Table of Contents

[**I.** **Cấu trúc class** 1](#_Toc53530987)

[**a.** **CGame** 1](#_Toc53530988)

[**b.** **IScene** 2](#_Toc53530989)

[**c.** **CplayScene: public IScene** 2](#_Toc53530990)

[**d.** **CGameObject:** 2](#_Toc53530991)

[**e.** **StaticObject:** 2](#_Toc53530992)

[**f.** **DynamicObject:** 2](#_Toc53530993)

[**g.** **Animator:** 2](#_Toc53530994)

[**h.** **CAnimations** (alias name: AnimationLib) 3](#_Toc53530995)

[**i.** **CAnimation** 3](#_Toc53530996)

[**j.** **CAnimationFrame** 3](#_Toc53530997)

[**k.** **Csprites** (alias name: SpriteLib) 3](#_Toc53530998)

[**l.** **CSprite** 3](#_Toc53530999)

[**m.** **Dinput** 3](#_Toc53531000)

[**II.** **Code-flow** 4](#_Toc53531001)

[**a.** **Mọi thứ bắt đầu ở hàm main** 4](#_Toc53531002)

[**b.** **Hàm Run** 4](#_Toc53531003)

[**c.** **Hàm Update** 5](#_Toc53531004)

[**d.** **Hàm Render** 6](#_Toc53531005)

[**III.** **Cách xem code** 6](#_Toc53531006)

1. **Cấu trúc class**
   1. **CGame**
      1. Mô tả: Là một Singleton quản lý các object của DirectX, các GameScene, Direct Input
      2. Chức năng:

* Cung cấp API để các sprite vẽ lên màn ảnh
* Load thông tin setting của game, thông tin về các Scene
* Quản lý các Scene
* Thực hiện lấy dữ liệu keyboard và mouse
  1. **IScene**
     1. Mô tả: Interface cho một Scene
  2. **CplayScene: public IScene**
     1. Mô tả: Quản lý các GameObject bên trong một PlayScene
     2. Chức năng:
* Load dữ liệu về Scene từ file txt bao gồm: Texture, Sprites, Animations, GameObjects, Map
* Lưu trữ thông tin các GameObject có trong Scene
* Thực hiện gọi hàm Update, Render cho các GameObject
  1. **CGameObject:** 
     1. Mô tả: Base class cho mọi GameObject trong Scene
  2. **StaticObject:**
     1. Mô tả: Các GameObject tĩnh, chỉ có một Sprite
  3. **DynamicObject:**
     1. Mô tả: Là những GameObject có khả năng di chuyển, gồm nhiều Animation khác nhau
     2. Chức năng:
* Chứa Animator hỗ trợ việc vẽ Animation
  1. **Animator:**
     1. Mô tả: Hỗ trợ DynamicObject vẽ các animation
     2. Chức năng:
* Lưu trữ frame và animation hiện tại mà GameObject đang sử dụng
* Toàn quyền quyết định việc update frame cho GameObject
* Gọi Animation để vẽ frame cần thiết cho DynamicObject
  1. **CAnimations** (alias name: AnimationLib)
     1. Mô tả: Singleton đóng vai trò là thư viện chứa thông tin tất cả các animation
     2. Chức năng:
* Lưu các animation theo ID
  1. **CAnimation**
     1. Mô tả: Chứa các AnimationFrame
     2. Chức năng:
* Vẽ các AnimationFrame dựa vào currentFrame mà Animator cung cấp
  1. **CAnimationFrame**
     1. Mô tả: Chứa thông tin về sprite và thời gian của sprite đó trong animation
  2. **Csprites** (alias name: SpriteLib)
     1. Mô tả: Singleton đóng vai trò là thư viện chứa thông tin tất cả các sprites
  3. **CSprite**
     1. Mô tả: Giữ thông tin về một Sprite
     2. Chức năng:
* Sử dụng API của class CGame để vẽ sprite của chính nó lên màn hình
  1. **Dinput**
     1. Mô tả: Là một Singleton lưu trữ trạng thái keyboard và mouse
     2. Chức năng:
* Sử dụng DirectInput để lấy các thông tin về trạng thái keyboard và mouse hiện tại

1. **Code-flow**
   1. **Mọi thứ bắt đầu ở hàm main**

INT WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,

PSTR lpCmdLine, INT nCmdShow)

{

HWND hwnd =  CreateGameWindow(hInstance, 800, 600);

game = CGame::GetInstance();

game->Init(hwnd);

game->Load(L"game-info.txt");

Run();

return 0;

}

Chương trình sẽ khỏi tạo một cửa sổ bằng function CreateGameWindow với 2 tham số phía sau lần lượt là width và height của cửa số muốn tạo

Sau đó ta gọi hàm Init của lớp CGame để tạo ra các object directx cần thiết cho game

Sau khi Init thì gọi hàm Load, chức năng của hàm dùng để load thông tin về các Scene của game và lưu các thông tin lại

Cuối cùng là hàm Run(), chứa vòng lặp chính của game

* 1. **Hàm Run**

int Run()

{

MSG msg;

int done = 0;

DWORD frameStart = GetTickCount();

DWORD tickPerFrame = 1000 / MAX\_FRAME\_RATE;

* msg: lưu trữ msg của hệ thống windows msg, dùng để biết khi người dùng nhấn nút esc game
* done: trạng thái của game
* frameStart: thời gian bắt đầu của frame trước đó, tính từ thời điểm chạy chương trình (đơn vị: ms)
* tickerFrame: thời gian chờ tối thiểu giữa 2 frame (đơn vị: ms)

while (!done)

{

if (PeekMessage(&msg, NULL, 0, 0, PM\_REMOVE))

{

if (msg.message == WM\_QUIT) done = 1;

TranslateMessage(&msg);

DispatchMessage(&msg);

}

DWORD now = GetTickCount();

// dt: the time between (beginning of last frame) and now

// this frame: the frame we are about to render

DWORD dt = now - frameStart;

CGame::GetInstance()->ProcessKeyboard();

if (dt >= tickPerFrame)

{

frameStart = now;

Update(dt);

Render();

}

else

Sleep(tickPerFrame - dt);

}

Mỗi lần lặp lại của vòng loop, chương trình sẽ gọi funtion ProcessKeyboard của CGame để thông báo lên class Dinput thực hiện lấy trạng thái keyboard và mouse

Mỗi vòng lặp sẽ gọi function Update và Render

* 1. **Hàm Update**

void Update(DWORD dt)

{

game->GetCurrentScene()->Update(dt);

}

Gọi Update cho Scene hiện tại đang play

Scene sẽ gọi Update cho các GameObject bên trong scene

* 1. **Hàm Render**

game->GetCurrentScene()->Render();

Scene sẽ lần lượt gọi Render của GameObject -> Animation -> Sprite

Sprite sẽ gọi class CGame để thực hiện vẽ sprite lên màn hình

1. **Cách xem code**

1. Hiểu được vòng lặp trong hàm main, tạm thời đừng quan tâm tới bên trong hàm Update và Render của PlayScene làm gì

2. Hiểu được hàm ProcessKeyboard của DInput

3. Hiểu được cấu trúc bên trong của class CGame gồm những thành phần nào, tạm thời đừng quan tâm tới cách implement các function của nó

4. Hiểu được cấu trúc của một PlayScene

5. Sau đó hãy tưởng tượng hàm Update của Scene mà chúng ta gọi ở vòng lặp hàm main sẽ lần lượt gọi Update cho các GameObject

6. Cuối cùng xem hàm Render của Scene, hãy cố gắng đi từ từ theo hàm này từ trên xuống theo hướng GameObject -> Animator -> Animation -> Sprite