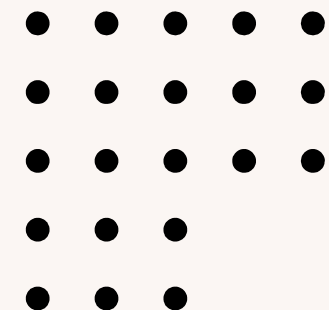
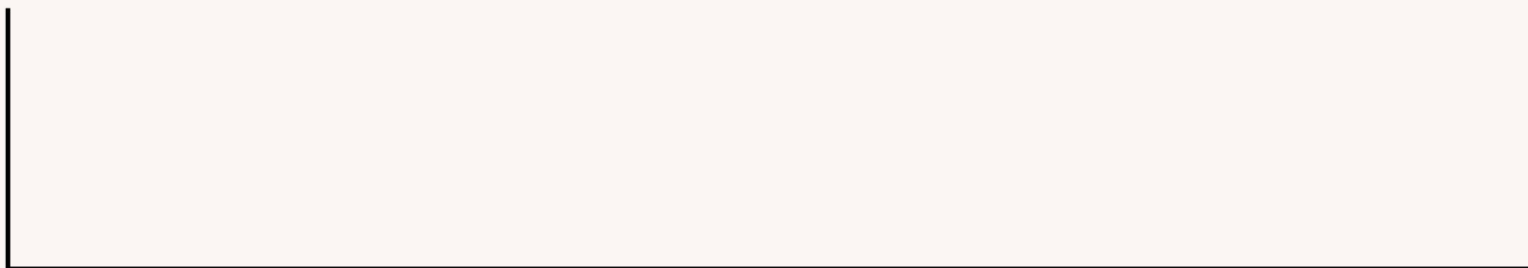
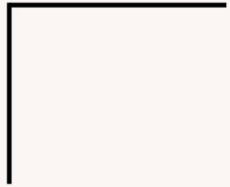


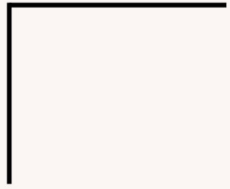
DESENVOLVIMENTO DE JOGOS

GAME MAKER

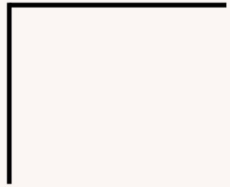




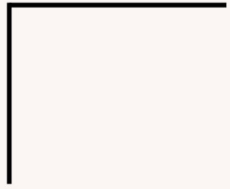
GameMaker é uma ferramenta projetada para capacitar você e sua equipe a fazer jogos novos e inovadores, bem como idéias de protótipos da maneira mais rápida e intuitiva possível através de múltiplas plataformas alvo. Destina-se principalmente como uma ferramenta para fazer jogos 2D - embora jogos 3D sejam perfeitamente realizáveis - e vem com uma série de ferramentas e editores para ajudá-lo a realizar seus sonhos e idéias, com seu projeto final sendo portado através de múltiplas plataformas a partir dos mesmos recursos básicos iniciais.



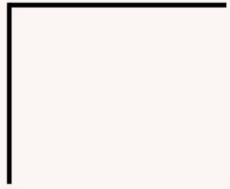
O GameMaker foi inicialmente criado como sendo uma ferramenta puramente de animação, pelo professor universitário Mark Overmars. Em 1999, a ferramenta foi então lançada ainda sob o nome de Animo, nome esse baseado na sua função original.



Após isso, Overmars continuou o desenvolvimento do Animo de modo que se tornasse uma ferramenta completa para o desenvolvimento de games. No ano de 2007, Overmars ingressou na Yoyo Games dando continuidade ao aperfeiçoamento da ferramenta. Em 2012 foi lançado o GameMaker Studio e sua segunda versão, GameMaker Studio 2, em 2017. Em janeiro de 2021 o GameMaker foi adquirido pela empresa Opera dando continuidade ao seu desenvolvimento.



A partir daí o GameMaker teve, e ainda está tendo, um crescimento exponencial, tanto de funcionalidades quanto relacionado ao número de usuários que buscam uma game engine mais especializada e completa.



O GameMaker hoje já oferece ferramentas necessárias e interessantes ao desenvolvimento e publicação de jogos 2D, como editor de scripts (fazendo uso da linguagem de programação GameMaker Language – GML e também da possibilidade do Visual Scripting), sprites, animação, sons, tilesets, fontes, entres outras.

Object: obj_Nano_Spider

Name: obj_Nano_Spider

Sprite:  25 x 25

Collision Mask: Same As Sprite ☒

☒ Visible ☐ Solid

☐ Persistent ☐ Uses Physics

Events

Parent

Physics

Events

Create

Destroy

Step

obj_Enemy_Parent

obj_Wall

Draw

Add Event

Parent

obj_Enemy_Parent

Children

Physics

Density: 0.5

Restitution: 0.1

Collision Group: 0

Linear Damping: 0.1

Angular Damping: 0.1

Friction: 0.2

☐ Sensor ☒ Start Awake ☐ Kinematic

Modify Collision Shape

```
Code
obj_Enemy_Parent x
1 var a,xoff,yoff;
2
3 a = point direction(x, y, other.x, other.y);
4 xoff = lengthdir(x(1, a));
5 yoff = lengthdir(y(1, a));
6
7 x += xoff;
8 y += yoff;
9 other.x += xoff;
10 other.y += yoff;
11
12 /*
13 if !spotted
14 (
15 direction+=5*turn;
16 other.direction+=5*turn;
17 )
18 */
```

Physics Collision Shape

Circle

X: 12 Y: 12

Radius: 100

Frame: 2

Grid: 32 x 32

Sprite: spr_Enemy_Nano_Spider

Name: spr_Enemy_Nano_Spider


Size: 25 x 25

Origin: 8 x 12

Texture Settings: ☐ Tile Horizontally ☐ Tile Vertically ☐ Separate Texture Page

Group: Game

+ Collision Mask

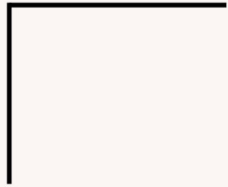


Speed: 1

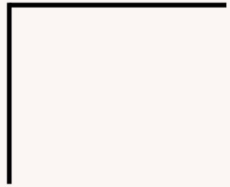
Frames per game frame

Current frame: 1 of 4 fr

```
Code
shaderGBush x shaderGBush x
1 // Simple passthrough vertex shader
2 //
3 //
4 attribute vec3 in_Position;          // (x,y,z)
5 //attribute vec3 in_Normal;          // (x,y,z) unused in this shader.
6 attribute vec4 in_Colour;            // (r,g,b,a)
7 attribute vec2 in_TextureCoord;      // (u,v)
8
9 varying vec2 v_Texcoords;
10 varying vec4 v_Colour;
11
12 void main()
13 {
14     vec4 object_space_pos = vec4( in_Position.x, in_Position.y, in_Position.z, 1.0);
15     gl_Position = gm_Matrices[MATRIX_WORLD_VIEW_PROJECTION] * object_space_pos;
16
17     v_Colour = in_Colour;
18     v_Texcoord = in_TextureCoord;
19 }
20
21
```



Também a game engine exporta para a maior parte das plataformas existentes, como Desktop, Consoles, Mobile, Web e UWP. Mesmo sendo bem completa, os desenvolvedores internos da ferramenta ainda estão buscando constante atualização, como performance de runtime, interface, editores, dentre outros, de modo a torná-la cada vez mais eficiente.



Por essas razões o GameMaker tem sido cada vez mais adotado por desenvolvedores independentes e profissionais. Além disso, a ferramenta também tem sido considerada ótima para o ensino e aperfeiçoamento de desenvolvedores de jogos digitais, em cursos técnicos, universidades e pós-graduações.

Glossário

1.Sprites:

- **Definição:** Imagens ou animações que representam personagens, objetos, ou outros elementos visuais em um jogo.

2. Objects (Objetos):

- **Definição:** Componentes centrais no GameMaker que representam elementos do jogo. Eles usam sprites para a representação visual e contêm eventos e ações que definem seu comportamento.

3. Rooms (Salas):

- **Definição:** Equivalentes a níveis ou fases em um jogo. São espaços onde os objetos são colocados e onde a maior parte da ação do jogo ocorre.

4. Events (Eventos):

- **Definição:** Ações específicas que ocorrem em um jogo, como pressionar uma tecla, colidir com outro objeto, ou passagem de tempo. São utilizados para acionar respostas nos objetos.

5. Actions (Ações):

- **Definição:** Comandos que os objetos executam em resposta a um evento. Podem ser tão simples quanto mover um objeto ou mais complexos, envolvendo a execução de código.

6. GML (GameMaker Language):

- **Definição:** A linguagem de programação usada no GameMaker. É uma linguagem de scripting que permite aos usuários escreverem lógicas de jogo complexas.

7. Drag and Drop (Arrastar e Soltar):

- **Definição:** Uma interface visual no GameMaker que permite aos usuários criar lógicas de jogo sem escrever código, através do arrastar e soltar de ações em eventos.

8. Assets:

- **Definição:** Recursos usados em um jogo, incluindo sprites, sons, músicas, scripts, e outros dados como fontes e texturas.

9. Layer (Camada):

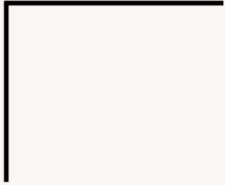
- **Definição:** Em um Room, as camadas são usadas para organizar a renderização de objetos, sprites, ou outros elementos, como fundos. Camadas podem ser sobrepostas para criar efeitos de profundidade.

10. Scripts:

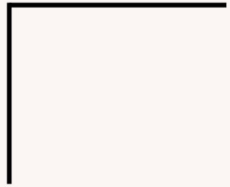
- **Definição:** Blocos de código GML que podem ser usados para executar funções mais complexas dentro de um objeto ou em toda a lógica do jogo.

11. Debugger:

- **Definição:** Uma ferramenta que permite aos desenvolvedores testar e depurar seu jogo, verificando erros e monitorando o comportamento de variáveis e operações em tempo real.



Conhecendo o programa



Quando você abrir o GameMaker pela primeira vez, você será solicitado a entrar em sua conta YoYo ou Opera Account. Uma vez registrado e logado, a IDE principal do GameMaker será aberta na página inicial

IDE Menu Bar

File Edit Build Windows Tools Marketplace Layouts Help Source Control

IDE v2.3.5.589 Runtime v2.3.5.458

Version Details



gmatharoo

Last Login:
Tuesday, 26 October, 2021 9:13 AM

User Details

GET STARTED

Project Options

New

Open

Import

Tutorials

Opens in web browser

RECENT PROJECTS

Recent Projects



Dungeons Modified - Tuesday, 26 October, 2021 12:44 PM
I:\User Files\Documents\GameMakerStudio2\Dungeons Modified\Dungeons Modified.yyp



GMC_JAM_40 - Tuesday, 26 October, 2021 12:44 PM
I:\User Files\Documents\GameMakerStudio2\GMC_JAM_40\GMC_JAM_40.yyp



crafter-2.3 - Tuesday, 26 October, 2021 12:44 PM
I:\User Files\Documents\GameMakerStudio2\crafter-2.3\crafter-2.3.yyp



Fire Jump Demo Project - Tuesday, 26 October, 2021 12:44 PM
I:\User Files\Documents\GameMakerStudio2\Fire Jump Demo Project\Fire Jump Demo Project.yyp



BLOG

VIEW ALL BLOGS



Useful Resources



Quando você inicia o GameMaker pela primeira vez, você poderá ver uma descrição para cada elemento da tela inicial, pairando sobre ele. Aqui, você pode clicar no botão "SKIP to Setup Wizard" para começar a criar seu primeiro projeto com a ajuda do Setup Wizard, que o levará através dos passos necessários para criar seu primeiro jogo no GameMaker.

1

"New Project"

Start from scratch with a new empty project or start from a template; one we've included or one of your own

SKIP to Setup Wizard

GET STARTED

1

New

2

Open

3

Import

4

Tutorials

Opens in web browser

Elementos da página inicial

A página inicial contém os seguintes elementos:

- Barra de menu IDE: Isto mostra os menus que você pode utilizar em toda a IDE; mais informações são dadas em uma seção abaixo.
- Opções de projeto: A seção Projetos é onde você pode criar, abrir ou importar projetos. Isto é explicado com mais detalhes mais abaixo nesta página.
- Projetos recentes: Aqui você pode ver uma lista de projetos anteriores que você pode abrir. Você também pode ver o nome completo do projeto e path, e clicando no botão esquerdo do mouse abrirá o projeto. Você também pode switch a visualização desta seção entre o modo tile e o modo lista, pressionando os botões no canto superior direito.

Elementos da página inicial

A página inicial contém os seguintes elementos:

- Detalhes da versão: Esta seção fornece detalhes sobre a versão atual IDE que está sendo utilizada, assim como a atual versão Tempo de execução. Você também será notificado de quaisquer mudanças disponíveis no IDE ou no Runtime nesta seção.
- Detalhes do usuário: Isto mostra o nome de usuário de sua conta YoYo e a última vez que você entrou no site.
- Recursos úteis: Isto mostra vários tiles em que você pode clicar para acessar os recursos oficiais do tutorial do GameMaker.

Após fazer o login e iniciar um novo projeto, o GameMaker o levará ao espaço de trabalho inicial com algumas janelas básicas acopladas à IDE. Em geral, o workspace é simplesmente uma área onde você pode organizar os diferentes assets para seu jogo enquanto você está trabalhando:

Quick Buttons

Compile Target

Asset Browser

Inspector

Workspace

Output

Dock Show/Hide

The screenshot displays the Godot Engine 4.0 IDE interface. The top menu bar includes File, Edit, Build, Windows, Tools, Marketplace, Layouts, Help, and Source Control. The top status bar shows the version (v2022.900.0.209) and runtime version (v2022.900.0.189). The interface is divided into several panels and docks:

- Inspector:** Located on the left, it shows the selected object (obj_Control) and its properties. It includes a search bar, a list of objects (obj_Control, obj_Player), and a section for the selected object's properties (Name, Sprite, Collision Mask, Visible, Solid, Persistent, Uses Physics, Events, Parent, Physics, Variable Definitions). A message at the bottom states: "This Object currently has no Events. Add Events here or via the editor".
- Workspace 1:** The central area for editing the scene. It shows a 2D scene with a character sprite (obj_Player) and a control object (obj_Control). The Events panel on the right shows a list of events (Create - Initialize, Destroy - Destroy Event, Step - Basic Movement, obj_Control - Col Ide) and a button to Add Event. The Physics panel shows properties for the selected object (Density: 0.5, Restitution: 0.1, Collision Group: 1, Linear Damping: 0.1, Angular Damping: 0.1, Friction: 0.2, Sensor, Start Awake, Kinematic) and a button to Modify Collision Shape. The Physics Collision Shape panel shows a circle shape with a radius of 122 and a position of (36, 22).
- Asset Browser:** Located on the right, it shows a list of assets (obj_Control, obj_Player) and a search bar. It includes a section for Quick Access (Recent, Favourites, Room Order, Saved Searches, Tags, Game Options) and a list of asset types (Animation Curves, Extensions, Fonts, Notes, Objects, Paths, Rooms, Scripts, Sequences, Shaders, Sounds, Sprites, Tile Sets, Timelines).
- Output:** Located at the bottom, it shows the output of the program. It includes a search bar and a list of output messages (C:\ProgramData\GameMakerStudio2-Beta\Cache\runtimes\runtime-2022.900.0.189\windows\x64\Runner.exe DONE (0), Igor complete, elapsed time 00:00:04.9732214s for command "C:\ProgramData\GameMakerStudio2-Beta\Cache\runtimes\runtime-2022.900.0.189\bin\igor\windows\x64\Igor.exe", SUCCESS: Run Program Complete).

Como você pode ver, a inicial workspace está em uma aba na parte superior da tela (e você pode renomeá-la clicando duas vezes na aba), mas você pode criar mais workspaces para o projeto clicando no

ao lado, dando-lhe várias possibilidades workspaces para qualquer projeto único. Por exemplo, talvez você esteja trabalhando em interações entre o jogador e vários inimigos objects, assim você teria o jogador no seu próprio workspace e o inimigo objects em outro, e talvez outro workspace apenas para mostrar o scripts que ambos usam.

Outra característica importante de workspaces é que você pode clicar na aba e - ainda segurando o botão do mouse para baixo - arrastá-lo da janela principal IDE para sua própria janela individual, tornando muito fácil organizar as coisas se você estiver usando - por exemplo - vários displays.

Você também funde estas janelas secundárias workspace novamente na principal, arrastando a aba de volta para a primeira janela. Note que, embora você pareça ter duas instâncias do IDE funcionando quando você faz isso, elas são ambas para o mesmo projeto e você não pode ter um projeto em um e outro no outro a menos que você abra especificamente duas instâncias do GameMaker.

Quando você iniciar o GameMaker pela primeira vez, seu workspace já será populado por um par de janelas que serão "encaixadas" no IDE

Ativos - À direita da tela você pode encontrar o [Navegador de ativos](#). Aqui é onde você pode criar e editar os diferentes assets que seu jogo utiliza, assim como gerar e alterar configurações, opções de jogo, ordem room e outras coisas. Assets são criados clicando com o botão direito do mouse em em uma pasta asset (ou em qualquer lugar na principal Asset Vista do navegador) e selecionando Criar, ou clicando no ícone mais na parte superior e selecionando um asset na janela Criar Ativo (note que com este método você pode criar vários assets do mesmo tipo de cada vez, definindo o valor na parte inferior)

Saída - Você também pode ver na imagem acima da Janela de Saída. Há uma série de sub guias nesta janela relacionadas ao Controle de Fonte, Pesquisa e Depuração, sendo a guia inicial para a saída do console/compilador, que mostra o que está acontecendo quando você está compilando um jogo para teste ou quando está criando um pacote final para distribuição. Isto também mostrará quaisquer mensagens de debug que você escolher enviar de seu projeto em runtime, e pode ser configurado para mostrar diferentes quantidades de informações das Preferências Gerais. Se você fechar a janela de saída e desejar recuperá-la, você pode usar o Menu Windows.

Navegador de assets

O núcleo de seu jogo será criado a partir de assets adicionados ao Navegador de Ativos localizado por padrão à direita da IDE. Aqui é onde você pode adicionar tudo que seu jogo requer para rodar, incluindo um jogo room, sprites, objects, paths e uma série de outras coisas. Um jogo básico no GameMaker exigirá um room para rodar (novos projetos serão sempre criados com um room asset já criado), e normalmente pelo menos um object e um sprite, embora você provavelmente usará muito mais!

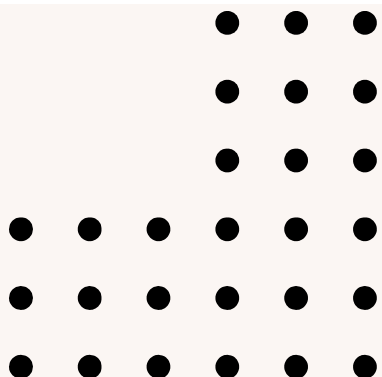


Ao trabalhar com o navegador Asset, você pode usar o botão direito do mouse em qualquer recurso ou pasta para abrir um menu de opções:

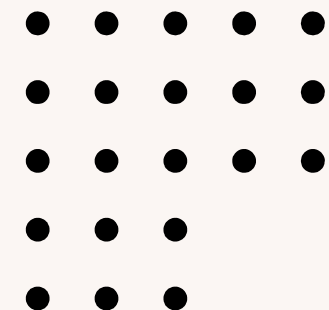
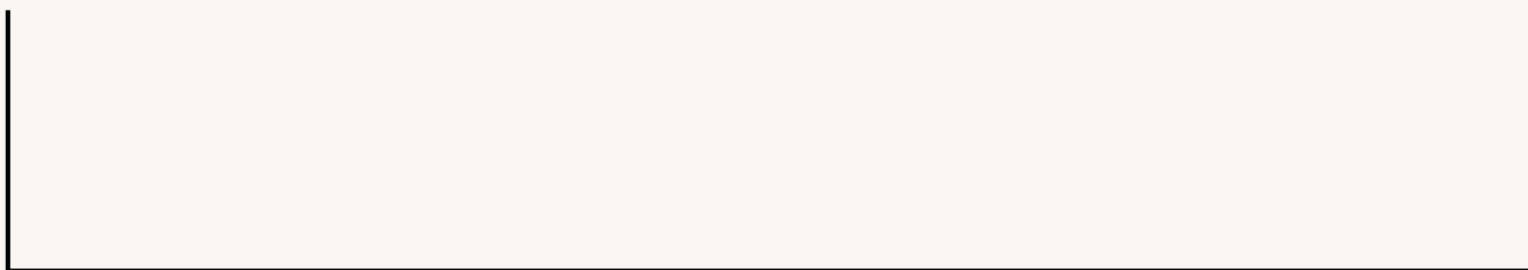
- Criar - Criar um novo asset. Ao selecioná-lo, você receberá uma lista dos tipos asset disponíveis que podem ser criados para que você possa selecionar um. Note que se você abrir este menu em um asset específico, haverá uma opção adicional para criar um recurso se o mesmo tipo listado logo abaixo para que você não tenha que procurar o tipo asset na lista.
- Edit (All) - Abra o editor de propriedades de recursos para o(s) ativo(s) ou grupo de pastas selecionado(s) (note que você também pode clicar duas vezes em um único asset para abri-lo para sua edição).
- Edit Tags - Isto abrirá o editor de tags onde você poderá adicionar ou remover tags ao(s) ativo(s) ou grupo(s) de pastas selecionado(s).

- Renomear - Renomeie o asset ou o grupo de pastas.
- Duplicar - Criar um duplicado do asset ou assets que foram selecionados.
- Favorito - Marque um asset ou assets como sendo um favorito, adicionando-o à seção Acesso Rápido no topo do navegador Asset.
- Adicionar Existente - Adicionar existente assets de uma pasta de projeto GameMaker diferente. Observe que assets criado usando uma versão anterior à 2.3.0 exigirá conversão, pois o formato do arquivo do projeto mudou com a atualização 2.3.0. Isto significa que pode ser preferível importar o projeto inteiro e converter tudo e depois salvá-lo em um novo local se o assets for importado freqüentemente para outros projetos.

- Renomear - Renomeie o asset ou o grupo de pastas.
- Duplicar - Criar um duplicado do asset ou assets que foram selecionados.
- Favorito - Marque um asset ou assets como sendo um favorito, adicionando-o à seção Acesso Rápido no topo do navegador Asset.
- Adicionar Existente - Adicionar existente assets de uma pasta de projeto GameMaker diferente. Observe que assets criado usando uma versão anterior à 2.3.0 exigirá conversão, pois o formato do arquivo do projeto mudou com a atualização 2.3.0. Isto significa que pode ser preferível importar o projeto inteiro e converter tudo e depois salvá-lo em um novo local se o assets for importado freqüentemente para outros projetos.



PROJETINHO GAME DO DINO



Começamos criando um novo projeto vazio de jogo

Welcome To GameMaker



COMEÇAR

Novo

Abrir

Importar

Tutoriais



Abre no navegador da Web

SELECT PROJECT TYPE



Game



Live Wallpaper



Game Strip

SELECT TEMPLATE

Mostrar TODOS



Blank Game



SHOOTER



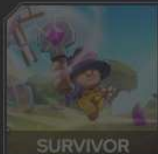
ACTION



RUNNER



SHOOTER



SURVIVOR



PUZZLE



MATCH 3



BRICK BREAKER

BLANK GAME



DESCRIPTION

Create an empty GameMaker project. Build your game up from a blank canvas.

Nome do Projeto:

Dino game 1

Local:

C:\Users\WITOR\Documents\GameMakerStudio2

Cancelar

Vamos Lá!

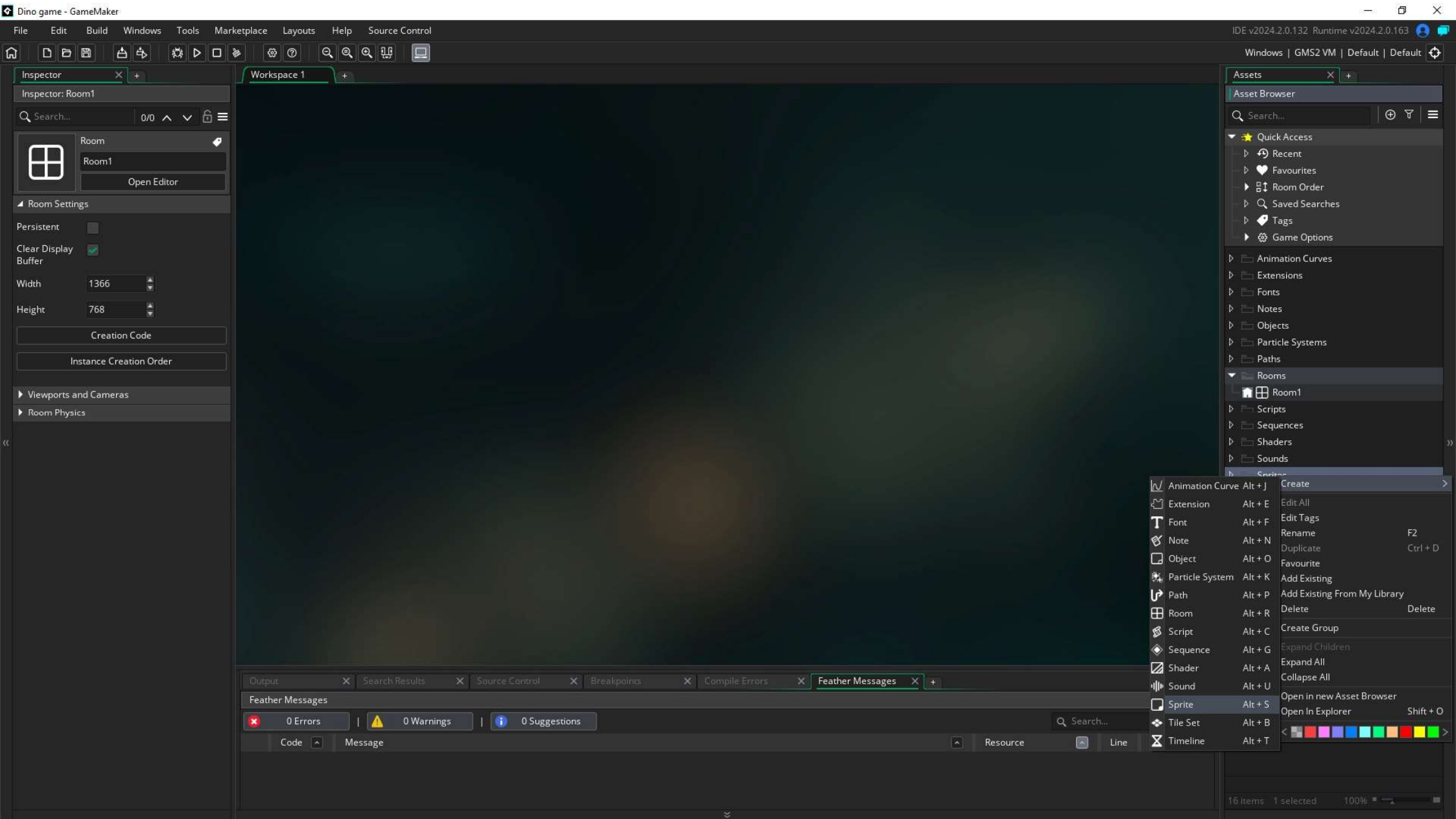


MAKE YOUR
OWN ARCADE
GAME IN
15 MINUTES

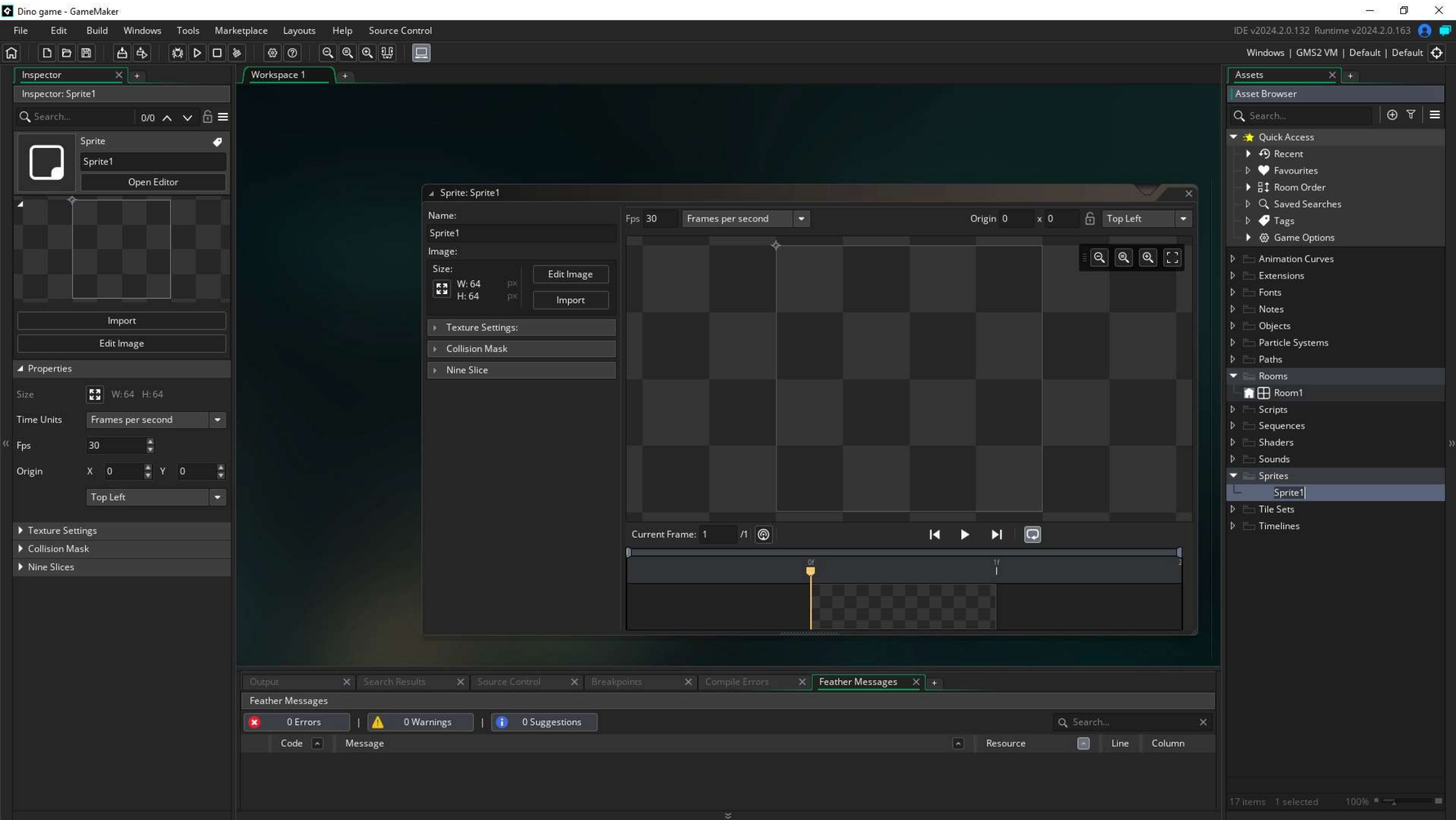


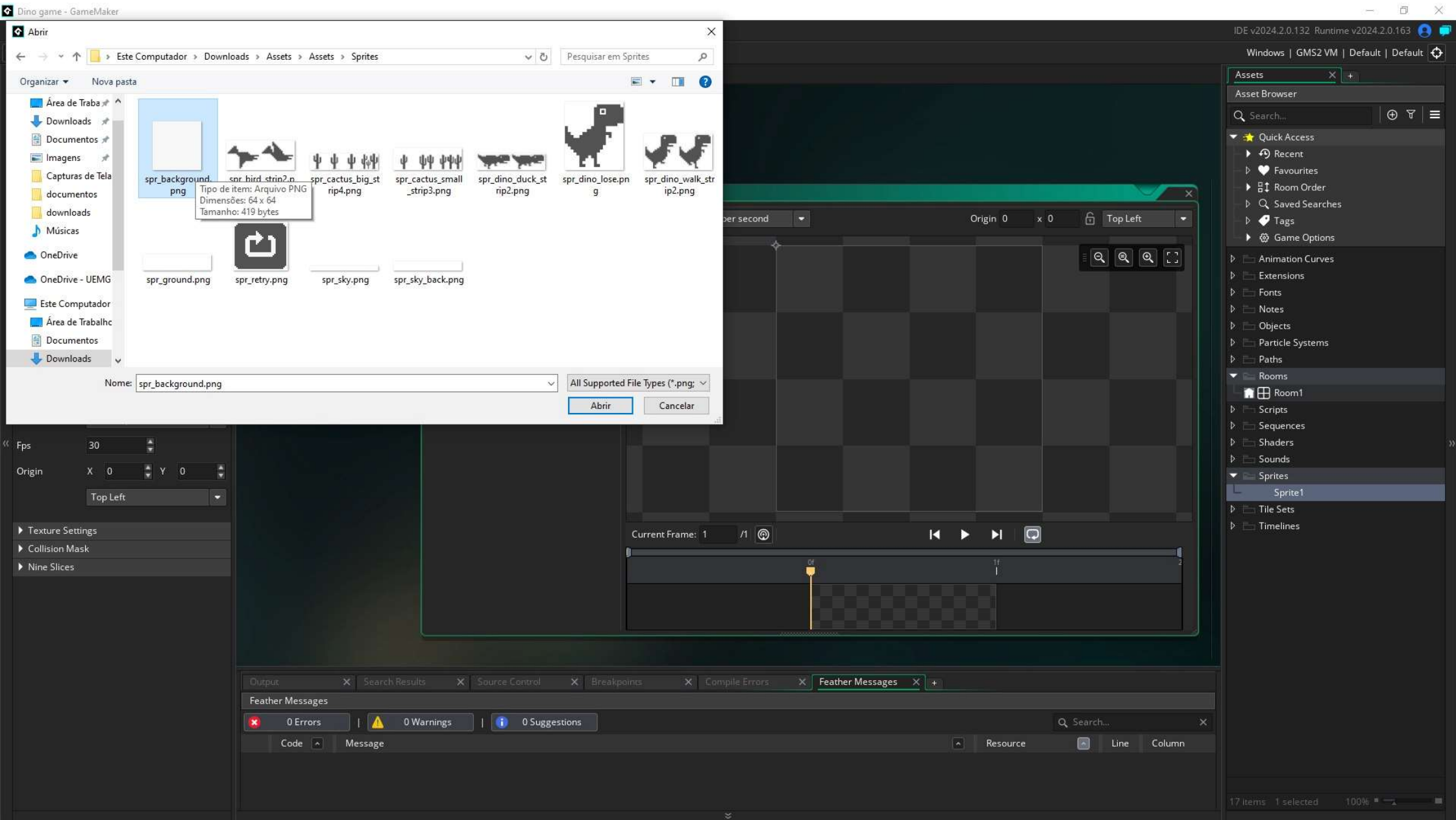
Com o projeto criado, podemos navegar nas pastas de assets e afins, na barra lateral do lado direito e mudar o tamanho da tela do jogo ao ir na pasta cenas ou rooms e dentro dela selecionar o room 1. No lado esquerdo as configurações do room irão aparecer, podemos colocar algo como 1024 x 576 ou maior.

Agora, precisamos adicionar sprites no jogo. Clicando com o botão direito sobre a pasta sprites, que vai estar vazia, podemos selecionar a primeira opção (criar ou create) e selecionar sprite

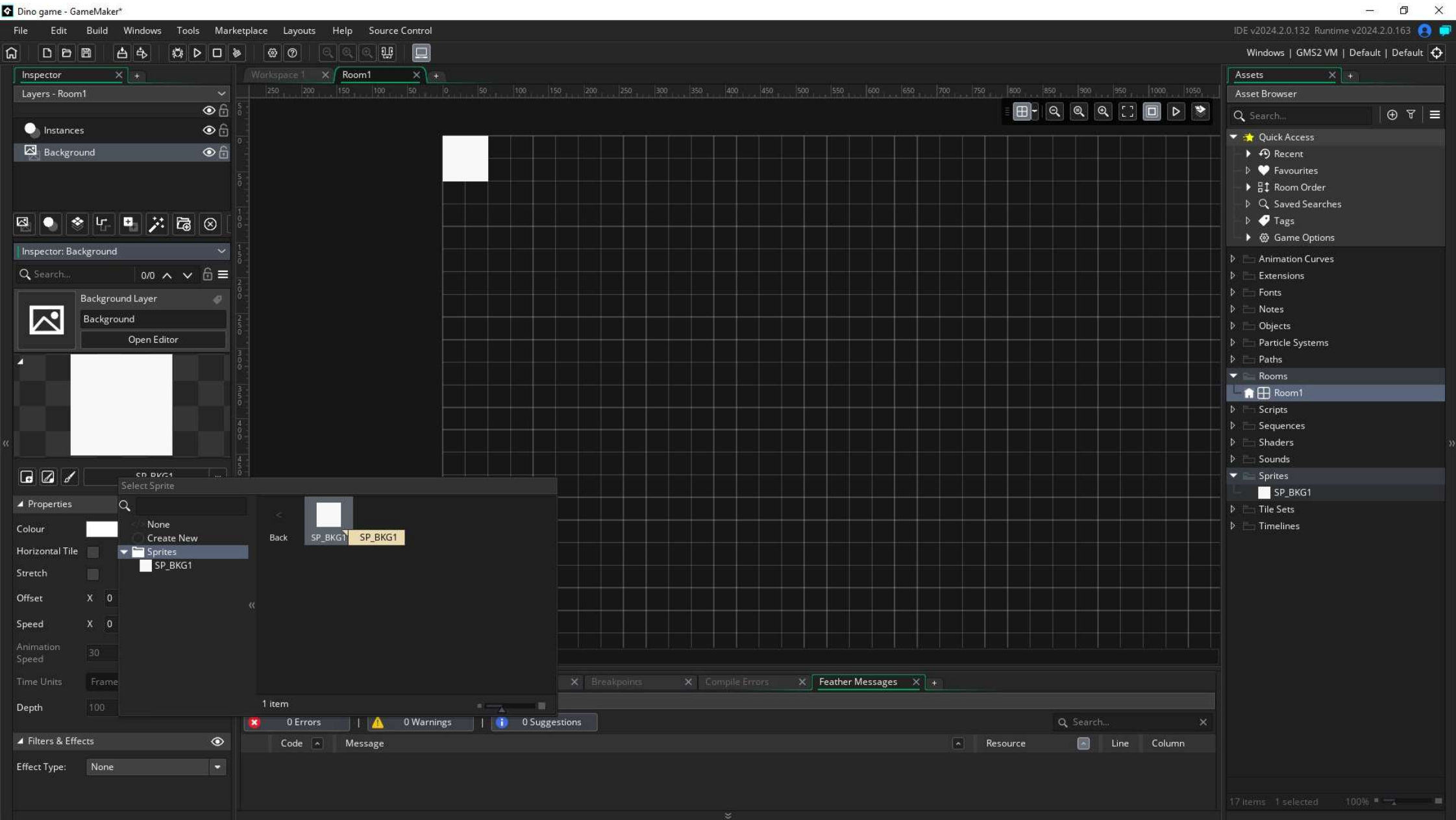


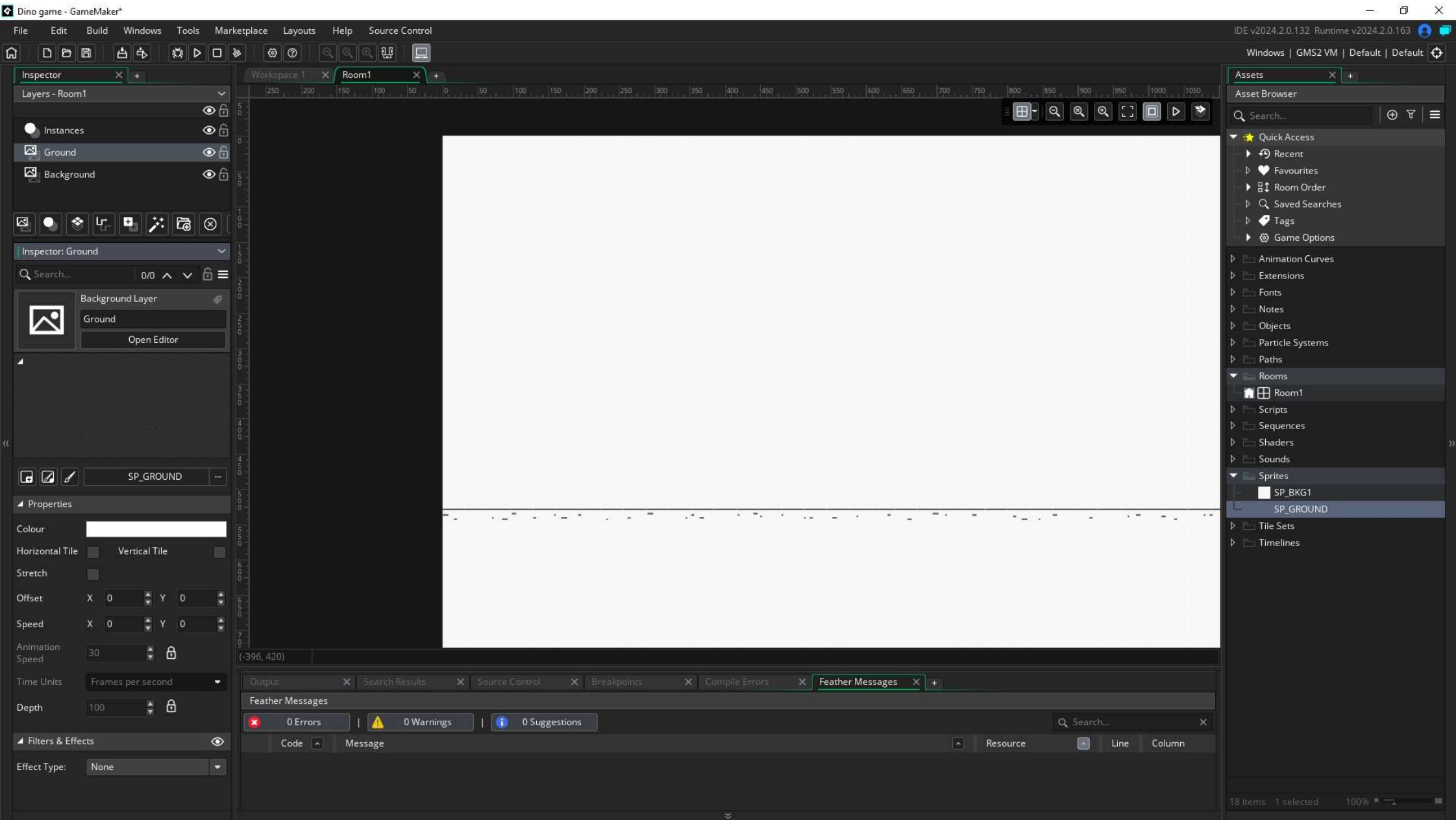
Dentro da janela de criação de sprite, vamos importar os assets que baixamos, primeiramente, vamos selecionar o background. Para fins de evitar conflito de nomenclatura, podemos adicionar SP_ antes do nome background ou apenas BGK.



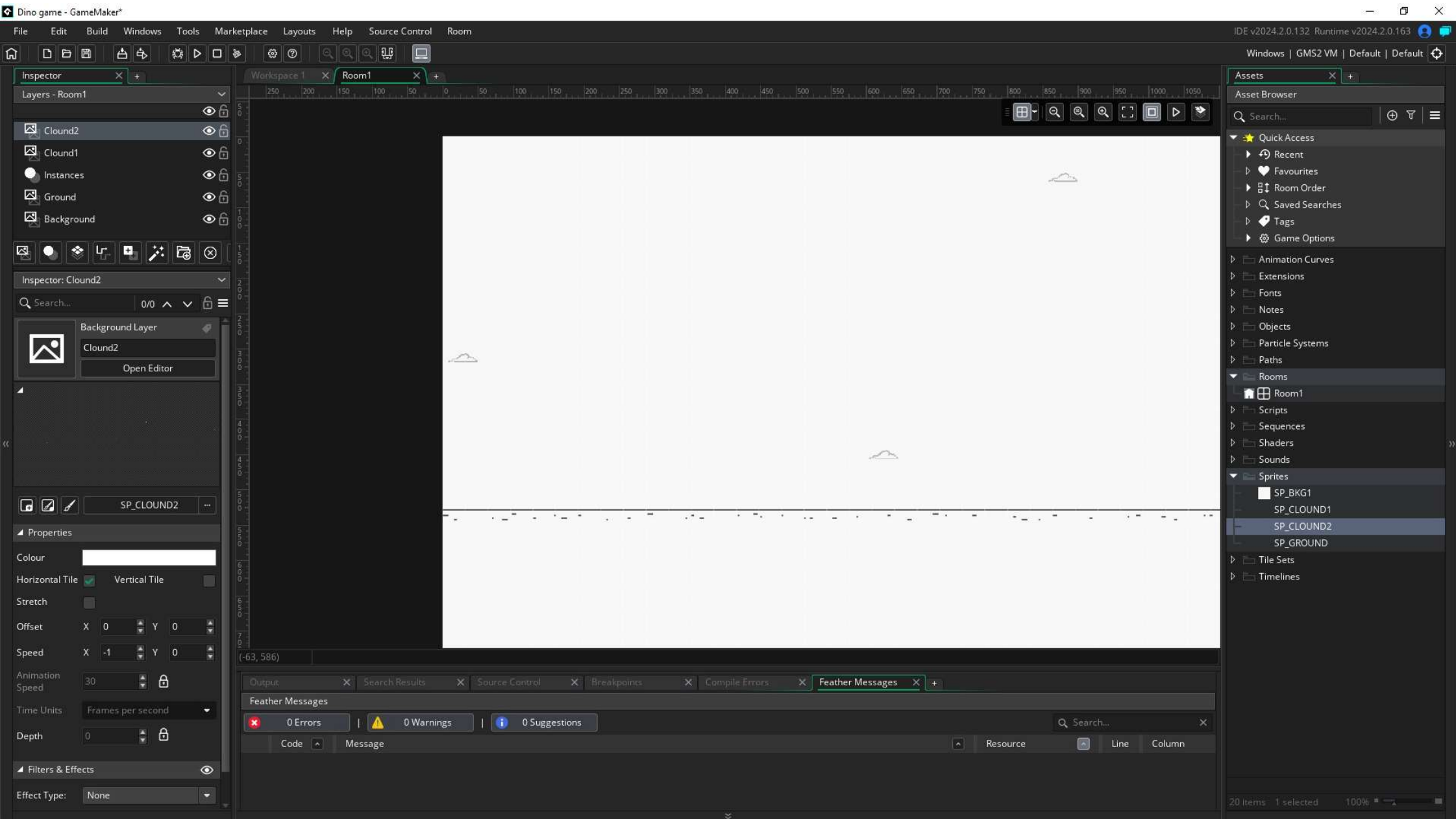


Em seguida, clicar 2X no room 1, para poder abrir suas configurações de layer no inspector do lado esquerdo. Lá teremos a configuração das camadas da cena e uma delas sendo o background. Ao selecionarmos essa layer, iremos adicionar o sprite criado, e ao adicionarmos, iremos marcar as caixas Horizontal Tile e Vertical tile, para expandir a imagem. Em seguida, iremos adicionar mais uma layer, para colocar outros elementos da cena.

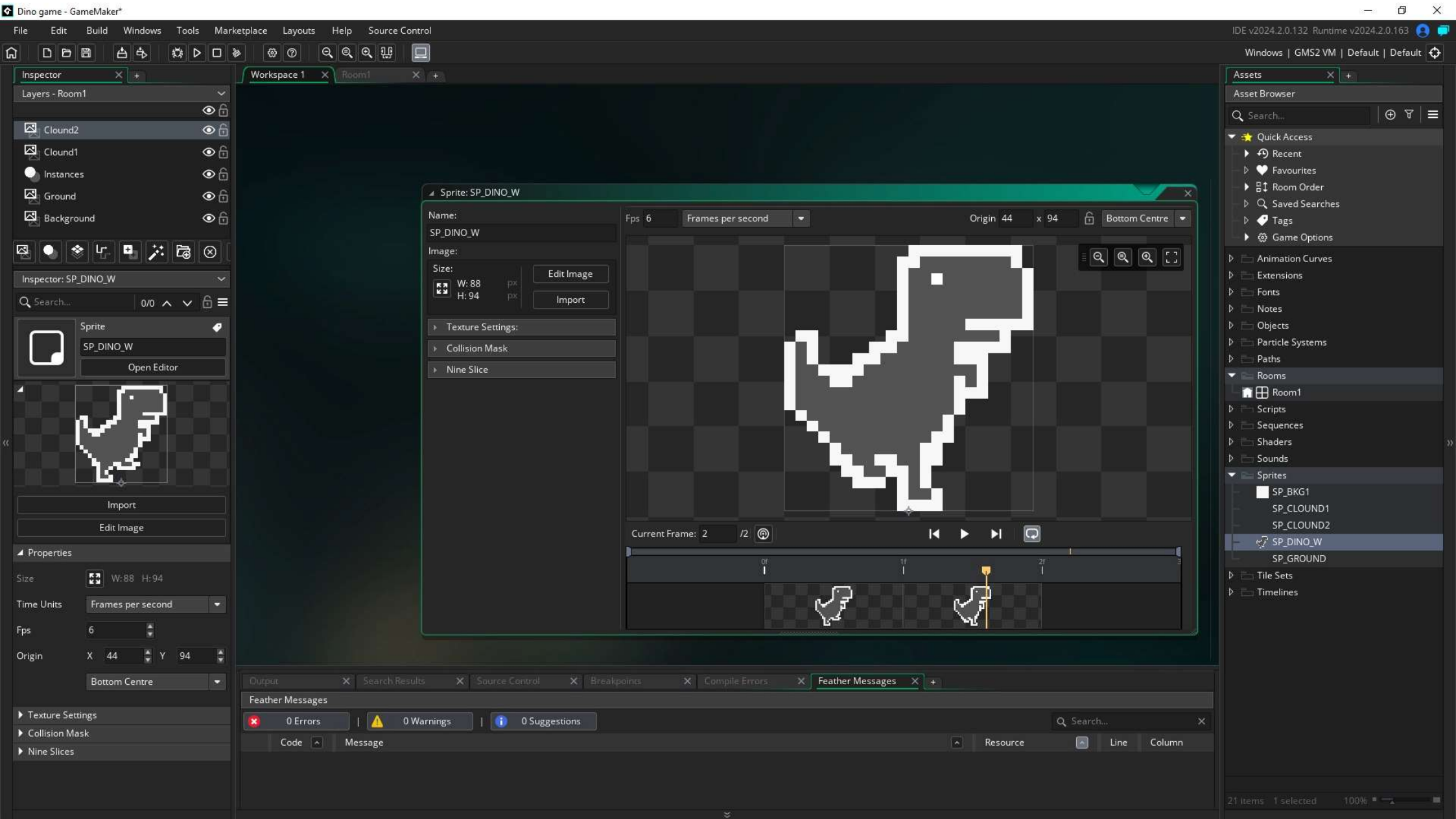




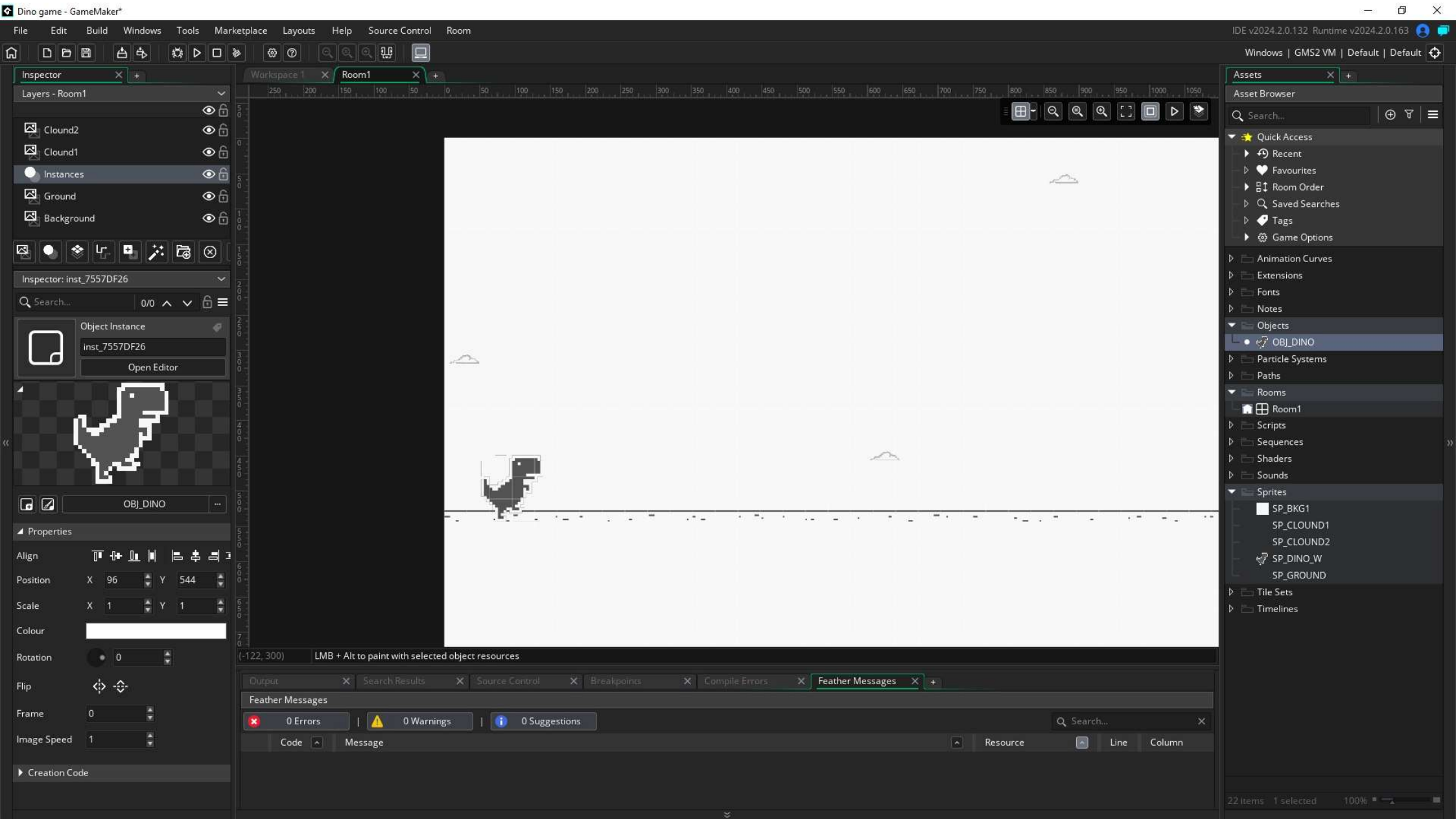
Agora, podemos adicionar uma “movimentação ao chão” para a camada ground criada. Primeiramente, selecionamos a caixa Horizontal Tile E em Speed, colocamos -1. Em seguida, iremos adicionar as nuvens, fazendo dois assets separados por numeração. Criamos 2 layers novas, uma para cada sprite de nuvem e nelas, após adicionar o sprite de nuvem, marcamos a opção de Horizontal Tile, e colocamos velocidades variadas para cada, como - 1 e - 0.5



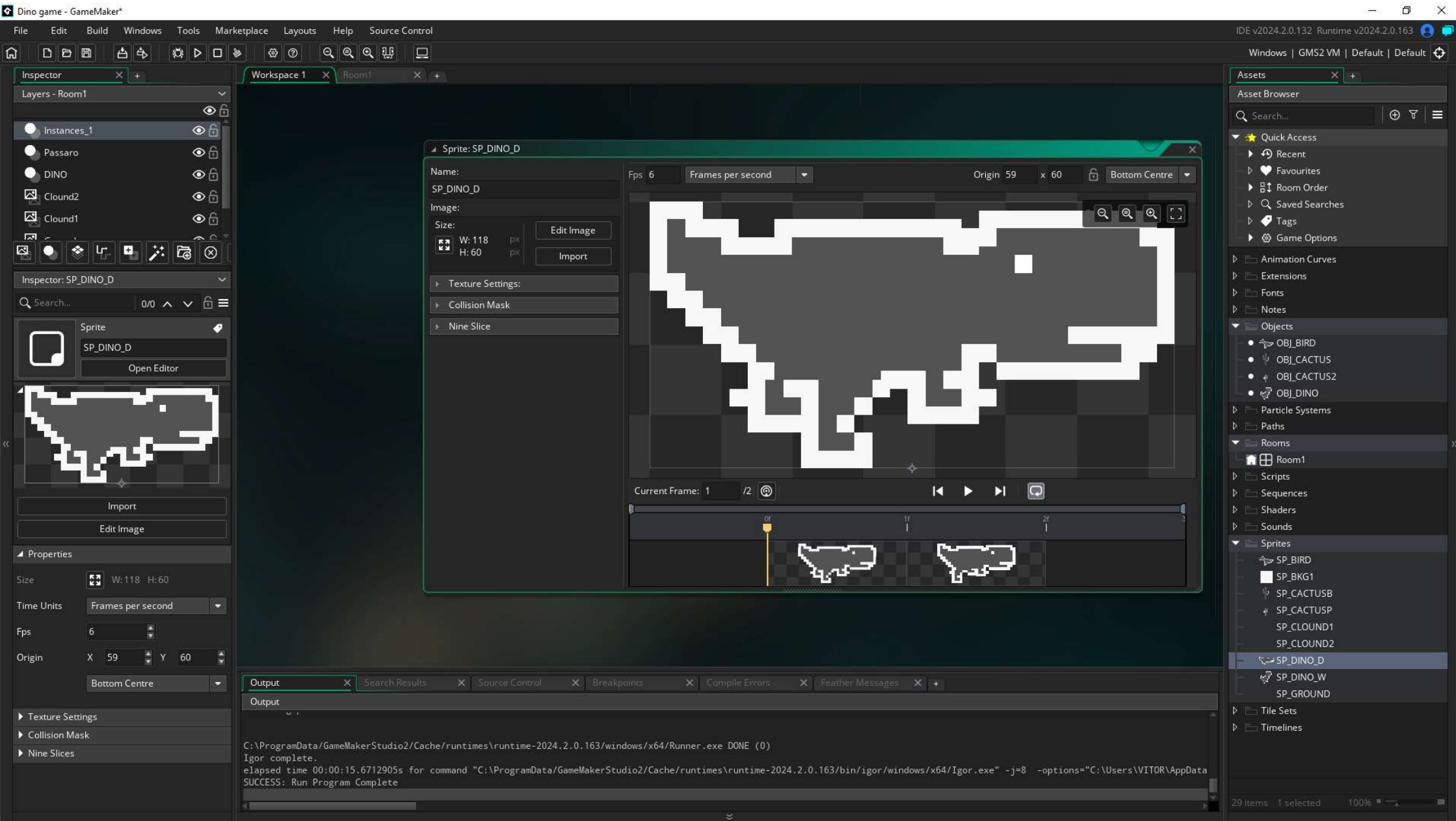
Feito isso, iremos adicionar nosso personagem, o dino, com um sprite que possui animação. Na caixa de criação de sprites, selecionar o arquivo do dino walk (imagem com dois dinos) nele iremos mudar de 30 fps para 6, para reduzir a velocidade de animação e definiremos sua origem para botton centre (origem de desenho do objeto)



Agora adicionaremos um objeto, clicando com o botão direito sobre a pasta objects. Nesse object, iremos nomear de OBJ_DINO e dentro de suas configurações, adicionar o sprite do dino. Feito isso, podemos selecionar o objeto e arrasta-lo para dentro da cena (podemos ver isso dentro da layer instance)

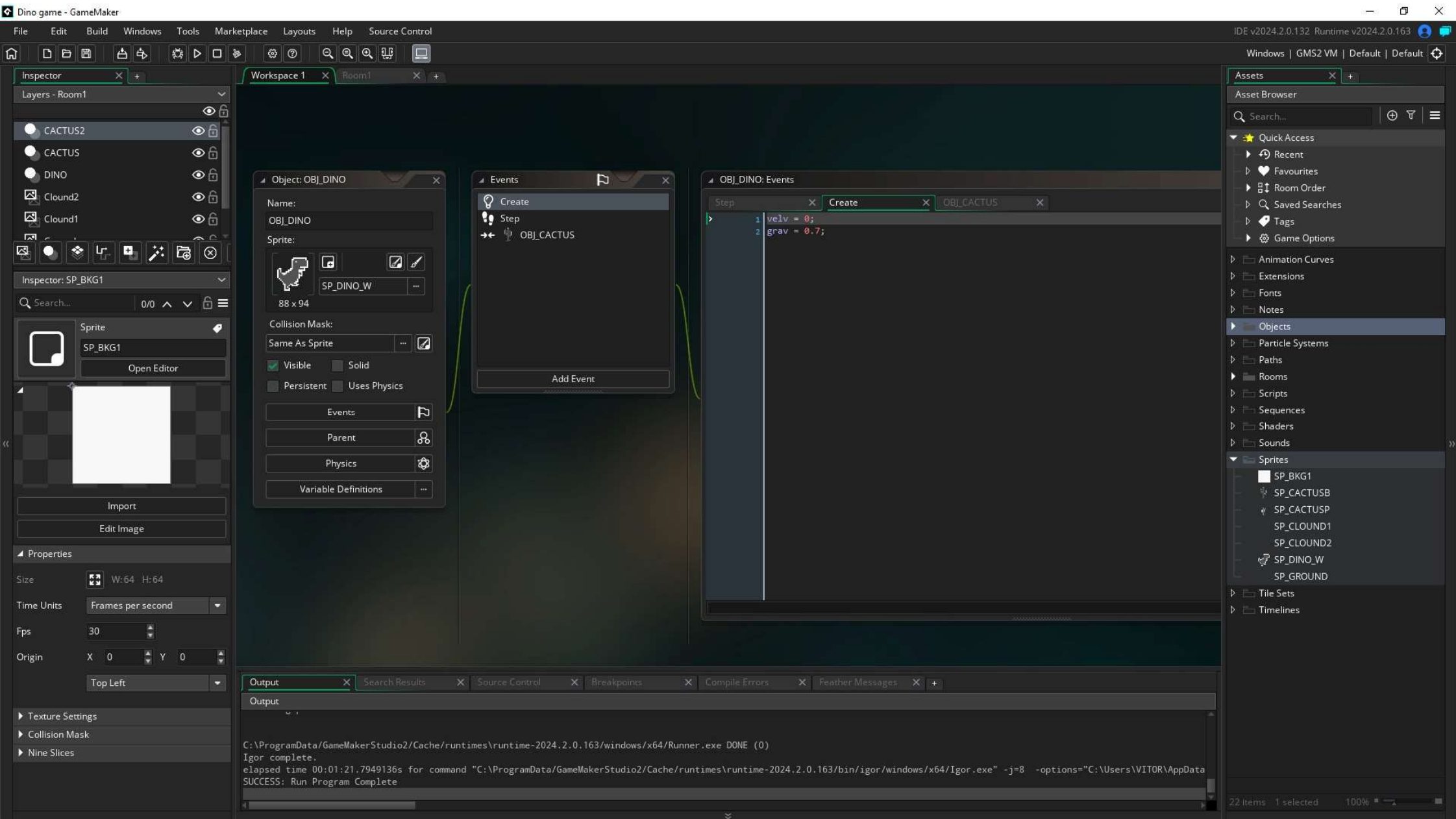


Agora iremos adicionar o sprite para o movimento de correr abaixado do Dino. De forma similar a feita anteriormente, selecionamos a opção de adicionar sprite, importamos o dino abaixado, definimos seu nome, definimos o centro do frame como botton centre e a velocidade para 8 fps.



Agora, iremos voltar para o workspace inicial, iremos no obj criado e nele, na janela eventos, adicionaremos o evento create e o evento step. No evento create, iremos colocar nossa variável de velocidade vertical e de gravidade

```
velv = 0;  
grav = 0.7;
```

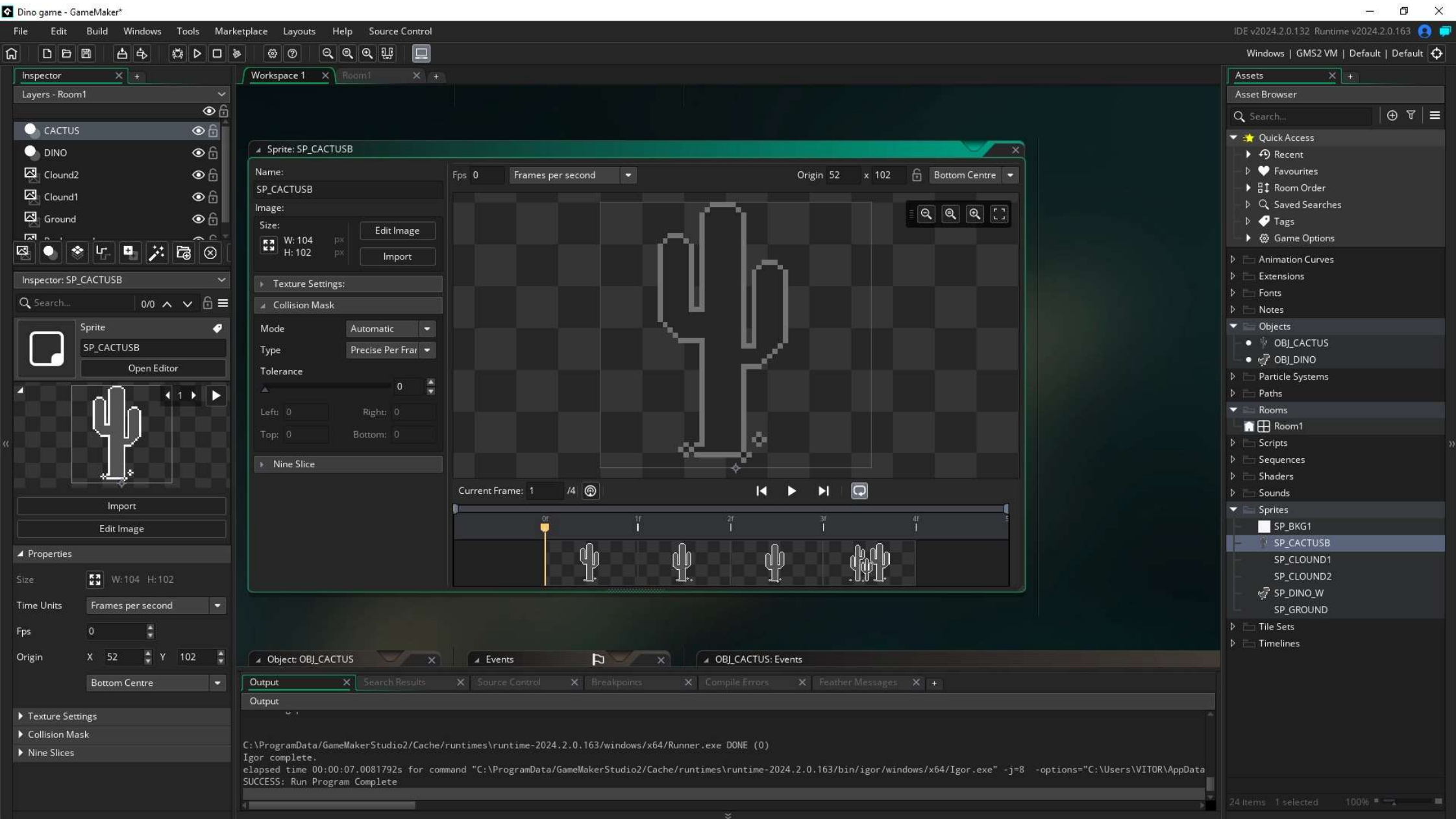



Adicionaremos também um evento step simples e nele adicionaremos o seguinte código:

```
1 if (y >= 544)
2 {
3     y = 544;
4     velv = 0;
5     if (keyboard_check_pressed(vk_space))
6     {
7         velv -= 18;
8     }
9
10    image_speed = 1;
11    if (keyboard_check(vk_down))
12    {
13        sprite_index = SP_DINO_D
14    }
15 }
16 else
17 {
18     sprite_index = SP_DINO_W
19 }
20 }
```

```
21 else
22 {
23     if (keyboard_check(vk_down))
24     {
25         grav = 3;
26     }
27     else
28     {
29         grav = 0.7;
30     }
31
32     velv += grav;
33     image_speed = 0;
34 }
35 y = y + velv;
```

Agora nosso personagem pula e sua animação para ao saltar e retoma ao tocar o “solo”. Então, adicionaremos os elementos do cenário, sendo eles os cactus grandes. Na janela de criação, colocar 0 frames p/s e colocar a origem como botton centre e em colision mask, selecionar o tipo como precise per frame.



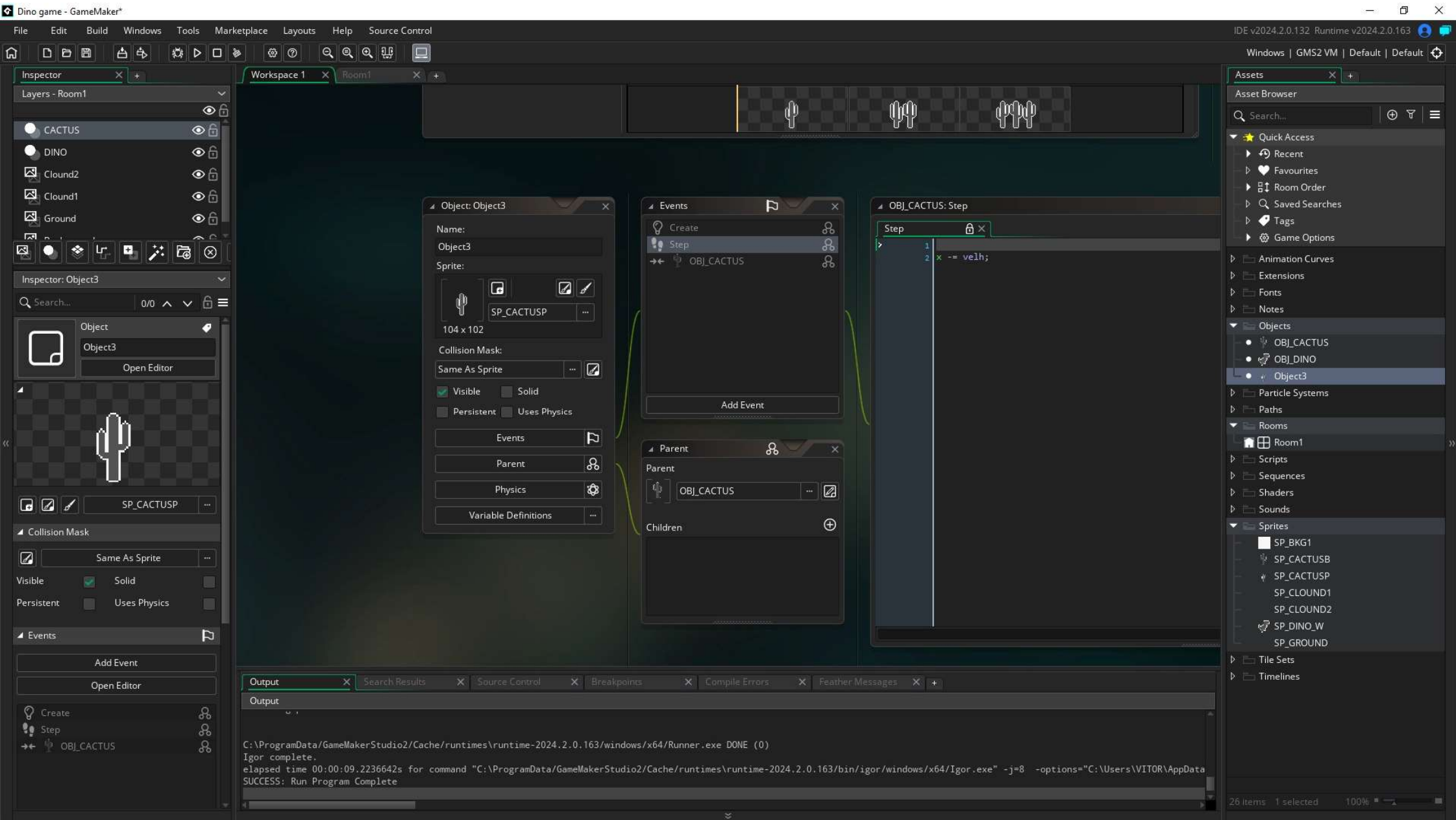
Em seguida, iremos criar um objeto para o sprite do cactus, com os eventos de create, step e collision. Em create, adicionaremos a função de randomizar o spawn do sprite e modelo de cactus e sua velocidade em velh (velocidade horizontal) e uma condição para destruição dos sprites após saírem da tela.

```
image_index = irandom(image_number - 1)  
velh = 7
```

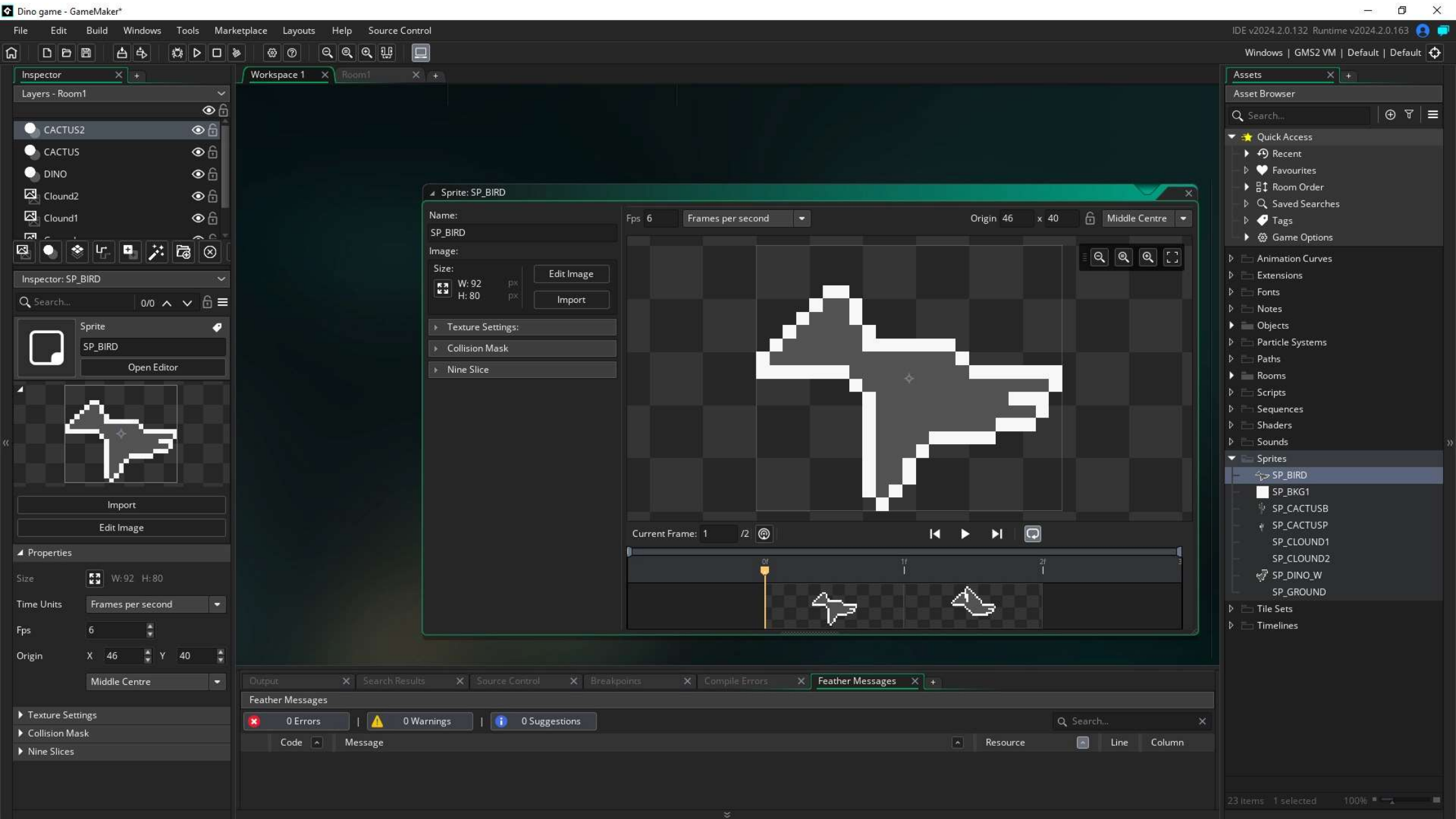
Em **step**, iremos organizar sua movimentação em -x
x-= velh; if (x <= -64) {

```
instance_destroy();  
}
```

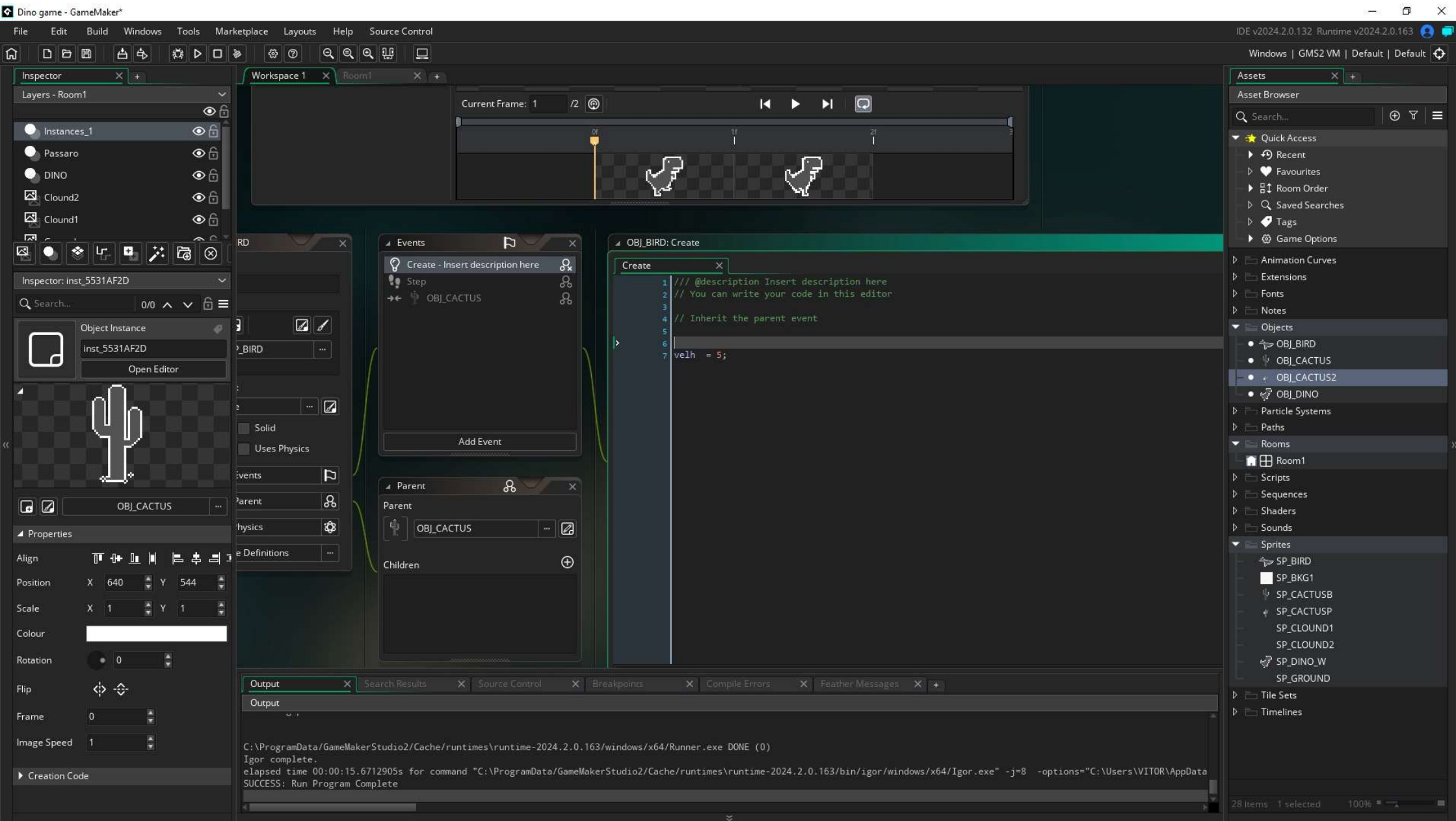
Agora iremos adicionar um sprite para os cactus menores, seguindo as mesmas configurações dos maiores, porém para não repetirmos toda a programação, iremos adicionar uma relação de parentesco entre o cactus já criado



Feito isso, iremos adicionar os obstáculos aéreos, no caso, os pássaros. Da mesma forma anterior, iremos selecionar e adicionar os sprites para os pássaros a partir da pasta. Após nomeá-lo, vamos definir sua origem para o centro do frame e definir o fps da animação para 6 fps. Precisamos também arrumar a hitbox para o frame e podemos fazer isso indo em collision mask, definindo o modo para manual e então achatando a hitbox

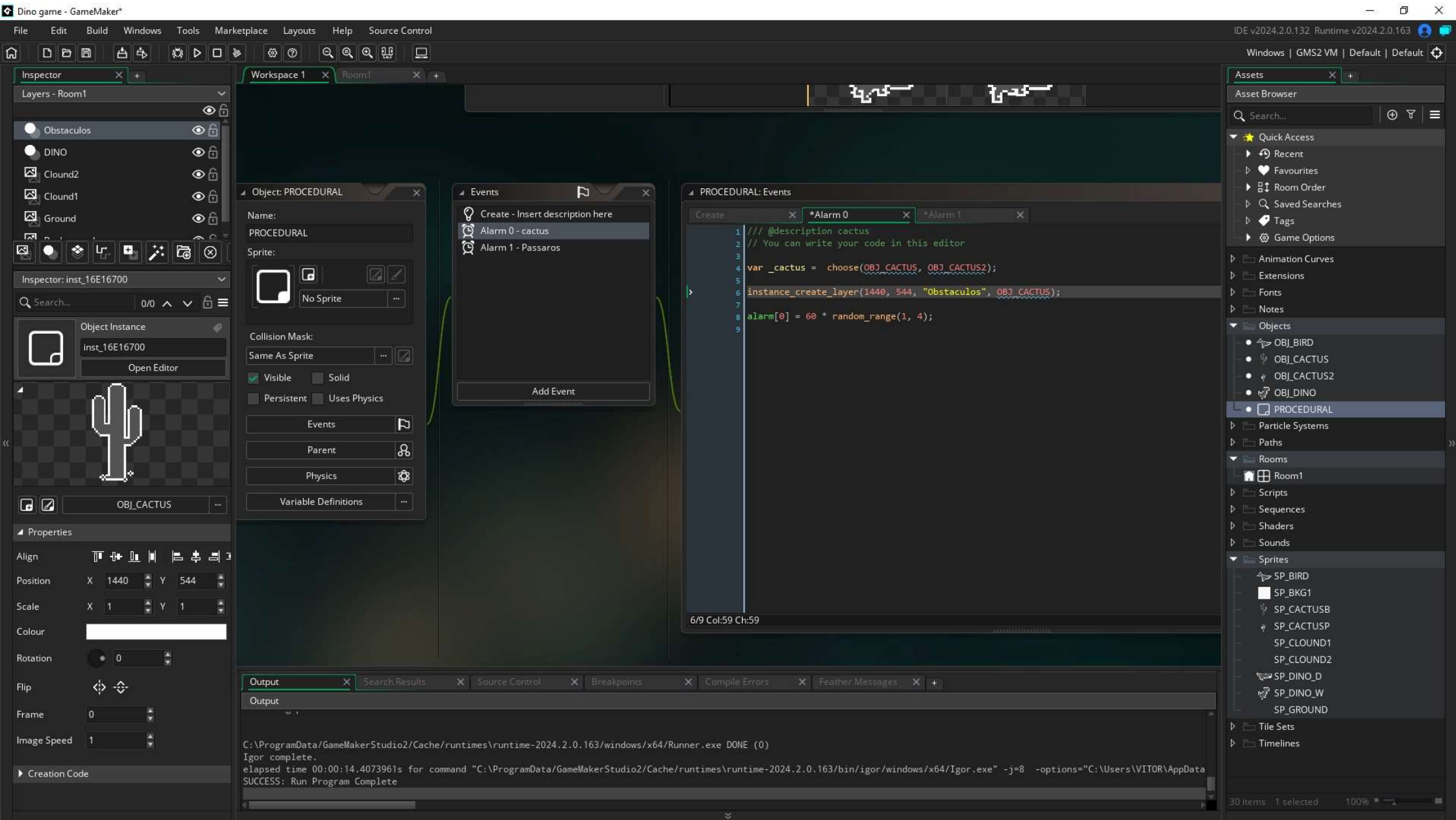


Agora, criaremos um novo material e adicionaremos o alvo como o sprite do passaro. Após isso, iremos adicionar o cactus como parente para o passaro, porém com algumas diferenças. No create, vamos clicar com o direito e escolher Herdar o código diretamente do parente, e iremos adicionar uma variável de velocidade horizontal para ele.



Agora podemos configurar a criação automática de obstáculos. Primeiramente, criamos um objeto novo, podemos nomeá-lo de procedural. Dentro deste objeto iremos criar um evento chamado alarme. O alarme será responsável por criar, em um determinado espaço de tempo, um determinado objeto.

Neste gerador é criada um processo de criação de uma instância em uma layer, pegando a **posição x, y, nome da layer e nome do objeto** e definindo que o alarme vai ser de 60 X um valor randomico entre 1 e 4. Adicionamos tambem a variável que busca um modelo secundário do cactus. Em Create, definir: alarm [0] = 60; alarm [1] = 120;



```
var _cactus = choose(OBJ_CACTUS, OBJ_CACTUS2);
```

```
instance_create_layer(1440, 544, "Obstaculos", _cactus);
```

```
alarm[0] = 60 * random_range(1, 4);
```


Agora criamos mais um alarme, agora para o pássaro. Porém, como não temos variantes, iremos utilizar apenas a função de desenho e alarme.

```
instance_create_layer(1088, 400, "Obstaculos", OBJ_BIRD);
```

```
alarm[1] = 60 * random_range(2, 6);
```

A ideia é criar um sistema de pontuação que permaneça armazenado até o fechamento da janela, mas que durante os restarts do jogo, guarde o maior valor e vá aumentando a medida em que o personagem não triga o evento restart.

Iremos criar um script na pasta de scripts. **Neste script, iremos definir as variáveis globais.** Uma variável global no GameMaker é uma variável que pode ser acessada e modificada a partir de qualquer objeto ou script dentro do jogo. Isso é útil quando você precisa compartilhar informações entre diferentes partes do jogo, como pontuação, configurações de jogo, ou estados que precisam ser consistentemente acessíveis em todo o jogo.

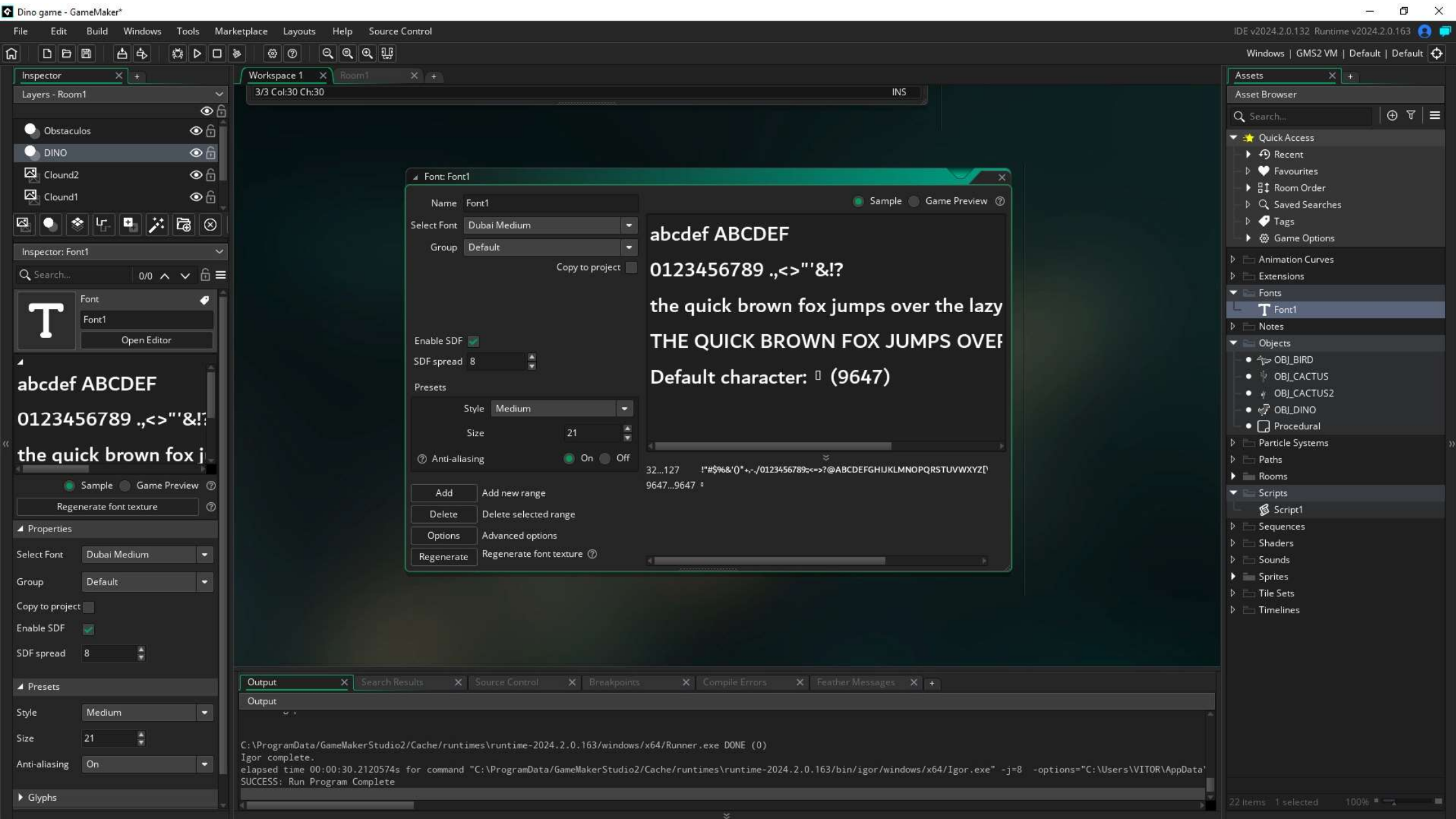
Agora, no script definir as duas variáveis:

```
global.pontos = 0;
```

```
global.pontos_mais_altos = 0;
```

```
global.end = false;
```

Para seleccionarmos a fonte correta, primeiro precisamos criar uma na pasta fonts.
Lá iremos definir qual a fonte, qual o tamanho da fonte, estilo e afins.



Voltamos agora para o objeto “procedural” e adicionaremos um novo evento, chamado Draw GUI, que irá desenhar uma interface do usuário. Nele iremos criar o mostrador da pontuação, definir sua cor e definir a fonte e a posição com o ponto de retirada da informação para a contagem de pontos. Adicionaremos também uma variável para arredondar os pontos

```
draw_set_color(c_black); draw_set_font(Font1); var  
_pontos = "SC: " + string(round(global.pontos));  
draw_text(490, 20, _pontos);
```

```
var _pontos_Altos = "HS: " + string(round(global.pontos_mais_altos));  
draw_text(490, 60, _pontos_Altos);
```

Agora iremos adicionar um evento step no obj procedural.

Neste step iremos definir a velocidade da contagem de pontos:

```
global.pontos += 0.5;
```


precisamos criar novas circunstancias para o restart do game. Podemos criar um novo sprite com o icone de restart. Colocamos seu ponto de referência em midle centre e o fps em 0. Em seguida criamos um objeto com o nome de restart e nele adicionamos a sprite criada. Vamos criar uma nova layer de instanciamento e chama-la de Restart e posicionar o novo objeto no centro da tela e tirar a visualização (icone de olho ao lado da layer)

No novo objeto restart, vamos adicionar um evento de mouse (left released).
Dentro deste evento vamos adicionar:

```
game_restart();
```

```
global.pontos = 0;
```

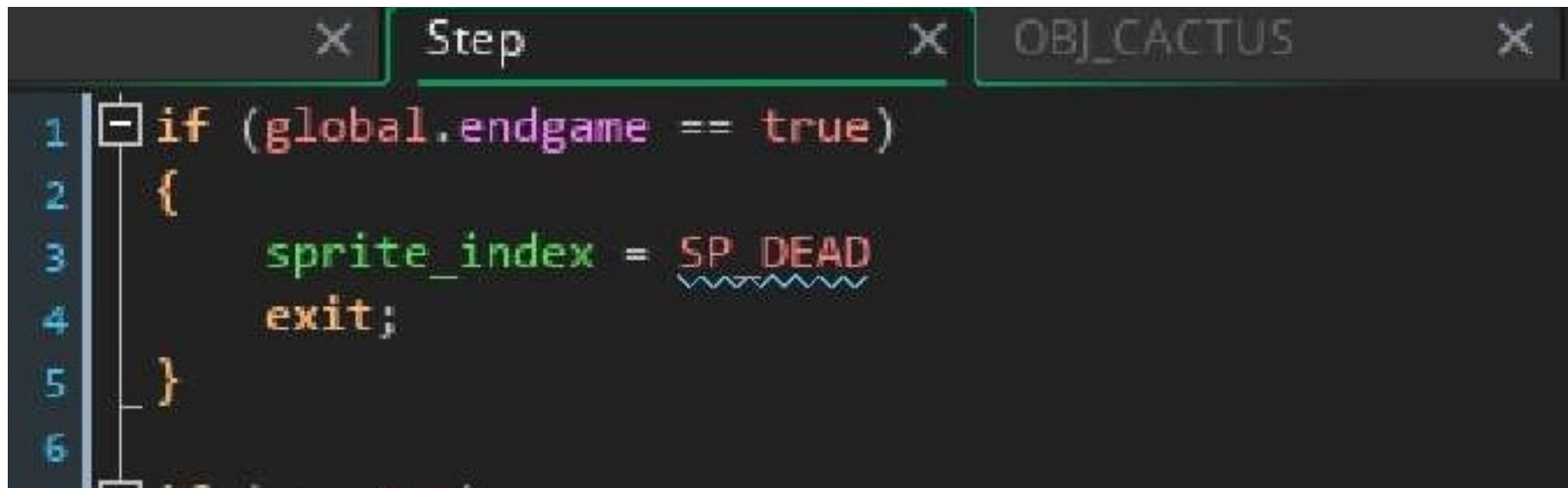
```
global.endgame = false;
```

No objeto Dino, no evento de colisão do cactus, adicionar:



```
1 if (global.endgame == false)
2 {
3     global.endgame = true;
4
5     layer_set_visible("Restart", true);
6
7     layer_hspeed("Ground", 0);
8     layer_hspeed("Clound1", 0);
9     layer_hspeed("Clound2", 0);
10
11
12 if (global.pontos > global.pontos_mais_altos)
13 {
14     global.pontos_mais_altos = global.pontos;
15 }
16 }
```

No objeto Dino, no evento de step, adicionar antes do código anterior:



```
1 if (global.endgame == true)
2 {
3     sprite_index = SP_DEAD
4     exit;
5 }
6
```