Curved World

Thank you for your purchase of **Curved World**.

Please consider writing a review or just rate the asset:

https://www.assetstore.unity3d.com/en/#!/content/26165

For any question or help use forum:

http://forum.unity3d.com/threads/curved-world-2.344041/

What is Curved World and how it works

Curved World is a vertex transformation shader for creating various shader bending effects.
It is like displace or heighmap shader where texture is used to adjust mesh vertex position, but instead of a texture Curved World uses mesh vertex world-space position to calculate how it will be transformed.

Being a shader Curved World does not modify real mesh it renders. If for example mesh is 'flat' before using Curved World shader it still will be 'flat' after, just rendered differently. Because of it physics, animations, path finding and other game features are not effect and do not need any modifications.

If object needs to be moved from position A to position B along path C, after using Curved World shader everything will be the same.

Real scene

Rendered

Let's have a quick look on an example to understand core features of the Curved World.

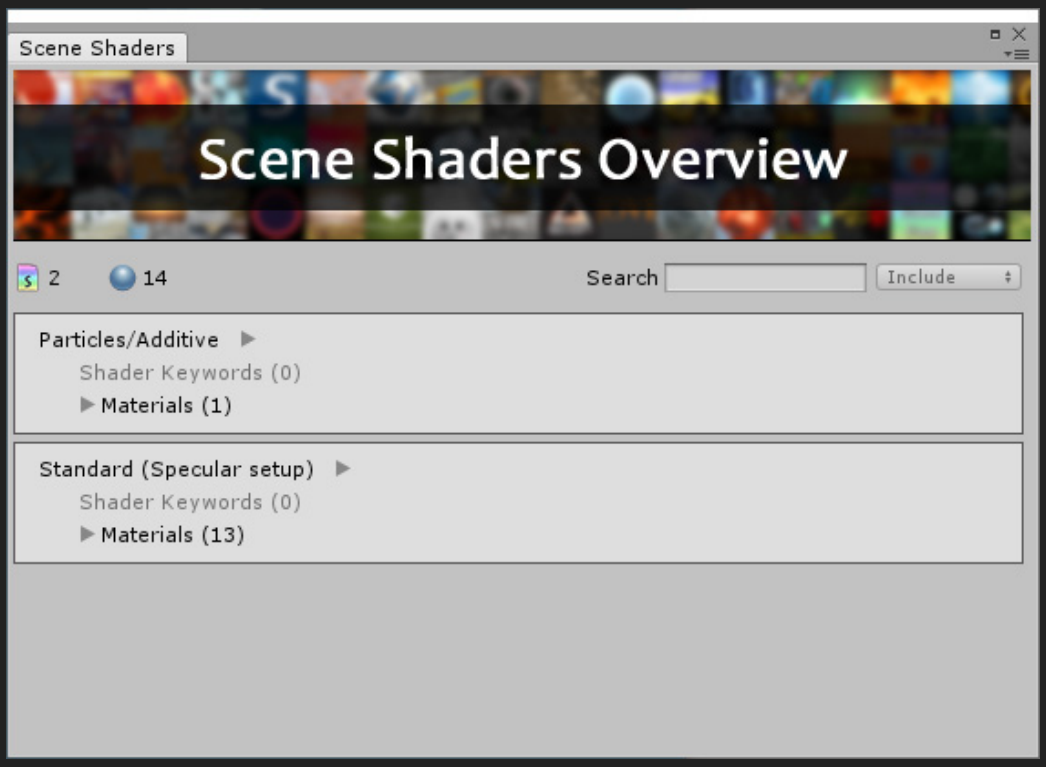
Open Tutorial example scene and run it.
It is a simple runner type scene. Objects are spawned at some distance from the player, moved along path (Z axis in this case) and are destroyed behind the camera. Nothing related to the Curved World.

Unleash Curved World in three easy steps:

1) Open Menu -> Window -> VacuumShaders -> Curved World Settings window and make sure **Classic Runner** bend type is active. If not, select it and push Update Shaders button. Close window.

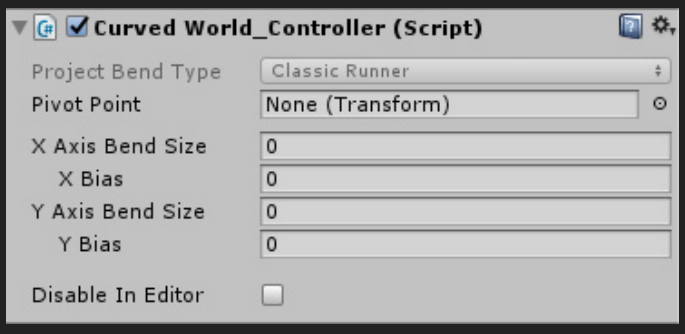
2) As bend effect is rendered only by using Curved World shaders, scene material shaders must be changed to the proper ones.
Currently scene meshes use Unity built-in Standard and Particle shaders. Curved World package contains exactly the same shaders but with integrated vertex transformation. We need to use them.

To make shader swap easier scene contains Materials gameobject with reference to all used materials. Change their shaders from Unity built-in Standard to VacuumShaders -> Curved World -> Standard and Particle shaders.
Or use editor helper tool from Menu -> Windows -> VacuumShaders -> Scene Shaders Overview.



3) Shaders with integrated Curved World have no special parameters inside material editor to alter bend effect, instead everything is controlled just from one CurvedWorld_Controller script that updates all required parameters globally for all shaders.

Assign CurvedWorld_Controller script to the gameobject with the same name already in the scene.



If run scene now it still will be without bending effect.
We just need to change Bend Size parameters. Try -0.6 and 4.0 for X and Y Axis Bend Size accordingly.
Now world is curved.

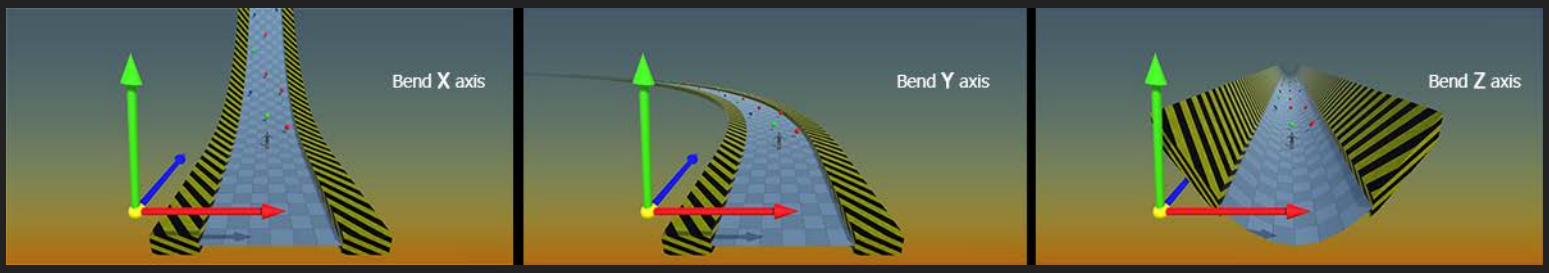
Curved World Off Curved World On

To summarize how to use Curved World.
1) Set desired bend type from Settings window and recompile shaders (only shaders inside Curved World package will be recompiled).
2) Use Curved World shaders (package comes with many different shaders, they can be imported from Settings window) or use custom ones (check CustomShaders.pdf file).
3) Update bending parameters from CurvedWorld_Controller script.

That's all.

Bend Types

Curved World offers several bending effects. Each one bends scene meshes along world **X**, **Y** or **Z** axis in a particular way and have simple requirements toward scene design and object placement.

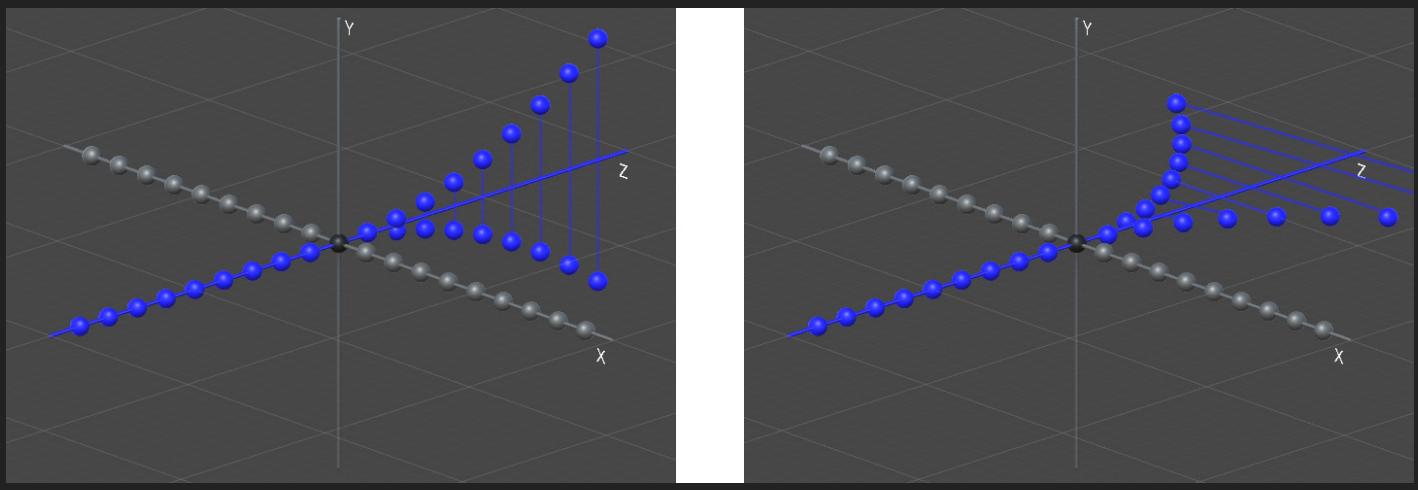


Note, bend effects are based on parabola equation and vertex bending strength is proportional to the distance from pivot point. The farther vertex is from the pivot point the more bending strength it has.

Pivot point position can be controlled from CurvedWorld_Controller script. If it is not defined then world (0, 0, 0) coordinate is used.

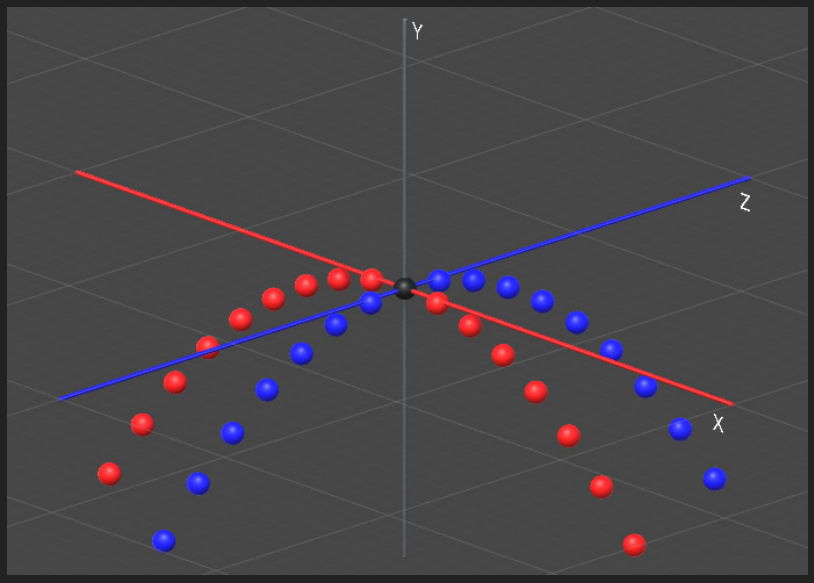
- **Classic Runner** - Scene objects must be located along Z axis (like in runner games).
  Vertices can be moved along X axis (for up/down effect) and Y axis (for left/right rotating effect).
  Vertices behind pivot point do not bend.

  Check 1. Classic Runner example scene.



- **Little Planet** – Scene objects are located on XZ plane (common flat/horizontal design). Vertices can be moved along X and Z axis separately for creating Cylindrical Rolloff effect or together for Spherical World effect.

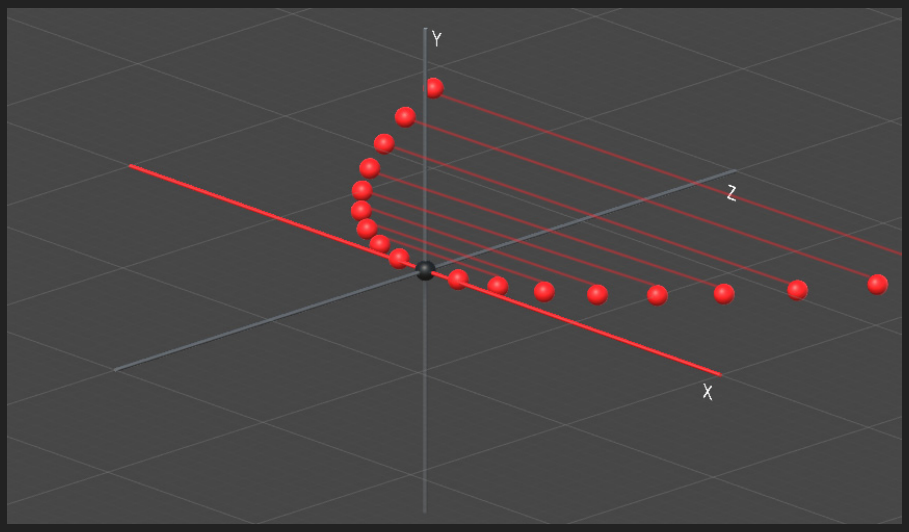
  Check 2. Little Planet (Nightmare) example scene.
  It is Unity Survival Shooter Tutorial from the Asset Store, but with Curved World shaders.



- **Universal** – Similar bending effect as **Little Planet** but with all three **X**, **Y** and **Z** axis bend controls.

- **Cylindrical Tower –** Scene object should be located on XY plane and bending along Y axis will create effect of cylinder.

  Check 4. Cylindrical Tower example scene.



- **Perspective 2D** - Similar to **Cylindrical Tower** type, but only for 2D sprite projects.
  Bending effect depends on camera position and rotation.
  Pivot point always is camera's screen center point.
  Active camera type must be Perspective, not Orthographic.

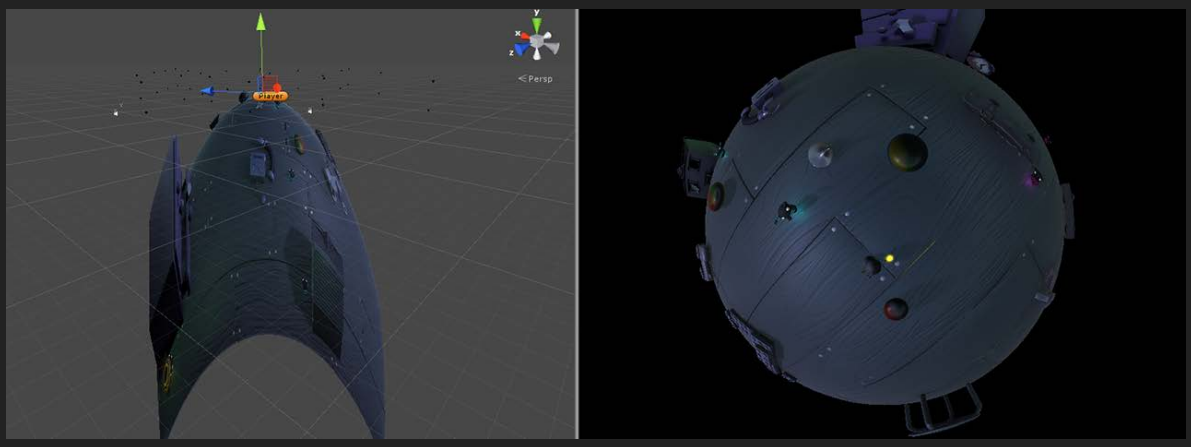
  Check 5. Perspective 2D example scene.

Bend effects created by shaders are designed to be observed only from specific points, from different points of view they are completely useless.

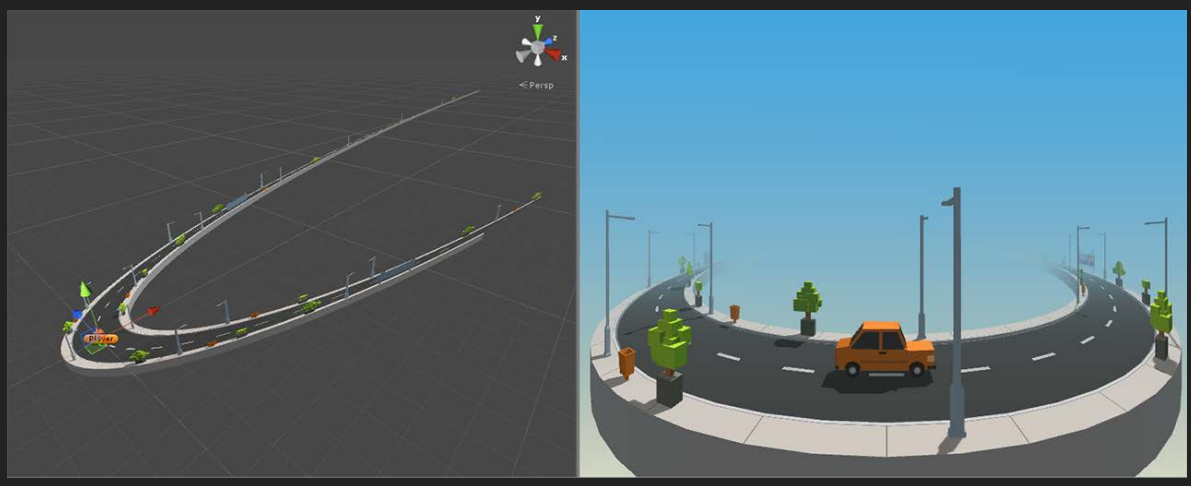
Here are examples how various bend effects look from side (wrong) and correct view positions:

Classic Runner

Little Planet



Cylindrical Tower



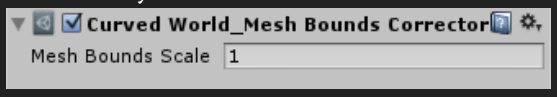
Mesh disappearing

Curved World bends mesh only if it is visible to a camera. If mesh goes beyond camera view frustum, it is culled and not rendered. In some cases this may create mesh disappearing problem.

Curved World has 2 solution for solving this problem.

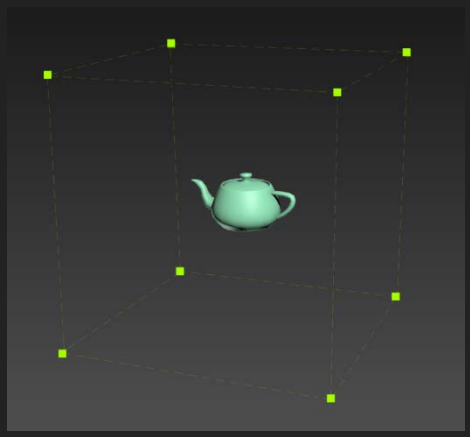
1. **Eagle Eye** script – Overrides camera's field of view before culling the scene, and restores it before rendering. Works only if attached to the active camera.

   Keep its value as low as possible.
2. **Mesh Bounds Corrector** script – Overrides individual mesh renderers bound component to scale it and make visible to camera even if it is outside its view frustum.
   Script also is necessary if mesh is not visible to dynamic light source (it has its own field of view) and if mesh is not visible excludes it from shadow receiving/casting pass.
   Works only if attached to the active non static gameobject with Renderer component.



   Keep its value as low as possible.

3. Manual user solution for static objects - It is necessary to add 8 'dead' vertexes to the mesh manually using any of 3D modeling software and inside Unity in mesh import settings disable Optimize Mesh check box (Unity removes not used vertices).



Scaling mesh's bounding box in 3Ds Max.
8 vertices will be added to the mesh.
8 vertex add only 64 bytes to the mesh and don't participate into mesh rendering, as they have no triangles attached.