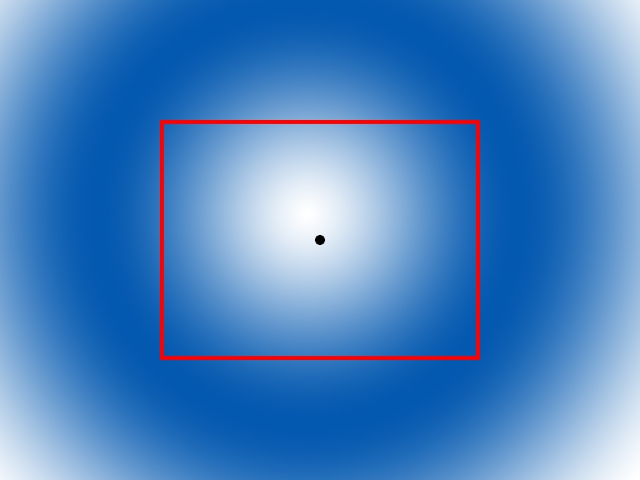
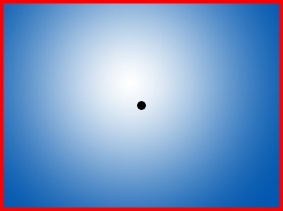
Cho tới giờ, chúng ta mới chỉ xử lý các level kích thước của màn hình. Với srcolling, bạn có thể điều hướng qua các mức độ của bất kỳ size nào bằng cách render mọi thứ so với một camera.

Nguyên tắc cơ bản của scrolling là bạn có một HCN mà có chức năng như camera:



Và sau đó bạn chỉ render cái nằm trong camera, cái mà bao gồm những thứ render so với camera hoặc chỉ hiện phần đối tượng bên trong camera.



|  |
| --- |
| //The dimensions of the level  const int LEVEL\_WIDTH = 1280;  const int LEVEL\_HEIGHT = 960;  //Screen dimension constants  const int SCREEN\_WIDTH = 640;  const int SCREEN\_HEIGHT = 480; |

Vì level không còn là size của màn hình nên chúng ta phải có một tập có biến riêng biệt để định nghĩa level size.

|  |
| --- |
| //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 relative to the camera**  **void render( int camX, int camY );**          //Position accessors          int getPosX();          int getPosY();      private:          //The X and Y offsets of the dot          int mPosX, mPosY;          //The velocity of the dot          int mVelX, mVelY;  }; |

Lần này điểm phải render so với camera, vì thế hàm render sẽ lấy vị trí camera.

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

Và lần này khi di điểm, chúng ta cần check xem điểm có ra ngoài level không thay vì ra ngoài màn hình như trước kia, vì màn hình sẽ đi chuyển xung quanh level.

|  |
| --- |
| void Dot::render( int camX, int camY )  {      //Show the dot relative to the camera      gDotTexture.render( mPosX - camX, mPosY - camY );  } |

Bây giờ, khi chúng ta render đối tượng lên màn hình, chúng ta render chúng so với camera nên sẽ trừ đi phần bù camera.

|  |
| --- |
| //Main loop flag              bool quit = false;              //Event handler              SDL\_Event e;              //The dot that will be moving around on the screen              Dot dot;              //The camera area              SDL\_Rect camera = { 0, 0, SCREEN\_WIDTH, SCREEN\_HEIGHT }; |

Trước khi chúng ta vào main loop, chúng ta định nghĩa điểm và camera mà sẽ dùng.

|  |
| --- |
| //Move the dot                  dot.move();                  //Center the camera over the dot                  camera.x = ( dot.getPosX() + Dot::DOT\_WIDTH / 2 ) - SCREEN\_WIDTH / 2;                  camera.y = ( dot.getPosY() + Dot::DOT\_HEIGHT / 2 ) - SCREEN\_HEIGHT / 2;                  //Keep the camera in bounds                  if( camera.x < 0 )                  {                      camera.x = 0;                  }                  if( camera.y < 0 )                  {                      camera.y = 0;                  }                  if( camera.x > LEVEL\_WIDTH - camera.w )                  {                      camera.x = LEVEL\_WIDTH - camera.w;                  }                  if( camera.y > LEVEL\_HEIGHT - camera.h )                  {                      camera.y = LEVEL\_HEIGHT - camera.h;                  } |

Sau khi ta di điểm, ta muốn thay đổi vị trí camera để điểm ở giữa. Chúng ta không muốn camera đi ra ngoài level nên ta sẽ giữ nó trong giới hạn sau khi di chuyển.

|  |
| --- |
| //Clear screen                  SDL\_SetRenderDrawColor( gRenderer, 0xFF, 0xFF, 0xFF, 0xFF );                  SDL\_RenderClear( gRenderer );                  //Render background                  gBGTexture.render( 0, 0, &camera );                  //Render objects                  dot.render( camera.x, camera.y );                  //Update screen                  SDL\_RenderPresent( gRenderer ); |

Sau khi camera ở đúng vị trí, chúng ta render phần background bên trong camera và sau đó render điểm so với vị trí camera.