Rapport du projet de fin du module C++

Réaliser par : Zine Ali

Sommaire :

Introduction

- Contexte et objectifs du projet.

  L’objectif principal de ce projet est de maitriser la programmation orientée objet par la mise en place d ‘un jeu vidéo 2D, un jeu de type d'arcade.

- Description du jeu et de ses principales fonctionnalités.

  Le jeu se déroule en scrollant horizontalement, chacun avec ses propres ennemis et défis. Pour progresser dans le jeu, Mario doit sauter sur les ennemis, éviter les obstacles et atteindre la fin de chaque niveau.

- Informations générales sur Godot engine.

Godot Engine est un moteur de jeu open source dédié au développement de jeux vidéo Il a été créé en 2007 par Juan Linietsky et Ariel Manzur et est maintenant 2D et 3D. développé et maintenu par la communauté de développeurs Godot.

Méthodologie

- Outils de travails (Godot engine, C++, Photoshop, Visual studio 2022).

- La bibliothèque GDNative

  GDNative est une bibliothèque qui permet à Godot, un moteur de jeu open source, de charger des bibliothèques C/C++ natives dynamiquement. Cela signifie que vous pouvez écrire du code C/C++ et l'inclure dans votre projet Godot, ce qui vous permet d'étendre

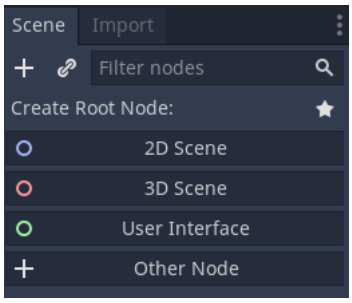
les fonctionnalités du moteur et d'optimiser les performances de votre jeu. GDNative vous donne également la possibilité de réutiliser du code existant, comme des bibliothèques de physique ou de rendu, dans votre projet Godot.

- Inclusion des librairies C++ sur Visual Studio 2022.

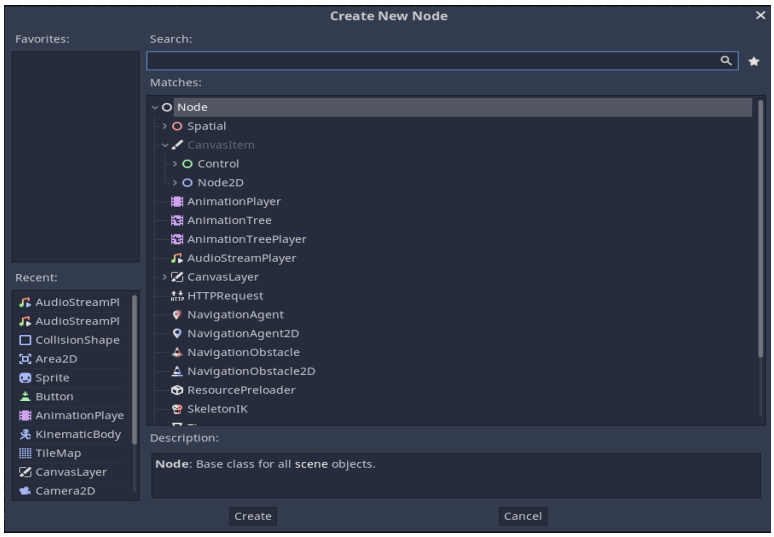
Ouvrez votre projet Visual Studio et allez dans le menu "Fichier" puis sélectionnez "Ajouter" et "Existing Item".Naviguez jusqu'à l'emplacement où se trouve la bibliothèque C++ que vous souhaitez inclure et sélectionnez-laCliquez sur "Ajouter" pour ajouter la bibliothèque C++ à votre projet.Allez dans le menu "Projet" puis sélectionnez "Propriétés".Dans la fenêtre "Propriétés du projet", allez dans l'onglet "Liens" et ajoutez l'emplacement des fichiers de la bibliothèque C++ dans la section "Chemin de la bibliothèque". Vous pouvez également ajouter le nom de la bibliothèque dans la section "Fichiers de bibliothèque".

- Processus et étapes de réalisation du projet.

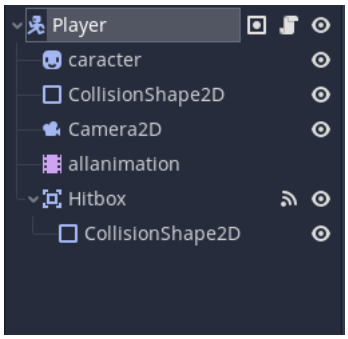
➔ Pour crée un nouveau projet il suffit de cliquer sur "Créer un projet" pour créer le projet. Godot ouvrira alors votre projet et vous pourrez commencer à travailler dessus.



-> Ensuite sélectionné 2D scène .



➔ Pour ajouter un nœud fils au nœud parents, on clique sur le bouton «+» La fenêtre ci-dessous apparait et nous donne un tas de choix de nœuds fils a ajouté comme (Sprite, collisionShape, camera …..).en suite on pourrait commence a ajoute notre caractère , lui ajouter une collisionShape ,une camera et une animation .



➔ En ce qui concerne la partie du script Player, on lui attache un script en cliquant sur Attach script ensuite en saisie la classe Player, en donne à ce script la bibliothèques GDnative.

Script du joueur dans le fichier Player.cpp :

```cpp
/*----------------Mouvements du joueur----------------*/
void Player::UpdateMotionFromInput()
{
    velocity = Vector2(0, 0);
    velocity.y += GRAVITY;

    //Pointeur "_all " de type ->AnimationPlayer

    AnimationPlayer* _all = (AnimationPlayer*)get_node("allanimation");

    Input* i = Input::get_singleton();

    //Pointeur "_Flip " de type ->Sprite

    Sprite* _Flip = (Sprite*)get_node("caracter");

    //Condition du Flip du joueur
    if (flip_H == false) {
        _Flip->set_flip_h(0);
    }
    else
        _Flip->set_flip_h(1);

    // input "ui_left" pour aller vers la gauche
    if (i->is_action_pressed("ui_left")) {
        _all->play("walk");
        velocity.x -= speed;
        flip_H = true;

    }
```

```cpp
    }
    //input "ui_right" pour aller vers la gauche
    else if (i->is_action_pressed("ui_right")) {
        _all->play("walk");
        velocity.x += speed;
        flip_H = false;

    }
    // Input "jump" pour que le joueur saute
    if (i->is_action_just_pressed("jump") && is_on_floor()) {
        velocity.y -= jump_Force;
    }
    // Si le joueur n'est pas sur la terre l'animation "jump" apparait
    if (!is_on_floor()) {
        _all->play("jump");

    }
    // Condition si le joueur dépasse une altitude de 1000 il refait la scéne
    if (get_global_position().y >= 1000)
    {
        get_tree()->reload_current_scene();
    }

    // Condition si le joueur est sur la terre l'animation "idle" apparait
    if (velocity.x == 0 && is_on_floor()) {
        _all->play("idle");
    }
}
```

```cpp
// Si le joueur entre dans la zone collisionShape du groupe enemy il meurt et refait la scéne
    void Player::_on_Hitbox_body_entered(Node* body)
    {
        if (body->is_in_group("enemy"))
        {
            get_tree()->reload_current_scene();
        }


    }
```

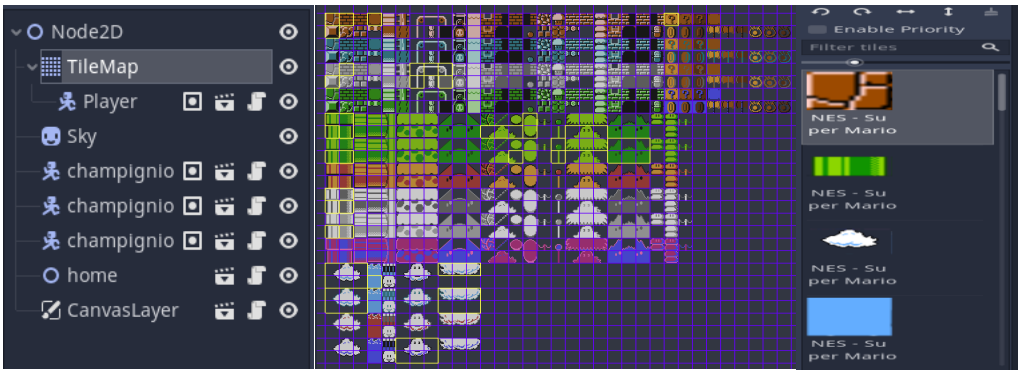
Script du joueur dans le fichier Player.h :

➔ Toutes les variables utilisées dans le fichier.cpp du Player.

```
class Player : public KinematicBody2D
{

// structure Godot
private:
    GODOT_CLASS(Player, KinematicBody2D)

// Methodes
public:
    static void _register_methods();
    void _init();
    void _on_Hitbox_body_entered(Node* body);
    void _process(float delta);


    Player();
    ~Player();


//  Variables du charactere
public:
    const int speed = 200;
    const int GRAVITY = 400;
    const int jump_Force = 8000;
    const int Max_Fall_speed = 100;
    bool flip_H = false;
    bool flip_V = false;


private:
```

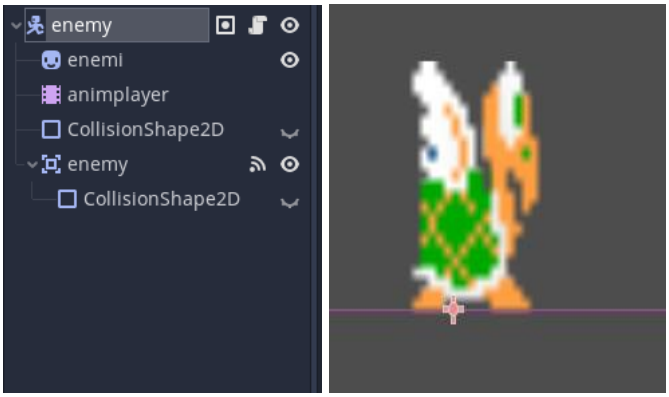
Création du Level :

Afin de construite notre level1 j'ai utilisé nœud fils appelé TileMap, Un TileMap est un objet de scène dans Godot qui permet de créer des niveaux de jeu à partir de tuiles. Les tuiles sont des carrés de texture qui peuvent être disposés de manière répétitive pour former des motifs complexes.



Création d'ennemie :

Ennemie 1



Script enemy 1 :

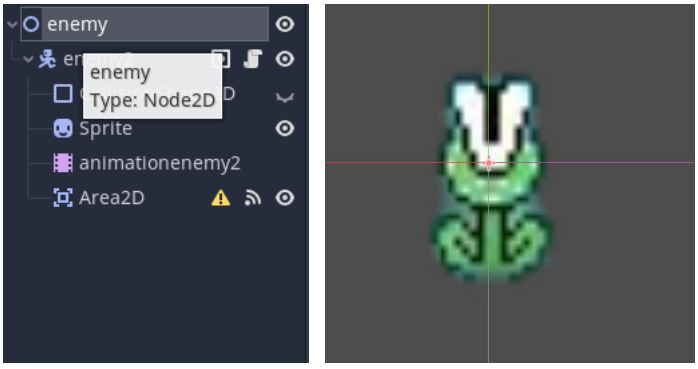
```
void enemy::_process(float delta)
{// Pointeur "_a" de type "AnimationPlayer"
    AnimationPlayer* _a = (AnimationPlayer*)get_node("animplayer");

    Sprite* _ene = (Sprite*)get_node("enemi");
    /*---------Mouvement enemy1-----------*/
    move_and_slide(velocity, Vector2(0, 0).UP);

    velocity = Vector2(0, 0);

    velocity.y += GRAVITY;

    velocity.x -= speed * dir;
    _a->play("moove");
    //Condition si l'enemy1 trouve un mur et change de direction

    if (is_on_wall()) {
        dir *= -1;
        if (dir < 0) {
            _ene->set_flip_h(0);
        }
        else
            _ene->set_flip_h(1);
    }
```

```
// si le joueur entre en collision avec la zone collisionShap  l'enemy1 meurt
    void enemy::_on_enemy_body_entered(Node * body) {

        if (body->is_in_group("player")) {

            queue_free();
        }
    }
```

➔ **Les variables utilisées déclarer dans le fichier.h et utilisées dans le fichier.cpp enemy1**

```
public:
    static void _register_methods();
    void _init();
    void _process(float delta);
    void _on_enemy_body_entered(Node* body);


    enemy();
    ~enemy();
public:
    // variables

    const int speed = 100;
    const int GRAVITY = 30;
     int dir = -1;
    bool hit_wall = false;
    bool hit_player = false;
    bool die = true;
    bool flip_H = false;
    bool flip_V = false;


private:
    Vector2 velocity;
};
```

Ennemie 2 :



Script enemy 2 :

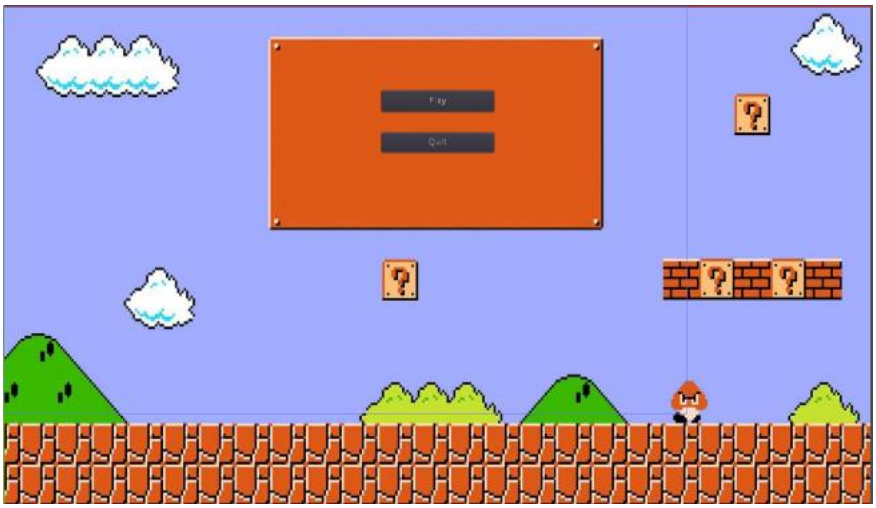
```
/*--------Mouvement enemy2-----------*/
void enemy2::_process(float delta)
{// Pointeur "_a" de type "AnimationPlayer"
    AnimationPlayer* _a = (AnimationPlayer*)get_node("animationenemy2");

   //Mouvements enemy2 suivant l'axe Y
    move_and_slide(velocity, Vector2(0, 0).UP);

    velocity = Vector2(0, 0);

    velocity.y -= speed * dir;

    velocity.y += GRAVITY;

    // Animation "eat"
    _a->play("eat");

}
// Si le joueur entre dans la zone collisionShape du groupe enemy il meurt et refait la scéne

void enemy2::_on_Area2D_body_entered(Node* body) {

    if (body->is_in_group("player")) {

        get_tree()->reload_current_scene();
    }
}
```

➔ **Les variables utilisées déclarer dans le fichier.h et utilisées dans le fichier.cpp enemy2**

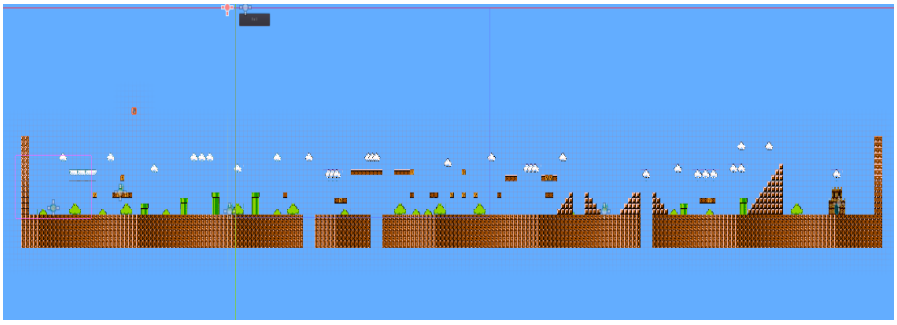
```cpp
namespace godot {
    class enemy2 : public KinematicBody2D
    {

        //  Structure Godot
    private:
        GODOT_CLASS(enemy2, KinematicBody2D)


    public:
        static void _register_methods();
        void _init();
        void _process(float delta);
        void _on_Area2D_body_entered(Node* body);


        enemy2();
        ~enemy2();
        // Constantes
    public:
        const int speed = 100;
        const int GRAVITY = 30;
        int dir = -1;



    private:
        Vector2 velocity;
    };
```

Résultats

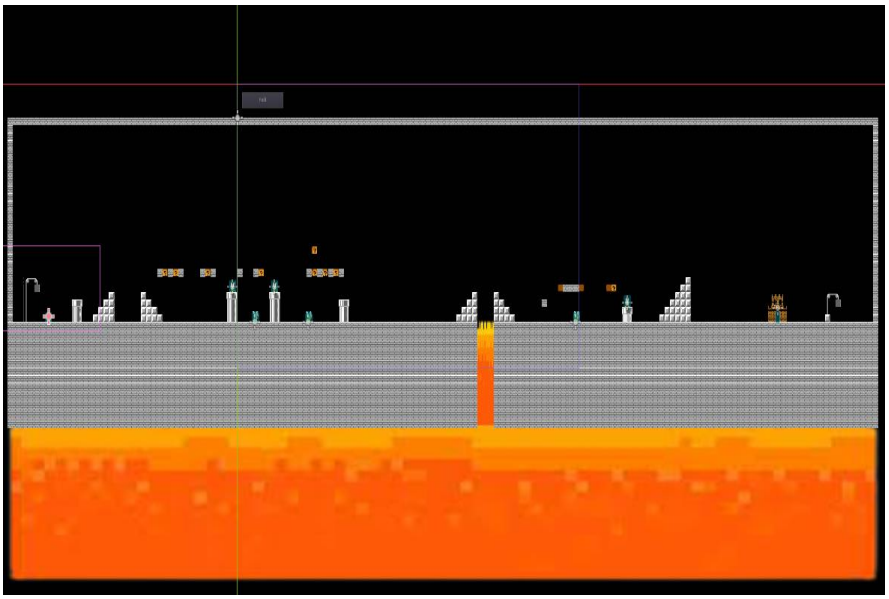
- **Menu :**



- **Level 1 :**



- **Level 2 :**



- **Scène de fin du jeu :**

Conclusion:

La réalisation de ce projet intitule game2d m’as parmi d’acquérir et de maitrise de nouvelles compétences dans la programmation oriente objet en C++, notamment la manipulation de nouveaux outils informatique, ce projet va me permettre de m’approfondir dans la programmation des application dans les années avenir .