# 4 Diseño

En esta sección se hace una primera aproximación a la implementación del sistema en el lenguaje elegido, C++. Por ello, en primer lugar se describen, mediante diagramas de flujo, los algoritmos de conversión de coordenadas, tanto en un sentido como en otro. La traslación de dichos algoritmos a C++ es inmediata.

Seguidamente se hace una breve introducción al lenguaje C++ porque, aunque los aspectos más curiosos ya se han descrito en los distintas subapartados de la sección 2.3 Aspectos teóricos, es interesante tener unos conocimientos básicos del lenguaje en concreto y de programación orientada a objetos en general para afrontar con garantías la sección 5 Interfaces de Usuario, donde se analizan las distintas clases y las operaciones de que disponen para implementar los distintos casos de uso.

## 4.1 Algoritmos de zoom y windowing y conversión

En esta sección se detallan los algoritmos usados para convertir coordenadas locales en globales y viceversa, así como los algoritmos que se siguen para realizar las operaciones de zoom y windowing.

### 4.1.1 Algoritmos de conversión de coordenadas

Realmente coexisten sistemas de representación distintos para describir el mismo conjunto de puntos. Uno de estos sistemas es el sistema global o absoluto de coordenadas. En él las coordenadas de un punto cualquiera constantes. El otro sistema de representación es el que se usa para localizar a un punto dentro de una imagen. La imagen tiene un sistema de representación que usa el pixel como unidad de medida. Tanto si se está visualizando una panorámica de toda la red vial como si se está analizando, gracias al zoom, un área menor, el tamaño del objeto imagen será el mismo, una vez decidido éste. Al sistema de coordenadas asociado a la imagen se le denomina sistema de coordenadas locales o relativas. El término relativas viene por el hecho de que las coordenadas de un cierto nodo en este sistema varían según se haya realizado o no un zoom y de la zona en que se realice dicho zoom, mientras que un punto cualquiera, independientemente del lugar donde aparezca representado en la pantalla, siempre tendrá las mismas coordenadas globales o absolutas.

El usuario puede interaccionar con una imagen para seleccionar un elemento (nodo o tramo). Es por ello absolutamente necesario tener una transformación bidireccional que permita obtener las coordenadas globales de un punto a partir de las coordenadas relativas y vicecersa. Por supuesto que para cualquiera de estas dos transformaciones es necesario conocer la información del entorno.

Partiendo de las coordenadas globales de un punto $(x,y)$, los pasos para transformar estas coordenadas a coordenadas relativas son: en primer lugar añadir el desplazamiento.

$$x1' = x1 + desplazamiento\_x$$
$$y1' = y1 + desplazamiento\_y$$

Se recuerda que las variables de entorno *desplazamiento_x* y *desplazamiento_y* almacenan el desplazamiento necesario que hay que añadir a las coordenadas globales de un punto para que la red vial aparezca representada en el centro de la imagen.

Una vez corregido el desplazamiento añadido, se trata solamente de transformar un sistema de coordenadas en otro escalado y desplazado. Este desplazamiento del que se habla es el que introduce el rectángulo limite. El rectángulo limite se supone que tiene como coordenadas (*lx1, lx2, ly1,ly2*). Se recuerda que este rectángulo almacena en coordenadas globales la porción de red vial que debe representarse en la imagen.

$$x1'' = (x1' - lx1) \times entorno$$

$$y1'' = Alto - (y1' - ly1) \times entorno$$

Los valores x1´´ e y1´´ son las coordenadas relativas, *Alto* es el alto de la imagen en pixeles. La variable entorno es la variable de entorno, que tenía la siguiente expresión:

$$entorno1 = \frac{Ancho\_imagen}{Ancho\_rectángulo\_limite} = \frac{Ancho\_imagen}{lx2 - lx1}$$

$$entorno2 = \frac{Alto\_imagen}{Alto\_rectángulo\_limite} = \frac{Alto\_imagen}{ly1 - ly2}$$

$$entorno = \min\{entorno1, entorno2\}$$

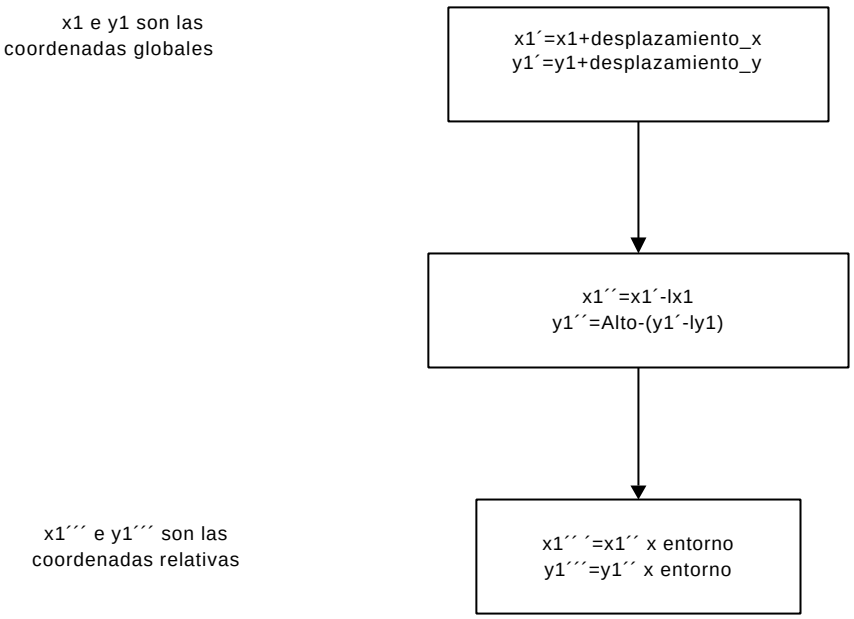


Figura 4.1.1.1: Paso de coordenadas globales a relativas

Sólo resta describir el algoritmo para transformar de coordenas relativas a coordenadas globales. Suponiendo que el punto tiene como coordenadas relativas (*x*,*y*):

*x1´=x1/entorno*
*y1´=y1/entorno*
*x1´´=x1´+lx1*
*y1´´=y1´+ly1*
*x1´´´=x1´´-desplazamiento_x*
*y1´´´=y1´´-desplazamiento_y*

El punto en coordenadas globales es el (*x1´´´, y1´´´*). El rectángulo limite se supone de nuevo que tiene como coordenadas (*lx1, lx2, ly1,ly2*). Se recuerda que este rectángulo almacena coordenadas globales.

Estos dos algoritmos representados en un diagrama de flujo quedarían como se muestra en las figuras 4.1.1.1 y 4.1.1.2.

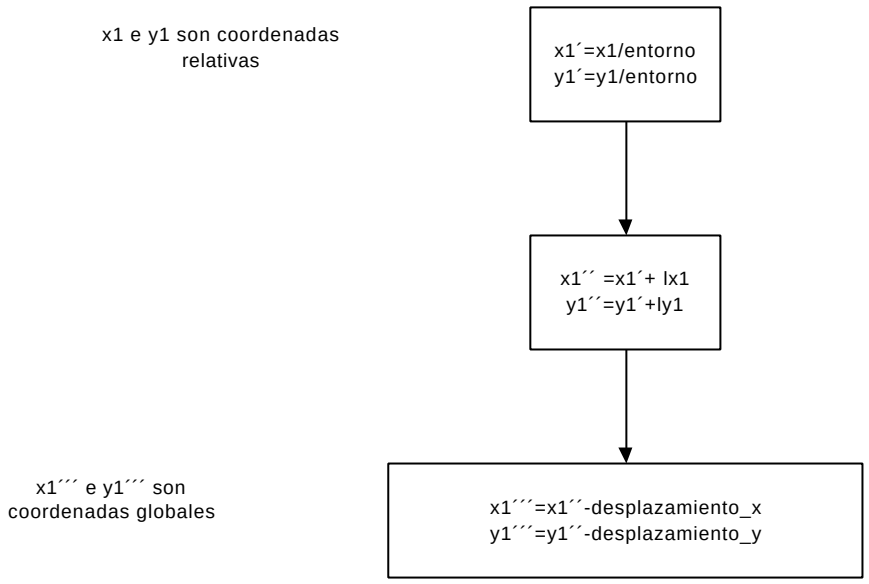


Figura 4.1.1.2: Paso de coordenadas relativas a globales

### 4.1.2 Algoritmos de zoom y windowing

Se denota por zoom la acción de ampliar, es decir, representar con mayor resolución, una porción seleccionada de una imagen. Por windowing se entiende precisamente el proceso contrario, es decir, reducir la resolución, reducir la imagen.

Es claro que al tener la imagen mayor resolución serán menos los elementos mostrados en dicha imagen. Por el contrario, al realizar una reducción, aparecerán más elementos en la imagen.

El algoritmo para realizar zoom se muestra en la figura 4.1.2.2.

El algoritmo para realizar windowing no se incluye porque es análogo al de realizar zoom: la única diferencia entre ambos es el último proceso, que en el algoritmo de realizar windowing se llamaría realizar windowing en lugar de realizar zoom.

Se juega con dos imágenes, la imagen principal y la auxiliar. La imagen principal es la imagen que representa la red vial en pantalla, ya sea en el interfaz de representación gráfica o en cualquier otro interfaz. La imagen auxiliar se usa para llevar a cabo el algoritmo, pero no es visible. La activación o desactivación se hace mediante el indicador booleano *Activado_caja.*

En primer lugar, cuando el usuario pincha con el ratón sobre la imagen se produce el evento *onMouseDown*. Esto activa (*Activado_caja=true*) el zoom e inicializa el valor del rectángulo definido por las propiedades de *zoom*. En este momento dicho rectángulo queda reducido a un punto.

A partir de este instante tendrán lugar sucesivos eventos *onMouseMove* provocados por el arrastre del ratón sobre la imagen para seleccionar la zona sobre la que realizar el zoom.

Como previamente se ha producido un evento *onMouseDown*, el zoom estará activado, por lo que se copia la porción de imagen limitada por el rectángulo seleccionado por el usuario para realizar zoom de la imagen auxiliar en la imagen principal.

La primera vez que ocurre esto no tiene mucho sentido, ya que la imagen auxiliar en principio estará vacía. Tampoco tiene importancia porque inicialmente el rectángulo seleccionado se reduce a un punto.

Como ya se ha dicho, en este momento tendrán lugar sucesivos eventos *onMouseMove*, ya que el usuario arrastrará el ratón sobre la imagen para seleccionar la zona a ampliar. Ahora es cuando se va a apreciar la utilidad de la imagen auxiliar.

La segunda vez que se produce el evento *onMouseMove* el usuario ya ha arrastrado el ratón lo suficiente como para seleccionar un pequeño rectángulo. En primer lugar la porción de imagen auxiliar se copia de nuevo en la imagen principal. Esto sigue sin tener mucho sentido.

A continuación se actualizan las coordenadas del zoom, es decir, la aplicación obtiene las coordenadas del rectángulo que está seleccionando el usuario.

Seguidamente se copia la porción de la imagen principal en la imagen auxiliar, teniendo en cuenta que ahora el rectángulo seleccionado ya es algo mayor. Y continuación se dibuja un rectángulo de color sobre la imagen principal.

Esto implicaría que no se vería la red vial, sólo un rectángulo de color. Lo que ocurre realmente es que siguen produciéndose eventos *onMouseMove.* Esto provoca que de nuevo se copie la imagen auxiliar en la principal. Teniendo en cuenta que entre cada evento *onMouseMove* el desplazamiento es mínimo (teóricamente un pixel), lo que esto provoca es que el usuario vea la red vial pintada sobre el rectángulo de color.

Este efecto se muestra en la figura 4.1.2.1.

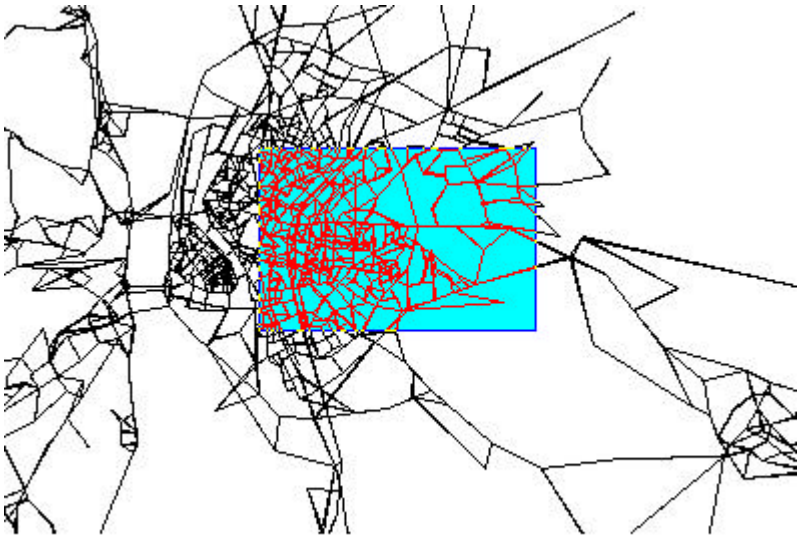

Figura 4.1.2.1: Proceso de selección de área para realizar zoom o windowing

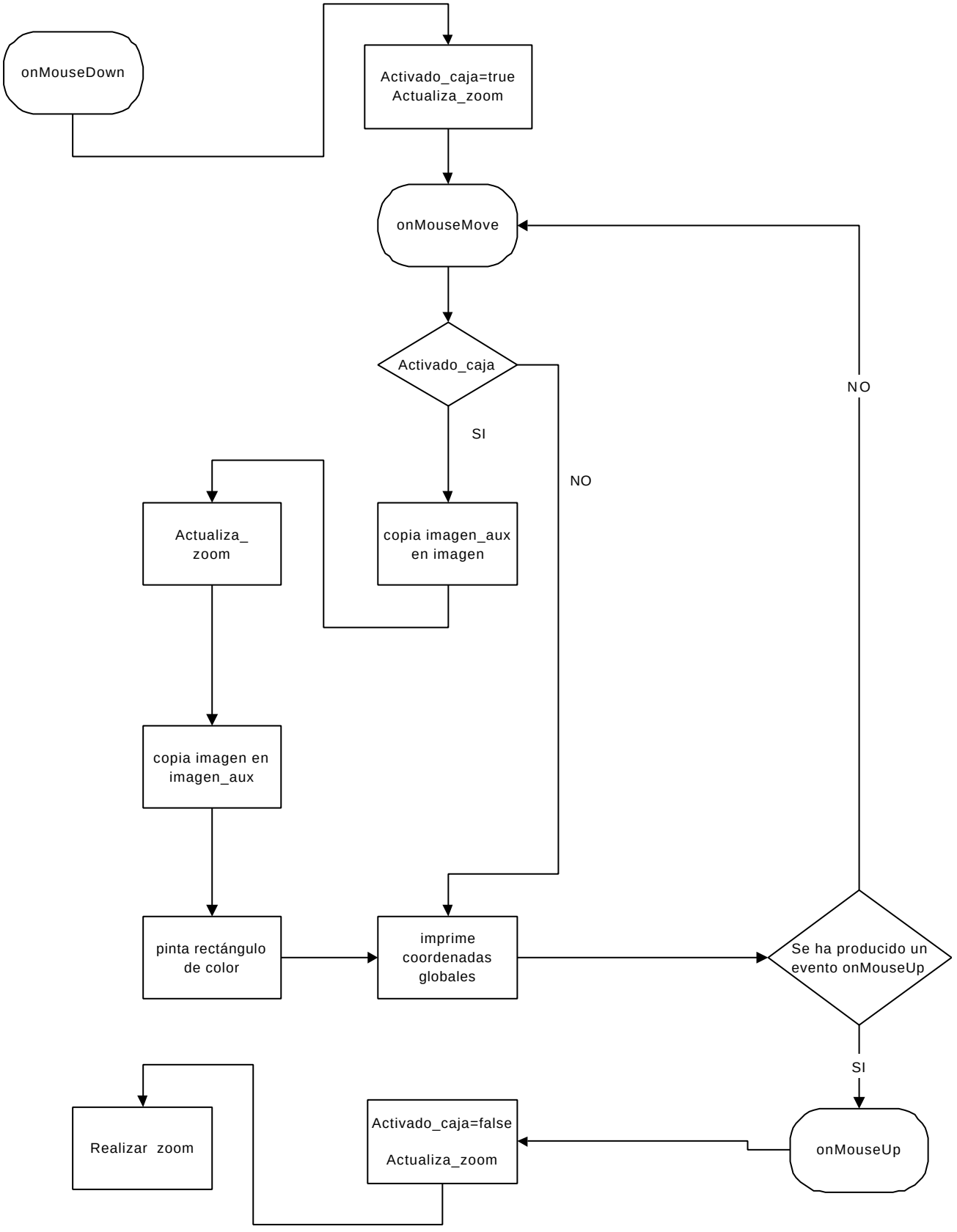


Figura 4.1.2.2: Diagrama de zoom y windowing

Finalmente, cuando el usuario haya seleccionado la zona que le interesa, liberará el botón el ratón, lo que produce un evento *onMouseUp*.

Hay que tener en cuenta que el evento *onMouseMove* se produce continuamente mientras el ratón se desplaza sobre la imagen, esté pulsado algún botón del ratón o no. De ahí la necesidad de activar y desactivar el zoom.

Cuando se produce el evento *onMouseUp* se desactiva el zoom, se toman las coordenadas del rectángulo seleccionado por el usuario, y se realiza el zoom.

En todo momento, está activado o no el zoom, cuando se produce el evento *onMouseMove* se ejecuta la acción *imprime coordenadas globales.* Realmente, se llama a una función que calcula las coordenadas globales del punto donde está el ratón y las imprime en la barra de tareas.

El proceso *Actualizacion_zoom* consiste simplemente en actualizar las coordenadas del rectángulo que ha seleccionado el usuario. Este proceso es simple: la primera vez que se llama a *Actualizacion_zoom* se almacena el punto donde está el puntero del ratón. Cada vez que se produce un evento *onMouseMove, onMouseUp* o *onMouseDown* se pueden obtener las coordenadas del punto en el que está situado el ratón.

El rectángulo seleccionado por el usuario está definido por dos puntos, los vértices superior izquierdo e inferior derecho.

A partir del evento *onMouseDown* el zoom está activado. Cuando el zoom está activado, el proceso *Actualizacion_zoom* modifica uno de los puntos a la posición donde está el ratón pero deja el otro fijo. De esta manera cuando se desactiva el zoom al llegar el evento *onMouseUp* uno de los puntos del zoom era el inicial y el otro el final.

Según la posición relativa de estos dos puntos se calcula el rectángulo que ha seleccionado el usuario y ya se puede realizar el zoom. Este cálculo se realiza en el propio proceso *Realizar_zoom* que se detalla en la figura 4.1.2.3.

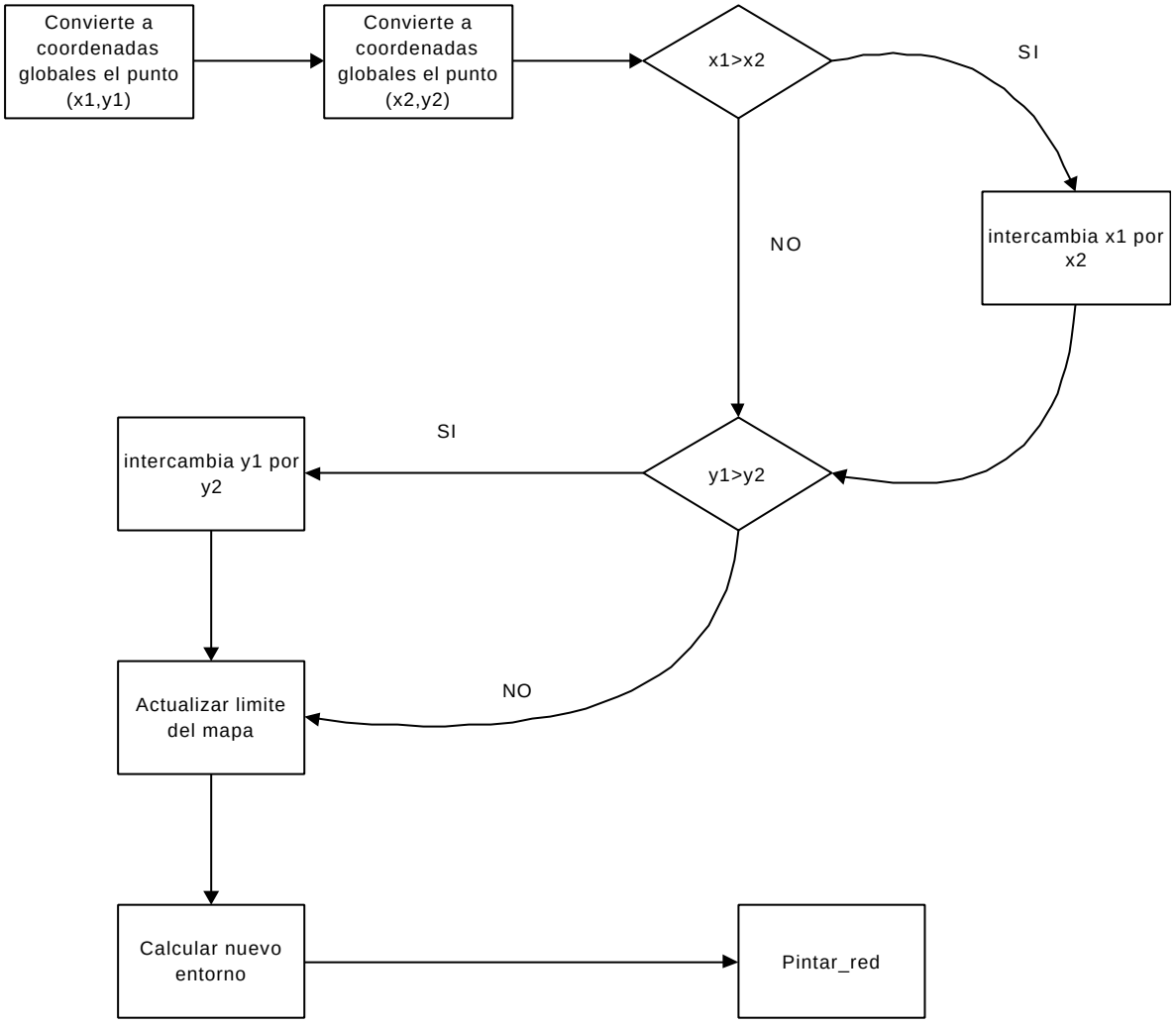


Figura 4.1.2.3: Realizar zoom

Lo primero que hace este proceso es convertir las coordenadas del rectángulo seleccionado por el usuario a coordenadas globales (esta operación se detallará a continuación).

A continuación se comprueba si *x1* es mayor que *x2*. El valor *x1* contiene la coordenada x del punto inicial del rectángulo. El punto *x2* contiene la coordenada x final del rectángulo. Algo análogo ocurre con *y1* e *y2* pero con la coordenada y.

El proceso intercambio es, como su nombre indica, intercambiar los valores.

Este proceso hace que, sea cual sea la dirección en la que el usuario ha arrastrado el ratón para seleccionar el rectángulo, los puntos (*x1*,*y1*) y (*x2*,*y2*) contienen los vértices superior izquierdo e inferior derecho del rectángulo seleccionado por el usuario, respectivamente.

Tal como se muestra en la figura 4.1.2.4, hay cuatro maneras de crear un rectángulo arrastrando el ratón. En dicha figura se representan como dos puntos gruesos el punto

origen y el final, y con una flecha dentro del rectángulo la dirección de arrastre del ratón. En el caso a), tras finalizar la selección del rectángulo, el proceso de intercambio no tiene que realizar ninguna modificación.

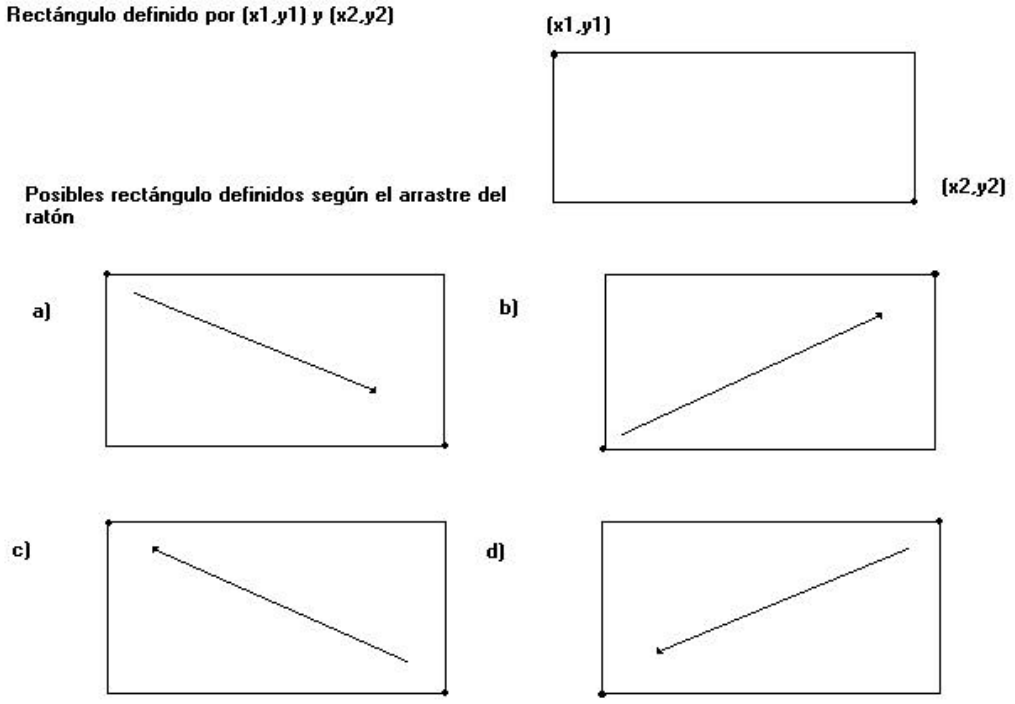


Figura 4.1.2.5: Formas de crear un rectángulo arrastrando el ratón

En el caso b) se debe intercambiar *y1* por *y2*. En el caso c) se debe intercambiar *x1* por *x2* e *y1* por *y2*. Finalmente, en el caso d) sólo es preciso intercambiar *x1* por *x2*.

Tras realizar los intercambios adecuados según el caso, se tendría el rectángulo delimitado por los puntos como aparece en la parte superior de la imagen.

El proceso actualizar limite del mapa lo que hace es modificar el valor limite: el rectángulo limite almacena el rectangulo en coordenadas globales que define la porción de red vial a representar. Las coordenadas del rectángulo zoom deben ser directamente las nuevas coordenadas del rectángulo limite. Para realizar windowing es algo más complicado, como también se verá.

A continuación se calculan las nuevas variables de entorno: el entorno y el desplazamiento necesario para representar la imagen centrada. Finalmente se representa la nueva imagen.

La figura 4.1.2.5 representa el mismo algoritmo pero para realizar windowing.

La única diferencia con el algoritmo de realizar zoom se produce a la hora de calcular el nuevo valor para el rectángulo límite. Las operaciones que se realizan para calcular ese nuevo valor son: en primer lugar calculan dos cocientes, uno para cada lado, entre el

rectángulo *límite* y la porción de área seleccionada. Estos cocientes reciben los nombres de *razon_x* y *razon_y*.

Llamando a las coordenadas del nuevo límite *Nuevo_lx1*, *Nuevo_lx2*, *Nuevo_ly1* y *Nuevo_ly2*, a las coordenadas actuales del límite *lx1*, *lx2*, *ly1* y *ly2* y denotando por *x1*, *x2*, *y1* e *y2* a las coordenadas globales del rectángulo seleccionado por el usuario:

*Nuevo_lx1=lx1-(x1-lx1)*razon_x;*
*Nuevo_lx2=lx2+(lx2-x2)*razon_x;*
*Nuevo_ly1=ly1-(y1-ly1)*razon_y;*
*Nuevo_ly2=ly2+(ly2-y2)*razon_y;*

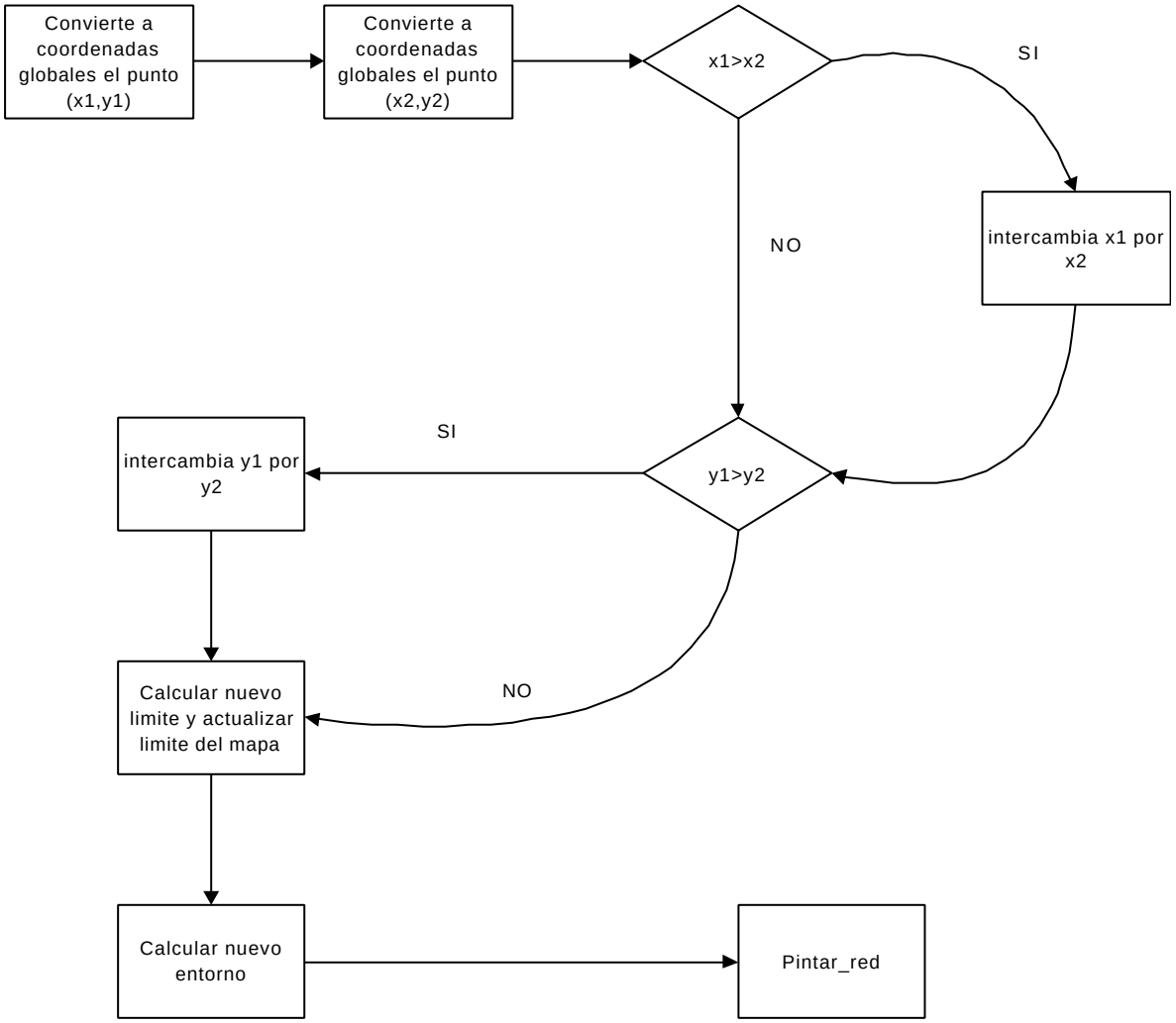


Figura 4.1.2.5: Realizar windowing