Surface Tension

Liquid Effects in The Last of Us

Eben Cook
Lead Visual Effects Artist, Naughty Dog Inc.

Surface Tension

Liquid Effects in The Last of Us

Eben Cook
Lead Visual Effects Artist, Naughty Dog Inc.

Game Blood Review


gamesradar.
[F] SEARCH CORPSE
35

Visual Target

Visual Target

Visual Target

- Wanted liquid to live until it hit a surface

Visual Target

- Wanted liquid to live until it hit a surface
- Wanted to model the physics of liquids as much as possible in very little time per frame

Visual Target

- Wanted liquid to live until it hit a surface
- Wanted to model the physics of liquids as much as possible in very little time per frame
- Wanted blood to be exagerated, but believable

Visual Target

- Wanted liquid to live until it hit a surface
- Wanted to model the physics of liquids as much as possible in very little time per frame
- Wanted blood to be exagerated, but believable
- Wanted blood to make it feel like there were consequences to combat

First Attempt

First Attempt

- Tried to use ribbons and sprites

First Attempt

- Tried to use ribbons and sprites
- Resulted in high particle counts

First Attempt

- Tried to use ribbons and sprites
- Resulted in high particle counts
- Needed a lot to happen between frame 0 and frame 1 to get good looking blood shapes to happen with the ribbons

First Attempt

- Tried to use ribbons and sprites
- Resulted in high particle counts
- Needed a lot to happen between frame 0 and frame 1 to get good looking blood shapes to happen with the ribbons
- Could not define initial state

First Attempt

- Tried to use ribbons and sprites
- Resulted in high particle counts
- Needed a lot to happen between frame 0 and frame 1 to get good looking blood shapes to happen with the ribbons
- Could not define initial state
- Ribbons didn't break up into drops and so expanded gaining more mass

This blood was going to be a tough problem.
So we looked at it more closely.

The problem could be broken into
two major challenges.

The problem could be broken into
two major challenges.

Animation

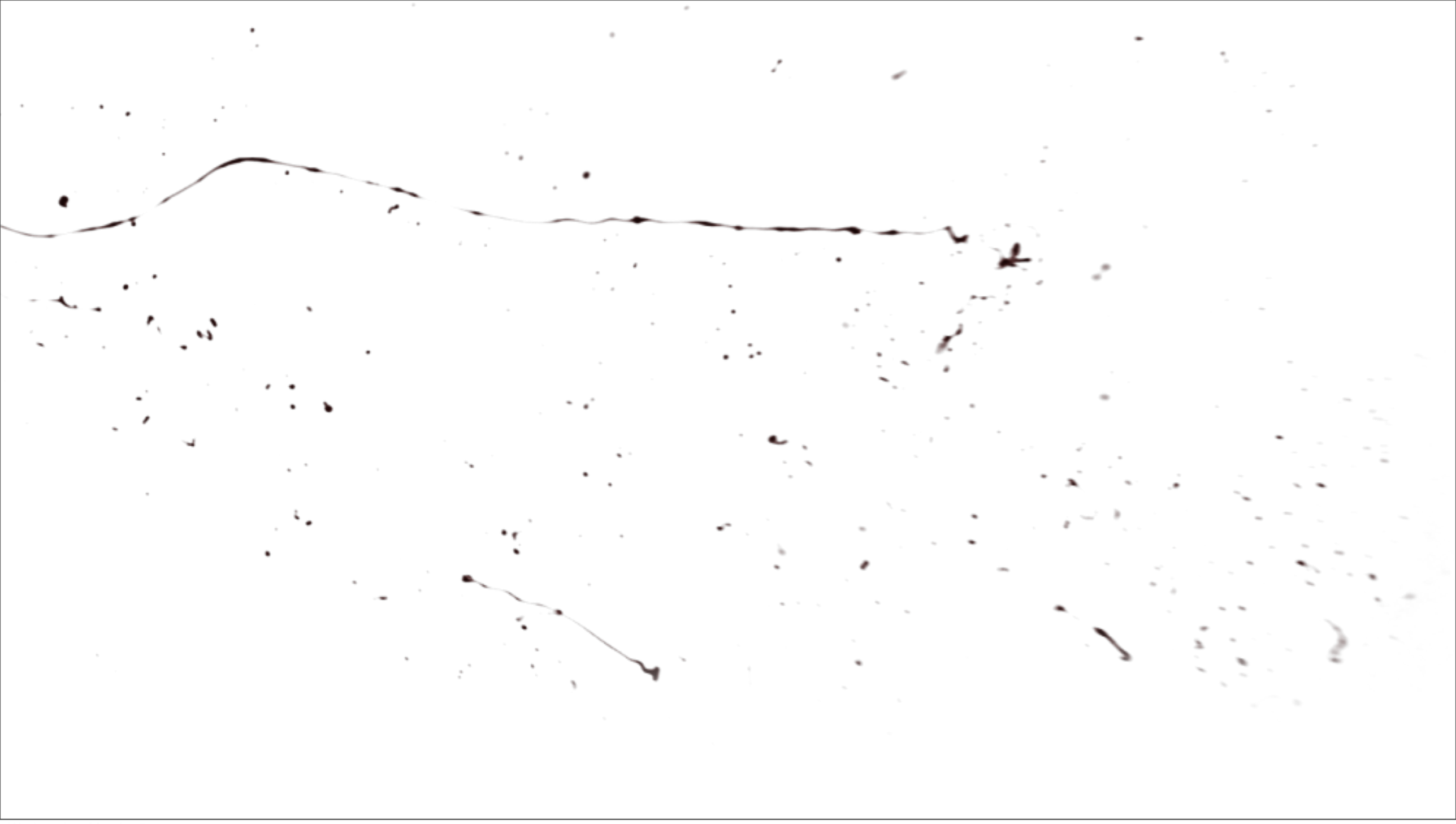
The problem could be broken into
two major challenges.

Animation

Shading

Animation

Video Copilot: Action Essentials 2



Compensate for scaling

Animation

- Overall expansion (scaling)
- Border contraction

Animation

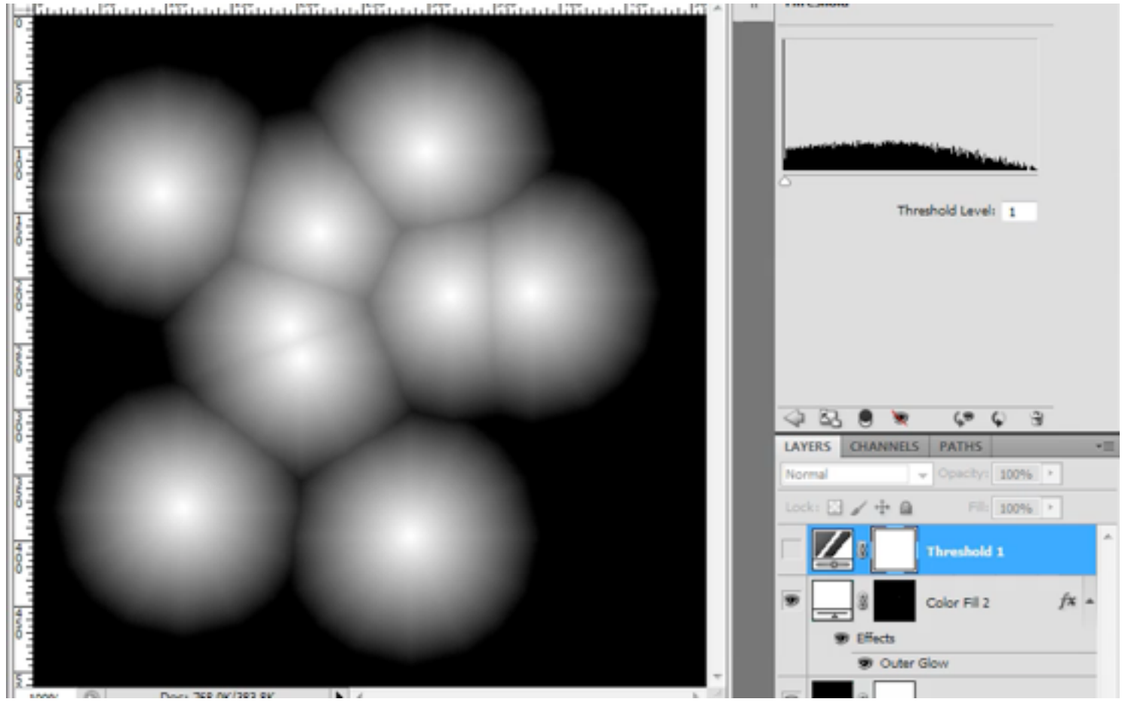
- Overall expansion (scaling) > Sprite scaling
- Border contraction > Shader

Animation: Border Contraction

From Glops to Drops

Animation: Border Contraction

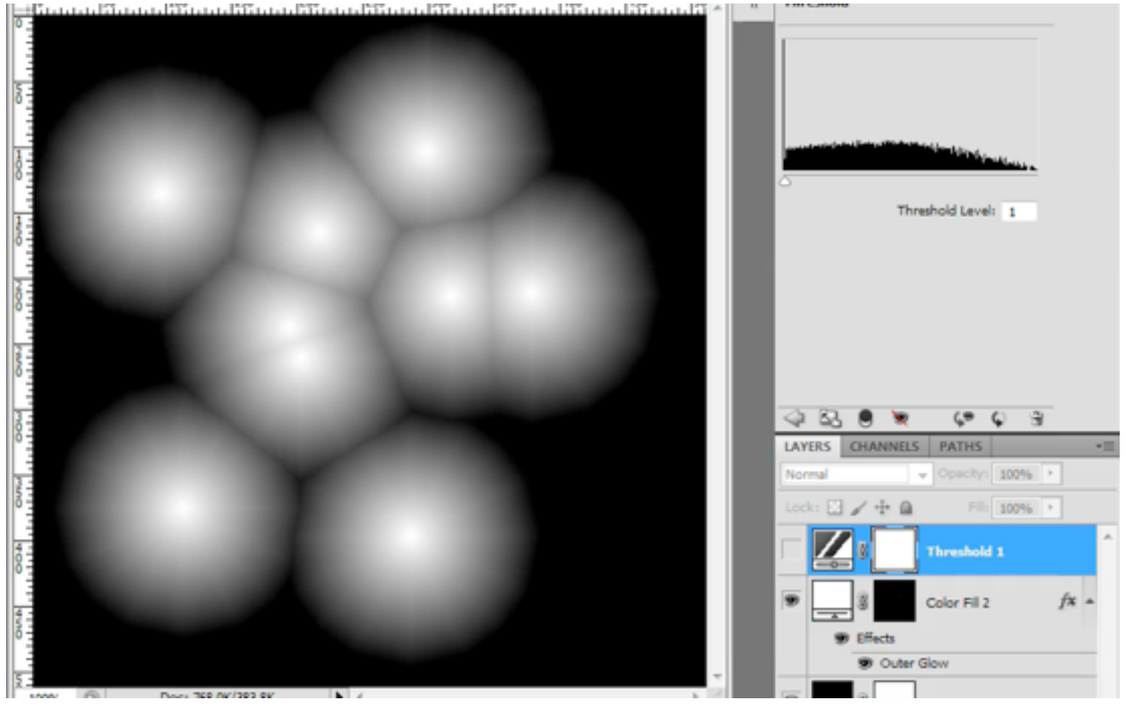
From Glops to Drops



Animation: Border Contraction

From Glops to Drops

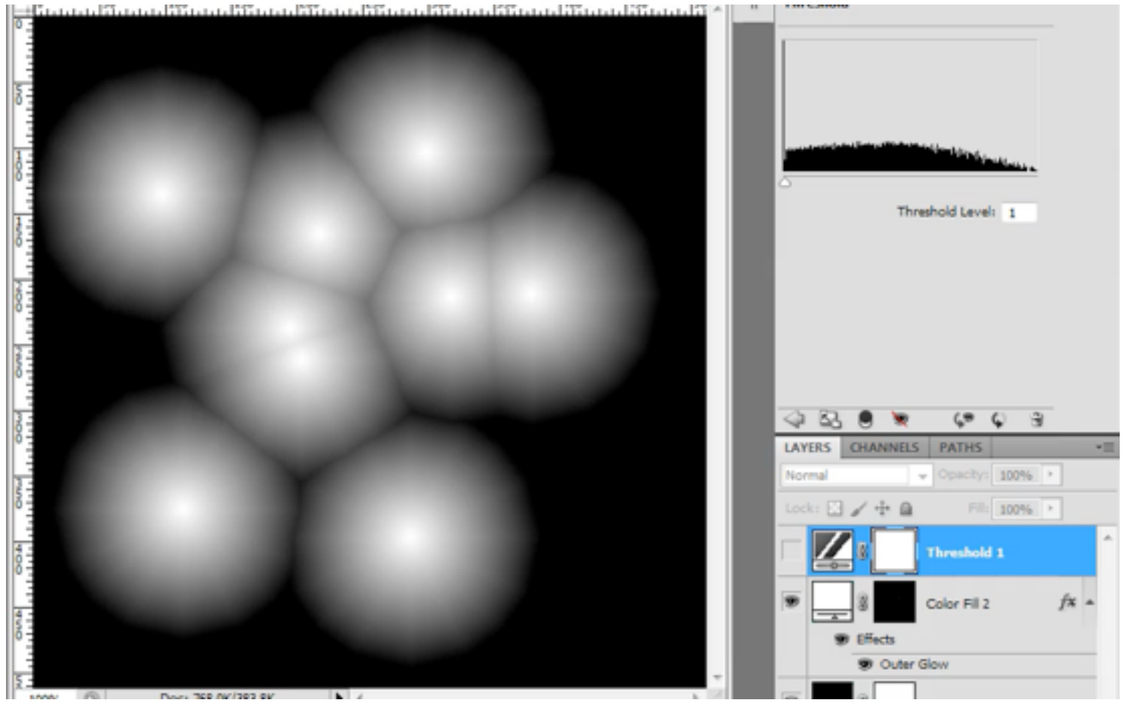
- Threshold animation



Animation: Border Contraction

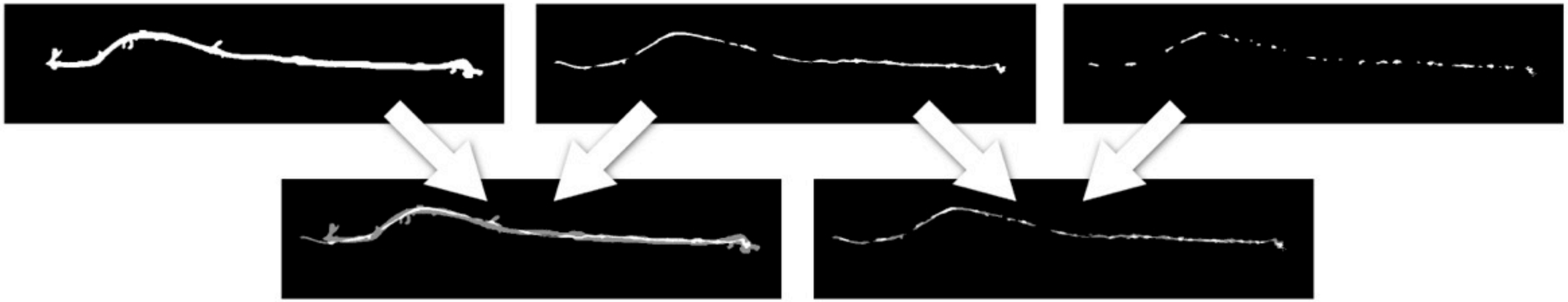
From Glops to Drops

- Threshold animation
- Rule of Threshold Animation:
  Every frame must be contained within the borders of the previous frame.



Animation: Border Contraction

From Glops to Drops

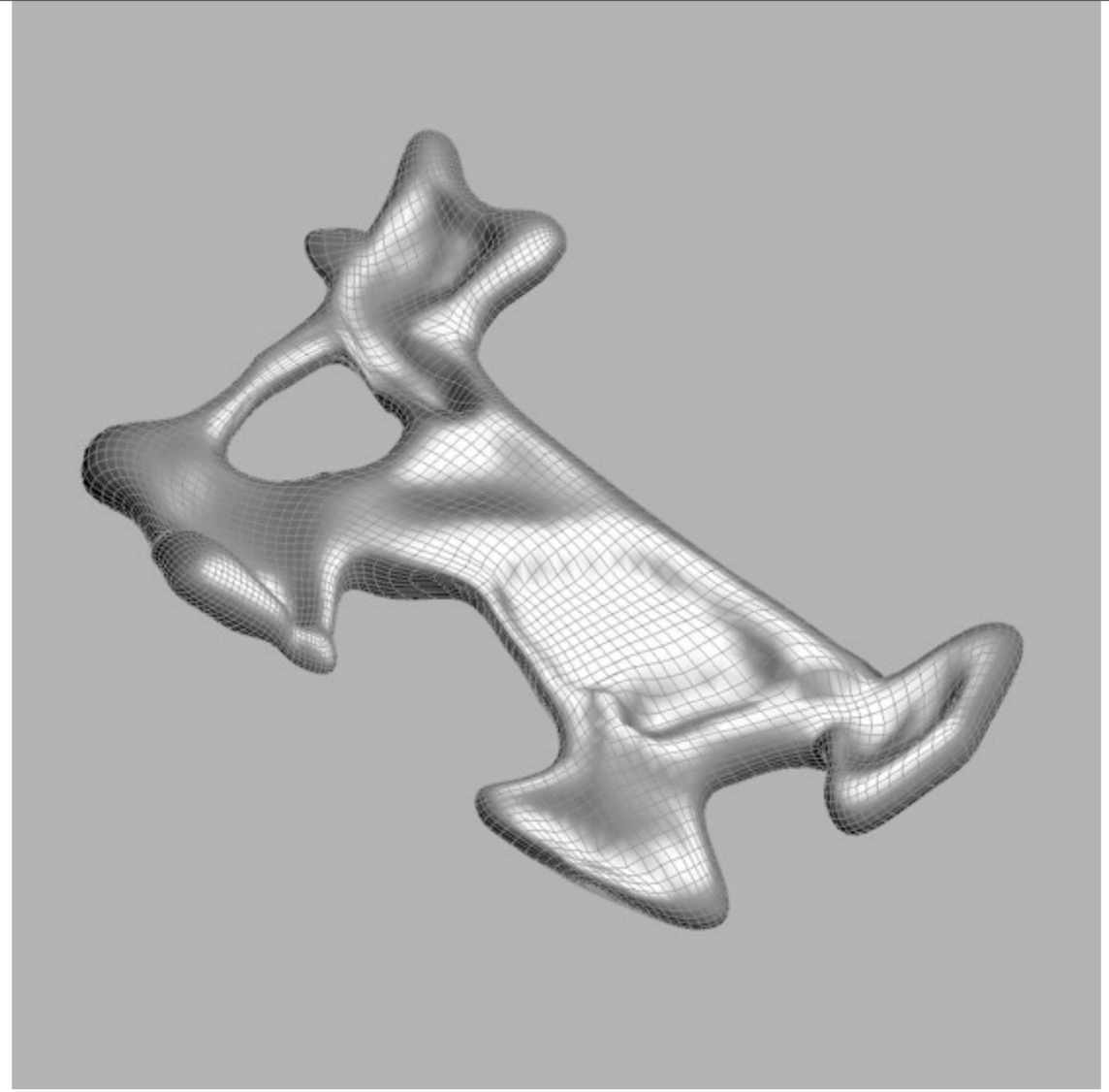


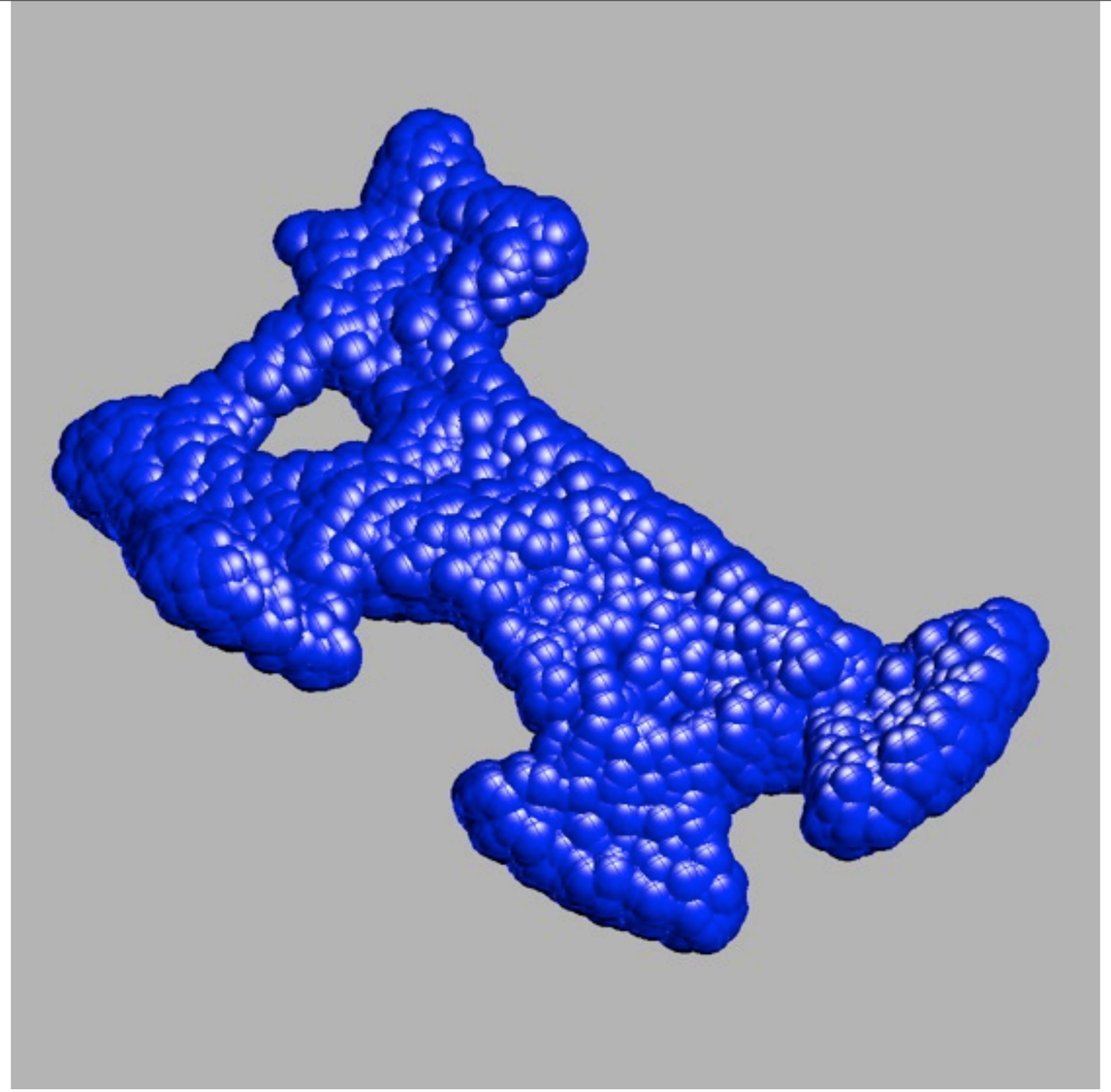
Rule of Threshold Animation:

Every frame must be contained within the borders of the previous frame.

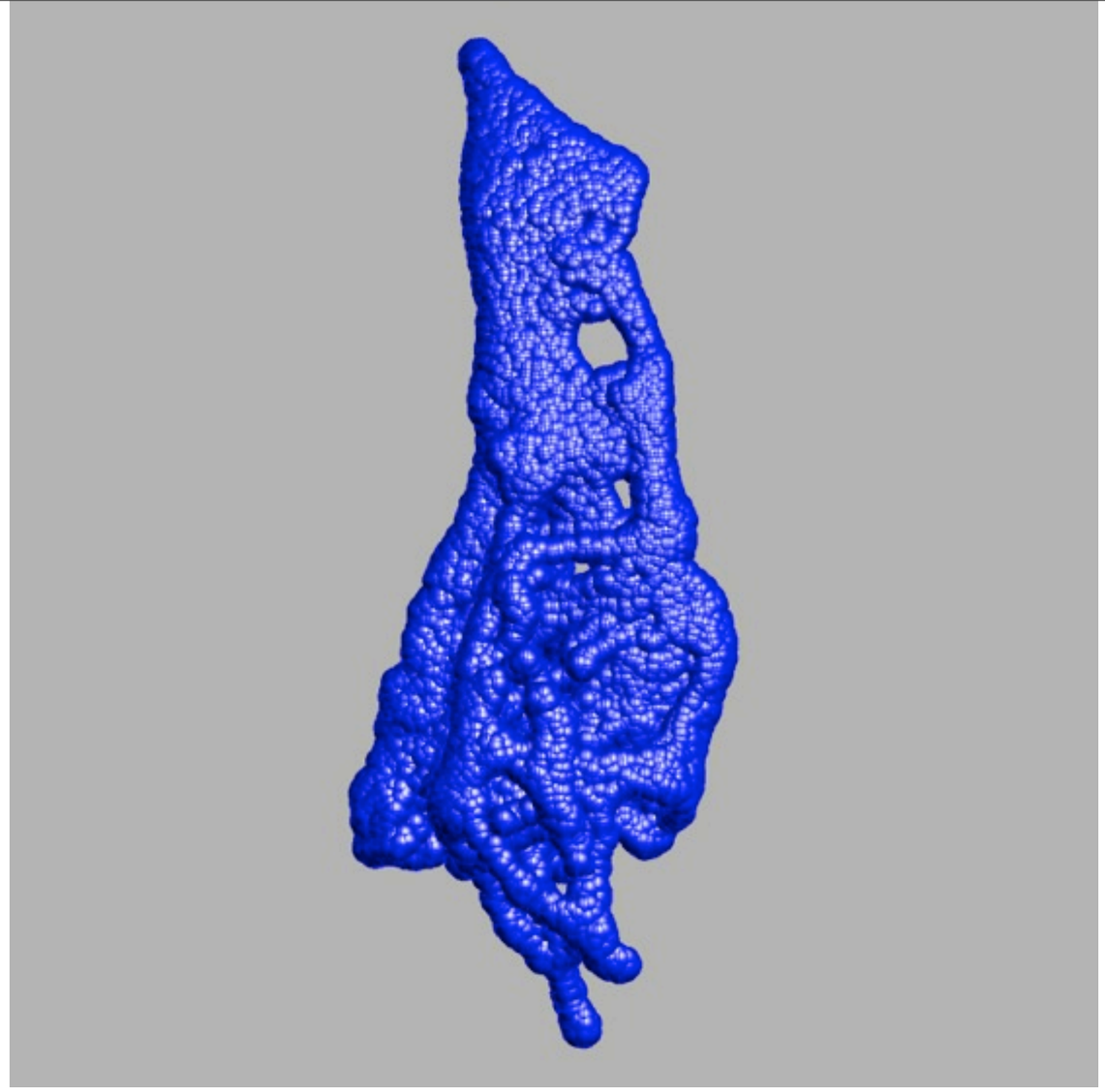
How can we create these threshold maps?

- First attempt: hand paint
  - Result didn't feel natural
  - Labor intensive
- Decided to try fluid particle sim





Tried filling hand sculped shape

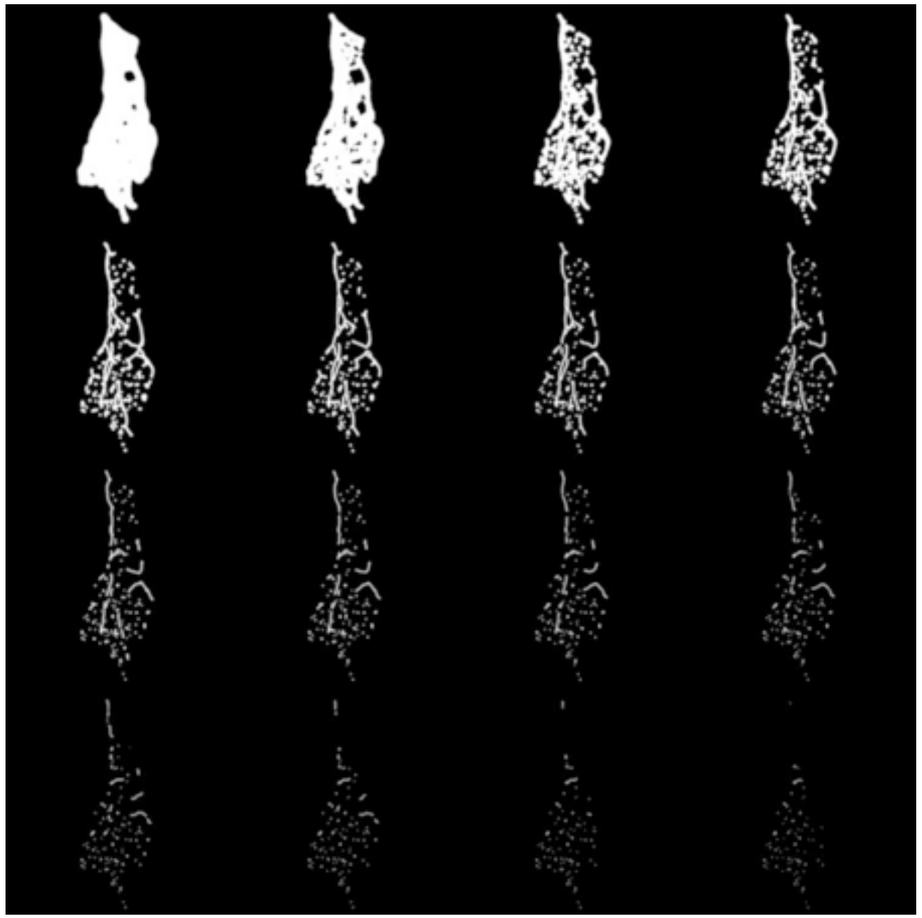


Now we use a shape created in Real Flow

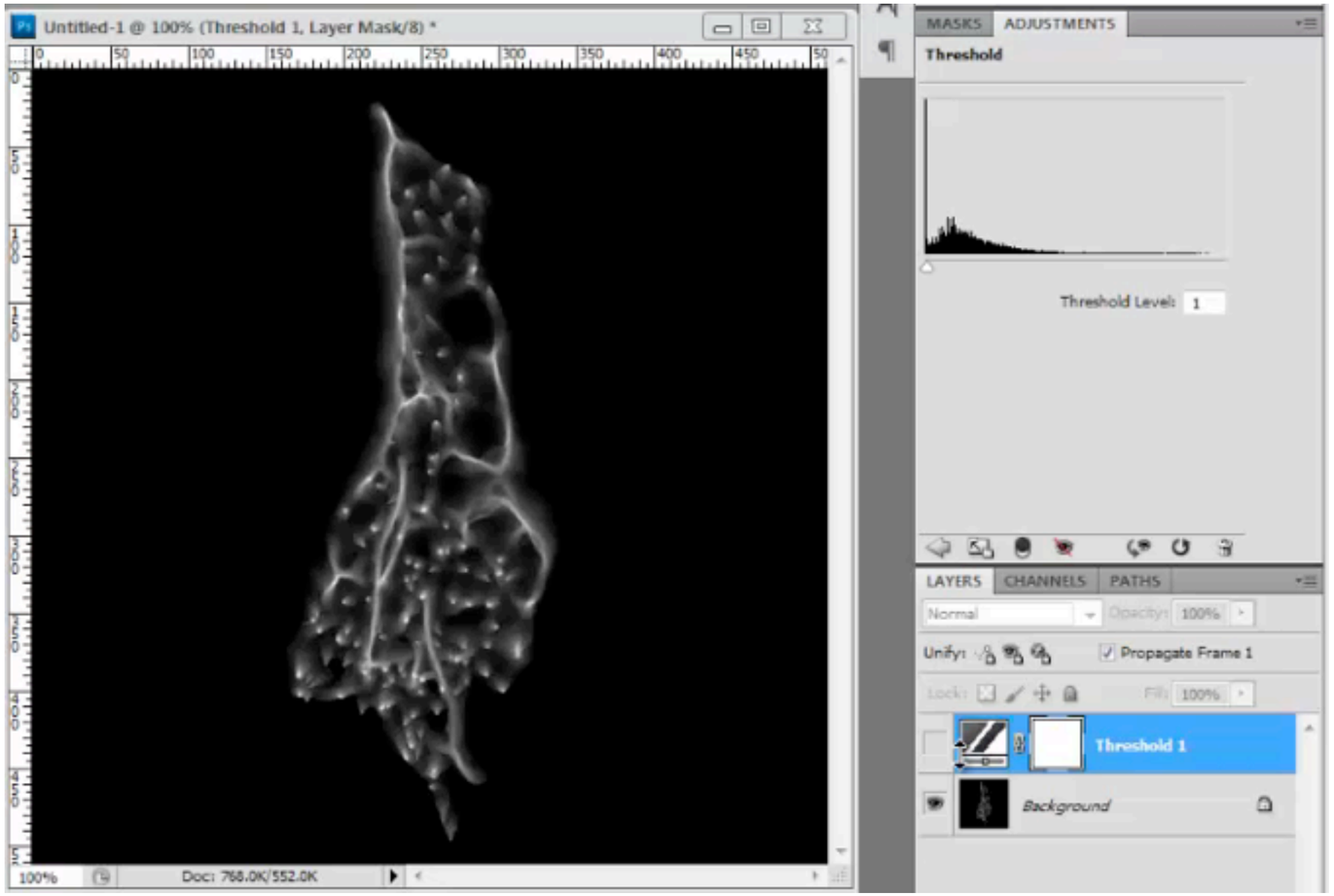
Surface tension makes it condense somewhat

But remember, the shape is expanding...

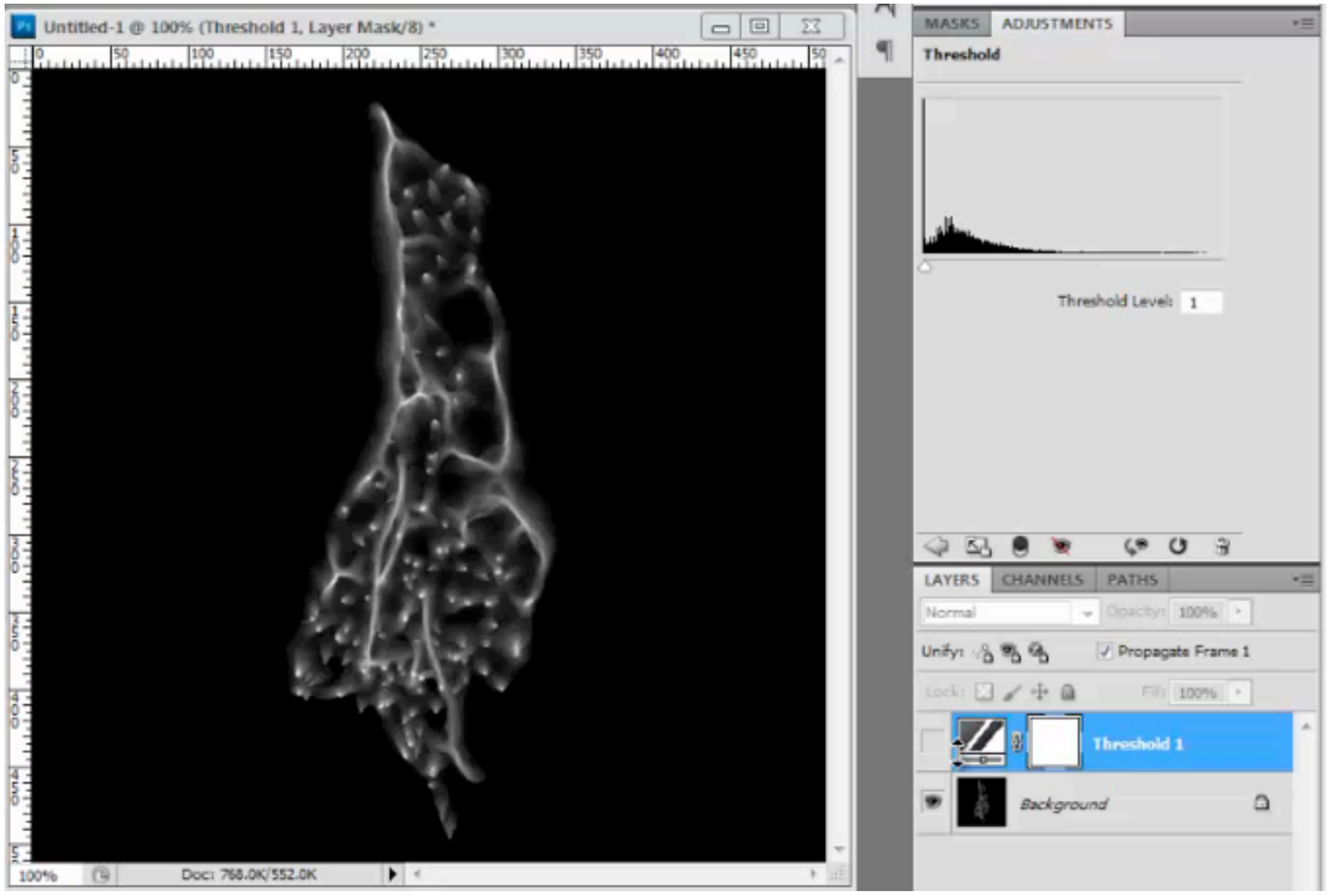
Give all particles outward initial velocity
Zoom the camera out at the same rate

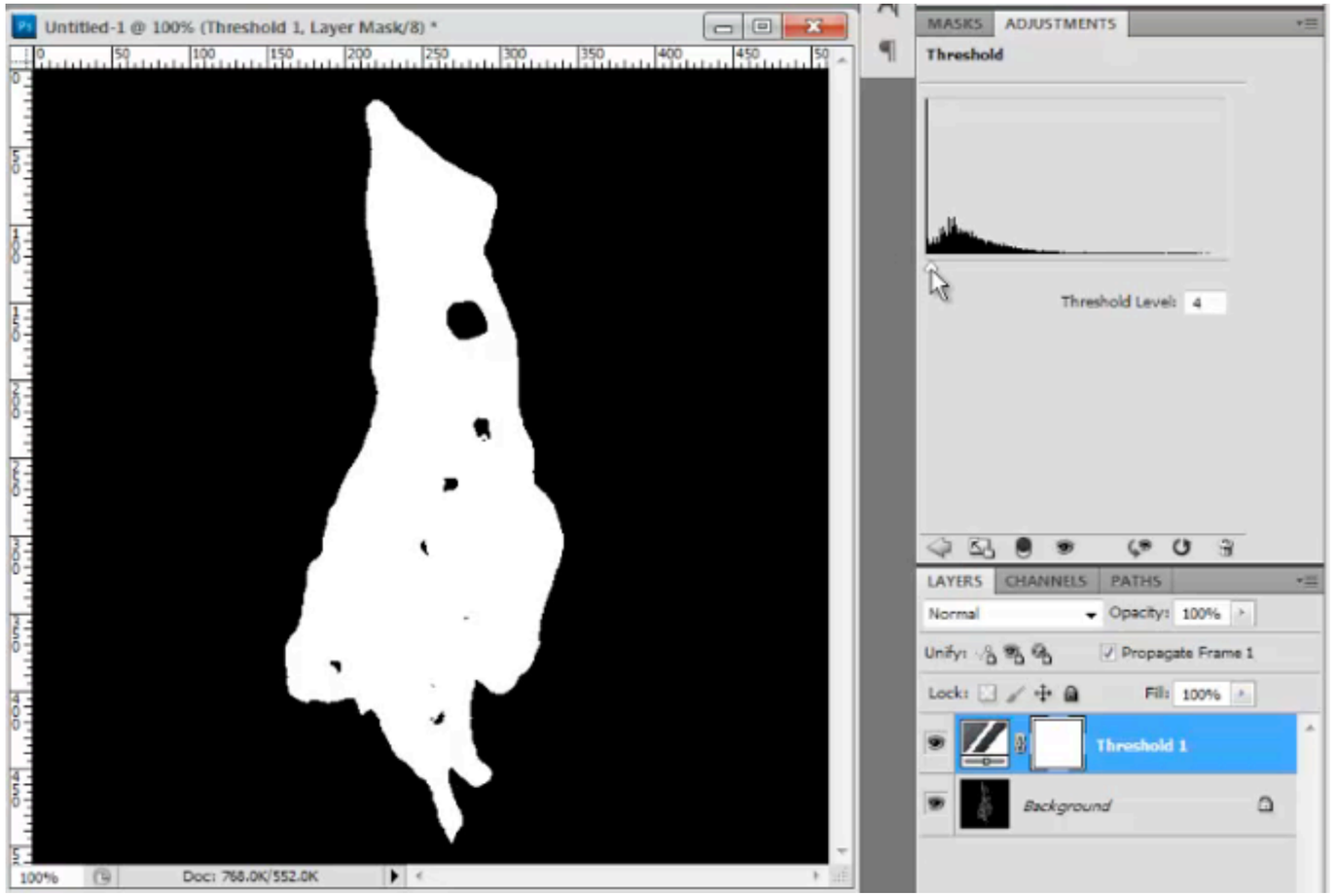


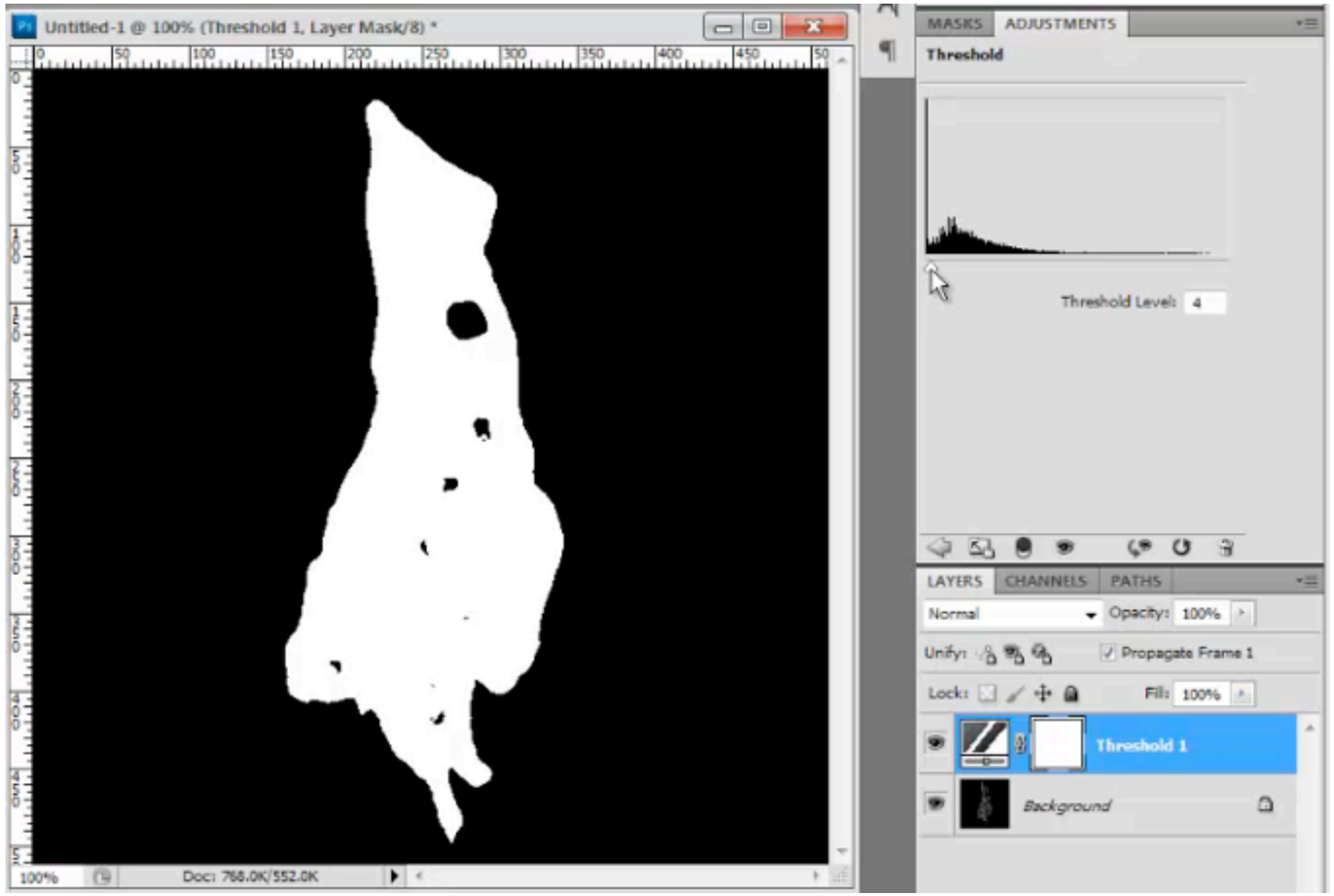
Divide pixel values by number of frames,
then add images all together



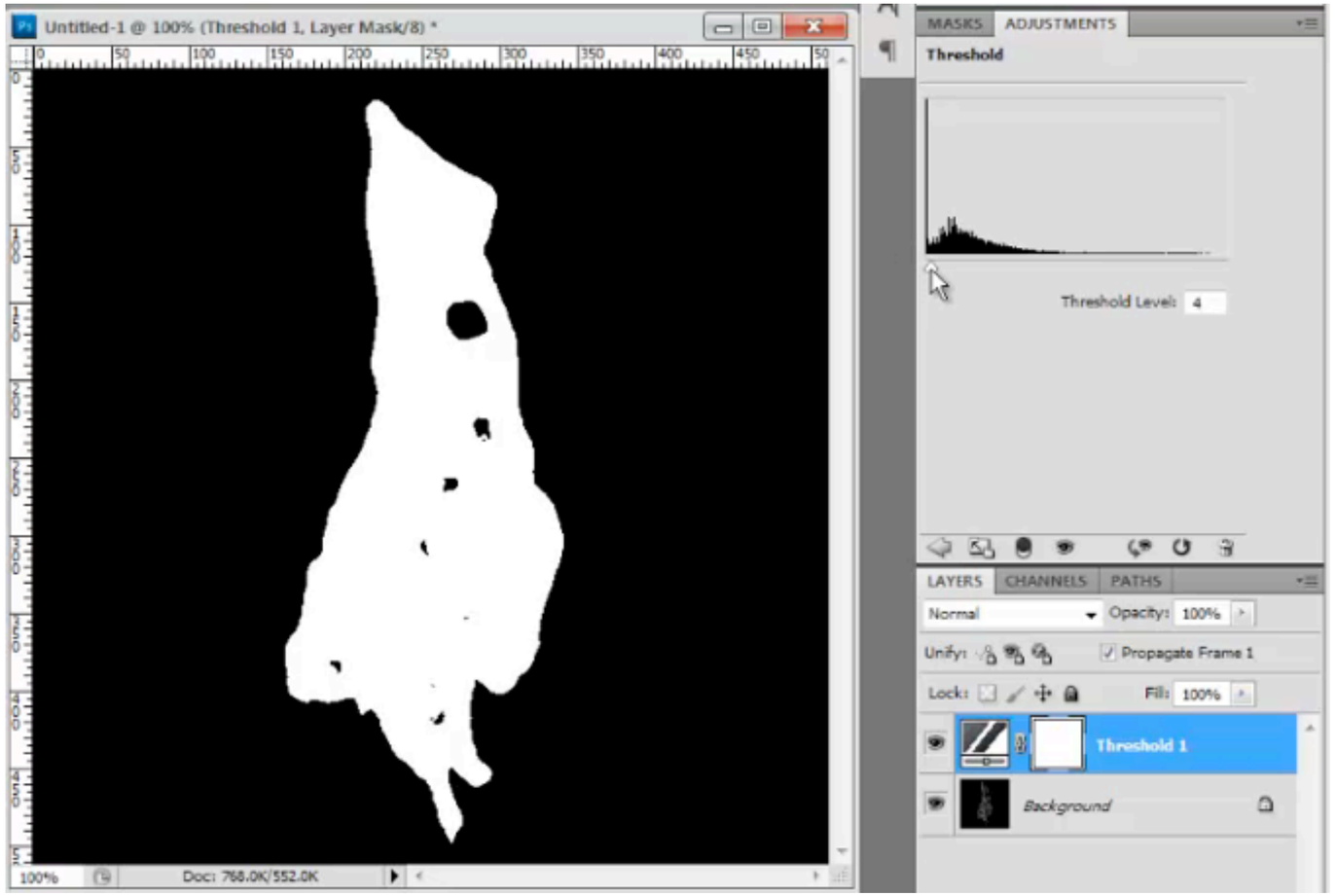
Then you have a threshold texture that breaks into strings and drops...


Untitled-1 @ 100% (Threshold 1, Layer Mask/8) *
100%
Doc: 768.0K/552.0K
MASKS
ADJUSTMENTS
Threshold
Threshold Level: 1
LAYERS
CHANNELS
PATHS
Normal
Opacity: 100%
Unify:
Propagate Frame 1
Lock:
Fill: 100%
Threshold 1
Background


Untitled-1 @ 100% (Threshold 1, Layer Mask/8) *
100%
Doc: 768.0K/552.0K
MASKS
ADJUSTMENTS
Threshold
Threshold Level: 4
LAYERS
CHANNELS
PATHS
Normal
Opacity: 100%
Unify:
Propagate Frame 1
Lock:
Fill: 100%
Threshold 1
Background



Its animation is very close to the rendered simulation...


Untitled-1 @ 100% (Threshold 1, Layer Mask/8) *
100%
Doc: 768.0K/552.0K
MASKS
ADJUSTMENTS
Threshold
Threshold Level: 4
LAYERS
CHANNELS
PATHS
Normal
Opacity: 100%
Unify:
Propagate Frame 1
Lock:
Fill: 100%
Threshold 1
Background

...and resembles an element in the reference

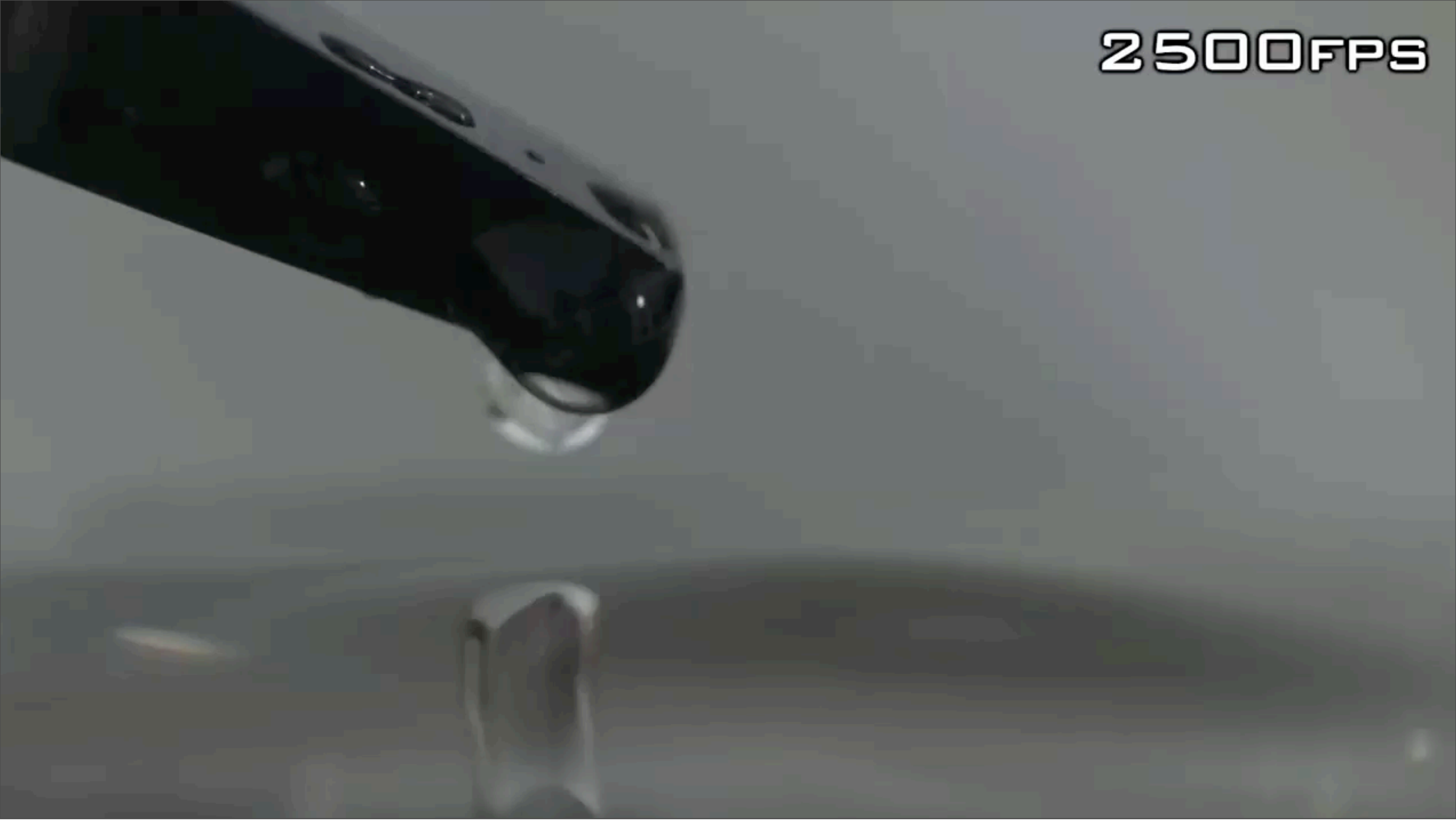
The problem could be broken into
two major challenges.

Animation

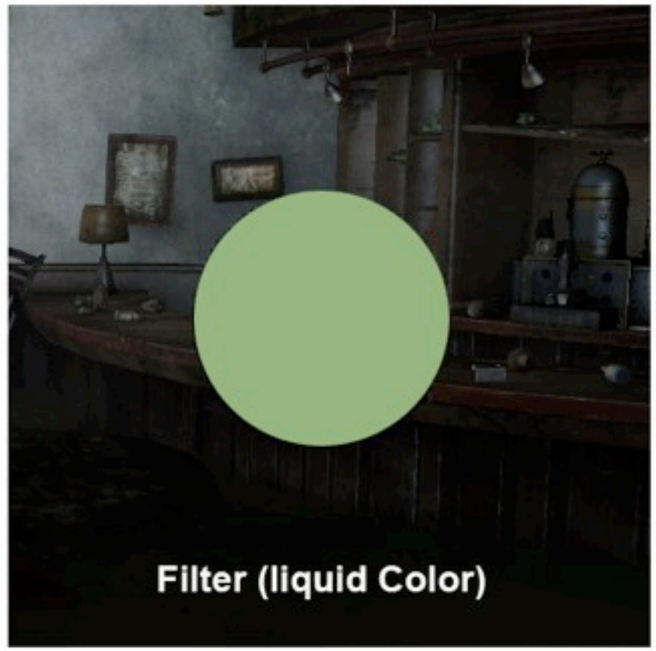
Shading

The problem could be broken into
two major challenges.

Shading


2500FPS

*



+

Passes


FPS: 30.0

Shading

Reflection and Refraction

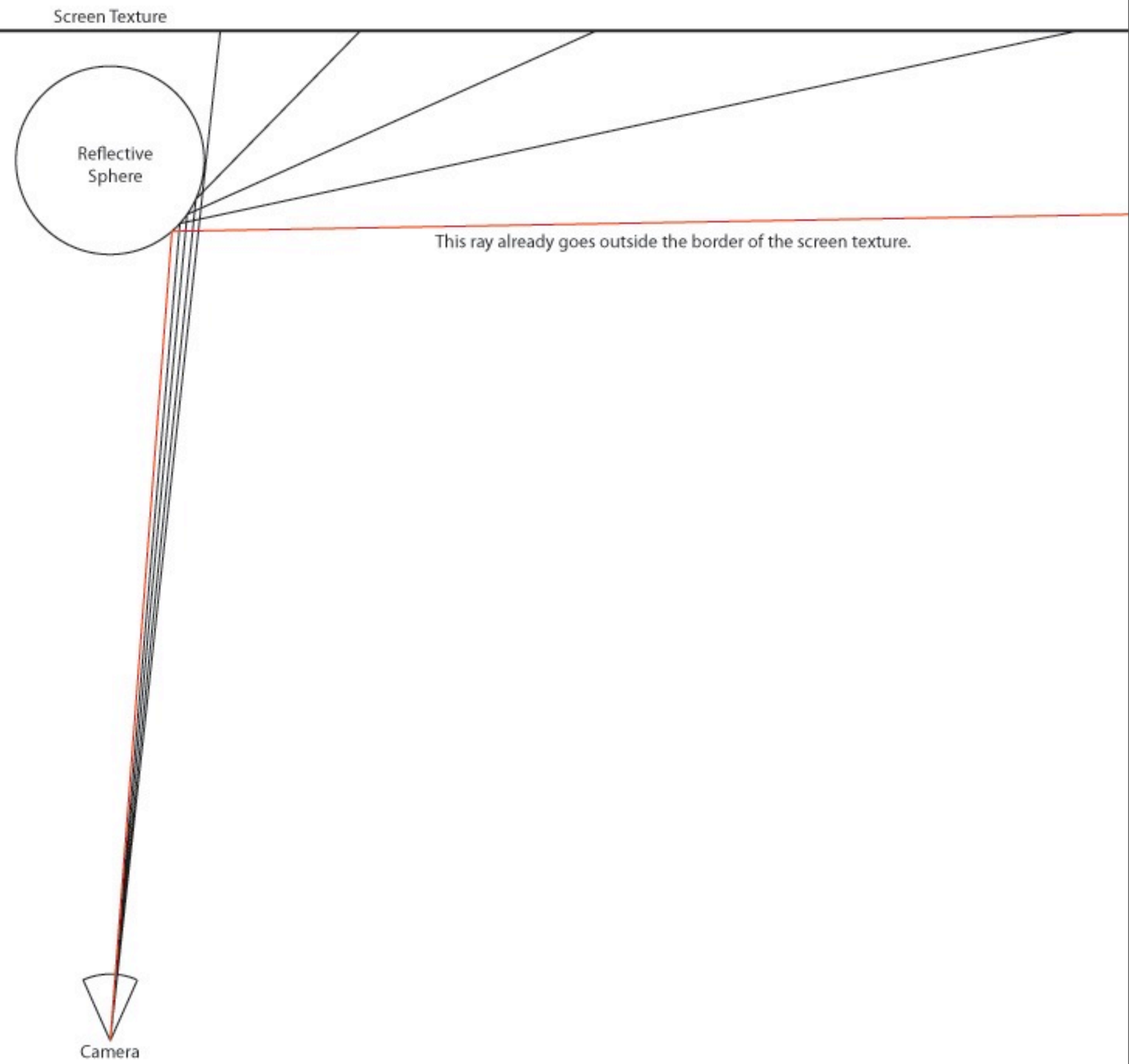
- Sample screen texture with UV offset
- Offset direction is opposite for reflection and refraction

When sampling the Screen Texture for refraction, the edges of the reflective surface (where normals are perpendicular to the viewing angle) do not require offsetting the UV.

The closer the normal gets to the camera vector, the closer to the edge of the screen texture you need to sample.

Eventually, rays will go outside the bounds of the Screen Texture.

To deal with this, we exagerate the fresnel effect and reduce the amount of offsetting.



Normals

- Needed for UV distortion for reflection and refraction.
- Needed for fresnel.
- Needed for specular lighting.

Normals

What should they look like, and where would they come from?

Normals

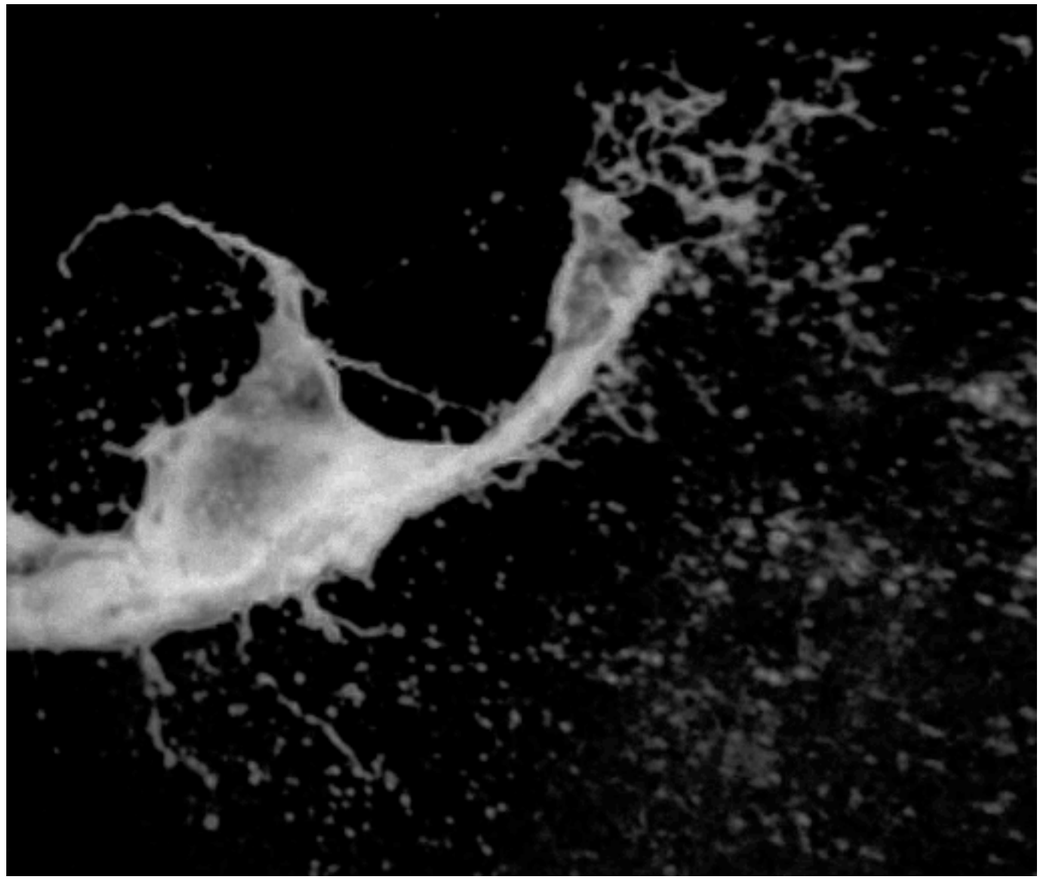
What should they look like, and where would they come from?

- They should change as the shape changes from threshold animation.

Normals

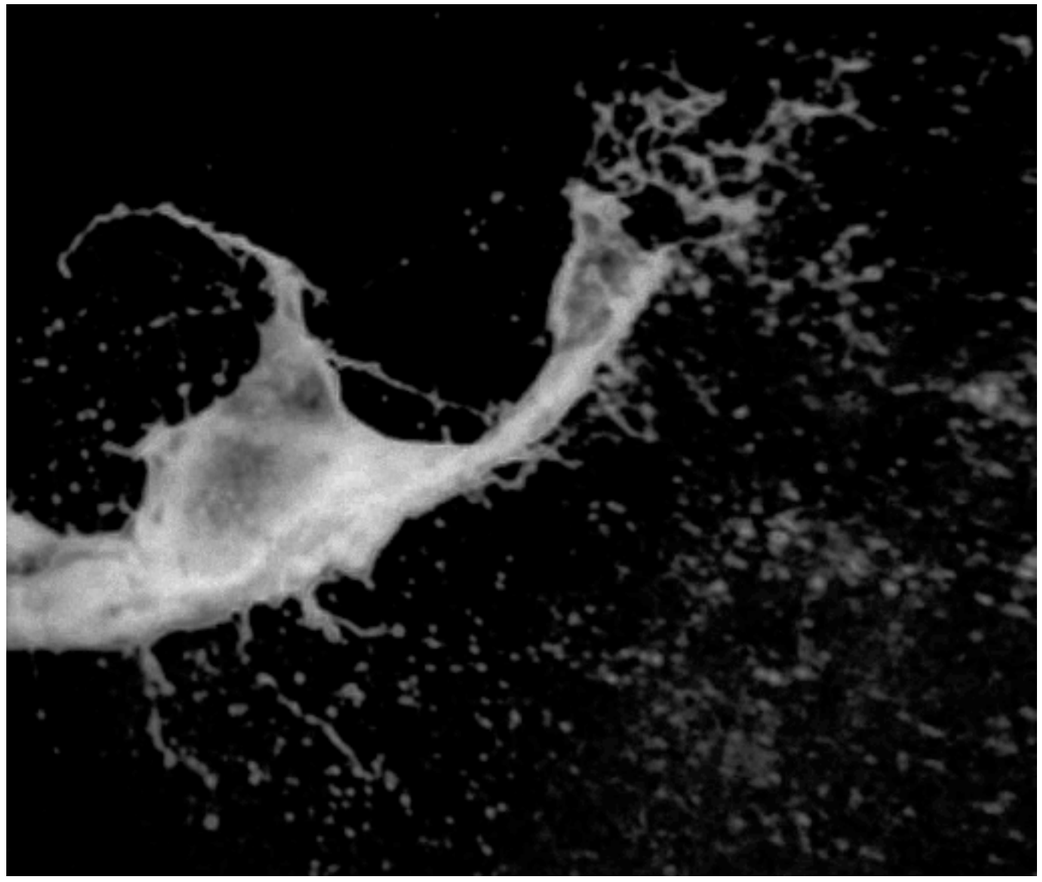
What should they look like, and where would they come from?

- They should change as the shape changes from threshold animation.
- Normals from the simulation would have been dandy, but we wanted to avoid flipbooks. So I had to generate normals procedurally.



Start as strands connected by thin film.

End as a strand or drop.



Start as strands connected by thin film.

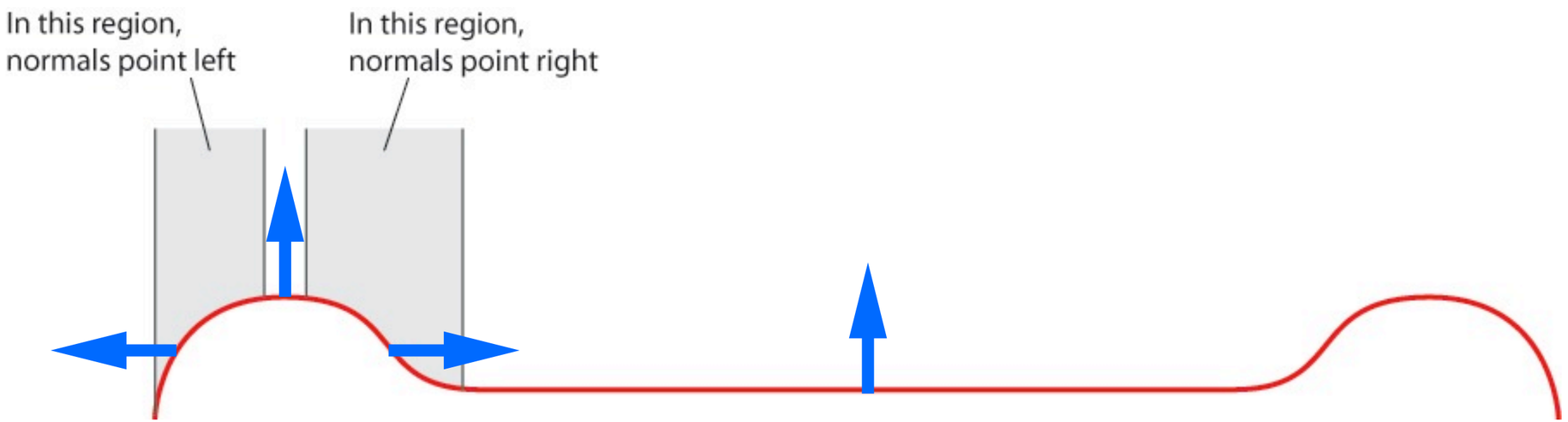
End as a strand or drop.

The normals need to animate along with the border.

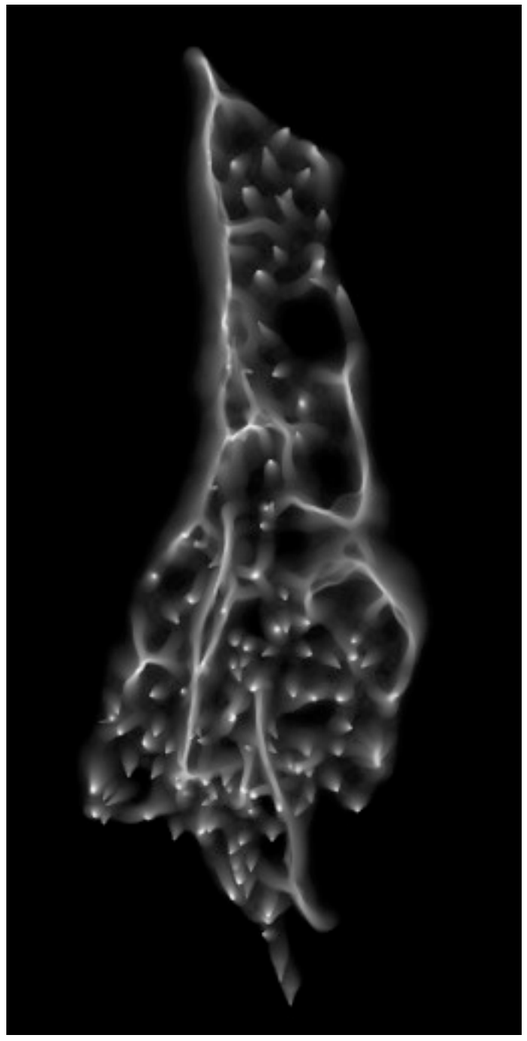
How can we derive these regions and generate normals for them?

How can we derive these regions and generate normals for them?

How can we derive these regions and generate normals for them?



How can we derive these regions and generate normals for them?



Range of 0.02 and 0.05

Range of 0.05 and 0.1

Range of 0.1 and 0.3

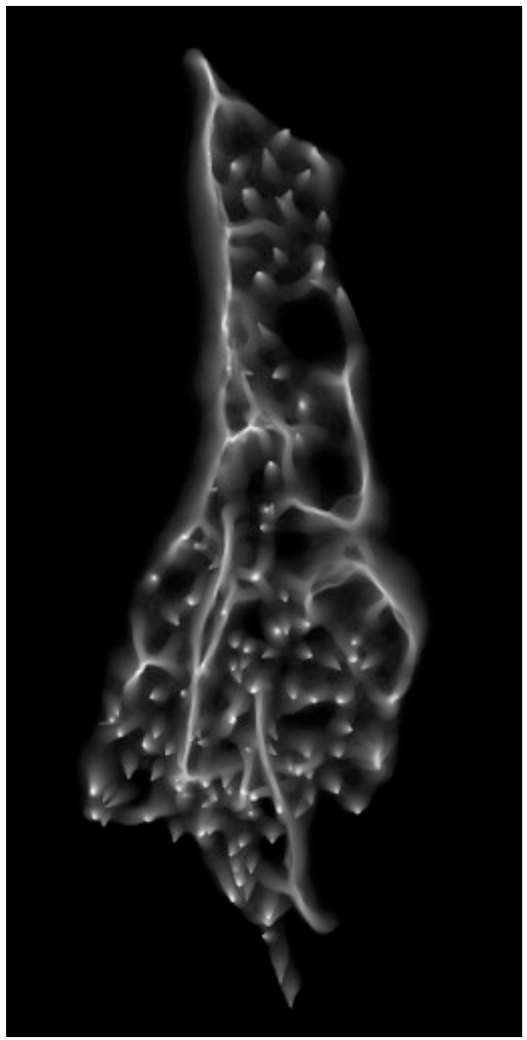
Remap the values in the threshold texture to get a gradients within each of the regions.

Then use the gradients to blend between upward pointing normals and normals that point perpendicular to the borders.

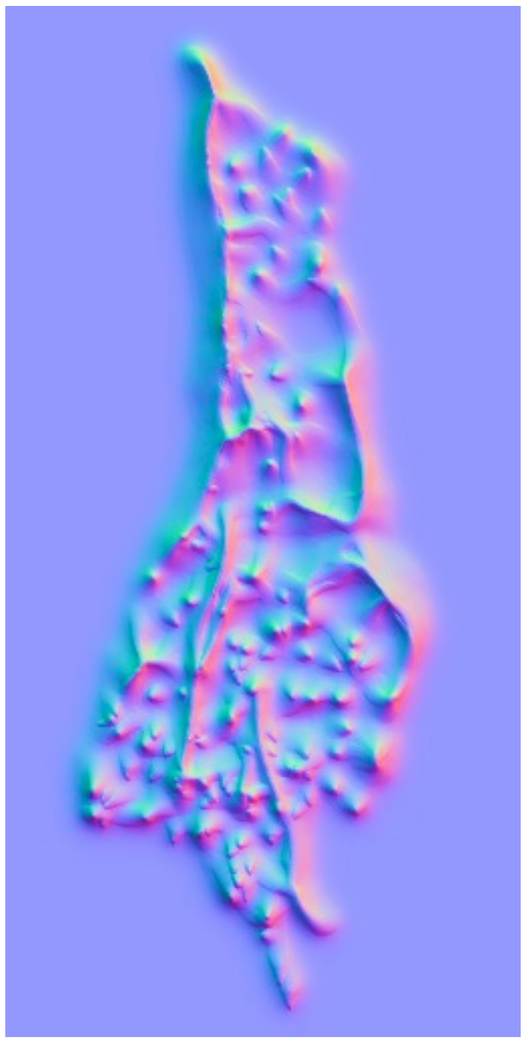
0 = normals pointing perpendicular to border (flip for interior region)
1 = normals pointing up (sprite's normal)

Next question: How do we get normals pointing perpendicular to the borders?

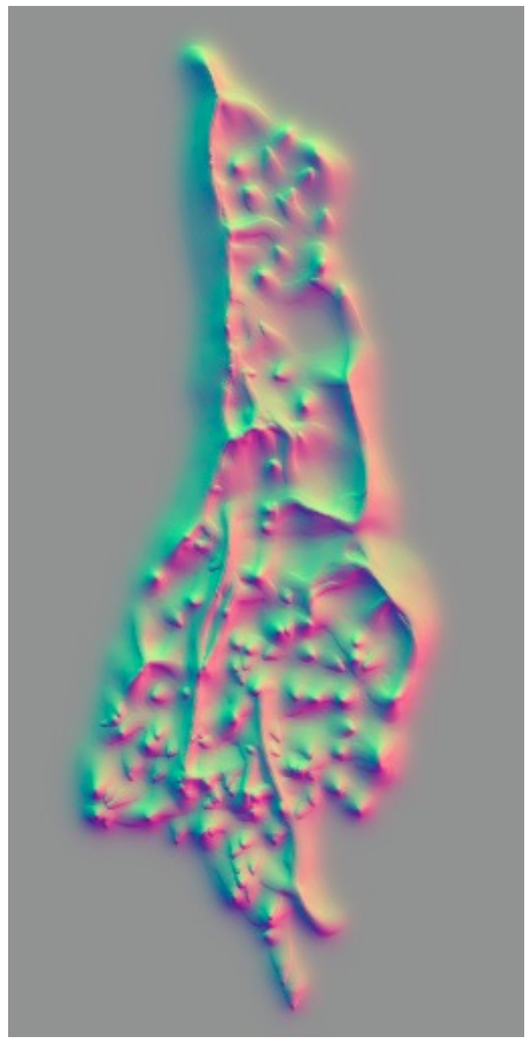
To get normals pointing perpendicular to borders...

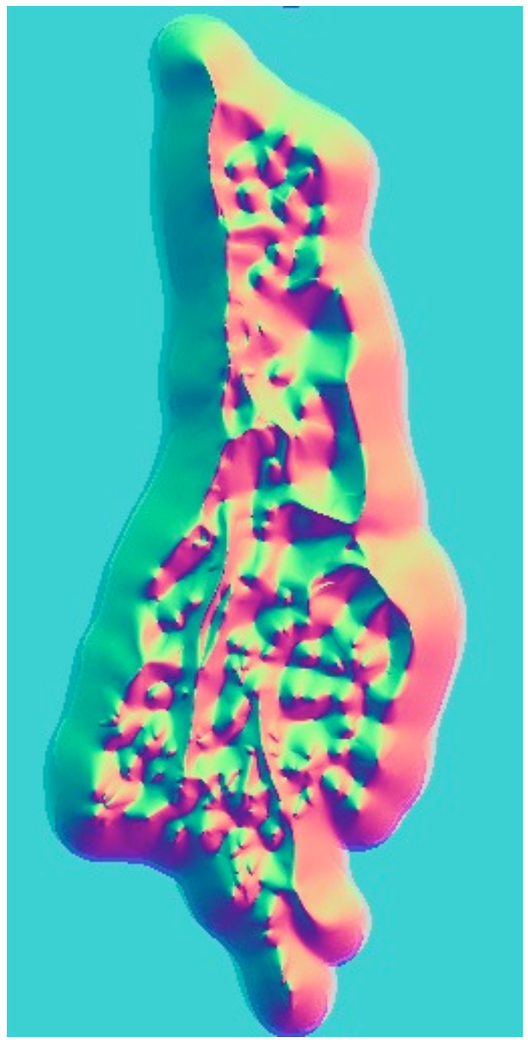


Treat threshold texture as height map and generate normals



Make blue channel 50% gray (eliminate z component)

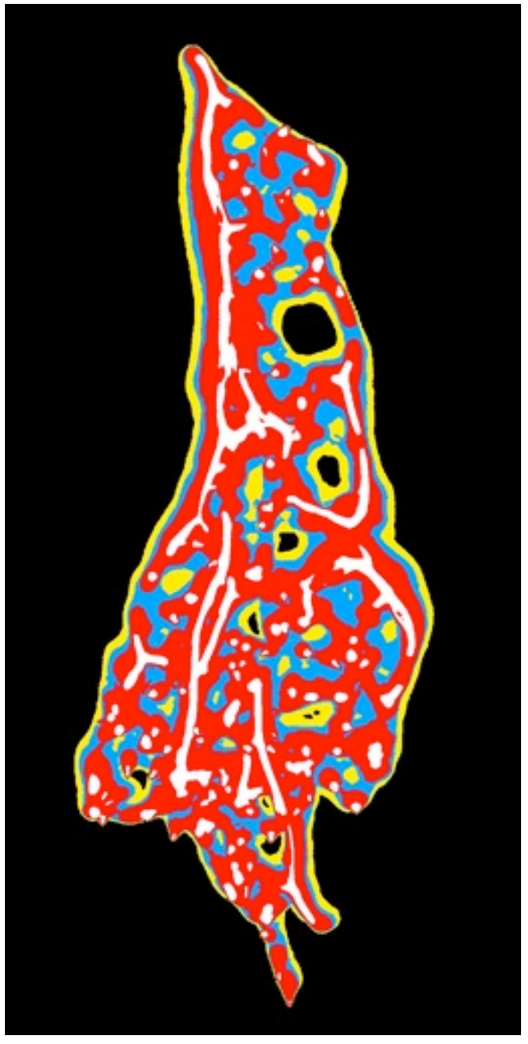




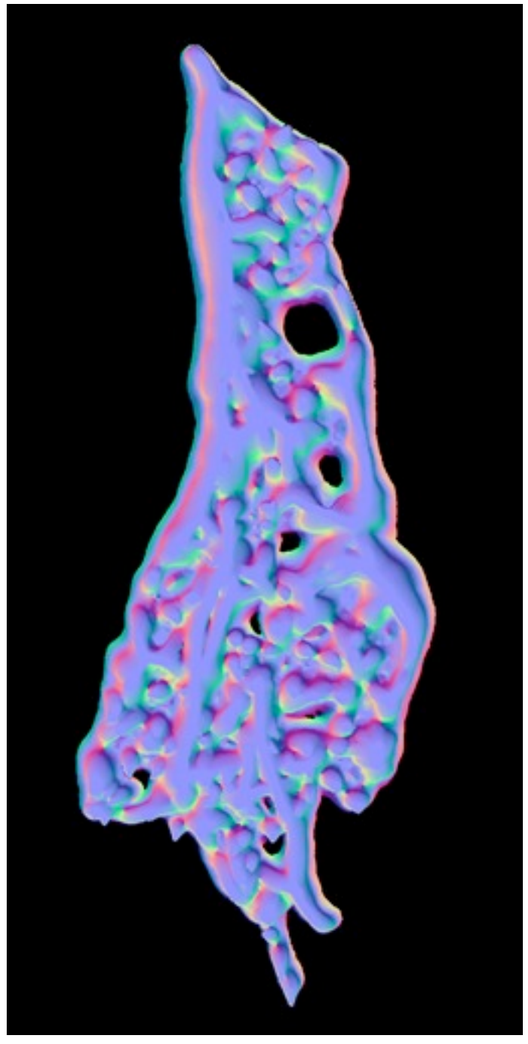
Normalize Vectors

Generating Animating Normals

Take these normals...



Blend them using the gradients in these regions...



To get these normals that animate with the border.

That's the bulk of it.

And in the end,
they are on screen
for about a second.
:-/

Q&A