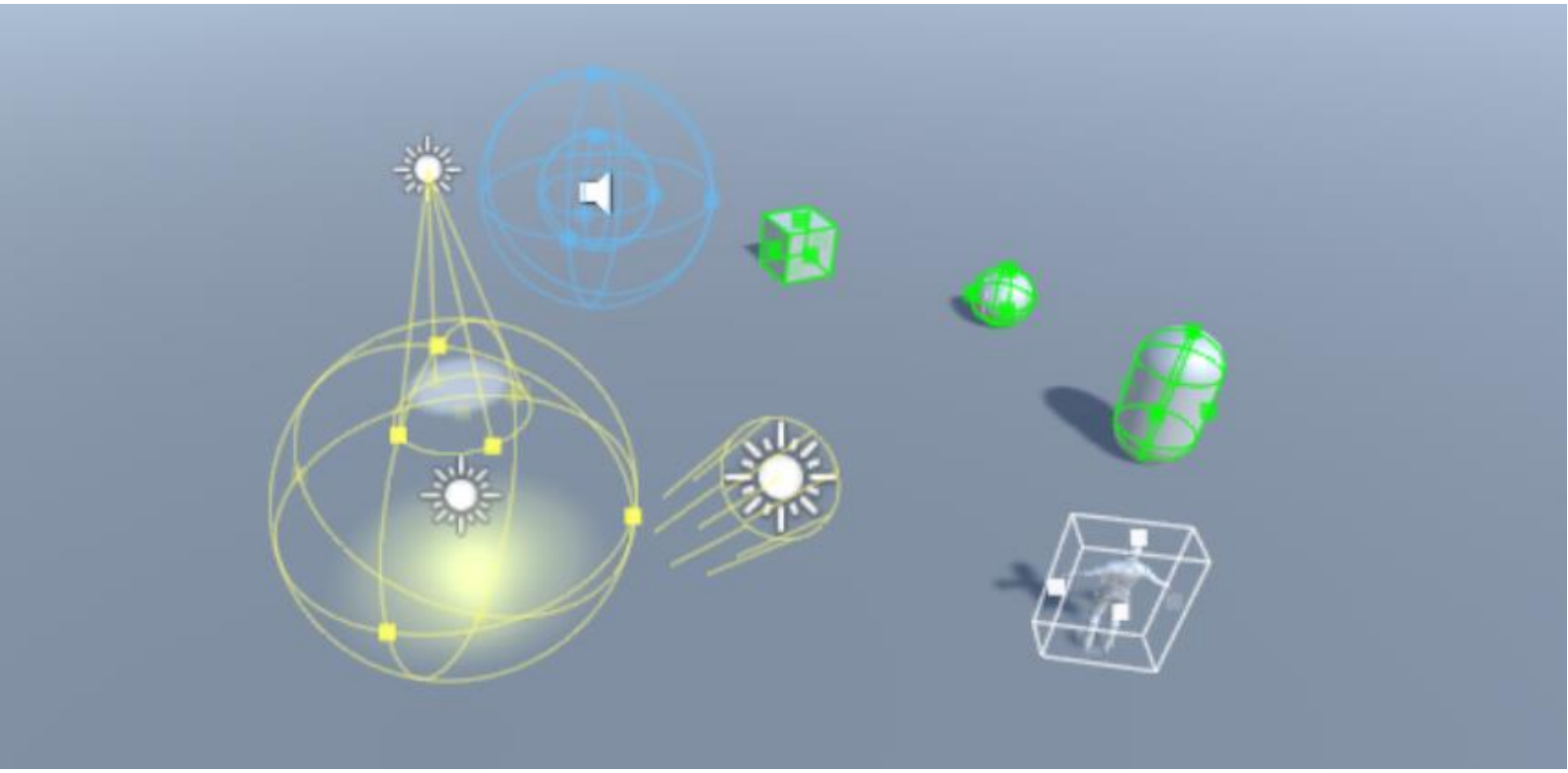

V1.3.3 24.05.2017

Table of contents

Introduction

Runtime Gizmos is a set of scripts, which will help you to implement gizmo editing in your scene/level editor.

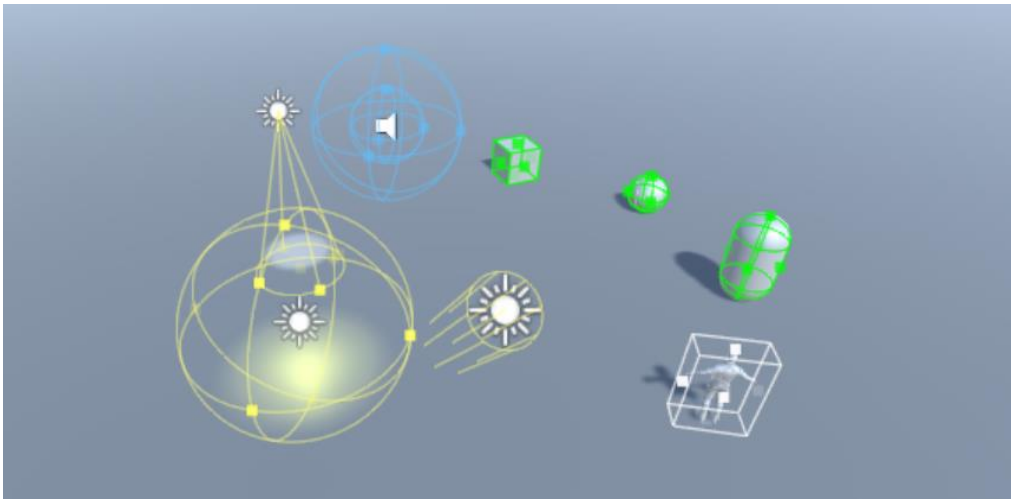

Fig.1 – Runtime Gizmos

Features

- Box, Sphere, Capsule collider gizmos
- Point, Spot, Direct light gizmos
- Audio Source, Reverb Zone gizmos
- SkinnedMeshRenderer gizmo
- Undo & Redo;
- Extensible base classes

Package Structure

Gizmos for Colliders, Lights and AudioSources in **Assets/Battlehub/RTGizmos**

Common classes located in **Asset/Battlehub/RTCommon**,

Helper classes in **Assets/Battehub/Utils,**

Demo scene in **Assets/Battlehub/RTGizmos/Demo**

Each folder organized as following:

/Scripts – for runtime scripts

/Scripts/Editor for editor scripts

/Prefabs for prefabs

/Shader/Resources for shaders

1. Runtime Gizmos

There are nine gizmos included in this package: **BoxColliderGizmo**, **SphereColliderGizmo**, **CapsuleColliderGizmo**, **PointLightGizmo**, **SpotlightGizmo**, **DirectionalLightGizmo**, **AudioSourceGizmo**, **AudioReverbZoneGizmo** and **SkinnedMeshRendererGizmo**. Each gizmo could be used to control certain properties of corresponding components. Runtime Gizmos behaves almost identical to their equivalents in unity editor. There are also several base classes which might be helpful for implementation of additional gizmos. All gizmos, their base classes, rendering classes and all required shaders can be found in **Assets/Battlehub/RTGizmos folder.**

Each Gizmo script allows you to choose raycasting camera, selection margin (in screen space coordinates), target object, size of grid, line color, handle color and selection color as well as the key which will switch position gizmo to “Unit Snapping mode”

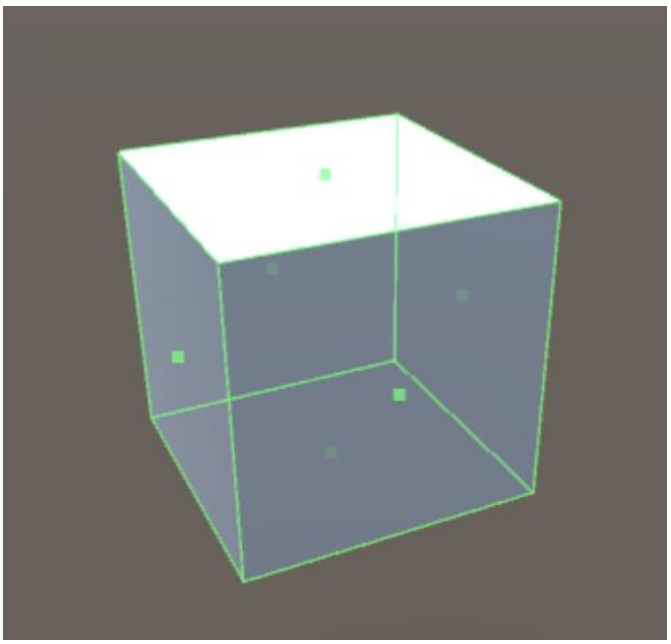

Fig. 4.1 Runtime Gizmo

1.1 Base Gizmo

Located in Assets/Battlehub/RTGizmos/Scripts/BaseGizmo.cs

BaseGizmo is base class for BoxGizmo, SphereGizmo, CapsuleGizmo and ConeGizmo. All gizmos have following settings:

- **Grid Size** – size of step in unit snapping mode (1.0 by default)
- **Line Color** –color of gizmo lines;
- **Handles Color –** color of handle

- **Selection Color** – color of selected handle.
- **Enable Undo** – if set to true then RuntimeGizmo will write all changes to undo stack (true by default)
- **Unit Snap Key** – key wich will switch RuntimeGimzo to unity snapping mode (Left Ctrl by default);
- **Target** – reference to target object

1.2 Box Gizmo

Located in Assets/Battlehub/RTGizmos/Scripts/BoxGizmo.cs

Base class for all gizmos that have box shape:

- BoxColliderGizmo,
- SkinnedMeshRendererGizmo

1.3 Sphere Gizmo

Located in Assets/Battlehub/RTGizmos/Scripts/SphereGizmo.cs

Base class for all gizmos that have sphere shape:

- Sphere Collider Gizmo,
- Pointlight Gizmo,
- Audio Source Gizmo,
- Audio Reverb Zone Gizmo

1.4 Capsule Gizmo

Located in Assets/Battlehub/RTGizmos/Scripts/CapsuleGizmo.cs

Base class for all gizmos that have capsule shape:

- Capsule Collider Gizmo

1.5 Cone Gizmo

Located in Assets/Battlehub/RTGizmos/Scripts/ConeGizmo.cs

Base class for all gizmos that have cone shape:

- Spotlight Gizmo

1.6 Box Collider Gizmo

Located in Assets/Battlehub/RTGizmos/Scripts/BoxColliderGizmo.cs

Box Collider Gizmo could be attached to object with BoxCollider

1) Create GameObject with BoxCollider
2) Assign BoxColliderGizmo script to it

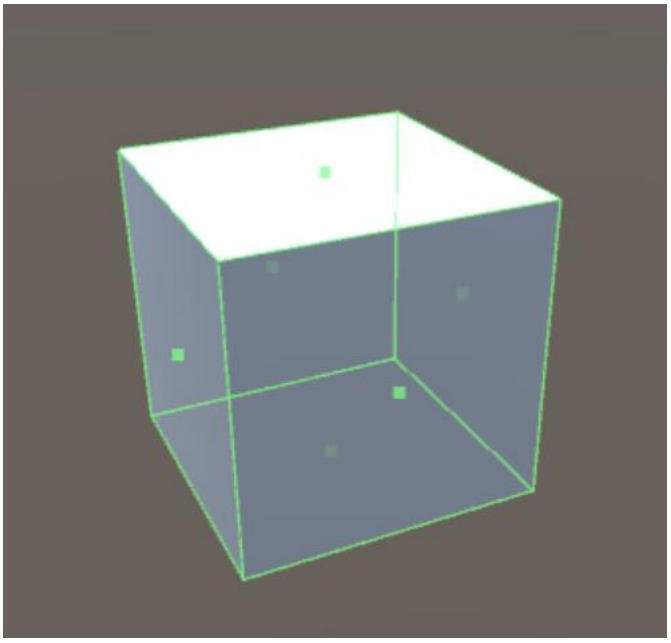

Fig. 4.2 Box Collider Gizmo

1.7 Sphere Collider Gizmo

Located in Assets/Battlehub/RTGizmos/Scripts/SphereColliderGizmo.cs

Sphere Collider Gizmo could be attached to object with SphereCollider

1) Create GameObject with Sphere Collider
2) Assign SphereColliderGizmo script to it

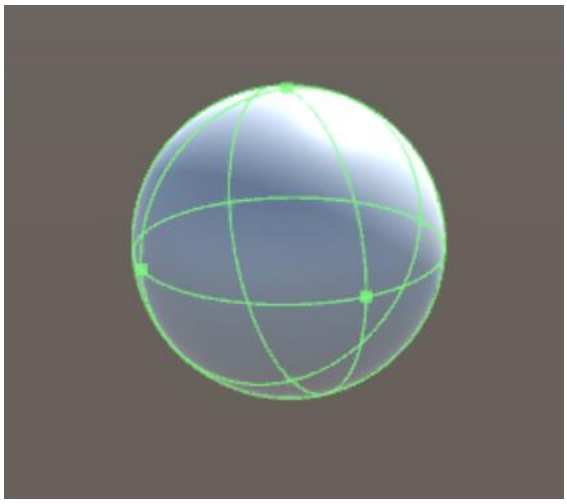

Fig. 4.3 Sphere Collider Gizmo

1.8 Capsule Collider Gizmo

Located in Assets/Battlehub/RTGizmos/Scripts/CapsuleColliderGizmo.cs

Capsule Collider Gizmo could be attached to object with CapsuleCollider

1) Create GameObject with Capsule Collider
2) Assign CapsuleColliderGizmo script to it

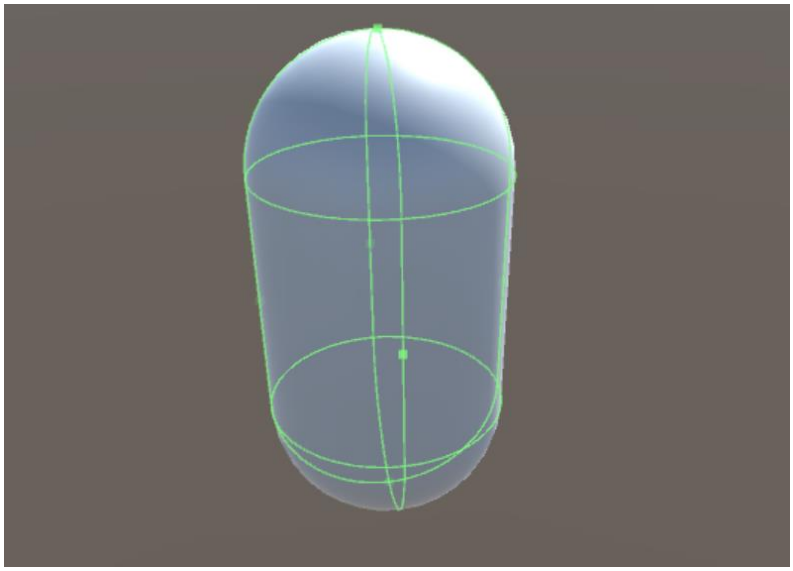

Fig.4.4 Capsule Collider Gimzo

1.9 Point Light Gizmo

Located in Assets/Battlehub/RTGizmos/Scripts/PointLightGizmo.cs

Point Light Gizmo could be attached to Point Light

1) Create Point Light
2) Assign LightGizmo script to it

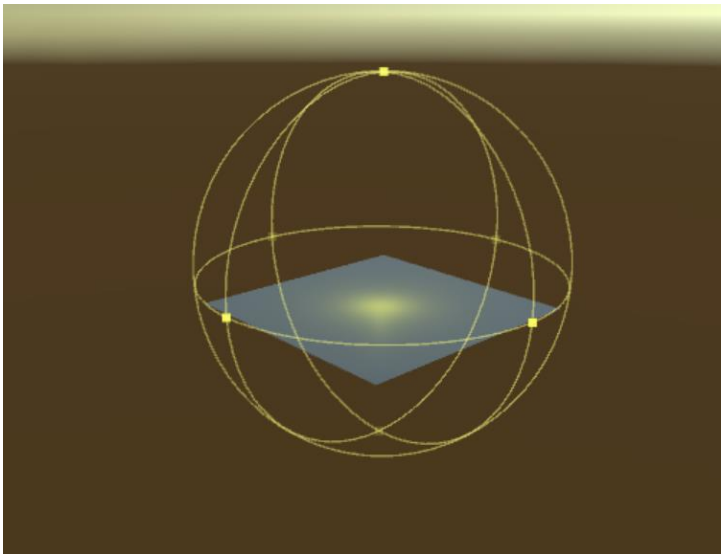

Fig.4.5 Point Light Gizmo

1.10 Spot Light Gizmo

Located in Assets/Battlehub/RTGizmos/Scripts/SpotLightGizmo.cs

Spot Light Gizmo could be attached to Spot Light

1) Create Spot Light
2) Assign LightGizmo script to it

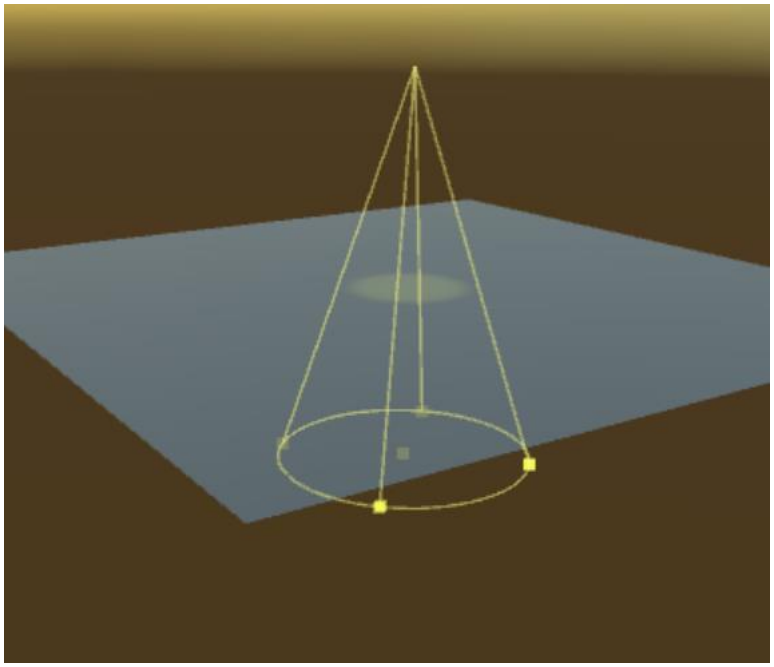

Fig.4.6 Spot Light gizmo

1.11 Directional Light Gizmo

Located in Assets/Battlehub/RTGizmos/Scripts/DirectionalLightGizmo.cs

Directional Light Gizmo could be attached to DirectionalLight

1) Create Directional Light
2) Assign Light Gizmo Script to it

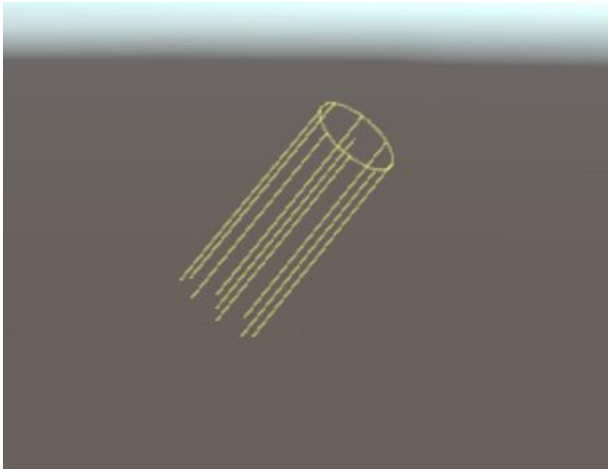

Fig.4.7 Directional Light Gizmo

1.12 Audio Source Gizmo

Located in Assets/Battlehub/RTGizmos/Scripts/AudioSourceGizmo.cs

Audio Source Gizmo could be attached to AudioSource

1) Create Audio Source
2) Assign Audio Source Gizmo Script to it

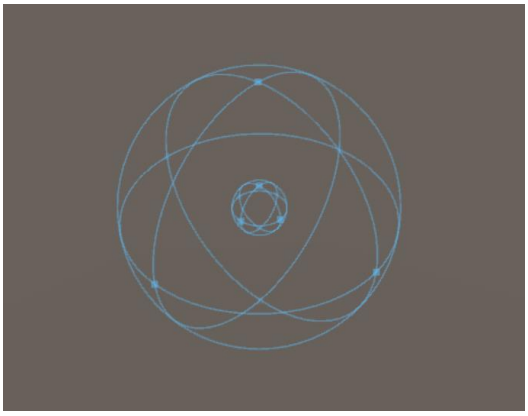

Fig. 4.8 Audio Source Gizmo

1.13 Audio Reverb Zone Gizmo

Located in Assets/Battlehub/RTGizmos/Scripts/AudioReverbZoneGizmo.cs

Same as AudioSouce Gizmo

1.14 SkinnedMeshRenderer Gizmo

Located in /Battlehub/RTGizmos/Scripts/SkinnedMeshRendererGizmo.cs

SkinnedMeshRenderer Gizmo could be attached to object with SkinnedMesh

1) Create GameObject with SkinnedMeshRenderer
2) Assign SkinnedMeshRenderer Gizmo Script to it

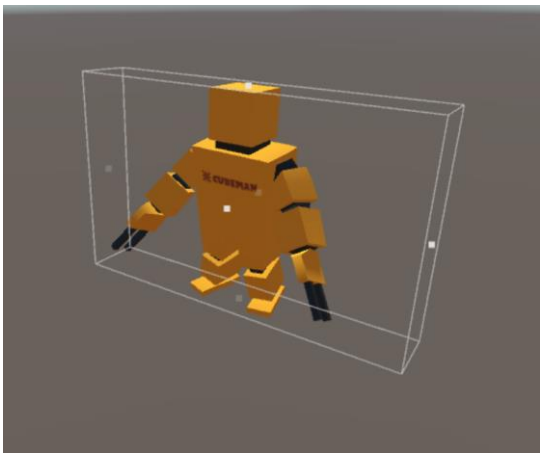

Fig. 4.9 Skinned Mesh Renderer Gizmo

2. Gizmos Rendering

2.1 IGL

Located in Assets/Battlehub/RTCommon/Scripts/GLRenderer.cs

Implement this interface to make object available to GLRenderer

```
public interface IGL {
        void Draw();
}
```

2.2 GLRenderer

Located in Assets/Battlehub/RTCommon/Scripts/GLRenderer.cs

GLRenderer is a singleton used by GLCamera script to render all registered objects. Register object for rendering by calling `public void Add(IGL gl)` method. Cancel object rendering by calling `public void Remove(IGL gl)` method

Example usage:

```
private void OnEnable() {
    if (GLRenderer.Instance != null) {
        GLRenderer.Instance.Add(this);
    }
}

private void OnDisable(){
    if (GLRenderer.Instance != null) {
        GLRenderer.Instance.Remove(this);
    }
}
```

2.3 GLCamera

Located in Assets/Battlehub/RTCommon/Scripts/GLCamera.cs

Attach this script to any camera and GL graphics will be rendered with this camera

```
[ExecuteInEditMode]
public class GLCamera : MonoBehaviour
{
    private void OnPostRender()
    {
        if(GLRenderer.Instance != null)
        {
            GLRenderer.Instance.Draw();
        }
    }
}
```

2.4 Runtime Gizmos

Located in Assets/Battlehub/RTGizmos/Scripts/RuntimeGizmos.cs

This class contains rendering code and helper methods for all runtime gizmos

Scale of dragable handles of runtime gizmos:

```
public const float HandleScale = 2.0f;
```

Draw Selected handle using following method:

```
public static void DrawSelection(
        Vector3 position, Quaternion rotation, Vector3 scale, Color color)
```

Following method is used to draw handles for Box, Sphere and Capsule Gizmos:

```
public static void DrawCubeHandles(
        Vector3 position, Quaternion rotation, Vector3 scale, Color color)
```

Following method is used to draw handles for Cone Gizmo (SpotLight gizmo):

```
public static void DrawConeHandles(
        Vector3 position, Quaternion rotation, Vector3 scale, Color color)
```

Draw Cone with specific height and radius:

```
public static void DrawWireConeGL(
float height, float radius, Vector3 position, Quaternion rotation, Vector3 scale, Color color)
```

Draw capsule along axis (axis: 0 – x, 1 – y, 2 – z)

```
 public static void DrawWireCapsuleGL(
        int axis, float height, float radius, Vector3 position,
        Quaternion rotation, Vector3 scale,  Color color)
```

Draw Directional Light gizmo:

```
public static void DrawDirectionalLight(
        Vector3 position, Quaternion rotation, Vector3 scale, Color color)
```

Draw Sphere:

```
public static void DrawWireSphereGL(
        Vector3 position, Quaternion rotation, Vector3 scale, Color color)
```

Draw Cube:

```
public static void DrawWireCubeGL(
      ref Bounds bounds, Vector3 position, Quaternion rotation, Vector3 scale, Color color)
```

3 Common Infrastructure

3.1 RuntimeUndo

Located in Battlehub/RTCommon/Scripts/RuntimeUndo.cs

This static class is used to record changes, maintain undo/redo stack and perform undo and redo operations.

```
public static class RuntimeUndo
```

Events:

- Raised before undo operation

  ```
  public static event RuntimeUndoEventHandler BeforeUndo;
  ```

- Raised after undo operation

  ```
  public static event RuntimeUndoEventHandler UndoCompleted;
  ```

- Raised before redo operation

  ```
  public static event RuntimeUndoEventHandler BeforeRedo;
  ```

- Raised after redo operation

  ```
  public static event RuntimeUndoEventHandler RedoCompleted;
  ```

- Raised whenever one of the following operations performed: Store, Restore, Purge

  ```
  public static event RuntimeUndoEventHandler StateChanged;
  ```

Properties & Fields:

- Maximum number of records in stack

  ```
  public const int Limit = 8192;
  ```

- Enables or disables Undo&Redo

  ```
  public static bool Enabled { get; set; }
  ```

- True if can perform undo operation

  ```
  public static bool CanUndo { get;}
  ```

- True if can perform redo operation

  ```
  public static bool CanRedo { get;}
  ```

Methods:

- Begin Record multiple changes

  ```
  public static void BeginRecord();
  ```

- End Record multiple changes

  ```
  public static void EndRecord();
  ```

- Register Create object operation:

```
public static void BeginRegisterCreateObject(GameObject g)
public static void RegisterCreatedObject(GameObject g)
```

- Register Destroy object operation:

```
public static void BeginDestroyObject(GameObject g)
public static void DestroyObject(GameObject g);
```

Note that object isn't actually destroyed. It only marked as destroyed. It will be destroyed during Purge operation.

- Record value of member of target object:

```
public static void RecordValue(object target, MemberInfo memberInfo)
```

- Record transform:

```
public static void RecordTransform(Transform target);
```

- Record selection:

```
public static void RecordSelection()
```

- Record object:

```
public static void RecordObject(object target, object state, ApplyCallback
applyCallback, PurgeCallback purgeCallback)
```

To use this method you have to implement apply and purge callback methods. See RecordTransform implementation for more details.

- Performs redo operation:

```
public static void Redo()
```

- Performs undo operation:

```
public static void Undo()
```

- Purge all records. Stack will be cleared, all "marked as destroyed" objects will be destroyed using Object.DestoryImmdiate

```
public static void Purge();
```

- Create new stack and store current undo&redo stack**:**

```
public static void Store()
```

- Restore previouly stored stack:

```
public static void Restore()
```

3.2 RuntimeUndoComponent

Located in Battleuhb/RTCommon/Scripts/RuntimeUndoComponent.cs

Add gameObject with this script to scene to preform undo & redo operations using CTRL + Z, CTRL + Y keys. (or SHIFT + Z, SHIFT + Y inside of unity editor)

Support

If you have any questions, suggestions, you want to talk or you have some issues please send mail to Vadim.Andriyanov@outlook.com or Battlehub@outlook.com.