BASEMARK CL SPECIFICATION

Rauli Laatikainen

Teemu Virolainen

Rightware 2011

Version history		
20.1.2011	1.2	Release version. All references of OpenCLBenchmark are now BasemarkCL
18.10.2011	1.1	Added more information about how different scores are shown in the report. Cosmetic changes to make score reporting clearer.
13.10.2011	1.0	Release candidate version.

Table of Contents

RIGHTWARE OY, NIITTYKATU 6 C
FIN-02200 ESPOO, FINLAND
TEL. +358 9 8554322
FAX. +358 9 8554323
VAT.NO: 2307199-5

Introduction

This specification presents Rightware's Benchmark for OpenCL 1.1 (Khronos Group, 2008), Basemark CL. The benchmark is aimed to test full profile of OpenCL with most tests designed in Embedded Profile in mind. Basemark CL uses OpenGL ES 2.0 (Khronos Group, 2009) or OpenGL 2.1 for rendering depending on the platform.

OpenCL is Khronos Group's framework, which allows developers to leverage the processing power of Central processing units (CPU), Graphic processing units (GPU) and other processing units for general purpose computation.

Benchmarking the OpenCL device performance has become important as OpenCL is already available on desktop platforms and the powerful GPUs and multicore CPUs are also already available in mobile phones and other embedded devices.

The Basemark CL gives developers and hardware manufacturers the necessary tools to analyze platform capabilities for parallel computation.

Benchmark Technology

Benchmarking

Basemark CL consists of multiple tests. The tests are divided into three categories: Image manipulation tests, physics simulation tests and general tests.

Image manipulation tests run image filters on image and video data. Image manipulation tests can be used to benchmark bandwidth or calculation capabilities. Physics simulation tests run physics simulations and visualize the result with OpenGL. The general tests aim to test general capabilities of the platform such as speed of online compiler or raw arithmetic performance.

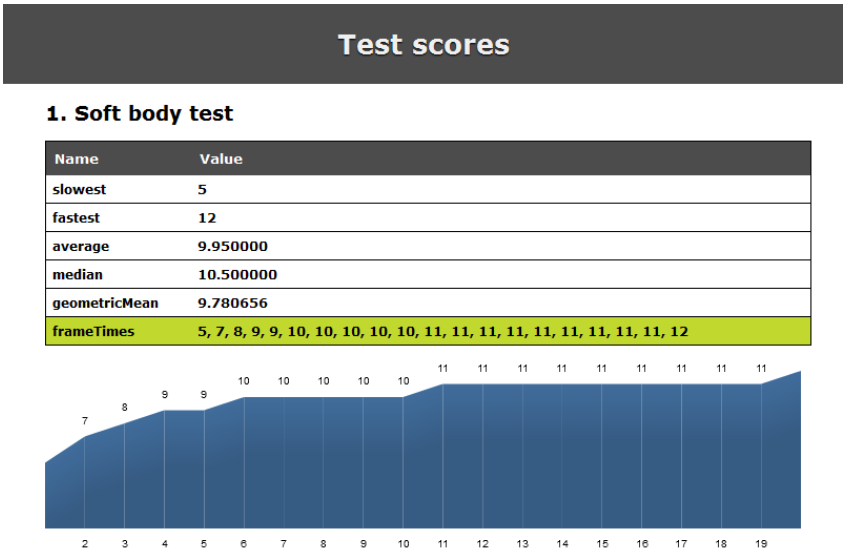
The purpose of Basemark CL is not to supply an optimal solution to any of the featured algorithms or techniques the benchmark employs, but to provide user an objective tool that one may use to compare the performances of different hardware platforms and API implementations. The tests try to use the language built-in functionality as much as possible, which makes it possible for driver developers to benchmark driver implementations.

Benchmarking data is collected by internal timers within the program. In Basemark CL timing is only done for the OpenCL calls. Rightware does not calculate the time spent on OpenGL rendering or other tasks. After a benchmark run the collected data is stored in XML files. Report generation and contents are described in detail in the section below.

RIGHTWARE OY, NIITTYKATU 6 C
FIN-02200 ESPOO, FINLAND
TEL. +358 9 8554322
FAX. +358 9 8554323
VAT.NO: 2307199-5

Reporting

During the benchmark run the benchmark stores the results internally. After the benchmark run is over this data is stored in XML log file. The XML file can be read directly or opened with web browser. We provide a XLS file, which produces a browser viewable document from XML and visualizes the available data with graphs and tables.



Test scores

1. Soft body test

Name	Value
slowest	5
fastest	12
average	9.950000
median	10.500000
geometricMean	9.780656
frameTimes	5, 7, 8, 9, 9, 10, 10, 10, 10, 10, 11, 11, 11, 11, 11, 11, 11, 11, 11, 12

Sample of the test scores

The benchmark report also contains information about the used platform, OpenCL platform, OpenCL capable devices, OpenGL settings and EGL/WGL settings.

Scores and Parameters

The tests are scored based on time used in completing the test workload. The tests are subdivided into small units are called frames. We track the time used to complete each frame and visualize this as a graph on the reports. Frame contains only the time spent on OpenCL related calls – time spent in OpenGL rendering and other operations is disregarded.

The benchmark always uses NULL as local workgroup size for OpenCL kernels, thus allowing the device to select the size. This was selected as the OpenCL platforms are very heterogeneous, thus making an independent fair and useful choice of local size practically impossible.

All kernels are compiled with flags –cl-strict-aliasing and –cl-fast-relaxed-math. The flags were selected to represent a desire to get the best performance out of the system on algorithms which are not sensitive to IEEE specific behavior. As these flags simply allow the implementation to do certain types of optimizations, it does not mandate them; there is no reason to run the tests without them.

Benchmark Score

The individual test scores are calculated as geometric mean of inverse of frame durations (measured in seconds).

The overall benchmark score is computed as weighted geometric mean of the individual test scores.

$$OpenCL\ Benchmark\ Score = \left(\prod_{i=1}^{n} Score_i^{\,Weight_i}\right)^{\frac{1}{\sum Weights}}$$

The weights are as follows:

Test	Weight
Soft Body Simulation	1
Fluid Simulation	1
SPH Fluid	1
Wave Simulation	1
All Image Manipulation tests combined	2
All Video Manipulation tests combined	1
3D Mandelbulb	0.5
Julia Fractal	0.5

The weights are a balance of test coverage with the estimated normal use cases. As the physics tests have the most variable workloads they have the largest weight.

Any quantitative comparison of scores should be done by calculating the ratio of the scores [J. E. Smith. 1988].

The main score for the benchmark requires OpenGL and OpenCL interoperability and full profile support. The requirements for receiving official Benchmark Score are described in “Run Rules” section. The final report received after running the benchmark contains information whether OpenGL interoperability was used and whether full or embedded profile was used.

As OpenCL is used in many devices and applications with no OpenGL support we also provide a score for those devices. This functionality is provided as a standalone program which does not require graphical user interface at all. This version of the benchmark executes tests without rendering to screen and scores them. The score received this way is not comparable with scores reported by OpenGL and OpenCL interoperability version.

We also provide additional Embedded Profile support for testing purposes. Embedded profiles runs a slightly modified set of tests and contains versions of kernels and tests which have smaller workload. If you wish to receive score for embedded profile you should set “OpenCLFullProfile” –flag to 0 in application.cfg file. The actual workloads of the Embedded Profile tests are subject to change and should not be used to compare implementations.

RIGHTWARE OY, NIITTYKATU 6 C
FIN-02200 ESPOO, FINLAND
TEL. +358 9 8554322
FAX. +358 9 8554323
VAT.NO: 2307199-5

Benchmark Configuration

Configuration of the benchmark is done by editing “application.cfg” text file. The file is located on mass storage and is read upon launch. The file contains nonperformance related configuration, such as size of window, booth mode and default OpenCL device selection. Anything which relates to performance is not configurable. The settings file itself contains description and small documentation about each option.

Profiling data provided by OpenCL can be enabled in settings and when enabled the data for each call is provided with the normal report. The profiling must be disabled on official benchmark runs.

The used computation device can be selected from application configuration file or upon launching the application. This makes it possible to benchmark the performance of all devices available in the platform. The used device information is shown in the generated report file.

Command line options

The executable can be also controlled with command line parameters. The command line parameters are explained in User Guide.

Run rules

To receive the official benchmark score the following rules must be followed:

Running the benchmark

- All tests except the online compiler test must be run completely, by either selecting run all tests from the menu or by setting the autorun setting on from the config file
- All settings available in “application.cfg” may be modified and their settings are visible in the xml result
    - Resolution should still be meaningful for target platform. E.g. fullscreen. Screen size does not affect the benchmarked computation load.
    - Modifying OpenGL context and surface settings through application.cfg can be done freely. Surface and context sensitive OpenGL execution is outside of benchmark timing so it does not affect results
- Command line parameters –on-measure-start and –on-measure-stop must not be used

Platform

- The device which is to be benchmarked must support OpenCL 1.1 Full Profile or higher
- OpenCL implementation must follow the OpenCL 1.1 specification
- OpenGL 2.1 (or newer) or OpenGL ES 2.0 (or newer) and the extension cl_khr_gl_sharing must be present

Runtime optimization

- Driver level optimizations specifically made to improve performance of solely this benchmark are not allowed. Attempts to recognize that Basemark CL is running and do runtime modifications based on that information is forbidden.

RIGHTWARE OY, NIITTYKATU 6 C
FIN-02200 ESPOO, FINLAND
TEL. +358 9 8554322
FAX. +358 9 8554323
VAT.NO: 2307199-5

  - For example, if OpenCL kernels supplied with the benchmark are cosmetically changed the results are expected to remain the same.
  - Detecting Basemark CL OpenCL kernels and replacing them with others is not allowed
  - Returning values with unusually poor accuracy from half_ and native_ prefixed functions just for this benchmark is forbidden.
- If you run benchmark multiple times in sequence on same device results are expected not to vary; intention of this is to prevent possibility of using first run of benchmark as 'offload to CL host' run and second one to get improved results

Contents

- Modifying OpenCL kernel code is not allowed. The modifications are checked by doing MD5 hash check on the kernel source files. Supplying preprocessor defines to kernel code to clBuildProgram through source code or with driver level modifications is not allowed.
- The asset files must be in their original format. The modifications are checked by doing MD5 hash check on the data files.

Compiling

- Modifying any benchmark source code is not allowed, except what is required to port the Benchmark into new platform, e.g. paths for asset files and GL/CL include header file locations
- Rightware allows using any common optimization flags when compiling the benchmark from source. Majority of the timed benchmark work is done in OpenCL, so compiler flags do not greatly affect the results of the benchmark
- The behavior of the benchmark relative to any external API (OpenCL, OpenGL) must not change. For example if we'd record all OpenCL calls from the benchmark they must remain identical order and have the same set of parameters regardless of any compile time optimizations used.
  - Variation which does not affect OpenCL workload in floating point parameters due to precision changes is allowed.

Releasing results

- Released or published results must follow the aforementioned run rules
- Unmodified report xml must be provided
- Sufficient information for reproducing the results must be provided
  - Used hardware/devices, drivers and their versions
- It is expected that the benchmark result can be reproduced by obtaining the product in question and simply installing the benchmark and running it. Scores made with unreleased implementation should be met or exceeded on the final published version of said implementation.
- Contact information to an individual who can provide the used executable and material for review if requested by Rightware
- Hardware and software used to produce publicly released results is expected to be brought widely available at latest within the next 3 months after the report publication date

RIGHTWARE OY, NIITTYKATU 6 C
FIN-02200 ESPOO, FINLAND
TEL. +358 9 8554322
FAX. +358 9 8554323
VAT.NO: 2307199-5

Exceptions

- If for some reason you can't meet all these requirements, please contact Rightware for approval of required changes prior to releasing your results

Kanzi Engine

Basemark CL™ is built on Rightware's Kanzi® engine, which enables the development of visually advanced applications with 2D and 3D content on any platform supporting Khronos Group's OpenGL ES 1.1, OpenGL ES 2.0 or OpenGL 2.1 standards. Kanzi is written completely in ANSI-C and is portable to any platform with a standards conforming C compiler and an implementation of at least one of the supported graphics API's.

Benchmark Tests

Basemark CL contains three kinds of tests: Image manipulation tests, physics simulation tests and the feature tests.

Image manipulation tests

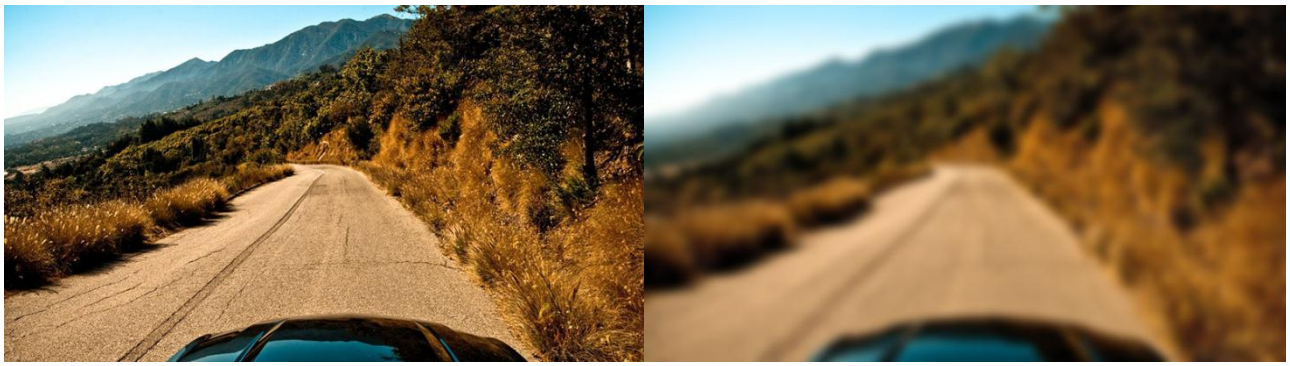
Benchmarking image manipulation performance is useful due to fact that image manipulation filters are widely used in image processing and computer machine vision applications. Various image manipulation filters are used in machine vision to find out discontinuities and remove unwanted features from images. Image processing in embedded devices has been increasingly important since the advent of cheap digital cameras. OpenCL devices may allow fast image enhancement on a modern smartphone without degrading usability nor requiring application specific circuits to perform this task.

Image manipulation tests run filters on image data and produce the filtered output on the screen. The implemented filters are listed below in separate sub paragraphs. The images in the paragraphs demonstrate how the corresponding filters affect the image. In addition performance of some of the image processing filters produces a good indication towards performance of other signal processing tasks, such as audio processing.

The image manipulation filters are applied to video streams in a separate test. This allows benchmarking the bandwidth limitations from moving data from Host to OpenCL device (e.g from CPU to GPU) and provides a good real world use case. Time spent decoding the video is not taken into account when determining benchmark score.

Smoothing

Image smoothing is implemented using Gaussian blur filter. Gaussian blur is a filter where blur weights are based on the distance from the center of the blur. As the Gaussian kernel is separable the blur is performed in horizontal and in vertical passes, thus having access patterns which resemble 1 dimensional convolution.



Smoothing filter

Color Correction

Image color correction is implemented using histogram correction. A histogram is representation of image brightness value distribution. The histogram is calculated for the image and is used to color correct the image.

Histogram test is not used in embedded profile.

Color correction

Noise Reduction

The noise reduction test is implemented with Median filter. Median filter is used to remove salt and pepper noise from source image data. Median filter produces the filtering based on the median of adjacent pixels.

Noise reduction

Surface Aware Smoothing

Surface aware image smoothing is implemented using bilateral blur. Bilateral blurring kernel is like weighted by distance from center of the blur like Gaussian blur and in addition it considers the radiometric differences between central value and neighboring pixels. Bilateral blur preserves sharp edges by systematically looping through each pixel and adjusting weights to the adjacent pixels accordingly. (Carlo Tomasi, Roberto Manduchi, 1998)

Surface aware smoothing

Sharpening

Laplacian of Gaussian filter is used to enhance edges of images. First the image is blurred with Gaussian in order to reduce high frequency noise, then Laplacian filter is applied to remove low frequency components and the result is summed into the original unfiltered image.

Sharpening

Physics tests

Physical simulations are good area for leveraging extra computing power which OpenCL brings. Physics simulations are already heavily used in PC-games, but not to a greater extent in mobile games or in applications doing heavy 3D physics simulation. With the help of good OpenCL performance it is possible for future mobile games and applications to include more physics based animations and game elements.

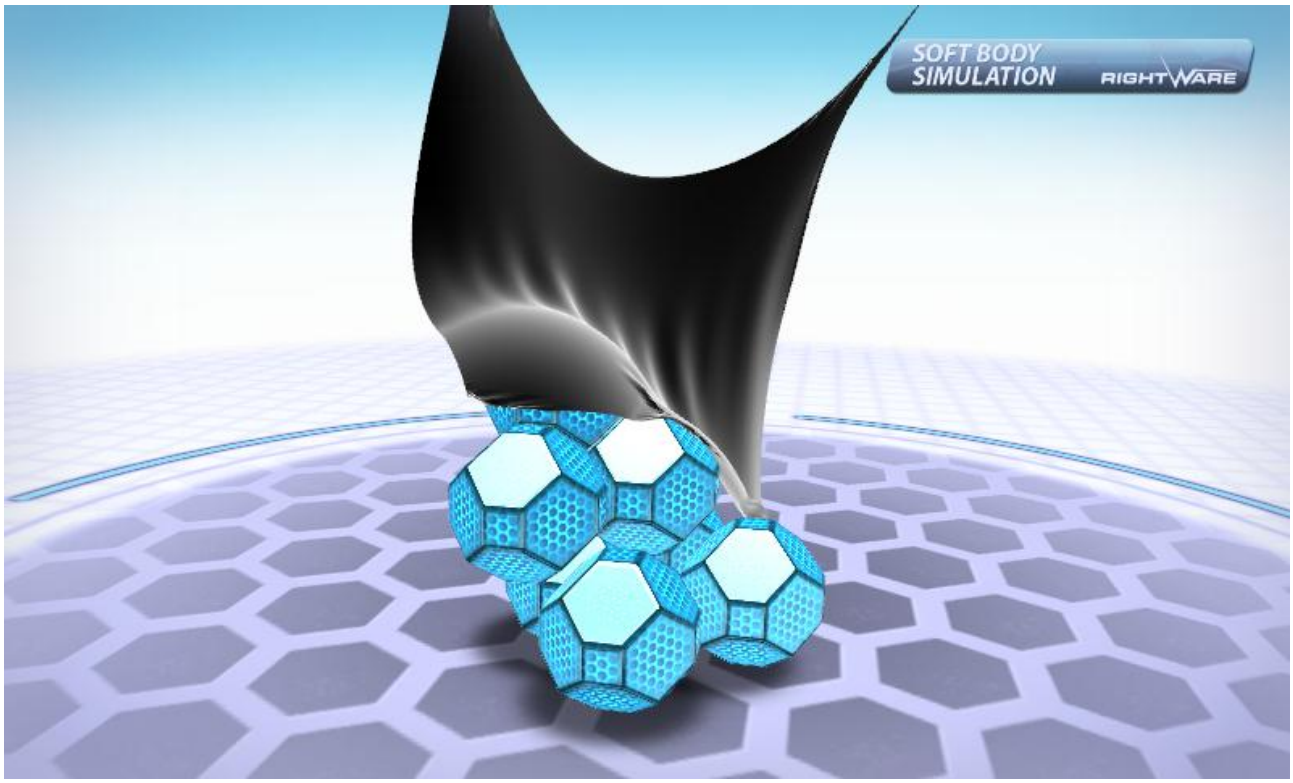
Basemark CL contains fluid and cloth simulations. Cloth simulations are widely used in games to add extra realism to cloths of characters, flags and other soft body items. Fluid simulations are used in games to simulate smoke, water and other materials. The realistic simulation adds possibility for player interaction with visuals, which is not possible with currently used artist made animations.

3D Soft Body Simulation

Soft bodies are deformable objects. The shape of the objects can change during the execution of the simulation.

Soft bodies are implemented by using mass points. Each mass point is connected with springs to four or more adjacent mass points. Forces (gravity etc.) are applied on mass points each frame and the spring constrains are solved during the frames. The complexity of the simulation can be adjusted by changing the amount of these mass points on the objects.

The simulation is rendered in 3D using OpenGL. The simulator will step on fixed time step to prevent unstabilities in the simulation on lower frame rates and to produce same payload every run time independant from frame rate.



Cloth Simulation

The implementation is based on Thomas Jakobsen's Advanced Character Physics paper (Jakobsen, 2003).

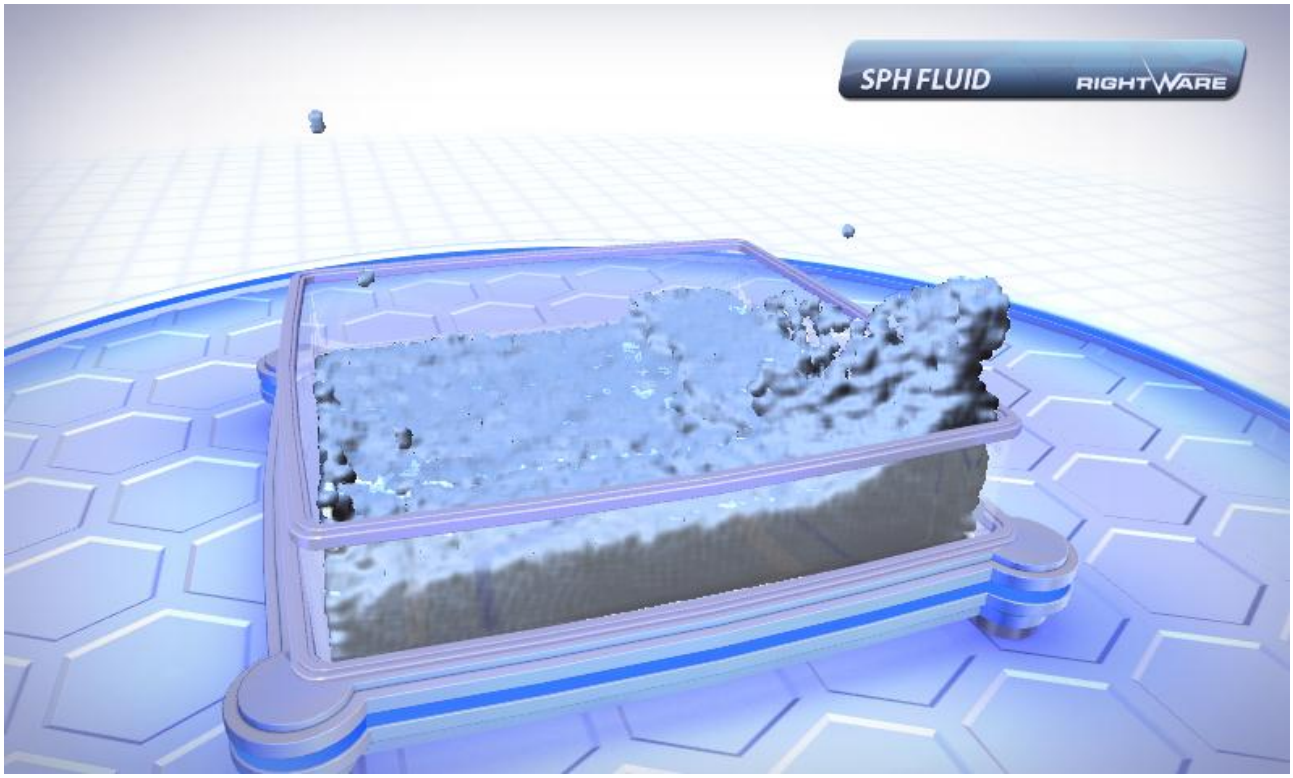
Smoothed Particle Hydrodynamics

Smoothed particle hydrodynamics (SPH) is a method used for simulating fluids and is increasingly used in real-time applications where interactivity is more important than accuracy. The fluid is divided into set of discrete particles with mass and size (smoothing distance). The physical properties of a single particle are calculated by summing the relevant properties of all particles within the smoothing distance.

RIGHTWARE OY, NIITTYKATU 6 C
FIN-02200 ESPOO, FINLAND
TEL. +358 9 8554322
FAX. +358 9 8554323
VAT.NO: 2307199-5

In order to optimize the search for neighbouring particles the simulation area is divided into discrete grid with side equal to the smoothing length. The particles are sorted based on their position in the grid using bitonic sort so the neighbour search has to only check particles residing in neighbouring voxels.

After calculating acceleration from the forces applied by the neighbours new position and velocity are calculated and the process is repeated.



Smoothed Particle Hydrodynamics Simulation

2D Fluid Simulation

A 2D grid based fluid simulation is implemented using Navier-Stokes equations. The density and velocity fields for the fluids are stored in buffers. Simulation on the data is updated each frame.

Initially the velocity and density buffers are set to contain the initial state of the fluid simulation. After the initialization each frame the update routine adds new forces, do the diffuse step and move density and velocity fields. The new forces are preset by the test to make the fluid movement look interesting. The diffuse step calculates the spread of the density and move step updates density fields taking into account the fluid velocity and updates the velocity buffer.

Density data is visualized using OpenGL. The fluid is rendered as a 2D OpenGL texture in orthogonal projection. The density information is shown by selecting color for each cell based on the density in the cells.

The implementation is based on Jos Stam's Real-Time Fluid Dynamics for Games presentation from Games Developer Conference 2003 (Jos Stam, 2003).

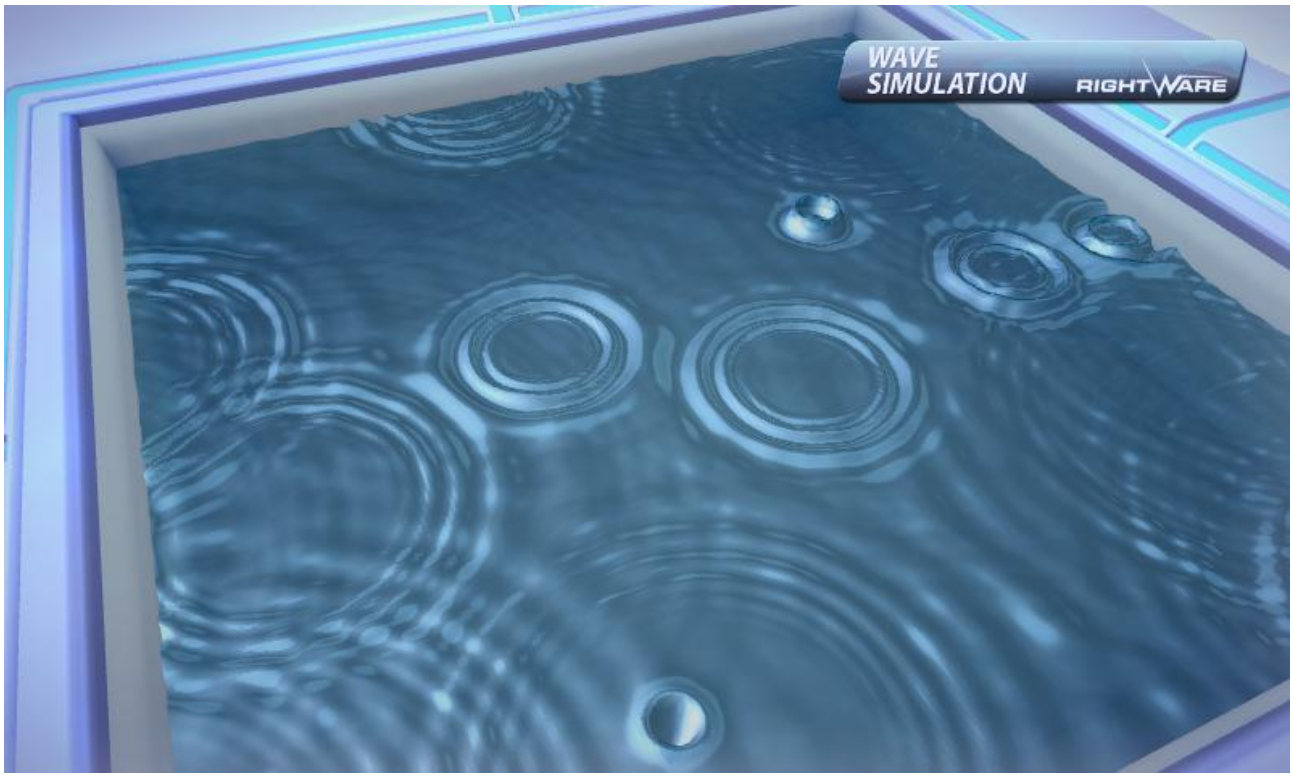
The test allows configuring the grid size for adjusting the computational complexity.

2D Fluid Simulation

Wave Simulation

Deep water wave propagation is simulated using Fast Fourier Transform based method where a propagation kernel is applied to a height field in frequency domain. Inverse transform is applied to get into spatial domain in order to get reflections from the boundaries and then forward transform is used to get back into frequency domain.

The score from this test gives indication to how well the device performs in other scenarios where Fourier transform is needed.



Fast Fourier Transform based wave simulation

General Test

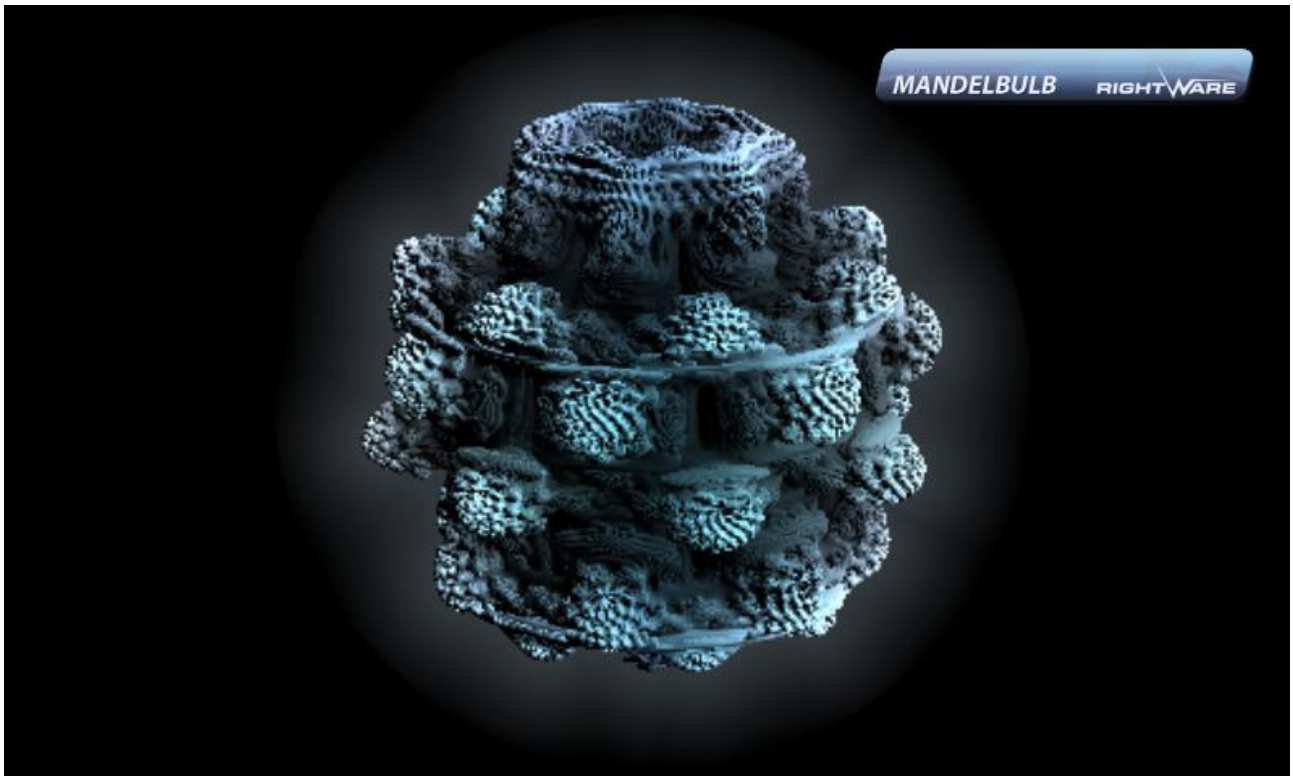
The general tests are more focused and aim to test only the performance of single or couple of features of the hardware at a time. The general tests are implemented to help hardware manufacturers to identify the bottlenecks in their configurations.

3D Mandelbulb Ray Tracing

The Mandelbulb fractal is a type of 3D fractal. The fractal was recently discovered by Daniel White and Paul Nylander.

3D fractal is rendered using ray tracing. The ray tracing for the 3D fractal uses the basic ray tracing formula. For each point in the canvas a ray is casted towards to the object to be rendered. Nearest distance of a point to the surface of the Mandelbulb is calculated. The power of the Mandelbulb is changed constantly so the polynomial solution for power 8 Mandelbulb is not used, therefore this test benchmarks the performance of built-in mathematical functions in the same vein as Julia fractal benchmarks the raw ALU performance. If the point is within predefined distance of the fractal surface the ray tracing is stopped and lighting calculations are done, if not the point is moved along the ray forward until the surface is found or predefined maximum iteration count is exceeded.

RIGHTWARE OY, NIITTYKATU 6 C
FIN-02200 ESPOO, FINLAND
TEL. +358 9 8554322
FAX. +358 9 8554323
VAT.NO: 2307199-5



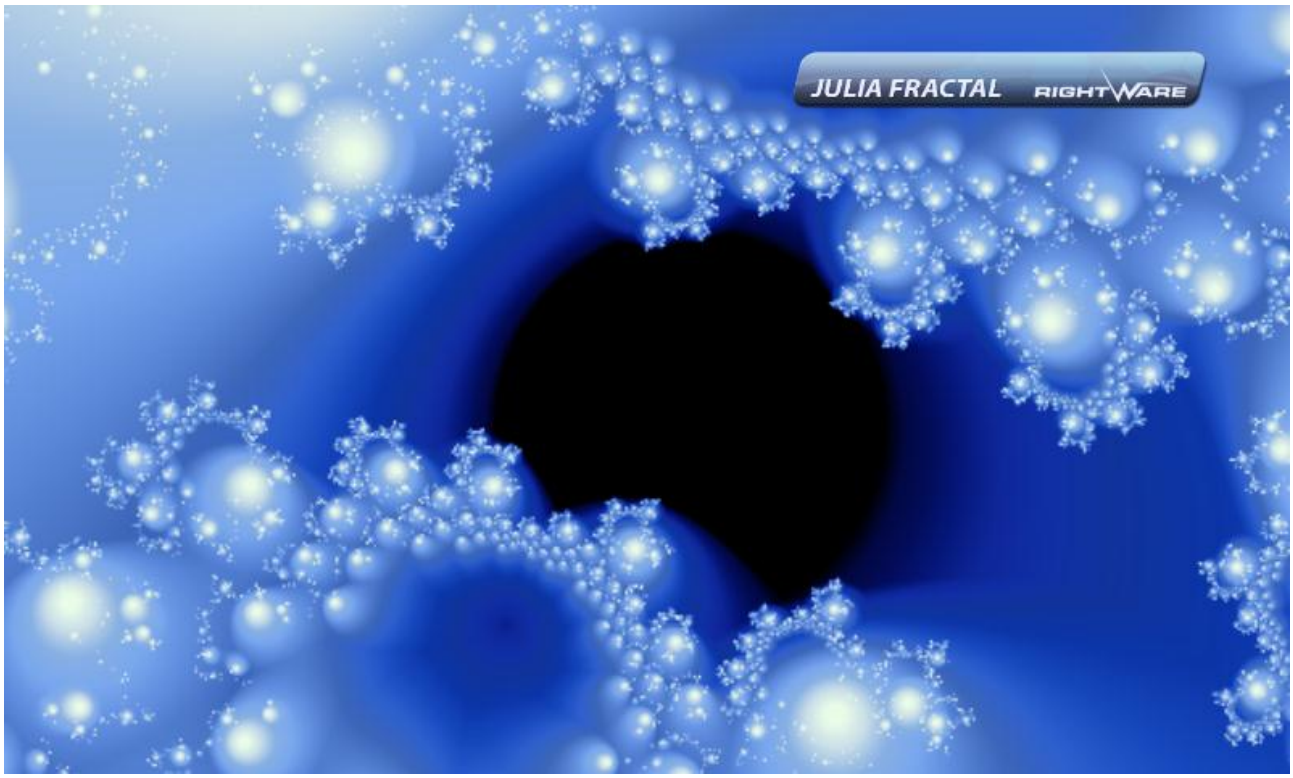
3D Mandelbulb

The lighting is calculated with regular Blinn-Phong shading model. The shading model requires surface normal to be available. The surface normal can be calculated from 3D gradient of the nearest distance estimator. Approximation of ambient occlusion is applied to enhance the surface details. The ambient occlusion is approximated by stepping away from hit point along the surface normal and finding nearest point on the fractal surface. This information can be used to produce a convincing looking approximation of ambient occlusion.

Julia Fractal

Julia fractal calculation is used to test general calculation power of the OpenCL implementation. Fractal calculation is a good application for testing raw arithmetic logic unit performance, because the computational load can be adjusted easily by changing iteration count per pixel and test's memory bandwidth requirements are almost nonexistent.

The fractal image is calculated on a texture object and rendered with OpenGL. Version of the fractal calculation which does not provide any visual feedback will also be provided for devices where there is no support for OpenGL.



Julia fractal

Online Compiler Test

The online compiler test measures how fast the OpenCL implementation can compile kernel code. The test compiles all kernels used in application and records the time spent on the operation. The test applies different OpenCL compiler settings and sets available preprocessor defines for some runs of test.

The score from this test do not affect the Basemark CL final score.

Bibliography

J. E. Smith. 1988. Characterizing computer performance with a single number. *Commun. ACM* 31, 10 (October 1988), 1202-1206. DOI=10.1145/63039.63043 http://doi.acm.org/10.1145/63039.63043

B. Jacob and T. Mudge, Notes on Calculating Computer Performance, Technical report CSE-TR-231-95, Department of Electrical Engineering and Computer Science, University of Michigan, USA, March 1995.

Carlo Tomasi, Roberto Manduchi. (1998). Bilateral Filtering for Gray and Color Images

Daniel White. The unravelling of the Real 3D Manbdelbulb, (http://www.skytopia.com/project/fractal/mandelbulb.html)

Jos Stam. (2003). Real-Time Fluid Dynamics for Games.

Khronos Group. (2008). *OpenGL ES Common/Common-Lite Profile Specification Version 1.1.12 (Full Specification).*

Khronos Group. (2009). *OpenGL ES Common Profile Specification Version 2.0.24 (Full Specification).*

Sylvain Paris, Pierre Kornprobst, Jack Tumblin, Frédo Durand, A Gentle Introduction to Bilateral Filtering and its Applications, (http://people.csail.mit.edu/sparis/bf_course/)

Thomas Jakobsen. (2003). Advanced character physics

Questions & Contact Information

If you have any questions about the BDP product cycle, or need further information about anything of the above, please don't hesitate to contact us at:

Mr. Teemu Uotila
Director, Benchmarks
Email: teemu.uotila@rightware.com
Tel: +358-9-855-4322

Rightware
http://www.rightware.com

Niittykatu 6 C
FIN-02200 Espoo
FINLAND
Fax: +358-9-855-4323