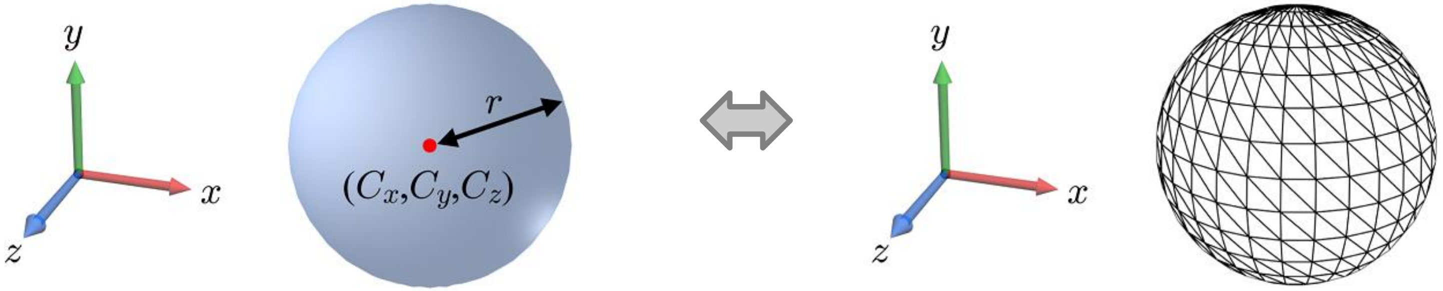
Chapter III
Modeling

Polygon Mesh

- In real-time applications such as games, the *polygon mesh* representation dominates. It is not an accurate representation but an approximate one, where the mesh vertices are the points *sampling* the smooth surface.

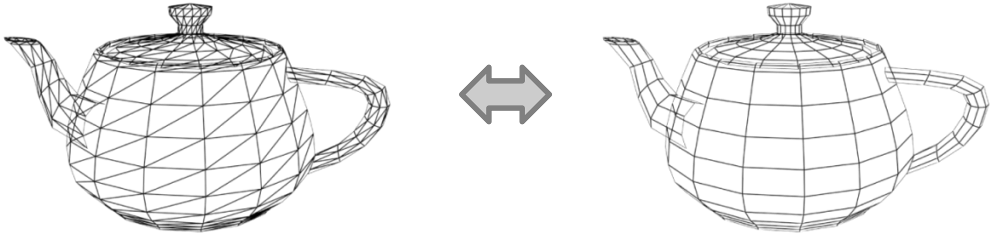


$$(x - C_x)^2 + (y - C_y)^2 + (z - C_z)^2 = r^2$$

- The simplest polygon is a triangle. The only polygon OpenGL ES supports is triangle and so polygon mesh in OpenGL ES equals triangle mesh.
- In a typical closed mesh, the number of triangles is approximately twice the number of vertices, i.e., given *n* vertices, we have about 2*n* triangles.

Polygon Mesh (cont'd)

- Even though the triangle mesh is far more popular in general, the quad mesh is often preferred especially for modeling step.



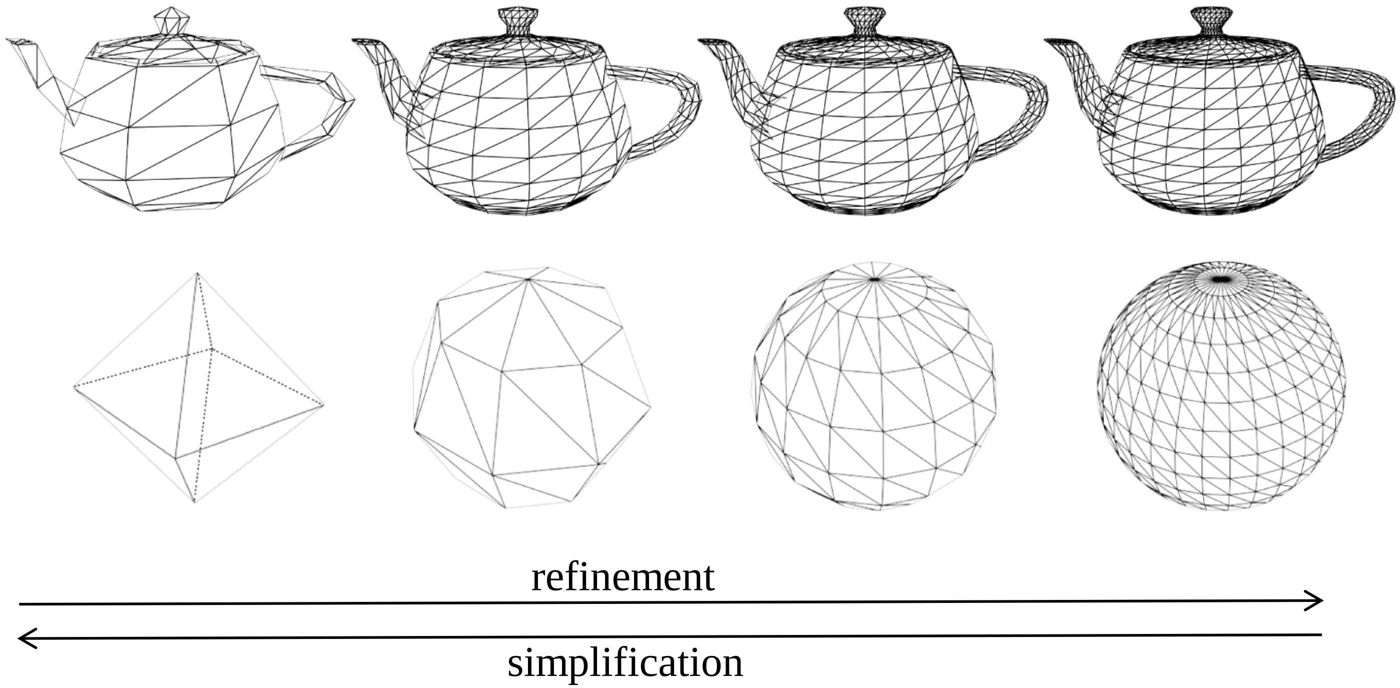
triangle mesh

quad mesh

- It is straightforward to convert a quad mesh into a triangle mesh. Each quad is split into two triangles.

Polygon Mesh (cont'd)

- Levels of detail (LOD)



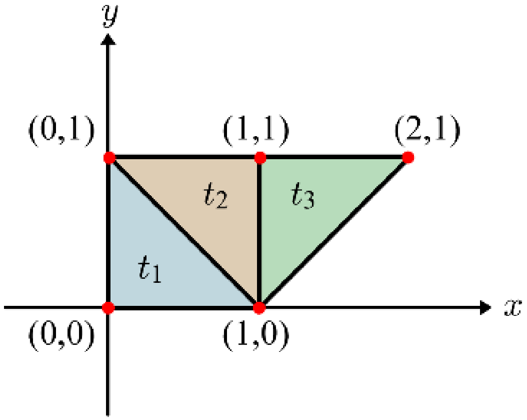
refinement

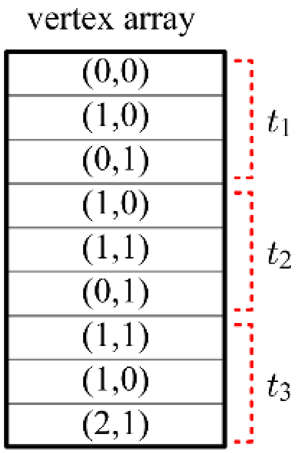
simplification

- Tradeoff between accuracy and efficiency: As the resolution is increased, the mesh becomes closer to the original curved surface, but the time needed for processing the mesh is increased.

Polygon Mesh – Non-indexed Representation

- The vertices are enumerated in a memory space, named *vertex array*.
- Three vertices are read in linear order to make up a triangle.



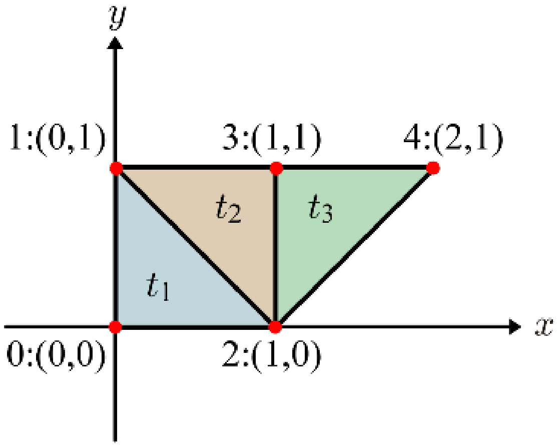


vertex array	
(0,0)	t_1
(1,0)	
(0,1)	
(1,0)	t_2
(1,1)	
(0,1)	
(1,1)	t_3
(1,0)	
(2,1)	

- It is inefficient because the vertex array contains redundant data.

Polygon Mesh – Indexed Representation

- A better method is using a separate *index array*.
  - A vertex appears only once in the vertex array.
  - Three indices per triangle are stored in the index array.



vertex array

0	(0,0)
1	(0,1)
2	(1,0)
3	(1,1)
4	(2,1)

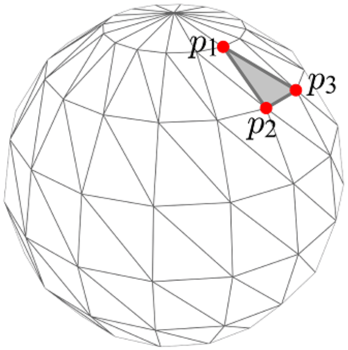
index array

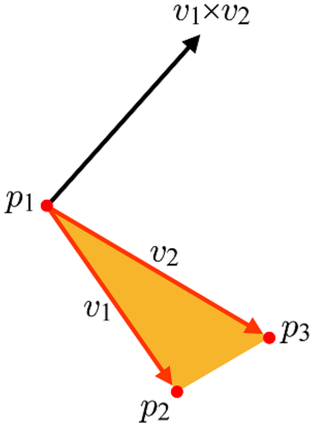
0	t_1
2	
1	
2	t_2
3	
1	
3	t_3
2	
4	

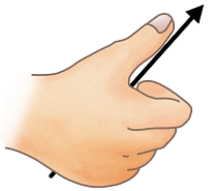
- The data stored in the vertex array are not restricted to vertex positions but include a lot of additional information. (You will see the data throughout this class.) Therefore, the vertex array storage saved by removing the duplicate data outweighs the additional storage needed for the index array.

Surface Normals

- Surface normals play a key role in computer graphics.
- Given triangle $\langle p_1, p_2, p_3 \rangle$, let v_1 denote the vector connecting the first vertex (p_1) and the second (p_2). Similarly, the vector connecting the first vertex (p_1) and the third (p_3) is denoted by v_2. Then, the *triangle normal* can be computed using the *cross product* based on the *right-hand rule*.





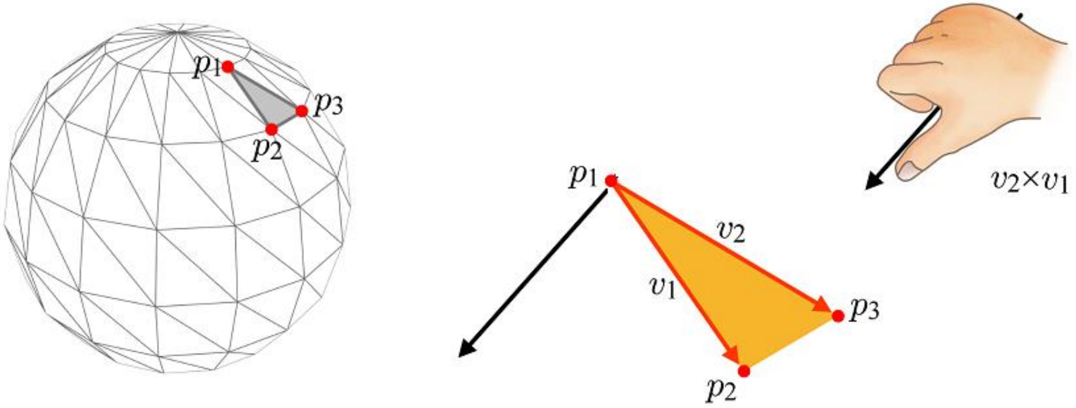


$$\frac{v_1 \times v_2}{||v_1 \times v_2||}$$

- Every normal vector is made to be a unit vector by default.
- Note that p_1, p_2, and p_3 are ordered *counter-clockwise* (CCW).

Surface Normals (cont'd)

- What if the vertices are ordered *clockwise* (CW), i.e., $\langle p_1, p_3, p_2 \rangle$?
- The vector connecting the first vertex (p_1) and the second (p_3) and that connecting the first vertex (p_1) and the third (p_2) define the triangle normal.

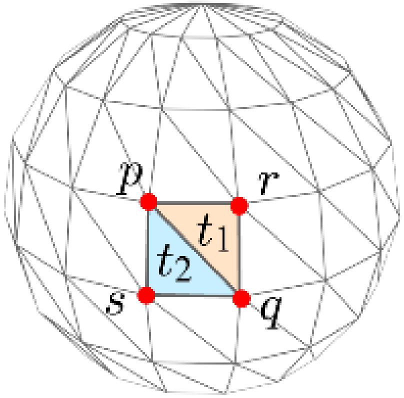


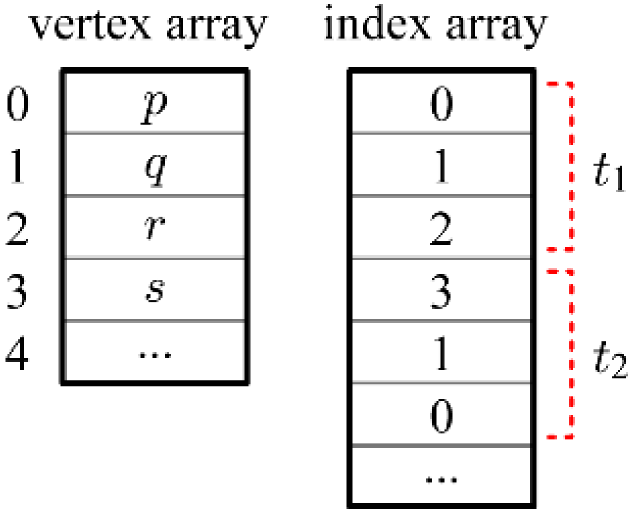
$$\frac{v_2 \times v_1}{||v_2 \times v_1||}$$

- The normal is in the opposite direction!
- The normal direction depends on the vertex order, i.e., whether $\langle p_1, p_2, p_3 \rangle$ or $\langle p_1, p_3, p_2 \rangle$.
- In computer graphics, surface normals are supposed to point out of the polyhedron. For this, we need CCW ordering of the vertices, i.e., $\langle p_1, p_2, p_3 \rangle$, instead of $\langle p_1, p_3, p_2 \rangle$.

Surface Normals (cont'd)

- In the index array, the vertices are ordered CCW.





vertex array

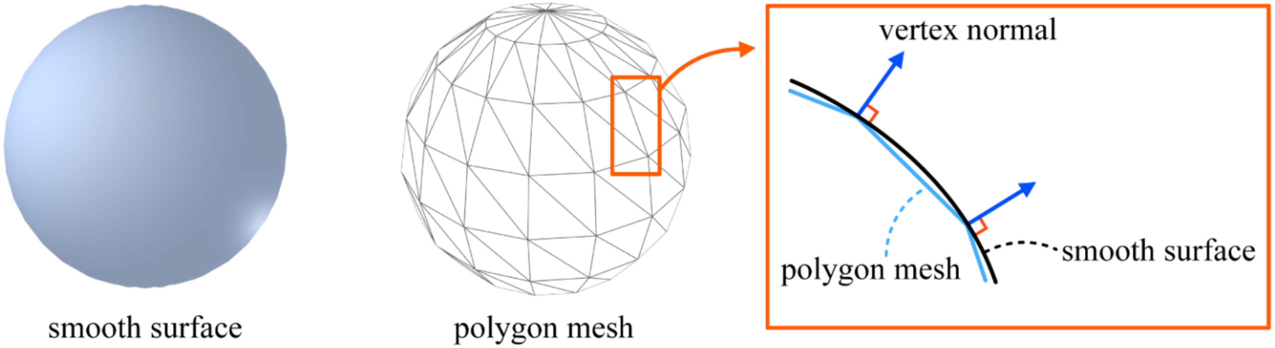
0	p
1	q
2	r
3	s
4	...

index array

0	t_1
1	
2	
3	t_2
1	
0	
...	

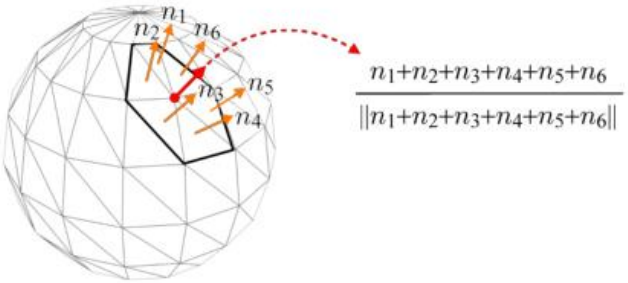
Surface Normals (cont'd)

- As will be discussed later, what is really important is the *vertex normal*.
- The polygon mesh is an approximation of a smooth surface, and the mesh vertices are the points *sampling* the smooth surface. So, the vertex normal should approximate the normal of the smooth surface at the vertex position.



smooth surface   polygon mesh

- A vertex normal is usually computed by averaging the normals of all polygons sharing the vertex.

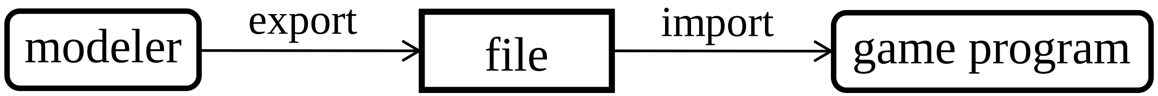


$$\frac{n_1+n_2+n_3+n_4+n_5+n_6}{||n_1+n_2+n_3+n_4+n_5+n_6||}$$

- Modeling packages such as 3ds Max do compute vertex normals.
- Vertex normals are an indispensable component of the vertex array.

Export and Import

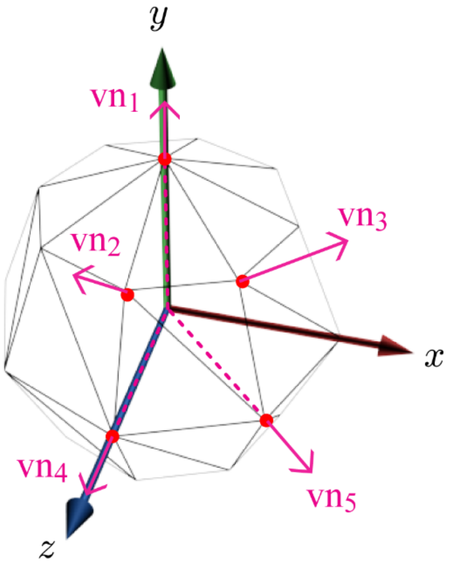
- Game objects and related data created using off-line graphics packages are stored in files and passed to the run-time game program.
  - The process of outputting the data in a format suitable for other applications is called *export*.
  - On the other hand, taking such exported data is called *import*.



- For exporting and importing, simple scripts or programs are used. For example, 3ds Max provides MAXScript and an exporter can be written using MAXScript.
- In 3ds Max, a lot of file formats are supported for export. Among the popular is .obj file.

Export and Import (cont'd)

- Consider a low-resolution mesh of a unit sphere.
  - The spherical coordinates are uniformly sampled at every 45 degrees such that the mesh is composed of 26 vertices in total.
  - Shown below are some fractions of the .obj file.



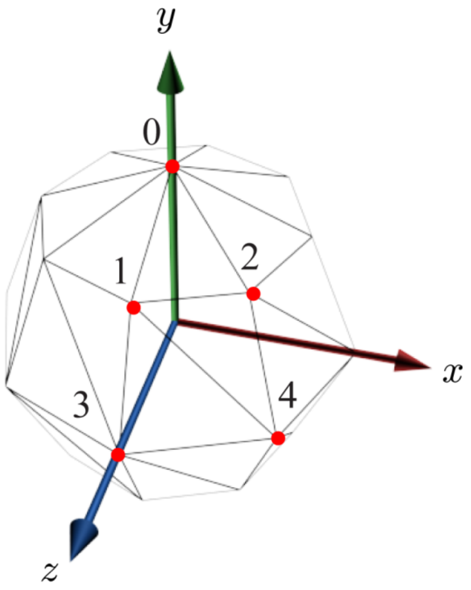
```
v    0.000  1.000  0.000
v    0.000  0.707  0.707
v    0.500  0.707  0.500
v    0.000  0.000  1.000
v    0.707  0.000  0.707
                ⋮
v    0.000 -1.000  0.000
# 26 vertices
```

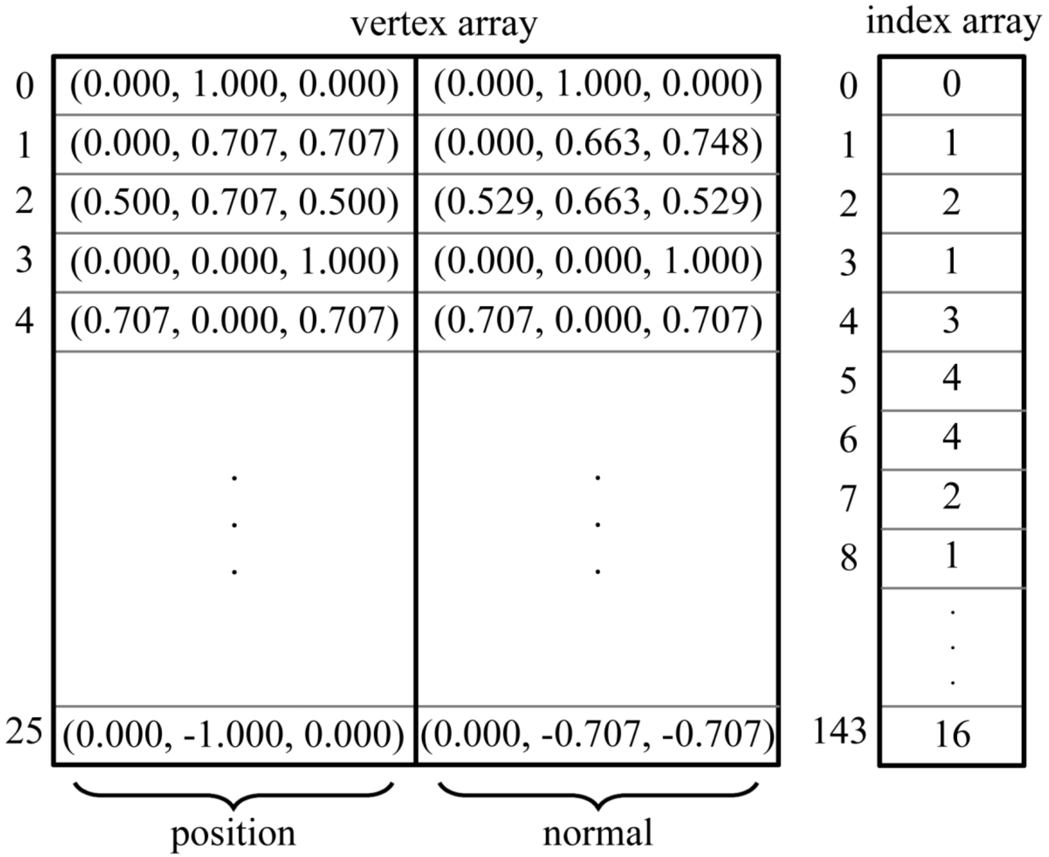
```
vn  0.000  1.000  0.000
vn  0.000  0.663  0.748
vn  0.529  0.663  0.529
vn  0.000  0.000  1.000
vn  0.707  0.000  0.707
                ⋮
vn  0.000 -0.707 -0.707
# 26 vertex normals
```

```
g Sphere001
f  1//1  2//2  3//3
f  2//2  4//4  5//5
f  5//5  3//3  2//2
            ⋮
f  18//18  10//10  17//17
# 48 faces
```

Export and Import (cont'd)

- The triangle mesh stored in .obj file is imported into OpenGL ES to fill the vertex and index arrays.
- As the mesh is composed of 48 triangles, the index array has 144 (48 times 3) elements.





vertex array

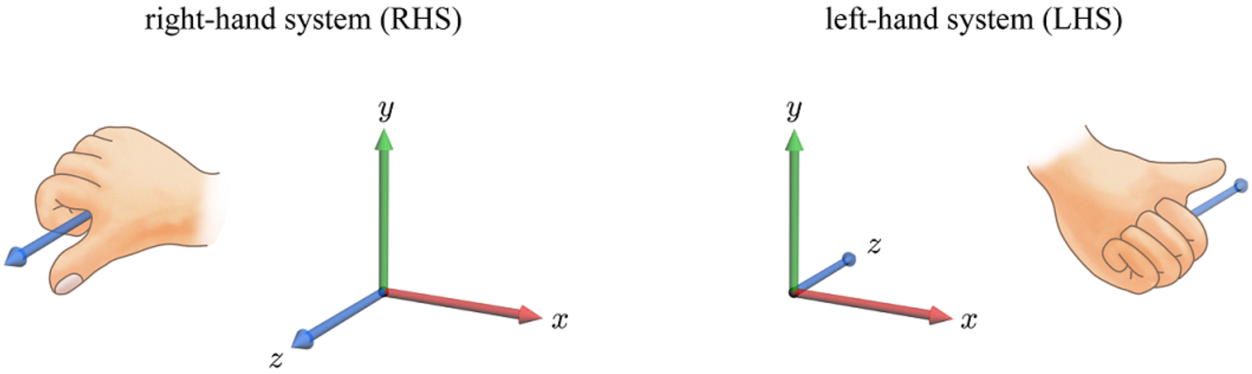
	position	normal
0	(0.000, 1.000, 0.000)	(0.000, 1.000, 0.000)
1	(0.000, 0.707, 0.707)	(0.000, 0.663, 0.748)
2	(0.500, 0.707, 0.500)	(0.529, 0.663, 0.529)
3	(0.000, 0.000, 1.000)	(0.000, 0.000, 1.000)
4	(0.707, 0.000, 0.707)	(0.707, 0.000, 0.707)
	⋮	⋮
25	(0.000, -1.000, 0.000)	(0.000, -0.707, -0.707)

index array

0	0
1	1
2	2
3	1
4	3
5	4
6	4
7	2
8	1
	⋮
143	16

Right-hand and Left-hand Systems

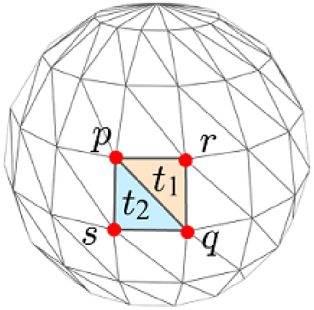
- Right-hand system vs. left-hand system



right-hand system (RHS)

left-hand system (LHS)

- So far, we implicitly assumed RHS and used the right hand for defining the triangle normals. In LHS, the triangle normal is defined using the left hand.
- To make the normals point out of the object, RHS adopts CCW ordering whereas LHS adopts CW ordering.



for RHS

$t_1 = \langle p,q,r \rangle$

$t_2 = \langle s,q,p \rangle$

for LHS

$t_1 = \langle p,r,q \rangle$

$t_2 = \langle s,p,q \rangle$

- OpenGL ES uses the RHS.