

5110597: Computaional Frabication

Final Project: Lithophane

Member:

5931040421 Mr.Pongsathorn Kerdphol

Description:

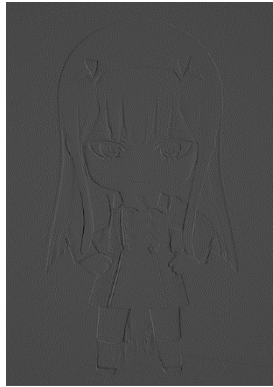
โปรเจคนี้จะเป็นการสร้าง 3D model จากรูปภาพ เมื่อปริ้นออกมาแล้วด้วย 3D Printer จะลักษณะเหมือนงานแกะสลัก เมื่อนำผลงานมาวางหน้าแหล่งกำเนิดแสง จะเกิดเป็นรูปภาพ



รูปตัวอย่าง



รูปขาวดำ



รูป 3D model (Top view)

Code:

```

main.py
23 v1 = []
24 v2 = []
25 # top
26 for i in range(height-1):
27     v1 = []
28     v2 = []
29     for j in range(width):
30         v1.append((j * output_width / width,
31                 i * output_height / height,
32                 (255 - img[i, j]) * layer / 255 + thickness))
33         v2.append((j * output_width / width,
34                 (i+1) * output_height / height,
35                 (255 - img[i+1, j]) * layer / 255 + thickness))
36     geometry.quad(v1, v2)
37
38 # side width
39 v1 = []
40 v2 = []
41 for i in range(width):
42     v1.append((i * output_width / width,
43             0 * output_height / height,
44             (255 - img[0, i]) * layer / 255 + thickness))
45     v2.append((i * output_width / width,
46             0 * output_height / height,
47             0))
48     geometry.quad(v1, v2)
49
50 v1 = []
51 v2 = []
52 for i in range(width):
53     v1.append((i * output_width / width,
54             (height-1) * output_height / height,
55             (255 - img[height-1, i]) * layer / 255 + thickness))
56     v2.append((i * output_width / width,
57             (height-1) * output_height / height,
58             0))
59     geometry.quad(v1, v2)
60
61 # side height
62 v1 = []
63 v2 = []
64 for i in range(height):
65     v1.append((0 * output_width / width,
66             i * output_height / height,
67             (255 - img[i, 0]) * layer / 255 + thickness))
68     v2.append((0 * output_width / width,
69             i * output_height / height,
70             0))
71     geometry.quad(v1, v2)
72
73 v1 = []
74 v2 = []
75 for i in range(height):
76     v1.append(((width-1) * output_width / width,
77             i * output_height / height,
78             (255 - img[i, width-1]) * layer / 255 + thickness))
79     v2.append(((width-1) * output_width / width,
80             i * output_height / height,
81             0))
82     geometry.quad(v1, v2)
83
84 # bottom
85 v1 = []
86 v2 = []
87 v1.append((0 * output_width / width, 0 * output_height / height, 0))
88 v2.append((0 * output_width / width, (height-1) * output_height / height, 0))
89 v1.append(((width-1) * output_width / width, 0 * output_height / height, 0))
90 v2.append(((width-1) * output_width / width, (height-1) * output_height / height, 0))
91     geometry.quad(v1, v2)
92
93 return geometry

```

```

geo.py
13 (b, c, a) if b < a and c < a and b <= c else (c, b, a)
14
15 )
16
17 )
18
19
20 class GEO:
21     """Geometric for GEO"""
22     def __init__(self):
23         self.points = {}
24         self.edges = set()
25         self.faces = set()
26
27     def add_points(self, point = (0, 0, 0)):
28         if point not in self.points.keys():
29             self.points[point] = len(self.points)
30             return self.points[point]
31
32     def quad(self, v1, v2):
33         if len(v1) != len(v2): return
34         if len(v1) == 1:
35             p1 = self._add_points(v1[0])
36             p2 = self._add_points(v2[0])
37             self.edges.add(sorted_tuple((p1, p2)))
38         else:
39             for i in range(1, len(v1)):
40                 p1 = self._add_points(v1[i-1])
41                 p2 = self._add_points(v1[i])
42                 p3 = self._add_points(v2[i-1])
43                 p4 = self._add_points(v2[i])
44                 self.edges.add(sorted_tuple((p1, p2)))
45                 self.edges.add(sorted_tuple((p2, p3)))
46                 self.edges.add(sorted_tuple((p3, p4)))
47                 self.edges.add(sorted_tuple((p4, p1)))
48                 self.faces.add(sorted_tuple_3((p1, p2, p3)))
49                 self.faces.add(sorted_tuple_3((p2, p3, p4)))
50                 self.faces.add(sorted_tuple_3((p3, p4, p1)))
51                 self.faces.add(sorted_tuple_3((p4, p1, p2)))
52
53     def save_stl(self, filename = 'geo.stl'):
54         # Create the mesh
55         faces = np.array(list(self.faces))
56         output = mesh.Mesh(np.zeros(faces.shape[0], dtype=mesh.Mesh.dtype))
57         points = []
58         for k in self.points.keys():
59             t = self.points[k]
60             points[t] = np.array(list(k))
61
62         for i, f in enumerate(faces):
63             for j in range(3):
64                 output.vectors[i][j] = points[f[j]]
65
66         # write the mesh to file "cube.stl"
67         output.save(filename)
68
69     def __str__(self):
70         return "points size: {}, edges size: {}, faces size: {}".format(len(self.points), len(self.edges), len(self.faces))
71
72
73

```

ในไฟล์ Main.py จะเกี่ยวกับการคำนวณจุด point cloud ของแต่ละ pixel เพื่อไปสร้าง mesh

ในไฟล์ Geo.py จะเกี่ยวกับการสร้าง mesh และ stl file