This is a readme from an old example version of the graph codes. The workhorse files have been updated since then with more features and more capabilities, but it roughly works still. I have also added more examples in a subfolder “useful scripts”. These are files that do useful things, but are not necessarily in a userfriendly condition. 1-6-2020

Files are now in kollar lab shared google drive.

Okay. Here's a set of the codes that we use as of 8/13/18.

“/ourphoton/Alicia/Layouts/ExamplePythonPackage\_8-12-18”

Although if you’ve found this read me and haven’t found those files, then something is badly wrong.

**The whole package runs in python 2.7**

I’m serious. Python 2.7, not 3 or 5 million.

Don’t come crying to me if you try to change it over and nothing works, and you can’t read any of the precompiled examples. Found in

“ourphoton/Alicia/Layouts/HyperbolicPickles”

The work horse is three file: LayoutGenerator, EuclideanLayoutGenerator, and GeneralLayoutGenerator.

Along with those I've included a series of files whose names start Ex\_ These are sample codes that do a bunch of the operations that we typically do.

LayoutGenerator is the oldest code that was originally written for hyperbolic lattices and contains one class: PlanarLayout

EuclideanLayoutGenerator is a second generation code that does normal 2D lattices. It assumes the existence of a Bravais lattice and contains two classes: EuclideanLayout, and UnitCell

GeneralLayoutGenerator is the newest generation of code. It has a couple of classes in it and a few helper functions: GeneralLayout, TreeResonators, shift, split, rotate, generate\_line\_graph...

LayoutGenerator is a bit different from the other two codes in that it cannot generate a layout lattice by itself. Instead it takes a set of resonators as input and then computes anything that you want. This code came into existence because we started to do things like splitting lattices and taking line graphs and decorating. The idea was then that GeneralLayout is sort of like a wrapper for all the stuff that you might ever want to make and it unifies other mode specific codes that generate lattices.

TreeResonators is a class that makes lattices which are trees

The rest of the helper functions are so that we can take previously compiled layouts and start splitting and line graphing and making the crazy examples.

Each of these files has a relatively detailed doc string in the header describing what is in it, what the built in methods are, sample syntax, etc.

They can also be run as python scripts themselves, in which case they test out a lot of methods of the classes that they contain. (One word of warning, this was the code I used to originally develop a lot of the functionality that we have, so some of the operations are done at a very low level. There may well exist automated functions which do the function of some of the blocks of code and example figures found here.)

I believe that all of the example codes will run on your computer as long as you are connected to ourphoton.

UnitCellTester: shows how to define unit cells and computer bands structures using the UnitCell class

SamplePlots\_Collected: this contains the code for a lot of the figures in the first hyperbolic paper

HarshExamples and JakeExamples: these are sample files that I wrote to show two new students how to use this code. They are my attempts to illustrate the syntax for most of the basic functionality that we have. They don't show the fanciest things, but they have been stripped down and don't have the vestiges of earlier versions commented out and cluttering them up.

MathPhysPaperPlots4: this is the code which currently produces most of the figures for the newest paper. It shows in newest ways that we have to make figures, and the color conventions that we have settled on. The DefaultFig is particularly nice for this. (One thing to be a little bit careful of. This file makes a LOT of figures and will take a while to compile all the necessary objects. There is a section with boolean flags to turn the different figures on and off if you don't want it to do everything at the same time.)

C60.py is one more highly advanced example. In this code I essentially redefined the function from PlanarLayout that makes hyperbolic tilings to try to make some special spherical tilings with different polygons. It currently makes four different carbon fullerenes, and then some crazy examples where the polygon changes radially as you go out.

I'm mostly passing it along because it's fun to play with. It's probably not very educational since it involves hacking up the guts of a very low level function in a not very generalizable way.