CS455 Assignment 3

Anand Singh Kunwar (13110)

October 2016

1 Use-Case/Purpose

This code is a very crude example of a graphic rendering engine. However for the simplicity of the problem and demonstration we aren't dealing with graphics and we render text-animation. Here in the example we render the famous ROFLCopter animation as a series of Frames. We have 2 files *IoC.py* and *Traditional.py* each corresponding to its title.

Listing 1: IoC.py

```
import time
class Frame(object):
    def __init__(self, Frame):
        self.Frame = Frame

class Engine(object):
    def __init__(self):
        # Default Frame Rate is 24 fps
        self.FrameRate = 24
        self.Scene = []
        self.Loop = True

    def AddFrame(self, frame):
        """For Adding a Frame to the Scene"""
        self.Scene.append(frame)

    def RenderFrame(self, frame):
        """A Crude Frame Renderer"""
        print frame.Frame

    def Render(self):
        """Render Function to be called by main"""
        while True:
            for frame in self.Scene:
                time.sleep(1.0/self.FrameRate)
```

```
                print '\n'*50    # Clearing Frame Hack
                self.RenderFrame(frame)
            if not self.Loop:
                break
def main():
    engine = Engine()

    ROFLCopter1 = open('roflcopter1', 'r').read()
    ROFLCopter2 = open('roflcopter2', 'r').read()

    engine.AddFrame(Frame(ROFLCopter1))
    engine.AddFrame(Frame(ROFLCopter2))
    engine.Render()

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

Listing 2: Tradition.py

```
import time
class Frame(object):
    def __init__(self, Frame):
        self.Frame = Frame
def main():
    ROFLCopter1 = open('roflcopter1', 'r').read()
    ROFLCopter2 = open('roflcopter2', 'r').read()
    Scene = []
    FrameRate = 24
    Loop = True

    # Constructing Scene
    Scene.append(Frame(ROFLCopter1))
    Scene.append(Frame(ROFLCopter2))

    # Rendering Here
    while True:
        for frame in Scene:
            time.sleep(1.0/FrameRate)
            print '\n'*50    # Clearing Frame Hack
            print frame.Frame
        # Loop Condition Check
        if not Loop:
            break

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

2 Advantages of Inversion of Control over Traditional

Although the code may seem more in our application in the case of Inversion of Control, this adds quality attributes like testability, modifiability etc. We can have default configurations in our Engine Part. If we already have the Engine/Framework ready we can make our application (here the ROFLCopter animation) with just lesser lines of code in comparison to our Traditional Way. The control is with the Engine/Framework. The drawing/rendering part is all handled by the engine and when to call what. We only supply the configurations/scenes and the rest of the jobs are handled by the framework itself. There is even an option of default configuration built into the engine which can be altered if wanted. Hence this provides an easy to make application by using the Engine without much configuration and code (Look at the main block).