\*\* global variables! \*\*

public enum JSFGameState{GamePending,GameActive,GameFinalizing,GameOver};

public class JSFGameManager : MonoBehaviour {

\*\*\* other unchanged code segments\*\*\*

// start game preparation

***void initializeGame() {***

boxPadding = 1f-(paddingPercentage/100); // set the padding value

pieceTypes = pieceManager.GetComponents<JSFPieceDefinition>();

panelTypes = panelManager.GetComponents<JSFPanelDefinition>();

// support sub-scripts initialization

audioScript = GetComponent<JSFAudioPlayer>();

animScript = GetComponent<JSFDefaultAnimations>();

// creates a 2D board

board = new JSFBoard[boardWidth,boardHeight];

//

// loop to create the board with blocks

//

switch(boardType){

case JSFBoardType.Square : /// For square type

// for the board width size

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

// for the board height size

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

// create board centralized to the game object in unity

Vector3 pos = new Vector3( x - (boardWidth/2.0f) + 0.5f, y -(boardHeight/2.0f) + 0.5f, 0);

board[x,y] = new JSFBoard(this, new int[2]{x,y}, pos\*size ) ;

//place a cube here to start with...

board[x,y].createObject(pieceTypes[0], ranType());

}

}

break;

case JSFBoardType.Hexagon : /// For hexagon type

// for the board width size

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

// for the board height size

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

Vector3 pos;

if(x%2 == 0){ // displacement for hexagon type

// create board centralized to the game object in unity

pos = new Vector3( (x - (boardWidth/2.0f) + 0.5f)\*0.865f, y -(boardHeight/2.0f) + 0.75f, 0);

} else {

// create board centralized to the game object in unity

pos = new Vector3( (x - (boardWidth/2.0f) + 0.5f)\*0.865f, y -(boardHeight/2.0f) + 0.25f, 0);

}

board[x,y] = new JSFBoard(this, new int[2]{x,y}, pos\*size ) ;

//place a cube here to start with...

board[x,y].createObject(pieceTypes[0], ranType());

}

}

break;

}

foreach(JSFBoard \_board in board){

\_board.initNeighbourReferences();

}

}

\*\*\* other unchanged code segments\*\*\*

// increase the score counter (for external scripts to update)

***public void increaseScore(int num, int x, int y) {***

num = JSFRelay.onScoreIssue(num,x,y); // relay call for modified score

if(currentCombo > 0){

num = (int) (num \* (1.5+(currentCombo/10.0)) ); // increase with multiplier from combo

}

if(JSFUtils.vm.displayScoreHUD && board[x,y].scoreHUD != null){ // display the HUD?

board[x,y].scoreHUD.display(num);

}

score += num; // add to the game score

}

\*\*\* other unchanged code segments\*\*\*

#region HEXAGON related functions

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

// HEXAGON FUNCTIONS

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

// returns the unsquiggled Hexagon grid

public int[] hexUnsquiggleArray(int[] array){

return new int[] { array[0],array[1] - array[0] + (array[0]/2)};

}

// returns a vector3 array for distance calculation

public Vector3 hexGetCalcVector(int[] array){

array = hexUnsquiggleArray(array);

return new Vector3(array[0],array[1],(array[0] + array[1])\*-1);

}

#endregion HEXAGON related functions

#region game-start sequence

public void StartGame() { // when the game is actually running...

if(gameState == JSFGameState.GamePending){

gameState = JSFGameState.GameActive; // change the state to active...

// Initialize Timers and settings

StartCoroutine(updater()); // initiate the update loop

canMove = true; // allows player to move the pieces

// call the gameStart for the board objects

foreach(JSFBoard \_board in board){

\_board.onGameStart();

}

JSFRelay.onGameStart();

} else {

Debug.Log("Game already started... cannot start the game again!");

}

}

#endregion game-start sequence

#region Unity Functions

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

// UNITY FUNCTIONS

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

***void Awake () { // board needs to be initialized before other scripts can access it***

JMFUtils.gm = this; // make a easy reference to the GameManager ( this script ! )

JMFUtils.wc = GetComponent<WinningConditions>(); // make a easy reference to the WinningConditions script~!

JMFUtils.vm = GetComponent<VisualManager>(); // make a easy reference to the VisualManager script~!

JMFRelay.onPreGameStart();

initializeGame();

preGameSetup();

// p.s. this is initially from JSFRelay.cs script (in the onGameStart() function… I moved it here

// init the board objects

foreach(JSFBoard \_board in board){

\_board.init();

}

canMove = false; // initially cannot be moved...

gameState = JSFGameState.GamePending; // game is waiting to be started...

}

void Start() { // when the game is actually running...

// Initialize Timers and settings

StartCoroutine(updater()); // initiate the update loop

StartCoroutine(boardCheckLooper()); // initiate the check loop

JSFRelay.onGameStart();

}

// Update is called once per frame

void Update () {

// woohoo~ nothing here??

}

#endregion Unity Functions

}