Université Abdelmalek Essaadi
Faculté ses Sciences et techniques de Tanger
Département Génie Informatique
LST GI S5
Algorithmique et POO C++

PROJET 'JEUX 2D ROLLER SPLAT

- **PRÉPARÉ PAR :**
  - -- IMAMI MOHAMED --
  - -- L'KHAMAL FADOUA --
  - -- EL OUAHABI IBRAHIM --
- **ENCADRE PAR :**
  - -- P. Ikram Ben Abdel Ouahab --
  - -- P. EL AACHAK LOTFI --

LST GI/S5/Groupe 2
GI-51 POO en C++

- **lien Github :** https://github.com/mohamedimami/game_rollersplate_poo_cpp

PLAN DE TRAVAIL

-- Introduction Général --

-- Développement --

- Définition des éléments du travail -

- Méthode du travail -

-- Conclusion --

-- Bibliographie --

-- Introduction General --

-- -- -- -- -- --

- La programmation orienté objet est un paradigme au sein de la programmation informatique. Il s'agit d'une représentation des choses, un modèle cohérant partager à travers différents langages qui permettent leur usage.

- Le c++ offre un mécanisme de classe rassemblant donnés et traitement, sont principe c'est de créer et faire interagir des briques logicielles que l'ont appelé objet, et l'objet représente une idée un concept ou on entité du monde physique.

- Cocos2d est un framework libre en Python, permettant de développer des applications ou des jeux vidéo. Des jeux comme FarmVille, Geometry Dash ou Angry Birds Fight! ont été développés avec Cocos2D.

- **But** : L'objectif principal de ce projet est de maitriser la programmation orientée objet par la mise en place d 'un jeu vidéo 2D, le jeu proposé s'appelle roller Splat, c'est un jeu qui a connu un grand succès dans les plateformes mobile.

-- Développement --

✓ Installation du Cocos 2D

- **On a installé cocos2d-x version 4-0 du site cocos2d-x.ong, et python version 2.7.16 du python.org. L'installation du python est nécessaire pour le fonctionnement du cocos2d. Et on a installé aussi CMake, est un système de construction logicielle multiplateforme qui permet cocos de créer plusieurs projets dans des différentes plateforme.**

- **Après l'installation, on va ouvrir l'invite de commande et on tape plusieurs commandes :**

  - `C:\Users\mrbra>python` **: vérifie si python est bien installé.**

  - `C:\Users\mrbra>cmake` **: vérifie si CMake est bien installé.**

  - `C:\Users\mrbra>cocos new LA3BA -l cpp -p com.sonarsystems.game` **: cette commande crée le dossier cocos avec le nom indiqué après new**

  - `C:\game-c++\LA3BA\proj.win32>cmake .. -G "Visual Studio 17-2022" -AWin32` **: Pour le fonctionnement de proj.win , tous les fichiers de cocos et visual studio**

✓ Les éléments de notre projets

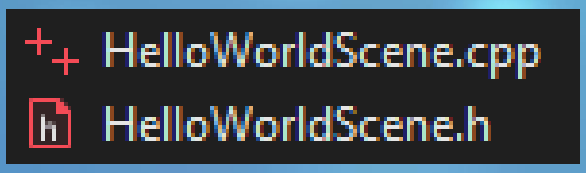
- **L'objectif de Roller Splat est de recouvrir le sol d'un labyrinthe avec de la peinture dans chaque niveau amusant et captivant.**
  On a programmé ce jeux 2d en utilisant la programmation orienté objet.
  D'abord on a utilisé les classes qui permettent de représenter des structure complexes dans un langage de programmation, qui a 2 composants (méthodes / attributs).

➢ AppDelegate :

- **Le délégué d'application est un objet qui reçoit des notifications lorsque l'objet UIapplications atteint certain état.**

- **AppDelegate.h : son rôle c'est de définit les méthodes et les fonctions concernant cocos.certain état.**
- **AppDelegate.cpp : chargé du définir la taille, couleur, structure, Background .... De l'élément cocos**

➢ Helloworldscene :



- **Helloworldscene.h : Pour la déclaration des fonctions qui crée les scènes**
- **Helloworldscene.cpp : le déplacement des éléments du menu**

➢ Helloworldscene :

- **On va déclarer les fonctions qui va nous créer des scènes, ces dernières c'est pour partir d'une page à l'autre.**
- **On va déclarer d'abord le menu avec ces éléments dans hellowordscene.h**

```
#ifndef __HELLOWORLD_SCENE_H__
#define __HELLOWORLD_SCENE_H__

#include "cocos2d.h"

class HelloWorld : public cocos2d::Scene
{
public:
    static cocos2d::Scene* createScene();
    virtual bool init();
    // a selector callback
    void menuCloseCallback(cocos2d::Ref* pSender);
    // implement the "static create()" method manually
    CREATE_FUNC(HelloWorld);
    //on a cr?e un sprite 'mySprite' qui contient l'arri?re-plan de notre jeu
    cocos2d::Sprite* mySprite;
    //ici on a d?clar? les ?l?ments de notre menu principal
    void Play(Ref* pSender);
    void Settings(Ref* pSender);
    void Help(Ref* pSender);
};
#endif // __HELLOWORLD_SCENE_H__
```

- **et après dans cpp on va afficher notre menu et déclarer des fonctions pour indiquer les éléments de ce dernier. Ainsi que déclarer les vecteurs qui sont responsable du déplacement des éléments du menu.**

```
#include "HelloWorldScene.h"
#include "GameScene.h"
#include "SettingsScene.h"
#include "GameOver.h"
USING_NS_CC;
Scene* HelloWorld::createScene()
{
    return HelloWorld::create();
}
// Print useful error message instead of segfaulting when files are not there.
static void problemLoading(const char* filename)
{
    printf("Error while loading: %s\n", filename);
    printf("Depending on how you compiled you might have to add 'Resources/' in front of filenames
```

```
// on "init" you need to initialize your instance
bool HelloWorld::init()
{
    ////////////////////////////
    // 1. super init first
    if (!Scene::init())
    {
        return false;
    }
```

```
    //D�claration de 'visibleSize' et 'origin' :
    auto visibleSize = Director::getInstance()->getVisibleSize();
    Vec2 origin = Director::getInstance()->getVisibleOrigin();

    //on a ajout� cette image qui pr�sente notre arri�re-plan :
    auto mySprite = Sprite::create("back ground2.png");
```

- **Et aussi on va remplir les éléments du menu avec leur sous menu.**

```
 //on a ajout� les boutons de notre menu sous formes des images png :
    auto menu_item_1 = MenuItemImage::create("play.png", "play.png",
CC_CALLBACK_1(HelloWorld::Play, this));
    auto menu_item_2 = MenuItemImage::create("settings.png", "settings.png",
CC_CALLBACK_1(HelloWorld::Settings, this));
    auto menu_item_3 = MenuItemImage::create("help.png", "help.png",
CC_CALLBACK_1(HelloWorld::Help, this));

    //ici on a cr�� notre menu qui contient ces boutons et on a les ajuster verticalement
    auto* menu = Menu::create(menu_item_1, menu_item_2, menu_item_3, NULL);
    menu->alignItemsVertically();
    this->addChild(menu);
```

➢ GameScene :

- **Dans GameScene.h, on va lié cocos avec notre scéne :**

```
#include "cocos2d.h"
```

- **Et après on va crée un class hérité par cocos 2d contient une scéne qui est une fonction de cocos :**

```
class GameScene : public Layer
{
```

- **Une même classe héritée va créer d'autre scène :**

```
public:
    static Scene* scene();
```

- **et dans GameScene.cpp on va déclarer les fonction utiles**

```
#include "GameScene.h"
#include "GameSprite.h"
#include "Header.h"
#include "Win.h"
#include "GameOver.h"
```

➢ GameSprite :

- **Et maintenant dans l'élément GameSprite.h on va créer toutes les probabilités qu'on peut utiliser dans la fonction Sprite qui se trouve déjà dans cocos.**

```
bool iscolored;
    GameSprite();
    virtual ~GameSprite();
    static GameSprite* gameSpriteinit();
    void changecolor(Color3B _color);
```

- **et dans GameSprite.cpp on va initialiser le constructeur et déclarer un destructeur vide, ainsi que créer des nouveau Sprite et remplir la fonction choisis.**

```
GameSprite::GameSprite(void) {
    this->iscolored = false;
};
GameSprite::~GameSprite(void) {

}
GameSprite* GameSprite::gameSpriteinit() {
    auto sprite = new GameSprite();
    sprite->initWithFile("triiiq.png");
    return sprite;
}
```

```
void GameSprite::changecolor(Color3B _color) {
    this->setColor(_color);
    this->iscolored = true;
}
```

➢ Pour créer le niveau 1 :

- **On va d'abord appeler cocos et sprite dans GameScene.h .**

```
#include "cocos2d.h"
#include "GameSprite.h"
```

- **Et déclarer les variables qui contrôle le déplacement du map, la direction des fonctions du cocos .**

```
private:
    Vector<GameSprite*> _spritevector;
    Vector<GameSprite*> _collidsprites;
    bool ismoving = false;
    int _direction_x = 0;
    int _direction_y = 0;
    int end;
    int laith = 0;
```

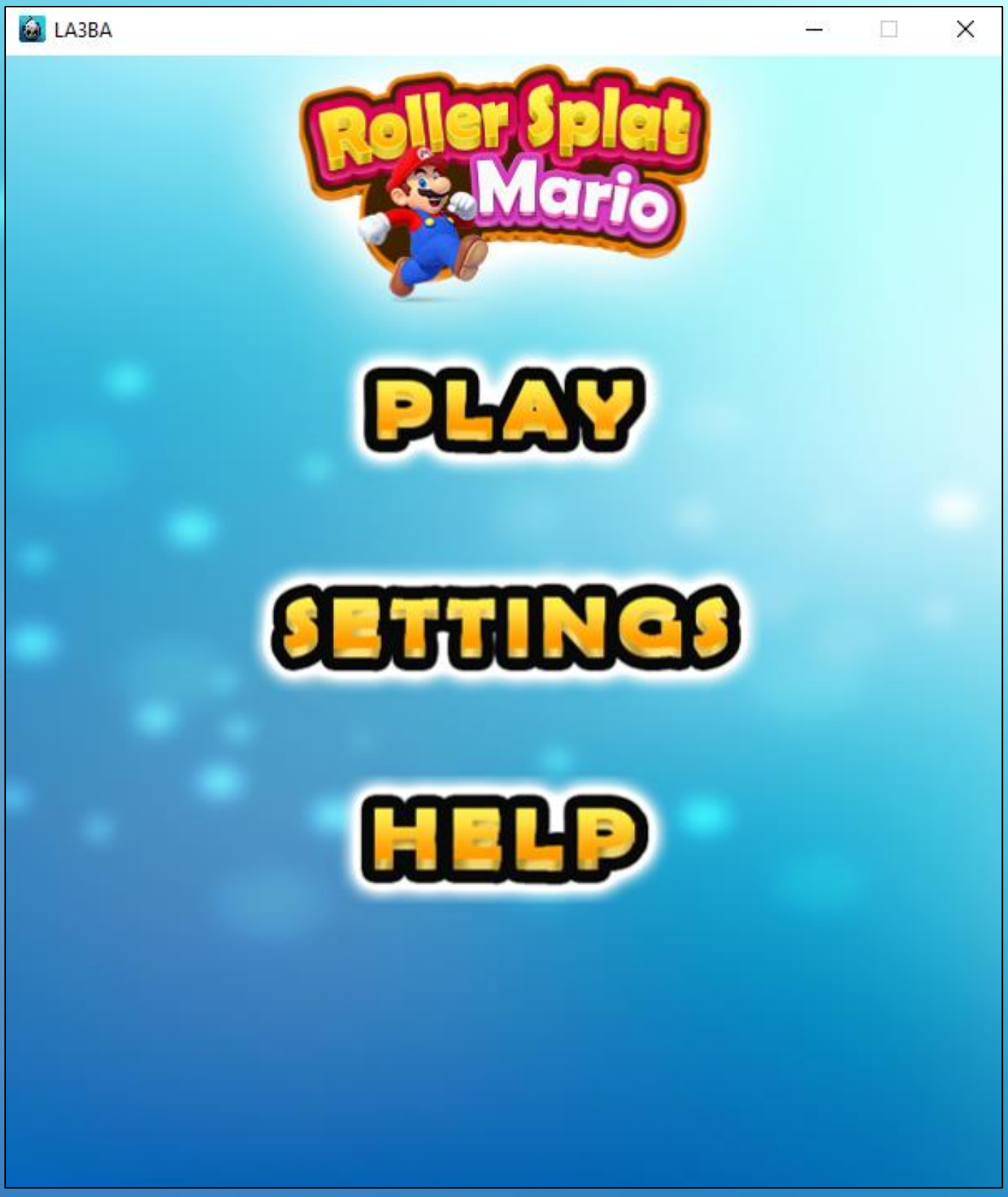
- **Après dans GameScene.cpp, on va positionner le mur du jeu et donner une direction 0 au vecteur x y pour commencer, ainsi qu'on va déclarer un destructeur vide, et donner une condition d'arrêt.**

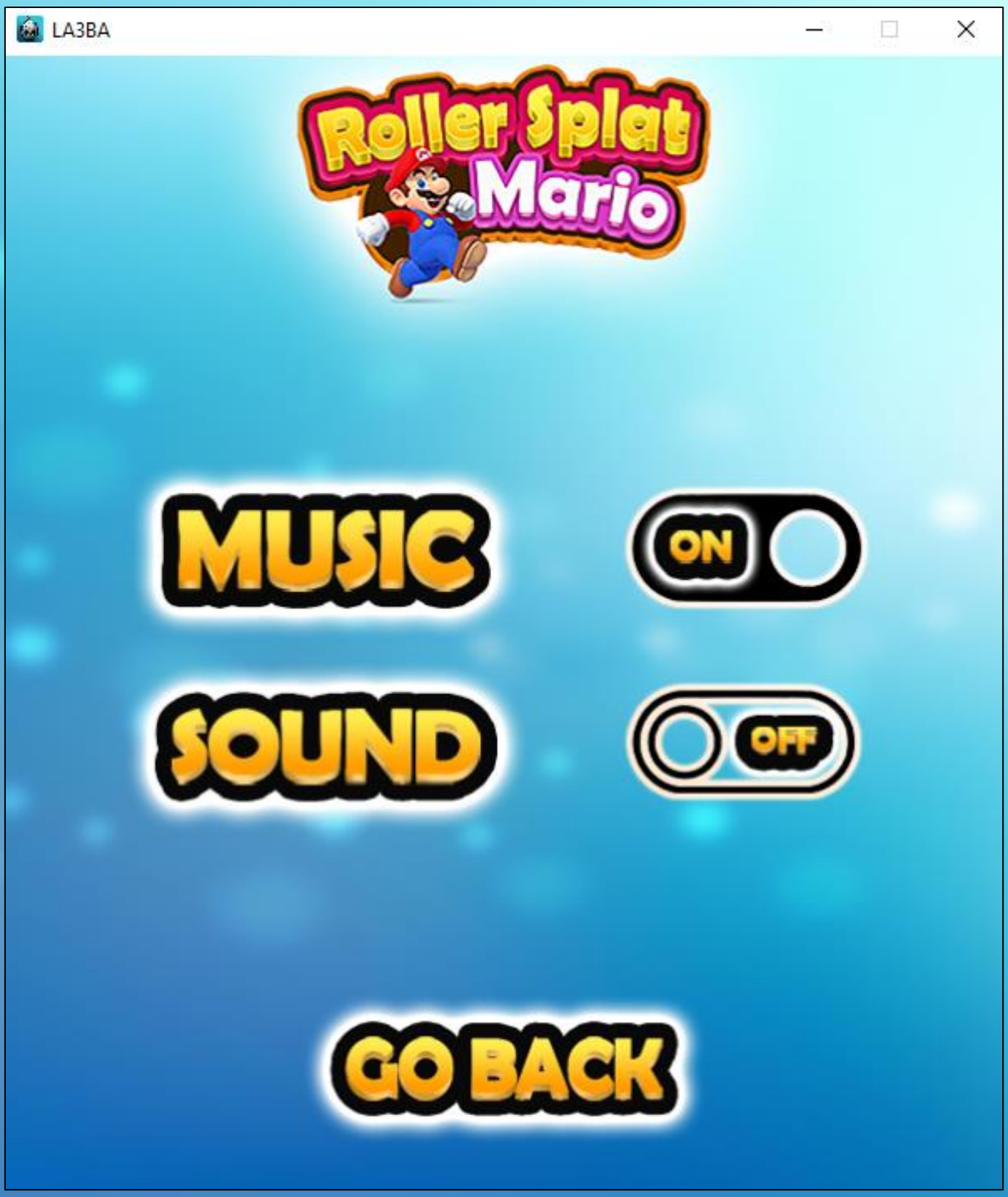
```
int arraylevel1[32][2] = { {2,2},{2,3},{2,4},{2,5},{2,6},{2,7},{2,8},{2,9},{2,10},
{2,11},{3,2},{3,3},{3,4},{3,5},{3,6},{3,7},{3,8},{3,9},{3,10},{3,11},{4,2},{5,2},{5,4},
{5,5},{5,6},{5,7},{5,8},{5,9},{5,10},{5,11},{5,12},{6,2} };
```

```
GameScene::~GameScene() {

}
void GameScene::update(float dt) {
    if (ismoving == true) {
        _ball->setPosition(_ball->getPosition().x + _direction_x, _ball->getPosition().y +
_direction_y);
    }
    for (auto _sprites : _collidsprites) {
        if (_direction_x == 0 && _direction_y == 1) {
            if (_ball->getPositionX() == _sprites->getPositionX() && _ball->getPositionY() ==
_sprites->getPositionY() - 10) {
                ismoving = false;
            }
}
        if (_direction_x == 0 && _direction_y == -1) {
            if (_ball->getPositionX() == _sprites->getPositionX() && _ball->getPositionY() ==
_sprites->getPositionY() + 10) {
                ismoving = false;
```

```
}
        }
        if (_direction_x == 1 && _direction_y == 0) {
            if (_ball->getPositionX() == _sprites->getPositionX() - 10 && _ball->getPositionY() ==
_sprites->getPositionY()) {
                ismoving = false;
            }
        }
        if (_direction_x == -1 && _direction_y == 0) {
            if (_ball->getPositionX() == _sprites->getPositionX() + 10 && _ball->getPositionY() ==
_sprites->getPositionY()) {
                ismoving = false;
            }
        }

    }
    for (auto _mysprites : _spritevector) {
        //ici lorqsue la balle se d?place , le couleur se change en arri?re de la balle :
        if (_ball->getPosition() == _mysprites->getPosition() && _mysprites->iscolored == false) {
            _mysprites->changecolor(Color3B(139, 139, 0)); //ici on a pr?cis? le couleur qui va
changer
            laith++;
        }
    }

    //ici lorsque le chemin est totalemnt color? , la scene va changer vers le scene de 'Win' qui
va nous passer au niveau suivant :
    if (laith == end) {
        count++;
        auto scene = Scene::create();
        auto layer = Win::create();
        scene->addChild(layer);
        director->getInstance()->replaceScene(scene);

        if (count == 3) {
            auto scene = Scene::create();
            auto layer = GameOver::create();
            scene->addChild(layer);
            director->getInstance()->replaceScene(scene);
        }

    }
}
```

-- Résultat Final --



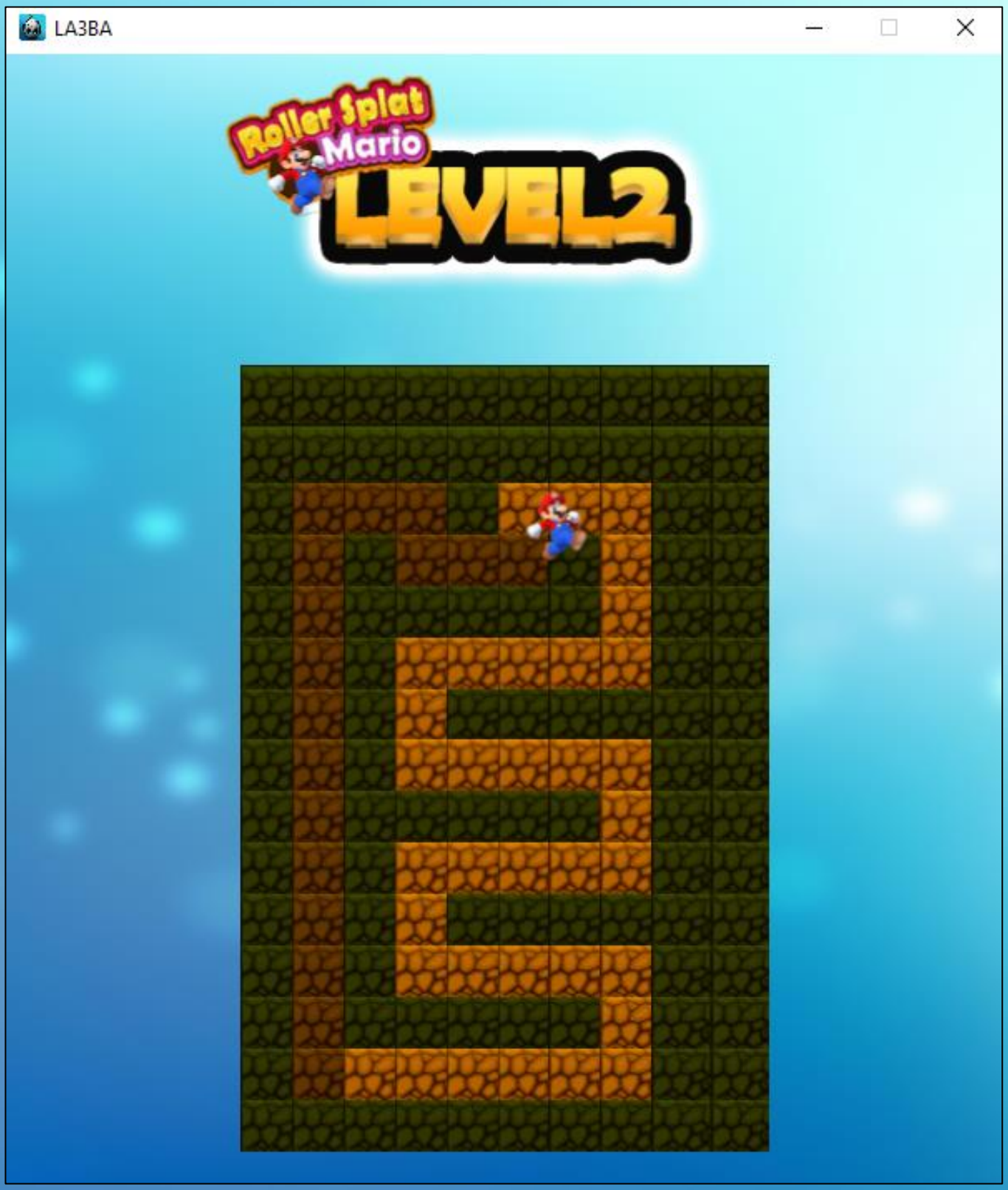

LA3BA
Roller Splat Mario
MUSIC
ON
SOUND
OFF
GO BACK

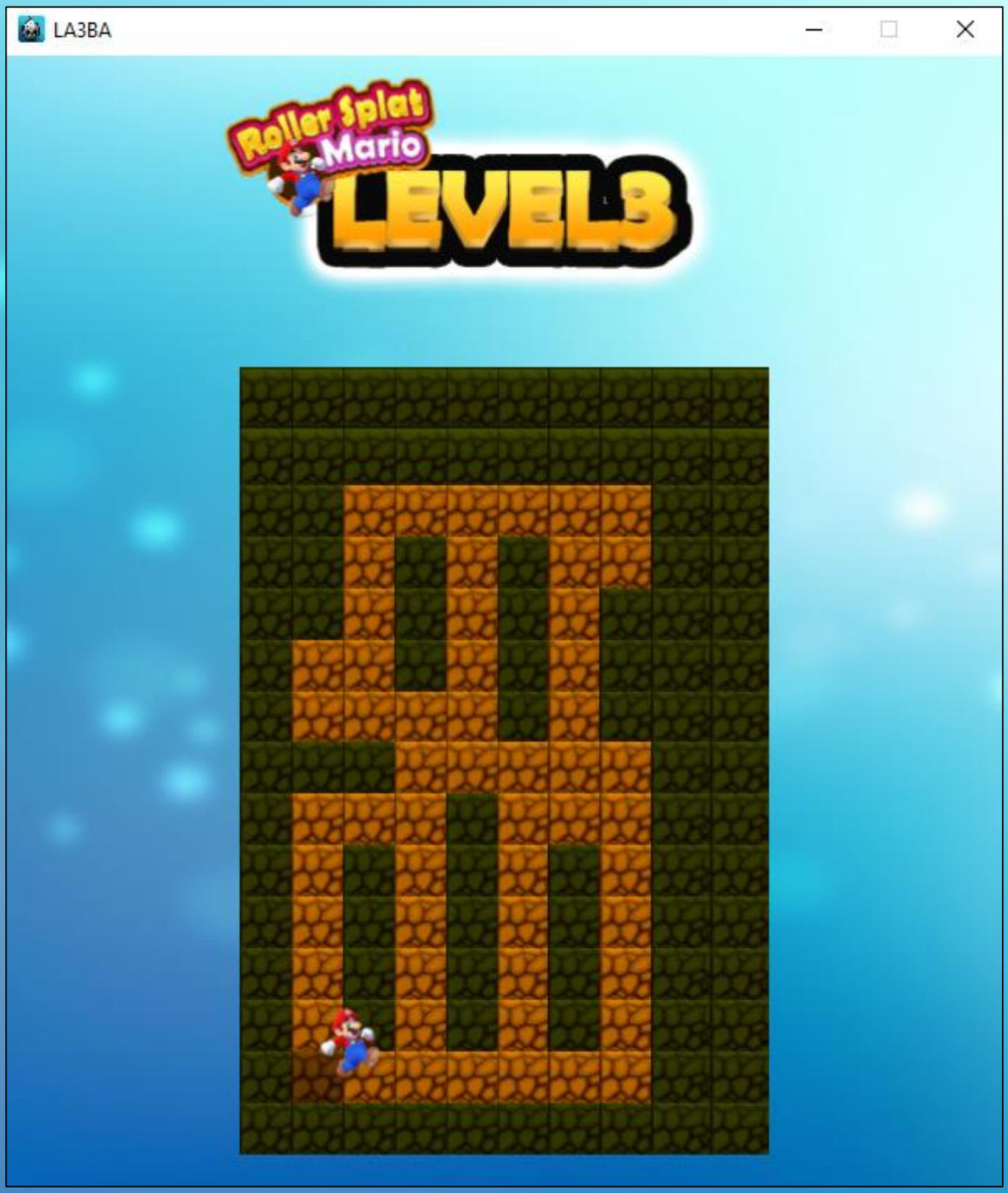

LA3BA
Roller Splat Mario
LEVEL 1

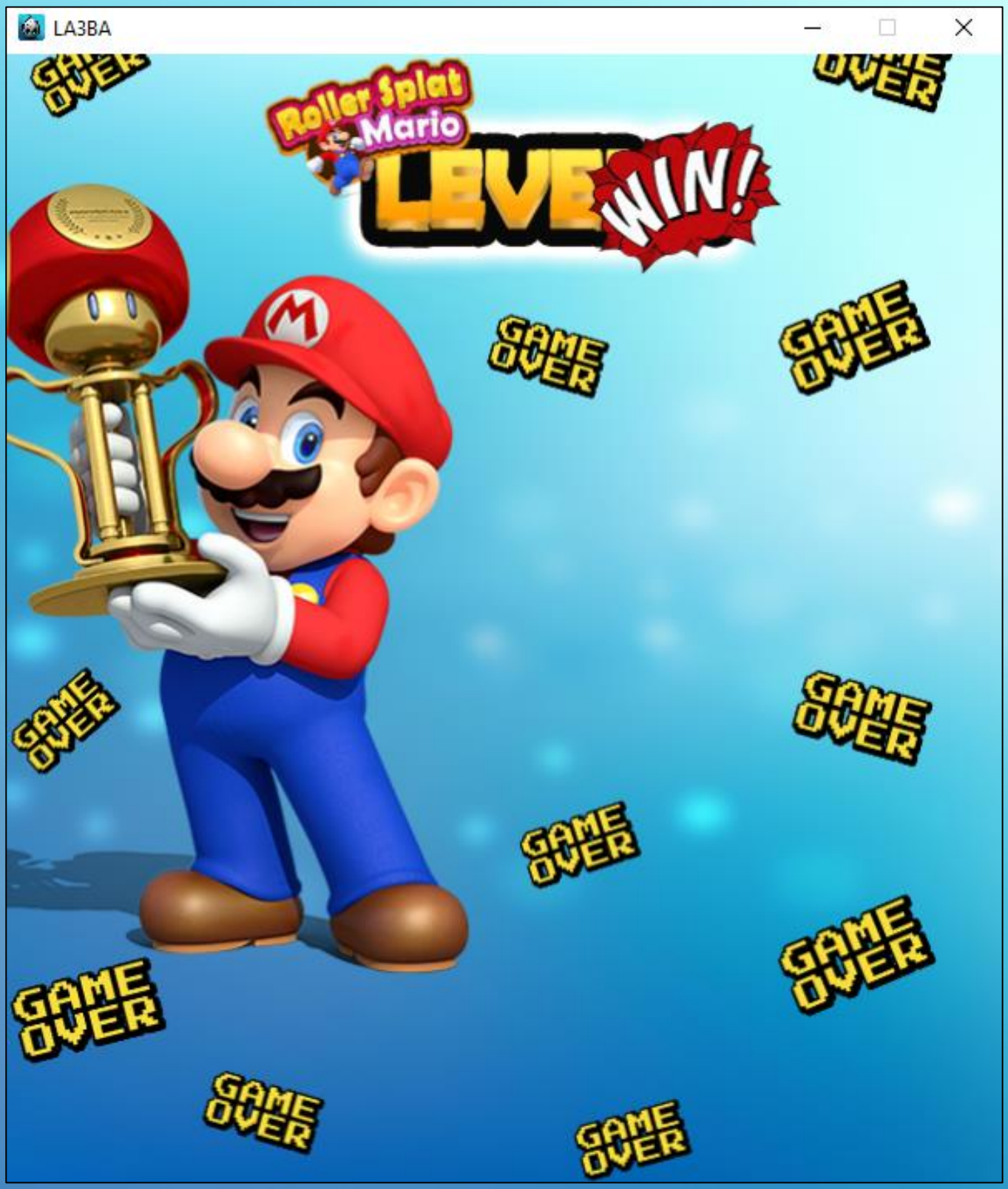

LA3BA
Roller Splat Mario
LEVEL
WIN!
NEXT

Roller Splat Mario

LA3BA




LA3BA
Roller Splat Mario
LEVEL3


LA3BA
Roller Splat Mario
LEVEL WIN!
GAME OVER
GAME OVER
GAME OVER
GAME OVER
GAME OVER
GAME OVER
GAME OVER
GAME OVER
GAME OVER
GAME OVER

-- Conclusion --

-- -- -- -- -- --

- **Tout au long de la préparation de notre jeu, nous avons essayé de pratiquer les connaissances requises durant notre cours du programmation orienté objet, et aussi notre connaissance sur ce qu'on a trouvé concernant cocos2d.**
- **L'objectif c'est de concevoir et programmer un jeu, il nous a donné la possibilité de maitriser et découvrir une nouvelle approche de la programmation..**

-- Bibliographie --

- https://www.youtube.com/watch?v=dYTfFeJdfdY
- https://www.youtube.com/watch?v=qXqgSNUf9Cc&list=PLRtjMdoYXLf4od_bOKN3WjAPr7snPXzoe
- VOTRE POLYCOPIE