Algorithm

1. Initialization:

- Set up constants such as window dimensions, colors, speeds, and frequencies.

- Import necessary libraries (`pygame`, `random`, `SimpleGE`).

2. Define Game Classes:

- `Tornado`, `Player`, and `Obstacle` classes:

- `Tornado`: Represents the tornado object. It rotates continuously.

- `Player`: Represents the player-controlled object. It can move horizontally and vertically.

- `Obstacle`: Represents the obstacles that move downward from the top of the screen.

3. Main Game Class (`TornadoGame`):

- Initialize the game window with the specified title, width, height, and background color.

- Create instances of the tornado, player, and initialize an empty list for obstacles.

- Initialize the score.

4. Game Loop:

- Continuously update the game state and render the graphics until the game ends.

- Update the positions and states of the tornado, player, and obstacles.

- Check for collisions between the player and obstacles.

- Update the score based on the player's performance.

- Render the updated game state on the screen.

5. Spawn Obstacles:

- Randomly generate obstacles at the top of the screen at predefined intervals.

6. Move Obstacles:

- Move each obstacle downward at a constant speed.

7. Check Collisions:

- Check if the player collides with any obstacles. If a collision occurs, end the game.

8. Update Score:

- Increase the score each time the player successfully avoids an obstacle.

9. Render:

- Draw the tornado, player, obstacles, and score on the game window.

10. End Game:

- If the player collides with an obstacle, end the game and display "Game Over!".

This algorithm outlines the steps involved in running the tornado game and provides a structured approach to understanding its functionality and flow.