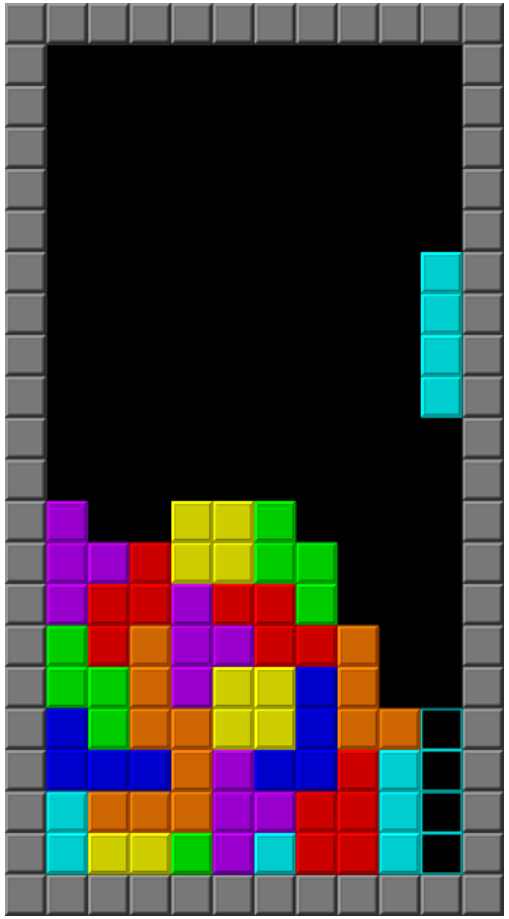


Projecte: Tetris

Introducció Llibreria Gràfica

Metodologia de la Programació

Curs 2023 - 2024

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

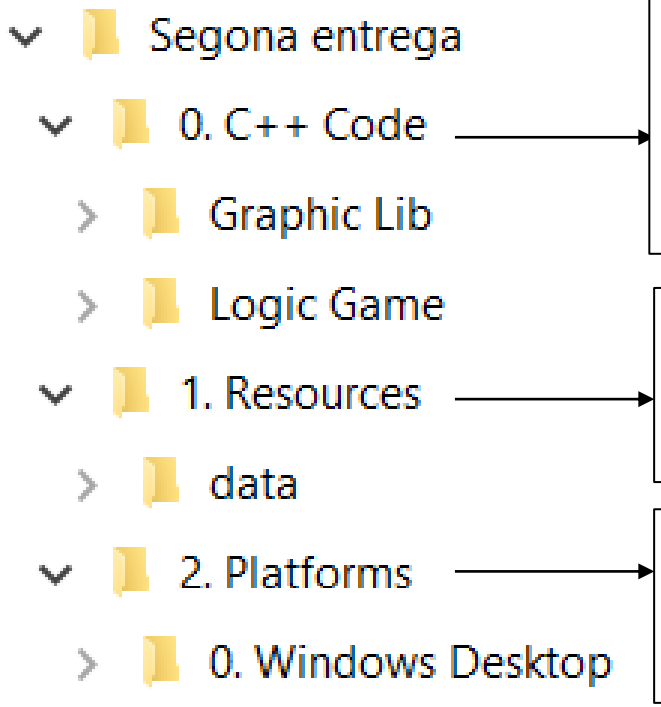
Passos a seguir per poder utilitzar la llibreria gràfica:

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 `\0. C++ Code\LogicGame`) els fitxers del vostre codi de la primera versió del projecte (classes `Figura`, `Tauler`, `Joc`, ...)

Utilització de la llibreria gràfica

Estructura de directoris del projecte:



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 fitxers per inicialitzar el tauler i els moviments a reproduir.

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

Codi que inicialitza la llibreria i executa el joc en mode gràfic. L'heu de reproduir quan vulgueu executar una partida (en mode normal o de test) des del menú del joc

```
int main(int argc, const char* argv[])
{
    //Instruccions necesaries 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'ut
    Screen pantalla(SCREEN_SIZE_X, SCREEN_SIZE_Y);
    //Mostrem la finestra grafica
    pantalla.show();

    Partida game;

    Uint64 NOW = SDL_GetPerformanceCounter();
    Uint64 LAST = 0;
    double deltaTime = 0;
    do
    {
        LAST = NOW;
        NOW = SDL_GetPerformanceCounter();
        deltaTime = (double)((NOW - LAST) / (double)SDL_GetPerformanceFrequer
        // Captura tots els events de ratolí i teclat de l'ultim cicle
        pantalla.processEvents();

        game.actualitza(deltaTime);
        // Actualitza la pantalla
        pantalla.update();
    } while (!Keyboard_GetKeyTrg(KEYBOARD_ESCAPE));
    // Sortim del bucle si pressionem ESC
    //Instruccio necesaria per alliberar els recursos de la llibrer
    SDL_Quit();
```

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 i inicialitza l'objecte de tipus `Partida` 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 `Partida`** que controla què passa en una iteració del joc. Aquesta és la funció que **haurem de modificar** per posar-hi el codi que controla el funcionament de la partida.

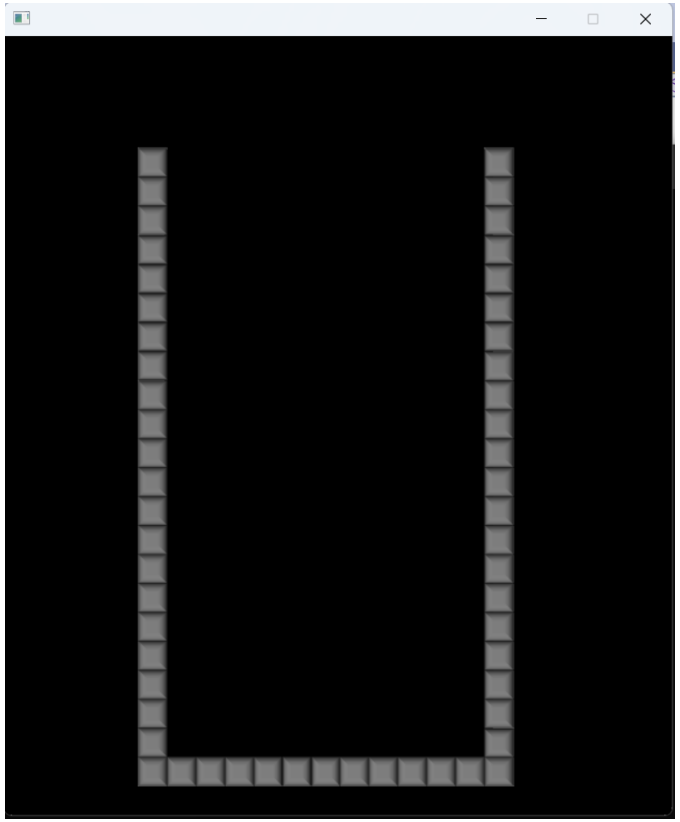
Refresca la pantalla redibuixant tots els gràfics a la seva posició actual

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

Dibuixa un gràfic en una posició determinada de la pantalla. Paràmetres:
- Nom del gràfic a dibuixar
- Posició X i Posició Y on dibuixar-lo. Si l'últim paràmetre és true, la posició correspon (x, y) correspon al punt central del gràfic. Si és false, a les coordenades de la cantonada superior esquerra del gràfic.

ATENCIÓ: Posició X correspon a la columna i Posició Y a la fila

```
#include "GraphicManager.h"

void Joc::actualitza(double deltaTime)
{
    GraphicManager::getInstance()->drawSprite(GRAFIC_FONS, 0, 0, false);
    GraphicManager::getInstance()->drawSprite(GRAFIC_TAULER, POS_X_TAULER, POS_Y_TAULER, false);}
```

```
typedef enum {
    GRAFIC_FONS = 0,
    GRAFIC_TAULER,
    GRAFIC_QUADRAT_GROC,
    GRAFIC_QUADRAT_BLAUCEL,
    GRAFIC_QUADRAT_MAGENTA,
    GRAFIC_QUADRAT_TARONJA,
    GRAFIC_QUADRAT_BLAUFOSC,
    GRAFIC_QUADRAT_VERMELL,
    GRAFIC_QUADRAT_VERD,
    GRAFIC_NUM_MAX
} IMAGE_NAME;
```

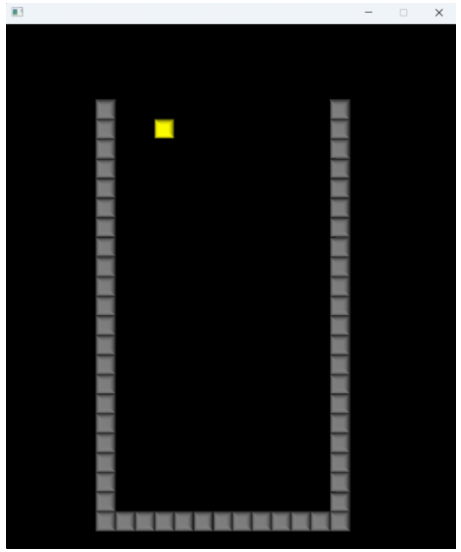
Definicions al fitxer `InfoJoc.h`

```
// Posició del tauler a la pantalla
const int POS_X_TAULER = 120;
const int POS_Y_TAULER = 100;
```

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

- 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

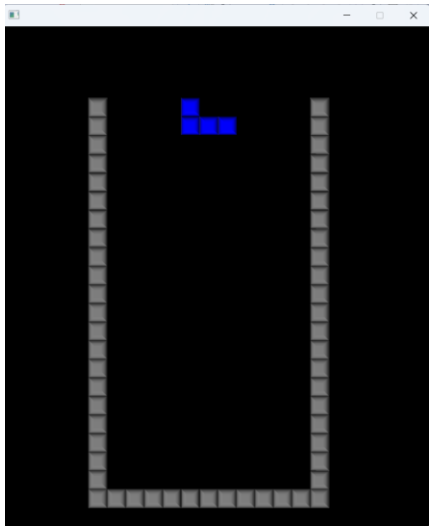
- Anem ara a dibuixar el tauler de joc i un quadrat groc a la posició `(2, 3)` del tauler.

```
void Joc::actualitza(double deltaTime)
{
    int fila = 2;
    int columna = 3;
    GraphicManager::getInstance()->drawSprite(GRAFIC_FONS, 0, 0, false);
    GraphicManager::getInstance()->drawSprite(GRAFIC_TAULER, POS_X_TAULER, POS_Y_TAULER, false);
    GraphicManager::getInstance()->drawSprite(GRAFIC_QUADRAT_GROC,
        POS_X_TAULER + (columna * MIDA_QUADRAT), POS_Y_TAULER + ((fila - 1) * MIDA_QUADRAT), false);
}
```

Definicions al fitxer `InfoJoc.h`

```
// Mida dels quadrats que formen el tauler
const int MIDA_QUADRAT = 26;
```

Utilització de la llibreria gràfica



Exercici

Dibuixar a pantalla el gràfic amb el tauler i la figura representada a l'atribut `m_forma` a la posició del tauler indicada per `m_fila` i `m_columna`

Utilització de la llibreria gràfica

Captura dels events del teclat

- Volem dibuixar per pantalla un quadrat groc que es vagi movent cap a la dreta cada cop que es prem la tecla de la fletxa dreta.

Retorna true si s'ha pressionat la tecla que es passa com a paràmetre

```
void Joc::actualitza(double deltaTime)
{
    GraphicManager::getInstance()->drawSprite(GRAFIC_FONS, 0, 0, false);
    GraphicManager::getInstance()->drawSprite(GRAFIC_TAULER, POS_X_TAULER, POS_Y_TAULER, false);
    if (Keyboard_GetKeyTrg(KEYBOARD_RIGHT))
        if (m_columna < N_COL_TAULER)
            m_columna++;
    GraphicManager::getInstance()->drawSprite(GRAFIC_QUADRAT_GROC, POS_X_TAULER + (m_columna * MIDA_QUADRAT),
        POS_Y_TAULER + ((m_fila - 1) * MIDA_QUADRAT), false);
}
```

Definicions de les tecles al fitxer `keyboard_sdl.h`

```
#define KEYBOARD_A                  SDLK_a
#define KEYBOARD_B                  SDLK_b
...
#define KEYBOARD_RIGHT              SDLK_RIGHT
#define KEYBOARD_LEFT               SDLK_LEFT
#define KEYBOARD_DOWN               SDLK_DOWN
#define KEYBOARD_UP                 SDLK_UP
```

Utilització de la llibreria gràfica

Exercici

Dibuixar a pantalla el gràfic amb el tauler i la figura representada a l'atribut `m_forma` a la posició del tauler indicada per `m_fila` i `m_columna`. Moure la figura cap a la dreta o l'esquerra si es pressionen les tecles de les fletxes corresponents, comprovant que no ens passem dels limits del tauler.

Utilització de la llibreria gràfica

Introduir un temps d'espera per actualitzar la pantalla

- Dibuixar per pantalla un quadrat groc a la posicio indicada per `m_fila` i `m_columna`. Fer que cada mig segon baixi una fila.

```
void Joc::actualitza(double deltaTime)
{
    GraphicManager::getInstance()->drawSprite(GRAFIC_FONS, 0, 0, false);
    GraphicManager::getInstance()->drawSprite(GRAFIC_TAULER, POS_X_TAULER, POS_Y_TAULER, false);
    m_temps += deltaTime;
    if (m_temps > 0.5)
    {
        if (m_fila < N_FILES_TAULER)
            m_fila++;
        m_temps = 0.0;
    }
    GraphicManager::getInstance()->drawSprite(GRAFIC_QUADRAT_GROC,
            POS_X_TAULER + (m_columna * MIDA_QUADRAT),
            POS_Y_TAULER + ((m_fila - 1) * MIDA_QUADRAT), false);
}
```

Temps que ha passat des de l'última crida al mètode

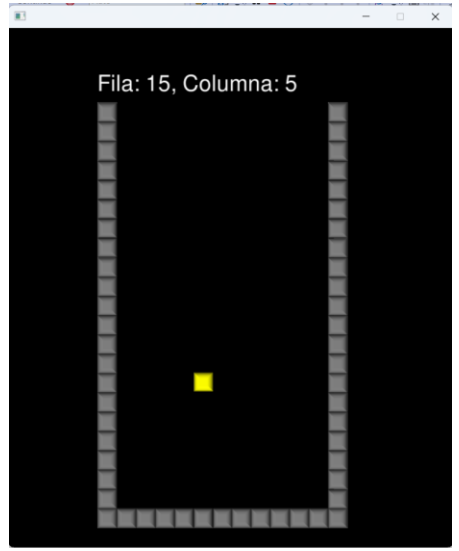
Actualitza el temps d'espera amb el temps que ha passat des de l'última crida al mètode `actualitza`. Només actualitza la fila si ha passat mig segon.

Utilització de la llibreria gràfica

Exercici

Dibuixar per pantalla un quadrat groc a la posicio indicada per `m_fila` i `m_columna`. Moure el quadrat cap a la dreta o l'esquerra si es pressionen les tecles de les fletxes corresponents, comprovant que no ens passem dels limits del tauler. Fer que cada segon baixi una fila, comprovant que no ens passem del limit inferior del tauler.

Utilització de la llibreria gràfica



Mostrar text per pantalla

- Mostrar la posició actual del quadrat a dalt del tauler

```
void Joc::actualitza(double deltaTime)
{
    GraphicManager::getInstance()->drawSprite(GRAFIC_FONS, 0, 0, false);
    GraphicManager::getInstance()->drawSprite(GRAFIC_TAULER, POS_X_TAULER, POS_Y_TAULER, false);
    m_temps += deltaTime;
    if (m_temps > 0.5)
    {
        if (m_fila < N_FILES_TAULER)
            m_fila++;
        m_temps = 0.0;
    }
    GraphicManager::getInstance()->drawSprite(GRAFIC_QUADRAT_GROC,
            POS_X_TAULER + (m_columna * MIDA_QUADRAT),
            POS_Y_TAULER + ((m_fila - 1) * MIDA_QUADRAT), false);
    string msg = "Fila: " + to_string(m_fila) + ", Columna: " + to_string(m_columna);
    GraphicManager::getInstance()->drawFont(FONT_WHITE_30, POS_X_TAULER, POS_Y_TAULER - 50, 1.0, msg);
}
```

Mostra per pantalla el text guardat a `msg` a la posició `POS_X_TAULER, POS_Y_TAULER - 50` amb escala de mida 1.0

Utilització de la llibreria gràfica

`InfoJoc.h`:

- Definició de constants amb la mida i posició dels objectes del joc que poden ser necessàries per col·locar i visualitzar els objectes a pantalla.
- Definició de tipus comuns necessaris
- Afegiu la definició d'altres constants i tipus que pugueu necessitar

```
// Tamany de la pantalla gràfica
const int SCREEN_SIZE_X = 600;
const int SCREEN_SIZE_Y = 700;

// Mida dels quadrats que formen el tauler
const int MIDA_QUADRAT = 26;

// Tamany del tauler
const int N_FILES_TAULER = 21;
const int N_COL_TAULER = 11;

// Posició del tauler a la pantalla
const int POS_X_TAULER = 120;
const int POS_Y_TAULER = 100;
```

```
typedef enum
{
    NO_FIGURA = 0,
    FIGURA_O,
    FIGURA_I,
    FIGURA_T,
    FIGURA_L,
    FIGURA_J,
    FIGURA_Z,
    FIGURA_S
} TipusFigura;
```

```
typedef enum
{
    COLOR_NEGRE = 0,
    COLOR_GROC,
    COLOR_BLAUCEL,
    COLOR_MAGENTA,
    COLOR_TARONJA,
    COLOR_BLAUFOSC,
    COLOR_VERMELL,
    COLOR_VERD,
    NO_COLOR
} ColorFigura;
```

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
{
    FONT_WHITE_30 = 0,
    FONT_RED_30,
    FONT_GREEN_30,

    FONT_NUM_MAX
} FONT_NAME;
```

```
typedef enum
{
    GRAFIC_FONS = 0,
    GRAFIC_TAULER,
    GRAFIC_QUADRAT_GROC,
    GRAFIC_QUADRAT_BLAUCEL,
    GRAFIC_QUADRAT_MAGENTA,
    GRAFIC_QUADRAT_TARONJA,
    GRAFIC_QUADRAT_BLAUFOSC,
    GRAFIC_QUADRAT_VERMELL,
    GRAFIC_QUADRAT_VERD,
    GRAFIC_NUM_MAX
} IMAGE_NAME;
```

Exercici

Mostrar l'estat inicial del joc amb les peces tauler col·locades i la figura a la seva posició inicial.

1. Afegir al projecte tots els fitxers de codi del lliurament parcial amb la implementació de les classes `Figura`, `Tauler` i `Joc`
2. Afegir a la classe `Partida` un atribut de tipus `Joc` per guardar la informació de la partida.
3. Afegir un mètode `inicialitza` de la classe `Partida` perquè inicialitzi el tauler a partir d'un fitxer cridant al mètode `incialitza` de la classe `Joc`
4. Afegir a la classe `Figura` un mètode `dibuixa` que dibuixi la figura a la seva posició actual.
5. Afegir a la classe `Tauler` un mètode `dibuixa` que dibuixi totes les peces ja col·locades al tauler.
6. Afegir a la classe `Joc` un mètode `dibuixa` que dibuixi el tauler i la figura utilitzant el mètode `dibuixa` de les classes `Tauler` i `Figura`.
7. Modificar el mètode `actualitza` de la classe `Joc` perquè dibuixi tot el tauler cridant al mètode `dibuixa` de la classe `Partida.`