A Project Report on

Analysis and Representation of a Circle in Hexagonal Grid using Python

Submitted to the Department of Information Technology

For the partial fulfilment of the B. Tech - M. Tech Dual Degree in Information Technology

by

Koushik Biswas and Prit Raj
Registration/Exam Roll number: 510814053 and 510814056
of 2014-19
B. Tech - M. Tech Dual Degree, 3rd year

Under the supervision of

Dr. Arindam Biswas

Department of Information Technology
INDIAN INSTITUTE OF ENGINEERING SCIENCE AND TECHNOLOGY, SHIBPUR

December, 2016

Department of Information Technology

Indian Institute of Engineering Science and Technology,

Shibpur

CERTIFICATE

This is to certify that the work presented in this report entitled "**Analysis and Representation of a Circle in Hexagonal Grid using Python**", submitted by **Koushik Biswas** and **Prit Raj**, having the examination roll numbers **510814053** and **510814056**, has been carried out under my supervision for the partial fulfilment of the degree of Bachelor of Technology in Information Technology during the session 2016-17 in the Department of Information Technology, Indian Institute of Engineering Science and Technology, Shibpur.

_____

Dr. Arindam Biswas
Associate Professor
Department of Information Technology
Indian Institute of Engineering Science
and Technology, Shibpur

_____--

Dr. Arindam Biswas
Head of the Department
Department of Information Technology
Indian Institute of Engineering Science
and Technology, Shibpur

_____

Dr. Amit Kumar Das
Dean (Academic)
Indian Institute of Engineering Science
and Technology, Shibpur

Date: Dec 06, 2016

Acknowledgements

The success and final outcome of this project required a lot of guidance and assistance from many people and we are extremely fortunate to have got this all along the completion of my project work. Whatever we have done is only due to such guidance and assistance and we would not forget to thank them.

We thank our mentor **Dr. Arindam Biswas**, for giving us an opportunity to work in on this project and providing us all support and guidance which made me complete the project on time . The faculty and staff of our department were helpful as well, we thank them for their support.

Lastly, we would like to thank our team members for all their insights and friends for being our moral support and encouraging us.

Date: Dec 06, 2016

Koushik Biswas
Department of Information Technology
Indian Institute of Engineering Science
and Technology, Shibpur

Prit Raj
Department of Information Technology
Indian Institute of Engineering Science
and Technology, Shibpur

Abstract

The basic problem of at hand is to represent a circle in a hexagonal grid using python and then analyse and compare the results. But, why do we even need that when we already know the more intuitive and simpler approach i.e. to represent a circle in the square grid. The square grid is quite familiar to us since we have studied it for some time, but it is not sufficient for all graphical applications.

We wanted to have some understanding of hexagonal grid because it is frequently used to build maps of strategic and war games. We were pretty interested in how these games work, so given the opportunity we decided to act on it. Thus, representing a circle in the hexagonal grid is just a way to understand the properties of the hexagonal grid.

One of many advantages of using hexagonal grid is its resemblance with the arrangements of photoreceptors in human eye while the other advantages being better pixel connectivity and higher angular resolution. The hexagonal sampling also gives less quantization error when compared to square sampling. Thus, it can be used for better edge detection as well.

As we can see that a hexagonal grid has may advantages so implementing a circle in a hexagonal grid will reveal some of those advantages as we proceed along this report.

Contents

1 INTRODUCTION

The primary advantage of a hexagonal grid over a traditional square grid is that the distance between the center of each hex cell (or hex) and the center of all six adjacent hexes is constant. By comparison, in a square grid, the distance from the center of each square cell to the center of the four diagonal adjacent cells it shares a corner with is greater than the distance to the center of the four adjacent cells it shares an edge with. The constant distance of a hex grid is desirable for games in which the measurement of movement is a factor. The other advantage is the fact that neighbouring cells always share edges; there are no two cells with contact at only one point.

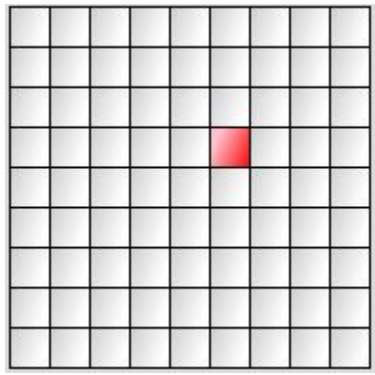
The hexagonal grid system is preferred by many researcher over the more popular square grid system because of the above given reasons. Most of the input and display devices used today uses square lattice to map images, although hexagonal lattice can have higher connectivity, higher resolution, lower quantization error and higher symmetry. As hexagons are more rounder than squares hexagonal coordinate system is much more suitable for image processing operations. This report draws light upon the efficiency of hexagonal grid system by implementing circle drawing algorithm in hexagonal coordinate system.

We will begin with a simple introduction to different types of grids and compare some of its properties. Next we will focus on hexagonal grids only and study it in detail, this will include different terms and definitions of the hexagonal along with many of its properties like neighbourhood, distance and different conventions for coordinate systems used in hexagonal grids. Finally, we would propose an approach for drawing a circle in the hexagonal grid.

2 TYPES OF GRIDS

Quite simply Grids are built from a repetition of simple shapes. The cells of the grids can be used to represent different shapes with varying degree of accuracy. Few of the major grids are described below:-

2.1 Square Grid

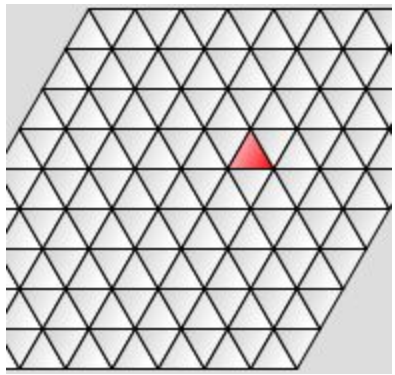


The most common grid is a square grid. It's simple, easy to work with, and maps nicely onto a computer screen. Locations can use the familiar cartesian coordinates (x, y) and the axes are orthogonal. The square coordinate system is the same even if your map squares are angled on screen in an isometric or axonometric projection.

Neighbourhood:

i) 4-n -> Common Side
ii) 8-n -> Common Vertex or Side

2.2 Triangular Grid

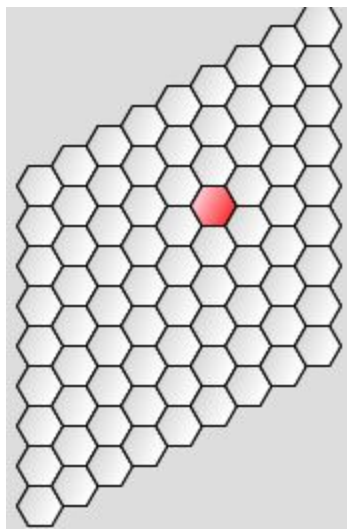


Triangles are common in 3d graphics but are rarely used for game maps. A big disadvantage of triangle maps, aside from unfamiliarity, is the large perimeter and small area (the opposite of a hexagon). In 3d graphics, triangles are the only shape that is planar; squares and hexagons can be "bent", sometimes in impossible ways.

Neighbourhood:

i) 3-n -> Common Side
ii) 6-n -> Common Vertex or Side

2.3 Hexagonal Grid



Hexagons have been used in some board and computer games because they offer less distortion of distances than square grids. This is in part because each hexagon has more non-diagonal neighbors than a square. (Diagonals distort grid distances.) Hexagons have a pleasing appearance and occur in nature (for example, honeycombs). In this project, we'll use hexagons that have pointy tops and flat sides, but the math works the same if we want flat tops and pointy sides.

Neighbourhood:

i) 6-n -> Common Side

Therefore, All Neighbours of a hexagonal grid share a side.

3 PRELIMINARIES AND DEFINITIONS

3.1 Terms

a) Neighbourhood

A neighbourhood of a point is a set of points containing that point where one can move some amount away from that point without leaving the set. In case of grids, neighbourhood of a cell means the set of cells which either share a side or a vertex with that cell.

b) Chain Code

A chain code is a more succinct way of representing a list of points than a simple [(x1, y1), (x2, y2).....]. It may be defined either with respect to the pixels or the boundaries between pixels. It is a simple integral representation of movement from pixel to pixel (cell to cell).
Example :-

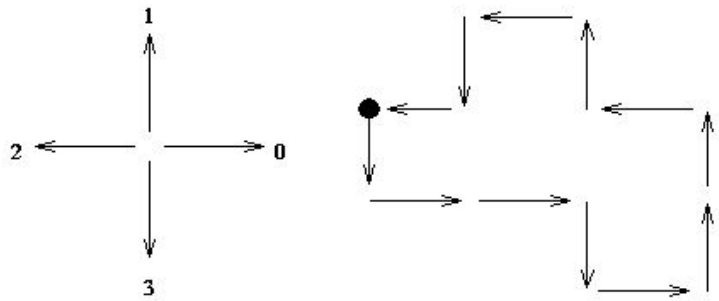


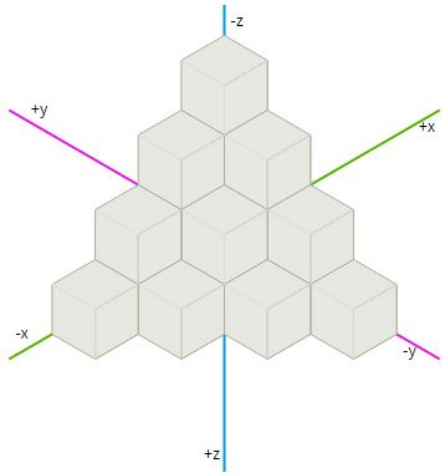
Figure 6.6 *Chain code in 4-connectivity, and its derivative. Code: 3, 0, 0, 3, 0, 1, 1, 2, 1, 2, 3, 2, derivative: 1, 0, 3, 1, 1, 0, 1, 3, 1, 1, 3, 1.*

c) Offset Coordinates

It is a type of Coordinate system for hexagonal grids. Every other column or row is offset in the horizontal or vertical direction given us a total of 4 variants:-
i) Odd - r Horizontal Layout
i) Even - r Horizontal Layout
i) Odd - r Vertical Layout
i) Even - r Vertical Layout

d) Cube Coordinates

Another way to look at hexagonal grids is to see that there are *three* primary axes, unlike the *two* we have for square grids. There's an elegant symmetry with these. The three axes x, y, z have the constraint x+y+z=0. This symmetry allows more intuitive understanding of the algorithms in hexagonal grid.



e) Axial Coordinates

The axial coordinate system is built by taking two of the three coordinates from a cube coordinate system. Since we have a constraint such as x + y + z = 0, the third coordinate is redundant. Thus Axial coordinates is just a simpler form of cube coordinates. We will be using Axial coordinates in this project.

3.2 Formulas

a) Distance

For distance in cubic coordinates:-
cube_distance(a, b) = $(|a.x - b.x| + |a.y - b.y| + |a.z - b.z|)/2$
For distance in axial coordinates:-
axial_distance(a, b) = $(|a.q - b.q| + |a.q + b.r - b.q - b.r| + |a.r - b.r|)/2$

b) Conversion of Axial Coordinates to Pixel on Screen

hex_to_pixel (hex) = $size.\sqrt{3}\left(hex.q + \frac{hex.r}{2}\right)$

c) Parametric form of Cubic Coordinates

$$x = 2 \cdot \frac{r}{\sqrt{3}}\left(\sin\ \theta + \frac{2 \cdot \pi}{3}\right)$$

$$y = 2 \cdot \frac{r}{\sqrt{3}}(\sin\ \theta)$$

$$z = 2 \cdot \frac{r}{\sqrt{3}}\left(\sin\ \theta - \frac{2 \cdot \pi}{3}\right)$$

d) Parametric form of Circle in Axial Coordinates

$$x = 2 \cdot \frac{r}{\sqrt{3}}\left(\sin\ \theta + \frac{2 \cdot \pi}{3}\right) + px$$

$$y = 2 \cdot \frac{r}{\sqrt{3}}(\sin\ \theta) + py$$

where (x, y) is axial coordinate and (px, py) is the center of the circle.

4 PROBLEM DEFINITION

Analysis and representation of a circle in the Hexagonal Grid using Python. We have to make a circle in the Hexagonal Grid using Python and any of its graphic libraries. Here we will be using Turtle a LOGO like graphic library which follows only the most basic of commands like rotation and linear movement.

5 PROPOSED APPROACHES

We have used two different algorithms in drawing the circle. One is a naive approach based on the parametric form of the circle in the Hexagonal Grid, the other is using the Bressenham Algorithm

5.1 Naive Approach

.

We simply plot the (x, y) coordinates on the hexagonal grid for full 360 degrees rotation using the parametric form of the Circle.

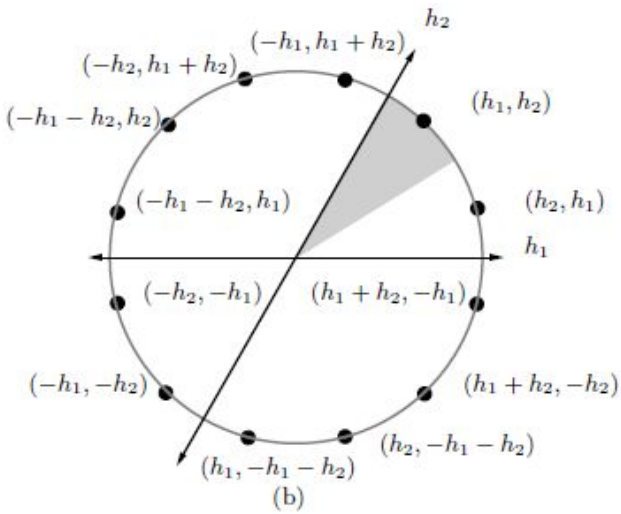
Algorithm 1 Drawing a Circle using the parametric equation of a Circle

---

Input: r -> radius, (px, py) -> center of circle
Output: A set of points along the circumference of Circle
Steps of the algorithm:
1: Declare x, y
2: **for** t = 0 to 360 $\in V$ **do**
3: $x = \left[2 \cdot \frac{r}{\sqrt{3}} \left(\sin\ \theta + \frac{2 \cdot \pi}{3}\right)\right] + px$
4: $y = \left[2 \cdot \frac{r}{\sqrt{3}} (\sin\ \theta)\right] + py$
5: draw_hex(x, y)
6: **end for**

5.2 Bressenham Algorithm

.We cannot display a continuous arc on the raster display. Instead, we have to choose the nearest pixel position to complete the arc. Thus we approximate the circle by choosing the next best point on the arc so that our circle is continuous and a good approximation of a perfect circle. Finally we can get the Chain Code of the arc constructed using this method.



The number of symmetrical points generated on a circle in hexagonal grid is 12. Since, a complete circle can be drawn by exploiting the symmetry it suffices to detail how to draw an arc of 30 degrees as shown in the figure.

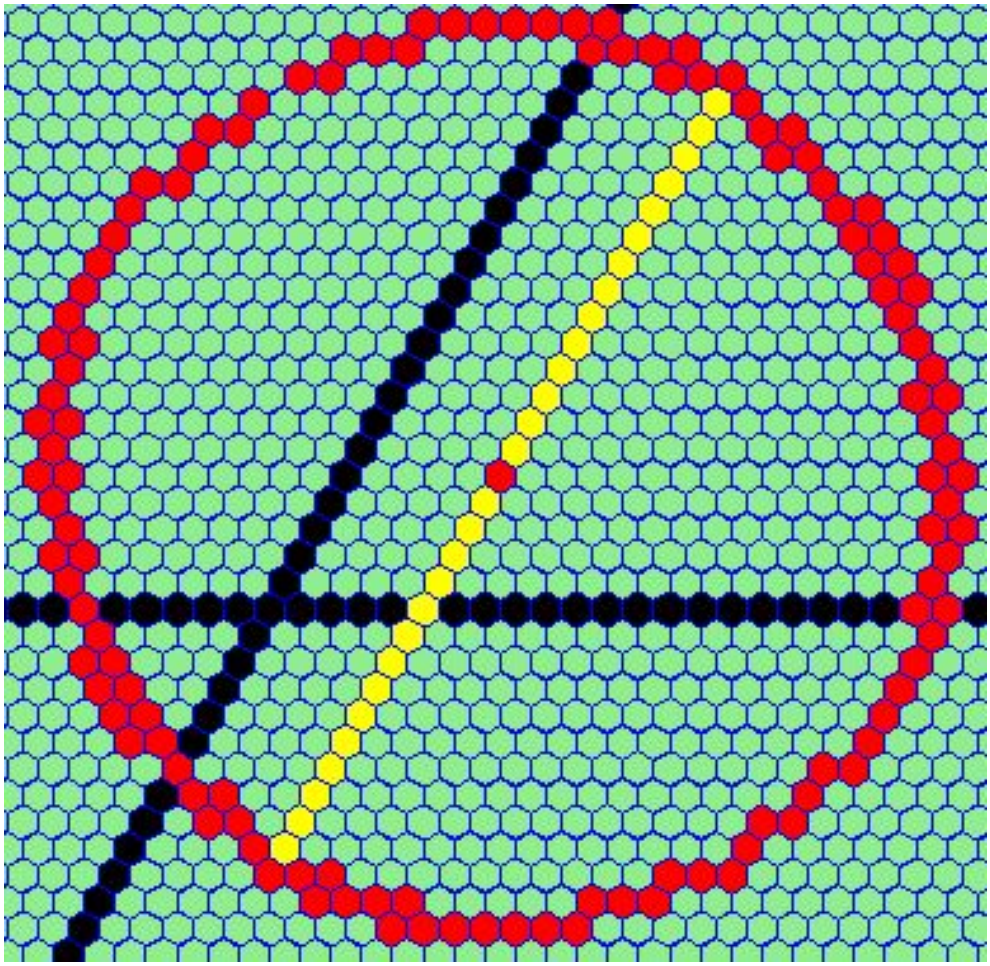
Algorithm 1 Drawing a Circle using Bressenham Algorithm

Input: r -> radius, (cx, cy) -> center of circle
Output: A set of points along the circumference of Circle
Steps of the algorithm:

```
1: x=0, y=r, p=1/4
2: while y>=x do
3:     draw_hex(cx+x, cy+y)
4:       if p<0 then
5:               p=p+2*x+y+5/2
6:       else
7:               p=p+x-y+11/4
8:               y=y-1
9:       end if
10:      x=x+1
11: end while
```

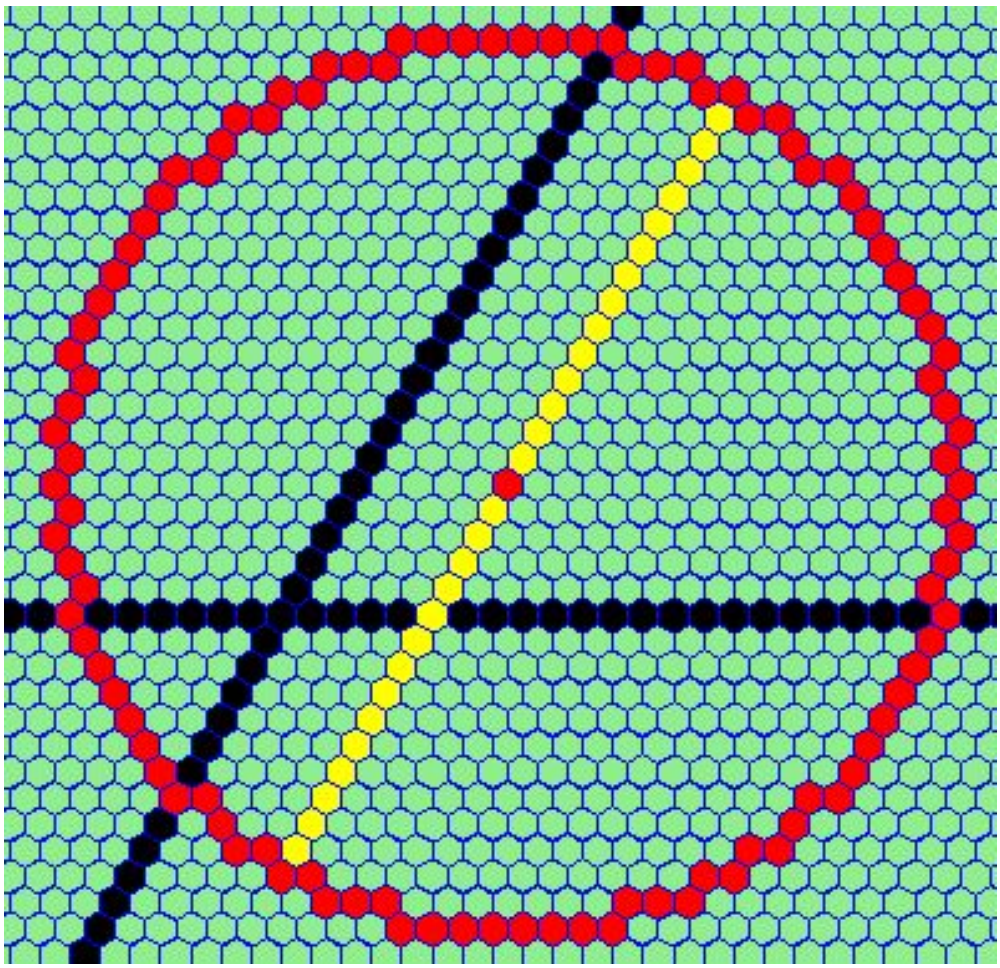
6 EXPERIMENTAL RESULTS

6.1 Result for Naive Approach



The circle shown in the above figure has a center at (5, 5) and a radius of 15. It was drawn using the naive approach. As we can see the naive approach doesn't work. There are two mistakes in the above circle - i) The circle is not fully connected at one point there is a gap. ii) There are more than 2 neighbours of a point on the circumference at certain places, this should not happen. The reason we get the above faults in this approach is because we are taking the floor of the floating point values obtained for each angle from 0 to 360 degrees. So at certain angles more than 1 cell becomes a good approximation of the circle and in this case we select both these cells. This approach is slower too because it requires a lot of floating point operations.

6.2 Result for Bressenham Algorithm



The circle shown in the above figure has a center at (5, 5) and a radius of 15. It was drawn using the Bressenham Algorithm. It is a significant improvement over the naive approach as we can see in the above figure the faults of the previous naive approach has been corrected. There is no disconnectedness or more than 2 neighbours of a single cell on the circumference of the circle. This is because we only allow two types of movement while selecting the next cell i.e. (x, y) -> (x+1, y-1) or (x+1, y) . Thus we only move from one cell to its neighbour. Also in this algorithm we only do integer operations so, it is much faster than the previous approach.

7 CONCLUSIONS

We have drawn a circle in hexagonal using two different methods, the parametric form of the circle and the Bressenham Algorithm. From the results obtained we can easily conclude that the Bressenham Algorithm is better than the two. There is no disconnected links and the distance from the center of the circle to the circle is constant, i.e. there not more than two neighbours of any cell on the circumference of the Circle.

But more importantly we realised that a circle can be more accurately represented in the hexagonal grid than in a square grid. This is due to the increased symmetry of the hexagonal grid.

References

[1] Lee Middleton, Jayanthi Sivaswamy, and Professor Sameer Singh, *Advances in Pattern Recognition* Springer, 2005.

[2] Herman Tulleken, Hex Grid Geometry for Game Developers, 2015

[3] Innchyn Her, Geometric Transformations on Hexagonal Grids, IEEE Transactions on Image Processing , VOL. 4, NO. 9, September 1995

[3] URL: "http://www.redblobgames.com/grids/hexagons" , Hexagonal Grids

[4]URL: "http://www-cs-students.stanford.edu/~amitp/gameprog.html#hex" Amit's game blog

Appendix I
Source Code of Bressenham Algorithm

```
import collections
import math
import turtle

SIZE = int(input("Size of each hexagon in the grid: "))
COOD = int(input("Grid size: "))
RAD = int(input("Radius of the required circle: "))
CX, CY = map(int,input("Center of Circle: ").split())

chain_code=[]

Point = collections.namedtuple("Point", ["x", "y"])
Hex = collections.namedtuple("Hex", ["q", "r"])

def hex_to_pixel(hex):
    x = SIZE * math.sqrt(3) * (hex.q + hex.r/2)
    y = SIZE * 3/2 * hex.r
    return Point(x, y)

sc=turtle.Screen()
tr=turtle.Turtle()

def init_turtle():
    tr.hideturtle()
    sc.bgcolor("lightgreen")
    sc.title("Hex")
    tr.color("blue")
    tr.pensize(1)
    tr.fillcolor('red')

def draw_hex(hex, fill):
    tr.penup()
    tr.seth(0)
    tr.setpos(hex_to_pixel(hex))
    tr.forward(SIZE)
    tr.left(90)
    tr.pendown()
    if fill:
        tr.begin_fill()
    for i in range(6):
        tr.forward(SIZE);
        tr.left(60)
    if fill:
```

```
        tr.end_fill()

def draw_hexgrid():
    sc.tracer(0, 0)
    for i in range(-COOD, COOD):
            for j in range(-COOD, COOD):
                    draw_hex(Hex(i, j), False)
    tr.fillcolor('black')
    for i in range(-COOD, COOD):
        draw_hex(Hex(i, 0), True)
    for j in range(-COOD, COOD):
        draw_hex(Hex(0, j), True)
    sc.update()
    sc.tracer(4,20)

def draw_sym(x, y):
    draw_hex(Hex(CX+x, CY+y), True)
    draw_hex(Hex(CX+y, CY+x), True)
    draw_hex(Hex(CX+x+y, CY-x), True)
    draw_hex(Hex(CX+x+y, CY-y), True)
    draw_hex(Hex(CX+y, CY-x-y), True)
    draw_hex(Hex(CX+x, CY-x-y), True)
    draw_hex(Hex(CX-x, CY-y), True)
    draw_hex(Hex(CX-y, CY-x), True)
    draw_hex(Hex(CX-x-y, CY+x), True)
    draw_hex(Hex(CX-x-y, CY+y), True)
    draw_hex(Hex(CX-y, CY+x+y), True)
    draw_hex(Hex(CX-x, CY+x+y), True)

def draw_circle(r):
    x, y=0, r
    d=0.25
    while x<y:
        draw_sym(x, y)
        x+=1
        if d<0:
            d+=2*x+y+2.5
            chain_code.append(0)
        else:
            y-=1
            d+=(x-y)+2.75
            chain_code.append(1)
    draw_sym(x, y)

def draw_diameter():
    tr.fillcolor('yellow')
    for j in range(1, RAD):
        draw_hex(Hex(CX+0, CY+j), True)
        draw_hex(Hex(CX+0, CY-j), True)

init_turtle()
```

```
draw_hexgrid()
tr.fillcolor('red')
draw_hex(Hex(CX, CY), True)
draw_circle(RAD)
draw_diameter()
print("Chain Code: ")
print(chain_code)
sc.mainloop()
```