6.837 F99 Assignment 5

Polygon Scan Conversion

Our tour of the classical graphics pipeline is nearly complete. Today, we considered 3D *scan-conversion* algorithms, which operate by considering each raster of the viewport in turn, and processing (and depth-ordering) *all* primitives which might contribute pixels to that raster. This assignment challenges you to implement a scan conversion algorithm for overlapping 3D triangles.

We have modified the Inventor `SceneViewer` by adding several buttons to the left of the viewport, to produce a new program called `ivscan`. While viewing an Inventor scene, pressing any of these buttons causes a series of C++ *callbacks* to occur. One callback per polygon edge is made to `EdgeRec::Init()`. One callback per frame is made to `RenderScene()`. We have supplied incomplete versions of both routines. The assignment challenges you to complete the critical inner operations of both, producing a semi-interactive polygon scan converter.

`EdgeRec::Init()` takes as input a single polygon `Edge` (with 2D screen-space vertices, eye-space z values, lighting information, etc., as defined in `Edge.h`). It is called **exactly once per edge for each triangle that survives clipping** (which we perform in `ScanWrap.C`). **You must fill in `EdgeRec::Init()` so that it correctly initializes each field of the `EdgeRec`.**

`RenderScene()` is passed an `Environment` variable, and a function `setPixel()`. The `Environment` data structure contains a viewport descriptor, a blank framebuffer, and a pointer to an `EdgeRecTable`. (The `EdgeRecTable` is one concrete implementation – an array of linked lists of `EdgeRecs` – of the abstract "Event Bucket" data structure described in class.) `RenderScene()` is called **exactly once for each rendered scene** (i.e., each time one of the scan conversion buttons in `ivscan` is pressed). Finally, you can use the function `setPixel()` to write a color value to a specified pixel of the framebuffer, as in Assignment 2.

Your job is to fill in `RenderScene()` so that it produces, through a series of calls to `RenderScanLine()` [and thus to `setPixel()`], a correct scan-converted image of the scene. Here, "correct" means two things: first, that at each pixel the correct triangle contributes its color at the pixel; and second, that the computed color there is the correct Gouraud interpolation of the colors at the vertices.

Extensive comments to guide you through both tasks can be found in the files `README`, `EdgeRec.C`, `EdgeRecList.C`, and `Scan.C`. **You need modify only these three `C` files to complete this assignment.**

The Point: First, you will implement the correct initialization of the `EdgeRec` and AEL (Active Edge List) data structures described in class. Second, you will implement the incremental maintenance along polygon edges of x, y, depth, and color in order to achieve correct scan conversion, visibility determination, and Gouraud interpolation of shading, respectively. Third, you will understand the (integer-centered) screen-space coordinate model, and implement a sampling rule which renders polygon meshes without dropouts or overdraws. Fourth, if you wish (i.e., for extra credit) you may also implement a more powerful scan converter; per-pixel Phong shading; real clipping; better interactivity; texture mapping, etc. See the `README` file for details.

Tools: On Athena, type (if necessary)

```
attach 6.837
```

Make a working directory, for example by doing

```
mkdir /working
```

Copy the assignment source there:

```
cp /mit/6.837/ivscan/ivscan.tar ~/working
```

Go to your working directory:

```
cd ~/working
```

Unpack the assignment source, producing directory `ivscan`:

```
tar xvf ivscan.tar
```

Go to the `ivscan` directory, read the `README`, and `make` the executable:

```
cd ivscan; cat README; make
```

Now you can run the (incomplete) executable :

```
ivscan foo.iv
```

where `foo.iv` is any inventor file.

A "reference executable" can be found at `/mit/6.837/ivscan/ivscan`.

Evaluation: We expect your `ivscan` to correctly scan-convert all of the included test Inventor files, **from any view point**. We will also test it against several more challenging Inventor objects.

To turn in (details TBA by email, and online):

- Your modified `EdgeRec.C`, `EdgeRecList.C`, and `Scan.C`;
- **Any other `.C` or `.h` files you modify**;
- Your **stripped** executable `ivscan`.

We will test your source for compilation using the assignment `Makefile`. Thus you **must** include any files that you have modified (we will supply unmodified files from the assignment source).

This assignment is due 5pm Friday, 15 October 1999.