Vertex Arrays (VAs)
Vertex Buffer Objects (VBOs),
Vertex Array Objects (VAOs)

C. Andujar, A. Vinacua

Nov 2015

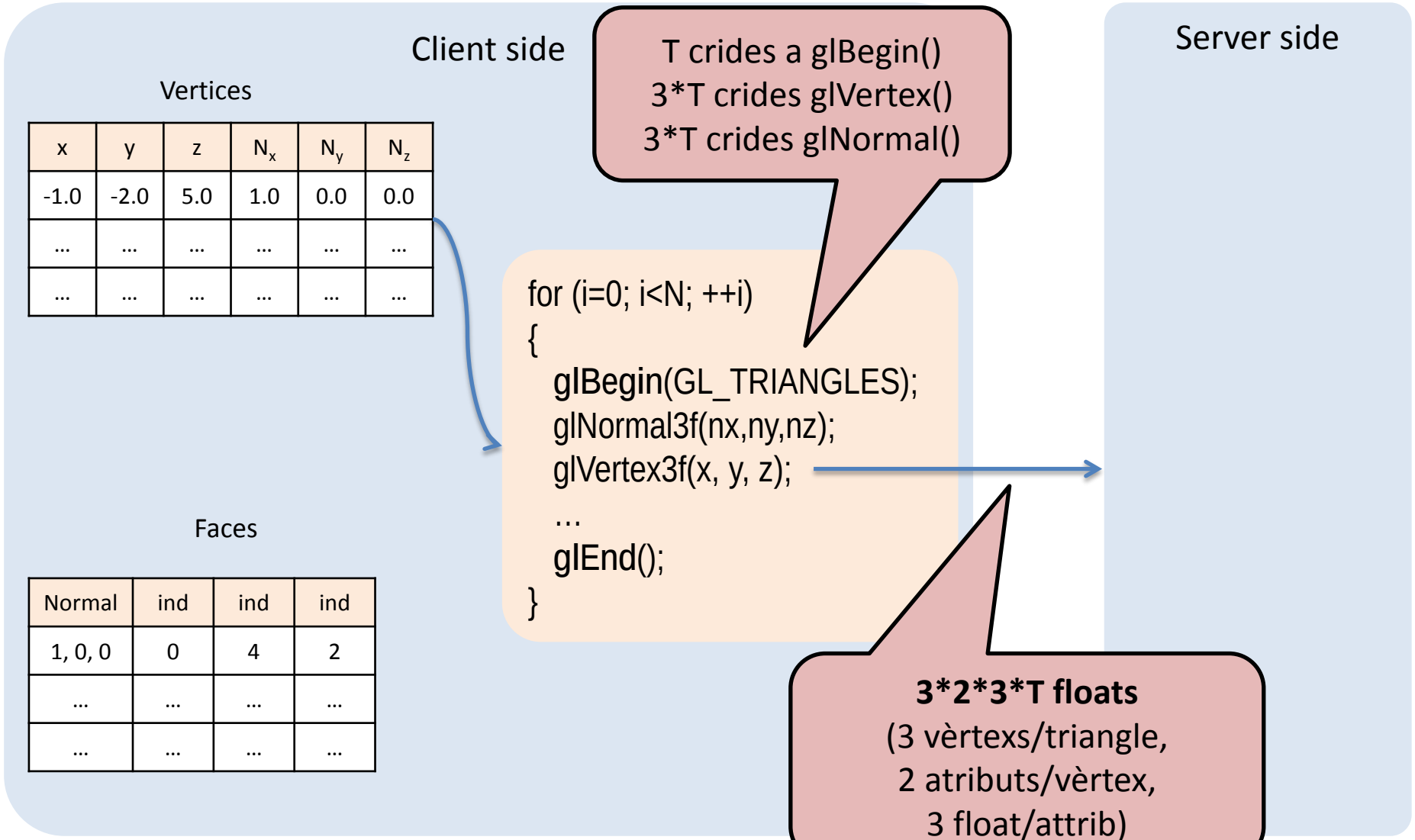
Formes de pintar geometria

- Mode immediat (glBegin,glEnd) Compatibility
- Usant Vertex Arrays (VAs) Core & Compatibility
- Usant Vertex Buffer Objects (VBOs) Core & Compatibility

Mode immediat

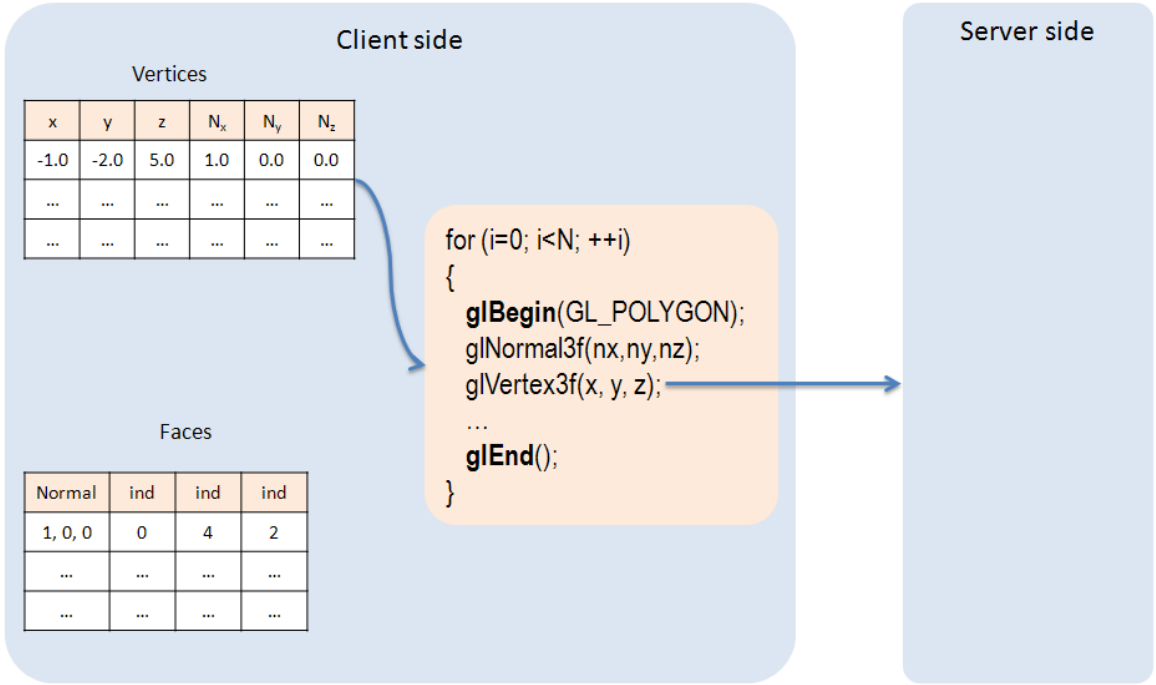
```
for (i=0; i<T; ++i) {
    glBegin(GL_TRIANGLES);
    glNormal3f(…);
    glVertex3f(…);

    glNormal3f(…);
    glVertex3f(…);

    glNormal3f(…);
    glVertex3f(…);
    glEnd();
}
```

Mode immediat



Mode immediat

- Senzill, fàcil de depurar, flexible...
- Moltes crides a funcions
- Cal transferir totes les dades cada frame

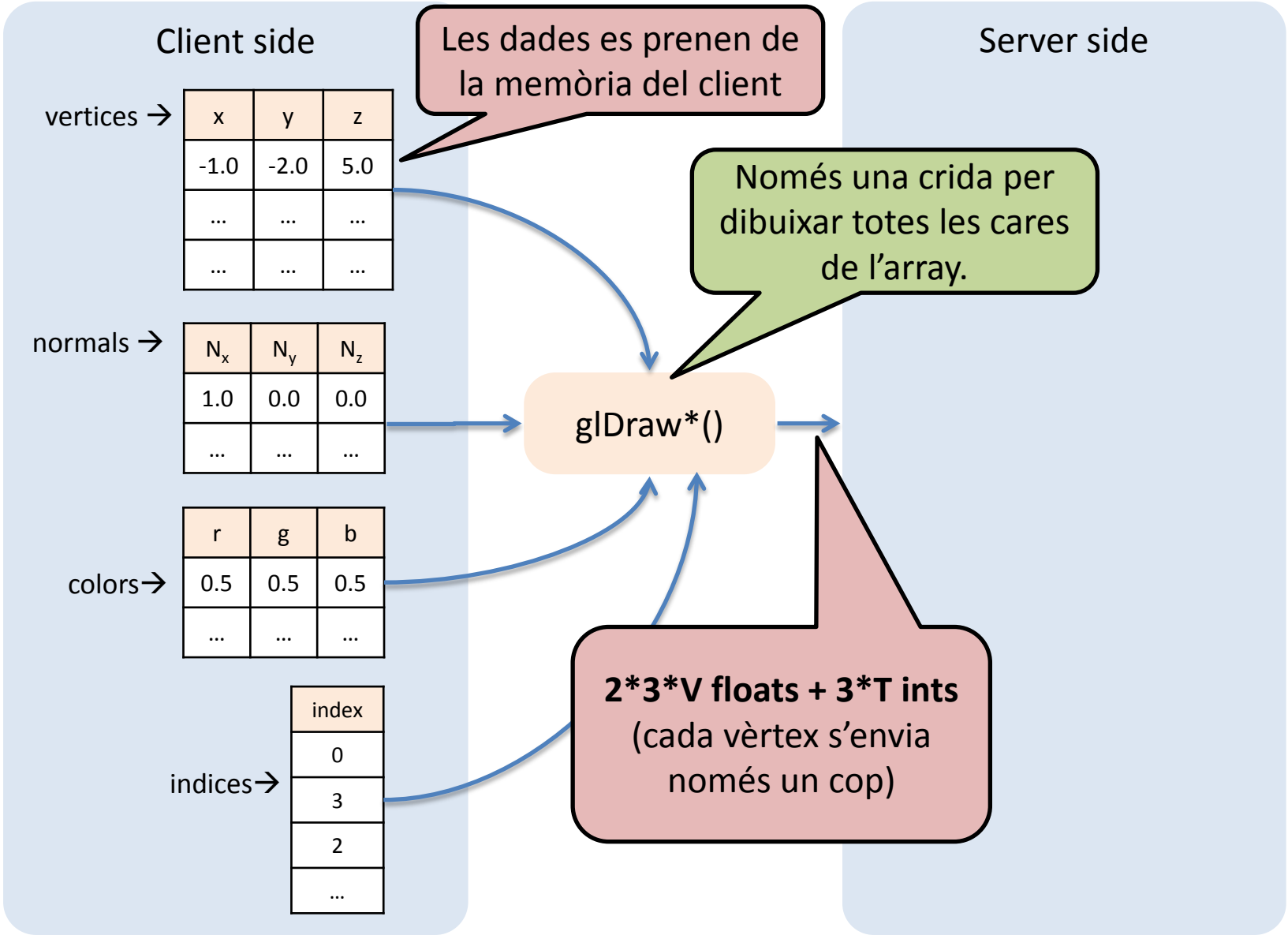


Vertex Arrays

Objectius:

- Reduir crides a OpenGL
- Enviar (i processar) un cop cada vèrtex

Vertex Arrays



Vertex Arrays

glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT, indices)

❶ ❷ ❸ ❹

❶ És la primitiva: GL_TRIANGLES, GL_QUADS ...

❷ És el número d'índexos a l'array (ex. 12 triangles → 12*3=36)

❸ És el tipus dels índexs (normalment GL_UNSIGNED_INT)

❹ És l'apuntador a l'array amb els índexs (que haurem definit previament)

Quins atributs (normal, color, coords textura…) s'usaran?
Com s'especifiquen els apuntadors a aquests atributs?

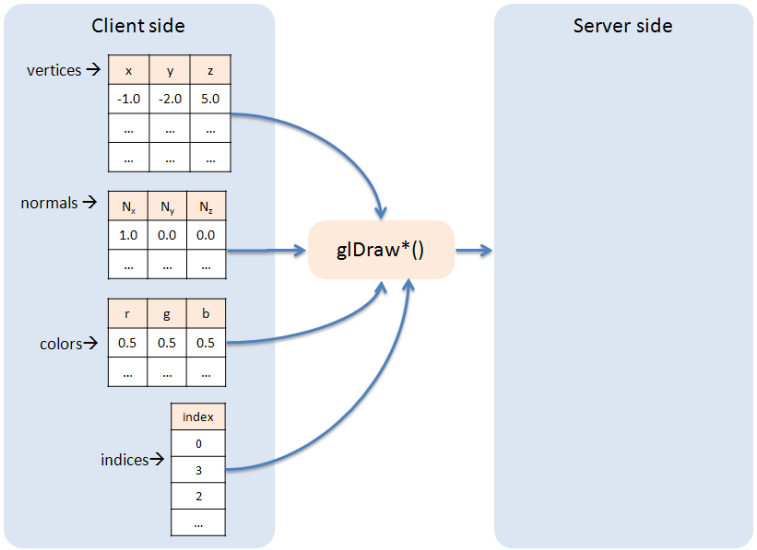
Vertex Arrays

```
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, (GLvoid*)verts);
glEnableVertexAttribArray(0);
```

```
void glVertexAttribPointer(
  GLuint index,                 // VS:  layout (location = 0) in vec3 vertex;
  GLint size,                   // Num de coordenades (1,2,3,4)
  GLenum type,                  // Tipus de cada coordenada: GL_FLOAT o GL_DOUBLE
  GLboolean normalized,         // Per convertir a valors en [0,1]
  GLsizei stride,               // Normalment 0 (un array per cada atribut)
  const GLvoid * pointer);      // Apuntador a les dades
```

Vertex Arrays - resum

- Una única crida a funció (per objecte)
- Els vèrtexs s'envien un cop
- Menys flexible que el mode immediat
- Encara cal transferir moltes dades cada frame



Vertex buffer object

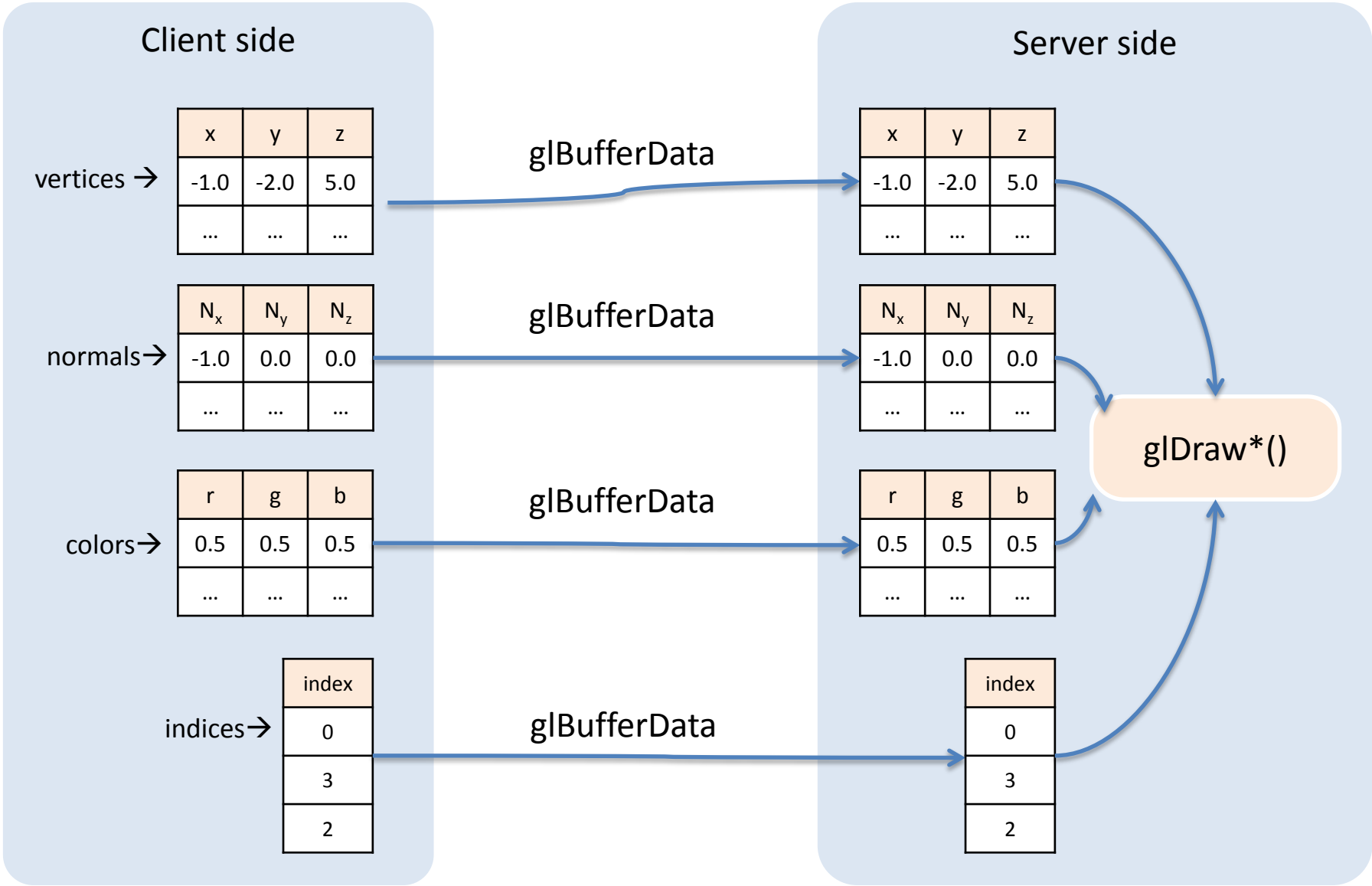
Objectiu:

- Evitar transferir les dades cada frame

Idea:

- Emmagatzemar les dades del VA al servidor.

Vertex buffer object



EXEMPLE: DRAW-VBO-EXT

Setup: 1/3

```
// Step 1: Create and fill STL arrays (coords, normals, idx…)
vector<float> coords;         // x,y,z,x,y,z…
vector<float> normals;        // nx,ny,nz,nx,ny,nz…
vector<unsigned int> indices; // i0,i1,i2 i3,i4,i5…
for (…)
   vertices.push_back(x);
   vertices.push_back(y);
   vertices.push_back(z);

for (…)
   indices.push_back(index);
```

Setup: 2/3

```
// Step 2: Create VAO & empty buffers (coords, normals, indices)
 GLuint VAO;
 glGenVertexArrays(1, &VAO);

 GLuint coordBufferID;
 glGenBuffers(1, &coordBufferID);

 GLuint normalBufferID;
 glGenBuffers(1, &normalBufferID);

 GLuint indexBufferID;
 glGenBuffers(1, &indexBufferID);
```

Setup: 3/3

```
// Step 3: Define VBO data (coords, normals, indices)
 glBindVertexArray(VAO);
 glBindBuffer(GL_ARRAY_BUFFER, coordBufferID); // VAO
 glBufferData(GL_ARRAY_BUFFER, sizeof(float)*coords.size(),
&coords[0], GL_STATIC_DRAW); // ONCE
 glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0); // VAO
 glEnableVertexAttribArray(0); // VAO
 glBindBuffer(GL_ARRAY_BUFFER, normalBufferID); // VAO
 glBufferData(GL_ARRAY_BUFFER, sizeof(float)*normals.size(),
&normals[0], GL_STATIC_DRAW); // ONCE
 glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 0, 0); // VAO
 glEnableVertexAttribArray(1); // VAO
 glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, indexBuffersID); // VAO
 glBufferData(GL_ELEMENT_ARRAY_BUFFER,
sizeof(int)*indices.size(),&indices[0], GL_STATIC_DRAW); // ONCE
 glBindVertexArray(0);
```

Draw

```
// Draw
glBindVertexArray(VAO);
glDrawElements(GL_TRIANGLES, numIndices, GL_UNSIGNED_INT,
(GLvoid*) 0); //numIndices=indices.size()
glBindVertexArray(0);
```

```
// Draw multiples instances
glBindVertexArray(VAO);
glDrawElementsInstanced(GL_TRIANGLES, numIndices,
GL_UNSIGNED_INT, (GLvoid*) 0, numInstances);
glBindVertexArray(0);
```

VS: int gl_InstanceID → instance number (0…numInstances-1)

Clean up

```
// Clean up
glDeleteBuffers(1, &coordBufferID);
glDeleteBuffers(1, &normalBufferID);

glDeleteBuffers(1,  &indexBufferID);

glDeleteVertexArrays(1, &VAO);
```

Vertex Buffer Objects - resum

- Una única crida a funció
- Els vèrtexs s'envien un cop (*)
- Les dades es transfereixen al servidor
- Menys flexible que el mode immediat