

Aula 2 - Mais conceitos de POO e introdução à lógica de jogos

Pedro Lamkowski

LTIA

Dead Battery Studio

Conceitos em classes de objetos

Modificadores de acesso

```
public int num;  
public ClasseExemplo objExePublico;  
private ClasseExemplo objExePrivado;  
public void MetodoExemplo();
```

Modificadores de acesso indicam se propriedades ou métodos podem ser acessados fora de seu escopo.

Modificadores de acesso

```
class ClassePublica{ // o seu modificador padrão é
'internal'.
    private void Exemplo(int a){
        // ...
    }
    public int ExemploPublic(int a){
        Exemplo(a);
        // ...
        return 1;
    }
    public ClassePublica() {}
}

class Classe1{
    ClassePublica objDeClassePublica = new ClassePublica();
}
```

```
class Classe1{
    ClassePublica objDeClassePublica = new ClassePublica();
    // objDeClassePublica.Exemplo(10);
    // ^ não funciona pois Exemplo() é private.
    int a = objDeClassePublica.ExemploPublic(10);
}
```

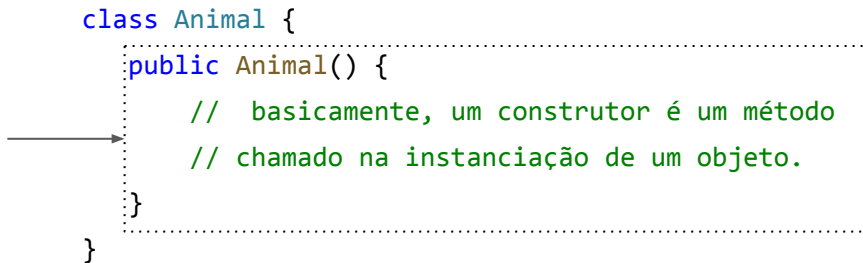
Modificadores de acesso

<i>private</i>	O tipo ou membro só pode ser acessado por código que esteja na mesma classe ou struct.
<i>public</i>	Acesso no mesmo assembly ou em outro assembly que o referencia.
<i>protected</i>	O tipo ou membro só pode ser acessado por código que esteja na mesma classe ou struct, ou por qualquer classe que seja derivada.
<i>internal</i>	Acesso somente no mesmo assembly.

Construtor

Quando uma classe ou struct é criada, seu construtor é chamado. Eles possuem o mesmo nome da classe/struct e geralmente servem para inicializar valores.

Construtor
sempre possui o
modificador de
acesso 'public'.



```
class Animal {  
    public Animal() {  
        // basicamente, um construtor é um método  
        // chamado na instânciação de um objeto.  
    }  
}
```

Herança

```
class Animal {  
    protected int idade;  
    public string nome;  
    public Animal() { }  
}  
  
class Mamifero : Animal {  
    public Mamifero(string nome){  
        idade = 0;  
        this.nome = nome;  
    }  
}
```

Mamífero é derivada de Animal e possui acesso aos campos de Animal.

A palavra chave 'this' referência o escopo da classe. Assim, a linha

```
this.nome = nome;
```

atribui o valor recebido como parâmetro no construtor ao membro `nome` da classe Mamífero que por sua vez vem da classe Animal.

Overload

Sobrecarga. É possível definir métodos que possuem o mesmo nome, mas que possuem parâmetros de entrada ou saída diferentes.

```
float[] CriaVetor(float x, float y) {  
    return new float[3] {x,y,0};  
    // Esse código poderia ser:  
    // return CriaVetor(x,y,0);  
}  
  
float[] CriaVetor(float x, float y, float z){  
    // Para iniciar um vetor:  
    // float[] a = new float[3];  
    // Acesso: a[0] = 10f;  
    return new float[3] {x,y,z};  
}
```


Interface

Interfaces possuem declarações que classes ou structs podem implementar.

Não definem o comportamento, mas obrigam que as classes/structs que usam da interface tenham que definir o comportamento.

```
interface IDesenhavel {  
    void Desenhar();  
}  
  
class Lousa : IDesenhavel {  
    public Lousa() { }  
    void Desenhar() {  
        // ... Implementação de desenhar ...  
    }  
}
```

Modificador static

Algo static pertence ao tipo e não ao objeto. Uma classe static não pode ser instanciada e seus métodos/propriedades também precisam ser static.

Membros (métodos, propriedades) static são constantes a todas as instâncias de uma classe.

Modificador static

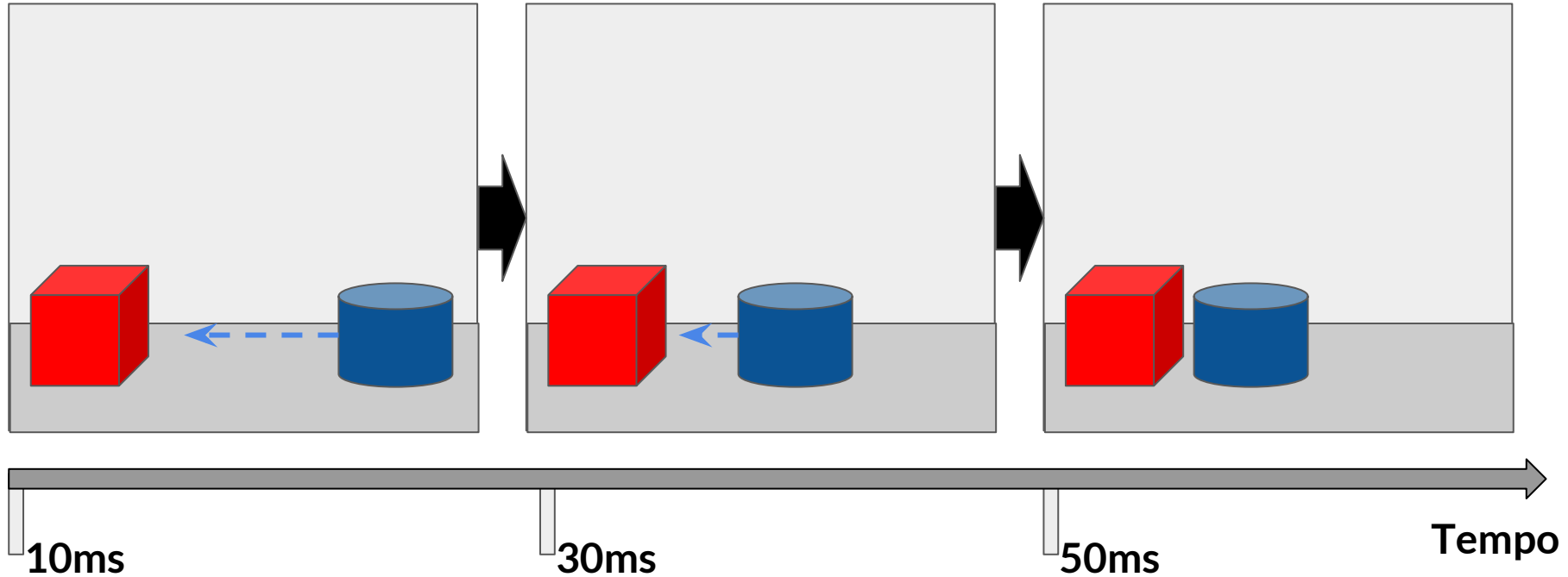
```
class Animal {  
    public string nome;  
    public int idade;  
    public static int quantAnimaisNoZoo;  
    public Animal(string nome, int idade) {  
        this.nome = nome;  
        this.idade = idade;  
    }  
}
```

```
class Zoo {  
    Animal[] animais;  
  
    public Zoo(int numeroDeAnimais) {  
        animais = new Animal[numeroDeAnimais];  
    }  
  
    void AdicionarAnimal(string nome, int idade) {  
        a[Animal.quantAnimaisNoZoo] = new Animal(nome,idade);  
        Animal.quantAnimaisNoZoo += 1;  
    }  
}
```

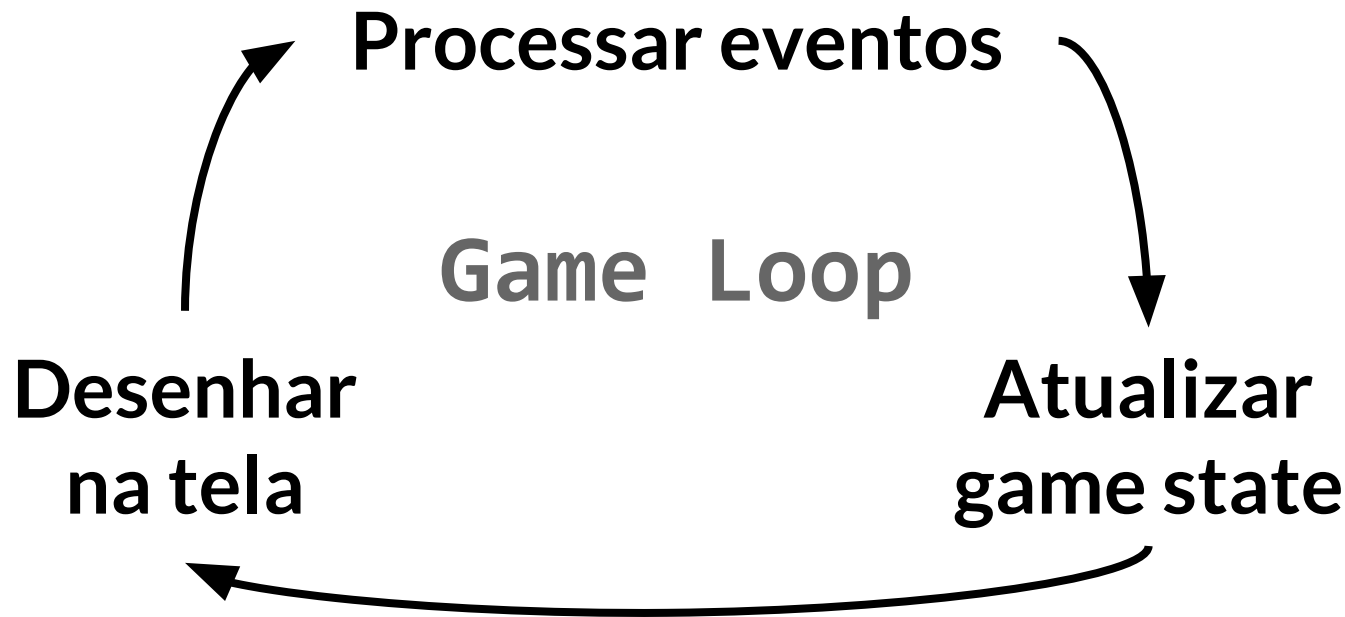
O que é realmente um jogo para o computador?



Entendendo como um jogo funciona



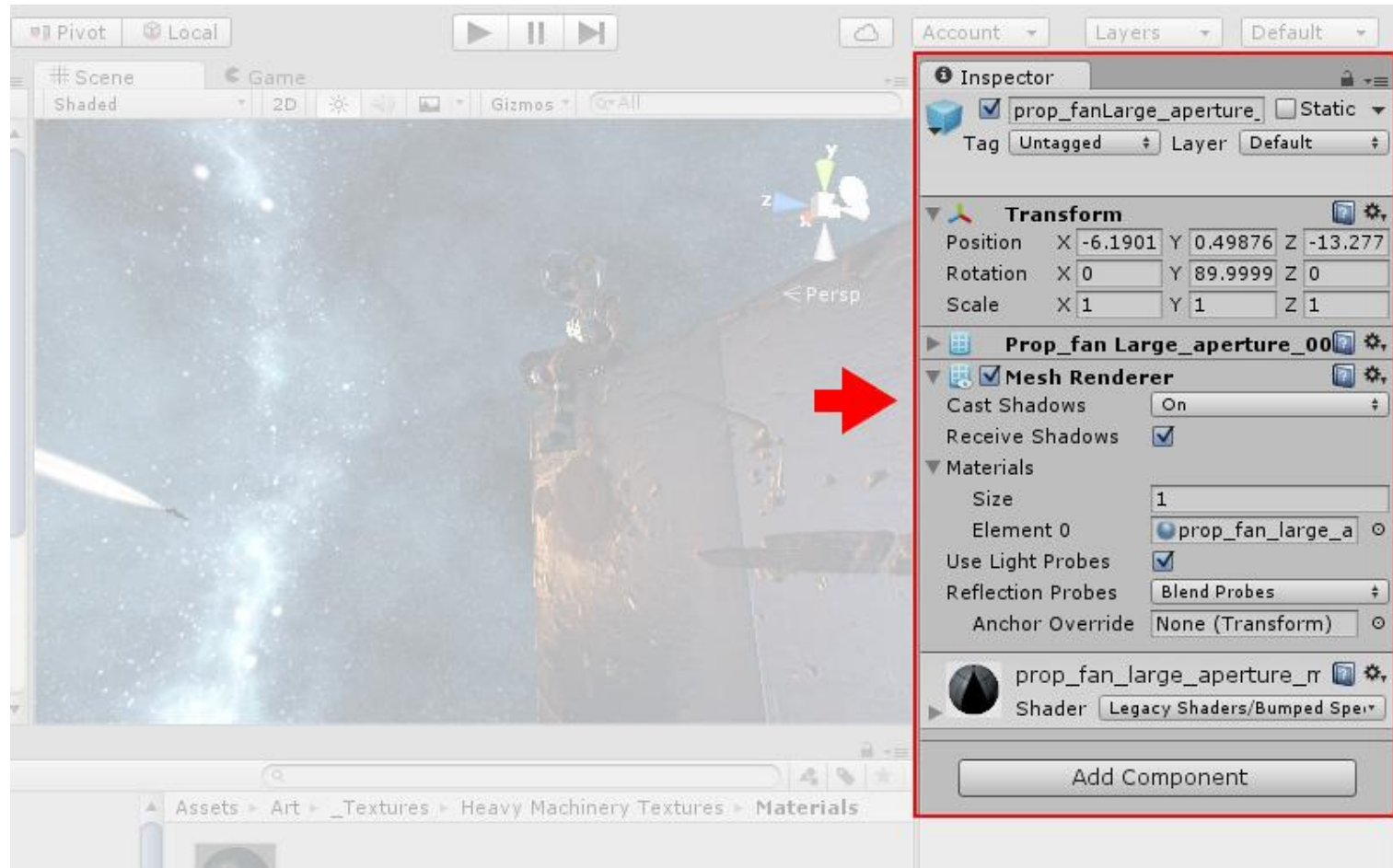
Entendendo como um jogo funciona



Unity - Componente base

Todo objeto que é visto em um jogo feito em unity deriva de uma classe base chamada MonoBehaviour. Convencionalmente, qualquer objeto em uma cena é um GameObject.

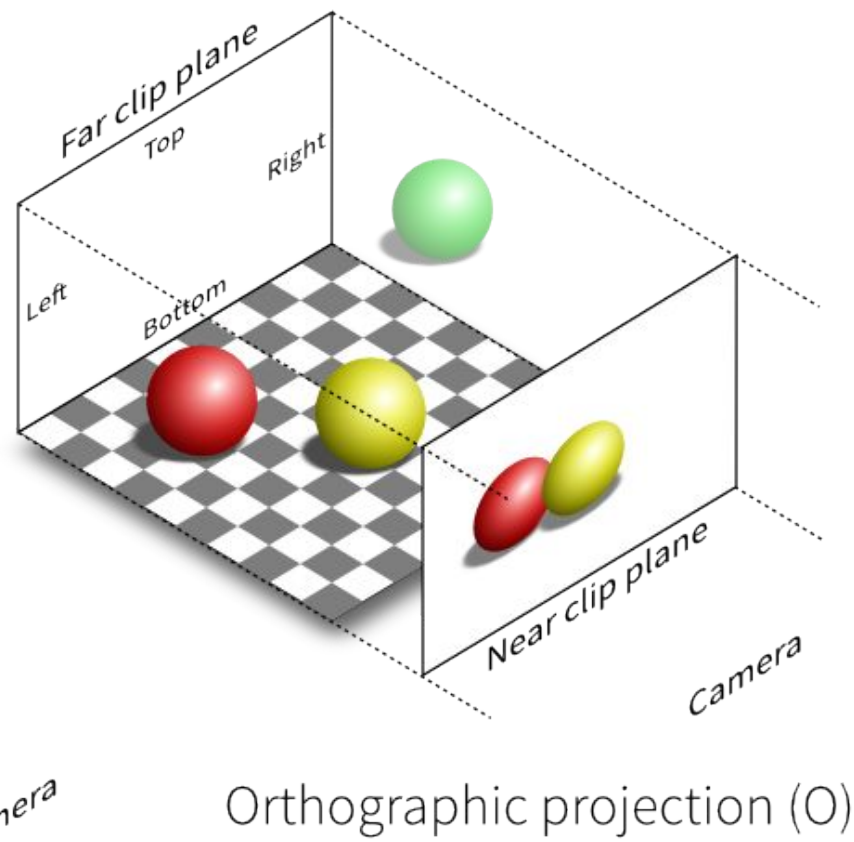
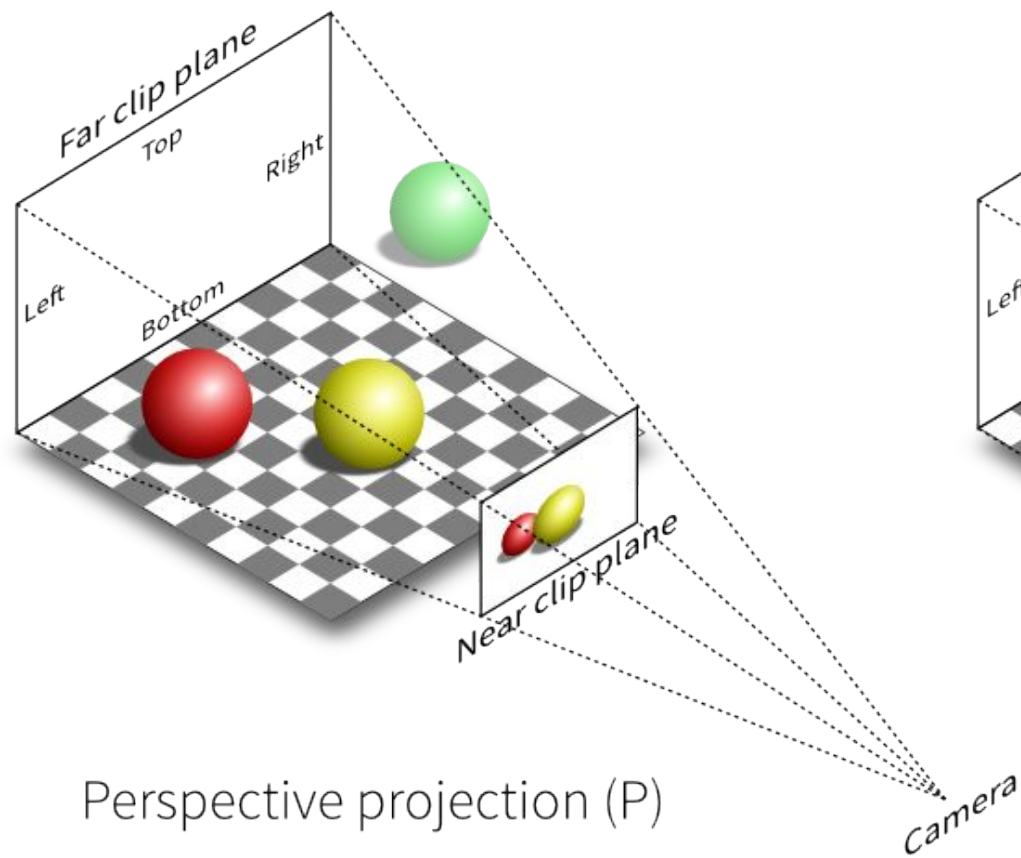
O inspector mostra todos os componentes que um GameObject possui. Cada componente na realidade também é uma classe.



Janela inspector - Documentação Unity

2D vs 3D

- Para a Unity, essencialmente tudo é 3D.
- Desenvolver em 2D é uma mudança de câmera e perspectiva.
- Mesmo em 2D, lembre-se que, espacialmente, estamos trabalhando em 3 dimensões.

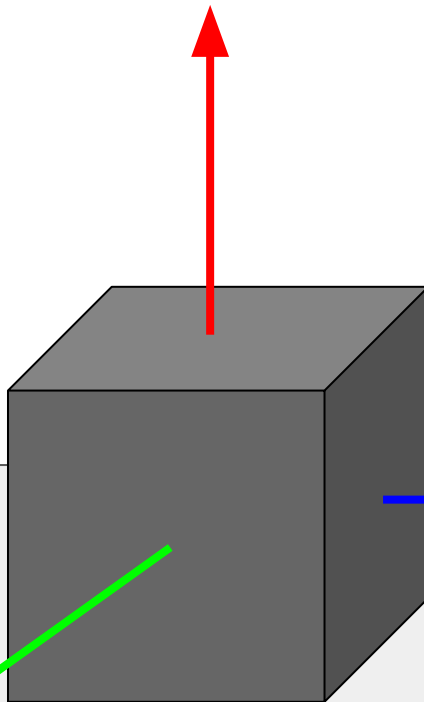


Espaços
globais

Espaços
Locais

Rotação

Parentalidade



Off-topic

Sugestão de vídeo:

(<https://www.youtube.com/watch?v=L3wScHE28K8>)

Qual a pessoa que mais me motiva a acreditar que os jogos importam?

Esse cara →



Yoko Taro, diretor (Drakengard, NieR, NieR:Automata)