

От `<div>` к WebGL



★ Star

27.6k

На примерах **PixiJS**

1. Убираем <div>

2.

```
let canvas = document.createElement('canvas');
canvas.width = appWidth;
canvas.height = appHeight;
let gl = canvas.getContext('webgl',
    {antialias: false, alpha: false, premultipliedAlpha: false, stencil: true});

document.body.appendChild(canvas);
```

3. PROFIT!

Меня зовут Иван

Закончил МехМат МГУ, много участвовал в олимпиадах по программированию.

С 2015 года занимаюсь WebGL, PixiJS

В 2012—2013, ещё до agar.io сделал игру <http://gameofbombs.com> на canvas-2d, бомбер на 2000 человек на одной карте.




Еще о себе

Работаю некромантом в <https://crazypanda.ru>, делаю конвертер Flash-графики на html5.

4 года помогаю на форумах и в чатах по разработке на Canvas 2d/WebGL.

Знаю много чужих кейсов, консультирую проекты с тяжёлыми случаями.



Необходимый
минимум
знаний по WebGL



Минимальный WebGL в рixi

```
1 const app = new PIXI.Application();
2 document.body.appendChild(app.view);
3
4 const geometry = new PIXI.Geometry()
5   .addAttribute('aVertexPosition', // the attribute name
6     [-100, -50, // x, y
7       100, -50, // x, y
8       0.0, 100.0], // x, y
9     2) // the size of the attribute
10
11   .addAttribute('aColor', // the attribute name
12     [1, 0, 0, // r, g, b
13      0, 1, 0, // r, g, b
14      0, 0, 1], // r, g, b
15     3); // the size of the attribute
16
17 const shader = PIXI.Shader.from(`
18
19   precision mediump float;
20   attribute vec2 aVertexPosition;
21   attribute vec3 aColor;
22
23   uniform mat3 translationMatrix;
24   uniform mat3 projectionMatrix;
25
26   varying vec3 vColor;
27
28   void main() {
29     vColor = aColor;
30     gl_Position = vec4((projectionMatrix * translationMatrix *
31       vec3(aVertexPosition, 1.0)).xy, 0.0, 1.0);
32   }`);
```

hello world

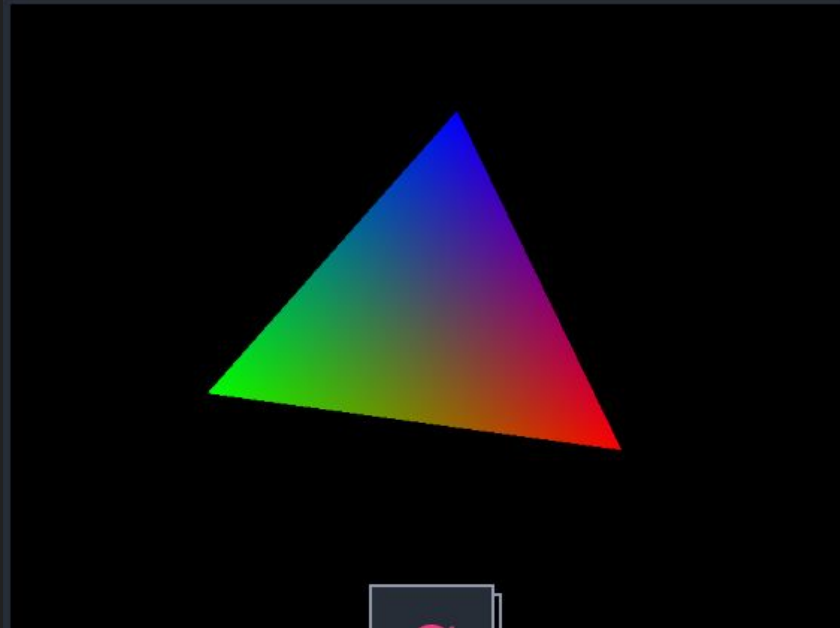
коорды вершины

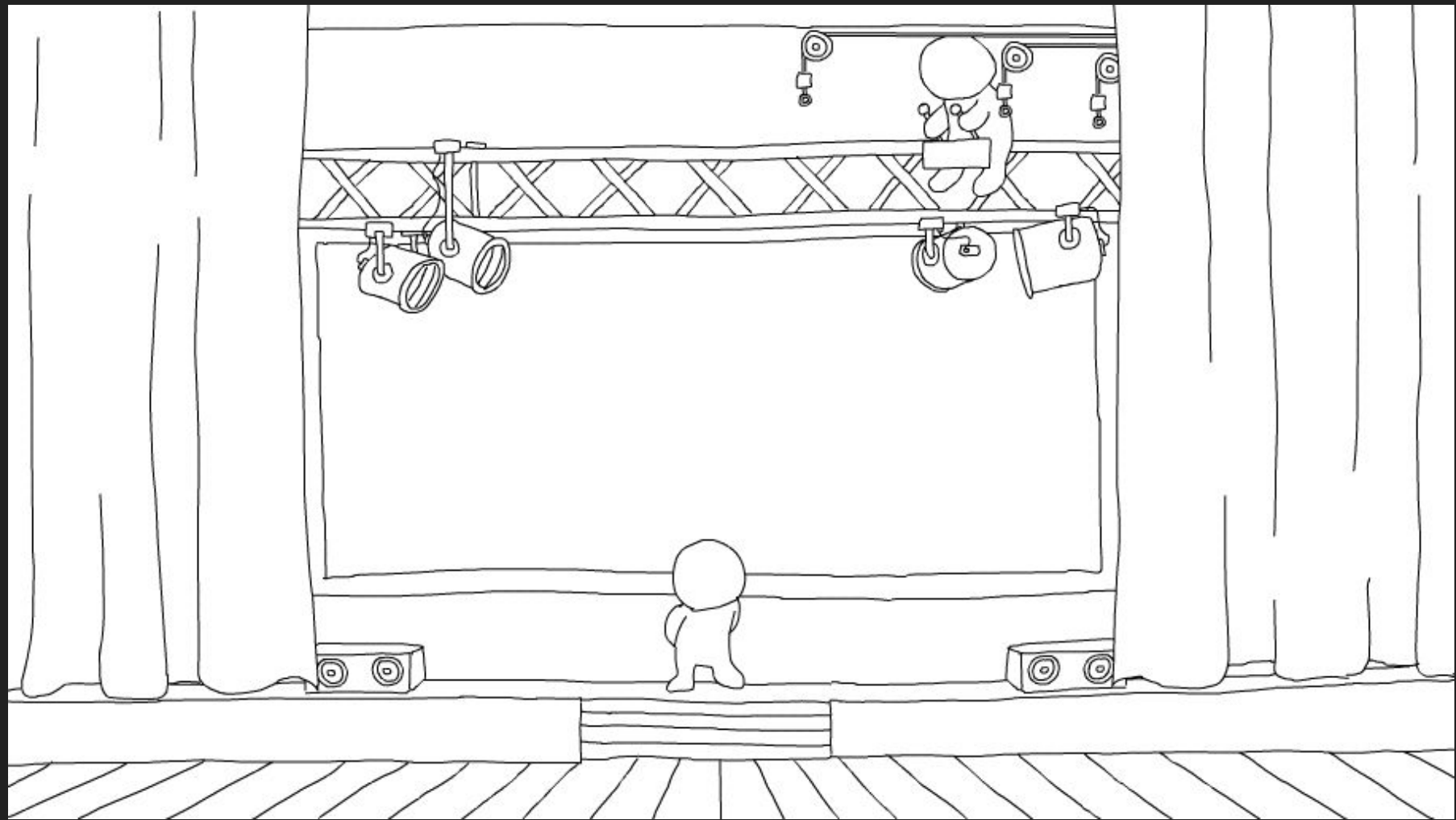
цветик, по 3

камера position, scale

сюда приходит координата

Colored Triangle





Rendering in WebGL



Для чего используют в продакшне





Это не конь!

HIERARCHY

- Root
 - camera
 - inside
 - light
 - light-interior
 - floor
 - BMW_i8
 - materials
 - car
 - body
 - cap-petrol
 - cap-electricity
 - wheels
 - interior
 - body
 - armchair
 - steering-wheel
 - screen-middle
 - screen-audio
 - screen-panel
 - labels
 - steerParent
 - doors
 - left
 - right
 - glass
 - blobs
 - augmented
 - ground
 - pivot
 - camera-out
 - _testSteer



ASSETS

Search: How do L... | Library

- scripts
- car
 - materials
 - leather-second-door
 - leather-second-interior
 - leather-third-interior
 - metal-shine-body
 - metal-shine-interior
 - metal-shine-wheel
 - mirror-door
 - panel-top-interior
 - plastic-black-body
 - plastic-blue-body
 - plastic-blue-interior
 - plastic-blue-wheel
 - plastic-dark-door
 - plastic-dark-interior
 - models
 - textures
 - ground
 - other
 - skyboxes

MATERIAL

ID: 1974175
Name: screen
Tags: Add Tag +
Runtime: yes
Type: material
Preload:
Size: 0 B
Source: none
Download

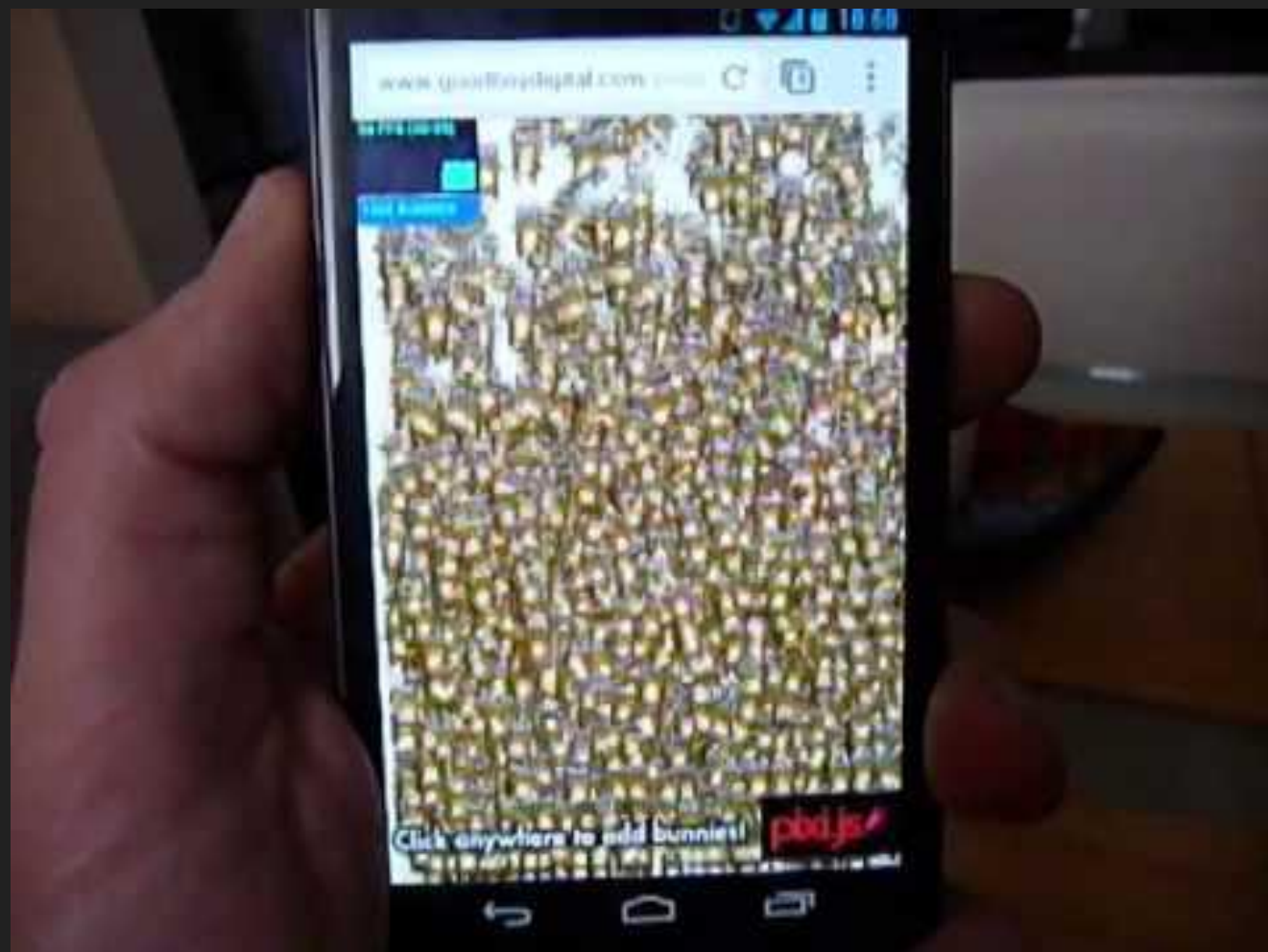
MATERIAL

Shading: Physical

- OFFSET & TILING
- AMBIENT
- DIFFUSE
- SPECULAR
- EMISSIVE
 - Screen
 - UV0
 - RGB

Vertex Color





Клуб 100k кроликов — это круто!

Разные элементы

- Кролики в игре
- Котировки на бирже
- Здания, точки на карте

Рецепт

- лоу-левел
(Sprite/Graphics/Tilemap)
- хай-левел от кодера

Комбинаций много, решений много,

Можно делать доклад на каждой конфе.





PixiJS

• Rendering

• Animating

• AdjustmentFilter

• AdvancedBloomFilter

• AlphaFilter (pixi.js)

• AsciiFilter

• BevelFilter

• BloomFilter

enabled

blur 1

blurX 1

blurY 1

• BlurFilter (pixi.js)



WebGL не нужен





```
function horizontal(sx, sy, len) {  
  for (var i=0;i<len;i++) {  
    curve(sx, 0, sx, sy);  
    curve(0, sy, sx, sy);  
    curve(sx, 0, sx, -sy);  
    curve(0, -sy, sx, -sy);  
  }  
}
```



Кривые руки рисуют кривые

CSS/SVG и так многое умеет.



- <https://keithclark.co.uk/articles/creating-3d-worlds-with-html-and-css/> - на ЧИСТОМ CSS
- ThreeJS имеет SVGRenderer, частичная поддержка материалов.
- canvas-2d более ограничен: нет перспективы

</вступление>



Альфа-блендинг

Research #2

Математик отвечает:

решений наверное много, но вот я что-то подобрал.

Можно складывать два соседних объекта по формуле

$$R = S + D \times (1 - S_A)$$

Result, Source, Destination: каждый имеет компоненты R^*A , G^*A , B^*A , A

Ассоциативность: в каком порядке делать сложения - всё равно, главное чтобы не менять объекты их местами (левый с правым)

Research #1

Берём свободного математика и просим его подобрать функцию

- в ряду стоят несколько светящихся объектов: светимость RGB, а прозрачность alpha (opacity)
- нужно превратить их в один такой объект (схлопнуть)
- если объект непрозрачный ($\alpha=1$), то то что позади него ничего не значит
- если объект полностью прозрачный ($\alpha=0$) то не влияет на результат
- частичная прозрачность наверное какая-то линейная

Кошмарный результат

Не понятно. Почему нельзя просто R, G, B, A

$$R_A = S_A + D_A \times (1 - S_A)$$

$$R_{RGB} \times R_A = S_{RGB} \times S_A + D_{RGB} \times D_A \times (1 - S_A)$$

Premultiplied Alpha

$$R = S + D \times (1 - S_A)$$

Not Premultiplied Alpha

$$R_A = S_A + D_A \times (1 - S_A)$$

$$R_{RGB} = (S_{RGB} \times S_A + D_{RGB} \times D_A \times (1 - S_A)) / R_A$$

А где тут фон-то?

```
varying vec2 vTextureCoord;
```

```
uniform sampler2D uSampler; ← Source
```

```
uniform float uAlpha;
```

```
void main(void)
```

```
{
```

Destination ?? нет, только Result

```
    gl_FragColor = texture2D(uSampler, vTextureCoord) * uAlpha;
```

```
}
```

Blending в WebGL

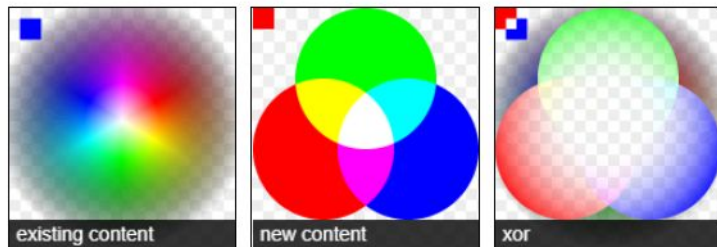
Porter Duff

- выбор из набора коэффициентов F
- $+$, $-$, \max , \min
- отдельно для цветов, отдельно для альфы (separate)
- https://threejs.org/examples/webgl_materials_blending_custom

$$R = S \times F_S + D \times F_D$$

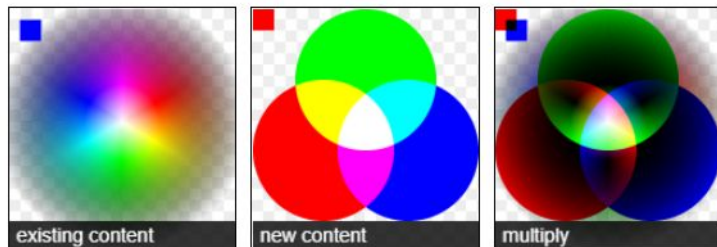
xor

Shapes are made transparent where both overlap and drawn normal everywhere else.



multiply

The pixels of the top layer are multiplied with the corresponding pixel of the bottom layer. A darker picture is the result.



Alpha compositing CSS / SVG

<https://developer.mozilla.org/en-US/docs/Web/CSS/mix-blend-mode> -

Огромный выбор режимов

<https://www.w3.org/TR/compositing-1/#blending>

Все формулы известны

<https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/globalCompositeOperation>

Они все есть в canvas 2d

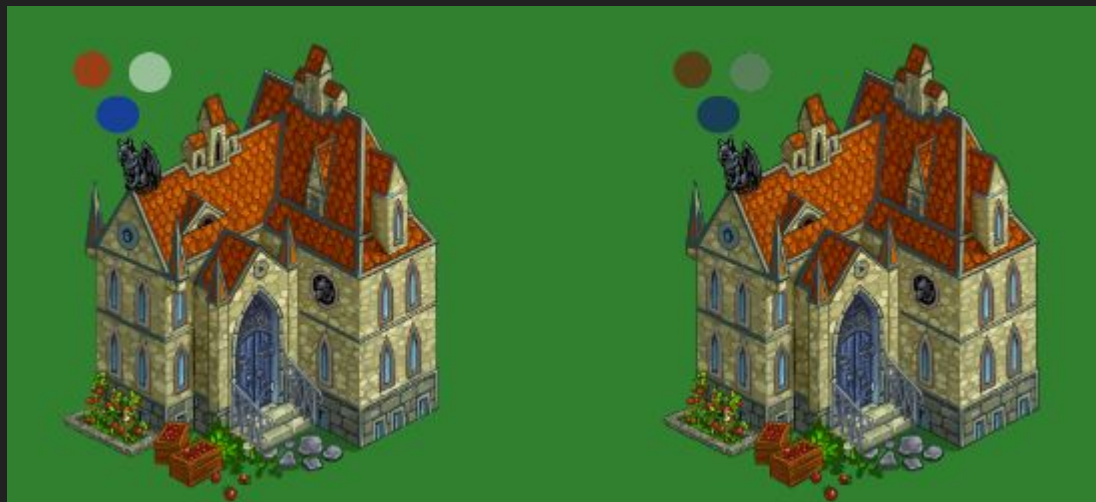
Можно помножить картинку на цвет с помощью доп. canvas

Разные текстуры, разные режимы premultiplied

NPM как PMA



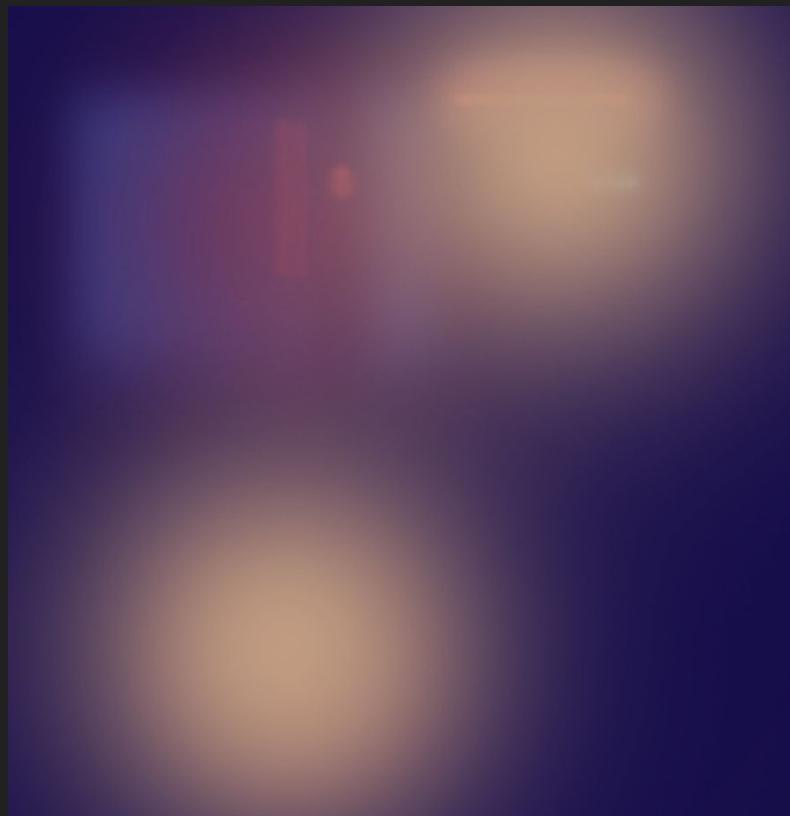
Всё ОК



Прошлись второй раз



Original



MUTE



Mult
+
Add



Overlay



Hardlight



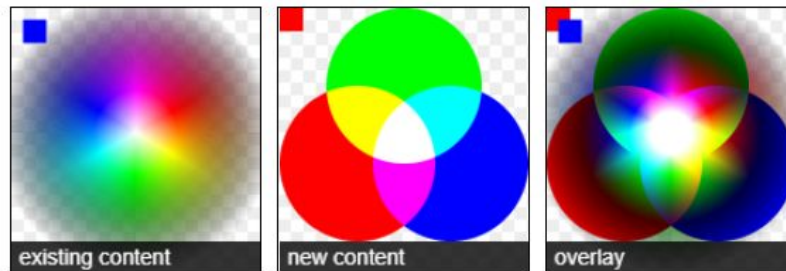
Advanced Blending

- По двум RGB выдать результат
- Альфу учесть в конце:
 - 0 - не меняем фон,
 - 1 - берем результат.
- WebGL только на шейдере :(

$$C_r = (1 - ab) \times C_s + ab \times B(C_b, C_s)$$

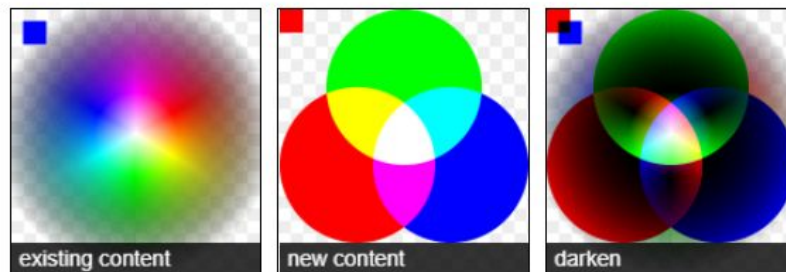
overlay

A combination of multiply and screen. Dark parts on the base layer become darker, and light parts become lighter.



darken

Retains the darkest pixels of both layers.



</alpha-compositing>

Advanced Blend in WebGL

```
uniform sampler2D uSampler[2];
uniform vec4 uColor;

void main(void)
{
    vec4 source = texture2D(uSampler[0], textureCoord) * uColor;
    vec4 target = texture2D(uSampler[1], vMapCoord);
    if (source.a == 0.0) {
        gl_FragColor = vec4(0, 0, 0, 0);
        return;
    }
    vec3 Cb = source.rgb/source.a, Cs;
    if (target.a > 0.0) {
        Cs = target.rgb / target.a;
    }
    vec3 multiply = Cb * Cs * 2.0;
    vec3 Cb2 = Cb * 2.0 - 1.0;
    vec3 screen = Cb2 + Cs - Cb2 * Cs;
    vec3 B;
    if (Cs.r <= 0.5) {
        B.r = multiply.r;
    } else {
        B.r = screen.r;
    }
}
```

```
if (Cs.g <= 0.5) {
    B.g = multiply.g;
} else {
    B.g = screen.g;
}
if (Cs.b <= 0.5) {
    B.b = multiply.b;
} else {
    B.b = screen.b;
}
vec4 res;
res.xyz = (1.0 - source.a) * Cs + source.a * B;
res.a = source.a + target.a * (1.0-source.a);
gl_FragColor = vec4(res.xyz * res.a, res.a);
```

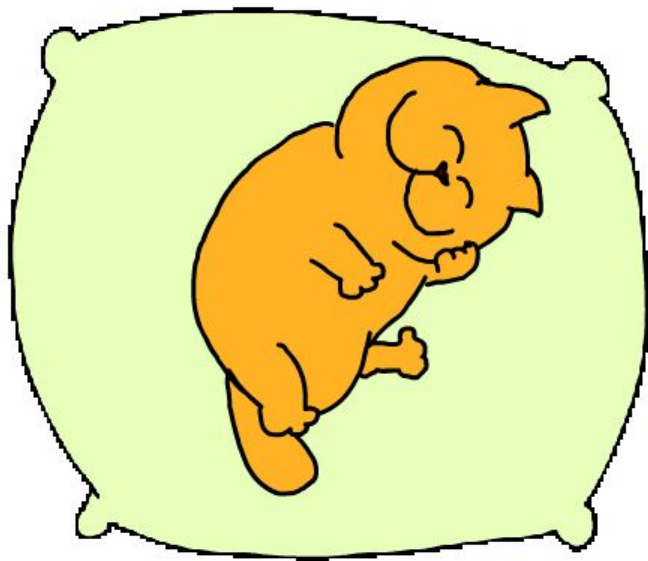


Сглаживание

gettyimages®

James Warwick

АА и **smoothing**: люди путают



AA и **smoothing**: люди путают

2Д-терминология

Текстурная фильтрация

- Внутри
- PER TEXTURE
- Почти ничего не стоит
- Обычно это и есть **smoothing**

Антиалиасинг

- Геометрия
- PER CANVAS
- Затратно

Фильтрация в CSS/SVG



Фильтрация в CSS/SVG

- Она либо есть, либо её нет. ON/OFF
- Думать не надо
- Можно применять к canvas: `crisp-edges` / `pixelate`

Наивный подход: Поворот не туда 🤪



1. Для каждого пикселя текстуры рассчитаем куда он попадёт
2. Получается надо “накапливать” цвет в пикселе
3. Доп. память для хорошей точности - не вариант!

Подозрительная `texture2D` в шейдере

```
varying vec2 vTextureCoord;
```

```
uniform sampler2D uSampler;
```

```
uniform float uAlpha;
```

```
void main(void)
```

```
{
```

```
    gl_FragColor = texture2D(uSampler, vTextureCoord) * uAlpha;
```

```
}
```

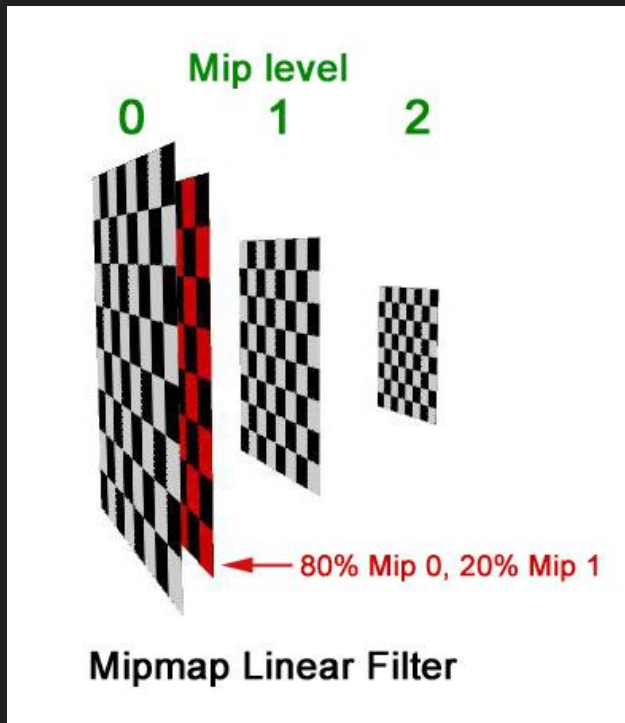
???



Вопросы к texture2D

1. что если мы попали между пикселями текстуры
2. 🚗 уехали за границу?
3. как оно определяет сжатие/растяжение
4. сколько раз мы можем это делать для одного пикселя экрана

Mipmap: downscale по степеням двойки



- **WebGL1** даёт это сделать **ТОЛЬКО** для **power-of-two** текстур:

256x256

512x1024

...

- **WebGL2** — для всех 🙌

Mipmap exploration

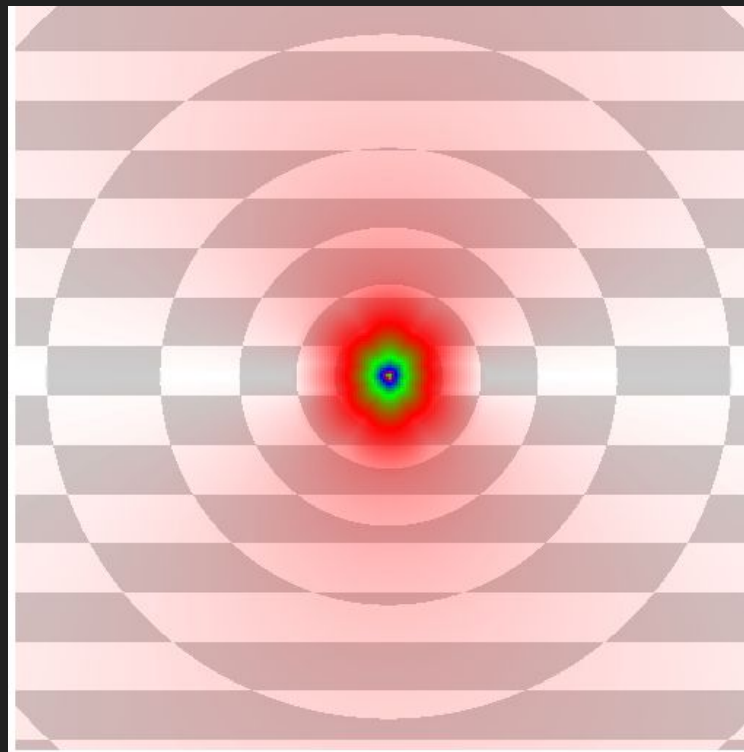
- Уровень считается по ddx, ddy
- ddx, ddy считается по тому что запросил

шейдер в соседних пикселях

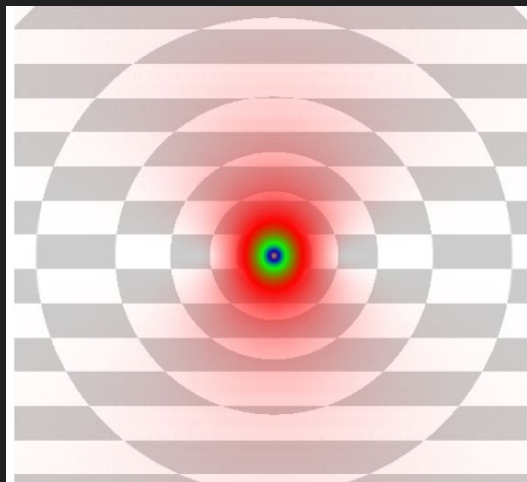
- формулы разные на разных устройствах

$$u = (x^2 + y^2)^{0.1}$$
$$v = y$$

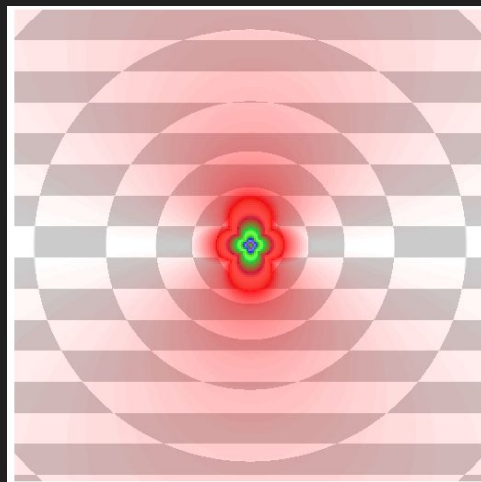
<https://github.com/Busyrev/mipExploration>



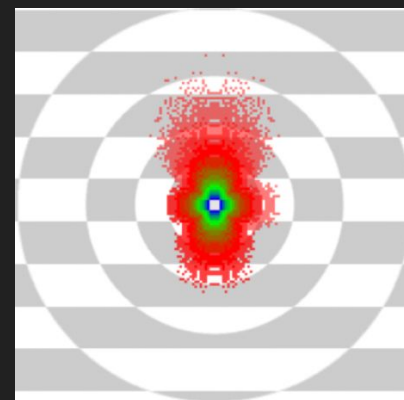
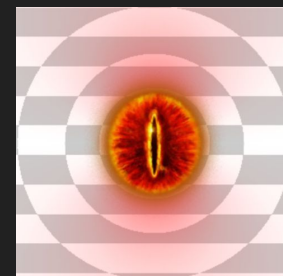
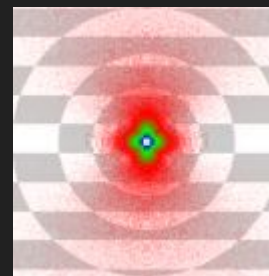
Ideal



isotropy
must die



Cursed GPU



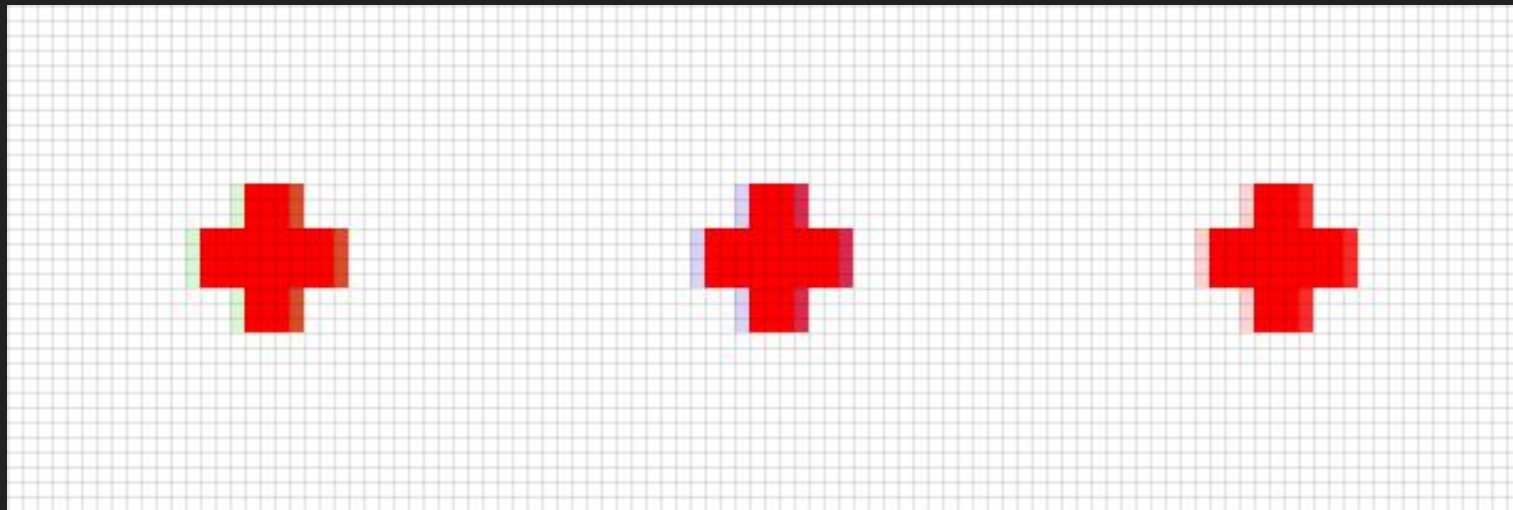
Ответы про texture2D

1. Есть NEAREST - ближайший и LINEAR - по четырём соседним
2. CLAMP (default) / REPEAT (опять pow2)
3. Скейл определяется по координатам которые запросил шейдер в соседних пикселях. Какие mip-уровни берутся зависит от девайса.
4. Кол-во вызовов зависит от девайса и системы. В Windows XP/7 ещё и ограничение на то как использовать его с условным оператором!
 - **Это не функция, это элемент многопоточности**

LINEAR filtering: premultiplied alpha strikes again!

Почему лучше делать premultiply.

<http://www.adriancourreges.com/blog/2017/05/09/beware-of-transparent-pixels/>



Make it bleed! (extrude)

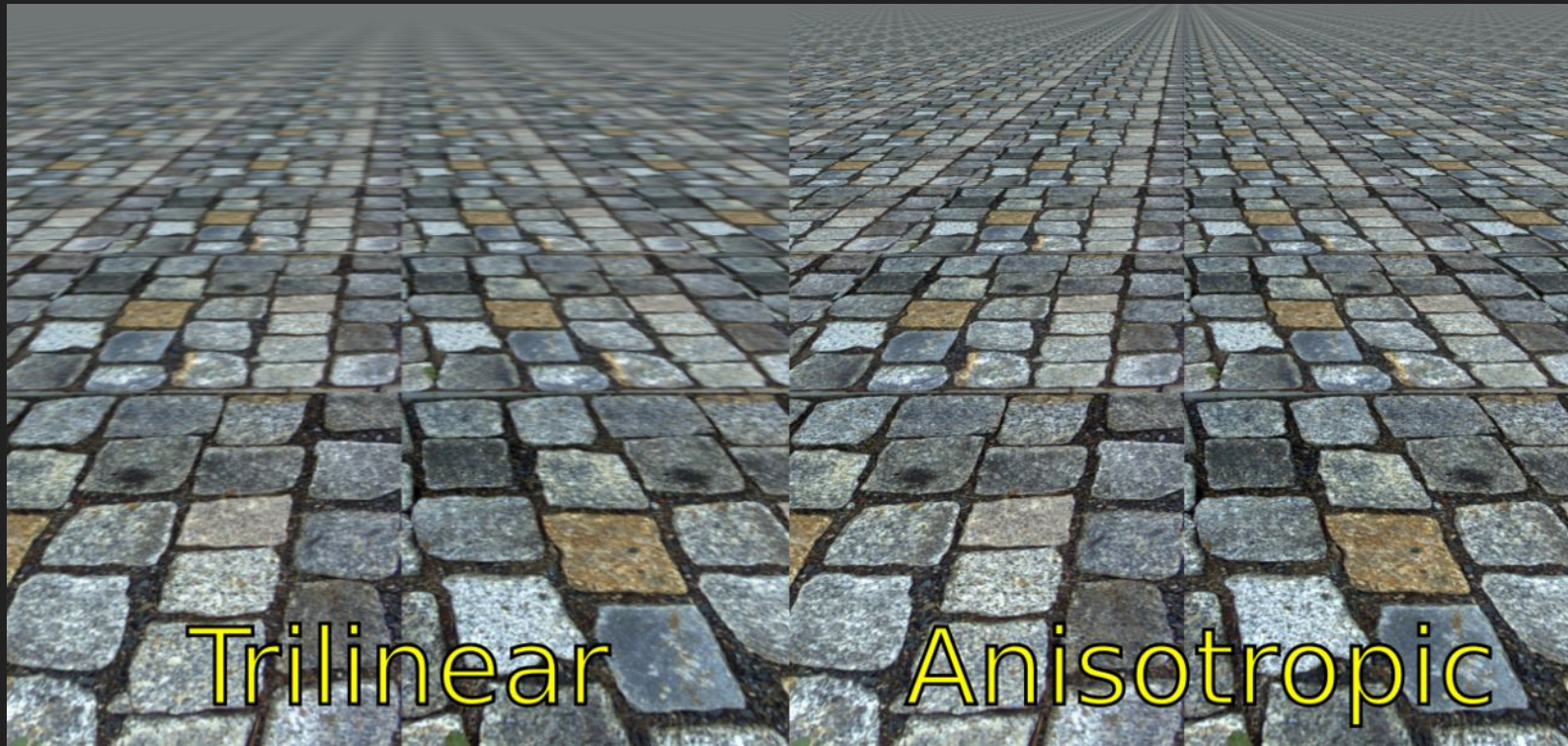


Google Fail at downscale 🤦

- <https://skia.org/user/modules/pathkit>
WebASM version of Skia PathKit
- Пытаемся нарисовать логотип Google на webgl-1
- мipmap работает только на row2
- Так не доставайся же ты никому!
- А WebGL 2 там пока не работает...



3D: anisotropic filtering



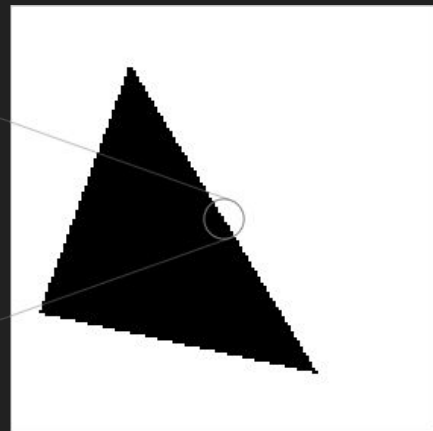
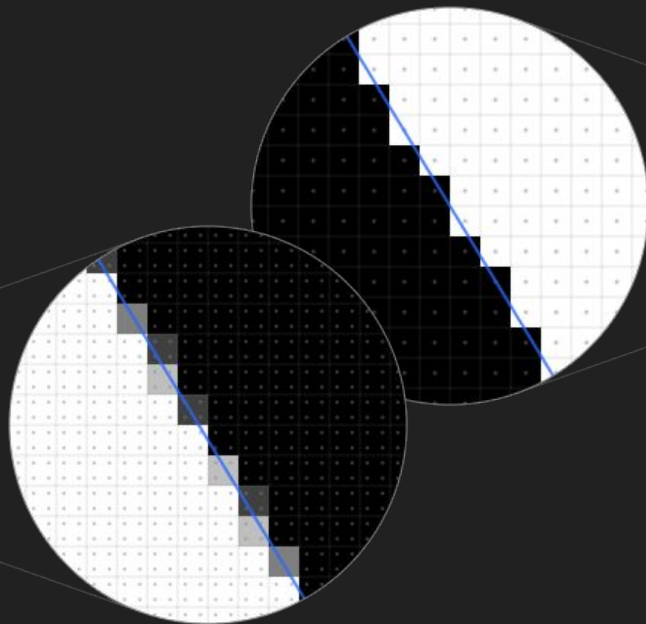
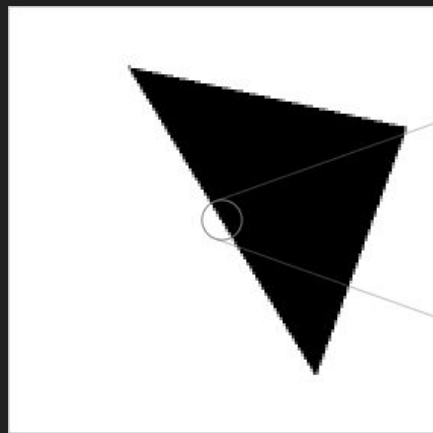
```
</texture-filtering>
```






Антиалисинг

Откуда берется лесенка?



Типы AA в браузере

Analytical AA

делается на CPU

Multi-Sampling AA

жрет доп память
дорого но не как скейл

Conflation (слияние) соседних фигур

Google Doodle

Memorial Day 2018

CreateJS shapes



<https://www.google.com/doodles/veterans-day-2018>

Решение с SVG

```
<defs>
<filter id="filter">
  <feComponentTransfer color-interpolation-filters="sRGB">
    <feFuncA type="table" tableValues="0 1 1"></feFuncA>
  </feComponentTransfer>
</filter>
</defs>
<g filter="url(#filter)">
```



Проблемы с MSAA в WebGL

- На WebGL1 (~50% устройств) только главный фреймбуфер
- Не все рендереры умеют использовать MSAA в WebGL2, архитектурные проблемы

Параметр при создании контекста { antialias: true }

Ретиновый макбук взлетает

</antialiasing>

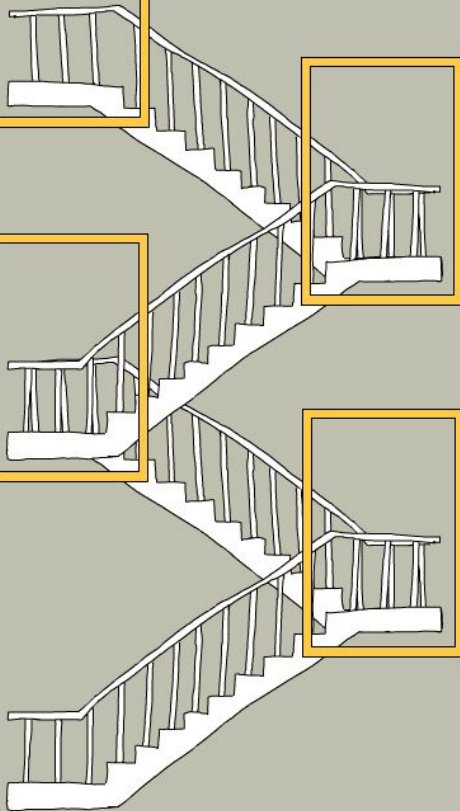
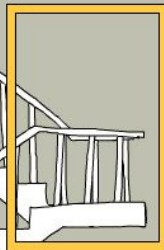
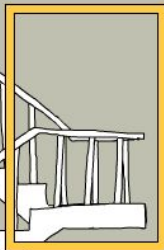
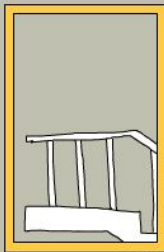
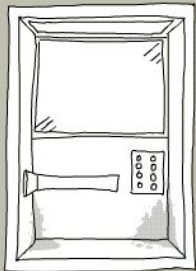
😍 WebGL нужен 😍

Композитные фильтры



Фильтры в SVG/CSS

- box-shadow на самом деле комбинация фильтров, кешируется
- <https://yoksel.github.io/svg-filters-02-2018/> - удобный редактор композитных фильтров, можно разобрать тень на части
- Все умеют работать в sRGB и linear RGB : разные градиенты, преобразования цвета
- Тормоза. Изменение свойств => перерисовка слоя, анимациям тяжело
- Canvas 2d умеет использовать SVG фильтры
- Набор заданных компонент, новые сделать нельзя



Фильтры в PixiJS

- Только в WebGL-режиме. На canvas-2d ищем героя!
<https://github.com/pixijs/pixi-filters> отдельный репозиторий для коллекции
- <https://github.com/pixijs/pixi.js/wiki/v5-Creating-filters> описание сложностей с написанием шейдера
- Нет редактора, но JS-код понятный
- Не тормоз, но и не газ: штук 20 на экране, или несколько полноэкранных выдержит.
- Нет разницы между анимацией и статикой, хотите кешировать - делайте это сами, PixiJS не следит за изменениями параметров

Фильтры в PIXIJS, dropShadow

Код Javascript

```
apply(filterManager, input, output, clear) {
    const target = filterManager.getFilterTexture();

    this._tintFilter.apply(filterManager, input, target, true);
    this._blurFilter.apply(filterManager, target, output, clear);

    if (this.shadowOnly !== true) {
        filterManager.applyFilter(this, input, output, false);
    }

    filterManager.returnFilterTexture(target);
}
```

Код шейдера

```
varying vec2 vTextureCoord;
uniform sampler2D uSampler;
uniform float alpha;
uniform vec3 color;

uniform vec2 shift;
uniform vec4 inputSize;

void main(void){
    vec4 sample = texture2D(uSampler, vTextureCoord - shift * inputSize.zw);

    // Premultiply alpha
    sample.rgb = color.rgb * sample.a;

    // alpha user alpha
    sample *= alpha;

    gl_FragColor = sample;
}
```

BeatFlyer

TRY IT NOW

www.beatflyer.com

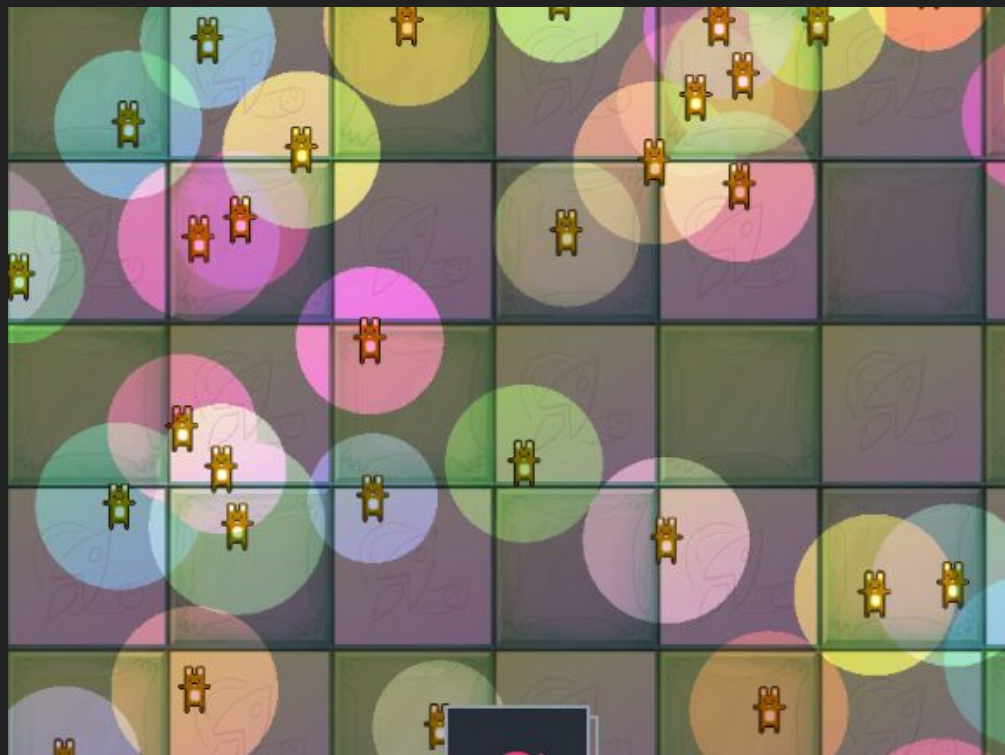
A close-up photograph of a slice of chocolate cake. The cake is multi-layered, with a thick, dark chocolate frosting between the layers. The top of the slice is decorated with several small, light-colored flowers with green centers. The slice is served on a white, shallow plate. To the left of the slice is a whole red apple and a sprig of small purple flowers. In the background, a glass of water is partially visible. The background wall has a floral pattern in shades of red and white.

Слои и композитная демка

Простая сцена со светом

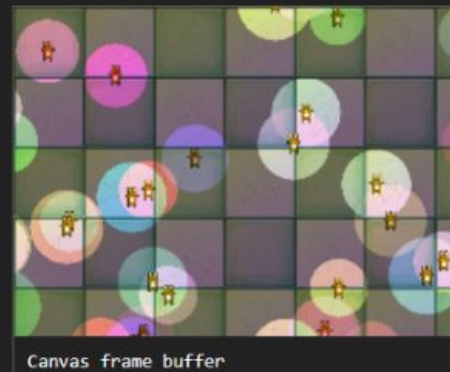
Опять кролики!

- Цветные круги добавляются в контейнер кролика
- Помечаются что должны быть нарисованы в другом контейнере
- На другом контейнере фильтр!
- Воспроизводимо на SVG и canvas 2d



SpectorJS (расширение в chrome store)

- Даёт список WebGL команд одного фрейма
- Показывает промежуточный результат
- Можно сохранить отчёт
- Спасибо BabylonJS!



Tues 8/6

Evening



КОНКЛЮЖИ



+/- WebGL 🤞

- Можно такое “BAU” которое не сделаешь на CSS/SVG, ну или сделаешь но через странные места
- Чтобы получить схожую картинку придётся много учить, опыт важен
- Можно использовать коллективный опыт WebGL-сообщества который с каждым годом растёт (айда к нам в чатики!)
- За GLSL придётся продать душу
- Ограничение на количество `<canvas>` на экране
- Получающуюся картинку можно отскейлить браузером через CSS

PixiJS

- 100.000 кроликов!
- Мы уже почти всё починили
- Простая абстракция от WebGL, который изначально сложный
- Мощная система работы с фильтрами, но без кеша
- Частичная поддержка вектора и интеграция с SVG (pixi5-svg)
- Сцена а-ля Flash , без layout
- Можно написать шейдера
- Даёт измазаться в генерации текстур
- Можно врубить MSAA и убрать швы

CSS & SVG 🐱

- Layout!
- Почти идеальный вектор. Идеальный на Nvidia 10xx
- Лучшие средства инспекции в браузере
- Покрывает почти все потребности эффектов на современных сайтах
- Встроенные фильтры, тени кешируются и не тормозят

Canvas 2D

- То ускорен на GPU, то — нет
- Рендерит SVG
- Простое API для рисования, возможности от SVG Path
- Фиг отрубишь антиалиасинг
- Есть уже везде!
- Всегда хороший скейл
- Держит тыщи кроликов
- Фильтры от SVG без кеша с тормозами (но есть!)

ССЫЛКИ ССЫЛКИ ССЫЛКИ

<https://github.com/pixijs/pixi.js>

PixiJS репозиторий, wiki (важно!) со ссылками

<https://www.html5gamedevs.com/forum/15-pixijs/>

PixiJS подфорум на html5gamedevs

<https://t.me/gamedevforweb> , https://t.me/threejs_ru

Русскоязычные telegram чаты о WebGL

<https://beatflyer.com/>

приложение использующее множество фильтров

Добивание

https://www.researchgate.net/publication/262357352_GPU-accelerated_Path_Rendering

nvidia пытается в векторную графику (2012)

<https://docs.google.com/document/d/1bs27MuOGB6OQrIQ2yNbStUwQFWI3MvxtIELIIMyIVgA/edit>

Analytical Anti-Aliasing in Skia, на самом деле есть в сорцах самой Skia

https://developer.nvidia.com/gpugems/GPUGems3/gpugems3_ch25.html

Rendering Vector Art on the GPU

<https://habr.com/ru/post/141910/>

Первый понитред на хабре, про canvas-2d и турбо-паскаль.

Добавки!!!

<http://www.adriancourreges.com/blog/2017/05/09/beware-of-transparent-pixels/>

статья по premultiplied alpha

<https://github.com/Busyrev/mipExploration>

тест на особенности выбора уровня mip

<https://ciechanow.ski/alpha-compositing/>

хорошая статья по alpha compositing

<https://codepen.io/stefanweck/pen/Vbgeax>

фильтр с дождиком

Про себя

<https://twitter.com/ivanpopelyshev>

<https://github.com/ivanpopelyshev/>

https://www.youtube.com/channel/UCaHarZvr6y1Og4dMLE_nbWQ

Анимация!

<https://www.youtube.com/channel/UCZfYIIsIIUFyfmLodIpzU0g>



Ссылка на слайды

<https://bit.ly/2Nt7L1h>

