TINYRASTER TECHNICAL REPORT

J.Pritchard – U1661665

Report.

Within this report, details on the running of this TinyRaster implementation can be found along with an analysis and references used in its creation.

The core of operations lies in DrawLine2D, which is an implementation of the common Bresenham line drawing algorithm. This is extended to calculate the colour of each pixel along the line for interpolation and is capable of then alpha blending the colour with an existing colour on screen to draw translucent shapes. The function then accounts for thickness, adding pixels to the side of the line if the gradient is between -1 and 1, above and below if not.

DrawUnfilledPolygon2D calls DrawLine2D for each pair of vertices provided to create the bounds of the polygon specified.

ScanLineFillPolygon2D creates a scanline LookUp Table, populates it and draws the results. It's capable of interpolating colour due to colour calculations made in 3c. The function itself is split into a few sections.

1. DrawUnfilledPolygon2D is called to get the outline of the shape.
2. Each vertex is checked for suitability to be acknowledged by the LUT. Read: Has one adjacent vertex below it and the other above it along with not being on the same Y as either adjacent vertex. This result is stored in an array of bool xORAdjacentVertexIsHigher.
3. The LUT is populated.
    a. Each scanline checks every edge (pair of vertices) for an intersection point.
    b. The intersection point eligibility is checked.
        i. If the intersection point is one of the vertices of the edge being checked, the xORAdjacentVertexIsHigher array is consulted for that point's eligibility.
    c. If the point can be counted, the colour is calculated then both it and the position stored for sorting.
4. The LUT is sorted using an insertion sort and each item pushed back into its correct position in an std::vector.
5. From bottom to top, each pair of vertices in the LUT is then drawn using DrawLine2D.

DrawCircle2D dynamically calculates the number of segments of a circle needed (the implementation forces at least 16), then using the parametric equation of a circle to calculate all the points in octant 2 of the circle. Reflections are then used to calculate octant 1, then 7 and 8, then the entire left side of the circle. The resulting array of vertices is then passed to either DrawUnfilledPolygon2D or ScanLineFillPolygon2D depending on what's required.

On reflection, there are areas in which this work could be improved.

DrawLine2D could have a capping function as right now, lines with a thickness of greater than one have an *incorrect* cap. Due to the implementation of DrawLine2D, any polygons created using it are subject to bad line connections. ScanLineFillPolygon2D could use more refinement to prevent an alpha blending bug present due to the function overwriting existing pixels unnecessarily. DrawCircle2D also needs a better method for the segment calculation.

However, the program as a whole is robust and capable of most tasks required. It won't struggle to accomplish anything, however still has minor details needing fixing.

References.

Geeks for Geeks. (n.d.). *Insertion Sort*. Retrieved from: https://www.geeksforgeeks.org/insertion-sort/

Minsi, C., (2018), *Lecture 14: Raster Graphics and 2D Line Drawing*, Retrieved from https://unilearn.hud.ac.uk/bbcswebdav/pid-2328303-dt-content-rid-3660854_1/courses/CFT2112-1718/CFT2112_Lec14-s.pdf Alternative link: https://unilearn.hud.ac.uk/webapps/blackboard/content/listContent.jsp?course_id=_29589_1&content_id=_2328302_1

Minsi, C., (2018), *Lecture 15-01: Polygons and Circles*, Retrieved from: https://unilearn.hud.ac.uk/bbcswebdav/pid-2332419-dt-content-rid-3675193_1/courses/CFT2112-1718/CFT2112_Lec15-s.pdf Alternative link: https://unilearn.hud.ac.uk/webapps/blackboard/content/listContent.jsp?course_id=_29589_1&content_id=_2332413_1

Minsi, C., (2018), *Lecture 15-02: Polygon Filling*, Retrieved from: https://unilearn.hud.ac.uk/bbcswebdav/pid-2332420-dt-content-rid-3675198_1/courses/CFT2112-1718/CFT2112_Lec16-s.pdf Alternative link: https://unilearn.hud.ac.uk/webapps/blackboard/content/listContent.jsp?course_id=_29589_1&content_id=_2332413_1

Minsi, C., (2018), *CFT2112: Porting Bresenham Line Drawing to TinyRaster*, Retrieved from: https://unilearn.hud.ac.uk/bbcswebdav/pid-2332837-dt-content-rid-3675795_1/courses/CFT2112-1718/line_drawing.pdf Alternative link: https://unilearn.hud.ac.uk/webapps/blackboard/content/listContent.jsp?course_id=_29589_1&content_id=_2332413_1

Minsi, C., (2018), *Lecture 16: Colour Representations and Interpolated Filling*, Retrieved from: https://unilearn.hud.ac.uk/bbcswebdav/pid-2334072-dt-content-rid-3679848_1/courses/CFT2112-1718/CFT2112_Lec16.pdf Alternative link: https://unilearn.hud.ac.uk/webapps/blackboard/content/listContent.jsp?course_id=_29589_1&content_id=_2334053_1

Minsi, C., (2018), *Tutorial Notes: Scanline Filling Polygons*, Retrieved from: https://unilearn.hud.ac.uk/bbcswebdav/pid-2334073-dt-content-rid-3679849_1/courses/CFT2112-1718/scanline_fill.pdf Alternative link: https://unilearn.hud.ac.uk/webapps/blackboard/content/listContent.jsp?course_id=_29589_1&content_id=_2334053_1

Minsi, C., (2018), *Lecture 17: Colour Blending*, Retrieved from: https://unilearn.hud.ac.uk/bbcswebdav/pid-2335227-dt-content-rid-3684517_1/xid-3684517_1 Alternative link: https://unilearn.hud.ac.uk/webapps/blackboard/content/listContent.jsp?course_id=_29589_1&content_id=_2335219_1