**In JSFRelay.cs**

public static class JSFRelay {

static JSFGameManager gm {get{return JSFUtils.gm;}} // getter methods for gameManager reference

static JSFWinningConditions wc {get{return JSFUtils.wc;}} // getter methods for WinningConditions reference

public delegate void onEventDelegate();

public static onEventDelegate dlgOnGameStart;

public static onEventDelegate dlgOnPlayerMove;

public static onEventDelegate dlgOnBoardStabilize;

public static onEventDelegate dlgOnCombo;

public static onEventDelegate dlgOnComboEnd;

public static onEventDelegate dlgOnNoMoreMoves;

public static onEventDelegate dlgOnBoardReset;

public delegate void onEventDelegateSwipe(JSFGamePiece gp, bool isBoardCheck);

public static onEventDelegateSwipe dlgOnSwipeAdded;

public static onEventDelegateSwipe dlgOnSwipeBackTracked;

public delegate void onEventDelegateSwipeValidating(JSFGamePiece gp);

public static onEventDelegateSwipeValidating dlgOnSwipeValidating;

public delegate void onEventDelegateSwipeValidated(bool status, int ChainLength);

public static onEventDelegateSwipeValidated dlgOnSwipeValidated;

public delegate int onEventDelegateScore(int num,int x, int y);

public static onEventDelegateScore dlgOnScoreIssue;

public delegate void onEventDelegateClick(int x, int y);

public static onEventDelegateClick dlgOnPieceClick;

public static onEventDelegateClick dlgOnPanelClick;

public static void onPreGameStart (){ // called before GameManager does anything... NOTHING IS SET UP YET

dlgOnGameStart = null;

dlgOnPlayerMove = null;

dlgOnBoardStabilize = null;

dlgOnCombo = null;

dlgOnComboEnd = null;

dlgOnNoMoreMoves = null;

dlgOnBoardReset = null;

dlgOnPieceClick = null;

dlgOnPanelClick = null;

dlgOnScoreIssue = null;

dlgOnSwipeAdded = null;

dlgOnSwipeBackTracked = null;

dlgOnSwipeValidating = null;

dlgOnSwipeValidated = null;

// -----------------------------------

// your own stuff here...

//

// WARNING : board HAS NOT been set up yet... you can do final board modifications here...

// stuff like abilities that modify the current GameManager set up before board inits

// e.g., board size, board width/height etc...

// -----------------------------------

}

// when the board has been finalized, and are being initiated ( GUI output of pieces and panels )

public static void onGameStart (){

// -----------------------------------

// your own stuff here...

// -----------------------------------

if(dlgOnGameStart != null) dlgOnGameStart.Invoke();

}

public static void onPlayerMove() { // called when player makes a move

// custom piece / panels onPlayerMove function call

for (int x = 0; x < gm.boardWidth ; x++){

for (int y = 0; y < gm.boardHeight ; y++) {

if(gm.board[x,y].isFilled){

gm.board[x,y].piece.pd.onPlayerMove(gm.board[x,y]);

}

gm.board[x,y].panel.pnd.onPlayerMove(gm.board[x,y].panel);

}

}

// -----------------------------------

// your own stuff here...

// -----------------------------------

if(dlgOnPlayerMove != null) dlgOnPlayerMove.Invoke();

}

// called when all pieces stop moving and suggestion is being calculated

public static void onBoardStabilize (){

onComboEnd(); // end the combo when board stabilizes

// custom piece / panels onBoardStabilize function call

for (int x = 0; x < gm.boardWidth ; x++){

for (int y = 0; y < gm.boardHeight ; y++) {

if(gm.board[x,y].isFilled){

gm.board[x,y].piece.pd.onBoardStabilize(gm.board[x,y]);

}

gm.board[x,y].panel.pnd.onBoardStabilize(gm.board[x,y].panel);

}

}

// -----------------------------------

// your own stuff here...

// -----------------------------------

if(dlgOnBoardStabilize != null) dlgOnBoardStabilize.Invoke();

}

public static void onCombo(){ // called directly after combo+1, but before GUI output

// -----------------------------------

// your own stuff here...

// -----------------------------------

if(dlgOnCombo != null) dlgOnCombo.Invoke();

}

public static void onComboEnd(){

if(gm.currentCombo > 7){

gm.audioScript.comboHighFx.play(); // play sound for hi combo

} else if(gm.currentCombo > 5){

gm.audioScript.comboMidFx.play(); // play sound for mid combo

} else if(gm.currentCombo > 3){

gm.audioScript.comboLowFx.play(); // play sound for low combo

}

gm.currentCombo = 0; // reset combo counter...

// -----------------------------------

// your own stuff here...

// -----------------------------------

if(dlgOnComboEnd != null) dlgOnComboEnd.Invoke();

}

public static void onNoMoreMoves(){ // called before board reset happens

// -----------------------------------

// your own stuff here...

// -----------------------------------

if(dlgOnNoMoreMoves != null) dlgOnNoMoreMoves.Invoke();

}

public static void onBoardReset(){ // called after board reset happens

JSFSwipeManager.voidCurrentSwipe(); // voids any left over swipe on the old board

// -----------------------------------

// your own stuff here...

// -----------------------------------

if(dlgOnBoardReset != null) dlgOnBoardReset.Invoke();

}

public static void onPieceClick(int x, int y){

gm.board[x,y].pd.onPieceClicked(gm.board[x,y].piece); // call PieceDefinition's onPieceClick

// the panel click is here because the panel will be next to be clicked after the piece

onPanelClick(x,y); // call PanelDefinition's onPanelClick

// -----------------------------------

// your own stuff here...

// x / y is the board position of which the piece located was clicked.

// e.g., JSFUtils.gm.board[x,y] ....

// -----------------------------------

if(dlgOnPieceClick != null) dlgOnPieceClick.Invoke(x,y);

}

public static void onPanelClick(int x, int y){

gm.board[x,y].panel.pnd.onPanelClicked(gm.board[x,y].panel); // call PanelDefinition's onPanelClick

// -----------------------------------

// your own stuff here...

// x / y is the board position of which the piece located was clicked.

// e.g., JSFUtils.gm.board[x,y] ....

// -----------------------------------

if(dlgOnPanelClick != null) dlgOnPanelClick.Invoke(x,y);

}

// the "RAW" score given for destroyed pieces / matches of an individual box

// the score HAS NOT been multiplied by combo bonus yet~!

public static int onScoreIssue(int scoreGain, int x, int y){

int modifiedGains = scoreGain;

// -----------------------------------

// your own stuff here...

// -----------------------------------

// modifiedGains = something else?? ;

if(dlgOnScoreIssue != null) modifiedGains = dlgOnScoreIssue.Invoke(modifiedGains,x,y);

return modifiedGains;

}

// ==============================

// Swipe related stuff

// ==============================

// when a piece has been ADDED (already happened!) to the swipe list

public static void onSwipeAdded(JSFGamePiece gp, bool isBoardCheck){

gp.pd.onSwipeAdded(gp,isBoardCheck); // piece definition relay call

// -----------------------------------

// your own stuff here...

// you can call JSFSwipeManager for swipe related variables

// e.g. > "JSFSwipeManager.swipeColor" <-- returns slotNum of type int

// -----------------------------------

if(dlgOnSwipeAdded != null) dlgOnSwipeAdded.Invoke(gp,isBoardCheck);

}

// when a swipe was back tracked... (already happened!)

public static void onSwipeBackTracked(JSFGamePiece gp, bool isBoardCheck){

gp.pd.onSwipeBackTracked(gp,isBoardCheck); // piece definition relay call

// -----------------------------------

// your own stuff here...

// you can call JSFSwipeManager for swipe related variables

// e.g. > "JSFSwipeManager.swipeColor" <-- returns slotNum of type int

// -----------------------------------

if(dlgOnSwipeBackTracked != null) dlgOnSwipeBackTracked.Invoke(gp,isBoardCheck);

}

public static void onSwipeValidating(JSFGamePiece gp){

// -----------------------------------

// your own stuff here... before piece validation

// -----------------------------------

gp.pd.onSwipeValidating(gp); // piece definition relay call

if(dlgOnSwipeValidating != null) dlgOnSwipeValidating.Invoke(gp);

// -----------------------------------

// your own stuff here... after piece validation

// -----------------------------------

}

// when a swipe sequence has been validated.. (already happened!)

public static void onSwipeValidated(bool status, int ChainLength){

// -----------------------------------

// your own stuff here...

//

// chainLength = the number of pieces linked in the validated swipe

// status == false means the swipe sequence was rejected

// status == false means the swipe sequence was accepted and the pieces are already destroyed

//

// you can call JSFSwipeManager for swipe related variables

// e.g. > "JSFSwipeManager.swipeColor" <-- returns slotNum of type int

// -----------------------------------

if(dlgOnSwipeValidated != null) dlgOnSwipeValidated.Invoke(status,ChainLength);

}

}