Treball fet per:

Alejandro Blesa Berrocal i Pablo Vicente Roura

INFORME – Tres en ratlla amb tauler abstracte

---

1. Introducció

Aquest projecte implementa un **tauler abstracte** per al joc del tres en ratlla amb dues fases: **col·locació** i **moviment**. El sistema està pensat per ser reutilitzable amb diferents *drivers* (tant de consola com gràfics).

Aquesta pràctica s'ha realitzat en equip seguint un **estil funcional amb closures (`nonlocal`)**, tal com s'indica a l'enunciat de l'assignatura.

---

2. Descripció del joc

El joc es juga en un **tauler de 3×3** amb **4 pedres per jugador**.

- **Fase 1:** els jugadors col·loquen alternativament les seves pedres en caselles buides.
- **Fase 2:** quan totes les pedres estan col·locades, els jugadors poden **moure una de les seves pedres** a una casella buida.

Hem escollit la variant **estàndard**, en la qual **guanya el jugador que aconsegueix fer tres en ratlla**. La variant contrària (*misère*) no s'ha implementat perquè només requeriria canviar el missatge de final de partida (de "has guanyat" a "has perdut"), ja que la lògica de detecció és la mateixa.

El joc està dissenyat per poder **canviar la mida del tauler i el nombre de pedres** mitjançant constants, i funciona tant en **mode text** com en **mode gràfic (Pygame)**.

Hi han dos modes de joc, "misery" perd el que fa el tres en ratlla i "default", guanya qui fa el tres en ratlla, si el mode de joc no correspon a cap dels dos, es jugarà en mode default.

---

3. Arquitectura del programa

El sistema està format per dos components principals:

- `abs_board.py`, que implementa el tauler abstracte i tota la lògica del joc.
- **Drivers**, que gestionen la interacció amb l'usuari (consola i Pygame).

El tauler abstracte no mostra informació directament, sinó que ofereix funcions (`stones`, `select_st`,`move_st`, `draw_txt`) que el driver utilitza per interactuar amb el joc.

Hem utilitzat **GitHub** com a eina de control de versions i treball col·laboratiu. Això ens ha permès treballar des de diferents ordinadors, repartir-nos les tasques i finalment unir el treball mitjançant **merges**, obtenint una única versió final del projecte.

---

4. Funcionalitats implementades

Funcionalitat	Responsable
Representació del tauler 3×3	Alejandro
Fase de col·locació de pedres	Alejandro
Fase de moviment de pedres	Pablo
Canvi de torns	Pablo
Comprovació de moviments vàlids	Pablo
Detecció de tres en ratlla	Pablo
Dibuix del tauler en mode text	Alejandro
Neteja de consola a cada acció	Pablo
Interfície gràfica amb Pygame	Alejandro

---

5. Verificació i proves

S'han realitzat diverses **partides completes** per comprovar que totes les regles funcionen correctament.
S'ha verificat que:

- No es permeten moviments invàlids.
- Els torns alternen correctament.
- El joc detecta correctament el final de partida.

Abans d'implementar cada funcionalitat, definíem **exemples d'entrades i sortides esperades**. Després comparàvem el comportament real del programa amb aquests resultats previstos, assegurant-nos que totes les combinacions possibles funcionaven correctament.

---

6. Dificultats

Les principals dificultats han estat la gestió de les **dues fases del joc** i el **control de l'estat mitjançant closures**. En particular, controlar el canvi de torns va ser complex, i es va solucionar mitjançant una variable `selected_player` que indica quin jugador té la pedra seleccionada.

Un altre problema important va ser la **coordinació inicial** del treball, ja que al principi cada membre utilitzava estructures de dades diferents. Això es va resoldre mitjançant reunions i refactorització del codi per unificar criteris.

---

7. Treball en equip

El projecte s'ha realitzat en parelles utilitzant **GitHub**.
Abans de començar, vam definir:

- Com representar el tauler
- Com emmagatzemar les pedres
- Quines funcions faria cada membre

Cada setmana fixàvem objectius i al final revisàvem els resultats i fèiem proves conjuntes per assegurar el correcte funcionament.

---

8. Conclusions

Aquest projecte ens ha permès entendre millor la **programació funcional**, l'ús de **closures (`nonlocal`)** i el **treball en equip amb GitHub**. Hem après que una bona planificació i una distribució clara de tasques és

essencial per evitar errors i conflictes quan es programa en equip.