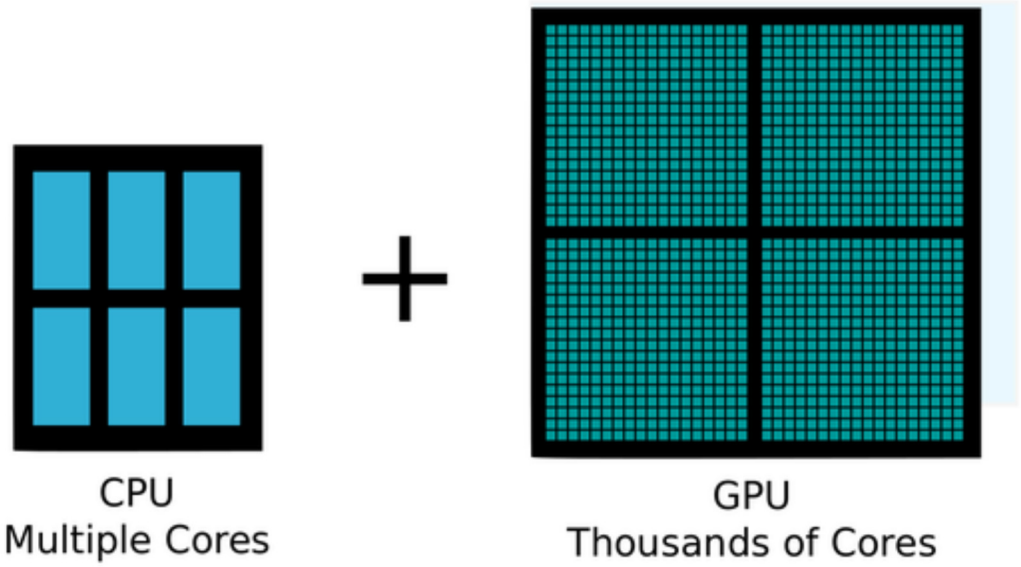
(19) Brett Bergan's answer to How does a GPU manage 1920*1080 pixels individually at 120Hz?

A GPU doesn't exactly manage pixels individually.

Color data for each pixel is stored for each pixel in a frame buffer. Eight bits per color per pixel—so essentially 6MB in all for 1080p. A frame buffer can display a still frame at a thousand frames per second without a GPU at all if it wants to, since nothing changes in the frame buffer.

The mouse cursor (or sprite) is handled differently than graphics and moves independently of the rest of the items on the screen using the mouse cursor image in a hardware buffer with its location governed by an X-register and a Y-register.

For a 3D game to run at 120fps, a GPU has (let's just say 2000) GPU cores. These cores are subdivided into streaming multiprocessors that (in 2020) all have 64 GPU cores running on a common cache. Also each individual core has its own cache for doing its immediate task. This is normally 32K for data and 32K for its immediate instruction or algorithm.



> A RTX 2060 has 1920 CUDA cores, 120 TMU's (texture mapping units), and 48 ROP's (raster operation pipelines)

A CPU core can only handle one or two threads per clock cycle, while a GPU core can also handle one thread per clock cycle. But when you link hundreds of them or thousands of them together, each managing a small section of the task, the overall job of rendering a screen of graphics is extremely quick. Rendering an image might require hundreds of steps or stages in all. But when each step only takes a few hundred nanoseconds to compute, the the 8-millisecond (8-million-nanosecond) frame time required for 120fps is very achievable.

A 5GHz CPU can compute the locations of a complex wireframe vertex array very quickly, while a GPU has tools and storage specifically designed to render textures and lighting effects to bring the wireframe to life.

It is a constant volley between the CPU and GPU, exchanging real-time data and flipping at back and forth to each other. CPU creates vertex array—flips it to GPU. GPU renders frame—notifies CPU when finished. CPU updates position and creates new vertex array—and so on.

July 20, 2020 Edit:

It just occurred to me that I left out the most interesting part. 3D rendering uses a technique called DBSC. The double-buffer swap chain uses two frame buffers. One frame buffer functions as a canvas so that the renderer can “paint” its picture. When the picture is done, its image gets swapped to the monitor. The first frame buffer is erased and the renderer goes to work painting the next picture. The buffers are swapped. One always displays and image on the screen while the other is being processed.

6K views ·

View Upvoters

.

View Sharers

· Answer requested by