COMP220: Graphics & Simulation

4: Meshes

Module Roadmap

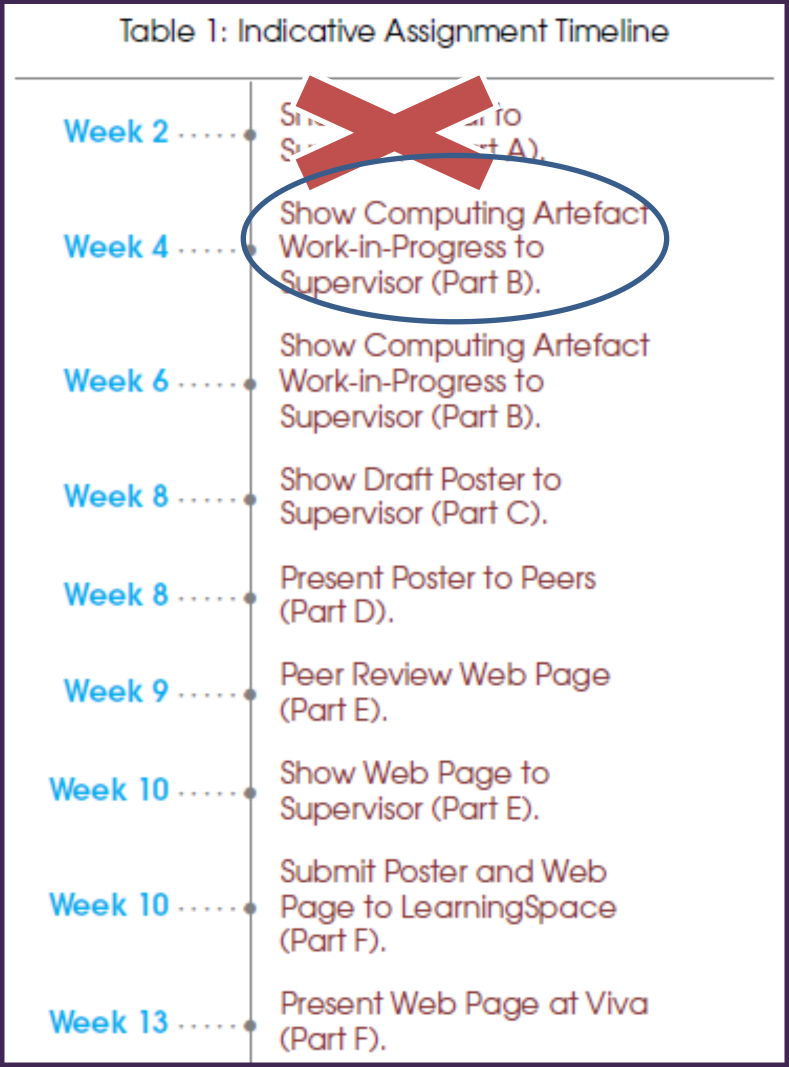


Table 1: Indicative Assignment Timeline

	Week 2	~~Sh[illegible] to S[illegible]t A).~~
Worksheet A	Week 4	Show Computing Artefact Work-in-Progress to Supervisor (Part B).
Worksheet B	Week 6	Show Computing Artefact Work-in-Progress to Supervisor (Part B).
Worksheet C	Week 8	Show Draft Poster to Supervisor (Part C).
	Week 8	Present Poster to Peers (Part D).
	Week 9	Peer Review Web Page (Part E).
Worksheet D	Week 10	Show Web Page to Supervisor (Part E).
	Week 10	Submit Poster and Web Page to LearningSpace (Part F).
	Week 13	Present Web Page at Viva (Part F).

More complex meshes

Winding order

Winding order

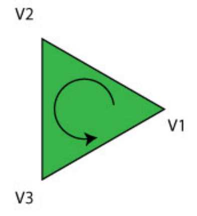
- It is sometimes important to know which side of a triangle is the “front” and which is the “back”

Winding order

- It is sometimes important to know which side of a triangle is the “front” and which is the “back”
- OpenGL determines this by **winding order**

Winding order

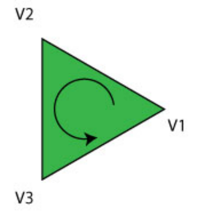
- It is sometimes important to know which side of a triangle is the "front" and which is the "back"
- OpenGL determines this by **winding order**



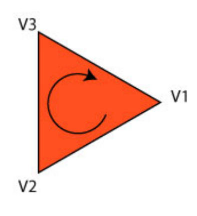
If the vertices go **anticlockwise**, you are looking at the **front**

Winding order

- It is sometimes important to know which side of a triangle is the "front" and which is the "back"
- OpenGL determines this by **winding order**



If the vertices go **anticlockwise**, you are looking at the **front**



If the vertices go **clockwise**, you are looking at the **back**

Backface culling

Backface culling

```
glEnable(GL_CULL_FACE);
```

Backface culling

```
glEnable(GL_CULL_FACE);
```

- This will cause only the front faces of triangles to be drawn

Backface culling

```
glEnable(GL_CULL_FACE);
```

- This will cause only the front faces of triangles to be drawn
- Triangles whose front face is not visible will be **culled**

Backface culling

```
glEnable(GL_CULL_FACE);
```

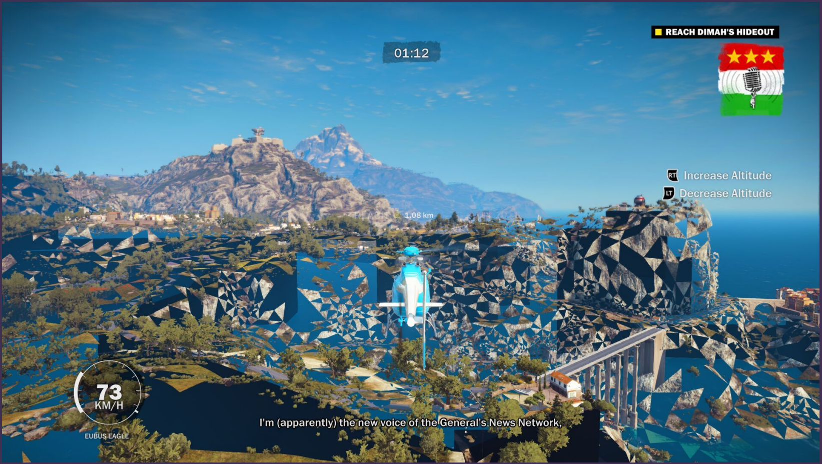
- This will cause only the front faces of triangles to be drawn
- Triangles whose front face is not visible will be **culled**
- Culled faces are not passed through the rasteriser or fragment shader

Backface culling

```
glEnable(GL_CULL_FACE);
```

- This will cause only the front faces of triangles to be drawn
- Triangles whose front face is not visible will be **culled**
- Culled faces are not passed through the rasteriser or fragment shader
- Saves time, and should make no difference to appearance — as long as all meshes are closed and have correct winding

When backface culling goes bad?



Vertices

Interleaved Vertices

Interleaved Vertices

- Up until this point we have been storing vertex positions as floats

Interleaved Vertices

- Up until this point we have been storing vertex positions as floats
- If we need a vertex to have colours, we can store these in a separate Vertex Buffer

Interleaved Vertices

- Up until this point we have been storing vertex positions as floats
- If we need a vertex to have colours, we can store these in a separate Vertex Buffer
- Or we can create a **C structure** which represents a Vertex, which has member variables which represent positions, colours, normals etc

Interleaved Vertices

- Up until this point we have been storing vertex positions as floats
- If we need a vertex to have colours, we can store these in a separate Vertex Buffer
- Or we can create a **C structure** which represents a Vertex, which has member variables which represent positions, colours, normals etc
- This is known as Interleaved Vertices and in

Interleaved Vertices

- Up until this point we have been storing vertex positions as floats
- If we need a vertex to have colours, we can store these in a separate Vertex Buffer
- Or we can create a **C structure** which represents a Vertex, which has member variables which represent positions, colours, normals etc
- This is known as Interleaved Vertices and in **MOST** cases is more efficient

Vertex Structure 1

```
struct Vertex
{
    float x,y,z;
};

Vertex v[]={{-0.5f,-0.5f,0.0f},
            {0.5f,-0.5f,0.0f},
            {0.0f,0.5f,0.0f}};
```

Vertex Structure 2

```
struct Vertex
{
    float x,y,z;
    float r,g,b,a;
};

Vertex v[]={{-0.5f,-0.5f,0.0f,1.0f,0.0f,0.0f ↩
    ,1.0f},
            {0.5f,-0.5f,0.0f,0.0f,1.0f,0.0f ↩
                ,1.0f},
            {0.0f,0.5f,0.0f,0.0f,0.0f,1.0f,1.0 ↩
                f}};
```

Changes to the Vertex Buffer

Changes to the Vertex Buffer

- There will be a slight change to our vertex buffer

Changes to the Vertex Buffer

- There will be a slight change to our vertex buffer
- We have to take into account the size of the Vertex structure and the number of vertices in the buffer

Vertex Buffer Changes - Old version

```
glBufferData(GL_ARRAY_BUFFER, sizeof( ↩
    g_vertex_buffer_data),   ↩
    g_vertex_buffer_data, GL_STATIC_DRAW);
```

Vertex Buffer Changes - new version

```
glBufferData(GL_ARRAY_BUFFER, 3* sizeof(Vertex ↩
    ), v, GL_STATIC_DRAW);
```

Changes to the Vertex Array

Changes to the Vertex Array

- Since the layout of the vertices have changed in memory, we need to update the Vertex Array Object to reflect this

Changes to the Vertex Array

- Since the layout of the vertices have changed in memory, we need to update the Vertex Array Object to reflect this
- Remember that the VAO describes the format of the vertices to the pipeline and enables the binding of vertex data to attributes in the shader

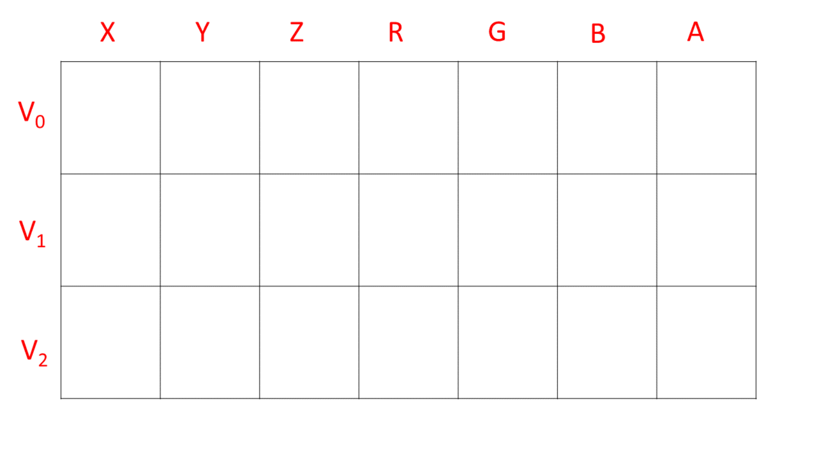
Vertex Array Object - Old version

```
    glEnableVertexAttribArray(0);
    glVertexAttribPointer(0,3,GL_FLOAT,GL_FALSE,0,( ↩
        void*)0);
```

Vertex Array Object - New version

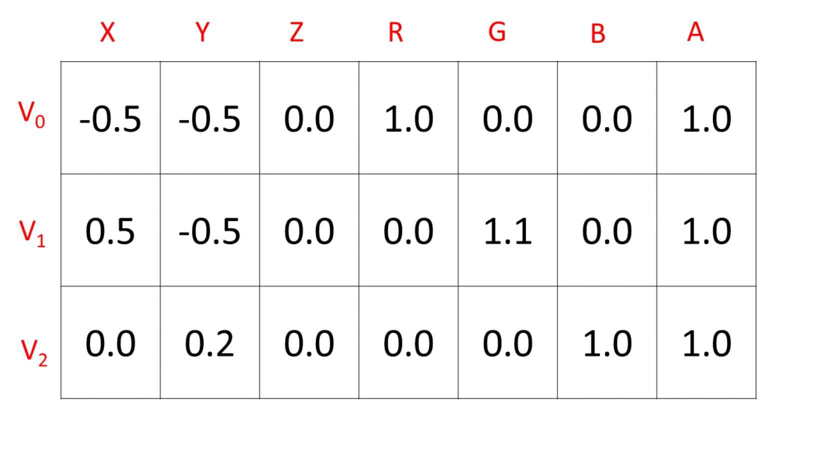
```
    glEnableVertexAttribArray(0);
    glVertexAttribPointer(0,3,GL_FLOAT,GL_FALSE,sizeof ↩
        (Vertex),(void*)0);

    glEnableVertexAttribArray(1);
    glVertexAttribPointer(1,4,GL_FLOAT,GL_FALSE,sizeof ↩
        (Vertex),(void*)(3*sizeof(float)));
```

Memory and Vertex Array Object 1



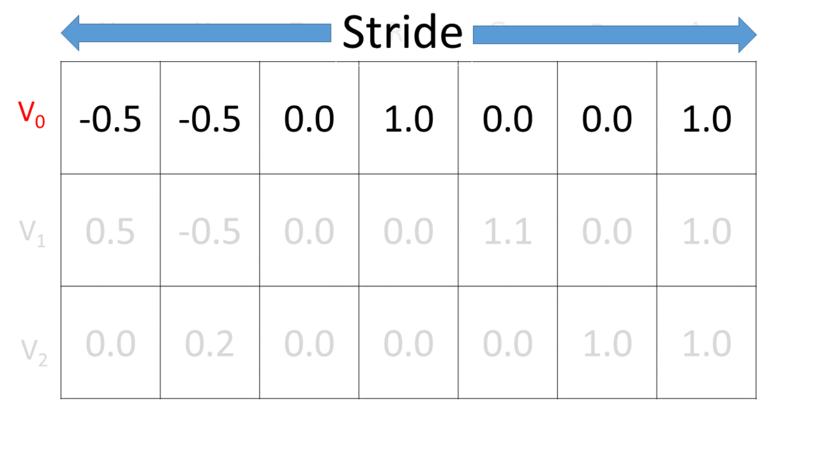
	X	Y	Z	R	G	B	A
V_0							
V_1							
V_2							

Memory and Vertex Array Object 2



	X	Y	Z	R	G	B	A
V_0	-0.5	-0.5	0.0	1.0	0.0	0.0	1.0
V_1	0.5	-0.5	0.0	0.0	1.1	0.0	1.0
V_2	0.0	0.2	0.0	0.0	0.0	1.0	1.0

Memory and Vertex Array Object 3 - Stride



Stride

V_0	-0.5	-0.5	0.0	1.0	0.0	0.0	1.0
V_1	0.5	-0.5	0.0	0.0	1.1	0.0	1.0
V_2	0.0	0.2	0.0	0.0	0.0	1.0	1.0

Memory and Vertex Array Object 3 - Offset

Offset = 3 * sizeof(float)

	X	Y	Z	R	G	B	A
V_0	-0.5	-0.5	0.0	1.0	0.0	0.0	1.0
V_1	0.5	-0.5	0.0	0.0	1.1	0.0	1.0
V_2	0.0	0.2	0.0	0.0	0.0	1.0	1.0

Element Buffer

Element Buffer

Element Buffer

- If we look at the cube sample, we are sending 36 vertices

Element Buffer

- If we look at the cube sample, we are sending 36 vertices
- This is a bit wasteful considering that some of these vertices are duplicates

Element Buffer

- If we look at the cube sample, we are sending 36 vertices
- This is a bit wasteful considering that some of these vertices are duplicates
- We can use an **Element Buffer** to optimise our drawing

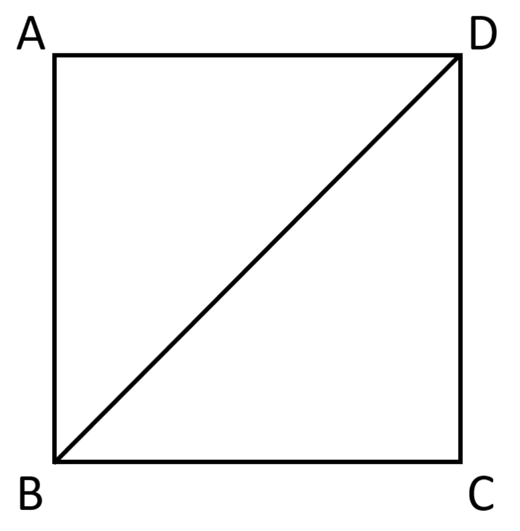
Element Buffer

- If we look at the cube sample, we are sending 36 vertices
- This is a bit wasteful considering that some of these vertices are duplicates
- We can use an **Element Buffer** to optimise our drawing
- An Element Buffer holds an integer which is an offset into a Vertex Buffer

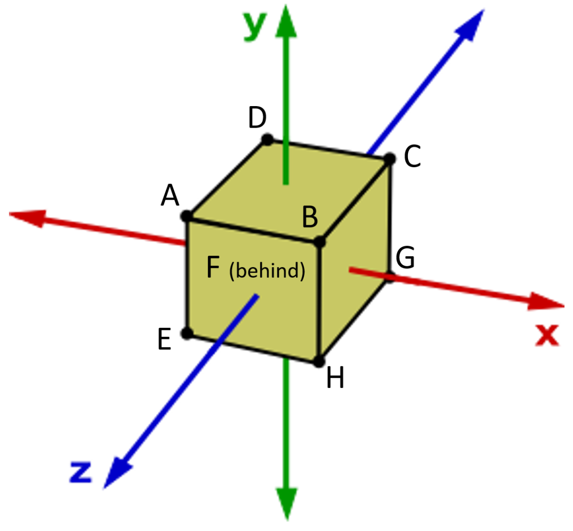
Creating & Using Element Buffer

Live Coding

Exercise 1 - Let’s draw a square!



Exercise 2 - Let's draw a cube!



Exercise 3 - Element Buffer

- Create a cube using an Element Buffer
- Create a function which fills a Vertex Buffer and Element Buffer for drawing a Sphere

Further Reading - Interleaved Vertices

- iOS Development Docs - `https://developer.apple.com/library/content/documentation/3DDrawing/Conceptual/OpenGLES_ProgrammingGuide/TechniquesforWorkingwithVertexData/TechniquesforWorkingwithVertexData.html`
- To interleave or not to interleave - `https://anteru.net/blog/2016/02/14/3119/index.html`
- Vertex Specification Best Practices - `https://www.khronos.org/opengl/wiki/Vertex_Specification_Best_Practices`

Further Reading - Element Buffer

- VBO indexing - `http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-9-vbo-indexing/`
- Element Buffer - `https://goharsha.com/lwjgl-tutorial-series/element-buffer-objects/`