# FEDAY2019

## 第5届 FEDAY

2019.09.21 / 成都

主办方 w3ctech 前端周刊

赞助商 UCLOUD TURING 图灵教育 Broadview www.broadview.com.cn 又拍云 APICloud CCIMAGE 欢迎加入

支持社区 V2EX 掘金 segmentfault 思否 云社区 FCC freecodecamp 前端早读课 码云 gitee.com 开源中国 oschina.net 语雀 yuque.com 优设 uisdc.com

# 你不知道的GPU

## —— 前端、图形系统与数据可视化

月影　2019.09

## 吴亮（月影）

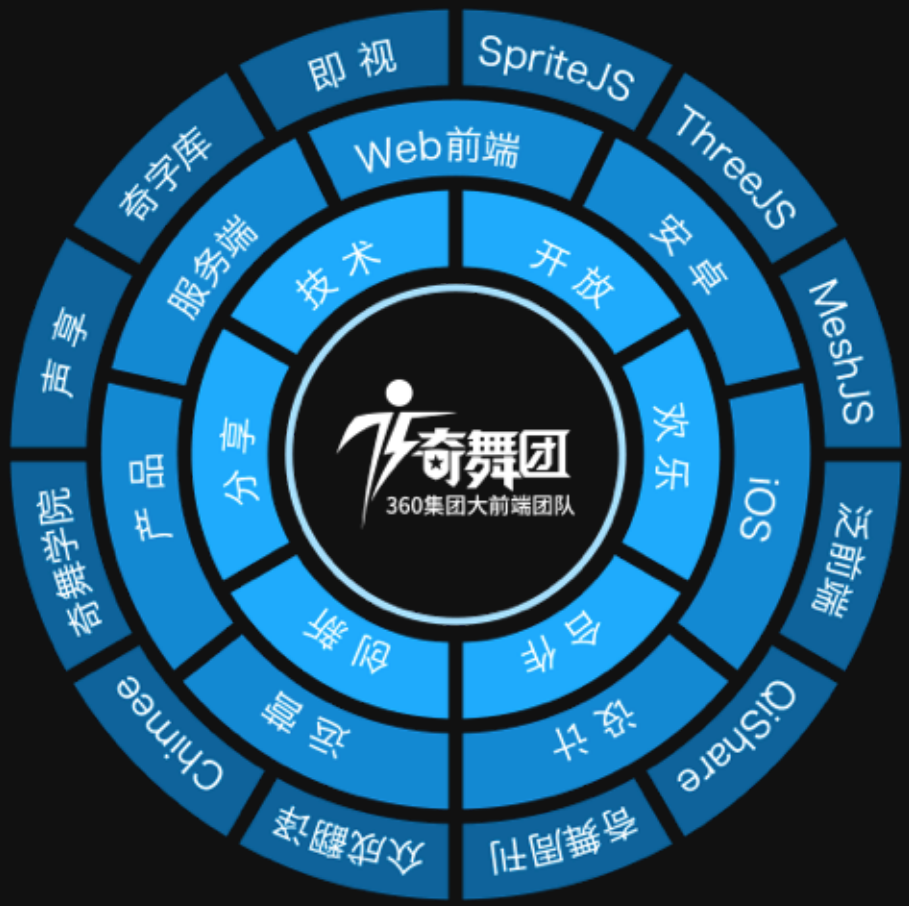
@十年踪迹

github.com/akira-cn

360高级技术总监 | Web前端主任架构师

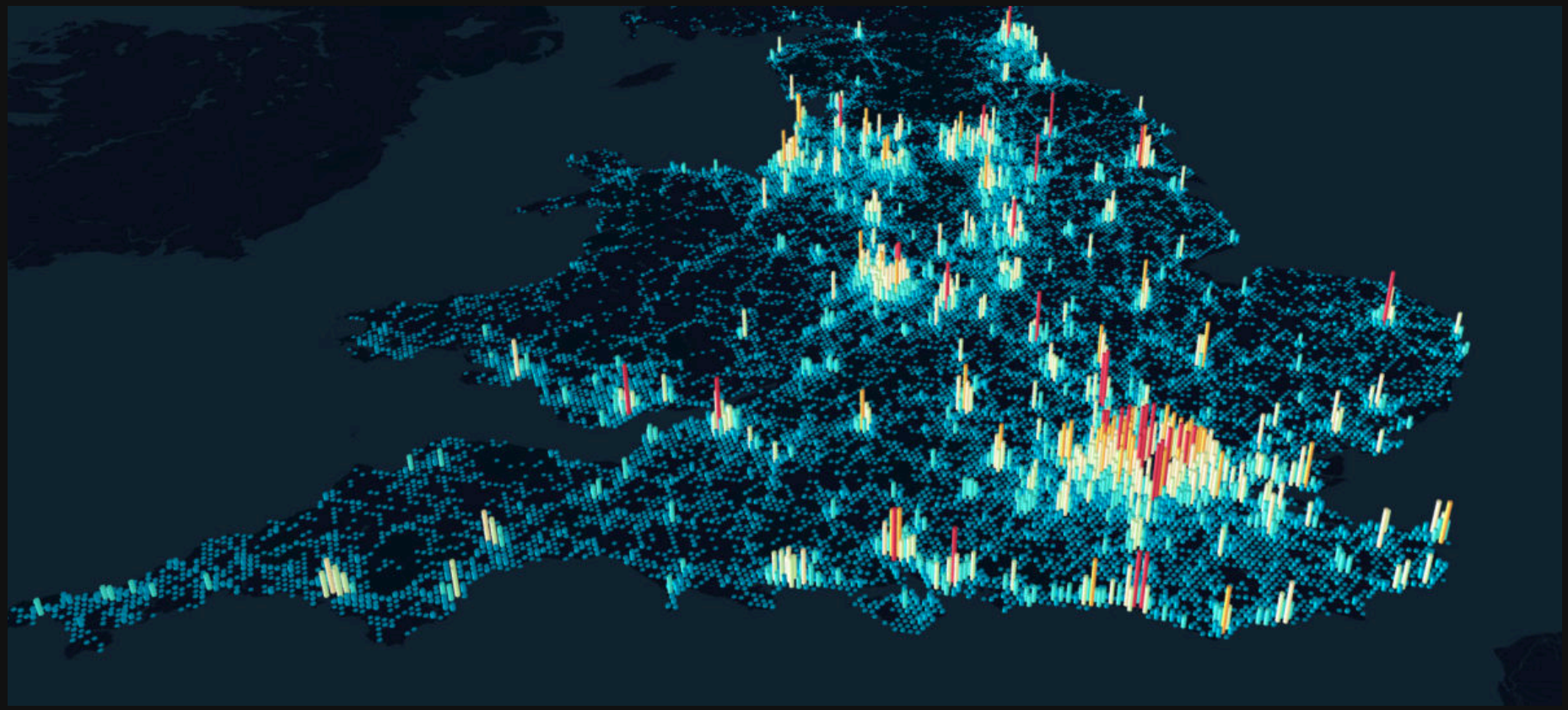
奇舞团 | Qtest | TC委员 | w3ctech | SpriteJS

《JavaScript王者归来》 | 《程序员成长的烦恼》




奇舞团（75team.com）是360集团最大的大前端团队，TC39和W3C会员，拥有Web前端、服务端、Android、iOS、设计、产品、运营等岗位人员80余名，旗下的开源框架和技术品牌有SpriteJS、ThinkJS、MeshJS、Chimee、QiShare、声享、即视、奇字库、众成翻译、奇舞学院、奇舞周刊、泛前端分享等。

# Data Visualization



# Why a new engine for D.V.?

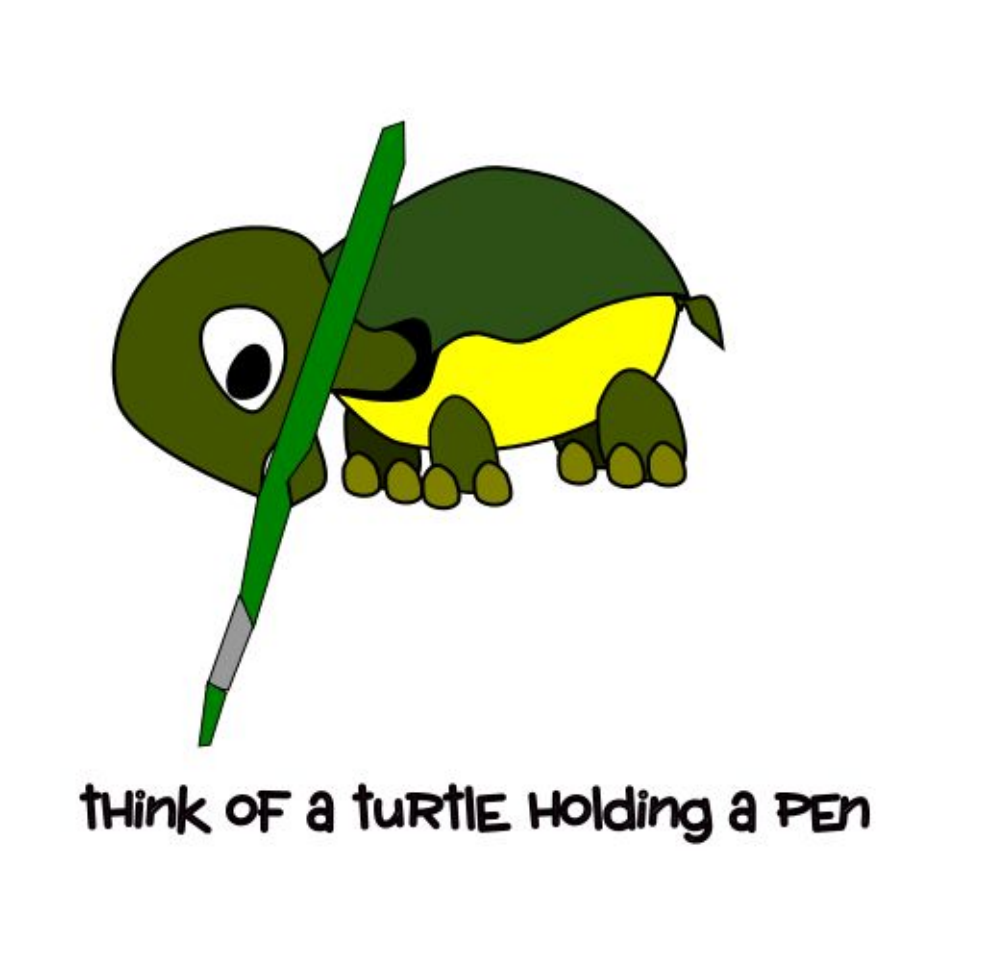
Mass rendering of large-scale data.

Efficient Utilization of GPU.

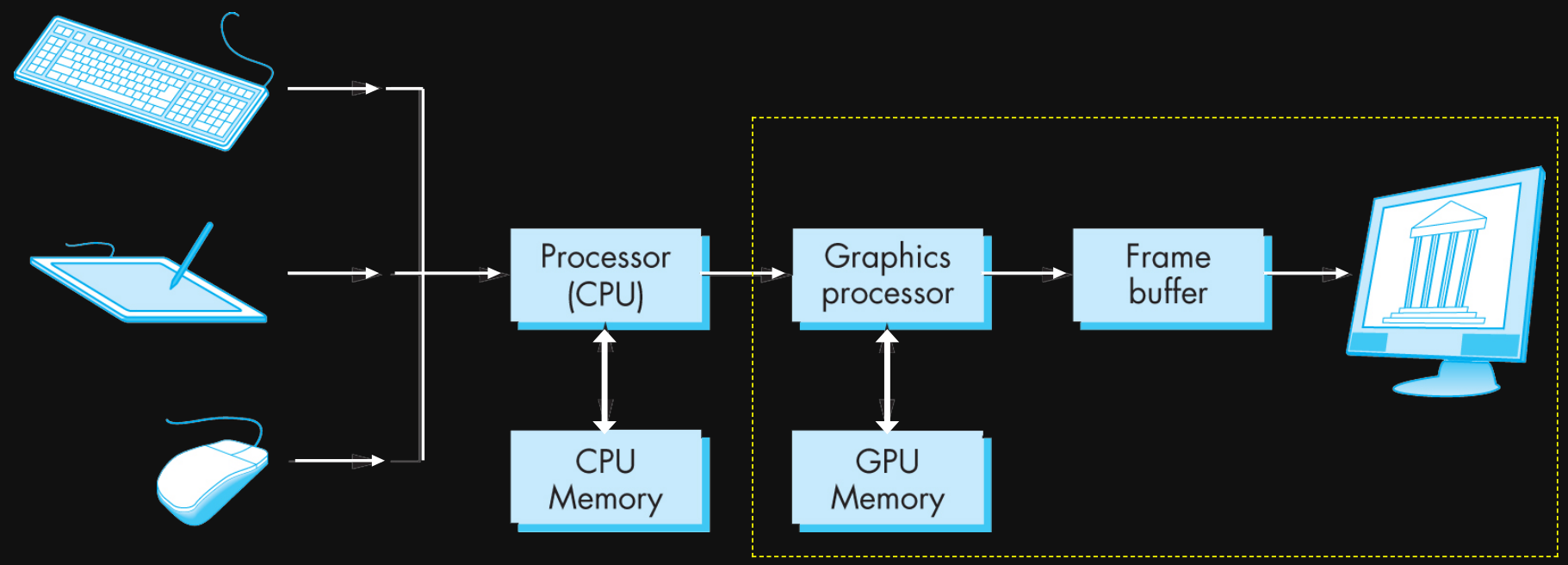
Easy to Use.

GPU≠WebGL≠3D

# Simple (Turtle) Graphics



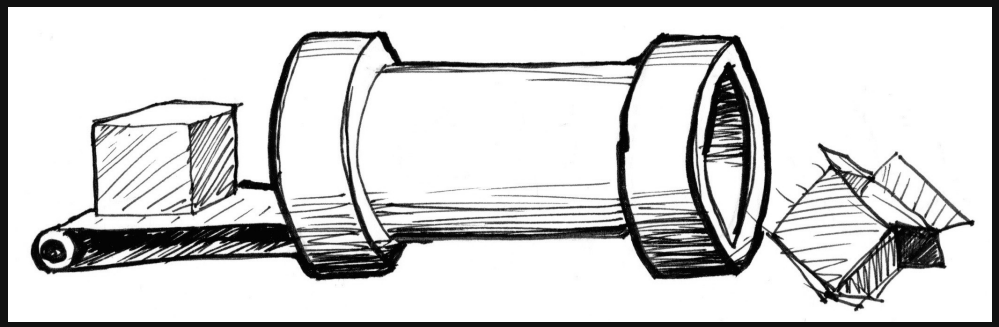
# Modern Graphics System

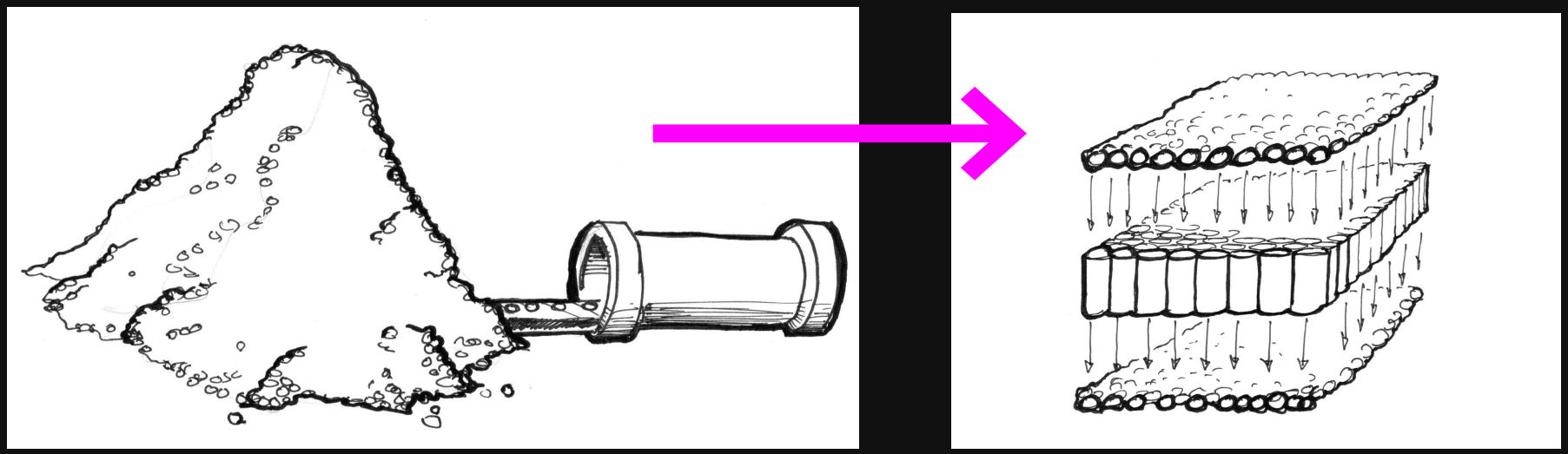


# Graphics pipeline (The GPU)

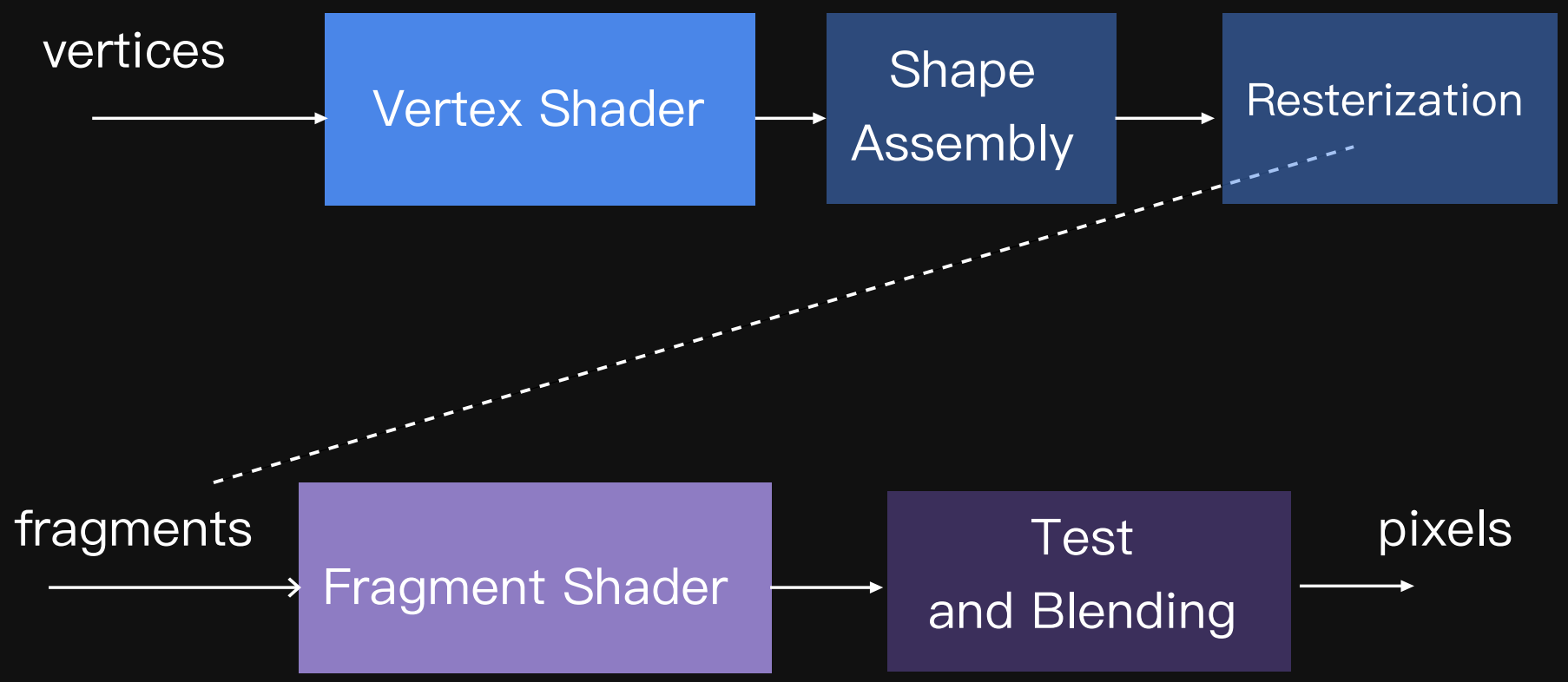
Data → Processor → Frame buffer → Pixels

# CPU vs. GPU

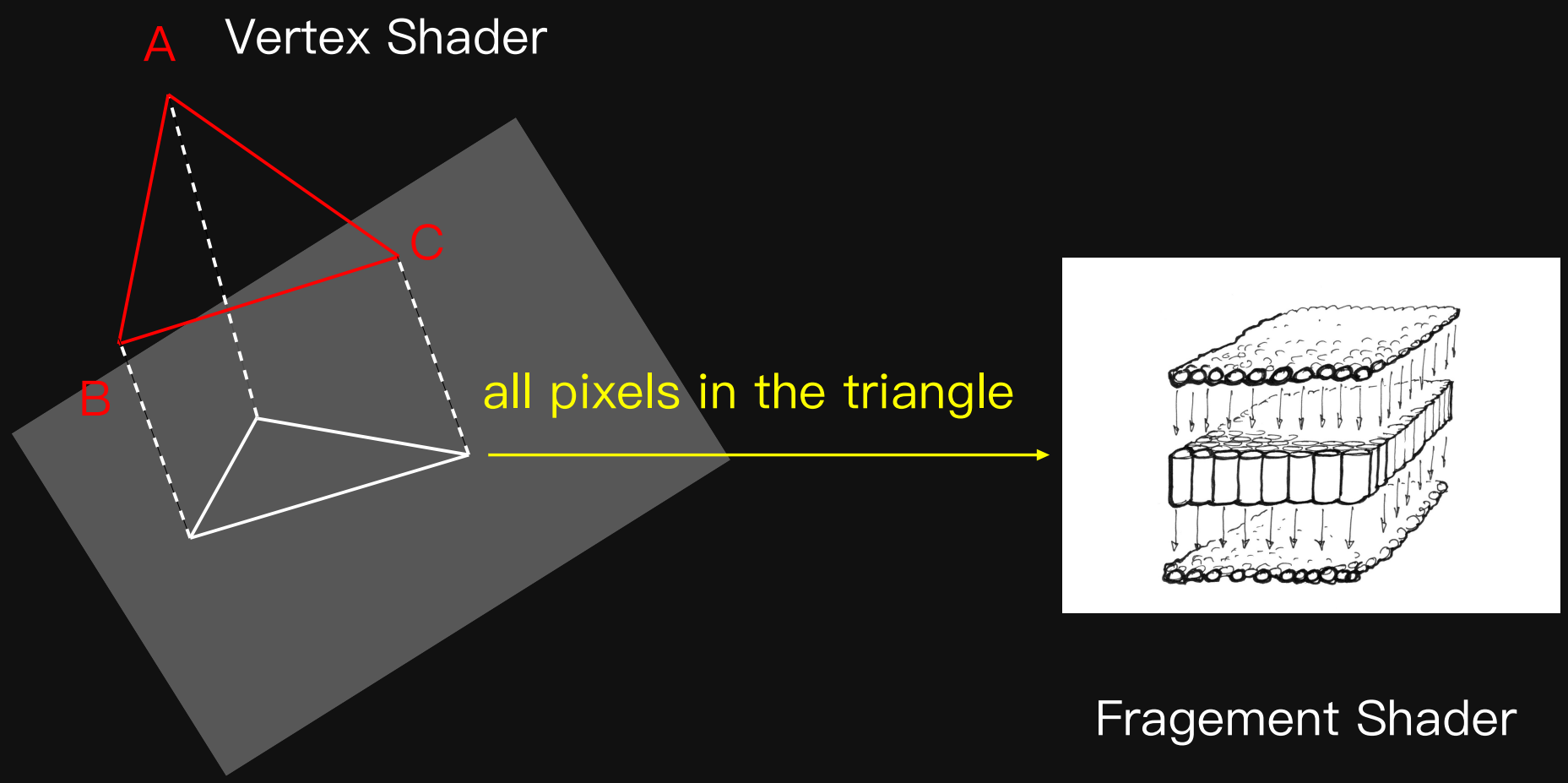




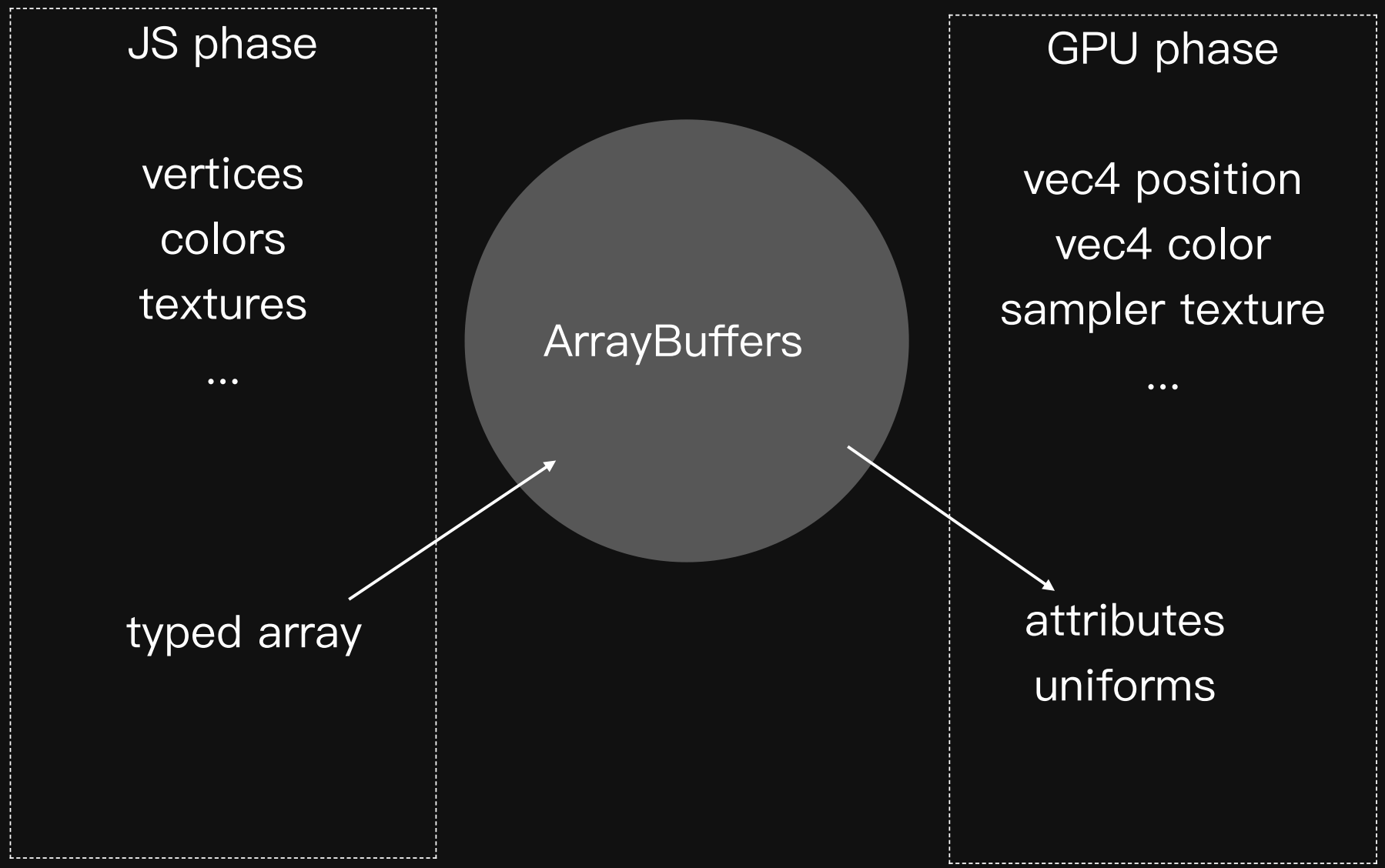
# WebGL pipeline



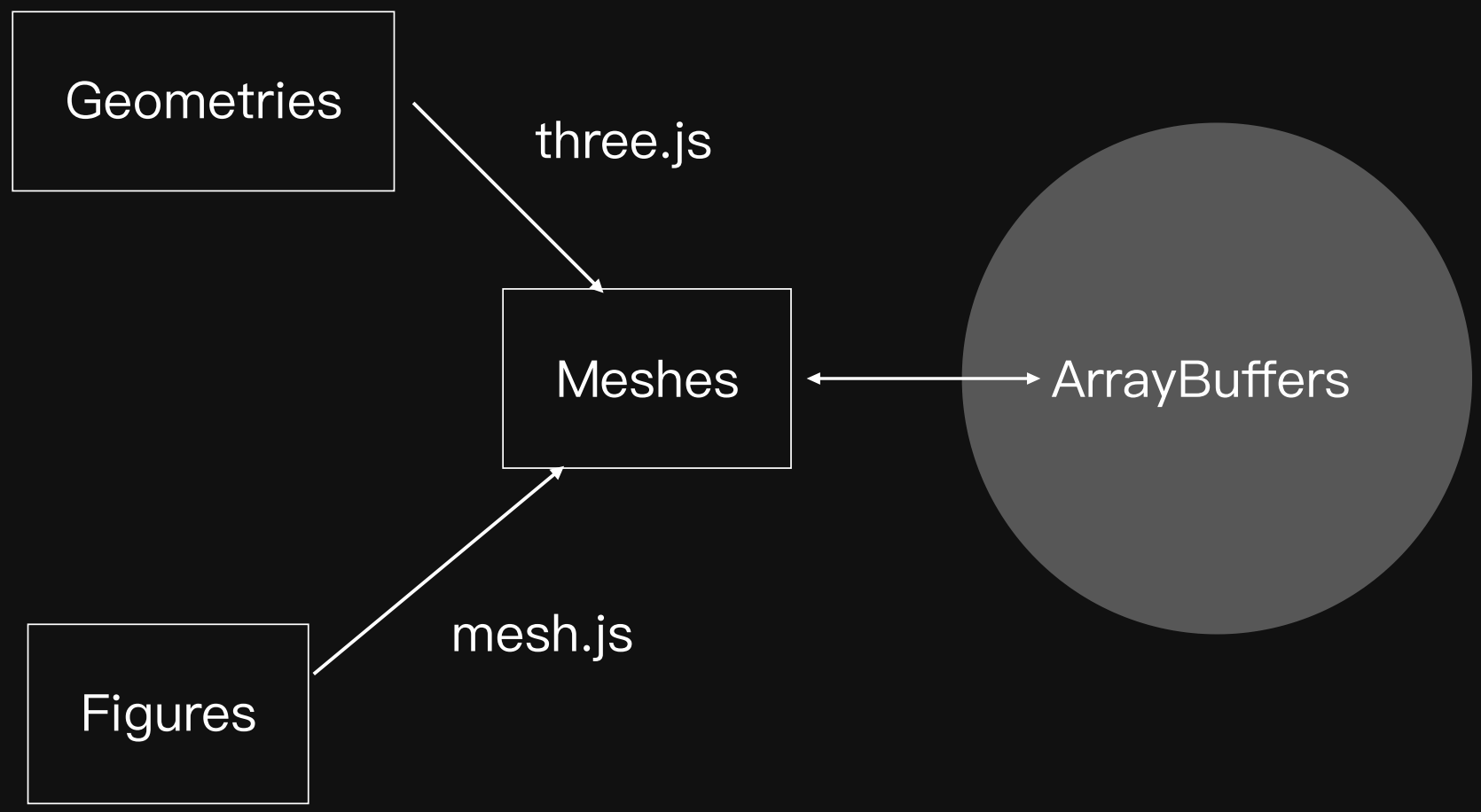
# Parallel processing



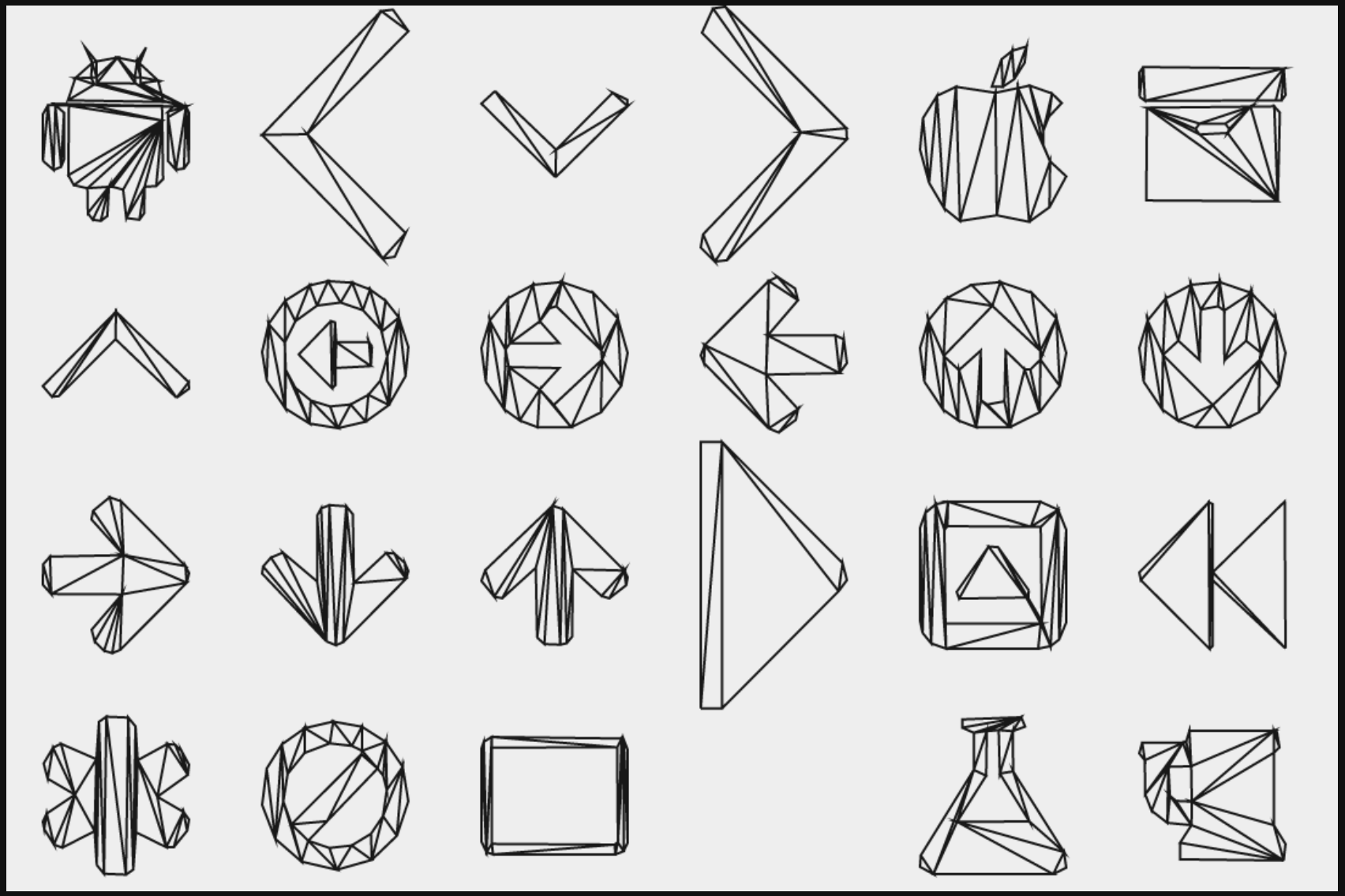
# JavaScript to GPU



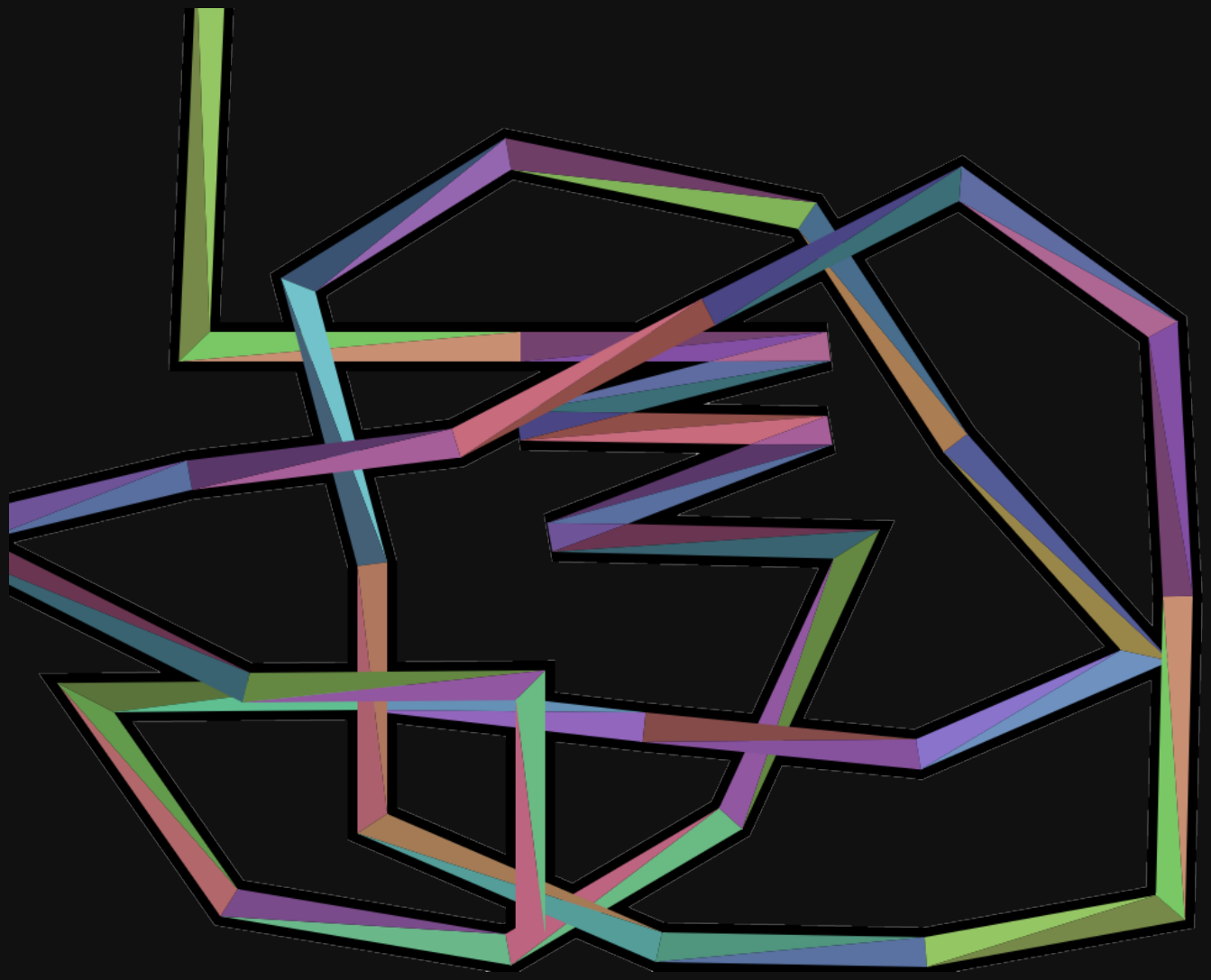
# Create Object Model



# Path Contours & Triangulations



# Polyline Strokes

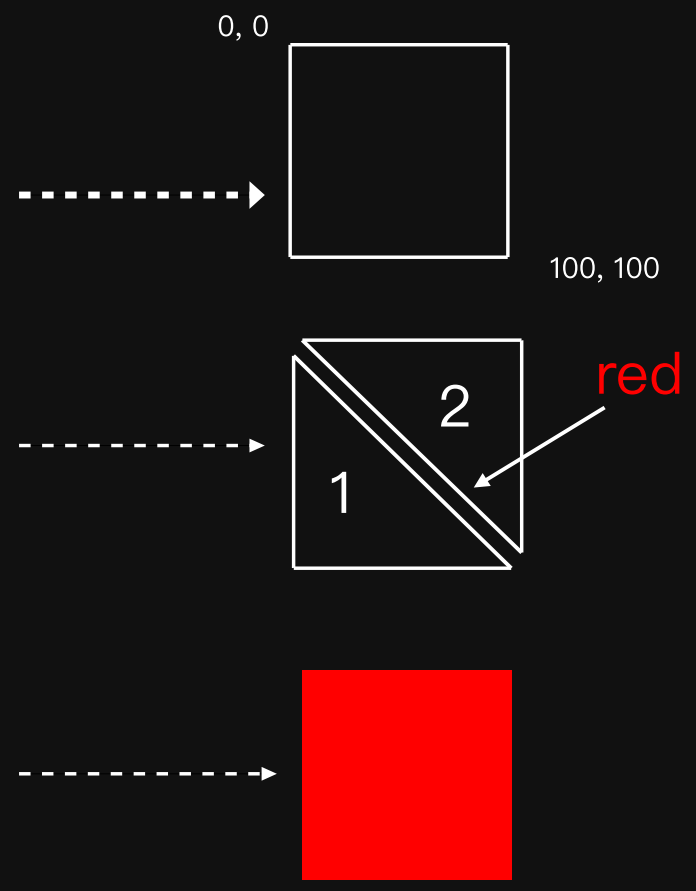


# Mesh.js

```
const figure = new Figure2D();
figure.addPath('M0 0L100 0L100 100L0 100z');
```

```
const mesh = new Mesh2D(figure, resolution);
mesh.setFill({color: [1, 0, 0, 1]});
```

```
const renderer = new Renderer(canvas);
renderer.drawMeshes([mesh]);
```



# Draw meshes

https://codepen.io/akira-cn/embed/WNezZqw/?height=300&th...

# Mesh API vs. Canvas API

```
const canvas = document.querySelector('canvas');

const figure = new Figure2D();
figure.rect(0, 0, 100, 100);

const mesh = new Mesh2D(figure, canvas);
mesh.setFill({color: [1, 0, 0, 1]});

const renderer = new Renderer(canvas);
renderer.drawMeshes([mesh]);
```

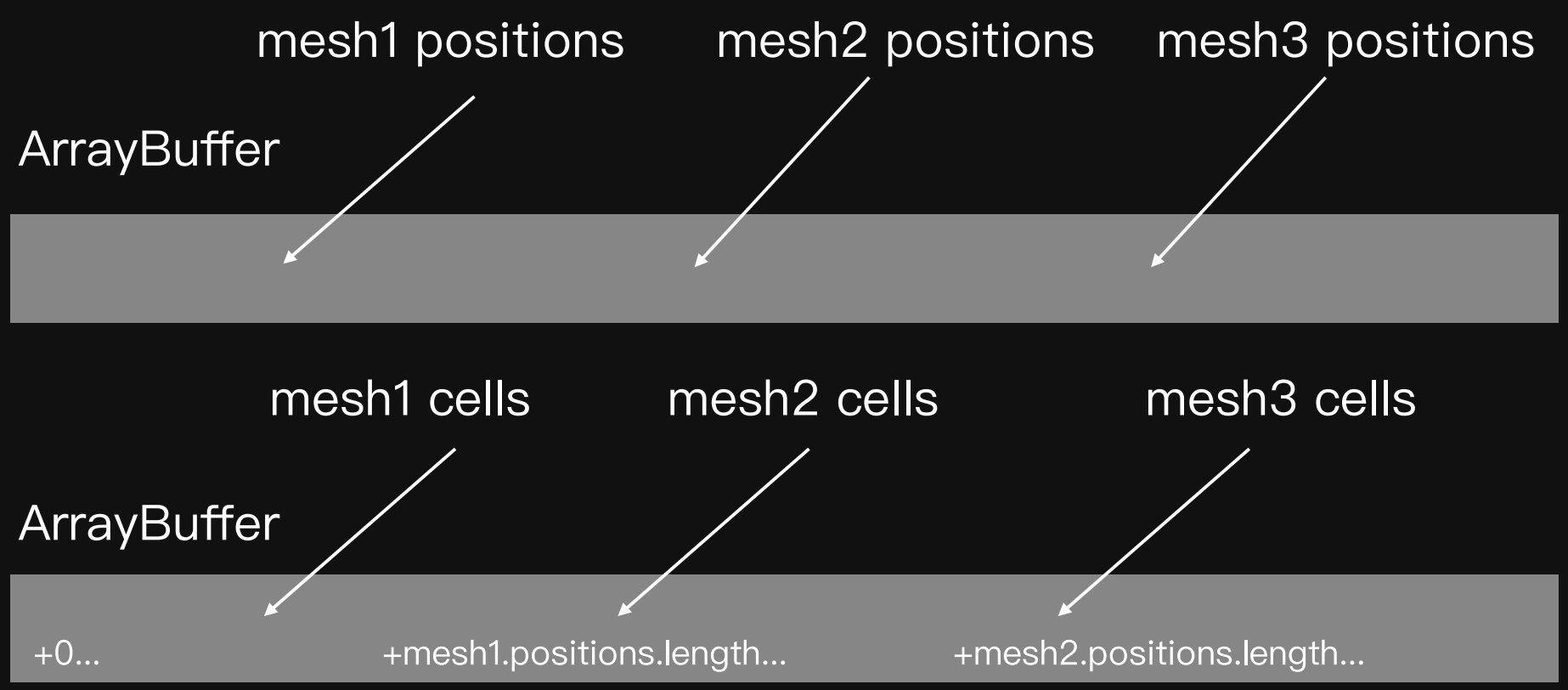
```
const canvas = document.querySelector('canvas');

const context = canvas.getContext('2d');

context.rect(0, 0, 100, 100);

context.fillStyle = 'red';

context.fill();
```

# Mesh Compression

```
// mesh1 :

{
  positions: [[x11, y11], [x12, y12], [x13, y13]],
  cells: [[0, 1], [1, 2], [2, 0]],
  attributes: {
    a_color: [[1, 0, 0], [1, 0, 0], [1, 0, 0]],
  },
  uniforms: {
    ...
  }
}

// mesh2 :

{
  positions: [[x21, y21], [x22, y22], [x23, y23]],
  cells: [[0, 1], [1, 2], [2, 0]],
  attributes: {
    a_color: [[0, 1, 0], [0, 1, 0], [0, 1, 0]],
  },
  uniforms: {
    ...
  }
}
```

```
// compressed: mesh1 + mesh2

{
  positions: Float32Array(x1, y1, x2, y2,
    x3, y3, x1, y1,
    x2, y2, x3, y3),
  cells: Uint16Array(0, 1, 1, 2, 2, 0,
    3, 4, 4, 5, 5, 3),
  attributes: {
    a_color: Uint8Array(
      255, 0, 0, 255, 0, 0, 255, 0, 0,
      0, 255, 0, 0, 255, 0, 0, 255, 0,
    ),
  },
  uniforms: {
    ...
  }
}
```

# Mesh Compression



# Mesh Compression

https://codepen.io/akira-cn/embed/gOYeGyV/?height=300&th...

# Transforms

https://codepen.io/akira-cn/embed/OJLvxdL/?height=300&the...

# Shader optimization

```
uniform float u_colorSteps[40];
uniform float u_radialGradientVector[6];

void main() {
  ...

  if(u_radialGradientVector[2] > 0.0 || u_radialGradientVector[5] > 0.0) {
    radial_gradient(color, u_radialGradientVector, u_colorSteps);
  }
}
```

# Shader optimization

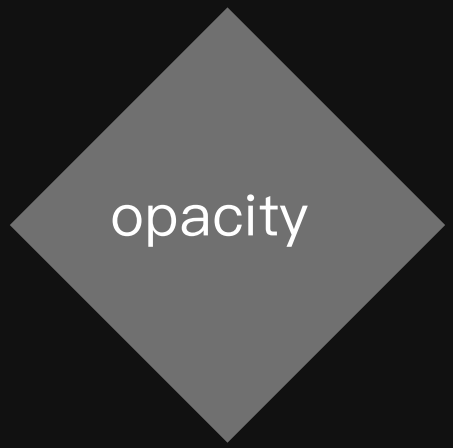
```
uniform float u_colorSteps[40];
uniform float u_radialGradientVector[6];

void main() {
  ...

  if(u_radialGradientVector[2] > 0.0 || u_radial[illegible]die[illegible][5] > 0.0) {
    radial_gradient(color, u_radialGradientV[illegible]or, u_colorSteps);
  }
}
```

# Shader optimization

```
uniform float u_colorSteps[40];
uniform float u_radialGradientVector[6];

void main() {
  ...

  if(u_radialGradientVector[2] > 0.0 || u_radial[illegible] > 0.0) {
    radial_gradient(color, u_radialGradientV[illegible]or, u_colorSteps);
  }
}
```

```
#ifdef GRADIENT
uniform float u_colorSteps[40];
uniform float u_radialGradientVector[6];
#endif

void main() {
  ...
#ifdef GRADIENT
  if (u_radialGradientVector[2] > 0.0 || u_radialGradientVector[5] > 0.0) {
    radial_gradient(color, u_radialGradientVector, u_colorSteps);
  }
#endif
}
```

# Blending





```
get enableBlend() {
  if(this[_blend] === true || this[_blend] === false) return this[_blend];
  return this[_uniforms].u_opacity < 1.0
    || this[_strokeColor] != null && this[_strokeColor][3] < 1.0
    || this[_fillColor] != null && this[_fillColor][3] < 1.0
    || this[_uniforms].u_colorMatrix != null && this[_uniforms].u_colorMatrix[18] < 1.0;
}
```

# Offscreen Canvas + Worker

```
const worker = new Worker('./worker.js');
const canvas = document.querySelector('canvas');
const offscreenCanvas = canvas.transferControlToOffscreen();
worker.postMessage({
  type: 'created',
  canvas: offscreenCanvas
}, [offscreenCanvas]);
```

```
importScripts('mesh.js');

self.addEventListener('message', (evt) => {
  const canvas = evt.data.canvas;
  const renderer = new Renderer(canvas);
  ...
}
```

# Offscreen Canvas + Worker

https://meshjs.org/demo/#/worker/flappy_bird

# User Program

https://codepen.io/akira-cn/embed/GRKxMBe/?height=300&t...

# forward and backward

Renderer Context

| | |
|---|---|
| webgpu | working |
| webgl2 | done |
| webgl | done |
| canvas2d | partial |

# Q & A

https://meshjs.org – A graphics system born for visualization.