CLOTH DYNAMICS

Documentation 1.8

Thank you for purchasing Cloth Dynamics!
Your financial support maintains the quality of this asset.

Introductions:

Before you get started some things you should know:

Cloth Dynamics uses **Unity 2019.4.28 or higher** please upgrade your Unity version if have a lower version.
(e.g. 2019.4.9 is not higher than 2019.4.28)
Also install "**Burst**" from the package manager!

Cloth Dynamics uses **custom shaders** to display the cloth simulation and the skinned body. These shaders will be automatically applied to your meshes, if you don't have applied them already to the material.

Depending on your Render-Pipeline these shaders are different e.g. Built-In RP uses custom **Surface Shaders.** With **HDRP** and **URP** the shaders are made with the **Unity Shader Graph** and can be copied and modified.
*(Only **Unity 2019 URP** does not use the Unity Shader Graph, because 2019 does not support Lit shaders in the Shader Graph.)*

If you want to use your **own custom shader** the naming of the shader is important!
If you make a custom skinning shader your shader name must include "**skinning**".
If you make a custom cloth shader your shader name must include "**cloth**".

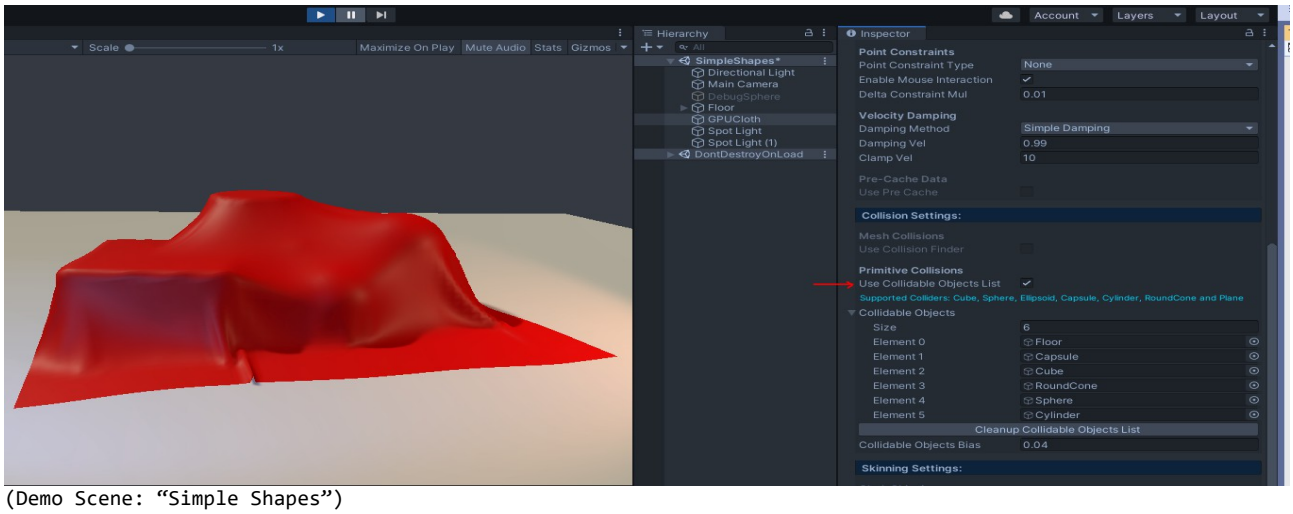
So if you apply a shader, that has this naming, to the material, Cloth Dynamics will not replace the shader any more.

Currently there is no support for double sided cloth meshes, please keep your cloth single sided and the shader will render both sides. If you want to give your cloth a thickness appearance you need a custom cloth shader that has offset passes, like a fur shader or you can use your double sided "thick" cloth mesh as a proxy mesh.

In **Unity 2021.2 or higher** you can set the GPU Skinning script to "Transfer Data", which let you keep the original shader. However the cloth shader will still need it's custom shader, because it gets also data from the cloth simulation and not just from the skinning motion.

1. Create a simple cloth object:

You can create a cloth object by adding the GPUClothDynamics component to your cloth mesh.
Select your mesh's GameObject and „Add Component" → „GPU Cloth Dynamics" to it.



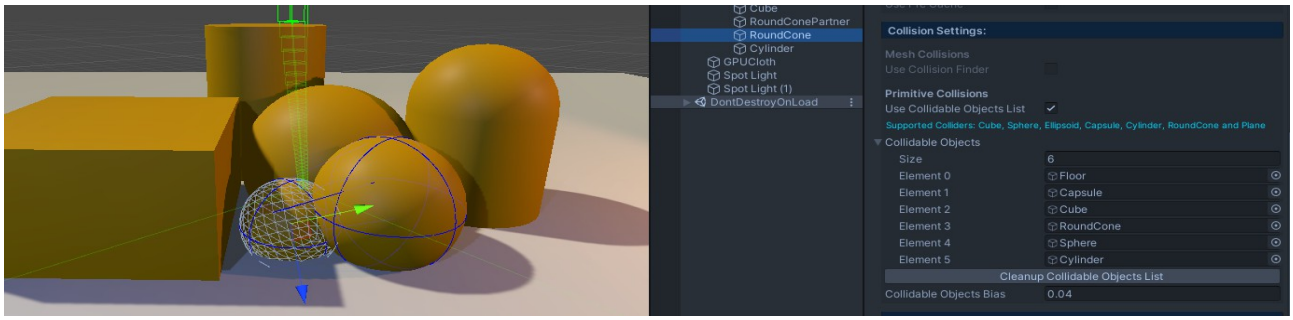
(Demo Scene: "Simple Shapes")

The next step is to add some colliders.
If you don't add any colliders the cloth will just fall through everything.

For a simple collision you can add a „Collidable Object" to the list, which can be found in the Collision Settings of the GPUClothDynamics component.

Create a plane and drag & drop the plane into the CollidableObjects list.
Pro Tip: You can toggle UsePlaneScaleY in the Advanced Mode to use the plane's y-scale-axis as an additional multiplier for the cloth-plane bias or you use a cube collider for the plane object.

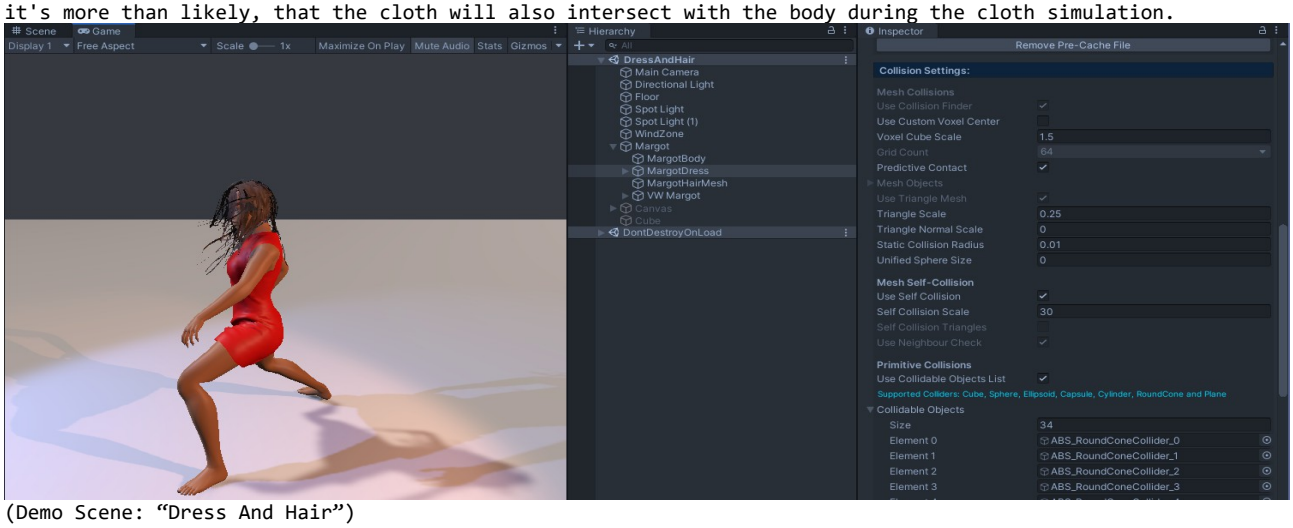
Now, the cloth will collide with a infinite plane, if you add an Unity physics component the cloth system will detect that component and create a corresponding primitive for your collision.
Supported Colliders are: Cube, Sphere, Ellipsoid, Capsule, Cylinder, RoundCone and Plane.



Ellipsoids will be used if you scale a Sphere with different scale-axis values.
RoundConeCollider and CylinderCollider are extra components to get detected by the Cloth Dynamics and RoundCones need another Transform added in the Inspector Settings of the RoundCones component.

Please keep in mind that the CollidableObjects list should be used only for a few objects, because the cloth vertices will iterate through the complete list for each vertex.
2. Create a character cloth object:

If you want to create a character cloth object like a t-shirt, a dress or pants, you also just add the GPUClothDynamics to your mesh.
However, it makes sense to use a skinned mesh cloth, which animates the cloth even if there are no GPUClothDynamics applied.
Important: The cloth should have a skinning, that shows the cloth always on top of the character's body.
The cloth should have the possibility to blend to the original position outside the body when the movement of the character is too abrupt. If the skinned cloth is hidden under the body during the character's animation, it's more than likely, that the cloth will also intersect with the body during the cloth simulation.



(Demo Scene: “Dress And Hair”)

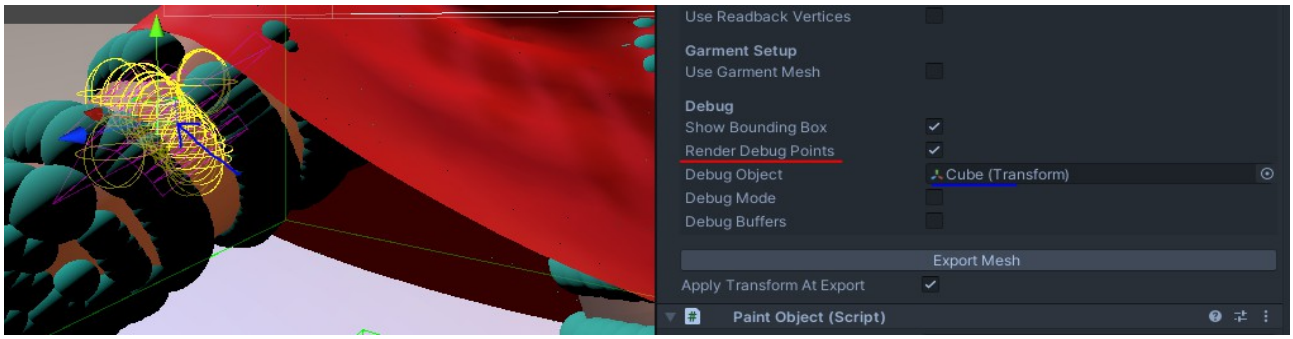
For characters or other more complex meshes you can use the “Collision Finder”, it is an internal octree collision system of the GPUClothDynamics and can be turned on by the toggle UseCollisionFinder in the Collision Settings. Of course, you can also stick to traditional shape-collisions which might be more performant in some cases. For that, you use the CollidableObjects list, at best with RoundCones.

The CollisionFinder let you add meshes as colliding objects. These meshes can have a GPUSkinning component on it, to transfer the skinned animation of the body mesh to the cloth object. You can also use a DualQuaternionSkinner component, which has a more natural skin-deformation algorithm.
It's also possible to add the cloth itself for self collision (or you switch on the self collision toggle).

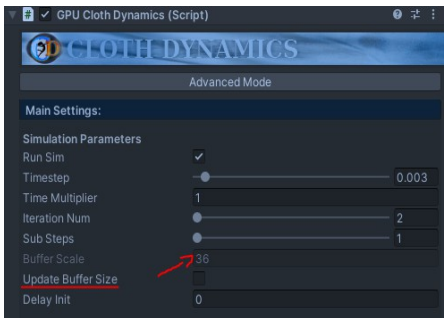
There is also another option: you can drag & drop the parent object of the skinned body in the mesh objects list, if it has a AutomaticBoneSpheres component applied. This component creates spheres based on the bone-mesh structure of your character automatically. It mainly works with the humanoid animation type of your rigged character. These spheres will be used as collision spheres in the octree system of the CollisionFinder when you add the GameObject to the collidable objects list of the GPUClothDynamics component.
You can also export the created spheres as Unity spheres or export RoundCones, which are more efficient than sphere colliders. They will be automatically added to your active cloth objects in the collidable objects list.

Currently the CollisionFinder only supports cloth point-to-sphere collision on mesh surfaces. So the mesh vertices (e.g. of a character) will be converted to sphere points. You can also activate triangle collision, this will create a sphere collision on each triangle surface, that moves along the triangle surface to the colliding cloth vertex point.(In the picture below you see the “sliding” spheres in yellow.)

You can visualize the collision spheres by turning on the RenderDebugPoints option in the Debug Settings. Add a GameObject in the DebugObject field, it's position will visualize the current voxel size and the colliding spheres of this voxel.



The BufferScale will be affected by the UpdateBufferSize if you turn it on the simulation will track how many collisions are happening during runtime and will adjust the BufferScale for you. You just have to copy the value after play-mode to your BufferScale.



This is not done automatically to prevent accidentally high scale buffer sizes, which could happen if you use very high scaling values for you collision spheres like the Vertex, Triangle or Auto Sphere Scale values.

Hint: Most of the parameters have tooltips on it, when you hover with the mouse cursor over them.
(Does not work in play-mode!)
Therefore, only a few parameters will be explained here, also some parameters are self-explanatory e.g. Gravity or RunSim.

Main Settings:

Timestep defines the sub steps per frame, it iterates through the simulation process and is also depended on the delta time provided by Unity.
Timestep will affect how precise your collisions will be and how good the constraints between the cloth vertices will be solved.
Time Multiplier changes the simulation time and can be used to speed up or slowdown it's behaviour.

IterationNum is basically the same as Timestep but inverted, that means, higher values will create better collisions but lower values make the simulation more performant.

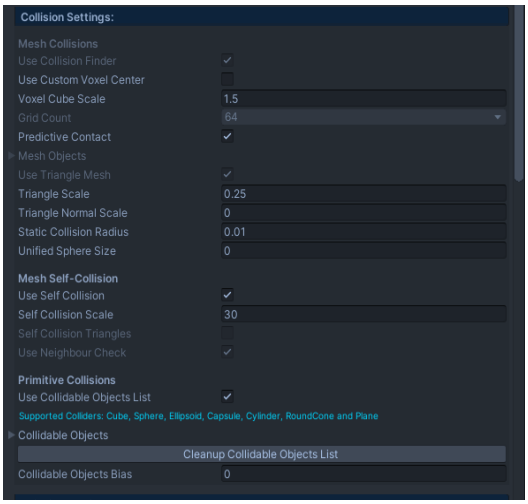
SubSteps are subdividing the full iteration loop and should be only set higher if necessary, because it's very performance heavy.

Static friction will be applied on all colliding particles/spheres.
Dynamic friction will affect collisions with objects from the CollidableObjects list.
You can also add a ClothFrictionCollider component to customize the static friction of the primitive colliders.

As mentioned before the BufferScale set the size of the main buffers for the CollisionFinder and should be set low.

You can now also automatically pre cache the created cloth mesh data, this will reduce the loading time. If the mesh changes (or Unity version changes) you have to remove the cache and it will create a new one for you.

Collision Settings:



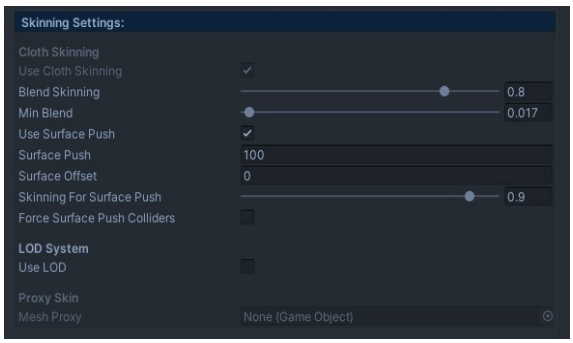
The VoxelCubeScale defines the Size of the green cube in the editor view. It handles the size for the octree collision system, which will be used if you activate the CollisionFinder.

The Vertex and TriangleScale increases the radius of the collision spheres.
The NormalScale moves the centre of these spheres inside the mesh in inverse normal direction.
(Negative values will move them outside of the mesh body.)

The StaticCollisionRadius will help to detect colliding spheres earlier if they come in their radius, however, due to the fact that the system is voxel based and limited by the voxel detections, it does not have a big impact. Normally, lower values will be better for the performance.

The SelfCollisionScale is default set very low, but you can increase the scale until you see some jittering between the vertices of the cloth and than slowly decrease it again. Values between 10 and 20 should be fine for a human-sized cloth object that uses a ~1.8 unity-units high body.

Skinning Settings:

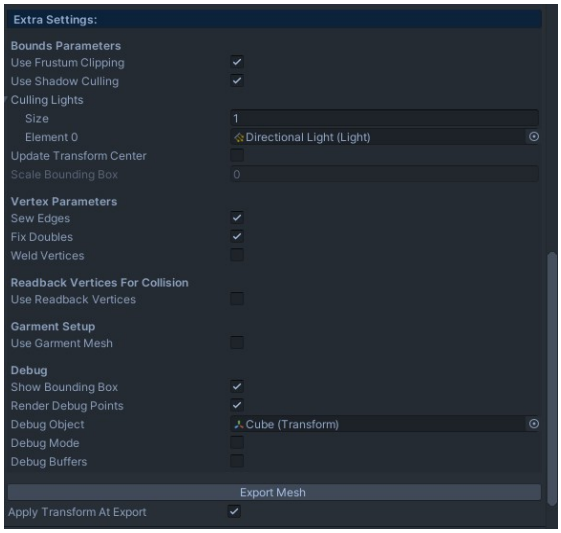


If you have a skinned mesh as a cloth object, the cloth skinning is working and you can blend from cloth sim to skinned cloth animation.
You **should** also add a PaintObject and colour the fixed vertices red. The red vertices will tell the cloth sim to use the Skinning, if they are black, it will use the cloth simulation for these black vertices.
The SurfacePush moves the vertices, that accidentality fall inside the body mesh, to the outside surface of the body.
You need a body mesh in the MeshObjects list or a CollidableObject that can be used as a surface.

SurfacePush is recommended if you have a character that moves very fast or jumps frames. It's important, that the skinned cloth does not intersect with the mesh body. You can test it when you set the MinBlend to 1, that will turn off the simulation and use 100% skinned cloth animation.
It's also possible to add an AutomaticSkinning component, but this is not recommended because it tends to make visual mistakes in the skinning, it's better to setup your cloth skinning in a modelling program.

LOD System can blend from the skinned mesh to the cloth simulated mesh, it has a distance value LodDist that tells you how far the camera needs to move away from the cloth to only see the skinned cloth, not the simulation e.g. a value of 8 means that when the camera is 8 units away you will only see the normal skinning animation. This helps to improve the performance, when using many characters. The LODCurve set the interpolation for these two states (sim → skinned mesh).

Proxy Skin can be activated by drag- and dropping a proxy mesh into the MeshProxy slot. It basically transforms the vertex positions of the proxy mesh with the vertices of the cloth mesh. Because you normally have more vertices on the proxy mesh the shader needs to blend them with the original cloth mesh, therefore you can adjust the weighting with the different weight values. The SkinPrefab will be generated automatically as a file in the Resources folder. If you click on "Remove Skin Prefab" you will only remove it from the inspector, you can still assign it again or overwrite it. Be careful not to overwrite other skin prefabs that have the same name. In that case you need to rename your cloth GameObject's name.



Extra Settings:

The Camera Frustum can be used to clip/stop cloth simulations that are outside of it. Also light(s) can be added to use it's shadow direction to stop clipping if the shadow is still visible.

You can also use UpdateTransformCenter to reposition the bounding box of the cloth. This is only useful for "free" cloth objects. It is possible to scale the bounding box like a cube e.g. set ScaleBoundingBox to 1, if you set it to 0, it will use the original size.

SewEdges will weld the unconnected overlapping edges,
WeldVertices will weld the complete mesh.
FixDoubles removes the double vertices from the simulation.

You can also read back vertices to the CPU. They will be created as collider spheres during runtime.
This happens async and might have a small delay.
UseGarmentMesh shrinks the green painted edges to a fixed length so you can "sew" cloth parts together.

DebugBuffer shows you the collision buffer size and updates the current usage of it to check if the size is big enough to fill all collisions in it.

RenderDebugPoints should be used together with the UpdateBufferSize to get a visual feedback of the runtime state of the CollisionFinder.

The button "Export Mesh" let you export the current state of the simulated mesh.
It can be reused as a cloth starting point. This is especially helpful for the Garment Setup.

Create Garment:

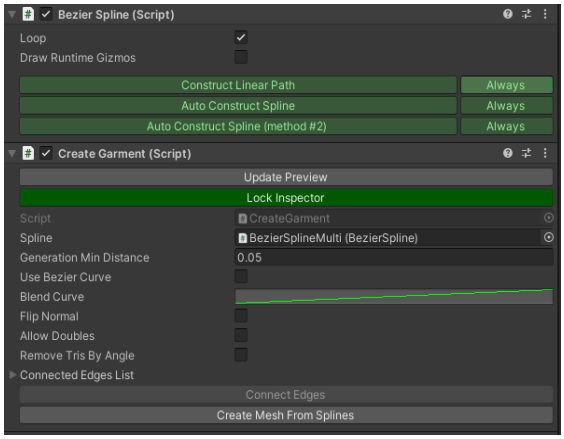
You can create a simple cloth object by combining two or more bezier splines.

First create a spline from the Menu "ClothDynamics" and add a CreateGarment component to convert the spline into a triangulated mesh.

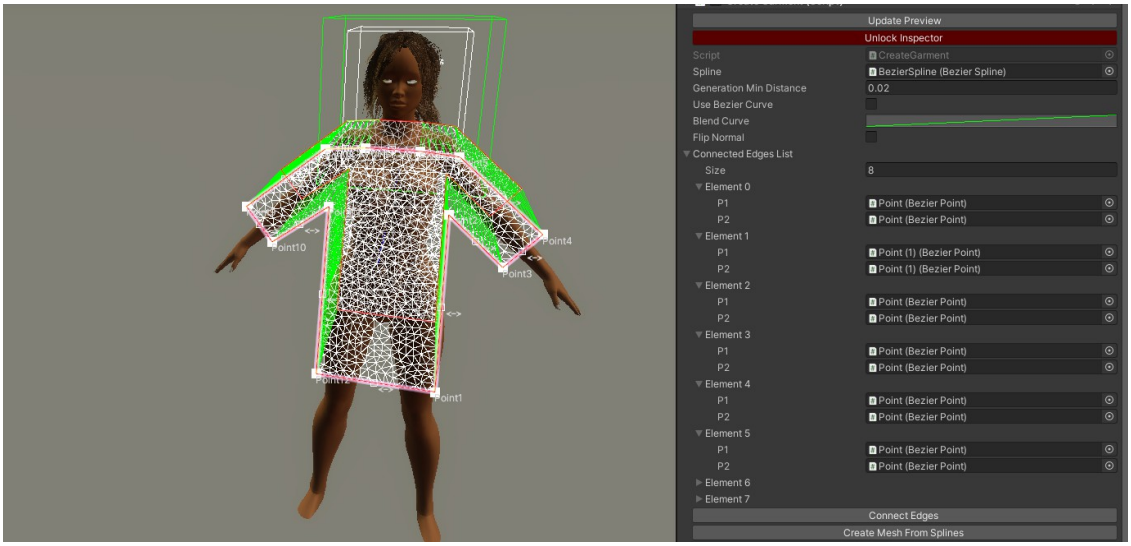
Use "Lock Inspector" and select the edges with Shift + Mouse Click.

You can connect edges of different splines by selecting the edges in the editor view and click on "Connect Edges".

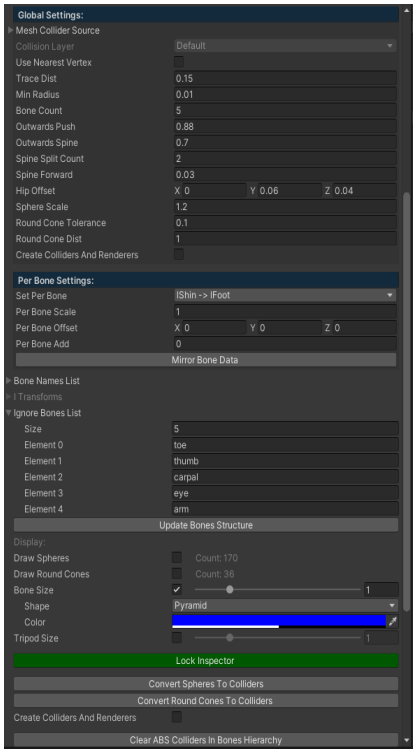
If you finished connecting all edges you can create the full connected cloth mesh by clicking on
"Create Mesh From Splines".



It's important to check the blue normal on the surface in the editor view, if it points in the wrong direction you can flip the surface with the toggle FlipNormal.



Create Garment is a very simple script, it's not meant for runtime creations and also can only handle very simple cloth structures. If you want to make more complex cloth meshes I recommend using an external app!



Automatic Bone Spheres:

Important: This component only works, if your character model does **not** have Optimized GameObjects in the rig import settings activated! It also needs to be set to **Humanoid**!

This component creates spheres automatically inside the character's body. It uses the bones structure for the positioning and the mesh surface for the radius size of the spheres.

TraceDist let you set the distance how far the ray-casting should start outside of the mesh body. It will ray-cast to find collision points on the surface and calculates the radius from the average.

BoneCount defines how many spheres per bone should be created. You can add or remove spheres individually per bone in the “Per Bone Settings”.

In the “Per Bone Settings” you can also select the individual bone and adjust the scale and offset.

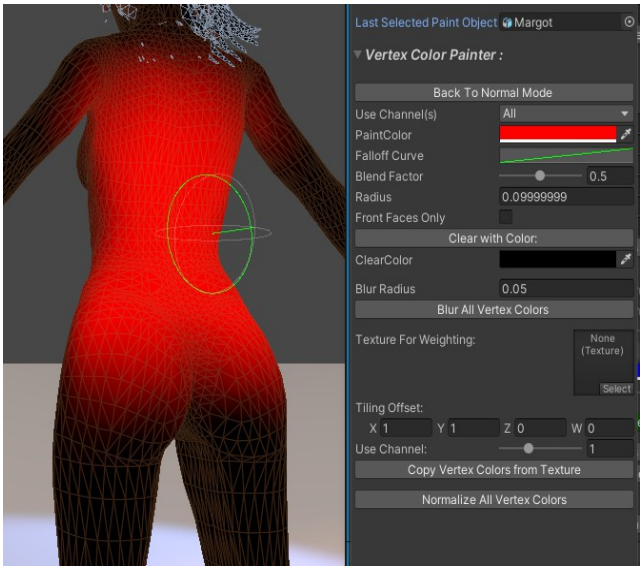
If you press the “Lock Inspector” button you can select each bone in the editor view. This works only if you turn on the BoneSize Toggle to render the bone structure in the viewport.

The IgnoreBonesList helps you to exclude bones that are not needed e.g. if you only have a cloth skirt you don't need the upper body, so you add the spine name to your ignore list and the bone renderer will only display and use the legs.

If you finished your bone-sphere setup, you can drag & drop the GameObject that has this component applied to your MeshObjects list in the GPUClothDynamics component.
Alternative, you can convert the spheres to Unity's sphere colliders or convert them to RoundCones, which are two spheres that are connected.

The CollidableObjects list will have these objects added automatically. You can also find them in the bones hierarchy of your character model.

Vertex Color Painter:



The Vertex Paint Editor Window helps you to define the areas where your cloth will collide.

All vertices that are **red** will be included in the cloth collision system.
All **black** vertices will be ignored, they automatically reduce the buffer/memory size of the simulation and therefore increases the performance.

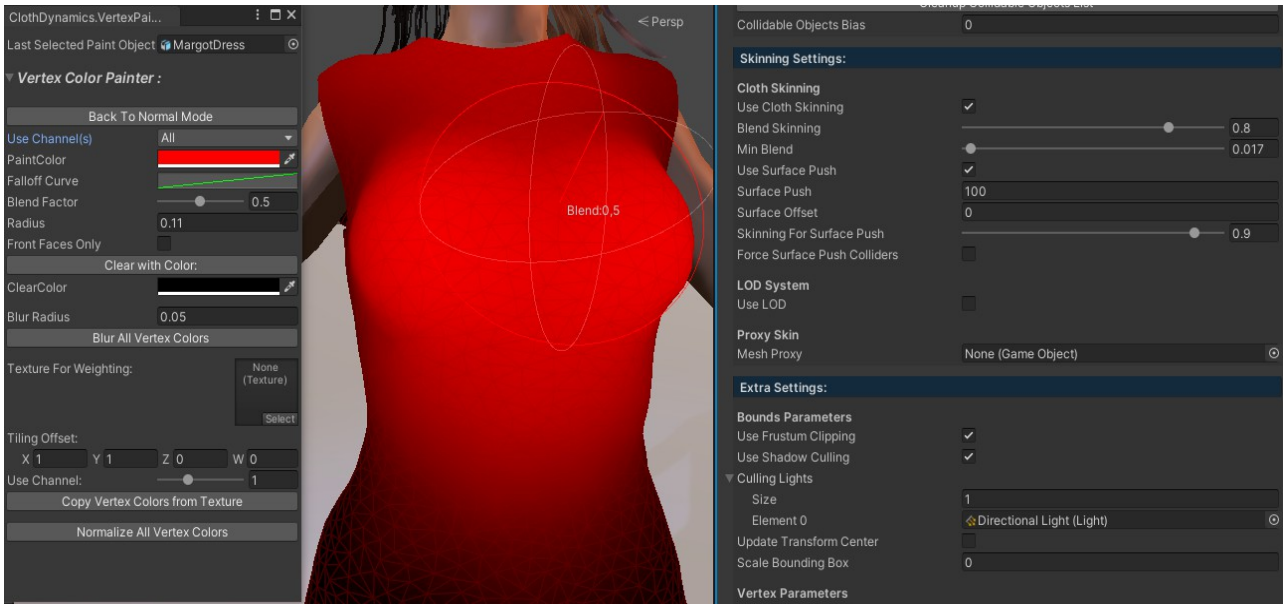
That's why it is highly recommended to always paint the colliding body to exclude the unnecessary vertices.

The cloth can also have a PaintObject applied. This is useful if you want to determine which vertices (**red**) are blending to the skinned mesh animation and which ones are fully simulated by the Cloth Dynamics (**black**).

Don't forget to add a GPUSkinning component to the cloth object to transfer the skinning data to the cloth simulation. You need a SkinnedMeshRenderer on the cloth or an AutomaticSkinning component.

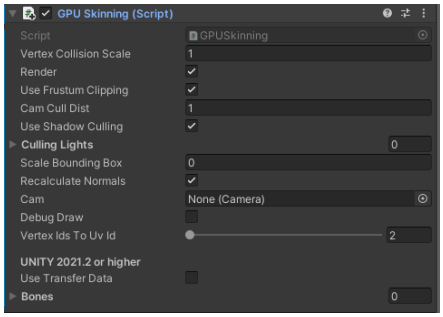
Controls:
If you just press or drag the LeftMouseButton you can paint the selected colour.
If you press Alt + LeftMouseButton you can subtract colour from the vertex, or in other words, paint black.
You can use Shift + MouseWheel to set the Radius size in the editor scene view.
If you use Control + MouseWheel to set the BlendFactor, 1 means the skinning will be active to 100%.

Keep in mind, that the painting is radius based, so also back-facing vertices will be affected, if you paint with a high radius value. You can turn on "**Front Face Only**" if you want to only affect the visible faces!



(Here the red painted upper body part will be controlled by the skinning, the black part by the cloth sim.)

GPU Skinning:



The GPU Skinning script should be applied to the body meshes and the cloth meshes, when they are using a Skinned Mesh Renderer.

Normally Cloth Dynamics does this automatically for you but in some cases you want to adjust them so you apply them before you hit Play.

“Use Frustum Clipping” culls the mesh depending on your Main Camera frustum. The same goes for **“Use Shadow Clipping”**, however it also uses a light source to cull the shadow correctly. This only works with a Directional Light.

If you see that your character disappears, even it's in the view frustum, you can increase the **“Scale Bounding Box”** value. If you leave it at zero it will use it's default bounding box, but it will not update the bounds during movement, because only your GPU gets the movement data. So you might want to scale the bounds to make sure the character is always visible.

With **Unity 2021.2 or higher** you can now also “Use Transfer Data” and keep the Skinned Mesh Renderer alive. However in some cases (normally a cloth mesh) you need to replace the Root Bone of your Skinned Mesh Renderer with the lowest bone in the hierarchy e.g. the MargotDress mesh has the Root Bone set to hip, this will not work because the transformation data does not fit any more, so you have to replace it with the “VW Margot” bone, which is the actual root bone of this character. Now you also need to reset the transformation to zero. This is a beta version feature of Unity, so it might not work in all cases.
Use Debug Draw to see if the Skinning is applied correctly!

GPU Blend Shapes:

This is an Addon script for the GPU Skinning script and only works with it; Dual Quaternion Skinner is currently not supported, but has it's own way of using blend shapes (however, it doesn't always work well).

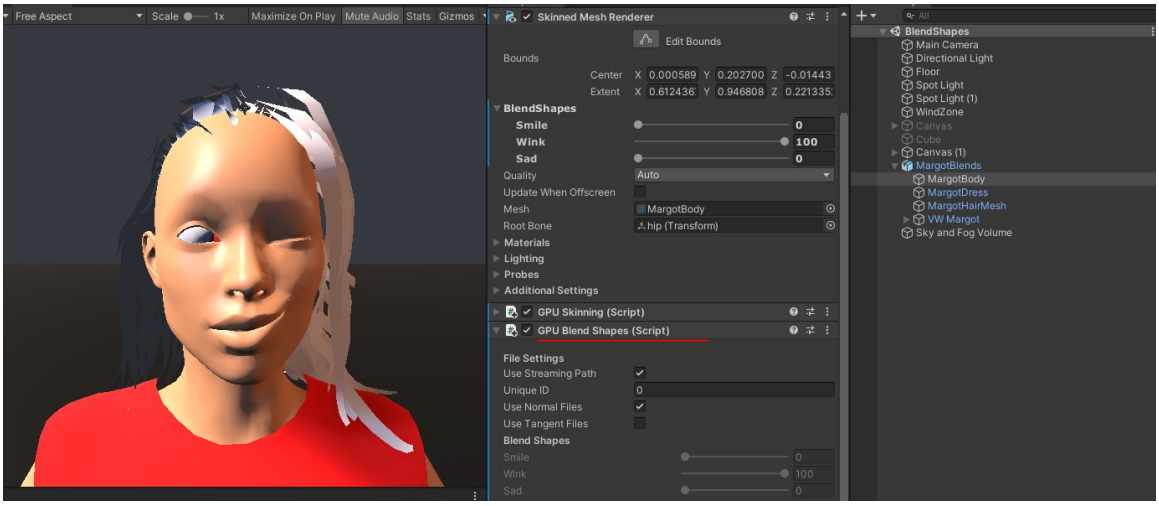
GPU Blend Shapes picks up the blend shapes of the Skinned Mesh Renderer and converts the data to textures, which the GPU can read via shader. The script writes the cached data to the “StreamingAssets” folder. *Normally Unity automatically copies these data to your build app, but if something goes wrong you can check this folder and see if still there.*
If you don't want to use the StreamingAssets folder you can toggle this and it will use a Resources folder. This might be more secure. The data would be saved in the folder: “ClothDynamics/Resources/BlendShapes”.

The “**Unique ID**” can normally be set to 0, but if you want to create unique blend textures you can set a different value here and it will create a new texture set for only this mesh, otherwise these textures will be shared between the same meshes.

The “**Use Normal Files**” will add normal maps, this is recommended! Normally you don't see much visual changes with the tangents, that is why they are disabled as default.

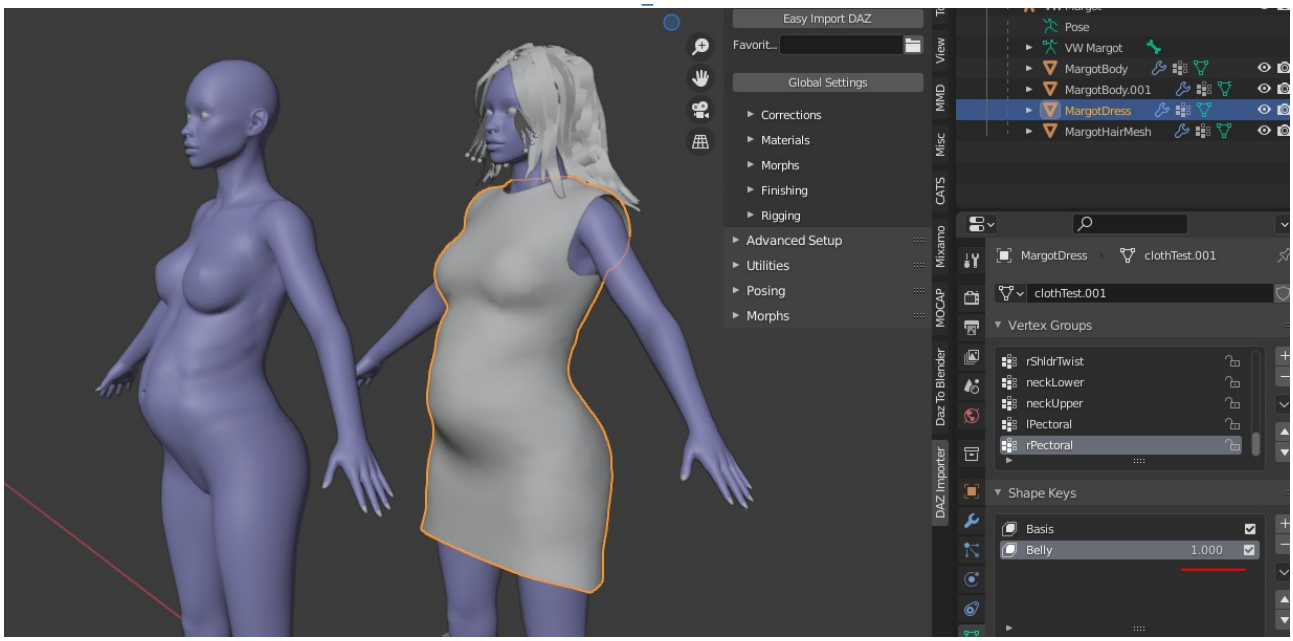
At runtime you can modify the Blend Shapes values, via script or inspector. Just wait a second to let CD convert the skinned mesh to a mesh renderer.

```
public class Test : MonoBehaviour
{
    WaitForSeconds waitForSeconds = new WaitForSeconds(1);

    Unity-Nachricht | 0 Verweise
    IEnumerator Start()
    {
        yield return waitForSeconds;
        var blends = GetComponent<GPUBlendShapes>();
        blends.SetBlendShapeWeight(blends.GetBlendShapeIndex("Smile"), 100);
        blends.SetBlendShapeWeight(blends.GetBlendShapeIndex("Wink"), 0);
    }

    Unity-Nachricht | 0 Verweise
    void Update()
    {
        var blends = GetComponent<GPUBlendShapes>();
        blends.SetBlendShapeWeight(blends.GetBlendShapeIndex("Smile"), (Mathf.Sin(Time.time)+1)*50.0f);
    }
}
```

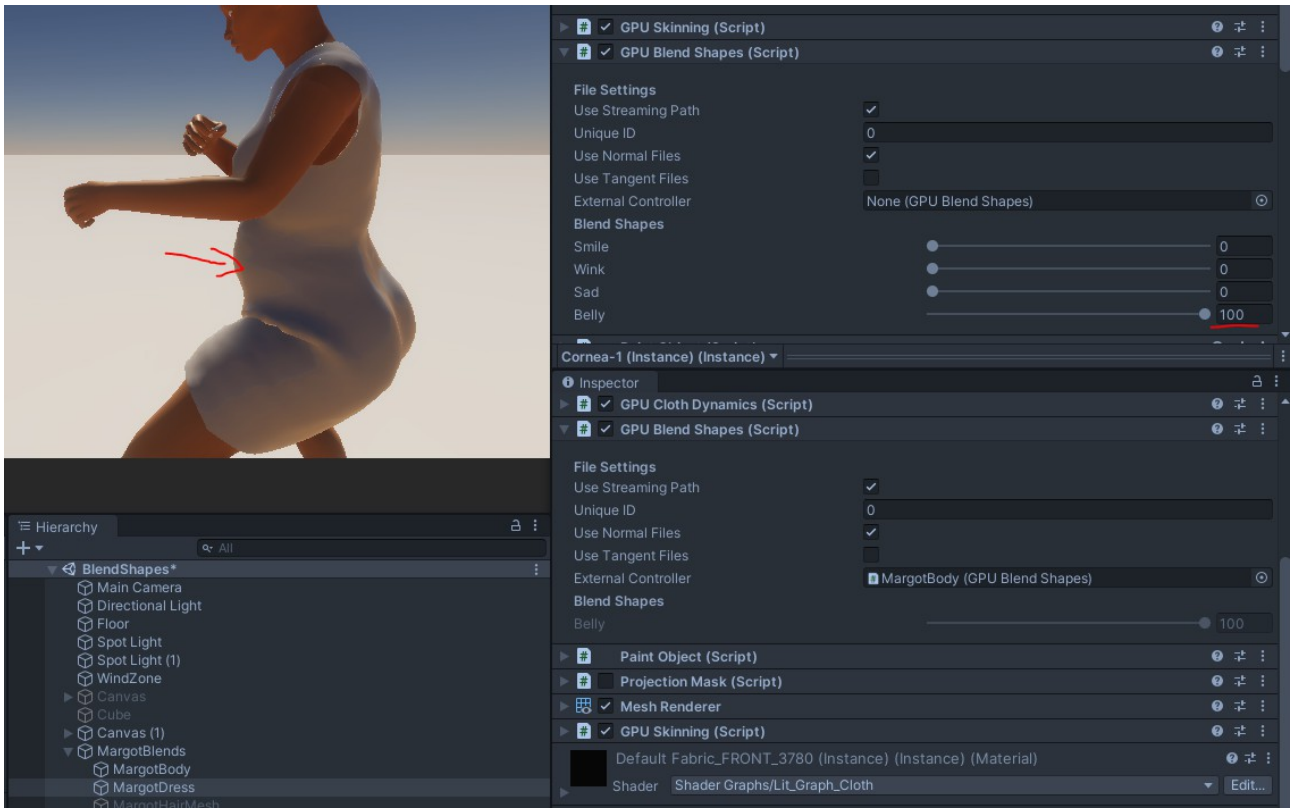


GPU Blend Shapes with Cloth Interaction:

If you want your Blend Shapes to interact with your cloth you need to add this Blend Shape also to your cloth mesh. The follow screenshot shows you the Blender file of the MargotBlends character, which is also included in the examples folder, under MargotBlends_BlenderFile.zip. You can see that the character's cloth also has the belly deformation, which is important so the cloth can blend to the deformed position when using “Cloth Skinning” or “Surface Push”.



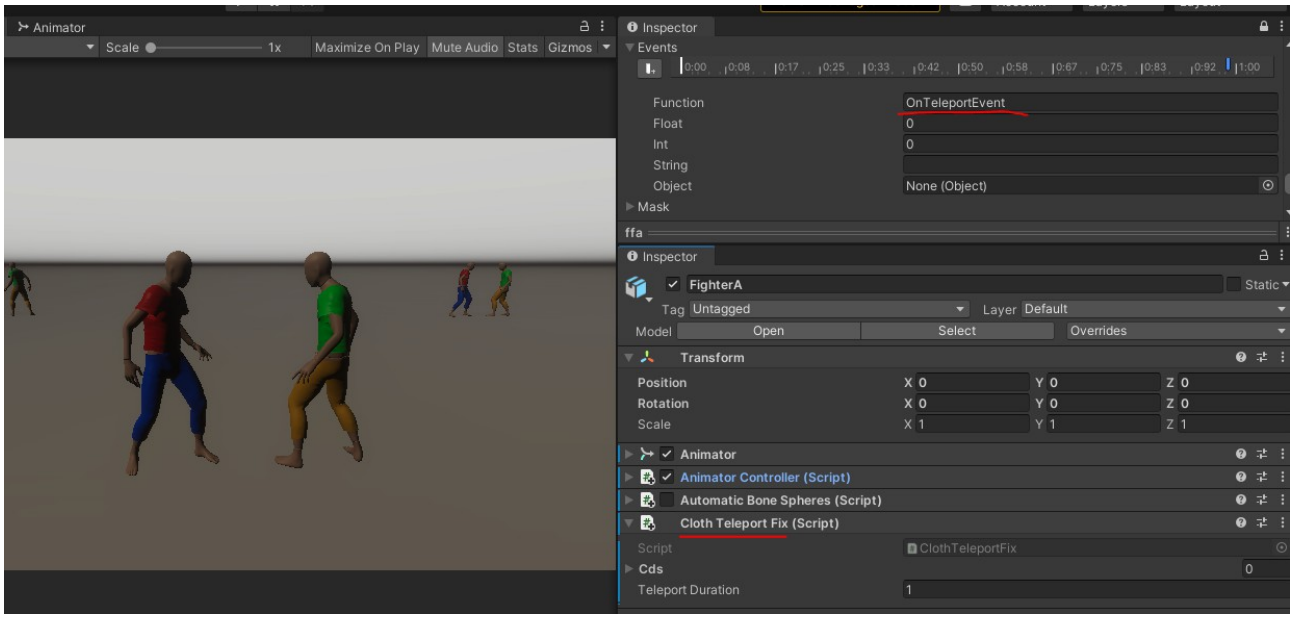
After exporting the Body and the Cloth with the blend shapes you just need to add the GPUBlendShapes script to your cloth mesh, that's it! When you edit the Blend Shapes at runtime, the cloth blend will automatically follow the body blend. See the **BlendShapes scene** example:



Be aware that quick blend changes can destroy the cloth-body overlapping situation, so it is always useful to create a script that blends smoothly over time.

Cloth Teleport Fix:

The Teleport Fix is a simple way to get rid of the frame jumps, that happen when the position changes abrupt after 1 frame. Normally the cloth will be stretched between the positions, so the script will freeze the cloth for the time the position changes. Therefore the animation triggers the “**OnTeleportEvent**” before the change happens.



Projection Mask:

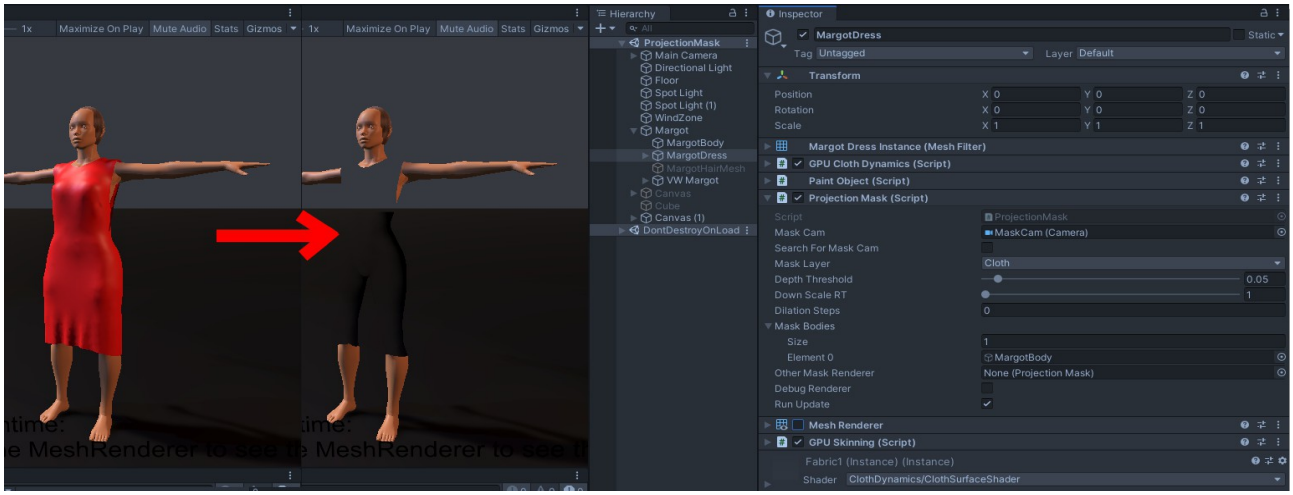
The **Projection Mask** script can hide unwanted overlaps between the skin of the body mesh and the cloth's surface.
Sometimes it can happen that you see small parts of the skin come through the cloth mesh, in that case **Projection Mask** can help you.
You need to add the script to your cloth object and set the Mask Layer to a new unused layer. Best practice would be to create a new layer in the layer manager of Unity. I would call the layer “Cloth”. This layer is used by a generated camera to render the mask for your body meshes.
You can add a body to the “Mask Bodies” or let the script find them.
The Mask Cam can be generated during runtime and then copy and paste in after Play Mode stops. You should then attach the new Mask Cam as a child of your Main Camera.
*Also when you are using **HDRP** you need to set the Background Color to white and the Volume Layer to None. This can only be done manually (sorry!).*

After this, you can continue to add the new GPUSkinningMask shader to your body materials, this also happens automatically but it's better to set it manually and adjust the material/shader.
*Sometimes it happens in **HDRP** that the Emission Value is set to 1 even if it's set to 0 in the original material. If your material appears to be white, check this value in the material properties. In older unity versions, there is a bug, that the emission colour overwrites the base colour. Just use the emission colour but turn of the emission value, if you don't need emission. Make sure you turn on **Alpha Clipping**!*

If you use Debug Renderer you should see an extra blue cloth in the Scene View, which should be the same cloth with the same transforms as the original one.

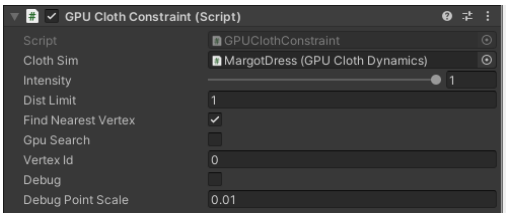
Now you can check if the “Mask” is working correctly, by turning off the Renderer of your cloth mesh. If the body underneath is invisible you did everything right! You can now adjust the Depth Threshold, the higher the value the more cloth gets cut out.

You can also play with the Down Scale value, this helps with performance but makes it less precise. Dilation Steps adds extra pixels to the mask so you can shrink the mask to the centre, this is useful, when you see the invisible body part under the cloth, even the cloth is visible.



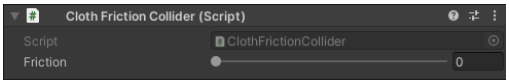
GPU Cloth Constraint:

The **GPU Cloth Constraint** script can be applied to another Game Object (e.g. Sphere), not the cloth object, and works as an external constraint that will move the nearest cloth vertex to it's position. The Intensity value can be used to blend between “no impact” or “pinned” (0 → 1). If you keep “**Find Nearest Vertex**” on, it will always try to find the nearest vertex when you switch from 0 to 1. If you turn it off it will keep the last found cloth vertex. “**Dist Limit**” is the radius in which the script will search for the nearest vertex. The script only looks for it in the applied “**Cloth Sim**” mesh object.



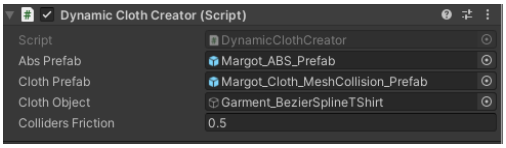
Cloth Friction Collider:

The **Cloth Friction Collider** script let you set the friction of a primitive object. This only works with objects that are in the CollidableObjects list!



Dynamic Cloth Creator:

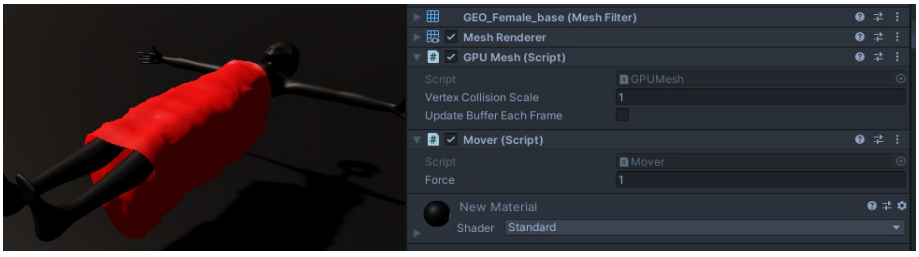
The **Dynamic Cloth Creator** script will add a GPU Cloth Dynamics script to your cloth mesh at runtime. This is currently an experimental feature and has some limitations. Basically you can add the script to the root of your character (normally where the Animator component sits) and the script will create the cloth for you when you enter Play Mode. You need to setup an ABS (Automatic Bone Spheres) script of your character and save it as a prefab. You also need a prefab of your GPU Cloth Dynamics script. Finally you can set the cloth mesh object, which should be a child of your character object. See the example scene “DynamicClothCreation” for more details.



GPU Mesh:

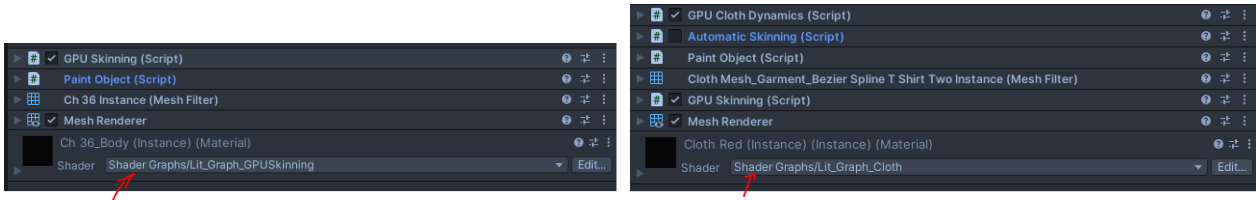
The **GPU Mesh** should be used instead of GPU Skinning when your body mesh does not have a Skinned Mesh Renderer or you want to use other meshes e.g. furniture as collision objects. The GPU Mesh is meant for static collision objects, but you could deform the vertices and use the “**Update Buffer Each Frame**” option. However, this will affect performance because it will write all the vertex data each frame from CPU to GPU.

The “**Vertex Collision Scale**” can be used if the created collision spheres are to big or small, for the collision. You can check the size visually when you turn on the “**Render Debug Points**” option in the Extra Settings of your Cloth Dynamics inspector.
There is also an example scene called “GPUMeshTest”.



Build Settings:

Before you build your project you should include the used cloth shaders in a Shader Variant Collection.

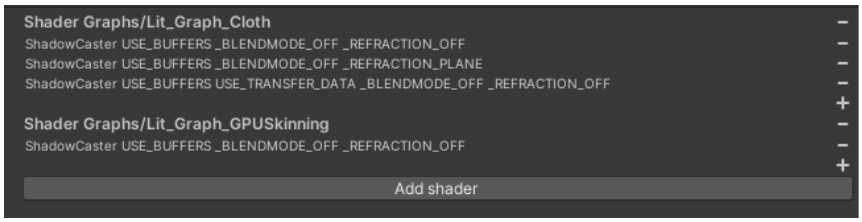


To find out which shaders you are using hit the **Play** button and select the body GameObject and the cloth GameObject and check which shaders are applied to their material(s). Depending on your Render-Pipeline they can differ.

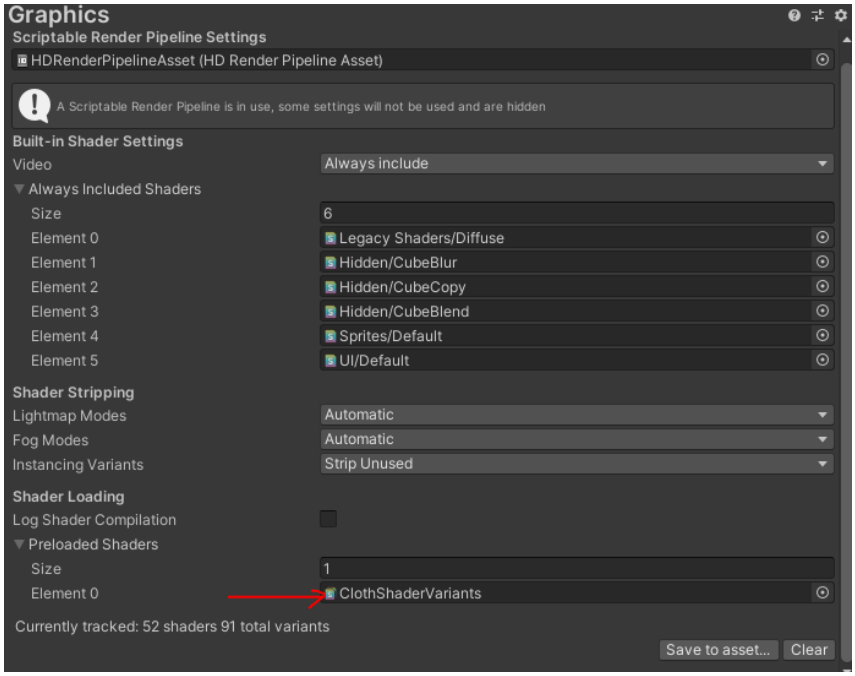
Right click somewhere and create a **"Shader Variant Collection"**. You need to add your shaders here and the Keywords that you are using e.g. USE_BUFFERS..., USE_TRANSFER_DATA..., USE_BLEND_SHAPES... depending on what your are doing, normally you only need UseBuffers; TransferData is for the new 2021.2 feature and blend shapes are only needed for the GPU Blend Shapes script, which also might need normals and tangents defined.

You also have to add the "...Mask" Shaders if you are using a "Projection Mask" component on your cloth.

This is how a Shader Variant Collection looks like for HDRP with LitCloth and GPUSkinning:



After that you can drag and drop it to the Preloaded Shaders list in the Graphics tab:



Now you can build your project, I hope everything works out for you!
If you have questions contact me via forum or email, thanks!

FAQs:

1. My cloth starts to wiggle and the collision with my body mesh seems wrong, what can I do?
Try to increase the BufferScale or press UpdateBufferSize at runtime. If that doesn't work set the bending and compression stiffness to 0 and slowly increase them to test how much they affect the cloth behaviour. It's also important that the distances of the mesh-edges have more or less the same length.
Use the CreateGarment component or a 3rd party tool to create a proper cloth mesh e.g. Blender's "Garment Tool" or the software "Marvelous Designer". Furthermore, activating SimpleDamping might help.

2. Does the cloth sim run on moblie devices?
Yes, a part of the cloth sim can run on **modern** mobile devices (Android was tested). But you have to turn off the CollisionFinder, because it's currently too memory heavy for the mobile hardware, maybe at some point there will be more support for mobile, but currently the main focus is on PC and console hardware (not tested yet).

3. My cloth-mesh collision is not as good as expected, what can I do?
Decrease the Timestep value and try using triangle-mesh collision, also play with the stiffness values. It also might help if you change the VoxelCubeScale and the GridCount. Combining different collision system helps too, to improve collision results e.g. using MeshObjects together with CollidableObjects.
Toggle Predictive Contact, but be careful this might also affect the performance.

4. Sometimes my cloth starts to fly away or has "ghost" forces applied, what's going on?
This could be caused by a too small VoxelCubeScale value, if you need to double this value you should consider using the next higher GridCount value to keep performance consistent, higher GridCount values will cause more memory allocation but might be worth trying.

5. Can I adjust the thickness behaviour of my cloth object?
Currently not directly, but you could fake a heavy or light behaviour by changing the gravity value per GPUClothDynamics component during runtime.

6. Can I use high-res cloth models with this tool?
Yes, but you should consider using a low poly cloth version of it and instead use the high-res cloth as a ProxyMesh, which can be found in the Skinning Settings. The low poly cloth will be invisible, but it will control the high-res model underneath. A ProxyMesh also makes sense if your cloth model has extra parts attached or the edge-distances are not uniform. In that case, the high-res cloth should have a uniform triangle mesh as a control mesh.

7. Can this asset be used as a hair simulation system?
Yes, there is also an example scene, however it's not directly supported yet. You can also try to use a low-res cloth hair and use high-res hair meshes as a ProxyMesh. There might be a better solution coming soon.

8. Is the physical simulation accurate?
No, it's like any other "Position Based Dynamics" simulation only an approximation and needs tweaking to look "real".

9. Is this currently the fastest cloth simulation?
The main reason why this asset was created is the fact, that other cloth solutions on the market are currently too slow, independent of CPU or GPU approach. Of course, you can also make this asset quite slow, if you use the "wrong" settings or too high poly objects. However, some solutions might have better quality in their collision behaviour, this asset is focused on performance and will be used in a production title.

10. Can I use LOD to improve performance?
Yes, the GPUClothDynamics has a simple LOD system that blends from cloth simulation to skinned cloth animation when the camera moves away. Furthermore, the cloth simulation can be culled by the camera frustum.

11. Does this asset work with OpenGL?
Yes, but you have to allow 'unsafe' code in the Player Settings of your Unity Editor and add the scripting define symbol: *CD_UNSAFE_CODE* . This is not mandatory, but highly recommended, if using OpenGL.

12. Does this asset work with **WebGL**?
No, because WebGL has some memory limitations that CD currently can't match, maybe in the future.

13. Can I create cloth physics at runtime?
Yes, but this is limited. You can use the **Dynamic Cloth Creator** and setup some prefabs to make it work and also have a cloth mesh, that fits the character. See the "DynamicClothCreation" example scene for more informations.

14. Does CD support UMA models?
No, because UMA is a procedural generated character tool, so it would only work when you use the Dynamic Cloth Creator. However, this script is limited so I would not recommend it. Also there might be some meshes that work and others not. There are also other procedural character tools, to support them all would take too much time, nevertheless I'm planning to improve the Dynamic Cloth Creator, so UMA might be an option in the future, but it will never be fully supported. If you know how to code, you can adjust the Dynamic Cloth Creator for your own needs and might get the UMA system to work.

15. Is it possible to use the HDRP silk shader with this?
You can change the silk shader very easily in the Shader Graph, you just need to add the ClothSubGraph.

16. Does it support VR?
Not tested, if you want to try it out, just buy CD and if it doesn't work for you, you can get a refund.

17. Does CD run on any mobile device?
No, it only runs on modern mobile devices and also can not use the Mesh Collision System, it only supports primitive objects like spheres or cubes. It's been tested on Android and iOS.

18. How good is the performance?
The performance is better than other CPU based cloth systems, because it mainly uses the GPU to calculate the cloth simulation.

For more informations checkout the tooltips on the parameters in the inspector or contact the developer.
Thank you for supporting this asset,
it wouldn't be possible without you and the help and contributions from other developers.

****** BURST ******
To enable Burst compilation install Burst from Unity Package Manager.
Note that Burst does not support cross-compilation in some cases (e.g. Windows to Mac OS).

The main reference for this project is based on the following github repo:

https://github.com/Ninjajie/Fusion

Other third party tools that are used in this project and also refer to the MIT License:

Bezier Solution: https://github.com/yasirkula/UnityBezierSolution

Delaunator: https://github.com/nol1fe/delaunator-sharp

Skinner: https://github.com/keijiro/Skinner

GPU Skinning: https://github.com/sugi-cho/Unity-InternalSkinningTest

Dual Quaternion Skinner: https://github.com/ConstantineRudenko/DQ-skinning-for-Unity

OpenGL Async Readback: https://github.com/yangrc1234/UnityOpenGLAsyncReadback