Oude code (zonder onderzoek Design Patterns)

using System.Collections;  
using System.Collections.Generic;  
using UnityEngine;  
  
public class CellSplit : MonoBehaviour {  
  
    public float splitTime = 5f;  
    private float TimeLevel0 = 0.5f;  
    private float TimeLevel1 = 1.0f;  
    private float TimeLevel2 = 2.5f;  
    private float TimeLevel3 = 3.5f;  
    private float TimeLevel4 = 5.0f;  
    private float TimeLevel5 = 8.0f;  
  
    public float splitSize = 1f;  
    private float CellSize0 = 0.125f;  
    private float CellSize1 = 0.110f;  
    private float CellSize2 = 0.090f;  
    private float CellSize3 = 0.075f;  
    private float CellSize4 = 0.060f;  
    private float CellSize5 = 0.045f;  
  
    private float X = 5f;  
    private float Y = 8f;  
  
    public int cellLevel = 0;  
    public int cellKind = 0;  
  
    public GameObject Cell;  
    private GameObject[] CellArray;  
  
    public bool CoreCell = false;  
  
    private GameObject CCO;  
    public GameObject ParentCell;  
  
  
  
    *//Initialization*  
    public void Awake () {  
        CCO = GameObject.Find ("CellCollection");  
        CellArray = new GameObject[2];  
  
        if (CoreCell) {  
            ParentCell = gameObject;  
            StartSplit ();  
        }  
    }  
          
  
    *//Generate 2 cells below*  
    private void Split(){  
        CellOneSplit ();  
        CellTwoSplit ();  
    }  
  
  
    *//Generate the first cell*  
    private void CellOneSplit(){  
        *//Create Cell 1*  
        Vector2 CellPos1 = new Vector2 (transform.position.x - (splitSize \* X), transform.position.y -  (splitSize \* Y));  
        GameObject Cell1 = GameObject.Instantiate (Cell, CellPos1, Quaternion.identity);  
  
        *//Set Parent*  
        Cell1.transform.SetParent(CCO.transform);  
  
        *//Set Cell properties*  
        CellSplit CS1 = Cell1.GetComponent<CellSplit>();  
        CS1.cellLevel = cellLevel + 1;  
        CS1.ParentCell = gameObject;  
        CS1.cellKind = 1;  
  
        *//Start CellSplit*  
        CS1.StartSplit();  
    }  
  
  
    *//Generate the second cell*  
    private void CellTwoSplit(){  
        *//Create Cell 2*  
        Vector2 CellPos2 = new Vector2 (transform.position.x + (splitSize \* X), transform.position.y -  (splitSize \* Y));  
        GameObject Cell2 = GameObject.Instantiate (Cell, CellPos2, Quaternion.identity);  
  
        *//Set Parent*  
        Cell2.transform.SetParent(CCO.transform);  
  
        *//Set Cell properties*  
        CellSplit CS2 = Cell2.GetComponent<CellSplit>();  
        CS2.cellLevel = cellLevel + 1;  
        CS2.ParentCell = gameObject;  
        CS2.cellKind = 2;  
  
        *//Start CellSplit*  
        CS2.StartSplit();  
    }  
  
  
    *//Start the splitting process*  
    public void StartSplit(){  
        SetSplitValues ();  
        StartCoroutine (Timer ());  
    }  
  
  
    *//Set SplitTime and CellSize*  
    private void SetSplitValues(){  
        switch (cellLevel){  
        case 0: splitTime = TimeLevel0; splitSize = CellSize0;   
            break;  
        case 1: splitTime = TimeLevel1; splitSize = CellSize1;  
            break;  
        case 2: splitTime = TimeLevel2; splitSize = CellSize2;  
            break;  
        case 3: splitTime = TimeLevel3; splitSize = CellSize3;  
            break;  
        case 4: splitTime = TimeLevel4; splitSize = CellSize4;  
            break;  
        case 5: splitTime = TimeLevel5; splitSize = CellSize5;  
            break;  
        }  
  
        transform.localScale = new Vector3 (splitSize, splitSize, splitSize);  
    }  
  
  
    *//Timer before Splitting*  
    private IEnumerator Timer(){  
        yield return new WaitForSeconds (splitTime);  
        if (cellLevel < 5) {  
            Split ();  
        }  
    }  
  
  
    *//Set timer for Single Splitting*  
    private IEnumerator SingleTimer(int Kind){  
        yield return new WaitForSeconds (splitTime);  
        if (Kind == 1) {  
            CellOneSplit ();  
        }  
        if (Kind == 2) {  
            CellTwoSplit ();  
        }  
    }  
  
  
    *//When Child has been destroyed*  
    public void Dead(int Kind){  
        StartCoroutine (SingleTimer (Kind));  
    }  
  
}*//CLASS*

Nieuwe Code (Re-design van code-structuur)

using System.Collections;  
using System.Collections.Generic;  
using UnityEngine;  
  
public class CellSplitting : MonoBehaviour {  
  
    public GameObject ParentCell;  
    public GameObject ChildCell1;  
    public GameObject ChildCell2;  
  
    public float splitTime = 1f;  
  
    public bool hasParent = true;  
    public bool hasChildren = true;  
  
    public bool inSplitTime1 = false;  
    public bool inSplitTime2 = false;  
  
    public bool FrozenCell = false;  
  
    private SpriteRenderer SR;  
    public Sprite CellNormal;  
    public Sprite CellFrozen;  
  
  
    *//Initialization*  
    private void Awake () {  
        *//Components*  
        SR = GetComponent<SpriteRenderer>();  
  
        *//Security*  
        if (!hasParent) {  
            ParentCell = gameObject;  
        }  
        if (!hasChildren) {  
            ChildCell1 = gameObject;  
            ChildCell2 = gameObject;  
        }  
  
        *//Initialization*  
        if (hasParent) {  
            ParentCell = transform.parent.gameObject;  
        }  
        if (hasChildren) {  
            ChildCell1 = transform.GetChild (0).gameObject;  
            ChildCell2 = transform.GetChild (1).gameObject;  
        }  
    }  
  
  
    *// Update is called once per frame*  
    private void Update () {  
        if (hasChildren) {  
            CheckActive ();  
            CheckFrozen ();  
        }  
    }  
  
  
    *//Checks if Cell1 or Cell2 has been destroyed*  
    private void CheckActive(){  
        if (!ChildCell1.activeSelf && !inSplitTime1) {  
            StartCoroutine (SplitTime (1));  
            inSplitTime1 = true;  
        }  
        if (!ChildCell2.activeSelf && !inSplitTime2) {  
            StartCoroutine (SplitTime (2));  
            inSplitTime2 = true;  
        }  
    }  
          
  
    *//Respawn the cell after a period of time*  
    private IEnumerator SplitTime(int WhichCell){  
        yield return new WaitForSeconds (splitTime);  
        if (gameObject.activeSelf) {  
            switch (WhichCell) {  
            case 1:  
                ChildCell1.SetActive (true);  
                inSplitTime1 = false;  
                break;  
            case 2:  
                ChildCell2.SetActive (true);  
                inSplitTime2 = false;  
                break;  
            }      
        }   
    }  
  
  
    private void CheckFrozen(){  
        if (!ChildCell1.activeSelf && !ChildCell2.activeSelf) {  
            SR.sprite = CellNormal;  
            FrozenCell = false;  
        }   
        else {  
            SR.sprite = CellFrozen;  
            FrozenCell = true;  
        }  
    }  
          
}*//CLASS*