Vì giờ chúng ta đã biết cách render, xử lý input và xử lý với thời gian. Đó là mọi thứ cần thiết để di chuyển một vật trên màn hình. Ở đây, chúng ta sẽ làm một chương trình cơ bản với một điểm di chuyển xung quanh.

|  |
| --- |
| // The dot that will move around on the screen  class Dot  {  public:      // The dimensions of the dot      static const int DOT\_WIDTH = 20;      static const int DOT\_HEIGHT = 20;      // Maximum axis velocity of the dot      static const int DOT\_VEL = 10;      // Initializes the variables      Dot();      // Takes key presses and adjusts the dot's velocity      void handleEvent(SDL\_Event &e);      // Moves the dot      void move();      // Shows the dot on the screen      void render();  private:      // The X and Y offsets of the dot      int mPosX, mPosY;      // The velocity of the dot      int mVelX, mVelY;  }; |

* Đây là class cho điểm mà chúng ta chuẩn bị di chuyển quanh màn hình. Nó có vài hằng số để xác đinh tọa độ và vận tốc. Nó có constructor, xử lý event, một hàm để di chuyển nó mỗi khung hình và một hàm để render nó. Còn dữ liệu thành viên thì gồm có biến cho vị trí x/y và tốc độ x/y.

|  |
| --- |
| Dot::Dot()  {      // Initialize the offsets      mPosX = 0;      mPosY = 0;      // Initialize the velocity      mVelX = 0;      mVelY = 0;  } |

Hàm tạo chỉ đơn giản khởi tạo các biến.

|  |
| --- |
| void Dot::handleEvent(SDL\_Event &e)  {      // If a key was pressed      if (e.type == SDL\_KEYDOWN && e.key.repeat == 0)      {          // Adjust the velocity          switch (e.key.keysym.sym)          {          case SDLK\_UP: mVelY -= DOT\_VEL; break;          case SDLK\_DOWN: += DOT\_VEL; break;          case SDLK\_LEFT: mVelX -= DOT\_VEL; break;          case SDLK\_RIGHT: mVelX += DOT\_VEL; break;          }      } |

* Trong event handler, chúng ta sẽ thiết lập vận tốc dựa vào phím đang được nhấn.
* Bạn có thể tự hỏi tại sao chúng tôi sẽ không đơn giản chỉ tăng vị trí khi nhấn phím. Nếu chúng ta chỉ tăng vị trí x mỗi lần chúng ta nhấn phím right, thì chúng ta sẽ phải nhấn liên tục để giữ nó di chuyển. Bằng cách thiết lập vận tốc, chúng ta chỉ phải giữ phím một lần.
* Nếu bạn tự hỏi tại sao chúng tôi (Điệp ngữ hả :v) kiểm tra xem key repeat = 0, thì bởi vì key repeat được bật mặc định và nếu bạn nhấn và giữ phím, nó sẽ report là nhiều lần nhấn phím. Điều này nghĩa là chúng ta phải check xem phím được nhấn là cái đầu tiên không vì chúng ta chỉ quan tâm đến phím đầu tiên được nhấn.
* Cho những ai chưa học vật lý, vận tốc là tốc độ và hướng của một vật. Nếu vật chuyển động sáng phải 10 pixel/khung hình thì nó có vận tốc là 10, cùng tốc độ đó nhưng di chuyển sang trái thì có vận tốc là -10. Nếu vận tốc là 10 thì sau 10 khung hình nó đã đi được 100 pixel.

|  |
| --- |
| // If a key was released      else if (e.type == SDL\_KEYUP && e.key.repeat == 0)      {          // Adjust the velocity          switch (e.key.keysym.sym) {          case SDLK\_UP: mVelY += DOT\_VEL; break;          case SDLK\_DOWN: mVelY -= DOT\_VEL; break;          case SDLK\_LEFT: mVelX += DOT\_VEL; break;          case SDLK\_RIGHT: mVelX -= DOT\_VEL; break;          }      } |

* Khi chúng ta nhả một phím, chúng ta sẽ undo cái vận tốc vừa thay đổi.

|  |
| --- |
| void Dot::move()  {      // Move the dot left or right      mPosX += mVelX;      // If the dot went too far to the left or right      if ((mPosX < 0) || (mPosX + DOT\_WIDTH > SCREEN\_WIDTH))      {          // Move back          mPosX -= mVelX;      } |

* Đây là hàm chúng ta gọi mỗi khung hình để di chuyển điểm.
* Đầu tiên, chúng ta di chuyển điểm dọc theo trục x dựa trên vận tốc của nó. Sau đó, chúng ta kiểm tra xem chấm đã di chuyển ra ngoài màn hình chưa. Nếu rồi thì undo cái chuyển động ở trục x.

|  |
| --- |
| // Move the dot up or down      mPosY += mVelY;      // If the dot went too far up or down      if ((mPosY < 0) || (mPosY + DOT\_HEIGHT > SCREEN\_HEIGHT))      {          // Move back          mPosY -= mVelY;      }  } |

* Tương tự với trục y.

|  |
| --- |
| void Dot::render()  {      // Show the dot      gDotTexture.render(mPosX, mPosY);  } |

Trong hàm render, chúng ta render texture của điểm tại vị trí của điểm.

|  |
| --- |
| // Main loop flag              bool quit = false;              // Event handler              SDL\_Event e;              // The dot that will be moving around on the screen              Dot dot; |

Trước khi vào main loop, chúng ta khai báo điểm.

|  |
| --- |
| // 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 input for the dot                      dot.handleEvent(e);                  }                  // Move the dot                  dot.move();                  // Clear screen                  SDL\_SetRenderDrawColor(gRenderer, 0xFF, 0xFF, 0xFF, 0xFF);                  SDL\_RenderClear(gRenderer);                  // Render objects                  dot.render();                  // Update screen                  SDL\_RenderPresent(gRenderer);              } |

* Cuối cùng, chúng ta dùng điểm vào trong main loop. Trong event loop, chúng ta xử lý event cho điểm. Sau đó chúng ta update vị trí của điểm và render nó lên màn hình.
* Bây giờ, chúng ta đang tính vận tốc dựa trên lượng di chuyển mỗi khung hình. Trong phần lớn game, vận tốc được tính theo mỗi giây. Lý do làm nó theo khung hình là vì nó đơn giản hơn, nhưng nếu bạn biết vật lý thì không khó để update vị trí của một điểm theo thời gian.