**Akademia Górniczo-Hutnicza**

**Wydział Informatyki**

Projekt nr 3
Z przedmiotu Informatyka Medyczna

# *"Wizualizacja obrazów medycznych w 3D"*

Autor: *Jacek Tyszkiewicz*
Kierunek studiów: *Informatyka*

Kraków, 2025

# Spis treści

# 1. Wizualizacja obrazów w mecycynie

## 1.1 Cel projektu:

Celem projektu jest zapoznanie się z metodami wizualizacji 3D w medycynie.

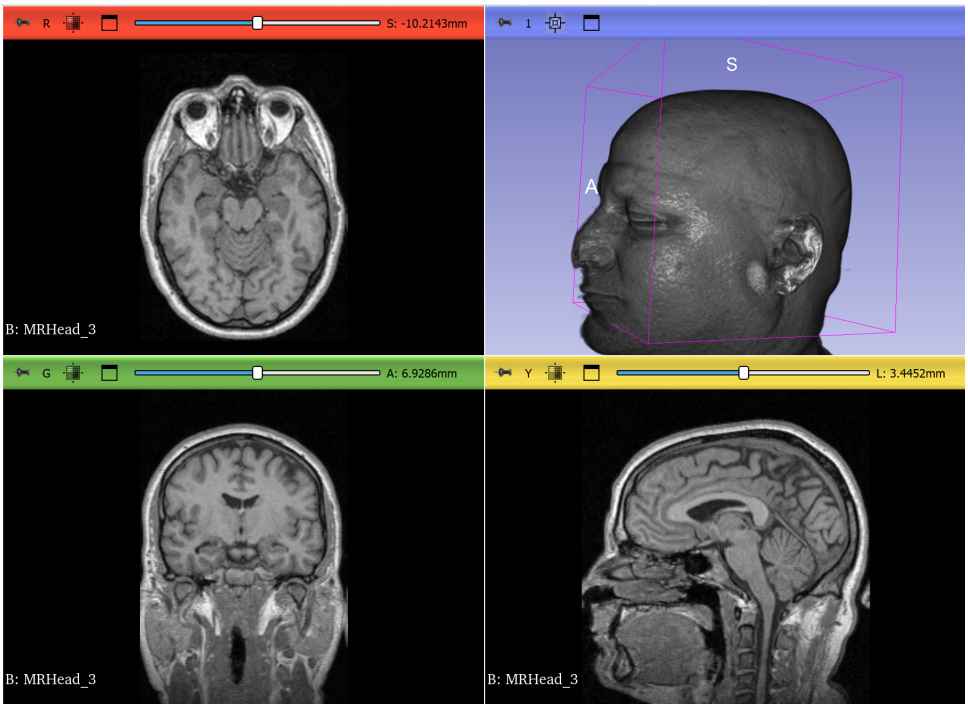
## 1.2 ZADANIE 1 - Slicer 3D(1 pkt)

– Zainstaluj **Slicer 3D**

  – https://download.slicer.org/

– Uruchom wizualizację 3D jednego z przykładowych zestawów danych

Umieść w raporcie zrzut ekranu z wizualizacją




**Rys. 1.1.** Wizualizacja programu 3D Slider

## 1.3 ZADANIE 2 (2 pkt)

- Zaimplementuj wizualizację za pomocą **izopowierzchni**.
- Dodaj suwak umożliwiający zmianę wartości `iso`.
- Wykorzystaj:
  - `vtkContourFilter` (ustawienie wartości `iso`)
  - `vtkColorTransferFunction` (dodanie punktów RGB)
  - `vtkPolyDataMapper`
  - `vtkActor()` (zamiast `vtkImageActor`)
- Umieść w raporcie wynikową wizualizację.

Najważniejsze fragmenty kodu:

Utworzenie izopowierzchni

```
contour = vtk.vtkContourFilter()
contour.SetInputConnection(reader.GetOutputPort())
contour.SetValue(0, 100)
```

Dodanie kolorów RBG do powierzchni - zależnej od intensywności

```
colorFunc = vtk.vtkColorTransferFunction()
colorFunc.AddRGBPoint(0, 0.0, 0.0, 1.0)
colorFunc.AddRGBPoint(500, 1.0, 1.0, 0.0)
colorFunc.AddRGBPoint(1000, 1.0, 0.0, 0.0)
```

Mapowanie danych 3D oraz przypisanie im kolorów

```
mapper = vtk.vtkPolyDataMapper()
mapper.SetInputConnection(contour.GetOutputPort())
mapper.SetLookupTable(colorFunc)
mapper.SetScalarRange(imageData.GetScalarRange())
```

Renderowanie i utworzenie okna

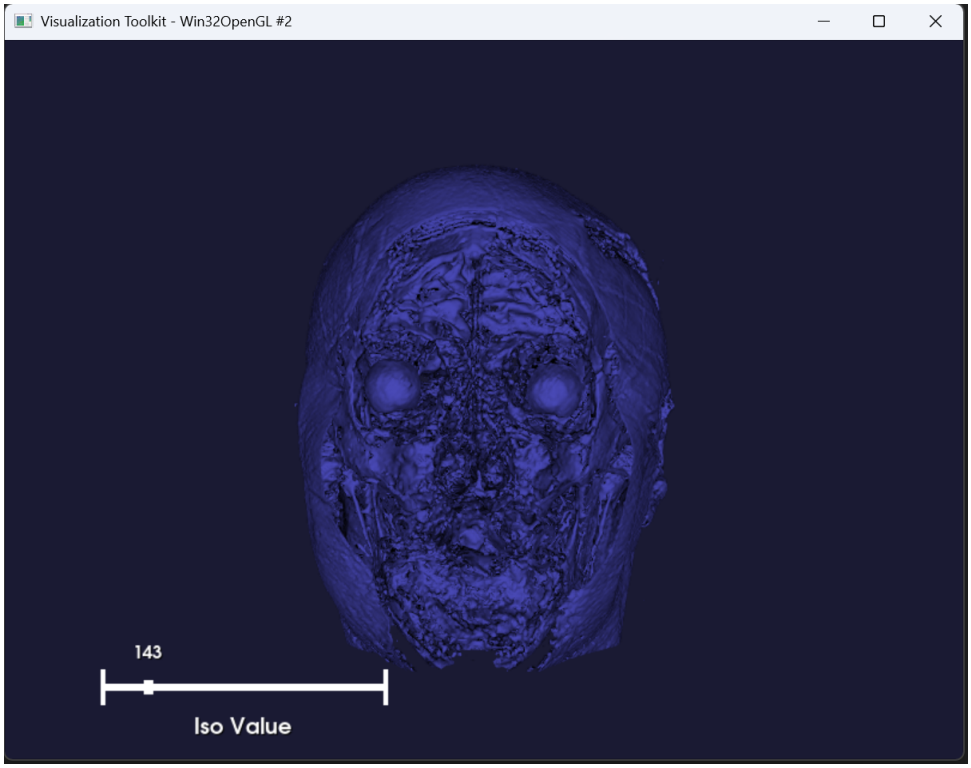
```
renderer = vtk.vtkRenderer()
renderer.SetBackground(0.1, 0.1, 0.2)
renderer.AddActor(actor)

renderWindow = vtk.vtkRenderWindow()
renderWindow.AddRenderer(renderer)
```

```
renderWindow.SetSize(800, 600)

interactor = vtk.vtkRenderWindowInteractor()
interactor.SetRenderWindow(renderWindow)
```

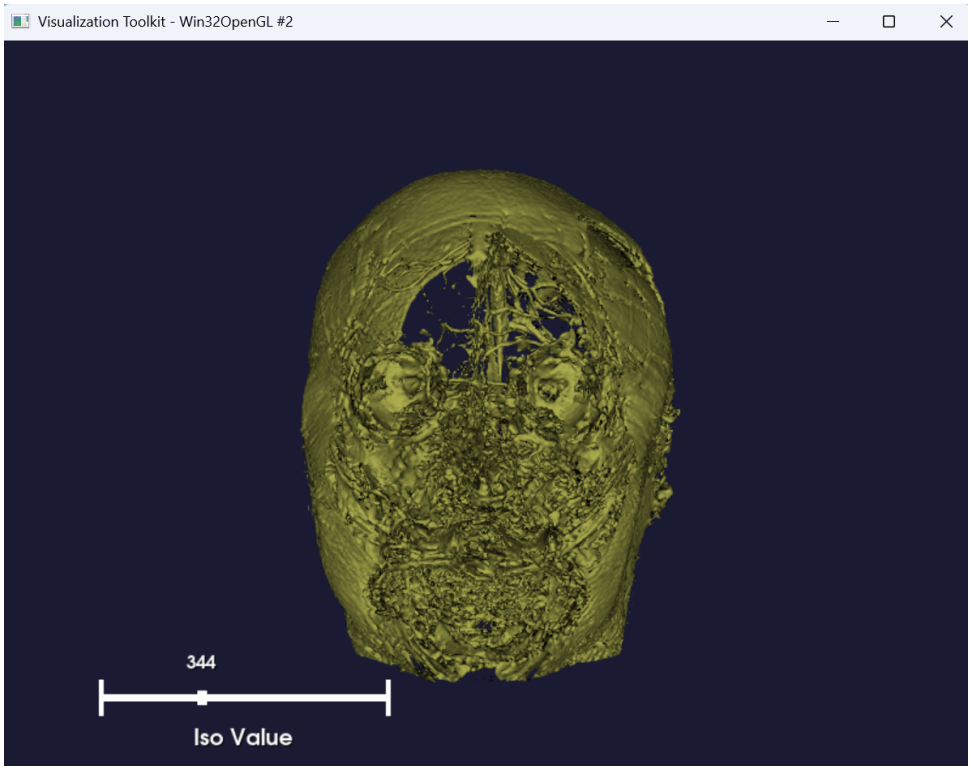
Klasa reprezentująca suwak do zmiany wartości iso

```
class IsoCallback:
    def __init__(self, contour, renderWindow):
        self.contour = contour
        self.renderWindow = renderWindow
    def __call__(self, caller, event):
        value = caller.GetSliderRepresentation().GetValue()
        self.contour.SetValue(0, value)
        self.renderWindow.Render()

sliderRep = vtk.vtkSliderRepresentation2D()
# ustawienia pozycji, zakresu, etykiety...
sliderRep.SetMinimumValue(0)
sliderRep.SetMaximumValue(1000)
sliderRep.SetValue(100)
sliderRep.SetTitleText("Iso Value")

sliderWidget = vtk.vtkSliderWidget()
sliderWidget.SetInteractor(interactor)
sliderWidget.SetRepresentation(sliderRep)
sliderWidget.SetAnimationModeToAnimate()
sliderWidget.EnabledOn()
sliderWidget.AddObserver("InteractionEvent", IsoCallback(contour,
    renderWindow))
```

Wyniki działania programu:




**Rys. 1.2.** Wyniki działania programu 2




**Rys. 1.3.** Wyniki działania programu 2

## 1.4 ZADANIE 3 (2 pkt)

- Zaimplementuj wizualizację za pomocą bezpośredniego **renderowania objętości**.
- Dodaj suwak umożliwiający zmianę punktu w funkcji transferu (kolor lub przezroczystość).
- Wykorzystaj:
  - `vtkSmartVolumeMapper`
  - `vtk.vtkVolume` *(aktor!)*
  - `vtkColorTransferFunction`
  - `vtkPiecewiseFunction` *(funkcja przezroczystości)*
- Umieść w raporcie wynikową wizualizację.

Automatyczne mapowanie danych objętościowych.

```
volumeMapper = vtk.vtkSmartVolumeMapper()
volumeMapper.SetInputConnection(reader.GetOutputPort())
```

Sworzenie aktora objętościowego, który służy do wyświetlania danych medycznych w formie trójwymiarowej objętości (ang. volume rendering).

```
volume = vtk.vtkVolume()
volume.SetMapper(volumeMapper)
volume.SetProperty(volumeProperty)
```

Kolorowanie obiektu na podstawie wartości intensywności

```
colorFunc = vtk.vtkColorTransferFunction()
colorFunc.AddRGBPoint(0, 0.0, 0.0, 0.0)
colorFunc.AddRGBPoint(500, 1.0, 1.0, 0.0)
colorFunc.AddRGBPoint(1000, 1.0, 0.0, 0.0)
```

funkcja przezroczystości, która określa jakie wartości intensywności są bardziej lub mniej przezroczyste.

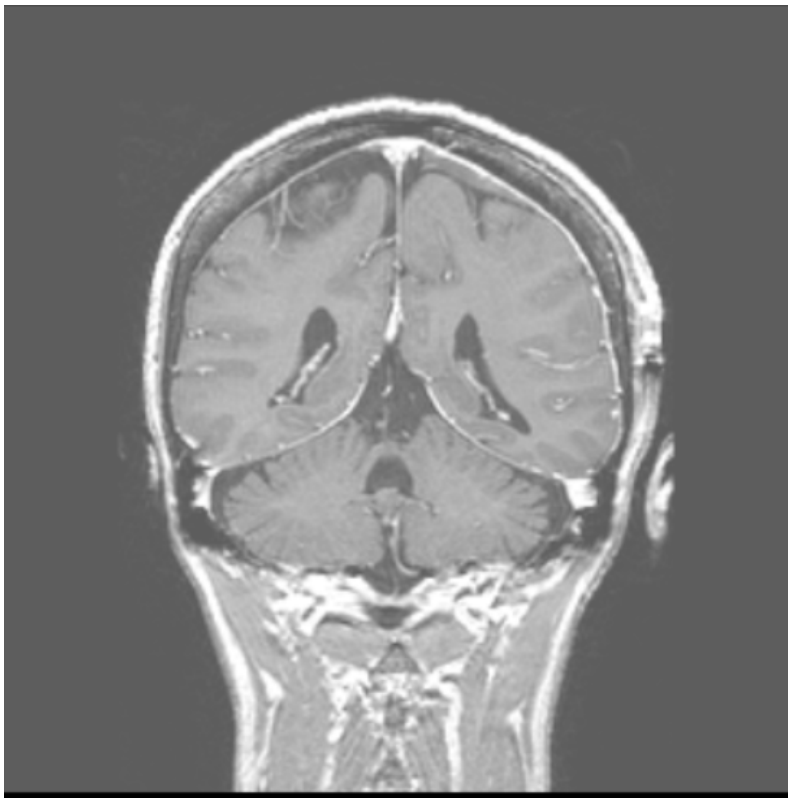
```
opacityFunc = vtk.vtkPiecewiseFunction()
opacityFunc.AddPoint(0, 0.0)
opacityFunc.AddPoint(500, 0.3)
opacityFunc.AddPoint(1000, 0.9)
```

Połączenie funkcji kolorów i przezroczystości:

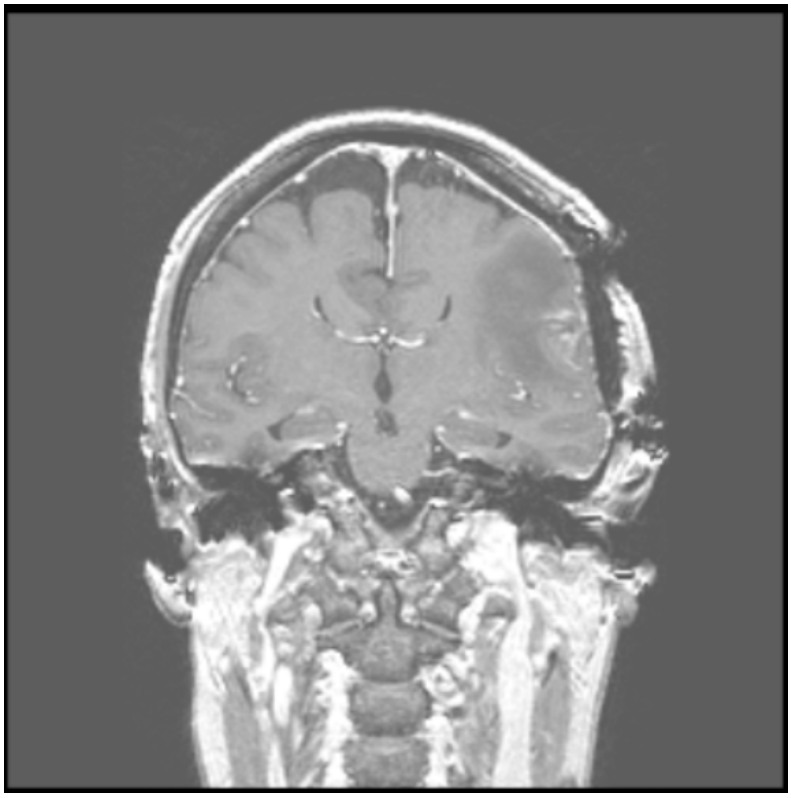
```
volumeProperty = vtk.vtkVolumeProperty()
volumeProperty.SetColor(colorFunc)
volumeProperty.SetScalarOpacity(opacityFunc)
volumeProperty.ShadeOn()
```

Suwak

```
class OpacityCallback:
    def __call__(self, caller, event):
        value = caller.GetSliderRepresentation().GetValue()
        opacityFunc.RemoveAllPoints()
        opacityFunc.AddPoint(value, 0.8)
        renderWindow.Render()
```



**Rys. 1.4.** Wyniki działania programu 3



**Rys. 1.5.** Wyniki działania programu 3