School of Electronic

Engineering and

Computer Science

Final Report

Programme of Study:

Bsc Computer Science

Project supervisor:

Dr William Marsh

Project title:

Pygame Simplified

Student name:

Fatima Abukar

Date: 22/04/2014

Abstract

Many students aged fourteen to sixteen who take the General Certificate of Secondary Education (GCSE) in Computer Science find programming difficult. Teachers find it difficult to engage and motivate students into programming with the current set of resources provided by programming languages, to build motivating programs such as computer games. Rick Barnes, a teacher of computing at Bedale High School echoes this. He commented on a forum for computing teachers, Computing at Schools (CAS): *"I have a significant member of pupils in my GCSE classes that think it (programming) is hard…Most of the tutorials they find online are harder to follow when they struggle to engage with the real world problem."* (Barnes, 2014).

This project is a teaching project, it provides resources for teachers that engage and simplify programming for GCSE students who program in Python. The most popular teaching resource for teachers: Pygame, Turtle Graphics for TK Python and Greenfoot all have limitations. In short, to write programs in Pygame, Object Oriented Programing, this is not a GCSE concept, there is limited scope on what on you can achieve with Turtle Graphics and Greenfoot a popular game API is for Java.

This project, Pygame Simplified provides a simplified API for the Pygame library. This report covers the background and research of resources used by teachers. First, GCSE requirements are identified. The limitations of the three main popular API/libraries used by teachers are outlined. A comparative analysis of Pygame is made with Greenfoot where a number of goals, which inspire the design the Pygame Simplified API, are drawn. These goals are expanded where requirements for Pygame Simplified is set. The requirements are designed and implemented.

Pygame Simplified is refined with testing and evaluation by teachers teaching GCSE Computer Science. Teachers give suggestions and improvements for the Pygame Simplified API based on its suitability for GCSE students. There are two versions to Pygame Simplified, Pygame Simplified version I and Pygame Simplified II. A program code generator (suggested by a teacher) for the Pygame Simplified API is also implemented. This helps beginner GCSE programmers. Feedback from teachers suggests the Pygame Simplified API is a useful and engaging resource, which simplifies programming for GCSE students. Students are encouraged to apply GCSE programming constructs and are motivated with a variety of games that can be programmed using the Pygame Simplified API.

Table of Contents

Tables

Figures

Chapter 1 Introduction

1.1 Context

In September 2012, Michael Gove (the current education secretary for the government) set to replace GCSE Information Communication Technology (ICT) with Computer Science, in the GCSE curriculum (Tickle, 2012). ICT is a qualification that heavily focuses on the use of and application of technology, whereas Computer Science focuses building computer systems, which can be used. (CAS working group, 2012)

Students who follow GCSE Computer Science find programming difficult due to the lack of resources available to engage them into programming. Performing basic input and output operations in a text editor disengages many students from programming. Sweigard an author of *'Inventing Computer games with Python'* echoes this. *"Programming isn't hard. But it is hard to find learning materials that teach you to do interesting things with programming."* (Sweigart, 2003).

GCSE constructs are easier to apply to Python programs. Pygame is a library for Python where computer games can be programmed, however, applying GCSE constructs to a game using Pygame is difficult in Pygame. Objectives and aims that simplify programming computer games (using Pygame) for GCSE students are set.

1.2 Objectives and Aims

This project looks at providing resources for teachers to engage motivate and simplify programming for students. The Pygame Simplified API ensures students are able to develop their programming skills through programming computer games.

The objectives of this project:

- Produce an API suitable for GCSE students, which allows students to program computer games using GCSE constructs.
- Provide an API that can be used by teachers to help students with controlled assessments.
- Improvements of the Pygame Simplified API based on feedback from evaluators.
- Provide a set of resources for Pygame Simplified to teachers teaching Computer Science. This includes:
    - Documentation of the API
    - Worksheets for Pygame Simplified
    - Tutorials
    - Example applications built using the Pygame Simplified API

1.3 Motivation

I started programming in the procedural programming module in my first year of University. I learnt to program in Java. As a side-learning tool to help us grasp aspects of programming with an engaging outcome, it was recommended to use Greenfoot. Greenfoot is an API for Java, which allows you to program computer games. I tried using Greenfoot, but after a couple of hours I realised my programming ability was not sufficient. Greenfoot heavily

focuses on Object-Oriented programming; this goes beyond the requirements in this module. I was demotivated to make games In Greenfoot, due to my lack of programming ability. I started learning programming the dull way, typing simple programs into a text editor. As my programming ability developed, my motivation for programming came from creating interesting applications such as games. This helped me grasp the core programming concepts as well inspired me to program something interesting and useful.

When programming a game, you can see exactly what is happening and why. Programming constructs are applied to real world applications. This is something that is not visible in programming simple input and output programs in a text editor. This project aims to remove this problem for all students. As of yet there are no API's, which for Python that allows you to program computer games where GCSE constructs are easy to apply.

1.4 Report Outline

This report is split into ten chapters. Each chapter examines the different stages of Pygame Simplified.

Chapter 2 covers the background and research for this project. It examines GCSE requirements, why Python is the best language for this project and the API's and libraries used by teachers as a teaching resource.

Chapter 3 gives a comparative analysis of Greenfoot and Pygame, the successes and limitations these two programs are detailed using a sample game programed by the author, from these goals for the Pygame Simplified API are drawn.

Chapter 4 sets the function and non- functional requirements for the Pygame Simplified API and program code generator.

Chapter 5 covers the design and implementation of Pygame Simplified version I and Pygame Simplified version II. The design and implementation of Pygame Simplified's program code generator is also covered.

Chapter 6 introduces a range of games that can be programmed using the Pygame Simplified API. A comparison of the sample game introduced in *chapter 3* is made.

Chapter 7 covers the evaluation of Pygame Simplified it details the process of evaluation, the tools used to evaluate Pygame Simplified and how Pygame Simplified meets the objectives set in Chapter 1. Limitations and future work of Pygame Simplified are outlined.

Chapter 8 concludes this project. This chapter covers the challenges faced in this project and what I have learnt from the project and.

Chapter 9 has the references for this report.

Chapter 10 has the appendixes for Pygame Simplified a link to supporting material is given.

Chapter 2 Background and Research

This chapter provides background and research of GCSE Computer Science and the APIs/ libraries used by teachers to teach Computer Science GCSE students. Section 2.1 introduces the requirements for GCSE Computer Science. Section 2.2.1 examines why Python is suitable choice to teach programming to GCSE students. Sections 2.2.2 and 2.2.3 looks at current libraries in Python, which are used as resources to teach programming, the limitations of these libraries are outlined. Section 2.3 looks at Greenfoot, a popular computer games API.

2.1 GCSE Computer Science

Many students aged fourteen to sixteen across England follow GCSE Computer Science. The GCSE curriculum set by the government offers a comprehensive guide on the level of understanding students must have to complete their course. Assessment Quality Alliance (AQA), Edexcel and Oxford Cambridge and RSA (OCR) are exam boards, which provide assessments and examinations to GCSE students. Table 1 (below) illustrates the releases of each exam board. OCR has provided GCSE Computing since 2011 (OCR, 2014) Edexcel and AQA have introduced GCSE Computer Science to fit the new curriculum introduced by the government. In this report, 'GCSE Computer Science' is the adopted term to refer to Computing and Computer Science.

Examination board	**Information**
Oxford Cambridge and RSA (OCR)	GCSE Computing, latest specification 2012.
Assessment Quality Alliance (AQA)	GCSE Computer Science, first teaching 2014.
Edexcel	GCSE Computer Science, first teaching 2013.

Table 1: Information on GCSE examination boards.

The introduction of GCSE Computer Science means there is a short supply of teachers with Computer Science qualifications (Tickle, 2012). ICT teachers take part in projects such as the Teaching London Computing Project (TLCP). TLCP offers an intensive project, workshops and resources to help prepare ICT teachers for the GCSE Computer Science curricular[1].

Computing at Schools (CAS), is a forum designed for teachers closely works with examination boards such as Edexcel to provide a network for teachers to converge ideas on the approaches to teaching Computer Science. Many new teachers whom take courses such as TLCP use CAS to share resources that is useful to the Computer Science curriculum produced using APIs and libraries available in their chosen programming language.

2.1.1 GCSE requirements

A set of requirements is drawn from all three-examination boards that provide GCSE Computer Science. Teachers teaching Computer Science must follow these.

The fundamental aspect that all three-examination boards require students to understand is how to solve problems (AQA; OCR; Edexcel; 2014). Students are required given a set

[1] '*Teaching London Computing Project*' (2014), Website: http://teachinglondoncomputing.org/ Accessed on 10/03/2014 5

problem to understand, define and solve the problem using their own algorithm. Students must be able to understand how to use GCSE programming constructs such as: variables and type declarations, repetition, command sequences and selection to perform basic input and output programs (Edexcel, 2014). Command sequences include conditions such as `if-statements` and `switch` statements. Understanding and application of repetition includes `for-loops` and `while loops`. Students must also know how to use data structures that hold large quantities of data (such as arrays). A good structure of program code is also emphasised by all three-exam boards. Program code must be well documented and readable. (OCR, 2014).

Students are examined on their application of these requirements through a controlled assessment (provided by each exam board). Table 2 (below) shows a brief description of the controlled assessments and tasks these exam boards provide and programming languages schools can adopt. All three-examination boards set computer games as a controlled assessment task.

Examination Board	**Options of Programming languages**	**Task**
Edexcel	Python Java C- derived language	Program a computer game.
OCR	No restrictions on programming languages.	Design and program a computer game.
AQA	Python Delphi Java C – derived language	A choice between: • A computer game application • A mobile phone application • An application for a client (a gardening application).

Table 2: Information on examination boards for controlled assessments

2.2 Python

Python is a suggested programming language by exam boards for controlled assessments (table 2, above shows this). It has many benefits to other programming languages such as C derived languages and Java. GCSE constructs are easy to apply. This makes it an attractive choice for many schools to teach it to GCSE students:

- Python forces indentation on the user encouraging good program structure.
- Python is a loosely typed programming language providing ease for students.
- Python uses three times less program code other programming languages such as Java.

Programs will not compile if scripts are not indented. Indentation is something new programmers find difficult in other programming languages like Java, and C derived languages. Python strict rule means programmers get use to writing well-structured program code. This is a GCSE construct and is emphasised by exam boards for students to apply to their programs.

Variables are declared without types. In programming languages like Java and C derived languages, variable declaration is mandatory. As Python is loosely typed, beginner programmers do not have to worry about declaring their variables and may display a level of ease to users, which will encourage them to continuing programming. However, critics argue that variables without type declaration displays confusion. Computing teacher Mark Clarkson commented on CAS: "*one thing to be wary of is that Python hides the importance of declaring appropriate data types as it doesn't need variable declaration and can change variables data type on the fly"* (Clarkson, 2012). Mark is suggesting after initialising a variable to a data type, variables data type can be changed to represent another variable of a different type. This is likely to cause confusion. Figure 2.2.1 below (Python), shows how the type of a variable can be changed to refer to another type (in line 4). If the same were attempted in Java or C a compile error would occur. As a counter argument, other teachers disagree suggesting Pythons lack of declaration of data type is simpler than declaring data types: *"having to declare variables with specific data types in C isn't any more complex than having to remember and make a comment when initialising Python variables with the correct data type"* (Dreadman, 2013).

Java

```
1  public class printAnumber
2. {
3.        public static void main(String args[])
4.        {
6.           int a  = 6
7.           System.out.println(a);
8.        }
9. }
```

Python

```
1. a=6
2. print (a);
3. #Illustrate change in data type of a variable
4. a = "this is a String"
5. print(a);
-------------------------------------------------
Output:
this is a string
```

Figure 2.2.1: A Java program and Python program, which declares and prints a variable

In Python, less lines of program code increases productivity and programmers are able to produce small programs in a short space of time. (David Beazley et al., 2002). In figure 2.2.1 (above), it takes 9 lines of program code in Java to do the same thing in Python, which only takes 2 lines of program code (lines 1 and 2). Pythons syntax is easy to understand its syntax is in 'plain English' making program code easy to read, this is a GCSE construct. This makes Python a good choice to teach and engage students into programming. Simon Johnson a teacher of Computing echoes this: *"...It's natural syntax means it is easy to understand - you can't get any easier than "print 'hello world'"!"* (Johnson, 2012).

On the other hand, Python displays data in a complex manor in comparison to other programming languages such as Java. Understanding data structures is a requirement for GCSE students. Ian Hartley, a computing teacher commented on CAS:*"I dislike Python due to its data model".* (Hartley, 2014). In Java, one simple data structure is used to illustrate how

large quantities of data are kept these are arrays. Python on the other hand has a handful of data structures, which do different things. It has lists, tuples and dictionaries. Python lists are like arrays in Java where elements can be added, removed or indexed. Lists are mutable types meaning that elements in the list can be changed. Python tuples are immutable. Data contained in a tuple is heterogeneous, whereas elements in a list are homogenous. Tuples are used as keys in dictionaries. Dictionaries represent a more complex data structure like Javas Hash Maps. Java only has one main simple data structure to store quantities of data. Python has a combination of data structures, which do different things. Python's data complexity however, could be erased with simple understanding of the use of each data structure.

Despite some complexities involved in programming in Python such as lack of data type declaration and data structure handling, the benefits of using Python for this project far outweighs the drawbacks. This project addresses Python data complexity with a simpler solution. Python has many benefits which makes it simple to apply GCSE constructs, the force of indentation to write well structured code and the less lines of program code required are important factors which supports the API produced in this project. Python is a suitable language echoed by many teachers who teach Python, making it popular: *"Having taught Python, I can see why it is an easy choice for teachers! ….I find Java a tricky language in terms of syntax with my students, I think Python works better for GCSE"* - Simon Johnson computing teacher (Johnson, 2013).

2.2.1 Pygame

Pygame is a library for Python, Pygame is used to program computer games in Python. It contains thirty-seven modules (Pygame documentation, 2013), where users can import and use the functions provided in these modules. Pygame starts at the library stage in Pythons hierarchical structure (figure 2.2.1.1).

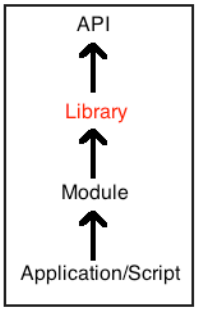


Figure 2.2.1.1: Pythons hierarchical structure

Pygame has many modules, which simplifies programming games for Python programmers. Program code is abstracted so that programmers can program games in a fewer lines of program code than that required in Python alone. For example, the Image module in Pygame allows the programmer to call a set of functions, which allow the user to load an image for example. By calling the load function in Pygame (figure 2.2.1.2), Program code is simplified to one line. This would lead to a lengthy algorithm if Python alone were used.

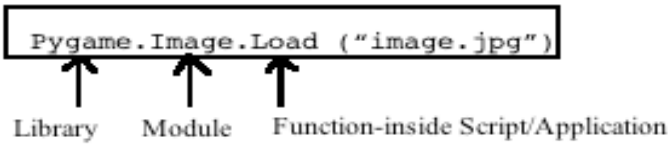


Figure 2.2.1.2: Loading an image in Pygame

The structure programmers must follow in Pygame is too complex for GCSE students. For programmers to program well structured programs, knowledge of Object Oriented Programming is required. This is not a GCSE construct. Figure 2.2.1.3(below) 'Catch A Fish' is an example of a well-structured game programmed in Pygame.

An example of a well-structured game programmed in Pygame

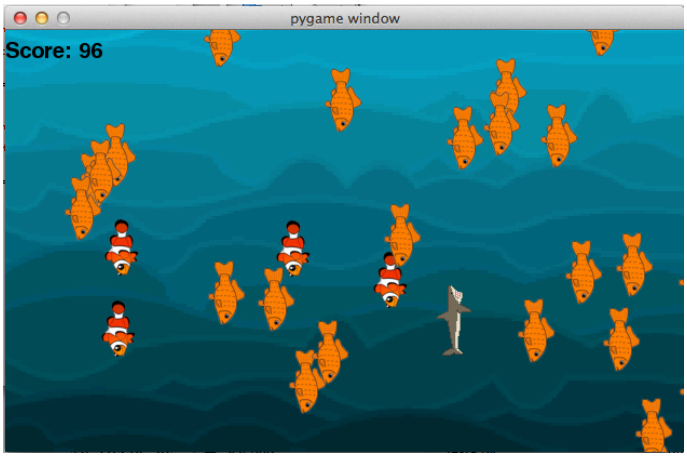


Figure 2.2.1.3: 'Catch A Fish' a simple Object Oriented game using Pygame

To program 'Catch A Fish' four classes are defined, this is illustrated in figure 2.2.1.4 below (See *Chapter Appendix*, A, part 10.1 for full program code).

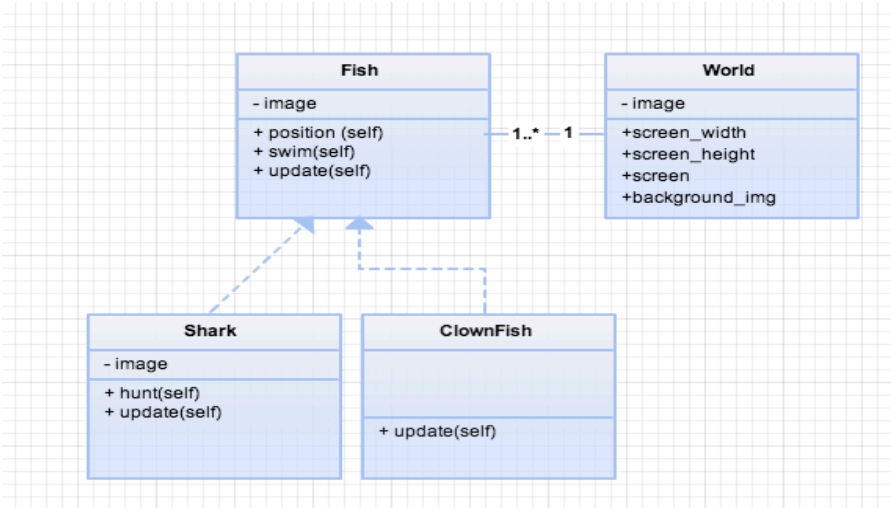


Figure 2.2.1.4: 'Catch A Fish' class diagram

The `Shark` and `ClownFish` class either inherits from the super class `fish`, or defines behaviour, which is unique to that `Fish`. For example, in the real world sharks hunt fishes this is illustrated in the game where the shark hunts the clownfish and the goldfish. This game uses Pygame Sprite groups. Sprite groups hold a collection of sprites. To use sprite groups, the `Sprite` class from Pygame is inherited. The `add()`, `remove()` and `update()` functions from the `Sprite` class are called to manipulate the collection of sprites: goldfishes and clown fish.

The program structure in this game ensures that program code is readable and reusable. The users code, which can be programmed in a separate client class, or within the python script,

only has 45 lines of program code. This program structure also ensures program code is decoupled.

Programmers can program games without any Object Oriented Programming. This comes at a cost, a tightly coupled program with several lines of program code. This contradicts Pythons simplicity goal, where programs may become unreadable. (This limitation is analysed in *Chapter 3 section 3.2*, with an example game that only uses GCSE constructs using Pygame).

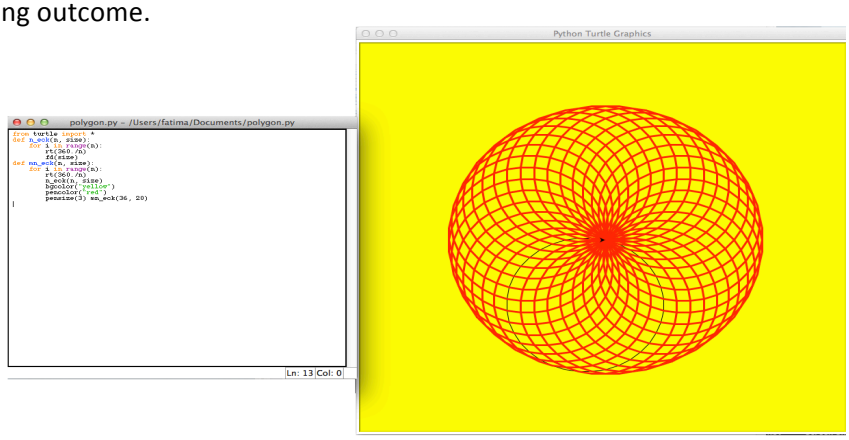
A teacher of Computing agrees with Pygame's complexity by commenting on CAS: *"I'm trying to avoid using it (Pygame) this year as I think it might be too complicated for some of my students who've done less than 6 months programming".* (Ames,2013).

The structure of Pygame simplifies programming computer games from using Python alone. The motivation behind programming in Pygame is to engage programmers. However, Pygame provides difficulty for beginner programmers. To write well structured program code, which is a requirement for GCSE students (OCR, 2014), students are required some knowledge of Object Oriented Programming. This is not a GCSE construct. Pygame alone is not a suitable resource for teachers that teach GCSE Computer students programming.

2.2.2 Turtle graphics for TK Python

Turtle Graphics is an interactive interface, which teaches programming through graphics. Turtle is a library for Python and is built in Python. Turtle graphics is used to illustrate programming graphics in a functional and an Object Oriented way. (Python documentation, 2013).

Turtle graphics functional implementation provides a level of ease for students. For example, to move a turtle a simple command `turle.forward(200)` is provided (figure 2.2.2.1). A function takes an argument *n* and returns a value. Where the distance of the Turtle to be moved is specified is the argument *n* (200). Students can be creative with their commands; figure 2.2.2.1 illustrates the use of these constructs, with an aesthetically pleasing outcome.

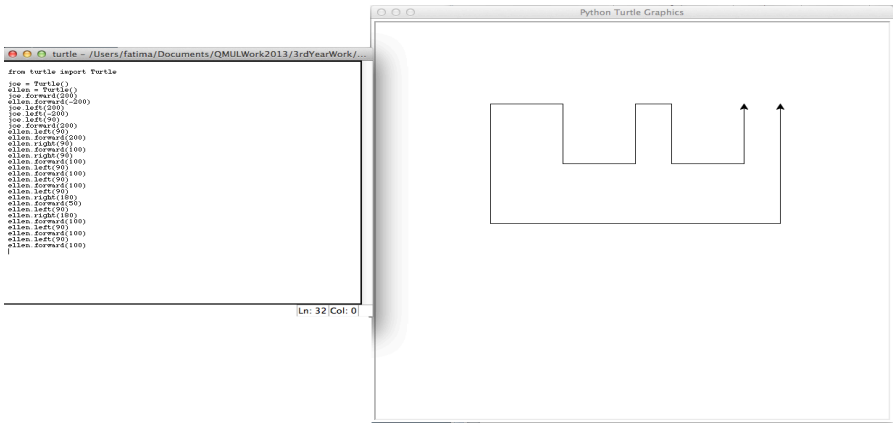


2.2.2. 1: A circle pattern using Turtle graphics (inspired by Wikipedia example)[2]

[2] Inspiration taken from Wikipedia (2013)
http://commons.wikimedia.org/wiki/File:Remi_turtlegrafik.png (accessed on 2/12/2013).

Programs can be made using Turtle graphics using Object Oriented Programming. This is shown in figure 2.2.2.2 (below). Two objects: Joe and Ellen are created. They both represent two different turtles, by instantiation each object to the turtle object. For example `joe = Turtle();` This turtle Joe will call on functions to only be performed on Joe. The use of Object Orientation however is entirely up to the programmer. This is not forced on the programmer. Programs can be created just using functional programming.

Programmers can easily apply GCSE constructs to a Turtle Graphics program. `for-loops` allows programmers to make interesting programs. Figure 2.2.2.1(above) was programmed using a `for-loop`. The loop ensured that the pattern was repeated until a full circle was created.



2.2.2. 2: An example of an Object Oriented program in Turtle Graphics

The turtle graphics website provides a good set of resources. It provides users with compressive documentation that provides sample code. Teachers can use turtle graphics as a teaching resource. Because of the functional programming required in programming Turtle graphics, teachers are able to produce worksheets that demonstrate the use of programming constructs that is required at GCSE level. Examples of resources that have been created from Turtle graphics can be found on the CAS website[3].

However, Turtle graphics is limited. After using Turtle for the first half an hour, programmers can get dis-engaged. The graphics created to begin with is fascinating, but the scope of what you can achieve is limited. You can only create graphics, which after a while can be disengaging.

Turtle Graphics is a great tool to teach GCSE students programming. Its natural syntax is suitable for GSCSE students. Teachers have a choice of teaching functional programming and Object Oriented Programming. The knowledge of Object Oriented Programming, programming is not forced on the programmer to write a structured program like Pygame. There are many great teaching resources for Turtle graphics. Despite all this, Turtle graphics is limited on what you can achieve. Programming Graphics alone may disengage users, there is no range to what you can achieve.

[3]Turtle graphics resources able on CAS, 2014
http://community.computingatschool.org.uk/resources/428 Accessed on 19/04/2014

2.3 Greenfoot

Greenfoot is an interactive environment designed to teach students programming. Games, graphics and stimulations can be built using Greenfoot **(**Kolling, 2010). Greenfoot runs in Java and is programmed in Java.

The Greenfoot API is used with an Integrated development Environment (IDE). When a Greenfoot project is started, a graphical user interface is presented. (Figure 2.3.1 below)

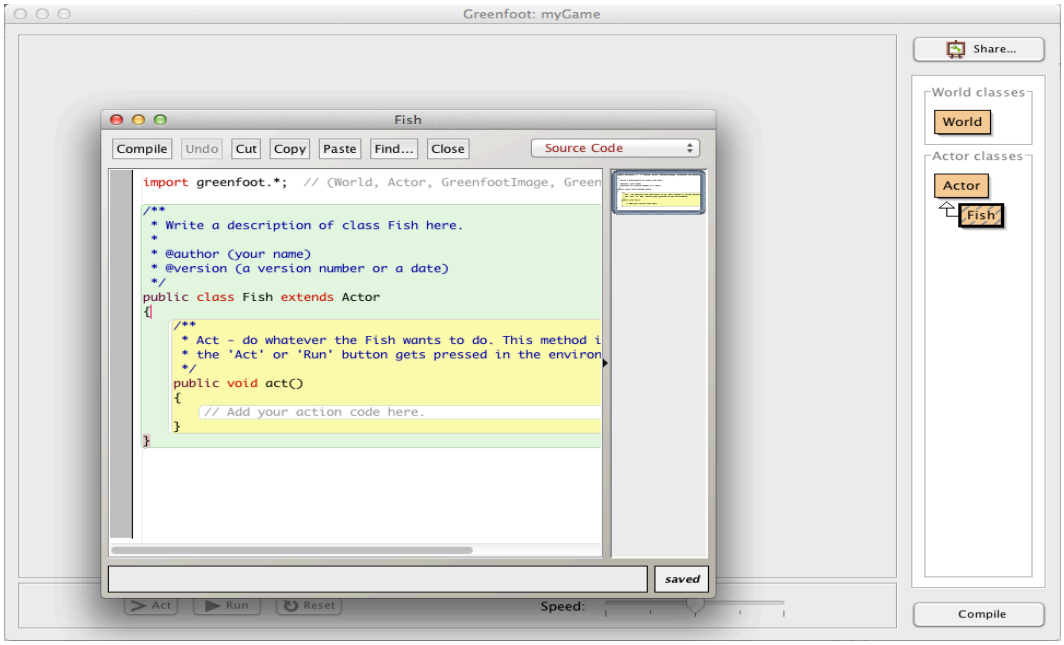


Figure 2.3.1: A generated program using the Greenfoot IDE

When a user creates a scenario a template is generated. The template generates the Object Oriented Programming program code. Users must define `Actors` and `Worlds` in their game. When the user creates an `Actor` or `World`, Greenfoot's `Actor/World` class is inherited. The act method in the `Actor` class is overridden (Figure 2.3.1) program code to make an actor behave is placed in this method. Greenfoot graphical visualisation makes it visually appealing for programmers and is likely to engage programmers to make games.

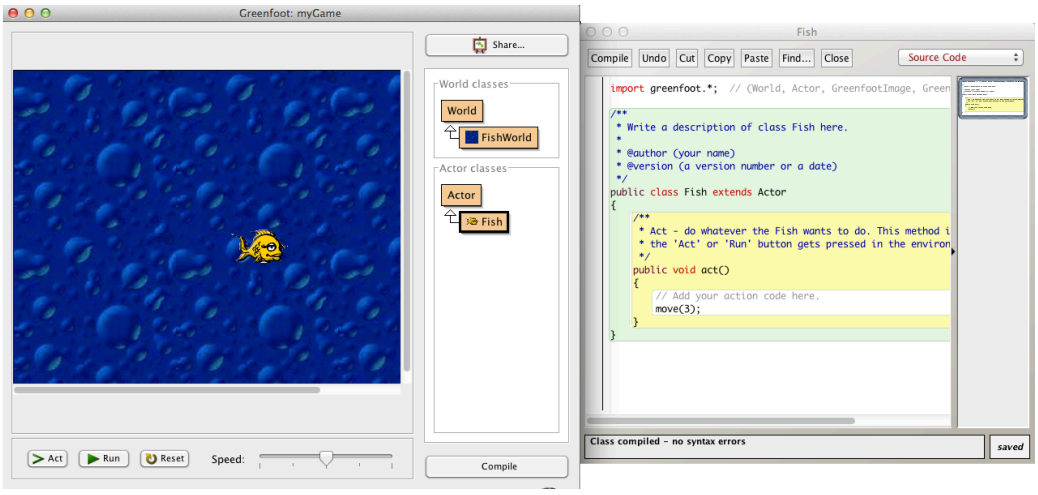


Figure 2.3.2: Moving an actor in Greenfoot

Very little programing is required to make an Actor behave. Once `Actor(s)` and a `World` are defined, its image is set. In Figure 2.3.2 (above), the `Actor's` image is set to a fish (One which Greenfoot provides) and the background is set to a water scene. One line of programming is required by the user to move an actor: `move(3);` The Fish class (in figure 2.3.2 above) inherits the Actor class, the move overrides the move method in the Actor class. This is a good design element of Greenfoot's API, although Object Oriented Programming is used, it not required by the user at the early stages of programming a game in Greenfoot.

A range of games can be created using the Greenfoot API unlike Turtle graphics; this diversifies Greenfoot and is likely to engage different types of users. Figure 2.3.3 is an example of a different type of game programmed using Greenfoot. This is a piano game, where the user can click different keys on the piano and a sound reflecting that key is played.

Figure 2.3.3: Piano game (example scenario from the Greenfoot website)[4]

Despite Greenfoot's attempt to ease the required knowledge of Object Oriented Programming to newcomers, the manipulation of objects is required by the user to program complete games (This is covered in detail in *Chapter 3, section 3.3 Analysis of Greenfoot*). Object Oriented programming is not a GCSE construct; many students are likely to find programming in Greenfoot without guidance difficult. Mark Clarkson, a teacher of Computing at Egglescliffe School shared on CAS, *"I considered it (Greenfoot), but the game-based focus makes it not quite right for the OCR GCSE tasks IMO..."*. Another down point of Greenfoot is that it is for Java this cannot be used by Python developers.

2.4 Summary

This chapter introduced GCSE requirements. It looked at the background of existing systems, which are used as teaching resources by teachers to help teach GCSE constructs to GCSE students. The libraries/API examined have limitations. To use Pygame, a compromise of using GCSE constructs must be made. Programmers must use Object Oriented Programming,

[4] My heart will goon . (2012). *piano-my project.* Available: http://www.greenfoot.org/scenarios/7263. Last accessed 19 April 2014.

which is not a GCSE construct but ensure well structured program code, which is a GCSE construct. This project solves this problem and aims to eliminate this compromise. Turtle is limited on what you can do and Greenfoot is for Java.

The middle point of figure 2.4.1 (below) is where Pygame Simplified is. It overcomes the limitations of Turtle graphics. This project is influenced by Greenfoot but eases knowledge of Object Oriented Programming required. *Chapter 3* examines in detail how GCSE constructs can be used with Greenfoot and Pygame. Goals for the Pygame Simplified API are drawn from this analysis.

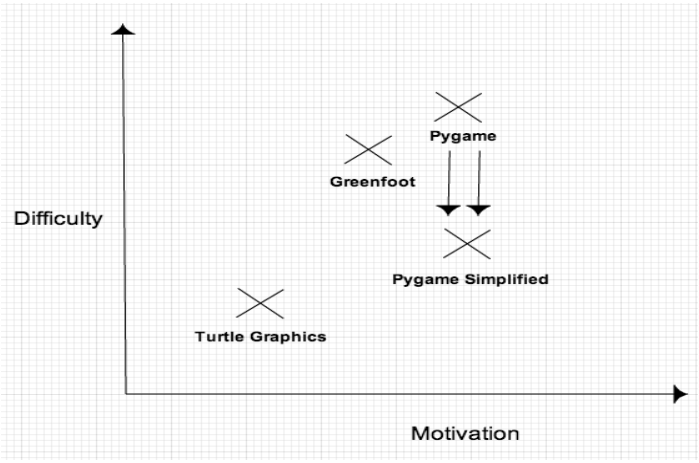


Figure 2.4. 1: Where this project lies amongst other teaching resources

Chapter 3 Analysis of sample game

As we have seen in the previous chapter, well structured programs can be written in Pygame. This chapter analyses how Pygame can be used with GCSE constructs (section 3.2), using a sample game introduced in section 3.1. A comparative analysis is made with Greenfoot in section 3.3, which has a number of techniques that simplifies programming for students and engages programmers. From analysing Pygame and Greenfoot, goals for the Pygame Simplified API are drawn in section 3.5.

3.1 Game scenario

This chapter analyses a balloon game. In this game, there are set of balloons and an arrow. The arrow can only move from left to right on the screen. When the user clicks their mouse, an arrow is fired. If this then hits a balloon, a balloon pop sound is played. There is a timer, which decrements from twenty seconds. Each time a balloon is popped the player score increments by one. The user must pop as many balloons as they can in the twenty seconds provided.

3.2 Analysis of Pygame

Figure 3.2.1 (below) is the Pygame example of the balloon game. References to program code in figure 3.2.2 for this balloon game are made throughout this section and section 3.2 with a comparative analysis made in section 3.3. This program is simplified and is programmed only using GCSE programming constructs. Trigonometry was used to rotate the arrows in the balloon game this has been omitted in the program (figure 3.2.2) to suit the programming abilities of GCSE students.

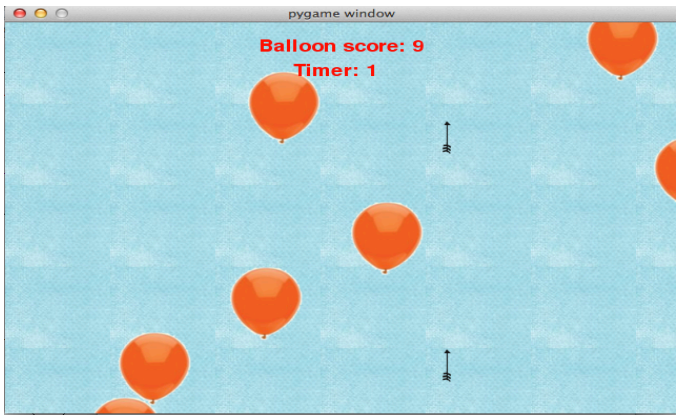


Figure 3.2. 1: The balloon game programmed in Pygame

In order to fully understand the complexities involved in programming a game in Pygame using only GCSE constructs a subset of questions are drawn from requirements outlined in controlled assessments provided by GCSE examination boards:

1. **Can GCSE students follow the Pygame's style of programming?**
2. **Does Pygame's library design suit the programming ability required by GCSE students?**
3. **How are GCSE constructs applied to programming a game in Pygame?**

```
    #part 1 - Import library
1   import pygame,sys
2   from pygame.locals import *
3   import math
4   import random
5
6   #part 2 - Initialise the game
7   pygame.init()
8   pygame.mixer.init()
9
10  #set varibales
11  track_arrows=[]
12  target_timer=50
13  target_timer1=0
14  targets_array=[[480,100]]
15  target_speed = 2
16  arrow_speed=10
17  distance_from_height= 64
18  stop_watch =20
19  score =0
20
21  window=pygame.display.set_mode((640, 480))
22  clock=pygame.time.Clock()
23  game_font = pygame.font.Font(None,30)
24
25  #score text loading
26  score_text=game_font.render('Balloon score: '+str(score), 2,[255,0,0])
27  box_size=score_text.get_rect()
28  scoreXpos=(640-box_size[2])/2
29  window.blit(score_text,[scoreXpos,20])
30
31  #timer text loading
32  stop_watch_text=game_font.render('Timer: '+str(stop_watch), 2,[255,0,0])
33  box_size=stop_watch_text.get_rect()
34  stop_watchXpos=(640-box_size[2])/2
35  window.blit(stop_watch_text,[stop_watchXpos,20])
36
37  #part 3 - Load image
38  sky = pygame.image.load("resources/images/background.png")
39  arrow = pygame.image.load("resources/images/arrow.png")
40  target_img1 = pygame.image.load("resources/images/balloon.png")
41
42  #part 3.1 - Load audio
43  pop_ballon_sound = pygame.mixer.Sound("resources/audio/balloon-pop.wav")
44  pop_ballon_sound.set_volume(0.05)
45
46
47  #part 4 – while user does not exit game
48  while True:
49
50      mousex, mousey = pygame.mouse.get_pos()
51      arrowx =mousex
52      arrowy = 400
53      target_timer-=1
54
55      for x in range(0, int(640/sky.get_width()+1)):
56          for y in range(0, int(480/sky.get_height()+1)):
57              window.blit(sky,(x*100,y*100))
58
59      #part 6.1 – draw an arrow and set the position of an arrow
60      window.blit(arrow, (mousex, 400))
61      arrow_position= ((arrowx,arrowy))
62
63      score_text=game_font.render('Balloon score: '+str(score), 1,[255,0,0])
64      window.blit(score_text,[scoreXpos,20])
65
66      #part 6.2 - Draw track_arrows
67      for bullet in track_arrows:
68          index=0
69          bullet[1]-=0
70          bullet[2]-=5
71
72          if bullet[1]<-distance_from_height or bullet[1]>640 or bullet[2]<-distance_from_height or
73  bullet[2]>480:
```

```
74                  track_arrows.pop(index)
75              index+=1
76              for arrows in track_arrows:
77                  arrow1 = pygame.image.load("resources/images/arrow.png")
78                  window.blit(arrow1, (arrows[1], arrows[2]))
79
80          #part 6.3 - Draw balloons
81          if target_timer==0:
82              targets_array.append([random.randint(50,610), 640])
83              #reset the timer
84              target_timer=100-(30*2)
85          index=0
86
87          for Target in targets_array:
88              if Target[0]<-distance_from_height:
89                  targets_array.pop(index)
90              Target[1]-=target_speed
91
92              #part 6.3.1 - hit balloons
93              target_rect=pygame.Rect(target_img1.get_rect())
94              target_rect.top=Target[1]
95              target_rect.left=Target[0]
96              if target_rect.left<distance_from_height:
97                  targets_array.pop(index)
98
99              #6.3.2 - Check for collisions
100             index1=0
101             for bullet in track_arrows:
102                 bullrect=pygame.Rect(arrow.get_rect())
103                 bullrect.left=bullet[1]
104                 bullrect.top=bullet[2]
105                 if target_rect.colliderect(bullrect):
106                     score+=1
107                     pop_ballon_sound.play()
108                     targets_array.pop(index)
109                     track_arrows.pop(index1)
110                 index1+=1
111
112
113             #part 6.3.3 - Next balloon to be drawn
114             index+=1
115         for Target in targets_array:
116             window.blit(target_img1, Target)
117
118         #decrement the clock watch
119         seconds = clock.tick()/1000.0
120         stop_watch-=seconds
121         display_stop_watch=math.trunc(stop_watch)
122         stop_watch_text=game_font.render('Timer: '+str(display_stop_watch), 1,[255,0,0])
123         window.blit(stop_watch_text,[stop_watchXpos,50])
124
125
126         if display_stop_watch==0:
127             print(" Your score is: ", score)
128             score=0
129             stop_watch =20
130
131         pygame.display.update()
132         for event in pygame.event.get():
133             # check if the event is the X button
134             if event.type==pygame.QUIT:
135                 # if it is quit the game
136                 pygame.quit()
137                 sys.exit(0)
138             if event.type==pygame.MOUSEBUTTONDOWN:
139                 #This movement value is stored in the track_arrows array.
140                 track_arrows.append([(mousey-arrow_position[1],mousex-
141     arrow_position[0]),arrow_position[0],arrow_position[1]
```

Figure 3.2. 2: Program code for the balloon game (by the author) - a demonstration of only using GCSE constructs

1. **Can GCSE students follow the Pygame's style of programming?**

The style of programing used in the balloon game ensures students are able to see exactly how the game is executed, this is a style enforced by Pygame. In figure 3.1.2(Part 4) line 49 states `'while True'` keep running the game (until the user quits the game). All program code, which changes at each run of the game are placed in this while loop (from lines 51 to 142). For example, each time the loop is executed the arrow and balloons must be drawn (at the positions they have moved to). Any changes to the arrow and balloons are updated. Program code above the while loop represents program code that only needs to be executed once. These are global variables. Figure 3.2.3 (below) is a diagram that shows how this game is structured. This style enforced by Pygame makes this game easy to follow. GCSE students are able to follow how a game is executed.

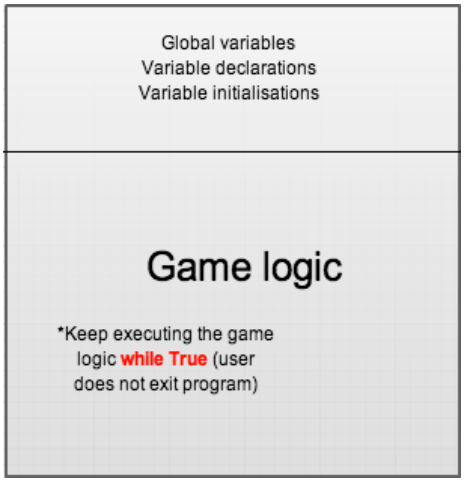


Figure 3.2.3: Program structure of the balloon pop game for Pygame

Students are required to write programs where structure makes the whole program easy to read. (Edexcel, 2013). This game is tightly coupled; all the program code is in one Python script, which makes program code difficult to read. Program code, which shows the behaviour of arrows and balloons, is not clear. By looking at figure 3.2.2, it's not easy to see where an arrow or a balloon is made. This style of programming makes it difficult for GCSE students to follow it all the way through.

1. **Does Pygame's library design suit the programming ability required by GCSE students?**

The Pygame library design is a loosely coupled with high cohesion. Modules do not rely on one another and methods in each module are related. Programmers can use any of these modules depending on their objective. Table 3 (below) summarises the modules and methods used from the Pygame library in the balloon game.

However, Pygame's library design is not suitable for functional programming, which is easily understood by GCSE students. In the balloon game, program code is tightly coupled. There is a lot of program code repetition. The line number column in table 3 illustrates this. For example, an image is drawn on to a surface using the `blit()` method; this is used seven times at different parts of the game. Program code logic requires an image is drawn at these different places. In part 6.3.3 (figure 3.2.2) a balloon is drawn after a collision. In part 6.2 arrows are drawn after the mouse position for its x position is set. This dependency in

program code logic makes the program code difficult to read and difficult for programmer to construct a game.

Table 3: Pygame modules used in the balloon game (Pygame, 2009)

Pygame Modules	Functions	Description	Line number
	`quit()`	Used to quit the game.	137
`font`	`font.Font()` `font.render()`	Creates a font object Used to draw font to the screen.	 24, 27,33,123
`time`	`time.Clock()` `clock.tick()`	Creates a time object Used to keep track of time.	23 120
`Mouse`	`mouse.get_pos()`	Gets the position of the mouse cursor.	51
`sys`	`exit()`	Used to exit the Pygame window	138

`Rect`	`Rect.colliderect()`
	
	`Rect()`
		Checks if two rectangles collide.
		
		Stores the co-ordinates for a rectangle.
			94, 103
			
			94, 103

`Image`	`get_rect()`
	
	`image.load()`
		Gets the rectangle of a surface.
		
		Loads an image to the window surface.
			28,34,94,103
			39,40,41,178

`Display`	`display.update()`
	
	`display.set_mode()`
		Updates the Pygame screen.
		
		Sets the display mode –width and height of a Pygame window.
			132
			
			22

`Surface`	`window.blit()`	Draws an image to another image.	30,36,58,61,79 ,117,124

`Event`	`event.get()`
	
	`event.type()`
		Gets the event from the Pygame class.
		Gets the type of event.
			138
			
			135,139

`Mixer`	`mixer.Sound()`
	`Sound.play()`
	`Sound.set_volume()`
		Creates a sound object
		Plays a sound
		Sets the volume for a sound
			44
			108
			45

2. How are GCSE constructs applied to programming a game in Pygame?

Programmers are encouraged to use GCSE programming constructs. To program a game, a problem is set to the user. Programmers ask: *'how should the game be developed?'* In the case of the balloon game, this question leads to further questions:

1. How do I create multiple actors such as arrows?
2. How do I make actors move?
3. How do I set the position of actors?

These set of questions encourage the programmer to form an algorithm, which will answer these questions and solve the programmer's problem. The questions a programmer asks encourages them to think of which programming constructs they should use. In balloon game, the first question is answered with `lists`. The balloon game uses lists to store

balloons (line 15) and arrows (line 12). Applying lists is a GCSE construct; other GCSE programming constructs are used. If a programmer requires an element in a list a `for-loop` is required. Elements in a list are accessed through a `for-loop`. For example, to remove a balloon, which is hit by an arrow, the balloons list is iterated through (line 88-115). Other programming constructs are also used, such as `if statements`. If a balloon goes of screen (line 89), then the balloon is removed from the list holding balloons (line 90). Solving problems and constructing algorithms is a core requirement set out by all three-examination boards, this game programed in Pygame makes good use of GCSE constructs.

Despite the use of GCSE constructs, data manipulation in Python is complex as examined in *chapter 2 sections 2.1.* This game uses `tuples` and `lists`. A tuple is used for the position of an arrow (line 62). The arrow has an x and y. The co-ordinate x is set to the mouse's x position and the y position is set to 400, which is just above the bottom of the screen. This means an arrow can move across the screen on the x-*axis* at the bottom of the screen. A tuple is an immutable type meaning the x and y of the arrow cannot be assigned to a new position throughout this game, these positions are held. This is to prevent unconscious aliasing in the game. If a `list` or a variable was used for this, with out knowing the users can alias the x and y co-ordinates of the arrow to new positions. If attempted on `tuple` an error is thrown. This data representation is complex and will cause confusion for GCSE students. Pygame does have an alternative called sprite groups (briefly described in *Chapter 2*). However, to use a sprite group, Object Oriented Programming is required. This alternative is not suitable for GCSE students.

3.3 Analysis of Greenfoot

This section gives a comparative analysis to the issues and questions raised in *section 3.2 Analysis of Pygame.* The same game is programmed in Greenfoot and the same set of questions in the previous section is asked. Figure 3.3.1 (below) is the Greenfoot example of the balloon game. (Full program code can be found in the *Chapter 10 Appendices*, A part 10.2).

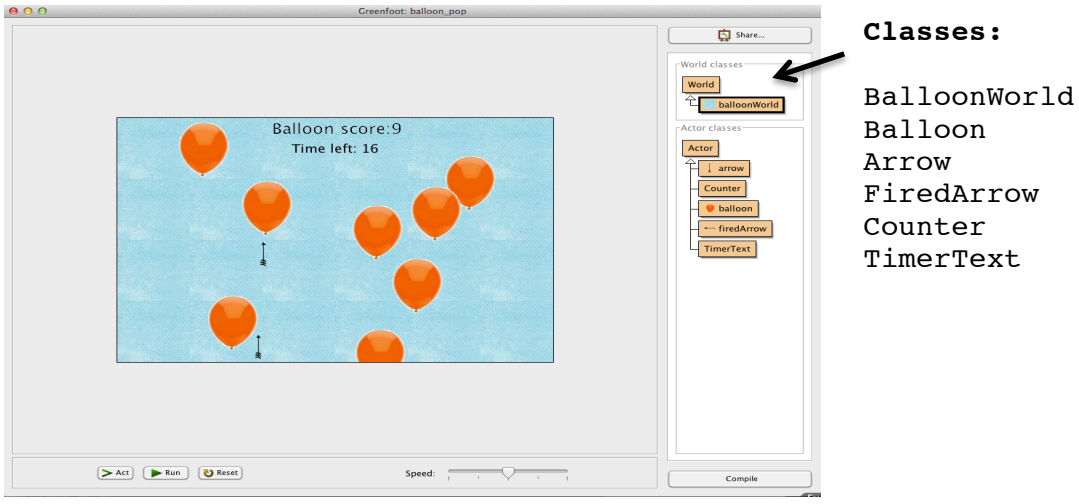


Figure 3.3.1: The balloon pop game programed in Greenfoot[5]

[5] *Greenfoot provides the counter class, this class contains a set a methods to ensure the user can add a score to their game. The counter class is used in this game.*

1. **Can GCSE students follow the Greenfoots style of programming?**

In Greenfoot, the user is required to create the classes for each of the Actors and World in a game. These classes extend Greenfoot's Actor and World class. Figure 3.3.1 (above) shows the classes used in the balloon game. Having different classes for each Actor and World makes the program code in figure 3.3.1 readable. Behaviours for each Actor are separated into different classes. Figure 3.3.2 is the `Firedarrow` class in the balloon game. The `Firedarrow` class extends Greenfoot's Actor class. In this class, the act method overrides the Actor classes act method this method implements an actor's action. Unlike Pygame, Greenfoot provides a template to programmers so they do not have to implement the Object Oriented Programming themselves. In Pygame, programmers are required to construct classes and methods themselves. Greenfoot programming style means students are able to form the logic required in a game in a loosely coupled format.

```
1  public class FiredArrow extends Actor
2  {
3      //private int direction;
4      public void act()
5      {
6          // Add your action code here.
7          move(5);
8          of_screen();
9          if (getWorld() != null)
10             collide_with_balloon();
11     }
12     public firedArrow(int dir)
13     {
14         dir = -90;
15         setRotation(dir);
16     }
17
18     public void collide_with_balloon()
19     {
20         Actor balloon=getOneIntersectingObject(balloon.class);
21         if(balloon !=null)
22         {
23             ((balloon)balloon).pop();
24             getWorld().removeObject(balloon);
25             getWorld().removeObject(this);
26             Greenfoot.playSound("balloon-pop.wav");
27         }
28
29     }
30
31     public void of_screen()
32     {
33         if(getX()>=getWorld().getWidth()-1)
34             getWorld().removeObject(this);
35         else if (getX() <=1)
36             getWorld().removeObject(this);
37         else if (getY() >=getWorld().getHeight()-1)
38             getWorld().removeObject(this);
39         else if(getY() <=1)
40             getWorld().removeObject(this);
41     }
42 }
```

Figure 3.3. 2: Program code for the firedArrow class

2. **Does Greenfoots API design suit the programming ability required by GCSE students?**

Greenfoot's API design simplifies programming for users. Program logic to make actors behave is provided for users. Because of Greenfoots Object Oriented nature methods are overridden from the `Actor` class. Functions such as `move()` and `setRotation()` can

be called. To move a fired arrow, the `move()` method from Greenfoot's Actors class is used (figure 3.3.2, line 7), an integer argument `distance` is passed. This represents the distance the `Actor` should move. This design element encourages GCSE students to understand GCSE programming constructs. In Pygame, the user is required to create their own program code to make Actors move (a fired arrow is moved in Lines 70 and 71 in the Pygame version of the balloon game Figure 3.2.2). This increased the complexity in programming the balloon game. Due to the closely coupled programming in the Pygame version of the balloon game, it is not clear to the user where an `Actor` is moved. This makes the program hard to follow. In the Greenfoot version (figure 3.3.2) it is clear where the actor is moved.

3. **How are GCSE constructs applied to programming a game in Greenfoot?**

Object Oriented Programming is required in Greenfoot's balloon game this adds some complexity to programming. Lines 17 to 25 checks if a fired arrow collides with a balloon. In line 19, a balloon object is created to check if a `firedArrow` collides with a balloon using the `getOneIntersectingObject()` method from Greenfoot's `Actor` class. The balloon `Actor` created is passed as an argument to the method. This line of programming is difficult to construct without knowledge of classes and objects. Line 22 adds another element of complexity. Type casting is used to increment the counter (timer) if there is a collision. Type casting is a heavy topic in Object Oriented Programming, this not a GCSE construct, unless guidance is provided (by teachers) GCSE students will find this difficult.

In comparison to Greenfooot, Pygame's solution to checking for collisions uses GCSE constructs. In the Pygame version of the balloon pop game (Line 102, line 111- Figure 3.2.2). A `for loop` is used to access all the elements in an arrows list. The rectangle of an arrows image (used to check for collisions) is checked against the target rect (balloons rectangle) using the `colliderect()` collision function. If there is a collision, the score variable is incremented and a balloon pop sound is played.

3.4 Goals for Pygame Simplified

Greenfoot indicates a number of techniques, which inspires the design of the Pygame Simplified API. An API that encapsulates Pygame's useful modules is designed with the following set of goals:

- Readable program code with minimal Object Oriented Programming required by programmers to program a simple game.
- Template that provides programmers program code that will help get them started.
- Methods which contain game logic to set properties of `Actors` and `Worlds` are provided to the user in a simplified interface.
- GCSE constructs are simple to apply to a game.
- A simple structure to represent actors that minimise the need for Pythons data structures.

3.5 Summary

This chapter gave a comparative analysis of Greenfoot and Pygame. From analysis Pygame its clear that GCSE constructs can be used with a compromise to writing a closely coupled program. Greenfoot showed the opposite. Goals from this analysis are drawn which aims to provide a solution to the problems identified. These goals are expanded on in the next chapter (*Chapter 4 Requirements*).

Chapter 4 Requirements

This chapter introduces the requirements for Pygame Simplified. Section 4.1 covers the required resources for teachers. Section 4.2 looks at the functional requirements for Pygame Simplified version I. Section 4.3 assesses the functional requirements for Pygame Simplified version II. Section 4.4 details the non-requirements for the both versions of Pygame Simplified. Section 4.5 examines the functional and non-functional requirements for the Pygame Simplified (GUI) code generator.

4.1 Resources for teachers

The Pygame simplified API contains resources to help teachers and students use Pygame Simplified version I and Pygame Simplified version II. These resources are provided on a public website.

Installation: A comprehensive guide with illustrations on how to install Pygame Simplified, Pygame and Python is provided. Installation instructions for the different version of Python and Pygame for Windows, Mac and Linux are also provided.

Documentation: All the modules and methods in Pygame Simplified API are documented. A comprehensive explanation of how methods work is given. Versions of Pygame Simplified are also documented; explanation of different features in Pygame Simplified API is given.

Tutorial: A tutorial for Pygame Simplified for both versions of Pygame Simplified is given. Instructions on how non-trivial methods are used (with sample program code and a template) are provided.

Worksheets: Pygame Simplified provides three worksheets. Two of the worksheets are for Pygame Simplified version I API (but can be used with the version II). The third worksheet can only be used with Pygame Simplified version II. The worksheets give a step-by-step guide on how to build a simple game using the Pygame Simplified API. Programmers are given a chance to enhance this game with exercises provided at the end of the worksheet.

Example games: A range of example games using the Pygame Simplified API is available. Games, which show the variety that can be achieved with Pygame Simplified, are provided.

Feedback: Users can give feedback on Pygame Simplified. This allows improvements of Pygame Simplified to be carried out. Users of Pygame Simplified can give feedback through a questionnaire or a blog. Users are also given the opportunity to report problems with Pygame Simplified or bugs through the Pygame Simplified email (available on the Pygame Simplified website).

4.2 Functional requirements - Pygame Simplified version I

This section looks at the functional requirements Pygame Simplified version I. They represent the core functionality needed in the Pygame Simplified API.

Actor: An actor object, which has a number of states and behaviours.

World: An actor lives in a world. This world contains a number of states.

Move: All Actors in a game must be able to move. They must be able to move by a specific distance. Actors must also be able to move from a point a to b. Point a and b represent co-ordinates x and y.

Speed: The speed of an actor can be determined and altered. Speed is altered when an actor moves from x to y.

Rotate: All actors can be rotated. There are three types of rotation: `setrotation`, `turntowards` and `turn`. `setrotation` allows the actor to rotate at any angle between 0 and 360°. `turntowards` turns an actor towards an x and y position. `turn` allows the actor to turn by an amount.

Collision: If an actor collides with another actor, there is a collision. A method to illustrate a collision is available.

Location: An actor starts from x and y position, which is the location of an actor. Each actor has a location, which altered in the game.

Image: Each actor has an image. A world (an actor lives in) also has an image. The user sets an image of an actor and world.

Add actors: This feature allows actors to be placed in a word, which they live in. Adding an actor represents the birth of that actor, and the first time it appears in a scenario.

Remove actors: An actor can be removed from a world. If for example, an action such as a collision occurs, the actor that has been collided with may be removed from the world.

Exit: The user can exit a game by clicking an exit button on a graphical user interface.

Text: Text can be added to a world. Text may be drawn represent a pre-defined element of a game. For example, for an actors score to be visible for a user, the score can be set as text to the screen.

Keyboard interaction: This allows the programmer to check if a key on a keyboard has been pressed or lifted an action, which changes an actor's behaviour can be defined. For example, if a user presses the right arrow an actor may move by a specific distance.

Mouse interaction: This allows the programmer to check if a mouse has been clicked. For example, if a mouse has been clicked the programmer may define an action that changes an actor's behaviour.

4.3 Functional requirements - Pygame Simplified version II

Pygame Simplified version II has all the features of the Pygame Simplified version I with the following enhancements:

Create multiple actors: Multiple actors can be created through a method call. Actors could be apart of a group of actors, which share the same characteristics such as image, or a number of actors can be generated to a random position on the screen. Multiple actors have the following features:

- **Add**: add actors to a group
- **Remove**: remove actors from a group
- **Count**: check how many actors are inside a group

Collisions: this feature checks if an actors has collided with multiple actors.

Sound: A programmer can add sound to their game. The programmer defines the sound they would like to play, they can set the volume of a sound, play the sound, pause the sound and stop the sound.

Timer: A timer can be set. The number the timer is set is defined by the user. Programmers can choose to increment or decrement their timer.

4.4 Non- functional requirements Pygame Simplified API

For the Pygame Simplified API to be usable, the following non-functional requirements are drawn for Pygame Simplified version I and Pygame Simplified version II.

- **Documentation**: Documentation that allows the user to use the Pygame Simplified. This includes resources that help the user to use the Pygame Simplified API, such as worksheets, example games and tutorials.
- **Portability**: Pygame Simplified is tested against Python 2 and 3. Pygame Simplified is also tested on Windows, Mac and Linux platforms.

4.5 Pygame Simplified (GUI) program code generator

4.5.1 Functional requirements

The Pygame Simplified code generator simplifies using the Pygame Simplified API for GCSE students. The Pygame Simplified program code generator uses Pygame Simplified version II. The Pygame Simplified code generator has the following requirements:

Set the world: The programmer can set the name of the world for their game. They can also set the width and height of the world and set the image of their world. A programmer can only define one world for each game and set the image of a world.

Add actors: The programmer can set the number of actors they would like to add to their scenario. They can set the name of each actor, the actor's image and its location.

Generate program code: Once the user has defined the properties for their world and actors, they can generate the program code for this.

Simple Integrated development Environment (IDE): The programmers generated program code is placed in a text editor; the programmer can edit the program code in the text editor and their program code. The simple IDE has the following features and requirements:

- **Run program code:** The programmer can run their program code in the text editor.
- **Save/Open program code:** The user can save their program code they can retrieve this program code at a later time.

- **Generate Pygame Simplified template code:** The user can regenerate their program code.

4.5.2 Non- functional requirements

- **Installation**: Simple installation instructions.
- **Usability**: The GUI program code generator can be used with minimal errors.

4.6 Summary

This chapter examined the requirements for the Pygame Simplified API and program code generator. The functional and non-functional requirements for Pygame Simplified version I and version II were examined. Functional and non-functional requirements for the Pygame Simplified (GUI) code generator were also assessed. Chapter 5 looks at the implementation of these requirements.

Chapter 5 Design and Implementation

This Chapter covers the design and implementation of the Pygame Simplified API and program code generator. Section 5.1 examines the design and implementation of Pygame Simplified version I. Section 5.2 covers the design and implementation of Pygame Simplified version II. Section 5.3 looks at the design and implementation of the Pygame Simplified (GUI) program code generator.

5.1 Development Environment

Pygame Simplified is developed in Python 3 and Pygame 3. The program code generator is developed in PyQt4 and QScintilla. PyQt4 a GUI tool kit for Python 3. QScintilla is a library that comes with PyQt4, it is used to build the simple IDE for the program code generator.

5.2 Pygame Simplified version I

5.2.1 Design

The requirements detailed in Chapter 4 are encapsulated into suitable Python modules for importation and use in a simple client class. Each module contains a class, which has the behaviours and functionalities. For example, the `Actor` class has functions, which define the behaviours for an `Actor` such as movement and rotation. The modules are placed into a `PygameSimplified` package (figure 5.2.1). The Pygame Simplified API contains the following four classes:

- **Actor:** The main character of a game.
- **World:** Where the actor lives.
- **Event:** Control actors through mouse and keyboard events. This class uses methods from Thedeus Brugess '*wrapper for the Pygame input module' (Brugess,2009)* provided on the Pygame website, methods used from this wrapper are documented in *section 5.2.2 Implementation*.
- **Text:** A string of text can be set and drawn to the world.

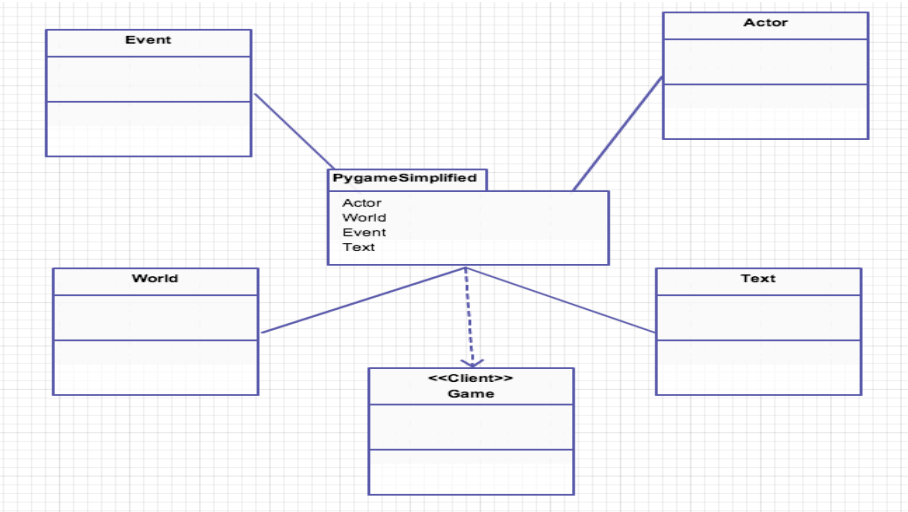


Figure 5.2. 1: A class diagram of the structure for the Pygame Simplified API

5.2.2 Implementation

Table 5.2.1.2 below gives a detailed description of how each method in each class is implemented and abstracted from the Pygame library.

Table 4: implementation of methods in Pygame Simplified version I API

Method	Description
Actor class The `Actor` class contains the behaviours and functionalities for an actor. An actor is created by creating an instance of the `Actor` class: `actorname=Actor()`. Functions (explained in detailed below) can be called on this instance to make actors behave.	
`update()`	The update function allows an actor's position *dx, dy* and rotation (which are defined as private methods) to be updated. Each time the game is run, the properties of an individual actor are updated.
`set_location(x,y)`	An actor's location can be set through this function. The initial x and y position of an actor is initialised in the constructor as global variables, this function changes these global variables to the x and y positions passed as arguments.
`set_image(filename)`	This function takes a string filename as an argument; the string passed must have the complete class path to where the image is saved. This function is implemented as a wrapper of the `Pygame.image.load(filename)` function Pygame provides.
`move(distance)`	This function allows an actor to be moved. It takes an integer argument `distance`. This represents the distance an actor should move from its current location. This function is implemented using trigonometry. *dx* and *dy* are the change in distance on the x and y co-ordinates. The user passes their distance value, the distance an actor moves on the x and y co-ordinates are calculated. On the *x-axis* the distance is calculated using *cos*. On the *y-axis* the distance using *sin* is calculated. Both *cos* and *sin* take an argument theta which is the angle. The angle is the radiant measured in radians, which is a standard measurement for angles. The angle in radian is calculated from the angle in degrees using the following formula: one $radian = (\frac{\pi}{180})$
`move_by(x,y)`	This function takes two arguments: an x and y position, where an actor can move by. The move by allows an actor to move from one point to another, and not just a constant distance, which is illustrated in the move function. The x and y co-ordinates is passed as arguments and added to the global x and y variables of the actor. This x and y position is multiplied by the speed set for the actor.
`set_speed(speed)`	This function takes an integer argument speed. The speed represents how fast the actor can move along the x and y co-ordinates. The speed (global variable) of an actor is altered in the `move_by()` function.

set_rotation(rotation)	This function allows the actors rotation to be set. It takes an argument rotation, which represents the amount degrees the actor is rotated by. Firstly, the rotation argument is checked to see if this is greater than or equal to 360 if so, the rotation is the reminder of the rotation and 360, this is found by taking the modulus of that number. This will never give a number greater than 360. If the rotation is less than 0 (a negative number), 360 is added to the remainder of this negative number to get a number between 0 and 360.
turn(turn_by)	This function takes an integer turn_by as an argument. The integer turn_by represents the amount an actor turns. The amount that is passed by the user is added to the global variable rotation. Rotation is updated in the private function `__rotation()` which is updated the `update()` function. The `set_rotation()` function is called to set the angle, which the character turns by.
turn_towards(x,y)	This function takes integers x and y as arguments, this is the positions an actor faces at an angle. The angle is calculated using trigonometry. The radiant, which is an arc in a circle, is taken using Pythons `math.atan(`y,x`)` function. The atan2 function returns a radiant, the vector in the plane from the origin to point `(`y,x`).` The arguments in this function are the passed x angle minus the current x, and the y angle minus the current y. This gives the new positions the actor is facing on the x and y co-ordinates. The radiant is converted to degrees (using formula 5.1 below), by multiplying the current radiant by $\frac{180}{\pi}$, which is equivalent to one radian. The rotation of the actor, based on the new x and y is determined by calling the `set_rotation(rotation)` function, which takes the degrees as an argument. This sets the angle the actor is facing. $degrees = radiant \times (\frac{180}{\pi})$ Formula 5.1
intersects(Actor)	This is a collision function; this function takes an argument Actor. This function returns a Boolean true or false. Pygame's `rect.colliderect()` method is used, this checks if the actor calling this functions rectangle collides with the actor passed in the functions rectangle. If so, a Boolean collision is set to true, otherwise the Boolean collision is false, this function returns this Boolean collision.
collide_point(pos)	This is also a collision function. This function checks if an actor has collided with an x and y position of a mouse. It takes integer argument pos, which is x and y position that is tested for collision. It returns a Boolean true or false. This function uses Pygame's `rect.collidepoint()` method. If there is a collision, this method returns true, otherwise it returns false.

World class

A `world` is where an actor lives. A world is created by creating an instance of the `world` class: `worldname=World(width,height).` The arguments `width` and `height` are integers; functions (explained in detail below) are called on this instance.

update()	This function updates the world. It does not take any arguments or return anything. This function is a wrapper of Pygame's `Pygame.display.update()` method.
set_caption(caption)	This function sets the caption, which is the title of a Pygame Simplified window. This function takes a string argument caption. This function is a wrapper of Pygame's `Pygame.display.set_caption(caption)` method.
set_tile_background(image)	This function sets the background image of the world, which is tiled. It takes an image argument, which is an image that is tiled. This function is implemented as a wrapper of the `Pygame.image.load(filename)` method Pygame provides.
set_background(image)	This function sets the background image of the world. This function does not tile the image, but takes the whole image that is set as an argument. This function a wrapper of the `Pygame.image.load(filename)` method Pygame provides.
set_background_fill_colour((red,blue,green))	This function sets the background colour. It takes three integer arguments: red, blue and green. This function is implemented by setting a global variable `colour` to the colour that's passed in as arguments. The colour variable is initialised in the constructor. The colour set becomes the background colour.
pack()	This function sets the window size to the size of the background image. This function uses Pygame's `Pygame.display.set_mode()` method. This takes the size of the background image. The current window size is replaced by the size of the image.
draw_tile_background(imagex, imagey)	This function draws the tiled image. It takes the size of the *imagex* and *imagey*. The image set is tiled next to each other using a nested for loop. One for the *imagex* and *imagey*. All the elements in the for loops are drawn to the screen at are multiplied by the values passed as arguments.
draw_background(x,y)	This function draws the background image set by the user. It takes arguments x and y. This represents the position the image should be drawn to the world. This method uses Pygame's window.blit() method. This draws the passed image and the x and y position of the image to the screen.
draw_actor(Actor,x,y)	This method draws an actor to the screen. This method takes arguments Actor, x and y. The actor argument is the actor that will be drawn to the screen. The x and y arguments represents the position the actor is to be drawn. This method uses Pygame's window.blit() method. This draws the actors image and the x and y position of the image to the screen.
remove_actor(Actor)	This function removes an actor from the screen. This function takes an argument Actor the actor is removed by setting the location of the actors off the screen.
exit_game()	This function allows the user to exit the world when the X button on the Pygame Simplified window is clicked. This function uses Pygame's events module to exit the game.

Text Class

The `Text` class is where all the functions for displaying text on `World` are. An instance of the text object must be created in order to use functions in this class to be use, for example: `textname = Text(World)`, the functions (detailed below) can be called on this instance.

`set_text(text)`	This function allows you to set the text of text label. It takes a string text label argument. This method uses Pygame render method provided in the font module. This method takes the text argument a Boolean (true is the font is to be rendered) and the colour of the text. The colour of the text is set in a separate method. It sets the properties of the text. If the user does not set the colour, a default colour black is used.
`set_font(foname,fontsize)`	This method sets the font name and size of a text. It takes the arguments `fontname` and `fontsize`. Font name is the name of the font that should be used for example, 'Calibri'. The font size is the size of the text that is used. This method uses the `SysFont()` method provided by Pygame which takes the font name and font size.
`set_colour((red,blue,green))`	This function sets the colour of a font. It takes integer arguments red, blue and green. This makes the colour of the text. This method sets the global variable `colour_text` to the values passed as arguments. This colour is set to the text in the `set_text()` method.
`draw_text(x,y)`	This function draws the text to the screen. It takes two integer arguments x and y. These arguments are the position where the text should be drawn. This method uses Pygame's window.blit() method. This draws the passed image and the x and y position of the image to the screen.

Event Class

The `Event` class contains all the functions (detailed below) for controlling actors. To use the functions in the `Event` class (detailed below) functions are called on the `Event` class name, for example: Event.mouse_down("button1").

`update()`	This function updates events in the events class. An event changes state each time the while loop is run, this function ensures these states are updated. This method updates the mouse events and key events; it also updates the position of the mouse cursor.
`is_key(keyname)`	This function checks if a key has been pressed. It takes an argument key name. This can be a string key that is: "left","right","up","down" which is a wrapper of the left, right, up and down keys. This function also takes the key names (that represent keys on a keyboard) provided by Pygame. This function uses Thedeus Brugess wrapper, except where the user can input a strings as a key name.
`key_up(keyname)`	This function checks if a key has been released. This function takes an argument key name, which represents the key that has been released. This function uses Thedeus Brugess wrapper.
`mouse_down(button)`	This function checks if a mouse has been held down. It checks against three types of mouse buttons: 'button1', 'button2' and 'button3'. Either of these mouse buttons can be passed as a string argument, for example: 'button1'. This method uses Thedeus Brugess wrapper.

contains(key)	This method checks if the event class has a key. It takes an argument key. This method uses a dictionary, which contains all the keys in the Events wrapper class. This method can be used to see if user wants to quit the game (with the `quit_game()`) method. This method uses Thedeus Brugess wrapper.
quit_game()	This function quits the game if called. It calls the `Pygame.quit()` method.
set_CursorPosOnImage(sizex,sizey)	This function set the image size, this is so when a cursor hovers over image, the size of that image can be noted so the cursor is on the mid point of an image (this is implemented in the `set_MouseCursorPos()`. It takes the arguments *sizex* and *sizey*. This is the size of the image.
set_MouseCursorPos(x,y)	This function sets the cursor position on a mouse. It takes an integer argument x and y. This sets the image size passed in the `setCursorPosOnImage()` so that the mouse cursor is on the image, rather than away from the image.

5.3 Pygame Simplified version II

Pygame Simplified version II is an enhancement of Pygame Simplified version I. Multiple actors can be created and generated to the screen with no extra Object Oriented Programming required. Pygame Simplified version II also has two extra classes: `Sound` and `Timer`.

5.3.1 Design

A façade design pattern and the singleton design pattern are used to improve the usability of Pygame Simplified. The façade pattern simplifies the importations required by a user to one line. The Singleton ensures users are aware that creating more than one world is not allowed. Pygame Simplified version II the following additions to Pygame Simplified version I:

- **Actor:** Multiple actors can be generated or created in an actors group.
- **Sound:** A sound can be played, paused and stopped. The volume can also be set.
- **Timer:** A timer can set this is incremented or decremented.

Façade design pattern

In Pygame Simplified version I, the classes in the Pygame Simplified API are imported separately. Up to four lines of imports is needed. The addition of the two classes in Pygame Simplified Version II means six import lines would be needed. A façade design pattern is used reduce the number of importations (figure 5.3.1.1, below) to one line. A façade provides a simplified interface to program code that is not visible to a programmer, in Pygame Simplified version II all the imported modules from the Pygame Simplified API are inside one façade class `PygameSimplified`, the user is not aware of all these imports. The classes in the Pygame Simplified API including the façade class are in a package (PGS) away from the programmers view. The façade class (`PygameSimplified`) is called in the client class the programmer can use either of the following two import mechanisms:

```
1.from  PGS.PygameSimplified import * #imports all classes
2.from  PGS.PygameSimplified  import  Actor,World,Text,Sound  #imports
individual classes
```

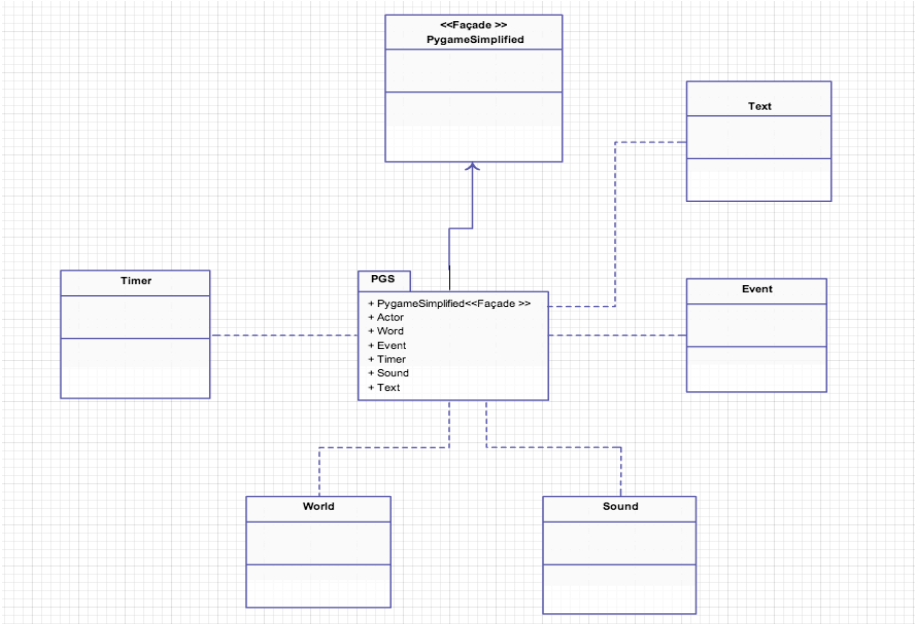


Figure 5.3.1.1: A class diagram of the facade design used in Pygame Simplified

Singleton design pattern

In Pygame Simplified version I, users create more than one world in a single game. Creating more than one instance of a world in a game is illogical as actors can only be at one place at a given time. The singleton pattern ensures only one instance of the World object can be created. If the programmer tries to create more than one instance of a world in a single client class, an error is thrown. Figure 5.2.1.2 below is program code for the singleton pattern used in the world class. The singleton pattern is used in the `__init__` function this is a constructor and is called every time an instance of the World class is created. In figure 5.2.1.2, a global private variable `__singleton` is initialised to a none object. When the user creates an instance of the world class, the `__init__` constructor checks if the `__singleton` contains a object, if it does an error is thrown (as an instance of the world class has been create) if it doesn't the singleton variable is initialised to the world, this is recorded if another instances of the world class is created by the user.

```
__singleton = none
.
.
__init__(self,width,height):
   if World.__singleton:
        raise Exception( "you cannot have more than one world in a single game")
   World._singleton =self
```

Figure 5.3.1.2: Program code for the singleton pattern used in the World class

5.3.2 Implementation

Pygame's `Sprite` module has classes, which simplifies creating sprites (actors) in a game. Pygame Simplified uses the container class `Group` in the `Sprite` module to create multiple actors and the `sprite` class to check if an actor collides with another actor in an actors group.

5.3.2.1 Multiple actors

As we saw in *Chapter 2* in the *'Catch A Fish'* game, Pygame's `sprite` class needed to be inherited so the container class `Group` could be called to create multiple goldfishes and clown fish. This requires programmers to use Object Oriented Programming, Pygame Simplified eliminates this requirement by inheriting the `Sprite` class in the `Actor`'s class. Figure 5.3.2.1.1 (below) shows how Object Oriented Programming is used in the actor class so users do not need to worry about this in their game.

In Pygame, Sprite groups allow multiple actors which share similar characteristics to be added to one group, checking for collisions, drawing actors, updating actors and making actors behave are performed for all sprite's in that group instead of having to do this for every single sprite. Sprites must differ in some way the exact same sprites cannot be added to the sprites group more than once, just like you wouldn't duplicate a class to create two of the same actors in a Pygame program, sprites in sprite groups are not duplicated.

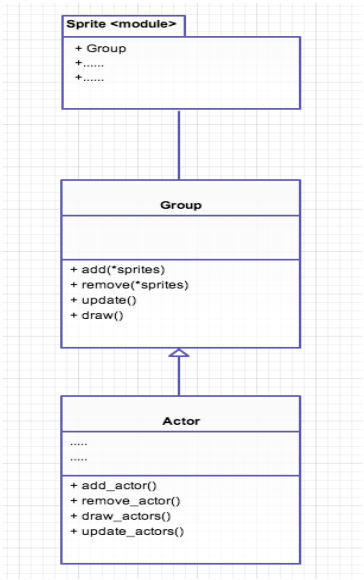


Figure 5.3.2.1. 1: A class diagram, the abstraction of the actor group from the `Group` class

We refer to sprite groups as actors groups in Pygame Simplified. Methods from Pygame's `Sprite` class are encapsulated into functions. The `make_actor_group()` function initialises Pygame's Sprite group to a global variable `actor` program code for this is illustrated in figure 5.3.2.1.2 below. This actor variable is calls functions in the group class (figure 5.3.2.1.1), a different actor is created for each instance of the world object.

```
def make_actor_group(self):
        self.actors = pygame.sprite.Group()
```

Figure 5.3.2.1. 2: Program code for the make actor group function

Actors called on this function acts as a 'super actor' actor's, which share characteristics with this actor but may differ in some way, are added to this group. For example, in the 'Catch A Fish' game goldfishes, clown fishes are fish. They both move from the same position on the screen and a shark eats both. However, they both have a different image. Figure 5.3.2.1.3 below shows how the `fish` instance can be used as a 'super actor' (line 10 using the `make_actor_group()` function) this holds the `goldFish` and `clownFish` in line 11. The fishes can still be set to have different characteristics, in line 15,16 the fishes differ in

image. Actor groups and 'super actors' reduce the lines of program code for the user program code is not repeated. In line 22 both clown fishes and goldfishes are updated once, they are also drawn once in line 26.

```
1.  from PGS.PygameSimplified import *
2.
3.  fishWorld = World(630,390)
4.
5.  clownFish = Actor()
6.  goldFish = Actor()
7.  shark= Actor()
8.  fish = Actor()
9.
10. fish.make_actor_group()
11. fish.add_actor((clownFish,goldFish))
12.
13. fishWorld.set_background("images/sea.png")
14. goldFish.set_image("images/goldfish.png")
15. clownFish.set_image("images/clownfish.png")
16. shark.set_image("images/shark.png")
17.
18. clownFish.set_location(100,100)
19. goldFish.set_location(200,100)
20.
21. while True:
22.     fish.update_actors()
23.
24.     fishWorld.draw_background(0,0)
25.     fishWorld.draw_actor(shark, 200,400)
26.     fish.draw_actors(fishWorld.window)
```

Figure 5.3.2.1. 3: Program code to demonstrate 'Catch A Fish' example using a 'super actor' in Pygame Simplified

In Pygame actors can be duplicated, an actor's image can be drawn n times this is illustrated in the '*Catch A Fish'* game introduced in Chapter 2 Figure 5.3.2.1.4 (below) shows how this is done. Pygame Simplified reduces the six lines of program code required for this to one. Programmers can create more than one actor of the same actor through the `generate_rand_actors()` function. This function allows more than one actor of the same actor to be added to a world at a random position. This function is preferred in the 'Catch A Fish' game. In this game, we want to generate many clown fish and goldfish and regenerate them as the shark eats them. This function takes an integer number, which the number of actors to be generated, the image of the actor (as they are duplicates of the same actor) and the world the fishes are drawn on. Figure 5.3.2.1.5 shows the implementation of this function. A for loop allows the number of actors entered by the user to be created. This uses the actor's group `add_actor()` function, which adds a single created actor into an actors group. All the actors in this group are iterated through, the x and y positions of all the actors are set to a random location using Pythons `randrange()` function, the image of these actors are set.

```
for i in range(5):
    clown=ClownFish("clownfish.png")
    # Set a random location for the clown fish
    clown.rect.x = random.randrange(fishWorld.screen_width)
    clown.rect.y = random.randrange(fishWorld.screen_height)
    clownfish_list.add(clown)
    all_sprites_list.add(clown)
```

Figure 5.3.2.1. 4: Program code for generating many actors in Pygame's 'Catch A Fish' game

```
def generate_rand_actors(self, number, image, World):
        for i in range(number):
            self.actor=Actor()
            self.add_actor(self.actor)
            for actor in self.actors:
                self.actor.x = random.randrange(World.width)
                self.actor.y = random.randrange(World.height-50)
                self.actor.set_image(image)
```

Figure 5.3.2.1.5: Program code from Pygame Simplified's Actors class for generating random actors

10 goldfishes can be generated in the following way using Pygame Simplified version II API:

```
goldFish.generate_rand_actors(10,"images/goldfish.png",fishWorld)
```

To enforce the use of GCSE constructs, the `remove_actor_from_group()` function removes one actor from a `generate_rand_actors()` group, to remove *n* actors a for loop is required. This function still uses Pygame's `remove()` function from the `group` class. The `remove_actor_from_group()` function also checks if the actors group is empty by calling the `actors_group_length()` function in the Pygame Simplified's Actors class. An error is thrown if this group is empty (Figure 5.3.1.6 below).

```
def remove_actor_from_group(self):
        if self.actor_group_length()==0:
            raise Exception("You cant remove from an empty actors list.")
        else:
            for actors in self.actors:
                for k in range(1):
                    self.actors.remove(actors)
                break
```

Figure 5.3.2.1.6: Program code from Pygame Simplified's Actor class to remove an actor

Figure 5.3.1.7 below shows how three goldfishes can be removed from the `generate_rand_actors()` group in Pygame Simplified:

```
for n in range(3):
      goldfish.remove_actor_from_group()
```

Figure 5.3.2.1.7: Program code to remove an actor from a generate random actors group using Pygame Simplified

5.3.2.2 Collisions

Pygame's `spritecollide()` is used to check if an actor has collided with another actor in an actors group. Figure 5.3.1.8 below shows this.

```
#if actor collides with actor group, remove actor
    def collide_group(self, actor):
         collision = pygame.sprite.spritecollide(actor, self.actors, True)
         if collision:
             return True
         return False
```

Figure 5.3.2.1. 8: Program code from Pygame Simplified's Actor class which checks for collisions

This function takes an `Actor` as an argument, it checks if the actor argument collides with another actor in the actor group. The collision variable is set to a list, which holds all of the actors that are intersecting with another actor. Pygame's `spritecollide` takes an argument, which is the actor that is checked for collision, the actors group and a Boolean argument set to `True`. The if statement checks if there is a collision, If there is Pygame `spritecollide` takes care of removing the actor in the actors group.

5.3.2.3 Sound and Timer Class

Implementation of other functions in the Pygame Simplified Version II are summarised in table 5 below.

Table 5: Implementation of methods in Pygame Simplified Version II

Method	Description
Sound Class The `Sound` class is where all the methods for displaying sounds in a game are. In order to use methods (detailed below) in this class, an instance of the `Sound` class: `mysound` = Sound().	
`set_sound(filename)`	This method sets a sound that is to be played. It takes a string argument `filename` this is the class path of where the sound file (that is to be place) is saved. This method uses Pygame's `mixer.Sound()` method which takes a filename.
`play()`	This method plays the sound that is set. This method uses Pygame's Sound class. The sound object created is called on this method.
`pause()`	This method pauses the sound that is set. This method uses Pygame's Sound class. The sound object created is called on this method.
`stop()`	This method stops the sound that is set. This method uses Pygame's Sound class. The sound object created is called on this method.
`set_volume(volume)`	This method sets the volume of the sound that is set. This method takes an float argument which represents how high or low the sound should be played at. This method uses Pygame's Sound class. The sound object created is called on this method.
Timer The `Timer` class is where methods (detailed below) for incrementing and decrementing a timer are. To use either of these two methods the class name `Timer` must be called on the method you would like to use: `Timer.decrement_timer(timer)` Where the argument is an integer representing the number the timer should be incremented or decremented by.	
`decrement_timer(timer)`	This method decrements a timer that is set. It takes an integer argument timer, which is the number that the timer should be decremented from.
`increment_timer(timer)`	This method increments a timer that is set. It takes an integer argument timer, which is the number that the timer should be incremented from.

5.4 Program code generator

The Pygame Simplified program code generator is an additional feature to Pygame Simplified. The program code generator uses Pygame Simplified version II. It is not required to use the Program code generator to use the Pygame Simplified API.

5.4.1 Wireframe GUI designs

Before the Pygame Code generator is implemented a how the GUI should look is sketched. This design is implemented. The Program code generator has two GUI windows: a program code generator form window and an IDE window. When the user inputs the details of their world and actor, the information is interpreted into program code this is generated into a text editor. The user can run this program code.

Program code generator form window

Figure 5.4.1.1 below is a hand drawn sketch of the program code generator form. The form is split up into three sections: world's details, actor's details and recorded actors. In Figure 5.4.1.1 an annotation of the functionality of each of these sections is given.

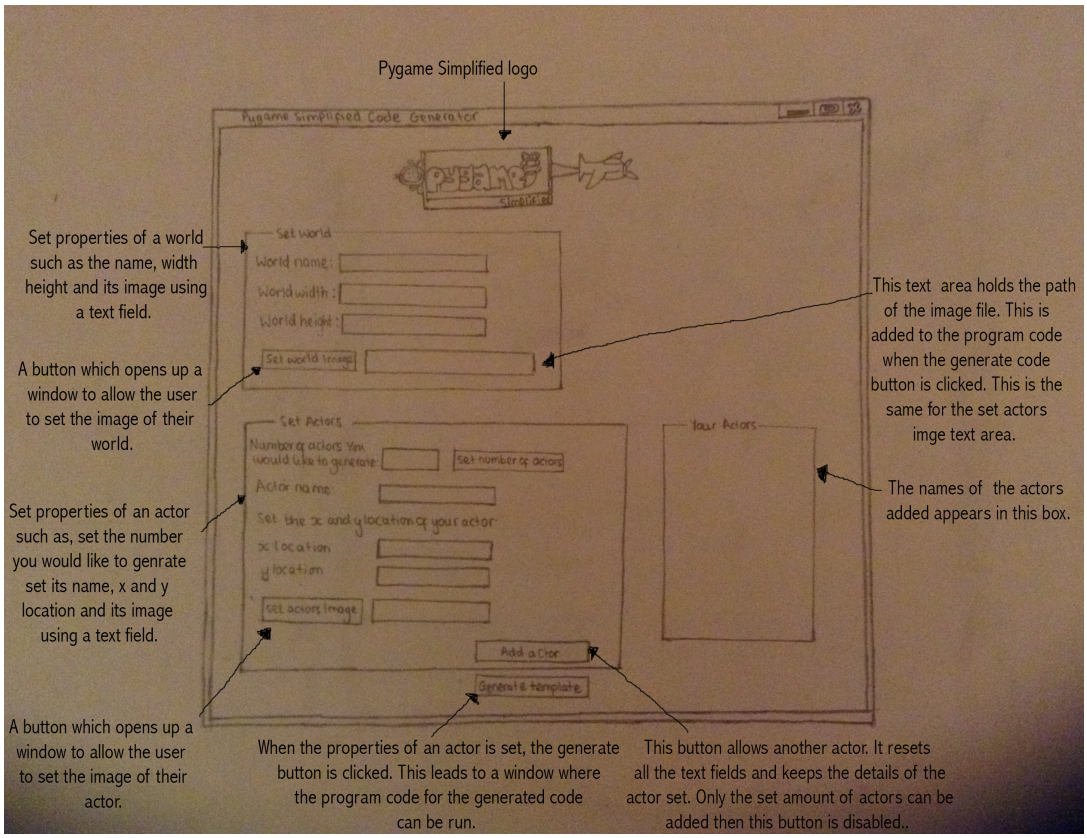


Figure 5.4.1. 1: A sketch of the program code generator form window

IDE window

When the user has entered the details of their actors and world, they generate the program code and are presented with a text editor. They can run their generated program on this text editor. This text editor supports functionality such as saving program code and opening program code. An annotation of the functionality of each of these sections is given in Figure 5.4.1.2.

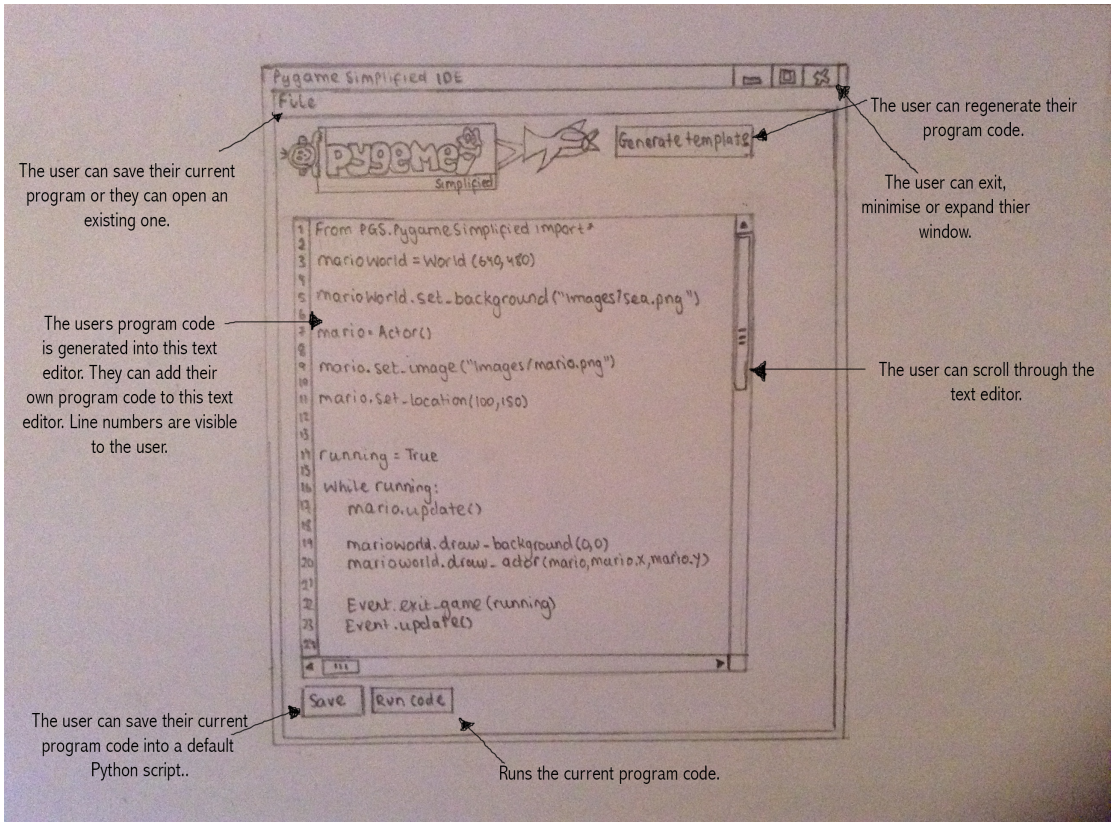


Figure 5.4.1. 2: A sketch of the IDE window

5.4.2 High level design

Pygame Simplified code generator has two Python scripts each script has two classes:

- PGSMainCodeGenerator.py
  - MainWindow.class
  - GroupedBox.class
- PGSTextEditor.py
  - PGSTextEditor.class
  - FromWidget.class

PGSMainCodeGenerator.py

This class has functions for the main GUI code generator window. This python script has two classes: The `MainWindow` class and `GroupedBox` class. The `MainWindow` class has the GUI components for GUI generator window, the buttons, text labels, the Pygame Simplified logo and text areas are created in this class. The `GroupedBox` class has a constructor which is called in the `MainWindow` class to create the rectangles that separate actor's details from the world details.

PGSTextEditor.py

This class creates the Pygame Simplified IDE. The `PGSTextEditor` class creates the text editor components for the Pygame Simplified text editor. The FromWidget class has functionality that makes the text editor into a simple IDE, such as the run feature (which runs the program code entered into the text editor).

5.4.3 Low level design

The Pygame Simplified GUI program code generator is implemented with the model, view and controller design pattern. Functions are split and grouped with model, view and controller. Table below explains the implementation of classes and functions in the `PGSMainCodeGenerator` and `PGSTextEditor` Python scripts.

Table 7: Implementation of functions in Pygame Simplified's program code generator

PGSMainCodeGenerator.py	
MainWindow Class	
Functions	Description
View functions	
disable_actor_details()	This function disables the actor's details until the user sets the number of actors they would like to generate.
create_text_areas()	This function creates all the text areas for code generator GUI. The size and location of each text area is set.
create_buttons()	This function creates all the buttons for code generator GUI. Each button is set to false until all the details of actors/world are entered.
create_labels()	This method creates all the text labels for code generator GUI. The text and location for each label is set.
set_background_colour()	This function sets the background colour of the GUI.
set_pgs_logo_image()	This function sets the Pygame Simplified logo to the GUI.
clear_text_areas()	The text field for the actor's details are cleared when the user adds an actor.
Model functions	
number_of_actors()	This function gets the number of actors in the `set_number_of_actors` text field and stores this inside a variable. This function is called when the user clicks the set number of actors button.
initialise_actors_details()	This function initialises the data structures that hold actor's details.
check_actor_exsits()	This function checks if an actor name has been used. If it has, then the actor exists and the user is asked to use another name.
generate_code()	This function preforms the logic for generating the program code from the details the user has entered. The program code generated is placed into python a template file. The details of the actors and worlds stored in the data structures are retrieved and passed into the Pygame Simplified functions, which are generated into a text editor. This method is called when the user clicks the generate code button.
Controller functions	
open_text_edit_window()	This method opens the text editor window when the user clicks the generate code button.
open_actor_image_file()	This function opens a file using PYQT4's FileDialog class. The class path of the file chosen is taken and reduced to the folder name and the image name. The actors image text area is set to this file path, which is passed to the Pygame Simplified's `set_image()`

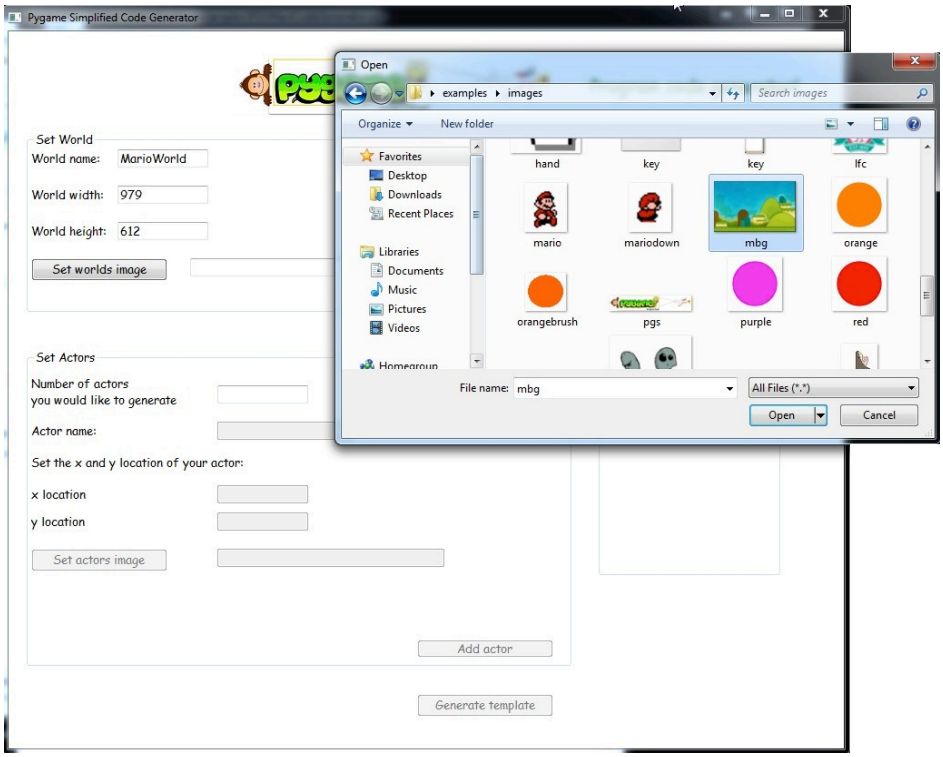
	function, to set the actors image.
open_world_image_file()	This function does the same as the function above, but sets the world's image.
signals()	This function holds signals for this Python script. Signals connect the button with an action.
GroupedBox Class	
	This class inherits the QWidget. An instance of this class is called in the MainWindow class, so a grouped box (rectangles) can separate actors details from worlds details.
PGSTextEdit.py	
PGSTextEditor Class	
SaveFile()	This function performs the save file action for when the user choose to save their file through the file menu.
openaFile()	This function does the same as the function above, but opens a file.
FromWidget Class	
create_buttons()	This function creates all the buttons for the text editor window.
buttons_controller()	This function holds the signals for the buttons created. Signals connect the button with an action.
run_code_button_clicked()	This function creates an object of Pythons interpreter class. The text from the text editor is retrieved and this is passed in the `runcode()` function in the interpreter class. The text in the text editor is run through this function.
gen_code_button_clicked()	This function askes if the user is sure they would like to regenerate their program code, if they do their program code is generated.
save ()	This function performs a save action for the save button. If the user saves their program code, it is saved in a default python script *'mygame.py'*.
askQuestion()	This function asks the user if they are sure they would like to save their Python script (a dialogue window). If they select yes, the `save()` function is called.

5.4.4 A run through of the program code generator

This section shows the different steps involved in using Pygame Simplified program code generator.

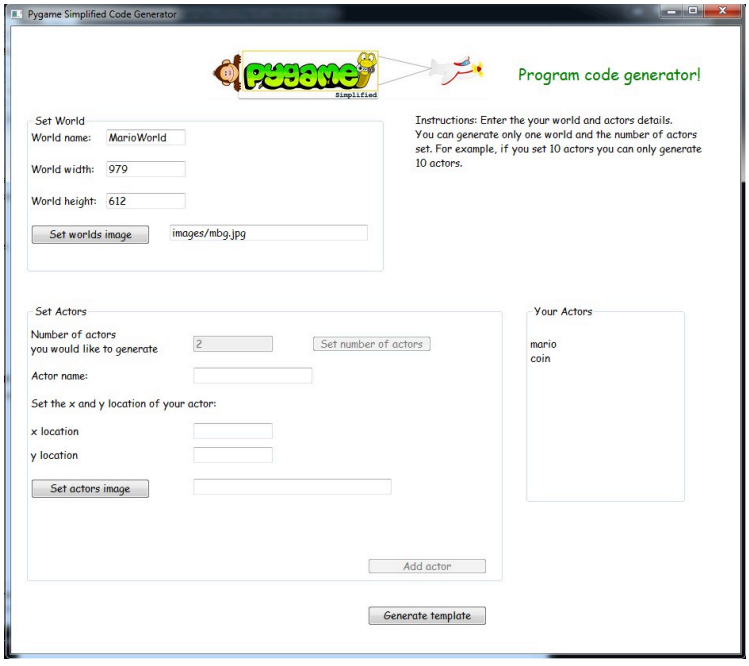
Step 1: Set World properties

The name of the world is set, the width and the height of the world are also set. Clicking on the set world image button sets the image of the world this opens the file chooser. (This is shown in the image below).



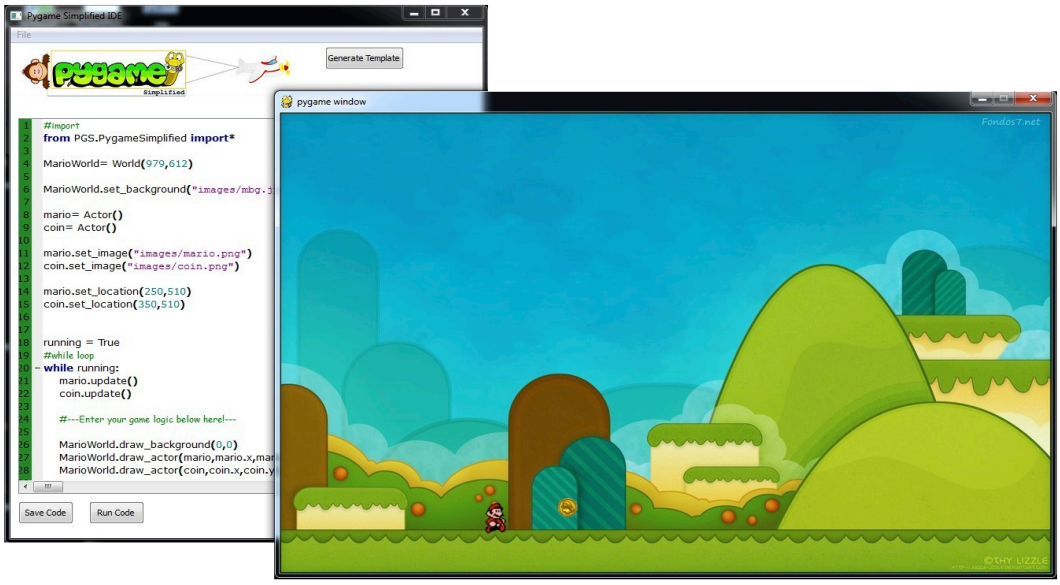
Step 2: Set Actors properties

The number of actors to be generated is set. Above two actors `Mario` and `Coin` are set. The details of this actor are set, this is visible to the user in the 'your actors' section. The actors `Mario` and `Coin` are recorded so the user knows they have entered this (this is shown in the image below).



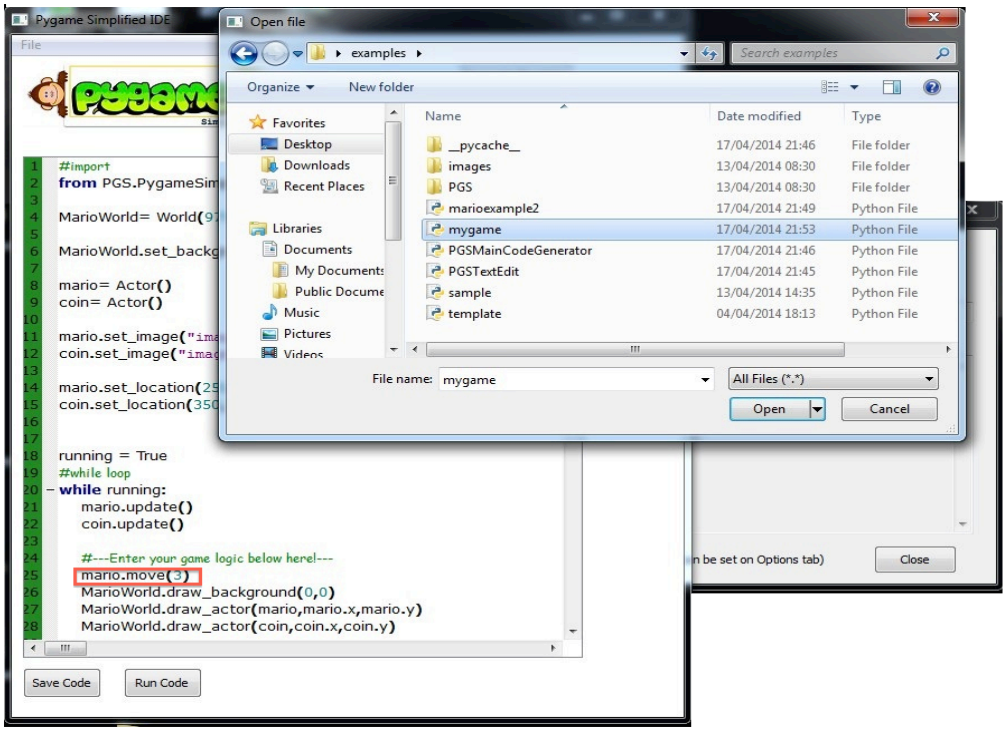
Step 3: Generate program code

The details set by the user is generated onto the Pygame Simplified text editor (image below). The user does not need to add any program code at this point. When they click the run button, a window with the set properties appears.



Step 4: Add your own program code and save your program code.

The user can add their own program code (highlighted in red in the image below) and they can save their program code. The user can open an exiting Python script by clicking the file button.



5.5 Summary

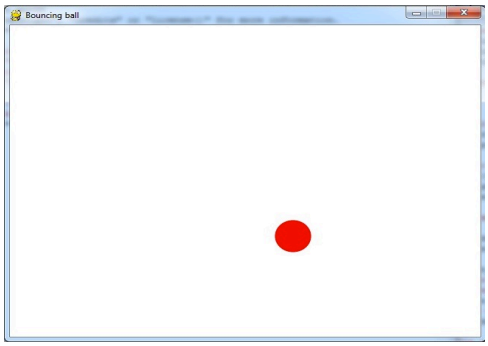
This chapter covered the design and implementation of the Pygame Simplified API (version I and version II). This chapter also covered the design and implementation of the program code generator. The next chapter (*Chapter 6*) shows the different games that can be made with the Pygame Simplified API.

Chapter 6 Example Games using the Pygame Simplified API

This chapter shows the range of games that can be programmed using Pygame Simplified API. Section 6.1 gives examples and details of these games, and a difficultly rating. Chapter 6.2 compares the Pygame Simplified Balloon game to the original balloon game introduced in *Chapter 3 Section 3.2*. The solution to problems Identified in the Pygame balloon game detailed.

6.1 Example Games

A goal of Pygame Simplified (Chapter 2 – section 3.4) is to be able to program a variety of games using the Pygame Simplified API. Below are examples a programming difficulty rating is given. All these games are available on the Pygame Simplified website: http://webprojects.eecs.qmul.ac.uk/fa303/pgs/

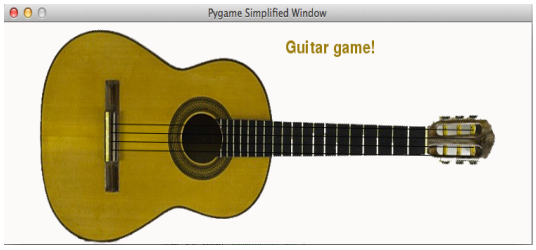


This is a simple bouncy ball example. A ball bounces of the wall. This is an example for the very beginner Pygame Simplified developers.

Programming difficulty rating: very easy

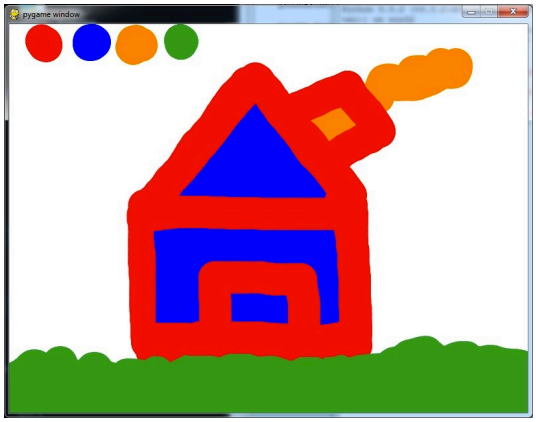
This is a piano game. The user plays the piano by clicking on the black or white keys. When a piano key is pressed, a sound representing that key is played. This game is a representation of the variety of games that can be made using the Pygame Simplified API.

Programming difficulty rating: easy



This is a guitar game. When the user clicks on a string, a sound for that string is played.

Programing difficulty rating: easy

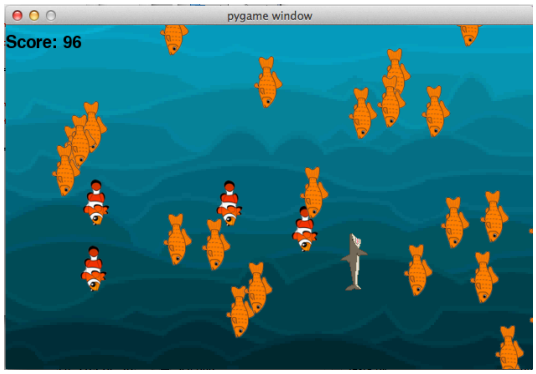


This is a paint application. This is another example of the variety of applications that can be made using Pygame Simplified. The colours on the left hand side of the screen represent the different paintbrushes that can be used. The brushes are dragged to draw a painting.

Difficulty rating: easy

This is a simple car racing game. This is an arcade type game. The car moves lane by lane and when it reaches the bottom of the screen, it the car re-appears from the top. A lap counter is kept which it is incremented.

Difficulty rating: very easy



This is 'catch a fish' game first introduced in chapter 2. This game uses Pygame Simplified's version II. The shark eats the goldfish and clown fish. More points are scored for eating the clown fish.

Difficulty rating: Medium

6.2 A Comparison of Pygame Simplified to Pygame

Using the Lines of code metric (LOC), where comments and empty spaces are ignored as the Python interpreter ignores these, the balloon game programmed in Pygame Simplified is programed in 51 lines of program code. Whereas, the balloon game programmed using Pygame alone is programed using 101 lines of program code. Programming a game in Pygame Simplified can be done in half the program code. (See *Chapter Appendix* A, 10.1 for full program code).

In Pygame Simplfied's balloon game, the only Object Oriented Programming required is when instances of classes are created so functions from these classes can be called. Figure 6.2.1 (below) shows this. If the program code generator is used, they do not need to worry about the Object Oriented Programming to create `Actors` and a `World`.

```
world = World(700,400)

balloon = Actor()
firedArrow = Actor()
arrow = Actor()

mytimer = Timer()
score_text = Text(world)
..
```

Figure 6.2. 1: The only Object Oriented programming used in Pygame Simplified

Program code in Pygame Simplified is readable you can tell what each line of program code does without needing to read comments. Figure 6.2.2 below shows how collisions are manipulated in Pygame Simplified and Pygame.

Collisions in Pygame Simplified

```
for arrows in firedArrow:
        firedArrow.move_by(0,-10)
        if balloon.collide_group(arrows):
            score = score+1
            balloon_sound.play()
```

Collisions in Pygame

```
index1=0
        for bullet in track_arrows:
            bullrect=pygame.Rect(arrow.get_rect())
            bullrect.left=bullet[1]
            bullrect.top=bullet[2]
            if target_rect.colliderect(bullrect):
                score+=1
                pop_ballon_sound.play()
                targets_array.pop(index)
                track_arrows.pop(index1)
            index1+=1
```

Figure 6.2. 2: Program code to implement collisions in a game using Pygame Simplified and Pygame

Pygame Simplified does the exact same as Pygame but with less lines of program code. One compromise was made in the Pygame balloon game (*Chapter 3*), GCSE constructs were used without any Object Oriented Programming but program code was difficult to read. Pygame Simplified eliminates this compromise to no Object Oriented Programming if program code generator is used, or only instances of objects are created if Pygame Simplified alone is used. In addition, GCSE constructs are simple to use in Pygame Simplified, this is illustrated in figure 6.2.2 above.

6.3 Summary

This chapter examined the different types of games that can be made with Pygame Simplified. It compared Pygame to Pygame Simplified. *Chapter 7* looks at the feedback given by teachers and students who have evaluated Pygame Simplified.

Chapter 7 Evaluation

This chapter covers the evaluation of Pygame Simplified. Section 7.1 examines the process of evaluation. Section 7.1.1 introduces the tools used to evaluate Pygame Simplified. Section 7.1.2 looks at the findings from the evaluation of Pygame Simplified. Section 7.2 examines the project achievements, the achievements of Pygame Simplified are compared to the objectives and requirements set for Pygame Simplified in *Chapter 3 and Chapter 4*. Finally, section 7.4 details future work for Pygame Simplified.

7.1 Evaluation process

Teachers evaluated Pygame Simplified. But first, Pygame Simplified was beta tested by Computer Science students this ensured any problems or improvements to Pygame Simplified were made before teacher's evaluated Pygame Simplified.

A fast-track ethics application was made before teachers could evaluate Pygame Simplified. This ensured evaluating Pygame Simplified was ethical. This application highlights that it is entirely up to teachers to evaluate Pygame Simplified with their students. The research ethics application was successful and the project was deemed low-risk. Guidelines in the ethics application are outlined in the communication with teachers. (The application and Approval letter can be found in *Chapter 10 Appendices,* B).

Teachers on the Teaching London Computing Project (TLCP) project volunteered to evaluate Pygame Simplified through Dr William Marsh. An email (see *Chapter 10 Appendices*, C) was sent to the teachers who volunteered to evaluate Pygame Simplified some teachers replied with interest and agreed to evaluate Pygame Simplified.

A CAS account was created. Pygame Simplified is added as a resource for teachers to evaluate (see *Chapter 10 Appendices*, C part 10.7). Some teachers volunteered to evaluate Pygame Simplified.

7.1.1. Evaluation tools

A questionnaire and a blog are designed to gather feedback on the Pygame Simplified API. (See *Chapter 10 Appendices*, D). A total of two teachers took the questionnaire (other teachers chose to give the feedback through email). The questionnaire has a set of questions asking teachers about the suitability of Pygame Simplified for GCSE students. Teachers who tested Pygame Simplified through CAS commented on the resource created.

In the email was sent to the teachers who volunteered to evaluate Pygame Simplified, a link the Pygame Simplified website: http://webprojects.eecs.qmul.ac.uk/fa303/pgs/
which has all the resources for Pygame Simplified, a link to a questionnaire and the blog is given to teachers so they could give their thoughts and experience of using Pygame Simplified, the information gathered from this is used to make improvements on Pygame Simplified.

7.1.2 Findings

Improvements
Results from the questionnaire indicate improvements could be made to Pygame Simplified.

In the questionnaire the following question is asked:

"Do you think installation of Pygame Simplified would be easy to follow by GCSE students? Please rate accordingly installation for Pygame Simplified is:"

Figure 7.1.2.1 below shows the result from this question. One teacher said:

"There were some issues initially but these were dealt with quickly and effectively from then on the installation was easy".

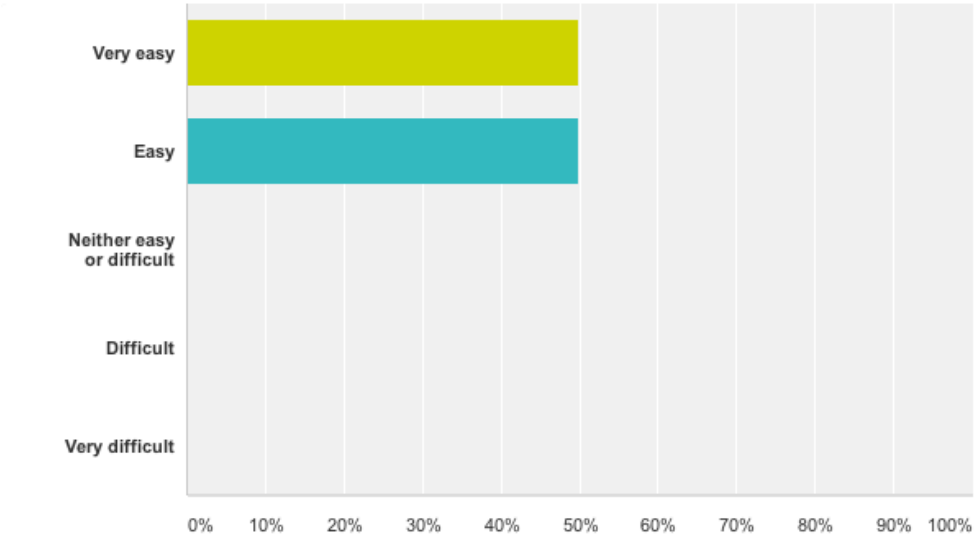


Figure 7.1.2. 1: Result from questionnaire- on how easy it is to install Pygame Simplified.

Another teacher e-mailed me regarding installation: *"I have asked the technician to install Python 32 bit and then Pygame on the school network. Hopefully this will be ready next week. Is it necessary that Python 3.3.2 installed? Can it be a later or an earlier version? News from our network provider is that 3.3.1 is more stable to carry out more complex codes."*

To answer this question Pygame Simplified was tested with Python 3.3.1. I found that Pygame Simplified could be used with Python 3.3.1. To eliminate any further confusion, Pygame Simplified has been tested Python 2 and all versions of Python 3.

I also found that Pygame 32-bit is installed, Python 32-bit must be used. The 64-bit version of Pygame game is not available on the official Pygame website however a website provides the 64-bit version is available. This teacher was emailed with the following information (see *Chapter 10, Appendices,* C). The documentation for installation of Pygame, Pygame Simplified and Python is updated with all this information on the Pygame Simplified website. No further issues of installation have been reported.

Megan Ryan a student who tested Pygame Simplified said:

"Whilst doing the paint worksheet I had no idea how to compile or run the application when you asked me to. As I said I've not used Python before."

Soon after this, a solution to this problem was updated on the worksheet.

Suggestions

After evaluating Pygame Simplified, George a teacher at Haringey School suggested Pygame Simplified could have: "*A simple GUI allowing adding actors to the world, then open the code view to add code controlling the actors etc. This make shift solution will definitely put Python in a secure position with GCSE and KS3.*" George's comment was taken on board a program code generator, which sets properties of the world and actors, is available.

Andy Diament a computing lecturer at Penwith College evaluated Pygame Simplified version I commented on CAS: "…It *will be interested to see how this progress".*

Following his comment, additional classes and functions are integrated into Pygame Simplified version II.

7.2 Project achievements

Below, a description of how the objectives of this project are met is given. These are the achievements of this project (results from the questionnaire is used, this is available in *Chapter 10 Appendices*, D).

7.2.1 Objectives met

The following goals and objectives for Pygame Simplified were met:

- ✓ **An API suitable for GCSE students, which allows students to program computer games using GCSE constructs.**

100% of teachers who took the questionnaire said they agreed when asked if Pygame Simplified is not too difficult for GCSE students. GCSE constructs are encouraged on users in the Pygame Simplified version I and Pygame Simplified version II.

- ✓ **An engaging tool for students at GCSE level**

100% of teachers who took the questionnaire said they agreed when asked if Pygame Simplified is engaging for GCSE students. The variety of games that can be programmed using Pygame Simplified is an indication that Pygame Simplified is engaging.

- ✓ **An API, which minimises Object Oriented Programming**

The Pygame Simplified API is designed so that Object Oriented Programming is minimised. In fact, the only required knowledge of Object Oriented Programming is when users create instances of the class's available in the Pygame Simplified API. This is eased with the program code generator, where the users program code for creating objects of `Actors` and `World` is done for them once they enter the details for `Actors` and `World`. The program code generator suits GCSE students George Dong (teacher) suggested *"…This make shift solution will definitely put Python in a secure position with GCSE and KS3"*.

- ✓ **An adequate API that can be used by teachers to help students with controlled assessments.**

John Andrews, a teacher of Computer Science at St Gregory the Great Catholic School Oxford commented on Pygame Simplified: *"...As my Y10 students are doing GCSE coursework now and 'simplified' may make their hangman games more involving."* This gives strong indication that Pygame Simplified can help students with their controlled assessments.

- ✓ **A structure to represent actors an alternative to using Pythons complex data structures.**

Pygame Simplified's Actors group is an alternative structure to using Pythons complex data structures. The actors group encapsulates Pygame's Sprite group. Actors with similar properties can be added to a single sprite group. The properties they share can be applied at once. For example, if they move at the same time, then the `move()` function is called on all the actors in this group. The `generate_ran_actor()` function is an alternative actors group, which holds a number of actors defined by the user. `Lists`, `tuples` or `dictionaries` are not required to manipulate actors. Actors groups also contribute to minimising Object Orientated Programming in a game users do not need to make classes they only need to make Actors groups.

- ✓ **Improvements of the Pygame Simplified API based on feedback from evaluators.**

The development of the Pygame Simplified Code generator is an achievement. It is a solution to a problem identified in Pygame Simplified by a teacher. Although, teachers have not tested the program code generator, it meets the requirements set. Basic usability testing is carried out such as:

- Checking if the user has already entered an actors name
- Check if the user has already used the actors name as a world name
- Ask if the user would like re-generate or save their program before these actions are done.

Pygame Simplified meets other objectives. Resources for teachers such as: worksheets, template for games, tutorials on how to use methods in Pygame Simplified and example programs programed in Pygame Simplified is available for teachers and students. All of this is available on the Pygame Simplified website.

7.2.2 Feedback by evaluators of Pygame Simplified

Teachers agree that Pygame Simplified is an engaging tool, which simplifies programming for GCSE students:

Rosaleen Moore, an ICT teacher commented on CAS: *"I haven't used Pygame to any great depth but this looks like a great tool to introduce events & classes at an engaging level".*

Andy Diament, a computing Lecturer at Penwith College: *"I don't know Pygame, but ive been meaning to try but this seems to reduce it to four classes and seems an easy route into making a game...this is less daunting…I teach in a college and this may be useful for my weakest students to learn some programming…Will be interested to see how this progresses... These are very useful modules. Thanks"*

Javier de Las Heras an ICT Co-ordinator: *"...It looks great. I am going to try it myself, then with a group of year 10 students".*

Geroge Dong, an ICT teacher at Haringey School: "*I have followed your "cargame" worksheet and made it to the end! Well done! It is so much easier to use than pygame! A good problem identification skill and a good solution too.*"

Interest from other teachers where present (see Chapter 10 Appendices, E).

Megan Ryan, a student: "*That was fun!...I have not used Python before and found it really good fun to use. The examples at the end of the worksheet were really useful, reinforced what you had explained in the main text and got me interested in further enhancing what I had already created. The installation guide was really easy to follow, screenshots were useful. The examples you provide in the tutorial page are useful also, helped me understand the concepts I was reading about.*"

Maximilian Droog Hayes, a student: *"In very few lines of code you could actually see an interesting, moving output. I remember when I first started to program; it would take a lot more confusing programming to see something appear on the screen. The guide clearly talks you through how to get something initially working; something you can easily mess about with and alter in order to see what each bit of code does and change the output on the screen. I found it very engaging to be able to make a game so quickly and expand on it with limitless possibilities!"*

7.3 Project limitations

Pygame Simplified has limitations. Implementation of the program code generator can be limitless. Further development of the Pygame Simplified program code generator has been omitted due to time constraints. Alpha testing has been carried out on the programme code generator. The program code generator is yet to be evaluated by teachers, further few tests are a required before the program code generator is realised for evaluation. The Pygame Simplified program code generator needs to be packaged into an executable. Once Pygame Simplified code generator is packaged and small alpha testing is carried out, it is ready for beta testing by teachers and students. The program code generator fulfils all the requirements it sets. Installation instructions on how to download supporting materials for the program code generator are given *in Chapter 10 Appendices –supporting materials*.

7.4 Future work

Improvements to the Pygame program code generator can be made, such improvements include:

- Reduce the errors users can make and improve the usability of the program code generator.
- Package Pygame Simplified into an executable. Pygame, Python, PyQT4 and Pygame Simplified would need pre-installed on the executable.
- Enhanced features such as:
    - Delete actors and edit actors that have already been added.
    - A choice to add sound and timer to a game.

Further development of the program code generator can be done in increments. The incremental life cycle can be followed where requirements specification, design, implementation and testing are completed in a number of increments.

Chapter 8 Conclusion

8.1 Challenges faced in this project

Installation of software did cause some problems. My laptop operates on a Macintosh OS X. Pygame 3 is the latest release of Pygame and is only compatible with Python 3. This is the latest version Python. Python 3 is realised for windows only. This problem was resolved Windows on Boot Camp is installed on my Macintosh laptop so that Pygame Simplified could be developed.

Many teachers said they would evaluate Pygame Simplified but couldn't. Except for John Andrews, no other teachers evaluated Pygame Simplified version II. Despite this, the evaluators I had from Pygame Simplified version I and John was sufficient to draw a conclusion.

8.2 What I have learnt from the project

I am very happy with the outcome of this project. This project required a good understanding and application of many areas in Computer Science. I have applied knowledge and skills that I have learnt in many of my university modules. I had no prior knowledge of Python before this project. Programming in Python is a skill I have picked up during this project. I have thoroughly enjoyed working on this project and developing it in Python.

8.3 Conclusion

More can be done to enhance Pygame Simplified. However, Pygame Simplified meets all the objective, aims and requirements for this project. The problem description of this project is many students find programming difficult. Even with the resources (Turtle Graphics, Greenfoot and Pygame) aimed to engage and motivate students into programming have limitations, which disengage students or make programming difficult. This project solves this problem. *Chapter 7 Evaluation* shows teachers who have evaluated Pygame Simplified agree with this (this is evident in the results from teachers in the questionnaires and comments on CAS). Of course, any beginner programmer and not just GCSE students can use Pygame Simplified.

Table 2 in *Chapter 2* shows tasks examination board set in controlled assessments are computer games; students are required to program computer games and use GCSE constructs. Pygame Simplified can be used by teachers for controlled assessments or help students with controlled assessments, not only does the John Andrews (A teacher) agree with this but the Pygame Simplified API requires GCSE constructs to be used in order to program games. We saw in *Chapter 6* Pygame Simplified programs are readable and well structured. There is minimal Object Oriented Programming required and a program in Pygame Simplified can be achieved in 50% less program code than Pygame (using LOC metric). A variety of programs can be programmed using Pygame Simplified, which aims to motivate different types of users, just like Greenfoot does (Examined in *Chapter 2*). The program code generator is an additional feature. Teacher George Dong suggests *" This make shift solution will definitely put Python in a secure position with GCSE and KS3."* Therefore, it is concluded that Pygame Simplified is an engaging tool, which simplifies programming for GCSE students and overcomes the limitations of Pygame, Greenfoot and Turtle graphics.

Chapter 9 References

1. Barnes, R. (2014). *'Why may it be difficult to engage GCSE students into programming?'.* Available: http://community.computingatschool.org.uk/forums/23/topics/2233. Last accessed 19 April 2014.

2. Tickle, L. (2012). *How teachers and schools are preparing for the new computer science GCSE.* Available: http://www.theguardian.com/teacher-network/2012/sep/13/computer-science-gcse-teachers-schools. Last accessed 19 April 2014.

3. Sweigart, A. (2008-2012). Who is this book for?. *Inventing Computer games with Python*. 2nd ed. United States: Creative Commons Attribution-Noncommercial-Share Alike. Page 6.

4. Oxford Cambridge and RSA. (2012). *Introduction of GCSE computer Science* OCR'. Available: http://www.ocr.org.uk/Images/72936-specification.pdf. Last accessed 19 April 2014.

5. Assessment Quality Alliance. (2014). *GCSE Specification.* Available: http://filestore.aqa.org.uk/subjects/AQA-GCSE-COMPSCI-W-SP.PDF. Last accessed 19 April 2014.

6. Edexcel. (2013). *GCSE in Computer Science Specification.* Available: http://www.edexcel.com/migrationdocuments/GCSE%20New%20GCSE/9781446908358_GCSE_Lin_CompScie_web.pdf. Last accessed 19 April 2014.

7. Clarkson, M. (2012). *Teaching with Python.* Available: http://community.computingatschool.org.uk/forums/1/topics/26#post_176. Last accessed 19 April 2014.

8. Dreadman, P. (2013). *Which programming language should I teach?.* Available: http://community.computingatschool.org.uk/forums/63/topics/1394#post_14292. Last accessed 19 April 2014.

9. Johnson, S. (2013). *Which programming language should I teach?.* Available: http://community.computingatschool.org.uk/forums/63/topics/1394#post_13982. Last accessed 19 April 2014.

10. Hartley, I. (2014). *Why may it be difficult to engage GCSE students into programming?.* Available: http://community.computingatschool.org.uk/forums/23/topics/2233#post_25218. Last accessed 19 April 2014.

11. Johnson, S. (2013). *Which programming language should I teach? .* Available: http://community.computingatschool.org.uk/forums/63/topics/1394#post_13982. Last accessed 19 April 2014.

12. Ames, D. (2013). *Is PyGame ok for the Gaming unit of AQA GCSE?.* Available: http://community.computingatschool.org.uk/forums/23/topics/733#post_6386. Last accessed 19 April 2014.

13. Kolling, M. (2010). Introduction. In: Hirsch, M. Dunkelberger,T. Horton, MJ, Haggerty,M. Michael, *An Introduction to programming with Greenfoot*. New Jersey: Pearson. p2.

14. Clarkson, M. (2012). *Greenfoot in GCSE Computing.* Available: http://community.computingatschool.org.uk/forums/23/topics/426#post_3651. Last accessed 19 April 2014.

15. Pygame Community. (2009). *About.* Available: http://www.pygame.org/wiki/about. Last accessed 19 April 2014.

16. Bissex, P. (2000). *Understanding Tuples Vs. Lists in Python.* Available: http://news.e-scribe.com/397. Last accessed 19 April 2014.

17. Python Community. (2014). *The Python Tutorial.* Available: https://docs.python.org/3.3/tutorial/. Last accessed 19 April 2014.

18. Brugess,T. (2009). *Wrapper for the Pygame input module.* Available: http://www.pygame.org/wiki/InputWrapper?parent=CookBook . Last accessed 19 April 2014.

19. Beazley, D. Holden, S. (2002). Introduction. In: *Python Web Programming.* United States: .p12.

Chapter 10 Appendices

Supporting Material

The supporting material is in a folder:
`./SupportingMaterial`
All the extra program code and emails in these appendices can be found in the supporting materials folder. Source code and how to install resources for the program code generator is also in this folder.

Appendix A. Program code

The images for all the program code can be found:
`./SupportingMaterial/ProgramCode/images`

10.1 Catch A Fish game – program code

`Pygame version`
The full source code for the Catch A Fish game in chapter 2 can be found in:

`./SupportingMaterial/ProgramCode/CatchAFish.py.`

`Pygame Simplified version`
The full source code for the Catch A Fish game in chapter 6 can be found in:

`./SupportingMaterial/ProgramCode/CatchAFishPygameSimplified.py`

10.2 Balloon Game – program code

The balloon game for Pygame Simplified in chapter 6 -source code can be found in:

`./SupportingMaterial/ProgramCode/BalloonGame.py`

10.3 Balloon Game Greenfoot – program code

The balloon game for Greenfoot in chapter 3 section 3.3 - source code can be found in:

`./SupportingMaterial/ProgramCode/Greenfoot`

Appendix B. Ethics application and Approval

10.4 Ethics application

The ethics application can be found in:
`./SupportingMaterial/ EthicsApplication.pdf`

10.5 Ethics approval

The ethics approval can be found in:
`./SupportingMaterial/fasttrackAbukar10.02.14.pdf`

Appendix C. Communication with teachers

10.6 Email sent to teachers

Below is a copy of the email sent to all teachers who agree to try Pygame Simplified. A reminder to this email and an email sent to teachers who commented on a thread I started can be found:

`./SupportingMaterial/CommunicationWithTeachers/Emails.PDF`

Email sent to all teachers who volunteered to evaluate Pygame Simplified.

Dear…

Thank you for agreeing to evaluate Pygame Simplified. Pygame Simplified is a simplified library for Pygame. Pygame Simplified aims to take a simpler approach of to programming games, and minimises the need for object-oriented programming, thus calling a set of functions is all you need to program games.

Pygame Simplified has the following features:

- How to install Pygame Simplified in easy steps.
- Supporting documentation for the Pygame Simplified library.
- Tutorial for Pygame Simplified (shows how methods and classes are used).
- Two worksheets for Pygame Simplified
    1. A car racing game
    2. A paint game.
- Invent your own computer game – where anyone can get involved and invent a computer game using Pygame Simplified. People are encouraged to share their games they have made using Pygame Simplified so others can learn from them. This features on the website.

The Pygame Simplified library is available on this public website (with all features included):

http://webprojects.eecs.qmul.ac.uk/fa303/pgs/index.html

For evaluating Pygame Simplified, this short questionnaire is designed (Any feedback is much appreciated):

https://www.surveymonkey.com/s/FMK2SW9

A short discussion is on this blog is also available:
http://fatimaabukar.wordpress.com/

You can find a direct link to this on the blog section of the Pygame Simplified website. This blog is for discussions and opinions on teaching resources in Python and any discussions and thoughts you may have on Pygame Simplified.

I am currently in the process of releasing an advanced library for Pygame Simplified. This is built on top of this simplified Library and contains enhanced features such as the ability to add multiple Actors. I will update the website within the next few days (hopefully). An additional worksheet for this library will be added and all the documentation for this will be added. This will not affect the running of this simple library, as versions of Pygame Simplified will be documented on the Pygame Simplified website. The current questionnaire is designed for the first version of Pygame Simplified. A discussion for the second (more advanced version) will be posted on the Pygame Simplified, if you would like to try this library.

Please do not hesitate to email me back with any queries.

Once again, thank you.

--
Fatima Abukar,
Final year undergraduate (Computer Science)
School of Electronic Engineering and Computer Science
Queen Mary, University of London,
Mile End Road, London E1 4NS.

10.7 Message put on Computing At School CAS

There are two messages on the CAS website which invite teachers to try Pygame Simplified, both are in:
`./SupportingMaterial/CommunicationWithTeachers/ResourceAddedToCAS.PDF`

Appendix D. Questionnaire

10.8 A copy of the questionnaire

A copy of the questionnaire can be found in:
`./SupportingMaterial/Questionnaire/copy.PDF`

10.9 The results from the questionnaire

The results from the questionnaire can be found in:
`./SupportingMaterial/Questionnaire/Results.PDF`

Appendix E. Interest shown by teachers in Pygame Simplified

Interest in Pygame Simplified shown by teachers through a 'like' on the Pygame Simplified resource on the CAS website.

Appendix F. Images used in Pygame Simplified

All the images used in Pygame Simplified API are from Google images. They have been edited using Adobe Photoshop.

http://www.google.com/imghp