Escuela Profesional de
Ciencia de la Computación

ICC Fase 1

Computer Graphics

Introduction to 3D modeling

MSc. Vicente Machaca Arceda

Universidad Nacional de San Agustín de Arequipa

April 14, 2021

Overview

vmachacaa@unsa.edu.pe

Table of Contents

vmachacaa@unsa.edu.pe

GPU

Definition

Graphics Processing Unit (GPU) is a programmable processor specialized for rendering all images on the computer's screen [1].

vmachacaa@unsa.edu.pe

GPU

GPU vs CPU

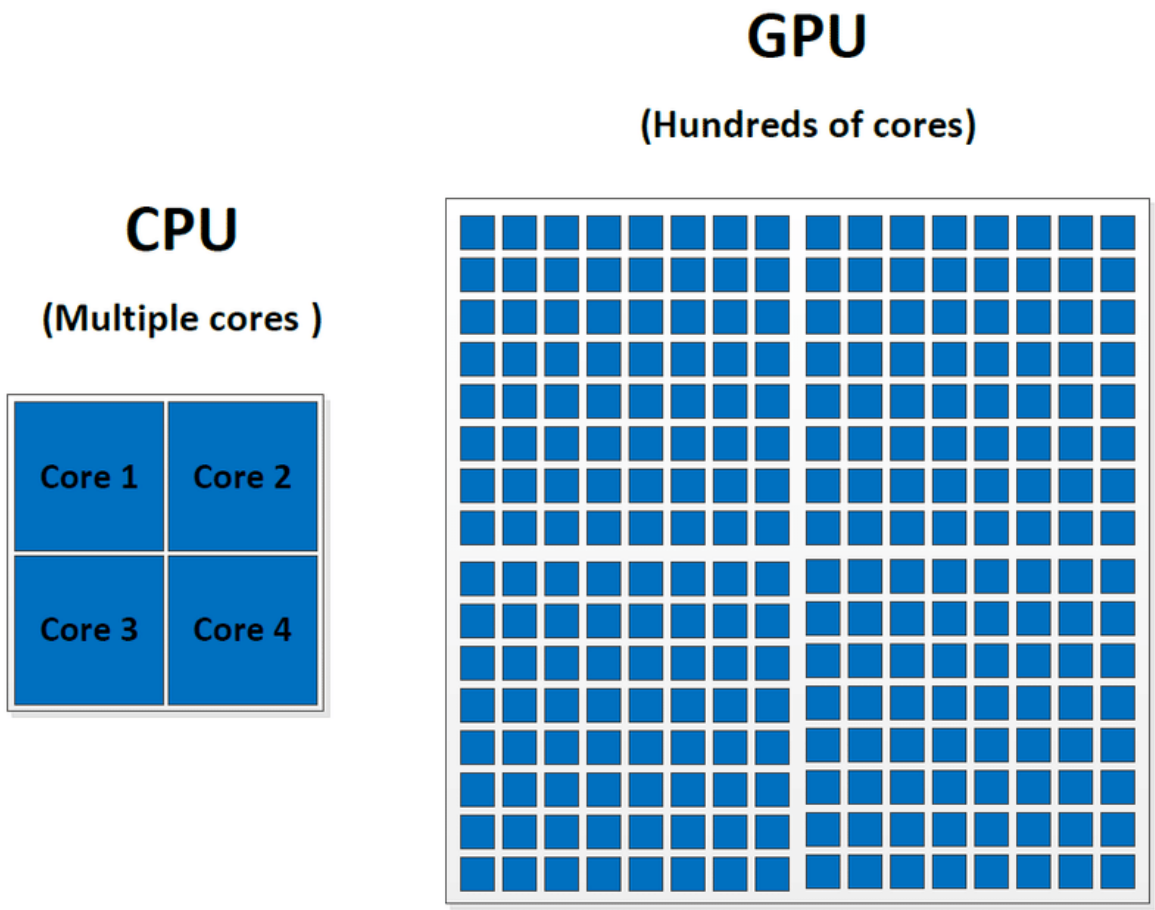


Figure: CPU vs GPU.

vmachacaa@unsa.edu.pe

GPU

GPU vs CPU

CPU is designed to handle a wide-range of tasks quickly but are **limited in concurrency**. A GPU is designed to quickly render high-resolution images and video **concurrently**.

vmachacaa@unsa.edu.pe

GPU

GPU vs CPU

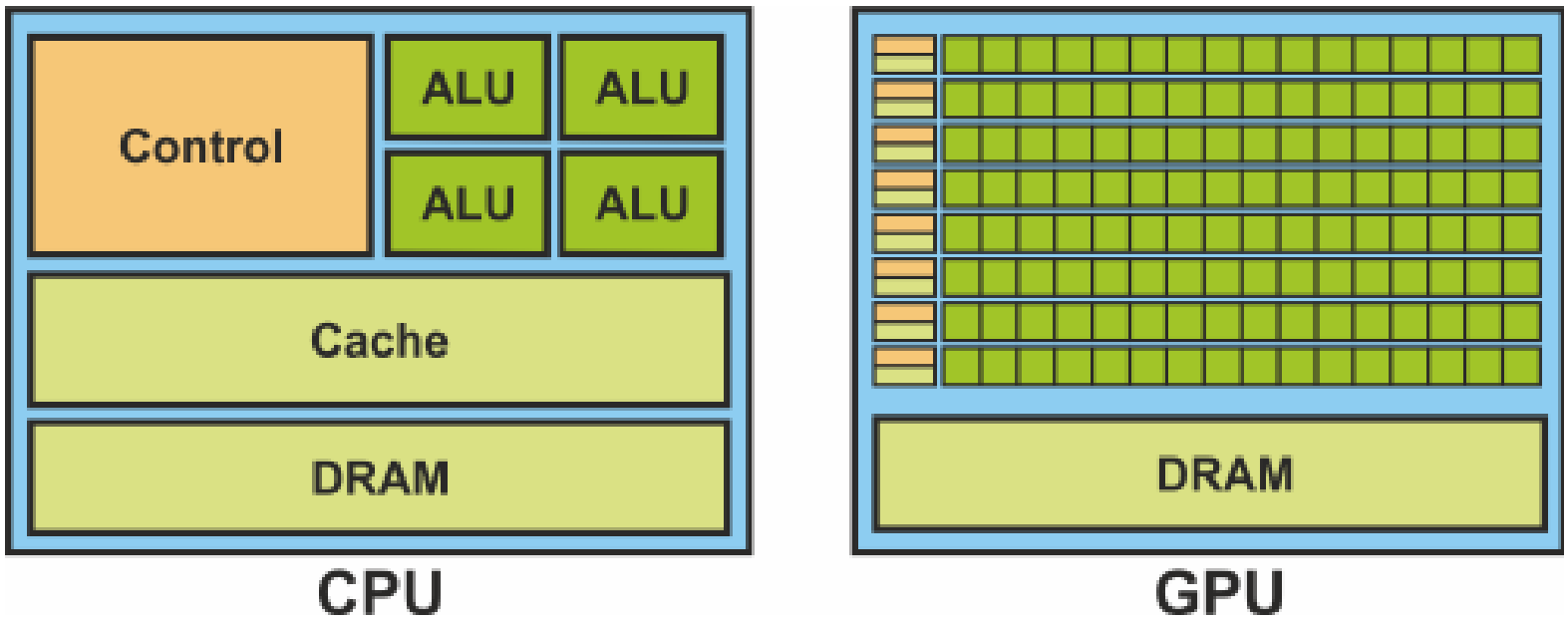


Figure: CPU vs GPU.

vmachacaa@unsa.edu.pe

GPU

GPU vs CPU

Video

vmachacaa@unsa.edu.pe

Table of Contents

vmachacaa@unsa.edu.pe

OpenGL

Definition

Open Graphics Library (OpenGL) is a cross-platform Application Programming Interface (API) for modeling, rendering and animation of 2D and 3D models [2].

OpenGL is liked a document that describe a set of functions. Each GPU builder must implement these functions [3].

By using OpenGL, a developer can use the same code to render graphics on a Mac, Windows, Linux, or mobile devices [3].

vmachacaa@unsa.edu.pe

Table of Contents

vmachacaa@unsa.edu.pe

PyOpenGL

PyOpenGL is the most common cross platform Python binding to OpenGL and related APIs. The binding is created using the standard ctypes library.

To install PyOpenGL:

```
pip3 install PyOpenGL PyOpenGL_accelerate
pip3 install pygame
```

vmachacaa@unsa.edu.pe

PyOpenGL

Hello World

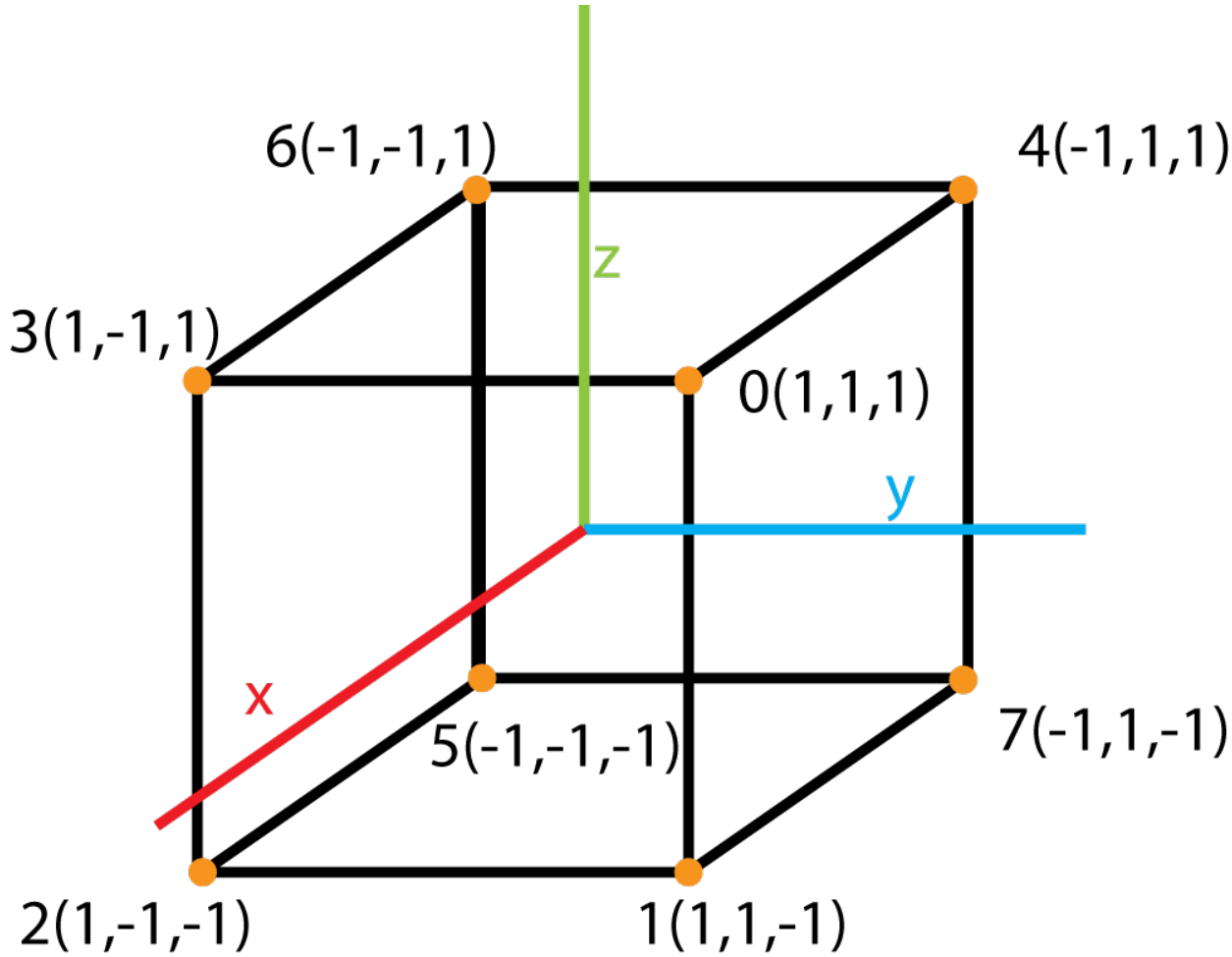
```
import pygame as pg
from pygame.locals import *

from OpenGL.GL import *
from OpenGL.GLU import *

cubeVertices = ((1,1,1),(1,1,-1),(1,-1,-1),
    (1,-1,1),(-1,1,1),(-1,-1,-1),
    (-1,-1,1),(-1,1,-1))
cubeEdges = ((0,1),(0,3),(0,4),(1,2),
    (1,7),(2,5),(2,3),(3,6),
    (4,6),(4,7),(5,6),(5,7))
cubeQuads = ((0,3,6,4),(2,5,6,3),(1,2,5,7),
    (1,0,4,7),(7,4,6,5),(2,3,0,1))

```

vmachacaa@unsa.edu.pe

PyOpenGL

Hello World

cubeVertices = ((1,1,1),(1,1,-1),(1,-1,-1),(1,-1,1),(-1,1,1),(-1,-1,-1),(-1,-1,1),(-1,1,-1))

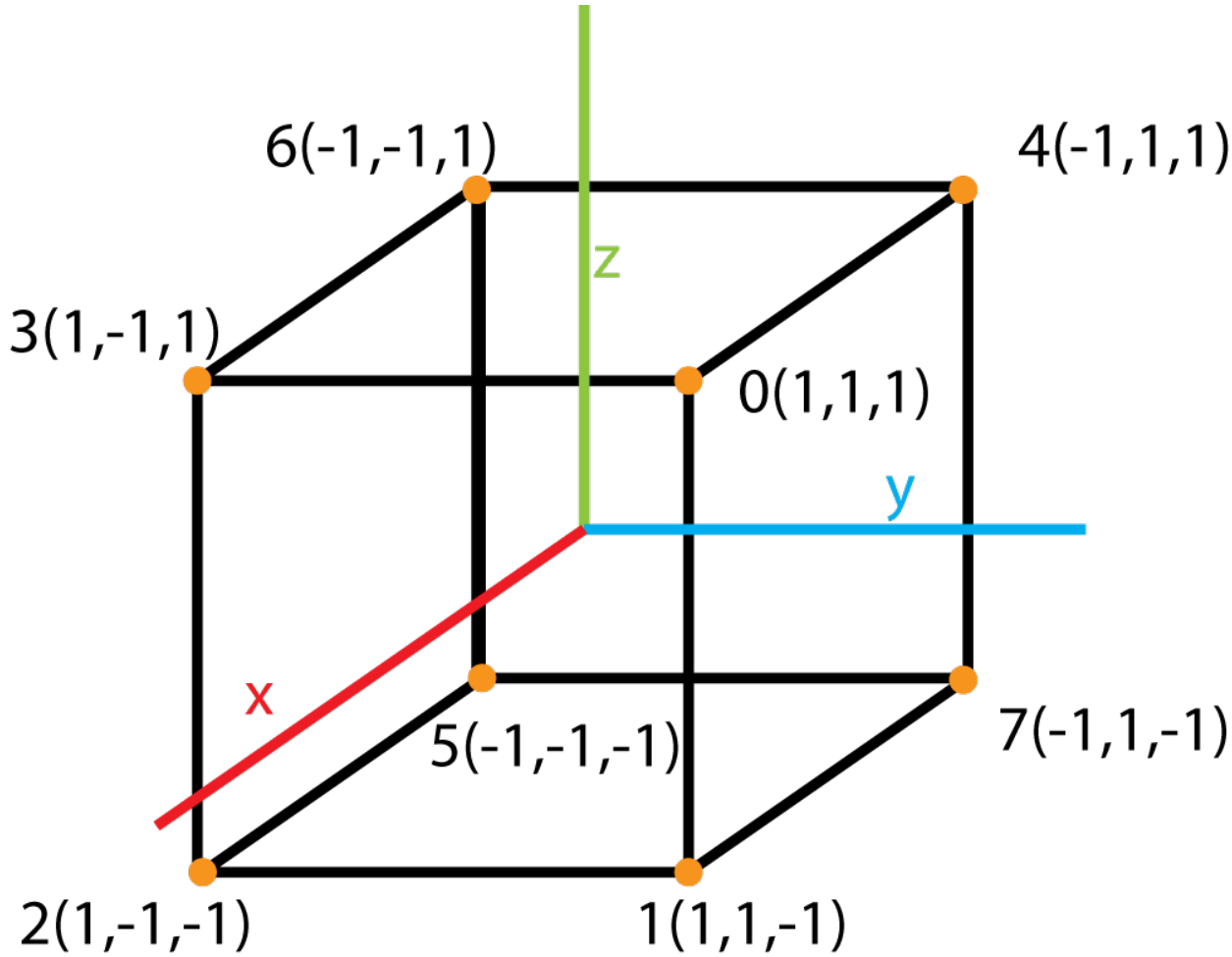


vmachacaa@unsa.edu.pe

PyOpenGL

Hello World

cubeEdges =
((0,1),(0,3),(0,4),(1,2),(1,7),(2,5),(2,3),(3,6),(4,6),(4,7),(5,6),(5,7))

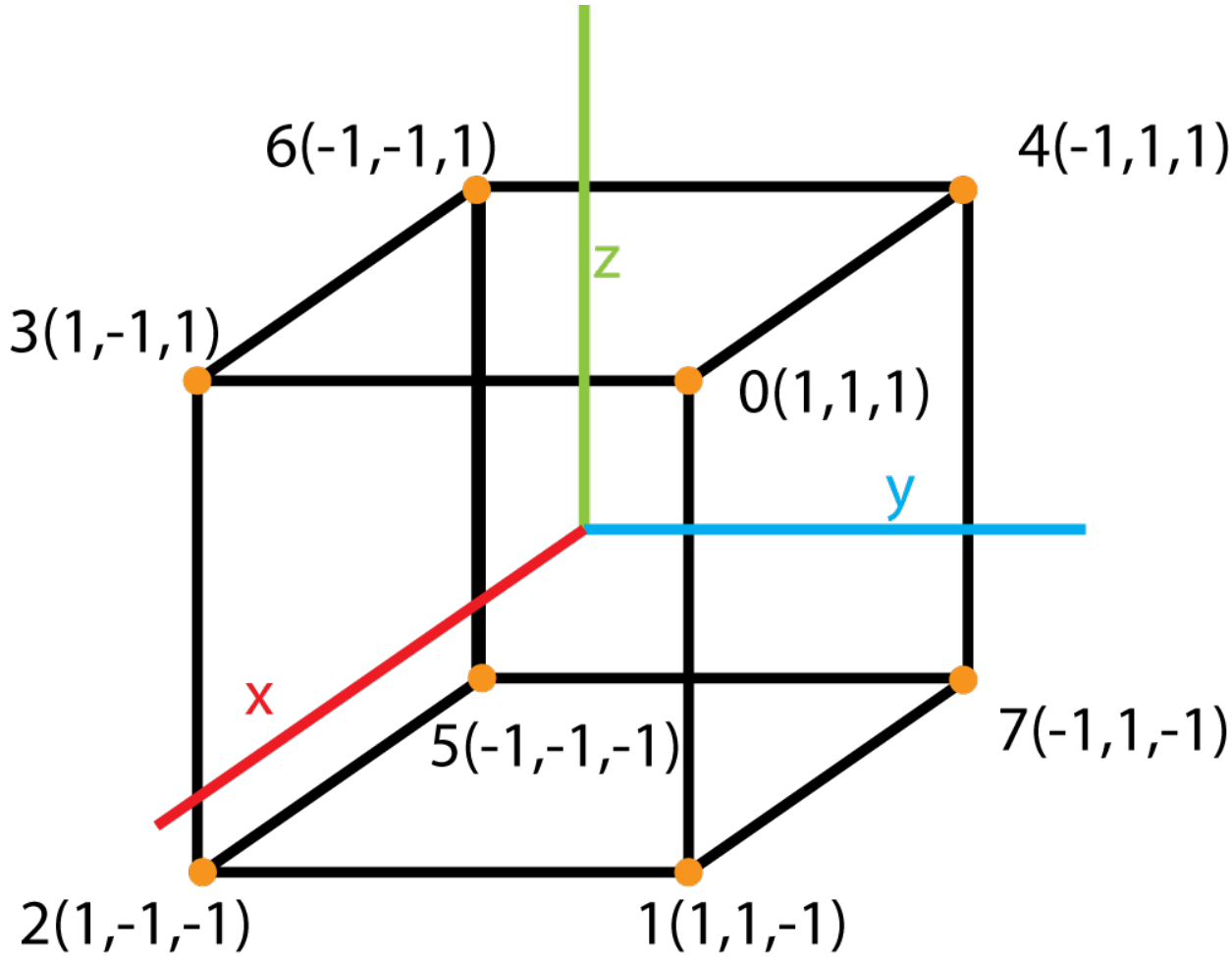


vmachacaa@unsa.edu.pe

PyOpenGL

Hello World

cubeQuads =
((0,3,6,4),(2,5,6,3),(1,2,5,7),(1,0,4,7),(7,4,6,5),(2,3,0,1))



vmachacaa@unsa.edu.pe

PyOpenGL

Hello World

```
def wireCube():
  glBegin(GL_LINES)
  for cubeEdge in cubeEdges:
    for cubeVertex in cubeEdge:
      glVertex3fv(cubeVertices[cubeVertex])
  glEnd()
```

vmachacaa@unsa.edu.pe

PyOpenGL

Hello World

```
def main():
    pg.init()
    display = (1680, 1050)
    pg.display.set_mode(display, DOUBLEBUF|OPENGL)
    gluPerspective(45, (display[0]/display[1]), 0.1, 50.0)
    glTranslatef(0.0, 0.0, -5)

    while True:
        for event in pg.event.get():
            if event.type == pg.QUIT:
                pg.quit()
            quit()
            glRotatef(1, 1, 1, 1)
            glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
            wireCube()
            pg.display.flip()
            pg.time.wait(10)

```

vmachacaa@unsa.edu.pe

PyOpenGL

Hello World

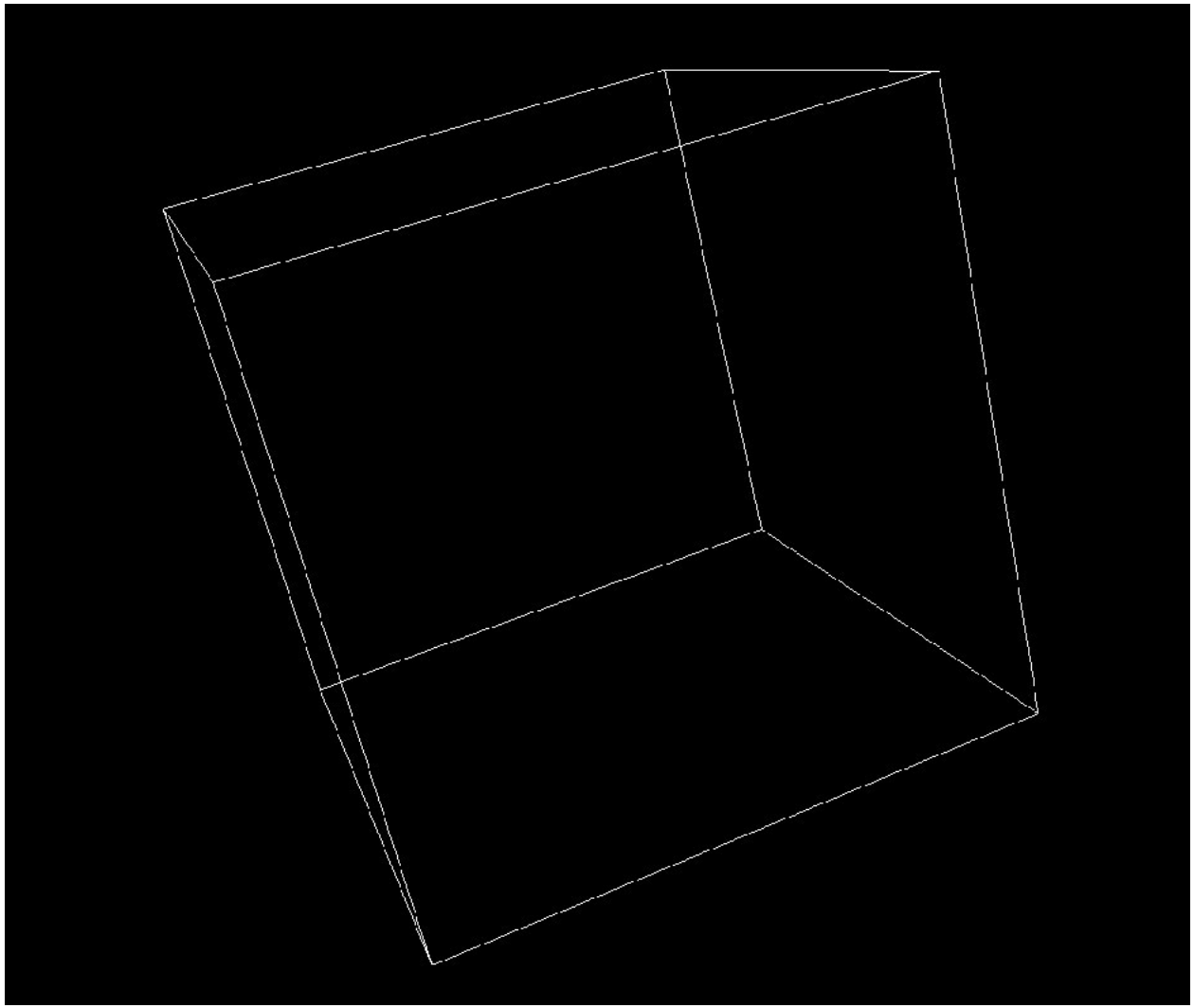


Figure: OpenGL hello world

vmachacaa@unsa.edu.pe

Table of Contents

vmachacaa@unsa.edu.pe

VTK

Definition

The Visualization Toolkit (VTK) is open source software for manipulating and displaying scientific data. It comes with state-of-the-art tools for 3D rendering, a suite of widgets for 3D interaction, and extensive 2D plotting capability.

vmachacaa@unsa.edu.pe

VTK

Samples

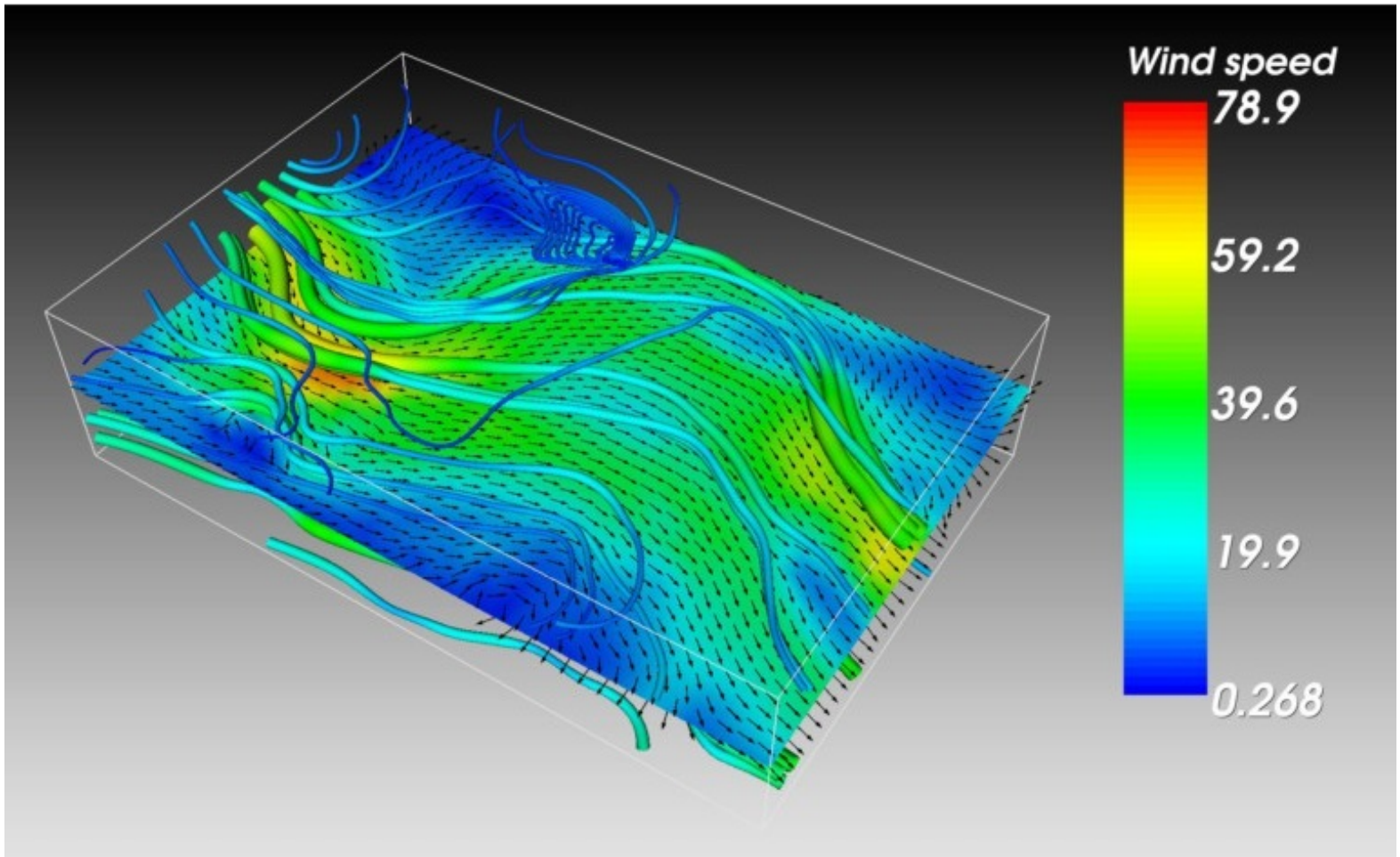


Figure: VTK wind speed sample

vmachacaa@unsa.edu.pe

VTK

Samples

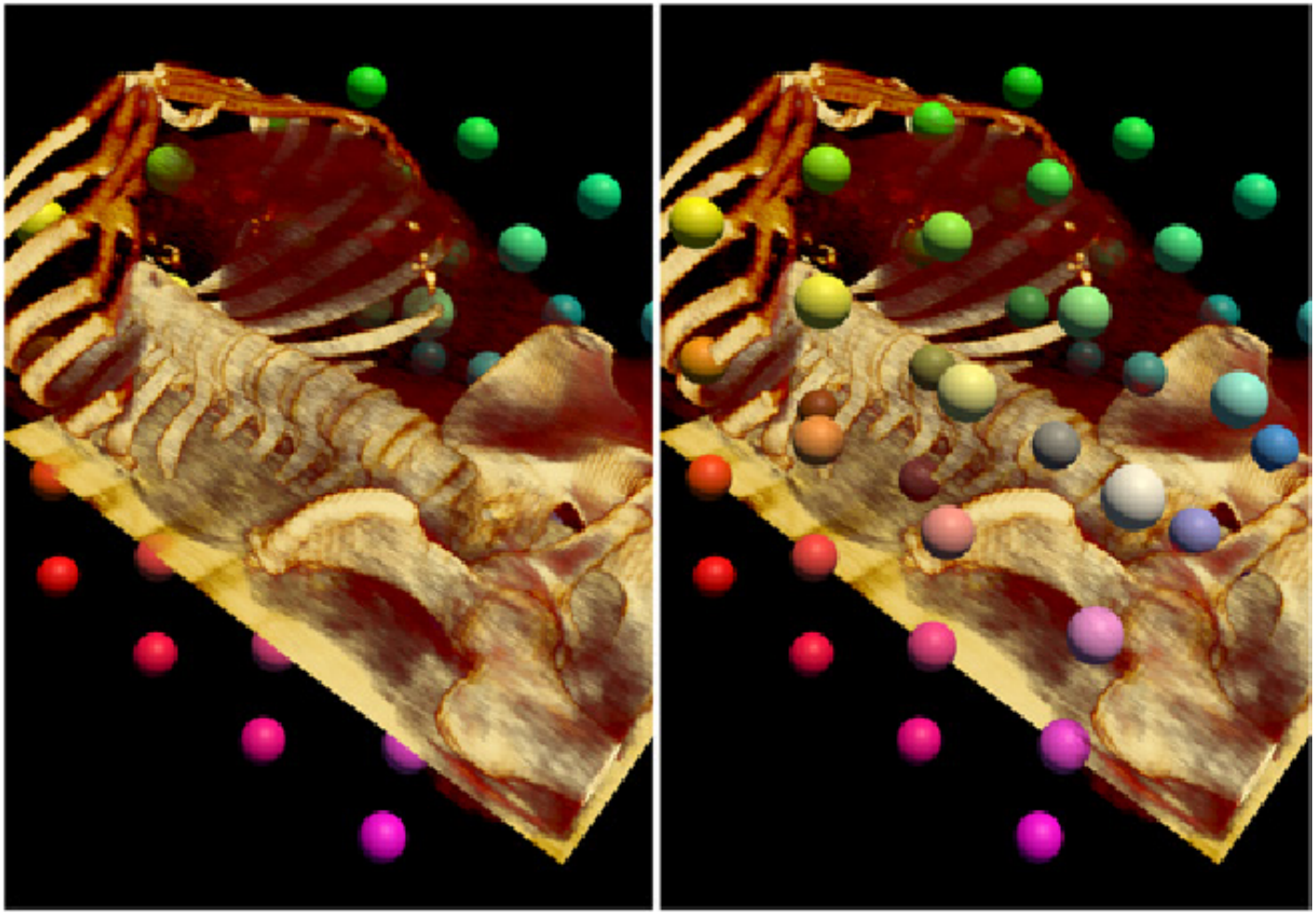


Figure: VTK in medicine.

vmachacaa@unsa.edu.pe

VTK

Samples

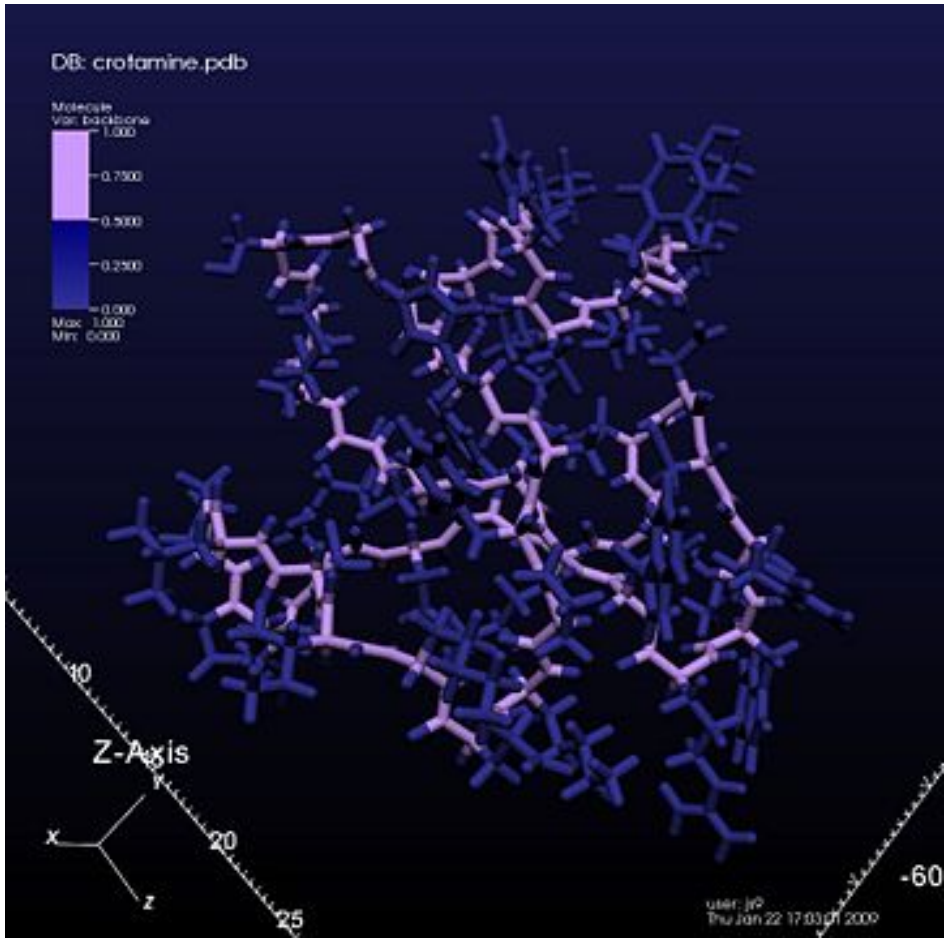


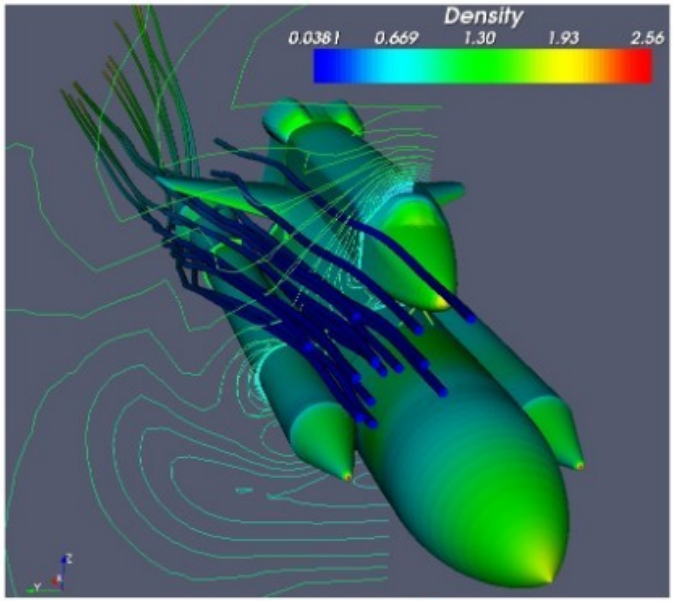
Figure: VTK for molecular visualization.

vmachacaa@unsa.edu.pe

VTK

Limitations

VTK has decent rendering performance and is good for rapid prototyping of 3D visualization tools. Not suitable for rendering large realistic 3D scenes with lots of dynamic content (i.e., games)



vmachacaa@unsa.edu.pe

Table of Contents

vmachacaa@unsa.edu.pe

VTK

Pipeline

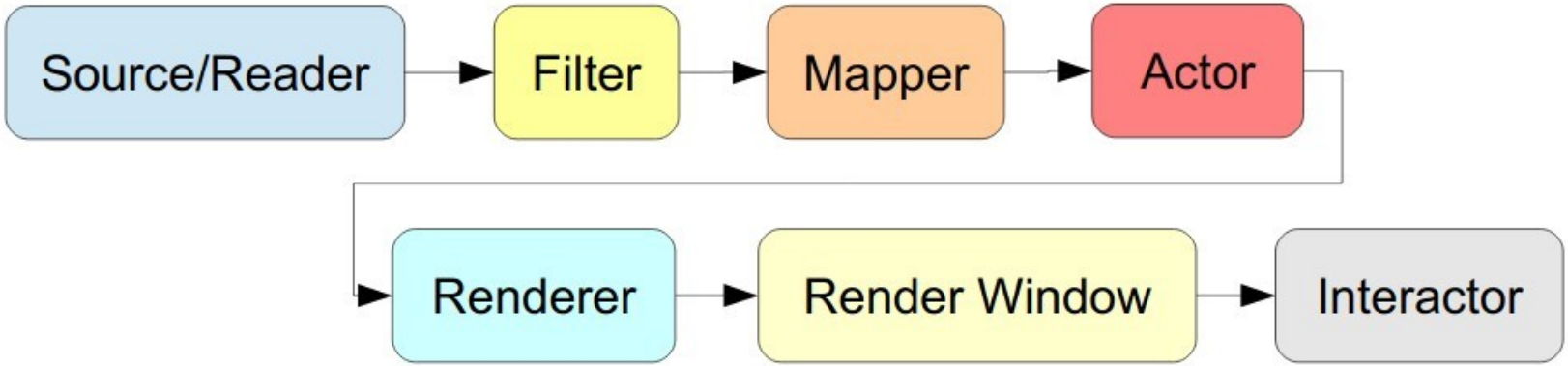


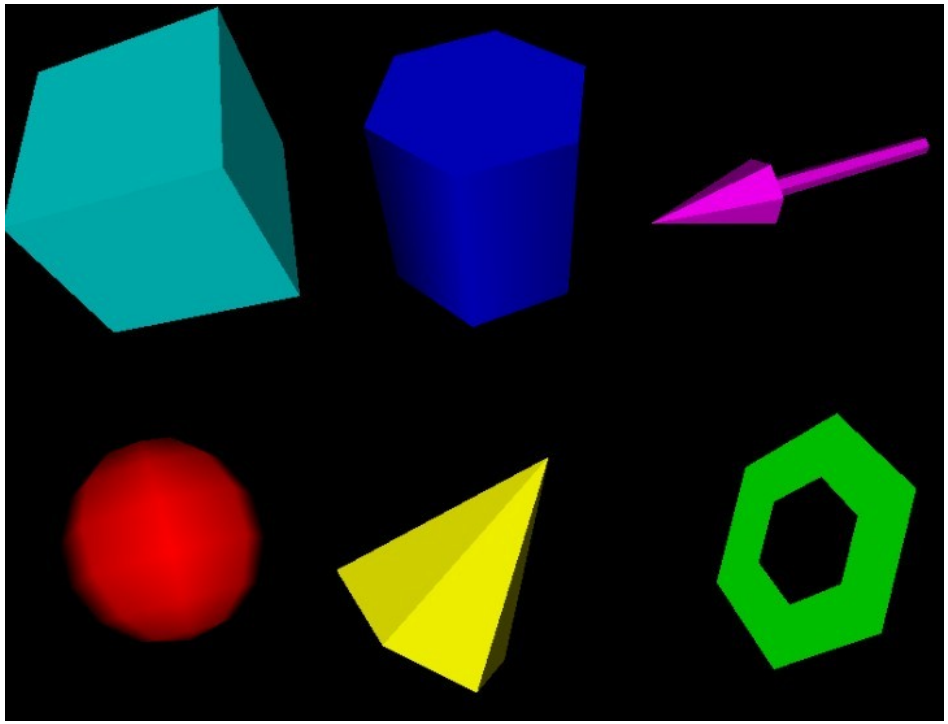
Figure: VTK pipeline.

vmachacaa@unsa.edu.pe

VTK

Source

VTK provides various source classes that can be used to construct simple geometric objects.



source/reader → filter→ mapper → actor → renderer → renderWindow → interactor

vmachacaa@unsa.edu.pe

VTK

Reader

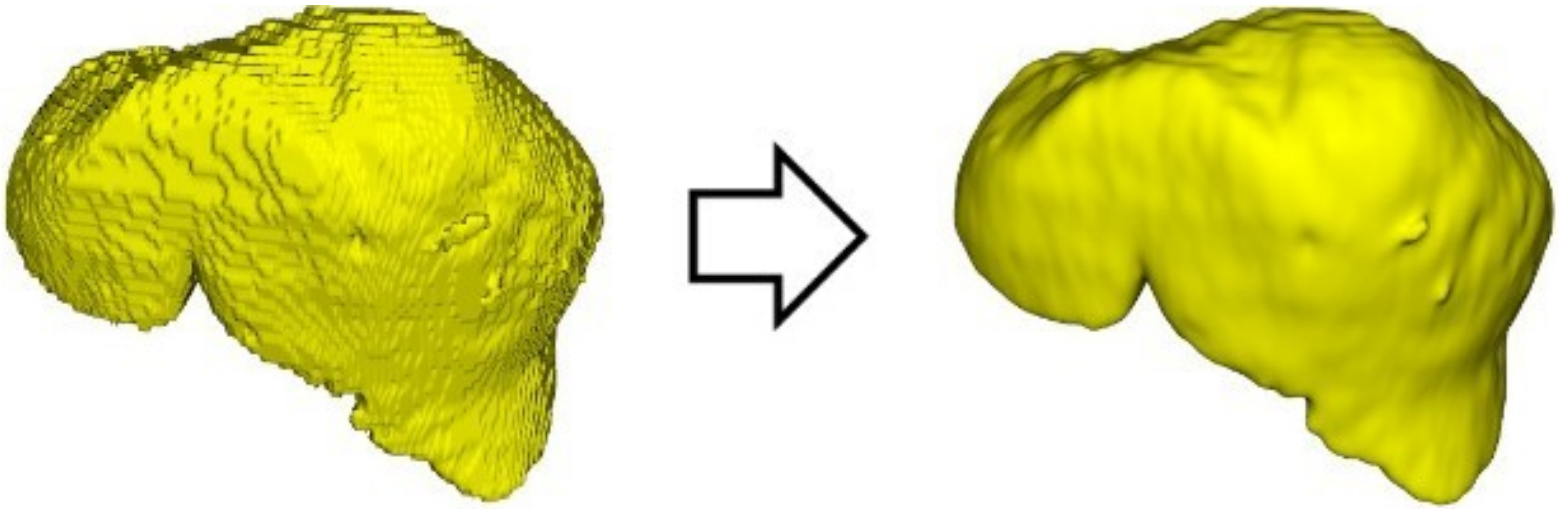
Reads data from file. You can use, e.g., **vtkStructuredPointsReader** to read a volumetric image from a .vtk file.

source/**reader** → filter→ mapper → actor → renderer → renderWindow → interactor

vmachacaa@unsa.edu.pe

VTK

Filters



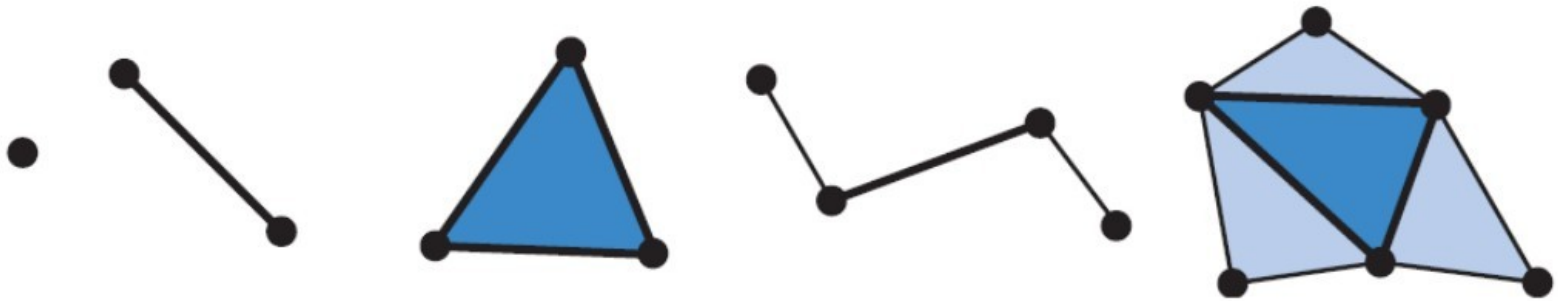
source/reader → **filter**→ mapper → actor → renderer → renderWindow → interactor

vmachacaa@unsa.edu.pe

VTK

Mappers

Maps data to graphics primitives (points, lines, or triangles) that can be displayed by the renderer.



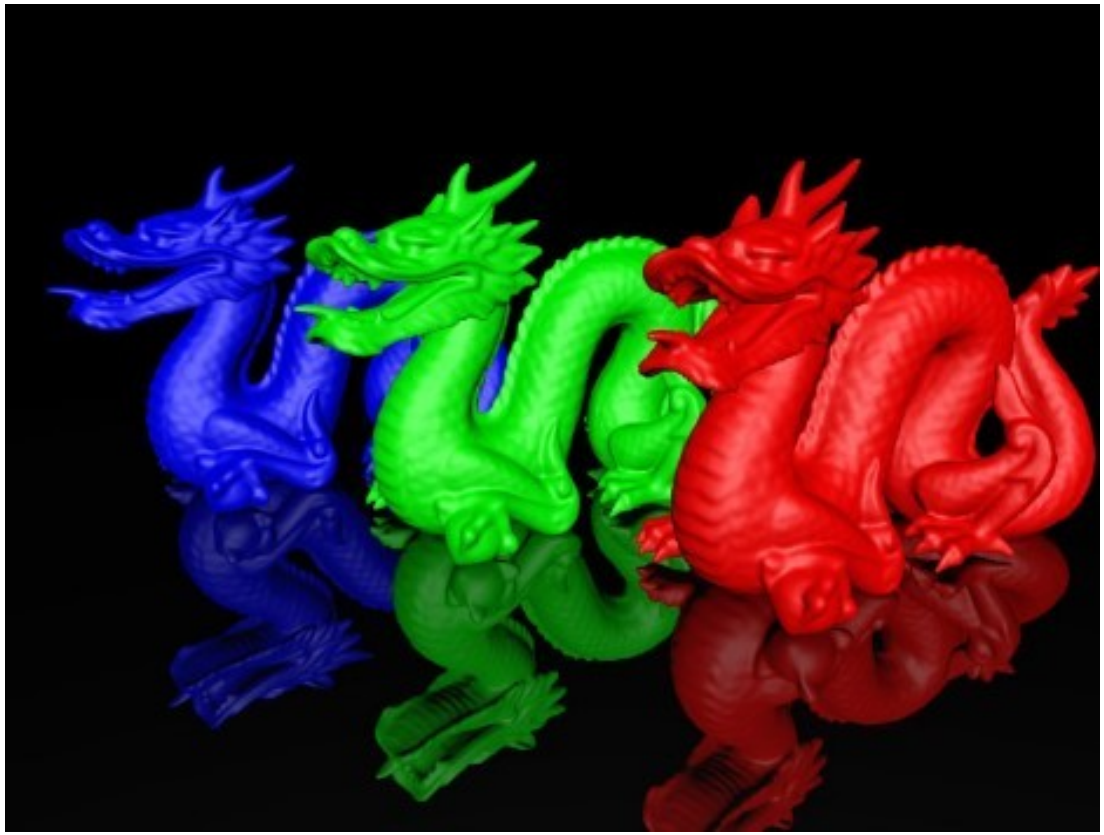
source/reader → filter→ **mapper** → actor → renderer → renderWindow → interactor

vmachacaa@unsa.edu.pe

VTK

Actor

vtkActor represents an object.

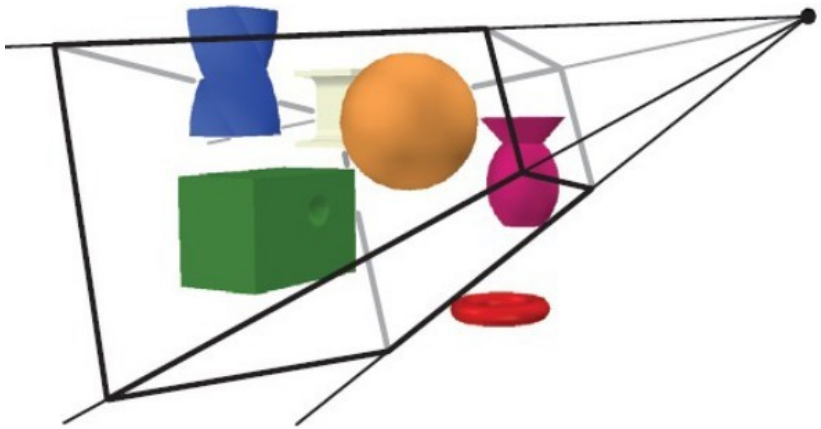


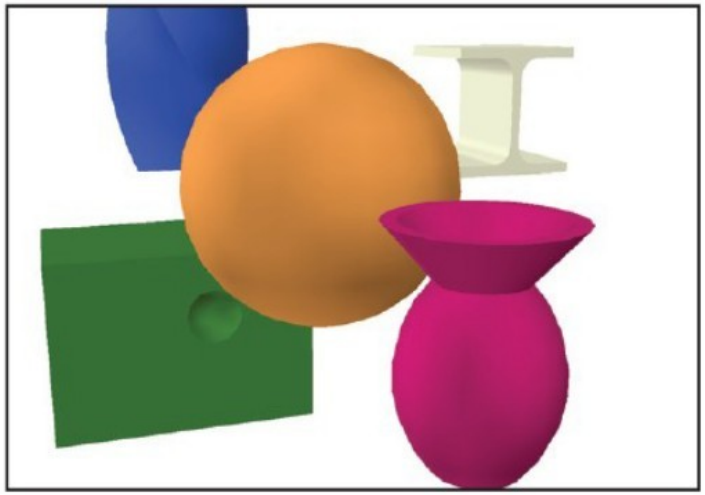
source/reader → filter→ mapper → **actor** → renderer → renderWindow → interactor

vmachacaa@unsa.edu.pe

VTK

Renderer



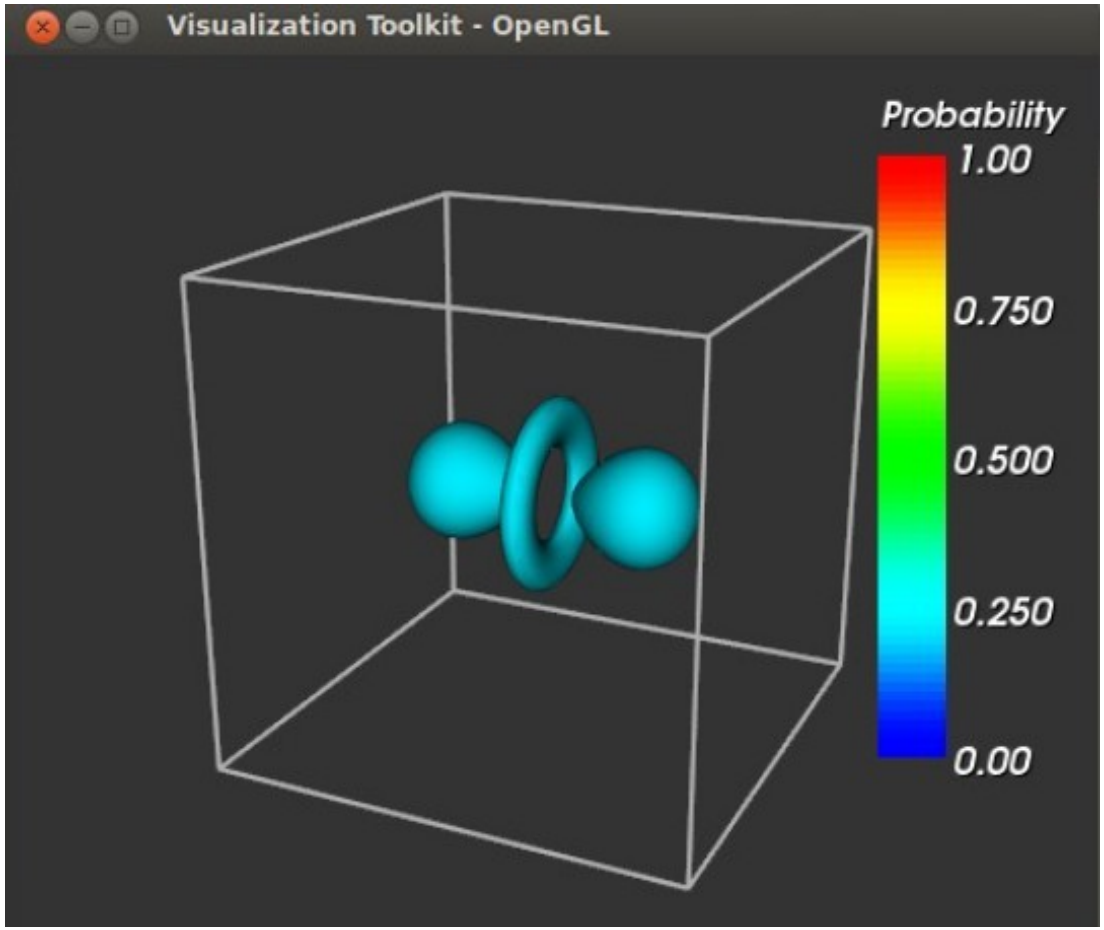


source/reader → filter→ mapper → actor → **renderer** → renderWindow → interactor

vmachacaa@unsa.edu.pe

VTK

RenderWindow



source/reader → filter→ mapper → actor → renderer → **renderWindow** → interactor

vmachacaa@unsa.edu.pe

VTK

Interactor

The vtkRenderWindowInteractor class provides platform-independent window interaction via the mouse and keyboard

source/reader $\rightarrow$ filter$\rightarrow$ mapper $\rightarrow$ actor $\rightarrow$ renderer $\rightarrow$ renderWindow $\rightarrow$ **interactor**

vmachacaa@unsa.edu.pe

Table of Contents

vmachacaa@unsa.edu.pe

VTK

Cube

to install VTK

```
pi3 install vtk
```

vmachacaa@unsa.edu.pe

VTK

Cube

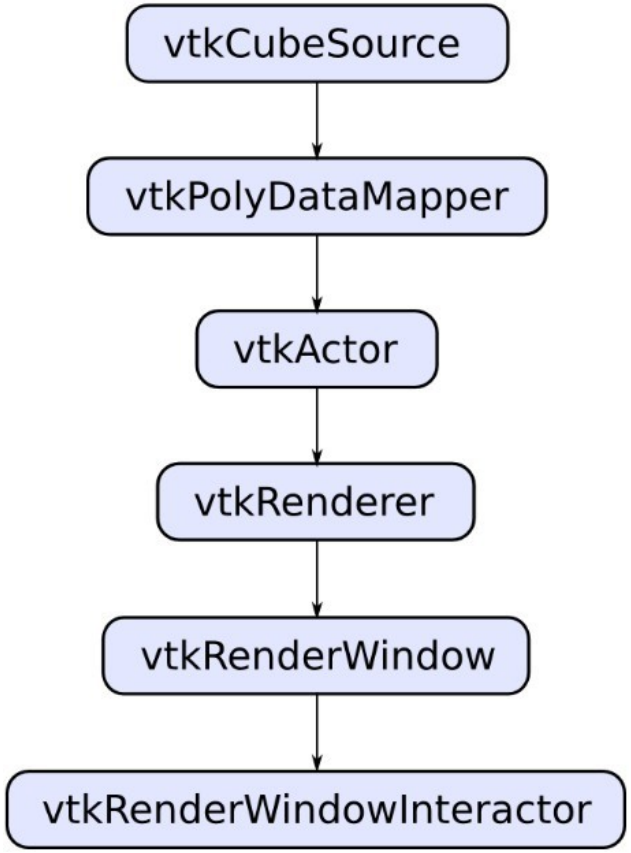


Figure: VTK pipeline.

vmachacaa@unsa.edu.pe

VTK

Cube

```
import vtk
# source
cube = vtk.vtkCubeSource()
cube.Update()

# mapper
cube_mapper = vtk.vtkPolyDataMapper()
if vtk.VTK_MAJOR_VERSION <= 5:
  cube_mapper.SetInput(cube.GetOutput())
else:
  cube_mapper.SetInputData(cube.GetOutput())

# actor
cube_actor = vtk.vtkActor()
cube_actor.SetMapper(cube_mapper)
cube_actor.GetProperty().SetColor(1.0, 0.0, 0.0)
```

vmachacaa@unsa.edu.pe

VTK

Cube

```
#renderer
renderer = vtk.vtkRenderer()
renderer.SetBackground(0.0, 0.0, 0.0)
renderer.AddActor(cube_actor)

#renderWindow
render_window = vtk.vtkRenderWindow()
render_window.SetWindowName('Simple VTK scene')
render_window.SetSize(400, 400)
render_window.AddRenderer(renderer)

#interactor
interactor = vtk.vtkRenderWindowInteractor()
interactor.SetRenderWindow(render_window)
```

vmachacaa@unsa.edu.pe

VTK

Cube

```
# Initialize the interactor and
# start the rendering loop
interactor.Initialize()
render_window.Render()
interactor.Start()

```

vmachacaa@unsa.edu.pe

VTK

Cube

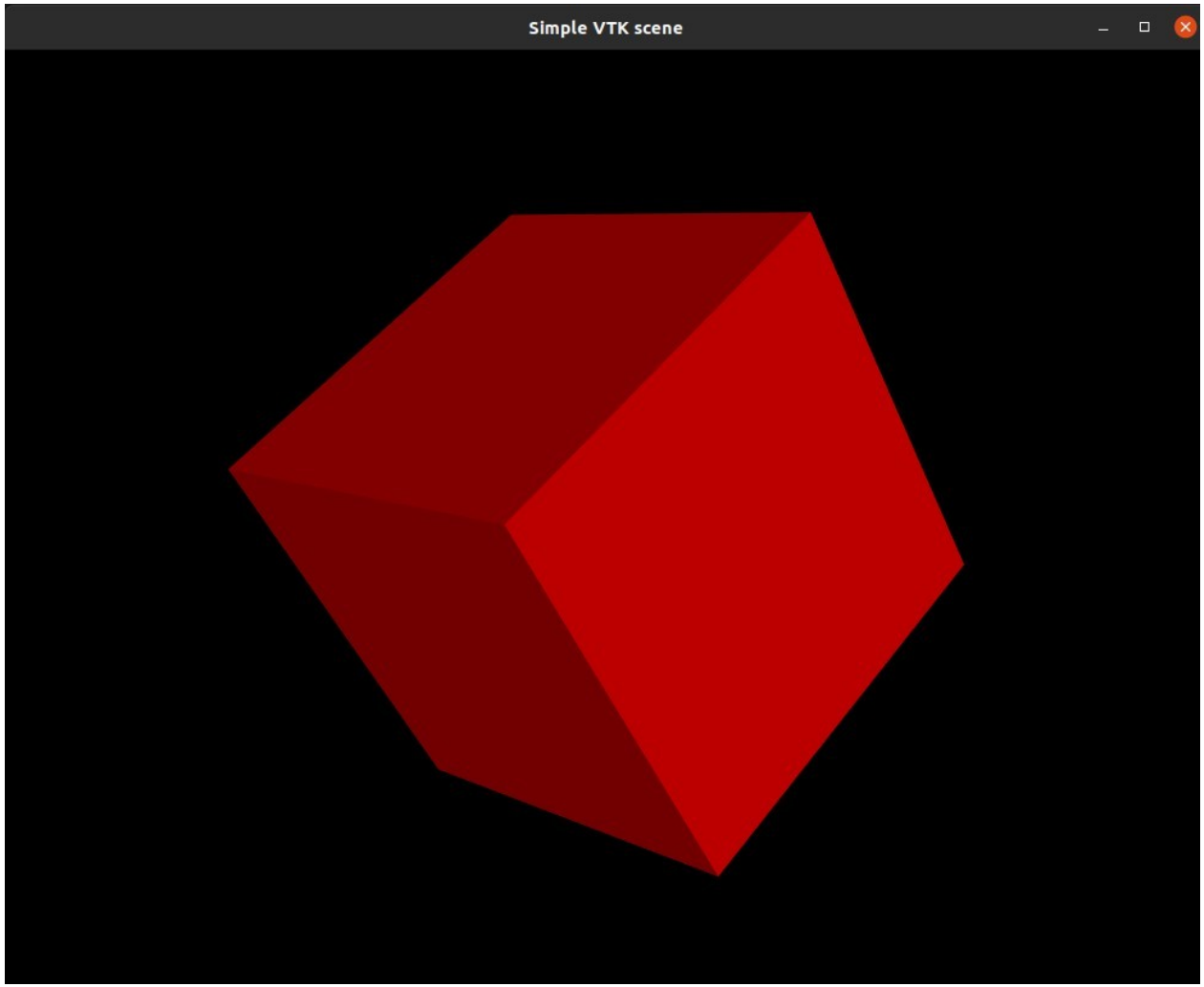


Figure: VTK cube.

vmachacaa@unsa.edu.pe

Table of Contents

vmachacaa@unsa.edu.pe

VTK

Cylinder

```
cylinder = vtk.vtkCylinderSource()
cylinder.SetRadius(20)
cylinder.SetHeight(50)
cylinder.SetResolution(10)
cylinder.Update()

mapper = vtk.vtkPolyDataMapper()
mapper.SetInputData(cylinder.GetOutput())

actor = vtk.vtkActor()
actor.SetMapper(cube_mapper)
actor.GetProperty().SetColor(0.0, 1.0, 0.0)
actor.RotateX(30.0)
```

vmachacaa@unsa.edu.pe

VTK

Cylinder

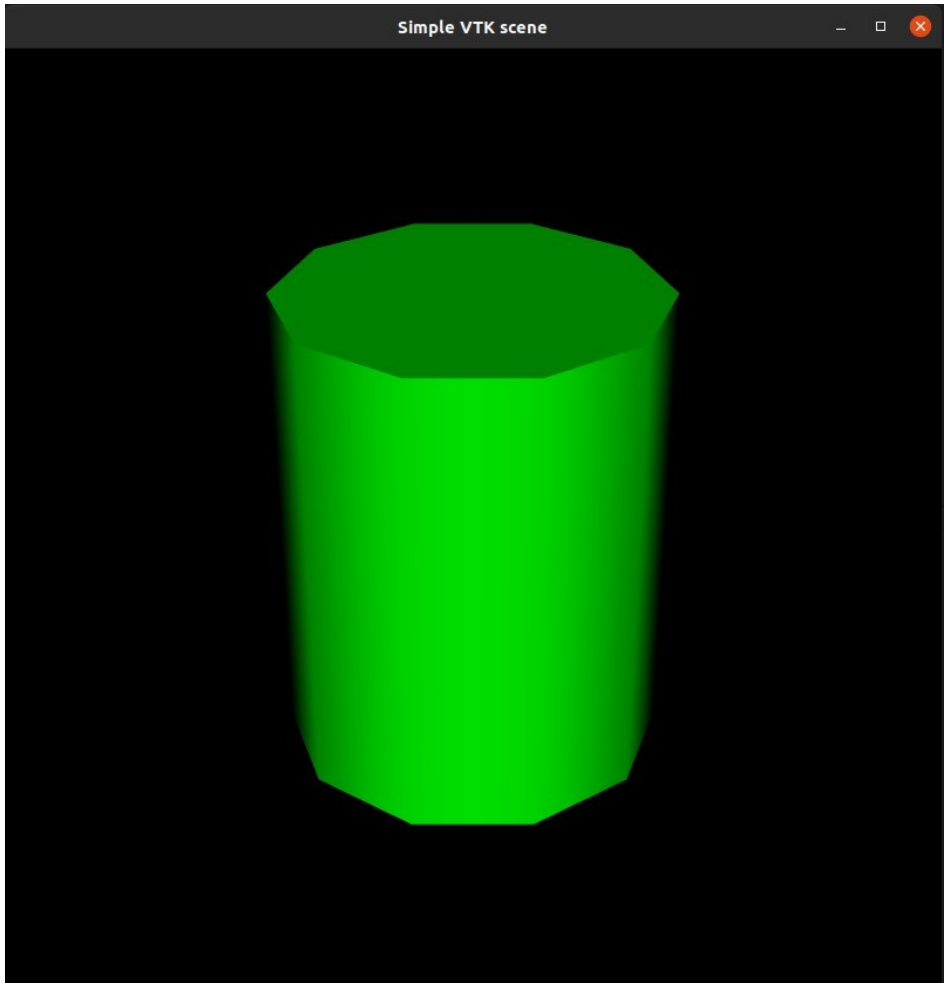


Figure: VTK cylinder.

vmachacaa@unsa.edu.pe

Table of Contents

vmachacaa@unsa.edu.pe

VTK

Axes

```
cylinder = vtk.vtkCylinderSource()
cylinder.SetRadius(0.2)
cylinder.SetHeight(0.5)
cylinder.SetResolution(10)
cylinder.Update()

mapper = vtk.vtkPolyDataMapper()
mapper.SetInputData(cylinder.GetOutput())

actor = vtk.vtkActor()
actor.SetMapper(mapper)
actor.GetProperty().SetColor(0.0, 1.0, 0.0)

```

vmachacaa@unsa.edu.pe

VTK

Axes

```
#axes
transform = vtk.vtkTransform()
transform.Translate(0.0, 0.0, 0.0)
axes = vtk.vtkAxesActor()
axes.SetUserTransform(transform)

#renderer
renderer = vtk.vtkRenderer()
renderer.SetBackground(0.0, 0.0, 0.0)
renderer.AddActor(actor)
renderer.AddActor(axes)

```

vmachacaa@unsa.edu.pe

VTK

Axes

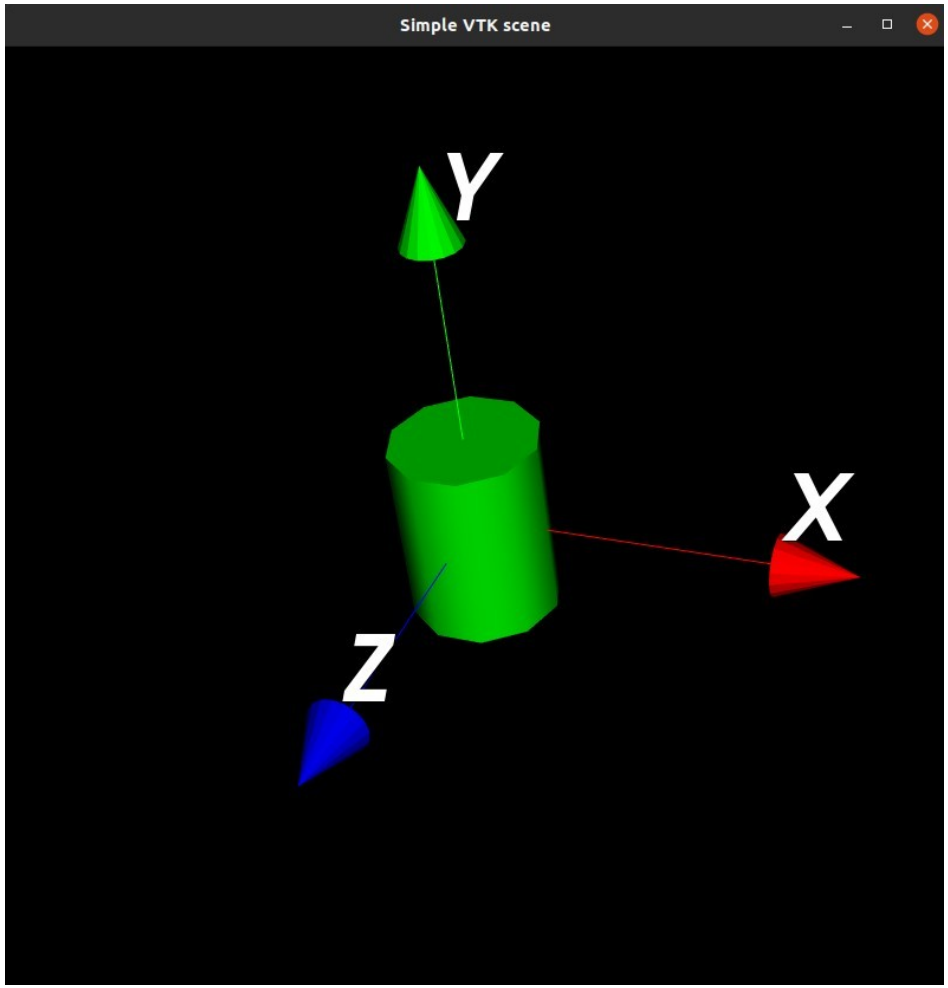


Figure: VTK axes.

vmachacaa@unsa.edu.pe

Table of Contents

vmachacaa@unsa.edu.pe

VTK

Camera

```
#camera
camera = vtk.vtkCamera()
camera.SetFocalPoint(0,0,0)
camera.SetPosition(10,10,10)

#renderer
renderer = vtk.vtkRenderer()
renderer.SetBackground(0.0, 0.0, 0.0)
renderer.AddActor(actor)
renderer.AddActor(axes)
renderer.SetActiveCamera(camera)
```

vmachacaa@unsa.edu.pe

VTK

Camera

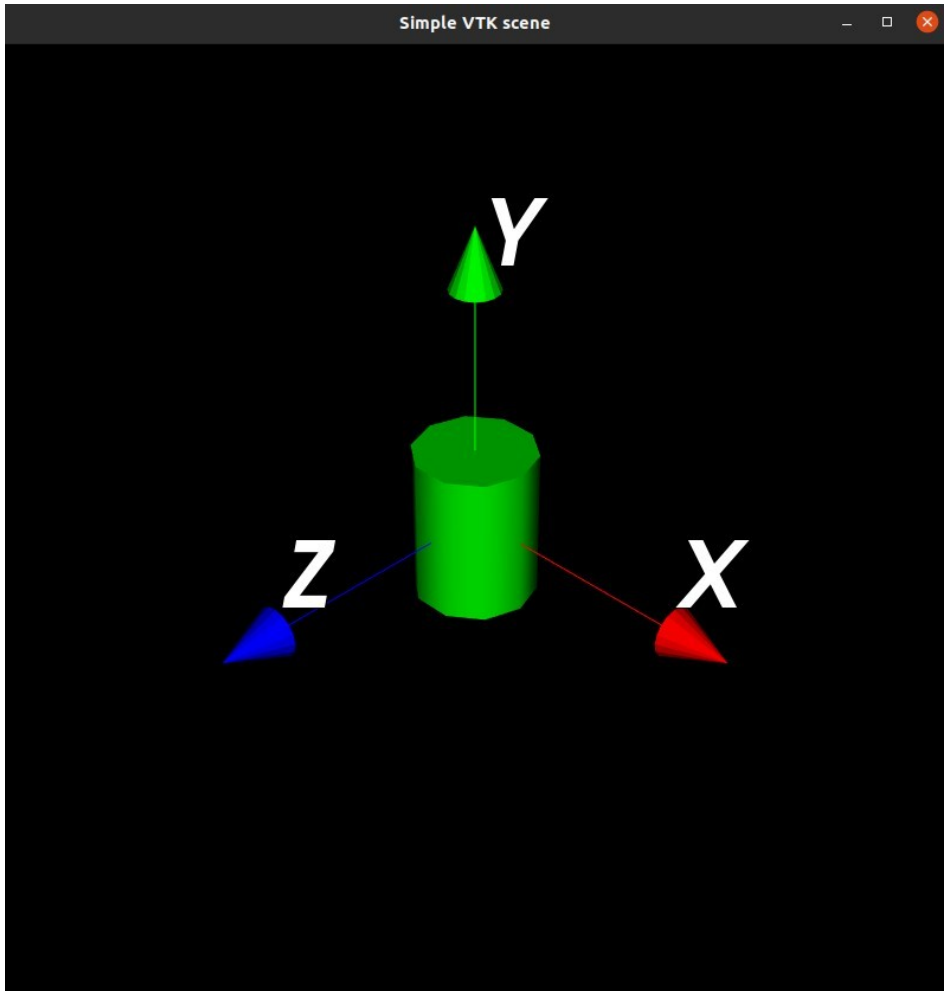


Figure: VTK camera.

vmachacaa@unsa.edu.pe

VTK

Book and Documentation

VTK user's guide

Documentation

vmachacaa@unsa.edu.pe

Table of Contents

vmachacaa@unsa.edu.pe

VTK

Exercise

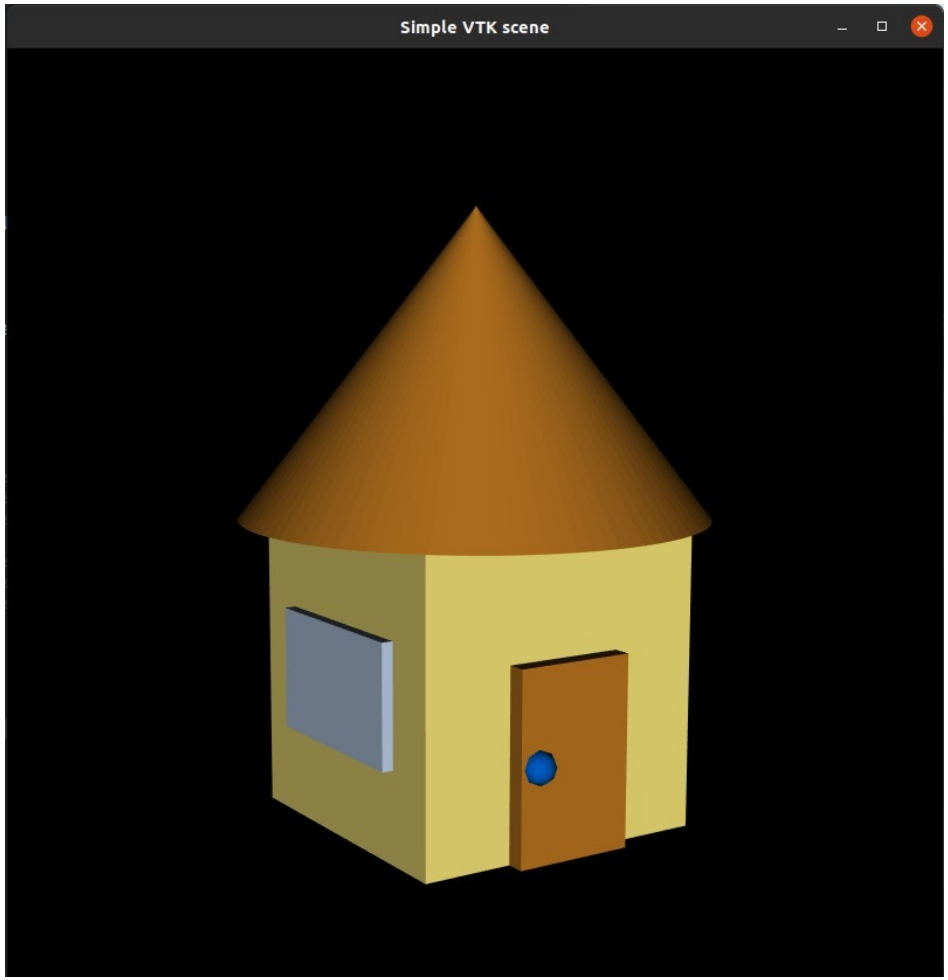


Figure: VTK exercise.

vmachacaa@unsa.edu.pe

References I


pcmag, “Computer science dictionary.” [Online]. Available: https://www.pcmag.com/encyclopedia/term/gpu


M. Segal and K. Akeley, “The opengl graphics system: A specification (version 4.0 (core profile), mar. 2012.”


V. S. Gordon and J. L. Clevenger, *Computer Graphics Programming in OpenGL with C++*. Stylus Publishing, LLC, 2020.

vmachacaa@unsa.edu.pe

Questions?

vmachacaa@unsa.edu.pe