EN

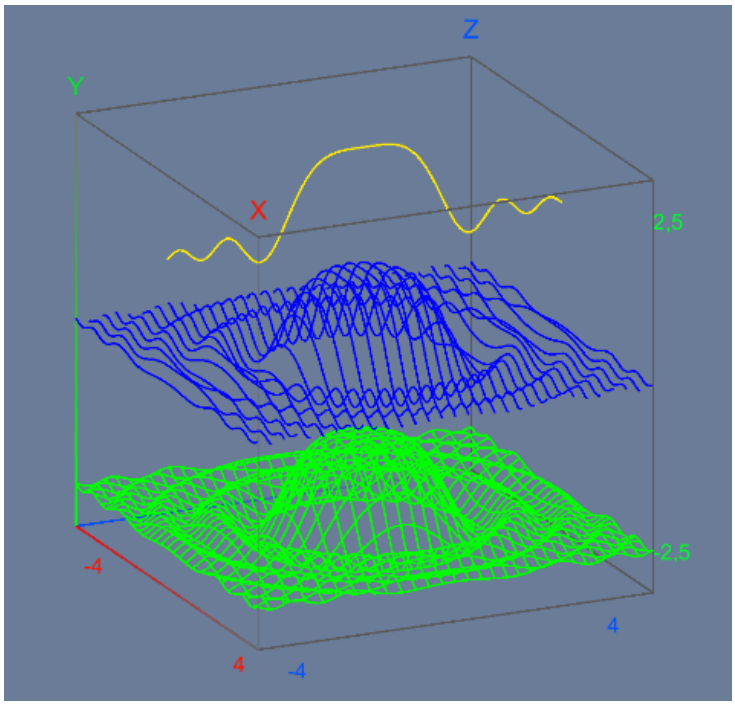
EN

Table of contents

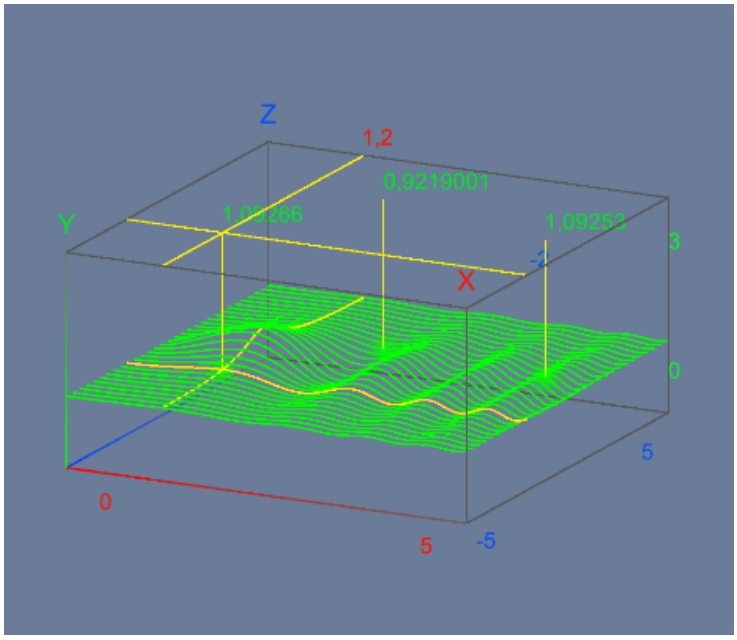
Introduction

Asset Graph3D allows you to place on the Canvas the desired number of panels (Panel_Graph3D) for building 3D graphs.

Panel_Graph3D is a prefab that is built for the most convenient editing and creating panels to your liking.
Each Panel_Graph3D panel can include more than one graph in the form of a curve (Curve), an array of parallel curves (Curves) and a plane - grid (Surface).

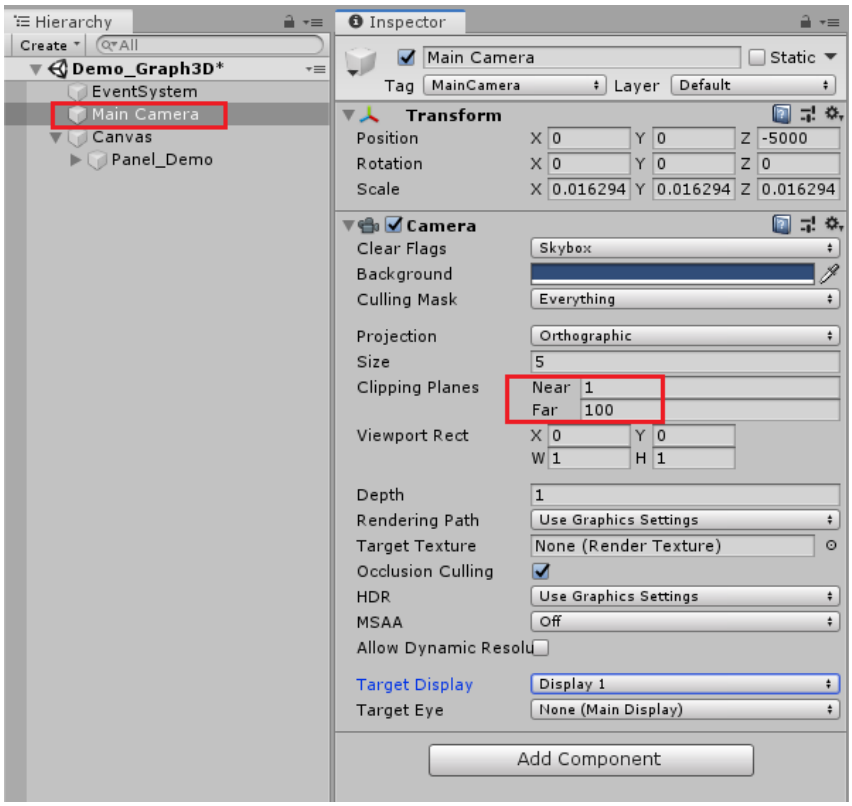


On each graph, you can display several point markers with function values and argument values. The appearance of the marker can be configured through a script, as well as by editing prefab pMerker.



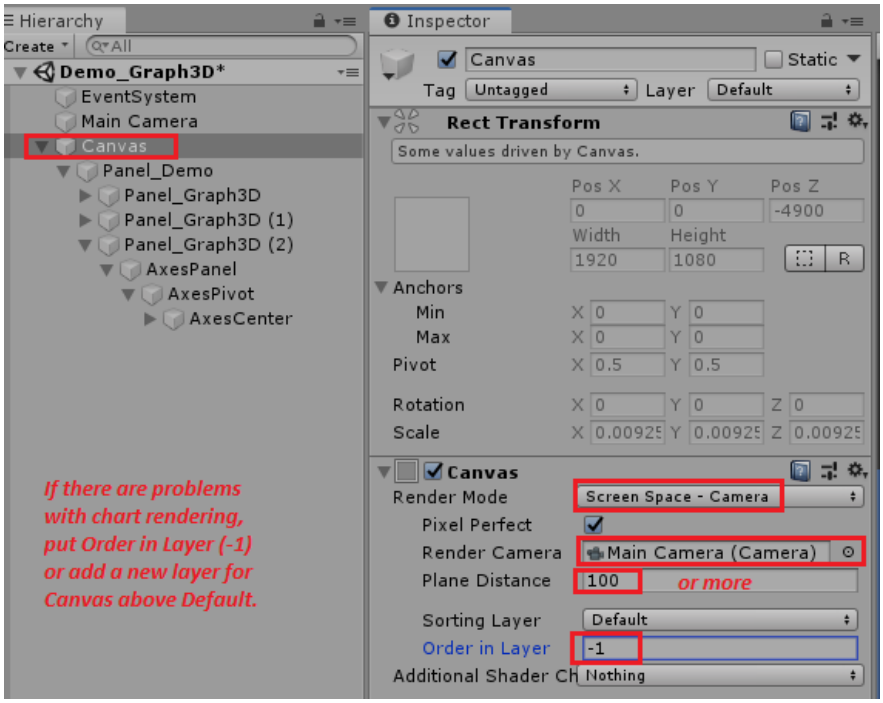
1. Project setup

1.1. Camera setup

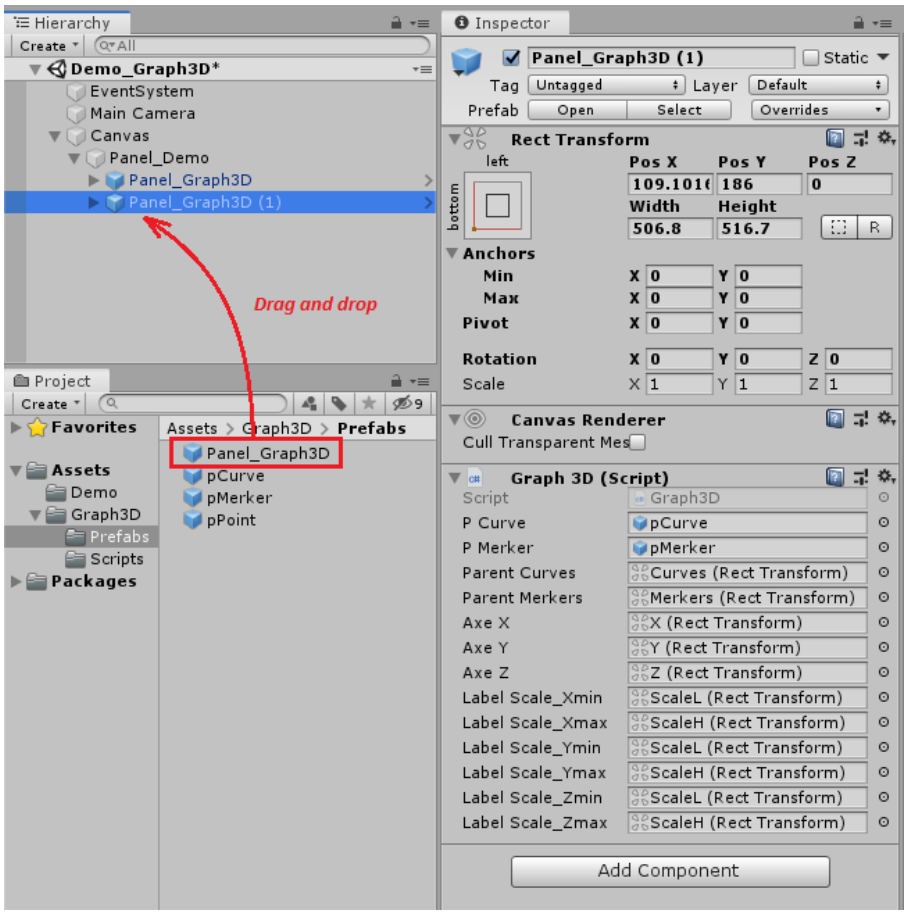


If the image of the graphic panel is missing or cropped, set the appropriate parameters for the camera visibility distance.

1.2. Canvas setup

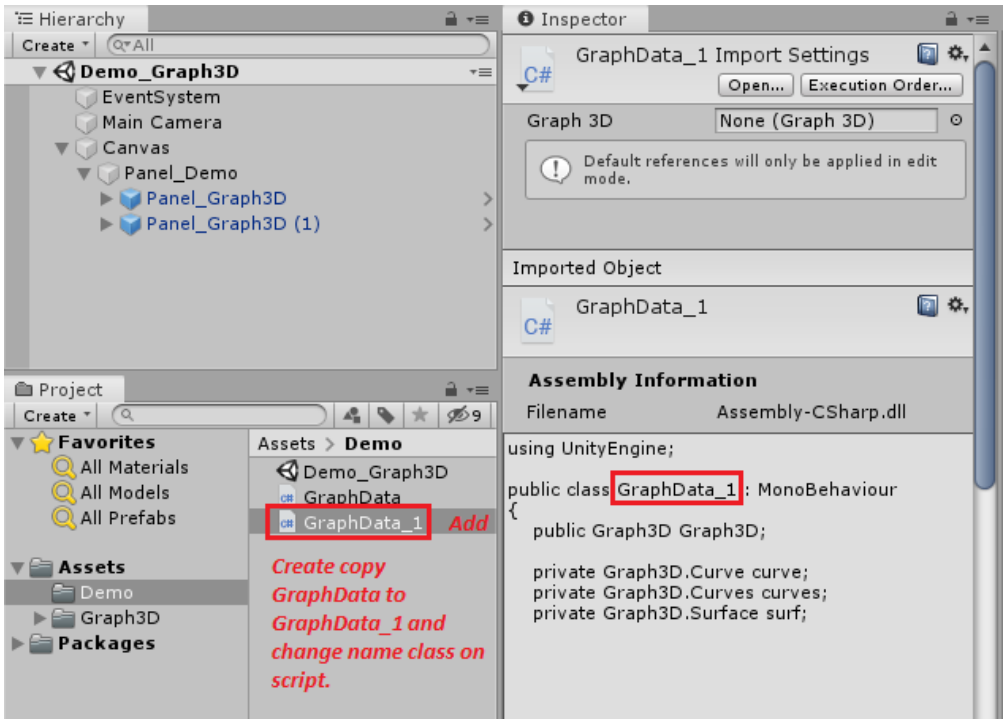


1.3. Adding a panel to graph(s) Panel_Graph3D



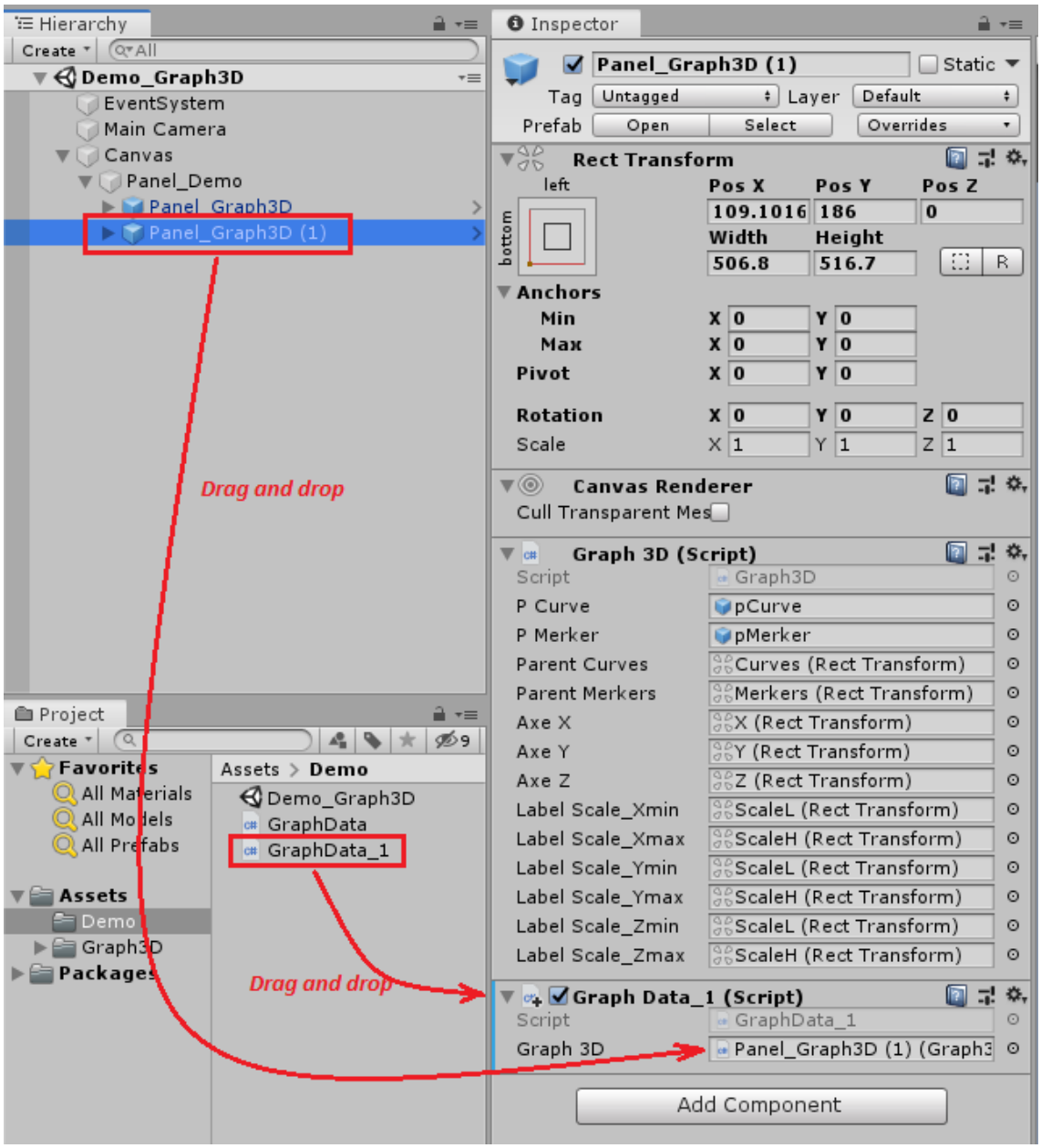
To add a new panel, drag prefab Panel_Graph3D to the desired location in the project hierarchy.

1.4. Setting a new panel Panel_Graph3D



For the new panel Panel_Graph3D, you need to create a new script for the parameters of the graph(s) of this panel.

- Make a copy of the existing GraphData script (for example, in the file manager (explorer) of your project).
- Give any name to the new script (for example GraphData_1)
- Open a new script (GraphData_1) and change the class name to the script name (in the example GraphData_1)
- Link the script (GraphData_1) of the parameters of the graph (s) to the new Panel_Graph3D (1) panel as in the figure below.



- Do not forget to attach the panel object itself (Panel_Graph3D (1)) in the script field.

Now you can start creating the graph(s) in the Panel_Graph3D!

2. Creating graph(s) in Panel_Graph3D

To do this, you need to fill in the data script (GraphData_1), focusing on the template.

2.1. Create an object or graph objects from the types: Graph3D.Curve, Graph3D.Curves, Graph3D.Surface.

```
public class GraphData_1 : MonoBehaviour
{
   public Graph3D Graph3D;      //Ссылка на панель отображения графиков

   public Graph3D.Curve curve;          //Одна кривая
   public Graph3D.Curves curves;        //Массив кривых
   public Graph3D.Surface surf;         //Поверхность в виде сетки кривых

private void Awake()
   {
      curve = new Graph3D.Curve(Graph3D);
      curves = new Graph3D.Curves(Graph3D);
      surf = new Graph3D.Surface(Graph3D);
                  . . .
```

2.1.1. Fill the data structure of objects

The default graph data structure is maximally filled when creating an object.

It is necessary to fill in only the necessary parameters for each type of graph.

Description of the data can be found in the script Graph3D.cs.

2.1.2. Required parameters for Graph3D.Curve

```
private void Awake()
   {
                                    . . .
// Set the type of graph - Curve with a constant parameter on the X, Y or Z axis
//Possible options: Curve_Xconst, Curve_Yconst, Curve_Zconst
      curve.data.typeGraph = Graph3D.TypeGraph.Curve_Xconst;    //X axis selected as an example

// Choose the location of the function and parameters along the axes of the graph panel
// Possible options: Y_XZ, Y_ZX, X_YZ, X_ZY, Z_XY, Z_YX
      curve.data.typeFunc = Graph3D.TypeFunc.Z_YX;
// As an example, a function along the Z axis with parameter 1 on the Y axis and parameter 2 on the X axis

// Assign the function to the given delegate float a (float b; float c)
      curve.data.func = funcABC;
// Link to a function with 2 parameters
// private float funcABC ( float b, float c )         //  Z_YX :     Z - function, Y – “b”,  X – “c”
// {
//    return b * b + c * c;    //return function
// }
```

```
// If a function with one parameter, then the other should be = 0, for example
// private float funcABC(0.0f , float c)
// {
//     return  c * c;
// }

// Set the scale in physical units for the axis of parameter 1 (in example X)
       curve.data.X.scale.L = -20.0f;    // Nearest to the center of coordinates
       curve.data.X.scale.H = 20.0f;     // Far from the center of coordinates
// Conditions: H> L or H <L; H! == L

// Set the scale in physical units for the axis of parameter 2 1 (in example Y)
       curve.data.Y.scale.L = -20.0f;          // Nearest to the center of coordinates
       curve.data.Y.scale.H = 20.0f;           // Far from the center of coordinates
// Conditions: H> L or H <L; H! == L

// Set the scale in physical units for the function axis (in example Z)
       curve.data.Z.scale.L = 100.0f;
       curve.data.Z.scale.H = 1000.0f;

// Set the number of curve segments (smoothness) along the axis (in example Y)
       curve.data.Y.segments = 20;

//Set the value of the constant parameter (in example X)
       curve.data.X.argConst = 5.0f;          // Graph3D.TypeGraph.Curve_Xconst

// Parameters that will not be used (for this example):
   //curve.data.X.segments
    //curve.data.Z.segments
    //curve.data.Y.argConst
    //curve.data.Z.argConst
```

2.1.3. Additional Options for Graph3D.Curve

```
private void Awake()
   {
                              . . .
// Autoscale along the axis of the function (in example Z)
       curve.data.autoScaleAxeFunc = true;          //def = false
// The flag will require double recalculation of the function (costly in time about 2 times)
// Using the flag still requires setting the maximum limits of scale.L and scale.H

// Scale for the graph3D panel axes in screen units (pixels)
       curve.data.X.scaleAxe.LScreen
       curve.data.X.scaleAxe.HScreen
       curve.data.Y.scaleAxe.LScreen
       curve.data.Y.scaleAxe.HScreen
       curve.data.Z.scaleAxe.LScreen
       curve.data.Z.scaleAxe.HScreen
// These parameters can be set in the script, but it is provided to do this interactively in the inspector of
//the editor (see section 3.1)
```

2.1.4. Required Parameters for Graph3D.Curves

```
private void Awake()
   {
                                        . . .
// Set the type of graph - Curves with constant parameters on the X, Y or Z axis
// Options: XCurves, YCurves, ZCurves
         curves.data.typeGraph = Graph3D.TypeGraph.ZCurves;   //Как пример выбрана ось Z

// Choose the location of the function and parameters along the axes of the graph panel
// Options: Y_XZ,  Y_ZX,  X_YZ,  X_ZY,  Z_XY,  Z_YX
          curves.data.typeFunc = Graph3D.TypeFunc.Y_XZ;
// As an example, a function along the Y axis with parameter 1 on the X axis and parameter 2 on the Z axis

// Assign the function to the given delegate float a (float b; float c)
          curves.data.func = funcABC;
// Link to a function with 2 parameters
// private float funcABC ( float b, float c )         //  Y_XZ :      Y - function, X – “b”,  Z – “c”
// {
//    return b * b + c * c;    //return function
// }
// If a function with one parameter, then the other should be = 0, for example
// private float funcABC(0.0f , float c)
// {
//    return  c * c;
// }

// Set the scale in physical units for the axis of parameter 1 (in example X)
         curves.data.X.scale.L = -20.0f;      // Nearest to the center of coordinates
         curves.data.X.scale.H = 20.0f;       // Far from the center of coordinates
// Options: H > L or H < L;   H !== L


// Set the scale in physical units for the axis of parameter 2 (in example Z)
         curves.data.Z.scale.L = -20.0f;      // Nearest to the center of coordinates
         curves.data.Z.scale.H = 20.0f;       // Far from the center of coordinates
// Options: H > L or H < L;   H !== L

// Set the scale in physical units for the axis of the function (in example Y)
         curves.data.Y.scale.L = 100.0f;      // Nearest to the center of coordinates
         curves.data.Y.scale.H = 1000.0f;   // Far from the center of coordinates
// Options: H > L or H < L;   H !== L

// Set the number of curve segments (smoothness) along the axis (in example X)
         curves.data.X.segments = 20;

// Set the number of parallel curves (in example Z)
         curves.data.Z.curves = 10;            // Graph3D.TypeGraph.ZCurves

// Parameters that will not be used (for this example):
          // curves.data.Y.segments
          // curves.data.Z.segments
          // curves.data.X.argConst
```

```
        // curves.data.Y.argConst
        // curves.data.Z.argConst
        // curves.data.X.curves
        // curves.data.Y.curves
```

2.1.5. Additional options for Graph3D.Curves

```
private void Awake()
  {
                        . . .
// Autoscale along the axis of the function (in example Y)
     curve.data.autoScaleAxeFunc = true;          //def = false
// The flag will require double recalculation of the function (costly in time about 2 times)
// Using the flag still requires setting the maximum limits of scale.L and scale.H

// Scale for the graph3D panel axes in screen units (pixels)
     curve.data.X.scaleAxe.LScreen
     curve.data.X.scaleAxe.HScreen
     curve.data.Y.scaleAxe.LScreen
     curve.data.Y.scaleAxe.HScreen
     curve.data.Z.scaleAxe.LScreen
     curve.data.Z.scaleAxe.HScreen
// These parameters can be set in the script, but it is provided to do this interactively in the inspector of
//the editor (see section 3.1)
```

2.1.5. Required Parameters for Graph3D.Surface

```
private void Awake()
  {
                                  . . .
// Set The Type Of The Graph - Surface
// Possible options: in the future there will be a choice of Grid or Polygon
       surf.data.typeGraph = Graph3D.TypeGraph. Surface;

// Choose the location of the function and parameters along the axes of the graph panel
// Options: Y_XZ,  Y_ZX,  X_YZ,  X_ZY,  Z_XY,  Z_YX
     surf.data.typeFunc = Graph3D.TypeFunc.X_YZ;
// As an example, a function along the X axis with parameter 1 on the Y axis and parameter 2 on the Z axis

// Assign the function to the given delegate float a (float b; float c)
       surf.data.func = funcABC;
// Link to a function with 2 parameters
// private float funcABC ( float b, float c )        //  X_YZ :     X - function, Y – “b”,  Z – “c”
// {
//    return b * b + c * c;    //return function
// }
// If a function with one parameter, then the other should be = 0, for example
// private float funcABC(0.0f , float c)
// {
//    return  c * c;
// }
```

```
// Set the scale in physical units for the axis of parameter 1 (in example Y)
        surf.data.Y.scale.L = -20.0f;          // Nearest to the center of coordinates
        surf.data.Y.scale.H = 20.0f;           // Far from the center of coordinates
// Options: H > L or H < L;   H !== L

// Set the scale in physical units for the axis of parameter 2 (in example Z)
        surf.data.Z.scale.L = -20.0f;          // Nearest to the center of coordinates
        surf.data.Z.scale.H = 20.0f;           // Far from the center of coordinates
// Options: H > L or H < L;   H !== L

// Set the scale in physical units for the axis of the function (in example X)
        surf.data.X.scale.L = 100.0f;                  // Nearest to the center of coordinates
        surf.data.X.scale.H = 1000.0f;       // Far from the center of coordinates

// Set the number of curve segments along the axes of parameter 1 and 2 (smoothness)
        surf.data.Y.segments = 20;
        surf.data.Z.segments = 20;

// Set the number of parallel curves for parameter 1 and 2 (in the example Y and X)
        surf.data.Y.curves = 10;
        surf.data.Z.curves = 10;

// Parameters that will not be used (for this example):
   // surf.data.X.segments
    // surf.data.X.argConst
    // surf.data.Y.argConst
    // surf.data.Z.argConst
    // surf.data.X.curves
    // surf.data.Y.curves
    // surf.data.Z.curves
```

2.1.6. Additional options for Graph3D.Surface

```
private void Awake()
   {
                                . . .
// Autoscale along the axis of the function (in example Z)
      curve.data.autoScaleAxeFunc = true;          //def = false
// The flag will require double recalculation of the function (costly in time about 2 times)
// Using the flag still requires setting the maximum limits of scale.L and scale.H

// Scale for the graph3D panel axes in screen units (pixels)
       curve.data.X.scaleAxe.LScreen
       curve.data.X.scaleAxe.HScreen
       curve.data.Y.scaleAxe.LScreen
       curve.data.Y.scaleAxe.HScreen
       curve.data.Z.scaleAxe.LScreen
       curve.data.Z.scaleAxe.HScreen
// These parameters can be set in the script, but it is provided to do this interactively in the inspector of
//the editor (see section 3.1)
```

2.2. Calculation of graphs

At this stage, using the DataSet () method, according to the data completed above, the graphs will do the calculation in memory without displaying on the screen. This method will be useful for preparing graphs, for example, through coroutines.

```
private void OnEnable()
  {
          carve.DataSet();
          carves.DataSet();
          surf.DataSet();
                                . . .
```

2.3. Display graphs

The Show method Show(Color color) displays the graph on the screen, provided that the method DataSet ( )has been made. The color of the graph curves is set by the parameter.

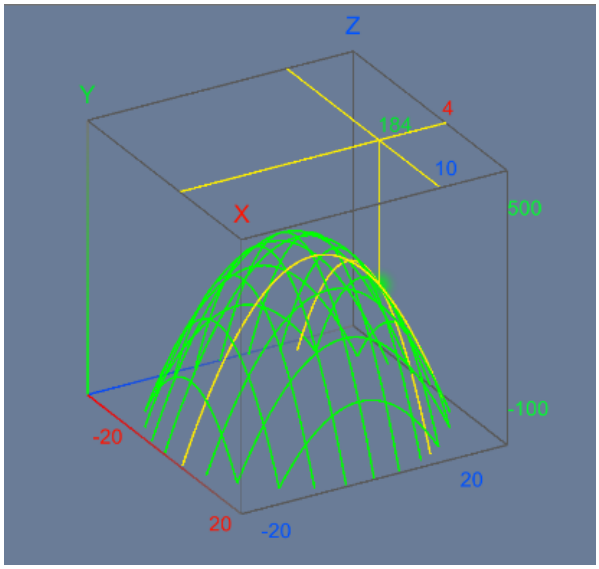
```
private void OnEnable()
  {
                . . .
        carve.Show(Color.yellow);
        carves.Show(Color.blue);
        surf.Show(Color.green);
```

2.4. Merkers on the graph

2.4.1. Merker display

There may be several Merkers on the graph.

The merker is a function point on the graph with certain coordinates that are specified through the parameters of the function.



- Use the method to create a merker:

Merker.Set(string name, float arg1, float arg2),
where name – name of merker, arg1, arg2 – parameters for graph function func(float b, float c), arg1 – “b”, arg2 – “c”, dec - a number of simbols after comma.

- Use the methods to configure the merker ch.2.5.2
- Use the method to display:

Merker.Show(string name, bool active),
where name – name of merker, active – visibility.

In our example:

```
private void OnEnable()
  {
                . . .
        carve.Merker.Set(“m1”, 10.0f, 15.0f);
        carves.Merker.Set(“m1”, 10.0f, 15.0f);
        surf.Merker.Set(“m2”, 10.0f, 15.0f);
        carve.Merker.Show(“m1”, true);
        carves.Merker.Show(“m1”, true);
        surf.Merker.Show(“m2”, true);
```

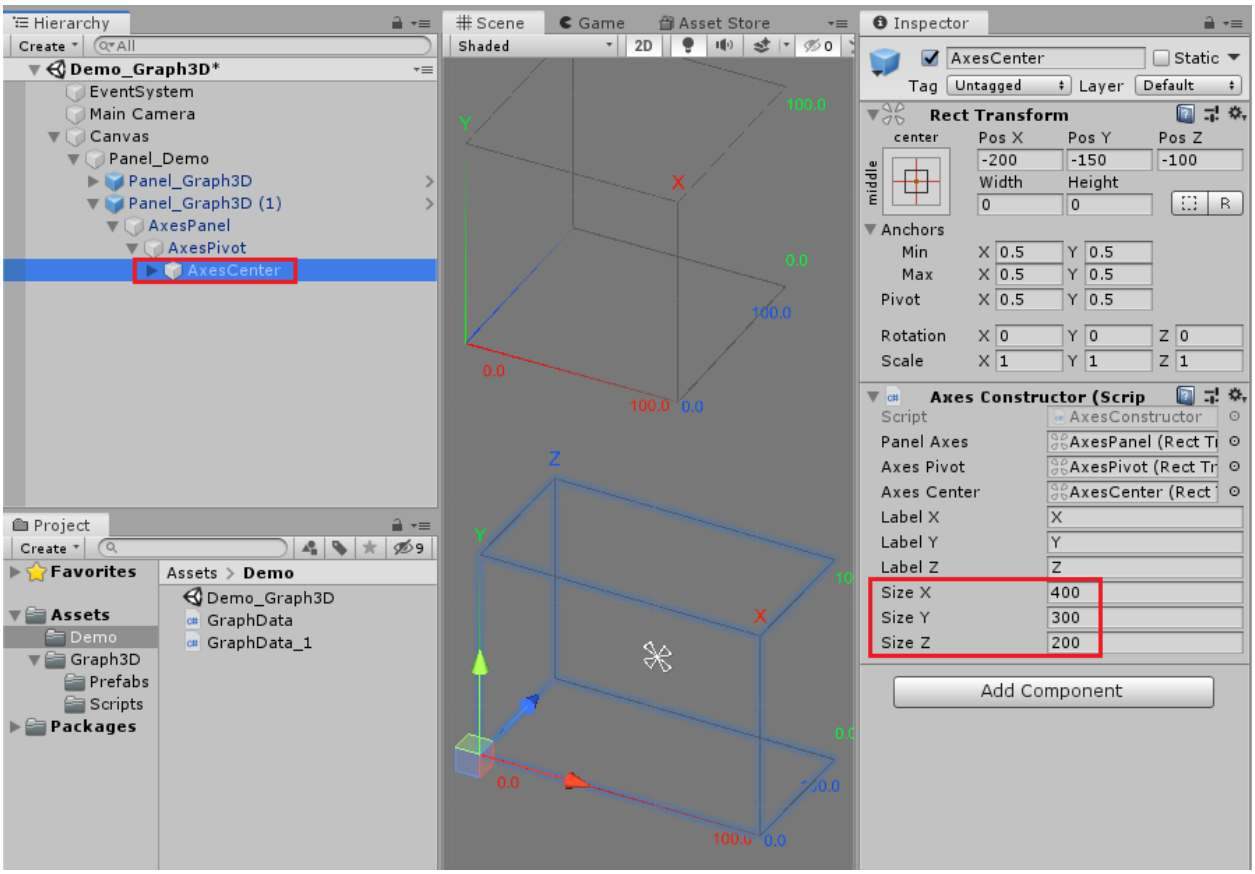
2.4.2. Additional methods for merker:

```
//Destroy the merker from the list and remove its object
   Merker.Destroy(string name)

//Destroy all merkers from the list and delete its objects
   Merker.DestroyAll ()

//Assign activity to curves of the functions passing through the marker
   Merker.SetCurvesActive(string name, bool arg1 = true, bool arg2 = true)

//Assign activity to auxiliary lines along the axes
  Merker.SetAxeLinesActive(string name, bool func = true, bool arg1 = true , bool arg2 = true)

//Assign activity to merker value
    Merker.SetValuesActive(string name, bool func = true, bool arg1 = true, bool arg2 = true)

//Assign color to auxiliary lines of the merker
   Merker.SetLinesColor(string name, Color funcLine, Color arg1Line, Color arg2Line)

//Assign color to merker values
   Merker.SetValuesColor(string name, Color funcValue, Color arg1Value, Color arg2Value)

//Assign the color to the mutually perpendicular curves of the Mercker function
   Merker.SetCurvesColor(string name, Color func_arg1, Color func_arg2)
```

3. Editing the graph panel in the editor inspector

3.1. Set the dimensions of the coordinate axes of the graph panel

AxesCenter - an object of the center of the graph coordinates, contains the XYZ axes, here you can set the dimensions of the axes, see the figure below. The AxesConstructor script is responsible for creating the coordinates.



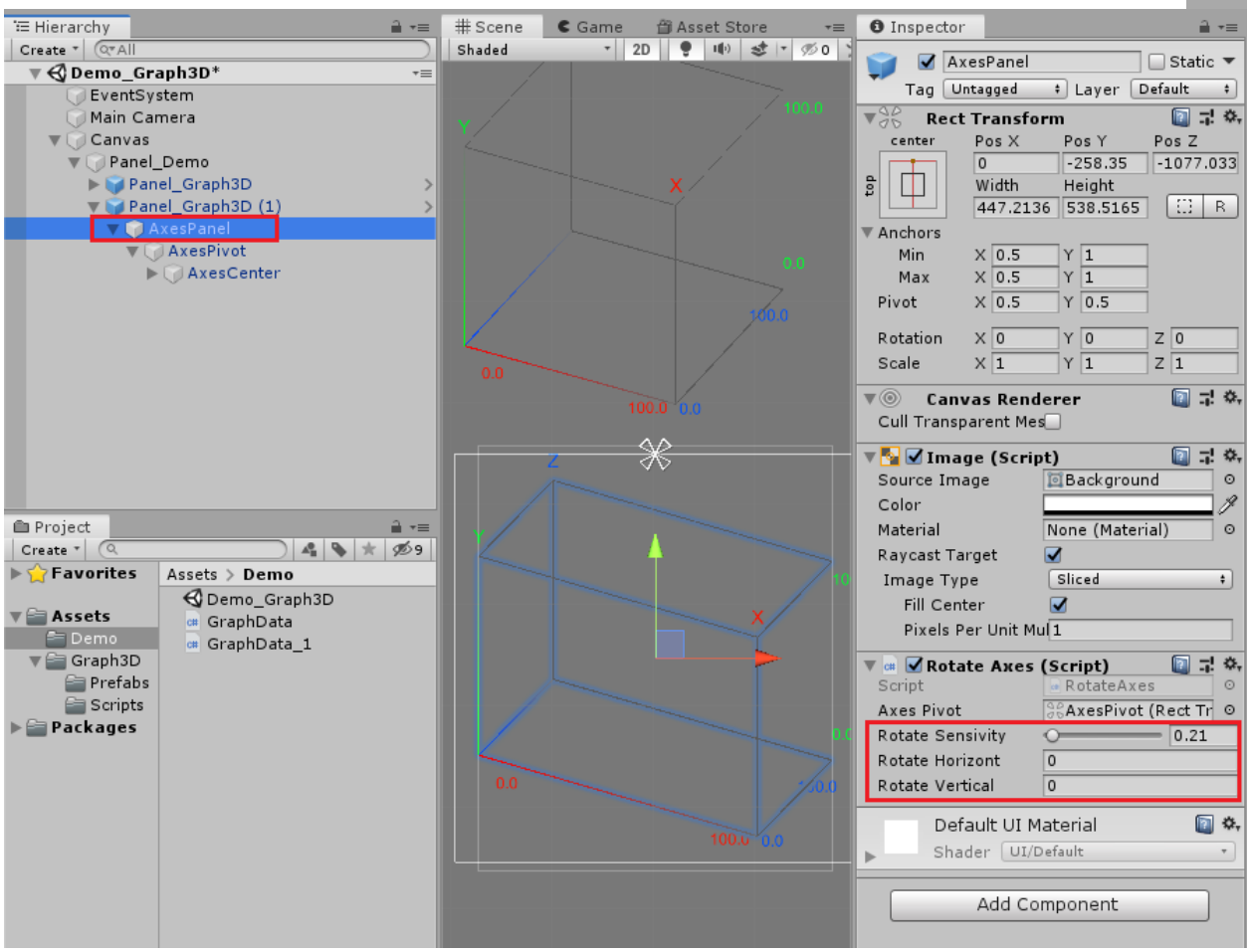
3.2. Set the rotation angles of the coordinate axes

AxesPanel determines the area to touch when rotating in runtime

3.2.1. Define angles using a script

The angles are set relative to the center of the AxesPivot coordinate axis shape and are located in the AxesPanel object as variables of the RotateAxes script (see the figure below).

After starting the application, the graph will occupy the initial position, which you set in the editor, but it will be possible to rotate it on the screen.

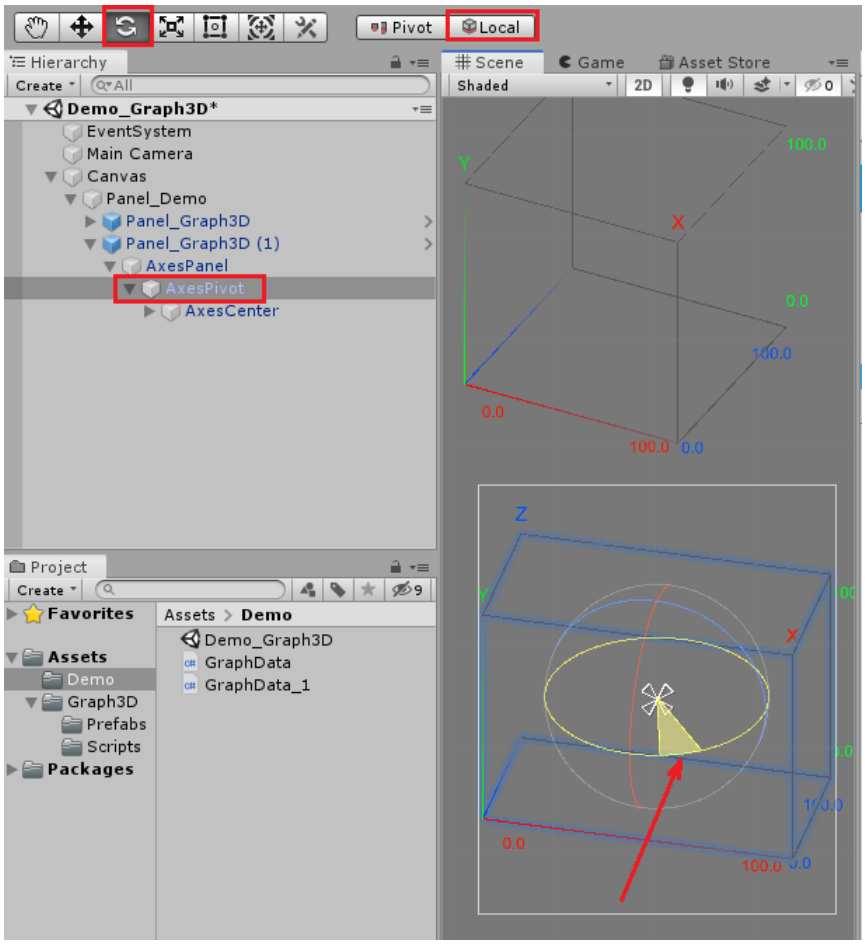


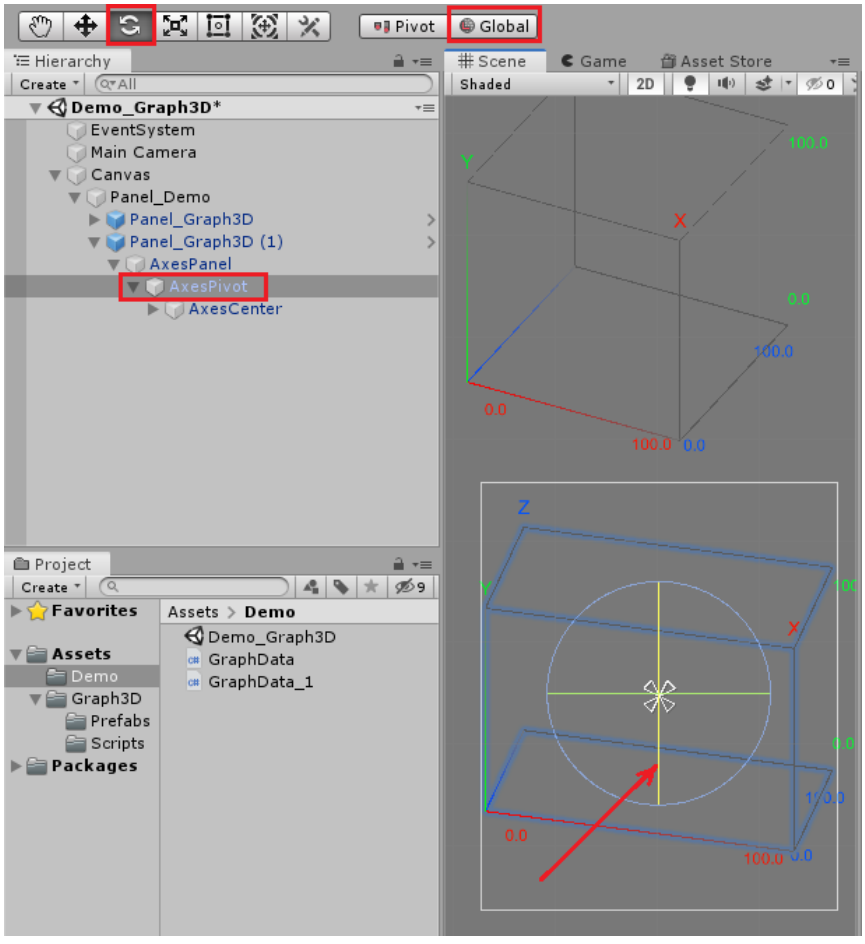
3.2.2. Define angles directly using editor tools

To make the turns correctly, follow the rules strictly:

- To rotate **horizontally**, select everything that is necessary in the figure below and do it in the local system - **Local**.

- To rotate **vertically**, select everything you need in the figure below and do it in the global system - **Global**.

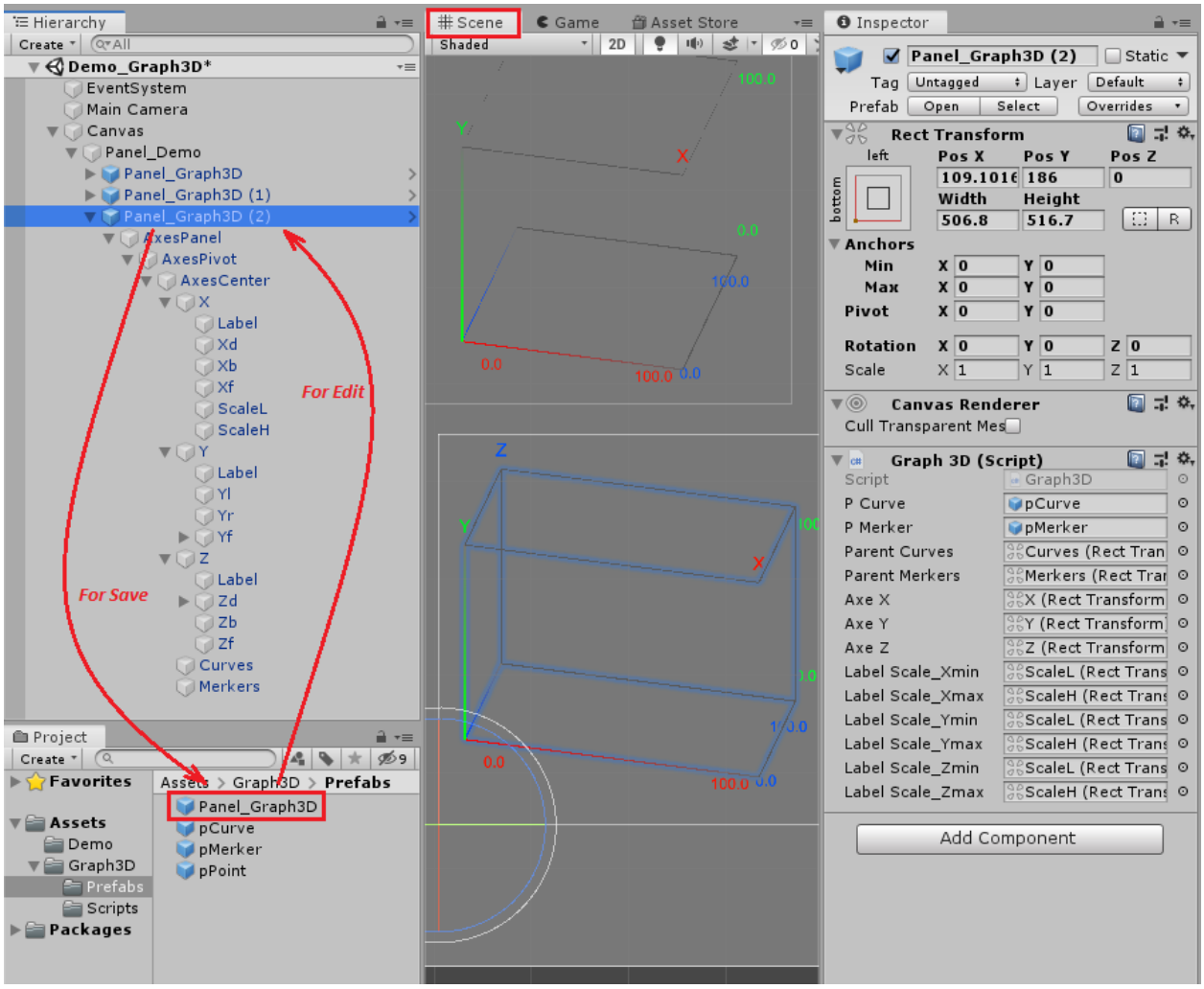

Pivot
Local
Hierarchy
Create
All
Demo_Graph3D*
EventSystem
Main Camera
Canvas
Panel_Demo
Panel_Graph3D
Panel_Graph3D (1)
AxesPanel
AxesPivot
AxesCenter
Scene
Game
Asset Store
Shaded
2D
Project
Favorites
Assets
Demo
Graph3D
Prefabs
Scripts
Packages
Assets > Demo
Demo_Graph3D
GraphData
GraphData_1


Pivot
Global
Hierarchy
Create
All
Demo_Graph3D*
EventSystem
Main Camera
Canvas
Panel_Demo
Panel_Graph3D
Panel_Graph3D (1)
AxesPanel
AxesPivot
AxesCenter
Scene
Game
Asset Store
Shaded
2D
Project
Favorites
Assets
Demo
Graph3D
Prefabs
Scripts
Packages
Assets > Demo
Demo_Graph3D
GraphData
GraphData_1

4. Editing prefabs Panel_Graph3D and pMerker

Most likely you will want to change these prefabs at your discretion.

If you try to open the prefab itself for editing in the Scene window, it will probably not be rendered as it should. Therefore, it is better to edit prefabs as objects. At the end of editing, the object is dragged back to the prefab, thereby saving the new prefab.



RU

Оглавление

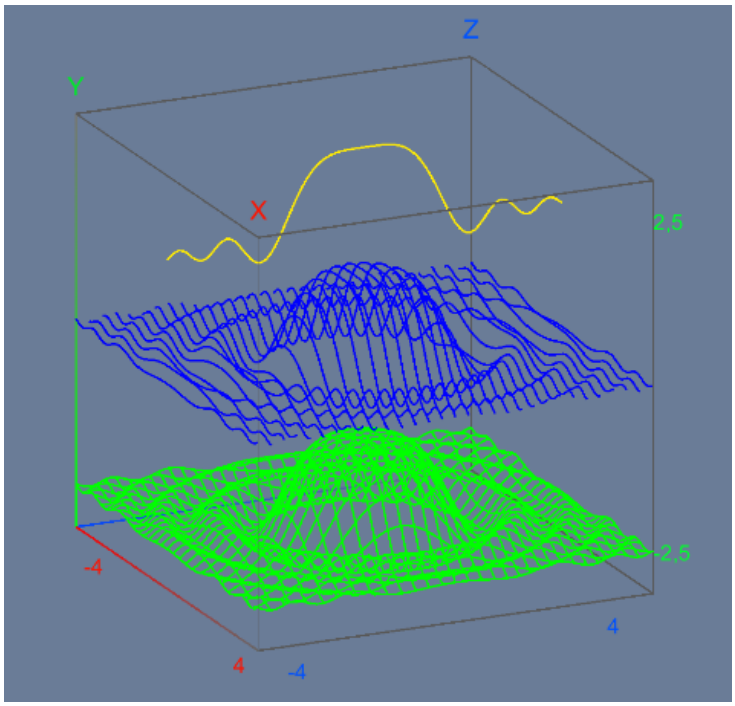
RU

Введение

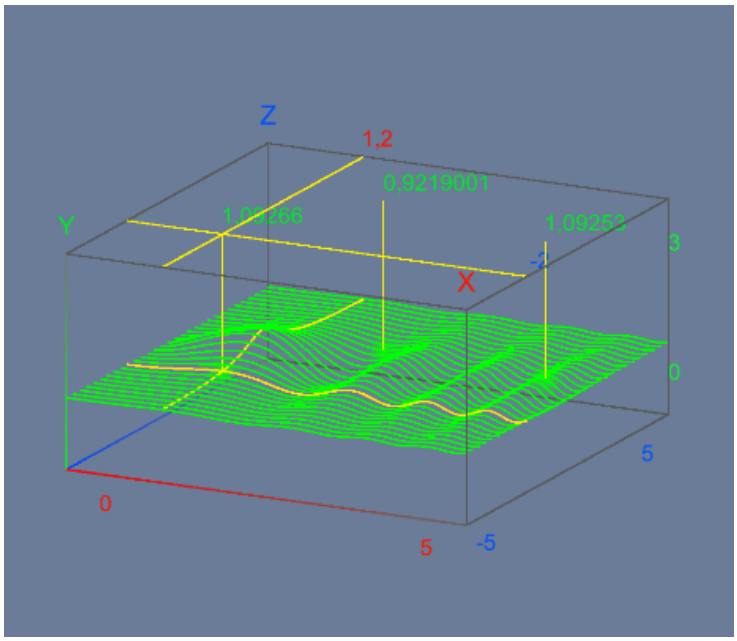
Asset Graph3D позволяет разместить на Canvas желаемое количество панелей (Panel_Graph3D) для построения 3D графиков.

Panel_Graph3D представляет собой prefab, который собран для наиболее удобного редактирования и создания панелей по своему вкусу.

Каждая панель Panel_Graph3D может включать более одного графика ввиде: кривой (Curve), массива параллельных кривых (Curves) и плоскости - сетки (Surface).

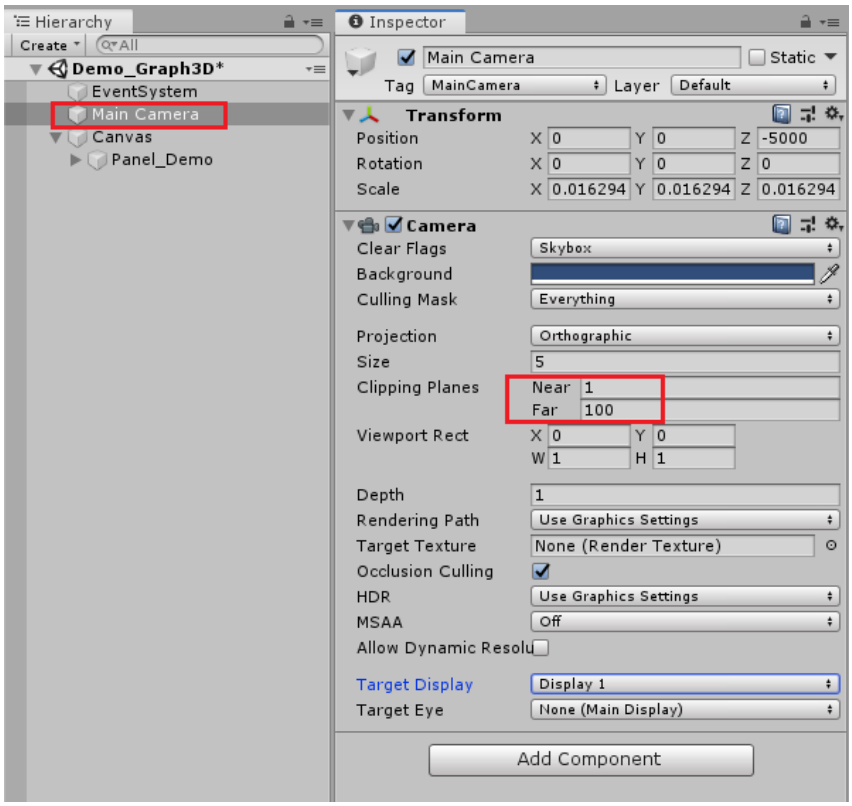


На каждый график можно вывести несколько маркеров-точек со значениями функции и значениями аргументов. Вид маркера может быть настроен через скрипт, а также редактированием prefab pMerker.



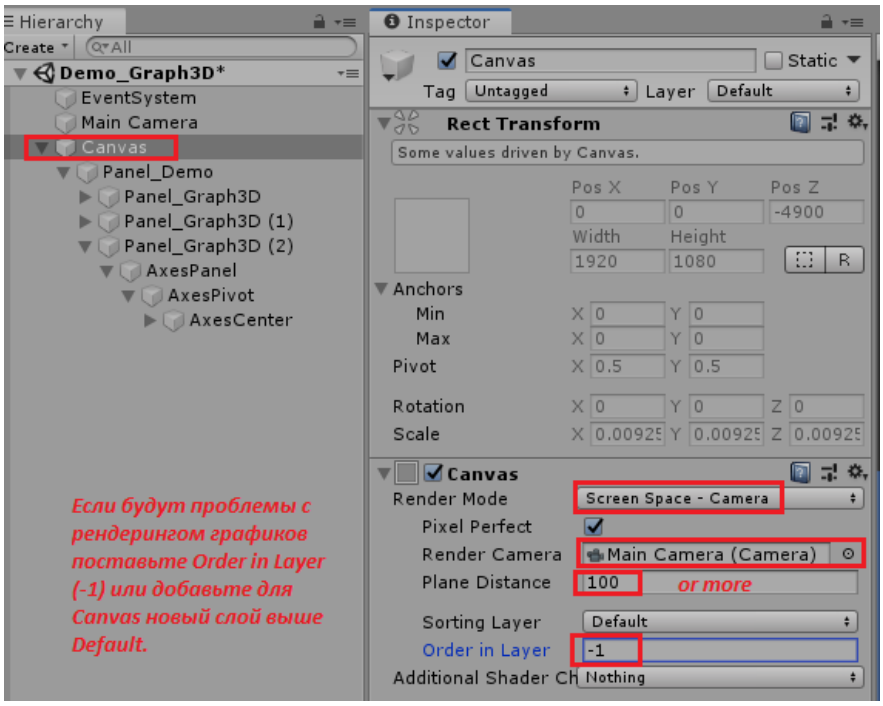
1. Настройка проекта

1.1. Настройка камеры

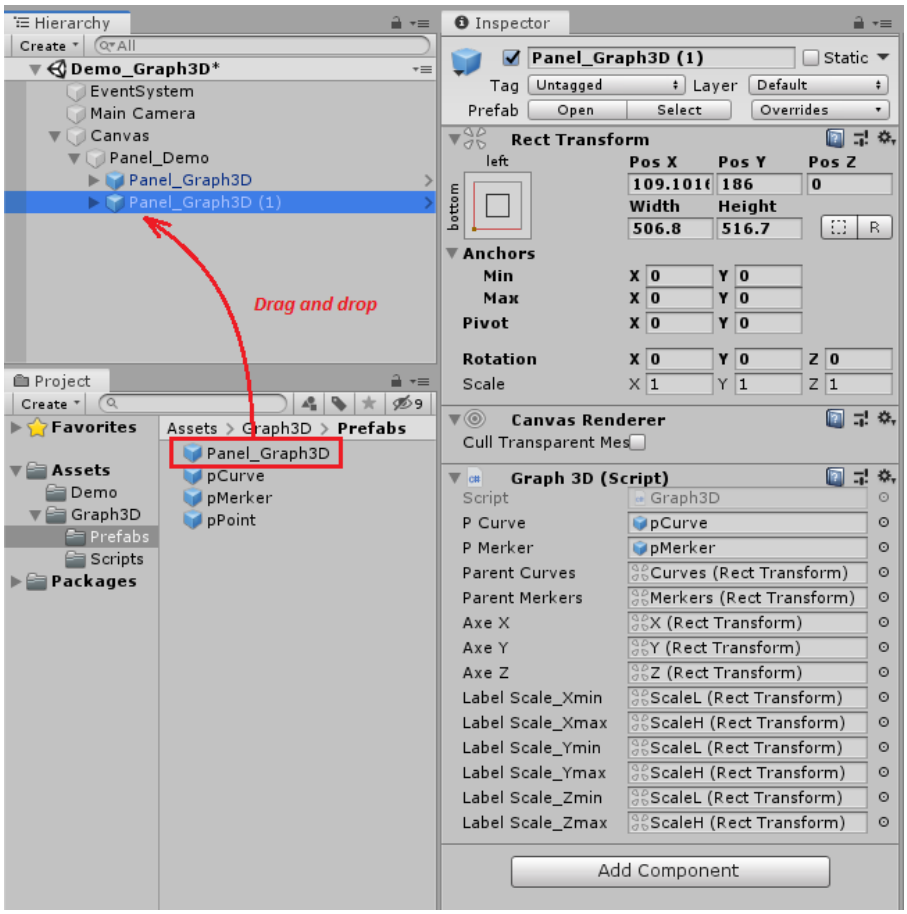


Если изображение панели графика отсутствует или выглядит обрезанным – задайте подходящие параметры расстояния видимости камеры.

1.2. Настройка Canvas

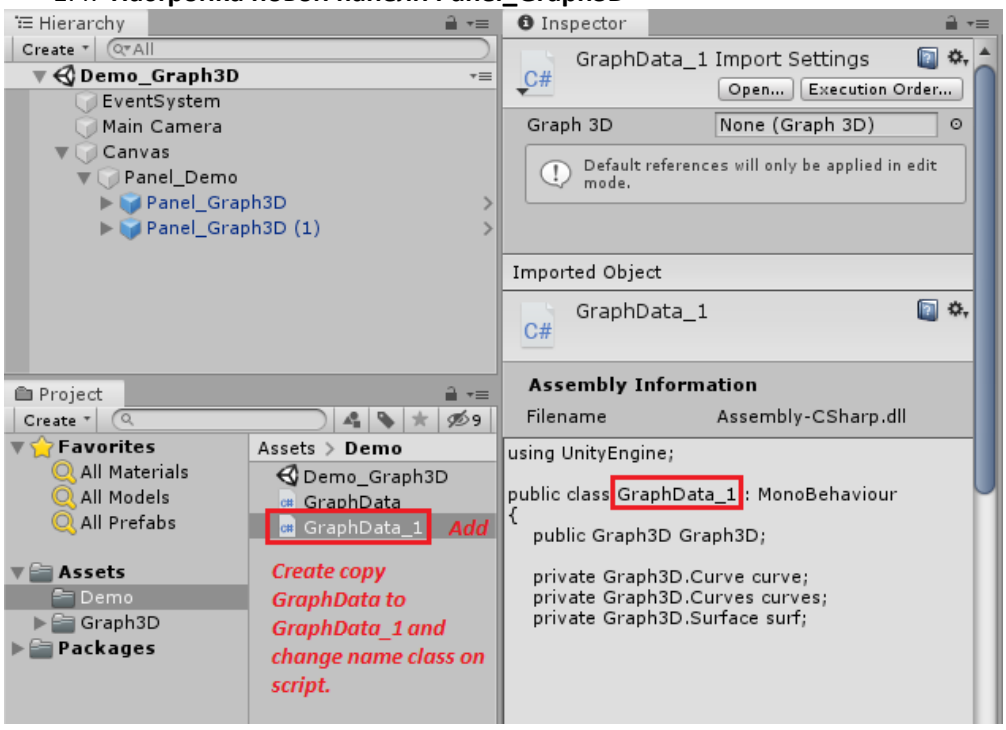


1.3. Добавление панели для графика(ов) Panel_Graph3D



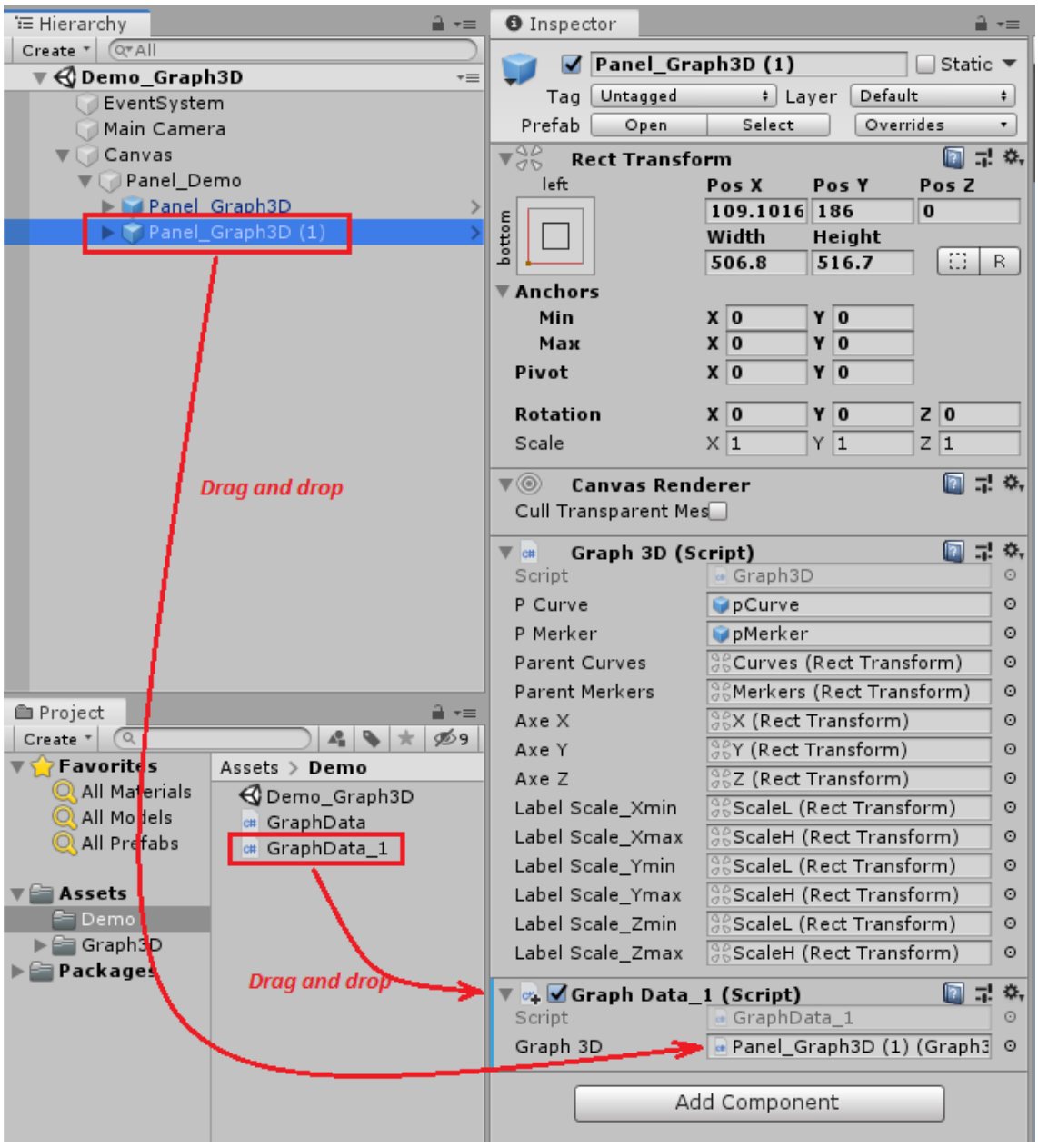
Для добавления новой панели - перетаскиваем Prefab Panel_Graph3D на желаемое место в иерархии проекта.

1.4. Настройка новой панели Panel_Graph3D



Для новой панели Panel_Graph3D необходимо создать новый скрипт для параметров графика(ов) этой панели.

- Сделайте копию существующего скрипта GraphData (например в диспетчере файлов вашего проекта).
- Задайте любое имя новому скрипту (например GraphData_1)
- Откройте новый скрипт (GraphData_1) и измените имя класса на имя скрипта (в примере GraphData_1)
- Привяжите скрипт (GraphData_1) параметров графика(ов) к новой панели Panel_Graph3D(1) как на рисунке ниже.



- Не забудьте привязать сам объект панели (Panel_Graph3D(1)) в поле скрипта.

Теперь можно приступать к созданию графика(ов) на панели Panel_Graph3D !

2. Создание графика(ов) на панели Panel_Graph3D

Для этого необходимо заполнить скрипт данных (GraphData_1) ориентируясь на шаблон.

2.1. Создаем объект или объекты графика из типов: Graph3D.Curve, Graph3D.Curves, Graph3D.Surface.

```
public class GraphData_1 : MonoBehaviour
{
   public Graph3D Graph3D;      //Ссылка на панель отображения графиков

   public Graph3D.Curve curve;          //Одна кривая
   public Graph3D.Curves curves;        //Массив кривых
   public Graph3D.Surface surf;         //Поверхность в виде сетки кривых

private void Awake()
   {
      curve = new Graph3D.Curve(Graph3D);
      curves = new Graph3D.Curves(Graph3D);
      surf = new Graph3D.Surface(Graph3D);
                  . . .
```

2.2. Заполняем структуру данных объектов

Структура данных графика по умолчанию максимально заполнена при создании объекта. Необходимо заполнить только необходимые параметры для каждого из типа графиков. Описание данных можно посмотреть в скрипте Graph3D.cs

2.2.1. Необходимые параметры для Graph3D.Curve

```
private void Awake()
   {
                                    . . .
//Задаем тип графика - Curve  с постоянным параметром на оси X, Y или Z
//Возможные варианты: Curve_Xconst, Curve_Yconst, Curve_Zconst
      curve.data.typeGraph = Graph3D.TypeGraph.Curve_Xconst; //Как пример выбрана ось X

//Выбираем расположение функции и параметров по осям панели графика
//Возможные варианты: Y_XZ,  Y_ZX,  X_YZ,  X_ZY,  Z_XY,  Z_YX
      curve.data.typeFunc = Graph3D.TypeFunc.Z_YX;
//Как пример – функция по оси Z с параметром 1 на оси Y и параметром 2 на оси X

//Присваиваем функцию по заданному делегату float a (float b; float c)
      curve.data.func = funcABC;
//Ссылка на функцию с 2-мя параметрами
// private float funcABC ( float b, float c )         //  Z_YX :     Z - function, Y – “b”,  X – “c”
// {
//    return b * b + c * c;    //return function
// }
```

```
//Если функция с одним параметром, то другой должен быть = 0, например
// private float funcABC(0.0f , float c)
// {
//     return  c * c;
// }

//Задаем шкалу в физических единицах для оси параметра 1 (в примере Y)
      curve.data.X.scale.L = -20.0f;          //Ближняя к центру  координат
      curve.data.X.scale.H = 20.0f;           //Дальняя от центра координат
//Условия: H > L or H < L;   H !== L

//Задаем шкалу в физических единицах для оси параметра 2 (в примере X)
      curve.data.Y.scale.L = -20.0f;          //Ближняя к центру  координат
      curve.data.Y.scale.H = 20.0f;           //Дальняя от центра координат
//Условия: H > L or H < L;   H !== L

//Задаем шкалу в физических единицах для оси функции (в примере Z)
      curve.data.Z.scale.L = 100.0f;
      curve.data.Z.scale.H = 1000.0f;

//Задаем количество сегментов кривой (гладкость) вдоль оси (в примере Y)
      curve.data.Y.segments = 20;

//Задаем значение постоянного параметра (в примере X)
      curve.data.X.argConst = 5.0f;           // Graph3D.TypeGraph.Curve_Xconst


//Параметры которые не будут использоваться (для данного примера):
    //curve.data.X.segments
    //curve.data.Z.segments
    //curve.data.Y.argConst
    //curve.data.Z.argConst
```

2.2.2. Дополнительные параметры для Graph3D.Curve

```
private void Awake()
   {
                              . . .
//Автомасштаб по оси функции (в примере Z).
      curve.data.autoScaleAxeFunc = true;          //def = false
//Флаг потребует двойного перерасчета  функции  (затратно по времени примерно в 2 раза)
//Применение флага все равно требует задания максимальных пределов scale.L и scale.H

//Шкала для осей панели Graph3D в единицах экрана (пикселях)
curve.data.X.scaleAxe.LScreen
curve.data.X.scaleAxe.HScreen
curve.data.Y.scaleAxe.LScreen
curve.data.Y.scaleAxe.HScreen
curve.data.Z.scaleAxe.LScreen
curve.data.Z.scaleAxe.HScreen
//Данные параметры можно выставить в скрипте, но предусмотрено это делать интерактивно в
//инспекторе редактора (см. п. 3.1)
```

2.2.3. Необходимые параметры для Graph3D.Curves

```
private void Awake()
   {
                                        . . .
//Задаем тип графика - Curves  с постоянными параметрами на оси X, Y или Z
//Возможные варианты: XCurves, YCurves, ZCurves
      curves.data.typeGraph = Graph3D.TypeGraph.ZCurves;   //Как пример выбрана ось Z

//Выбираем расположение функции и параметров по осям панели графика
//Возможные варианты: Y_XZ,  Y_ZX,  X_YZ,  X_ZY,  Z_XY,  Z_YX
      curves.data.typeFunc = Graph3D.TypeFunc.Y_XZ;
//Как пример – функция по оси Y с параметром 1 на оси X и параметром 2 на оси Z

//Присваиваем функцию по заданному делегату float a (float b; float c)
      curves.data.func = funcABC;
//Ссылка на функцию с 2-мя параметрами
// private float funcABC ( float b, float c )         //  Y_XZ :      Y - function, X – “b”,  Z – “c”
// {
//    return b * b + c * c;    //return function
// }

//Если функция с одним параметром, то другой должен быть = 0, например
// private float funcABC(0.0f , float c)
// {
//    return  c * c;
// }

//Задаем шкалу в физических единицах для оси параметра 1 (в примере X)
      curves.data.X.scale.L = -20.0f;         //Ближняя к центру  координат
      curves.data.X.scale.H = 20.0f;          //Дальняя от центра координат
//Условия: H > L or H < L;   H !== L

//Задаем шкалу в физических единицах для оси параметра 2 (в примере Z)
      curves.data.Z.scale.L = -20.0f;         //Ближняя к центру  координат
      curves.data.Z.scale.H = 20.0f;          //Дальняя от центра координат
//Условия: H > L or H < L;   H !== L

//Задаем шкалу в физических единицах для оси функции (в примере Y)
      curves.data.Y.scale.L = 100.0f;
      curves.data.Y.scale.H = 1000.0f;

//Задаем количество сегментов кривой (гладкость) вдоль оси (в примере X)
      curves.data.X.segments = 20;

//Задаем количество параллельных кривых (в примере Z)
      curves.data.Z.curves = 10;    // Graph3D.TypeGraph.ZCurves

//Параметры которые не будут использоваться (для данного примера):
    // curves.data.Y.segments
    // curves.data.Z.segments
    // curves.data.X.argConst
    // curves.data.Y.argConst
```

```
// curves.data.Z.argConst
// curves.data.X.curves
// curves.data.Y.curves
```

2.2.4. Дополнительные параметры для Graph3D.Curves

```
private void Awake()
   {
                              . . .
//Автомасштаб по оси функции (в примере Z).
      curve.data.autoScaleAxeFunc = true;          //def = false
//Флаг потребует двойного перерасчета  функции  (затратно по времени примерно в 2 раза)
//Применение флага все равно требует задания максимальных пределов scale.L и scale.H

//Шкала для осей панели Graph3D в единицах экрана (пикселях)
curve.data.X.scaleAxe.LScreen
curve.data.X.scaleAxe.HScreen
curve.data.Y.scaleAxe.LScreen
curve.data.Y.scaleAxe.HScreen
curve.data.Z.scaleAxe.LScreen
curve.data.Z.scaleAxe.HScreen
//Данные параметры можно выставить в скрипте, но предусмотрено это делать интерактивно в
//инспекторе редактора (см. п. 3.1)
```

2.2.5. Необходимые параметры для Graph3D.Surface

```
private void Awake()
   {
                                    . . .
//Задаем тип графика - Surface
//Возможные варианты: в дальней шем будет выбор Grid or Polygons
      surf.data.typeGraph = Graph3D.TypeGraph. Surface;

//Выбираем расположение функции и параметров по осям панели графика
//Возможные варианты: Y_XZ,  Y_ZX,  X_YZ,  X_ZY,  Z_XY,  Z_YX
      surf.data.typeFunc = Graph3D.TypeFunc.X_YZ;
//Как пример – функция по оси X с параметром 1 на оси Y и параметром 2 на оси Z

//Присваиваем функцию по заданному делегату float a (float b; float c)
      surf.data.func = funcABC;
//Ссылка на функцию с 2-мя параметрами
// private float funcABC ( float b, float c )         //  X_YZ :      X - function, Y – “b”,  Z – “c”
// {
//    return b * b + c * c;    //return function
// }

//Если функция с одним параметром, то другой должен быть = 0, например
// private float funcABC(0.0f , float c)
// {
//    return  c * c;
// }
```

```
//Задаем шкалу в физических единицах для оси параметра 1 (в примере Y)
      surf.data.Y.scale.L = -20.0f; //Ближняя к центру  координат
      surf.data.Y.scale.H = 20.0f;  //Дальняя от центра координат
//Условия: H > L or H < L;   H !== L

//Задаем шкалу в физических единицах для оси параметра 2 (в примере Z)
      surf.data.Z.scale.L = -20.0f; //Ближняя к центру  координат
      surf.data.Z.scale.H = 20.0f;              //Дальняя от центра координат
//Условия: H > L or H < L;   H !== L

//Задаем шкалу в физических единицах для оси функции (в примере X)
      surf.data.X.scale.L = 100.0f;
      surf.data.X.scale.H = 1000.0f;

//Задаем количество сегментов кривой вдоль осей параметра 1 и 2 (гладкость)
      surf.data.Y.segments = 20;
      surf.data.Z.segments = 20;

//Задаем количество параллельных кривых для параметра 1 и 2 (в примере Y и X)
      surf.data.Y.curves = 10;
      surf.data.Z.curves = 10;

//Параметры которые не будут использоваться (для данного примера):
    // surf.data.X.segments
    // surf.data.X.argConst
    // surf.data.Y.argConst
    // surf.data.Z.argConst
    // surf.data.X.curves
    // surf.data.Y.curves
    // surf.data.Z.curves
```

2.2.6. Дополнительные параметры для Graph3D.Surface

```
private void Awake()
   {
                              . . .
//Автомасштаб по оси функции (в примере Z).
      surf.data.autoScaleAxeFunc = true; //def = false
//Флаг потребует двойного перерасчета  функции  (затратно по времени примерно в 2 раза)
//Применение флага все равно требует задания максимальных пределов scale.L и scale.H

//Шкала для осей панели Graph3D в единицах экрана (пикселях)
surf.data.X.scaleAxe.LScreen
surf.data.X.scaleAxe.HScreen
surf.data.Y.scaleAxe.LScreen
surf.data.Y.scaleAxe.HScreen
surf.data.Z.scaleAxe.LScreen
surf.data.Z.scaleAxe.HScreen
//Данные параметры можно выставить в скрипте, но предусмотрено это делать интерактивно в
//инспекторе редактора (см. п. 3.1)
```

2.3. Делаем расчет графиков

На данном этапе методом DataSet() по заполненным выше данным графики сделают расчет в памяти без вывода на экран. Данный метод будет полезен для подготовки графиков например через корутины.

```
private void OnEnable()
  {
          carve.DataSet();
          carves.DataSet();
          surf.DataSet();
                                . . .
```

2.4. Вывод графиков на экран

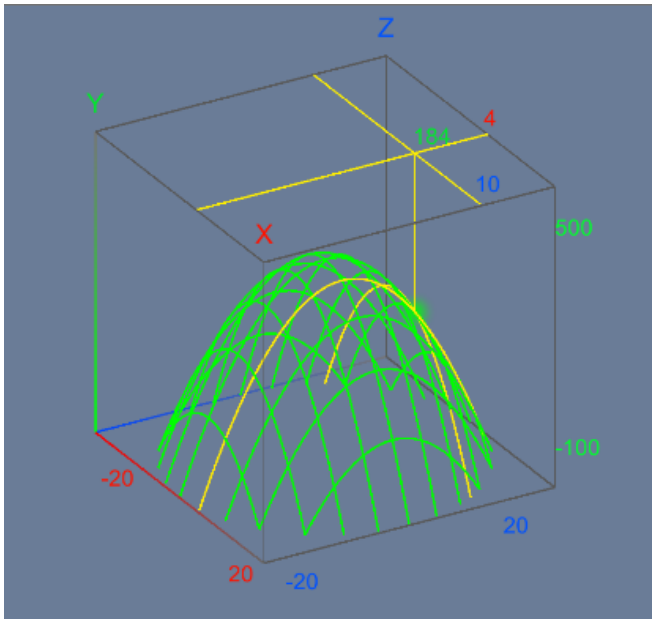
Метод Show(Color color) выводит график на экран при условии что был сделан метод DataSet(). Цвет кривых графика задается параметром.

```
private void OnEnable()
  {
                . . .
        carve.Show(Color.yellow);
         carves.Show(Color.blue);
         surf.Show(Color.green);
```

2.5. Меркеры на графике

2.5.1. Вывод меркера на экран

Меркеров на графике можно выставить несколько**.**
Меркер представляет собой точку функции на графике с определенными координатами, которые задаются через параметры функции.



- Для создания меркера используйте метод:
  Merker.Set(string name, float arg1, float arg2),

где name – имя меркера, arg1, arg2 – параметры точки для функции графика func(float b, float c), arg1 – “b”, arg2 – “c”, dec - количество знаков после запятой.

- Для настройки меркера используйте методы п 2.5.2
- Для вывода на экран испльзуйте метод:
  Merker.Show(string name, bool active),
  где name – имя меркера, active – видимость.

В нашем примере:

```
private void OnEnable()
  {
                . . .
        carve.Merker.Set(“m1”, 10.0f, 15.0f);
        carves.Merker.Set(“m1”, 10.0f, 15.0f);
        surf.Merker.Set(“m2”, 10.0f, 15.0f);
        carve.Merker.Show(“m1”, true);
        carves.Merker.Show(“m1”, true);
        surf.Merker.Show(“m2”, true);
```

2.5.2. Дополнительные методы для меркера:

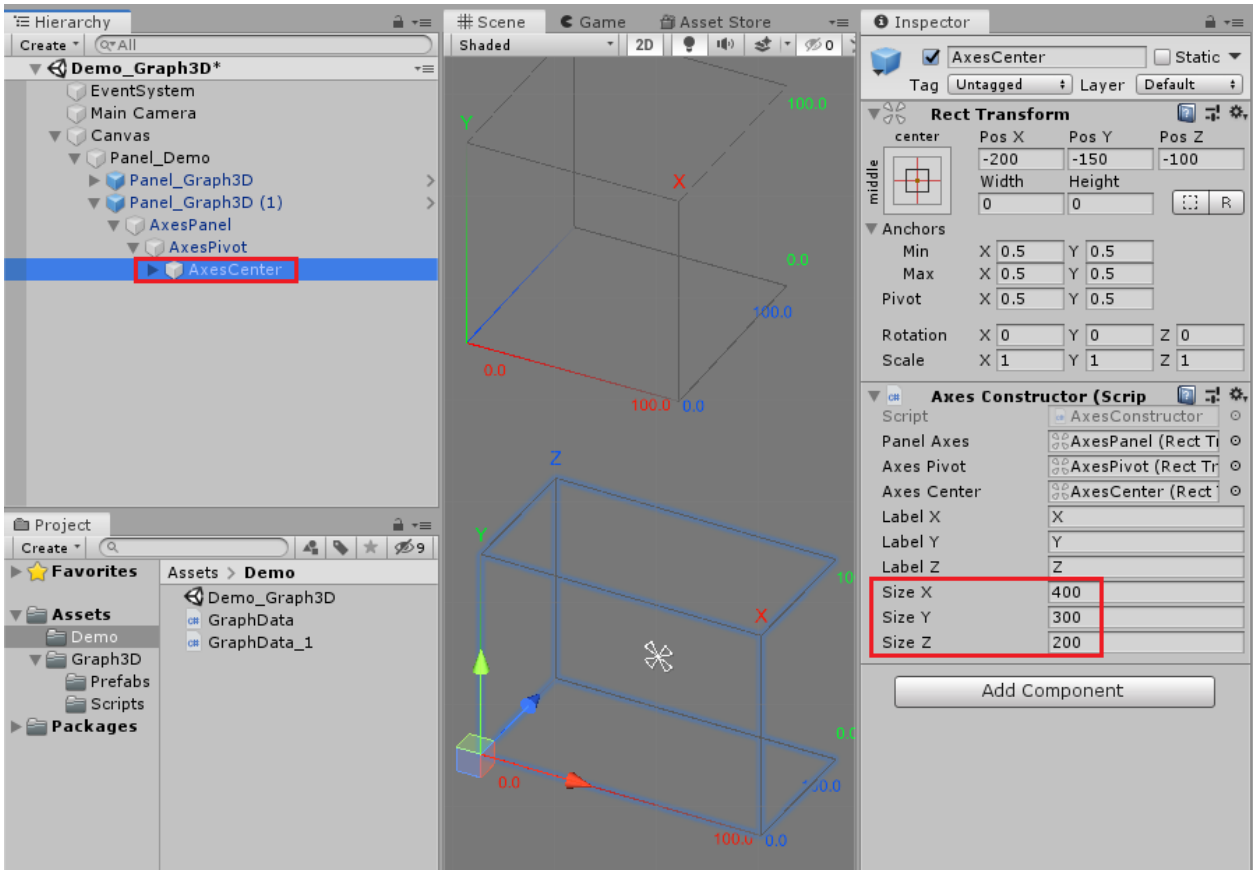
```
//Уничтожить меркер из списка и удалить его объект
   Merker.Destroy(string name)

// Уничтожить все меркеры из списка и удаленить его объекты
  DestroyAll()

//Назначить кривым функции, проходящих через меркер видимость
   Merker.SetCurvesActive(string name, bool arg1 = true, bool arg2 = true)

//Назначить вспомогательным линиям по осям видимость
   Merker.SetAxeLinesActive(string name, bool func = true, bool arg1 = true , bool arg2 = true)

//Назначить вывод значений меркера видимость
    Merker.SetValuesActive(string name, bool func = true, bool arg1 = true, bool arg2 = true)

//Назначить цвет вспомогательным линиям меркера
   Merker.SetLinesColor(string name, Color funcLine, Color arg1Line, Color arg2Line)

//Назначить цвет значениям меркера
   Merker.SetValuesColor(string name, Color funcValue, Color arg1Value, Color arg2Value)

//Назначить цвет взаимоперпендикулярным кривым функции меркера
   Merker.SetCurvesColor(string name, Color func_arg1, Color func_arg2)
```

3. Редактирование панели графика в инспекторе редактора

3.1. Задаем размеры координатных осей панели графика

AxesCenter – объект центра координат графика, содержит оси XYZ, здесь можно выставить размеры осей, смотри рисунок ниже.

Скрипт AxesConstructor отвечает за построение координат.



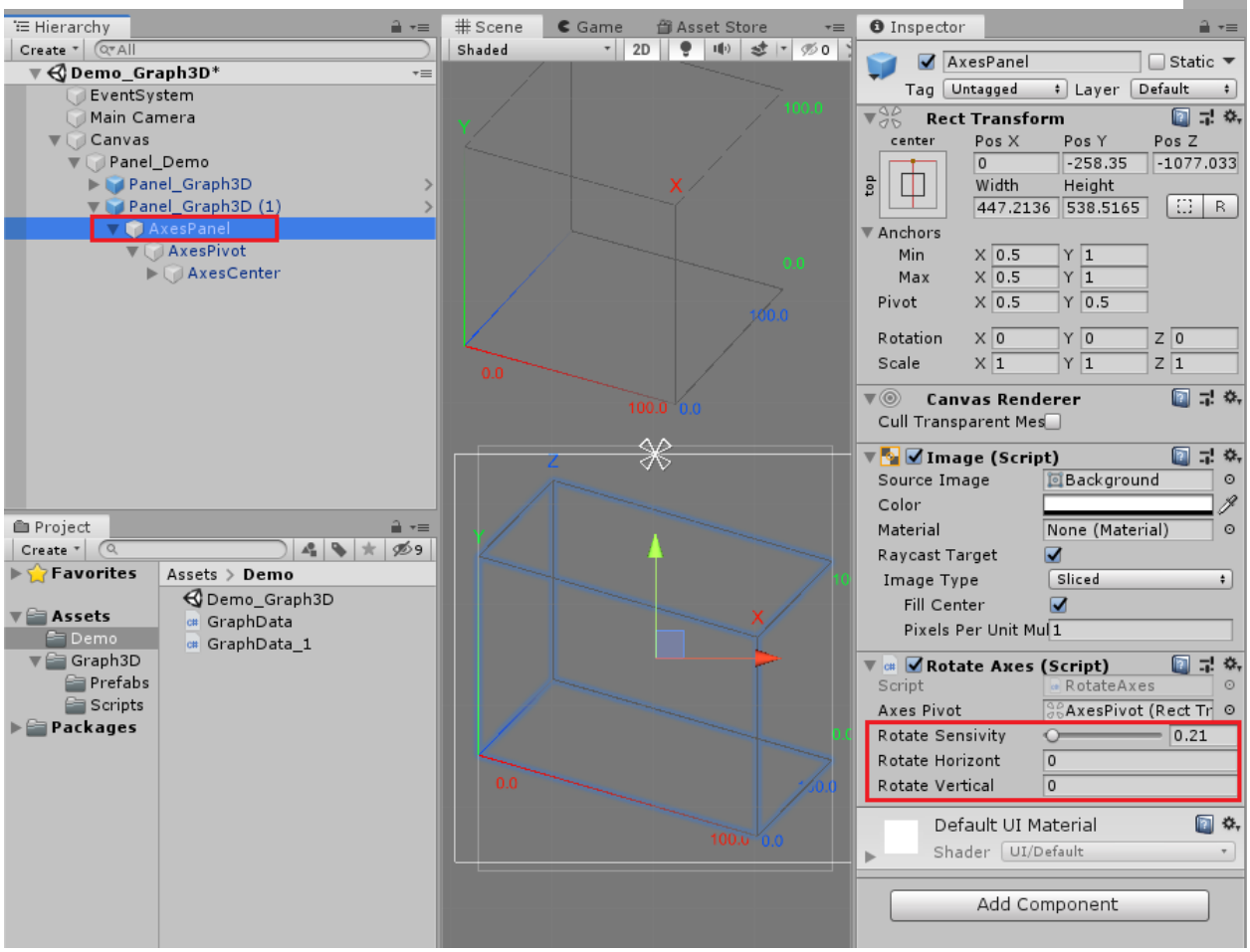
3.2. Задаем углы поворота координатных осей

AxesPanel определяет площадь для касания при вращении в runtime.

3.2.1. Задаем углы с помощью скрипта

Углы задаются относительно центра фигуры координатных осей AxesPivot и находятся в объекте AxesPanel как переменные скрипта RotateAxes (смотри рисунок ниже).

После запуска приложения график займет начальное положение, которое вы выставите в редакторе, но при этом будет возможность вращать его на экране.

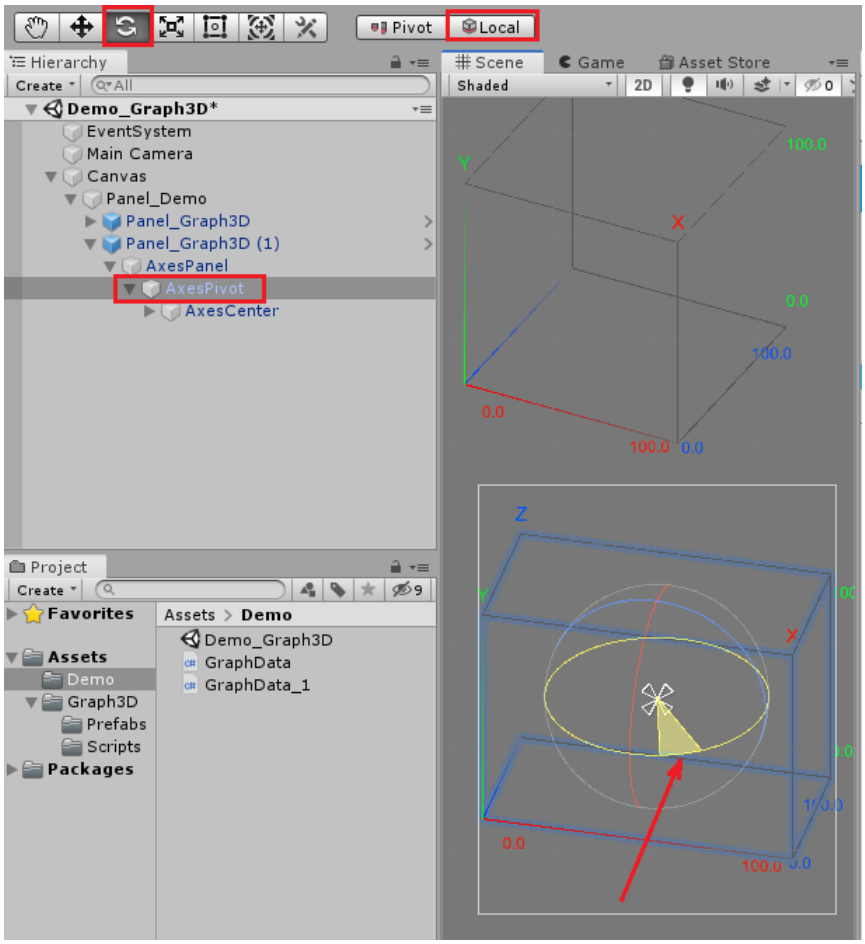


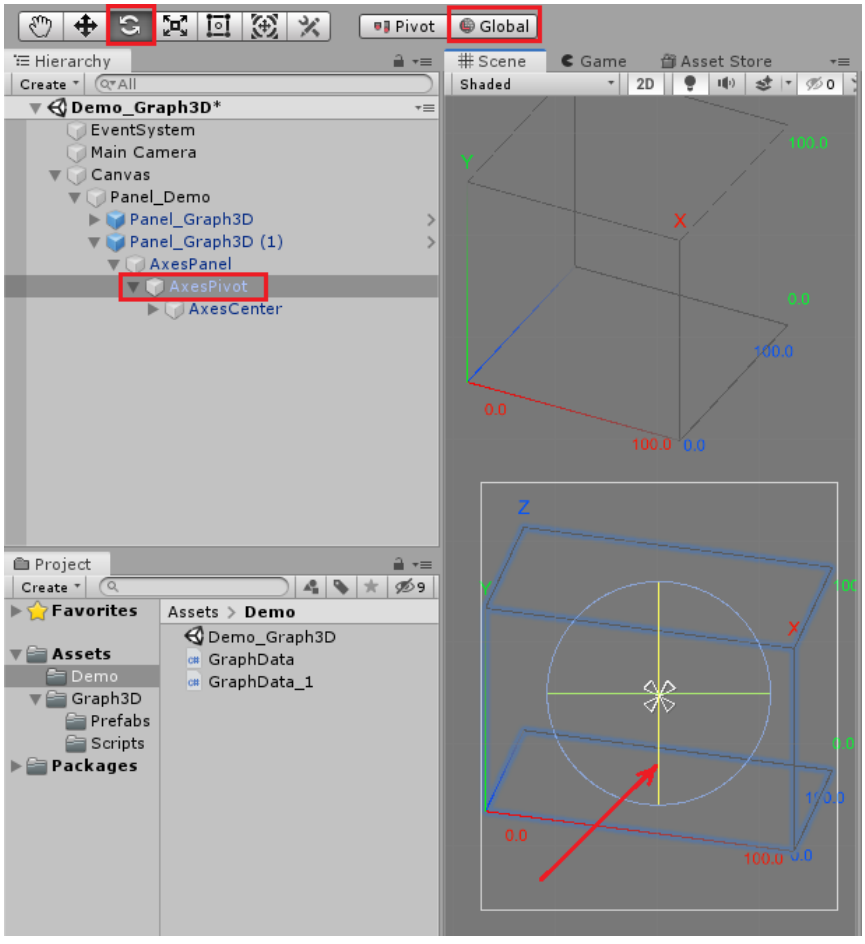
3.2.2. Задаем углы напрямую с помощью инструментов редактора

Чтобы правильно сделать повороты - выполните строго по правилам:

- Для поворота по **горизонтали** выберете все что необходимо на рисунке ниже и делайте это в локальной системе – **Local**.

- Для поворота по **вертикали** выберете все что необходимо на рисунке ниже и делайте это в глобальной системе – **Global**.


Pivot
Local
Hierarchy
Create
All
Demo_Graph3D*
EventSystem
Main Camera
Canvas
Panel_Demo
Panel_Graph3D
Panel_Graph3D (1)
AxesPanel
AxesPivot
AxesCenter
Scene
Game
Asset Store
Shaded
2D
Project
Favorites
Assets
Demo
Graph3D
Prefabs
Scripts
Packages
Assets > Demo
Demo_Graph3D
GraphData
GraphData_1


Pivot
Global
Hierarchy
Create
All
Demo_Graph3D*
EventSystem
Main Camera
Canvas
Panel_Demo
Panel_Graph3D
Panel_Graph3D (1)
AxesPanel
AxesPivot
AxesCenter
Scene
Game
Asset Store
Shaded
2D
Project
Favorites
Assets
Demo
Graph3D
Prefabs
Scripts
Packages
Assets > Demo
Demo_Graph3D
GraphData
GraphData_1

RU

4. Редактирование prefabs Panel_Graph3D и pMerker

Скорее всего вы захотите изменить данные префабы по своему усмотрению.

Если попытаться открыть сам префаб для редактирования в окне Scene он возможно будет не визуализирован как положено. Поэтому править префабы лучше как объекты. По окончании редактирования объект перетаскивается обратно на префаб, тем самым сохраняя новый префаб.