**Visualisation Assignment**

**Approach**

The visualisation was based around the Marching Cubes algorithm. The data is loaded from the 2D image slices and stored in a vtkImageData structure, effectively making a 3D point cloud where the scalar value of each point in the data structure is pixel value in the image slice. The Marching Cubes can be performed directly on this data structure. In my implementation, up to two isovalues can be supported and visualised either together or separately. The isovalues can be specified upon the function call.

Having two isovalues allows for rendering of different parts of the model. For example, in the CThead dataset, the skin has an isovalue of about INSERT ISOVALUE. The bone of the skull has an isovalue of INSERT ISOVALUE. By rendering them separately it lets us see the different in the parts of the body that are comprised of different materials. This is shown in figure 1.

**System Architecture**

The binary data is read one byte at a time and each scalar is saved in an array. This array is iterated through and used to fill a 3-dimensional vtkImageData array, a form of structured data, where each point in the array corresponds to a 3D coordinate, and the value of each point is the value of the pixel in the image. The vtk marching cubes implementation uses this data to generate the model. If the relevant options are set, the largest region of the volume is found with the others discarded, and smoothing is applied. If specified, clipping is done. The clipping can be updated later once the model is being rendered. The data is then sent to a vtkPolyDataMapper. This maps the geometric data to graphics primitives ready for rendering. This process is summarised in figure 1.

Binary data

3D geometric data

Marching Cubes model

(Largest region extraction, smoothing, clipping)

Mapper

Graphical model

Figure 1. The system architecture used in this renderer

**Results**

The calculated isosurfaces can now be visualised, as shown in figure 1. The rendered volume can be rotated, and the camera can zoom in and out. The inside of the model can also be viewed, as shown in figure 2. By clipping the model, we can view a cross section of the model in any of the three planes. This is very helpful as it allows us to view the internal structure of the model. The clipping is determined by an intersection of the model with a 2D plane, with the part of the model on one side of the plane being rendered, and the other part not. The parts of the model that are touching the clipping plane have been coloured differently from the rest of the model. This is to allow highlighting of the shape of the model at this cross section. The position of the clipping plane can be moved via keyboard input.

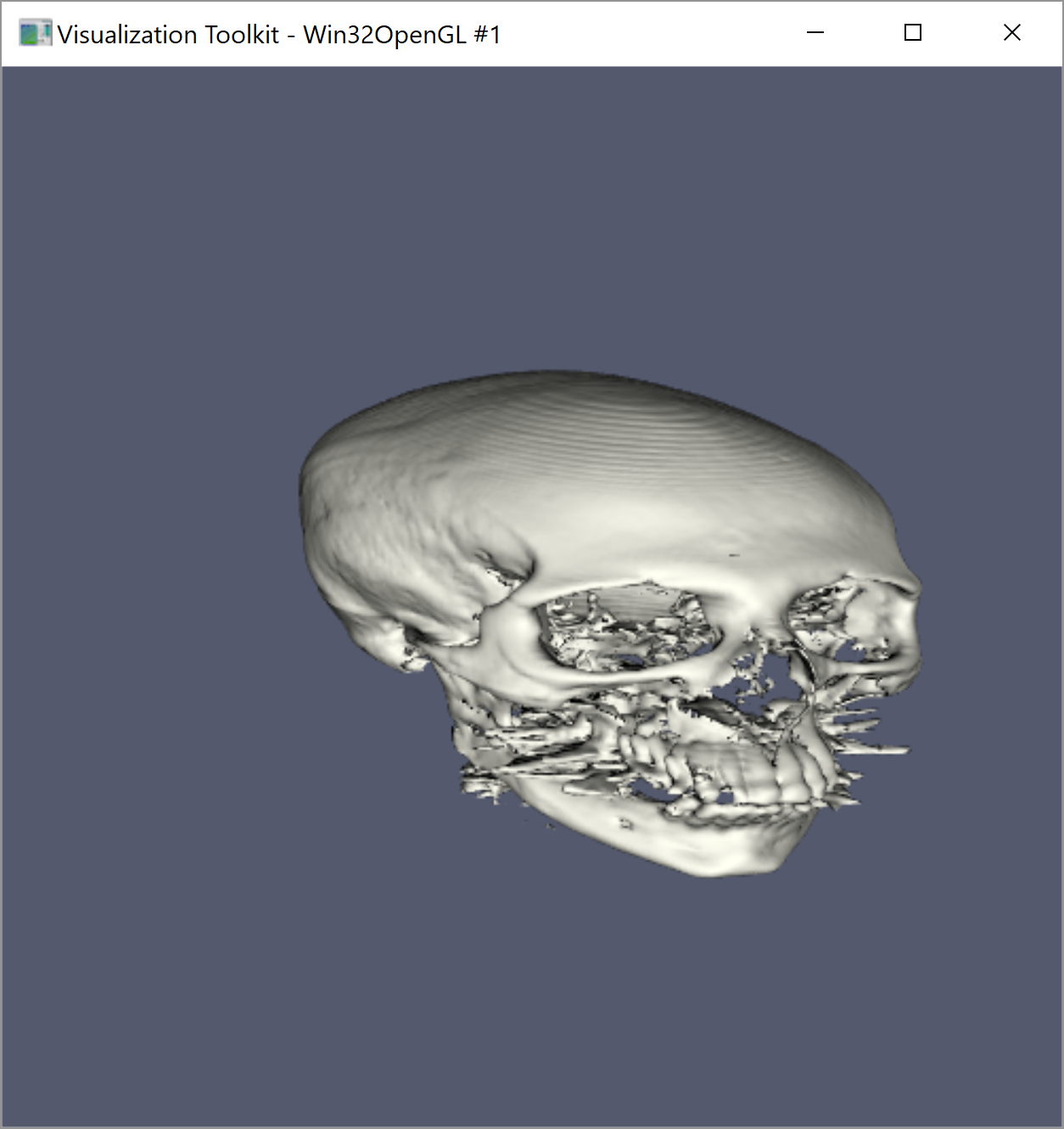


Figure 2. Skin isosurface on the left, bone isosurface on the right

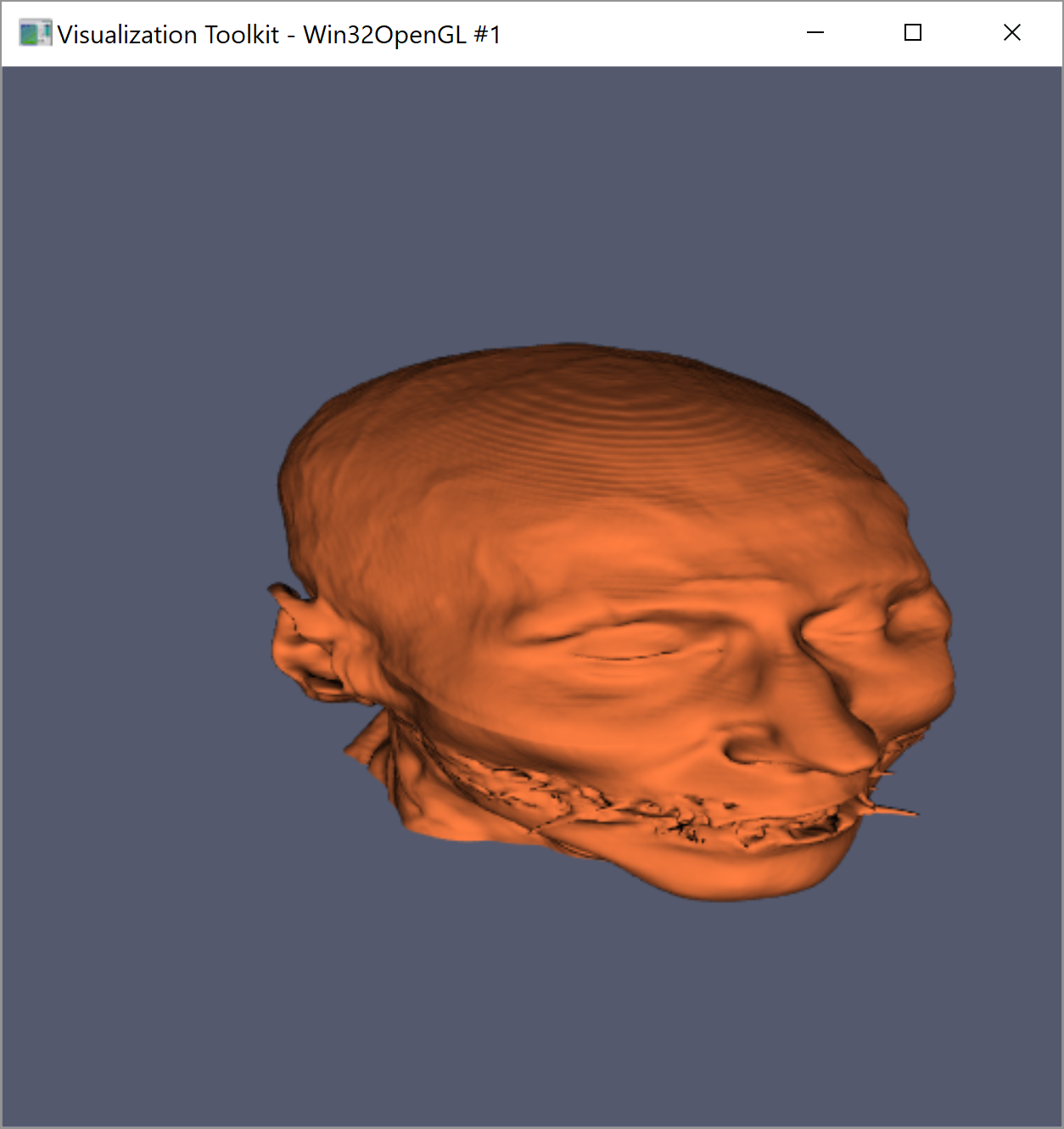
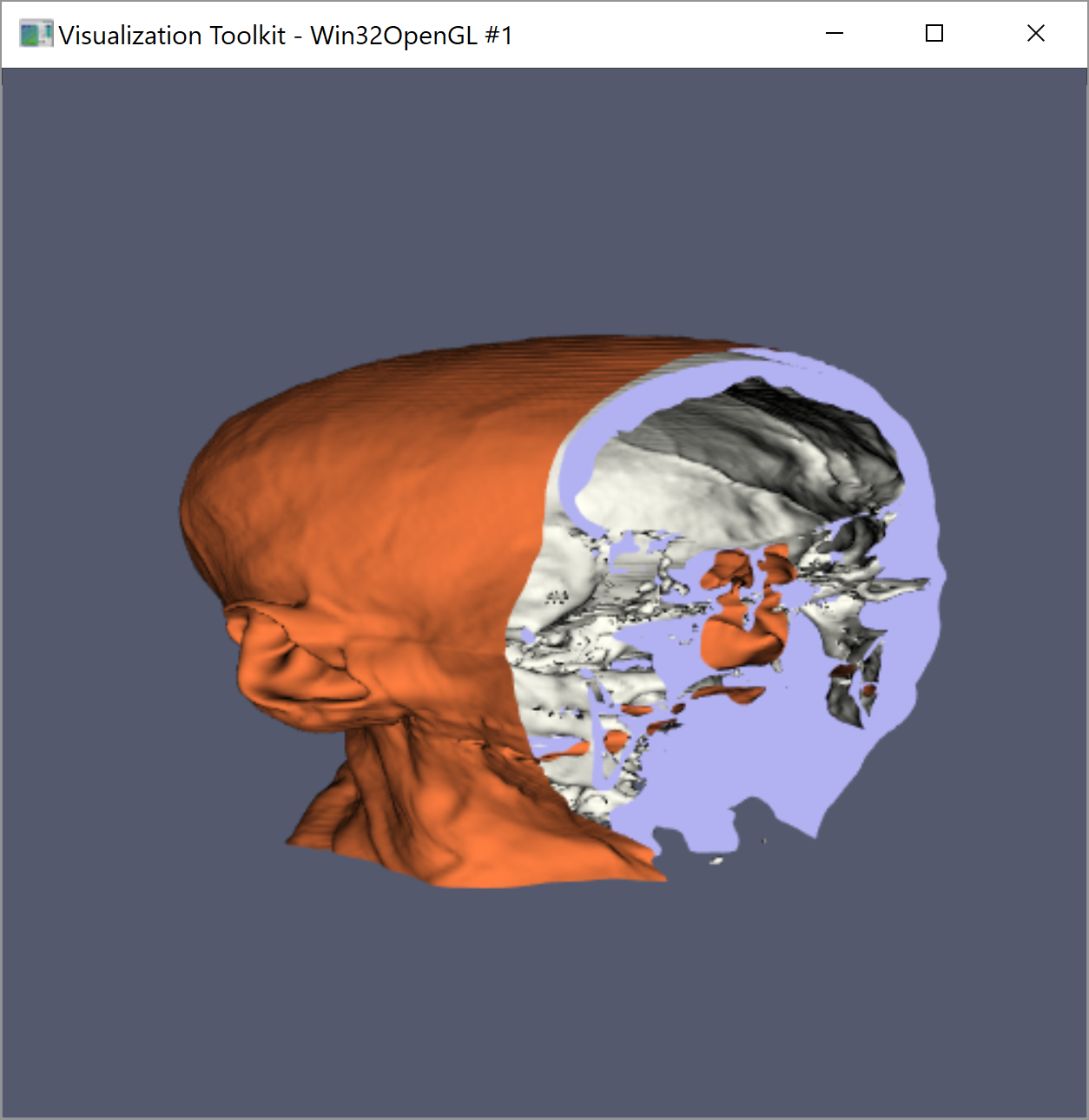
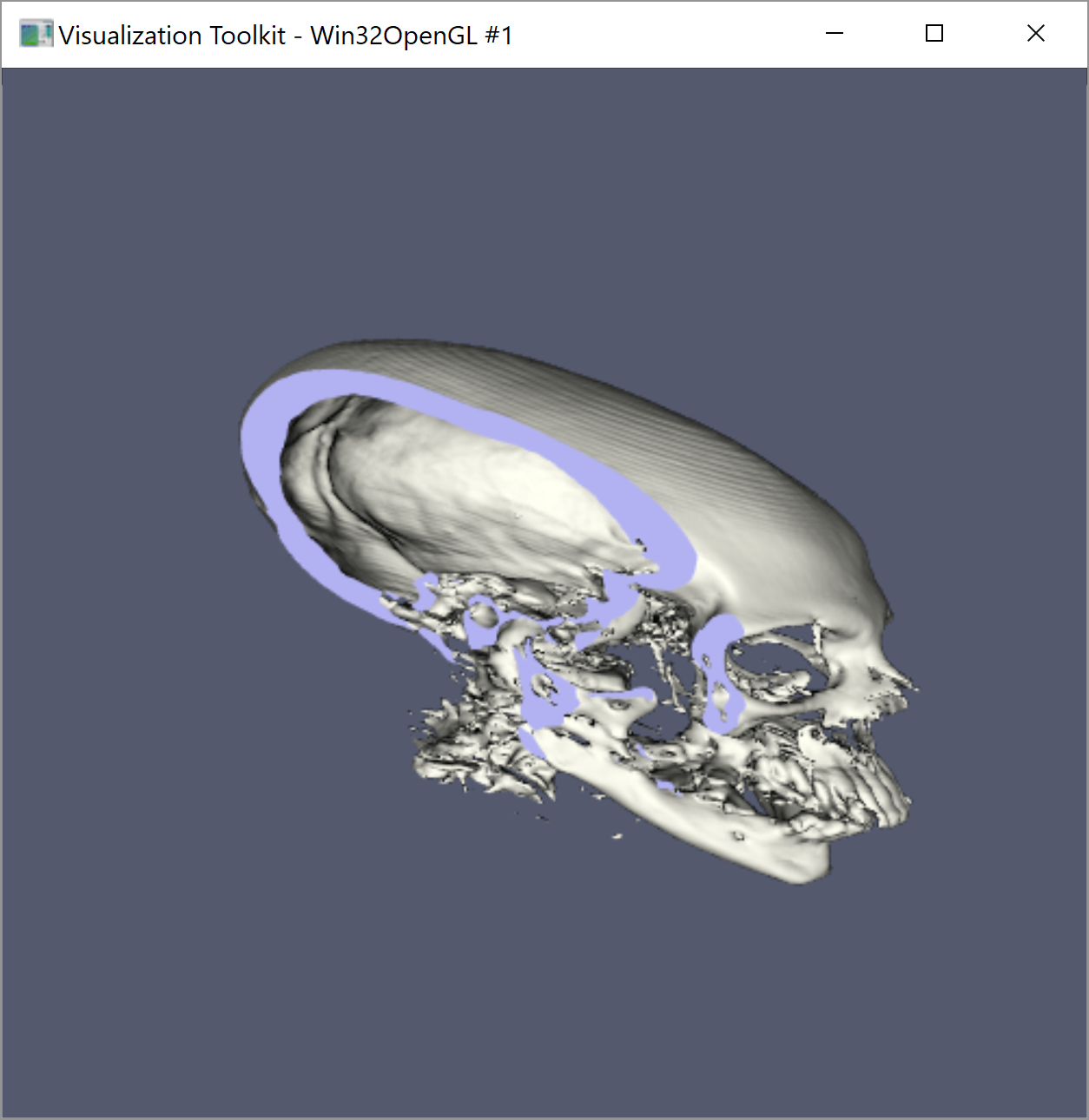
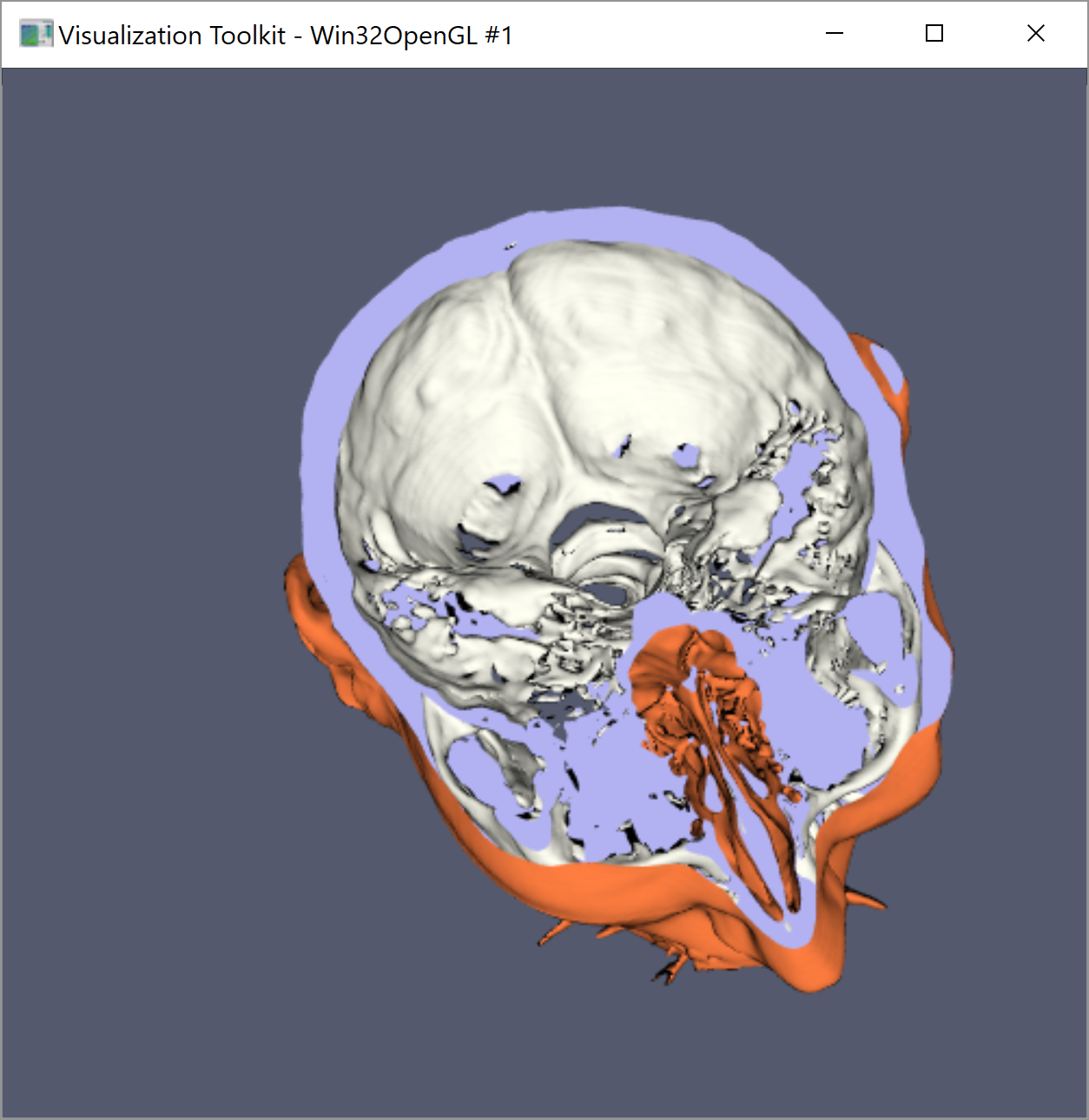
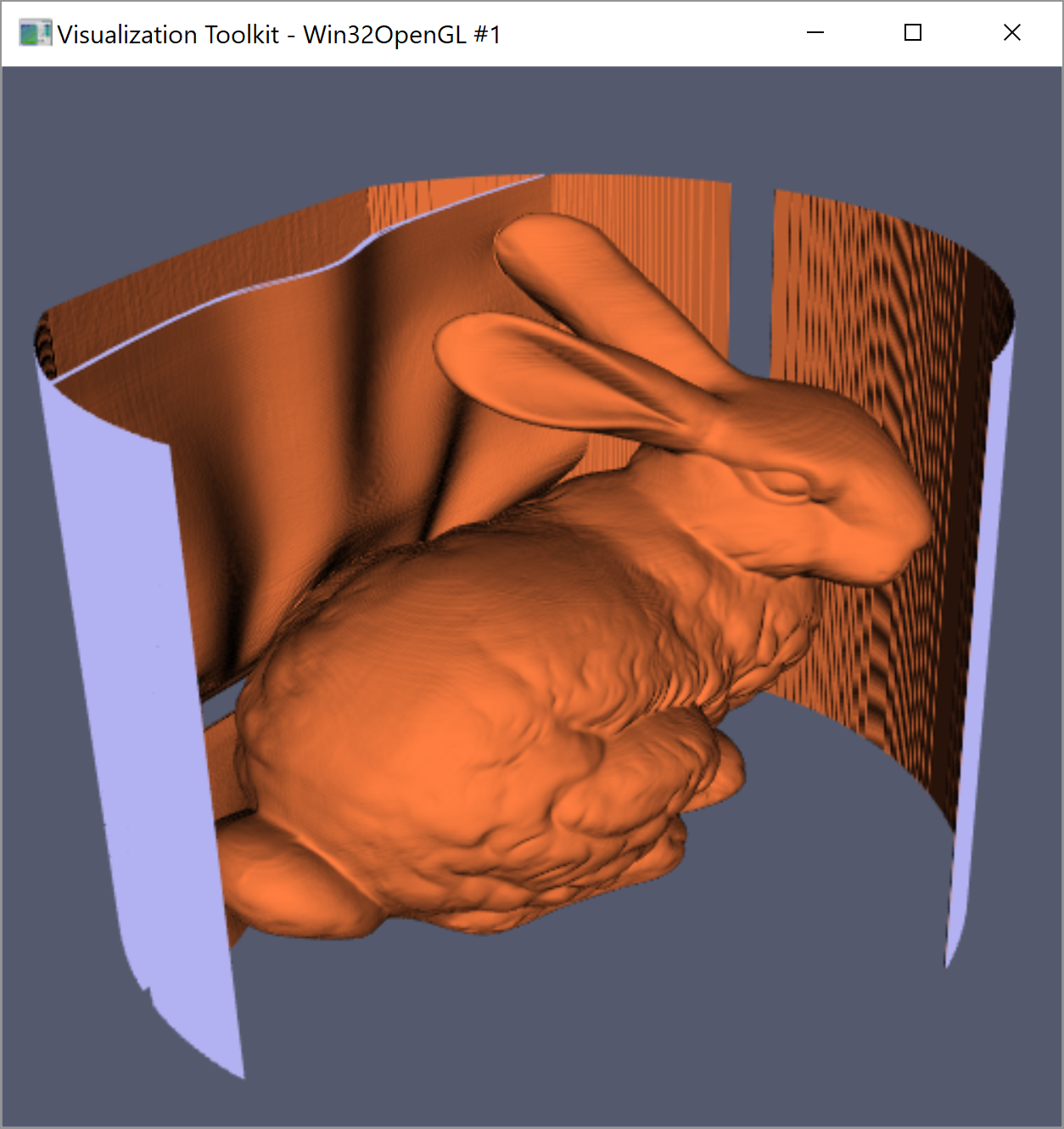
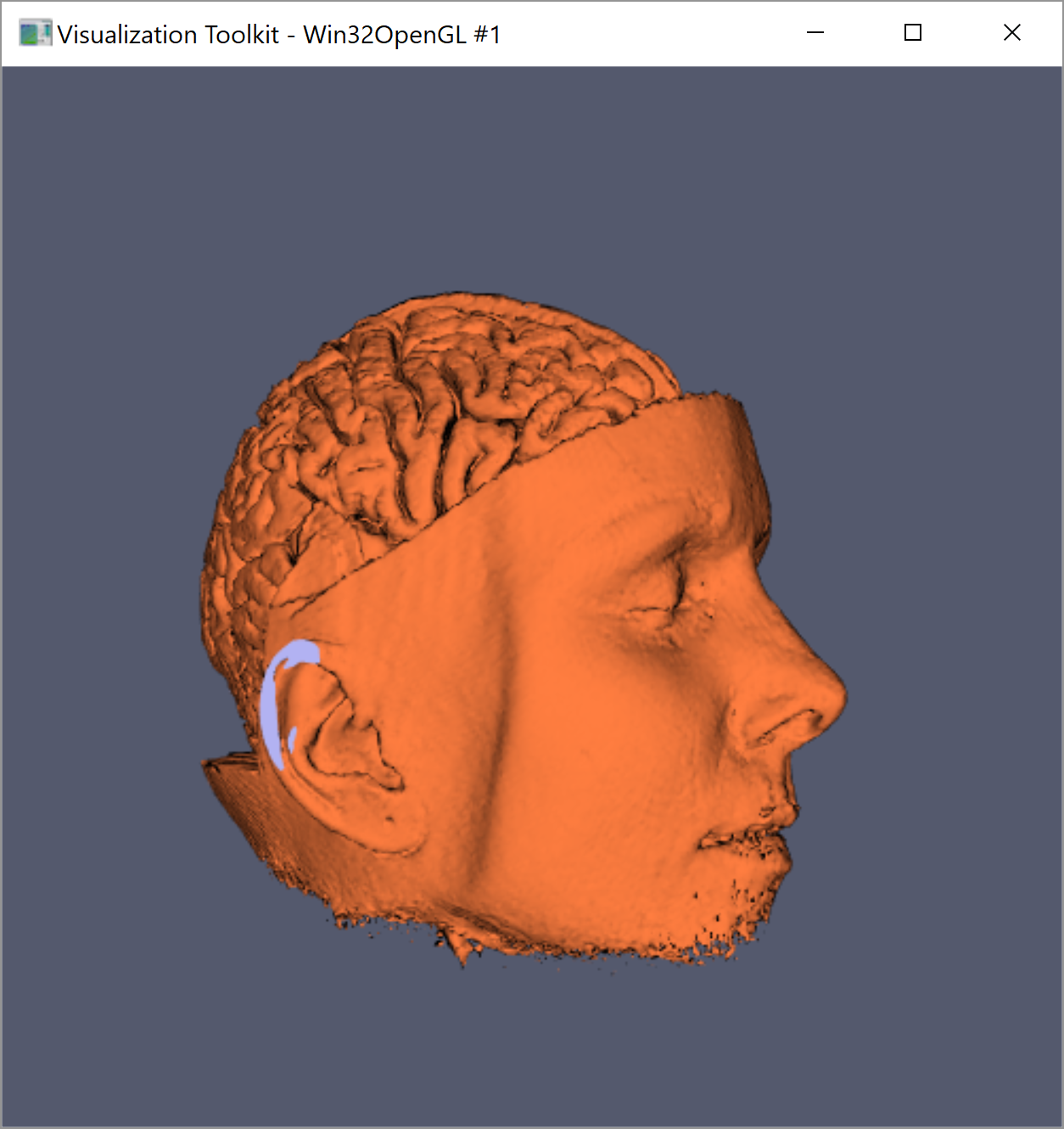


Figure 3. Cross-section visualisation in the X direction (left), Y direction (middle) and Z direction (right)



Other datasets did not benefit from multiple isovalues, as there was only one clear volume to render. Figure 3 shows the MRbrain dataset and the bunny dataset being rendered – two cases where only a single isovalue was sufficient. MRbrain gave the best model with an isovalue around 33% the value of the highest voxel value WHICH IS?, and bunny gave the best model with an isovalue around 1% the value of the highest voxel value WHICH IS?.

Figure 4. MRbrain dataset (right) and bunny dataset (left)



**Issues Faced**

There were sometimes issues with the initial data when rendering some volumes. For example, the CThead dataset has some flash-like artefacts around the mouth, presumably from the scanning machine that took the original images. This can be seen in figure 3.

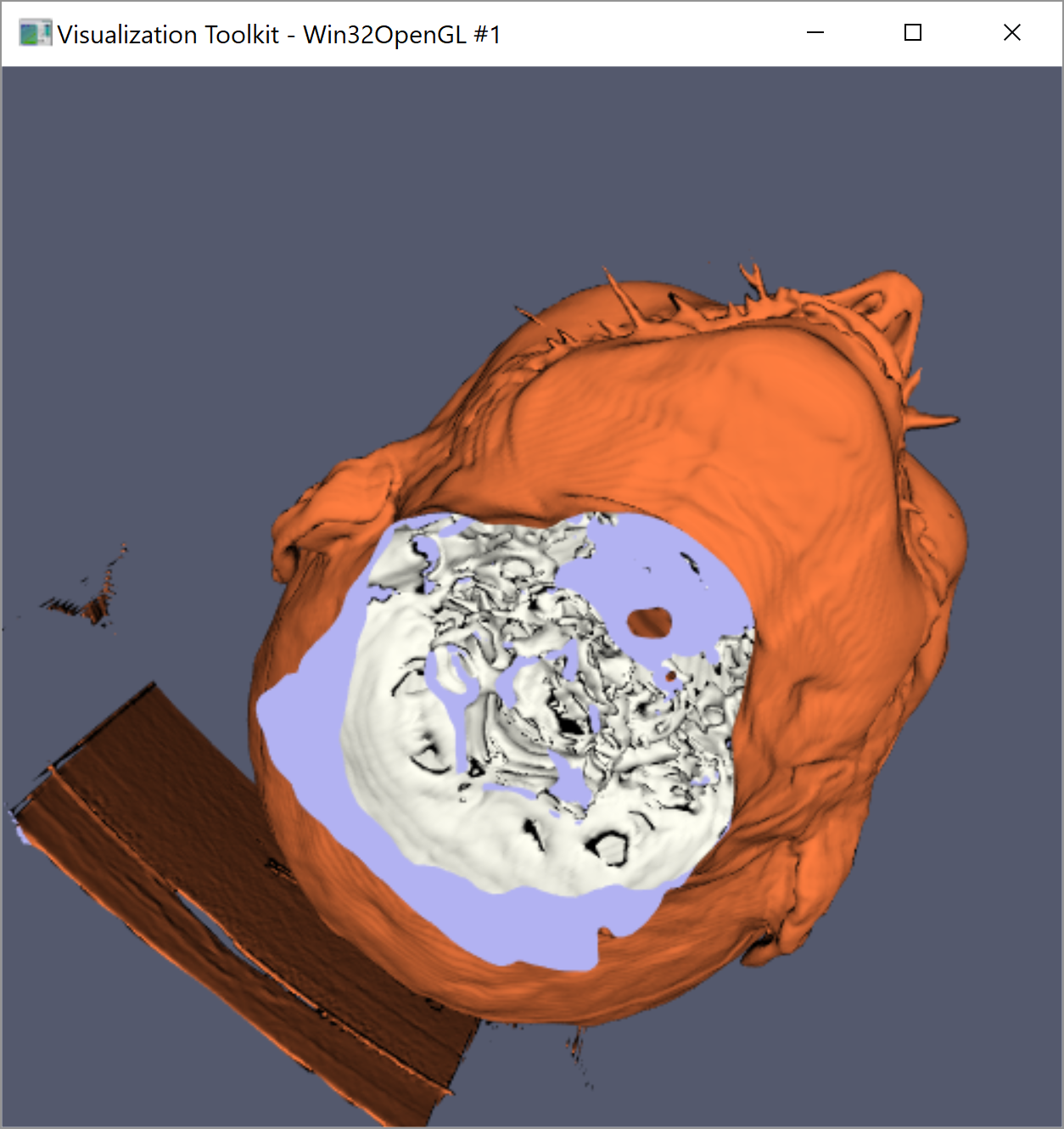


Figure 5. Artefacts in the original data (right) that are present in the generated model (left)

Another issue that affected the overall quality of the final model was the presence of artefacts from the borders in the original data. These borders could arise if the original scan image was circular and has been padded to make it a rectangle. This is the case with the bunny and CThead datasets, as shown in figure 4.

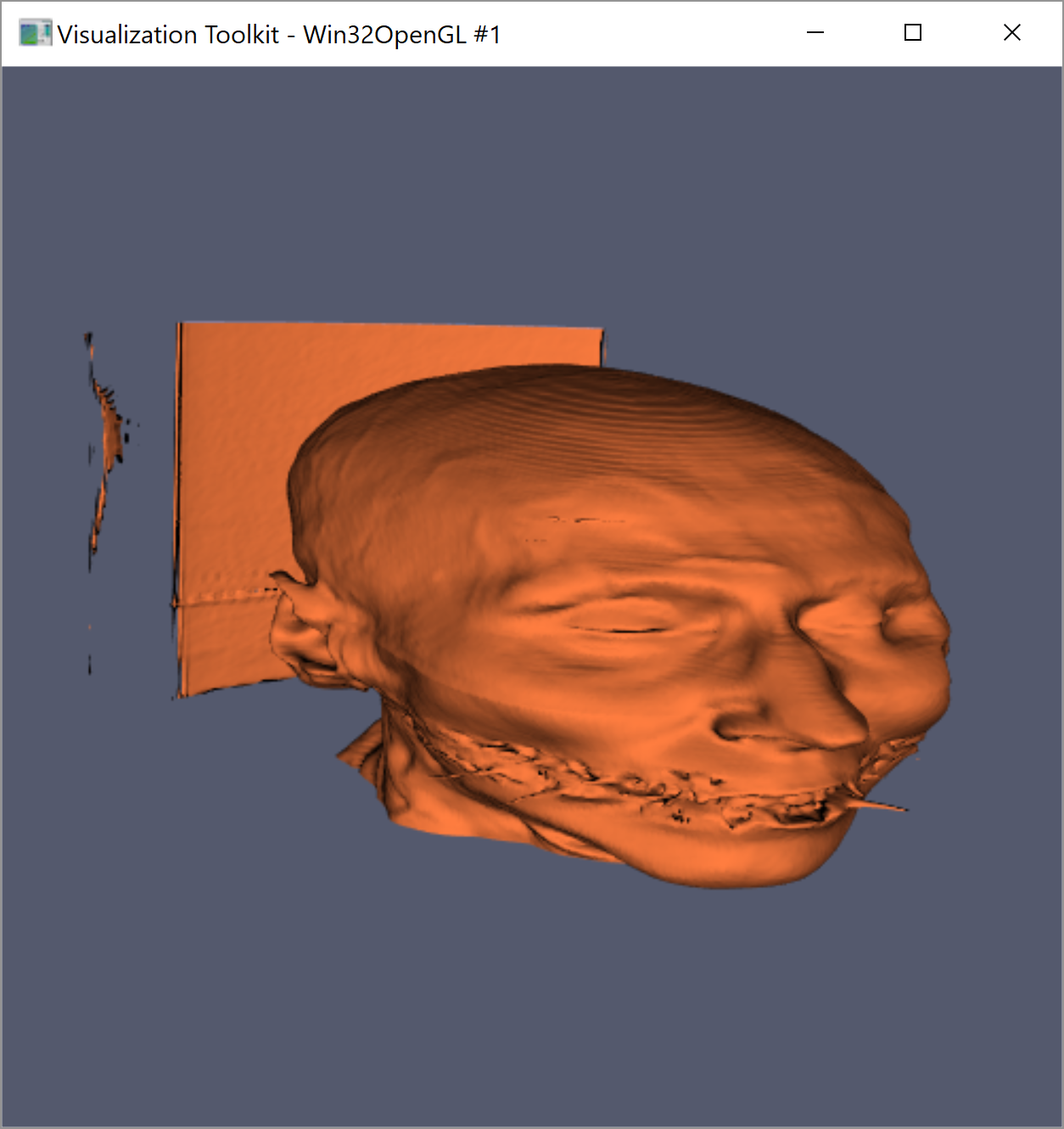
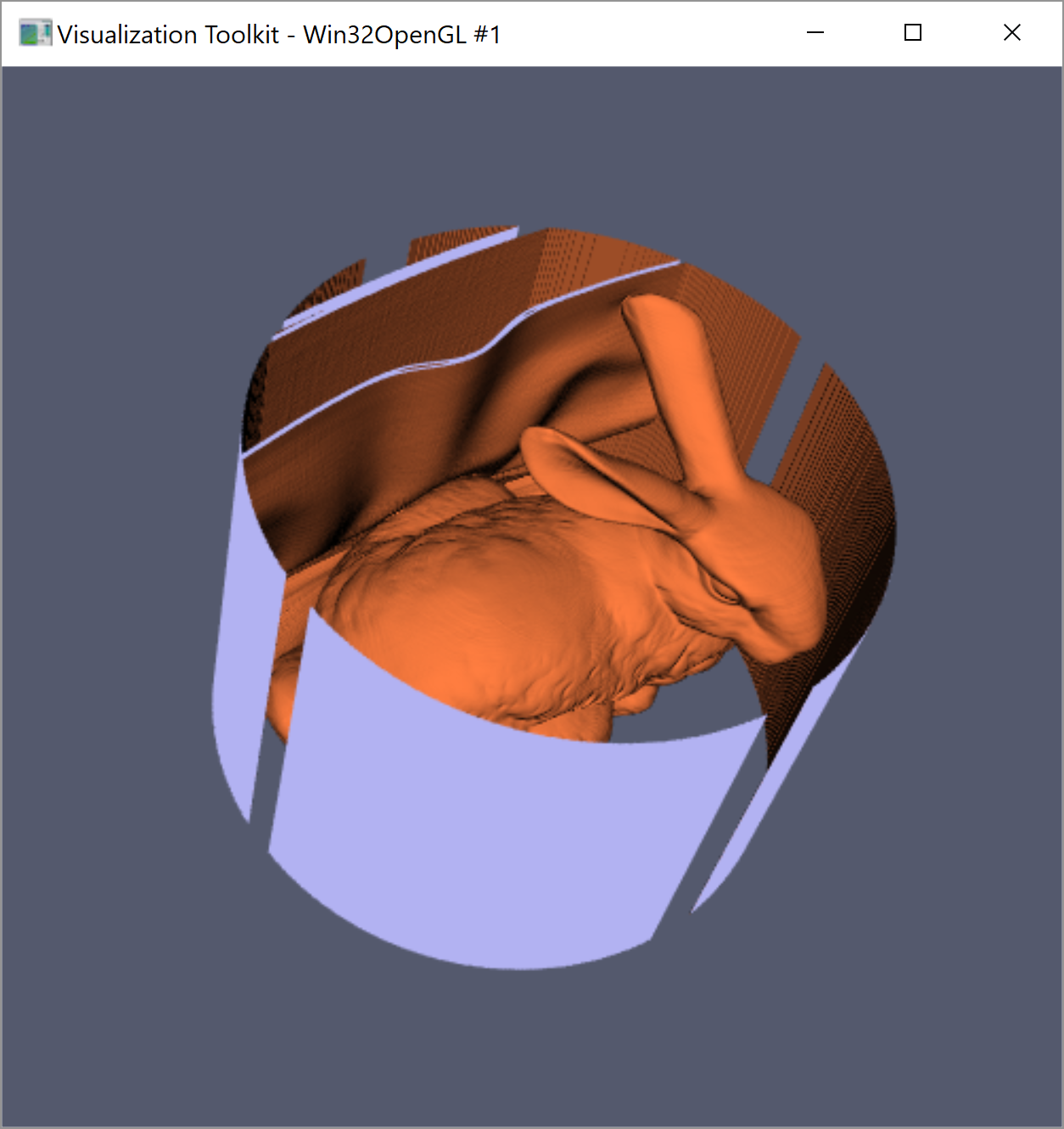


Figure 6. Extra parts to the model that are not part of the desired object. The rabbit has the circular wall around it, and the head has the wall at the back

This was mitigated by using vtkConnectivityFilter. This separates the model into separate regions that are connected. There is then an option to keep only the largest region, which is usually the part of the model that we want. This is true for the CThead dataset, where the largest region is the head. In figure 4 there is a wall-like artefact behind the head. However, it has been removed for the final render, as seen in figure 1. The bunny dataset was not as easy, and only one of the four circular panels as seen in figure 4 was separate to the rest of the model. As such, the final render of the rabbit contains the undesired panels that surround the rabbit.

**Other Features**

The colour orange was chosen as it is a bright colour that lets the shape of the model show up well with the lighting. It also contrasts nicely with the colour chosen for the cross-sectional view, again allowing for ease when making out the shape of the visualised model. The white was chosen for the CThead skull to reflect the colour of bone.

To do:

* CHECK FIGURE NUMBERS
* Insert isovalues
* Tidy up code
* Do mean/max thing?
* Comment on how picking a marching cube value is hard, and wrong value can mess it up?