**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM**

**ĐỒ ÁN MÔN HỌC**

**GAME RẮN SĂN MỐI BẰNG SDL2**

Ngành: **CÔNG NGHỆ THÔNG TIN**

Chuyên ngành: **CÔNG NGHỆ PHẦN MỀM**

Giảng viên hướng dẫn :Lê Viết Tuấn

Sinh viên thực hiện :

Đoàn Xuân Trường MSSV:1511061422

Mạch Huệ Mẫn MSSV:1511061220

Nguyễn Minh Trung MSSV:1511061418

Nguyễn Bình Phương Giang MSSV:1511061056

Lớp: 15DTH13

TP. Hồ Chí Minh, 2018

MỤC LỤC

[CHƯƠNG 1: CƠ SỞ LÝ THUYẾT 3](#_Toc517047512)

[Thư viện lập trình đồ họa SDL 2 3](#_Toc517047513)

[**Gi**ớ**i thi**ệ**u** 3](#_Toc517047514)

[**Cài đặt SDL 2** 3](#_Toc517047515)

[**Sử dụng SDL 2** 5](#_Toc517047516)

[**a. Khởi tạo môi trường đồ họa** 5](#_Toc517047517)

[**b. Đối tượng Renderer** 7](#_Toc517047518)

[**c. Một số thao tác dựng hình cơ sở** 8](#_Toc517047519)

[**d. Xử lý sự kiện** 10](#_Toc517047520)

[**e. Xử lý va chạm đơn giản** 11](#_Toc517047521)

[CHƯƠNG 2: GAME GREEN SNAKE 11](#_Toc517047522)

[**2.1 Giao diện chính** 12](#_Toc517047523)

[**2.2 Cách chơi** 12](#_Toc517047524)

[**2.3 Các thư viện sử sụng trong game** 14](#_Toc517047525)

[CHƯƠNG 3: KẾT LUẬN 14](#_Toc517047527)

# CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

## Thư viện lập trình đồ họa SDL 2

## **Gi**ớ**i thi**ệ**u**

**SDL 2 (Simple DirectMedia Layer version 2) là** một thư viện lập trình có khảnăng trừu tượng hóa các phần cứng đồ họa, âm thanh, thiết bị I/O**.** Sử dụng SDL, các lập trình viên có thể xây dựng các ứng dụng giải trí, trò chơi điện tử, ứng dụng truyền thông đa phương tiện trên nhiều hệ điều hành khác nhau như Linux, macOS, Windows…

Thư viện lập trình SDL 2 được chia thành nhiều hệ thống nhỏ hơn (sub**-**system) như **Video, Audio, CD-**ROM, Timer…Ngoài các hệ thống con cơ bản này, **SDL 2 còn cung** cấp thêm các thư viện chính thức riêng biệt khác như:

* SDL\_image: cho phép đọc các định dạng ảnh phổ biến như JPEG, PNG, BMP…
* SDL\_mixer: cung cấp các hàm audio cho phép thực hiện việc hòa âm (mixing), đọc các tập tin audio như OGG, MP3…
* SDL\_net**:** hỗ trợ lập trình mạng.
* SDL\_ttf**:** hỗ trợ sử dụng các font chữ TrueType.
* SDL\_rtf**:** hỗ trợ hiển thị tài liệu dạng Rich Text Format (RTF).

**SDL 2** được xây dựng bằng ngôn ngữ lập trình C, vì vậy C cũng chính là ngôn ngữ được hỗ trợ chính thức bởi thư viện lập trình này, ngoài ra ta cũng có thể sử dụng **SDL 2** với C++**.**

## **Cài đặt SDL 2**

**SDL 2** hiệntại hỗ trợ các trình biên dịch như Visual Studio và MinGW.Trong nội dung bài hướng dẫn này, ta chỉ quan tâm đến việc cài đặt SDL 2.0 vào Visual Studio, cụ thể là phiên bản 2015. Đối với các phiên bản x`cũ hoặc mới hơn, quá trình cài đặt thư viện cũng tươngtự, chỉ có một số điều chỉnh nhỏ do sự khác biệt về bố trí thư mục cài đặt của các phiên bản Visual Studiolà không giống nhau**.**

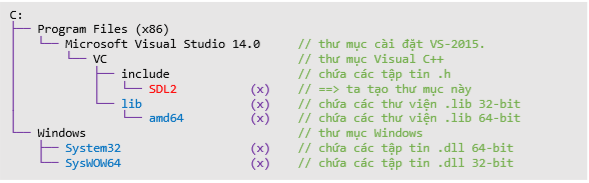
Việc cài đặt SDL **2** vào hệ thống sẽ giúp cho việc xây dựng, biên dịch và liên kết **các** dự án phần mềm **có** sử dụng thư viện nàyđược thuận tiện và dễ dàng.

Trước tiên, ta cần tải về bộ thư viện lập trình **SDL 2** trực tiếp từ trang chủ tạiđịa chỉ: https://www.libsdl.org/download-2.0.php**.** Tại đây, ta có thể tải về mã nguồn của thư viện, các thư viện liên kết động để sử dụng với các ứng dụng SDL **2** và bộ thư viện hỗ trợ phát triển ứng dụng (***development library*).** Do cần sử dụng SDL **2** để xây dựng các ứng dụngtrên môi trường Visual Studio**, ta** tải về tập tin chứa thư viện phát triển của **SDL** phiên bản **2.0.8:** SDL2-devel-2.0.8-VC.ziptrên nền **Windows.**

Sau khi tải về tập tin trên, giải nén nội dung tập tin này vào một thư mục và thực hiện các bước cài đặt như sau:

* Giả sử Visual Studio 2015 được cài đặt trên ổ đĩa C:của hệ điều hành tại thư mục C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC**.** Khi đó thư mục chứa các tập tin header của **Visual C++** sẽđược đặt trong thư mục includevà thư mục chứa các thư viện liên kết sẽ được đặt trong thư mục lib**, (Hình 1)**. Thư mục Windowscũng được cài đặt trên ổ đĩa C:của hệthống.Các thư mục cần thao tác **trong quá** trình cài đặt thư viện SDL 2 được đánh dấu (x)**.**
* Tạo thư mục SDL2 bên trong thư mục includecủa **Visual C++,** sau đó **copy toàn** bộ các tập tin header (.h) trong thư mục includecủa thư viện **SDL 2** vừa giải nén vào thư mục include/SDL2vừa tạo. **Thao tác này cho phép ta** sử dụng các tập tin header của SDL**2 thông qua** chỉ thị #includeở đầu tập tin mã nguồn của mọi project phần mềmmà không cần phải sao chép chúng vào từng thư mục riêng biệt của project đó**.**
* Tương tự, copy tất cả các tập tin .libtừ thư mục lib/x86của thư viện SDL **2** đã giải nén vào thư mục libcủa **Visual C++.** Copy tất cả các tập tin Copy tất cả các tập tin .libtừ thư mục lib/x64vào thư mục lib/amd64của Visual C++.Để liên kết các thư viện SDL **2** vào dự án phần mềm, ta sử dụng chỉ thị #pragmatrực tiếp trong tập tin mã nguồn của dự án**.**

Copy tập tin SDL2.dlltừ thư mục lib/x86vào thư mục Windows/SysWOW64và tập tin SDL2.dlltừ thư mục lib/x64vào thư mục Windows/System32**.**



**Hình 1. Cấu trúc thư mục Visual Studio 2015 và Windows trên ổ đĩa C:**

## **Sử dụng SDL 2**

Cách thức vận dụng các hàm API của thư viện SDL **2** được giới thiệu chi tiết tại trang wiki của thư viện nàytại địa chỉ: https://wiki.libsdl.org. Bài hướng dẫn này chỉ trình bày sơ lược các hàm API sẽ được sử dụng trong các bài tập sẽ được giới thiệu mà thôi.

### **a. Khởi tạo môi trường đồ họa**

Để khởi tạothư viện, xây dựng, hủy bỏ các cửa sổ dựng hình và chấm dứt phiên làm việc với SDL **2**, ta sử dụng cáchàm sau đây**:**

int SDL\_Init(Uint32 flags)

Khởi tạo thư viện SDL **2**. Hàm này thực chất sẽ thực hiện các công việc khởi tạo cho từng thư viện con ứng với giá trị nhận được từ tham số flags**.** Giá trị thường hay được sử dụng của tham số flags **là** SDL\_INIT\_EVERYTHING**. Ta có th**ể **k**ế**t h**ợ**p vi**ệ**c kh**ở**i t**ạ**o nhi**ều thư việ**n con thông qua toán t**ử|**. Ví d**ụ**, n**ế**u ta ch**ỉ **c**ầ**n kh**ở**i t**ạ**o** thư việ**n con x**ửlý đồ **h**ọ**a và âm thanh, giá tr**ịflags **s**ẽ **là** SDL\_INIT\_VIDEO | SDL\_INIT\_AUDIO**. Hàm này tr**ả **v**ề **tr**ị **0 n**ế**u vi**ệ**c kh**ở**i t**ạo đượ**c th**ự**c hi**ệ**n thành công.**

* void SDL\_Quit(void)

Sử dụng hàm này để dọn dẹp và thu hồi tài nguyên đã cấp phát cho các thư viện con và chấm dứt phiên làm việc với SDL **2.** Hàm này không nhận tham số và cũng **k**hông trả ra bất kỳ giá trị nào.

* SDL\_Window \* SDL\_CreateWindow( const char \* title, int x, int y, int w, int h,

Uint32 flags)

Hàm này thực hiện việc tạo ra một cửa sổ SDL **2** với các tham số như sau:

* title: Tiêu đề của cửa sổ SDL.
* x, y: tọa độ của cửa sổ.Có thể là một giá trị cụ thể hoặc một trong các hằng số SDL\_WINDOWPOS\_CENTEREDhoặc SDL\_WINDOWPOS\_UNDEFINED**.**
* w, h: Kích thước chiều rộng và chiều cao của cửa sổ, tính theo pixel.

flags**:** Cờ quy định trạng thái cửa sổ sẽ được tạo ra, một số giá trị có thể sử dụng **bao** gồm, ta có thể kết hợp các giá trị này thông qua toán tử |**:**

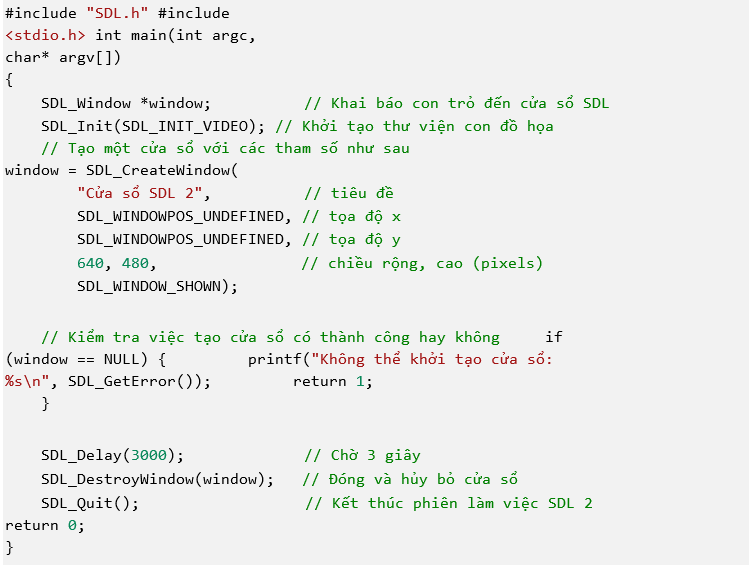
* + SDL\_WINDOW\_SHOWN**:** Cửa sổ được tạo ra sẽ xuất hiện trên màn hình, giá trị này là mặc định.
  + SDL\_WINDOW\_FULLSCREEN**:** Cửa sổ được tạo ra sẽcó độ phân giải toàn màn hình.
  + SDL\_WINDOW\_FULLSCREEN\_DESKTOP**:** Cửa sổ được tạo ra sẽ chiếm toàn bộ màn hình **desktop**, không có thanh tiêu đề, không có viền cửa sổvà các nút chức năng**.**
  + SDL\_WINDOW\_OPENGL**:** Cửa sổ được tạo ra sẽ hỗ trợ dựng hình với OpenGL.

Hàm này khi thực hiện thành công sẽ trả về một con trỏ đến cửa sổ đã được tạo ra. Ngược lại, một giá trị NULLsẽ được trả về.

 void SDL\_DestroyWindow(SDL\_Window \* wnd)

Tagọi hàm này để hủy bỏ cửa sổ SDL đã được tạo bởi hàm SDL\_CreateWindow, trỏ đến bởi tham số wnd**.** Cửa sổ sẽ được hủy bỏ **và** thu hồi các tài nguyên đã cấp phát.

Ví dụ:



Lưu ý:với các ứng dụng có dùng thư viện SDL **2, hàm** mainphải có dạng như **sau:**

int main(int argc, char\* argv[]) { // Cho phép nhận tham số trên dòng lệnh // . . . return 0;

}

### **b. Đối tượng Renderer**

Để thực hiện các thao tác dựng hình **2D** với SDL **2**, ta cần có một đối tượng quản lý việc dựng hình **này** trên các cửa sổ SDL đã được tạo ra.Đối tượng quản lý việc dựng **hình này** có tên gọi là SDL\_Renderervà được liên kết với một cửa sổ thông qua con trỏ bởi đối tượng SDL\_Window**.** Khi đó mọi thao tác dựng hình với đối tượng SDL\_Renderersẽ được kết xuất ra trên cửa sổ của đối tượng SDL\_Window**.** Dưới đây là một số hàm **API** liên quan đến việc **t**ạo và hủy đối tượn SDL\_Renderer**:**

 SDL\_Renderer \* SDL\_CreateRenderer(SDL\_Window \* wnd, int index, Uint32 flags): Tạo đối tượng quản lý ngữ cảnh dựng hình 2D cho một cửa sổ SDL.Ý nghĩa của **c**ác tham số được mô tả như sau**:**

* wnd**:** Con trỏ đến một cửa sổ SDL đã có**.**
* index**:** chỉ mục của driverdựng hình cần khởi tạo, thông thường là **-1** để khởi tạo driver dựng hình đầu tiên mà SDL tìm thấy trong hệ thống**.**
* flags**:** cờ mô tả trạng thái ngữ cảnh dựng hình cần tạo, kết hợp với nhau bằng toán tử |**.** Gồm có một số giá trị như sau:
  + SDL\_RENDERER\_SOFTWARE: Sửdụng driver phần mềm để dựng hình**.** Chỉ sử dụng cho mục đích duy trì tính tương thích trên các hệ thống không có phần cứng tăng tốc đồ họa.o SDL\_RENDERER\_ACCELERATED: Sử dụng phần cứng tăng tốc đồ họa để dựng hình. Đây là giá trị thường được sử dụng nhất.
  + SDL\_RENDERER\_PRESENTVSYNC: Đồng bộ tốc độ frame với tốc độ làm tươi của **màn hình thông qua** cơ chế đồng bộ dọc (***Vertical Synchronization, V-Sync*).** Với cơ chế đồng bộ này, tốc độ frame tối đa của ứng dụng sẽ xấp xỉ 50hoặc **60 frame/giây** tùy thuộc vào thiết bị hiển thị.

Hàm này trả về con trỏ đến đối tượng ngữ cảnh dựng hình **2D** SDL\_Renderernếu thành công. Ngược lại, hàm trả về một trị NULL**.**

 void SDL\_DestroyRenderer(SDL\_Renderer \* renderer)

**Hàm này** thực hiện việc hủy bỏ đối tượng ngữ cảnhdựng hình đã được tạo bởi hàm SDL\_CreateRenderer**.** Các tài nguyên đã cấp phát cho đối tượng dựng hình này cũng được thu hồi.

### **c. Một số thao tác dựng hình cơ sở**

Phần này trình bày một số thao tác dựng hình cơ sở với SDL 2. **Ta** cần lưu ý rằng **SDL 2** dựng hình sử dụng cơ chế ***double-buffering* (Hình 2).** Mọi thao tác dựng hình sẽ kết xuất kết quả lên một vùng nhớ đệm **(*framebuffer*).** Nội dung vùng nhớ đệm này sẽ được hiển thị lêncửa sổhoặc màn hình thông qua một lời gọi hàm sau khi ta đã hoàn thành việc dựng hình cho một frame.

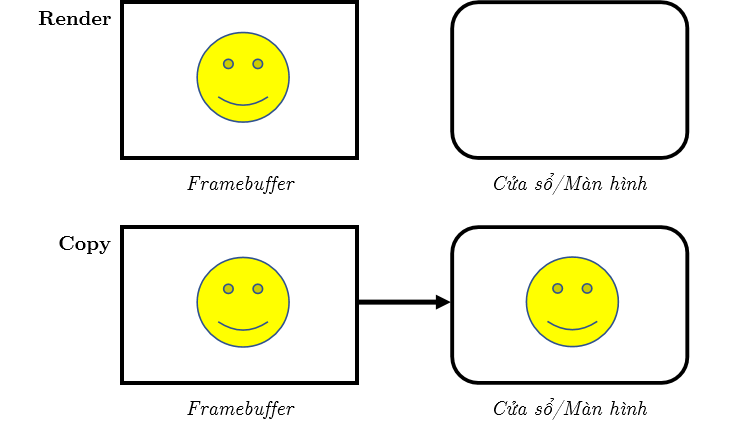
* int SDL\_SetRenderDrawColor(SDL\_Renderer \* rend, Uint8 r, Uint8 g, Uint8 b, Uint8 a) Thiết lập giá trị màu cho các thao tác dựng hình kế tiếptrên đối tượ**ng** SDL\_Renderer**.** Giá trị màu được cho bởi bộ 4 r**,** g**,** b **và** a, lần lượt xác định cường độ của các thành phần đỏ, xanh lá, xanh dương và alpha**.** Giá trịcủa tham sốathông thường được sử dụng là SDL\_ALPHA\_OPAQUE**.**
* int SDL\_RenderClear(SDL\_Renderer \* rend)

Xóa toàn bộ nội dung của framebuffer với giá trị **màu** đã được thiết lập bởi hàm

SDL\_SetRenderDrawColor**.**

* void SDL\_RenderPresent(SDL\_Renderer \* rend)

Tiến hành sao chép toàn bộ nội dung của vùng nhớ đệm dựng hình lên cửa sổ hoặc thiết bị hiện thị



**Hình 2. Cơ chế double-buffering**

int SDL\_RenderDrawLine(SDL\_Renderer \* rend, int x1, int y1, int x2, int y2) Vẽ một đoạn thẳng bắt đầu từ điểm có tọa độ ,sử dụng màu được thiết lập bởi hàm SDL\_SetRenderDrawColor**.**

* int SDL\_RenderDrawRect(SDL\_Renderer \* rend, const SDL\_Rect \* rect)

Tương tự hàm SDL\_RenderDrawRect, tuy nhiên hàm này vẽ một hình chữ nhật được tô màu, với màu sắc được xác định bởi hàm SDL\_SetRenderDrawColor.

* int SDL\_RenderFillRect(SDL\_Renderer \* rend, const SDL\_Rect \* rect)

Tương tự hàm SDL\_RenderDrawRect, tuy nhiên hàm này vẽ một hình chữ nhật được tô màu, với màu sắcđược xác định bởi hàm SDL\_SetRenderDrawColor**.**

### **d. Xử lý sự kiện**

**SDL 2 cho phép** kiểm soát khá nhiều loại sự kiện xảy ra trên cửa sổ dựng hình. **Các** sự kiện này được phân loại dựa trên thiết bị phát sinh ra sự kiện đó, ví dụ như các sự kiện phát sinh từ bàn phím, chuột, **joystick**…Thông tin về các sự kiện xảy ra trên cửa sổ được mô tả bởi một cấu trúc dữ liệu cótên gọi SDL\_Event**. P**hần nàychỉ trình bày một số hàm xử lý các sự kiện nhận được từ bàn phím để kiểm soát hoạt động của ứng dụng **mà thôi.**

* int SDL\_PollEvent(SDL\_Event \* e)

Lấy một sự kiện ra khỏi hàng đợi sự kiệnvà lưu vào trong tham số ecó kiểu dữ liệu SDL\_Event**.** Nếu hàng đợi sự kiện rỗng, hàm này vẫn kết thúcvà trả vềtrị **,** ngược lại, hàm sẽ trả về trị **.** Đây là hàm thường được chọn để sử dụng trong vòng lặp sự kiện của các ứng dụng viết trên nền SDL **2.**

* int SDL\_PushEvent(SDL\_Event \* e)

Đưa một sự kiện mô tả bởi tham số evào hàng đợi sự kiện.

* SDL\_WaitEvent(SDL\_Event \* e)

**C**hờ lấy sự kiện tiếp theora khỏi hàng đợi**.** Nếu hàng đợi sự kiện là rỗng, hàm sẽ ngưng hoạt động của ứng dụng để chờ cho đến khi xuất hiện một sự kiện trong hàng đợi để lấy ra.

**C**ó 2 loại sự kiện được phát sinh từ bàn phím **là** SDL\_KEYDOWN **và** SDL\_KEYUP**. Ta** sử dụng các trường của cấu trúc dữ liệu SDL\_KeyboardEventđể trích xuất thông tin từcác sự kiện **này.** SDL\_KeyboardEventthực chất là một unioncủa cấu trúc dữ liệu SDL\_Event**.**

Cấu trúc dữ liệu SDL\_KeyboardEventgồm có các trường sau:

* Uint32 type**:** mô tả loại sự kiện là SDL\_KEYDOWNhoặc SDL\_KEYUP**.**
* Uint32 timestamp**:** thời điểm xảy ra sự kiện**.**
* Uint32 windowID**:** ID của cửa sổ đang nhận sự kiện, nếu có.
* Uint8 state**:** trạng thái của phím là SDL\_PRESSEDhoặc SDL\_RELEASED**.**
* Uint8 repeat**: khác** nếu **phím** đang được nhấn lặp lại.

SDL\_Keysym keysym**:** Là một cấu trúc dữ liệu chứa **các** thông tin mô tả về phím đang xuất hiện bên trong sự kiệ**n (**SDL\_Keysym**)**

Cấu trúc dữ liệu SDL\_Keysymmô tả các thông tin về một phím khi được nhấn hoặc nhả **trên bàn phím:**

* SDL\_Scancode scancode**:** là một enummô tả mã phím vật lý**.** Chi tiết về enumnày được **trình bày** tại địa chỉ https://wiki.libsdl.org/SDL\_Scancode**.**
* SDL\_Keycode sym**:** là một enum **mô** tả mã bàn phím ảo (có thể đã được ánh xạ lại)**.** Chi tiết về enumnày được trình bày tại địa chỉ: https://wiki.libsdl.org/SDL\_Keycode**.**
* Uint16 mod**:** giá trị mô tả phím chức năng kèm theo (nếu có), là một tổ hợp của **các** giá trị sau đây: KMOD\_LSHIFT**,** KMOD\_RSHIFT**,** KMOD\_LCTRL**,** KMOD\_RCTRL**,** KMOD\_LALT**,** KMOD\_RALT**,**

KMOD\_NUM**,** KMOD\_CAPS…

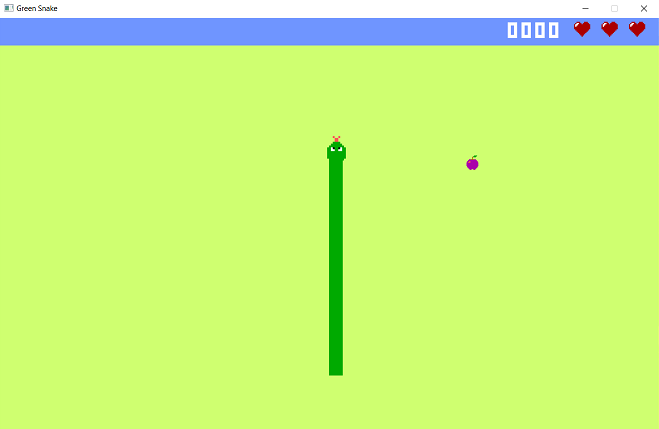
### **e. Xử lý va chạm đơn giản**

SDL\_bool SDL\_HasIntersection(const SDL\_Rect \* a, const SDL\_Rect \* b)

Kiểm tra xem hình chữ nhật acó giao với hình chữ nhật bhay không. Nếu có chúng có va chạm.

# CHƯƠNG 2: GAME GREEN SNAKE

## **2.1 Giao diện chính**



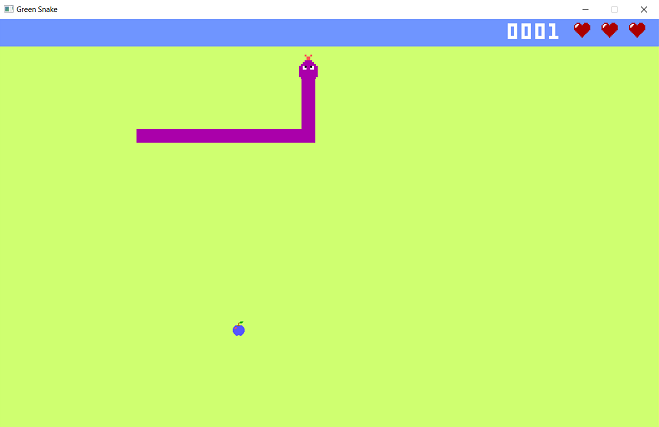
**Hình 3: Giao diện chính của game**

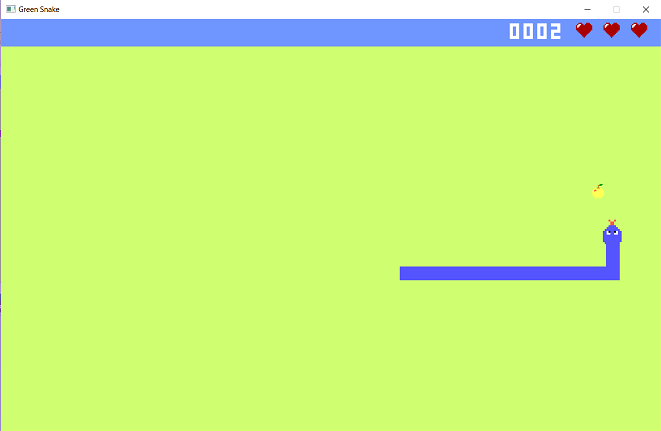
## **2.2 Cách chơi**

Người chơi sẽ sử các “ phím left,down,up,right ” trên bàn phím để di chuyển rắn tới các mồi là các quả táo nhìu màu sắc,sau khi ăn mỗi quả táo ,rắn sẽ tự động chuyển màu tương ứng với màu quả táo.

Màn hình game có xuất hiện điểm số,mỗi lần ăn một quả táo sẽ lên 1 điểm

Mỗi người chơi sẽ có 3 mạng ,mỗi lần rắn đụng thân sẽ mất đi 1 mạng,sau khi hết 3 mạng trò chơi kết thúc





Rắn đổi màu

 Mạng và điểm số của game

## **2.3 Các thư viện sử sụng trong game**

Sdl.h

Stdio.h

Time.h

SDL\_mixer.h

# CHƯƠNG 3: KẾT LUẬN

Green snake là game giải trí được xây dựng bằng thư viện SDL2 thông qua phần mềm Visual Studio 2015.Green Snake sẽ có nhiều hướng phát triển thêm cho game như:

+ Tạo thêm menu cho game

+ Tăng độ khó của game

+ Pause game

+ Thêm nhiều màn mới

…