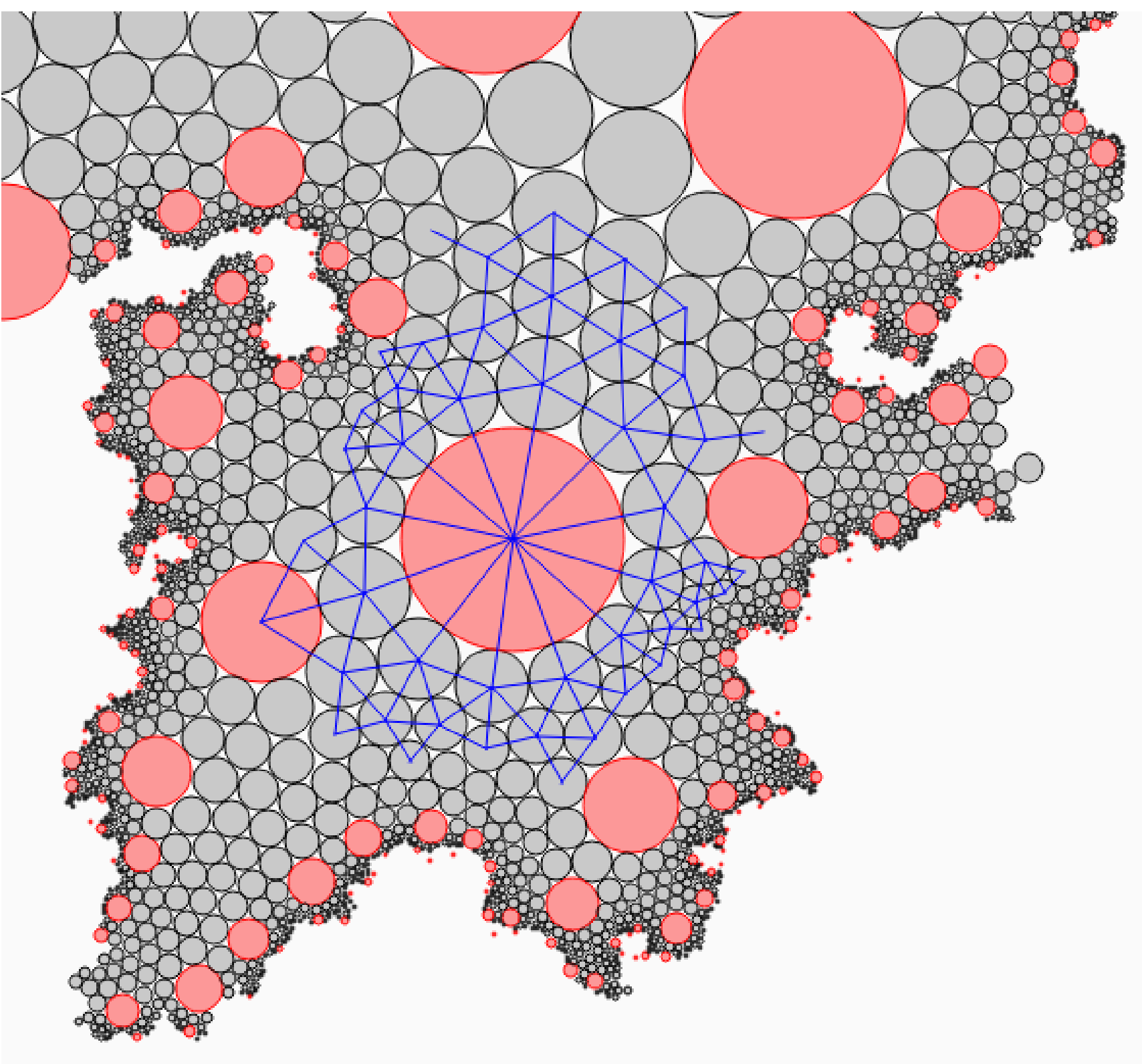


**Overview**

The goal of our project was to visualize and explore circle packing projective structures by building an interactive GUI python program. Our faculty supervisor, Prof. David Dumas, and our graduate mentor, Ellie Dannenberg, assembled a robust circle packing platform before the semester of which we utilized greatly. The mathematical essence and the starting point for our research originated from a 2003 paper by Kojima, Mizushima, and Tan.

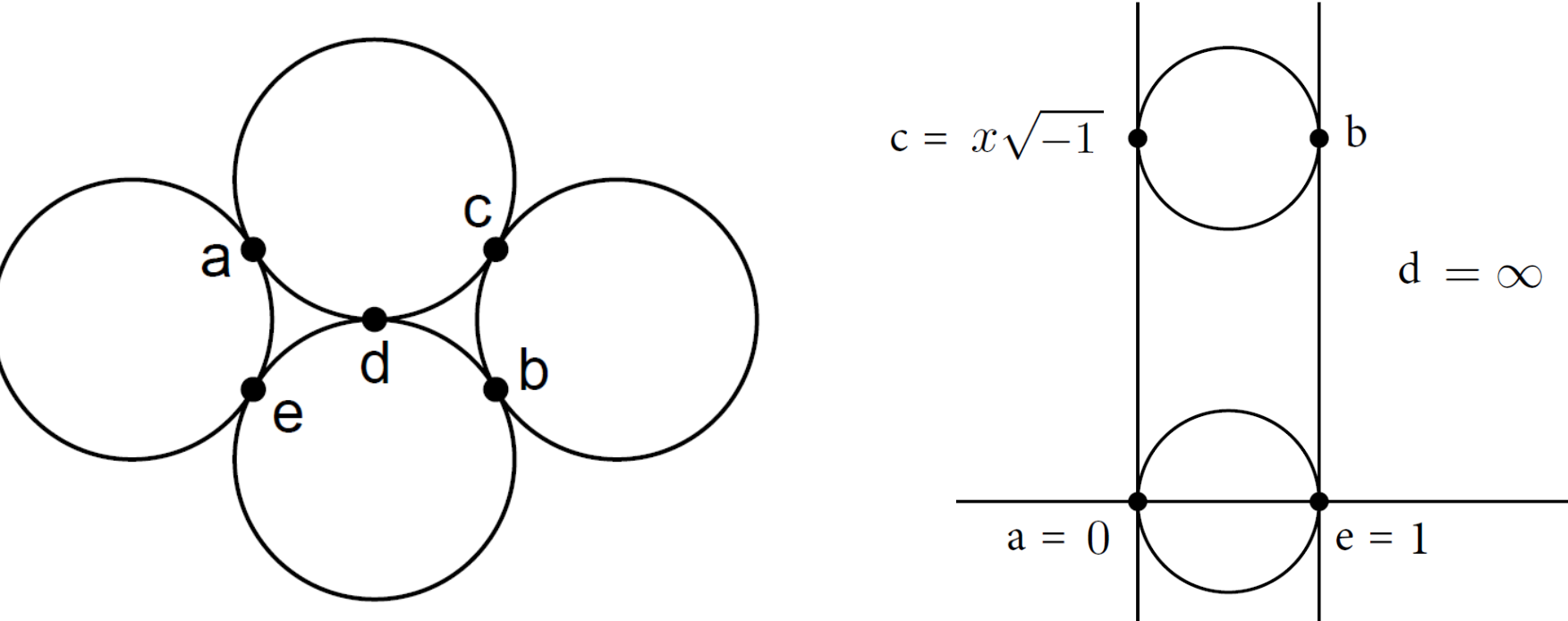
**Circle Packings**

There are several notions of circle packing in common use. In this project, our convention is that a circle packing is an arrangement of circles in the plane with tangencies but no overlaps, and where the gaps in between the circles are curvilinear triangles.



**The Cross Ratio Parameter**

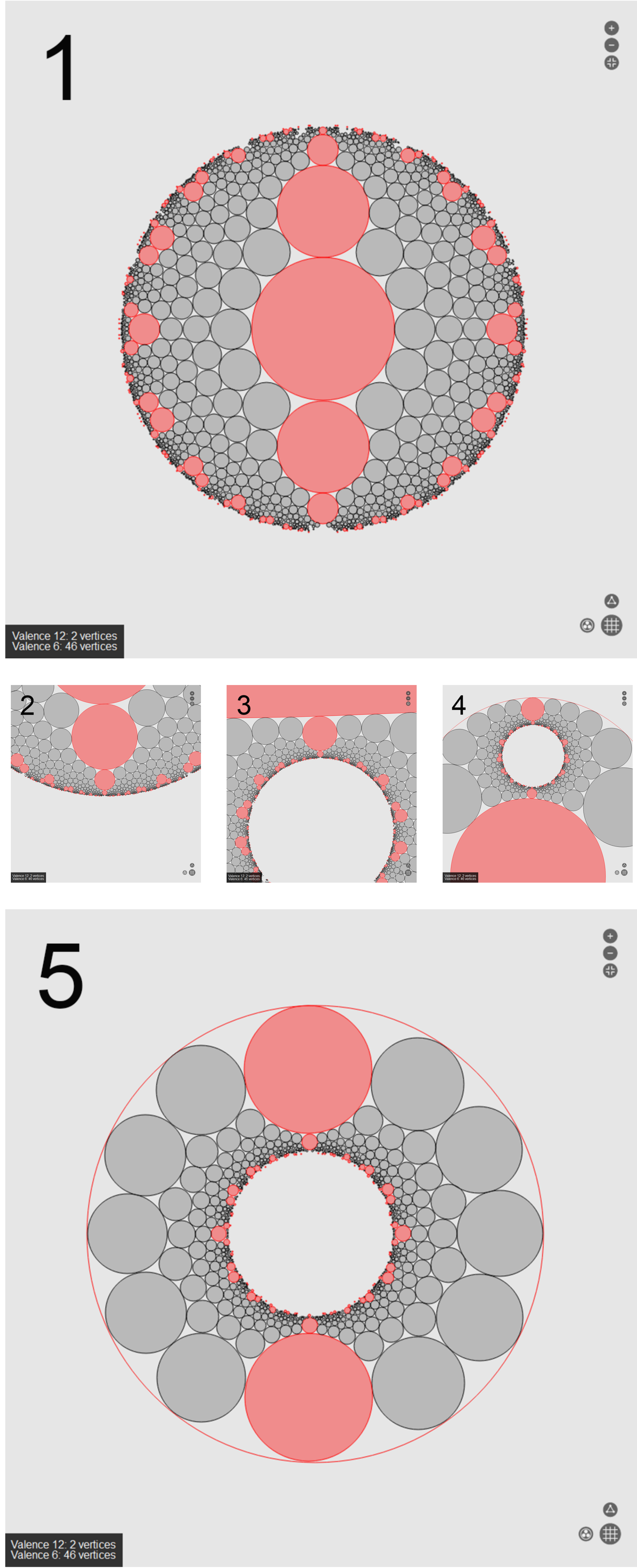
In a circle packings dual graph, there is a real number associated with each edge. This number is computed as the cross ratio of four tangency points associated to that edge, hence it is called the cross ratio invariant of the edge.



With the tangency points labeled as above, the cross ratio invariant is equal to  $\chi(a, b, c, d) = \frac{(a-c)(b-d)}{(a-d)(b-c)}$ . It can also be computed by normalizing the figure to the right, where the cross ratio invariant is  $x$ . We normalize the 4 local tangent circles by taking  $a$  to 0,  $e$  to 1, and  $d$  to  $\infty$ .

**Equivalence of Circle Packings**

The following sequence of images are of a circle packing becoming inverted by Möbius transformations. Circle packings are **equivalent** if they are related by a Möbius transformation. For example, all of the images below are equivalent to one another.



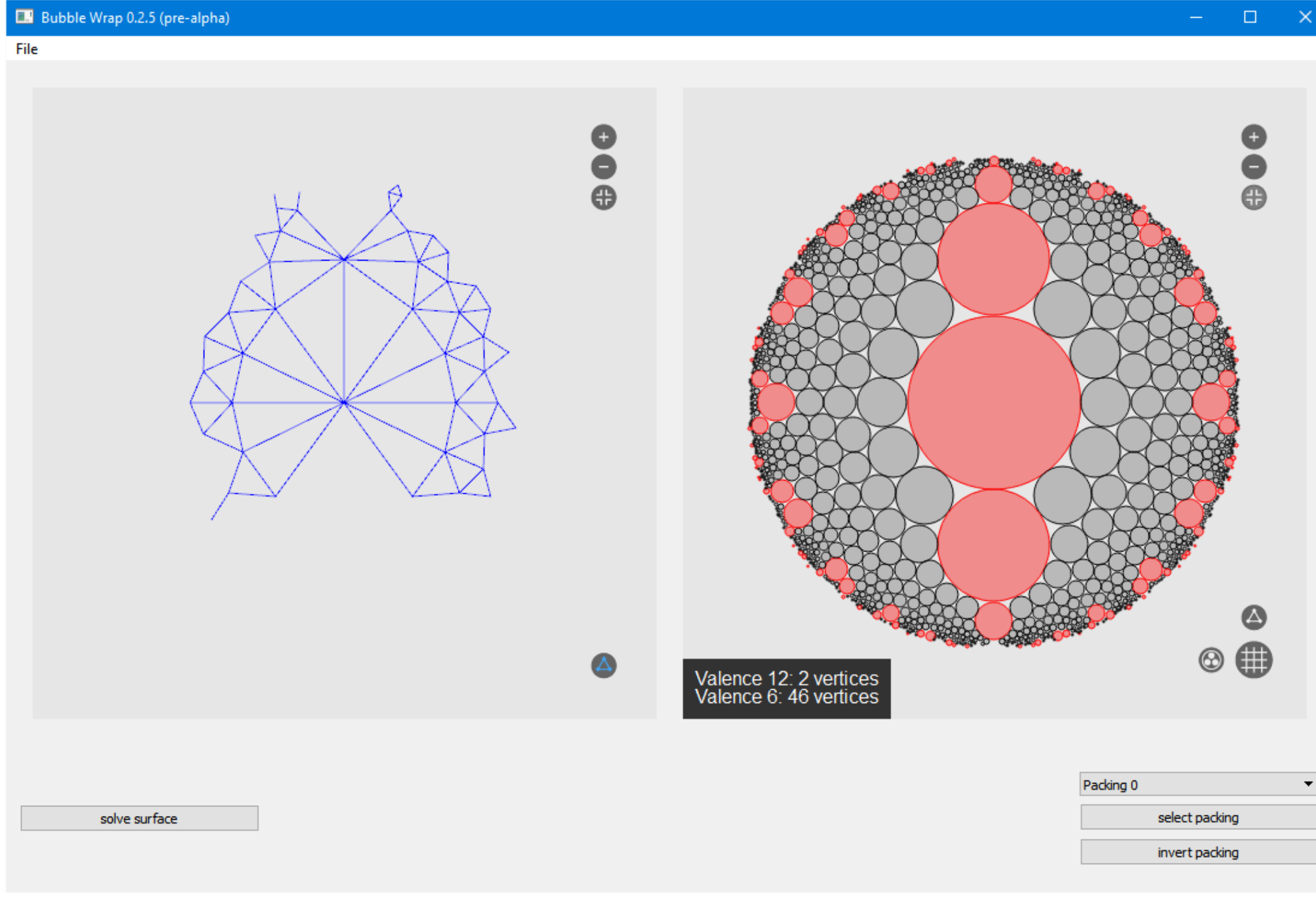
Valence 12, 2 vertices  
 Valence 6, 46 vertices

**Motivation to Build an Interactive Application**

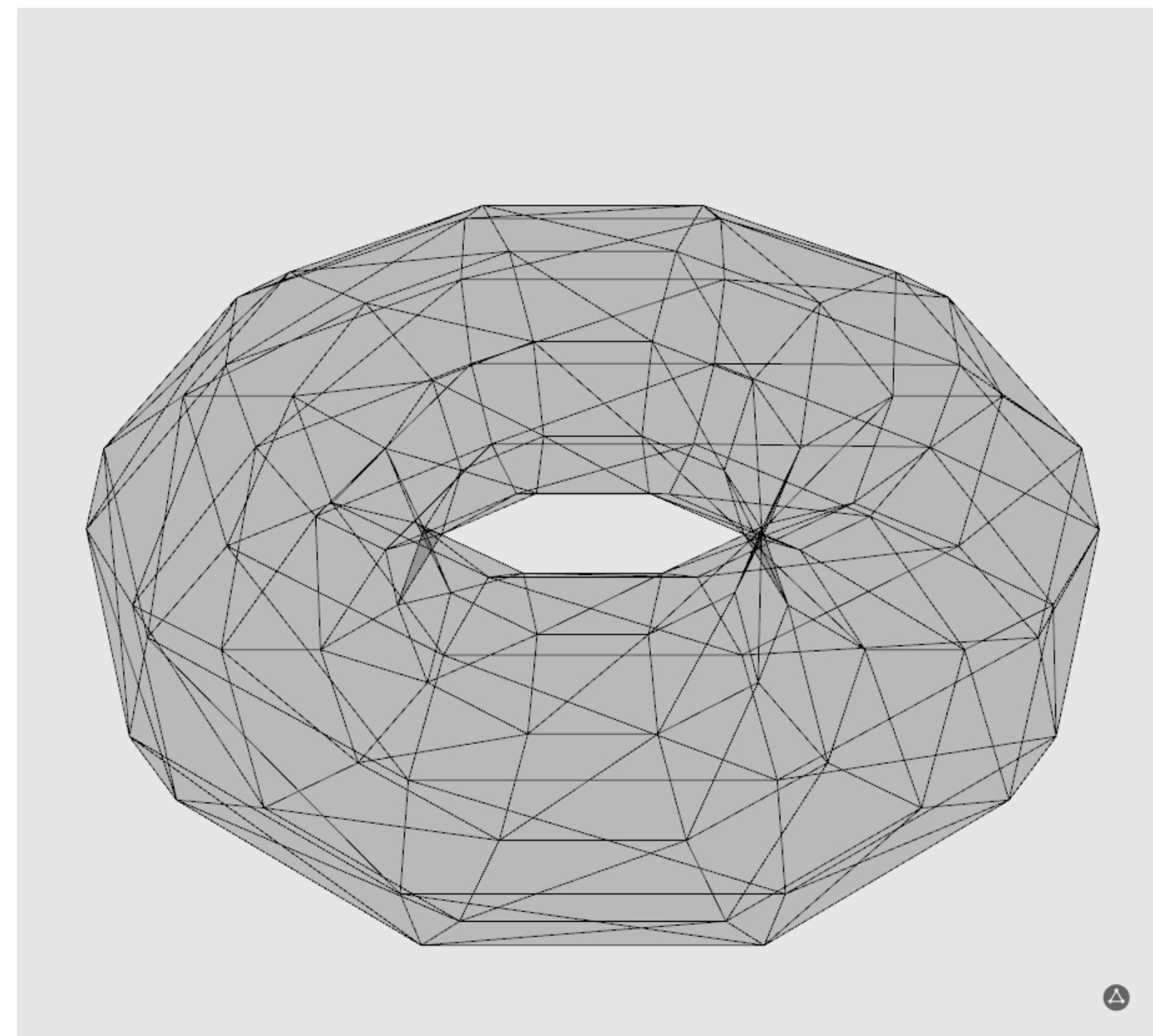
The moduli spaces of circle packing projective structures have interesting geometric properties. Circle Packings can be reshaped through Möbius transformations while still retaining the original cross ratios (a parameter used to create a circle packing.) This research focused on creating tools for building and exploring the moduli space of circle packing projective structures with a given dual graph on a surface.

**The Application**

We named the application **Bubble Wrap** because it involves wrapping circles ("bubbles") around a surface. The picture below is a preview of the application's UI.



The left display shows the dual graph of the circle packing, cut open and flattened on the plane. The right display shows the full packing. The packing shown above is of a genus 2 surface. If a circle packing file contained a 3D embedding, the left view could also be used to display the object in 3D space. Below is an example of a genus 1 surface rendered on the left display.



**Optimization**

The UI needed to maintain a level of responsiveness while computing complex Möbius transformations for each circle on the screen. Since python can only run on a single core, we quickly ran into a barrier.

Our fix was to have two separate lists of circle objects. The first list would contain all the original circles. The second list contained only circles visible to the user in its current state. Once the user modified the circle packing in any way, a second thread would start a calculation in the background to find the appropriate circles to display in its new state. Below is the code used to initiate the new thread.

```


if self.force_update:
    self.force_update = False
    # If so, optimize circles

# cancel old thread
if self.optimize_thread is not None and self.optimize_thread.isRunning():
    self.optimize_thread.cancel = True

# begin new optimization thread
self.optimize_thread = tools.OptimizeCirclesThread(self, self.uecp.circles, self.uecp.circles_optimize,
                                                    self.uecp.packing_trans[0], self.display_params)
self.optimize_thread.start(priority=QThread.LowestPriority)
    
```

**Custom GUI Elements**

Since PyQt5 does not support GUI elements on top of a canvas element, we created our own. We created simple python objects to hold the size, image, and position of each button. When the canvas is drawing the circles it also draws the buttons on top. We also designed our own buttons using Adobe Illustrator. Below are the graphics used.



**References**

S. Kojima, S. Mizushima, and S.P. Tan. *Circle Packings on Surfaces With Projective Structures*. J. Differential Geometry **63**(2003), 349–397.

M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. 3rd Ed. (2008) Springer.