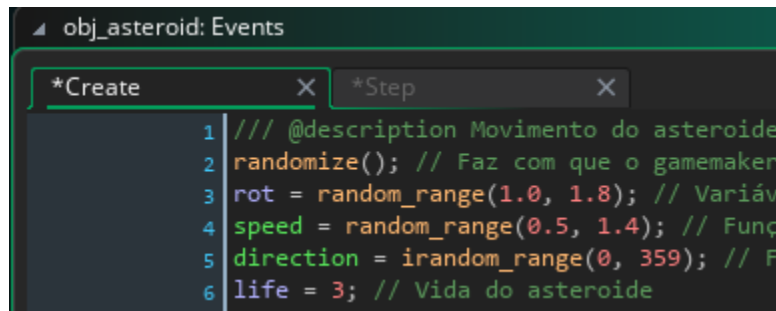
Colisão do projétil:

- Adicionar um evento de colisão com o asteroide no obj_bullet



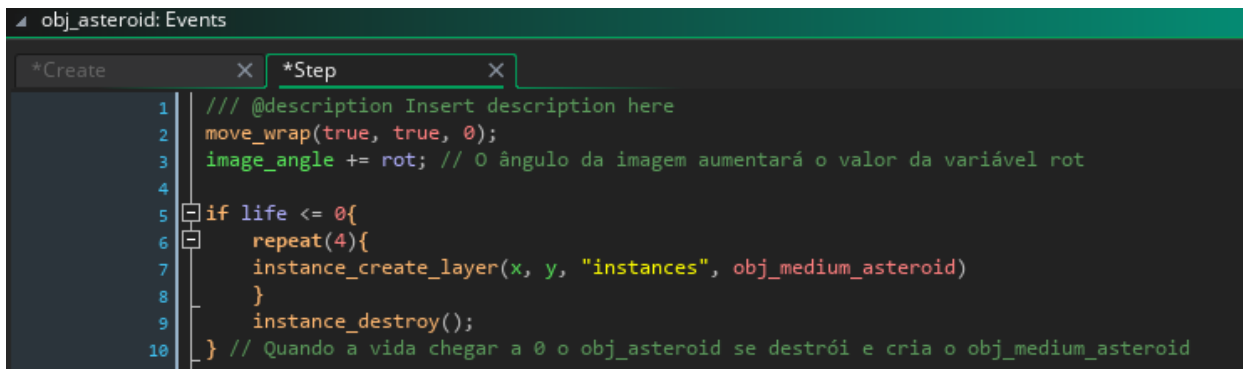
```
1 // @description Insert description here
2 // You can write your code in this editor
3 with(other){ // modifica o código do objeto no qual está colidindo
4     life -= 1;
5 }
6
7 instance_destroy();// Apaga a instância do obj_bullet
```

- Adicionar a variável life(vida) no evento criar do asteroide



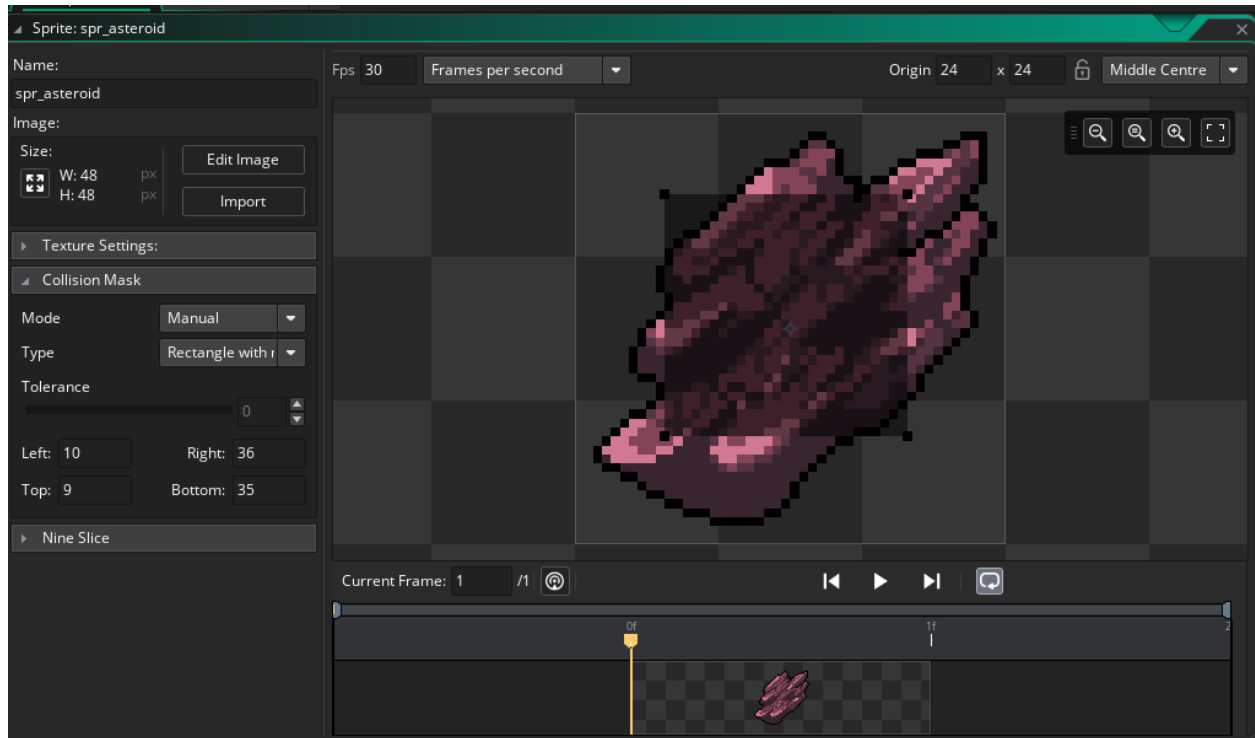
```
1 // @description Movimento do asteroide
2 randomize(); // Faz com que o gamemaker
3 rot = random_range(1.0, 1.8); // Variável
4 speed = random_range(0.5, 1.4); // Função
5 direction = irandom_range(0, 359); // Função
6 life = 3; // Vida do asteroide
```

- Adicionar código para o asteroide se destruir e criar outros menores

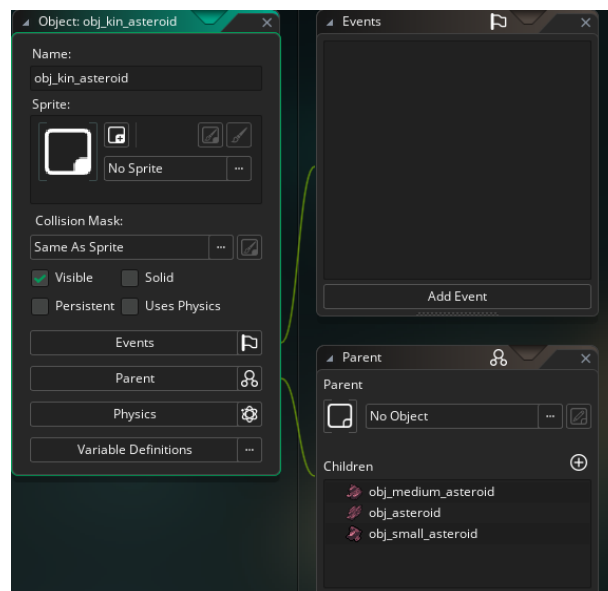


```
1 // @description Insert description here
2 move_wrap(true, true, 0);
3 image_angle += rot; // O ângulo da imagem aumentará o valor da variável rot
4
5 if life <= 0{
6     repeat(4){
7         instance_create_layer(x, y, "instances", obj_medium_asteroid)
8     }
9     instance_destroy();
10 } // Quando a vida chegar a 0 o obj_asteroid se destrói e cria o obj_medium_asteroid
```


- Ajustar máscara de colisão do asteroide



- Criar os sprites e objetos dos asteroides médios e pequenos e seguir os mesmos passo do asteroide grande
- Criar um objeto para a família dos asteroides



-
- Modificar evento de colisão do obj_bullet colocando o obj_kin_asteroid (Família dos asteroides)

