Lecture 37: Splatting 3

Order, and a Real Scene

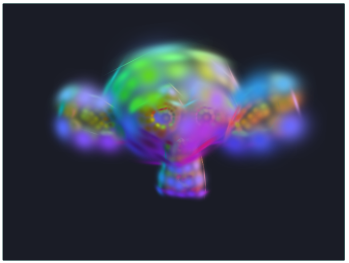
CS 453 — Computer Graphics

Fakhir Shaheen

Forman Christian University

Lecture 36: her face showed through. Today: sort, then a scene

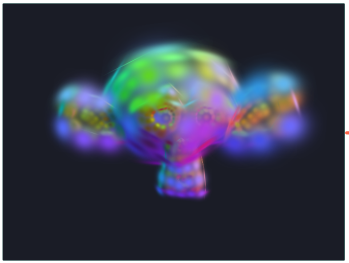
lecture 36: the file's order



her face shows through

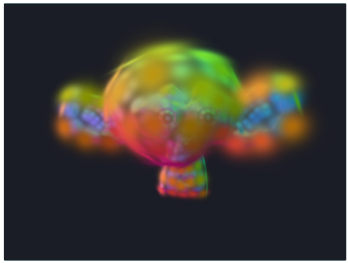
Lecture 36: her face showed through. Today: sort, then a scene

lecture 36: the file's order



her face shows through

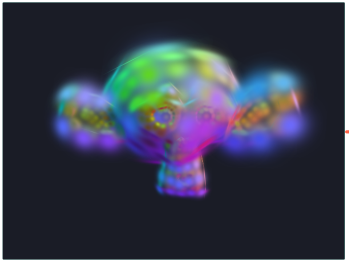
today: sorted



the back of her head

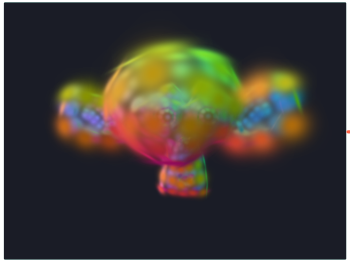
Lecture 36: her face showed through. Today: sort, then a scene

lecture 36: the file's order



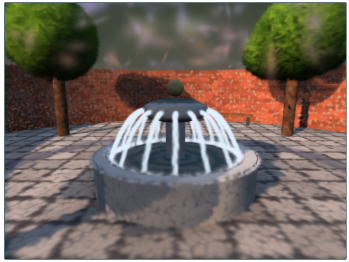
her face shows through

today: sorted



the back of her head

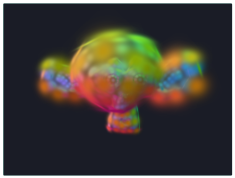
then a whole scene



110,394 blobs

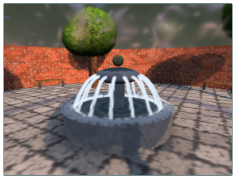
Today: three rounds of slides, then you try

1 **Sort**



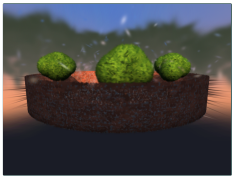
then You Try 1, 2

2 **A captured scene**



then You Try 3, 4

3 **What blobs cannot do**



then You Try 5

Open the Splat Lab now: the course calendar, lecture 37's row.

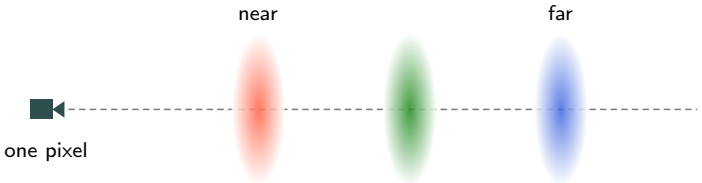
Pairs: one screen for two.

Round 1

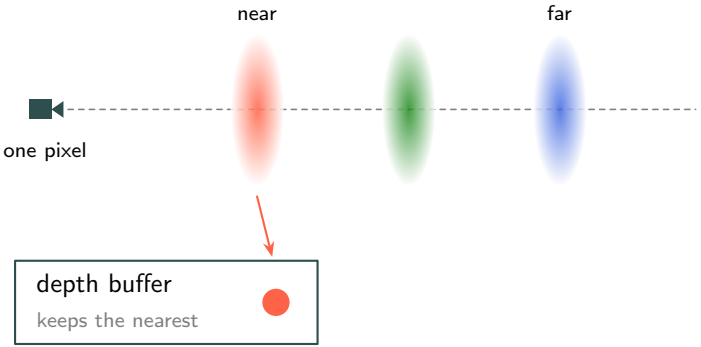
Sort

far to near, every time the camera moves

The depth buffer keeps one surface. A blend needs all of them

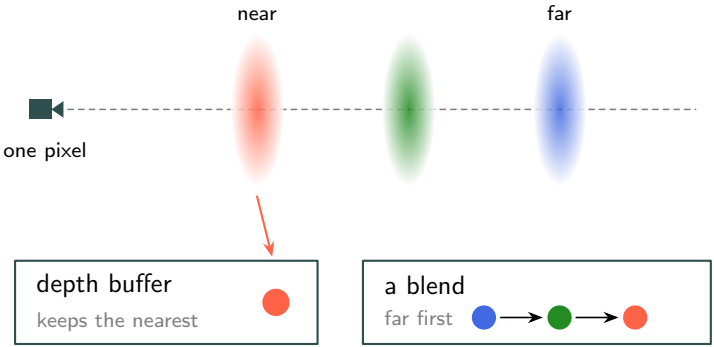


The depth buffer keeps one surface. A blend needs all of them



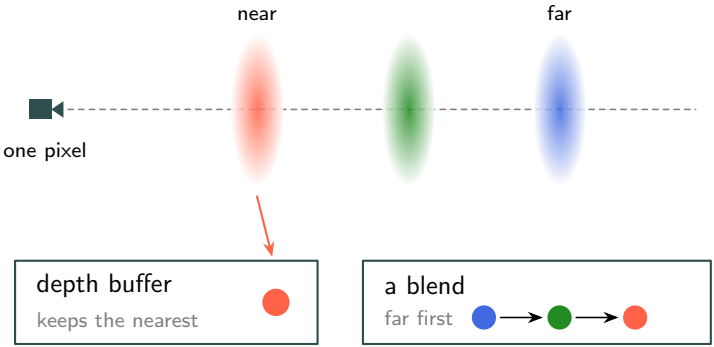
- Lecture 24: keep the nearest.

The depth buffer keeps one surface. A blend needs all of them



- Lecture 24: keep the nearest.
- A blend needs every one, in order.

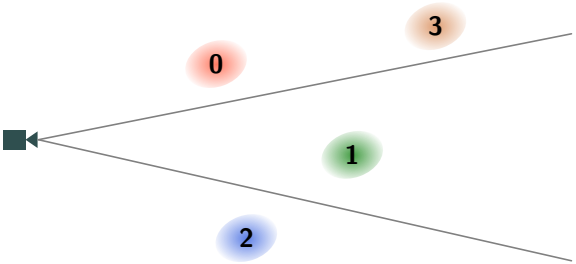
The depth buffer keeps one surface. A blend needs all of them



- Lecture 24: keep the nearest.
- A blend needs every one, in order.
- So: sort.

Sort far to near, every time the camera moves

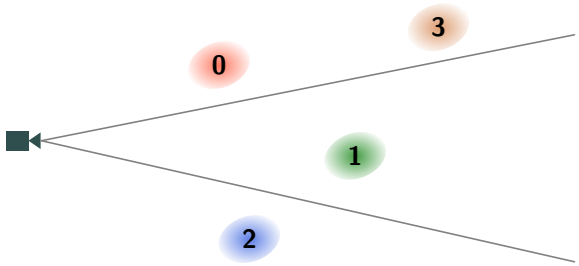
seen from above, numbered as in the file



- Distance: in front of the camera.

Sort far to near, every time the camera moves

seen from above, numbered as in the file

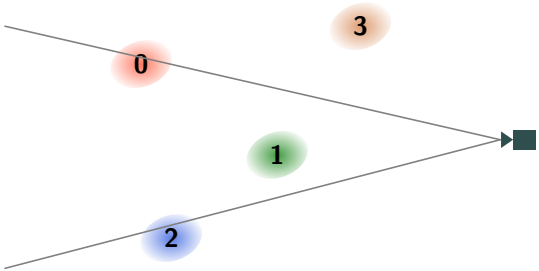


order: 3 1 2 0 far → near

- Distance: in front of the camera.
- Draw the farthest first.

Sort far to near, every time the camera moves

seen from above, numbered as in the file



order: 0 2 1 3 far → near

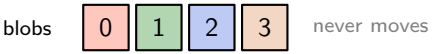
- Distance: in front of the camera.
- Draw the farthest first.
- The camera moves: sort again.

The order: splat.py writes it, vs_main reads it

splat.py, every frame:

```
depth = -(centres @ V[2, :3] + V[2, 3])
```

V's third row: lecture 28



▶ **Run it in the Splat Lab:**
Thick blobs, sorted far to near

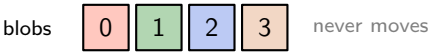
The order: splat.py writes it, vs_main reads it

splat.py, every frame:

```
depth = -(centres @ V[2, :3] + V[2, 3])
order = np.argsort(-depth)
```

V's third row: lecture 28

biggest distance first



never moves

▶ **Run it in the Splat Lab:**
Thick blobs, sorted far to near

The order: splat.py writes it, vs_main reads it

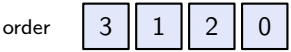
splat.py, every frame:

```
depth = -(centres @ V[2, :3] + V[2, 3])
order = np.argsort(-depth)
device.queue.write_buffer(order_buffer, 0, order)
```

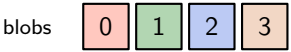
V's third row: lecture 28

biggest distance first

- A second storage buffer: the order.



new each frame



never moves

▶ **Run it in the Splat Lab:**
Thick blobs, sorted far to near

The order: splat.py writes it, vs_main reads it

splat.py, every frame:

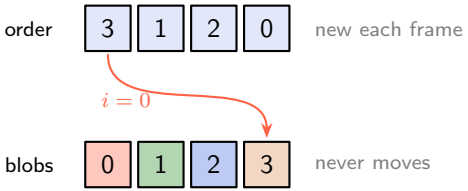
```
depth = -(centres @ V[2, :3] + V[2, 3])
order = np.argsort(-depth)
device.queue.write_buffer(order_buffer, 0, order)
```

V's third row: lecture 28

biggest distance first

vs_main, for the i-th square:

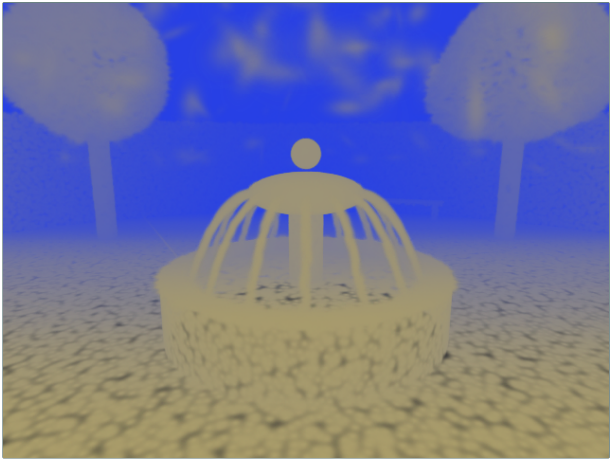
```
let blob = blobs[order[i]];
```



- A second storage buffer: the order.
- Lecture 31: a buffer of triangles.

▶ **Run it in the Splat Lab:**
Thick blobs, sorted far to near

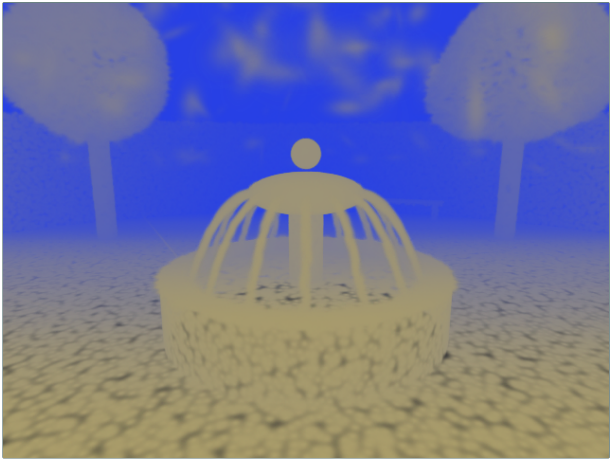
Each blob coloured by its turn: blue first, yellow last



drawn first — drawn last

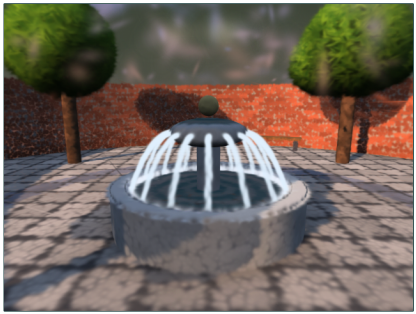
- Sky and wall: first.
- The near ground: last, on top.

Each blob coloured by its turn: blue first, yellow last



drawn first — drawn last

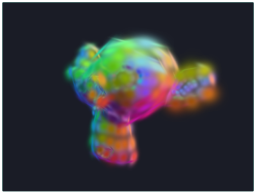
- Sky and wall: first.
- The near ground: last, on top.



the real colours

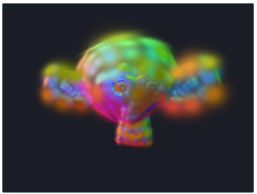
Suzanne fixed: the same blobs, sorted

from the front



the file's order

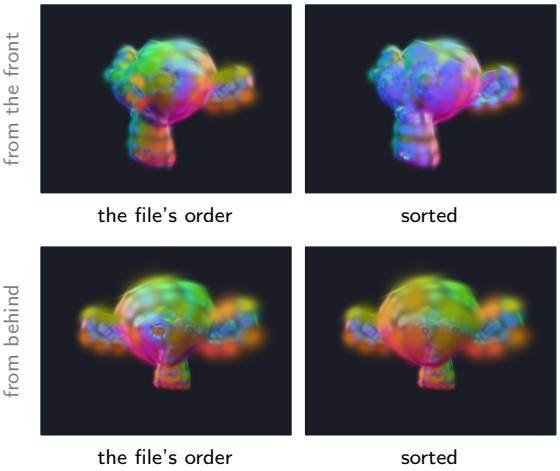
from behind



the file's order

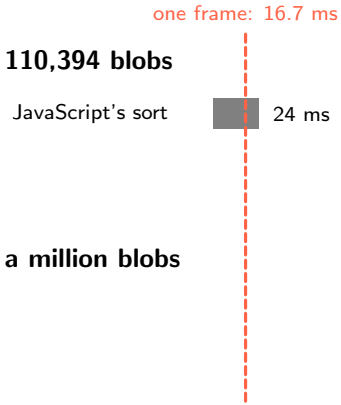
- Lecture 36: her face shows through.

Suzanne fixed: the same blobs, sorted



- Lecture 36: her face shows through.
- Sorted: right from every side.

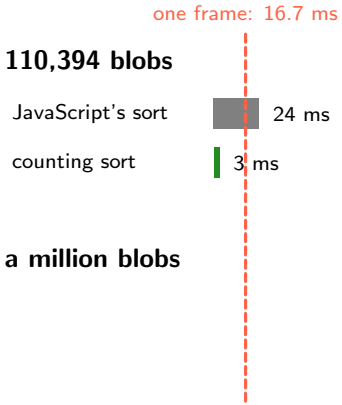
What a sort costs



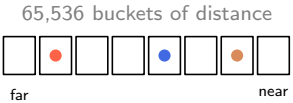
- A sort must fit in a frame.

Measured in node on the build machine (sort_time.js). The lab sorts this way, as antimatter15/splat does.

What a sort costs

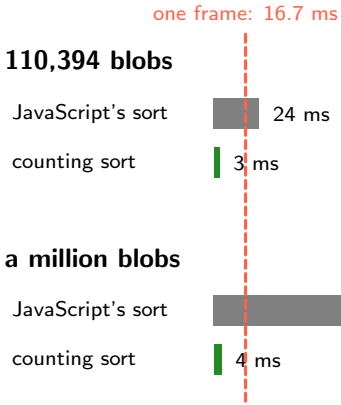


- A sort must fit in a frame.
- Drop each blob in a bucket.

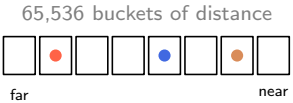


Measured in node on the build machine (sort_time.js). The lab sorts this way, as antimatter15/splat does.

What a sort costs



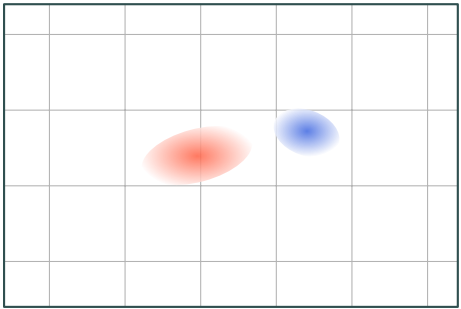
- A sort must fit in a frame.
- Drop each blob in a bucket.



Measured in node on the build machine (sort_time.js). The lab sorts this way, as antimatter15/splat does.

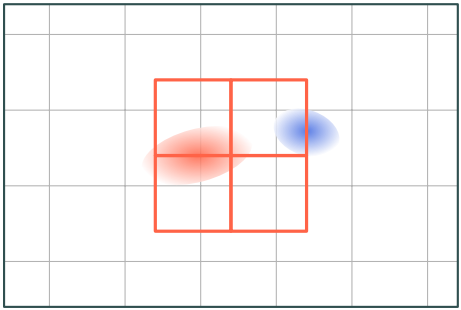
The paper's way: tiles, one GPU sort, front to back

the screen, in 16×16 pixel tiles



The paper's way: tiles, one GPU sort, front to back

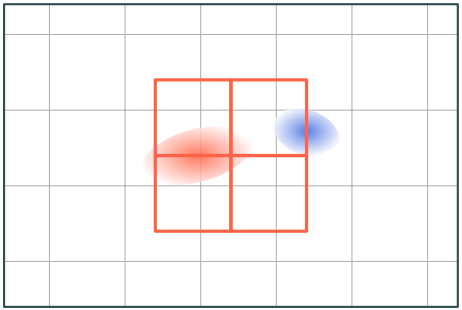
the screen, in 16×16 pixel tiles



one copy for each tile it touches

The paper's way: tiles, one GPU sort, front to back

the screen, in 16×16 pixel tiles

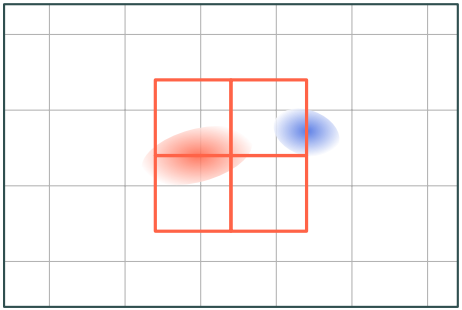


one copy for each tile it touches

one GPU sort of all copies,
by tile, then by distance

The paper's way: tiles, one GPU sort, front to back

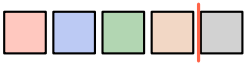
the screen, in 16×16 pixel tiles



one copy for each tile it touches

one GPU sort of all copies,
by tile, then by distance

one tile, near first:



stop: the pixel is full

Kerbl et al., 2023: section 6; appendix C (stop before 0.9999). Front to back needs its own blend formula.

You Try 1: turn the sort off

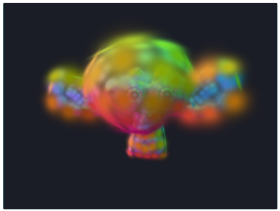
In the Splat Lab, choose **You Try 1**. Guess first, then change the code.

`// sort: far to near` change this line

Suzanne, from behind

- **The sort off.**
- Guess: her face?

now



after your change



You Try 1: turn the sort off

In the Splat Lab, choose **You Try 1**. Guess first, then change the code.

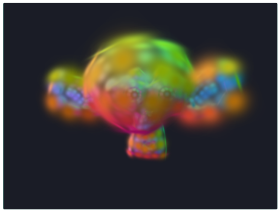
`// sort: far to near` change this line

Suzanne, from behind

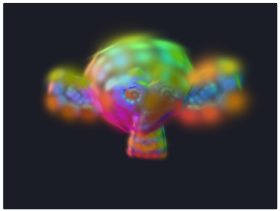
answer: `// sort: off`

- **The sort off.**
- Guess: her face?
- It shows through again.

now



after your change



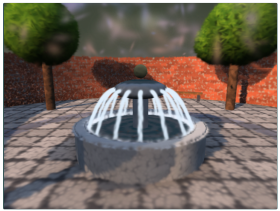
You Try 2: sort near to far, same blend

In the Splat Lab, choose **You Try 2**. Guess first, then change the code.

`// sort: far to near` change this line

- **Near to far.**
- Guess: what covers it?

now



after your change



You Try 2: sort near to far, same blend

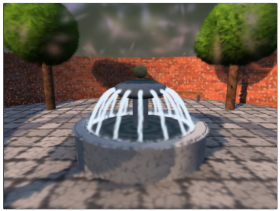
In the Splat Lab, choose **You Try 2**. Guess first, then change the code.

`// sort: far to near` change this line

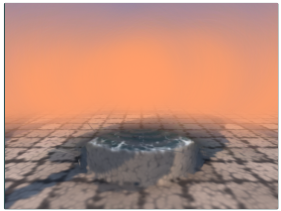
answer: `// sort: near to far`

- **Near to far.**
- Guess: what covers it?
- The sky: drawn last, on top.

now



after your change

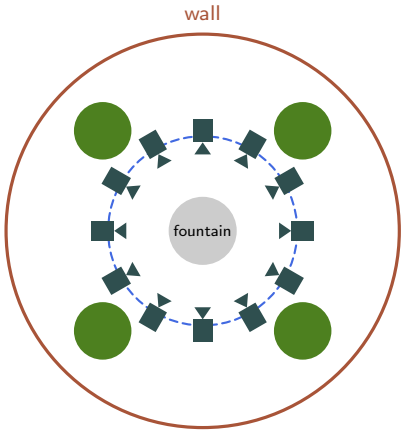


Round 2

A captured scene

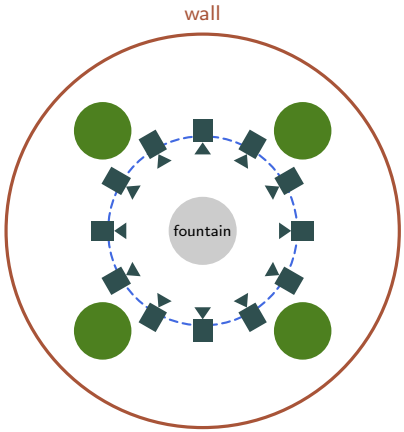
photos along a path become blobs

Photos taken along a path become a scene of blobs



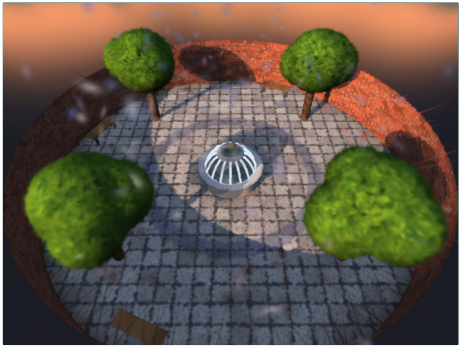
the path: a photo every few steps

Photos taken along a path become a scene of blobs



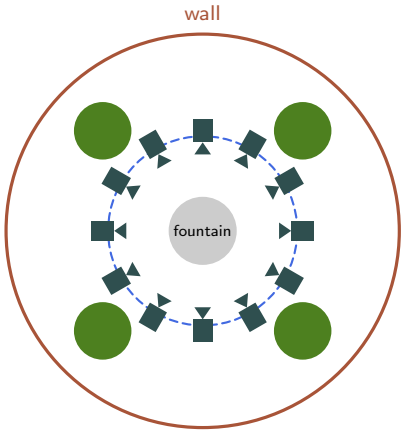
the path: a photo every few steps

lecture 38

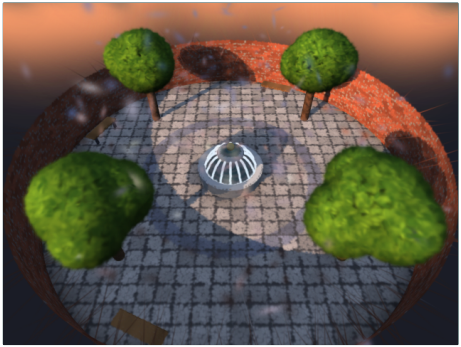


110,394 blobs, from above

Photos taken along a path become a scene of blobs



lecture 38



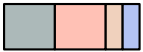
110,394 blobs, from above

the path: a photo every few steps

Today's courtyard is a stand-in, made by make_scene.py and stored as a capture is.

What is in the file, for each blob

.splat: 32 bytes



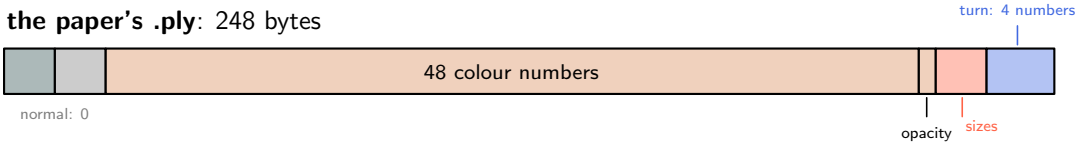
position sizes colour, opacity turn

What is in the file, for each blob

.splat: 32 bytes



the paper's .ply: 248 bytes

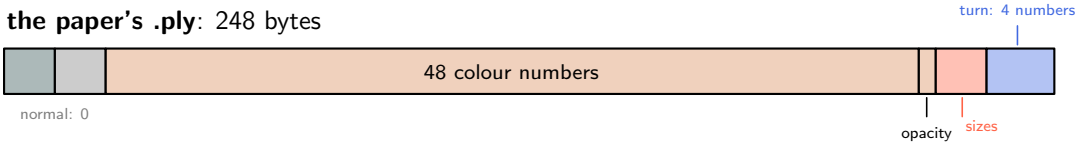


What is in the file, for each blob

.splat: 32 bytes



the paper's .ply: 248 bytes



courtyard: $110{,}394 \times 32$ bytes $= 3.5$ MB

a million blobs in .ply: 248 MB

the paper's scenes: 1 to 5 million blobs

Three sizes and a turn make lecture 36's three arrows

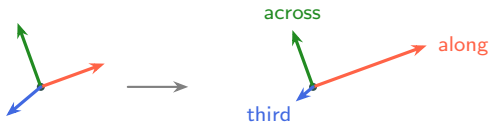


the turn

4 numbers w, x, y, z

- The paper's word: a quaternion.

Three sizes and a turn make lecture 36's three arrows



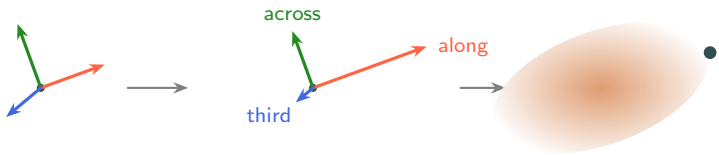
the turn

4 numbers w, x, y, z

times the 3 sizes

- The paper's word: a quaternion.

Three sizes and a turn make lecture 36's three arrows



the turn

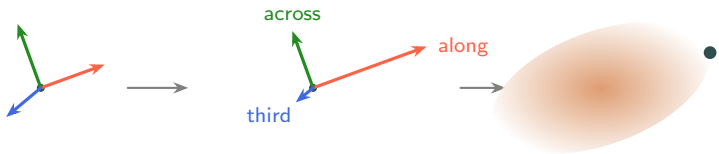
4 numbers w, x, y, z

times the 3 sizes

lecture 36's blob

- The paper's word: a quaternion.
- Lecture 36, slide 7: RS.

Three sizes and a turn make lecture 36's three arrows



- The paper's word: a quaternion.
- Lecture 36, slide 7: RS.

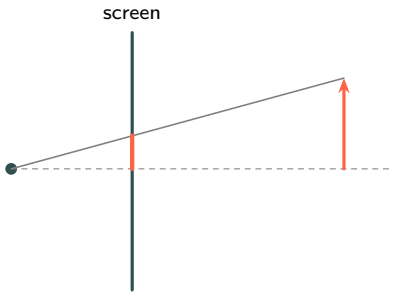
the turn — times the 3 sizes — lecture 36's blob

4 numbers w, x, y, z

splat.py, arrows(), first column, for reading:

```
column1 = [1 - 2 * (y * y + z * z), 2 * (x * y + w * z), 2 * (x * z - w * y)]
along = [v * sizes[0] for v in column1]
```

One arrow's shadow on the screen

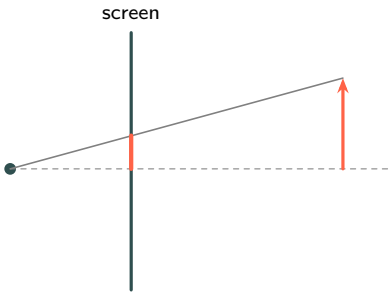


$f \times \text{size} / \text{dist}$

lecture 28, slide 21

- Across: lecture 28's rule.

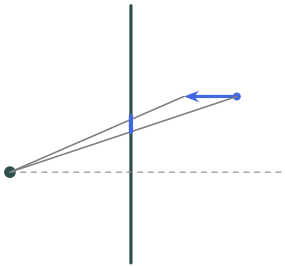
One arrow's shadow on the screen



across the view

$f \times \text{size} / \text{dist}$

lecture 28, slide 21



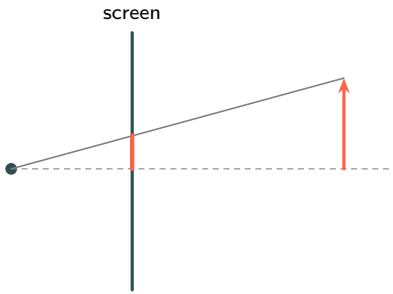
toward the camera

it leans out

only off the middle

- Across: lecture 28's rule.
- Toward you: a lean.

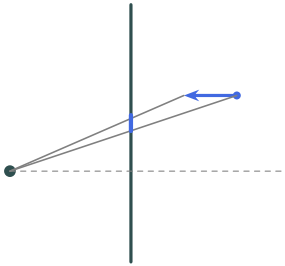
One arrow's shadow on the screen



across the view

$f \times \text{size} / \text{dist}$

lecture 28, slide 21



toward the camera

it leans out

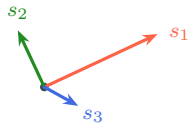
only off the middle

- Across: lecture 28's rule.
- Toward you: a lean.

```
return u.focal / dist * vec2f(d.x + t.x * d.z / dist, d.y + t.y * d.z / dist);
```

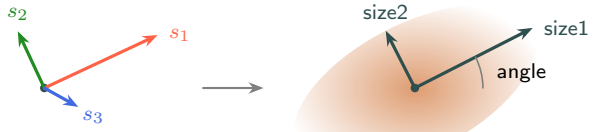
on_screen(d, t): the arrow d at the point t, both seen from the camera; in pixels

Three shadows make one ellipse: lecture 35's blob



three shadows

Three shadows make one ellipse: lecture 35's blob

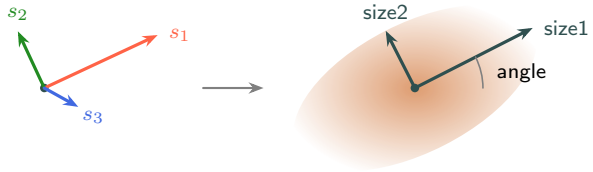


three shadows

one ellipse

- Two sizes, an angle.

Three shadows make one ellipse: lecture 35's blob



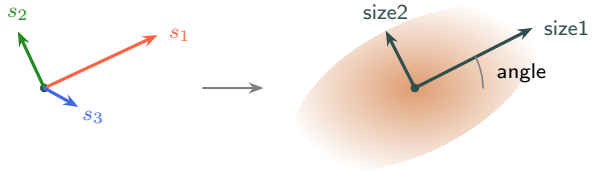
three shadows one ellipse

- Two sizes, an angle.
- Three numbers say the shape.

wide $= \sum x^2$ lean $= \sum xy$ tall $= \sum y^2$

lecture 36's $(RS)(RS)^\mathsf{T}$, on the screen

Three shadows make one ellipse: lecture 35's blob



three shadows one ellipse

- Two sizes, an angle.
- Three numbers say the shape.

$$\text{wide} = \sum x^2 \quad \text{lean} = \sum xy \quad \text{tall} = \sum y^2$$

lecture 36's $(RS)(RS)^\top$, on the screen

$$\text{angle} = \tfrac{1}{2}\,\text{atan2}(2\,\text{lean},\ \text{wide} - \text{tall})$$

Worked: one blob's ellipse

$f = 400$ pixels. Centre straight ahead, dist 2.

along $(0.1, 0.1, 0)$ across $(-0.05, 0.05, 0)$ third $(0, 0, 0.2)$

Try first: the three shadows, then the sizes.

Worked: one blob's ellipse

$f = 400$ pixels. Centre straight ahead, dist 2.

along $(0.1, 0.1, 0)$ across $(-0.05, 0.05, 0)$ third $(0, 0, 0.2)$

Try first: the three shadows, then the sizes.

shadows: $(20, 20)$ $(-10, 10)$ $(0, 0)$



third: a dot

- An arrow at you: no shadow.

Worked: one blob's ellipse

$f = 400$ pixels. Centre straight ahead, dist 2.

along $(0.1, 0.1, 0)$ across $(-0.05, 0.05, 0)$ third $(0, 0, 0.2)$

Try first: the three shadows, then the sizes.

shadows: $(20, 20)$ $(-10, 10)$ $(0, 0)$

wide 500 lean 300 tall 500



third: a dot

- An arrow at you: no shadow.

Worked: one blob's ellipse

$f = 400$ pixels. Centre straight ahead, dist 2.

along $(0.1, 0.1, 0)$ across $(-0.05, 0.05, 0)$ third $(0, 0, 0.2)$

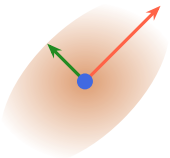
Try first: the three shadows, then the sizes.

shadows: $(20, 20)$ $(-10, 10)$ $(0, 0)$

wide 500 lean 300 tall 500

angle $45°$ sizes 28.3 and 14.1 pixels

$= \sqrt{800}$ and $\sqrt{200}$: the two shadows' lengths



third: a dot

- An arrow at you: no shadow.

vs_main for a thick blob

```
let blob = blobs[order[i]];
let q = reach * square[k];          // lecture 36
```

▶ **Run it in the Splat Lab:**
Thick blobs, sorted far to near

vs_main for a thick blob

```
let blob = blobs[order[i]];
let q = reach * square[k];          // lecture 36

if (-t.z < 0.2 || abs(centre.x) > 1.3 * centre.w || ...)  { not drawn }
```

behind the camera, or far off the screen (the paper's 1.3)

▶ **Run it in the Splat Lab:**
Thick blobs, sorted far to near

vs_main for a thick blob

```
let blob = blobs[order[i]];
let q = reach * square[k];          // lecture 36

if (-t.z < 0.2 || abs(centre.x) > 1.3 * centre.w || ...)  { not drawn }

let s1 = on_screen(seen(along), t);   // and s2, s3
```

behind the camera, or far off the screen (the paper's 1.3)

▶ **Run it in the Splat Lab:**
Thick blobs, sorted far to near

vs_main for a thick blob

```
let blob = blobs[order[i]];
let q = reach * square[k];          // lecture 36

if (-t.z < 0.2 || abs(centre.x) > 1.3 * centre.w || ...)  { not drawn }

let s1 = on_screen(seen(along), t);   // and s2, s3

let wide = s1.x * s1.x + s2.x * s2.x + s3.x * s3.x + 0.3;  // lean, tall
```

behind the camera, or far off the screen (the paper's 1.3)

+0.3: at least about a pixel wide (the paper's code)

▶ **Run it in the Splat Lab:**
Thick blobs, sorted far to near

vs_main for a thick blob

```
let blob = blobs[order[i]];
let q = reach * square[k];          // lecture 36

if (-t.z < 0.2 || abs(centre.x) > 1.3 * centre.w || ...)  { not drawn }

let s1 = on_screen(seen(along), t);   // and s2, s3

let wide = s1.x * s1.x + s2.x * s2.x + s3.x * s3.x + 0.3;  // lean, tall

let angle = 0.5 * atan2(2.0 * lean, wide - tall);
let size1 = sqrt(middle + r);   let size2 = sqrt(middle - r);
```

behind the camera, or far off the screen (the paper's 1.3)

+0.3: at least about a pixel wide (the paper's code)

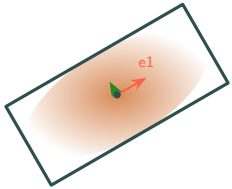
▶ **Run it in the Splat Lab:**
Thick blobs, sorted far to near

vs_main for a thick blob

```
let blob = blobs[order[i]];
let q = reach * square[k];          // lecture 36

if (-t.z < 0.2 || abs(centre.x) > 1.3 * centre.w || ...)  { not drawn }

let s1 = on_screen(seen(along), t);   // and s2, s3

let wide = s1.x * s1.x + s2.x * s2.x + s3.x * s3.x + 0.3;  // lean, tall

let angle = 0.5 * atan2(2.0 * lean, wide - tall);
let size1 = sqrt(middle + r);   let size2 = sqrt(middle - r);

let pixels = q.x * e1 + q.y * e2;
out.pos = centre + vec4f(2.0 * pixels / u.resolution * centre.w, 0.0, 0.0);
```

behind the camera, or far off the screen (the paper's 1.3)

+0.3: at least about a pixel wide (the paper's code)



the square, on the screen

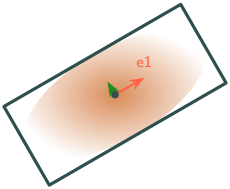
▶ **Run it in the Splat Lab:**
Thick blobs, sorted far to near

vs_main for a thick blob

```
let blob = blobs[order[i]];
let q = reach * square[k];          // lecture 36

if (-t.z < 0.2 || abs(centre.x) > 1.3 * centre.w || ...)  { not drawn }

let s1 = on_screen(seen(along), t);   // and s2, s3

let wide = s1.x * s1.x + s2.x * s2.x + s3.x * s3.x + 0.3;  // lean, tall

let angle = 0.5 * atan2(2.0 * lean, wide - tall);
let size1 = sqrt(middle + r);   let size2 = sqrt(middle - r);

let pixels = q.x * e1 + q.y * e2;
out.pos = centre + vec4f(2.0 * pixels / u.resolution * centre.w, 0.0, 0.0);
```

behind the camera, or far off the screen (the paper's 1.3)
+0.3: at least about a pixel wide (the paper's code)

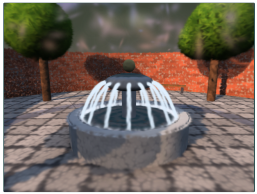
fs_main: lecture 36's, unchanged.



the square, on the screen

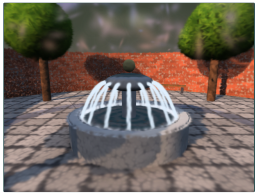
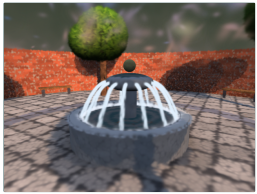
▶ **Run it in the Splat Lab:**
Thick blobs, sorted far to near

The courtyard, walked round the path



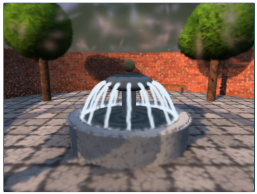
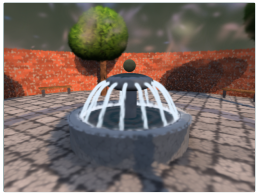
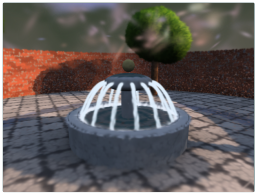
110,394 blobs, sorted every frame.

The courtyard, walked round the path

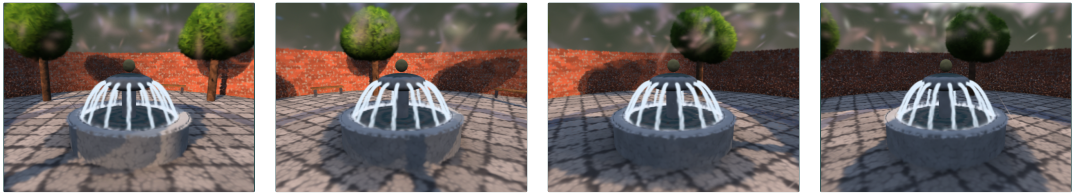
110,394 blobs, sorted every frame.

The courtyard, walked round the path

110,394 blobs, sorted every frame.

The courtyard, walked round the path



110,394 blobs, sorted every frame.

In the lab: **On the path**, then **Walk**.

You Try 3: see the cloud

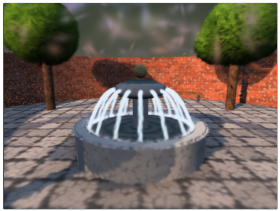
In the Splat Lab, choose **You Try 3**. Guess first, then change the code.

```
let grow = 1.0;
let along = grow * blob.along;
```

change this line

- **A fifth of the size.**
- Guess: made of what?

now



after your change

?

You Try 3: see the cloud

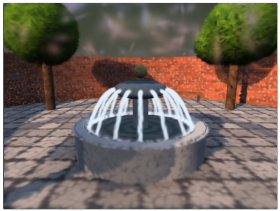
In the Splat Lab, choose **You Try 3**. Guess first, then change the code.

```
let grow = 1.0;
let along = grow * blob.along;
```

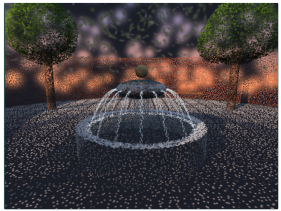
change this line

answer: `let grow = 0.2;`

now



after your change



- **A fifth of the size.**
- Guess: made of what?
- Dots. Nothing between them.

You Try 4: leave out the faint blobs

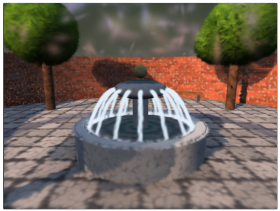
In the Splat Lab, choose **You Try 4**. Guess first, then change the code.

```
let faintest = 0.0;
```
change this line

```
if (opacity < faintest) { discard; }
```
discard: no colour here

- **Under 0.5: left out.**
- Guess: what goes?

now



after your change



You Try 4: leave out the faint blobs

In the Splat Lab, choose **You Try 4**. Guess first, then change the code.

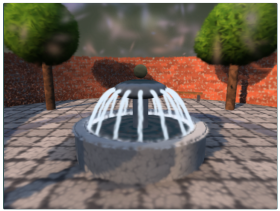
```
let faintest = 0.0;
```
change this line

```
if (opacity < faintest) { discard; }
```
discard: no colour here

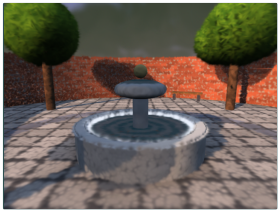
- **Under 0.5: left out.**
- Guess: what goes?
- The water: all faint blobs.

answer: `let faintest = 0.5;`

now



after your change



Round 3

What blobs cannot do

the flaws, in pictures

Popping: the order flips, the colour jumps

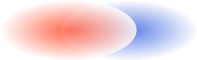
from above





here

what the camera sees



red on top

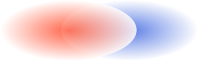
Popping: the order flips, the colour jumps

from above





what the camera sees



red on top



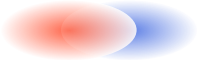
blue on top: a pop

Popping: the order flips, the colour jumps

from above



what the camera sees



red on top



blue on top: a pop



One distance a blob, at its centre: the order flips at once.

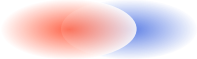
Popping: the order flips, the colour jumps

from above





what the camera sees



red on top



blue on top: a pop

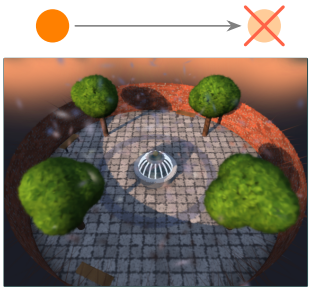
One distance a blob, at its centre: the order flips at once.

The 1.3 cut at the screen's edge: a blob appears at once.

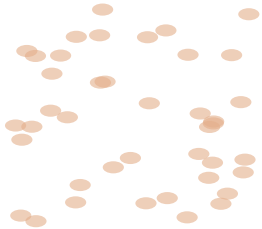
Kerbl et al. 2023, section 7.4: "Gaussians suddenly switching depth/blending order".
Since then: StopThePop (Radl et al., SIGGRAPH 2024) re-sorts for each small part of the screen.

The file holds only coloured blobs

The light is painted on

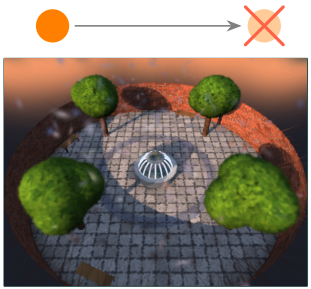


move the sun: the shadows stay



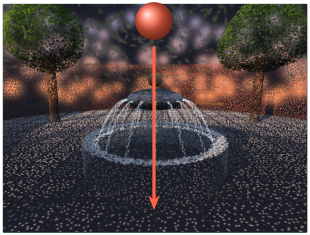
The file holds only coloured blobs

The light is painted on

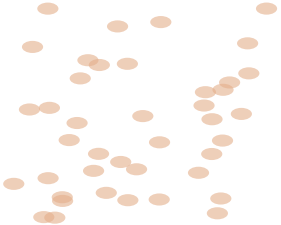


move the sun: the shadows stay

No surface

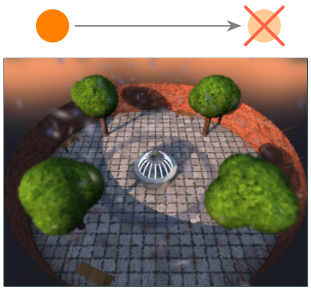


nothing to walk on or bump into



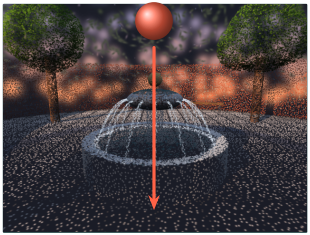
The file holds only coloured blobs

The light is painted on



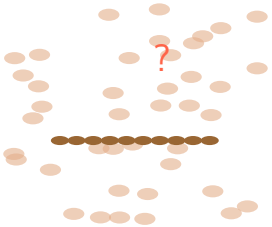
move the sun: the shadows stay

No surface



nothing to walk on or bump into

No objects



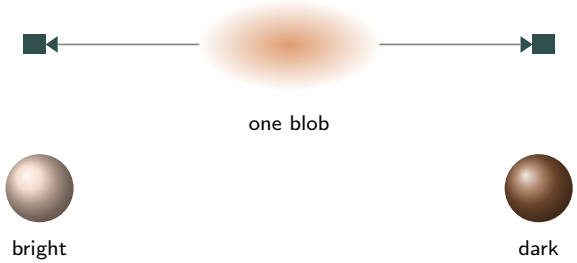
which blobs are the bench?

Colour that changes with the view



one blob

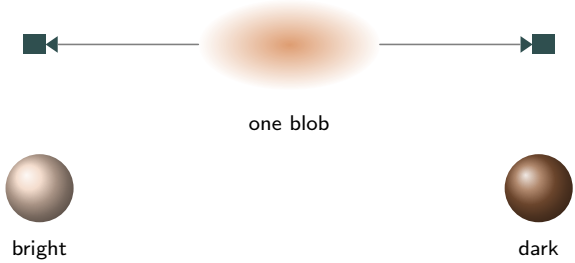
Colour that changes with the view



the paper: **48** colour numbers a blob

- A shine moves with the eye.
- The paper's word: spherical harmonics.

Colour that changes with the view



- A shine moves with the eye.
- The paper's word: spherical harmonics.

the paper: **48** colour numbers a blob

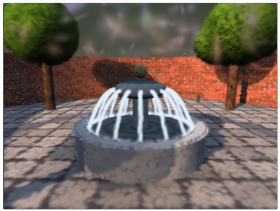
ours: 3, the same from every side

You Try 5: leave the path

In the Splat Lab, choose **You Try 5**. Guess first. No code today.

Press **Outside the wall**, or fly: W A S D.

now



after your change

?

- **Leave the path.**
- Guess: still real?

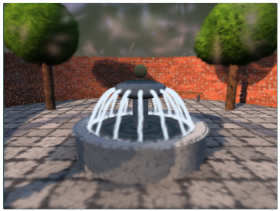
You Try 5: leave the path

In the Splat Lab, choose **You Try 5**. Guess first. No code today.

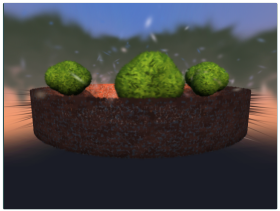
Press **Outside the wall**, or fly: W A S D.

make_scene.py put blobs only where the path could see.

now



after your change



- **Leave the path.**
- Guess: still real?
- Spikes, smudges, nothing outside.

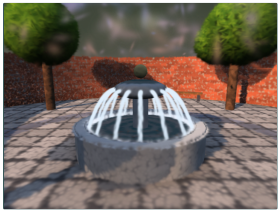
You Try 5: leave the path

In the Splat Lab, choose **You Try 5**. Guess first. No code today.

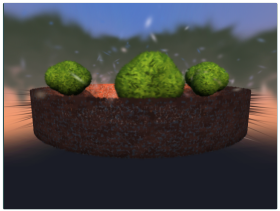
Press **Outside the wall**, or fly: W A S D.

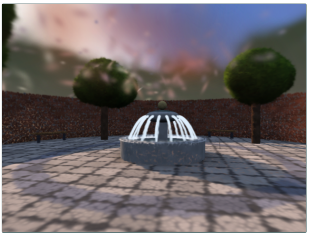
make_scene.py put blobs only where the path could see.

now



after your change





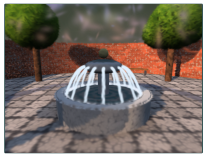
off the path, by the wall

- **Leave the path.**
- Guess: still real?
- Spikes, smudges, nothing outside.

Triangles and blobs



triangles

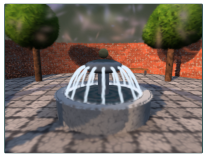


blobs

Triangles and blobs



triangles



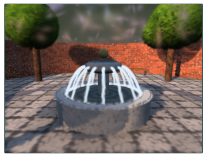
blobs

	triangles	blobs
made from photos	✗	✓
looks real for free	✗	✓

Triangles and blobs



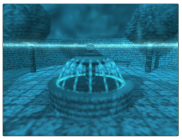
triangles



blobs

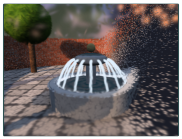
	triangles	blobs
made from photos	✗	✓
looks real for free	✗	✓
relight it	✓	✗
bump into it	✓	✗
edit one thing	✓	✗

Now make your own picture



A hologram

`scan lines, see-through`



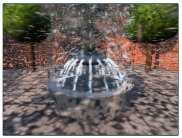
Blown away

`blobs drift off as dust`



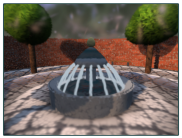
A camera lens

`out of focus: grow`



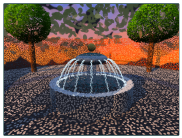
It rains into place

`each falls at its time`



A ripple

`a wave lifts the blobs`



A painting of dots

`small, hard, bright`

All six are in the Splat Lab, under **Ideas**. Change one. Mix two. Show your neighbour.

Summary

1	Sort	far to near, every time the camera moves
2	The order buffer	`blobs[order[i]]`
3	A scene file	32 or 248 bytes a blob
4	A thick blob on screen	three shadows $\rightarrow$ wide, lean, tall
5	What it gets wrong	popping; painted light; off the path

Next lecture: where the blobs come from

a photo



Photo: "Empty apartment room with corner windows" by aismallard, CC BY-SA 3.0.

- Photos go in.

Next lecture: where the blobs come from

a photo



blobs that fit it



Photo: "Empty apartment room with corner windows" by aismallard, CC BY-SA 3.0.

- Photos go in.
- Blobs nudge themselves to fit.

Exit Ticket

Today's question *(2 minutes)*

(a) $f = 400$ pixels. A blob's arrow $(0.2, 0, 0)$ lies across the view, at dist 4. How long is its shadow on the screen?

(b) A capture of 2 million blobs is stored in the paper's .ply file (248 bytes a blob). How many MB?

CS 453 — Exit Ticket #37

Name: _____

Roll #: _____ Date: _____

Answer:

Hand it to me on your way out.
No name, no attendance.