**Epic E1 Simulation Sandbox Ready (Week 1–2)**

| **#** | **Story** | **Acceptance / DoD** | **Implementation Checklist** |
| --- | --- | --- | --- |
| **E1-1** | *As a researcher I want to build and launch the stock pybullet drones sim so that I can confirm my tool-chain works.* | conda env list shows **drones**; python -m gym\_pybullet\_drones.examples.pid runs without error. | bash\nconda create -n drones python=3.10\nconda activate drones\npip install -e gym-pybullet-drones/ # local clone\npip install stable-baselines3[extra] gymnasium torch\n Installation recipe follows official docs ([github.com](https://github.com/utiasDSL/gym-pybullet-drones)) |
| **E1-2** | *I can spawn two Crazyflie-2X quads in a head-to-head arena and step the Gym env from Python.* | A 200-step loop prints non-NaN obs and rew. | python\nenv = MultiHoverAviary(num\_drones=2, obs='kin', act='vel', gui=True) # API uses enums internally\nobs, \_ = env.reset()\nfor \_ in range(200):\n obs, rew, term, trunc, \_ = env.step(np.zeros(env.action\_space.shape))\n Defaults from learn.py example ([raw.githubusercontent.com](https://raw.githubusercontent.com/utiasDSL/gym-pybullet-drones/main/gym_pybullet_drones/examples/learn.py)) |
| **E1-3** | *I have a new custom env DogfightAviary that inherits MultiRLAviary.* | pytest tests/test\_dogfight\_env.py passes space-shape assertions. | *Create* uav-intent-rl/envs/DogfightAviary.py:python\nclass DogfightAviary(MultiRLAviary):\n EPISODE\_LEN\_SEC = 30\n DEF\_DMG\_RADIUS = 0.3 # m, hit if closer and within FOV\n def \_computeReward(self):\n return self.\_calc\_hits() - 0.01 # shaping\n def \_computeTerminated(self):\n return self.\_blue\_down() or self.\_red\_down()\n Use \_getDroneStateVector() helpers exactly like HoverAviary ([raw.githubusercontent.com](https://raw.githubusercontent.com/utiasDSL/gym-pybullet-drones/main/gym_pybullet_drones/envs/HoverAviary.py)) |
| **E1-4** | *Episode video & CSV logs are saved for post-mortem.* | After env = Monitor(...) a runs/2025-07-xx/ folder contains .mp4 + progress.csv. | Wrap env with gymnasium.wrappers.RecordVideo and RecordEpisodeStatistics; pass render\_mode="rgb\_array" if headless. |

**Epic E2 Scripted Baseline Opponent (Week 2)**

| **Story** | **DoD** | **Tasks** |
| --- | --- | --- |
| *E2-1 Red drone follows a scripted “pursue & fire” policy so that Blue has a stationary adversary.* | 100 episodes, Red hits Blue ≥ 65 % of runs. | 1. Add uav-intent-rl/policies/scripted\_red.py.2. Implement simple proportional controller: target Blue’s XY, maintain z=1 m, shoot when dist<0.3 m.3. Unit-test with deterministic seed. |
| *E2-2 Arena resets with random spawn poses (±π yaw, 2–4 m separation) to avoid over-fitting.* | Histograms of spawn distance show uniform distribution. | Modify DogfightAviary.reset() to randomise initial\_xyzs before calling super().reset(). |

**Epic E3 PPO Baseline (No Opponent Modeling) (Week 3–4)**

| **Story** | **DoD** | **Implementation Notes** |
| --- | --- | --- |
| *E3-1 Blue learns PPO policy against fixed Red.* | Averaged over 10 eval runs, win-rate ≥ 60 % by 3 M steps. | *Config*: <project>/configs/ppo\_nomodel.yaml -> n\_envs: 8, γ: 0.99, lr: 3e-4, clip: 0.2.Use SB3 vectorised env wrapper make\_vec\_env(DogfightAviary).Log with TensorBoard. |
| *E3-2 Best checkpoint exported.* | models/baseline\_no\_model.zip committed and loads without error. | Call model.save(...) after EvalCallback threshold reached (pattern copied from learn.py) ([raw.githubusercontent.com](https://raw.githubusercontent.com/utiasDSL/gym-pybullet-drones/main/gym_pybullet_drones/examples/learn.py)) |

**Epic E4 Self-Play League (Week 5–6)**

| **Story** | **DoD** | **Tasks** |
| --- | --- | --- |
| *E4-1 Convert env to Ray RLlib MultiAgentEnv so both drones have policies.* | rllib rollout script works; printed sample shows two policy ids. | Implement policy\_mapping\_fn that assigns "blue" / "red".Use PPO with shared weights (config["share\_observations"] = True) to speed learning. |
| *E4-2 League Elo table auto-generated weekly.* | CSV artifacts/elo\_matrix.csv produced by cron job; heat-map appears in README. | Store past checkpoints every 0.5 M steps; run round-robin evaluation script that fills matrix. |

**Epic E5**Recurrent PPO with LSTM Opponent Modeling (Week 7 – 8)

| **Story** | **DoD** | **Key Code Touchpoints** |
| --- | --- | --- |
| *E5-1*  As a researcher I want **Blue** to adopt the **AMF architecture** so its policy explicitly conditions on a learned opponent feature vector. | • Network compiles & trains without NaNs.• policy\_forward() returns (action, value, h\_opp) where h\_opp ∈ ℝ^{32}.• One‑hour smoke‑train reaches >0 opponent‑prediction accuracy (>20 %) and positive episode reward. | 1. Subclass nn.Module → AMFPolicy (PyTorch).   • Shared torso encodes observation → latent.   • Opponent head: h\_opp = MLP(latent) → logits → CE loss.   • Fusion: torch.cat([latent, h\_opp]) → actor/critic heads. 2. Wrap in SB3 via CustomCombinedExtractor (if using features extractor API) or custom policy class. 3. Add λ‑weighted CE loss: L\_total = L\_PPO + λ·L\_ce.  Config entry amf\_lambda default 0.5.4. Unit test shapes & forward pass with dummy tensor. |
| *E5-2*   |  |  | | --- | --- | |  | As an engineer I want reproducible training so I can benchmark AMF vs. pure PPO. | | configs/ppo\_amf.yaml committed.• TensorBoard logs both **reward** and **opponent‑acc** curves. | 1. Copy ppo\_nomodel.yaml; add amf\_lambda, policy="AMFPolicy", log\_h\_opp=True.2. Register custom metrics via SB3 callback (on\_rollout\_end).3. Spawn 8 envs, train 3 M steps; save models/blue\_amf.zip when EvalCallback hits ≥ 60 % win‑rate vs. scripted Red. |

**Epic E6**  AMF‑Style Latent Feature Fusion (Opponent‑Aware PPO) (Week 9 – 10)

| **Story** | **DoD** | **Experiments** |
| --- | --- | --- |
| *E6-1*  As a scientist I want to measure how much AMF beats a **pure PPO baseline**. | Blue‑AMF wins ≥ 70 % of 200 games vs. baseline\_no\_model.zip (95 % CI). | 1. Freeze baseline; evaluate with evaluate.py --blue\_a blue\_amf --blue\_b baseline\_no\_model --n\_games 200.2. Log CSV of results + CI to artifacts/bench\_amf\_vs\_baseline.csv |
| *E6-2*  As a reviewer I need evidence that **latent fusion**, not just the CE loss, drives gains. | • Ablation model with detach(h\_opp) sees ≥ 10 pp drop in win‑rate. | Train control run (--detach\_fusion=True) for 3 M steps.2. Evaluate vs. baseline; record win‑rate.3. Plot bar chart in notebooks/ablation\_amf.ipynb. |
| As a hyper‑param tuner I want to know the best λ. | • Grid search λ ∈ {0.1,0.3,0.5,1.0} identifies λ\* with highest eval win‑rate. | Use Optuna loop calling train.py --config configs/ppo\_amf.yaml --amf\_lambda {λ}.2. Save study to runs/optuna\_amf.db; auto‑plot optimisation history. |

**Epic E7 Behaviour Visualisation & Analysis (Week 11)**

| **Story** | **DoD** | **Assets** |
| --- | --- | --- |
| *E7-1 Trajectory plots highlight anticipatory manoeuvres.* | figs/intent\_vs\_nomodel.svg shows Blue-model flanking earlier than baseline. | Dump JSON trajectory dicts; use Matplotlib to overlay xy paths and scatter shot-events. |
| *E7-2 MP4 demo clip recorded.* | media/dogfight\_intent\_demo.mp4 plays in README. | Use env record=True; trim with ffmpeg. |

**Epic E8 Paper & Repo Packaging (Week 12)**

| **Story** | **DoD** | **Checklist** |
| --- | --- | --- |
| *E8-1 6-page draft with reproducibility checklist ready for arXiv.* | paper/draft.pdf builds via make and cites Panerati et al. 2021. | Include install snippet from project README; cite gym-pybullet-drones IROS-21 paper ([utiasdsl.github.io](https://utiasdsl.github.io/gym-pybullet-drones/)) |
| *E8-2 Public GitHub with one-line training command.* | README.md first code block runs python train.py --config configs/ppo\_intent.yaml. | Push models ≥ 50 MB to Git-LFS. |

**Additional Engineering Tips**

* **Coding pattern:** keep env-specific constants (hit-radius, ammo) in envs/config.py, import into env and reward-calc code to avoid magic numbers.
* **Data management:** each training run writes under runs/YYYY-MM-DD\_HH-MM-SS/ (TensorBoard + checkpoints + videos) to keep artefacts tidy.
* **CI:** add a GitHub Action that runs pytest && python smoke\_train.py --steps 200 on every push. The smoke script uses local=True flag in the example ([raw.githubusercontent.com](https://raw.githubusercontent.com/utiasDSL/gym-pybullet-drones/main/gym_pybullet_drones/examples/learn.py)) to keep the job under 5 minutes.
* **Hyper-param tuning:** once pipeline stabilises, integrate Optuna via SB3’s HyperOptCallback to search λ, lr, clip-range.
* **Sample efficiency:** if training speed drags, bump n\_envs to 32 with make\_vec\_env("shared\_memory") (PyBullet is CPU-bound but scales well across cores).