## **11. Displaying the 3DCG model**

The contents of sections 09. and 10. are integrated here.

In this section, polygons can finally be displayed in color.

(It lacks reality because it is colored without going through the process of 3DCG's original light source, object reflectance, and camera imaging.))

(5 weeks is short to be able to do everything

Although not specified In any section, be sure to put ic2\_DefaultModel.txt in your project directory before running it.

### **11.01. Program coexistence**

The source files in section 09.03. and 10.02. are carefully created so that they do not interfere with each other.

(If you can list a number of problems that may occur if you are not careful, you are a full-fledged programmer.)

From section 10, bring only the source file to read the file.

**[Program 11-01]**

1. ic2-CommonHeaders.h
2. ic2-ModelHeaders.h
3. 07-03-EmbededObjects.c
4. 07-04-Initialization.c
5. 07-05-MainFunction.c 08-01
6. -GLTools.c
7. 09-02-Projection. c
8. 09-03-Callback.c
9. 09-03-Rendering.c
10. 10-02-ReadModel.c

**Exercise**

**11-01-ex1**: The main function

Explain why and the source file containing main() in section 09.03 (07-05-MainFunction.c) and the source file (10-02-MainFunction.c) in section 10.02. should not be placed in the same project.

**11-01-ex2**: Exchange of the main function

In 11-01-ex1, what happens and what kind of program is generated when only 10-02-MainFunction.c is placed in the project. Report and explain the reason.

**11-01-ex3**: Making a file-read library

Make the source group brought in from section 10.02. Into a file-read library so that the source group of section 09.03. can be compiled. The new library name should be "IC2ReadModel".

### **11.02. Program integration**

Integrate two header files into one.

As for the integration of the header files, only one place needs to be changed in the program source files.

**[Program]**

1. ic2-CommonHeaders.h (Difference from 11-01)
2. 11-02-ReadModel.c (Difference from 10-02)
3. 07-03-EmbededObjects.c, 07-04-Initialization.c, 07-05-MainFunction.c, 08-01-GLTools.c, 09-02-Projection.c, 09-03-Callback.c, 09-03-Rendering.c

**Exercise**

**11-02-ex1**: Header file Integration

Indicate the rewritten part for the header integration. Explain the reason for the rewriting in detail.

### **11.03. Program full integration**

Rewrite main() to receive a model from a model file.

It is necessary to carefully compare 07-05-MainFunction.c (the source file that contains the original main() function in section 09.03.) by checking 10-02-MainFunction.c well.

(In such a case, an application named [meld](http://meldmerge.org/) may be more convenient than eclipse. It is not included in the current computer system ...)

Please read and understand the revised part in the following program.

**[Program]**

1. 11-03-MainFunction.c (Difference from 07-05)
2. 07-03-EmbededObjects.c, 07-04-Initialization.c, 08-01-GLTools.c, 09-02-Projection.c, 09-03-Callback.c, 09-03-Rendering.c, 11-02-ReadModel.c, ic2-CommonHeaders.h

**Exercise**

**11-03-ex1**: Runtime arguments

Explain about the runtime arguments of the program in this section.

**11-03-ex2**: Error traps in 11-03-MainFunction.c

Please enumerate and explain each of the error traps in 11-03-MainFunction.c and their countermeasures.

**11-03-ex3**: Meld

Prepare two text files with similar contents, start the editor Meld, and use it.

(This is an assignment only for students who are building their own environment.)

(Although it cannot be edited, users of the computer room on the 5th floor can execute the diff command on the command line to automatically extract the differences. Let's do it)

(If you want to just visualize it, you can use the [diffchecker](https://www.diffchecker.com/) site for example. It has become a convenient world.)

### **11.04. Polygon display**

Finally, prepare the function ic2\_DrawModel() that draws the loaded polygon list.

ic2\_DrawModel() is described in 11-04-FileObjects.c.

Since all polygons are triangular patches, we use the format GL\_TRIANGLES for drawing.

In event-driven programming, there is no good way to pass data etc. to the function to be called when an event occurs, so global variables are used.

Here, we make firstpatchptr as a global variable so that we can pass the firstpatchptr, which holds the patch list of the model, to ic2\_DrawModel().

**[Program 11-04]**

1. ic2-CommonHeaders.h (difference from 11-03)
2. 11-04-FileObjects.c (new)
3. 11-04-MainFunction.c (difference from 11-03)
4. 11-04-Rendering. c (Difference from 09-03)
5. 07-03-EmbededObjects.c, 07-04-Initialization.c, 08-01-GLTools.c, 09-02-Projection.c, 09-03-Callback.c, 11 -02-ReadModel.c

**Exercise**

**11-04-ex1**: External declaration

By the external declaration (extern) of the header file, answer whether it is possible to assign a value to the externally declared variable, along with the reason.

**11-04-ex2**: Drawing format

Explain the drawing format of GL\_TRIANGLES.

### **11.05. Polygon display with depth management**

We are able to draw the model! Wmmm … something is strange.

This is because the program in the previous section "overwrites polygons in the order they were drawn".

To avoid this, we use a CG method called Depth buffering.

(It is a.k.a. Z-buffer method)

(It was a tremendous shock when I first saw real-time polygon rendering being executed in hardware using the Depth buffer ... decades after that.)

The Depth buffer is the same size as the image displayed in CG. For each pixel, the value corresponding to the distance is entered instead of the RGB value.

Using Depth buffering requires a three-step procedure.

1. Assert at boot time that you plan to use the Depth buffer.

2. Initialize the Depth buffer every time you rewrite a new frame.

3. Consult the value of the Depth buffer to make a decision to draw or not on rendering the polygon.

(Refer to the Depth buffer, and if it is found that the drawing part (pixel) in the polygon comes to the front with respect to the camera, i.e. the distance is shorter than the one in the depth buffer, paint the pixel with the color of the polygon and replace the Depth value of the pixel. )

Below is the program with three changes.

**[Program]**

1. 11-05-Initialization.c (difference from 07-04)
2. 11-05-Rendering.c (difference from 11-04)
3. 07-03-EmbededObjects.c, 08-01-GLTools. c, 09-02-Projection.c, 09-03-Callback.c, 11-02-ReadModel.c, 11-04-FileObjects.c, 11-04-MainFunction.c, ic2-CommonHeaders.h

**Exercise**

**11- 05-ex1**: Depth buffering

Explain Depth buffering (Z-buffer method). In particular, explain how the values of the Depth buffer are changed.

**11-05-ex2Explain**: Depth buffer initialization

Explain how the contents of the buffer should be initialized when the Depth buffer is initialized during the execution of the program.

**11-05-ex3**: Simultaneous display of logo and 3DCG

Modify the program so that both the logo and the 3DCG from a file are rendered at the same time. The logo should be of your own. The logo may be partially buried in the 3DCG model.  
Also, at this time, replace the rotation, translation, and scaling with your original ones. (Similar to the original program, you need to control them periodically)

**11-05-ex4**: Periodic motion of 3DCG

Similar to 09-03-ex5, modify the program so that the 3DCG translates and rotates periodically. The logo should be placed still.

**11-05-ex5**: Number of polygons and system load

Use a 3DCG file with a large number of polygons to investigate the relationship between the number of polygons and the system load.

**11-05-ex6**: Investigation of a large number of polygons and system load

Prepare a model with 10,000 polygons or more and examine the system load of the program in this section.

**11-05-ex7**: One rotation instruction by keyboard

When h, j, k, l of the keyboard is pressed once, 3DCG makes one rotation to the left, one rotation to the right, one rotation to the top, and one rotation to the bottom, respectively. Rewrite the program to achieve these functions.

**11-05-ex8**: Continuous rotation instruction by keyboard

Rewrite the program so that 3DCG rotates counterclockwise, clockwise, up, and down respectively while holding down h, j, k, l on the keyboard.

**11-05-ex9**: Addition of translation instructions by keyboard

In addition to 11-05-ex8, rewrite the program so that 3DCG can be translated back and forth, left, right, up, and down while pressing the key using the cursor keys on the keyboard.

You can use the [glutSpecialFunc () function](https://www.opengl.org/resources/libraries/glut/spec3/node54.html#SECTION00089000000000000000) to set up a callback for non-standard keys.

Extra (3DCG model file)

A doll data No.02 (data/Rin.txt)

(It is a real 3DCG model with 15,561 polygons. Perhaps is it a celebrity?)

(I pulled it from a certain place, so do not make it public.)

(Because there is no texture, It doesn't have black eyes ...)

(If you have free time, please send me a polygon that corresponds to the black eyes.)