University of Colorado Boulder

Undergraduate Honors Thesis

Congruence Subarrangements of the Schmidt Arrangement

Evan Oliver

Defense Committee:

Dr. Katherine E. Stange, Department of Mathematics, Supervisor

Dr. Nathaniel Thiem, Department of Mathematics

Dr. Mark J. Ablowitz, Department of Applied Mathematics

April 11, 2016

1 Introduction

We begin with background in algebraic number theory, specifically studying quadratic fields K and rings of integers inside those fields $\mathcal{O}_K$. From there, we study properties of the matrix group $PSL_2(\mathcal{O}_K)$ and its congruence subgroup $PSL_2(\mathfrak{p})$ for some prime ideal $\mathfrak{p}$. The Schmidt arrangement is the orbit of the real line under Möbius transformations described by the matrices in $PSL_2(\mathcal{O}_K)$. We then examine how these arrangements are affected when the transformations are limited to the matrices in $PSL_2(\mathfrak{p})$; this forms what we will call a congruence-$\mathfrak{p}$ subarrangement of the full arrangement. We prove some tangency properties of the congruence subarrangement and finally conjecture that the congruence-(2) subarrangement of the Schmidt arrangement of the Eisenstein integers has an Apollonian circle packing structure.

2 Quadratic Number Fields

Algebraic number theory is the study of algebraic integers and the number fields they inhabit. The number fields we will be studying are called quadratic fields, and here we will investigate how to determine the integers inside a quadratic field.

Definition 2.1. *The **algebraic integers** are defined as complex roots of monic polynomials with integer coefficients. In other words, they are elements of the set* $\{x \in \mathbb{C} \,|\, x^n + a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_1 x + a_0 = 0$, *for some* $n \in \mathbb{Z}^+$, $a_j \in \mathbb{Z}$ *for* $0 \leq j \leq n-1\}$.

This set in its entirety is difficult to study, so by imposing restrictions on these polynomials we get a more manageable subset of algebraic integers which themselves have interesting properties. In our case we restrict these polynomials to have degree 2.

Definition 2.2. *Let L be a field. By adjoining an algebraic integer ξ to L, we construct a field that contains ξ as an element. We call this **field extension** $L(\xi)$ and its elements are all numbers of the form* $\{\sum_{i=0}^{n} a_i \xi^i \,|\, a_i \in L, n \in \mathbb{N}\}$.

A field extension can be thought of as a vector space spanned by $\{1_L, \xi, \xi^2, \ldots\}$ with elements of L as scalars. The dimension of such a vector space is called the degree of the extension. If $\xi \in L$, the degree of the extension is 1 and we say the extension is trivial. If the extension has degree 2, we call it a quadratic extension. It is possible for the degree of an extension to be higher than 2, but for our intents and purposes we will focus on quadratic extensions and all of the following field extensions in this paper will be quadratic extensions.

By extending the rational numbers to contain roots of second-degree polynomials, we get another field called a quadratic field. Let $p(x) = x^2 + Bx + C$ be a monic, second-degree polynomial where $B, C \in \mathbb{Z}$. By the quadratic formula, solutions of $p(x) = 0$ are given by $x = \dfrac{-B \pm \sqrt{B^2 - 4C}}{2}$. Since $B^2 - 4C \in \mathbb{Z}$ we can rewrite it as $B^2 - 4C = \Delta A^2$ where $\Delta, A \in \mathbb{Z}$, $A \geq 1$, and Δ is square-free (that is, not a multiple of

any squared integer greater than 1). Then the roots of $p(x)$ are given by $x = \frac{-B \pm \sqrt{\Delta A^2}}{2} = \frac{-B \pm A\sqrt{\Delta}}{2}$. By extending $\mathbb{Q}$ to $\mathbb{Q}(\sqrt{\Delta})$ we now have a field containing these roots to our quadratic equation $p(x) = 0$, thus the term "quadratic field". If $\Delta = 1$, then we have a trivial field extension, so henceforth let's assume that $\Delta \neq 1$. Also, we can express the discriminant D of $\mathbb{Q}(\sqrt{\Delta})$ as $B^2 - 4C$ in terms of the integers B and C in the polynomial $p(x)$ mentioned above.

Inside a quadratic field, there exists a ring of elements that have structure that is similar to the integers inside of the rationals.

Theorem 2.1. *Given a quadratic field $\mathbb{Q}(\sqrt{\Delta})$ for a square-free integer $\Delta \in \mathbb{Z}$ define ω as follows*

$$\omega = \begin{cases} \sqrt{\Delta}, & \text{if } \Delta \equiv 2, 3 \pmod 4 \\ \frac{1+\sqrt{\Delta}}{2}, & \text{if } \Delta \equiv 1 \pmod 4 \end{cases}$$

Then the algebraic integers in $\mathbb{Q}(\sqrt{\Delta})$ are exactly the numbers $a + b\omega \in \mathbb{Q}(\sqrt{\Delta})$ where $a, b \in \mathbb{Z}$.

Proof. Note that we need only examine monic quadratic polynomials since any element $a + b\sqrt{\Delta} \in \mathbb{Q}(\sqrt{\Delta})$ is the root of its minimal polynomial $x^2 - 2ax + (a^2 - b^2\Delta) \in \mathbb{Z}[x]$. This then requires that $2a \in \mathbb{Z}$ and $a^2 - b^2\Delta \in \mathbb{Z}$. If we take $a = \frac{1}{2}\alpha$ for some $\alpha \in \mathbb{Z}$ then we have $(\frac{1}{2}\alpha)^2 - b^2\Delta = \frac{1}{4}\alpha^2 - b^2\Delta \in \mathbb{Z}$. Since Δ is squarefree, we have $\Delta \not\equiv 0 \pmod 4$ so we must also have that $b^2 \in \frac{1}{4}\mathbb{Z}$, so $b = \frac{1}{2}\beta$ for some $\beta \in \mathbb{Z}$. Now we have $\frac{1}{4}\alpha^2 - \frac{1}{4}\beta^2\Delta \in \mathbb{Z}$, so $\alpha^2 - \beta^2\Delta \in 4\mathbb{Z}$. So roots to our polynomial must satisfy $\alpha^2 \equiv \beta^2\Delta \pmod 4$.

Solutions to $\alpha^2 \equiv \beta^2\Delta \pmod 4$ must be examined by cases concerning $\Delta \pmod 4$. If $\Delta \equiv 2 \pmod 4$, this implies that $\alpha^2 \equiv 2\beta^2 \pmod 4$, which is true only if $\alpha \equiv \beta \equiv 0 \pmod 2$. This in turn gives us $\alpha = 2k$ and $\beta = 2l$ for some integers $k, l \in \mathbb{Z}$. By this, we have $a + b\sqrt{\Delta} \in \mathbb{Z}[\sqrt{\Delta}]$ is an algebraic integer in $\mathbb{Q}(\sqrt{\Delta})$ if $\Delta \equiv 2 \pmod 4$. A nearly identical proof shows that if $\Delta \equiv 3 \pmod 4$ then the integers in $\mathbb{Q}(\sqrt{\Delta})$ are also all the elements of $\mathbb{Z}[\sqrt{\Delta}]$.

The only other case we must consider is $\Delta \equiv 1 \pmod 4$ (since Δ is squarefree, we need not consider $\Delta \equiv 0 \pmod 4$). In this case, our solutions must satisfy $\alpha^2 \equiv \beta^2 \pmod 4$. This is true precisely when $\alpha \equiv \beta \pmod 2$. If $\alpha \equiv \beta \equiv 0 \pmod 2$, we obtain algebraic integers $k + l\sqrt{\Delta}$ where $k, l \in \mathbb{Z}$ as above. However, if we take $\alpha \equiv \beta \equiv 1 \pmod 1$, then we can write $\alpha = 2i + 1$ and $\beta = 2j + 1$ for some $i, j \in \mathbb{Z}$. Then $a + b\sqrt{\Delta} = \frac{1}{2}\alpha + \frac{1}{2}\beta\sqrt{\Delta} = \frac{1}{2}(2i + 1) + \frac{1}{2}(2j + 1)\sqrt{\Delta}$. If we take $2j + 1 = l$ and $i = k + \frac{l-1}{2}$ we have the following series of equalities

$$\begin{aligned} \frac{1}{2}(2i + 1) + \frac{1}{2}(2j + 1)\sqrt{\Delta} &= \frac{1}{2}(2(k + \frac{l-1}{2}) + 1) + \frac{1}{2}l\sqrt{\Delta} \\ &= k + l\frac{1+\sqrt{\Delta}}{2} \end{aligned}$$

This solution, along with the solution in the case where $\alpha \equiv \beta \equiv 0 \pmod 2$ imply that $k + l\frac{1+\sqrt{\Delta}}{2}$ for $k, l \in \mathbb{Z}$ are the integers inside $\mathbb{Q}(\sqrt{\Delta})$ when $\Delta \equiv 1 \pmod 4$.

So far we have shown that any algebraic integer will be in $\mathbb{Z}[\omega]$, now we will show that ω is an algebraic integer. First, suppose $\Delta \equiv 1 \pmod 4$, then $\omega = \frac{1+\sqrt{\Delta}}{2}$ and the minimal polynomial for this

element is $x^2 - x + (\Delta - 1)/4$. Now we obtain the following system of equations.

$$\begin{aligned}\omega^2 - \omega + \frac{\Delta - 1}{4} &= \left(\frac{1 + \sqrt{\Delta}}{2}\right)^2 - \frac{1 + \sqrt{\Delta}}{2} - \frac{\Delta - 1}{4} \\ &= \frac{1 + 2\sqrt{\Delta} + \Delta}{4} - \frac{1 + 2\sqrt{\Delta} + \Delta}{4} \\ &= 0\end{aligned}$$

Now, if $\Delta \equiv 2, 3 \pmod 4$ then $\omega = \sqrt{\Delta}$ and our minimal polynomial is more simply $x^2 - \Delta$. Since $\omega \in \mathcal{O}_K$, our ring of integers is $\mathbb{Z}[\omega]$ by Theorem 4.2.2 in [5]. □

Prime ideals in $\mathbb{Z}$ are not necessarily prime when extended to the ring of integers $\mathcal{O}_K$ of a quadratic field K. If an ideal (p) is prime in $\mathcal{O}_K$, we say p is inert in $\mathcal{O}_K$. Otherwise, p may be the product of two prime ideals $\mathfrak{p}_1$ and $\mathfrak{p}_2$. If $\mathfrak{p}_1 = \mathfrak{p}_2$ we say p ramifies in $\mathcal{O}_K$, and if $\mathfrak{p}_1 \neq \mathfrak{p}_2$ we say p splits in $\mathcal{O}_K$.

Theorem 2.2. *Let (p) be an ideal in $\mathbb{Z}$. When (p) is extended to ring of integers inside a quadratic field, it is at most the product of two other prime ideals in the ring of integers.*

Proof. Let K be a quadratic field and $\mathcal{O}_K$ be the ring of integers in K. Suppose that for some $n \in \mathbb{Z}$, where $n > 2$ we have $(p) = \mathfrak{p}_1 \dots \mathfrak{p}_n$ for some prime ideals $\mathfrak{p}_1, \dots, \mathfrak{p}_n \in \mathcal{O}_K \subset \mathcal{O}_K$, then $|\mathcal{O}_K/(p)| = p^k = |\mathcal{O}_K/\mathfrak{p}_1| \times \dots \times |\mathcal{O}_K/\mathfrak{p}_n|$ where $k = 0, 1, 2$. If for some $i, j \in \{1, \dots, n\}$ we have $|\mathcal{O}_K/\mathfrak{p}_i| = p^2$, then the other residue fields have cardinality 1 and thus are trivial rings. If $|\mathcal{O}_K/\mathfrak{p}_i| = |\mathcal{O}_K/\mathfrak{p}_j| = p$, then all other residue fields must be the trivial ring. So (p) is the product of at most 2 other prime ideals. □

In order to determine whether a prime ideal splits, is inert, or ramifies we must use a tool developed by algebraic number theorists called the Kronecker symbol, $\left(\frac{D}{p}\right)$ where D is the the discriminant of K.

Definition 2.3. *The **Kronecker symbol** $\left(\frac{a}{b}\right)$ for $a, b \in \mathbb{Z}$ is defined by the following cases. If b is an odd prime, then*

$$\left(\frac{a}{b}\right) = \begin{cases} 0, \text{ if } b|a \\ \text{if } b \nmid a, \begin{cases} 1, \text{ if there exists some } x \in \mathbb{Z} \text{ such that } x^2 \equiv a \pmod b \\ -1, \text{ else} \end{cases} \end{cases}$$

If $b = 2$ then

$$\left(\frac{a}{2}\right) = \begin{cases} 0, & \text{if } a \equiv 0 \pmod 2 \\ 1, & \text{if } a \equiv \pm 1 \pmod 8 \\ -1 & \text{if } a \equiv \pm 3 \pmod 8 \end{cases}$$

Theorem 2.3. *The Kronecker symbol can be used to determine whether a prime p splits, ramifies, or is inert in a ring of integers $\mathcal{O}_K$ with discriminant D in the following way,*

$$\text{if } \left(\frac{D}{p}\right) = \begin{cases} 0, & p \text{ ramifies in } \mathcal{O}_K \\ 1, & p \text{ splits in } \mathcal{O}_K \\ -1, & p \text{ is inert in } \mathcal{O}_K \end{cases}$$

As an example let our motivating quadratic equation be $x^2 + 3 = 0$. Then our quadratic field is $K = \mathbb{Q}[\sqrt{-3}]$ with discriminant $D = -3$. The ring of integers $\mathcal{O}_{\mathbb{Q}[\sqrt{-12}]} = \mathbb{Z}\left[\frac{1+\sqrt{-3}}{2}\right]$ are known as the Eisenstein Integers. The ideal (2) is prime in $\mathbb{Z}$, and $\left(\frac{-3}{2}\right) = -1$, so (2) is inert and remains prime in the Eisenstein Integers. Now let's consider the ideals (5) and (3). These are prime ideals in $\mathbb{Z}$ but they behave differently in the Eisenstein Integers. Now we have $\left(\frac{-3}{5}\right) = 1$ so (5) splits in this ring of integers. $\left(\frac{-3}{3}\right) = 0$ so (3) ramifies in the Eisenstein integers.

More information and resources used in this section can be found in [4],[5], and [1].

3 Matrix Groups

Under the operations of matrix addition and multiplication, groups of matrices exist. For instance, under the operations of matrix addition, the collection of all 3 by 3 matrices with integer entries form a group. The zero matrix serves as the identity in this case and for any matrix A it has an inverse $-A = -1 \cdot A$. Of course, for addition to be defined on this group, we need the matrices to have the same dimensions. We also need our entries to come from a group under addition. In general, if we define a group of matrices under addition we call this group $M_n(S)$ of matrices with dimension $n \times n$, where the entries come from a additive group S.

For matrix groups under multiplication, we must impose some more restrictions on which matrices we allow. Matrix multiplication is only defined on matrices when the left matrix has the same number of columns as the matrix on the right, so we will assume that we have square matrices of dimension $n \times n$. We also only consider invertible matrices in this group, since each matrix must have an inverse in the group. Of course, the entries in each matrix must come from a ring which we will call S. As for notation, we call this group $GL_n(S) = \{A \in M_n(S) \mid A \text{ is invertible}\}$. If we think of $M_n(S)$ as a ring under addition and matrix multiplication then $GL_2(S)$ is the group of units in $M_n(S)$.

4 Möbius Transformations

Definition 4.1. *The projective completion of the complex numbers, denoted $\mathbb{P}^1(\mathbb{C})$, is the set of ordered pairs under the equivalence relation $\sim$, where $(X, Y) \sim (Z, W)$ if there exists $u \in \mathbb{C}$ such that $X = uZ$ and $Y = uW$. In other words*

$$\mathbb{P}^1(\mathbb{C}) = \{(X, Y) \in \mathbb{C} \times \mathbb{C} - (0, 0)\}/\sim .$$

Theorem 4.1. *There exists an isomorphism $\varphi : \mathbb{P}^1(\mathbb{C})\backslash\{[1, 0]\} \to \mathbb{C}$ defined $\varphi([X, Y]) \mapsto \frac{X}{Y}$ where $[X, Y]$ represents the equivalence class containing (X, Y) under the relation $\sim$.*

Proof. Any element $z \in \mathbb{C}$ can be represented as a quotient of two elements in $\mathbb{C}$. Mathematically, that is $z = \frac{p}{q}$ corresponds to $(p, q) \in (\mathbb{C}^2 - \{(0, 0)\})/\sim$. In this way we have a correspondence from $(X, Y) \in$

$(\mathbb{C}^2 - \{(0,0)\})/\sim \mapsto z = \dfrac{X}{Y} \in \mathbb{C}$. Through this map, our equivalence relation accounts for equivalent fractions, as denoted by the following equality in $\mathbb{C}$ to $\dfrac{X}{Y} = \dfrac{uZ}{uW} = \dfrac{Z}{W}$. □

Note that this maps all the elements of $\mathbb{P}^1(\mathbb{C})$ except for $[1,0]$ to elements of $\mathbb{C}$. So we will add another point, ∞, to $\mathbb{C}$ and which corresponds to $[1,0]$ via our map φ.

Definition 4.2. *The extended complex plane, denoted $\widehat{\mathbb{C}}$, is defined as $\mathbb{C} \cup \{\infty\}$ where $\{\infty\}$ is identified as the equivalence class of all elements $\{\frac{x}{0} \mid x \in \mathbb{C}^*\}$.*

Definition 4.3. *A Möbius Transformation is defined to be any function $f : \widehat{\mathbb{C}} \to \widehat{\mathbb{C}}$, $f(z) = \dfrac{az+b}{cz+d}$ where a, b, c, d are complex numbers satisfying $ad - bc \neq 0$.*

Any Möbius transformation can be expressed as a composition of magnifications, rotations, translations, and inversions. More importantly, Möbius transformations map circles and lines to other circles and lines, and they preserve angles locally.

Proposition 4.1. *Möbius Transformations express scaling, rotation, translation, and inversion.*

Proof. First note the general forms of scaling, rotation, translation, and inversion maps. For $z \in \mathbb{C}$, scaling and rotation maps are of the form $z \mapsto az$, $a \in \mathbb{C}^*$, translation maps are of the form $z \mapsto z + a$, $a \in \mathbb{C}$, and the inversion map is of the form $z \mapsto \dfrac{1}{z}$. □

Certainly all these actions can be defined by a Möbius transformation, however we have a stronger theorem stating that every Möbius transformation is some composition of these functions.

Proposition 4.2. *Every Möbius transformation is a composition of rotations, scalings, translations, and inversions.*

Proof. If $c \neq 0$ then we have

$$\frac{az+b}{cz+d} = \frac{a}{c} - \frac{ad-bc}{c} \cdot \frac{1}{cz+d}.$$

And in the event that $c = 0$, since we also require $ad - bc \neq 0$, we have

$$\frac{az+b}{cz+d} = \frac{az+b}{d} = \frac{a}{d}z + \frac{b}{d}.$$

□

It is convenient to represent Möbius transformations with matrices, where the parameters of the transformation correspond to a matrix in $GL_2(\mathbb{C})$ via the following map,

$$f(z) = \frac{az+b}{cz+d} \mapsto \begin{pmatrix} a & b \\ c & d \end{pmatrix}.$$

5 Bianchi Group

Definition 5.1. *A **Bianchi Group** is a group of determinant 1, two by two matrices whose entries come from a ring of integers inside a number field modulo scaling by a constant.*

The Bianchi Group is most commonly referred to as $PSL_2(\mathcal{O}_K)$. This group preserves enough information about the ring of integers it comes from to allow us to study the integers through its behavior.

5.1 Congruence Subgroup

Definition 5.2. *We define the **modular group** as*

$$\Gamma(a) = \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in PSL_2(\mathbb{Z}) \,\middle|\, a \equiv d \equiv 1 \pmod{p} \text{ and } b \equiv c \equiv 0 \pmod{p} \right\}.$$

The modular group is the set of all matrices in $PSL_2(\mathbb{Z})$ whose elements reduce component-wise to the identity matrix over a prime ideal.

Definition 5.3. *Let $\mathfrak{p}$ be a prime ideal in $\mathcal{O}_K$. The **congruence subgroup** over $\mathfrak{p}$ of $PSL_2(\mathcal{O}_K)$ is defined*

$$PSL_2(\mathfrak{p}) = \left\{ \begin{pmatrix} \alpha & \gamma \\ \beta & \delta \end{pmatrix} \in PSL_2(\mathcal{O}_K) \,\middle|\, \alpha \equiv \delta \equiv 1 \pmod{\mathfrak{p}} \text{ and } \beta \equiv \gamma \equiv 0 \pmod{\mathfrak{p}} \right\}.$$

That is, the congruence subgroup is the subset of $PSL_2(\mathcal{O}_K)$ whose elements reduce component-wise to the identity matrix over a prime ideal. The congruence subgroup is essentially the modular group in a general ring of integers.

We will show that the group of Möbius transformations under function composition is isomorphic to a subgroup of $PGL_2(\mathbb{C})$.

Proposition 5.1. *Matrix representations of Möbius transformations preserves function composition by matrix multiplication.*

Proof. Let f, g be Möbius transformations defined as

$$f(z) = \frac{az+b}{cz+d},\ g(z) = \frac{\alpha z+\beta}{\gamma z+\delta}$$

And by composing $f(g(z))$ we obtain

$$\begin{aligned} f(g(z)) &= \frac{a\left(\dfrac{\alpha z+\beta}{\gamma z+\delta}\right)+b}{c\left(\dfrac{\alpha z+\beta}{\gamma z+\delta}\right)+d} \\ &= \frac{(a\alpha+b\gamma)z+a\beta+b\delta}{(c\alpha+d\gamma)z+c\beta+d\delta} \end{aligned}$$

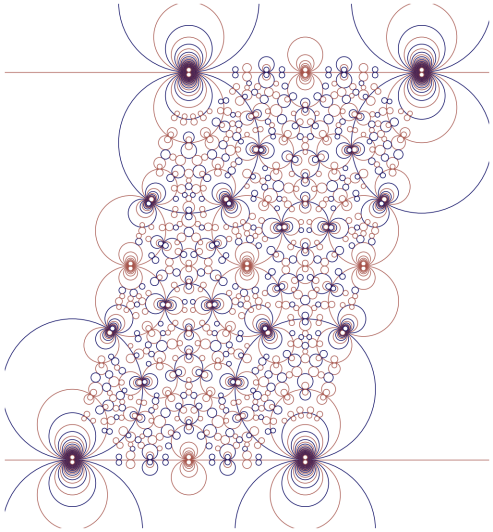


Figure 1: The Schmidt arrangement of $\mathcal{O}_{\mathbb{Q}(\sqrt{-11})}$ [3]

Which corresponds to the matrix obtained when multiplying the corresponding matrices for f and g. That is

$$f(g(z)) \to \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} = \begin{pmatrix} a\alpha + b\gamma & a\beta + b\delta \\ c\alpha + d\gamma & c\beta + d\delta \end{pmatrix}$$

□

So this map is a bijection between Möbius Transformations and $PGL_2(\mathbb{C})$.

6 Schmidt Arrangement

The following section includes results studied and written in conjunction with fellow mathematicians of University of Colorado Boulder, Andrew Jensen, Cherry Ng, Tyler Schrock and Katherine E. Stange.

Definition 6.1. *The **Schmidt Arrangement** of a ring of integers, $\mathcal{O}_K$, is the orbit of the extended real line under all Möbius transformations parameterized by matrices in $PSL_2(\mathcal{O}_K)$.*

Schmidt arrangements are studied as they visualize properties of the ring of integers that generates them. For example, if a Schmidt arrangement is connected, then it is a Euclidean domain.

Definition 6.2. *The **stabilizer** of $\widehat{\mathbb{R}}$ is the subgroup of transformations in $PSL_2(\mathcal{O}_K)$ that take the extended real line to itself.*

The stabilizer of the Bianchi Group is denoted $\mathrm{Stab}(\widehat{\mathbb{R}})$. This group is exactly the matrices with real integer entries, that is $\mathrm{Stab}(\widehat{\mathbb{R}}) = PSL_2(\mathbb{Z})$.

Generally, the Schmidt Arrangement of a ring of integers entirely consists of tangent circles, however this is not always the case. If $\mathcal{O}_K$ contains units other than 1 and -1, then $\mathcal{S}_K$ may have circles that overlap.

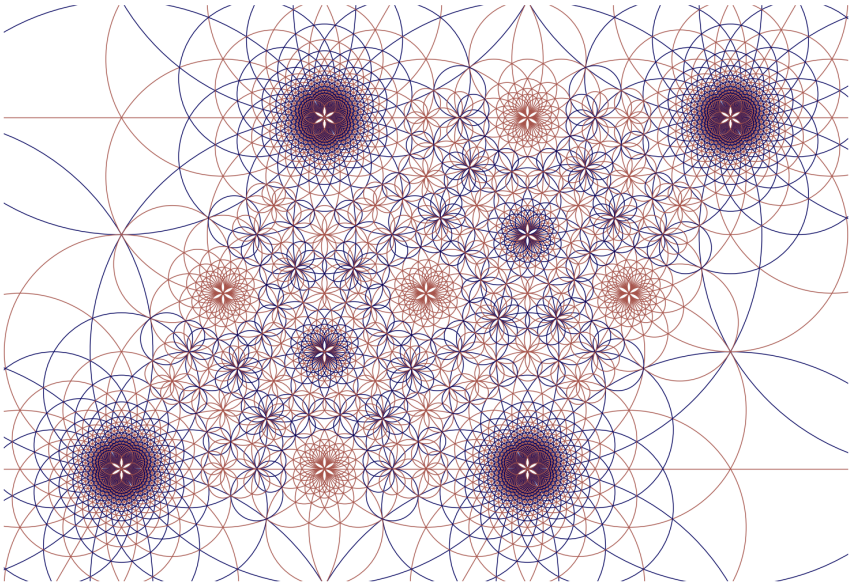


Figure 2: Schmidt arrangement of the Eisenstein integers[3]

These overlapping circles intersect only at angles given by those units in radians [3]. There are only two cases where this occurs, namely $\mathcal{S}_{\mathbb{Q}(\sqrt{-1})}$ and $\mathcal{S}_{\mathbb{Q}(\sqrt{-3})}$. In the case of $\mathcal{S}_{\mathbb{Q}(\sqrt{-1})}$ the Schmidt arrangement is easily separated into two tangent subarrangements that are identical when either one of them is rotated by $\pi/2$ radians. However, this is not the case for $\mathcal{S}_{\mathbb{Q}(\sqrt{-3})}$. The process of splitting $\mathcal{S}_{\mathbb{Q}(\sqrt{-3})}$ into tangent subarrangements involves analyzing the orbit of the real line under congruence subgroups of the Bianchi Group.

Theorem 6.1. *Let $\mathfrak{a}$ be a prime ideal in $\mathcal{O}_K$. Then*

$$\mathrm{Stab}_{PSL_2(\mathfrak{a})}(\widehat{\mathbb{R}}) = \mathrm{Stab}_{PSL_2(\mathcal{O}_k)}(\widehat{\mathbb{R}}) \cap PSL_2(\mathfrak{a}) = \Gamma(p)$$

where p is the generator of $\mathfrak{a} \cap \mathbb{Z}$.

Proof. Let φ be the map that reduces elements of $\mathcal{O}_K$ modulo $\mathfrak{a}$. Then

$$\mathrm{Stab}_{PSL_2(\mathfrak{a})}(\widehat{\mathbb{R}}) = \mathrm{Stab}_{PSL_2(\mathcal{O}_K)}(\widehat{\mathbb{R}}) \cap PSL_2(\mathfrak{a}) = \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in PSL_2(\mathbb{Z}) \,\middle|\, a, d \in \varphi^{-1}(1) \text{ and } b, c \in \varphi^{-1}(0) \right\}$$

So $a \equiv 1 \pmod{\mathfrak{a}}$, which means $a - 1 \in \mathbb{Z} \cap \mathfrak{a}$. If $\mathfrak{a}$ sits over $(p) \subset \mathbb{Z}$ then $a - 1 \in p\mathbb{Z}$. Therefore $a \equiv 1 \pmod{p}$. The same argument applies to d. A similar argument proves that b and c are equivalent to $0 \pmod{p}$. Thus $\mathrm{Stab}_{PSL_2(\mathcal{O}_k)}(\widehat{\mathbb{R}}) \cap PSL_2(\mathfrak{a}) = \Gamma(p)$. □

Theorem 6.2. *All circles in the Schmidt arrangement of the congruence subgroup passing through a particular point are tangent at that point.*

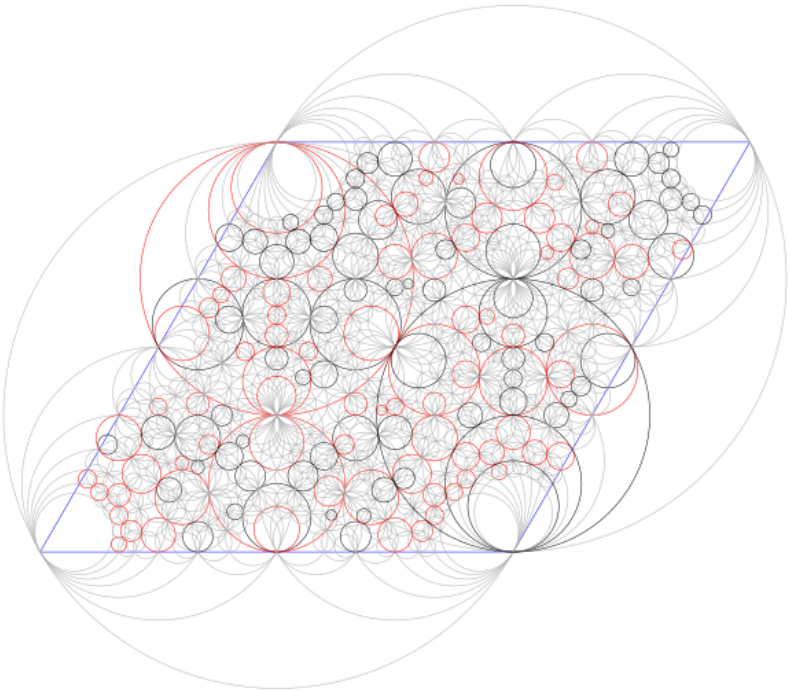


Figure 3: Congruence subarrangement of $PSL_2(2\mathcal{O}_{\mathbb{Q}[\sqrt{-3}]})$. The black and red circles belong to the congruence subarrangement. The grey circles are the rest of the Schmidt arrangement.

Proof. Consider the circles passing through 0. These are all images of $\widehat{\mathbb{R}}$ under Möbius transformations of the form

$$\begin{pmatrix} \gamma & 0 \\ \delta & 1 \end{pmatrix} \in PSL_2(\mathfrak{a})$$

Such matrices are congruent to $I_2 \mod \mathfrak{a}$. This implies that γ is a unit in $\mathfrak{a}$. In fact, $\gamma \equiv 1 \mod \mathfrak{a}$, which means $\gamma = \pm 1$. Since the curvature-center is $i\gamma$ and curvature is purely real, this means that the center of each circle is purely imaginary. Since every circle passing through the origin has a purely imaginary center, this means they are all tangent at the origin. This extends to circles passing through all other points by the conformality of Möbius transformations. □

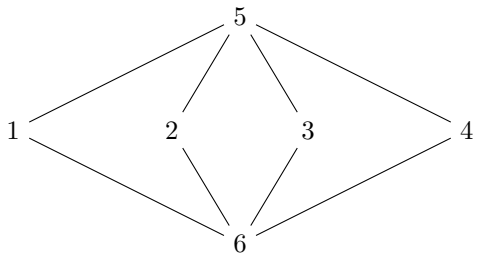
By these theorems, we can find a tangent subarrangement in the Schmidt arrangement of the Eisenstein integers by using only the circles from a congruence subgroup. Its cosets in the full arrangement will also be tangent by the conformality of Möbius transformations. In fact, so long as the ideal is proper, it won't contain the units that allow for non-tangent intersections to occur.

6.1 Apollonian Circle Packing Structure

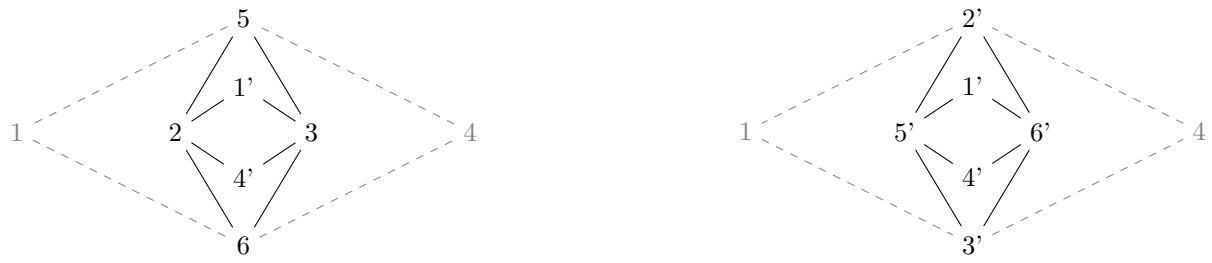
Schmidt arrangements frequently feature an Apollonian circle packing structure that will generate the whole picture. We propose that there exists a similar circle packing structure that will generate the congruence-(2)

Schmidt arrangement of the Eisenstein integers.

The congruence subarrangement of the Schmidt arrangement of the Eisenstein integers $PSL_2(2)$ exhibits 6-tuples having the following tangency graph:



We can define a **swap** analogous to the swaps described in classical Apollonian circle packings. For example, we can swap out circles 1 and 4 to obtain 1' and 4' in the following manner



where the picture on the right has been relabeled so that the numbers 5 and 6 are associated with the two circles that are tangent to 4 circles and the numbers 1, 2, 3, 4 are associated with the four circles that have 2 tangencies. More precisely, the relabeling is described by

$$2 \to 5' \qquad 3 \to 6' \qquad 5 \to 2' \qquad 6 \to 3'.$$

Let $v_1, v_2, \ldots, v_6$ be the images of the six circles described above under the Pedoe map described on page 16 of "The Apollonian structure of Bianchi groups." (Note: The interiors of all 6 circles must be disjoint, so a circle containing all the other circles must have negative curvature.) Then we can perform the swaps by using a linear dependence between the 6 vectors. The linear dependence between our 6 original circles is

$$v_1 = -v_3 + 2v_5 + 2v_6 \qquad v_4 = -v_2 + 2v_5 + 2v_6.$$

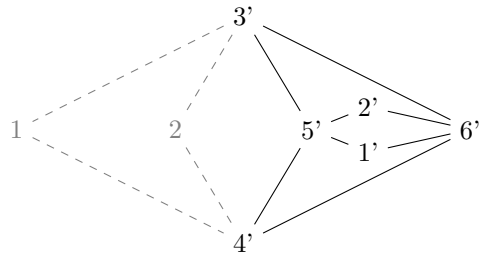
Thus the linear dependence that we use to swap v_1 for v_1' and to swap v_4 for v_4' is

$$v_1' = -v_6 + 2v_2 + 2v_3 \qquad v_4' = -v_5 + 2v_2 + 2v_3.$$

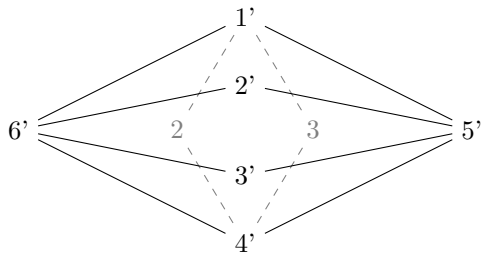
An example for a similar packing structure in $\mathbb{Q}(\sqrt{-7})$ is on p.38 of [2] which serves as an analogue of what is described next. Let matrix M be the matrix whose columns are circles 2,3,5 and 6 under the Pedoe map. We may safely exclude circles 1 and 4 because they can be written as a linear combination of circles 2,3,5 and 6 using the relations given above.

Then the matrix A_{ij} swaps circles i and j for i' and j'. Our convention is to orient the new diagram such that 1', 2', 3', and 4' can be read left-to-right, wrapping around from the right-most edge to the left-most edge when necessary. Then using this orientation, 5' is the four-tangency circle on"top" and 6' is the four-tangency circle on "bottom." The matrices and corresponding diagrams are given below:

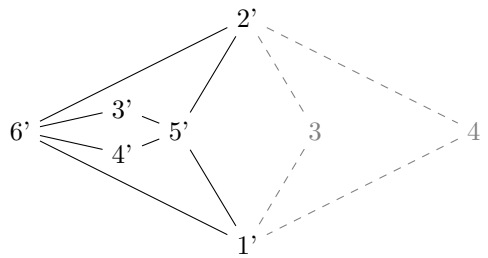
$$\mathrm{A}_{12} = \begin{pmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 2 \end{pmatrix}$$



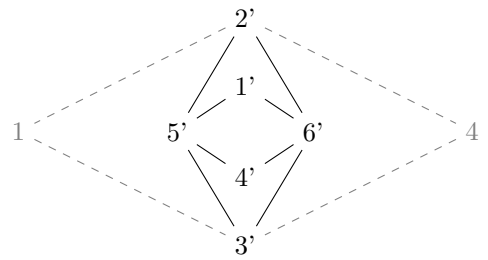
$$\mathrm{A}_{23} = \begin{pmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 2 & 2 \\ 0 & 1 & 2 & 2 \end{pmatrix}$$



$$\mathrm{A}_{34} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 2 \end{pmatrix}$$



$$\mathrm{A}_{14} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$



As an example, we will perform these circle swaps on an arbitrary set of tangent circles in $PSL_2(2)$. Figure 4 shows how these swaps affect a group of tangent circles. This tangency pattern certainly exists in the full Schmidt arrangement of the Eisensteins. It also appears to be visible in the congruence 2 subarrangement of the Eisenstein integers, however proof of this is part of ongoing research.

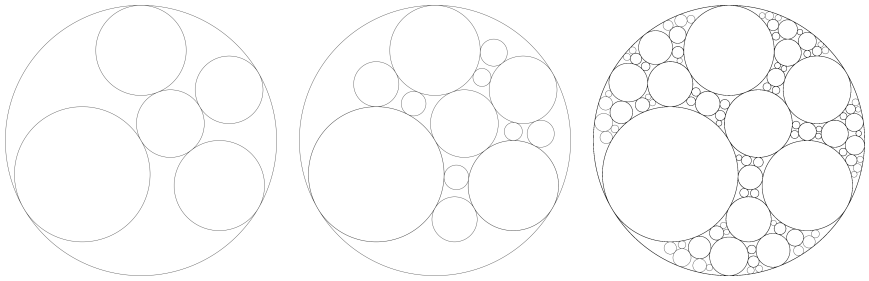


Figure 4: A group of 6 tangent circles from $PSL_2(2)$ and how the image changes after performing the swaps described above once and then multiple times.

References

[1] Pierre Samuel. *Algebraic Theory of Numbers: Translated from the French by Allan J. Silberger*. Courier Corporation, 2013.

[2] Katherine E Stange. "The Apollonian structure of Bianchi groups". In: *arXiv preprint arXiv:1505.03121* (2015).

[3] Katherine E Stange. "Visualising the arithmetic of imaginary quadratic fields". In: *arXiv preprint arXiv:1410.0417* (2014).

[4] Ian Stewart and David Tall. *Algebraic number theory and Fermat's last theorem*. CRC Press, 2015.

[5] Mak Trifković. *Algebraic theory of quadratic numbers*. Springer, 2013.

7 Appendix

7.1 Congruence Picture Code

Below is the Sage code used to create the congruence picture.

```
def matrixColor (alpha, beta, gamma, delta):
#print "reduced entries:" , F(alpha), F(beta), F(gamma), F(delta)
if matrixInBase(alpha,beta,gamma,delta):
#Blue
return [.07,.44,.70]
elif matrixInBase(alpha + (1+a)*gamma/2, beta + (1+a)*delta/2, gamma, delta):
#Red
return [.55,.11,.11]
else:
#Grey
return [.75,.75,.75]
print "success"


#box 1
#set up ring of integers, import statements, etc.
from itertools import *
from time import time
K.<a> = NumberField(x^2+3)
R = K.ring_of_integers()
X = K.ideal(2)
M = matrix([[1,0],[(a+1)/2,1]])
F = R.residue_field(X)
print "success"
#box 2
#generate list of Eisensteins integers within a particular norm
normMax=9
#Change this to change list of integers
v1=1
v2=(a+1)/2
v1scalar=1
v2scalar=0
integers=[0]
```

```
while v1scalar<normMax:
v2scalar=0
while v2scalar<normMax:
v=v1scalar*v1+v2scalar*v2
if sqrt(norm(v))<normMax:
if True:
integers.append(v)
integers.append(v*(1+a)/2)
integers.append(v*((1+a)/2)^2)
integers.append(v*((1+a)/2)^3)
integers.append(v*((1+a)/2)^4)
integers.append(v*((1+a)/2)^5)
v2scalar=v2scalar+1
v1scalar=v1scalar+1
print len(integers)
print integers

#box 3

#generate dictionary of circles
#Note: Norm 20 takes about 13 hours to run
startTime = time()
print "here"
alphaDelta=[]
circlePoints={}
b=sqrt(-3)
numberOfCircles=0
#the parallelogram shape
parallelogram = line([(0,0),(1,0)]) + line([(1,0), (1.5,sqrt(3)/2)]) + line([(0,0),(.5, sqrt(3)/2)]) + l
for count1 in [1,..,len(integers)-1]:
for count2 in [1,..,len(integers)-1]:
alpha = integers[count1]
beta = integers[count2]
I = R.ideal(alpha)
J = R.ideal(beta)
#        print (alpha,beta)
```

```
#        print "coprime?", I.is_coprime(J)
if I.is_coprime(J):
r = I.element_1_mod(J)
alphaDelta.append(r)
gamma = (r-1)/beta
delta = r/alpha
#            if (sqrt(norm(gamma*conjugate(delta)- conjugate(gamma)*delta)))==sqrt(3):
#                if (alpha*delta-gamma*beta) == 1:
#                    print "alpha:",alpha
#                    print "beta:",beta
#                    print "gamma:",gamma
#                    print "delta:",delta
#            if K(alpha).mod(X) == K(1).mod(X):
#                if K(beta).mod(X) == K(0).mod(X):
#                    if K(delta).mod(X) == K(1).mod(X):
#                        if K(gamma).mod(X) <> K(0).mod(X):
#                            gamma = gamma - alpha*gamma
# check if det=1
if alpha*delta-gamma*beta == 1:
#bigMatrix = matrix([[1,0],[0,1]]) #
#bigMatrix = matrix([[1,0],[a*(1/2)+(1/2),1]])
#circleMatrix = matrix([[alpha,beta],[gamma,delta]])
#newMatrix = bigMatrix*circleMatrix
#print newMatrix[0][0] == alpha
#alpha = newMatrix[0][0]
#beta = newMatrix[0][1]
#gamma = newMatrix[1][0]
#delta = newMatrix[1][1]
#print alpha
if norm(gamma*conjugate(delta)- conjugate(gamma)*delta)!=0:
radius = 1/(sqrt(norm(gamma*conjugate(delta)- conjugate(gamma)*delta)))
#Add this if statement if you want to restrict by circles of a particular radius
#But it's better to do it in the next box. Doing it here save very little processing time
#                if radius == 1/(5*sqrt(3)):
center = (alpha*conjugate(delta) - beta*conjugate(gamma))/(gamma*conjugate(delta) - conjugate(gamma)*del
realCenter = RR(center.trace())/2
```

```
imaginaryCenter = RR((2*center-center.trace())/a)/2
H = realCenter
k = imaginaryCenter*sqrt(3)
U = H - (1/sqrt(3))*k
V = (2/sqrt(3))*k
t = 0
while V > 1:
H = H - (1/2)
k = k - (sqrt(3)/2)
V = (2/sqrt(3))*k
t = t - a/2 - (1/2)
while V < 0:
H = H + (1/2)
k = k + (sqrt(3)/2)
V = (2/sqrt(3))*k
t = t + a/2 +(1/2)
while U < 0:
H = H + 1
U = H - (1/sqrt(3))*k
t = t + 1
while U > 1:
H = H - 1
U = H - (1/sqrt(3))*k
t = t - 1
alpha = alpha + t*gamma
beta = beta + t*delta
color = matrixColor(alpha, beta, gamma, delta)
if norm(gamma*conjugate(delta)- conjugate(gamma)*delta)!=0:
center = (alpha*conjugate(delta) - beta*conjugate(gamma))/(gamma*conjugate(delta) - conjugate(gamma)*del
realCenter = RR(center.trace())/2
imaginaryCenter = RR((2*center-center.trace())/a)/2
H = realCenter
k = imaginaryCenter*sqrt(3)
circlePoints[str(H) + "," + str(k) + "," + str(radius)]=[H,k,radius,color]
#            else:
#                print "Did not satisfy conditions"
```

```
#Saves the dictionary to a file so that we can use it later without needing to take another 12 hours to
#Unfortunately, I never got the load to work
#It would load the dictionary, and I could print the dictionary, but I could never loop through it as ei
#save(circlePoints, DATA + "circlePoints")
endTime = time()
#Calculate and print the time spent creating the dictionary
timePassed = endTime-startTime
days = timePassed // 86400
hours = timePassed // 3600 % 24
minutes = timePassed // 60 % 60
if days > 0:
print days,"days",hours,"hours",minutes,"minutes and some odd seconds"
elif hours > 0:
print hours,"hours",minutes,"minutes and some odd seconds"
elif minutes > 0:
print minutes,"minutes and some odd seconds"
else:
print "some odd seconds" #because my second calculating code never worked :(
print "success"
#print circlePoints

#box 4
startTime = time()
#Boo :( This load won't let me interact with circlePoints like I want to ):
#But if you've just generated the array in box 3, it works fine
#print "loading circlePoints..."
#circlePoints = load(DATA + "circlePoints")
#print "loaded criclePoints!"
#print circlePoints
#print "startTime:",startTime
alphaDelta=[]
circles=circle((0,0),0)
#circlePoints={}
b=sqrt(-3)
numberOfCircles=0
parallelogram = line([(0,0),(1,0)], rgbcolor=[.75,.75,.75]) + line([(1,0), (1.5,sqrt(3)/2)], rgbcolor=[
```

```
#print "circlePoints:",circlePoints
for key in circlePoints:
#print key
circles = circles+circle((circlePoints[key][0],circlePoints[key][1]),circlePoints[key][2], rgbcolor = c
circles = circles + parallelogram
endTime = time()
#Print the time passed
timePassed = endTime-startTime
days = timePassed // 86400
hours = timePassed // 3600 % 24
minutes = timePassed // 60 % 60
if days > 0:
print days,"days",hours,"hours",minutes,"minutes and some odd seconds"
elif hours > 0:
print hours,"hours",minutes,"minutes and some odd seconds"
elif minutes > 0:
print minutes,"minutes and some odd seconds"
else:
print "some odd seconds"
#Show the figures
circles.show(figsize=20,axes=False)
print "success"
```

7.2 Circle Swap Image Code

Below is the Sage code used to create the Circle Swap Pictures

```
def conj(x):
y = x - 2*I*(imag_part(x))
return y


#This function creates a vector given the center (as imaginary coordinate) and curvature
def findVector(center, curv):
realcenter = real_part(center)
imaginarycenter = imag_part(center)
cocurvature = ((center*conj(center)-1)/curv).simplify()
result = [cocurvature, curv, realcenter, imaginarycenter]
return result
```

```
print "Circle1:"
print findVector((sqrt(-3)/3)+((1+sqrt(-3))/2),sqrt(3))
print "Circle2:"
print findVector(sqrt(-3)/3,sqrt(3))
print "Circle3:"
print findVector((sqrt(-3)/3)+2,sqrt(3))
print "Circle4:"
print findVector(1+sqrt(-3)/6,2*sqrt(3))

#Testing circles!
#TestCircle = circle((0, 1/3*sqrt(3)), 1/(sqrt(3)))

circles = circle((1/2,sqrt(3)/6),1/sqrt(3), rgbcolor = (0,0,0))
circles = circles + circle((1/4,sqrt(3)/12),1/(2*sqrt(3)), rgbcolor = (0,0,0))
circles = circles + circle((5/8,5*sqrt(3)/24),1/(4*sqrt(3)), rgbcolor = (0,0,0))
circles = circles + circle((7/8,7*sqrt(3)/24),1/(4*sqrt(3)), rgbcolor = (0,0,0))
circles = circles + circle((1/2,7*sqrt(3)/18),1/(3*sqrt(3)), rgbcolor = (0,0,0))
circles = circles + circle((5/6,sqrt(3)/18),1/(3*sqrt(3)), rgbcolor = (0,0,0))

#circles = circles + TestCircle

#We start with a matrix, "M" where the rows represent the vectors for circles 1, 2, 3, 4, 5, and 6 by ou

M = matrix(6,4,[0,2*sqrt(3),sqrt(3)/2,1/2,2*sqrt(3),3*sqrt(3),3*sqrt(3)/2,7/2,4*sqrt(3),4*sqrt(3),7*sqr
show(M)
show(circles,figsize = 20, axes=False)
circles.save('posterimage1.png',figsize = 20, axes = False)

firstSwap = matrix(4,4,[0,0,0,-1,0,0,1,0,1,0,0,2,0,1,0,2])
secondSwap = matrix(4,4,[0,0,-1,0,0,0,0,0,-1,1,0,2,2,0,1,2,2])
thirdSwap = matrix(4,4,[0,0,1,0,0,0,0,0,-1,1,0,0,2,0,1,0,2])
fourthSwap = matrix(4,4,[0,0,1,0,0,0,0,0,1,1,0,0,0,0,1,0,0])

#Takes M as argument assuming it satisfies conditions above
#Returns image "Circles"
#If numberOfIterations is greater than 4, it exits the recursive process
```

```
#Given the matrix M, this function performs each of the four possible swaps and adds the new circles to
#It then recursively calls itself on each of the new configurations of six circles
def DoSwaps(matrix,numOfIts,listOfMatrices):

#CHANGE THIS TO CHANGE NUMBER OF ITERATIONS

#Exit Condition
if (numOfIts > 5):
return

#Calls each of the swaps
else:
DoFirstSwap(matrix,numOfIts,listOfMatrices)
DoSecondSwap(matrix,numOfIts,listOfMatrices)
DoThirdSwap(matrix,numOfIts,listOfMatrices)
DoFourthSwap(matrix,numOfIts,listOfMatrices)
return

#multiplies matrix by firstSwap and adds the new matrix to listOfMatrices and then calls DoSwaps again
def DoFirstSwap(matrix,numOfIts,listOfMatrices):

newMatrix = matrix*firstSwap
listOfMatrices.append(newMatrix)
DoSwaps(newMatrix,(numOfIts+1),listOfMatrices)
return

#multiplies matrix by secondSwap and adds the new matrix to listOfMatrices and then calls DoSwaps again
def DoSecondSwap(matrix,numOfIts,listOfMatrices):

newMatrix = matrix*secondSwap
listOfMatrices.append(newMatrix)
DoSwaps(newMatrix,(numOfIts+1),listOfMatrices)
return

#multiplies matrix by thirdSwap and adds the new matrix to listOfMatrices and then calls DoSwaps again
def DoThirdSwap(matrix,numOfIts,listOfMatrices):
```

```
newMatrix = matrix*thirdSwap
listOfMatrices.append(newMatrix)
DoSwaps(newMatrix,(numOfIts+1),listOfMatrices)
return

#multiplies matrix by fourthSwap and adds the new matrix to listOfMatrices and then calls DoSwaps again
def DoFourthSwap(matrix,numOfIts,listOfMatrices):

newMatrix = matrix*fourthSwap
listOfMatrices.append(newMatrix)
DoSwaps(newMatrix,(numOfIts+1),listOfMatrices)
return



def MatrixToPicture(matrix,circles):
matrixt = matrix.transpose()
firstCircleMatrix = list(-1*matrixt[0]+2*matrixt[2]+2*matrixt[3])
firstCircleList = (firstCircleMatrix[0],firstCircleMatrix[1],firstCircleMatrix[2],firstCircleMatrix[3])
secondCircleMatrix = -1*matrixt[1]+2*matrixt[2]+2*matrixt[3]
secondCircleList = (secondCircleMatrix[0],secondCircleMatrix[1],secondCircleMatrix[2],secondCircleMatri
circles = circles + VectorToPicture(firstCircleList)
circles = circles + VectorToPicture(secondCircleList)
#show(circles)
#circles = circles + VectorToPicture(matrixt[0])
#circles = circles + VectorToPicture(matrixt[1])
#circles = circles + VectorToPicture(matrixt[2])
#circles = circles + VectorToPicture(matrixt[3])
return circles

#Takes circle vector as a list and returns the graphics object for that circle
def VectorToPicture(vector):
if vector[1]==0:
return circle((0,0),0)
if vector[1] > 50*sqrt(3):
return circle((0,0),0)
```

```
if vector[2]/vector[1]>4:
return circle((0,0),0)
if vector[2]/vector[1]<-2:
return circle((0,0),0)
pic = circle((vector[2]/vector[1],vector[3]/vector[1]),1/vector[1], rgbcolor = (0,0,0))
return pic

print "START"
#circles = circle((0,0),0)
N = (matrix(4,4,[2*sqrt(3),3*sqrt(3),3*sqrt(3)/2,7/2,4*sqrt(3),4*sqrt(3),7*sqrt(3)/2,7/2,2*sqrt(3), 4*s
#N = (matrix(4,4,[2*sqrt(3),2*sqrt(3),2*sqrt(3),1,4*sqrt(3),sqrt(3),2*sqrt(3),1,2*sqrt(3),sqrt(3),sqrt(3
show(N)
listOfMatrices = [N]
DoSwaps(N,1,listOfMatrices)
print "MIDDLE"
print len(listOfMatrices)
for x in (0..len(listOfMatrices)-1):
if x%50==0:
print x
circles = MatrixToPicture(listOfMatrices[x],circles)
show(circles, axes = False, figsize = 20)
print "END"

circles.save('posterimage3.png',figsize = 20, axes = False)
```