Buffer Geometry

A. Augusto de Sousa

Defining a mesh… "manually"

- Face by face
  - Edge by edge (with multiple repetitions)
    - Vertex by vertex (with multiple repetitions)
- Illumination calculation
  - On Vertices
    - Potential multiple repetitions
      - (one vertex belongs to 2 or more polygons)
    - Normals are needed

Defining meshes from Geometry (ThreeJS)

- EX: Sphere

```
var sphere=new THREE.SphereGeometry( this.sphereRadius, 8, 8 )
this.sphereMesh = new THREE.Mesh( sphere, this.sphereMaterial );
```

  - The mesh construction is "hidden" but it…
    - Uses vertices coordinates…
    - Defines edges…
    - Defines faces…
    - Lower potential for repetitions!

More Efficient than "manually"

Limited to some geometries (sphere, cylinder, box…)

Buffer Geometry

- Main objectives:
  - Optimize the use of the geometry entities
  - Optimize the communications with the Graphics Board
- How:
  - Store low-level information in buffers
    - Vertices Coordinates, Normals, Colours, Coordinates *(s,t)* of textures
  - Description of faces
    - List of Vertices
      - Less repetitions

Buffer Geometry

- Simple Information
  - Minimize additional calculations (...)
  - Delivered directly to the Graphics Board
- Normals, according to:
  - Flat Shading
  - Smooth (Gouraud) Shading
  - Calculated from the Surface Shape

Buffer Geometry

Position

	x	*y*	*z*
1			
2			
3			
4			
…			

Normals

	nx	*ny*	*nz*
1			
2			
3			
4			
…			

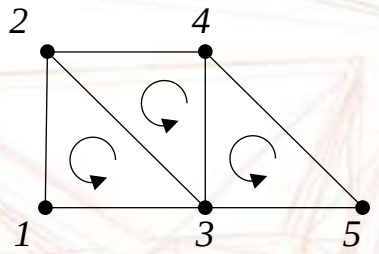
Colours

	R	*G*	*B*
1			
2			
3			
4			
…			

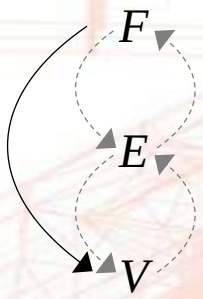
• • •

Faces/Triangles

	Va	*Vb*	*Vc*
1	1	3	2
2	4	2	3
2	3	5	4
4	6	4	5
…	…	…	…

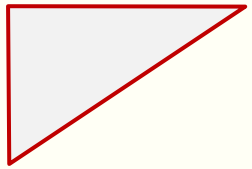


- Uncomplete Polygon Mesh Representation



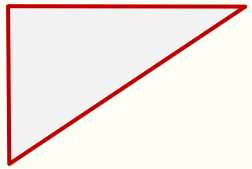
Buffer Geometry

- Ex: Triangle Strip
  - 3 initial vertices define a triangle

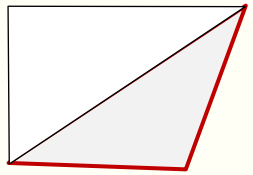


Buffer Geometry

- Ex: Triangle Strip
  - 3 initial vertices define a triangle

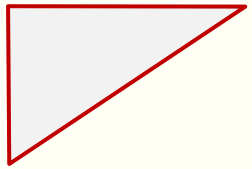


  - 1 more vertex defines a second triangle

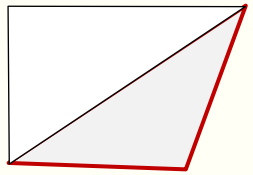


Buffer Geometry

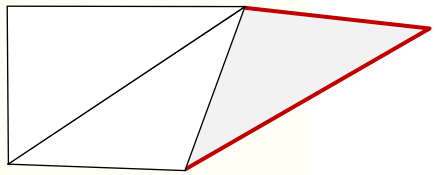
- Ex: Triangle Strip
  - 3 initial vertices define a triangle



  - 1 more vertex defines a second triangle

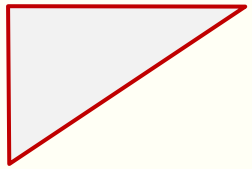


  - Each new vertex defines a new triangle

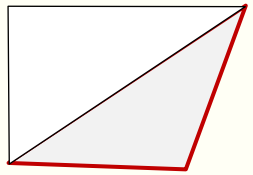


Buffer Geometry

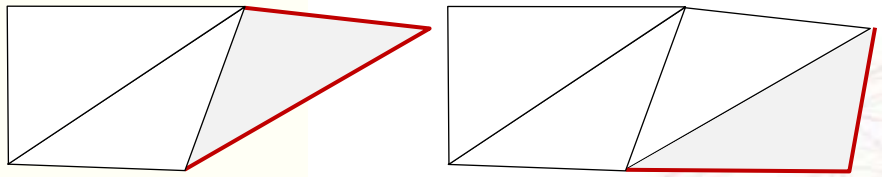
- Ex: Triangle Strip
  - 3 initial vertices define a triangle



  - 1 more vertex defines a second triangle

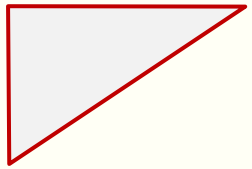


  - Each new vertex defines a new triangle

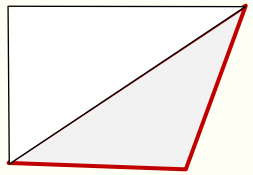


Buffer Geometry

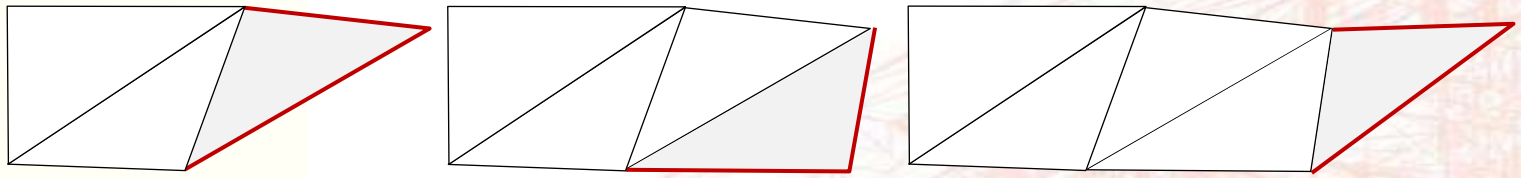
- Ex: Triangle Strip
  - 3 initial vertices define a triangle



  - 1 more vertex defines a second triangle

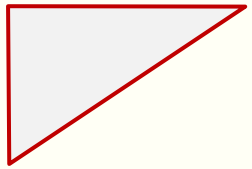


  - Each new vertex defines a new triangle

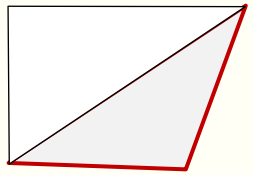


Buffer Geometry

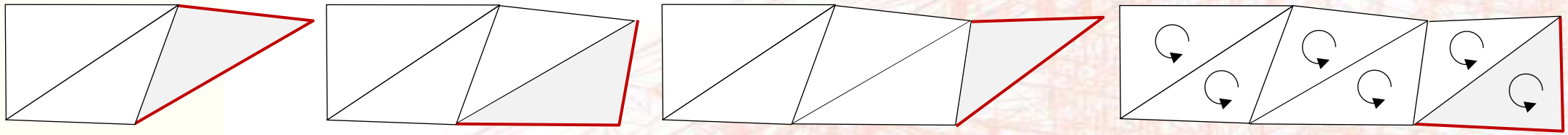
- Ex: Triangle Strip
  - 3 initial vertices define a triangle



  - 1 more vertex defines a second triangle



  - Each new vertex defines a new triangle

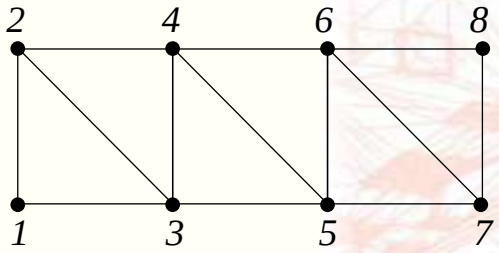


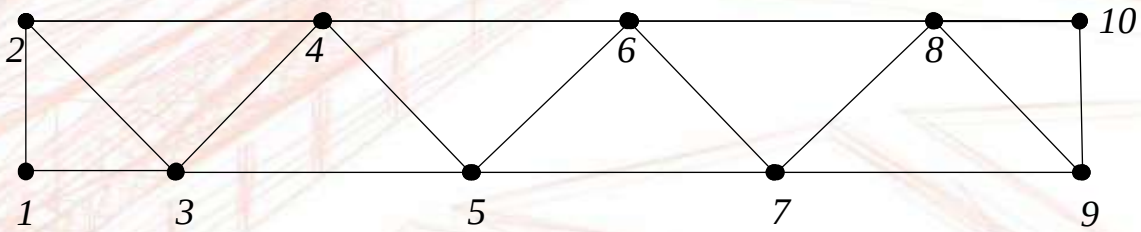
Nbr Vert = Nbr faces + 2

Nbr Edges = 2 * Nbr faces + 1

Buffer Geometry

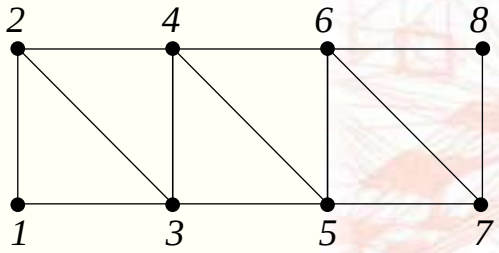
- Single Triangle Strip

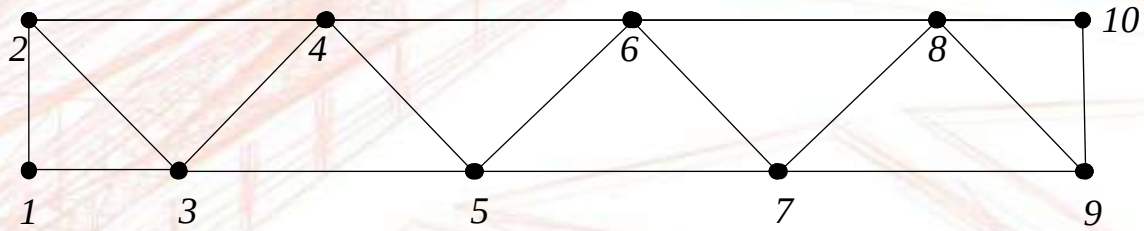




Buffer Geometry

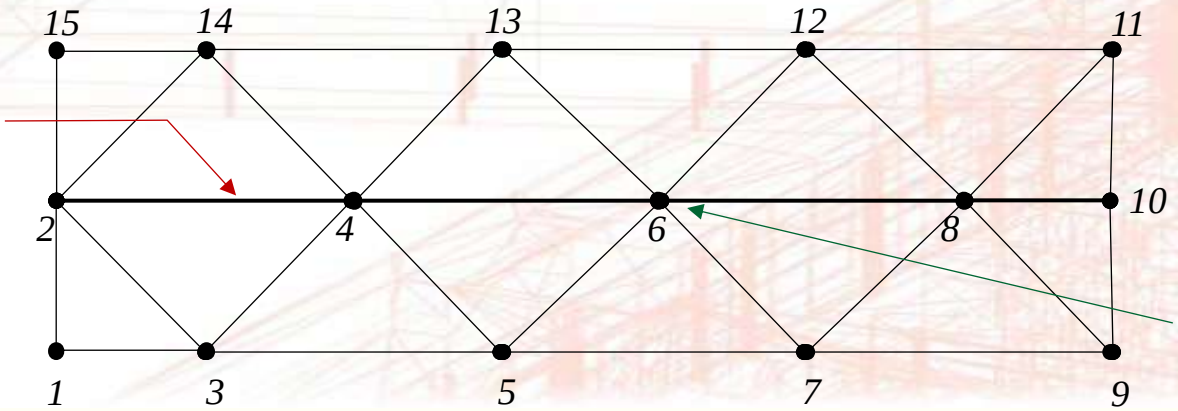
- Single Triangle Strip





- Multiple Triangle Strip
  - Larger surface

Edges visited twice still exist

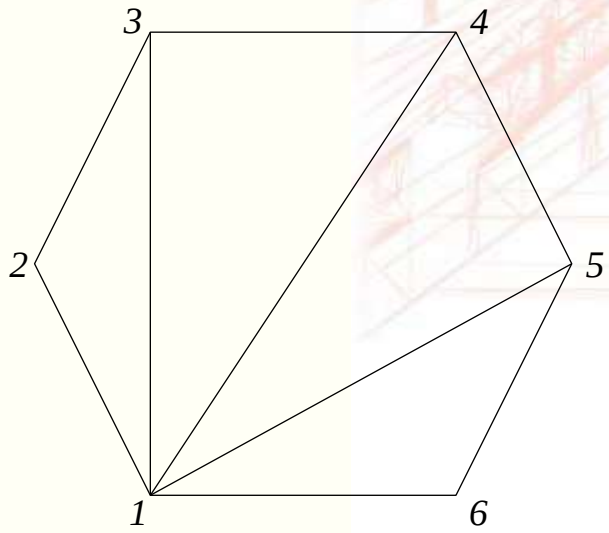


Vertices are not repeated

Buffer Geometry

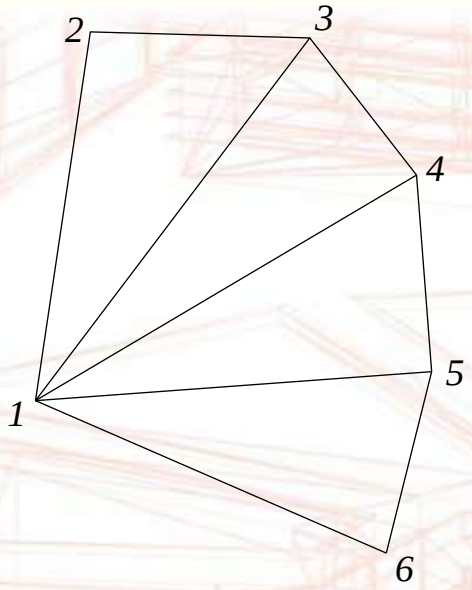
- Ex: Triangle Fan
  - First vertex starts every triangles

Regular Polygon



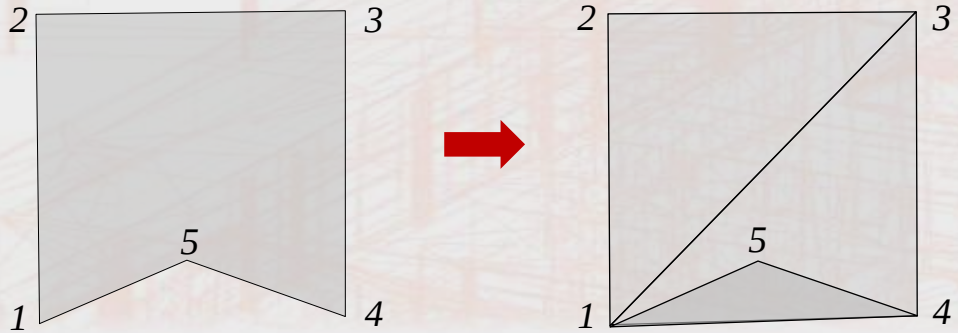
Faces
1, 3, 2
1, 4, 3
1, 5, 4
1, 6, 5

Irregular Polygon



Faces
1, 3, 2
1, 4, 3
1, 5, 4
1, 6, 5

Care with concavities…



Faces
1, 3, 2
1, 4, 3
1, 5, 4

ThreeJS: thousands of triangles in a cube

```
const geometry = new THREE.BufferGeometry();
const positions = [];
const normals = [];
const colors = [];

const color = new THREE.Color();
const pA = new THREE.Vector3();
const pB = new THREE.Vector3();
const pC = new THREE.Vector3();

const cb = new THREE.Vector3();
const ab = new THREE.Vector3();

for ( let i = 0; i < this.triangles; i ++ ) {

  //vertex a of the triangle
    const ax = ... ;
    const ay = ... ;
    const az = ... ;

 //vertex b of the triangle
    const bx = ... ;
    const bz = ... ;

 //vertex c of the triangle
    const cx = ... ;
    const cy = ... ;
    const cz = ... ;

    positions.push( ax, ay, az );
    positions.push( bx, by, bz );
    positions.push( cx, cy, cz );
```

```
 // flat face normal
    const nx = ...; // one normal, flat shading
    const ny = ...;
    const nz = ...;

 // the normal is common to the 3 vertices
 // 3 pushes of the same normal
    normals.push( nx, ny, nz );
    normals.push( nx, ny, nz );
    normals.push( nx, ny, nz );

 // color
    const vr = ...;
    const vg = ...;
    const vb = ...;
    color.setRGB( vr, vg, vb );

 // the color is common to the 3 vertices
 // 3 pushes of the same color
    colors.push( color.r, color.g, color.b);
    colors.push( color.r, color.g, color.b);
    colors.push( color.r, color.g, color.b);

}// end of the cycle for

// data of ALL triangles have been pushed
```

```
function disposeArray() {
    this.array = null;
}


// assign values to geometry; uses the stacks
geometry.setAttribute( 'position',... positions );
geometry.setAttribute( 'normal', ... normals);
geometry.setAttribute( 'color', ... colors);

geometry.computeBoundingSphere();

const material = ...;


// obtain mesh from the geometry
let mesh = new THREE.Mesh( geometry, material );

// cast and receives shadows
mesh.castShadow = true;
mesh.receiveShadow = true;

// ...

this.app.scene.add( mesh );
```

Buffer Geometry

- Practical work 2:
  - A polygon primitive
    - Stacks
    - Slices
    - Color_P
    - Color_C