AUTHOR	Michael Soler
CONTACT	michael.soler.beatty@gmail.com
Unity Ver.	2018.3.5f1

Index

1.Description of the package.

From cardboard buddies we pretend to give the best packages to our customers with simplicity and transparency. This package allows the user to create a realtime trading interface that simulates a stock market value.

This asset contains the following:

- Scripts that generate the chart from generated data.
- Scripts that control the realtime ticks.
- Scripts that change the appearance of the chart and its parts.
- Scripts that control the selectable line and gives the values of the chart.
- Buying, selling and closing operations with a specific volume.

The asset contains the necessary models, textures and prefabs shown in the video.

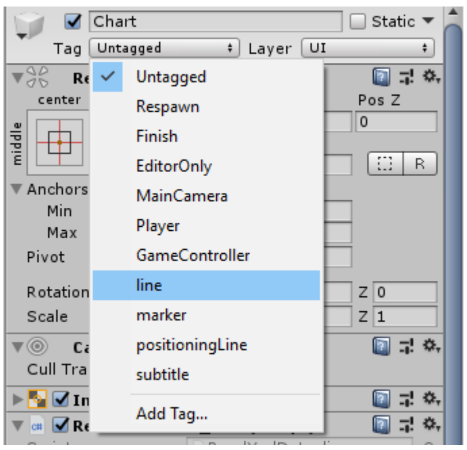
For further information please contact michael.soler.beatty@gmail.com.

2.Dependencies of the package (importing).

This package does not need any other package to work.

3.Colliders, tags and physics

UI elements are used to generate events. Please check that the following tags are in your project:



4.Scripting and dynamics

We use videotutorials to explain the package dynamics and the scripts. The main scripts used in this asset are.

```
public class GenerateDataStream : MonoBehaviour
{

    //public initial capital and benefit
    public float capital;
    public float initialCapital=1000;
    Text capital_txt;
    Text benefit_txt;

    // line parameters
    [Range(0,15)]
    public float L_width = 5f;

    [Range(0, 15)]
    public float M_width = 5f;

    [Range(2, 10)]
    public int nb__div = 5;

    [Range(0, 10)]
    public int HL_width = 1;

    [Range(0, 500)]
    public int nb_initial_Ticks = 200;

    public int arraySize = 100000;

    // line prefab for UP and DOWN behaviour
    public GameObject prefab_UP, prefab_DOWN, prefab_HORIZ;

    //time counting event
    float elapsed=100000, elapsed2;

    //these two arrays contain the values for the line chart
    public float[] x_value;
    public float[] y_Open_value;
    public float[] y_Close_value;
    public float[] y_min_value;
    public float[] y_Max_value;

    //maximum values for x and y
    public float xmax, ymax, xmin, ymin;

    //tick_latency [s]
    public float tick_latency=1;
    //tick_duration [s]
    public float tick_duration = 10;

    //this is the value of the market at the instant given
    float y_value;

    //this is the initial conditions for the simulation
    public float min_seed_value = 100;
    public float max_seed_value = 200;
    public float vol=10;

    //these are the gameobjects containing the markers
    GameObject[] goMaxMin;
    GameObject[] goOpenClose;

    // intermediate vectors
    Vector3[] vMax ;
    Vector3[] vmin ;
    Vector3[] vop ;
    Vector3[] vcl ;

    //this is the horizontal line
    Transform lineH;

    //these are the variables used to change the size of the render of the
chart
    float tf_FactorA;
    float tf_FactorB;
    float a;
    float b;

    // colors for the markers
    public Color colUP, colDOWN;

    //fork
    public float fork = 1;

    //variables for trading
    Text buy_txt, sell_txt;
    Button buy_but, sell_but, close_but;


    // selling and buying values
    float set_buy=0;
    float set_sel=0;
    float y_buy, y_sell;

    //this is the trading state
    //    0--> none,  1-->buy  2-->sell
    public int tradingState=0;

    // line horizontal for trading
    public GameObject tradingLine;

    //volume value
    InputField input_Volume_txt;
    int val_volume;
}
```

```
public class OverLine : MonoBehaviour
{
    // Start is called before the first frame update
    Transform lineTF;

    void Start()
    {
        lineTF = GameObject.FindGameObjectWithTag("positioningLine").transform;
    }

    // Update is called once per frame
    public void onEnter()
    {
        lineTF.transform.localPosition = new
Vector3(transform.localPosition[0],0,0);

        //set the text and images to true
        transform.GetChild(0).GetComponent<Image>().enabled = true;
        transform.GetChild(1).GetComponent<Image>().enabled = true;
        transform.GetChild(2).GetComponent<Text>().enabled = true;
        transform.GetChild(3).GetComponent<Text>().enabled = true;
        transform.SetAsLastSibling();
    }

    public void onExit()
    {
        //set the text and images to false
        transform.GetChild(0).GetComponent<Image>().enabled = false;
        transform.GetChild(1).GetComponent<Image>().enabled = false;
        transform.GetChild(2).GetComponent<Text>().enabled = false;
        transform.GetChild(3).GetComponent<Text>().enabled = false;
    }
}
```

5. Video tutorials

We have a video tutorial explaining how package mechanics works.

https://youtu.be/76A5bPuFsno

We also comment the scripts in this video tutorial.

https://youtu.be/tfQPfCtAdkQ

6. Exporting to android

Please notice that the pointer enter and exit event work differently in Android.