The Art of Graphics Programming

Week 3: *Geometric Container Architecture*

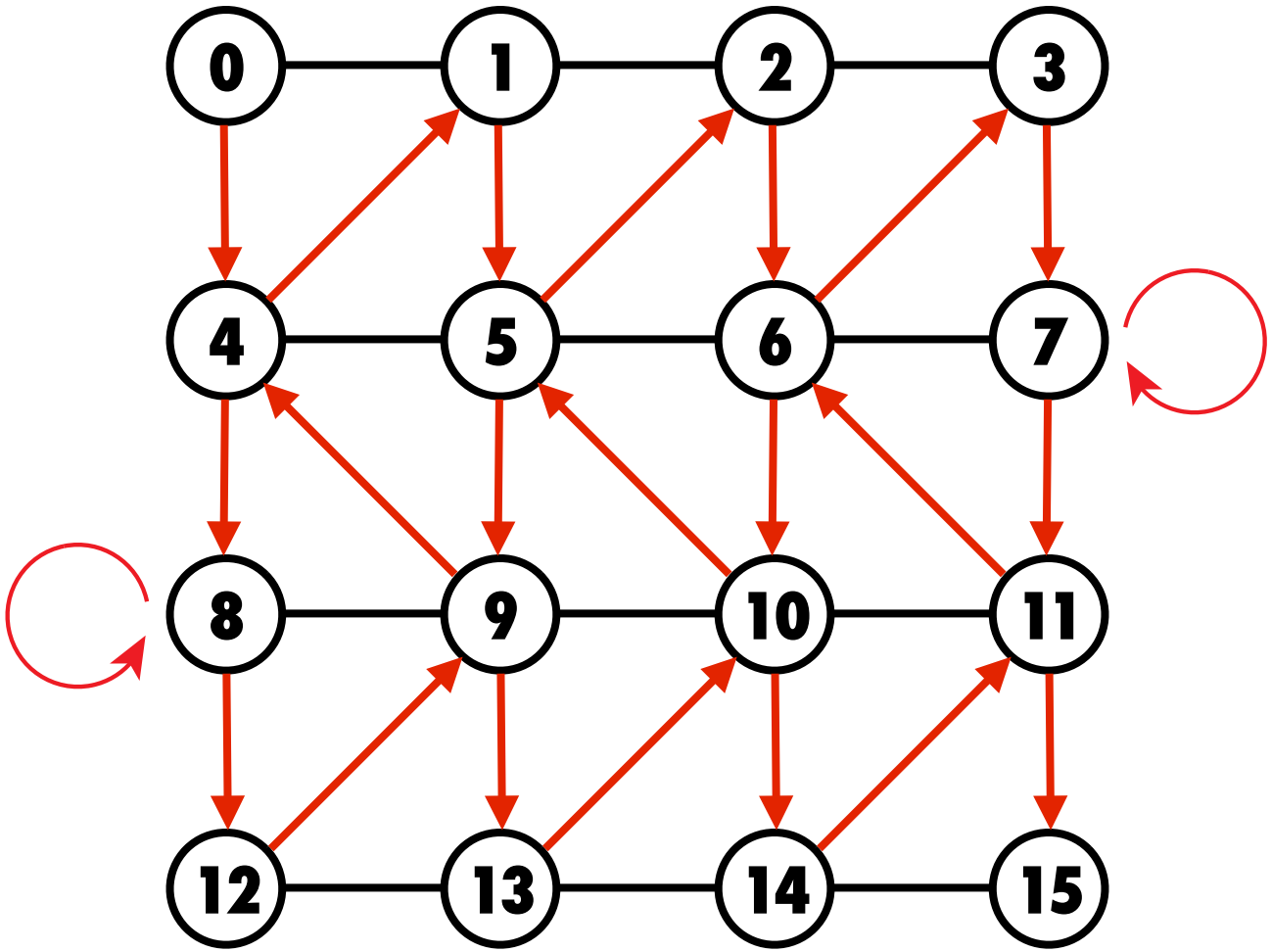
Patrick Hebron

Triangle strips provide an efficient, generalizable organizational structure for geometric mesh data.

But to achieve this efficiency, they require the vertices that compose the mesh's component triangles to be ordered in a particular manner.

Let's look at one vertex-ordering pattern that can be applied to 2D meshes as well as a variety of 3D mesh-based geometries.

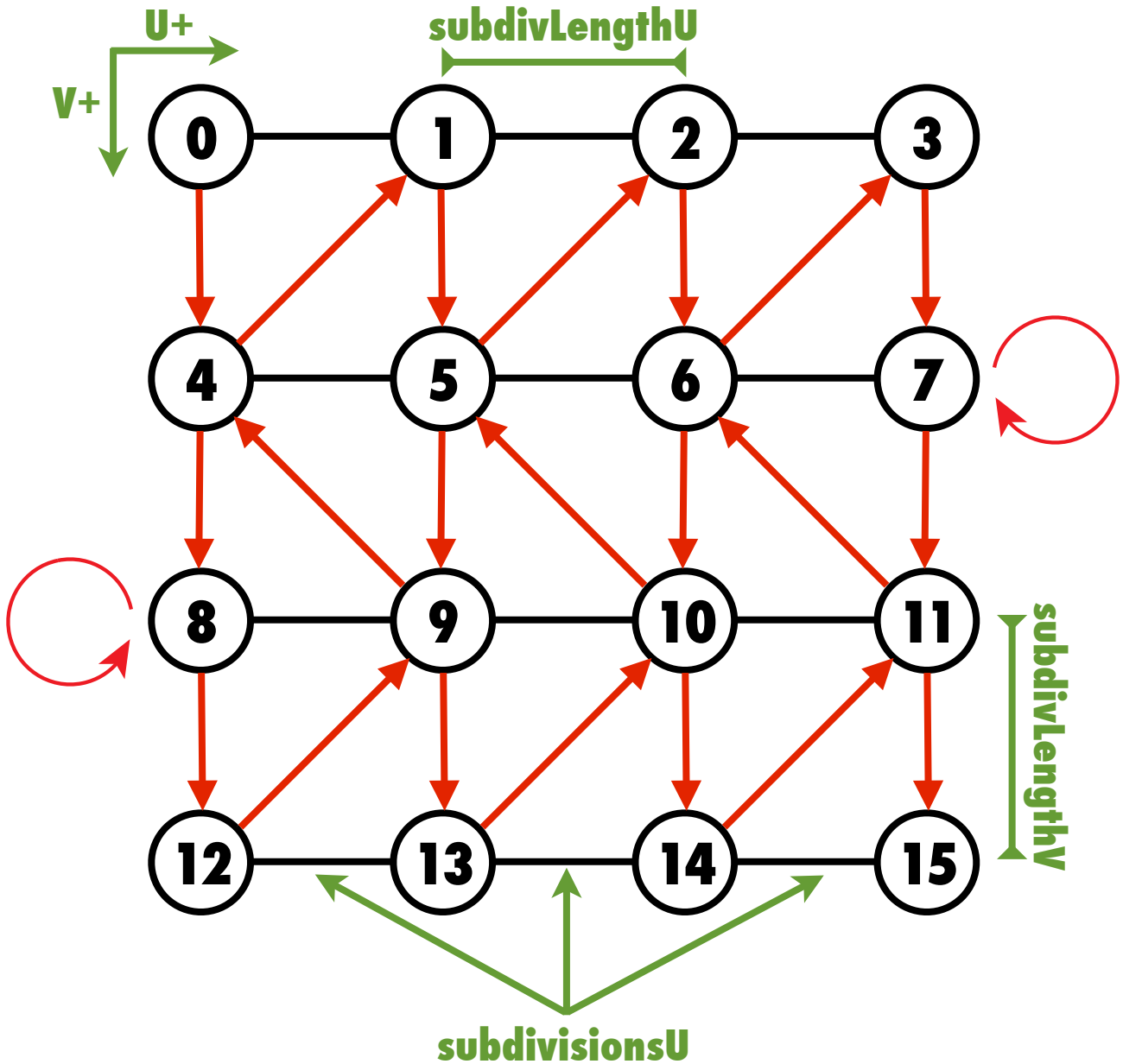
A Pattern for Generating Triangle Strip Meshes



How do we turn this into code?

Initial Parameters and State

```
subdivLengthU   = 100.0;
subdivLengthV   = 100.0;
subdivisionsU   = 3;
subdivisionsV   = 3;
reachedEnd      = false;
goRight         = true;
stepA           = true;
currU           = 0;
currV           = 0;
```



The Iteration (Pseudo-) Algorithm

```
while( end of mesh has not been reached ) {
    Add vertex for {currU, currV} indices.

    if( end of row has been reached ) {
        Add degenerate triangle.
        Reset/toggle state variables for next row.
        Check if current row is final one.
    }

    Based on current row and step type, set next {currU, currV} indices.
    Alternate step type for next iteration of while-loop.
}
```

The Iteration Algorithm

```
while( !reachedEnd ) {
    // Compute 2D XY coordinates from UV coordinates
    PVector currVert = new PVector( subdivLengthU * currU, subdivLengthV * currV, 0.0 );
    triStripContainer.addVertex( currVert );

    // Check for row ending
    if( !stepA && ( (goRight && currU == subdivisionsU + 1) || (!goRight && currU == 0) ) ) {
        // When turning around, we add the last point again as a pivot and reverse normal.
        triStripContainer.addVertex( currVert );
        triStripContainer.addVertex( currVert );
        // Reset to step type A
        stepA   = true;
        // Alternate between row types
        goRight = !goRight;
        // Check for end of mesh
        reachedEnd = (currV == subdivisionsV + 1);
    }

    // Compute next step in triangle strip UV pattern
    if( goRight ) {
        if(stepA) {
            currV++;
        }
        else {
            currU++;
            currV--;
        }
    }
    else {
        if(stepA) {
            currV++;
        }
        else {
            currU--;
            currV--;
        }
    }

    // Alternate between step types
    stepA = !stepA;
}
```

Class exercise: Walk the Algorithm!

(Iterate through the triangle strip algorithm on paper)

The two-dimensional UV coordinates allow us to situate a vertex within the two-dimensional space of a surface.

Once we've worked out how to compose a 2D triangle strip mesh, we can map the surface into three dimensional XYZ coordinates to form various 3D geometries including cylinders, cones, toruses and spheres.

Since much of the work in creating each type of geometry is common to each other type, we will begin to migrate the code we've developed towards a "Mesh Factory" class that can quickly and internally generate all mesh types and insert them into the scenegraph.

We'll also move the representation of a Triangle and a Vertex into separate classes so that the components of our data cross-reference one another. Later, as we apply our meshes to different geometric problems, this will help us access the necessary components in more intuitive ways.

Extending the Platform Architecture