# Scalable Rendering for Graphics and Game Engines

Antonio Chica Calaf
achica@cs.upc.edu

Marc Comino Trinidad
mcomino@cs.upc.edu

## LEGACY OPENGL

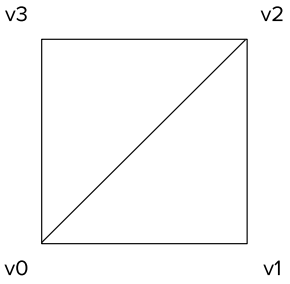
### CPU

```
// Render Time
glBegin(GL_TRIANGLES);
   glVertex3fv(v0);
   glVertex3fv(v1);
   glVertex3fv(v2);

   glVertex3fv(v0);
   glVertex3fv(v2);
   glVertex3fv(v3);
glEnd();
```

### GPU

// Render Time
Receives 9 x F floats.




## VERTEX ARRAYS

### CPU

```
// Render Time
GLfloat vertices[] = {v0, v1, v2, v3};
GLubyte indices[] = {0,1,2,0,2,3}.

glEnableClientState(GL_VERTEX_ARRAY);
glVertexPointer(3, GL_FLOAT, 0, vertices);

glDrawElements(GL_TRIANGLES, 6,
GL_UNSIGNED_BYTE, indices);

glDisableClientState(GL_VERTEX_ARRAY);
```

### GPU

// Render Time
Receives 3 x V floats and 3 x F integers.

| VERTEX BUFFER OBJECTS | |
|---|---|
| **CPU** | **GPU** |
| // Initialization Time<br>GLuint vbo_v_id, vbo_n_id, faces_id;<br><br>glGenBuffers(1, &vbo_v_id);<br>glBindBuffer(GL_ARRAY_BUFFER, vbo_v_id);<br>glBufferData(GL_ARRAY_BUFFER, dataSize, vertices, GL_STATIC_DRAW);<br><br>glGenBuffers(1, &vbo_n_id);<br>glBindBuffer(GL_ARRAY_BUFFER, vbo_n_id);<br>glBufferData(GL_ARRAY_BUFFER, dataSize, normals, GL_STATIC_DRAW);<br><br>glGenBuffers(1, &faces_id);<br>glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, faces_id);<br>glBufferData(GL_ELEMENT_ARRAY_BUFFER, dataSize, indices, GL_STATIC_DRAW);<br><br>// Render Time<br>glBindBuffer(GL_ARRAY_BUFFER, vbo_v_id);<br>glVertexPointer(3, GL_FLOAT, 0, 0);<br>glEnableClientState(GL_VERTEX_ARRAY);<br><br>glBindBuffer(GL_ARRAY_BUFFER, vbo_n_id);<br>glNormalPointer(3, GL_FLOAT, 0, 0);<br>glEnableClientState(GL_NORMAL_ARRAY);<br><br>glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, faces_id)<br><br>glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_BYTE, 0);<br><br>glDisableClientState(GL_VERTEX_ARRAY);<br>glDisableClientState(GL_NORMAL_ARRAY);<br><br>glBindBuffer(GL_ARRAY_BUFFER, 0);<br>glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 0); | // Initialization Time<br>Receives and **stores** 3 x V floats and 3 x F integers..<br>Optionally receives and **stores** 3 x V for each additional attribute. |

| VERTEX ARRAY OBJECTS | |
|---|---|
| **CPU** | **GPU** |
| // Initialization Time<br>GLfloat data[] = {v0, n0, v1, n1, v2, n2, v3, n3};<br>GLuint vbo_id;<br><br>glGenBuffers(1, &vbo_id);<br>glBindBuffer(GL_ARRAY_BUFFER, vbo_v_id);<br>glBufferData(GL_ARRAY_BUFFER, dataSize, data, GL_STATIC_DRAW);<br><br>GLuint vao_id;<br>glGenVertexArrays(1, &vao_id_);<br>glBindVertexArray(vao_id_);<br>glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, dataSize, 0, 0);<br>glEnableVertexAttribArray(0);<br><br>glBindVertexArray(vao_id_);<br>glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, dataSize, stride, offset);<br>glEnableVertexAttribArray(1);<br><br>glBindVertexArray(0);<br>glBindBuffer(GL_ARRAY_BUFFER, vbo_v_id);<br><br>// Render Time<br>glBindVertexArray(vao_id_);<br>glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, faces_id)<br><br>glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_BYTE, 0);<br><br>glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 0);<br>glBindVertexArray(GL_ARRAY_BUFFER, 0); | // Initialization Time<br>Receives and **stores** 3 x V floats and 3 x F integers..<br>Optionally receives and **stores** 3 x V for each additional attribute. |