DESCRIPTION OF EXAMPLES 2 TO 6

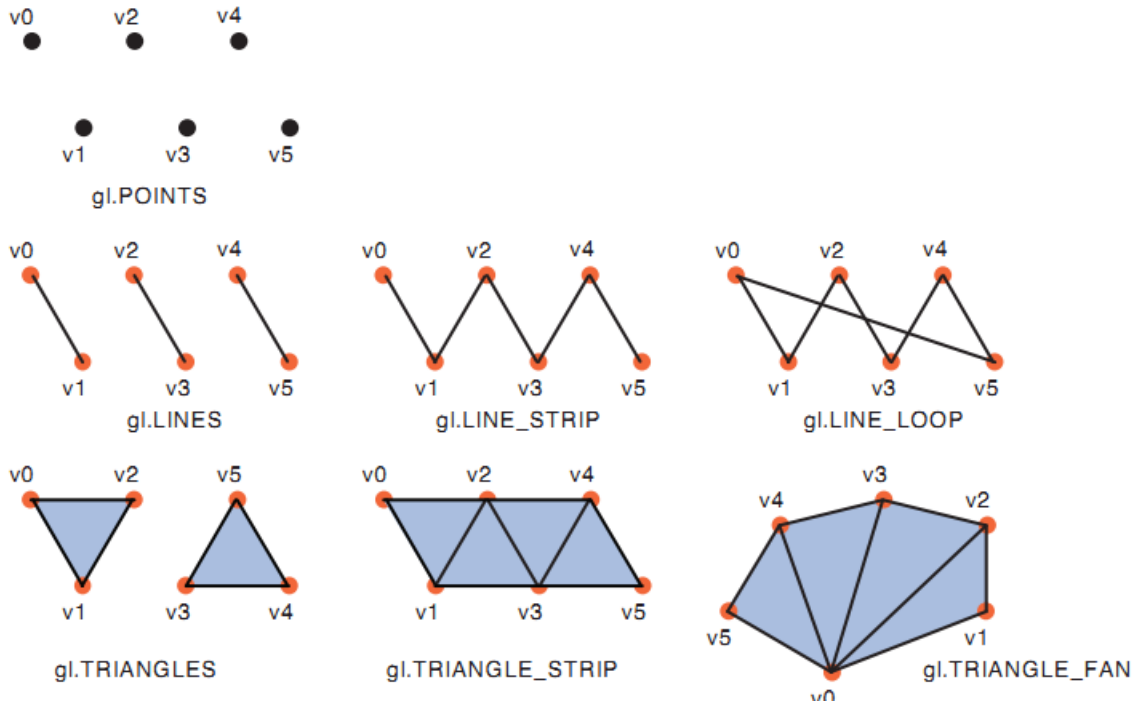
I - Example 2

In this example the student can learn:

1) How to use VBOs (Vertex Buffer Objects) to feed vertex data (coordinates and color) to the vertex shader of a WebGL program.
2) How to interlace position and color vertex data in a single VBO, and how to discriminate this data when accessing it.
3) How to use two different arrays to store separately position and color vertex data, and how to use two different VBOs to access this data from the shader.
4) How to store data from different meshes in a single array/VBO, and how to use offsets and count of vertices in *gl.drawArrays()* to choose which of them will be rendered.
5) How to use two different VBOs to hold two different meshes, and how to switch between them for render.
6) How to use the viewport to control the 2D area, within the canvas, to which the output will be rendered.
7) The effect of changing the render mode parameter of gl.drawArrays().

What to do with example 2?

1) Understand how strides and offsets are defined in *vertexAttribPointer* to access the right data. See what happens when stride size is changed.
2) Check how *switch_set()* changes the VBO which feeds *point_position* and *point_color*.
3) Use *viewport* to change position and size of rendering area.
4) Change the mode of *gl.drawArrays* to each one of its possible values (LINES, LINE_STRIP, LINE_LOOP, TRIANGLES, TRIANGLE_STRIP, TRIANGLE_FAN). See and interpret the effect.



II – Example 3

In this example the student can learn:

1) How to use a triangle strip to reduce the number of vertices required to describe a mesh.
2) How the vertex shader can be used to modify in the GPU the position of vertices in the mesh, thus avoiding the need to do these modifications with the CPU.

What to do with example 3?

1) Understand the definition of the triangle strip.
2) Check how the mesh is accumulatively deformed by the vertex shader when *Redraw* is pressed. Note that the total deformation must be kept in the JavaScript program.
3) Change the draw mode to LINE_STRIP, and interpret the output.

III – Example 4

In this example the student can learn:

1) How to use uniform variables to send control information or transformation matrices to the vertex shader.
2) How to use transformation matrices to rotate, translate or scale the vertices.
3) How to compose all these transformations into a single matrix, the Model matrix.
4) A way to draw separately parts of the scene with different properties (in this case, a couple of coordinate axis, which must not be modified by the transformations applied to the figure)

What to do with example 4?

1) Understand clearly how the transformations are being applied.
2) Understand how and when is each transformation matrix computed, used and applied. Understand well the order in which transforms are being applied.

IV – Example 5

In this example the student can learn:

1) How to draw a scene in 3D space.
2) How to change the eye position and orientation, and how to obtain and use the View transformation matrix.
3) How to use projections (orthographic or perspective) to define the viewing volume, and how to obtain and use their respective transformation matrices.
4) How to use the Z-buffer to get the objects correctly rendered according to the depth (Z-coordinate) of their vertices.

What to do with example 5?

1) Understand well how is the final transformation matrix calculated and applied. Understand well the order in which the individual transformations are applied.
2) Understand well the definition and use of the clipping planes in the viewing volume, in particular of the near and far planes.
3) Experiment with the values of camera position and orientation, and with the dimensions of the clipping planes, to see its effect.

V – Example 6

In this example the student can learn:

1) How to reduce the amount of memory required by vertex data in complex meshes, by using a vertex array and an index array, which describes the geometry referencing by index the sequence of vertices which form the GL primitives to render.
2) How these two arrays can be bound to VBOs, and how to use *gl.drawElements()* to request a render in which the data fed to the shader from the vertex VBO is read in the order given by the index VBOs.

What to do with example 6?

1) Thoroughly understand the indexing mechanisms.
2) Change the indexing order to reverse order in which the triangles are rendered, and check the output.
3) Disable the Z-buffer check and repeat reversing the rendering order. Understand the behavior of a “painter algorithm”, and the role of the Z-buffer.