Scan converting polygons

Primitives

Polygons:
- convex
- non-convex
- with holes
- with self-intersections

We focus on the convex case
Scan Conversion of Convex Polygons

General idea:
- decompose polygon into tiles
- scan convert each tile, moving along one edge

Convex Polygons

Scan convert a convex polygon:

```c
void ScanY (Vertex2D v[], int num_vertices, int bottom_index)

array of vertices in counterclockwise order
array size
number of the vertex with min. y coordinate
```

1. Find left edge of a tile:
   - go around **clockwise**, starting from v[bot], until find an edge such that it is not contained inside a scan line:

2. Similarly, find the right edge of a tile.
3. Scan convert all scan lines going from left to right edges
Convex Polygons

```c
void ScanY(Vertex2D v[], int num_vertices, int bottom_index) {

  // Initialize variables
  for(remaining_vertices = num_vertices;
      remaining_vertices > 0; )
  {
    // Find the left top row candidate
    // Determine the slope and starting x location for the left tile edge
    // Find the right top row candidate
    // Determine the slope and starting x location for the right tile edge
    for(row = bottom_row; row < left_top_row &&
        row < right_top_row; row++)
    {
      ScanX(ceil(left_pos),ceil(right_pos),row);
      left_pos += left_step;
      right_pos += right_step;
    }
    bottom_row = row;
  }
}
```

Initialization

- Keep track of the numbers of the vertices on the left and on the right:
  ```c
  int left_edge_end = bottom_index;
  int right_edge_end = bottom_index;
  ```

- This is the first row of a tile:
  ```c
  int bottom_row = ceil(v[bottom_index].y);
  ```

- These are used to store the candidates for the top row of a tile:
  ```c
  int left_top_row = bottom_row;
  int right_top_row = bottom_row;
  ```

- Keep track of the intersections of left and right edges of a tile with horizontal integer lines:
  ```c
  float left_pos, right_pos, left_step, right_step;
  ```

- Number of remaining vertices:
  ```c
  int remaining_vertices;
  ```

- A couple of auxiliary variables: int edge_start; int row;
Find a tile

Find the left top row candidate
while( left_top_row <= bottom_row && remaining_vertices > 0 )
{ Move to next edge:
  edge_start = left_edge_end;
  Be careful with C % operator, (N-1) % M will give -1 for
  N = 0, need to use (N+M-1) % M to get (N-1) mod M = N-1
  left_edge_end = (left_edge_end+num_vertices-1)%num_vertices;
  left_top_row = ceil(v[left_edge_end].y);
  remaining_vertices--; }

We found the first edge that sticks out over bottom_row
determine the slope and starting x location for the left tile edge.
if(left_top_row > bottom_row )
{ left_step = (v[left_edge_end].x - v[edge_start].x)/
             (v[left_edge_end].y - v[edge_start].y);
  left_pos = v[edge_start].x +
             (bottom_row-v[edge_start].y)*left_step;
}

Find a tile

Compute increment in y direction and starting/ending (left/right) point for the first scan of a tile
Find a tile

Find the right top row candidate; determine the slope and starting x location for the right tile edge. Exactly as for the left edge.

Scan convert a single row:

```c
void ScanX(int left_col, int right_col, int row, int R, int G, int B) {
    if (left_col < right_col) {
        for (int x = left_col; x < right_col; x++) {
            draw_pixel(x, y);
        }
    }
}
```

© 1999, Denis Zorin