Euler operator

Human-Centered CAD Lab.

Euler operator

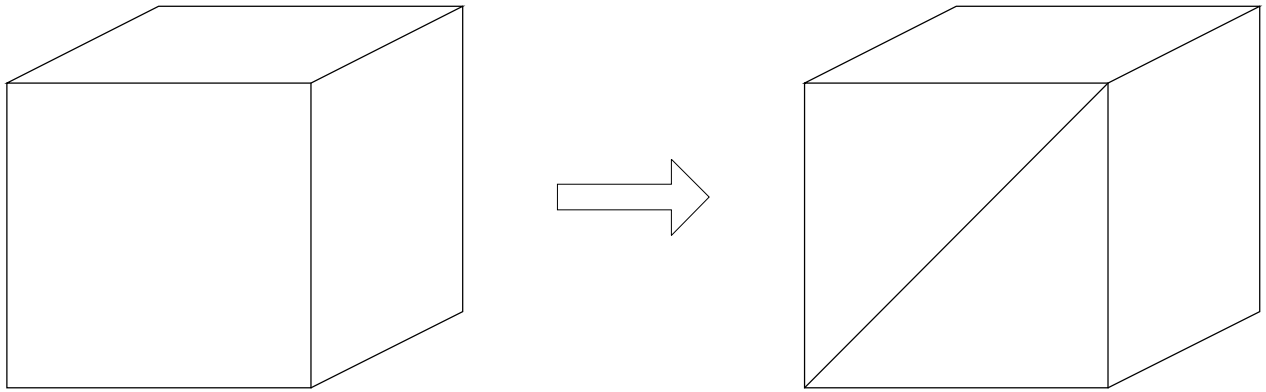
- Facilitates the process of adding or deleting faces, edges, vertices during model modification
- Guarantees consistent topology data
- Shows systematically how a modeling command is implemented
- Data structure can be separated from the modeling program

Euler operator – cont'

- Example Euler operators accepted by Stroud, Hillyard, Braid
  - MEV(Make Edge Vertex) & KEV(Kill Edge Vertex)
  - MEF(Make Edge Face) & KEF(Kill Edge Face)
  - MEKH(Make Edge Kill Hole) & KEMH(Kill Edge Make Hole)
  - etc.

Euler operator – cont’

- A change in a topology entity always accompany changes in other topology entities



  - Changes in faces caused by an addition of edge

Euler operator – cont'

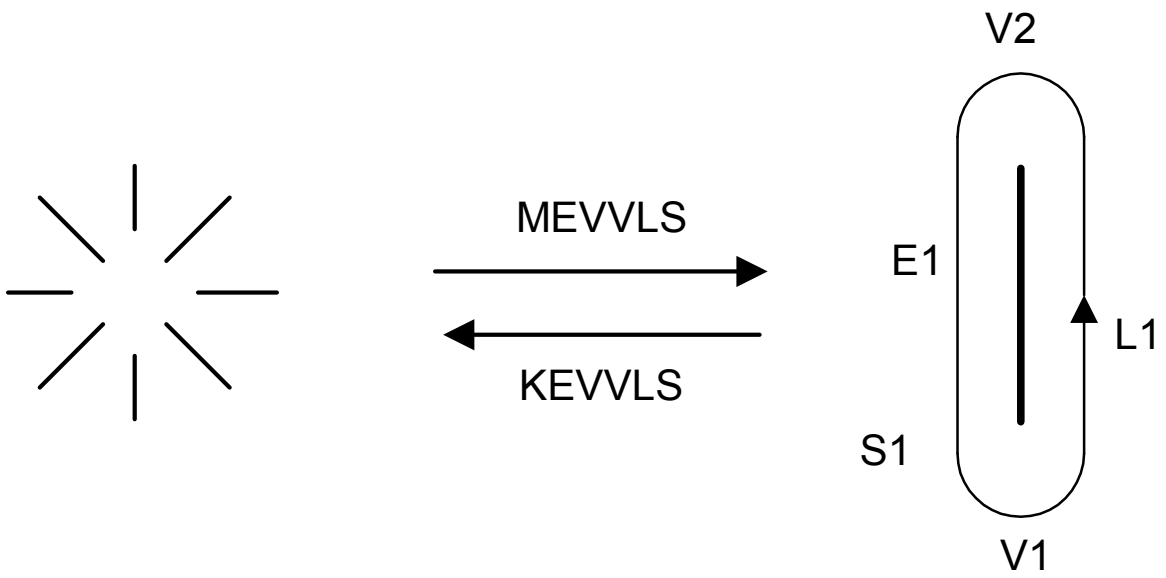
- Better to manipulate a group of topology entities as a group
  → Euler operator

- 5 independent variables in Euler equations
  → Minimum of 5 Euler operators are needed

- Different set of Euler operators are used in different modeling systems

Euler operators used in SNUMOD

- 1. MEVVLS and KEVVLS
  - make (kill) an edge, two vertices, a peripheral loop, and a shell

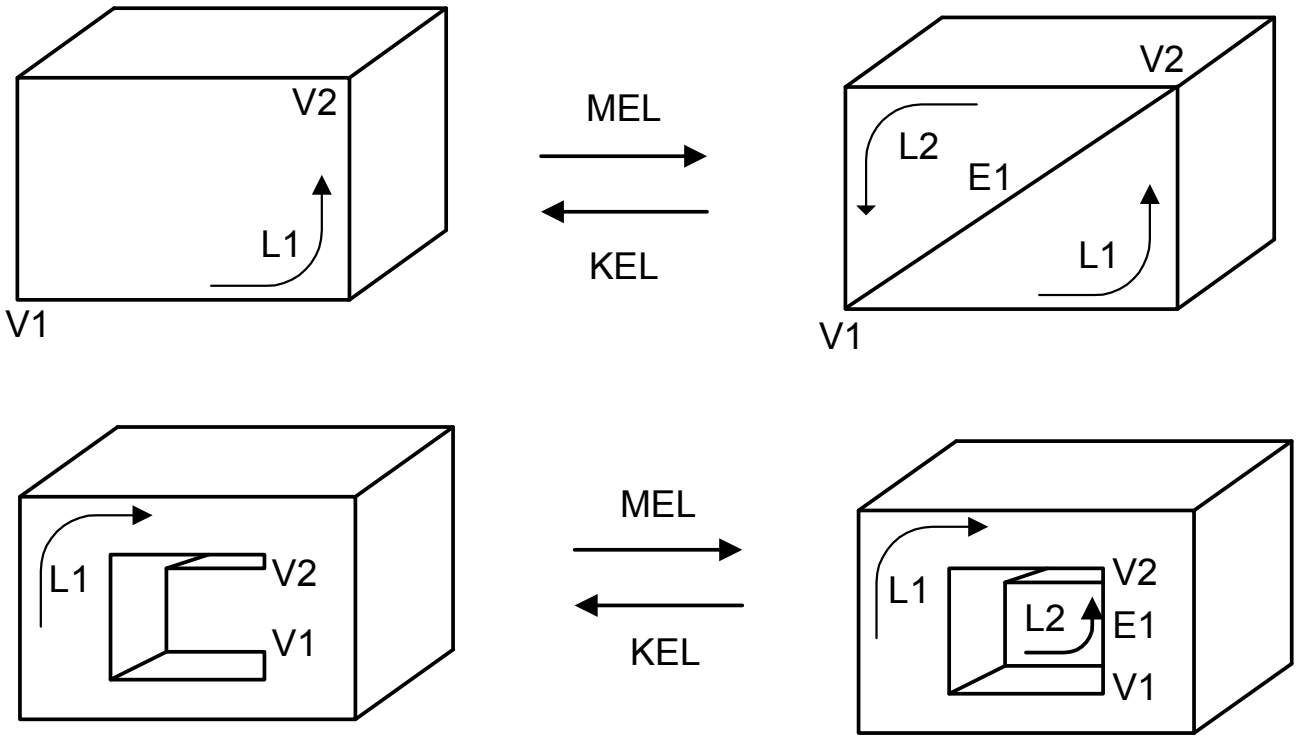


Euler operators used in SNUMOD – cont'

- Allocates memory space for a shell
- Creates two vertices, an edge between the vertices, and a peripheral loop
- A face with empty geometry is created
  - MEVVLS (B(solid), &E1, &V1, &V2, &L1, &S1, X1, Y1, Z1, X2, Y2, Z2)
  - KEVVLS (B, E1, V1, V2, L1, S1, &X1, &Y1, &Z1, &X2, &Y2, &Z2)
    - Inverse operation is used in implementing 'Undo' operation

Euler operators used in SNUMOD – cont'

- 2. MEL and KEL
  - make(kill) an edge and a peripheral loop

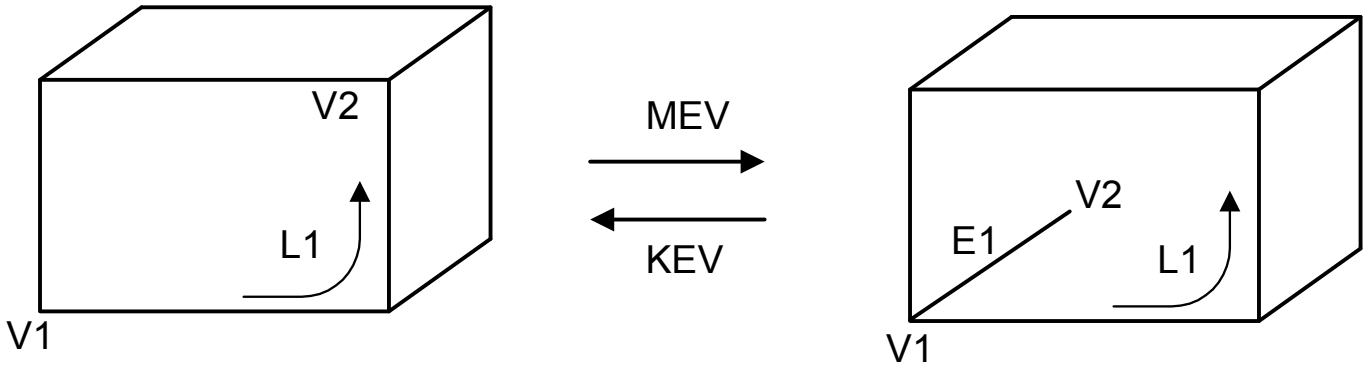


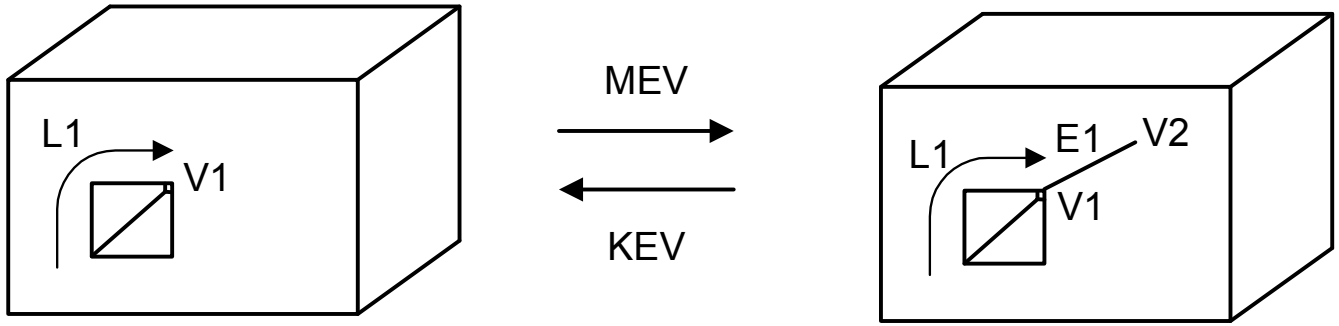
Euler operators used in SNUMOD – cont'

- Divide original loop L1,into two new loops L1 and L2.
- Update edge connections at V1 and V2, and edge-loop relations
  - MEL (B, L1, V1, V2, &E1, &L2)
  - KEL (B, &L1, &V1, &V2, E1, L2)

Euler operators used in SNUMOD – cont'

- 3. MEV and KEV (make (kill) an edge and a vertex)





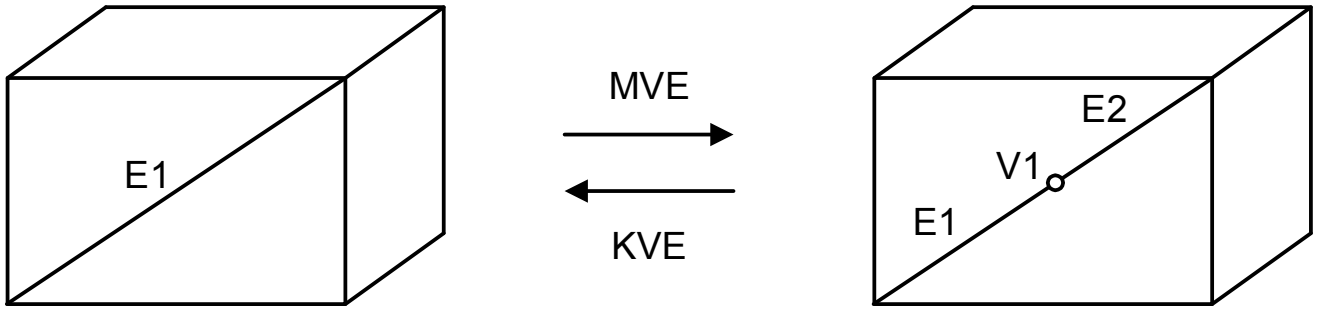
Euler operators used in SNUMOD – cont'

- Create edge E1 from vertex V1 located on loop L1
- Add the edge E1 to L1
- New vertex V2 is created at (X, Y, Z)
- Edge connection at V1 is updated (E1 is included in L1)

  - MEV (B, L1, V1, &E1, &V2, X, Y, Z)
  - KEV (B, &L1, &V1, E1, V2, &X, &Y, &Z)

Euler operators used in SNUMOD – cont'

- 4. MVE and KVE (make (kill) a vertex and an edge)



  - Split edge E1 by adding V1 on E1 at (X, Y, Z)
  - E1 is replaced by two new edges E1, E2

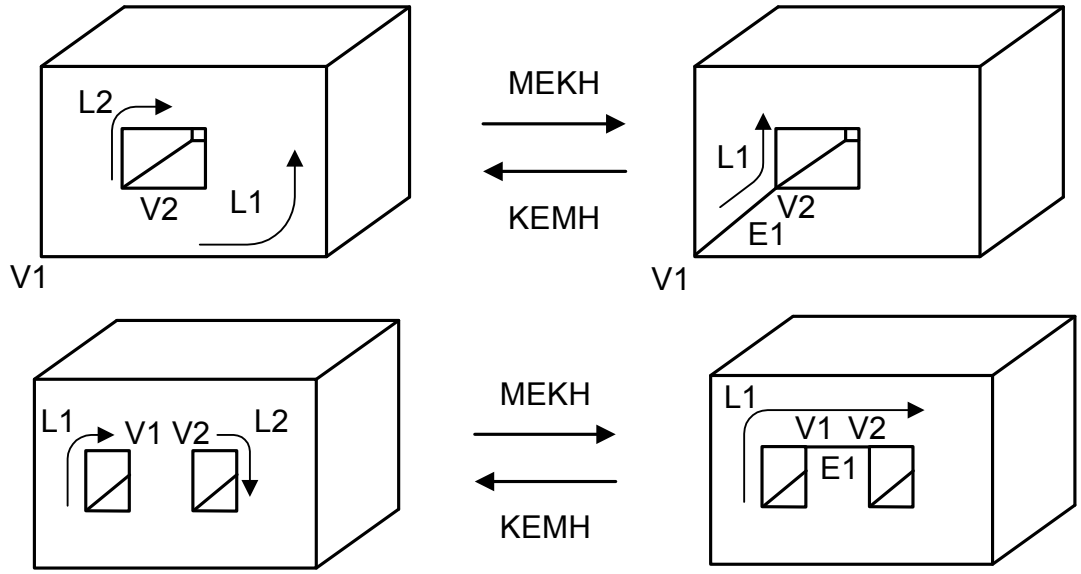
  - MVE (B, E1, &V1, &E2, X, Y, Z)
  - KVE (B, &E1, V1, E2, &X, &Y, &Z)

Euler operators used in SNUMOD – cont'

```
MVE ( B, E1, V1, E2, x, y, z )
    Body            *B ;
    Edge            *E1 ;
    Vertex          **V1 ;
    Edge            **E2 ;
    double          x, y, z ;
    {
      ( *V1 ) = malloc ( sizeof ( Vertex ) ) ;
      ( *E2 ) = malloc ( sizeof ( Edge ) ) ;
      ( *E2 )->tail_vertex = *V1 ;
      ( *E2 )->head_vertex = E1->head_vertex ;
      ( *E2 )->right_leg = E1 ;
      ( *E2 )->left_leg = E1 ;
      ( *E2 )->right_arm = E1->right_arm ;
      ( *E2 )->left_arm = E1->left_arm ;
      ( *E2 )->right_loop = E1->right_loop ;
      ( *E2 )->left_loop = E1->left_loop ;
      ( *V1 )->edge = *E2 ;
      ( *V1 )->point.x = x ; ( *V1 )->point.y = y ; ( *V1 )->point.z = z ;
      E1->right_arm = E1->left_arm = *E2 ;
      E1->head_vertex = *V1 ;
    }
```

Euler operators used in SNUMOD – cont'

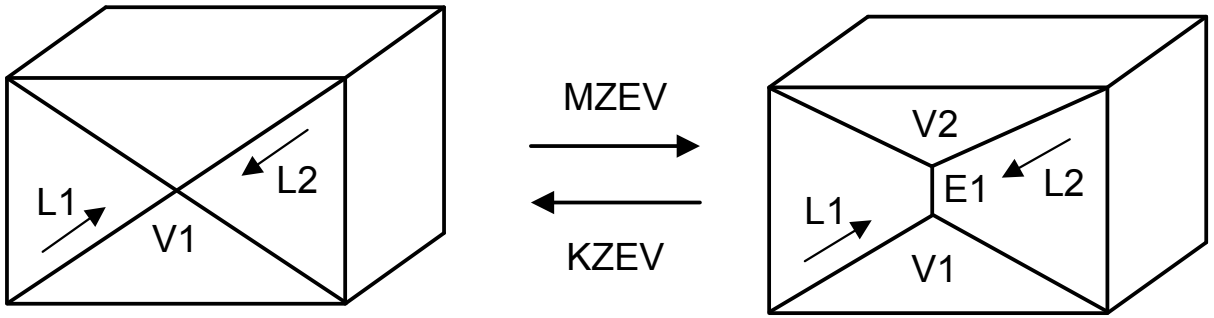
- 5. MEKH and KEMH
  - make (kill) an edge and kill(make) a hole loop



  - Connects vertex V1 on L1 and V2 on L2 by adding E1
    - MEKH (B, V1, V2, L1, L2, &E1)
    - KEMH (B, &V1, &V2, &L1, &L2, E1)

Euler operators used in SNUMOD – cont'

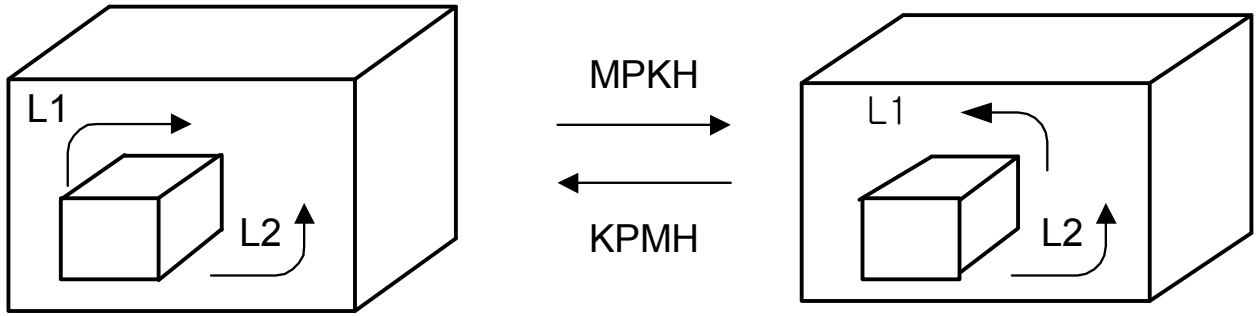
- 6. MZEV and KZEV
  - make (kill) a zero-length edge and a vertex



  - Split vertex V1 into V1 and V2 by adding E1
  - E1 is created such that it belongs to both L1 and L2
    - MZEV (B, L1, L2, V1, &E1, &V2)
    - MEKH (B, &L1, &L2, &V1, E1, V2)

Euler operators used in SNUMOD – cont'

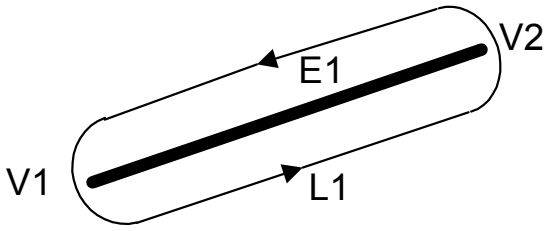
- 7. MPKH and KPMH
  - make (kill) a peripheral loop and kill (make) a hole loop



  - Convert a hole loop L1 to a peripheral Loop
    - MPKH (B, L1, &L2)
    - KPMH (B, L1, L2)

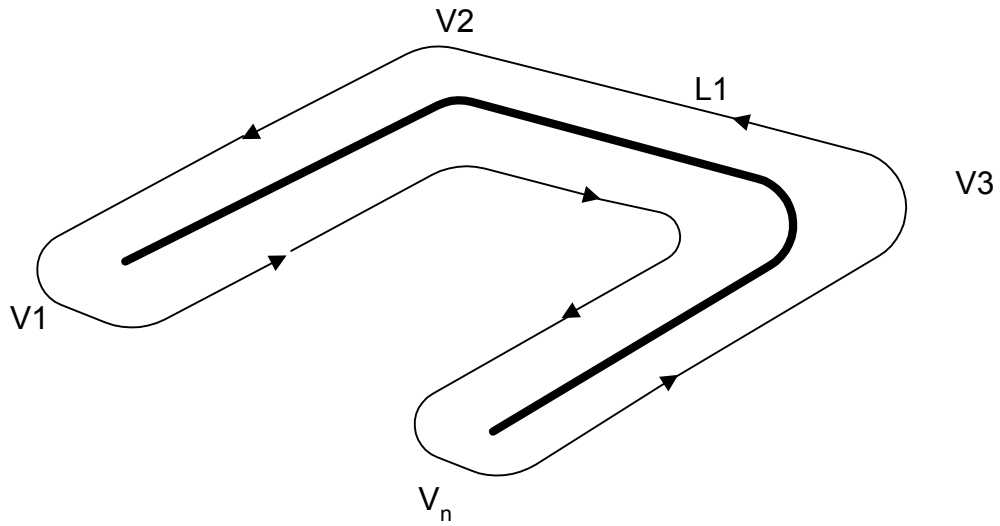
Applications of Euler operators

- Translational sweeping
  - Input : n points Pi (I = 1, …, n) on XY-plane & height of the object
  - 1. An object composed of one edge E1, two vertices V1 and V2, and one loop L1 is created by MEVVLS



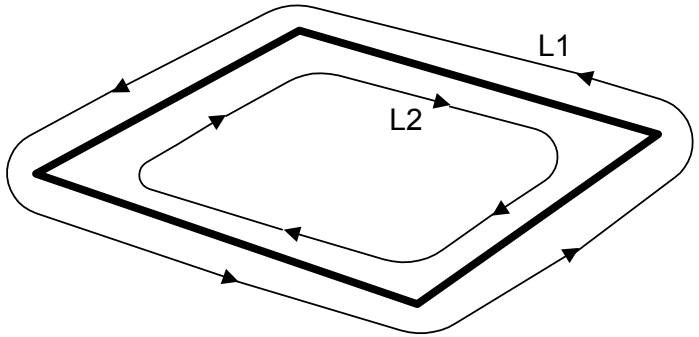
Applications of Euler operators – Translational sweeping

- 2. MEV operation is applied (n-2) times

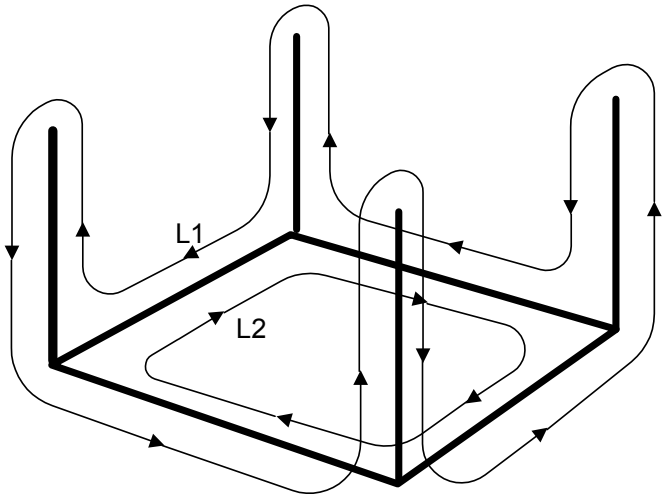


Applications of Euler operators – Translational sweeping

- 3. MEL is applied

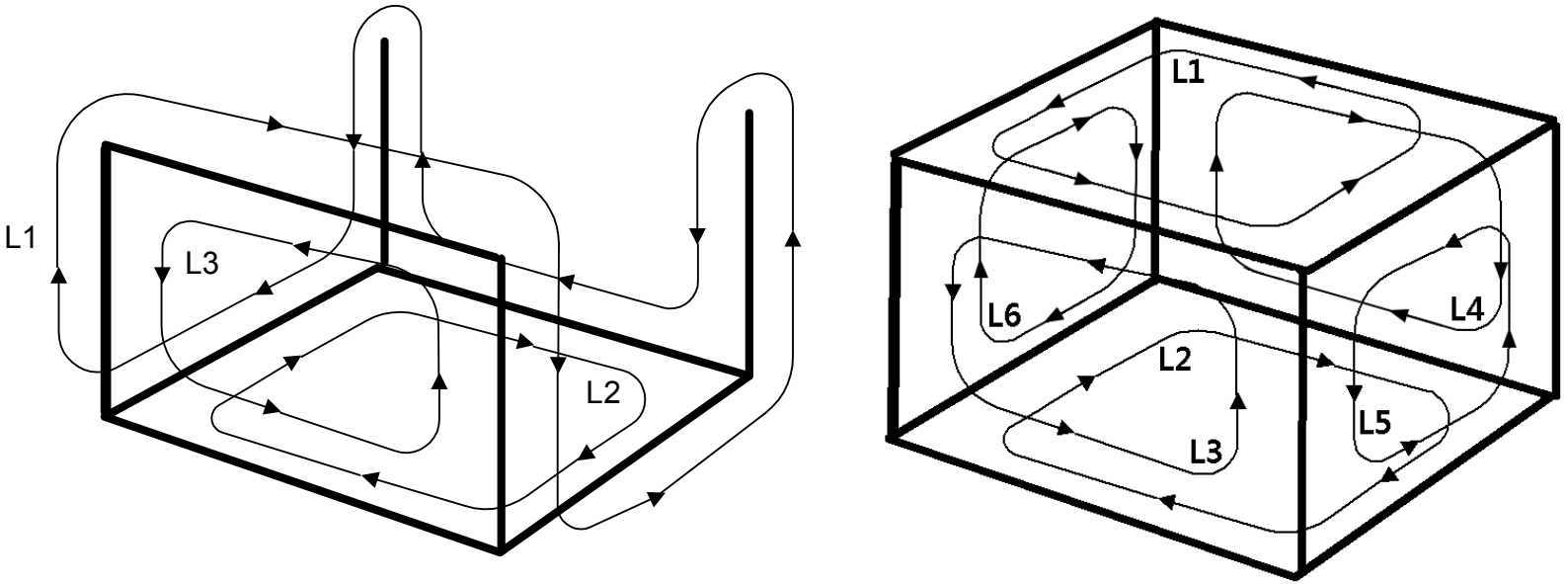


- 4. MEV is applied n times



Applications of Euler operators – Translational sweeping

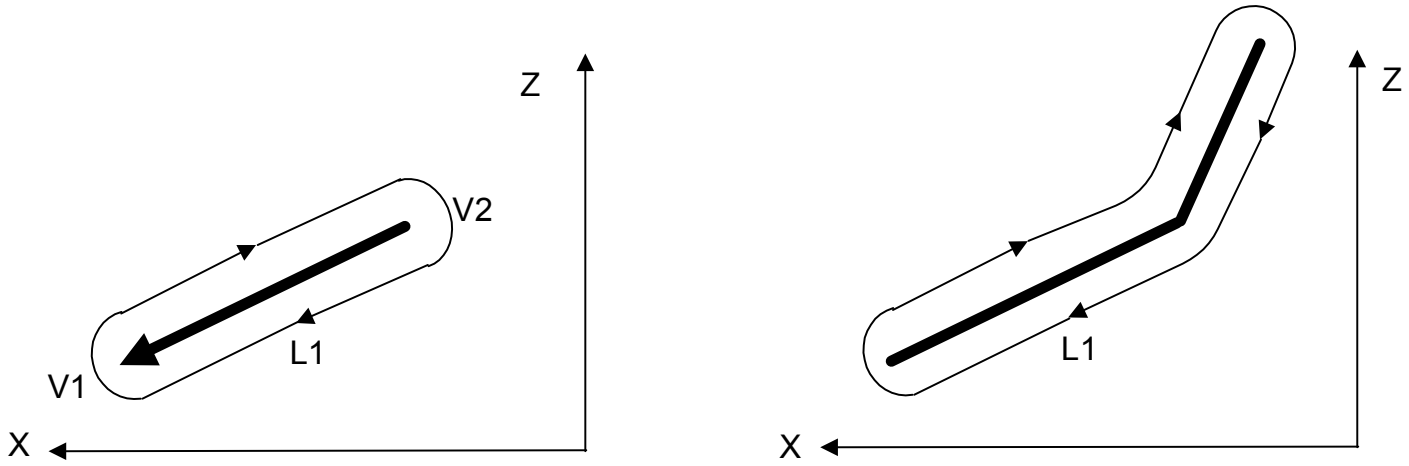
- 5. MEL is applied n times → L1 becomes the Loop of top face



Applications of Euler operators – Rotational Sweeping

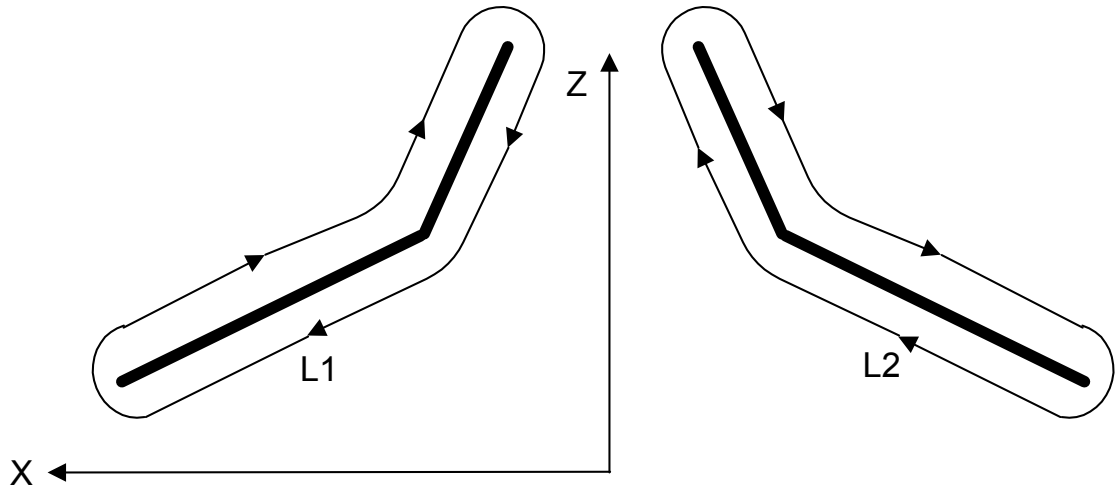
- Input : Pi (i = 1,…., n) on XZ plane
  Rotate with Z-axis

  - 1. MEVVLS from P1 and P2
    MEV (n-2) times



Applications of Euler operators – Rotational Sweeping

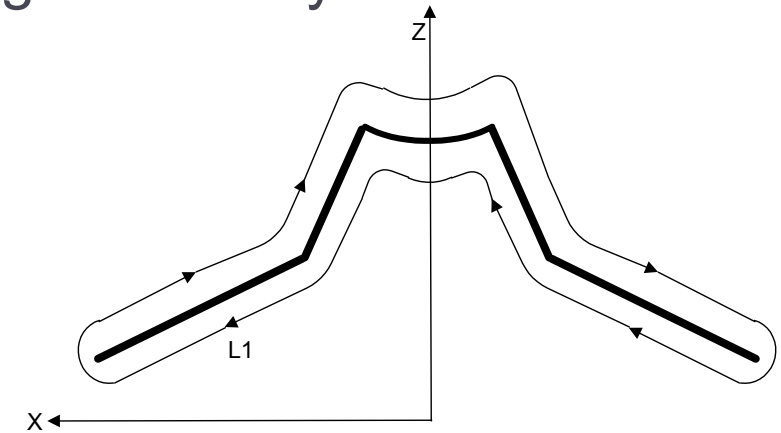
- 2. Symmetric profile lines w.r.t. YZ plane are created by Euler operators as in step1.



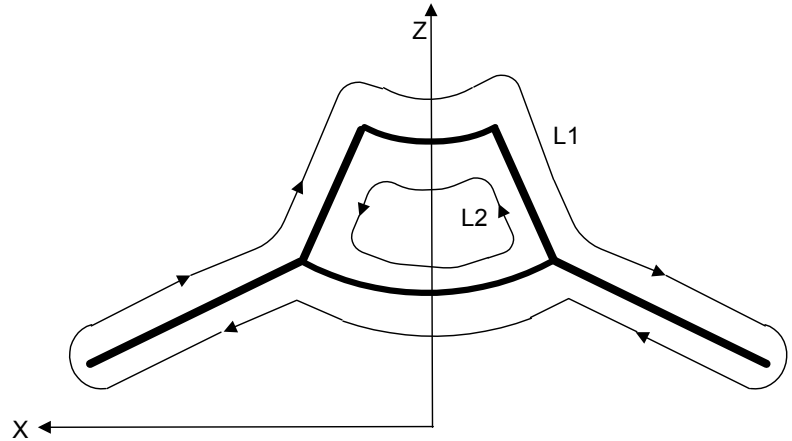
- 3. L2 becomes the hole loop of L1's hole loop by KPMH, and the shell associated with L2 is eliminated

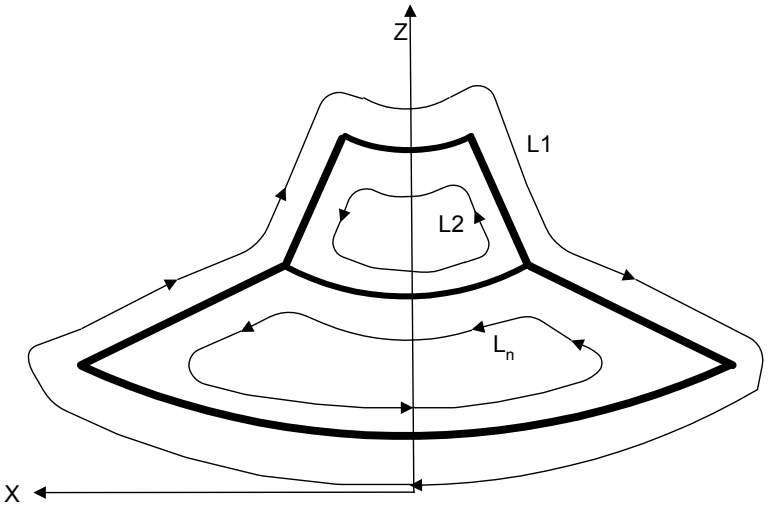
Applications of Euler operators – Rotational Sweeping

- 4. L2 is merged to L1 by MEKH



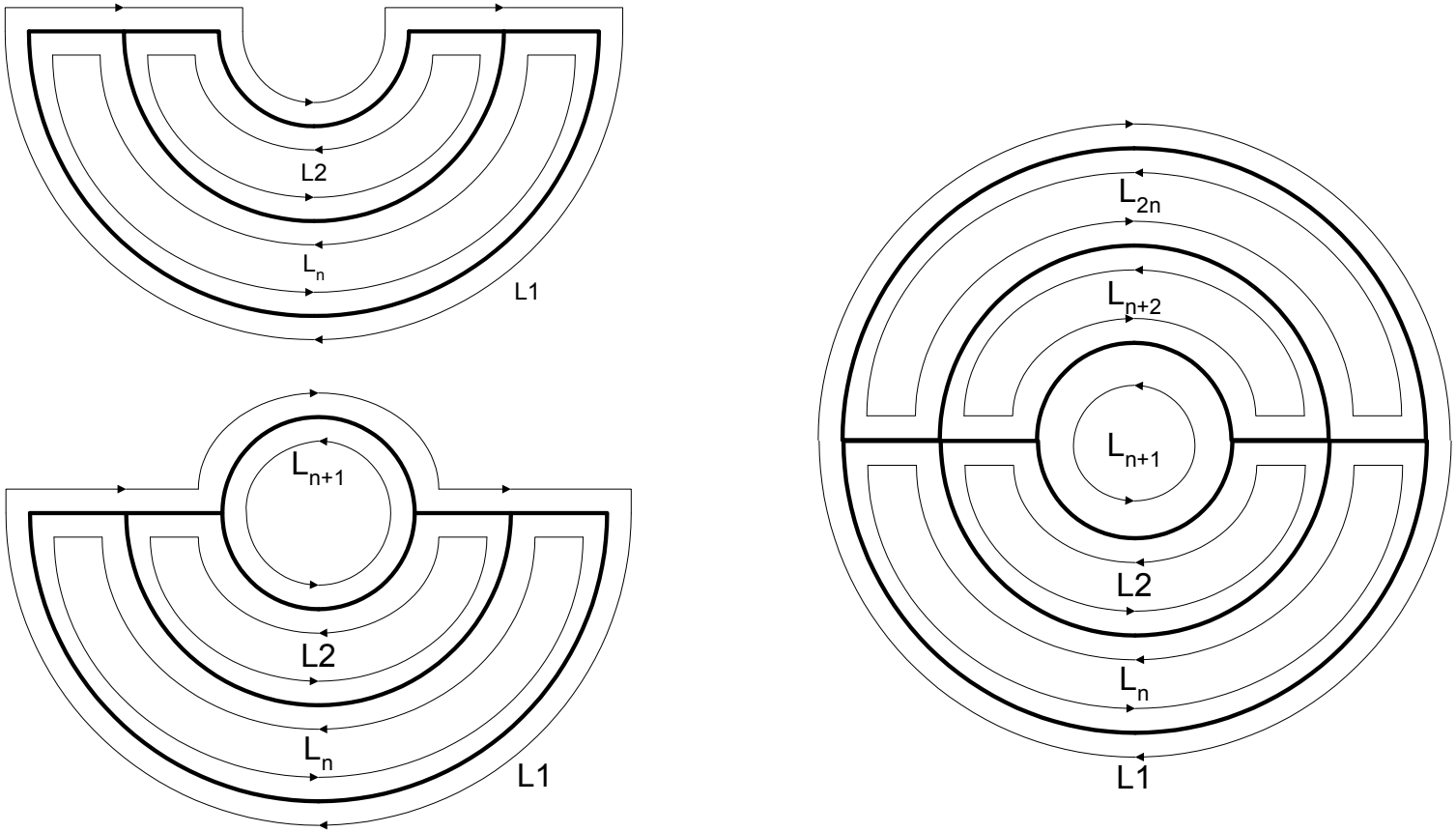
- 5. MEL is applied (n-1) times



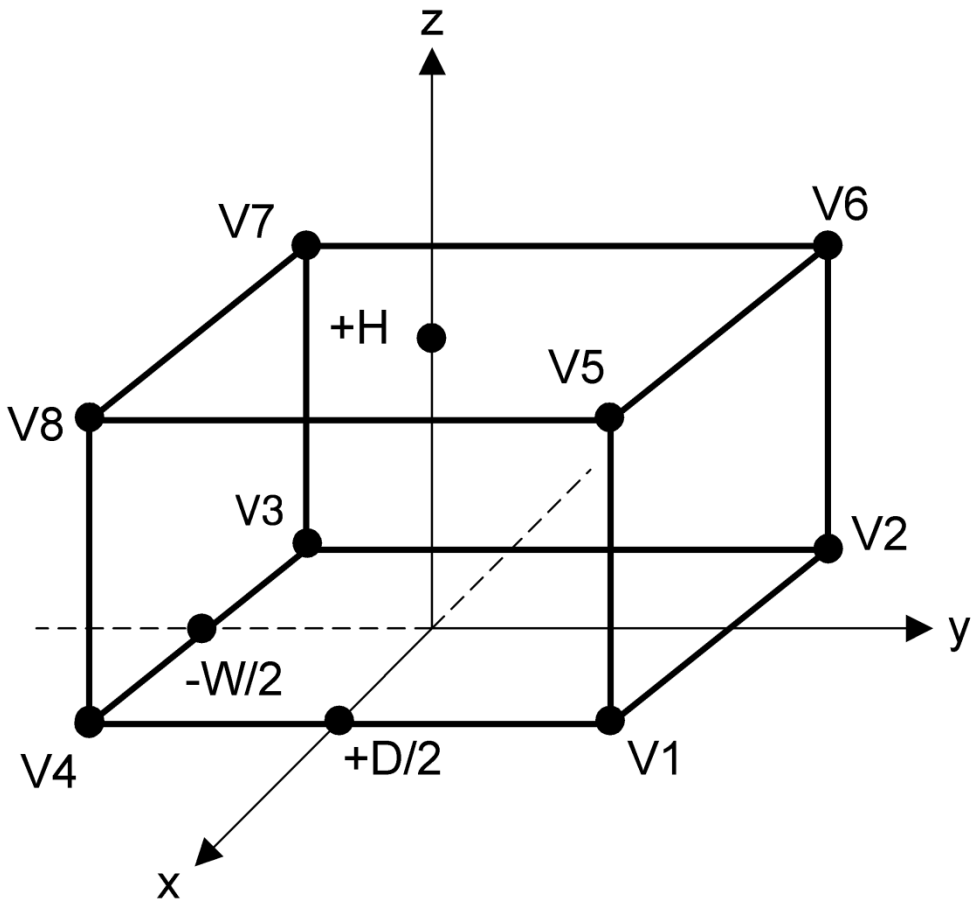


Applications of Euler operators – Rotational Sweeping

- 6. MEL is applied n times to create rear faces



Applications of Euler operators – Primitive creation



Applications of Euler operators – Primitive creation

```
Body            *Create_Block (W, D, H )
    double      W, D, H;
    {
      Body      *B;
      Shell     *S;
      Loop      *L1, *L2, *L3, *L4, *L5, *L6;
      Edge      *E1, *E2, *E3, *E4, *E5, *E6;
      Vertex    *V1, *V2, *V3, *V4, *V5, *V6, *V7, *V8;

      B = malloc ( sizeof ( Body ) );
      MEVVLS ( B, &E1, &V1, &V2, &L1, &S, D/2, W/2, 0, -D/2, W/2, 0 );
      MEV ( B, L1, V2, &E2, &V3, -D/2, -W/2, 0 );
      MEV ( B, L1, V2, &E3, &V4, D/2, -W/2, 0 );
      MEL ( B, L1, V4, V1, &E4, &L2 );
      MEV ( B, L1, V1, &E5, &V5, D/2, W/2, H );
      MEV ( B, L1, V2, &E6, &V6, -D/2, W/2, 0 );
      MEV ( B, L1, V3, &E7, &V7, -D/2, -W/2, H );
      MEV ( B, L1, V4, &E8, &V8, D/2, -W/2, H );
      MEL ( B, L1, V5, V6, &E9, &L3 );
      MEL ( B, L1, V6, V7, &E10, &L4 );
      MEL ( B, L1, V7, V8, &E11, &L5 );
      MEL ( B, L1, V8, V5, &E12, &L6 );
      return ( B );
    }
```

Example of Modeling command

- Splitting Hexagon
  - MPKH will detect the existence of separate shells

    and store the split hexahedra into two separate shells.