E:\0NEU\CS5100\Mohammed\drone_env_limited.py

```python
import numpy as np
import random

class DroneCoverageEnvAdaptiveLimited:
    """
    Center-based squares environment.

    * training mode => spawns are random (we handle that in step if we see 'SPAWN_RANDOM')
    * test_mode => same approach, but the code is effectively the same
    """

    def __init__(self, config):
        self.N = config["N"]
        self.M = config["M"]
        self.available_sizes = config["available_sizes"]
        self.max_drones = config["max_drones"]

        self.obstacle_percent = config["obstacle_percent"]

        # We'll read coverage_multiplier from config
        self.coverage_multiplier = config.get("coverage_multiplier", 5.0)

        self.alpha = config["alpha_env"]
        self.beta = config["beta_env"]
        self.gamma_penalty = config["gamma_penalty_env"]
        self.stall_threshold = config["stall_threshold_env"]
        self.max_steps = config["max_steps_env"]

        self.test_mode = config.get("test_mode", False)

        self.done = False
        self.obstacles = set()
        self.num_free_cells = self.N * self.M
        self.drones = []

        self.previous_coverage = 0
        self.stall_counter = 0
        self.steps_taken = 0

    def reset(self):
        self.done = False
        self._generate_obstacles()
        self.drones = []
        self.previous_coverage = 0
        self.stall_counter = 0
        self.steps_taken = 0
        return self._get_observation()

```

```
    def _generate_obstacles(self):
        total_cells = self.N * self.M
        num_obs = int(self.obstacle_percent * total_cells)
        all_cells = [(x, y) for x in range(self.N) for y in range(self.M)]
        chosen = random.sample(all_cells, num_obs)
        self.obstacles = set(chosen)
        self.num_free_cells = total_cells - num_obs

    def _get_observation(self):
        drone_list = []
        for d in self.drones:
            drone_list.append((d["cx"], d["cy"], d["size"], d["active"]))
        return {
            "drones": drone_list,
            "obstacles": list(self.obstacles)
        }

    def step(self, action):
        """
        We have:
          - action["type"] == "SPAWN_RANDOM" => spawn the drone randomly
          - action["type"] == "ACT" => "REMOVE" or "STAY"
          - action["type"] == "NOOP"
        """
        if action["type"] == "SPAWN_RANDOM":
            self._spawn_random_drone(action.get("size", 3))

        elif action["type"] == "ACT":
            idx = action.get("drone_index", -1)
            mv = action.get("move", None)
            self._act_on_drone(idx, mv)
        # else NOOP => do nothing

        coverage_count, overlap_count = self._compute_coverage_and_overlap()

        coverage_fraction = coverage_count / float(self.num_free_cells) if
self.num_free_cells>0 else 1.0
        overlap_fraction  = overlap_count / float(self.num_free_cells) if
self.num_free_cells>0 else 0.0
        uncovered_fraction= 1.0 - coverage_fraction
        num_active = sum(d["active"] for d in self.drones)

        # reward formula with bigger coverage multiplier
        reward = (
            self.coverage_multiplier*coverage_fraction
            - self.alpha*overlap_fraction
            - self.beta*uncovered_fraction
            - self.gamma_penalty*num_active
        )

```

```
        # check terminal
        if coverage_count >= self.num_free_cells:
            self.done = True

        if coverage_count > self.previous_coverage:
            self.previous_coverage = coverage_count
            self.stall_counter = 0
        else:
            self.stall_counter += 1
            if self.stall_counter >= self.stall_threshold:
                self.done = True

        self.steps_taken += 1
        if self.steps_taken >= self.max_steps:
            self.done = True

        return self._get_observation(), reward, self.done, {}

    def _spawn_random_drone(self, size):
        # If at max drones, skip
        if len(self.drones) >= self.max_drones:
            return
        if size not in self.available_sizes:
            size = random.choice(self.available_sizes)

        # pick random free cell?
        # or just random cell ignoring obstacles
        rx = random.randint(0, self.N - 1)
        ry = random.randint(0, self.M - 1)

        self.drones.append({"cx": rx, "cy": ry, "size": size, "active": True})

    def _act_on_drone(self, idx, move):
        if idx < 0 or idx >= len(self.drones):
            return
        d = self.drones[idx]
        if move == "REMOVE":
            self.drones.pop(idx)
            return
        if move == "STAY":
            # do nothing, remain active
            return

        # Should never get here if we only have "REMOVE"/"STAY"
        return

    def _compute_coverage_and_overlap(self):
        cover_count = {}
        for d in self.drones:
            if not d["active"]:
```

```
                continue
            cx, cy, s = d["cx"], d["cy"], d["size"]
            half = (s-1)//2
            for dx in range(-half, half+1):
                for dy in range(-half, half+1):
                    gx = cx+dx
                    gy = cy+dy
                    if 0 <= gx < self.N and 0 <= gy < self.M:
                        if (gx,gy) not in self.obstacles:
                            cover_count[(gx,gy)] = cover_count.get((gx,gy), 0) + 1

        coverage_count = sum(1 for v in cover_count.values() if v >= 1)
        overlap_count  = sum(1 for v in cover_count.values() if v >= 2)
        return coverage_count, overlap_count
```