Puzzle construction

Montag, 20. Juli 2020 21:51

Geogebra is a great tool to construct 3d objects because it offers several shortcuts like:

- Rotating an object around an axis
- Mirroring an object through an axis, plane or point
- Defining points as intersections of two or more objects.

Geogebra files (.ggb) are renamed .zip files.
When renamed the contents of this .zip folder are easily accessible.
The most important file is **geogebra.xml**, which stores all information about the construction of the object.

By reading this file, we can extract the points defined in geogebra and save them into python code automatically.
That way we can use a powerful, simple and free tool to create the puzzles with a nice GUI.

The points are marked with the <element> tag. More precisely:

```
<element type="point3d" label="O7">
```

The type specifies that it is a point (not a plane or circle etc.).
The label specifies the name of the object in geogebra.
This tag is closed by:

```
</element>
```

Inside this tag there are more tags, which are always closed immediately:

- <objColor> specifies the color of an object as three rgb-values in range 0-255
  ```
  <objColor r="255" g="153" b="0" alpha="0.0"/>
  ```
- <coords> specifies the coordinates
  ```
  <coords x="0.1" y="1.0" z="0.2" w="1.0"/>
  ```
  - I don't yet know whether these are always cartesian or if they can be polar or cylindrical coordinates too.
  - There is a fourth specified coordinate "w" as if geogebra could handle 4d objects
  - These coordinates are not exact! Rounding is probably recommended.
- <pointSize> specifies the size of the point.
  ```
  <pointSize val="5"/>
  ```
- <auxiliary> specifies if the object is marked as auxiliary (DE: "Hilfsobjekt"). This tag is absent for non-auxiliary objects, otherwise it has value "true".
  ```
  <auxiliary val="true"/>
  ```
- <show> specifies whether or not the object and it's label are visible in geogebra
  ```
  <show object="true" label="false" ev="4"/>
  ```

The program may offer some buttons to alter these points after reading them:

- Snap to sphere
- *Snap to tetrahedron*
- Snap to cube
- *Snap to octahedron*
- *Snap to dodecahedron*
- *Snap to icosahedron*

We can do this with the following steps:

1. Calculate the CoM of all points of the puzzle
2. Determine shape to snap to:
   a. Define shape around points:
      i. Determine the maximum coordinate of any point as sidelength of the cube
      ii. Determine maximum distance from the CoM of one point as radius of the sphere
   b. Define unit shape:

        i. Always project to the cube with sidelength 2 around the origin (0,0,0) that is aligned with the axis
        ii. Always project to the unit sphere around the origin (0,0,0)
3. Move all points by -CoM to have a shape around the origin.
4. Project all points to the defined shape by drawing a line from (0,0,0) through that point. The new point is given as the intersection of this line with the shape.
    a. Projecting to a sphere is easy:
        i. normalize the distance to the origin
        ii. Keep all signs
    b. Projecting to a cube is also easy:
        i. Normalize the distance to the origin using the maximum norm
        -> divide coordinates of each point by the maximum coordinate of that point
        ii. For this to work we must of course take the maximum of the absolute value of all the coordinates.
    c. Projecting to other platonic solids by projecting to planes
    Determining the angles in spherical coordinates may help to decide which plane to project to.