**Введение**

В современном IT-мире видеочипы уже давно заняли обширную нишу в вычислении больших объёмов данных. Однако, программирование для использование их мощностей до сих пор не приведено к каком-либо общим стандартам, как в области графических так и неграфических вычислений. Таким образом для решения конкретных задач необходимо понимать все преимущества и недостатки всех основных стандартом.

**1 Инструменты графического программирование на GPGPU**

**1.1 OpenGL**

OpenGL (Open Graphics Library — открытая графическая библиотека, графическое API) — спецификация, определяющая независимый от языка программирования платформонезависимый программный интерфейс для написания приложений, использующих двухмерную и трёхмерную компьютерную графику.

Включает более 250 функций для рисования сложных трёхмерных сцен из простых примитивов. Используется при создании компьютерных игр, САПР, виртуальной реальности, визуализации в научных исследованиях. На платформе Windows конкурирует с Direct3D.

**Спецификация**

На базовом уровне, OpenGL — это просто спецификация, то есть документ, описывающий набор функций и их точное поведение. Производители оборудования на основе этой спецификации создают реализации — библиотеки функций, соответствующих набору функций спецификации. Реализация использует возможности оборудования там, где это возможно. Если аппаратура не позволяет реализовать какую-либо возможность, она должна быть эмулирована программно. Производители должны пройти специфические тесты (conformance tests — тесты на соответствие) прежде чем реализация будет классифицирована как OpenGL реализация. Таким образом, разработчикам программного обеспечения достаточно научиться использовать функции, описанные в спецификации, оставив эффективную реализацию последних разработчикам аппаратного обеспечения.

Эффективные реализации OpenGL существуют для Windows, Unix-платформ, PlayStation 3 и Mac OS. Эти реализации обычно предоставляются изготовителями видеоадаптеров и активно используют возможности последних. Существуют также открытые реализации спецификации OpenGL, одной из которых является библиотека Mesa. Из лицензионных соображений Mesa является «неофициальной» реализацией OpenGL, хотя полностью с ней совместима на уровне кода и поддерживает как программную эмуляцию, так и аппаратное ускорение при наличии соответствующих драйверов.

Спецификация OpenGL пересматривается консорциумом ARB (Architecture Review Board), который был сформирован в 1992 году. Консорциум состоит из компаний, заинтересованных в создании широко распространённого и доступногоAPI. Согласно официальному сайту OpenGL, членами ARB с решающим голосом на ноябрь 2004 года являются производители профессиональных графических аппаратных средств SGI, 3Dlabs, Matrox и Evans & Sutherland (военные приложения), производители потребительских графических аппаратных средств ATI и NVIDIA, производитель процессоров Intel, и изготовители компьютеров и компьютерного оборудования IBM, Apple, Dell, Hewlett-Packard и Sun Microsystems, а также один из лидеров компьютерной игровой индустрии id Software. Microsoft, один из основоположников консорциума, покинула его в марте 2003 года. Помимо постоянных членов, каждый год приглашается большое количество других компаний, становящихся частью OpenGL ARB в течение одного года. Такое большое число компаний, вовлеченных в разнообразный круг интересов, позволило OpenGL стать прикладным интерфейсом широкого назначения с большим количеством возможностей.

Курт Экли (Kurt Akeley) и Марк Сигал (Mark Segal) являются авторами оригинальной спецификации OpenGL. Крис Фрэзиер (Chris Frazier) редактировал версию 1.1. Йон Лич (Jon Leech) редактировал версии с 1.2 по версию 2.0.

Архитектура OpenGL ориентируется на следующие две задачи:

* Скрыть сложности адаптации различных 3D-ускорителей, предоставляя разработчику единый API.
* Скрыть различия в возможностях аппаратных платформ, требуя реализации недостающей функциональности с помощью программной эмуляции.

Основным принципом работы OpenGL является получение наборов векторных графических примитивов в виде точек, линий и многоугольников с последующей математической обработкой полученных данных и построением растровой картинки на экране и/или в памяти. Векторные трансформации и растеризация выполняются графическим конвейером (graphics pipeline), который по сути представляет собой дискретный автомат. Абсолютное большинство команд OpenGL попадают в одну из двух групп: либо они добавляют графические примитивы на вход в конвейер, либо конфигурируют конвейер на различное исполнение трансформаций.

OpenGL является низкоуровневым процедурным API, что вынуждает программиста диктовать точную последовательность шагов, чтобы построить результирующую растровую графику (императивный подход). Это является основным отличием от дескрипторных подходов, когда вся сцена передается в виде структуры данных (чаще всего дерева), которое обрабатывается и строится на экране. С одной стороны, императивный подход требует от программиста глубокого знания законов трёхмерной графики и математических моделей, с другой стороны — даёт свободу внедрения различных инноваций.  
Расширения

Стандарт OpenGL, с появлением новых технологий, позволяет отдельным производителям добавлять в библиотеку функциональность через механизм расширений. Расширения распространяются с помощью двух составляющих: заголовочный файл, в котором находятся прототипы новых функций и констант, а также драйвер устройства, поставляемого разработчиком. Каждый производитель имеет аббревиатуру, которая используется при именовании его новых функций и констант. Например, компания NVIDIA имеет аббревиатуру NV, которая используется при именовании её новых функций, как, например, glCombinerParameterfvNV(), а также констант, GL\_NORMAL\_MAP\_NV. Может случиться так, что определённое расширение могут реализовать несколько производителей. В этом случае используется аббревиатура EXT, например, glDeleteRenderbuffersEXT. В случае же, когда расширение одобряется консорциумом ARB, оно приобретает аббревиатуру ARB и становится стандартным расширением. Обычно расширения, одобренные консорциумом, включаются в одну из следующих спецификаций OpenGL.  
Список зарегистрированных расширений можно найти в официальной базе расширений.

**Дополнительные библиотеки**

Существует ряд библиотек, созданных поверх или в дополнение к OpenGL. Например, библиотека GLU, являющаяся практически стандартным дополнением OpenGL и всегда её сопровождающая, построена поверх последней, то есть использует её функции для реализации своих возможностей. Другие библиотеки, как, например, GLUT и SDL, созданы для реализации возможностей, недоступных в OpenGL. К таким возможностям относятся создание интерфейса пользователя (окна, кнопки, меню и др.), настройка контекста рисования (область рисования, использующаяся OpenGL), обработка сообщений от устройств ввода/вывода (клавиатура, мышь и др.), а также работа с файлами. Обычно, каждый оконный менеджер имеет собственную библиотеку-расширение для реализации вышеописанных возможностей, например, WGL в Windows или GLX в X Window System, однако библиотеки GLUT и SDL являются кросс-платформенными, что облегчает перенос написанных приложений на другие платформы.

Библиотеки GLEW (The OpenGL Extension Wrangler Library) и GLEE (The OpenGL Easy Extension library) созданы для облегчения работы с расширениями и различными версиями OpenGL. Это особенно актуально для программистов в Windows, так как заголовочные и библиотечные файлы, поставляемые с Visual Studio, находятся на уровне версии OpenGL 1.1.

OpenGL имеет только набор геометрических примитивов (точки, линии, многоугольники) из которых создаются все трёхмерные объекты. Порой подобный уровень детализации не всегда удобен при создании сцен. Поэтому поверх OpenGL были созданы более высокоуровневые библиотеки, такие как Open Inventor и VTK. Данные библиотеки позволяют оперировать более сложными трёхмерными объектами, что облегчает и ускоряет создание трёхмерной сцены.

GLM (OpenGL Mathematics) — вспомогательная библиотека, предоставляющая программистам на C++ классы и функции для выполнения математических операций. Библиотека может использоваться при создании 3D-программ с использованием OpenGL. Одной из характеристик GLM является то, что реализация основана на спецификации GLSL. Исходный код GLM использует лицензию MIT.

**Независимость от языка программирования**

Для подтверждения независимости от языка программирования были разработаны различные варианты привязки (binding) функций OpenGL или полностью перенесены на другие языки. Одним из примеров может служить библиотека Java 3D, которая может использовать аппаратное ускорение OpenGL. Прямая привязка функций реализована в Lightweight Java Game Library, которая имеет прямую привязку OpenGL для Java. Sun также выпустила версию Java OpenGL(JOGL), которая предоставляет прямую привязку к Си-функциям OpenGL, в отличие от Java 3D, которая не имеет столь низкоуровневой поддержки. Официальный сайт OpenGL имеет ссылки на привязки для языков Java, Фортран 90, Perl,Pike, Python, Ada, Visual Basic и Pascal. Имеются также варианты привязки OpenGL для языков C++ и C#.

**1.2 OpenCL**

OpenCL (от [англ.](https://ru.wikipedia.org/wiki/Английский_язык) Open Computing Language — открытый язык вычислений) — [фреймворк](https://ru.wikipedia.org/wiki/Фреймворк) для написания [компьютерных программ](https://ru.wikipedia.org/wiki/Компьютерная_программа), связанных с [параллельными вычислениями](https://ru.wikipedia.org/wiki/Параллельные_вычисления) на различных [графических](https://ru.wikipedia.org/wiki/Графический_процессор) ([англ.](https://ru.wikipedia.org/wiki/Английский_язык)GPU) и [центральных процессорах](https://ru.wikipedia.org/wiki/Центральный_процессор) ([англ.](https://ru.wikipedia.org/wiki/Английский_язык)CPU), а также [FPGA](https://ru.wikipedia.org/wiki/FPGA). Вo фреймворк OpenCL входят [язык программирования](https://ru.wikipedia.org/wiki/Язык_программирования), который базируется на стандарте [C99](https://ru.wikipedia.org/wiki/C99), и [интерфейс программирования приложений](https://ru.wikipedia.org/wiki/Интерфейс_программирования_приложений) ([англ.](https://ru.wikipedia.org/wiki/Английский_язык) API). OpenCL обеспечивает параллелизм на уровне инструкций и на уровне данных и является реализацией техники [GPGPU](https://ru.wikipedia.org/wiki/GPGPU). OpenCL является полностью [открытым стандартом](https://ru.wikipedia.org/wiki/Открытый_стандарт), его использование не облагается лицензионными отчислениями.

Цель OpenCL состоит в том, чтобы дополнить [OpenGL](https://ru.wikipedia.org/wiki/OpenGL) и [OpenAL](https://ru.wikipedia.org/wiki/OpenAL), которые являются открытыми отраслевыми стандартами для [трёхмерной компьютерной графики](https://ru.wikipedia.org/wiki/Трёхмерная_графика) и звука, пользуясь возможностями [GPU](https://ru.wikipedia.org/wiki/GPU). OpenCL разрабатывается и поддерживается некоммерческим консорциумом [Khronos Group](https://ru.wikipedia.org/wiki/Khronos_Group), в который входят много крупных компаний, включая [AMD](https://ru.wikipedia.org/wiki/AMD), [Apple](https://ru.wikipedia.org/wiki/Apple), [ARM](https://ru.wikipedia.org/wiki/ARM_(компания)), [Intel](https://ru.wikipedia.org/wiki/Intel), [Nvidia](https://ru.wikipedia.org/wiki/Nvidia), [Sony Computer Entertainment](https://ru.wikipedia.org/wiki/Sony_Computer_Entertainment), [Sun Microsystems](https://ru.wikipedia.org/wiki/Sun_Microsystems) и другие.

OpenCL первоначально был разработан в компании Apple Inc. Apple внесла предложения по разработке спецификации в комитет Khronos. Вскоре компания [AMD](https://ru.wikipedia.org/wiki/AMD)решила поддержать разработку OpenCL (и [DirectX 11](https://ru.wikipedia.org/wiki/DirectX_11)), который должен заменить фреймворк [Close to Metal](https://ru.wikipedia.org/wiki/Close_to_Metal).

OpenCL 1.1 был представлен организацией Khronos Group 14 июня [2010 года](https://ru.wikipedia.org/wiki/2010_год). В новой версии значительно расширены функциональные возможности для параллельного программирования, гибкость и производительность, а также добавлены новые возможности:

* Новые типы данных, включая 3-компонентные векторы и дополнительные форматы изображений.
* Обработка команд из нескольких потоков хоста и обработки буфера между несколькими устройствами.
* Операции по регионам буфера включая чтение, запись и копирование 1D, 2D или 3D прямоугольных областей.
* Расширенное использование события для управления и контроля выполнения команд.
* Улучшенное взаимодействие с OpenGL за счет эффективного обмена изображениями.
* Партицирование устройств — возможность разбиения на уровне OpenCL-приложения устройства на несколько подустройств для непосредственной привязки работ к конкретным вычислительным блокам, резервирования ресурсов для более приоритетных задач или более эффективного совместного использования аппаратных ресурсов, таких как кэш.
* Раздельная компиляция и связывание объектов — появилась возможность создания динамических библиотек, позволяющих использовать в сторонних программах, ранее реализованные подпрограммы с OpenCL-вычислениями.
* Расширенная поддержка изображений, включая возможность работы с одномерными изображениями и массивами одномерных или двухмерных изображений. Кроме того, в расширении для организации совместного доступа (sharing) добавлена возможность создания OpenCL-изображения на основе отдельных текстур OpenGL или массивов текстур.
* Встроенные OpenCL-ядра теперь позволяют использовать возможности специализированного или непрограммируемого аппаратного обеспечения и связанных с ним прошивок. Например, появилась возможность использования возможностей и более тесной интеграции с фреймворком OpenCL таких устройств, как DSP-процессоры или видео кодировщики/декодировщики.
* Возможность бесшовного совместного использования поверхностей (Media Surface Sharing) между OpenCL и API DirectX 9/11.
* OpenCL 2.0 был представлен 22 июля 2013 год и стандартизирован 18 ноября того же года.

Особенности языка

Ключевыми отличиями используемого языка от Си (стандарт ISO 1999 года) являются:

* отсутствие поддержки указателей на [функции](https://ru.wikipedia.org/wiki/Функция_(программирование)), [рекурсии](https://ru.wikipedia.org/wiki/Рекурсия), [битовых полей](https://ru.wikipedia.org/wiki/Битовое_поле), массивов переменной длины (VLA), стандартных заголовочных файлов;
* расширения языка для параллелизма: векторные типы, синхронизация, функции для Work-items/Work-Groups;
* квалификаторы типов паияти: \_\_global, \_\_local, \_\_constant, \_\_private;
* иной набор встроенных функций.

**1.3 DirectX**

DirectX (от англ. direct — прямой, непосредственный) — это набор API, разработанных для решения задач, связанных с программированием под Microsoft Windows. Наиболее широко используется при написании компьютерных игр. Пакет средств разработки DirectX под Microsoft Windows бесплатно доступен на сайте Microsoft. Зачастую обновленные версии DirectX поставляются вместе с игровыми приложениями.

**DirectX API**

Практически все части DirectX API представляют собой наборы COM-совместимых объектов.

В целом, DirectX подразделяется на:

* DirectX Graphics, набор интерфейсов, ранее (до версии 8.0) делившихся на:
* DirectDraw : интерфейс вывода растровой графики. (Его разработка давно прекращена)
* Direct3D (D3D): интерфейс вывода трёхмерных примитивов.
* DirectInput: интерфейс, используемый для обработки данных, поступающих с клавиатуры, мыши, джойстика и пр. игровых контроллеров.
* DirectPlay: интерфейс сетевой коммуникации игр.
* DirectSound: интерфейс низкоуровневой работы со звуком (формата Wave)
* DirectMusic: интерфейс воспроизведения музыки в форматах Microsoft.
* DirectShow: интерфейс, используемый для ввода/вывода аудио и/или видео данных.
* DirectX Instruments — технология, позволяющая на основе мультимедийного API DirectX создавать и использовать программные синтезаторы. В отличие от DX-плагинов, такие программы могут полностью управляться по MIDI и служат главным образом не для обработки, а для синтеза звука. Технология DXi была популярна в 2001—2004 гг., особенно в программных продуктах Cakewalk, но со временем проиграла «войну форматов» технологии VST отSteinberg.
* DirectSetup: часть, ответственная за установку DirectX.
* DirectX Media Objects: реализует функциональную поддержку потоковых объектов (например, кодировщики/декодировщики)
* Direct2D : интерфейс вывода двухмерной графики



Рисунок - DirectX Taxonomy

**Direct3D 10** — набор [API](https://ru.wikipedia.org/wiki/API) функций для взаимодействия с видеокартой; поддерживается аппаратно видеокартами класса NV [GeForce](https://ru.wikipedia.org/wiki/GeForce) 8x00, ATI [Radeon](https://ru.wikipedia.org/wiki/Radeon) 2x00 и выше. Direct3D 10 (D3D10) — компонент интерфейса программирования приложений (англ. API) DirectX 10, 10-я версия Direct3D, преемник Direct3D 9. Direct3D 10 обеспечивает функции для взаимодействия операционной системы и приложений с драйверами видеокарты. Эти функции привязаны к операционной системе в линейке Windows и доступны в [Windows Vista](https://ru.wikipedia.org/wiki/Windows_Vista) и [Windows 7](https://ru.wikipedia.org/wiki/Windows_7). Частично D3D10 работает на видеокартах уровня Direct3D 9.

**2 Инструменты неграфического программирования на GPGPU**

**2.1 ATI Stream technology**

**ATI Stream Technology** (ранее как [*ATI*](https://ru.wikipedia.org/wiki/ATI) *FireStream* и *AMD Stream Processor*) — API с открытым исходным кодом, которое предоставляет возможность разработчикам использовать вычислительные возможности GPU . Он предоставляет возможность использования шейдеров графического процессора для запуска вычислительных программ. Интерфейс программирования осуществляется через [OpenCL](https://ru.wikipedia.org/wiki/OpenCL). Это дает возможность ускорения вычислений, и может быть использовано, в том числе, в игровой сфере, для ускорения просчетов физики, если движок физики поддерживает OpenCL.

Областями применения ATI Stream являются также приложения, требовательные к вычислительному ресурсу, такие, как финансовый анализ или обработка сейсмических данных. Использование потокового процессора позволило увеличить скорость некоторых финансовых расчётов в 55 раз по сравнению с решением той же задачи силами только центрального процессора.

**2.2 CUDA**

**CUDA** ([англ.](https://ru.wikipedia.org/wiki/Английский_язык) *Compute Unified Device Architecture*) — программно-аппаратная архитектура [параллельных вычислений](https://ru.wikipedia.org/wiki/Параллельные_вычисления), которая позволяет существенно увеличить вычислительную производительность благодаря использованию [графических процессоров](https://ru.wikipedia.org/wiki/Графический_процессор) фирмы [Nvidia](https://ru.wikipedia.org/wiki/Nvidia).

CUDA [SDK](https://ru.wikipedia.org/wiki/SDK) позволяет программистам реализовывать на специальном упрощённом диалекте [языка программирования Си](https://ru.wikipedia.org/wiki/Си_(язык_программирования)) алгоритмы, выполнимые на графических процессорах Nvidia, и включать специальные функции в текст программы на Си. Архитектура CUDA даёт разработчику возможность по своему усмотрению организовывать доступ к набору инструкций графического ускорителя и управлять его памятью.

**Программная архитектура**

Первоначальная версия CUDA [SDK](https://ru.wikipedia.org/wiki/SDK) была представлена [15 февраля](https://ru.wikipedia.org/wiki/15_февраля) [2007 года](https://ru.wikipedia.org/wiki/2007_год). В основе [интерфейса программирования приложений](https://ru.wikipedia.org/wiki/Интерфейс_программирования_приложений) CUDA лежит язык [Си](https://ru.wikipedia.org/wiki/Си_(язык_программирования)) с некоторыми расширениями. Для успешной трансляции кода на этом языке в состав CUDA SDK входит собственный Си-компилятор командной строки **nvcc** компании Nvidia. Компилятор **nvcc** создан на основе открытого компилятора [Open64](https://ru.wikipedia.org/wiki/Open64) и предназначен для трансляции host-кода (главного, управляющего кода) и device-кода (аппаратного кода) (файлов с расширением .cu) в объектные файлы, пригодные в процессе сборки конечной программы или библиотеки в любой среде программирования, например, в [NetBeans](https://ru.wikipedia.org/wiki/NetBeans).

В архитектуре CUDA используется модель памяти [грид](https://ru.wikipedia.org/wiki/Грид), [кластерное моделирование](https://ru.wikipedia.org/wiki/Кластер_(группа_компьютеров)) [потоков](https://ru.wikipedia.org/wiki/Многопоточность) и [SIMD](https://ru.wikipedia.org/wiki/SIMD)-инструкции. Применима не только для высокопроизводительных графических вычислений, но и для различных научных вычислений с использованием видеокарт nVidia. Ученые и исследователи широко используют CUDA в различных областях, включая [астрофизику](https://ru.wikipedia.org/wiki/Астрофизика), [вычислительную биологию](https://ru.wikipedia.org/wiki/Вычислительная_биология) и [химию](https://ru.wikipedia.org/wiki/Вычислительная_химия), [моделирование динамики жидкостей](https://ru.wikipedia.org/wiki/Моделирование_жидкости), [электромагнитных взаимодействий](https://ru.wikipedia.org/wiki/Электромагнитное_взаимодействие), [компьютерную томографию](https://ru.wikipedia.org/wiki/Компьютерная_томография), [сейсмический анализ](https://ru.wikipedia.org/w/index.php?title=Сейсмический_анализ&action=edit&redlink=1) и многое другое. В CUDA имеется возможность подключения к приложениям, использующим[OpenGL](https://ru.wikipedia.org/wiki/OpenGL) и [Direct3D](https://ru.wikipedia.org/wiki/Direct3D). CUDA — [кроссплатформенное программное обеспечение](https://ru.wikipedia.org/wiki/Кроссплатформенное_программное_обеспечение) для таких [операционных систем](https://ru.wikipedia.org/wiki/Операционная_система) как [Linux](https://ru.wikipedia.org/wiki/Linux), [Mac OS X](https://ru.wikipedia.org/wiki/Mac_OS_X) и [Windows](https://ru.wikipedia.org/wiki/Windows).

**Оборудование**

Платформа CUDA впервые появились на рынке с выходом [чипа NVIDIA восьмого поколения](https://ru.wikipedia.org/wiki/GeForce_8) G80 и стала присутствовать во всех последующих сериях графических чипов, которые используются в семействах ускорителей [GeForce](https://ru.wikipedia.org/wiki/GeForce), [Quadro](https://ru.wikipedia.org/wiki/Nvidia_Quadro) и [NVidia Tesla](https://ru.wikipedia.org/wiki/NVidia_Tesla).

Первая серия оборудования, поддерживающая CUDA SDK, G8x, имела 32-битный [векторный процессор](https://ru.wikipedia.org/wiki/Векторный_процессор) [одинарной точности](https://ru.wikipedia.org/wiki/Число_одинарной_точности), использующий CUDA SDK как API (CUDA поддерживает тип double языка Си, однако сейчас его точность понижена до 32-битного с [плавающей запятой](https://ru.wikipedia.org/wiki/Числа_с_плавающей_запятой)). Более поздние процессоры GT200 имеют поддержку 64-битной точности (только для SFU), но производительность значительно хуже, чем для 32-битной точности (из-за того, что SFU всего два на каждый потоковый мультипроцессор, а скалярных процессоров — восемь). Графический процессор организует аппаратную многопоточность, что позволяет задействовать все ресурсы графического процессора. Таким образом, открывается перспектива переложить функции [физического ускорителя](https://ru.wikipedia.org/wiki/Физический_ускоритель) на графический ускоритель (пример реализации — [PhysX](https://ru.wikipedia.org/wiki/PhysX)). Также открываются широкие возможности использования графического оборудования компьютера для выполнения сложных неграфических вычислений: например, в [вычислительной биологии](https://ru.wikipedia.org/wiki/Вычислительная_биология) и в иных отраслях науки.

**Преимущества**

По сравнению с традиционным подходом к организации вычислений общего назначения посредством возможностей графических API, у архитектуры CUDA отмечают следующие преимущества в этой области:

* [Интерфейс программирования приложений](https://ru.wikipedia.org/wiki/Интерфейс_программирования_приложений) CUDA (CUDA API) основан на стандартном языке программирования Си с некоторыми ограничениями. По мнению разработчиков, это должно упростить и сгладить процесс изучения архитектуры CUDA
* Разделяемая между потоками память (shared memory) размером в 16 Кб может быть использована под организованный пользователем кэш с более широкой полосой пропускания, чем при выборке из обычных текстур
* Более эффективные транзакции между памятью центрального процессора и видеопамятью
* Полная аппаратная поддержка целочисленных и побитовых операций
* Поддержка компиляции GPU кода средствами открытого [LLVM](https://ru.wikipedia.org/wiki/Low_Level_Virtual_Machine)

## Ограничения

* Все функции, выполнимые на устройстве, не поддерживают рекурсии (в версии CUDA Toolkit 3.1 поддерживает указатели и рекурсию) и имеют некоторые другие ограничения

**2.3 OpenMP**

OpenMP (Open Multi-Processing) — открытый стандарт для [распараллеливания программ](https://ru.wikipedia.org/wiki/Распараллеливание_программ) на языках [Си](https://ru.wikipedia.org/wiki/С_(язык_программирования)), [Си++](https://ru.wikipedia.org/wiki/Си%2B%2B) и [Фортран](https://ru.wikipedia.org/wiki/Фортран). Дает описание совокупности [директив компилятора](https://ru.wikipedia.org/wiki/Директива_(программирование)), библиотечных [процедур](https://ru.wikipedia.org/wiki/Процедура_(программирование)) и [переменных окружения](https://ru.wikipedia.org/wiki/Переменные_среды), которые предназначены для программирования [многопоточных приложений](https://ru.wikipedia.org/wiki/Многопоточность) на [многопроцессорных системах](https://ru.wikipedia.org/wiki/Симметричная_мультипроцессорность) с [общей памятью](https://ru.wikipedia.org/wiki/Разделяемая_память).

**2.4 CuDNN**

Библиотека NVIDIA CUDA® Deep Neural Network (cuDNN) является GPU-ускорением библиотека примитивов для [глубоких нейронных сетей](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&prev=search&rurl=translate.google.ru&sl=en&u=https://developer.nvidia.com/deep-learning&usg=ALkJrhifPGu9cPbtYnQRRY78SCc1yoBi5w). cuDNN обеспечивает очень точную настройку реализации для стандартных процедур, таких как переднюю и заднюю свертке, объединение, нормализации и активации слоев. cuDNN является частью[NVIDIA Deep Learning SDK](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&prev=search&rurl=translate.google.ru&sl=en&u=https://developer.nvidia.com/deep-learning-sdk&usg=ALkJrhjE0JzYAWN0H2LhBgYs0yd0zieU-Q).

Глубокие исследователи обучения и рамочные разработчики по всему миру полагаются на cuDNN для высокопроизводительных GPU ускорения. Это позволяет им сосредоточиться на обучении нейронных сетей и разработке программных приложений, а не тратить время на настройку производительности GPU на низком уровне. cuDNN ускоряет широко используемые глубокие основы обучения, в том числе [Caffe](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&prev=search&rurl=translate.google.ru&sl=en&u=http://caffe.berkeleyvision.org/&usg=ALkJrhhvNJd-vA509Q6gVZoOkYM9OE7T6g), [TensorFlow](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&prev=search&rurl=translate.google.ru&sl=en&u=https://www.tensorflow.org/&usg=ALkJrhi_ONqzMslWTOkC_4qZdruhtBnDHg), [Теано](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&prev=search&rurl=translate.google.ru&sl=en&u=http://deeplearning.net/software/theano/&usg=ALkJrhhW54WeaicFLS8n4bJ2pcMGGEJFRg), [Факел](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&prev=search&rurl=translate.google.ru&sl=en&u=http://torch.ch/&usg=ALkJrhhUM9VB9lSKNCqrGG6nd5t51NlvlQ) и [CNTK](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&prev=search&rurl=translate.google.ru&sl=en&u=http://www.cntk.ai/&usg=ALkJrhi8dug7UPX4lO7juKzgFrWiGpMRtw).

cuDNN находится в свободном доступе для членов [Accelerated программы Developer Computing](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&prev=search&rurl=translate.google.ru&sl=en&u=https://developer.nvidia.com/accelerated-computing-developer&usg=ALkJrhhJFoZaKt4Q3dqobtzOmfSKa5W9sA).

Ученые и исследователи данных могут воспользоваться cuDNN путем загрузки [Deep рамки обучения](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&prev=search&rurl=translate.google.ru&sl=en&u=https://developer.nvidia.com/deep-learning-frameworks&usg=ALkJrhjocRBnQ5vYVomDUxJ_cYkfbS_6CQ) или [NVIDIA Digits](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&prev=search&rurl=translate.google.ru&sl=en&u=https://developer.nvidia.com/digits&usg=ALkJrhjCJ5ZbqLwUh_eR61ytLNkra6lvrQ). Digits позволяет в интерактивном режиме управлять данными, выполнять обучение на нескольких графических процессоров, а также экспортировать наилучшую модель исполнительскую для развертывания без необходимости написания кода.

Ключевая особенность

* Прямые и обратные пути для многих распространенных типов слоев, таких как пулинговой, ЛРН, LCN и пакетной нормализации, Relu, сигмовидной, SoftMax и TANH
* Прямые и обратные свертках процедуры, в том числе кросс-корреляции, предназначенный для сверточных нейронных сетей
* Повторяющиеся Neural Networks (LSTM / ГРУ / РНН), которые обеспечивают до 6-кратного ускорения в Факеле
* Произвольное упорядочение измерение, беговое и субрегионов для 4d тензоров означает простую интеграцию в любой нейронной сети реализации
* Прямые и обратные пути для многих распространенных типов слоев, таких как пулинговой, РЕЛУ, сигмовидной, SoftMax и TANH
* Функции преобразования тензор
* Контекст основе API позволяет легко многопоточности

cuDNN поддерживается на Windows, Linux и MacOS систем с Паскаля, Кеплера, Максвелла, [Tegra K1 или Tegra X1 графических процессоров.](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&prev=search&rurl=translate.google.ru&sl=en&u=https://developer.nvidia.com/embedded/jetpack&usg=ALkJrhgcDuqb_Vu-fNQ37EiwBy2A-rT-Rg)

**2.5 ArrayFire**

ArrayFire – это быстрая библиотека программ для вычислений на графических процессорах (GPU computing) с простым в использовании программным интерфейсом. ArrayFire предоставляет набор функций на базе массива, что существенно упрощает GPU-разработку. Продукт доступен для C, C++ и Fortran, и интегрируется с оборудованием AMD, Intel и NVIDIA.

Библиотека ArrayFire крайне проста в работе. Несколько строк кода в ArrayFire могут заменить десятки строк сырого GPU-кода, что позволит существенно сэкономить время и снизить затраты на разработку.

Преимущества:

* Ускорение процесса написания кода.
* Удобный и простой интерфейс.
* Широкий выбор графических функций.
* Совместимость с любым оборудованием.
* Наглядное графическое отображение функций.

**Список используемых источников**

1. <http://optic.cs.nstu.ru/files/GPU/05.html> \*(OpenGL, DirectX)
2. <https://ru.wikipedia.org/wiki/OpenCL> \*(OpenCL)
3. <https://ru.wikipedia.org/wiki/AMD_FireStream> \*(ATI Stream Technology)
4. <https://ru.wikipedia.org/wiki/CUDA> \*(CUDA)
5. <http://www.thg.ru/graphic/ati_stream/ati_stream-01.html> \*(Тесты)
6. <https://ru.wikipedia.org/wiki/OpenMP> \*(OpenMP)
7. <http://www.softkey.ru/catalog/programs/142925/arrayfire#.V3oYc9_S2b8> \*(ArrayFire)
8. <https://developer.nvidia.com/cudnn> \*(CuDNN)