Twisty Puzzle Analysis version 2.0

by ***Sebastian Jost***

around *end of July to early August 2020*

What can it do?

This version focusses on the 3d animation using `vpython`

It also implements a console interface to control the animation. Overall it can do the following things:

- import .ggb files into vpython 3d points
- save a puzzle imported like that in a new .xml file
- visually define moves by clicking on the points
- automatically calculate the rotation axis for every move and animate it
- load a previously saved puzzle with it's moves from a .xml file

How does it do that?

The animation is completely done with the `vpython` library. The file operations are done using `xml.ElementTree` Objects. For better readability they are saved via `lxml` and not the standard `xml` module.

Move animation

The rotation axis for each move are calculated every time the move is performed. This is done by first determining the center of mass of all points in the given cycle.

Given that `CoM` and a cycle `(a,b,...)` we consider the vectors `CoM-A` and `CoM-B`, where `A` and `B` are the 3d points corresponding to the indeces `a` and `b`. The Axis of rotation is then given by the cross product of these two vectors: `rot_axis = (CoM-A) x (CoM-B)`.

However this does not work if there are only two points in the cycle as in that case the two vectors `CoM-A` and `CoM-B` are collinear. (vpython provides a method to check this)

In that case the rotation axis is calculated as the vector `CoM - Puzzle_CoM` where `Puzzle_CoM` is the center of mass of all points in the puzzle. This rotation axis means that the points in a 2-cycle are always rotated by 180° around their centerpoint.

Even this does not cover all possible cases. There are still possible issues with this system:

Issues

1. Consider a 2-cycle with center of mass `CoM = Puzzle_CoM`. Then the rotation axis described before is not clearly defined. As a real-world example where this happens, consider a 'slice move' on a floppy cube. The same issue may occur with other cuboids.

   It seems like there is no obvious or easy solution to this problem. However it can be solved by defining **one** rotation axis for the entire move. This is tricky with geared puzzles though as there one rotation

axis is *not* sufficient to describe all the movement in one move.

2. The puzzle information was stored in a difficult to understand dictionary with lots of values of different types. Additionally there was no clear separation between animation control and console inputs and outputs in the code. They got intertwined as the project progressed.

3. The .ggb file import still leaves behind ugly unnecessary files: the `.ggb` file renamed to `.zip` (geogebra files are just renamed zips) and the definition file `geogebra.xml`. Both of these are probably not needed or should at least be deleted after the import is done.

Dependecies

non-standard library modules:

- vpython
- colored
- lxml

Conclusion

The basic elements used for the file import, export and the animation are sufficiently seperated into files and will most likely be reused for future versions.

Also the interaction methods for the different console commands are a nice baseline and establish what basic functions are required.