Chuyển kiểu của file Game từ module ( 1 bọc các hàm) thành class cho nên các hàm Init, Run, End sẽ không còn static nữa mà phải tạo một instance MainGame và khởi tạo trước vòng lặp để gọi các hàm đó ra. Mục đích là vì hàm static k thể đặt làm hàm virtual nên phải có những chỉnh sửa như vậy, việc đặt virtual rất quan trọng vì các lớp Scene-n sẽ phải tái định nghĩa những hàm đó và các instance từ các scene đó là dạng các con trỏ nên cần binding during runtime (dùng virtual thay vì khai báo như thường rồi override lại sẽ chỉ binding when compiling).

Đổi tên SceneManager thành GameManager vì đơn giản sẽ chỉ khai báo những scene cần quản lí ở dạng Game (rồi chọn Scene nào mới tiến hành khởi tạo chồng lên con trỏ Game đó, Nam nếu làm game cờ vua r sẽ hiểu).

Các class Scene1, Scene2,.. sẽ kế thừa từ lớp game

Code thêm hàm lắng nghe key\_pressed( để làm thao tác nhảy mario)

Poll\_Keyboard();

ProcessKeyboard();

//check for left arrow

if (Key\_Hold(DIK\_LEFT))

mario->set\_x(mario->x() - 5);

if (Key\_Hold(DIK\_RIGHT))

mario->set\_x(mario->x() + 5);

Đây là hàm Key\_Hold

Muốn thêm vào những thao tác bấm phím thì trong ProcessKeyboard()

void Game::ProcessKeyboard()

{

//Collect all buffered events

DWORD dwElements = KEYBOARD\_BUFFER\_SIZE;

if (dikeyboard!=nullptr)

HRESULT hr = dikeyboard->GetDeviceData(sizeof(DIDEVICEOBJECTDATA), \_KeyEvents, &dwElements, 0);

//Scan through all data, check if the is pressed or released

for (DWORD i = 0; i < dwElements; i++)

{

int KeyCode = \_KeyEvents[i].dwOfs;

int KeyState = \_KeyEvents[i].dwData;

if ((KeyState & 0x80) > 0)

Key\_Pressed(KeyCode);

}

}

void Game::Key\_Pressed(int KeyCode)

{

switch (KeyCode)

{

case DIK\_SPACE:

mario->set\_y(mario->y() - 20);

break;

}

}

Đấy những cái case trong key\_pressed là chỗ nếu để thêm thao tác tương ứng với phím thoải mái đó

Dời hàm vẽ của sprite vào trong gameobject