

# Linguagem Lua



# Origem

- Linha do Tempo (Versões):



- Criadores:

~ Desenvolvida pelo Instituto Tecgraf da PUC-Rio

- Influências:

~ Inspirado por Lisp, Scheme, Python e Pascal

# Classificação

- Linguagem de Script
- Interpretada e de Tipagem Dinâmica:
  - ~ Lua oferece um ambiente de interpretação REPL e possui tipagem dinâmica

# Metatabelas e Metamétodos

- Exemplo de uso: Vetores 3D e suas aplicações em computação gráfica.
- Tabelas e Orientação a Objetos
  - ~ Metatabelas controlam tabelas utilizando de metamétodos (permite a emulação de orientação a objetos).

# Exemplo em Lua

```
Vector3.__call = function(t,x,y,z)
    return Vector3.New(x,y,z)
end

function Vector3.New(x, y, z)
    local v = {x = x or 0, y = y or 0, z = z or 0}...
    setmetatable(v, Vector3)
    return v
end

...
function Vector3:Set(x,y,z)
    self.x = x or 0
    self.y = y or 0
    self.z = z or 0
end

function Vector3:Get()
    return self.x, self.y, self.z
end

function Vector3:Clone()
    return Vector3.New(self.x, self.y, self.z)
end
```

# Exemplo em C#

```
public partial struct Vector3 : IEquatable<Vector3>, IFormattable
{
    // *Undocumented*
    public const float kEpsilon = 0.00001F;
    // *Undocumented*
    public const float kEpsilonNormalSqrt = 1e-15F;

    // X component of the vector.
    public float x;
    // Y component of the vector.
    public float y;
    // Z component of the vector.
    public float z;
```

```
Vector3.__div = function(va, d)
    return Vector3.New(va.x / d, va.y / d, va.z / d)
end

Vector3.__mul = function(va, d)
    if type(d) == "number" then
        return Vector3.New(va.x * d, va.y * d, va.z * d)
    else
        local vec = va:Clone()
        vec:MulQuat(d)
        return vec
    end
end

Vector3.__add = function(va, vb)
    return Vector3.New(va.x + vb.x, va.y + vb.y, va.z + vb.z)
end


Vector3.__sub = function(va, vb)
    return Vector3.New(va.x - vb.x, va.y - vb.y, va.z - vb.z)
end

Vector3.__unm = function(va)
    return Vector3.New(-va.x, -va.y, -va.z)
end
```

```
[MethodImpl(MethodImplOptions.AggressiveInlining)]
public static Vector3 operator+(Vector3 a, Vector3 b) { return new Vector3(a.x + b.x, a.y + b.y, a.z + b.z); }
// Subtracts one vector from another.
[MethodImpl(MethodImplOptions.AggressiveInlining)]
public static Vector3 operator-(Vector3 a, Vector3 b) { return new Vector3(a.x - b.x, a.y - b.y, a.z - b.z); }
// Negates a vector.
[MethodImpl(MethodImplOptions.AggressiveInlining)]
public static Vector3 operator-(Vector3 a) { return new Vector3(-a.x, -a.y, -a.z); }
// Multiplies a vector by a number.
[MethodImpl(MethodImplOptions.AggressiveInlining)]
public static Vector3 operator*(Vector3 a, float d) { return new Vector3(a.x * d, a.y * d, a.z * d); }
// Multiplies a vector by a number.
[MethodImpl(MethodImplOptions.AggressiveInlining)]
public static Vector3 operator*(float d, Vector3 a) { return new Vector3(a.x * d, a.y * d, a.z * d); }
// Divides a vector by a number.
[MethodImpl(MethodImplOptions.AggressiveInlining)]
public static Vector3 operator/(Vector3 a, float d) { return new Vector3(a.x / d, a.y / d, a.z / d); }

// Returns true if the vectors are equal.
public static bool operator==(Vector3 lhs, Vector3 rhs)
{
    // Returns false in the presence of NaN values.
    float diff_x = lhs.x - rhs.x;
    float diff_y = lhs.y - rhs.y;
    float diff_z = lhs.z - rhs.z;
    float sqrmag = diff_x * diff_x + diff_y * diff_y + diff_z * diff_z;
    return sqrmag < kEpsilon * kEpsilon;
}
```





Na linguagem Lua é utilizado metatabelas para substituir o uso de Orientação a Objetos no que tange o uso de sobrecarga de operadores.

Na linguagem C# para executarmos a mesma função utilizamos de métodos dentro de uma classe que faz essa sobrecarga.

# Fontes:

<http://webserver2.tecgraf.puc-rio.br/lua/local/docs.html>

<http://webserver2.tecgraf.puc-rio.br/lua/local/manual/5.4/>

<https://www.lua.org/doc/hopl.pdf>

Exemplo de Vetor em C#:

<https://github.com/Unity-Technologies/UnityCsReference/blob/master/Runtime/Export/Math/Vector3.cs>