## مقدمة

هدفنا من هذا البحث هو الحصول على صورة واقعية لمشاهد حقيقية أو تخيلية، وبما أنّ تمثيل الصورة يتطلب تحديد لون كل pixel من شاشة العرض. لذلك سنبدأ بتعريف اللون ومن ثمّ ننطلق إلى كيفية حسابه.

## تعريف اللون

الضوء هو موجة كهرطيسيّة ويتم تحديد لونه عن طريق إدراك العين لتوزع الطاقة الطيفي للموجة الضوئية .

تحديد اللون يتمثّل بتحديد المركبات *(R,G,B)* لكل *pixel*. والخطوة التي تسبقها هي تحديد هذه المركبات لجميع نقاط المشهد المراد تصويره.

وهذا ما تؤمنه *radiosity method* .

# *RADIOSITY*

- *radiosity* لسطح هي معدل الطاقة المغادرة عن هذا السطح في واحدة الزمن في واحدة السطح (*energy per unit time per unit area*) ، حيث تتضمن الطاقة الصادرة عن هذا السطح نفسه إضافةً إلى الطاقة المنعكسة من بقية السطوح إلى هذا السطح.

- إن هذا المفهوم يمكن أن يستخدم لتمثيل ومحاكاة انتقال الطاقة الإشعاعية بين سطوح المشهد المرسوم في صورة على الحاسب، على اعتبار أن جميع السطوح المرسومة هي منابع ضوء ثانوية إضافة إلى المنابع الضوئية الأساسية في الصورة الممثلة في فضاء ثلاثي البعد.

# *RADIOSITY* معادلات

$$\begin{pmatrix}\text{outgoing}\\ \text{power}\\ \text{of elem } i\end{pmatrix}=\begin{pmatrix}\text{power}\\ \text{emitted}\\ \text{by elem } i\end{pmatrix}+\begin{pmatrix}\text{power}\\ \text{reflected}\\ \text{by elem } i\end{pmatrix}$$

$$\begin{pmatrix}\text{outgoing}\\ \text{power}\\ \text{of elem } i\end{pmatrix}=\begin{pmatrix}\text{power}\\ \text{emitted}\\ \text{by elem } i\end{pmatrix}+\begin{pmatrix}\text{reflectance}\\ \text{of elem } i\end{pmatrix}\times\sum_{\text{elem } j}\begin{pmatrix}\text{fraction of power}\\ \text{leaving elem } j \text{ that}\\ \text{arrives at elem } i\end{pmatrix}\times\begin{pmatrix}\text{outgoing}\\ \text{power}\\ \text{of elem } j\end{pmatrix}$$

Let

$A_i$ = area of element $i$ (computable)

$e_i$ = radiant emitted flux density of element $i$ (given)

$\rho_i$ = reflectance of element $i$ (given)

$b_i$ = radiosity of element $i$ (unknown)

$F_{ij}$ = form factor from $i$ to $j$ = fraction of power leaving $i$ that arrives at $j$

يمكن كتابة المعادلة السابقة بالشكل:

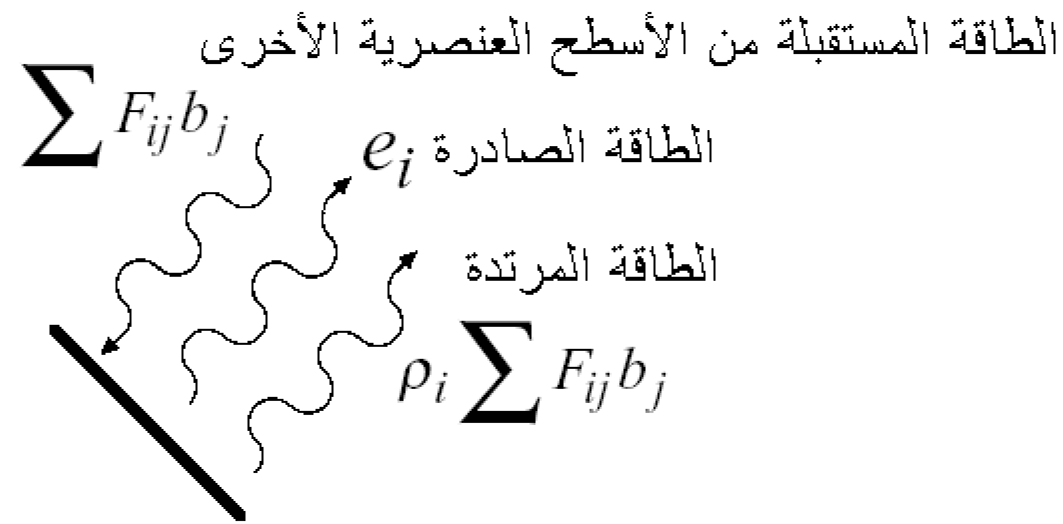
$$A_i b_i = A_i e_i + \rho_i \sum_{j=1}^{n} F_{ji} A_j b_j$$

بالتقسيم على $A_i$ :

$$b_i = e_i + \rho_i \sum_{j=1}^{n} F_{ji} \frac{A_j}{A_i} b_j$$

وبما أن: $F_{ji}(A_j / A_i) = F_{ij}$ ، بالتالي:

$$b_i = e_i + \rho_i \sum_{j=1}^{n} F_{ij} b_j$$

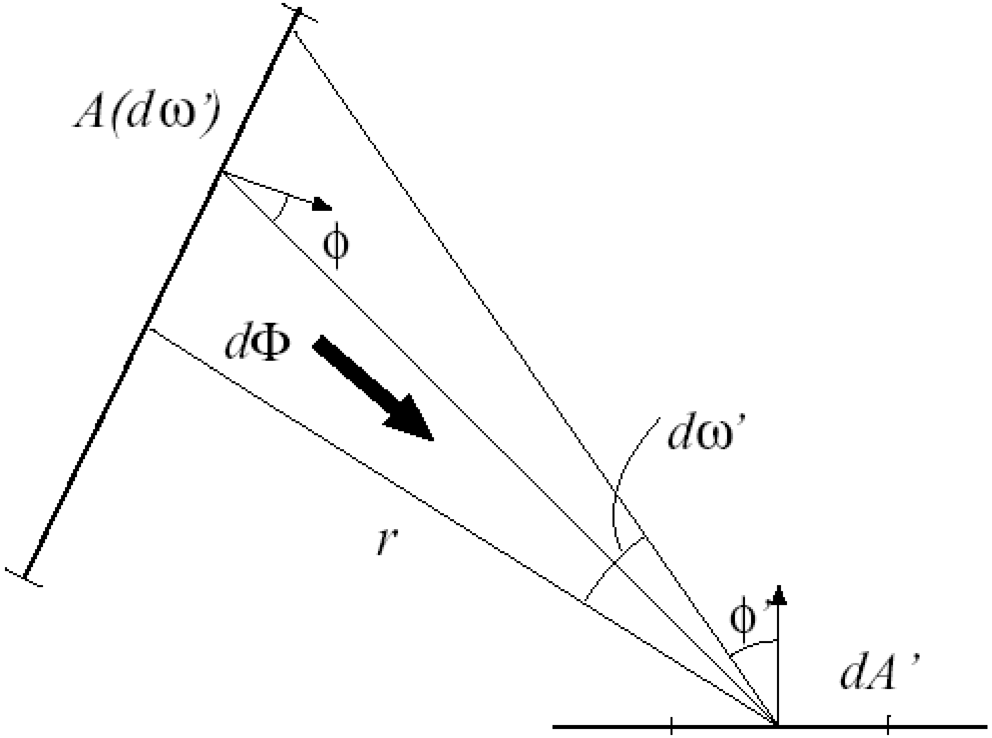


# *FORM FACTOR*

لنعتبر سطح عنصري تفاضلي $dA$ للسطح $A$ ، الطاقة الكلّيّة المغادرة للسطح $dA$ في واحدة الزمن هي $B\cdot dA$ . بينما التدفق ضمن الزاوية الصلبة $d\omega$ هو $d\Phi = I\cdot dA\cdot\cos\phi\cdot d\omega$ حيث $\phi$ هي الزاوية بين الناظم على السطح والاتجاه المعتبر.
الطاقة الكلية هي تكامل لأجزاء الطاقة المتدفقة عبر السطح والقادمة من جميع الجهات:

$$B = \frac{1}{dA}\int^{2\pi}\frac{d\Phi}{d\omega}\cdot d\omega = \int^{2\pi} I\cdot\cos\phi\cdot d\omega = I\int_{\theta=0}^{2\pi}\int_{\phi=0}^{\pi/2}\cos\phi\cdot\sin\phi\cdot d\theta d\phi = I\cdot\pi$$

حيث أن $d\omega = \sin\phi\cdot d\theta d\phi$




الطاقة المتبادلة بين سطحين تفاضليين إذا كانا مرئيين بالنسبة لبعضهما:

$$d\Phi = I \cdot \frac{dA \cdot \cos\phi \cdot dA' \cdot \cos\phi'}{r^2}$$

حساب الطاقة المتبادلة من سطح محدود $\Delta A_j$ إلى آخر $\Delta A_i$ في واحدة الزمن يتم تقسيم كلا السطحين إلى عناصر متناهية في الصغر وطاقتها يتم جمعها أو مكاملتها:

$$\Delta\Phi_{ji} = \int\limits_{\Delta A_i}\int\limits_{\Delta A_j} B_j \cdot H_{ij} \cdot \frac{dA_i \cdot \cos\phi_i \cdot dA_j \cdot \cos\phi_j}{\pi \cdot r^2}$$

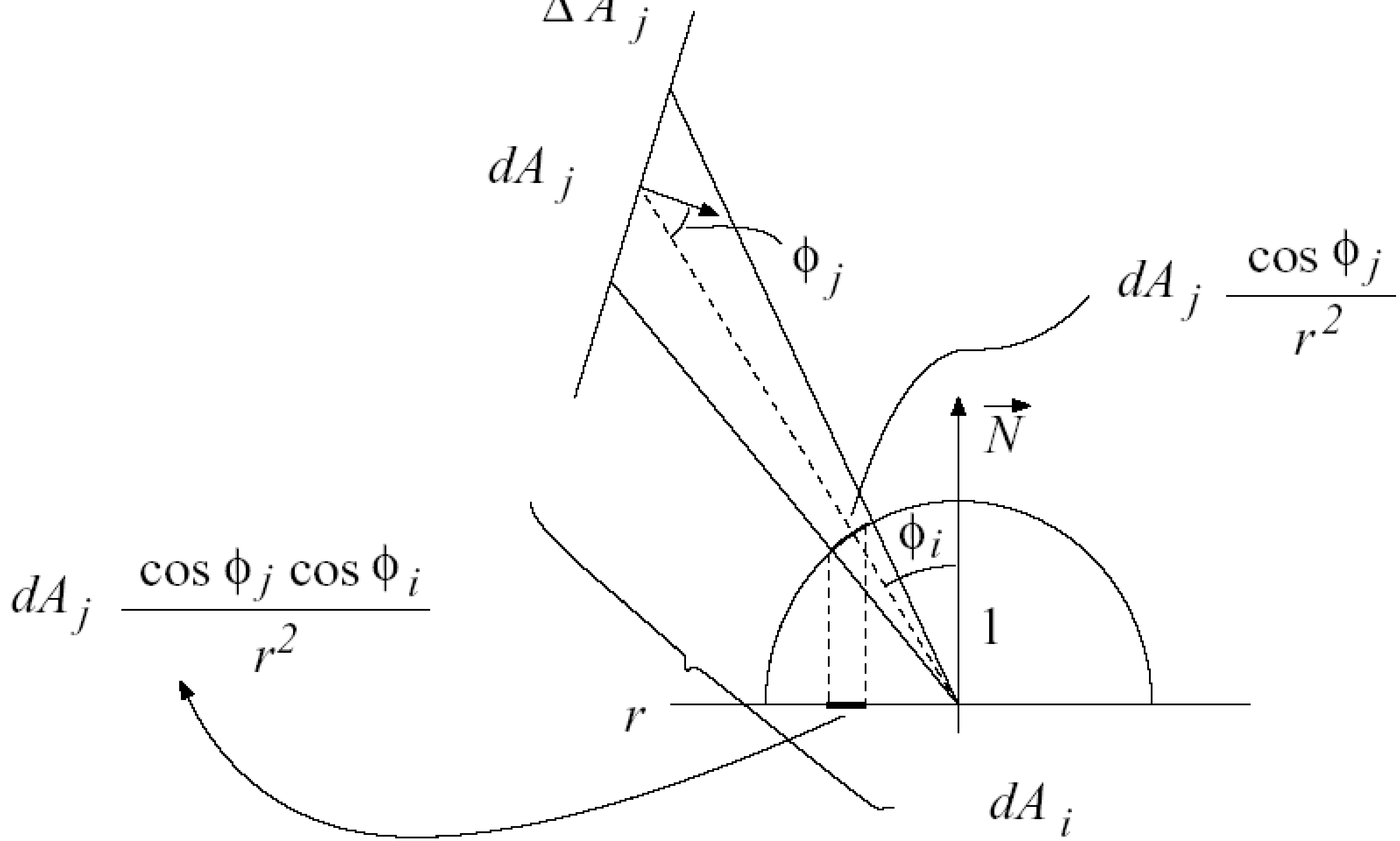
بالتعريف إن *form factor* $F_{ji}$ هو نسبة هذه الطاقة إلى الطاقة الكلية المغادرة للسطح $j$ $(B_j \cdot \Delta A_j)$

$$F_{ji} = \frac{1}{\Delta A_j} \cdot \int\limits_{\Delta A_i}\int\limits_{\Delta A_j} H_{ij} \cdot \frac{dA_i \cdot \cos\phi_i \cdot dA_j \cdot \cos\phi_j}{\pi \cdot r^2}$$

$$B_i = E_i + \rho_i \cdot \sum_j B_j \cdot F_{ij}$$

$$\begin{bmatrix} (1-\rho_1 F_{11}) & -\rho_1 F_{12} & \cdots & -\rho_1 F_{1N} \\ -\rho_2 F_{21} & (1-\rho_2 F_{22}) & \cdots & -\rho_2 F_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ -\rho_N F_{N1} & -\rho_N F_{N2} & \cdots & (1-\rho_N F_{NN}) \end{bmatrix} \cdot \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_N \end{bmatrix}$$

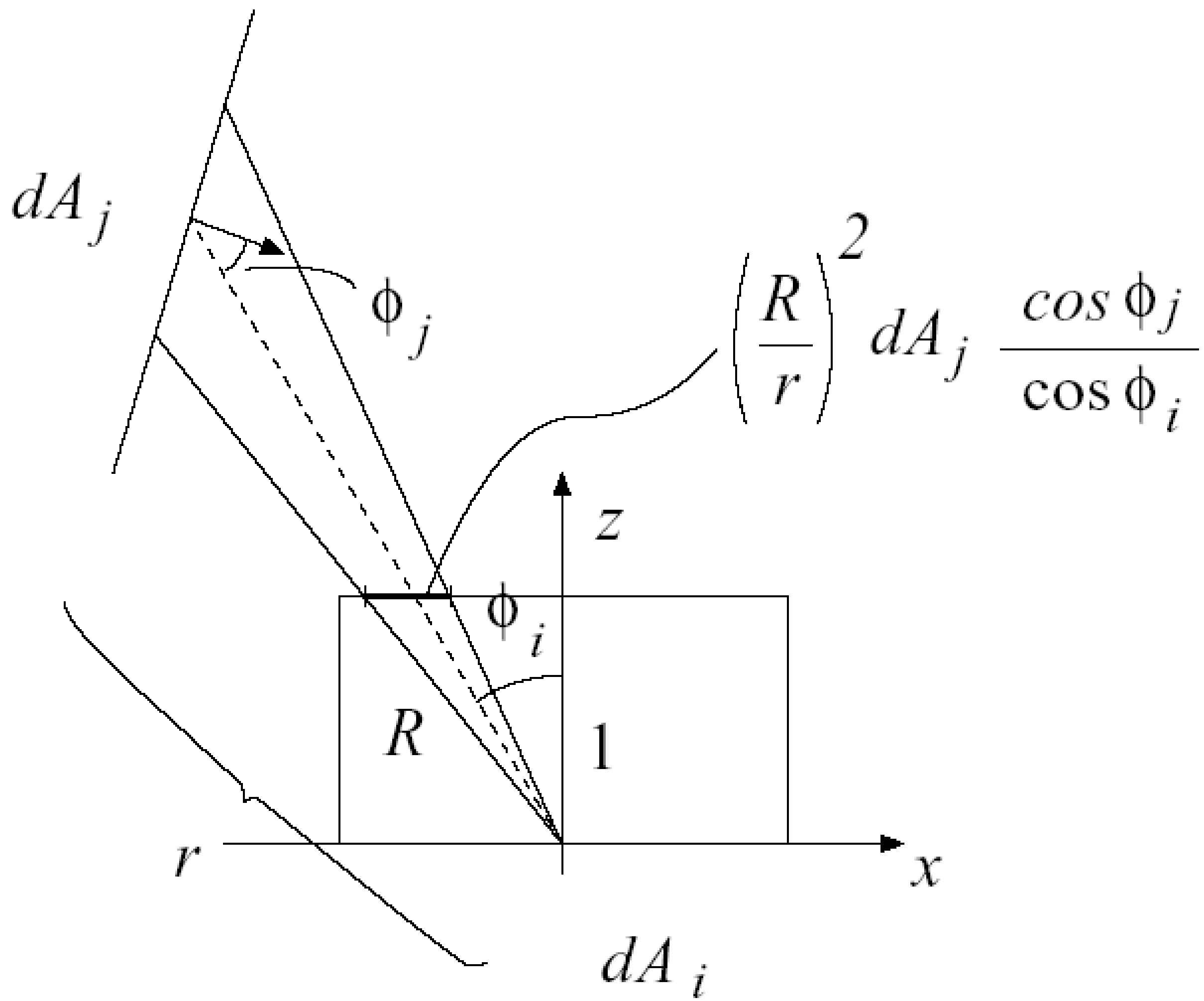
# *HEMISPHERE ANALOGUE*



$$d_i F_{ij} = \int_{\Delta A_j} H_{ij} \cdot \frac{\cos\phi_i \cdot \cos\phi_j}{\pi \cdot r^2} \cdot dA_j$$

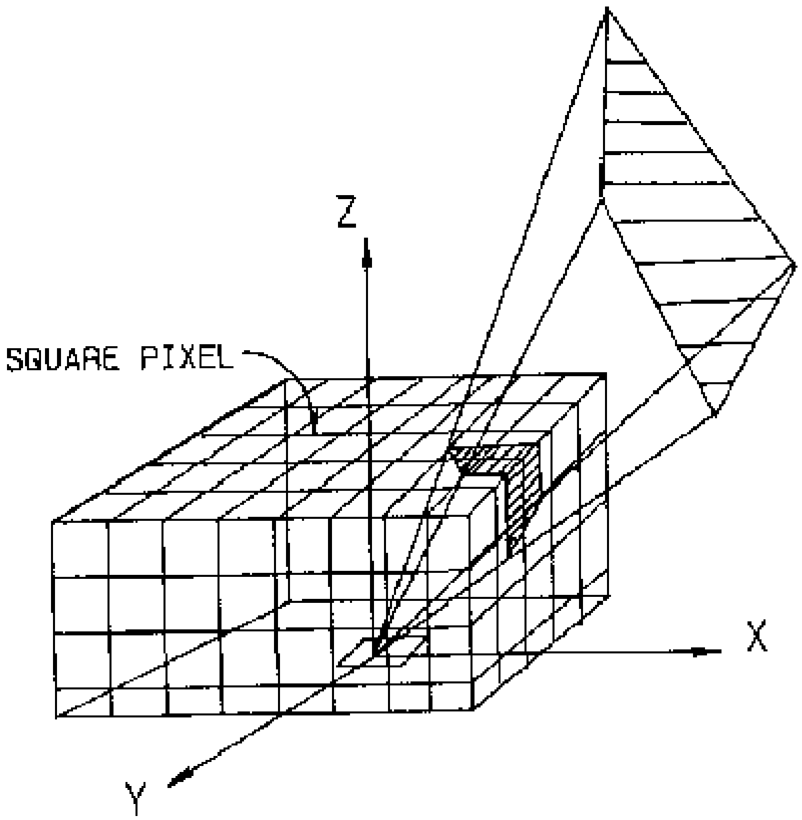
$$F_{ij} = \frac{1}{\Delta A_i} \int_{\Delta A_i} d_i F_{ij} \cdot dA_i \approx d_i F_{ij} = \int_{\Delta A_j} H_{ij} \cdot \frac{\cos\phi_i \cdot \cos\phi_j}{\pi \cdot r^2} \cdot dA_j$$

# HEMICUBE ANALOGUE




$$T\left(dA_j\right) = H_{ij} \cdot \left(\frac{R}{r}\right)^2 \cdot dA_j \cdot \frac{\cos\phi_j}{\cos\phi_i} = H_{ij} \cdot \frac{dA_j \cdot \cos\phi_i \cdot \cos\phi_j}{\pi \cdot r^2} \cdot \frac{\pi}{\cos^4\phi_i}$$

حيث $R = 1/\cos\phi_i$




$$\omega_z(x,y) = \frac{\cos^4 \phi_i}{\pi} = \frac{1}{\pi\left(x^2 + y^2 + 1\right)^2}$$

$$\omega_y(x,z) = \frac{\cos^4 \phi_i}{\pi} = \frac{z}{\pi\left(x^2 + z^2 + 1\right)^2}$$

$$\omega_x(y,z) = \frac{\cos^4 \phi_i}{\pi} = \frac{z}{\pi\left(y^2 + z^2 + 1\right)^2}$$

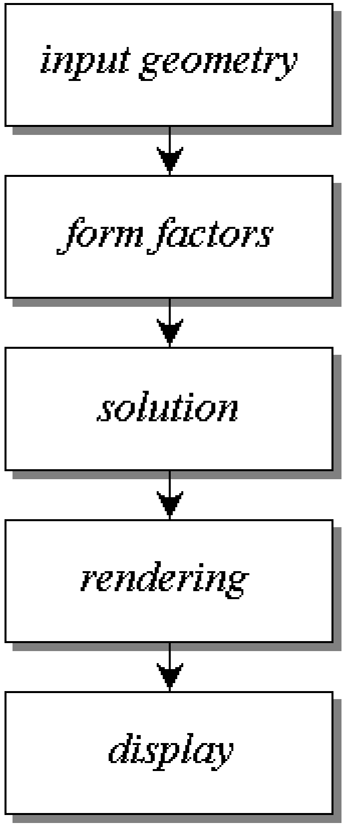
$$d_i F_{ij} = \int_{\Delta Aj} T\left(dA_j\right) \cdot \frac{\cos^4 \phi_i}{\pi}$$

$$d_i F_{ij}^{top} = \int_{T(\Delta Aj)} H_{ij}(x, y) \cdot \frac{1}{\pi\left(x^2 + y^2 + 1\right)^2} \cdot dxdy \approx$$

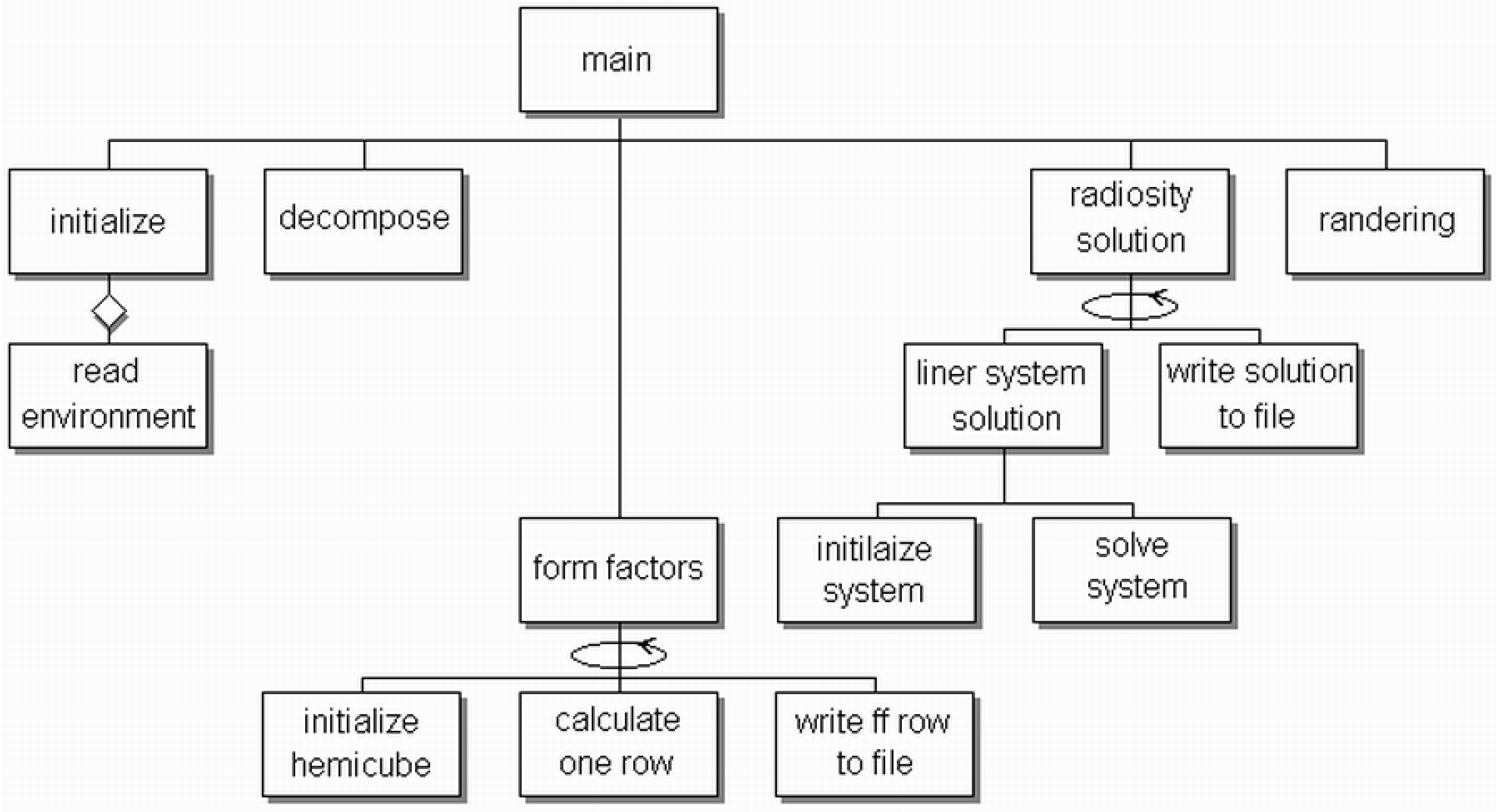
$$\sum_{x=-P/2}^{P/2-1} \sum_{y=-P/2}^{P/2-1} H_{ij}(X, Y) \cdot \omega_z(X, Y) \cdot \frac{1}{P^2}$$

# *PROGRAM SUMMARY*

## *PROGRAM FLOW*



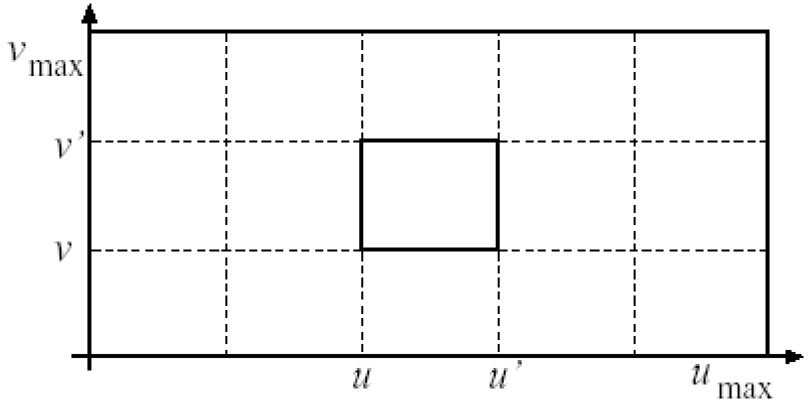
# *STRUCTURE CHART*



# *MAIN ALGORITHMS:*

## *1. Environment decomposition:*

*Given* $0 \leq u \leq u_{\max}$ *and* $0 \leq v \leq v_{\max}$*:*




```
DecomposeQuad (n, m)
    S = {}
    u[ i ] = 0 ;
    for i = 1 to n do
        u[ i + 1 ] = uMax * I / n ;
        v[ j ] = 0 ;
        for j = 1 to m do
            v[ j + 1 ] = vMax * j / m ;
            add the quadrilateral { ( u[ i ], v[ j ] ),
                                    ( u[ i + 1 ], v[ j ] ),
                                    ( u[ i + 1 ], v[ j + 1 ] ),
                                    ( u[ i ], v[ j + 1 ] ) } to S ;
            v[ j ] = v[ j + 1 ] ;
        endfor
        u[ i ] = u[ i + 1 ] ;
    endfor
    return S ;
end
```

## *2. Form factor calculation:*

```
for i = 1 to patchCount do
   for j = 1 to N do F[ i ][ j ] = 0;
for i = 1 to N do
   place hemicube on patch i ;
   project environment onto hemicube ;
   for k = 0 to pixelCount do
      if ( pixel[ k ] > 0) then
         F[ i ][ pixel[ k ] ] += deltaFormFactor [ k ] ;
   endfor
endfor
```

*3. Radiosity solution:*

```
while ( not converged ) do
   limit = smallest floating point value ;
   for i = 1 to patchCount do
      previous = radiosity [ i ] ;
      radiosity[ i ] = emission[ i ] ;
      for j = 1 to patchCount do
         radiosity[ i ] +=coefficient[ i ][ j ] * radiosity[ j ] ;
         limit =
            max((radiosity[i] - previous)/radiosity[i]), limit ) ;
      endfor
   if ( limit < epsilon )
      solution converged ;
endwhile
```