Tearing Cloth in Blender

Alexa Orosz, Steven Utley

Advanced Graphics Spring 2019

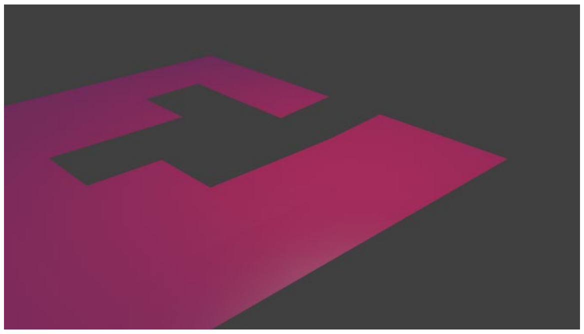


Figure 1: torn cloth

Abstract

The creation of tearable cloth in Blender has been requested for a few years by users of the software. Working with the existing tools and cloth structure in Blender proved difficult to manipulate, and in order to manipulate the spring-mass system correctly, we have implemented a visualization system to see the forces on the springs. Tearing cloth exists in graphics already, but in Blender there are only a few expensive plugins, some of which do not even work with the cloth soft body system and fake the tearing process.

Here we describe our developments required in order to implement tearing cloth through the Blender API using Python. First, we created a system to visualize the forces on the springs. Secondly, we developed a way to determine when the cloth was stretched beyond a force threshold that we determined. Thirdly, we used internal tools to slice the cloth one the threshold was reached.

1 Introduction

1.1 Motivation

Blender is a free and open source software for the entire 3D pipeline and some newly implemented 2D features. Within this framework, there are also opportunities for extension in almost every conceivable way using Python and Blender Objects. Instead of just creating a tearable cloth that runs on our code base, we wanted to create something useful for everyone. After going on several Blender forums to learn more about the cloth's underlying structure, we saw that there were several requests for Blender to implement a way to tear the cloth as there was no existing way to do this with soft bodies. There are existing scripts and addons available, but the most realistic are for animation effects or are rigid body objects made to look like cloth, and do not actually use the cloth system. This is not helpful is you want to use the cloth properties.

Blender already has the underlying structure of a well-developed spring-mass system. In addition to that, there are several tools, such as bisect and edge split, to cut open meshes. The tools to create a torn mesh are already there, and one can easily make an already torn cloth. The challenge comes from making the cloth tear automatically when sufficient force is enacted upon it, either from a solid object, or the edges of the cloth being torn apart for example.

1.2 Blender Cloth

Blender, as already stated, has an existing spring-mass system, with four types of springs: tension, compression, shear, and angular. It also has the property of plasticity, which allows it maintain deformations after being subjected to stress, and does not return to its original shape. At first, the system used Provot correction, as we did in our second homework, but was eventually phased some of it out and with another unspecified technique. Later, Mezger's Cloth Collision Model was implemented alongside a process called Time Steps made in-house.

A Blender object can be designated as "cloth" in the Physics tab after creation. From there, other objects that will interact with it can be created, lighting and materials or UVs can be applied, particles can be added, and from there the simulation can be run.

It also has multiple settings that already exist that can be adjusted to change the look, mass, stiffness, and amount of bend (cloth wrinkling). As such, many types of cloth can be simulated with minimal effort on the part of the creator, and other addons, like tearing can be applied to all of them without having to change anything. In addition, changes can applied to spring (cloth velocity where higher is smoother), air (to slow falling), and velocity (damping to help cloth reach resting position).

We also used the Pinning setting in our simulation, which holds specified vertices in place. There is the option to pin it as strongly as desired, but we just chose an irremovable pin so the cloth would not come loose during testing.

1.3 Cloth Tearing

Cloth tearing in Graphics involves splitting the cloth mesh as a result of forces on the simulated cloth exceeding the reasonable threshold of strain that the cloth can withstand. In order to be realistic, it attempts to mimic the path the rip will take through the most strained or weakest parts of the cloth, taking into account the integrity of the type of cloth and the amount of strain it can take. For example, a cloth simulation of jeans being torn would take more force and the rip would face more resistance than that of a simulation of cotton.

We originally considered using the base of the spring-mass system, and though we did not go through with that idea, we did gain a large knowledge base that helped us with our final implementation.

In *Four Large Steps in Cloth Simulation*, we learned of a new way to create cloths by using a system of particles and internal forces between them to simulate a cloth structure rather than the vertices and spring-mass. The movement came from scalar potential energy and calculations depend intensely on implicit integration, and can handle collisions well. This would allow for extremely accurate simulations but the time to fully and correctly implement this was beyond our scope.

From there we moved into the basics of cloth in *Tearing Cloth*, which helped us determine several issues to consider, such as where the rupture will occur, the path of the rupture, the structural changes that will happen as a result of the split, and preserving the integrity of the cloth. In this implementation, they use two types of springs including a bend-shear spring connecting adjacent triangles, and an irregular triangular structure. The bend-shear springs are created dynamically during the ripping process after the old ones are destroyed after the springs exceed a certain threshold. The tearing is determined by finding what spring is under stress, and takes past strain into consideration. This was an excellent basis that helped us determine our plan of action when it came to our final implementation, though it was modified for the underlying structure that exists in Blender.

Perhaps optimistically, we also attempted to see how quickly we could create this simulation in order to make this real-time, back when we were considering just making a code base. Fast Simulation of Cloth Tearing is less concerned with physically accurate results, such as we were, and instead focused on speed and computation. While we did have real-time, or close to real time as a goal, we did not need insane blazing speed. Similarly to the other papers we examined, they implemented a mesh of triangles, with a half-edge data structure, and a physics particle for each vertex on top of that. To tear the mesh the target vertex is split into two, the new being a replica of the old, directly in the same spot, but unattached to one side. The "old" vertex is unattached to the "new" vertexes' side. A new particle is then created for this new vertex to help continue the simulation. Naturally, this would have been inconvenient to create for us, as we do not have the time or perhaps the ability to create a complete physics based system.

Lastly, we took a look at extremely realistic tears courtesy of *Simulation of Tearing Cloth with Frayed Edges* which not only dealt with tearing a cloth realistically including the right kind of edges that a woven material would have. They too used dynamic edges and a spring-mass system, fortunately similar to the structure used by us in our past assignment. Surprisingly to us, we ended up following this idea and implementation quite closely, at least for the base layer.

The actual structure of this mesh for the paper has two layers, the base continuous cloth and the top layer of a yarn model more commonly used in engineering. Where there are no tears, the continuous model is used using the familiar structural, shear, and flexion springs and when their strain threshold is exceeded, they tear. In that over-strained area, the continuous layer is blended with the yarn layer, which is made up of mass particle pairs. In over-strained regions, these particle loosen and separate, spreading to other areas, resulting in the desired rips and frays.

Moving forward, we kept two main ideas points in the forefront:

1. Springs can take a certain amount of strain, up to a predetermined threshold depending on the cloth's type and integrity
2. When this level is reached, the springs shall degenerate and the cloth will split at that point.

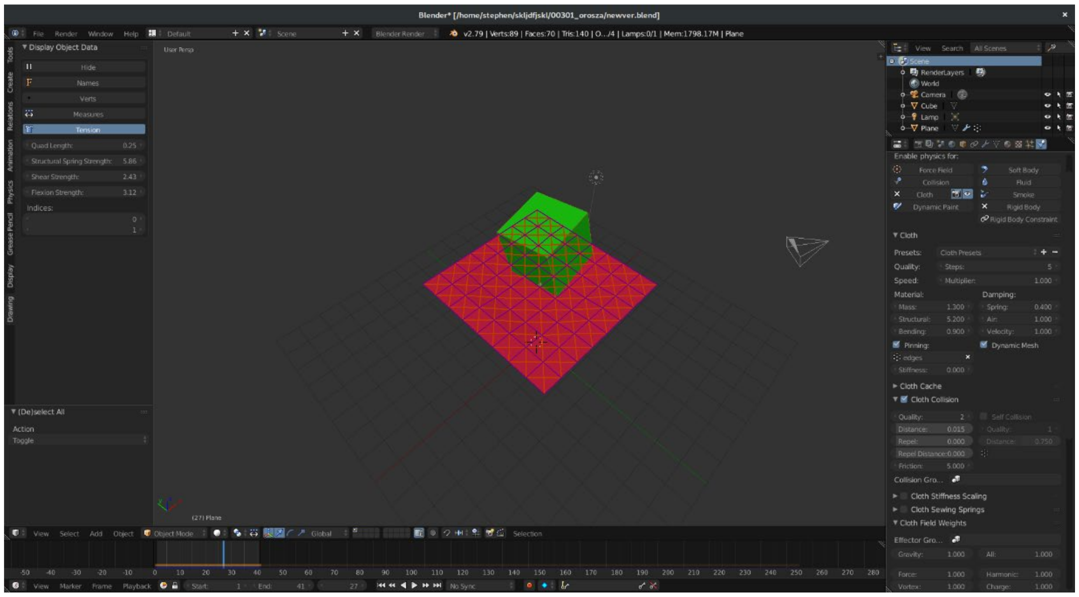
2 Visualizing Spring Forces

So in order to properly do this in Blender, we had to accomplish few things. We first had to find the springs to observe and change their properties in order to perform correct tests. Secondly, we decided to visualize these springs and the strain on them to track the forces enacted during our simulations and tests.

.

2.1 Setting Up the Addon

In order for us to start our addon, we register two classes to handle the user interface we created for the visualization. One is for the panel and the other is for the drawing inside


Blender* [/home/stephen/skljdfjskl/00301_orosza/newver.blend]
File Render Window Help Default Scene Blender Render
v2.79 | Verts:89 | Faces:70 | Tris:140 | O.../4 | Lamps:0/1 | Mem:1798.17M | Plane
Tools Create Relations Animation Physics Grease Pencil Display Drawing
User Persp
(27) Plane
View Search All Scenes
Scene
RenderLayers
World
Camera
Cube
Lamp
Plane
View Select Add Object Object Mode Global Selection
View Marker Frame Playback No Sync

Display Object Data
Hide
Names
Verts
Measures
Tension
Quad Length: 0.25
Structural Spring Strength: 5.86
Shear Strength: 2.43
Flexion Strength: 3.12
Indices:
0
1

(De)select All
Action
Toggle

Enable physics for:
Force Field | Soft Body
Collision | Fluid
Cloth | Smoke
Dynamic Paint | Rigid Body
Rigid Body Constraint

Cloth
Presets: Cloth Presets
Quality: Steps: 5
Speed: Multiplier: 1.000
Material: | Damping:
Mass: 1.300 | Spring: 0.400
Structural: 5.200 | Air: 1.000
Bending: 0.900 | Velocity: 1.000
Pinning: | Dynamic Mesh
edges
Stiffness: 0.000
Cloth Cache
Cloth Collision
Quality: 2 | Self Collision
Distance: 0.015 | Quality: 1
Repel: 0.000 | Distance: 0.750
Repel Distance: 0.000
Friction: 5.000
Collision Gro...
Cloth Stiffness Scaling
Cloth Sewing Springs
Cloth Field Weights
Effector Gro...
Gravity: 1.000 | All: 1.000
Force: 1.000 | Harmonic: 1.000
Vortex: 1.000 | Charge: 1.000

Start: 1 End: 41 27

Figure 2: *The UI created to visualize and edit springs, with debug info*

the renderer. This was previously attached to the animation hook, however this proved ineffective for our needs as this method did not allow for updates at every view change, while the other classes allowed for these updates to occur. Ideally, we would have liked to keep it within the animation hook in order to minimize how often the physics are checked. Unfortunately, the physics remains grouped with the tension visualization as they work on the same data structures and would be difficult to separate. This results in cloth physics being checked on every visual change (including changing the view position, for instance).

2.2 Seeing is Believing

Then we moved on to creating the visualization itself. Currently, we select the chosen object, in our case, the cloth, and fetch two sets from it. One set is full of the original vertices from where the cloth was created or placed at the beginning of the simulation or animation. The other contains the current vertices and is updated at every time step of the animation. The former is retrieved via a helper method and the latter is directly passed to us during the simulation.

After collecting this information, we move into finding the spring connections between these vertex points. Using the starting vertex set, we iterate through every pair of vertices to determine if there is a connection between them. If they have 1 unit between them, they are connected by a structural spring. If the distance is $\sqrt{2}$, then they are connected by a shear spring. Then, we compare this information to our current vertex set, and use them to render our springs and calculate how tense each spring is.

3 Creating Tears

We calculate the tenseness of the spring at every time step, allowing us to test the result against the threshold to determine if the strain the spring is under is too much to handle. If that is the case, we know the edge must tear. Here, we move to cutting the overstrained edges.

To start, the edge container is iterated through, and if we find our strained edge, we

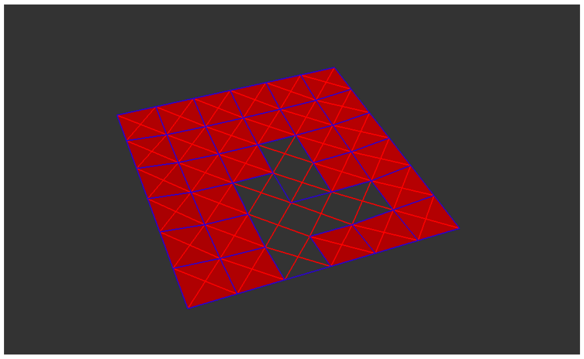


Figure 3: *The cloth model after several successive tears*

move to the next step. We then take the object mesh and convert it into a bmesh so we can edit it using one of the tools available. At first, we considered using the Bisect tool, which allows the user to cut the mesh in two along a custom plane. It either can simply cut the mesh, fill the hole created by the cut, remove the geometry on one side, or cut in a straight plane, but we were only interested in the base tool. In the end, we went with Edge Split, which creates a hole using two or more interior edges that are selected. Those selected edges are duplicated to form the hole border. The strained edges are selected and are split so they tear and create a hole. Finally, the bmesh is loaded back into the stored mesh, overwriting the previous version.

4 Implementation

To implement this method, we created a new Blender scene that was completely empty for each demo, so that we would have a fresh scene every time without any artifacts that we could unknowingly have left behind. We then created a rectangular mesh and made it a cloth object, pinning the exterior edges to hold the cloth in place in the air so we could enact a sufficient force on the inside edges of the cloth.

Next, we set the parameters for our springs and run through several simulations to observe the springs to see if the cloth

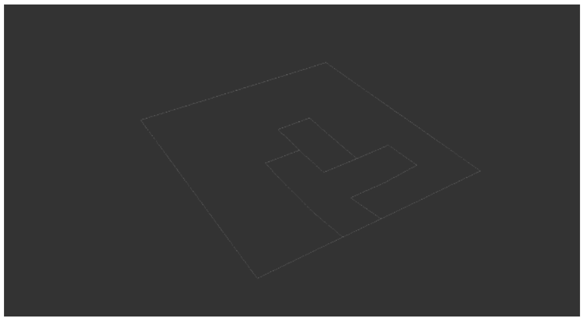


Figure 4: *The wireframe after several successive tears*

behaved correctly. Then, we created another object, a box to interact with the cloth, and set its starting position to a point above the cloth so it would drop down and impact the cloth with, hopefully, enough force to tear the cloth at impact point. We had to set the box to a specific set of parameters so it would interact with the cloth, out first challenge to overcome. The rigid body setting was changed to active for the box and passive with soft body for the cloth mesh.

From there we ran the simulation, testing for contact between the two and tracking the stress on the springs. Then, we implemented the bisect tool, but were only able to make it work if it was precutting the mesh, which is cheating according to the goal we had set for ourselves. As such, we changed tactics and implemented the mesh split instead.

To test this, we created multiple scenes and dropped boxes onto the cloth, adjusting code and cloth parameters as needed.

After many runs, we discovered that our tension threshold was too low, and our unexpectedly powerful cloth was able to perform well enough to catch the box and stretch out, but not tear. We played around with the settings, eventually adjusting the tension to a level that would allow us to demonstrate the script we had created, showing the tears and the spring tension.

5 Results

The successful simulation we managed to create was a decent approximation of torn cloth. The tension in the springs exceeded our set threshold when we dropped the cube onto the cloth, and the cloth did tear, creating a hole. We were able to run this simulation multiple times with the same cloth to create larger holes and a cloth that was further degraded. We were also able to replicate these results, at first.

We achieved what we had set out to do in our most basic outline; create a cloth that would tear when subjected to sufficient forces. As we had also created a force visualization aspect to our addon, we could monitor the levels of tension as well, allowing for live testing that would give the user visual feedback.

Finally, our program supports different kinds of cloth, different strain thresholds for the springs, and different stiffness settings for the springs. Because of this high level of personalization, it can be used for technical and theatrical simulations and animations.

6 Implementation Issues and Future Work

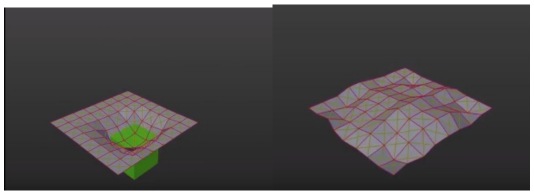


Figure 5: *Examples of the cloth and visualizations working correctly, but not the tear*

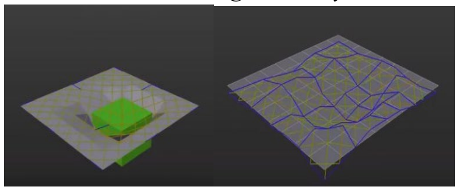


Figure 6: *Examples of the spring visualization glitching*

While implementing these addons, we ran into several issues, the majority of which were caused by our limited experience with Blender scripting before this, and not our understanding of the underlying structures or coding ability.

First, we tried to put the visualization components in the animation hook, which ended up not updating as we wanted because the cloth simulation was under the physics header, and was run in the drawing side of things, not the animation. From time to time, they would also come unattached from the cloth and we had to reset the scene.

When we attempted to first drop the box onto the cloth, it phased straight through and since they were not interacting, the spring tension would not change, and we could not even begin looking at data to decide what the limits would be.

Before we completely implemented one algorithm to tear the cloth, we tested out several options through the user interface, and the bisect tool seemed very promising as it could create a rupture and a rupture path that did not have to follow any existing edges. This was all well and good in terms of looks, but we could not get it working right on the Python side. Of course, we had to switch and chose Edge Split. This also did not perform when we implemented it until we finally discovered that our tension threshold was too low for the impact.

Finally, it ended up breaking when we attempted to open it in a new file. It would have the right edges selected, transform into a bmesh to be cut, and then we would receive the error of no edges selected, despite the fact we put in checks that one edge or more were in fact selected. While older versions still display the desired behavior, a more thorough examination of the root cause remains to be done.

At the moment, the algorithm results in somewhat awkward and blocky holes, although this is in part a function of the small mesh sizes. Modifying the chosen tools (for instance, bisecting over-strained planes along a particular angle instead of only cutting edges) would enable us to create a more realistic split.

We would also love to tear the cloth in more creative way, either by pulling it apart, twisting it, pushing it down on a sharp object, or having it get "caught" on something to show that as long as the tension is over the threshold out cloth will tear in the proper way like our addon advertises. Eventually, if this becomes as bug free as we can make it, this addon can be released into the Blender store for others to use.

7 Work Distribution

Alexa researched the underlying structures of Blender cloth and how to split/cut meshes, set up test cases, and helped implement the actual cloth tearing. Steven implemented the visualization of the springs and their tension, helped implement the actual cloth tearing, and debugged code.

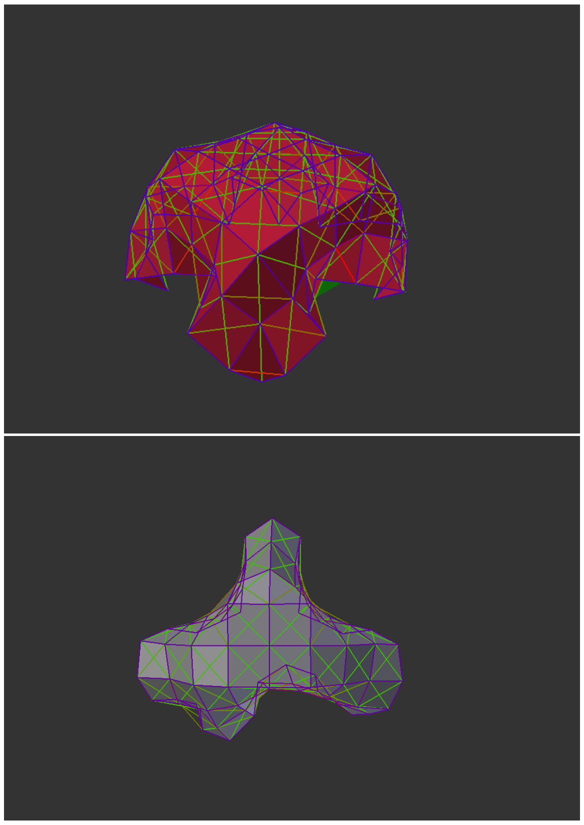


Figure 7: *Here we can see the spring forces visualized from different angles, with the red ones signifying stressed springs*

References

Baraff, David, and Andrew Witkin. *Large Steps In Cloth SImulation*. Siggraph, July 1998, www.cs.cmu.edu/~baraff/papers/sig98.pdf.

"Blender Reference Manual." *Blender 2.79 Reference Manual - Blender Manual*, docs.blender.org/manual/en/latest/index.html.

Metaaphanon, Napaporn, et al. *Simulation of Tearing Cloth with Frayed Edges*. Pacific Graphics, 2009, web.media.mit.edu/~bandy/cloth/PG09cloth.pdf.

Onal, Emre, and Veysi Isler. *Cloth Tearing Simulation*. seer.ufrgs.br/jis/article/viewFile/41187/30099.

Souza, Marco Santos, et al. *Fast Simulation of Cloth Tearing*. SBC Journal on Interactive Systems, 2014, seer.ufrgs.br/jis/article/viewFile/41187/30099.