Draw Debug Tools Documentation

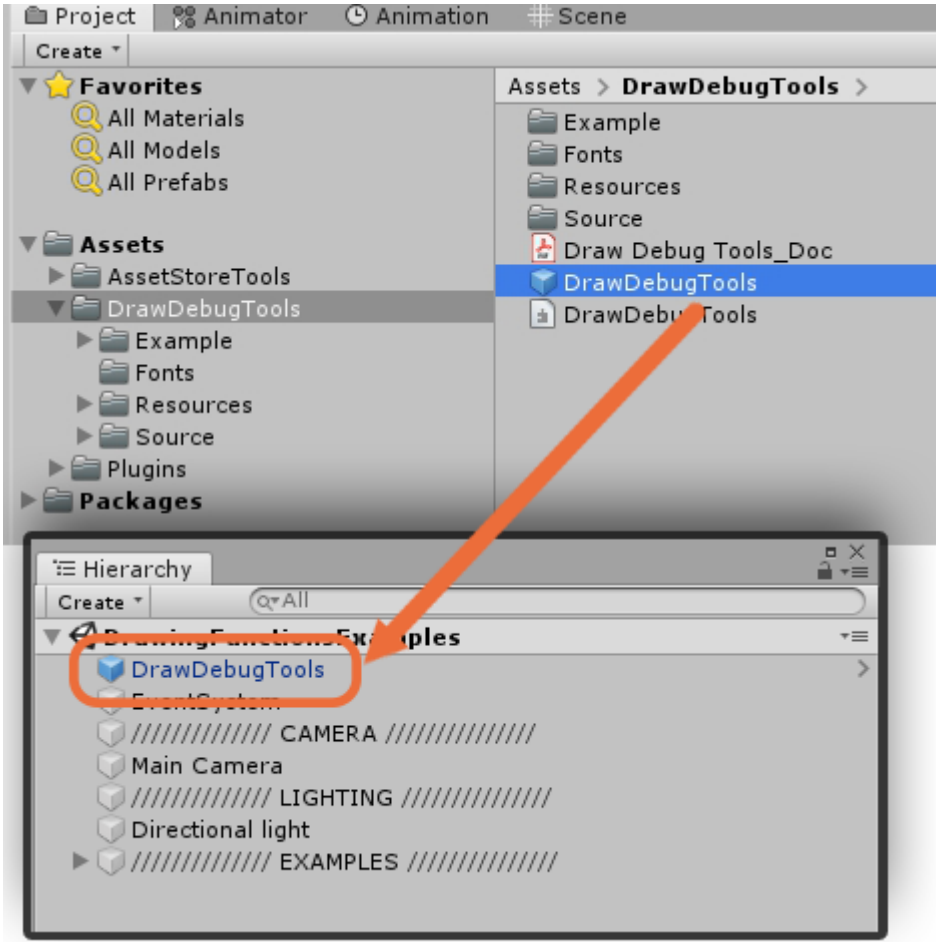
Overview

Draw Debug Tools, help you visualize your game play, by providing many functions to draw shapes and tools to make development fast and enjoyable experience.

Unity Asset Store URL:https://assetstore.unity.com/packages/slug/163706

How To Use

Drag and drop DDT prefab to your scene:



You don't have to put the prefab in every scene, put it only in you main scene, it will not get destroyed when you switch scenes.

Features

- Offers multiple functions to draw different shapes: line, box, sphere, capsule, arrow, cylinder, camera shape... and many many more
- All Shapes are drawn and visible in scene view, game view and the builds.

- Visualize Raycast Hit infos and other datas are coming soon (Collision infos, Rigidbody forces….)
- Offers an on-screen log message function
- Easy to setup and easy to use
- The package includes many examples that demonstrate how to use it
- It has everything you need to visualize gameplay datas
- Support VR
- Optimized to work without impacting your game performance
- Call draw functions from anywhere in your code
- Works on all platforms
- Support all rendering pipelines: HDRP, URP, SRP
- Dedicated support

Debug Camera

Enabling debug camera let you move around your level freely, it also let you control time scale, you can stop time, slow it down, or speed it up.

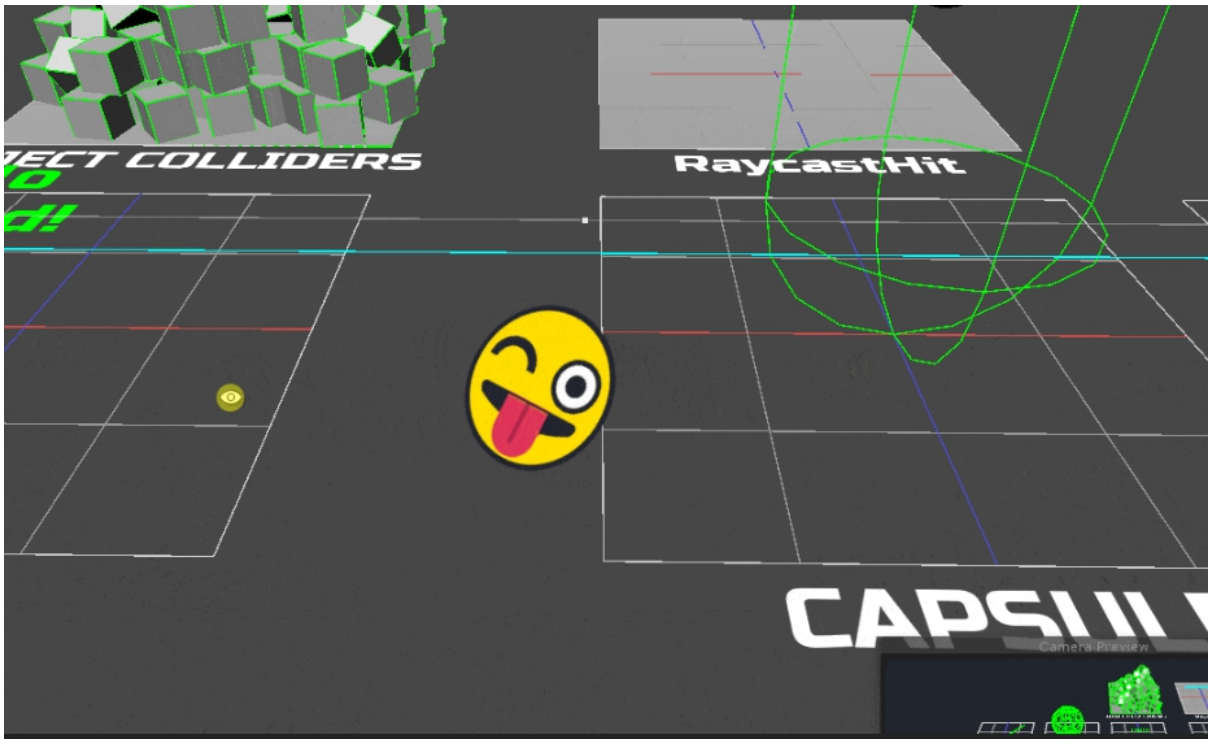
To enable debug camera you have to hit F9 in your keyboard, or touch with 4 fingers if you are running your game on android or IOS.

When debug camera is enabled you can aim at an object to display information like it's name, distance from camera, UVs, list of materials...etc.

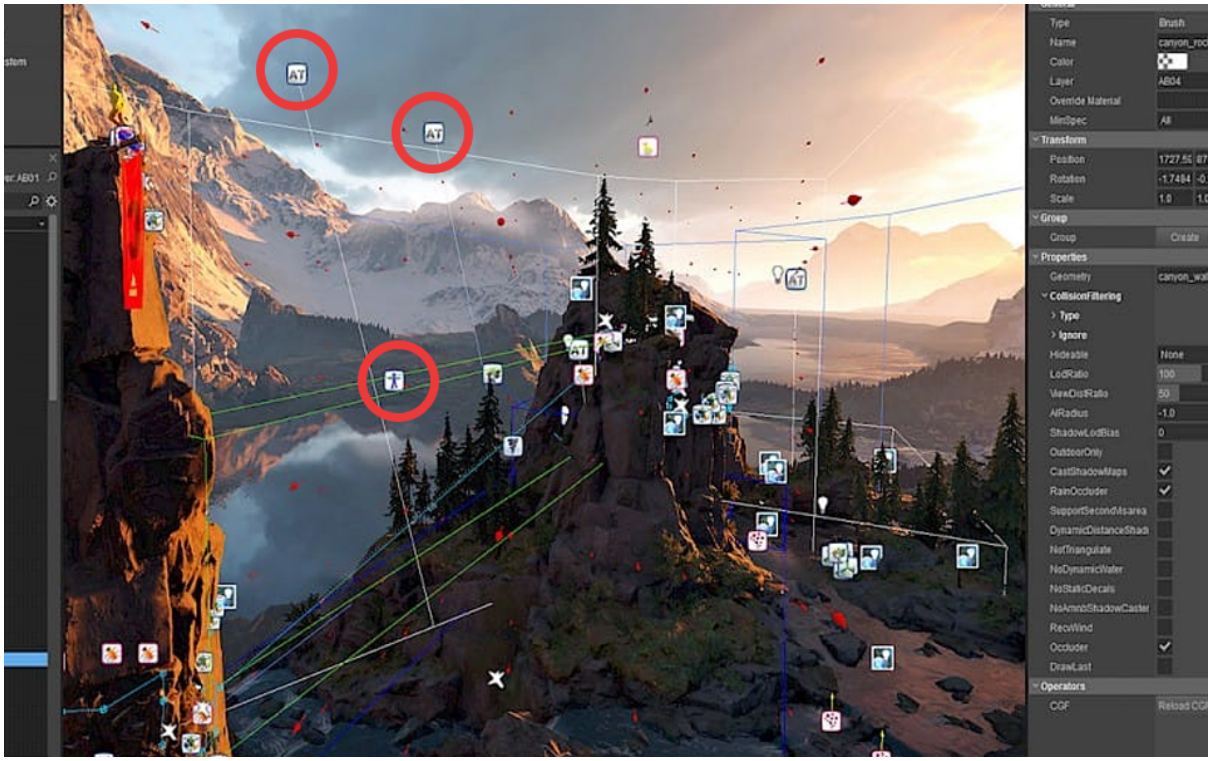
Billboard

Overview

Billboard is a plane with texture mapped onto it. You can make it either always face camera or face a fixed direction.



Billboards are useful when you want to associate an icon or an image to an object or a shape while debugging, like in this example:



How To Use

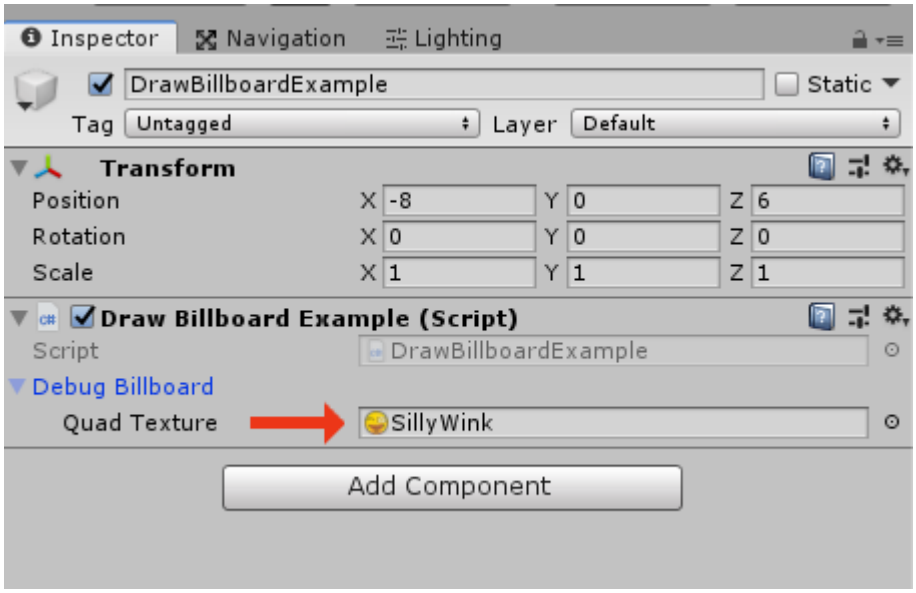
1. To use billboard, first you need to create a variable of type *DebugBillboard*:

```
public DebugBillboard m_DebugBillboard;
```

2. Add the billboard object to the list, so that DDT draw it:

```
voidStart()
{
    DrawDebugTools.AddBillboardToDrawList(m_DebugBillboard);
}
```

3. Select the object with the script and set a texture asset in the inspector:



4. Set billboard position:

```
void Update()
{
  // Set billboard position
  m_DebugBillboard.Position = transform.position + new Vector3(0.0f, 1.0f, 0.0f);
}
```

Billboard Methods

Change the width and height of a billboard:

```
m_DebugBillboard.Width = 1.0f;
m_DebugBillboard.Height = 1.0f;
```

Change visibility of the billboard:

```
m_DebugBillboard.IsHidden = false;
```

Make the billboard always face camera by setting IsBillboard to true:

```
m_DebugBillboard.IsBillboard = true;
```

If IsBillboard is set to false, you can set billboard rotation as you like:

```
m_DebugBillboard.EulerRotation = Vector3.zero;
```

When you no longer need a billboard, remove it by calling this method:

```
DrawDebugTools.RemoveBillboardFromList(m_DebugBillboard);
```

> **Note**: If you know you will remove the billboard at some point, make sure it's not null before using it in your code.

Agent Path Visualization

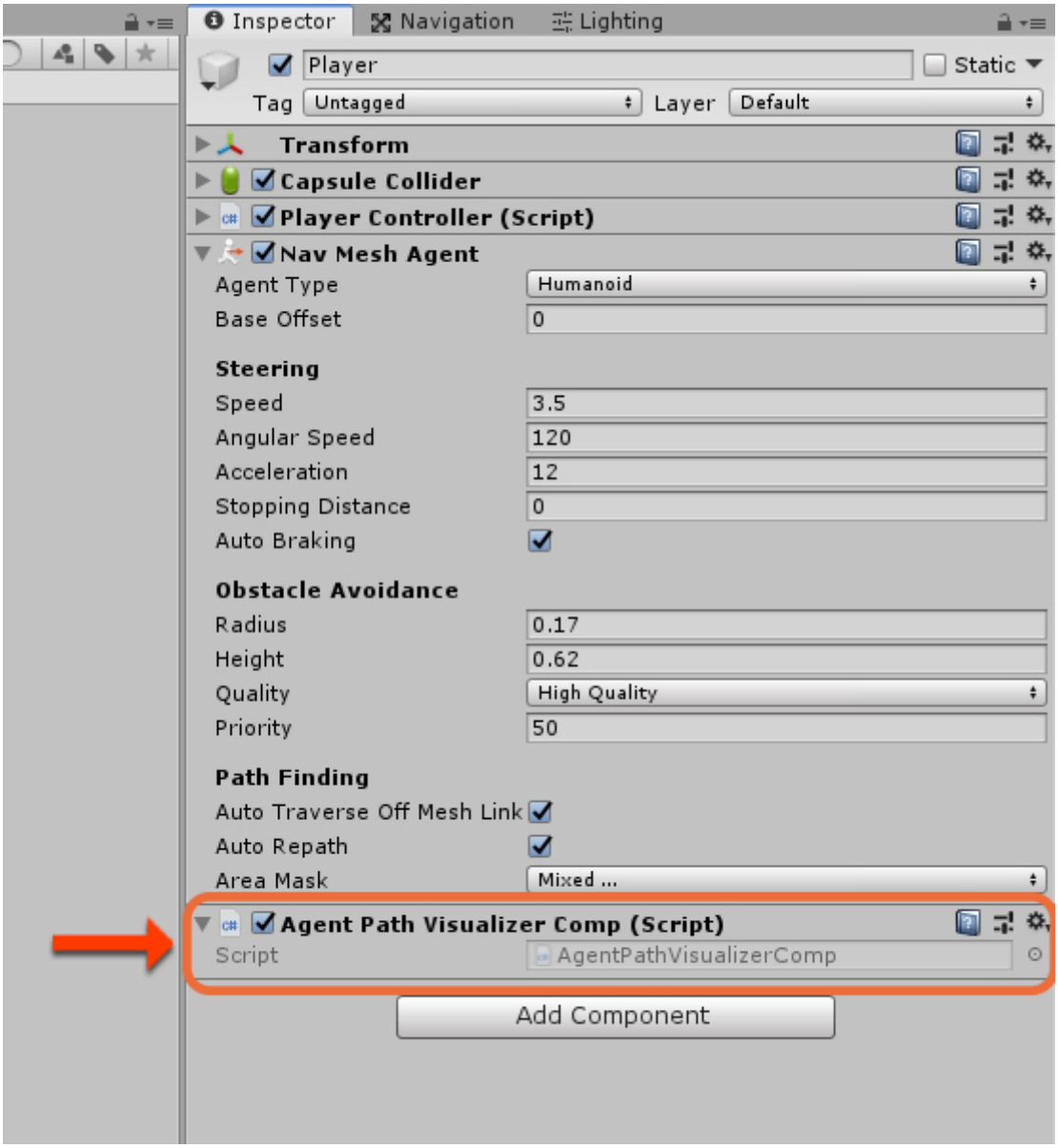
Overview

AgentPathVisualizerComp is component that you attach to an AI agent object that allow you to visualize its navigable path:

How To Use

1. Select the game object that has the NavMeshAgent component attached to it.

2. Add ***AgentPathVisualizerComp*** component to the game object:



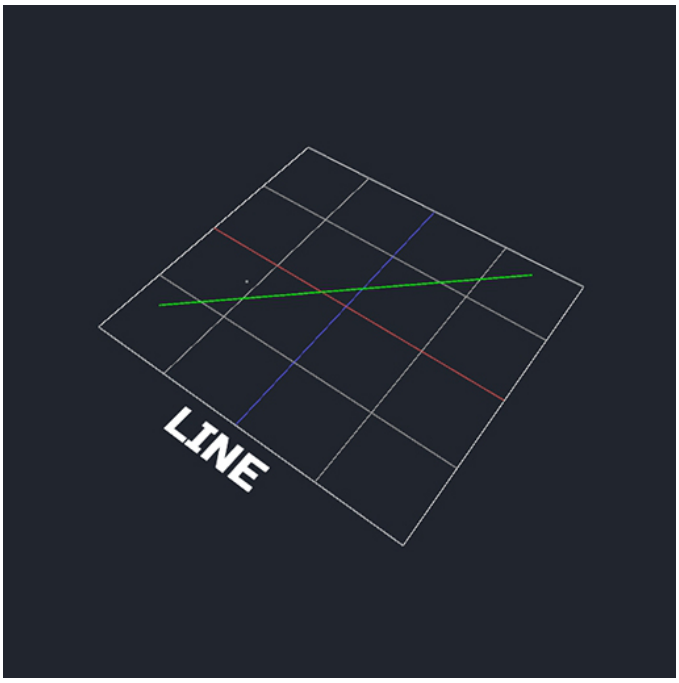
3. When you hit play or launch your game, you will see the path of the NavMesh Agent when it starts to move.

Draw Functions

Line

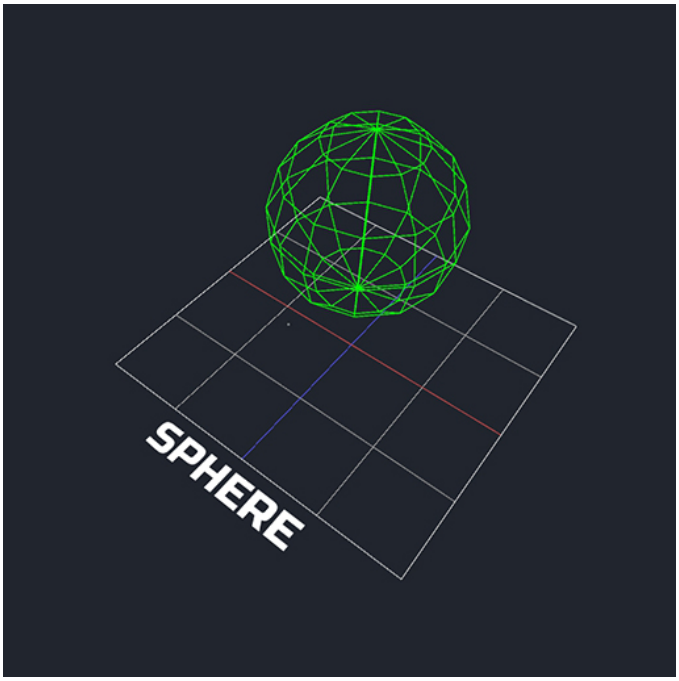
```
/// <summary>
/// Draw a 3D line in space
/// </summary>
```

```
/// <param name="LineStart">Position of the line start</param>
/// <param name="LineEnd">Position of the line end</param>
/// <param name="Color">Color of the line</param>
/// <param name="LifeTime">Line life time</param>
public static void DrawLine(Vector3 LineStart, Vector3 LineEnd, Color Color, float Lif
eTime = 0.0f)
```



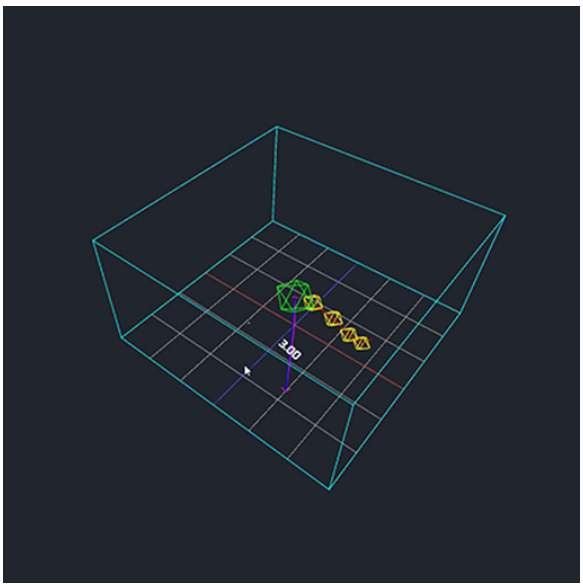
Sphere

```
/// <summary>
/// Method to draw a wire sphere
/// </summary>
/// <param name="Center">Position of the sphere</param>
/// <param name="Radius">Raduis of the sphere</param>
/// <param name="Segments">Segments count that form the sphere</param>
/// <param name="Color">Color of the sphere</param>
/// <param name="LifeTime">Lifetime before stop drawing the sphere</param>
public static void DrawSphere(Vector3 Center, float Radius, int Segments, Color Color,
float LifeTime = 0.0f)
```



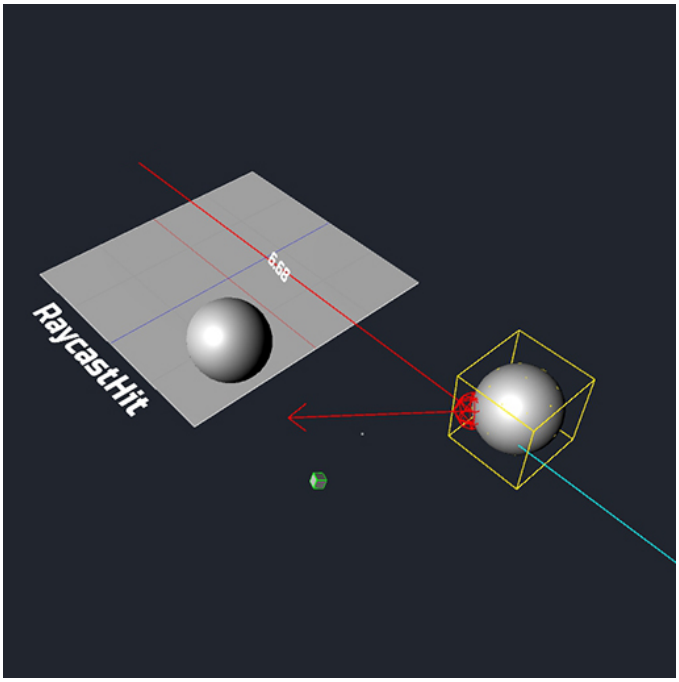
Box

```
/// <summary>
/// Draw a box
/// </summary>
/// <param name="Center">Center position of the box</param>
/// <param name="Rotation">Rotaion of the box</param>
/// <param name="Size">The size of the box</param>
/// <param name="Color">Color of the box</param>
/// <param name="LifeTime">Box life time</param>
public static void DrawBox(Vector3 Center, Quaternion Rotation, Vector3 Size, Color Co
lor, float LifeTime = 0.0f)
```



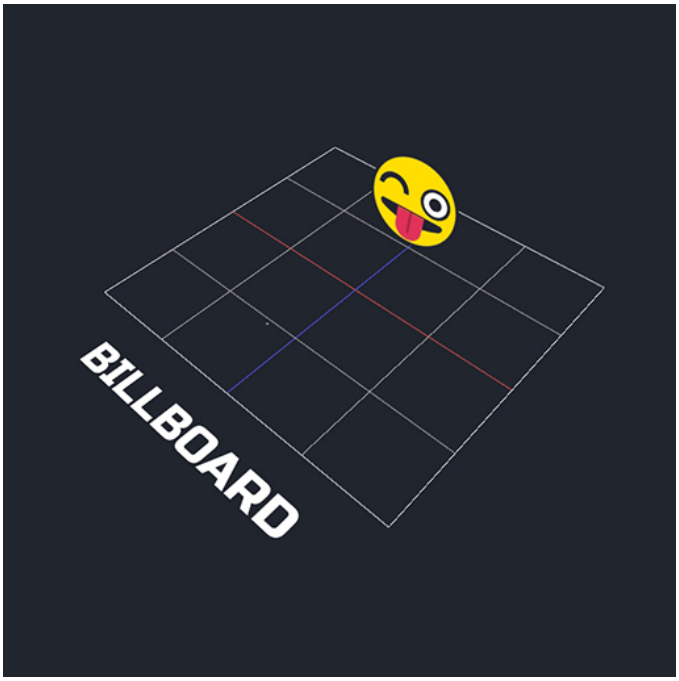
Ray Cast Hit

```
/// <summary>
/// Draw RaycastHit structures infos
/// </summary>
/// <param name="Origin">The starting point of the ray in world coordinates</param>
/// <param name="Direction">The direction of the ray</param>
/// <param name="MaxDistance">Max distance the ray should check for collisions</param>
/// <param name="HitInfos">Info about where the closest collider was hit</param>
/// <param name="LifeTime">Draw life time</param>
public static void DrawRaycastHit(Vector3 Origin, Vector3 Direction, float MaxDistanc
e, RaycastHit HitInfos, float LifeTime = 0.0f)
```



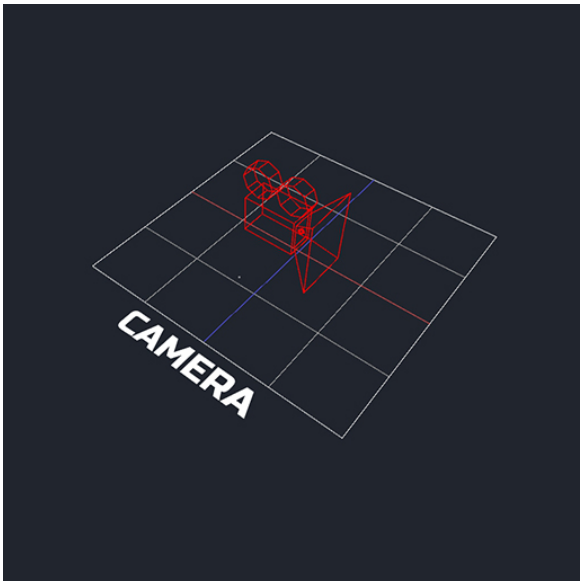
Billboard

```
public DebugBillboard m_DebugBillboard;

void Start()
{
    DrawDebugTools.AddBillboardToDrawList(m_DebugBillboard);
}

void Update()
{
    // Set billboard position
    m_DebugBillboard.Position = transform.position + new Vector3(0.0f, 1.0f, 0.0f);
}
```



Main Camera

```
/// <summary>
/// Draw a 3D representation of the main camera
/// </summary>
/// <param name="Color">Camera shape color</param>
/// <param name="Scale">Scale of the shape</param>
/// <param name="LifeTime">Shape life time</param>
public static void DrawActiveCamera(Color Color, float LifeTime = 0.0f)
```



Camera

```
/// <summary>
/// Draw a 3D representation of a camera
/// </summary>
/// /// <param name="Cam">Camera to draw</param>
/// <param name="Color">Camera representation color</param>
/// <param name="Scale">Scale of the shape</param>
/// <param name="LifeTime">Shape life time</param>
public static void DrawCamera(Camera Cam, Color Color, float LifeTime = 0.0f)
```

Camera Frustum

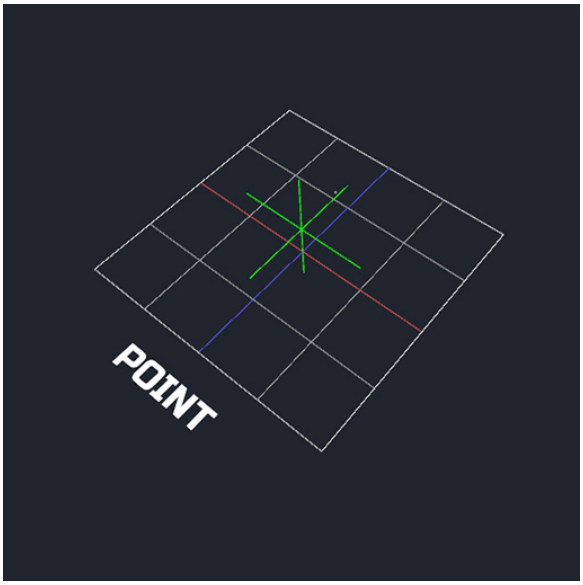
```
/// <summary>
/// Draw camera frustum
/// </summary>
/// <param name="Camera">Target camera</param>
/// <param name="Color">Frustum color</param>
/// <param name="LifeTime">Frustum life time</param>
public static void DrawFrustum(Camera Camera, Color Color, float LifeTime = 0.0f)
```

Grid

```
/// <summary>
/// Draw a grid in 3D space
/// </summary>
/// <param name="Position">Position of the grid</param>
/// <param name="GridSize">Grid size</param>
/// <param name="CellSize">Grid cell size</param>
/// <param name="LifeTime">Grid life time</param>
public static void DrawGrid(Vector3 Position, float GridSize, float CellSize, float Li
feTime)
```

Point

```
/// <summary>
/// Draw a 3D point in space
/// </summary>
/// <param name="Position">Position of the point</param>
/// <param name="Size">Size of the point</param>
/// <param name="Color">Color of the point</param>
/// <param name="LifeTime">Point life time</param>
public static void DrawPoint(Vector3 Position, float Size, Color Color, float LifeTime
= 0.0f)
```

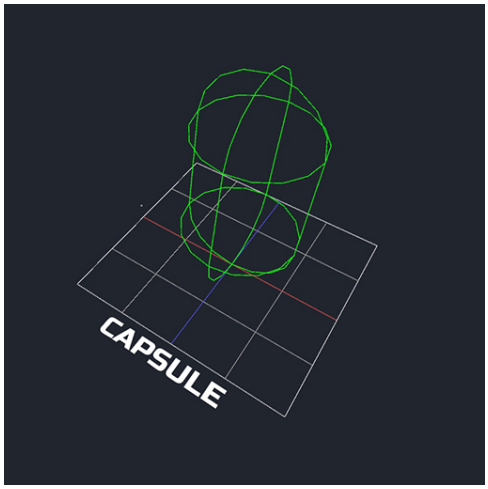


Bounds

```
/// <summary>
/// Draw bounds
/// </summary>
/// <param name="InBounds">Bounds to draw</param>
/// <param name="Color">Drawing colorolor</param>
/// <param name="LifeTime">Draw life time</param>
public static void DrawBounds(Bounds InBounds, Color Color, float LifeTime = 0.0f)
```

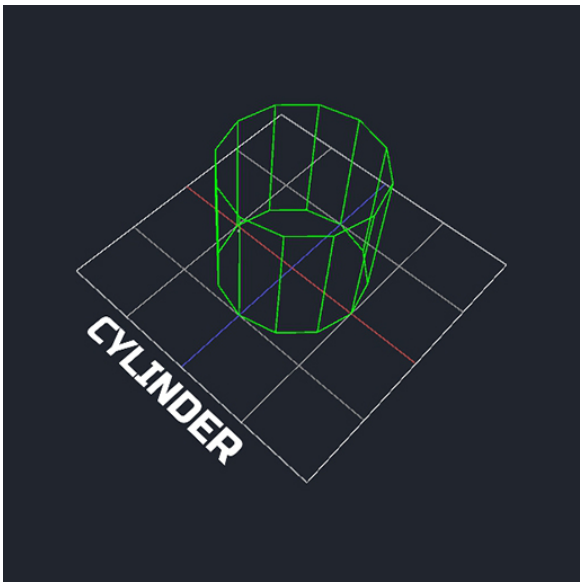
Capsule

```
/// <summary>
/// Draw 3D capsule
/// </summary>
/// <param name="Center">Center position of the capsule</param>
/// <param name="HalfHeight">Capsule half height</param>
/// <param name="Radius">Capsule radius</param>
/// <param name="Rotation">Capsule rotation</param>
/// <param name="Color">Capsule color</param>
/// <param name="LifeTime">Capsule life time</param>
public static void DrawCapsule(Vector3 Center, float HalfHeight, float Radius, Quatern
ion Rotation, Color Color, float LifeTime = 0.0f)
```



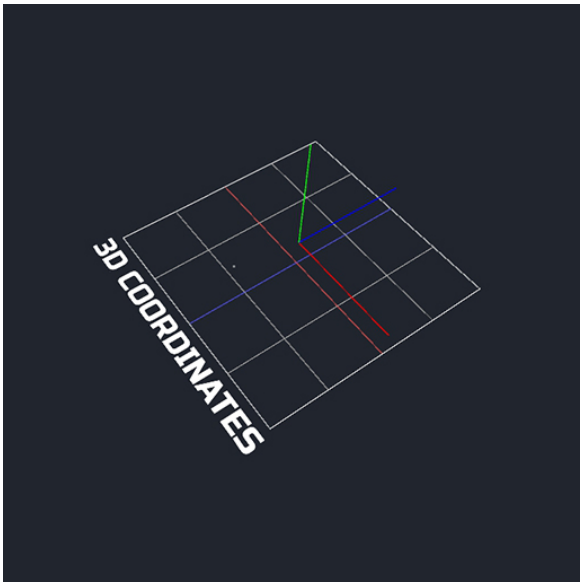
Cylinder

```
/// <summary>
/// Draw a 3D cylinder
/// </summary>
/// <param name="Start">Cylinder start position</param>
/// <param name="End">Cylinder end position</param>
/// <param name="Radius">Cylinder radius</param>
/// <param name="Segments">Cylinder segments count</param>
/// <param name="Color">Color of the cylinder</param>
/// <param name="LifeTime">Cylinder life time</param>
public static void DrawCylinder(Vector3 Start, Vector3 End, float Radius, int Segment
s, Color Color, float LifeTime = 0.0f)
```



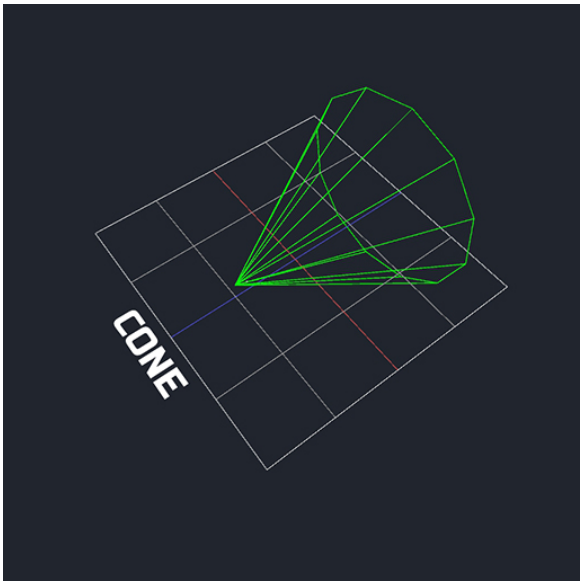
3D Coordinates

```
/// <summary>
/// Draw a 3D coordinates
/// </summary>
/// <param name="Position">Position of the coordinates</param>
/// <param name="Rotation">Rotation of the coordinates</param>
/// <param name="Scale">Scale of the coordinate</param>
/// <param name="LifeTime">Coordinates lifetime</param>
public static void Draw3DCoordinates(Vector3 Position, Quaternion Rotation, float Scal
e, float LifeTime = 0.0f)
```



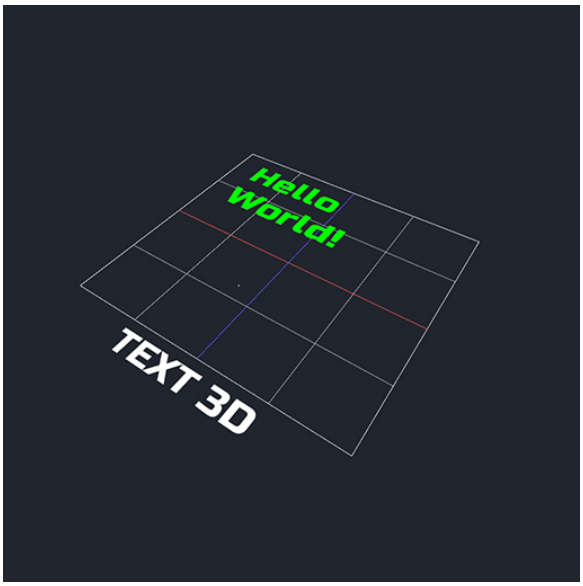
Cone

```
/// <summary>
/// Draw a 3D cone
/// </summary>
/// <param name="Position">Cone position</param>
/// <param name="Direction">Cone direction</param>
/// <param name="Length">Cone length</param>
/// <param name="AngleWidth">Cone angle with</param>
/// <param name="AngleHeight">Cone angle height</param>
/// <param name="Segments">Cone segments count</param>
/// <param name="Color">Cone color</param>
/// <param name="LifeTime">Cone life time</param>
public static void DrawCone(Vector3 Position, Vector3 Direction, float Length, float A
ngleWidth, float AngleHeight, int Segments, Color Color, float LifeTime = 0.0f)
```



3D Text

```
/// <summary>
/// Draw 3D text
/// </summary>
/// <param name="Position">Position of the text</param>
/// <param name="Rotation">Rotation of the text</param>
/// <param name="Text">Text string</param>
/// <param name="Anchor">Text anchor</param>
/// <param name="TextColor">Text color</param>
/// <param name="TextSize">Text size</param>
/// <param name="LifeTime">Text life time</param>
public static void DrawString3D(Vector3 Position, Quaternion Rotation, string Text, Te
xtAnchor Anchor, Color TextColor, float TextSize = 1.0f, float LifeTime = 0.0f)
```



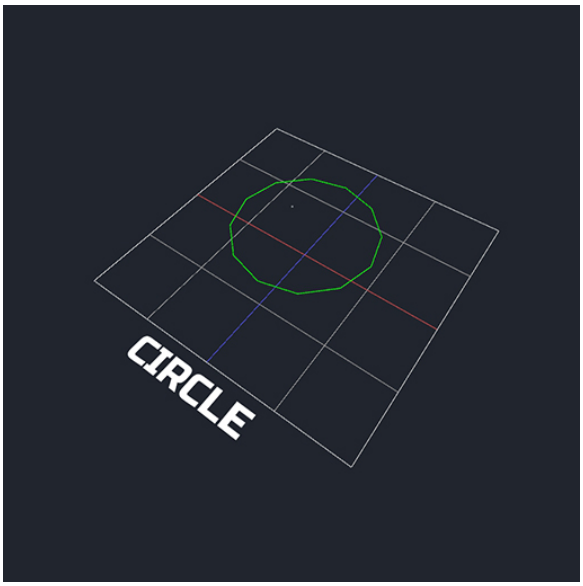
Circle

Draw a simple 3D circle

```
/// <summary>
/// Draw a 3D circle
/// </summary>
/// <param name="Center">Centre position of the circle</param>
/// <param name="Rotation">Rotation of the circle</param>
/// <param name="Radius">Radius of the circle</param>
/// <param name="Segments">Segments count in the circle</param>
/// <param name="Color">Color of the circle</param>
/// <param name="LifeTime">Circle life time</param>
public static void DrawCircle(Vector3 Center, Quaternion Rotation, float Radius, int S
egments, Color Color, float LifeTime = 0.0f)
```

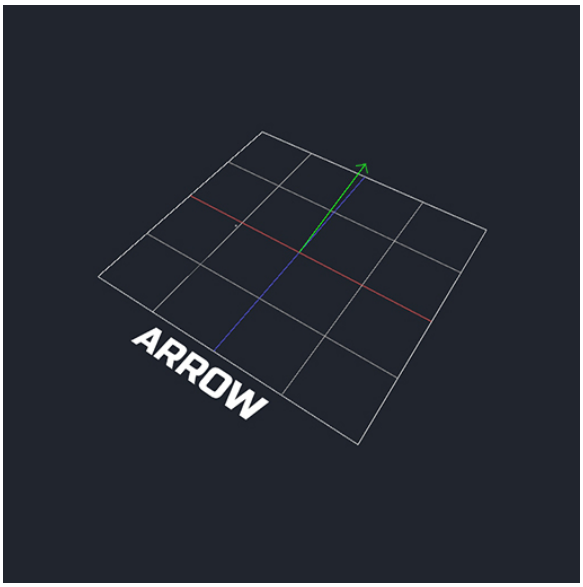
Draw a 3D circle on plane defined by axis (XZ, XY, YZ)

```
/// <summary>
/// Draw a 3D circle on a plane defined by axis (XZ, XY, YZ)
/// </summary>
/// <param name="Center">Centre position of the circle</param>
/// <param name="Radius">Radius of the circle</param>
/// <param name="Segments">Segments count in the circle</param>
/// <param name="Color">Color of the circle</param>
/// <param name="DrawPlaneAxis">Plane axis to draw circle in (XZ, XY, YZ)</param>
/// <param name="LifeTime">Circle life time</param>
public static void DrawCircle(Vector3 Center, float Radius, int Segments, Color Color,
EDrawPlaneAxis DrawPlaneAxis = EDrawPlaneAxis.XZ, float LifeTime = 0.0f)
```



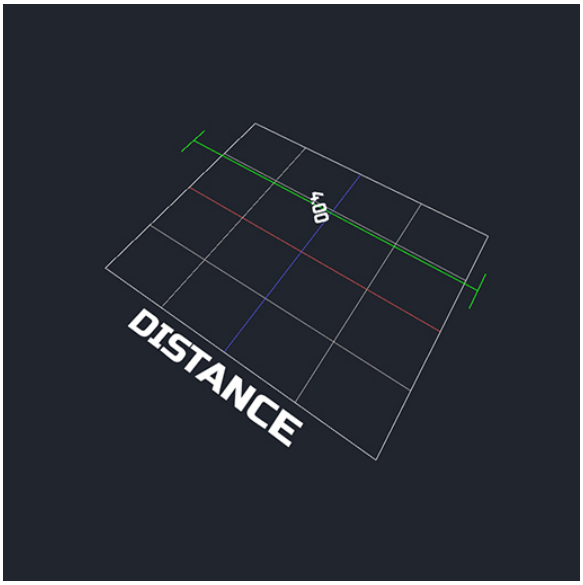
Arrow

```
/// <summary>
/// Draw directional arrow
/// </summary>
/// <param name="ArrowStart">Arrow start position</param>
/// <param name="ArrowEnd">Arrow end position</param>
/// <param name="ArrowSize">Arrow size</param>
/// <param name="Color">Arrow color</param>
/// <param name="LifeTime">Arrow life time</param>
public static void DrawDirectionalArrow(Vector3 ArrowStart, Vector3 ArrowEnd, float Ar
rowSize, Color Color, float LifeTime = 0.0f)
```



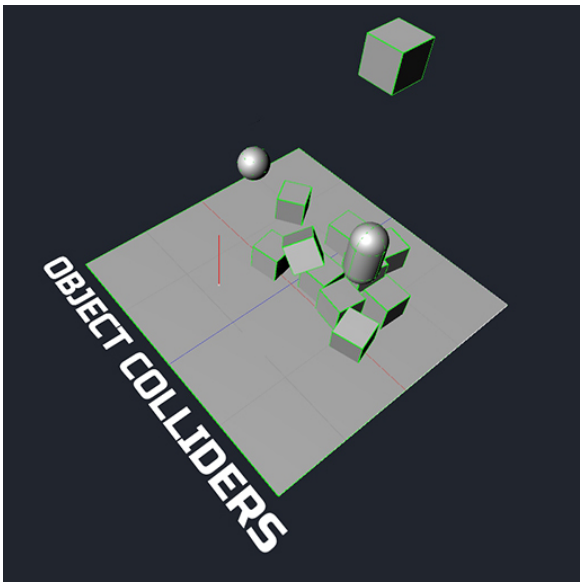
Distance

```
/// <summary>
/// Draw a mesure tool to mesure distance between two points
/// </summary>
/// <param name="Start">Start position</param>
/// <param name="End">End position</param>
/// <param name="Color">Color of the mesure tool</param>
/// <param name="LifeTime">Draw life time</param>
public static void DrawDistance(Vector3 Start, Vector3 End, Color Color, float TextSiz
e = 1.0f, float LifeTime = 0.0f)
```



Object Colliders

```
/// <summary>
/// Draw colliders compoenent on game object and its children
/// </summary>
/// <param name="Object">Game object that its colliders will be drawn</param>
/// <param name="Color">Colliders color</param>
/// <param name="LifeTime">Drawing time</param>
public static void DrawObjectColliders(GameObject Object, Color Color, float LifeTime
 = 0.0f)
```



Log Message

```
/// <summary>
/// Log message text on screen
/// </summary>
/// <param name="LogMessage">Log message string</param>
/// <param name="Color">Log messsage color</param>
/// <param name="LifeTime">Log life time</param>
public static void Log(string LogMessage, Color Color, float LifeTime = 0.0f)
```

Simplified log function, with white as default color for text:

```
/// <summary>
/// Simplified function to log message text on screen
/// </summary>
/// <param name="LogMessage">Log message string</param>
/// <param name="LifeTime">Log life time</param>
public static void Log(string LogMessage, float LifeTime = 0.0f)
```