

5.3

Red Alert Project



My Code Review Edition

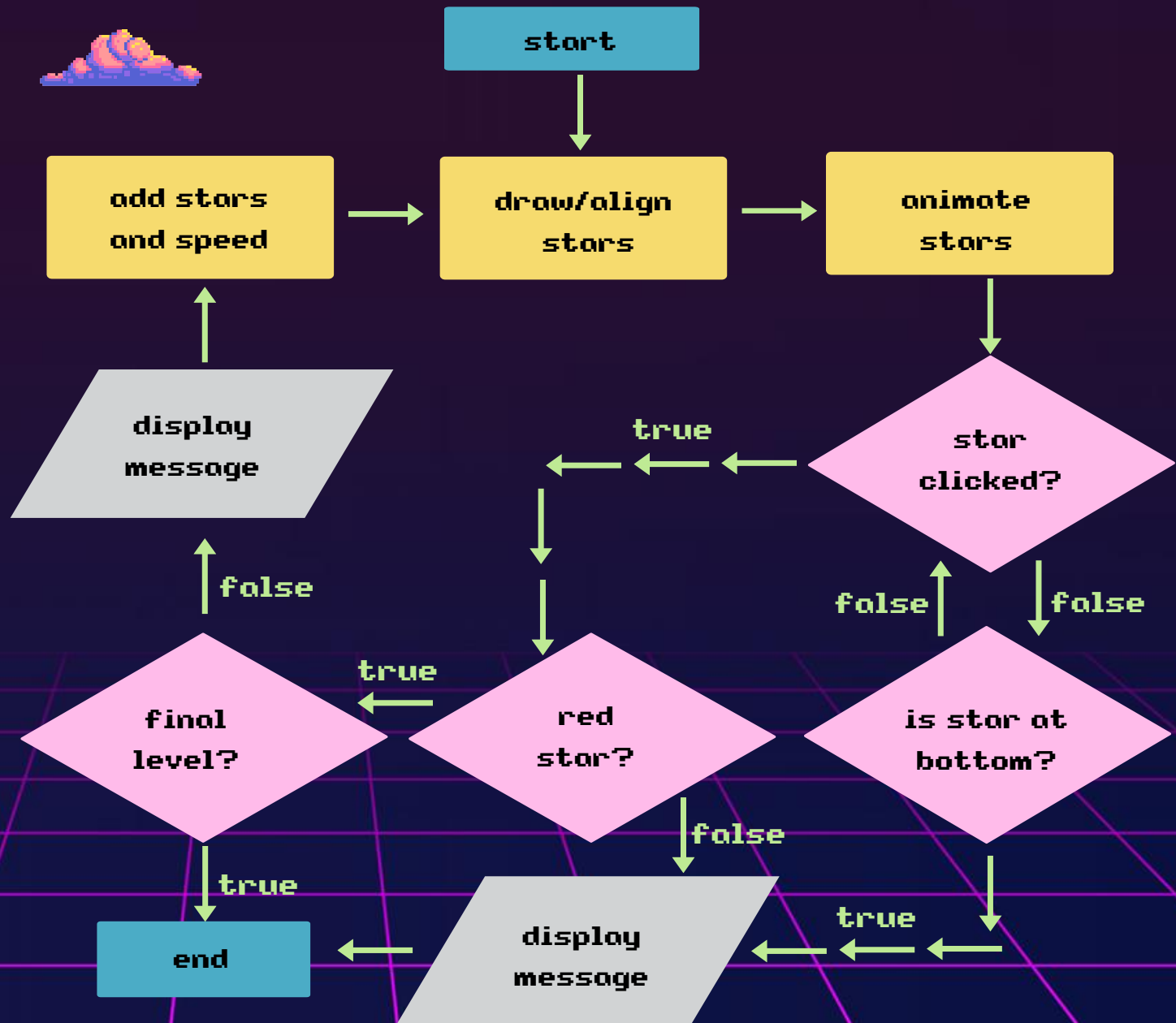


What is a code review?

- A **code review** is an exercise programmers do to check the quality of a piece of code.
- We conduct code reviews to **identify bugs**, **enhance the quality of existing code**, and consequently help all programmers code well.
- You will be guided to do a code review on your work.
- **What can you learn better today from your own mistakes?**
- The goal is **not** to get your code to work perfectly like the teacher's!
- The goal is to **learn and be a better version of yourself**.

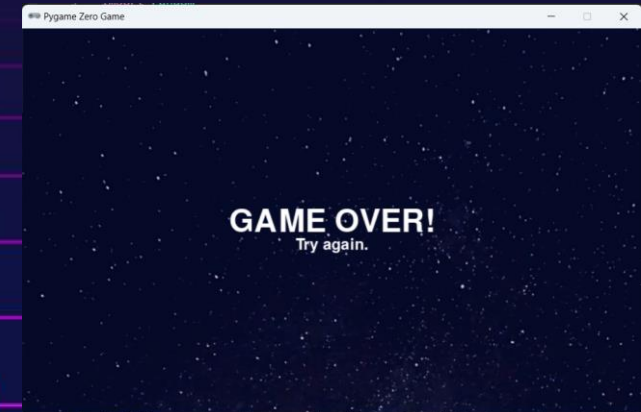
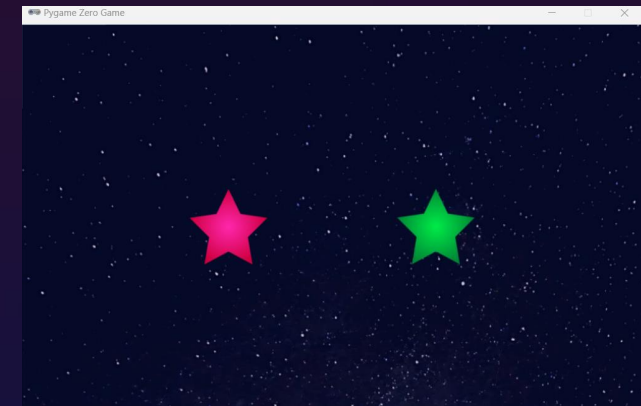


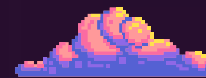
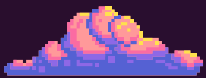
Flowchart Recop



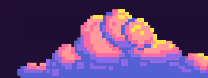
Over to you from here

- When you run the sample, the screen loads Level 1 and animates the stars until they reach the bottom ending the game.
- Complete the **on_mouse_down()** function to react to the player's mouse clicks.
- Write code for the **click_red_star()** function that stops the animations and updates the current level if the game is not complete.
- Write code for the **stop_animations()** function which accepts the list of stars as a parameter. This is a simple function that stops the animation of each star.





Look through a magnifying glass

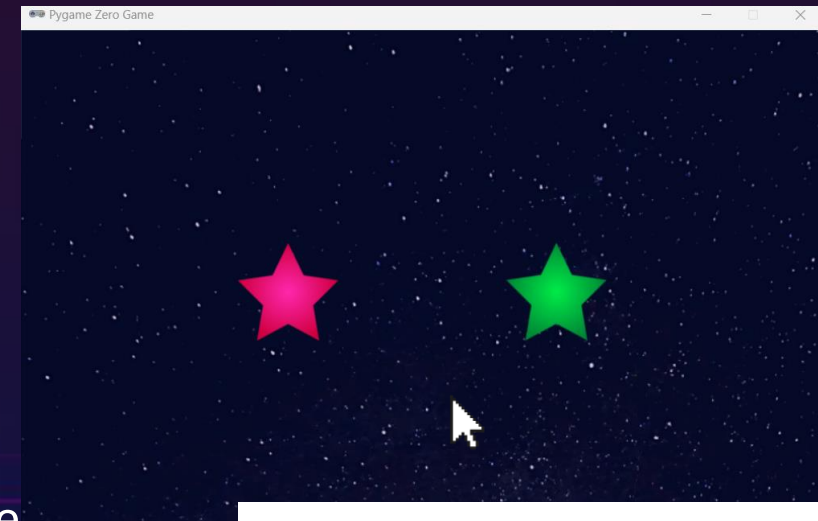


- We have given you instructions on how to complete the game.
- You have three tasks to complete in this exercise.
- Read the instructions carefully and implement the TODO exercises in the sample code as best as you can.
- You may always refer to code we wrote in class and search a little bit on the Internet.



5.3a Handle Mouse Clicks

- You were meant to write code for the `on_mouse_down()` function.
- This is **invoked automatically** every time the player clicks on the screen.
- Pygame Zero passes the click **position** as a parameter.
- You had to check that the click position **collided** with the **red star**.
- This task was hard because Actors are complex variables.



```
1  def on_mouse_down(pos):
2      if alien.collidepoint(pos):
3          print("Eek!")
4      else:
5          print("You missed me!")
```

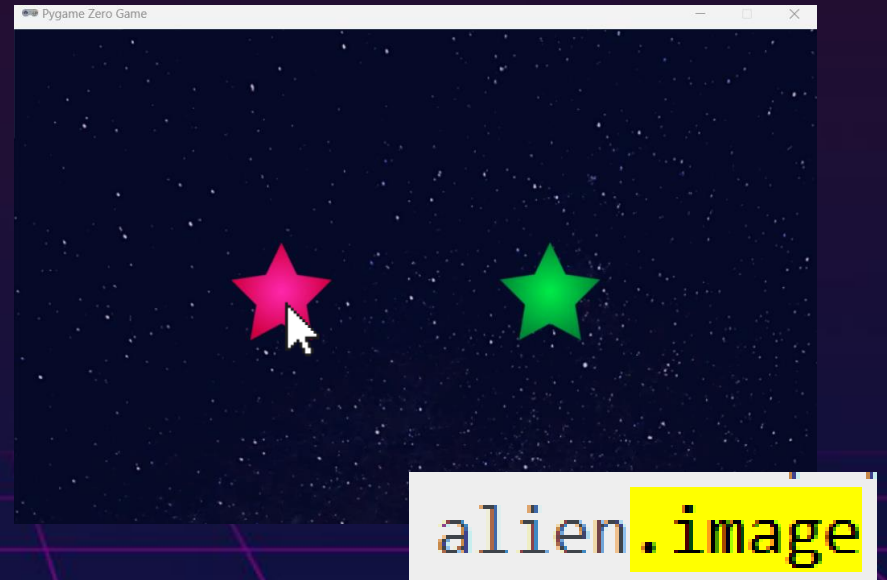

5.3a Handle Mouse Clicks

- Look at your code and compare it to your teacher's code snippet below which is circled in red.
- Which pieces are the same? Which pieces are different?
- Do you understand your code better? Do you understand her code better?
- Do your comments make code easier to read? Do her comments make code easier to read?
- If you are inspired to change your work, you may do so but – **only if you understand**.

```
# go through the list of stars and for every star Actor
for star in stars:
    # use collidepoint() function to check whether the player clicked on one of the stars
    if star.collidepoint(pos):
        if star.image == "red_star": # if they did, check whether the star image is red_star
            on_red_star_click()
        else:
            handle_game_over()
```

5.3a Handle Mouse Clicks

- Not only did you need to check that the click position was on a star – you also needed to check that the star was **red**.
- When we created our star Actors in previous parts we used an **image**.
- We could read back the value of the image we used easily by using the **image variable**.
- When the player clicked on the red star then you needed to call a function **on_red_star_click()**.
- Otherwise, you needed to call a function **handle_game_over()**.



5.3a Handle Mouse Clicks

- Look at your code and compare it to your teacher's code snippet below which is circled in red.
- Which pieces are the same? Which pieces are different?
- Do you understand your code better? Do you understand her code better?
- Do your comments make code easier to read? Do her comments make code easier to read?
- If you are inspired to change your work, you may do so but – **only if you understand**.

```
# go through the list of stars and for every star Actor
for star in stars:
    # use collidepoint() function to check whether the player clicked on one of the stars
    if star.collidepoint(pos):
        if star.image == "red_star": # if they did, check whether the star image is red_star
            on_red_star_click()
        else:
            handle_game_over()
```



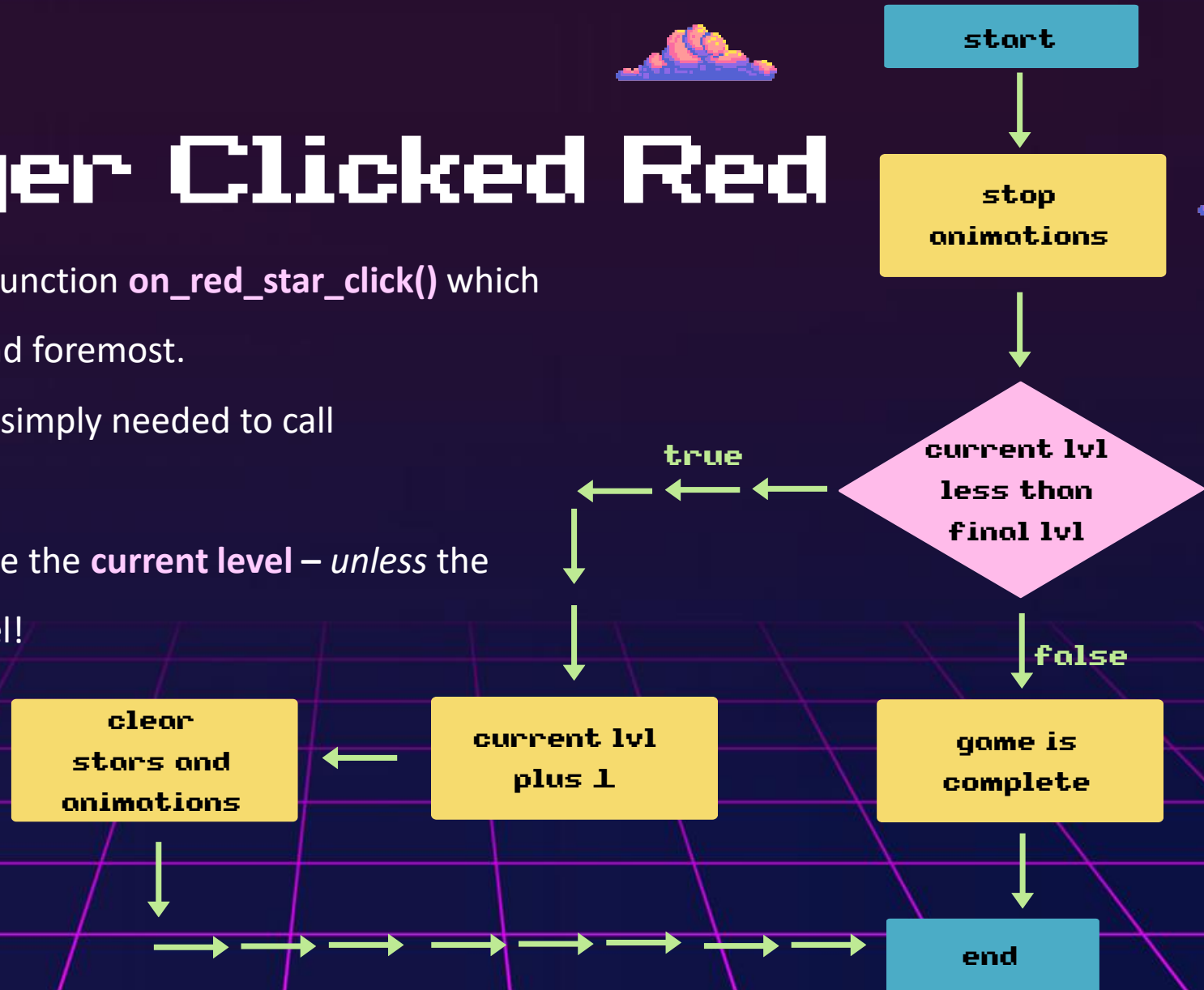
Did you make a change?

Even if your change is very small always
run your code to test and see if there are
new errors **in the place you changed.**



5.3b Player Clicked Red

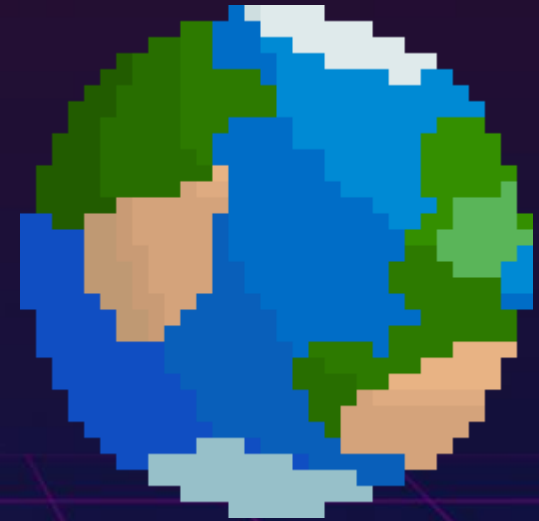
- You were asked to define a function `on_red_star_click()` which stops the animations first and foremost.
- To stop the animations, you simply needed to call `stop_animations()` function.
- After, you needed to increase the **current level** – *unless* the player reached the final level!



5.3b Player Clicked Red


- We also explained that this function needs access to **global variables**.
- We also told you that the **first line of code inside the function** should be:

```
global current_level, stars, animations, game_complete.
```



5.3b Player Clicked Red

- Look at your code and compare it to your teacher's code snippet below which is circled in red.
- Which pieces are the same? Which pieces are different?
- Do you understand your code better? Do you understand her code better?
- Do your comments make code easier to read? Do her comments make code easier to read?
- If you are inspired to change your work, you may do so but – **only if you understand**.



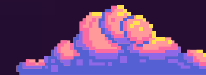
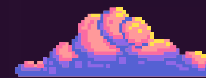
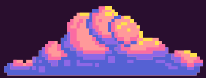
```
# a function on_red_star_click()
def on_red_star_click():
    global current_level, stars, animations, game_complete
    stop_animations()
    if current_level < FINAL_LEVEL:
        # the player moves up to another level, unless
        current_level += 1
        stars.clear()
        animations.clear()
    else:
        # player reached final level
        game_complete = True
```


Three pixel art clouds are positioned at the top of the slide. The first cloud is on the left, the second is in the center, and the third is on the right. They are rendered in a pixelated style with shades of blue, purple, and yellow.

Did you make a change?

Even if your change is very small always
run your code to test and see if there are
new errors **in the place you changed.**





5.3c Stop Animations

- You needed to define a function called **stop_animations()**.
- Like Actors, Animations are also complex data structures.
- Animations in Pygame Zero have a Boolean variable **running** which stores whether the animation is being played. We gave the hint which is on the right.
- If the animation is being played, then you should make it stop by using the **stop()** function.
- **Many students need to practice writing functions properly.**

```
# get info about animation
is_played = my_animation.running




# stop animation
my_animation.stop()
```





5.3c Stop Animations

- Look at your code and compare it to your teacher's code snippet below which is circled in red.
- Which pieces are the same? Which pieces are different?
- Do you understand your code better? Do you understand her code better?
- Do your comments make code easier to read? Do her comments make code easier to read?
- If you are inspired to change your work, you may do so but – **only if you understand**.



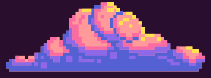
```
# a function stop_animations()
def stop_animations():
    global animations
    # loop through animations[]
    for animation in animations:
        if animation.running == True:
            animation.stop()
```



Did you make a change?

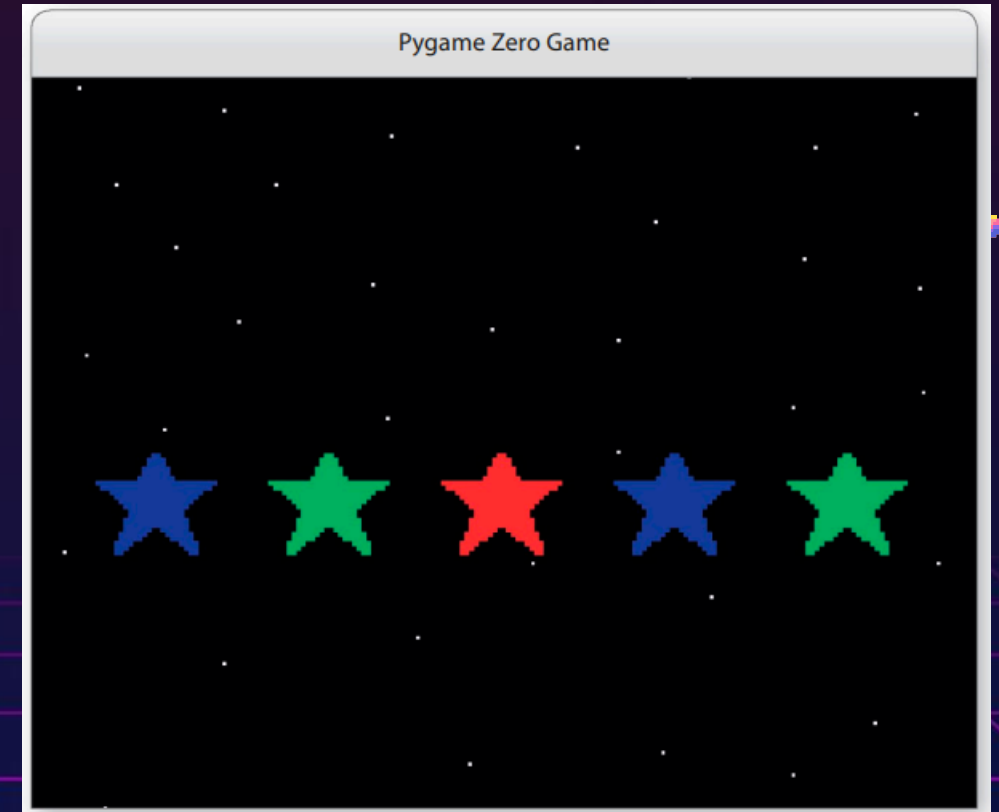
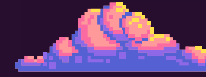
Even if your change is very small always
run your code to test and see if there are
new errors **in the place you changed.**





That's it!

- If you have completed the code review and made the right fixes, then you should be able to play the game.
- Run your code and try to play the game to see **if it works better than the last time.**
- The most important part of this exercise is **that you learned from your mistakes.**





You have leveled up!



Congratulations on completing your first
big project.

