Telluride

A Viewfactor Based Radiative Heat Transfer Model for Telluride

Austen Minnich
Michael Hall
John Turner
LANL
UC Berkeley
Outline

1. Where this fits into Telluride
2. Radiosity Ideas and Background
3. View factors
4. The algorithm
5. Results from the algorithm
6. Future Work
Before the casting operation begins, the mold is heated.

The final temperature distribution of the mold is given by:

\[ \frac{\partial \rho T}{\partial t} + \rho \mathbf{u} \cdot \nabla T = \nabla \cdot (\mathbf{k} \nabla T) + q_{\text{conv}} + q_{\text{cond}} + q_{\text{rad}} \]

The conduction and convection modules are already in Telluride.

We need the final temperature distribution of the mold to pass to Telluride.

Where this fits into Telluride.
We need the radiation term, \( q_{\text{rad}} \).

Currently, Telluride uses the boundary condition:

\[
\begin{align*}
(q_{\text{rad}} w - \nu L) \sigma \kappa V &= -\nabla \mathcal{O}
\end{align*}
\]

So we need a more accurate model to make sure the final distribution of the mold is correct.

Since thermal radiation varies as \( T^4 \), radiative heat transfer is very important in determining the final temperature. \( K \) is the only factor for viewfactors, emissivity, etc.

\( K \) is the only factor for viewfactors, emissivity, etc.

For the \( q_{\text{rad}} \) term:
This is what we have done this summer.

Our code takes into account:

- viewfactors
- occlusion
- radiation

A more accurate model.
Radiosity is often used in graphics to model scenes with diffuse surfaces. It has one very simple idea: The total energy leaving a face is equal to the sum of the emitted energy and the reflected energies. Mathematically, this reads:

$$\mathbf{B}_i = \sigma e T^4_i + \rho_i \sum_{j} \mathbf{B}_j \mathbf{F}_{ij} \rho_i$$

$$\mathbf{F}_{ij}$$ is the view factor from face $i$ to face $j$. $\mathbf{B}_i$ are the total energies from faces $i$, $\rho_i$ is the reflectance of face $i$. $\sigma e$ is the Stefan-Boltzmann constant. $T^4_i$ is the temperature of face $i$.

Radiosity is often used in graphics to model scenes with diffuse surfaces.
If we rewrite the last equation like this:

\[ \sum_{\nu} \mathcal{J} B \mathcal{L} \mathcal{d} - B = \nu \mathcal{L} \mathcal{d} \]

This is the deceptively simple radiosity equation.

We can turn the last expression into a system of equations:

\[
\begin{bmatrix}
\frac{u}{ \mathcal{L} \mathcal{d} } \\
\vdots \\
\frac{z}{ \mathcal{L} \mathcal{d} } \\
\frac{1}{ \mathcal{L} \mathcal{d} } \\
\end{bmatrix} = \begin{bmatrix}
u_{\mathcal{B}} \\
\vdots \\
\nu_{\mathcal{B}} \\
\nu_{\mathcal{B}} \\
\end{bmatrix} \begin{bmatrix}
I \\
\vdots \\
\vdots \\
I \\
\end{bmatrix}
\]

\[= \begin{bmatrix}
u_{\mathcal{B}} \\
\vdots \\
\vdots \\
\nu_{\mathcal{B}} \\
\end{bmatrix} \begin{bmatrix}
u_{\mathcal{B}} \\
\vdots \\
\vdots \\
\nu_{\mathcal{B}} \\
\end{bmatrix} = \begin{bmatrix}
u_{\mathcal{B}} \\
\vdots \\
\vdots \\
\nu_{\mathcal{B}} \\
\end{bmatrix}
\]
The Radiosity Equation

Properties of the Radiosity Equation

1. The solution of the Radiosity equation is the total energy leaving each face.

2. It is guaranteed to converge by Gauss-Seidel or other iterative methods because it is diagonal dominant.

3. In order to generate this system of equations, we must calculate view factors.

4. Once we solve the system, we can easily determine the net radiant flux, $q_{rad}$. 

The Radiosity Equation
But first we must figure out the viewfactors.

**Definition:** A viewfactor is the fraction of energy emitted from an area $A^i$ in all directions directly intercepted by another area $A^j$.

The formal mathematical definition of a viewfactor is:

$$dF_{ij} = \frac{\int_{-\pi}^{\pi} \int_{-\pi}^{\pi} \frac{n^i \cdot n^j \cdot \rho}{\pi \| R_{ij} \|^2} d\theta d\phi}{\int_{-\pi}^{\pi} \int_{-\pi}^{\pi} \rho \cdot n^i \cdot n^j \cdot V_{ij} d\theta d\phi} = F_{ij}$$

We evaluate this expression in its vector form:

$$dF_{ij} = \frac{\int_{-\pi}^{\pi} \int_{-\pi}^{\pi} \frac{n^i \cdot n^j \cdot \rho}{\pi \| R_{ij} \|^2} d\theta d\phi}{\int_{-\pi}^{\pi} \int_{-\pi}^{\pi} \rho \cdot n^i \cdot n^j \cdot V_{ij} d\theta d\phi} = F_{ij}$$

Viewfactors
Figure 1: Example between two differential areas
Why viewfactors are important

• Viewfactors are essential to obtaining an accurate solution.

• They tell us how much energy goes from each face i to every other face j.

• This leads to a very accurate model for radiative heat transfer.

• This makes our final temperature distribution more accurate.

• So viewfactors are essential to obtaining an accurate solution.
Figure 2: The Nusselt unit sphere.

FIGURE 5.26: Geometry of unit-sphere method for obtaining configuration factors.
The algorithm is:

We need to form the viewfactor matrix so we can solve the system of equations. But we can't just calculate the viewfactors for every face. We have to make sure that the faces are visible to each other first.
Test to see if faces i and j see each other

For all faces i and j test to see if

If yes calculate view factor

If no compare view factor

another pair

If yes calculate

solve system of equations

when finished

Output solution

Figure 3: General flow chart for the algorithm
First Dot Product Test: Check to see if the faces point in the same direction by taking the dot product of the two area vectors. Check if the dot product is positive or negative.

Figure 4: dot product is negative

Figure 5: dot product is positive

This would pass.

This would not pass.
This test eliminates those faces pointing in opposite directions.

The Second Dot Product Test.
Occlusion

If the pair of faces pass the dot product tests, then we know they face each other.

But, we need to make sure there is nothing in the way before we can calculate the view factor.

In other words, we need to make sure, from the point of view of face \( i \), that there is no face \( k \) in the way of face \( j \).

Figure 7: Example of a possible occlusion.
The Occlusion Routine

Occlusion is very hard to detect. Here is how the program does it.

For each face j

Rotate Coordinate System

Project into xy plane

For each face i

For each face k

Project into xy plane

Compare Projections

If overlap: use closest face in calculation

If none overlap: use face j in calculation

If overlap: store index of overlapping face

If no overlap: use another face

If none overlap: use face j in calculation

If overlap: use closest face in calculation

Figure 8: Flow chart for the occlusion subroutine.
The Viewfactor Matrix

The viewfactor matrix is an $n \times n$ matrix which contains all the viewfactors from every face $i$ to every other face $j$ ($n$ is the number of faces). The viewfactor matrix is an $n \times n$ by $n$ matrix which contains all the viewfactors from every face $i$ to every other face $j$.

- If a face $j$ is occluded by a face $k$, the viewfactor for face $j$ is set to 0.
- If a face $j$ is occluded by a face $k$, the viewfactor for face $j$ is set to 0.
- If a face $j$ is occluded by a face $k$, the viewfactor for face $j$ is set to 0.
- If a face $j$ is occluded by a face $k$, the viewfactor for face $j$ is set to 0.

All of these tests help fill the viewfactor matrix.
Problems with our Occlusion Routine

Our routine does not handle partial occlusions.

Figure 9: Our occlusion subroutine will register this as an occlusion even though face k occludes very little of face j.

Erroneous Occlusion
Relating Radiosity and Net Radiant Flux

After we have solved the system of equations, we have a matrix

\[ \sum_B f_i = \mathbf{I} \]

\[ \mathbf{I} - \mathbf{B} = \mathbf{b} \]

To get the net radiant flux, all we need to do is this:

1. Temperature distribution of the mold.
2. Temperature (remember, we are still trying to get the final temperature of each face).
3. After we have solved the system of equations, we have a matrix

Relating Radiosity and Net Radiant Flux
Relating Net Radiant Flux and Temperature

Once we have the net radiant flux, we get temperature by:

\[ q_{\text{rad}} h + q_{\text{conv}} h + q_{\text{cond}} h = \frac{\Delta T}{\Delta t} \]

Using this program for \( q_{\text{rad}} \), we can now get an accurate temperature distribution for the mold.
To test the program, we ran the program on several spherical meshes and looked at how each sphere interacted with the others. Thus an area that looks colder means that that area has less net radiant flux. This means that there is more incident radiation on that area.

Here are some pictures.
So far we have worked on this code independent of Telluride. We now need to:

• Integrate the program into Telluride. John will probably do this after the workshop.
• Parallelize the code. This will also probably happen after the workshop.

Future Work