新版奖励函数拟定思路

为了解决这个问题，我们需要重新定义“生存模式”的目标。对于速度处于劣势的猎物来说，生存的关键不在于跑得远，而在于**跑得巧**。它需要学会利用**障碍物**作为掩体，来阻断追捕者的路线，为自己争取时间和空间。

因此，我们的新思路是：

1. **保持分层结构**：`DANGER_ZONE` 的概念依然有效，它帮助智能体在不同危险等级下切换核心目标。
2. **重新定义“生存模式”的奖励**:
   - **旧逻辑 (错误)**: 奖励与追捕者的直线距离。
   - **新逻辑 (正确)**: 奖励那些能够**有效利用障碍物进行规避**的行为。当猎物处于危险中时，如果它能跑到一个障碍物的后面，让“追捕者-障碍物-猎物”形成一条直线，那么它就成功地为自己制造了掩护。
3. **如何量化“利用障碍物”**:
   一个简单而有效的方法是：在生存模式下，我们奖励猎物**靠近那个“最具战略价值”的障碍物**。哪个障碍物最具战略价值？就是那个能最有效阻挡追捕者的障碍物。
   我们可以这样定义它：
   - 计算追捕者到每个障碍物的距离。
   - 计算猎物到每个障碍物的距离。
   - 一个好的战略位置是：猎物靠近某个障碍物，同时追捕者离这个障碍物很远。
   - 因此，我们可以将奖励设置为 `(追捕者到障碍物的距离) / (猎物到障碍物的距离)` 的函数。这个比值越大，说明这个障碍物作为掩体越安全。
4. **简化实现**:
   为了避免过于复杂的计算，我们可以采用一个更直观的简化版本：在“生存模式”下，智能体的主要奖励不再是远离追捕者，而是**缩短与“最佳掩体”障碍物之间的距离**。同时，我们仍然保留一个较小的惩罚项，以防它完全不顾追捕者。

新一版的 `def agent_reward(self, agent, world):`

请**用下面这个全新的 `agent_reward` 函数，完整地替换**掉你 `simple_tag.py` 文件中现有的同名函数。`reward` 和 `adversary_reward` 函数保持不变。

```python
# 文件位置: multiagent/scenarios/simple_tag.py

def agent_reward(self, agent, world):
    prey = agent # 当前 agent 就是猎物
    predator = self.adversaries(world)[0]

    # --- 1. 定义权重和超参数 ---
    W_REACH_SHELTER = 300.0
    W_CAUGHT = -300.0
    W_HIT_OBSTACLE = -150.0
    DANGER_ZONE = 0.4  # 危险区域阈值，可以调整

    # “生存模式”权重
    W_SHIELDING = 7.0  # 奖励利用障碍物进行躲避的行为

    # “寻路模式”权重
    W_SHELTER_SEEKING = 3.0
    W_PREDATOR_AVOIDANCE = -1.5

    W_OBSTACLE_AVOIDANCE = -2.5
    W_TIME_PENALTY = -0.1

    rew = 0.0

    # --- 2. 计算距离 ---
    dist_to_predator = np.linalg.norm(prey.state.p_pos - predator.state.p_pos)
    shelters = world.check
    if shelters:
        dist_to_closest_shelter = min([np.linalg.norm(prey.state.p_pos - s.state.p_pos) for
    else:
        dist_to_closest_shelter = float('inf')

    obstacles = world.landmarks
    if obstacles:
        dists_to_obstacles = [np.linalg.norm(prey.state.p_pos - o.state.p_pos) for o in obst
        dist_to_closest_obstacle = min(dists_to_obstacles)
    else:
        dist_to_closest_obstacle = float('inf')

    # --- 3. 计算事件奖励 ---
    if shelters and dist_to_closest_shelter < prey.size + shelters[0].size:
        rew += W_REACH_SHELTER
    if dist_to_predator < prey.size + predator.size:
```

```
            rew += W_CAUGHT
        if obstacles and dist_to_closest_obstacle < prey.size + obstacles[0].size:
            rew += W_HIT_OBSTACLE

        # --- 4. 分层计算过程奖励 ---
        if dist_to_predator < DANGER_ZONE:
            # **【新】生存模式**: 追捕者太近，首要任务是利用障碍物躲避
            if obstacles:
                best_shield_reward = -float('inf')
                # 遍历所有障碍物，找到最佳的掩体
                for obs in obstacles:
                    # 追捕者到障碍物的距离
                    dist_pred_to_obs = np.linalg.norm(predator.state.p_pos - obs.state.p_pos)
                    # 猎物到障碍物的距离
                    dist_prey_to_obs = np.linalg.norm(prey.state.p_pos - obs.state.p_pos)

                    # 好的掩体：追捕者离它远，而猎物离它近
                    # 我们用 (dist_pred_to_obs / (dist_prey_to_obs + 1e-6)) 这个比值来衡量掩体的好
                    shield_reward = dist_pred_to_obs / (dist_prey_to_obs + 1e-6)
                    if shield_reward > best_shield_reward:
                        best_shield_reward = shield_reward

                # 给予利用最佳掩体的奖励
                rew += W_SHIELDING * best_shield_reward
        else:
            # **寻路模式**: 相对安全，首要任务是去庇护所
            if shelters:
                rew += W_SHELTER_SEEKING * (1 / (dist_to_closest_shelter + 0.1))
            rew += W_PREDATOR_AVOIDANCE * (1 / (dist_to_predator + 0.1))

        # 对障碍物的通用惩罚
        if obstacles and dist_to_closest_obstacle < 0.2:
            rew += W_OBSTACLE_AVOIDANCE * (1 / (dist_to_closest_obstacle**2 + 0.1))

        # --- 5. 通用时间惩罚 ---
        rew += W_TIME_PENALTY

        return rew
```

新版逻辑解读

1. **分层逻辑不变**: `if dist_to_predator < DANGER_ZONE:` 依然是模式切换的开关。

2. **“生存模式”彻底重写**:
    - 不再奖励单纯的跑远。
    - 而是遍历所有障碍物，为每一个障碍物计算一个“掩护质量分”。这个分数的核心思想是：如果追捕者离这个障碍物很远，而猎物离这个障碍物很近，那么这个障碍物就是一个好掩体，分数就高。
    - 智能体会得到一个与“最佳掩护质量分”成正比的奖励 (`W_SHIELDING * best_shield_reward`)。
    - 这会激励猎物在被追赶时，主动跑向那些能有效阻隔追捕者路线的障碍物后面，从而学会**智能规避**，而不是**无脑逃跑**。
3. **“寻路模式”保持不变**: 在安全时，主要目标依然是前往庇护所。

这个新版本的奖励函数更符合博弈论和实际的追逃策略，应该能显著改善你的智能体在被追赶时的表现。