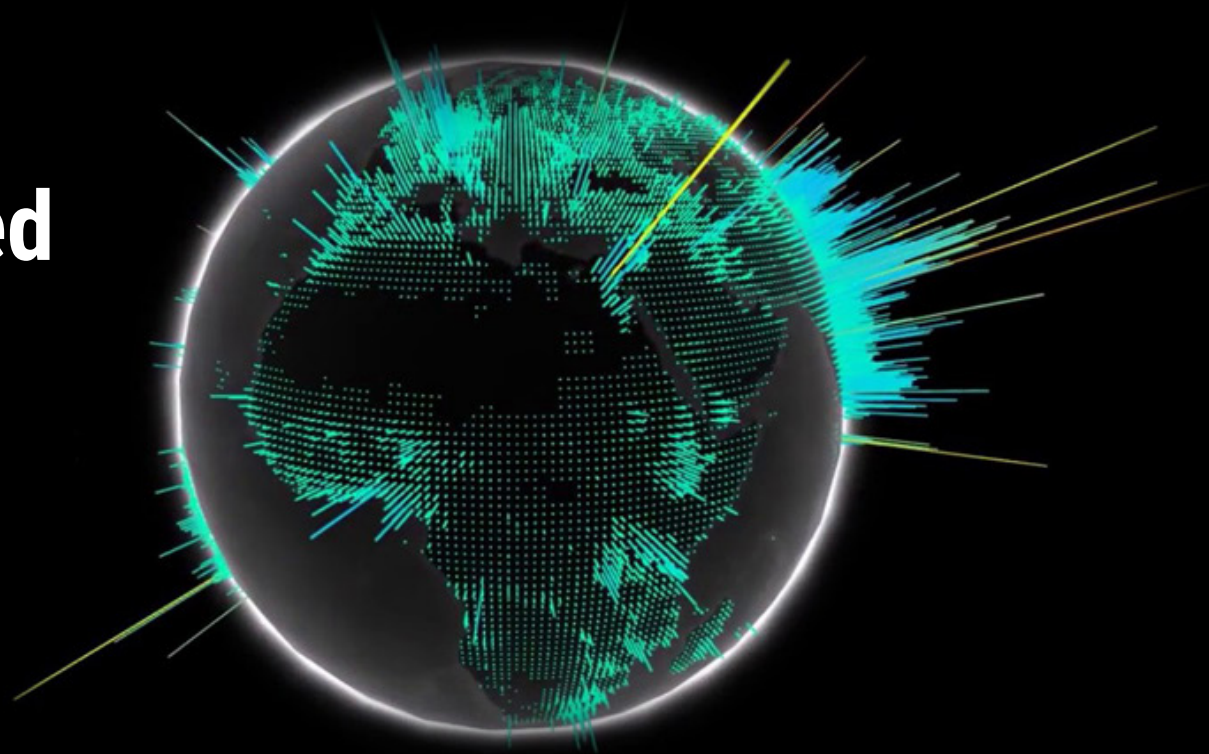


Data Generated Dashboard

Using Data from Grasshopper to
Generate UIs in Unity



<https://github.com/keijiro/GeoVfx>

SHANAIRE BLYTHE

17th March 2023

Objectives For the Session

- 1 Exporting Data From Grasshopper to CSV
- 2 Reading CSV Data in to Unity
- 3 Create a Dashboard from Data
- 4 Add Basic Interrection
- 5 Exporting Unity Model to WebGL (Convert Project Over)

Resources

<https://resources.shanaiworks.co.uk>

Link to download resources



Name: Shanaire Blythe

Qualification: MArch Architecture

Occupation: Software Developer

Interests: Concept Art, VFX, Drawing, Martial Arts and more.



Name: Shanaire Blythe

Qualification: MArch Architecture

Occupation: Software Developer

Interests: Concept Art, VFX, Drawing, Martial Arts and more.

```
ClientRpc
void RpcSpawnObject(GameObject instance)
{
    ClientScene.RegisterPrefab(instance);
    NetworkServer.Spawn(instance); // send spawn message to
clients
//    NetworkServer.SpawnWithClientAuthority(obj,
instance);
    Debug.Log("Send arccos_2");
}
```

```
public void RpccreateInstance(GameObject[] objects)
{
    if (!instance)
    {
        instance = GameObject.Instantiate(objects[type], new
Vector3(x, y, z), new Quaternion(rx, ry, rz, rw)) as GameObject;
        ClientScene.RegisterPrefab(instance);
        RpcSpawnObject(instance);
//        NetworkServer.Spawn(instance);
//        NetworkServer.Listen(7777);
        Debug.Log("Registering server callbacks");
        Debug.Log("Send arccos");
    }
}
```

```
public void clearInstance()
{
    if (instance)
    {
        GameObject.Destroy(instance);
    }
}
```

```
[XmlAttribute("CubeCollection")]
public class CubeCollection
{
    [XmlAttribute("alldata"), XmlArrayItem("storeobj")]
    public List<storeobj> alldata;

    public CubeCollection()
    {
        alldata = new List<storeobj>();
    }
}
```

```
public void Save(string path)
{
    var serializer = new XmlSerializer(typeof(CubeCollection));
    using (var stream = new FileStream(path, FileMode.Create))
    {
        serializer.Serialize(stream, this);
    }
}
```

```
public class manager : NetworkBehaviour {
```

//COLLABORATE

//Shanaire Blythe

with Adam Brennan and Kit Yeung Chan

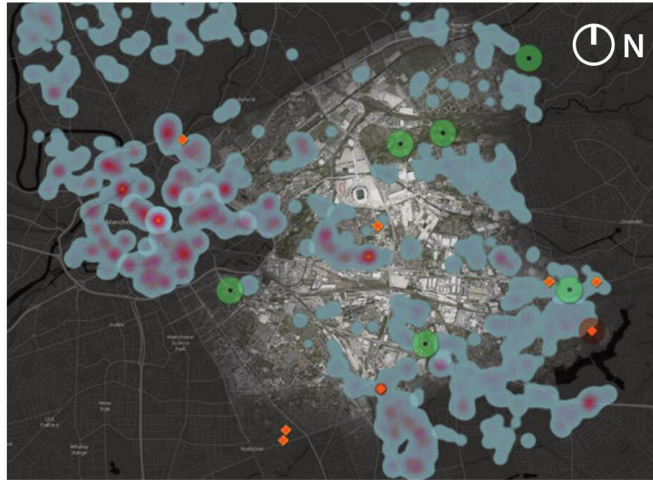
CPU - M.Arch

SITE ANALYSIS

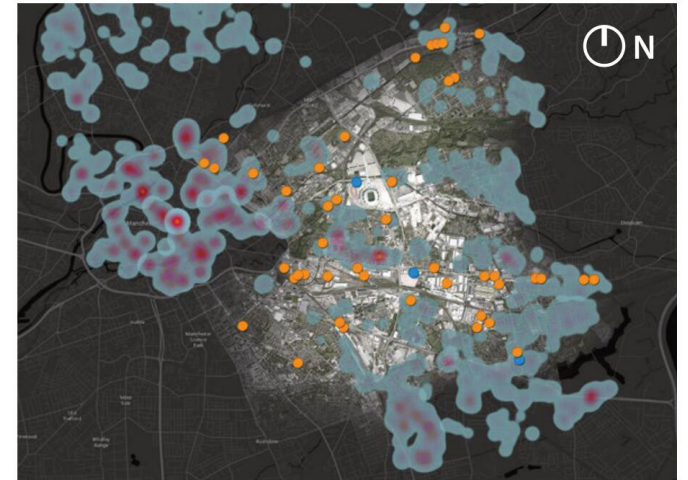
Investment and Amenities

By drawing maps consisting of residential property prices, vacant plots and amenities such as schools and parks, we aimed to understand how investment in East Manchester is affected. As the current regeneration schemes are focused on residential development we looked at house prices and how they differ based on their location in Manchester as a whole and their distribution in relation to the amenities in East Manchester.

As can be seen by the colour scale, East Manchester's properties fall into the lower end of the price spectrum. When compared with the city centre this price drop is very apparent and the further into the area houses are, the lower the prices seem to be. An expected result is that prices would be marginally higher for areas located near parks and schools.



Schools, Parks & Properties Price



Unoccupied sites & Properties Price



PANARCHY

East Manchester

Applying the panarchy to East Manchester, we can use it to represent three different scales in society which are the Macro, Meso and the Micro scales. We will relate these three scales to:

Regeneration Planners - Government level actors who are able to make policy decisions and top-down effects on East Manchester.

Developers - Any non-government actors who transform the urban morphology through land development and building.

Citizens - Residents of East Manchester who are also non-government.

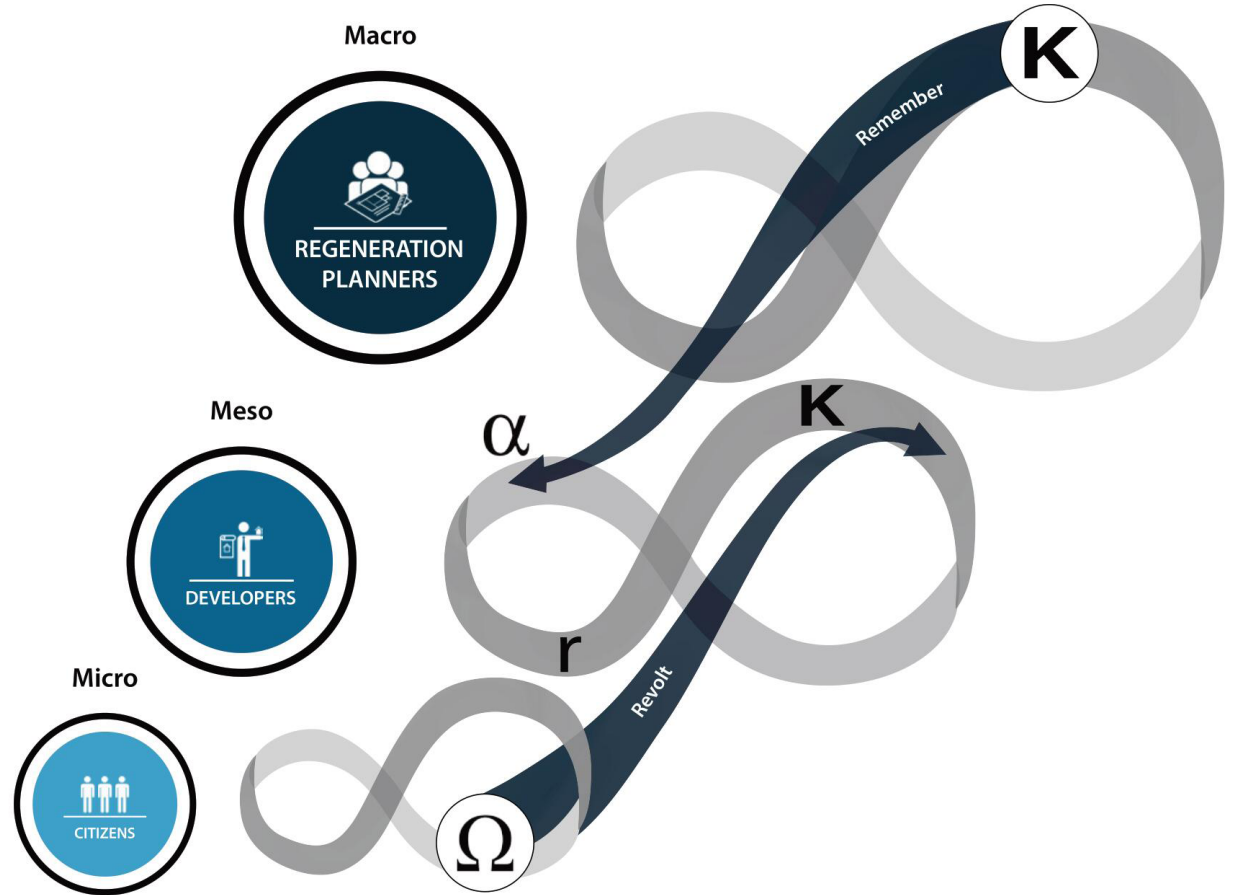
Rather than the city being looked at like the typical hierarchical top-down structure, the panarchy views these scales as different cycles where there is a level of interaction between them rather than as separate entities in total isolation from other levels in a hierarchy. The three groups are representative of organisations or groups that are around their size in society. Using this approach gives us the opportunity to explore interactions between levels, which is crucial to our approach to create better interaction between them.

By assuming a city evolves across different levels and at different speeds this sets out requirements for the proposed digital tool:

In terms of users, it implies that feedback should come from agents at different levels in order to best represent adaptation.

The model must also record data and interactions between the levels in order to test links and thus integrate or reject them.

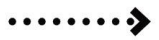
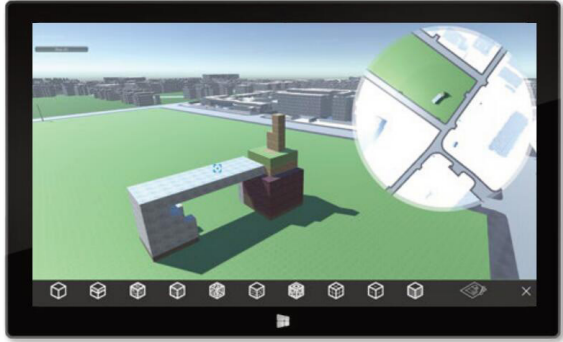
It should allow individual actions combine together to make large patterns, such as the movements of people, information, goods or money.



OUTCOME

Possibilities

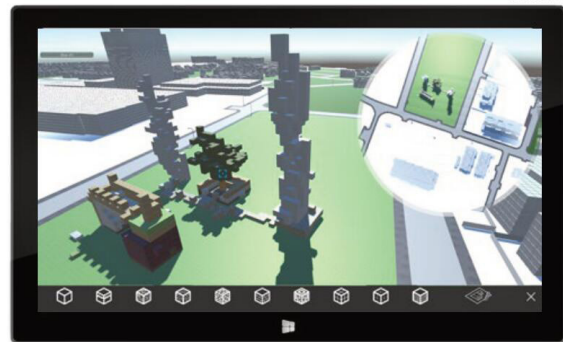
Week1



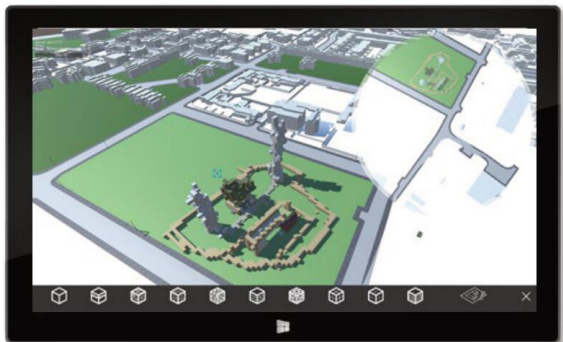
Week2



Week3



Week4



COLLABORATE

Creation





Name: Shanaire Blythe

Qualification: MArch Architecture

Occupation: Software Developer

Interests: Concept Art, VFX, Drawing, Martial Arts and more.

MF3D



Architectural Visualisation

Bonded Warehouse
18 Lower Byrom Street
Manchester
M3 4AP



MF3D



Working for 3 years +

Bonded Warehouse
18 Lower Byrom Street
Manchester
M3 4AP



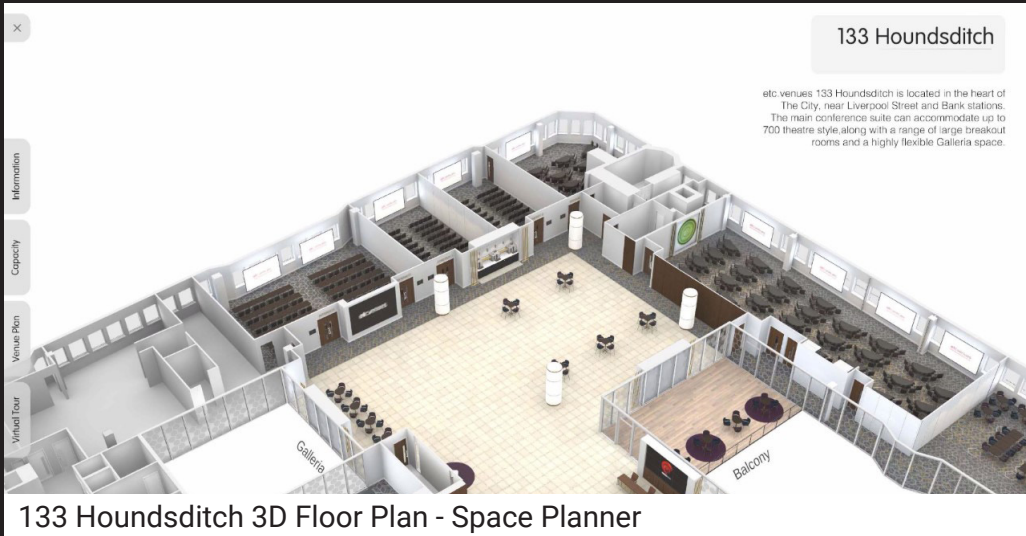
MF3D



Creating Interactive Applications

Bonded Warehouse
18 Lower Byrom Street
Manchester
M3 4AP

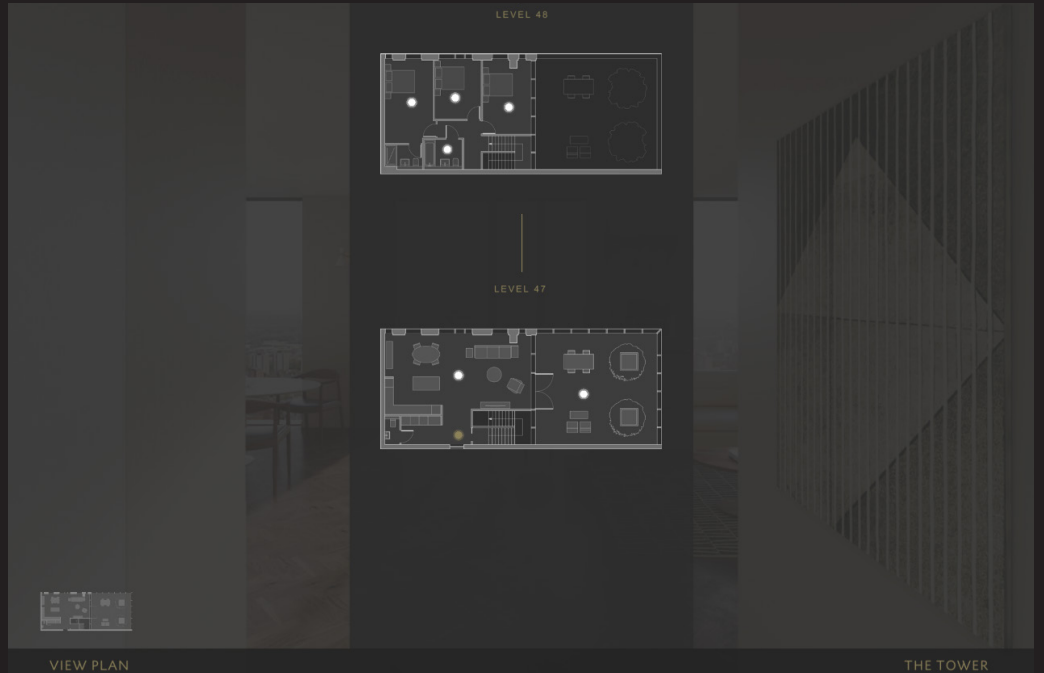




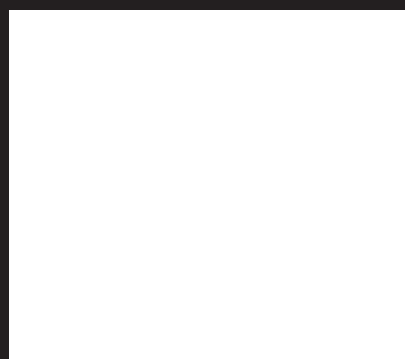
133 Houndsditch 3D Floor Plan - Space Planner



Bonded Warehouse (Android / iPad App)



One ST Johns



3D Floor Plan Static Web Application



Name: Shanaire Blythe

Qualification: MArch Architecture

Occupation: Software Developer

Interests: Concept Art, VFX, Drawing, Martial Arts and more.

Exporting Data From Grasshopper

CSV

```
1 Date,Price
2 01/01/2020,1741005
3 01/02/2020,1760434
4 01/03/2020,1730654
5 01/04/2020,1650120
6 01/05/2020,1643456
7 01/06/2020,1530076
```

CSV files are expressed by the comma that separates the column of data, with each line representing a different data record.

JSON

```
1 {
2   "widget": {
3     "visible": "on",
4     "window": {
5       "title": "Sample Konfabulator Widget",
6       "name": "main_window",
7       "width": 500,
8       "height": 500
9     },
10    "image": {
11      "src": "Images/Sun.png",
12      "name": "sun1",
13      "hOffset": 250,
14      "vOffset": 250,
15      "alignment": "center"
16    },
17    "text": {
18      "data": "Click Here",
19      "size": 36,
20      "style": "bold",
21      "name": "text1",
22      "hOffset": 250,
23      "vOffset": 100,
24      "alignment": "center",
25      "onMouseUp": "sun1.opacity = (sun1.opacity / 100) * 90;"
26    }
27  }
28 }
```

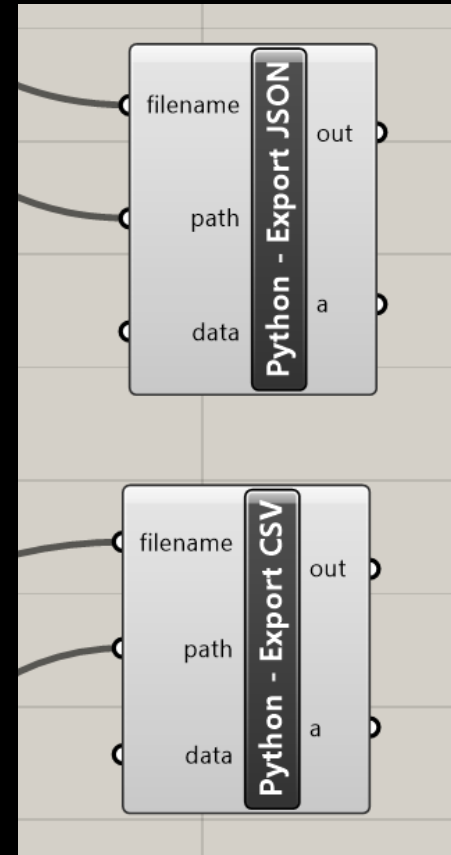
JSON is a text format that is used to represent structured data, which is based on how JavaScript Notation, using key value pairs.

CSV & JSON

To save the data from Grasshopper we will be using the python node. This will be used to format and save the data as either CSV or JSON.

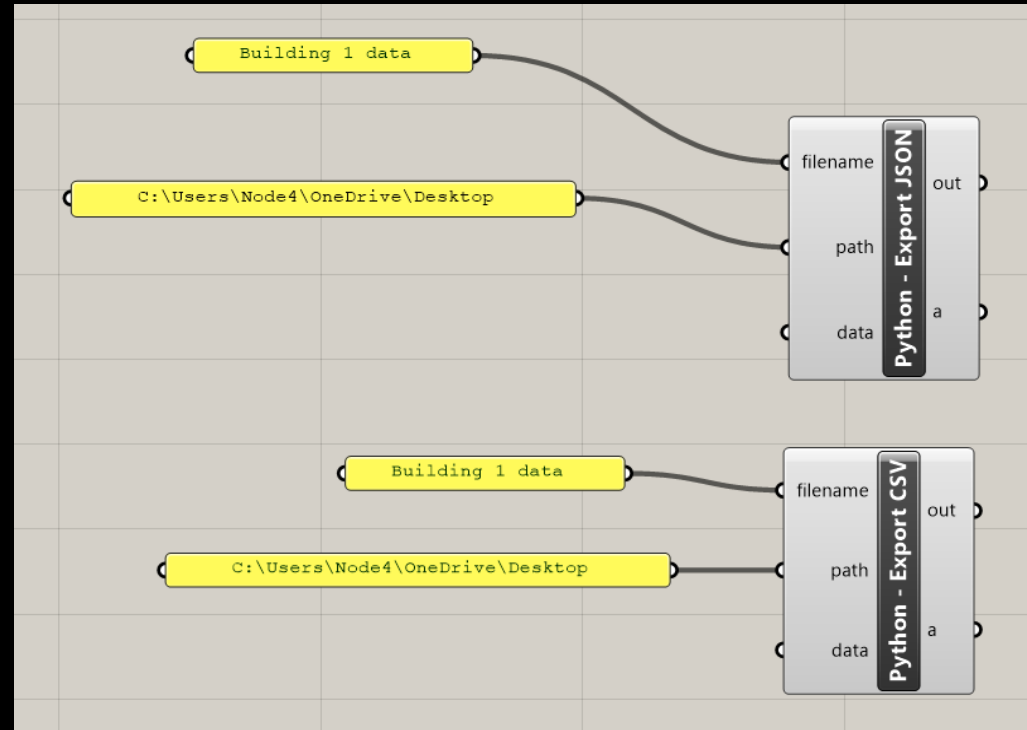
Instructions:

- + Create Python Node
- + Add the following input variables
 - filename = Name of the file (panel)
 - path = Path Save folder (panel)
 - data = data to be saved (gh data)



CSV & JSON

For this example, we will be using the desktop to store the data.



CSV

Creating a CSV file will be much simpler than JSON, for this all we will need to import is the “os” library, which gives us access to the ability to controlling different file path on the system

Instructions:

- + Import os library
- + Create a main function to control execution order
- + Create a csvData variable, which specifies the first row of the csv file

```
10
11 import rhinoscriptsyntax as rs
12 import os
13
14 csvData = "index, height"
15
16 def main():
17     pass
18
19 main()
```

JSON

A JSON file follows a similar structure to a dictionary in python. So we will be working with the data as a dictionary, then save it to JSON afterwards.

Instructions:

- + Import os library
- + Import json library
- + Create a main function to control execution order

```
11 import rhinoscriptsyntax as rs
12 import os
13 import json
14
15 """
16 {
17     ...."name": "_name",
18     ...."height":[
19     ....1,2,4
20     ....]
21 }
22 """
23 |
24 def main():
25     ....
26     ....pass
27
28 main()
29
```

Demo

CSV

Example code for CSV export

```
11 import rhinoscriptsyntax as rs
12 import os
13 import math
14
15 def SaveCSV(p_dir, p_folder, p_filename, p_csvData):
16     if (os.path.exists("{0}/{1}".format(p_dir, p_folder))):
17         with open("{0}/{1}/{2}.csv".format(p_dir, p_folder, p_filename), "w") as file:
18             file.write(p_csvData)
19     else:
20         os.mkdir("{0}/{1}".format(p_dir, p_folder))
21         SaveCSV(p_dir, p_folder, p_filename, p_csvData)
22
23 def StructureCSV(p_data, p_csvData):
24     for index, item in enumerate(p_data):
25         p_csvData += "\n{0},{1},{2}".format(index, item, math.floor(item))
26     return p_csvData
27
28 def main():
29     csvData = "index, height, int height"
30     folder = "ghData"
31     path = os.path.join(currDir, folder)
32     os.chdir(path)
33     currDir = os.getcwd()
34     csvData = StructureCSV(data, csvData)
35     SaveCSV(currDir, folder, filename, csvData)
36
37 main()
```

JSON

Example code for JSON export

```
10
11 import rhinoscriptsyntax as rs
12 import os
13 import json
14
15 """
16 {
17     ...."name": "_name",
18     ...."height":[
19     ....1,2,4...
20     ....]
21 }
22 """
23
24 def SaveJSON(p_dir, p_folder, p_filename, p_json):
25     ....if (os.path.exists("{0}/{1}/".format(p_dir, p_folder))):
26     ....with open("{0}/{1}/{2}.json".format(p_dir, p_folder, p_filename), "w") as file:
27     ....file.write(json.dumps(p_json))
28     ....else:
29     ....os.mkdir("{0}/{1}/".format(p_dir, p_folder))
30     ....SaveJSON(p_dir, p_folder, p_filename, p_json)
31
32 def StructureJSON(p_Data):
33     ....jsData = json.loads("{}")
34     ....jsData["name"] = "building data"
35     ....jsData["height"] = p_Data
36     ....return jsData
37
38 def main():
39     ....folder = "ghData"
40     ....
41     ....os.chdir(path)
42     ....currDir = os.getcwd()
43     ....
44     ....json = StructureJSON(data)
45     ....SaveJSON(currDir, folder, filename, json)
46
47     main()
48
```

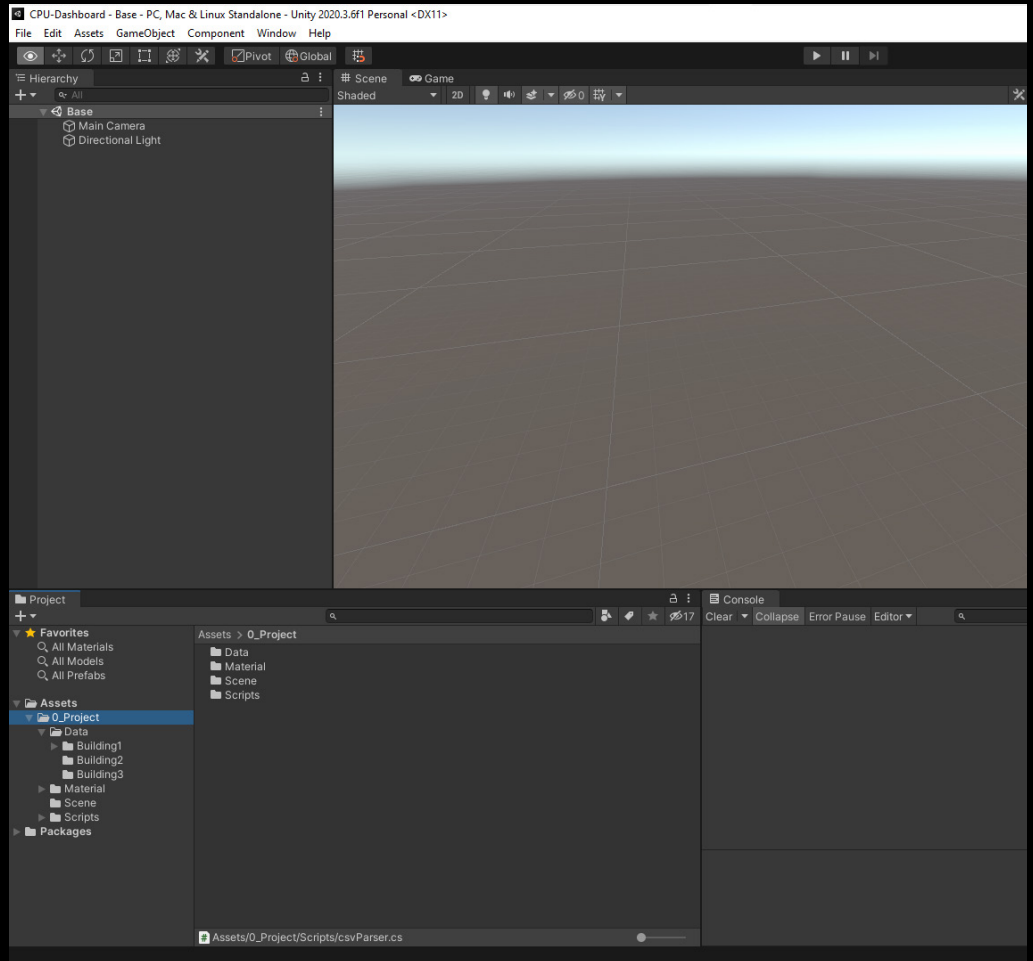
Read Exported Data Into Unity

Create a CSV or JSON Parser

To load the exported data or any data that you have on file, we will need to write a parser so that unity can understand the structure and fields.

Instructions:

- + Open Unity
- + Create a blank Scene & Load it
- + Create a Simple File Structure
 - 0_Project
 - Data
 - Material
 - Scene
 - Script

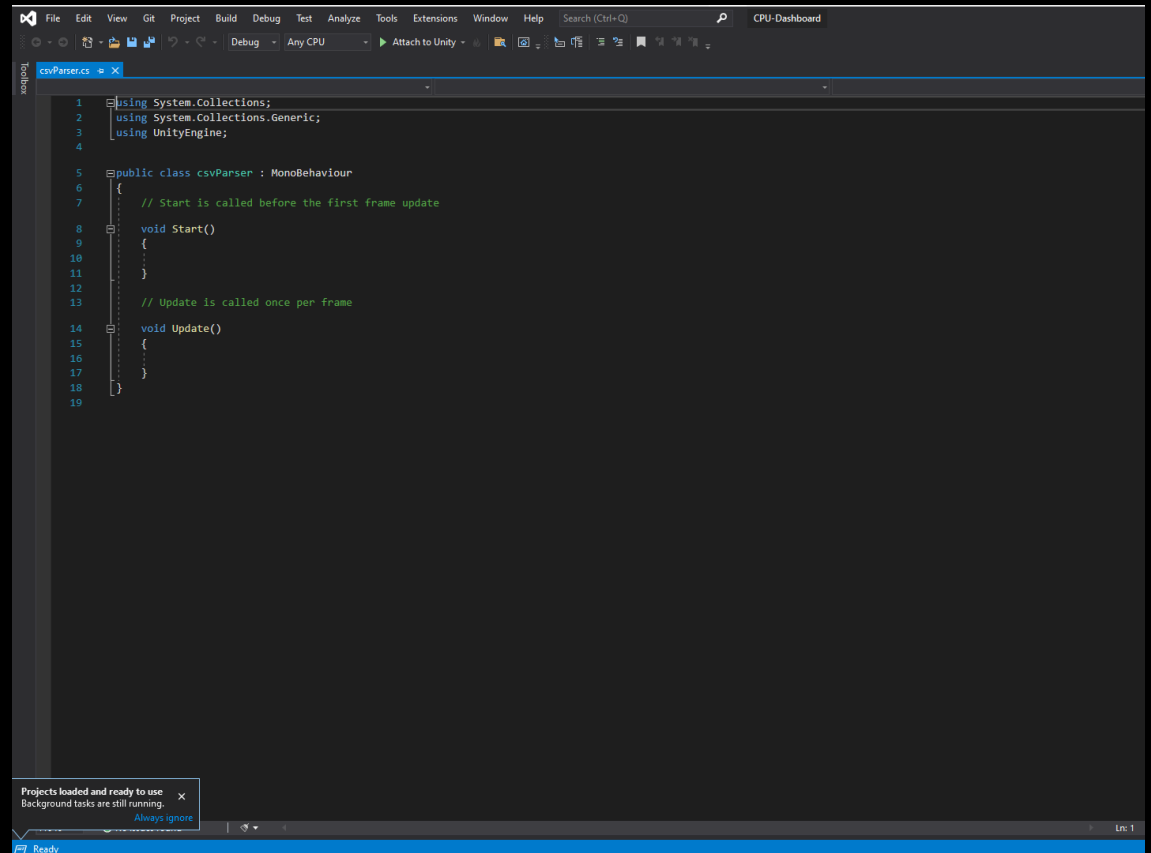


Create a CSV or JSON Parser

Now you can add your data files into the project inside of the data folder.

Inside the **scripts folder** create a file called **csvParser** and open the file in your text editor of choice.

One thing to remember is that we want the data to only load at the **Start of the Application**

A screenshot of the Visual Studio IDE showing a C# script named 'csvParser.cs'. The code is as follows:

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class csvParser : MonoBehaviour
6 {
7     // Start is called before the first frame update
8     void Start()
9     {
10    }
11 }
12
13 // Update is called once per frame
14 void Update()
15 {
16 }
17 }
18
19
```

The IDE interface includes a menu bar (File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help), a toolbar, and a status bar at the bottom showing 'Ready'. A small notification box at the bottom left says 'Projects loaded and ready to use. Background tasks are still running. Always ignore'.

Demo

Create a CSV Parser

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

[ExecuteInEditMode]
@UnityScript 4 references
public class csvParser : MonoBehaviour
{
    private static csvParser Instance = null;

    1 reference
    public static csvParser GS_Instance
    {
        get
        {
            if (Instance == null)
            {
                Instance = FindObjectOfType<csvParser>();
            }
            return Instance;
        }
    }

    1 reference
    public BuildingDataParser G_GetBuildingParserData
    {
        get
        {
            return buildingDataParser;
        }
    }

    [SerializeField] TextAsset csvFile;
    [SerializeField] BuildingDataParser buildingDataParser;
    [SerializeField] bool isRunOperation;

    @UnityMessage 0 references
    void Start()
    {
        if (buildingDataParser == null)
        {
            LoadCSVFile(csvFile.text);
        }
    }

    @UnityMessage 0 references
    private void Update()
    {
        if (isRunOperation)
    }
```

```
41
    @UnityMessage 0 references
    private void Update()
    {
        44
        {
            45
            if (isRunOperation)
            {
                46
                isRunOperation = false;
                47
                LoadCSVFile(csvFile.text);
            }
        }
    }

    2 references
    private void LoadCSVFile(string p_csvText)
    {
        52
        var csvRows = p_csvText.Split('\n');
        53
        var csvRowHeaders = csvRows[0].Split(',');
        54
        buildingDataParser = new BuildingDataParser(csvRowHeaders[0], csvRowHeaders[1], csvRowHeaders[2]);
        55
        56
        for (int item = 1; item < csvRows.Length; item++)
        {
            58
            var csvRowItems = csvRows[item].Split(',');
            59
            var Data = new BuildingDataParser_BuildingDataParser_Data();
            60
            61
            Data.index = int.Parse(csvRowItems[0]);
            62
            Data.Height = float.Parse(csvRowItems[1]);
            63
            Data.intHeight = float.Parse(csvRowItems[2]);
            64
            buildingDataParser.parser_Datas.Add(Data);
        }
    }

    [System.Serializable]
    5 references
    public class BuildingDataParser
    {
        [System.Serializable]
        2 references
        public class BuildingDataParser_Header
        {
            76
            public string indexName;
            77
            public string HeightName;
            78
            public string intHeightName;
        }

        [System.Serializable]
        3 references
        public class BuildingDataParser_Data
        {
            82
            83
            84
            85
        }
    }
```

```
61
    var Data = new BuildingDataParser_BuildingDataParser_Data();
    62
    Data.index = int.Parse(csvRowItems[0]);
    63
    Data.Height = float.Parse(csvRowItems[1]);
    64
    Data.intHeight = float.Parse(csvRowItems[2]);
    65
    buildingDataParser.parser_Datas.Add(Data);
    66
    67
    68
    69
    70
    71
    [System.Serializable]
    3 references
    public class BuildingDataParser
    {
        [System.Serializable]
        2 references
        public class BuildingDataParser_Header
        {
            76
            public string indexName;
            77
            public string HeightName;
            78
            public string intHeightName;
        }

        [System.Serializable]
        3 references
        public class BuildingDataParser_Data
        {
            82
            83
            84
            85
            86
            public int index;
            87
            public float height;
            88
            public float intHeight;
        }

        public BuildingDataParser_Header parser_Header;
        91
        public List<BuildingDataParser_Data> parser_Datas = new List<BuildingDataParser_Data>();
        92
        93
        1 reference
        public BuildingDataParser(string p_indexName, string p_HeightName, string p_intHeight)
        {
            94
            parser_Header = new BuildingDataParser_Header();
            95
            parser_Header.indexName = p_indexName;
            96
            parser_Header.HeightName = p_HeightName;
            97
            parser_Header.intHeightName = p_intHeight;
            98
            99
            100
            101
            102
        }
    }
```

Sample Code for CSV Parser

Create a JSON Parser

```
File Edit View Git Project Build Debug Test Analyze Tools Extensions Window
Assembly-CSharp jsonParser
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 [ExecuteInEditMode]
6 @ Unity Script (1 asset reference) | 4 references
7 public class jsonParser : MonoBehaviour
8 {
9     private static jsonParser Instance = null;
10     public static jsonParser GS_Instance
11     {
12         get
13         {
14             if (Instance == null)
15             {
16                 Instance = FindObjectOfType<jsonParser>();
17             }
18             return Instance;
19         }
20     }
21     1 reference
22     public BuildingDataJSONParser_G_BuildingData
23     {
24         get { return buildingData; }
25     }
26     [SerializeField] bool isRunOperation;
27     [SerializeField] TextAsset jsonData;
28     [SerializeField] BuildingDataJSONParser buildingData;
29
30     // Start is called before the first frame update
31     @ Unity Message | 0 references
32     void Start()
33     {
34         // json
35     }
36
37     // Update is called once per frame
38     @ Unity Message | 0 references
39     void Update()
40     {
41         if (isRunOperation)
42         {
43             isRunOperation = false;
44             LoadJSONParser(jsonData.text);
45         }
46     }
47 }
110% No issues found
```

```
File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q)
Assembly-CSharp jsonParser
27     }
28     }
29     }
30     }
31     }
32     }
33     }
34     }
35     }
36     }
37     }
38     }
39     }
40     }
41     }
42     }
43     }
44     }
45     }
46     }
47     }
48     }
49     }
50     }
51     }
52     }
53     }
54     }
55     }
56     }
57     }
110% No issues found
```

Sample Code for JSON Parser

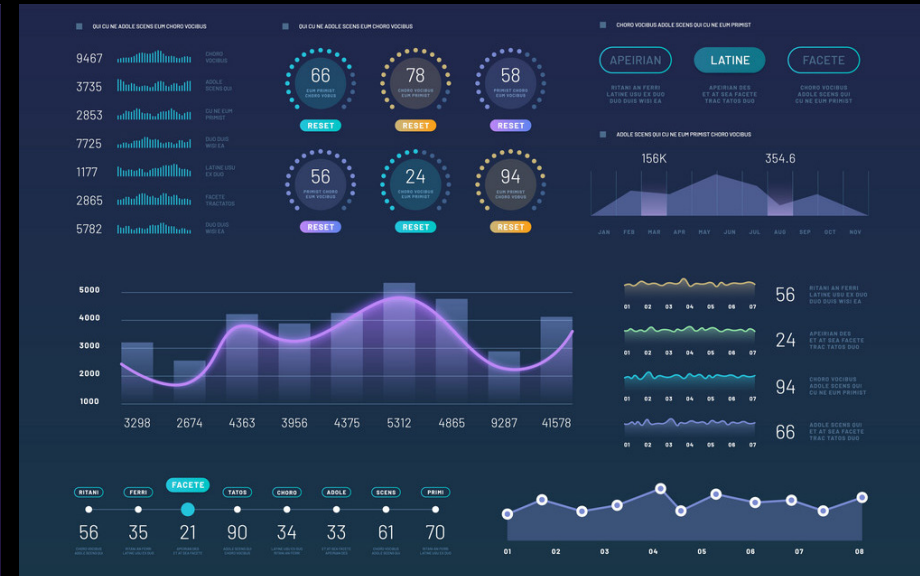
Create a Dashboard from Data

Creating a Visual Dashboard

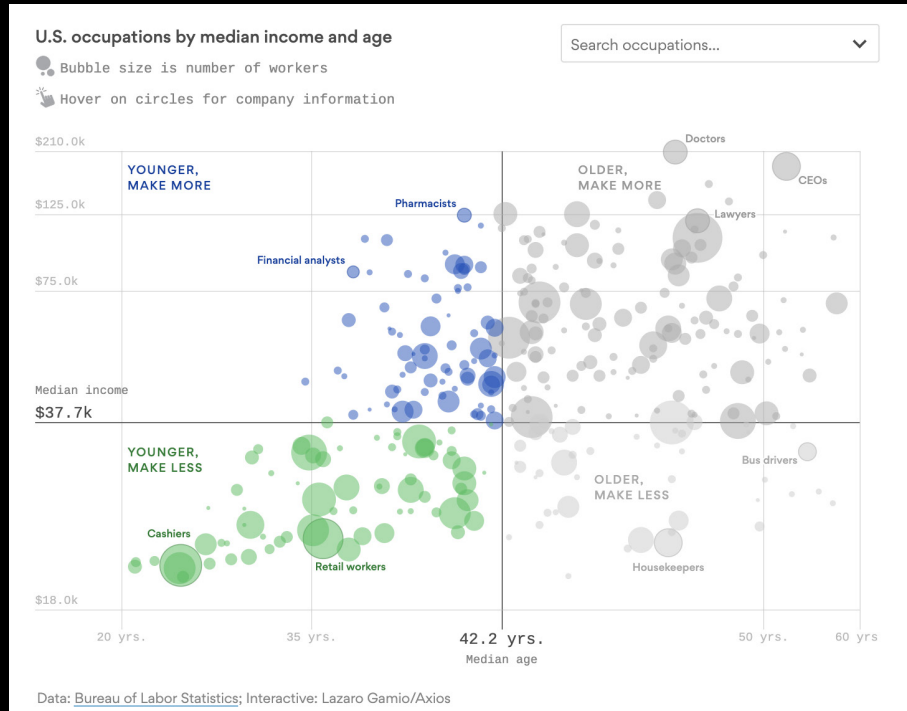
To achieve a visual dashboard you need to understand the data you have and the different possible ways to visualise it for better comprehension through a dashboard.

This would require looking at examples to see how similar data is represented, when look at the tools available to us to do so.

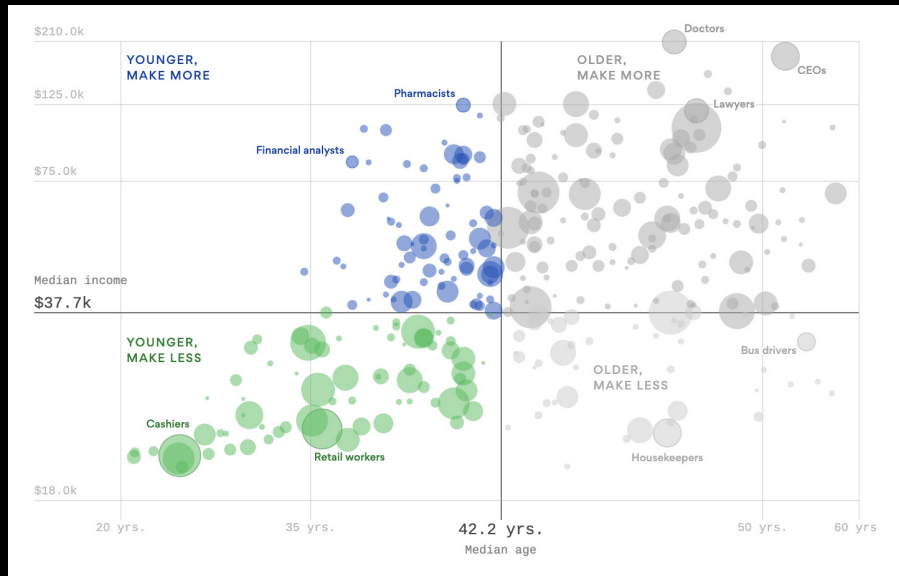
Creating a Visual Dashboard



Creating a Visual Dashboard



Creating a Visual Dashboard

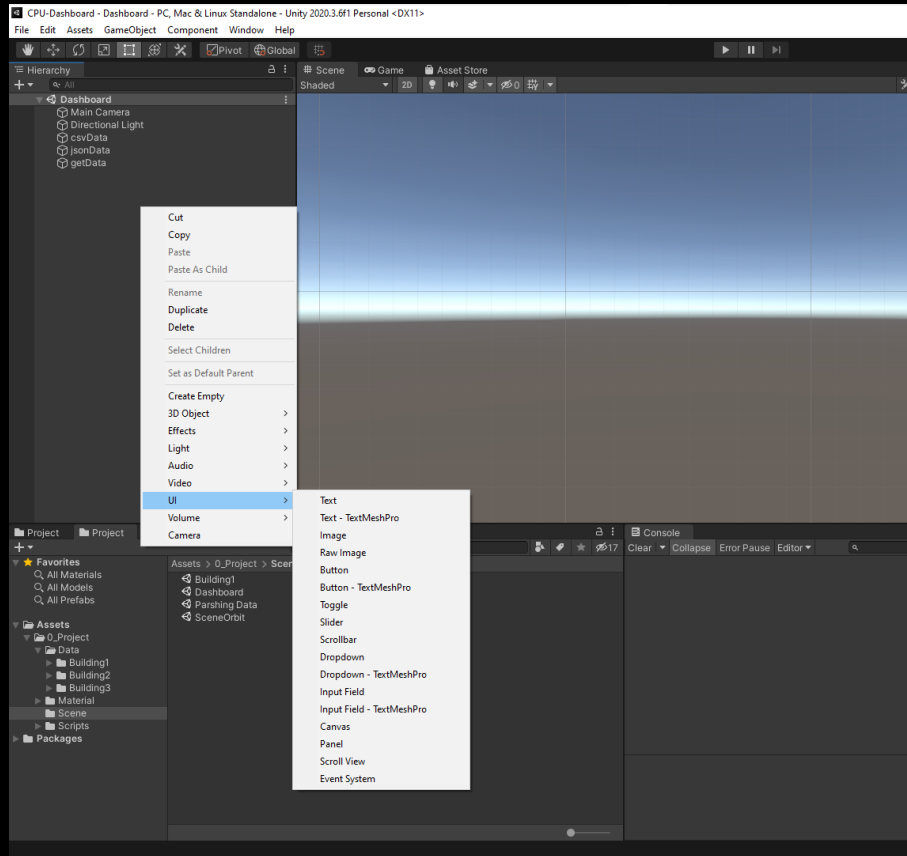


Chosen Dashboard to Create

Creating a Visual Dashboard

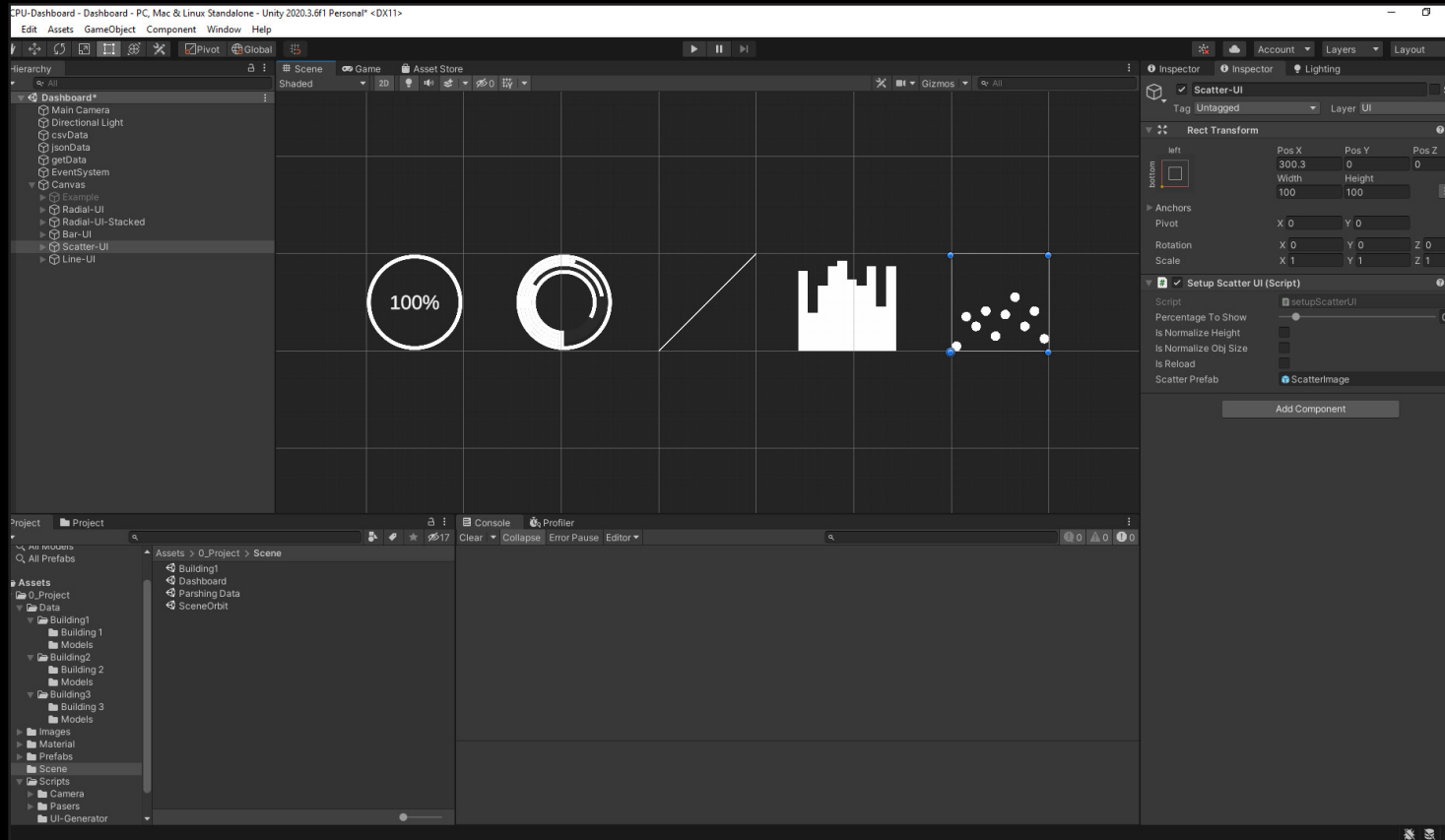
We will need to utilise the UI system inside of Unity which can be found in the Right click context menu over the Hierarchy and under **UI**. A canvas is required as a container for the UI elements

You can use many third party plugins to offset much of the setup but some of these come with a cost associated with them, which you can find on the asset store.



Demo

Creating a Visual Dashboard



Mock UI

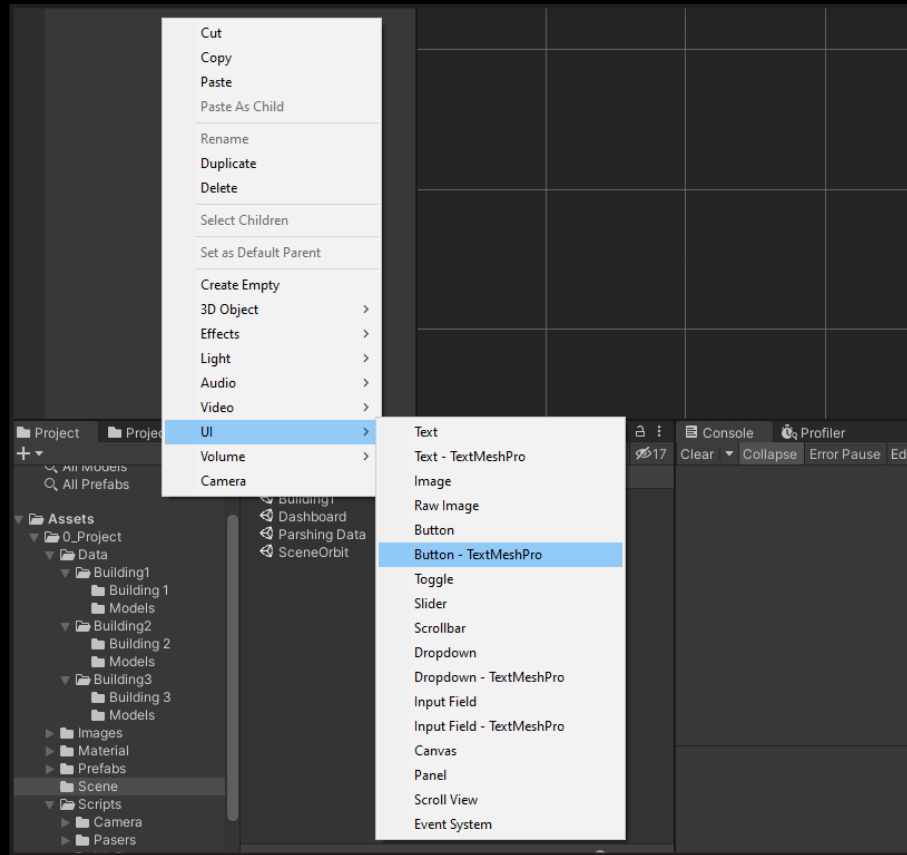
Add Basic Interaction

Basic Interaction

Unity offers a variety of tool to add interaction to our scene. This includes buttons, sliders, toggles and input fields. You can mix an add different things together to create custom interfaces.

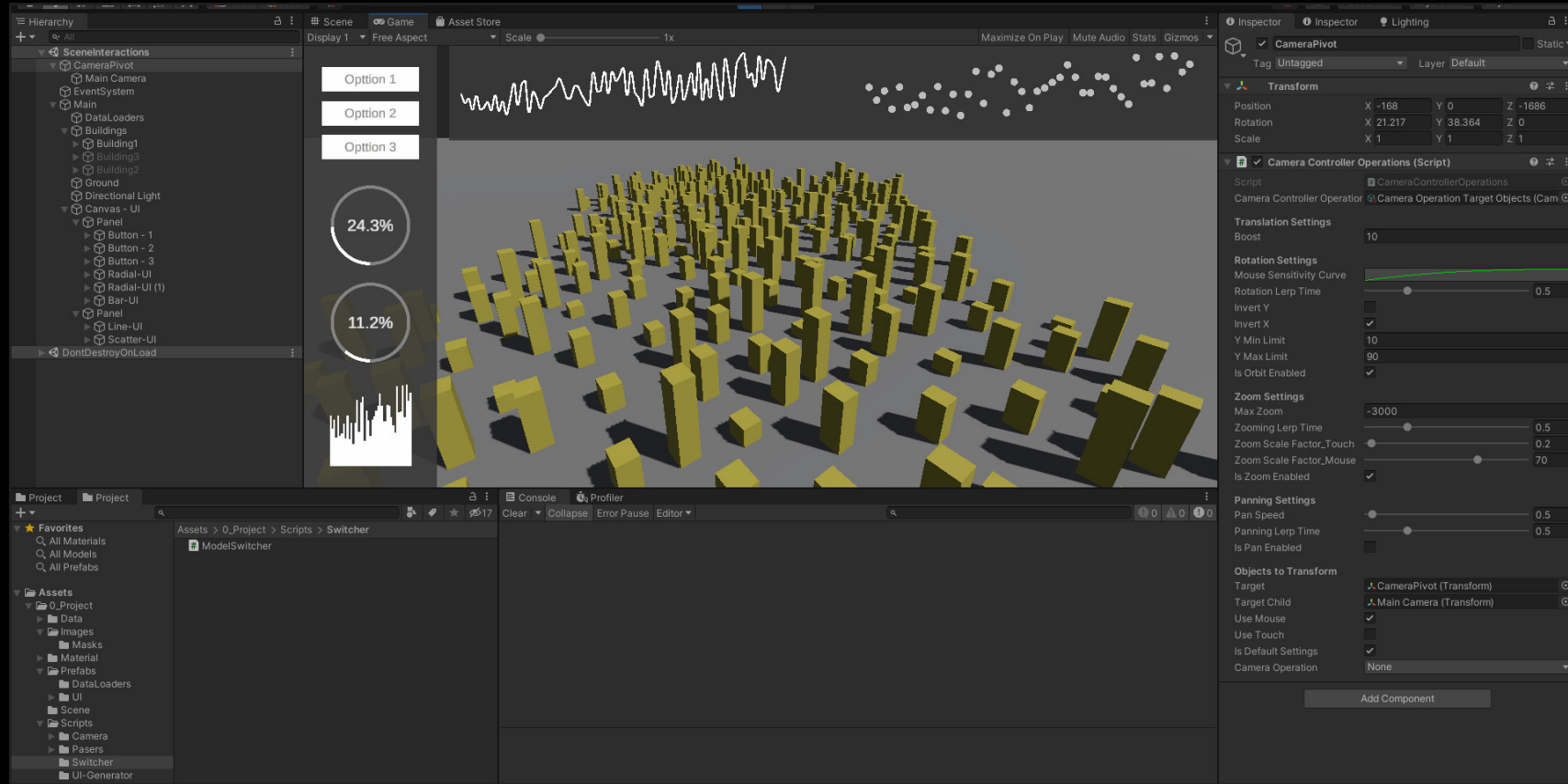
An Orbit Script will also be provided.

We will go through setting up the initial scene.



Demo

Add Basic Interaction

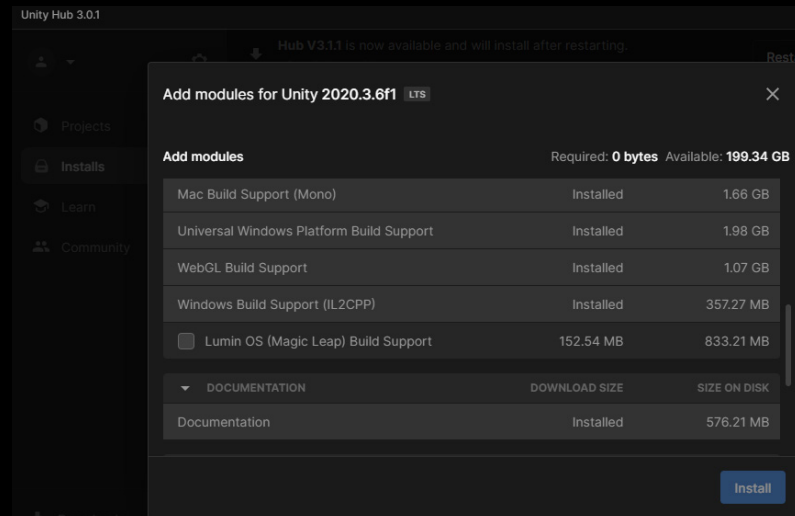
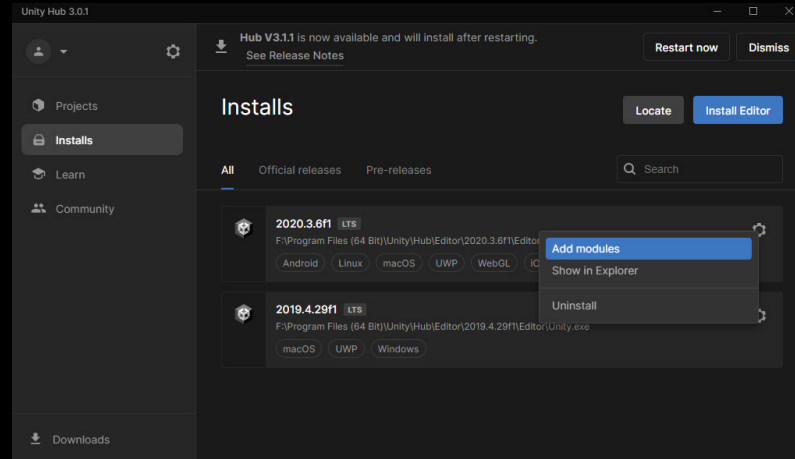


Export to WebGL

Export To WebGL

WebGL is a module that allows you to be able to export a web version of your project, which you can then later host.

It does need to be installed from the Add module section in side of Unity Hub. This does need to relate to the version of Unity you are used. Each different version will require its own installation.



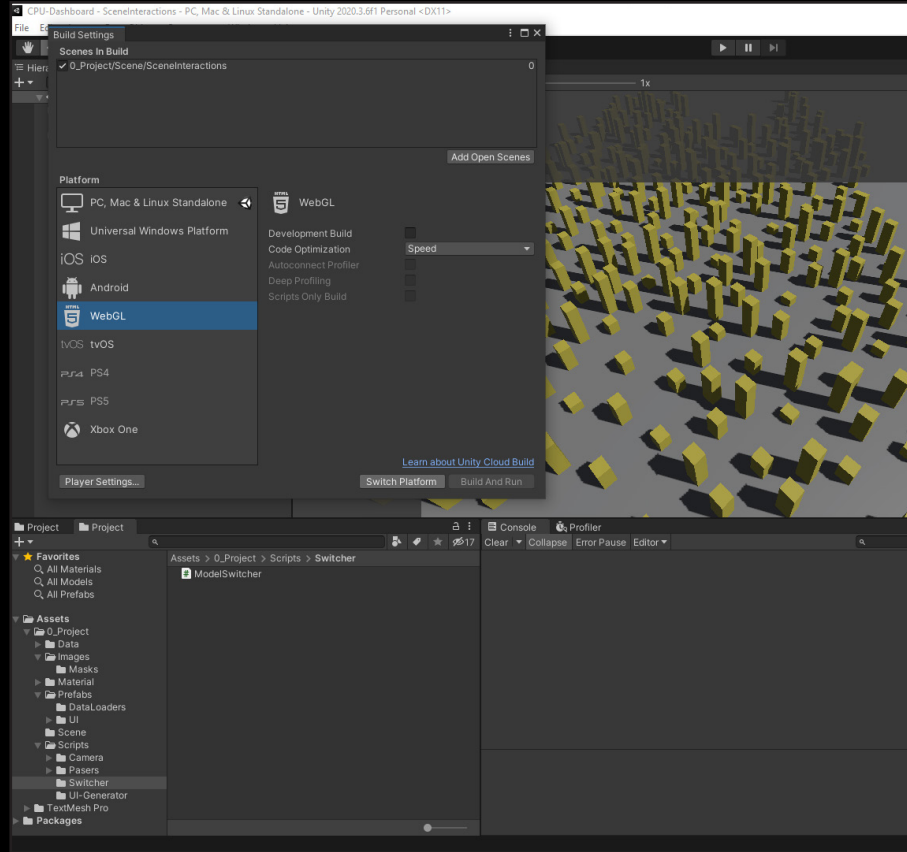
Exported Project Files

To Export the Project

Instruction:

- + Go to File > Build Settings
- + Add the Scene to the Scene in build Section
- + If You haven't switched to WebGL, click WebGL under platform and click Switch Platform.
- + Now Click Build to Export.
- + If all goes well you will see build succeeded when finished.

*If you have issues in running the export try changing Compression Format to disabled (File > Build Settings > Player Settings > Publishing Settings > Compression Format)

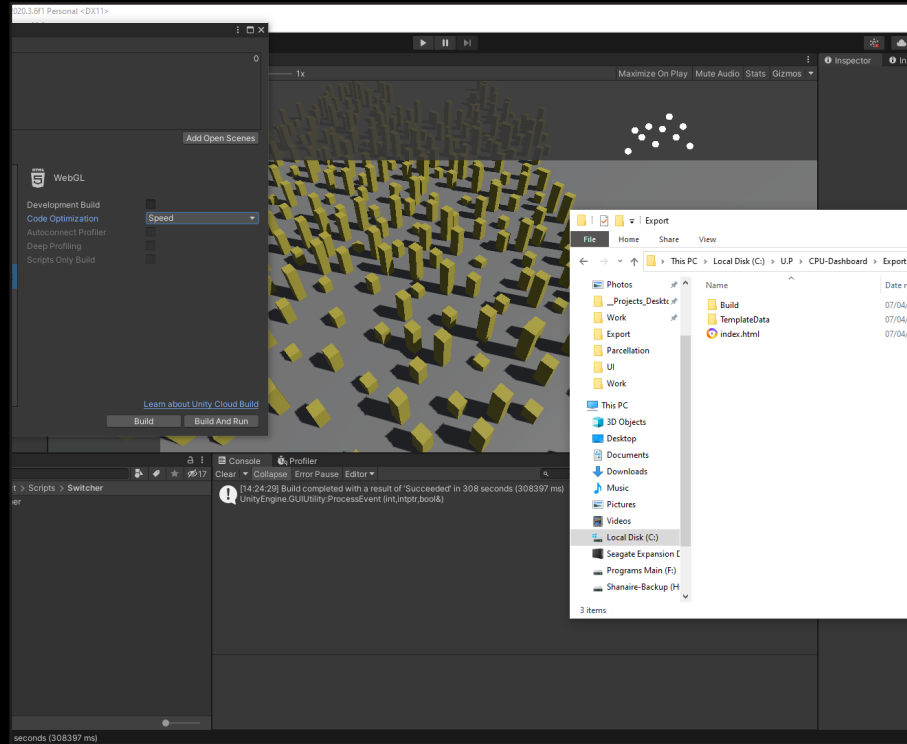


Export To WebGL

With the project now exported you should have files similar to that on the screen shot on the right.

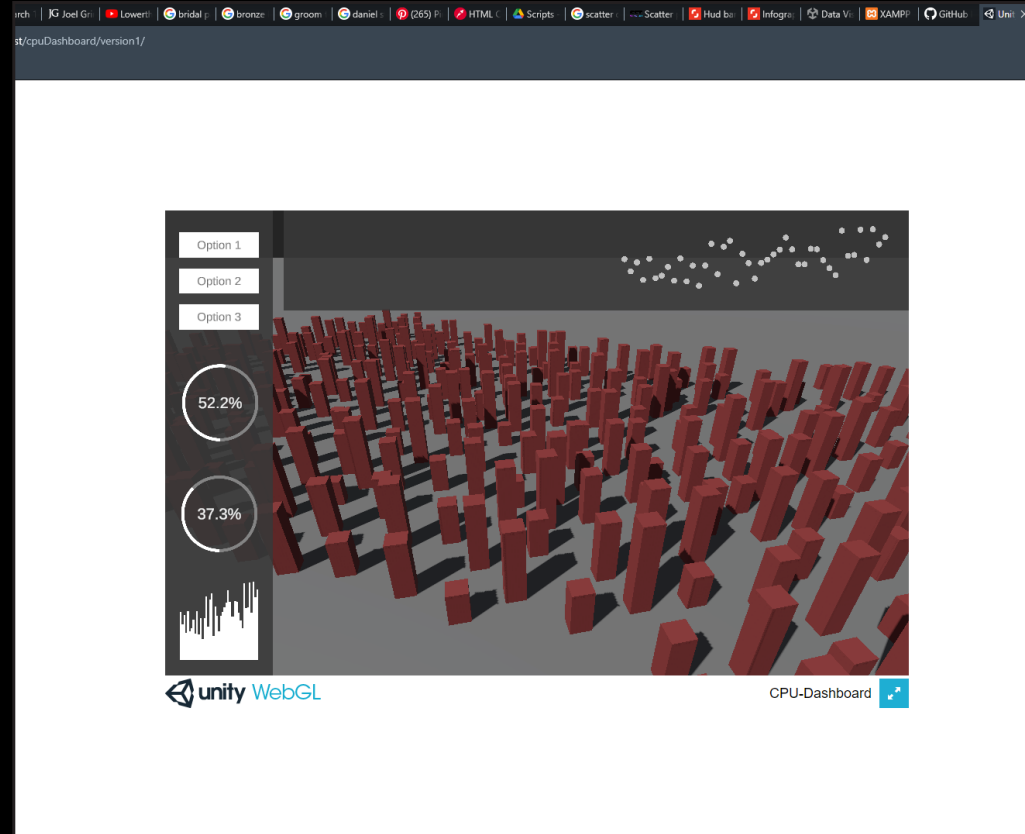
To view the files, now you need to either have a webserver running locally or over the internet to view the content.

You can use XAMPP to run a local server for testing purposes.



Export To WebGL

With the project now exported you should have files similar to that on the screen shot on the right.

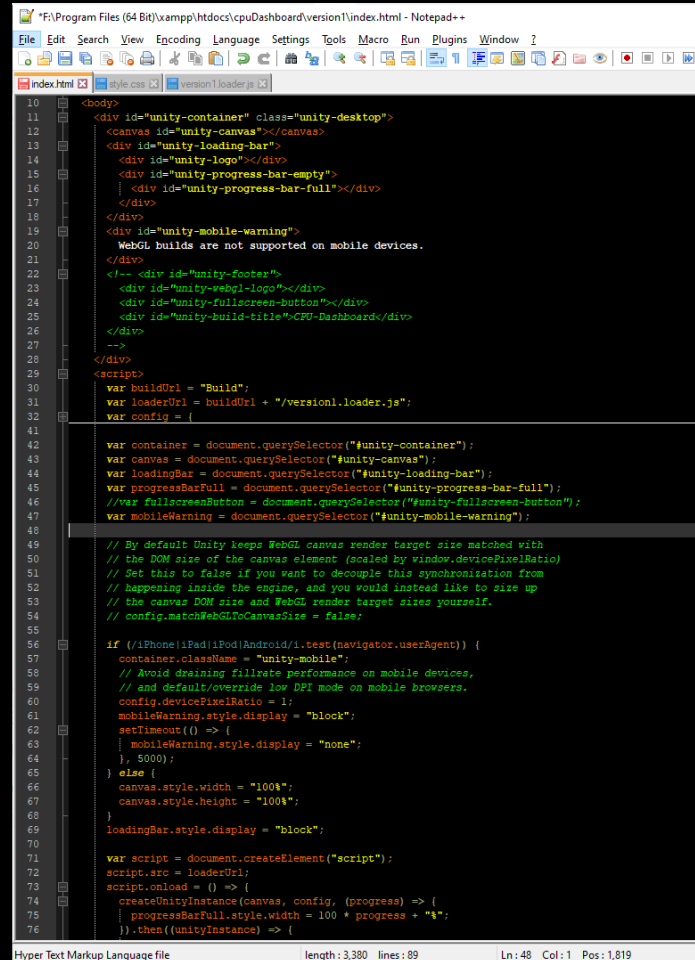


Remove Unity Footer & make fullscreen

To Remove the footer, comment out the all in the div tag from line 22 to 26 in the index.html file.

Also comment out the var fullscreenButton line on line 46 and 78 to 80. To prevent a null reference.

On line 12, remove the width and height next to the unity-canvas id and on line 66 and 67 change the values from a fixed one to 100% next to the unity-canvas id.

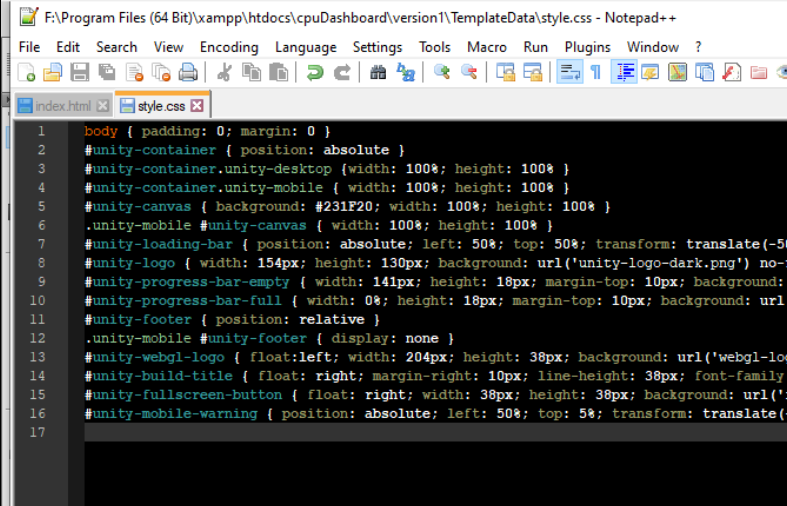


```
10 <body>
11 <div id="unity-container" class="unity-desktop">
12 <canvas id="unity-canvas"></canvas>
13 <div id="unity-loading-bar">
14 <div id="unity-logo"></div>
15 <div id="unity-progress-bar-empty">
16 | <div id="unity-progress-bar-full"></div>
17 </div>
18 </div>
19 <div id="unity-mobile-warning">
20 WebGL builds are not supported on mobile devices.
21 </div>
22 <!-- <div id="unity-footer">
23 <div id="unity-webgl-logo"></div>
24 <div id="unity-fullscreen-button"></div>
25 <div id="unity-build-title">CPU-Dashboard</div>
26 </div>
27 -->
28 </div>
29 <script>
30 var buildUrl = "Build";
31 var loaderUrl = buildUrl + "/version.loader.js";
32 var config = {
33
34 var container = document.querySelector("#unity-container");
35 var canvas = document.querySelector("#unity-canvas");
36 var loadingBar = document.querySelector("#unity-loading-bar");
37 var progressBarFull = document.querySelector("#unity-progress-bar-full");
38 //var fullscreenButton = document.querySelector("#unity-fullscreen-button");
39 var mobileWarning = document.querySelector("#unity-mobile-warning");
40
41 // By default Unity keeps WebGL canvas render target size matched with
42 // the DOM size of the canvas element (scaled by window.devicePixelRatio)
43 // Set this to false if you want to decouple this synchronization from
44 // happening inside the engine, and you would instead like to size up
45 // the canvas DOM size and WebGL render target sizes yourself.
46 // config.matchWebGLToCanvasSize = false;
47
48 if (!/iPhone|iPad|iPod/Android/.test(navigator.userAgent)) {
49 container.className = "unity-mobile";
50 // Avoid draining fillrate performance on mobile devices,
51 // and default/override low DPI mode on mobile browsers.
52 config.devicePixelRatio = 1;
53 mobileWarning.style.display = "block";
54 setTimeout(() => {
55 mobileWarning.style.display = "none";
56 }, 5000);
57 } else {
58 canvas.style.width = "100%";
59 canvas.style.height = "100%";
60 }
61 loadingBar.style.display = "block";
62
63 var script = document.createElement("script");
64 script.src = loaderUrl;
65 script.onload = () => {
66 createUnityInstance(canvas, config, (progress) => {
67 | progressBarFull.style.width = 100 * progress + "%";
68 }).then((unityInstance) => {
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

Remove Unity Footer & make fullscreen

Update the css file in the TemplateData folder to reflect the screenshot, as this will allow the application to now launch filling the screen.

Line 3 and Line 5, will need to be updated to 100% width and height.



```
F:\Program Files (64 Bit)\xampp\htdocs\cpuDashboard\version1\TemplateData\style.css - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
index.html style.css
1 body { padding: 0; margin: 0 }
2 #unity-container { position: absolute }
3 #unity-container.unity-desktop { width: 100%; height: 100% }
4 #unity-container.unity-mobile { width: 100%; height: 100% }
5 #unity-canvas { background: #231F20; width: 100%; height: 100% }
6 .unity-mobile #unity-canvas { width: 100%; height: 100% }
7 #unity-loading-bar { position: absolute; left: 50%; top: 50%; transform: translate(-50%, -50%); }
8 #unity-logo { width: 154px; height: 130px; background: url('unity-logo-dark.png') no-repeat center center; }
9 #unity-progress-bar-empty { width: 141px; height: 18px; margin-top: 10px; background: #231F20; }
10 #unity-progress-bar-full { width: 0%; height: 18px; margin-top: 10px; background: url('unity-progress-bar-full.png') no-repeat center center; }
11 #unity-footer { position: relative }
12 .unity-mobile #unity-footer { display: none }
13 #unity-webgl-logo { float:left; width: 204px; height: 38px; background: url('webgl-look.png') no-repeat center center; }
14 #unity-build-title { float: right; margin-right: 10px; line-height: 38px; font-family: sans-serif; font-size: 14px; }
15 #unity-fullscreen-button { float: right; width: 38px; height: 38px; background: url('fullscreen.png') no-repeat center center; }
16 #unity-mobile-warning { position: absolute; left: 50%; top: 5%; transform: translate(-50%, -50%); }
17
```

* Once these settings have been changed, you will only need to update the build folder, you will need to make sure that you have a version in a different location to where you are building the actual project, so that you don't need to repeat the previous steps.

Points to Consider

Points to Consider

Check The Responsiveness of the application on different screen sizes

(Look at utilizing anchors on UI elements and aspect ratio filter script)

**Utilising baking lighting to improve performance
rather than real-time lights**

Optimising 3D models when importing into Unity

(This is related to polygon count, and less is better for good FPS)

Make sure scale is 1 to 1 from the native modelling package

(Ideal for calculating lighting)

Stay Curious!