# Курсовая работа по Чуканову (GPGPU)

Что делать сначала:

1. Установить любую достаточно свежую версию Visual Studio (у меня 2012 Express), если ее еще нет.
2. Установить CUDA SDK (первая ссылка в гугле, там должно быть developers.nvidia.com…), по ходу установки привязав его к установленной Visual Studio. Там, кажется, в самом начале установки он скажет, что нашел такую-то версию студии и попросит подтвердить.
3. Вполне может быть такое, что он вашу студию не найдет – у меня так было, потому что у меня не полная версия студии а Express. Если так, то все равно продолжаем установку CUDA SDK, просто потом придется немного похимичить (https://devtalk.nvidia.com/default/topic/589217/vs2012-express-and-cuda-5-5-error).
4. Если на вашем компьютере нет видеокарты NVDIA (скорее всего, об этом есть наклейка на нем самом; либо можно так: Пуск – Выполнить - control /name Microsoft.DeviceManager – Видеоадаптеры), то вам придется пользоваться функциями OpenCL, CUDA SDK их поддерживает. Иначе вам повезло и можно пользоваться функциями CUDA, но как это делать, я не знаю, поэтому писать буду про OpenCL.
5. Для того, чтобы проверить, что OpenCL нормально работает, можно запустить проект из архива OpenCL-learning-VS2012-master. Его надо распаковать, открыть в студии, а потом изменить настройки проекта, чтобы подключить заголовочные файлы и библиотеки из CUDA SDK (https://medium.com/@pratikone/opencl-on-visual-studio-configuration-tutorial-for-the-confused-3ec1c2b5f0ca).
6. По идее, все должно пройти хорошо и проект скомпилируется, но после запуска сразу умрет. Это нормально. Чтобы починить, в строке 51 меняем platforms[1] на platforms[0]. Все должно случиться и на экране красиво сложатся два массива.

Что делать потом:

Сама задача такая – надо напрограммировать одну из двух задач:

1. Визуализировать систему частиц, которые бегают по экрану, сталкиваясь и отталкиваюсь.
2. Визуализировать систему частиц, которая под действием гравитации падает на наклонную плоскость, частицы потом отталкиваются и куда-то бегут.

В обоих случаях в систему все время поступают новые частицы, на каждом шаге по времени.

Чтобы понять, как это примерно может выглядеть, в архиве есть пример oclNbody.zip. По-хорошему, его можно собрать, но у меня не получилось. Но, если залезть в папку \NVIDIA GPU Computing SDK\OpenCL\bin\win64\Debug, там лежит exe-шник, которые можно запустить. Если все в порядке, то там появится красивая картина. Там частицы не сталкиваются, а как-то хаотически движутся туда-сюда.

После того, как каким-то образом эта задача будет запрограммирована, нужно будет оценить, насколько хорошо все написано. Это делается с помощью программы nvvp.exe, которая устанавливается вместе с CUDA SDK. Там как-то надо химичить, чтобы в нее можно было пихать OpenCL-приложения, я пока с этим еще не заморачивался.

Так вот, мы запихиваете в nvvp свою написанную программу и nvvp говорит, что чем эта программа плоха. Например, она памяти много ест или еще что. Вы это исправляете, прогоняете программу через nvvp еще раз и nvvp должен показать, что вы все исправили и все хорошо.

Вроде, на этом все.

Продолжение (от 18.12)

Оказалось, что не все.

Если вы счастливый обладатель карты NVIDIA, то nvvp у вас будет работать. А если нет – то он вам не поможет и вам нужно другое ПО для анализа программы.

Для тех, у кого карта AMD, такое есть – называется CodeXL. Вроде даже говорят, что помогает. Но карта AMD (как и NVIDIA) – это большое счастье, которое не у всех есть, в частности, на той машине, где я работаю, такой нет.

На моей машине установлена графическая карта от Intel. Поэтому для анализа я скачал программу Intel VTune Amplifier XE 2016 (ссылка на рутрекер <https://dostup.website/http://rutracker.org/forum/viewtopic.php?t=5174470>). Запускаете ее, там жмем Basic Hotspots Analysis и отмечаем галкой Analyze GPU usage.

После того, как вы скормите ему ваше приложение, потом посмотрите, как оно работает, и вырубите, он выплюнет немного данных. Если у вас хороший процессор, можно попробовать попросить его следить за OpenCL вызовами (галка чуть ниже, где Analyze GPU usage), и он выплюнет много информации, которая реально полезна. Если процессор не очень – он за OpenCL следить не умеет и привет.

Содержательная информация находится на вкладке Graphics, там внизу есть много кривых, одна из них называется GPU Computing Threads… Чем она больше, тем больше производительность. После того, как вы поменяете код, эта кривая должна стать выше и среднее значение, которое там пишется, должно вырасти.

По идее, код нужно менять в зависимости от того, какие данные вам выдал Amplifier. Он должен говорить, как хорошо используется память, как много стоят потоки и т.д. Мне он этих данных не выдал, поэтому код я менял из общих соображений (они есть в лекции #7). И даже что-то получилось. Надеюсь, у вас тоже получится)

Обновление (от 20.12)

Выложил свою лабу (NBodyProblem.zip), должна работать, если поставить CUDA SDK и DirectX11. Чтобы проверить, что DirtectX11 правильно установился (его надо сначала установить), советую взять по вот этой ссылке тьюториалы по работе с ним и проверить, что они как-то собираются (<https://sites.google.com/site/primatcg/laboratornye-1/tasks/zadanie0podgotovka>). Если не собираются, то не хватило библиотек и надо качать DirtectX SDK (<https://www.microsoft.com/en-us/download/details.aspx?id=6812>). Но по идее, должно хватать, это я на всякий случай.