Lecture 35: Splatting 1

A Picture from Blobs

CS 453 — Computer Graphics

Fakhir Shaheen

Forman Christian University

A photo, rebuilt from soft blobs

a photograph



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

A photo, rebuilt from soft blobs

a photograph



5000 soft blobs



each blob: a centre, two sizes, an angle, a colour, an opacity

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

A photo, rebuilt from soft blobs

a photograph



5000 soft blobs



each blob: a centre, two sizes, an angle, a colour, an opacity

500 blobs, drawn hard



each blob as a solid ellipse: what the picture is made of

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

A photo, rebuilt from soft blobs

a photograph



5000 soft blobs



each blob: a centre, two sizes, an angle, a colour, an opacity

500 blobs, drawn hard



each blob as a solid ellipse: what the picture is made of

No triangles and no rays: only blobs, blended one over another.

Today: draw one blob, then many. Four lectures later: a whole 3D scene.

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

The idea comes from a 2023 paper

Kerbl, Kopanas, Leimkühler and Drettakis, *3D Gaussian Splatting for Real-Time Radiance Field Rendering*, SIGGRAPH 2023. Best in Show, SIGGRAPH 2023 Technical Papers.

photos of a place



The idea comes from a 2023 paper

Kerbl, Kopanas, Leimkühler and Drettakis, *3D Gaussian Splatting for Real-Time Radiance Field Rendering*, SIGGRAPH 2023. Best in Show, SIGGRAPH 2023 Technical Papers.

photos of a place





blobs that fit

lecture 38

The idea comes from a 2023 paper

Kerbl, Kopanas, Leimkühler and Drettakis, *3D Gaussian Splatting for Real-Time Radiance Field Rendering*, SIGGRAPH 2023. Best in Show, SIGGRAPH 2023 Technical Papers.

photos of a place

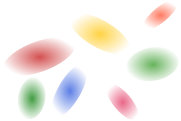


blobs that fit



lecture 38

millions of blobs in 3D



lectures 36, 37

The idea comes from a 2023 paper

Kerbl, Kopanas, Leimkühler and Drettakis, *3D Gaussian Splatting for Real-Time Radiance Field Rendering*, SIGGRAPH 2023. Best in Show, SIGGRAPH 2023 Technical Papers.

photos of a place



→

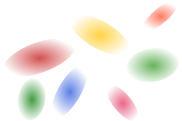
blobs that fit



lecture 38

→

millions of blobs in 3D



lectures 36, 37

→

a picture from



today, in 2D

The idea comes from a 2023 paper

Kerbl, Kopanas, Leimkühler and Drettakis, *3D Gaussian Splatting for Real-Time Radiance Field Rendering*, SIGGRAPH 2023. Best in Show, SIGGRAPH 2023 Technical Papers.

photos of a place

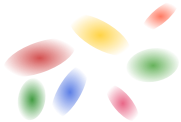


blobs that fit



lecture 38

millions of blobs in 3D



lectures 36, 37

a picture from



today, in 2D

The paper drew captured scenes in real time: 30 frames a second or more, at 1080p.

Its blobs are called Gaussians. Until lecture 36 we say blob.

Who uses it today (November 2026)

1 On a phone

Scaniverse (Niantic Spatial), a free app, makes a blob scene on the phone itself.

Who uses it today (November 2026)

1 On a phone

Scaniverse (Niantic Spatial), a free app, makes a blob scene on the phone itself.

2 A file standard

Khronos, who keep glTF, released a blob extension for testing in February 2026.

Who uses it today (November 2026)

1 On a phone

Scaniverse (Niantic Spatial), a free app, makes a blob scene on the phone itself.

2 A file standard

Khronos, who keep glTF, released a blob extension for testing in February 2026.

3 Maps of whole cities

Cesium streams blob scenes for maps, coarse far away and fine up close (April 2026).

Who uses it today (November 2026)

1 On a phone

Scaniverse (Niantic Spatial), a free app, makes a blob scene on the phone itself.

2 A file standard

Khronos, who keep glTF, released a blob extension for testing in February 2026.

3 Maps of whole cities

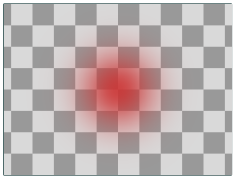
Cesium streams blob scenes for maps, coarse far away and fine up close (April 2026).

4 Not built in yet

Unreal Engine and Blender: only through add-ons, for now.

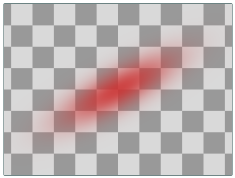
Today: three rounds of slides, then you try

1 One blob



then You Try 1, 2

2 Stretch and turn



then You Try 3, 4

3 Many blobs



then You Try 5, 6

Open the Blob Lab now. It is on the course calendar, in lecture 35's row.

A laptop or a phone will do. Work in pairs: one screen for two people is enough.

Flat pictures today: lecture 26's centred x and y, no camera and no 3D.

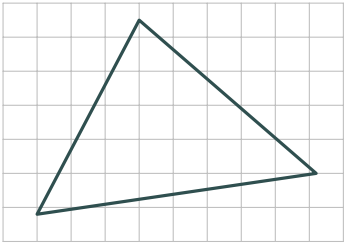
Round 1

One blob

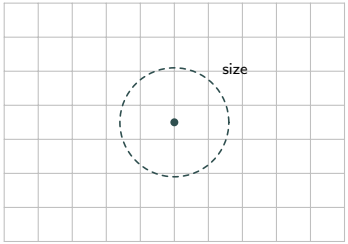
a weight between 0 and 1

A triangle: in or out. A blob: a weight

lecture 24 (slide 26)



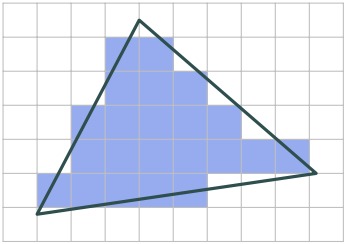
today: a blob



- Lecture 24: a pixel's centre is in the triangle or it is not.

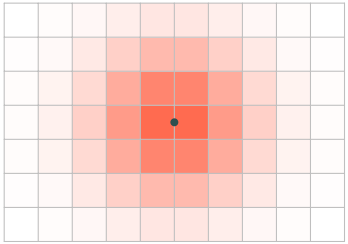
A triangle: in or out. A blob: a weight

lecture 24 (slide 26)



a pixel is in (1) or out (0)

today: a blob

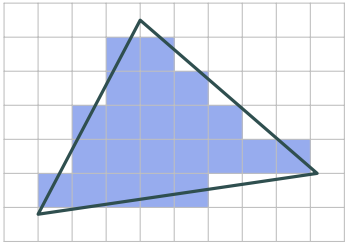


the darker, the bigger the weight

- Lecture 24: a pixel's centre is in the triangle or it is not.
- A blob gives every pixel a weight between 0 and 1.

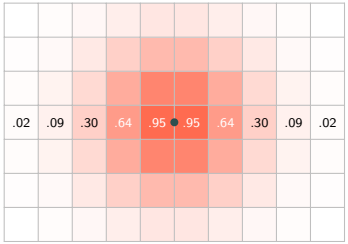
A triangle: in or out. A blob: a weight

lecture 24 (slide 26)



a pixel is in (1) or out (0)

today: a blob



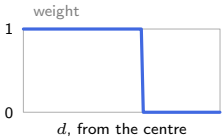
the darker, the bigger the weight

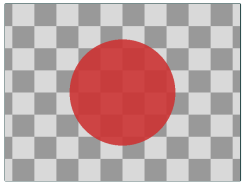
the middle row's weights, printed

- Lecture 24: a pixel's centre is in the triangle or it is not.
- A blob gives every pixel a weight between 0 and 1.
- 1 at the centre, falling smoothly as we move away. There is no edge.

Lecture 26's step and smoothstep, and the blob

lecture 26, slide 19





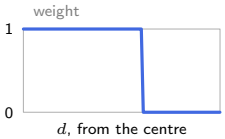
`1.0 - step(0.6, d)`

- step: in or out. A hard edge.

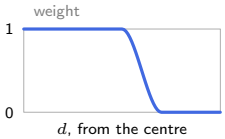
▶ **Run it in the Blob Lab:**
One round blob

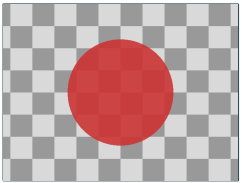
Lecture 26's step and smoothstep, and the blob

lecture 26, slide 19

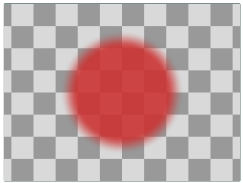


lecture 26, slide 20





`1.0 - step(0.6, d)`



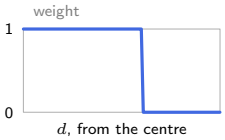
`1.0 - smoothstep(0.5, 0.7, d)`

- step: in or out. A hard edge.
- smoothstep: soft, but only in a thin band at the edge.

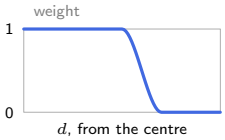
▶ **Run it in the Blob Lab:**
One round blob

Lecture 26's step and smoothstep, and the blob

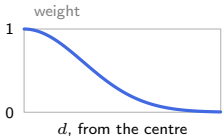
lecture 26, slide 19

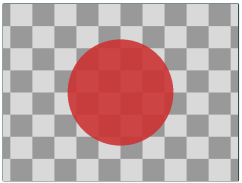


lecture 26, slide 20

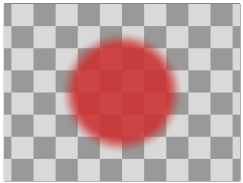


today

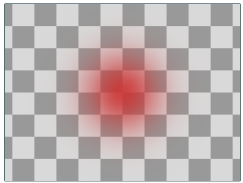




`1.0 - step(0.6, d)`



`1.0 - smoothstep(0.5, 0.7, d)`



`exp(-d*d / (2.0*size*size))`

d: the pixel's distance from the centre. Blob size 0.3; disc radius 0.6.

exp(v) is e^v: built into WGSL

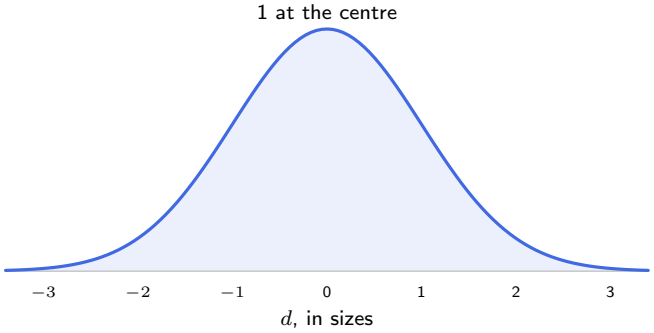
- step: in or out. A hard edge.
- smoothstep: soft, but only in a thin band at the edge.
- The blob: soft all the way out. Its weight never quite reaches 0.

▶ **Run it in the Blob Lab:**
One round blob

The weight falls with distance

$$\text{weight} = \exp\left(-\frac{d^2}{2\,\text{size}^2}\right)$$

d: how far the pixel is from the centre. size: how far the blob spreads.

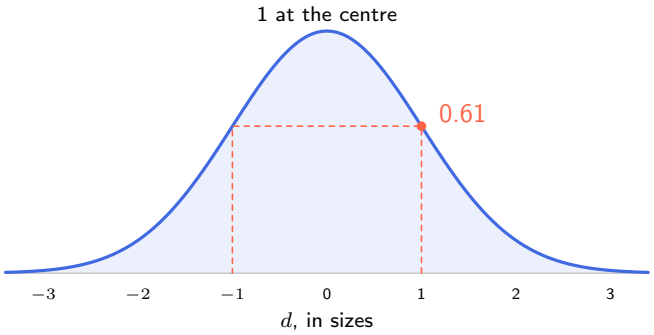


- The pixel at the centre: $d = 0$, so the weight is 1.

The weight falls with distance

$$\text{weight} = \exp\left(-\frac{d^2}{2\,\text{size}^2}\right)$$

d: how far the pixel is from the centre. size: how far the blob spreads.

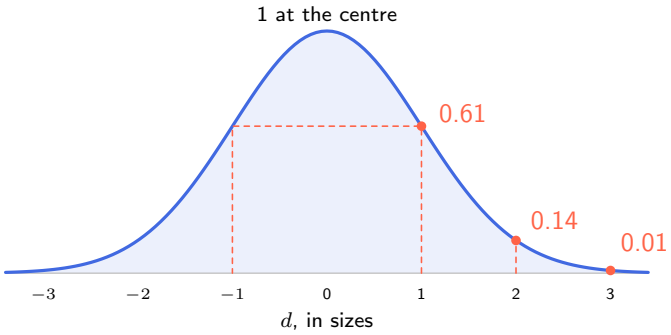


- The pixel at the centre: $d = 0$, so the weight is 1.
- One size away: 0.61.

The weight falls with distance

$$\text{weight} = \exp\left(-\frac{d^2}{2\,\text{size}^2}\right)$$

d: how far the pixel is from the centre. size: how far the blob spreads.

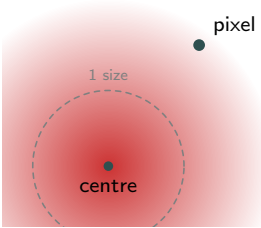


- The pixel at the centre: $d = 0$, so the weight is 1.
- One size away: 0.61.
- Two sizes: 0.14. Three: 0.01. Past three sizes we can stop drawing.

Worked: one blob's weight at a pixel

A blob: centre $(0.2, 0.1)$, size 0.25, opacity 0.8. A pixel at $(0.5, 0.5)$.

Try first: the weight at the pixel, and a = opacity $\times$ weight.



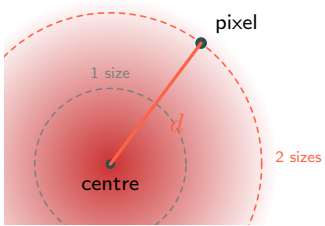
Worked: one blob's weight at a pixel

A blob: centre $(0.2, 0.1)$, size 0.25, opacity 0.8. A pixel at $(0.5, 0.5)$.

Try first: the weight at the pixel, and a = opacity $\times$ weight.

$$d = \sqrt{0.3^2 + 0.4^2} = 0.5$$

- The pixel is two sizes from the centre.



Worked: one blob's weight at a pixel

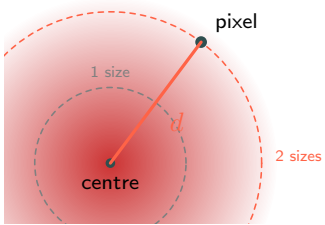
A blob: centre $(0.2, 0.1)$, size 0.25, opacity 0.8. A pixel at $(0.5, 0.5)$.

Try first: the weight at the pixel, and a = opacity $\times$ weight.

$$d = \sqrt{0.3^2 + 0.4^2} = 0.5$$

$$\text{weight} = e^{-2} = 0.14$$

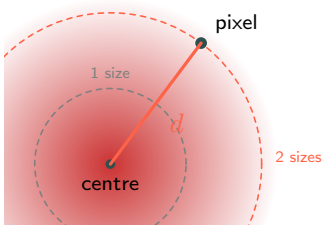
- The pixel is two sizes from the centre.
- Slide 9: at two sizes, 0.14.



Worked: one blob's weight at a pixel

A blob: centre $(0.2, 0.1)$, size 0.25, opacity 0.8. A pixel at $(0.5, 0.5)$.

Try first: the weight at the pixel, and a = opacity $\times$ weight.



$$d = \sqrt{0.3^2 + 0.4^2} = 0.5$$

$$\text{weight} = e^{-2} = 0.14$$

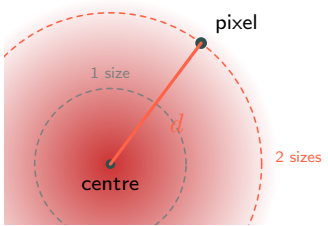
$$a = 0.8 \times 0.14 = 0.11$$

- The pixel is two sizes from the centre.
- Slide 9: at two sizes, 0.14.
- Opacity scales it down.

Worked: one blob's weight at a pixel

A blob: centre $(0.2, 0.1)$, size 0.25, opacity 0.8. A pixel at $(0.5, 0.5)$.

Try first: the weight at the pixel, and a = opacity $\times$ weight.



$$d = \sqrt{0.3^2 + 0.4^2} = 0.5$$

$$\text{weight} = e^{-2} = 0.14$$

$$a = 0.8 \times 0.14 = 0.11$$

$$c = 0.11 \cdot \text{colour} + 0.89 \cdot \text{background}$$

- The pixel is two sizes from the centre.
- Slide 9: at two sizes, 0.14.
- Opacity scales it down.
- The pixel is mostly background, with a little red.

Opacity: lecture 3's blend, one blob over the picture

lecture 3, slide 8: alpha blending

$$C = \alpha F + (1 - \alpha) B$$

- α parts of the new colour F, $1 - \alpha$ parts of the background B.

▶ **Run it in the Blob Lab:**
One round blob

Opacity: lecture 3's blend, one blob over the picture

lecture 3, slide 8: alpha blending

$$C = \alpha F + (1 - \alpha) B$$

today: the blob's α at a pixel

```
let a = opacity * weight;
let c = mix(background, colour, a);
```

mix: lecture 26, slide 11, with $t = a$

- α parts of the new colour F, $1 - \alpha$ parts of the background B.
- A blob's α at a pixel is its opacity times its weight.

▶ **Run it in the Blob Lab:**
One round blob

Opacity: lecture 3's blend, one blob over the picture

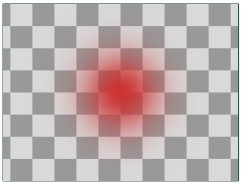
lecture 3, slide 8: alpha blending

$$C = \alpha F + (1 - \alpha) B$$

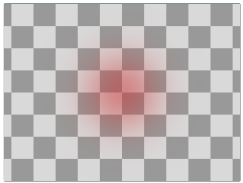
today: the blob's α at a pixel

```
let a = opacity * weight;
let c = mix(background, colour, a);
```

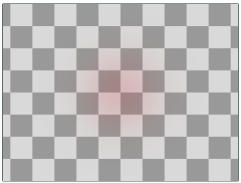
mix: lecture 26, slide 11, with $t = a$



opacity 1.0



opacity 0.5



opacity 0.2

- α parts of the new colour F, $1 - \alpha$ parts of the background B.
- A blob's α at a pixel is its opacity times its weight.
- A low opacity: we see the checks through the blob.

▶ **Run it in the Blob Lab:**
One round blob

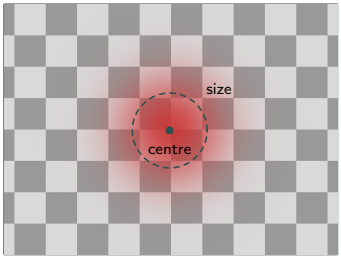
One blob in code

```
let background = checks(x, y);
```

the grey squares (given in the file)

```
let centre = vec2f(0.0, 0.0);
let size = 0.3;
let colour = vec3f(0.8, 0.2, 0.2);
let opacity = 0.9;
```

the blob's four numbers



x, y: this pixel, in lecture 26's centred coordinates. The lab has the whole file.

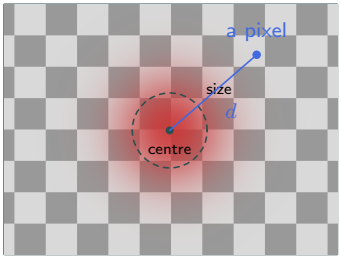
▶ **Run it in the Blob Lab:**
One round blob

One blob in code

```
let background = checks(x, y);
```
the grey squares (given in the file)

```
let centre = vec2f(0.0, 0.0);
let size = 0.3;
let colour = vec3f(0.8, 0.2, 0.2);
let opacity = 0.9;
```
the blob's four numbers

```
let d = length(vec2f(x, y) - centre);
let weight = exp(-d * d / (2.0 * size * size));
```
slide 9



x, y: this pixel, in lecture 26's centred coordinates. The lab has the whole file.

▶ **Run it in the Blob Lab:**
One round blob

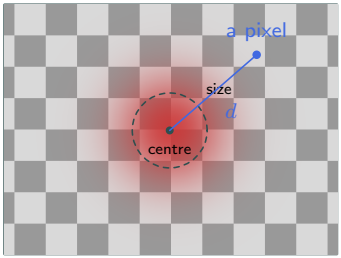
One blob in code

```
let background = checks(x, y);
```
the grey squares (given in the file)

```
let centre = vec2f(0.0, 0.0);
let size = 0.3;
let colour = vec3f(0.8, 0.2, 0.2);
let opacity = 0.9;
```
the blob's four numbers

```
let d = length(vec2f(x, y) - centre);
let weight = exp(-d * d / (2.0 * size * size));
```
slide 9

```
let a = opacity * weight;
let c = mix(background, colour, a);
```
slide 11



x, y: this pixel, in lecture 26's centred coordinates. The lab has the whole file.

▶ **Run it in the Blob Lab:**
One round blob

You Try 1: a smaller blob

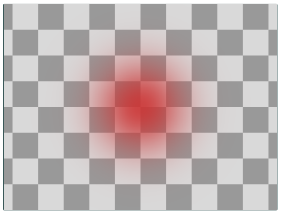
In the Blob Lab, choose **You Try 1**. Guess first, then change the code.

```
let centre = vec2f(0.0, 0.0);
let size = 0.3;
let colour = vec3f(0.8, 0.2, 0.2);
```

change this line

- **Make the blob's size 0.1.**
- Guess first: does the blob get smaller, fainter, or both?

now



after your change



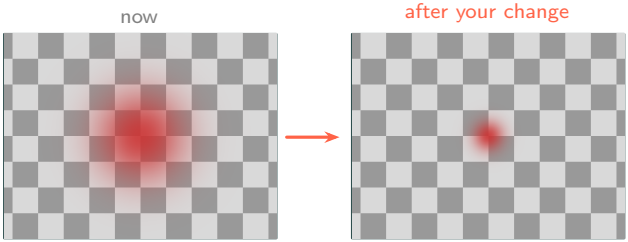
You Try 1: a smaller blob

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

```
let centre = vec2f(0.0, 0.0);
let size = 0.3;
let colour = vec3f(0.8, 0.2, 0.2);
```

change this line

answer: `let size = 0.1;`



- **Make the blob's size 0.1.**
- Guess first: does the blob get smaller, fainter, or both?
- Only smaller. At its centre the weight is still 1, so it is just as red there.

You Try 2: see through it

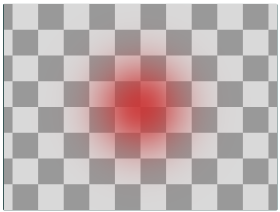
In the Blob Lab, choose **You Try 2**. Guess first, then change the code.

```
let colour = vec3f(0.8, 0.2, 0.2);
let opacity = 0.9;
```

change this line

- **Make the blob's opacity 0.4.**
- Guess first: what do you see at the blob's centre?

now



after your change



You Try 2: see through it

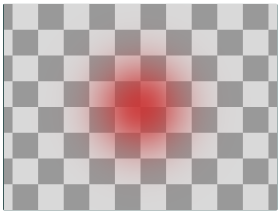
In the Blob Lab, choose **You Try 2**. Guess first, then change the code.

```
let colour = vec3f(0.8, 0.2, 0.2);
let opacity = 0.9;
```

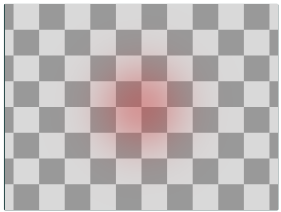
change this line

answer: `let opacity = 0.4;`

now



after your change



- **Make the blob's opacity 0.4.**
- Guess first: what do you see at the blob's centre?
- The checks, through a pink haze: at the centre $a = 0.4$, so 60% of the pixel is background.

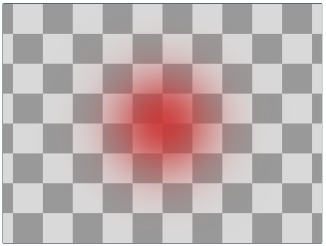
Round 2

Stretch and turn

two sizes and an angle

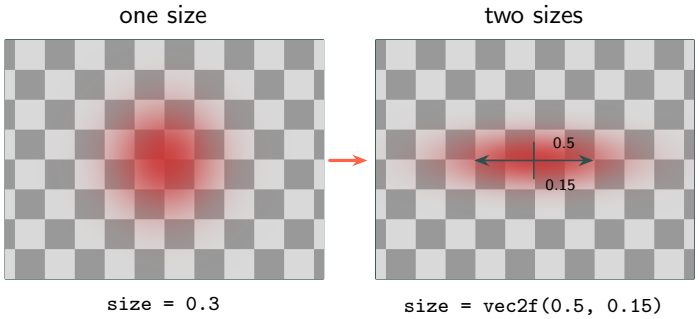
Two sizes and an angle: lecture 7's scale and rotation

one size



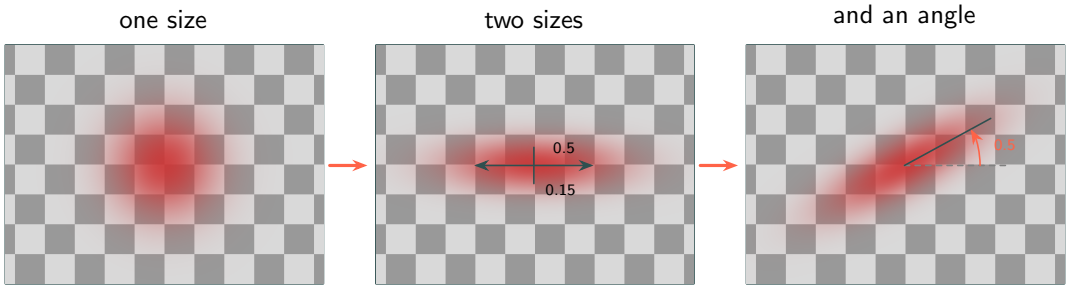
`size = 0.3`

Two sizes and an angle: lecture 7's scale and rotation



Lecture 7 (slide 38): unequal scales stretch a circle into an ellipse.

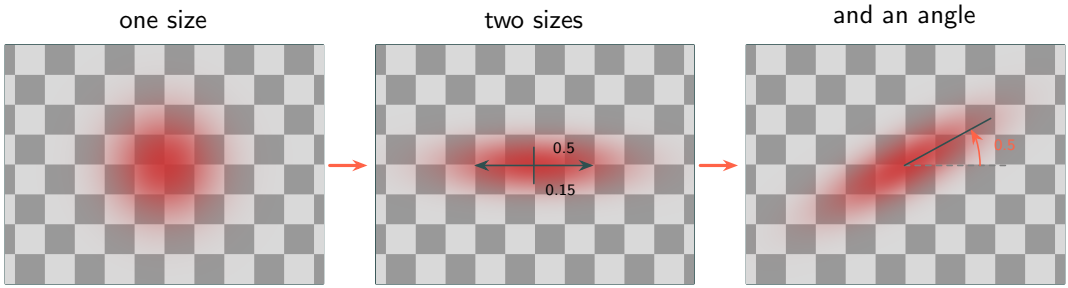
Two sizes and an angle: lecture 7's scale and rotation



Lecture 7 (slide 38): unequal scales stretch a circle into an ellipse.

Lecture 7 (slide 42): a rotation turns it, keeping its lengths.

Two sizes and an angle: lecture 7's scale and rotation



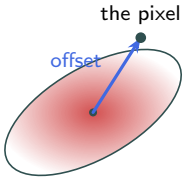
Lecture 7 (slide 38): unequal scales stretch a circle into an ellipse.

Lecture 7 (slide 42): a rotation turns it, keeping its lengths.

The blob's shape is now three numbers: two sizes and an angle (in radians).

To ask a pixel, undo the turn and the stretch

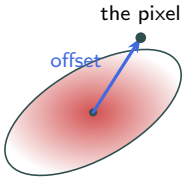
1. from the centre



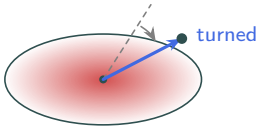

```
offset = vec2f(x, y) - centre
```

To ask a pixel, undo the turn and the stretch

1. from the centre



2. turn back by the angle

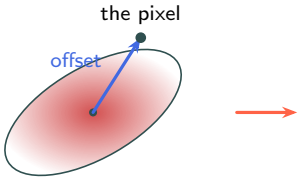


`offset = vec2f(x, y) - centre`

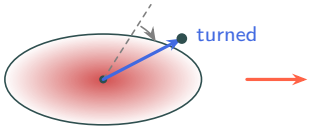
`turned: rotate by −angle`

To ask a pixel, undo the turn and the stretch

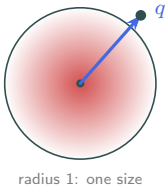
1. from the centre



2. turn back by the angle



3. divide by the two sizes



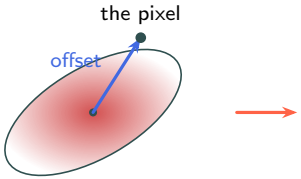

```
offset = vec2f(x, y) - centre
```

turned: rotate by −angle

```
q = turned / size
```

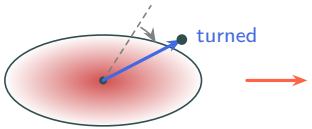
To ask a pixel, undo the turn and the stretch

1. from the centre



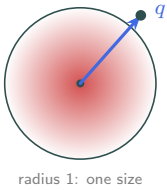
`offset = vec2f(x, y) - centre`

2. turn back by the angle



`turned: rotate by −angle`

3. divide by the two sizes



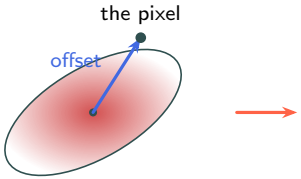
`q = turned / size`

$$\text{weight} \;=\; \exp\!\left(-\tfrac{1}{2}\, q \cdot q\right)$$

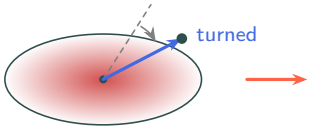
the round blob's question, now asked about q

To ask a pixel, undo the turn and the stretch

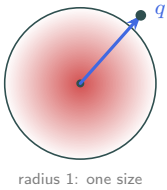
1. from the centre



2. turn back by the angle



3. divide by the two sizes



`offset = vec2f(x, y) - centre`

`turned:  rotate by −angle`

`q = turned / size`

$$\text{weight} \;=\; \exp\!\left(-\tfrac{1}{2}\, q \cdot q\right)$$

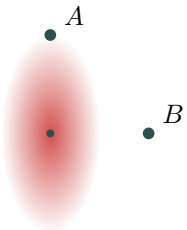
the round blob's question, now asked about q

Both sizes equal: $q \cdot q = d^2/\text{size}^2$, and this is slide 9's weight.

Worked: two pixels, the same distance from the centre

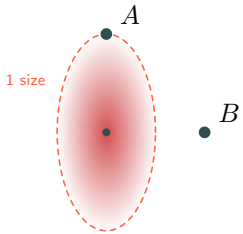
A blob: centre $(0, 0)$, size $(0.5, 0.25)$, angle 1.57 (a quarter turn). Pixels $A\,(0, 0.5)$ and $B\,(0.5, 0)$.

Try first: each pixel's weight. Both are 0.5 from the centre.



Worked: two pixels, the same distance from the centre

A blob: centre $(0, 0)$, size $(0.5, 0.25)$, angle 1.57 (a quarter turn). Pixels $A\,(0, 0.5)$ and $B\,(0.5, 0)$.
Try first: each pixel's weight. Both are 0.5 from the centre.



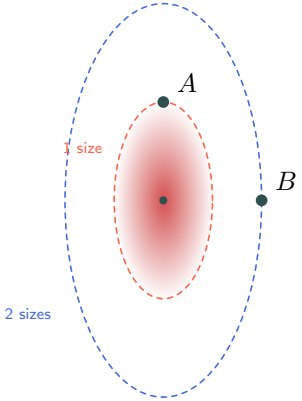
A: turned $(0.5, 0)$, $q = (1, 0)$, weight $e^{-0.5} = 0.61$

turned: cos and sin of 1.57 are 0 and 1

Worked: two pixels, the same distance from the centre

A blob: centre $(0, 0)$, size $(0.5, 0.25)$, angle 1.57 (a quarter turn). Pixels $A\,(0, 0.5)$ and $B\,(0.5, 0)$.

Try first: each pixel's weight. Both are 0.5 from the centre.



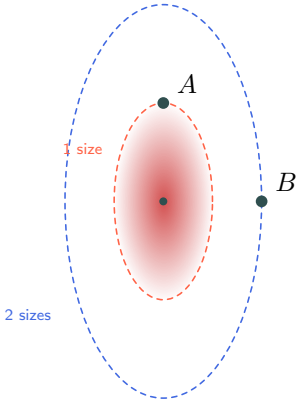
A: turned $(0.5, 0)$, $q = (1, 0)$, weight $e^{-0.5} = 0.61$

B: turned $(0, -0.5)$, $q = (0, -2)$, weight $e^{-2} = 0.14$

turned: cos and sin of 1.57 are 0 and 1

Worked: two pixels, the same distance from the centre

A blob: centre $(0, 0)$, size $(0.5, 0.25)$, angle 1.57 (a quarter turn). Pixels $A\,(0, 0.5)$ and $B\,(0.5, 0)$.
Try first: each pixel's weight. Both are 0.5 from the centre.



A: turned $(0.5, 0)$, $q = (1, 0)$, weight $e^{-0.5} = 0.61$

B: turned $(0, -0.5)$, $q = (0, -2)$, weight $e^{-2} = 0.14$

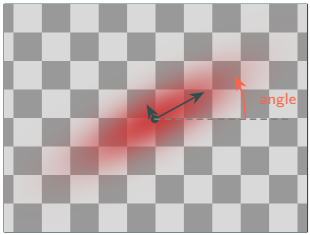
turned: cos and sin of 1.57 are 0 and 1

The same distance, but along the blob it is one size; across it, two.

The stretched blob in code

```
let size = vec2f(0.5, 0.15);
let angle = 0.5;
```

along the blob and across it; the turn, in radians



▶ **Run it in the Blob Lab:**
One blob, stretched and turned

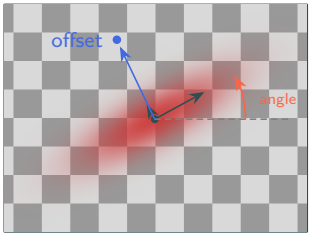
The stretched blob in code

```
let size = vec2f(0.5, 0.15);
let angle = 0.5;
```

along the blob and across it; the turn, in radians

```
let offset = vec2f(x, y) - centre;
```

from the centre to this pixel



▶ **Run it in the Blob Lab:**
One blob, stretched and turned

The stretched blob in code

```
let size = vec2f(0.5, 0.15);
let angle = 0.5;
```

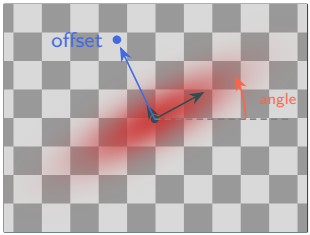
along the blob and across it; the turn, in radians

```
let offset = vec2f(x, y) - centre;
```

from the centre to this pixel

```
let turned = vec2f(cos(angle) * offset.x + sin(angle) * offset.y,
                  -sin(angle) * offset.x + cos(angle) * offset.y);
```

lecture 7's rotation by −angle: the offset, turned back



▶ **Run it in the Blob Lab:**
One blob, stretched and turned

The stretched blob in code

```
let size = vec2f(0.5, 0.15);
let angle = 0.5;
```

along the blob and across it; the turn, in radians

```
let offset = vec2f(x, y) - centre;
```

from the centre to this pixel

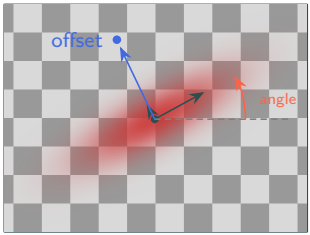
```
let turned = vec2f(cos(angle) * offset.x + sin(angle) * offset.y,
                   -sin(angle) * offset.x + cos(angle) * offset.y);
```

lecture 7's rotation by $-$angle: the offset, turned back

```
let q = turned / size;
let weight = exp(-0.5 * dot(q, q));
```

each part divided by its own size

dot: lecture 27's



Then, as in Round 1: `a = opacity * weight`, `c = mix(background, colour, a)`.

▶ **Run it in the Blob Lab:**
One blob, stretched and turned

You Try 3: a quarter turn

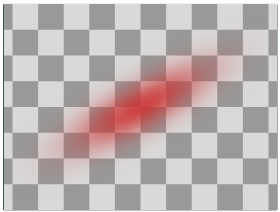
In the Blob Lab, choose **You Try 3**. Guess first, then change the code.

```
let size = vec2f(0.5, 0.15);
let angle = 0.5;
```

change this line

- **Turn the blob to 1.57, a quarter turn.**
- Guess first: which way does the blob point now?

now



after your change



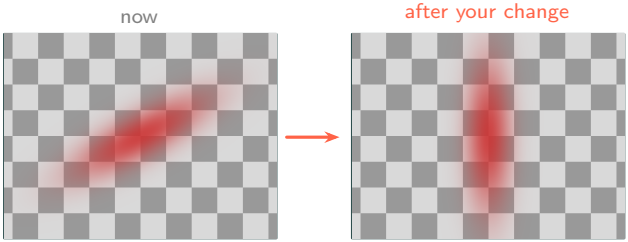
You Try 3: a quarter turn

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

```
let size = vec2f(0.5, 0.15);
let angle = 0.5;
```

change this line

answer: `let angle = 1.57;`



- **Turn the blob to 1.57, a quarter turn.**
- Guess first: which way does the blob point now?
- Straight up. With angle 0 it lies along x; 1.57 radians is 90 degrees.

You Try 4: both sizes the same

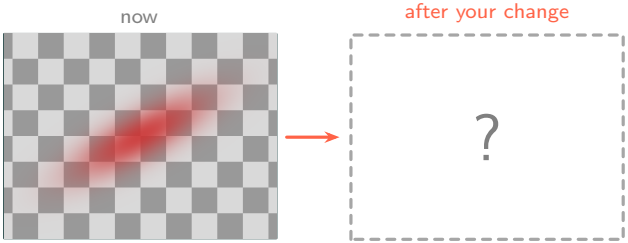
In the Blob Lab, choose **You Try 4**. Guess first, then change the code.

```
let size = vec2f(0.5, 0.15);
let angle = 0.5;
```

change this line

Then change the angle, and run it again.

- **Make both sizes 0.3. Then change the angle.**
- Guess first: what does the angle change now?



You Try 4: both sizes the same

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

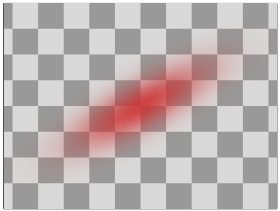
```
let size = vec2f(0.5, 0.15);
let angle = 0.5;
```

change this line

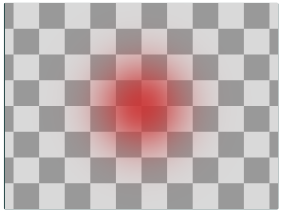
Then change the angle, and run it again.

answer: `let size = vec2f(0.3, 0.3);`

now



after your change



- **Make both sizes 0.3. Then change the angle.**
- Guess first: what does the angle change now?
- Nothing. A round blob looks the same at every angle: it is Round 1's blob again.

Round 3

Many blobs

each one over the picture so far

A blob is a struct; the flower is a list of nine

```
struct Blob {
    centre:  vec2f,
    size:  vec2f,
    angle:  f32,
    colour:  vec3f,
    opacity:  f32,
};
```

Round 2's numbers, kept together: a struct, like lecture 31's Corner (slide 6).

▶ **Run it in the Blob Lab:**
A flower of 9 blobs

A blob is a struct; the flower is a list of nine

```
struct Blob {
    centre:  vec2f,
    size:  vec2f,
    angle:  f32,
    colour:  vec3f,
    opacity:  f32,
};
```

Round 2's numbers, kept together: a struct, like lecture 31's Corner (slide 6).



```
const count = 9;      how many blobs: change it when you add one
var<private> blobs = array<Blob, count>(
    Blob(vec2f(0.0, -0.5), vec2f(0.03, 0.45), 0.0, vec3f(0.2, 0.55, 0.2), 0.95),   // the stem
    Blob(vec2f(-0.2, -0.5), vec2f(0.2, 0.06), -0.55, vec3f(0.3, 0.65, 0.25), 0.95), // a leaf
    Blob(vec2f(0.261, 0.22), vec2f(0.25, 0.11), 0.26, vec3f(0.9, 0.3, 0.45), 0.85), // petal 1
    ... petals 2 to 6 ...
    Blob(vec2f(0.0, 0.15), vec2f(0.12, 0.12), 0.0, vec3f(1.0, 0.8, 0.15), 1.0),   // the centre
);
```

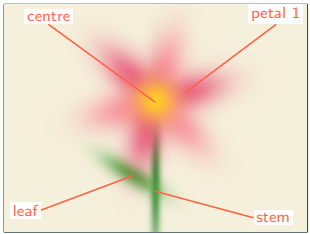
var¡private¿: a list that every run of fs_main can read.
The first blob in the list is drawn first: it is at the back.

▶ **Run it in the Blob Lab:**
A flower of 9 blobs

A blob is a struct; the flower is a list of nine

```
struct Blob {
    centre:  vec2f,
    size:  vec2f,
    angle:  f32,
    colour:  vec3f,
    opacity:  f32,
};
```

Round 2's numbers, kept together: a struct, like lecture 31's Corner (slide 6).




```
const count = 9;      how many blobs: change it when you add one
var<private> blobs = array<Blob, count>(
    Blob(vec2f(0.0, -0.5), vec2f(0.03, 0.45), 0.0, vec3f(0.2, 0.55, 0.2), 0.95),   // the stem
    Blob(vec2f(-0.2, -0.5), vec2f(0.2, 0.06), -0.55, vec3f(0.3, 0.65, 0.25), 0.95), // a leaf
    Blob(vec2f(0.261, 0.22), vec2f(0.25, 0.11), 0.26, vec3f(0.9, 0.3, 0.45), 0.85), // petal 1
    ... petals 2 to 6 ...
    Blob(vec2f(0.0, 0.15), vec2f(0.12, 0.12), 0.0, vec3f(1.0, 0.8, 0.15), 1.0),   // the centre
);
```

var¡private¿: a list that every run of fs_main can read.
The first blob in the list is drawn first: it is at the back.

▶ **Run it in the Blob Lab:**
A flower of 9 blobs

The loop: each blob over the picture so far

```
var c = background;        c: the picture so far
for (var i = 0; i < count; i++) {
    let b = blobs[i];
    let a = b.opacity * weight(vec2f(x, y), b);
    c = mix(c, b.colour, a);
}
```

weight: slide 19's lines, in a function. mix: slide 11's blend.

- Lecture 31 (slide 11) looped over 968 triangles. Here, over the blobs.

The loop: each blob over the picture so far

```
var c = background;        c: the picture so far
for (var i = 0; i < count; i++) {
    let b = blobs[i];
    let a = b.opacity * weight(vec2f(x, y), b);
    c = mix(c, b.colour, a);
}
```

weight: slide 19's lines, in a function. mix: slide 11's blend.

c, after the first k blobs of the list:



after 1

- Lecture 31 (slide 11) looped over 968 triangles. Here, over the blobs.
- Each blob is blended over c in turn: the last one ends on top.

The loop: each blob over the picture so far

```
var c = background;        c: the picture so far
for (var i = 0; i < count; i++) {
    let b = blobs[i];
    let a = b.opacity * weight(vec2f(x, y), b);
    c = mix(c, b.colour, a);
}
```

weight: slide 19's lines, in a function. mix: slide 11's blend.

c, after the first k blobs of the list:



after 1 after 2

- Lecture 31 (slide 11) looped over 968 triangles. Here, over the blobs.
- Each blob is blended over c in turn: the last one ends on top.

The loop: each blob over the picture so far

```
var c = background;        c: the picture so far
for (var i = 0; i < count; i++) {
    let b = blobs[i];
    let a = b.opacity * weight(vec2f(x, y), b);
    c = mix(c, b.colour, a);
}
```

weight: slide 19's lines, in a function. mix: slide 11's blend.

c, after the first k blobs of the list:



after 1 after 2 after 3

- Lecture 31 (slide 11) looped over 968 triangles. Here, over the blobs.
- Each blob is blended over c in turn: the last one ends on top.

The loop: each blob over the picture so far

```
var c = background;        c: the picture so far
for (var i = 0; i < count; i++) {
    let b = blobs[i];
    let a = b.opacity * weight(vec2f(x, y), b);
    c = mix(c, b.colour, a);
}
```

weight: slide 19's lines, in a function. mix: slide 11's blend.

c, after the first k blobs of the list:



after 1 after 2 after 3 after 5

- Lecture 31 (slide 11) looped over 968 triangles. Here, over the blobs.
- Each blob is blended over c in turn: the last one ends on top.

The loop: each blob over the picture so far

```
var c = background;        c: the picture so far
for (var i = 0; i < count; i++) {
    let b = blobs[i];
    let a = b.opacity * weight(vec2f(x, y), b);
    c = mix(c, b.colour, a);
}
```

weight: slide 19's lines, in a function. mix: slide 11's blend.

- Lecture 31 (slide 11) looped over 968 triangles. Here, over the blobs.
- Each blob is blended over c in turn: the last one ends on top.

c, after the first k blobs of the list:



after 1 after 2 after 3 after 5 after 8

The loop: each blob over the picture so far

```
var c = background;          c: the picture so far
for (var i = 0; i < count; i++) {
    let b = blobs[i];
    let a = b.opacity * weight(vec2f(x, y), b);
    c = mix(c, b.colour, a);
}
```

weight: slide 19's lines, in a function. mix: slide 11's blend.

- Lecture 31 (slide 11) looped over 968 triangles. Here, over the blobs.
- Each blob is blended over c in turn: the last one ends on top.

c, after the first k blobs of the list:



after 1 after 2 after 3 after 5 after 8 after 9

Worked: two blobs, two orders

Black background. A red blob with $a = 0.8$ and a blue blob with $a = 0.5$ at one pixel.
Try first: the pixel's colour after red then blue, and after blue then red.

red, then blue

▶ **Run it in the Blob Lab:**
Two blobs: the order

Worked: two blobs, two orders

Black background. A red blob with $a = 0.8$ and a blue blob with $a = 0.5$ at one pixel.
Try first: the pixel's colour after red then blue, and after blue then red.

red, then blue

after red: $(0.8,\ 0,\ 0)$

▶ **Run it in the Blob Lab:**
Two blobs: the order

Worked: two blobs, two orders

Black background. A red blob with $a = 0.8$ and a blue blob with $a = 0.5$ at one pixel.

Try first: the pixel's colour after red then blue, and after blue then red.

red, then blue

after red: $(0.8,\ 0,\ 0)$

after blue: $0.5\,(0.8, 0, 0) + 0.5\,(0, 0, 1) = (0.4,\ 0,\ 0.5)$





red, then blue

▶ **Run it in the Blob Lab:**
Two blobs: the order

Worked: two blobs, two orders

Black background. A red blob with $a = 0.8$ and a blue blob with $a = 0.5$ at one pixel.

Try first: the pixel's colour after red then blue, and after blue then red.

red, then blue

after red: $(0.8,\ 0,\ 0)$

after blue: $0.5\,(0.8, 0, 0) + 0.5\,(0, 0, 1) = (0.4,\ 0,\ 0.5)$





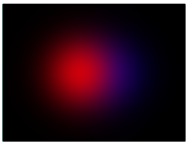
red, then blue

blue, then red

after blue: $(0,\ 0,\ 0.5)$

after red: $0.2\,(0, 0, 0.5) + 0.8\,(1, 0, 0) = (0.8,\ 0,\ 0.1)$





blue, then red

▶ **Run it in the Blob Lab:**
Two blobs: the order

Worked: two blobs, two orders

Black background. A red blob with $a = 0.8$ and a blue blob with $a = 0.5$ at one pixel.
Try first: the pixel's colour after red then blue, and after blue then red.

red, then blue

after red: $(0.8,\ 0,\ 0)$

after blue: $0.5\,(0.8, 0, 0) + 0.5\,(0, 0, 1) = (0.4,\ 0,\ 0.5)$





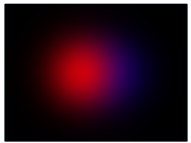
red, then blue

blue, then red

after blue: $(0,\ 0,\ 0.5)$

after red: $0.2\,(0, 0, 0.5) + 0.8\,(1, 0, 0) = (0.8,\ 0,\ 0.1)$





blue, then red

The same two blobs in the other order: a different colour.

The one blended last keeps the most of its own colour.

▶ **Run it in the Blob Lab:**
Two blobs: the order

A photo from 50, 500 and 5000 blobs

the photo, 160×120



57,600 numbers

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

▶ **Run it in the Blob Lab:**
A photo from 50 blobs

A photo from 50, 500 and 5000 blobs

the photo, 160 × 120



57,600 numbers

50 blobs



450 numbers

9 numbers a blob: a centre (2), two sizes, an angle, a colour (3), an opacity.

▶ **Run it in the Blob Lab:**
A photo from 50 blobs

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

A photo from 50, 500 and 5000 blobs

the photo, 160 × 120



57,600 numbers

50 blobs



450 numbers

500 blobs



4,500 numbers

9 numbers a blob: a centre (2), two sizes, an angle, a colour (3), an opacity.

▶ **Run it in the Blob Lab:**
A photo from 50 blobs

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

A photo from 50, 500 and 5000 blobs

the photo, 160 × 120	50 blobs	500 blobs	5000 blobs
			
57,600 numbers	450 numbers	4,500 numbers	45,000 numbers

9 numbers a blob: a centre (2), two sizes, an angle, a colour (3), an opacity.

▶ **Run it in the Blob Lab:**
A photo from 50 blobs

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

A photo from 50, 500 and 5000 blobs

the photo, 160 × 120



57,600 numbers

50 blobs



450 numbers

500 blobs



4,500 numbers

5000 blobs



45,000 numbers

9 numbers a blob: a centre (2), two sizes, an angle, a colour (3), an opacity.

Who chose these blobs? A program moved them, step by step, until the picture matched the photo. Lecture 38 shows how.

▶ **Run it in the Blob Lab:**
A photo from 50 blobs

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

You Try 5: the centre at the back

In the Blob Lab, choose **You Try 5**. Guess first, then change the code.

```
var<private> blobs = array<Blob, count>(
    Blob(vec2f(0.0, -0.5), vec2f(0.03, 0.45), 0.0, vec3f(0.2, 0.55, 0.2), 0.95),   // the stem
    ... a leaf, six petals ...
    Blob(vec2f(0.0, 0.15), vec2f(0.12, 0.12), 0.0, vec3f(1.0, 0.8, 0.15), 1.0),   // the centre
);
```

now



after your change



- **Move the centre's line to the top of the list.**
- Guess first: what happens to the yellow centre?

You Try 5: the centre at the back

In the Blob Lab, choose **You Try 5**. Guess first, then change the code.

```
var<private> blobs = array<Blob, count>(
    Blob(vec2f(0.0, -0.5), vec2f(0.03, 0.45), 0.0, vec3f(0.2, 0.55, 0.2), 0.95),   // the stem
    ... a leaf, six petals ...
    Blob(vec2f(0.0, 0.15), vec2f(0.12, 0.12), 0.0, vec3f(1.0, 0.8, 0.15), 1.0),   // the centre
);
```

answer: the centre's line goes first, above the stem's

now



after your change



- **Move the centre's line to the top of the list.**
- Guess first: what happens to the yellow centre?
- Blended first, it is at the back: the petals cover most of it.

You Try 6: a second leaf

In the Blob Lab, choose **You Try 6**. Guess first, then change the code.

```
const count = 9;
var<private> blobs = array<Blob, count>(
    Blob(vec2f(0.0, -0.5), vec2f(0.03, 0.45), 0.0, vec3f(0.2, 0.55, 0.2), 0.95),   // the stem
    Blob(vec2f(-0.2, -0.5), vec2f(0.2, 0.06), -0.55, vec3f(0.3, 0.65, 0.25), 0.95), // a leaf
```

now



make this



- **Add a second leaf, after the first.**
- Guess first: in front of the stem, or behind it?

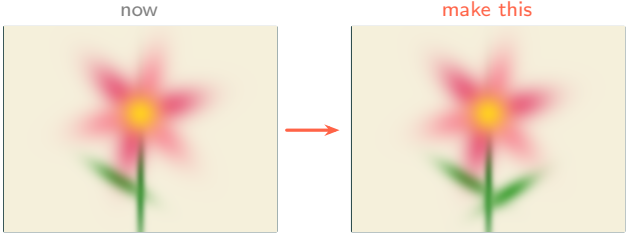
You Try 6: a second leaf

In the Blob Lab, choose **You Try 6**. Guess first, then change the code.

```
const count = 9;
var<private> blobs = array<Blob, count>(
    Blob(vec2f(0.0, -0.5), vec2f(0.03, 0.45), 0.0, vec3f(0.2, 0.55, 0.2), 0.95),   // the stem
    Blob(vec2f(-0.2, -0.5), vec2f(0.2, 0.06), -0.55, vec3f(0.3, 0.65, 0.25), 0.95), // a leaf
```

answer:

```
const count = 10;
Blob(vec2f(0.2, -0.62), vec2f(0.2, 0.06), 0.55, vec3f(0.3, 0.65, 0.25), 0.95), // after the leaf
```



- **Add a second leaf, after the first.**
- Guess first: in front of the stem, or behind it?
- In front: it comes after the stem in the list. count becomes 10.

What blobs get wrong: the order

centre last



centre first



the same nine blobs, in two orders: two different pictures

Swap two blobs and the picture changes.

What blobs get wrong: the order

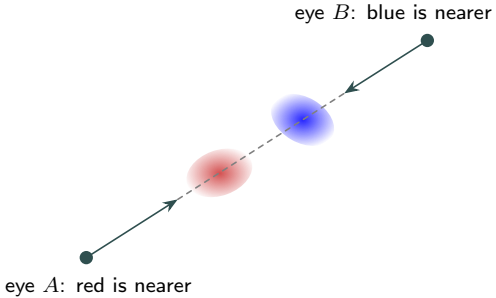
centre last



centre first



the same nine blobs, in two orders: two different pictures

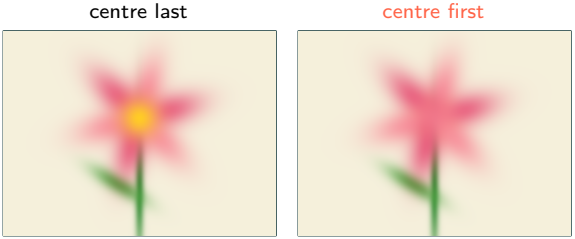


seen from above

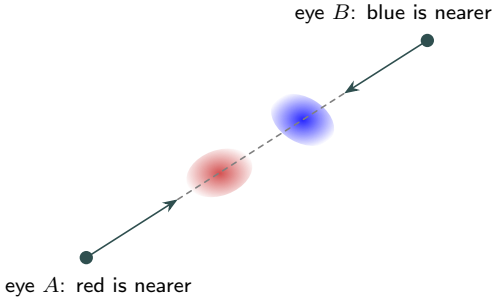
Swap two blobs and the picture changes.

In 3D, which blob is nearer depends on where the eye is.

What blobs get wrong: the order



the same nine blobs, in two orders: two different pictures



seen from above

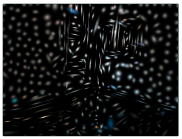
Swap two blobs and the picture changes.

In 3D, which blob is nearer depends on where the eye is.

So the right order changes every time the camera moves.

Lecture 37 sorts the blobs for every frame.

Now make your own picture



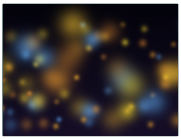
The photo, 500 blobs

`move the mouse: shrink`



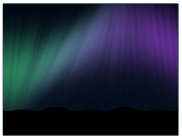
A flower that opens

`12 petals in a loop`



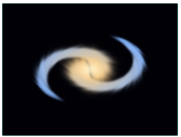
City lights at night

`80 round blobs, hash`



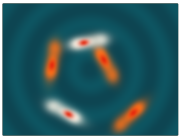
The northern lights

`60 thin, faint blobs`



A spiral galaxy

`600 blobs on two arms`



A koi pond

`4 blobs a fish, u.time`

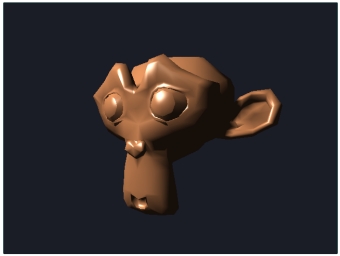
All six are in the Blob Lab, under **Ideas**. Change one. Mix two. Show your neighbour.

Summary

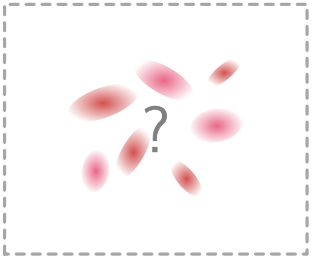
1	One blob	weight $= \exp(-d^2/(2\,\text{size}^2))$ 1 at the centre, 0.61 at one size, 0.01 at three; no edge
2	Opacity	$a =$ opacity $\times$ weight; `c = mix(c, colour, a)` lecture 3's blend: a parts blob, $1 - a$ parts what is there
3	Stretch and turn	turn the offset back, divide by the two sizes then the same weight: $\exp(-\frac{1}{2}\, q \cdot q)$
4	Many blobs	a list, each blended over the picture so far 9 numbers a blob; 5000 blobs rebuild a photo
5	What it gets wrong	the order matters swap two: a different picture; in 3D it changes with the eye

Next lecture: a blob in 3D

lecture 29: Suzanne's triangles



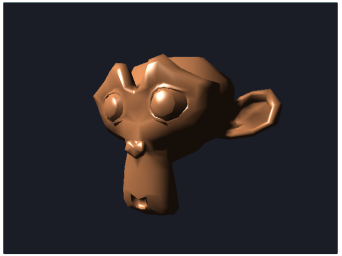
lecture 36: each corner a blob



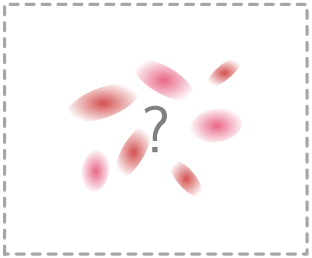
- A blob in 3D: a centre, three sizes, a turn, a colour, an opacity.

Next lecture: a blob in 3D

lecture 29: Suzanne's triangles



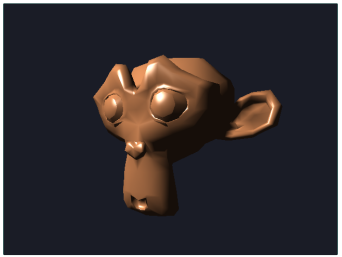
lecture 36: each corner a blob



- A blob in 3D: a centre, three sizes, a turn, a colour, an opacity.
- On the screen it is a 2D blob: today's ellipse.

Next lecture: a blob in 3D

lecture 29: Suzanne's triangles



lecture 36: each corner a blob



- A blob in 3D: a centre, three sizes, a turn, a colour, an opacity.
- On the screen it is a 2D blob: today's ellipse.
- Drawn in the file's order, Suzanne's far side will show through her face. Why?

Exit Ticket

Today's question *(2 minutes)*

(a) A round blob has size 0.2 and opacity 0.5. A pixel is 0.2 from its centre. What is the weight there? What is a?

(b) On a black background, a red blob with $a = 0.5$ is blended, then a green blob with $a = 0.5$. What colour is the pixel?

CS 453 — Exit Ticket #35

Name: _____

Roll #: _____ Date: _____

Answer:

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