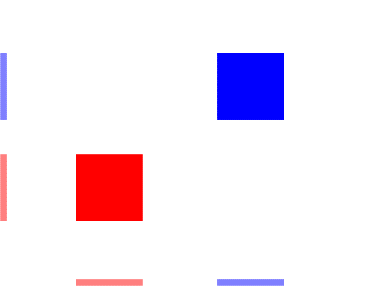
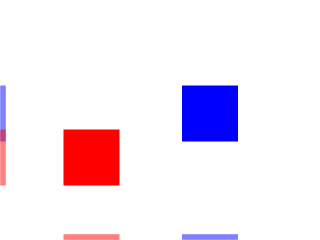
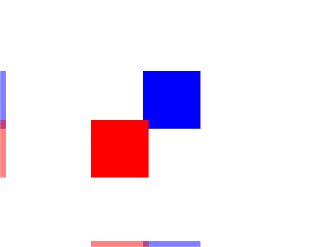
# 1. Mở đầu

* Trong game, bạn thường xuyên cần xem hai vật thể có chạm nhau không. Trong những game đơn giản, điều này bình thường được thực hiện với phát hiện va chạm bounding box (hộp bao quanh).
* Hộp va chạm là cách tiêu chuẩn dể kiểm tra va chạm giữa hai vật thể. Hai đa giác va chạm nhau nếu chúng không tách rời.
* Ở đây chúng ta có hai box ko va chạm. Như bạn có thể thấy, x y đều tách rời: 
* Còn ở đây thì va chạm trên y nhưng tách rời trên x: 
* Đây là khi 2 box va chạm, cả x và y đều va chạm: 

|  |
| --- |
| // 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 and checks collision      void move(SDL\_Rect &wall);      // 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;  **// Dot's collision box**  **SDL\_Rect mCollider;**  }; |

* Đây là điểm trong bài trước. Chúng ta có thêm dữ liệu thành viên mới là mCollider đại diện cho collision box. Hàm di chuyển sẽ lấy HCN là collision box làm tường.

|  |
| --- |
| Dot::Dot()  {      // Initialize the offsets      mPosX = 0;      mPosY = 0;  **// Set collision box dimension**  **mCollider.w = DOT\_WIDTH;**  **mCollider.h = DOT\_HEIGHT;**      // Initialize the velocity      mVelX = 0;      mVelY = 0;  } |

Trong hàm tạo, chú ý khởi tạo kích thước của collider.

|  |
| --- |
| void Dot::move(SDL\_Rect &wall)  {      // Move the dot left or right      mPosX += mVelX;      mCollider.x = mPosX;      // If the dot collided or went too far to the left or right      if ((mPosX < 0) || (mPosX + DOT\_WIDTH > SCREEN\_WIDTH) || checkCollision(mCollider, wall))      {          // Move back          mPosX -= mVelX;          mCollider.x = mPosX;      }      // Move the dot up or down      mPosY += mVelY;      mCollider.y = mPosY;      // If the dot collided or went too far up or down      if ((mPosY < 0) || (mPosY + DOT\_HEIGHT > SCREEN\_HEIGHT) || checkCollision(mCollider, wall))      {          // Move back          mPosY -= mVelY;          mCollider.y = mPosY;      }  } |

* Đây là hàm di chuyển mới, giờ sẽ kiểm tra xem chúng ta có va chạm tường không. Nó hoạt động như trước, chỉ là giờ nó làm cho điểm không đi ngược lại nếu ra ngoài màn hình hay chạm phải tường.
* Đầu tiên chúng ta di chuyển điểm dọc theo trục x, nhưng chúng ta cũng phải thay đổi vị trí của colllider. Bất cứ khi nào chúng ta thay đổi vị trí của điểm, vị trí của collider cũng phải thay đổi theo.
* Sau đó chúng ta sẽ kiểm tra xem điểm có đi quá màn hình hay chạm tường không. Nếu có chúng ta phải di chuyển nó trở lại.

|  |
| --- |
| bool checkCollision(SDL\_Rect a, SDL\_Rect b)  {      // The sides of the rectangles      int leftA, leftB;      int rightA, rightB;      int topA, topB;      int bottomA, bottomB;      // Calculate the sides of rect A      leftA = a.x;      rightA = a.x + a.w;      topA = a.y;      bottomA = a.y + a.h;      // Calculate the sides of rect B      leftB = b.x;      rightB = b.x + b.w;      topB = b.y;      bottomB = b.y + b.h; |

Đây là nơi kiểm tra va chạm. Đoạn code này tính trên dưới trái phải của collision box.

|  |
| --- |
| // If any of the sides from A are outside of B      if (bottomA <= topB)      {          return false;      }      if (topA >= bottomB)      {          return false;      }      if (rightA <= leftB)      {          return false;      }      if (leftA >= rightB)      {          return false;      }      // If none of the sides from A are outside B      return true;  } |

* Đây là nơi chúng ta kiểm tra sự va chạm trên các trục x, y. Nếu va chạm sẽ trả về true.
* Chú ý: SDL có vài hàm phát hiện va chạm dựng sẵn, nhưng trong bài này chúng ta sẽ tự triển khai.

|  |
| --- |
| // Main loop flag              bool quit = false;              // Event handler              SDL\_Event e;              // The dot that will be moving around on the screen              Dot dot;              // Set the wall              SDL\_Rect wall;              wall.x = 300;              wall.y = 40;              wall.w = 40;              wall.h = 400; |

* Trước khi vào main loop, chúng ta khai báo điểm và vị trí của tường.

|  |
| --- |
| // 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 and check collision                  dot.move(wall);                  // Clear screen                  SDL\_SetRenderDrawColor(gRenderer, 0xFF, 0xFF, 0xFF, 0xFF);                  SDL\_RenderClear(gRenderer);                  // Render wall                  SDL\_SetRenderDrawColor(gRenderer, 0x00, 0x00, 0x00, 0xFF);                  SDL\_RenderDrawRect(gRenderer, &wall);                  // Render dot                  dot.render();                  // Update screen                  SDL\_RenderPresent(gRenderer);              } |

* Đây là hàm main của chúng ta với xử lý sự kiện của điểm, di chuyển và liên tục kiểm tra va chạm với tường và cuối cùng render tường và điểm lên màn hình.

# 2. Nâng cao

Hai phần tiếp theo đây là tham khảo thêm. Những điều này dành cho xử lý va chạm nâng cao hơn so với beginner chúng ta.

* Bây giờ khi ta muốn làm cái gì đó cơ bản như tetris, kiểu phát hiện va chạm này ổn. Nhưng với những thứ như mô phỏng vật lý lại khó hơn rất nhiều.
* Với những thứ như mô phỏng cơ thể không thể bị bẻ cong, chúng ta có logic của chúng ta để làm mỗi khung hình:

1. Áp dụng tất cả lực lên cái vật trong cảnh (trọng lực, gió, lực đẩy, v.v)
2. Di chuyển vật bằng cách áp dụng gia tốc và vận tốc theo vị trí.
3. Kiểm tra va chạm cho tất cả vật thể và tạo một tập hợp các contact. Contact là một CTDL mà điển hình chứa pointers để hai vật thể va chạm, một vector bình thường từ vật 1 sang vật 2, và lượng thâm nhập của các vật.
4. Lấy tập hợp các liên hệ ghi nhận được và giải quyết các va chạm. Điều này thường liên quan đến việc kiểm tra lại các liên hệ (trong giới hạn) và giải quyết chúng.

* Bây giờ nếu như bạn hầu như không học cách phát hiện va chạm, điều này sẽ nằm ngoài tầm với bạn bây giờ. Điều này sẽ cần cả một bộ hướng dẫn (mà hiện tại ta không có thời gian để làm) để giải thích nó. Không những vậy, nó còn liên quan đến vector trong toán và lý. Chỉ cần ghi nhớ điều này khi bạn cần một game có nhiều vật thể va chạm và tự hỏi làm thế nào để tất cả cấu trúc cho một công cụ vật lý hoạt động.
* Một thứ khác là các box chúng ta có ở đây là AABBs hoặc trục bao ngoài box. Điều này nghĩa là chúng có cạnh được căn với trục x và y. Nếu bạn muốn có box được xoay, bạn có thể vẫn dùng kiểm tra tách rời trên OBBs (oriented bounding boxes – hộp giới hạn định hướng). Thay vì chiếu các góc lên trục x, y, bạn sẽ chiếu các góc lên trục I, J. Bạn cũng có thể mở rộng lên thành đa giác.