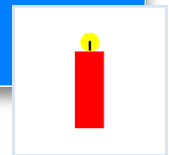
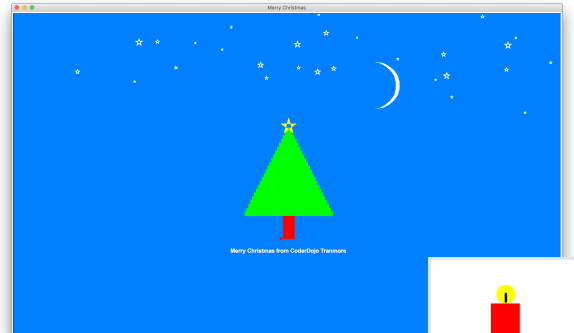
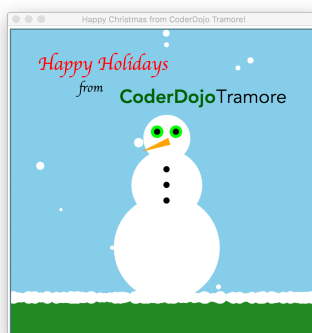
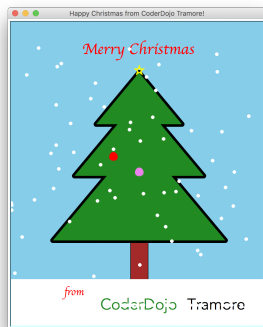


Happy Holidays

Introduction

Today we are going to use our hard working turtles to build some holiday e-cards. You can build anything you wish, for example, the following are based on examples I found online



I will go over the steps to build the first e-card and the rest is up to you!

First to help things along, we are going to change the way we use turtle graphics. Instead of starting our programmes with the code

```
1 import turtle
2
3 screen = turtle.Screen()
4 bob = turtle.Turtle()
5
6 # start drawing
```

we are going to use a small library called `my_library` as follows

```
8 from my_library import *
9
10 # start drawing
```

This library has functions to:

- ① **Draw a ball**
`draw_ball(x, y, radius, color, fill_color)`
- ② **Draw a rectangle**
`draw_rectangle(x, y, width, height, color, fill_color)`
- ③ **Draw a triangle**
`draw_triangle(x, y, base, height, color, fill_color)`
- ④ **Draw a star**
`draw_star(x, y, size, color, fill_color)`

CoderDojo, Tramore, Waterford. (kmurphy@wit.ie)

This work is licensed under a Creative Commons "Attribution-NonCommercial-ShareAlike 3.0 Unported" license.





1 Before we start — let's test our library

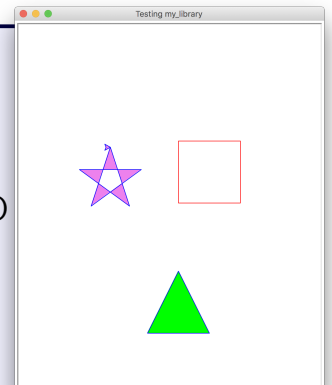


Create file called `test_my_library.py` and insert the following code.

```

1 from my_library import *
2
3 w = 500
4 h = 600
5 setup(w, h, "Happy Christmas from CoderDojo Tramore!")
6
7 draw_rectangle(10,10,100,100, 'red')
8 draw_triangle(10,-100, 100, 100, 'blue', 'green')
9
10 draw_star(-100, 100, 100, 'blue', 'violet')

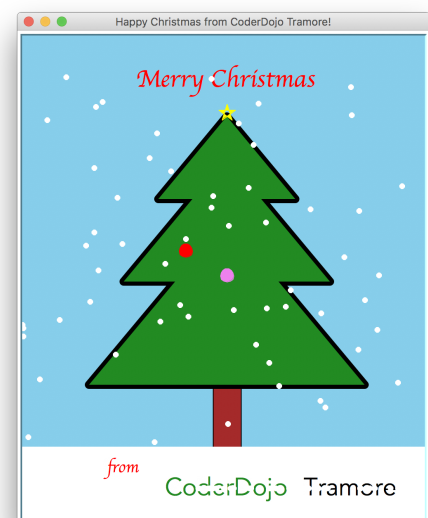
```



2 Our First Card — A Tree with Decorations

Let's start by creating the tree with decorations example I showed you earlier. We will need to

- ① **Setup the screen (window)**
- ② **Draw the tree**
I used three triangles and one rectangle for the trunk.
- ③ **Draw the sky and ground**
I changed the background colour and use a rectangle for the ground.
- ④ **Draw the decorations**
I used balls, but only two so it is a bit sad.
- ⑤ **Write our message**
- ⑥ **Generate moving snow**
I used a new turtle for each snowflake.



Note

I decided to make all distances relative to the width and height of the card. This make the drawing commands a little more complicated but I think you all can handle it. Please ask me to explain anything you don't understand. **Remember, never just type code you don't understand — I could be just stealing your credit card number!**

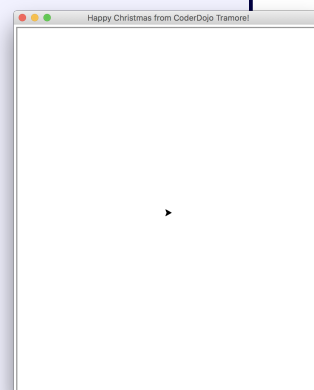


2.1 Setup the screen

 Create file called `tree_with_decorations.py` and insert the following code.

`tree_with_decorations`

```
1 from my_library import *
2
3 w = 500
4 h = 600
5 setup(w, h, "Happy Christmas from CoderDojo Tramore!")
6
7 # ### my data ###
8
9 # ### my functions ###
10
11 # ### my drawings ###
12
13 # draw background
14
15 # draw trunk
16
17 # draw tree
18
19 # draw ground
20
21 # draw star
22
23 # draw decorations
24
25 # draw message – Merry Christmas
26
27 # draw snow falling (for a short while)
28
29 # draw message – from
30
31 # draw snow falling (for a short while)
32
33 # draw message – CoderDojo
34
35 # draw snow falling (for a short while)
36
37 # draw message – Tramore
38
39 # draw snow falling ... forever ....
```



Note:

- This is a relatively complex scene so all of the comments in the above file are placeholders to help use keep our code organised — always a good idea. All of the drawing will be in section `### my drawings ###` and any data and functions that I need to create my drawings will be in `### my data ###` and `### my functions ###` respectively.




2.2 Draw the tree

Drawing the tree trunk is easy — it is just a rectangle.

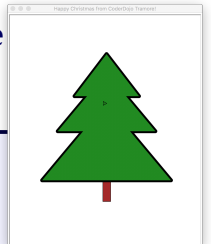
 Insert the following line into `tree_with_decorations.py` at the appropriate section.

```
tree_with_decorations
17 # draw trunk
18 draw_rectangle(-0.02*w, -0.3*h, 0.04*w, 0.5*h, 'black', 'brown')
```

To get a nice looking tree I draw it twice. First drawing the outline using a thick pen in black, then a second time only drawing the inside.

 Insert the following line into `tree_with_decorations.py` at the appropriate

```
tree_with_decorations
20 # draw tree
21 bob.pensize(10)
22 draw_triangle(0, 0.33*h, 0.33*w, 0.18*h, 'black')
23 draw_triangle(0, 0.25*h, 0.50*w, 0.25*h, 'black')
24 draw_triangle(0, 0.12*h, 0.66*w, 0.33*h, 'black')
25 draw_triangle(0, 0.33*h, 0.33*w, 0.18*h, None, 'forest green')
26 draw_triangle(0, 0.25*h, 0.50*w, 0.25*h, None, 'forest green')
27 draw_triangle(0, 0.12*h, 0.66*w, 0.33*h, None, 'forest green')
```




2.3 Draw the sky and ground

 Insert the following line into `tree_with_decorations.py` at the appropriate section.

```
tree_with_decorations
13 # draw background
14 screen.bgcolor("sky blue")
15 #bob.hideturtle()
```



Notice the commented line. We will uncomment this later, it is easier to see if something goes wrong if we can see the turtle moving.

 The ground is just a rectangle — As a test of your understanding of the code so far, you will have to come up with the line to draw the ground yourself!

2.4 Draw the star

 You draw the star using `draw_star` comment — let's see if you can place that yourself!



2.5 Draw the decorations

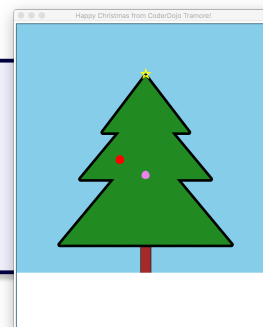
The decorations are a number of balls placed at various points on the tree. We could pick random positions but it usually generates clumps of decorations and we want them to be more spread out¹ So we will manually set the position and colour of each ball. In the following code I have created two decorations — you should try to add more yourself.



Insert the following line into `tree_with_decorations.py` at the appropriate section.

```

36 # draw decorations
37 decorations = [ (-0.1, 0.05, 'red'), (0, 0, 'violet') ]
38
39 for x,y,c in decorations:
40     draw_ball(w*x,h*y, 8,c,c)
  
```



Add more decorations!

2.6 Write our message(s)

To write a message we:

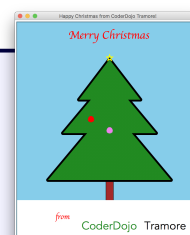
- Move our hard working turtle, `bob`, to the desired position.
- Set the colour we want for the message.
- Pick the font family, size, and weight (bold, normal).
- Tell `bob` what to write.



Insert the following line into `tree_with_decorations.py` at the appropriate section.

```

42 # draw message - Merry Christmas
43 jump(0, 0.4*h)
44 bob.color("red")
45 font = ("Apple Chancery", 30, "bold")
46 bob.write("Merry Christmas", font=font, align="center")
  
```



Insert the code at the appropriate sections needed to generate all of the text in the above figure.

¹This is a common issues with generating random data — it often has bigger clumps than we would like. The solution to this is to use quasi-random which is distinct from pseudo-random.



2.7 Generate moving snow

OK, to get snowflakes falling, we are going to create A LOT OF TURTLES, each turtle will represent a snowflake. This might appear crazy, but it works if:

- We change how the turtles appear by changing their shape to a 'circle', changing their colour to white, making them smaller.
- We point the turtles downwards so that when they move forward they look like there are falling.
- We keep a list of all turtles so when they get to the bottom of the screen we ... 'kill them off'.



We need to keep a list of snowflakes so add the following code segments **at the appropriate sections**.

tree_with_decorations

```
7 # ### my data ###
8 snowflakes = []
```

Next we add our functions for creating and move the snowflakes

tree_with_decorations

```
10 # ### my functions ###
11
12 def snow_fall():
13     rate_of_snow_flakes = 3
14     make_new_snowflake = (random.randint(0, rate_of_snow_flakes) == 0)
15     if make_new_snowflake:
16         # create new snowflake and set properties
17         snowflake = turtle.Turtle()
18         snowflake.shape('circle')
19         snowflake.color('white')
20         snowflake.shapesize(.3)
21         snowflake.right(90)
22         snowflake.penup()
23         x = random.randint(-0.5*w, 0.5*w)
24         y = 0.5*h
25         snowflake.setposition(x,y)
26
27         # add this snowflake to my list
28         snowflakes.append(snowflake)
29
30     for snowflake in snowflakes:
31         move_snowflake(snowflake)
32         if snowflake.ycor() <= -0.5*h:
33             snowflake.clear()
34             snowflakes.remove(snowflake)
35         del snowflake
36     screen.update()
37
38 def move_snowflake(snowflake):
39     snowflake.forward(2)
```



Finally to see snow falling we need to insert the following code AFTER our 'Merry Christmas' message code

tree_with_decorations

```
80 # draw snow falling (for a short while)
81 for k in range(50):
82     snow_fall()
```



Insert the above code after each message and at the end of your program insert the next bit of code (this parts will run forever).

tree_with_decorations

```
108 # draw snow falling ... forever ....
109 while true:
110     snow_fall()
```

2.8 Exercises

We're done ... but ...

- What about having random size snowflakes?
- What about adding a wind that causes the snowflakes to fall to the side?
- What about making the decorations sparkle?
- What about ... you get the idea!