Using The Penrose Empires Software

Jason Healy

jhealy@logn.net

February 26, 2001

Contents

1 Introduction

This document contains the official documentation for the Java Penrose Empires package. In addition to explaining the features of the software, this document gives some background on the theory of Penrose tilings. An understanding of Penrose tilings, while not essential in using the software, does allow for a better comprehension of the software's features and behavior. I strongly urge anyone with interest in the software to learn more about the theory of the tilings.

For the terminally impatient, Section 3 contains a quick introduction to using the software.

The original version of this software was written as part of my Senior Honors Thesis at Williams College (1999-2000). Links to my thesis are available on the Penrose Empires web site, and I encourage anyone who would like a deeper understanding of the concepts behind this software to read it. While long (over 80 pages), and technical (no flowery language allowed in theses), it does thoroughly explain everything my software does.

I appreciate all forms of feedback on this software and documentation. If you have any questions, comments, or suggestions, please contact me.

Thank you,

Jason Healy
jhealy@logn.net

2 Background

Penrose tilings are geometric tilings formed from two *prototiles*, called *Kites* and *Darts*, examples of which can be seen in Figure 2. There are rules that govern the way these tiles may ajoin each other, and when these rules are followed an *aperiodic tiling* results.

Aperiodic, as the name implies, means that the tiling does not have any regularly repeating subsections; no finite subsection of the tiling can be used to create a legal Penrose tiling. Thought of another way, there exists no finite "rubber stamp" that can create a legal Penrose tiling.

Because of these rules that dictate which tiles may be placed against other tiles, there exist certain configurations of tiles that are not legal (as they would break one or more rules). Conversely, because of these rules, certain configurations of tiles must *always* exist. For example, in all Penrose tilings a Dart, as shown in Figure 1, must always contain two Kites in its convex section, as shown in Figure 2. Because the two Kites are the only tiles that can be placed against the Dart in this way, the Kites are said to be *forced*. Thus, a *forced tile* is a tile
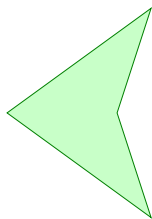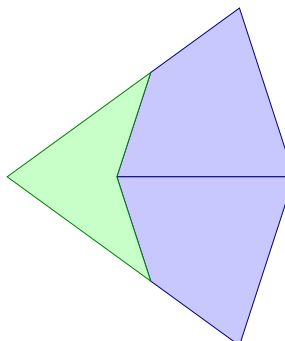
Figure 1: A single Dart.

Figure 2: A single Dart with both forced Kites shown.

that must exist in a position relative to some other tile whose position is fixed.

Given an initial patch of tiles, the set of tiles forced by this initial patch are called the *empire* of that patch. In the figures above, the two Kites constitute the empire of the Dart.

My software seeks to automatically find the empires of arbitrary patches of Penrose tiles. Given an initial patch of tiles, it will find as many tiles in the empire as it can. Note that this software is a work in progress; it does not necessarily find *all* forced tiles (indeed, because Penrose tilings are infinite, doing so would be extremely difficult!).

The software finds the empires by using *Ammann bars* (or *Ammann lines*, or *Musical sequences*). (If you do not know what Ammann bars are, then I strongly suggest you consult another reference for a more detailed explanation. What follows is only a cursory introduction.)

Ammann bars can be thought of as decorations on the Penrose tiles. When tiles ajoin one another, the Ammann bars should continue from one tile into the next in a straight, unbroken line. In this sense, Ammann bars enforce the matching rules that dictate tile placement. Tiles with the Ammann markings are shown in Figures 3 and 4.

Because the Ammann bars stretch infinitely in one dimension, however, they do serve another purpose. When we put a tile down in the plane, we can imagine extending the Ammann bars from the tile in all directions. As we add more tiles to the plane, these lines will intersect, forming patterns. When enough lines intersect, the patterns that form are such that only certain tiles can be placed on top of the patterns without disturbing the Ammann bars. When this happens, the tile that matches the pattern of Ammann bars is forced.

My software finds the intersections of Ammann bars, looks for patterns, and then draws all of the forced Penrose tiles. The results are then displayed for viewing
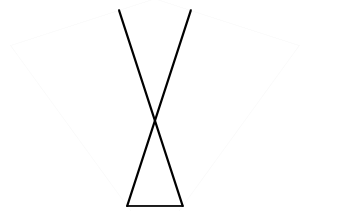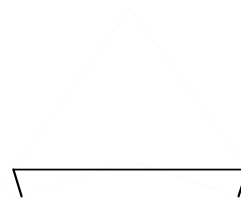
Figure 3: A Kite with Ammann markings.

Figure 4: A Dart with Ammann markings.

or exporting. For examples of how the Penrose tiles look when superimposed on the Ammann bars, please see Appendix A.

If you are interested in the details of how all of this actually works, please consult my Senior Thesis. It describes all the theory needed to understand exactly what the software does and how it does it. Please note: I have glossed over approximately 70 pages of material in this section. If you really want to know, you'll have to read the thesis!

If you just want to start playing with the program, read on...

3   Quick Start

To start using the software, you will need either the Binary JAR or the Source JAR, both of which are available from the web site: http://www.logn.net/penrose/.

3.1   Binary JAR Instructions

If you downloaded the Binary JAR file, then you can start running the software by typing

```
java -jar PenroseEmpires.jar
```

The software will automatically start the correct class, and the GUI will come up. From there, start clicking!

3.2   Source JAR Instructions

If you downloaded a Source JAR, you will need to unpack and compile the sources first. This can be done by typing the following commands:

```
jar xf PenroseEmpiresSource.jar
javac net/logn/util/vectoroutput/*.java
javac net/logn/penrose/*.java
```

Now you are ready to run the software:

`java net.logn.penrose.PenroseApplet`

(Note: though the name of the main class is "PenroseApplet", the software adds application features (such as menus) when run from the command line. You should always run the software as an application to have access to these features.)
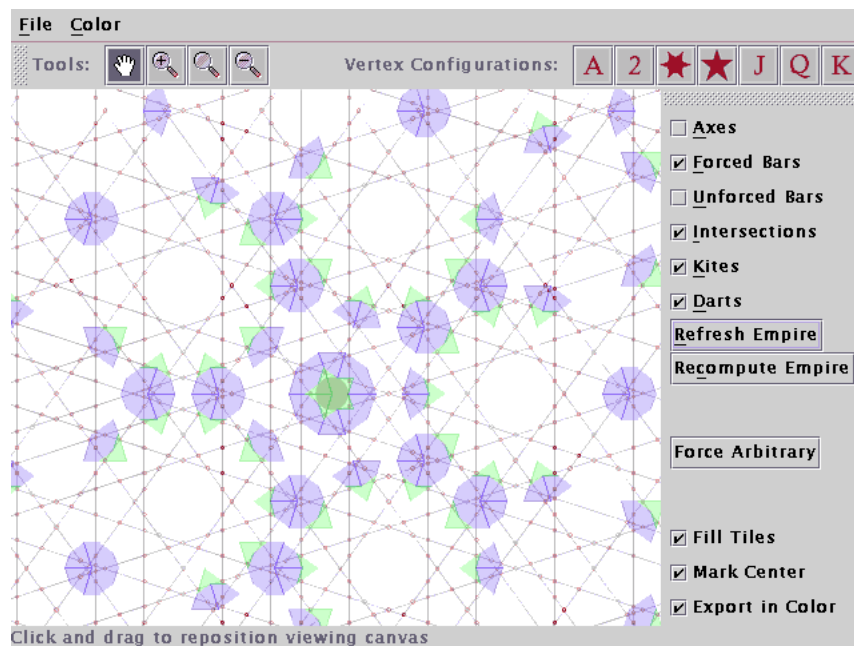
4   Graphical User Interface Mode

Figure 5: The Graphical User Interface.

The Penrose Empires software may be run in a Graphical User Interface (GUI) mode, and most of the functionality exists only in this mode. However, a Command Line Interface (CLI) mode also exists, and is discussed in Section 5. A picture of the GUI is shown in Figure 5.

7

4.1 Toolbar Commands

The toolbar has four basic commands that affect the position of the tiling on the screen. This allows the user to examine any part of the tiling.

4.1.1 Move Tool

The Move Tool is used to position the Penrose tiling in the window provided. Clicking on the image will result in centering the image at that point. Clicking and dragging allows the user to reposition the image. Note that the image is not rendered outside the bounds of the viewable area, and so moving the image may result in part of the screen not being updated. You must recompute the empire (described below) in order to refresh these areas of the screen.

4.1.2 Zoom Tool

The Zoom Tool zooms in or out of the image by an arbitrary amount. Clicking on the tool brings up a dialog box asking how much of a zoom factor should be used. Numbers between 0 and 1 are for zooming out, and numbers greater than 1 are for zooming in.

After selecting a zoom factor, clicking on the image will zoom in (or out) and center on that point, using the zoom factor entered above.

4.1.3 Zoom In Tool

The Zoom In Tool behaves like the Zoom Tool, only the zoom factor is always 2. No dialog asking for the zoom factor is presented to the user.

4.1.4 Zoom Out Tool

The Zoom Out Tool behaves like the Zoom Tool, only the zoom factor is always 0.5 (zoom out by a factor of 2). No dialog asking for the zoom factor is presented to the user.

4.2 Vertex Configurations

While the user can construct their own Ammann bar configurations (see below), they may find it helpful to start with one of the standard seven vertex configurations. Clicking on one of the buttons shown below will erase the current configuration and replace it with the configuration yielding the named vertex

configuration. For examples of what the vertex configurations look like, refer to Appendix A.

By default, the software uses the Sun configuration.

A load the Ace configuration.
2 load the Deuce configuration.
★ load the Sun configuration.
★ load the Star configuration.
J load the Jack configuration.
Q load the Queen configuration.
K load the King configuration.

4.3   Drawing Options

These check boxes on the right-hand side of the screen represent what features will be drawn to the screen or exported to a file. Note that a recomputation should be performed (see below) before all features will be drawn correctly. Once a recomputation has occurred, however, one need only to check or uncheck the desired features and press the **Refresh** button to update the tiling. The **Recompute** button only needs to be used if the image is moved or zoomed.

4.4   Other Options

Below the **Refresh** and **Recompute** buttons are check boxes affecting how the tiling is rendered to the screen or exported to a file. They do not affect the computation of the tiling, and may be changed at any time. They do not require a recomputation to take effect; a refresh will suffice.

4.4.1   Fill Tiles

Determines whether the tile shapes should be filled or drawn only as an outline. When not **Export In Color** is not selected (see below), it is often best to uncheck this box and only draw outlines.

4.4.2   Mark Center

When checked, a unit circle is drawn at the center of the tiling. Note that the GUI provides current coordinates based on where the pointer is, so this feature is most useful for locating the center of the plane in exported or printed tilings.

4.4.3 Export In Color

When checked, exporting or printing will occur in color. Note that some printers do not print well in color, and that the file size of color exporting will be larger. Often it is best to uncheck **Fill Tiles** when not exporting in color, or else all objects will be filled with the same shade of gray.

4.5 Refreshing and Recomputing

These two buttons account for most of the action that the software performs. Because the computations can be very expensive, it is essential to know when to use each button.

4.5.1 Refresh Empire

Use this button whenever an option has changed and the screen should be refreshed. Examples include changing what features to display, what colors to use, and whether to fill shapes or not. Additionally, refreshing eliminates the "jaggies" when zooming in or out. It does not find any new tiles in the empire, nor does it draw information outside the area that was defined when the last recomputation occurred. Thus, if the portion of the plane under examination changes substantially, a recomputation will be necessary.

4.5.2 Recompute Empire

Use this button whenever new information needs to be computed. This button should be used when the user moves the portion of the tiling under consideration, zooms in or (more importantly) out, forces new bars, or changes the current vertex configuration.

4.5.3 Progress of Recomputation

Recomputation can be a time-consuming task. In order to show progress to the user, features are drawn as they are found. Here is a brief outline of what goes on during a recomputation:

1. The screen is cleared, and all data structures are cleared.

2. All intersection points between Ammann bars are found.

3. These points are sorted into "boxes" to reduce the number of comparisons performed on each point.

4. The points, and the boxes they belong to, are displayed to the screen

5. The software begins to search for Penrose tiles. As they are found, they are drawn to the screen. If new Ammann bars are added to the tiling, then the recomputation begins again with step 2. This may happen several times if a lot of new information is found in the tiling.

6. When all forcings are complete, the tile information is dumped into arrays for easier access. The screen refreshes with the final tiling displayed.

For a better understanding of what goes on at each step, you are encouraged to read my thesis. The details of what goes on during these computations are beyond the scope of this document.

4.6   Forcing Arbitrary Bars

In order to test arbitrary configurations of tiles, an editor built into the system allows the user to add Ammann bars to the plane. The button that performs this action is labeled **Force Arbitrary**.

Clicking on that button brings up a new window like the one shown in Figure 6. The user can select the sequence, bar number, and length of the bar to force. If you are not familiar with "bar numbers," "lengths," and "sequences" you should read my thesis, as these terms have very specific meanings.

To force an arbitrary bar, take the following steps:

1. Each of the five musical sequences in the plane has a rotation that is a multiple of 72°. Click on the rotation for the sequence you would like to alter.

2. Next, select the bar number that you would like to force. All bars have a "center" at bar 0 (zero). Positive bars lie in the direction of the sequence, and negative bars lie 180° along the sequence, relative to the zero bar. (If you are not familiar with the concept of a "zero bar," please refer to my thesis.)

3. Now click the **Preview** button to see the approximate region where the bar will be forced. Note that because forcings are relative, the bar may not appear in its exact location; it may appear to be at a **L** interval when the user has selected a **S**, or vice-versa.

4. When ready to permanently force a bar, click the **Force** button. The screen will update, showing the new forced bar, along with any additional bars that it forces. Note that the **Recompute** button must be used to find any features forced by these new bars.

Figure 6: The Bar-Forcing Window.

5. More bars may now be forced, or the **Done** button can be clicked to close the Forcing window.

Please note, you cannot "undo" a bar forcing. Because forcing one bar causes an infinite number of other bars to be forced as well, the operation cannot be reversed.

4.7   Menu Options

When run as an application, the software adds a menu bar to the application frame. The options available in the menu can only be executed with filesystem access privileges.

4.7.1   File Menu

The **File** menu contains actions to save the state of the program, print the current rendering, or export the current rendering to a file.

Open/Save Tiling Configuration: A **tiling configuration** consists of all the Ammann bars in the plane. If one creates custom Ammann bar configurations using the forcing window then they can be saved and loaded using these menu options. This prevents the time-consuming task of repeatedly building up the Ammann bar configurations by hand.

(A tiling configuration consumes less than 1 kilobyte of disk space, and the storage requirements do not change base on the number of Ammann bars saved.)

Open/Save System Properties: The **system properties** contain all the information about how the program behaves. This includes the colors used for rendering and all the checkbox options used for drawing objects to the screen.

To start the application and automatically load a set of saved system properties, add the properties file as an argument to the software:

```
java -jar PenroseEmpires.jar properties-file
```

Additionally, the properties file can be used when the software is run in Command Line Interface mode (see Section 5). It can be easier to run the software in GUI mode and save the system properties than to configure them manually on the command line.

Export Rendering: This submenu allows the user to export the rendering currently on the screen to a high-quality graphics format such as EPS or PDF. These vector formats are of much higher quality than the raster graphics used in the application window, and will also print with higher quality than Java's built-in printing methods.

The PNG format is also supported, provided that the user downloads and installs the Java Advanced Imaging API. Note that exported PNG graphics share the same graphics bug that the native windowing code contains. Thus, PNGs may exhibit slight drawing errors. See Section 6 for more information.

Print: Selecting the print option allows direct printing to any attached printer. Note that the printing is performed from a raster format, and so may result in slightly jagged images. For the highest quality output, export the rendering to EPS or PDF and then print those images directly.

Quit: Quits the software. The state of the program is not saved.

4.7.2 Color Menu

The **Color** menu allows the user to customize the display colors used by the software. Selecting a color to change brings up a dialog box with a color picker. Simply selected the desired color, and then refresh the tiling.

To save the current color selections, the user must select **Save System Properties** from the **File** menu.

5 Command Line Interface Mode

In addition to the standard GUI mode, the Penrose Empires software can be run in Command Line Interface (CLI) mode. While the CLI mode is not interactive, and thus does not allow for exploration of tilings, it does have a few key

advantages:

- The software is scriptable, and can have all options supplied on the command line or in a properties file.

- No GUI code is required; CLI mode does not require a graphical environment to be run.

- The command line version uses an iterative system to find tiles; memory consumption is *greatly* reduced.

The last point is perhaps the most important. The GUI mode requires finding all tiles before displaying them. For large numbers of tiles, the storage requirements will often cause the JVM to run out of heap space. The CLI mode, however, streams the tiles out to disk as they are found. The result is a program that is currently bounded only by the number of Ammann bars that need to be found. (This bound may be removed in future releases, resulting in software that is bounded only by the disk space required to store the final image.)

As a "proof of concept", a tiling was rendered with dimensions of 5000x5000 units (recall that a single tile is on the order of 1 unit). The resulting compressed file was 220MB, much larger than could have been held in RAM. Nevertheless, the software completed the computation in less memory than the GUI version would have used for a tiling of 100x100 units.

To run the software in CLI mode, use the following command (it should all be on a single line):

```
java -cp PenroseEmpires.jar net.logn.penrose.PenroseTiling
    tiling-configuration output-file eps|pdf [properties-file]
```

Note that the `-cp` option must be used instead of the `-jar` option, because the class name is being specified manually.

The *tiling-configuration* should be a configuration file saved from the GUI. (The seven vertex configurations (see Appendix A) are included in the source distribution.)

The *output-file* should be a file that the user has write permissions to, and will contain the final rendered image.

The *eps* or *pdf* option must be specified to define what format the final rendering should take. Currently, these are the only two formats supported.

Finally, the software takes an optional *properties-file* argument, which points to a file containing all the properties for the image rendering. These options include colors, drawing features, and the area to render. The properties file may be created in the GUI, or created as a text file (the source distribution has a sample file).

If the *properties-file* option is omitted, then the default properties will be used. The user may also override the default properties by using the Java System Properties. The key and parameter values are the same as those used in the properties file. For a listing of all properties, run the software with no arguments:

```
java -cp PenroseEmpires.jar net.logn.penrose.PenroseTiling
```

As the software runs, it provides information to the standard output to let the user know how much of the computation has been completed, as well as a rough estimation of the time remaining in the computation. The information is given in terms of "blocks", which refers to the areas of the plane that the rendering is divided into for processing.

6   Known Bugs

While I have tried to make the software as efficient and correct as possible, no software would be complete without some bugs. Here are the ones I know about:

- The most prominent bug is actually contained inside the JVM itself. There exists a bug where clipped lines occasionally result in areas of the screen being obscured. In my software this shows up in both the GUI and in exported PNG files. It does not affect the vector output formats (PDF and EPS). For more information on the bug, please see Java Bug #4363202.

- The software does not yet draw Penrose Rhomb tiles, though I hope to add this feature in the future. (There are sections of code that attempt to add the rhombs, but I have commented them out because they do not work correctly.)

- The GUI mode tends to run out of memory for large tilings. If you get `java.lang.OutOfMemoryError` errors when trying to compute a tiling, you have two options:

  1. Give the JVM more memory. `java -mx`N`M ...` will give Java a larger heap size (replace N with the memory amount, in megabytes, that you would like to give to Java). 128MB should be sufficient for most tilings (tilings that are larger are difficult to see on the screen, as the tiles become so small).
  2. Use the software in Command Line mode (see Section 5). Command Line mode uses a more efficient rendering method that is far less likely to run out of memory. One can usually get a good idea of the area one wants to examine in the GUI, and then actually render the tiling in CLI mode.

If you find any bugs, or have any suggestions for making the program more useful, please e-mail me!

A Vertex Configurations

Figures 7, 8, 9, 10, 11, 12, and 13 show the seven standard vertex configurations that can occur in a Penrose tiling. The filled tiles represent those in the named vertex configuration. The hollow tiles represent some of the tiles in the empire of that configuration (not all configurations force tiles, however). The Ammann bars shown are those forced by the vertex configuration. The names of the configurations are those used by John Conway, of Princeton University. See Appendix B for more information.

These configurations are provided as a quick reference. For more information regarding these initial vertex configurations, including the offsets of the Ammann bars that produce them, please refer to Chapter 6 of my thesis. It contains detailed diagrams and tables which outline how to construct the seven vertex configurations from Ammann bars.

B Further Reading

Below are some pointers to other documents that you may find helpful in learning about Penrose tilings (including a shameless plug for my thesis).

1. J. H. Conway and J. C. Lagarias. Tiling with polyominoes and combinatorial group theory. *Journal of Combinatorial Theory*, Series A, 53:183 208, 1990.

2. Jason Healy. Automatic Generation of Penrose Empires. Technical report, Williams College Department of Computer Science, 2000.

3. Branko Grünbaum and G. C. Shephard. *Tilings and Patterns, chapter 10.* W. H. Freeman and Company, New York, 1987.

4. Linden Minnick. Generalized forcing in aperiodic tilings. Technical report, Williams College Department of Computer Science, 1998.

5. Roger Penrose. The Rôle of aesthetics in pure and applied mathematical research. *Bulletin of the Institute of Mathematics and Its Applications*, 10:266 271, 1974.

6. Joshua E.S. Socolar. Growth rules for quasicrystals. In D. P. DiVincenzo and P. J. Steinhardt, editors, *Quasicrystals: The State of the Art, volume 11 of Directions in Condensed Matter Physics.* World Scientific, Singapore, 1991.
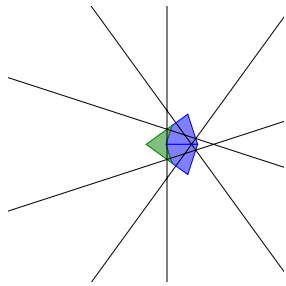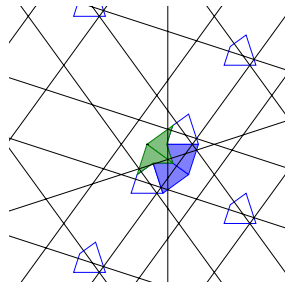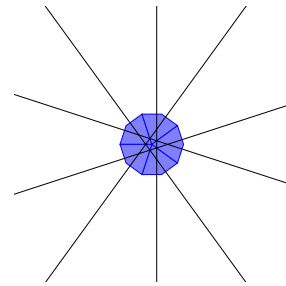
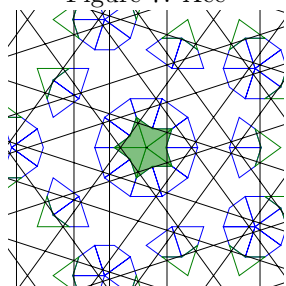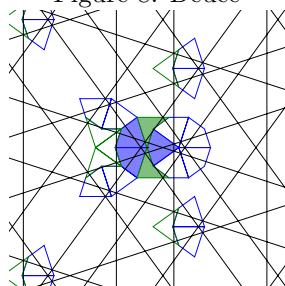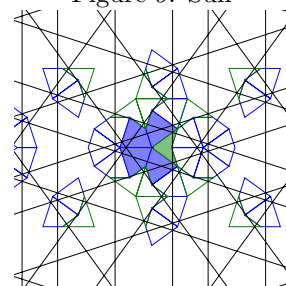Figure 7: Ace

Figure 8: Deuce

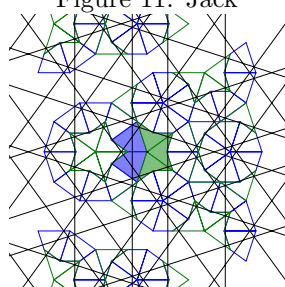
Figure 9: Sun

Figure 10: Star

Figure 11: Jack

Figure 12: Queen

Figure 13: King