PHYS-4007/5007: Computational Physics

Tutorial #9
Animation Using Turtle

1 Introduction: Creating a Sample Code

Log into your Linux account and open a Linux terminal window. Now cd to your python subdirectory and make a subdirectory called animation, then change directory to this subdirectory:

```
> cd python
> mkdir animation
> cd animation
```

Open a new file called "myturtle.py" (without the double quotes) with emacs at the Linux prompt:

```
> emacs myturtle.py &
```

where the ampersand symbol (&) puts the emacs session in background so that you can still enter commands at the Linux prompt.

In your emacs editor GUI, type in the following lines:

```
import time
import turtle
from turtle import *

# Create a screen object
screen = turtle.Screen()
screen.bgcolor("lightblue")

# Create a turtle object
pen = turtle.Turtle()
pen.shape("turtle") # Change the turtle's shape
pen.color("green")
pen.pensize(3)

# Draw a square
```

```
for _ in range(4):
    pen.forward(100)
    pen.left(90)

# Now draw a geometric pattern
for steps in range(100):
    for c in ('blue', 'red', 'green'):
        color(c)
        forward(steps)
        right(30)

# Keep the window open until manually closed
turtle.done()
```

Now run this code with:

```
> python3 myturtle.py
```

Note how this code creates figures using animation.

2 The Turtle Package

Python Turtle is a built-in Python library that provides a simple and engaging way to introduce programming concepts, especially for beginners. It allows users to control a virtual "turtle" that draws on a canvas, much like a pen on paper.

Key Features and Concepts:

- **Turtle Object:** The central element is the Turtle object, which represents the drawing entity. It has a position, an orientation (direction), and a pen.
- **Pen Attributes:** The pen has attributes like color, width, and an on/off state (referred to as down() and up()). When the pen is down(), the turtle draws as it moves; when it's up(), it moves without drawing.
- **Movement Commands:** The turtle responds to commands relative to its own position, such as:
  - forward(distance): Moves the turtle forward by a specified distance.
  - backward(distance): Moves the turtle backward.
  - left(angle): Turns the turtle left by a specified angle in degrees.

    - right(angle): Turns the turtle right.

- **Drawing Commands:**
    - circle(radius): Draws a circle.
    - dot(size, color): Draws a dot.
    - begin_fill() and end_fill(): Used to fill shapes with a specified color.
- **Screen Object:** The Screen object represents the drawing window or canvas. It allows for setting background color, adding shapes, and controlling the overall display.

3 Additional Shapes and Printing Text

Go back to your animation.py code and enter the following coding just prior to the turtle.done() command:

```
# Send your turtle back to its starting-point
# (useful if it has disappeared off-screen):
home()

# Home is at (0, 0).

# Wait a few seconds before clearing the screen
time.sleep(2)

# Clear the window so we can start anew:
clearscreen()

# Draw a star shape using red lines, filled in with yellow:
color('red')
fillcolor('yellow')

# Just as up() and down() determine whether lines will be drawn,
# filling can be turned on and off:
begin_fill()

# Next well create a loop:
while True:
    forward(200)
    left(170)
    if abs(pos()) < 1:
        break

# abs(pos()) < 1 is a good way to know when the turtle
# is back at its home position.

# Finally, complete the filling:
end_fill()

# Wait a few seconds before clearing screen
time.sleep(2)

# Clear the window so we can start anew:
clearscreen()

# Move the position of the cursor
turtle.setx(0)
turtle.sety(200)

# Clear the window so we can start anew:
clearscreen()

# Let the user know the program is done
turtle.write("This animation is done.", align="center",
             font=("Courier", 20, "bold"))
```

Remember, the last line of your code should be the turtle.done() command.

4 Carry On Yourself with Turtle

Until the end of class time, do a web search of "python turtle" to find useful turtle commands to draw additional shapes and carry out other commands in turtle. For instance, try to create the following animations:

- Draw a 3-dimension cube and have it tumble across the screen.
- Draw a 3-dimensional sphere and have it rotate.
- Using your "orbits.py" Python code, animate a planet's orbit about the Sun.