CARTILHA CLOC

CLOC - JOGOS DIGITAIS

Aracaju: THP, 1ª edição, 2023.

Ed-mundo

CLOC

LÓGICA CRESCIMENTO CRIATIVIDADE OPORTUNIDADE

COORDENAÇÃO DO PROJETO E AUTORIA Renata Santos Celestino (CLOC)

COORDENAÇÃO DE TS Raí Thales da Silva Gomes

CO-AUTOR José Adelmo Teodózio Santos Júnior

PROJETO GRÁFICO Jéssica Oliveira Sá
Vítor Bahia

REVISÃO Mariana Carolina de Almeida Souza

Realização: The Human Project

Patrocínio: CMDCA Conselho Municipal dos Direitos da Criança e do Adolescente de Santa Luzia do Itanhi · FMDCA

Apoio: elo · SIEMENS | Fundação · kapitalo investimentos

CLOC - Jogos Digitais

Sumário

< história >

01

Aula 01
- Módulo história dos jogos

Passo 1: Contextualização/Engajar

A história dos jogos digitais teve início quando os acadêmicos começaram a projetar jogos simples, simuladores e programas de inteligência artificial, como parte de suas pesquisas em ciência da computação. Somente a partir das décadas de 1970 e 1980 é que os jogos eletrônicos se tornaram populares, quando jogos de arcade, console de jogos eletrônicos e jogos de computador foram introduzidos ao público em geral. Desde então, os jogos eletrônicos tornaram-se uma forma popular de entretenimento e uma parte da cultura moderna em diversas regiões do mundo.

Os jogos foram desenvolvidos para ampliar a mente e desenvolver melhor o cérebro nas atividades escolares, como uma consequência das pesquisas da computação em áreas como a inteligência artificial. A comercialização do UNIVAC I (considerado o primeiro computador comercial da história) em 1951 abriu caminho para a adoção dos computadores por instituições acadêmicas, órgãos de pesquisa e empresas em todo o mundo desenvolvido. Devido ao alto custo, grande consumo de energia e a necessidade de se empregar uma equipe altamente treinada para manter e operar as máquinas, a tecnologia da computação ficou inicialmente limitada às organizações maiores. Por conta disso, a criação dos primeiros jogos eletrônicos limitou-se a testes e demonstrações de teorias relacionadas a

TEMA:

Surgimento de alguns Jogos

> Duração:

2h

> Materiais necessários:

Notebooks, projetor, Internet, caderno e lápis.

> Objetivo:

Mostrar um pouco da história dos jogos trazendo assim uma introdução do que será visto nos próximos conteúdos.

áreas como a interação humano-computador, a aprendizagem adaptativa e estratégia militar.

Devido à falta da documentação de muitos desses testes, é difícil de se determinar qual teria sido o primeiro jogo eletrônico criado. Alguns dos primeiros jogos conhecidos incluem Nimrod (1951), uma máquina feita sob encomenda pela Ferranti para o Festival da Grã-Bretanha e na qual se poderia jogar o jogo matemático Nim; OXO (1952), criado por Alexander S. Douglas para o computador EDSAC e que simulava o jogo da velha; e Hutspiel (1955), um jogo de guerra construído pelo exército dos Estados Unidos para simular um conflito com a União Soviética na Europa.

Um jogo que se destaca nesse período inicial (e também por futuramente ter sido objeto de várias disputas judiciais de patente) é Tennis for Two (1958), criado pelo físico norte-americano William Higinbotham[1] para entreter os convidados no dia da visita anual realizada pelo Laboratório Nacional de Brookhaven. Este programa simulava uma partida de tênis exibida na tela de um osciloscópio. Um ponto piscando representava a bola e os jogadores controlavam seu movimento por cima de uma linha vertical que representava a rede. Não havia na imagem a representação dos jogadores, apenas da 'bola' e da 'quadra' de tênis, numa vista lateral.

Passo 2: Atividade de Pesquisa e Fixação

Pedir para os alunos usarem os notebooks para pesquisar a história de algum jogo que tenha a criação antiga.

Separar 15 minutos para esse momento e pedir para fazer anotações no caderno.

Passo 3: Desafio/ compartilhar

Cada aluno irá pegar suas anotações e apresentar a história para os demais alunos.

< requisitos >

02

Aula 02
- Módulo requisitos do jogo

Passo 1: Contextualização/Engajar

TEMA:

Quais os requesitos para a criação de um jogo.

> Duração:

2h

> Materiais necessários:

Notebooks, projetor, Internet, caderno e lápis.

> Objetivo:

Mostrar qual a utilidade/ função de cada requisito.

Os requisitos para fazer um jogo podem variar dependendo do tipo de jogo e da plataforma em que você pretende lançá-lo. No entanto, aqui estão alguns requisitos comuns para a criação de um jogo:

Conceito e design do jogo: Antes de começar a desenvolver um jogo, é importante ter uma ideia clara do conceito e do design do jogo. Isso envolve definir o gênero, a história, os personagens, a jogabilidade, os objetivos e os desafios do jogo.

Plataforma de desenvolvimento: Você precisará escolher uma plataforma de desenvolvimento adequada ao seu projeto. Isso pode incluir engines de jogos, como a Unity, Unreal Engine ou GameMaker, ou até mesmo ferramentas de desenvolvimento específicas para consoles.

Habilidades de programação: Se você está criando um jogo que requer programação personalizada, é necessário ter habilidades de programação relevantes. Isso pode envolver linguagens de programação como C++, C#, Python ou JavaScript, dependendo da plataforma de desenvolvimento escolhida.

Arte e design gráfico: Os jogos exigem recursos visuais, como sprites, modelos 3D, animações, cenários e efeitos visuais. É importante ter habilidades em design gráfico, modelagem 3D, animação ou trabalhar com artistas para criar os elementos visuais do seu jogo.

Som e música: O áudio desempenha um papel crucial nos jogos. Você precisará de habilidades de design de som e música, ou colaborar com profissionais de áudio, para criar efeitos sonoros imersivos, trilhas sonoras e diálogos, se necessário.

Testes e refinamento: À medida que você desenvolve seu jogo, é importante realizar testes frequentes para identificar bugs, problemas de jogabilidade e otimizações necessárias. É essencial realizar interações e refinamentos para aprimorar a experiência do jogo.

Passo 2: Atividade de Pesquisa e Fixação

Pedir para os alunos pesquisarem qual design eles vão querer abordar/desenvolver, qual a plataforma que será utilizada e que tipo de som vai ser inserido.

Separar 15 minutos para esse momento e pedir para fazer anotações no caderno.

Passo 3: Desafio

Fazer uma lista de tudo que achou interessante na pesquisa e ter como base essa pesquisa para começar a desenvolver.

Lembrando que devemos direcionar os alunos com as pesquisas e as anotações.

< Diferenças entre jogos >

03

Aula 03
- Diferenças entre jogos

Passo 1: Contextualização/Engajar

Diferenças principais entre jogo eletrônico e jogo de tabuleiro

Por mais que seus estilos até mesmo busquem inspirações um no outro, os jogos eletrônicos comumente possuem a necessidade de aparelhos eletrônicos focados para o funcionamento, como televisores e consoles diferenciando da necessidade e jogo específico.

Já ao contrário de jogos de tabuleiro onde muitas vezes é necessário apenas uma superfície física plana onde o jogo acontece, e em alguns casos, será necessário dados que ao rolarem, determinam os resultados e/ou peças para movimentar.

O que são jogos eletrônicos?

Jogos eletrônicos ou videogame são jogos que acontecem em aparelhos eletrônicos, tais como computadores, aparelhos celulares, ou até mesmo fliperamas com jogos específicos.

Tendo sua origem muito mais moderna que no outro caso, datado no começo dos anos 60 como foi o caso de "Tennis for Two", considerado por muitos como um dos primeiros jogos eletrônicos criados.

E de lá para cá, obteve um grande avanço de mercado de jogos, incluindo a evolução gráfica e de jogabilidade. Não podemos esquecer de citar também a inclusão da inteligência

TEMA:

As diferenças entre jogos

> Duração:

2h

> Materiais necessários:

Notebooks, projetor, Internet, caderno e lápis.

> Objetivo:

Mostrar as principais diferenças entre jogos eletrônicos e jogos de tabuleiro.

artificial nos jogos, deixando os games ainda mais realistas.

Tipos de jogos eletrônicos e exemplos

Graças a tecnologia por trás dos jogos eletrônicos, uma enorme variedade de games são produzidos todos os anos.

Alguns deles são focados em habilidades cognitivas, como raciocínio lógico. Mas em muitos outros casos, os jogos mesmo não focando nisso, acabam abrangendo esses estímulos psicológicos.

Alguns exemplos são:

> **RPG** – Roleplaying game ou jogo de interpretação, como Final Fantasy.

> **FPS** – First Person Shooter ou jogo de tiro em primeira pessoa, como Call of Duty.

> **Simuladores** – Jogos focados em simular situações, desde ser um prefeito construindo uma cidade até ser um caminhoneiro, como em Euro Truck Simulator.

> **Esportes** – Jogo de controle de equipes esportivas, como FIFA.

> **Puzzle** – Jogos focados em quebra cabeça, e raciocínio, como em Portal 2.

> **Party Games** – Jogos focados em múltiplos jogadores participando em diversos eventos, tal como em Mario Party.

> **Ação** – Jogos de luta, como Mortal Kombat.

Entre esses e muitos outros estilos de jogos, com histórias mais profundas ou somente com tiros apenas pela satisfação de derrotar hordas de inimigos.

O que são jogos de tabuleiro?

Os jogos de tabuleiro são os jogos que induzem a criatividade do jogador, normalmente realizados em uma superfície plana, com uma coleção de regras para cada jogo ou atividade durante as partidas normalmente buscando realizar um objetivo para garantir a vitória.

Focando muito mais em criatividade, os jogos de tabuleiro tem sua origem muito mais antiga que jogos eletrônicos, datando de séculos atrás, em civilizações antigas. Jogos mais novos foram surgindo e aumentando a popularidade, como em 1944 com o surgimento do Banco Imobiliário.

Métricas e tipos de jogos de tabuleiro

Desde as regras mais complexas ou até as mais simples, a variedade de jogos de tabuleiro não deixa nada a desejar. Alternando entre focadas em divertimento ou a melhorar pensamentos estratégicos e leitura comportamental para descobrir quem é quem no jogo.

Possuindo tamanha variedade, é possível escolher sua própria preferência em questão de gosto.

Alguns exemplos são:

> **Jogos de guerra -** normalmente com objetivo de conquistar o tabuleiro, como em Avalon hill axis and allies.

> **Jogos cooperativos -** contando com a ajuda dos outros jogadores, como em Pandemic.

> **Hidden Traitor (Descobrir ou enganar seus amigos)-** como em Hitler secreto ou the resistance.

> **Jogos educacionais -** focando em aprendizado, como Racha-Cuca.

Seja pela estratégia ou pela companhia durante os jogos, jogos de tabuleiro não ficam pra trás em conteúdo em uma era moderna.

Passo 2: Atividade de Pesquisa e Fixação

Pedir para os alunos pesquisarem a história de um jogo eletrônico, uma história de um jogo de tabuleiro e fazer anotações.

Ideias para pesquisas: Qual o objetivo do jogo; Quais as regras; Qual o gênero; Quando surgiu; Quem criou.

Passo 3: Desafio/ compartilhar

Pedir para os alunos apresentarem suas pesquisas explicando cada tópico, e fazer algumas perguntas simples no momento da explicação do aluno só para testar se os alunos realmente estão compreendendo de fato as pesquisas.

< características >

04

Aula 04 - Características

TEMA:

Características principais para um jogo

> Duração:

2h

> Materiais necessários:

Notebooks, projetor, Internet, caderno e lápis.

> Objetivo:

Criar um roteiro com as principais características para desenvolver um jogo.

Passo 1: Contextualização/Engajar

Você tem que pensar em alguns detalhes antes de começar a desenvolver o jogo em si. Veja exemplos básicos:

Os alunos precisam ter escrito esses tópicos para ter como base, poder desenvolver o jogo e tentar responder.

1. Quais são os seus pontos fortes enquanto designer?
2. E os pontos fracos?
3. Que tipo de jogo você é capaz de desenvolver com as suas habilidades atuais?
4. A que gênero o jogo vai pertencer?
5. Que mecânicas de jogos que já existem você quer usar de base?
6. E quais delas você considera frustrantes?

7. Quais jogos servem de inspiração para você?

8. Como o seu jogo vai ser diferente de tantos outros?

9. Quais vão ser as principais características do jogo?

10. Quanto tempo você vai levar para desenvolver esses detalhes?

11. Há algum detalhe que você não queira incluir?

12. Quais deles são dispensáveis, caso seja necessário abrir mão de algo?

13. O jogo tem uma história?

14. Qual é a relação entre a jogabilidade e a história?

15. Que estilo artístico você quer colocar no jogo?

16. Como você pode criar esse estilo?

Passo 2: Atividade de Pesquisa e Fixação

Quando os alunos começarem a responder vão existir algumas dúvidas, podendo usar o computador para pesquisar.

Essa lista não será preenchida em um dia ou pode ser mudada com o decorrer das aulas. O objetivo é fazer com que os alunos tenham essa lista como base e seguir.

< revisão >

05

Aula 05 - HTML

Passo 1: Revisão

Revisão e passo a passo de códigos de HTML.

Acesse o QR Code para mais conteúdo

Pode haver algumas dúvidas, é importante deixar um espaço para os alunos pesquisarem.

TEMA:

HTML

> Duração:

2h

> Materiais necessários:

Notebooks, projetor e Internet.

> Objetivo:

Relembrar/ensinar um pouco de programação.

< jogos programáveis e no-code >

06

Aula 06
- Programáveis e no-code

Passo 1: Contextualização/Engajar

A diferença entre jogos programáveis e jogos no-code reside principalmente no processo de desenvolvimento e nas habilidades necessárias para criar um jogo.

Jogos Programáveis: Os jogos programáveis referem-se a jogos em que o desenvolvedor precisa escrever código para criar a lógica do jogo, definir comportamentos, implementar mecânicas e lidar com aspectos técnicos. Isso geralmente envolve o uso de linguagens de programação, como C++, C#, Python, JavaScript, entre outras. O desenvolvedor precisa ter conhecimento de programação e lógica de programação para criar o jogo. Esse tipo de desenvolvimento oferece maior flexibilidade e controle, permitindo a criação de recursos personalizados e a implementação de lógica complexa.

Jogos No-Code: Jogos no-code, por outro lado, são jogos criados sem a necessidade de escrever código manualmente. Essas plataformas de desenvolvimento no-code fornecem ferramentas visuais e interfaces intuitivas que permitem aos usuários criarem jogos através de uma abordagem mais "arrastar e soltar". As pessoas sem experiência em programação ainda podem criar jogos usando essas plataformas, pois elas geralmente oferecem uma biblioteca de recursos pré-fabricados, scripts pré-definidos e uma interface visual para

TEMA:

Jogos programáveis e no-code

> Duração:

2h

> Materiais necessários:

Notebooks, projetor, Internet, caderno e lápis.

> Objetivo:

Mostrar as principais diferenças entre jogos programáveis e jogos no-code.

definir lógica e comportamentos. Embora possa haver opções limitadas em comparação com a programação manual, as plataformas no-code são mais acessíveis e permitem que pessoas sem experiência em programação possam criar jogos funcionais de forma mais rápida e fácil.

Passo 2: Atividade de Pesquisa e Fixação

Pedir para os alunos pesquisarem 3 jogos programáveis e 3 jogos no-code.

> Separar 20 minutos para os alunos pesquisarem.

Passo 3: Desafio/ compartilhar

Pedir para os alunos escolherem um jogo de cada e compartilhar com os demais.

> Lembrando que pode acontecer dos alunos fazerem pesquisas parecidas, isso não é problema, cada aluno tem seu jeito de compartilhar.

< Hora do Código >

07

Aula 07
– A Hora do Código

Passo 1: Contextualização/Engajar

TEMA:

A Hora do Código

> Duração:

2h

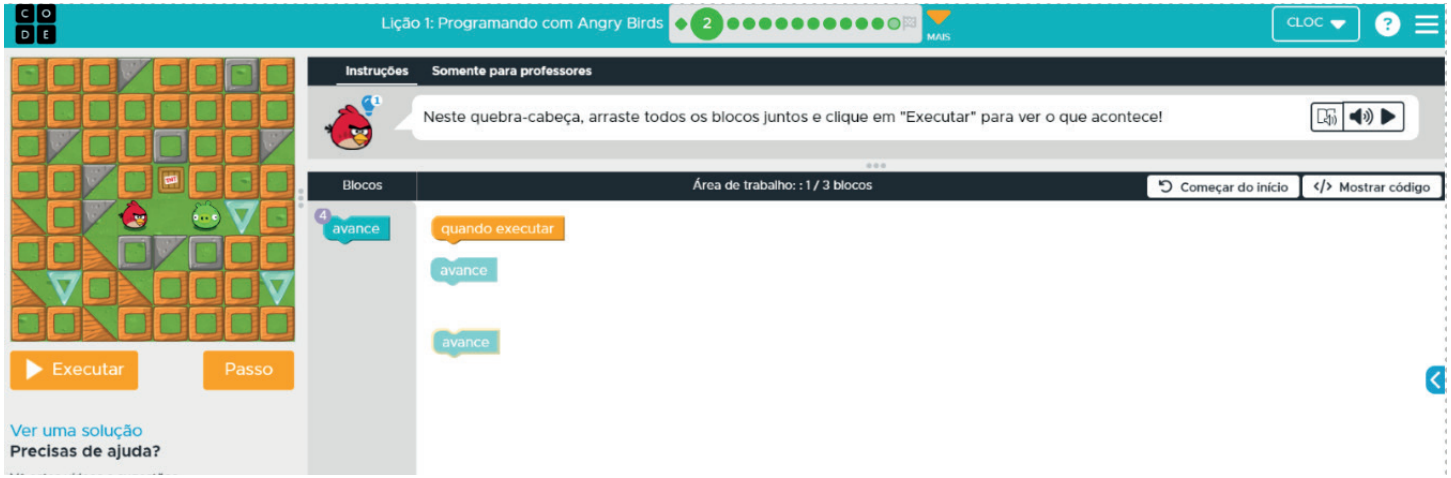
> Materiais necessários:

Notebooks, projetor e Internet.

> Objetivo:

Mostrar um pouco de lógica em forma de jogo de raciocínio.

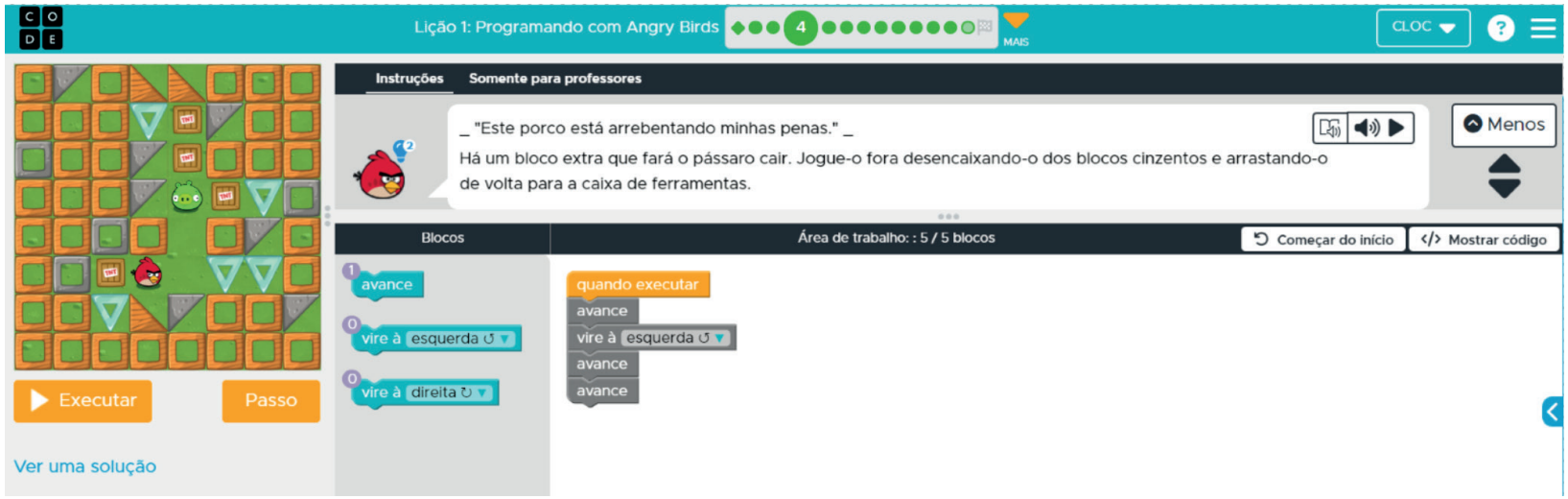
Como os alunos não conhecem a plataforma é importante explicar como funciona e qual o objetivo, pode fazer o primeiro desafio para simplificar a explicação.



É um formato de quebra-cabeça, com as funções de arrastar e colar bem simples, o objetivo é fazer com que o passarinho da cor vermelha encontre o porquinho da cor verde.

Passo a passo

Conectando 2 “avance” em baixo do bloco “quando executar”, como na imagem, para assim adquirir o resultado.



E cada fase concluída os desafios começam a ficar difíceis, como o exemplo da imagem acima do nível 4, onde já começa a aparecer novos blocos, como virar para direita e esquerda, mas segue com o mesmo objetivo: o passarinho chegar até o porquinho.

É importante deixar os alunos pensarem um pouco e cada bloco a ser colocado, já ir testando para assim conseguir identificar o erro ou o que falta a ser acrescentado.

Link da plataforma https://code.org/

< criando um jogo >

08

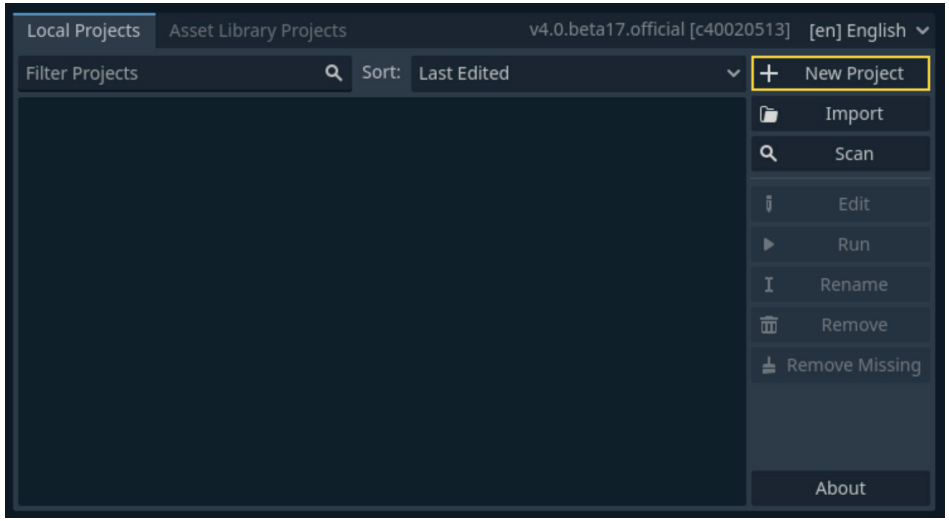
Aula 08
- Desenvolvendo um jogo

Passo 1: Contextualização/Engajar

A primeira plataforma apresentada será **Godot**.

Configurando o projeto

Nesta primeira parte vamos configurar e organizar o projeto. Inicie o Godot e crie um novo projeto.



TEMA:

Configurando e organizando o projeto

> Duração:

2h

> Materiais necessários:

Notebooks, projetor, Internet, caderno e lápis.

> Objetivo:

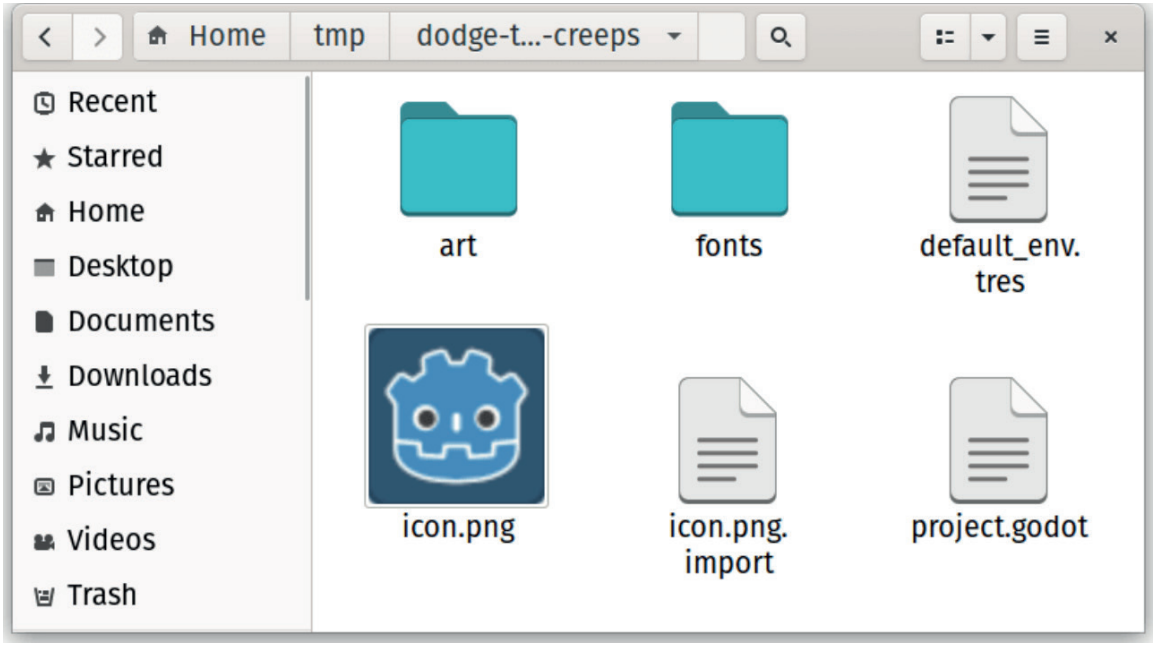
Mostrar a plataforma Godot e as primeiras configurações.

Ao criar o novo projeto, você só precisa escolher um Caminho de Projeto válido. Você pode deixar as outras configurações padrão de lado.

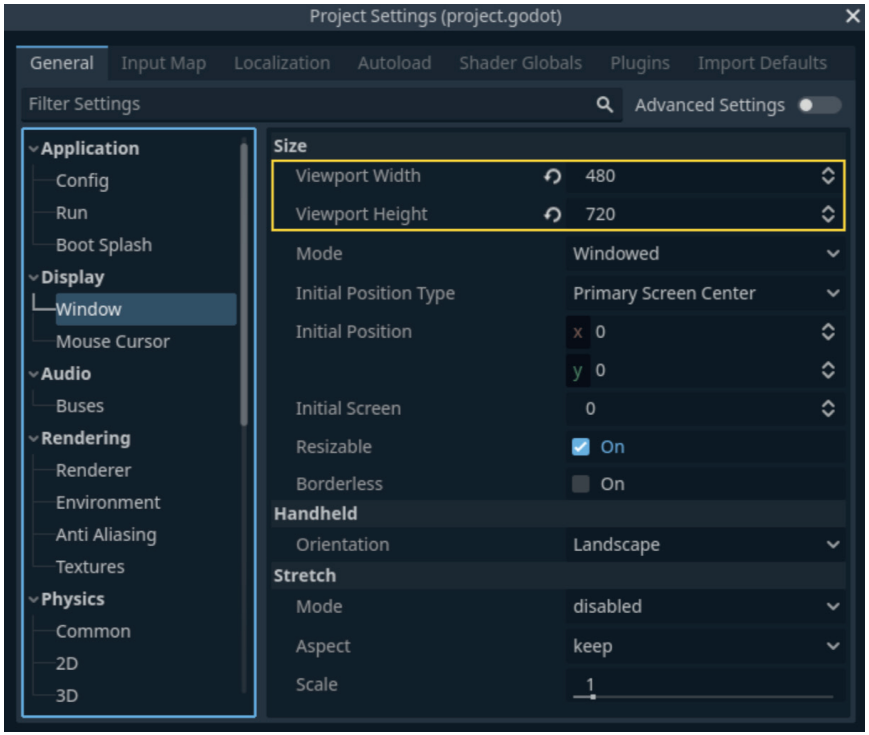
GDScriptC#C++

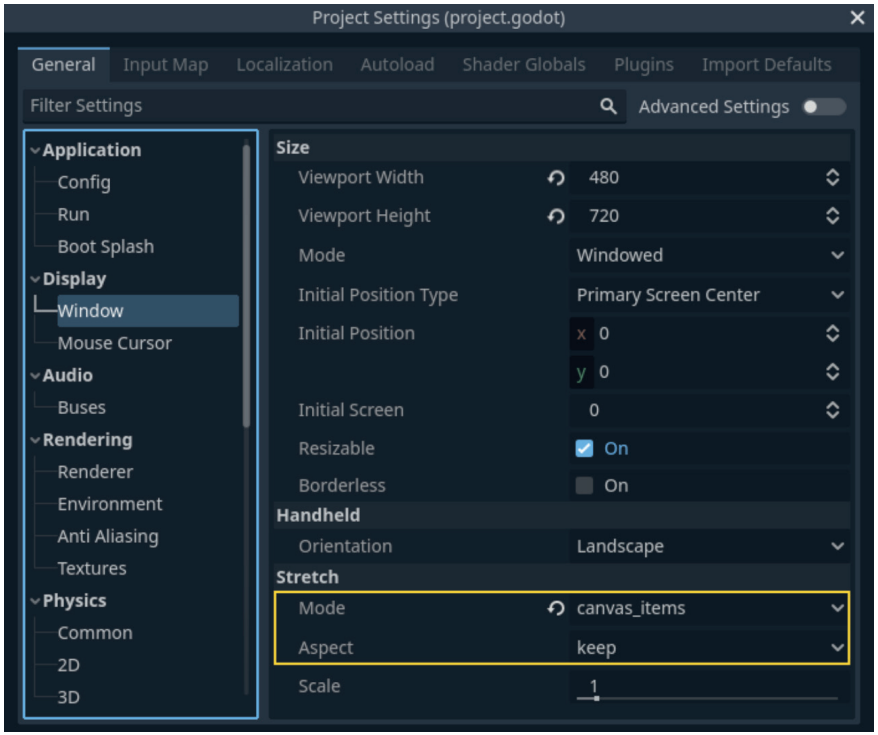
Baixe dodge_the_creeps_2d_assets.zip. O arquivo contém as imagens e sons que você usará para fazer o jogo. Extraia o arquivo e mova os diretórios art/ e fonts/ para o diretório do seu projeto.

Sua pasta de projeto deve se parecer com isto.



Este jogo foi projetado para o modo retrato, então precisamos ajustar o tamanho da janela do jogo. Clique em Projeto -> Configurações do Projeto para abrir a janela de configurações do projeto e na coluna da esquerda, abra a guia Exibir -> Janela. Lá, defina Largura da viewport para 480 e Altura da viewport para 720.



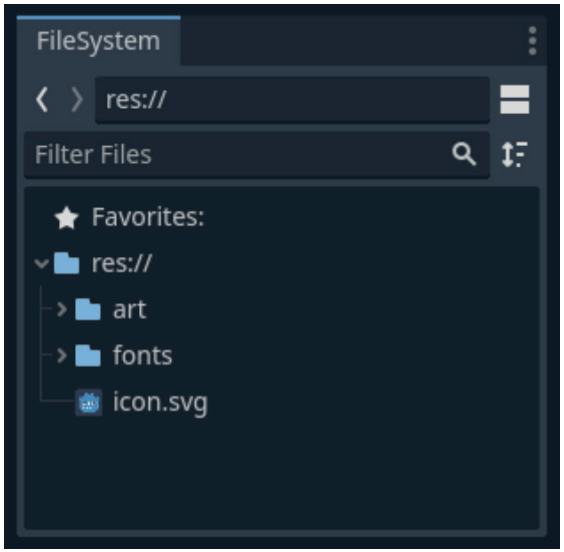


Além disso, nas opções de Esticar, defina Mode como canvas_items e Aspect to keep. Isso garante que o jogo seja dimensionado de forma consistente em telas de tamanhos diferentes.

Organizando o projeto

Neste projeto, faremos 3 cenas independentes: Player, Mob e HUD, que combinaremos dentro da cena Main do jogo.

Em um projeto maior, pode ser útil criar pastas para armazenar as várias cenas e seus scripts, mas para este jogo relativamente pequeno, você pode salvar suas cenas e scripts na pasta raiz do projeto, referenciada como res://. Você pode ver as pastas do seu projeto no painel Arquivos no canto inferior esquerdo.



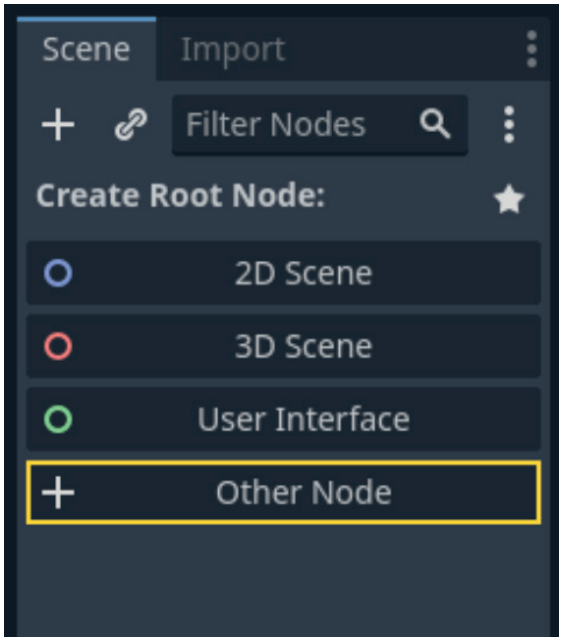
Com o projeto em seu devido lugar, estamos prontos para conceber a cena do jogador na próxima lição.

Criando a cena do jogador

Com as configurações do projeto feitas, podemos começar a trabalhar no personagem controlado pelo jogador.

A primeira cena definirá o objeto Player. Um dos benefícios de criar uma cena Player separada é que podemos testá-la separadamente, mesmo antes de criarmos outras partes do jogo.

Estrutura de nós

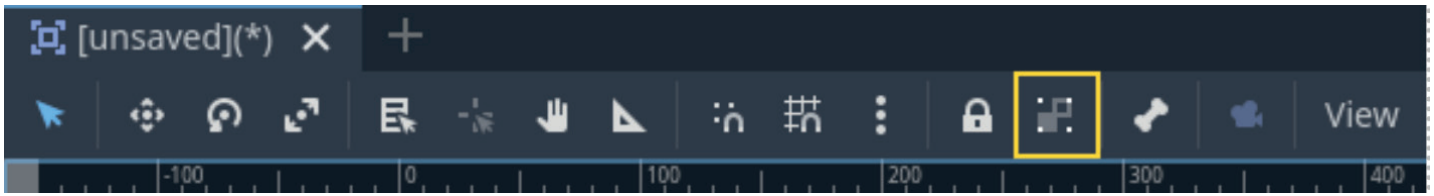


Para iniciarmos, precisamos escolher o nó raiz para o objeto jogador. Como regra geral, o nó raiz da cena deve refletir a funcionalidade desejado do objeto: o que o objeto é. Clique no botão “Outros Nós” e adicione um nó Area2D à cena.

Godot vai mostrar um ícone de aviso próximo do nó na árvore de cenas. Você pode ignorá-lo por enquanto; vamos falar disso mais tarde.

Com a Area2D, nós podemos detectar objetos que se sobreponham ou vão de encontro ao jogador. Mude seu nome para Player com um clique duplo no nome do nó. Já que nós configuramos o nó raiz, nós agora podemos inserir nós adicionais para adicionar mais funcionalidades.

Antes de adicionarmos qualquer filho ao nó Player, queremos ter certeza de que não os moveremos ou redimensionaremos acidentalmente clicando neles. Selecione o nó e clique no ícone à direita do cadeado. Sua dica de ferramenta.

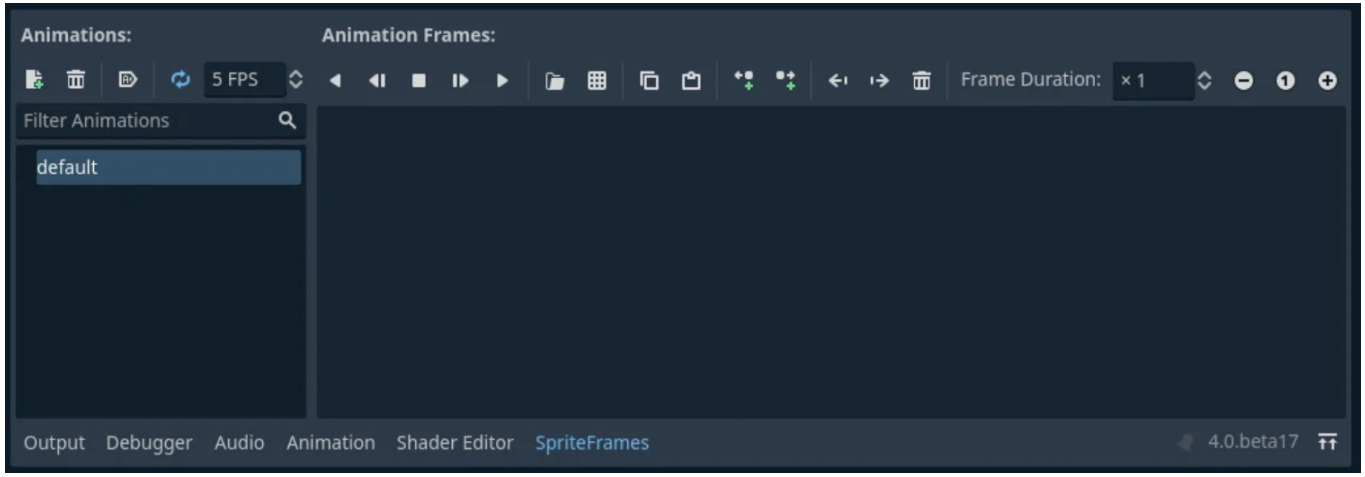


Salve a cena. Clique em Cena -> Salvar ou pressione Ctrl + S no Windows/Linux ou Cmd + S no macOS.

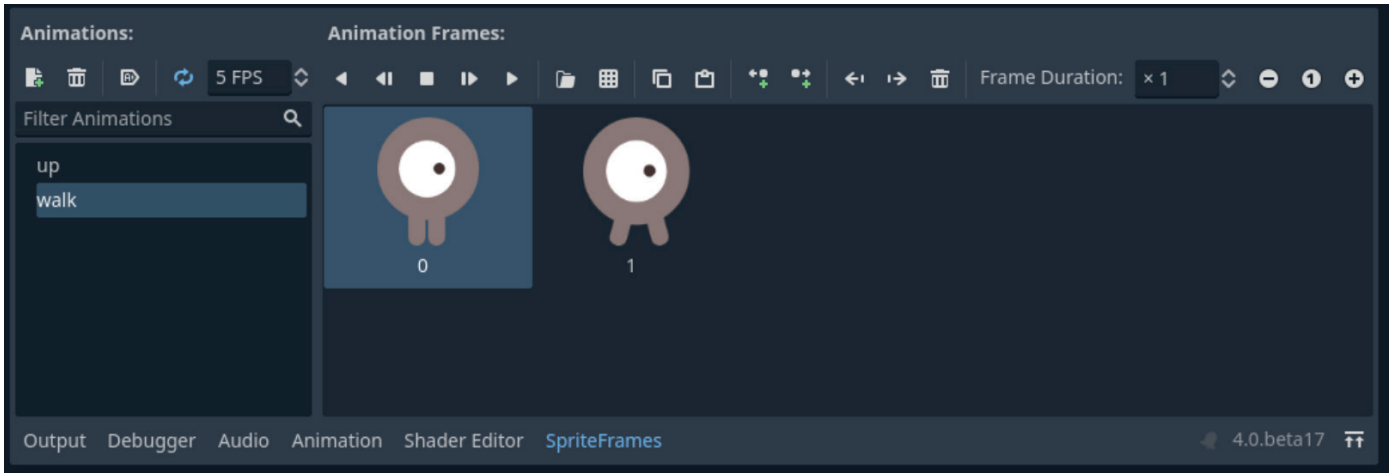
> **NOTA:** Para esse projeto, vamos seguir as convenções de nomeação do Godot.
>
> **GDScript:** Classes (nós) usam o estilo **PascalCase** (IniciaisMaiúsculas), variáveis e funções usam snake_case (minúsculas_separadas_por_sublinha) e constantes usam **ALL_CAPS** (TODAS_MAIÚSCULAS). Para mais conteúdo pesquise na web "Guia de Estilo GDScript".

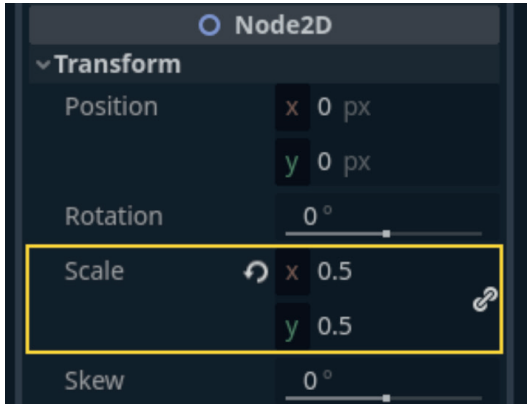
Animação por Sprites

Clique no nó Player e adicione (Ctrl + A) um nó filho AnimatedSprite2D. O AnimatedSprite2D cuidará da aparência e das animações do nosso player. Observe que há um símbolo de aviso próximo ao nó. Um AnimatedSprite2D requer um recurso SpriteFrames, que é uma lista das animações que ele pode exibir. Para criar um, encontre a propriedade Sprite Frames na aba Animation no Inspector e clique em "[empty]" -> "New SpriteFrames". Clique novamente para abrir o painel "SpriteFrames".



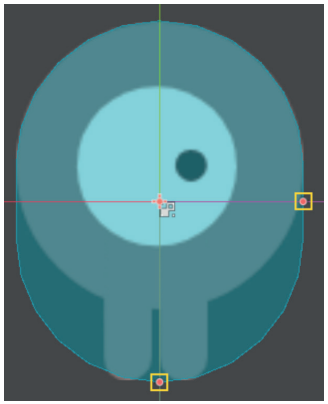
À esquerda está uma lista de animações. Clique no "padrão" e renomei-o para "andar". Em seguida, clique no botão "Adicionar Animação" para criar uma segunda animação chamada "up". Encontre as imagens do player na guia "Sistema de Arquivos" – elas estão na pasta de arte que você descompactou anteriormente. Arraste as duas imagens de cada animação, denominadas playerGrey_up[1/2] e playerGrey_walk[1/2], para o lado "Animation Frames" do painel da animação correspondente.



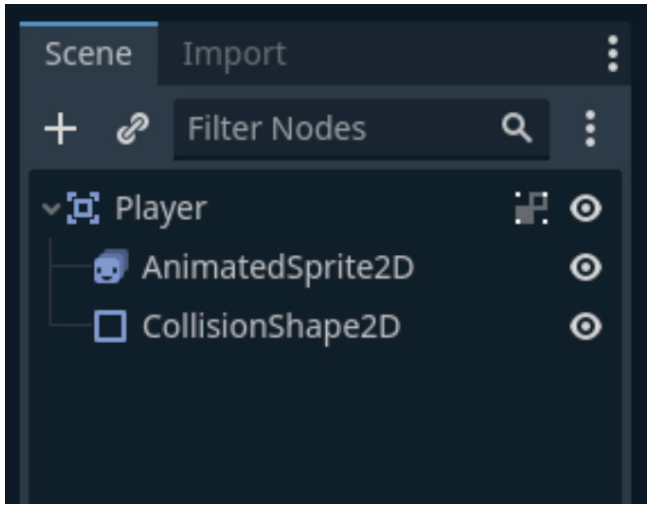


As imagens dos jogadores são um pouco grandes para a janela do jogo, então precisamos reduzi-las. Clique no nó AnimatedSprite2D e defina a propriedade Scale como (0,5, 0,5). Você pode encontrá-lo no Inspetor sob o título Node2D.

Por fim, adicione um CollisionShape2D como filho de Player. Isso determinará a "hitbox" do jogador ou os limites de sua área de colisão. Para este personagem, um nó CapsuleShape2D oferece o melhor ajuste, então próximo a "Forma" no Inspetor, clique em "[vazio]" -> "Novo CapsuleShape2D". Usando as alças de dois tamanhos, redimensione a forma para cobrir o sprite.



Quando tiver finalizado, sua cena Player deveria se parecer conforme imagem ao lado.



Certifique-se de salvar a cena novamente após as alterações.

Na próxima parte, adicionaremos um roteiro ao nó jogador para movê-lo e animá-lo. Depois, configuraremos a detecção de colisão para saber quando o jogador é atingido por alguma coisa.

Passo 2: Atividade de Pesquisa e Fixação

Separar 15 alunos no final para uma pequena pesquisa sobre o Godot, a plataforma tem duas maneiras de desenvolver o jogo que são em 2D e em 3D.

A atividade é pesquisar as diferenças entre o Godot 2D e Godot 3D.

Passo 3: Desafio

Pedir para os alunos anotarem as diferenças, será utilizada para tirar dúvidas com decorrer dos próximos desafios em aula.

< Programando o Jogador >

09

Aula 09
– Programando o jogador

Passo 1: Contextualização/Engajar

TEMA:

Momento de desenvolver o jogador

> Duração:

2h

> Materiais necessários:

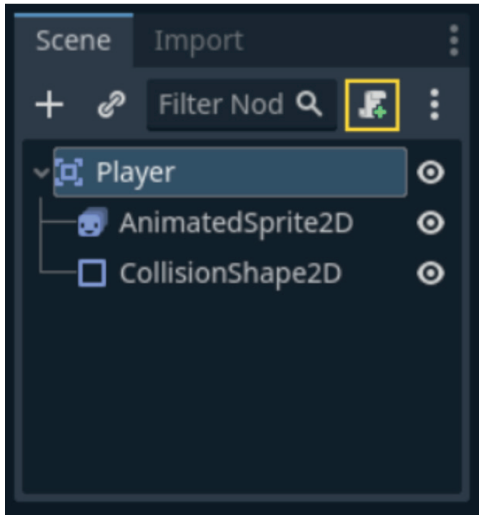
Notebooks, projetor, Internet, caderno e lápis.

> Objetivo:

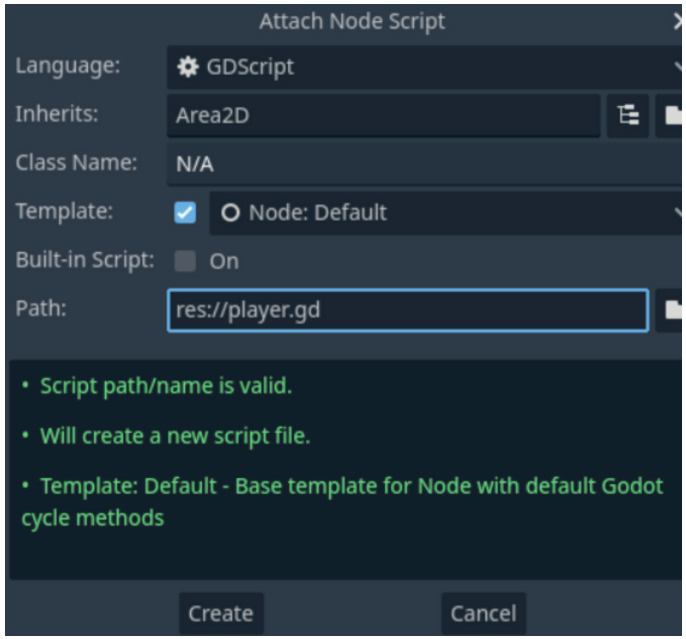
Criar os movimentos do jogador principal.

Nesta lição, adicionaremos movimento e animação do jogador e o configuraremos para detectar colisões.

Para isso, precisamos adicionar algumas funcionalidades que não conseguimos em um nó embutido, então adicionaremos um script. Clique no nó Player e depois clique no botão “Adicionar Script”:

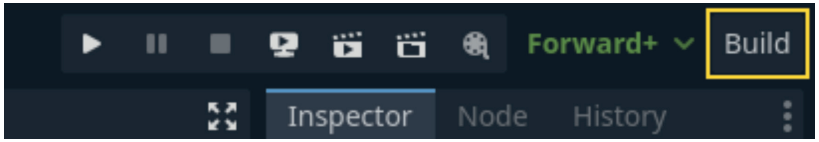


Na janela de configurações do script, você pode deixar as configurações padrão inalteradas. Basta clicar em “Criar”:

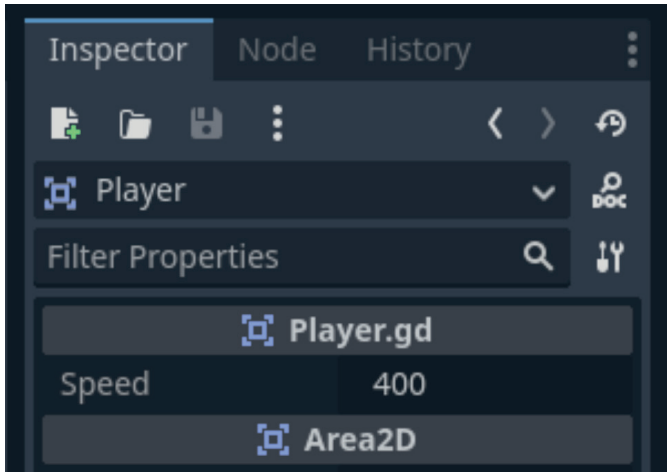


GDScript

Usar a palavra-chave export na primeira variável, speed, nos permite definir seu valor pelo Inspetor. Isto pode ser útil para os valores que você deseja ser capaz de ajustar do mesmo jeito que se faz com as propriedades internas de um nó. Clique no nó Jogador e você verá que a propriedade agora aparece na seção “Variáveis de Script” do Inspetor. Lembre-se, se você modificar o valor aqui, ele irá sobrepor o valor que está escrito no roteiro.



Uma compilação manual também pode ser acionada no Painel MSBuild. Clique na palavra “MSBuild” na parte inferior da janela do editor para revelar o painel MSBuild e clique no botão “Build”.

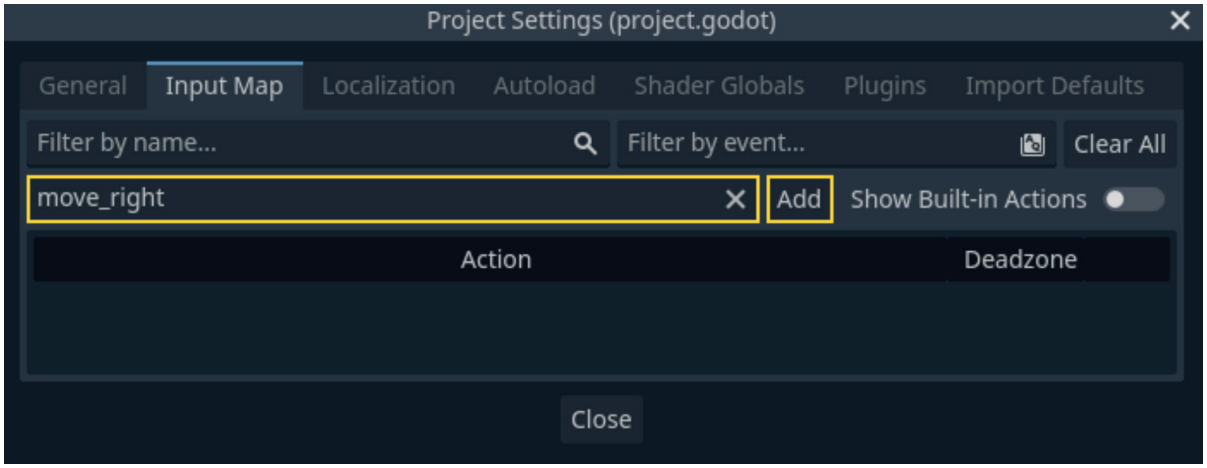


A função _ready() é chamada quando um nó entra na árvore de cena, que é uma boa hora para descobrir o tamanho da janela do jogo:
Agora podemos usar a função _process() para definir o que o jogador fará. _process() é chamada a cada quadro, então usaremos isso para atualizar os elementos de nosso jogo que esperamos que mudem frequentemente. Para o jogador, precisamos fazer o seguinte:

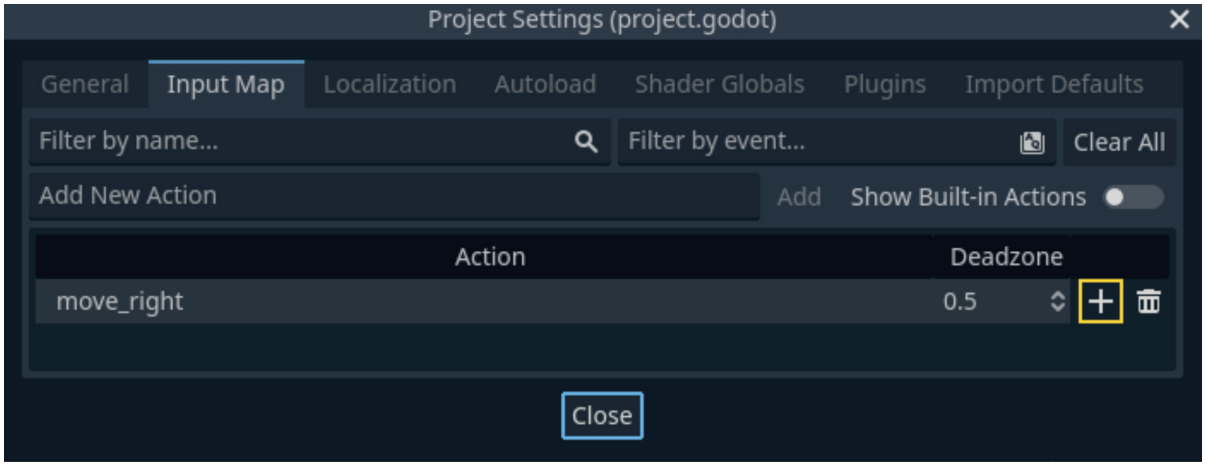
- Verificar as entradas.
- Movimentar na direção desejada.
- Reproduzir a animação apropriada.

Primeiro, precisamos verificar as entradas – o jogador está pressionando uma tecla? Para este jogo, temos 4 entradas de direção para verificar. Ações de entrada são definidas nas Configurações do Projeto na aba “Mapa de entrada”. Nela você pode definir eventos personalizados e atribuir diferentes teclas, eventos de mouse ou outras entradas para eles. Para este jogo, vamos mapear as teclas de seta do teclado para as quatro direções.

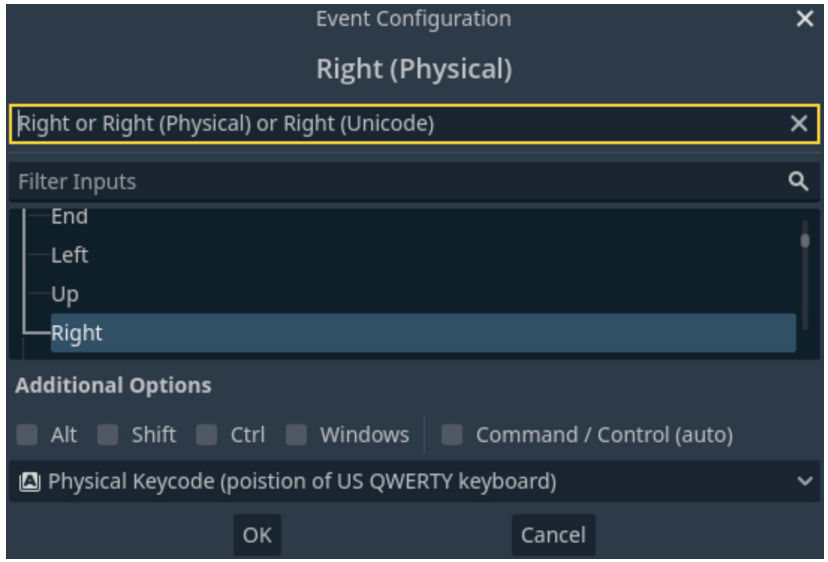
Clique em Projeto -> Configurações do Projeto para abrir a janela das configurações do projeto e clique na aba Mapa de Entrada, no topo. Digite “mover_direita” na barra do topo e clique no botão “Adicionar” para adicionar a ação mover_direita.



Precisamos atribuir uma chave para esta ação. Clique no ícone "+" à direita para abrir a janela do gerenciador de eventos.



O campo "Ouvindo entrada..." deve ser selecionado automaticamente. Pressione a tecla "direita" do teclado e o menu deverá ficar assim agora.

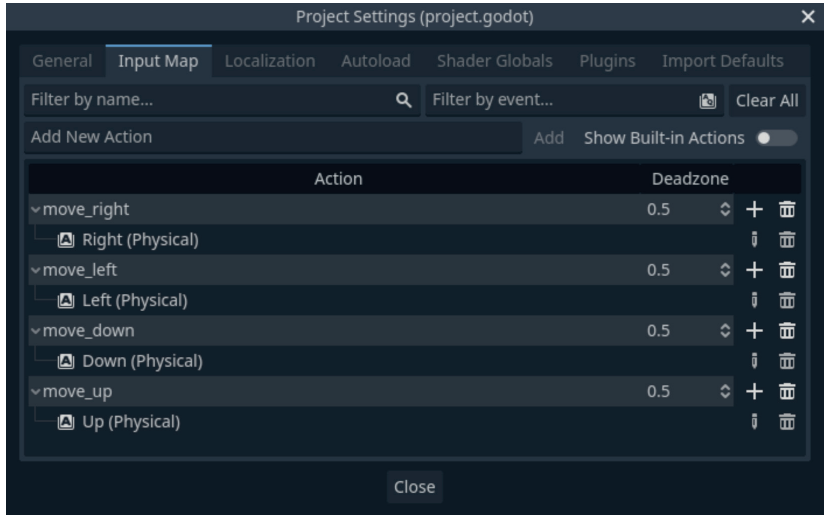


Selecione o botão “ok”. A tecla “direita” agora está associada à ação move_right.

Repita estes passos para adicionar três outros mapeamentos:

1. mover_esquerda mapeado para a tecla seta para esquerda.
2. mover_cima mapeado para tecla seta para cima.
3. E mover_baixo mapeado para tecla seta para baixo.

Sua aba de mapa de entrada deveria se parecer com isto:



Clique no botão “Fechar” para fechar as configurações de projeto.

Podemos evitar isso se normalizarmos o vetor da velocidade, o que significa que podemos definir seu comprimento (módulo) para 1 e multiplicar pela intensidade da velocidade desejada. Isso resolve o problema de movimentação mais rápida nas diagonais.

Isto define um sinal personalizado chamado “hit” (atingir) que faremos com que nosso jogador emita (envie) ao colidir com um inimigo. Iremos utilizar Area2D para detectar a colisão. Selecione o nó Jogador e clique na aba “Nó” (ao lado da aba “Inspetor”) para ver a lista de sinais que o jogador pode emitir:

Script/código e a explicação do jogador

Acessar pasta Script_Jogador do pendrive.

Com o jogador funcionando, vamos trabalhar com o inimigo na próxima lição.

< Programando o Inimigo >

10

Aula 10
– Programando o inimigo

Passo 1: Contextualização/Engajar

Criando o inimigo

Agora é hora de criarmos os inimigos de que nosso jogador terá que se desviar. Seu comportamento não será tão complexo: inimigos irão nascer aleatoriamente nos cantos da tela, escolher uma direção aleatória e irão se mover em linha reta.
Nós iremos construir uma cena Inimigo, que poderemos então instanciar para criar uma quantidade de inimigos independentes no jogo.

Configuração de nós:

Pesquise por:

RigidBody2D (chamado Turba)

TEMA:

Momento de desenvolver o inimigo do jogador

> Duração:

2h

> Materiais necessários:

Notebooks, projetor, Internet, caderno e lápis.

> Objetivo:

Criar os movimentos do inimigo do jogador.

Links de docs:

> **AnimatedSprite2d** – https://docs.godotengine.org/pt_BR/4.x/classes/class_animatedsprite2d.html#class-animatedsprite2d

> **CollisionShape2D** – https://docs.godotengine.org/pt_BR/4.x/classes/class_collisionshape2d.html#class-collisionshape2d

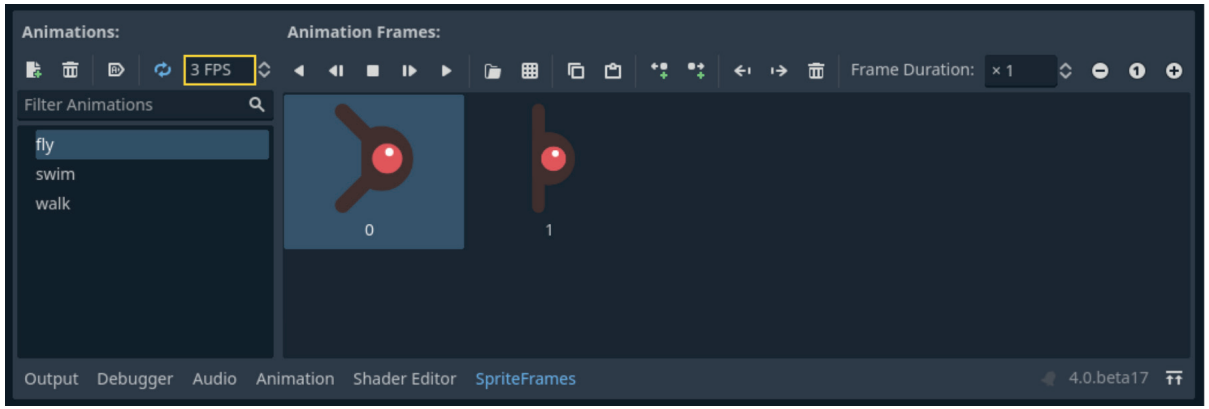
> **VisibleOnScreenNotifier2D** – https://docs.godotengine.org/pt_BR/4.x/classes/class_visibleonscreennotifier2d.html#class-visibleonscreennotifier2d

Não se esqueça de configurar os filhos para que não possam ser selecionados, assim como você fez na cena Jogador.

Nas propriedades RigidBody2D, defina Gravity Scale como 0, para que o mob não caia. Além disso, na seção CollisionObject2D, desmarque 1 dentro da propriedade Mask. Isso garantirá que os mobs não colidam uns com os outros.

Configure o AnimatedSprite2D como você fez para o player. Desta vez temos 3 animações: voar, nadar e andar. Existem duas imagens para cada animação na pasta de arte.

A propriedade Animation Speed deve ser definida para cada animação individual. Ajuste para 3 para todas as 3 animações.



Você pode usar os botões “Reproduzir animação” à direita do campo de entrada e Velocidade da animação para visualizar suas animações.

Vamos selecionar uma das animações aleatoriamente para que os inimigos tenham alguma variedade.

Assim como as imagens dos jogadores, essas imagens de mob precisam ser reduzidas. Defina a propriedade Scale do AnimatedSprite2D como (0,75, 0,75).

Como na cena do Jogador, adicione um CapsuleShape2D para as questões. Para alinhar a forma com a imagem, você precisa definir a propriedade Rotation Degrees (grau de rotação) como 90 na seção Transform no Inspetor.

Salve uma cena.

Script/código e a explicação do inimigo

Acesse a pasta “Script do inimigo” do pen drive.

Isto completa a cena Turba.

Com o jogador e os inimigos prontos, na próxima parte os juntaremos em uma nova cena. Faremos inimigos aparecerem aleatoriamente na área do jogo, moverem-se adiante, tornando nosso projeto um jogo de fato.

< Programando a cena principal >

11

Aula 11
- Programando a cena principal

TEMA:

Momento de desenvolver a cena principal

> Duração:

2h

> Materiais necessários:

Notebooks, projetor, Internet, caderno e lápis.

> Objetivo:

Criar a cena onde será unida a cena do jogador e a cena do inimigo.

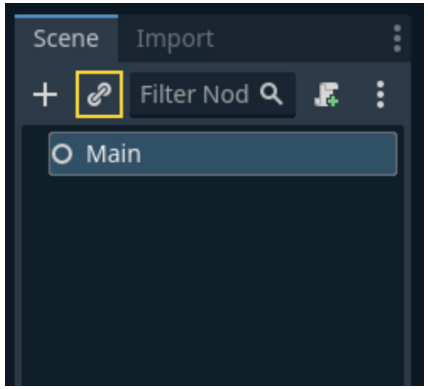
Passo 1: Contextualização/Engajar

A cena principal do jogo

Agora é hora de juntar tudo o que fizemos em uma cena jogável.

Crie uma nova cena e adicione um :ref: Node <class_Node> chamado Main. (O motivo para usarmos um Node ao invés de um Node2D é que esse será um contêiner para lidar com a lógica do jogo, pois ele não necessita de funcionalidade 2D.)

Clique no botão Instância (representado por um ícone de elo de corrente) e selecione seu player.tscn salvo.



> **Timer (nomeado MobTimer)** – para controlar a frequência com que a turba é gerada.
>
> **Timer (nomeado ScoreTimer)** - Para incrementar a pontuação a cada segundo.
>
> **Timer (nomeado StartTimer)** - para dar um atraso antes de começar.
>
> **Marker2D (named StartPosition)** - to indicate the player's start position.

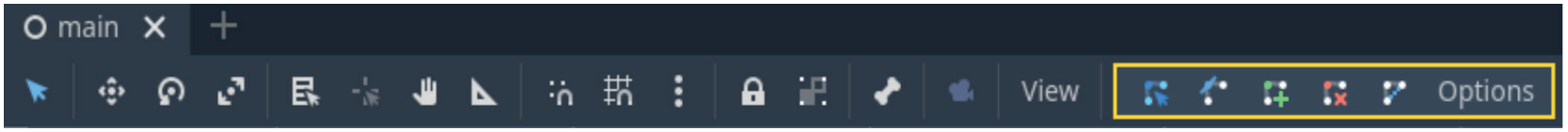
Defina a propriedade Wait Time (tempo de espera) para cada um dos nós Timer da seguinte forma:

1. MobTimer: 0.5
2. ScoreTimer: 1
3. StartTimer: 2

Além disso, configure a propriedade One Shot ("uma só vez") de StartTimer para "Ativo" e Position do nó StartPosition para (240, 450).

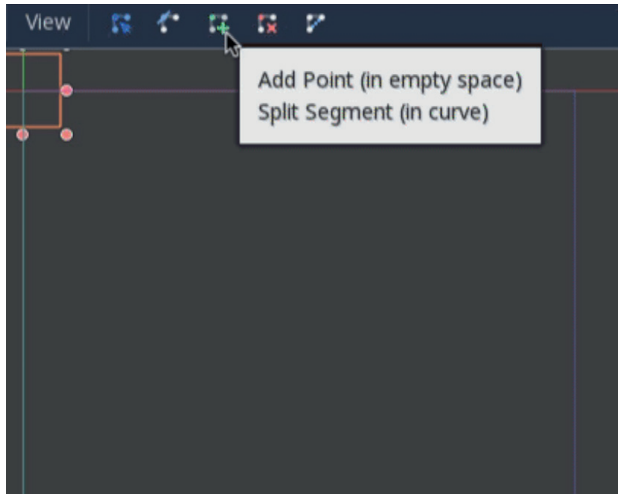
Gerando monstros

O nó Principal ficará gerando novos inimigos, e nós queremos que eles apareçam em lugares aleatórios nos cantos da tela. Adicione um nó Path2D chamado CaminhoTurba como filho de Principal. Quando você selecionar Path2D, aparecerão alguns botões novos na parte superior do editor:



Selecione o do meio ("Adicionar Ponto") e desenhe o caminho clicando para adicionar os pontos nos cantos mostrados. Para que os pontos se encaixem na grade, certifique-se de que "Use Grid Snap" e "Use Smart Snap" estejam selecionados. Essas opções podem ser encontradas à esquerda do botão "Bloquear", aparecendo como um ímã próximo a alguns pontos e linhas que se cruzam, respectivamente.

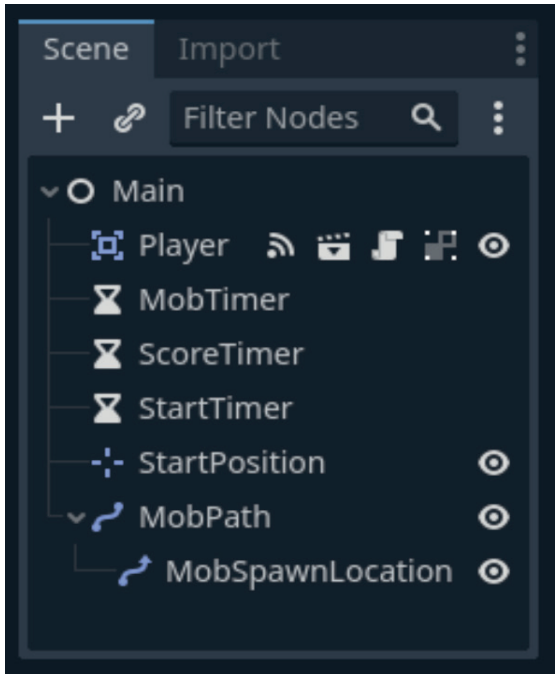
Importante: Desenhe o caminho em sentido horário ou sua turba vai surgir apontando para fora em vez de para dentro!



Depois de colocar o ponto 4 na imagem, clique no botão “Fechar Curva”, e sua curva estará completa.

Agora que o caminho está definido, adicione um nó PathFollow2D como filho de CaminhoTurba e dê o nome de LocalGeraçãoTurba. Esse nó vai rotacionar automaticamente e seguir o caminho conforme ele se move, para que possamos usá-lo para selecionar uma posição e uma direção aleatória ao longo do caminho.

Sua cena deve se parecer com isso:



Script/código da cena principal

Acesse a pasta “Script principal” do pen drive.

Importante: Por que PI? Em funções que requerem ângulos, Godot usa radianos, não graus. Pi representa meia volta em radianos, cerca de 3,1415 (há também TAU que é igual a 2*PI). Se você se sentir mais confortável trabalhando com graus, precisará usar as funções deg_to_rad() e rad_to_deg() para converter entre os dois.

< Monitor de alerta >

12

Aula 12
-Monitor de alerta

Passo 1: Contextualização/Engajar

Monitor de alerta

A peça final que nosso jogo precisa é uma Interface de Usuário (UI) para exibir coisas como pontuação, uma mensagem de “game over” e um botão de reiniciar.

Crie uma nova cena e adicione um nó CanvasLayer chamado HUD. “HUD” significa “heads-up display”, uma exibição informativa que aparece como uma sobreposição na parte superior da visualização do jogo.

O nó CanvasLayer nos permite desenhar nossos elementos de interface em uma camada acima do resto do jogo, de forma que as informações que ela mostra não sejam cobertas por quaisquer elementos do jogo, como o jogador ou os inimigos.

O HUD exibirá as seguintes informações:

1. Pontuação alterada para ScoreTimer.
2. Uma mensagem, como “Game Over” ou “Prepare-se!”
3. Um botão “Iniciar” para começar o jogo.

TEMA:

Criar o monitor de alerta

> Duração:

2h

> Materiais necessários:

Notebooks, projetor, Internet, caderno e lápis.

> Objetivo:

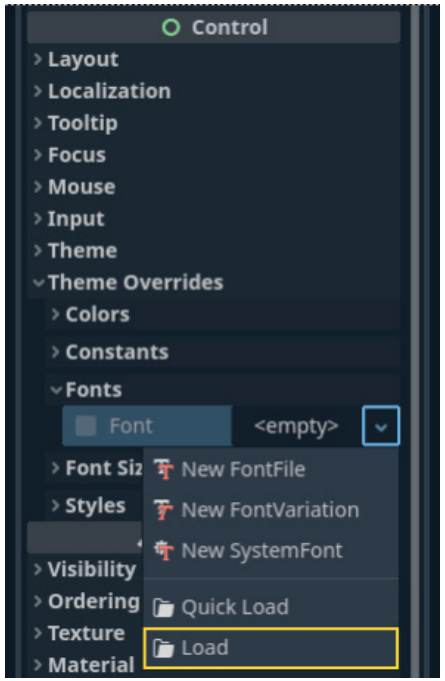
Criar uma nova cena e adicione um nó CanvasLayer chamado HUD.

4. O nó básico para elementos de interface é Control. Para criar nossa interface, usaremos dois tipos de nós Control: Label e Button.

Crie os seguintes itens como filhos do nó HUD:

1. Label nomeado ScoreLabel.
2. Etiqueta nomeada MessageLabel.
3. Botão nomeado StartButton.
4. Timer nomeado MessageTimer.

Clique no ScoreLabel e digite um número no campo Text no Inspetor. A fonte padrão para nós Controle é pequeno e não escala bem. Há um arquivo de fonte incluído nos assets do jogo chamado “Xolonium-Regular.ttf”. Para usar esta fonte, faça o seguinte para cada um dos três nós Controle:



Em “Substituições de tema > Fontes”, escolha “Carregar” e selecione o arquivo “Xolonium-Regular.ttf”.

O tamanho da fonte ainda é muito pequeno, aumente para 64 em “Substituições de tema > Tamanhos de fonte”. Depois de fazer isso com o ScoreLabel, repita as alterações para os nós Message e StartButton.

Âncoras: os nós de controle têm posição e tamanho, mas também possuem âncoras. As âncoras definem a origem - o ponto de referência para as arestas do nó.

Organize os nós conforme mostrado abaixo. Você pode arrastar os nós para posicioná-los manualmente ou, para um posicionamento mais preciso, usar “Predefinições de âncora”.

Marcador de pontuação¶
Adicione o texto 0.

Defina o “Alinhamento Horizontal” e “Alinhamento Vertical” para Centro.

Escolha a parte superior central “Anchor Preset”.

Mensagem¶
Adicione o texto Evite os arrepios!.

Defina o “Alinhamento Horizontal” e “Alinhamento Vertical” para Centro.

Defina o “Modo Autowrap” para Word, caso contrário o rótulo permanecerá em uma linha.

Em “Control - Layout/Transform” defina “Size X” como 480 para usar toda a largura da tela.

Escolha o centro “Predefinição de âncora”.

Marcador de pontuação

1. Adicione o texto 0.
2. Defina o “Alinhamento Horizontal” e “Alinhamento Vertical” para Center.
3. Escolha a “Predefinição de âncora”. Center Top

Messagem

1. Adicione o texto .Dodge the Creeps!
2. Defina o “Alinhamento Horizontal” e “Alinhamento Vertical” para Center.
3. Defina o “Modo Autowrap” para Word, caso contrário o rótulo permanecerá em uma linha.
4. Em “Control - Layout/Transform” defina “Size X” para 480 e usar toda a largura da tela.
5. Escolha a “Predefinição de âncora” Center.

Botão de iniciar

1. Adicione o texto Start.
2. Em “Control - Layout/Transform”, defina “Tamanho X” como 200, “Tamanho Y” como 100 e adicionar um pouco mais de preenchimento entre a borda e o texto.
3. Escolha a “Predefinição de âncora” .Center Bottom
4. Em “Control - Layout/Transform”, defina “Posição Y” como 580.

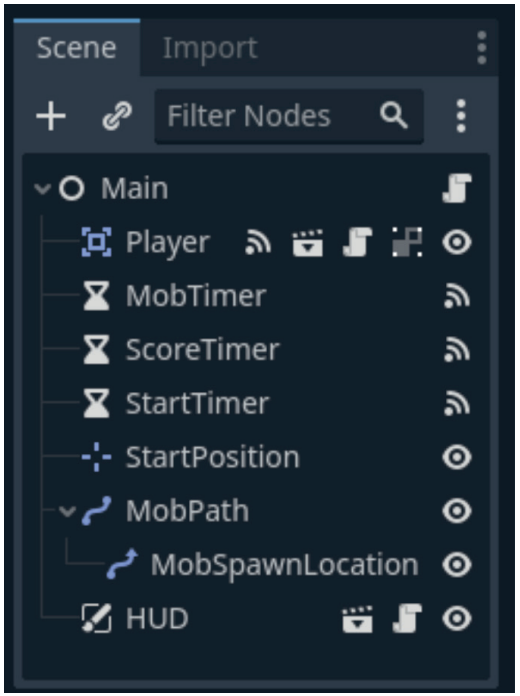
No MessageTimer, defina o Wait Time (Tempo de Espera) para 2 e configure a propriedade One Shot (Apenas uma Vez) como “Ativo”.

Script/Código HUD

Acesso a pasta “HUD” no pen drive.

Conectando HUD a Principal

Agora que terminamos de criar a cena HUD, salve-a e volte para a Principal. Crie uma instância da cena HUD como fez com a cena Jogador, e coloque-a no final da árvore. A árvore completa deveria se parecer como a imagem ao lado.



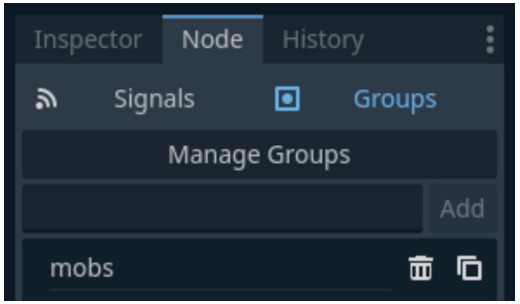
Agora precisamos conectar a funcionalidade de HUD ao roteiro de Principal. Isso exige algumas adições à cena Principal:

Agora você está pronto para jogar! Clique no botão “Reproduzir o Projeto”. Você será solicitado a selecionar uma cena principal, então escolha main.tscn.

Removendo criaturas antigas

Se você jogar até o “Game Over” e iniciar um novo jogo, as criaturas do jogo anterior ainda poderão estar na tela. Seria melhor se todas elas desaparecessem no início de cada partida. Nós só precisamos de um jeito de falar para que todos os inimigos se autodestruam. Nós podemos fazer isso com a funcionalidade “grupo”.

Na cena do Inimigo, selecione o nó raiz e clique na aba “Nó” próximo ao Inspetor (no mesmo lugar onde você encontra os sinais do nó). Próximo a “Sinais”, clique em “Grupos”, você pode digitar o nome do novo grupo e clicar em “Adicionar”.



Agora todos os inimigos estarão no grupo “inimigos”. Podemos então adicionar a seguinte linha à função game_over() em Main:

A função “call_group()” chama a função passada como parâmetro em cada nó do grupo - neste caso nós estamos falando para cada inimigo se autodestruir.

O jogo está quase pronto nesse ponto. Na próxima e última parte, daremos um melhor acabamento adicionando um plano de fundo, música em “looping” e alguns atalhos de teclado.

< Terminando >

13

Aula 13 -Terminando

TEMA:

Momento de design final

> Duração:

2h

> Materiais necessários:

Notebooks, projetor, Internet, caderno e lápis.

> Objetivo:

Agora completamos toda a funcionalidade do nosso jogo.

Passo 1: Contextualização/Engajar

Terminando

Agora completamos toda a funcionalidade do nosso jogo. Abaixo alguns passos restantes para dar uma apimentada e melhorar a experiência do jogo.

Sinta-se livre para expandir a jogabilidade com suas próprias ideias.

Plano de Fundo

O fundo cinza padrão não é muito atraente, então vamos mudar sua cor. Uma maneira de fazer isso é usar um nó ColorRect. Torne-o o primeiro nó em Principal para que seja desenhado atrás dos outros nós. ColorRect possui apenas uma propriedade: Color. Escolha uma cor de sua preferência e selecione “Layout” -> “Anchors Preset” -> “Full Rect” na barra de ferramentas na parte superior da janela de visualização ou no inspetor para cobrir a tela.

Você também pode adicionar uma imagem de plano de fundo, se tiver uma, ao usar um nó TextureRect.

Efeitos sonoros

Som e música podem ser uma forma mais eficaz de adicionar um atrativo à experiência de jogo. Na pasta de ativos do seu jogo, você tem dois arquivos de áudio: “House In a Forest Loop.ogg” para música de fundo e “gameover.wav” para quando o jogador perde.

Adicione dois nós AudioStreamPlayer como filhos do Principal. Nomeie um deles como Musica e o outro como SomDeMorte. Em cada um, clique na propriedade Stream (“fluxo”), selecione “Carregar” e escolha o arquivo sonoro correspondente.

Todo o áudio é importado automaticamente com a configuração Loop desativada. Se você quiser que a música faça um loop contínuo, clique na seta do arquivo Stream, selecione Tornar único, clique no arquivo Stream e marque a caixa Loop.

Para reproduzir música, adicione $Musica.play() na função new_game() e $Musica.stop() na função game_over().

Por fim, adicione $SomDeMorte.play() na função game_over().

```
func game_over():
    ...
    $Music.stop()
    $DeathSound.play()

func new_game():
    ...
    $Music.play()
```

Atalho de teclado

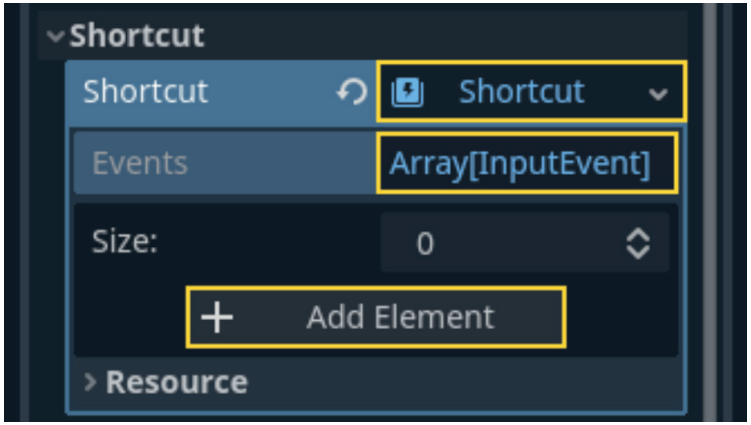
Como o jogo é jogado com controles de teclado, seria conveniente se pudéssemos iniciar o jogo pressionando uma tecla do teclado. Uma maneira de fazer isso é usando a propriedade “Atalho” do nó Button.

Numa lição anterior, criamos quatro ações de entrada para mover o personagem. Criaremos uma ação de entrada semelhante ao botão de início.

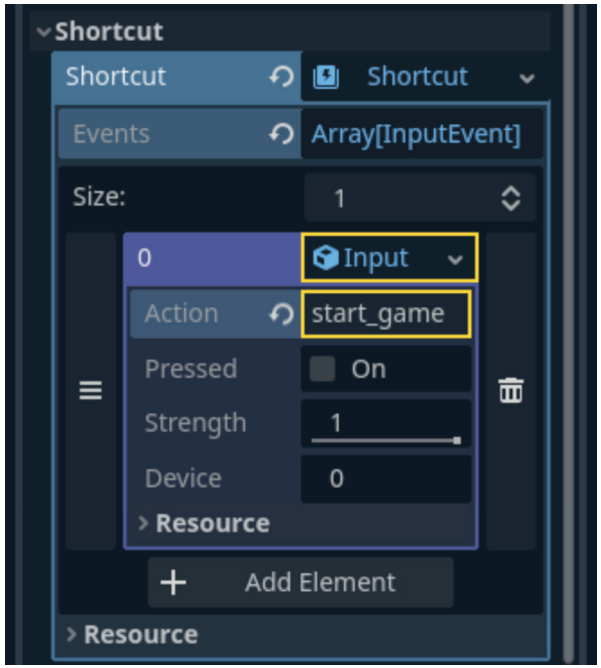
Selecione “Projeto” -> “Configurações do Projeto” e clique na aba “Mapa de Entrada”. Da mesma forma que criou as ações de entrada para movimentação, crie uma nova entrada chamada start_game e adicione uma tecla mapeada para Enter.

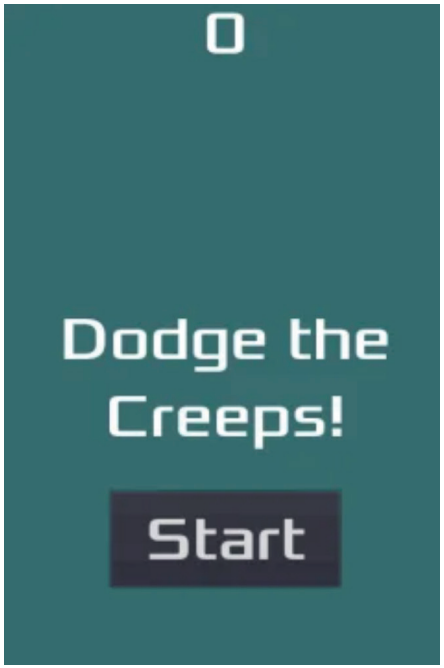
Agora seria um bom momento para adicionar suporte ao controlador, se você tiver um disponível. Anexe ou emparelhe seu controlador e, em cada ação de entrada para a qual deseja adicionar suporte de controlador, clique no botão “+” e pressione o botão, d-pad ou direção do stick correspondente que deseja mapear para a respectiva ação de entrada.

Na cena do HUD, selecione o StartButton e encontre sua propriedade Shortcut no Inspector. Crie um novo recurso de atalho clicando dentro da caixa, abra o array Events e adicione um novo elemento de array clicando em Array[InputEvent] (tamanho 0).



Crie uma nova InputEventAction e nomeie-a como start_game.





Agora, quando o botão iniciar aparecer, você pode clicar nele ou pressionar Enter para iniciar o jogo.

E com isso, você completou seu primeiro jogo 2D em Godot.

Você acabou de fazer um personagem controlado pelo jogador, inimigos que são gerados aleatoriamente pela tela de jogo, contador de pontos, implementar um fim de jogo e repetição, interface de usuário, sons, e mais. Congratulações!!!

Ainda há muito a aprender, mas você pode dar um tempo e apreciar sua conquista.

Realização:

The Human Project

Patrocínio:

CMDCA
CONSELHO MUNICIPAL DOS DIREITOS DA CRIANÇA E DO ADOLESCENTE DE SANTA LUZIA DO ITANHI

FMDCA

Apoio:

elo

SIEMENS | Fundação

kapitalo investimentos