HORIZON BEND

Ver.1 04.02.2016

Introduction

Horizon Bend (HB) is an editor extension for Unity3D, which is hopefully will help you to create «horizon bending» effect in your project and resolve concomitant problems.

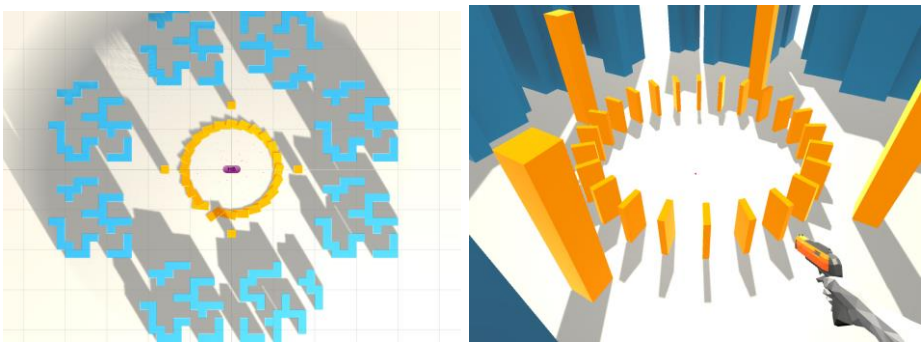

Fig.1 Scene without bending

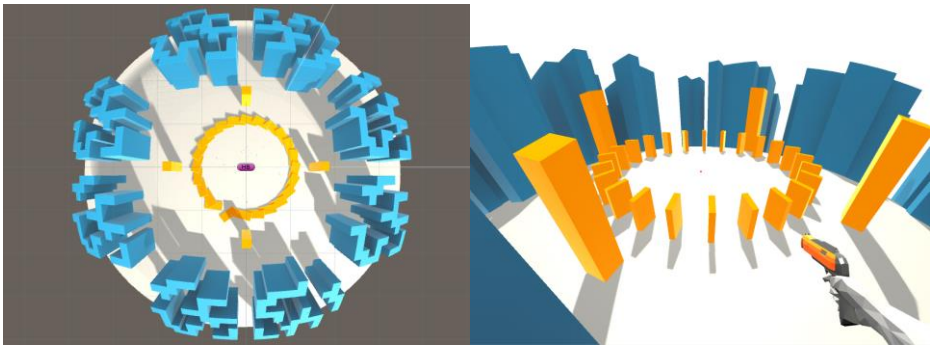

Fig.2 Scene with XZ Y-up bending

HB **has various configurations** allowing you to bend objects in your scene along different axes.

HB **contains about 100 modified shaders** (Standard, Terrain, Nature, Legacy, Mobile, Particles, etc.) The difference with original shaders lies in modification of vertex program. We use well-known approach and offset vertices using the function of parabola. Input of this function based on current HB configuration, bending parameters and position of camera.

There are **four main problems** may occur when using horizon-bending effect and all of these problems **have resolution**:

1. Undesired frustum culling;
2. Irrelevant rays returned by Camera.ScreenPointsToRay function;
3. Straight lines of LineRenderers;
4. Irrelevant point/spot light positions;

Package Structure

HorizonBending Package is located in **Assets/Battlehub/HorizonBending**

Package does not include files with *.shader ext to prevent immediate shader compilation. Instead we use *.hbst files (hbst stands for horizon bend shader template). Shader templates are located in Assets/Battlehub/HorizonBending/Shader/_Templates folder. They are used by **Horizon Bend Configurator** (see next section) to create shader files with required functionality. Templates is almost the same as the original shaders, the only difference is horizon bending functionality. **If you do not need some of shaders in your project, you could simply remove corresponding hbst files. It could signigicantly reduce compile time.**

Editor folder contains ExecuteOnStart script which is used to display Configurator window right after import. ExecuteOnStart will be automatically removed from project.

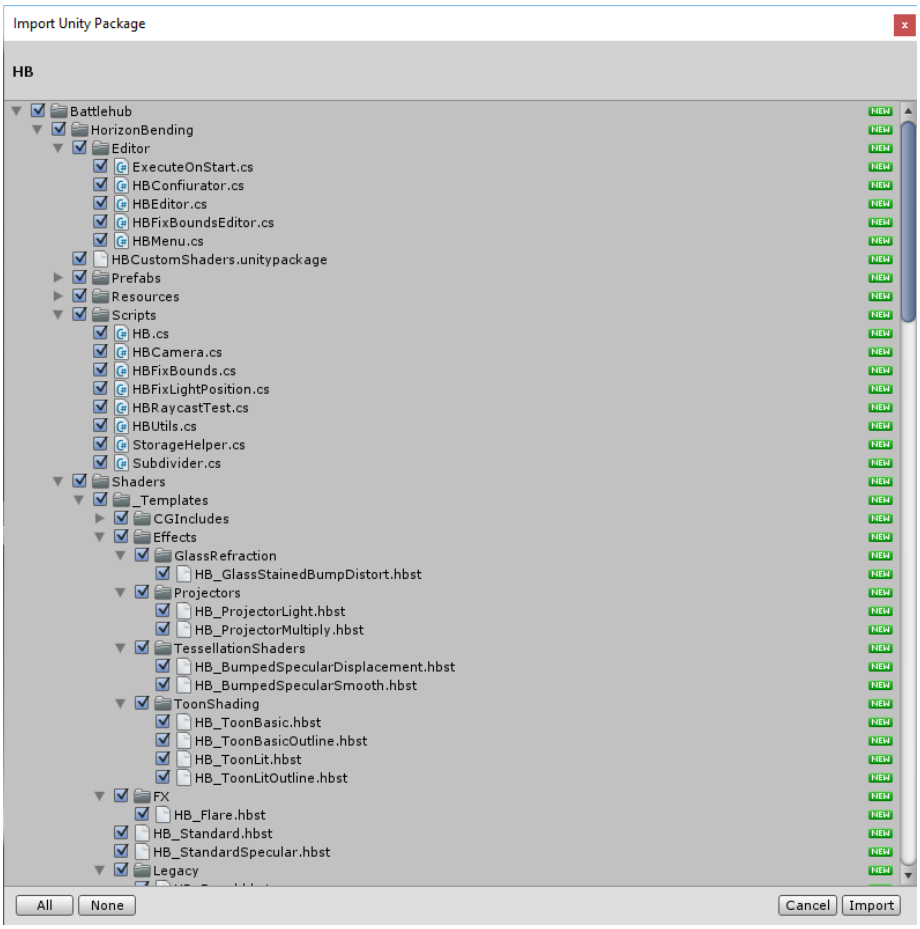

Fig.3 Package Structure

Configuration

Horizon Bend Configuration window, will be opened right after you import Horizon Bend package to your project. Alternatively, you can access this window using Tools->Horizon Bend->Configure menu item

Fig.4 Configuration window

We need this configurator to avoid multicompile and multiple shader variants. However, you can always choose “Multicompile” profile and all shaders will be created with #pragma multicompile directive. Although all configurations will be accessible, it could take a huge amount of time to compile shaders with all features. We suggest you to use “One Shader Feature” profile, which is default.

To choose required variant you should use «World Up Axis» dropdown and «Bend Axis» dropdown. All HB Configurations are shown in Fig. 5.

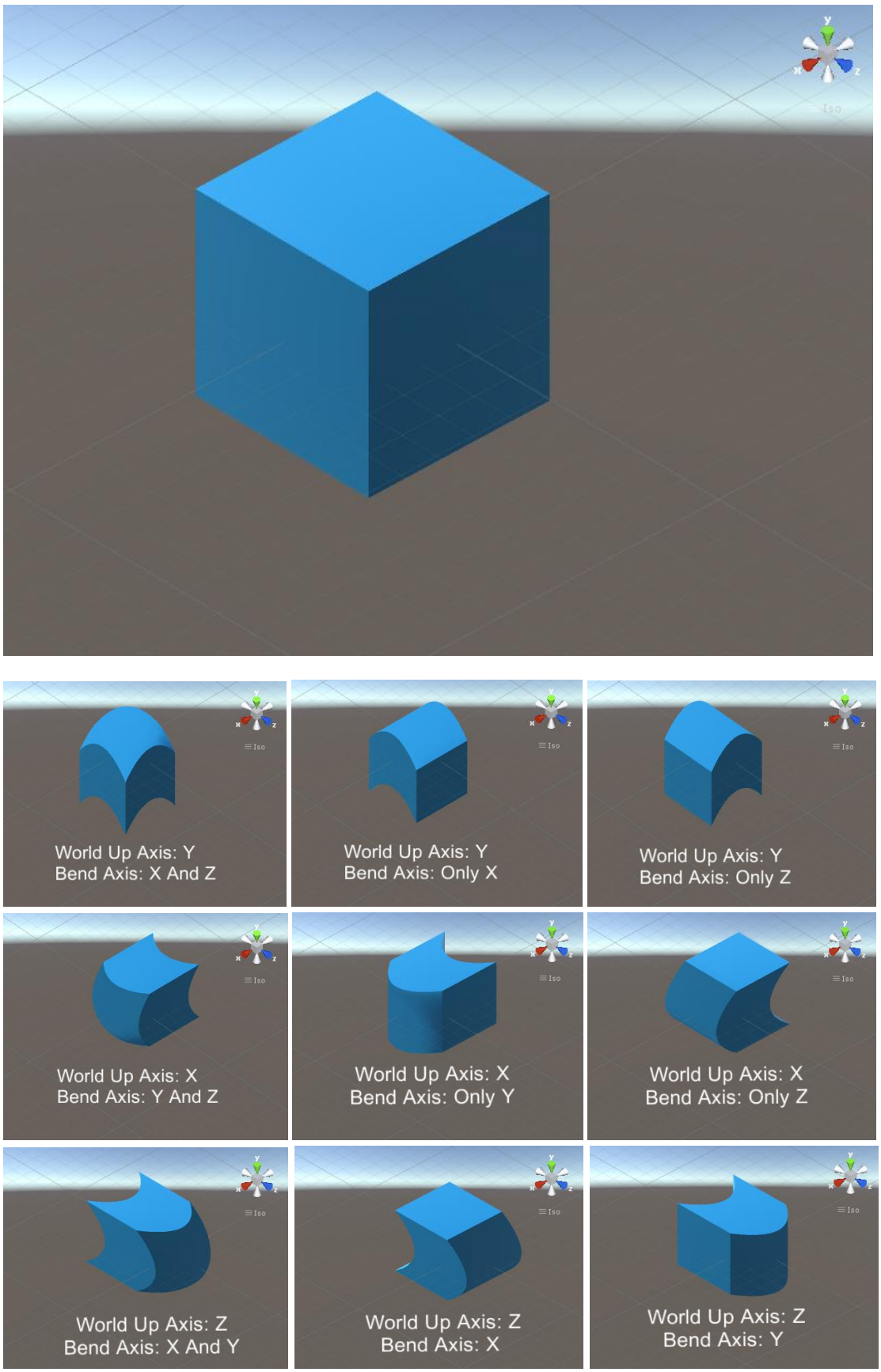

Fig.5 Different HB Configurations

Menu

There are five menu items:

1. Configure is to open Configuration window.
2. Apply is to apply horizon bending to all applicable objects and materials in scene and to create HB.prefab with HB Script attached. HB Script allows you to control horizon-bending parameters. *(In current version Undo won't work, use Remove menu item instead)*
3. Remove used to undo things made by Apply command.
4. Create prefab. Why use this menu item? See corresponding section.
5. Subdivide mesh used to increase vertices count of selected meshes.

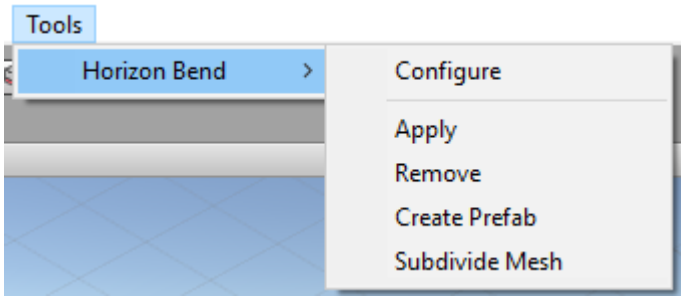

Fig.6 Menu

Apply

If you click Apply then HBEditor.Apply() method will be invoked.

This method do following:

1. Backup HB settings if HB gameobject instantiated
2. Make shallow rollback
3. Find all materials and game objects with renderers in scene
4. Group renderers by mesh and transformation (to work with meshes per group not per object)
5. Replace Default materials 'Default-Material', 'Default-Particle', etc. (because there is no way to save changes to these materials)
6. Replace original shaders with horizon bending shaders (prefixed with HB_) (To find replacement shaders we use name comparison)
7. Create HBFixBounds components and attach them to suitable objects
8. Fix bounds of objects with HBFixBounds component (FixBounds method)
9. Apply default horizon bending settings
10. Restore settings from backup if exist.

Create Prefab

Horizon Bend could modify meshes in some cases. You can drag and drop your object to prefabs folder, but prefab will be unable to find modified mesh. Use create prefab menu item instead. First, it will save modified meshes then it will create prefab or override existing prefab. Created prefabs located in Battlehub\HorizonBending\Prefabs

Subdivide

It is important to know, why do you need to subdivide meshes in some cases?

Suppose, you have a big quad in your scene. It used as a floor of the level in your game. As you know, mesh of the Quad primitive has only four vertices. You probably already figured out that if you apply horizon-bending then all vertices of the quad will be moved down using same offset. Look at the Fig.7.

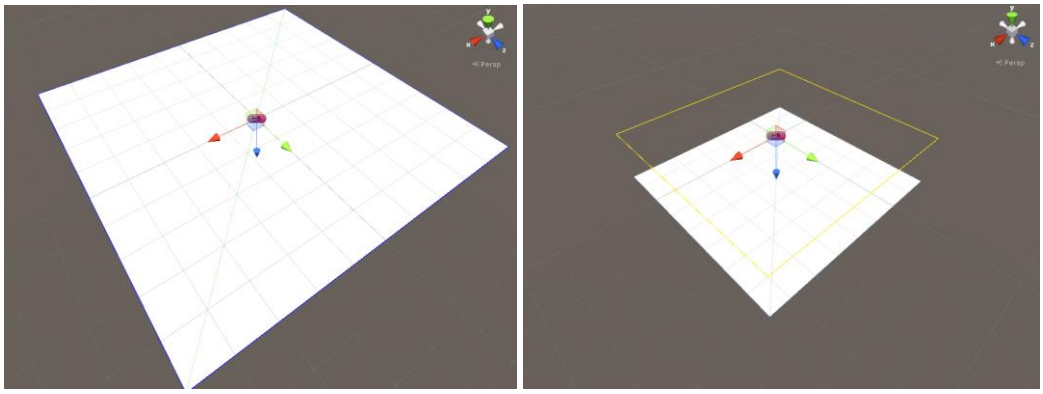

Fig.7 Quad primitive as a Floor

This is not what we want. To resolve this problem use Subdivide menu item.

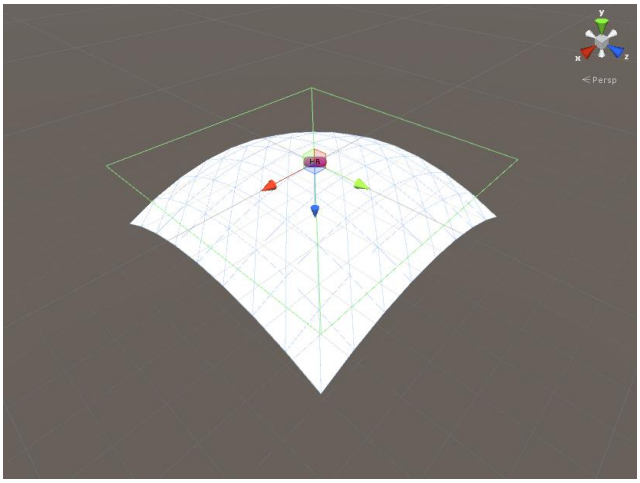

Fig.8 Subdivided quad

Curvature, Flatten, HorizonOffset

To control horizon bending you can use following properties of HB script:

1. Curvature parameter controls degree of steepness of parabola. *If set to 1 then at distance of ~63 units there will be ~1 unit offset*
2. Flatten parameter controls size of flat area.
3. HorzionXOffset, HorizonYOffset and HorizonZOffset parameters is used to control offset of the vertex of parabola

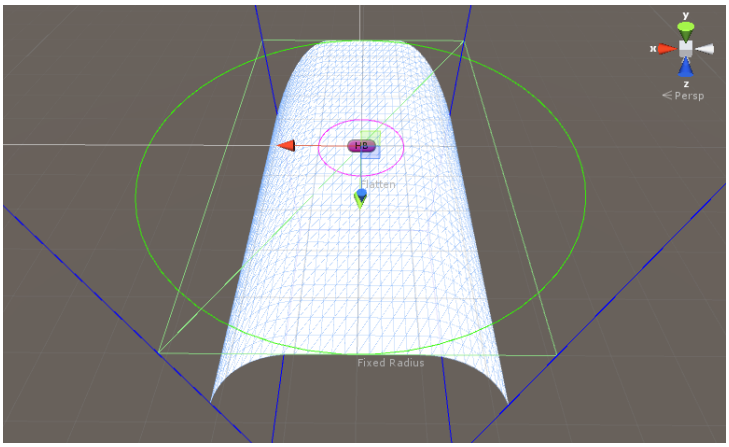

Fig.9 Curvature 60, Flatten 10

Objects Disappear

Horizon bend may cause undesired frustum culling side effect.

Here is the example:

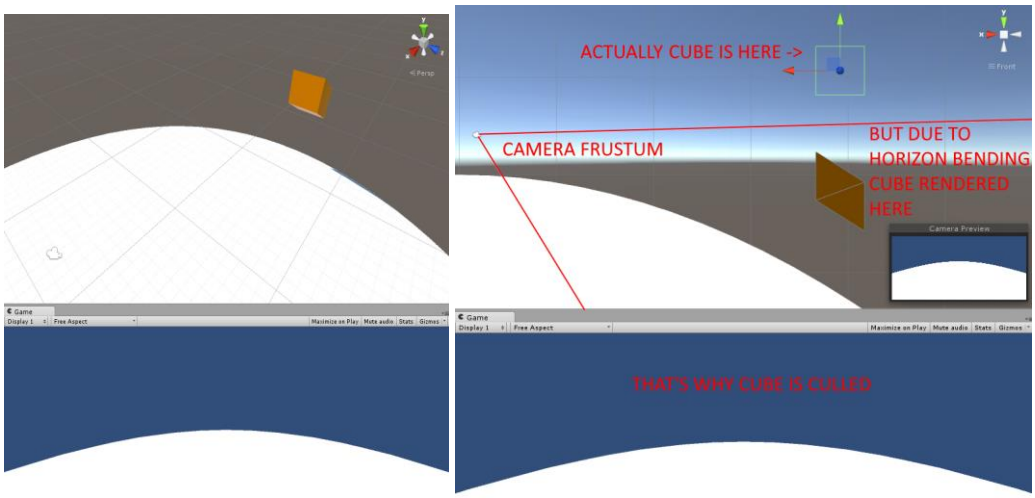

Fig.10 Undesired frustum culling

There is two solutions of this problem. First is to increase object bounds, second is to increase camera field of view (or orthographic size in case of orthographic camera)

Fixing Object Bounds

Bigger bounding boxes can prevent frustum culling. To increase size of bounding boxes select game object with name ‘HB’ in hierarchy. If you unable to find this game object click Tools->Horizon Bend->Apply. You will see following:

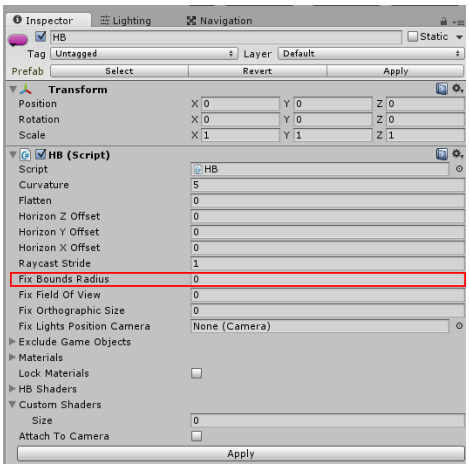

Fig.11 HB Script properties

Fix Bounds Radius is a radius of circle around camera with objects inside. These objects are garanteed to be visible and won’t accidently dissapear (if they remains inside of this circle). The greater this value the bigger bounding box will be. You should keep it as low as possible. When you click Apply button, bounds of all suitable objects will be fixed. To do this, HB will invoke HBFixBounds.FixBounds method of HBFixBounds component on suitable objects.

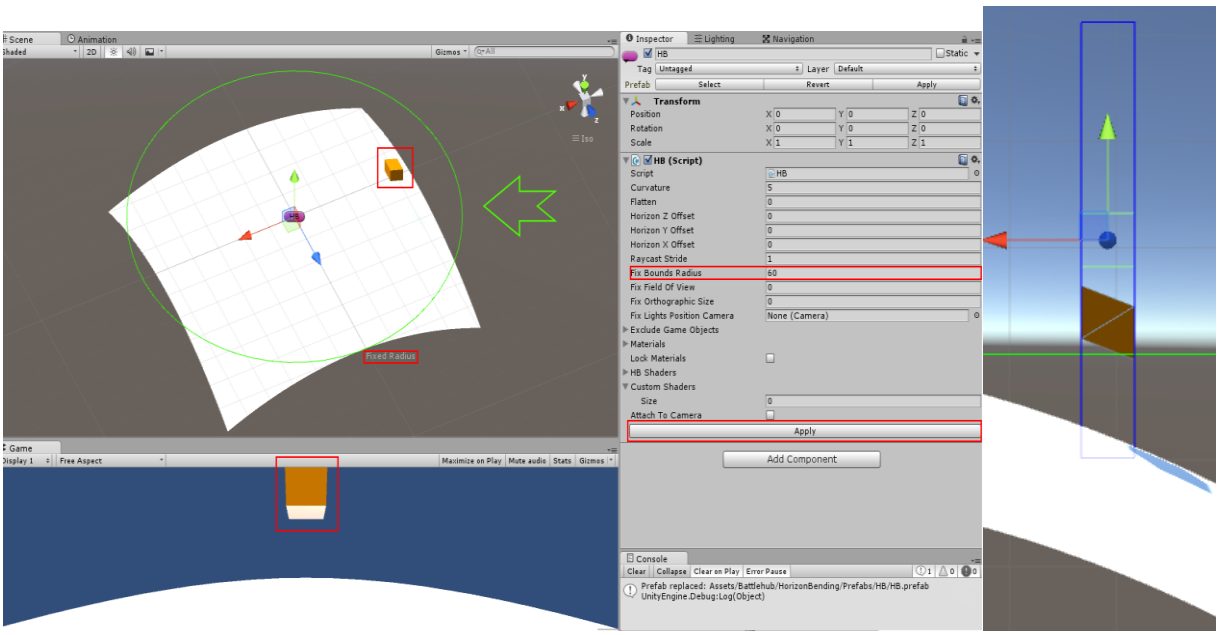

Fig.12 Increasing Fix Bounds Radius

HBFixBounds

If you for some reason want to fix bounding box manually do following: select object with HBFixBounds component, select Override Bounds, specify Bounds, select Lock and click FixBounds button.

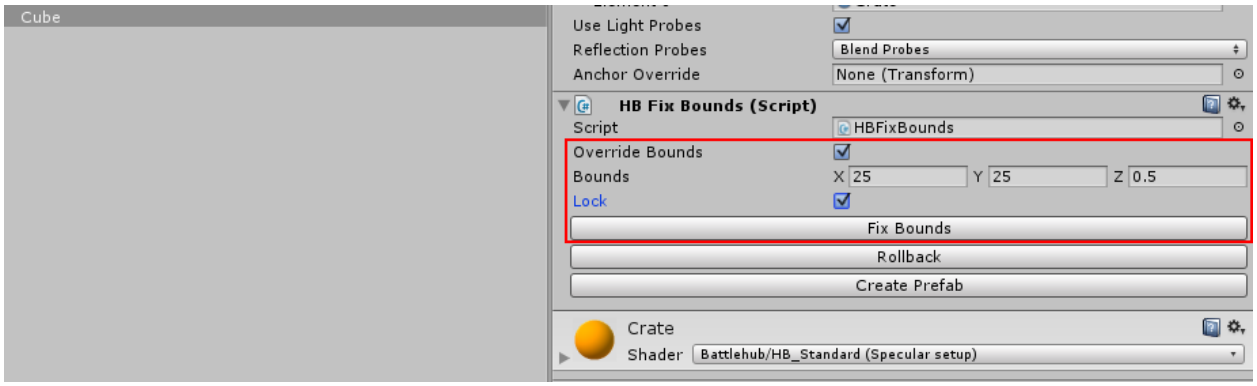

Fig.13 Overriding Bounds

HBFixBounds behave differently for static, non-static and object's with SkinnedMeshRenderer.

To modify bounds of non static objects HBFixBounds use Mesh.bounds property. To fix bounds of static objects HBFixBounds use slightly different approach. Each submesh of static game object is combined with so called "FixBoundsMesh". FixBoundsMesh contains eight zero area triangles. Yes, to trick static batcher we actually need triangles not dead end vertices. SkinnedMeshRenderer.localBounds fixed at runtime.

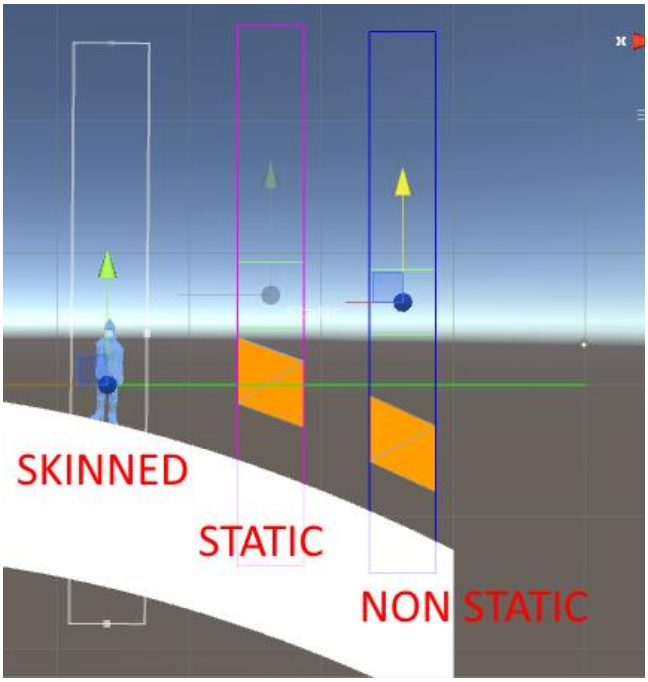

Fig.14 Defferent types of fixed bounding boxes

Particle System disappear

With particle system we need to take different approach. There is no accessible bounding box. We can edit pivot property of PS Renderer but it will cause big troubles with Stretched billboards. Alternatively we can scale up particle system and then undo scale in vertex program, but we believe this is not very good approach.

To prevent particle system frustum culling perspective camera field of view (or orthographc camera orthographic size) need to be changed.

Find ‘HB’ object and select it. Set Fix Field Of View property to positive value.

This value will be added to effective camera field of view during OnPreCull. Original field of view value will be restored OnPreRender. This functionality can be found in HBCamera.cs. ***NOTE:*** **HBCamera component automatically added to all cameras in scene. It is important to keep Fix Field Of View value as low as possible!**

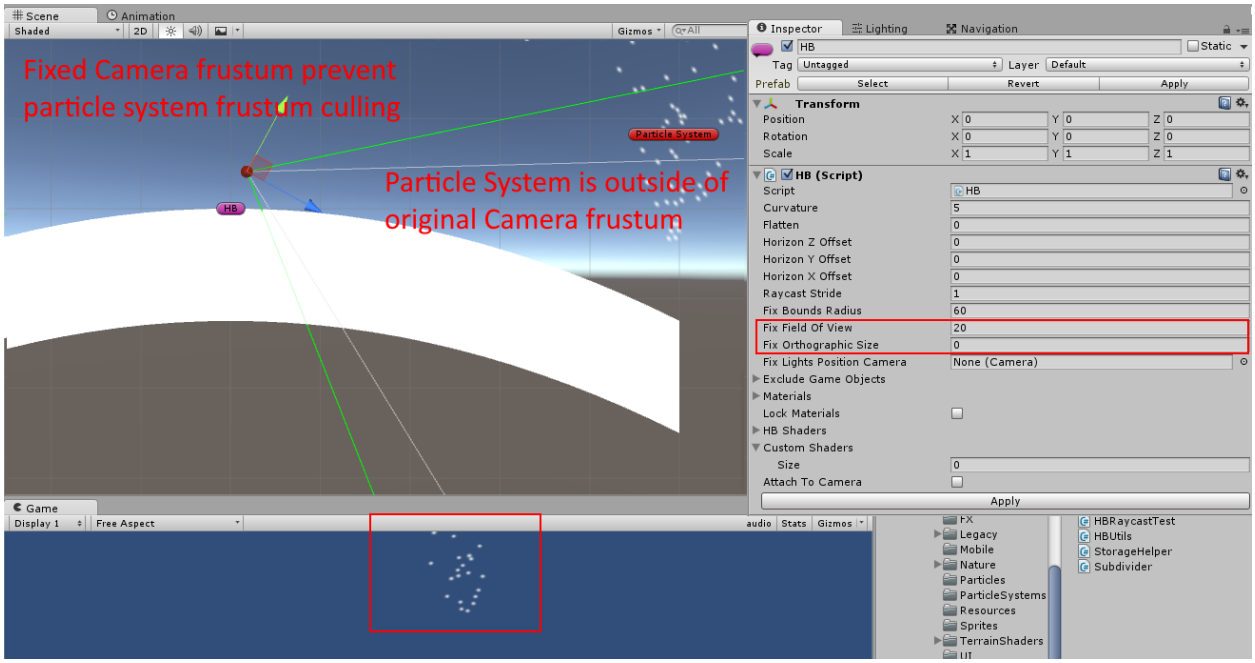

Fig.15 Fixing field of view

Terrain or grass disappear

Use the same approach as with particle system. **It is important to keep Fix Field Of View value as low as possible!**

Raycasting

In general case all raycasting code does not need to be changed. The exception is the code which create rays based on user input. For example you may use Camera.ScreenPointToRay to create picking ray, or use Camera.forward to create aiming ray. Look at the next picture.

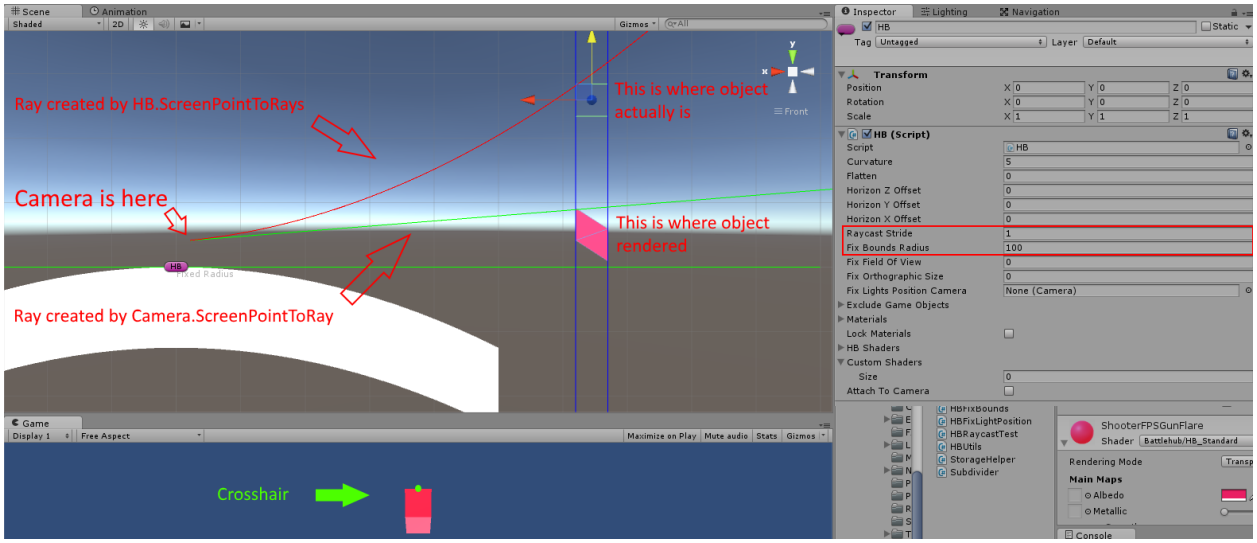

Fig.16 Raycasting

Ray created by Camera.ScreenPointToRay won't work because it goes through point where object is rendered but not actually located. Object collider is not affected by horizon bend and remains in different place. Physics.Raycast won't hit any objects.

To resolve this problem use HB.ScreenPointToRays and HB.Raycast methods. Both methods are controlled by HB.RaycastStride property and HB.FixBoundsRadius property of HB script. FixBoundsRadius property specifies radius inside of which raycasing is performed. RaycastStride specifies how precise raycasting will be (Larger values -> smaller precision. **Keep this value as large as possible**)

Here is the sample code from RaycastTest.cs file:

```
if(Input.GetMouseButtonDown(0))
{
        HB.DebugScreenPointsToRay(Camera.main);
        Ray[] rays;
        float[] maxDistances;
        HB.ScreenPointToRays(Camera.main, out rays, out maxDistances);

        RaycastHit hitInfo;
        if(HB.Raycast(rays, out hitInfo, maxDistances))
        {
                //Transform point to horizonbending space
                hitInfo.point -= HB.GetOffset(hitInfo.point, Camera.main.transform);

                Debug.Log(hitInfo.transform.gameObject.name);
        }
}
```

LineRenderer

In some cases problem with line renderers may arise. This problem is the same as with low poly meshes. Look at the Fig.17

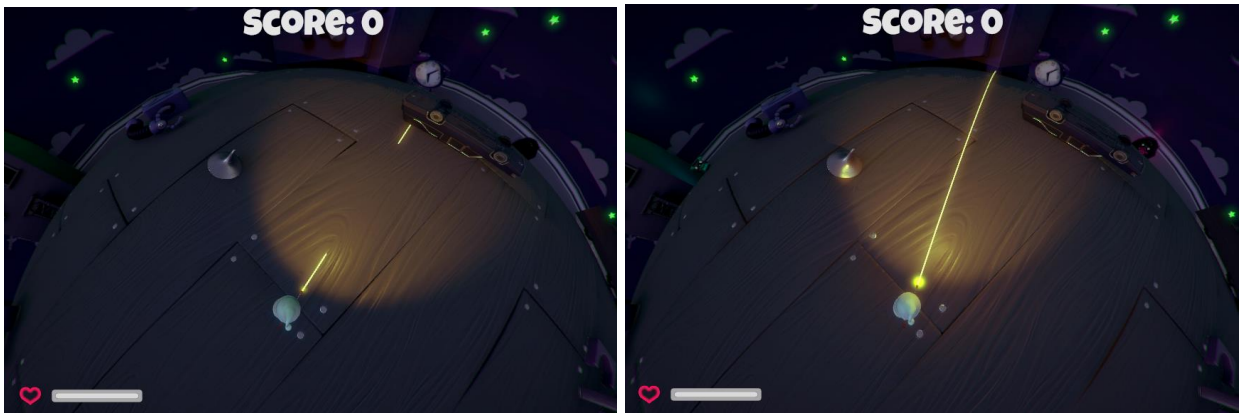

Fig.17 LineRenderer problem (left) and solution(right)

To fix problems use following method

```
HB.FixLineRenderer(lineRenderer, start, end);
```

HB.FixLineRenderer method use HB.RaycastStride property to control LineRenderer precision (higher value -> less precision)

Pointlights and Spotlights

If you want to move pointlight or spotlight to horizonbending space you should add HBFixLightPosition script to it. Then you must specify camera by setting HB.FixLightPositionCamera property. **Note: As you may notice with this script you can set positions of lights relative to single camera. This could be a serious limitation if you rendering using more than one camera at the same time.**

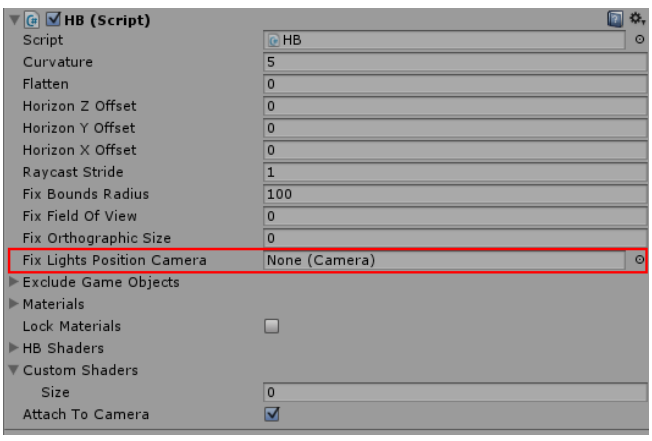

Fig.18 Fix Lights Position Camera property

Custom Shaders

Horizon Bend package has replacements for almost all standard Unity3D 5.3.0 shaders (including legacy, terrain, nature, particles, etc). However this is common situation when your project has it's own custom shaders.

To integrate your shader to HB you should do the following:

1. Open *.shader file and determine type of shader. http://docs.unity3d.com/Manual/SL-Reference.html
2. If your shader is fixed function shader you have to rewrite it…
3. If your shader is surface shader or fragment/vertex shader you should insert `#include "Assets/Battlehub/HorizonBending/Shaders/CGIncludes/HB_Core.cginc"` between CGPROGRAM and ENDCG
4. If your shader is surface find `#pragma surface` and add `vertex:hb_vert` to the end of the line
5. If your shader is fragment/vertex insert `HB(v.vertex)` into beginning of vertex program (if you have float4 posWorld variable defined use `HORIZON_BEND(v.vertex, posWorld)`)
6. Add your shader to CustomShaders array of HB script
7. Update HB.prefab located in Assets\Battlehub\HorizonBending\Prefabs\HB folder

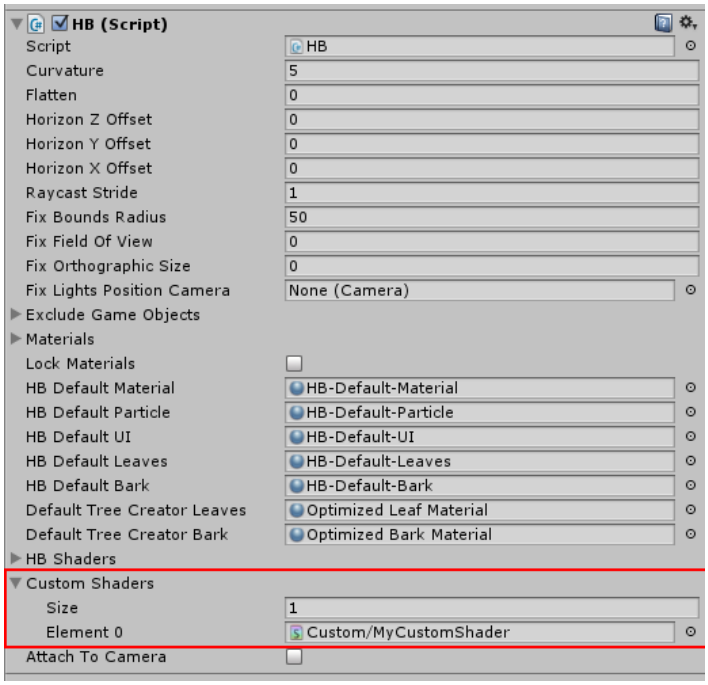

Fig.19 Custom Shaders Array

Lock Materials

If you want to restrict access to certain materials then remove them from Materials array of HB script and select LockMaterials. Then click Apply.

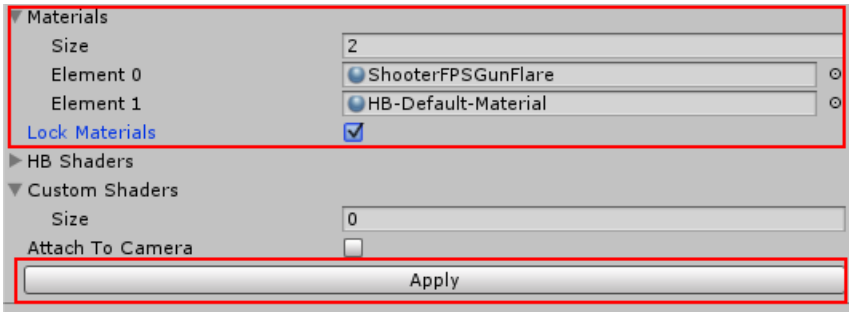

Fig.20 Lock Materials

Exclude GameObjects

You may want to prohobit modifications of some game objects by HB script. Just drag them into HB.ExcludeGameObjects array

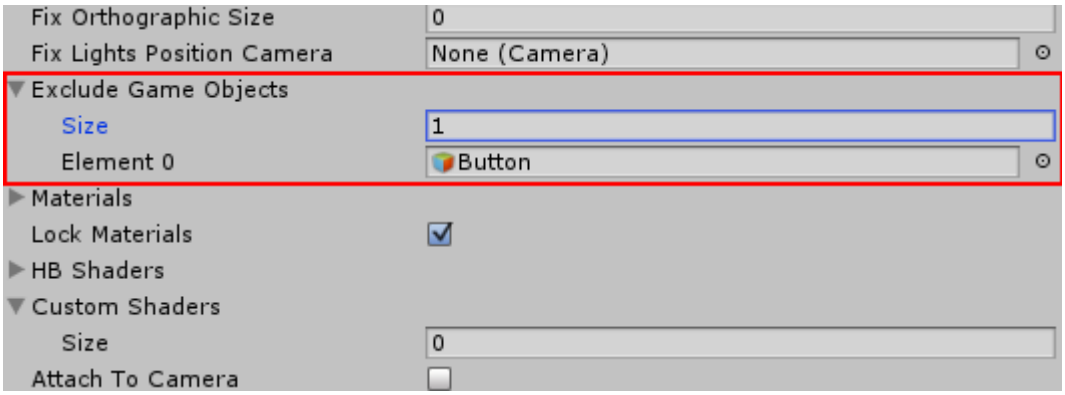

Fig.21 Exclude Game Objects

Attach To Camera

If selected then rendering behavior is the same as in the final build. Unselected state is useful for debugging and navigation in sceneview**. NOTE: AttachToCamera is set to true atomatically when you hit play. To prevent this find and comment following line in HB script** `//AttachToCamera = Application.isPlaying;`

HB Script

static class in Battlehub.HorizonBending

Description

HB class implements main functionality of a package (horizon bending and material/object management)

Static Functions

Apply Horizon Bending Parameters

```
1) public static void ApplyCurvature(float curvature)
2) public static void ApplyFlatten(float flatten)
3) public static void ApplyHorizonOffset(float horizonX, float horizonY, float horizonZ)
4) public static void ApplyAll(float curvature, float flatten, float horizonX, float horizonY,
   float horizonZ)
```

Change Horizon Bending Parameters using delta

```
5) public static void ChangeCurvature(float delta)
6) public static void ChangeFlatten(float delta)
7) public static void ChangeHorizonOffset(float deltaX, float deltaY, float deltaZ)
```

Make Bounding Boxes larger

```
8) public static void FixSkinned(SkinnedMeshRenderer skinned, Vector3 extents)
9) public static void FixSkinned(SkinnedMeshRenderer skinned, float fixBoundsRadius)
10) public static void FixBounds(MeshFilter meshFilter, Vector3 extents)
11) public static void FixBounds(MeshFilter meshFilter, float fixBoundsRadius)
12) public static void FixMesh(MeshFilter meshFilter, Vector3 extents)
13) public static void FixMesh(MeshFilter meshFilter, float fixBoundsRadius)
```

Increase vertices count of lines

```
14) public static void FixLineRenderer(LineRenderer lineRenderer, NavMeshPath path)
15) public static void FixLineRenderer(LineRenderer lineRenderer, Vector3 start, Vector3 end)
```

Get Settings

```
16) public static HBSettings GetSettings(bool attachToCameraInEditor)
17) public static HBSettings GetSettings()
```

Get vertex offset caused by HB at specified position relative to camera

```
18) public static Vector3 GetOffset(Vector3 atPosition, Transform cameraTransform)
```

Equivalent of Camera.ScreenPointToRay

```
19) public static void ScreenPointToRays(Camera camera, out Ray[] rays, out float[] maxDistances)
```

Convert camera position and camera forward to array of rays

```
20) public static void CameraToRays(Transform camerTransform, out Ray[] rays, out float[]
    maxDistances)
```

Raycast using results of ScreenPointsToRays or CameraToRays function

```
21) public static bool Raycast(Ray[] rays, out RaycastHit hitInfo, float[] maxDistances, int
    layerMask = 0x7FFFFFFF)
```

Raycast all using results of ScreenPointsToRays or CameraToRays function

```
22) public static List<RaycastHit> RaycastAll(Ray[] rays, float[] maxDistances, int layerMask)
```

Transform RaycastHit.point to HorizonBending space

```
23) public static RaycastHit FixRaycastHit(RaycastHit hit, Transform cameraTransform, Vector3
    rayOrigin)
```

Transform RaycastHit.point to HorizonBending space and fix distance

```
24) public static RaycastHit FixRaycastHitDistance(RaycastHit hit, Vector3 rayOrigin)
```

Supported Shaders

1) Standard
2) Standard Specular
3) Legacy (Opaque, Lightmapped, Reflective, SelfIllumin, Transparent, Cutout)
4) Particles and Particle Systems
5) Mobile (including Particles)
6) Nature (SpeedTree, SoftOcclusion, TreeCreator)
7) Terrain (Diffuse, Specular, Standard, BillboardTree, Grass)
8) Glass Refraction
9) Projectors
10) Tesselation Shaders
11) Toon Shading
12) Unlit
13) Sprites
14) Water
15) UI (Default and DefaultFont only)
16) Several custom shaders in HBCustomShaders.unitypackage

Limitations and Issues

Battlehub/Legacy Shaders/HB_VertexLit **no specular color**

Battlehub/Legacy Shaders/Transparent/HB_VertexLit **no specular color**

Battlehub/Legacy Shaders/Transparent/Cutout/HB_VertexLit **no specular color**

Battlehub/Legacy Shaders/Self-Illumin/HB_VertexLit **no emission**

Battlehub/Legacy Shaders/Ligthmapped/HB_VertexLit **no specular color**

Reflection in water shaders may work wrong

HB.FixFieldOfView positive value with forward rendering path may cause point and spotlight clipping (to address this issue light range should be increased)

HBFixLightPosition script can set positions of lights relative to **single camera.** It could be a serious limitation if you rendering using more than one camera at the same time.

Support

Thank you for your time, review and support.

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

I will be much appreciate if you could help me to edit this document and fix grammar and other mistakes.