**Introducing PyGame**

Now that we have some basic programming skills under our belts, we can leave our console programming behind for something a little more fancy. To do this we are going to use the PyGame toolkit.

Pygame is a set of modules that contain classes and functions that we can use to create graphical programs. To understand how they work, we first need to understand the difference between a console and an event-driven GUI (graphical user interface).

A **console** is a monochrome (black and white) screen with only keyboard input and text output. It does not allow colour, sound, graphics, or mouse input. A **Graphical User Interface** is a window that allows all of these things – buttons to press with the mouse, text boxes to write into, coloured graphics and so on. This document window is an example of a GUI.

Writing GUI programs is exciting, but a little more complicated than console programming. With the console you only had to worry about one thing at a time. With GUI programming we have to constantly be aware of many things – is the mouse moving? is it clicking something? – is the keyboard being pressed? If so – which key? the timer – are we supposed to update something? Programming to monitor events and constantly update the output is called **Event-Driven Programming**. For Event-Driven programming to work, we must constantly check for events using a **while** **loop** I call the **game loop**.

**Installing an IDE**

Although Replit recognizes the Pygame module, we probably should use a proper IDE (Integrated Developer’s Environment) installed into your computer for this unit. To do this, you will need to install:

* python version 3
* an IDE such as WingIDE 101
* the pygame library

All three of these items can be found in the Google Drive in the “downloads” folder and downloaded and installed onto your computer. You must install these items one at a time in the order shown above (python, then Wing, then pygame). If you have any difficulty doing this, contact your teacher.

Once all three are installed, create an empty program and type this into your WingIDE editor and run it.

import pygame

If it works, you should see no problems. If it doesn’t work, you will get this error and should contact your teacher:

File "None", line 1, in <module>

builtins.ImportError: No module named 'pygame'

If your code is working, save it as game1.py. DO NOT SAVE YOUR PROGRAM AS PYGAME.PY. That name is taken (for the Pygame module).

**Programming with Pygame**

There is a fair amount of coding just to get a blank screen to pop up. Here is the basic code:

import pygame

# --- screen setup ---

width = 320

height = 240

size = width, height # size is a tuple

screen = pygame.display.set\_mode(size)

#-------- game loop ----------

gameOn = True

while gameOn:

for event in pygame.event.get():

if event.type == pygame.QUIT: # if user hits X button

gameOn = False # quit the game loop

pygame.display.flip() # draw the screen

pygame.quit() # close the window

What does this code do?

First, it imports the pygame module into our program:

import pygame

Then it sets the window size:

width = 320

height = 240

size = width, height # size is a tuple

You can change these values to any dimensions. Try it!

The next line creates a window:

screen = pygame.display.set\_mode(size)

**screen** is a variable to represent the screen. We could have used any variable name, like x or y, but **screen** is a more meaningful name. Anytime we want to change the screen in some way, we can access it through this variable.

**pygame** is the main module. Inside the pygame module is a class called **display**. The display class has a bunch of functions having to do with the display screen. The function we are using is called **set\_mode**(). The set\_mode() function sets the resolution of the screen. We send it one argument – the size of the screen we want (in this case, 320 by 240).

**Choosing a Colour**

A console is always black and white, but a GUI can have colour. Colour codes are in RGB code (Red-Green-Blue), which uses values from 0 to 255 for each colour. In the space before the game loop begins, insert these lines:

white = 255,255,255 # Red-Green-Blue colour code

screen.fill(white)

Since we chose the highest value of each colour, we get white. Try messing with these numbers to see the effect on the screen.

**The Game Loop**

This is the main loop of our program. This loop allows us to constantly monitor the mouse, the keyboard, and update the screen. We will insert a game loop after we create the white screen.

What does the Game Loop do? It...

* continuously checks for any event (like a mouse click, keyboard click etc.)
* draws the display over and over (we can’t tell, but it is redrawing very fast).

Note: Make sure you are saving your program as you go.

**Adding Pictures to your Window**

A picture is called a **bitmap**. It is called this because it is actually a map of bits each representing a coloured pixel to place on the “map”.

Go to the “Bitmaps” folder in this folder (for unit 6 on Google Drive). Find “ball.bmp” and download it to the same folder where you saved your program.

Now add the next bit of code (in blue below) into the spot before the game loop:

screen = pygame.display.set\_mode(size)

screen.fill(white)

ball = pygame.image.load("ball.bmp") #load the bitmap

ballX = 0 # set the coordinates

ballY = 0

screen.blit(ball, (ballX, ballY)) # draw the ball

gameOn = True

while gameOn:

etc.

The screen.blit() function is one we will use often – it is how we draw a bitmap on our screen.

If the ball does not show up on your screen, ask your teacher for help.

Exercise

Try messing with your program in the following ways:

* change the size of the screen (how big is the monitor?)
* change the colour of your screen background
* place the ball at a different location (change ballX or ballY values)