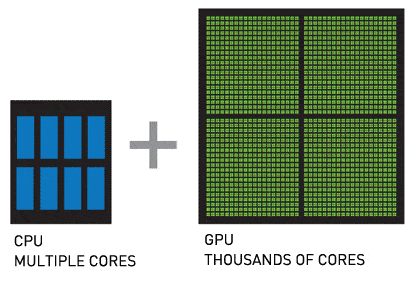
## **From Wikipedia** <https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#Hardware_support>

**From MakeTechEasier** <https://www.maketecheasier.com/difference-between-cpu-and-gpu/>

## **What’s the Difference Between a CPU and a GPU?**



If a CPU is a Leatherman, a GPU is a very sharp knife. You can’t tighten a hex bolt with a knife, but you can definitely cut some stuff.

A GPU can only do a fraction of the many operations a CPU does, but it does so with incredible speed. A GPU will use hundreds of cores to make time-sensitive calculations for thousands of pixels at a time, making it possible to display complex 3D graphics. However, as fast as a GPU can go, it can only really perform “dumb” operations.

For example, a modern GPU like the Nvidia GTX 1080 has 2560 shader cores. Thanks to those cores, it can execute 2560 instructions, or operations, during one clock cycle.

**GPU programming concepts**[[edit](https://en.wikipedia.org/w/index.php?title=General-purpose_computing_on_graphics_processing_units&action=edit&section=13)]

**Computational resources**[[edit](https://en.wikipedia.org/w/index.php?title=General-purpose_computing_on_graphics_processing_units&action=edit&section=14)]

There are a variety of computational resources available on the GPU:

* Programmable processors – vertex, primitive, fragment and mainly compute pipelines allow programmer to perform kernel on streams of data
* Rasterizer – creates fragments and interpolates per-vertex constants such as texture coordinates and color
* Texture unit – read-only memory interface
* Framebuffer – write-only memory interface

In fact, a program can substitute a write only texture for output instead of the framebuffer. This is done either through [Render to Texture](https://en.wikipedia.org/wiki/Framebuffer_object#Uses) (RTT), Render-To-Backbuffer-Copy-To-Texture (RTBCTT), or the more recent stream-out.

**Textures as stream**[[edit](https://en.wikipedia.org/w/index.php?title=General-purpose_computing_on_graphics_processing_units&action=edit&section=15)]

Since textures are used as memory, texture lookups are then used as memory reads. Certain operations can be done automatically by the GPU because of this.

**Kernels**[[edit](https://en.wikipedia.org/w/index.php?title=General-purpose_computing_on_graphics_processing_units&action=edit&section=16)]

[Compute kernels](https://en.wikipedia.org/wiki/Compute_kernel) can be thought of as the body of [loops](https://en.wikipedia.org/wiki/Loop_(computing)). For example, a programmer operating on a grid on the CPU might have code that looks like this:

*// Input and output grids have 10000 x 10000 or 100 million elements.*

void transform\_10k\_by\_10k\_grid(float in[10000][10000], float out[10000][10000])

{

**for** (int x = 0; x < 10000; x++) {

**for** (int y = 0; y < 10000; y++) {

*// The next line is executed 100 million times*

out[x][y] = do\_some\_hard\_work(in[x][y]);

}

}

}

On the GPU, the programmer only specifies the body of the loop as the kernel and what data to loop over by invoking geometry processing.

#### Flow control[[edit](https://en.wikipedia.org/w/index.php?title=General-purpose_computing_on_graphics_processing_units&action=edit&section=17)]

In sequential code it is possible to control the flow of the program using if-then-else statements and various forms of loops. Such flow control structures have only recently been added to GPUs.[[36]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-book-36) Conditional writes could be performed using a properly crafted series of arithmetic/bit operations, but looping and conditional branching were not possible.

Recent GPUs allow branching, but usually with a performance penalty. Branching should generally be avoided in inner loops, whether in CPU or GPU code, and various methods, such as static branch resolution, pre-computation, predication, loop splitting,[[37]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-Tutorial_on_eliminating_branches-37) and Z-cull[[38]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-survey-38) can be used to achieve branching when hardware support does not exist.

### GPU methods[[edit](https://en.wikipedia.org/w/index.php?title=General-purpose_computing_on_graphics_processing_units&action=edit&section=18)]

#### Map[[edit](https://en.wikipedia.org/w/index.php?title=General-purpose_computing_on_graphics_processing_units&action=edit&section=19)]

*Main article:*[*Map (parallel pattern)*](https://en.wikipedia.org/wiki/Map_(parallel_pattern))

The map operation simply applies the given function (the kernel) to every element in the stream. A simple example is multiplying each value in the stream by a constant (increasing the brightness of an image). The map operation is simple to implement on the GPU. The programmer generates a fragment for each pixel on screen and applies a fragment program to each one. The result stream of the same size is stored in the output buffer.

#### Reduce[[edit](https://en.wikipedia.org/w/index.php?title=General-purpose_computing_on_graphics_processing_units&action=edit&section=20)]

*Main article:*[*Fold (higher-order function)*](https://en.wikipedia.org/wiki/Fold_(higher-order_function))

Some computations require calculating a smaller stream (possibly a stream of only 1 element) from a larger stream. This is called a reduction of the stream. Generally, a reduction can be performed in multiple steps. The results from the prior step are used as the input for the current step and the range over which the operation is applied is reduced until only one stream element remains.

#### Stream filtering[[edit](https://en.wikipedia.org/w/index.php?title=General-purpose_computing_on_graphics_processing_units&action=edit&section=21)]

Stream filtering is essentially a non-uniform reduction. Filtering involves removing items from the stream based on some criteria.

#### Scan[[edit](https://en.wikipedia.org/w/index.php?title=General-purpose_computing_on_graphics_processing_units&action=edit&section=22)]

The scan operation, also termed [*parallel prefix sum*](https://en.wikipedia.org/wiki/Prefix_sum#Parallel_algorithm), takes in a vector (stream) of data elements and an [(arbitrary) associative binary function '+' with an identity element 'i'](https://en.wikipedia.org/wiki/Monoid). If the input is [a0, a1, a2, a3, ...], an *exclusive scan* produces the output [i, a0, a0 + a1, a0 + a1 + a2, ...], while an *inclusive scan* produces the output [a0, a0 + a1, a0 + a1 + a2, a0 + a1 + a2 + a3, ...] and [does not require an identity](https://en.wikipedia.org/wiki/Semigroup) to exist. While at first glance the operation may seem inherently serial, efficient parallel scan algorithms are possible and have been implemented on graphics processing units. The scan operation has uses in e.g., quicksort and sparse matrix-vector multiplication.[[34]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-goddeke2010-34)[[39]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-39)[[40]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-40)[[41]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-41)

#### Scatter[[edit](https://en.wikipedia.org/w/index.php?title=General-purpose_computing_on_graphics_processing_units&action=edit&section=23)]

The [scatter](https://en.wikipedia.org/wiki/Scatter_(vector_addressing)) operation is most naturally defined on the vertex processor. The vertex processor is able to adjust the position of the [vertex](https://en.wikipedia.org/wiki/Vertex_(geometry)), which allows the programmer to control where information is deposited on the grid. Other extensions are also possible, such as controlling how large an area the vertex affects.

The fragment processor cannot perform a direct scatter operation because the location of each fragment on the grid is fixed at the time of the fragment's creation and cannot be altered by the programmer. However, a logical scatter operation may sometimes be recast or implemented with another gather step. A scatter implementation would first emit both an output value and an output address. An immediately following gather operation uses address comparisons to see whether the output value maps to the current output slot.

In dedicated [compute kernels](https://en.wikipedia.org/wiki/Compute_kernel), scatter can be performed by indexed writes.

#### Gather[[edit](https://en.wikipedia.org/w/index.php?title=General-purpose_computing_on_graphics_processing_units&action=edit&section=24)]

[Gather](https://en.wikipedia.org/wiki/Gather_(vector_addressing)) is the reverse of scatter, after scatter reorders elements according to a map, gather can restore the order of the elements according to the map scatter used. In dedicated compute kernels, gather may be performed by indexed reads. In other shaders, it is performed with texture-lookups.

#### Sort[[edit](https://en.wikipedia.org/w/index.php?title=General-purpose_computing_on_graphics_processing_units&action=edit&section=25)]

The sort operation transforms an unordered set of elements into an ordered set of elements. The most common implementation on GPUs is using [radix sort](https://en.wikipedia.org/wiki/Radix_sort) for integer and floating point data and coarse-grained [merge sort](https://en.wikipedia.org/wiki/Merge_sort) and fine-grained [sorting networks](https://en.wikipedia.org/wiki/Sorting_networks) for general comparable data.[[42]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-merrill-thesis-42)[[43]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-modern-gnu-43)

#### Search[[edit](https://en.wikipedia.org/w/index.php?title=General-purpose_computing_on_graphics_processing_units&action=edit&section=26)]

The search operation allows the programmer to find a given element within the stream, or possibly find neighbors of a specified element. The GPU is not used to speed up the search for an individual element, but instead is used to run multiple searches in parallel.[[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)] Mostly the search method used is [binary search](https://en.wikipedia.org/wiki/Binary_search) on sorted elements.

#### Data structures[[edit](https://en.wikipedia.org/w/index.php?title=General-purpose_computing_on_graphics_processing_units&action=edit&section=27)]

A variety of data structures can be represented on the GPU:

* Dense [arrays](https://en.wikipedia.org/wiki/Array_data_structure)
* [Sparse matrixes](https://en.wikipedia.org/wiki/Sparse_matrix) ([sparse array](https://en.wikipedia.org/wiki/Sparse_array))  – static or dynamic
* Adaptive structures ([union type](https://en.wikipedia.org/wiki/Union_type))

## **Applications[**[**edit**](https://en.wikipedia.org/w/index.php?title=General-purpose_computing_on_graphics_processing_units&action=edit&section=28)**]**

The following are some of the areas where GPUs have been used for general purpose computing:

* [Automatic parallelization](https://en.wikipedia.org/wiki/Automatic_parallelization)[[44]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-44)[[45]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-45)[[46]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-46)
* [Computer clusters](https://en.wikipedia.org/wiki/Computer_cluster) or a variant of a [parallel computing](https://en.wikipedia.org/wiki/Parallel_computing) (using [GPU cluster](https://en.wikipedia.org/wiki/GPU_cluster) technology) for highly calculation-intensive tasks:[[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)]
  + [High-performance computing](https://en.wikipedia.org/wiki/High-performance_computing) (HPC) [clusters](https://en.wikipedia.org/wiki/Computer_cluster), often termed [supercomputers](https://en.wikipedia.org/wiki/Supercomputer)
    - including cluster technologies like [Message Passing Interface](https://en.wikipedia.org/wiki/Message_Passing_Interface), and [single-system image](https://en.wikipedia.org/wiki/Single-system_image) (SSI), [distributed computing](https://en.wikipedia.org/wiki/Distributed_computing), and [Beowulf](https://en.wikipedia.org/wiki/Beowulf_(computing))
  + [Grid computing](https://en.wikipedia.org/wiki/Grid_computing) (a form of distributed computing) ([networking](https://en.wikipedia.org/wiki/Computer_network) many heterogeneous computers to create a virtual computer architecture)
  + [Load-balancing clusters](https://en.wikipedia.org/wiki/Computer_cluster#Attributes_of_clusters), sometimes termed a [server farm](https://en.wikipedia.org/wiki/Server_farm)
* [Physical based simulation](https://en.wikipedia.org/wiki/Computational_physics) and [physics engines](https://en.wikipedia.org/wiki/Physics_engine)[[25]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-Joselli-25) (usually based on [Newtonian physics](https://en.wikipedia.org/wiki/Newtonian_physics) models)
  + [Conway's Game of Life](https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life), [cloth simulation](https://en.wikipedia.org/wiki/Cloth_simulation), fluid [incompressible flow](https://en.wikipedia.org/wiki/Incompressible_flow) by solution of [Euler equations (fluid dynamics)](https://en.wikipedia.org/wiki/Euler_equations_(fluid_dynamics))[[47]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-47) or [Navier–Stokes equations](https://en.wikipedia.org/wiki/Navier%E2%80%93Stokes_equations" \o "Navier–Stokes equations)[[48]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-48)
* [Statistical physics](https://en.wikipedia.org/wiki/Statistical_physics)
  + [Ising model](https://en.wikipedia.org/wiki/Ising_model)[[49]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-49)
* [Lattice gauge theory](https://en.wikipedia.org/wiki/Lattice_gauge_theory)[[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)]
* [Segmentation](https://en.wikipedia.org/wiki/Segmentation_(image_processing)) – 2D and 3D[[50]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-50)
* [Level set methods](https://en.wikipedia.org/wiki/Level_set_methods)
* [CT](https://en.wikipedia.org/wiki/Computed_tomography) reconstruction[[51]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-51)
* [Fast Fourier transform](https://en.wikipedia.org/wiki/Fast_Fourier_transform)[[52]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-52)
* GPU learning – [machine learning](https://en.wikipedia.org/wiki/Machine_learning) and [data mining](https://en.wikipedia.org/wiki/Data_mining) computations, e.g., with software BIDMach
* [k-nearest neighbor algorithm](https://en.wikipedia.org/wiki/K-nearest_neighbor_algorithm)[[53]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-53)
* [Fuzzy logic](https://en.wikipedia.org/wiki/Fuzzy_logic)[[54]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-54)
* [Tone mapping](https://en.wikipedia.org/wiki/Tone_mapping)
* [Audio signal processing](https://en.wikipedia.org/wiki/Audio_signal_processing)[[55]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-55)
  + Audio and sound effects processing, to use a [GPU](https://en.wikipedia.org/wiki/GPU) for [digital signal processing](https://en.wikipedia.org/wiki/Digital_signal_processing) (DSP)
  + [Analog signal processing](https://en.wikipedia.org/wiki/Analog_signal_processing)
  + [Speech processing](https://en.wikipedia.org/wiki/Speech_processing)
* [Digital image processing](https://en.wikipedia.org/wiki/Digital_image_processing)
* [Video processing](https://en.wikipedia.org/wiki/Video_processing)[[56]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-56)
  + Hardware accelerated video decoding and post-processing
    - [Motion compensation](https://en.wikipedia.org/wiki/Motion_compensation) (mo comp)
    - Inverse [discrete cosine transform](https://en.wikipedia.org/wiki/Discrete_cosine_transform) (iDCT)
    - Variable-length decoding (VLD), [Huffman coding](https://en.wikipedia.org/wiki/Huffman_coding)
    - Inverse quantization ([IQ](https://en.wikipedia.org/wiki/IQ) (not to be confused by Intelligence Quotient))
    - In-loop deblocking
    - Bitstream processing ([CAVLC](https://en.wikipedia.org/wiki/CAVLC)/[CABAC](https://en.wikipedia.org/wiki/CABAC)) using special purpose hardware for this task because this is a serial task not suitable for regular GPGPU computation
    - [Deinterlacing](https://en.wikipedia.org/wiki/Deinterlacing)
      * Spatial-temporal deinterlacing
    - Noise reduction
    - Edge enhancement
    - Color correction
  + Hardware accelerated video encoding and pre-processing
* [Global illumination](https://en.wikipedia.org/wiki/Global_illumination) – [ray tracing](https://en.wikipedia.org/wiki/Ray_tracing_(graphics)), [photon mapping](https://en.wikipedia.org/wiki/Photon_mapping), [radiosity](https://en.wikipedia.org/wiki/Radiosity_(computer_graphics)) among others, [subsurface scattering](https://en.wikipedia.org/wiki/Subsurface_scattering)
* Geometric computing – [constructive solid geometry](https://en.wikipedia.org/wiki/Constructive_solid_geometry), distance fields, [collision detection](https://en.wikipedia.org/wiki/Collision_detection), transparency computation, shadow generation
* Scientific computing
  + [Monte Carlo simulation](https://en.wikipedia.org/wiki/Monte_Carlo_simulation) of light propagation[[57]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-Alerstam2009_JBO-57)
  + [Weather forecasting](https://en.wikipedia.org/wiki/Weather_forecasting)
  + [Climate research](https://en.wikipedia.org/wiki/Climate_research)
  + [Molecular modeling on GPU](https://en.wikipedia.org/wiki/Molecular_modeling_on_GPU)[[58]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-Hasan_Khondker_S._2014_pp._612%E2%80%9317-58)
  + [Quantum mechanical](https://en.wikipedia.org/wiki/Quantum_mechanical) physics
  + [Astrophysics](https://en.wikipedia.org/wiki/Astrophysics)[[59]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-59)
* [Bioinformatics](https://en.wikipedia.org/wiki/Bioinformatics)[[60]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-60)[[61]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-Manavski2008-61)
* [Computational finance](https://en.wikipedia.org/wiki/Computational_finance)
* [Medical imaging](https://en.wikipedia.org/wiki/Medical_imaging)
* [Clinical decision support system](https://en.wikipedia.org/wiki/Clinical_decision_support_system) (CDSS)[[62]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-62)
* [Computer vision](https://en.wikipedia.org/wiki/Computer_vision)[[63]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-63)
* [Digital signal processing](https://en.wikipedia.org/wiki/Digital_signal_processing) / [signal processing](https://en.wikipedia.org/wiki/Signal_processing)
* [Control engineering](https://en.wikipedia.org/wiki/Control_engineering)[[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)]
* [Operations research](https://en.wikipedia.org/wiki/Operations_research)[[64]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-64)[[65]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-65)[[66]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-BaumeltZdenek-66)
  + Implementations of: the GPU Tabu Search algorithm solving the Resource Constrained Project Scheduling problem is freely available on GitHub;[[67]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-67) the GPU algorithm solving the Nurse Rerostering problem is freely available on GitHub.[[68]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-68)
* [Neural networks](https://en.wikipedia.org/wiki/Neural_network)
* [Database](https://en.wikipedia.org/wiki/Database) operations[[69]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-69)[[70]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-70)[[71]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-71)
* [Computational Fluid Dynamics](https://en.wikipedia.org/wiki/Computational_Fluid_Dynamics) especially using [Lattice Boltzmann methods](https://en.wikipedia.org/wiki/Lattice_Boltzmann_methods)
* [Cryptography](https://en.wikipedia.org/wiki/Cryptography)[[72]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-72) and [cryptanalysis](https://en.wikipedia.org/wiki/Cryptanalysis)
* Performance modeling: computationally intensive tasks on GPU[[58]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-Hasan_Khondker_S._2014_pp._612%E2%80%9317-58)
  + Implementations of: [MD6](https://en.wikipedia.org/wiki/MD6), [Advanced Encryption Standard](https://en.wikipedia.org/wiki/Advanced_Encryption_Standard) (AES),[[73]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-73)[[74]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-74) [Data Encryption Standard](https://en.wikipedia.org/wiki/Data_Encryption_Standard) (DES), [RSA](https://en.wikipedia.org/wiki/RSA_(algorithm)),[[75]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-75) [elliptic curve cryptography](https://en.wikipedia.org/wiki/Elliptic_curve_cryptography) (ECC)
  + [Password cracking](https://en.wikipedia.org/wiki/Password_cracking)[[76]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-gtri-76)[[77]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-msnbc-77)
  + [Cryptocurrency](https://en.wikipedia.org/wiki/Cryptocurrency) transactions processing ("mining") ([Bitcoin mining](https://en.wikipedia.org/wiki/Bitcoin_network#Bitcoin_mining))
* [Electronic design automation](https://en.wikipedia.org/wiki/Electronic_design_automation)[[78]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-78)[[79]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-79)
* [Antivirus software](https://en.wikipedia.org/wiki/Antivirus_software)[[80]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-80)[[81]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-81)
* [Intrusion detection](https://en.wikipedia.org/wiki/Intrusion_detection)[[82]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-82)[[83]](https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#cite_note-83)
* Increase computing power for [distributed computing](https://en.wikipedia.org/wiki/Distributed_computing) projects like [SETI@home](https://en.wikipedia.org/wiki/SETI@home" \o "SETI@home), [Einstein@home](https://en.wikipedia.org/wiki/Einstein@home" \o "Einstein@home)

Further Must be read <https://linuxhint.com/gpu-programming/>