Class Diagrams

Piece	
Attributes	bool 2D array for position
	integer for Color
	board_object to which it belongs to
Functions	get_position() where the start of the 2D position array is present in our actual board.
	place_piece(a,b, boardobj) where we place the piece on a,b coordinates of the board. (We move the piece onto the board object) The starting of the 2D array for piece is placed at the location a,b.
	check_valid() tells us if the current position of piece on the board is valid or not.

When a user touches on a piece and tries move it to a certain location, we store the position of the piece originally (if it's already on the board), then we move the piece to the certain location and if it's invalid then we move it back to its original position.
(If it's a piece not on the board yet, then we can move it back outside the board)

Board	
Attributes	bool 2D array stores which locations there are pieces already present.
	int percentage_completed.
	list of piece_objects present on the board.
Functions	add_piece(pieceObj) will add the given piece to the board. (mark positions are used on the boardObj 2D array)
	check_finish() will tell us if the entire board is completely filled or not.

When the move_piece function succeeds, ie the piece is put in a valid location on the board, then the add_piece function is called on the board which would add this piece to the list of piece_objects on our board.

Controller	
Attributes	int difficulty (of the game/level)
	int level_no
	list of piece_objects in our game
	board_object in our game.
	piece_object pickup location.
	piece_object would be the piece we currently have picked up. (would be NULL if no piece is picked)
	bool picked_piece if we are currently holding a piece.
Functions	generate_pieces(board_object) will randomly create pieces and return a list of piece_objects which we will put in our attribute piece_objects.
	initialize_board() for any other initializations required.
	pick_piece(a,b) will check if a piece exists at coordinates a,b and if it does, then it will pick up the piece.
	drop_piece(a,b) will call the move_piece(a,b,boardObj) function which will place the piece on the board if it's a valid location.
	user_input() function which reads the user's touches and on the start of a touch it calls the pick_piece(a,b) function. On the end of a touch, we call drop_piece(a,b) function (Here a,b are the coordinates on the 2D array of the board where we want to place our piece) The touch coordinates on the screen are mapped with the location of the board on the screen and if it's within the board then it's converted into the board coordinates a,b and then the drop_piece function is called.

When a piece is picked up, the pickup location stores touch converted coordinates (x,y) - piece_position coordinates(a,b) = (x-a,y-b). -> offset of pick within the piece.
So when we try place the piece onto the board, the coordinates we send to the function are (x'-(x-a),y'-(y-b)), since the place_piece function takes the leftmost coordinates of the 2D array of the piece. So if we touch the piece at location (x,y) and move it onto the board to (x',y') location then we are basically moving the position of the piece(leftmost point) to (x'-offset1,y'-offset2) where offset1=(x-a) and offset2=(y-b).

Our entire game screen is a 2D array where the board is a part of this screen which is another 2D array. The pieces initially are placed outside the board and each of them have a position(x,y) on the game screen 2D array.

So the way how this works is:

1. User touches the screen, and we convert the touch coordinates into array coordinates(x,y) and if there is a piece at that location then we call the pick_piece function.
2. Then as the user drags the piece across, we move the piece along the touch location.
3. When the touch is complete, we convert the touch coordinates to array coordinates(x,y) and if those coordinates are within the board then we call the drop_piece function.
4. This would then check if the piece is placeable on the board at that location and if it can be placed, then we place it there and we add the pieceObj to the boardObj list of pieces.
5. If the piece is not placeable at that location, ie it's not inside the board or there is another piece at that location already (it overlaps) then we put the piece back in it's original location.

Layout: (Entire screen is an array, and our board is a part of the array)

```
1111111111111111
1112222222221111
1112222222221111
1112222222221111
1112222222221111
1112222222221111
1111111111111111
1111111111111111
1133313331133311
1133313331133311
1133313331133311
1111111111111111
```

The 1's display the entire game screen as an array.
The 2's are where the board is located.
The 3's are the pieces placed on our screen initially.