Trong bài trước, chúng ta đã làm một bộ đếm thời gian cơ bản. Trong bài này chúng ta sẽ là một cái có thể start/stop/pause.

|  |
| --- |
| //The application time based timer  class LTimer  {      public:          //Initializes variables          LTimer();          //The various clock actions          void start();          void stop();          void pause();          void unpause();          //Gets the timer's time          Uint32 getTicks();          //Checks the status of the timer          bool isStarted();          bool isPaused();      private:          //The clock time when the timer started          Uint32 mStartTicks;          //The ticks stored when the timer was paused          Uint32 mPausedTicks;          //The timer status          bool mPaused;          bool mStarted;  }; |

* Để làm những tính năng mới này, chúng ta sẽ làm một timer class. Nó có tất cả các hàm cơ bản để start/stop/pause/unpause booj đếm và kiểm tra trạng thái của nó. Về dữ liệu thành viên, chúng ta có thời gian bắt đầu như trước, một biến lưu thời gian khi dừng, và flag trạng thái để theo dõi xem bộ đếm đang chạy hay ngừng.

|  |
| --- |
| LTimer::LTimer()  {      //Initialize the variables      mStartTicks = 0;      mPausedTicks = 0;      mPaused = false;      mStarted = false;  } |

* Constructor sẽ khởi tạo dữ liệu thành viên bên trong.

|  |
| --- |
| void LTimer::start()  {      //Start the timer      mStarted = true;      //Unpause the timer      mPaused = false;      //Get the current clock time      mStartTicks = SDL\_GetTicks();      mPausedTicks = 0;  } |

* Hàm bắt đầu thiết lập flag bắt đầu và flag tạm ngưng, lấy thời gian start của bộ đếm và khởi tạo thời gian pause là 0. Trong bộ đếm này, nếu chúng ta muốn restart nó, chúng ta có thể gọi hàm start ra một lần nữa. Vì chúng ta có thể start bộ đếm nếu nó đang pause hoặc running, nên chúng ta chắc chắn đã xóa dữ liệu pause.

|  |
| --- |
| void LTimer::stop()  {      //Stop the timer      mStarted = false;      //Unpause the timer      mPaused = false;      //Clear tick variables      mStartTicks = 0;      mPausedTicks = 0;  } |

* Hàm stop cơ bản chỉ là khởi tạo lại các biến.

|  |
| --- |
| void LTimer::pause()  {      //If the timer is running and isn't already paused      if( mStarted && !mPaused )      {          //Pause the timer          mPaused = true;          //Calculate the paused ticks          mPausedTicks = SDL\_GetTicks() - mStartTicks;          mStartTicks = 0;      }  } |

* Khi pause, chúng ta sẽ kiểm tra xem timer đang chạy không vì pause một timer chưa start không hợp lý tí nào. Nếu timer đang chạy thì ta đặt pause flag, lưu thời gian mà timer pause ở mPausedTicks, và reset lại thời gian start.

|  |
| --- |
| void LTimer::unpause()  {      //If the timer is running and paused      if( mStarted && mPaused )      {          //Unpause the timer          mPaused = false;          //Reset the starting ticks          mStartTicks = SDL\_GetTicks() - mPausedTicks;          //Reset the paused ticks          mPausedTicks = 0;      }  } |

* Khi chúng ta unpause cái timer thì cần xem xem timer đang chạy hay pause vì chúng ta không thể unpause một timer đã stop hay đang chạy. Chúng ta thiết lập lại pause flag và đặt lại thời gian start mới.

|  |
| --- |
| Uint32 LTimer::getTicks()  {      //The actual timer time      Uint32 time = 0;      //If the timer is running      if( mStarted )      {          //If the timer is paused          if( mPaused )          {              //Return the number of ticks when the timer was paused              time = mPausedTicks;          }          else          {              //Return the current time minus the start time              time = SDL\_GetTicks() - mStartTicks;          }      }      return time;  } |

Việc lấy thời gian hơi phức tạp một chút vì timer của chúng ta có thể run/pause/stop. Nếu nó top, chỉ cần trả về 0, Nếu nó pause, chỉ cần trả về thời gian đã lưu khi dừng. Còn đang chạy ko pause thì trả lại thời gian tương đối với khi nó start.

|  |
| --- |
| bool LTimer::isStarted()  {      //Timer is running and paused or unpaused      return mStarted;  }  bool LTimer::isPaused()  {      //Timer is running and paused      return mPaused && mStarted;  } |

Ở đây chúng ta có một số accessor để kiểm tra trạng thái của timer.

|  |
| --- |
| //Main loop flag              bool quit = false;              //Event handler              SDL\_Event e;              //Set text color as black              SDL\_Color textColor = { 0, 0, 0, 255 };              //The application timer              LTimer timer;              //In memory text stream              std::stringstream timeText; |

Trước khi vào main loop, chúng ta khai báo timer object và string stream để biến thời gian sang văn bản.

|  |
| --- |
| //Reset start time on return keypress                      else if( e.type == SDL\_KEYDOWN )                      {                          //Start/stop                          if( e.key.keysym.sym == SDLK\_s )                          {                              if( timer.isStarted() )                              {                                  timer.stop();                              }                              else                              {                                  timer.start();                              }                          }                          //Pause/unpause                          else if( e.key.keysym.sym == SDLK\_p )                          {                              if( timer.isPaused() )                              {                                  timer.unpause();                              }                              else                              {                                  timer.pause();                              }                          }                      } |

* Khi chúng ta nhấn phím s, ta sẽ kiểm tra xem timer đã start chưa. Nếu rồi thì stop nó, nếu chưa thì start. Khi chúng ta nhấn p, chúng ta sẽ kiểm tra xem timer đã pause chưa. Nếu rồi thì unpause, nếu chưa thì pause.

|  |
| --- |
| //Set text to be rendered  timeText.str( "" );  timeText << "Seconds since start time " << ( timer.getTicks() / 1000.f ) ;  //Render text  if( !gTimeTextTexture.loadFromRenderedText( timeText.str().c\_str(), textColor ) )  {      printf( "Unable to render time texture!\n" );  }  //Clear screen  SDL\_SetRenderDrawColor( gRenderer, 0xFF, 0xFF, 0xFF, 0xFF );  SDL\_RenderClear( gRenderer );  //Render textures  gStartPromptTexture.render( ( SCREEN\_WIDTH - gStartPromptTexture.getWidth() ) / 2, 0 );  gPausePromptTexture.render( ( SCREEN\_WIDTH - gPausePromptTexture.getWidth() ) / 2, gStartPromptTexture.getHeight() );  gTimeTextTexture.render( ( SCREEN\_WIDTH - gTimeTextTexture.getWidth() ) / 2, ( SCREEN\_HEIGHT - gTimeTextTexture.getHeight() ) / 2 );  //Update screen  SDL\_RenderPresent( gRenderer ); |

* Trước khi render, chúng ta sẽ viết thời gian hiện tại tới string stream. Lý do ta chia nó cho 1000 là vì ta muốn đơn vị là giây, có có 1s = 1000ms.
* Sau đó, ta render text tới một texture và cuối cùng vẽ tất cả texture lên màn hình.
* Để tránh timer bị nhấp nháy do render width thay đổi liên tục, chúng ta có thể dùng một biến lấy max các width trước đó: mWidth = std::max(mWidth, gTimeTextTexture.getWidth());