10.10 Constructive Solid Geometric (CSG)

- Objects are created as combination of simple primitives, such as rectangular blocks or cylinders, via (regularized) Boolean set operations ($\cup$* (union), $\cap$* (intersection), $-*$ (difference))
10.10 Constructive Solid Geometric (CSG)

More complicated examples:

- The construction process is usually recorded in a CSG tree.
- A CSG tree is an ordered binary tree: non-terminal nodes are operators ($\cup^*, \cap^*, -^*$), terminal nodes are primitives.
10.10 Constructive Solid Geometric (CSG)

An example of 2D CSG object:
Non-terminal nodes are called *composites*;

Terminal nodes are called *primitives*.
10.10.1 Regular Sets & Regular Set Operations

- A set $X$ is a regular set if it equals the closure of its interior, i.e.,

$$X = Cl(Int(X))$$

where $Cl$ and $Int$ denote the closure and interior of a set, respectively.

**Example:** sets which are not regular

![Diagram of dangling face and edge]
10.10.1 Regular Sets & Regular Set Operations

- The regularization of a set $X$ is the closure of its interior, i.e.,
  \[ Re(X) = Cl(Int(X)) \]

  where $Re$ denotes the regularization of a set.

- Regularized set operators:
  \[ X < rop > Y = Cl(Int( X < op > Y)) \]

  "op": conventional set operator $\{\cup, \cap, \neg\}$
  "rop": regularized set operator $\{\cup^*, \cap^*, \neg^*\}$
Example:

The regularized set intersection of $A$ and $B$ below is empty but the conventional intersection of $A$ and $B$ is not.

![Diagram of 3D and 2D set intersections]
10.10.2 Data Structures

**Composite:**
- Name
- Operator
- L_T_PTR
- R_T_PTR

**Primitive:**
- Name
- Type
- Local_To_Global_Transformation
- Global_To_Local_Transformation
- Surface_PTRs
10.10.2 Data Structures

- To construct solids, position primitives in proper location, scale and rotate them to get the desired size and orientation in global coordinate system.

- E.g., to position a block in the global coordinate system, we need:
  - Reference point (R):
  - X direction coordinate (X):
  - Y direction coordinate (Y):
  - Z length:
10.10.3 Set Membership Classification

Divide-and-conquer is used in finding solutions to
The following membership classification problems:

(1) Given 2 regular sets $X$ (candidate set) and $S$ (reference set), find out if $X$ a member of $S$
(2) Given 2 regular sets $X$ and $S$, divide $X$ into parts that are inside, on the boundary of, and outside $S
10.10.3 Set Membership Classification

The membership classification function $M[., .]$ is defined as follows:

$$M[X, S] = (X_{inS}, X_{onS}, X_{outS})$$

where

- $S : reference set; \quad X : candidate set$
- $X_{inS} = X \quad (Int(S))$
- $X_{onS} = X \quad B(S)$
- $X_{outS} = X \quad C(S)$
10.10.3 Set Membership Classification

Classification process: \textit{divide-and-conquer}

If $S$ is not a primitive, classify $X$ with respect to the subtrees of $S$ and then "combine" the results to get the classification of $X$ with respect to $S$.

If $S$ is a primitive, the problem can not be decomposed further, a "primitive evaluator" is used.
10.10.3 Set Membership Classification

Procedure $M[X, S]$

if ( $S$ is a primitive ) then
    return Prim - $M[X, S]$
else
    combine($M[X, \text{LT}(S)]$, $M[X, \text{RT}(S)]$, Root($S$))

Example: $S = A <\text{rop}> B$

Combine($M[X, A]$, $M[X, B]$, $<\text{rop}>$)

Prim-$M[X, A]$    Prim-$M[X, B]$
10.10.3.1 Point Membership Classification

P: point        S: CSG object

Procedure \text{PMC}[P, S]
if ( S is a primitive ) then
  return \text{Prim-PMC}[P, S]
else
  combine(\text{PMC}[P, LT(S)], \text{PMC}[P, RT(S)], \text{Root}(S))

\text{Prim-PMC} classifies a point against a primitive.
This is the trivial part.
10.10.3.1 Point Membership Classification

"Combine" procedure is to determine whether a point is "in", "on", or "out" a CSG tree (object)

\[ S = A <\text{rop}> B \]

by combining the classification of the point w. r. t. A and B

Done by using a look-up table:
10.10.3.1 Point Membership Classification

Ambiguous cases: P is "on" A and "on" B in both cases

For example: \( S = A \cap^* B \)
- P is "on" \( S = A \cap^* B \) in case (a)
- P is "out" \( S = A \cap^* B \) in case (b)
10.10.3.1 Point Membership Classification

Remedy: using the *regular neighborhood method*

S is a regular set in $\mathbb{R}^3$ and $P$ is a point in $\mathbb{R}^3$. The regular neighborhood of $P$ with respect to $S$ with radius $r > 0$, denoted $N(P, S; r)$, is defined as follows:

$$N(P, S; r) \equiv B(P; r) \cap^* S$$

where $B(P, r)$ is the open ball of radius $r$ about $P$. 
10.10.3.1 Point Membership Classification

- Point $P$ is "inside" $S$ iff $N(P, S ; r) = \text{Cl} (B (P, r))$ for some $r > 0$.
- Point $P$ is "outside" $S$ iff $N(P, S ; r) = \emptyset$ for some $r > 0$.
- Point $P$ is "on" $S$ iff $N(P, S ; r) = \emptyset$ and $N(P, S ; r) \neq \text{Cl} (B (P, r))$ for any $r > 0$.

If $S = A \text{ <op> } B$ then $N(P, S ; r) \equiv N(P, A ; r) \text{ <op> } N(P, B ; r)$ where "op" is a regularized set operator.
10.10.3.1 Point Membership Classification

Use the following look-up tables to resolve ambiguous cases:

<table>
<thead>
<tr>
<th>$\cap^*$</th>
<th>Left</th>
<th>Right</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left</td>
<td>Left</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>Right</td>
<td>$\emptyset$</td>
<td>Right</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\cup^*$</th>
<th>Left</th>
<th>Right</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left</td>
<td>Left</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>Right</td>
<td>$\emptyset$</td>
<td>Right</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$-$</th>
<th>Left</th>
<th>Right</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left</td>
<td>$\emptyset$</td>
<td>Left</td>
</tr>
</tbody>
</table>

Example: consider the figures on page 15 again
(1) $S = A \cap^* B$

consider case (a) first. In this case,
$N(P, A ; r ) = \text{left}; \quad N(P, B ; r ) = \text{left}$

So, $N(P, S ; r ) = N(P, A ; r ) \cap^* N(P, B ; r ) = \text{left}.$

Hence, $P$ is "on" $S$.
10.10.3.2 Line Membership Classification

**L**: a line,  **S**: a CSG object

**Procedure** \( LMC[L, S] \)

- for each primitive \( P \) in \( S \) do
  - intersect \( L \) with \( P \)
  - enter point(s) of intersections into PList

- if PList contains more than one point then
  - order the points by sorting them along \( L \)
  - Add the endpoints of \( L \) into PList at the ends

- for each segment* in PList do
  - \( m \leftarrow \) midpoint of the segment
  - case \( PMC[m, S] \) of
    - "in": add segment to in-list
    - "on": add segment to on-list
    - "out": add segment to out-list

*segment*: a portion of \( L \) defined by two consecutive points of PList
10.10.3.2 Line Membership Classification

Example for LMC:

\[ S = A \cup^* B \cup^* C \cup^* D \]
\[ L = [P_1, P_2] \]
10.10.3.2 Line Membership Classification

Example for LMC:

L intersects $A$ at: $P_3, P_4$
L intersects $B$ at: null
L intersects $C$ at: $P_5, P_6$
L intersects $C$ at: $P_7, P_8$

After sorting, $PList = P_3, P_5, P_4, P_6, P_7, P_8$

After adding $P_1$ and $P_2$ into $PList$, we have $P_1, P_3, P_5, P_4, P_6, P_7, P_8, P_2$
10.10.3.2 Line Membership Classification

Example for LMC:

Mid-point of $[P_1, P_3]$ is "out" S
$[P_3, P_5]$ "in"
$[P_5, P_4]$ "in"
$[P_4, P_6]$ "in"
$[P_6, P_7]$ "out"
$[P_7, P_8]$ "in"
$[P_8, P_2]$ "out"

Hence, in-list = $[P_3, P_5], [P_5, P_4], [P_4, P_6], [P_7, P_8]$  
on-list = null  
out-list = $[P_1, P_3], [P_6, P_7], [P_8, P_2]$
10.10.3.2 Line Membership Classification

Example for LMC:

(If necessary, merge connecting segments in the same category into one segment)

After merging,

\[
\text{in-list} = [P_3, P_6], [P_7, P_8] \\
\text{on-list} = \text{null} \\
\text{out-list} = [P_1, P_3], [P_6, P_7], [P_8, P_2]
\]
10.10.4 Object Generation and Display - ray-tracing based

Ray Casting Model
For a focal point (view point) and a rectangular pixel array (screen), generate a ray through each pixel and find the first surface the ray intersects.
10.10.4 Object Generation and Display - ray-tracing based

For each ray, the following information will be returned:

Ray parameters: \( t[1], t[2], \ldots, t[n] \)
Surface pointers: \( S[1], S[2], \ldots, S[n] \)

where \( n \) is the number of ray-solid intersections. The ordered list of ray parameters denotes the enter-exit points. The list of surface pointers are pointers to the surfaces through which the ray passes.
10.10.4.2 How to use the information

- Wire Frame Drawings
- Shaded Images
- Volume Computation
10.10.4.3 Ray Casting Algorithm

RAYCAST uses three coordinate systems:
- scene (global) coordinate system
- primitives’ local coordinate systems
- screen coordinate system

Given a ray originating in the screen coordinate system, RAYCAST transforms it into the local coordinate systems of the primitives via the scene (global) coordinate system to find the ray-solid intersection points.
RAYCAST does not find the points where rays enter or exit a solid. Rather, it finds the ray parameters that designate those points. Solids and surface equations are never explicitly transformed. Primitives’ global to local transforms are only used to transform rays. The effect of scaling, rotating, and translating a solid is achieved by scaling, rotating and translating the rays.
10.10.4.3 Ray Casting Algorithm

**In-Out Classification**
Given a ray and a solid (a CSG tree), RAYCAST classifies the ray with respect to the solid and returns the classification (information describing the ray-solid intersection) to the caller. RAYCAST starts at the top of the CSG tree, recursively descends to the bottom, classifies the ray with respect to the primitives, and then work its way up by recursively combining the classifications of the left and right subtrees.
Example:
10.10.4.3 Ray Casting Algorithm

(i) Intersecting rays with primitives:

(ii) Combining left and right classifications:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Left</th>
<th>Right</th>
<th>Composite</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\cap$</td>
<td>IN</td>
<td>IN</td>
<td>IN</td>
</tr>
<tr>
<td></td>
<td>IN</td>
<td>OUT</td>
<td>IN</td>
</tr>
<tr>
<td></td>
<td>OUT</td>
<td>IN</td>
<td>IN</td>
</tr>
<tr>
<td></td>
<td>OUT</td>
<td>OUT</td>
<td>OUT</td>
</tr>
<tr>
<td>$\cup$</td>
<td>IN</td>
<td>IN</td>
<td>IN</td>
</tr>
<tr>
<td></td>
<td>IN</td>
<td>OUT</td>
<td>OUT</td>
</tr>
<tr>
<td></td>
<td>OUT</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td></td>
<td>OUT</td>
<td>OUT</td>
<td>OUT</td>
</tr>
<tr>
<td>$\setminus$</td>
<td>IN</td>
<td>IN</td>
<td>OUT</td>
</tr>
</tbody>
</table>
Example: solid lines represent ray segments inside the solid and dashed lines represent ray segments outside the solid.

\[ S = L \cup^* R \]
End of 10.10