

OpticStudio®

Programming OpticStudio
Self-Study: User-Defined NSC
Objects

UDO Advantages

- Advantages of defining object via a DLL, rather than POB, CAD, etc.
 - UDO generally traces faster, with higher numerical precision
 - Any mathematically describable shape can be created
 - DLL description is parametric: object dynamically regenerated as parameters change
 - Ideal for interactive design and optimization
- Single object can contain multiple complex curved surfaces, each with own defining function

UDO DLL Details

- DLL first called to create list of triangular facets approximating shape
- OpticStudio traces rays non-sequentially
- If ray hits a facet, DLL called to determine exact intercept and normal vector
 - Implemented using an iterative routine (sample routine provided in all sample UDOs)
 - If object has multiple curves, each can have own iteration routine
- These are the only purposes the DLL serves
 - OpticStudio handles NSC raytracing, object nesting, refraction, reflection, diffraction, optical path length, scattering, etc.
 - Completely separates object definition from optical physics

DLL Functions

- Values are passed between OpticStudio and DLL with a data array
 - Array contains all parameter data from NSC editor and ray data
- DLL has two functions
 - UserParamNames(): returns parameter and face group names
 - UserObjectDefinition(): contains object definition

UserParamNames Function

- Data value passed to DLL
 - Data <= 0: OpticStudio wants name of particular face (i.e. -1 => face 1)
 - Data > 0: OpticStudio wants name of parameter names

UserObjectDefinition Function

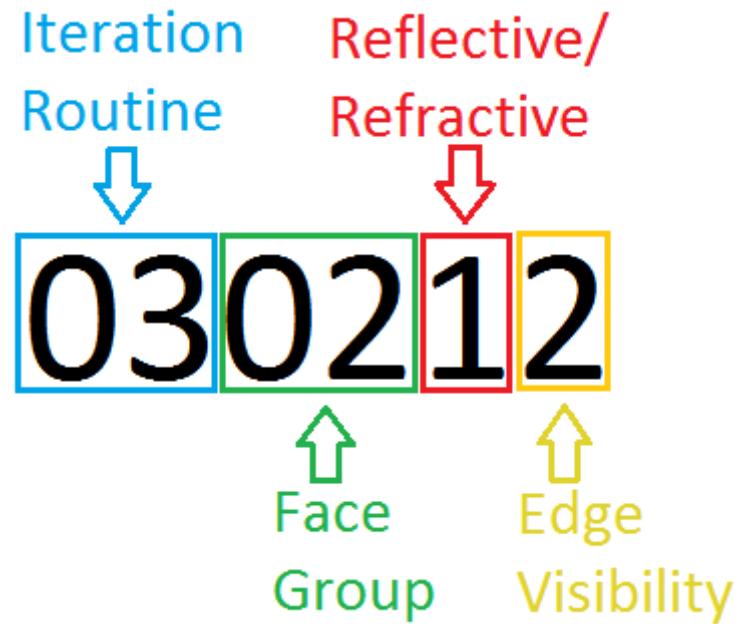
- Data[1] indicates what DLL should compute
 - Data[1] = 0
 - Place total number of triangular facets in Data[10]
 - Indicate if object is volume or shell in Data[11]
 - Indicate if object is diffractive in Data[12]
 - Data[1] = 1: list of triangles that compose tessellation
 - triangle info placed in tri_list array (more soon)
 - Data[1] = 2: ray intercept and normal vector (after iteration)
 - OpticStudio provides ray coordinates and direction cosines on tessellation; DLL must iterate to exact surface
 - DLL returns path length to exact surface and normal vector at surface intercept point
 - Data[1] = 3: polarization data
 - Data[1] = 4: default/safe parameter values

UserObjectDefinition Function Cont'd.

- Data[100] indicates the number of data parameters passed to the DLL
- Data[100+] are the values in parameter cells
 - Data[101] = parameter 1, Data[102] = parameter 2, etc.

Tri_List Array

- DLL calculates 10 values for each triangle
 - First 9 are three sets of X,Y,Z coordinates that define triangle
 - 10th value is 6 digit integer that indicates optical properties



Defining Triangle Properties

- The "visible" flag is in the 1's place.
 - If all 3 sides of a triangle are visible, use 0.
 - If the side from point 1 to point 2 is invisible, use 1.
 - If the side from point 2 to point 3 is invisible, use 2.
 - If the side from point 3 to point 1 is invisible, use 4.
 - Add these flags if more than one side is invisible. For example, if all 3 sides are invisible, use 7. (Note invisible simply means the side is not drawn on lens renderings, it has nothing to do with ray tracing. It is provided to make on-screen rendering as informative as possible.)
- The "reflective" flag is in the 10's place. This allows a triangular facet area to be defined as reflective or refractive.
 - Use 0 for refractive .
 - Use 1 for reflective.

Defining Triangle Properties

- The "coat/scatter group" is a 2 digit integer in the 1,000 and 100's place.
 - This allows thin-film coating and optical scattering functions to be defined for each triangular facet area.
- The "exact group" flag is a 2 digit integer in the 100,000 and 10,000's place.
 - The exact group number indicates which set of exact formulas is to be used to iterate to the actual object surface. The exact flags are used in the intercept portion of the code discussed later.