Volumetric Image Visualization

Alexandre Xavier Falcão

LIDS - Institute of Computing - UNICAMP

afalcao@ic.unicamp.br

Transparency is a valuable resource to view the location of one object inside the other.

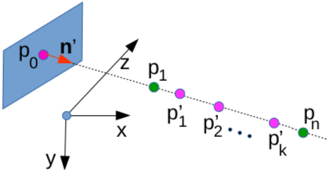
Introduction

Transparency is a valuable resource to view the location of one object inside the other.

This lecture covers object transparency in the surface rendering algorithm.

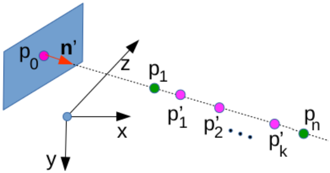
Ray casting with transparency

Let $0 \leq o_i.\alpha \leq 1$ be the opacjty of object o_i and p'_i, $i = 1, 2, \ldots, k$, be the **first** intersection points of a viewing ray and $1 \leq k \leq c$ **distinct** objects of the scene.



- We will assume that refraction does not change the direction of the ray.

Let $0 \leq o_i.\alpha \leq 1$ be the opacjty of object o_i and p'_i, $i = 1, 2, \ldots, k$, be the **first** intersection points of a viewing ray and $1 \leq k \leq c$ **distinct** objects of the scene.



- We will assume that refraction does not change the direction of the ray.

- Being the first intersection points for each object guarantees that they are visible surfaces — i.e., $0 < \langle o_i.\mathbf{n}(p'_i), -\mathbf{n}' \rangle \leq 1$, $1 \leq i \leq k$ —, depending on the transparencies of their occluding objects o_j, $1 \leq j < i$, only.

Ray casting with transparency

- Let $r(p'_i)$ be the reflection of the light source at each point p'_i, $i = 1, 2, \ldots, k$, as computed by the Phong's model.

Ray casting with transparency

- Let $r(p'_i)$ be the reflection of the light source at each point p'_i, $i = 1, 2, \ldots, k$, as computed by the Phong's model.

- Due to color reflectance, $\mathbf{c}(p'_i) = r(p'_i) \times o_i.\mathbf{r}$, where $o_i.\mathbf{r} = (o_i.R, o_i.G, o_i.B)$, is the reflected light by object o_i.

Ray casting with transparency

- Let $r(p'_i)$ be the reflection of the light source at each point p'_i, $i = 1, 2, \ldots, k$, as computed by the Phong's model.
- Due to color reflectance, $\mathbf{c}(p'_i) = r(p'_i) \times o_i.\mathbf{r}$, where $o_i.\mathbf{r} = (o_i.R, o_i.G, o_i.B)$, is the reflected light by object o_i.
- Due to opacity, only a portion $o_1.\alpha \times \mathbf{c}(p'_1)$ returns to the observer from o_1, a portion $(1 - o_1.\alpha) \times o_2.\alpha \times \mathbf{c}(p'_2)$ returns from o_2, and so on, such that the light reflected towards the observer is:

$$\mathbf{c}(p'_1, p'_2, \ldots, p'_k) = \sum_{i=1}^{k} \left[\Pi_{j=1}^{i-1}(1 - o_j.\alpha)\right] \times o_i.\alpha \times \mathbf{c}(p'_i)$$

Ray casting with transparency

- Let $r(p'_i)$ be the reflection of the light source at each point p'_i, $i = 1, 2, \ldots, k$, as computed by the Phong's model.

- Due to color reflectance, $\mathbf{c}(p'_i) = r(p'_i) \times o_i.\mathbf{r}$, where $o_i.\mathbf{r} = (o_i.R, o_i.G, o_i.B)$, is the reflected light by object o_i.

- Due to opacity, only a portion $o_1.\alpha \times \mathbf{c}(p'_1)$ returns to the observer from o_1, a portion $(1 - o_1.\alpha) \times o_2.\alpha \times \mathbf{c}(p'_2)$ returns from o_2, and so on, such that the light reflected towards the observer is:

$$\mathbf{c}(p'_1, p'_2, \ldots, p'_k) = \sum_{i=1}^{k} \left[\Pi_{j=1}^{i-1} (1 - o_j.\alpha) \right] \times o_i.\alpha \times \mathbf{c}(p'_i)$$

- For the rendition $\hat{J} = (D_J, \mathbf{J})$, $\mathbf{J}(p) \leftarrow \mathbf{c}(p'_1, p'_2, \ldots, p'_k)$.

Graphical context

The graphical context gc may then store.

- Transformations ϕ^{-1} and ϕ_r^{-1}.
- Phong parameters k_a, k_s, k_d, n_s, H, $d_{\min}$, $d_{\max}$.
- The scene $\hat{I} = (D_I, I)$ and its object label image $\hat{L}(D_I, L)$, such that $L(p) = i$, $i = 1, 2, \ldots, c$, when $p \in D_I$ belongs to one of the c objects, or $L(p) = 0$ otherwise.

Graphical context

- look-up table $\mathbf{n}[i]$ with pre-computed normal vectors.
- A normal index map N with the address $N(p) = i$ of the normal vector in $\mathbf{n}[i]$ for each boundary voxel $p \in D_I$, or *nil* when p is not a boundary voxel.
- A table with the visual attributes for each object o_i, $i = 1, 2, \ldots, c$,
  - color $o_i.\mathbf{r} = (o_i.R, o_i.G, o_i.B)$, where $o_i.X \in [0, 1]$,
  - opacity $0 < o_i.\alpha \leq 1$, and
  - visibility $o_i.v \in \{0, 1\}$.

Surface rendering algorithm with opacities

Input : Graphical context gc, and viewing angles α and β.
Output: Rendition $\hat{J} = (D_J, \mathbf{J})$.

```
n' ← φ_r^{-1}(n), where n = (0, 0, 1, 0).
For each p ∈ D_J do
    p_0 ← φ^{-1}(p).
    Find P = {p_1, p_n} by solving ⟨p_0 + λn' − f.c, f.n⟩ = 0
    for each face f ∈ F of the scene, whenever they exist.
    if P ≠ ∅ then
        J(p) ← ComputeColorAlongRay(gc, P).
```

Surface rendering algorithm with opacities

- ComputeColorAlongRay($gc, \mathcal{P}$) is the DDA algorithm modified to compute the Phong's model with opacities along a viewing ray.

$$\mathbf{c}(p'_1, p'_2, \ldots, p'_k) = \sum_{i=1}^{k} \left[\Pi_{j=1}^{i-1}(1 - o_j.\alpha)\right] \times o_i.\alpha \times \mathbf{c}(p'_i)$$

Surface rendering algorithm with opacities

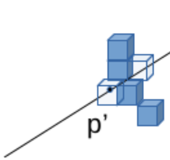
- ComputeColorAlongRay$(gc, \mathcal{P})$ is the DDA algorithm modified to compute the Phong's model with opacities along a viewing ray.

$$\mathbf{c}(p'_1, p'_2, \ldots, p'_k) \;=\; \sum_{i=1}^{k} \left[\Pi_{j=1}^{i-1}(1 - o_j.\alpha)\right] \times o_i.\alpha \times \mathbf{c}(p'_i)$$

- The accumulated transparency $\left[\Pi_{j=1}^{i-1}(1 - o_j.\alpha)\right]$ can be stored in a variable t, initially set to 1 and used for early ray termination.

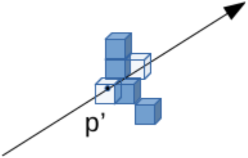
Surface rendering algorithm with opacities

- A surface point flag $f_s \in \{0, 1\}$ to stop the adjacency search that avoids missing a boundary voxel along the ray.

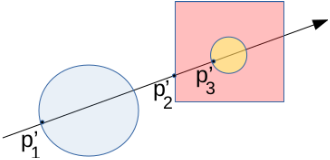


Surface rendering algorithm with opacities

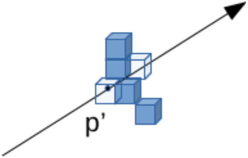
- A surface point flag $f_s \in \{0, 1\}$ to stop the adjacency search that avoids missing a boundary voxel along the ray.



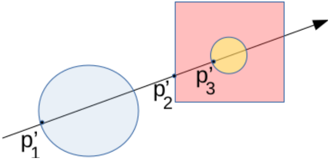
- Object flags $o_i.f \in \{0, 1\}$, $i = 1, 2, \ldots, c$, to avoid projecting more than one boundary voxel of a same object.



- A surface point flag $f_s \in \{0, 1\}$ to stop the adjacency search that avoids missing a boundary voxel along the ray.



- Object flags $o_i.f \in \{0, 1\}$, $i = 1, 2, \ldots, c$, to avoid projecting more than one boundary voxel of a same object.



- The total color $\mathbf{c}_t = \mathbf{c}(p'_1, p'_2, \ldots, p'_k)$ along the ray.

Computing color along a viewing ray

1 If $p_1 = p_n$ then set $n \leftarrow 1$.

2 Else

3 Set $D_x \leftarrow x_{p_n} - x_{p_1}$, $D_y \leftarrow y_{p_n} - y_{p_1}$, $D_z \leftarrow z_{p_n} - z_{p_1}$.

4 If $|D_x| \geq |D_y|$ and $|D_x| \geq |D_z|$ then

5 Set $n \leftarrow |D_x| + 1$, $d_x \leftarrow sign(D_x)$, $d_y \leftarrow \frac{d_x D_y}{D_x}$, and $d_z \leftarrow \frac{d_x D_z}{D_x}$.

6 Else

7 If $|D_y| \geq |D_x|$ and $|D_y| \geq |D_z|$ then

8 Set $n \leftarrow |D_y| + 1$, $d_y \leftarrow sign(D_y)$, $d_x \leftarrow \frac{d_y D_x}{D_y}$, and $d_z \leftarrow \frac{d_y D_z}{D_y}$.

9 Else

10 Set $n \leftarrow |D_z| + 1$, $d_z \leftarrow sign(D_z)$, $d_x \leftarrow \frac{d_z D_x}{D_z}$, and $d_y \leftarrow \frac{d_z D_y}{D_z}$.

Computing color along a viewing ray

Set $k \leftarrow 1$, $t \leftarrow 1.0$, $o_i.f \leftarrow 0$, $i = 1, 2, \ldots, c$, $p' \leftarrow (x_{p_1}, y_{p_1}, z_{p_1})$, and $\mathbf{c}_t \leftarrow (0, 0, 0)$.

While $k \leq n$ and $t > \epsilon$, do

  Set $p' \leftarrow (\lceil x_{p'} \rceil, \lceil y_{p'} \rceil, \lceil z_{p'} \rceil)$ and $f_s \leftarrow 0$.

  For each $q \in \mathcal{A}_1(p')$ and while $f_s = 0$, do

    If $N(q) \neq nil$ and $o_{L(q)}.f = 0$ then

      Set $f_s \leftarrow 1$.

      If $o_{L(q)}.v = 1$ and $o_{L(q)}.\alpha > 0$, do

        $\mathbf{c}_t \leftarrow \mathbf{c}_t + t \times o_{L(q)}.\alpha \times r(q) \times o_{L(q)}.\mathbf{r}$.

        $t \leftarrow t \times (1 - o_{L(q)}.\alpha)$ and $o_{L(q)}.f \leftarrow 1$.

  Set $p' \leftarrow (x_{p'}, y_{p'}, z_{p'}) + (d_x, d_y, d_z)$

return $\mathbf{c}_t$.

Note that the rendition might change the original color of the objects.