8. [5 points] Radiosity is best for: (Answer by circling one or more of i., ii., iii. as appropriate.)

i. Tracking the flow of ambient light throughout a scene.

ii. Tracking the flow of diffusely reflected light throughout a scene.

iii. Tracking the flow of specularly reflected light throughout a scene.

9. [15 points] In radiosity, the calculation of form factors $F_{i,j}$ depends partly on computing the **visibility** between one patch P_i and another patch P_j. Describe two methods for computing the visibility between a given pair of patches in a scene. (Mathematical formulas are not required, but be sure to explain all of the algorithmic elements.)

10. [15 points] Describe how depth-of-field should be implemented for ray tracing.

8. [30 points] Several kinds of bounding volumes can be used to enclose objects to help prune intersection testing, including bounding spheres, AABB's, OBB's and k-DOP's. Explain what AABB's and OBB's and k-DOP's are, and their important properties.

10. [20+20 points] The course discussed three methods to numerically solve the Radiosity Equation $\mathbf{B} = \mathbf{E} + M\mathbf{B}$: the Jacobi method, the Gauss-Seidel method, and the Shooting (Southwell, of Progressive Refinement) method (in order of increasing sophistication). Recall that M is an $n \times n$ matrix.

Write out the algorithm (as pseudo-code) for the Gauss-Seidel method.

Problem 10 continued: Now write out the algorithm for the Shooting method.

11. [10 points] (Radiosity) Explain what form factors $F_{i,j}$ are. Explain how ray tracing can be used to compute the form factors $F_{i,j}$. Do form factors need to be recomputed for each change in viewpoint?

12. [10 points] Describe how soft shadows can be implemented for ray tracing.

6. [20 points] This problem is about creating depth-of-field in ray tracing by jittering the eye position: this moves the eye position by choosing multiple jittered eye displacements for casting k^2 many rays and averaging the results. It is customary to use ***different*** jittered eye displacements for each pixel.

Explain what kinds of problems could happen if instead the ***same*** jittered eye displacements were used for every pixel. (That is, instead of choosing different k^2 many random eye displacements for each pixel, you used the same k^2 eye displacements for all the pixels.)

What kinds of undesirable visual effects might occur? Would these effects be more noticeable for items near the focal plane, or far from the focal plane?