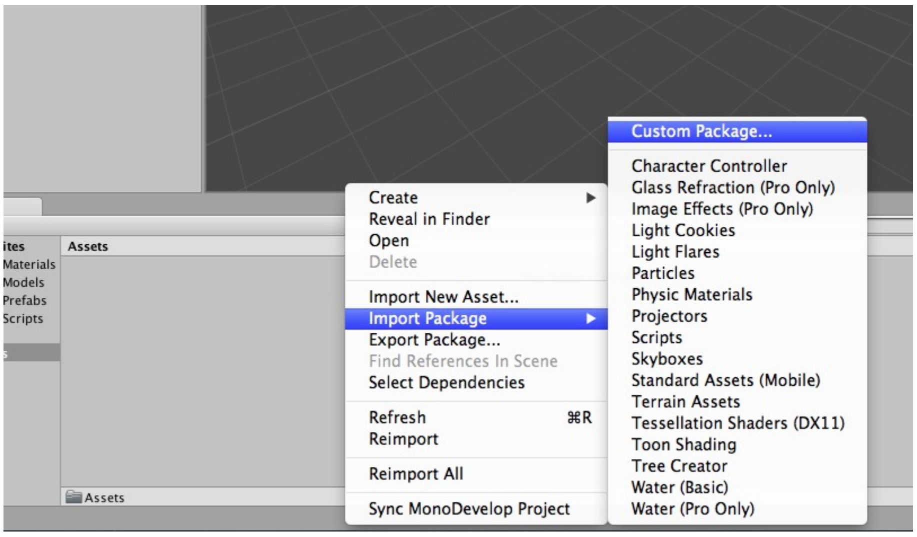
Quickstart Guide

1.- Include the Obi package in your project. Right click in the project window, “Import Package”   “Custom Package”, then select Obi.unitypackage.



2.- Move the “Obi/Editor Default Resources” and “Obi/Gizmos” folders to the Assets folder in your project.

3.- In your scene hierarchy, select the GameObject you wish to simulate as a cloth object. Then go to “Component” -> “Physics” -> ”Obi” -> “Obi Cloth”.

4.- You're done!

Support / Contact

If you have any suggestions, questions or issues, contact the developer at:

arkano22+obi@gmail.com

Obi Cloth Component

When adding a Obi Cloth component to a GameObject, it makes a copy of its mesh and creates a physical representation of it: a particle for each of its vertices, with several constraints holding them together.

General parameters

There are a few general parameters for the cloth simulation:

- Area density: This is mass per surface unit, in Kg/square unit. High values will make your cloth behave like very dense fabric upon collision, while low values will yield silk-like collision behaviour.

- World velocity scale: Amount of world-space object velocity applied to the cloth. If set to zero, the cloth will not be affected at all by object movements. If set to 1, cloth will fully react to object movements.

- Damping: This parameter controls particle velocity damping. Set to 0, particles will freely move without losing velocity. Set to 1, particles aren't allowed to move relative to the cloth, except when involved in a collision.

- Squared sleep velocity: Sets the maximum velocity below which particles are allowed to go to sleep. Set this to 0 to deactivate particle sleeping.

- Squared wakeup velocity: Sets the minimum velocity at which asleep particles are allowed to wake up. This should be a few times higher than Squared sleep velocity.

- Minimum wake time: This is the amount of seconds a particle must be under the squared sleep velocity before going to sleep.

- Aerodynamics:
  - Enabled: whether aerodynamic forces should be calculated or not.
  - Wind: Wind force vector in world space.
  - Air density: Air density in kg/m3. This should be left at the default value unless you know what you're doing.
  - Drag coefficient: how much drag forces affect the cloth.
  - Lift coefficient: how much lift forces affect the cloth.

- Recalculate bounds: Forces the cloth object to recalculate its bounds after each update step. If you disable this, the initial -undeformed- mesh bounds will be used troughout the simulation.

- Simulate when invisible: By default, cloth objects aren't simulated at all when no one is watching them. Setting this to true forces the cloth to continue simulation even when not visible by any camera.

Constraints

The Obi Cloth solver is in charge of applying any external forces (like gravity or wind) to the particles, and making sure they also satisfy a bunch of constraints. Constraints are organized in 6 groups, each of them having independent parameters:

- Stretch constraints: These try to hold the particles together, to preserve structural integrity of the cloth.
- Bend constraints: Impose a condition on how much the cloth can bend.
- Self collision constraints: Try to avoid the cloth passing trough itself.
- Collision constraints: Try to keep the cloth from passing trough colliders.
- Pin constraints: Try to keep a cloth particle attached to a rigidbody.
- Shape matching constraints: Try to make the cloth keep a certain shape.

Each of these groups have different parameters, but there are a few that they all have in common:

- Enabled: Is this constraint group affecting the cloth at all? By default all constraint groups are enabled, altough this is rarely what you want for production-ready cloth. You should disable any group that is not critical for the final look of your simulations.
- Iterations: How many times per physics timestep should these constraints be evaluated? A high iteration count will keep the cloth closer to the ground-truth solution. Keep it low if the constraints aren`t very important for your cloth and you want to get better performance. The default is 3.
- Evaluation order: This tells the solver in which order to evaluate the constraints in this group. There are two modes:
    - In SEQUENTIAL mode, all constraints are evaluated in the order they were created (which is determined by the original mesh vertex ordering) and each constraint “sees” the adjustments made by all previous constraints. This ensures quick convergence, so your constraints will need few iterations to look good. However it is not very stable when several constraints are fighting for control, so there are some use-cases where this mode is not a good choice. For technical users, this is Gauss-seidel.
    - In PARALLEL mode, all constraints are evaluated but their adjustments are not immediately applied to the cloth. Instead, they are stored, averaged, and then the final result is then used to adjust cloth vertices. This yields a very stable simulation even with lots of constraints applied at once, however more iterations are needed for “hard” constraints. Use this mode if you want to trade performance -with high iteration counts- or quality -with low iteration counts- for stability. For technical users, this is Jacobi.

- SOR factor: SOR stands for Sequential Over-Relaxation. When trying to satisfy constraints in parallel mode, one way to improve convergence is to "over-relax" the constraints. That is, if moving a particle 2 units to the left would satisfy the constraint for now, why not moving it 3 units? This is exactly what this factor is used for. 1 is the default value, which does not perform over-relaxation at all. 2 is the maximum, which allows you to perform twice as much relaxation on constraints. High values are used to help speed up convergence for parallel evaluation order. This parameter does nothing in sequential mode.

All in all, these parameters allow you to adjust performance, quality and stability of each constraint group. Quick rule of thumb:

- High performance, high quality, low stability: low iteration count, sequential mode.
- High performance, low quality, high stability: low iteration count, parallel mode.
- Low performance, high quality, low stability: high iteration count, sequential mode. (Not very useful)
- Low performance, high quality, high stability: high iteration count, parallel mode.

Keep in mind that these parameters can be independently adjusted for each group, so you can have stretch constraints perfoming 10 iterations in parallel per frame, while bending constraints are only evaluated twice in sequence, for example. Constraint evaluation is interleaved over each step, so no biasing results from wildly different parameter settings between groups.

Stretch constraints

- Stretching scale: Values higher than 1 will make cloth expand when simulating, values lower than 1 will make it shrink. 1 is the default value.

- Stretching stiffness: Determines how strong the forces holding the cloth object vertices together are. Low values will yield elastic cloth, while high values will make your cloth completely inelastic.

- Compression stiffness: Determines how strong the forces keeping cloth vertices apart are. Low values will yield easily compressible cloth, while high values will not allow adjacent vertices to come close together.

- Tearable: Turn this on to allow your cloth to be torn apart when under too much strain.

- Tear Factor: How much your cloth edges can be stretched from their initial lenght before being torn apart. 1 means the slightest stretching will make them tear (fairly delicate cloth), 2 means they will be able to stretch to the double of their original lenght before tearing, and so on.

Bend constraints

- Bending stiffness: Determines how strong the forces trying to keep the cloth from bending are. Low values will make your cloth easily bendable, high values will make it harder to bend.
- Max bending: Determines how much bending your cloth can undergo before the bending constraints kick in to try and keep it from bending. Use low values for fairly rigid cloth.

Self collision constraints

Self collisions only have one parameter of their own, and that is friction. High values will make the cloth stick to itself, low values will cause it to slide off. In Obi, only vertex-vertex self collisions are performed, and the collision thickness is automatically set depending on your mesh triangle density.

This allows for quite performant self collision checks. Still, self collisions involve heavy computation and should be disabled unless completely necessary.

Collision constraints

- Friction: How much friction will take place between the cloth and other objects. 1 means as much friction as possible, 0 means no friction at all, which will make the cloth behave as if everything was made of ice.
- Virtual particles: Use this when your cloth's triangles are larger than the objects collising with it. This will sprinkle additional particles inbetween your cloth's vertices only during collision detection, that will redistribute collision information back to the cloth's vertices after registering a collision.
- Virtual particle coordinates: Barycentric coordinates of your virtual particles. You can have as many virtual particles per mesh triangle as you wish, though tipically 1-3 are enough. Make sure your barycentric coordinates add up to 1 or your virtual particles might end up outside mesh faces, causing spurious collisions.

Pressure Constraints

These are only available if your cloth mesh is closed, that is, it has no holes in it (think of a balloon of any shape).

If enabled, your cloth will strive to keep a certain air pressure inside of it.

- Pressure: Amount of air pressure inside your cloth. This is not a physically-based value, values higher than 1 will cause over-pressure inside the cloth and it will grow in size. Values lower than 1 will cause negative pressure.

Pin Constraints

You can pin cloth vertices to rigidbodies, kinematic or not. To do so, select the cloth vertices you want to pin, and under "Pin Constraints" in the inspector, click on "Add Pin Constraint". One pin constraint will be created for each vertex you have selected, all pin constraints will be listed in the inspector.

Then you can assign one rigidbody to each constraint. By default the offset value (position in object space to which the particle is pinned) is automatically calculated depending on the initial position of the particle relative to the center of the rigidbody, but can also be set manually.

Shape Matching Constraints

- Grab Shape: This button tells the shape matching constraints to get the current cloth configuration as its reference shape.

- Matching stiffness: How much should the constraints strive to keep the reference shape. Low values will cause the cloth to take longer to recover its shape after being deformed, a value of 1 will make it mimic a rigid body.

- Linear deformation: How much the reference shape is allowed to undergo a linear transformation (stretch/squash/shear). Use a high value of this parameter to achieve a softbody-like simulation.

Obi Distance Fields

Obi doesn't support collision with arbitrary triangle meshes trough MeshColliders. Instead, it precomputes collision data in “distance fields”.

Distance fields encode the distance from any point in space to the closest point in the mesh surface. While this seems quite expensive in terms of memory at first, Obi's distance fields are highly compressed, and they allow you to specify the maximum error allowed in compression.

Once a distance field has been created for a mesh, getting the nearest surface point and normal from any position in space is very, very fast, even for large and/or complex meshes. This allows for realtime interaction between your cloth and arbitrary geometry, whithout the use or manual set up of any additional colliders.

What are they good for

Distance fields can be used to make your cloth collide with any mesh. Big, dense meshes (level geometry, for instance) are well managed by distance fields. They allow meshes to be rotated, translated and uniformly scaled in realtime at no cost.

Distance fields won´t work if the mesh they represent intersects itself, is non-manifold, or has any kind of malformed geometry. However holes in the mesh are supported.

Procedural meshes are supported, at the added cost of having to recompute the distance field at runtime each time the mesh shape changes.

Skinned meshes are not supported. For skinned meshes, it is better to use sphere, box or capsule colliders.

How to use them

Distance fields are assets, not components (they derive from ScriptableObject instead of MonoBehaviour). Being assets, memory can be shared between multiple instances of an object using the same distance field.

You can create a new distance field in the editor. In the inspector, assign a mesh to it, set a maximum allowed error and click “Generate” to create the distance field. The approximate size of the distance field in kilobytes is given, so you can tell how much size in disk/memory is the asset going to consume.

- High quality gradient: By default, field gradient at any point in space is reconstructed from the distance. However this results in discontinuities at certain points in the field, and can affect collision response quality. If you need perfect collision response with this mesh, check this option. Having high quality gradient does come with a price, and that is increased memory usage.

- Max Depth: Maximum depth of the hierarchical structure used to represent the distance field. This is used to stablish an upper bound on the calculation time of the distance field, in case the mesh is very detailed and the DF needs a lot of levels to accurately represent its surface.

- Max Error: Error tolerance allowed when generating the distance field. The generation process will adaptively refine (using deeper/finer levels) those areas of the mesh that otherwise would have higher error. Using both Max Error and Max Depth you can fine tune the amount of detail of your Distance Field.

- Bounds padding: Amount of padding added to the bounds of the object when generating the distance field. This should be equal or higher that the contact offset used by any cloth colliding with this distance field.

Obi World

The old “colliders” list in Collision Constraints has been deprecated in favour of a new component: Obi World.

Obi Worlds allow your cloth objects to be efficiently tested for collisions against vast amounts of colliders, thanks to its hierarchical representation of the scene.

How to use them

Add a Obi World component to any object on your scene. Then either click on the “Manage world” button in its inspector, or open the ObiWorld Manager (Windows->ObiWorld Manager). Then select all the objects you want to collide with the cloth, and the cloth itself and add them to the world by clicking on ”Add Selected Objects” in the manager. All cloth objects and colliders within the same Obi World will collide with each other.

You can add and remove objects programatically too. To add an object to a world:

```
ObiActor a = object.AddComponent<ObiActor>();
a.Initialize(obiWorld);
```

To remove the object from the world, just delete its ObiActor component. If there's more than one component its means the object belongs to mulyiple worlds, and you should choose the component with the correct “world” property.

You can create a small script that adds the object in question to an ObiWorld in Awake(), that way you can make sure every object you instantiate is added to an ObiWorld.

You can have multiple ObiWorlds, too. This way you can control which objects collide with which.