ChatGPT

Step 6 – Pianificazione delle iterazioni e del monitoraggio

Analisi degli strumenti esistenti

Nel repository Evo-Tactics sono già presenti strumenti per tracciare lo stato di avanzamento e aggiornare automaticamente la documentazione:

- **Tracker giornaliero:** il workflow `daily-tracker-refresh.yml` esegue ogni giorno un job che lancia lo script `scripts/daily_tracker_refresh.py` per aggiornare un indice e le barre di progresso nel README [1]. Il workflow committa direttamente le modifiche in `main`.
- **Registro dei tracker:** il file `config/tracker_registry.yaml` descrive le tabelle da mostrare nel `docs/00-INDEX.md` e i progress indicator da visualizzare nel README. Ogni elemento della sezione `progress_sections` specifica il metodo di calcolo (checkbox o manuale), il file sorgente e i link di riferimento [2].
- **Checklist e log:** molte checklist operative e processuali (es. `docs/checklist/milestones.md`, `docs/checklist/project-setup-todo.md`) utilizzano caselle di spunta che possono essere conteggiate dal tracker. Esistono inoltre log periodici (es. `logs/traits_tracking.md`, `logs/qa/latest-dashboard-metrics.json`) che registrano metriche e risultati di audit.
- **Roadmap e milestone:** la roadmap operativa definisce slot settimanali, checklist di revisione e milestone con scadenze. Questi documenti forniscono un quadro temporale per organizzare gli sprint.

Proposta di pianificazione iterativa

1. **Definizione degli sprint**
2. Adottare sprint di 1–2 settimane. Ogni sprint deve avere obiettivi chiari (es. completare una feature del dataset, implementare una parte di HUD, sviluppare test Playwright) e deliverable misurabili.
3. Utilizzare le colonne `Backlog → In Progress → Review QA → Done` della board GitHub per spostare le issue man mano che avanzano (vedi Step 3).
4. Definire le milestone su GitHub (ad esempio `Sprint 2025-W46`) e associare le issue relative allo sprint.
5. **Aggiornamento del tracker e dell'indice**
6. Mantenere aggiornato il file `config/tracker_registry.yaml` aggiungendo o rimuovendo entry quando cambiano i documenti di riferimento. Assicurarsi che ogni area rilevante (dataset, pipeline web, log & metriche, appendici) disponga di un indicatore di avanzamento.
7. Verificare che i file associati ai progress indicator utilizzino caselle di spunta `[ ]` e `[x]` coerenti. Il job `daily-tracker-refresh.yml` provvederà a contare le checkbox e ad aggiornare il README [1]. Per indicatori qualitativi (ad es. "Canvas e integrazioni"), impostare manualmente un valore fra 0 e 1 nel registro [3].

8. **Riunioni di pianificazione e retrospettiva**

9. **Planning meeting** a inizio sprint: definire gli obiettivi, riesaminare il backlog e aggiornare la tracker registry.
10. **Daily sync**: breve aggiornamento su progressi e blocchi; eventuali modifiche alle issue o ai log.

11. **Sprint review & retro**: a fine sprint, rivedere i deliverable e aggiornare le checklist/milestone; discutere ciò che ha funzionato e cosa migliorare. Le note di retro possono essere aggiunte ai log (es. `logs/web_status.md` per la web pipeline).

12. **Monitoraggio continuo delle metriche**

13. **Copertura dei tratti/specie:** monitorare i report generati dalle pipeline (`reports/trait_progress.md`, `data/derived/analysis/trait_coverage_report.json`) per verificare l'avanzamento del dataset. Aggiornare `logs/traits_tracking.md` con osservazioni chiave.
14. **Telemetria e HUD:** usare i log di telemetria e il file `docs/process/telemetry_ingestion_pipeline.md` per tracciare la pipeline di ingestione. I progressi sull'overlay HUD possono essere tracciati in `docs/process/qa_hud.md`.

15. **QA metrics:** i file `logs/qa/latest-dashboard-metrics.json` e `logs/qa/dashboard_metrics.jsonl` registrano snapshot e storici delle metriche; inserire i link di riferimento nel `tracker_registry.yaml` per visualizzare l'andamento.

16. **Integrazione con la CI/CD**

17. Configurare il workflow `daily-tracker-refresh.yml` in modo che, oltre ad aggiornare README e 00-INDEX, esporti un JSON riepilogativo (`--export-json`) che possa essere utilizzato per dashboard esterne.

18. Collegare l'aggiornamento dei tracker con notifiche (es. via Slack o GitHub Discussions) quando l'avanzamento di una sezione scende sotto una soglia o una milestone sta per scadere.

19. **Gestione delle dipendenze e sincronizzazione**

20. Utilizzare i log di sync (`logs/chatgpt_sync.log`) per tenere traccia delle operazioni automatiche, assicurandosi che eventuali aggiornamenti dei dataset o del codice siano discussi durante il daily sync.
21. Mantenere allineata la board con i report dei daily PR (es. `docs/chatgpt_changes/daily-pr-summary-2025-10-30.md`) per evitare duplicazioni.

Prossimi passi

- Rivedere e, se necessario, ampliare il file `config/tracker_registry.yaml` per riflettere tutte le aree del progetto su cui sono in corso lavori.
- Allineare le issue e le milestone correnti con la nuova pianificazione degli sprint.
- Aggiornare i membri del team sulle modalità di compilazione delle checklist e sull'utilizzo del tracker automatico.

Conclusioni

Organizzando il lavoro in sprint brevi, utilizzando il tracker automatico e aggiornando regolarmente le checklist, il team potrà monitorare in maniera trasparente il progresso su tutte le aree (dataset, pipeline web, telemetria, appendici). La combinazione di board GitHub, log e progress indicator renderà più facile individuare i blocchi, pianificare le priorità e comunicare gli avanzamenti a tutte le parti coinvolte.

---

[1] daily-tracker-refresh.yml
https://github.com/MasterDD-L34D/Game/blob/HEAD/.github/workflows/daily-tracker-refresh.yml

[2] [3] tracker_registry.yaml
https://github.com/MasterDD-L34D/Game/blob/HEAD/config/tracker_registry.yaml