Image Manipulation Techniques Exploring Escher and Non-Euclidean Space

Independent Study

Hannah Apuan 873000708

5-31-2017

TABLE OF CONTENTS

TIMELINE AND MILESTONES

After discussing with Scott Leutenegger potential topics for my Independent Study, I decided to attempt to recreate the Droste Effect computationally. I reference a publication called "The Droste Effect Image Transformation" by Jos Leys. This paper describes the mathematical transformations required to create the Droste Effect. These include transformations across the *z* axis. As it is not yet the end of the quarter, all of this information is considered a work-in-progress. Upon the initialization of the project, Scott said that I should create a timeline with milestones. I have created the following timeline which I have kept up to date. I discussed with Scott biweekly and gave him updates on my progress biweekly.

WEEK 1:

- Start research on what software/programming language/programming packages are being used.

WEEK 3:

- Read through paper
- have general algorithm determined and how it is going to be implemented (with
- math technique and with which software/visual representation method)
- Plan to implement in 3 different ways (maybe different mathematical techniques,
- different software, or visual representation methods (ie 2d, 3d)
- Have notes explicitly defining plan, tools I plan to use (graphics software, math)
- initial thoughts (Introduction) written for paper discussing steps taken which
- includes diagrams of complete planned process
- (Since this paper is mostly mathematical, I want to focus my writing about how I implemented it computationally.)

WEEK 5:

- Have initial algorithm figured out more explicitly.
- Have initial implementation working
- Have writing done about algorithm implementation
- Created GitHub repo with readme

WEEK 7:

- Have developed implementation for each method
- Develop further algorithm implementation writing
- Have writing done about issues/solutions, conclusion, and afterthoughts

WEEK 9:

- Fine tune paper and algorithm techniques
- Have final code uploaded to GitHub
- Have final implementation for all three ways

GITHUB CODE

My code for all my work is uploaded to GitHub. (link: https://github.com/hannahapuan/droste-effect)

I will also attach some of the main code documents on here if you do not wish to look at the GitHub. However, I will continue to update the GitHub

My initial implementation was with a programming language called Python using the Python math package Numpy.

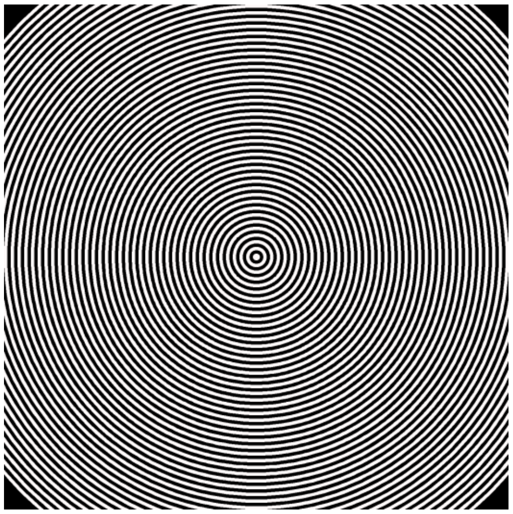
CREATEDROSTE.PY

CODE

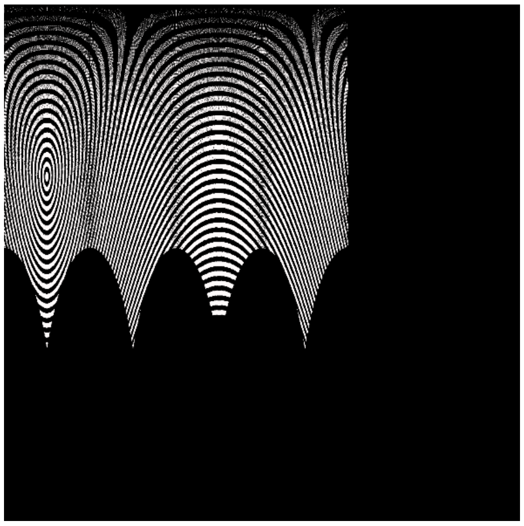
```python
import numpy as np
import math
from PIL import Image, ImageDraw

def main():
    print("main code")

    getImage()

#def cartpol(x, y):

#    return(a, b)

def polcart(r, p):
    x = r * np.cos(p)
    y = r * np.sin(p)
    return(x, y)

def getImage():
    count = 0
    img = Image.new('RGB',(1500,1500),"black")
    pixels = img.load()
    pixel = pixels[0,0] #first pixel value
    #for x in range(img.size[0]): #make black and white stripes
    #    for y in range(img.size[1]):
    #        count = count + 1
    #        if count > 150:
    #            pixels[x,y] = (254,254,254)
    #        if count > 300:
    #            count = 0
    radius = 1000
    draw = ImageDraw.Draw(img)
    x = 1500/2
    y = 1500/2
    flip = True
    white = (254,254,254)
    black = (0,0,0)
    numCircles = 0
    while numCircles < 100:
        print('==========')
        if flip == True:
          #  print(flip)
           # print("1:", x-radius,y-radius, x+ radius, y+radius)
            draw.ellipse((x-radius, y-radius, x + radius, y + radius), fill=(0,0,0))
            flip = False
        else:
           # print(flip)
           # print("2:",x-radius,y-radius, x+ radius, y+radius)
            draw.ellipse((x-radius, y-radius, x + radius, y + radius),
fill=(254,250,250))
            flip = True
        radius = radius - 10
        numCircles = numCircles + 1

    img.save('before.png','png')
    img2 = Image.new('RGB',(1500,1500),"black")
    pixels2 = img2.load()
    count = 0
    width = 1000
    height = 1000
    maxradius = np.sqrt(width**2 + height**2)/2
    rscale = width / maxradius
    tscale = height/(2*math.pi)
    for y in range(0,height):
        dy = y - height/2
        for x in range(0, width):
            dx = x - width/2
            t = np.arctan2(dy,dx)%(2*math.pi)
            r = np.sqrt(dx**2+dy**2)

            pixels2[int(t*tscale),int(r*rscale)] = pixels[x,y]
    img2.save('after.png','png')

if  __name__ =='__main__':main()
```

INPUT AND OUTPUT

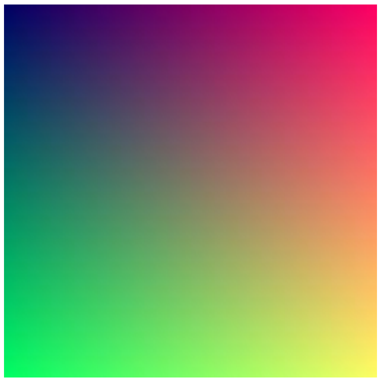
Input:



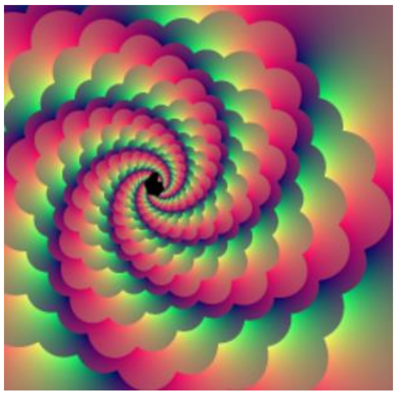
Output:



Input:

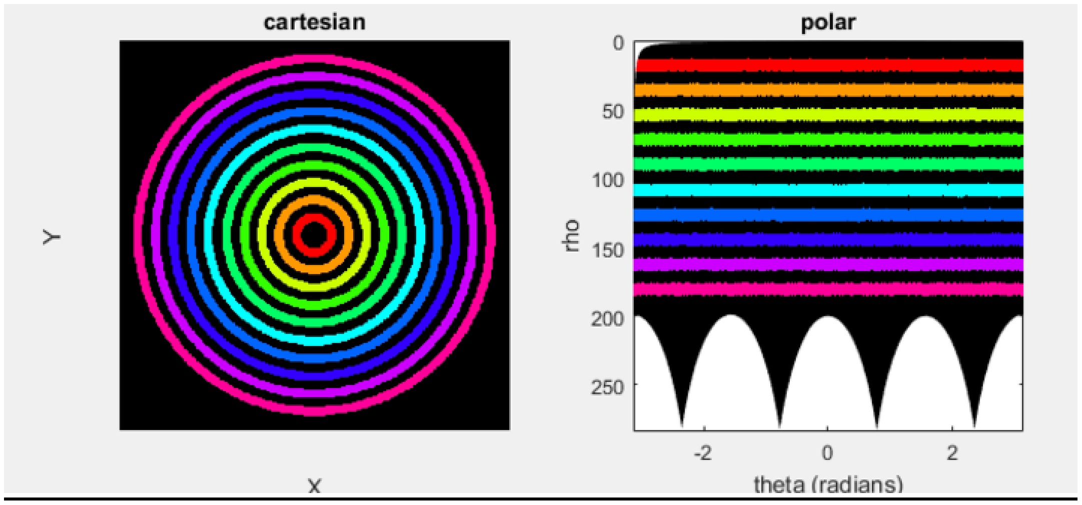


Output:



After implementing it in Python, I was unhappy with the results because of the way Numpy rounds the arctangent function, which is what creates the black space in the output in the first example's output. The above images were created completely by me.

ROUNDING ERROR:



Because of the rounding error, I decided to move to a web version using Javascript and HTML5's Canvas.

During the next week, I plan to create a web-based GUI which will allow people to create their own Droste Effect in their web browser.

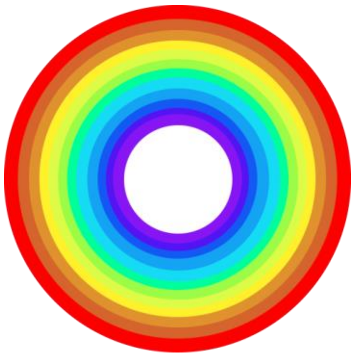
DROSTE-EFFECT-WEB.HTML

CODE

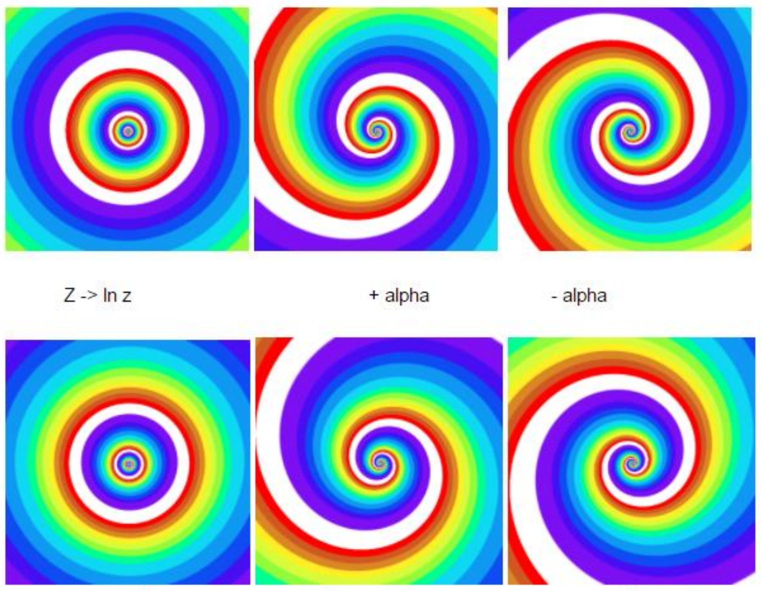
```
<!DOCTYPE html>
<html>
<body>

<h1>droste-effect</h1>
<p>Enter values:</p>

c:
<p id="c"></p>

<script>
init();

    //#pixel
function init(){
    var ARGrot = 0.0;
    var ARGwid = 2.0;
    var ARGheigh = 2.0;
    var ARGcenlarge = math.complex(0.0,0.0);
    var ARGcensmall = math.complex(.5,.5);
    var ARGfac = .25;    //range = [0.0,1.0]
    var ARGmode = 1;

//////////////////////////////end
    var rot = ARGrot*pi/180;
    var plar = math.complex(0.0,0.0);
    var rad = 0;
    var radin = 0;
    var i = math.complex(0.0,1.0);

    var c = math.complex(ARGcenlarge + (ARGcensmall-ARGcenlarge) / (1-
ARGfac*exp(i*rot));
    var plar=ARGcenlarge+ARGwid/2+i*ARGheigh/2;
    var rad= math.re(plar-c);
    var d = 1 / ARGfac;
    var m = math.complex(0,0);
    var alfa = 0;
    var sc = 0;
    /***var states: 0 = zero; 1 = alfa; 2 = 180 - alfa; 3 = 180; 4 = 180 + alfa; 5 = -
alfa***/
    switch(ARGvar){
        case 0: //zero
            sc = 1;
            alfa = 0;
        break;
        case 1: //alfa
            m = 2*math.pi-rot+i*math.log(d);
        break;
        case 2: //180-alfa
            m = 2*math.pi-rot+i*math.log(d)
            alfa=math.atan2(m);
            sc=math.cos(alfa)*2*math.pi/(2*math.pi-rot),alfa=math.pi+alfa;
        break;
        case 3: //180
            sc=1;
            alfa=math.pi;
        break;
        case 4: //180+alfa
            m = 2 * math.pi + rot + i * math.log(d);
            alfa=math.atan2(m);
            sc=math.cos(alfa)*2*math.pi/2*math.pi+rot);
            alfa=math.pi-alfa;
        break;
        case 5: //-alfa
            m = 2 * math.pi + rot + i * math.log(d);
            alfa = math.atan2(m);
            sc=math.cos(alfa)*2*math.pi/(2*math.pi+rot);
            alfa = 2*math.pi-alfa;
        break;
        var beta = math.atan(rot/math.log(d));
        //TODO: fix this complex on argument thing
        var scale = math.complex(d^(sc*math.exp(i*(alfa-beta)/math.cos(beta))));
    }
        var beta = math.atan(rot/math.log(d))
        //TODO: fix this complex on argument thing
        var scale = math.complex(d^(sc*math.exp(i*(alfa-beta)/math.cos(beta))));
        if(math.abs(scale) < 1){
            scale = 1/scale;
        }
        var scalesize = math.abs(scale)
        var scaleangle=atan2(scale)
        document.getElementById("c").innerHTML(c);
        //document.getElementById("scaleangle").innerHTML(c+(#center-c)/scale);
        //document.getElementById("newcenter").innerHTML(scalesize*#magn);
    }
    //c is a complex
    function transform(c){
        var dist = 0;
        var dik = ARGwid/100;
        var x = 0;
        var y = 0;
        var os - 0;
        var os1 = 0;
        var pt1 = 0;
        var pt2 = 0;
        var h1 = 0;
        var h2 = 0;
        var h3 = 0;
        var h4 = 0;
        var z = c;
        if(ARGmode == "Crop"){
            if ((math.abs(math.re(z-ARGcenlarge))<ARGwid/2 && math.abs(math.im(z-
ARGcenlarge))<ARGheig/2 ) &&
            (math.abs(math.re((z-ARGcensmall)*math.exp(-i*rot)))>ARGwid*ARGfac/2 || \
            math.abs(math.im((z-ARGcensmall)*math.exp(-i*rot)))>ARGheig*ARGfac/2) ) || \
            (math.abs(math.re((z-ARGcensmall)*math.exp(-i*rot)))<ARGwid*ARGfac/2 && \
            math.abs(math.im((z-ARGcensmall)*math.exp(-i*rot)))<ARGheig*ARGfac/2 && \
            (math.abs(math.re(z-ARGcenlarge))>ARGwid/2 || math.abs(math.im(z-
ARGcenlarge))>ARGheig/2 ))
            if((math.re(c) + math.im(c)%2 == 0){


            }
        }
        var ptn = 0;
        ptn=ARGwid;
        var cnt = 1;
        while (cnt < 7){
            if (cnt==1){
                pt1=ARGcenlarge+ptn/25;
                pt2=ARGcenlarge-ptn/25;
            }else if (cnt==2){
                pt1=ARGcenlarge+i*ptn/25;
                pt2=ARGcenlarge-i*ptn/25;
            }else if(cnt==3){
                pt1=ARGcensmall+ptn/25;
                pt2=ARGcensmall-ptn/25;
            }else if (cnt==4){
                pt1=ARGcensmall+i*ptn/25;
                pt2=ARGcensmall-i*ptn/25;
            }else if (cnt==5){
                pt1=c+ptn/25;
                pt2=c-ptn/25;
            }else if (cnt==6){
                pt1=c+i*ptn/25;
                pt2=c-i*ptn/25;
            }
            test1 = (z-pt2)/(pt1-pt2);
            if(math.re(test1)<0){
                dist=math.abs(test1);
            }else if(math.re(test1)>1){
                dist=math.abs(test1-1);
            }else{
                dist=math.abs(im(test1));
            }
            dist=dist* math.abs(pt2-pt1);
            if(dist<dik/2){
                //TODO: ???
                //#solid = true
            }
            cnt = cnt + 1;
        }
    }
}
</script>
</body>
</html>
```

INPUT/OUTPUT

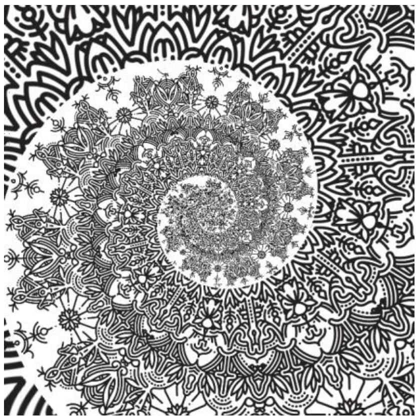
Input:



Output:



GALLERY



PAPER: *IMAGE MANIPULATION TECHNIQUES EXPLORING ESCHER AND NON-EUCLIDEAN SPACE*

NOTE:

This paper is written in Latex, which a text markup language which is often used to write theses and books. I wrote it in Latex because of the mathematical notation required and pseudocode, which are both easier to write in Latex than another word processor like Word or Pages. The Latex code can be found below the screenshots of the pdf. (See next page)

Image Manipulation Techniques Exploring Escher and Non-Euclidean Space

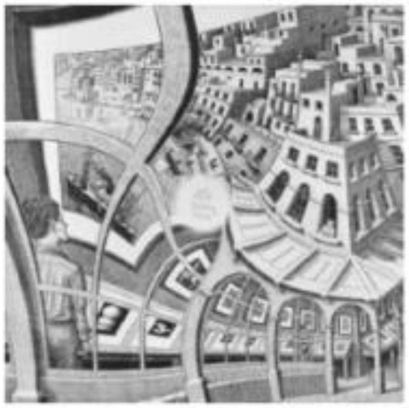
Hannah Apuan

Abstract

This paper seeks to reproduce computationally the Droste Effect image manipulation drawing from methods described by Jos Leys graphical approach, H. W. Lenstra and B. de Smits mathematical approach, and M. C. Eschers artistic approach.

Introduction

Artwork depicting non-Euclidean space with modern computational power implements the same hyperbolic geometry concepts M. C. Escher used introduced through his artwork in the mid-20th century. These concepts include the Droste Effect image transformation, which Escher implemented in 1956 with his piece Print Gallery (Fig. 1). Later, H. W. Lenstra and B. de Smit presented a mathematical approach to the Droste Effect, which Jos Ley later implemented graphically. This paper aims to use these implementations to computationally generate the Droste Effect in both 2d and 3d graphically represented spaces.



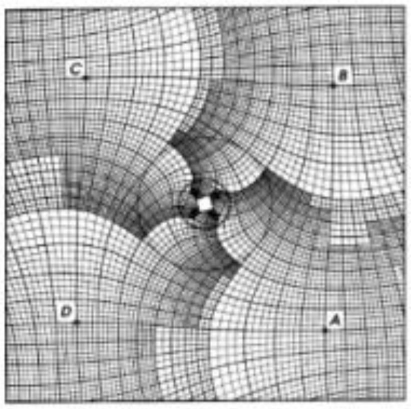


Figure 1: M.C. Escher's "Print Gallery"

Figure 2: Escher's Grid

Approach

Jos Leys explains the overall method for creating graphically the Droste Effect in three stages using Lentsras mathematical method. The same technique will be computationally applied in both two-dimensional and three-dimensional space. Stated generally, the steps are a logarithmic transformation ($z \mapsto \ln z$), rotation and scaling ($z \mapsto zfe^{i\alpha}$), and then exponentiation ($z \mapsto ez$).

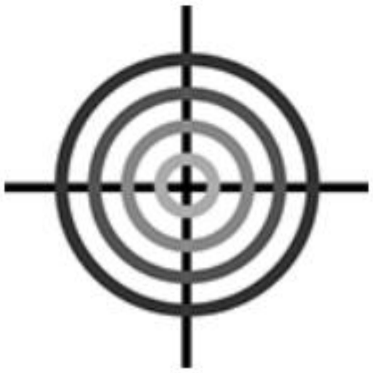
Algorithm: Droste Effect in 2D Space

Using Python 2.7.9 with Python packages Numpy, Math, and PIL (Python Imaging Library), the algorithm is presented in the three steps: transformation by log(z), rotation and scaling, and exponentiation.

The first step is to make the transformation of $z \mapsto log(z)$ This transformation is a complex logarithm function, which is inverse to the complex exponential function. Given that z and w are complex numbers, the logarithm of a z is w such that $e^w = z$. Given this information, in polar form, if $z = re^{i\Theta}$ with $r > 0$, then one logarithm of z is $w = \ln r + i\Theta$.

Polar to Linear Code

```
width = 1000
height = 1000
maxradius = np.sqrt(width**2 + height**2)/2
rscale = width / maxradius
tscale = height/(2*math.pi)
for y in range(0,height):
    dy = y - height/2
    for x in range(0, width):
        dx = x - width/2
        t = np.arctan2(dy,dx)\%(2*math.pi)
        r = np.sqrt(dx**2+dy**2)

        pixels2[int(t*tscale),int(r*rscale)] = pixels[x,y]
```



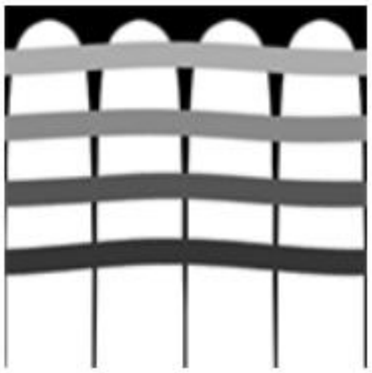


Figure 3: Before: Concentric Circles on a Cartesian Plane

Figure 4: After: Polar to Cartesian transformation $ln(z) \mapsto z$

Works Cited

1. Lenstra H, de Smith B. Escher and the Droste effect, $< http://escherdroste.math.leidenuniv.nl/$ 2. Department of Mathematics Cornell University. M.C. Escher and Hyperbolic Geometry,
$< http://www.math.cornell.edu/ mec/Winter2009/Mihai/ >$

LATEX CODE

```
\documentclass[12pt]{article}

\usepackage{lingmacros}
\usepackage{tree-dvips}
\usepackage{algorithm}
\usepackage{listings}
\newcommand\tab[1][1cm]{\hspace*{#1}}
\usepackage{amsmath,amssymb,latexsym,graphicx,verbatim}
\usepackage[noend]{algpseudocode}

\makeatletter
\def\BState{\State\hskip-\ALG@thistlm}
\makeatother

\begin{document}
\section*{Image Manipulation Techniques Exploring Escher and Non-Euclidean Space}
Hannah Apuan
\subsubsection*{Abstract}
\tab \textit{This paper seeks to reproduce computationally the Droste Effect image
manipulation drawing from methods described by Jos Ley’s graphical approach, H. W.
Lenstra and B. de Smit’s mathematical approach, and M. C. Escher’s artistic approach.}


\subsection*{Introduction}
\tab Artwork depicting non-Euclidean space with modern computational power implements
the same hyperbolic geometry concepts M. C. Escher used introduced through his artwork
in the mid-20th century. These concepts include the Droste Effect image transformation,
which Escher implemented in 1956 with his piece “Print Gallery” (Fig. 1).

    Later, H. W. Lenstra and B. de Smit presented a mathematical approach to the Droste
Effect, which Jos Ley later implemented graphically. This paper aims to use these
implementations to computationally generate the Droste Effect in both 2d and 3d
graphically represented spaces.
\begin{figure}[htbp]
    \tab \includegraphics{pg.PNG} \tab \includegraphics{grid.PNG}
    \caption{M.C. Escher's "Print Gallery"}
    \caption{Escher's Grid}
\end{figure}
\subsection*{Approach}
\tab Jos Leys explains the overall method for creating graphically the Droste Effect in
three stages using Lentsra’s mathematical method. The same technique will be
computationally applied in both two-dimensional and three-dimensional space. Stated
generally, the steps are a logarithmic transformation $( z \mapsto $ln$ z )$, rotation
and scaling $( z \mapsto zfe^{i \alpha} )$, and then exponentiation $( z \mapsto ez )$.


\subsubsection*{Algorithm: Droste Effect in 2D Space}
\tab Using Python 2.7.9 with Python packages Numpy, Math, and PIL (Python Imaging
Library), the algorithm is presented in the three steps: transformation by log$(z)$,
rotation and scaling, and exponentiation
```

```
    tscale = height/(2*math.pi)
    for y in range(0,height):
        dy = y - height/2
        for x in range(0, width):
            dx = x - width/2
            t = np.arctan2(dy,dx)\%(2*math.pi)
            r = np.sqrt(dx**2+dy**2)

            pixels2[int(t*tscale),int(r*rscale)] = pixels[x,y]
\end{lstlisting}

\begin{figure}[htbp]
    \tab \includegraphics{before1.PNG} \tab \includegraphics{after1.PNG}
    \caption{Before: Concentric Circles on a Cartesian Plane}
    \caption{After: Polar to Cartesian transformation $ln(z) \mapsto z$}
\end{figure}

\pagebreak
\subsection*{Works Cited}
1. \tab Lenstra H, de Smith B. Escher and the Droste effect, \\
\tab\tab$<http://escherdroste.math.leidenuniv.nl/〉 $
2. \tab Department of Mathematics Cornell University. M.C. Escher and Hyperbolic
Geometry,\\ \tab\tab$<http://www.math.cornell.edu/~mec/Winter2009/Mihai/>$



\end{document}
```

SOURCES

[1] Jos Leys, The Droste effect image transformation, Computers & Graphics, Volume 31, Issue 3, 2007, Pages 516-523, ISSN 0097-8493, http://dx.doi.org/10.1016/j.cag.2006.12.001.

(http://www.sciencedirect.com/science/article/pii/S0097849306002433)