Trong SDL, thuật ngữ "mouse focus" có nghĩa là trạng thái mà một cửa sổ SDL đang nhận các sự kiện chuột, cho phép người dùng tương tác với cửa sổ đó bằng chuột. Khi một cửa sổ có mouse focus, nó sẽ nhận tất cả các sự kiện chuột, chẳng hạn như nhấn và thả nút chuột, di chuyển chuột và sự kiện bánh xe chuột, cho đến khi mà nó mất mouse focus.

Mặc định, một cửa sổ SDL sẽ không có mouse focus khi nó được tạo ra. Để nhận các sự kiện chuột, cửa sổ phải yêu cầu mouse focus bằng cách sử dụng hàm SDL\_SetWindowInputFocus(). Sau khi có được mouse focus, cửa sổ sẽ tiếp tục nhận các sự kiện chuột cho đến khi nó bị mất mouse focus.

Mouse focus là một khái niệm quan trọng trong lập trình SDL, cho phép bạn tạo ra các ứng dụng tương tác phản hồi với đầu vào của người dùng. Ví dụ, nếu bạn đang xây dựng một trò chơi, bạn sẽ cần phải có mouse focus cho cửa sổ trò chơi của bạn để nhận các sự kiện chuột và cho phép người dùng tương tác với trò chơi bằng chuột.

SDL cũng hỗ trợ thay đổi kích thước Windows. Khi bạn có window thay đổi được kích thước, sẽ có thêm một vài events cần xử lý, điều mà ta sẽ làm ở đây.

|  |
| --- |
| class LWindow  {  public:      // Intializes internals      LWindow();      // Creates window      bool init();      // Creates renderer from internal window      SDL\_Renderer \*createRenderer();      // Handles window events      void handleEvent(SDL\_Event &e);      // Deallocates internals      void free();      // Window dimensions      int getWidth();      int getHeight();      // Window focii      bool hasMouseFocus();      bool hasKeyboardFocus();      bool isMinimized();  private:      // Window data      SDL\_Window \*mWindow;      // Window dimensions      int mWidth;      int mHeight;      // Window focus      bool mMouseFocus;      bool mKeyboardFocus;      bool mFullScreen;      bool mMinimized;  }; |

* Đây là window class của chúng ta, chúng ta sẽ sử dụng một wrapper cho SDL\_Window. Nó có constructor, initializer tạo cửa sổ, một hàm để tạo một renderer từ window, một event handler, một deallocator, và vài hàm accessor để lấy các thành phần khác nhau từ window.
* Về mặt thành viên data, chúng ta có window chúng ta gói lại, kích thước window, và flag cho các loại tiêu điểm mà window có. Chúng ta sẽ đi sâu hơn trong chương trình.

|  |
| --- |
| // Our custom window  LWindow gWindow;  // The window renderer  SDL\_Renderer \*gRenderer = NULL;  // Scene textures  LTexture gSceneTexture; |

* Khai báo các biến cục bộ

|  |
| --- |
| LWindow::LWindow()  {      // Initialize non-existant window      mWindow = NULL;      mMouseFocus = false;      mKeyboardFocus = false;      mFullScreen = false;      mMinimized = false;      mWidth = 0;      mHeight = 0;  } |

* Trong constructor khởi tạo các biến

|  |
| --- |
| bool LWindow::init()  {      // Create window      mWindow = SDL\_CreateWindow("SDL Tutorial", SDL\_WINDOWPOS\_UNDEFINED, SDL\_WINDOWPOS\_UNDEFINED, SCREEN\_WIDTH, SCREEN\_HEIGHT, SDL\_WINDOW\_SHOWN | **SDL\_WINDOW\_RESIZABLE**);      if (mWindow != NULL)      {          mMouseFocus = true;          mKeyboardFocus = true;          mWidth = SCREEN\_WIDTH;          mHeight = SCREEN\_HEIGHT;      }      return mWindow != NULL;  } |

* Hàm khởi tạo sẽ tạo window với SDL\_WINDOW\_RESIZABLE cho phép cửa sổ của chúng ta có thể thay đổi được kích thước. Nếu hàm thành công, ta thiết lập flag liên quan và kích thước. Sau đó chúng ta trả về window có NULL hay không.

|  |
| --- |
| SDL\_Renderer \*LWindow::createRenderer()  {      return SDL\_CreateRenderer(mWindow, -1, SDL\_RENDERER\_ACCELERATED | SDL\_RENDERER\_PRESENTVSYNC);  } |

* Ở đây chúng ta xử lý việc tạo renderer từ thành viên window. Chúng ta trả lại renderer được tạo vì việc render sẽ được xử lý bên ngoài class.

|  |
| --- |
| switch (e.window.event)          {          // Get new dimensions and repaint on window size change          case SDL\_WINDOWEVENT\_SIZE\_CHANGED:              mWidth = e.window.data1;              mHeight = e.window.data2;              SDL\_RenderPresent(gRenderer);              break;          // Repaint on exposure          case SDL\_WINDOWEVENT\_EXPOSED:              SDL\_RenderPresent(gRenderer);              break; |

* Khi chúng ta có một window, chúng ta sẽ muốn kiểm tra SDL\_WindowEventID để xem là loại sự kiện. SDL\_WINDOWEVENT\_SIZE\_CHANGED là sự kiện resize, vì thế chúng ta sẽ lấy kích thước và refresh ảnh trên màn hình.
* SDL\_WINDOWEVENT\_EXPOSED chỉ có nghĩa là window bị che khuất nhưng giờ không bị nữa nên chúng ta muốn vẽ lại window.

|  |
| --- |
| // Mouse entered window          case SDL\_WINDOWEVENT\_ENTER:              mMouseFocus = true;              updateCaption = true;              break;          // Mouse left window          case SDL\_WINDOWEVENT\_LEAVE:              mMouseFocus = false;              updateCaption = true;              break;          // Window has keyboard focus          case SDL\_WINDOWEVENT\_FOCUS\_GAINED:              mKeyboardFocus = true;              updateCaption = true;              break;          // Window lost keyboard focus          case SDL\_WINDOWEVENT\_FOCUS\_LOST:              mKeyboardFocus = false;              updateCaption = true;              break; |

SDL\_WINDOWEVENT\_ENTER/ SDL\_WINDOWEVENT\_LEAVE xử lý khi chuột di chuyển và và ra khỏi window.

SDL\_WINDOWEVENT\_FOCUS\_GAINED / SDL\_WINDOWEVENT\_FOCUS\_LOST phải có khi window lấy input từ bàn phím. Vì chú thích của chúng ta theo dõi mouse/keyboard focus, chúng ta thiết lập flag cập nhật chú thích khi bất kỳ sự kiện nào xảy ra.

|  |
| --- |
| // Window minimized          case SDL\_WINDOWEVENT\_MINIMIZED:              mMinimized = true;              break;          // Window maximized          case SDL\_WINDOWEVENT\_MAXIMIZED:              mMinimized = false;              break;          // Window restored          case SDL\_WINDOWEVENT\_RESTORED:              mMinimized = false;              break; |

Cuối cùng, ở đây chúng ta xử lý khi window được thu nhỏ, phóng to hoặc khôi phục lại từ thu nhỏ.

|  |
| --- |
| // Update window caption with new data          if (updateCaption)          {              std::stringstream caption;              caption << "SDL Tutorial - MouseFocus:" << ((mMouseFocus) ? "On" : "Off") << " KeyboardFocus:" << ((mKeyboardFocus) ? "On" : "Off");              SDL\_SetWindowTitle(mWindow, caption.str().c\_str());          } |

Nếu chú thích cần được cập nhật, chúng ta load stringstream với dữ liệu cập nhật và cập nhật chú thích với SDL\_SetWindowTitle.

|  |
| --- |
| // Enter exit full screen on return key      else if (e.type == SDL\_KEYDOWN && e.key.keysym.sym == SDLK\_RETURN)      {          if (mFullScreen)          {              SDL\_SetWindowFullscreen(mWindow, SDL\_FALSE);              mFullScreen = false;          }          else          {              SDL\_SetWindowFullscreen(mWindow, SDL\_TRUE);              mFullScreen = true;              mMinimized = false;          }      } |

Trong bản demo này, chúng ta sẽ chuyển đổi fullscreen với phím return. Chúng ta có thể thiết lập fullscreen sử dụng SDL\_SetWindowFullscreen .

|  |
| --- |
| int LWindow::getWidth()  {      return mWidth;  }  int LWindow::getHeight()  {      return mHeight;  }  bool LWindow::hasMouseFocus()  {      return mMouseFocus;  }  bool LWindow::hasKeyboardFocus()  {      return mKeyboardFocus;  }  bool LWindow::isMinimized()  {      return mMinimized;  } |

Đây là các accessor ta dùng.

|  |
| --- |
| // Create window          if (!gWindow.init())          {              printf("Window could not be created! SDL Error: %s\n", SDL\_GetError());              success = false;          }          else          {              // Create renderer for window              gRenderer = gWindow.createRenderer();              if (gRenderer == NULL)              {                  printf("Renderer could not be created! SDL Error: %s\n", SDL\_GetError());                  success = false;              } |

Trong hàm khởi tạo, chúng ta tạo window và renderer chỉ lần này với window wrapper (trình đóng gói hết vào 1 window)

|  |
| --- |
| void close()  {      // Free loaded images      gSceneTexture.free();      // Destroy window      SDL\_DestroyRenderer(gRenderer);      gWindow.free();      // Quit SDL subsystems      IMG\_Quit();      SDL\_Quit();  } |

Trong hàm dọn dẹp chúng ta vẫn deallocate window và renderer của chúng ta.

|  |
| --- |
| // While application is running              while (!quit)              {                  // Handle events on queue                  while (SDL\_PollEvent(&e) != 0)                  {                      // User requests quit                      if (e.type == SDL\_QUIT)                      {                          quit = true;                      }                      // Handle window events                      gWindow.handleEvent(e);                  }  **// Only draw when not minimized**  **if (!gWindow.isMinimized())**                  {                      // Clear screen                      SDL\_SetRenderDrawColor(gRenderer, 0xFF, 0xFF, 0xFF, 0xFF);                      SDL\_RenderClear(gRenderer);                      // Render text textures                      gSceneTexture.render((gWindow.getWidth() - gSceneTexture.getWidth()) / 2, (gWindow.getHeight() - gSceneTexture.getHeight()) / 2);                      // Update screen                      SDL\_RenderPresent(gRenderer);                  }              } |

Trong hàm main, chúng ta phải bảo đảm truyền events tới window wrapper để xử lý sự kiện resize và trong phần code rendering, ta phải đảm bảo để chỉ render khi window không minimize vì có thể gây một số lỗi khi chúng ta cố gắng render tới một minimized window.