

## Candy Match 3 Starter Kit

Candy Match Starter Kit is an easiest way of making match three kind of puzzle game. Swap and match colourful items to get at least 3 in a row as you play.

### Important Classes

- PlayingObject.cs
- SwapTwoObjects.cs
- GameManager.cs
- ColumnScript.cs
- ColumnManager.cs
- GameOperations.cs

## PlayingObject.cs

This script is attached to the game item (call playing object from here). It holds its placement on the board and keeps reference of its neighbour playing objects.

CheckIfCanBurst() : This function is called to check if the current playing object make a sequence of at least 3 objects in a row to explode itself.

```
public class PlayingObject : MonoBehaviour
{
    public ColumnScript myColumnScript; //reference to the columnScript to this this playing object belongs
    public int indexInColumn; //index of item in the column.
    public bool isTraced = false;
    public bool burst = false;
    public PlayingObject[] adjacentItems; //left,right,up,down
    public bool isSelected = false;
    Vector3 initialScale;

    void Start ()
    {
        initialScale = transform.localScale;
        int ind = Random.Range(0, 6);

        adjacentItems = new PlayingObject[4];
    }

    //Check if the object is place in such a way that it can form a combination that can be destroyed.
    internal bool CheckIfCanBurst(string objName,int parentIndex)...

    internal bool isMovePossible()...

    internal bool isMovePossibleInDirection(int dir)...
```

## SwapTwoObjects.cs

This script is attached to GameManager GameObject. It swaps two playing object if there is any possible combinations with that move.

```
neOperations.cs  GameManager.cs  ColumnManager.cs  SwapTwoObject.cs  PlayingObjectTouch.cs
SwapTwoObject
Start()

//Swap item1 and item2 if possible and then check the board for burst combinations.

internal void SwapTwoItems(PlayingObject item1 , PlayingObject item2)...

void ChangePositionBack()...

void Swap(PlayingObject item1, PlayingObject item2)...

}
```

## GameManager.cs

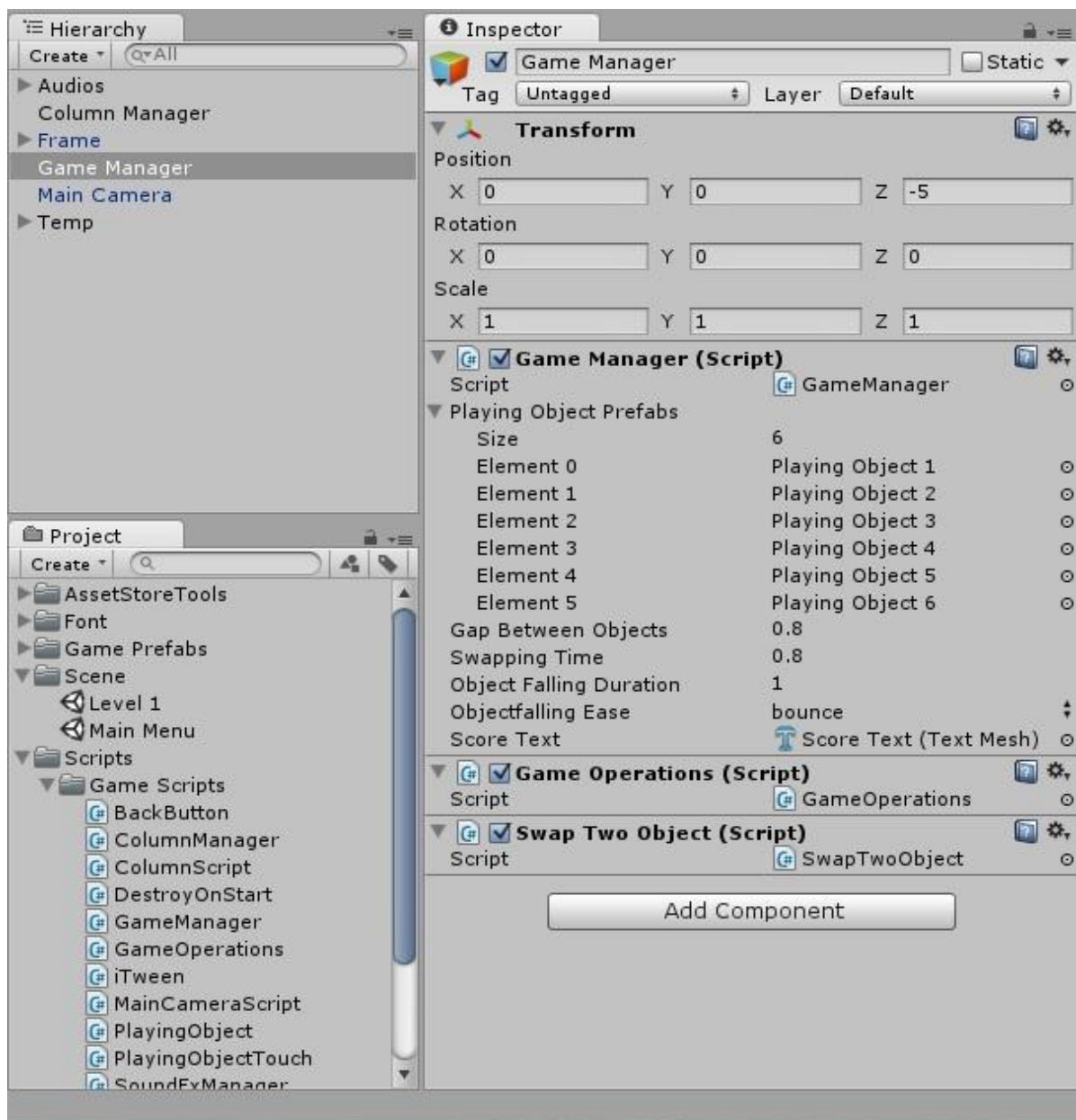
GameManager class is the manager class where you will need to assign the playing objects prefabs. As shown in the picture we have 6 different Playing Objects.

The scripts comes with other parameters which can be tweaked to change the look and feel of game play.

The class has two static attributes :

- a) numberOfColumns
- b) numberOfRows

These attributes can be set before loading the scene to let the game populate the desired number of rows and columns.

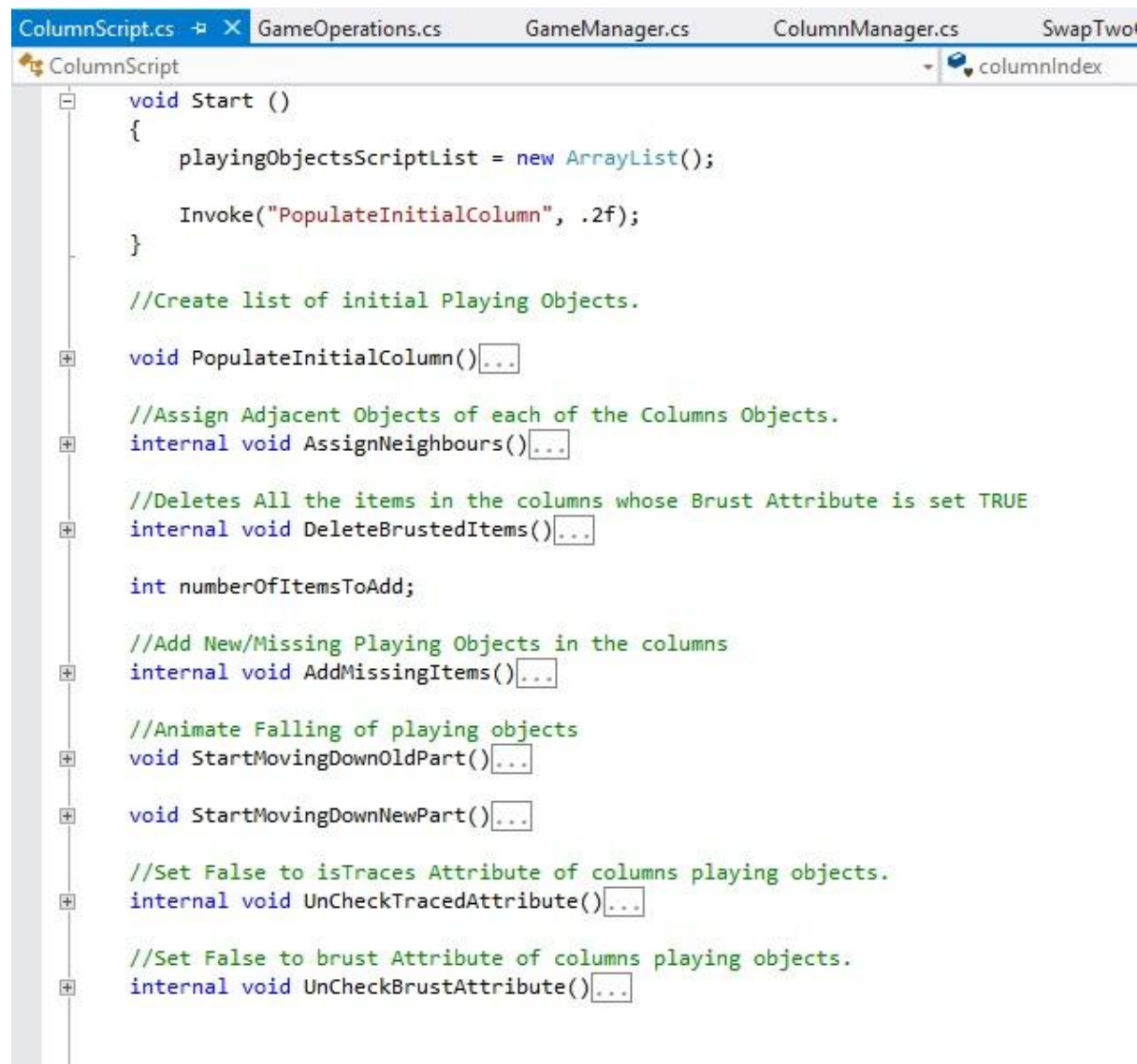


## ColumnScript.cs

Each Column in the game has ColumnScript.cs attached which takes care of all the Playing Objects in its column.

Responsibilities:

- Populate initial list of playing object (column).
- Assign Neighbour to each of the playing objects.
- Check for all the bursted playing object and remove it from the board.
- Add Missing playing objects after removing bursted one from board.



```
ColumnScript.cs X GameOperations.cs GameManager.cs ColumnManager.cs SwapTwo
ColumnScript
columnIndex

void Start ()
{
    playingObjectsScriptList = new ArrayList();

    Invoke("PopulateInitialColumn", .2f);
}

//Create list of initial Playing Objects.

void PopulateInitialColumn()...

//Assign Adjacent Objects of each of the Columns Objects.
internal void AssignNeighbours()...

//Deletes All the items in the columns whose Brust Attribute is set TRUE
internal void DeleteBrustedItems()...

int numberOfItemsToAdd;

//Add New/Missing Playing Objects in the columns
internal void AddMissingItems()...

//Animate Falling of playing objects
void StartMovingDownOldPart()...

void StartMovingDownNewPart()...

//Set False to isTraces Attribute of columns playing objects.
internal void UncheckTracedAttribute()...

//Set False to brust Attribute of columns playing objects.
internal void UncheckBrustAttribute()...
```

## ColumnManager.cs

This script adds initial number of columns and place them on board.

## GameOperations.cs

This is the master script that does all the game play operations. It checks the board state recursively and ask each of the game columns to assign neighbours for all the playing objects.

