

План доклада

1

Обзор средств

2

Задача с AI

3

Добавляем GPU

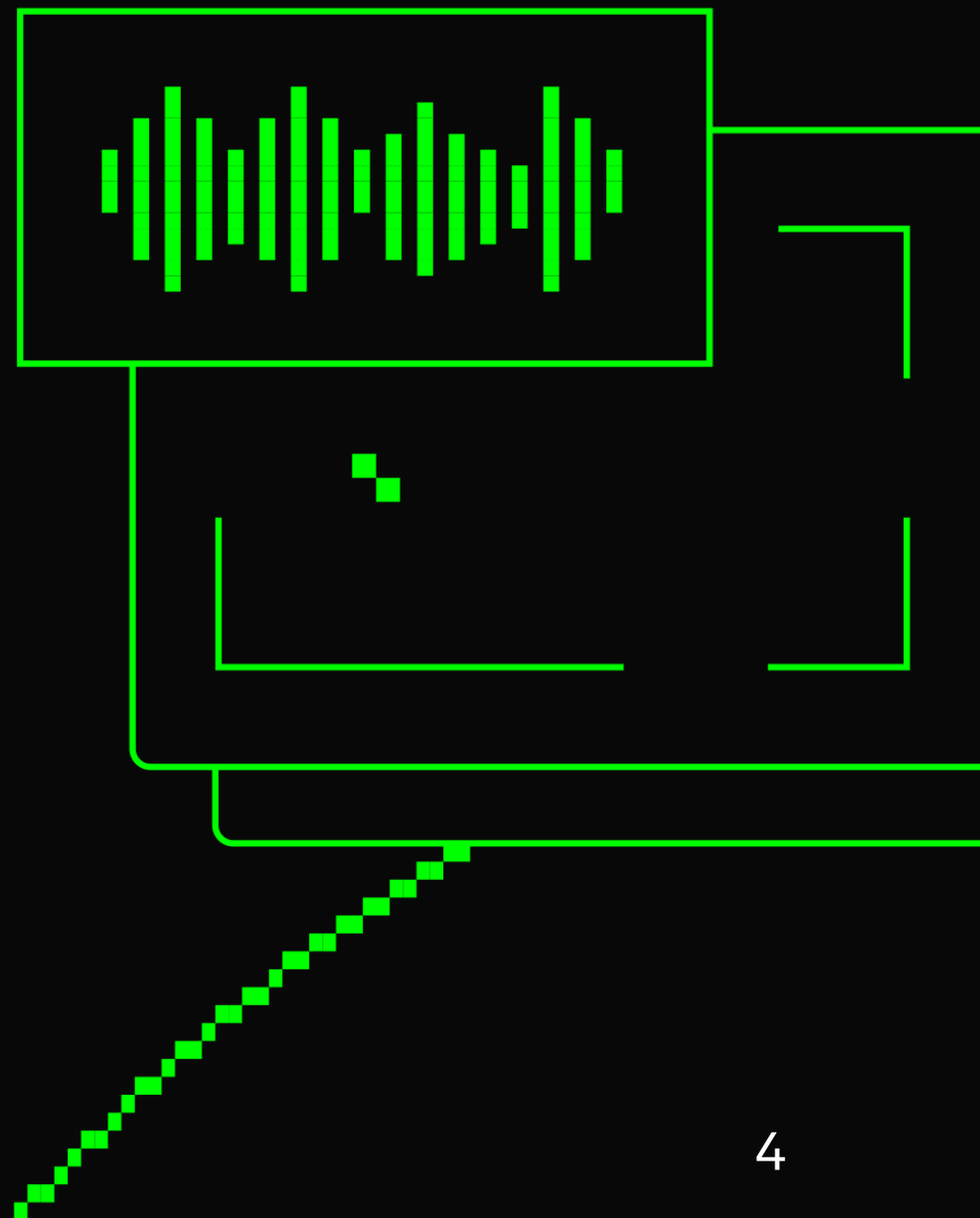
4

Обработка
вотдельном
потоке

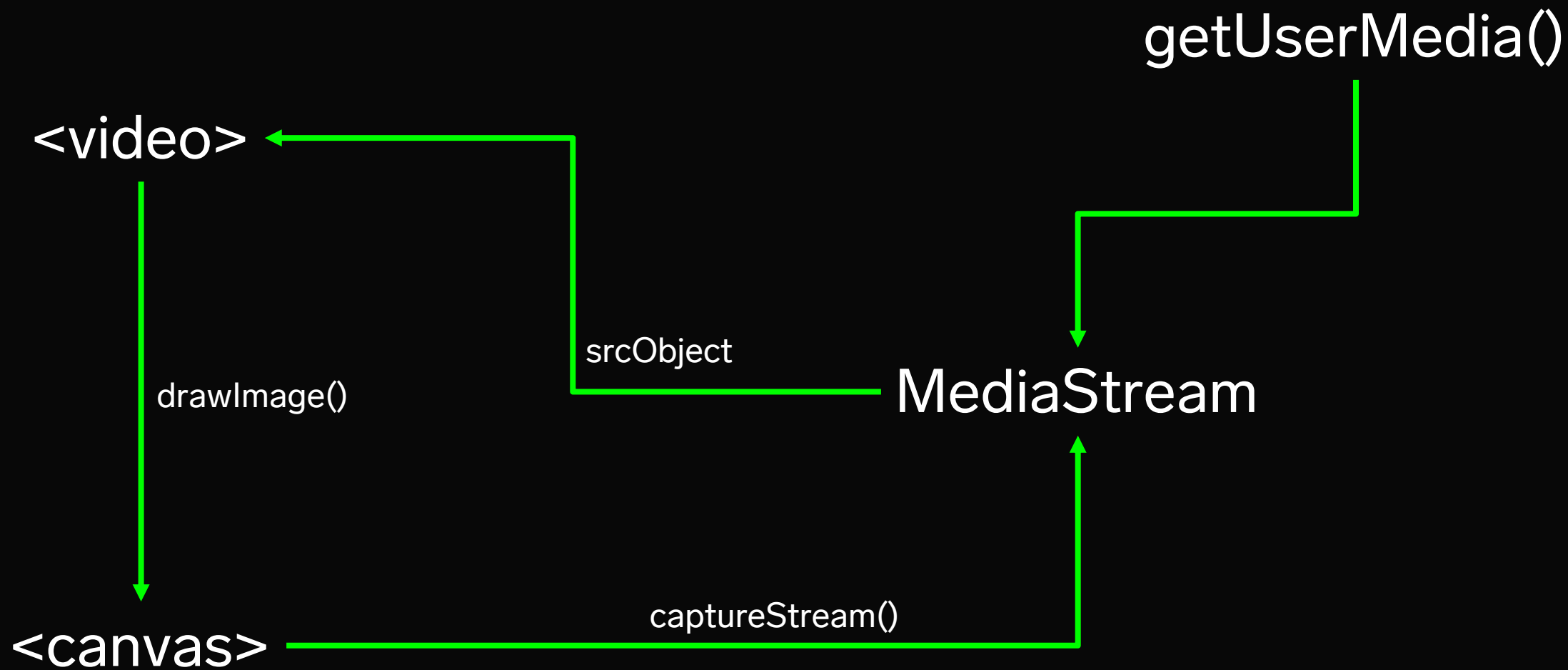
5

Финальные
штрихи

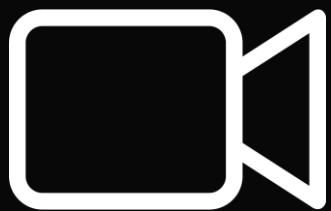
Обзор средств



Видео на фронте



Задача: Ватермарк



Ватермарк: код

```
const videoElement = document.createElement('video');
const mediaStream = await navigator.mediaDevices.getUserMedia({ video: true });
videoElement.autoplay = true;
videoElement.srcObject = mediaStream;

// позже, после события canplay
const canvas = document.createElement('canvas');
canvas.width = videoElement.videoWidth;
canvas.height = videoElement.videoHeight;
const videoCtx = canvas.getContext('2d');

videoCtx.drawImage(videoElement, 0, 0,
    canvas.width, canvas.height);
```

Ватермарк: ещё код

```
const LogoElement = document.createElement('img');  
LogoElement.setAttribute('src', LOGO_URL);
```

```
videoCtx.globalAlpha = 0.5;
```

```
// позже, load на изображении уже сработал  
videoCtx.drawImage(LogoElement,  
    canvas.width - LogoElement.naturalWidth - 10,  
    10,  
    LogoElement.naturalWidth,  
    LogoElement.naturalHeight  
);
```

```
canvas.captureStream(20); // MediaStream
```

Цикл обработки

```
requestAnimationFrame();
```

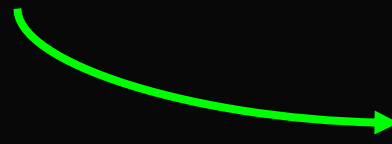
60fps 16ms 120fps 8ms 240fps 4ms

document.hidden

```
videoElement.requestVideoFrameCallback();
```

setTimeout setInterval

```
setTimeout(() => {  
  render();  
}, 1000 / targetFramerate);
```



WebWorker

MediaStream[Track]

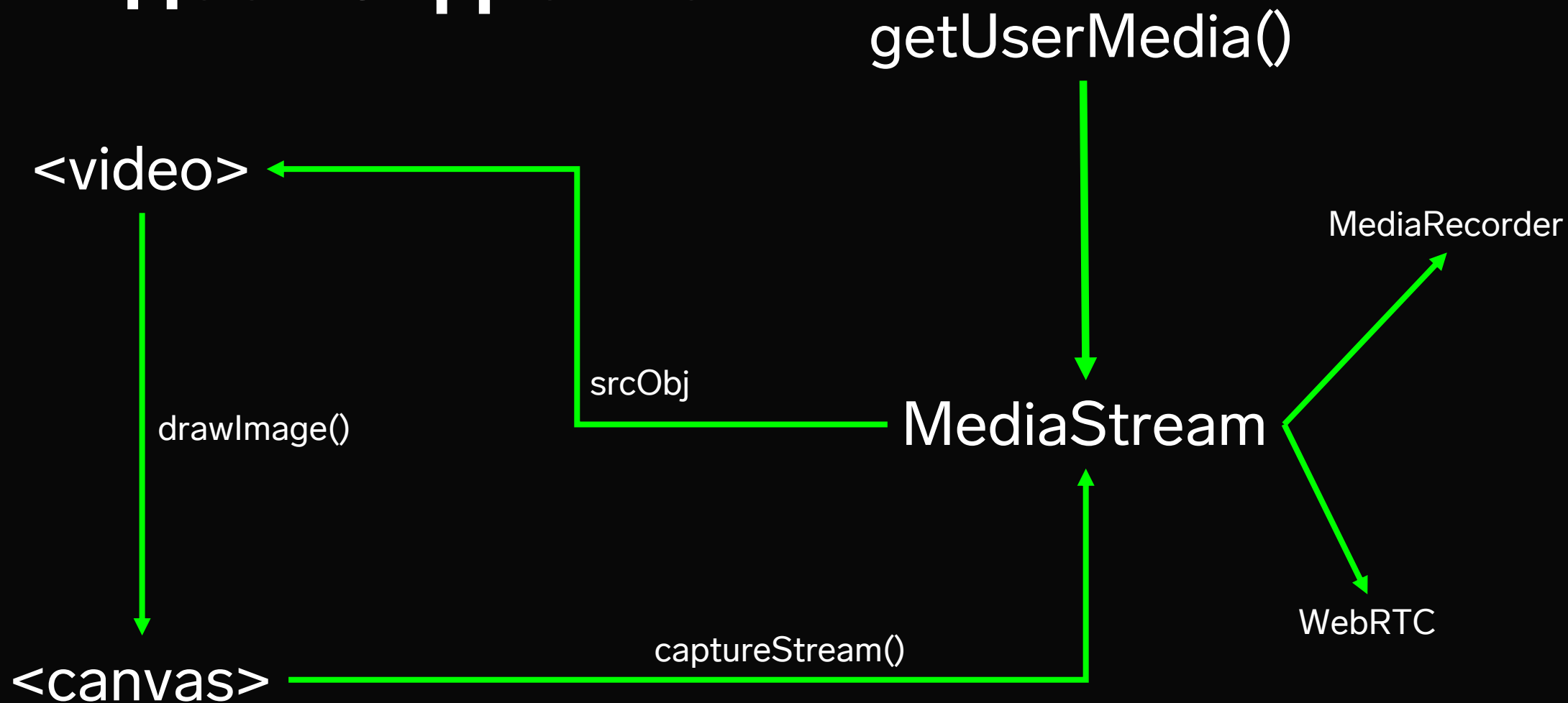
Отдать

- <video>
- WebRTC
- MediaRecorder, ImageCapture

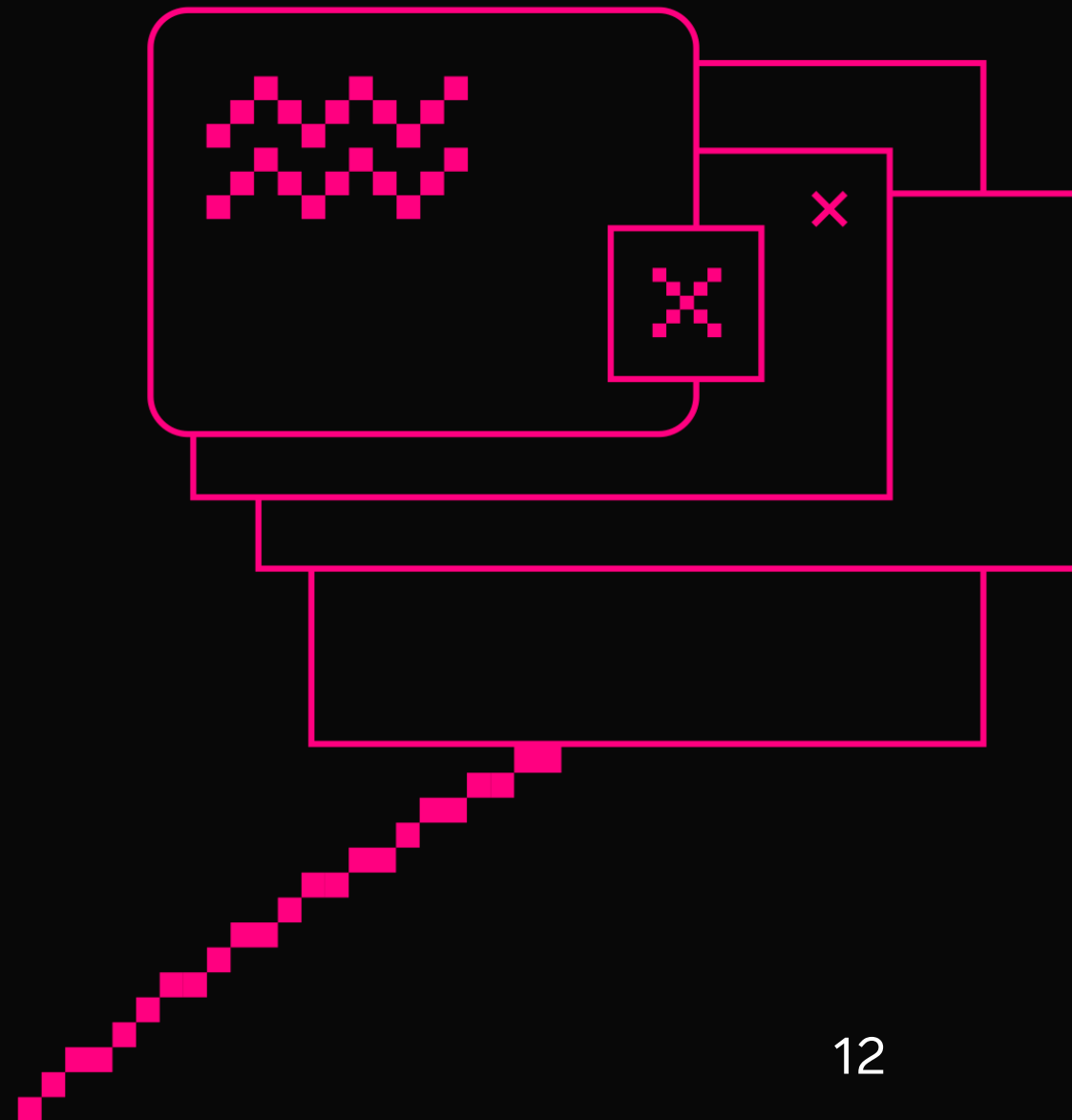
Получить

- WebRTC
- getUserMedia()
- getDisplayMedia()
- captureStream()

Видео на фронте



Задача с AI



Задача: замена фона

1



3



2



4



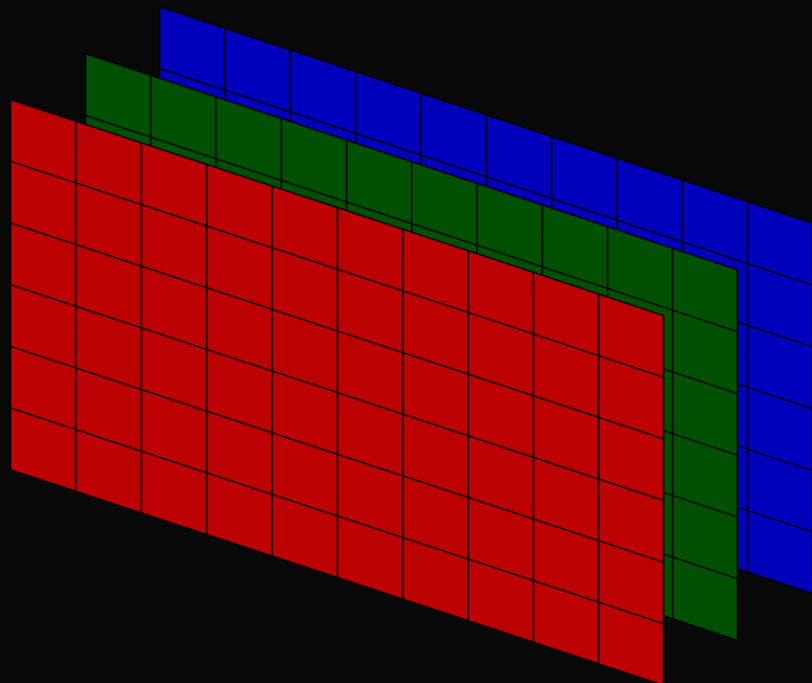
Основа решения задачи — ML модель

TFLite + WASM

Карточка модели

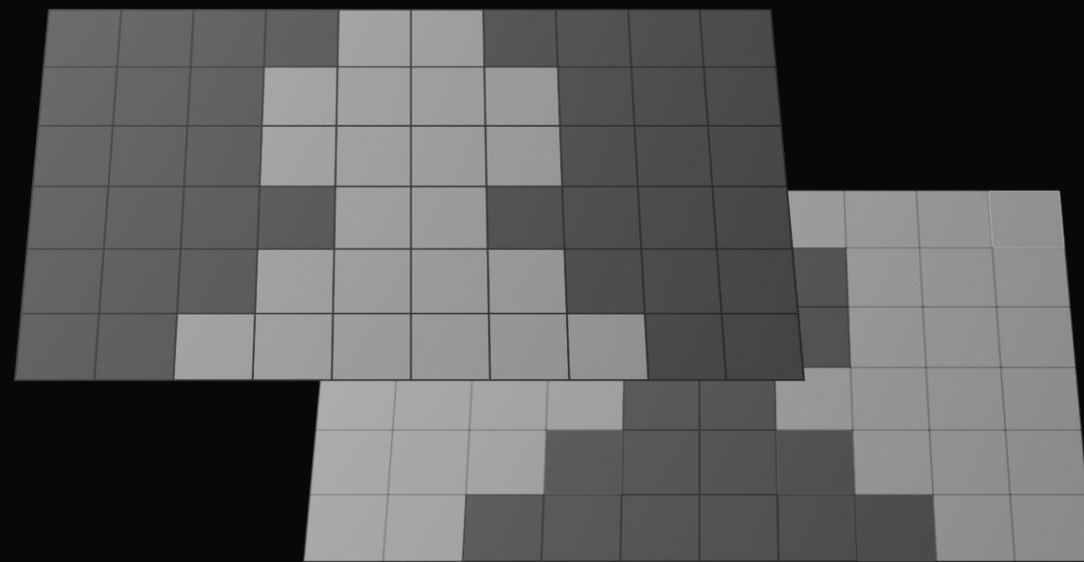
Немного про ML модель

ВХОД



256 x 144 RGB, 3 канала

ВЫХОД



256 x 144, 2 канала

Немного про ML модель

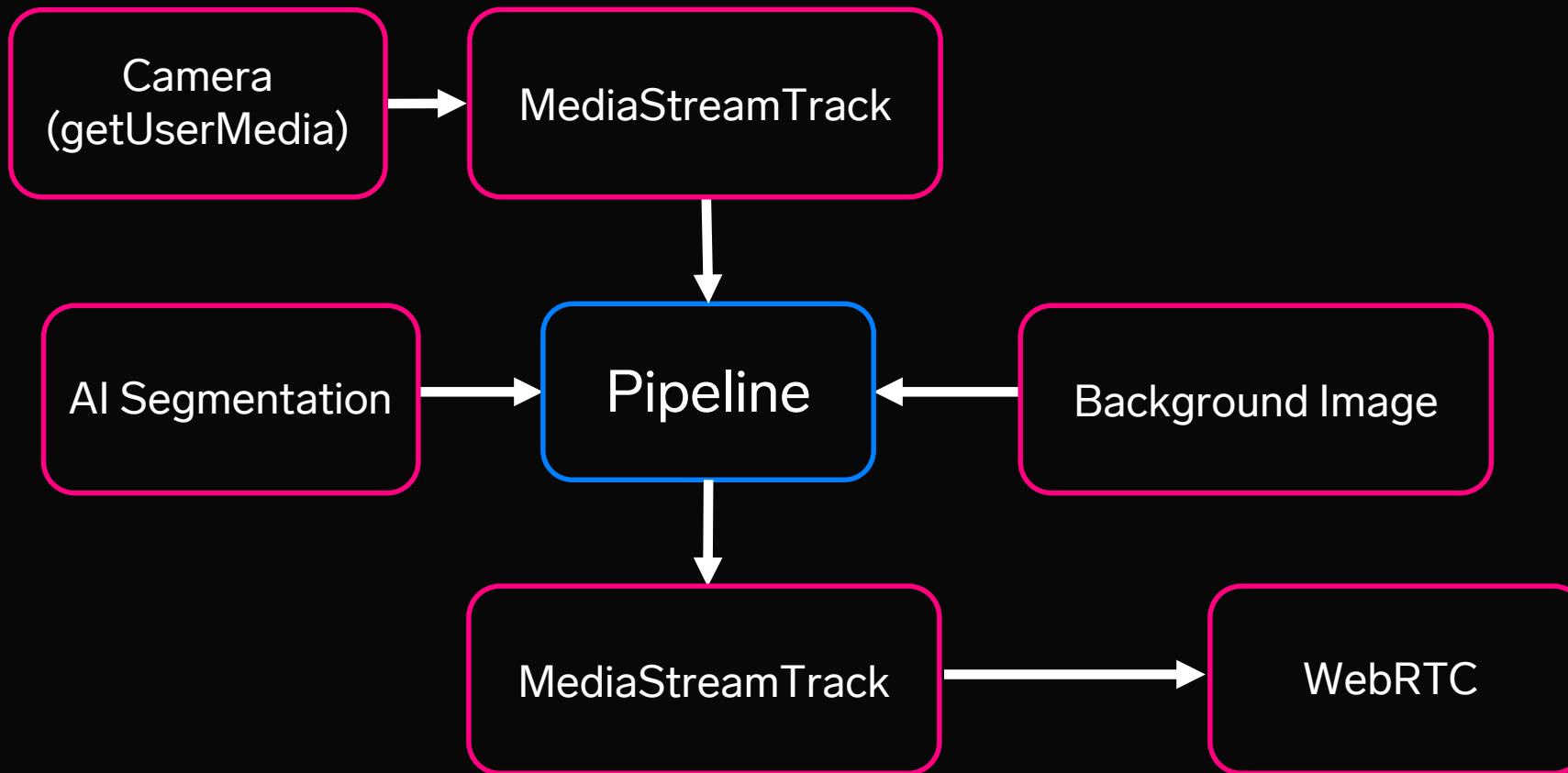
Вход модели — плоский массив пикселей 0...1

```
tflite.HEAPF32[inputMemoryOffset + i] = pixelData[i] / 255;
```

```
tflite._runInference();
```

```
maskData[j] = tflite.HEAPF32[outputMemoryOffset + j] * 255;
```

Формализуем задачу



Как из видео получить сырые пиксели для модели?

```
const videoCanvas = document.createElement('canvas');
videoCanvas.width = 256; videoCanvas.height = 144;
const videoCtx = videoCanvas.getContext('2d');

// позже...
videoCtx.drawImage(inputVideoElement, 0, 0,
    inputVideoElement.width, inputVideoElement.height, 0, 0, 256, 144);
const imageData = videoCtx.getImageData(0, 0, 256, 144);

const dataArray = imageData.data; // Uint8ClampedArray
```


Композиция кадра

```
const outputCanvas = document.createElement('canvas');  
outputCanvas.width = 1280; outputCanvas.height = 720;  
const outputCtx = outputCanvas.getContext('2d');
```

```
const segmentationCanvas = document.createElement('canvas');  
videoCanvas.width = 256; videoCanvas.height = 144;  
const segmentationCtx = segmentationCanvas.getContext('2d');
```

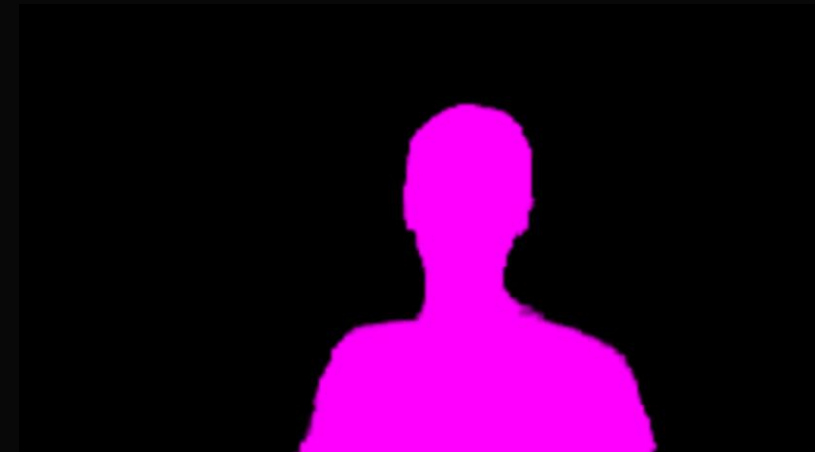
```
const segmentationMask = new ImageData(typedArray, 256, 144);  
segmentationCtx.putImageData(segmentationMask, 0, 0);
```

Композиция кадра

```
outputCtx.filter = 'blur(4px)';
```

```
outputCtx.drawImage(  
    segmentationCtx, 0, 0,  
    256, 144, 0, 0,  
    1280, 720,  
);
```

```
outputCtx.filter = 'none';
```



Композиция кадра

```
outputCtx.globalCompositeOperation = 'source-in';
```

```
// Рисуем человека по маске
```

```
outputCtx.drawImage(videoCanvas, 0, 0);
```



source-in

```
outputCtx.globalCompositeOperation = 'destination-over';
```

```
// Рисуем фон за человеком
```

```
outputCtx.drawImage(backgroundImage, 0, 0);
```



destination-over

Композиция кадра



```
outputCanvas.captureStream(20); // MediaStream
```

Оценим реализацию

Понятные фронту API,
Поддерживаемый код



Браузерная
совместимость



Мало возможностей
canvas для сложных
задач



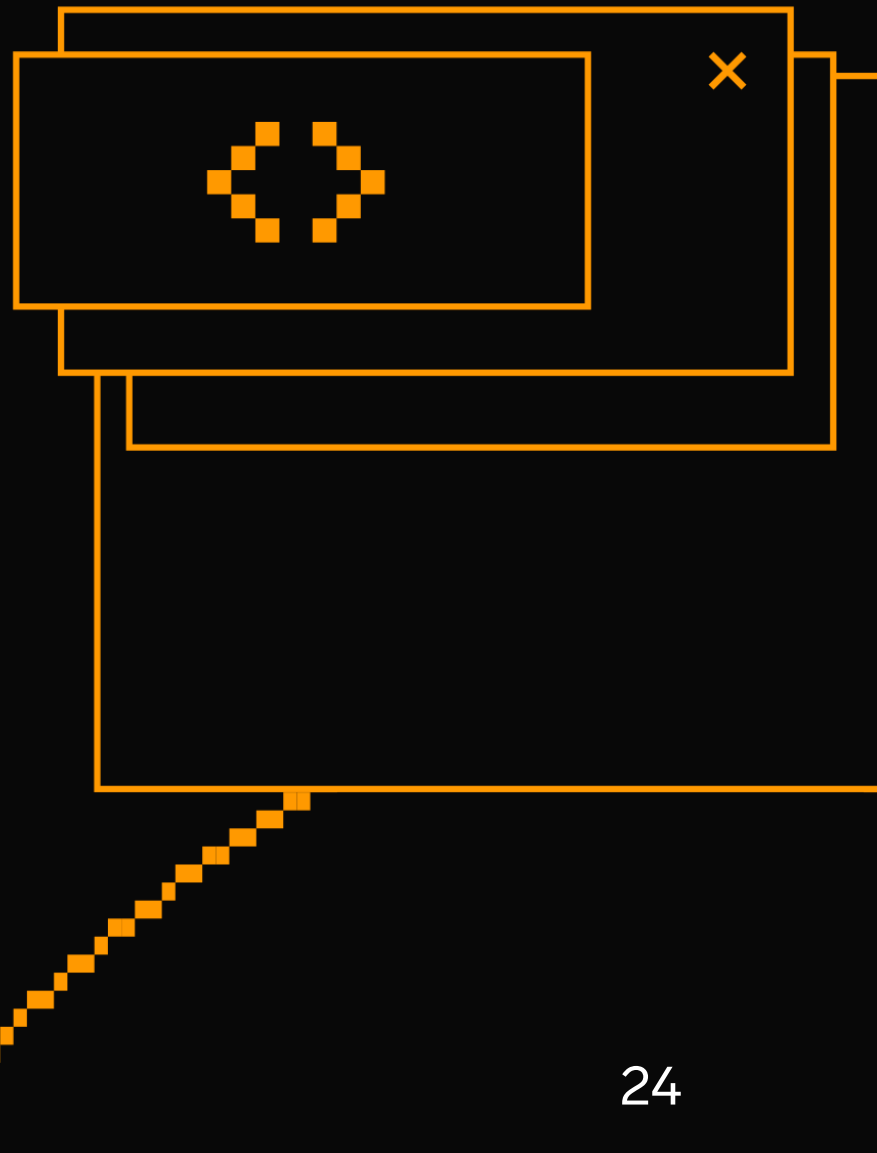
Медленно для тяжелого
реалтайма



Загружен основной поток



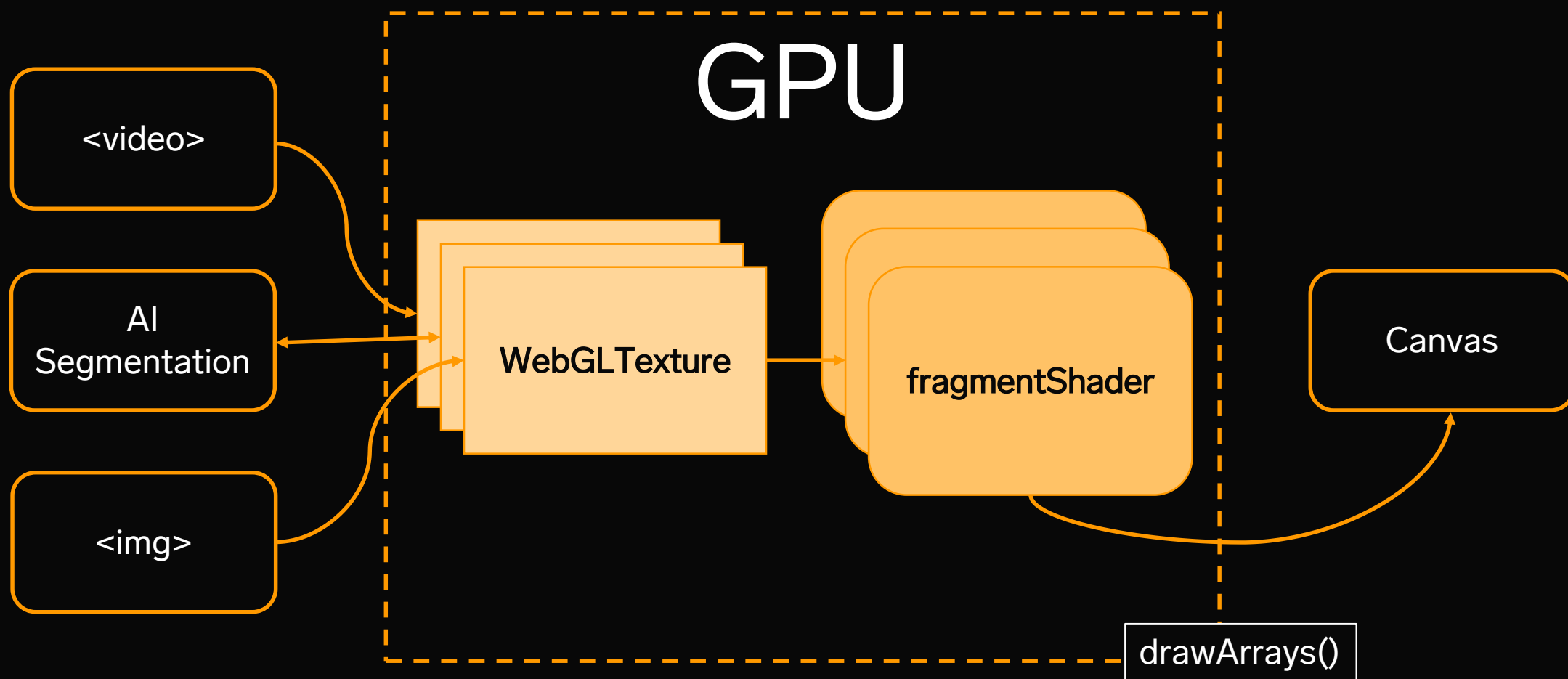
Добавляем GPU



Пайплайн на WebGL

подсмотрели реализацию у [Volcomix](#)

Пайплайн на WebGL



Пайплайн на WebGL



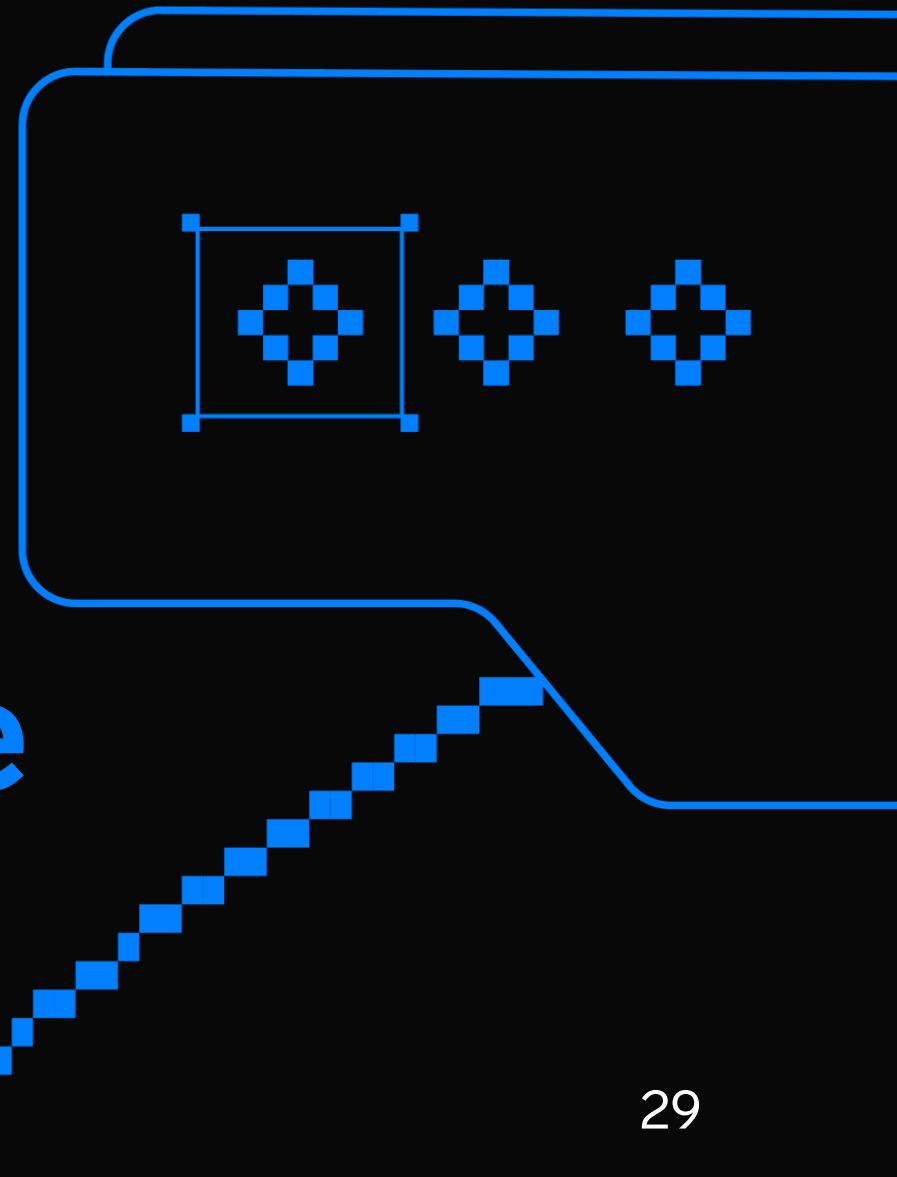
- Производительность:
 - VideoElement можно загружать напрямую (без canvas)
`WebGLRenderingContext: texImage2D()`
 - Можно читать пиксели напрямую
 - `WebGLRenderingContext: readPixels()`
 - Быстрая работа с графикой, эффектами благодаря шейдерам
- Продвинутые эффекты
 - Хитрое сглаживание маски
 - Коррекция освещенности (light-wrap)

Пайплайн на WebGL



- Совместимость (зоопарк GPU)
- Стабильность
- Сложно разрабатывать и вносить изменения (нужно разбираться в GL)
- При выполнении ML модели на CPU каждый кадр надо пересылать между CPU и GPU

Обработка в **отдельном потоке**



Освободим основной поток

WebWorker

Transferable Object

ArrayBuffer

Transferable Object

- Содержит произвольные бинарные данные
- Не содержит информации о размерах или каналах картинки
- Генерируется и потребляется разными API (WebGL)
- Для чтения или записи можно обернуть в TypedArray

ArrayBuffer

Transferable Object

2D

```
const typedArray = context.getImageData().data;  
typedArray.buffer; // ArrayBuffer
```

```
context.putImageData(new ImageData(  
new Uint8ClampedArray(arrayBuffer)), width, height);
```

WebGL

```
texImage2D(...); // ArrayBuffer в текстуре  
context.readPixels(...); // взять ArrayBuffer из WebGL
```

Как вынести AI в отдельный поток

```
this.worker.postMessage(  
  {  
    action: 'segmentation',  
    imageData: this.workerBuffer,  
  },  
  // список объектов,  
  // управление которыми передаем  
  // в другой поток  
  [this.workerBuffer],  
);
```

```
self.postMessage(  
  {  
    type: 'segmentation_done',  
    maskData: bufferData,  
  },  
  [bufferData],  
);
```

Как вынести **ВСЮ**
обработку
в отдельный поток?

Offscreen Canvas



69



105



17

Transferable Object

- Позволяет рисовать на canvas в воркерах
- Является Transferable, но контекст (2d, webgl, ...) между потоками переносить нельзя
- Не имеет визуального представления (при создании через конструктор)

ImageBitmap

Transferable Object



50



42



15

- Растровое изображение, имеет размерность
- Можно быстро переложить на canvas
- Без компрессии
- Нельзя переиспользовать

Скорость потребления памяти

$1280 \text{ ширина} * 720 \text{ высота} * 1 \text{ байт} * 3 \text{ канала} * 20 \text{ кадров/с} \approx 52,75 \text{ Мб/с}$

ImageBitmap



50



42



15

Transferable Object

```
createImageBitmap(ImageElement | VideoElement | Canvas  
| OffscreenCanvas | Blob | ImageData | ...)
```

2D

```
context.drawImage(imageBitmap,...)
```

WebGL

```
context.texImage2D(...); // ImageBitmap в текстуру
```

VideoFrame

Transferable Object



94



130



16.4

- Ссылка на кадр во внутренностях браузера
- Имеет формат (и он не всегда RGBA)
- Можно переложить на canvas
- Без компрессии
- Нельзя переиспользовать

VideoFrame

Transferable Object



94



130



16.4

```
new VideoFrame(ImageElement | VideoElement | CanvasElement  
| ImageBitmap | OffscreenCanvas | ArrayBuffer)
```

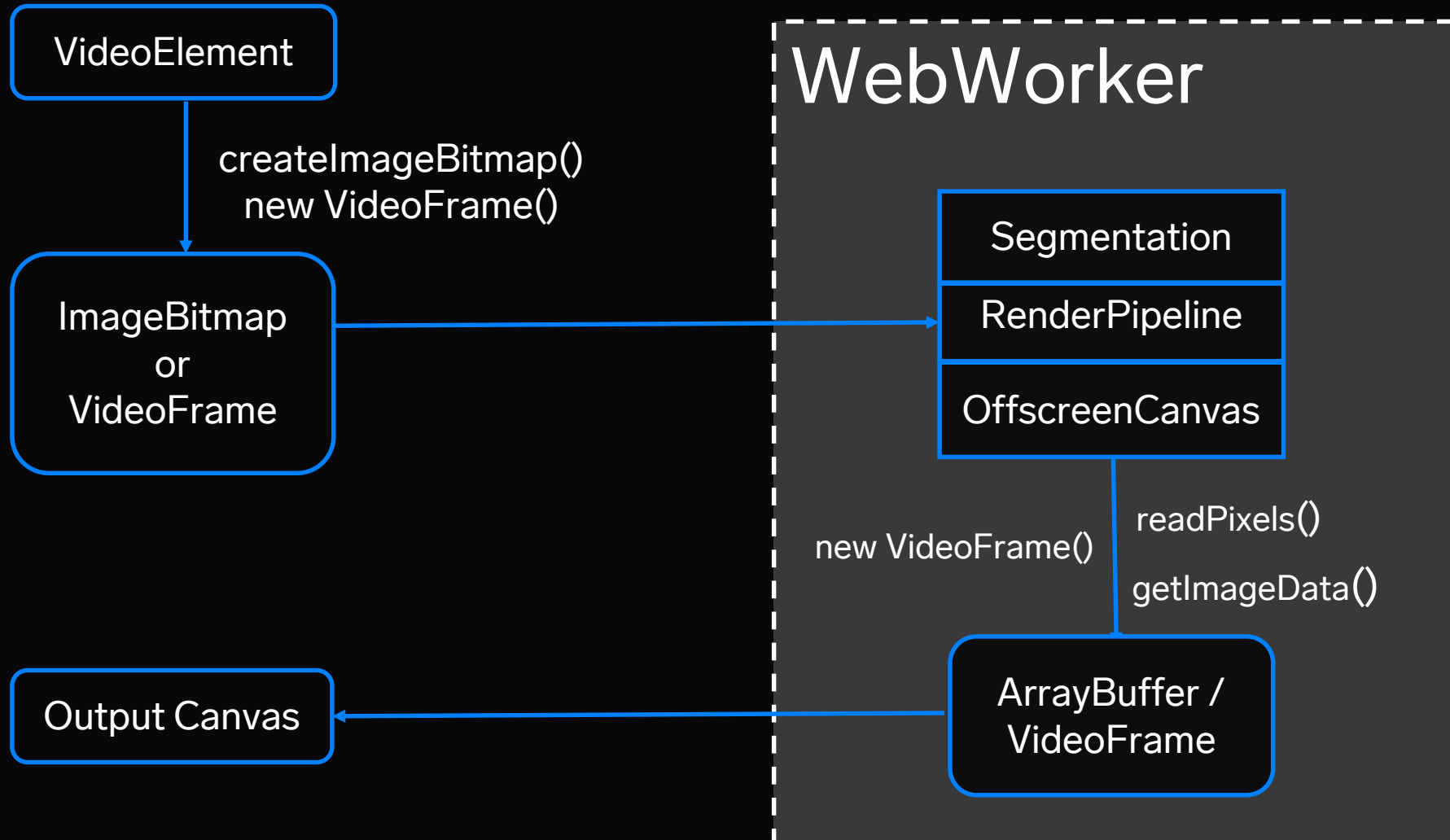
2D

```
context.drawImage(videoFrame,...)
```

WebGL

```
context.texImage2D(...); // VideoFrame в текстуре
```

Весь пайплайн в отдельном потоке



Canvas.transferControlToOffscreen()

- Позволяет получить OffscreenCanvas прокси из Canvas
- Отображение в основном потоке, рендер – в WebWorker
- Отрисовка может быть остановлена, если документ свернут

[Примеры кода и демо](#)



69

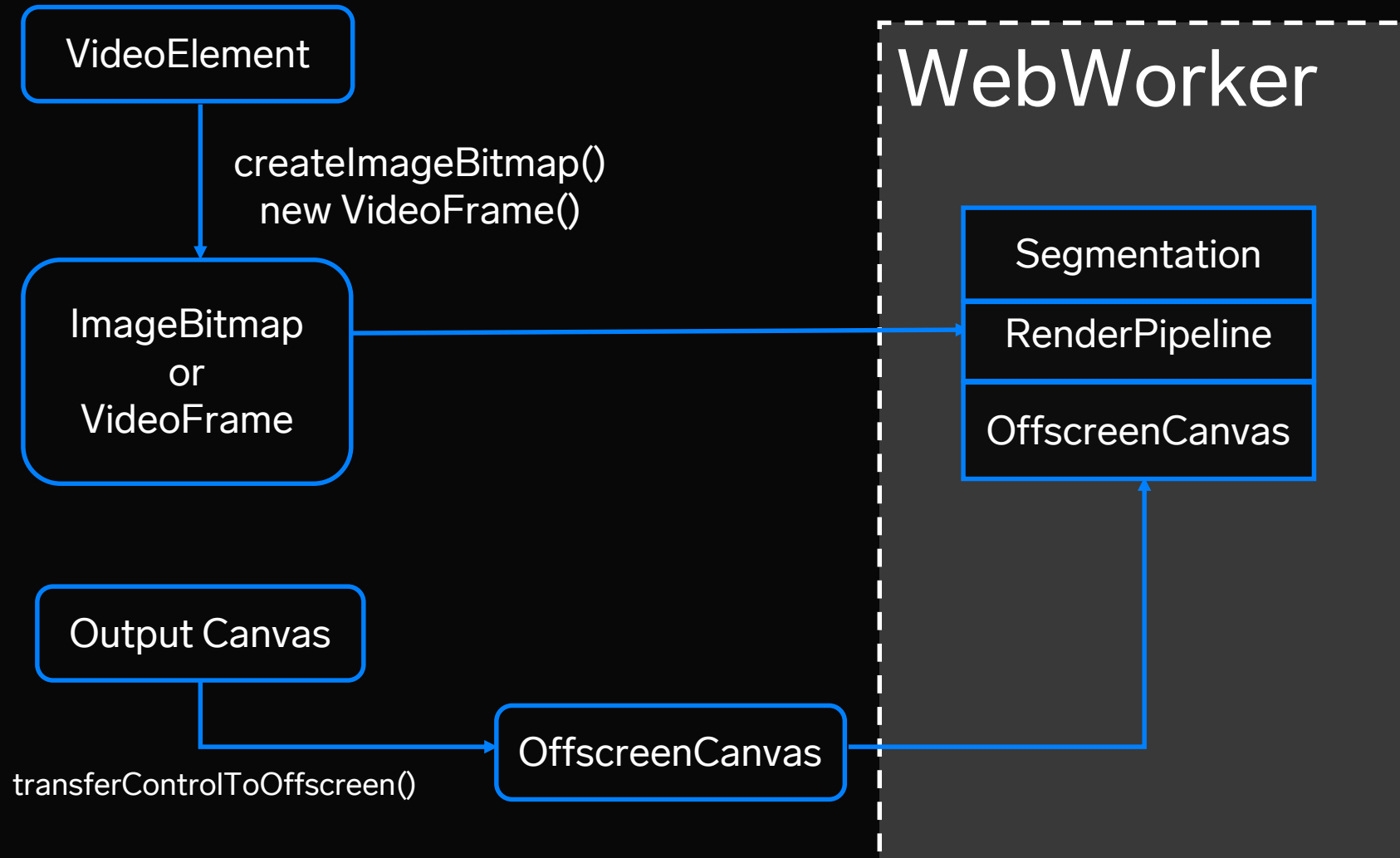


105



17

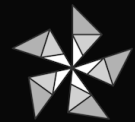
Весь пайплайн в отдельном потоке





Финальные Штрихи

Заменим AI модель?



Onnx runtime web

WASM

WebGL

`Tensor.fromTexture()`
`Tensor.texture`

WebGPU

`Tensor.fromGpuBuffer()`
`Tensor.gpuBuffer`

Можно не пересылать данные между GPU ↔ CPU

Дополнительно

ElementCapture



BrowserCaptureMST

W3C Draft

: restrictTo()

Out of Spec

SharedArrayBuffer

ES 2026

InsertableStreamsForMST



MediaStreamTrackProcessor

W3C Draft



18

MediaStreamTrackGenerator

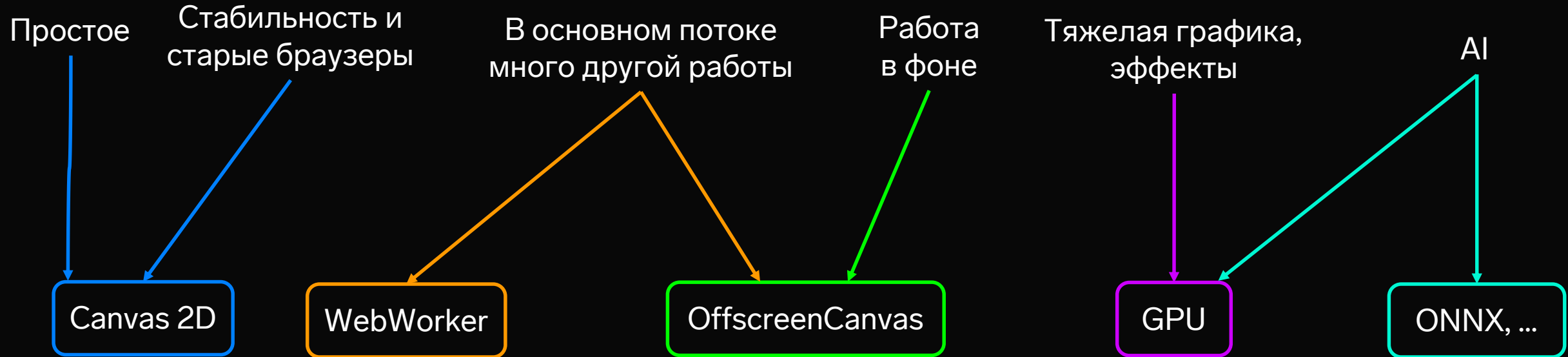
Out of Spec



VideoTrackGenerator

W3C Draft

Так как же обрабатывать видео?



Пользуйтесь профилированием

Спасибо за внимание!

Материалы, ссылки



Оценить доклад



Вопросы?

Контур