Configuration

**Note:**

All configuration files are text files, in which any content between /\* and \*/ or following // are comments. The comments have no meaning, and can be deleted from configuration files; their function is only for improving readability.

If an item in configuration file is a file name, its directory is that of the configuration file. The only exception is the three file names in configure.txt on webserver, whether their directories are absolute or relative is decided by other parameters in the same configuration file.

**Table of Contents**

[1. Configuration Graphics 2](#_Toc475120754)

[2. Files on Web Server 3](#_Toc475120755)

[3. Root Data Configuration File 4](#_Toc475120756)

[4. Shader Configuration File 4](#_Toc475120757)

[5. Render List File 5](#_Toc475120758)

[6. Part Parameter File 5](#_Toc475120759)

[7. Part List File 8](#_Toc475120760)

[8. Part Mesh File 8](#_Toc475120761)

[9. User Configuration File 10](#_Toc475120762)

[10. User Parameter File 10](#_Toc475120763)

[11. User Assembly File 10](#_Toc475120764)

[12. System Assembly File 10](#_Toc475120765)

[13. Scene Assembly File 11](#_Toc475120766)

[14. Component Assembly File 11](#_Toc475120767)

[15. Scene Parameter File 12](#_Toc475120768)

[16. Camera File 15](#_Toc475120769)

[17. Change Part File 16](#_Toc475120770)

[18. Change Component File 16](#_Toc475120771)

## Configuration Graphics

Part Parameter File

Part List File

Part User Name

Part System Name

Part Mesh File

Part Description File

Part Sound File

Part Material File

Content Type File

Shader Configuration File

User Configuration File

Shader Driver Name

Shader Render List File

Shader Programs Files

User Name

User Assembly File

User Parameter File

User Password

Interface.jsp on web server

System Assembly File

Root Data Configuration File

Proxy Root Directory

Language Change File

Assembly Name

Scene Parameter File

Scene Assembly File

Webpage Title

Change Part File

Camera File

Collector Stack File

Change Component File

Component Assembly File

Top Assemble Parameters

part

user

scene

## Files on Web Server

Configuration begins with files on Web Server, which [at least](https://www.baidu.com/link?url=1v8chQTtA89jLGOYJE0fwsAShZNRoTaH16oHi5-gXakDhVWya-MBVvdB1qYzpSNl_mey78Q2tRst47iKgP9Iv3Zo8cENWYMgLLIR4qpGvBqcYWUPBRSqnjr6zSz3cXqT&wd=&eqid=95f9e6b70004317a0000000556b5a2b7) include one JSP file: interface.jsp. This file is server-side interface between client browser and web server; client browsers call this file through Ajax for executing engine functions on web server.

The content in this file includes two files name and a directory name.

The first file name is that of **Root Data Configuration File**, from which engine can find all configuration data.

The second file name is that of **Buffer Data Configuration File**. It is a text file, with some records in it. Each record contains three fields. The first field is a sub-directory name, the second field is an URL, These two field means that data in the sub-directory is buffer data for the URL. The third field is also a substitutional URL, if requested file doesn’t exist in the sub-directory, client-side browser call the substitutional URL through Ajax for downloading that file.

The third item is a temporary directory name. Its subdirectory **Proxy Root Directory** is a directory for temporary files. These files existing within this directory can be accessed by File-Proxy-Server. To improve its performance, engine can be configured NOT providing file download services. When need download files from engine, client-browsers can download them from File-Proxy-Server? The files in **Proxy Root Directory** is these files that can be downloaded through File-Proxy-Server.

To increase safety and security, we suggest NOT deploying engine data and temporary files on web directory in which files can be downloaded from internet directly. If there is not any safety and security issue, you can also deploy all data on web server and use relative directories.

The other two temporary subdirectories are for storing compressed files. One directory is subdirectory **gzip\_data\_root\_directory** , it is for storing compressed file of **Root Data File Directory**, the other is **gzip\_proxy\_root\_directory**, it is for storing compressed file of **Proxy Root Directory**. When client browsers download files from server, they can download compressed files instead of uncompressed ones. In this way, downloading speed can be increased and downloading time-length can be shortened.

Besides the two items above, there often exists the following files in different language directories such English, Chinese, etc.

1. draw.jsp: this file is the web page file to display 3D scene.
2. display.jsp: this file is used to display engine status.
3. design.jsp: this file is used to modify parameters of parts and components in rendering scene, for example, component location, part color and texture, .etc.
4. parameter.jsp: this file is used to modify engine parameters.
5. view.jsp: this file is used to modify camera parameters.
6. tag.jsp: this file is used to modify movement tags.

## Root Data Configuration File

The content in **Root Data Configuration File** includes several lines; each line contains one parameter value. To increase readability, often each line begins with a parameter name comment between /\* and \*/. Some parameter values are file names (often with subdirectory), whose relative directory is that of **Root Data ConfigurationFile**.

The content in **Root Data Configuration File** includes:

1. file\_system\_charset: this is charset name for storage on local file system. If it is default\_charset, it will use charset on local computer.
2. server\_response\_charset: this is charset name for web server to response data to web browser. If it is default\_charset, it will use charset on local computer.
3. system\_directory\_name: name of sub-directory for storing system data, it exists directory of **Root Data ConfigurationFile.**
4. shader file\_name: **Shader Configuration File** name, from this file engine can find all configuration data about shader programs and rendered part.
5. user\_file\_name: **User Configuration File** name, from this file engine can find all configuration data about users. An important user configuration data is about assembly configuration, from which engine construct rendered scene.
6. log\_file\_name:log file name, engine output its display information into this file, its directory is **Proxy Root Directory.**
7. response\_block\_size: when web server response data to web browser, this parameter determines buffer size for reading data from file system.
8. engine\_expire\_time\_length: if a client program has not communicated with engine for a very long time, engine will treat the client program as a dead one and destroys all related data of that client program. This parameter tells engine how long the expiring time is. Its unit is nanosecond. When timeout, engine related data will be deleted.
9. engine\_effective\_time\_length: this parameter is similar to engine\_expire\_time\_length, the difference is that when timeout, engine is only treated as ineffective, only when eclipse time length exceeds engine\_expire\_time\_length, engine related data will be deleted.
10. engine\_load\_time\_length: when client side engine is running at backend, this parameter identify how long time engine do a download request.
11. engine\_touch\_time\_length: when client side engine is running at backend, this parameter identify how long time engine tells server it keeps alive.
12. file\_buffer\_expire\_time\_length: when a client browser has downloaded a file from server, it will buffer it in local file system. When it need the file again in the future, it will get it from local file system instead of from remote web server. In this way, download bandwidth can be decrease. The parameter file\_buffer\_expire\_time\_length tells client browser how long time the file will buffered in local file system. Its unit is second.
13. max\_engine\_kernel\_number: to prevent from overload, engine maintains no more than max\_engine\_kernel\_number scenes.
14. max\_component\_number: to prevent from overload, engine maintains no more than max\_component\_number components.
15. max\_request\_number: this parameter identify how many part request will be buffered.
16. file\_download\_cors\_string: cors means Cross-origin resource sharing. This parameter is cors string for file download
17. exchange\_bigend\_littleend\_flag: when creates mesh data for parts, to decrease data capacity, engine employ binary data format. The parameter exchange\_bigend\_littleend\_flag identifies whether big-end format or little-end format is employed.
18. debug\_mode\_flag: if this parameter is true, error message will be displayed through alert() function, if this parameter is false, error message will be displayed through console.log() function.

## Shader Configuration File

The content in **Shader Configuration File** is configuration of render programs. It contains some records, and each record has one render driver class name and a render list file name, whose relative directory is that of **Shader Configuration File**.

1. The first item is render driver class name. Engine uses the java class name to create a render driver object on web server.
2. The second item is **render List File** name. From this file engine can find all part-related information that engine and the driver objects exploits to render scenes.

Example (list nine record):

*driver\_audio.extended\_render\_driver driver\_general\audio\render.list*

*driver\_caption.extended\_render\_driver driver\_general\caption\render.list*

*driver\_coordinate.extended\_render\_driver driver\_general\coordinate\render.list*

*driver\_general.extended\_render\_driver driver\_general\general\render.list*

*driver\_proxy.extended\_render\_driver driver\_general\proxy\render.list*

*driver\_ruler.extended\_render\_driver driver\_general\ruler\render.list*

*driver\_sky\_box.extended\_render\_driver driver\_general\sky\_box\render.list*

*driver\_tag.extended\_render\_driver driver\_general\tag\render.list*

*driver\_text.extended\_render\_driver driver\_general\text\render.list*

## Render List File

The content in **Render List file** are some records, and each record has two file names. Their directory is that of **Render List file.**

Sometime there may exist user-defined parameters following the two file names. The number of user-defined parameters is defined by render driver’s get\_parameter method.

The first file name is **System Render Parameter File**, which contains system parameters for render program.

The second file name is **Part List File**, which contains all part-related information.

Example (only list one record):

*parameter.list part.list*

## Part Parameter File

The content in **Part Parameter File** includes several lines; each line contains one parameter value. To increase readability, often each line begins with a parameter name comment between /\* and \*/.

The content in **Part Parameter File** includes:

1. simple\_mesh\_file\_name:
2. buffer\_object\_file\_name: these two parameters identify simple mesh file name and buffer object file name, Each part has a mesh file; it contains all geometry data about the part. The difference between mesh file and simple mesh file is that a mesh file contains part vertex data, while a simple mesh file doesn’t. Engine creates buffer object file from the part mesh file; client browser downloads the buffer object file, and uses it to render component. If buffer object file has been created, engine can just load part geometry data from simple mesh file, and do not need loading part vertex data. In this way, engine can start much faster.
3. audio\_file\_name: in this file stores part audio description for different language.
4. process\_sequence\_id: a scene contains many parts. Which one is processed first? Which one is processed last? This parameter identifies part processing sequence. The smaller is this parameter, the earlier process is processed. If two parts has same process sequence id, the bigger is the part volume, the earlier part is processed. If two parts also has same part volume, the part that has fewer vertices is processed first.
5. max\_buffer\_object\_data\_length：when buffer-object-files are downloaded from web-server, they are combined together and stored in some WebGL buffer objects on GPU. This parameter determines how much data are stored in a buffer object.
6. scale\_value：part geometry may be defined by other software. Different software may use different unit of measurement. Some may use meter, others may uses millimeter. The parameter scale\_value is used to ajust part geometry data.
7. assembly\_precision: this parameter is used for lod (level-of-detail); it identifies whether display an assembly component or display its descendant’s components in a component tree.
8. discard\_precision: this parameter is also used for lod (level-of-detail); it identifies whether display a component or discard its display.
9. bottom\_box\_discard\_precision: this parameter is also used for lod (level-of-detail); it identifies the discard\_precision parameter of its bottom box component.
10. create\_buffer\_object\_bitmap: buffer object file has three geometry data: face, edge and point. Often it is unnecessary to create all of them. The create\_buffer\_object\_bitmap tells engine which geometry data it should create.
11. max\_buffer\_data\_number: graphics engine can be divided into two parts: server–side engine runs on webserver, while and client-side engine runs in browser. Component render data on both sides should be consistent. If data at server-side is modified, server-side engine will transfer the modified data to client-side engine in browser. Client-side engine cache several updating data packets from server. The parameter max\_buffer\_data\_number identifies how many updating data packet are stored on client-side engine. This multi-cache mechanism help implementing gradual data updating, preventing from transferring whole rendering data from web server.
12. max\_part\_loader\_thread\_number: To improve part loading performance, engine create several threads to load part data fast. The parameter max\_part\_loader\_thread\_number tells engine how many threads to create for part data loading.
13. max\_mesh\_load\_number: This parameter identifies how many files can be downloaded synchronously from server.
14. do\_create\_bottom\_box\_flag: if a component is very far away to camera, when engine displays it on the screen, it is unnecessary to display the component in details; sometimes it is enough to just display its box. This parameter tells engine whether or not to create a part of bottom box.
15. delete\_redundant\_data\_flag: this parameter identifies whether or not deletes redundant data when loading part vertex data.
16. combine\_data\_flag: this parameter identifies whether or not combines part vertex data when loading them.
17. create\_normal\_flag: this flag identifies whether or not creating mesh normal data when load part mesh file.
18. only\_free\_vertex\_memory\_flag: When part data has been loaded into memory and its buffer object files has been created, some part data can be released to save memory space. If only\_free\_vertex\_memory\_flag parameter is true, only vertex data is released, most geometry information is still stored in memory. If only\_free\_vertex\_memory\_flag parameter is false, all vertex data is released, no geometry information is released in memory.
19. create\_mesh\_comment\_flag: when create mesh file, this parameter identifies whether or not create the comment.

## Part List File

The content in **Part List File** is some records, each record has six items. The first item is part user name. Engine uses part user name when it displays part information. The second item is part system name. When engine creates a scene, it uses part system name to set up relationship between part and component. The third item is **Part Mesh File** name; the content in this file defines the part’s geometry. The fourth item is part material file name; the content in this file is defined by part driver. The fifth item is part description file name; this text file contains part information. The sixth item is part audio file name. Engine may play this audio file when user does some part-related operation.

Sometime there may exist user-defined parameters following the six items. The number of user-defined parameters is defined by render driver’s create\_part\_driver() method or part driver’s extended\_part\_driver() method.

Example (only list one record, no user-defined parameter):

*background*

*background*

*background\background.mesh*

*background\background.material*

*background\background.description*

*background\background.ogg*

## Part Mesh File

A part mesh consists of several bodies. The second parameter is body number. Following body number are definitions of all bodies.

A body consists of several faces. The first parameter in body definition is body name. The second parameter is face number of that body. Following face number are definitions of all faces.

A face consists of face facet and face loop. The first parameter in face definition is face name. The second parameter is face type. The third parameter is face surface parameter number, with 0 or more parameter values. Following these are definitions of face facet and face loop.

The first parameter in face facet is face attribute number. Second part of face facet definition is face vertex definition. Third part of face facet definition is face normal definition. If face attribute number is greater than zero, Next part of face facet definition are definitions of face attribute vertex. Last part of face facet definition is definition of face primitive.

The definitions of face vertex, normal, and attribute are similar. First is vertex number, after that is vertex definitions. A vertex has 4 floating numbers, 3 floating number is vertex position, 1 floating number is vertex extra-data.

The first parameter in face primitive definition is primitive number, after that are primitive definitions.

The first parameter in primitive definition is material definition, it has 4 floating numbers. The second parameter in primitive definition is vertex indexes. Vertex indexes are integer numbers; they are indexes into face vertex. The third parameter in primitive definition is normal indexes. Normal indexes are also integer numbers; they are indexes into normal vertex. If part has face attributes, the definitions of all attribute indexes will follow normal index definition. The vertex indexes, normal indexes, and attribute indexes are all terminated with a integer number that is less than zero.

The first parameter in face loop is face loop number. Following face loop number are definitions of all face loops.

The first parameter in each face loop is face edge number. The second parameter is definitions of curve parameter number and curve parameter values. The third parameter is curve material definition (two floating number). The forth parameter defines edge start point. If it is start\_not\_effective, the edge has no start point; otherwise it is start\_ effective, with six floating numbers following it, three floating numbers define edge start point position, one floating number is edge start point extra data, two floating numbers define edge start point material. The fifth parameter defines edge end point. If it is end\_not\_effective, the edge has no end point; otherwise it is end\_effective, with six floating numbers following it, three floating numbers define edge end point position, one floating numbers define edge end point extra data, two floating numbers define edge end point material. The fifth parameter defines edge tessellation number. The last is tessellation definitions. One definition has six floating number, three floating numbers define tessellation position, one numbers define tessellation extra data, two floating numbers define tessellation material.

## User Configuration File

The content in **User Configuration File** are some records, and each record has four items. The first item is user name. The second item is user password. These two items are used for user authentication. The third item is **User Assembly File** name, the file content is **System Assembly File** names. The fourth item is **User Parameter File** name; the file content defines all sorts of user-related parameter. The relative directory of the third and fourth items is that of **User Configuration File**.

Example (only list one record):

NoName NoPassword NoName/assembly.txt NoName/parameter.txt

## User Parameter File

The first item in **User Parameter File** is user\_max\_component\_number, its identifies how many component a user can create. The second item is user\_max\_engine\_kernel\_number, it identifies how many scene a user can create. The third parameter is time\_increase\_step, it identifies user-tick-advance speed.

The fourth items are max\_time\_step, time\_increase\_step\* max\_time\_step identifies max-time-length.

The last items are delay\_time\_length,it tells client-browser how long to delay before they do rendering request.

## User Assembly File

The content in **User Assembly File** is System Assembly File names, these files are used when engine creates scenes.

## System Assembly File

The content in System Assembly File is some records, each record has six items. The first item is assembly name, user uses assembly name to identify assemblies to create scene. The second item is webpage title. The third item is **Scene Assembly File** name, whose content defines how to construct a scene. The fourth and fifth items is two ratio values for creating top assemble-part. If they are zero, No top assemble-part is created. The sixth item is **scene parameter file** name; it defines some scene-related parameters.

Example (only list one record):

my\_assembly\_name my\_webpage\_title assembly/part/assembly.txt 0 0 parameter.txt

## Scene Assembly File

The content in Scene Assembly File is some records, each record has two items. The first item is a Boolean value (yes or no, true or false). When engine create a component list, the first item identifies whether or not components are shown in component list. The second item is **Component Assembly File**, whose content defines components that exist in scene, and where the components are located in coordinate systems.

Example:

no camera.SLDASM.assemble

no movement.assemble

yes SLDASM.assemble

## Component Assembly File

The content in a **Component Assembly File** is a component tree. The first record in **Scene Assembly File** defines component root tree, the other records define component sub tree whose parent is component root. All component trees that are defined in **Scene Assembly File** constitute whole scene.

A component tree is made up of component nodes; some nodes are sub trees (they have children); other nodes are leaves (they have no child).

The first item in a node is **component name**, whose function is to identify a component.

The second item in a node is **part name**; it identifies which part data this component exploits to render it.

The third item in a node is **component location**. It is a 4\*4 matrix (16 floating number). The component location identifies relative location to its parent node.

Following third item is 0 or more parameter. Part driver decide how many parameters exist here. Some part drivers need no parameter, while others may need many parameter.

The last item is **child number**. If this node is a leaf, this item is zero. If this item bigger than zero, it means this node is a sub tree and following child number is the definition of the sub tree.

Example:

example\_assembly

no\_part

1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1

3

system\_information\_component

system\_information\_part

1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1

0

manipulator\_component

manipulator\_part

1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1

0

camera\_operation\_component

camera\_operation\_part

1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1

0

## Scene Parameter File

The content in **Scene Parameter File** includes several lines; each line contains one parameter value. To increase readability, often each line begins with a parameter name comment between /\* and \*/.

Engine uses the parameter values to construct scene.

The content in **Scene Parameter File** includes:

1. proxy\_directory\_name: Each scene has a scene directory. Some scenes directories canbe grouped into subdirectory. Parameter proxy\_directory\_name is the name of the subdirectory.
2. shader\_file\_name: This item is two file names, one is file name of **Shader Configuration File** for the scene, the other is file name of **Shader Configuration File** for the scene group. Their directory is that of scene or that of scene group. The content is same as that in **Shader Configuration File**.
3. camera\_file\_name: **Camera File** Name. When engine construct a scene, it defines some cameras. The parameters of these cameras are defined in **Camera File.**
4. change\_part\_file\_name: **Change Part File** Name. The content inthis file is string pairs for part searching. Engine searches part objects by part name. It first changes searching name by content in **Change Part File**, and then uses the changed search name for part object search operation.
5. change\_component\_file\_name: **Change Component File** Name. The content in this file is string pairs for component searching. Engine searches component objects by component name. It first changes searching name by content in **Change Component File**, and then uses the changed search name for component object search operation.
6. initialization\_file\_name: File name of JavaScript initialization function for scene. When scene has been created at client-size browser, engine executes functions in this file to initialize the scene. User can store some JavaScript code in this file for initialization. There are three initialization files, the first for each scene type, the second for each scene, the third for file name in proxy directory of each scene.
7. scene\_cors\_string:
8. multiparameter\_number: a component can have several render parameter. This parameter identifies number of render parameter.
9. initial\_parameter\_channel\_id: this parameter identifies which render parameter is employed for rendering component.
10. default\_display\_bitmap：When scene is created, this parameter identifies default display bitmap of scene components.
11. component\_collector\_stack\_file\_name: engine maintains a component collector stack; stack information can be saved in a file, or restored from that file. This parameter is name of that file.
12. component\_collector\_parameter\_channel\_id: This parameter tells engine which render parameter is employed for these components in collector stack. This parameter are integers, the first is number of render parameter, following IDs of render parameter.
13. max\_component\_collector\_number: engine maintains a component collector stack for every scene, this parameter tells engine maximum of component collectors in the stack.
14. max\_camera\_return\_stack\_number : to provide camera rollback mechanism, engine maintains a camera parameter stack for every scene. Once modify camera parameter, last camera parameter is push into camera parameter stack. When do camera rollback, camera parameter is pop from the stack. This parameter tells engine maximum of camera parameter stack items.
15. max\_modifier\_container\_number: engine manages several modifier containers. This parameter identifiers how many modifier containers the engine manages.
16. create\_top\_part\_assembly\_precision:
17. create\_top\_part\_discard\_precision:
18. discard\_top\_part\_component\_precision: The three parameters above define how to create top assembly parts. create\_top\_part\_assembly\_precision is assembly precision of the created top assembly parts. create\_top\_part\_discard\_precision is discard precision of the created top assembly parts. discard\_top\_part\_component\_precision decides which components is discard when create assembly parts.
19. touch\_time\_length: If component parameters are modified on web server, engine may delay updating parameter cache on client browser. If a component is touched, it means parameter cache on client browser should update immediately if its parameter is modified. If a component is not touched, it means updating parameter cache on client browser can be delayed. The parameter touch\_time\_length defines whether a component is touched. If (component touch time + touch\_time\_length)>=current time, it means that the component is touched; If (component touch time + touch\_time\_length)<current time, it means that the component is not touched.
20. most\_delete\_keep\_number:
21. most\_append\_number:
22. most\_update\_parameter\_number:
23. most\_update\_location\_number: The four parameters above limit updating parameter cache on client browser. Parameter most\_delete\_keep\_number defines the most deleting component number from a scene. Parameter most\_append\_number define the most appending component number in a scene. Parameter most\_update\_parameter\_number defines the most updating number of component render parameter. Parameter most\_update\_location\_number defines the most updating number of component location parameter.
24. display\_precision: Computer precision is usually very high; sometimes it’s unnecessary to exploit a too high precision. For example, 1.0 and 0.999999999999999 has usually no visible different effect. The display\_precision parameter tells engine displaying precision when it displays some numerical values.
25. display\_assemble\_depth: When creates component tree webpage, rather than display all component nodes, engine only displays several levels of component tree from current component node. The parameter display\_assemble\_depth defines how depth in component tree to display in the component tree webpage.
26. component\_sort\_type: When creates component list, engine should first do component sort. Parameter component\_sort\_type defines how to sort component, its valus is “xyz”, “xzy”, “yxz”, “yzx”, “zxy”,or “zyx”. For exampes, “xyz” means first sort components according to component X coordinates, the Y coordinates, and then Z coordinates.
27. component\_sort\_min\_distance: When engine does component sort, if distance of two components is very small in X, Y, or Z direction, engine treats their coordinates as the same. Parameter component\_sort\_min\_distance defines the minimum value when doing coordinate comparison.
28. proxy\_response\_length: user JavaScript program at browser side can downloaded scene data from proxy server to increase data download speed. When response data length is greater than this parameter, user JavaScript program at browser side will download data from proxy. Whereas when response data length is less than this parameter, it will download data from server directly.
29. gzip\_response\_length: When a web server responses any data to client browser, this parameter decides whether or not the responses data is compressed before data transferring. Transferring compressed data can reduce network bandwidth but increases transferring delay, whereas transferring uncompressed data can decrease transferring delay, but need more network bandwidth. When response data length is greater than this parameter, engine will transfer gzip data. Whereas When response data length is less than this parameter, engine will transfer original data directly.
30. not\_do\_ancestor\_render\_flag:
31. test\_display\_assembly\_flag : If a component has descendants and drivers, and there have some problems when its descendants are displayed. The problems may be solved by displaying the component. The parameter not\_do\_ancestor\_render\_flag identifies whether or not solving the problem in this way. The parameter test\_display\_assembly\_flag identifies whether or not ignoring component’s display\_assembly\_flag.
32. do\_discard\_lod\_flag:
33. do\_selection\_lod\_flag: these are two lod flags. When do\_discard\_lod\_flag is no, engine will do discard operation when implementing lod. When do\_ selection\_lod\_flag is no, engine will select a suitable driver to display component when implementing lod.

## Camera File

The content in **Camera File** are some records, each record defines one camera. A camera record has 13 lines defining 13 parameters of that camera. To increase readability, often each line begins with a parameter name comment between /\* and \*/.

The content in **Camera File** includes:

Example:

1. switch\_time: When engine switches camera location, the parameter switch\_time defines how long time engine need to complete location modification.
2. distance: camera location is defined upon camera component. The coordinate system of a camera is that of camera component. Camera center point is coordinate origin. Camera eye point is the point located at [0, 0, distance]. The parameter distance defines how far away camera eye point from center point.
3. fovy: fovy parameter of camera.
4. near: this parameter defines how far away the near plane of camera to camera eye point, its value is that near parameter multiply by distance parameter.
5. far: this parameter defines how far away the far plane of camera to camera eye point, its value is that far parameter multiply by distance parameter.
6. low\_precision\_scale:
7. high\_precision\_scale: the two parameters above are lod(level of detail) parameters. When mouse button is pressed, engine uses low\_precision\_scale parameter; when mouse button is released, engine uses high\_precision\_scale parameter.
8. eye\_location\_precision: When the location of camera component is modified and should response the modification to client browser, this parameter is used to reduce the size of modification data. Computer precision is usually very high; sometimes it’s unnecessary to exploit a too high precision. For example, 1.0 and 0.999999999999999 has usually no visible different effect. The eye\_location\_precision parameter tells engine data precision when engine responses the modification data.
9. frustrum: This parameter is a Boolean value, it defines camera type. If it is true, the camera is a perspective projection camera; If it is false, the camera is a orthogonal projection camera.
10. movement:
11. direction:
12. change\_type\_flag：the three parameters above are Boolean values, they define how to modify camera. Parameter movement defines whether or not doing camera location modification. Direction defines whether or not doing camera direction modification. When modify view size, we can change camera distance parameter, we can also change camera fovy parameter. change\_type\_flag parameter defines which modification(distance or fovy).
13. scale: When focuses camera on some components, this parameter defines view size of the components.
14. name: this parameter is camera component name.

## Change Part File

The content in **Change Part File** is string pairs for part searching. Engine searches part objects by part name. It first changes searching name by content in **Change Part File**, and then uses the changed search name for part object search operation.

## Change Component File

The content in **Change Component File** is string pairs for component searching. Engine searches component objects by component name. It first changes searching name by content in **Change Component File**, and then uses the changed search name for component object search operation.