FollowCam.cs

```
using UnityEngine;
using System.Collections;

public class FollowCam : MonoBehaviour {
    static public FollowCam    S; // a FollowCam Singleton

    // fields set in the Unity Inspector pane
    public float            easing = 0.01f;
    public Vector2          minXY;
    public bool  _____________________________;

    // fields set dynamically
    public GameObject       poi; // The Point of Interest
    public float            camZ; // The desired Z pos of the Camera

    void Awake() {
        S = this;
        camZ = this.transform.position.z;
    }

    void FixedUpdate () {
        Vector3 destination;
        // If there is no poi, return to P:[0,0,0]
        if (poi == null) {
            destination = Vector3.zero;
        }  else {
            // Get the position of the poi
            destination = poi.transform.position;
            // If poi is a Projectile, check to see if it's at rest
            if (poi.tag == "Projectile") {
                // if it is sleeping (i.e. not moving)
                if ( poi.GetComponent<Rigidbody>().IsSleeping() ) {
                    // show wide angle
                    poi = null;
                    MissionDemolition.SwitchView("Both");
                    return;
                }
            }
        }
        // Limit the X & Y to minimum values
        destination.x = Mathf.Max( minXY.x, destination.x );
        destination.y = Mathf.Max( minXY.y, destination.y );
        // Interpolate from the current Camera position towards destination
        destination = Vector3.Lerp( transform.position, destination, easing );
        // Retain a destination.z of camZ
        destination.z = camZ;
        // Set the Camera to the destination
        transform.position = destination;
        // Set the orthographicSize of the Camera to keep Ground in view
        this.GetComponent<Camera>().orthographicSize = destination.y + 10;
    }
}
```

Goal.cs

```
using UnityEngine;
using System.Collections;

public class Goal : MonoBehaviour {
    // A static field accessible by code anywhere
    static public bool        goalMet = false;

    void OnTriggerEnter( Collider other ) {
        // When the Trigger is hit by something
        // Check to see if it's a Projectile
        if ( other.gameObject.tag == "Projectile" ) {
            // If so, set goalMet to true
            Goal.goalMet = true;
            // Also set the alpha of the color to higher opacity
            Color c = GetComponent<Renderer>().material.color;
            c.a = 0.9f;
            GetComponent<Renderer>().material.color = c;

        }
    }
}
```

ProjectileLine.cs

```
using UnityEngine;
using System.Collections;
// Remember, the following line is needed to use Lists
using System.Collections.Generic;

public class ProjectileLine : MonoBehaviour {
    static public ProjectileLine S; // Singleton

    // fields set in the Unity Inspector pane
    public float            minDist = 0.1f;
    public bool  ____________________________;

    // fields set dynamically
    public LineRenderer     line;
    private GameObject      _poi;
    public List<Vector3>    points;

    void Awake() {
        S = this;
        // Get a reference to the LineRenderer
        line = GetComponent<LineRenderer>();
        // Disable the LineRenderer until it's needed
        line.enabled = false;
        // Initialize the points List
        points = new List<Vector3>();
    }

    // This is a property (i.e. a method masquerading as a field)
    public GameObject poi {
        get {
            return( _poi );
        }
        set {
            _poi = value;
            if ( _poi != null ) {
                // When _poi is set to something new, it resets everything
                line.enabled = false;
                points = new List<Vector3>();
                AddPoint();
            }
        }
    }

    // This can be used to clear the line directly
    public void Clear() {
        _poi = null;
        line.enabled = false;
        points = new List<Vector3>();
    }

    public void AddPoint() {
        // This is called to add a point to the Line
        Vector3 pt = _poi.transform.position;
        if ( points.Count > 0 && (pt - lastPoint).magnitude < minDist ) {
            // If the point isn't far enough from the last point, it returns
            return;
        }
        if ( points.Count == 0 ) {
            // If this is the launch point...
            Vector3 launchPos = Slingshot.S.launchPoint.transform.position;
```

```
61              Vector3 launchPosDiff = pt - launchPos;
62              // ...it adds an extra bit of line to aid aiming later
63              points.Add( pt + launchPosDiff );
64              points.Add(pt);
65              line.SetVertexCount(2);
66              // Sets the first two points
67              line.SetPosition(0, points[0] );
68              line.SetPosition(1, points[1] );
69              // Enables the LineRenderer
70              line.enabled = true;
71          }  else {
72              // Normal behavior of adding a point
73              points.Add( pt );
74              line.SetVertexCount( points.Count );
75              line.SetPosition( points.Count-1, lastPoint );
76              line.enabled = true;
77          }
78      }
79
80      // Returns the location of the most recently added point
81      public Vector3 lastPoint {
82          get {
83              if (points == null) {
84                  // If there are no points, returns Vector3.zero
85                  return( Vector3.zero );
86              }
87              return( points[points.Count-1] );
88          }
89      }
90
91      void FixedUpdate () {
92          if ( poi == null ) {
93              // If there is no poi, search for one
94              if (FollowCam.S.poi != null) {
95                  if (FollowCam.S.poi.tag == "Projectile") {
96                      poi = FollowCam.S.poi;
97                  }  else {
98                      return; // Return if we didn't find a poi
99                  }
100             }  else {
101                 return; // Return if we didn't find a poi
102             }
103         }
104         // If there is a poi, it's loc is added every FixedUpdate
105         AddPoint();
106         if ( poi.GetComponent<Rigidbody>().IsSleeping() ) {
107             // Once the poi is sleeping, it is cleared
108             poi = null;
109         }
110     }
111 }
```

Slingshot.cs

```
using UnityEngine;
using System.Collections;

public class Slingshot : MonoBehaviour {
    static public Slingshot    S;

    // fields set in the Unity Inspector pane
    public GameObject        prefabProjectile;
    public float             velocityMult = 4f;
    public bool  _____________________________;
    // fields set dynamically
    public GameObject        launchPoint;
    public Vector3           launchPos;
    public GameObject        projectile;
    public bool              aimingMode;

    void Awake() {
        // Set the Slingshot Singleton S
        S = this;

        Transform launchPointTrans = transform.FindChild("LaunchPoint");
        launchPoint = launchPointTrans.gameObject;
        launchPoint.SetActive( false );
        launchPos = launchPointTrans.position;
    }

    void OnMouseEnter() {
        //print("Slingshot:OnMouseEnter()");
        launchPoint.SetActive( true );
    }

    void OnMouseExit() {
        //print("Slingshot:OnMouseExit()");
        launchPoint.SetActive( false );
    }

    void OnMouseDown() {
        // The player has pressed the mouse button while over the Slingshot
        aimingMode = true;
        // Instantiate a Projectile
        projectile = Instantiate( prefabProjectile ) as GameObject;
        // Start it at the launchPoint
        projectile.transform.position = launchPos;
        // Set it to isKinematic for now
        projectile.GetComponent<Rigidbody>().isKinematic = true;
    }

    void Update() {
        // If the Slingshot is not in aimingMode, don't run this code
        if (!aimingMode) return;
        // Get the current mouse position in 2D screen coordinates
        Vector3 mousePos2D = Input.mousePosition;
        // Convert the mouse position to 3D world coordinates
        mousePos2D.z = -Camera.main.transform.position.z;
        Vector3 mousePos3D = Camera.main.ScreenToWorldPoint( mousePos2D );
        // Find the delta from the launchPos to the mousePos3D
        Vector3 mouseDelta = mousePos3D-launchPos;
        // Limit the mouseDelta to the radius of the Slingshot SphereCollider
        float maxMagnitude = this.GetComponent<SphereCollider>().radius;
        if (mouseDelta.magnitude > maxMagnitude) {
```

```
61              mouseDelta.Normalize();
62              mouseDelta *= maxMagnitude;
63          }
64          // Move the Projectile to this new position
65          Vector3 projPos = launchPos + mouseDelta;
66          projectile.transform.position = projPos;
67
68          if ( Input.GetMouseButtonUp(0) ) {
69              // The mouse has been released
70              aimingMode = false;
71              projectile.GetComponent<Rigidbody>().isKinematic = false;
72              projectile.GetComponent<Rigidbody>().velocity = -mouseDelta * velocityMult;
73              FollowCam.S.poi = projectile;
74              projectile = null;
75              MissionDemolition.ShotFired();
76          }
77      }
78
79  }
```