Prompt:

Objective: Write a Python script for a ball bouncing inside of a rotating square.

Context: This is to test your ability to code not only in python, but to simulate the physics of a ball bouncing.

Instructions: Use as many python libraries as needed. The Ball should never leave the square. It should be able to collide with any and all sides of the square. The Ball should bounce off at the correct angle that it approached from.

Constraints: No Constraints

Desired Output: A 2D rotating square, that drops a ball from the middle of the square at the beginning of the codes execution. The Ball will fall and bounce of the square as close as possible to reality. The Code should run continuously until the user keyboard interrupts the program.

Perplexity:

20250307-0444-32.6
023303.mp4

Certainly! I'll create a Python script that simulates a ball bouncing inside a rotating square using the Pygame library for visualization and basic physics calculations. This script will meet all the requirements you've specified.

Here's the Python script:

[Python Script”
This script does the following:

1. Sets up a Pygame window with a black background.
2. Creates a white square that rotates slowly.
3. Initializes a red ball in the center of the square.
4. Simulates gravity and bouncing physics for the ball.
5. Checks for collisions between the ball and the sides of the rotating square.
6. Updates the ball's position and velocity based on collisions and gravity.
7. Continuously renders the scene until the user closes the window.

To run this script, you'll need to have Pygame installed. You can install it using pip:

text

```
pip install pygame
```

Then, save the script to a file (e.g., bouncing_ball.py) and run it using Python:

text

python bouncing_ball.py

The simulation will continue running until you close the window or interrupt the program with a keyboard interrupt (Ctrl+C).

This implementation provides a realistic simulation of a ball bouncing inside a rotating square, with proper collision detection and response, gravity, and elasticity. The ball will never leave the square and will bounce off at the correct angle based on its approach.

The result seemed accurate at the beginning of the animation, but the moment it involved sliding and not bouncing the ball fell out of the square.

CoPilot:

20250307-0447-36.8103996.mp4

Alright, let's tackle this challenge! We'll create a 2D simulation with a ball bouncing inside a rotating square using the Pygame library. The Pygame library is excellent for creating 2D games and simulations in Python. I'll write the script step-by-step:

1. Importing required libraries
2. Setting up the Pygame environment and screen
3. Defining the ball and square properties
4. Implementing the rotation and movement logic
5. Simulating the physics for the bouncing ball

Here's the code:

[CODE]

This script simulates a ball bouncing inside a rotating square. The ball is affected by gravity and will bounce off the sides of the square. The code runs continuously until the user interrupts the program with a keyboard input.

Feel free to tweak the properties and constants to achieve the desired behavior! Let me know if you have any questions or need further assistance.

The Square was way off it seemed to get the coordinates wrong having the square rotate around a axis not inside of the square, and when the ball started in the center and fell straight down never was inside the square.

ChatGPT 4o:

20250307-0452-43.2991831.mp4

Here's a Python script that simulates a ball bouncing inside a rotating square. It uses pygame for visualization and numpy for vector calculations to ensure realistic physics. The ball follows realistic collision physics, preserving its velocity components upon impact.
[CODE]
This script creates a realistic simulation where the ball bounces off the rotating square while obeying collision physics. The square continuously rotates, and the ball interacts dynamically with its edges, reflecting appropriately.

Let me know if you need any modifications or explanations! 🚀

This is by far the best one, it did import Numpy which none of the others did which could have given it an advantage but was in the prompt. The ball bounced around correctly. The physics weren’t perfect, but the ball never did anything magical.