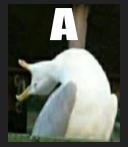
IFPE
Open Source

Obrigado pela presença!

Perguntas a qualquer hora, chat ou mic

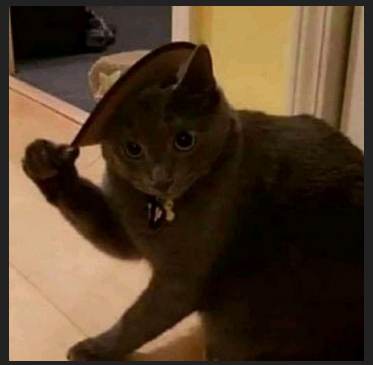
Qualquer um pode fazer um workshop

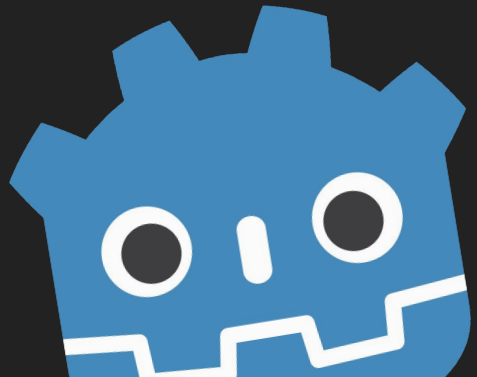
G 

meDev com Godot

E pong também

IFPE
Open Source





São muitas...

Mas por que a Godot?

A Escolha

Vai da sua demanda

Generalistas/Especialistas

Unity

GODOT
Game engine

GameMaker Studio 2™

UNREAL ENGINE

RPG MAKER MV

A Escolha

A Unity é podre...

O mercado também :(

GODOT

Game engine

As vantagens da Godot

- Grátis

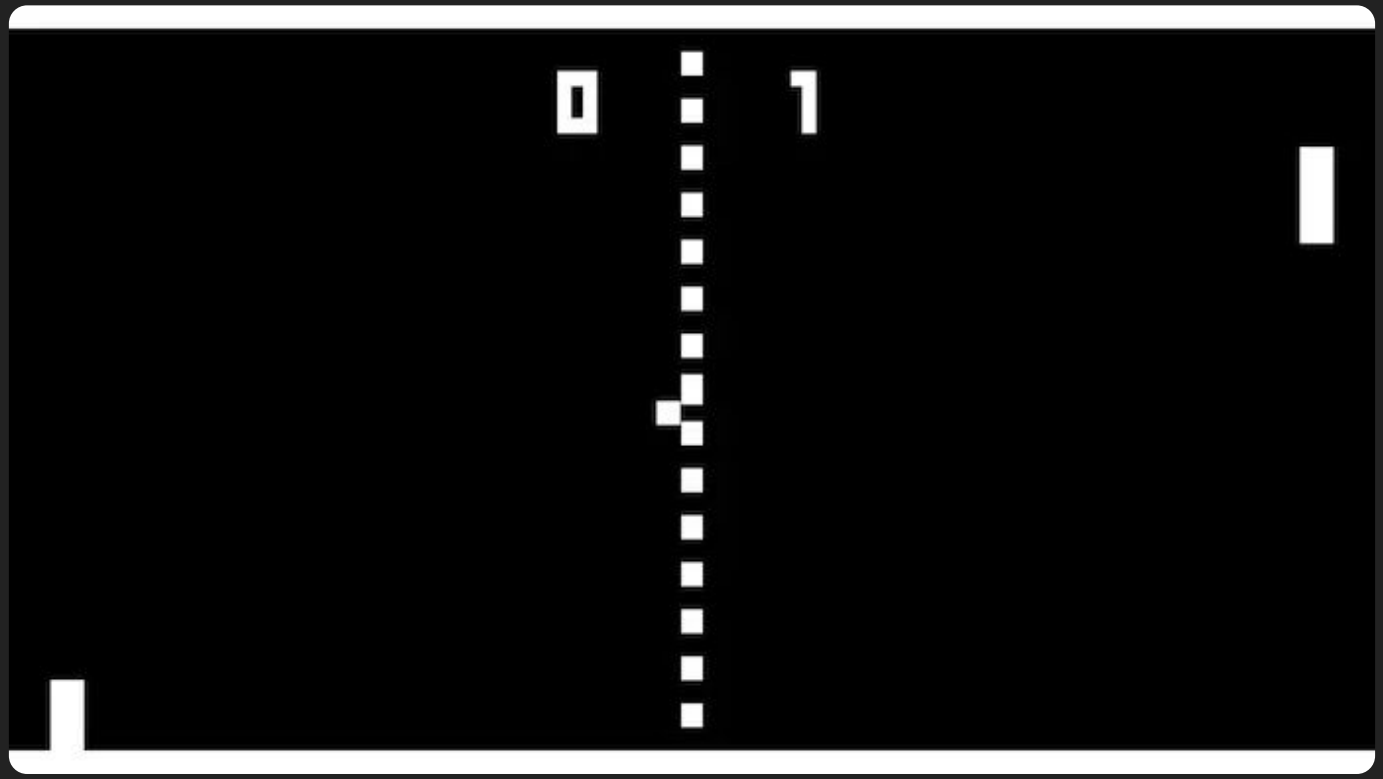
- UI: a mistura perfeita

- O jogo é completamente seu, sem créditos

- Open Source

- QUALQUER LINGUAGEM O_O

What are we gonna do?



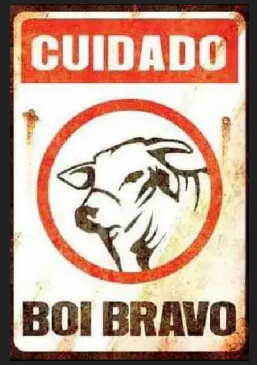
Objetivo

Uma história legal

William Higinbotham
1958

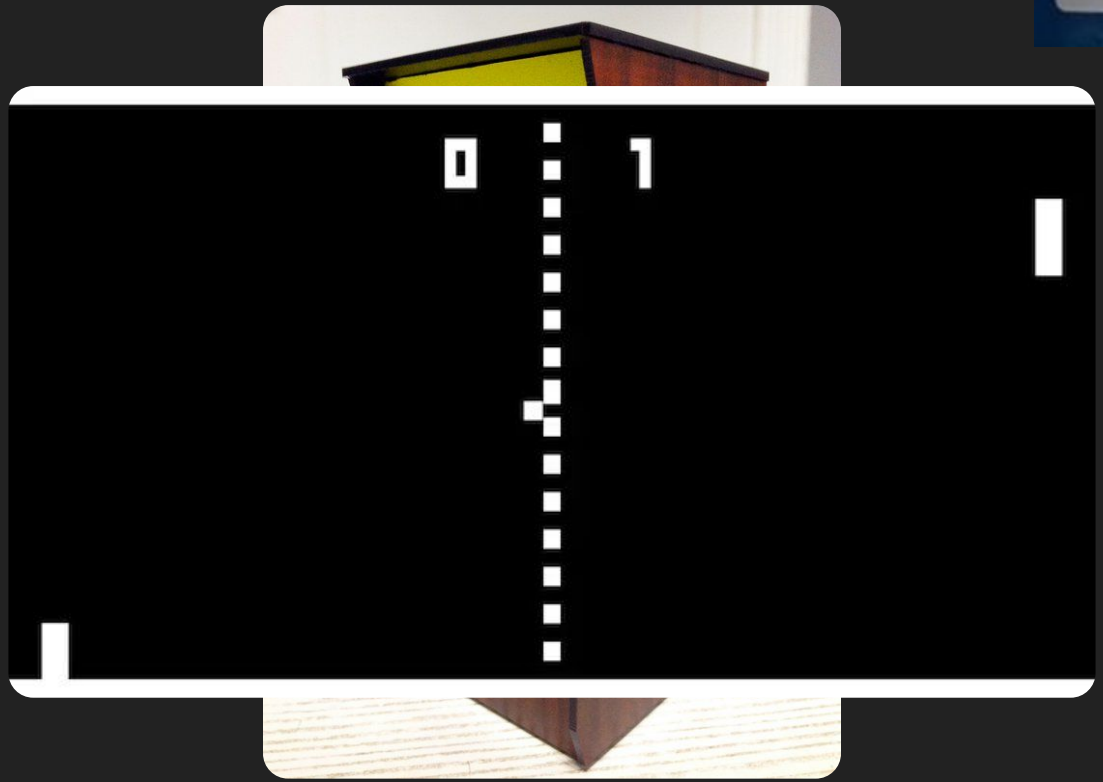
Uma história legal

Ralph Baer
1972



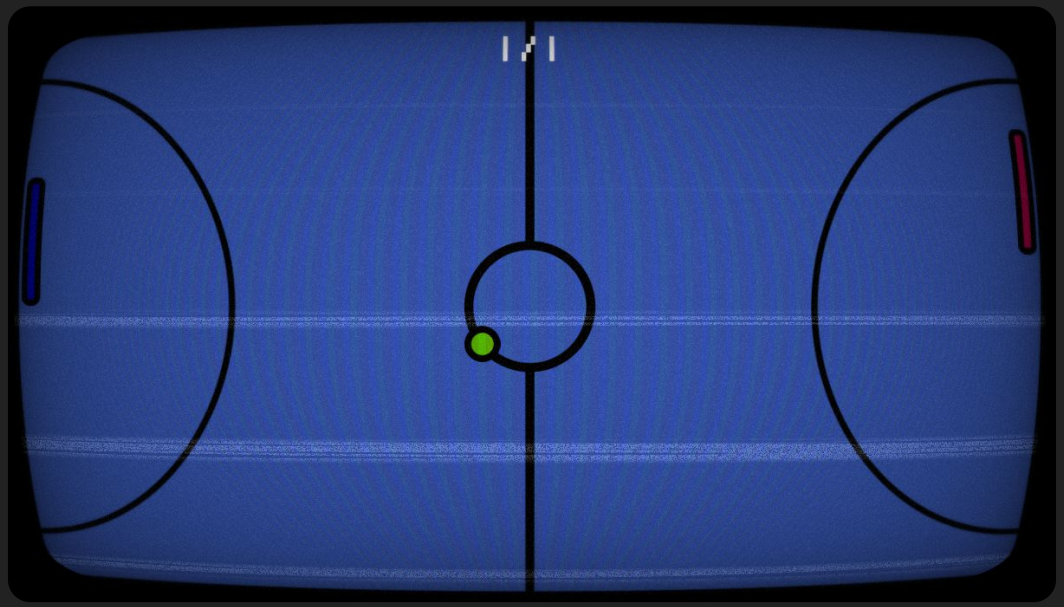
Uma história legal

Atari
1972



Uma história legal

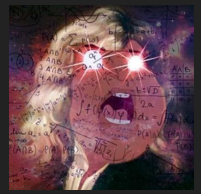
A gente
2021



É fácil?

Conhecimentos necessários para fazer o jogo

- Manipulação de vetores e ângulos



- Lógica de programação

e só :D

Requisitos

Fundamentos da Matemática

A palavra com “V” O - O

Vetores

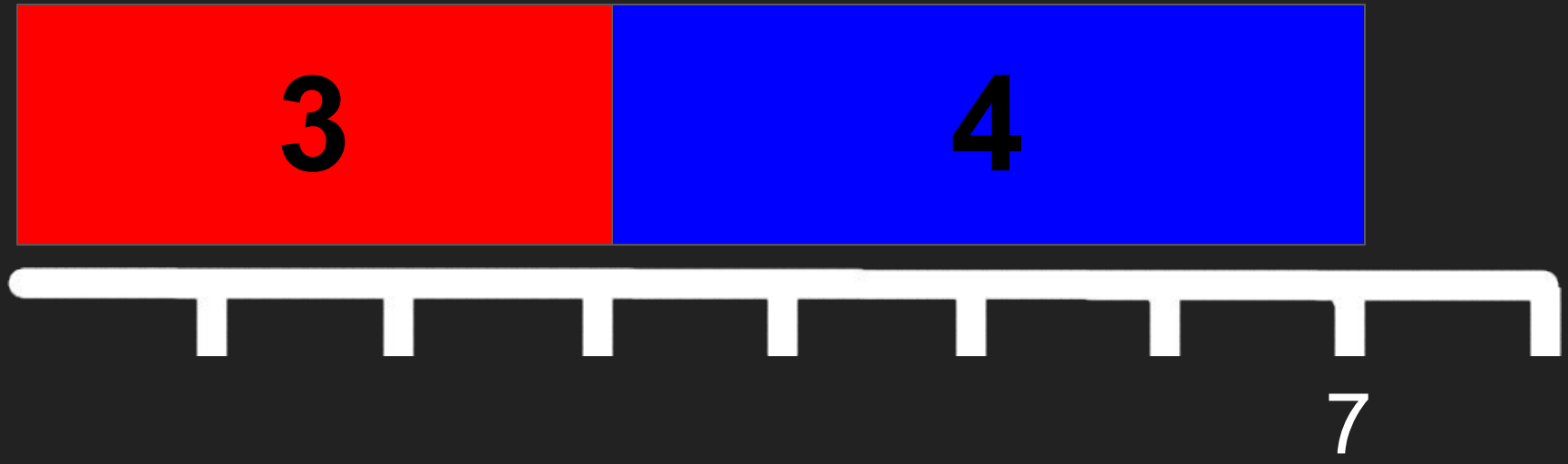
Pense em números

Ou setas no espaço

Ou matrizes

Na verdade, não importa

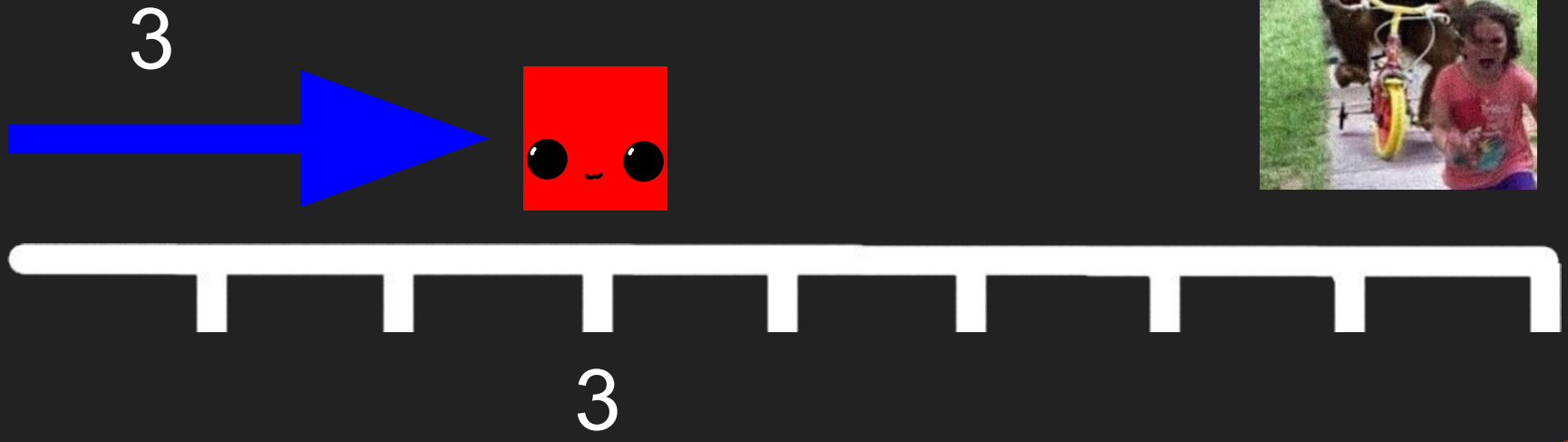
Réguas



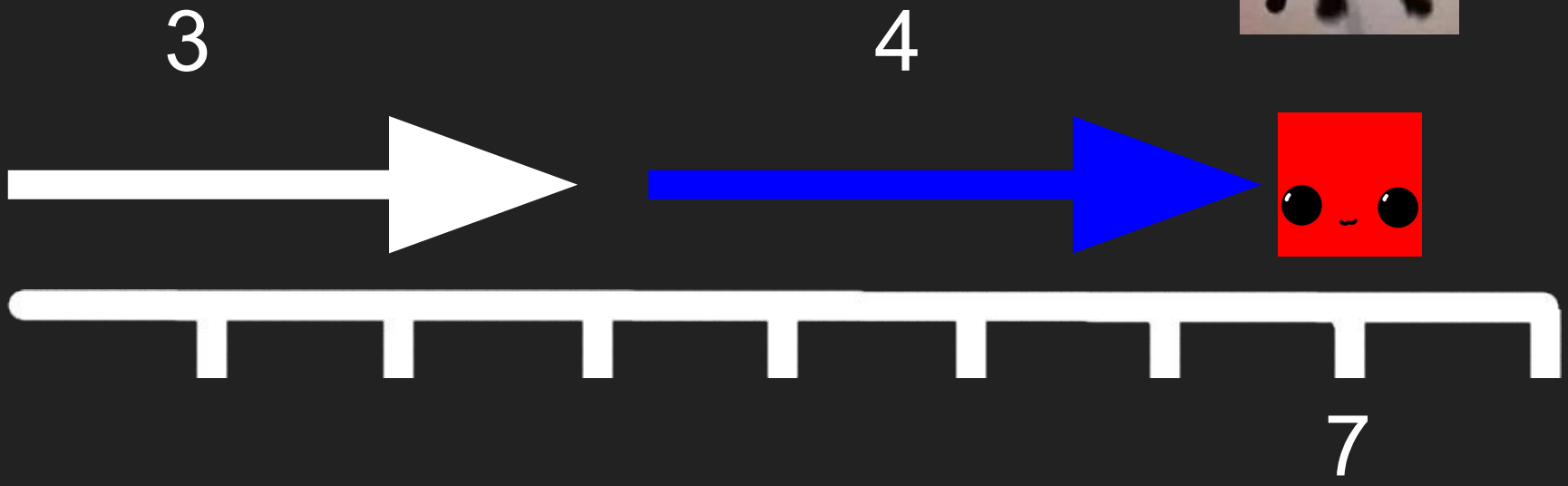
Réguas indo pro outro lado

Adamastor é real e pode te machucar

Adamastor é real e pode te machucar

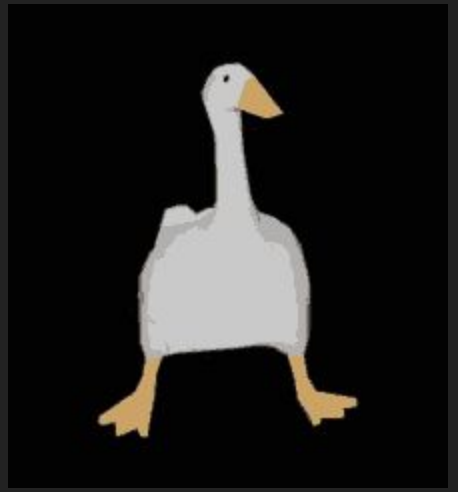


Adamastor é real e pode te machucar

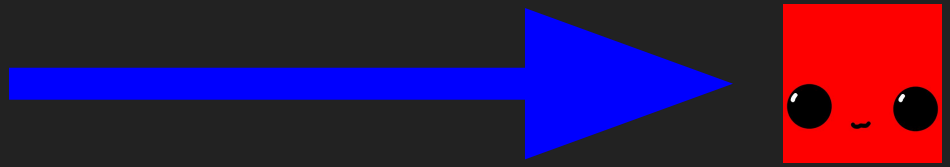


Adamastor é real e pode te machucar

Adamastor é real e pode te machucar



4



4

Adamastor é real e pode te machucar

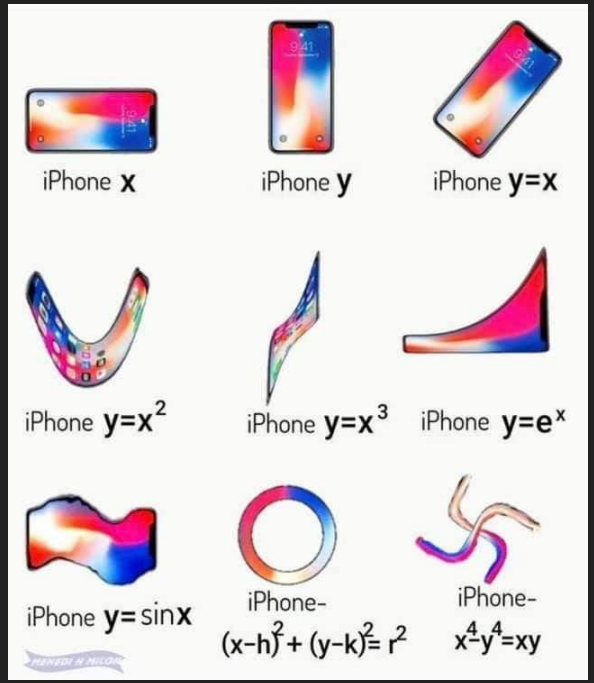
Convenção

Vetores

Sistema cartesiano

Sentido da seta

Vetores equivalentes

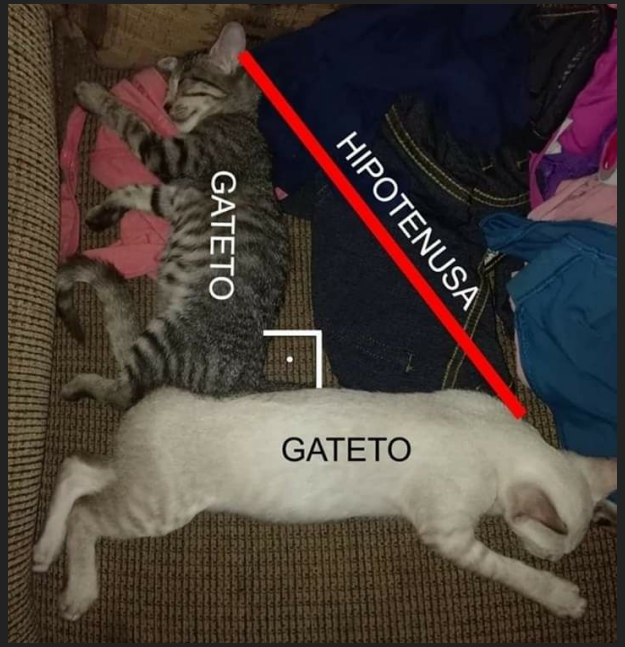


A palavra com “A” O - O

Ângulos

Definição

Graus e radianos (e GRADOS)



Fundamentos da programação

O que é programação?

Basicamente, se resume à instruções em binário para a CPU.

exceto shaders...

Evolução da programação

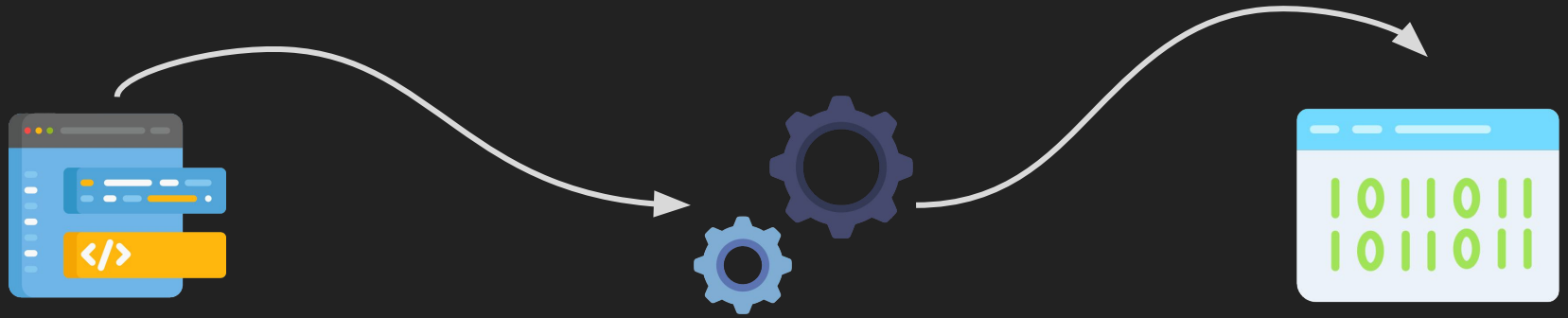
Os primeiros computadores não tinham uma linguagem de programação

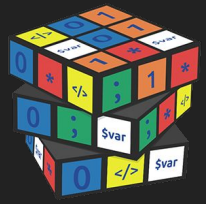
A programação deles era feita através de plugs e botões.

Evolução da programação

Com a evolução dos computadores programar se tornou mais fácil

Agora com a invenção das linguagens de programação, tornou-se possível criar um código fonte, escrito em linguagem de alto nível, o qual passa por um compilador, que o traduz para linguagem de baixo nível.

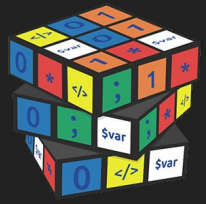




Principais tópicos

A base da lógica permanece sempre a mesma, portanto vamos abordar três tópicos aqui.

- Variáveis;
- Estruturas de Decisão;
- Métodos.

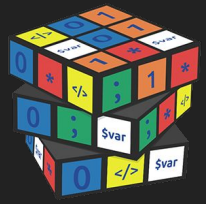


Por que não veremos estruturas de repetição

Geralmente quando estudamos fundamentos da programação, também também se vê estruturas de repetição. Mas isso não será abordado aqui por 2 motivos:

- As estruturas de repetição são usadas para fazer loops infinitos, na Godot esse loop já vem pronto;
- No jogo que vamos fazer aqui, não vamos precisar desse tipo de estrutura.

SE VOCÊ QUISER VER AS ESTRUTURAS DE REPETIÇÃO, ENTRE NO DISCORD DO IFPE OPEN SOURCE, ESTAMOS PREPARANDO ALGUNS PROJETOS E TALVEZ ROLE UM BOOTCAMP

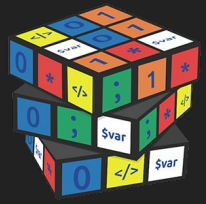


Variáveis

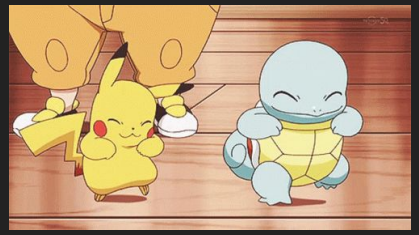
São espaços na memória que armazenam uma informação.

Elas podem armazenar valores de vários tipos, sejam números, palavras, listas, dicionários ou vetores

```
1  var num = 10
2  var nome = 'Roger'
3  var vetor = Vector2(8,5)
4
```

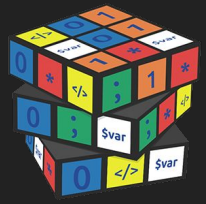


Variáveis



Esses valores podem ser usados, referenciados e modificados a qualquer momento.

```
1   var nome = 'Roger'
2   print(nome)
3
```

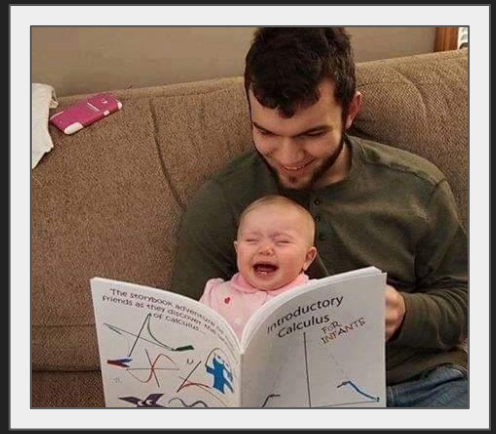


Operadores

São, basicamente, símbolos utilizados para representar uma operação matemática, lógica ou relacional entre dois valores.

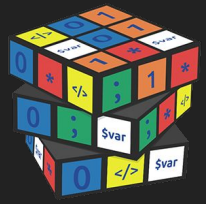
Eles se dividem em 3 tipos:

- Operadores Aritméticos;
- Operadores relacionais;
- Operadores lógico.



Operadores

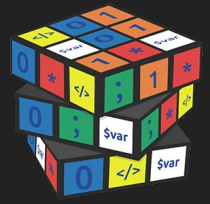
A seguir, está a lista de operadores suportados pela godot e sua precedência.



Estruturas de Decisão

As estruturas de decisão tem a utilidade de fazer o computador realizar ações diferentes baseado nas condições que você colocar.

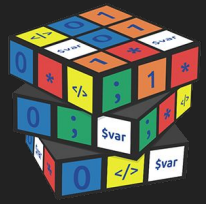
```
4       var num = 10
5
6       if num < 10:
7           print("num menor que 10")
```



Estruturas de Decisão

É possível adicionar outra condicional à esta estrutura, o ELSE, que será executado quando a condição do IF não for atendida.

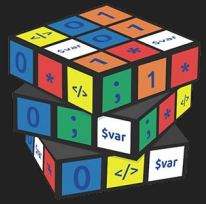
```
6   var num = 10
7
8   if num < 10:
9       print("num menor que 10")
10  else:
11      print("num maior ou igual a 10")
```



Estruturas de Decisão

Nada te impede de colocar uma estrutura de controle dentro de outra estrutura de controle, criando uma estrutura aninhada.

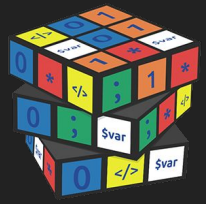
```
var num = 10

if num < 10:
    if num % 2 == 0:
        print("num é número par e menor que 10")
else:
    if num < 15:
        print("num menor que 15")
    else:
        print("num maior ou igual a 15")
```



Estruturas de Decisão

É possível fazer isso de outra forma, que deixa o código mais limpo e fácil de compreender, usando operadores lógicos.

```
6  var num = 10
7
8  if num < 10 and num % 2 == 0:
9      print("num é número par e menor que 10")
10 else:
11     if num < 15:
12         print("num menor que 15")
13     else:
14         print("num maior ou igual a 15")
```



Estruturas de Decisão

Outra maneira de organizar o código é adicionando outra condicional à esta estrutura, o ELIF.

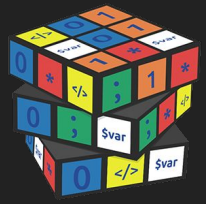
```
5
6   var num = 10
7
8   if num < 10 and num % 2 == 0:
9       print("num é número par e menor que 10")
10  elif num < 15:
11      print("num menor que 15")
12  else:
13      print("num maior ou igual a 15")
14
```

Estruturas de Decisão

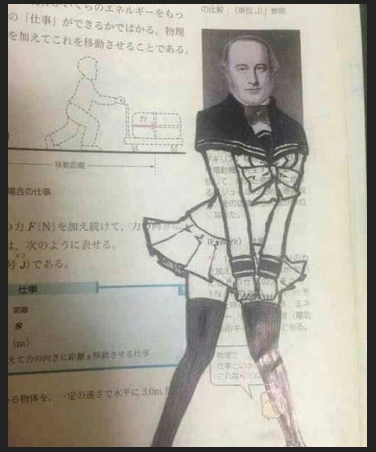
Outra estrutura de decisão é o match.

Essa é a sua estrutura básica:

```
5
6  match x:
7      1:
8          print("x é igual a 1")
9      2:
10         print("x é igual a 2")
11     "teste":
12         print("x é igual a 'teste'")
13
```



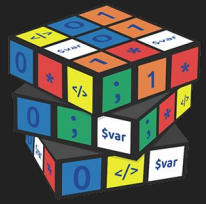
Estruturas de Decisão



É possível adicionar um método padrão ao match.

isso pode ser feito colocando um “_” no lugar do padrão

```
5
6  match x:
7      1:
8          print("x é igual a 1")
9      2:
10         print("x é igual a 2")
11     _:
12         print("x não é igual a 1 ou 2")
13
```

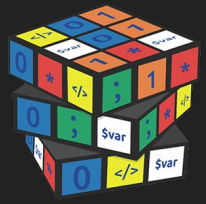


Estruturas de Decisão

Também é possível fazer várias verificações em uma única linha.

isso pode ser feito separando os padrões por vírgula.

```
5
6  match x:
7      1, 2, 3:
8          print("x é igual a 1, 2 ou 3")
9      _:
10         print("x é diferente de 1, 2 ou 3")
11
```

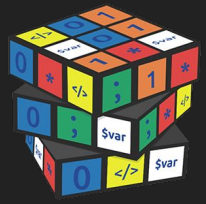


Funções

As funções são pedaços de código que podem ser reutilizados a qualquer momento no código.

Essa é a sintaxe básica para a definição de uma função:

```
2
3  func some_function(param1, param2):
4      print(param1)
5      print(param2)
6      return param1 + param2 # Return é opcional; sem ele 'null' é retornado.
7
```



Funções

É possível definir os tipos e os valores padrões dos parâmetros

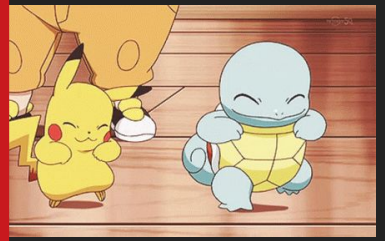
```
3 func some_function(param1: int, param2: String):
4     print(param1)
5     print(param2)
6     return param1
7
```

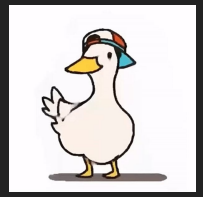
```
3 func rogerio(param1 := 10, param2 := "nice"):
4     print(param1)
5     print(param2)
6     return param1
```

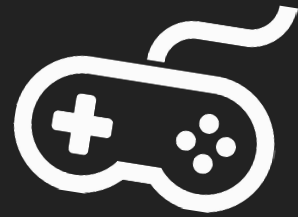
Desafios

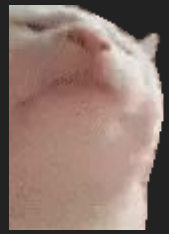
- Adicionar um segundo player local
- Adicionar controle de ângulo do batedor
- Adicionar um Menu Principal
- Adicionar uma tela de Game Over
- Upgrade de pontos para a bolinha
- Upgrade de velocidade para a bolinha

Obrigado!









Colocar a mão na massa

Créditos

Fontes das imagens/vídeos usados:

- https://www.gamedevdrops.com/melhores-game-engines/
- https://youtu.be/to-50Kydrj4
- Icons made by Good Ware from www.flaticon.com
- Icons made by juicy_fish from www.flaticon.com
- Icons made by Freepik from www.flaticon.com
- Tehilson Rocha

graphic design is my passion.