gTangle Documentation

1. General

Team members: Jyoti Sunkara, 2018101044 Aryamaan Jain, 2019121002 Ammar Ahmed, 2018101058 Aman Goel, 2018101005

Team number: 19

Team name: Lab Team

Project id: 16

TA assigned: Prajwal Krishna

Presentation date: 20 Nov 11:40-11:55

Link to github repo: https://github.com/Digital-Image-Processing-IIITH/project-lab-team

Link to paper: http://pellacini.di.uniroma1.it/publications/gtangle16/gtangle16-paper.pdf

To run code:
1. Change directory to code.
2. Run with command make all
3. Run ./tangles

Additional dependencies: This was developed in linux. Not tested for others.

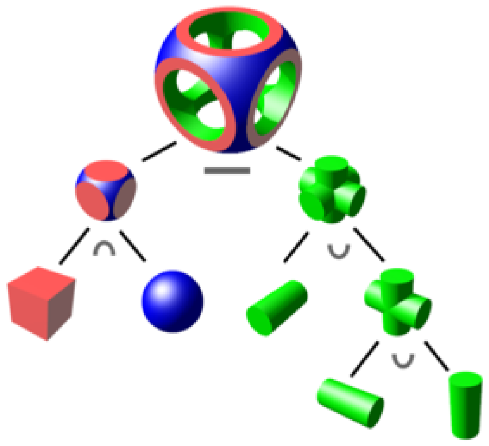
2. General concepts used in code

2.1. CSG tree

Constructive solid geometry (CSG; formerly called computational binary solid geometry) is a technique used in solid modeling. Constructive solid geometry allows a modeler to create a complex surface or object by using Boolean operators to combine simpler objects, potentially generating visually complex objects by combining a few primitive ones.

In 3D computer graphics and CAD, CSG is often used in procedural modeling. CSG can also be performed on polygonal meshes, and may or may not be procedural and/or parametric.

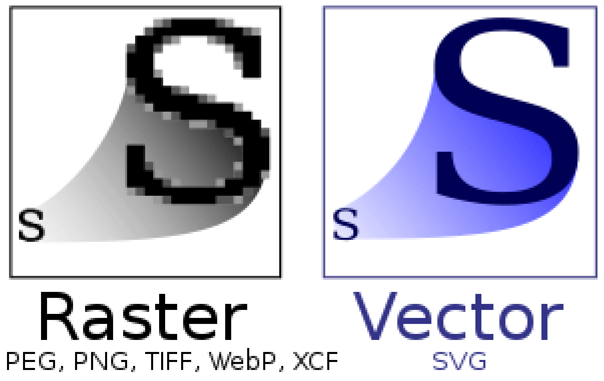
Contrast CSG with polygon mesh modeling and box modeling.



2.2. SVG

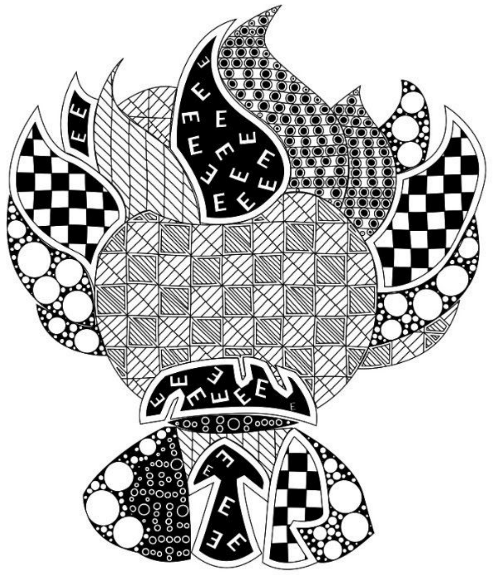
Scalable Vector Graphics (SVG) is an Extensible Markup Language (XML)-based vector image format for two-dimensional graphics with support for interactivity and animation. The SVG specification is an open standard developed by the World Wide Web Consortium (W3C) since 1999.

SVG images and their behaviors are defined in XML text files. This means that they can be searched, indexed, scripted, and compressed. As XML files, SVG images can be created and edited with any text editor, as well as with drawing software.



2.3. Tangles

Tangles are a form of structured pen-and-ink 2D art characterized by repeating, recursive patterns. This paper contains methods to procedurally generate tangle drawings, seen as recursively split sets of arbitrary 2D polygons with holes, with anisotropic and non-stationary features.



2.4. Operators

1. grouping operators, that don't actually modify any shape, but only re-label sets of shapes by manipulating their group and shape ids
2. geometric operators, that subdivide shape groups by splitting, outlining and placing objects in them
3. decorative operators that simply modify the final appearance of a shape, without further subdivisions. This paper supports two decorative operators:
    a. filling that sets the background color of a shape
    b. stippling that stipples the shape with geometrical details like dots, dashes, curves.

2.5. Grammar

This paper uses group grammars, derived from set grammars, to automatically generate tangles from any valid starting arrangement of shapes. The core idea of our formalization is that all the grammar operators work on groups of shapes, both in input and output, subdividing the plane in new groups, or re-labeling the shapes in meaningful ways. A shape is uniquely identified as S = <t, g, s, Θ, d>

3. Files

3.1. gTangle Folder

- animator.cpp: file used to give results when code runs in animator mode.
- csg_tree.h: header file for handling csg tree operations
- input_figures.cpp: used for taking input basic shapes or loading svg file
- tangle_utilities.h: utility file for handling tangles
- animator.h: header file for animator.cpp
- input_figures.h: header file for input_figure.cpp
- rule.cpp: used to handle grammar rules.
- draw.cpp: used to draw basic shapes and primitives.
- rule.h: header file for rule.cpp
- threadpool.h: header file for threadpool.cpp
- animator_matrix.cpp: used for basic transformation in animator mode
- draw.h: header file for draw.cpp
- main.cpp: one of the main program entry point
- time_manager.cpp: used for handling timeline in animator mode.
- animator_matrix.h: header file for animator_matrix.h
- shape.cpp: file for handling shapes
- time_manager.h: header file for time_manager.cpp
- expansion_manager.cpp: used to handle expansion steps
- main_tag_svg.cpp: used for tagging svg in gui
- shape.h: header file for shape.cpp
- clipper_methods.cpp: used for clipping out polygons
- expansion_manager.h: header file for expansion_manager.cpp
- ui.h: header file for ui.cpp
- clipper_methods.h: header file for clipper_methods.cpp
- main_timeline.cpp: another main file for handling timeline based execution
- svg.cpp: file for handling svg based operations
- yocto_math.h: basic math library
- grammar_core.cpp: file for parsing grammar
- svg.h: header file for svg.cpp
- common.h: common header file for storing basic program constants
- grammar_core.h: header file for grammar_core.cpp
- operator.cpp: file for handling operators used to produce tangles
- csg_tree.cpp: file for handling csg tree operations
- operator.h: header file for operator.cpp
- tangle_utilities.cpp: contains basic tangle utility functions

3.2. grammars folder

Contains grammar used for producing various outputs.

3.3. resources folder

Contains images, fonts and svg’s used by the author.

4. Challenges faced

- Initially this project was done in python but due to the heavy dependencies on external libraries, we had to shift to cpp. The python code is available in the python folder.
- The UI for this paper was written for mac os using xcode. We therefore were not able to port the code directly and not provide a UI.