Reflection Groups

and utilising them to tessellate the
hyperbolic plane.

Lawrence Green
The University of York
Department of Mathematics
May 16, 2020

Abstract

This project begins by introducing the concept of reflections to the reader in a rigorous fashion, and how reflections produce a group structure. This group structure is then explored and morphed into a linear algebra construction. These linear algebra constructions are called root systems, and are used to classify the reflection groups. After briefly introducing hyperbolic geometry, the reader is taken step by step through the construction of a program which will generate an image based on a particular reflection group.

Contents

1. Reflection Groups

1.1. **Introduction to reflections.** In order to begin studying reflection groups it would be a good idea to have a concrete idea of what a reflection actually is in mathematical terms. First we need to fix the space we are working in, throughout this projects we will denote real valued $n \times n$ Euclidean space as V. We will equip these spaces with a positive definite symmetric bilinear form.

Definition 1. A positive definite symmetric bilinear form is a map $(\cdot,\cdot) : V \times V \to k$ which satisfies the following axioms:

B1: $(x, y) = (y, x)$
B2: $(x + y, z) = (x, z) + (y, z)$
B3: $(\lambda x, y) = \lambda(x, y)$
B4: $(x, x) > 0 \ \forall x \neq 0$.

Such that the above axioms are satisfied for all x, y, z in V and $\lambda \in k$. where k is the underlying field of scalars for V, in our case that is $\mathbb{R}$. Condition **B1** is the symmetric property, the other conditions imply linearity of the first argument, but by **B1** both arguments are linear. It is important to note that **B4** is the positive definite condition, and maps satisfying just axioms **B1** - **B3** are considered symmetric bilinear forms. It will be useful to keep these conditions in mind as we will relax them later when we move into hyperbolic space.

If you where to think about reflections that take place in the real world naturally your mind drift towards mirrors; in fact using mirrors and the bilinear form we can start to mathematically define a reflection on V. We start by considering the orthonormal group denoted $O(V)$ and is defined in [5] as:

$$O(V) = \{T : V \to V \mid T \text{ is linear and } (Tx, Ty) = (x, y) \text{ for all } x, y \in V\}$$

This collection of functions is the set of distance preserving linear maps, which is dependant on the bilinear form chosen. All reflections are distance preserving and as such are all contained within this group.

Definition 2. A linear map s_α which is in $O(V)$ is called a reflection if and only if: the transformation s_α maps α to $-\alpha$, and the set of fixed points of s_α is exactly a hyperplane in V. The hyperplane that is fixed by s_α is the "Mirror" of the reflection, which is denoted H_α and is defined as the following :

$$H_\alpha = \{x \in V \mid (x, \alpha) = 0\}$$

The vector α is called the root of the reflection because it determines the mirror, and thus the entire reflection.

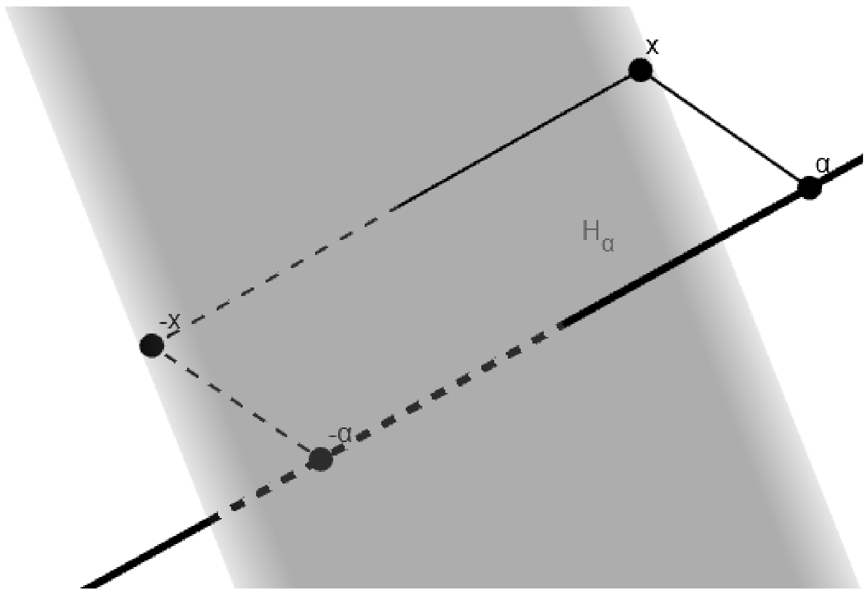


Figure 1. A reflection in $\mathbb{R}^3$

Theorem 1. *A reflection with root α can be expressed with the equation:*

$$s_\alpha x = x - \frac{2(x,\alpha)}{(\alpha,\alpha)}\alpha$$

Proof. This proof is based on the one found in [5]. Take an element $y \in H_\alpha$ and because α and y are orthogonal to each other, any element $x \in V$ can be decomposed into $x = \lambda\alpha + \mu y$ [7] for λ and μ in k. Using the decomposition of x, by linearity $s_\alpha x$ can be decomposed into the following:

$$s_\alpha x = s_\alpha(x - \lambda\alpha + \lambda\alpha) = s_\alpha(x - \lambda\alpha) + s_\alpha(\lambda\alpha) = s_\alpha(\mu y) + s_\alpha(\lambda\alpha)$$

Since $y \in H_\alpha$, $\mu y = x - \lambda\alpha$ remains fixed under s_α. By construction λ can be any scalar, chose $\lambda = \frac{(\alpha,x)}{(\alpha,\alpha)}$ and insert the values for λ and μ into the above equation, which simplifies as such:

$$s_\alpha x = x - \lambda\alpha - \lambda\alpha = x - 2\lambda\alpha = x - \frac{(\alpha,x)}{2(\alpha,\alpha)}\alpha$$

□

1.2. **Finite reflection groups.** Now that reflections have been defined, by taking the operation of function composition a group structure can be found and examined. To begin only the finite case will be considered, but it can extended to the infinite case.

Definition 3. We call a set $G \subset O(V)$ a finite reflection group if G is a finite group and is generated by reflections. in this context the term generated is similar to that of generation by an ideal:

$$G = \langle s_\alpha : s_\alpha \text{ is a reflection} \rangle$$

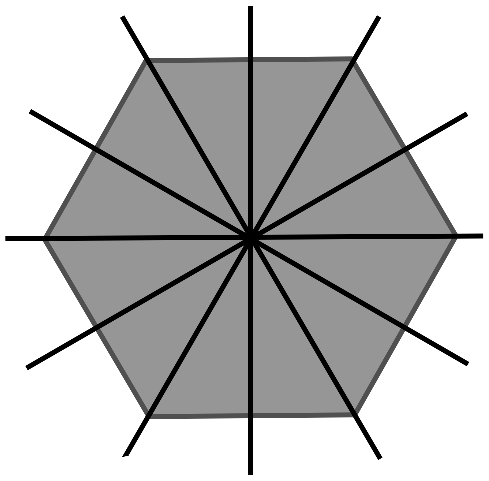


FIGURE 2. The reflectional symmetries of a hexagon, where the mirrors are represented by the lines bisecting the hexagon.

A rather trivial application of finite reflection groups is to represent the symmetries of regular n-sided polygons; this includes the reflectional and rotational symmetries. A reflection group can represent rotational symmetries by composing two reflections to form a rotation.

In the case of regular polygons, a maximum of $4n$ reflections encapsulate all of its symmetries; n rotations, which require 2 reflections each and the 1 reflection for each edge and vertex.

To simplify these groups, it can be shown that only 2 reflections are required to generate the entire group [2]. Reflection groups of this kind are referred to as planar dihedral groups, with $I_2(n)$ being the group for the cosponsoring n-sided polygon.

1.3. **Root systems.** When studying reflection groups the goal is to analyse the geometry of the reflecting hyperplanes, and how the group action of W affects these hyperplanes in V. When working in two dimensions this is easy to analyse, as a picture can be drawn and studied.

In higher dimensions this becomes harder to do; as usual the solution is to turn to an algebraic representation, from which the reflection group itself, along with the geometric configuration of the hyperplanes can be reconstructed.

Take the set of vectors $\{\alpha \mid s_\alpha \in W\}$ which are all roots for a reflection in W, for each vector α extend it infinitely to form a line L_α. By observing the lines L_α under the action of W the following theorem from [4] begins to demonstrate how W acts on V.

Theorem 2. *If $T \in O(V)$ and α is any non zero vector in V then:*

$$T s_\alpha T^{-1} = s_{T\alpha}$$

Proof. Using the definition for a reflection, it is sufficient to check just two conditions. The first step is to check if the root is mapped to its negative, the second step is to check that a hyperplane is fixed by this map. Finally it is also important to show that this reflection is the correct one, and that the fixed hyperplane is consistent with the chosen root. First check that $Ts_\alpha T^{-1}$ maps $T\alpha$ to $-T\alpha$ making it good candidate for the root of this reflection.

$$(Ts_\alpha T^{-1})T\alpha = Ts_\alpha(T^{-1}t)\alpha = T(s_\alpha\alpha) = -Ts_\alpha$$

The next step involves not only checking that a hyperplane is fixed, but that the hyperplane orthogonal to $T\alpha$ is fixed. Since $T \in O(V) \Rightarrow (x, y) = (Tx, Ty)$, take an element $x \in H_\alpha$ and apply $Ts_\alpha T^{-1}$ to it: $(Ts_\alpha T^{-1})(Tx) = Ts_\alpha x = Tx$. Combining the above facts it becomes apparent that when $x \in H_\alpha \Leftrightarrow Tx \in H_{T\alpha}$ □

Corollary 2.1. *In particular if* $T \in W$ *then* $s_\alpha \in W \Leftrightarrow s_{T\alpha} \in W$

Proof. Since W is a group it is closed under the group operation, so trivially $s_{T\alpha} = Ts_\alpha T^{-1} \in W$ □

It can be inferred from this proof that W permutes the set of all lines with the form L_α for each reflection in W. As pointed out in [4]; it's important to know that only the lines L_α are determined by the reflection group, not the actual vectors α. However, by taking the two unit vectors with opposite sign on these lines, the action of W will be stable on this set of vectors. The set of vectors $\{\alpha, -\alpha : s_\alpha \in W\}$ is called a root system.

Due to these lines being partitioned and this set of vectors being based on the lines obtained, the set will be stable under the action of W. Meaning that when equipped with a reflection group, a root system encapsulates some kind of geometric configuration of the reflecting hyperplanes. To keep things simple, it's possible to define axiomatically whether a set of vectors is a root system for the associated reflection group.

Definition 4. Take Φ to be a root system, if it is a non empty set of non zero vectors in V; such that the set satisfies the following axioms [4]:

R1: $\Phi \cap L_\alpha = \{\alpha, -\alpha\}$ for all $\alpha \in \Phi$
R2: $s_\alpha\Phi = \Phi$ for all $\alpha \in \Phi$

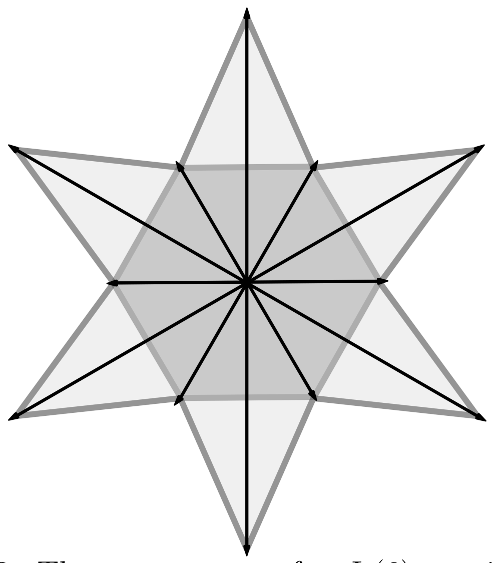


FIGURE 3. The root system for $I_2(6)$ sat inside a 6 pointed star

The above construction of a set is a root system; but when taking a set to be a root system from the axioms, there are some details that need to be considered. For instance the root system might not be know to exhaust every reflection in W, all that is required is that the axioms hold. Furthermore a root system doesn't need to be made of unit vectors, it just needs need to be stable under the action of W. In fact the root system for $I_2(6)$ fits within a 6 pointed star, with roots of varying length. Any reflection group with a finite associated root system is finite, and vice versa.

Now that root systems have been introduced as a linear algebra construction, they can be partitioned into 2 different sets. A root system contains both α and $-\alpha$, as such the next logical step is to split Φ into a "positive" and "negative" system. Label the positive system as Φ^+ and the negative system as Φ^-; where precisely one of $\{\alpha, -\alpha\}$ is contained in Φ^+, and the other one is in Φ^-.

It is important to note that the definition of positive is not so straight forward. It is not necessarily the case that all vectors on one side of the axis have the same sign. The approach taken within [4] is to rearrange the space into elements smaller and larger than 0. This uses a method called "total ordering". In order to streamline this project, a positive system will be chosen relative to a vector in V.

This method of defining a positive system comes from [3]. Chose a vector t, such that $(t, \alpha) \neq 0$ for every α in Φ. Define the positive and negative systems relative to t as:

$$\Phi_t^+ := \{\alpha \in \Phi \mid (t, \alpha) > 0\}$$
$$\Phi_t^- := \{\alpha \in \Phi \mid (t, \alpha) < 0\}$$

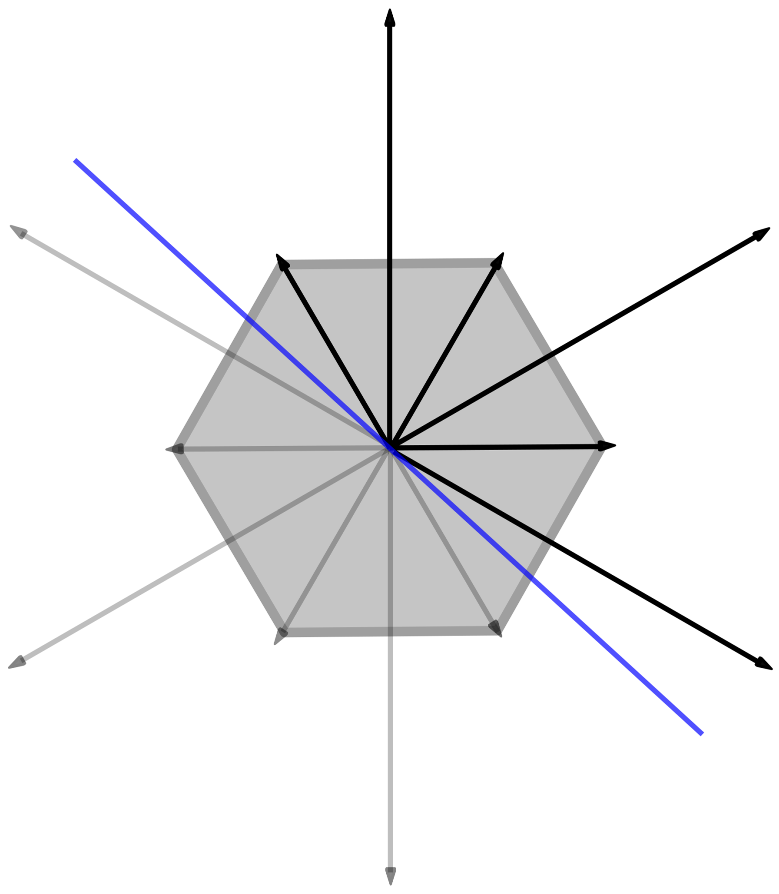


FIGURE 4. A choice of positive system for $I_2(6)$, determined by the highlighted blue vector.

Using this method, all vectors in the positive system lie on one side of the hyperplane orthogonal to t, with all the negative roots on the other side. The above figure is an example of only one of the many choices for a positive system in $I_2(6)$.

1.4. **Simple systems.** A root system is perfectly capable of reconstructing an entire reflection group, in fact within [5] root systems are called a linear algebra version of a reflection groups. A root system without an associated reflection group can construct a reflection group W by:

$$W(\Phi) = \text{The reflection group generated by : } \{s_\alpha : \alpha \in \Phi\}$$

Unfortunately the size of a root system is comparable to that of the original reflection group, and may actually be very large in comparison to the dimension of the space. take $I_2(80)$, clearly this is going to have a large root system. To make classification easier it is possible to construct a derivative system, which is smaller and easier to work with.

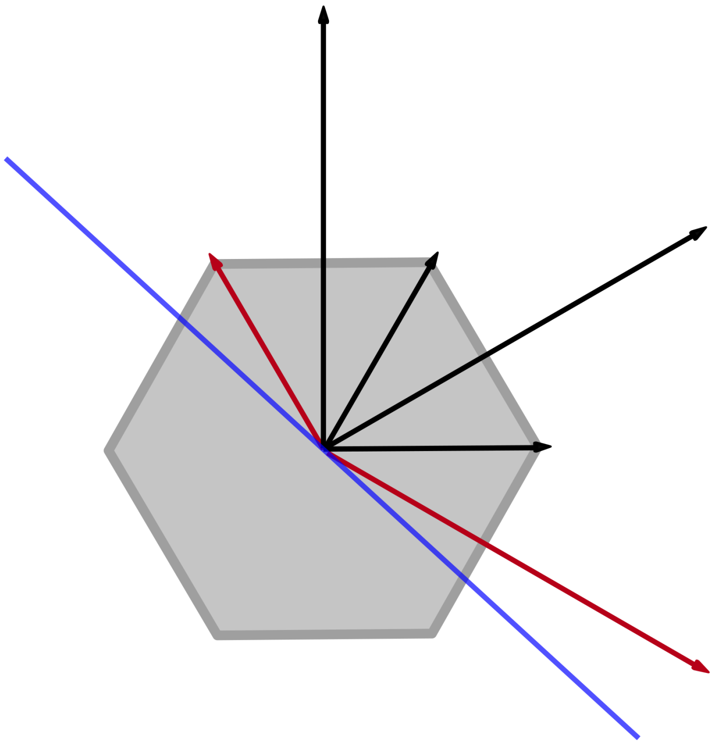


FIGURE 5. A simple system for $I_2(6)$, where the simple roots are highlighted in red. In this figure the negative roots have been omitted.

The aim of this section is to introduce a special smaller set which can generate the whole reflection group.

Definition 5. Given a root system $\Phi \subset V$ then the set $\Delta \subset \Phi$ is a simple system if it satisfies the following axioms:

- **S1:** Every element $\gamma \in \Delta$ is k linearly independent.
- **S2:** Every element $\alpha \in \Phi$ is a k linear combination of elements in Δ, where all the coefficients of that combination have the same sign.

Using sets that satisfy these axioms, the definition for positive and negative systems can be expanded to no longer rely on a vector t. Say that Φ has a simple system Δ, and take an $\alpha \in \Phi$. This element is defined to be positive, if and only if its coefficients are all non negative. From this point on; if a positive system is not defined relative to a vector, then one can assume it was defined by an axiomatically formed simple system.

There are now two types of positive system: one that has been defined relative to a vector, and another which is defined relative to a simple system. The aim of this section is to show that simple systems are a good candidate to generate and classify reflection groups. In order for this to be the case, simple systems need to be a consistent concept. This means that simple systems must satisfy the following:

- **A1:** Every simple system must generate a positive system.
- **A2:** Every positive system relative to a vector should contain a simple system, from which it can be reconstructed.
- **A3:** A chosen simple system must uniquely define a positive system and vice versa.
- **A4:** Every simple system contained within the same root system must generate the geometry of the same reflectional group.

The first requirement has already been shown, simply by choosing how the term "positive" is defined. The second and third requirements show that the relationship between simple systems and positive systems is well defined. Finally, the last requirement shows that simple systems can be used as a classification tool for reflection groups. The following lemma shows the relationship between vectors in a simple system, and is required to prove a well defined relationship between simple and positive systems.

Lemma 1. *If $\alpha, \beta \in \Delta$ where $\alpha \neq \beta$, then $(\alpha, \beta) \leq 0$*

Proof. This proof was found in [5]. Suppose for a contradiction that $(\alpha, \beta) > 0$, then the reflection s_α applied to β has the equation below:

$$s_\alpha(\beta) = \beta - 2\frac{(\alpha, \beta)}{(\alpha, \alpha)}\alpha$$

Since both α and β are in the root system $s_\alpha(\beta) \in \Phi$. Take Φ^+ to be the positive system generated by Δ, exactly one of $s_\alpha(\beta)$ or $-s_\alpha(\beta)$ is in Φ^+. Without loss of generality choose $s_\alpha(\beta) \in \Phi^+$. The assumption states that $\frac{(\alpha,\beta)}{(\alpha,\alpha)} > 0$, which means that $s_\alpha(\beta)$ is a linear combination of Δ with positive coefficients. Rearrange the equation above to form:

$$\beta = s_\alpha(\beta) + 2\frac{(\alpha, \beta)}{(\alpha, \alpha)}\alpha = \sum_i \lambda_i \gamma_i + 2\frac{(\alpha, \beta)}{(\alpha, \alpha)}\alpha = \sum_i \lambda_i \gamma_i + \lambda\alpha$$

Where γ_i are the elements of Δ with corresponding coefficients λ_i. Since β is in the simple system, label it's corresponding coefficient λ_β. If $\lambda_\beta < 1$, then β is a positive linear combination of the other elements of Δ. If $\lambda_\beta \geq 1$, then 0 is a positive linear combination of Δ. Both these scenarios cause a contradiction. $\square$

The next step is to take a root system Φ, and partition it using a vector t. A candidate simple system can then be found within Φ_t^+; by using the lemma above, the candidate system can be verified to fulfil the axioms of a simple system.

Theorem 3. *Every positive system determined by a vector contains a simple system.*

Proof. Given a root system Φ which has been partitioned by a vector t, into positive and negative systems: Φ_t^+ and Φ_t^-. Choose the smallest subset $\Delta \subset \Phi_t^+$, such that every $\alpha \in \Phi_t^+$ is a linear combination of Δ with positive coefficients. Such a set exists, this can be verified by taking Φ_t^+ and removing linearly dependent elements. Now it is possible to verify that this candidate is a simple system.

By its construction the candidate set Δ satisfies **S2**, it remains to show **S1**. In order to show this Lemma 1 can be used, as its proof only relies on the set satisfying **S2**.

For a contradiction assume that Δ is not linearly independent; this means 0 can be written as a linear combination of Δ, such that not all terms are 0. This can be written as, $\sum_i \lambda_i \gamma_i = 0$. Since $0 \notin \Phi_t^+$, the coefficients can be negative. Split the sum into the positive and negative coefficients:

$$\sum_i \lambda_i \gamma_i = \sum_i \omega_i \alpha_i - \sum_j \mu_j \beta_j = 0$$

$$\sum_i \omega_i \alpha_i = \sum_j \mu_j \beta_j$$

Where all $\omega_i, \mu_i \geq 0$ and $(\alpha_i), (\beta_j)$ are distinct subsets of Δ. Now apply the bilinear form to the left hand side, since the bilinear form is positive definite its evaluation will be bounded below by 0.

$$(\sum_i \omega_i \alpha_i, \sum_i \omega_i \alpha_i) = (\sum_i \omega_i \alpha_i, \sum_j \mu_j \beta_j) = \sum_i \omega_i \sum_j \mu_j (\alpha_i, \beta_j)$$

By applying lemma 1 each term (α_i, β_j) is less than or equal to 0. Each sum term is positive, thus the whole sum is bounded above by 0. Thus $(\sum_i \omega_i \alpha_i, \sum_i \omega_i \alpha_i) = 0$ which implies $\sum_i \omega_i \alpha_i = 0$, but this is a contradiction and thus Δ is linearly independent. $\square$

Looking at the root system for $I_2(6)$, then following the guide in the above proof a simple system can be found. Take the positive system Φ_t^+ for $I_2(6)$, which was determined in figure 4. Any two vectors on the plane which are not scalar multiples, form an interior region. Any point in this region can be written as a linear combination of the two bounding vectors, such that the coefficients of that combination are all non negative. This means that the two vectors which bound the others in their interior region form the simple system. In this case that means the two vectors closest to H_t within Φ_t^+ form the simple system.

During the above proof; when taking the smallest set to be the candidate simple system, a uniqueness is implied. This means that a simple system is also uniquely determined by it's positive system. Since a simple system uniquely defines a positive system, their relationship is well defined. This means it now also makes sense to talk about a simple system defined relative to a vector t. Take Δ_t to be the unique simple system determined by the partition of Φ into Φ_t^+ and Φ_t^-.

There is still a chance that a different choice of simple system might result in different geometric configuration of reflecting hyperplanes when reconstructing. For the sake of consistency a different choice of simple system in the same root system should generate the same geometry.

The ideal result would be that the positive systems generated by two different simple systems would be conjugate under W. Here conjugate means there is an element $T \in W$ such that $T\Phi_1^+ = \Phi_2^+$. In simple terms this means that the generated positive systems have the same geometric configuration but just at a different orientation. Checking that this is true is much easier for two systems which have been partitioned by a vector, the following lemma comes from [3].

Lemma 2. *Given a positive system generated by a vector t, and therefore a simple system Δ_t. If $T \in W$, then $T\Phi_t^+ = \Phi_{T(t)}^+$. By extension $T\Delta_t = \Delta_{T(t)}$*

Proof.

$$\begin{aligned} T\Phi_t^+ &= T\{\alpha \in \Phi_t^+\} = T\{\alpha \in \Phi : (t,\alpha) > 0\} \\ &= \{T\alpha : \alpha \in \Phi \text{ and } (T(t), T(\alpha)) > 0\} \\ &= \{\beta \in \Phi : (T(t), \beta) > 0\} = \Phi_{T(t)}^+ \end{aligned}$$ □

Reinterpreting this result; if there are two simple systems partitioned by vectors s and t, and there exists a linear map $T \in W$ such that $T(t) = s$, then the two systems are conjugate in W. However, it still remains to show that axiomatic simple systems for the same reflection group are conjugate. This is a more useful result as it covers all simple systems; since it only requires the axioms, and simple systems relative to a vector still fulfil the axioms. The following two results come from [4] and verify the consistency of all simple systems.

Lemma 3. *Let Δ be a simple system contained in a root system Φ. Let Φ^+ be the positive system generated by Δ. if $\gamma \in \Delta$ then:*

$$s_\gamma(\Phi^+ \backslash \{\gamma\}) = \Phi^+ \backslash \{\gamma\}$$

Proof. Take a $\alpha \in \Phi^+$ such that $\alpha \neq \gamma$. Write α as a linear combination: $\alpha = \sum_i \lambda_i \gamma_i$ where $\lambda \geq 0$ and $\gamma_i \in \Delta$. Because $\alpha \neq \gamma$ and the only multiple of γ in Φ^+ is γ, there must be a coefficient $\lambda_j > 0$ such that $\gamma_j \neq \gamma$. Now apply the refection s_γ to α:

$$s_\gamma(\alpha) = \alpha - 2\frac{(\gamma,\alpha)}{(\gamma,\gamma)}\gamma = \sum_i \lambda_i \gamma_i - 2\frac{(\gamma,\alpha)}{(\gamma,\gamma)}\gamma$$

Which is a linear combination of elements in Δ. Since Φ is a root system, $s_\gamma(\alpha) \in \Phi$. This implies all the coefficients have to have the same sign. Since there is a positive coefficient λ_j, all the coefficients for $s_\gamma(\alpha)$ are ≥ 0. Thus $s_\gamma(\alpha) \in \Phi^+$.

Finally $s_\gamma(\alpha) \neq \gamma$ this is because it would result in the following contradiction:

$$\alpha = s_\gamma(s_\gamma)\alpha = s_\gamma(s_\gamma(\alpha)) = s_\gamma(\gamma) = -\gamma$$

Which is a contradiction because $-\gamma \notin \Phi^+$. This means that s_γ maps $\Phi^+ \backslash \{\gamma\}$ into itself by a linear map and hence injectively. The conclusion being that the mapping is hence onto and thus, $s_\gamma(\Phi^+ \backslash \{\gamma\}) = \Phi^+ \backslash \{\gamma\}$. □

Theorem 4. *Any two positive systems which are generated by some simple systems in Φ are conjugate under W, and thus so are the simple systems themselves.*

Proof. Let Φ_1^+ and Φ_2^+ be the positive systems generated by simple systems which are both in W. Take r to be:

$$r = |\Phi_1^+ \cap \Phi_2^-|$$

Where $|\cdot|$ represents the cardinality of a set. If $r = 0$ then obviously $\Phi_1^+ = \Phi_2^-$, and the sets are trivial conjugate. However if $r > 0$, then there are some vectors $\alpha \in \Phi_1^+$ which are not in Φ_2^+ because they are in Φ_2^-. This implies that Δ_1 cannot be contained entirely within Φ_2^+. Choose such an $\alpha \in \Delta_1$ such that $\alpha \in \Phi_2^-$.

Using Lemma 2 $s_\alpha(\Phi_1^+)$ is the same set except $s_\alpha(\alpha)$ is mapped to $-\alpha$. Since $\alpha \in \Phi_2^-$, $-\alpha \in \Phi_2^+$. Thus:

$$|s_\alpha(\Phi_1^+) \cap \Phi_2^-| = r - 1$$

Thus by induction an element $w = s_1, ..., s_n \in W$ is created such that $w\Phi_1^+ = \Phi_2^+$. And as such the two positive systems are conjugate, the same for their corresponding simple systems. □

In summary, The geometry generated by simple systems is invariant under the choice of simple system. This means that they are well behaved, seeming as they are a much smaller than the original system it makes sense to study reflection groups through their simple systems. The next step in this process is to make sure that a simple system for a reflection group can reconstruct the group itself.

Definition 6. Given an $\alpha \in \Phi$ which for a fixed simple system Δ can be uniquely decomposed into $\sum_i \lambda_i \gamma_i$, with $\gamma_i \in \Delta$ and λ_i of all the same sign. call the sum : $\sum_i \lambda_i$ the height of α relative to Δ.

Theorem 5. *Given a fixed simple system Δ, then the reflection group W is generated by reflections $\{s_\alpha : \alpha \in \Delta\}$.*

Proof. The strategy for this proof is to take G to be the subgroup of W which was generated by the above strategy, and then show that they are in fact the same group.

Suppose $\alpha \in \Phi^+$, consider the set $G\alpha \cap \Phi^+$. This set is not empty, because α is in $G\alpha$ due to the identity element being in G. Chose an element β in this set such that β has the smallest possible height, and claim that this implies $\beta \in \Delta$. Write $\beta = \sum_i \lambda_i \gamma_i$, where this is the standard decomposition by simple reflections in Δ. and now note that:

$$0 < (\beta, \beta) = (\beta, \sum_i \lambda_i \gamma_i) = \sum_i \lambda_i (\beta, \gamma_i)$$

This forces $(\beta, \gamma_i) > 0$ for some $\gamma_i \in \Delta$. If $\beta = \gamma_i$ then $\beta \in \Delta$ and the claim is proved. Consider otherwise that $\beta \neq \gamma_i$. Look at the root $s_\lambda(\beta)$, since $\beta \in \Phi^+$ by Lemma 2 this root is positive. $s_\lambda(\beta) = \beta - 2\frac{(\beta,\lambda)}{(\lambda,\lambda)}\lambda$. Which means a positive multiple of λ has been subtracted from β, and thus the height of $s_\lambda(\beta)$ is less than the height of β. Since $s_\lambda \in G$ then $s_\lambda(\beta) \in G\alpha$, but this contradicts β being the element of smallest height. As such $\beta \in \Delta$.

We can use the above part of the proof to imply some kind of correspondence between the root systems. Claim that $G\Delta = \Phi$. In the claim above, it was shown that for any positive element $\alpha \in \Phi^+$ then $G\alpha$ meets Δ. Take the element $(s_1, ..., s_n) \in G$ generated by simple reflections, such that $(s_1, ..., s_n)\alpha = \gamma$ for some $\gamma \in \Delta$. repeatedly rearrange these elements:

$$\begin{gathered}
(s_1, ..., s_n)(\alpha) = \gamma \\
(s_1)(s_1, ..., s_n)(\alpha) = s_1(\gamma) \\
(s_1(s_1))(s_2, ..., s_n)(\alpha) = s_1(\gamma) \\
(s_2, ..., s_n)(\alpha) = s_1(\gamma) \\
... \\
\alpha = (s_n, ..., s_1)(\gamma)
\end{gathered}$$

clearly $(s_n, ..., s_1) \in G$ as it is generated by simple reflections. For any positive element $\alpha \in \Phi^+$, then $\alpha \in G\Delta$. Thus $\Phi^+ \subset G$. However, take any negative element $\alpha \in \Phi^-$. This implies that $-\alpha \in \Phi^+$, and thus there exists a $w \in G$ such that $-\alpha = w(\gamma\ )$ for some $\gamma \in \Delta$. Since w is a linear map, the negative sign can be moved inside w. This means that: $\alpha = w(-\gamma) = w(s_\gamma(\gamma)) = (ws_\gamma)(\gamma)$. Since $ws_\gamma \in G$, any $\alpha \in \Phi^-$ is also contained in $G\Delta$. combining these results $\Phi \subset G\Delta$.

Finally combine these results into the proof of the theorem statement. Take any generator s_α of the reflection group W. $\alpha \in \Phi$ so write $\alpha = w\gamma$ for some $w \in G$ and $\gamma \in \Delta$. Using Theorem 2, $s_\alpha = s_{w\gamma} = ws_\gamma w^{-1} \in G$. Meaning every generator of W is in G, since G is a subgroup of W this implies that $W = G$. □

1.5. **Categorising reflection groups.** A simple system has finally been verified to be the special generating set for any reflection group, thus if one wanted to categorise reflection groups, using simple systems would be the right way to go. In fact simple systems lead to a level of abstraction above reflection groups, into a much more general set of groups.

Definition 7. A group W with a generating set $S \subset W$ such that :

$$W = \langle s_\alpha \in S \mid (s_\alpha s_\beta)^{m(\alpha,\beta)} = 1 \rangle$$

is called a Coxeter group. Where $m(\alpha, \beta)$ denotes the order of the product of reflections $s_\alpha s_\beta$. This is defined to be the smallest integer n such that $(s_\alpha s_\beta)^n = 1$.

One can see the relationship between these groups and reflection groups, the generating set S is the equivalent of simple systems. Coxeter groups and reflection groups are very closely related, so close that the set of finite reflection groups is exactly the set of finite Coxeter groups. Despite being very closely related, this project does not focus on Coxeter groups, the main purpose of introducing them is to use their convenient presentation.

Definition 8. Given a reflection group W with a simple system Δ. Construct a graph such that each vertex represents a single vector in the simple system. Any two distinct vertices α, β are joined by a line if $m(\alpha, \beta) \geq 3$. This graph is called the Coxeter graph for the reflection group W.

Taking the dihedral groups as an example, all planar dihedral groups have a simple system with only two elements. Since the polygon has n vertices, ideally the composition of the two simple roots would generate a rotation which returns to the starting point after n iterations. In terms of a Coxeter graph this means their order would be n, thus the two roots are joined by an arc labelled n.

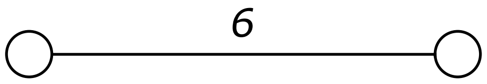


Figure 6. Coxeter graph for $I_2(6)$

The order of two roots $m(\alpha, \beta)$ shows the relationship between these two vectors. The angle between the two roots can be reconstructed from this function, and thus from the graph. If $m(\alpha, \beta) = n$ then the angle between α and β is $\pi - \frac{\pi}{n}$. Thus the graph for $I_2(6)$ says it's two simple roots have an internal angle of $\frac{5\pi}{6}$, which is incredibly useful for reconstructing a simple system from a Coxeter graph.

Any distinct vertices which do not have a line connecting them are said to have an order of 2, this results in an angle of $\frac{pi}{2}$ meaning the mirrors are orthogonal and do not affect each other. Finally if Coxeter graphs uniquely determine a reflection group, then the study of reflection groups can be reduced to the study of Coxeter graphs.

Theorem 6. *Any two reflection groups with the same Coxeter graph are conjugate in V.*

Proof. Let W_1 and W_2 be two Reflection groups with the same Coxeter graph, such that they have simple systems Δ_1 and Δ_2 respectively. Since simple systems for the same reflection group are conjugate, the choice of either simple system does not matter. Write Δ_1 as $\{\gamma_1, .., \gamma_n\}$, and Δ_2 as $\{\alpha_1, ..., \alpha_n\}$.

What does it mean for the simple systems if their reflection groups have the same graph? It means that the relationship between any two roots are the same, specifically the angle between them. Since the bilinear form determines the angle between vectors, by having the same angle between them they have the same bilinear form. Written explicitly this can be interpreted as: $(\gamma, \gamma_j) = (\alpha_i, \alpha_j)$ for every i and j in the range $[1, n]$.

Define a linear map $T : V \to V$ such that $T\gamma_i = \alpha_i$ for all i, by using linearity extend this definition to the whole space. Now according to theorem 2, $Ts_{\gamma_i}T^{-1} = s_{T(\gamma_i)} = s_{\alpha_i}$. Since every root in the simple systems are conjugate, and the reflection groups which are generate by those conjugate simple systems, are themselves conjugate. □

2. Hyperbolic Geometry

This section serves as a very brief introduction into the world of hyperbolic geometry, the intention of this is not to rigorously explore and prove results about hyperbolic geometry; instead, the aim of this section is to be able to use and understand hyperbolic space for the use of reflection groups.

2.1. **Introduction to non-Euclidean geometry.** Euclid introduced his concepts for geometry using an axiomatic system; however, there was one axiom that caused a bit of contention. This axiom was referred to as "Euclid's parallel postulate", which in modern language states that the internal angles for a triangle must add up to exactly 180 degrees. This axiom caused a problem as it did not seem to be independent of the others, by attempting to ignore this axiom a brand new breed of geometry was created.

There are two different directions this axiom could be taken, either the internal angles of a triangle sum to greater than or less than 180 degrees. Spherical geometry comes from the internal angles summing to greater than 1800 degrees, where has hyperbolic geometry has its origins in the opposite case. This implies that these two types of geometry are opposite sides of the same coin, within [6] this is referred to as "spherical-hyperbolic duality".

Since spherical geometry takes place on the surface of a sphere with a real radius, the duality of hyperbolic and spherical space continues into the definition of the plane; which has some imaginary radius.

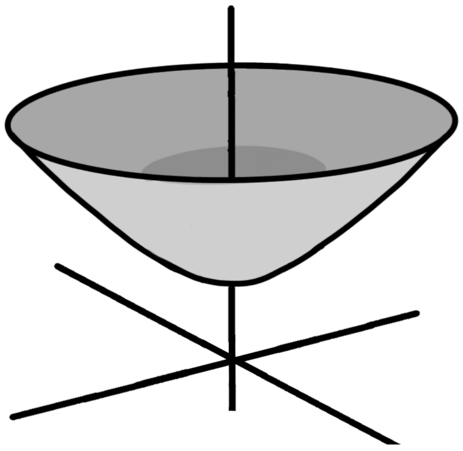


FIGURE 7. Hyperbolic 2-space sitting in $\mathbb{R}^3$.

2.2. **Hyperbolic plane.** In order to first define a hyperbolic plane, an inner product must be introduced.

Definition 9. Given two vectors x and y in $\mathbb{R}^n$ and define the Lorentzian inner product as:

$$x \circ y := x_1y_1 + x_2y_2 + ... - x_ny_n$$

Strictly speaking exactly one of these indices needs to be negative, but it does not have to be the last one. This inner product will act as the symmetric bilinear form for hyperbolic space. Earlier it was noted that a hyperbolic space was a sphere of imaginary length, that length function is defined using the inner product.

$$||x|| = i$$
$$(x \circ x)^2 = (-1)^2$$

Thus hyperbolic n space can be defined as all the points in x in $\mathbb{R}^{n+1}$, such that $x \circ x = -1$. However, this results in two sheets in hyperbolic space; therefore take hyperbolic n-space to be:

$$H^n : \{x \in \mathbb{R}^{n+1} \mid x \circ x = -1 \text{ and } x_{n+1} > 0\}$$

Throughout this project, the exploration of hyperbolic space will be restricted to the hyperbolic plane, which from this point onwards will be referred to as the hyperboloid.

2.3. **Hyperbolic reflections.** In order for reflection groups to be valid in hyperbolic space, it would be required that hyperbolic reflections don't map any points in hyperbolic space to outside that space.

Theorem 7. *H^n is closed under hyperbolic reflections.*

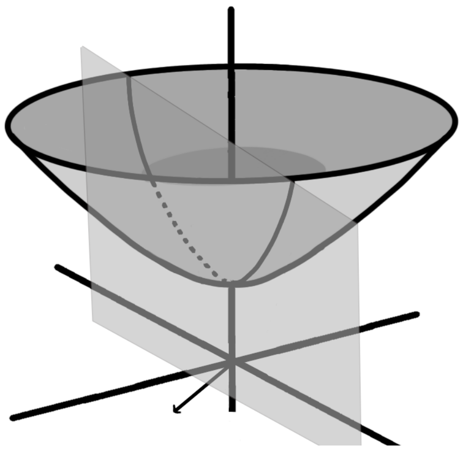


FIGURE 8. Hyperplane fixed by a hyperbolic reflection.

Proof. Let x be in H^n, which means $x \circ x = -1$. and check if $s_\alpha x \circ s_\alpha x$ is equal to -1.

$$\begin{aligned}
s_\alpha x \circ s_\alpha &= (x - 2\frac{(x,\alpha)}{(\alpha,\alpha)}\alpha) \circ (x - 2\frac{(x,\alpha)}{(\alpha,\alpha)}\alpha) \\
&= (x - 2\frac{x \circ \alpha}{\alpha \circ \alpha}\alpha) \circ (x - 2\frac{x \circ \alpha}{\alpha \circ \alpha}\alpha) \\
&= x \circ x - 4\frac{x \circ \alpha}{\alpha \circ \alpha}(x \circ \alpha) + (2\frac{x \circ \alpha}{\alpha \circ \alpha})^2 \alpha \circ \alpha \\
&= x \circ x - 4\frac{(x \circ \alpha)^2}{\alpha \circ \alpha} + 4\frac{(x \circ \alpha)^2}{\alpha \circ \alpha} \\
&= x \circ x = -1
\end{aligned}$$

□

This means that hyperbolic space is closed under the action of any composition of reflections, which means it is valid to explore reflection groups in hyperbolic space. Restricting to just the hyperboloid; hyperplanes fixed by reflections are just hyperbolic lines, which are the hyperboloid intersected with the 2 dimensional Euclidean plane which is orthogonal to the root of the reflection.

Take a reflection which acts on hyperbolic 2 space, such that the root of this reflection has a non zero z component. The hyperplane which is fixed by this reflection does not intersect the hyperboloid at the base. This means that the mirror associated with this reflection will appear to be off centre, but this is still a valid reflection as all roots are valid. This immediately shows that there is a difference between hyperbolic and euclidean reflection groups.

Although a very brief introduction to hyperbolic geometry, this is actually all that is required for this project. However, if the reader would like a deeper dive into non Euclidean geometry, it can be found in [6].

3. Rendering hyperbolic imagery

3.1. **introduction.** Throughout this project the aim has been to generate some kind of image to represent a reflection group. To meet this goal a programming approach has been chosen. In order to create the diagrams, thousands of data points need to be explored and classified. This means that speed and efficiency is paramount, to this end C++ has been selected.

An old file format called .ppm has been selected for this project. Despite the format being old, it is very easy to write to a file. This simplifies the process of outputting an image. The next goal is to test drawing a circle, which would represent an empty Poincaré disk. Simply take every single point in the pixel array and check if that point is within the radius of the circle, then set the appropriate pixels to grey.

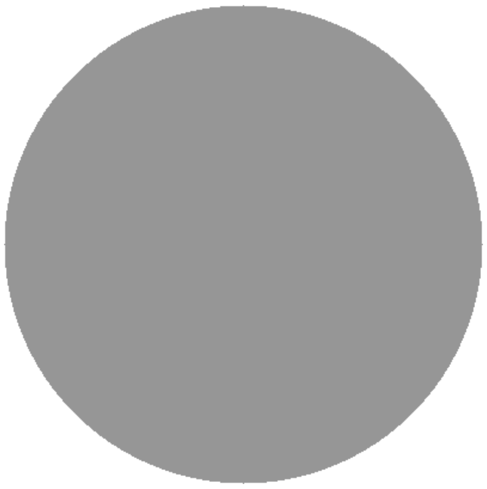


Figure 9. Empty Poincaré disk

```cpp
//width and height of image based of radius of unit disk
int w = radiusInPixels * 2 + 1;
ppmInit(w, w);
for (int y = -radiusInPixels; y < radiusInPixels + 1; y++)
{
    for (int x = -radiusInPixels; x < radiusInPixels + 1; x++)
    {
        //check we're in the circle
        if (x * x + y * y <= radiusInPixels * radiusInPixels)
        {
            ppmSetPixel(newColour(150, 150, 150), x, y);
        }
    }
}
//write the pixel data
ppmWritePixelData();
```

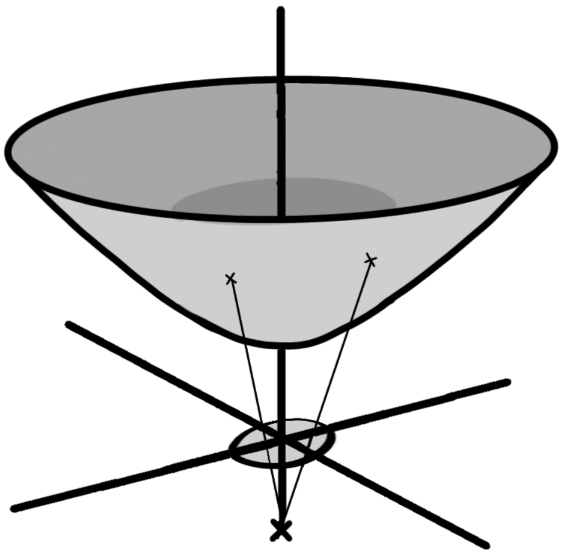


FIGURE 10. Diagram of a identifing points to the hyperboloid.

The above code also shows the implementation of .ppm writing. The method *"ppmInit"* simply initialises an array of colours on the stack. Furthermore *"ppmSetPixelData"* places the origin of the image at the centre, this means the conversion from Euclidian coordinates to pixel coordinates doesn't need to be handelled outside the writing method. Finally to actually write the pixel data to a file the *"ppmWritePixelData"* method is called.

3.2. **Poincaré projection.** The Poincaré disk takes a point on the unit disk and uniquely maps it to a point on the hyperboloid. Take a point on the unit disk (x, y) and draw a line that passes through that point and $(0, 0, -1)$. This line will have equation $L = (0, 0, -1) + (x, y, 1)t$, and solve for t. Since the equation of the hyperboloid is $\|(x, y, z)\|_L = -1$.

$$\|L\|_L = \|(0,0,-1) + (x,y,1)t\|_L = \|(xt, yt, t-1)\|_L = -1$$

$$\begin{aligned}(xt)^2 + (yt)^2 - (t-1)^2 &= -1 \\ (x^2+y^2)t^2 - (t^2 - 2t + 1) &= -1 \\ (x^2+y^2-1)t^2 + 2t &= 0 \\ t[(x^2+y^2-1)t + 2] &= 0\end{aligned}$$

Solving for this equation the end result is $t = 0$ or $t = -\frac{2}{x^2+y^2-1}$. The first case comes from the base point resting on the lower sheet of the hyperboloid. The second value for t is where the line encounter the upper sheet, which is the correct corresponding point.

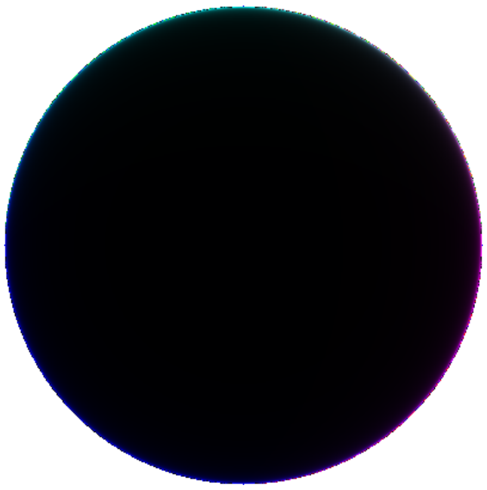


FIGURE 11. Hyperboloid with (x, y, z) coordinates casted into colour

```
point poincareRayTrace(float x, float y)
{
    long double a = x * x + y * y - 1;
    //b = 2 :: c = 0
    //solve quadratic equation for t
    long double t = -2 / a;
    //now evaluate the line
    return newPoint(t * x, t * y, t - 1);
}
```

The entire representation of the hyperbolic plane can be found, by applying this method to every pixel which has been found on the Poincaré disk. This is very similar to ray tracing algorithms.

Now it is important to talk about how data will be stored within this project. A data structure called *"dualPoint"* has been created. This data type stores the coordinates of the hyperboloid and the associated pixel coordinate. This data type is chosen to allow the classification to take place in hyperbolic space, without needing to recalculate which pixel it represents.

The next step is to see if a hyperboloid has accurately been captured. For every point in the hyperbolic data, write the red, green and blue values of a pixel to be the x, y, z values of the corresponding point on the hyperboloid. The following image results.

The change in colour is very hard to spot on such a diagram. By zooming in, the colours around the edges of the disk become more apparent. This is because the distance to the hyperboloid becomes exponentially larger as a point approaches the edge of the disk. As a result, the colours towards the centre of the disk appear to hardly change in comparison to the edge cases.

FIGURE 12. Zoomed in example of the above hyperboloid

Instead of wasting the majority of colours on the edge of the disk, it is possible to normalise the colours so that the change is more apparent. In order to normalise these points the sigmoid function is necessary, take $\sigma : (-\infty, \infty) \to (0, 1)$ such that:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Now this time, the diagram can be rendered again with a more uniform colour distribution. Take any point p such that p is on the unit disk, with h being the corresponding point on the hyperbola. The colour of the pixel at point p has rgb value:

$$r = 255 \times \sigma(h.x)$$
$$g = 255 \times \sigma(h.y)$$
$$b = 255 \times \sigma(h.z)$$

There is some odd artefacts around the edge of the disks in both images. This is caused by the incredibly large numbers on the edge of the disk. When being raised to an exponential there are some underflow and overflow concerns. So to solve this issue, add some error checks into the sigmoid function

Thankfully C++ has a method to check for this kind of underflow and overflow errors. The check is within the standard library and is called through *"std::isnan"* where nan stands for "Not a Number".

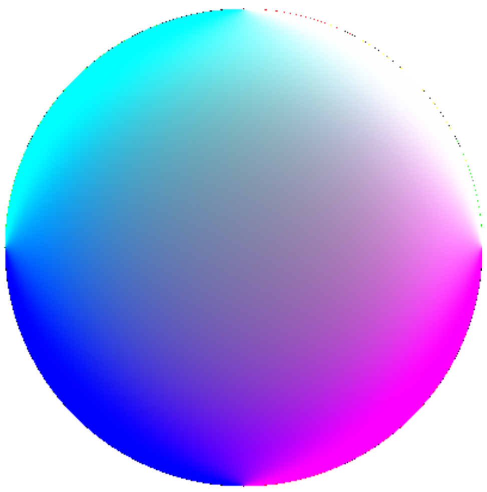


FIGURE 13. Hyperboloid with normalised colours

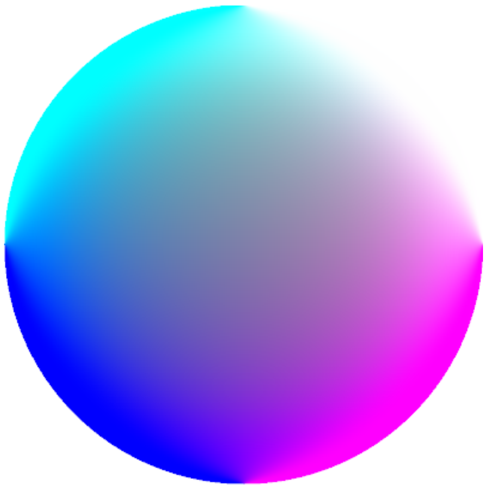


Figure 14. Hyperboloid with error handling

```
long double sig(long double x )
{
    long double r = 1 / (1 + exp2l(-x) );
    if (std::isnan(r))
    {
        return sign(x) * 1;
    }
    else
    {
        return r;
    }
}
```

The edge case errors have been dealt with, and as such the result the image is a smoothed. With a bit of thought, this diagram does show that the hyperboloid has been modelled accurately. For instance the top left of the diagram tends towards a white colour, this means that (x, y, z) tends towards (∞, ∞, ∞). as the diagram tends to the top right.

3.3. **Fundamental domains.** All the points on the hyperbolic plane have been collected and identified to a pixel. The next step is to take a reflection group and create an image which has some mathematical interpretation. The concept that generates the most coherant imagery is fundamental domains. There are many different appraoches to defining a fundamental domain. The following definition comes from [1].

Definition 10. Take a topological space V with a group W acting on it. A fundamental domain is an open subset $F \subset V$ such that F contains exactly one point from each orbit of the action of W.

For the purposes of this project, the definition can be rewritten. Within [3] a fundamental domain with respect to a reflection group W is defined to be a set which satisfies the following:

F1: F is open

F2: $F \bigcap TF = \emptyset$ for all $T \in W$ such that T is not the identity.

F3: $V = \bigcup\{\overline{TF} : T \in W\}$.

Where the line denotes the closure of the set. The aim of this chapter is to come up with an easier way of explicitly describing a fundamental domain for W. Start with a reflection group W, and find a simple system for that group. The easiest way is to find a simple system is to generate one relative to a vector t, label this simple system as Δ_t. Denote Δ_t^* to be the dual basis of Δ_t in V, if $\Delta_t = \{\gamma_1, ..., \gamma_n\}$ then $\Delta_t^* = \{\beta_1, ..., \beta_n\}$ is defined to be the set such that $(\gamma_i, \beta_j) = \delta_{ij}$ for all i, j, where δ_{ij} is defined as:

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j. \\ 0, & \text{Otherwise.} \end{cases}$$

Such a secondary set exists and emerges due to the construction of simple systems. Create a subset of V that depends on the dual basis, and thus also dependant on the vector t.

$$F_t = \{x \in V \mid x = \sum_{i=1}^{n} \lambda_i \beta_i, \text{ such that } \lambda_i > 0\}$$

Computing the dual space for a simple system is extra work, as such the next step is to remove the β terms. Take every $x \in F_t$, then compute the bilinear form of x and γ_j. $(x, \gamma_j) = \sum_{i=1}^{n} (\lambda_i \beta_i, \gamma_j) = \lambda_j(\beta_j, \gamma_j) = \lambda_j$

Using the expressions for the coefficients x can be rewritten as $x = \sum_{i=1}^{n} (x, \gamma_i)\beta_i$. The logic for the set F_t can be reformulated into the following: if $x \in V$ and $(x, \gamma_i) > 0$ for all $\gamma_i \in \Delta$, then $x \in F_t$, as such F_t can be redefined as the following.

$$F_t = \bigcap_{i=1}^{n} \{x \in V : (x, \gamma_i) > 0\}$$

Theorem 8. *F_Δ is a fundamental domain for W.*

Proof. This proof is based of the one found within [3], each condition for a set to be a fundamental domain will be checked in turn, starting with the first one. **F1:** Each set $\{x \in V : (x, \gamma_i) > 0\}$ is the points on one side of the hyperplane fixed by s_{γ_i}, as such is an open set. F_Δ being an intersection of finite open sets is itself open.

F2: Take $T \in W$ to be a non identity transformation. Assume for a contradiction that $F_t \cap TF_t$ is not empty, and let x be in $(F_t \cap TF_t)$. Since x is F_t this implies that $(x, \gamma_i) > 0$ for all $\gamma_i \in \Delta_t$. Consider the positive system Φ_t^+ which was generated by Δ_t, the linearity of the bilinear form implies that $(x, \alpha) > 0$ for every $\alpha \in \Phi_t^+$. This implies that the positive system relative to t and x are the same: explicitly that $\Phi_t^+ = \Phi_x^+$. Now consider $x \in TF_t$ implies that $T^{-1} \in F_t$, by a similar argument $\Phi_t^x = \Phi_{T^{-1}x}^+$. Equating these two positive systems and applying lemma 2, one reaches the result: $T\Phi_x^+ = \Phi_x^+$, and the only transformation in W that allows this is the identity map, therefore a contradiction has been reached

F3: Take any $x \in V$ and construct a linear map $T \in W$ such that $(Tx, \alpha) \geq 0$ for every Δ_t. Informally the existence of this function can be seen through the reorientation of the simple roots to enclose x, however [3] has a more rigorous proof for this statement. Thus by inverting this transformation x can be placed into a fundamental domain: $x \in T^{-1}(\overline{F_t}) = \overline{(T^{-1}F_t)}$. Since any generic $x \in V$ can be placed into a fundamental domain the whole space is the union of these domains. $\square$

A fundamental domain can be interpreted as all of the points in V, such that they are inside space bounded by the hyperplanes H_γ for every γ in the simple system. Now that F_t has be confirmed to be a fundamental domain, it can now be found using code. For this project the user will be required to enter the the simple system; from there the program will compare every point on the hyperbolic plane, and if that point is in F_t then it will be added to the set which represents a fundamental domain.

For example, take a simple system $\Delta = \{(1,0,0),(0,1,0)\}$. A fundamental domain for this system will be all the points on the hyperboloid which have a positive x and y component.In the figure below, the domain will be highlighted in dark blue, and all the other points will maintain their dummy rainbow colouring.

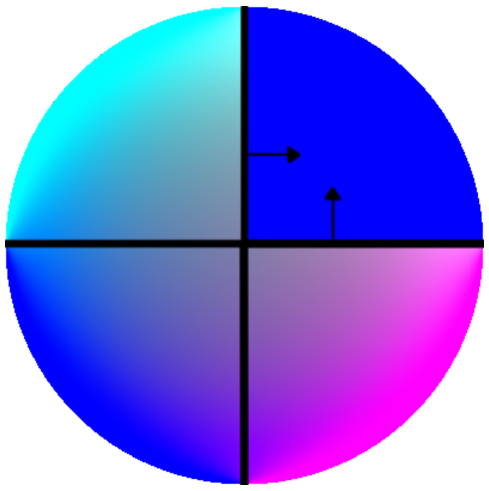


Figure 15. One fundamental domain with the hyperplanes for the simple system labelled.

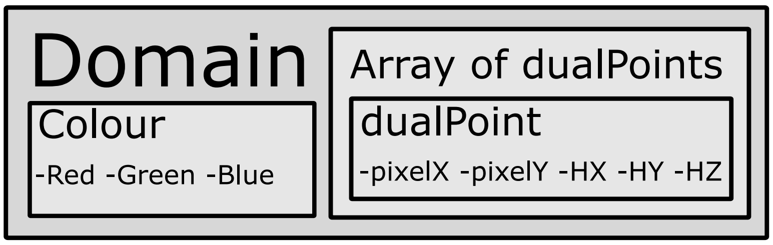


FIGURE 16. Data structure implemented within this program

Before displaying the code used to find a fundamental domain, it is important to talk about how data is stored and used within this program. A "dualPoint" contains both the coordinate of a point on the hyperboloid, and the coordinate of the pixel that this point is associated with. A domain is an array of dualPoints with a colour associated, thus a fundamental domain is stored in memory as a "Domain" data type.

```
domain findFundamentalDomain(domain& space,
                             std::vector<point> simpleSystem)
{
    domain newDomain;
    newDomain.c = newColour(0, 0, 0);
    //for every point in the space
    for (size_t i = 0; i < space.points.size(); i++)
    {
        bool inDomain = true;
        //for each point in the simple system
        for (size_t j = 0; j < simpleSystem.size(); j++)
        {
            if (!(lInnerProd(space.points.at(i).H,
                             simpleSystem.at(j)) >= 0))
            {
                //the point isnt in this domain
                inDomain = false;
                break;
            }
        }
        //got here so the point is in the space
        if (inDomain)
        {
            newDomain.points.push_back(space.points.at(i));
            //remove this point so it doesn't get classified again
            space.points.erase(space.points.begin() + i);
            i--;
        }
    }
    return newDomain
}
```

Finally the code to classify a fundamental domain can be summarised as: comparing all the points on the hyperboloid to the hyperplanes fixed by a simple system, and then assigning that domain a unique colour. The domain is then drawn by looping over every dualPoint in the domain and placing the domains colour at the pixel coordinate $(\text{pixel}X, \text{pixel}Y)$.

Now that one fundamental domain can be found, the rest of the space can be partitioned into fundamental domains. The definition of fundamental domains actually hints at the method to finding all of them: this is done by taking a domain and applying a linear transformation to it, this transformation must come from the reflection group generated by the simple system. However it is not possible to take the already found domain and actually transform the points, this is because can easily miss miss the points on the hyperboloid which relate to a pixel, which would leave gaps in the image. The actual solution is to transform the simple system.

Continuing from the previous example, find the first fundamental domain for the reflection group using $\Delta = \{(1,0,0),(0,1,0)\}$ as its simple system. To find the second fundamental domain apply a linear transformation to every element of that simple system. In this case the linear transformation is just the reflection with root $(0,1,0)$, meaning the new domain should be flipped along the y axis. Give each domain a different colour and the file outputted is the figure below.

The whole point of this program is to automate the process of finding these domains, so requiring the user to create the appropriate linear transformation is not desired, and so the program should automate this. The automation of finding the correct linear map will result in full classification the of the plane, and as such will be handled in the next subsection.

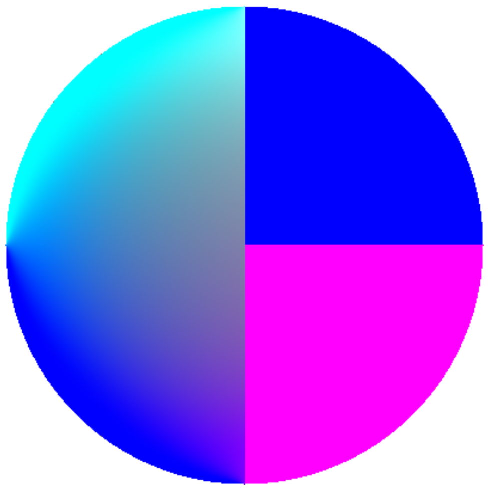


FIGURE 17. Two fundamental domains drawn with different colours.

```
//Find two fundamental domains for example simple system
//space is the points which represent the points on the hyperboloid
domain d1 = findFundamentalDomain(space, simpleSystem);
//find the next one by reflecting the simple system by
//one of the simple reflections
std::vector<point> newSystem;
for (size_t i = 0; i < simpleSystem.size(); i++)
{
    newSystem.push_back(evaluateHReflection(
                simpleSystem.at(i), simpleSystem.at(0)));
}
domain d2 = findFundamentalDomain(space, newSystem);
//write these to file
//....
```

3.4. **Tessellation.** The linear transformations which are in W are all compositions of reflections, which have roots in the chosen simple system for W. Thus by repeatedly composing reflections, all of the linear transformations can be found.

The easiest way to find all of the linear transformations in W is to use a branching tree structure. Where each node of the tree represents a found domain. A new transformation is found by composing the current node's transformation with any one of the reflections in the simple system. When finding a new linear transformation, applying the same reflection as the previous layer obviously undoes the reflection, and thus should be avoided.

Rather than store the linear transformation, this tree's nodes will hold the simple system after being transformed, this is to save computation time. In order to find the next layer of nodes: locate every single node on the bottom layer of the tree, create n new sub nodes for each node in that layer. These sub nodes are formed by applying each reflection in the original simple system to that node's internal simple system. After a subnode has a simple system, the corresponding fundamental domain is then found. If a subnode's fundamental domain is empty, then the subnode is abandoned.

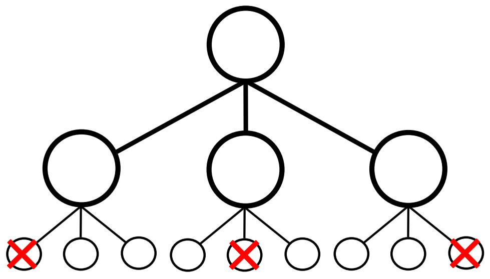


FIGURE 18. Branching structure of finding all linear transformations

A tree is represented by a vector of nodes; the first fundamental domain found by the program is always stored in the first slot, this is so the original simple system can be always be found for creating new layers. The code for putting one new node on the tree as well as creating the next layer on the tree has been included below.

```cpp
void putNodeOnTree(domain& space, std::vector<point> system,
          std::vector<Node>& tree, Node& parent)
{
    //Create the domain for this node
    domain d = findFundamentalDomain(space, system);
    //don't push this node if the domain has no points
    if (d.points.size() == 0)
    {
        return;
    }
    //Parent now needs to gain a child
    Node n;
    n.layer = parent.layer + 1;
    n.d = d;
    n.simpleSystem = system;
    n.parent = &parent;
    parent.children.push_back(&n);
    tree.push_back(n);
}

void addLayerToTree(domain& space, std::vector<Node>& tree, int x)
{
    std::vector<point> simpleSystem = tree.at(0).simpleSystem;
    std::vector<point> currentystem = simpleSystem;
    std::vector<Node> nodes = nodesOnLayer(tree, x);
    //for each node : i
    for (size_t i = 0; i < nodes.size(); i++)
    {
        //create n different simple systems for the ith node,
        //ie ith node can have n children
        for (size_t n = 0; n < simpleSystem.size(); n++)
        {
            //create the new simple system for this node
            for (size_t k = 0; k < simpleSystem.size(); k++)
            {
                currentystem.at(k) = evaluateHReflecion(
                    nodes.at(i).simpleSystem.at(k), simpleSystem.at(n));
            }
            putNodeOnTree(space, currentystem,
                      tree, nodes.at(i));
        }
        // erase the simple system to save space
        nodes.at(i).simpleSystem.clear();
    }
}
```

Since some reflection groups have infinite linear transformations, it is possible that this algorithm will find new fundamental domains forever. To combat an infinite loop, the user must specify the maximum number of layers that the tree can iterate through.

The process for finding all the fundamental domains of a simple system can be implemented in one function. Create the domain which corresponds to the original simple system, and create a tree with this domain as the starting node. Once a tree has been created, add as many layers onto the tree as the user specifies.

Classifying the entire hyperboloid with finite reflection group takes a finite number of iterations on the tree, computing extra layers after full classification does not have any negative affects as the unclassified space is empty and so the tree will exit quickly. For an infinite reflection group the hyperboloid can never be entirely classified, and as such the the maximum number of iterations specified by the user will relate to the level of detail around the edges of the Poincaré disk.

```cpp
std::vector<domain> findAllDomains(domain& space,
    std::vector<point> simpleSystem, unsigned int iterations)
{
    //create the first element
    domain first = findFundamentalDomain(space, simpleSystem);
    Node n;
    n.d = first;
    n.layer = 0;
    n.simpleSystem = simpleSystem;
    //create the tree
    std::vector<Node> tree;
    tree.push_back(n);
    //create the as many new layers as specified
    for (size_t i = 0; i < iterations; i++)
    {
        std::cout << "We're on layer :" << i << std::endl;
        addLayerToTree(space, tree, i);
    }
    //turn tree into domains
    std::vector<domain> domains;
    for (size_t i = 0; i < tree.size(); i++)
    {
        domains.push_back(tree.at(i).d);
    }
    return domains;
}
```

To generate an example let W be a reflection group with chosen simple system $\Delta = \{(-2, 0, -0.9), (1, 1, -0.2)\}$. This reflection group was chosen because it demonstrates some nice hyperbolic lines, and has a finite number of fundamental domains. Each new layer classifies more fundamental domains and colours them a brighter shade of purple.

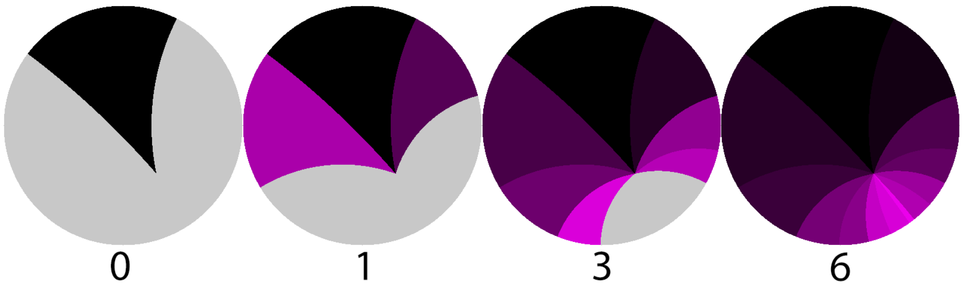


FIGURE 19. Diagram of all the domains found after a different number of tree iterations

Using the Coxeter diagram, it is possible to construct the fundamental domains for the example used throughout this project which is $I_2(6)$. The Coxeter diagram has 2 vertices, and thus 2 vectors in the simple system. Since the two vertices are connected by a weighting of 6, the angle between the two roots is $\frac{5\pi}{6}$. The orientation of this system does not matter, so simply choose one vector to be $(1, 0, 0)$. The other vector will be a unit vector which is at an angle of $\frac{5\pi}{6}$ radians from the first vector, since the z coordinate is 0 the other vector is $(cos(\frac{5\pi}{6}), sin(\frac{5\pi}{6}), 0)$.

The simple system can be entered into the program's source code. When setting the maximum number of iterations, a minimum of 5 iterations will be required to generate the the full diagram.

This system doesn't have a hyperbolic component and thus there isn't a hyperbolic line on the diagram, so what happens when a vector is altered to have a non zero z component?

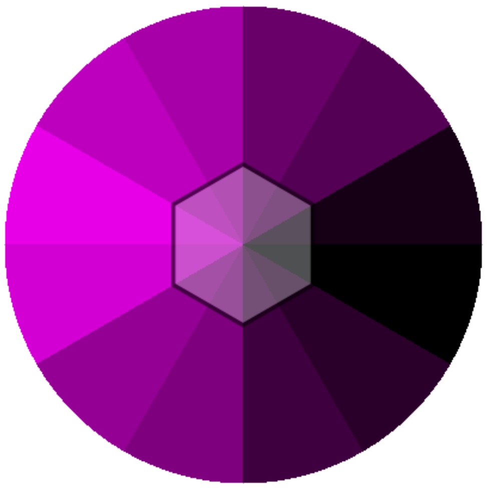


FIGURE 20. Fundamental domains of $I_2(6)$ on the hyperboloid with a hexagon superimposed

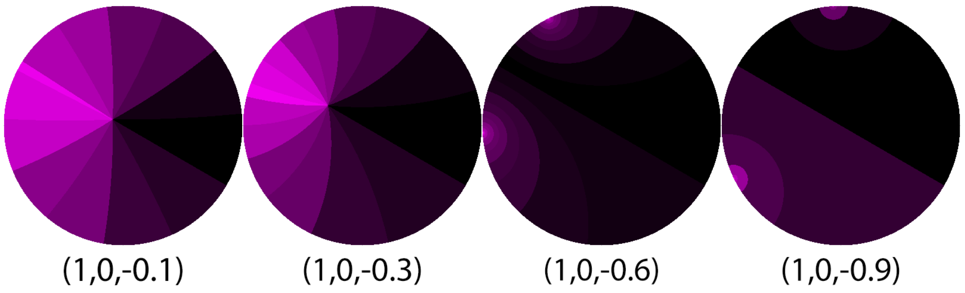


FIGURE 21. Progressively increasing magnitude of the z component for the first vector of $I_2(6)$

The figure above shows what happens to the fundamental domains as the z component is altered, as it increases in magnitude; the lines begin to curve more and the centre shifts. Eventually the centre moves all the way to a point at infinity, and the lines curve so much that the domains become separated circles. What are these showing? Do they still segment the hyperboloid into the symmetries of some hyperbolic polygon?

After attempting and failing to draw a polygon onto one of the images such that it could potentially generate these segmentations, it seems unlikely that these fundamental domains still represent the symmetries of a polygon. A polygon can't be drawn because the number of domains is inconsistent.

So what is actually happening? The first fundamental domain is generated by being bound between two hyperplanes. Then by taking repeated reflections, this shape is tessellated across the entire hyperboloid. It appears to no longer represent the symmetries of a polygon, but the tiling of a shape across the plane. This means that reflection groups can be utilised to create diagrams of hyperbolic tessellation.

How would one utilise reflection groups to tessellate a regular polygon across the hyperbolic plane? A Euclidean polygon can be made by combining right angle triangles, the same applies for hyperbolic polygons. In order to create a triangle via a fundamental domain it needs to be bound by 3 hyperplanes generated by the simple system. The third vector in the simple system could be guessed until the desired result is found, or the correct system can be generated using a Coxeter graph.

For example to start tiling a hexagon, take the Coxeter graph for $I_2(6)$ and add another vertex to this graph. Since this is a right angle triangle, this new vertex needs to be orthogonal to one of the vertices and thus the new vertex is only connected to one other vertex. For now, choose the weighting of the newly connected verticies to be 4. Translating this diagram into a root system, and changing the weights will be explored below.

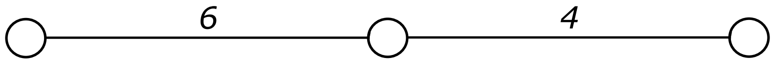


FIGURE 22. A Coxeter graph for hexagonal tessellation.

Take the simple system for $I_2(6)$ which was generated earlier and label it $\Delta = \{a, b, c\}$; with $a = (1, 0, 0)$, $b = (cos(\frac{5\pi}{6})sin(\frac{5\pi}{6}), 0)$ and c is the vector corrosponding to the new vertex on the graph. The angle between a and c is $\pi - \frac{1}{2} = \frac{\pi}{2}$, and the angle between b and c is $\pi - \frac{1}{4} = \frac{3\pi}{4}$. The dot and cross product can be used to calculate the x, y and z components for c, these values are relative to each other as the magnitude of c is not yet known.

$$\begin{gathered} a \cdot c = x = cos(\tfrac{\pi}{2}) = 0 \\ b \cdot c = y \sin(\tfrac{5\pi}{6}) = \cos(\tfrac{4\pi}{3}) \\ |a \times c|^2 = |(0, -z, y)|^2 = z^2 + y^2 = \sin(\tfrac{\pi}{2}) = 1 \end{gathered}$$

Combining these equations together, the x and y values are obvious, however the z component is less clear. The z compontent appears to be defined as: $z = \sqrt{1 - y^2}$, but y can be larger than 1; however an imaginary z component doesn't make sense. The relationship between these 3 values is the only important thing, so the imaginary value can be compensated for by taking the absolute value and flipping the sign of the z component. Summerising these calculations, the new vector is: $c = (0, \frac{\cos(\frac{3\pi}{4})}{\sin(\frac{5\pi}{6})}, -\sqrt{|1 - y^2|})$. Adding c to the simple system and generating the first fundamental domain, a hyperbolic triangle is generated.

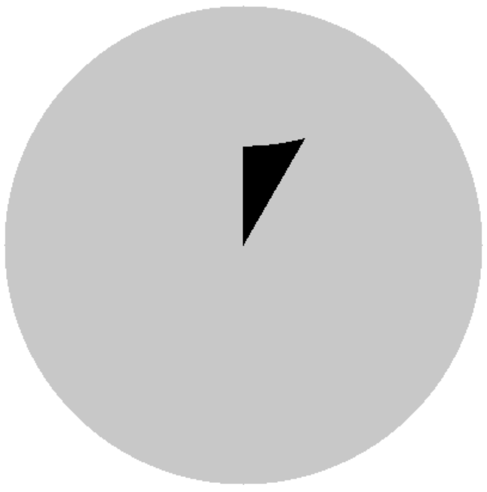


FIGURE 23. Singular hyperbolic trinagle from the simple system.

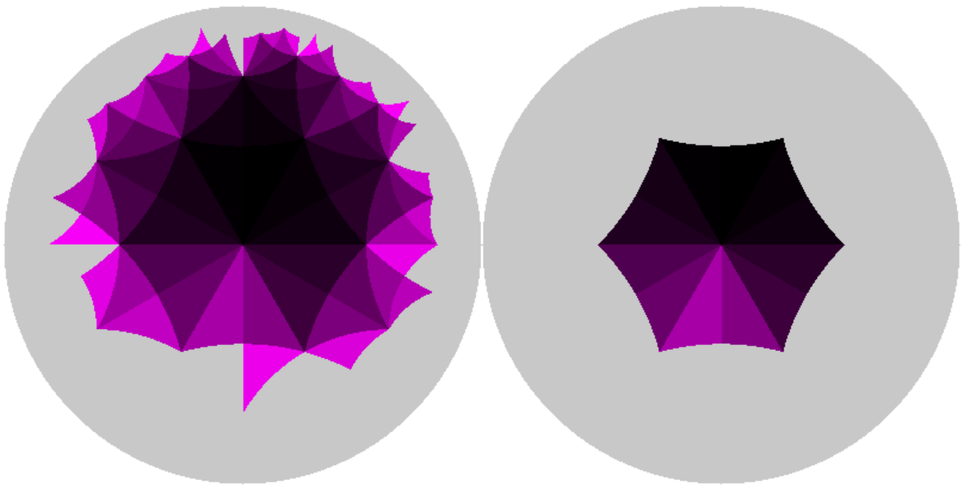


Figure 24. Hexagonal tesselation via triangles after 6 tree itterations.

After allowing the tree to take 6 itterations, the domains drawn complete a hyperbolic hexagon in the centre of the Poincaré disk. There are extra triangles that have been found throughout the 6 itterations, which have been removed in the right hand image in the figure above.

One thing that is becoming a problem with these images is the amount of domains being found, because they are theoretically infinite in number the difference in each domain's colour becomes hard to distinguish. Luckily the domains generated as chidldren in the tree all touch each other, therefore two colours is enough to distinguish the domains. A node in the tree will now store a boolean value, which represents if the domain is colour 1 or colour 2; when a child node is created it's colour value is the logial NOT of its parent, this fully colours the domains in a pattern which is clear.

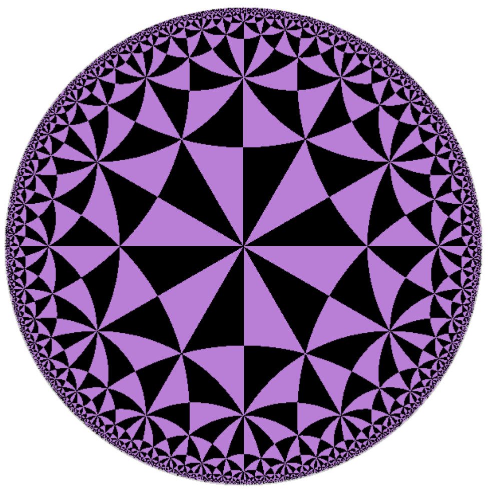


Figure 25. Full Hexagonal tesselation via hyperbolic triangles.

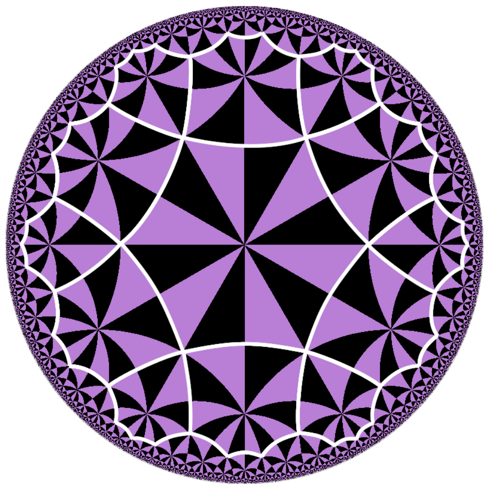


Figure 26. Hexagonal tesselation where the hexagons are labelled

Finally a full diagram can be created, taking the same simple system from before and set the image size to be larger, and allow the tree to take a maximum of 100 iterations. The resulting image is shown above, however it is not entirely clear where the hexagons are. The edges of the hyperbolic polygons are made up of two edges made by a triangle. Appropriatley selecting the correct edges the hexagons can be found.

Changing the weight between the first and second node on the coxeter graph will result in polygons with a different number of sides, what happens when the weight between the second and third node is changed? Testing the different results, making this weighting larger makes the central polygon larger. Finalising this project, some extra examples of hyperbolic tiling will be shown.

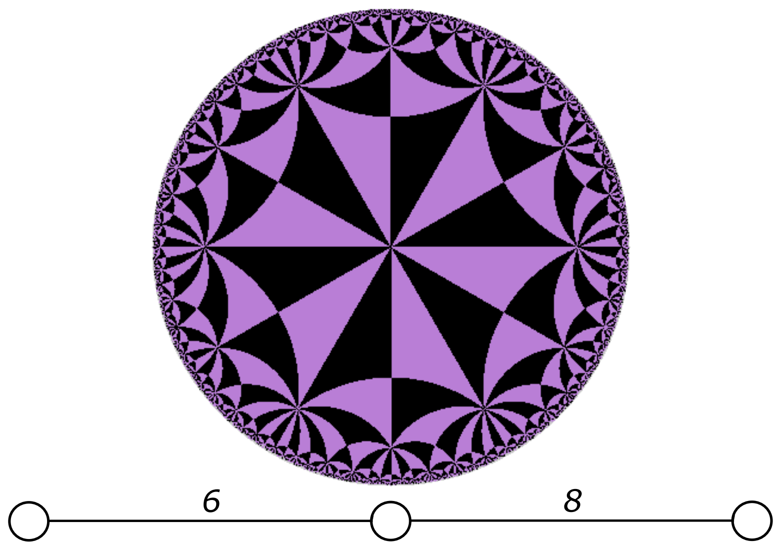
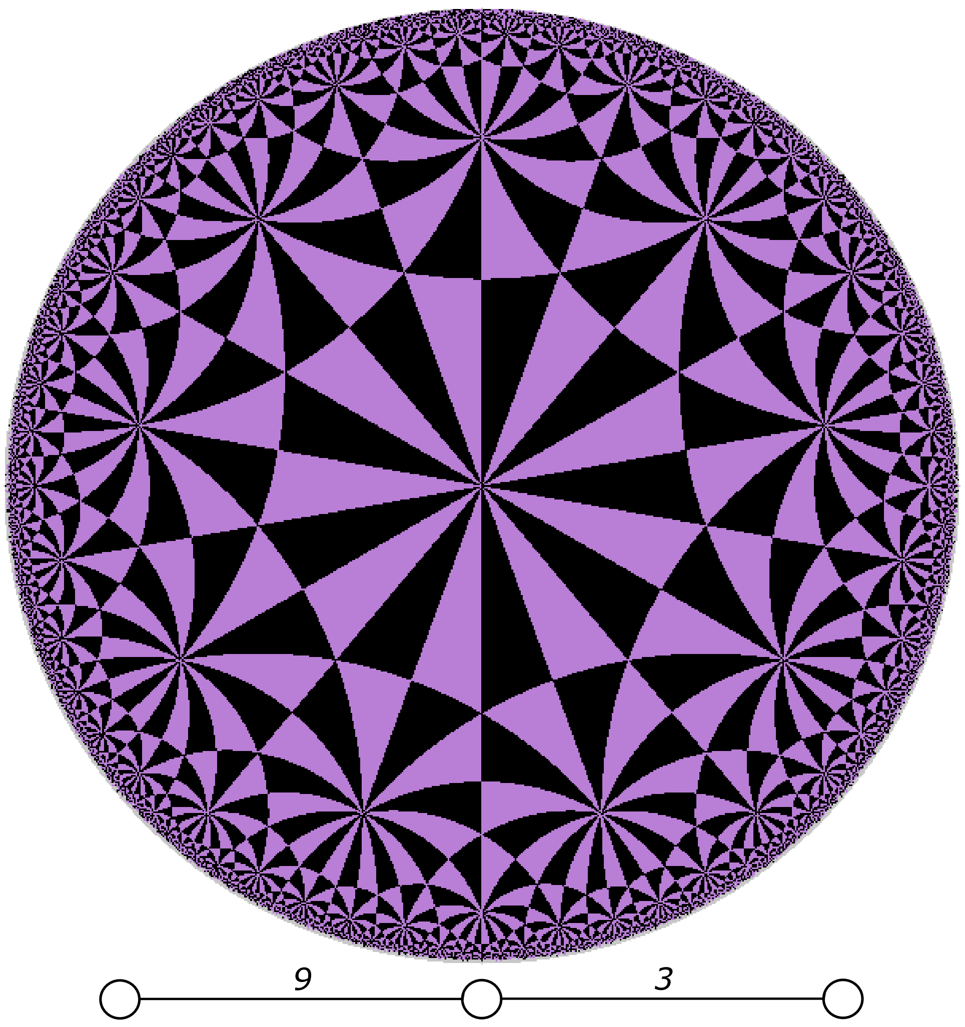
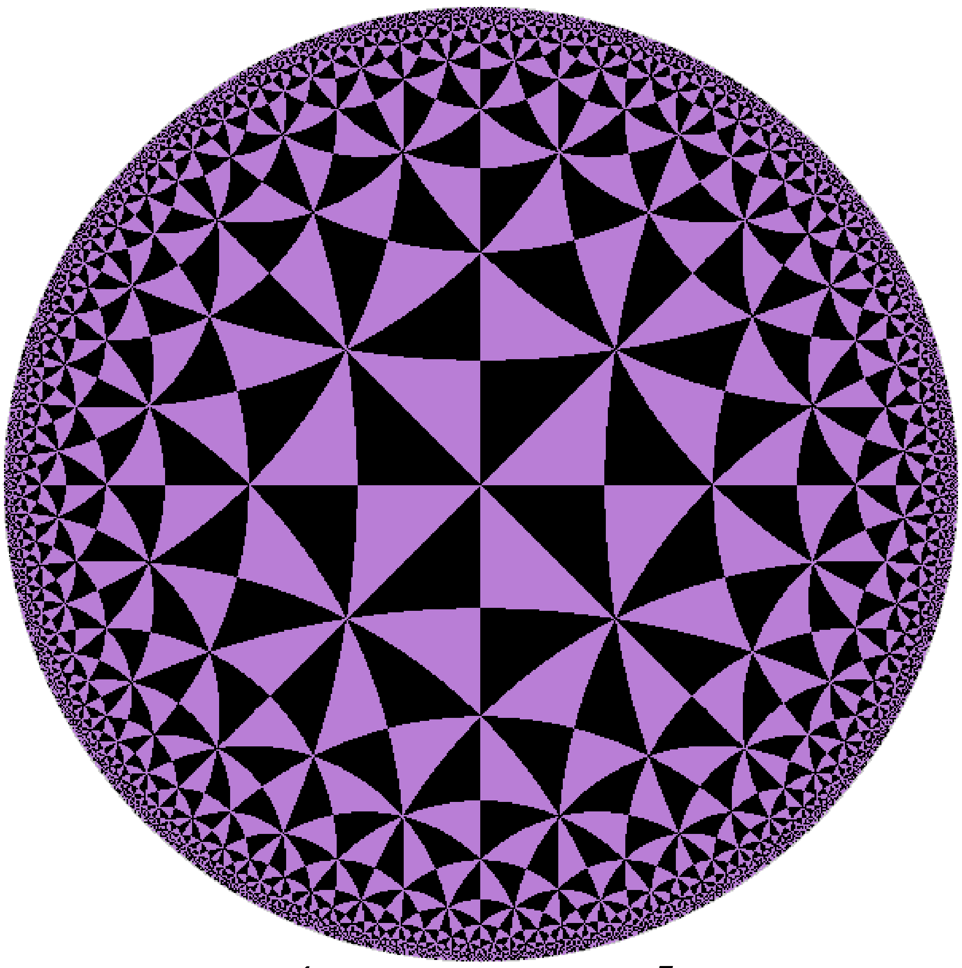
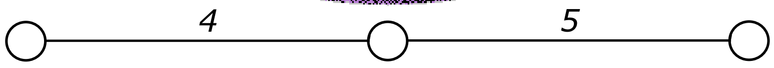
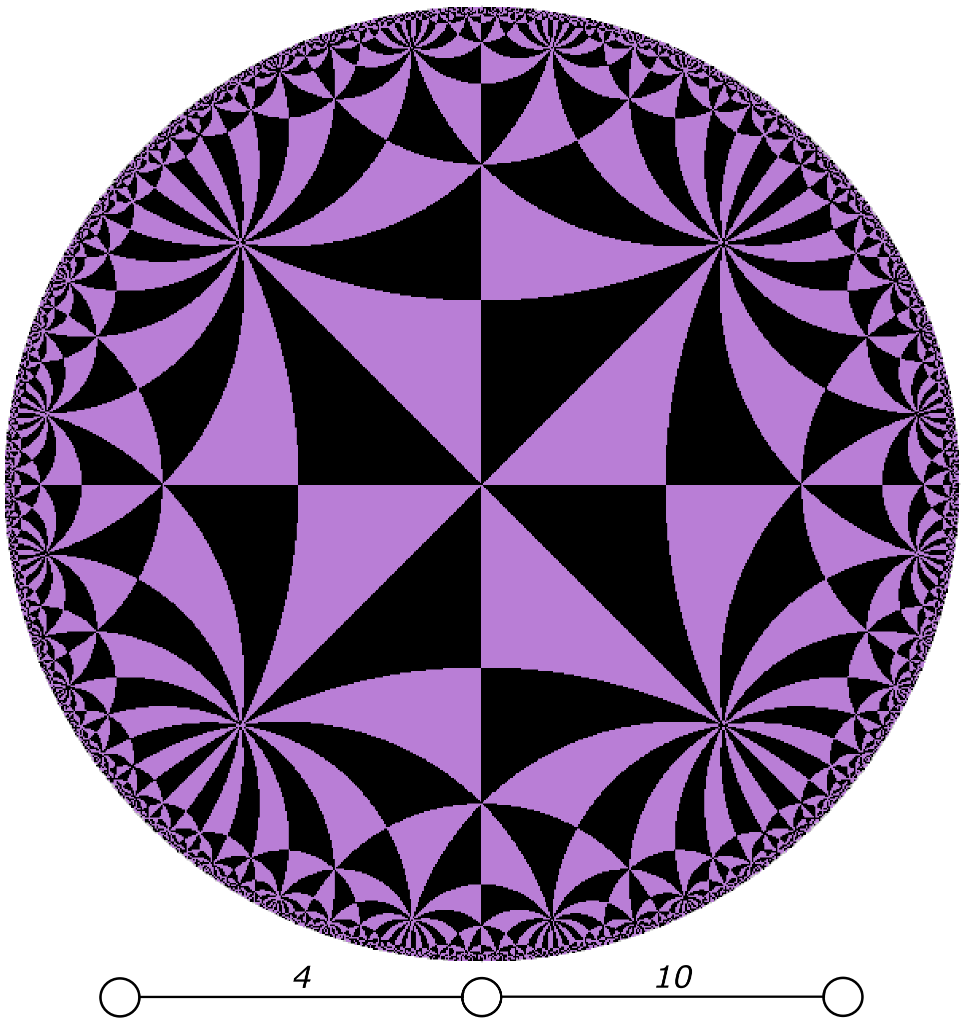


Figure 27. Fundamental domains of an altered Coxeter graph.


9
3




4
5



4. Conclusion

The main goal of this project has always been to generate mathematical imagery based on the fundamental domains of reflection groups. Although imagery of this type exists, I could not find any resources that explain the process taken; I believe this project's main value comes from explaining the programming steps taken to create this imagery. Reflection groups will generate this kind of imagery in any space, and I believe an adventurous reader could follow on with this project and recreate it in Euclidean or spherical space. That being said, reflection groups as a topic are much deeper than just the imagery that they produce; this report has sacrificed the classification side of reflection groups in order to focus more closely on the imagery. If a reader found the first half of the project more interesting, then I would recommend that they look further into reflection groups with a side of Lie algebras.

References

[1] G.N. Arzhantseva et al. *Geometric Group Theory: Geneva and Barcelona Conferences*. Trends in Mathematics. Birkhäuser Basel, 2007. ISBN: 9783764384128.

[2] A.V Borovik and A Borovik. *Mirrors and Reflections - The Geometry of Finite Reflection Groups*. Universitext. Springer New York, 2010. ISBN: 978-0-387-79065-7.

[3] L.C. Grove and C.T. Benson. *Finite Reflection Groups*. Graduate Texts in Mathematics. Springer New York, 1996. ISBN: 9780387960821.

[4] J.E. Humphreys. *Reflection Groups and Coxeter Groups*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 1990. ISBN: 9780521436137.

[5] R. Kane. *Reflection Groups and Invariant Theory*. CMS Books in Mathematics. Springer, 2001. ISBN: 9780387989792.

[6] J. Ratcliffe. *Foundations of Hyperbolic Manifolds*. Graduate Texts in Mathematics. Springer New York, 2006. ISBN: 9780387331973.

[7] N. Young. *An Introduction to Hilbert Space*. Cambridge mathematical textbooks. Cambridge University Press, 1988. ISBN: 9780521337175.