

Tartalomjegyzék

Mellékletek	1
M.1. Ender nyomtató URDF modellje	1
M.2. Ender nyomtató URDF modellje JSON formátumba	5
M.3. Craftbot nyomtató URDF modellje	7
M.4. Craftbot nyomtató URDF modellje JSON formátumba	11
M.5. Printer.ts	12
M.6. index.html	46
M.7. controls.html	49
M.8. tree.html	52
M.9. Fan.ts	55
M.10.TemperatureSimulator.ts	58
M.11.Telefon állvány alkatrészrajza	59

Mellékletek

M.1. Ender nyomtató URDF modellje

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <robot name="craftbot">
3   <!-- links: main serial chain -->
4   <link name="base">
5     <visual>
6       <origin rpy="0 0 0" xyz="0 0 0"/>
7       <geometry>
8         <mesh filename="base.mesh"/>
9       </geometry>
10    </visual>
11  </link>
12  <link name="x_axis">
13    <visual>
14      <origin rpy="0 0 0" xyz="0 0 0"/>
15      <geometry>
16        <mesh filename="x_axis_old.mesh"/>
17      </geometry>
18    </visual>
19  </link>
20  <link name="y_axis">
21    <visual>
22      <origin rpy="0 0 0" xyz="0 0 0"/>
23      <geometry>
24        <mesh filename="y_axis.mesh"/>
25      </geometry>
26    </visual>
27  </link>
28  <link name="z_axis">
29    <visual>
30      <origin rpy="0 0 0" xyz="0 0 0"/>
31      <geometry>
32        <mesh filename="z_axis.mesh"/>
33      </geometry>
34    </visual>
```

```

35 </link>
36 <link name="Spool">
37   <visual>
38     <origin rpy="0 0 0" xyz="0 0 0"/>
39     <geometry>
40       <mesh filename="Filament Spool.mesh"/>
41     </geometry>
42   </visual>
43 </link>
44 <link name="spoolDisp">
45   <visual>
46     <origin rpy="0 0 0" xyz="0 0 0"/>
47   </visual>
48 </link>
49 <link name="spoolControl">
50   <visual>
51     <origin rpy="0 0 0" xyz="0 0 0"/>
52   </visual>
53 </link>
54 <link name="zInControl">
55   <visual>
56     <origin rpy="0 0 0" xyz="0 0 0"/>
57   </visual>
58 </link>
59 <link name="zInDisp">
60   <visual>
61     <origin rpy="0 0 0" xyz="0 0 0"/>
62   </visual>
63 </link>
64 <link name="zOutDisp">
65   <visual>
66     <origin rpy="0 0 0" xyz="0 0 0"/>
67   </visual>
68 </link>
69 <link name="zOutControl">
70   <visual>
71     <origin rpy="0 0 0" xyz="0 0 0"/>
72   </visual>
73 </link>
74 <link name="xInControl">
75   <visual>
76     <origin rpy="0 0 0" xyz="0 0 0"/>
77   </visual>
78 </link>
79 <link name="xInDisp">
80   <visual>
81     <origin rpy="0 0 0" xyz="0 0 0"/>
82   </visual>
83 </link>

```

```

84 <link name="controlPanel">
85   <visual>
86     <origin rpy="0 0 0" xyz="0 0 0"/>
87   </visual>
88 </link>
89 <link name="table">
90   <visual>
91     <origin rpy="0 0 0" xyz="0 0 0"/>
92   </visual>
93 </link>
94 <link name="fan">
95   <visual>
96     <geometry>
97       <mesh filename="ext_fan.mesh"/>
98     </geometry>
99     <origin rpy="0 0 0" xyz="0 0 0"/>
100   </visual>
101 </link>
102 <!-- x:x y:-z z:y -->
103 <joint name="joint_1" type="prismatic">
104   <origin rpy="0 0 0" xyz="0 0 0"/>
105   <parent link="base"/>
106   <child link="z_axis"/>
107   <axis xyz="0 0 1"/>
108 </joint>
109 <joint name="joint_2" type="prismatic">
110   <origin rpy="0 0 0" xyz="0 0 0"/>
111   <parent link="base"/>
112   <child link="y_axis"/>
113   <axis xyz="0 -1 0"/>
114 </joint>
115 <joint name="joint_3" type="prismatic">
116   <origin rpy="-1.5707963 0 0" xyz="0 0 0"/>
117   <parent link="z_axis"/>
118   <child link="x_axis"/>
119   <axis xyz="1 0 0"/>
120 </joint>
121 <joint name="joint_4" type="fixed">
122   <origin rpy="0 0 0" xyz="0 0 0"/>
123   <parent link="base"/>
124   <child link="Spool"/>
125   <axis xyz="0 0 0"/>
126 </joint>
127 <joint name="joint_5" type="fixed">
128   <origin rpy="0 0 0" xyz="1 -8.2 50"/>
129   <parent link="base"/>
130   <child link="spoolDisp"/>
131   <axis xyz="0 0 0"/>
132 </joint>

```

```

133 <joint name="joint_6" type="fixed">
134   <origin rpy="0 0 0" xyz="1 -8.2 17"/>
135   <parent link="z_axis"/>
136   <child link="spoolControl"/>
137   <axis xyz="0 0 0"/>
138 </joint>
139 <joint name="joint_7" type="fixed">
140   <origin rpy="0 0 0" xyz="-7 -11 19.364"/>
141   <parent link="z_axis"/>
142   <child link="zInControl"/>
143   <axis xyz="0 0 0"/>
144 </joint>
145 <joint name="joint_8" type="fixed">
146   <origin rpy="0 0 0" xyz="8.0312 -10.47 19.362"/>
147   <parent link="z_axis"/>
148   <child link="zInDisp"/>
149   <axis xyz="0 0 0"/>
150 </joint>
151 <joint name="joint_9" type="fixed">
152   <origin rpy="0 0 0" xyz="8.0312 -10.47 19.362"/>
153   <parent link="z_axis"/>
154   <child link="zOutDisp"/>
155   <axis xyz="0 0 0"/>
156 </joint>
157 <joint name="joint_10" type="fixed">
158   <origin rpy="0 0 0" xyz="15.0312 -10.47 19.362"/>
159   <parent link="z_axis"/>
160   <child link="zOutControl"/>
161   <axis xyz="0 0 0"/>
162 </joint>
163 <joint name="joint_11" type="fixed">
164   <origin rpy="0 0 0" xyz="0.047873 -0.2563 -0.1987"/>
165   <parent link="x_axis"/>
166   <child link="xInControl"/>
167   <axis xyz="0 0 0"/>
168 </joint>
169 <joint name="joint_12" type="fixed">
170   <origin rpy="0 0 0" xyz="0.047873 -0.1543 -0.1987"/>
171   <parent link="x_axis"/>
172   <child link="xInDisp"/>
173   <axis xyz="0 0 0"/>
174 </joint>
175 <joint name="joint_13" type="fixed">
176   <origin rpy="-1.1485098 0 0" xyz="34.1785 -22.2322 7.78085"/>
177   <parent link="base"/>
178   <child link="controlPanel"/>
179   <axis xyz="0 0 0"/>
180 </joint>
181 <joint name="joint_14" type="fixed">

```

```

182   <origin rpy="0 0 0" xyz="0 0 0"/>
183   <parent link="y_axis"/>
184   <child link="table"/>
185   <axis xyz="0 0 0"/>
186 </joint>
187 <joint name="joint_15" type="revolute">
188   <origin rpy="0 0 0" xyz="0.0399744 -0.132313 -0.222813"/>
189   <parent link="x_axis"/>
190   <child link="fan"/>
191   <axis xyz="0 0 1"/>
192 </joint>
193 </robot>

```

M.1.1. kódrészlet. Ender nyomtató URDF modellje

M.2. Ender nyomtató URDF modellje JSON formátumba

```

1  {
2    "robot": {
3      "$": {
4        "name": "ender"
5      },
6      "link": [
7        {
8          "$": {
9            "name": "base"
10         },
11         "visual": [
12           {
13             "origin": [
14               {
15                 "$": {
16                   "rpy": "0 0 0",
17                   "xyz": "0 0 0"
18               }
19             ]
20           },
21           "geometry": [
22             {
23               "mesh": [
24                 {
25                   "$": {
26                     "filename": "base.mesh"
27                   }
28                 }
29               ]
30             }

```

```

31     ]
32   }
33 ]
34 },
35 ...
36 ],
37 "joint": [
38   {
39     "$": {
40       "name": "joint_1",
41       "type": "prismatic"
42     },
43     "origin": [
44       {
45         "$": {
46           "rpy": "0 0 0",
47           "xyz": "0 0 0"
48         }
49       }
50     ],
51     "parent": [
52       {
53         "$": {
54           "link": "base"
55         }
56       }
57     ],
58     "child": [
59       {
60         "$": {
61           "link": "z_axis"
62         }
63       }
64     ],
65     "axis": [
66       {
67         "$": {
68           "xyz": "0 0 1"
69         }
70       }
71     ]
72   },
73   ...
74 ]
75 }
76 }

```

M.2.1. kódrészlet. Ender URDF modellje json formátumban

M.3. Craftbot nyomtató URDF modellje

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <robot name="craftbot">
3   <!-- links: main serial chain -->
4   <link name="base">
5     <visual>
6       <origin rpy="0 0 0" xyz="0 0 0"/>
7       <geometry>
8         <mesh filename="Body.mesh"/>
9       </geometry>
10    </visual>
11  </link>
12  <link name="Head">
13    <visual>
14      <origin rpy="0 0 0" xyz="0 0 0"/>
15      <geometry>
16        <mesh filename="Head.mesh"/>
17      </geometry>
18    </visual>
19  </link>
20  <link name="Rail">
21    <visual>
22      <origin rpy="0 0 0" xyz="0 0 0"/>
23      <geometry>
24        <mesh filename="Rail.mesh"/>
25      </geometry>
26    </visual>
27  </link>
28  <link name="Work_plate">
29    <visual>
30      <origin rpy="0 0 0" xyz="0 0 0"/>
31      <geometry>
32        <mesh filename="Work_plate.mesh"/>
33      </geometry>
34    </visual>
35  </link>
36  <link name="Spool">
37    <visual>
38      <origin rpy="0 0 0" xyz="0 0 0"/>
39      <geometry>
40        <mesh filename="Filament Spool.mesh"/>
41      </geometry>
42    </visual>
43  </link>
44  <link name="spoolDisp">
45    <visual>
46      <origin rpy="0 0 0" xyz="0 0 0"/>
47    </visual>
```

```

48 </link>
49 <link name="spoolControl">
50   <visual>
51     <origin rpy="0 0 0" xyz="0 0 0"/>
52   </visual>
53 </link>
54 <link name="zInControl">
55   <visual>
56     <origin rpy="0 0 0" xyz="0 0 0"/>
57   </visual>
58 </link>
59 <link name="zInDisp">
60   <visual>
61     <origin rpy="0 0 0" xyz="0 0 0"/>
62   </visual>
63 </link>
64 <link name="zOutDisp">
65   <visual>
66     <origin rpy="0 0 0" xyz="0 0 0"/>
67   </visual>
68 </link>
69 <link name="zOutControl">
70   <visual>
71     <origin rpy="0 0 0" xyz="0 0 0"/>
72   </visual>
73 </link>
74 <link name="xInControl">
75   <visual>
76     <origin rpy="0 0 0" xyz="0 0 0"/>
77   </visual>
78 </link>
79 <link name="xInDisp">
80   <visual>
81     <origin rpy="0 0 0" xyz="0 0 0"/>
82   </visual>
83 </link>
84 <link name="controlPanel">
85   <visual>
86     <origin rpy="0 0 0" xyz="0 0 0"/>
87   </visual>
88 </link>
89 <link name="table">
90   <visual>
91     <origin rpy="0 0 0" xyz="0 0 0"/>
92   </visual>
93 </link>
94 <link name="fan">
95   <visual>
96     <geometry>

```

```

97     <mesh filename="ext_fan.mesh"/>
98   </geometry>
99   <origin rpy="0 0 0" xyz="0 0 0"/>
100 </visual>
101 </link>
102 <!-- x:x y:-z z:y -->
103 <joint name="joint_1" type="prismatic">
104   <origin rpy="0 0 0" xyz="0 0 23.72"/>
105   <parent link="base"/>
106   <child link="Work_plate"/>
107   <axis xyz="0 0 -1"/>
108 </joint>
109 <joint name="joint_2" type="prismatic">
110   <origin rpy="0 0 0" xyz="0 -7.37 0"/>
111   <parent link="base"/>
112   <child link="Rail"/>
113   <axis xyz="0 1 0"/>
114 </joint>
115 <joint name="joint_3" type="prismatic">
116   <origin rpy="0 0 0" xyz="-16.4 0 0"/>
117   <parent link="Rail"/>
118   <child link="Head"/>
119   <axis xyz="1 0 0"/>
120 </joint>
121 <joint name="joint_4" type="fixed">
122   <origin rpy="0 0 1.5707963" xyz="-40 -15 -5"/>
123   <parent link="base"/>
124   <child link="Spool"/>
125 </joint>
126 <joint name="joint_5" type="fixed">
127   <origin rpy="0 0 0" xyz="-30.8312 4.2 44.6"/>
128   <parent link="Body"/>
129   <child link="spoolDisp"/>
130   <axis xyz="0 0 0"/>
131 </joint>
132 <joint name="joint_6" type="fixed">
133   <origin rpy="0 0 0" xyz="-26.8312 7.77 50.32"/>
134   <parent link="Body"/>
135   <child link="spoolControl"/>
136   <axis xyz="0 0 0"/>
137 </joint>
138 <joint name="joint_7" type="fixed">
139   <origin rpy="0 0 0" xyz="-26.8312 3.77 54.32"/>
140   <parent link="Body"/>
141   <child link="zInControl"/>
142   <axis xyz="0 0 0"/>
143 </joint>
144 <joint name="joint_8" type="fixed">
145   <origin rpy="0 0 0" xyz="-26.8312 4.77 54.32"/>

```

```

146     <parent link="Body"/>
147     <child link="zInDisp"/>
148     <axis xyz="0 0 0"/>
149 </joint>
150 <joint name="joint_9" type="fixed">
151     <origin rpy="0 0 0" xyz="-26.8312 4.77 54.32"/>
152     <parent link="Body"/>
153     <child link="zOutDisp"/>
154     <axis xyz="0 0 0"/>
155 </joint>
156 <joint name="joint_10" type="fixed">
157     <origin rpy="0 0 0" xyz="-26.8312 4.77 84.32"/>
158     <parent link="Body"/>
159     <child link="zOutControl"/>
160     <axis xyz="0 0 0"/>
161 </joint>
162 <joint name="joint_11" type="fixed">
163     <origin rpy="0 0 0" xyz="1.0312 -5.47 67.63"/>
164     <parent link="Head"/>
165     <child link="xInControl"/>
166     <axis xyz="0 0 0"/>
167 </joint>
168 <joint name="joint_12" type="fixed">
169     <origin rpy="0 0 0" xyz="1.0312 -5.47 50.362"/>
170     <parent link="Head"/>
171     <child link="xInDisp"/>
172     <axis xyz="0 0 0"/>
173 </joint>
174 <joint name="joint_13" type="fixed">
175     <origin rpy="0 0 0" xyz="0.1 -19.5 46.7"/>
176     <parent link="Body"/>
177     <child link="controlPanel"/>
178     <axis xyz="0 0 0"/>
179 </joint>
180 <joint name="joint_14" type="fixed">
181     <origin rpy="0 0 0" xyz="0 0 0"/>
182     <parent link="Work_plate"/>
183     <child link="table"/>
184     <axis xyz="0 0 0"/>
185 </joint>
186 <joint name="joint_15" type="revolute">
187     <origin rpy="4.71 0 0" xyz="1.1 -8.6 42.19"/>
188     <parent link="Head"/>
189     <child link="fan"/>
190     <axis xyz="0 1 0"/>
191 </joint>
192 </robot>

```

M.3.1. kódrészlet. Craftbot nyomtató URDF modellje

M.4. Craftbot nyomtató URDF modellje JSON formátumba

```
1 {
2   "robot": {
3     "$": {
4       "name": "craftbot"
5     },
6     "link": [
7       {
8         "$": {
9           "name": "base"
10        },
11        "visual": [
12          {
13            "origin": [
14              {
15                "$": {
16                  "rpy": "0 0 0",
17                  "xyz": "0 0 0"
18                }
19              }
20            ],
21            "geometry": [
22              {
23                "mesh": [
24                  {
25                    "$": {
26                      "filename": "Body.mesh"
27                    }
28                  }
29                ]
30              }
31            ]
32          }
33        ],
34      },
35      ...
36    ],
37    "joint": [
38      {
39        "$": {
40          "name": "joint_1",
41          "type": "prismatic"
42        },
43        "origin": [
44          {
45            "$": {
46              "rpy": "0 0 0",
47              "xyz": "0 0 23.72"
```

```

48     }
49     }
50     ],
51     "parent": [
52     {
53         "$": {
54             "link": "base"
55         }
56     }
57     ],
58     "child": [
59     {
60         "$": {
61             "link": "Work_plate"
62         }
63     }
64     ],
65     "axis": [
66     {
67         "$": {
68             "xyz": "0 0 -1"
69         }
70     }
71     ]
72     },
73     ...
74 ]
75 }
76 }

```

M.4.1. kódrészlet. Craftbot URDF modellje json formátumban

M.5. Printer.ts

```

1  // import statements
2  import { Billboard, Mesh, Node, Vector3, wom } from "maxwhere";
3  import * as THREE from "../utils/three.module";
4  import path from "path";
5  import { GcodeReader, ParsedResult } from "./GcodeReader";
6  import { ipcMain, dialog, ipcRenderer } from "electron";
7  import { makeCurveMesh } from "./CurveMesh";
8  import { printerOptions } from "../index";
9  import { resolve } from "path";
10
11 import womAsync from '../utils/womAsync.module';
12 import { beginLoadTextViaDialog } from './OpenFileDialog';
13 import { readFileSync, readSync } from "fs";

```

```

14 import { Fan } from "./Fan";
15 import { TemperatureSimulator } from "./TemperatureSimulator";
16 import * as fs from 'fs';
17 import * as xml2js from 'xml2js';
18 const euler2quaternion = require('euler-to-quaternion');
19 import { threadId } from "worker_threads";
20 import { Quaternion } from "src/utils/three.module";
21
22 const TAG = '[3DPW/Printer]';
23
24 const FILAMENT_TUBULAR_SEGMENTS = 50;
25 const FILAMENT_RADIUS_SCALE = 0.04;
26 const FILAMENT_RADIAL_SEGMENTS = 3;
27 const FILAMENT_MATERIAL_NAME = 'Physics/DebugDraw/VertexDrivenDepth';
28
29 const SLEEVE_TUBULAR_SEGMENTS = 30;
30 const SLEEVE_RADIUS_SCALE = 0.18;
31 const SLEEVE_RADIAL_SEGMENTS = 3;
32 const SLEEVE_MATERIAL_NAME = 'Physics/DebugDraw/VertexDrivenDepth';
33 const SLEEVE_COLOR = { r: 1.0, g: 1.0, b: 0.0 };
34
35 const WAIT_BETWEEN_COMMANDS_MS = 22;
36
37 function V3ToThree(pos: Vector3) {
38     return new THREE.Vector3(pos.x, pos.y, pos.z);
39 }
40
41 function V3ToMw(v: THREE.Vector3): Vector3 {
42     return {
43         x: v.x,
44         y: v.y,
45         z: v.z,
46     };
47 }
48
49 function eulerToQuat(euler: THREE.Vector3):Quaternion{
50     let eulerArray=[euler.x, euler.y, euler.z];
51     let quat=euler2quaternion(eulerArray);
52     return {
53         w: quat[0],
54         x: quat[1],
55         y: quat[2],
56         z: quat[3],
57     };
58 }
59
60 export interface IPrinter {
61     isPrinting: boolean;
62 }

```

```

63
64 /**
65  * Represents a 3DPrinter
66  */
67 export class Printer implements IPrinter {
68     // Declare fields
69     FILAMENT_COLOR = { r: 0.4, g: 0.4, b: 0.0 };
70     red :number=0.4;
71     green:number=0.4;
72     blue :number=0;
73     cubeWidth = 0.08;
74     filRes = 1000;
75     basePos: THREE.Vector3;
76     x_start_pos = new THREE.Vector3(0,0,0);
77     y_start_pos = new THREE.Vector3(0,0,0);
78     z_start_pos = new THREE.Vector3(0,0,0);
79     z_direction: number;
80     y_direction: number;
81     x_direction: number;
82     controlPanel_pos = new THREE.Vector3(0,0,0);
83     controlPanel_rpy = new THREE.Vector3(0,0,0);
84     spool_pos = new THREE.Vector3(0,0,0);
85     spoolDisp_pos = new THREE.Vector3(0,0,0);
86     spoolControl_pos = new THREE.Vector3(0,0,0);
87     zInControl_pos = new THREE.Vector3(0,0,0);
88     xInControl_pos = new THREE.Vector3(0,0,0);
89     zInDisp_pos = new THREE.Vector3(0,0,0);
90     xInDisp_pos = new THREE.Vector3(0,0,0);
91     zOutControl_pos = new THREE.Vector3(0,0,0);
92     zOutDisp_pos = new THREE.Vector3(0,0,0);
93     fan_pos = new THREE.Vector3(0,0,0);
94     x_rpy = new THREE.Vector3(0,0,0);
95     y_rpy = new THREE.Vector3(0,0,0);
96     z_rpy = new THREE.Vector3(0,0,0);
97     spool_rpy = new THREE.Vector3(0,0,0);
98     base_rpy = new THREE.Vector3(0,0,0);
99     fan_rpy = new THREE.Vector3(0,0,0);
100     fan_axis = new THREE.Vector3(0,0,0);
101     base_start_pos = new THREE.Vector3(0,0,0);
102     axis_x=[];
103     axis_y=[];
104     axis_z=[];
105     x_axisPos = new THREE.Vector3();
106     y_axisPos = new THREE.Vector3();
107     z_axisPos = new THREE.Vector3();
108     printerID: String;
109     basename: String;
110     tree0: String;
111     tree1: String;

```

```

112     tree2: String;
113     tree3: String;
114
115
116     // buildArray: Mesh[] = [];
117
118     headOffset: THREE.Vector3;
119
120     fanController?: Fan;
121     headTemperatureSensor?: TemperatureSimulator;
122
123     // filament
124     spoolControl_node: Mesh;
125     zInControl_node: Mesh;
126     zInDisp_node: Mesh;
127     spoolDisp_node: Mesh;
128     zOutDisp_node: Mesh;
129     zOutControl_node: Mesh;
130     xInControl_node: Mesh;
131     xInDisp_node: Mesh;
132     table_node: Node;
133     table_id="";
134     spoolDisp = new THREE.Vector3(0,0,0);
135     spoolControl = new THREE.Vector3(0,0,0);
136     zInControl = new THREE.Vector3(0,0,0);
137     zInDisp = new THREE.Vector3(0,0,0);
138     filHandleArr = [new THREE.Vector3(0,0,0), new THREE.Vector3(0,0,0), new THREE.Vector3(0,0,0), new THREE.
        Vector3(0,0,0)];
139     filamentBezier = new THREE.CubicBezierCurve3(new THREE.Vector3(0,0,0), new THREE.Vector3(0,0,0), new
        THREE.Vector3(0,0,0), new THREE.Vector3(0,0,0));// THREE.CubicBezierCurve3;
140     filament = makeCurveMesh(wom, {
141         materialName: FILAMENT_MATERIAL_NAME,
142     });
143     spoolDisp_created: boolean;
144     spoolControl_created: boolean;
145     zInControl_created: boolean;
146     zInDisp_created: boolean;
147
148     // sleeve
149     zOutDisp = new THREE.Vector3(0,0,0);
150     zOutControl = new THREE.Vector3(0,0,0);
151     xInControl = new THREE.Vector3(0,0,0);
152     xInDisp = new THREE.Vector3(0,0,0);
153     sleeveHandleArr = [new THREE.Vector3(0,0,0), new THREE.Vector3(0,0,0), new THREE.Vector3(0,0,0), new
        THREE.Vector3(0,0,0)];//: [THREE.Vector3, THREE.Vector3, THREE.Vector3, THREE.Vector3];
154     sleeveBezier = new THREE.CubicBezierCurve3(new THREE.Vector3(0,0,0), new THREE.Vector3(0,0,0), new THREE.
        Vector3(0,0,0), new THREE.Vector3(0,0,0));// THREE.CubicBezierCurve3; : THREE.CubicBezierCurve3;
155     sleeve = makeCurveMesh(wom, {
156         materialName: SLEEVE_MATERIAL_NAME

```

```

157     });
158
159     timeScale = 1.0;
160     dur = 0;
161     extrude = -1;
162     prevExtrude = 0;
163     isPrinting = false;
164     isExtruding = false;
165     command = {
166         coords: new THREE.Vector3(0, 0, 0),
167         feedRate: 2000,
168         extrusion: 0,
169     };
170     newPrint = true;
171
172     eps = 0.001;
173     buildEps = 0.0025;
174     buildCount = 0;
175     maxBuild = 20;
176     buildTubeSegments = 10;
177     xMoved = 0;
178     yMoved = 0;
179     zMoved = 0;
180     xMovedBuild = 0;
181     zMovedBuild = 0;
182     yMovedBuild = 0;
183
184     tubePointsArray: THREE.Vector3[] = [];
185     buildTube = makeCurveMesh(wom, {
186         materialName: FILAMENT_MATERIAL_NAME,
187     });
188     lastPrintInExtrude = false;
189
190     pendingStop: boolean;
191     player: NodeJS.Timeout | null;
192     rotate!: NodeJS.Timeout;
193
194     isCooling=false;
195
196     url_x: any;
197     url_y: any;
198     url_z: any;
199     url_fan?:any;
200     url_base: any;
201     url_spool: any;
202     base: Mesh;
203     x_axis: Mesh;
204     y_axis: Mesh;
205     z_axis: Mesh;

```

```

206   ext_fan ? : Mesh;
207   spool: Mesh;
208   printBaseCube: Node;
209   printBaseTube: Node;
210   printCoordTracker: Node;
211   cubeTracker: Node;
212   controlPanel: Billboard;
213
214   readCode: GcodeReader;
215   baseScale: THREE.Vector3;
216   componentID: String;
217   printerSource: string;
218   controlPanelWidth: number;
219   controlPanelHeight: number;
220   resolutionWidth: number;
221   resolutionHeight: number;
222   x_axis_scale: Vector3;
223   y_axis_scale: Vector3;
224   z_axis_scale: Vector3;
225
226   constructor(options: printerOptions) {
227       this.player = null;
228       this.pendingStop = false;
229       this.basename="";
230       this.tree0="";
231       this.tree1="";
232       this.tree2="";
233       this.tree3="";
234
235       // Adatok lekérése a configból
236       this.basePos = Object.assign(new THREE.Vector3(), options.position);
237       console.log("componentPos: "+options.position.x);
238       this.baseScale = Object.assign(new THREE.Vector3(), options.scale);
239       this.headOffset = Object.assign(new THREE.Vector3(), options.headOffset);
240       this.componentID = options.componentID;
241       this.printerID = options.componentID;
242       this.printerSource=options.source;
243       this.controlPanelWidth=options.controlPanelWidth;
244       this.controlPanelHeight=options.controlPanelHeight;
245       this.resolutionHeight=options.resolutionHeight;
246       this.resolutionWidth=options.resolutionWidth;
247
248       // Filament pontjainak állapota
249       this.spoolDisp_created=false;
250       this.spoolControl_created=false;
251       this.zInControl_created=false;
252       this.zInDisp_created=false;
253
254       // 3 tengely skálázása

```

```

255     this.x_axis_scale=new THREE.Vector3(1,1,1);
256     this.y_axis_scale=new THREE.Vector3(1,1,1);
257     this.z_axis_scale=new THREE.Vector3(1,1,1);
258     if(options.x_axis_scale){
259         this.x_axis_scale=options.x_axis_scale;
260     }
261     if(options.y_axis_scale){
262         this.y_axis_scale=options.y_axis_scale;
263     }
264     if(options.z_axis_scale){
265         this.z_axis_scale=options.z_axis_scale;
266     }
267
268
269     // xml parser
270     var parser = new xml2js.Parser();
271
272     // urdf modell beolvasása
273     const data=fs.readFileSync(resolve(__dirname,"../urdf_models", this.printerSource), 'utf8')
274     var datas= data;
275     console.log("beolvasta");
276
277     // alapértelmezett tengelyirányok
278     this.z_direction=1;
279     this.y_direction=1;
280     this.x_direction=1;
281
282     // xml to json
283     var printer_urdf="";
284     parser.parseString(data, function (err: any, result: String) {
285         console.log('Done');
286         printer_urdf = JSON.stringify(result);
287         var printer3d=data;
288         return data;
289     });
290     var printer3d=JSON.parse(printer_urdf);
291     console.log("printer_urdf");
292     console.log(printer_urdf);
293     console.log(printer3d);
294
295
296     // urdf model json objektumból történő feldolgozása
297     for(let i=0; i < printer3d.robot.link.length; i++){
298         console.log(i+": "+printer3d.robot.link[i].$.name)
299     }
300     var parents=[];
301     var children=[];
302     for(let i=0; i < printer3d.robot.joint.length; i++){

```

```

303     console.log(i+": "+printer3d.robot.joint[i].parent[0].$.link+" - "+printer3d.robot.joint[i].child[0].
$.link);
304     parents.push(printer3d.robot.joint[i].parent[0].$.link);
305     children.push(printer3d.robot.joint[i].child[0].$.link);
306 }
307 var highest='';
308 var parentCounter=0;
309 for(let i=0; i<parents.length;i++){
310     if(children.includes(parents[i])==false){
311         highest=parents[i];
312     }
313 }
314
315 for(let i=0; i<printer3d.robot.link.length;i++){
316     if(printer3d.robot.link[i].$.name == "base") {
317         this.basename=printer3d.robot.link[i].$.name;
318         console.log(printer3d.robot.link[i].$.name+" is the base");
319         for(let j=0; j<printer3d.robot.link.length;j++){
320             if(printer3d.robot.link[j].$.name == printer3d.robot.link[i].$.name){
321                 this.url_base = printer3d.robot.link[j].visual[0].geometry[0].mesh[0].$.filename;
322                 console.log("url_base: "+this.url_base);
323                 let origin=printer3d.robot.link[j].visual[0].origin[0].$.xyz;
324                 let xyzArray = origin.split(" ");
325                 console.log(xyzArray);
326                 let origin_x=parseFloat(xyzArray[0]);
327                 let origin_y=parseFloat(xyzArray[1]);
328                 let origin_z=parseFloat(xyzArray[2]);
329                 this.base_start_pos=new THREE.Vector3(origin_x, origin_z, -origin_y);
330                 console.log("base_start: "+this.base_start_pos.x+","+this.base_start_pos.y+","+this.
base_start_pos.z);
331
332                 let rpy=printer3d.robot.joint[i].origin[0].$.rpy;
333                 let rpyArray = rpy.split(" ");
334                 console.log(rpyArray);
335                 let rpy_x=parseFloat(rpyArray[0]);
336                 let rpy_y=parseFloat(rpyArray[1]);
337                 let rpy_z=parseFloat(rpyArray[2]);
338                 this.base_rpy=new THREE.Vector3(rpy_x, rpy_z, -rpy_y);
339                 console.log("base_rpy: "+this.base_rpy.x+","+this.base_rpy.y+","+this.base_rpy.z)
340             }
341         }
342         var printer3d_base=printer3d.robot.link[0].visual[0].geometry[0].mesh[0].$.filename;
343     }
344 }
345 console.log(highest);
346
347 for(let i=0; i<printer3d.robot.joint.length;i++){
348     if(printer3d.robot.joint[i].child[0].$.link == "controlPanel"){
349         for(let j=0; j<printer3d.robot.link.length;j++){

```

```

350     if(printer3d.robot.link[j].$.name == "controlPanel"){
351         let origin=printer3d.robot.joint[i].origin[0].$.xyz;
352         let xyzArray = origin.split(" ");
353         console.log(xyzArray);
354         let origin_x=parseFloat(xyzArray[0]);
355         let origin_y=parseFloat(xyzArray[1]);
356         let origin_z=parseFloat(xyzArray[2]);
357         this.controlPanel_pos=new THREE.Vector3(origin_x, origin_z, -origin_y);
358         console.log("controlPanel_pos: "+this.controlPanel_pos.x+" "+this.controlPanel_pos.y+" "+this.
controlPanel_pos.z)
359
360         let rpy=printer3d.robot.joint[i].origin[0].$.rpy;
361         let rpyArray = rpy.split(" ");
362         console.log(rpyArray);
363         let rpy_x=parseFloat(rpyArray[0]);
364         let rpy_y=parseFloat(rpyArray[1]);
365         let rpy_z=parseFloat(rpyArray[2]);
366         this.controlPanel_rpy=new THREE.Vector3(rpy_x, rpy_z, -rpy_y);
367         console.log("controlPanel_rpy: "+this.controlPanel_rpy.x+" "+this.controlPanel_rpy.y+" "+this.
controlPanel_rpy.z)
368     }
369 }
370 }
371 if(printer3d.robot.joint[i].child[0].$.link == "Spool"){
372     for(let j=0; j<printer3d.robot.link.length;j++ ){
373         if(printer3d.robot.link[j].$.name == "Spool"){
374             this.url_spool = printer3d.robot.link[j].visual[0].geometry[0].mesh[0].$.filename;
375             console.log("url_spool: "+this.url_spool);
376             let origin=printer3d.robot.joint[i].origin[0].$.xyz;
377             let xyzArray = origin.split(" ");
378             console.log(xyzArray);
379             let origin_x=parseFloat(xyzArray[0]);
380             let origin_y=parseFloat(xyzArray[1]);
381             let origin_z=parseFloat(xyzArray[2]);
382             this.spool_pos=new THREE.Vector3(origin_x, origin_z, -origin_y);
383             console.log("spool_pos: "+this.spool_pos.x+" "+this.spool_pos.y+" "+this.spool_pos.z)
384
385             let rpy=printer3d.robot.joint[i].origin[0].$.rpy;
386             let rpyArray = rpy.split(" ");
387             console.log(rpyArray);
388             let rpy_x=parseFloat(rpyArray[0]);
389             let rpy_y=parseFloat(rpyArray[1]);
390             let rpy_z=parseFloat(rpyArray[2]);
391             this.spool_rpy=new THREE.Vector3(rpy_x, rpy_z, -rpy_y);
392             console.log("spool_rpy: "+this.spool_rpy.x+" "+this.spool_rpy.y+" "+this.spool_rpy.z)
393         }
394     }
395 }
396 if(printer3d.robot.joint[i].child[0].$.link == "spoolDisp"){

```

```

397     for(let j=0; j<printer3d.robot.link.length;j++ ){
398         if(printer3d.robot.link[j].$.name == "spoolDisp"){
399             let origin=printer3d.robot.joint[i].origin[0].$.xyz;
400             let xyzArray = origin.split(" ");
401             console.log(xyzArray);
402             let origin_x=parseFloat(xyzArray[0]);
403             let origin_y=parseFloat(xyzArray[1]);
404             let origin_z=parseFloat(xyzArray[2]);
405             this.spoolDisp_pos=new THREE.Vector3(origin_x, origin_z, -origin_y);
406             console.log("spoolDisp_pos: "+this.spoolDisp_pos.x+";"+this.spoolDisp_pos.y+";"+this.
spoolDisp_pos.z)
407         }
408     }
409 }
410 if(printer3d.robot.joint[i].child[0].$.link == "spoolControl"){
411     for(let j=0; j<printer3d.robot.link.length;j++ ){
412         if(printer3d.robot.link[j].$.name == "spoolControl"){
413             let origin=printer3d.robot.joint[i].origin[0].$.xyz;
414             let xyzArray = origin.split(" ");
415             console.log(xyzArray);
416             let origin_x=parseFloat(xyzArray[0]);
417             let origin_y=parseFloat(xyzArray[1]);
418             let origin_z=parseFloat(xyzArray[2]);
419             this.spoolControl_pos=new THREE.Vector3(origin_x, origin_z, -origin_y);
420             console.log("spoolControl_pos: "+this.spoolControl_pos.x+";"+this.spoolControl_pos.y+";"+this.
spoolControl_pos.z)
421         }
422     }
423 }
424 if(printer3d.robot.joint[i].child[0].$.link == "zInDisp"){
425     for(let j=0; j<printer3d.robot.link.length;j++ ){
426         if(printer3d.robot.link[j].$.name == "zInDisp"){
427             let origin=printer3d.robot.joint[i].origin[0].$.xyz;
428             let xyzArray = origin.split(" ");
429             console.log(xyzArray);
430             let origin_x=parseFloat(xyzArray[0]);
431             let origin_y=parseFloat(xyzArray[1]);
432             let origin_z=parseFloat(xyzArray[2]);
433             this.zInDisp_pos=new THREE.Vector3(origin_x, origin_z, -origin_y);
434             console.log("zInDisp_pos: "+this.zInDisp_pos.x+";"+this.zInDisp_pos.y+";"+this.zInDisp_pos.z)
435         }
436     }
437 }
438 if(printer3d.robot.joint[i].child[0].$.link == "zInControl"){
439     for(let j=0; j<printer3d.robot.link.length;j++ ){
440         if(printer3d.robot.link[j].$.name == "zInControl"){
441             let origin=printer3d.robot.joint[i].origin[0].$.xyz;
442             let xyzArray = origin.split(" ");
443             console.log(xyzArray);

```

```

444         let origin_x=parseFloat(xyzArray[0]);
445         let origin_y=parseFloat(xyzArray[1]);
446         let origin_z=parseFloat(xyzArray[2]);
447         this.zInControl_pos=new THREE.Vector3(origin_x, origin_z, -origin_y);
448         console.log("zInControl_pos: "+this.zInControl_pos.x+","+this.zInControl_pos.y+","+this.
zInControl_pos.z)
449     }
450 }
451 }
452 if(printer3d.robot.joint[i].child[0].$.link == "zOutDisp"){
453     for(let j=0; j<printer3d.robot.link.length;j++ ){
454         if(printer3d.robot.link[j].$.name == "zOutDisp"){
455             let origin=printer3d.robot.joint[i].origin[0].$.xyz;
456             let xyzArray = origin.split(" ");
457             console.log(xyzArray);
458             let origin_x=parseFloat(xyzArray[0]);
459             let origin_y=parseFloat(xyzArray[1]);
460             let origin_z=parseFloat(xyzArray[2]);
461             this.zOutDisp_pos=new THREE.Vector3(origin_x, origin_z, -origin_y);
462             console.log("zOutDisp: "+this.zOutDisp_pos.x+","+this.zOutDisp_pos.y+","+this.zOutDisp_pos.z)
463         }
464     }
465 }
466 if(printer3d.robot.joint[i].child[0].$.link == "zOutControl"){
467     for(let j=0; j<printer3d.robot.link.length;j++ ){
468         if(printer3d.robot.link[j].$.name == "zOutControl"){
469             let origin=printer3d.robot.joint[i].origin[0].$.xyz;
470             let xyzArray = origin.split(" ");
471             console.log(xyzArray);
472             let origin_x=parseFloat(xyzArray[0]);
473             let origin_y=parseFloat(xyzArray[1]);
474             let origin_z=parseFloat(xyzArray[2]);
475             this.zOutControl_pos=new THREE.Vector3(origin_x, origin_z, -origin_y);
476             console.log("zOutControl_pos: "+this.zOutControl_pos.x+","+this.zOutControl_pos.y+","+this.
zOutControl_pos.z)
477         }
478     }
479 }
480 if(printer3d.robot.joint[i].child[0].$.link == "xInDisp"){
481     for(let j=0; j<printer3d.robot.link.length;j++ ){
482         if(printer3d.robot.link[j].$.name == "xInDisp"){
483             let origin=printer3d.robot.joint[i].origin[0].$.xyz;
484             let xyzArray = origin.split(" ");
485             console.log(xyzArray);
486             let origin_x=parseFloat(xyzArray[0]);
487             let origin_y=parseFloat(xyzArray[1]);
488             let origin_z=parseFloat(xyzArray[2]);
489             this.xInDisp_pos=new THREE.Vector3(origin_x, origin_z, -origin_y);
490             console.log("xInDisp_pos: "+this.xInDisp_pos.x+","+this.xInDisp_pos.y+","+this.xInDisp_pos.z)

```

```

491     }
492   }
493 }
494 if(printer3d.robot.joint[i].child[0].$.link == "xInControl"){
495   for(let j=0; j<printer3d.robot.link.length;j++ ){
496     if(printer3d.robot.link[j].$.name == "xInControl"){
497       let origin=printer3d.robot.joint[i].origin[0].$.xyz;
498       let xyzArray = origin.split(" ");
499       console.log(xyzArray);
500       let origin_x=parseFloat(xyzArray[0]);
501       let origin_y=parseFloat(xyzArray[1]);
502       let origin_z=parseFloat(xyzArray[2]);
503       this.xInControl_pos=new THREE.Vector3(origin_x, origin_z, -origin_y);
504       console.log("xInControl_pos: "+this.xInControl_pos.x+" "+this.xInControl_pos.y+" "+this.
xInControl_pos.z)
505     }
506   }
507 }
508 if(printer3d.robot.joint[i].child[0].$.link == "fan"){
509   this.isCooling=true;
510   for(let j=0; j<printer3d.robot.link.length;j++ ){
511     if(printer3d.robot.link[j].$.name == "fan"){
512       this.url_fan = printer3d.robot.link[j].visual[0].geometry[0].mesh[0].$.filename;
513       let origin=printer3d.robot.joint[i].origin[0].$.xyz;
514       let xyzArray = origin.split(" ");
515       console.log(xyzArray);
516       let origin_x=parseFloat(xyzArray[0]);
517       let origin_y=parseFloat(xyzArray[1]);
518       let origin_z=parseFloat(xyzArray[2]);
519       this.fan_pos=new THREE.Vector3(origin_x, origin_z, -origin_y);
520       console.log("xInControl_pos: "+this.fan_pos.x+" "+this.fan_pos.y+" "+this.fan_pos.z)
521
522       let rpy=printer3d.robot.joint[i].origin[0].$.rpy;
523       let rpyArray = rpy.split(" ");
524       console.log(rpyArray);
525       let rpy_x=parseFloat(rpyArray[0]);
526       let rpy_y=parseFloat(rpyArray[1]);
527       let rpy_z=parseFloat(rpyArray[2]);
528       this.fan_rpy=new THREE.Vector3(rpy_x, rpy_z, -rpy_y);
529       console.log("y_rpy: "+this.fan_rpy.x+" "+this.fan_rpy.y+" "+this.fan_rpy.z)
530
531       let axis_fan=printer3d.robot.joint[i].axis[0].$.xyz;
532       let axisArray = axis_fan.split(" ");
533       console.log(axisArray);
534       let fan_x=parseFloat(axisArray[0]);
535       let fan_y=parseFloat(axisArray[1]);
536       let fan_z=parseFloat(axisArray[2]);
537       this.fan_axis=new THREE.Vector3(fan_x, fan_z, -fan_y);
538       console.log("axis_xyz: "+this.fan_axis.x+" "+this.fan_axis.y+" "+this.fan_axis.z)

```

```

539     }
540   }
541 }
542 if(printer3d.robot.joint[i].child[0].$.link != "Spool" && printer3d.robot.joint[i].child[0].$.link !=
"fan"){
543   console.log(printer3d.robot.joint[i].axis[0].$.xyz);
544   let xyz=printer3d.robot.joint[i].axis[0].$.xyz;
545   let myArray = xyz.split(" ");
546   console.log(myArray);
547   let axis_x=parseInt(myArray[0]);
548   let axis_y=parseInt(myArray[1]);
549   let axis_z=parseInt(myArray[2]);
550
551
552   if(axis_x != 0){
553     this.x_direction=axis_x;
554     var printer3d_x=printer3d.robot.joint[i].child[0].$.link;
555     console.log(printer3d.robot.joint[i].child[0].$.link+" is the x_axis");
556     this.axis_x=myArray;
557     for(let j=0; j<printer3d.robot.link.length;j++ ){
558       if(printer3d.robot.link[j].$.name == printer3d_x){
559         this.url_x = printer3d.robot.link[j].visual[0].geometry[0].mesh[0].$.filename;
560         console.log("url_x: "+this.url_x);
561         let origin=printer3d.robot.joint[i].origin[0].$.xyz;
562         let xyzArray = origin.split(" ");
563         console.log(xyzArray);
564         let origin_x=parseFloat(xyzArray[0]);
565         let origin_y=parseFloat(xyzArray[1]);
566         let origin_z=parseFloat(xyzArray[2]);
567         this.x_start_pos=new THREE.Vector3(origin_x, origin_z, -origin_y);
568         console.log("x_startV3: "+this.x_start_pos.x+","+this.x_start_pos.y+","+this.x_start_pos.z);
569
570         let rpy=printer3d.robot.joint[i].origin[0].$.rpy;
571         let rpyArray = rpy.split(" ");
572         console.log(rpyArray);
573         let rpy_x=parseFloat(rpyArray[0]);
574         let rpy_y=parseFloat(rpyArray[1]);
575         let rpy_z=parseFloat(rpyArray[2]);
576         this.x_rpy=new THREE.Vector3(rpy_x, rpy_z, -rpy_y);
577         console.log("x_rpy: "+this.x_rpy.x+","+this.x_rpy.y+","+this.x_rpy.z)
578       }
579     }
580   }
581   if(axis_y != 0){
582     this.y_direction=axis_y;
583     var printer3d_y=printer3d.robot.joint[i].child[0].$.link;
584     console.log(printer3d.robot.joint[i].child[0].$.link+" is the y_axis");
585     this.axis_y=myArray;
586     for(let j=0; j<printer3d.robot.link.length;j++ ){

```

```

587         if(printer3d.robot.link[j].$.name == printer3d_y){
588             this.url_y = printer3d.robot.link[j].visual[0].geometry[0].mesh[0].$.filename;
589             console.log("url_y: "+this.url_y);
590             let origin=printer3d.robot.joint[i].origin[0].$.xyz;
591             let xyzArray = origin.split(" ");
592             console.log(xyzArray);
593             let origin_x=parseFloat(xyzArray[0]);
594             let origin_y=parseFloat(xyzArray[1]);
595             let origin_z=parseFloat(xyzArray[2]);
596             this.y_start_pos=new THREE.Vector3(origin_x, origin_z, -origin_y);
597
598             let rpy=printer3d.robot.joint[i].origin[0].$.rpy;
599             let rpyArray = rpy.split(" ");
600             console.log(rpyArray);
601             let rpy_x=parseFloat(rpyArray[0]);
602             let rpy_y=parseFloat(rpyArray[1]);
603             let rpy_z=parseFloat(rpyArray[2]);
604             this.y_rpy=new THREE.Vector3(rpy_x, rpy_z, -rpy_y);
605             console.log("y_rpy: "+this.y_rpy.x+" "+this.y_rpy.y+" "+this.y_rpy.z)
606         }
607     }
608 }
609 if(axis_z != 0){
610     this.z_direction=axis_z;
611     var printer3d_z=printer3d.robot.joint[i].child[0].$.link;
612     console.log(printer3d.robot.joint[i].child[0].$.link+" is the z_axis");
613     this.axis_z=myArray;
614     for(let j=0; j<printer3d.robot.link.length;j++){
615         if(printer3d.robot.link[j].$.name == printer3d_z){
616             this.url_z = printer3d.robot.link[j].visual[0].geometry[0].mesh[0].$.filename;
617             console.log("url_z: "+this.url_z);
618
619             let origin=printer3d.robot.joint[i].origin[0].$.xyz;
620             let xyzArray = origin.split(" ");
621             console.log(xyzArray);
622             let origin_x=parseFloat(xyzArray[0]);
623             let origin_y=parseFloat(xyzArray[1]);
624             let origin_z=parseFloat(xyzArray[2]);
625             this.z_start_pos=new THREE.Vector3(origin_x, origin_z, -origin_y);
626
627             let rpy=printer3d.robot.joint[i].origin[0].$.rpy;
628             let rpyArray = rpy.split(" ");
629             console.log(rpyArray);
630             let rpy_x=parseFloat(rpyArray[0]);
631             let rpy_y=parseFloat(rpyArray[1]);
632             let rpy_z=parseFloat(rpyArray[2]);
633             this.z_rpy=new THREE.Vector3(rpy_x, rpy_z, -rpy_y);
634             console.log("z_rpy: "+this.z_rpy.x+" "+this.z_rpy.y+" "+this.z_rpy.z)
635         }

```

```

636     }
637 }
638 }
639 }
640
641 // Részegységek
642 this.base = wom.create("mesh", {
643     id: `base${this.componentID}`, // this can be used to select this object with wom.select()
644     url: this.url_base, // the name of the mesh in the resources folder
645     position: this.basePos.add(this.base_start_pos),
646     orientation: eulerToQuat(this.base_rpy),
647     scale: { x: this.baseScale.x, y: this.baseScale.y, z: this.baseScale.z },
648     autophysical: false, // add a physical collision shape for this object (bounding box)
649 });
650
651 this.x_axis = wom.create("mesh", {
652     id: `x_axis${this.componentID}`, // this can be used to select this object with wom.select()
653     url: this.url_x, // the name of the mesh in the resources folder
654     position: this.x_start_pos,
655     orientation: eulerToQuat(this.x_rpy),
656     scale: this.x_axis_scale,
657     autophysical: true, // add a physical collision shape for this object (bounding box)
658 });
659
660 this.y_axis = wom.create("mesh", {
661     id: `y_axis${this.componentID}`, // this can be used to select this object with wom.select()
662     url: this.url_y, // the name of the mesh in the resources folder
663     position: this.y_start_pos,
664     orientation: eulerToQuat(this.y_rpy),
665     scale: this.y_axis_scale, // initial scale
666     autophysical: true, // add a physical collision shape for this object (bounding box)
667 });
668
669 this.z_axis = wom.create("mesh", {
670     id: `z_axis${this.componentID}`, // this can be used to select this object with wom.select()
671     url: this.url_z, // the name of the mesh in the resources folder
672     position: this.z_start_pos,
673     orientation: eulerToQuat(this.z_rpy),
674     scale: this.z_axis_scale,
675     autophysical: true, // add a physical collision shape for this object (bounding box)
676 });
677
678 this.controlPanel = wom.create("billboard", {
679     id: `component-tutorial-wom-manipulation-billboard${options.componentID}`,
680     url: resolve(__dirname, "../billboardResources/index.html"),
681     nodeIntegration: true, // need this for IPC communication
682     width: this.controlPanelWidth, // width of the billboard (in OGRE units)
683     height: this.controlPanelHeight, // height of the billboard (in OGRE units)
684     scaleFactor: 1, // do not allow automatic DPI scaling for this billboard

```

```

685     "resolution-width": this.resolutionWidth, // resolution of the billboard in pixels
686     "resolution-height": this.resolutionHeight, // resolution of the billboard in pixels
687     position: this.controlPanel_pos,
688     orientation: eulerToQuat(this.controlPanel_rpy),
689     physical: { raycast: true }, // accept mouse events to make it orbitable
690 });
691
692 this.printBaseCube = wom.create("node", {
693     id: `printBaseCube${this.componentID}`,
694 });
695
696 this.printBaseTube = wom.create("node", {
697     id: `printBaseTube${this.componentID}`,
698 });
699
700 this.printCoordTracker = wom.create("node", {
701     id: `printCoordTracker${this.componentID}`,
702     position: {
703         x: 0,
704         y: 0,
705         z: 0,
706     },
707 });
708
709 this.spool = wom.create("mesh", {
710     id: `spool${this.componentID}`, // this can be used to select this object with wom.select()
711     url: this.url_spool, // the name of the mesh in the resources folder
712     position: this.spool_pos,
713     orientation: eulerToQuat(this.spool_rpy),
714     scale: { x: 1, y: 1, z: 1 }, // initial scale
715     autophysical: true, // add a physical collision shape for this object (bounding box)
716 });
717
718 this.spoolDisp_node = wom.create("mesh", {
719     id: `spoolDisp${this.componentID}`, // this can be used to select this object with wom.select()
720     url: "myCube.mesh",
721     position: this.spoolDisp_pos,
722     orientation: { x: 0, y: 0, z: 0, w: 1 },
723     scale: { x: 0.0001, y: 0.0001, z: 0.0001 }, // initial scale
724 });
725
726 this.spoolControl_node = wom.create("mesh", {
727     id: `spoolControl${this.componentID}`, // this can be used to select this object with wom.select()
728     position: this.spoolControl_pos,
729     url: "myCube.mesh",
730     orientation: { x: 0, y: 0, z: 0, w: 1 },
731     scale: { x: 0.0001, y: 0.0001, z: 0.0001 }, // initial scale
732 });
733

```

```

734 this.zInControl_node = wom.create("mesh", {
735     id: `zInControl${this.componentID}`, // this can be used to select this object with wom.select()
736     url: "myCube.mesh",
737     position: this.zInControl_pos,
738     orientation: { x: 0, y: 0, z: 0, w: 1 },
739     scale: { x: 0.0001, y: 0.0001, z: 0.0001 }, // initial scale
740 });
741
742 this.zInDisp_node = wom.create("mesh", {
743     id: `zInDisp${this.componentID}`, // this can be used to select this object with wom.select()
744     url: "myCube.mesh",
745     position: this.zInDisp_pos,
746     orientation: { x: 0, y: 0, z: 0, w: 1 },
747     scale: { x: 0.0001, y: 0.0001, z: 0.0001 }, // initial scale
748 });
749
750 this.zOutControl_node = wom.create("mesh", {
751     id: `zOutControl${this.componentID}`, // this can be used to select this object with wom.select()
752     url: "myCube.mesh",
753     position: this.zOutControl_pos,
754     orientation: { x: 0, y: 0, z: 0, w: 1 },
755     scale: { x: 0.0001, y: 0.0001, z: 0.0001 }, // initial scale
756 });
757
758 this.zOutDisp_node = wom.create("mesh", {
759     id: `zOutDisp${this.componentID}`, // this can be used to select this object with wom.select()
760     url: "myCube.mesh",
761     position: this.zOutDisp_pos,
762     orientation: { x: 0, y: 0, z: 0, w: 1 },
763     scale: { x: 0.0001, y: 0.0001, z: 0.0001 }, // initial scale
764 });
765
766 this.xInControl_node = wom.create("mesh", {
767     id: `xInControl${this.componentID}`, // this can be used to select this object with wom.select()
768     url: "myCube.mesh",
769     position: this.xInControl_pos,
770     orientation: { x: 0, y: 0, z: 0, w: 1 },
771     scale: { x: 0.000001, y: 0.000001, z: 0.000001 }, // initial scale
772 });
773
774 this.xInDisp_node = wom.create("mesh", {
775     id: `xInDisp${this.componentID}`, // this can be used to select this object with wom.select()
776     url: "myCube.mesh",
777     position: this.xInDisp_pos,
778     orientation: { x: 0, y: 0, z: 0, w: 1 },
779     scale: { x: 0.000001, y: 0.000001, z: 0.000001 }, // initial scale
780 });
781
782 this.table_node = wom.create("node", {

```

```

783     id: `table${this.componentID}`,
784     position: { x: 0, y: 0, z: 0 },
785     orientation: { x: 0, y: 0, z: 0, w: 1 },
786   });
787
788   this.cubeTracker = wom.create("node", {
789     id: `cubeTracker${this.componentID}`,
790     position: this.table_id==this.y_axis.id ? { x: this.y_start_pos.x, y: this.y_start_pos.y, z: this.
791       y_start_pos.z } : { x: this.z_start_pos.x, y: this.z_start_pos.y, z: this.z_start_pos.z },//this.
792       z_start_pos,// this.printerSource == "ender.xml" ? { x: 0, y: 0, z: 0 } : { x: 0, y: 23.72, z: 0 },
793   });
794   console.log("this.z_start_pos: "+this.z_start_pos.x+"; "+this.z_start_pos.y+"; "+this.z_start_pos.z)
795   console.log("cubetracker: "+this.cubeTracker.getPosition().x+"; "+this.cubeTracker.getPosition().y+"; "
796     +this.cubeTracker.getPosition().z)
797
798   // Részegységek parentálása a urdf modellnek megfelelően
799   for(let i=0; i < printer3d.robot.joint.length; i++){
800     console.log(i+": "+printer3d.robot.joint[i].parent[0].$.link+ " - "+printer3d.robot.joint[i].child[0].
801       $.link);
802     parents.push(printer3d.robot.joint[i].parent[0].$.link);
803     children.push(printer3d.robot.joint[i].child[0].$.link);
804     var parentNode=this.base;
805     var childNode=this.x_axis;
806
807     if(printer3d.robot.joint[i].parent[0].$.link==printer3d_x){
808       var parentNode=this.x_axis;
809     }
810     if(printer3d.robot.joint[i].parent[0].$.link==printer3d_y){
811       var parentNode=this.y_axis;
812     }
813     if(printer3d.robot.joint[i].parent[0].$.link==printer3d_z){
814       var parentNode=this.z_axis;
815     }
816     if(printer3d.robot.joint[i].parent[0].$.link==printer3d_base){
817       var parentNode=this.base;
818     }
819     this.tree0=this.base.id;
820     if(printer3d.robot.joint[i].child[0].$.link==printer3d_x){
821       var childNode=this.x_axis;
822       console.log("xtengely: "+printer3d.robot.joint[i].parent[0].$.link+" "+this.basename);
823
824       if(printer3d.robot.joint[i].parent[0].$.link!=this.basename){
825         this.tree3=this.x_axis.id;
826         if(printer3d.robot.joint[i].parent[0].$.link==printer3d_y){
827           this.tree2=this.y_axis.id;
828           this.tree1=this.z_axis.id;
829         }
830         if(printer3d.robot.joint[i].parent[0].$.link==printer3d_z){
831           this.tree2=this.z_axis.id;

```

```

828         this.tree1=this.y_axis.id;
829     }
830 }
831 }
832 if(printer3d.robot.joint[i].child[0].$.link==printer3d_y){
833     var childNode=this.y_axis;
834     if(printer3d.robot.joint[i].parent[0].$.link!=this.basename){
835         this.tree3=this.y_axis.id;
836         if(printer3d.robot.joint[i].parent[0].$.link==printer3d_x){
837             this.tree2=this.x_axis.id;
838             this.tree1=this.z_axis.id;
839         }
840         if(printer3d.robot.joint[i].parent[0].$.link==printer3d_z){
841             this.tree2=this.z_axis.id;
842             this.tree1=this.x_axis.id;
843         }
844     }
845 }
846 if(printer3d.robot.joint[i].child[0].$.link==printer3d_z){
847     var childNode=this.z_axis;
848     if(printer3d.robot.joint[i].parent[0].$.link!=this.basename){
849         this.tree3=this.z_axis.id;
850         if(printer3d.robot.joint[i].parent[0].$.link==printer3d_y){
851             this.tree2=this.y_axis.id;
852             this.tree1=this.x_axis.id;
853         }
854         if(printer3d.robot.joint[i].parent[0].$.link==printer3d_x){
855             this.tree2=this.x_axis.id;
856             this.tree1=this.y_axis.id;
857         }
858     }
859 }
860 if(printer3d.robot.joint[i].child[0].$.link==this.basename){
861     var childNode=this.base;
862     this.tree0=this.base.id;
863 }
864 if(printer3d.robot.joint[i].child[0].$.link=="Spool"){
865     var childNode=this.spool;
866 }
867 if(printer3d.robot.joint[i].child[0].$.link=="spoolDisp"){
868     var childNode=this.spoolDisp_node;
869 }
870 if(printer3d.robot.joint[i].child[0].$.link=="spoolControl"){
871     var childNode=this.spoolControl_node;
872 }
873 if(printer3d.robot.joint[i].child[0].$.link=="zInControl"){
874     var childNode=this.zInControl_node;
875 }
876 if(printer3d.robot.joint[i].child[0].$.link=="zInDisp"){

```

```

877     var childNode=this.zInDisp_node;
878 }
879 if(printer3d.robot.joint[i].child[0].$.link=="zOutDisp"){
880     var childNode=this.zOutDisp_node;
881 }
882 if(printer3d.robot.joint[i].child[0].$.link=="zOutControl"){
883     var childNode=this.zOutControl_node;
884 }
885 if(printer3d.robot.joint[i].child[0].$.link=="xInControl"){
886     var childNode=this.xInControl_node;
887 }
888 if(printer3d.robot.joint[i].child[0].$.link=="xInDisp"){
889     var childNode=this.xInDisp_node;
890 }
891 if(printer3d.robot.joint[i].child[0].$.link=="table"){
892     var childNode=this.table_node;
893     this.table_id=parentNode.id;
894 }
895 if(printer3d.robot.joint[i].child[0].$.link=="fan"){
896     this.ext_fan = wom.create("mesh", {
897         id: `ext_fan${this.componentID}`, // this can be used to select this object with wom.select()
898         url: this.url_fan, // the name of the mesh in the resources folder
899         position: this.fan_pos,
900         orientation: eulerToQuat(this.fan_rpy),
901         scale: this.printerSource == "ender.xml" ? { x: 1, y: 1, z: 1 } : { x: 100, y: 100, z: 100}, //
initial scale
902         autophysical: false, // add a physical collision shape for this object (bounding box)
903     });
904     var childNode=this.ext_fan;
905     this.fanController = new Fan(100, 130, this.fan_axis, this.ext_fan, this);
906     this.headTemperatureSensor = new TemperatureSimulator(this.fanController, this, 20, 0.5);
907 }
908 if(printer3d.robot.joint[i].child[0].$.link!="controlPanel")
909 {
910     parentNode.appendChild(childNode);
911     console.log(childNode.id);
912     console.log(parentNode.id+".appendChild("+childNode.id+");");
913 }
914 }
915
916 // nyomtatást segítő node-ok parentálása
917 this.base.appendChild(this.printCoordTracker);
918 this.cubeTracker.appendChild(this.printBaseCube);
919 this.cubeTracker.appendChild(this.printBaseTube);
920 this.cubeTracker.appendChild(this.buildTube.getManualVisual());
921
922 // lekezeljük melyik alkatrész az asztal
923 if(this.table_id == this.y_axis.id){
924     this.y_axis.appendChild(this.cubeTracker)}

```

```

925     if(this.table_id == this.z_axis.id){
926         this.base.appendChild(this.cubeTracker)};
927
928     // vezérlőpanel parentálása
929     this.base.appendChild(this.controlPanel);
930
931
932     // Megvizsgáljuk minden pont létrejött-e a filament generálásához, ha igen, rájuk lehet illeszteni a görbét
933     this.zInDisp_node.on('created', ()=>{
934         console.log("this.zInDisp_created=true;");
935         this.zInDisp_created=true;
936         console.log(this.zInControl_created+" "+this.zInDisp_created+" "+this.spoolControl_created+" "+this.spoolDisp_created);
937         this.createFilament();
938     });
939     this.zInControl_node.on('created', ()=>{
940         console.log("this.zInControl_created=true;");
941         this.zInControl_created=true;
942         console.log(this.zInControl_created+" "+this.zInDisp_created+" "+this.spoolControl_created+" "+this.spoolDisp_created);
943         this.createFilament();
944     });
945     this.spoolControl_node.on('created', ()=>{
946         console.log("this.spoolControl_created=true;");
947         this.spoolControl_created=true;
948         console.log(this.zInControl_created+" "+this.zInDisp_created+" "+this.spoolControl_created+" "+this.spoolDisp_created);
949         this.createFilament();
950     });
951     this.spoolDisp_node.on('created', ()=>{
952         console.log("this.spoolDisp_created=true;");
953         this.spoolDisp_created=true;
954         console.log(this.zInControl_created+" "+this.zInDisp_created+" "+this.spoolControl_created+" "+this.spoolDisp_created);
955         this.createFilament();
956     });
957
958     // Megvizsgáljuk hogy az utolsó pont is létrejött-e a cső generálásához, ha igen, rájuk lehet illeszteni a görbét
959     this.xInDisp_node.on('created', ()=>{
960         this.createSleeve();
961     })
962
963     // nyomtató renderelése
964     wom.render(this.base);
965
966     // Alapértelmezett g-code amivel elindul a szimuláció
967     var pathDemoFile = path.resolve(

```

```

968     __dirname,
969     "../gcode_examples/steam.gcode"
970 );
971 const contentsDemoFile = readFileSync(pathDemoFile, { encoding: 'utf-8' })
972 this.readCode = new GcodeReader(contentsDemoFile);
973
974 // IPC kommunikáció a weblappal
975 // ipcMain.removeAllListeners("compdev-tutorial-wom-manipulation");
976 ipcMain.on("compdev-tutorial-wom-manipulation" , (event, obj) => {
977     if (this.controlPanel.webview.browserWindow.webContents == event.sender) {
978         console.log(TAG, obj.cmd);
979         switch (obj.cmd) {
980             case "reset":
981                 this.reset();
982                 break;
983             case "start":
984                 this.start();
985                 break;
986             case "stop":
987                 this.stop();
988                 break;
989             case "browse":
990                 this.browse();
991                 break;
992             case "setTimescale":
993                 this.timeScale = obj.value;
994                 break;
995             case "red":
996                 this.red = parseFloat(obj.r);
997                 this.FILAMENT_COLOR={r: this.red, g: this.green, b:this.blue};
998                 break;
999             case "green":
1000                 this.green = parseFloat(obj.g);
1001                 this.FILAMENT_COLOR={r: this.red, g: this.green, b:this.blue};
1002                 break;
1003             case "blue":
1004                 this.blue = parseFloat(obj.b);
1005                 this.FILAMENT_COLOR={r: this.red, g: this.green, b:this.blue};
1006                 break;
1007             case "tree":
1008                 event.reply('ipc-tree1', this.tree0);
1009                 event.reply('ipc-tree2', this.tree1);
1010                 event.reply('ipc-tree3', this.tree2);
1011                 event.reply('ipc-tree4', this.tree3);
1012                 break;
1013         }
1014     }
1015 });
1016

```

```

1017 // Mozgás figyelése
1018 wom.on(
1019   {
1020     type: "node-position",
1021     subject: this.printCoordTracker.id,
1022   },
1023   (event: object) => {
1024     try {
1025       // Tengelyek elmozdulása
1026       this.xMoved += Math.abs(this.x_axis.getPosition().x - this.x_axisPos.x)+Math.abs(this.x_axis.
1027         getPosition().z - this.x_axisPos.z)+Math.abs(this.x_axis.getPosition().y - this.x_axisPos.y);
1028       this.yMoved += Math.abs(this.y_axis.getPosition().z - this.y_axisPos.z)+Math.abs(this.y_axis.
1029         getPosition().x - this.y_axisPos.x)+Math.abs(this.y_axis.getPosition().y - this.y_axisPos.y);
1030       this.zMoved += Math.abs(this.z_axis.getPosition().x - this.z_axisPos.x)+Math.abs(this.z_axis.
1031         getPosition().z - this.z_axisPos.z)+Math.abs(this.z_axis.getPosition().y - this.z_axisPos.y);
1032
1033       this.xMovedBuild += Math.abs(this.x_axis.getPosition().x - this.x_axisPos.x)+Math.abs(this.
1034         x_axis.getPosition().z - this.x_axisPos.z)+Math.abs(this.x_axis.getPosition().y - this.x_axisPos.y);
1035       this.yMovedBuild += Math.abs(this.y_axis.getPosition().z - this.y_axisPos.z)+Math.abs(this.y_axis.
1036         getPosition().x - this.y_axisPos.x)+Math.abs(this.y_axis.getPosition().y - this.y_axisPos.y);
1037       this.zMovedBuild += Math.abs(this.z_axis.getPosition().x - this.z_axisPos.x)+Math.abs(this.z_axis.
1038         getPosition().z - this.z_axisPos.z)+Math.abs(this.z_axis.getPosition().y - this.z_axisPos.y);
1039
1040       // Tengelyek pozíciója
1041       this.x_axisPos = V3ToThree(this.x_axis.getPosition());
1042       this.y_axisPos = V3ToThree(this.y_axis.getPosition());
1043       this.z_axisPos = V3ToThree(this.z_axis.getPosition());
1044
1045       // nyomtatvány elhelyezése az asztalon
1046       if(this.table_id == this.y_axis.id){
1047         this.buildTube.setPosition({ x: 0, y: 0, z: this.y_axisPos.z-this.y_start_pos.z});
1048       }
1049       if(this.table_id == this.z_axis.id){
1050         this.buildTube.setPosition({ x: 0, y: this.z_axisPos.y-this.z_start_pos.y*this.baseScale.y-this.
1051           basePos.y, z: 0});} // this.buildTube.setPosition({ x: 0, y: -this.command.coords.z/10*2, z:
1052           0});
1053
1054       console.log("-this.command.coords.z/10*this.baseScale.y: "+-this.command.coords.z/10*this.
1055         baseScale.y+" ; "+"this.z_axisPos.y: "+this.z_axisPos.y+" ; "+"this.z_start_pos: "+this.z_start_pos.y+
1056         this.basePos.y);
1057
1058       // lelógó filament újrarajzolása
1059       if (Math.abs(this.zMoved) > this.eps * this.baseScale.x) {
1060         this.redrawFilament();
1061         console.log("cubetracker: "+this.cubeTracker.getPosition().x+"; "+this.cubeTracker.getPosition
1062           ().y+"; "+this.cubeTracker.getPosition().z)
1063       }
1064
1065       // cső újrarajzolása
1066       if (Math.abs(this.xMoved) > this.eps * this.baseScale.x) {

```

```

1055     this.xMoved = 0;
1056     // console.log("sleeve redraw start");
1057     this.redrawSleeve();
1058     // console.log("sleeve redraw end");
1059 }
1060 console.log("zaxispos "+this.z_axisPos.x+"; "+this.z_axisPos.y+"; "+this.z_axisPos.z);
1061 console.log("xaxispos "+this.x_axisPos.x+"; "+this.x_axisPos.y+"; "+this.x_axisPos.z);
1062
1063 //tubegeometry pontjai
1064 const headPosRelY = new THREE.Vector3(
1065     this.x_axisPos.x + (this.headOffset.x * this.baseScale.x),
1066     this.table_id ==this.y_axis.id ? this.z_axisPos.y + (this.headOffset.y * this.baseScale.y) :
this.z_axisPos.y + (this.headOffset.y * this.baseScale.y)-(this.z_axisPos.y-this.z_start_pos.y*this.
baseScale.y-this.basePos.y)*2,/+10.1+this.command.coords.z/10*4,
1067     this.table_id ==this.y_axis.id ? this.x_axisPos.z + (this.headOffset.z * this.baseScale.z)-
this.y_axisPos.z : this.x_axisPos.z + (this.headOffset.z * this.baseScale.z),/-27.46
1068 );
1069
1070 let curveOpt = {
1071     tubularSegments: this.buildTubeSegments,
1072     radius: 0.05 * this.baseScale.x,
1073     radialSegments: 4,
1074 };
1075 let color = this.FILAMENT_COLOR;
1076 console.log("filament_color: "+color.r+" ; "+color.g+" ; "+color.b)
1077
1078 if (
1079     Math.sqrt(
1080         this.xMovedBuild * this.xMovedBuild +
1081         this.yMovedBuild * this.yMovedBuild +
1082         this.zMovedBuild * this.zMovedBuild
1083     ) > this.buildEps
1084 ) {
1085     if (this.isExtruding) {
1086         this.lastPrintInExtrude = true;
1087         this.xMovedBuild = 0;
1088         this.yMovedBuild = 0;
1089         this.zMovedBuild = 0;
1090         if(this.tubePointsArray.length>0){
1091             console.log("tubepointsarray_diff: ("+headPosRelY.x+", "+headPosRelY.y+", "+headPosRelY.z+")
---- ("+this.tubePointsArray[this.tubePointsArray.length-1].x+", "+this.tubePointsArray[this.
tubePointsArray.length-1].y+", "+this.tubePointsArray[this.tubePointsArray.length-1].z+"");
1092         }
1093         this.tubePointsArray.push(
1094             headPosRelY
1095         );
1096
1097
1098         // nyomtatott kiskocka

```

```

1099         // @ts-expect-error instancedmesh exists
1100         let buildSegment = wom.create("instancedmesh", {
1101             id: `buildSegment${this.componentID}${Math.random() * 10000}${this.tubePointsArray.length
1102         }`, // this can be used to select this object with wom.select()
1103             url: "myCube.mesh", // the name of the mesh in the resources folder
1104             position: {
1105                 x: (this.x_axisPos.x - this.basePos.x) / this.baseScale.x + this.headOffset.x,
1106                 z: (this.z_direction*this.basePos.z - this.z_direction*this.y_axisPos.z) / this.baseScale
1107             .y + this.headOffset.z
1108             },
1109             orientation: { x: 0, y: 0, z: 0, w: 1 },
1110             scale: {
1111                 x: this.cubeWidth,
1112                 y: this.cubeWidth,
1113                 z: this.cubeWidth,
1114             },
1115             autophysical: false, // add a physical collision shape for this object (bounding box)
1116         });
1117         wom.render(buildSegment);
1118
1119         this.printBaseCube.appendChild(buildSegment);
1120         console.log("tubePointsArray :"+this.tubePointsArray)
1121
1122         if (this.tubePointsArray.length === this.maxBuild) {
1123             console.log(TAG, "Reached 100");
1124             const curve = new THREE.CatmullRomCurve3(this.tubePointsArray);
1125             console.log(this.buildTube.getPosition());
1126             this.buildTube.appendSection(curve, curveOpt, color);
1127             this.printBaseCube.clear();
1128             this.printBaseCube = wom.create("node", {
1129                 id: `printBaseCube${this.componentID}`,
1130                 position: {
1131                     x: 0,
1132                     y: 0,
1133                     z: 0,
1134                 },
1135             });
1136             this.cubeTracker.appendChild(this.printBaseCube);
1137
1138             this.tubePointsArray.length = 0;
1139             this.tubePointsArray.push(headPosRelY.clone());
1140
1141             console.log(this.buildTube.getNumberOfSections());
1142         }
1143     } else {
1144         if (this.lastPrintInExtrude) {
1145             console.log(TAG, "Didn't reach 100");
1146
1147             if (this.tubePointsArray.length > 1) {

```

```

1146         const curve = new THREE.CatmullRomCurve3(
1147             this.tubePointsArray
1148         );
1149         console.log(TAG, this.buildTube.getPosition());
1150         if(this.isPrinting){
1151             this.buildTube.appendSection(curve, curveOpt, color);}
1152         console.log(TAG, this.buildTube.getNumberOfSections());
1153     } else {
1154         console.log(TAG, "Not now!");
1155     }
1156     this.lastPrintInExtrude = false;
1157     this.printBaseCube.clear();
1158     this.printBaseCube = wom.create("node", {
1159         id: `printBaseCube${this.componentID}`,
1160         position: {
1161             x: 0,
1162             y: 0,
1163             z: 0,
1164         },
1165     });
1166     this.cubeTracker.appendChild(this.printBaseCube);
1167     this.tubePointsArray.length = 0;
1168 }
1169 }
1170 }
1171 } catch (error) {
1172     console.error(error);
1173 }
1174 }
1175 );
1176 }
1177
1178 doPrint() {
1179     if (this.readCode.currCommandIndex <= this.readCode.commands.length - 1) {
1180         this.scheduleNextCommand();
1181     } else {
1182         console.log(TAG, "The End");
1183     }
1184 }
1185
1186 animateMulti(animations: Array<{
1187     node: Node,
1188     to: Vector3,
1189 }>, cbComplete: () => void) {
1190     const promises = new Array<Promise<void>>();
1191     for (const anim of animations) {
1192         const promise = new Promise<void>(resolve => anim.node.animate('position', {
1193             reference: 'absolute',
1194             to: anim.to,

```

```

1195     duration: this.dur,
1196     loop: false,
1197     cubic: 'linear',
1198   }, resolve));
1199   promises.push(promise);
1200 }
1201
1202   Promise.all(promises).then(cbComplete);
1203 }
1204
1205   scheduleNextCommand() {
1206     if (this.pendingStop) {
1207       console.warn(TAG, 'scheduleNextCommand failed: stopped');
1208       this.pendingStop = false;
1209       return;
1210     }
1211
1212     if (this.readCode.currCommandIndex <= this.readCode.commands.length - 1) {
1213       this.animate(() => {
1214         this.player = setTimeout(() => {
1215           this.player = null;
1216           this.scheduleNextCommand();
1217         }, WAIT_BETWEEN_COMMANDS_MS);
1218       });
1219     } else {
1220       console.log(TAG, "Printing has ended");
1221     }
1222   }
1223
1224   animate(cbComplete: () => void) {
1225     this.command = this.readCode.nextCommand();
1226     console.log(this.command);
1227
1228     this.isExtruding = this.extrude < this.command.extrusion ? true : false;
1229     this.prevExtrude = this.extrude;
1230     this.extrude = this.command.extrusion;
1231
1232     let x_axisPos: THREE.Vector3 = V3ToThree(this.x_axis.getPosition());
1233     let y_axisPos: THREE.Vector3 = V3ToThree(this.y_axis.getPosition());
1234     let z_axisPos: THREE.Vector3 = V3ToThree(this.z_axis.getPosition());
1235
1236     this.spoolDisp = V3ToThree(this.spoolDisp_node.getPosition());
1237     this.spoolControl = V3ToThree(this.spoolControl_node.getPosition());
1238     this.zInControl = V3ToThree(this.zInControl_node.getPosition());
1239     this.zInDisp = V3ToThree(this.zInDisp_node.getPosition());
1240     console.log("spooldisp: "+this.spoolDisp.x)
1241
1242     let headPos = new THREE.Vector3(

```

```

1243     this.x_direction*(x_axisPos.x - this.basePos.x-this.x_start_pos.x*this.baseScale.x) / this.baseScale.
1244     x,
1245     -this.y_direction*(y_axisPos.z - this.basePos.z-this.y_start_pos.z*this.baseScale.y) / this.baseScale
1246     .y,
1247     this.z_direction*(z_axisPos.y - this.basePos.y-this.z_start_pos.y*this.baseScale.z) / this.baseScale.
1248     z
1249     ).multiplyScalar(10);
1250 let coordIn = new THREE.Vector3(
1251     this.command.coords.x,
1252     this.command.coords.y,
1253     this.command.coords.z
1254 );
1255 console.log("xdist");
1256 console.log(coordIn.x - headPos.x);
1257 console.log("ydist");
1258 console.log(coordIn.y-headPos.y);
1259 console.log("zdist");
1260 console.log(coordIn.z-headPos.z);
1261
1262 let diff: THREE.Vector3 = coordIn.clone().sub(headPos);
1263 let distance = diff.length();
1264 console.log("coordin: "+coordIn.y);
1265 console.log("headpos: "+headPos);
1266
1267 this.dur = (distance / this.command.feedRate) * 60 / this.timeScale;
1268 console.log("headposxyz: (" +headPos.x+", "+headPos.y+", "+headPos.z+") coordinxyz: (" + coordIn.x+", "
1269 + coordIn.y+", "+ coordIn.z+") distance: "+distance+" dur: "+this.dur);
1270
1271 console.log(TAG, `Duration: ${this.dur}`);
1272 console.log("basepos("+this.basePos.x+", "+this.basePos.y+", "+this.basePos.z+")");
1273
1274 console.log("axis_x: "+this.axis_x+" xstart: "+this.x_start_pos.x);
1275 console.log("axis_y: "+this.axis_y+" ystart: "+this.y_start_pos.z);
1276 console.log("axis_z: "+this.axis_z+" zstart: "+this.z_start_pos.y);
1277
1278 const targetPrintTracker = new THREE.Vector3(
1279     this.x_start_pos.x-this.command.coords.x*this.axis_x[0]/10,
1280     this.z_start_pos.z-this.command.coords.z*this.axis_z[2]/10,
1281     this.y_start_pos.y-this.command.coords.y*this.axis_y[1]/10
1282 );
1283
1284 const targetXAxis = new THREE.Vector3(this.x_start_pos.x+this.command.coords.x*this.axis_x[0]/10, 0, 0)
1285 ;
1286 const targetYAxis = new THREE.Vector3(0, 0, this.y_start_pos.z-this.command.coords.y*this.axis_y[1]/
1287 10);
1288 const targetZAxis = new THREE.Vector3(0, this.z_start_pos.y+this.command.coords.z*this.axis_z[2]/10, 0)
1289 ;
1290 console.log("buildtubepos "+this.buildTube.getPosition().y)

```

```

1285     this.animateMulti([
1286         {
1287             node: this.x_axis,
1288             to: V3ToMw(targetXAxis),
1289         },
1290         {
1291             node: this.y_axis,
1292             to: V3ToMw(targetYAxis),
1293         },
1294         {
1295             node: this.z_axis,
1296             to: V3ToMw(targetZAxis),
1297         },
1298         {
1299             node: this.printCoordTracker,
1300             to: V3ToMw(targetPrintTracker),
1301         },
1302     ], cbComplete);
1303 }
1304
1305 denimate() {
1306     this.x_axis.deanimate("position");
1307
1308     this.y_axis.deanimate("position");
1309
1310     this.z_axis.deanimate("position");
1311
1312     this.printCoordTracker.deanimate("position");
1313 }
1314
1315
1316 animateToStart() {
1317     this.isExtruding = false;
1318     this.isPrinting = false;
1319     let x_axisPos: THREE.Vector3 = V3ToThree(this.x_axis.getPosition());
1320     let y_axisPos: THREE.Vector3 = V3ToThree(this.y_axis.getPosition());
1321
1322     let headPos = new THREE.Vector3(
1323         (x_axisPos.x - this.basePos.x-this.axis_x[0]) / this.baseScale.x,
1324         (y_axisPos.z - this.basePos.z+this.axis_z[2]) / this.baseScale.y,
1325         (x_axisPos.y - this.basePos.y+this.axis_y[1]) / this.baseScale.z
1326     ).multiplyScalar(10);
1327     let coordIn = new THREE.Vector3(0, 0, 0);
1328
1329     let diff: THREE.Vector3 = coordIn.clone().sub(headPos);
1330     let distance = diff.length();
1331
1332     this.dur = (distance / this.command.feedRate) * 60 / this.timeScale;
1333     console.log(`distance to start: ${distance}`)

```

```

1334     console.log(`time to start: ${this.dur}`)
1335
1336     this.animateMulti([
1337         {
1338             node: this.x_axis,
1339             to: this.x_start_pos
1340         },
1341         {
1342             node: this.y_axis,
1343             to: this.y_start_pos
1344         },
1345         {
1346             node: this.z_axis,
1347             to: this.z_start_pos
1348         },
1349         {
1350             node: this.printCoordTracker,
1351             to: {
1352                 x: this.command.coords.x / 10,
1353                 y: this.command.coords.y / 10,
1354                 z: this.command.coords.z / 10,
1355             },
1356         }
1357     ], () => {
1358     });
1359 }
1360
1361 clearResources() {
1362     this.stop();
1363     if(this.isCooling){
1364         this.headTemperatureSensor.stopSimulation();
1365         this.fanController.stopAnimation();
1366     }
1367 }
1368
1369 stop() {
1370     console.log(this.isPrinting);
1371     console.log(this.readCode.currCommandIndex);
1372     console.log(this.readCode.commands.length - 2);
1373
1374     if (
1375         this.isPrinting &&
1376         this.readCode.currCommandIndex <= this.readCode.commands.length - 2
1377     ) {
1378         this.isPrinting = false;
1379         this.pendingStop = true;
1380         this.denimate();
1381         this.readCode.decreaseCommandIndex();
1382         this.newPrint = false;

```

```

1383     }
1384     console.log(TAG, this.newPrint);
1385 }
1386
1387 start() {
1388     if (
1389         !this.isPrinting &&
1390         this.readCode.currCommandIndex <= this.readCode.commands.length - 2
1391     ) {
1392         this.isPrinting = true;
1393         this.extrude = this.prevExtrude;
1394
1395         console.log(this.newPrint);
1396
1397         // clear unit cubes
1398         if (this.newPrint) {
1399             this.printBaseCube.clear();
1400             this.printBaseCube = wom.create("node", {
1401                 id: `printBase${this.componentID}`,
1402             });
1403
1404             this.cubeTracker.appendChild(this.printBaseCube);
1405             const buildTubeMaterialName = this.buildTube.getMaterialName();
1406             this.buildTube.clear();
1407             this.buildTube = makeCurveMesh(wom, {
1408                 materialName: buildTubeMaterialName,
1409             });
1410         }
1411
1412         if (this.pendingStop) {
1413             console.warn(TAG, 'pending stop flag didn\'t get cleared');
1414         }
1415
1416         this.pendingStop = false;
1417
1418         this.scheduleNextCommand();
1419     }
1420 }
1421
1422 reset() {
1423     this.stop();
1424     this.command = {
1425         coords: new THREE.Vector3(0, 0, 0),
1426         feedRate: 2000,
1427         extrusion: 0,
1428     };
1429     this.isExtruding = false;
1430     this.isPrinting = false;
1431     this.buildTube.clear();

```

```

1432     this.printBaseCube.clear();
1433     this.animateToStart();
1434     this.isPrinting = false;
1435     this.extrude = 0;
1436     this.prevExtrude = 0;
1437     this.isExtruding = false;
1438     this.readCode.reset();
1439
1440     this.printBaseCube = wom.create("node", {
1441         id: `printBase${this.componentID}`,
1442     });
1443
1444     this.cubeTracker.appendChild(this.printBaseCube);
1445     const buildTubeMaterialName = this.buildTube.getMaterialName();
1446     this.buildTube = makeCurveMesh(wom, {
1447         materialName: buildTubeMaterialName,
1448     });
1449     this.newPrint = true;
1450 }
1451
1452 browse() {
1453     const options = {
1454         extensions: ['gcode', 'nc'],
1455     };
1456     beginLoadTextViaDialog(options, (err, res) => {
1457         if (err !== null) {
1458             console.error(TAG, 'Error while loading gcode: ', err);
1459             return;
1460         }
1461
1462         console.log(TAG, 'Loading gcode from', res.path);
1463         this.readCode = new GcodeReader(res.contents);
1464     });
1465 }
1466
1467 createFilament() {
1468     if(this.zInControl_created && this.zInDisp_created && this.spoolControl_created && this.
        spoolDisp_created){
1469         console.log("filament is being generated");
1470         this.spoolDisp = V3ToThree(this.spoolDisp_node.getPosition());
1471         this.spoolControl = V3ToThree(this.spoolControl_node.getPosition());
1472         this.zInControl = V3ToThree(this.zInControl_node.getPosition());
1473         this.zInDisp = V3ToThree(this.zInDisp_node.getPosition());
1474         this.filHandleArr = [
1475             this.spoolDisp,
1476             this.spoolControl,
1477             this.zInControl,
1478             this.zInDisp,
1479         ];

```

```

1480     this.filamentBezier = new THREE.CubicBezierCurve3(...this.filHandleArr);
1481     console.log("spooldisp: "+this.spoolDisp.x+"; "+this.spoolDisp.y+"; "+this.spoolDisp.z)
1482     console.log("spooldisp_node: "+this.spoolDisp_node.getPosition().x+"; "+this.spoolDisp_node.
getPosition().y+"; "+this.spoolDisp_node.getPosition().z)

1483
1484     this.filament.appendSection(
1485         this.filamentBezier,
1486         {
1487             tubularSegments: FILAMENT_TUBULAR_SEGMENTS,
1488             radius: FILAMENT_RADIUS_SCALE * this.baseScale.x,
1489             radialSegments: FILAMENT_RADIAL_SEGMENTS
1490         },
1491         this.FILAMENT_COLOR
1492     );
1493     this.spoolDisp_created=false;
1494     this.spoolControl_created=false;
1495     this.zInControl_created=false;
1496     this.zInDisp_created=false;
1497 }
1498 }
1499 createSleeve(){
1500     this.zOutDisp = V3ToThree(this.zOutDisp_node.getPosition());
1501     this.zOutControl = V3ToThree(this.zOutControl_node.getPosition());
1502     this.xInControl = V3ToThree(this.xInControl_node.getPosition());
1503     this.xInDisp = V3ToThree(this.xInDisp_node.getPosition());
1504     this.sleeveHandleArr = [
1505         this.zOutDisp,
1506         this.zOutControl,
1507         this.xInControl,
1508         this.xInDisp
1509     ];
1510
1511     this.sleeveBezier = new THREE.CubicBezierCurve3(...this.sleeveHandleArr);
1512     this.sleeve = makeCurveMesh(wom, {
1513         materialName: SLEEVE_MATERIAL_NAME
1514     });
1515
1516     this.sleeve.appendSection(
1517         this.sleeveBezier,
1518         {
1519             tubularSegments: SLEEVE_TUBULAR_SEGMENTS,
1520             radius: SLEEVE_RADIUS_SCALE * this.baseScale.x,
1521             radialSegments: SLEEVE_RADIAL_SEGMENTS
1522         },
1523         SLEEVE_COLOR
1524     );
1525 }
1526 redrawFilament(){
1527     this.spoolDisp = V3ToThree(this.spoolDisp_node.getPosition());

```

```

1528     this.spoolControl = V3ToThree(this.spoolControl_node.getPosition());
1529     this.zInControl = V3ToThree(this.zInControl_node.getPosition());
1530     this.zInDisp = V3ToThree(this.zInDisp_node.getPosition());
1531     this.filHandleArr = [
1532         this.spoolDisp,
1533         this.spoolControl,
1534         this.zInControl,
1535         this.zInDisp,
1536     ];
1537     this.filamentBezier = new THREE.CubicBezierCurve3(...this.filHandleArr);
1538     console.log("spooldisp: "+this.spoolDisp.x+"; "+this.spoolDisp.y+"; "+this.spoolDisp.z)
1539     console.log("spooldisp_node: "+this.spoolDisp_node.getPosition().x+"; "+this.spoolDisp_node.getPosition
        ().y+"; "+this.spoolDisp_node.getPosition().z)

1540
1541     console.log("filament redraw start");
1542     this.filament.changeSection(
1543         0,
1544         this.filamentBezier,
1545         { tubularSegments: 50, radius: 0.04 * this.baseScale.x, radialSegments: 3 },
1546         this.FILAMENT_COLOR
1547     );
1548 }
1549 redrawSleeve(){
1550     this.zOutDisp = V3ToThree(this.zOutDisp_node.getPosition());
1551     this.zOutControl = V3ToThree(this.zOutControl_node.getPosition());
1552     this.xInControl = V3ToThree(this.xInControl_node.getPosition());
1553     this.xInDisp = V3ToThree(this.xInDisp_node.getPosition());
1554     this.sleeveHandleArr = [
1555         this.zOutDisp,
1556         this.zOutControl,
1557         this.xInControl,
1558         this.xInDisp
1559     ];
1560     console.log("X_axis_position: " + this.x_axis.getPosition().x+"; "+this.x_axis.getPosition().y+"; "+
        this.x_axis.getPosition().z);
1561     this.sleeveBezier = new THREE.CubicBezierCurve3(...this.sleeveHandleArr);
1562     this.sleeve.changeSection(
1563         0,
1564         this.sleeveBezier,
1565         { tubularSegments: 30, radius: 0.18 * this.baseScale.x, radialSegments: 3 },
1566         SLEEVE_COLOR
1567     );
1568 }
1569 }

```

M.5.1. kódrészlet. Printer.ts

M.6. index.html

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8" />
5     <meta
6       name="viewport"
7       content="width=device-width, initial-scale=1, shrink-to-fit=no"
8     />
9     <link
10      rel="stylesheet"
11      href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
12      integrity="sha384-Vkoo8x4CGs03+Hhvx8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
13      crossorigin="anonymous"
14    />
15     <link rel="stylesheet" href="control.css">
16   </head>
17
18   <body>
19     <div class="container" style="padding-top: 10px;">
20       <div class="row justify-content-center">
21         <a href="controls.html">
22           <button
23             type="button"
24             id="controls"
25             class="m-2 btn-lg btn-controls"
26             style="width: 150px;"
27           >
28             CONTROLS
29           </button>
30         </a>
31         <br />
32         <a href="tree.html">
33           <button
34             type="button"
35             id="tree"
36             class="m-2 btn-lg btn-controls"
37             style="width: 150px;"
38             onclick="tree()"
39           >
40             TREE VIEW
41           </button>
42         </a>
43       </div>
44
45       <div class="row justify-content-center">
46         <button
47           type="button"
```

```

48     id="browse"
49     class=" m-2 btn-lg btn-gray"
50     style="width: 150px;"
51     onclick="browse(); reset()"
52     >
53     BROWSE</button
54     ><br />
55 </div>
56
57 <div class="row justify-content-center">
58     <button
59         type="button"
60         class=" m-2 btn-lg btn-gray"
61         style="width: 120px;"
62         onclick="start()"
63     >
64     START</button
65     ><br />
66     <button
67         type="button"
68         class="btn btn-danger btn-lg m-2"
69         style="color: blak; font-weight: bold; width: 150px;"
70
71         onclick="stop()"
72     >
73     STOP</button
74     ><br />
75     <button
76         type="button"
77         class=" m-2 btn-lg btn-gray"
78         style="width: 120px;"
79         onclick="reset()"
80     >
81     RESET</button
82     ><br />
83 </div>
84 <div class="container">
85     <div class="row justify-content-center" id="rgb" style="border-radius: 4px; background-color: rgb
(102, 102, 0); padding: 5px;">
86         <input id="slideRed" type="range" min="0" max="1" value="0.4" step="0.01" style="accent-color:
rgb(255, 0, 0);" oninput="changeRGB()">
87         <input id="slideGreen" type="range" min="0" max="1" value="0.4" step="0.01" style="accent-color:
rgb(0, 128, 0);" oninput="changeRGB()">
88         <input id="slideBlue" type="range" min="0" max="1" value="0" step="0.01" style="accent-color: rgb
(0, 0, 255);" oninput="changeRGB()">
89
90     </div>
91 <div class="row justify-content-center" style="margin: 0 !important; padding: 0 !important;">
92     <div style="font-size: 20px; margin-top: 5px;">TIME SCALE</div><br />

```

```

93     </div>
94     <div class="row justify-content-center">
95         <input id="input-timescale" type="range" min="1" max="20" value="1" step="0.1" style="accent-color:
grey;" oninput="setTimescale(event)"><br />
96     </div>
97
98 </div>
99 <script
100     src="https://code.jquery.com/jquery-3.4.1.slim.min.js"
101     integrity="sha384-J6qa4849b1E2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRJSJoZ+n"
102     crossorigin="anonymous"
103     onload="window.$ = window.jQuery = module.exports;"
104 ></script>
105 <script
106     src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
107     integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
108     crossorigin="anonymous"
109 ></script>
110 <script
111     src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"
112     integrity="sha384-wfSDF2E50Y2D1uUdj003uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6"
113     crossorigin="anonymous"
114 ></script>
115
116 <script>
117     const { ipcRenderer } = require("electron");
118     const sendIpcMsg = (msg) => {
119         ipcRenderer.send("compdev-tutorial-wom-manipulation", msg);
120     };
121
122     ipcRenderer.on('ipc-tree', (event, obj) => {
123         console.log(obj);
124         document.getElementById("myspan").innerHTML=obj;
125     });
126
127     const reset = () => {
128         sendIpcMsg({ cmd: "reset" });
129     };
130
131     const start = () => {
132         sendIpcMsg({ cmd: "start" });
133     };
134
135     const stop = () => {
136         sendIpcMsg({ cmd: "stop" });
137     };
138
139     const browse = () => {
140         sendIpcMsg({ cmd: "browse" });

```

```

141     };
142
143     const tree = () => {
144         sendIpcMsg({ cmd: "tree" });
145     };
146
147     const setTimescale = (event) => {
148         const value = parseFloat(event.target.value);
149         sendIpcMsg({ cmd: "setTimescale", value });
150     };
151
152     const setR = (event) => {
153         const value = parseFloat(event.target.value);
154         console.log(value);
155         document.getElementById("rgb").style.borderColor="rgba("+value+"100,100,0.5)";
156         document.getElementById("rgb").value=value;
157     };
158
159     function changeRGB() {
160         var col, r, g, b;
161         r = document.getElementById("slideRed").value;
162         g = document.getElementById("slideGreen").value;
163         b = document.getElementById("slideBlue").value;
164         document.getElementById('rgb').style.backgroundColor = "rgb("+r*255+", "+g*255+", "+b*255+)";
165         sendIpcMsg({ cmd: "red", r });
166         sendIpcMsg({ cmd: "green", g });
167         sendIpcMsg({ cmd: "blue", b });
168     }
169
170 </script>
171 </body>
172 </html>

```

M.6.1. kódrészlet. index.html

M.7. controls.html

```

1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8" />
5     <meta
6       name="viewport"
7       content="width=device-width, initial-scale=1, shrink-to-fit=no"
8     />
9     <link
10      rel="stylesheet"
11      href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
12      integrity="sha384-Vkoo8x4CGs03+Hhvx8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
13      crossorigin="anonymous"

```

```

14 />
15 <link rel="stylesheet" href="control.css">
16 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome
    .min.css">
17 </head>
18
19 <body>
20   <div class="container">
21     <div class="row mt-4">
22       <div class="col-1">
23       </div>
24       <div class="col-2">
25         <a href="index.html" class="back">
26           <i class="fa fa-arrow-left" style="margin-bottom: 32px;"></i>
27         </a>
28       </div>
29     <div class="row justify-content-start">
30       <div class="col-1">
31       </div>
32       <div class="col-2">
33         BROWSE:
34       </div>
35       <div class="col-8" style="text-align: justify" style="margin-top: 16px;">
36         This brings up a file explorer window, select a .gcode file to be printed. If the printer is
        running, this button will also RESET the process.
37       </div>
38     </div>
39     <div class="row justify-content-start" style="margin-top: 16px;">
40       <div class="col-1">
41       </div>
42       <div class="col-2">
43         START:
44       </div>
45       <div class="col-8" style="text-align: justify" style="margin-top: 16px;">
46         The printing process will begin, or it will continue if it was stopped by the STOP button.
47       </div>
48     </div>
49     <div class="row justify-content-start" style="margin-top: 16px;">
50       <div class="col-1">
51       </div>
52       <div class="col-2">
53         STOP:
54       </div>
55       <div class="col-8" style="text-align: justify">
56         This only pauses the process, the print stays on the bed, while the printer head stays at the
        last given coordinate. The print can be unpaused by clicking on the START button.
57       </div>
58     </div>
59     <div class="row justify-content-start" style="margin-top: 16px;">

```

```

60     <div class="col-1">
61     </div>
62     <div class="col-2">
63         RESET:
64     </div>
65     <div class="col-8" style="text-align: justify">
66         This resets the printing, this way the print is completely stopped. The print will be removed
and the printing head moves back to the start position.
67     </div>
68 </div>
69 <div class="row justify-content-start" style="margin-top: 16px;">
70     <div class="col-1">
71     </div>
72     <div class="col-2">
73         TIME SCALE:
74     </div>
75     <div class="col-8" style="text-align: justify">
76         You can set the speed of the simulation. The default value is the real speed, the maximum is 20
times faster.
77     </div>
78 </div>
79 </div>
80 <script
81     src="https://code.jquery.com/jquery-3.4.1.slim.min.js"
82     integrity="sha384-J6qa4849b1E2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRStJoZ+n"
83     crossorigin="anonymous"
84     onload="window.$ = window.jQuery = module.exports;"
85 ></script>
86 <script
87     src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
88     integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
89     crossorigin="anonymous"
90 ></script>
91 <script
92     src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"
93     integrity="sha384-wfSDF2E50Y2D1uUdj003uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl30g8ifwB6"
94     crossorigin="anonymous"
95 ></script>
96 </body>
97 </html>

```

M.7.1. kódrészlet. controls.html

M.8. tree.html

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8" />
5     <meta
6       name="viewport"
7       content="width=device-width, initial-scale=1, shrink-to-fit=no"
8     />
9     <link
10      rel="stylesheet"
11      href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
12      integrity="sha384-Vkoo8x4CGs03+Hhvx8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
13      crossorigin="anonymous"
14    />
15    <link rel="stylesheet" href="control.css">
16    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome
17      .min.css">
18    <link href="https://unpkg.com/treeflex/dist/css/treeflex.css" rel="stylesheet">
19  </head>
20  <body style="background-color: white; line-height: 1rem !important;">
21    <div class="container">
22      <div class="row mt-4" style="margin-top: 0.5rem !important; line-height: 1rem !important;">
23        <div class="col-1">
24          </div>
25        <div class="col-2">
26          <a href="index.html" class="back-dark">
27            <i class="fa fa-arrow-left" style="margin-bottom: 32px;"></i>
28          </a>
29        </div>
30        <div class="col-6 justify-content-center text-center">
31          <button
32            type="button"
33            class="btn-gray tree-button"
34            id="send"
35            onclick="tree();"
36            style="height: 30px;"
37          >
38            SHOW TREE</button>
39          <br />
40        </div>
41      </div>
42    </div>
43    <div class="row justify-content-center">
44      <div id="tree-id" class="tf-tree example" style="content-visibility: hidden;">
45        <ul id="demo">
46          <li>
```

```

47         <span class="tf-nc" id="myspan"></span>
48     <ul>
49         <li><span id="myspan2" class="tf-nc"></span></li>
50         <li><span id="myspan3" class="tf-nc"></span>
51             <ul>
52                 <li><span id="myspan4" class="tf-nc"></span></li>
53             </ul>
54         </li>
55     </ul>
56 </li>
57 </ul>
58 </div>
59 </div>
60
61 <script
62     src="https://code.jquery.com/jquery-3.4.1.slim.min.js"
63     integrity="sha384-J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n"
64     crossorigin="anonymous"
65     onload="window.$ = window.jQuery = module.exports;"
66 ></script>
67 <script
68     src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
69     integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
70     crossorigin="anonymous"
71 ></script>
72 <script
73     src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"
74     integrity="sha384-wfSDF2E50Y2D1uUdj003uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl30g8ifwB6"
75     crossorigin="anonymous"
76 ></script>
77
78 <script>
79     // this is possible because we enabled nodeIntegration
80     const { ipcRenderer } = require("electron");
81     // helper function to send IPC messages
82     const sendIpcMsg = (msg) => {
83         ipcRenderer.send("compdev-tutorial-wom-manipulation", msg);
84     };
85
86     ipcRenderer.on('ipc-tree1', (event, obj) => {
87         console.log(obj);
88         document.getElementById("myspan").innerHTML=obj;
89     });
90     ipcRenderer.on('ipc-tree2', (event, obj) => {
91         console.log(obj);
92         document.getElementById("myspan2").innerHTML=obj;
93     });
94     ipcRenderer.on('ipc-tree3', (event, obj) => {
95         console.log(obj);

```

```

96     document.getElementById("myspan3").innerHTML=obj;
97   });
98   ipcRenderer.on('ipc-tree4', (event, obj) => {
99     console.log(obj);
100    document.getElementById("myspan4").innerHTML=obj;
101  });
102
103  const tree = () => {
104    sendIpcMsg({ cmd: "tree" });
105    const treestyle=document.querySelector('#tree-id');
106    treestyle.setAttribute('style', 'content-visibility: visible !important;');
107  };
108
109  const reset = () => {
110    sendIpcMsg({ cmd: "reset" });
111  };
112
113  const start = () => {
114    sendIpcMsg({ cmd: "start" });
115  };
116
117  const stop = () => {
118    sendIpcMsg({ cmd: "stop" });
119  };
120
121  const browse = () => {
122    sendIpcMsg({ cmd: "browse" });
123  };
124
125  const setTimescale = (event) => {
126    const value = parseFloat(event.target.value);
127    sendIpcMsg({ cmd: "setTimescale", value });
128  };
129  </script>
130 </body>
131 </html>

```

M.8.1. kódrészlet. tree.html

M.9. Fan.ts

```
1 import { Node } from "maxwhere";
2 import { ITemperatureSensor } from "../TemperatureSimulator";
3 import * as THREE from "../utils/three.module";
4 import { IPrinter } from "../Printer";
5
6
7 type FanState = {
8     status: 'idle';
9 } | {
10     status: 'rotating';
11     speed: number;
12 } | {
13     status: 'powered';
14     speed: number;
15 };
16
17 const FAN_MIN_SPEED_RPM = 0;
18 const FAN_MAX_SPEED_RPM = 700;
19 const FAN_ACCELERATION = 350;
20 const FAN_DECELERATION = 100;
21
22 export class Fan {
23     state: FanState;
24     sensor: ITemperatureSensor | null;
25
26     /**
27      * @param printer Reference to the printer
28      */
29
30     constructor(private minTemp: number, private maxTemp: number, private axis: THREE.Vector3, private node
31         : Node, private printer: IPrinter) {
32         this.state = {
33             status: 'idle',
34         };
35         this.sensor = null;
36     }
37
38     attachSensor(sensor: ITemperatureSensor) {
39         this.sensor = sensor;
40     }
41
42     detachSensor() {
43         this.sensor = null;
44     }
45
46     get speed() {
47         if (this.state.status === 'idle') {
```

```

47         return 0;
48     }
49
50     return this.state.speed;
51 }
52
53 get maxSpeed() {
54     return FAN_MAX_SPEED_RPM;
55 }
56
57 stopAnimation() {
58     this.node.deanimate("orientation");
59 }
60
61 startAnimation() {
62     const cbBeginAnimCycle = () => {
63         if (this.state.status === 'idle') {
64             // Fan is idle, stop animation
65             return;
66         }
67
68         // rotations per minute, make sure it isn't zero
69         const speed = Math.max(this.state.speed, 1);
70         // quarter rotations per minute
71         const qSpeed = 4 * speed;
72         // seconds per quarter rotation
73         let duration = 60 / qSpeed;
74
75         const rotation = {
76             axis: { x: this.axis.x, y: this.axis.y, z: this.axis.z },
77             angle: 90,
78         };
79
80
81         // Halve angle and duration until duration is under 100ms.
82         // We do this to make sure that the animations don't take too long
83         while (duration > 0.1) {
84             rotation.angle /= 2;
85             duration /= 2;
86         }
87
88         this.node.animate("orientation", {
89             reference: "relative",
90             to: rotation,
91             duration: duration,
92             loop: false,
93             cubic: "linear",
94         }, cbBeginAnimCycle);
95     };

```

```

96
97     cbBeginAnimCycle();
98 }
99
100 simulate(deltaTime: number) {
101     const currentTemp = (this.sensor !== null) ? this.sensor.temperature : -273.15;
102
103     if (this.printer.isPrinting) {
104         if (this.state.status === 'idle' || this.state.status === 'rotating') {
105             // Fan is not powered; turn it on
106             this.state = {
107                 status: 'powered',
108                 speed: FAN_MAX_SPEED_RPM,
109             };
110             this.startAnimation();
111         }
112     }
113
114
115     if (currentTemp > this.maxTemp) {
116         if (this.state.status === 'idle' || this.state.status === 'rotating') {
117             // Fan is not powered; turn it on
118             this.state = {
119                 status: 'powered',
120                 speed: FAN_MIN_SPEED_RPM,
121             };
122             this.startAnimation();
123         } else if (this.state.status === 'powered') {
124             // Fan is already powered; keep powered and increase fan speed
125             // until it reaches max speed
126             this.state = {
127                 status: 'powered',
128                 speed: Math.min(this.state.speed + FAN_ACCELERATION * deltaTime, FAN_MAX_SPEED_RPM),
129             };
130         }
131     } else if (currentTemp < this.minTemp) {
132         if (this.state.status === 'powered') {
133             // Turn off the power to the fan
134             this.state = {
135                 status: 'rotating',
136                 speed: this.state.speed,
137             };
138         } else if (this.state.status === 'rotating') {
139             // Fan is not powered but still has momentum; keep decelerating it
140             this.state = {
141                 status: 'rotating',
142                 speed: Math.max(FAN_MIN_SPEED_RPM, this.state.speed - FAN_DECELERATION * deltaTime),
143             };
144         }
145     }
146 }

```

```

145         if (this.state.speed <= FAN_MIN_SPEED_RPM) {
146             // The fan has come to a stop
147             this.state = {
148                 status: 'idle',
149             };
150         }
151     }
152 }
153 }
154 }

```

M.9.1. kódrészlet. Fan.ts

M.10. TemperatureSimulator.ts

```

1  import { Fan } from "../Fan";
2  import { IPrinter } from "../Printer";
3
4  const ROOM_TEMPERATURE = 21;
5
6  export interface ITemperatureSensor {
7      get temperature(): number;
8  }
9
10 export class TemperatureSimulator implements ITemperatureSensor {
11     updateTimer: NodeJS.Timer;
12     headTemperature: number;
13
14     /**
15      *
16      * @param fan Reference to the fan
17      * @param printer Reference to the printer
18      * @param generatedHeat Temperature increase per second when the printer is
19      * operating
20      * @param lostHeat Temperate decrease per second when the printer is not
21      * operating
22      */
23     constructor(
24         private fan: Fan,
25         private printer: IPrinter,
26         private generatedHeat: number,
27         private lostHeat: number) {
28         this.headTemperature = ROOM_TEMPERATURE;
29         fan.attachSensor(this);
30         this.updateTimer = setInterval(() => {
31             this.simulate(40 / 1000);
32         }, 40);

```

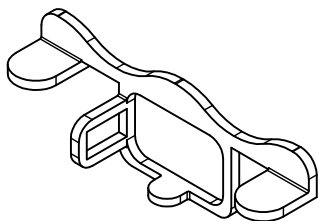
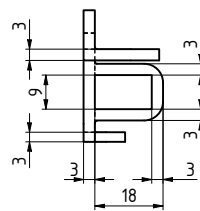
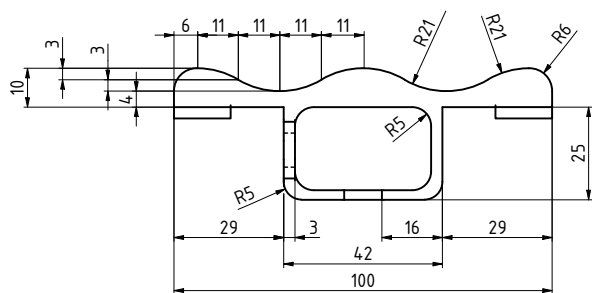
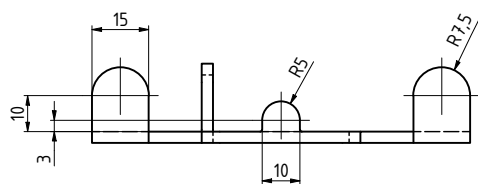
```

33     }
34
35     get temperature() {
36         return this.headTemperature;
37     }
38
39     simulate(deltaTime: number) {
40         const DELTA_TIME = 40 / 1000;
41         if (this.printer.isPrinting) {
42             this.headTemperature += this.generatedHeat * DELTA_TIME;
43         } else {
44             this.headTemperature -= this.lostHeat * DELTA_TIME;
45         }
46
47         if (this.fan.state.status === 'powered') {
48             // Fan negates temperature increase when it's at full speed
49             this.headTemperature -= this.generatedHeat * DELTA_TIME * this.fan.speed / this.fan.maxSpeed;
50         }
51
52         this.headTemperature = Math.max(ROOM_TEMPERATURE, this.headTemperature);
53
54         this.fan.simulate(deltaTime);
55     }
56
57     stopSimulation() {
58         if (this.updateTimer !== null) {
59             clearInterval(this.updateTimer);
60         }
61     }
62 }

```

M.10.1. kódrészlet. TemperatureSimulator.ts

M.11. Telefon állvány alkatrészrajza



Méretezett len lekerekítések R=2mm.

Név:	Gyártmány:		Méretarány:	BME MECHATRONIKA, OPTIKA ÉS GÉPÉSZETI INFORMATIKA TANSZÉK
Horváth Botond	Telefon állvány		1:1	
Dátum:	Megnevezés:		Vet. mód:	Rajzszám:
2023.11.04.	Alkatrészrajz			TDK-100-00
Ellenőrizte:	Anyag:	Tömeg:		Fájlnév: összeállítás2.iam
Décsei-Paróczy Annamária	PLA	5g		