1. Introduction to Section 4
2. Add Rounded Corners and Reset PlayerPrefs

* Apply RoundedRectangle graphic to all the tiles on the game field.
* Try to apply Rounded Rectangle to the whole GameField – won`t work because the Game field is bigger than the tiles and Rounded Rectangle is a small resolution Image.
* Apply 9-slicing to Rounded Rectangle. Set Image Type to Sliced in the Image Script on the Game field
* 9-slicing is a technique that is used to fill large areas with small images by stretching just the central part of the image and leaving the corners of the same size for all area sizes.
* PlayerPrefs.DeleteAll() clears PlayerPrefs in code. Add this line to Awake() in ScoreTracker and the stored value of HighScore will be reset to 0.
* Don’t forget to comment or delete this line if you want to track HighScores again.

1. Coroutines Example

* All the code in simple methods is executed momentarily. To introduce delays we can use coroutines.
* Compare execution of similar code in a method and in a coroutine.
* Coroutines can be used to observe changes in certain variables or wait for certain things in the program to happen.
* Review example with ObservingCoroutine that will wait till the variable coroutineCounter reaches certain value.

1. Preview of Delays

* Preview how the game looks after introducing delays between elementary moves.

1. Delays Code Review

* Game State enum and new variables (delay, lineMoveComplete, moveMade)
* Move method now calls MoveCoroutine if delay is greater than 0.
* Move Coroutine is similar to ObserverCoroutine in the Examples. It will initiate moves in each row or column by calling MoveOneLine…() coroutines and then wait until all the corotines will finish their execution.
* MoveOneLineUpIndexCoroutine() and MoveOneLineDownIndexCoroutine() will be similar to TheCoroutine() in the previous example. They will execute elementary moves in rows or columns and insert fixed delays between them.

1. Creating Animations for Tiles

* Make all the process for one Tile
* Create three animations: Idle, Merge and Appear through the Animation window.
* Animator component will be created automatically and attached to the Tile that we are animating.
* Establish transitions between animations in the Animator window.
* Create 2 Triggers to turn on the animations from code.
* Configure all the transitions. Transitions from Idle to Merge and Appear must have HasExitTime property unchecked and must have corresponding triggers added to conditions. Transitions back to Idle must have HasExitTime checked and ExitTime must be set to 1, so when Merge or Appear animations end, Animator will switch back to Idle state. Transtion duration has to be 0 in all cases.
* Disable LoopTime for Merge and Appear animations by locating them in the Project tab.
* Copy Animator component and paste it as new for all the other Tiles.

1. Enabling Animations in Code and Testing

* Get Animator component for each Tile to access animations in code.
* Write 2 methods in the Tile script that we will call from GameManager to perform animations for this tile.
* Insert code into GameManager that will initiate animations on merged tiles and the tiles that have just appeared.
* Test the animations by switching to Unity and entering play mode.

1. Bug Fixes and Touch Input Code Review

* Fix a bug that does not let us make moves if our last move has not moved or merged any tiles.
* In the GameManager move the line that switches State to GameState.Playing to the end of the method.
* Review TouchInputManager – script that will be similar to the InputManager, that will detect swipes on touch screens in case this game will be built and run on touch devices.