Lab #3

TU850/1 - Creative Coding

*Your working lab has to be shown to (and signed off by) the lab demonstrator between 12 noon and 2pm on Monday 10th of February, in CQ-233/CQ-236.*

**QUESTION 1**

**Instructions:**

Take the following code and run it:

|  |  |
| --- | --- |
| **A green turtle with a black background  AI-generated content may be incorrect.** | **Python with Turtle** |
| import turtle import math  # Screen setup screen = turtle.Screen() screen.bgcolor("pink") screen.tracer(0)  # Create turtle objects def create\_turtle(shape, color, size, pos):  t = turtle.Turtle()  t.shape(shape)  t.color(color)  t.shapesize(\*size)  t.penup()  t.goto(pos)  return t  # Eye properties eye\_radius = 50 pupil\_radius = 15 eye\_distance = 80  # Left eye left\_eye = create\_turtle("circle", "white", (5, 5, 1), (-eye\_distance, 0)) left\_pupil = create\_turtle("circle", "blue", (1.5, 1.5, 1), (-eye\_distance, 0))  # Right eye right\_eye = create\_turtle("circle", "white", (5, 5, 1), (eye\_distance, 0)) right\_pupil = create\_turtle("circle", "blue", (1.5, 1.5, 1), (eye\_distance, 0))  # Function to move pupils def move\_pupils(x, y):  for eye, pupil in [(left\_eye, left\_pupil), (right\_eye, right\_pupil)]:  ex, ey = eye.pos()  dx, dy = x - ex, y - ey  dist = math.sqrt(dx\*\*2 + dy\*\*2)  if dist > eye\_radius - pupil\_radius:  dx = dx / dist \* (eye\_radius - pupil\_radius)  dy = dy / dist \* (eye\_radius - pupil\_radius)  pupil.goto(ex + dx, ey + dy)  screen.update()  # Bind mouse movement screen.listen() screen.onscreenclick(move\_pupils, 1) # Click to test movement screen.ontimer(lambda: screen.update(), 10)  screen.mainloop() | |

Click on all four corners of the screen that is generated and screengrab each one:

A black and white square with cursors

AI-generated content may be incorrect.

**Submission:**

Paste all four screengrabs onto your Template document.

**QUESTION 2**

**Instructions:**

Change the eyecolour of both eyes from blue to brown, and screengrab that result.

**Submission:**

Paste a screengrab and the new code onto your Template document.

**QUESTION 3**

**Instructions:**

Run the following code:

|  |  |
| --- | --- |
| **A green turtle with a black background  AI-generated content may be incorrect.** | **Python with Turtle** |
| import turtle import math import random import time  # Set up the screen screen = turtle.Screen() screen.bgcolor("black") screen.setup(width=1.0, height=1.0) # Fullscreen mode turtle.colormode(255) # Enable RGB colors  # Create a turtle for waves t = turtle.Turtle() t.speed(16) # Fast drawing t.width(2) t.hideturtle()  # Turtle for Pollock-style splots splot\_turtle = turtle.Turtle() splot\_turtle.hideturtle() splot\_turtle.speed(0)  # Fast screen updates for smooth rendering turtle.tracer(6, 1)  # Define color sequence wave\_colors = [  (255, 0, 0), # Red  (255, 165, 0), # Orange  (255, 255, 0), # Yellow  (0, 255, 0), # Green  (148, 0, 211), # Violet  (0, 255, 0), # Green  (255, 255, 0), # Yellow  (255, 165, 0), # Orange  (255, 0, 0), # Red  (255, 165, 0), # Orange  (255, 255, 0), # Yellow  (0, 255, 0), # Green  (148, 0, 211), # Violet  (0, 255, 0), # Green  (255, 255, 0), # Yellow  (255, 165, 0) # Orange ]   # Function to draw Pollock-style paint splots def draw\_paint\_splot(x, y, size, color, fade\_steps=1):  size = int(size) # Ensure size is an integer  splot\_turtle.color(color)  splot\_turtle.width(2)  splot\_turtle.penup()  splot\_turtle.goto(x, y)   # Create a chaotic splatter effect with lines  for \_ in range(random.randint(8, 15)):  angle = random.randint(0, 360)  length = random.randint(size // 2, size) # Ensure size is an integer  width = random.randint(2, 6)   splot\_turtle.width(width)  splot\_turtle.setheading(angle)   splot\_turtle.pendown()  splot\_turtle.forward(length)  splot\_turtle.penup()  splot\_turtle.goto(x, y)   # Add small random dots to mimic paint drips  for \_ in range(random.randint(10, 20)):  dot\_x = x + random.randint(-size // 2, size // 2)  dot\_y = y + random.randint(-size // 2, size // 2)  dot\_size = random.randint(2, 6)   splot\_turtle.goto(dot\_x, dot\_y)  splot\_turtle.dot(dot\_size)   # Function to fade in a white splot def fade\_in\_splot(x, y, max\_size):  for step in range(1, 6): # 5 steps to fade in  draw\_paint\_splot(x, y, int(max\_size \* (step / 5)), "white") # Convert to int  time.sleep(0.2) # Delay between fade-in steps  screen.update()   # Function to draw densely packed sine waves across the full screen def draw\_fullscreen\_dense\_sine\_waves(wave\_count=180, amplitude=15, frequency=7):  screen\_width = screen.window\_width()  screen\_height = screen.window\_height()   x\_start = -screen\_width // 2  y\_start = screen\_height // 2   gap = screen\_height / wave\_count   black\_splot\_positions = []   for i in range(wave\_count):  # Select the color based on the sequence (each color lasts for 5 waves)  t.pencolor(wave\_colors[(i // 5) % len(wave\_colors)])   t.penup()  t.goto(x\_start, y\_start - (i \* gap))  t.pendown()   for j in range(screen\_width):  y\_offset = amplitude \* math.sin(math.radians(j \* frequency))  t.goto(x\_start + j, y\_start - (i \* gap) + y\_offset)   # Add a black splat every 10 waves  if i % 10 == 0:  black\_x = random.randint(-screen\_width // 2, screen\_width // 2)  black\_y = y\_start - (i \* gap) + random.randint(-20, 20)  draw\_paint\_splot(black\_x, black\_y, 80, "black")  black\_splot\_positions.append((black\_x, black\_y, i))   # After 5 waves, place a white splat at a random position with fade-in  for bx, by, wave\_index in black\_splot\_positions:  if i == wave\_index + 5:  white\_x = random.randint(-screen\_width // 2, screen\_width // 2)  white\_y = y\_start - (i \* gap) + random.randint(-30, 30)  fade\_in\_splot(white\_x, white\_y, 160)   # Draw full-screen sine waves with Pollock-style splots and fading white splots  draw\_fullscreen\_dense\_sine\_waves(wave\_count=180, amplitude=15, frequency=7)  # Keep the window open turtle.done() | |

WARNING: This should take at least a minute to fully draw, and you’ll need to use the complier installed on the machine you are using rather than one of the online compliers.

You should get something like this:

|  |
| --- |
| A rainbow chevron pattern with fireworks  AI-generated content may be incorrect. |

Do a screengrab of your result.

Please take your code and put it into ChatGPT and ask it to add a new feature into the code, whatever you like, it could be to add a new random shape, or a different colour splot at the end, or some user interactive feature. Explain the new feature in 50-100 words.

* Take the following code and add …:

**Submission:**

Submit the two screengrabs, as well as the code and the explaination, and paste it all onto your Template document.

|  |
| --- |
| Search in Brightspace for the following module: “**Creative Coding CMPU1042: 2024-25**” and please enroll. |
| e-mail me a completed solution to each of the above programs in your Template document. The Template document should be renamed as follows:   * Surname\_Firstname\_Student#\_\_Lab3.pdf * for example: Smith\_John\_D1234567\_Lab3.pdf   Send it to [Damian.X.Gordon@tudublin.ie](mailto:Damian.X.Gordon@tudublin.ie) with subject heading “DT850 CC Lab #3”, and put it in Brightspace as well. |