Angelo Daniel A. Dela Paz

4CSC

Machine Problem 2

**Cross using Lines and Quadratic Bezel**

**Python**

1. # Set up the plot
2. fig, ax = plt.subplots()
3. ax.set\_aspect("equal")
4. ax.set\_xlim(100, 500)
5. ax.set\_ylim(100, 550)
6. ax.set\_facecolor((240 / 255, 240 / 255, 240 / 255))
7. # Quadratic Bezier curve function
8. def quadratic\_bezier(t, p0, p1, p2):
9. x = (1 - t) \*\* 2 \* p0[0] + 2 \* (1 - t) \* t \* p1[0] + t\*\*2 \* p2[0]
10. y = (1 - t) \*\* 2 \* p0[1] + 2 \* (1 - t) \* t \* p1[1] + t\*\*2 \* p2[1]
11. return x, y
12. # Draw the cross using lines and quadratic Bezier curves
13. # Top part
14. ax.plot([250, 250], [280, 180], "k-", linewidth=4)
15. ax.plot([350, 350], [180, 280], "k-", linewidth=4)
16. ax.plot([290, 310], [140, 140], "k-", linewidth=4)
17. points = np.array(
18. [
19. quadratic\_bezier(t, (250, 180), (250, 140), (290, 140))
20. for t in np.linspace(0, 1, 101)
21. ]
22. )
23. ax.plot(points[:, 0], points[:, 1], "k-", linewidth=4)
24. points = np.array(
25. [
26. quadratic\_bezier(t, (310, 140), (350, 140), (350, 180))
27. for t in np.linspace(0, 1, 101)
28. ]
29. )
30. ax.plot(points[:, 0], points[:, 1], "k-", linewidth=4)
31. # Right part
32. ax.plot([350, 450], [280, 280], "k-", linewidth=4)
33. ax.plot([490, 490], [320, 340], "k-", linewidth=4)
34. ax.plot([350, 450], [380, 380], "k-", linewidth=4)
35. points = np.array(
36. [
37. quadratic\_bezier(t, (450, 280), (490, 280), (490, 320))
38. for t in np.linspace(0, 1, 101)
39. ]
40. )
41. ax.plot(points[:, 0], points[:, 1], "k-", linewidth=4)
42. points = np.array(
43. [
44. quadratic\_bezier(t, (490, 340), (490, 380), (450, 380))
45. for t in np.linspace(0, 1, 101)
46. ]
47. )
48. ax.plot(points[:, 0], points[:, 1], "k-", linewidth=4)
49. # Bottom part
50. ax.plot([250, 250], [380, 480], "k-", linewidth=4)
51. ax.plot([290, 310], [520, 520], "k-", linewidth=4)
52. ax.plot([350, 350], [380, 480], "k-", linewidth=4)
53. points = np.array(
54. [
55. quadratic\_bezier(t, (250, 480), (250, 520), (290, 520))
56. for t in np.linspace(0, 1, 101)
57. ]
58. )
59. ax.plot(points[:, 0], points[:, 1], "k-", linewidth=4)
60. points = np.array(
61. [
62. quadratic\_bezier(t, (310, 520), (350, 520), (350, 480))
63. for t in np.linspace(0, 1, 101)
64. ]
65. )
66. ax.plot(points[:, 0], points[:, 1], "k-", linewidth=4)
67. # Left part
68. ax.plot([250, 150], [280, 280], "k-", linewidth=4)
69. ax.plot([110, 110], [320, 340], "k-", linewidth=4)
70. ax.plot([250, 150], [380, 380], "k-", linewidth=4)
71. points = np.array(
72. [
73. quadratic\_bezier(t, (150, 280), (110, 280), (110, 320))
74. for t in np.linspace(0, 1, 101)
75. ]
76. )
77. ax.plot(points[:, 0], points[:, 1], "k-", linewidth=4)
78. points = np.array(
79. [
80. quadratic\_bezier(t, (110, 340), (110, 380), (150, 380))
81. for t in np.linspace(0, 1, 101)
82. ]
83. )
84. ax.plot(points[:, 0], points[:, 1], "k-", linewidth=4)
85. plt.show()

**OUTPUT**

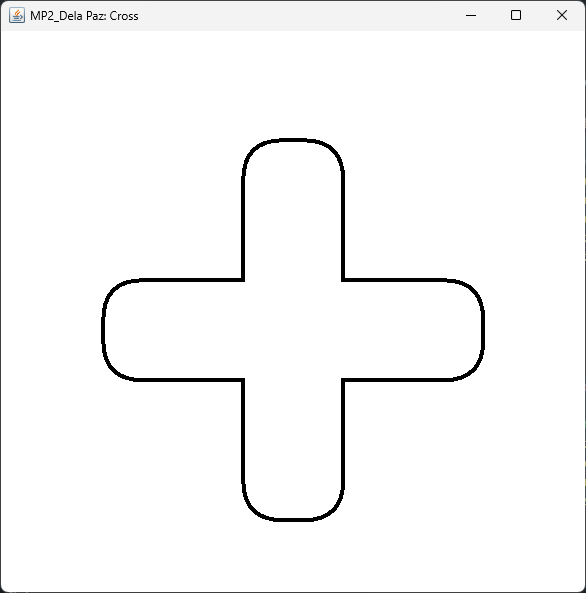
A black line on a white background

Description automatically generated

**Java**

1. import java.awt.\*;
2. import java.awt.geom.QuadCurve2D;
3. public class MP2\_DelaPaz extends Frame {
4. private void QuadraticBezier(Graphics2D g2d, int x0, int y0, int x1, int y1, int x2, int y2) {
5. QuadCurve2D q = new QuadCurve2D.Float();
6. q.setCurve(x0, y0, x1, y1, x2, y2);
7. g2d.draw(q);
8. }
9. public void paint(Graphics g) {
10. Graphics2D g2d = (Graphics2D) g;
11. // Set the stroke for drawing
12. g2d.setStroke(new BasicStroke(4));
13. g2d.setColor(Color.BLACK);
14. // Draw the cross using lines and quadratic Bezier curves
15. // Upper Curve
16. g2d.drawLine(250, 280, 250, 180);
17. g2d.drawLine(350, 180, 350, 280);
18. g2d.drawLine(290, 140, 310, 140);
19. QuadraticBezier(g2d, 250, 180, 250, 140, 290, 140);
20. QuadraticBezier(g2d, 310, 140, 350, 140, 350, 180);
21. // Right Curve
22. g2d.drawLine(350, 280, 450, 280);
23. g2d.drawLine(490, 320, 490, 340);
24. g2d.drawLine(350, 380, 450, 380);
25. QuadraticBezier(g2d, 450, 280, 490, 280, 490, 320);
26. QuadraticBezier(g2d, 490, 340, 490, 380, 450, 380);
28. // Bottom Curve
29. g2d.drawLine(250, 380, 250, 480);
30. g2d.drawLine(290, 520, 310, 520);
31. g2d.drawLine(350, 380, 350, 480);
32. QuadraticBezier(g2d, 250, 480, 250, 520, 290, 520);
33. QuadraticBezier(g2d, 310, 520, 350, 520, 350, 480);
35. // Left Curve
36. g2d.drawLine(250, 280, 150, 280);
37. g2d.drawLine(110, 320, 110, 340);
38. g2d.drawLine(250, 380, 150, 380);
39. QuadraticBezier(g2d, 150, 280, 110, 280, 110, 320);
40. QuadraticBezier(g2d, 110, 340, 110, 380, 150, 380);
41. }
42. // Main Method
43. public static void main(String[] args) {
44. Frame frame = new MP2\_DelaPaz();
45. frame.setTitle("MP2\_Dela Paz: Cross");
46. frame.setSize(600, 600);
47. frame.setBackground(Color.WHITE);
48. frame.setForeground(Color.BLACK);
49. frame.setVisible(true);
50. }
51. }

**Output**

****

**Cross using General Path**

**Python**

1. verticesA = [
2. (250, 280),  # Starting point
3. (250, 380),  # Straight line
4. (250, 420),  # Curve 1
5. (290, 420),  # Curve 1
6. (310, 420),  # Straight line
7. (350, 420),  # Curve 2
8. (350, 380),  # Curve 2
9. (350, 280),  # Ending point
10. ]
11. verticesB = [
12. (350, 280),  # Starting point
13. (450, 280),  # Straight line
14. (490, 280),  # Curve 1
15. (490, 240),  # Curve 1
16. (490, 220),  # Straight line
17. (490, 180),  # Curve 2
18. (450, 180),  # Curve 2
19. (350, 180),  # Ending point
20. ]
21. verticesC = [
22. (350, 180),  # Starting point
23. (350, 80),
24. (350, 40),
25. (310, 40),
26. (290, 40),
27. (250, 40),
28. (250, 80),
29. (250, 180),  # Ending point
30. ]
31. verticesD = [
32. (250, 180),  # Starting point
33. (150, 180),
34. (110, 180),
35. (110, 220),
36. (110, 240),
37. (110, 280),
38. (150, 280),
39. (250, 280),  # Ending point
40. ]
41. codes = [
42. path.Path.MOVETO,
43. path.Path.LINETO,
44. path.Path.CURVE3,
45. path.Path.CURVE3,
46. path.Path.LINETO,
47. path.Path.CURVE3,
48. path.Path.CURVE3,
49. path.Path.LINETO,
50. ]
51. pathA = path.Path(verticesA, codes)
52. pathB = path.Path(verticesB, codes)
53. pathC = path.Path(verticesC, codes)
54. pathD = path.Path(verticesD, codes)
55. patchA = patches.PathPatch(pathA, facecolor="none", lw=2)
56. patchB = patches.PathPatch(pathB, facecolor="none", lw=2)
57. patchC = patches.PathPatch(pathC, facecolor="none", lw=2)
58. patchD = patches.PathPatch(pathD, facecolor="none", lw=2)
59. fig, ax = plt.subplots()
60. ax.add\_patch(patchA)
61. ax.add\_patch(patchB)
62. ax.add\_patch(patchC)
63. ax.add\_patch(patchD)
64. ax.set\_xlim(100, 500)
65. ax.set\_ylim(0, 500)
66. plt.gca().set\_aspect("equal", adjustable="box")
67. plt.show()

**OUTPUT**

A black and white cross

Description automatically generated

**Java**

1. import java.awt.\*;
2. import java.awt.geom.GeneralPath;
3. public class MP2\_Cross extends Frame {
4. public void paint(Graphics g) {
5. Graphics2D g2d = (Graphics2D) g;
6. BasicStroke bs = new BasicStroke(10.0f);
7. g2d.setStroke(bs);
8. GeneralPath cross = new GeneralPath();
9. // Start
10. cross.moveTo(250, 280); // A
11. // Left Arm
12. cross.lineTo(150, 280);
13. cross.quadTo(110, 280, 110, 320);
14. cross.lineTo(110, 340);
15. cross.quadTo(110, 380, 150, 380);
16. cross.lineTo(250, 380);
17. // Bottom Arm
18. cross.lineTo(250, 480);
19. cross.quadTo(250, 520, 290, 520);
20. cross.lineTo(310, 520);
21. cross.quadTo(350, 520, 350, 480);
22. cross.lineTo(350, 380);
23. // Right Arm
24. cross.lineTo(450, 380);
25. cross.quadTo(490, 380, 490, 340);
26. cross.lineTo(490, 320);
27. cross.quadTo(490, 280, 450, 280);
28. cross.lineTo(350, 280);
29. // Top Arm
30. cross.lineTo(350, 180);
31. cross.quadTo(350, 140, 310, 140);
32. cross.lineTo(290, 140);
33. cross.quadTo(250, 140, 250, 180);
34. cross.lineTo(250, 280);
35. g2d.draw(cross);
36. }
37. public static void main(String[] args) {
38. Frame frame = new MP2\_Cross();
39. frame.setTitle("MP2: Cross");
40. frame.setSize(600, 600);
41. frame.setBackground(Color.WHITE);
42. frame.setForeground(Color.BLACK);
43. frame.setVisible(true);
44. }
45. }

**OUTPUTA screenshot of a computer

Description automatically generated**

**Christmas Tree**

**Python**

1. verticesBase = [(285, 100), (315, 100), (315, 70), (285, 70), (285, 100)]
2. verticesTree = [
3. (300, 100),
4. (150, 100),
5. (220, 250),
6. (200, 250),
7. (270, 400),
8. (250, 400),
9. (300, 550),
10. (350, 400),
11. (330, 400),
12. (400, 250),
13. (380, 250),
14. (450, 100),
15. (300, 100),
16. ]
17. verticesStar = [(300, 550), (280, 570), (300, 590), (320, 570), (300, 550)]
18. codesTrees = [
19. path.Path.MOVETO,
20. path.Path.LINETO,
21. path.Path.LINETO,
22. path.Path.LINETO,
23. path.Path.LINETO,
24. path.Path.LINETO,
25. path.Path.LINETO,
26. path.Path.LINETO,
27. path.Path.LINETO,
28. path.Path.LINETO,
29. path.Path.LINETO,
30. path.Path.LINETO,
31. path.Path.CLOSEPOLY,
32. ]
33. codesBase = [
34. path.Path.MOVETO,
35. path.Path.LINETO,
36. path.Path.LINETO,
37. path.Path.LINETO,
38. path.Path.CLOSEPOLY,
39. ]
40. codesTree = [
41. path.Path.MOVETO,
42. path.Path.LINETO,
43. path.Path.LINETO,
44. path.Path.LINETO,
45. path.Path.LINETO,
46. path.Path.LINETO,
47. ]
48. codesStar = [
49. path.Path.MOVETO,
50. path.Path.LINETO,
51. path.Path.LINETO,
52. path.Path.LINETO,
53. path.Path.CLOSEPOLY,
54. ]
55. pathBase = path.Path(verticesBase, codesBase)
56. pathTree1 = path.Path(verticesTree, codesTrees)
57. pathStar = path.Path(verticesStar, codesStar)
58. patchBase = patches.PathPatch(pathBase, edgecolor="Gray", facecolor="Gray", lw=1)
59. patchTree1 = patches.PathPatch(
60. pathTree1, edgecolor="YellowGreen", facecolor="YellowGreen", lw=1
61. )
62. patchStar = patches.PathPatch(pathStar, edgecolor="Orange", facecolor="Orange", lw=1)
63. fig, ax = plt.subplots()
64. ax.add\_patch(patchBase)
65. ax.add\_patch(patchTree1)
66. ax.add\_patch(patchStar)
67. ax.set\_xlim(0, 600)
68. ax.set\_ylim(0, 600)
69. plt.gca().set\_aspect("equal", adjustable="box")
70. plt.show()

**OUTPUT**

A green and orange tree

Description automatically generated

* 1. **Java**

1. import java.awt.\*;
2. import java.awt.geom.Area;
3. import java.awt.geom.GeneralPath;
4. public class MP2\_Tree extends Frame {
5. public void paint (Graphics g){
6. Graphics2D g2d = (Graphics2D) g;
7. BasicStroke bs = new BasicStroke(10.0f);
8. g2d.setStroke(bs);
9. GeneralPath base = new GeneralPath();
10. base.moveTo(300, 550);
11. base.lineTo(315, 550);
12. base.lineTo(315, 580);
13. base.lineTo(285, 580);
14. base.lineTo(285, 550);
15. base.lineTo(300, 550);
16. g2d.setPaint(Color.GRAY);
17. g2d.fill(base);
18. g2d.draw(base);
19. GeneralPath tree = new GeneralPath();
20. tree.moveTo(300, 100);
21. tree.lineTo(250, 250);
22. tree.lineTo(270, 250);
23. tree.lineTo(200, 400);
24. tree.lineTo(220, 400);
25. tree.lineTo(150, 550);
26. tree.lineTo(300, 550);
27. tree.moveTo(300, 100);
28. tree.lineTo(350, 250);
29. tree.lineTo(330, 250);
30. tree.lineTo(400, 400);
31. tree.lineTo(380, 400);
32. tree.lineTo(450, 550);
33. tree.lineTo(300, 550);
34. g2d.setPaint(Color.GREEN);
35. g2d.fill(tree);
36. g2d.draw(tree);
38. GeneralPath star = new GeneralPath();
39. star.moveTo(300, 90);
40. star.lineTo(280, 70);
41. star.lineTo(300, 50);
42. star.lineTo(320, 70);
43. star.lineTo(300, 90);
44. g2d.setPaint(Color.ORANGE);
45. g2d.fill(star);
46. g2d.draw(star);
47. }
48. public static void main (String[] args){
49. Frame frame = new MP2\_Tree();
50. frame.setTitle("MP2: Tree");
51. frame.setSize(600, 600);
52. frame.setBackground(Color.WHITE);
53. frame.setForeground(Color.BLACK);
54. frame.setVisible(true);
55. }
56. }

**OUTPUT**

**A screenshot of a computer

Description automatically generated**

**Conversion using Areas in Java**

**Union (Using the code from above)**

**// Convert cross to Area**

**Area crossArea = new Area(cross);**

**// Perform the union operation**

**Area unionArea = new Area(treeArea);**

**unionArea.add(crossArea);**

**// Draw the union of the tree and cross**

**g2d.setPaint(Color.GRAY);**

**g2d.fill(unionArea);**

**g2d.setPaint(Color.BLACK);**

**g2d.draw(unionArea);**

**}**

**OUTPUT**

**A computer screen shot of a black and grey figure

Description automatically generated**

**Intersection (Using the code from above)**

**// Perform the intersection operation**

**Area intersectionArea = new Area(treeArea);**

**intersectionArea.intersect(crossArea);**

**// Draw the intersection of the tree and cross**

**g2d.setPaint(Color.RED);**

**g2d.fill(intersectionArea);**

**g2d.setPaint(Color.BLACK);**

**g2d.draw(intersectionArea);**

**OUTPUT**

**A red arrow pointing up

Description automatically generated**

**Symmetric Difference (Using the code from above)**

**// Perform the symmetric difference operation**

**Area symmetricDifferenceArea = new Area(treeArea);**

**symmetricDifferenceArea.add(crossArea);**

**Area intersectionArea = new Area(treeArea);**

**intersectionArea.intersect(crossArea);**

**symmetricDifferenceArea.subtract(intersectionArea);**

**// Draw the symmetric difference of the tree and cross**

**g2d.setPaint(Color.MAGENTA);**

**g2d.fill(symmetricDifferenceArea);**

**g2d.setPaint(Color.BLACK);**

**g2d.draw(symmetricDifferenceArea);**

**OUTPUT**

**A screenshot of a computer

Description automatically generated**

**Relative Difference (Using the code from above)**

**// Convert cross to Area**

**Area crossArea = new Area(cross);**

**// Perform the relative difference operation (Tree - Cross)**

**Area relativeDifferenceArea = new Area(treeArea);**

**relativeDifferenceArea.subtract(crossArea);**

**// Draw the relative difference of the tree and cross**

**g2d.setPaint(Color.CYAN);**

**g2d.fill(relativeDifferenceArea);**

**g2d.setPaint(Color.BLACK);**

**g2d.draw(relativeDifferenceArea);**

**OUTPUT**

**A computer screen shot of a logo

Description automatically generated**