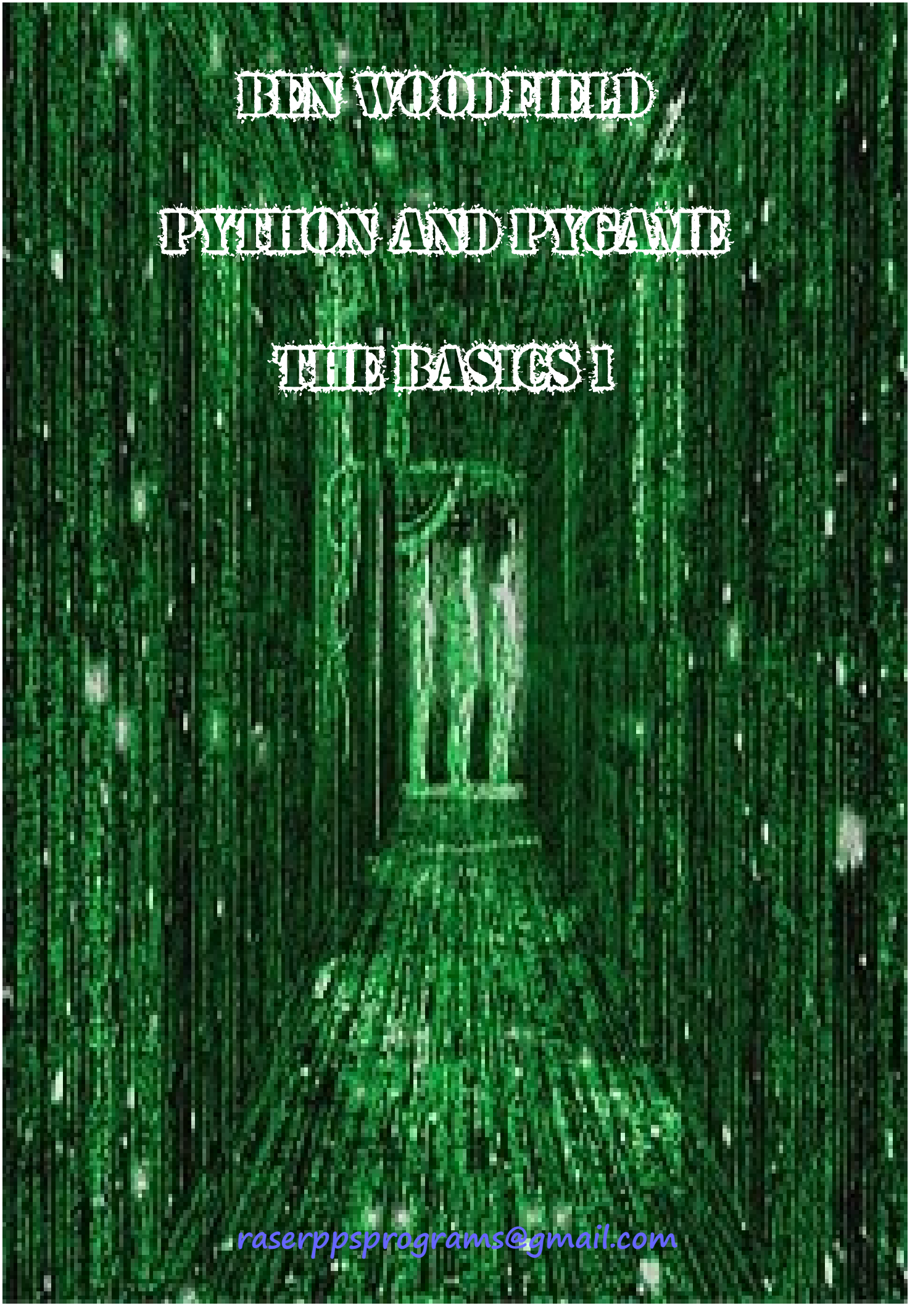
BEN WOODFIELD
PYTHON AND PYGAME
THE BASICS I
raserppsprograms@gmail.com

WELCOME TO PYTHON AND PYGAME THE BASICS 1

INTRODUCTION

Firstly, Well Done! For making the decision to learn and better yourself – I am by no means an expert, but I will share all that I know with you, in an understandable and easy to follow method.

All coding for this booklet is freely available – and should be kept together – if you have this booklet and no code, copy the following link into your browser address bar:

https://github.com/raseribanez/Python-Programming-Community---Pygame-Booklets/upload/master

I use Python 2.7.11 to write and test the code in this booklet

In this booklet I will cover the beginning steps of all game development. We will make a template program that can be used as a starting point for any games you co on to write.

I will try to understand each line of code in the template.

I have a few of these templates for different areas of Python, and later on they will save you a lot of time...however...at the early stages you should be getting familiar with typing these templates out manually – in the long run it will help you...if you just copy copy copy, paste paste paste, you won't really get a feel for the syntax and structure of Python.

Anyway....Lets get started,

Scroll to the next page.

CHAPTER 1: THE END

Well that was easy...What I mean by the end is – Here is the code for the end goal we are working towards. This is the template we are going to make with this booklet – I find it always helps to visually see the program first, then look at how it is written.

```
# Import the Pygame module into Python
import pygame, sys
from pygame.locals import *

# Initialise the game
pygame.init()

# Configure window size
DISPLAYSURF = pygame.display.set_mode((400, 300))

# Set a main title / caption
pygame.display.set_caption('A Template Program')

while True: # main game loop
  for event in pygame.event.get():
      if event.type == QUIT:
        pygame.quit()
          sys.exit()


pygame.display.update()  # Update the screen
```

I will explain each line in a little more detail.

The very first lines of code reads:

```
import pygame, sys
from pygame.locals import *
```

If you're familiar with Python already you may have come across importing modules. If not this is how you typically import Pygame into a Python program. Why import? If Python was crammed full of every module, and those modules weren't importable (loaded up every time) you'd have very slow programs, and Python itself would be a lot more dense. As it is, it is lightweight, easy to download, and takes just a minute to install. It is much more efficient to have modules saved elsewhere and call them into programs when needed.

We use the traditional method of importing for 'sys' and 'pygame', then in the second line we are importing a function from inside a module, this method replaces the need to type the module and the function names – like in regular Python. As you add things to your game you will need other modules, but for now this will get you going.

Next line - Initialization:

```
pygame.init()
```

This will initialize the modules within Pygame.

You won't always need to initialize every Pygame module for everything you do, for now it is simple so we will use this method, but if you just wanted to import a particular module from Pygame – lets say 'font' you *could* do it like this:

```
pygame.font.init()
```

Now to set the window size – we are storing it into a variable. And for now we don't need to be too fussy about the windows exact size, but later on, when you start to add backgrounds and graphics to your game, you will need to be able to set these values accordingly...NOTE the double parentheses ((brackets)).

```
DISPLAYSURF = pygame.display.set_mode((400, 300))
```

We have created the windows size by giving it the values of X=400, and Y=300. We also created a **Variable** called 'DISPLAYSURF'. It is more efficient sometimes to put something into a variable (or give it a name of our own). Doing this allows us to store pretty complex calculations or settings into a single word or variable name for use later on. The next time you want to use that line of code you can just call it into something else by using the variable name or DISPLAYSURF in our example. Here's the next line – we are setting a title:

```
pygame.display.set_caption('A Template Game Example')
```

The **String** value we set here will be shown in the top of the window – as the main Title, or name of your game. The only thing you need to worry about with this, is that your Title actually fits within the width of the window (the values we just looked at – 400 x 300).

Now, the main loop – to run what we have so far...

```
while True:
  for event in pygame.event.get():
      if event.type == QUIT:
        pygame.quit()
          sys.exit()
```

We use a **While Loop** to keep the game from quitting – this keeps it going until told otherwise. The only way the program will exit this loop is with a **Break** statement.

If we were to put this loop inside a **Function** we could also exit the loop with a **Return** statement.

This while loop can also be known as a *Game Loop* or a *Main Loop.* The main purposes of a Game Loop are the following:

- Handles **events** within the game
- Updates the **Game state**
- Draws the game state to our game screen

Now for the final line of code:

```
pygame.display.update()
```

Basically – each time Python runs through our while loop we just created, it updates by displaying what it is told to – for now that is a big old nothing – which Python shows as an empty black screen.

SUMMARY

Well Done for getting this far, and an even bigger well done if you wrote out each line of code, while working through each step of the booklet.

This might not seem like a lot, but we have covered a lot here in a few lines of code .

- `Importing modules`
- `Initializing modules`
- `pygame.display.set_mode()`
- `pygame.display.set_caption()`
- `pygame.display.update()`
- `While loops`
- `For loops`
- `If statements`

This template will be all you need to get started with some simple graphics and effects.

In a later chapter we will upgrade our template for more complex game development, and add classes and functions to our code. I will also show how to efficiently structure a filesystem in order to package a game properly for distribution and testing.

See you in Chapter 2!