Circle Packing Visualization

Kimberly Kim & Jacob Lewis

August 5th, 2016

Abstract

This is the final report of a summer 2016 project of the Mathematical Computing Laboratory. Alongside us was our faculty supervisor, Prof. David Dumas, and our graduate mentor, Ellie Dannenberg. The goal of our project was to visualize circle packing projective structures and explore their moduli spaces by building an interactive GUI python program.

A Brief Introduction to Circle Packing:

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.

We referenced the paper by Kojima, Mizushima and Tan heavily throughout our study, and they said, "Roughly, a circle packing on a closed orientable surface Σg of genus g, is a collection of closed disks on the surface such that the interiors of any two distinct disks are disjoint and the collection of interstices consists of a finite number of triangular interstices each bounded by three circular arcs" (Kojima, Mizushima & Tan, 356). And "for a circle packing on Σg, we assign a vertex to each circle and an edge joining two vertices for each tangency point between two circles. The graph τ on Σg obtained thus triangulates Σg and this is called the nerve, or the dual graph, of the circle packing" (Kojima, Mizushima & Tan, 356).

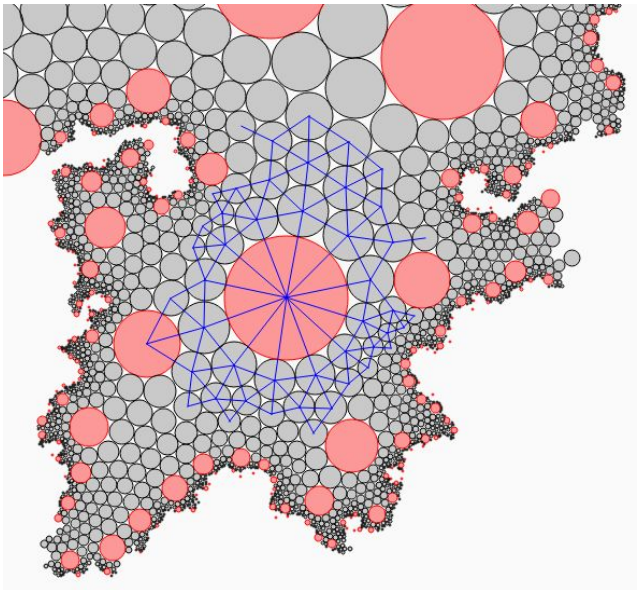

Figure 1: The dual graph is in the blue, and the red circles show the circles with 6 valence (i.e. tangent to six other circles)

Möbius Equivalence:
Two circle packings are equivalent if they are related by a möbius equivalence, or linear fractional transformation. The following sequence of images are of a circle packing becoming inverted by möbius transformations, therefore, each picture shows circle packings that are essentially "the same."

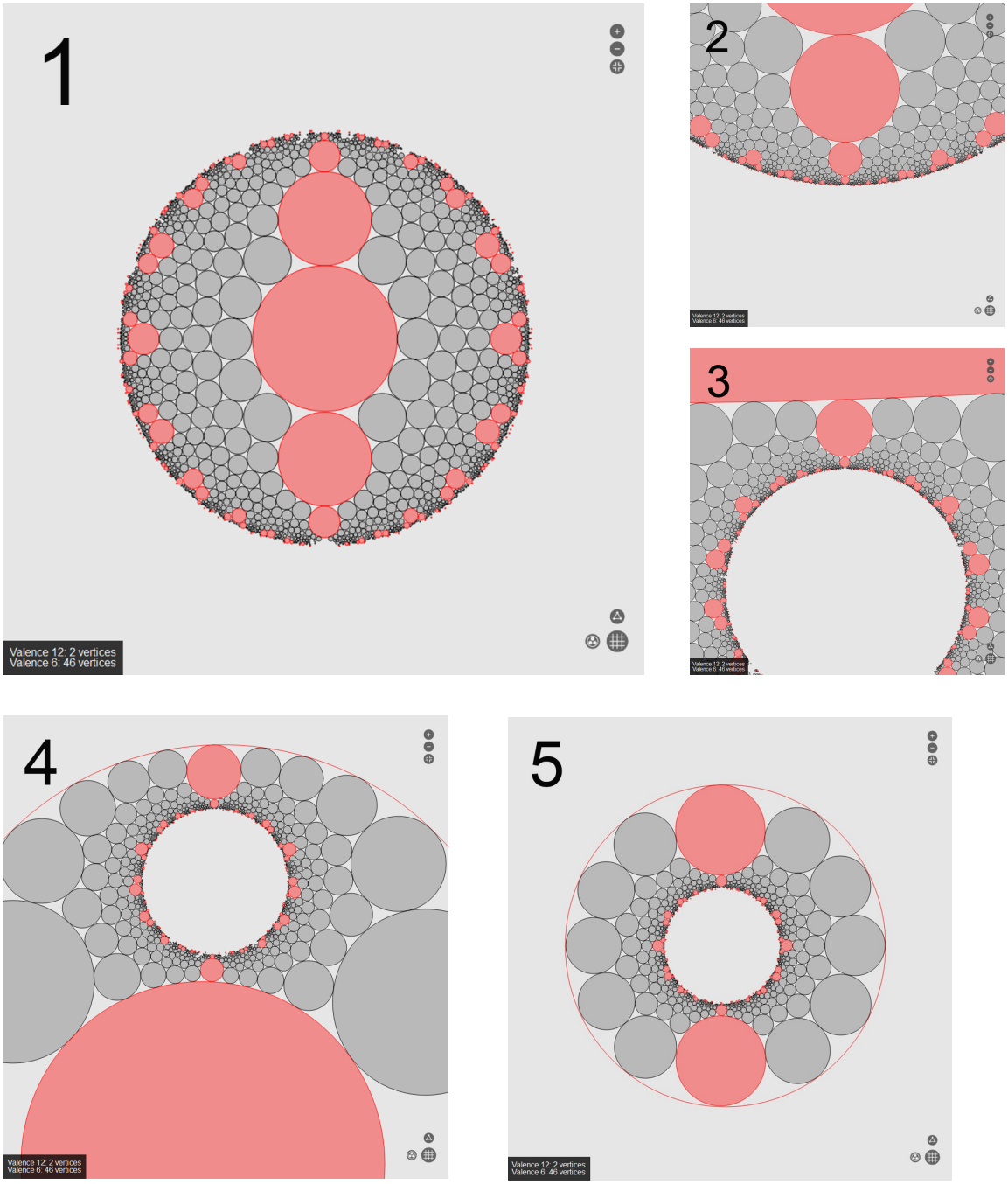

Figure 2: Equivalent circle packing related by möbius transformations. All of the images above are screenshots from the application.

The Cross Ratio Parameter:
In a circle packing's dual graph, there is a real number, a cross ratio invariant, associated with each edge. So "knowing the developing image of one of the interstices determines the developing image of the other interstices" (Kojima, Mizushima & Tan, 356). We call $\chi(a,b,c,d)$ the cross ratio of the four complex numbers $a,b,c,$ and d from the four circles as shown in the figure below. The cross ratio invariant of an edge E is defined to be $\chi(a,b,c,d)$. The four complex numbers ($a,b,c,$ and d) are the tangency points between circles as shown in Figure 3 (left). Then we normalize the 4 local tangent circles by taking a to zero, e to one, and d to infinity, as shown in Figure 3 (right). And $\chi(a,b,c,d)$ produces x , the cross ratio invariant or edge E.

$$\chi(a,b,c,d) = (a-c)(b-d) / (a-d)(b-c)$$

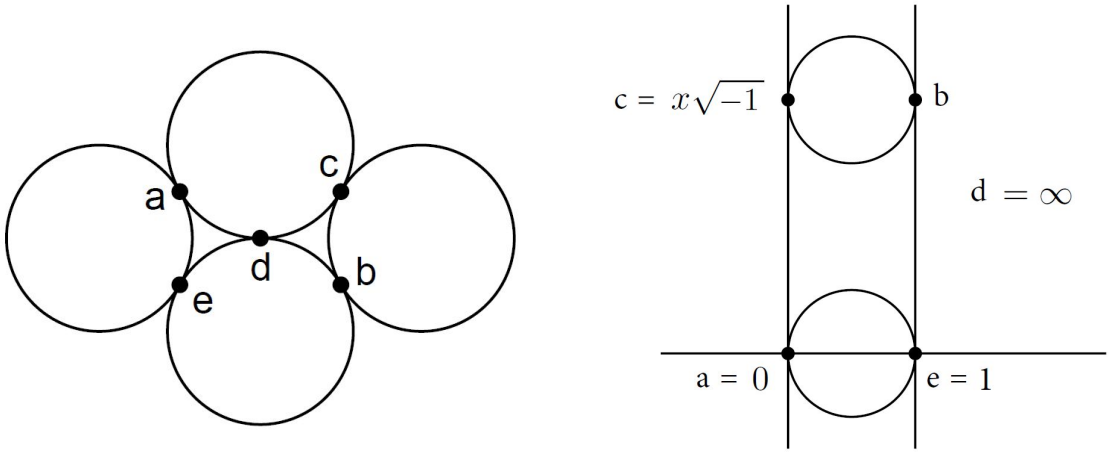

Figure 3: Configuration of circles corresponding to an edge of the dual graph: In the packing (left), and normalized (right)

Motivation:
We consider the moduli space of a circle packing projective structure to be the set of equivalence classes of circle packings with a given dual graph, where equivalence refers to the action of möbius transformations described above. This moduli space has interesting geometric properties. Circle Packings can be reshaped through möbius transformations while still retaining their original cross ratios. This project focused on creating tools for constructing circle packings and exploring the moduli space of circle packing projective structures on a surface.

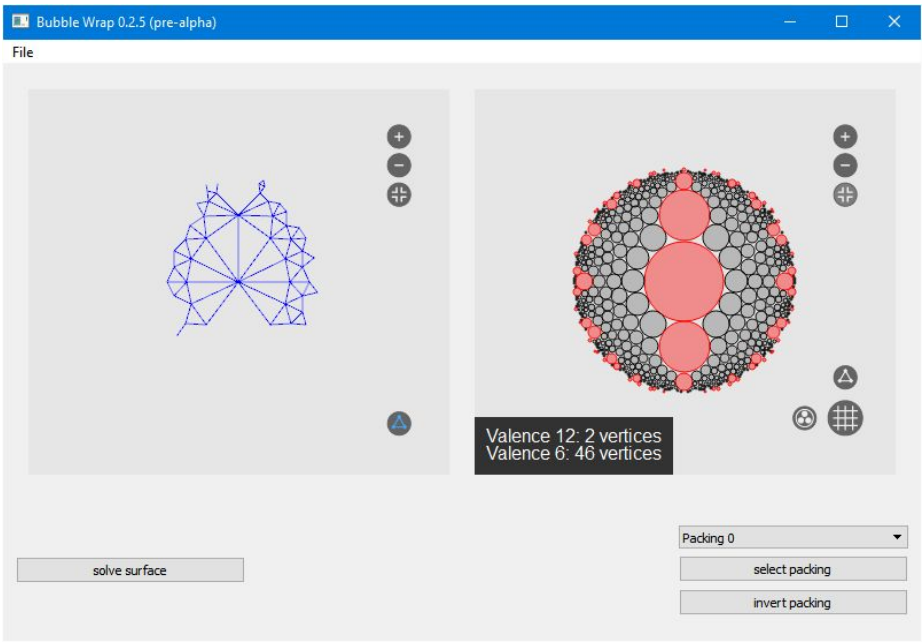

Figure 4: Screenshot of the app with a Fuchsian circle packing

Approach:
We wanted to both visualize the 3 dimensional surface as well as show the circle packing that it produced. We used a data structure called a doubly connected edge list (DCEL), which keeps a record of each face, edge and vertex of surface, to represent a dual-graph of a circle packing (We referenced de Berg-Cheong-von Kreveld-Overmars text for the structure of DCEL). The DCEL consists of vertex, edge, and face objects linked to one another by pointers (references). We were given a DCEL implementation in python that did not store coordinates in 3-space. We adapted this implementation by attaching coordinates to each vertex object and included support for saving this information to a JSON file. In addition, we created parameterized coordinate generators. We created two generators for both the cylinder and the torus. The coordinate generators take 4 parameters as input: two static scalar values and two variables that varied between 0 and 1. The two variables determined where the coordinates should be positioned and scaled. A typical invocation of our CoordinateVertex class to generate part of a right circular cylinder is as follows:

```
def circ_cylinder_param(h, r, i, j):
    ang = 2*math.pi*i
    x, z = r*math.cos(ang), r*math.sin(ang)
    y = h*j
    return CoordinateVertex(coords=(x, y, z))
```

Interpreting a Circle Packing:
A circle packing is stored in a JSON based file, and it is created with the information to reproduce the DCEL object in memory. It also contains cross ratio information generated while solving for a circle packing. We represent a circle as an anti-möbius map, which is stored as a 2x2 matrix. Then applying möbius transformations to the circles becomes a string of consecutive 2x2 matrix compositions.

Optimization:
The program that we wrote needed to maintain a level of responsiveness while computing complex möbius transformations for each circle on the screen. For a typical genus 2 circle packing of approximately 50 vertices, about 20,000 circles are loaded into memory. This means that 20,000 möbius transformation would need to be calculated every frame. To keep the application responsive, we strived to keep the frame-rate above 15fps. Which meant that the computer would need to effectively compute at least 300,000 möbius transformation every second. Since python can only run on a single core, we ran into a barrier.

Our solution to keep the application responsive was to have two separate lists of circle objects. The first list would contain all 20,000 original circles. The second list contained only circles visible to the user in its current state. If 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 and update the second list. This reduced 20,000 circles down to around 700 circles (on average). It is much more reasonable to meet 15fps with 700 circles with python. In the future, we hope to eliminate as many of the performance restrictive python calls as possible. If we were to use numpy purely, we could probably raise the number of visible circles above 300,000 while maintaining at least 15fps.

Responsive Mouse-driven Möbius transformations:
Applying a möbius transformation to a circle packing does not change any of the cross-ratios, so the identity of the circle packing remains the same. One way we allow the user to apply a möbius transformation to the picture is to allow the placement of two fixed points, and let the user select the image of a third point by clicking and dragging with the mouse. This makes seeing different transformations of the "same" circle packing possible.

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.