# Shadows in Volume Rendering using Inviwo

Luis Carranza
luis.carranza@estudiantat.upc.edu
Universitat Politècnica de Catalunya
Barcelona, Spain

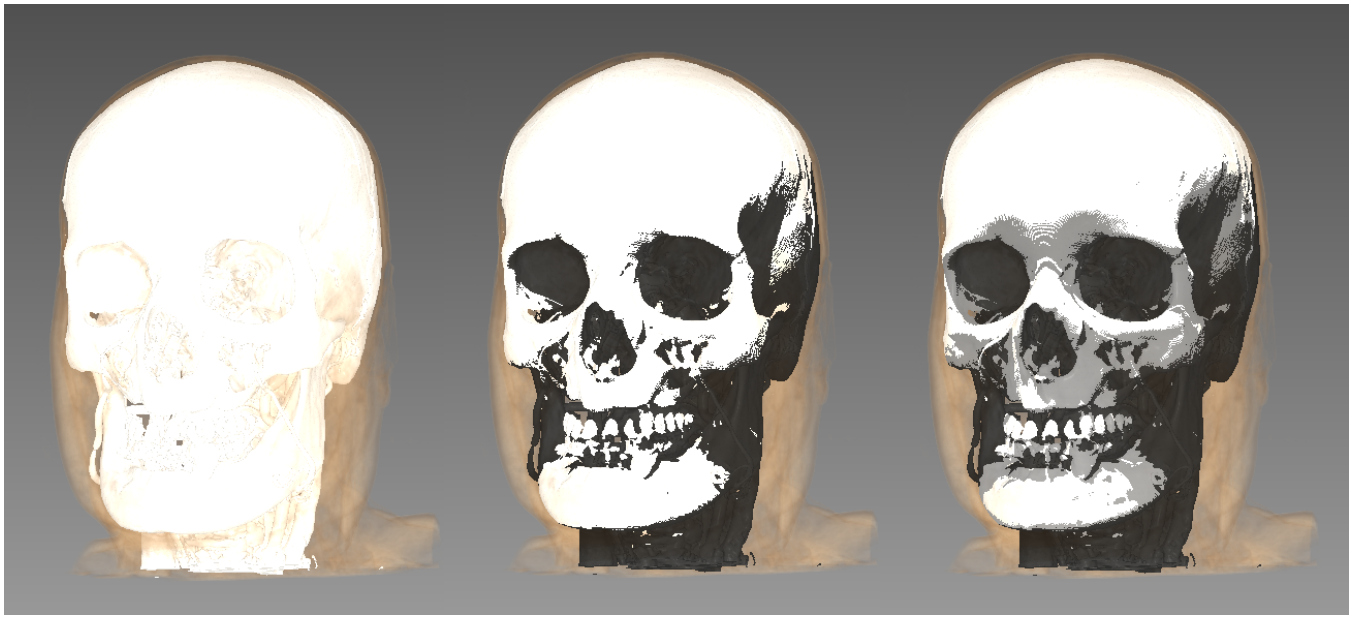

**Figure 1.** Left: No shadow technique. Middle: Hard shadows. Right: Soft shadows with 4 rays.

## Abstract

The project challenges students to implement the hard and soft shadowing techniques in a GPU-based raycaster. In this project, the implementation is be done using Inviwo.

***Keywords:*** shadowing techniques, Inviwo

## 1 Introduction

In order to enhance the volumetric rendering visualization, light positioning and shadows can add tridimensional information about the model. For this project, the tasks done can be divided in two sections: the configuration of the model in the Inviwo software and the implementation of shadowing techniques in the raycasting shader.

For the first section, a given model is uploaded and a set of processors are configured to allow the software render it properly. Also, the transfer function is composed to focus on the visualization of the bones rather than the skin.

In the second section, hard and soft shadowing techniques are implemented. To achieve this, the position where the ray from the camera stops and a light position will form another ray which will check for occlusions to determine an existing shadow.

## 2 Setup

The project has been developed in Windows OS and the software pre-installed to run it is the following:

- Inviwo Version 0.9.11
- MSVC 19.16.27032.1
- Qt Version 5.12.1

Once the software is installed and the workspace opened, the recommended visualization of windows for testing the parameters implemented can be seen in Figure 2. As for the configurable parameters in the raycasting shader, these can be found below the main function as showed in Figure 3.

## 3 Model and Transfer Function configuration

The volumetric model uploaded for this project is a nii format file which was placed inside the *Inviwo/0.9.11/data/volumes* folder. In order to get a proper visualization the following set of processors was configured in Inviwo as seen in Figure 4. In this workflow the Volume Source contains the .nii skull model, where the offset and scale parameters are computed automatically. This processor send the volumetric information to the Cube Proxy and Volume Raycaster processors. The Cube Proxy processor will allow the view of the interior

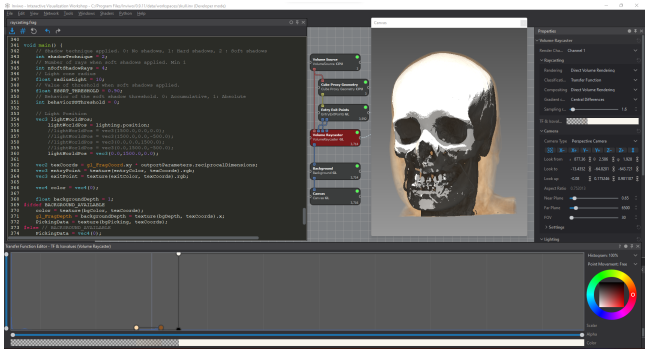

**Figure 2.** Recommended configuration of windows for testing parameters.

raycasting.frag

```
340
341 void main() {
342     // Shadow technique applied. 0: No shadows, 1: Hard shadows, 2 : Soft shadows
343     int shadowTechnique = 2;
344     // Number of rays when soft shadows applied. Min 1
345     int nSoftShadowRays = 4;
346     // Light cone radius
347     float radiusLight = 10;
348     // Value of threshold when soft shadows applied.
349     float ESSRT_THRESHOLD = 0.90;
350     // Behavior of the soft shadow threshold. 0: Accumulative, 1: Absolute
351     int behaviorSSThreshold = 0;
352
353     // Light Position
354     vec3 lightWorldPos;
355         lightWorldPos = lighting.position;
356         //lightWorldPos = vec3(1500.0,0.0,0.0);
357         //lightWorldPos = vec3(1500.0,0.0,-500.0);
358         //lightWorldPos = vec3(0.0,0.0,1500.0);
359         //lightWorldPos = vec3(0.0,1500.0,-500.0);
360         lightWorldPos = vec3(0.0,1500.0,0.0);
```

**Figure 3.** Configurable parameters in shader.

of the skull, by clipping the volumetric model with respect to the X, Y and Z axes in the world reference system. This processor will send the clipping modifications to the Entry Exit Points processor, which will generate two textures with the closest and furthest point volumetric positions the camera can view per pixel. These textures are sent to the Volume Raycaster processor, which in addition to a transfer function and the volumetric model information received, will obtain the color to render per pixel including the implemented shadows. Finally, a background the render image is sent to the background to add some background coloring and then to the canvas to visualize the final renderization.

As for the Transfer function, with the given model two categories are defined: the skin and the bones, other structures like the brain or veins tries to be discarded. Since the bones are the region to focus, an alpha 1.0 is set with a bone color, whereas the skin has an alpha of 0.03 with two skin colors for a more detailed visualization of the soft regions maintaining the transparency. The skin and bones can be seen separately in Figure 5. To maintain the alpha values constant among the volume, an alpha 0.0 is set immediately outside of the regions in the transfer function in order to get the correct position of ray termination when the camera ray direction touches the bone. By this, that position will be at the boundary of the bone so no self-collisions will occur when throwing a ray in light direction to determine the shadow.

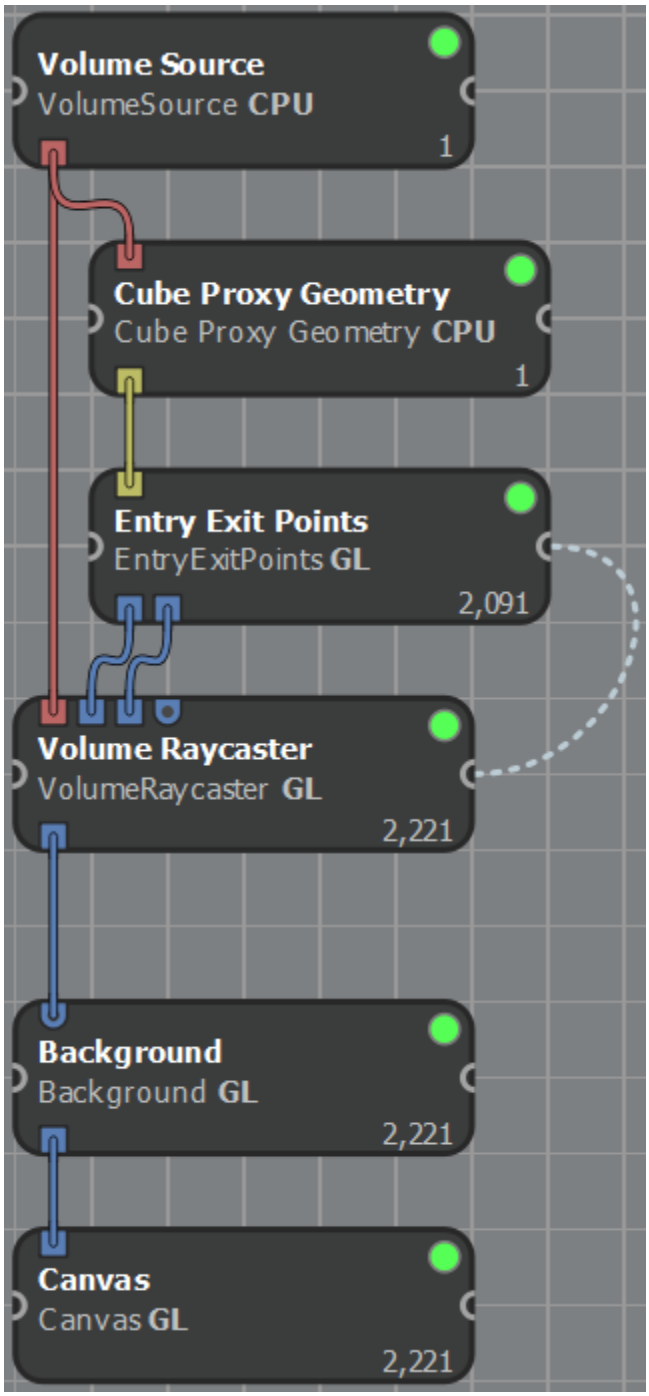


**Figure 4.** Processors workflow configuration in Inviwo.

## 4 Shadowing Technique

The shadowing techniques are implemented in the raycasting fragment, where the *rayTraversal* function is modified in

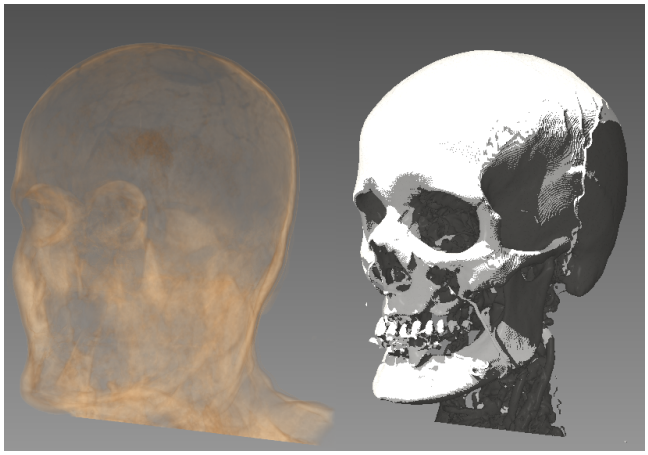

**Figure 5.** Skin and bones transfer function independent visualization.

order to evaluate shadows before returning the result. Specifically, once the *rayTraversal* sampling terminates, the last sample position (*samplePos*) is considered as the position the camera is looking to in that pixel. So, if by the end of the sampling the corresponding alpha reached the threshold, another ray is cast from *samplePos* to the light source direction to check for occlusions via a new function called *lightRayTraversal*. As previously seen, Figure 3 shows the configurable parameters in the shader with respect to the hard and soft shadows. To choose between no shadows, hard shadows, and soft shadows, the parameter is *shadowTechnique* with values 0, 1 and 2 respectively. In addition, *lightWorldPos* gets the light position set in Inviwo. While it works well with the view space, in the world space the range to move the light is very limiting, reason why this position can be modified within the shader and some examples are provided.

### 4.1 Hard shadows

Within the implementation of the hard shadow technique, *lightRayTraversal* throws a ray to check for occlusions and returns it in the alpha channel of the vector *hardShadow*. If this alpha is greater than the early ray termination threshold, it is classified as occlusion and the color return is 80% black and 20% the original color.

As for the *lightRayTraversal* function, samplePos and lightPos are sent as the entryPoint and exitPoint of the ray. The ray sampling is similar to the function *rayTraversal*, but the ray starts *raycaster.samplingRate*tIncr* after to avoid self occlusion. Furthermore, since is only needed the alpha channel from the sampling, the *APPLY CHANNEL CLASSIFICATION* suffices to get the alpha, updating the result by the result of that function divided by the samplingRate to maintain coherence when increasing this parameter for smoother results.

### 4.2 Soft Shadows

When applying soft shadows, additional parameters are set such as the radius of the light source, the number of rays per pixel, the value of the occlusion threshold for soft shadows, and the behaviour accumulative or absolute of the threshold.

Given the lightDirection, light center, and a radius, the equation of a circle perpendicular to the lightDirection can be generated for random samples simulating the area of the light. Each of these samples will execute the *lightRayTraversal* to check for occlussions, and an average of them will be used to determine the shadow, which could be maximum 80% black and 20% the original color. If the behavior is absolute, the *lightRayTraversal* function will use the value of the occlusion threshold for soft shadows instead of the general early threshold for an earlier termination.

## 5 Analysis

Figures 6 and 7 consistently shows that shadowing techniques enhances the tridimensional information in the image in comparison to not applying it. Furthermore, soft shadows add a sense of light size that adds more details in comparison to hard shadows like the cheek shadowing in the front-view.

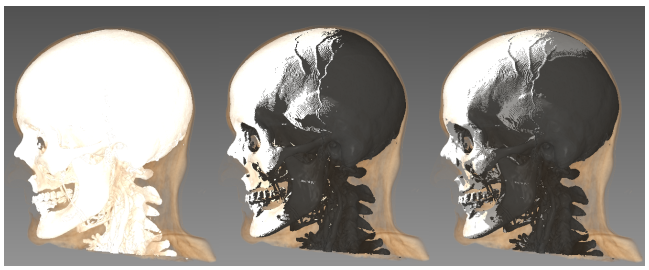

**Figure 6.** Side-view. Left: No shadow technique. Middle: Hard shadows. Right: Soft shadows with 4 rays.

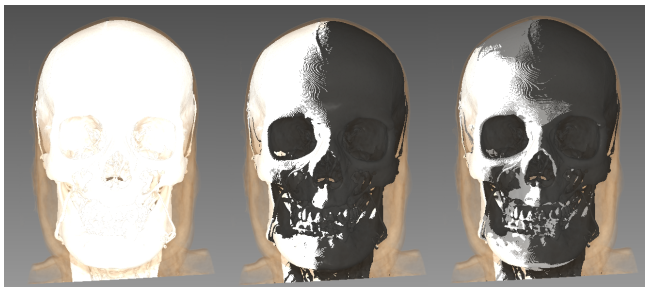

**Figure 7.** Front-view. Left: No shadow technique. Middle: Hard shadows. Right: Soft shadows with 4 rays.

### 5.1 Sampling Rate

With a sampling-rate of one there is some noise in both hard and soft shadows that can be smoothed by increasing this parameter. As seen in Figure 8, increasing this value to two reduces the noise while maintaining interactive response.

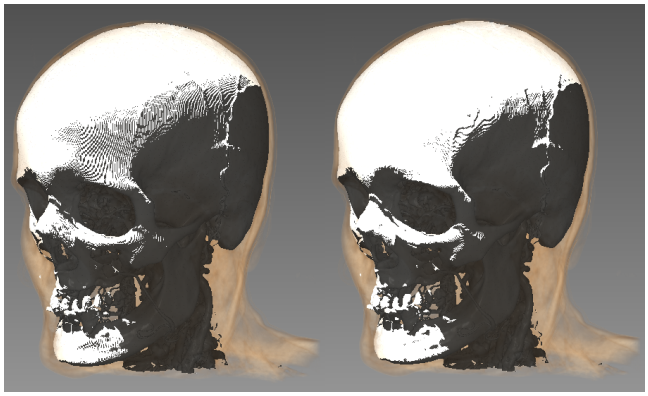

**Figure 8.** Left: Sampling rate = 1. Right: Sampling rate = 2.

### 5.2 Light radius

Bigger light radius maintaining the same number of rays would lead to incorrect shadows. As seen in Figure 9, from a radius of 1000 some occlusions start to appear incorrectly, so it is recommended to have a radius below this size for testing or change to hard shadows.

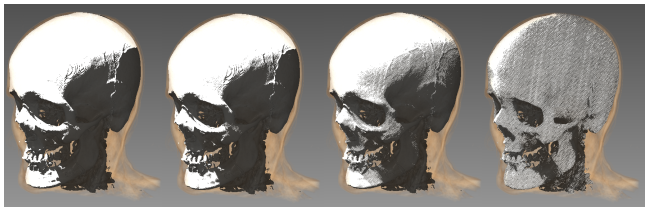

**Figure 9.** Soft shadows with 4 rays. Left: Radius = 10. Left-Center: Radius = 100. Right-Center: Radius = 1000. Right: Radius = 10000.

### 5.3 Number of rays

Increasing the number of rays for soft shadows increases the detail at cost of performance. As seen in Figure 10, increasing this value to five gives a good sampling while maintaining interactive response.

### 5.4 Cube Proxy Geometry

By pruning the volume for the entry and exit point, it is possible to see the interior of the volume. However, the *rayLightTraversal* function cannot ignore the voxels pruned. Therefore, the interior side of the model will always have shadows as seen in Figure 11

**Figure 10.** Left: nRays = 2. Left-Center: nRays = 4. Center: nRays = 5. Right-Center: nRays = 8. Right: nRays = 16.

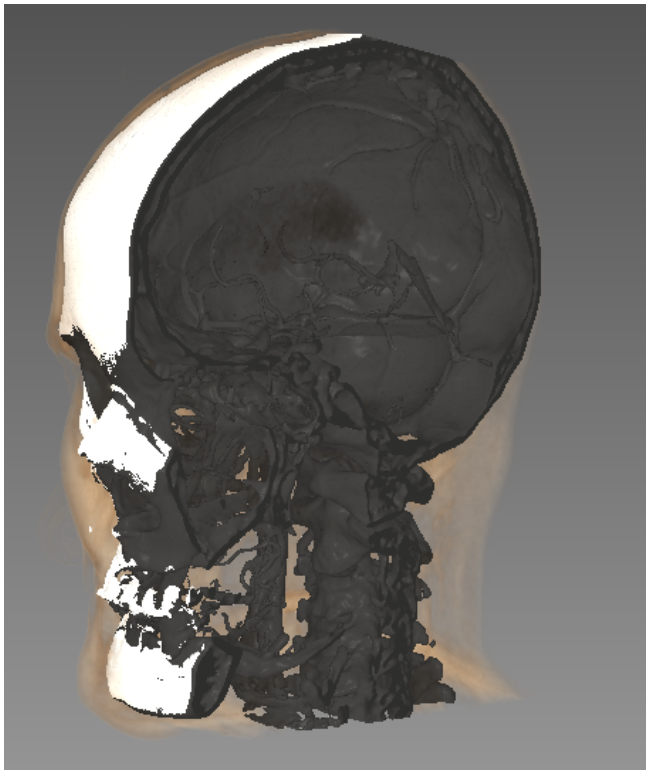

**Figure 11.** Occlusions with skull opened.