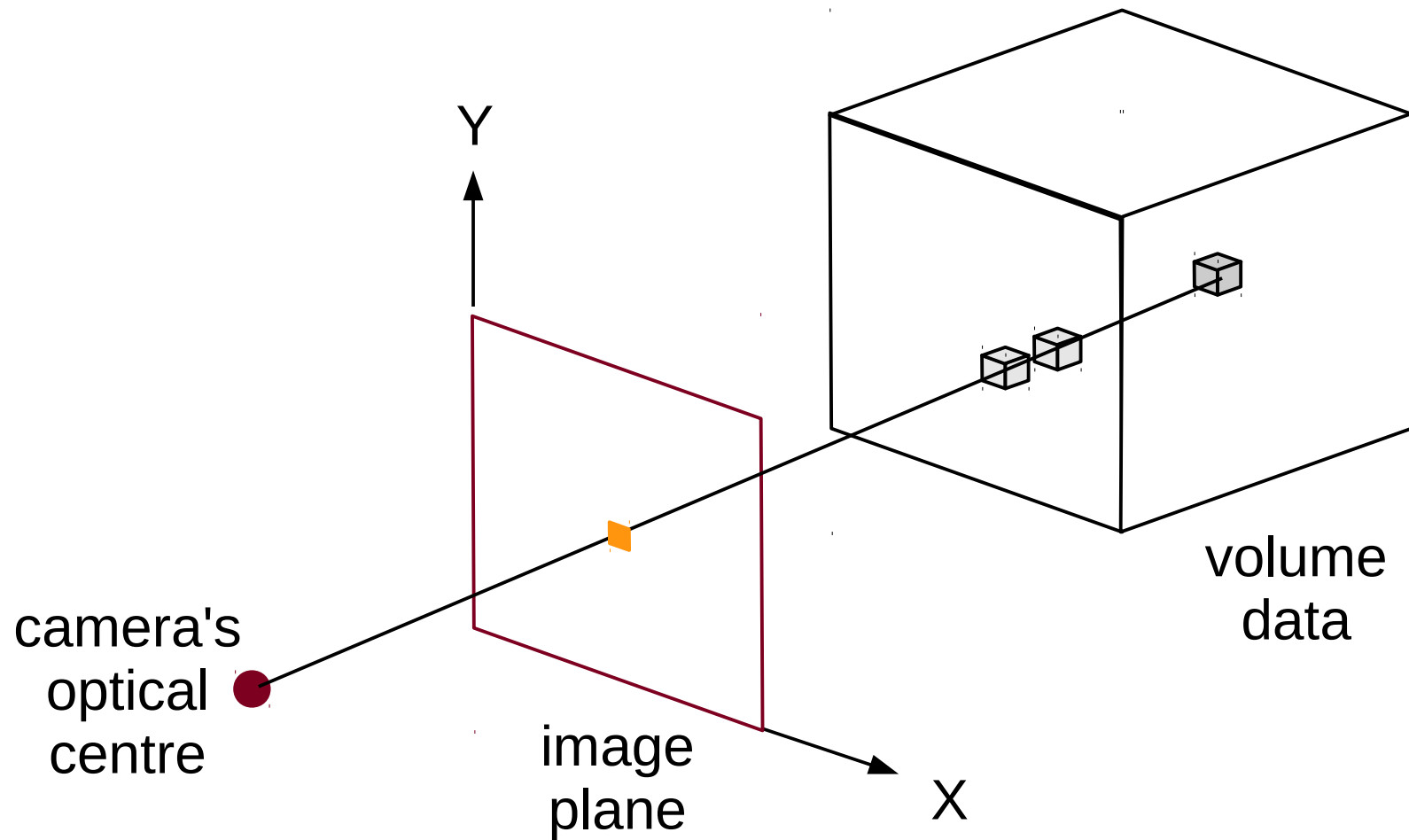


Leow Wee Kheng
CS3249 User Interface Development

Qt with VTK (Part 2)

Volume Rendering

- ⦿ A technique for rendering volume data.



⦿ Volume Rendering

- Map volume data to opacity values and colours.
- Cast ray from optical centre to volume data.
- Ray accumulates opacity values until totally opaque.
- Ray accumulates colours by blending with opacity values.
- Paint accumulated colour in image.

- ⊙ VTK has supports for volume rendering.
- ⊙ Main VTK components for volume rendering
 - reader: `vtkDICOMImageReader`
 - mapper: `vtkFixedPointVolumeRayCastMapper`
 - actor: `vtkVolume`
 - property: `vtkProperty`
 - opacity function: `vtkPiecewiseFunction`
 - colour function: `vtkColorTransferFunction`
 - renderer: `vtkRenderer`
 - render window: `vtkRenderWindow`
 - interactor: `vtkRenderWindowInteractor`
 - interactor style: `vtkInteractorStyleTrackballCamera`

```
VolumeRenderer::VolumeRenderer()
{
    // Initialisation
    reader = NULL;
    mapper = NULL;

    // Create GUI
    createWidgets();
    createActions();
    createMenus();
    createToolBars();
    createStatusBar();
    initSize();
}
```

```
void VolumeRenderer::createWidgets()
{
    // Create vtk objects
    vtkWidget = new QVTKWidget(this);

    // Create transfer function
    opacityFn = vtkPiecewiseFunction::New();
    colorFn = vtkColorTransferFunction::New();

    // Create volume property and attach transfer functions
    property = vtkVolumeProperty::New();
    property->SetIndependentComponents(true);
    property->SetScalarOpacity(opacityFn);
    property->SetColor(colorFn);
    property->SetInterpolationTypeToLinear();

    // Create actor
    actor = vtkVolume::New();
    actor->SetProperty(property);
}
```

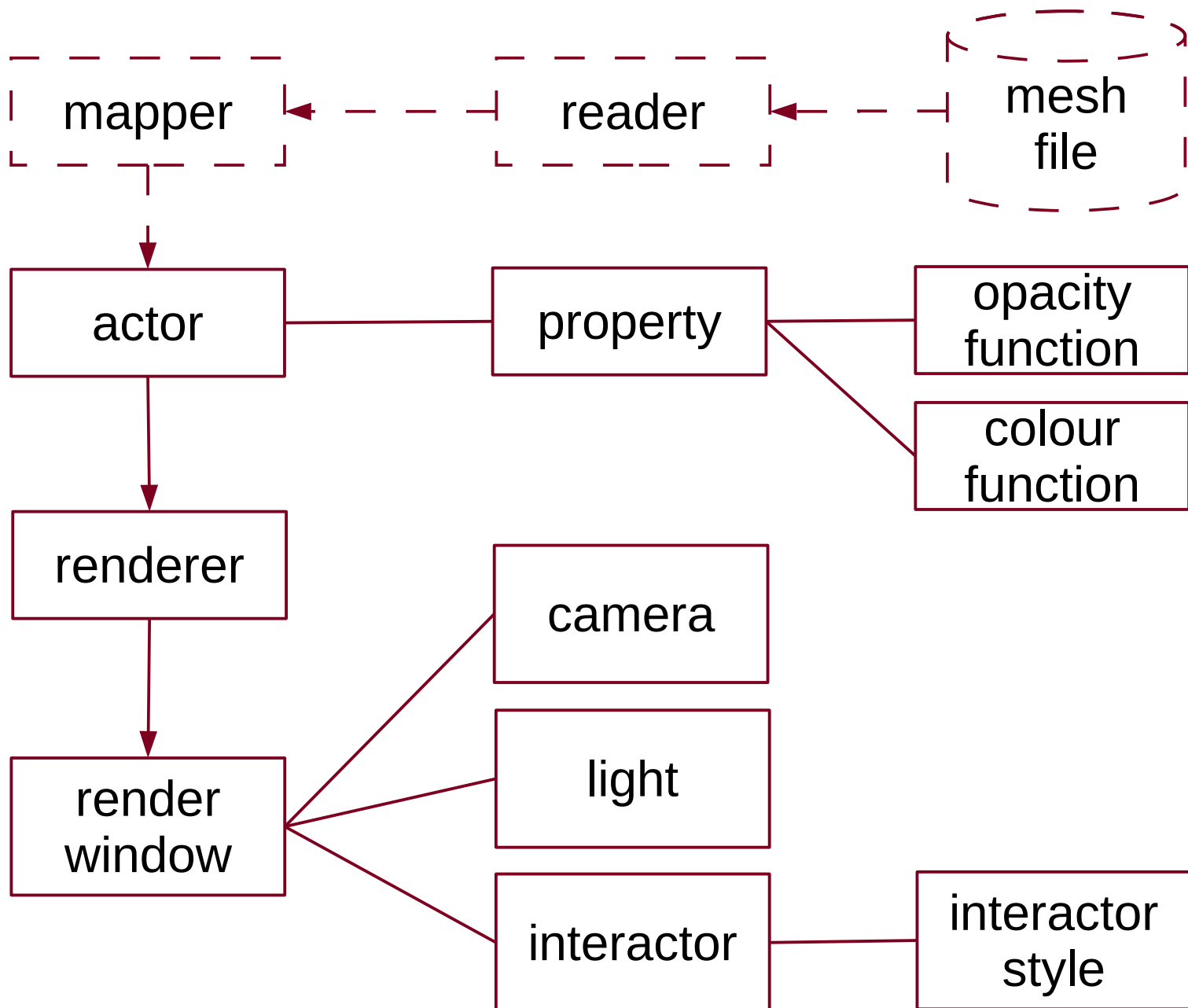
```
// Create renderer
renderer = vtkRenderer::New();
renderer->AddActor(actor);
renderer->SetBackground(0.0, 0.0, 0.0);

// QVTKWidget has render window and interactor
renderWindow = vtkWidget->GetRenderWindow();
renderWindow->AddRenderer(renderer);
interactor = renderWindow->GetInteractor();

style = vtkInteractorStyleTrackballCamera::New();
interactor->SetInteractorStyle(style);

// Central widget
setCentralWidget(vtkWidget);
```

⦿ After creating widgets...



⦿ Mapper is not yet created

- `vtkFixedPointVolumeRayCastMapper` without reader causes run-time warning message.

```

// Create combo boxes
viewTypeBox = new QComboBox(this);
QStringList choices;
choices << "MIP" << "CT All" << "CT Skin"
        << "CT Muscle" << "CT Organ" << "CT Bone";
viewTypeBox->insertItems(0, choices);
connect(viewTypeBox, SIGNAL(activated(int)),
        this, SLOT(setViewType(int)));

sampleDistanceBox = new QComboBox(this);
choices.clear();
choices << "1" << "1/2" << "1/4" << "1/8" << "1/16";
sampleDistanceBox->insertItems(0, choices);
connect(sampleDistanceBox, SIGNAL(activated(int)),
        this, SLOT(setSampleDistance(int)));

// Overall
setWindowTitle("Volume Renderer");
setWindowIcon(QIcon(":/images/viewer.png"));
}

```

- ◉ Combo boxes are added into tool bar.

```
void VolumeRenderer::createToolBars()
{
    fileToolBar = addToolBar(tr("&File"));
    fileToolBar->addAction(openAction);
    fileToolBar->addAction(saveAction);

    viewToolBar = addToolBar(tr("&View"));
    viewToolBar->addAction(resetViewAction);
    viewToolBar->addWidget(viewTypeBox);
    viewToolBar->addWidget(new QLabel("  Step  "));
    viewToolBar->addWidget(sampleDistanceBox);

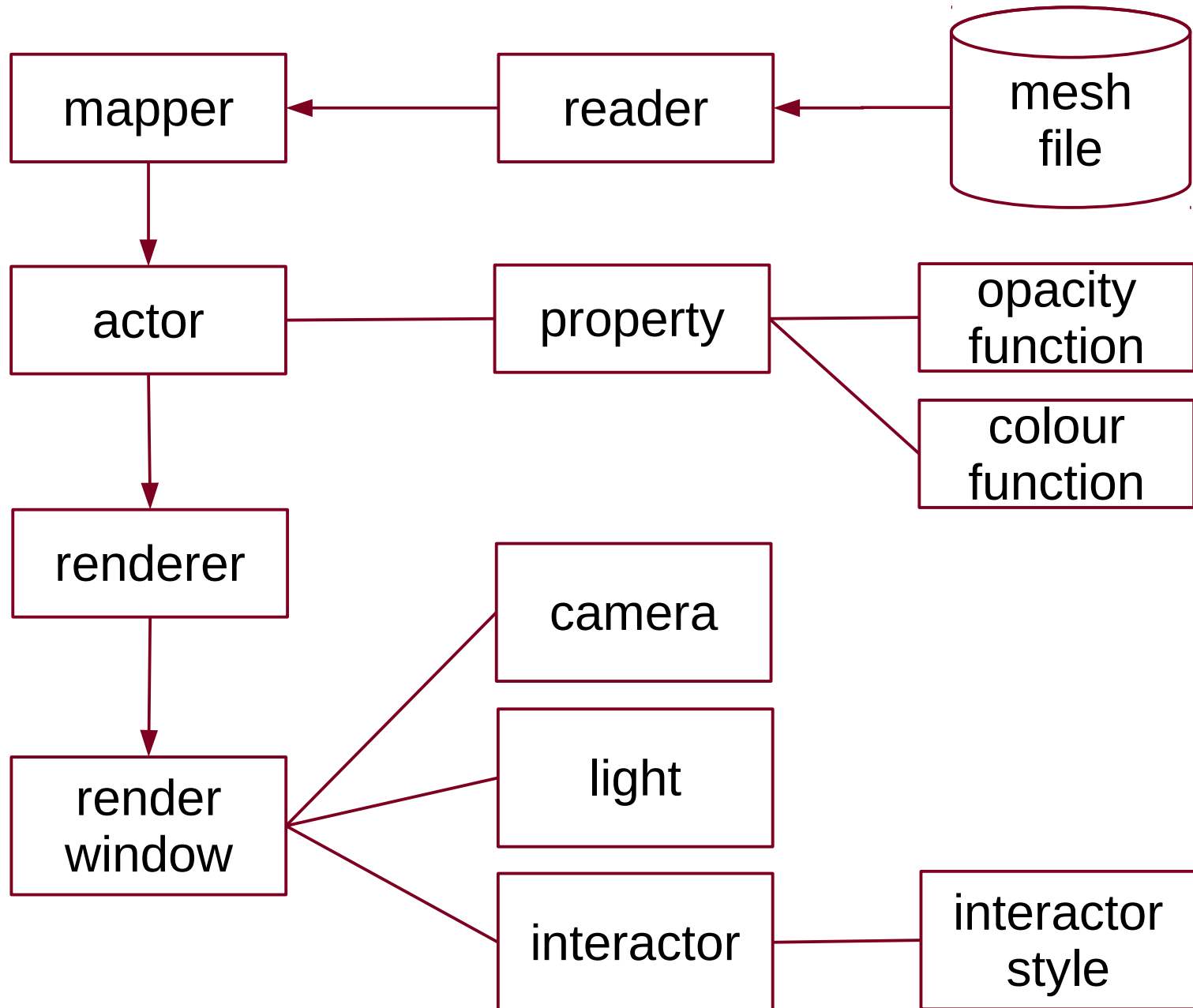
    helpToolBar = addToolBar(tr("&Help"));
    helpToolBar->addAction(helpAction);
    helpToolBar->addAction(aboutAction);
}
```

```
void VolumeRenderer::loadVolume(const QString &dirName)
{
    if (reader)
    {
        initCamera();
        sampleDistanceBox->setCurrentIndex(0);
        reader->Delete();
        reader = NULL;
    }

    reader = vtkDICOMImageReader::New();
    reader->SetDirectoryName(dirName.toAscii().data());
}
```

```
if (!mapper)
{
    mapper = vtkFixedPointVolumeRayCastMapper::New();
    actor->SetMapper(mapper);
    setBlendType(0);
}
mapper->SetInputConnection(reader->GetOutputPort());
renderer->ResetCamera();
getCameraParameters();
setWindowTitle("Volume Renderer - " + dirName);
}
```

⦿ After loading volume data...



Transfer Functions

- ⦿ Maps data value to output value.
- ⦿ Opacity transfer function
 - Maps voxel intensity to opacity value.
 - Selectively turn some voxels transparent to show interior.
- ⦿ Colour transfer function
 - Maps voxel intensity to colour value.
 - Selectively paint voxels with different colours.

⊙ VTK's general transfer function

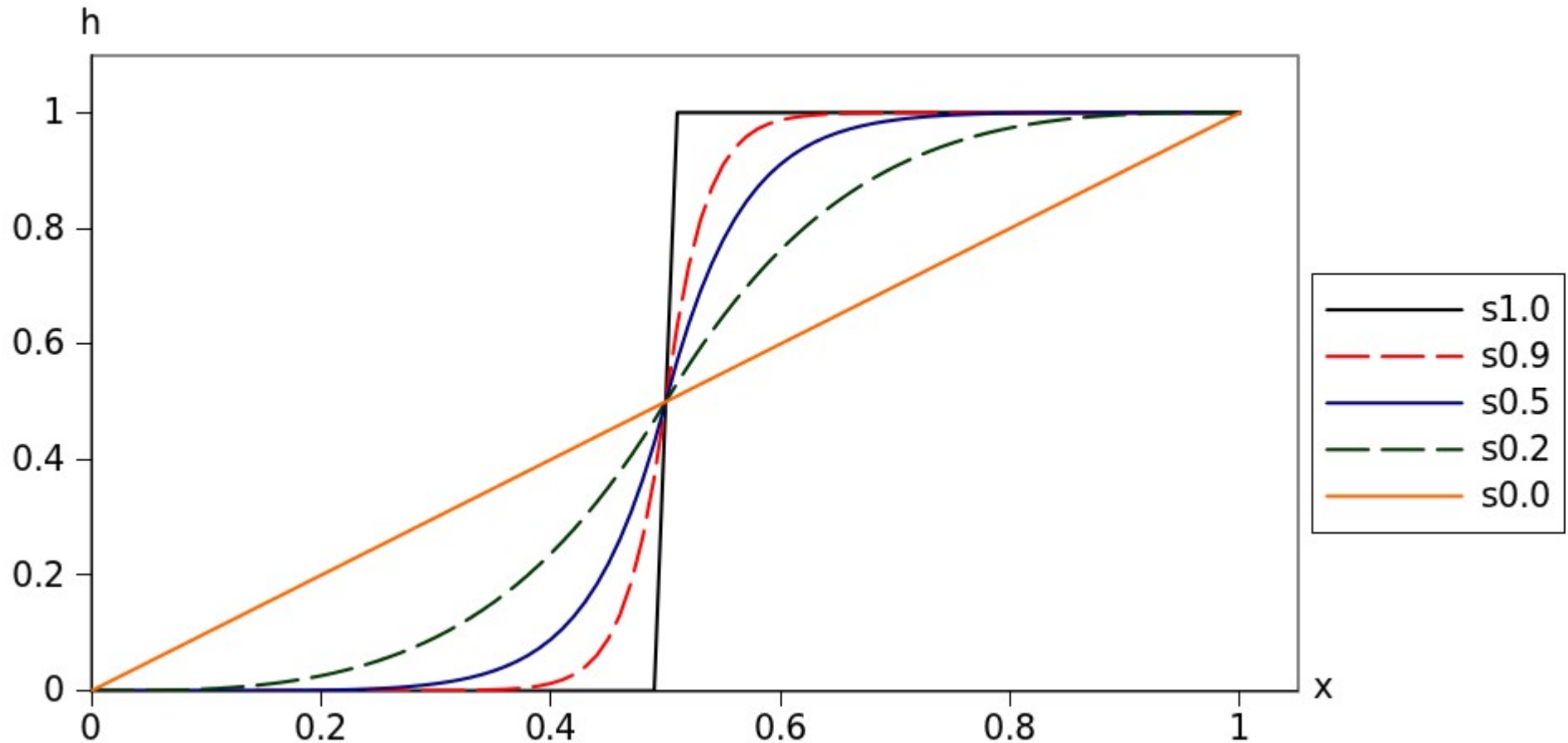
- `vtkPiecewiseFunction`
- Implemented as piecewise **Hermite** function.

```
vtkPiecewiseFunction *fn =  
    vtkPiecewiseFunction::New();  
fn->AddPoint(x1, y1, m1, s1);  
fn->AddPoint(x2, y2, m2, s2);
```

- $m1$: mid-point, x value of $(y1 + y2) / 2$
- $s1$: sharpness from $(x1, y1)$ to $(x2, y2)$.
 - 0: linear
 - $0 < s1 < 1$: curve
 - 1: constant

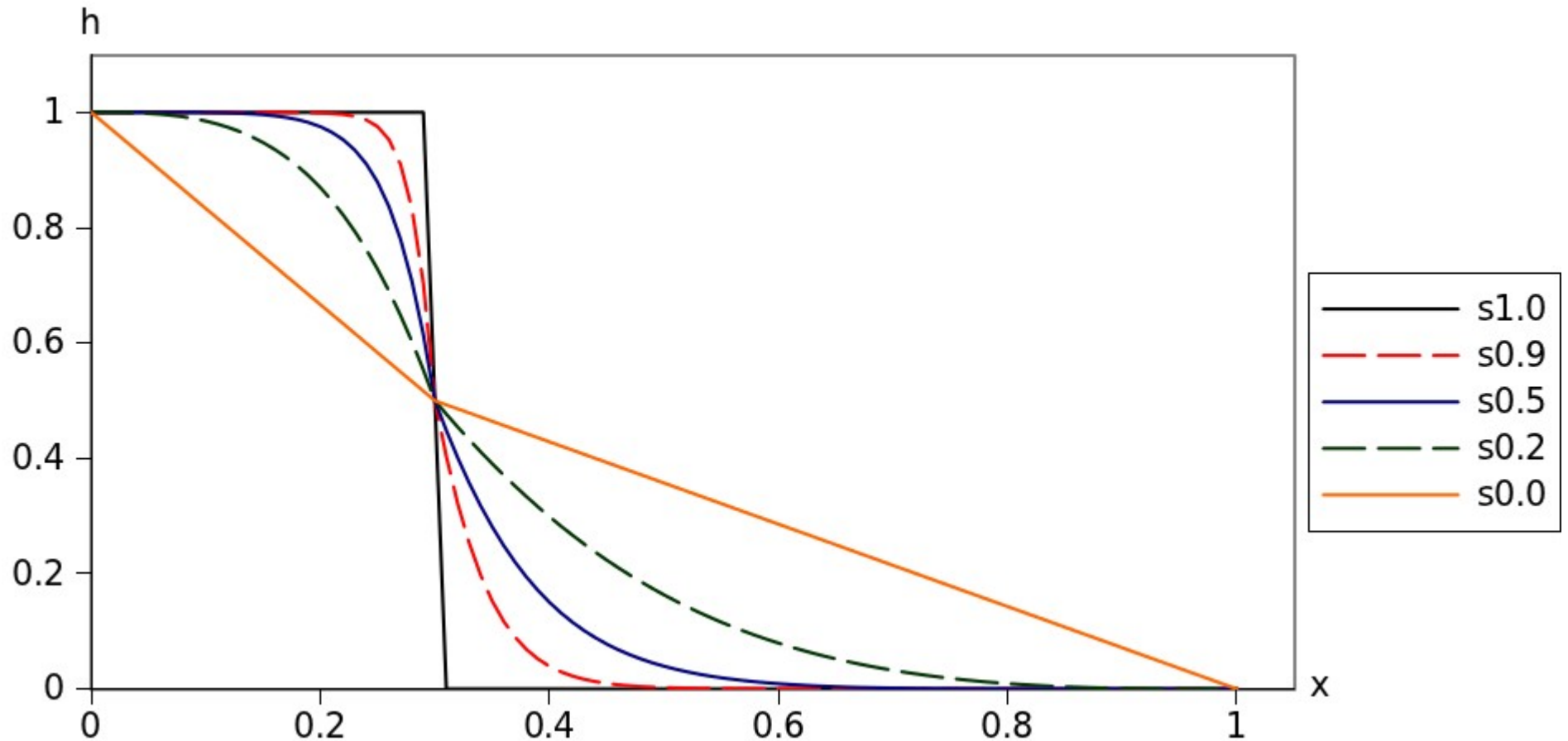
⊙ Plot of Hermite functions for various sharpness s .

```
fn->AddPoint(0.0, 0.0, 0.5, s);  
fn->AddPoint(1.0, 1.0, m2, s2);
```



⊙ Plot of Hermite functions for various sharpness s .

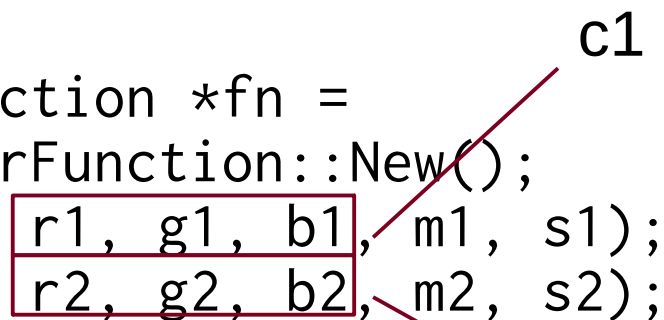
```
fn->AddPoint(0.0, 1.0, 0.3, s);  
fn->AddPoint(1.0, 0.0, m2, s2);
```



⊙ VTK's colour transfer function:

- `vtkColorTransferFunction`
- Also implemented as piecewise **Hermite** function.
- Maps input value to RGB or HSV colour value.

```
vtkColorTransferFunction *fn =  
    vtkColorTransferFunction::New();  
fn->AddRGBPoint(x1, r1, g1, b1, m1, s1);  
fn->AddRGBPoint(x2, r2, g2, b2, m2, s2);
```

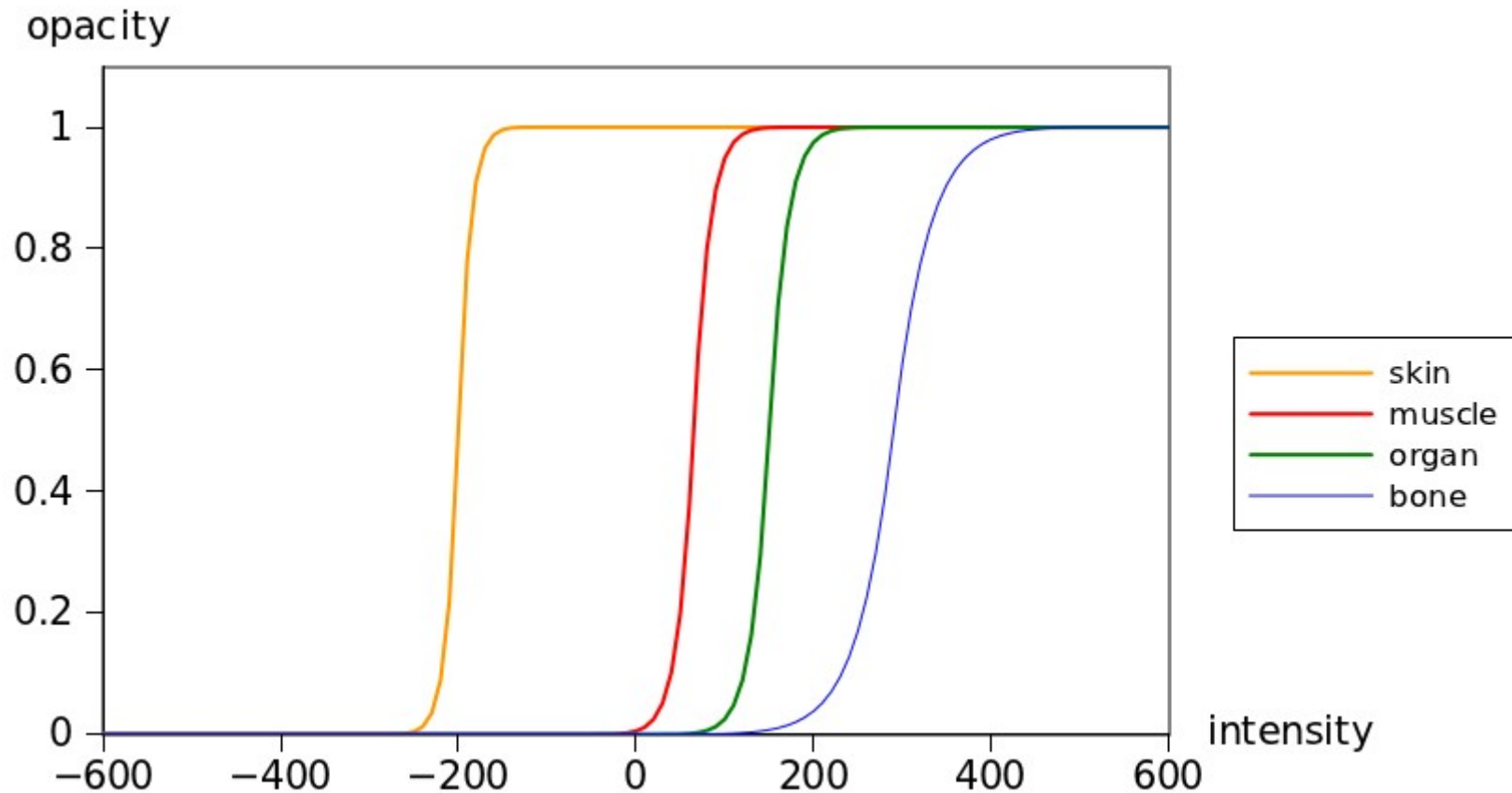


- `m1`: mid-point, x value of $(c1 + c2) / 2$.
- `s1`: sharpness from $(x1, c1)$ to $(x2, c2)$.
 - 0: linear
 - $0 < s1 < 1$: curve
 - 1: constant

- ⊙ Body tissues have different intensities in CT image:
 - air: -1000
 - fluid, transparent tape, hair strands: -1000 to -300
 - skin: -300 to 100
 - fat: -150 to 0
 - muscle: -70 to 100
 - organ: 0 to 250
 - bone: 50 to 1000
- ⊙ Use appropriate transfer functions depending on what we want to show.

⊙ Opacity Functions

- skin: make fluid, hair transparent, and skin opaque.
- muscle: make skin transparent and muscle opaque.

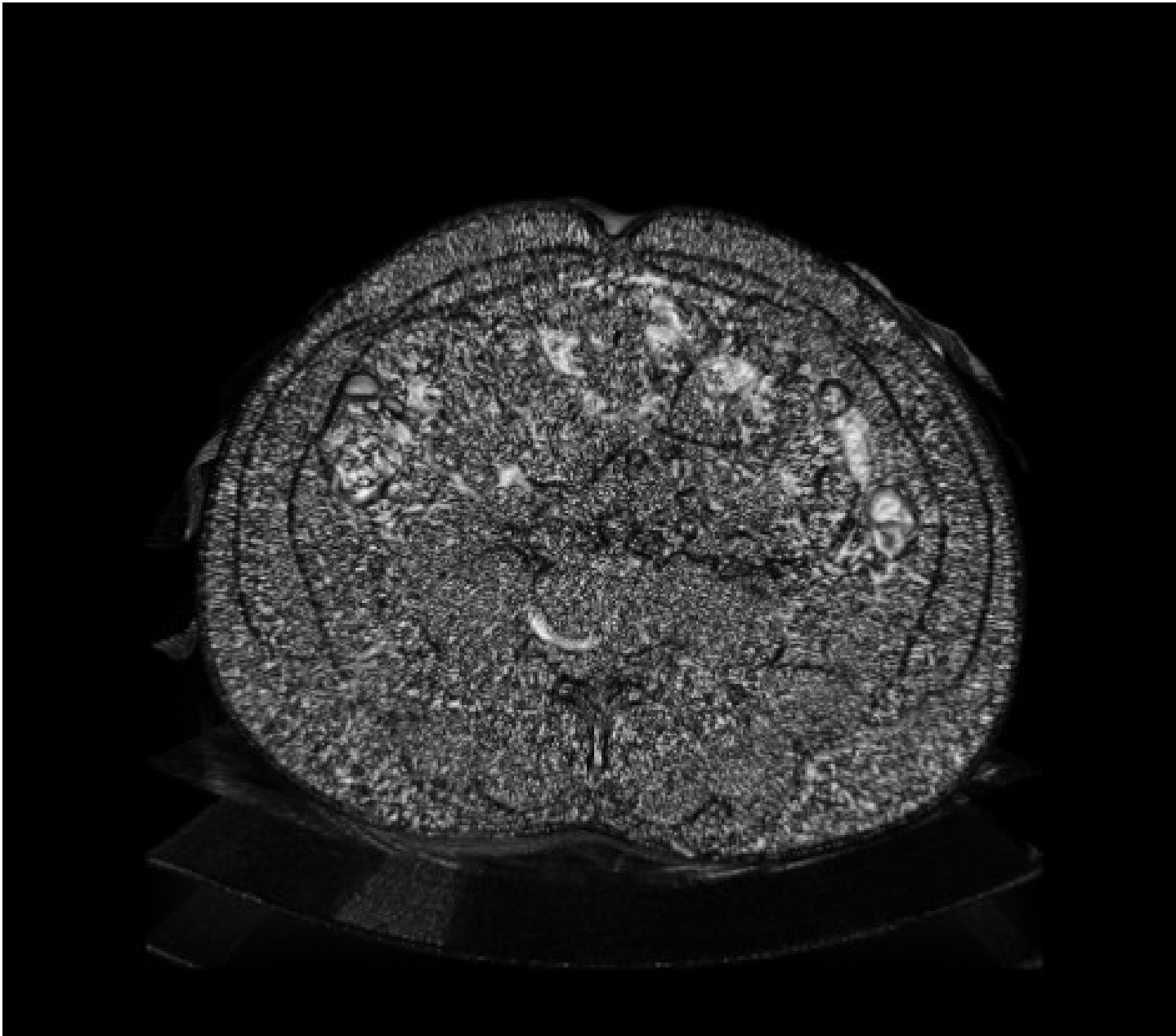


⦿ Example for skin:

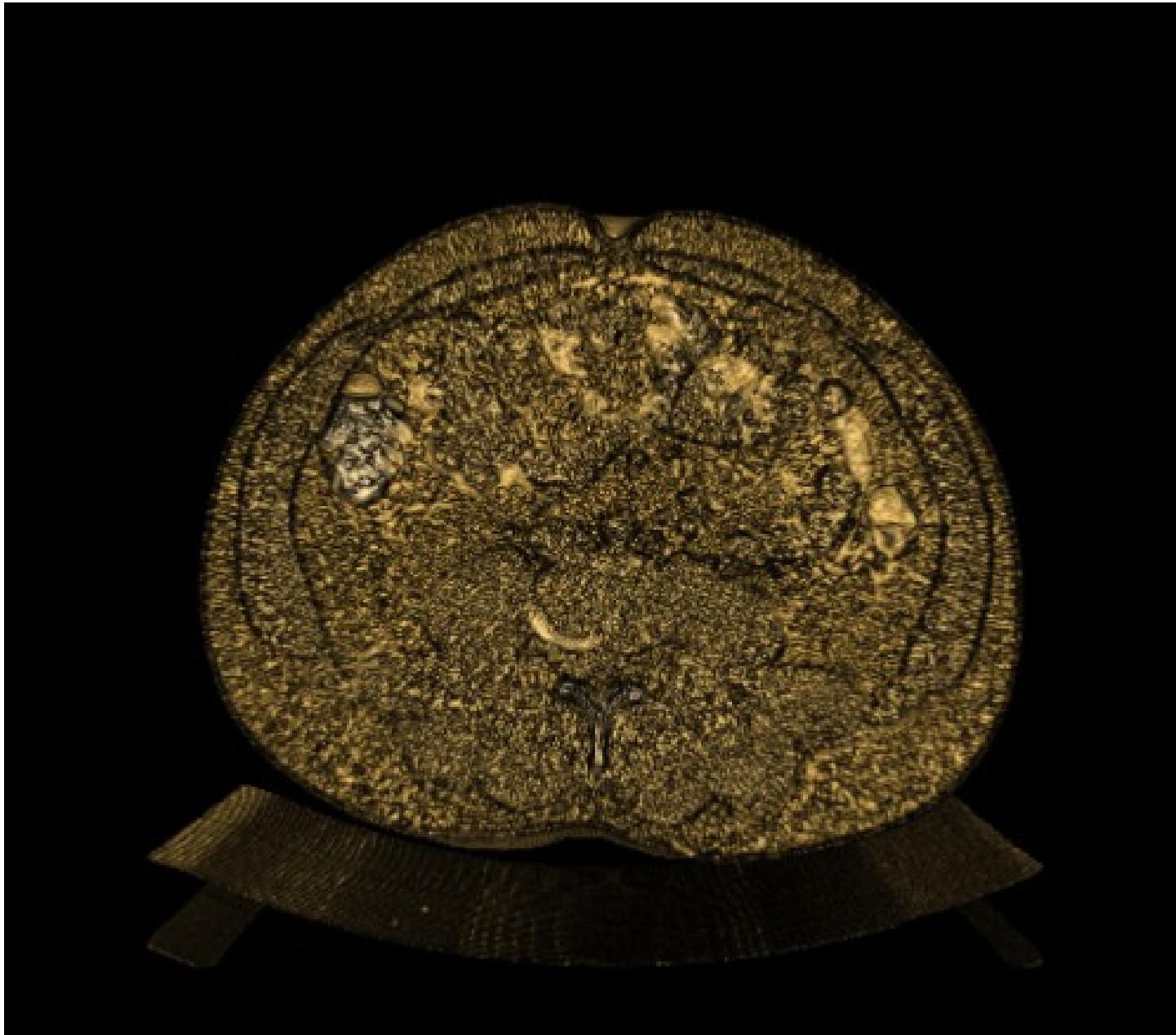
```
opacityFn->AddPoint(-2000, 0.0, 0.5, 0.0);  
opacityFn->AddPoint( -300, 0.0, 0.5, 0.5);  
opacityFn->AddPoint( -100, 1.0, 0.5, 0.0);  
opacityFn->AddPoint( 2000, 1.0, 0.5, 0.0);
```

```
colorFn->AddRGBPoint(-2000, 0.0, 0.0, 0.0, 0.5, 0.0);  
colorFn->AddRGBPoint( -300, 0.6, 0.4, 0.2, 0.5, 0.5);  
colorFn->AddRGBPoint( -100, 0.9, 0.7, 0.3, 0.5, 0.0);  
colorFn->AddRGBPoint( 2000, 1.0, 1.0, 1.0, 0.5, 0.0);
```

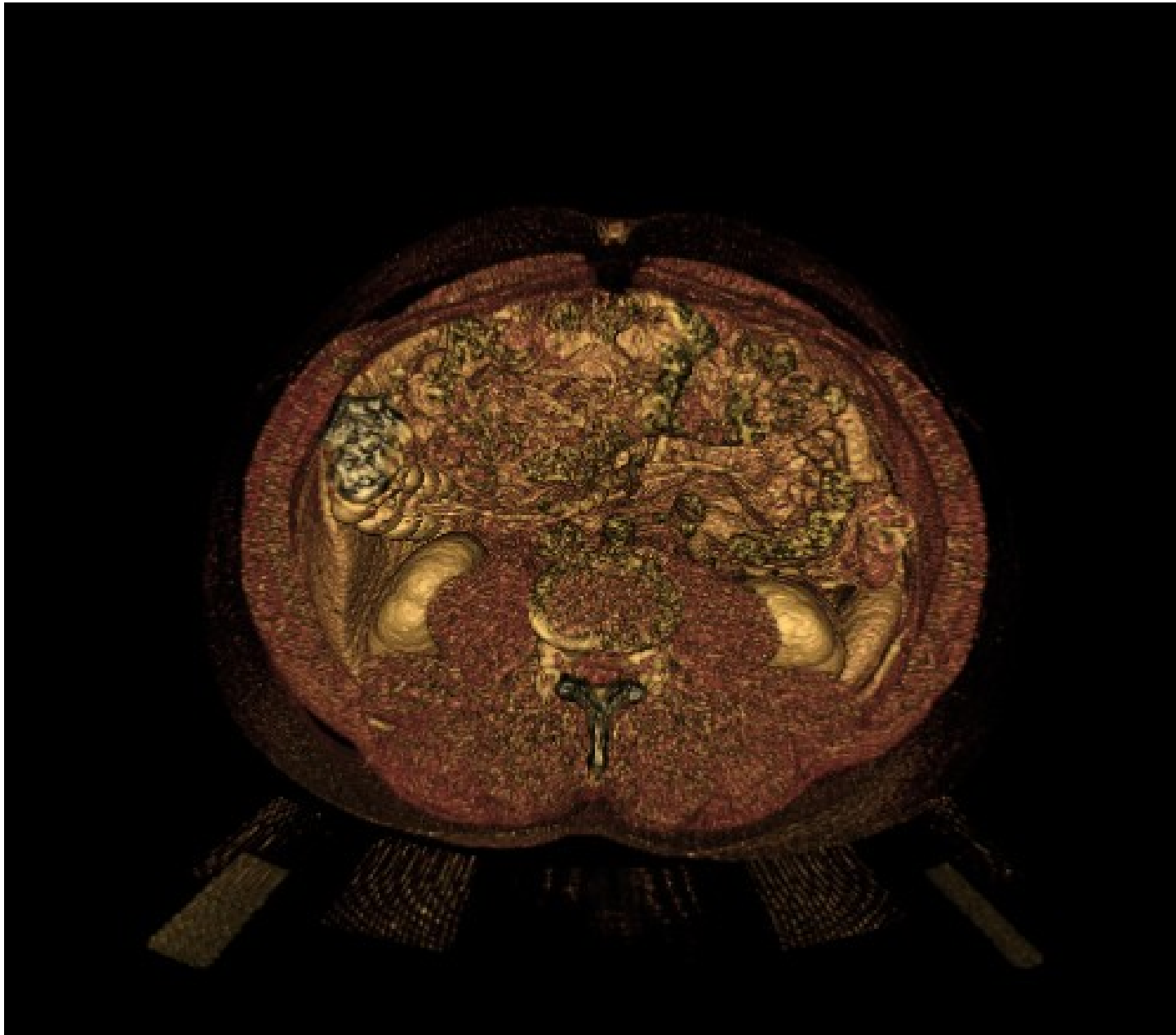
Volume rendering of everything



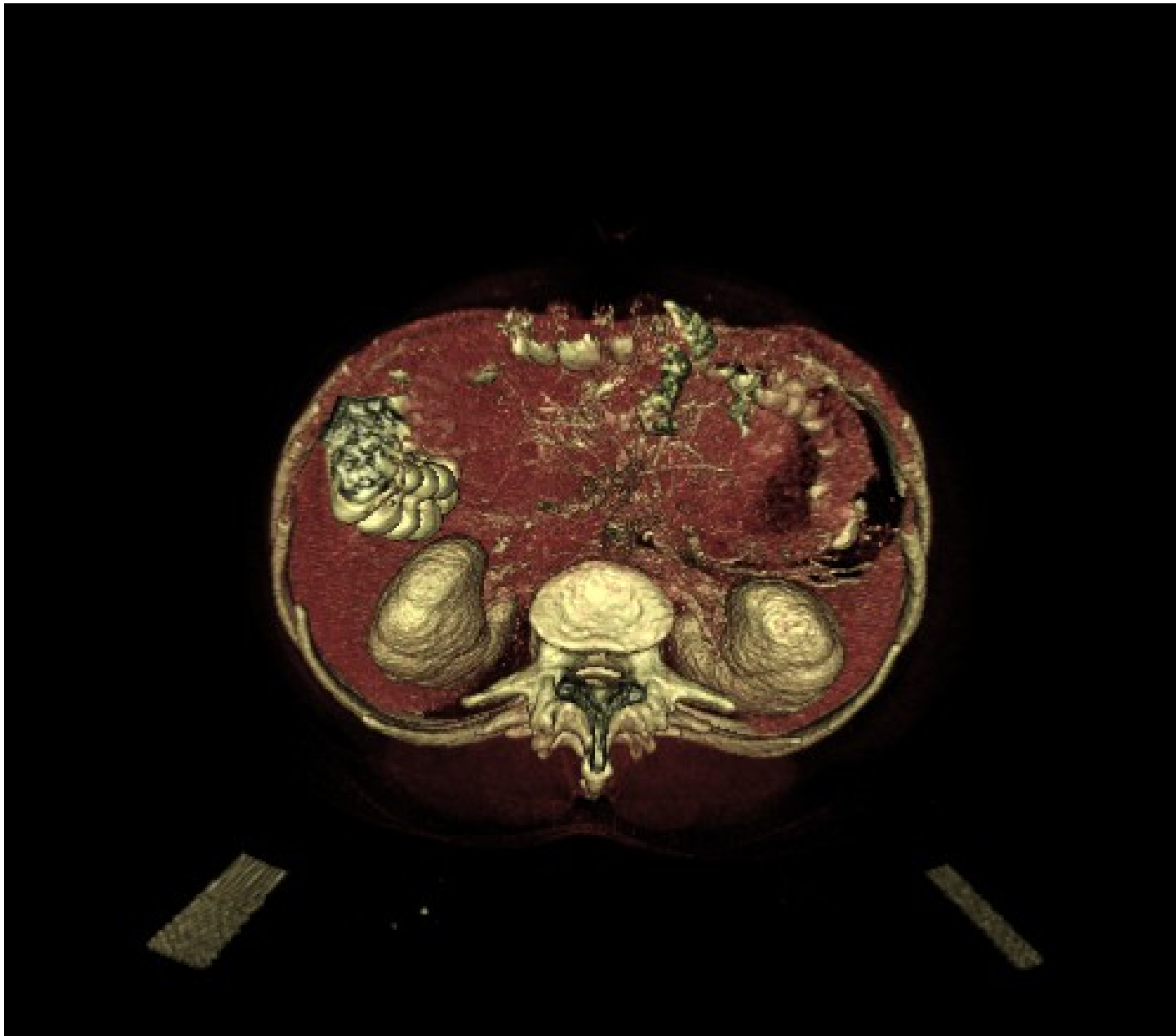
Volume rendering for skin



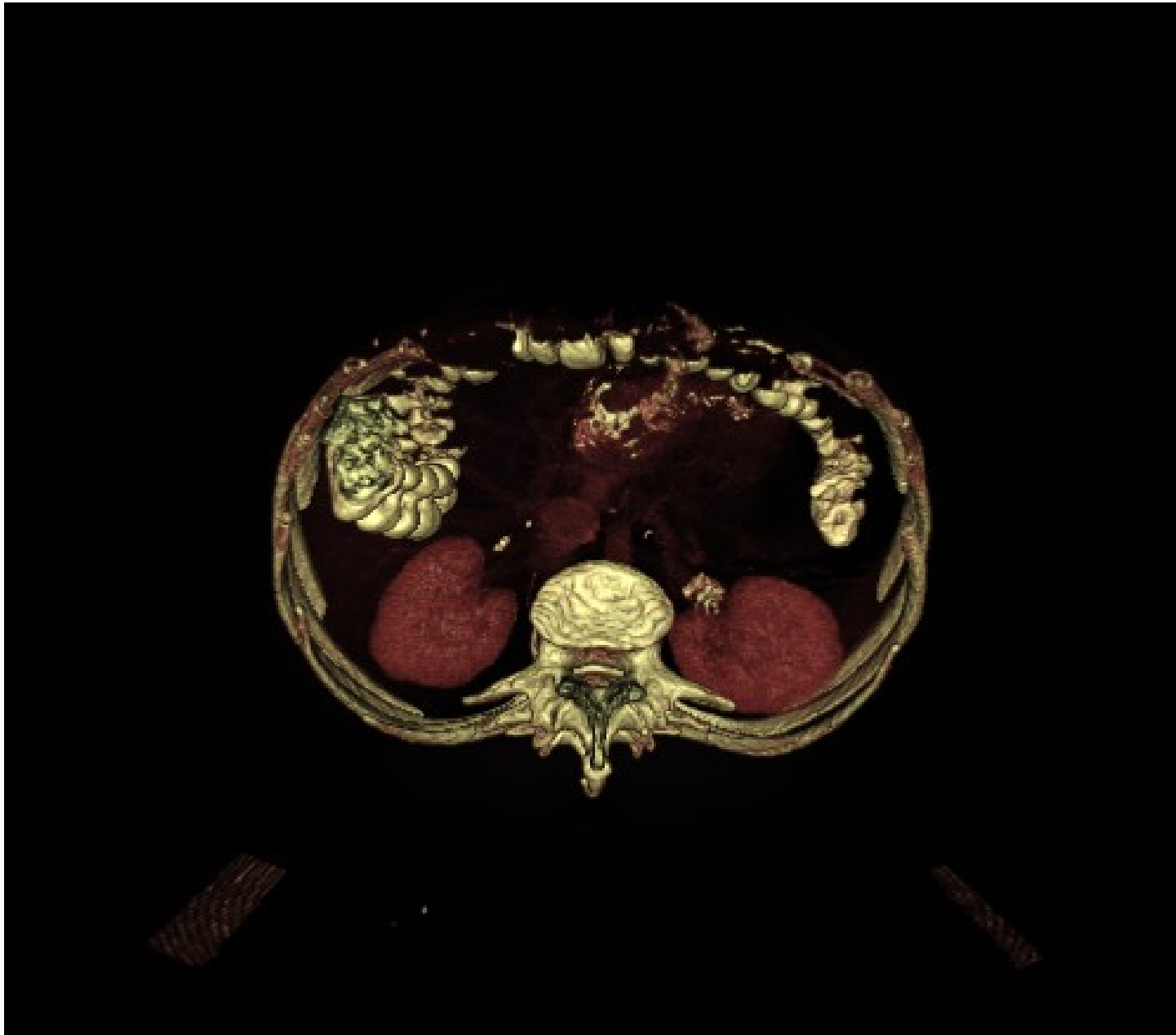
Volume rendering for muscle



Volume rendering for organ



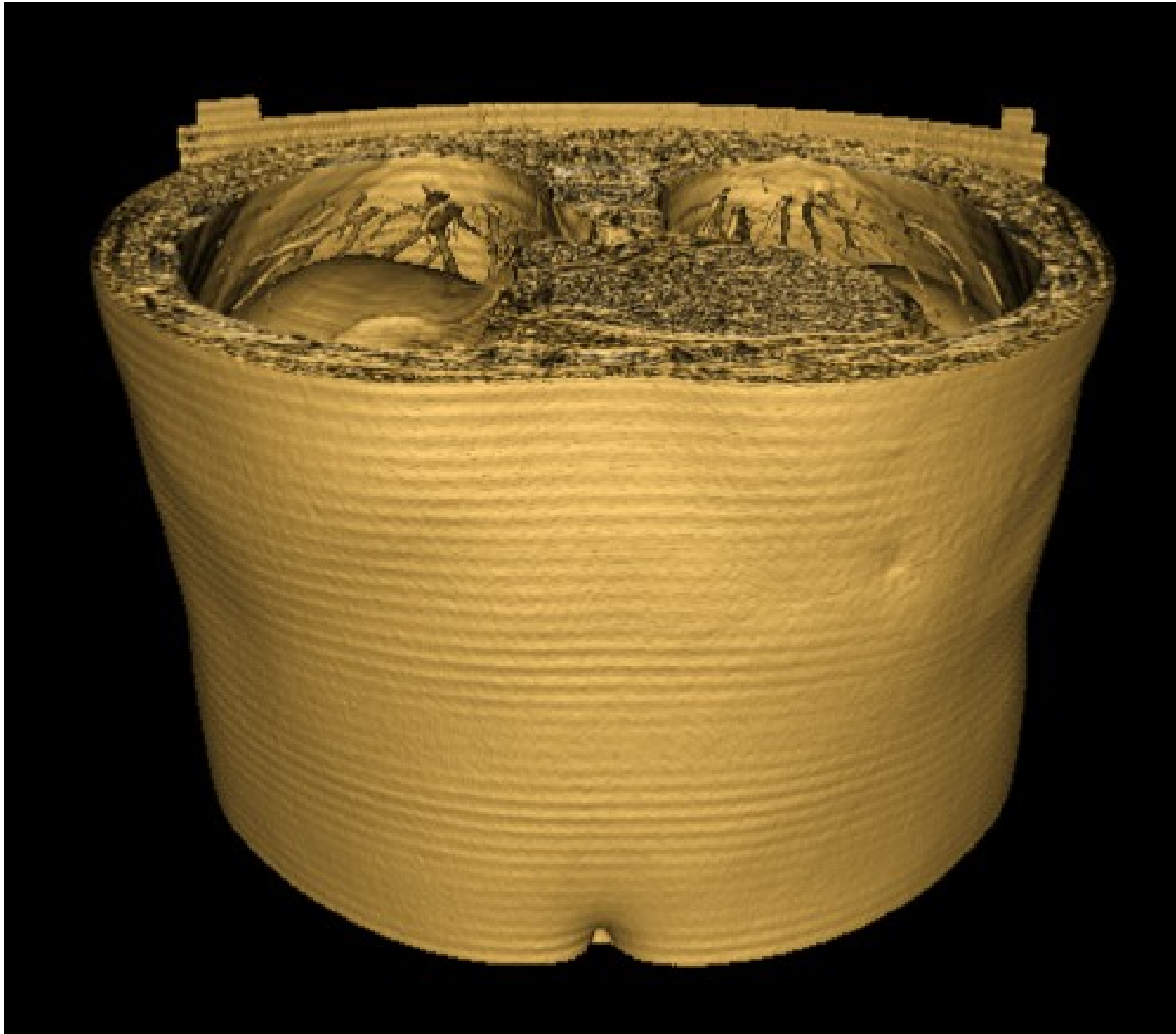
Volume rendering for bone



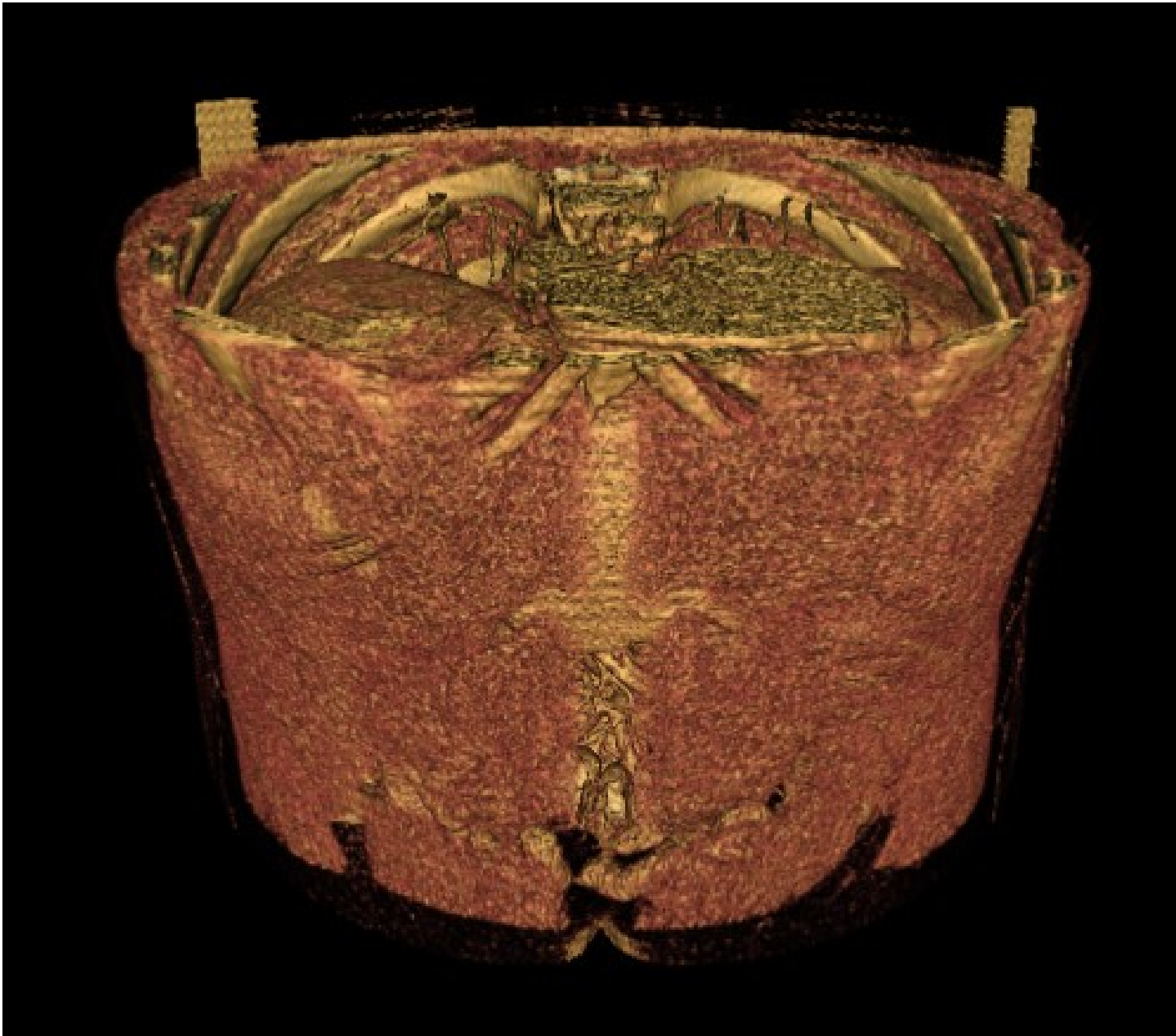
Volume rendering of everything



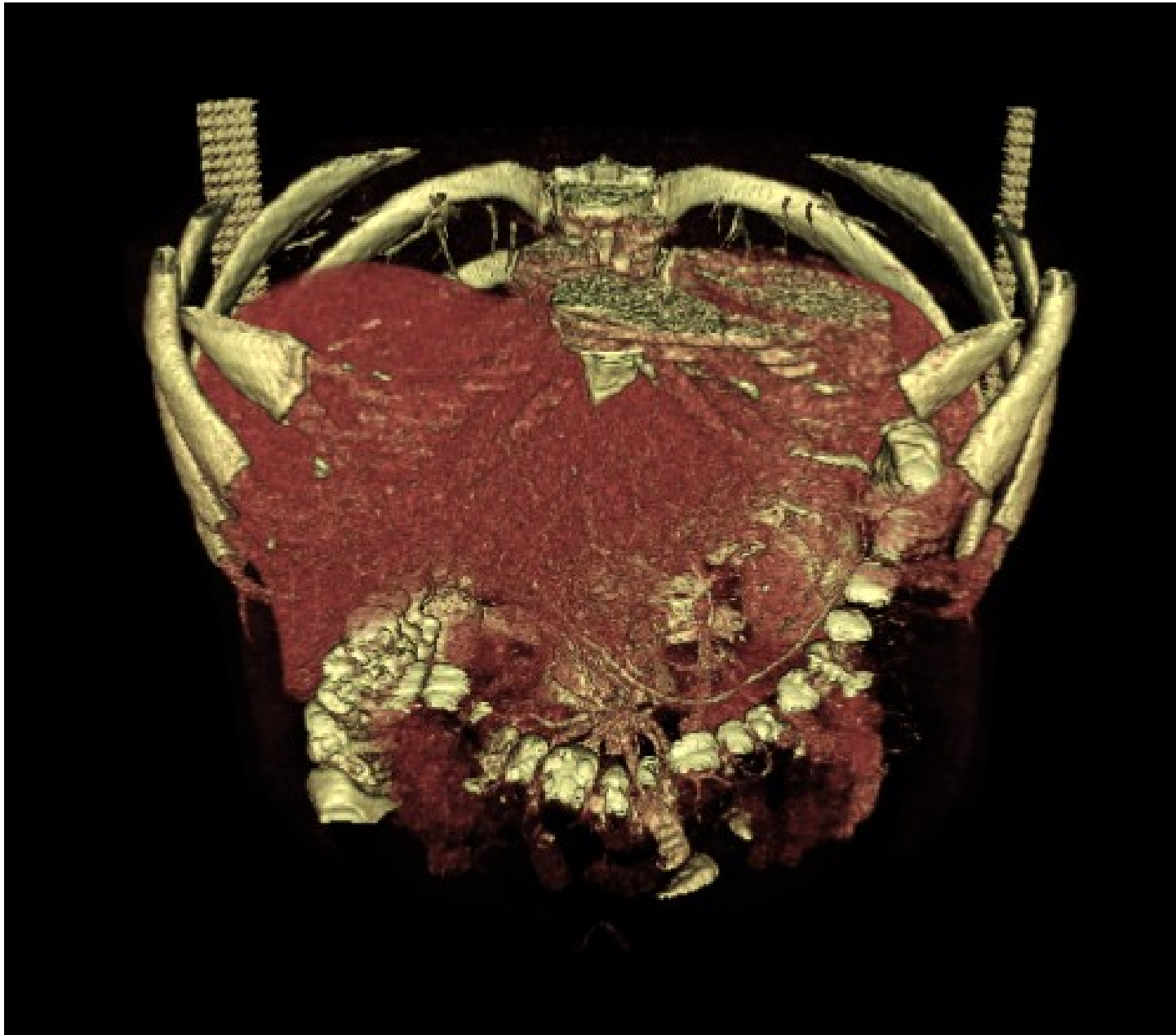
Volume rendering for skin



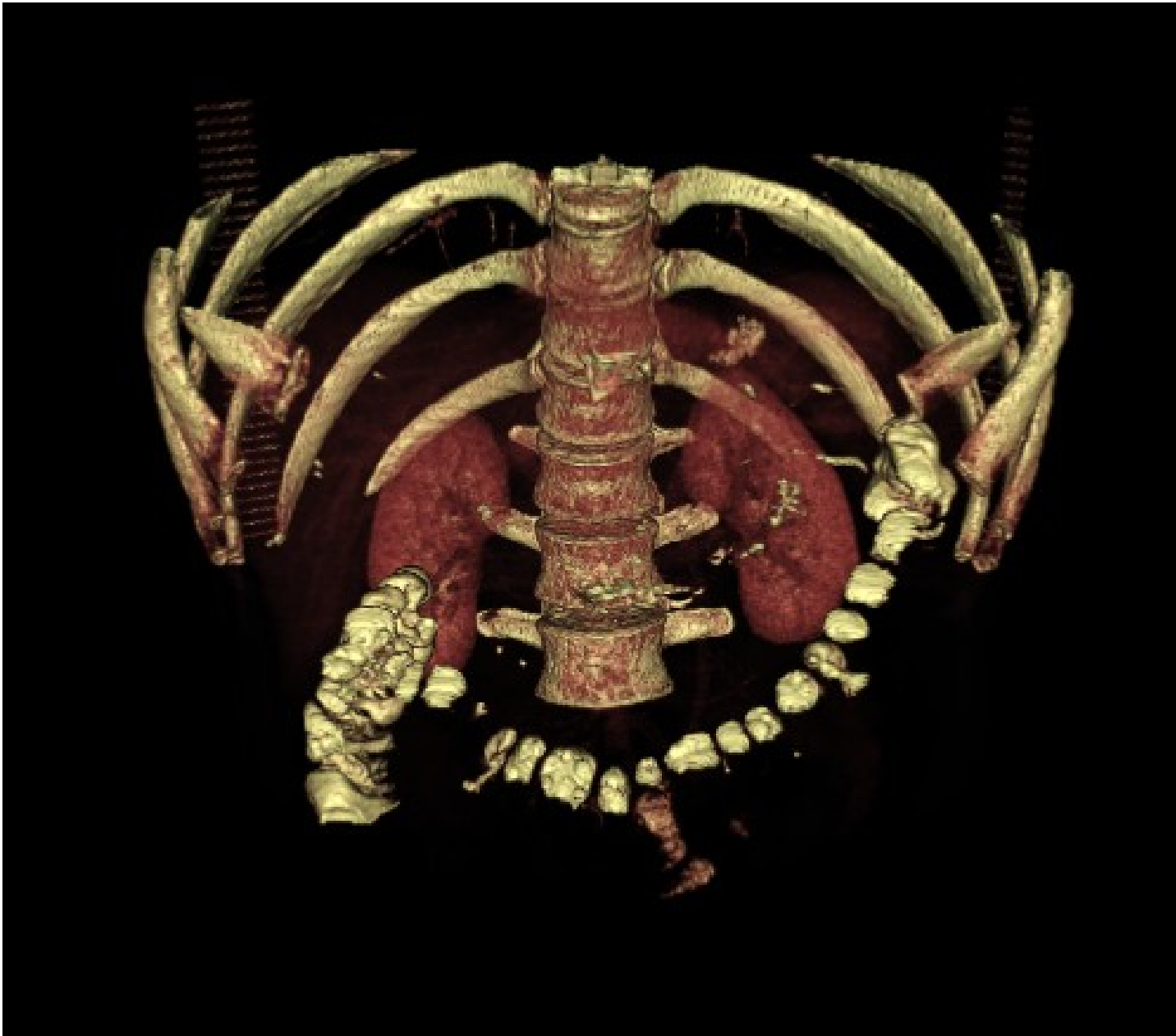
Volume rendering for muscle



Volume rendering for organ

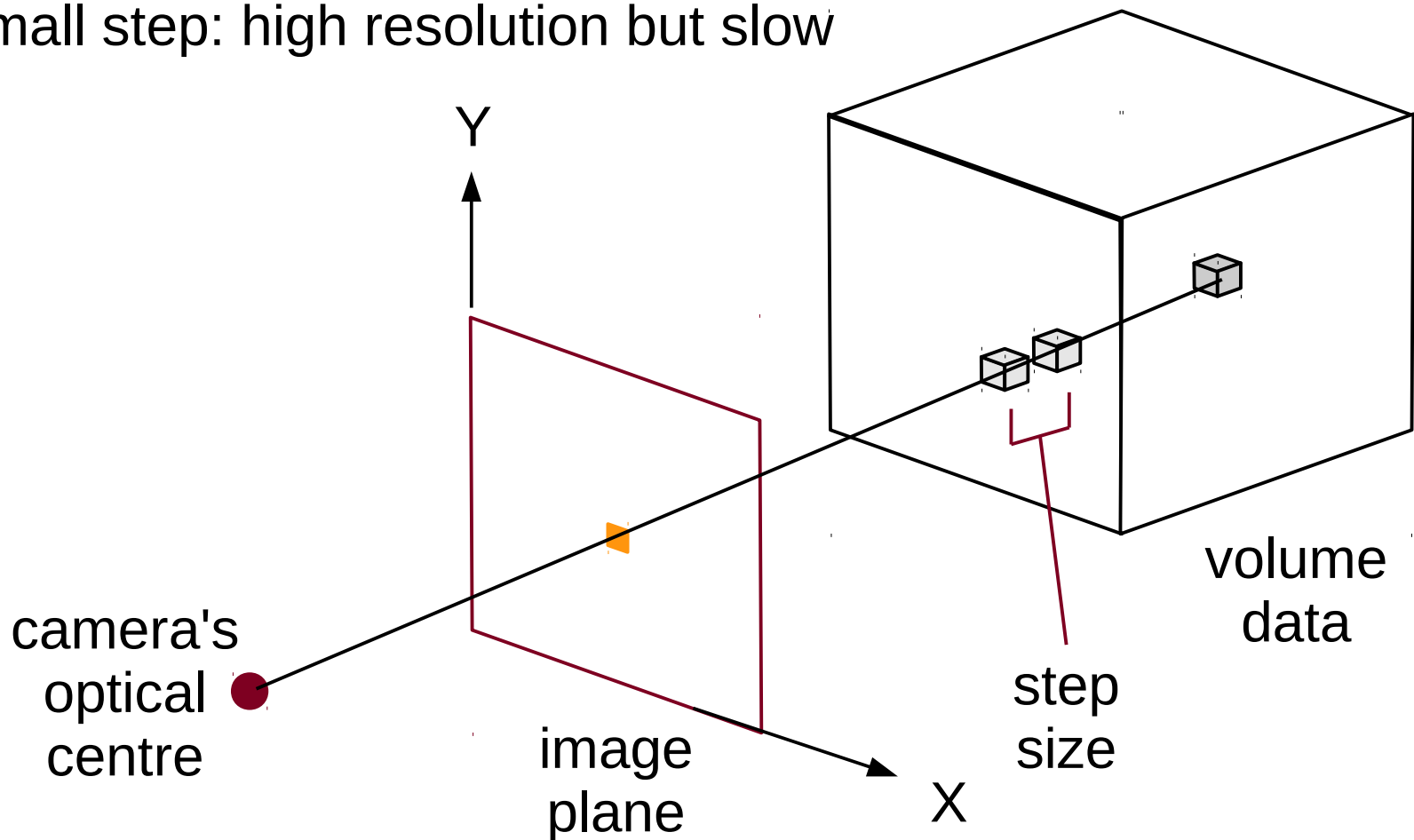


Volume rendering for bone



Rendering Resolution

- ⦿ Step size determines resolution
 - large step: fast but low resolution
 - small step: high resolution but slow

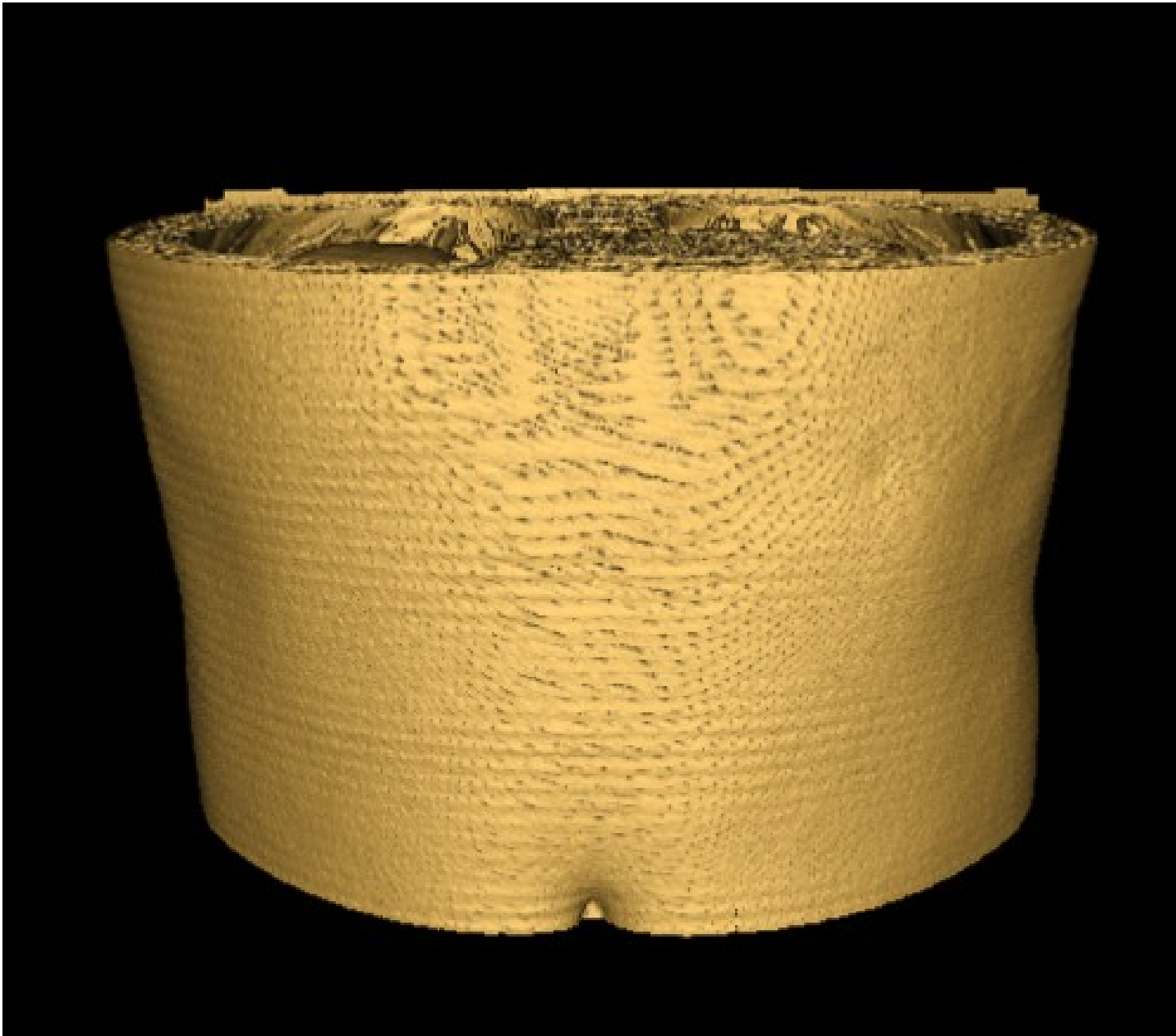


⦿ Set step size in mapper

```
void VolumeRenderer::setSampleDistance(int option)
{
    if (!reader)
        return;

    float distance = 1.0 / (1 << option);
    mapper->SetSampleDistance(distance);
    renderWindow->Render();
}
```

Step size = 1



Step size = $1/2$

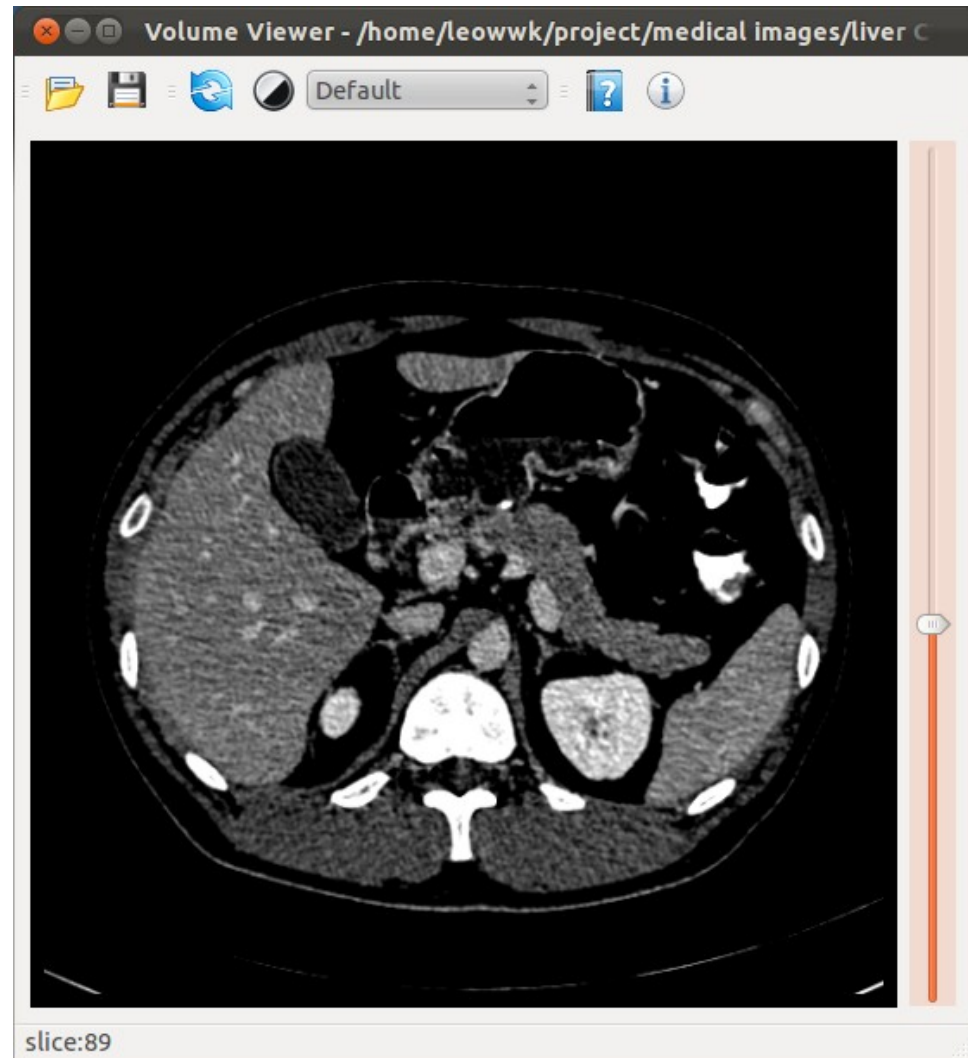


Step size = $1/4$



Volume Image Viewer

- ⦿ View volume image as a stack of 2D image slices.
- ⦿ Use QSlider to select the slice to view.



- ⦿ Use `vtkImageViewer2` to display image slice.
- ⦿ `vtkImageViewer2` packages the following together:
 - `vtkImageMapToWindowLevelColors`: mapper
 - `vtkImageActor`: actor
 - `vtkRenderer`: renderer
 - `vtkRenderWindow`: render window

```
void VolumeViewer::createWidgets()
{
    // Create vtk objects
    vtkWidget = new QVTKWidget(this);

    viewer = vtkImageViewer2::New();
    renderer = viewer->GetRenderer();
    renderWindow = viewer->GetRenderWindow();
    vtkWidget->SetRenderWindow(renderWindow);
    viewer->SetupInteractor(renderWindow->GetInteractor());
}
```

- ⦿ Creating `vtkImageViewer2` object without input causes VTK to produce run-time warning messages. But, can still work.


```
// Create Qt widgets
slider = new QSlider(Qt::Vertical);
slider->setRange(0, 1);
slider->setValue(0);
connect(slider, SIGNAL(valueChanged(int)), this,
        SLOT(setSlice(int)));
```

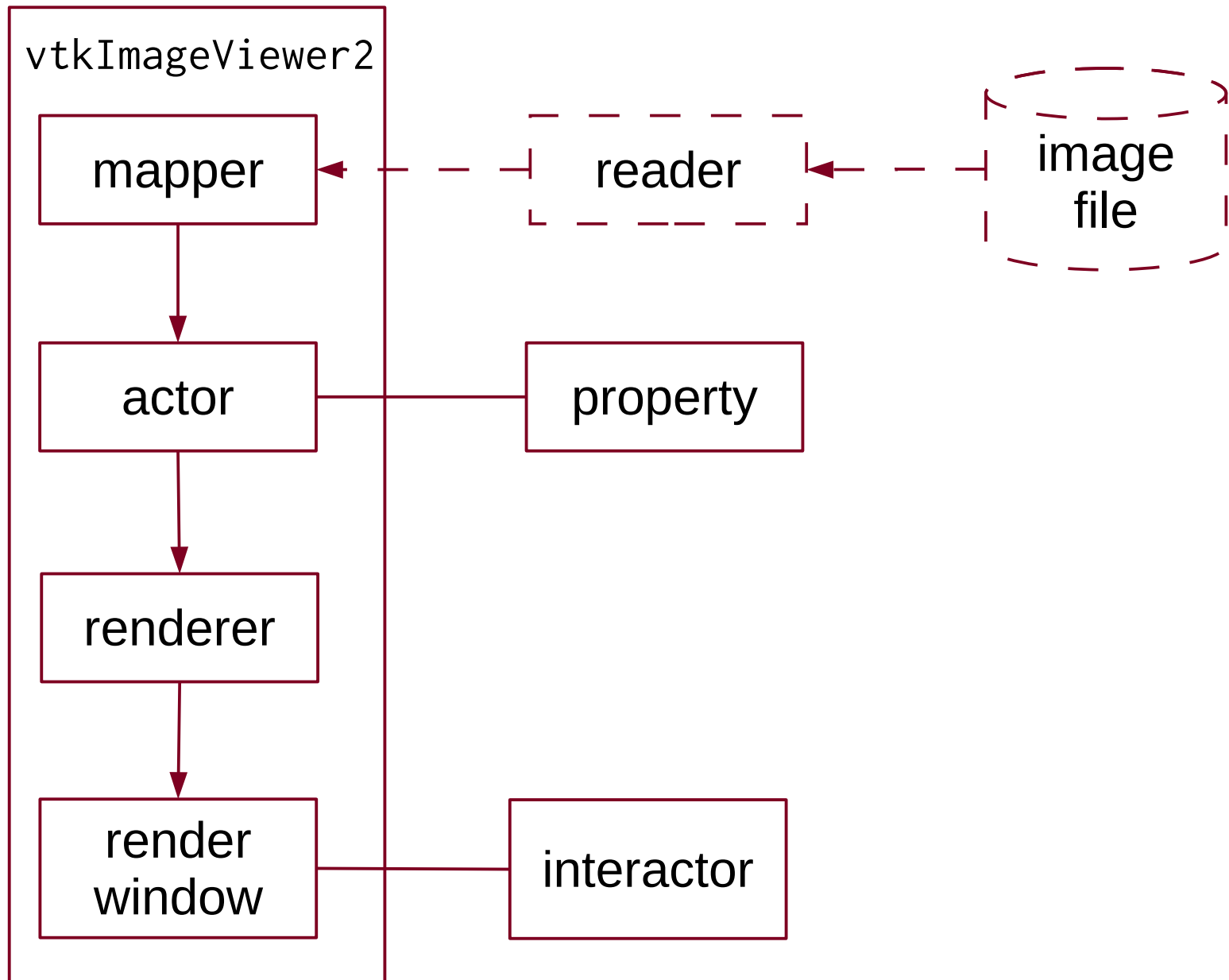
```
QHBoxLayout *hbox = new QHBoxLayout;
hbox->addWidget(vtkWidget);
hbox->addWidget(slider);
```

```
QWidget *main = new QWidget;
main->setLayout(hbox);
setCentralWidget(main);
```

```
// Create combo box
windowLevelBox = new QComboBox(this);
QStringList choices;
choices << "Default" << "CT Abdomen" << "CT Brain"
        << "CT Bone" << "CT Bone details" << "CT Head"
        << "CT Skin";
windowLevelBox->insertItems(0, choices);
connect(windowLevelBox, SIGNAL(activated(int)), this,
        SLOT(setWindowLevel(int)));

// Overall
setWindowTitle("Volume Viewer");
setWindowIcon(QIcon(":/images/viewer.png"));
}
```

⦿ After creating widgets...



```

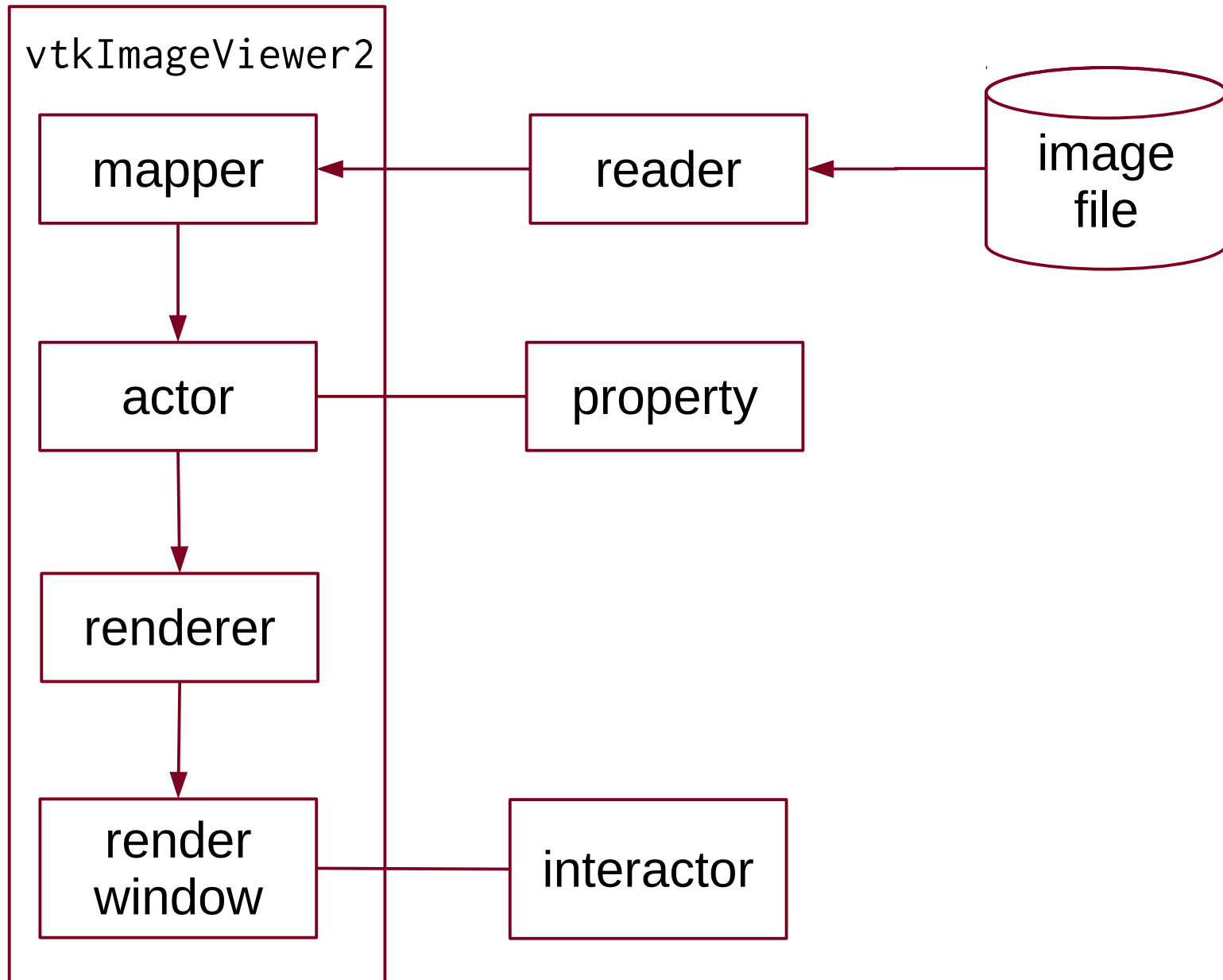
void VolumeViewer::loadVolume(const QString &dirName)
{
    if (reader) {
        initCamera();
        reader->Delete(); reader = NULL;
    }

    reader = vtkDICOMImageReader::New();
    reader->SetDirectoryName(dirName.toAscii().data());
    reader->Update();
    int *ip = reader->GetDataExtent();
    imageWidth = ip[1] - ip[0] + 1;
    imageHeight = ip[3] - ip[2] + 1;
    numSlices = ip[5] - ip[4] + 1;
    slider->setRange(0, numSlices - 1);

    viewer->SetInputConnection(reader->GetOutputPort());
    slider->setValue(0);
    renderer->ResetCamera();
    getCameraParameters();
    setWindowTitle("Volume Viewer - " + dirName);
}

```

⦿ After connecting reader to vtkImageViewer2...

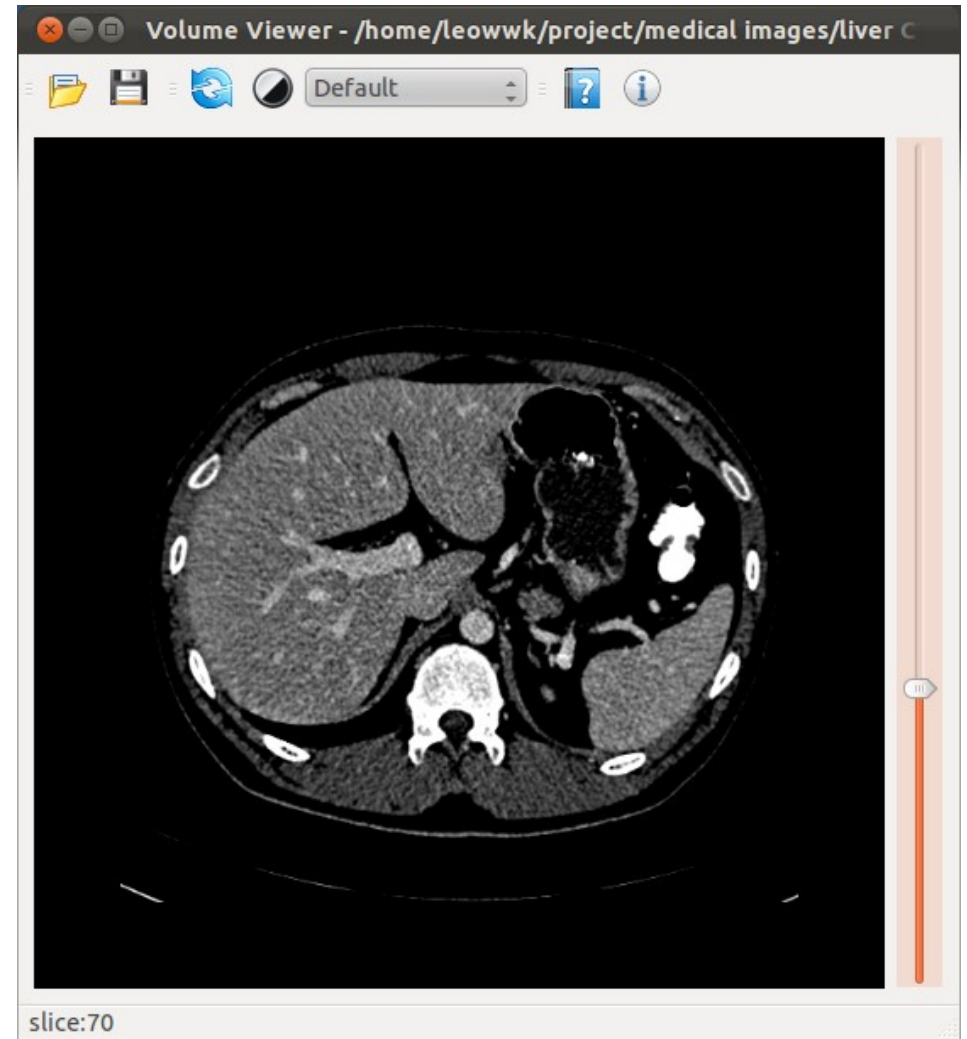
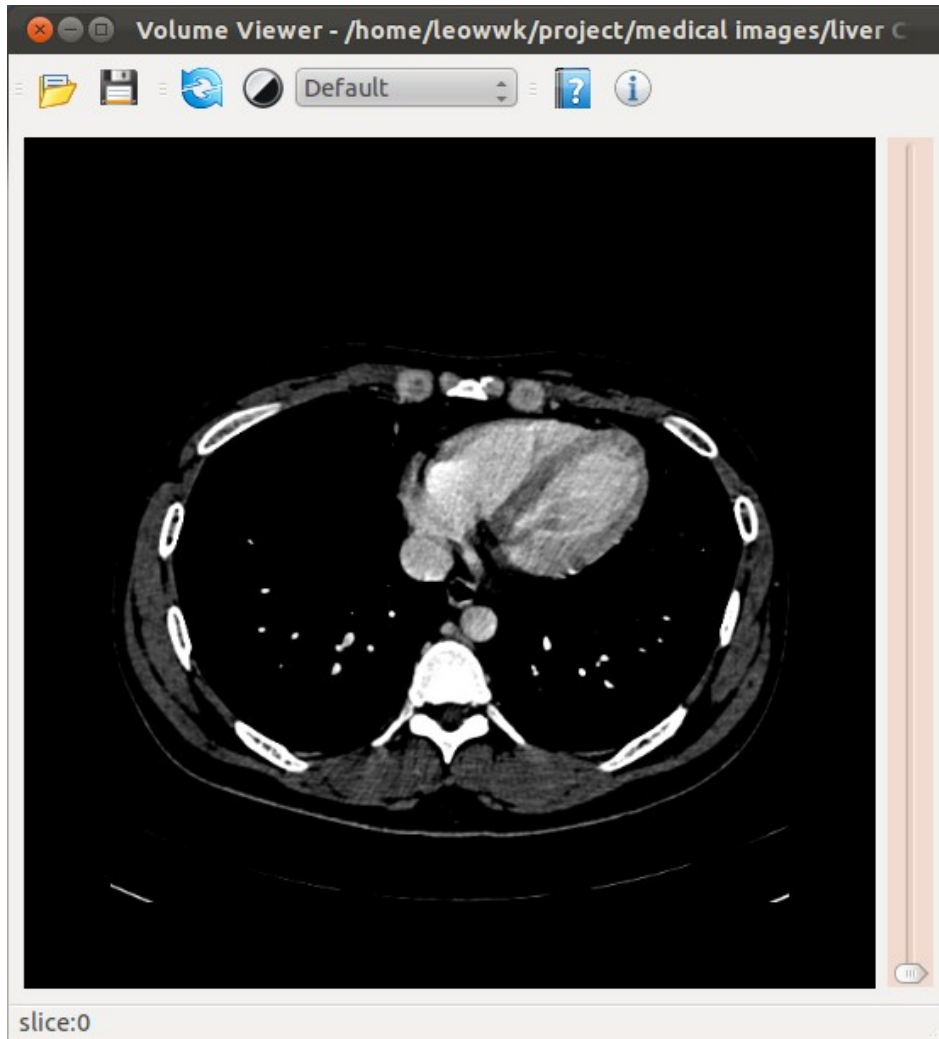


⦿ Set image slice to display

```
void VolumeViewer::setSlice(int slice)
{
    if (!reader)
        return;

    currentSlice = slice;
    viewer->SetSlice(slice);
    viewer->Render();
    statusBar()->showMessage(
        QString("slice:%1").arg(slice));
}
```

⦿ Examples:

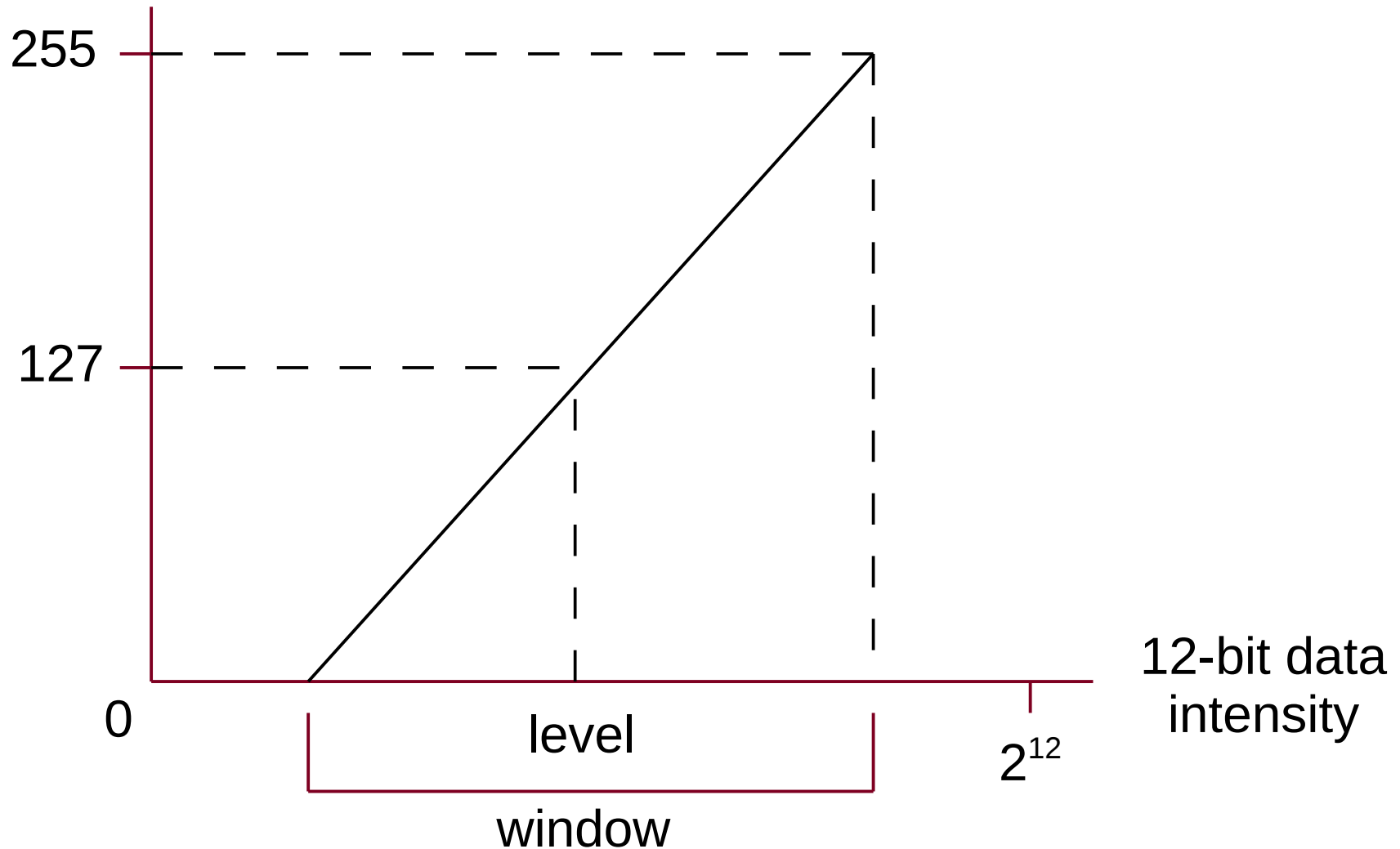


Intensity Window and Level

- ⦿ Normal images use 8-bit encoding for gray & colour.
- ⦿ Medical images (DICOM) use 12-bit encoding.
 - More than what a monitor can display.
- ⦿ Need to map 12-bit value to 8-bit display.
 - Called window and level in medical applications.
 - Effect is similar to contrast and brightness.

◉ Map 12-bit value to 8-bit display

8-bit display
intensity



```
void VolumeViewer::setWindowLevel(int index)
{
    int level, window;

    if (!reader)
        return;

    switch (index)
    {
        case 0: // Default
            level = 128;
            window = 256;
            break;

        case 1: // Abdomen
            level = 90;
            window = 400;
            break;

        ...
    }
}
```

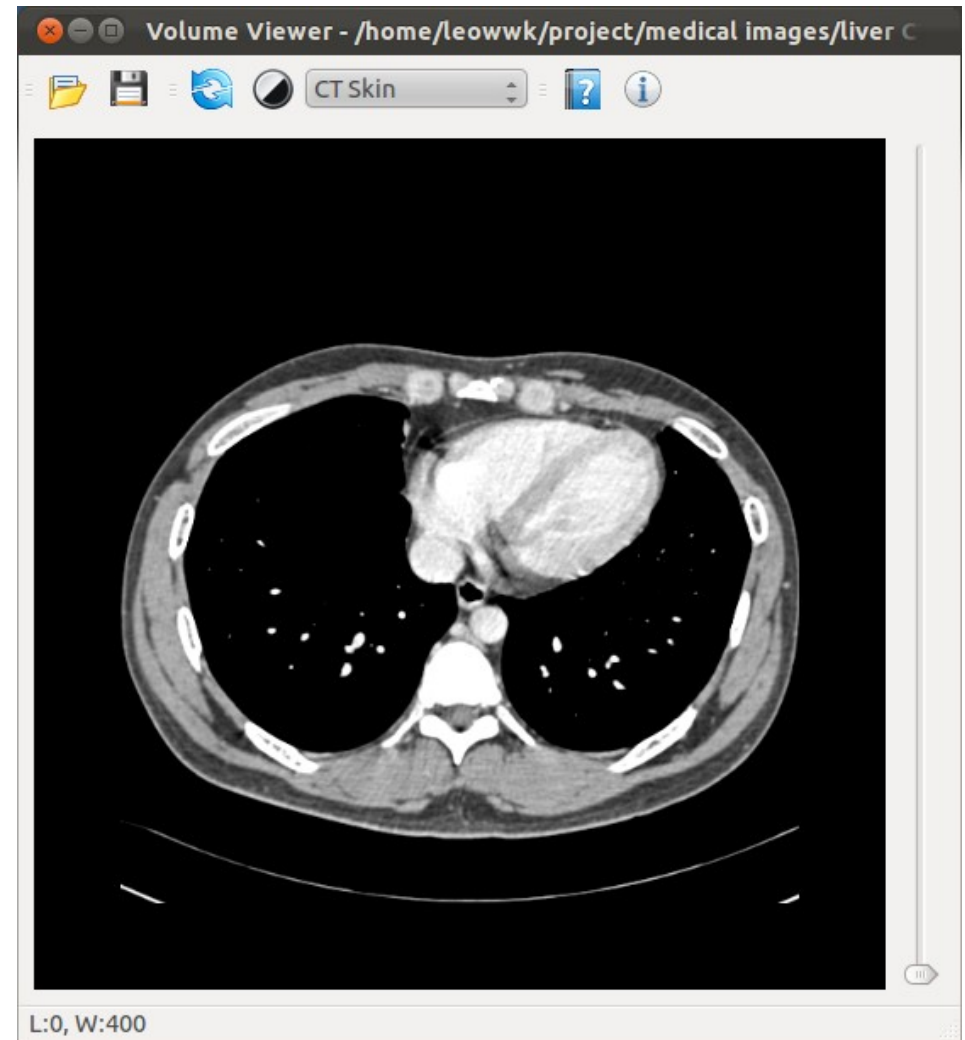
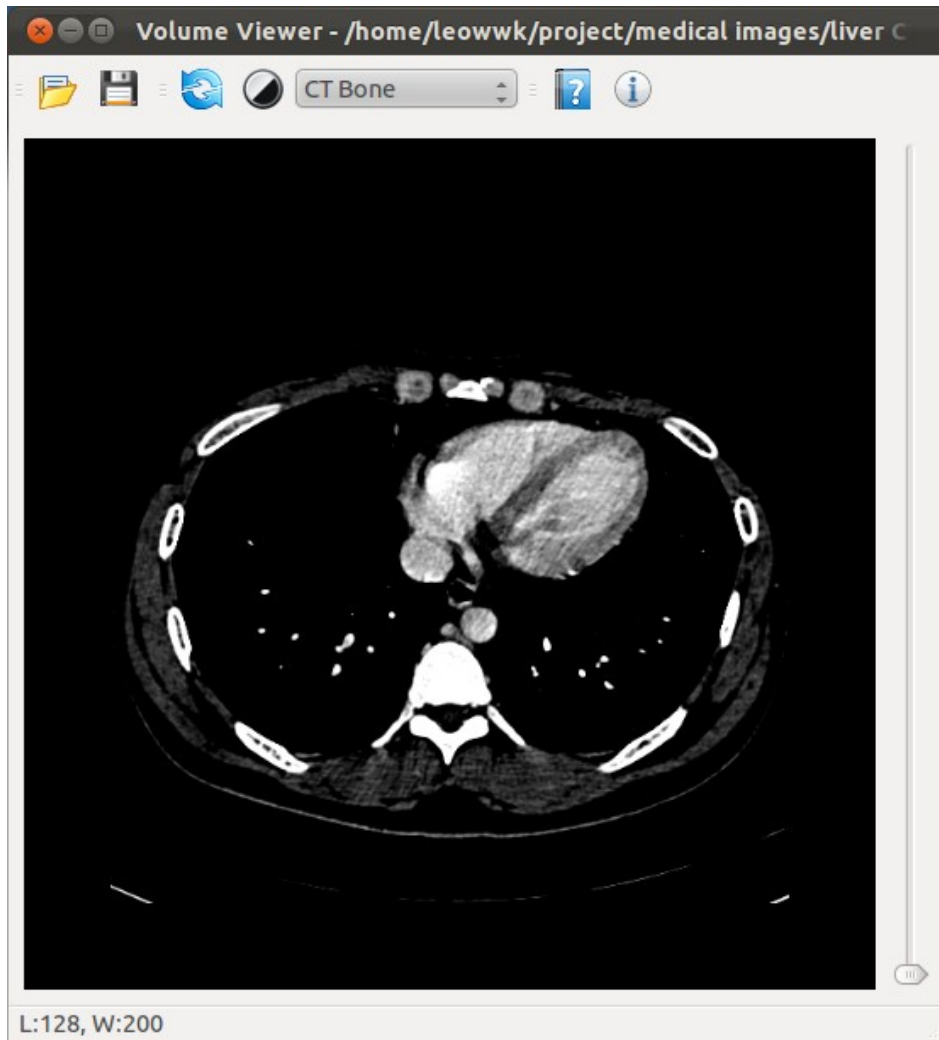
```
    case 6:  // Skin
        level = 0;
        window = 400;
        break;

    default:
        level = 128;
        window = 256;
        break;
}

viewer->SetColorLevel(level);
viewer->SetColorWindow(window);
viewer->Render();

statusBar()->showMessage(
    QString("L:%1, W:%2").arg(level).arg(window));
}
```

⦿ Examples:



```
void VolumeViewer::resetWindowLevel()
{
    if (!reader)
        return;

    viewer->SetColorLevel(128);
    viewer->SetColorWindow(256);
    viewer->Render();
    statusBar()->showMessage(QString("L:128, W:256"));
}
```

Summary

- ⦿ Volume rendering can render some voxels transparent.
 - Reveal interior of volume data.
 - Transfer functions map voxel intensity to opacity and colour.
 - Step size determines rendering resolution.
- ⦿ Volume image can be displayed as a stack of 2D image slices.
 - Use slider to control which slice to display.
 - Window-level maps 12-bit value to 8-bit display.

Exercises

- ⦿ Follow Lab 2 procedure to complete the programs of volume renderer and volume viewer.

Further Reading

- ⦿ Transfer functions: VTK documentation.
- ⦿ Volume rendering: [Schr2006] sections 7.3, 7.4.
- ⦿ QComboBox: [Blan2008], Qt Assistant.

References

- ⦿ J. Blanchette and M. Summerfield, *C++ GUI Programming with Qt 4*, 2nd ed., Prentice Hall, 2008.
- ⦿ W. Schroeder, K. Martin and B. Lorensen, *Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*, 4th Ed., Kitware Inc., 2006.