```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

public class Tile
{
	public GameObject tileObj;
	public string type;
	public Tile(GameObject obj,string t)
	{
		tileObj = obj;
		type = t;
	}
}

public class CreateGame : MonoBehaviour	//Script must be named this!
{
	GameObject tile1 = null;
	GameObject tile2 = null;

	public GameObject[] tile;
	List<GameObject> tileBank = new List<GameObject> ();

	static int rows = 8;
	static int cols = 6;
	bool renewBoard = false;
	Tile[,] tiles = new Tile[cols, rows];


	void ShuffleList()
	{
		System.Random rand = new System.Random ();
		int r = tileBank.Count;
		while (r > 1){
			r--;
			int n = rand.Next(r + 1);
			GameObject val = tileBank[n];
			tileBank [n] = tileBank[r];
			tileBank[r] = val;
		}
	}


	// Use this for initialization
	void Start ()
	{
```

```
        int numCopies = (rows * cols) / 3;
        for (int i = 0; i < numCopies; i++) {
                for (int j = 0; j < tile.Length; j++) {
                        GameObject o = (GameObject)Instantiate (tile [j],
                                new Vector3 (-10, -10, 0),
                                tile [j].transform.rotation);
                        o.SetActive (false);
                        tileBank.Add (o);
                }
        }

        ShuffleList();

        //initalize tile grid
        for (int r = 0; r < rows; r++)
        {
                for (int c = 0; c < cols; c++)
                {
                        Vector3 tilePos = new Vector3 (c, r, 0);
                        for (int n = 0; n < tileBank.Count; n++)
                        {
                                GameObject o = tileBank [n];
                                if (!o.activeSelf)
                                {
                                        o.transform.position =
                                                new Vector3 (tilePos.x,
                                                        tilePos.y,
                                                        tilePos.z);
                                        o.SetActive (true);
                                        tiles [c, r] = new Tile (o, o.name);
                                        n = tileBank.Count + 1;
                                }
                        }
                }
        }
}
// Update is called once per frame
void Update ()
{
        CheckGrid();
        if (Input.GetMouseButtonDown (0)) {
                Ray ray = Camera.main.ScreenPointToRay
                        (Input.mousePosition);
                RaycastHit2D hit =
                        Physics2D.GetRayIntersection (ray, 1000);

                if (hit) {
```

```
                tile1 = hit.collider.gameObject;
        }
}

//If finger up is detected after
// an initial tile has been chosen
else if (Input.GetMouseButtonUp (0) && tile1) {
        Ray ray = Camera.main.ScreenPointToRay
                (Input.mousePosition);
        RaycastHit2D hit =
                Physics2D.GetRayIntersection (ray, 1000);

        if (hit) {
                tile2 = hit.collider.gameObject;
        }

        if (tile1 && tile2)
        {
                int horzDist = (int)
                        Mathf.Abs (tile1.transform.position.x -
                                tile2.transform.position.x);
                int vertDist = (int)
                        Mathf.Abs (tile1.transform.position.y -
                                tile2.transform.position.y);
                if (horzDist == 1 ^ vertDist == 1) {
                        Tile temp = tiles [(int)tile1.transform.position.x,
                                (int)tile1.transform.position.y];
                        tiles [(int)tile1.transform.position.x,
                                (int)tile1.transform.position.y] =
                                tiles [(int)tile2.transform.position.x,
                                        (int)tile2.transform.position.y];
                        tiles [(int)tile2.transform.position.x,
                                (int)tile2.transform.position.y] = temp;

                        Vector3 tempPos = tile1.transform.position;
                        tile1.transform.position =
                                tile2.transform.position;
                        tile2.transform.position = tempPos;
                        //reset the touched tiles
                        tile1 = null;
                        tile2 = null;
                }
                else
                {
                        GetComponent<AudioSource> ().Play ();
                }
        }
```

```
        }
}

void CheckGrid()
{
        int counter = 1;
        //check in columns
        for (int r = 0; r < rows; r++) {
                counter = 1;
                for (int c = 1; c < cols; c++) {
                        if (tiles [c, r] != null && tiles [c - 1, r] != null)
                        {                                       //if the tiles exsist
                                if (tiles [c, r].type == tiles [c - 1, r].type) {
                                        counter++;
                                } else
                                        counter = 1; //reset counter
                                //if three are found remove them
                                if (counter == 3) {
                                        if (tiles [c, r] != null)
                                                tiles [c, r].tileObj.SetActive (false);
                                        if (tiles [c - 1, r] != null)
                                                tiles [c - 1, r].tileObj.SetActive
                                                (false);
                                        if (tiles [c - 2, r] != null)
                                                tiles [c - 2, r].tileObj.SetActive
                                                (false);
                                        tiles [c, r] = null;
                                        tiles [c - 1, r] = null;
                                        tiles [c - 2, r] = null;
                                        renewBoard = true;
                                }
                        }
                }
        }
        //check in rows
        for (int c = 0; c < cols; c++)
        {
                counter = 1;
                for (int r = 1; r < rows; r++)
                {
                        if (tiles[c,r] != null && tiles[c, r-1]
                                != null)
                                //if tiles exsist
                        {
                                if (tiles [c, r].type == tiles [c, r - 1].type)
                                {
                                        counter++;
```

```
                                }
                                else
                                        counter = 1; //reset counter
                                //if three are found remove them
                                if (counter == 3) {
                                        if (tiles [c, r] != null)
                                                tiles [c, r].tileObj.SetActive
                                                (false);
                                        if (tiles [c, r - 1] != null)
                                                tiles [c, r - 1].tileObj.SetActive
                                                (false);
                                        if (tiles [c, r - 2] != null)
                                                tiles [c, r - 2].tileObj.SetActive
                                                (false);
                                        tiles [c, r] = null;
                                        tiles [c, r - 1] = null;
                                        tiles [c, r - 2] = null;
                                        renewBoard = true;
                                }
                        }
                }
        }
        if (renewBoard)
        {
                RenewGrid();
                renewBoard = false;
        }
}

void RenewGrid()
{
        bool anyMoved = false;
        ShuffleList ();
        for (int r = 1; r < rows; r++) {
                for (int c = 0; c < cols; c++) {
                        if (r == rows - 1 && tiles [c, r] == null) {
//if in the top row and no tile
                                Vector3 tilePos = new Vector3 (c, r, 0);
                                for (int n = 0; n < tileBank.Count; n++) {
                                        GameObject o = tileBank [n];
                                        if (!o.activeSelf) {
                                                o.transform.position = new
                                                        Vector3 (tilePos.x, tilePos.y,
                                                                tilePos.z);
                                                o.SetActive (true);
                                                tiles [c, r] = new Tile (o, o.name);
                                                n = tileBank.Count + 1;
```

```
                                        }
                                    }
                                }
                                if (tiles [c, r] != null) {
                                    //drop down if space below is empty
                                    if (tiles [c, r - 1] == null) {
                                        tiles [c, r - 1] = tiles [c, r];
                                        tiles [c, r - 1].tileObj.transform.
                                        position = new Vector3 (c, r - 1, 0);
                                        tiles [c, r] = null;
                                        anyMoved = true;
                                    }
                                }
                            }
                        }
                        if (anyMoved)
                        {
                            Invoke ("RenewGrid", 0.5f);
                        }
                    }
}
```