



# Projecte: Escacs

## Introducció llibreria gràfica

Metodologia de la Programació

Curs 2021 - 2022

# Objectius de la sessió

1. Familiaritzar-se amb les funcions bàsiques de la llibreria gràfica que farem servir al projecte
2. Explicar les primeres tasques a realitzar per mostrar els gràfics per pantalla

# Utilització de la llibreria gràfica

## Passos a seguir per poder utilitzar la llibreria gràfica:

1. Descarregar i descomprimir el fitxer de Caronte corresponent a l'exemple d'utilització de la llibreria gràfica.
2. Obrir el projecte
  - Assegureu-vos que la **configuració** està fixada a **x86**
  - Assegureu-vos que a *Proyecto – Propiedades – Depuración*, la opció *Directorio de Trabajo* està posada a aquest valor:  
`$(ProjectDir)\..\..\1. Resources`
3. Afegiu al projecte (a la carpeta de codi \LogicGame\Chess) els fitxers del vostre codi de la primera versió del projecte (classes ChessPosition, Piece, ChessBoard)

# Utilització de la llibreria gràfica

## Estructura de directoris del projecte:

▼ Segona entrega

▼ 0. C++ Code

> Graphic Lib

> Logic Game

▼ 1. Resources

> data

▼ 2. Platforms

> 0. Windows Desktop

Carpeta que conté tot el codi C++ del projecte:

- **Graphic Lib: conté el codi de la llibreria gràfica que farem servir. NO S'HI HA D'AFEGIR NI MODIFICAR RES**
- *Logic Game*: conté el codi que implementa el joc. Aquí és on heu d'afegir tots els fitxers del codi que feu vosaltres. Quan afegiu un nou fitxer .h o .cpp, assegureu-vos d'afegir-lo també al projecte de Visual

Carpeta que conté tots els recursos necessaris per executar el joc: fitxers dels gràfics, fonts per mostrar el text i els diccionaris amb les paraules vàlides per cada idioma.

Carpeta que conté els fitxers de la solució en Visual Studio

- Per obrir el projecte heu d'obrir el fitxer `MP_Practica.sln` que hi ha a la carpeta *0. Windows Desktop*

# Utilització de la llibreria gràfica

**main.cpp:** inicialització de la llibreria i crida a la funció principal del joc.  
**NO L'HEU DE MODIFICAR**

```
int main(int argc, const char* argv[])
{
```

```
    //Instruccions necessaries per poder incloure la llibreria i que trobi el main
```

```
    SDL_SetMainReady();
```

```
    SDL_Init(SDL_INIT_VIDEO);
```

```
    //Inicialitza un objecte de la classe Screen que s'utilitza per gestionar la finestra gràfica.
```

```
    Screen pantalla(SCREEN_SIZE_X, SCREEN_SIZE_Y);
```

```
    //Mostrem la finestra grafica
```

```
    pantalla.show();
```

```
    CurrentGame game;
```

```
    game.init(GM_NORMAL, "data/Games/board.txt", "data/Games/movements.txt");
```

```
    do
```

```
    {
```

```
        // Captura tots els events de ratolí i teclat de la funció
```

```
        pantalla.processEvents();
```

```
        bool mouseStatus = Mouse_getBtnLeft();
```

```
        int mousePosX = Mouse_getX();
```

```
        int mousePosY = Mouse_getY();
```

```
        bool final = game.updateAndRender(mousePosX, mousePosY, mouseStatus);
```

```
        // Actualitza la pantalla
```

```
        pantalla.update();
```

```
    } while (!Keyboard_GetKeyTrg(KEYBOARD_ESCAPE));
```

```
    // Sortim del bucle si pressionem ESC
```

```
    game.end();
```

```
    SDL_Quit();
```

```
    return 0;
```

Crea l'objecte de tipus Screen que s'utilitza per gestionar la finestra gràfica.

Inicialitza la pantalla al tamany definit per SCREEN\_SIZE\_X i SCREEN\_SIZE\_Y

Fa que es mostri la finestra gràfica

Declara l'objecte de tipus CurrentGame que controlarà l'execució de la partida

Captura tots els events de teclat i ratolí que s'hagin produït des de l'última crida a la funció

Crida a la **funció principal de la classe CurrentGame** que controla què passa en una iteració del joc. Se li passen com a paràmetres la posició i l'estat del ratolí. Aquesta és la funció que **haurem de modificar** per posar-hi el codi que controla el funcionament de la partida.

Refresca la pantalla redibuixant tots els gràfics a la seva posició actual

Detecta si s'ha pressionat la tecla ESC per sortir del bucle del joc

# Utilització de la llibreria gràfica

## Visualització de gràfics

- Anem a dibuixar a pantalla el gràfic amb el tauler buit:



# Utilització de la llibreria gràfica

## Visualització de gràfics

```
#include "../GraphicManager.h"
```

```
bool CurrentGame::updateAndRender(int mousePosX, int mousePosY, bool mouseStatus)
{
    GraphicManager::getInstance()->drawSprite(IMAGE_BOARD, 0, 0);
}
```

```
typedef enum {
    IMAGE_BOARD,
    IMAGE_VALID_POS,
    IMAGE_PIECE_KING_BLACK,
    IMAGE_PIECE_KING_WHITE,
    IMAGE_PIECE_QUEEN_BLACK,
    IMAGE_PIECE_QUEEN_WHITE,
    IMAGE_PIECE_ROOK_BLACK,
    IMAGE_PIECE_ROOK_WHITE,
    ...
    IMAGE_PIECE_PAWN_BLACK,
    IMAGE_PIECE_PAWN_WHITE,
    IMAGE_NUM_MAX
} IMAGE_NAME;
```

Dibuixa un gràfic en una posició determinada de la pantalla. Paràmetres:

- Nom del gràfic a dibuixar
- Posició X i Posició Y on dibuixar-lo (coordenada de la cantonada superior esquerra del gràfic). ATENCIÓ: Posició X correspon a la columna i Posició Y a la fila

Al fitxer `GraphicManager.h`: definició de tots els noms de gràfics que es poden dibuixar.

# Utilització de la llibreria gràfica

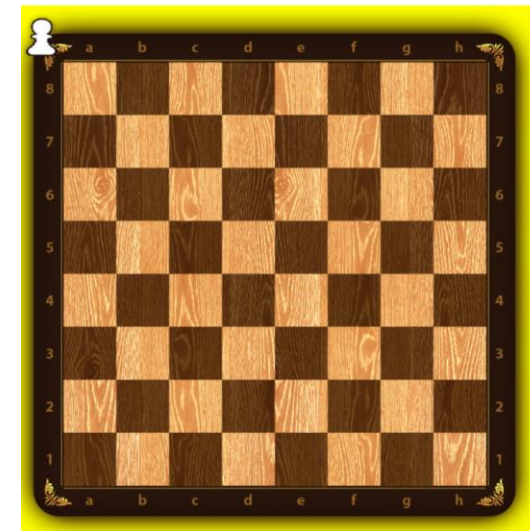
## Visualització de gràfics

- Anem ara a dibuixar el tauler de joc i un peó blanc a la posició (0,0) de la pantalla.

```
#include "../GraphicManager.h"
```

```
bool CurrentGame::updateAndRender (int mouseX, int mouseY, bool mouseStatus)
{
    GraphicManager::getInstance()->drawSprite(IMAGE_BOARD, 0, 0);
    GraphicManager::getInstance()->drawSprite(IMAGE_PIECE_PAWN_WHITE, 0, 0);
}
```

- Què passa si intercanviem l'ordre de les crides?
- Es gràfics es dibuixen a pantalla en l'ordre en què es fan les crides amagant els que s'hagin dibuixat abans a la mateixa posició. Ens hem d'assegurar de dibuixar primer els objectes del fons i després els que han d'estar en primer pla.





# Utilització de la llibreria gràfica

## Visualització de gràfics

- Com podem fer que la peça es dibuixi a la posició (0,0) del tauler?

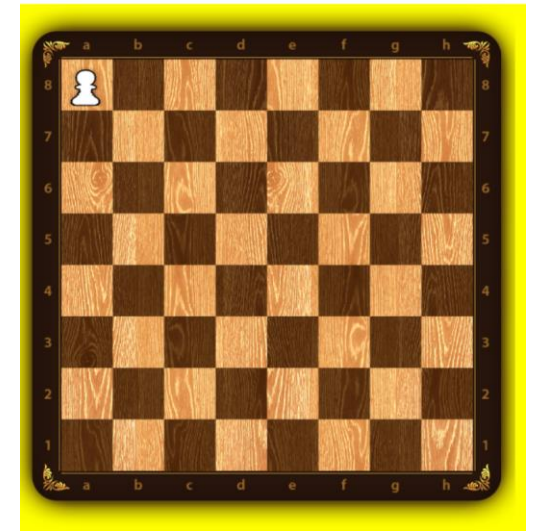
```
#include "../GraphicManager.h"
```

```
bool CurrentGame::updateAndRender (int mouseX, int mouseY, bool mouseStatus)
{
    GraphicManager::getInstance()->drawSprite(IMAGE_BOARD, 0, 0);
    GraphicManager::getInstance()->drawSprite(IMAGE_PIECE_PAWN_WHITE, CELL_INIT_X, CELL_INIT_Y);
}
```

Definicions al fitxer GameInfo.hpp

```
// Tamany en píxels dels gràfics de les peces
const int CELL_W = 62;
const int CELL_H = 62;

// Posició a pantalla de la casella superior esquerra del tauler
const int CELL_INIT_X = 55;
const int CELL_INIT_Y = 64;
```



**Exercici:** Com podem fer per dibuixar la peça a una casella qualsevol del tauler definida per les variables posX i posY?

# Utilització de la llibreria gràfica

## Captura dels events del ratolí

- Volem fer que el peó només es dibuixi si estem pressionant el botó del ratolí

Rep com a paràmetres:

- mousePosX, mousePosY: coordenades de la posició actual del ratolí
- mouseStatuts: indica si el botó esquerre de ratolí està pressionat (true) o no (false)

```
bool CurrentGame::updateAndRender(int mousePosX, int mousePosY, bool mouseStatus)
{
    int posX = 4;
    int posY = 6;
    GraphicManager::getInstance()->drawSprite(IMAGE_BOARD, 0, 0);
    if (mouseStatus)
        GraphicManager::getInstance()->drawSprite(IMAGE_PIECE_PAWN_WHITE,
            CELL_INIT_X + (posX * CELL_W), CELL_INIT_Y + (posY * CELL_H));
}
```

**Exercici:** Com podem fer per dibuixar el peó només si estem pressionant el ratolí i a més a més el ratolí està dins dels límits del tauler?

# Utilització de la llibreria gràfica

## Mostrar text per pantalla

- Mostrar la posició actual del ratolí a sota del tauler

```
bool CurrentGame::updateAndRender (int mouseX, int mouseY, bool mouseStatus)
{
    int posX = 4;
    int posY = 6;
    GraphicManager::getInstance()->drawSprite(IMAGE_BOARD, 0, 0);
    if ((mouseStatus) && (mousePosX >= CELL_INIT_X) && (mousePosY >= CELL_INIT_Y) &&
        (mousePosX <= (CELL_INIT_X + CELL_W * NUM_COLS)) &&
        (mousePosY <= (CELL_INIT_Y + CELL_H * NUM_ROWS)))
        GraphicManager::getInstance()->drawSprite(IMAGE_PIECE_PAWN_WHITE,
            CELL_INIT_X + (posX * CELL_W), CELL_INIT_Y + (posY * CELL_H));

    int posTextX = CELL_INIT_X;
    int posTextY = CELL_INIT_Y + (CELL_H * NUM_ROWS) + 60;

    std::string msg = "PosX: " + to_string(mousePosX) + ", PosY: " + to_string(mousePosY);
    GraphicManager::getInstance()->drawFont(FONT_RED_30, posTextX, posTextY, 0.8, msg);
}
```



Mostra per pantalla el text guardat a msg a la posició posTextX, posTextY amb escala de tamany 0.6

# Utilització de la llibreria gràfica

**GameInfo.hpp:** definició de constants amb el tamany i posició dels objectes del joc que poden ser necessàries per col·locar i visualitzar els objectes a pantalla

```
const int NUM_COLS = 8;
const int NUM_ROWS = 8;

// Tamany en píxels dels gràfics de les peces
const int CELL_W = 62;
const int CELL_H = 62;


// Posició a pantalla de la casella superior esquerra del tauler
const int CELL_INIT_X = 55;
const int CELL_INIT_Y = 64;

// Posició a pantalla de la casella superior esquerra del tauler per dibuixar les posicions vàlides
const int GREEN_INIT_X = 50;
const int GREEN_INIT_Y = 67;

// Tamany en píxels de la pantalla
const int SCREEN_SIZE_X = 750;
const int SCREEN_SIZE_Y = 850;
```

# Utilització de la llibreria gràfica

**GraphicManager.h:** definició de constants per poder dibuixar els gràfics i mostrar el text amb diferents tipus de fonts

```
typedef enum {  
    IMAGE_BOARD,  
    IMAGE_VALID_POS,   
    IMAGE_PIECE_KING_BLACK,  
    IMAGE_PIECE_KING_WHITE,  
    IMAGE_PIECE_QUEEN_BLACK,  
    IMAGE_PIECE_QUEEN_WHITE,  
    IMAGE_PIECE_ROOK_BLACK,  
    IMAGE_PIECE_ROOK_WHITE,  
    IMAGE_PIECE_BISHOP_BLACK,  
    IMAGE_PIECE_BISHOP_WHITE,  
    IMAGE_PIECE_KNIGHT_BLACK,  
    IMAGE_PIECE_KNIGHT_WHITE,  
    IMAGE_PIECE_PAWN_BLACK,  
    IMAGE_PIECE_PAWN_WHITE,  
    IMAGE_NUM_MAX  
} IMAGE_NAME;
```

S'utilitza per dibuixar un requadre verd transparent a sobre de les posicions dels moviments vàlids de la peça seleccionada.

```
typedef enum {  
    FONT_WHITE_30 = 0,  
    FONT_RED_30,  
    FONT_GREEN_30,  
  
    FONT_NUM_MAX  
} FONT_NAME;
```

# Exercici

Mostrar l'estat inicial del joc amb totes les peces col·locades a la seva posició inicial



# Exercici

Mostrar l'estat inicial del joc amb totes les peces col·locades a la seva posició inicial

1. Afegir al projecte tots els fitxers de codi del lliurament parcial amb la implementació de les classes `ChessPosition`, `Piece` i `Chessboard`  
(\*): Haureu de moure la declaració de `VecOfPositions` del fitxer `GameInfo.h` al fitxer `ChessPosition.hpp`
2. Afegir a la classe `CurrentGame` un atribut de tipus `Chessboard` per guardar la informació del tauler.
3. Modificar la implementació del mètode `init` de la classe `CurrentGame` perquè inicialitzi el tauler a partir d'un fitxer cridant al mètode `LoadBoardFromFile` de la classe `Chessboard`
4. Afegir a la classe `Piece` un mètode `Render` que dibuixi el gràfic que correspon al tipus de la peça a la posició `x` i `y` que se li passa com a paràmetre.
5. Afegir a la classe `Chessboard` un mètode `Render` que dibuixi totes les peces del tauler a la seva posició utilitzant el mètode `Render` de la classe `Piece` anterior.
6. Modificar el mètode `updateAndRender` de la classe `CurrentGame` perquè dibuixi tot el tauler cridant al mètode `Render` de la classe `Chessboard`