**A. Recommendations for the entire training**

**I. Training materials:**

- <https://docs.gameloft.org/3d-training/>

- the DirectX book (it explains the 3D concepts better than the below OpenGL ES book)  
- the OpenGL ES book.

These 2 books are in the archive:

<https://docs.gameloft.org/3d-training/#NewTraining_27_11_2012>

Other links:

*OpenGL ES:*

<http://www.khronos.org/opengles/sdk/docs/man/>

*GLSL*:

<http://www.opengl.org/sdk/docs/manglsl/>

*GLSL ES:*

<http://www.khronos.org/registry/gles/specs/2.0/GLSL_ES_Specification_1.0.17.pdf>

**II. Recommendations for implementing the tasks:**

1. You will start by studying the Readme.doc from the archive.

2. The application is simple enough to justify a simple structure. A simpler structure means the code is easier to understand and maintain. Please avoid complicating the application by many derivations.

3. For this application, you don’t need to use STL (you will see that geometry files were designed so that you can use directly simple arrays).

*Note:* In order to achieve the best performances and the best memory management you should use the simplest possible data structures which will allow you to complete your tasks. Please beware that STL classes have 2 big issues when used in games: the biggest issue is that they perform intensive allocations and deallocations for most of the operations, most of the time for very small buffers. This can lead to performance issues (the time needed by the memory manager to allocate or deallocate a buffer increases as the number of the allocated buffers increase) and to memory shortage due to memory fragmentation. The other issue is also performance related: using a simple data structure is usually faster than using complex ones. Another reason to avoid using the STL classes is that the memory manager cannot indicate correctly (i.e. the correct file and line) the memory leaks inside those classes.

4. Please keep the current settings of the framework application (it was designed to help you to identify memory leaks).

5. All the memory leaks reported by the application must be solved.

6. All the warnings must be solved too because they could hide some logic errors.

**B. Your first tasks:**

1. You will create a class Global where you will put all your global variables.
2. In the New Training Framework application

a) Add color to the triangle (a different color will be assigned to each vertex, obtain a gradient effect);

**Notes and tips:**

For this task, please study first:

<https://docs.gameloft.org/3d-training/#Introduction>

<https://docs.gameloft.org/3d-training/#The_rendering_pipeline>

b) Make the triangle to rotate per frame;

**Notes and tips:**

1. For this task, you need to study:

<https://docs.gameloft.org/3d-training/#Math_Prerequisites>

2. Compute and use world matrix for your triangle using math classes from Utilities library.

c) Add Camera class and implement methods to allow moving camera (keys W, A, S, D) and camera's rotation UP/ DOWN, LEFT/RIGHT (arrow keys);

**Notes and tips:**

1. Input

- you can make an InputManager (as singleton) or just declare a global variable. Regardless of the chosen method, you will have a variable which keeps on bits the keys.

- you can find the singleton design pattern explained here:

<https://docs.gameloft.org/wp-content/uploads/2012/04/SingletonSample1.txt>

- pay attention to variable bIsPressed from function Key() and set the bits for keys by taking bIsPressed into account.

2. For this task, you need to study:

<https://docs.gameloft.org/3d-training/#Math_Prerequisites>

<https://docs.gameloft.org/3d-training/#3_How_to_implement_the_camera_in_your_application>

3. If this is the first time you implement a camera, please choose “Simple approach” for camera presented on the site.

d) Load an .nfg file (geometry file from the resources package) and display (from scratch);

**Notes and tips:**

1. Create a class Model. Use also buffer objects (both vertex and index buffer objects). Keep in your class as data member only the information you really need, to avoid wasting memory (for example: you will notice that you just need to keep the handles to your buffer objects, but not the heap buffers you use only temporary to read data from geometry file)

2. For reading the geometry files, it’s simpler to use fscanf\_s.

e) Load and display a texture.

**Notes and tips:**

1. Please study first:

<https://docs.gameloft.org/3d-training/#Texturing>

2. Create a Texture class (please pay attention again to avoid adding data members you don’t really need to keep)

3. Use LoadTGA() function from Utilities library. Don’t forget to take into account bpp.

4. Activate also the mipmaps

5. Enable depth test.