Room Escape Game

Team 2

20131064 김동훈
20141440 이승원
20151506 정하빈

Overview

Task

1. Loading & Rendering Object
2. Movable First-Person View Camera
3. Object Selection

Demo

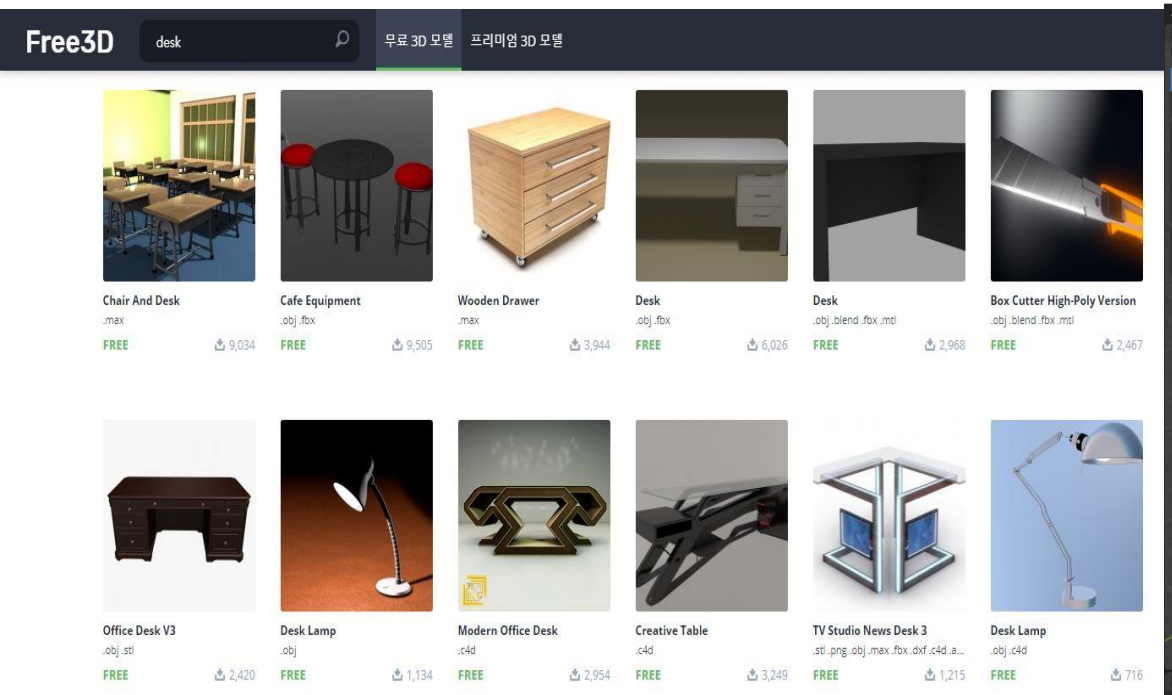
Game play demo

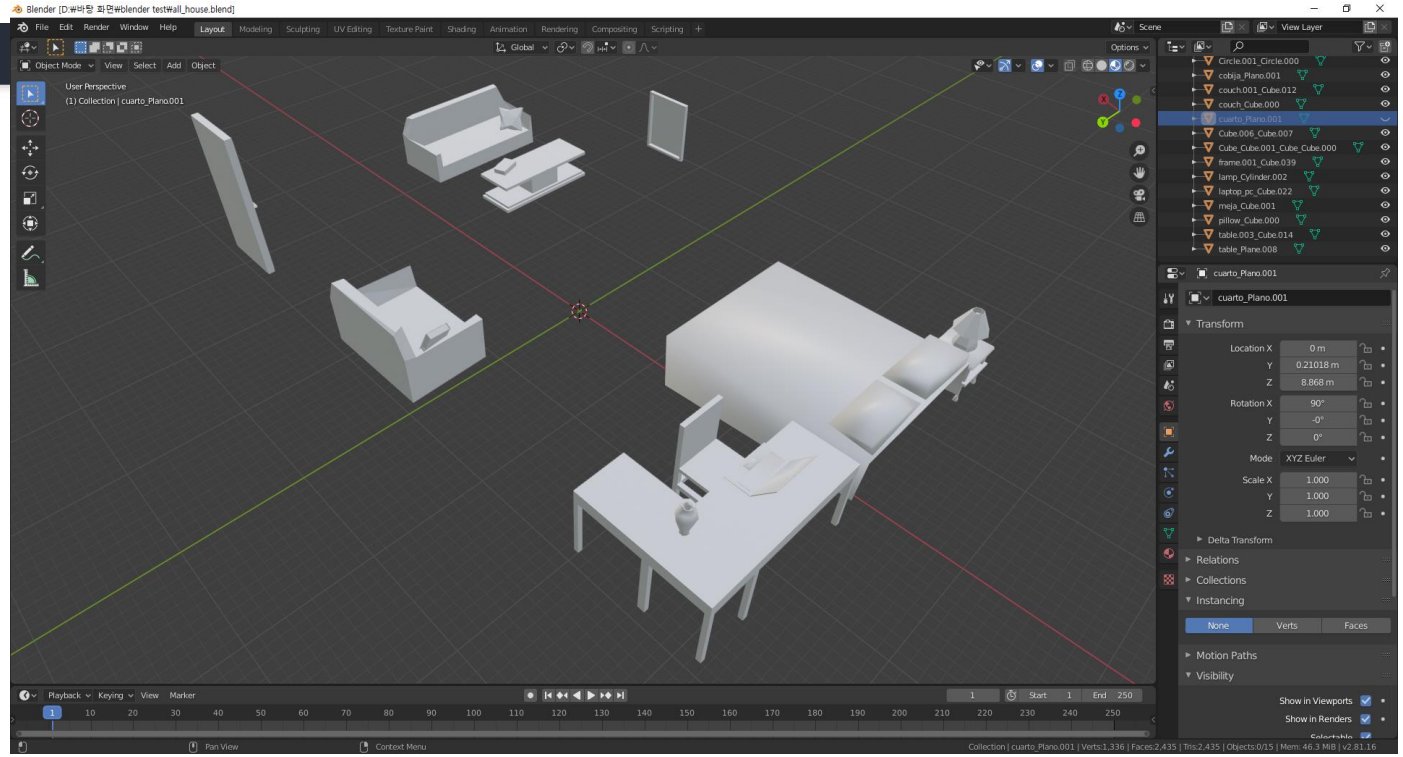
Loading & Rendering Objects

How to read and render object files?

Loading & Rendering Object

Blender





Loading & Rendering Object

Loading

.obj file format

```
v -2.011660 5.999999 6.400001
v 5.670000 4.970498 -3.407556
```

```
f 10/10/4 11/11/4 12/12/4
```

store to variables

```
if (strcmp(lineHeader, "v") == 0) {
    vec3 vertex;
    int lineData = fscanf(file, "%f %f %f\n", &vertex.x, &vertex.y, &vertex.z);
```

```
else if (strcmp(lineHeader, "f") == 0) {
    unsigned int vertexIndex[3], uvIndex[3], normalIndex[3];
    int matches = fscanf(file, "%d/%d/%d %d/%d/%d %d/%d/%d\n",
                        &vertexIndex[0], &uvIndex[0], &normalIndex[0],
                        &vertexIndex[1], &uvIndex[1], &normalIndex[1],
                        &vertexIndex[2], &uvIndex[2], &normalIndex[2]);
```

Loading & Rendering Object

Render (VBO)

VAO & VBO

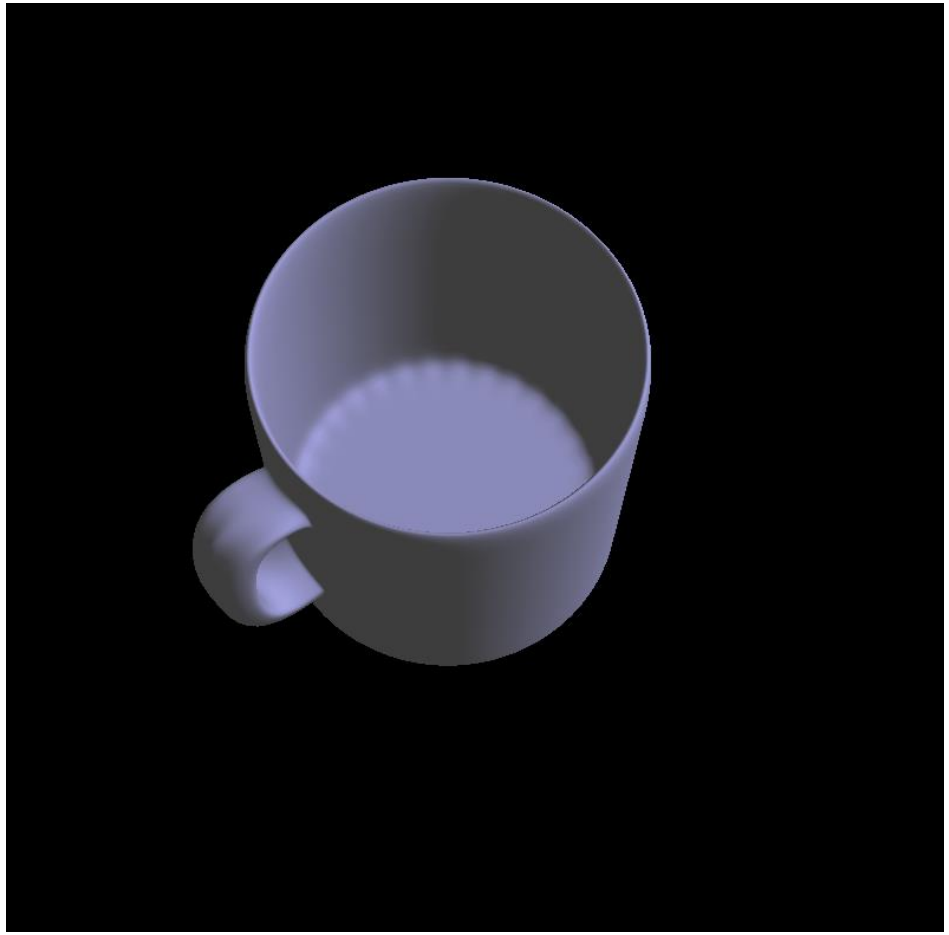
```
/* generate VAO */
glGenVertexArrays(1, &object->vao);
glBindVertexArray(object->vao);

/* generate VBO */
glGenBuffers(1, &object->vbo);
glBindBuffer(GL_ARRAY_BUFFER, object->vbo);
glBufferData(GL_ARRAY_BUFFER, object->vertices.size() * sizeof(vec3), &object->vertices[0], GL_STATIC_DRAW);
glBindBuffer(GL_ARRAY_BUFFER, 0);
```

Draw object

```
/* bind VAO & EBO */
glBindVertexArray(object->vao);
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, object->ebo);
glDrawElements(GL_TRIANGLES, (GLsizei)object->indices.size(), GL_UNSIGNED_INT, (void*)0);
```

Loading & Rendering Object

Render (VBO)

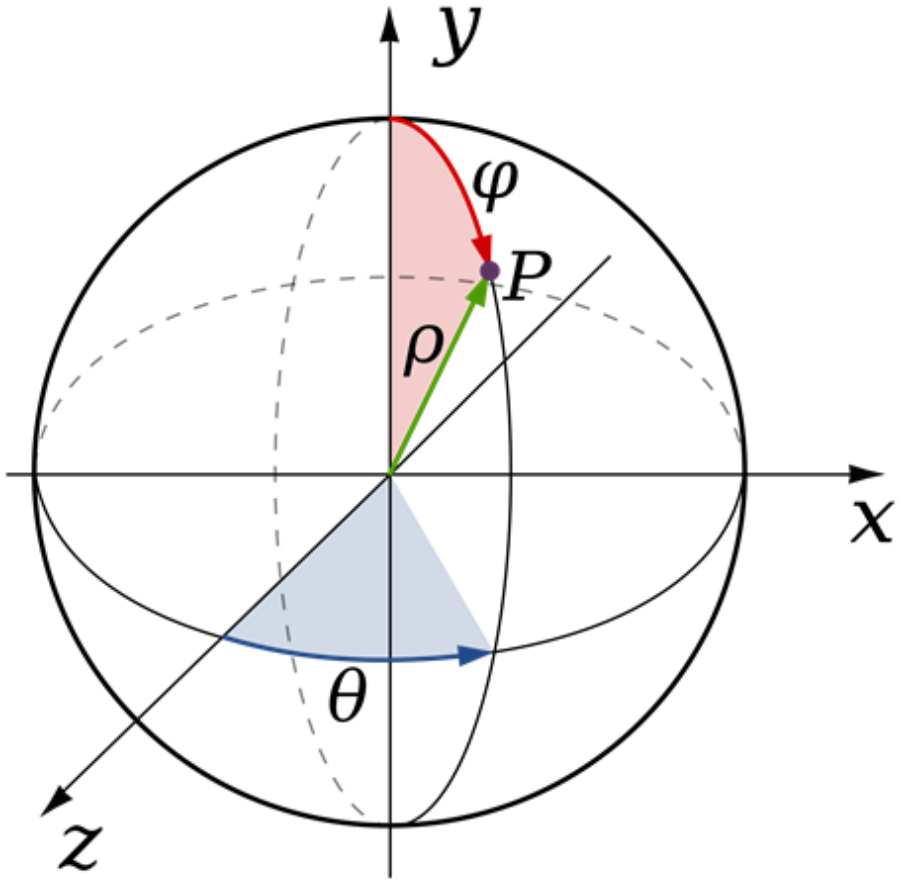


Movable First-Person View Camera

How to implement human's viewpoint?

Movable First-Person View Camera

Camera view direction



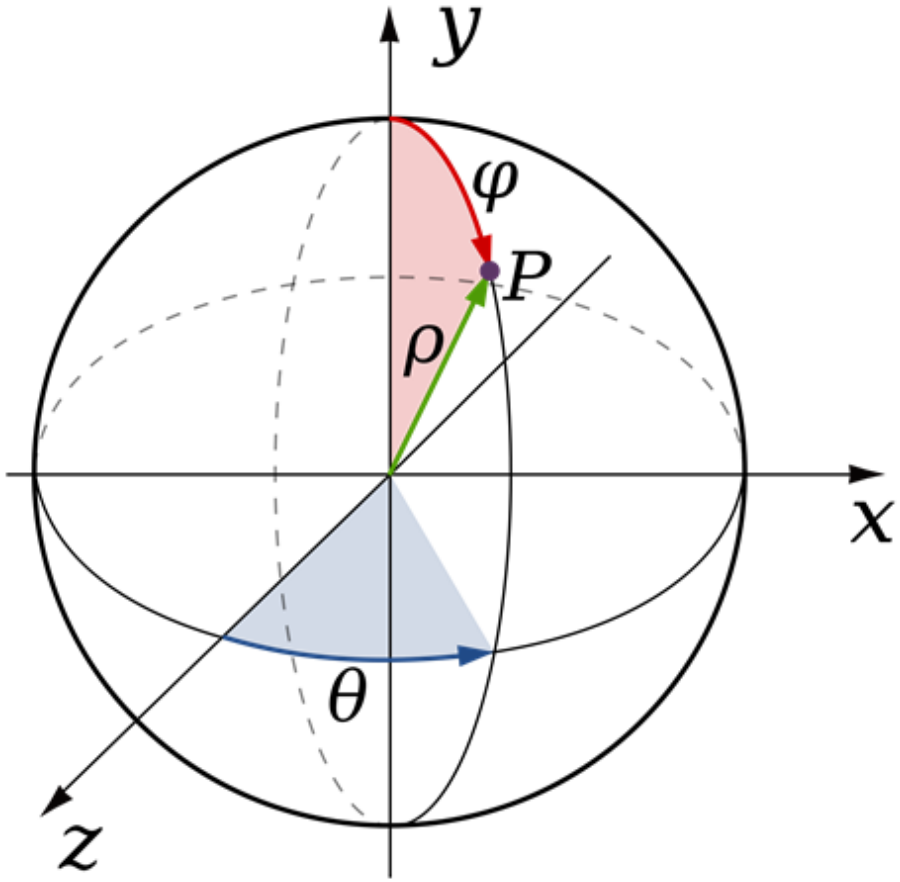
- Every point on a 3D unit sphere can be represented as two angles, theta and pi.

- Start with the point (0,1,0), rotate it about x-axis by pi
- Then rotate the result about y-axis by theta

- The view direction can be represented as OP vector (from the origin to that point).

Movable First-Person View Camera

Camera view direction

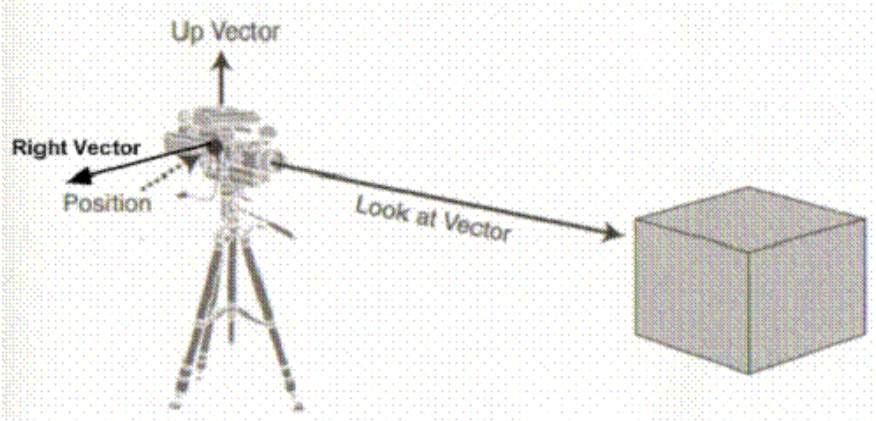



```
/* register callback function */
glutReshapeFunc(reshape);
glutDisplayFunc(display);
glutIdleFunc(idle);
glutKeyboardFunc(keyboard);
glutMouseWheelFunc(mouseWheel);
glutMouseFunc(pickObject);
glutPassiveMotionFunc(passiveMouseMove);
```

```
void passiveMouseMove(int x, int y)
{
    float mouseMove_x = (float)(mousePos[0] - x);
    float mouseMove_y = (float)(mousePos[1] - y);
    float rotateRate = 2400.0f;
    float tempAt[3];

    glutSetCursor(mouseMode);
    glutWarpPointer(screenSize / 2, screenSize / 2);
    mousePos[0] = screenSize / 2;
    mousePos[1] = screenSize / 2;

    theta += (float)(6.28318 * mouseMove_x / rotateRate);
    if (theta >= 6.28318)       theta -= 6.28318f;
    else if (theta <= -6.28318) theta += 6.28318f;

    pi += (float)(1.570795 * mouseMove_y / rotateRate);
    if (pi >= 1.570795)         pi = 1.570795f;
    else if (pi <= -1.570795)   pi = -1.570795f;

    rightVec = normalize(cross(at, up));

    at.x = 0;
    at.y = sin(pi);
    at.z = -cos(pi);

    tempAt[0] = cos(theta) * at.x + sin(theta) * at.z;
    tempAt[1] = at[1];
    tempAt[2] = -sin(theta) * at.x + cos(theta) * at.z;

    at.x = tempAt[0];
    at.y = tempAt[1];
    at.z = tempAt[2];
}
```

Movable First-Person View Camera

Camera position



- Right vector can be generated by cross-product Look at vector (view vector) and up vector ((0,1,0) in here).

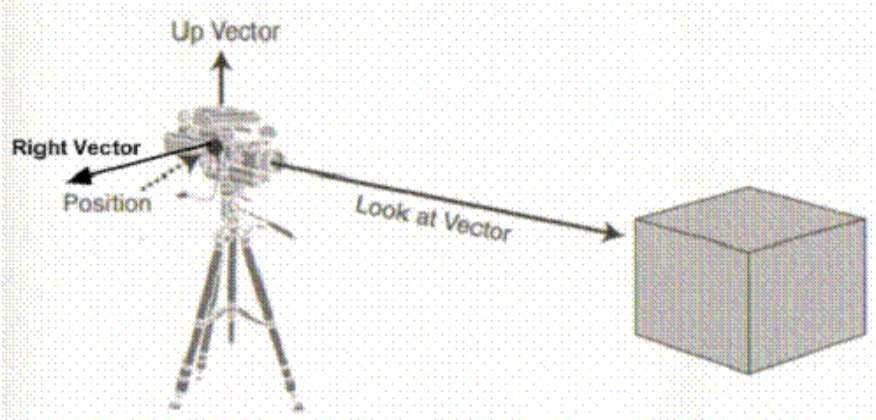
```
rightVec = normalize(cross(at, up));
```

- To maintain the height while moving camera, remove y value from view vector and right vector, then normalize them.

```
void keyboard(unsigned char key, int x, int y)
{
    float moveRate = 2.0;
    vec3 temp_at = normalize(vec3(at.x, 0, at.z));
    vec3 temp_right = normalize(vec3(rightVec.x, 0, rightVec.z));

    if (key == 'w' || key == 'W') {
        eyePosition += temp_at / moveRate;
    }
    else if (key == 's' || key == 'S') {
        eyePosition -= temp_at / moveRate;
    }
    else if (key == 'a' || key == 'A') {
        eyePosition -= temp_right / moveRate;
    }
    else if (key == 'd' || key == 'D') {
        eyePosition += temp_right / moveRate;
    }
    glutPostRedisplay();
}
```

Movable First-Person View Camera

Combining them



By combining camera position and camera view direction like this, we can implement movable first-person view camera.

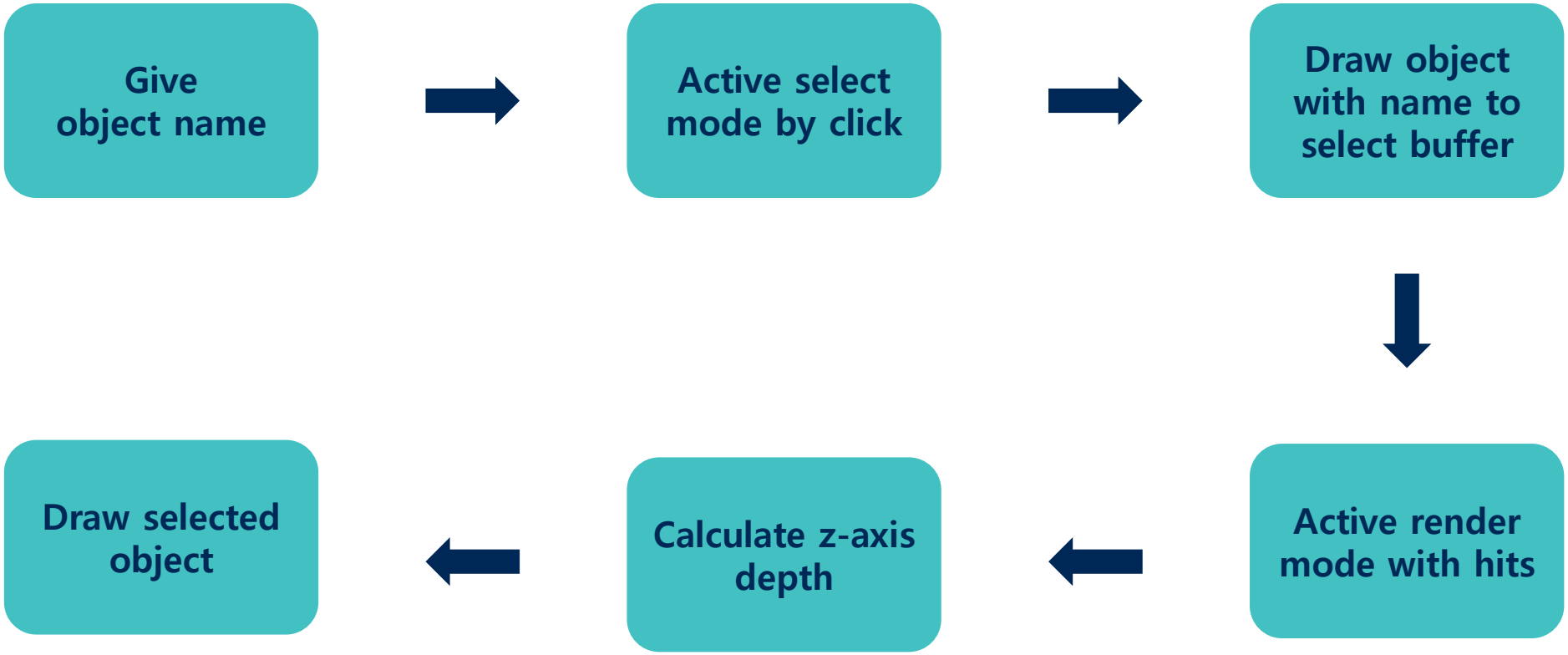
```
gluLookAt(eyePosition.x, eyePosition.y, eyePosition.z,
          eyePosition.x + at.x, eyePosition.y + at.y, eyePosition.z + at.z,
          0.0, 1.0, 0.0);
```

Object Select

How to select specific object?

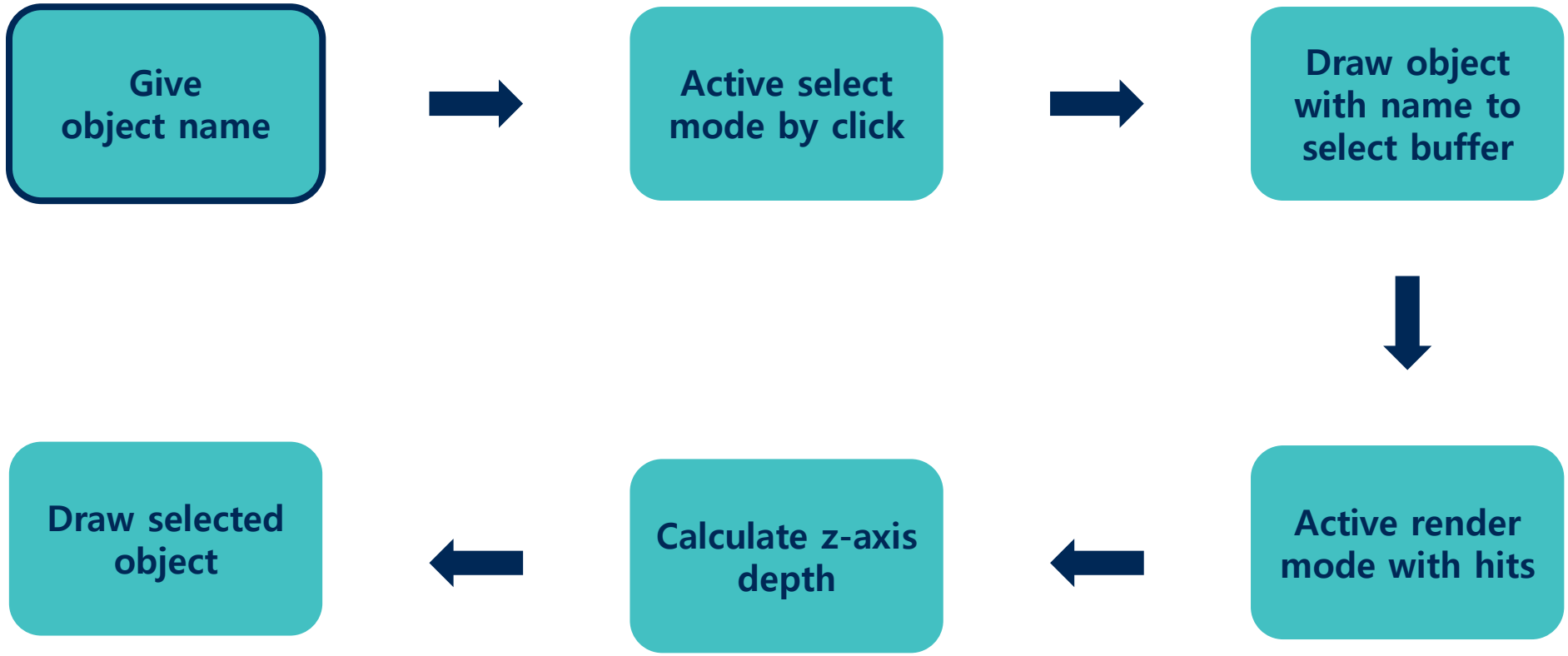
Object Select

Selection process



Object Select

Give object name



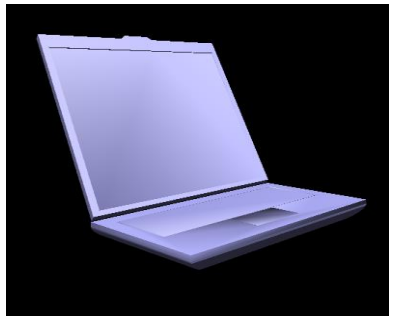
Object Select

Give object name

1. glPushName() : Set object name. It is a unique value.
2. Draw specific object.
3. Unbind name by glPopName().

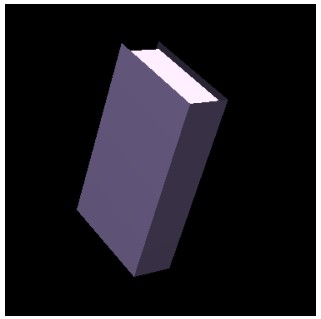
drawAllObjects()

```
{
    glPushMatrix();
    glPushName(obj_book->ID);
    {
        glColor3f(0.8, 0.7, 1.0);
        drawOBJ(obj_book);
    }
    glPopName();
    glPopMatrix();
}
```



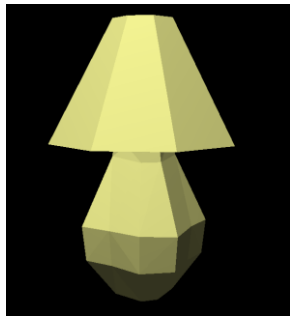
object : laptop

name : obj_laptop->ID



object : book

name : obj_book->ID

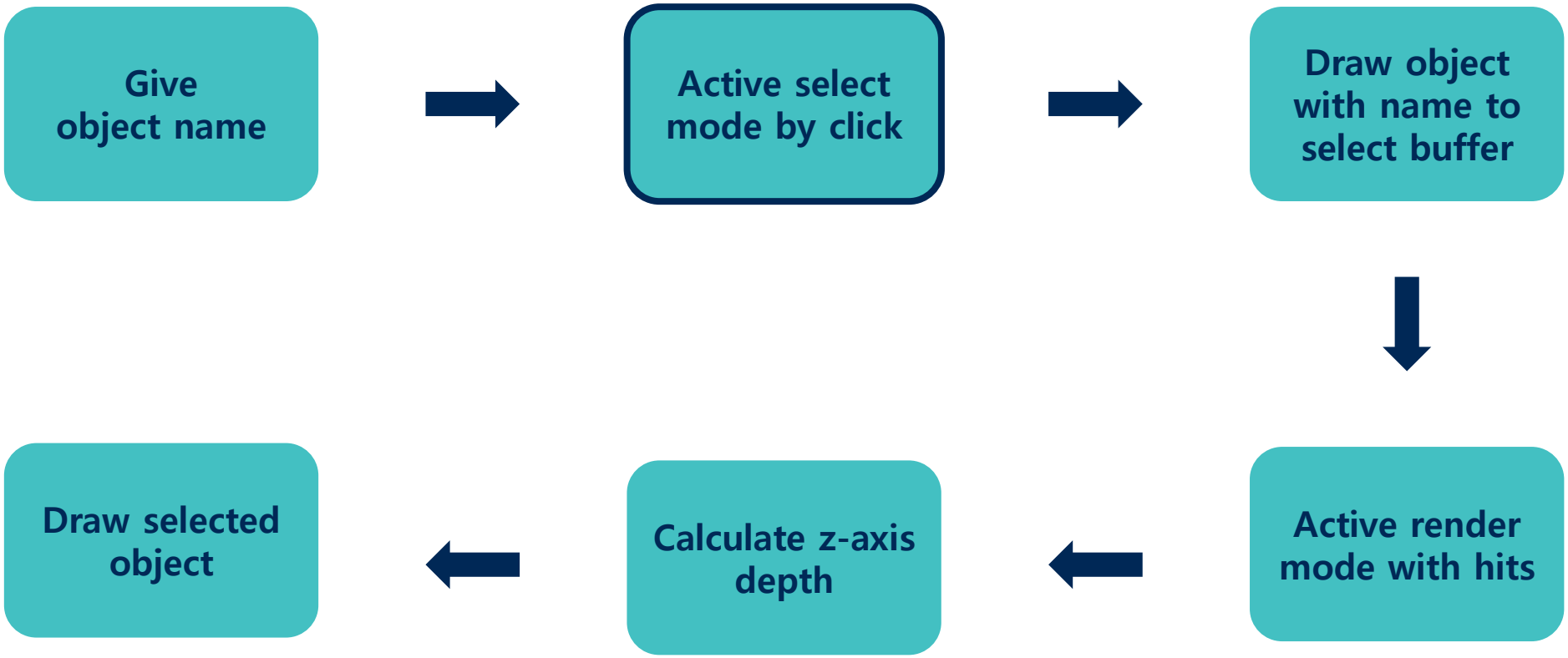


object : lamp

name : obj_lamp->ID

Object Select

Active select mode by click



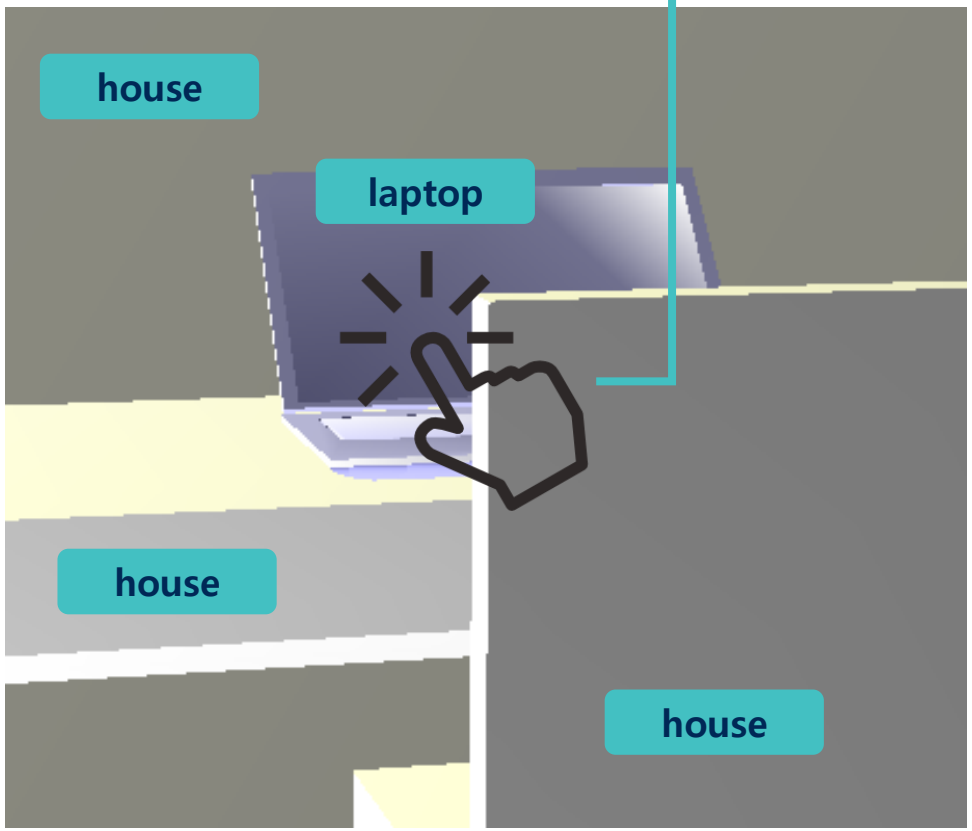
Object Select

Active select mode by click

pickObject()

```
{
    glSelectBuffer(bufferSize, selectBuf);
    (void)glRenderMode(GL_SELECT);

    glMatrixMode(GL_PROJECTION);
    glPushMatrix();
    {
        glLoadIdentity();
        gluPickMatrix((GLdouble)x, (GLdouble)(viewport[3] - y),
                      pickSize, pickSize, viewport);
        gluPerspective(40.0, shapeRatio, 1.0, 300.0);
        drawAllObjects(GL_SELECT);

        glMatrixMode(GL_PROJECTION);
    }
    glPopMatrix();

    hits = glRenderMode(GL_RENDER);
    processHits(hits, selectBuf);

    glutPostRedisplay();
}
```

glutMouseFunc(pickObject)



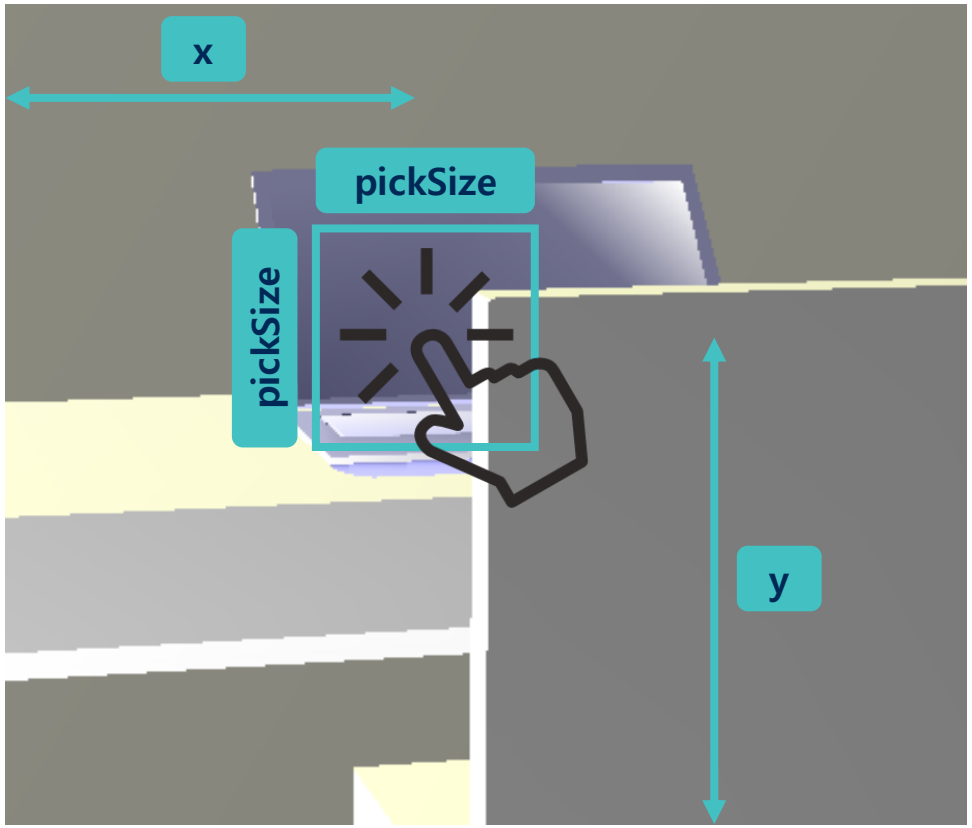
Object Select

Active select mode by click

pickObject()

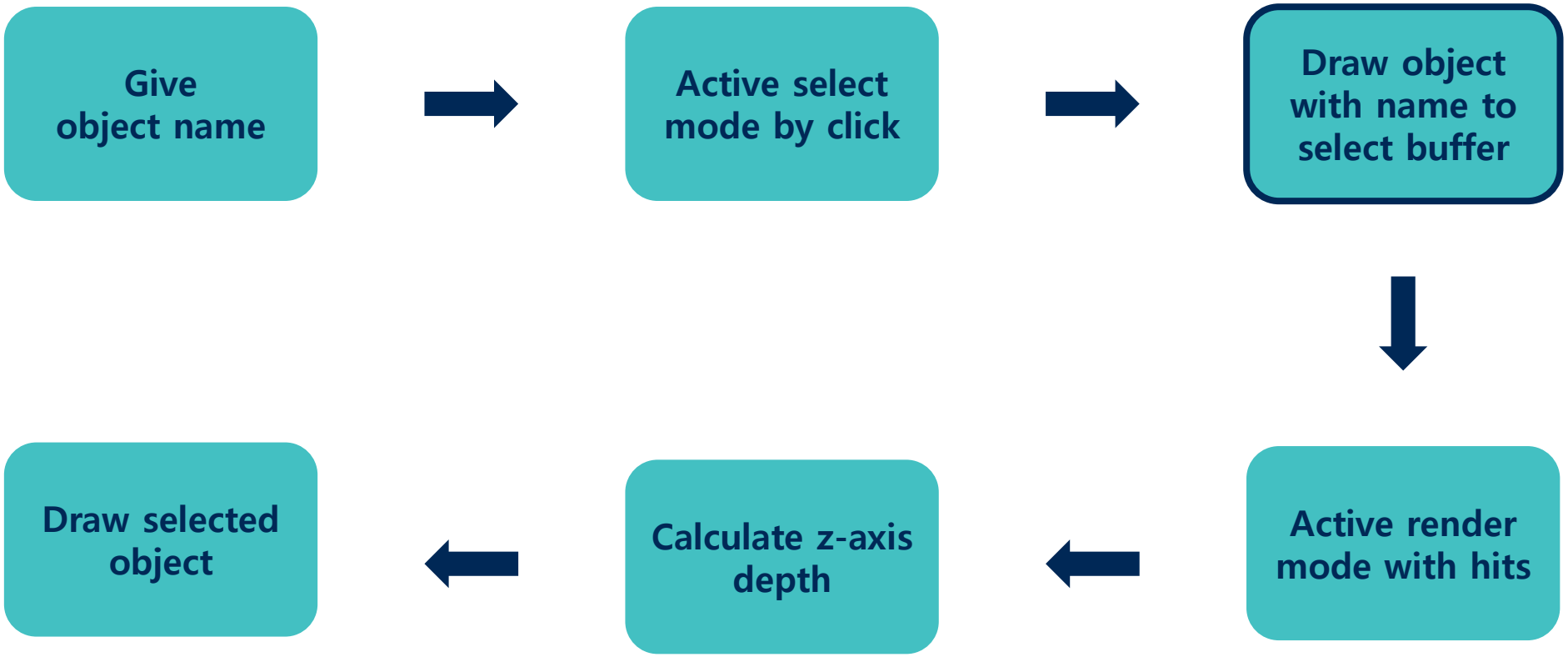
```
{
    glSelectBuffer(bufferSize, selectBuf);
    (void)glRenderMode(GL_SELECT);

    glMatrixMode(GL_PROJECTION);
    glPushMatrix();
    {
        glLoadIdentity();
        gluPickMatrix((GLdouble)x, (GLdouble)(viewport[3] - y),
                      pickSize, pickSize, viewport);
        gluPerspective(40.0, shapeRatio, 1.0, 300.0);
        drawAllObjects(GL_SELECT);

        glMatrixMode(GL_PROJECTION);
    }
    glPopMatrix();

    hits = glRenderMode(GL_RENDER);
    processHits(hits, selectBuf);

    glutPostRedisplay();
}
```

viewport(screenSize, screenSize)



Object Select

Draw object with name to select buffer



Object Select

Draw object with name to select buffer

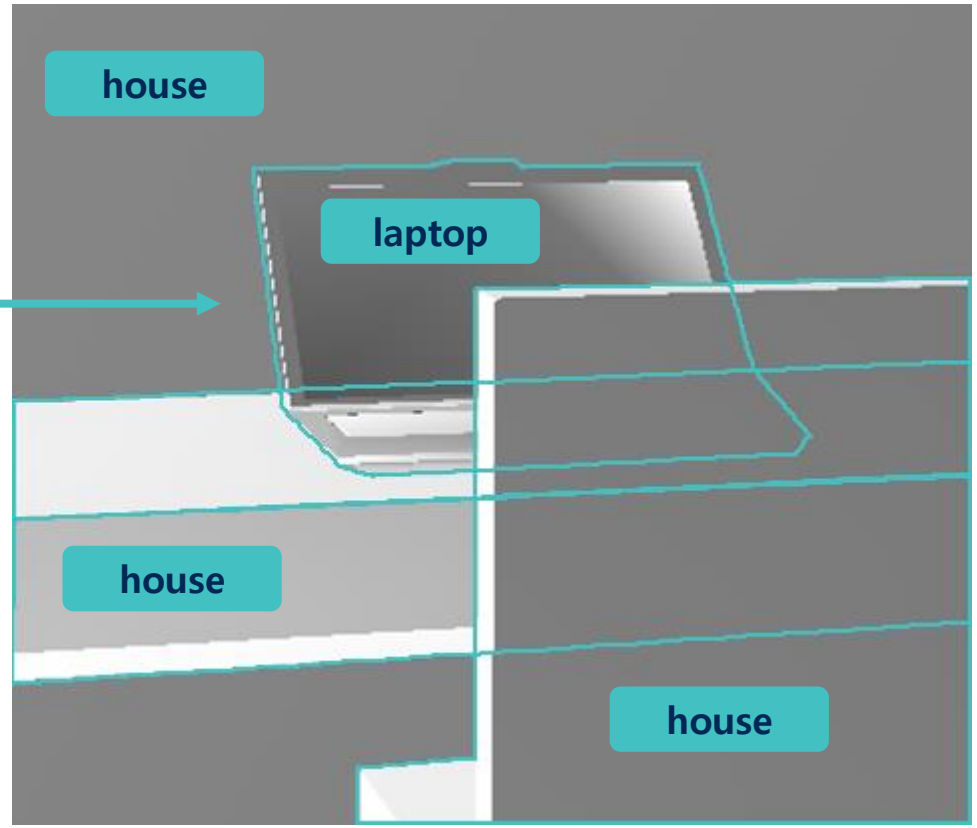
pickObject()

```
{
    glLoadIdentity();
    gluPickMatrix((GLdouble)x, (GLdouble)(viewport[3] - y),
                  pickSize, pickSize, viewport);
    gluPerspective(40.0, shapeRatio, 1.0, 300.0);
    drawAllObjects(GL_SELECT);
}
```

drawAllObjects()

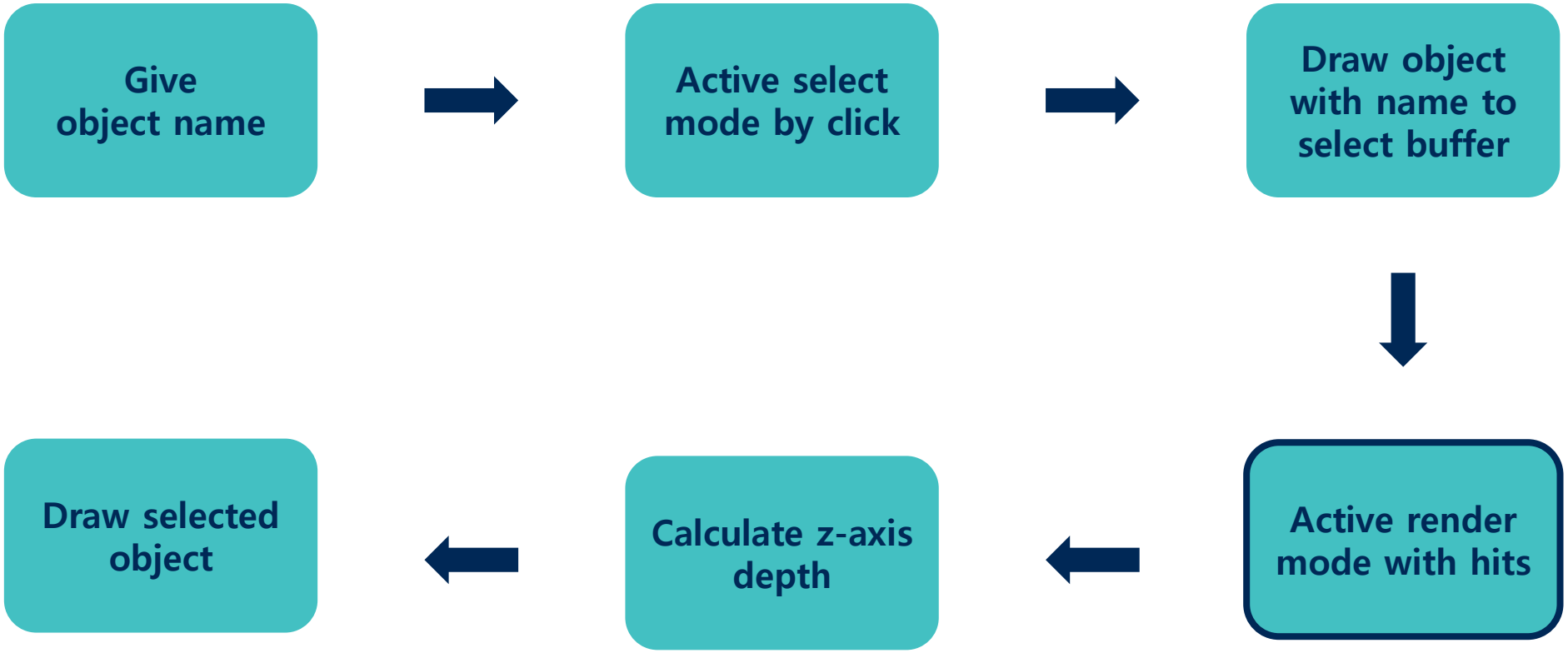
```
{
  if (mode == GL_SELECT) glLoadName(obj_laptop->ID);
  glPushMatrix();
  glPushName(obj_laptop->ID);
  {
    glColor3f(0.7, 0.7, 1.0);
    drawOBJ(obj_laptop);
  }
  glPopName();
  glPopMatrix();
}
```

drawAllObject(GL_SELECT)



Object Select

Active render mode with hits



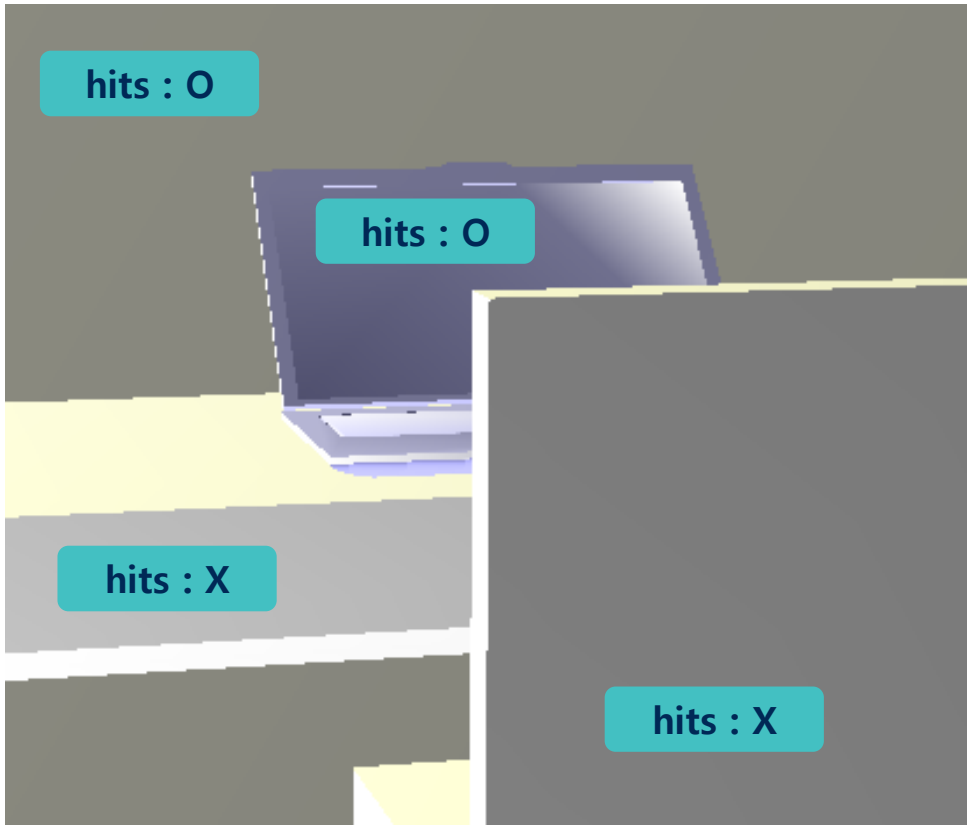
Object Select

Active render mode with hits

pickObject()

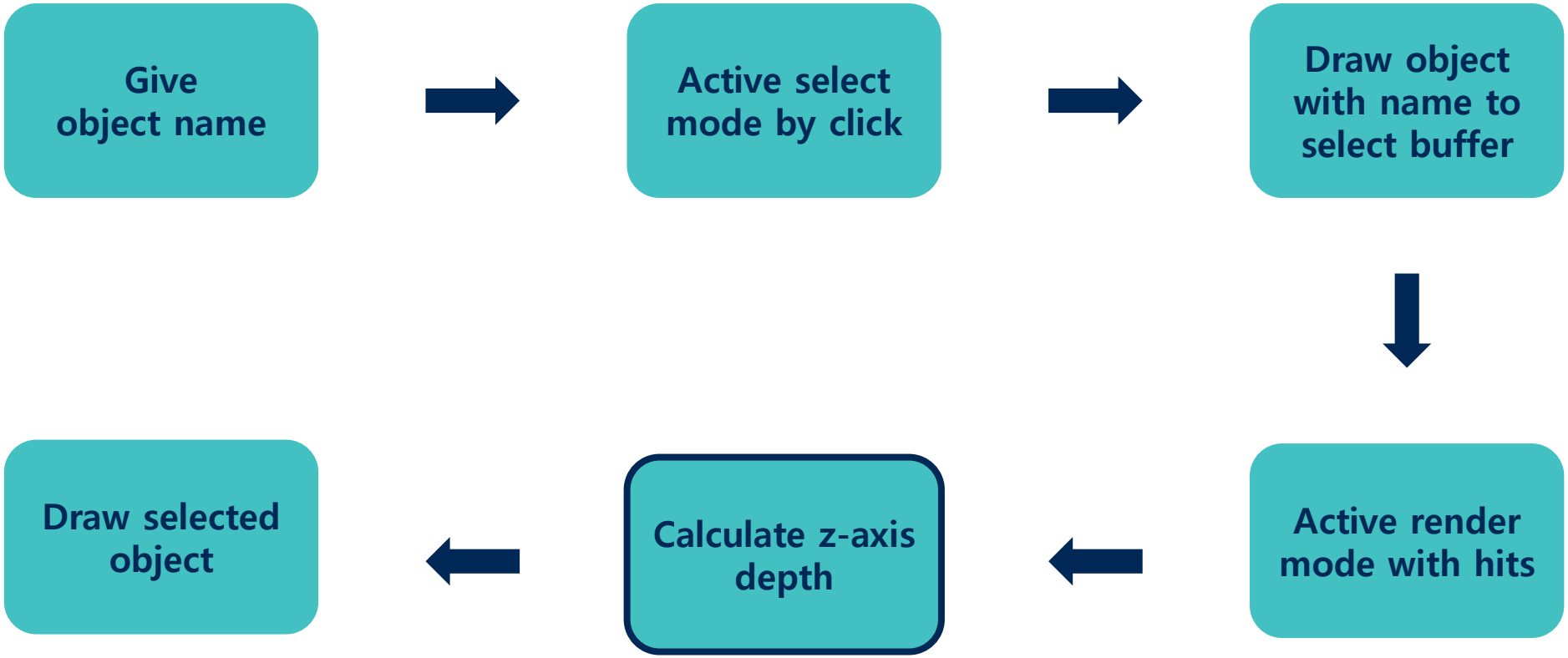
```
{
    glSelectBuffer(bufferSize, selectBuf);
    (void)glRenderMode(GL_SELECT);

    glMatrixMode(GL_PROJECTION);
    glPushMatrix();
    {
        glLoadIdentity();
        gluPickMatrix((GLdouble)x, (GLdouble)(viewport[3] - y),
                      pickSize, pickSize, viewport);
        gluPerspective(40.0, shapeRatio, 1.0, 300.0);
        drawAllObjects(GL_SELECT);

        glMatrixMode(GL_PROJECTION);
    }
    glPopMatrix();

    hits = glRenderMode(GL_RENDER);
    processHits(hits, selectBuf);

    glutPostRedisplay();
}
```

hits = glRenderMode(GL_RENDER)



Object Select

Calculate z-axis depth



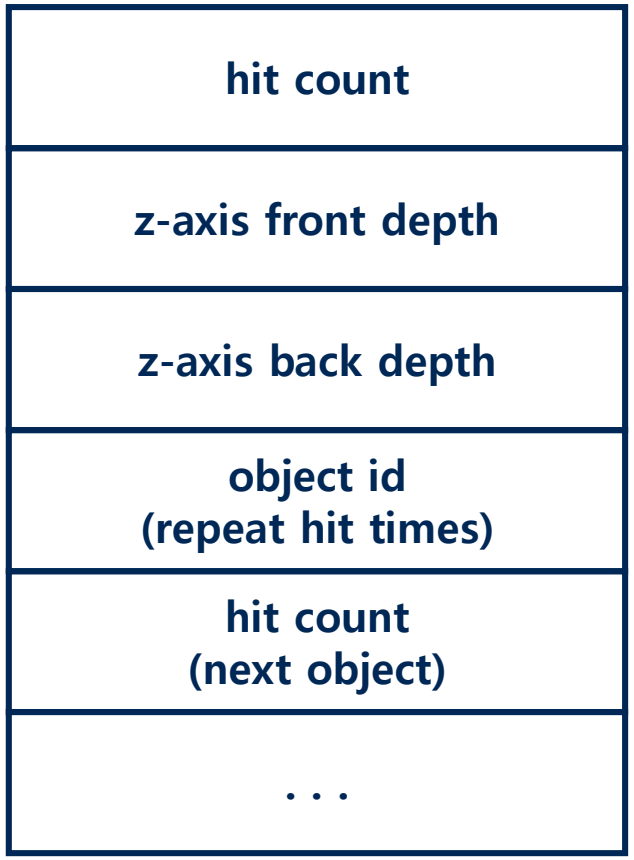
Object Select

Calculate z-axis depth

processHits(hits, buffer[])

```
{
    if (hits != 0) {
        for (unsigned int i = 0; i < hits; i++) {
            hitCounts = *ptr;
            ptr++;

            zFront = (float)*ptr / 0x7fffffff;
            ptr++;

            zBack = (float)*ptr / 0x7fffffff;
            ptr++;

            objectID = *ptr;

            for (unsigned int j = 0; j < hitCounts; j++)
                ptr++;

            depthBuf[objectID] = zFront - 1.0f;
        }

}
```

ptr = buffer[] stack



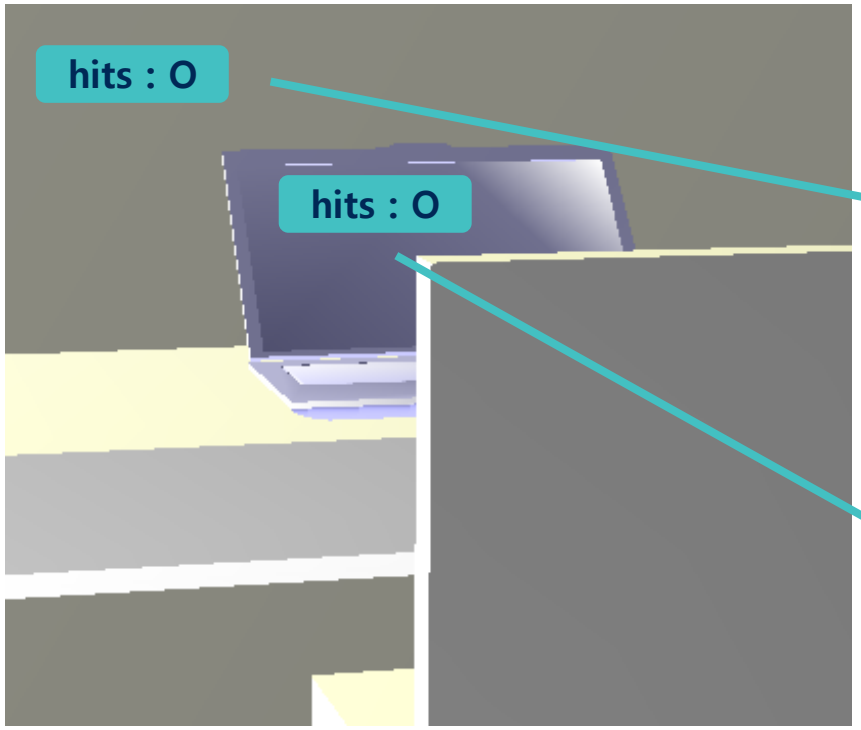
Object Select

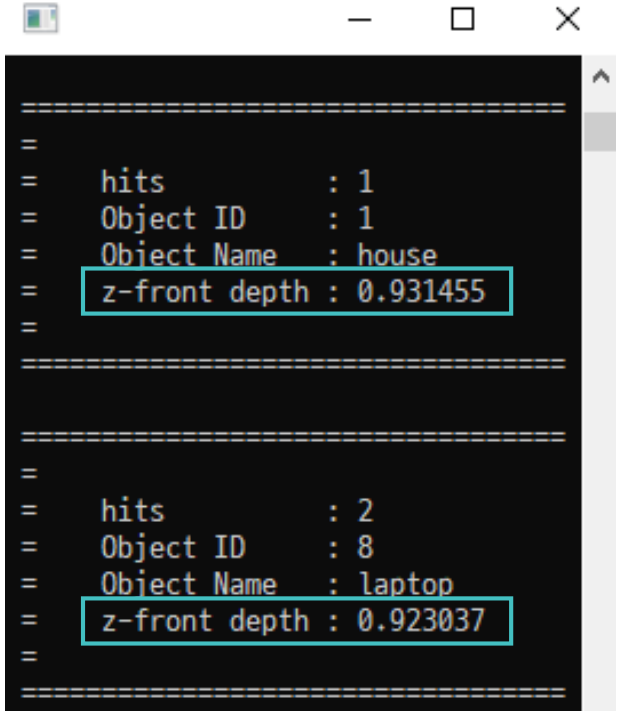
Calculate z-axis depth

processHits(hits, buffer[])

```
{
        depthBuf[objectID] = zFront - 1.0f;
    }


    for (int i = 0; i < bufferSize; i++) {
        if (zMin >= depthBuf[i]) {
            zMin = depthBuf[i];
            selectionID = i;
        }
    }

}
```

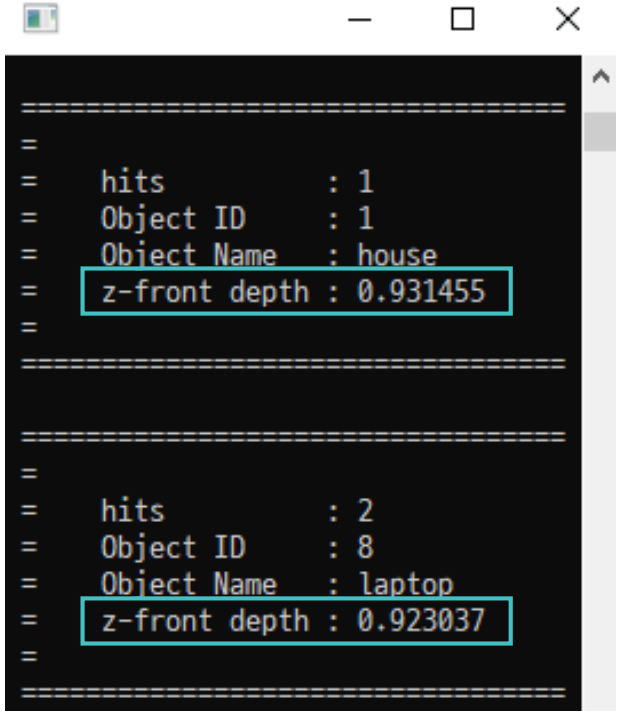
Find z-axis depth





Object Select

Calculate z-axis depth

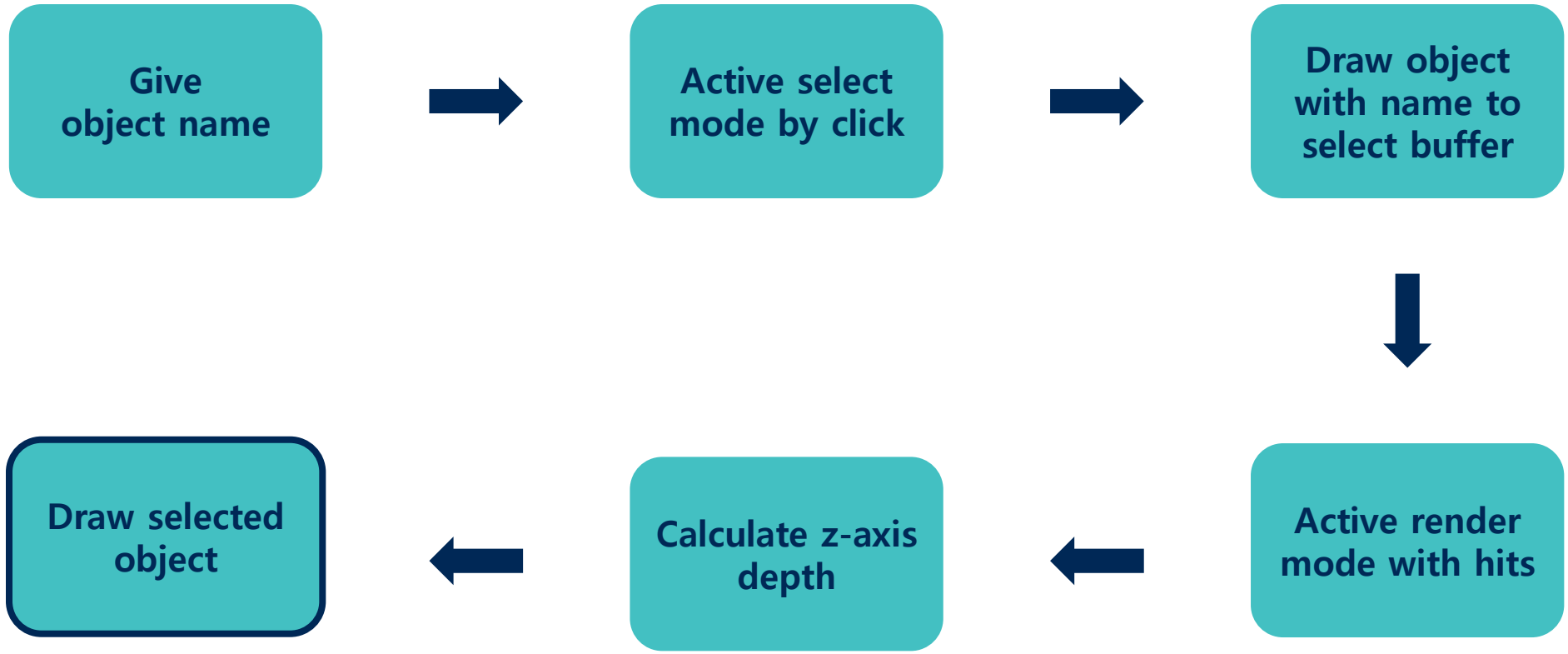


laptop z-depth < house z-depth

“laptop” is front.

Object Select

Draw selected object



Object Select

Draw selected object

updateSelection()

selectionCheck()

drawValidObjects()

```
{
  else if (obj_laptop->selection) {
      glPushMatrix();
      {
          selectionView(targetPosition, vec3(28.12, -7.70, 8.82));
          glColor3f(0.7, 0.7, 1.0);
          drawOBJ(obj_laptop);
      }
      glPopMatrix();
  }
}
```

drawValidObjects() : laptop

Demo