

Counting Active Aliens



In my Space Invaders method where I increase the speed based on how many aliens are dead, I want the function to only count the aliens that are active.

Got it — this is a good idea, and it'll make your speed adjustment more accurate since you only want to count the aliens that are still "alive" (active).

Assuming your aliens are in some sort of list (or group) inside your `Fleet` or similar class, and each alien has an `active` attribute (or maybe something like `alive` or `visible` — whatever you named it), you can filter them like this:

```
python

active_aliens = [alien for alien in self.aliens if alien.active]
active_count = len(active_aliens)
```

Then you can use `active_count` in your speed adjustment logic.

If you're currently doing something like:

```
python

dead_aliens = total_aliens - remaining_aliens
speed = base_speed + (dead_aliens * speed_increment)
```

You just want to update it so `remaining_aliens` counts **only active aliens**:

```
python

remaining_aliens = len([alien for alien in self.aliens if alien.active])
dead_aliens = total_aliens - remaining_aliens
speed = base_speed + (dead_aliens * speed_increment)
```

Alternative (if you already have a method or attribute for active aliens):

If your `Fleet` class already tracks active aliens, you could make a helper method:

```
python

def count_active_aliens(self):
    return sum(1 for alien in self.aliens if alien.active)
```

Then just call:

```
python

remaining_aliens = self.count_active_aliens()
```

If you show me your current method, I can give you the exact rewrite! 🚀 Want to?



No need. `active_aliens = [alien for alien in self.aliens if alien.active]` was more than enough.

Perfect — glad that was all you needed!

Simple and clean. List comprehensions are magic like that ✨

If you run into anything else, just let me know. 🚀



Can pygame play ogg files?

Yes!

Pygame **can** play `.ogg` files — in fact, `.ogg` is one of the formats that Pygame's mixer supports natively, and it's often preferred because it's compressed and smaller than `.wav`, but still keeps good quality.

So you can do things like:

```
python

import pygame

pygame.init()
pygame.mixer.init()

sound = pygame.mixer.Sound("sound.ogg")
sound.play()
```

Or for background music:

```
python

pygame.mixer.music.load("music.ogg")
pygame.mixer.music.play(-1) # -1 loops indefinitely
```

Just make sure your file path is correct, and it should work smoothly!

If you hit any crackling or distortion issues, that's usually down to the mixer's settings (sample rate, buffer size), and I can help you tweak that too. 🚀

Let me know if you want!