Immersive.Unity.Vis - A Library of Unity3D for Visualization in AR/VR/MR

Aidong Lu
aidong.lu@uncc.edu
University of North Carolina at Charlotte

July 2019

1 Introduction

With the advances of augmented reality (AR), virtual reality (VR), and mixed reality (MR), there is a need to visualize various data on devices beyond desktop computers.

This work provides a library of example immersive visualization and interaction projects built with Unity3D for AR (such as HoloLens) and VR (such as Oculus Rift and HTC Vive). Each example can be downloaded and run with Unity3D separately, which makes modification easy for developers. To deploy these examples on a particular device, the corresponding API should be downloaded and used to compile the executable program (please refer to the developer documentation of selected devices).

We have chosen most commonly used visualizations, including bar chart, line chart, scatter plot, etc. Most of our projects work by reading data from CSV files or random numbers. Interaction functions are also provided in some projects, such as mouse hover or selection.

The main purposes of this library is to provide the community of people who are interested in the field of immersive analytics and visualization, including students who are in related visualization, AR, VR, or gaming courses and developers who are learning new techniques.

All the examples are built with the Unity 2018.4 version unless specified otherwise. Most of them should work for the latest Unity version as we try to

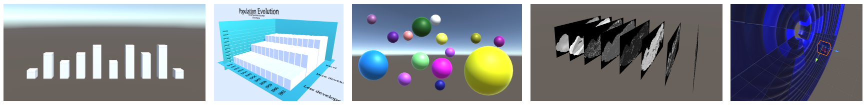
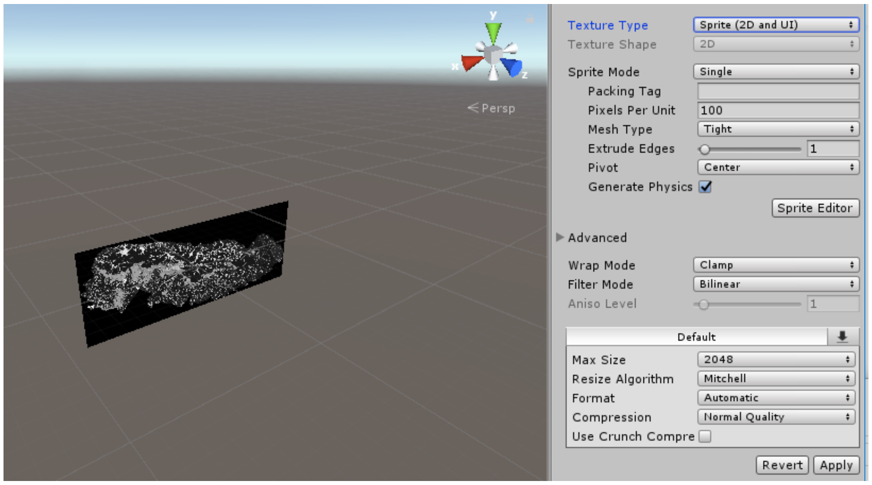


Figure 1: Examples of Immersive.Unity.Vis library.



Texture Type: Sprite (2D and UI)
Texture Shape: 2D
Sprite Mode: Single
Packing Tag:
Pixels Per Unit: 100
Mesh Type: Tight
Extrude Edges: 1
Pivot: Center
Generate Physics ☑
Advanced
Wrap Mode: Clamp
Filter Mode: Bilinear
Aniso Level: 1
Default
Max Size: 2048
Resize Algorithm: Mitchell
Format: Automatic
Compression: Normal Quality
Use Crunch Compre □

Figure 2: An image is converted to a sprite.

only use the basic Unity functions. We plan to continue to update the library and welcome new examples suggested by the community.

The following describes the functions of each example in this library, in the order of easy to hard. Students are advised to follow this order to review the examples.

2 Stacked Images

Let's start with a simple and useful example of stacked images. This example shows multiple images placed parallel in space.

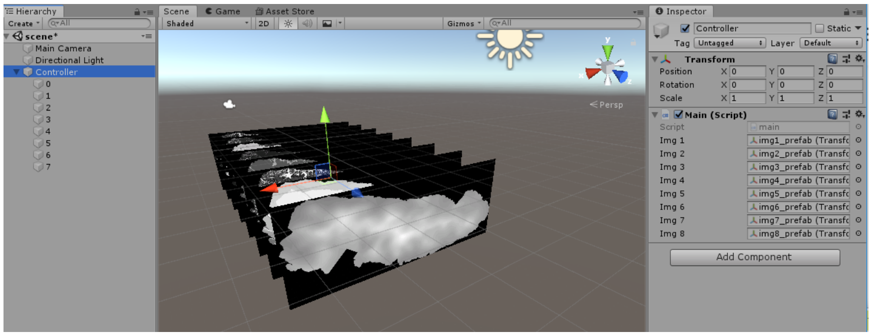
The first step is to import all images into the 'Asset' folder. As shown in Figure 2, the 'Texture Type' needs to be changed to 'Sprite (2D and UI)'. Apply the change and convert it to a prefab.

The second step is to specify the image prefabs in the controller, as shown in Figure 3. The controller also holds a script 'main', which places all images in space.

3 Bar Chart

This is a basic bar chart visualization. This example introduces the Unity 'prefab', which is a pre-designed model that can be created multiple times during run time. As shown in Figure 4, we create a 'bar' prefab by simply dragging a 'cube' object into the asset folder. All the bars in the bar chart are created by initializing this prefab. Two functions are provided in this example.

1. Bar chart

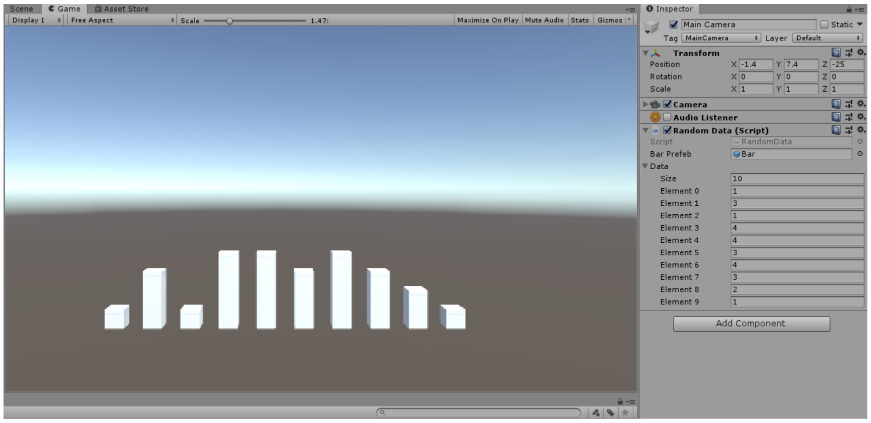


Controller ☑ □ Static
Tag Untagged Layer Default

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

☑ Main (Script)
Script main
Img 1 img1_prefab (Transfc
Img 2 img2_prefab (Transfc
Img 3 img3_prefab (Transfc
Img 4 img4_prefab (Transfc
Img 5 img5_prefab (Transfc
Img 6 img6_prefab (Transfc
Img 7 img7_prefab (Transfc
Img 8 img8_prefab (Transfc

Figure 3: The controller in the stacked-image example.



Main Camera ☑ □ Static
Tag MainCamera Layer Default

Transform
Position X -1.4 Y 7.4 Z -25
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

☑ Camera
□ Audio Listener
☑ Random Data (Script)
Script RandomData
Bar Prefeb Bar
Data
Size 10
Element 0 1
Element 1 3
Element 2 1
Element 3 4
Element 4 4
Element 5 3
Element 6 4
Element 7 3
Element 8 2
Element 9 1

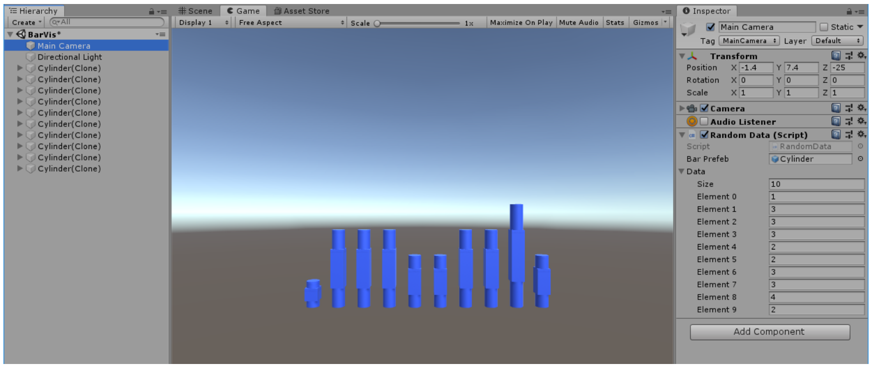
Figure 4: Bar chart example.

2. Random data generation

There are multiple ways to generate the basic bar chart. This example uses a prefab as the bar, which can be easily switched to different shapes instead of the regular bar. Figure 5 uses a different prefab with blue cube and cylinder to replace the regular bars. Note that the position of the prefab may affect the placement of shape in space. We have two options for locating the 'y' position.

4 Bubble Chart

This is a basic bubble chart visualization. This example generates a simple data file with 2 attributes, and computes the locations and sizes of bubbles to render the data elements. The parameters are chosen to avoid collision of bubbles. As shown in Figure 6, each bubble is rendered at different location and color. When hovering on a bubble, the color changes randomly to reflect the selection. Three functions are provided in this example.



BarVis*
Main Camera
Directional Light
Cylinder(Clone)
Cylinder(Clone)
Cylinder(Clone)
Cylinder(Clone)
Cylinder(Clone)
Cylinder(Clone)
Cylinder(Clone)
Cylinder(Clone)
Cylinder(Clone)
Cylinder(Clone)

☑ Main Camera □ Static
Tag MainCamera Layer Default

Transform
Position X -1.4 Y 7.4 Z -25
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

☑ Camera
□ Audio Listener
☑ Random Data (Script)
Script RandomData
Bar Prefeb Cylinder
Data

Size	10
Element 0	1
Element 1	3
Element 2	3
Element 3	3
Element 4	2
Element 5	2
Element 6	3
Element 7	3
Element 8	4
Element 9	2

Figure 5: Alternative forms of a bar chart. The prefab can be replaced with new shapes.

1. Bubble chart
2. Generate a simple data file with 2 attributes (dataset.cs)
3. Interactive mouse selection based on screen activity.

5 Scatter Plot

This is a basic scatter plot visualization. This example reads data from a csv file, and creates basic sphere objects as the points of 3D scatter plot. The data file contains the locations of points and an attribute value. As shown in Figure 7, each cluster is rendered with a selected color. Two functions are provided in this example.

1. Reading csv data file
2. Render scatter plot in 3D

6 Node-Link Diagram

This example visualizes a classical node-link diagram with the basic sphere model for nodes and Unity LineRenderer for links. As shown in Figure 8, a random procedure is used to determine the locations of nodes in 3D. Several functions are provided in the project:

1. Graph data with nodes and links as a csv file.
2. Generation of nodes and links based on data in run time.
3. Labels of nodes for their indexes.

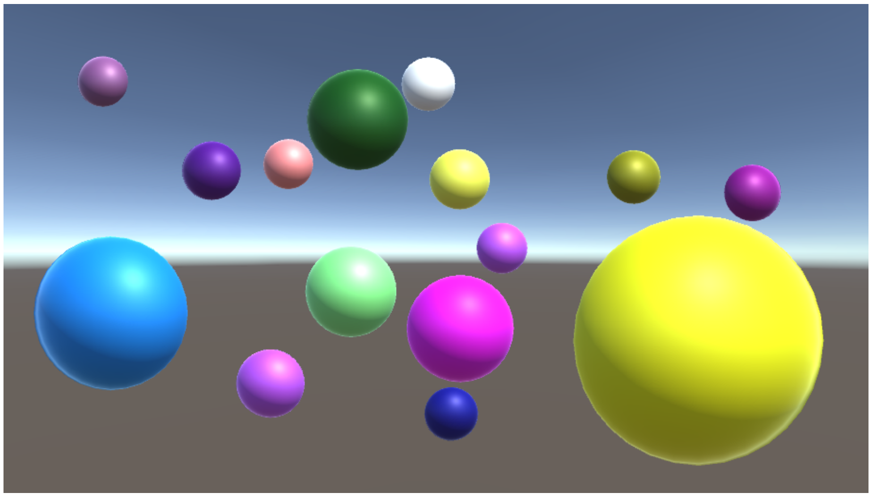


Figure 6: Bubble chart example.

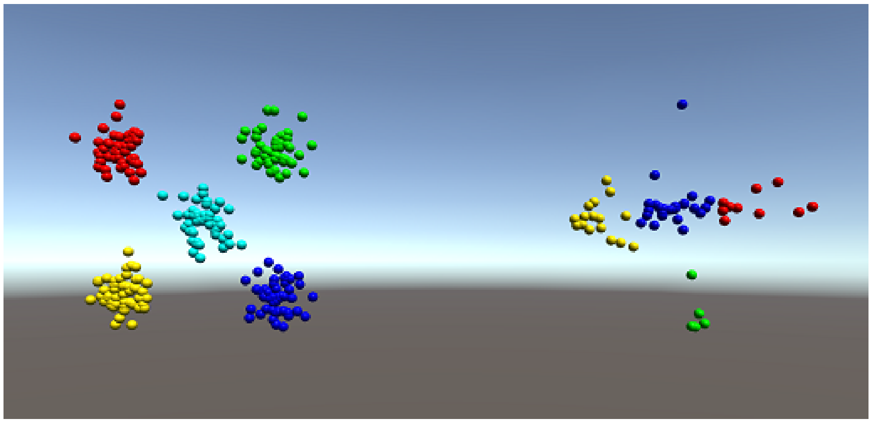


Figure 7: Scatter plot example.

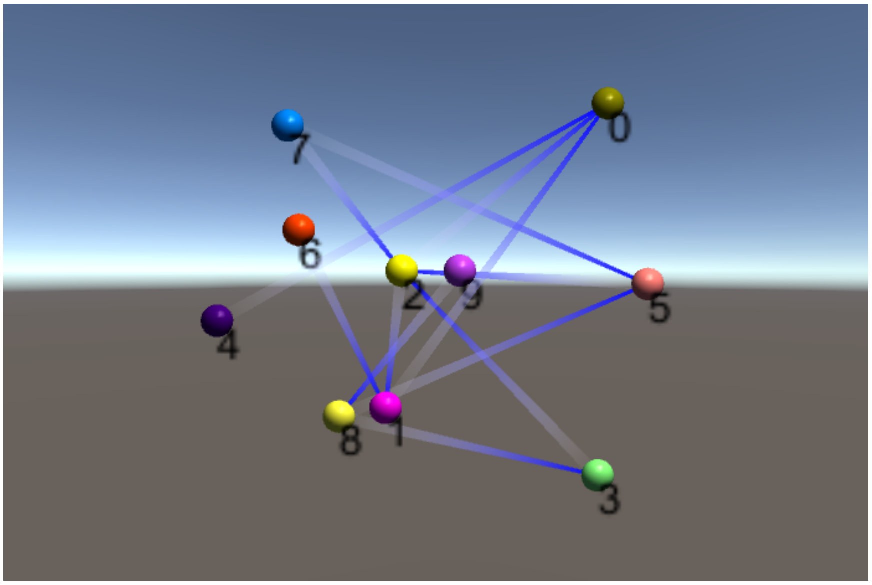


Figure 8: Nodelink diagram example.

7 Stacked Bars

This is an example of 2D stacked bars in 3D space. As shown in Figure 9, two data attributes are rendered with two layers of bar chart, one in gray and the other in green. There are three prefabs generated for rendering the bars (GameObject), axis (axis), and data values (New Text); all shown under the 'prefab' folder. All the prefabs and scripts are added to an empty object 'controller". Figure 10 shows the generated bars from prefab during run time. Three functions are provided in this example.

1. Reading csv data
2. Rendering 2D stacked bar chart
3. Showing data value

8 Bar Chart in 3D

This is a basic scatter plot visualization. To get started, double click the 3dChart scene under the 'scene' directory. This example reads data from a json file, and creates bars in 3D. All the scripts are added to the 'main camera'. As shown in Figure 11, the data is organized from low to high and can be interactive selected.

This example requires more Unity programming background on function calls and usage of prefabs. Several functions are provided in this example.

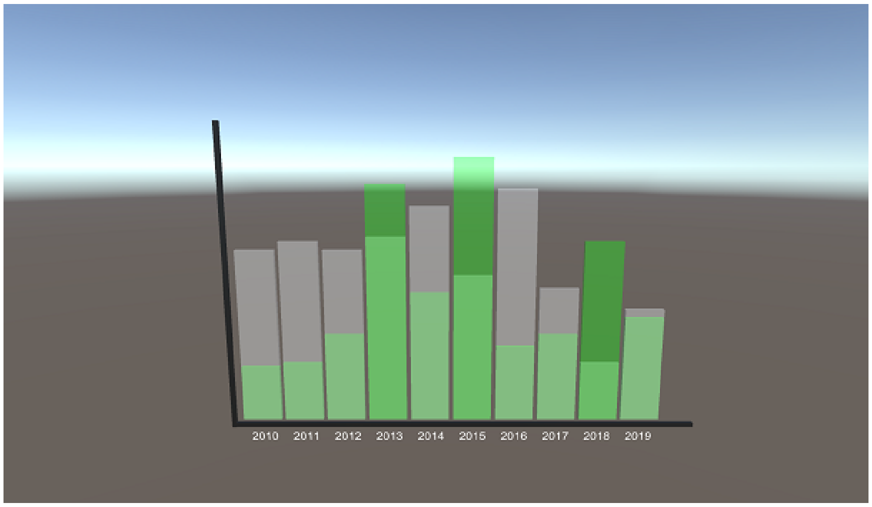


Figure 9: Stacked bars.

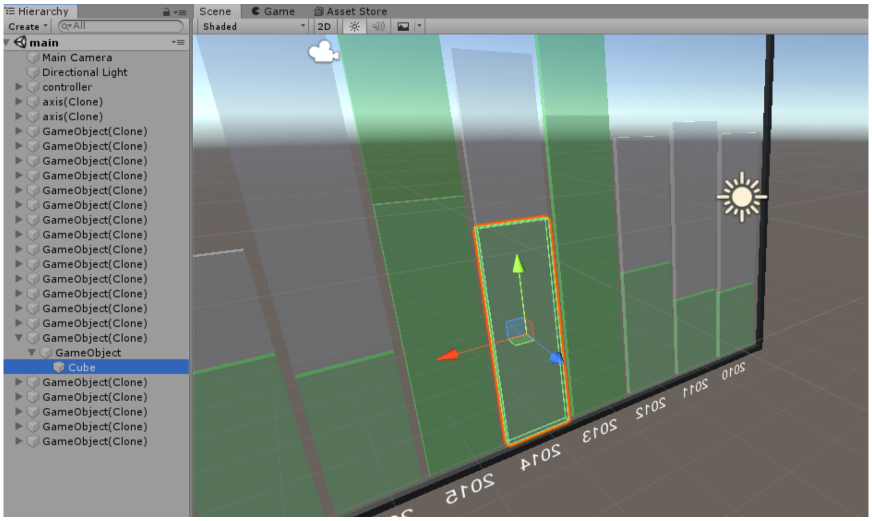


Figure 10: Stacked bars during run time.

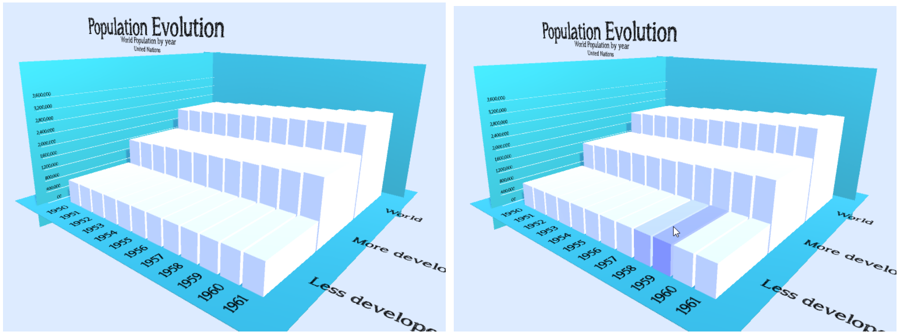


Figure 11: 3D bar chart example.

1. 3D bar chart
2. Labels for bar chart
3. Interactive mouse selection and text labels for showing data
4. Switching datasets from object ‘main camera’
5. Loading json dataset with the ‘Boomlagoon’ project [1]
6. Animated bar effect
7. Creating axis for bar chart

9 Circular Plot

This is a visualization of circular plot visualization, which is composed of 3D wedges. This example reads data from a csv file, and creates custom meshes as objects in the rendering. The data file contains a list of data values. As shown in Figure 12, Each object in the same customized model is colored based on the data value. Three functions are provided in this example.

1. Circular chart
2. Read csv data file (DataAdaptor.cs)
3. Create 2D and 3D custom meshes for special models. The meshes can be determined by the end points, which are manually edited, shown in Figure 13.

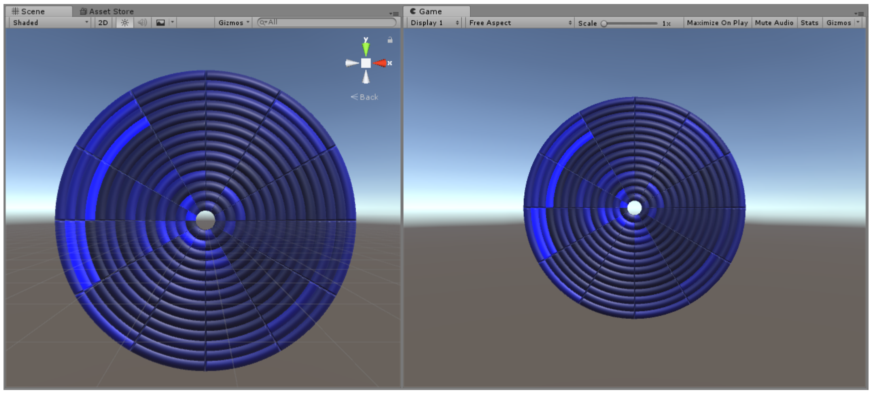
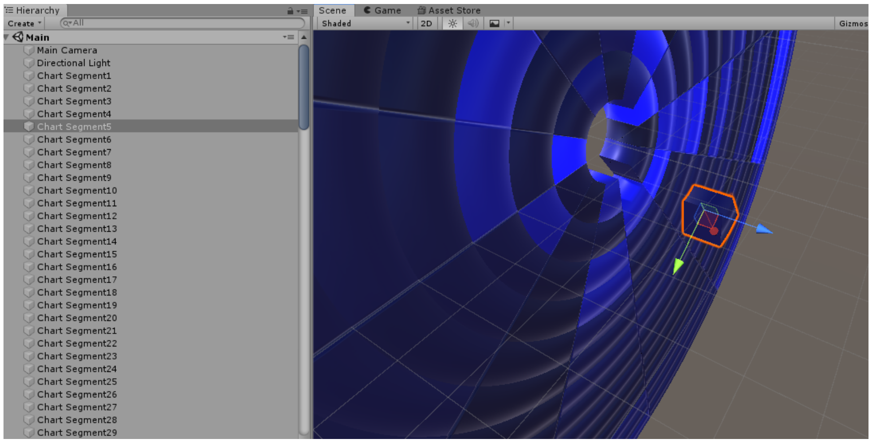
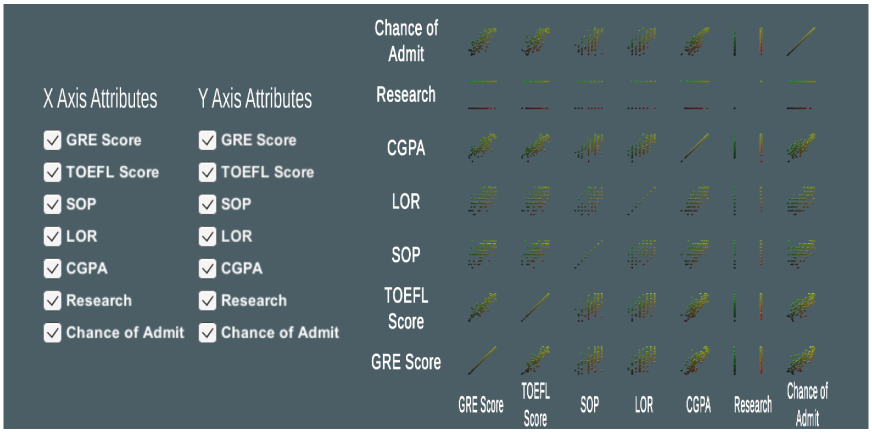


Figure 12: Circular plot example.



Main
Main Camera
Directional Light
Chart Segment1
Chart Segment2
Chart Segment3
Chart Segment4
Chart Segment5
Chart Segment6
Chart Segment7
Chart Segment8
Chart Segment9
Chart Segment10
Chart Segment11
Chart Segment12
Chart Segment13
Chart Segment14
Chart Segment15
Chart Segment16
Chart Segment17
Chart Segment18
Chart Segment19
Chart Segment20
Chart Segment21
Chart Segment22
Chart Segment23
Chart Segment24
Chart Segment25
Chart Segment26
Chart Segment27
Chart Segment28
Chart Segment29

Figure 13: The customized mesh in the circular plot.



X Axis Attributes

- ☑ GRE Score
- ☑ TOEFL Score
- ☑ SOP
- ☑ LOR
- ☑ CGPA
- ☑ Research
- ☑ Chance of Admit

Y Axis Attributes

- ☑ GRE Score
- ☑ TOEFL Score
- ☑ SOP
- ☑ LOR
- ☑ CGPA
- ☑ Research
- ☑ Chance of Admit

Figure 14: Scatter plot matrix example.

10 Scatter Plot Matrix

This example of scatter plot matrix visualizes the Graduate Admissions dataset [2], as shown in Figure 14.

This project introduces to the use of toggle button in Unity. When certain toggle buttons are checked the corresponding scatter plot matrix will be generated. Toggle buttons are part of Canvas under the 'UI' component in Unity. When a canvas is added to the scene Unity adds an Event System along with canvas. This Event System is responsible for detecting the keyboard, mouse and touch events and directs these events to the corresponding game object.

Here are three scripts in the project:

1. Parser.cs: This file is responsible of parsing the csv file and storing it into a Dictionary along with the headers.

2. Plot.cs: This script is used to plot the individual datapoints based on the values of the dataset. Since the values for certain columns like GRE, TOEFL.. are large, the script also normalizes their values so that the values lie within 0-1 range and the datapoints appear close to each other. With the help of these new noramlized values we then plot the datapoints using Instantiate() method and assign color.

3. AxisToggle.cs: This scripts maitains a list of toggle buttons which needs to be displayed in the Game view. OnToggleValueChanged() method is an event based method which is called when a toggle button is enabled or disabled.

11 Conclusion and Future Work

This library currently only contains a list of basic examples of immersive visualization and interaction. The main purpose is to provide examples to promote the research and education on the topic of immersive analytics. As the field is developing fast, other valuable resources are also available, such as [3]. We hope to continue to expand this library and maintain the codes for any issues and new versions of Unity3D.

12 Acknowledgement

This library is built from several main contributors and our students from immersive visualization courses. The main contributors include: Timothy Hayduk (graduated in May 2018), Willis Fulmer (graduated in Dec 2018), and Tahir Mahmood (graduated in May 2019).

This work was supported by the National Science Foundation under Grant Nos. 1629913.

References

[1] https://bitbucket.org/boomlagoon/boomlagoon-json/src/default/.

[2] https://www.kaggle.com/mohansacharya/graduate-admissions.

[3] https://github.com/ronellsicat/DxR.