CocoaHeads Aachen

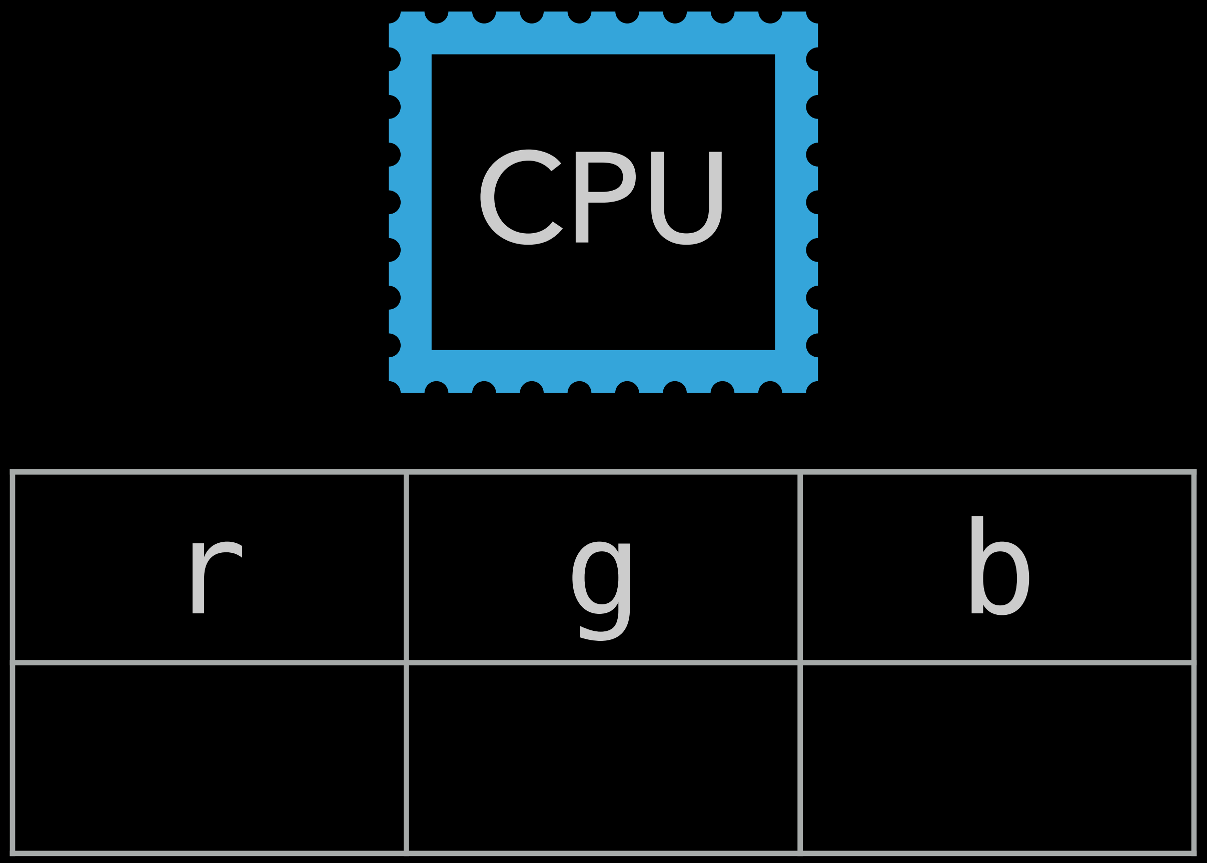
Alex Hoppen

August '19

Metal for Beginners

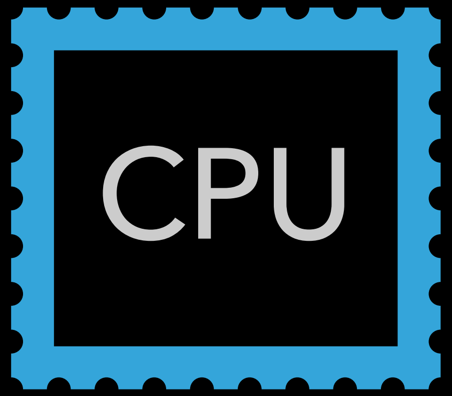
GPUS

Recap CPU



```
redTint(image, pos) {
  (r, g, b) = image[pos]
  r = r * 2
  if (r > 255) {
    r = 255
  }
  image[pos] = (r, g, b)
}
```

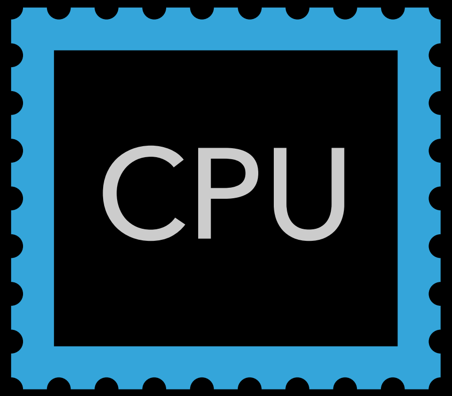
Recap Cpu



r	g	b
53	127	78

```
redTint(image, pos) {
  (r, g, b) = image[pos]
  r = r * 2
  if (r > 255) {
    r = 255
  }
  image[pos] = (r, g, b)
}
```

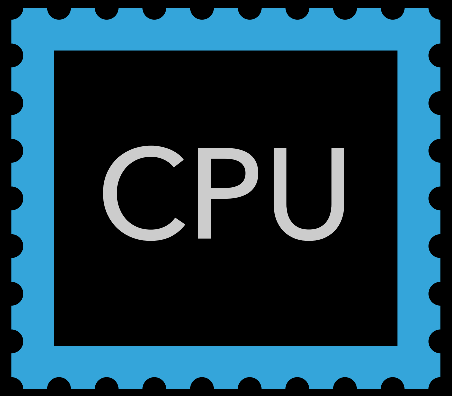
Recap CPU



r	g	b
53	127	78

```
redTint(image, pos) {
  (r, g, b) = image[pos]
  r = r * 2
  if (r > 255) {
    r = 255
  }
  image[pos] = (r, g, b)
}
```

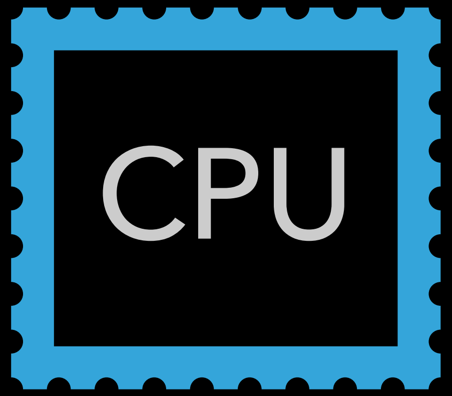
Recap Cpu



r	g	b
106	127	78

```
redTint(image, pos) {
  (r, g, b) = image[pos]
  r = r * 2
  if (r > 255) {
    r = 255
  }
  image[pos] = (r, g, b)
}
```

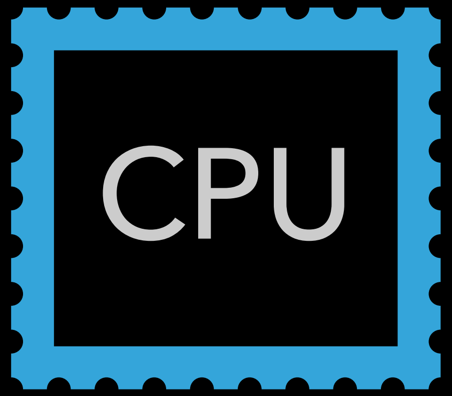
RECAP CPU



r	g	b
106	127	78

```
redTint(image, pos) {
  (r, g, b) = image[pos]
  r = r * 2
  if (r > 255) {
    r = 255
  }
  image[pos] = (r, g, b)
}
```

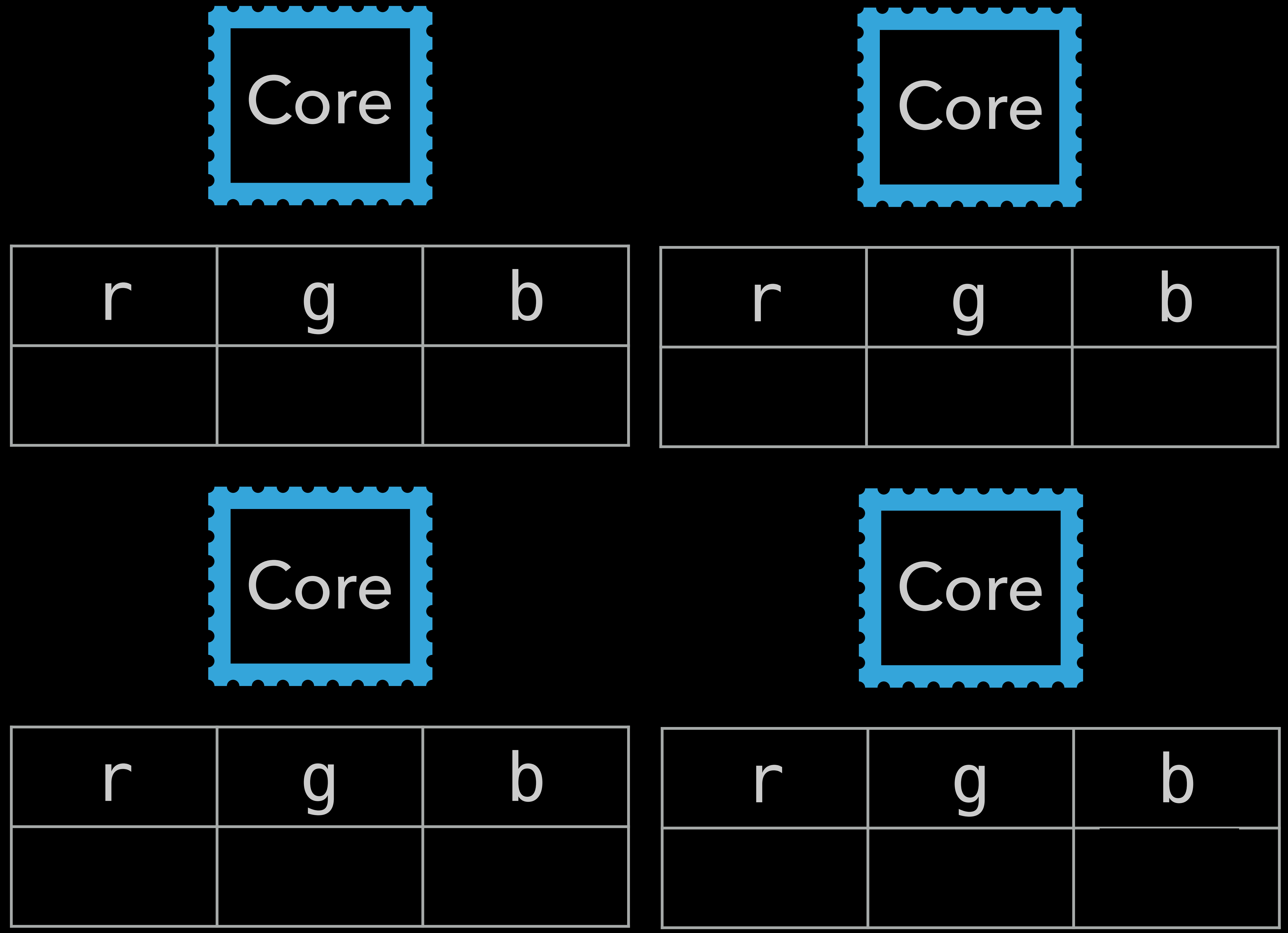
Recap CPU



r	g	b
106	127	78

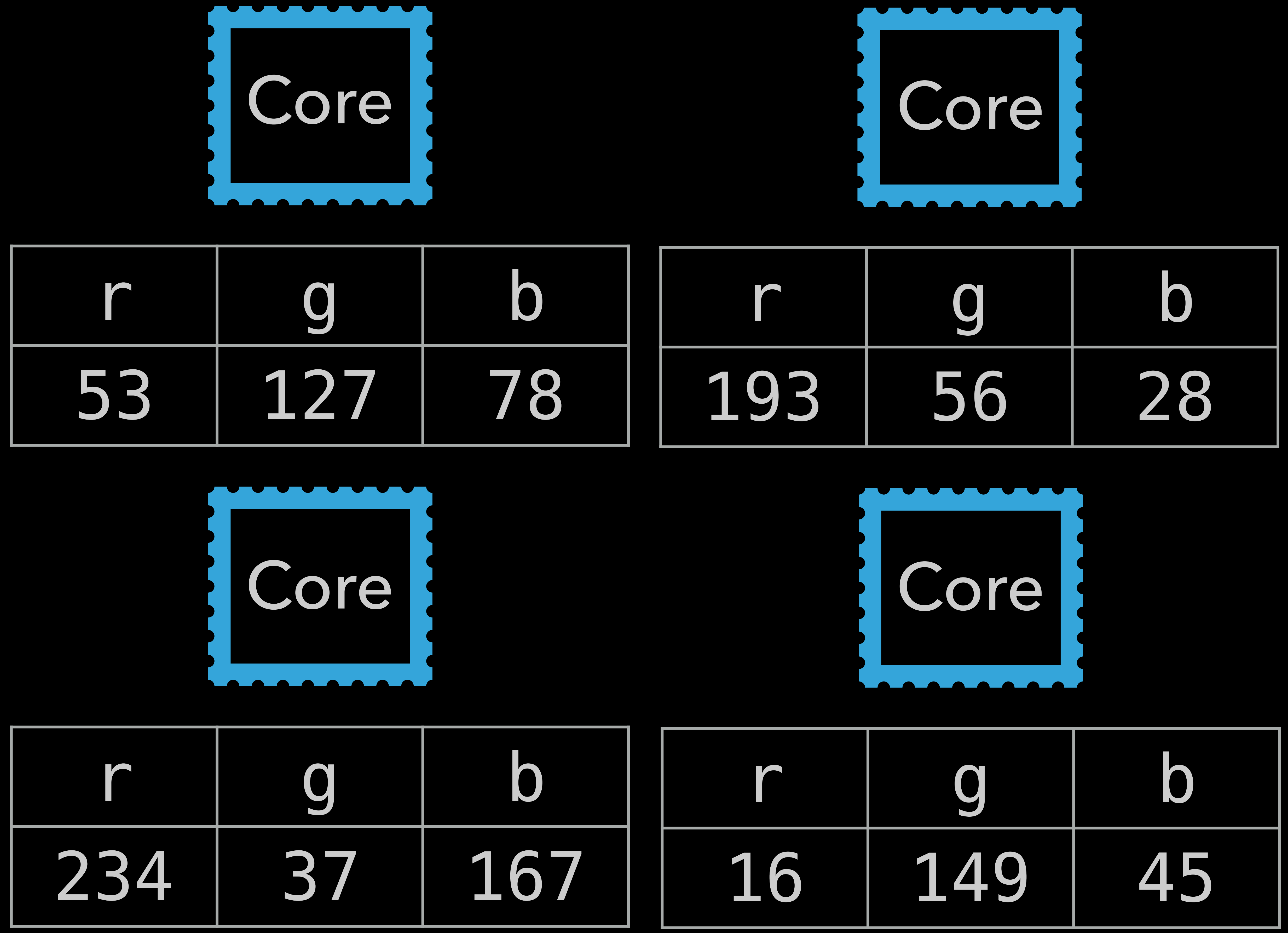
```
redTint(image, pos) {
  (r, g, b) = image[pos]
  r = r * 2
  if (r > 255) {
    r = 255
  }
  image[pos] = (r, g, b)
}
```

GPU Architecture



```
redTint(image, pos) {
  (r, g, b) = image[pos]
  r = r * 2
  if (r > 255) {
    r = 255
  }
  image[pos] = (r, g, b)
}
```

GPU ARCHITECTURE



r	g	b
53	127	78

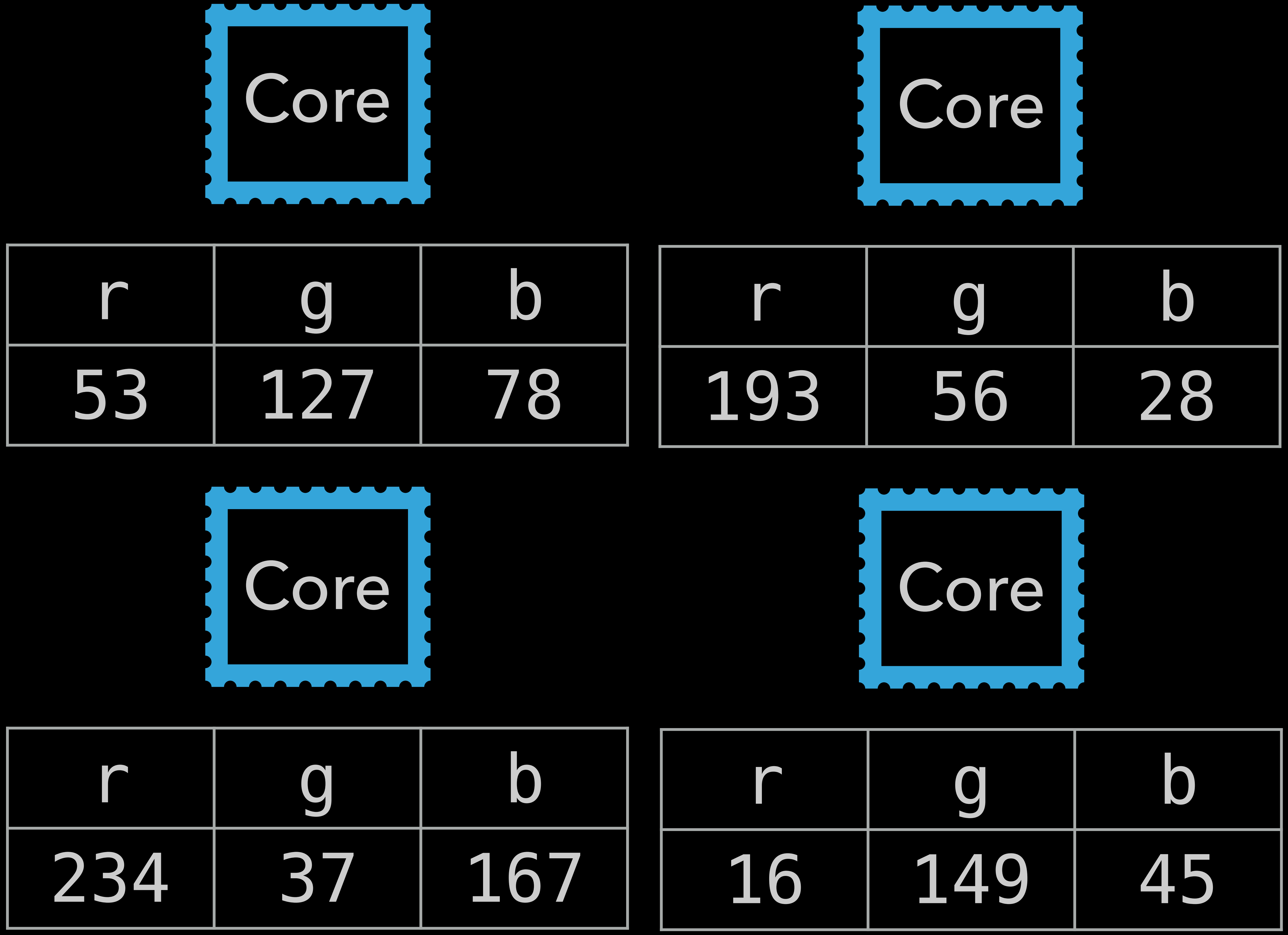
r	g	b
193	56	28

r	g	b
234	37	167

r	g	b
16	149	45

```
redTint(image, pos) {
  (r, g, b) = image[pos]
  r = r * 2
  if (r > 255) {
    r = 255
  }
  image[pos] = (r, g, b)
}
```

GPU Architecture



r	g	b
53	127	78

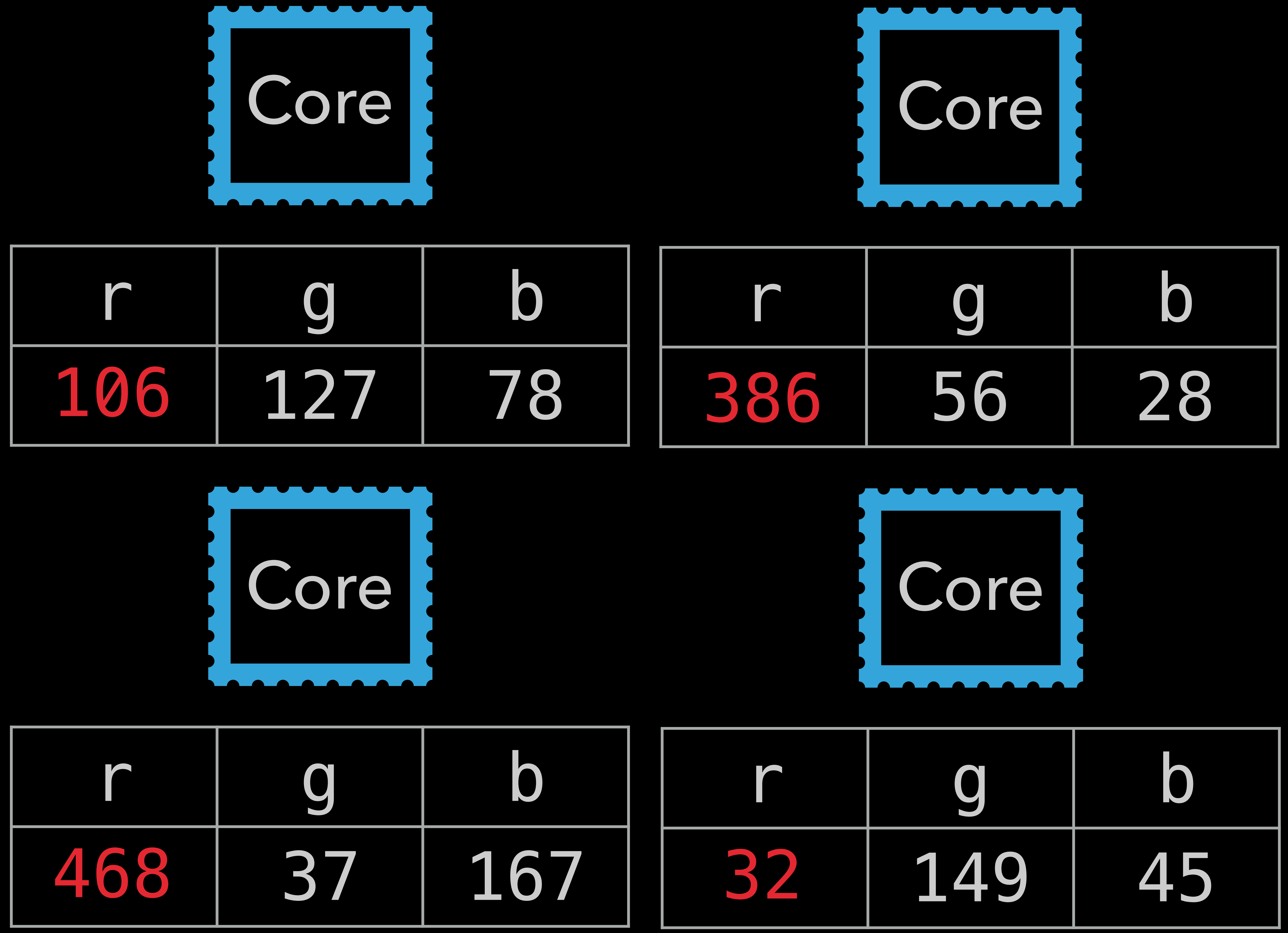
r	g	b
193	56	28

r	g	b
234	37	167

r	g	b
16	149	45

```
redTint(image, pos) {
  (r, g, b) = image[pos]
  r = r * 2
  if (r > 255) {
    r = 255
  }
  image[pos] = (r, g, b)
}
```

GPU Architecture



r	g	b
106	127	78

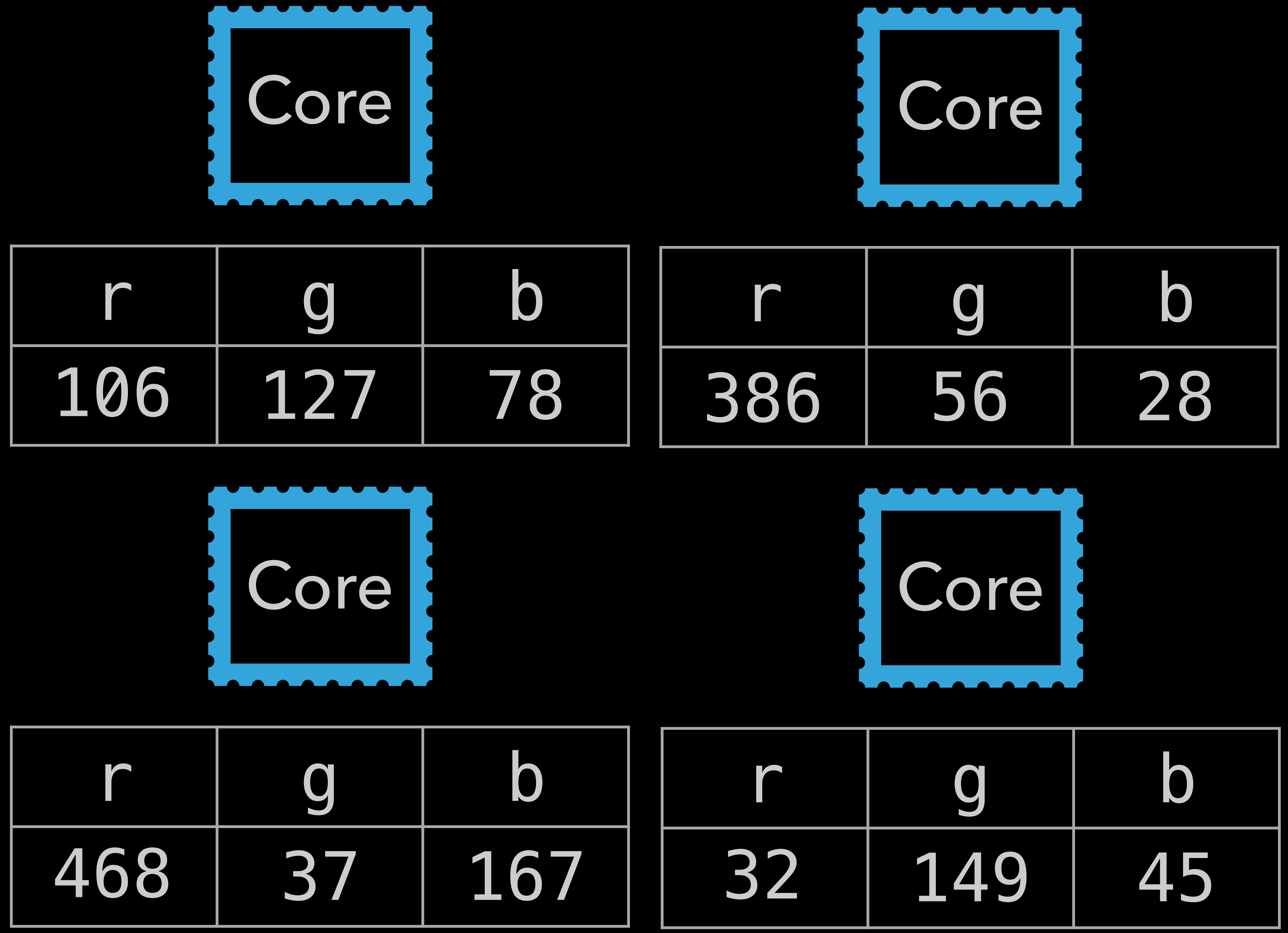
r	g	b
386	56	28

r	g	b
468	37	167

r	g	b
32	149	45

```
redTint(image, pos) {
  (r, g, b) = image[pos]
  r = r * 2
  if (r > 255) {
    r = 255
  }
  image[pos] = (r, g, b)
}
```

GPU Architecture



r	g	b
106	127	78

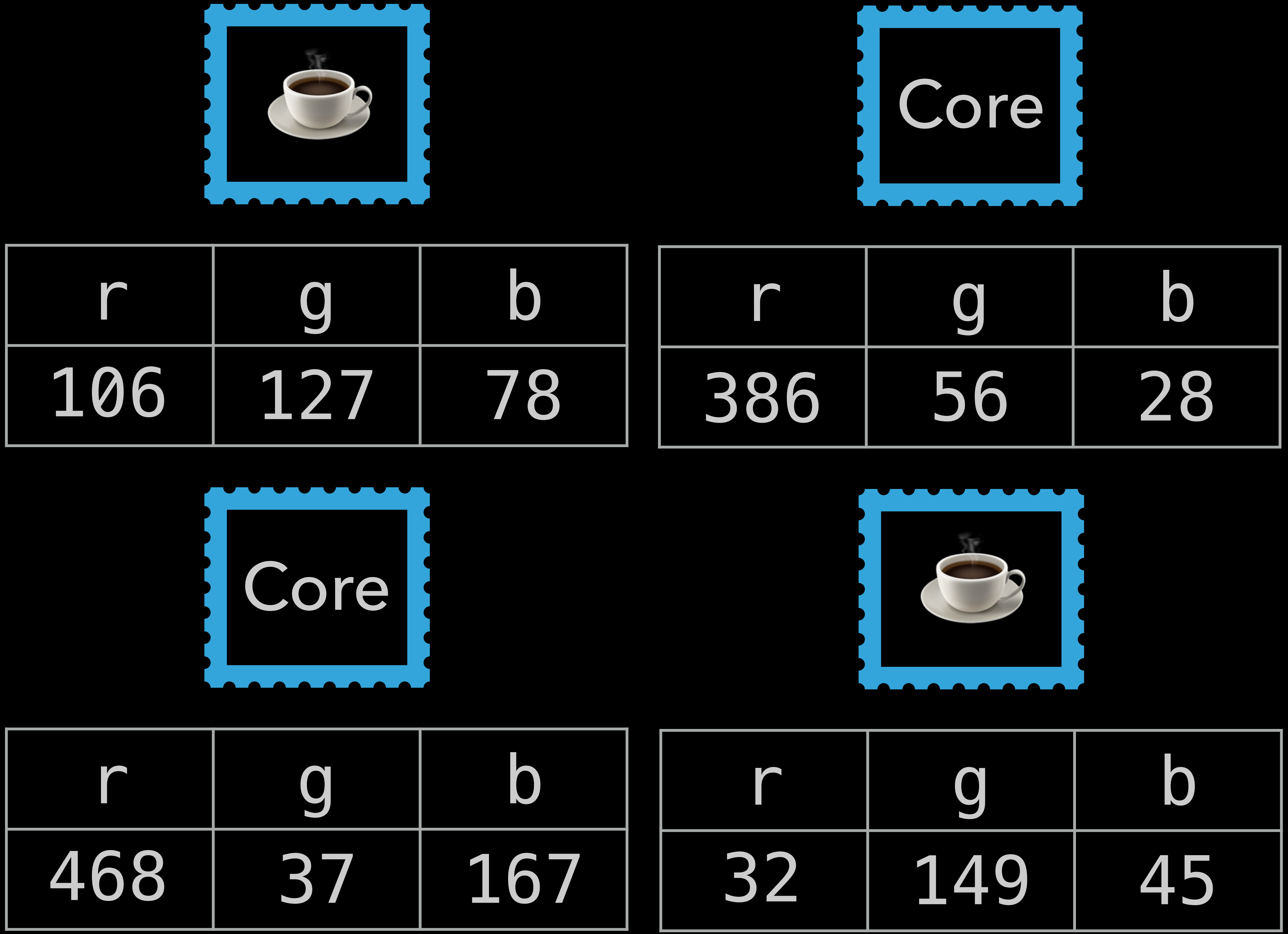
r	g	b
386	56	28

r	g	b
468	37	167

r	g	b
32	149	45

```
redTint(image, pos) {
  (r, g, b) = image[pos]
  r = r * 2
  if (r > 255) {
    r = 255
  }
  image[pos] = (r, g, b)
}
```

GPU ARCHITECTURE



r	g	b
106	127	78

r	g	b
386	56	28

r	g	b
468	37	167

r	g	b
32	149	45

```
redTint(image, pos) {
  (r, g, b) = image[pos]
  r = r * 2
  if (r > 255) {
    r = 255
  }
  image[pos] = (r, g, b)
}
```

GPU Architecture

☕

r	g	b
106	127	78

Core

r	g	b
386	56	28

Core

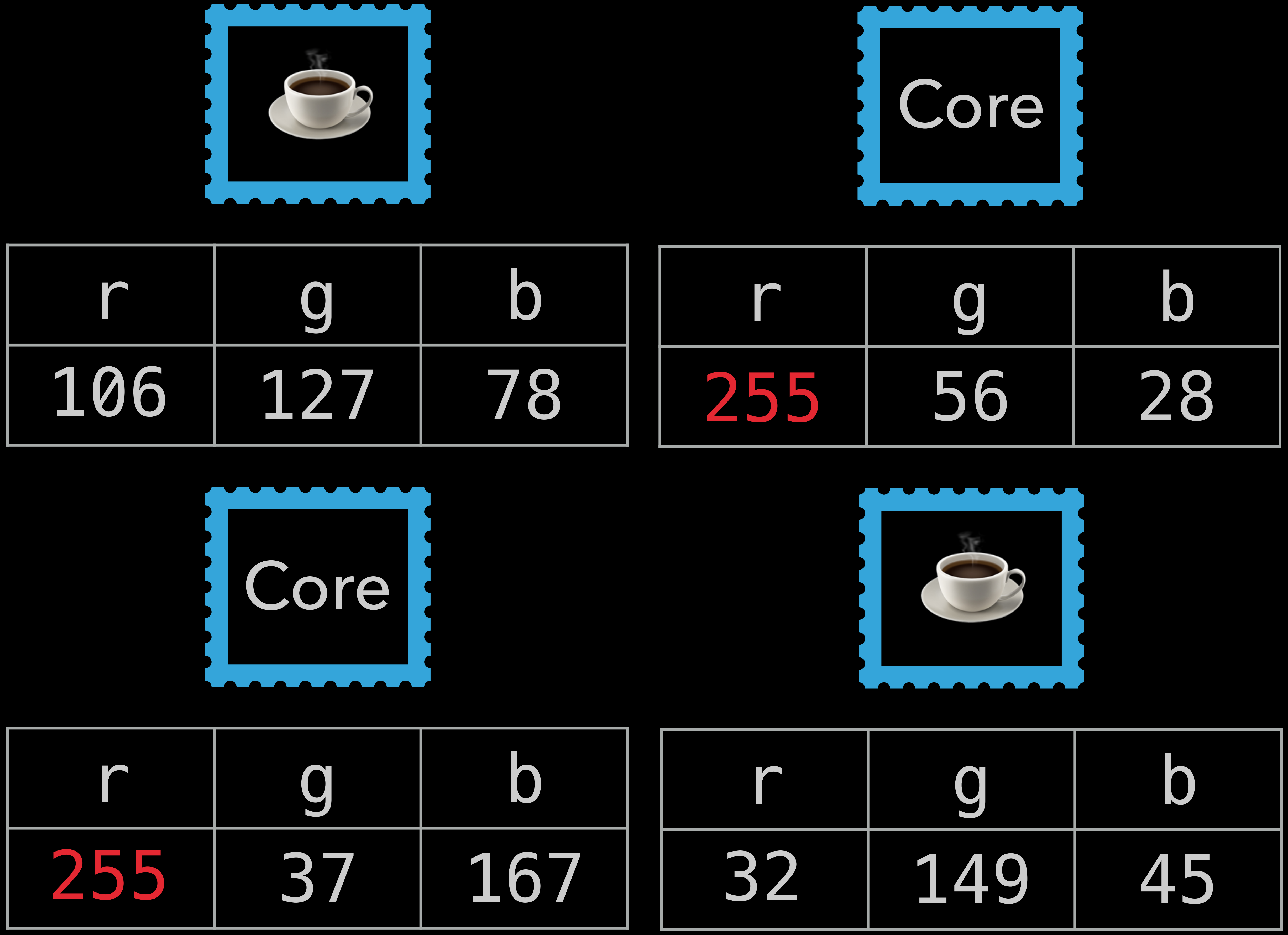
r	g	b
468	37	167

☕

r	g	b
32	149	45

```
redTint(image, pos) {
  (r, g, b) = image[pos]
  r = r * 2
  if (r > 255) {
    r = 255
  }
  image[pos] = (r, g, b)
}
```

GPU Architecture



r	g	b
106	127	78

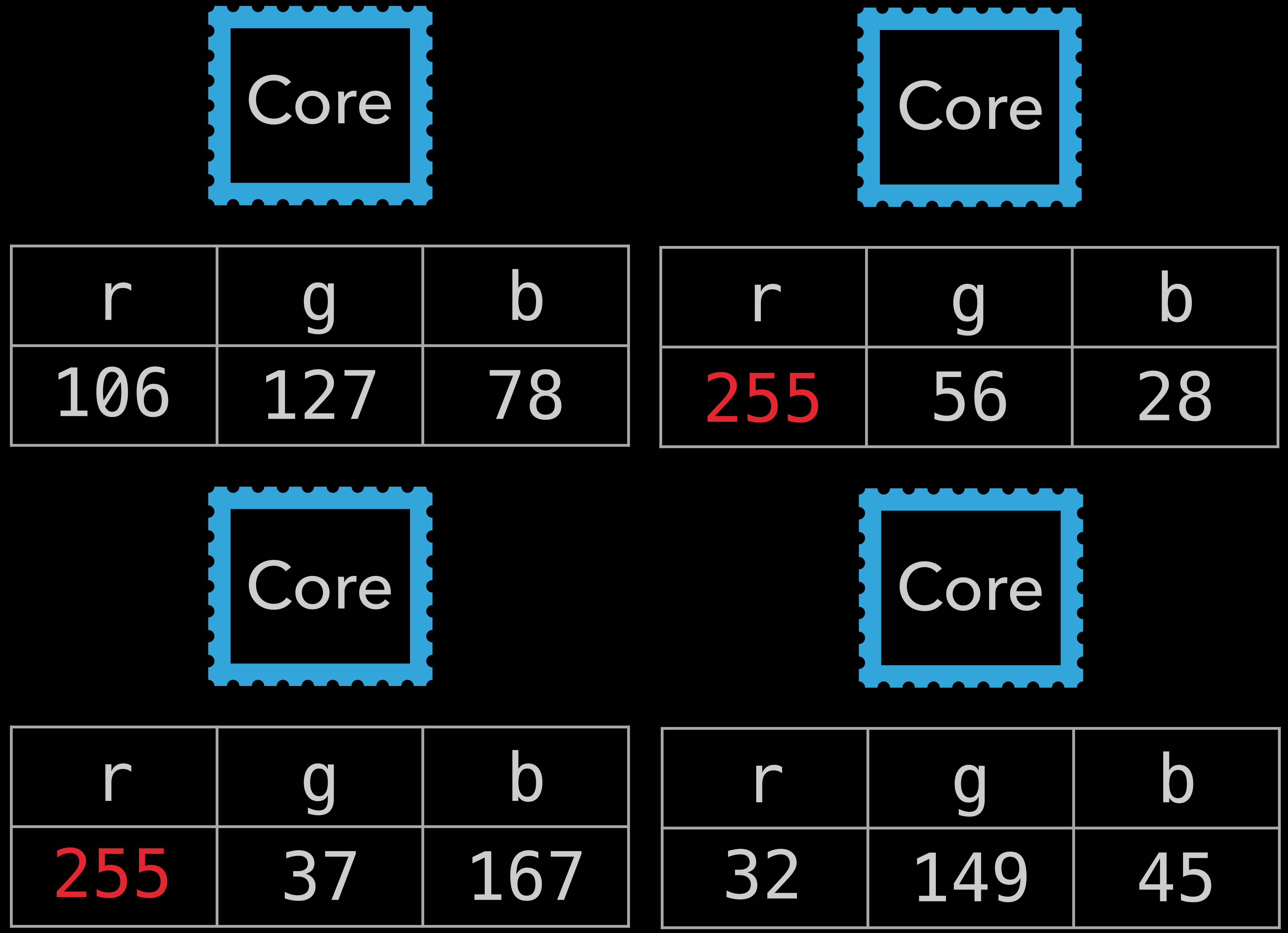
r	g	b
255	56	28

r	g	b
255	37	167

r	g	b
32	149	45

```
redTint(image, pos) {
  (r, g, b) = image[pos]
  r = r * 2
  if (r > 255) {
    r = 255
  }
  image[pos] = (r, g, b)
}
```

GPU Architecture



r	g	b
106	127	78

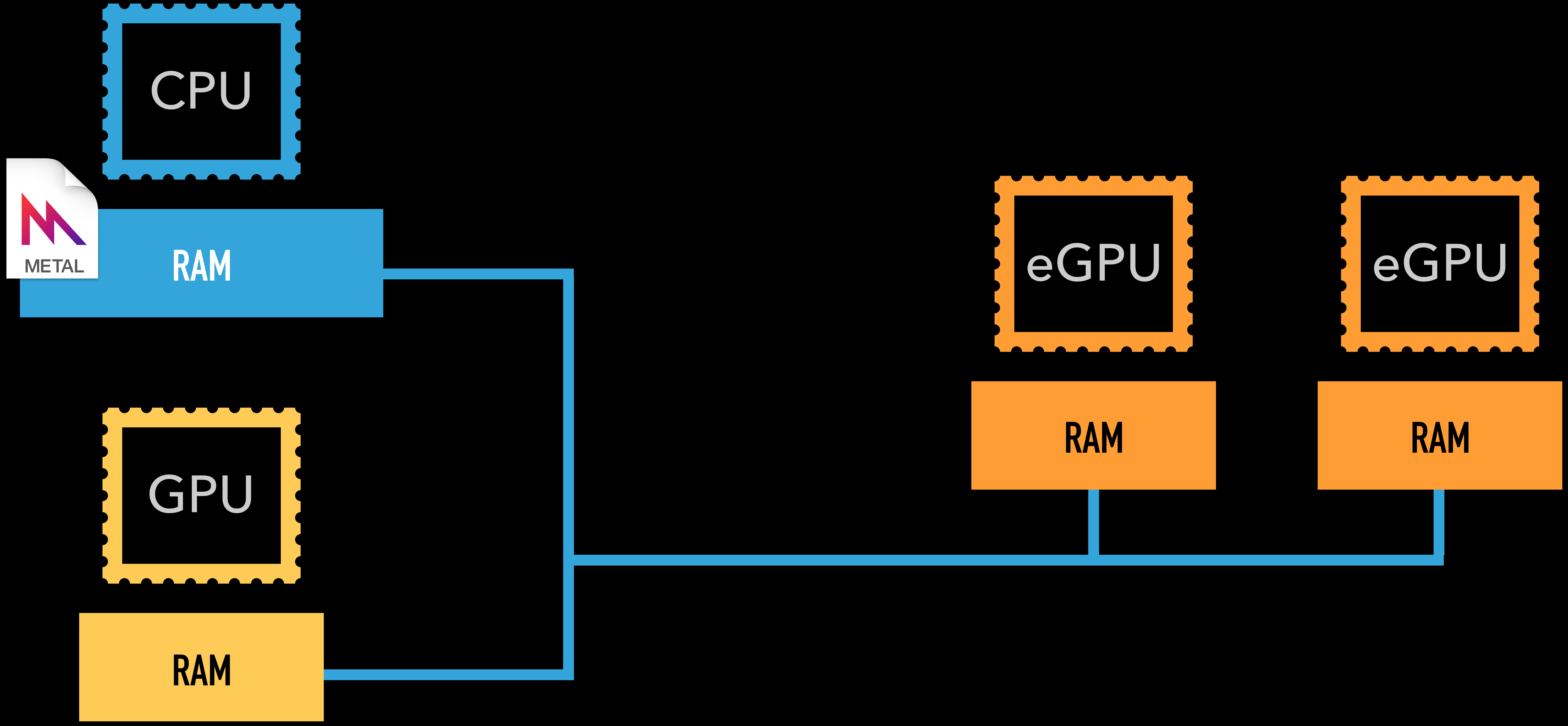
r	g	b
255	56	28

r	g	b
255	37	167

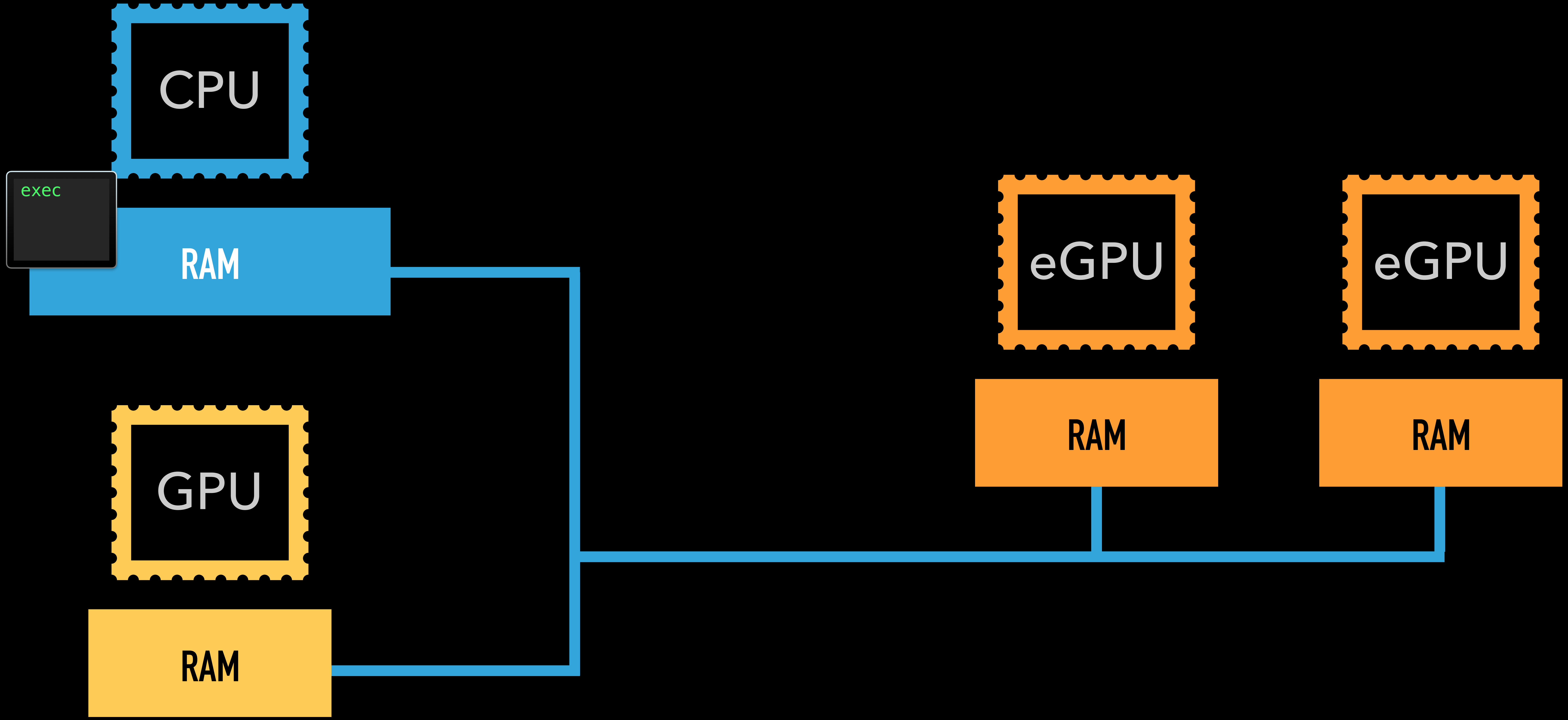
r	g	b
32	149	45

```
redTint(image, pos) {
  (r, g, b) = image[pos]
  r = r * 2
  if (r > 255) {
    r = 255
  }
  image[pos] = (r, g, b)
}
```

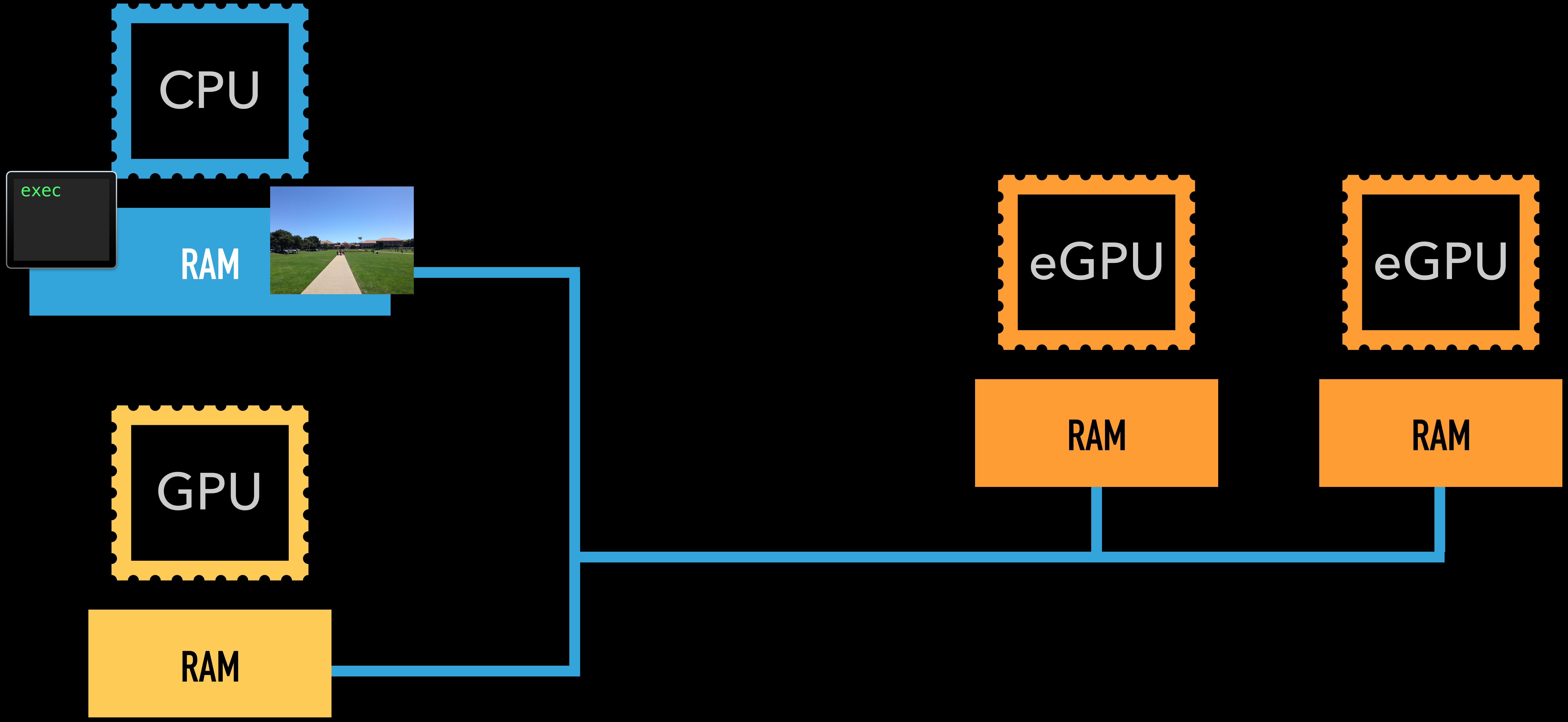
Graphics Structure on the Mac



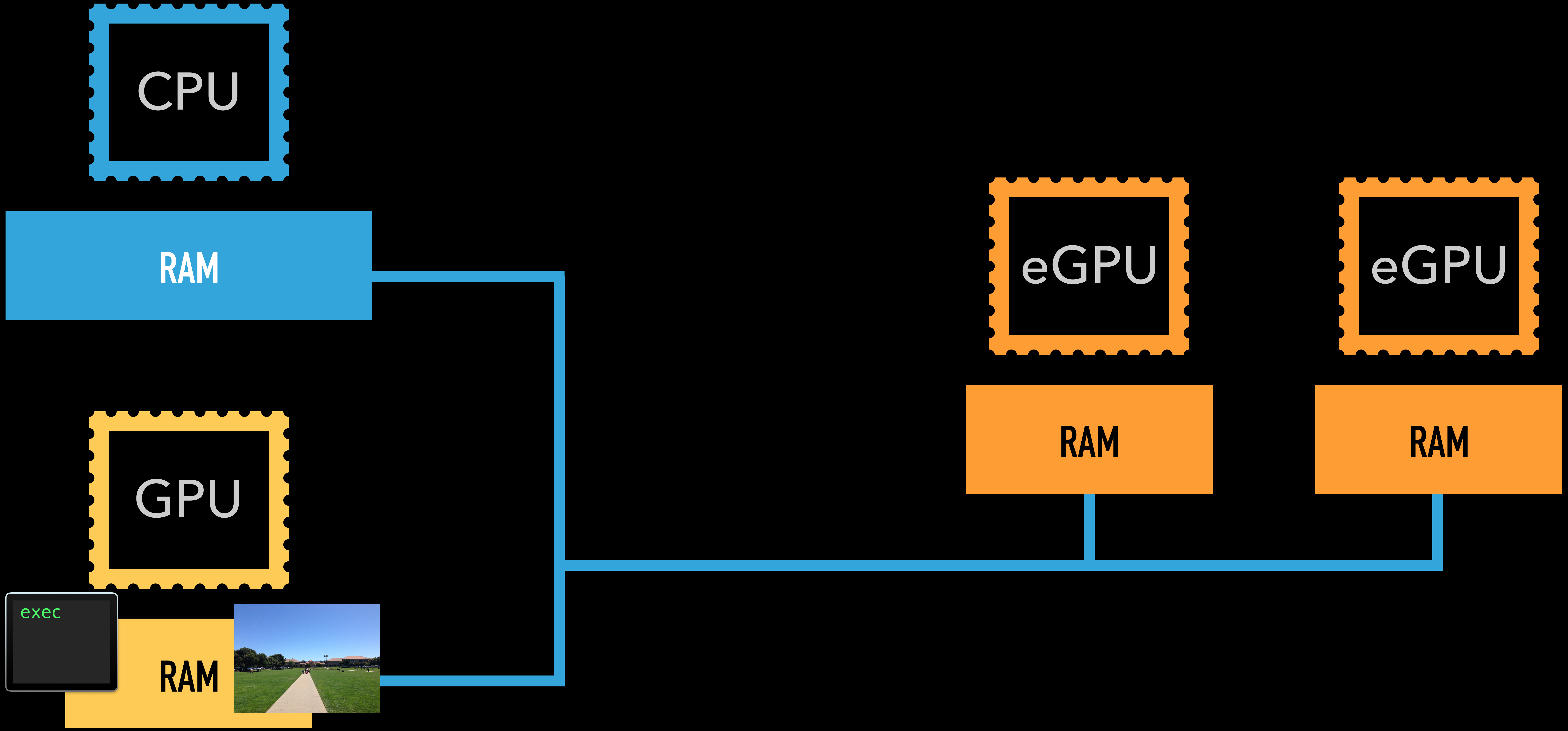
Graphics Structure on the Mac



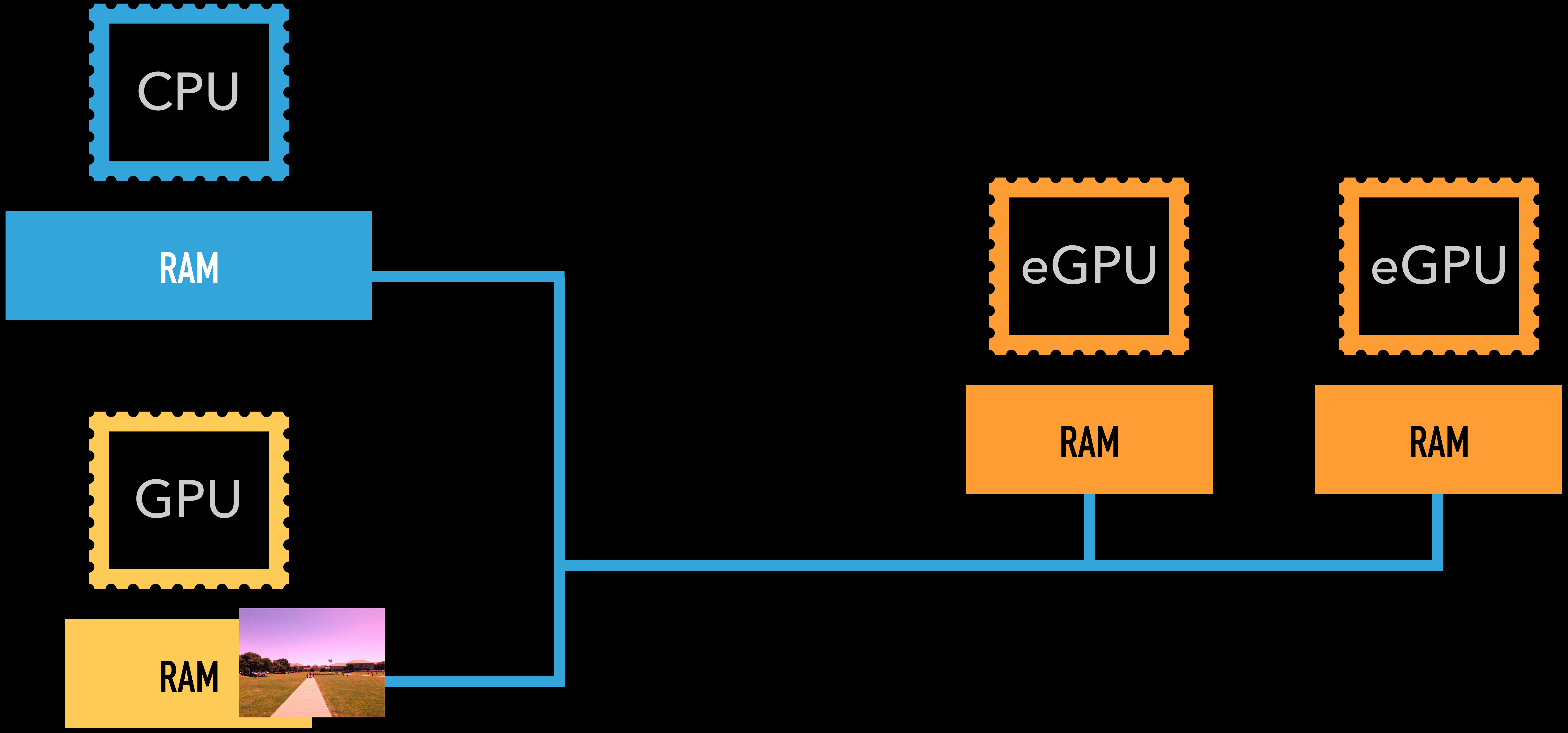
Graphics Structure on the Mac



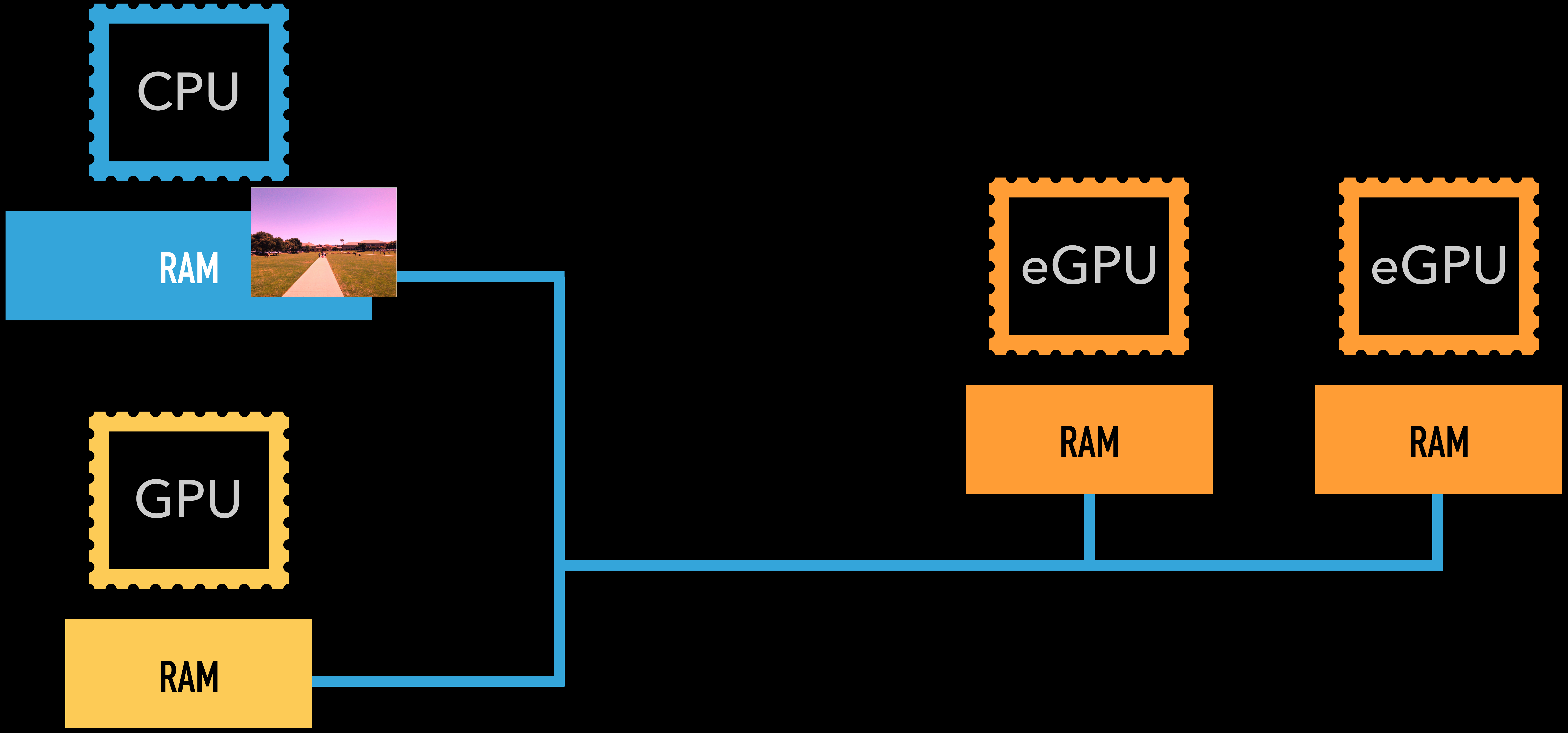
GRAPHICS STRUCTURE ON THE MAC



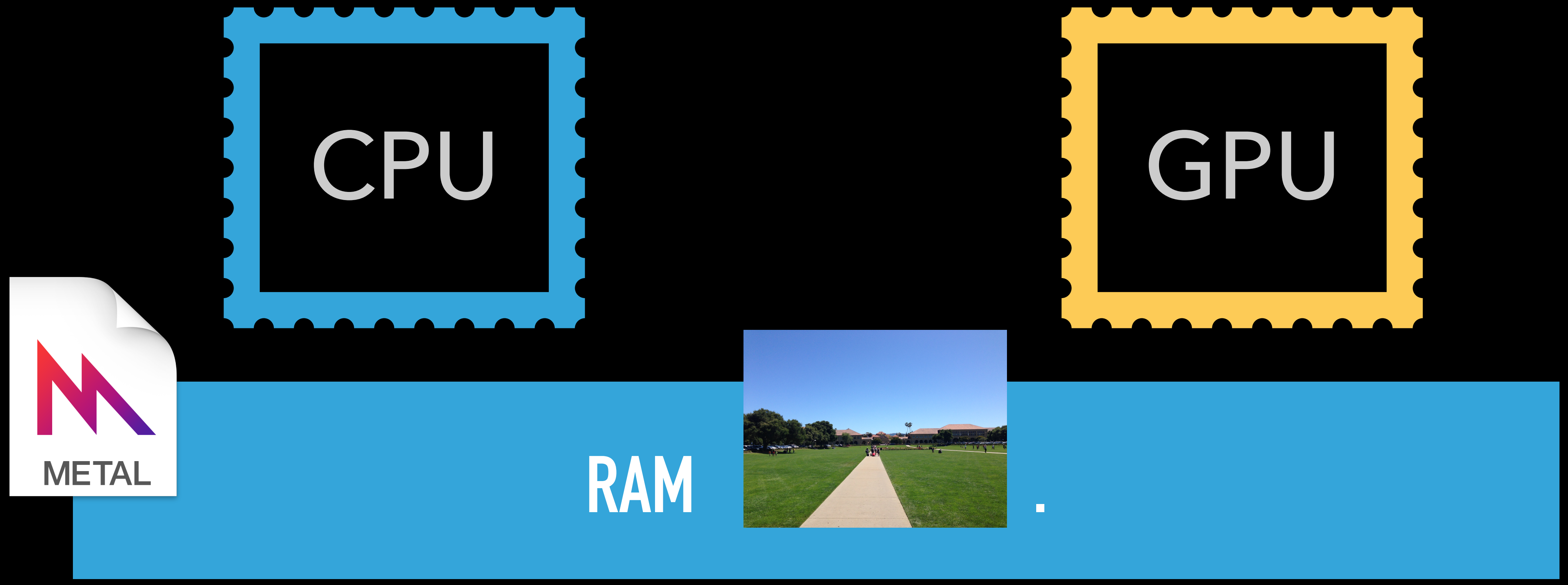
Graphics Structure on the Mac



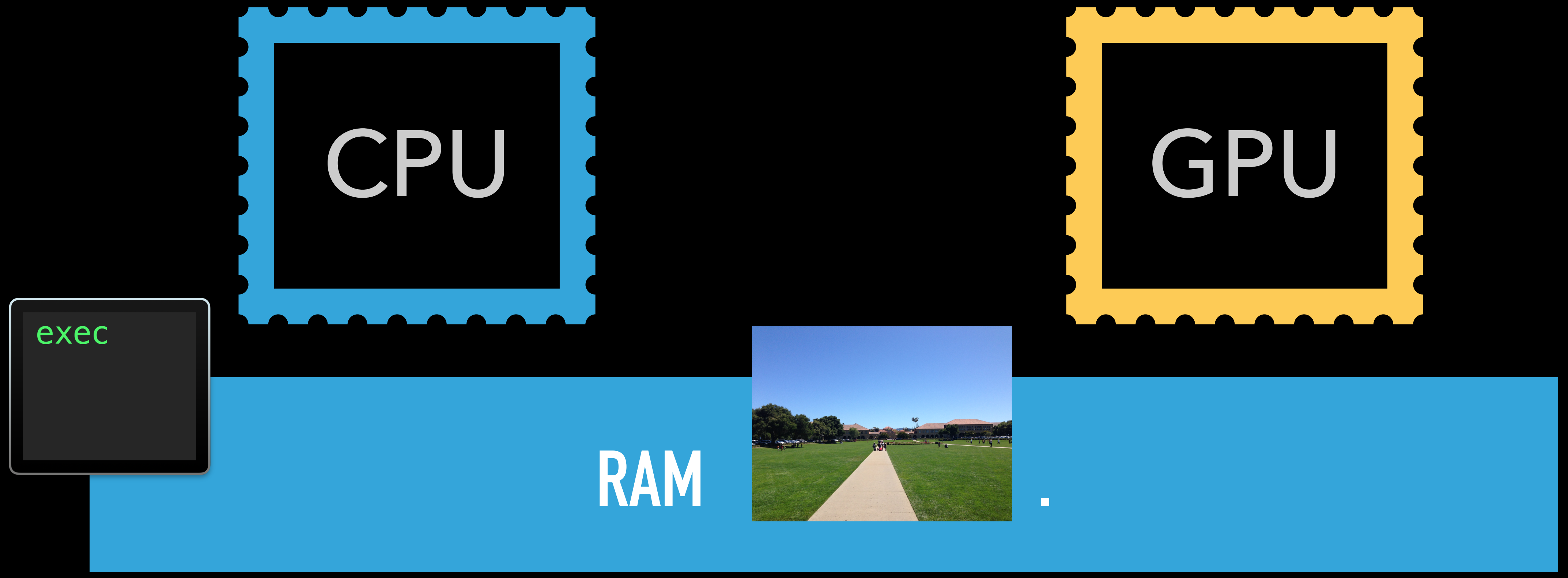
Graphics Structure on the Mac



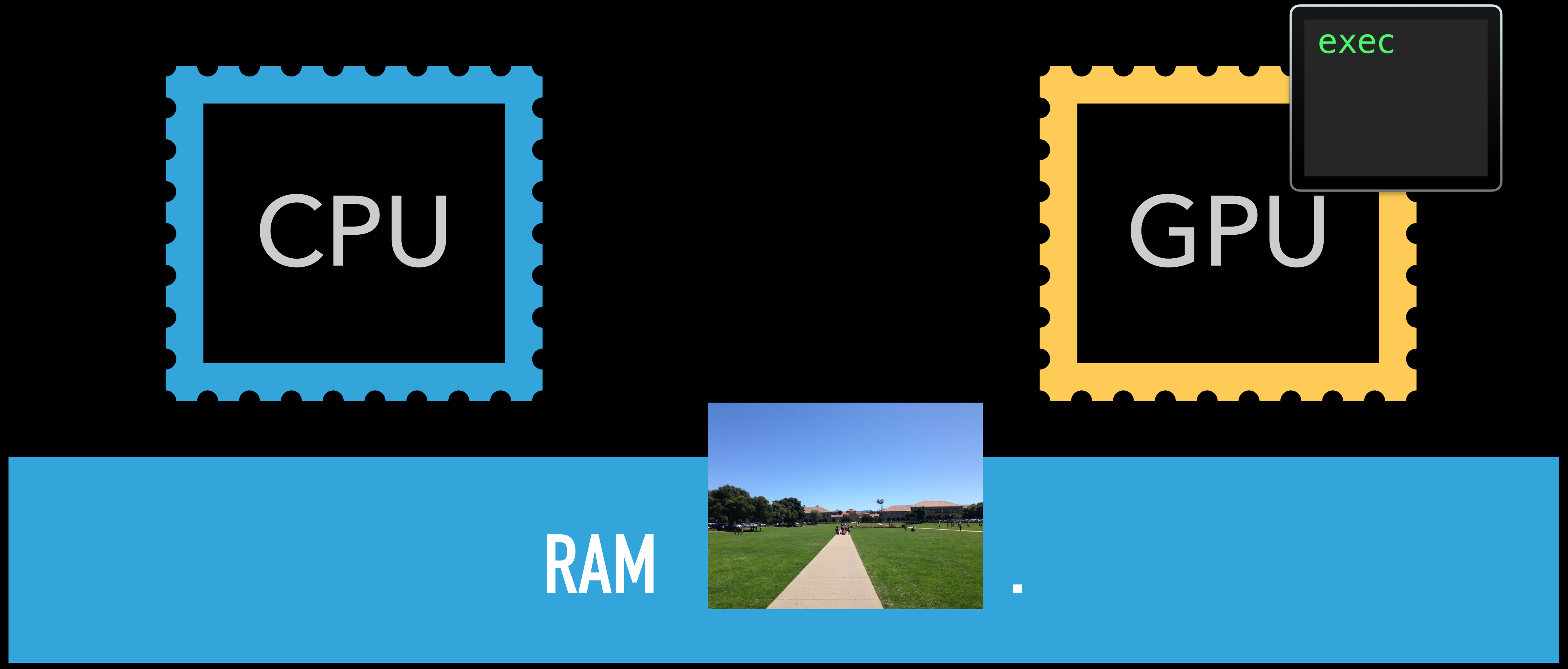
Graphics Structure on iOS (and Macs with Intel Graphics Card)



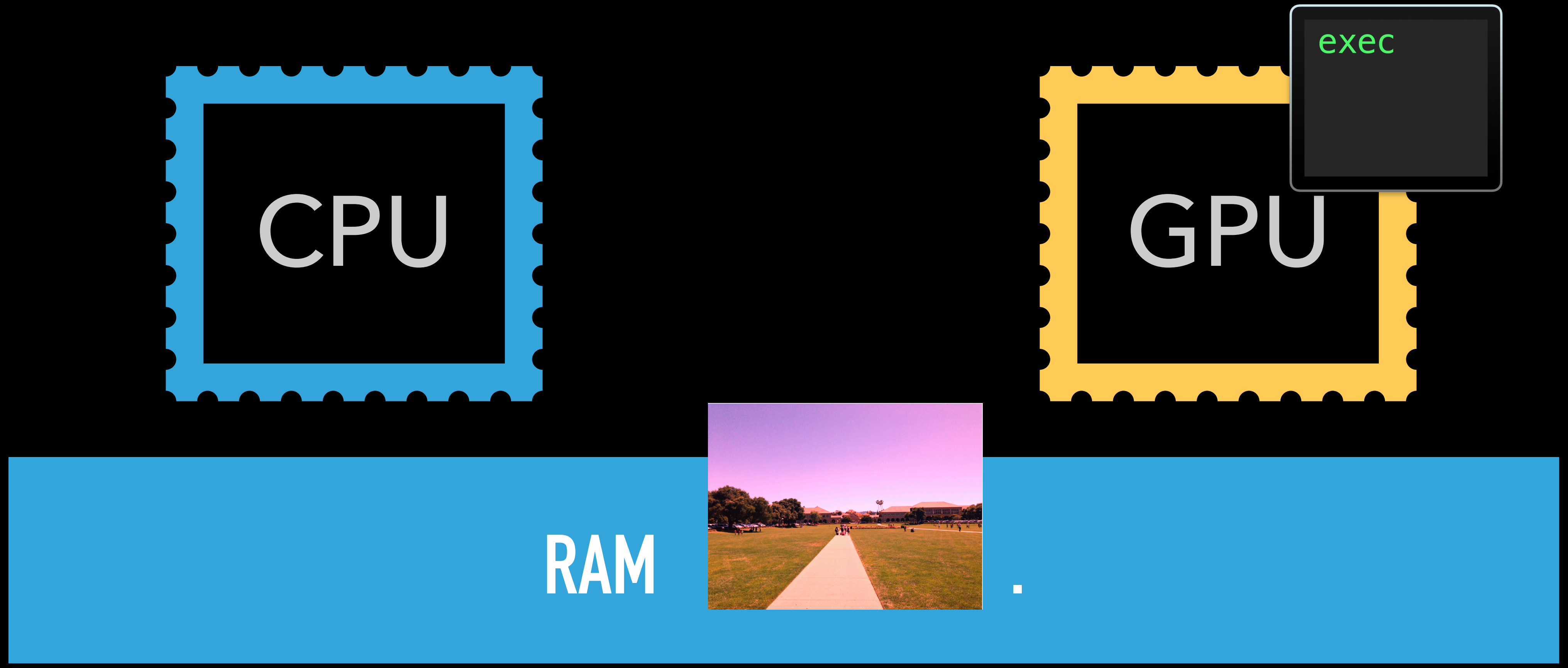
Graphics Structure on iOS (and Macs with Intel Graphics Card)



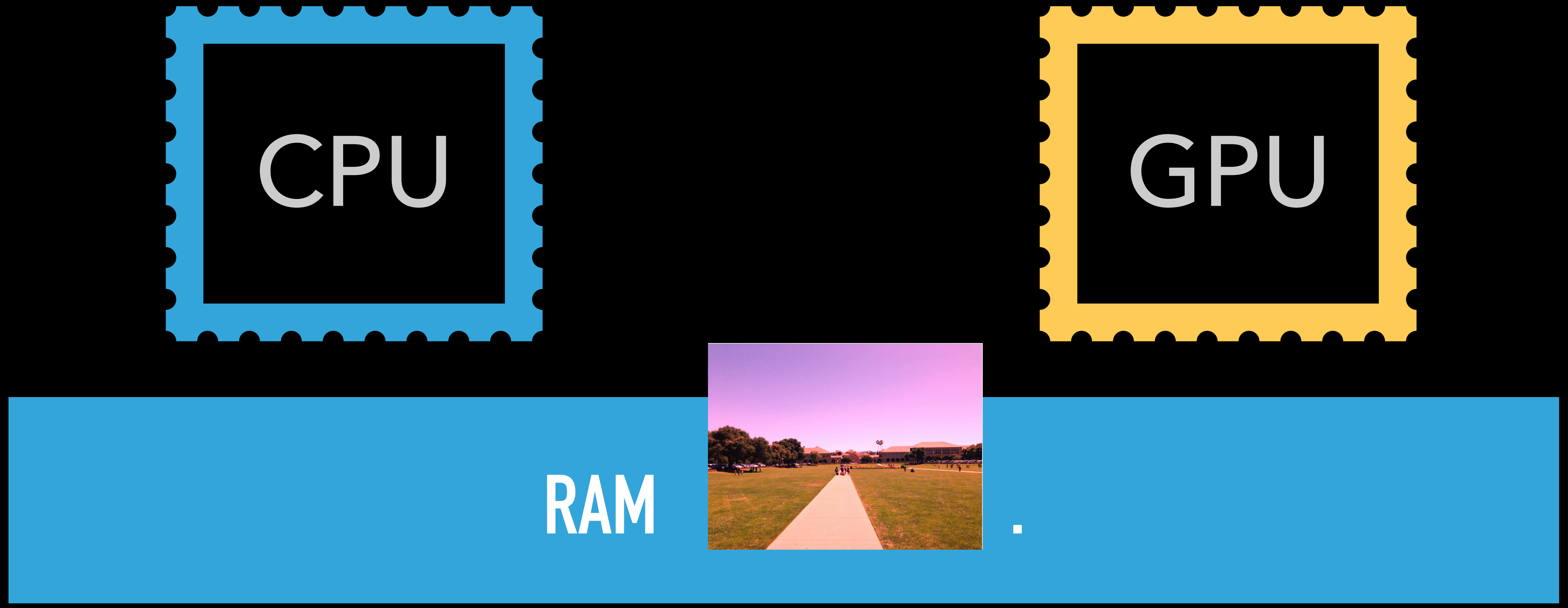
Graphics Structure on iOS (and Macs with Intel Graphics Card)



Graphics Structure on iOS (and Macs with Intel Graphics Card)

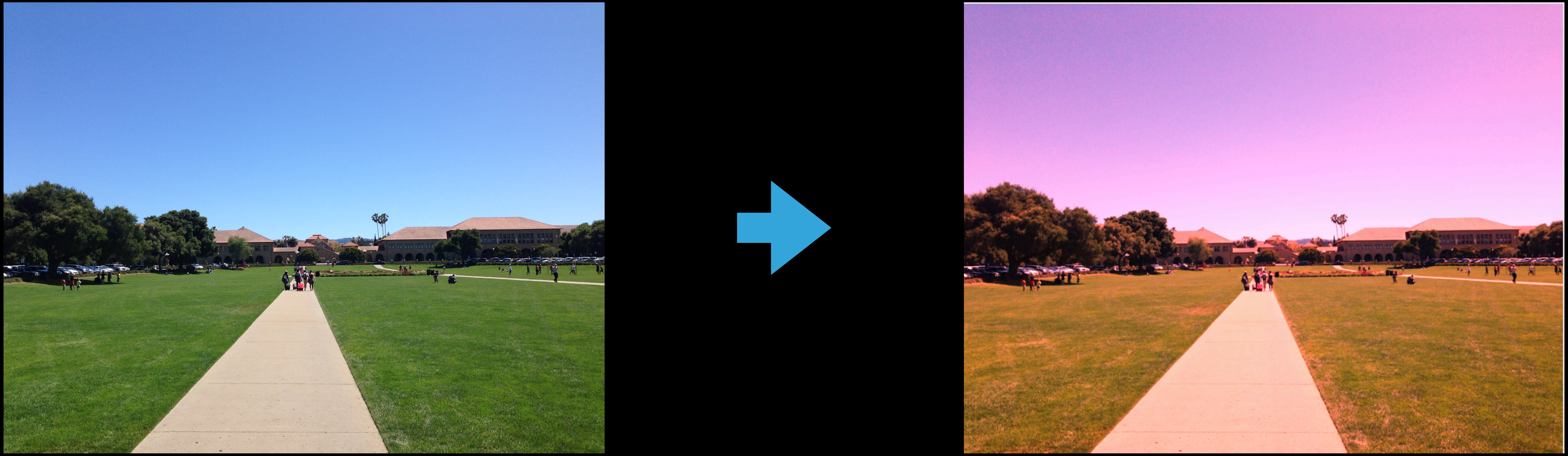


Graphics Structure on iOS (and Macs with Intel Graphics Card)



Metal

Our Goal



Our Plan

Our Plan

r	g	b	a
53	127	78	255

r	g	b	a
167	34	94	255

⋮

Our Plan

r	g	b	a
53	127	78	255

UInt32

r	g	b	a
167	34	94	255

●
●
●

Our Plan

r	g	b	a
53	127	78	255

UInt32

r	g	b	a
167	34	94	255

⋮

r	g	b	a
106	127	78	255

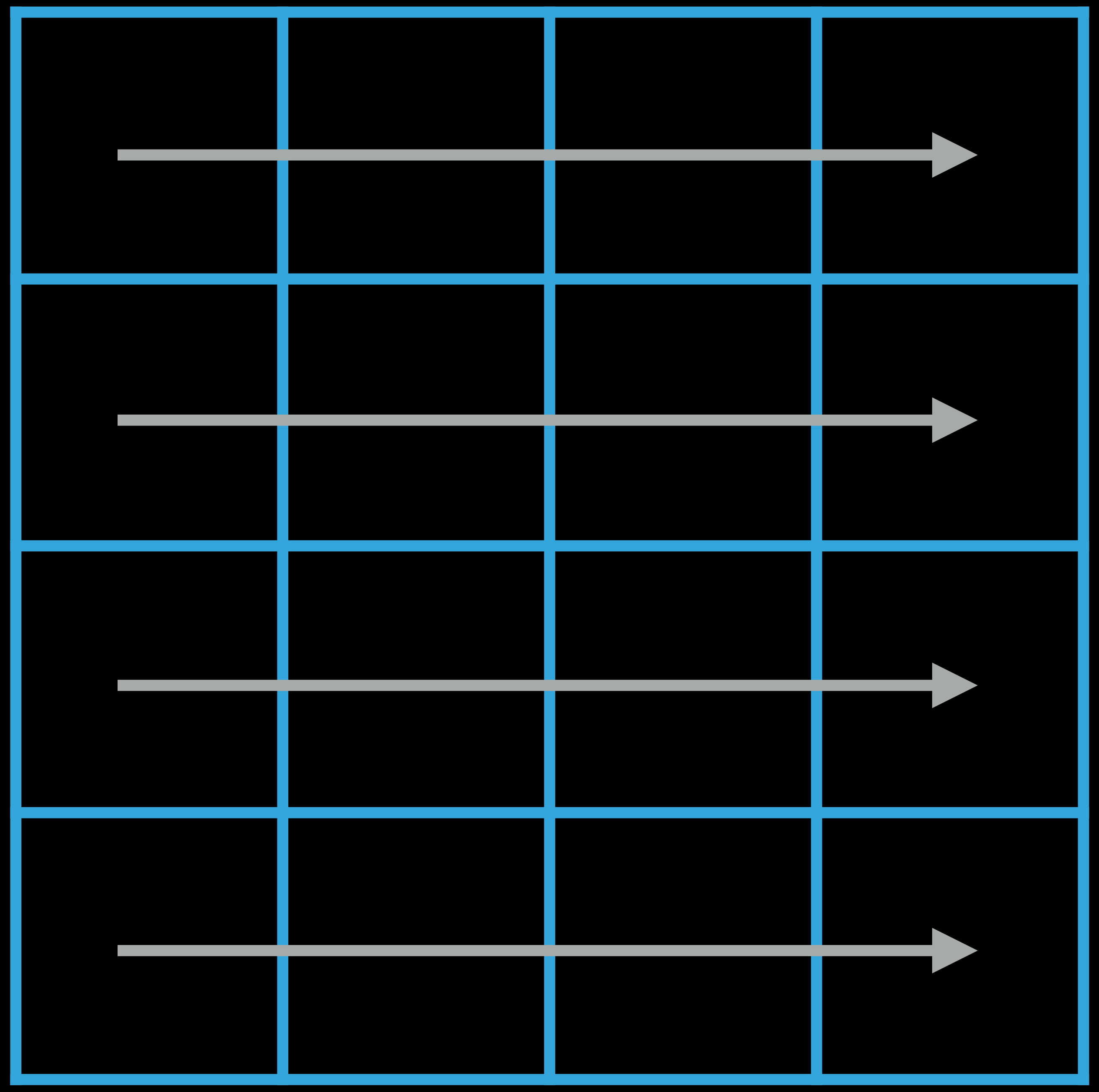
r	g	b	a
255	34	94	255

⋮

Metal with Compute Shaders

Demo

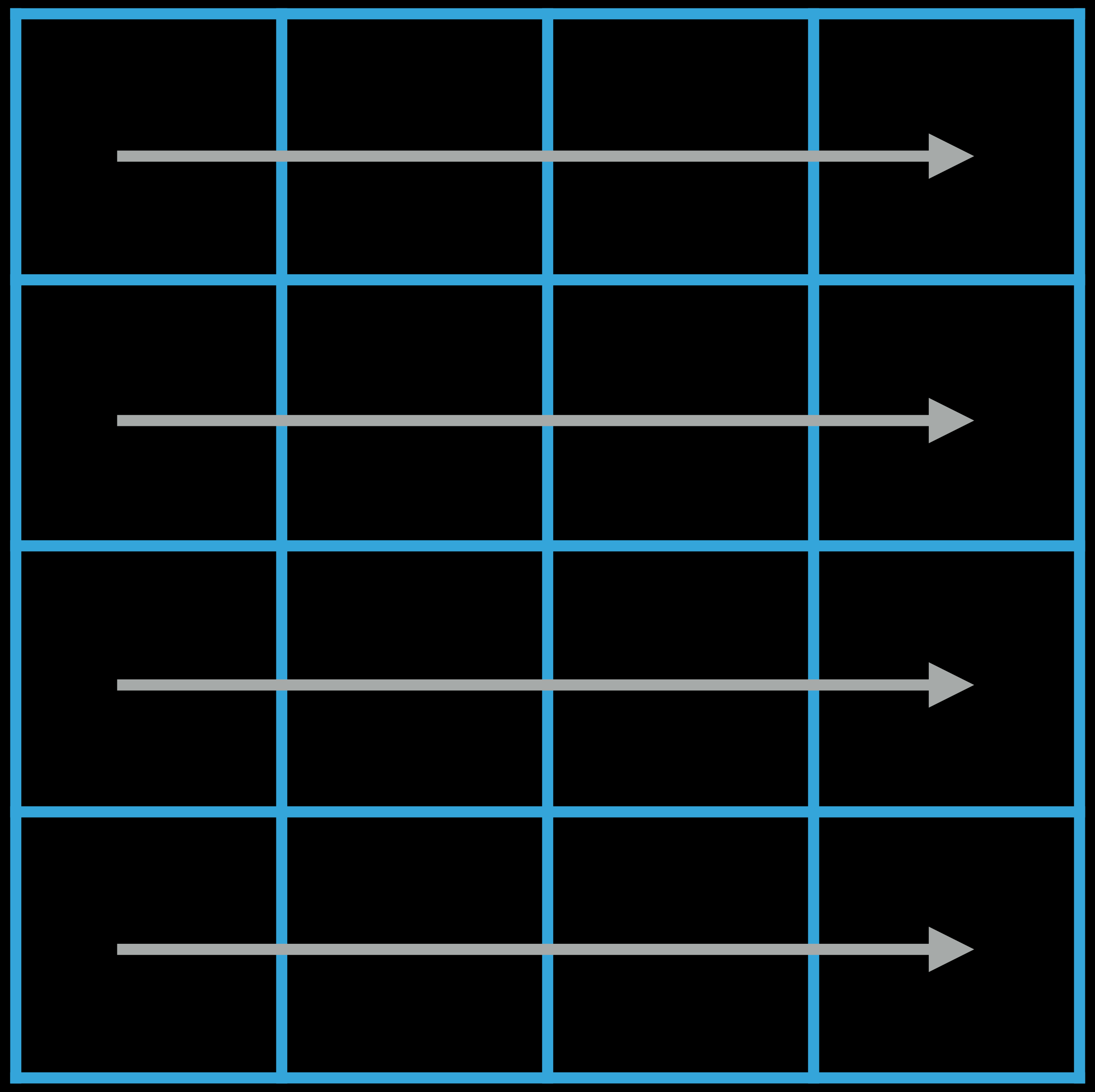
A Quick Note on Textures (and Cache Lines)

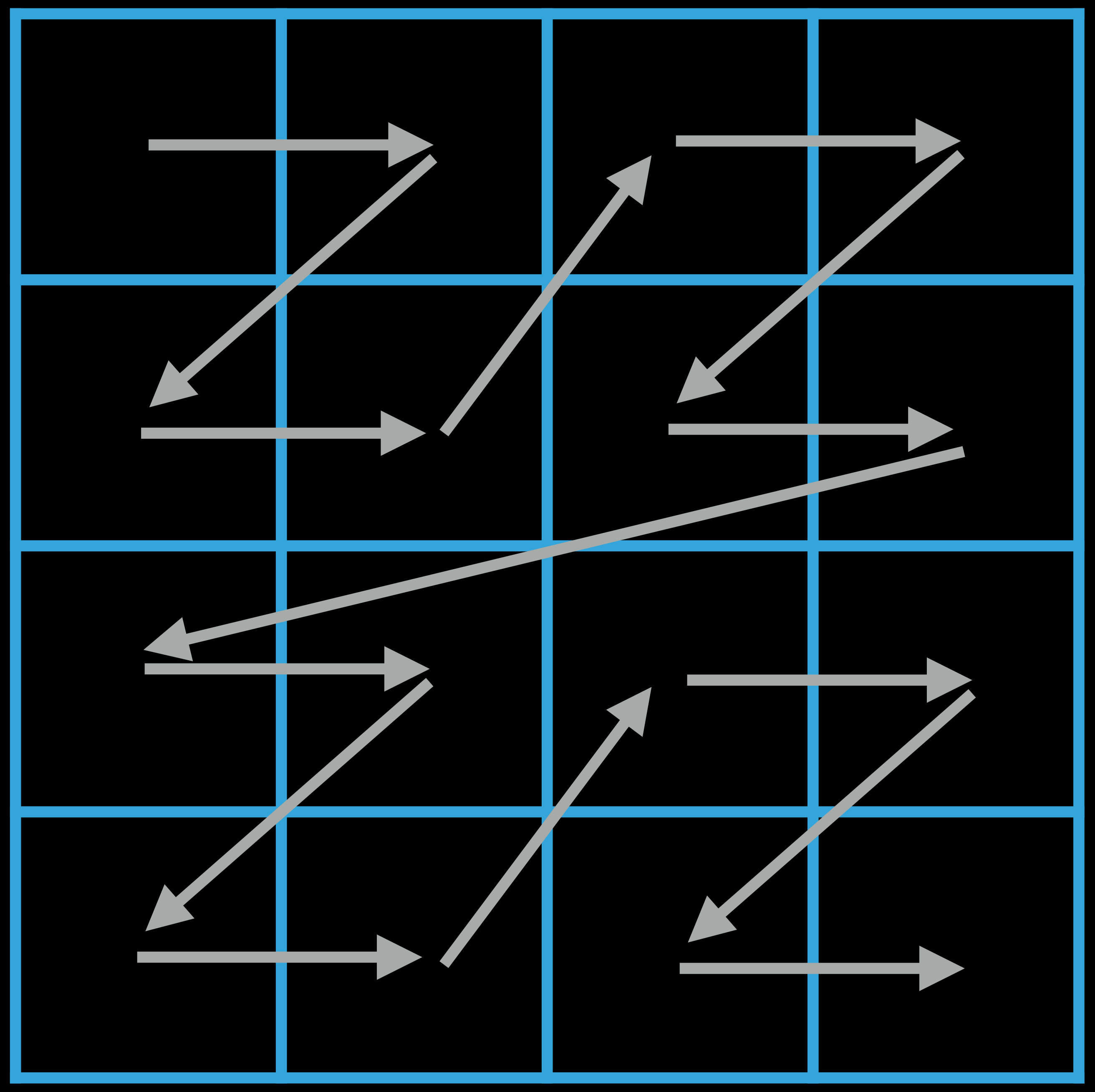


Raw Image Buffer

* or another similar pattern

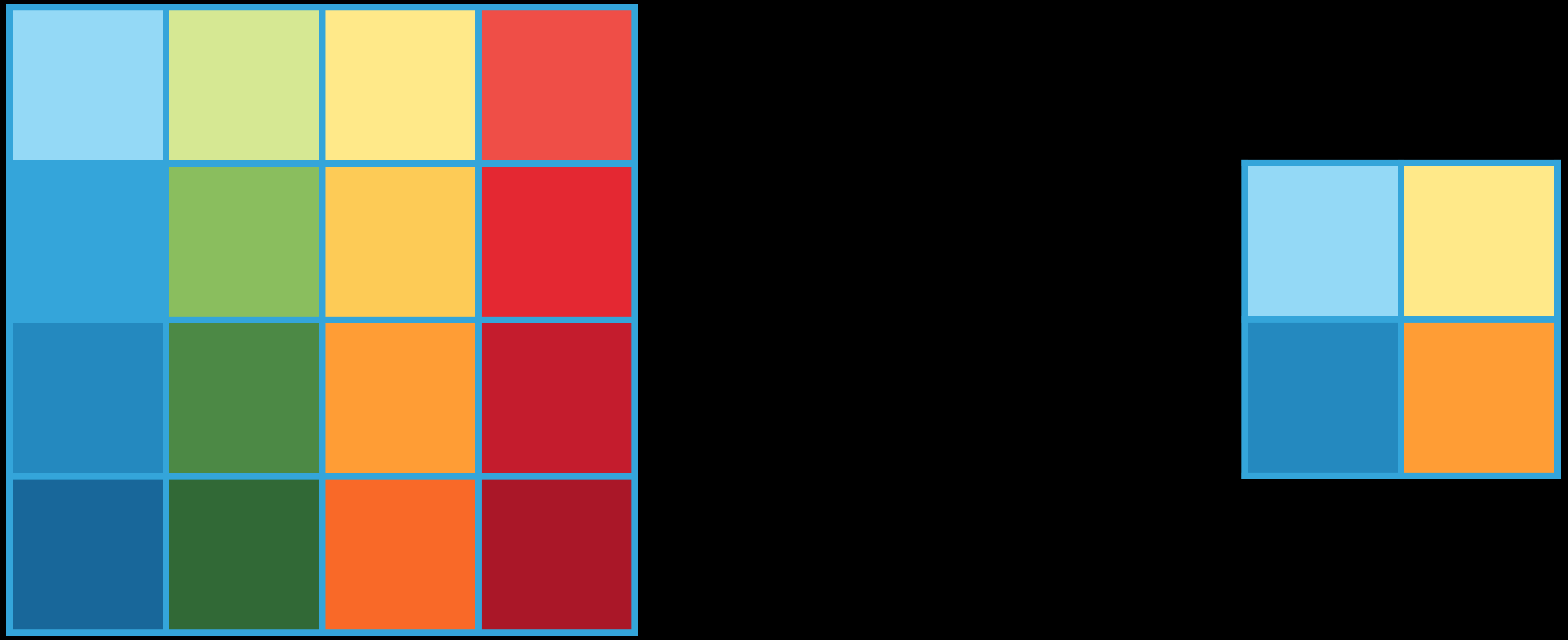
A Quick Note on Textures (and Cache Lines)


Raw Image Buffer

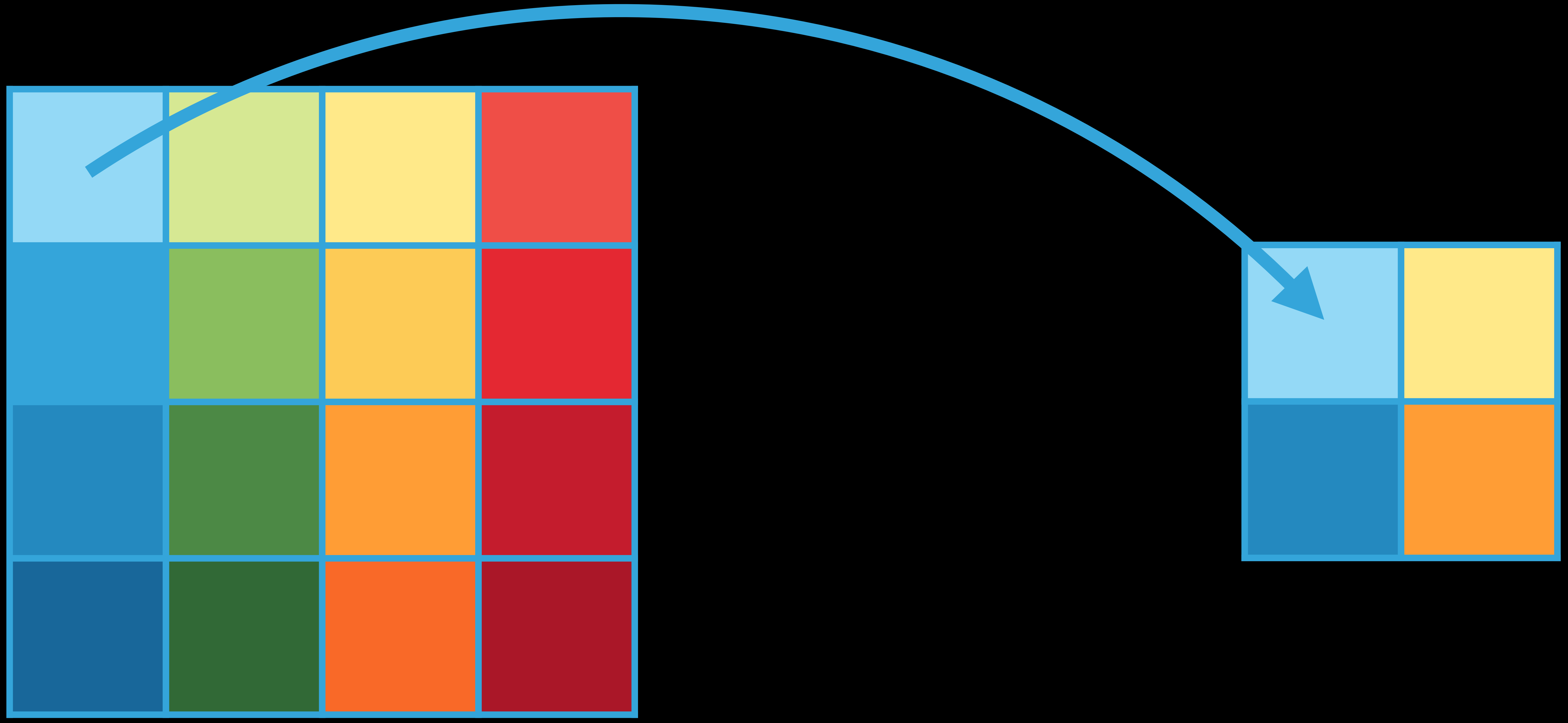

Metal Texture*

* or another similar pattern

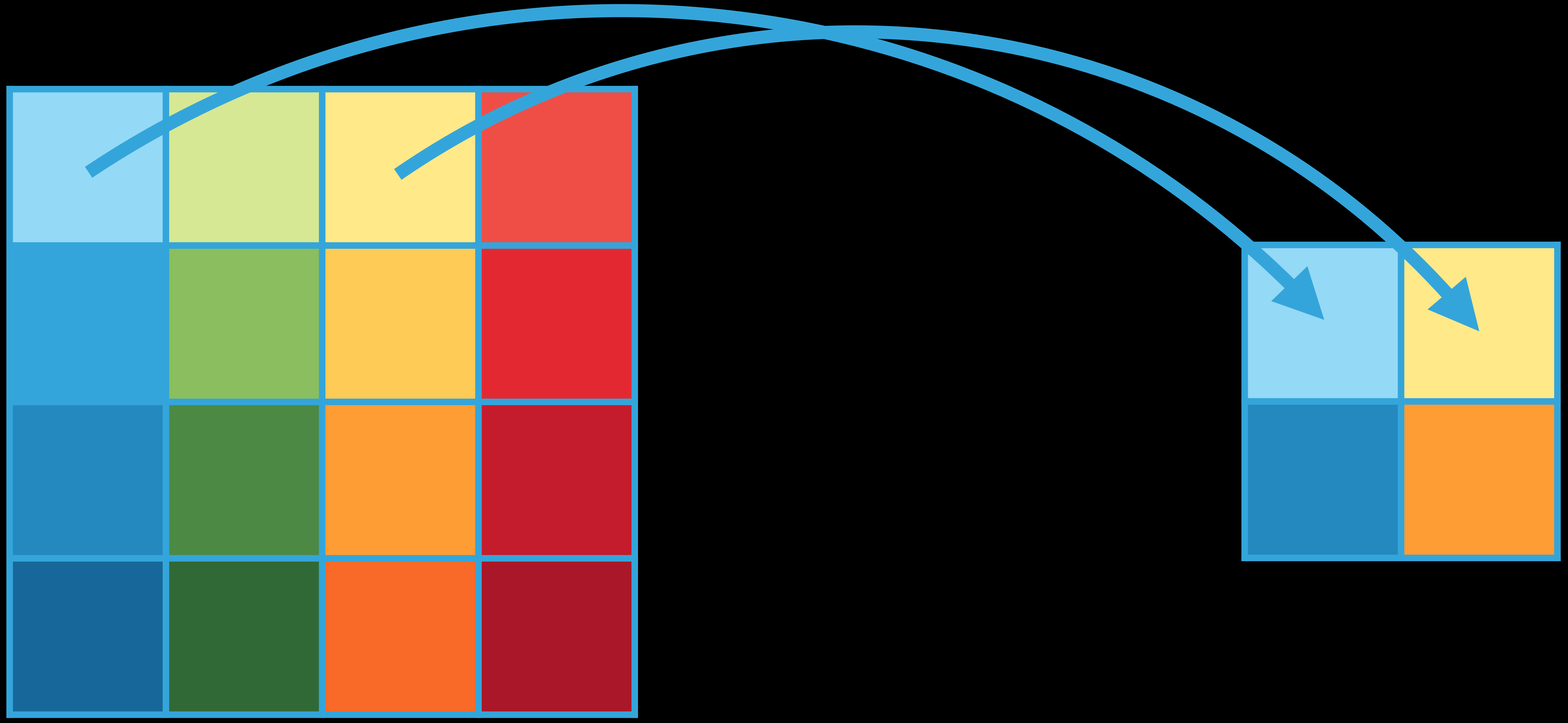
Sampling a Lower Resolution



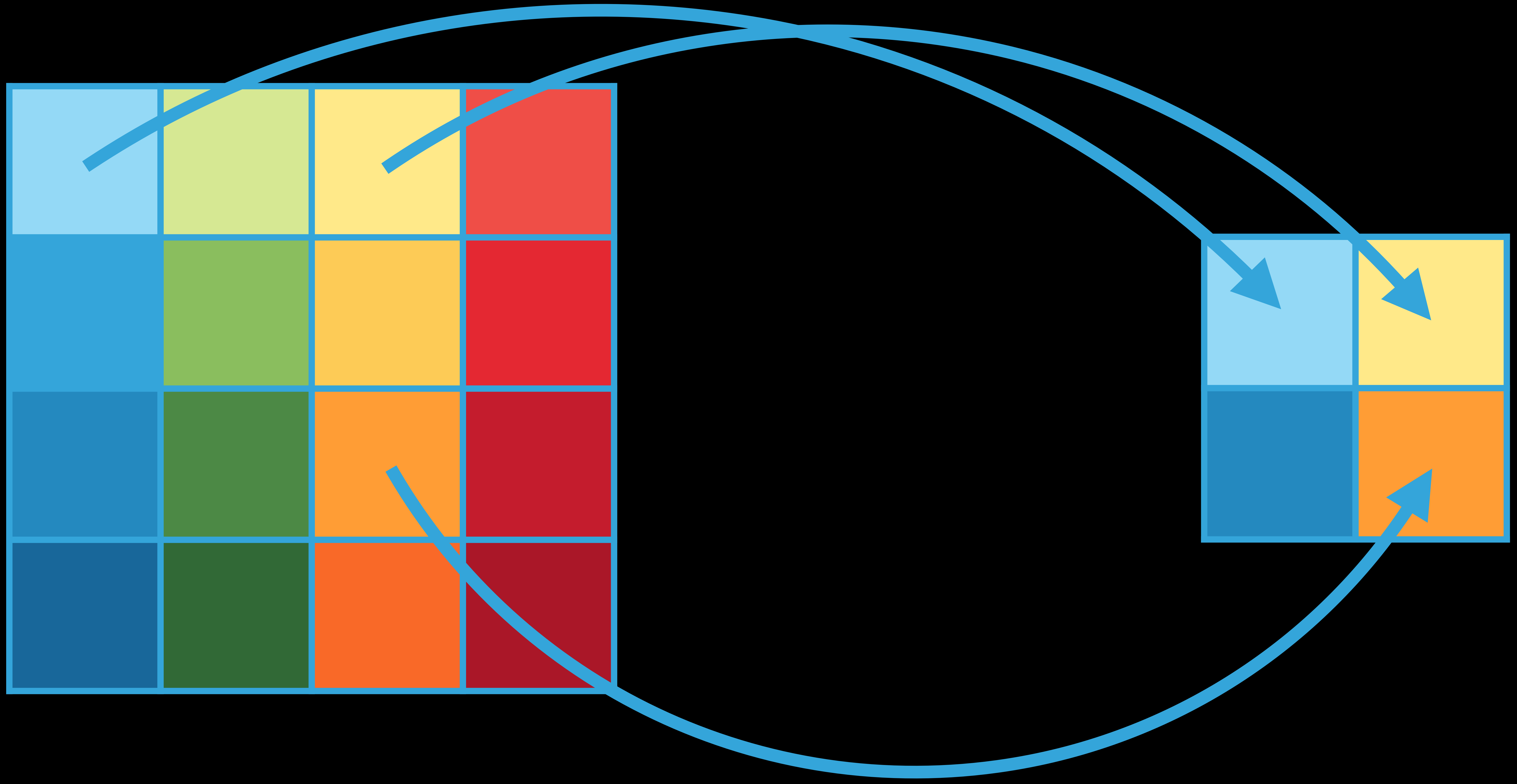
SAMPLING A LOWER RESOLUTION



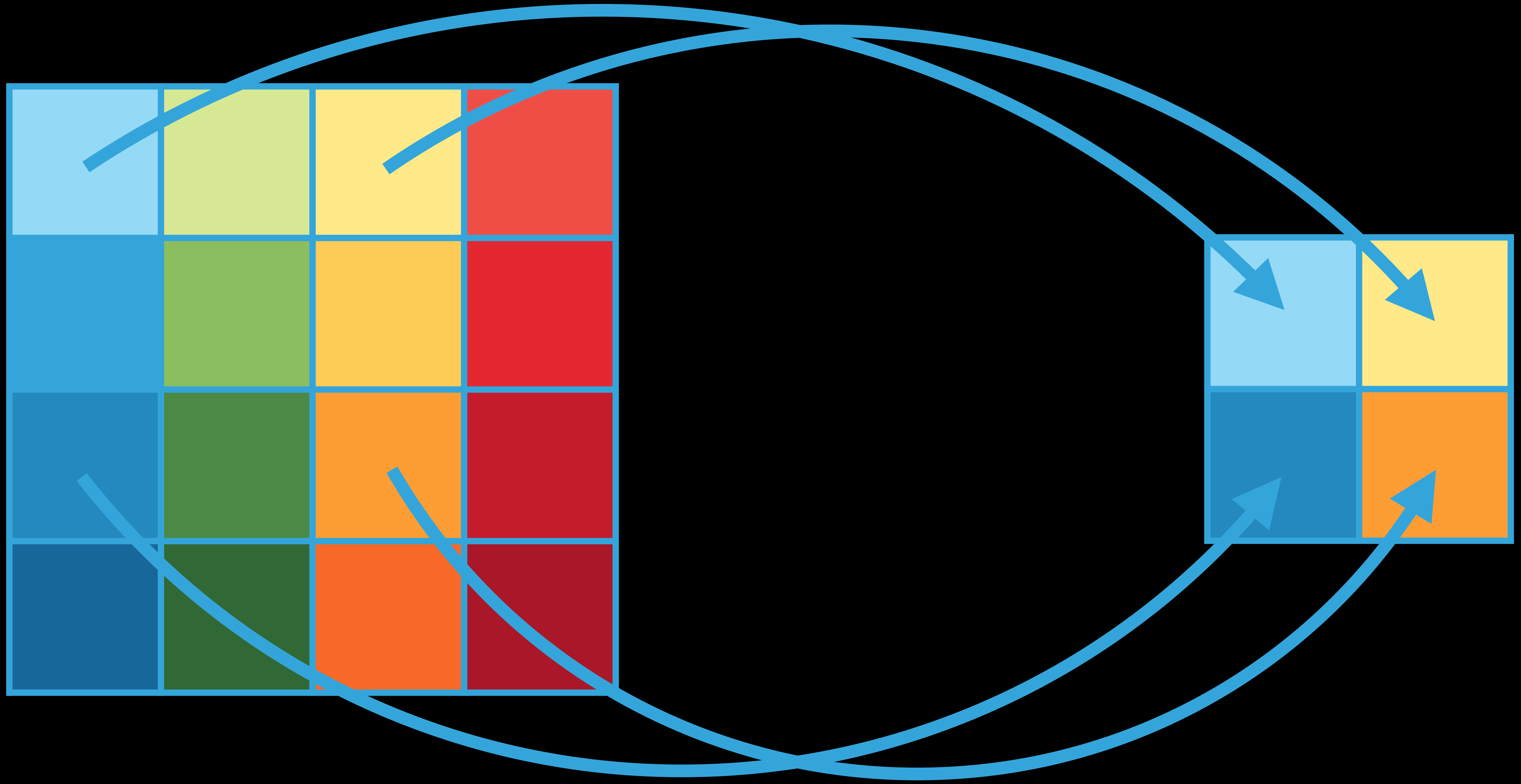
SAMPLING A LOWER RESOLUTION



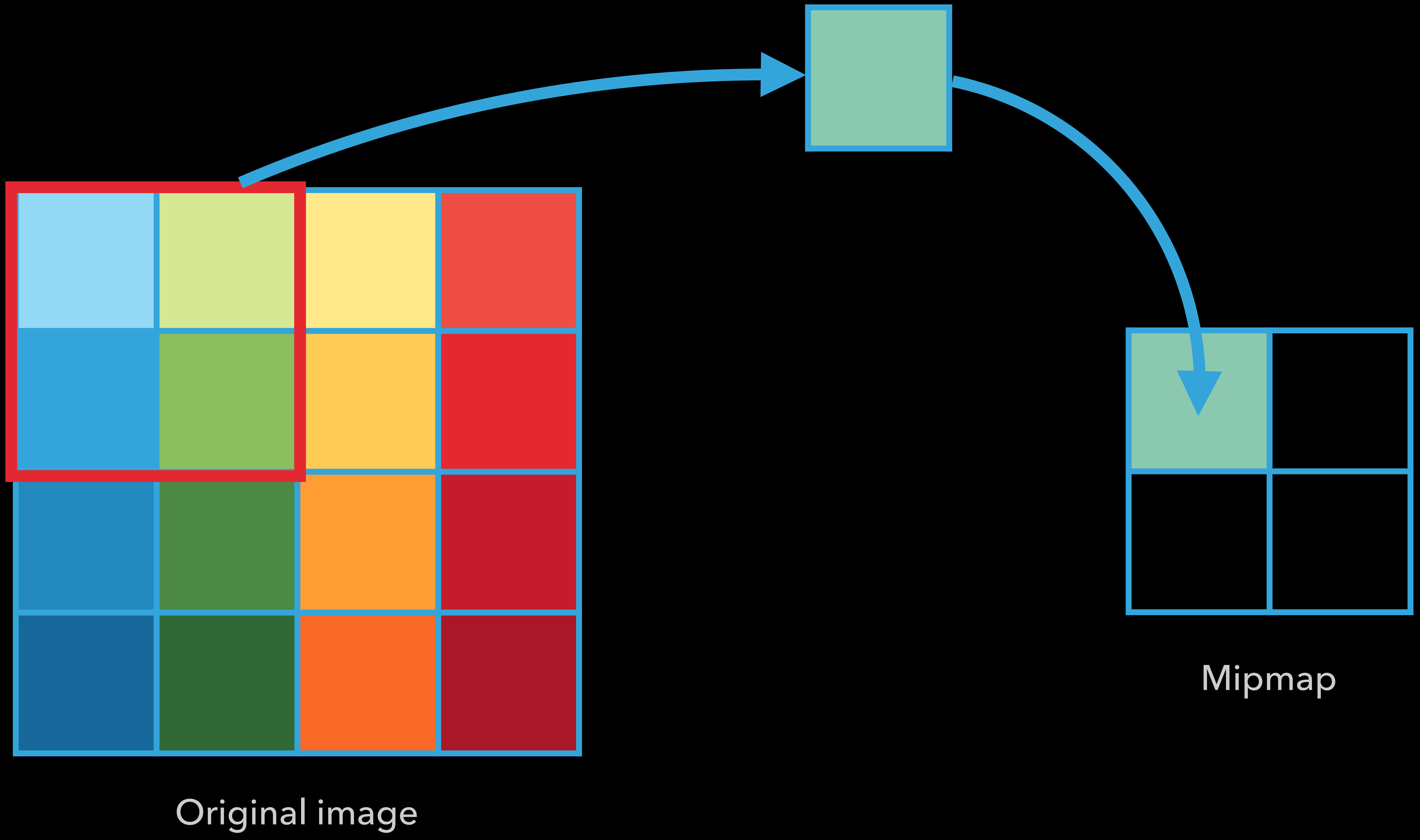
SAMPLING A LOWER RESOLUTION



SAMPLING A LOWER RESOLUTION



SAMPLING A LOWER RESOLUTION

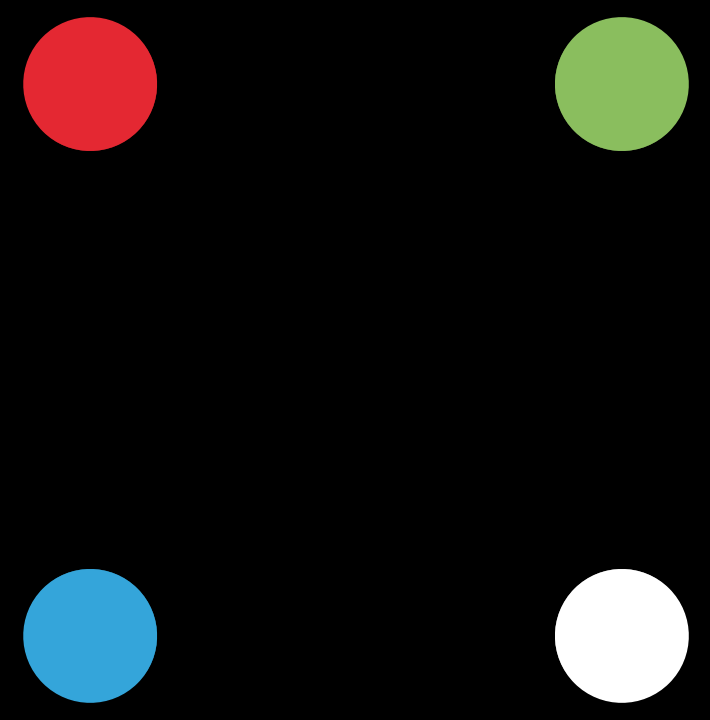


Compute Shaders With Textures instead of Buffers

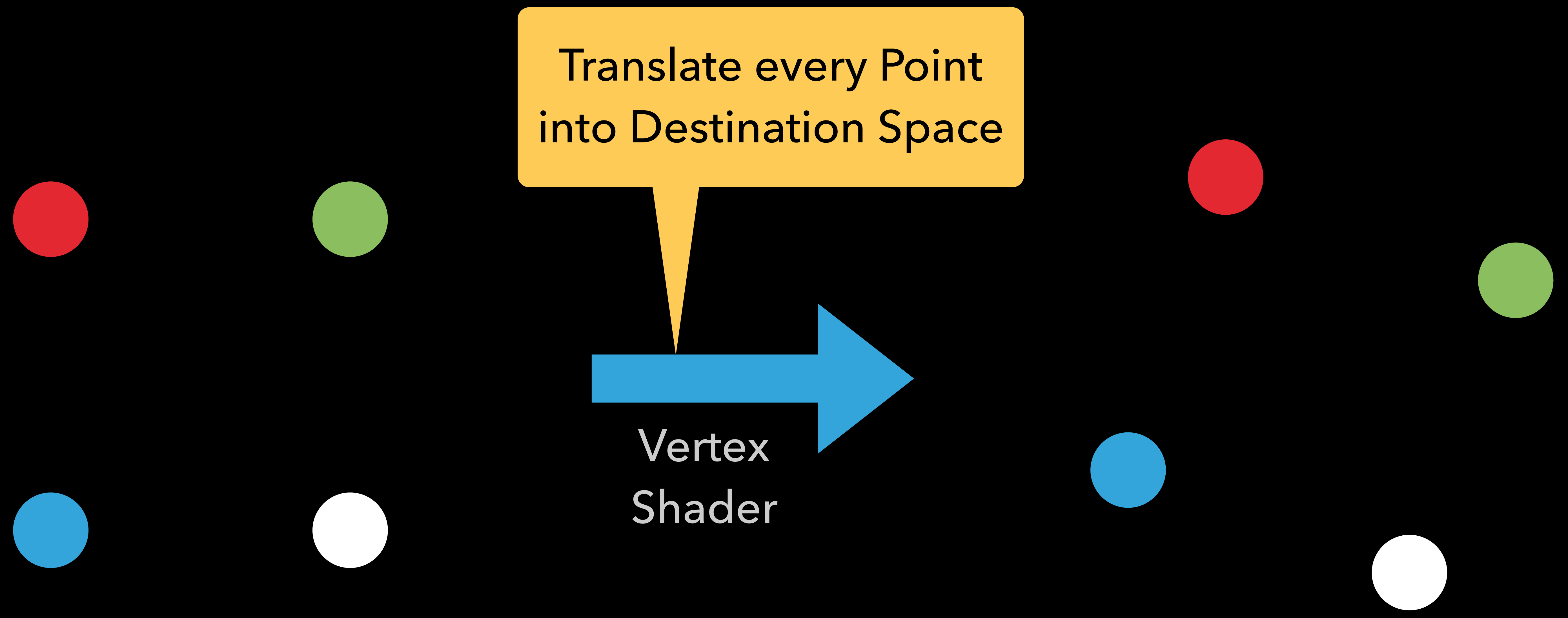
Demo

Our Goal

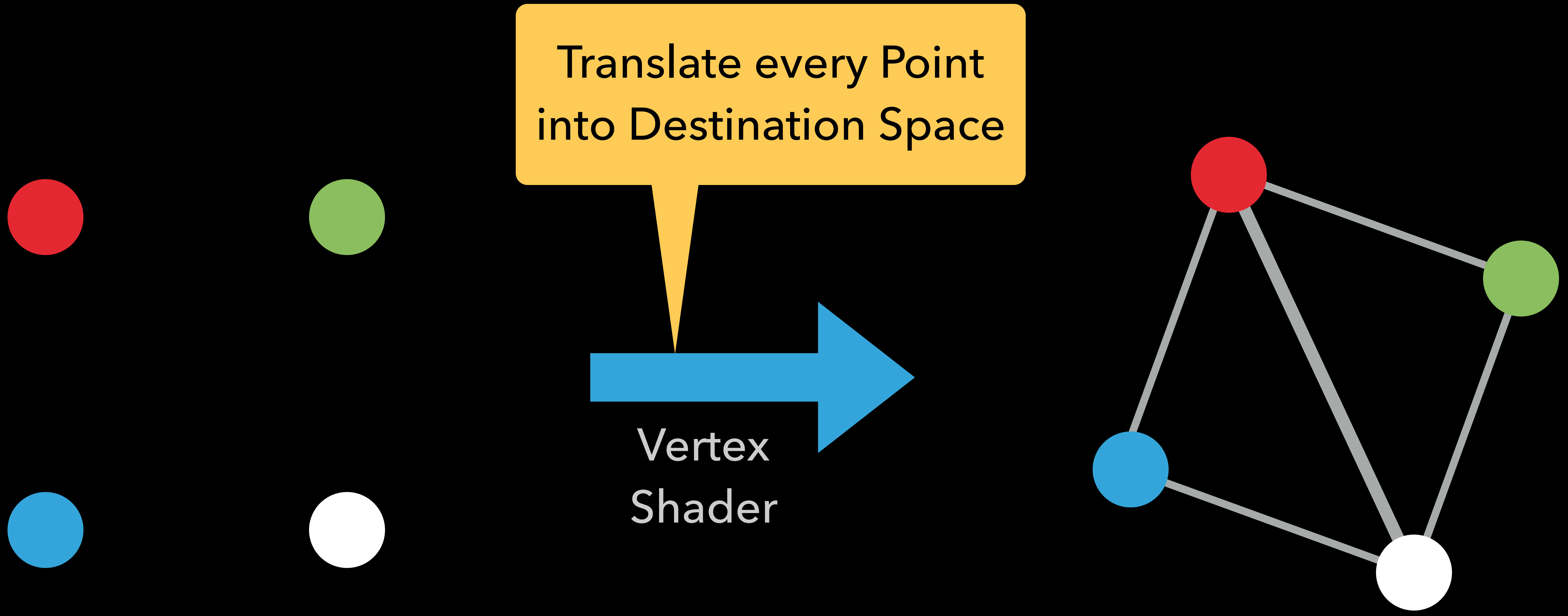
Our Plan



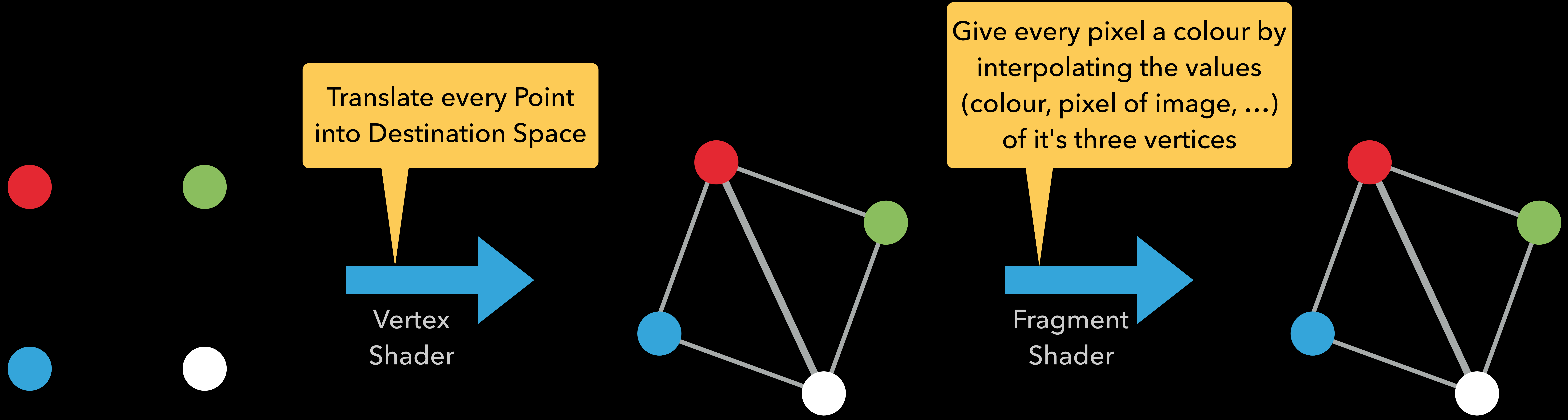
OUR PLAN


Translate every Point into Destination Space
Vertex Shader

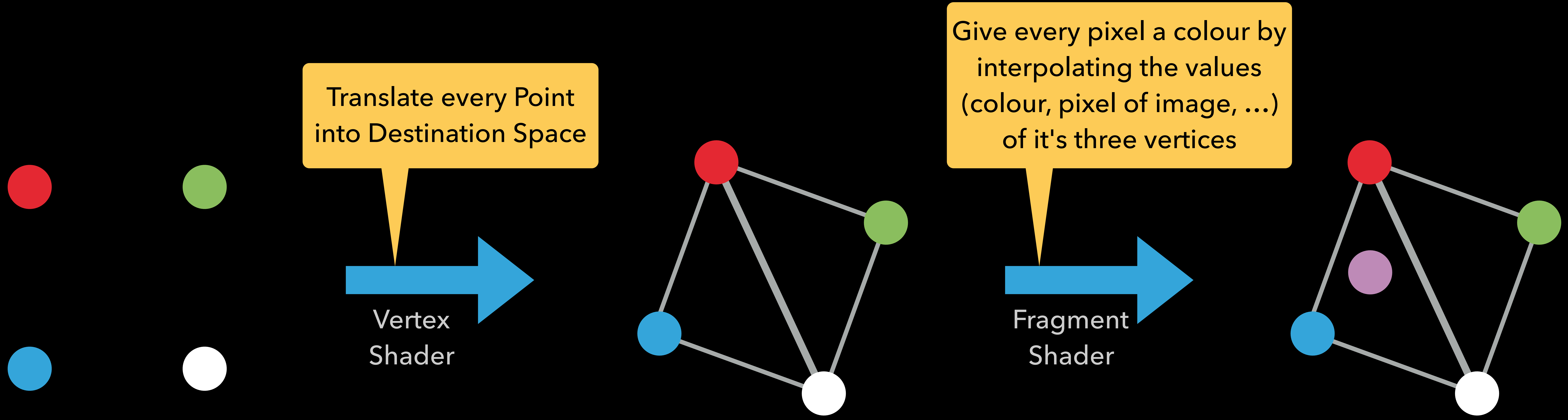
OUR PLAN


Translate every Point into Destination Space
Vertex Shader

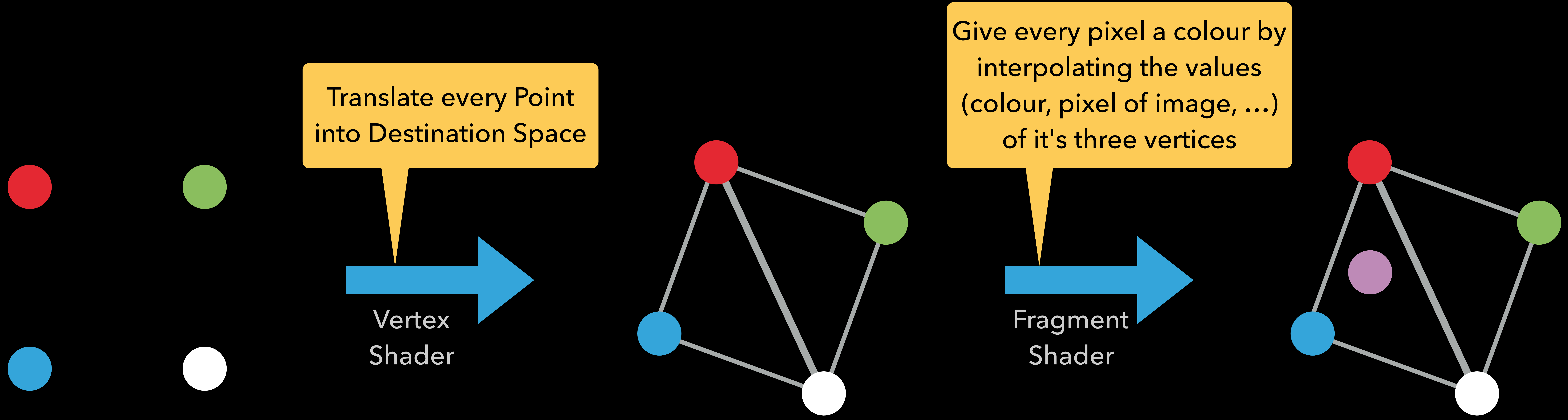
OUR PLAN


Translate every Point into Destination Space
Vertex Shader
Give every pixel a colour by interpolating the values (colour, pixel of image, ...) of it's three vertices
Fragment Shader

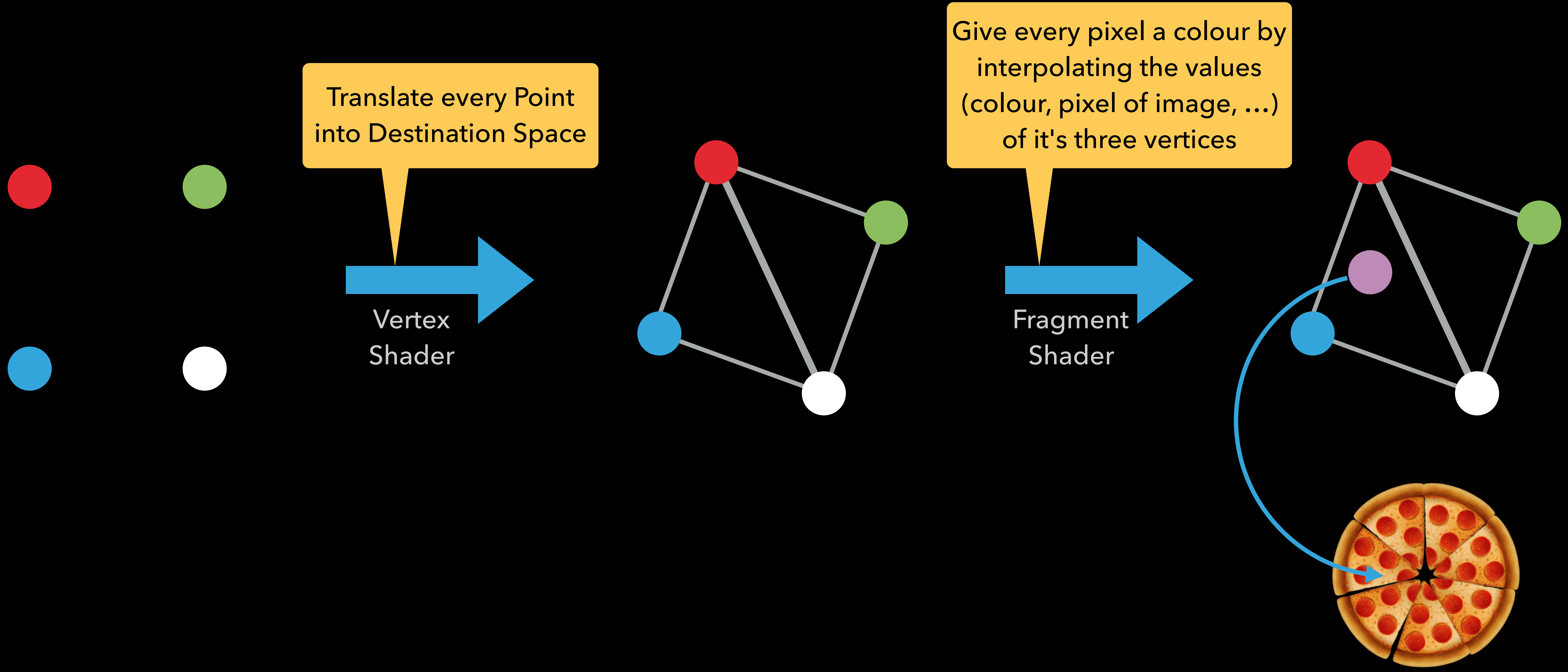
Our Plan


Translate every Point into Destination Space
Vertex Shader
Give every pixel a colour by interpolating the values (colour, pixel of image, …) of it's three vertices
Fragment Shader

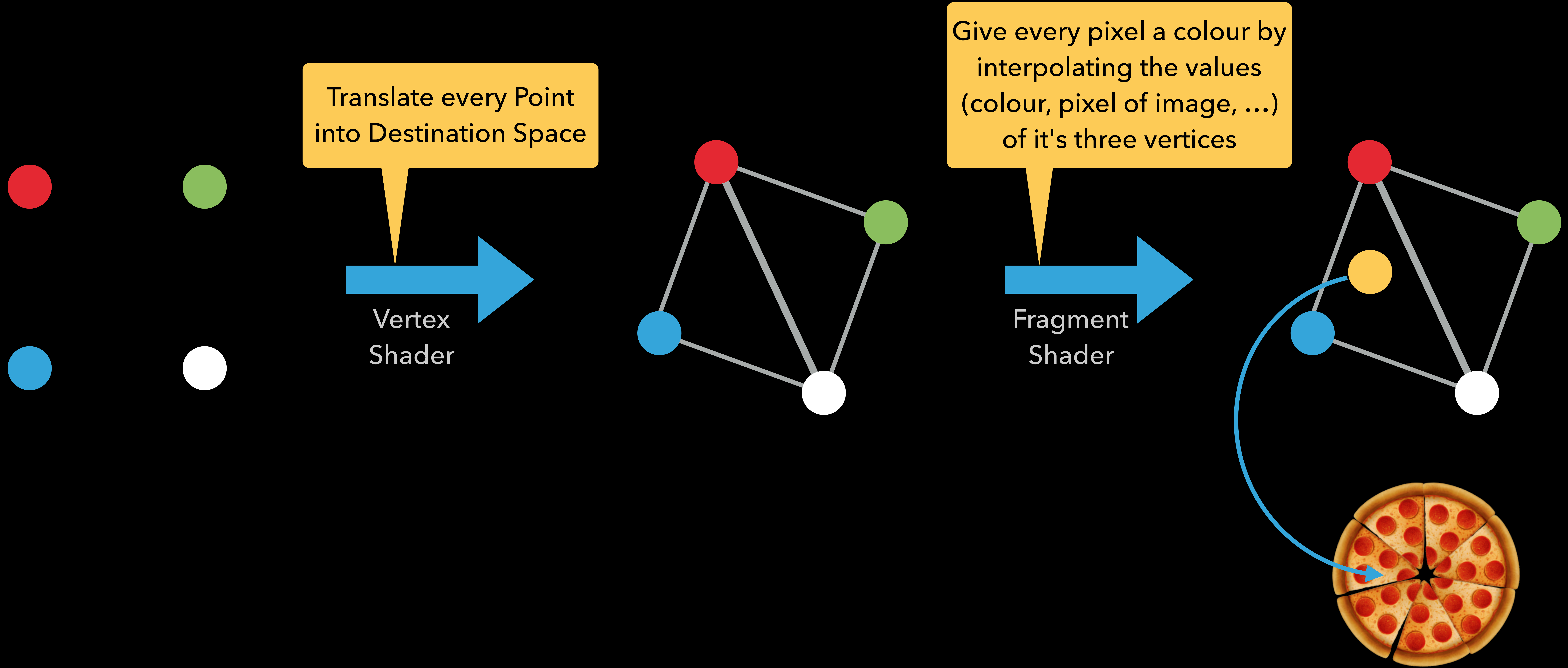
OUR PLAN


Translate every Point into Destination Space
Vertex Shader
Give every pixel a colour by interpolating the values (colour, pixel of image, …) of it's three vertices
Fragment Shader

Our Plan


Translate every Point into Destination Space
Vertex Shader
Give every pixel a colour by interpolating the values (colour, pixel of image, …) of it's three vertices
Fragment Shader

Our Plan


Translate every Point into Destination Space
Vertex Shader
Give every pixel a colour by interpolating the values (colour, pixel of image, …) of it's three vertices
Fragment Shader

Metal with Vertex/Fragment Shaders

Demo

METAL PERFORMANCE SHADERS

- Precompiled Metal shaders highly optimised for every GPU
- Applications
  - Extract statistical data from images
  - Neural networks
  - Matrix operations including equation system solving
  - Ray tracing

Metal requirements

~~iOS8 only – Mac support would require unified RAM~~

~~ARM64 only – iPhone 5s, iPad Air, iPad mini with Retina Screen~~

~~Device only – no simulator support~~

Xcode 6

Thank You

Alex Hoppen
https://alexhoppen.de