BÁO CÁO BÀI TẬP LỚN LẬP TRÌNH NÂNG CAO

Đề tài chọn: game 2048

Người làm: Nguyễn Chí Thành – MSSV 18020053

1. **Ý tưởng lập trình**

Có thể thấy rằng 2048 là 1 game có cách chơi khá đơn giản.

Thao tác chính của trò chơi là di chuyển theo 4 hướng: lên, xuống, trái phải. Mỗi thao tác di chuyển của người chơi, ta cần thực hiện các thao tác:

+ Di chuyển các ô dựa theo hướng mà người chơi mong muốn.

+ Gộp các cặp ô có giá trị giống nhau tạo thành ô mới, sau đó cộng điểm cho người chơi.

+ Tạo 1 ô mới ngẫu nhiên có giá trị 2 hoặc 4.

+ Kiểm tra xem trò chơi đã kết thúc hay chưa.

Để tạo các hình ảnh cho trò chơi, ta có thể sử dụng các tính năng tạo texture của SDL. Một lưu ý trong quá trình sử dụng các font và texture, ta cần giải phóng bộ nhớ sử dụng của chúng mỗi khi sử dụng xong để tránh việc bộ nhớ sử dụng tăng lên nhanh chóng và dẫn đến tràn bộ nhớ.

Ngoài ra, tại phần chính của chương trình, ta nên sử dụng hàm SDL\_Delay để giảm mức độ sử dụng CPU của chương trình khi đợi các event. Thông thường, việc chúng ta delay 10ms không làm ảnh hưởng đến hoạt động của chương trình.

1. **Cấu trúc chương trình**
2. **Grid**

Phần này nằm trong thư viện Grid. Tại đây tôi tạo 1 class Grid để lưu giá trị các ô trên màn chơi cùng 1 số hàm để thao tác với các giá trị này.

* 1. **Grid::init**

Hàm này có tác dụng tạo mới màn chơi với 2 ô có sẵn ở vị trí ngẫu nhiên.

* 1. **Grid::gridMove**

Hàm này có tác dụng di chuyển các ô theo hướng được truyền vào từ tham số. Khi thực hiện, hàm này sẽ thực hiện các thao tác sau:

* + Nếu trò chơi chưa kết thúc, được phép di chuyển.
  + Thực hiện các phép di chuyển, kết hợp, cộng điểm.
  + Nếu vẫn còn có ô trống, tạo thêm 1 ô ngẫu nhiên.
  + Kiểm tra xem trò chơi đã kết thúc hay chưa.

Ngoài ra còn một số hàm khác nhỏ hơn để được các hàm trên gọi trong quá trình xử lý:

* 1. **Grid::getAvailableCells**

Hàm này trả về 1 tập các tọa độ các ô còn trống.

* 1. **Grid::addRandom**

Thêm một ô có giá trị 2 / 4 vào 1 vị trí ngẫu nhiên còn trống.

1. **Button**

Thư viện tạo các nút cho chương trình.

* 1. **Constructor**

Khởi tạo nút với tham số truyền vào là nội dung chữ trên nút cùng hàm thực thi khi ấn vào nút.

* 1. **Button::setXY**

Đặt tọa độ của nút trên cửa sổ.

* 1. **Button::render**

Render nút lên cửa sổ với tọa độ X, Y đã được đặt.

* 1. **Button::width và Button::height**

Hàm trả về độ dài và độ rộng của nút.

1. **Controller**

Hỗ trợ sử dụng Game Controller cho trò chơi.

Trên hệ điều hành Windows, để sử dụng chức năng rung thì cần sử dụng Xbox controller hoặc giả lập game controller thành Xbox. Với tay cầm Dualshock 4 của Sony, ta cần dùng DS4Windows. Còn trên một số hệ điều hành Linux, tính năng rung được hỗ trợ bởi kernel.

1. **Font, Render**

2 thư viện này chỉ rút gọn các hàm của SDL giúp dễ dàng hơn trong việc lập trình. Thư viện Font hỗ trợ giải phóng bộ nhớ sử dụng sau khi khởi tạo 1 Font từ file ttf.

1. **Texture**

Thư viện tạo các texture của chương trình.

* 1. **Texture::loadText**

Tạo texture dạng chữ với input từ tham số truyền vào, cùng Font được khởi tạo từ thư viện Font.

* 1. **Texture::createBlank**

Tạo một texture trống cùng kích cỡ được truyền vào.

* 1. **Texture::free**

Giải phóng bộ nhớ mà texture đã được khởi tạo trên bộ nhớ.

* 1. **Texture::render**

Render texture đã được tạo ra lên màn hình với tọa độ được truyền vào

* 1. **Texture::width và Texture::height**

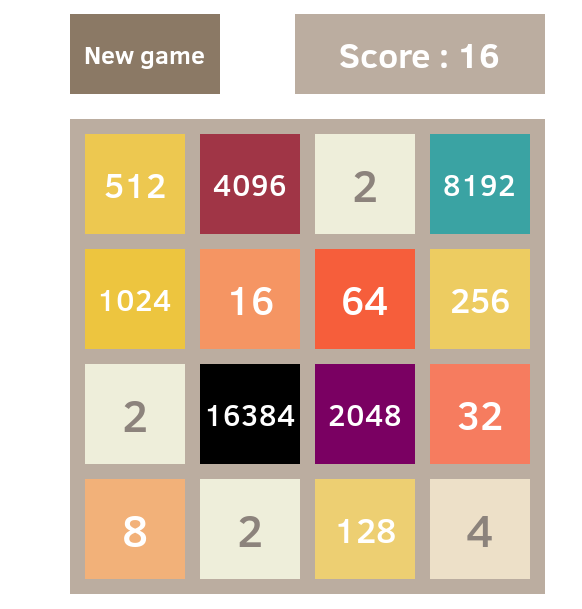
Trả về kích cỡ của texture được tạo ra.

1. **Window**

Thư viện này khởi tạo cửa sổ ban đầu bằng cách sử dụng SDL.

1. **Tổng kết**

Chương trình có sử dụng 1 số example code từ lazyfoo trong phần khởi tạo SDL để tiết kiệm thời gian code.

Các màu trong trò chơi được lấy 1 phần từ trò chơi gốc <http://git.io/2048>.