1.3

Gaige Ehrenworth

December 2019

1 Definition of Object Oriented and Imperative Programming Paradigms

By definition, the Imperative Programming paradigm uses statements to change the programs state. This is simple and a feature of languages such as C and Visual Basic. The paradigm focuses on imperative commands that explicitly accomplish some task in the program. There is also the Object Oriented paradigm which deals with data as objects with parameters, methods, and attributes. This paradigm is a feature of languages such as Python, C++, and Java. In a way, Object Oriented programming is a subset of imperative programming in terms of execution; it is distinct in terms of data. Both of these paradigms can be easily implemented in the C++ programming language, which was used for the completion of the assignment.

2 The Imperative PPM

For outputting the "hello world" equivalent of a PPM, the code needed was fairly simple:

```
cout << "P3\n" << 256 << " " << 256 << "\n255\n";
for(int r = 0; r < 256; ++r) {
    for(int g = 0; g<256; ++g) {
        cout << r << " " << g << " " << 120 << "\n";
    }
}
```

The head of the file is first output, then it is followed by the necessary RGB values for each cell on the image. The code is executed sequentially with consistent output for the values coded. All that's doing the heavy lifting is the for loops and the output is concatenated to a file from the command line. As for data, primitives such as integers are used instead of abstract types.

3 The Object Oriented PPM

Since Object Oriented programming is all about the object representation model, I first abstracted a class PPM:

```
class PPM {
    public:
    PPM();
    PPM(int, int);
    ~PPM();
    void makeHead();
    void makeBody();
    friend ostream& operator<<(ostream& os, const PPM& ppm);
    private:
    int imageSideLength;
    int blueValue;
    string body;
};
```

The needed variables from creating a PPM are attributes of the class while the instructions for creating the file from these are class methods. The while PPM file is ultimately stored as a string of the class to be later printed out via the overloaded ¡¡ operator. The object is created in memory at runtime, being further developed upon the calling of methods makeHead (the creation of the file text that defines image size) and makeBody(the creation of file text that contains RGB data. Multiple instances of a PPM can be created in memory, either with user defined instance variables or default parameters, where they can be further accessed independently from one another.

4 Comparison

The Imperative program acts like a script to a play: it's static and unchanging. The PPM is not really created, but outputted and must be assigned externally (ppm.exe ¿¿ output.ppm) lest it be lost. With the Object Oriented program, spaces in memory are assigned the data necessary for a PPM file. Multiple PPM files can be created with this program and be manipulated independently whereas this is not possible with the imperative program.

5 Conclusion

At first, it was daunting to compare Imperative and Object Oriented paradigms, especially when using the same language for both programs. However, the two programs did create an "apples vs oranges" scenario when compared. Object Oriented was executed in Imperative Style, but was its paradigm was all about the storing an manipulation of data in memory. The Imperative paradigm can be executed as many times as needed, but each time it is needed the program must be executed again. With the Object Oriented Paradigm, the program can be executed once and all created objects be accessed via their space in memory. Due to the imperative nature of OOP execution, it seems that Imperative Paradigm is a super-set of the Object Oriented Paradigm