CurvedUI 3.3 Documentation

**Preface & Contact**

👋Hello there! Thank you for choosing CurvedUI. If you run into any problems or want to offer feedback, reach me at **curvedui@chisely.com**. Have fun building curved interfaces!

* Daniel

[Preface & Contact](https://docs.google.com/document/d/1hJi6Q56f0D9yj0f1zG15-4XRnFZkREK9n8TF9Rnbn1w/edit#heading=h.hckiuhumfh1u)

[Getting started](https://docs.google.com/document/d/1hJi6Q56f0D9yj0f1zG15-4XRnFZkREK9n8TF9Rnbn1w/edit#heading=h.cggnk2c1kz09)

**First Steps - CurvedUI in a New Scene**

[**SteamVR - Getting Started**](https://docs.google.com/document/d/1hJi6Q56f0D9yj0f1zG15-4XRnFZkREK9n8TF9Rnbn1w/edit#heading=h.bybnwzj7via9)

[**OculusVR - Getting Started**](https://docs.google.com/document/d/1hJi6Q56f0D9yj0f1zG15-4XRnFZkREK9n8TF9Rnbn1w/edit#heading=h.y76yq6b0cayr)

[**Unity XR Toolkit & New Input System**](https://docs.google.com/document/d/1hJi6Q56f0D9yj0f1zG15-4XRnFZkREK9n8TF9Rnbn1w/edit#heading=h.80zd6zksh514)

[**VRTK - Getting Started**](https://docs.google.com/document/d/1hJi6Q56f0D9yj0f1zG15-4XRnFZkREK9n8TF9Rnbn1w/edit#heading=h.qenrlfadur7n)

[**MRTK - Getting Started**](https://docs.google.com/document/d/1hJi6Q56f0D9yj0f1zG15-4XRnFZkREK9n8TF9Rnbn1w/edit#heading=h.n1wkar37bjcy)

[**Other Platforms and SDKs - Custom Ray**](https://docs.google.com/document/d/1hJi6Q56f0D9yj0f1zG15-4XRnFZkREK9n8TF9Rnbn1w/edit#heading=h.g1ctt66wg0ez)

[Technical Stuff - How does it work?](https://docs.google.com/document/d/1hJi6Q56f0D9yj0f1zG15-4XRnFZkREK9n8TF9Rnbn1w/edit#heading=h.71wxkz1t8ose)

[FAQ & Common Problems](https://docs.google.com/document/d/1hJi6Q56f0D9yj0f1zG15-4XRnFZkREK9n8TF9Rnbn1w/edit#heading=h.9rerap6jh1ju)

[Can't select anything until the trigger is pressed.](https://docs.google.com/document/d/1hJi6Q56f0D9yj0f1zG15-4XRnFZkREK9n8TF9Rnbn1w/edit#heading=h.74tcneyb3xwo)

[Some buttons do not work!](https://docs.google.com/document/d/1hJi6Q56f0D9yj0f1zG15-4XRnFZkREK9n8TF9Rnbn1w/edit#heading=h.hndqfdmpwtd9)

[Elements are not curved if I instantiate them. What do?](https://docs.google.com/document/d/1hJi6Q56f0D9yj0f1zG15-4XRnFZkREK9n8TF9Rnbn1w/edit#heading=h.7f3yp68gfhzl)

[Button interactions are not accurate.](https://docs.google.com/document/d/1hJi6Q56f0D9yj0f1zG15-4XRnFZkREK9n8TF9Rnbn1w/edit#heading=h.92ev5610ayu0)

[Moving An Object To Another Canvas.](https://docs.google.com/document/d/1hJi6Q56f0D9yj0f1zG15-4XRnFZkREK9n8TF9Rnbn1w/edit#heading=h.xiva60z5sqae)

[Nothing happens when I change objects by code.](https://docs.google.com/document/d/1hJi6Q56f0D9yj0f1zG15-4XRnFZkREK9n8TF9Rnbn1w/edit#heading=h.1bjc9yarcoqw)

[Enable button is stuck on “Please Wait”](https://docs.google.com/document/d/1hJi6Q56f0D9yj0f1zG15-4XRnFZkREK9n8TF9Rnbn1w/edit#heading=h.rymnil4qfw9l)

[“The referenced script on this Behaviour is missing” Error](https://docs.google.com/document/d/1hJi6Q56f0D9yj0f1zG15-4XRnFZkREK9n8TF9Rnbn1w/edit#heading=h.wkbm0rfzs90a)

[Using SteamVR and native Oculus SDK in the same build](https://docs.google.com/document/d/1hJi6Q56f0D9yj0f1zG15-4XRnFZkREK9n8TF9Rnbn1w/edit#heading=h.dnjt78n3olkj)

[Scripting API](https://docs.google.com/document/d/1hJi6Q56f0D9yj0f1zG15-4XRnFZkREK9n8TF9Rnbn1w/edit#heading=h.6bt574hnerhu)

[CurvedUISettings](https://docs.google.com/document/d/1hJi6Q56f0D9yj0f1zG15-4XRnFZkREK9n8TF9Rnbn1w/edit#heading=h.dkjmzc3qpxf2)

[CurvedUIRaycaster](https://docs.google.com/document/d/1hJi6Q56f0D9yj0f1zG15-4XRnFZkREK9n8TF9Rnbn1w/edit#heading=h.qmzawc4foax1)

[CurvedUIInputModule](https://docs.google.com/document/d/1hJi6Q56f0D9yj0f1zG15-4XRnFZkREK9n8TF9Rnbn1w/edit#heading=h.wcltwv9q0u6o)

[CurvedUIVertexEffect](https://docs.google.com/document/d/1hJi6Q56f0D9yj0f1zG15-4XRnFZkREK9n8TF9Rnbn1w/edit#heading=h.sbbvdskdxfi5)

[Experimental Features](https://docs.google.com/document/d/1hJi6Q56f0D9yj0f1zG15-4XRnFZkREK9n8TF9Rnbn1w/edit#heading=h.dvzjg9qqnya2)

[TextMeshPro Support](https://docs.google.com/document/d/1hJi6Q56f0D9yj0f1zG15-4XRnFZkREK9n8TF9Rnbn1w/edit#heading=h.ywu0q5d4tc2n)

**Getting started**

**First Steps - CurvedUI in a New Scene**

1. Create a Canvas and switch it to World Space mode.
2. Add **CurvedUISettings** component to Canvas gameobject.
3. Set the desired angle in the **CurvedUISettings** component.
4. Chose a control method on **CurvedUISettings** and configure it.
5. Start the game.

You can change the angle at runtime using the Angleproperty in the **CurvedUISettings** component. CurvedUI will automatically update child gameobjects. Additionally, you can change more advanced settings in **CurvedUISettings** component like **Shape**, **Control Method** or whether the canvas should be **interactable** at all

**SteamVR - Getting Started**

1. Import **SteamVR plugin** to your project.
2. Add **CameraRig** prefab or **Player** prefab from SteamVR to the scene.
3. Add **CurvedUISettings** to your canvas.
4. Set CurvedUISettings' **Control Method** to STEAMVR
5. Press the "**Enable**" button in CurvedUISettings
6. (Optional) 5. Drop **CurvedUILaserPointer** prefab anywhere on the scene. It will be your laser guide.

You can also open the “SteamVR Starter” demo scene. It has been set up before for you. You only need to do #4 to make it work.

**OculusVR - Getting Started**

1. Add **OVRCameraRig** prefab from **Oculus Utilities** package to your scene.
2. Add **CurvedUISettings** to your canvas.
3. Set CurvedUISettings' **Control Method** to OCULUSVR
4. Press the "**Enable**" button in CurvedUISettings
5. (Optional) Import **OvrAvatar** package and add LocalAvatar prefab to your scene. This will let you see the controllers’ models.
6. (Optional) Place **CurvedUILaserPointer** prefab in the RightHandAnchor of oculus rig. It will be your laser pointer.

You can also open the “OculusVR Starter” demo scene. It has been set up before for you. You only need to do #4 to make it work.

|  |
| --- |
| 👉 Oculus **Hand Tracking and Interaction** support is now in the alpha stage. If you want access to this experimental version, please reach out to me at [curvedui@chisely.com](mailto:curvedui@chisely.com) with your Asset Store order number. |

**Unity XR Toolkit & New Input System**

1. **Set up Unity XR** (Jesus Christ it takes forever - what were they thinking?)
   1. In Package Manager
      1. Install XR Plugin Management
      2. Install XR Interaction Toolkit 1.0.0 or later.
      3. Install New Input Package (if not installed already)
      4. Install "Default Input Actions" under XR Interaction Toolkit/Samples
   2. In Project Settings / XR Plugin Management
      1. Enable "Initialize XR on Startup"
      2. Enable SDKs as needed per platform.
   3. Set up Controller Actions
      1. Add InputActionManager component somewhere on your scene
      2. Assign the XRI Default Input Actions as Action Asset on InputActionManager
      3. Set up Right XRController in XR Rig
         1. Check "use reference" next to Position Action
         2. Under Reference. select "XRI RightHand/Position"
         3. Check "use reference" next to Rotation Action
         4. Under Reference, select "XRI RightHand/Rotation"
         5. Check "use reference" next to UI Press Action
         6. Under Reference, select "XR Right Hand UI Press"
      4. Set up Left XRController in XR Rig
         1. Check "use reference" next to Position Action
         2. Under Reference. select "XRI LeftHand/Position"
         3. Check "use reference" next to Rotation Action
         4. Under Reference, select "XRI LeftHand/Rotation"
         5. Check "use reference" next to UI Press Action
         6. Under Reference, select "XR Left Hand UI Press"
2. **Test your XR Setup**
   1. Create a XR Rig > UI Canvas
   2. Make it a scale of 0.004 and position it in front of the XR Rig
   3. Add a button to it
   4. Change button highlight state to red and pressed state to green
   5. Press Play and see if you can click the button and get it to turn green
3. **Add CurvedUI**
   1. Import CurvedUI package to your project.
   2. Add CurvedUISettings to Canvas
   3. Select UNITY\_XR
   4. Press [ Enable ]
   5. Assign Controllers references under Control Method dropdown.
4. **Test your scene!**

**VRTK - Getting Started**

To use CurvedUI with VRTK, simply **remove (or disable)** all VRTK\_VRInputModule, VRTK\_VRInputModule, VRTK\_UIGraphicRaycaster, VRTK\_UICanvas and VRTK\_EventSystem scripts on the scene. These are used only for UI interaction - CurvedUI handles that stuff by itself. All other VRTK components will work as expected.

If you're using the VRTK pointer, make sure you set the canvas on a **separate layer** and check **Raycast My Layer Only** option on the **CurvedUISettings** component.

**Important**:

If you’re running a multi-platform VRTK setup and getting null reference errors from CurvedUI, there’s a simple fix. Using Unity Inspector, assign your CameraRig’s SteamVRControllerManager component to a field called “Steam VR ControllerManager” on the **CurvedUIInputModule**. The CurvedUIInputModule component is located on the EventSystem gameobject on the scene.

**MRTK - Getting Started**

CurvedUI x MRTK integration is not trivial and it does not come out of the box

However, it can be reliably achieved with some code modifications. The important thing is to let CurvedUI handle the UI and use MRTK for other stuff.

MRTK is pretty weirdly structured - the entire SDK throws errors and stops functioning completely if there is any other Input Module than its own at the start. However, the MRTKInputModule isn't needed for the rest of MRTK functions to work.

The trick is to have a separate inactive EventSystem gameobject on the scene with CurvedUIEventSystem and CurvedUIInputModule. Activate it a second after the scene starts and it takes over the UI input. I've created a simple script that performs this task and also feeds MRTK controllers' position and button states to CurvedUIInputModule.

What's needed:

* Custom MRTKCustomRay script.
* Modified CurvedUIInputModule script that allows it to stay dormant for a while and not mess with MRTK input.

Reach out to me at [curvedui@chisely.com](mailto:curvedui@chisely.com) with your Asset Store order number and I’ll be happy to share those files with you and guide you through the process!

**Other Platforms and SDKs - Custom Ray**

You can use any third-party physical or virtual remotes as CurvedUI input. To do this, set your CurvedUISettings to **Custom Ray** control method and place the following script somewhere on the scene: (Of course, you have to fill the variables there with your remote’s data)

void Update()

{

CurvedUIInputModule.CustomControllerRay = new Ray(yourRemoteTransform.position, yourRemoteTransform.forward);

CurvedUIInputModule.CustomControllerButtonState = isButtonOnYourRemotePressed;

}

You can also Add **ControllerLaserPointer** prefab as a child of your remote’s transform to give it a laser pointer.

**Technical Stuff - How does it work?**

CurvedUI works by bending the vertices of UI elements in Canvas’ local space. To achieve a nice, quality curve, a quad tessellation is performed first. The number of created quads depends on the desired angle and size of the UI object. Small UI objects may not get tessellated at all, while full-screen Panels will require a lot of quads to be drawn properly. CurvedUI also translates user input from InputModules on the scene to coordinates on curved canvas. Thanks to that, you can use a curved canvas just like you would a flat one, and don’t have to worry about coordinates in 3d space. This feat is achieved using a Raycast against a mesh collider created when you first start the scene.

**FAQ & Common Problems**

**The type or namespace name 'XRBaseController' could not be found**

This error shows up when CurvedUI is trying to find the **XR Interaction Toolkit** package, but it is not present in your project. CurvedUI requires this package to function, if you enabled its UNITY\_XR control method, or have imported the **New Input System** package from Unity registry.

You can find **XR Interaction Toolkit** package in the Package Manager in Unity editor. In some Unity versions, you may need to enable “Enable Pre-release Packages” in Project Settings / Package Manager before it shows up.

A **step-by-step guide** on how to configure CurvedUI with the new input system package and Unity XR here: [**Unity XR Toolkit & New Input System**](https://docs.google.com/document/d/10hNcvOMissNbGgjyFyV1MS7HwkXXE6270A6Ul8h8pnQ/edit#heading=h.80zd6zksh514)

**Some buttons do not work!**

Make sure all your interactive elements are inside the Canvas’ rectangle. You can see it as a white outline when you select the canvas gameobject.

For performance purposes, CurvedUI creates a collider that only covers objects inside the canvas’ rectangle.

**Elements are not curved if I instantiate them.**

When you add graphic elements to the curved canvas during runtime you have to add a **CurvedUIVertexEffect** component to them manually. You can do it with the following line:

YourNewObject.AddComponent<CurvedUIVertexEffect>();

Make sure you add the component **after** you make the element a child of the canvas. You can also make the CurvedUISettings component scan all its children and add necessary components by calling its **AddEffectToChildren()** function once.

**Button interactions are not accurate.**

Please make sure your interactive elements (buttons, sliders, dropdowns) are on z = 0 in relation to the canvas. To save performance, CurvedUI uses only one collider for the entire canvas. It is created at runtime and catches interactions at z position 0.

Make sure the proper camera is used for finding the controller/canvas interactions. If you have multiple cameras on the scene, make sure only one is tagged Main Camera. CurvedUI will find it automatically. Alternatively, manually assign your Main Camera to the Input Camera field on the Canvas component.

**Moving An Object To Another Canvas.**

When you move any object to a different canvas by changing its parent, you may find that its shape is not updated. This is because, for performance reasons, CurvedUI assumes that objects will not be moved between canvases. You can fix this by running the following code on the object after the move.

foreach (CurvedUIVertexEffect eff in  **YourMovedObject**.GetComponentsInChildren<CurvedUIVertexEffect>(true))

{

eff.FindParentSettings(true);

eff.SetDirty();

}

**Nothing happens when I change objects by code.**

CurvedUI looks for changes in Graphic components to know when a canvas object should be recalculated. Image, Text, RawImage are all examples of Graphic components. If you create a custom Graphic component, CurvedUI may have problems in finding out when a property has been changed by code. A change would be visible in the inspector, but not in the actual canvas object. You can then force CurvedUI object to redraw by using the following line:

YourObject.GetComponent<CurvedUIVertexEffect>().SetDirty();

Call it every time you change a property by code and it will work flawlessly.

**Enable button is stuck on “Please Wait”**

Open Unity’s **Build Settings** window and press the Enable button again. Some older unity versions do not allow changes to custom defines until the **Build Settings** window has been created (opened) at least once after opening Unity Editor

**The referenced script on this Behaviour is missing**

If you see this message on any of gameobjects in CurvedUI demo scenes, it most likely means something went **wrong with asset import**.

To fix this, open the Asset Store tab in Unity, **re-download** CurvedUI and click **Import** again.

**Using SteamVR and native Oculus SDK in the same build**

Word of advice - You should not do this if you plan on uploading your game to Oculus store. The build will be rejected if it contains any SteamVR code.

By manually adding "CURVEDUI\_TOUCH" and "CURVEDUI\_VIVE"  to your platform custom defines, you'll be able to switch between STEAMVR control method (based on SteamVR SDK) and OCULUSVR control method (Based on Oculus SDK) during runtime. Be sure to leave both rigs' gameobjects active on Start so CurvedUIInputModule can find all the necessary references on your scene.

See<https://docs.unity3d.com/Manual/PlatformDependentCompilation.html> for details on how to add to platform custom defines.

**Scripting API**

**CurvedUISettings**

Description

CurvedUISettings is the main component of CurvedUI and the only one you need to get started. Add it to a canvas you want to curve. It contains all the different settings you can change in CurvedUI. Automatically adds CurvedUIRaycaster to canvas at runtime to handle interactions.

Public

|  |  |
| --- | --- |
| **AddEffectToChildren** | Goes through the Canvas Hierarchy and adds CurvedUIVertexEffect to every gameobject that should be curved. Used when CurvedUISettings is first added to the canvas to prepare its objects. |
| **VertexPositionToCurvedCanvas** | Maps a world space position on a curved canvas (for example obtained from raycasting with canvas’ collider) to a local space position on the flat canvas. |
| **CanvasToCurvedCanvas** | Maps a position on a flat canvas (for example the position of a recttransform) to a world space position on a curved canvas. |
| **CanvasToCurvedCanvasNormal** | Returns a normal vector on a curved canvas from a flat canvas’ local space position. |
| **RaycastToCanvasSpace** | Tests a ray for a collision with the canvas and returns a collision point in flat canvas’ local space. Can be used for example to make a recttransform follow a laser pointer over the canvas. |
| **GetCyllinderRadiusInCanvasSpace** | Returns the radius of the curved canvas cylinder, expressed in Canvas' local space units. |
| **GetTesslationSize** | How big UI quads can get before they should be tessellated to look good on current canvas settings. Used by CurvedUIVertexEffect to determine how many quads need to be created for each graphic. |
| **BaseCircleSegments** | How many segments should the entire 360 deg. cylinder or sphere consist of. Used by CurvedUIVertexEffect |
| **Angle** | The measure of the horizontal arc of the Canvas. |
| **VerticalAngle** | Vertical angle of the canvas. Used in sphere shape and ring shape. |
| **Shape** | Current Shape of the canvas. |
| **Quality** | Multiplier used to determine how many segments a base curve of a shape has.  Default 1. Lower values greatly increase performance. Higher values give you a sharper curve. |
| **RingFill** | Fill of the ring in a ring shaped canvas. 0-1 |
| **SavedRadius** | Calculated radius of the curved canvas. Used By CurvedUIVertexEffect |
| **RingExternalDiameter** | External diameter of the ring shaped canvas, in canvas units |
| **RingFlipVertical** | Whether the center of the ring should be bottom or top of the canvas. |
| **PreserveAspect** | If enabled, CurvedUI will try to preserve the aspect ratio of the original canvas. |
| **Interactable** | Can the canvas be interacted with? |
| **BlocksRaycasts** | Will the canvas block raycasts. Settings this to false will destroy the canvas' collider. |
| **RaycastMyLayerOnly** | Should the raycaster take other layers into account to determine if canvas has been interacted with. |
| **SetAllChildrenDirty** | Forces all child CurvedUI objects to recalculate |
| **GetObjectsUnderScreenPos** | Returns all the canvas objects that are visible under given Screen Position. For example, using it with new Vector2(Screen.width, Screen.height) returns canvas gameobjects at the center of the screen. |
| **ForceUseBoxCollider** | Should the collider for this canvas always be created using more expensive box colliders? Default false. |

CurvedUISettings also contains shortcuts to following functions:

**CurvedUIRaycaster** functions:

Click, GetObjectsUnderPointer, GetObjectsUnderScreenPos, GetObjectsHitByRay, PointingAtCanvas

**CurvedUInputModule** functions:

ControlMethod, GazeUseTimedClick, GazeClickTimer, GazeClickTimerDelay, GazeTimerProgress

**CurvedUIRaycaster**

Description

Automatically added to any canvas with CruvedUISettings component on startup.

CurvedUIRaycaster handles translating interactions from any control method to canvas’ objects. It creates a curved collider and uses raycasts to determine the interaction position in world space which it then translates to a flat canvas position used by UI elements.

Public

|  |  |
| --- | --- |
| **RebuildCollider** | Forces the raycaster to rebuild its collider. Used by CurvedUISettings when changing canvas shape or angle. |
| **GetObjectsUnderPointer** | Returns gameobjects that are currently pointed at using any control method. |
| **GetObjectsUnderScreenPos** | Returns all the canvas objects that are visible under given Screen Position of Canvas’ EventCamera |
| **GetObjectsHitByRay** | Returns all the objects on this canvas  that are intersected by given Ray. |
| **Click** | Sends an OnClick event to every Button component under the pointer. |
| **PointingAtCanvas** | Returns true if the user's pointer is currently pointing inside this canvas. |

**CurvedUIInputModule**

Description

CurvedUIInputModule is automatically added to your eventSystem gameobject if you select a Control Method that requires it. It handles PointerEventData generation and platform-specific interactions. It inherits from StandaloneInputModule.

Private

|  |  |
| --- | --- |
| **disableOtherInputModulesOnStart** | Should other Input Modules be disabled when this one starts? Defaut true. You can disable this functionality, but some of the CurvedUI features may stop working. |

Public

|  |  |
| --- | --- |
| **Instance** | Shortcut to singleton CurvedUIInputModule |
| **ControlMethod** | Current **control method**. Decides how user can interact with the canvas. |
| **CustomControllerRay** | Sets the ray that will be used to discover canvas interactions when **Custom Ray Control Method** is chosen in the settings. This is a static method. |
| **CustomControllerButtonDown** | When **Custom Ray** **Control Method** is selected, Input module will use this bool to determine whether interaction button is pressed. This is a static bool. |
| **WorldSpaceMouseInCanvasSpace** | Used when **World Space Mouse control method** is picked in the settings. Returns the position of the world space pointer in Canvas' local space. You can use it to position an image on world space mouse pointer's position. |
| **WorldSpaceMouseInCanvasSpaceDelta** | The change in position of the world space mouse in canvas' units. |
| **WorldSpaceMouseSensitivity** | How many units in canvas space will the world space mouse move when you move your physical mouse by a pixel on the screen. |
| **CurrentPointedAt** | Gameobject we're currently pointing at. |
| **UsedHand** | Which VR Controller can be used to interact with canvas. Left, Right or Both. Default Right. Used both for Vive and Oculus control methods. |
| **SteamVRControllerManager** | Scene's controller manager. Used to get references for Vive controllers. |
| **Right** | Returns Right Vive Controller. Ask this component for any button states. |
| **Left** | Returns Left Vive Controller. Ask this component for any button states. |
| **GazeUseTimedClick** | Gaze Control Method. Should execute OnClick events on button after user points at them? |
| **GazeClickTimer** | Gaze Control Method. How long after user points on a button should we click it? Default 2 seconds. |
| **GazeClickTimerDelay** | Gaze Control Method. How long after user looks at a button should we start the timer? Default 1 second. |
| **GazeTimerProgress (read only)** | Gaze Control Method. How long till Click method is executed on Buttons under gaze? Goes 0-1. |
| **GazeTimedClickProgressImage** | Gaze Control Method. This Images's fill will be animated 0-1 when OnClick events are about |
| **OculusTouchInteractionButton** | Oculus Touch Control Method Which button on the controller will be used to interact with the canvas? |

**CurvedUIVertexEffect**

Description

A component added to all of the canvas gameobjects. It subdivides and curves canvas elements based on settings from its parent CurvedUISettings

Public

|  |  |
| --- | --- |
| **SetDirty** | Force mesh to be rebuilt during canvas' next update loop. Use it as a failsafe if for some reason your UI objects fail to update. |
| **TesselationRequired** | Force vertices to be subdivided again from original vertices. This also causes the curving to be recalculated.  Set by CurvedUISettings when its settings are updated. |
| **CurvingRequired** | Force vertices to be repositioned on the curved canvas.  Set by CurvedUISettings when its settings are updated. |
| **DoNotTesselate** | Check to skip tessellation pass on this object. CurvedUI will not create additional vertices to make this object have a smoother curve. Checking this can solve some issues if you create your own procedural mesh for this object. Default false. |

**Experimental Features**

**TextMeshPro Support**

CurvedUI supports basic functionality of TextMeshPro. To enable this feature, go to your player settings and add new custom define: CURVEDUI\_TMP

To learn how to do it, visit <http://docs.unity3d.com/Manual/PlatformDependentCompilation.html> and search for “Platform Custom Defines”

CurvedUI will properly bend the TMP text and automatically add needed components. It will also update the generated mesh whenever there is a change in TMP text properties.

However, scripts that modify vertices of TMP text object to achieve additional visual effects will not work with CurvedUI. In a case when a change to TMP object will not trigger new mesh generation by CurvedUI and no change will be visible, you can set Dirty flag to true on CurvedUITMP component to force new mesh generation.