

# Computação Gráfica

Aula 1: Introdução

# Professores

## Luciano P. Soares

Doutor em Engenharia de Computação  
pela Escola Politécnica da USP



## Fabio Orfali

Doutor em Ensino de Matemática e Ciências  
na Faculdade de Educação da USP



## Gustavo Braga

Engenheiro de Computação pelo Insper



## Pedro Emil Freme

Game Designer pela Fatec



# Sobre esse curso

Uma ampla visão geral dos principais tópicos e técnicas em computação gráfica: geometria, renderização, cores, texturas, iluminação, animação, imagens, etc.

Aprenda fazendo:

- Diversas atividades em aula para verificar, desenvolver e fixar suas habilidades;
- Desenvolvimento de projetos para colocar todo esse conhecimento para funcionar.

# Horário de Atendimento

Quartas-feiras das 18:00 as 19:30

Remota / Laboratório de Realidade Virtual

# Computação Gráfica

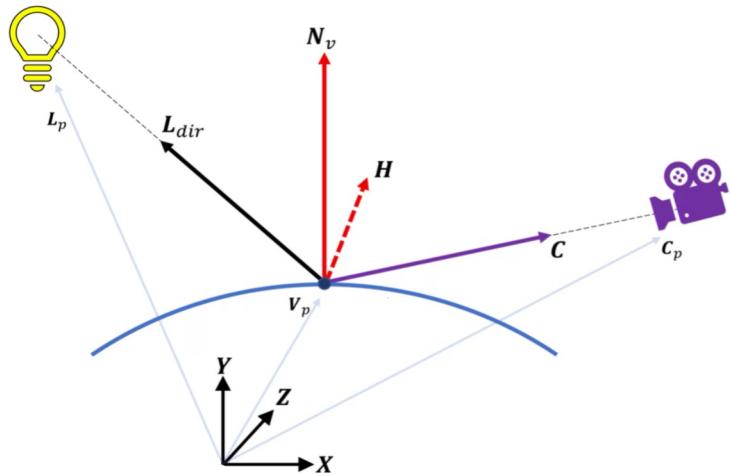
Essa primeira parte da disciplina é baseada no curso de Berkeley CS184 e Stanford CS248. Qualquer semelhança não é mera coincidência.

Assim, já agradeço a todos pelos materiais encontrados na internet: Pat Hanrahan, Ren Ng, Kayvon Fatahalian, Keenan Crane, Mark Pauly, Steve Marschner, Kayvon Fatahalian, dentre outros.



# Vai ter matemática nesse curso ?

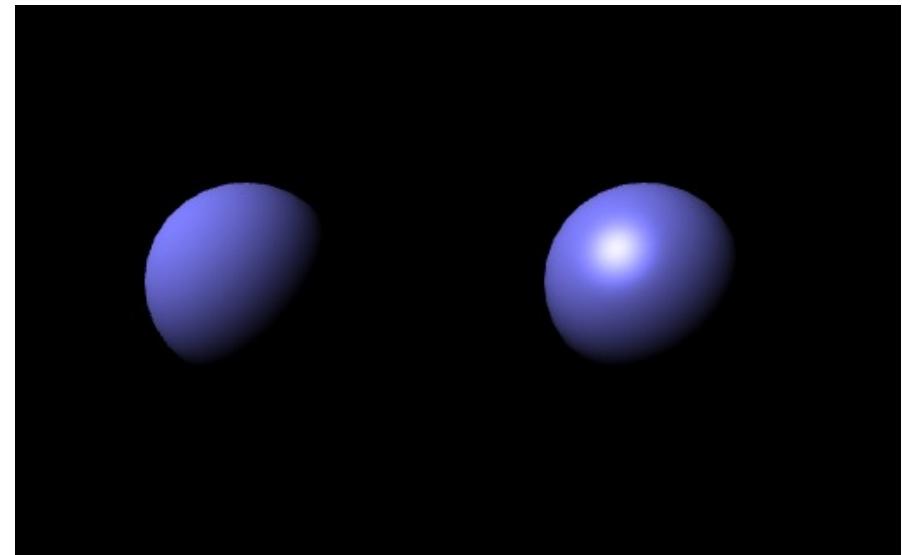
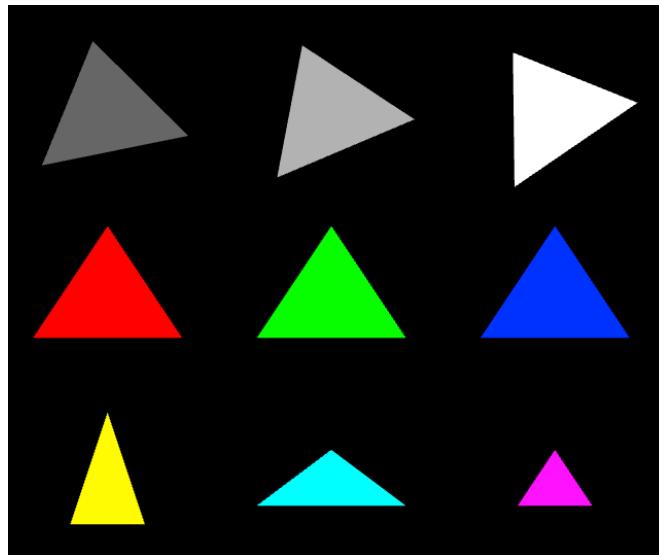
Sim. Neste curso iremos bem a fundo no funcionamento dos algoritmos que geram os gráficos, e vocês verão na prática como o conhecimento matemático é importante.



Avengers: Age of Ultron

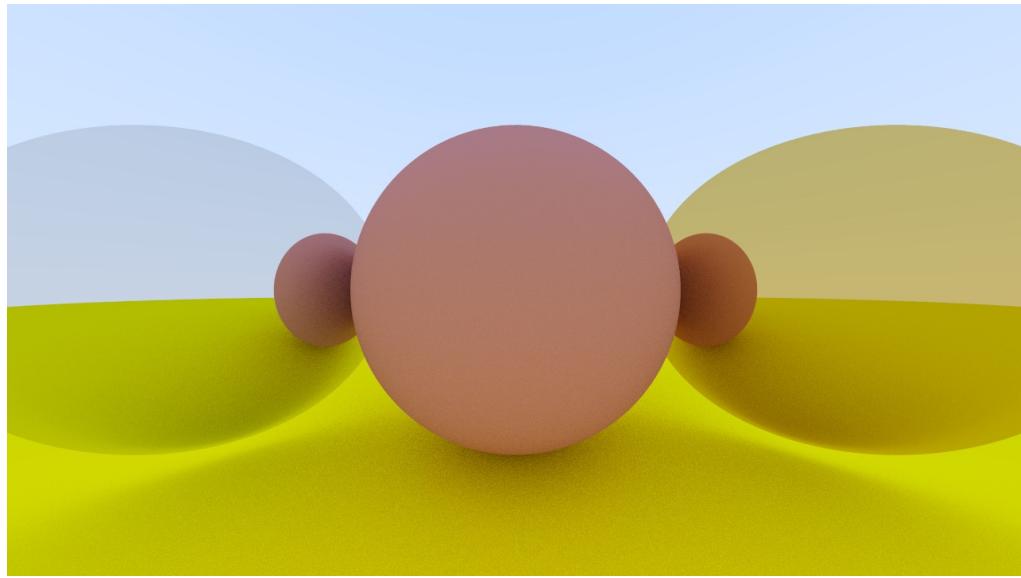
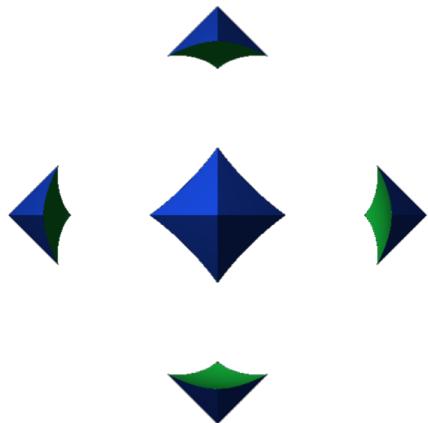
# E os resultados

Começaremos com resultados bem simples.  
Triângulos, Esferas,...



# Podemos ir mais longe?

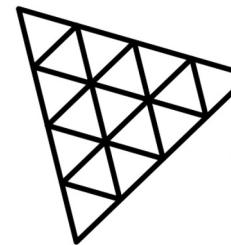
Na segunda parte do curso trabalharemos com ferramentas mais avançadas usando por exemplo *shaders*.



# O foco não é mergulhar nas APIs Gráficas



Microsoft®  
**DirectX®**



# O que é Computação Gráfica?

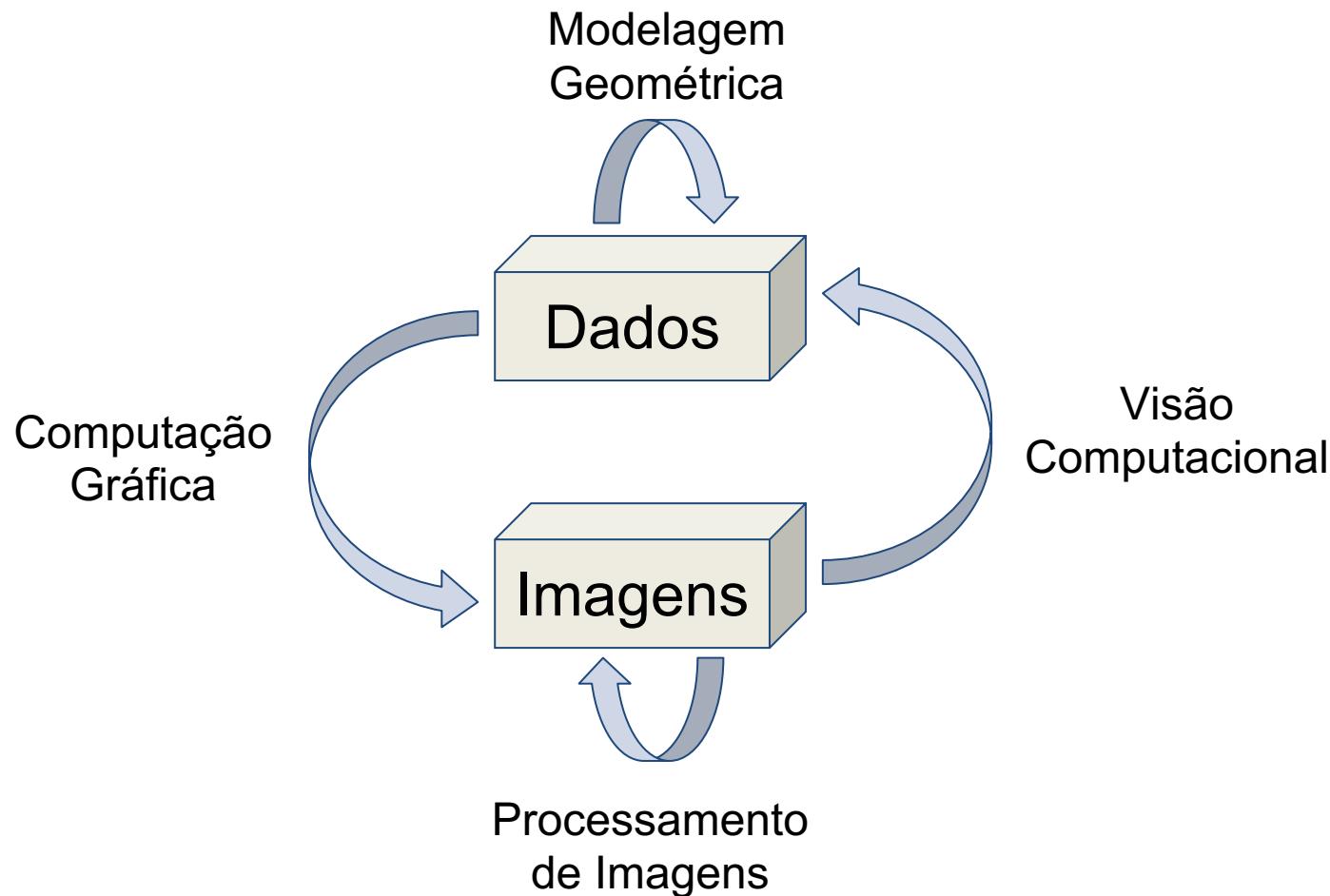
## Computação Gráfica

O uso de computadores para gerar e manipular informações visuais.



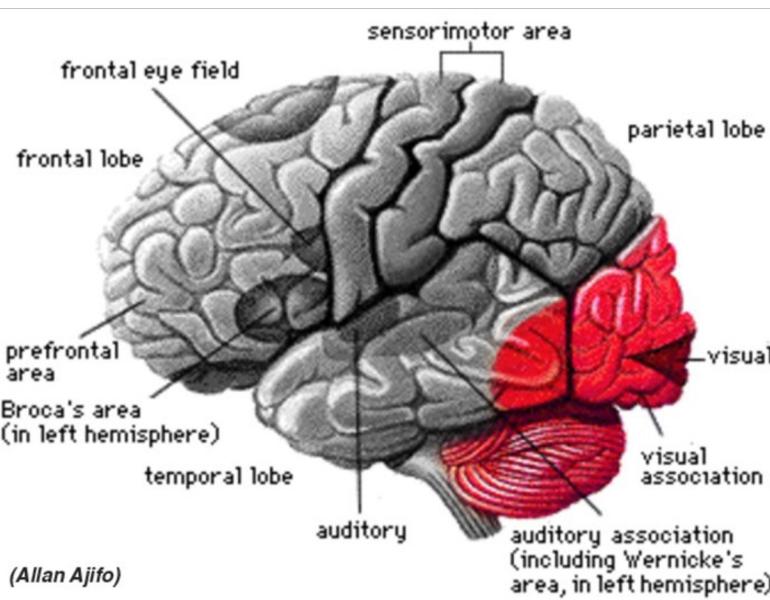
Lord of the Rings

# Onde está a Computação Gráfica?



# Por que Informação Visual

Cerca de 30% do cérebro é dedicado ao processamento visual...



**Os olhos são a porta de acesso ao cérebro com maior largura de banda!**

# Usos da Computação Gráfica

## Filmes



Avengers: Endgame



District 9



The Lion King

# SIGGRAPH 2021 Computer Animation Festival



<https://www.youtube.com/watch?v=3oXFgQTFx1M>

# Usos da Computação Gráfica

## Jogos Digitais



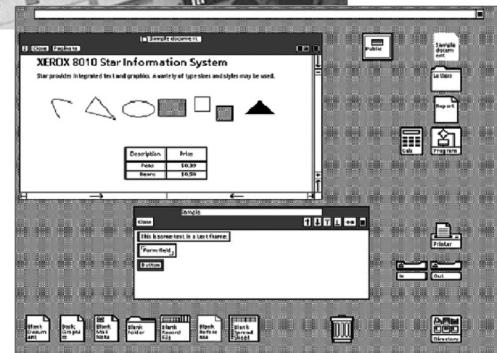
Uncharted 4 (Naughty Dog, 2016)



Assassin's Creed Odyssey (Ubisoft 2018)

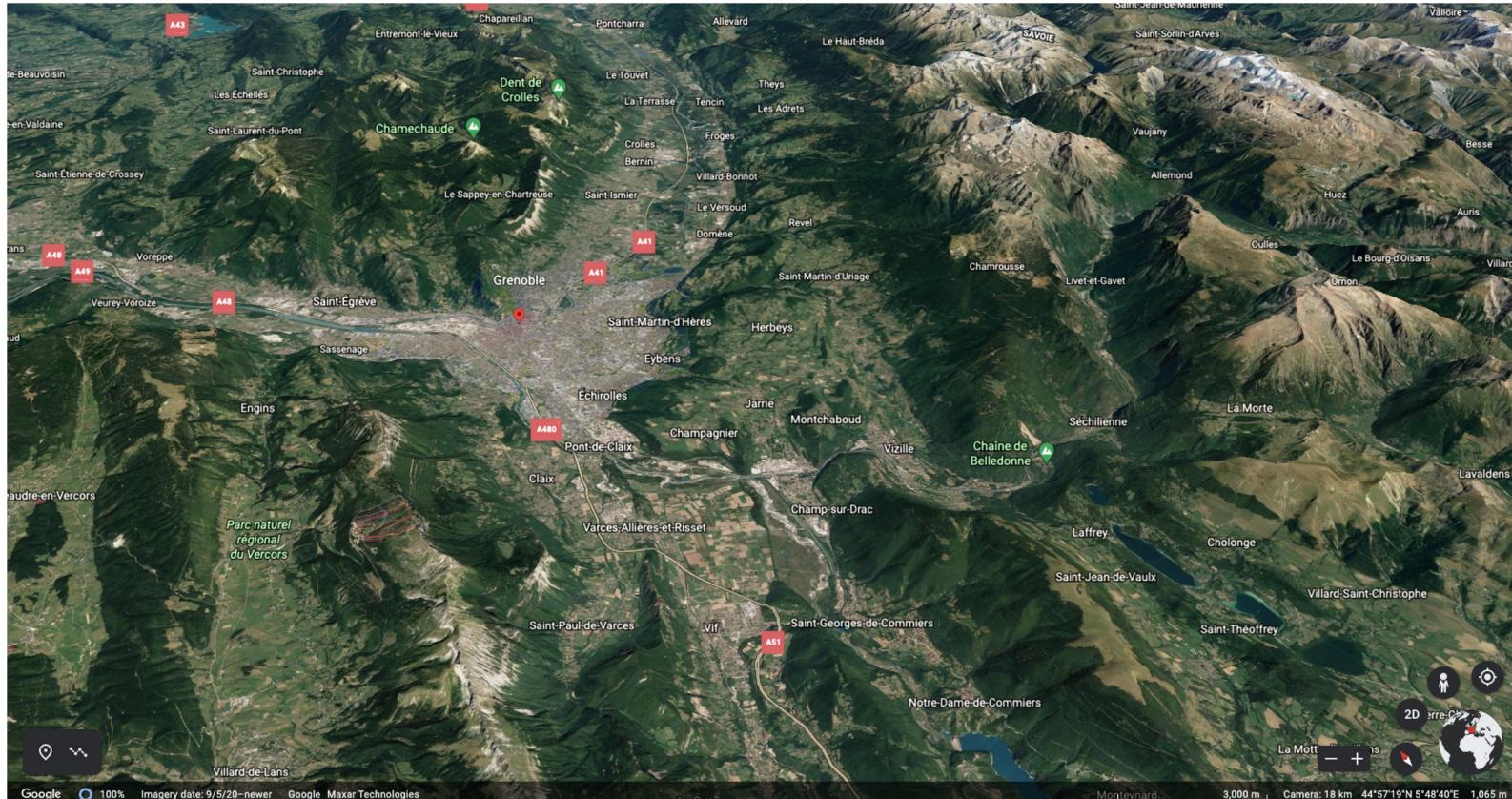
# Usos da Computação Gráfica

## Interface Gráfica / Graphical User Interface



# Usos da Computação Gráfica

## Imagen e terrenos para mapas



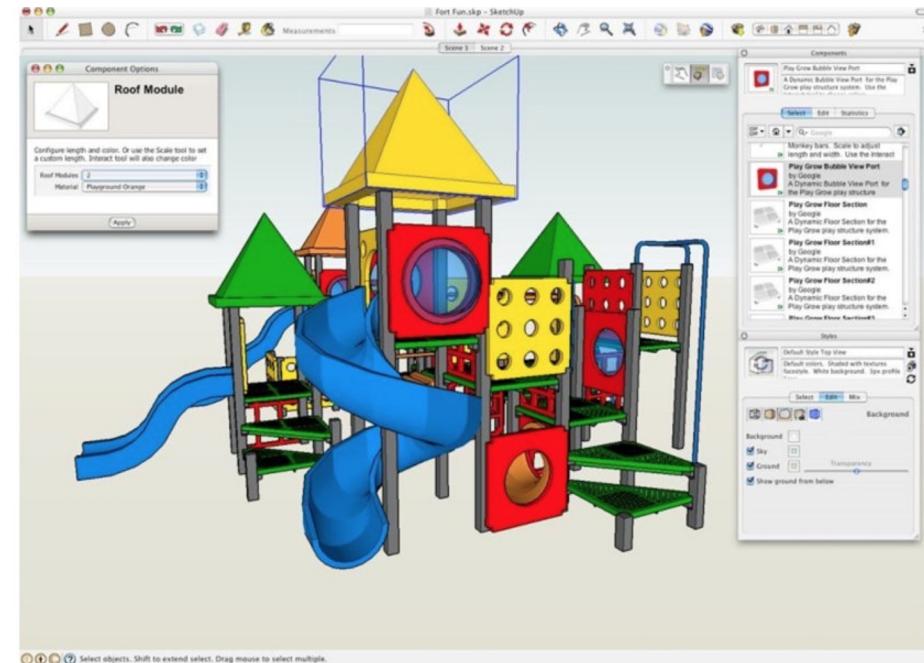
Mapas, fotos de satélite, imagens de ruas

# Usos da Computação Gráfica

## CAD (Computer-aided design)



**SolidWorks**



**SketchUp**

Para mecânica, arquitetura, eletrônica, ...

# Usos da Computação Gráfica

## Arquitetura



Bilbao Guggenheim, Frank Gehry



Heydar Aliyev Center, Zaha Hadid Architects

# Usos da Computação Gráfica

## Design de Produtos



Tesla Model X

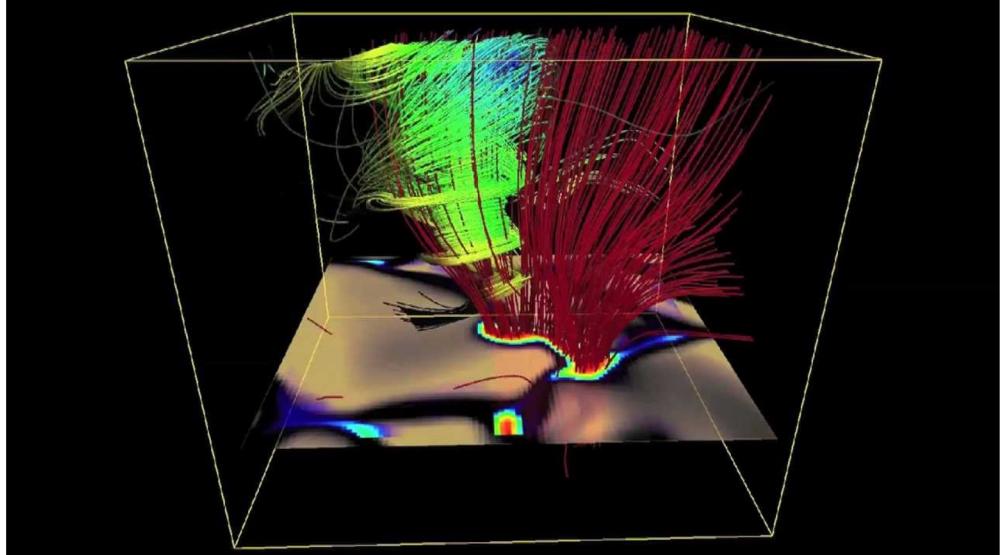
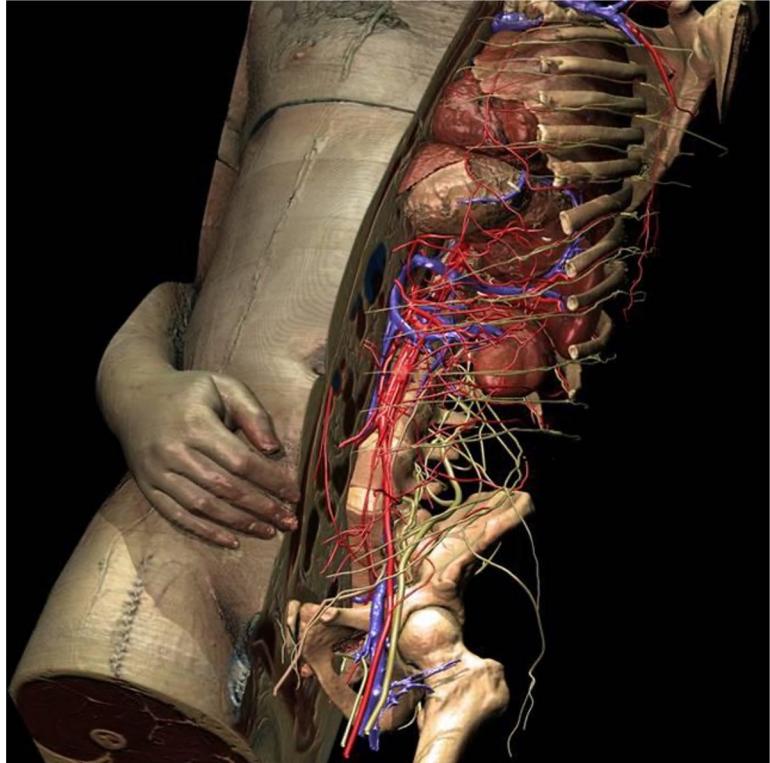


Ikea

75% do catálogo de produtos da Ikea são imagens renderizadas.

# Usos da Computação Gráfica

## Visualização



Científica, Engenharia, Medicina, Jornalismo, ...

# Usos da Computação Gráfica

## Simulações



**Driving simulator**  
**Toyota Higashifuji Technical Center**

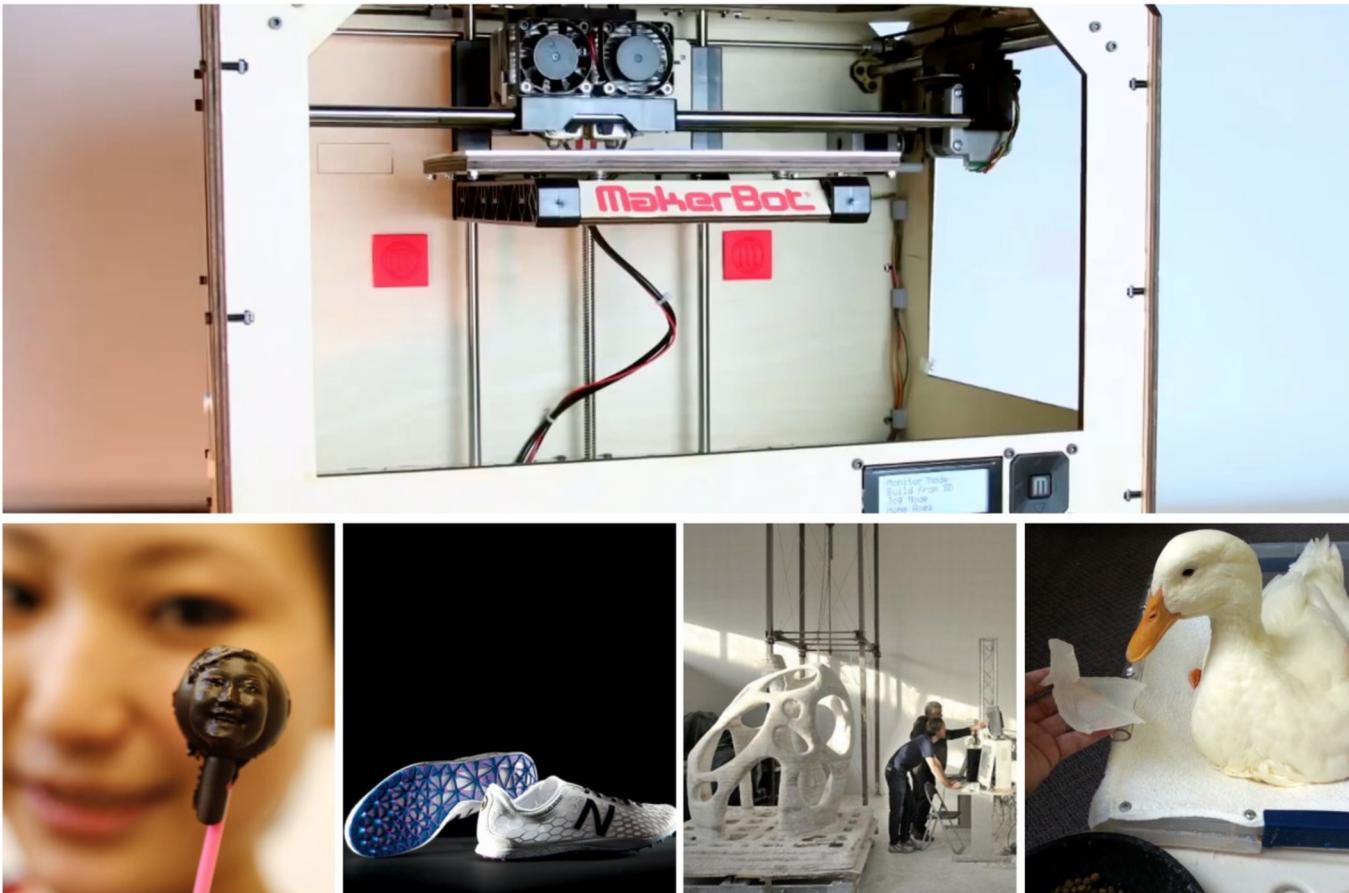


**da Vinci surgical robot**  
**Intuitive Surgical**

Simulador de voo, de direção, de cirurgia.

# Usos da Computação Gráfica

## Impressão 3D



# Fundamentos da Computação Gráfica

Todas essas aplicações demandam!

## Ciência e Matemática

- Física da luz, cores, óptica
- Cálculo de curvas, geometrias, perspectiva
- Amostragens

## Arte e psicologia

- Percepção: cores, movimento, qualidade de imagem
- Arte e Design: composição, forma, iluminação

# Supercomputação (GPUs)



## Theoretical Performance

Pixel Rate: 136.0 GPixel/s

Texture Rate: 420.2 GTexel/s

FP16 (half) performance: 26.90 TFLOPS (2:1)

FP32 (float) performance: 13.45 TFLOPS

FP64 (double) performance: 420.2 GFLOPS (1:32)

## Theoretical Performance

Pixel Rate: 443.5 GPixel/s

Texture Rate: 1,290 GTexel/s

FP16 (half): 82.58 TFLOPS (1:1)

FP32 (float): 82.58 TFLOPS

FP64 (double): 1,290 GFLOPS (1:64)

<https://www.techpowerup.com/gpu-specs/geforce-rtx-2080-ti.c3305>

<https://www.techpowerup.com/gpu-specs/geforce-rtx-4090.c3889>

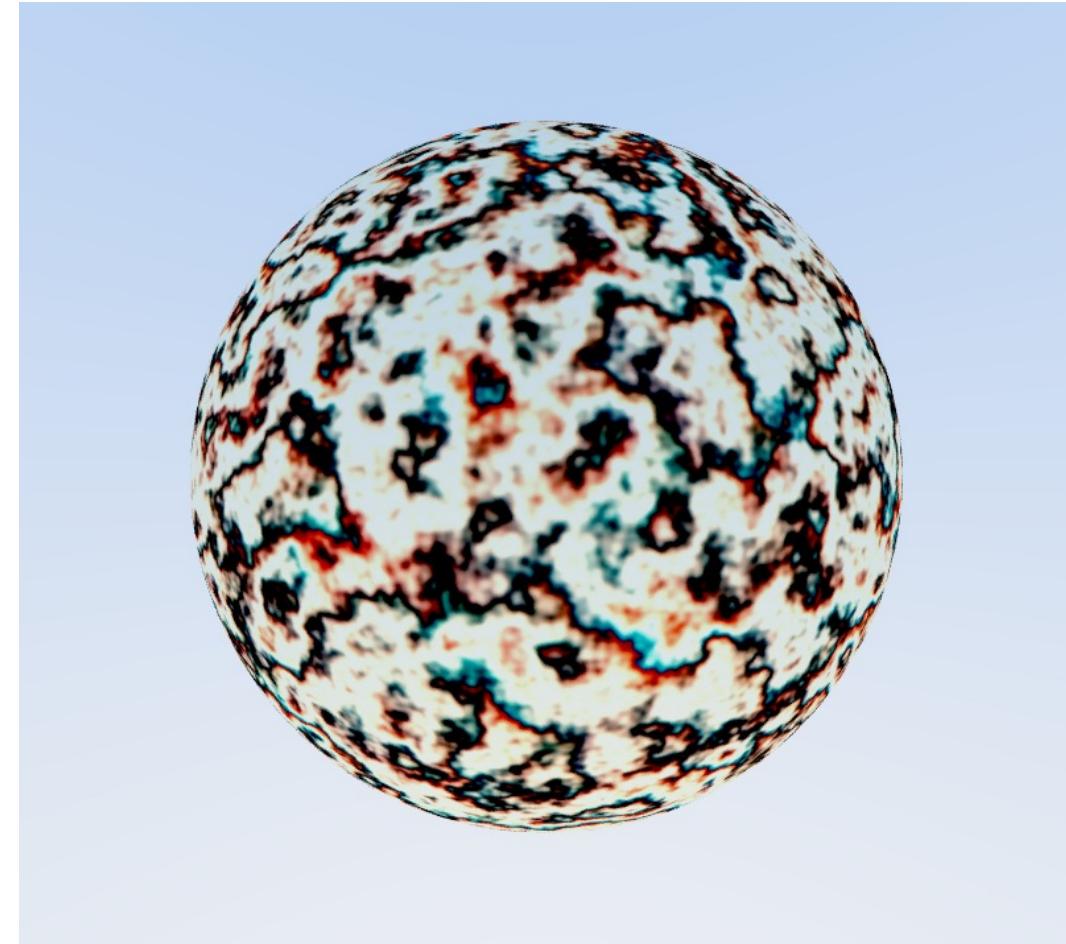
# Objetivos de Aprendizagem

Ao final da disciplina o estudante será capaz de:

- Implementar algoritmos diversos de renderização 3D.
- Desenvolver rotinas gráficas através de técnicas de álgebra linear.
- *Desenvolver shaders* programáveis em bibliotecas gráficas de baixo nível.
- Compreender os diversos elementos das pipelines gráficas.

# Aulas

- 01 - Introdução
- 02 - Desenhando Triângulos e X3D
- 03 - Renderização e Transformações Geométricas
- 04 - Coordenadas Homogêneas e Quatérnios
- 05 - Sistemas de Coordenadas
- 06 - Projeções Perspectiva
- 07 - Revisão 1
- 08 - Grafo de Cena
- 09 - Interpolação em Triângulos
- 10 - Anti-aliasing e Visibilidade
- 11 - Mapeamento
- 12 - Revisão 2
- 13 - Revisão 3
- 14 - Primitivas Geométricas
- 15 - Materiais e Iluminação
- 16 - Curvas e Animações
- 17 - Revisão 4
- 18 - Pipeline Gráfico
- 19 - Shaders
- 20 - Programmable Shaders
- 21 - Signed Distance Function
- 22 - Geometrias
- 23 - Ray Marching
- 24 - Sombras
- 25 - Ray Tracing
- 26 - Normal e Gradiente
- 27 - Aleatoriedade e Ruídos
- 28 - Revisão / Estúdio



# Bibliografia

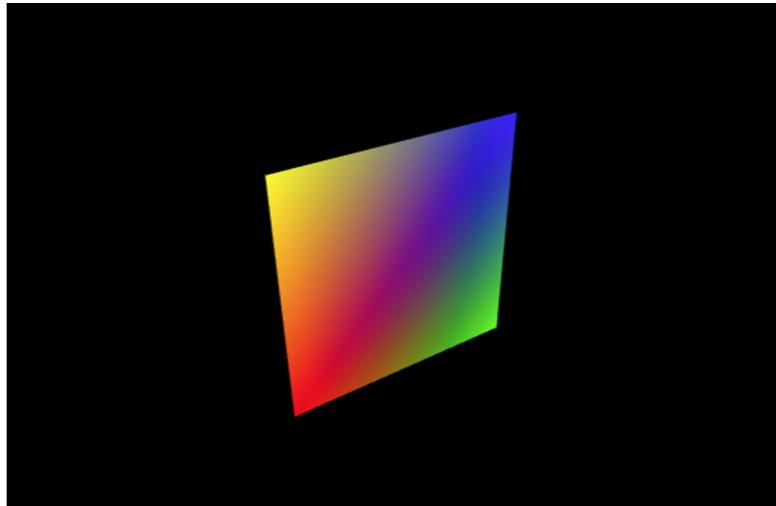
Os documentos passados nas aulas deveriam ser suficiente, mas se desejar, essas são excelentes fontes de informação:

<b>1</b>	HUGHES, John F.; DAM, Andries van; MCGUIRE, Morgan; SKLAR, David F.; FOLEY, James D.; FEINER, Steven K.; AKELEY, Kurt. <b>Computer Graphics: Principles and Practice.</b> 3 <sup>a</sup> Ed. Addison-Wesley Professional; 2013.
<b>2</b>	GREGORY, Jason. <b>Game Engine Architecture.</b> 3 <sup>a</sup> Ed. A K Peters/CRC Press; 2018.
<b>3</b>	LENGYEL, Eric. <b>Mathematics for 3D Game Programming and Computer Graphics.</b> 3 <sup>a</sup> Ed. Cengage Learning PTR; 2011.

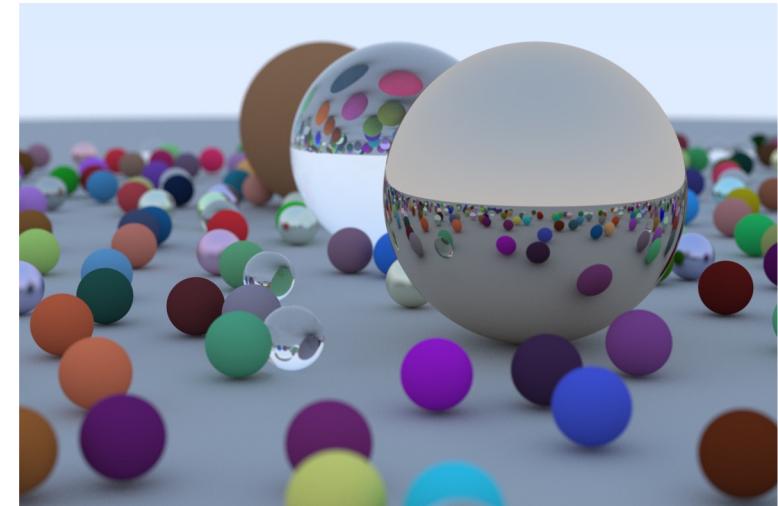
# Avaliação

- Projetos do Curso : Projetos 1 e 2
  - Projetos terão entregas parciais
  - Nota final é a média de todas as entregas parciais

Projeto 1: Rasterizador (CPU)



Projeto 2: Shaders (GPU)



# Política de Atrasos nas Entregas

**As entregas são até 11:59 do dia marcado.**

**Para cada dia atrasado a nota é reduzida em 1 ponto.**

(Isso não significa que todos que entregarem no prazo receberão 10, se o trabalho fosse receber 5 e chegou atrasado um dia, a nota do trabalho vai virar 4)

# Perguntas?



# Computação Gráfica

X3D e Linhas

# Como armazenar os dados da cena?

Nós precisamos de uma forma de armazenar os dados:

- Superfícies
- Materiais
- Animações
- Luzes
- Câmeras



# Extensible Markup Language (XML)

- XML é usado para estruturar dados
- XML é codificado em texto
- XML lembra o HTML e específica do XHTML
- XML usa muito texto para organizar dados
- XML é livre de licença, independente de plataforma e bem suportado

```
<?xml version="1.0" encoding="UTF-8"?>
<texto>
    <frase>hello world</frase>
</texto>
```

**Na prática as pessoas estão migrando de XML para JSON  
(JavaScript Object Notation)**

# XML - Elementos e Atributos

- O XML possui elementos que podem conter outros elementos
- Atributos são valores colocados dentro dos elementos

Elemento de Abertura: Shape  
Elemento de Abertura: TriangleSet  
Elemento Singleton: Coordinate, atributo: point  
Elemento de Fechamento: TriangleSet  
Elemento de Abertura: Appearance  
Elemento Singleton: Material, atributo: diffuseColor  
Elemento de Fechamento: Appearance  
Elemento de Fechamento: Shape

```
<Shape>
--- <TriangleSet>
----- <Coordinate point='-5 1 3 -3 2 1 -5 4 0' />
--- </TriangleSet>
--- <Appearance>
----- <Material diffuseColor='1 0 0' />
--- </Appearance>
</Shape>
```

# Computação Gráfica

X3D



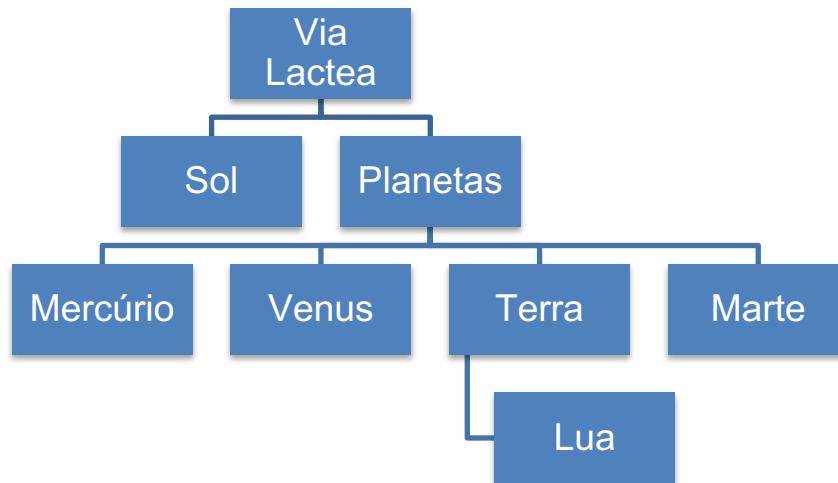
## Formato Universal de Transferência de dados 3D

- Um padrão aberto
- Fácil de entender e modelar
- Portável entre plataformas
- Fácil de ensinar e programar



# Grafo de Cena (Scene Graph)

Estrutura de dados hierárquica de objetos gráficos para uma determinada cena. Objetos são representados como nós de um grafo, para posterior renderização.



Veremos mais sobre Grafo de Cena nas próximas aulas.

# Exemplos de Tipos de Nós

## Geometrias

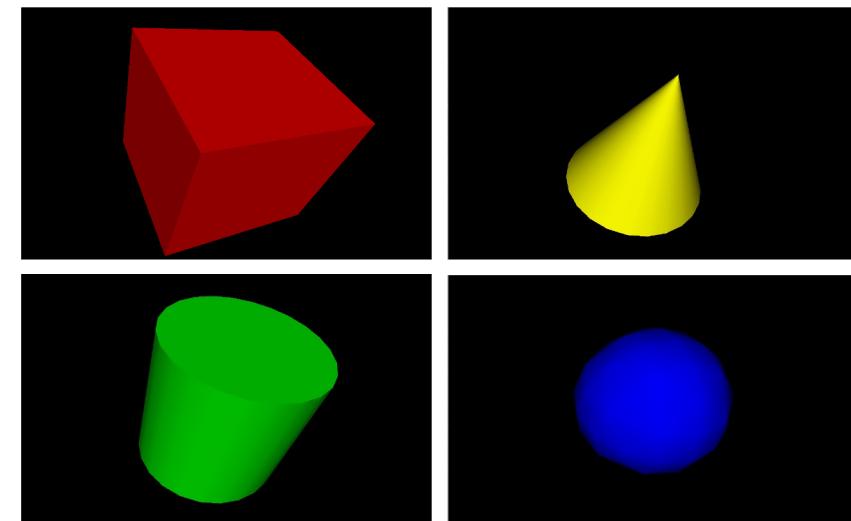
- retas, curvas
- cubos, esferas, cones, cilindros
- malhas de polígonos

## Controle

- Switch>Select
- Group

## Propriedades

- Cores
- Materiais
- Luzes
- Câmera

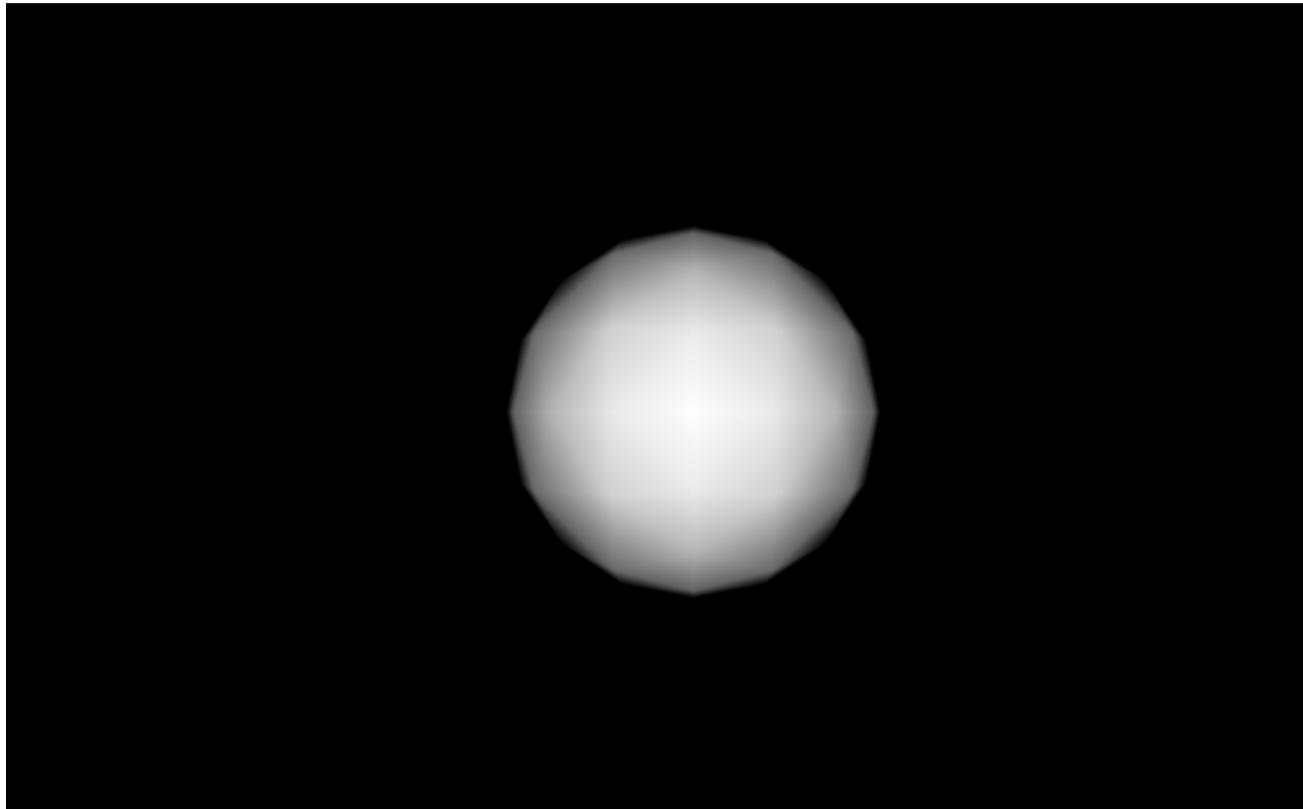


# X3D XML (exemplo)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "http://www.web3d.org/specifications/x3d-3.0.dtd">
<X3D profile="Immersive">
<Scene>
    <NavigationInfo type="ANY"/>
    <Transform>
        <Shape>
            <Appearance>
                <Material diffuseColor="1 1 1"/>
            </Appearance>
            <Sphere/>
        </Shape>
    </Transform>
</Scene>
</X3D>
```

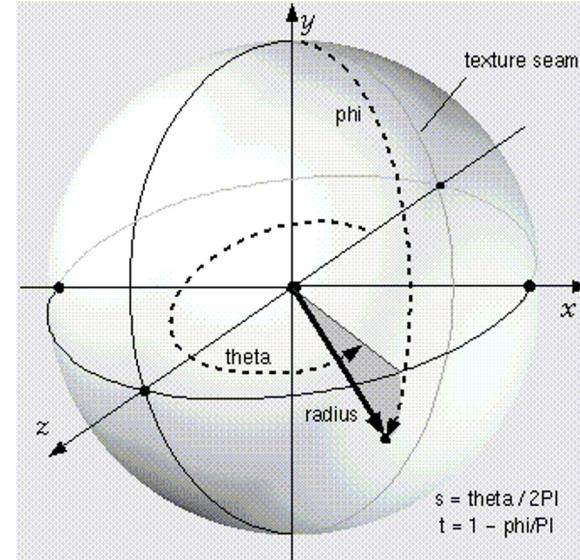
# Exemplo

Esfera Branca



# Abordagem do X3D (exemplo)

```
Sphere : X3DGeometryNode {  
    SFNode [in,out] metadata           NULL      [X3DMetadataObject]  
    SFFloat [ ]      radius            1          (0,∞)  
    SFBool [ ]      solid             TRUE  
}
```



# Especificação X3D

<https://www.web3d.org/documents/specifications/19775-1/V3.3/Part01/Architecture.html>



## Extensible 3D (X3D) Part 1: Architecture and base components

ISO/IEC 19775-1:2013



This document is Edition 3 of ISO/IEC 19775-1, Extensible 3D (X3D). The full title of this part of the International Standard is: *Information technology — Computer graphics, image processing and environmental data representation — Extensible 3D (X3D) — Part 1: Architecture and base components*. When navigating within this document, it is possible to return to the beginning of the document by clicking on the X3D logo.

Background	Clauses	Annexes
Foreword	1 <a href="#">Scope</a>	A <a href="#">Core profile</a>
Introduction	2 <a href="#">Normative references</a>	B <a href="#">Interchange profile</a>
	3 <a href="#">Terms, definitions, acronyms, and abbreviations</a>	C <a href="#">Interactive profile</a>
	4 <a href="#">Concepts</a>	D <a href="#">MPEG-4 interactive profile</a>
	5 <a href="#">Field type reference</a>	E <a href="#">Immersive profile</a>
	6 <a href="#">Conformance</a>	F <a href="#">Full profile</a>
	7 <a href="#">Core component</a>	G <a href="#">Recommended navigation behaviours</a>
	22 <a href="#">Environmental sensor component</a>	
	23 <a href="#">Navigation component</a>	
	24 <a href="#">Environmental effects component</a>	
	25 <a href="#">Geospatial component</a>	
	26 <a href="#">Humanoid animation (H-Anim) component</a>	
	27 <a href="#">NURBS component</a>	
	28 <a href="#">Distributed interactive simulation (DIS) component</a>	

alternativamente: <https://www.web3d.org/documents/specifications/19775-1/V3.3/Part01/nodeIndex.html>

# Componentes de Geometria 2D

<https://www.web3d.org/documents/specifications/19775-1/V3.3/Part01/components/geometry2D.html>



## Extensible 3D (X3D) Part 1: Architecture and base components

### 14 Geometry2D component



#### 14.1 Introduction

##### 14.1.1 Name

The name of this component is "Geometry2D". This name shall be used when referring to this component in the COMPONENT statement (see [7.2.5.4 Component statement](#)).

##### 14.1.2 Overview

This clause describes the Geometry2D component of this part of ISO/IEC 19775. This includes how two-dimensional geometry is specified and what shapes are available. [Table 14.1](#) provides links to the major topics in this clause.

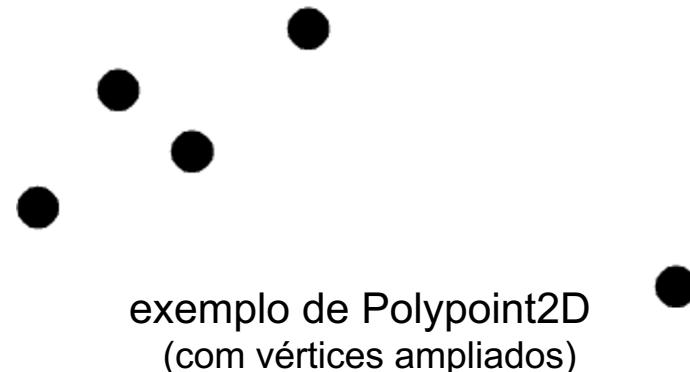
Table 14.1 – Topics

- [14.1 Introduction](#)
  - [14.1.1 Name](#)
  - [14.1.2 Overview](#)
- [14.2 Concepts](#)
  - [14.2.1 Overview of geometry](#)

# Polypoint2D

O nó **Polypoint2D** especifica uma série de vértices no sistema de coordenadas 2D local em cada um dos quais é exibido um ponto. O campo **points** especifica os vértices a serem exibidos.

```
Polypoint2D : X3DGeometryNode {  
    SFNode      [in,out]   metadata  NULL      [X3DMetadataObject]  
    MFVec2f    [in,out]   point     []          (-∞,∞)  
}
```

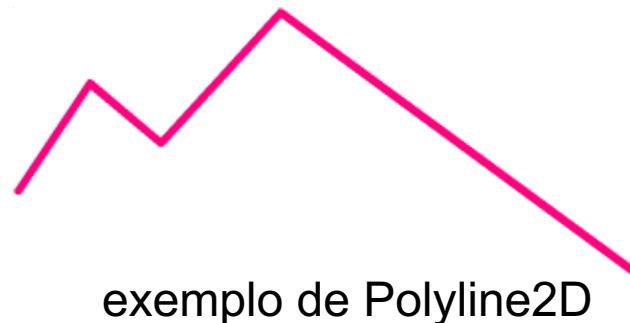


exemplo de Polypoint2D  
(com vértices ampliados)

# Polyline2D

O nó **Polyline2D** especifica uma série de segmentos de linha contiguos no sistema de coordenadas 2D local conectando os vértices especificados. O campo **lineSegments** especifica os vértices a serem conectados.

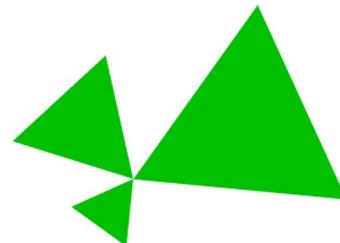
```
Polyline2D : X3DGeometryNode {  
    SFNode [in,out] metadata  
    MFVec2f [] lineSegments  
    NULL [] [X3DMetadataObject]  
    (-∞,∞)  
}
```



# TriangleSet2D

O nó **TriangleSet2D** especifica um conjunto de triângulos no sistema de coordenadas 2D local. O campo **vertices** especifica os triângulos a serem exibidos. O número de vértices fornecidos deve ser igualmente divisível por três. O excesso de vértices deve ser ignorado.

```
TriangleSet2D : X3DGeometryNode {  
    SFNode      [in,out] metadata NULL   [X3DMetadataObject]  
    MFVec2f    [in,out] vertices    [] (-∞,∞)  
    SFBool      [] solid        FALSE  
}
```



exemplo de TriangleSet2D

# Appearance

O nó **Appearance** especifica as propriedades visuais da geometria. O campo material, se especificado, deve conter um nó Material.

```
Appearance : X3DAppearanceNode {  
    SFNode [in,out] fillProperties      NULL [FillProperties]  
    SFNode [in,out] lineProperties     NULL [LineProperties]  
    SFNode [in,out] material          NULL [X3DMaterialNode]  
    SFNode [in,out] metadata           NULL [X3DMetadataObject]  
    MFNode [in,out] shaders            [] [X3DShaderNode]  
    SFNode [in,out] texture             NULL [X3DTextureNode]  
    SFNode [in,out] textureTransform   NULL [X3DTextureTransformNode]  
}
```

# Material

O nó **Material** especifica propriedades de material de superfície para nós de geometria associados e é usado pelas equações de iluminação X3D durante a renderização.

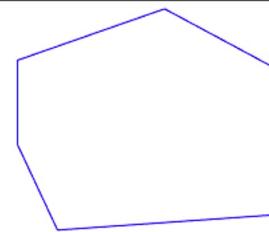
```
Material : X3DMaterialNode {  
    SFFloat  [in,out]  ambientIntensity  0.2          [0,1]  
    SFColor   [in,out]  diffuseColor      0.8 0.8 0.8 [0,1]  
    SFColor  [in,out] emissiveColor  0 0 0          [0,1]  
    SFNode    [in,out]  metadata        NULL         [X3DMetadataObject]  
    SFFloat   [in,out]  shininess       0.2          [0,1]  
    SFColor   [in,out]  specularColor   0 0 0          [0,1]  
    SFFloat   [in,out]  transparency    0            [0,1]  
}
```

# Exemplos



## X3D Example Archives: X3D for Web Authors, Chapter 10 Geometry 2D, Polyline 2D

Example of Polyline2D showing multiple 2D line segments.



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.3//EN" "http://www.web3d.org/specifications/x3d-3.3.dtd">
<X3D profile='Immersive' version='3.3' xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance' xsd:noNamespaceSchemaLocation ='http://www.web3d.org/specifications/x3d-3.3.xsd'>
<head>
  <component level='2' name='Geometry2D'/>
  <meta name='title' content='Polyline2D.x3d'/>
  <meta name='description' content='Example of Polyline2D showing multiple 2D line segments.'/>
  <meta name='creator' content='Leonard Daly and Don Brutzman'/>
  <meta name='created' content='17 April 2006'/>
  <meta name='modified' content='20 October 2019'/>
  <meta name='reference' content=' http://X3dGraphics.com '/>
  <meta name='reference' content=' https://www.web3d.org/x3d/content/examples/X3dResources.html '/>
  <meta name='rights' content='Copyright 2006, Daly Realism and Don Brutzman'/>
  <meta name='subject' content=' X3D book, X3D graphics, X3D-Edit, http://www.x3dGraphics.com '/>
  <meta name='identifier' content=' http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter10Geometry2D/Polyline2D.x3d '/>
  <meta name='generator' content='X3D-Edit 3.3, https://savage.nps.edu/X3D-Edit'/>
  <meta name='license' content=' ./license.html'/>
</head>
```

Index for DEF node : [MagentaAppearance](#)

## Exemplo:

<https://x3dgraphics.com/examples/X3dForWebAuthors/Chapter10Geometry2D/Polyline2DIndex.html>

# Visualizando X3D online

Uma forma é criar um HTML e chamar os scripts em Javascript para rodar seu X3D diretamente.

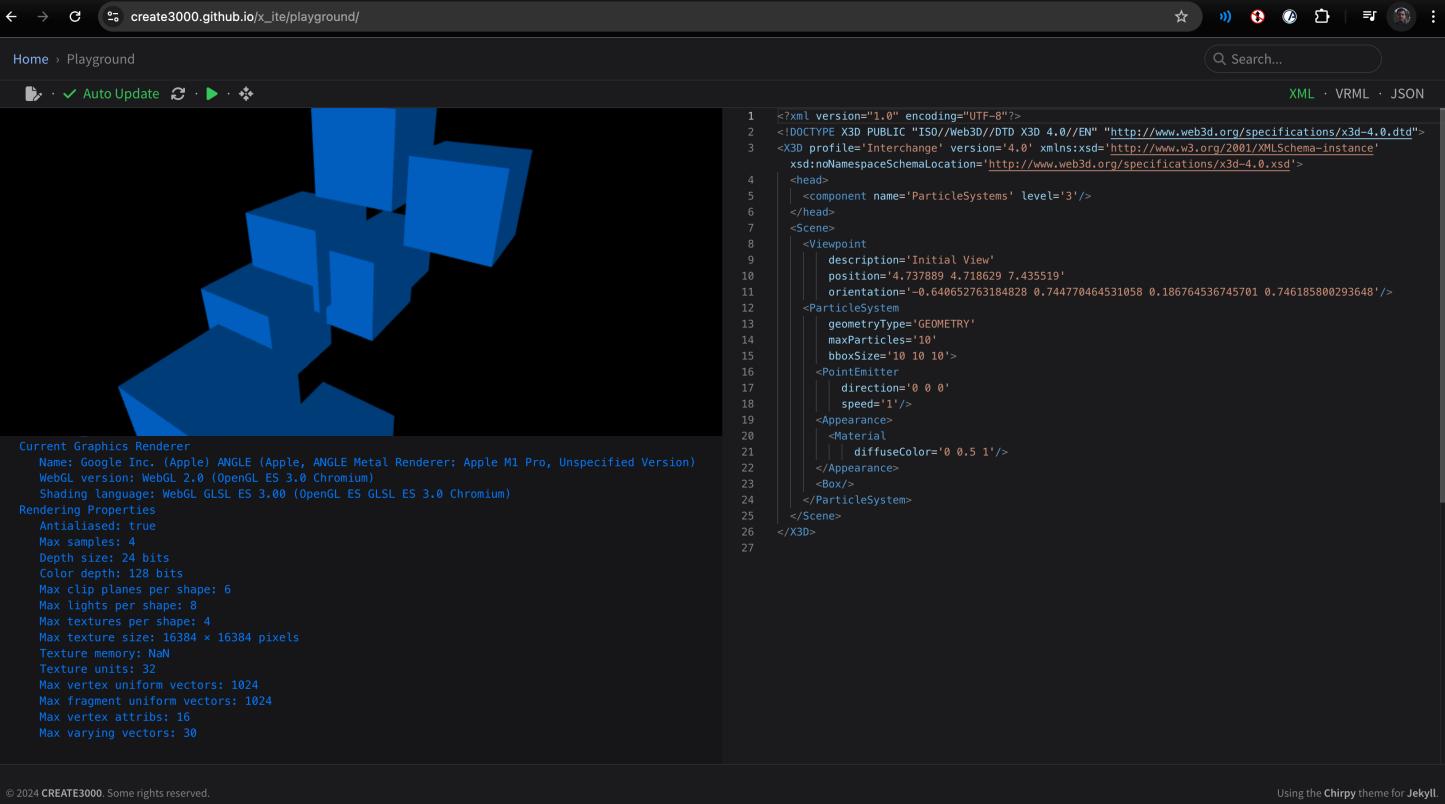
```
<html>
  <head>
    <script src="https://create3000.github.io/code/x_ite/latest/x_ite.min.js"></script>
  </head>
  <body>
    <x3d-canvas src="triang3d.x3d"></x3d-canvas>
  </body>
</html>
```

Outra forma é abrindo o arquivo X3D em alguma plataforma web que suporte o formato. Por exemplo:

[https://create3000.github.io/x\\_ite/playground/](https://create3000.github.io/x_ite/playground/)

# X\_ITE X3D Browser

[https://create3000.github.io/x\\_ite/playground/](https://create3000.github.io/x_ite/playground/)



The screenshot shows a web browser window displaying the X\_ITE X3D Browser playground. The main area shows a 3D scene consisting of several blue cubes arranged in a stepped, staircase-like pattern against a black background. Below the scene, there is a "Current Graphics Renderer" section and a "Rendering Properties" section.

**Current Graphics Renderer**

- Name: Google Inc. (Apple) ANGLE (Apple, ANGLE Metal Renderer; Apple M1 Pro, Unspecified Version)
- WebGL version: WebGL 2.0 (OpenGL ES 3.0 Chromium)
- Shading Language: WebGL GLSL ES 3.00 (OpenGL ES GLSL ES 3.0 Chromium)

**Rendering Properties**

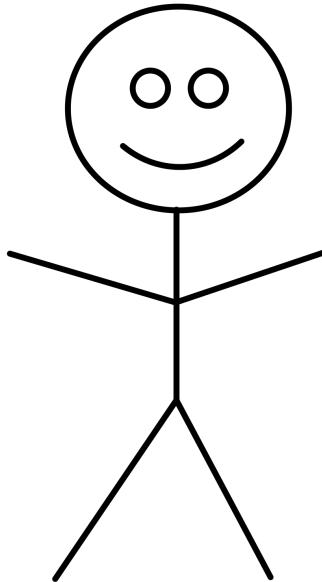
- Antialiased: true
- Max samples: 4
- Depth size: 24 bits
- Color depth: 128 bits
- Max clip planes per shape: 6
- Max lights per shape: 8
- Max textures per shape: 4
- Max texture size: 16384 x 16384 pixels
- Texture memory: NaN
- Texture units: 32
- Max vertex uniform vectors: 1024
- Max fragment uniform vectors: 1024
- Max vertex attrs: 16
- Max varying vectors: 30

On the right side of the browser window, the X3D XML code for the scene is displayed:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 4.0//EN" "http://www.web3d.org/specifications/x3d-4.0.dtd">
<X3D profile='Interchange' version='4.0' xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance'
xsd:NamespaceSchemaLocation='http://www.web3d.org/specifications/x3d-4.0.xsd'>
<head>
| <component name='ParticleSystems' level='3' />
</head>
<Scene>
| <Viewpoint
|   description='Initial View'
|   position='4.737889 4.718629 7.435519'
|   orientation='-0.640652763184828 0.744770464531058 0.186764536745701 0.746185800293648' />
<ParticleSystem
|   geometryType='GEOMETRY'
|   maxParticles='10'
|   bboxSize='10 10 10'>
|     <PointEmitter
|       direction='0 0 0'
|       speeds='1' />
|     <Appearance>
|       <Material
|         diffuseColor='0 0.5 1' />
|       </Appearance>
|       <Box/>
|     </ParticleSystem>
|   </Scene>
</X3D>
```

# Atividade – Crie um Boneco Palito em X3D

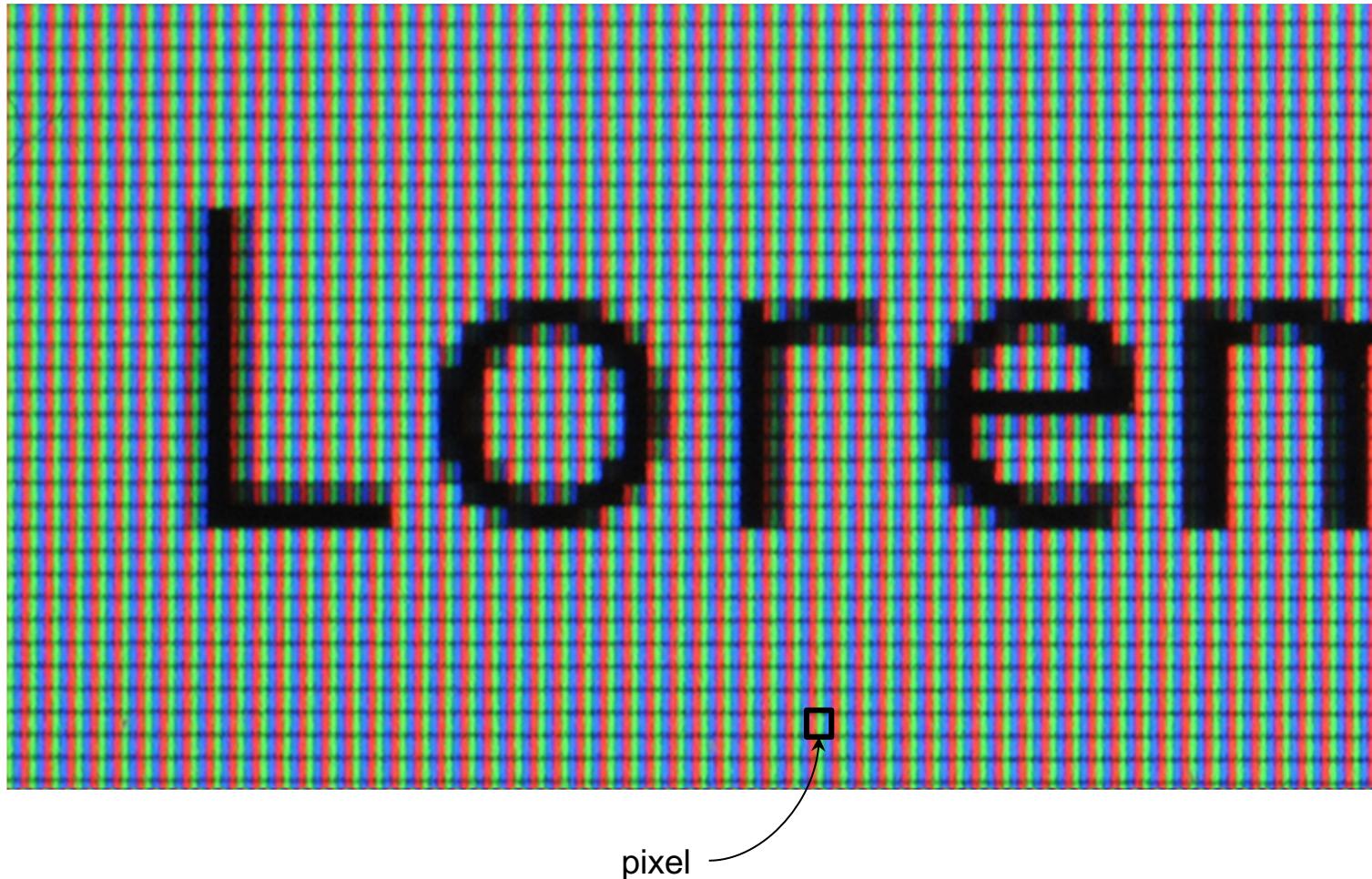
Desenhe um boneco palito em X3D. Use os recursos que achar mais conveniente para desenhar seu boneco.



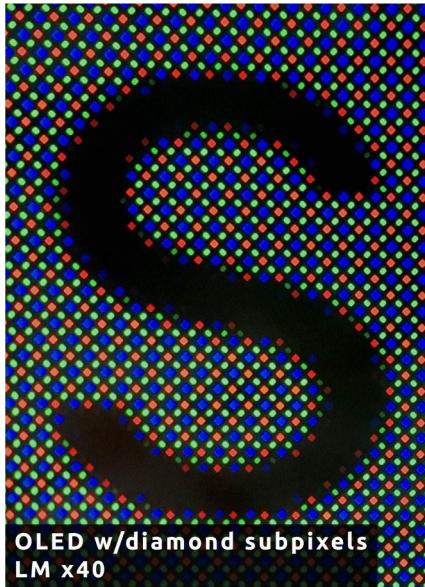
# Como se desenham linhas no computador?



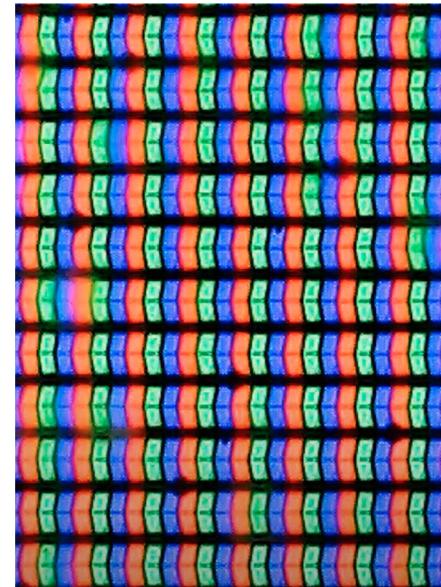
# Funcionamento dos Displays



# Funcionamento dos Displays



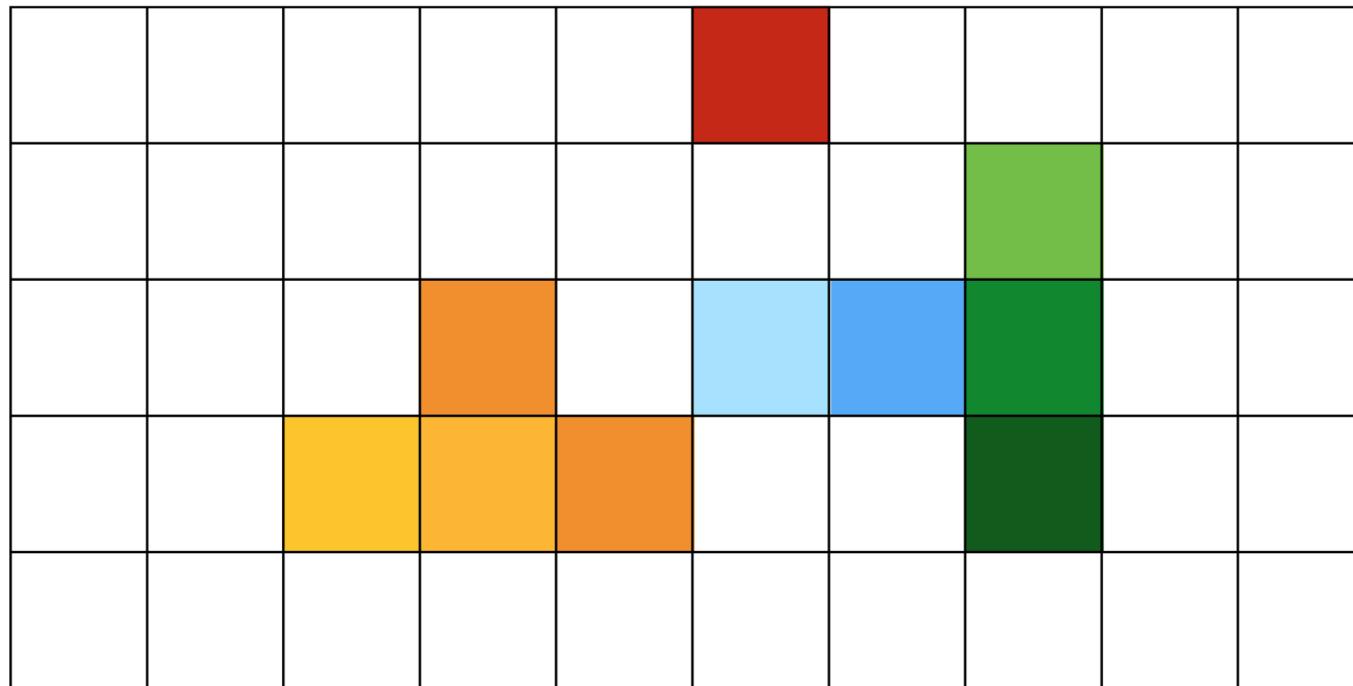
**iPhone X**



**iPhone 8**

# Como as imagens são representadas

As imagens são representadas com um grid 2D de pixels (**picture x element**). Cada pixel possui uma cor e brilho.



# Profundida de cores

A profundidade de cor indica a quantidade de memória usada para representar os canais de cores de cada pixel.

Tradicionalmente usamos 8 bits por canal de cor, ordenados em vermelho(R), Verde(G) e Azul(B), levando a 24 bits por pixel.

#FFFFFF	White RGB(255, 255, 255)
#FFC0CB	Pink RGB(255, 192, 203)
#FF0000	Red RGB(255, 0, 0)
#FFA500	Orange RGB(255, 165, 0)
#FFFF00	Yellow RGB(255, 255, 0)
#008000	Green RGB(0, 128, 0)
#00FFFF	Cyan/Aqua RGB(0, 255, 255)
#0000FF	Blue RGB(0, 0, 255)
#800080	Purple RGB(128, 0, 128)

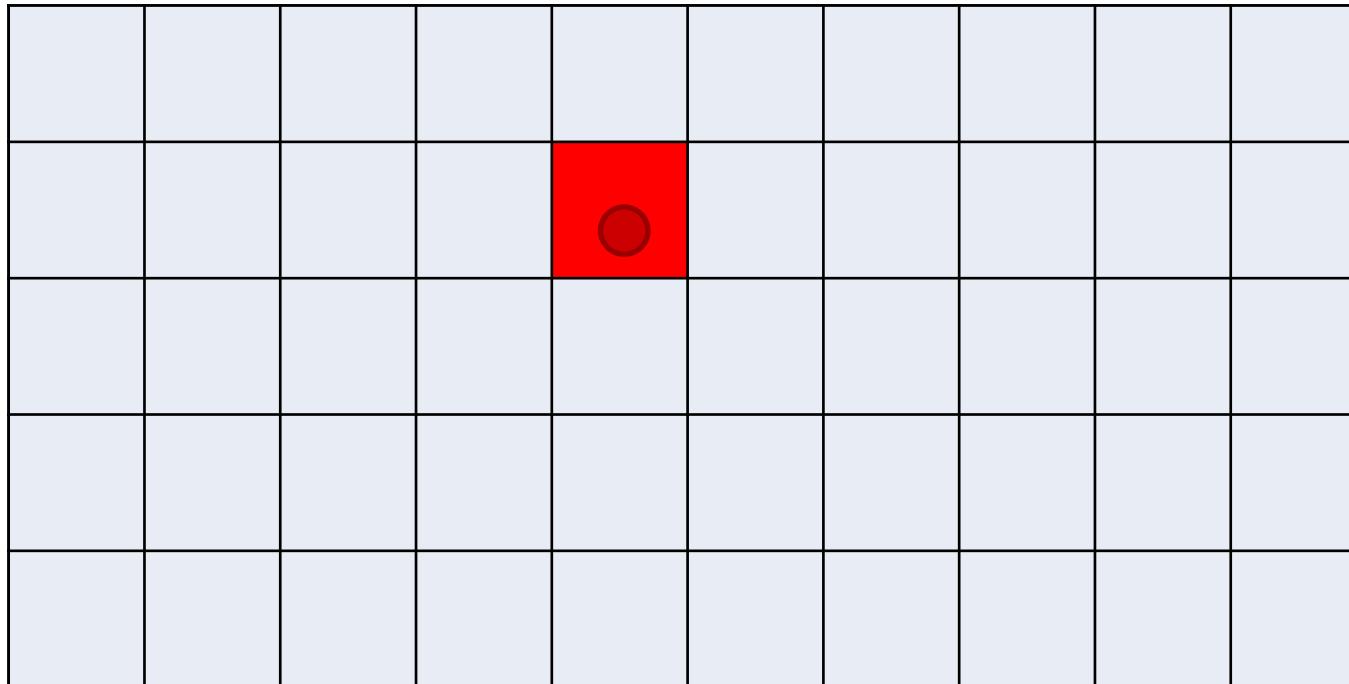
# Frame Buffer

Memória (usualmente na placa gráfica) que armazena esse grid de pixels para depois ser enviado para o display.



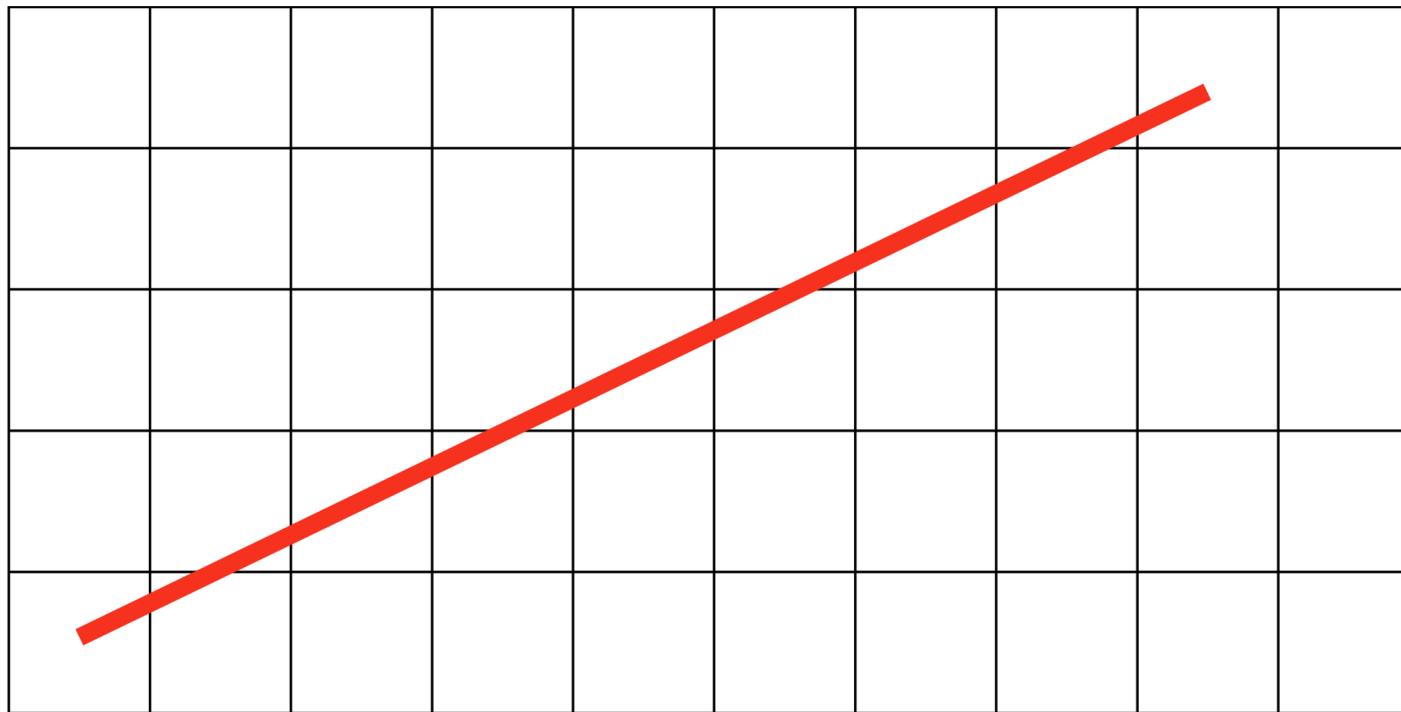
# Desenhando Pontos na Tela

Normalmente desenhamos um pixel quando queremos desenhar um ponto na tela.



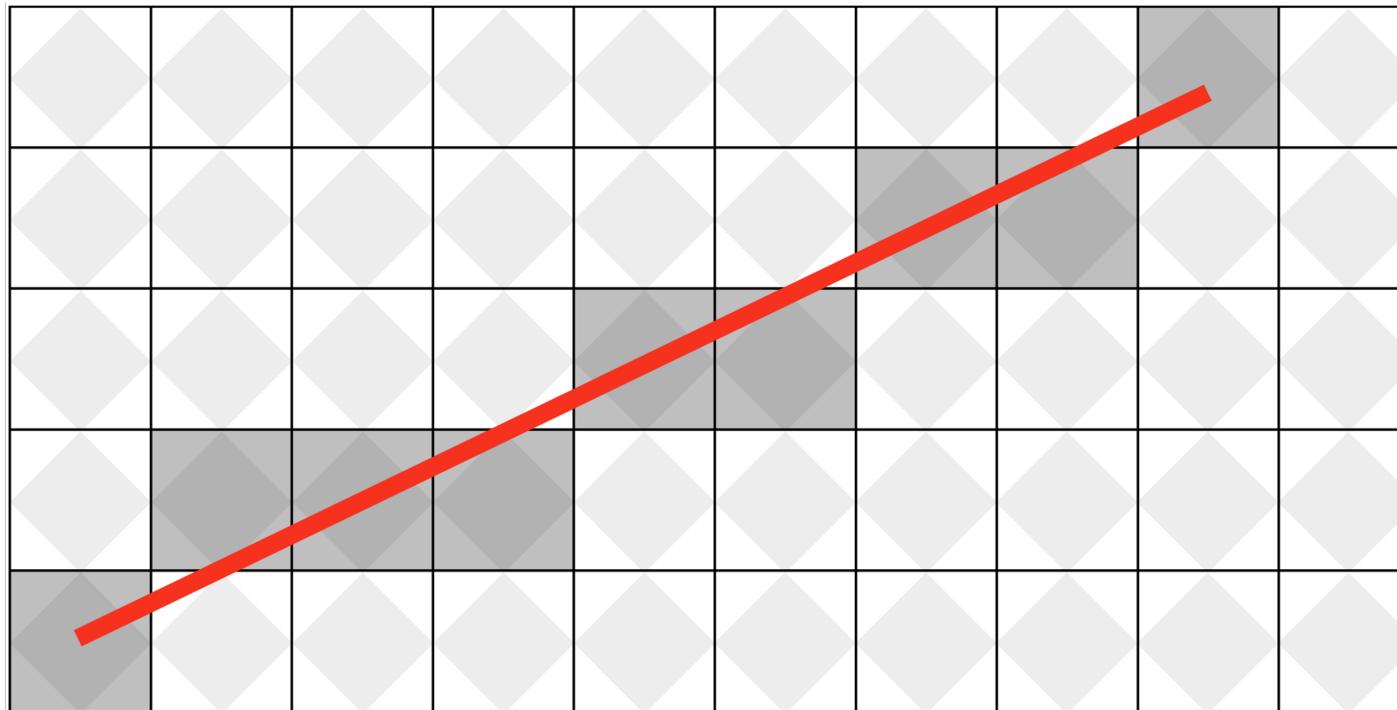
# Quais pixels eu devo colorir para ver a linha?

**"Rasterização": processo de converter um objeto contínuo (linha, polígono, etc) em uma representação discreta em um display "raster" (um grid de pixels)**



# Quais pixels eu devo colorir para ver a linha?

***Diamond rule* (usada em GPUs modernas):  
selecione o pixel se a linha passar pelo diamante  
associado.**

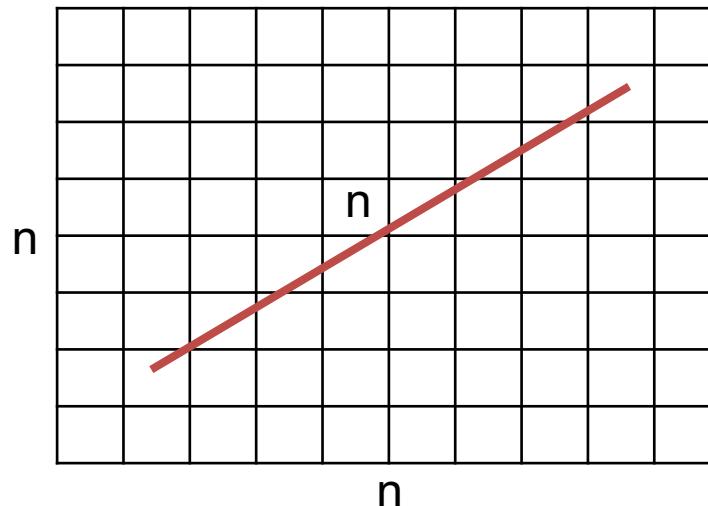


# Mas qual o algoritmo para selecionar os pixels?

**Uma solução seria ver pixel a pixel se a linha está interseccionando o pixel.**

$O(n^2)$  : para busca de pixels na imagens

enquanto temos  $O(n)$  pixels realmente necessários



# Rasterização Incremental de Linhas

Vamos assumir que a linha é representada por  $(u_1, v_1)$  e  $(u_2, v_2)$

Coeficiente angular da reta:  $s = (v_2 - v_1) / (u_2 - u_1)$

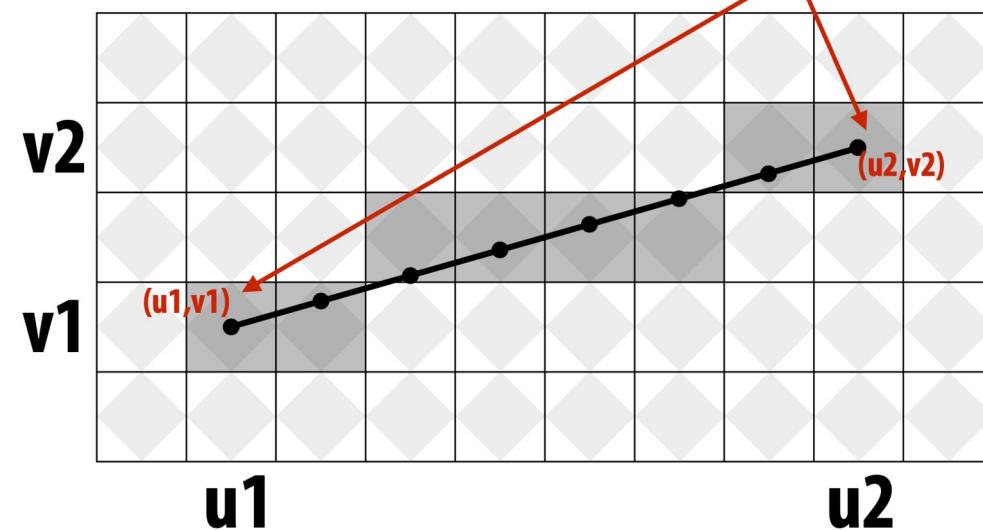
Considere o caso especial onde:

$$u_1 < u_2 \text{ e } v_1 < v_2$$
$$0 < s < 1$$

```
v = v1;  
for( u=u1; u<=u2; u++ )  
{  
    draw( u, round(v) )  
    v += s;  
}
```

Essa é a base para o  
**Algoritmo de Bresenham**

Assuma que os pixels estão  
coordenadas inteiras



# Algoritmo de Bresenham

## Bitmap/Bresenham's line algorithm

< [Bitmap](#)

### Task

Using the data storage type defined on the [Bitmap](#) page for raster graphics images, draw a line given two points with Bresenham's line algorithm.

### Contents [\[hide\]](#)

- 1 360 Assembly
- 2 Ada
- 3 ALGOL 68
- 4 Assembly
- 5 AutoHotkey
- 6 Autolt
- 7 bash
- 8 BASIC
- 9 Batch File
- 10 BBC BASIC
- 11 C
- 12 C#
- 13 C++
- 14 Clojure
- 15 CoffeeScript
- 16 Commodore Basic
- 17 Common Lisp
- 18 D
- 19 Delphi
- 20 E
- 21 Elm
- 22 Erlang
- 23 ERRE
- 24 Euphoria
- 25 F#
- 26 Factor

[http://rosettacode.org/wiki/Bitmap/Bresenham%27s\\_line\\_algorithm](http://rosettacode.org/wiki/Bitmap/Bresenham%27s_line_algorithm)

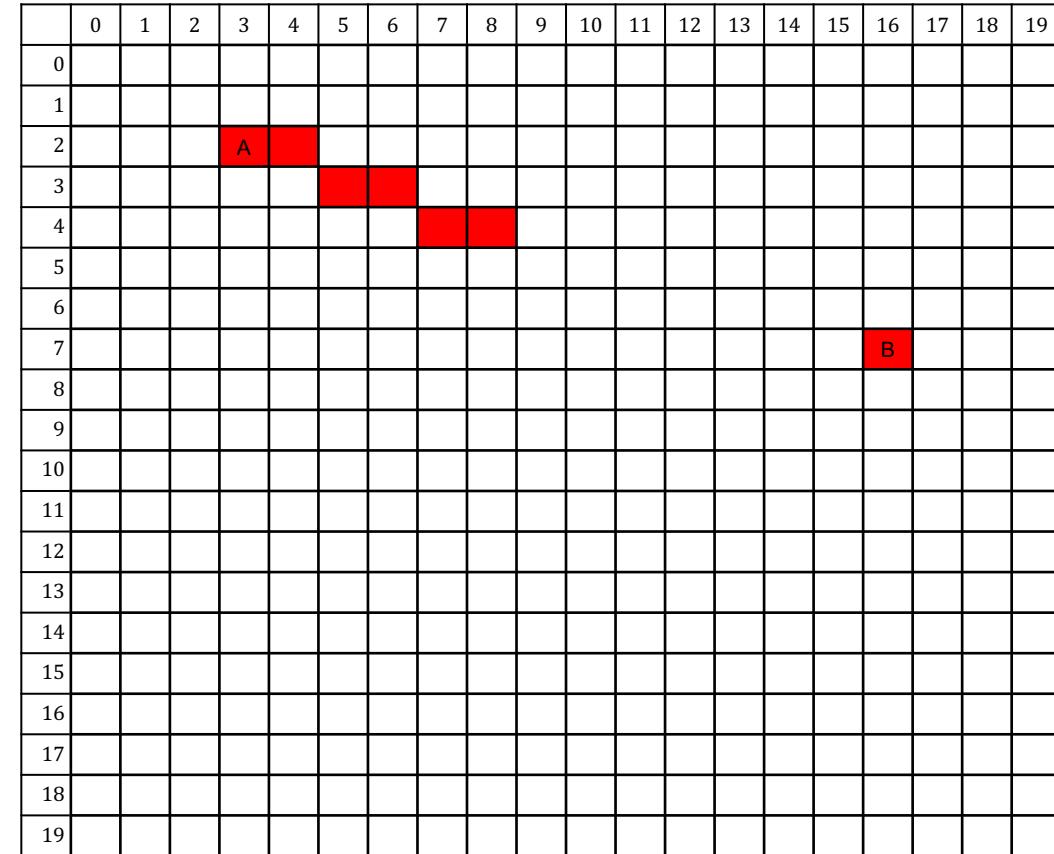
# Atividade - Exercício

Desenhe uma linha usando o algoritmo de Bresenham, ligando os pontos A(3, 2) e B(16, 7).

Base para o  
**Algoritmo de Bresenham**

```
v = v1;  
for( u=u1; u<=u2; u++ )  
{  
    draw( u, round(v) )  
    v += s;  
}
```

ca = 5/13 ~= 0.4  
v = 2.5, 2.9, 3.3, 3.7  
u = 3.5, 4.5, 5.5, 6.5



# Atividade - Resolva o Python Notebook

## 01 - Introdução

Na atividade a seguir você deverá desenhar uma linha preenchendo os pixels entre dois pontos. Para essa atividade vamos usar o Numpy e Matplotlib. Assim vamos carregar as bibliotecas e vamos definir valores de resolução (altura e largura) da nossa janela.

```
import numpy as np
import matplotlib.pyplot as plt

height = 8
width = 8
```

[ ]

Python

Agora vamos desenhar uma tela vazia. Verifique que criamos uma variável *frame\_buffer* que irá armazenar os valores dos pixels.

```
# Cria o espaço para a figura
fig, axes = plt.subplots()

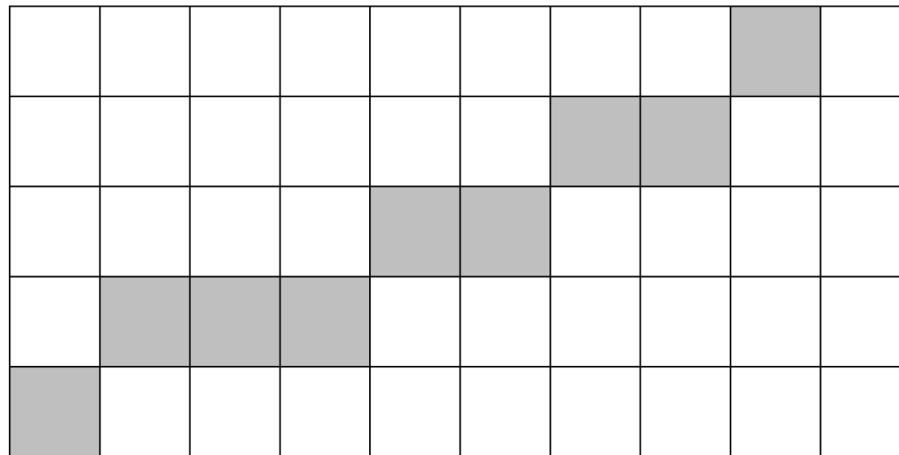
# Cria um buffer de imagem inicialmente completamente preto (limpo)
frame_buffer = np.zeros((height, width))

# Define o tamanho da imagem e que a origem é no canto superior esquerdo
extent = (0, width, height, 0)
```

# Próxima Aula

Na próxima aula, vamos falar sobre desenhar um triângulo no frame buffer de um computador.

E é muito mais interessante do que pode parecer ...  
Além disso, o que há com essas linhas "irregulares"?



# Computação Gráfica

Luciano Soares  
[<lpsoares@insper.edu.br>](mailto:lpsoares@insper.edu.br)

Fabio Orfali  
[<fabioO1@insper.edu.br>](mailto:fabioO1@insper.edu.br)