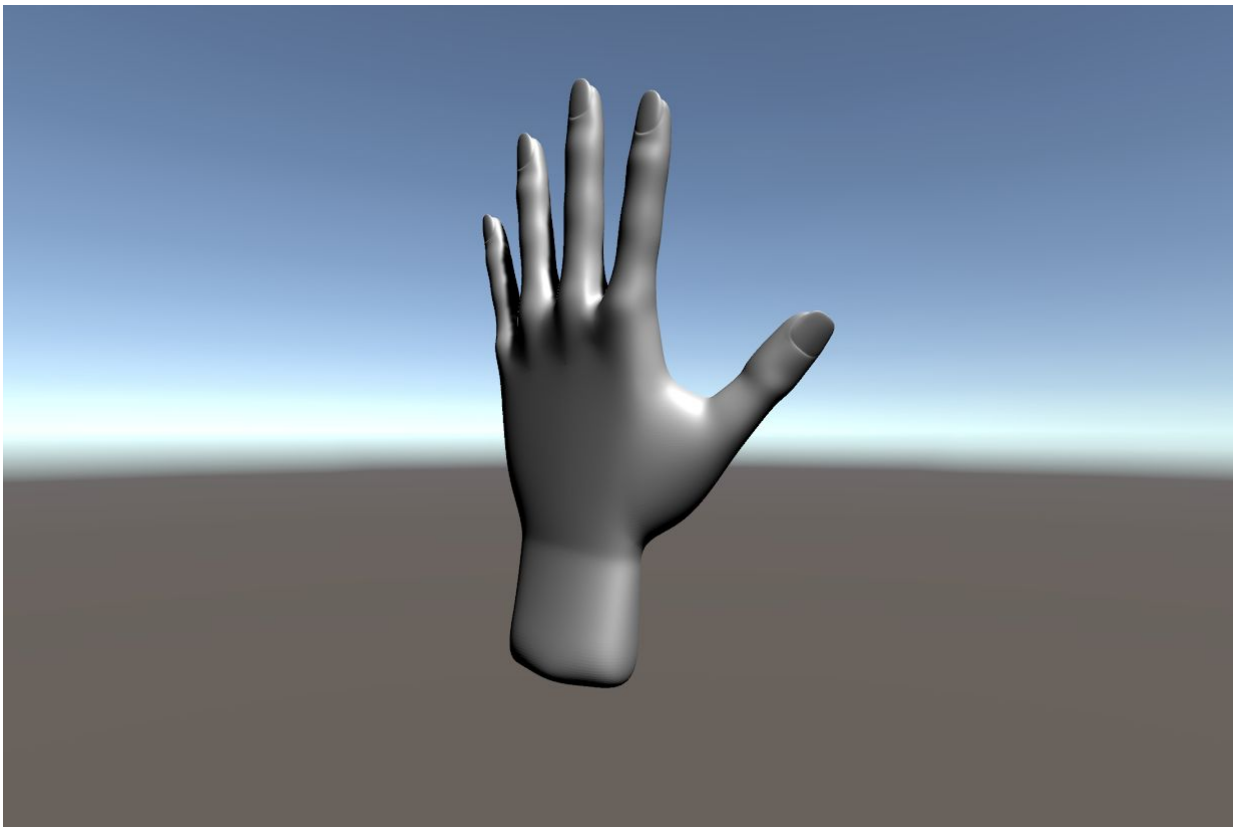
Exercise 3 - Subdivision Surfaces

In this exercise you will implement the Catmull-Clark subdivision algorithm.

The goal of this exercise is to learn about subdivision surfaces and working with mesh data structures.

You **must** submit this exercise in pairs.



EX3 Guidelines

- In this exercise you may only edit the file CatmullClark.cs, according to the instructions.
- You are given a folder of different OBJ mesh files which you can use to check your code.

General Guidelines

You may lose points for not following these guidelines.

- Make sure you are using **Unity 2019.3.3f1**
- Make sure that you understand the effect of each part of your code
- Make sure that your code does what it's supposed to do and that your results look the way they should
- Keep your code readable and clean! Avoid code duplication, comment non-trivial code and preserve coding conventions
- Keep your code efficient

Submission

Submit a single `.zip` file containing **only** the following files:

- **CatmullClark.cs**
- **`readme.txt`** that includes both partners' IDs and usernames. List the URLs of web pages that you used to complete this exercise, as well as the usernames of all students with whom you discussed this exercise

Deadline

Submit your solution via the course's moodle no later than **Wednesday, May 27 at 23:55**.

Late submission will result in 2^{N+1} points deduction where N is the number of days between the deadline and your submission. The minimum grade is 0, friday and saturday are excluded.

Part 0 / Setup

1. Download the exercise zip file from the course Moodle website and unzip it somewhere on your computer.
2. In Unity Hub, go to *Projects* and click the *Add* button on the top right. Select the folder that you have downloaded.
3. Open the project. Once Unity is open, double click the MainScene to open it.
4. Open the file CatmullClark.cs to edit it.

Note that when clicking "Subdivide" in the MeshSubdivider UI in the inspector, the function `Subdivide` of the class `CatmullClark` is called.

`QuadMeshData` documentation

In this exercise you will be working with quad-faced meshes, meaning each face contains 4 vertices rather than 3. To help you with this, you are given the class `QuadMeshData`.

`List<Vector3>` **`vertices`**

A list of mesh vertex positions

`List<Vector4>` **`quads`**

A list of quad mesh faces, where each face is defined by 4 indices of vertices in the `vertices` list

`QuadMeshData``(List<Vector3> vertices, List<Vector4> quads)`

`QuadMeshData` constructor - Initializes a `QuadMeshData` object with the given `vertices` and `quads`

`CCMeshData` documentation

In addition, you are provided with the helper class `CCMeshData` to hold the data needed for the Catmull- Clark subdivision algorithm. You will fill its properties as you implement the algorithm:

`List<Vector3>` **`points`**

Original mesh points (i.e. vertex positions)

`List<Vector4>` **`faces`**

A list of quad faces of the original mesh. Each face defined by 4 indices of vertices in the `points` list

`List<Vector4>` **`edges`**

A list of all unique edges in the mesh defined by points and faces, as explained in part 1 of this exercise

`List<Vector3>` **`facePoints`**

Face points, as described in the Catmull-Clark subdivision algorithm

`List<Vector3>` **`edgePoints`**

Edge points, as described in the Catmull-Clark subdivision algorithm

`List<Vector3>` **`newPoints`**

New locations of the original mesh points, as described in the Catmull-Clark subdivision algorithm

Part 1 / Getting Edges

In this part you will implement the function `GetEdges`. It receives a `CCMeshData` object, with its `points` and `faces` properties filled:

`List<Vector3>` **`points`** - a list of mesh vertex positions

`List<Vector4>` **`faces`** - a list of quad mesh faces, each face defined by 4 indices of vertices in the `points` list

- The function returns a `List<Vector4>` representing a list of all unique edges in the mesh defined by `points` and `faces`. Each edge is represented by a `Vector4` in the following format:

  ```
  Vector4(p1, p2, f1, f2)
  ```

  `p1`, `p2` are the 2 edge vertices, given by indices in the `point` list.

  `f1`, `f2` are the faces incident to the edge, given by indices in the `faces` list. If the edge belongs to one face only, `f2` is set to -1.

Part 2 / Catmull-Clark Subdivision

- In this part you will complete the implementation of the function `Subdivide`. The function receives 1 parameter:

  `QuadMeshData` **`meshData`** - a vertex and face representation of a 3D quad mesh

1. Implement the Catmull-Clark subdivision algorithm as learned in class. The general steps of the algorithm must be implemented in the given functions:
   - `List<Vector3> GetFacePoints(CCMeshData mesh)`
   - `List<Vector3> GetEdgePoints(CCMeshData mesh)`
   - `List<Vector3> GetNewPoints(CCMeshData mesh)`
2. After implementing these functions, use the completed `CCMeshData` object to construct and return a `QuadMeshData` representing the input mesh after one iteration of Catmull-Clark subdivision.

- You may add any function you wish to CatmullClark.cs.
- Efficiency is important - you don't have to implement the optimal solution, but you may lose points for inefficient code.
- Note that even if your code is very efficient, after a few iterations the mesh will become so big that subdividing might freeze Unity!
- Document your code and test it with different OBJ files to make sure everything works properly.

Good luck!