Weekly Thesis-work Report

Javier Fernández

February 16, 2010

1 General topics addressed

- Adding *rm-tex.sh* script for deleting unnecesary trash files created by LaTeX
- Using *Endrov* flow view. Try general filters, emphasis on Thresholding.
- Rasterization and triangle Rasterization research
- Plugin implementation (basic)

2 Main per-topic Issues and solutions

2.1 Endrov flows

Summary

- The filters are working properly. Evaluating on Thresholding gives a clean input.
  Solution: Was necessary to select the environment/(the flow) to make the images findable.

Remaining problems

- Cannot view the result of the filtering, i.e. pass the output to a channel or a proper place.
  Error: `Trying to overwrite data that has not been autogenerated` ⇒ **Solved:** Take a channel a rename by just clicking on it and rewriting

2.2 Rasterization research

General ideas

- Rasterization: Process of rendering *vector* information to convert it into a raster format. vectorized image ⇒ bitmap.
  In a nutshell is the process of computing the mapping from scene geometry to pixels.
- Rasterization is based in shape by shape analysis and rendering. Ray tracing can involve more than one object at the same time and is faster to compute ilumination (not quite if GPU are well used).
- **Bresenham's Algorithm**:

- Supposing (x_0, y_0) and (x_1, y_1) the endpoints of a line. And the standard pixel convention: pixel increase in the down and right directions, then:

$$y = \frac{y_1 - y_0}{x_1 - x_0}(x - x_0) + y_0$$

  And this implies that every ideal y for succesive integer values of x can be computed as:

$$y_n = y0 + m * n \Rightarrow m = \frac{y_1 - y_0}{x_1 - x_0}$$

- There is an error value due to the vertical distance between the rounded and the exact y values. The error increases gradually by the slope. If the error exceeds 0.5, the rasterization y is increased by 1 (next row) and the error decremented by 1.0.
  This is only for right-down direction
- **Generalization**:
  * If lines goes left-down ($x_0 > x_1$) instead of right-down ($x_0 < x_1$) then $swap(x0, x1)$
  * If lines goes up then check $y_0 > y_1$ and if true then step -1 instead of 1
  * If slope m is greater than 1 then reflect the steep line across the line $y = x$ to obtain a line with a small slope. To do this $swap(x_0, y_0)$ $swap(x_1, y_1)$ perform operations and $plot(y, x)$

Papers and links

- **Scan-conversion:** `http://www.devmaster.net/articles/software-rendering/part3.php` Drawing a polygon as a set of horizontal lines.
- **Basic T.Rasterization:** `http://joshbeam.com/articles/triangle_rasterization/` Basic triangle rasterization C++ (non-beautiful code)
- **Bresenham's line algorithm** `http://en.wikipedia.org/wiki/Bresenham%27s_line_algorithm`

3 Next-week suggestions

- Complete the plugin implementing the triangle rasterization
  - Try using int as triangle points
  - Design a good class scheme