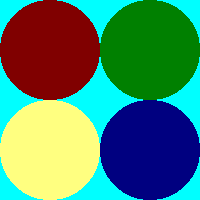
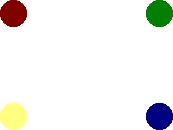
# Sprite sheet là gì?

* Trong đồ họa máy tính, **sprite** là một hình ảnh hai chiều hoặc hình ảnh động được tích hợp vào một cảnh lớn hơn, có thể dịch chuyển nó thông qua cách đặt tọa độ trong hàm vẽ lên màn hình.
* Thực tế trong 1 game, thường có từ 10 đến hàng trăm Sprite, việc tải riêng từng ảnh sẽ tiêu tốn năng suất tổ chức và chiếm nhiều bộ nhớ và mất nhiều thời gian để xử lý. Để quản lý các Sprite và tránh sử dụng quá nhiều hình ảnh người ta tạo ra **sprite sheet**.
* **Sprite sheet** là tập hợp nhiều Sprite đơn lẻ thành 1 tập tin duy nhất, giúp tăng tốc độ xử lý cho việc hiển thị hình ảnh lên màn hình. Do chỉ hiển thị 1 phần của bức ảnh (load 1 lần lên bộ nhớ) sẽ tốt hơn nhiều với việc lấy nhiều bức ảnh và hiển thị chúng.
* Dựa trên file sprite sheet, cho hiển thị từng sprite chuyển tiếp nhau 1 cách liên tục sẽ tạo ra cảm giác đối tượng như đang chuyển động.
* Khi thao tác với **sprite animation** thường gặp thêm khái niệm Tile, 1 sprite sheet lưu 1 bức ảnh là 1 chuỗi các Tile, mỗi Tile hiển thị 1 khung hình của chuỗi chuyển động, như vậy Tile cũng giống với Sprite/Frame.

# Trở lại bài học

Đôi khi bạn chỉ muốn render một phần của texture. Nhiều game cũng thích giữ nhiều ảnh trên sprite sheet thay vì trên nhiều texture. Sử dụng clip rendering, ta có thể xác định một phần của texture để render thay vì render hết.

Đây là sprite sheet:  và ta cần render mỗi sprite sang 4 góc: 

|  |
| --- |
| class Ltexture {  public:      // Initializes variables      LTexture();      // Deallocates memory      ~LTexture();      // Loads image at specified path      bool loadFromFile(std::string path);      // Deallocates texture      void free();      // Renders texture at given point      void render(int x, int y, SDL\_Rect \*clip = NULL);      // Gets image dimensions      int getWidth();      int getHeight();  private:      // The actual hardware texture      SDL\_Texture \*mTexture;      // Image dimensions      int mWidth;      int mHeight;  }; |

* Ở đây chúng tôi có một vài điều chỉnh nhỏ để hàm render nằm trong texture class. Hàm render bây giờ cho phép một HCN xác định phần nào của texture chúng ta muốn render. Chúng ta cho nó một biến mặc định là NULL trong trường hợp chúng ta muốn render toàn bộ texture.

|  |
| --- |
| // Scene sprites  SDL\_Rect gSpriteClips[4];  LTexture gSpriteSheetTexture; |

* Chúng ta cần một texture ảnh và 4 HCN để xác định sprite, đó là biến mà bạn đã thấy khởi tạo ở đây.

|  |
| --- |
| void LTexture::render(int x, int y, SDL\_Rect \*clip)  {      // Set rendering space and render to screen      SDL\_Rect renderQuad = {x, y, mWidth, mHeight};      // Set clip rendering dimensions      if (clip != NULL)      {          renderQuad.w = clip->w;          renderQuad.h = clip->h;      }      // Render to screen      SDL\_RenderCopy(gRenderer, mTexture, clip, &renderQuad);  } |

* Đây là hàm render mới cho texture class mà hỗ trợ clip rendering. Nó giống phần lớn với hàm trong phần trước nhưng có 2 điểm khác:
* Đầu tiên là khi bạn cắt ảnh, bạn sử dụng tọa độ của HCN cần cắt thay vì của texture, chúng ta sẽ set chiều dài/rộng của HCN đích (gọi là renderQuad) sang size của HCN cần cắt.
* Thứ hai là chúng ta sẽ truyền HCN cắt vào SDL\_RenderCopy() như là HCN nguồn. HCN nguồn xác định phần nào của texture bạn muốn render. Khi HCN nguồn là NULL thì sẽ render toàn bộ texture.

|  |
| --- |
| bool loadMedia(){      // Loading success flag      bool success = true;      // Load sprite sheet texture      if (!gSpriteSheetTexture.loadFromFile("dots.png")){          printf("Failed to load sprite sheet texture!\n");          success = false;      }      else {          // Set top left sprite          gSpriteClips[0].x = 0;          gSpriteClips[0].y = 0;          gSpriteClips[0].w = 100;          gSpriteClips[0].h = 100;          // Set top right sprite          …          // Set bottom left sprite          …          // Set bottom right sprite          …      }      return success;  } |

* Hàm load media sẽ load texture và xác định HCN cắt cho chu kỳ sprite nếu texture load thành công.

|  |
| --- |
| // 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;          }      }      // Clear screen      SDL\_SetRenderDrawColor(gRenderer, 0xFF, 0xFF, 0xFF, 0xFF);      SDL\_RenderClear(gRenderer);      // Render top left sprite      gSpriteSheetTexture.render(0, 0, &gSpriteClips[0]);      // Render top right sprite      …      // Render bottom left sprite      …      // Render bottom right sprite      …      // Update screen      SDL\_RenderPresent(gRenderer);  } |

* Cuối cùng là main loop, chúng ta sẽ render cùng 1 texture 4 lần, nhưng ở 4 phần khác nhau trong mỗi lệnh.