GRAPHICS
FINAL
ASSIGNMENT

TEXTURE MAPPING

Gautam Kumar 07CS1021
Naveen Kr. Sharma 07CS3027
Problem Statement:

Map a Texture to a Surface of Revolution generated by a Uniform Non-Rational B-Spline.

Features:

- Zoom in, zoom out, and panning
- Object translation and rotation
- Color palette for object/surface and illumination
- Specification of Illumination (3 light sources: provisional; light coordinates)

Extra Features:

- Saving the Rendered Image to a local file on the host’s machine.
- Changing the Material Properties, such as Shininess, Ambient Color etc.
- Wireframe model of the object.
- Triangulated Mesh of the object.
- Incorporation of 4 Different Light Sources and option to change their colors/position.
- Changing transparency attributes of the material.
Major Classes:

**BSpline:**
Implements the algorithms to obtain the spline.

Given 4 control points, it creates the spline. And also stores normal information that is used during surface computation.

It also has a method to return the spline in finite straight line segments of length N each.

**BresenhamAlgorithm:**
This class implements the Bresenham Algorithm and obtains a digital line between two GridPoints.

Main Functions Include:

- **getDDAWithStandardProperties():** gets a DDA line between two GridPoints a and b, such that a is left of b, and the slope of the line ab lies between 0 and 1.
- **getDDAWithNumPixelsKnowWithStProperties():** gets a DDA starting from a point x, y, given dx and dy for the line, such that the number of pixels in the digital line are known.
- **getDDA():** gets a line between two GridPoints with no constraint on them.
- **getThickDDA():** gets a DDA line between two GridPoints and a given thickness.

**Canvas**
Contains Various Parameters like gridSeparation, gridColor and axesColor and methods to draw pixels on the canvas.

**Rendering**
This class implements the creation and rendering of textured surface in Java3D environment.
- **createSurface (BSpline[] arr)**: Given the points on the approximated circle of revolution and approximated BSpline, this method constructs the surface using quads (ordered tuple of 4 points and the normals associated with them).
- **createSceneGraph()**: This method adds functionalities such as Rotation, Translation, Zoom In/Out of the rendered surface.
- **initLights()**: This method initializes the lighting conditions on the surface. It applies an Ambient Light Source, and a single Point Light Source. One more Point Light Source is provided which is turned OFF by default.
- **createAppearance()**: Sets the appearance attributes of the surface, such as Diffuse Color, Ambient Color, Specular Color and Shininess. It also defines how texture is to be applied on the object.
- **repeatImage (BufferedImage img)**: This method tiles the given texture side by side and creates a large enough image (wallpaper) to cover the whole surface.

### SplineCanvas
This class extends the original Canvas, to include spline related functions such as:

- **isXMonotone()**: checks the x-monotonicity of the set of clicked points.
- **isYMonotone()**: checks the y-monotonicity of the set of clicked points.
- **isMonotonicityMaintained (GridPoint p)**: checks the y-monotonicity of the set of points, when a new point is added.
- **recompute()**: recomputes the spline from the set of points.
- **modifySplinesWRT (GridPoint currentGridPoint)**: locally recomputes the part of spline being affected by the change in that gridPoint.
- **drawControlLines()**: draws the BresenhamLine joining the various clicked points in order.
- **getSplinesAsOne()**: combines a set of splines to get a single spline.

### Voxel
A GridPoint in 3 Dimension, with the added data of outwards normal direction (nX, nY, nZ).

### Major Steps of the Algorithm:

#### Spline Drawing:
We store all the points clicked by the user in an array of GridPoints. If the number of clicked points is more than 3, then we draw Uniform Non-Rational B-Spline(s) taking 4 points at a time.

The parametric curve is generated using the following equation:
\[ Q(t + t_i) = T_i \times M_{Bs} \times G_{Bsi} \]

\[ = \frac{(1-t)^3}{6} P_{i-3} + \frac{(3t^3 - 6t^2 + 4)}{6} P_{i-2} + \frac{(-3t^2 + 3t + 1)}{6} P_{i-1} + \frac{t^3}{6} P_i \]

where \( 0 \leq t < 1 \)

Here, \( T_i \) is defined as the row vector \( [(t - t_i)^3 \quad (t - t_i)^2 \quad (t - t_i) \quad 1] \)

\( M_{Bs} \) is the B-spline basis matrix, which relates the geometric constraints to the blending functions and the polynomial coefficients.

\[
M_{Bs} = \frac{1}{6} \begin{bmatrix}
-1 & 3 & -3 & 1 \\
3 & -6 & 3 & 0 \\
-3 & 0 & 3 & 0 \\
1 & 4 & 1 & 0 \\
\end{bmatrix}
\]

\( G_{Bsi} \) is the B-spline geometry vector.

\[
G_{Bsi} = \begin{bmatrix}
P_{i-3} \\
P_{i-2} \\
P_{i-1} \\
P_i \\
\end{bmatrix}
\]

\( P_i \)'s are the control points (1x3 matrix containing the x,y,z coordinates).

**Surface of Revolution**

**Generation of circle points and the normals at those points.**

We approximate the real circle of revolution, to be an ‘n’ sided polygon, where n is quite large. The appropriate value of ‘n’ for the surface is obtained from the maximum X-coordinate of B-spline, using the given formula:

\[
n = \frac{\pi}{\sin^{-1}\left(\frac{2}{\text{Max}\{x_i\}}\right)}
\]

This expression results in polygons with moderate side-lengths (neither too large, nor too small).
Decomposing spline into several straight line segments of equal length.
To get the decomposed spline, we remove every alternate point on the spline. This gives us an approximate BSpline from which we can generate the Quads composing the surface.

Generating Quads and creating surface.
After obtaining the approximated circles and approximated spline, we join the corresponding vertices of adjacent circles to get the quads. These quads are generated along the surface in circular order, spiraling downwards in anticlockwise manner.

Texture Mapping
Once the surface has been generated using the quads, we tile the given texture till we get an image large enough to cover the whole surface. Then we map each quad of the surface to portions on the texture image (generated wallpaper) in proper order, so as to maintain the continuity of texture in between the quads.
SCREENSHOTS
GRID ON
Drawing a Spline
Surface of Revolution Render
Rotation of the Render
Translation of the Render
Zoom - In
Showing The Quads
Triangulation obtained From Quads
Texture Menu
Changing the Texture
Loading Texture From File
Selecting Turtle Image
Cracked Window Texture
Light Menu
Adding a Light Source
Changing Light Color
Effect of Changing Light Color
A Diff Texture with 3 Light Sources
Material Properties

- Change Ambient Color
- Change Diffuse Color
- Change Emissive Color
- Change Specular Color
- Change Shininess
- Enable Transparency
- Change Transparency Value
Changing Specular Color
Effect of changing Specular Color
Changing Ambient Color
Enabling Transparency
Changing Transparency Value
Very High Transparency
Saving the Render