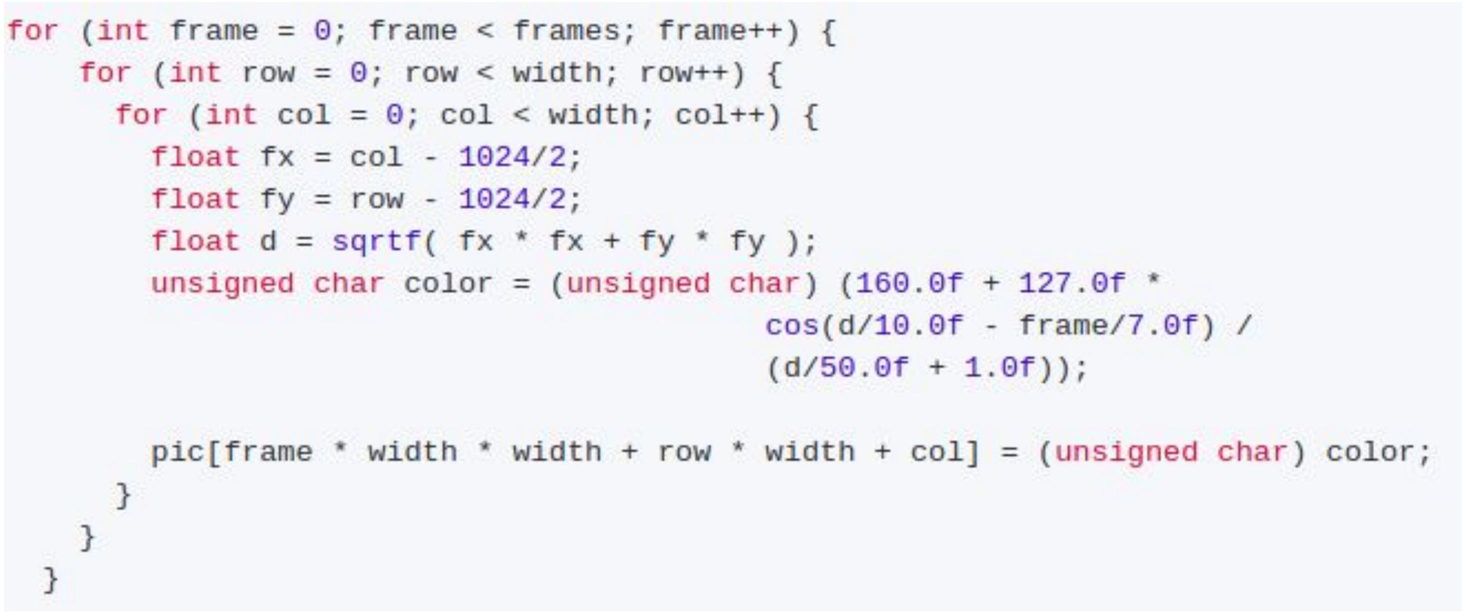
T8: Geração de Imagem em Paralelo com CUDA

Eduardo Rafael Hirt

Primeira parte

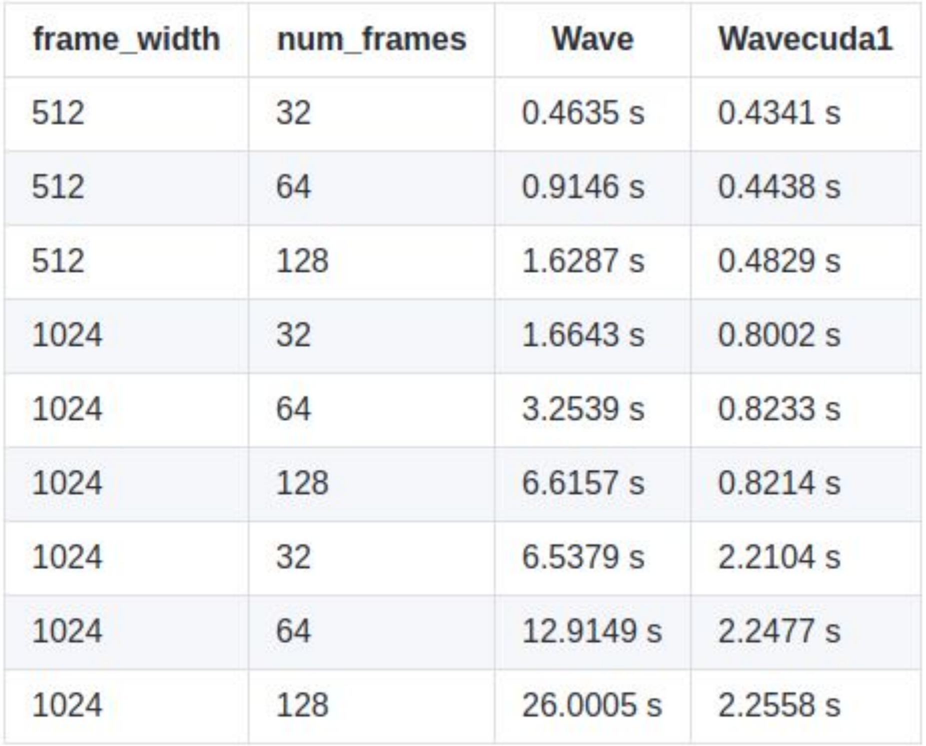
- Paralelização do laço mais externo:



```
for (int frame = 0; frame < frames; frame++) {
    for (int row = 0; row < width; row++) {
      for (int col = 0; col < width; col++) {
        float fx = col - 1024/2;
        float fy = row - 1024/2;
        float d = sqrtf( fx * fx + fy * fy );
        unsigned char color = (unsigned char) (160.0f + 127.0f *
                                          cos(d/10.0f - frame/7.0f) /
                                          (d/50.0f + 1.0f));

        pic[frame * width * width + row * width + col] = (unsigned char) color;
      }
    }
  }
```

```
__global__ void calcularFrame(unsigned char* pic, int width)
{
    int frame = threadIdx.x;
    for (int row = 0; row < width; row++) {
        for (int col = 0; col < width; col++) {
            float fx = col - 1024/2;
            float fy = row - 1024/2;
            float d = sqrtf( fx * fx + fy * fy );
            unsigned char color = (unsigned char) (160.0f + 127.0f *
                                                    cos(d/10.0f - frame/7.0f) /
                                                    (d/50.0f + 1.0f));

            pic[frame * width * width + row * width + col] = (unsigned char) color;
        }
    }

}
```

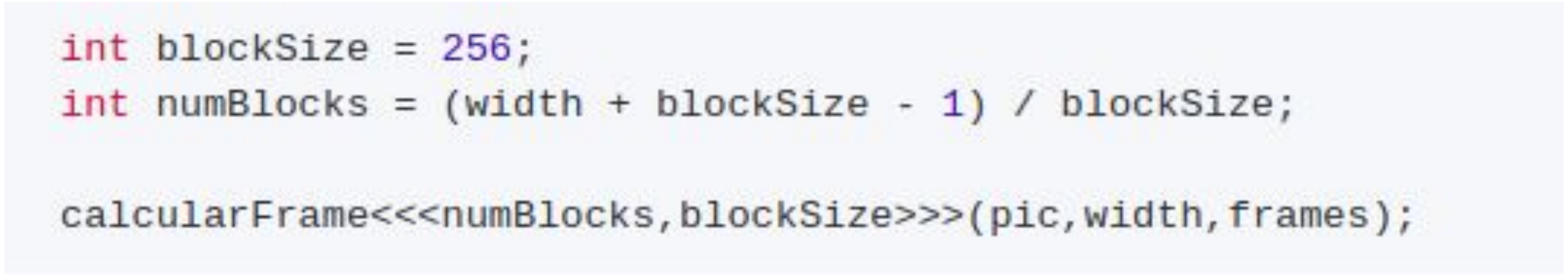
Resultados



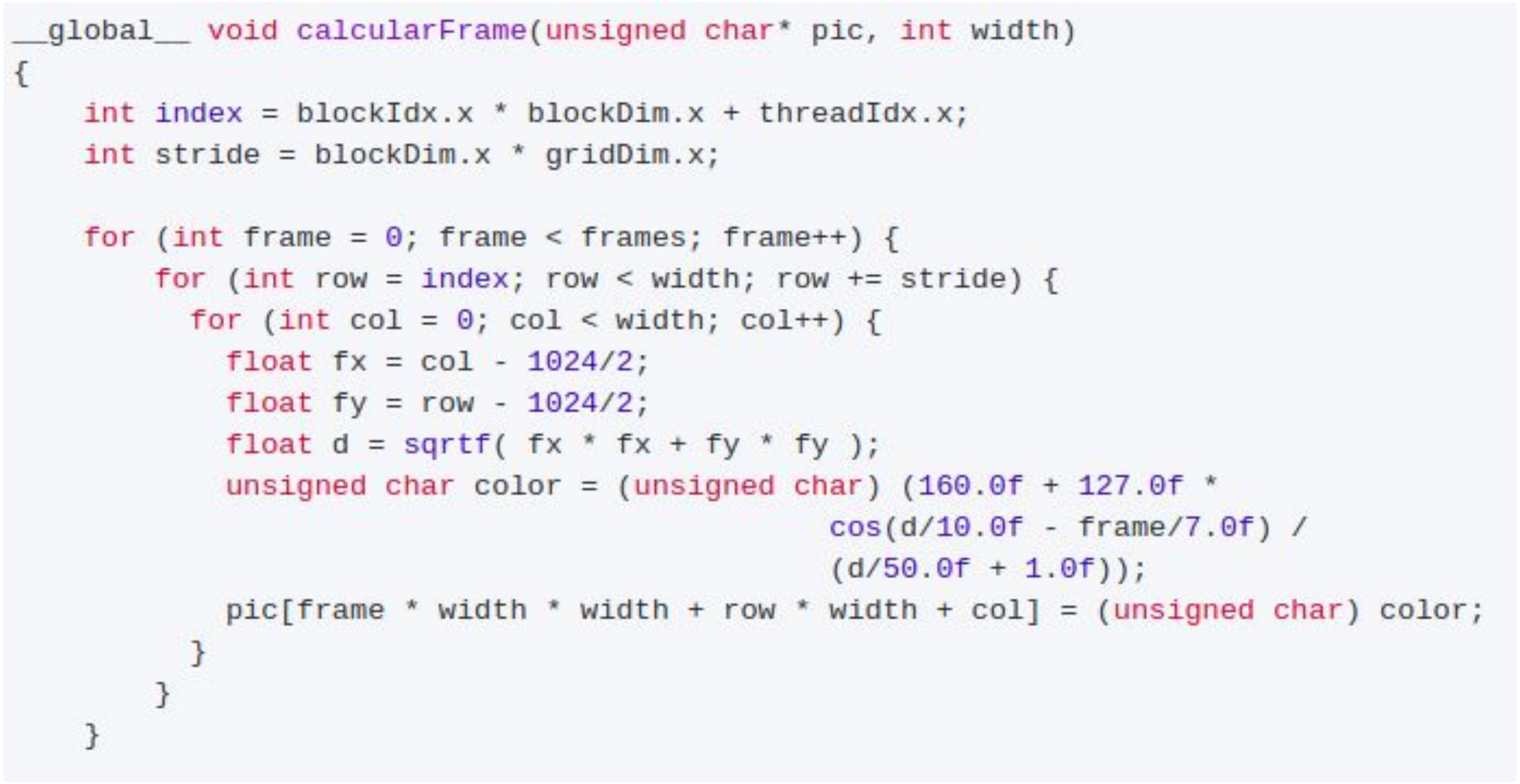
frame_width	num_frames	Wave	Wavecuda1
512	32	0.4635 s	0.4341 s
512	64	0.9146 s	0.4438 s
512	128	1.6287 s	0.4829 s
1024	32	1.6643 s	0.8002 s
1024	64	3.2539 s	0.8233 s
1024	128	6.6157 s	0.8214 s
1024	32	6.5379 s	2.2104 s
1024	64	12.9149 s	2.2477 s
1024	128	26.0005 s	2.2558 s

Segunda parte

- Paralelização feita em blocos

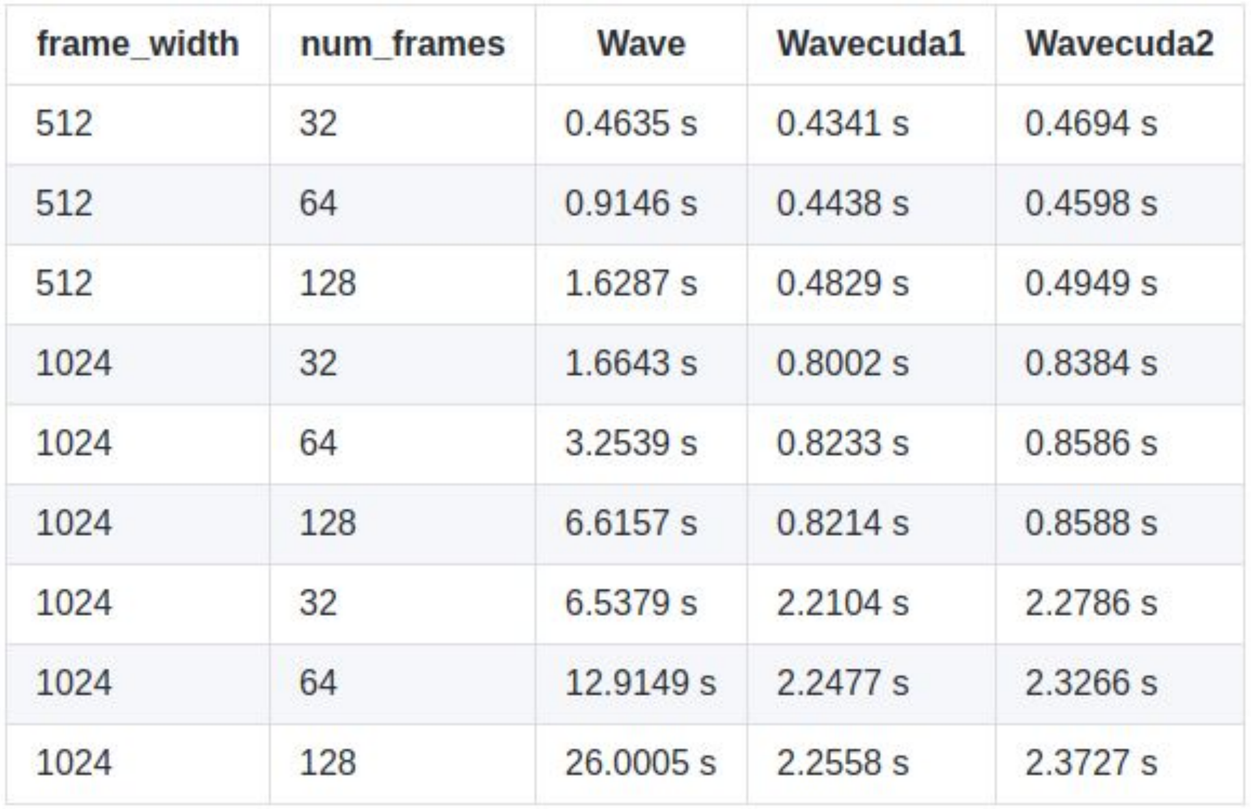


```
int blockSize = 256;
int numBlocks = (width + blockSize - 1) / blockSize;

calcularFrame<<<numBlocks,blockSize>>>(pic,width,frames);
```



```
__global__ void calcularFrame(unsigned char* pic, int width)
{
    int index = blockIdx.x * blockDim.x + threadIdx.x;
    int stride = blockDim.x * gridDim.x;

    for (int frame = 0; frame < frames; frame++) {
        for (int row = index; row < width; row += stride) {
          for (int col = 0; col < width; col++) {
            float fx = col - 1024/2;
            float fy = row - 1024/2;
            float d = sqrtf( fx * fx + fy * fy );
            unsigned char color = (unsigned char) (160.0f + 127.0f *
                                              cos(d/10.0f - frame/7.0f) /
                                              (d/50.0f + 1.0f));
            pic[frame * width * width + row * width + col] = (unsigned char) color;
          }
        }
    }
```

Resultados



frame_width	num_frames	Wave	Wavecuda1	Wavecuda2
512	32	0.4635 s	0.4341 s	0.4694 s
512	64	0.9146 s	0.4438 s	0.4598 s
512	128	1.6287 s	0.4829 s	0.4949 s
1024	32	1.6643 s	0.8002 s	0.8384 s
1024	64	3.2539 s	0.8233 s	0.8586 s
1024	128	6.6157 s	0.8214 s	0.8588 s
1024	32	6.5379 s	2.2104 s	2.2786 s
1024	64	12.9149 s	2.2477 s	2.3266 s
1024	128	26.0005 s	2.2558 s	2.3727 s