

how do i make it so that if i press key once, it runs until i press they key to deactivate it again:

```
if key == 'p' or key == 'P':
    x = self.shooter.get_x() + self.shooter.get_width() / 2 # centres x and y
    y = self.shooter.get_y() + self.shooter.get_height() / 2
    angle = self.shooter.get_angle() + (math.pi / 2) # ensures no rotation sets
angle to 90 degrees
```

```
self.missile_controller.aimline(x, y, angle)
```



Toggle Shield on Key Press (Press Once to Activate, Press Again to Deactivate)

To make the shield **toggle on/off** with a key press (e.g., `P`), modify your code like this:

1. Update `ShieldController` to Support Toggle

python

 Copy  Download

```
class ShieldController:
    def __init__(self, file, player_height):
        self.file = file
        self.player_height = player_height
        self.shield = None # Tracks if shield is active

    def toggle_shield(self, x, y, angle):
        if self.shield is None:
            # Activate shield
            self.shield = Shield(self.file, x, y, angle)
        else:
            # Deactivate shield
            self.shield = None

    def update_position(self, x, y, angle):
        if self.shield:
            # Update shield position to follow player
            self.shield.x = x - math.sin(np.radians(angle)) * self.player_height / 2
            self.shield.y = y - (self.player_height / 2 - math.cos(np.radians(angle)) * self.player_height / 2)
            self.shield.angle = np.radians(angle)

    def draw(self):
```

```
if self.shield:
    self.shield._draw() # Draw only if active
```

2. In Your Key Press Logic (Game Loop)

python

 Copy  Download

```
if key == 'p' or key == 'P':
    x = self.shooter.get_x() + self.shooter.get_width() / 2
    y = self.shooter.get_y() + self.shooter.get_height() / 2
    angle = self.shooter.get_angle() + (math.pi / 2) # Adjust angle as needed

    # Toggle shield on/off
    self.shield_controller.toggle_shield(x, y, angle)
```

3. In Your Game Loop (Update & Draw Continuously)

python

 Copy  Download

```
# Update shield position (if active)
self.shield_controller.update_position(
    self.shooter.get_x() + self.shooter.get_width() / 2,
    self.shooter.get_y() + self.shooter.get_height() / 2,
    self.shooter.get_angle() + (math.pi / 2)
)

# Draw shield (if active)
self.shield_controller.draw()
```

Key Features

- ✓ **Press P once** → Shield activates.
- ✓ **Press P again** → Shield deactivates.
- ✓ **Shield follows player** automatically.
- ✓ **No timers needed**—fully manual control.

Alternative: Global Toggle Variable (Simpler)

If you prefer to avoid modifying `ShieldController`, use a **global toggle variable**:

python

 Copy
  Download

```
# At the top of your game script:
shield_active = False
shield = None

# In key press logic:
if key == 'p' or key == 'P':
    shield_active = not shield_active # Toggle True/False
    if shield_active:
        x = self.shooter.get_x() + self.shooter.get_width() / 2
        y = self.shooter.get_y() + self.shooter.get_height() / 2
        angle =
```

no sorry i am now working wiht something called the aimline under the missile controller



Toggle Aim Line on Key Press (Press `P` to Activate/Deactivate)

Since you're now working with the `missile_controller.aimline()`, here's how to make it **toggle on/off** with the `P` key:

1. Add a Toggle State to Your Game Class

In your main game class (where you handle inputs), add a variable to track the aim line state:

python

 Copy
  Download

```
class YourGameClass:
    def __init__(self):
        self.aimline_active = False # Tracks if aim line is visible
```

2. Modify Key Press Logic to Toggle Aim Line

Update your key detection logic to toggle `aimline_active` and call `aimline()` only when active:

python

 Copy  Download

```

if key == 'p' or key == 'P':
    self.aimline_active = not self.aimline_active # Toggle state

if self.aimline_active:
    x = self.shooter.get_x() + self.shooter.get_width() / 2
    y = self.shooter.get_y() + self.shooter.get_height() / 2
    angle = self.shooter.get_angle() + (math.pi / 2)
    self.missile_controller.aimline(x, y, angle) # Activate aim line
# No 'else' needed-aimline() will stop when not called

```

3. Update Aim Line Position Continuously (Game Loop)

In your game loop, **only update the aim line if it's active**:

python

 Copy  Download

```

if self.aimline_active:
    x = self.shooter.get_x() + self.shooter.get_width() / 2
    y = self.shooter.get_y() + self.shooter.get_height() / 2
    angle = self.shooter.get_angle() + (math.pi / 2)
    self.missile_controller.aimline(x, y, angle) # Keeps aim line updated

```

Key Behavior

- **Press P once** → Aim line appears and follows the shooter.
- **Press P again** → Aim line disappears.
- **No timers**—fully manual toggle.

Alternative: Let `MissileController` Handle Toggle

If you prefer to encapsulate the logic inside `MissileController`:

python

 Copy  Download

```

class MissileController:
    def __init__(self):
        self.aimline_active = False

    def toggle_aimline(self, x, y, angle):

```

```
self.aimline_active = not self.aimline_active
if self.aimline_active:
    self.aimline(x, y, angle)

def update_aimline(self, x, y, angle):
    if self.aimline_active:
        self.aimline(x, y, angle)
```

Usage:

python

 Copy  Download

```
# On key press:
if key == 'p' or key == 'P':
    x = self.shooter.get_x() + self.shooter.get_width() / 2
    y = self.shooter.get_y() + self.shooter.get_height() / 2
    angle = self.shooter.get_angle() + (math.pi / 2)
    self.missile_controller.toggle_aimline(x, y, angle)

# In game loop:
if self.missile_controller.aimline_active:
    x = self.shooter.get_x() + self.shooter.get_width() / 2
    y = self.shooter.get_y() + self.shooter.get_height() / 2
    angle = self.shooter.get_angle() + (math.pi / 2)
    self.missile_controller.update_aimline(x, y, angle)
```

Why This Works

- **Toggle state** (`aimline_active`) ensures the aim line stays on/off until `P` is pressed again.
- **Position updates** only happen when active (efficient).
- **No changes to** `aimline()` **method**—works as-is.

Let me know if you'd like further adjustments! 

