Synthetic Geometry Toolbox | Group H02:

- CHENG, Tien-chun | 20537176
- PACHECO RODRIGUEZ, Wemp Santiago | 20543280
- PATUPAT, Albert John Lalim | 20544416

Project Background and Description

Synthetic geometry is the field of mathematics dealing with geometry without the use of coordinates or formulae. Mathematicians and computer scientists have been fascinated with the study of straightedge and compass constructions due to their relation with algebraic number fields and their diverse applications e.g. Geometric Cryptography. We will create an environment for classical constructions (see reference[1]), to allow user to experience and enjoy this important field of mathematics.

Expected End Results of the Project

At the end of this project, the team will have programmed a graphing software with the following features:

1. Imitates a compass and a straightedge, and automatically constructs more advanced objects based on these classical tools, such as midpoints, perpendicular lines, parallel lines, and circles passing through three selected points.
2. Supports addition, mutation, and deletion of points, which leads to adjustment of already constructed geometric objects such as lines, circles, and triangles.
3. Visualizes the graphs of user-defined geometric constructions.
4. Solves problems in Euclidea (see reference [2]), a game with tasks of constructing target geometric objects.
5. Supports construction of important triangle centers (see references [3],[4]).

OOP Technologies, Techniques, and Data Structure

- Inheritance: Create a geometry node base type, and all types of components (circles, lines, points) inherit from it.
- Virtual Functions: Capacity of resolution of the functions print(), print_GUI(), access() and destructor to the correct type within the main structure (Vector).
- Directed Acyclic Graph (DAG): for components dependencies, updates, and deletes
- Dynamic Resizable Array (STL vector): for storing geometric component nodes
- Map (STL map or unordered_map): for converting users specified labeling to identifier
- **External Libraries: iostream, cassert, vector, map, Qt**

References

[1] https://www.geogebra.org/geometry
[2] https://www.euclidea.xyz
[3] https://faculty.evansville.edu/ck6/encyclopedia/ETC.html
[4] http://mathworld.wolfram.com/KimberlingCenter.html

APPENDIX A

Sketch of Classes

1. GeoComponents
    a. Vector <GeoNode*> geo_components;
    b. void add_construction(<formal parameters>);
    c. void edit_construction(<formal parameters>);
    d. void remove_construction(unsigned int PID);
    e. void print_all_constructions();
    f. void print_all_GUI();
2. GeoNode
    a. unsigned int pid;
    b. bool user_defined;
    c. bool hidden;
    d. string label;
    e. const int num_dependances;
    f. const Geonode**;
    g. virtual void print_GUI() const override;
    h. virtual void print() const override;
    i. virtual void access(<formal parameters>) const override;
    j. virtual void mutate(<formal parameters>) const override;
    k. virtual ~GeoNode();
3. PointNode : public GeoNode
    a. double x, y;
    b. void (*construct)();
    c. virtual void print();
    d. virtual void print_GUI();
    e. virtual void access(<formal parameters>);
    f. virtual void mutate(<formal parameters>);
    g. several constructors
    h. several mutators
4. LineNode : public GeoNode
    a. double x_coeff, y_coeff, c_coeff;
    b. void (*construct)();
    c. virtual void print();
    d. virtual void print_GUI();
    e. virtual void access(<formal parameters>);
    f. virtual void mutate(<formal parameters>);
    g. several constructors
    h. several mutators
5. CircleNode : public GeoNode
    a. double center_x, center_y, radius;
    b. void (*construct)();
    c. virtual void print();
    d. virtual void print_GUI();
    e. virtual void access(<formal parameters>);
    f. virtual void mutate(<formal parameters>);
    g. several constructors
    h. several mutators
6. TriangleNode : public GeoNode
    a. double side_a, side_b, side_c;
    b. void (*construct)();
    c. virtual void print();
    d. virtual void print_GUI();
    e. virtual void access(<formal parameters>);
    f. virtual void mutate(<formal parameters>);
    g. several constructors
    h. several mutators