Appendix D.
Component Demonstration

Topic: Ignore

This example shows how to retrieve hidden line data using api_phl_retrieve.

```
// Provide a new ENTITY_LIST.
ENTITY_LIST phl_edgelist;
PHL_CAMERA* camera
ENTITY_LIST bodies

// Retrieve hidden line data of a given scene.
outcome result = api_phl_retrieve(
    bodies, token phl_edgelist, camera);
if (! result.ok()) return(result);

// Scan the resulting PHL_EDGEs.
int num_edges = phl_edgelist.count();
for (int i = 0; i < num_edges; i++) {

    // Get the i-th PHL_EDGE of the list.
    PHL_EDGE* phl_edge = phl_edgelist[i];

    // Get the real EDGE of the PHL_EDGE.
    EDGE* edge = phl_edge->edge();

    // Get the CURVE of the EDGE.
    CURVE* cuofed = edge->geometry();

    // Determine if the edge is reversed.
    logical rev = edge->sense() == REVERSED;

    // Get the owning body.
    BODY* body = phl_edge->body();
```

```
// Decide if the curve data should be transformed by the
// body transformation.
SPAtransf* draw_tra;
if  (phl_edge->face() == 0) {
    // regular edge: apply body transformation
    if  (body->transform() == 0) {
        draw_tra = 0;
    }
    else {
        draw_tra = &(body->transform()->transform());
    }
else {
    // silhouette edge: don't apply body transformation
    draw_tra = 0;
}

// Scan the PHL_SEGMENTs of this PHL_EDGE.
for (PHL_SEGMENT* phl_seg = phl_edge->phl_segment();
        phl_seg; phl_seg = phl_seg->next()) {

    // Get the start and end parameters of the segment.
    SPAparameter sta = phl_seg->start_pt();
    SPAparameter end = phl_seg->end_pt();

    // Swap and negate the parameters if the edge is reversed.
    if (rev) {
        SPAparameter tmp = sta;
        sta = - end;
        end = - tmp;
    }

    // Decide if the segment is "outer" or "inner."
    PhlSegSta sta = phl_seg->state();
    if  (sta == PHL_SEGMENT::OUT) {
        // Some action for an "outer" segment
    }
    else if (sta == PHL_SEGMENT::INN) {
        // Some action for an "inner" segment
    }
```

```
        // Decide if the segment is visible, hidden, or occluded.
        PhlSegVis vis = phl_seg->visibility();
        if  (vis == PHL_SEGMENT::VIS){
            // Some action for a visible segment
            do_vis_seg(cuofed, sta, end);
        }
        else if (vis == PHL_SEGMENT::HID){
            // Some action for a hidden segment
            do_hid_seg(cuofed, sta, end);
        }
        else if ( vis == PHL_SEGMENT::OCC){
            // Some action for an occluded segment
            do_occ_seg( cuofed, sta, end);
        }

    } // Next PHL_SEGMENT

} // Next PHL_EDGE

// Lose the PHL_EDGEs.
for(i = 0; i < num_edges; i++) {
    (*phl_edgelist)[i]->lose();
}

// Delete the ENTITY_LIST.

delete phl_edgelist;

// Delete the PHL_CAMERA, which is a copy of the one stored
// with the bodies.

camera->lose();
```