Virtual Lab Computer Science and Engineering

IIIT Hyderabad: Computer Graphics LAB

JUIT Lab: Computer Graphics LAB

(11BWCI671)

1. Hierarchical Transformations: 3D Articulated Arm
2. Rasterization : Polygon

---

1. Hierarchical Transformations: 3D Articulated Arm

This experiment demonstrates the end effect of hierarchical transformations learnt in the previous experiment using a 3D Articulated Arm setup.

Theory

Here in this experiment we put the concepts of multiple transformations learnt in *experiment 5a* to practice with an Articulated arm experiment.

A hierarchy is an organization of things into levels. In a very general sense, he things higher up in a hierachy have control over the lower things. Here this applies to the matrix transformations.

We constructed a hierarchical articulated arm model where each node/part follows a series of transformations from the root node to its current position. The top-most object in this hierarchy is the Shoulder followed by Elbow, Forearm, Wrist and Palm. Moving the shoulder moves the rest of the parts as all others derive their transformation from the shoulder. When you move the wrist only the palm moves while the shoulder, elbow and forearm remain unchanged. This is because the wrist is lower in hierarchy than your elbow and therefore has no control on it.

Objective

Objective of this experiment is to understand the effective tranformation due to a series of transformations and how to construct a hierarchical model.

Experiment

1. Please download the appropriate Virtual Lab Graphics package below.
   Download
2. Save the **VirtualLabGraphics.zip** file and double click it to extract.
3. Run the Start.jar in the extracted folder (VirtualLabGraphics folder) to start the experiments.
   - Windows/MacOSX users can start the experiment by double clicking the Start.jar file.

- Linux users should run Start.jar by executing the following command in the terminal.
  Change to the Virtual Lab Directory.
  $ *cd VirtualLabGraphics/*
  Execute the experiment using the command:
  $ *java -jar Start.jar*

4. **Alternate Method:** If you are not able to start the experiment by the above procedures, you can try the alternate method given below.
   - Alternatively, Linux/MacOSX users can start the experiment by as described below.
     Change to the Virtual Lab Directory.
     $ *cd VirtualLabGraphics/*
     Execute the experiment using the command:
     $ *./Experiment.sh 5b*
   - Alternatively, Windows users can start the experiment by double clicking the Experiment5b.bat file.

Procedure

This experiment introduces the concept of the hierarchical transformations and a hierarchical model.

The hierarchical articulated arm is provided. Notice the different nodes used to construct the model. The nodes in this model follows the hierarchy: Shoulder (highest), Elbow, Forearm, Wrist and Palm. Applying transformations to a node in the hierarchy applies the same transformations to all the lower nodes in the hierarchy in addition to their own transformations. For example, a rotation has been applied to the shoulder. As you drag the slider you can notice this rotation being applied to the other nodes toother nodes.

Try modifying the transformations in the hierarchy and notice the effective changes on the entire model.

Open the transormation nodes of Forearm and Palm and notice how a Cube has been moulded to a forearm and a palm using the respective transformations.

2. Rasterization : Polygon

In digital display systems, everything that is displayed, is displayed in terms of a smallest unit of display, which is called a pixel. This is in contrast to what we perceive the world to be as continuous. The space occupied by any image in digital display system is measured in terms of pixels. Thus we require to transform the continuous space in which we define he geometry of any figure to a discrete space for display in digital displays. This transformation is called **rasterization** or **scan conversion**

When a polygon is transformed from a set of edges in the continous form of $y = mx + c$ into a set of pixels occupied by the interior of the polygon, this transformation is called **polygon rasterization**. Thus when we apply polygon rasterization technique on a polygon, we obtain the set of pixels that are required to be filled in order to fill the entire interior area bounded by the edges of the polygon in the best possible manner. Some common algorithms to perform polygon rasterization are as follows:

1. Scan-Line Polygon Fill algorithm
2. Seed Fill or Boundary Fill algorithm

Here we have discussed only the Scan-Line Polygon Fill algorithm.

Theory

Scan Line Polygon Fill Algorithm

Pseudo Code of the algorithm

Input : Vertices of the polygon, (x_i, y_i) $\forall i = \{1, 2, ..., n\}$, n = #vertices in the polygon. We have the following data structures to maintain data:

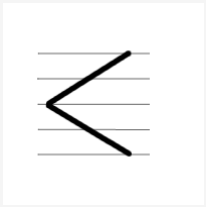
- Edge Table (ET) : Contains all the edges (except horizontal edges) of the polygon sorted by their smaller y-coordinates. All the edges having equal smaller y-coordinates are kept in the same bucket, where they are sorted by the corresponding x-coordinate.
  In each cell, the following information is maintained:
  i. y_upper : last scanline to consider for this edge
  ii. x_lower : corresponding x coordinate for the smaller value y value of the edge
  iii. 1/m : to compute the corresponding x coordinate of the edge for each scanline
- Active Edge Table (AET) : Contains the edges that intersect with the current scan line. For each scan line, the edges are sorted in an increasing order of the x coordinate of the point of intersection.
  In each cell, the following information is maintained:
  i. y_upper : last scanline to consider for this edge
  ii. x_lower : x coordinate value of the edge for the present scan line
  iii. 1/m : to compute the corresponding x coordinate of the edge for each scanline

Steps of the Algorithm

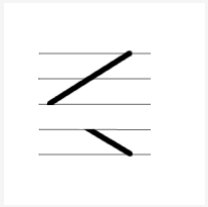
1. AET is initially empty.
2. scan_value = Value of y-coordinate for the first nonempty bucket. This is the first scan line from where filling will start.
3. Do
   - From ET[scan_value] bucket, all entries are inserted in AET which satisfy the condition : y_{min} = scan_value. This is to include the new edges which have their smaller y value equal to the present scan line.

- From AET, remove the entries which satisfy the condition, y_{max} = scan_value, i.e., remove the edges not intersecting with the next scanline
- Please note that there will be one entry for edges in Fig. 1 and two entries for edges in Fig. 2

- Figure 1 :



changes to



- Figure 2 :

- Sort all the entries in AET w.r.t. the x-coordinate of the intersection point of the edge with the present scan line.
- Fill the pixels in the present scanline from x-values mentioned in the odd entries of AET to the x-values mentioned in the next even entry of AET.

- If there are n entries in AET (n will always even), then the following pixels will be filled :

  - 1^{st} entry to 2^{nd} entry
  - 3^{rd} entry to 4^{th} entry
  - ...
  - $(n-1)^{th}$ entry to n^{th} entry

- Increment y by 1 (to the coordinate of the next scanline).
- For each nonvertical edge remaining in the AET, update x for the new y.

Until AET and ET are non empty

Objective

Objective of this experiment is to understand the steps of filling a polygon in a 2D frame buffer. Given the vertices of the polygon, the experiment outputs a filled polygon. Here we have demonstrated the steps of Scan Line Poylgon Fill algorithm.

Procedure

1. Fix the size of the frame buffer (the default parameters are already filled in the input boxes) and then click on **Enter**. You can also proceed directly to the experiment using the default parameters of the size of frame buffer and the vertices of a polygon, by clicking on **Start Experiment with Default Values**.
2. Enter the vertices of the polygon in the input box.
   Let, (x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5)
   represent the vertices of the polygon in order, i.e., the 1st vertex is connected to the 2nd and 2nd to the 3rd and so on. The last vertex is connected to the 1st to complete the polygon.
   The default vertices are already filled in the input boxes. Once you have finalized the coordinates, you can click on **Start Experiment** to begin with the experiment.
   You can also go **Back** to change the values of the frame buffer.
3. Throughout the experiment, you can click on **Next Iteration** to go tothe next step of the algorithm and **Previous Ieration** to go to the previous step.
4. Once the experiment ends, you can again experiment with a new set of vertices of the polygon.