```
// Chasing_Camera.cs

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

public class Chasing_Camera : MonoBehaviour
{
    public float m_distance = 0.0f;
    public float m_height;
    public float m_Time = 0.0f;
    //  灯篭の平均値
    public float m_RangeOfAverage = 0.0f;
    public float m_MoveLerpRate = 0.01f;
    public float m_SmoothTime;
    public float m_Half2Time;

    private Vector3 m_PreviousTarget;
    private Vector3 m_targetPosition;



    private Vector3 m_Current;
    private Vector3 currentVelocity = Vector3.zero;

    private Vector3 previousCenterPistion;
    ///8/15追加//////////
    public bool cameraPlan = false;
    ////ここまで///////////

    // Use this for initialization
    void Start()
    {
        //  現在の灯篭の座標を取得する
        m_Current = mostDitances();
        Camera.main.transform.LookAt(m_Current);
    }
```

```
    // Update is called once per frame
    void Update()
    {
        // カメラから最も遠い座標
        Vector3 cameraPosition = mostDitances();
        Vector3 velocity;

        // カメラから一番遠い座標か中間点に対して追従を行う
        updateCamera(cameraPosition);
        velocity = cameraPosition - m_PreviousTarget;
        velocity.Normalize();
        m_targetPosition = m_PreviousTarget + velocity;
        m_PreviousTarget = m_targetPosition;

        // 現在位置から目標位置までベクトルを徐々に変化させ追尾を行う
        m_Current = Vector3.SmoothDamp(m_Current, cameraPosition, ref currentVelocity,
m_SmoothTime);
        Camera.main.transform.LookAt(m_Current);
        cameraPlan = true;
    }

    /// <summary>
    /// カメラから最も遠い灯篭の座標を探す
    /// またはカメラから最も遠い灯篭が二個以上ある場合、最も遠い座標の中間点を探す
    /// </summary>
    /// <returns></returns>
    public Vector3 mostDitances()
    {
        //沈んでいない灯篭をリストに追加
        List<Tourou> tourous = new List<Tourou>();
        foreach (GameObject tourou in GameObject.FindGameObjectsWithTag("Tourou"))
        {
            Tourou t = tourou.GetComponent<Tourou>();
            if (t.tourouDown) continue;
            tourous.Add(t);
```

```
}

//一番遠い灯篭との距離
float mostFarDistance = 0.0f;
//一番遠い灯篭の座標
Vector3 mostFarPosition = transform.position;
foreach (Tourou t in tourous)
{
    Vector3 tourouPosition = t.transform.position;
    //灯篭とカメラとの距離
    float distance_ = (this.transform.position - tourouPosition).magnitude;

    //遠かったら変数に代入
    if (mostFarDistance <= distance_)
    {
        mostFarDistance = distance_;
        mostFarPosition = t.transform.position;
    }

    Debug.DrawLine(this.transform.position, tourouPosition, Color.red);
}

//一番遠い灯篭に近い灯篭をリストに追加
List<Vector3> mostFarDistances = new List<Vector3>();
mostFarDistances.Add(mostFarPosition);
foreach (Tourou t in tourous)
{
    Vector3 tourouPosition = t.transform.position;
    float length = Vector3.Distance(mostFarPosition, tourouPosition);

    if (length <= m_RangeOfAverage)
    {
        mostFarDistances.Add(tourouPosition);
    }
}
```

```
        //遠い灯篭の平均？中間地点の座標
        Vector3 totalPosition = new Vector3();
        foreach (Vector3 p in mostFarDistances)
        {
            totalPosition += p;
        }

        //中間だったらいいなぁ↓
        Vector3 cameraPosition = totalPosition / mostFarDistances.Count;

        return cameraPosition;
    }


    // カメラから一番遠い座標か中間点に対して追従を行う
    public void updateCamera(Vector3 targetPosition)
    {
        Vector3 velocity = transform.position - targetPosition;

        velocity.Normalize();

        Vector3 idealPos = velocity * m_distance + targetPosition;

        idealPos.y = m_height;

        transform.position = Vector3.Lerp(transform.position, idealPos, m_MoveLerpRate);
    }
}
```