ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**



**Họ và tên sinh viên nhóm tác giả đồ án:**

|  |  |  |
| --- | --- | --- |
| Sinh viên thực hiện: | | |
| STT | Họ tên | MSSV |
| 1 | Lê Huỳnh Thăng | 17520155 |
| 2 | Dương Thành Đạt | 17520040 |

**NINJA GAIDEN**

**Nhóm SE1 – Lớp SE102.J21**

Đồ án môn Nhập môn Phát triển game

GIÁO VIÊN HƯỚNG DẪN: Nguyễn Vĩnh Kha

Giới thiệu

Với sự hướng dẫn của thầy Nguyễn Vĩnh Kha, nhóm em đã sử dụng Visual Studio 2017 và Window SDK 10 cùng thư viện chính là DirectX9 SDK – June 2010 để hoàn thành đồ án mô phỏng game ninja gaiden. Bên cạnh đó, nhóm đã vận dụng kiến thức lập trình hướng đối tượng và sử dụng ngôn ngữ C++ để giải quyết các bài toán phát sinh khi phát triển game như va chạm, phân hoạch không gian, quản lí trạng thái của nhân vật,...

Trong báo cáo này sẽ mô tả tổng quan các lớp quan trọng, chức năng và cách vận hành của chúng trong game. Đồng thời, tự đánh giá một số ưu và khuyết điểm, cũng như một số khó khăn khi thực hiện đồ án

Các lớp chính

1. **DxFramework:**

Đây là framework để xử lí các vấn đề liên quan chủ yếu đến thư viện DirectX như xử lí âm thanh, hình ảnh và input

* 1. *dxgraphics:*

Xử lí các vấn đề về hình ảnh:

* int Init\_Direct3D(HWND, int, int, int): khởi tạo các biến cần dùng cho việc tải hình ảnh (vd: LPDIRECT3D9, LPDIRECT3DDEVICE9, LPDIRECT3DSURFACE9, LPD3DXSPRITE)
* LPDIRECT3DSURFACE9 LoadSurface(string, D3DCOLOR): tải hình ảnh lên trên surface
* LPDIRECT3DTEXTURE9 LoadTexture(string, D3DCOLOR): tải hình ảnh lên để vẽ trên spritehandler
  1. *dxinput:*

Xử lí các vấn đề về input:

* int Init\_DirectInput(HWND): khởi tạo các biến cần dùng cho việc xử lí input (vd: LPDIRECTINPUT8, LPDIRECTINPUTDEVICE8, LPDIRECTINPUTDEVICE8)
* int Init\_Keyboard(HWND): khởi tạo các định dạng cho bàn phím
* void Poll\_Keyboard(): lấy trạng thái của bàn phím
* int Key\_Down(int): trả về xem phím cần kiểm tra có được nhấn
* int Key\_Up(int): trả về xem phím cần kiểm tra có được thả
* void Kill\_Keyboard(): xóa và trả quyền sử dụng bàn phím cho hệ thống
* void Poll\_Mouse(): lấy trạng thái của chuột
* int Init\_Mouse(HWND): khởi tạo các định dạng cho chuột
* int Mouse\_Button(int) : trả về xem phím chuột cần kiểm tra có được nhấn
* int Mouse\_X(): lấy vị trí X của chuột
* int Mouse\_Y(): lấy vị trí Y của chuột
* void Kill\_Mouse(): xóa và trả quyền sử dụng chuột cho hệ thống
  1. *dxsound:*

Cùng với hai thư viện có sẵn là DSUtil và dxutil xử lí các vấn đề âm thanh

* int init\_SoundManager(HWND) : khởi tạo các biến cần dùng cho việc xử lí âm thanh (vd: CsoundManager)
* int addSound(string path): thêm âm thanh
* void playSound(SoundType name, bool isLoop = false, LONG volume = 2L): chạy âm thanh
* void stopSound(SoundType name): dừng âm thanh

1. **Original**:

Là những thư viện gốc – gồm những đối tượng cơ sở quan trọng để phát triển game nói chung

1. *Camera:*

Là một đối tượng đi theo nhân vật để “quay lại” những cảnh vật xung quanh nó

* D3DXVECTOR3 WorldToView(D3DXVECTOR3 pos): chuyển tọa độ trong thế giới thực thành tọa độ trong game
* void Update(int x): cập nhật lại vị trí của camera

1. *Grid:*

Dùng để phân hoạch không gian, giúp giảm chi phí tính toán trong quá trình xử lí. Ở đây, Grid chỉ mới áp dụng cho phân hoạch tường và đất (Wall)

* void GridClear(): xóa Grid
* void Add(Wall\* wall): thêm một Wall vào trong Grid
* vector<Wall\*> WallOut(RECT rect): lấy ra các Wall nằm trong vùng cần xét xử lí

1. *Sprite:*

Dùng để lưu thông tin của hình cần vẽ và vẽ nó lên spritehandler. Dùng cho các Sprite có kích thước giống nhau

* void Render(Camera camera): vẽ texture lên spritehandler với tâm là trọng tâm của hình
* void Render(Camera camera, RECT renderRect): vẽ texture lên spritehandler với tâm là ở giữa phía dưới chân hình ứng với hình chữ nhật bao khuôn đã cho
* void Render(Camera camera, RECT renderRect, D3DXVECTOR3 center): vẽ texture lên spritehandler với tâm đã cho ứng với hình chữ nhật bao khuôn đã cho
* virtual void KillSprite(): xóa các dữ liệu cần thiết trước khi hủy Sprite
* int GetWidth(): lấy chiều dài của 1 ô Sprite nhỏ
* int GetHeight(): lấy chiều cao của 1 ô Sprite nhỏ
* Box ToBox(): đóng bao Sprite thành hình chữ nhật, thường dùng để xét va chạm
* virtual void RectUpdate(): chuyển sang ô Sprite khác

1. *Tiles:*

Sprite của nhân vật không phải lúc này cũng cùng một kích cỡ. Dùng Tiles kèm với phương thức Render có truyền tham số hình chữ nhật trong Sprite sẽ khắc phục vấn đề này. Tiles có sử dụng thư viện hỗ trợ đọc file .xml sẽ được giới thiệu ở mục SupportFunction

* RECT getRectLocation(int ID): lấy hình chữ nhật tại vị trí sprite thứ ID
* void TilesClear(): xóa các dữ liệu cần thiết trước khi xóa Tiles

1. *State:*

Dùng để quản lí các trạng thái của nhân vật, thường là nhân vật chính. Bằng cách kế thừa và đa hình, việc quản lí các trạng thái này sẽ dễ hơn. Ở đây chỉ giới thiệu lớp cha là lớp State:

* virtual int HandleInput(StateInfo\* info) = 0: xử lí input và xử lí các ngoại lệ để chuyển trạng thái
* virtual void UpdateVelocity(LPSprite sprite): cập nhật vận tốc của nhân vật theo trạng thái
* virtual void UpdateStatus(LPSprite sprite) = 0: cập nhật vị trí của nhân vật theo trạng thái
* virtual void UpdateCollision(LPSprite sprite, Wall\* wall, Box Ryu) = 0: cập nhật va chạm của nhân vật theo trạng thái
* virtual Box ToBoxSword(LPSprite sprite): đóng bao vũ khí nhân vật thành hình chữ nhật, thường dùng để xét va chạm
* virtual Box ToBox(LPSprite sprite) : đóng bao nhân vật thành hình chữ nhật, thường dùng để xét va chạm

1. *ScoreBoard:*

Bảng điểm và những thông tin cần thiết khi chơi game

* void Update(int Stage, int lives, int RyuMP, bool TimeStop) : cập nhật ScoreBoard
* void Render(Camera camera, int RyuHP, SkillDefine skillType): vẽ ScoreBoard
* void ScoreBoardDelete(): xóa các dữ liệu cần thiết trước khi xóa ScoreBoard

1. *Stage:*

Dùng để lưu các dữ liệu liên quan đến màn chơi (vd: đường dẫn đến các file source, vị trí bắt đầu của nhân vật chính)

1. **SupportFunction:**

Bao gồm những thư viện hỗ trợ cho các công xử lí liên quan. Các thư viện này chỉ gồm các phương thức và không chia class

1. *Collision:*

Dùng để kiểm tra va chạm và chồng lấn hình

* float SweptAABB(Box box1, Box box2, float &normal\_x, float &normal\_y): kiểm tra va chạm của 2 hình chữ nhật và trả về thời gian va chạm và hướng va chạm
* bool OverlappedBox(Box box1, Box box2): kiểm tra 2 hình chữ nhật có chồng lấn lên nhau không

1. *TinyXml:*

Đây là một thư viện hỗ trợ tìm thấy trên mạng để hỗ trợ Tiles đọc các file .xml. Cùng với các phần mềm cắt và tạo SpriteSheet, đây là những bộ đôi khá mạnh trong việc hỗ trợ vẽ Sprite. Những phần mềm này sẽ được giới thiệu ở mục các phần mềm hỗ trợ.

1. *debug:*

Thư viện này hỗ trợ trong việc debug. Khác với các chương trình console bình thường, việc debug trong game gặp khá nhiều khó khăn vì việc xử lí đầu vào sẽ khiến game chạy trong chế độ debug (có breakpoint) khác với việc chạy game bình thường. Chính vì vậy, việc viết những thông tin cần quan sát ra Output (debug của chương trình) sẽ khiến việc sửa lỗi dễ hơn.

* void DebugOut(wchar\_t \*fmt, ...): sử dụng như printf, để viết ra Output những thông tin cần quan sát