

Fundamentos de Programação Aplicada a Jogos

Jarbas Baltazar



Sobre

31 anos, pai da Helena, esposo da Deisy, Engenheiro de Software especializado em .NET e Azure, com mais de 9 anos de experiência no setor. Tenho graduação em Análise e Desenvolvimento de Sistemas pela Estácio e uma vasta experiência no desenvolvimento de RPA, ERPs e Fintechs.

A minha paixão por tecnologia vai além do trabalho; aprecio jogos, séries e filmes, particularmente os de terror, e acredito que esses interesses contribuem para manter a criatividade e o entusiasmo que transmito para o meu trabalho.

Acredito na importância de criar soluções eficientes e integradas, e estou sempre em busca de novos conhecimentos para aprimorar meu trabalho e impactar positivamente os projetos em que estou envolvido.

O que Veremos



01

História e Evolução da
Programação de Jogos

02

Fundamentos Essenciais
de Programação

03

Lógica de Programação
em Jogos

04

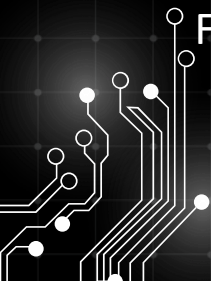
Ferramentas e Engines

05

Dicas e Boas Práticas

06

Tendências e Futuro da
Programação de Jogos



01

História e Evolução da Programação de Jogos



História e Evolução da Programação de Jogos



Tennis for Two

Criado em 1958 por William A. Higinbotham, foi um dos primeiros videogames.

Magnavox Odyssey

Lançado em 1972, foi o primeiro console de videogame. A ideia surgiu com Ralph Baer em 1966, que queria criar algo que pudesse interagir com a TV.

Computer Space

Lançado em 1971, foi o primeiro jogo de fliperama. Nolan Bushnell adaptou o jogo Spacewar!, de Steve Russell, para criar o Computer Space.

Nintendo 64, Sony Playstation e Sega

Lançados na década de 90, foram alguns dos destaques da quinta geração de consoles.

Jogos em 3D

Começaram a ser desenvolvidos na década de 90.

Realidade Virtual, Jogos Móveis e eSports

A década de 2010 foi marcada pela ascensão da realidade virtual (VR), com dispositivos como o Oculus Rift e o HTC Vive permitindo uma imersão nunca antes vista nos jogos.

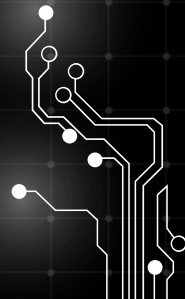


```
rel="stylesheet" href="http://1
pt type="text/javascript" src="http://1
type="text/javascript">
ction(){
onLoaded: function(request) {
    if (request.name == 'log_error') retu
    log.trace("Ajax.Request: " + (request
    )) + "...");
},
onComplete: function(request) {
    if (request.name == 'log_error') retu
},
onException: function(request, e) {
    if (request.name == 'log_error') retu
    log.fatal(request.url + ' : ' + e.nam
    .stack);
}
);
```

Fundamentos Essenciais de Programação

Consistem nos conceitos e técnicas básicas que todo programador deve entender para escrever, ler e depurar códigos. Estes conceitos servem de base para a programação em qualquer linguagem e são indispensáveis para o desenvolvimento de sistemas, aplicativos, jogos e afins

- Variáveis e Tipos de Dados
- Estruturas Condicionais
- Estruturas de Repetição
- Funções (ou Métodos)
- Estruturas de Dados
- Orientação a Objetos (OOP)
- Manipulação de Erros e Exceções



Variáveis e Tipos de Dados

Variáveis são recipientes ou lugares onde guardamos valores que o programa usa e manipula. Elas possibilitam atribuir nomes aos dados, simplificando sua utilização e compreensão.

Tipos de Dados indicam o tipo de informação que uma variável pode guardar, tais como números inteiros (int), números de ponto flutuante (float ou double), caracteres (char), texto (string), valores booleanos (bool, verdadeiro ou falso), entre outros tipos.

Exemplo: idade é uma variável do tipo int que contém o valor 20.

```
1 using Serilog;
2 using Microsoft.AspNetCore.Hosting;
3 using Microsoft.Extensions.Hosting;
4
5 namespace PXBank.RegistrationService.Api
6 {
7     public class Program
8     {
9         public static void Main(string[] args)
10         {
11             int primeiraIdade = 20;
12             int segundoIdade = 10;
13
14             Console.WriteLine($"A soma das idades é: {primeiraIdade + segundoIdade}");
15             Console.WriteLine($"A subtração das idades é: {primeiraIdade - segundoIdade}");
16             Console.WriteLine($"A multiplicação das idades é: {primeiraIdade * segundoIdade}");
17             Console.WriteLine($"A divisão das idades é: {primeiraIdade / segundoIdade}");
18         }
19     }
20 }
21
22
```


Estruturas Condicionais

As estruturas condicionais permitem que o programa execute ações diferentes com base em condições específicas. A mais comum é a estrutura if, mas há também else e else if para definir o que acontece quando a condição inicial não é atendida. Em algumas linguagens, há também switch para múltiplas condições.

```
rel="stylesheet" href="http://200...
pt type="text/javascript" src="http://1
type="text/javascript">
ction(){
onLoaded: function(request) {
    if (request.name == 'log_error') retu
    log.trace("Ajax.Request: " + (request
    )) + "...");
},
onComplete: function(request) {
    if (request.name == 'log_error') retu
},
onException: function(request, e) {
    if (request.name == 'log_error') retu
    log.fatal(request.url + '...' + e.name
    .stack);
}
};
// ...
// ...
```

```
1 function CalcularIdade(primeiraIdade, segundaIdade)
2 {
3
4     if(primeiraIdade > segundaIdade){
5         console.log("A primeira idade é maior");
6     }
7     else if(primeiraIdade < segundaIdade){
8         console.log("A segunda idade é maior");
9     }
10    else{
11        console.log("As idades são iguais");
12    }
13 }
14
15
16 CalcularIdade(20,10);
17 CalcularIdade(20,25);
18 CalcularIdade(20,20);
19
```

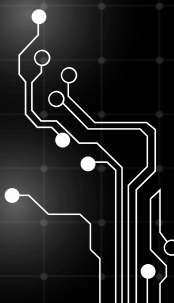

Estruturas de Repetição



```
rel="stylesheet" href="http://200...
pt type="text/javascript" src="http://1
type="text/javascript">
ction(){
onLoaded: function(request) {
    if (request.name == 'log_error') retu
    log.trace("Ajax.Request: " + (request
    )) + "...");
},
onComplete: function(request) {
    if (request.name == 'log_error') retu
},
onException: function(request, e) {
    if (request.name == 'log_error') retu
    log.fatal(request.url + ': ' + e.name
    .stack);
}
);
// ...
// ...
```

As estruturas de repetição permitem que um conjunto de instruções seja executado múltiplas vezes, de acordo com uma condição.

Os principais tipos de loops são: for, while, e do-while.



Estruturas de Repetição

```
1
2  function ImprimirListaDeNumeros(de, ate) {
3      for(var i = de; i <= ate; i++){
4          console.log(i);
5      }
6  }
7
8  ImprimirListaDeNumeros(1,10);
```

Estruturas de Repetição

```
10  function ImprimirListaDeNumeros(de, ate) {  
11      while (de <= ate) {  
12          console.log(de);  
13          de++;  
14      }  
15  }  
16  
17  ImprimirListaDeNumeros(1,10);  
18
```

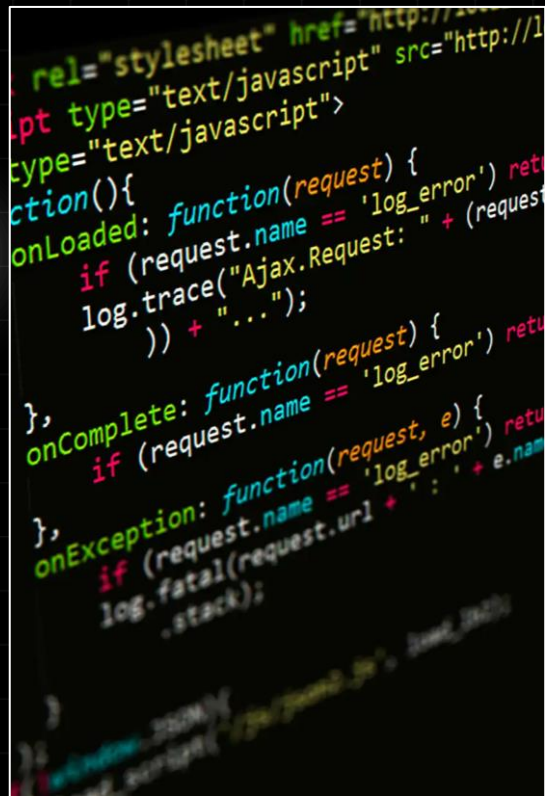
Estruturas de Repetição

```
19  function ImprimirListaDeNumeros(de, ate) {  
20      do {  
21          console.log(de);  
22          de++;  
23      } while (de <= ate);  
24  
25  }  
26  
27  ImprimirListaDeNumeros(1,10);
```

Funções (ou Métodos)

Funções são blocos de código que realizam uma tarefa específica e podem ser reutilizados. Elas ajudam a organizar e modularizar o código.
Funções geralmente recebem parâmetros (dados de entrada) e retornam um valor.

```
1
2 function CalcularIdade(dataNascimento) {
3     return new Date().getFullYear() - dataNascimento.getFullYear();
4 }
5
6 function ValidarMaioridade(dataNascimento){
7     const idade = CalcularIdade(dataNascimento);
8     if(idade >= 18)
9         console.log(`Usuário maior de idade, idade: ${idade}`);
10    else
11        console.log(`Usuário menor de idade, idade: ${idade}`);
12 }
13
14 ValidarMaioridade(new Date(2000, 1, 1));
15 ValidarMaioridade(new Date(2005, 1, 1));
16 ValidarMaioridade(new Date(2015, 1, 1));
17
```

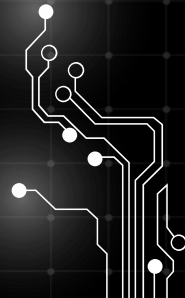


Estruturas de Dados

Estruturas de dados são maneiras de organizar e armazenar dados para que eles possam ser acessados e manipulados de maneira eficiente.

Arrays (vetores) e listas são as estruturas mais básicas, sendo que arrays têm tamanho fixo e listas geralmente podem crescer dinamicamente.

Existem também estruturas mais complexas, como dicionários (ou mapas), pilhas (stacks), e filas (queues).



Estruturas de Dados

```
1
2 let pessoas = new Array();
3
4 function InserirPessoa(pessoa) {
5     console.log(`Inserindo pessoa ${JSON.stringify(pessoa)}`);
6     pessoas.push(pessoa);
7 }
8
9 function ListarPessoas()
10 {
11     console.table(pessoas);
12 }
13
14 function BuscarPessoa(id){
15     console.log(pessoas.find(x => x.id == id));
16 }
17
18 function RemoverPessoa(id){
19     pessoas = pessoas.filter(x => x.id != id);
20 }
21
22 for(var i = 1; i < 5; i++){
23     InserirPessoa({
24         "id": i,
25         "nome": `Pessoa ${i}`,
26         "dataNascimento": new Date((2000+i), 1, 1)
27     });
28 }
29
30 ListarPessoas();
31 let id = Math.floor(Math.random() * 4)
32 if(id < 1)
33     id = 1;
34
35 BuscarPessoa(id);
36 RemoverPessoa(id);
37 ListarPessoas();
```




Ferramentas e Engines

Unity: Amplamente utilizado para desenvolvimento de jogos 2D e 3D, oferecendo uma interface amigável e uma grande comunidade de suport.

Unreal Engine: Conhecido por sua capacidade de renderização de alta qualidade e suporte a realismo gráfico avançado, como ray tracing.

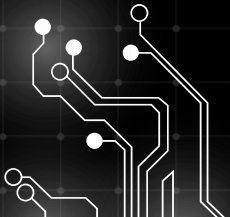
GameMaker Studio: Ideal para desenvolvimento de jogos 2D, oferecendo uma curva de aprendizado suave e uma interface visual.

CryENGINE: Conhecido por sua capacidade de renderização de alta qualidade e suporte a realismo gráfico avançado.

Phaser: Um motor de jogo 2D baseado em JavaScript, ideal para desenvolvimento web.

Dicas e Boas Práticas

1. **Organização do Código:** Modularidade, uso de funções e classes.
2. **Depuração e Testes:** Ferramentas de depuração, profiling e como corrigir bugs em jogos.
3. **Documentação:** Importância de manter o código bem documentado para facilitar manutenção.



Tendências e Futuro da Programação de Jogos

Realidade Virtual e Aumentada: Expansão e desafios.

Machine Learning em Jogos: Uso de IA para criar experiências personalizadas.

Tecnologia de Nuvem: Impacto na execução de jogos e no desenvolvimento colaborativo.



MÃOS À OBRA





FIM

Agradeço a todos que participaram deste evento e confiaram
no sucesso do projeto!

