Subject: implementation of mouse click selection methods into Eeschema

Author: Lorenzo Farina
Creation date: 2010-08-16
Last upgrade: 2010-09-27
Status: proposal

Brief

Implementation of mouse selection and element highlight for EEschema graphic structures. First step to perform connection between mouse and keyboard commands with block operations.

Summary of changes

At global scope (general.h, gr_basic.h, gr_basic.cpp):

- Added *LINE_SQUARE_CAP_DIMENSION* constant (here I used a macro for code coherency, even if a const global variable is generally better...)
- Added *SELECTION_BORDER_CLEARANCE* macro constant
- Added *GR_INVERT* drawing mode (and added support into *GRSetDrawMode* function. See notes)

Into SCH_ITEM virtual class (sch_item_struct.h, sch_item_struct.cpp):

- Added pure virtual method *DrawSelected*;
- Added public functions IsLogicallySelected and IsDrawnSelected
- Added protected internal structure *selection_status* with *drawn_selected* and *logically_selected* members;
- Initialized selection_status members to false into *SCH_ITEM* constructor.

For any of SCH_ITEM directly derived class (*SCH_BUS_ENTRY*, *SCH_COMPONENT*, *SCH_FIELD*, *SCH_JUNCTION*, *SCH_LINE*, *SCH_MARKER*, *SCH_NO_CONNECT*, *SCH_POLYLINE*, *SCH_SHEET*, SCH_TEXT; files: class_schematic_item_h, class_schematic_component.h, class_marker_sch.h, class_drawsheet.h, class_drawsheet.h, class_text-label.h, class_schematic_items.cpp, class_sch_cmp_field.cpp, class_marker_sch.cpp, class_drawsheet.cpp, class_text-label.cpp):

- Implemented method *DrawSelected*
- Function *Draw* sets *selection_status.drawn_selected* to false;

Modified method *WinEDA_SchematicFrame::ReturnBlockCommand* (block.cpp) The default block command is *BLOCK_SELECT_ITEMS_ONLY* (prev. unused into EEschema; particular situation may happen in library editor; I did not understand the scope of the object selection into libedit...). ALT key is not used since it causes the window to be moved into GTK2+. CTRL is used to DRAG; SHIFT+CTRL is used instead to COPY. The SHIFT alone is used to select multiple objects.

Modified *WinEDA_SchematicFrame::OnLeftClick* (onleftclick.cpp), adding code for selection of single items.

Added *SelectDrawStruct*, SelectItemList, *GetSelectedItemsCount*, *UnselectAll and UpdatePaintSelections* public functions to the WinEDA_SchematicFrame class (wxEeschemaStruct.h). Implementation of functions in eeselect.cpp.
Added eeselect.cpp to the list of compiled source file into CmakeLists.txt

Modified *WinEDA_SchematicFrame::HandleBlockEnd* function (block.cpp) to handle BLOCK_*SELECT_ITEMS_ONLY* case statement.

Modified *RedrawStructList* (eedraw.cpp) to apply selection when the function is not called while printing.

Modified *WinEDA_DrawFrame::HandleBlockBegin* (block_commande.cpp) in order to start a new block command on *BLOCK_SELECT_ITEMS_ONLY*.

Added the escape character to the list of the hot keys available. The command processed is the selection removal for every element (this modification interests hotkeys.h and hotkeys.cpp)

Added parameter to (one overloaded versions of) function *GRRect*. It now accepts also the line style (gr_basic.h, gr_basic.cpp). Modified static function *GRSRect* accordingly.

Modified *WinEDA_SchematicFrame::OnLeftDClick*: unselected every item.

Moved *SCH_COMPONENT::Draw* virtual function from class_schematic_component.h (defined into the class declaration) to class_schematic_component.cpp.

Modified *WinEDA_DrawFrame::OnRightClick* (onrightclick.cpp). When no block command is active and at least one element is selected, a block command can be started. This is a workaround since a more accurate drag and drop functionality is planned to be coded in future. However this should allow to keep modification in a fully working environment. If no elements are selected, the behavior should be the same as "usual".

Added static function *addMenusForSelectedElements* (onrightclick.cpp) and introduced menu entries to MOVE selected objects. At this point the other operations will be started from inside the MOVE command block.

Added Ids *ID_POPUP_SCH_MOVE_SELECTED*, *ID_POPUP_SCH_DELETE_SELECTED* and *ID_POPUP_SCH_COPY_SELECTED* into the popup Ids (eeschema_id.h)

Added popup menu command handlers inside schedit.cpp (at this time only the MOVE command is really handled since this command is the starting point for other commands.

Added *BLOCK_MOVE_SELECTED* block command to the list of available

commands (block_commande.h)

Added *PickSelectedItems* function (protos.h and locate.cpp)
Processed *BLOCK_MOVE_SELECTED* command into *HandleBlockEnd* (block.cpp)

Added *BLOCK_MOVE_SELECTED* case statement to HandleBlockPlace function (block.cpp), into SetMessageBlock (block_commande.cpp).

Additional commands for block (drag, mirror, delete, rotate, save, copy) are now available even for MOVE_SELECTED command (onrightclick.cpp). Note that drag command is not working now.

Added *UnselectAll* into *TestDanglingEnds* function. This should unselect elements when the netlist changes (avoid small redrawing issues, dangling_ends.cpp).

Notes and issues

- The SCH_POLYLINE class seems to be unused. Is it to be removed?
- The drawing modes applied into GRSetDrawMode are conditionally compiled. Since the wxGraphicsContext support is not fully functional, note that even GR_INVERT mode is compiled in the same way.
- The color inversion is performed twice when the element is drawn with overlapped areas. This is not extremely clean to see, specially for text. We could choose to avoid color inversion for certain components. Moreover, we could choose to draw the component in source inversion or any other calculated color, but in this case the routine should redraw every item that could have been picked (it seems to be simple, anyway). Another way it could be the use of Blit applied over the DC in INVERT mode...
- The black on white and white on black color inversion generates an invisible selected object. In that case we could just verify and use a different drawing method and color. However, probably for related issues, it seems that in the object color list the black and the white color are not selectable.
- Instead of having a separated list of selected elements (this approach is valid when the selection is only related to a block command), I preferred to give the propriety to the *SCH_ITEM* object: in this case the flag it is not related to the block command processed but it is a status of the object itself. This should keep the concept of selection simple; a pointer or vector containing the list of selected objects (or the BLOCK_SELECTOR object) could be used anyway when block operations are to be performed.
- The DRAG function is not working with preselection lists. This is because the drag command handler needs to rebuild the item list. This behavior should be changed with a better drag method. I've not written workarounds to keep the code size limited.
- There are several minor bugs in EEschema that will be posted in future and that may have influence on the result of the application. Just two of them to be discussed:
    - The netlist should be cleaned every new segment is added. Sometimes the user could add nodes over existing connections that do not cross two lines, thus creating coincident segments. At least a function should

remove this situation placing the node in the correct place.
    - I don't know if this was a decision or a bug; however I would expect that a connection that has one of the ends exactly on another segment is CONNECTED with the segment. The same cleanup function should add a node there. If I remove a node, the node should be placed again if the lines are as described; in the same way if I remove a segment and the node is no more existing, the node symbol should be removed.

Future work:

Connection with other hotkeys for DELETE and other functionality.
Implementation of MOVE, DRAG, DELETE, MIRROR, COPY, ROTATE for every selection list;
Addition of heuristics into drag operations: lines dragged should be orthogonal if the proper flag is selected;
Drag&Drop items (selected or not);
Better selection method, specially for component labels.

Conclusion

This is what I've written. I've tested it on GTK2+ only and compiled against wxW 2.9.1. I've tried to keep it backwards compatible, but it should be easy to correct small troubles.
These modifications are not extremely useful. I hope that drag&drop functionality for each component will be considered an interesting feature, but this is just the starting point (the complete integration with the existing code would be large and must be discussed...)
I've followed coding guidelines; note that they are not in my coding style (I prefer TABS instead of spaces...), so you will have to apologize me if you find small errors (they should be limited).
I've not deleted the old code if there are reasons to leave it in the place where it was, mainly when it could be useful later. I've lust commented out it.

I hope to hear from you what you think about that.

Best regards,

Lorenzo Farina