Documentation **V13**

Mesh Deformation Full Collection

PC - VR - Mobile

Thirteenth version of the Mesh Deformation Full Collection package.

Available Online Media

[Online media doesn't have to be the same version, but they are related to the actual version]

Package Overview **V13** [Youtube]

Package Documentation V12 [Youtube]

Online Documentation [Online Presentation]

13th Update (12/03/2020 dd/mm/yyyy)

- Added tunnel creator system
- Added FFD & simple deformer modifier
- Added new example scenes
- Updated documentation
- Fixed Example Scripts
- Updated Example Scenes
- Updated VR API
- Updated API
- Updated Mesh Editor Component
- Package cleanup [Unity 2018 and +]

Content

Basics

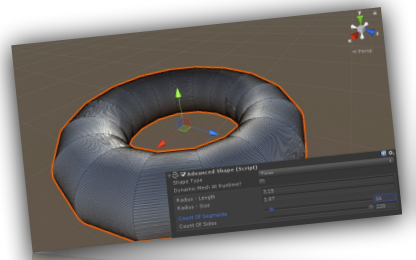
Basic information you should know

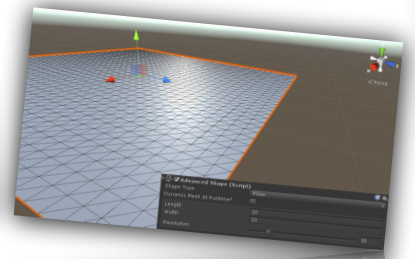
Introduction
How to set up
How to set up VR
Mobile

Introduction

- **MD [Mesh Deformation]** is a collection of all methods of mesh manipulation in Unity for absolute beginners or advanced users. The package consists of vertex editor, modifiers, vertex editor in VR, mesh collider controller, vertex & fragment shaders, physically based mesh and basic shapes generator. MD Package contains a lot of examples with explanations, source code and description.
- The package began to be developed in 2014 by Matej Vanco. It started very easily with very basic features such as generating generic points for mesh, basic controls of Skinned Mesh bones etc. The package was officially published on 12th of November 2015.
- Mesh Deformation is not very common thing in Unity, because Unity is not a modelling program. This package contains many systems with unique functionality. It's a collection of mesh deformations in Unity based on mathematical operations, combinations, logical connections and more.

In this documentation, you will learn basic application of the plugins, additional tools and you will also get some advanced tips and tricks.





How To Set Up

Video Tutorial

The *Mesh Editor* is very easy to set up in Unity editor or at runtime. To set up basic 'Mesh Editor' in **Unity editor** follow the steps at the **left side**, to set up 'Mesh Editor' **at runtime** follow the steps at the **right side** of the screen.

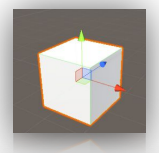
In Editor

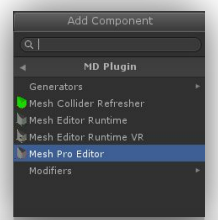
1. Select any object with Mesh Filter or Skinned Mesh Filter
[For example: default cube]

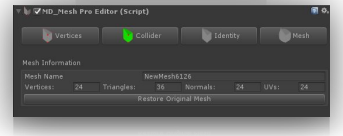
2. Search & add Mesh Pro Editor. The window will show up. Press *Yes* to create new reference
[If you'd like to create new reference]

3. Done, now you are free to choose a *selection mode* in the Mesh Pro Editor component.
[More about the components in MD Plugins category]

NOTICE
If your target object is Skinned Mesh Filter, press *Compile to Mesh Filter*







At Runtime

1. To edit any mesh at runtime, the Mesh Pro Editor component is required & you have to generate it's vertices
[Mesh Pro Editor allows you to generate vertices - just press Vertices button]

2. Select the main camera or any other object that will represent *'origin'* or controller of the Mesh Deformation at runtime
[Raycast needs an origin - choose a correct object for it]

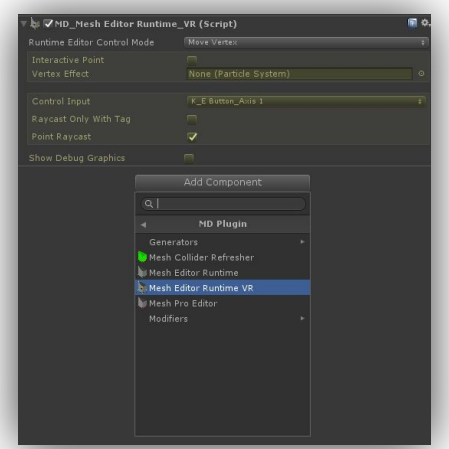
3. Search & add Mesh Editor Runtime

4. Done, your mesh is ready to be edited.

How To Set Up VR

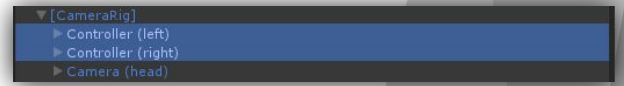
Video Tutorial

Surely, the VR mode works only at runtime. It's very simple. You need at least **one** object with 'Mesh Pro Editor' component to set up Mesh Editor for VR. It works on the **Controllers manipulation**. Steps how to do it are described below. The SteamVR plugin is required!



At Runtime

1. To edit any mesh at runtime for VR, choose one or both controllers for your VR.



2. Search & add Mesh Editor Runtime VR

3. Done, your headset is ready for mesh editor at runtime for VR

NOTICE

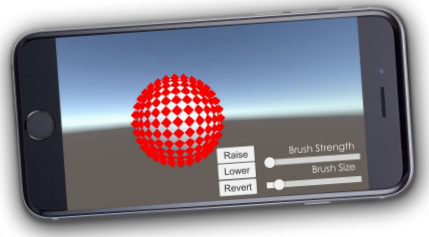
Mesh Editor in VR is compatible with **HTC Vive, Oculus Rift & Mobile Cardboard**

Mobile

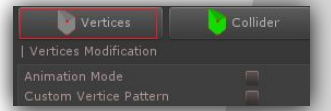
Video Tutorial

MD Package is totally ready for all mobile platforms. Everything what you need is your smartphone, **Unity 5 Remote application** and MD Package. It's easy to initialize, just drag and drop 'Mesh Editor Runtime' and check if 'Mobile Support' is enabled. You are done! **What about performance?** As usually, performance is still a big problem in games. MD Package contains many functions to save your performance as much as it's possible. Mesh Deformation Runtime for mobile is available only at runtime mode. But the mesh set up is the same.

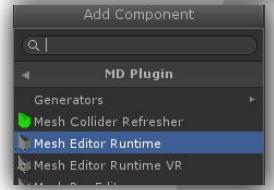
At Runtime



1. Meshes you want to edit must contain Mesh Pro Editor & you have to generate their vertices



2. Select main camera or any other object that will represent 'origin' or controller of the Mesh Deformation at runtime



3. Search & add Mesh Editor Runtime

4. Enable **Runtime Editor For Mobile.** Mesh is ready to be deformed.

MD Plugins

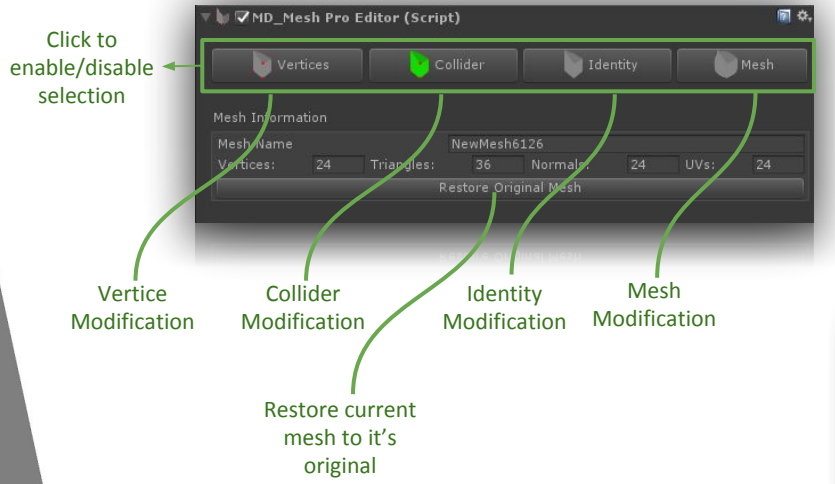
Description & API of the included Mesh Deformation plugins, **Tooltips are available**
Global namespace = **MD_Plugin**

Mesh Pro Editor
Mesh Editor Runtime
Mesh Editor Runtime VR
Mesh Collider Refresher
Mesh Sculpting Pro
Mesh Paint
Shapes

Mesh Pro Editor

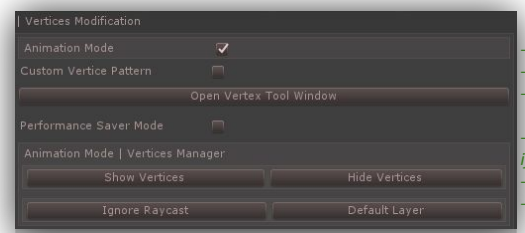
Pro Editor

Mesh Pro Editor is a representative component for mesh deformation package. It contains Vertices, Collider, Mesh Modification & Mesh Identity panels that can be set up. Also, It's possible to use internal API. To set up Mesh Pro Editor, select any object with 'Renderer component' and add the component. More information about the Mesh Pro Editor can be found below. Naming macro: MeshEditor_



API

public void MeshEditor_ShowHideVertices(bool Activation)
[Show/ Hide points if possible]
public void MeshEditor_IgnoreRaycastVertices(bool IgnoreRaycast)
[Set Raycast Layer to generated points]
public void MeshEditor_CreateVerticeEditor(bool PassTheVerticeLimit=false)
[Create vertice editor – points]
public void MeshEditor_ClearVerticeEditor()
[Clear created vertice editor – points]
public void MeshEditor_CombineMesh()
[Combine current mesh with it's submeshes (children)]
public void MeshEditor_QuickCombineMesh()
[Quick mesh combination with it's submeshes without any notice and new instance]
public void MeshEditor_CreateNewReference()
[Create new mesh reference - reset transform and matrix]
public void MeshEditor_RestoreMeshToOriginal()
[Restore current mesh to it's original state]
public void MeshEditor_CompileToMeshFilter()
[Compile current Skinned Mesh to Mesh Filter]
public void MeshEditor_SmoothMesh(float Intensity)
[Smooth current mesh by specific value]
public void MeshEditor_SubdivideMesh(float Level)
[Subdivide current mesh by specific level]



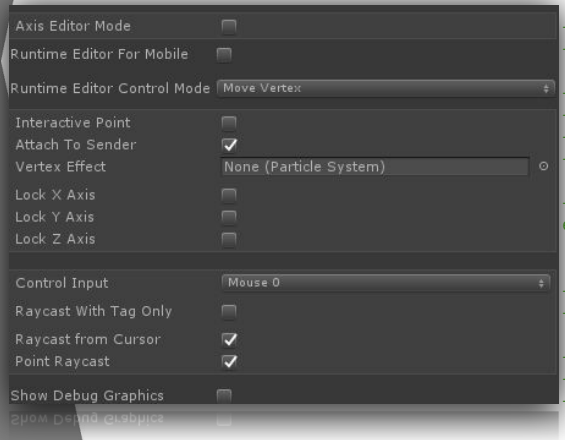
- Animation mode [keep the vertices]
- Custom Vertice Pattern [Customize vertices look]
- Open vertex tool window [Vertex Tool Win will show up]

- Performance Saver Mode [enable/disable this option]
*if Animation mode is **enabled***
- Show/Hide generated vertices
- Ignore/ Default layer for generated vertices

Mesh Editor Runtime

Pro Runtime

Mesh Editor Runtime allows you to edit mesh from the specific origin at runtime. Just select the main camera or object that will represent an 'origin' of the 'Mesh Editor Runtime'. You can choose control mode, input event, raycast filter, raycast origin and more. You can also use the **AXIS** mode which allows you to create *unity-like* editor with axis object *[that can be customized as well]*. Naming macro: NON_AXIS_ and AXIS_



- Axis Mode [Runtime editor mode]
- Runtime Editor For Mobile [mobile support]

- Control Mode [move,push,pull,sculpt]
- Interactive Point [point will move to raycast point]
- Attach To Sender [attach to raycast origin root]
- Vertex Effect [effect that will play if holding point]

- Lock X, Y and Z axes [limit vertices movement to the each axis]

- Control Input [input for vertice grap&drop]
- Raycast With Tag Only [filter for raycast by tag]

- Raycast From Cursor [otherwise from specific object]
- Point Raycast [if disabled: spherical raycast]
- Show Debug Graphics

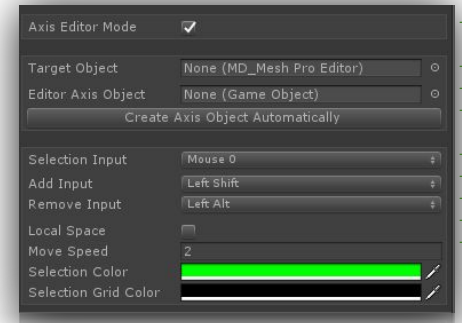
API

If AXIS MODE disabled

public void NON_AXIS_SwitchControlMode(int)
[Change current control mode by index - Pull,Push,Sculpt...]
public void NON_AXIS_MoveVertex(Ray, RaycastHit, bool)
[Move with vertices by custom raycast]
public void NON_AXIS_PushVertex(Ray, RaycastHit, bool)
[Push vertices by custom raycast]
public void NON_AXIS_PullVertex(Ray, RaycastHit, bool)
[Pull vertices by custom raycast]
public void NON_AXIS_SculptVertex(Ray, RaycastHit, bool)
[Sculpt vertices by custom raycast]
public void SS_Funct_ChangeRadius(Slider UI or float)
[Change radius of sculpt brush]
public void SS_Funct_ChangeStrength (Slider UI or float)
[Change strength of sculpt brush]

If AXIS MODE enabled

public void AXIS_SwitchTarget(MeshProEditor object)
[Change target for AXIS editor]
public void AXIS_Undo()
[Undo axis point selection]



- If Axis Editor Mode is enabled

- Target Object [target for editing at runtime]
- Editor Axis Object [axis object for editing]
- Create Axis Object Automatically

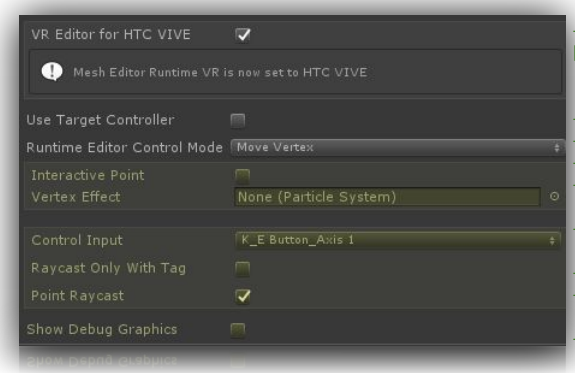
- Selection Input [input for selection]
- Add Input [input for add vertices from list]
- Remove Input [input for remove vertices from list]
- Local Space [Axis Object in local if enabled]
- Move Speed [Axis Object speed if selected]

Mesh Editor Runtime VR

Mesh Editor Runtime VR allows you to edit mesh from the specific controller at runtime for Virtual Reality. Just select one or both controllers of your VR device. The controller will represent an 'origin' of the Mesh Editor Runtime VR. You can choose control mode, input event, raycast filter, raycast origin etc. It's a very same principle as non-VR mesh editor runtime. Naming macro: MeshRuntimeVR_

API

public void MeshRuntimeVR_SwitchControlMode(int)
[Change current control mode by index - Move,Pull,Push,Sculpt...]
public void MeshRuntimeVR_MoveVertex(Ray, RaycastHit, bool)
[Move with vertices by custom raycast]
public void MeshRuntimeVR_PushVertex(Ray, RaycastHit, bool)
[Push vertices by custom raycast]
public void MeshRuntimeVR_PullVertex(Ray, RaycastHit, bool)
[Pull vertices by custom raycast]
public void MeshRuntimeVR_SculptVertex(Ray, RaycastHit, bool)
[Sculpt vertices by custom raycast]
public void SS_Funct_ChangeRadius(Slider UI or float)
[Change radius of sculpt brush]
public void SS_Funct_ChangeStrength (Slider UI or float)
[Change strength of sculpt brush]



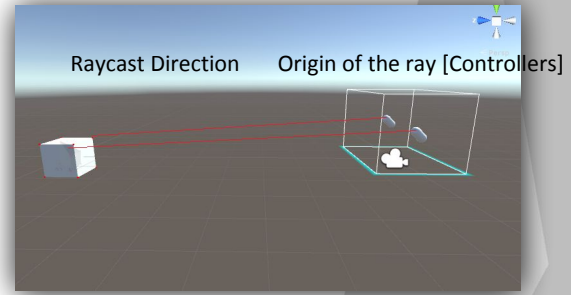
- VR Editor For HTC VIVE
[If disabled, editor is ready for Oculus]

- Use Target Controller [if disabled, target controller will be required]
- Control Mode [move,pull,push,sculpt vertex]
- Interactive Point [point will move to raycast point]
- Vertex Effect [effect that will play if holding point]

- Control Input [input for action]

- Raycast Only With Tag [filter for raycast by tag]
- Point Raycast [if disabled: spherical raycast]

- Show Debug Graphics [show debug in editor]



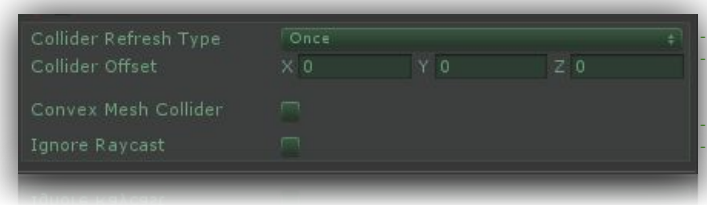
Mesh Collider Refresher

Mesh Collider Refresher allows you to update & refresh mesh collider at runtime. You can choose many modes such as refresh mesh collider once, every frame or by custom interval. You are also able to update the mesh collider manually or by the specific event action. In the refresh type 'once' you can set up your own mesh collider position offset. Simple example is described below.
Naming macro: MeshCollider_

API

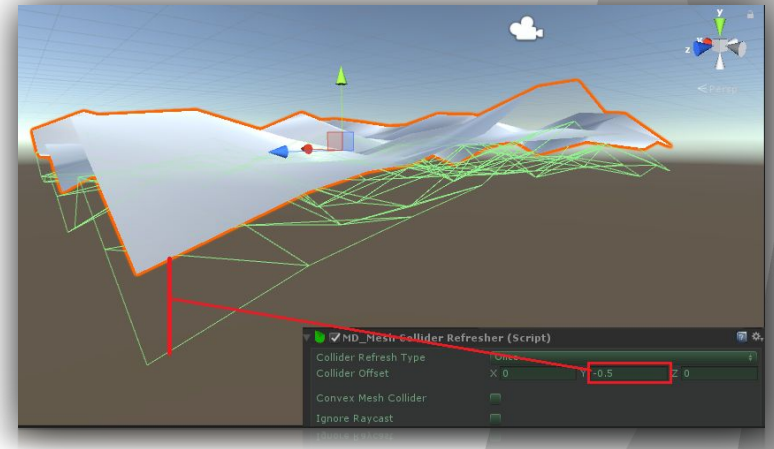
public void MeshCollider_UpdateMeshCollider()
[Update current mesh collider]

public static void MeshCollider_UpdateMeshCollider(GameObject Target, bool Convex)
[Update target mesh collider]



- Collider Refresh Type [Once, PerFrame, Interval, Never]
- Collider Offset [if once]

- Convex Mesh Collider
- Ignore Raycast



Mesh Sculpting Pro

Mesh Sculpting Pro allows you to deform any mesh in different and unusual ways. It's totally free tool to edit mesh in more-artistic way. You are able to change control mode from raise to lower and revert. You can also add your own method. Brush type and brush appearance is fully customizable with brush radius, strength etc. You can also check internal API which is very rich and full of very helpful methods. For those, who has still no idea what the Sculpting is, it's similar system to terrain editor. The compatibility is great for Mobile and VR as well. Mesh Sculpting is fully ready for extremely complex meshes! Naming macro: SS_Funct

API

public void SS_Funct_RestoreOriginal()
[Restore mesh to the first original mesh]

public void SS_Funct_BakeMesh()
[Restart mesh transform and matrix]

public void SS_Funct_DoSculpting(Vector3, Vector3, float, float, enum)
[Process sculpting on the current mesh]

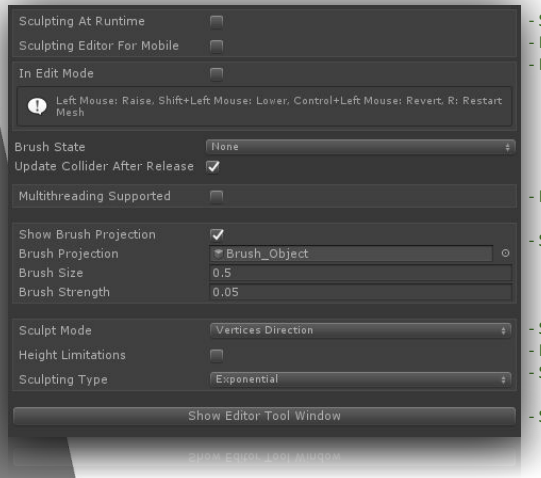
public void SS_Funct_RefreshMeshCollider()
[Refresh current mesh collider]

public void SS_Funct_ChangeBrushState(int)
[Change brush state 0 – none, 1 – raise, 2 – lower, 3 – revert]

public void SS_Funct_SetBasics(float, float, bool, Vector3, Vector3)
[Set basic parameters for current sculpting object]

public void SS_Funct_ChangeRadius(Slider UI or float)
[Change radius of sculpt brush]

public void SS_Funct_ChangeStrength (Slider UI or float)
[Change strength of sculpt brush]



- Sculpting At Runtime [if enabled, sculpting is ready for runtime]
- For Mobile [if enabled, mobile support is allowed]
- In Edit Mode [if enabled, in-editor mode is allowed]

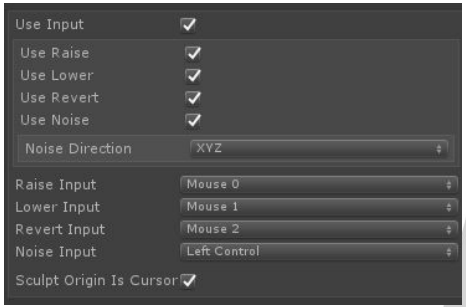
- Multithreading Supported [if enabled: ready for complex meshes]

- Show Brush Projection [show/hide brush in scene]

- Sculpt Mode [Vertices, Custom Dir, Internal Dir, Brush Dir]
- Height Limitations [limit vertices Y location]
- Sculpting Type [Exponential-smoother, Linear-blocky]

- Show Editor Tool Window [quick window for editor]

If Sculpting At Runtime is enabled



- Use Input [use input preset]
- Use each sculpting function

- Noise Direction [X,Y,Z]

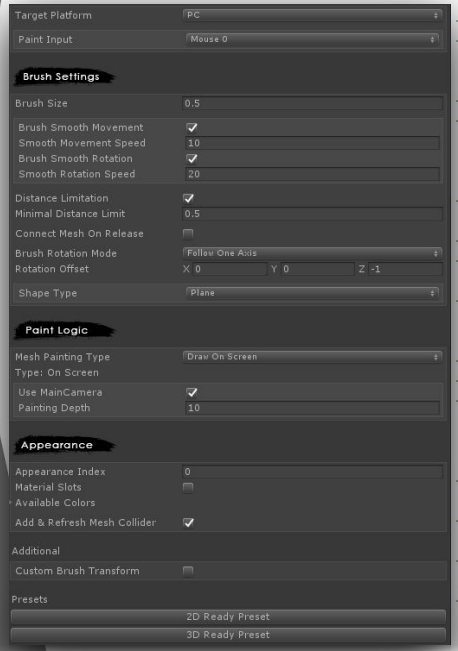
- Input for each function [if enabled]

- Sculpting Origin Is Cursor
[else: specific object-forward vector]

Mesh Paint

Mesh Paint

Mesh Paint allows you to paint mesh in 2D or in 3D. Each mode has a specific and unique options that can be set up. You can draw triangles in a row, planes or cubes. These very simple shapes can be also edited by materials or custom colors. Mesh Paint is compatible for VR as well. You are also able to choose 2D or 3D Preset. Naming macro: PUBLIC_



- Target Platform [PC, VR, Mobile]
- Paint Input [input to process painting]

- Brush Size [painting brush size]
- Brush Smooth Movement [enabled/disable smooth movement]

- Distance Limitation [drawn mesh smoothness]

- Connect Mesh On Release [connect mesh]
- Brush Rotation Mode [brush rotation mode - more in ToolTips]

- Shape Type [all types of shape]

- Mesh Painting Type [On Screen, Raycast and Custom]
- Use Main Camera [else you must enter specific camera]
- Painting Depth [Z depth from camera]

- Appearance index [from available colors or materials]

- Add Mesh Collider [if mesh is drawn]

- Custom Brush Transform [custom brush transform to follow]

- Presets

API

public void PUBLIC_CreateNewPaintPattern(string, bool)
[Create new paint-ready object & make it readable]

public void PUBLIC_PaintMesh(Vector3, MeshPaintModeInternal)
[Paint mesh on the specific location and meshpaintmode]

public void PUBLIC_ChangeBrushSize(float)
[Change Brush Size by float]

public void PUBLIC_IncreaseBrushSize(float)
[Increase Brush Size by float]

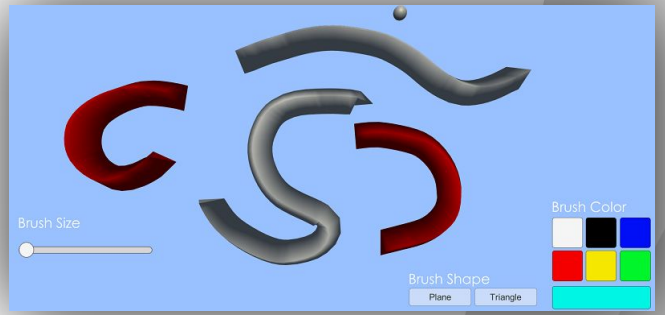
public void PUBLIC_DecreaseBrushSize(float)
[Decrease Brush Size by float]

public void PUBLIC_ChangeBrushSize(UI.Slider)
[Change Brush Size by UI Slider]

public void PUBLIC_EnableDisableDrawing(bool)
[Enable/ Disable custom drawing method]

public void PUBLIC_ChangeAppearanceIndex(int)
[Change selected appearance index by integer]

public void PUBLIC_ChangeShapeType(int)
[Change shape type by integer. 0 = Plane, 1 = Triangle, 2 = Cube]



Shapes

Mesh Deformation package also contains many basic and advanced shapes. You can easily create any shape in Editor *[GameObject/3D Object/MD Plugin]* or at runtime from script with Generator classes. There is also a special type of plane that contains many useful features like dynamic angle, customizable plane size & count of vertices.

Advanced shapes

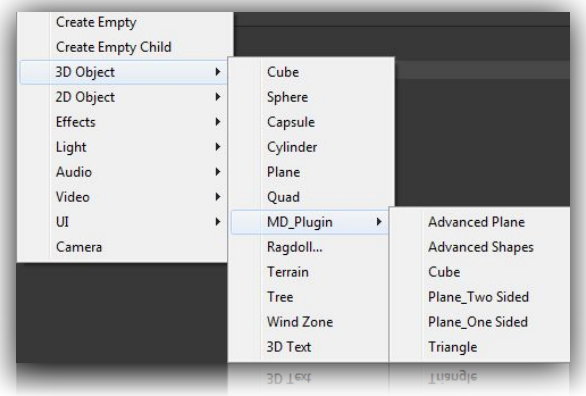
This category contains Advanced Plane generator & Advanced Shapes generator. The access is very easy and every step is described inside and outside the scripts. You are able to create advanced torus with custom vertices count, advanced cone, advanced custom box and many more.

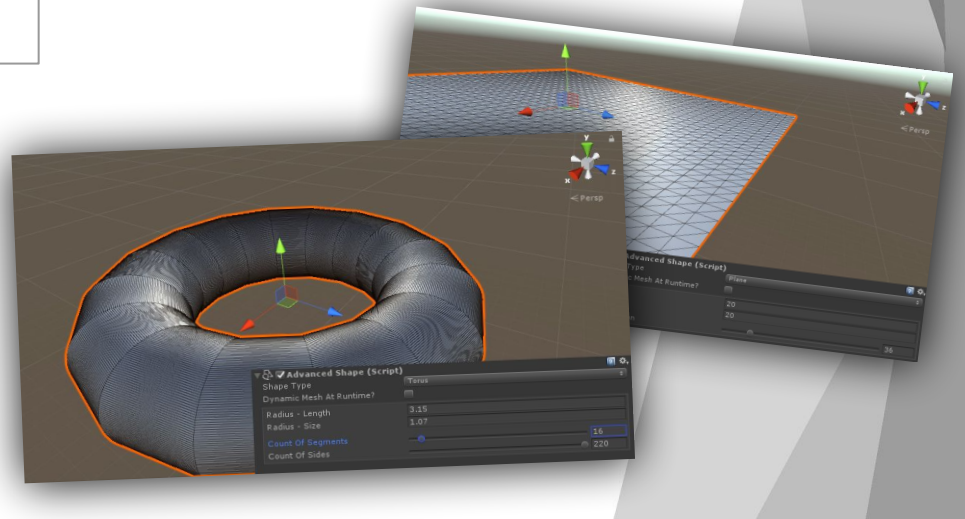
Hexagon Grid

Hexagon Grid allows you to generate hexagonal grid with customizable size, amount and offset.

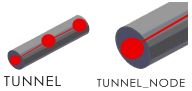
API

class AdvancedPlaneMesh_Generator.Generate()
[returns Gameobject of AdvancedPlane]
class AdvancedShape.Generate()
[returns Gameobject of AdvancedShape]
class CubeMesh_Generator.Generate()
[returns Gameobject of Cube]
class PlaneMesh_Generator.Generate()
[returns Gameobject of Two Sided Plane]
class PlaneMesh_Generator_S1.Generate()
[returns Gameobject of One Sided Plane]
class TriangleMesh_Generator.Generate()
[returns Gameobject of Triangle]

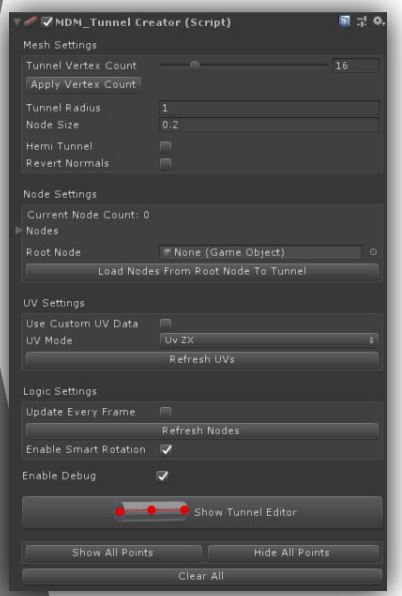




Tunnel Creator



Tunnel Creator allows you to make stunning ‘cylindrical’ meshes with many technical but also visual options. The system is based on nodes and weights. There also many additional options to make a turn, straight line or connect other ‘tunnels’. Tunnel Creator also contains its own editor window to make it much more user friendly. Tunnel Creator system contains 2 important components: **MDM_TunnelCreator** *(the root)* and **MDM_TunnelNodeUVData** *(the node uv data controller).* Naming macro: PUBLICtFunct_



- Tunnel vertex count (Apply must by pressed on change)

- Tunnel radius & Nodes size

- Hemi tunnel (half tunnel) & revert normals

- Nodes list
- Load nodes (as a childs) from specific tranform

- Use custom uv data (nodes will contain TunneNodeUVData)
- If option above is disable, use global UV data (simplier option)

- Update tunnel mesh every frame
- Refresh Tunnel mesh
- Use smart rotation (nodes will rotate towards previous node position)

- Show tunnel editor window

- Additional features

API

public void PUBLICtFunct_AddNode(Vector3, bool)
[Add node on specific position]
public void PUBLICtFunct_RemoveNode()
[Remove last node]
public void PUBLICtFunct_ClearAll()
[Clear all nodes]
public void PUBLICtFunct_RefreshNodes()
[Refresh current tunnel mesh]

public void PUBLICtFunct_ApplyVertexCount()
[Apply changed vertex count and refresh]
public void PUBLICtFunct_GroupNodesTogether()
[Group all nodes together in hierarchy]
public void PUBLICtFunct_UngroupNodes(Transform)
[Ungroup all nodes to 'empty' or to 'some object']

public void PUBLICtFunct_UpdateUVs(UVMode)
[Update UV sets with specific UV mode]

Modifiers

Description & API of the included modifiers, **Tooltips are available**
Global namespace = **MD_Plugin**

MDM_InteractiveLandscape
MDM_LandscapeTrackingGPU
MDM_MeshDamage
MDM_MeshNoise
MDM_Morpher
MDM_FFD
MDM_RaycastEvent
MDM_MeshFit
MDM_Twist
MDM_Bend

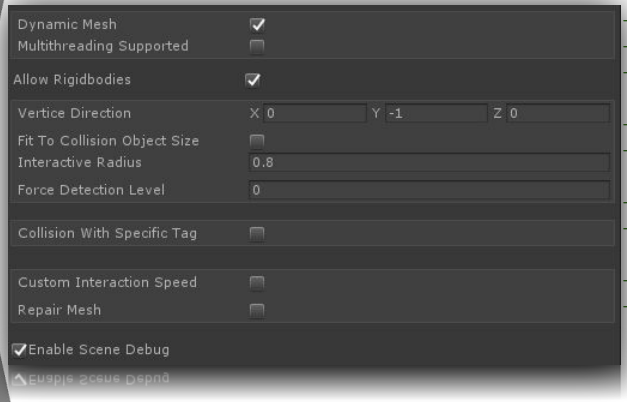
Interactive Landscape

MOD_Land

Interactive Landscape is a modifier that allows you to make a physically based landscape. You can simulate any kind of landscape. Snow, mud, water and more. It consists of a few settings that have to be set up. Naming macro: InteractiveLandscape_

API

public void InteractiveLandscape_ModifyMesh(Vector3, float, Vector3)
[Modify current mesh by required parameters]
public void InteractiveLandscape_ModifyMesh(RaycastEvent)
[Modify current mesh by custom RaycastEvent]

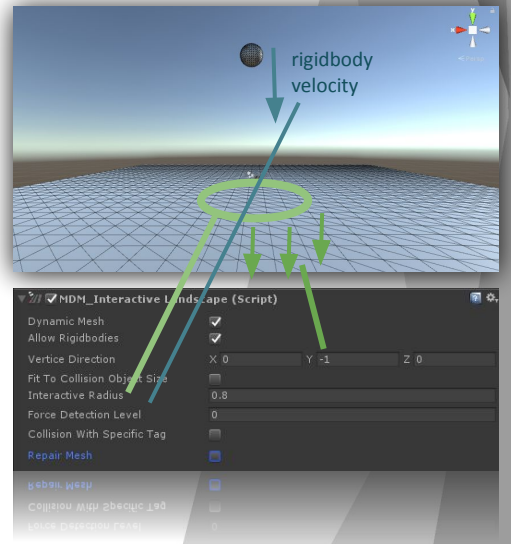


- Dynamic Mesh [update mesh every frame]
- Multithreading Supported [make it ready for complex meshes]
- Allowed Rigidbodies [allow rigidbody objects - takes more performance]

- Vertice Direction [vertice direction after interaction]
- Fit To Collision Object Size [interactive radius will be set to objects scale]

- Force Detection Level [rigidbody velocity]
- Collision With Specific Tag [filter objects to collide with specific tag]

- Custom Interaction Speed [smooth vertices movement after interaction]
- Repair Mesh [repair interacted vertices]

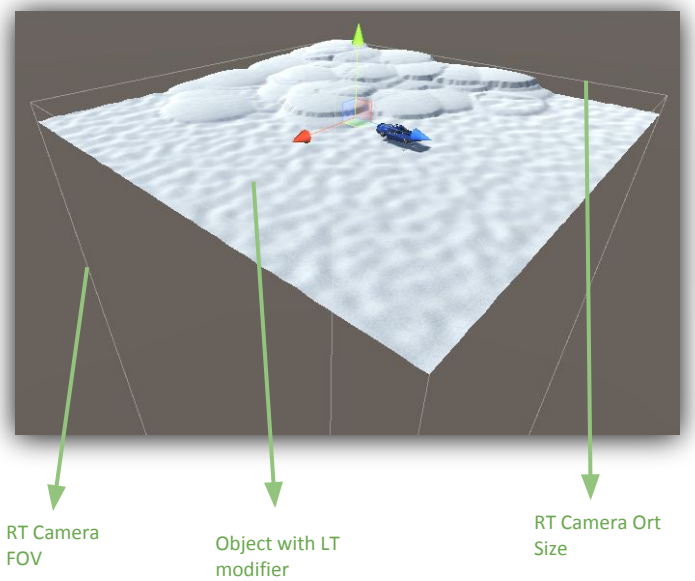


API Not Available

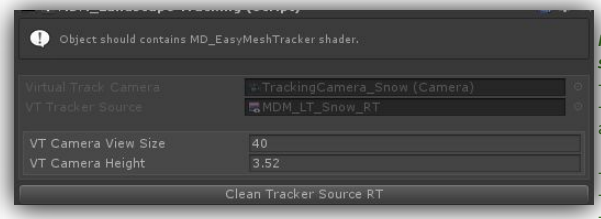
Landscape Tracking GPU

MOD_Track

Landscape Tracking allows you to simulate surface deformation without any FPS drop-down. The whole modifier controls one specific material that must contains **MD_EasyMeshTracker** shader. The shader controls surface with input RT [Render Texture] that captures one of the channels. The channel is captured thanks to selected layers that you can choose in the Landscape Tracking modifier. The principle of the modifier is the same as Interactive Landscape, but the technique is much different and performance speed is much faster and safer. The only obstacle might be accuracy and multiplatform-support.



[Video Tutorial]



NOTICE: Object should contain MD_EasyMeshTracker shader!

- Virtual Track Camera [will be created automatically]
- Tracker Source [RT Texture will be create automatically]

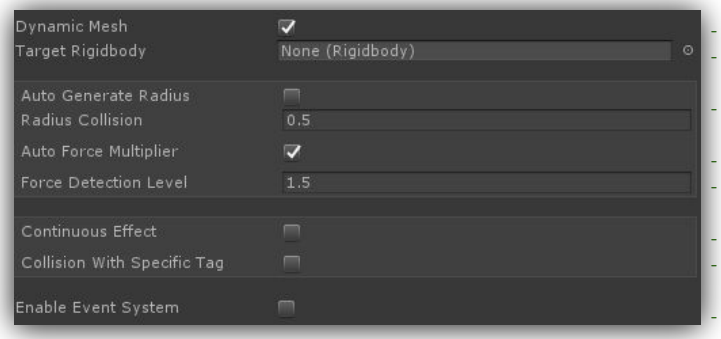
- Camera View Size [camera ort size]
- Camera Height [camera height of Y axis in local space]
- Clean Tracker Source RT [Reset surface]

Mesh Damage

Mesh Damage is a modifier that allows you to make a physically based mesh. According to performance, its very important how do you set up the modifier. It's very similar to Interactive Landscape, but the modifier receives impacts and vertices direction is automatically generated. Naming macro: MeshDamage_

API

public void MeshDamage_ModifyMesh(Vector3, float, Vector3)
[Modify current mesh by required parameters]
public void MeshDamage_RefreshVertices()
[Refresh vertices that have been already interacted]
public void MeshDamage_RepairMesh(float)
[Repair current mesh by specific speed]
public void MeshDamage_ModifyMesh(RaycastEvent)
[Modify current mesh by custom RaycastEvent]



- Dynamic Mesh [update mesh every frame]
- Target Rigidbody [leave it empty if you want use THIS object]

- Auto Generate Radius [radius will be generated by the object impact]

- Auto Force Multiplier [force will be generated by the object impact]
- Force Detection Level [minimal rigidbody velocity]

- Continuous Effect [vertices will be able to move over their limit]
- Collision With Specific Tag [filter for collision by tag]

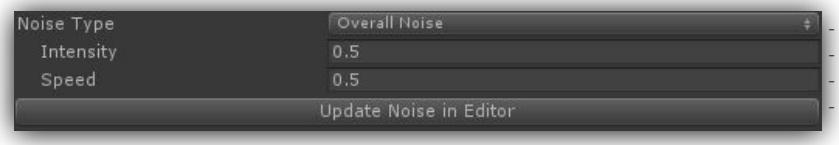
- Enable Event System [after collision]

Mesh Noise

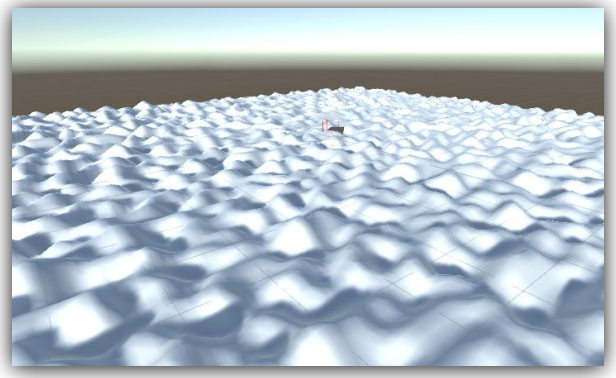
Mesh Noise allows you to perform perlin noise on any mesh. You can choose two types: Overall Noise & Vertical Noise. *Overall* noise has an impact on the whole mesh. You can set up intensity and speed of the noise. *Vertical* noise has an impact only for vertices of Y axis. This modifier works at runtime, but you can start this action by clicking on the button 'Update Noise In Editor'. Naming macro: MeshNoise_

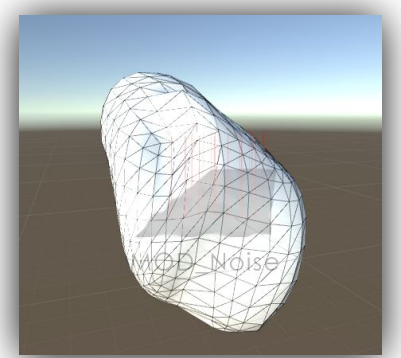
API

public void MeshNoise_UpdateVerticalNoise()
[Update current vertical noise]
public void MeshNoise_UpdateOverallNoise()
[Update current overall noise]



- Noise Type [Overall and Vertical]
- Intensity
- Speed
- Update Noise in Editor [update in editor]





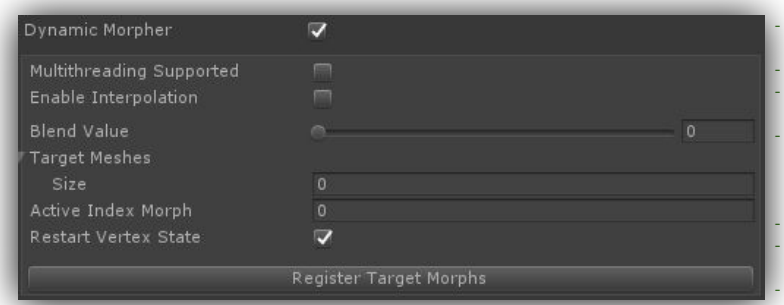
Morpher

Pro Morph

Morpher allows you to create a morphing effect between specific meshes. You need more than one mesh. The mesh will change its shape by the Blend value. Blend value is a normalized value. You can also switch target mesh by selected index. Morpher is ready for very complex meshes as its supported by multithreading. Naming macro: Morpher_

API

public void Morpher_ChangeMeshIndex(int)
[Change current target morph mesh index]
public void Morpher_SetBlendValue(Slider UI)
[Set current blend value]
public void Morpher_SetBlendValue(float)
[Set current blend value]
public void Morpher_SetBlendValue(int)
[Set current blend value]
public void Morpher_RefreshTargetMeshes()
[Refresh target meshes - target meshes must be registered]



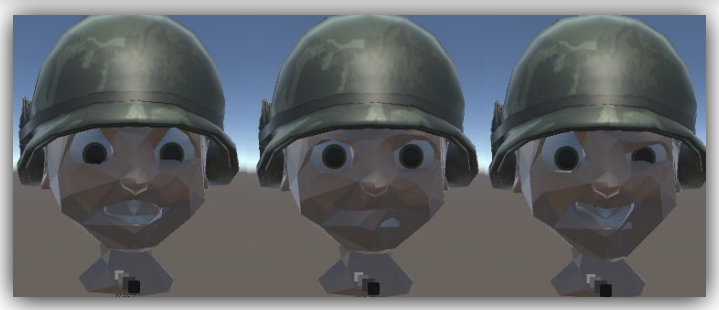
- Dynamic Morpher [update mesh every frame]

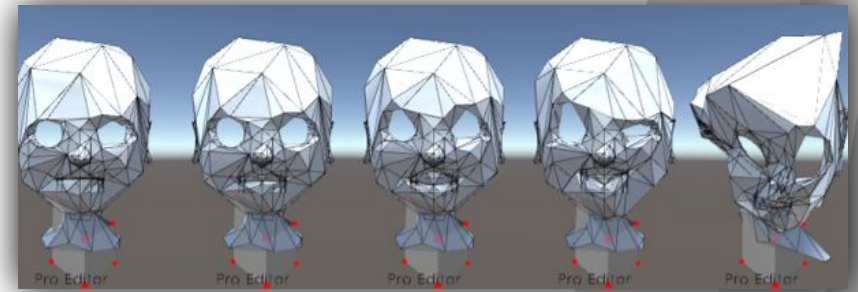
- Multithreading Supported [ready for complex morphs]
- Enable Interpolation [enable smooth transitions]

- Blend Value [blend between 2 meshes (original & target)]

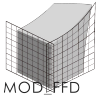
- Active Index Morph [current target mesh from list above]
- Restart Vertex State [restart vertices to original mesh after change]

- Register Target Morphs [register target meshes]





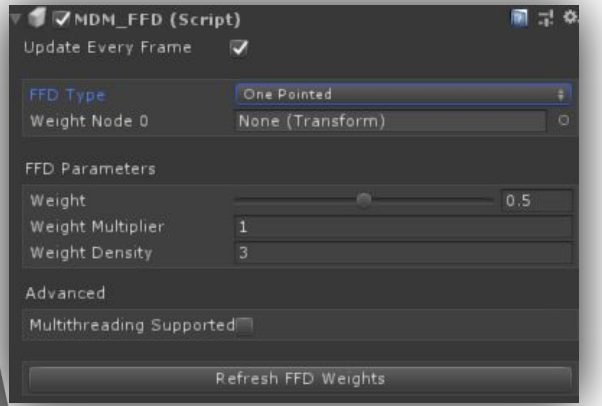
FFD (Free Form Deformation)



FFD allows you to use one of the most useful modifiers called FFD, free form deformation. The modifier simply deform mesh on specific point weights and distances. Current FFD supports 4-pointed weights with many options including weight customization, multiplication, amplification and more. FFD is ready for very complex meshes as its supported by multithreading. Before using the FFD modifier, you must register the specific point weights by pressing the **Refresh FFD Weights** button. Naming macro: FFD_

API

public void FFD_UpdateMesh()
[Update current mesh state (in case if Update Every Frame is disabled)]
public void FFD_ApplyWeights()
[Apply & register FFD weights]



- Dynamic FFD [update mesh every frame]

- FFD type (points count)
- Point object (can be ANY object)

- Weight value
- Weight multiplier
- Weight density

- Multithreading support

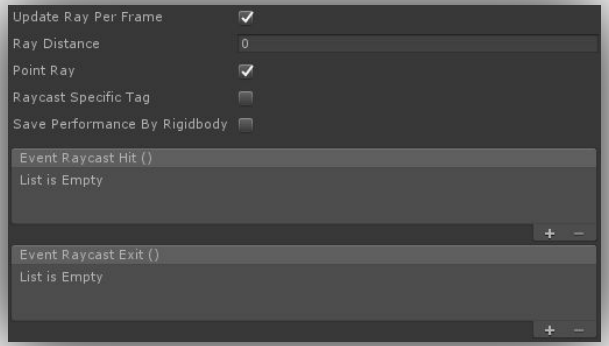
- Register & refresh FFD points

Raycast Event

Raycast event is a great tool to render event if raycast hits something. You can set up your own event system & raycast logic. The raycast event is also connected with Interactive Landscape. You can simulate tracks on the interactive landscape as it's shown in the examples.
Naming macro: RayEvent_

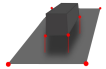
API

public bool RayEvent_IsRaycasting()
[Check if currently raycasting]
public void RayEvent_UpdateRaycastState()
[Update current Raycast]



- Update Ray Per Frame [update every frame]
- Ray Distance [raycast distance]
- Point Ray [otherwise: spherical ray]
- Raycast Specific Tag [filter for raycast by tag]
- Save Performance by Rigidbody
[regulate performance by rigidbody speed]
- Events after hit

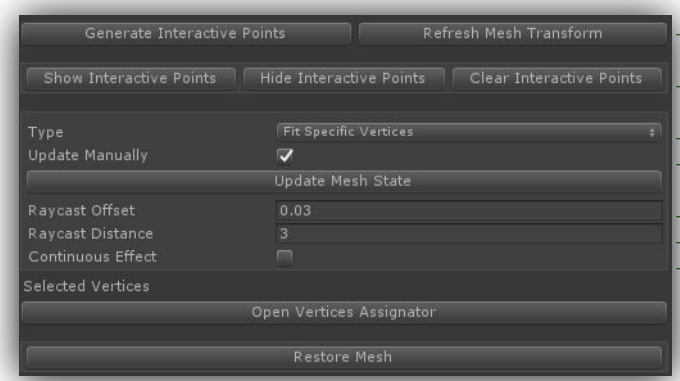
Mesh Fit



Mesh fit allows you to adjust mesh shape to surface with collider. You are able to 'fit' whole mesh [all vertices] or specific selected vertices – you can select which vertices will be interacting. With this tool you can easily create universal decals, volumetric mesh or even organic monsters. This modifier has been moved from Mesh Pro Editor Internal Modifiers to the External Single Modifiers. Naming macro: MeshFit_

API

public void MeshFit_ShowHidePoints(bool)
[Show/ Hide points]
public void MeshFit_GeneratePoints()
[Generate points on the mesh]
public void MeshFit_RestoreOriginal()
[Restore original mesh]
public void MeshFit_ClearPoints()
[Clear points on the mesh [if possible]]
public void MeshFit_BakeMesh()
[Reset mesh matrix transform [Set scale to 1 and keep the shape]]
public void MeshFit_UpdateMeshState()
[Update mesh state - process the points and make a 'mesh fit' action]

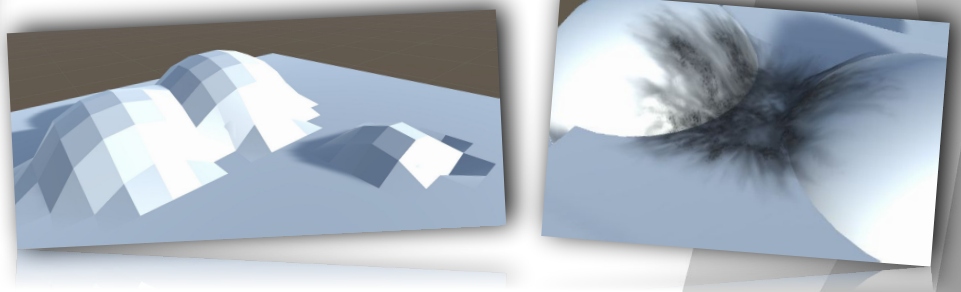


- Generate Points & Refresh Mesh Transform [reset it's matrix to 1]

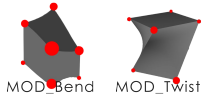
- Show, Hide & Clear Points

- Type [Fit Specific Vertices & Fit Whole Mesh]
- Update Manually [otherwise it will be updated every frame]

- Raycast Offset [raycast Y offset]
- Raycast Distance [maximum raycast distance]
- Continuous Effect [if enabled, the shape will continue it's deformation]



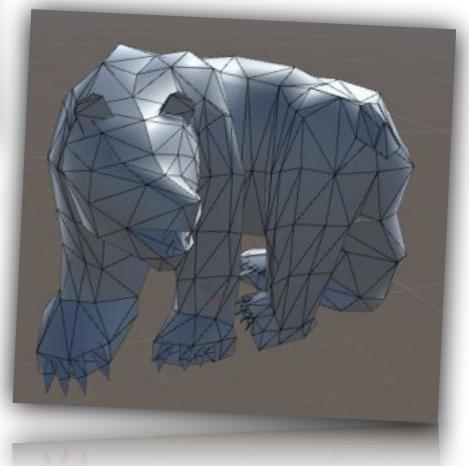
Bend & Twist

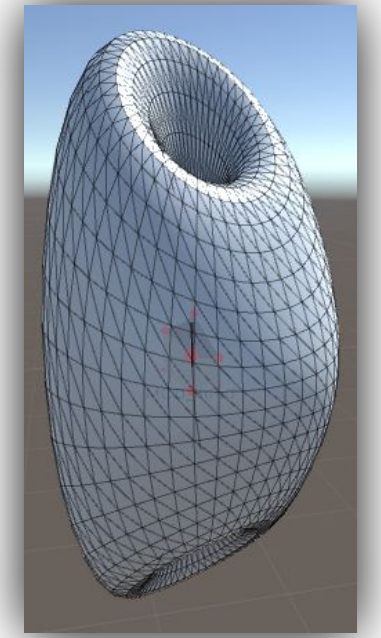


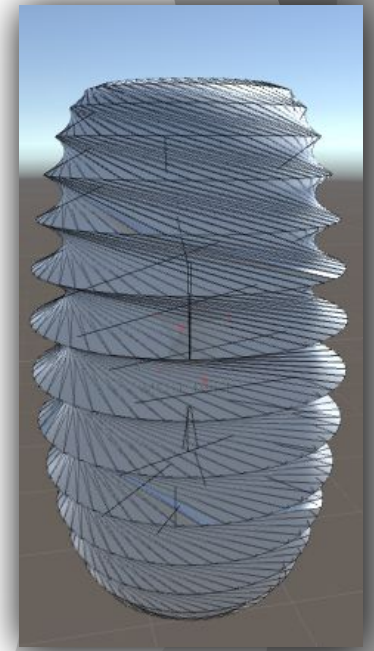
Bend & Twist belong to the default and well known modifiers. You can bend and twist any mesh at runtime. Both modifiers require bend direction and amount of the effect. Modifiers run at runtime only.

API

public void TWIST(float or UI Slider)
[Twist mesh - if twist modifier is applied]
public void BEND(float or UI Slider)
[Bend mesh - if bend modifier is applied]







Vertex & Fragment Shaders

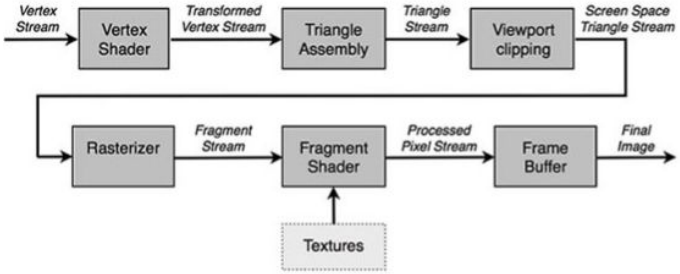
Description of shaders

About Shaders
Easy Application

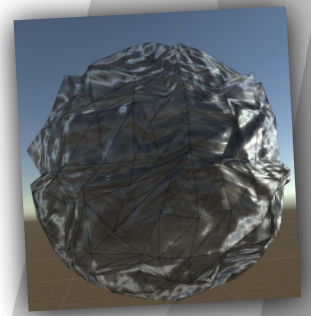
About Shaders

'Shaders' is a huge term in Unity. Mesh Deformation package works with *Fragment & Vertex shaders*, let's talk about them. *Vertex & Fragment* shaders manipulate Mesh vertices, pixels, segments, triangles and fragments. Everything works via GPU, so performance is much faster. You can also check 'gentle introduction' to shaders by Unity here.

Schematic explanation [image source - Unity]



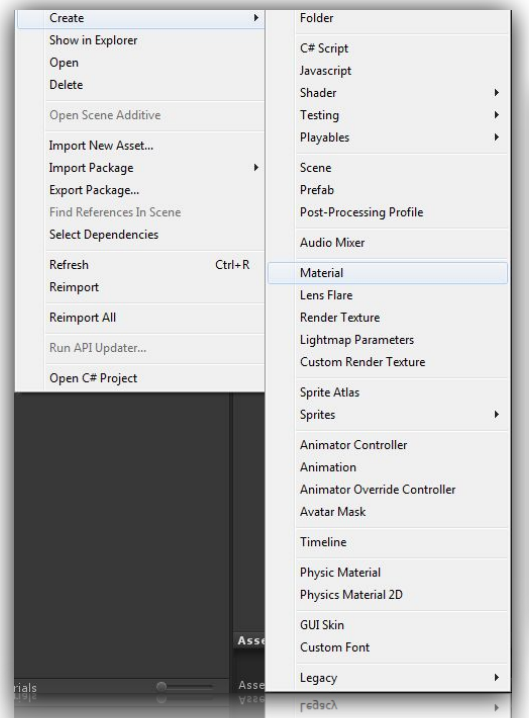
MD Package contains many useful shaders with amazing results. You can create swimming fishes, jumping blobs, organic cells, trees, deformed planes, noises [which is much better than physically based noises], tires, melting effects and so on. All what you need is a material source. Shaders in MD package are constructed to apply basic parameters like Color, Specular, Emission, Bump & Tiling. Every shader is unique and special. You can also use GPU Instancing to make your performance even better.



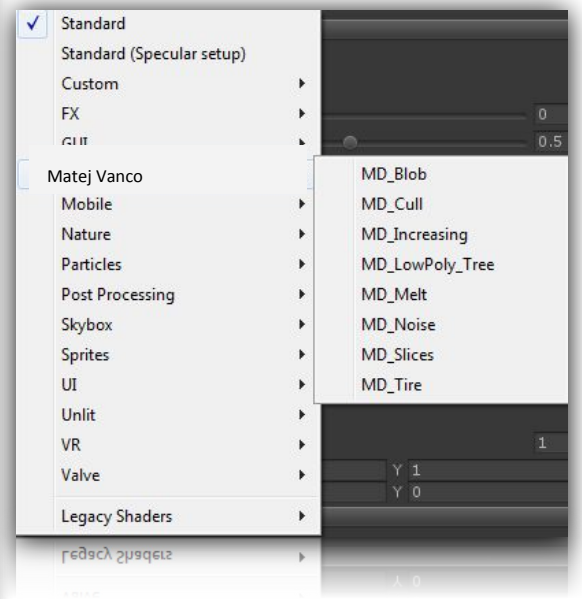
Easy Shaders Application

To apply shader on your object, create a new material and choose one of the shaders in **Matej Vanco/** path. **Please notice that the shaders work only in Unity Standard Render Pipeline!**

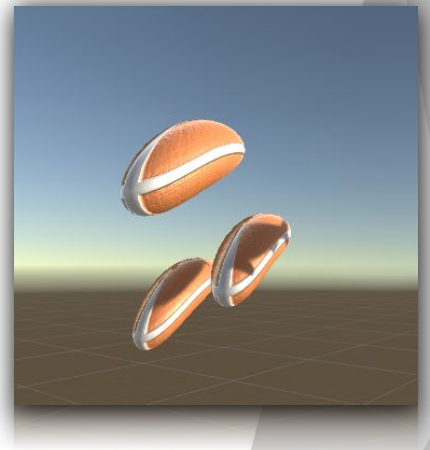
1. Create a new Material



2. Choose one of the shaders



3. Done!



Tips & Tricks

Tips & tricks of the Mesh Deformation package
Learn how and when to use them.
Read FAQ slide!

Snow Simulation
Face Morpher
Performance Saver
Multithreading
Vertex Tool Window
Performance
FAQ

Snow Simulation

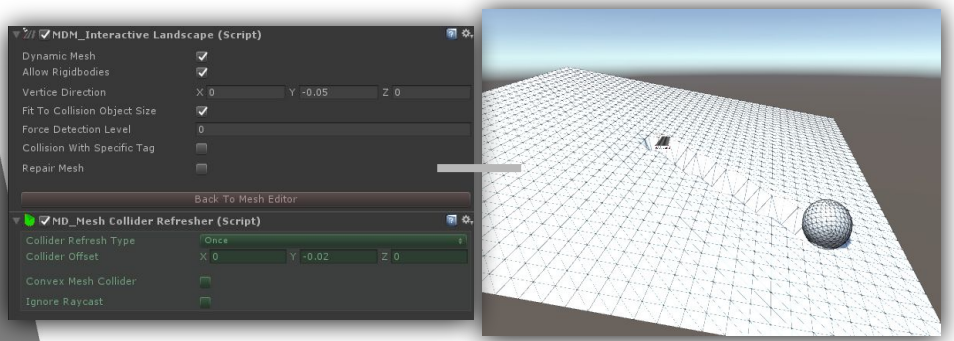
Video Tutorial

Snow simulation is one of the most searching topic. It looks amazing and professional. Surely, you are able to achieve this effect in MD Package. It's very easy to set up, but hard for the performance. We will have to optimize our mesh by reducing vertices. You can also see this beautiful effect in the example scene 'Winter Scene'.

You can also try another method via GPU! Check the new video-tutorial GPU vs CPU
Here

1. Create **Advanced Shapes** and generate **Plane** with resolution of value 40.
2. Go back to the 'Mesh Pro Editor', go to 'Mesh Identity' and choose **Interactive Landscape** modifier
3. Enable **Fit To Collision Object Size**, set **Vertice Direction** to –0.05 and add 'Mesh Collider Refresher'
4. Set 'Mesh Collider Refresher' to **Once** and change offset to Y = -0.02
5. Add any object with rigidbody and press play

Your result should look like this:



Snow simulation is not the only effect that looks amazing. It can be used for mud simulation, water simulation and even more.

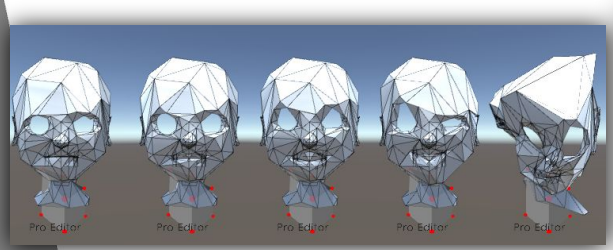
Face Morpher

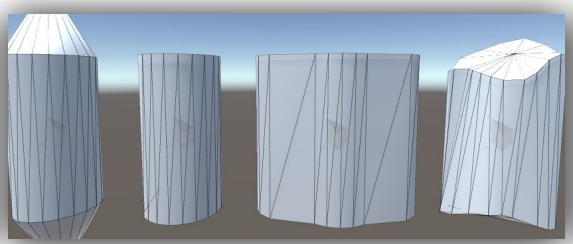
Video Tutorial

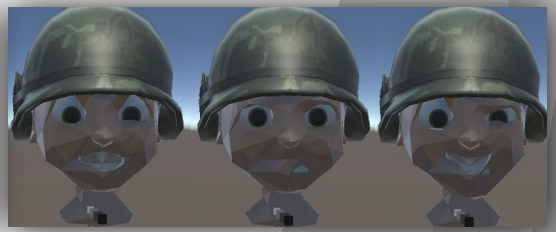
Face Morpher is one of the largest challenges in the 3D Graphics, especially in Unity. Morpher in MD Package is highly optimized so you can use a massive amount of vertices. You can manipulate with blending value and your mesh will change its form. You can also see an example in **MD Package/Example** Scenes. But how to apply your own morpher?

1. Create or import any mesh that you want to morph
2. Add to this mesh 'Morpher' component by MD Package
3. Assign **Target Meshes** that you want to morph [you can duplicate current mesh and change its form and save it to the asset folder as a new mesh or you can use the other imported meshes. NOTICE: All target meshes must have the same vertices array as the mesh with morph component]
4. If all meshes are ready, press **Register Target Morphs**
5. Done. Change **Active Index Morph** to see the result with other target meshes
6. If you'd rather have the vertices movement smooth, you can enable **Interpolation** in the top.
7. Disable/ enable **Restart Vertex State** if you want to restart vertices state after the index changed [enabled is recommended because vertices might be out of the origin]

Your result might look like this:





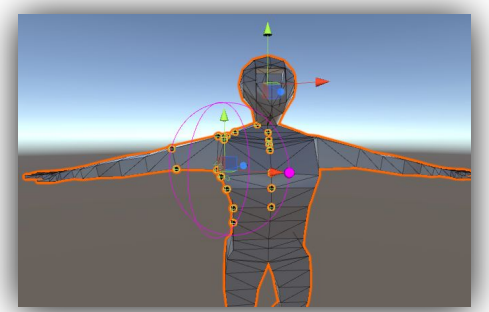


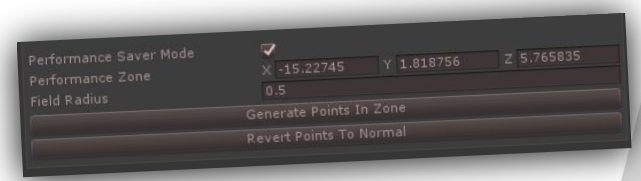
Performance Saver

Video Tutorial

Performance is very important, so I implemented few features in the Mesh Pro Editor. If you are editing a high-poly mesh, your performance could be very slow, sometimes very very slow. You would notice that of course. In this case, we ought use Performance Saver Mode in Vertices Modification as the **MD Plugin is not built for complex** Mesh Editor.

1. Select any object with 'Mesh Pro Editor' component
2. Select **Vertices Modification** and enable **Performance Saver Mode**
3. You can see 4 options: **Performance Zone, Field Radius** and **two Buttons for generation**
4. **Performance zone** is a Vector3 which represents the zone position, you can also see it in the editor Scene
5. **Field radius** is a sphere radius that will overlap points in the zone
6. **Locate Performance Zone** handler and move it to your mesh
7. Press **Generate Points In Zone**. Now you can see the generated points in the specific zone
8. Now your performance will be much faster and your work will be more efficient.



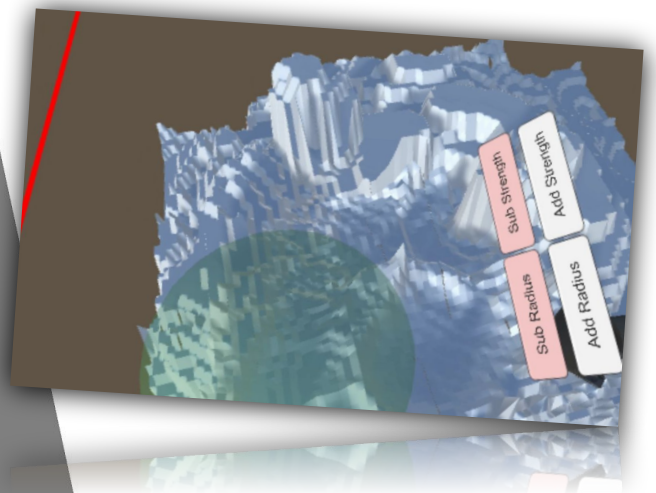


Multithreading

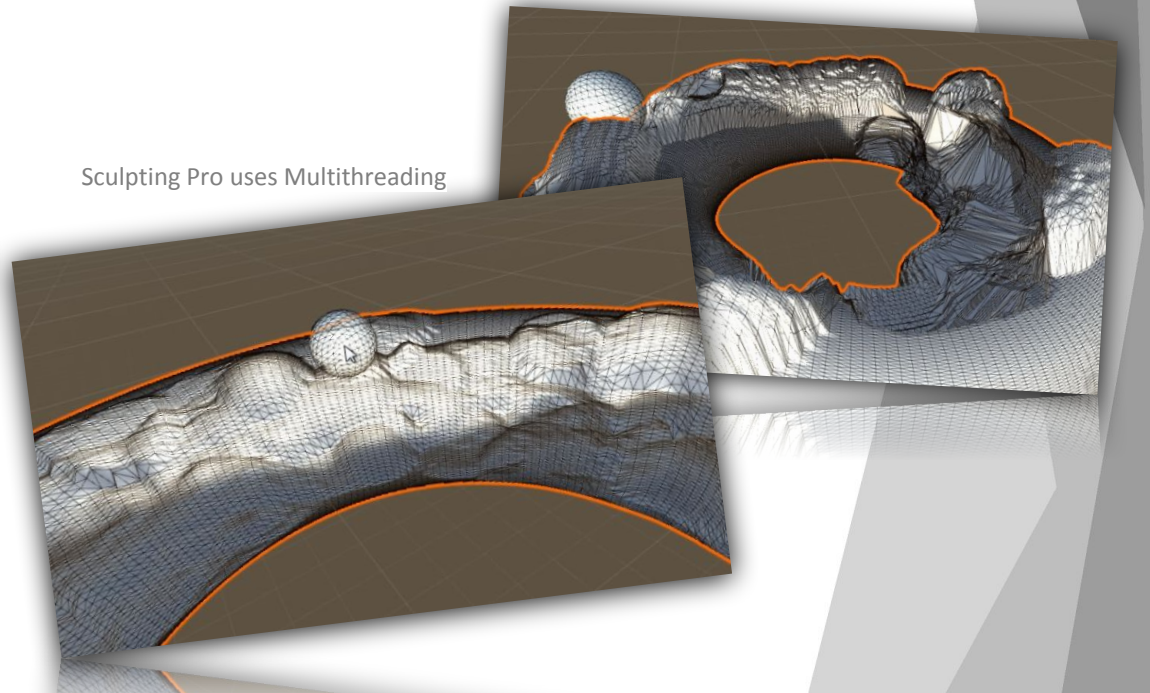
Video Tutorial

Multithreading is a great thing that allows you to save more than 80% of original performance. It consists of creating new threads for complex operations like *searching for nearest vertex point*... It's absolutely safe. If a component contains boolean **Multithreading Supported**, the component supports multithreading method. Multithreading Supported

Multithreading in VR



Sculpting Pro uses Multithreading



Vertex Tool Window

Video Tutorial

Vertex Tool Window is an additional feature in MD Package. You can access it in '**Window/VertexToolWindow**' or in the '**Mesh Pro Editor/Vertices Modification**'. It's an expansion for meshes, vertices and elements. Also, It contains its own API for internal use called **MD_VertexToolModifier**.

Attach 2 or more meshes [Meshes will be combined and will share the same material]

Clone selected mesh [You can see parameters below] Count, Position Offset, Rotation Offset.

Weld selected vertices [Vertices will be weld – they will split into one]

Relax selected vertices [Vertices will be normalized and their offset will be multiplied]

Group selected objects into included object below

Group enabled vertices [Additional group function to group enabled vertices in Performance Saver Mode]

API for internal use
namespace MD_Plugin

V_TOOL_Element_Attach()

V_TOOL_Element_Clone()

V_TOOL_Vertex_Weld()

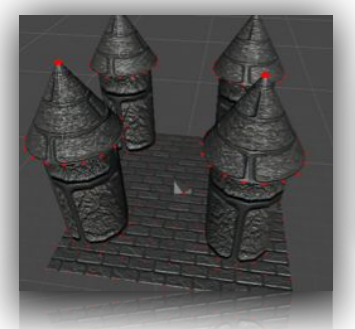
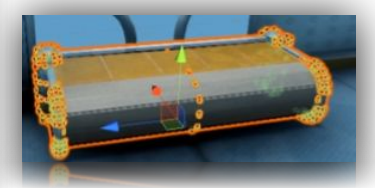
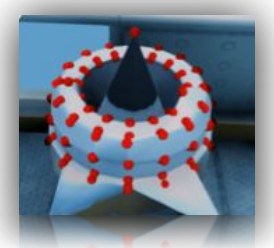
V_TOOL_Vertex_Relax()

Performance & Complexity

Mesh Deformation package has a limitless possibilities, but limited performance. That means, it will go harder with high poly meshes in some cases. Fortunately, you can avoid this problem. Please read the important limitations below that you should know before using MD Package.

Mesh Pro Editor allows you to edit mesh vertices directly in the editor. The component has a condition that if the mesh has more than **2000 vertices**, you will have to use *Performance Saver* [which might be an obstacle in some cases] or you will be forced to optimize your object to lower it's vertex count. Under the 2000 vertex count there shouldn't be a problem in mesh editing. This is not an issue or feature as the **MD Package is not focused on advanced Mesh Editor**. Mesh Pro Editor allows you to edit JUST mesh vertices and you can generate its mesh vertices as objects.

Conclusion: Yes, **you can edit High Poly meshes**, **but not with Mesh Pro Editor**. Mesh Pro Editor can be understood as a 'base station' for basic mesh information, plugin connection and basic mesh editor *[to 2000 vertices and less recommended]*. To edit High Poly models, I recommend using **multithreaded** modifiers like **Sculpting Pro** or **Mesh Drawing**.


FAQ

1. Is MD Package available for Mac or Linux?
Yes, it's available for all operating systems.

2. Can I use MD Package in my mobile game/application?
Yes, you can, but the mobile performance is very important.

3. Can I edit very complex meshes at runtime/ in editor?
Yes you can, but it depends which modifier will you choose to work with. Please read Performance & Complexity slide for more info.

4. Is it possible to export my mesh to OBJ format?
No, it is not possible with MD Package. Use a different external plugin. But you can save your mesh to Assets and create a prefab.

5. Do I need any programming skills to use MD Package?
No, you don't need any programming skills. But if you would like to make some complex operations, you can use the internal API.

6. Am I able to edit a Skinned Mesh object?
Yes you can, but not in the way you think. Skinned Mesh Renderers control your mesh by bones. In MD Plugin the Skinned Mesh Renderer objects have to be converted into Mesh Filters. That means, you will have 2 copies: Original skinned mesh object and editable mesh filter. Original Skinned mesh object will have the mesh source from the editable mesh filter, that means every change made on mesh filter will be applied into the original skinned mesh object.

7. Do I need latest Unity version to use the MD Package?
No, you don't. But it's much safer and better to use latest Unity Version. If you are going to use older Unity versions, the code conversion and compilation will be required and you can get some warning messages.

8. Is Mesh Tracker included in the Mesh Deformation Full Package?
No, it's not included in the package. But the package contains similar modifier called Landscape Tracking GPU.

9. Is MD Package available for WebGL?
Yes, but the multithreading method is not allowed so you have to save the performance on your own. [That means, you might experience some performance issues while sculpting high-poly meshes in built game in WebGL]

10. Does MD Package work with Unity HDRP?
Yes, but shaders won't work as they use Unity standard pipeline.

Thank you

Thank you for your attention. I hope you achieved what you wanted and became more experienced. Please, if you have any questions, contact me **here**.

Extras

If you were interested in the official example games for MD Package, you can check them here *[Just click the gif]*

By Matej Vanco 2020