Daniel Giaime
Final Project Check-off
Tuesday, December 5, 2017

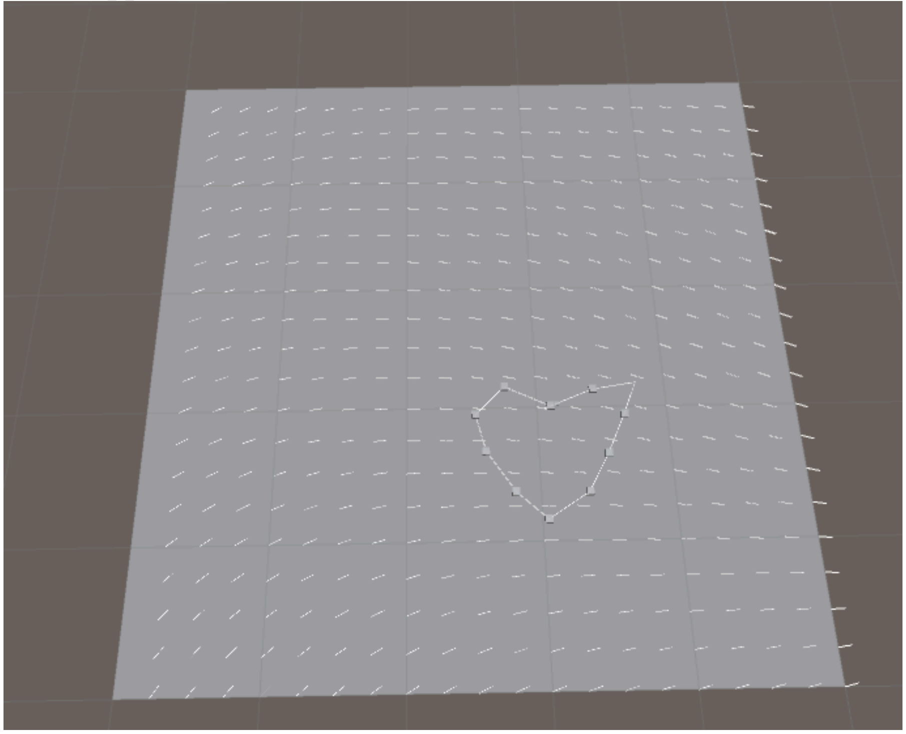
<u>Your Task:</u>

As a way to check your projects, please take snippets of your work and paste them into this document. Add a very brief description underneath each snippet.
The Snipping Tool is useful for capturing snippets of your screen.

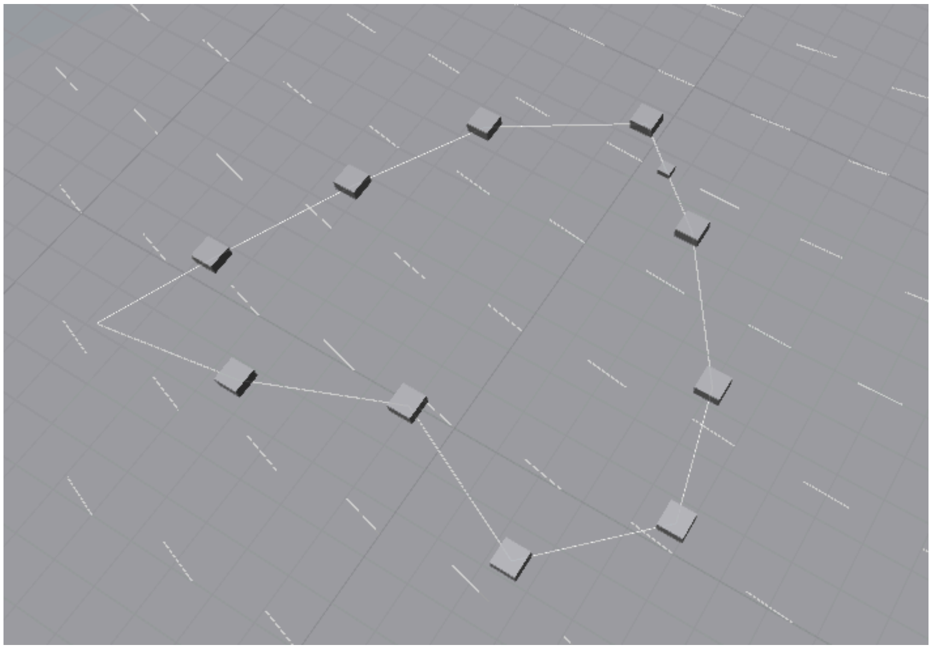
<u>Submission:</u>

Rename this document to: Lastname_FinalCheckoff.docx
Add your name to the top of this document.
Delete the instructions under “Your Task” and “Submission”
Upload this document to the dropbox by the **end of the day today (11:59 pm on Tuesday, December 5)**

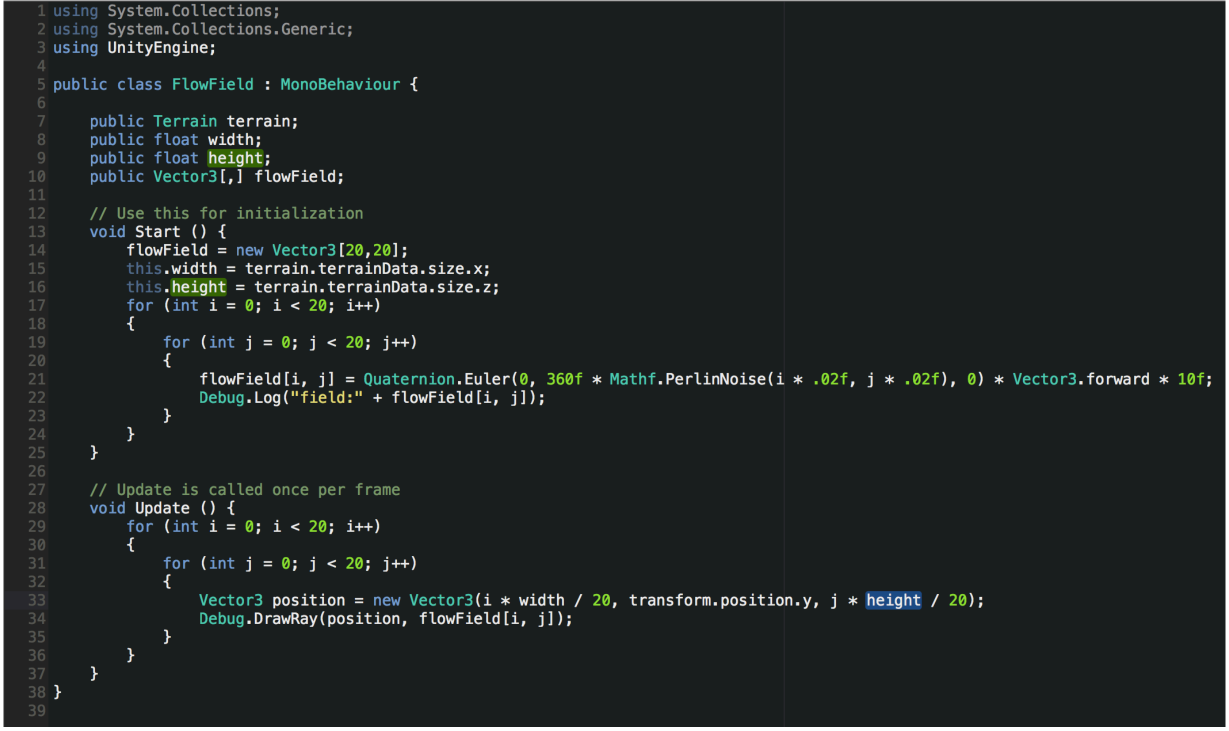
<u>Add snippets here:</u>



Flow Field and path drawing

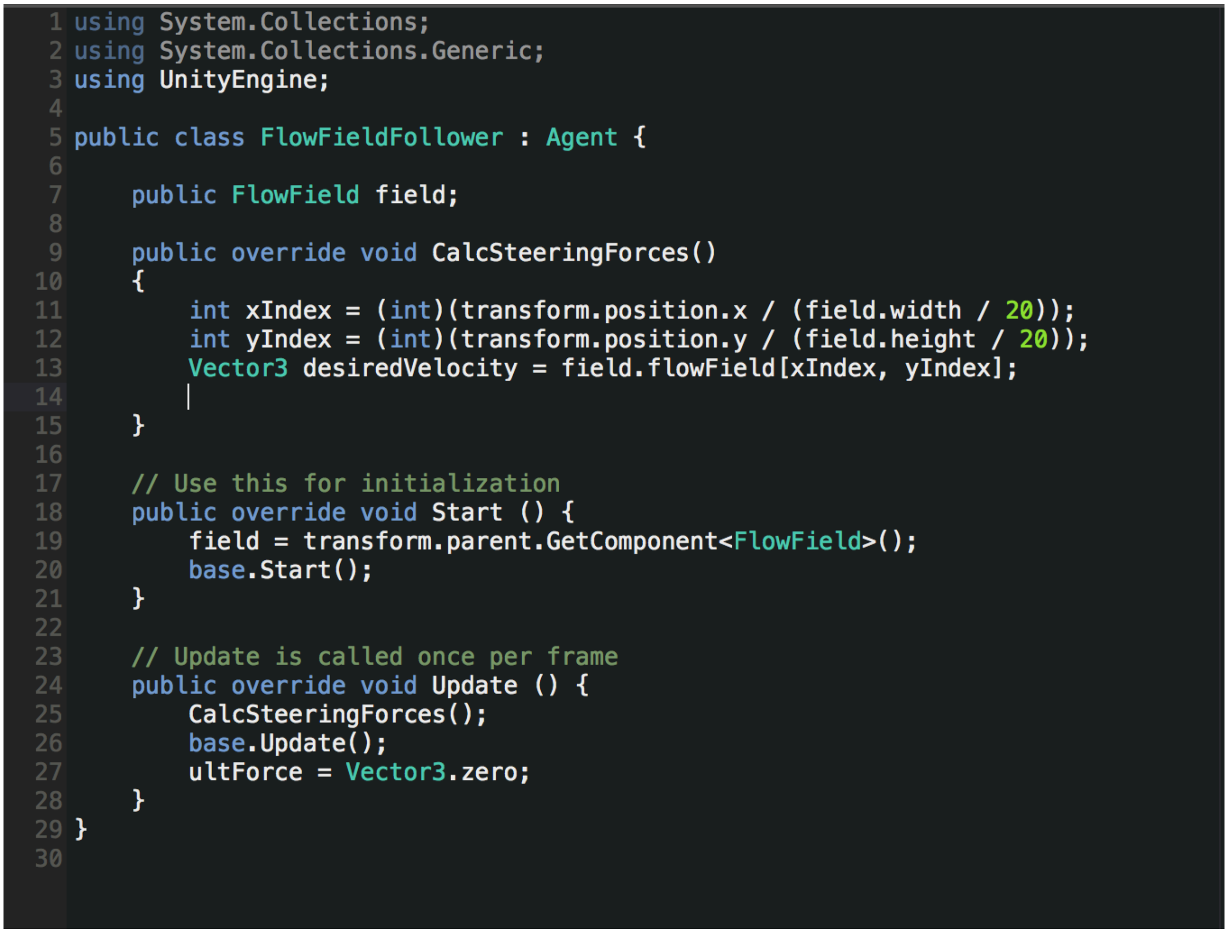


Path following path close up (tiny box is follower)



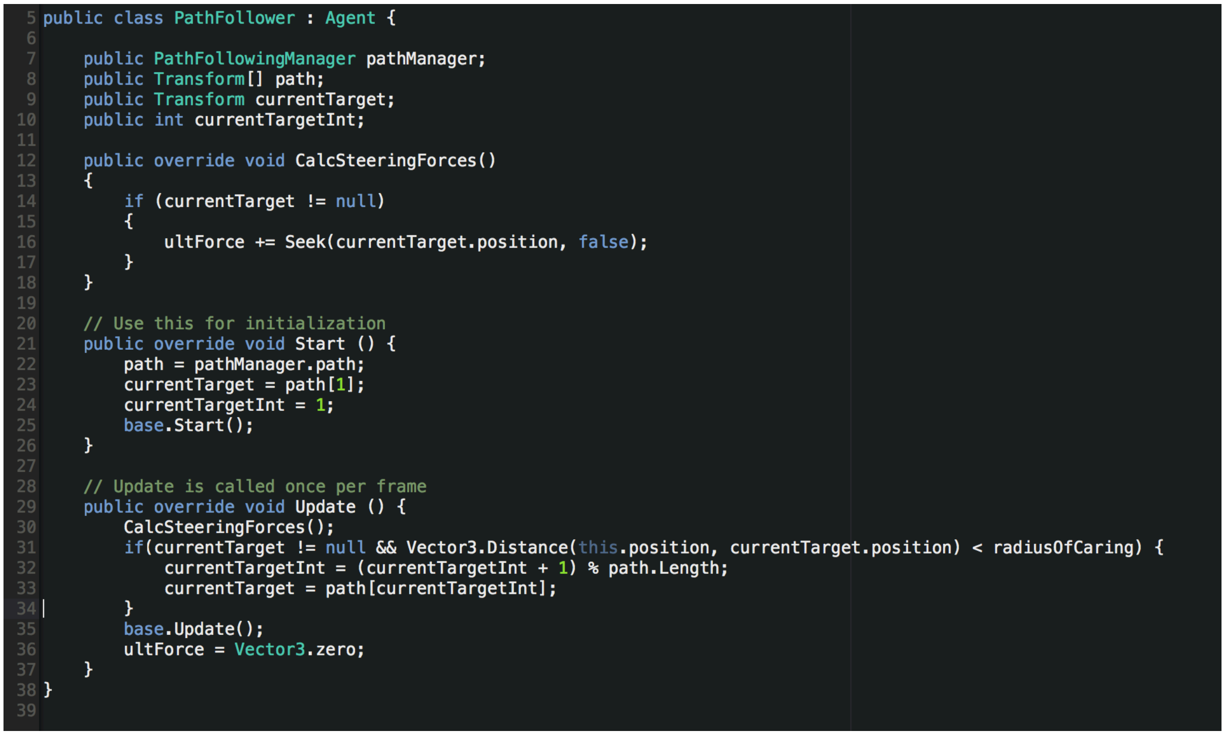
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class FlowField : MonoBehaviour {

    public Terrain terrain;
    public float width;
    public float height;
    public Vector3[,] flowField;

    // Use this for initialization
    void Start () {
        flowField = new Vector3[20,20];
        this.width = terrain.terrainData.size.x;
        this.height = terrain.terrainData.size.z;
        for (int i = 0; i < 20; i++)
        {
            for (int j = 0; j < 20; j++)
            {
                flowField[i, j] = Quaternion.Euler(0, 360f * Mathf.PerlinNoise(i * .02f, j * .02f), 0) * Vector3.forward * 10f;
                Debug.Log("field:" + flowField[i, j]);
            }
        }
    }

    // Update is called once per frame
    void Update () {
        for (int i = 0; i < 20; i++)
        {
            for (int j = 0; j < 20; j++)
            {
                Vector3 position = new Vector3(i * width / 20, transform.position.y, j * height / 20);
                Debug.DrawRay(position, flowField[i, j]);
            }
        }
    }
}

```

FlowField creation



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

public class FlowFieldFollower : Agent {

    public FlowField field;

    public override void CalcSteeringForces()
    {
        int xIndex = (int)(transform.position.x / (field.width / 20));
        int yIndex = (int)(transform.position.y / (field.height / 20));
        Vector3 desiredVelocity = field.flowField[xIndex, yIndex];
        
    }

    // Use this for initialization
    public override void Start () {
        field = transform.parent.GetComponent<FlowField>();
        base.Start();
    }

    // Update is called once per frame
    public override void Update () {
        CalcSteeringForces();
        base.Update();
        ultForce = Vector3.zero;
    }
}
```

FlowFieldFollower



```
public class PathFollower : Agent {

    public PathFollowingManager pathManager;
    public Transform[] path;
    public Transform currentTarget;
    public int currentTargetInt;

    public override void CalcSteeringForces()
    {
        if (currentTarget != null)
        {
            ultForce += Seek(currentTarget.position, false);
        }
    }

    // Use this for initialization
    public override void Start () {
        path = pathManager.path;
        currentTarget = path[1];
        currentTargetInt = 1;
        base.Start();
    }

    // Update is called once per frame
    public override void Update () {
        CalcSteeringForces();
        if(currentTarget != null && Vector3.Distance(this.position, currentTarget.position) < radiusOfCaring) {
            currentTargetInt = (currentTargetInt + 1) % path.Length;
            currentTarget = path[currentTargetInt];
        }
        base.Update();
        ultForce = Vector3.zero;
    }
}
```

Path Follower