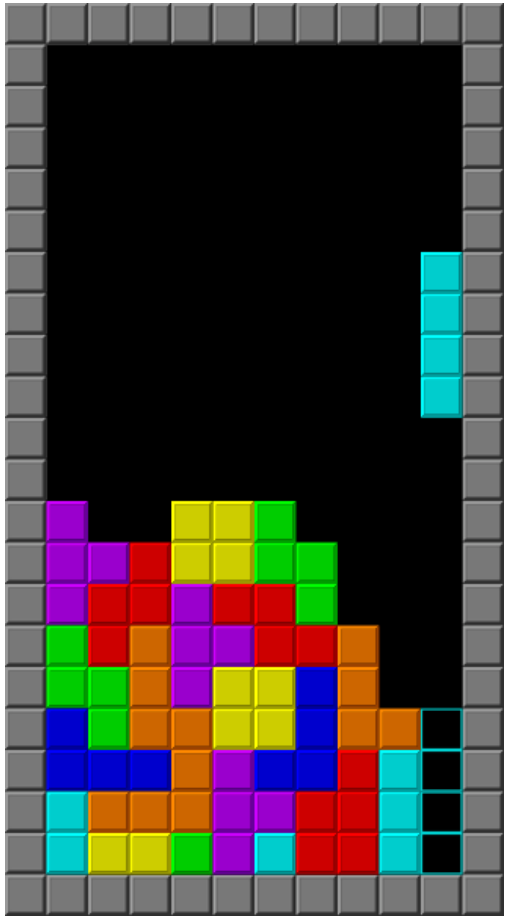


Projecte: Tetris

Segona part

Metodologia de la Programació

Curs 2023 - 2024

Recordem... Objectiu del projecte

Disseny i desenvolupament d'una versió completa del joc del **tetris** original, posant en pràctica els conceptes que anem explicant a les sessions de classe.

El vostre programa final haurà de permetre (com a mínim):

- **Jugar una partida**
  - Generació de les figures que van apareixent.
  - Permetre el moviment de la figura (dreta, esquerra i girs en les dues direccions) mentre va baixant pel tauler de joc.
  - Encaixar la figura amb les altres figures ja col·locades.
  - Eliminar les files completades i fer baixar les que estan per sobre.
  - Actualitzar la puntuació de la partida a mesura que es van eliminant files.
  - Incrementar el nivell (incrementar la velocitat del joc) en funció de la puntuació aconseguida.
  - Comprovar el final de la partida.
- **Visualització gràfica** de la partida i interacció amb el jugador per fer els moviments

Podreu afegir **funcionalitats extres**, si voleu, que comptaran positivament com a **punts addicionals** a la **nota final** del projecte.

Recordem... Planificació del projecte

Primera versió del projecte:

- Mostrar una figura al tauler de joc.
- Moviment de la figura (dreta, esquerra, avall i girs en les dues direccions), comprovant que el moviment és vàlid (no col·lisiona amb altres figures ja col·locades o amb els límits del tauler).
- Encaixar la figura amb les altres figures ja col·locades.
- Eliminar les files completades i fer baixar les que estan per sobre.
- Guardar l'estat actual del tauler en un fitxer.
- En aquesta primer versió treballarem sense visualització gràfica. Tindreu un test d'autoavaluació a Caronte per poder validar el correcte funcionament de les diferents funcionalitats.

Segona versió del projecte:

- Implementar la part gràfica del joc i la interacció del jugador amb el tauler.
- Implementar el desenvolupament complet d'una partida:
  - Generació de les noves figures que van apareixent.
  - Actualitzar la puntuació de la partida a mesura que es van eliminant files.
  - Incrementar el nivell (incrementar la velocitat del joc) en funció de la puntuació aconseguida.
  - Comprovar el final de la partida.
- Possibilitat d'afegir funcionalitats extra.

Segona versió del projecte

Noves funcionalitats

- Implementar la **part gràfica** del joc i la interacció del jugador amb el tauler:
  - Visualitzar gràficament el tauler de joc amb les peces ja col·locades i la figura que va caient.
  - Permetre fer el moviment i el gir de la figura amb el teclat.
  - Fer que la figura vagi caient pas a pas pel tauler fins que queda col·locada.
  - Mostrar com s'eliminen les files i cauen les peces que hi ha per sobre de les files eliminades.
  - Fer aparèixer una nova figura a dalt del tot del tauler quan la figura actual queda col·locada al tauler.
  - Mostrar la puntuació i el nivell actual del joc.
- Implementar el **desenvolupament complet d'una partida**:
  - Actualitzar la puntuació cada cop que es col·loca una figura o s'elimina alguna fila.
  - Canviar de nivell cada cop que la puntuació superi cert llindar. Canviar de nivell implica incrementar la velocitat de caiguda de la figura.
  - Finalitzar la partida si el tauler s'omple fins a dalt de tot.

Segona versió del projecte

Noves funcionalitats

- Implementar un **mode del joc de test**:
  - Inicialitzar l'estat del tauler a partir d'un fitxer de text.
  - L'ordre d'aparició de noves figures al tauler també s'inicialitza a partir d'un fitxer de text.
  - Els moviments de la figura no es fan per teclat sinó que es llegeixen d'un fitxer de text i es van executant de forma automàtica seguint l'ordre del fitxer.
- Mantenir una **llista de les millors puntuacions** de totes les partides que s'han jugat:
  - Inicialitzar la llista de puntuacions a partir d'un fitxer de text quan comença el programa.
  - Quan s'acaba una partida, demanar el nom del jugador i guardar la puntuació final i el nom del jugador a la posició de la llista que correspongui per mantenir l'ordre.
  - Poder visualitzar la llista de les millors puntuacions.
  - Guardar la llista de puntuacions actualitzada al fitxer de text quan acaba el programa.

Segona versió del projecte

Noves funcionalitats

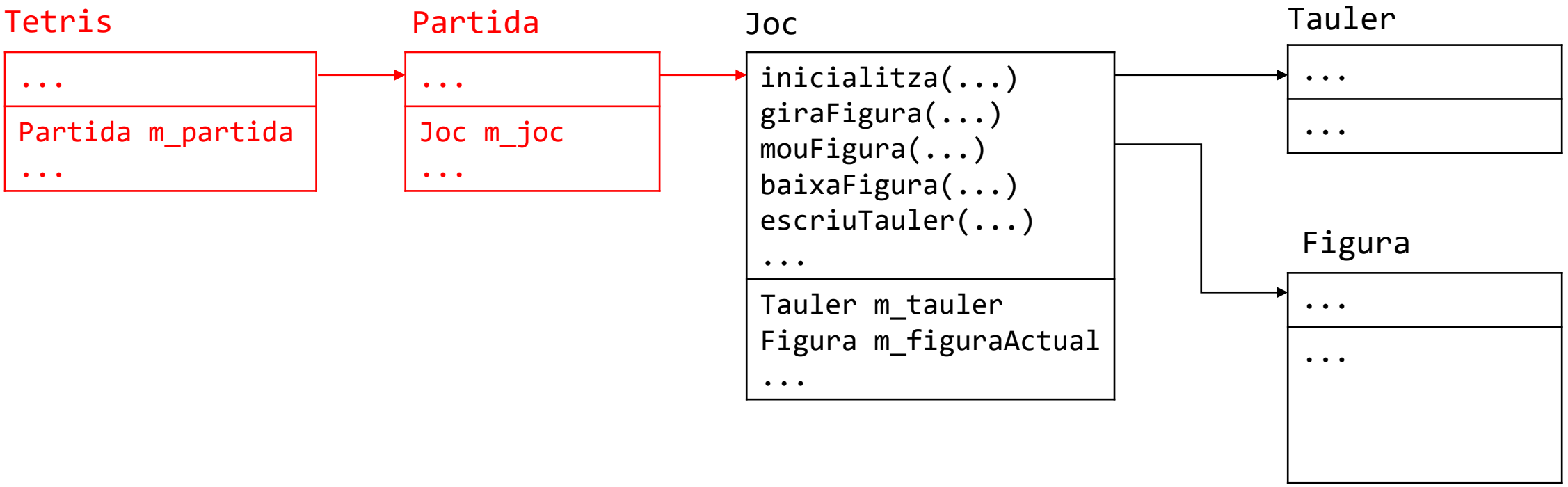
- Implementar un **menú** (en mode no gràfic) per poder escollir entre les diferents opcions del joc:
  - Jugar en mode normal
  - Jugar en mode test
  - Visualitzar llista de millors puntuacions
  - Sortir del programa

- Podeu afegir qualsevol **funcionalitat extra** que vulgueu, que comptaran positivament com a **punts addicionals** a la **nota final** del projecte: nous tipus de caramels o de peces al tauler, diferents objectius, canvis en la visualització, ...

Mireu el vídeo que hi haurà a Caronte mostrant el funcionament del joc

Segona versió del projecte: implementació

Estructura de classes



Noves classes

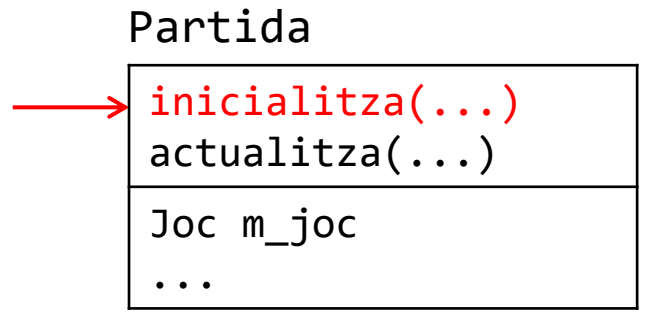
Partida

inicialitza(...) actualitza(...)
Joc m_joc ...

Classe `Partida`

- Guarda tota la **informació** necessària per gestionar la **visualització gràfica i el funcionament** d'una partida:
  - Un **objecte** de la classe **`Joc`** per poder gestionar el funcionament de la partida.
  - La **puntuació actual** del jugador.
  - El **nivell actual** del joc, que determina la velocitat de caiguda de les figures.
  - Si la partida es juga en mode test, la **seqüència d'aparició de les figures** i la **seqüència de moviments** de les figures, que s'hauran llegit dels fitxers de text.
  - Altres atributs que cregueu que puguin ser necessaris per gestionar la interacció gràfica o el funcionament de la partida.
- Us suggerim que tingui com a mínim aquests **mètodes**:
  - **`inicialitza`**: s'encarrega d'inicialitzar la partida en mode normal o en mode test.
  - **`actualitza`**: s'encarrega de cridar als mètodes de la classe `Joc` per fer els moviments de la figura en funció de l'entrada de tecla o del següent moviment en mode test i de la visualització de l'estat actual de la partida per pantalla.

Noves classes



Classe Partida: mètodes

```
void inicialitza(int mode, const string& fitxerInicial, const string& fitxerFigures,
        const string& fitxerMoviments);
```

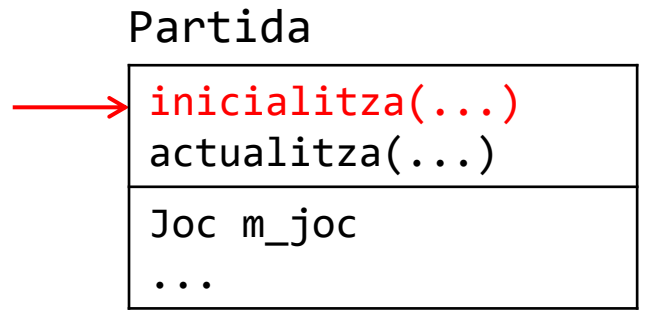
Paràmetres:

- `mode`: mode de joc de la partida, normal o test.
- `fitxerInicial`: fitxer amb l'estat inicial de la figura i del tauler al mode test. Segueix el **mateix format** que el **fitxer d'inicialització** de la **primera part del projecte**.
- `fitxerFigures`: fitxer amb la seqüència d'aparició de les figures al mode test.
- `fitxerMoviments`: fitxer amb la seqüència de moviments a realitzar en mode test.

Descripció:

- Aquest mètode es crida al principi de la partida i s'encarrega d'inicialitzar la partida en mode normal o en mode test.
  - Haurà de cridar al mètode `inicialitza` de la classe `Joc` i haurà d'inicialitzar tots els altres atributs de la classe per poder començar una nova partida.
  - Si estem en mode test haurà de llegir dels fitxers corresponents la seqüència d'aparició de les figures i la seqüència de moviments a realitzar i guardar-les en atributs de la classe. Tant la seqüència de figures com la seqüència de moviments s'hauran de guardar utilitzant **llistes dinàmiques amb nodes enllaçats**, com les que veurem en les properes sessions de classe.

Noves classes



Classe Partida: mètodes

```
void inicialitza(int mode, const string& fitxerInicial, const string& fitxerFigures,
        const string& fitxerMoviments);
```

fitxerFigures

```
tipus_figura_1 fila columna gir
tipus_figura_2 fila columna gir
...
```

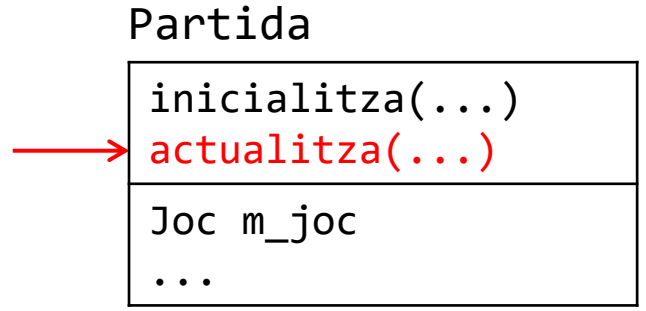
Codificats de la mateixa forma que al fitxer d'inicialització de la primera part del projecte

fitxerMoviments

```
tipus_moviment_1
tipus_moviment_2
...
```

```
typedef enum
{
    MOVIMENT_ESQUERRA = 0,
    MOVIMENT_DRETA = 1,
    MOVIMENT_GIR_HORARI = 2,
    MOVIMENT_GIR_ANTI_HORARI = 3,
    MOVIMENT_BAIXA = 4,
    MOVIMENT_BAIXA_FINAL = 5,
} TipusMoviment;
```

Noves classes



Classe `Partida`: mètodes

```
void actualitza(int mode, double deltaTime);
```

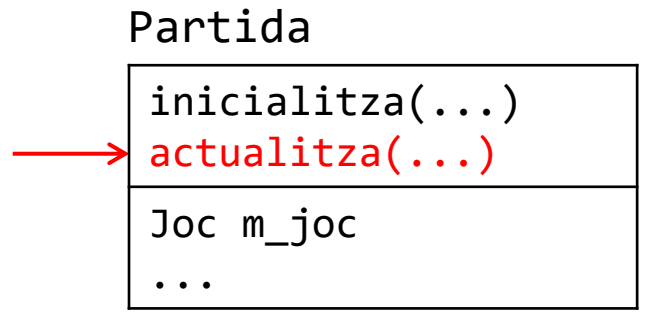
Paràmetres:

- `mode`: mode de joc de la partida, normal o test.
- `deltaTime`: temps que ha passat des de l'última crida al mètode `actualitza`. Ho farem servir per controlar quan hem de fer baixar la figura una fila.

Descripció:

- Aquest mètode es crida a cada cicle del joc des de la funció principal del joc.
- S'encarrega de:
  - Executar els moviments de la figura en funció de de l'entrada de teclat o de l'ordre de moviments en mode test, fent les crides corresponents als mètodes de la classe `Joc`.
  - Actualitzar la puntuació del joc.
  - Actualitzar el nivell del joc si la puntuació superar el llindar d'increment de nivell.
  - Mostrar per pantalla el tauler amb les peces col·locades, la figura a la seva posició actual, la puntuació actual i el nivell actual del joc.

Noves classes



Classe `Partida`: mètodes

```
void actualitza(int mode, double deltaTime);
```

Funcionament d'una partida:

Mode normal

- La figura es mou o gira si s'ha pressionat la tecla corresponent.
  - FLETXA_ESQUERRA / FLETXA_DRETA: desplaçaments laterals
  - FLETXA_AMUNT: gir horari
  - FLETXA_ABAIX: gir antI-horari
  - ESPAI: fa baixar la figura fins a quedar col·locada al tauler.
  - ESCAPE: surt de la partida.
- La figura baixa una fila si ha passat l'interval de temps que determina el nivell de joc des de l'últim moviment de baixar.
- El tipus de les noves figures que apareixen per la part superior del tauler es determina de forma aleatòria. També el gir i la posició inicial de la figura.

Noves classes

Classe `Partida`: mètodes

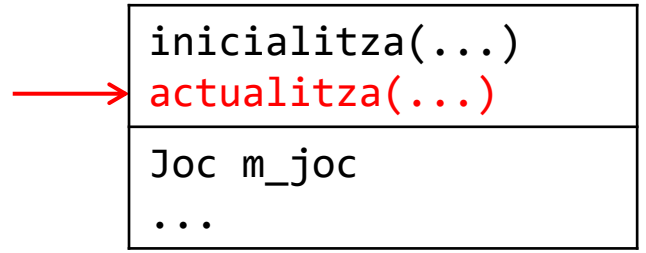
```
void actualitza(int mode, double deltaTime);
```

Funcionament d'una partida:

Mode test

- La figura es mou o gira segons la seqüència de moviments llegida del fitxer i codificada amb el tipus `TipusMoviment`.
- Només es fa un moviment si ha passat un cert interval de temps des de l'últim moviment.
- El tipus de les noves figures que apareixen per la part superior del tauler es determina a partir de la seqüència llegida del fitxer. També el gir i la posició inicial de la figura.

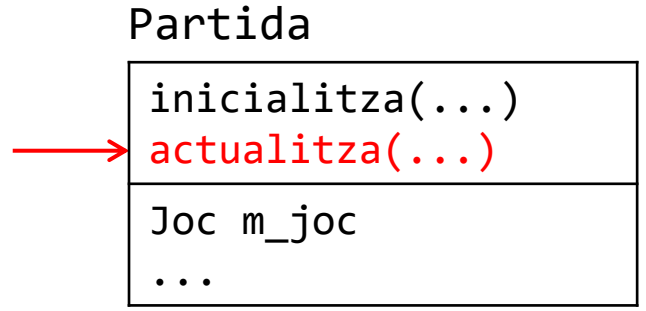
Partida



InfoJoc.h

```
typedef enum
{
    MOVIMENT_ESQUERRA,
    MOVIMENT_DRETA,
    MOVIMENT_GIR_HORARI,
    MOVIMENT_GIR_ANTI_HORARI,
    MOVIMENT_BAIXA,
    MOVIMENT_BAIXA_FINAL,
} TipusMoviment;
```

Noves classes



Classe Partida: mètodes

```
void actualitza(int mode, double deltaTime);
```

Funcionament d'una partida:

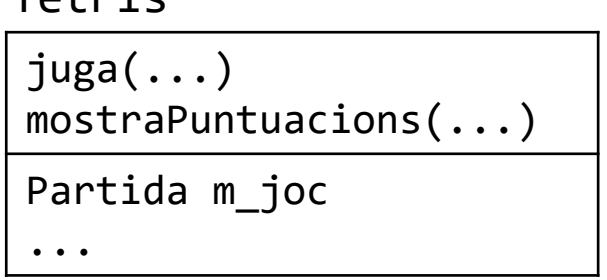
Actualització de la puntuació

- Cada cop que una figura queda col·locada al tauler s'incrementa la puntuació en 10 punts.
- Cada cop que s'elimina una fila del tauler s'incrementa la puntuació en 100 punts.
- Si s'eliminen dues files de cop, s'obté una puntuació extra de 50 punts, si se n'eliminen 3, 75 punts i si se n'eliminen 4, 100 punts.

Actualització del nivell del joc

- Cada 1000 punts s'incrementa el nivell del joc.
- Incrementar el nivell del joc implica disminuir l'interval de temps que ha de passar perquè la figura baixi una fila. El valor d'aquest interval de temps per cada nivell el podeu fixar segons el vostre criteri.

Noves classes



Classe `Tetris`

- Guarda tota la **informació** necessària per gestionar una **partida** i la **llista de puntuacions** històriques.
  - Un **objecte** de la classe **`Partida`** per poder gestionar el funcionament de la partida.
  - La **llista de puntuacions** històriques.
    - Aquesta llista s'ha de llegir d'un fitxer al principi del programa, s'ha d'anar actualitzant a mesura que es van jugant partides i s'ha de tornar a guardar al fitxer quan s'acaba el programa.
    - S'ha de guardar utilitzant alguna **classe de la llibreria estàndard: `forward_list` o `list`.**
    - Els valors en aquesta llista han d'estar sempre en ordre descendent de puntuació.
  - Altres atributs que cregueu que puguin ser necessaris per gestionar el menú del joc, la partida i les puntuacions.
- Us suggerim que tingui com a mínim aquests **mètodes**:
  - **`juga`**: s'encarrega d'inicialitzar i executar una partida, cridant als mètodes que calgui de la classe Partida, i seguint el que us expliquem a la introducció a la llibreria gràfica.
  - **`mostraPuntuacions`**: s'encarrega de mostrar per pantalla la llista de puntuacions històriques.

Menú principal del joc

- El programa tindrà un **menú** que s'executarà des del programa principal en mode no gràfic que tindrà aquestes quatre opcions:
  1. Jugar en mode normal: permetrà jugar una partida en mode normal. Al final de la partida es demanarà per teclat el nom del jugador i s'afegirà la puntuació a la llista de puntuacions històriques.
  2. Jugar en mode test: demanarà per teclat el nom dels fitxers necessaris per inicialitzar una partida en mode test i executarà tota la partida.
  3. Visualitzar llista de millors puntuacions
  4. Sortir del programa

Modificacions de classes existents

Classe `Joc`

- Us suggerim **afegir** un **mètode `dibuixa`** que permeti visualitzar per pantalla el tauler i la figura actual (cridant als mètodes corresponents de les classes `Tauler` i `Figura`).
- Haureu de **canviar** el mètode **`inicialitza`** per poder inicialitzar el joc en mode normal i llegir els fitxers amb la seqüència de figures i de moviments en mode test.
- Haureu d'afegir un **nou mètode** per **inicialitzar una nova figura** que apareix per dalt del tauler.
- Haureu d'afegir un **nou mètode** per **fer baixar la figura** de cop fins que queda col·locada al tauler.

Classe `Tauler`

- Us suggerim **afegir** un **mètode `dibuixa`** que permeti visualitzar per pantalla el tauler.

Classe `Figura`

- Us suggerim **afegir** un **mètode `dibuixa`** que permeti visualitzar per pantalla la figura actual

Consells d'implementació: Següents passos

1. Si cal, completeu el codi de la primera part del projecte. Comproveu el seu correcte funcionament amb el test que hi ha a Caronte.
2. Completeu els exercicis que us proposem per familiaritzar-vos amb la utilització de les funcions de la llibreria gràfica.
3. Creeu un nou projecte integrant el vostre codi de la primera part amb el codi de la llibreria gràfica. Comproveu que compila correctament sense fer encara cap crida a les funcions de la llibreria gràfica.
4. Implementeu només el codi necessari per visualitzar la figura actual i el tauler amb l'estat inicial de la partida que s'ha llegit des del fitxer de configuració. No contempleu encara la interacció amb el jugador a través del teclat.
5. Incorporeu la interacció amb el teclat per poder fer el moviment i el gir de la figura.
6. Integreu la caiguda de la figura pas a pas en intervals fixos i l'aparició de noves figures quan la figura actual queda col·locada.
7. Implementeu l'actualització de la puntuació i del nivell actual.
8. Incorporeu la possibilitat de jugar en els dos modes: normal i test.
9. Implementeu la detecció i la gestió del final de la partida.
10. Implementeu la inicialització i actualització de la llista de puntuacions.
11. Implementeu el menú del joc i la visualització de la llista de puntuacions.

Planificació del projecte

- **Lliurament final** del projecte: dimarts **4 de juny**
- Durant els dies del **5 al 7 de juny** farem l'**avaluació** amb entrevistes **online** a cada grup per Teams.
- A la sessió de classe del dimecres **22 de maig** dedicarem una estona a resoldre **dubtes i preguntes** sobre el projecte.