# Visualização de Dados
# Data Visualization
2021/2022

**Mestrado em Engenharia Informática**
**Mestrado em Informática**
**Especialização em Informática**
**Outros mestrados e pós-graduações**

**1st semester**

**Course slides**

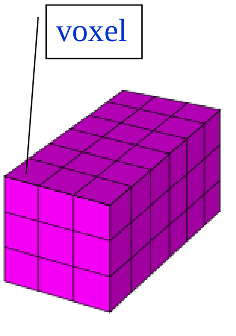
Beatriz Carmo

DI-FCUL

GU-VD-21-5



## Volume Visualization
## *(Visualização de dados volumétricos)*




- **Volume visualization** refers to the visualization of data arranged on a volume, usually on a regular or cartesian grid.

- The volume has cubic cells called **voxels**

# Volume Visualization
# *(Visualização de dados volumétricos)*

- An example of volumetric data is the data obtained in biomedical **imaging** from **magnetic resonance** (MR) (*ressonância magnética* ) or **computed tomography (**CT) (*tomografia computorizada*) scanners

- Another example of volumetric data is the data concerning meteorological studies, obtained either by simulation or by measurements, using parallel slices to organize data over time



# Volume Visualization

- **Volume Visualization (***Visualização de Dados Volumétricos ou Visualização Volumétrica***)** comprises two types of techniques:

  - ***Surface-fitting*** – the visualization is generated using geometric primitives such as lines or polygons.

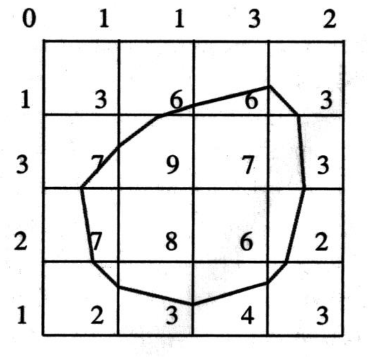
    Example: isolines or isosurfaces generated using *edge-tracking* or *marching-squares* (*cubes*) algorithms already studied.

  - ***Direct Volume Rendering*** or simply, ***Volume Rendering*** – operates directly on the dataset to produce an image without generating an intermediate geometric representation

# *Surface-fitting*

Contouring (*traçado de isolinhas e isosuperfícies*)

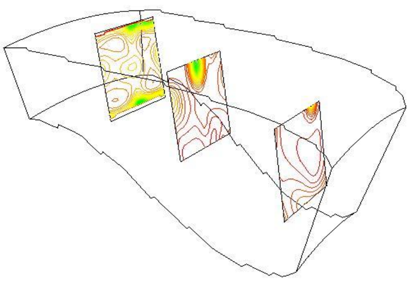
Isoline corresponding to value 5



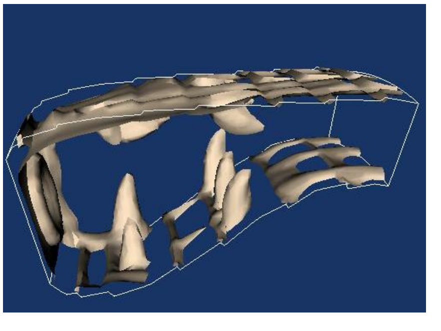



# *Surface-fitting*

Contouring (*traçado de isolinhas e isosuperfícies*)



Isolines



Isosurfaces

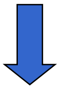
# *Volume Rendering*

- Volume rendering takes into account transparency :
  - Suppose we have a glass box containing not miscible gases of different opacity and color.
  - Inside the box, "clouds" will be visible, without walls separating them.



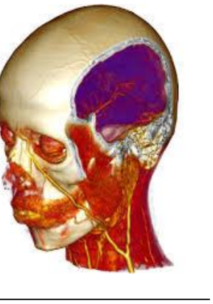
# *Volume Rendering*

In volumetric visualization we begin to create “clouds” setting a correspondence between the values of the variable and color values and opacity.

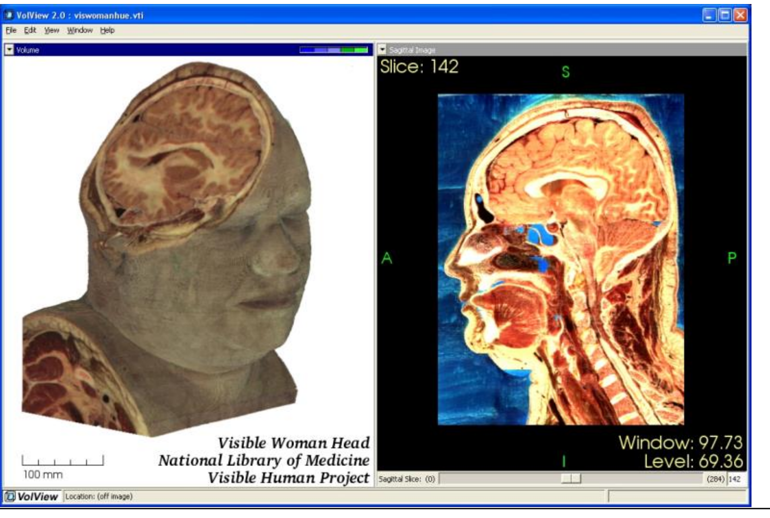


This step is called **classification**.

**Transfer functions** map values of the variable into color and opacity.



Volume rendering and image display from the visible woman dataset.
Author: Kitware, Inc.
http://www.vtk.org/VTK/project/imagegallery.php






# *Volume Rendering*

There are two types of methods to determine the value of each pixel of the image:

*Pixel* by *pixel*

- **image-order** (Visualização controlada pela imagem)

  based on **ray casting technique: for each pixel** of the image rays are cast into the volume, according to the projection parameters, to determine the pixel value (also known as ray-casting algorithm)

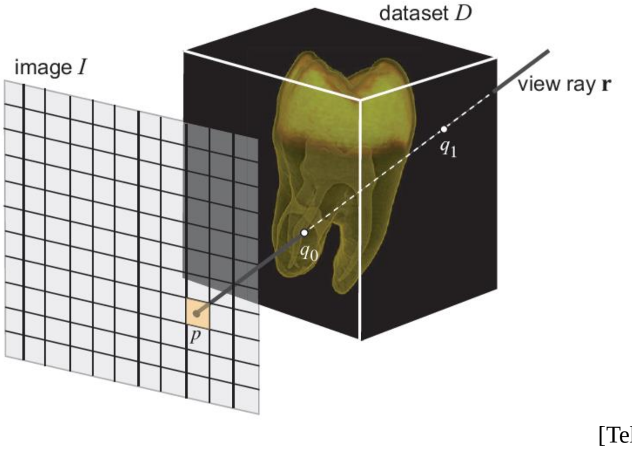
*voxel* a *voxel*

proj

- ***object-order*** **(**Visualização controlada pelos objectos)

  the **volume is traversed** taking into account **the organization of voxels** and the projection parameters, calculating the contribution of each voxel to generate the image (also known as projection algorithm)

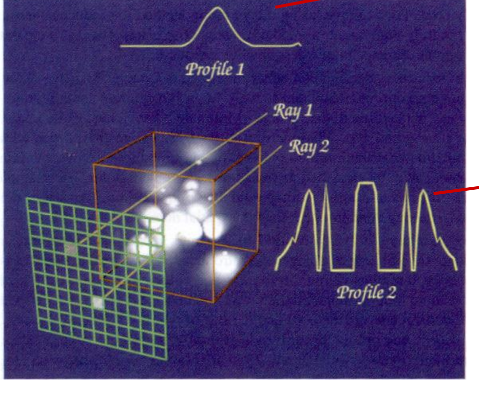
# *Volume Rendering* (***image-order***)




[Telea2014]



# *Volume Rendering* (***image-order***)




Values found in the path of ray 1

Values found in the path of ray 2

# *Volume Rendering* (***image-order***)

- The example in the previous figure shows the launch of 2 rays from the image pixels, using an orthographic projection parallel.

- There are **two important steps** to perform:

  - Determine the **values** of the variable **along the path of the rays** in the volume, ie, the graphs corresponding to the "Profile 1" and "Profile 2".

  - Use a rule **to combine the values obtained**. The rule an be expressed by a function, called **ray function**, to determine the value for the pixel.



# *Volume Rendering* (***image-order***)

Examples of **ray fuctions** to combine values:

a) Maximum value

b) Average value

c) The distance along a ray at which a scalar value at or above 30 is first encountered

d) Alpha compositing technique treats the values along the ray as samples of opacity accumulated per unit distance

# *Alpha compositing technique*

Transparency and its complement, opacity, are referred as *alpha* in computer graphics (*canal "alfa")*

- **Alpha** value is
  - **1** for opaque objects
  - **0** for objects completely transparents

- The alpha value can be stored together with the RGB values that define a color of a pixel, extending the RGB specification to RGBA**.**



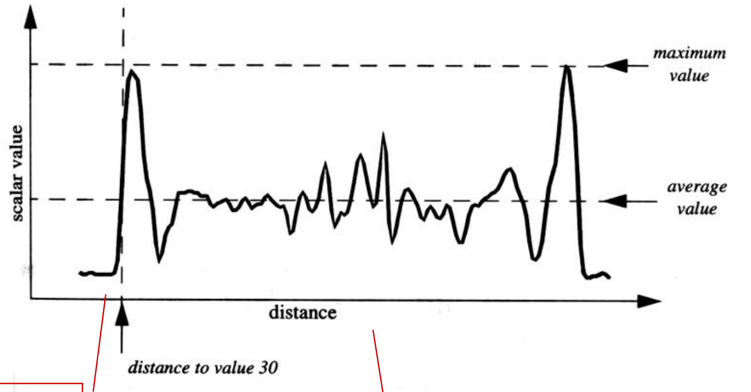
# *Alpha compositing technique*

- The resulting color takes into account all elements intersected using the following equations:
  - $R = A_s R_s + (1\text{-}A_s)\ R_b$  $G = A_s G_s + (1\text{-}A_s)\ G_b$
  - $B = A_s B_s + (1\text{-}A_s)\ B_b$  $A = A_s + (1\text{-}A_s)\ A_b$

where

- $\mathbf{A_s}$ is the **opacity** of element S
- $\mathbf{R_s}$ **,** $\mathbf{G_s}$ **e** $\mathbf{B_s}$ are the values R, G and B of the element
- $\mathbf{A_b}$ is the accumulated opacity behind the element
- $\mathbf{R_b}$ **,** $\mathbf{G_b}$ **e** $\mathbf{B_b}$ are the values R, G e B accumulated behind the element.

# *Volume Rendering* (***image-order***)




The y-axis represents the value of the variable

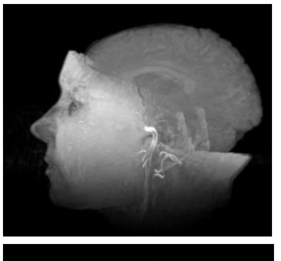
The x-axis indicates the distance to the projection plane

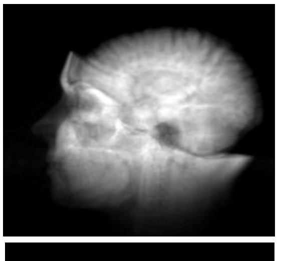


# *Volume Rendering* (***image-order***)

Images obtained with different ray functions:

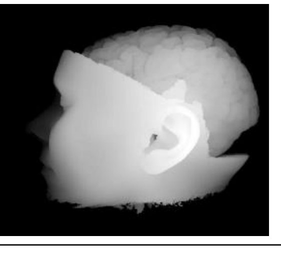
Maximum

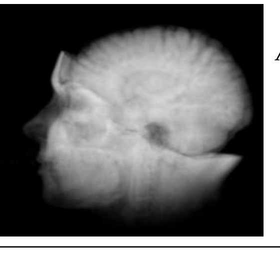




Average

Distance along the ray at which the scalar value is at or above 30 is first encountered





*Alpha compositing*

# *Volume Rendering* (***image-order***)

Determining the values along the path of each ray:

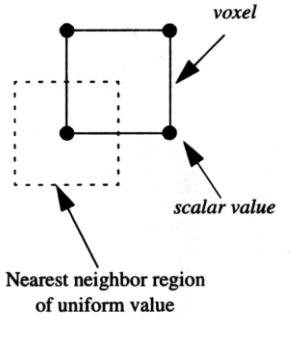
On what points?
How to calculate the value of the variable on that points?

- The value of the variable is known at the nodes of the grid, ie, the vertices of the voxels.
- However, it is necessary to determine the value of the variable at points where the ray passes
- For this purpose an interpolation function is used

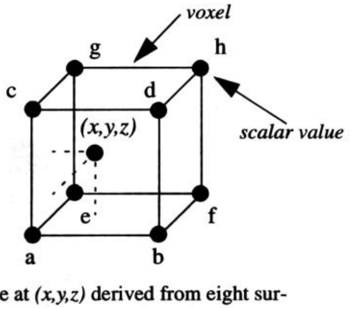


# *Volume Rendering* (***image-order***)

Examples of interpolation functions



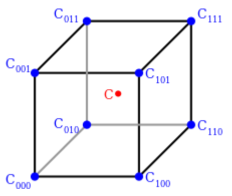

Returns the value of the nearest grid point
(*nearest neighbor interpolation*)

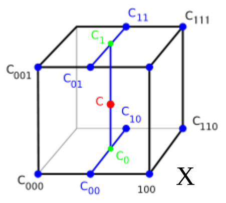



value at *(x,y,z)* derived from eight surrounding scalar values

Calculates the value of the variable at a given point by linear interpolation.
It takes into account the distance to the nodes of the grid by each of the principal axes (trilinear interpolation)

## Trilinear interpolation




The result of trilinear interpolation is <u>independent</u> of the order of the interpolation steps along the three axes.
In the example: first along horizontal axis, then along depth axis, and finally along vertical axis



# *Volume Rendering* (***image-order***)

## On what points?

There are different solutions

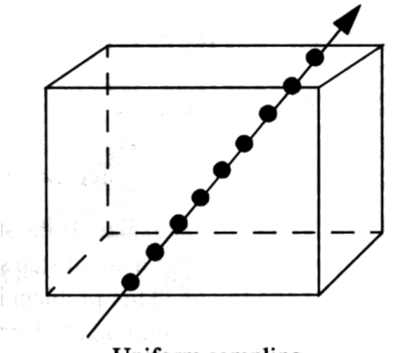
– calculate the value at equally spaced points along the ray

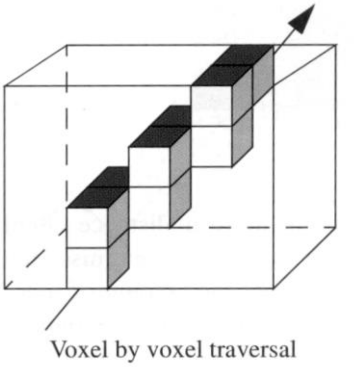
– determine the voxels that are intersected and get the values in these voxels

- The solution adopted depends on several factors:

  - interpolation technique used
  - ray function (*regra de composição de valores*)
  - Performance (processing time vs required accuracy)

# *Volume Rendering* (***image-order***)




Note: the direction of the ray depends on the parameters of the projection. Hence the rays can have directions not aligned with the principal directions of the grids.



# *Volume Rendering* (***object-order***)

Object-order (*visualização controlada pelos objectos*)

the volume is traversed taking into account the organization of voxels and the projection parameters, calculating the contribution of each voxel to generate the image

It is necessary to determine

- How to calculate the contribution of each voxel
- The order in which the volume is traversed

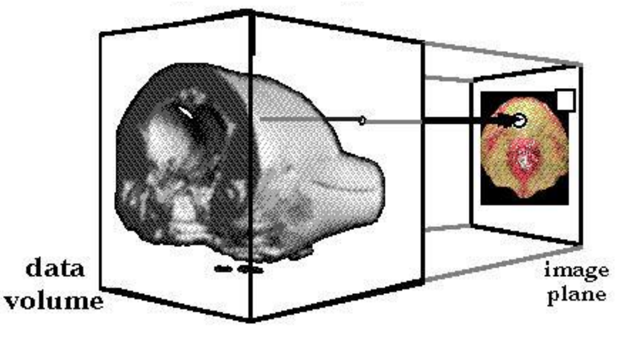
# *Volume Rendering* (***object-order***)

## How to calculate the contribution of each voxel

- When a voxel is processed, it is calculated its projection on the projection plane.
- In the pixels, where the voxel is projected, a composition rule is used to combine the information of this voxel with the information of the other voxels that are also projected into these pixels.



# *Volume Rendering* (*object-order*)




For each voxel, it is determined the area of the image where it is projected
http://www.siggraph.org/education/materials/HyperVis/vistech/volume/volume.htm

# *Volume Rendering* (*object-order*)

## What is the order in which the voxels are traversed?

The order in which the voxels are traversed depends on the composition rule used to combine the values that are associated with a pixel.

For example

If the composition rule is based on the **maximum value**, the distance from the voxel to the projection plane is not important

Any order can be used.



# *Volume Rendering* (*object-order*)

## What is the order in which the voxels are traversed?

When using the **alpha compositing rule** to combine color and opacity, it requires that voxels are traversed according to their distance to the projection (depth)

- back to front
- or front to back

When alpha composition is supported by graphical hardware

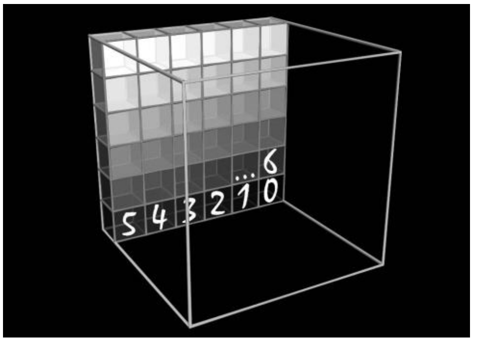
If a *software compositing method* is used, this order allows to stop calculations when a pixel reaches full opacity

# *Volume Rendering* (*object-order*)

The picture illustrates **back to front** order for an orthographic projection

Voxel traversal starts at the voxel that it is furthest from the view plane and then continues progressively to the closer voxels until all voxels are visited

**Triple nested loop**, from the outer to the inner loop:
•the planes in the volume are traversed,
•the rows in a plane are processed
•and the voxels along a line are visited




[Schroeder98,04]



# *Volume Rendering*

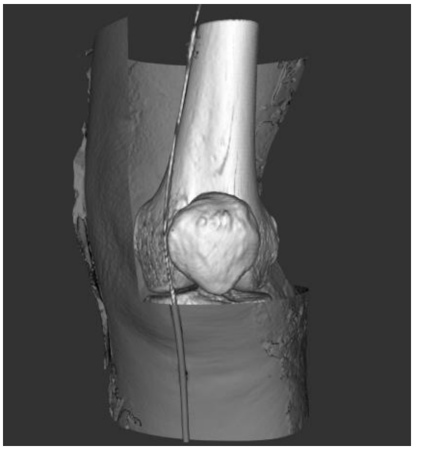
How to obtain a "isosurface" with the volume rendering technique?

# Volume rendering can be combined with other techniques

Volume rendering can be combined with other techniques that generate geometries.

In the picture

- the skin was drawn with an isosurface generated with a contouring method
- the bone was drawn with ray-casting technique (image-order)



(Fig. 7-24 de Schroeder, Martin e Lorensen, "The Visualization Toolkit")



# The use of stereo rendering (*imagens estereoscópicas*) to enhance the visualization of three-dimensional objects

# *Stereo rendering*

Simulation of 3D effect on a 2D display imitating the binocular parallax

That is, the effect generated by **simulating human vision with two eyes**: to images are produced from <u>slightly different points of view</u>

Our brain interprets these differences and identifies the depth of objects.

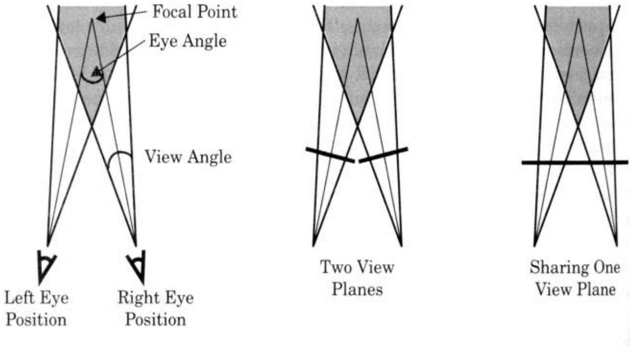
2 different projection planes (ex.: *head mounted displays*)

To generate the images it must be taken into account
- The distance between the two eyes
- And whether the two images are seen in one or two displays

1 projection plane



# *Stereo rendering*



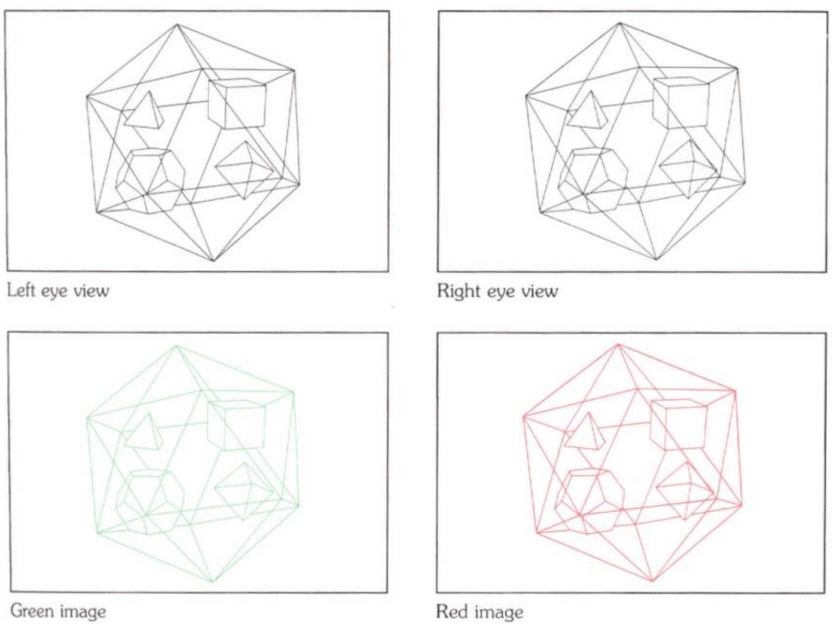

- The simplest model of the synthetic camera assumes that the projection plane is perpendicular to the direction in which one looks.
- This condition does not occur when using the same plane of projection for the images to the two eyes.
- In this case it is necessary to apply translation and shear transformations to the view frustrum.[Shroeder98]

# *Stereo rendering*

A common technique for creating stereo images, called **red-blue** and **red-green** or **red-cyan** or **anaglyphic stereo**, requires the user to wear glasses with a red filter for the left eye and a blue filter (or green or cyan) for the right eye.

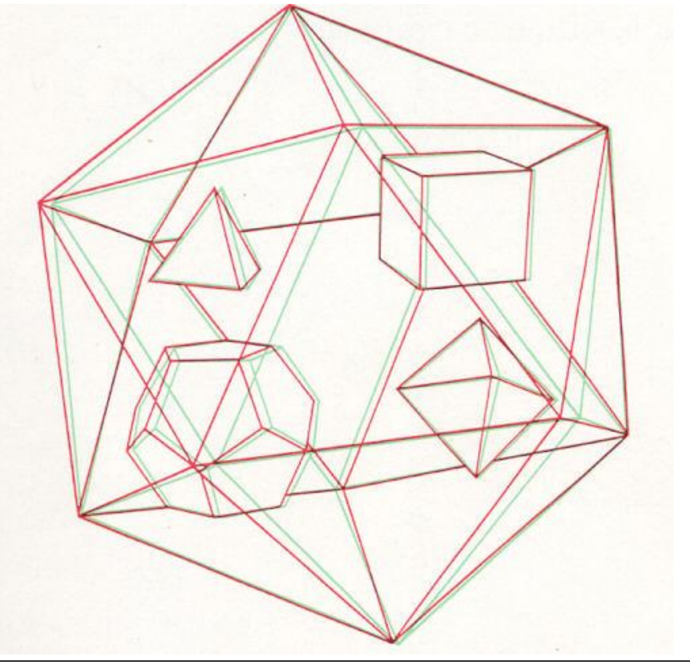


# *Stereo rendering*




[Vince]

*Stereo rendering*

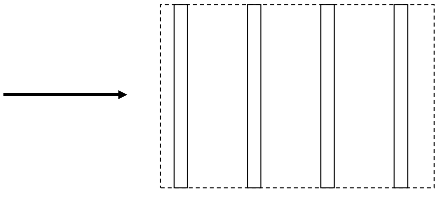


[Vince]



# Exercises

**1)** Consider the situation shown schematically in Figure



| | | | | |
|---|---|---|---|---|
| **Color:** | Blue | Red | Green | Yellow |
| **Alfa:** | 0,5 | 0,2 | 1,0 | 0,5 |

Yellow = (1,1,0)

- In a parallelepipedic volume there are 4 planes with the indicated RGBA values (figure shows a section of the volume) .
- Assuming that a volume rendering technique with alpha composition is used with the viewing direction indicated by the arrow, what is the resulting RGBA value? Explain the calculations made.
- Repeat the previous calculation assuming that the yellow plane changes its color to magenta (1,0,1).

**2)** The volume rendering can be generated by image-order or by object-order. What is the difference between the two techniques?

**3)** How is the projection type used in the calculations?



# Referências


- "The Visualization Toolkit – an Object-Oriented Approach to 3D Graphics", W. Schroeder, K. Martin, B. Lorensen, Prentice Hall PTR,

  2ª edição, 1998

  4ª edição, 2006
- "The VTK User's Guide", Kitware, Inc., 2006
- "Virtual Reality Systems", John Vince, ACM Press, Addison-Wesley1995
- João Cunha, Guiões VI0607, VI 0708, DI-FCUL
- Ana Paula Cláudio, Guiões CG0708, DI-FCUL
- Beatriz Carmo, Guiões FTV0708, DI-FCUL
- Beatriz Sousa Santos, Guiões CG, LEET, 2005,Univ. Aveiro

## Notes about projections – this topic will not be evaluated

Anexo para apoio à aula, mas que não faz parte da matéria

# Notas sobre Projecções Planares

- Para representar objectos tridimensionais em superfícies de visualização bidimensionais, temos de recorrer às projecções.

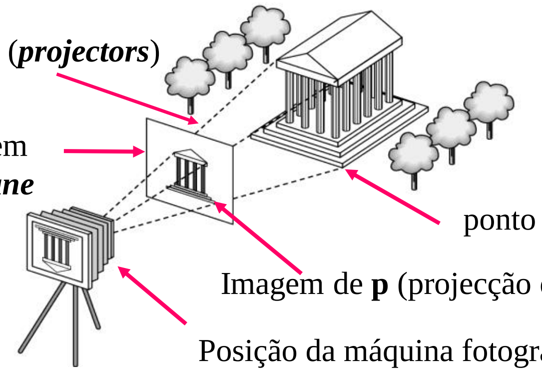
(*To represent three-dimensional objects on two-dimensional display surfaces, we have to resort to projections*)



As projecções usadas em Computação Gráfica são **projecções planares geométricas**, porque se projecta sobre um plano usando rectas como projectantes.

(*The projections used in Computer Graphics are* ***planar geometric projections****, because it is projected onto a plane using straight lines as projectors*).

**Modelo da Máquina Fotográfica Sintética (*Synthetic Camera Model*)**






# Projecções

- A projecção planar geométrica de um ponto é a intersecção da recta **projectante** desse ponto com o **plano de projecção**.

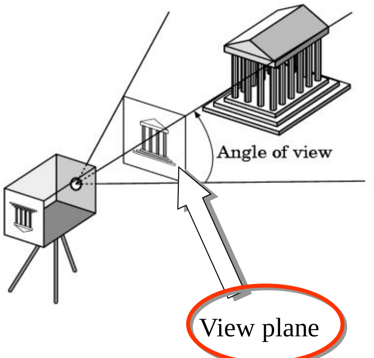
*(The geometric planar projection of a point is the intersection of the projecting line of that point with the projection plane.)*

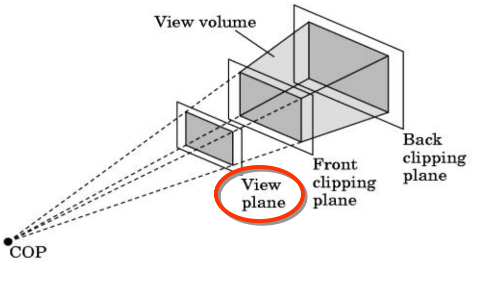
As projecções caracterizam-se pelo plano de projecção e pela forma de definir as projectantes.

(*The* ***projections*** *are* ***characterized*** *by the* ***projection plane*** *and by the way to define the* ***projectors***)



**Modelo da Máquina Fotográfica Sintética**

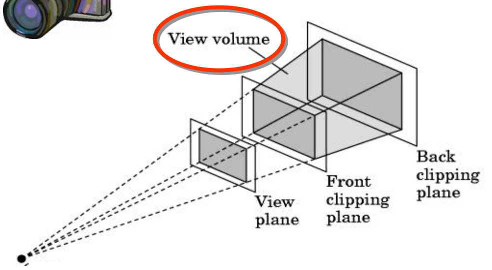


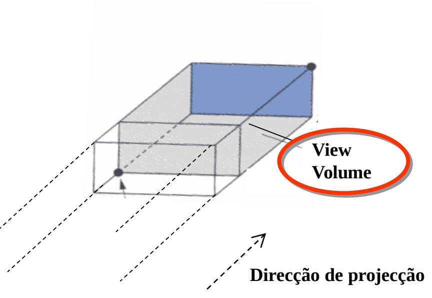




**COP = Centre of projection**

A máquina fotográfica virtual pode apenas captar a imagem dos objectos que se encontram no volume de visualização (*view volume*).
Para eliminar os objectos fora deste volume procede-se a uma operação de recorte (*clipping*).

## Modelo da Máquina Fotográfica Sintética




O volume de visualização (*view volume*) pode ser

um frustrum(*) ou um paralelepípedo.

Projecção perspectiva
**Perspective projection**

Projecção paralela
**Parallel projection**

(*) frustrum é a porção de um sólido, habitualmente um cone ou uma pirâmide

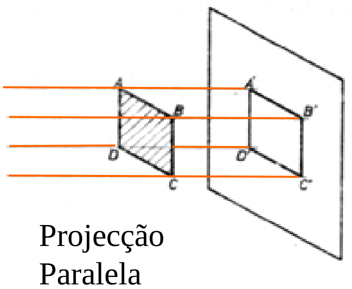


# Projecções

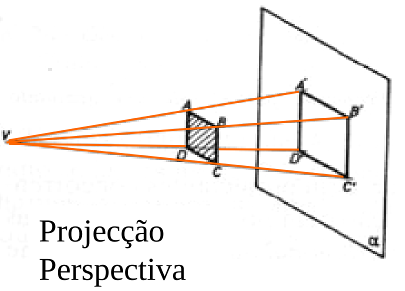
As projecções planares podem ser de dois tipos:

- **Paralelas**: projectantes paralelas entre si
- **Perspectivas**: projectantes convergem no centro de projecção

Encurtamento perspectivo

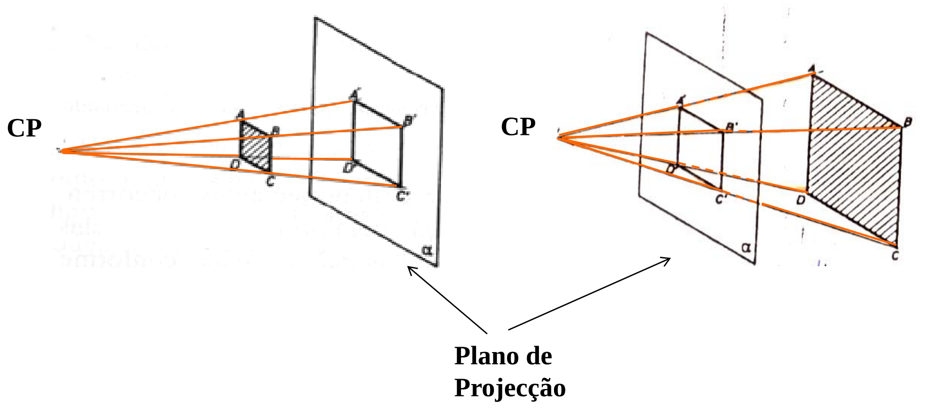


Projecção
Paralela



Projecção
Perspectiva

# Projecções Perspectivas




O tamanho da projecção obtida varia em função da posição relativa entre o objecto projectado, o Plano de Projecção e o Centro de Projecção.



# Projecções Perspectivas

O encurtamento perspectivo dá origem aos pontos de fuga.







# Projecções Perspectivas

Numa projecção perspectiva mantêm-se os paralelismos das rectas paralelas ao plano de projecção.



# Projecções Perspectivas
# Perspective projections

- Numa projecção perspectiva, as projecções das rectas paralelas a uma direcção D, **não paralela** ao plano de projecção, convergem para um **ponto de fuga**.

(*In a perspective projection, the projections of the straight lines parallel to a direction D, not parallel to the projection plane, converge to a* ***vanishing point*** *).*

# Projecções Perspectivas
# Perspective projections

Se D for uma direcção principal (direcção de um dos 3 eixos principais), o ponto de fuga designa-se por **ponto de fuga principal**; caso contrário, designa-se por **ponto de fuga secundário**.

Atenção: centro de projecção e ponto de fuga são conceitos diferentes

*If D is a main direction (direction of one of the 3 principal axes), the vanishing point is called the **axis vanishing point**; otherwise it is called a vanishing point.*

*Centre of projection and vanishing point are different concepts.*



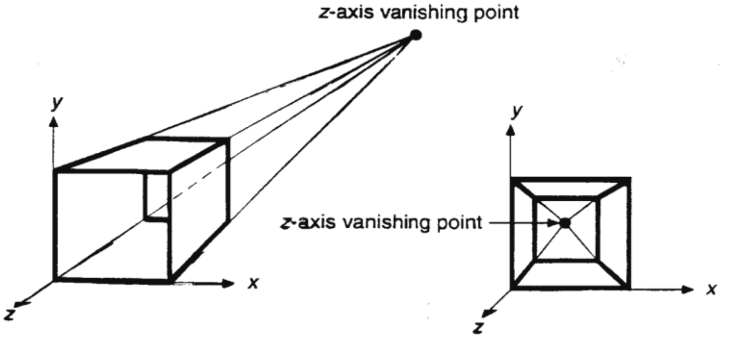
# Projecções Perspectivas
# Perspective projections

A perspective projection is defined by

- the centre of projection
- and projection plane

# Projecções Perspectivas

Exemplos:

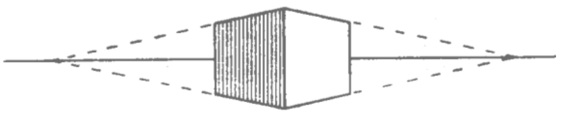



**1 PFP**

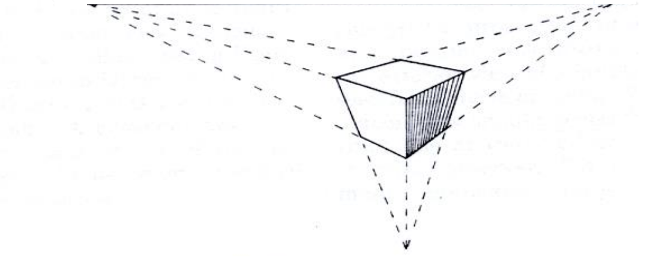


# Projecções Perspectivas

**Exemplos:**



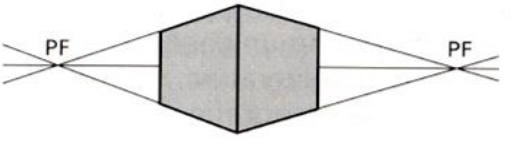
**2 PFP**



**3 PFP**

# Projecções Perspectivas

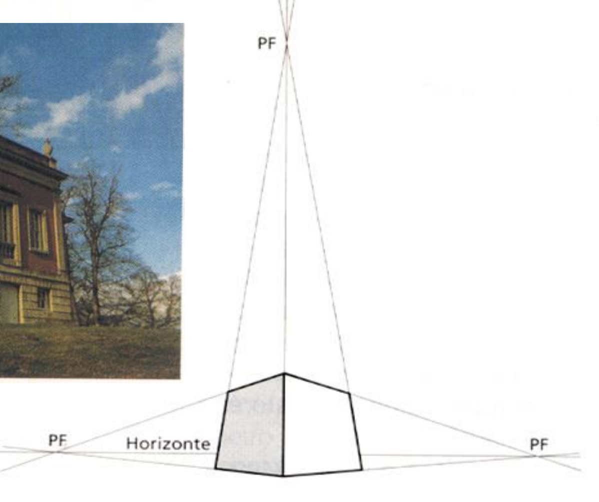
**Exemplos:**






# Projecções Perspectivas

**Exemplos:**



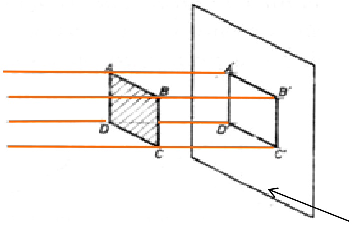
# Projecções Paralelas
# Parallel projections

A parallel projections is defined by

- the direction of the projectors
- and projection plane

As projectantes são paralelas entre si

**Direcção de Projecção**




**Plano de Projecção**



# Projecções Paralelas
# Parallel Projections

**Projecções Paralelas** (DP +PP)

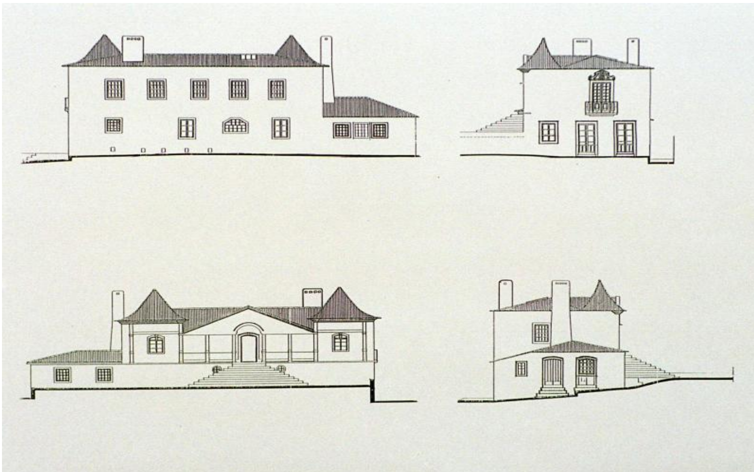
- **Ortográficas ou Ortogonais (orthographic projections)**
  The direction of projection (DP) is **perpendicular** to the projection plane (PP)

- **Oblíquas (oblique projections)**
  The direction of projection (DP) is not **perpendicular** to the projection plane (PP)

# Projecções Paralelas Ortogonais

Exemplos:

• **Alçados**

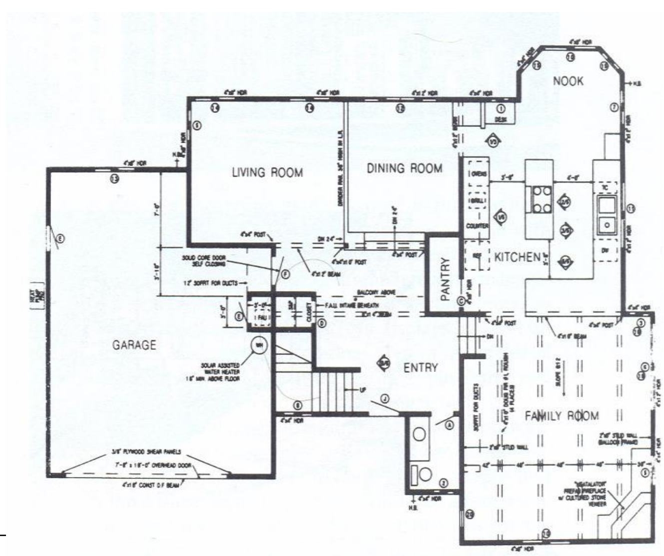




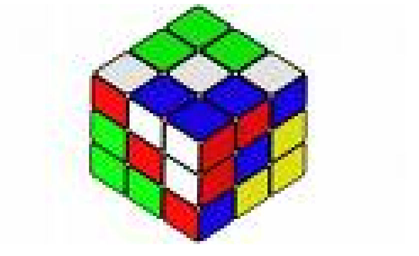
# Projecções Paralelas Ortogonais

Exemplos:

• **Planta**



# Projecções Paralelas Ortogonais



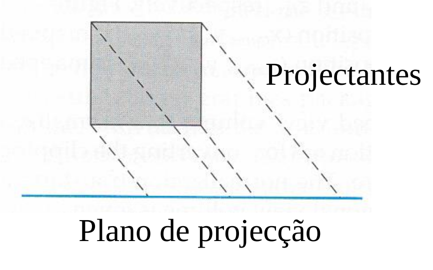
Exemplo:

- **Projecção Paralela Axonométrica Isométrica**

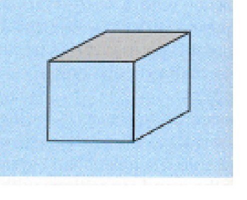


# Projecções Paralelas Oblíquas

Projecções paralelas obtidas por projectantes oblíquas ao plano de projecção




Projecção oblíqua do cubo: são mostradas várias faces

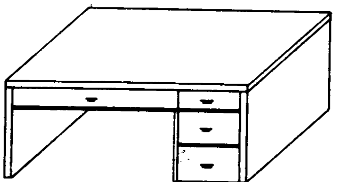


# Projecções Paralelas Oblíquas

**Cavaleira vs gabinete**

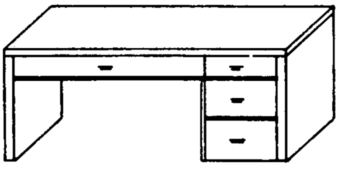
• **Projecção paralela oblíqua cavaleira (*cavalier*)**

Os comprimentos perpendiculares ao plano de projecção, quando projectados, mantêm a dimensão



• **Projecção paralela oblíqua gabinete (*cabinet*)**

Os comprimentos perpendiculares ao plano de projecção, quando projectados, são encurtados para metade.





# Referências


- Introduction to Computer Graphics,
  Foley, Van Dam, Feiner, Hughes, Phillips, Addison-Wesley, 1994
- Luís Veiga da Cunha, Desenho Técnico, Fundação Calouste Gulbenkian, Lisboa
- Interactive Computer Graphics- A Top-Down Approach Using OpenGL-
  Edward Angel, Addison-Wesley
- I. Carlbom, and J. Paciorek, “Planar Geometric Projections and Viewing Transformations”, Computing Surveys, 10(4), December 1978, pp 465-502
- Hearn, D., Baker, M. P., “Computer Graphics using OpenGL”, 3rd edition, Addison-Wesley, 2004
- João Cunha, Guiões Computação Gráfica, DI-FCUL
- Ana Paula Cláudio, Guiões Computação Gráfica, DI-FCUL
- Beatriz Sousa Santos, Guiões Computação Gráfica, LEET, Univ. Aveiro