An Industry-Standard Benchmark Consortium

OABench™ Version 2.0

Benchmark Name: Bezier

Highlights

- **Benchmarks the classic Bezier curve algorithm**
- **Interpolate a set of points defined by the four points of a Bezier curve (two end points, two intermediate points)**
- **Fixed point and floating point versions available**
- **A component of the EEMBC OAV2mark™**
- **Four new data files implemented in Version 2**
- **Bezier curves are the backbone of computer graphics, font renderings and design, and computer graphics.**
- **Implements Cyclical Redundancy Checksum (CRC) for self-checking in integer mode, and SNR for self checking in floating point mode.**

History, Application, and Restrictions

Pierre Étienne Bézier, a French engineer, created a mathematical numerical analysis technique for drawing parametric curves. The problem solved was how to draw curves based on fixed data points. The creator of the first algorithm to implement Bezier curves was Paul de Casteljau.

Bezier curves can be linear, quadratic, cubic, or even triangles. In computer science, one of the primary applications of Bezier curves is the creation and smoothing of fonts on-screen and in a printer for the printed page. For example, TrueType® fonts use Bezier curves. TrueType, PostScript®, Ghostscript, The GIMP, and many other applications use Bezier splines with cubic Bezier curves for drawing shapes. Translation, scaling, and rotation on the curve can be accomplished by applying the respective transform on the control points of the curve (the points).

As with all EEMBC source code, the Bezier benchmark is not to be used in any commercial product whatsoever.

Benchmark Description

In EEMBC's OABench office automation benchmark suite, the calculations interpolate a set of points defined by the four points of a Bezier curve. Two endpoints and two control points define the curve. The points are in 2D space, and are defined using floating point (double precision) or integer variables. The algorithm makes use of configuration constants in the header file `bez.h`. This includes the number of points to interpolate for each curve as well as the overall loop count. The main function makes use of a call to the test harness `malloc()` to create an array of curve structures for all the input data before processing starts. The first line in the input file defines the number of points following in the rest of the file.

This benchmark evaluates the parametric function for Bezier curve

Benchmark Description (continued)

```
P(t) = p0*(1-t)^3 + 3*p1*t*(1-t)^2 + 3*p2*(t^2)*(1-t) + p3*t^3
```

1000 iterations is the default, 10 for CRC verification runs. There are four data sets, one of which is reserved for profiling initialization.

Analysis of Computing Resources

The benchmark uses division, multiplication, and scalar processing. There are two loops (inner and outer), so efficient compilers and architectures can take advantage of this, but the function `interpolatePoints()` cannot be optimized away. This benchmark is almost exclusively CPU bound, and the quality of the math library has an effect on performance.

Optimizations Allowed

Out-of-the-Box / Standard C
Full Fury / Optimized

- The C code must not be changed for Out-of-the-Box unless it must be modified to get it to compile. All changes must be documented, authorized by the certification authority, and must not have a performance impact.
- For Out-of-the-Box, additional hardware can be used if it does not require code changes.
- All Optimized libraries must be part of the standard compiler package, and/or available to all customers.
- Test harness changes may be made for portability reasons if they do not impact performance.
- For Optimized, the basic algorithm may not be changed, but the code may be rewritten in assembler. Re-writing the code to take advantage of parallelism is allowed so long as the correct answers are achieved using any arbitrary keys (not just those supplied in the benchmark code). You may not optimize out the function `interpolatePoints()`.
- For Optimized, optimized libraries can be used if they are publicly available.
- For floating point, SNR is used to evaluate the quality of the output. Double precision is required to achieve certifiable SNR.
- For Optimized, in lining is allowed.
- Additional data files may be used during certification to ensure the correctness of the optimized benchmark. You should ***not*** assume data patterns during optimization.
- Profile directed optimization is allowed using training data set 1, bezdata1.txt.