UI infrastructure:

* controls.h :
  + defines a general control factory: a struct holding all necessary data - position, size, associated images, captions, surfaces and so on. it also defines certain behaviors, like an "on-click" handler, and a control-specific drawing function.
  + data structures for the UI tree storing (a linked list implementation of a tree including elements and lists).
* controls.c :
  + every control (window, panel, label, button) differs in the way it is initialized. we hold different initializes (new\_label, etc).
  + every control is drawn differently, and has a specific drawing function.
  + the maintanance of control data structures - creating, linking and freeing.
  + several misc. functions for control handling. **WHY not same?**

Game Logic and GUI:

* connect4\_bl.c
  + implements game functions required in project description: init, state children, score, etc.
  + implements additional functions such as victory check, game over check, move handle (validation, game object update, UI tree update), minimax call handle. There are used in our general\_game object.
* connect4\_bl.h - header file
* connect4\_ui.c
  + implements panel function required in project description.
  + implements ui victory handle - find victor, winning move, and update GUI to highlight it.
* connect4\_ui.h - header file
* ttc\_bl.c, ttc\_bl.h, ttc\_ui.c, ttc\_ui.h - much the same as the above, classes for handeling Tic Tac Toe logic and UI.
* reversi\_bl.c, reversi \_bl.h, reversi \_ui.c, reversi \_ui.h - much the same as the above, classes for handeling Reversi logic and UI.

gamesprog

* button\_actions.c - on\_click handlers for all buttons in the game. Has several types of handlers:
  + window change handlers - open new windows instead of current one, and managing game state, ui tree re-draws, etc.
  + game logic handlers - for Human involved games (not AI vs. AI), when a player makes a move, we call game-specific handlers to modify game state, and ui tree. we also call minimax, for computer moves.
* windows\_drawing.c - responsible for drawing of all UI windows
  + the game window, is initialised with a get\_default\_ui\_tree(), and then has a game-specific sub-tree added, according to chosen game.
  + run\_window is used for opening all game state invariant windows - all the menus for selecting game options, game types, etc.
  + init\_choice\_window - constructs all paramters for generic multiple selection windows (buttons for several level, etc).
  + start\_window handles ui\_tree creation for main menu
  + choice\_window - handles drawing of all choice windows, once their tree was built with with a relevan function.
  + notification\_window connstruct ui tree for notification windows
  + question window - draws and runs the constructed notification window. **RENAME**
* windows\_drawing.h
  + notification\_type enum - used to represent diferent types of notification windows - a yes/no window, or a simple notification.
  + user\_selection enum - represents the sort of menu to move to - game selection, load, etc...
* game\_enum.h - simply represents all games across several classes, when a game type is passed as a parameter.

minimax

* minimax.h
  + minimax data structures - original nodes, elements and list structures supplied for minimax in exercise 3. they are used by both by minimax, and game logic classes for creation of state children.
  + piece enum - defines the player (1, 2 or none) ,associated with a piece on the board
* minimax.c
  + data-structure implementation - allocation, linking, freeing, etc.
  + minimax tree construction - the alpha-beta function, receives several parameters, one of which is a pointer to a game-specific "state-children" function. ask david

save games

* save\_game.c, save\_game.h
  + handles the saving of a given game state to a file
  + handles reading of a saved game from files, to

check list:

1. pretty up and paramterizc contest

2. make sure reversi moves are all legal.

3.remove player paramter from scoring functions

4. c4\_copy and move : why board\_t ? why pass move\_row?

5. suggested move?!?!! this still exists in the game?

6. connect4.h - externs boarcount, main root

7. all printfs in ("ERROR")

8. what was the problem with ex3 ?

minimax.h - GROWS/COLS...

minimax.c get suggested?!

9. pause/unpause changes the caption of the button?

3. add sleep to see player change between AI V HUMAN games

controls generic button contstants

difficulties! add a label for difficulty for which player

ui files with constant control sizes

test missing files, missing fonts, corrupt files