UNIVERSITEIT
VAN
AMSTERDAM

Programming with Gaigen 2

Programming with Gaigen 2

Daniel Fontijne
University of Amsterdam
fontijne@science.uva.nl

Overview Part I

What if I just want to program using geometric algebra?

Overview Part I

What if I just want to program using geometric algebra?
(and want my program to be reasonably efficient?)

Overview Part I

What if I just want to program using geometric algebra?
(and want my program to be reasonably efficient?)

Part I: `GA Sandbox`

`GA Sandbox` is:

Overview Part I

What if I just want to program using geometric algebra?
(and want my program to be reasonably efficient?)

Part I: `GA Sandbox`

`GA Sandbox` is:

- C++ GA implementations generated by `Gaigen 2`.

Overview Part I

What if I just want to program using geometric algebra? (and want my program to be reasonably efficient?)

Part I: `GA Sandbox`

`GA Sandbox` is:

- C++ GA implementations generated by `Gaigen 2`.
- Extra utility code.

Overview Part I

What if I just want to program using geometric algebra?
(and want my program to be reasonably efficient?)

Part I: `GA Sandbox`

`GA Sandbox` is:

- C++ GA implementations generated by `Gaigen 2`.
- Extra utility code.
- A lot of example code.

Overview Part I

What if I just want to program using geometric algebra?
(and want my program to be reasonably efficient?)

Part I: `GA Sandbox`

`GA Sandbox` is:

- C++ GA implementations generated by `Gaigen 2`.
- Extra utility code.
- A lot of example code.
- Some exercises with solutions.

Overview Part I

What if I just want to program using geometric algebra?
(and want my program to be reasonably efficient?)

Part I: `GA Sandbox`

`GA Sandbox` is:

- C++ GA implementations generated by `Gaigen 2`.
- Extra utility code.
- A lot of example code.
- Some exercises with solutions.
- (comes with a book, too).

Overview Part II

Part II: `Gaigen 2`: a Geometric Algebra Implementation Generator.

Overview Part II

Part II: `Gaigen 2`: a Geometric Algebra Implementation Generator.

- Ideas behind `Gaigen 2`.

Overview Part II

Part II: `Gaigen 2`: a Geometric Algebra Implementation Generator.

- Ideas behind `Gaigen 2`.
- Installing `Gaigen 2`.

Overview Part II

Part II: `Gaigen 2`: a Geometric Algebra Implementation Generator.

- Ideas behind `Gaigen 2`.
- Installing `Gaigen 2`.
- Writing a specification of an algebra.

Overview Part II

Part II: `Gaigen 2`: a Geometric Algebra Implementation Generator.

- Ideas behind `Gaigen 2`.
- Installing `Gaigen 2`.
- Writing a specification of an algebra.
- Generating the code.

Overview Part II

Part II: `Gaigen 2`: a Geometric Algebra Implementation Generator.

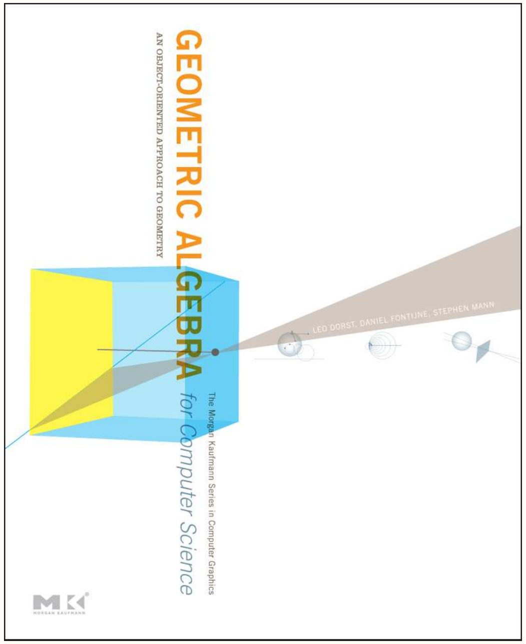
- Ideas behind `Gaigen 2`.
- Installing `Gaigen 2`.
- Writing a specification of an algebra.
- Generating the code.
- Profiling.

Part I: GA Sandbox

Part I: GA Sandbox

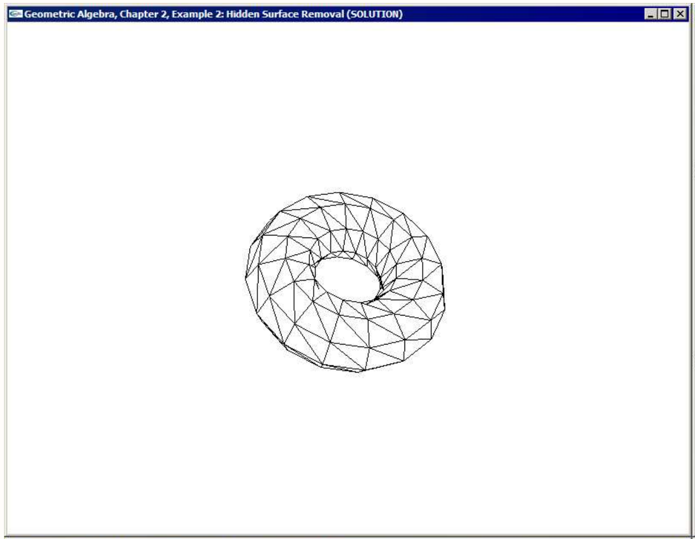
GA Sandbox Overview

`GA Sandbox` was created as a software companion to *Geometric Algebra for Computer Science*.

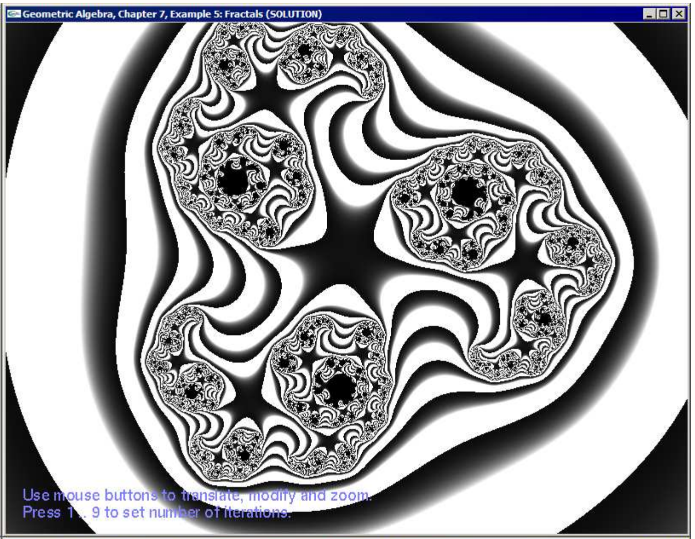

GEOMETRIC ALGEBRA for Computer Science
AN OBJECT-ORIENTED APPROACH TO GEOMETRY
LEO DORST, DANIEL FONTIJNE, STEPHEN MANN
The Morgan Kaufmann Series in Computer Graphics
MK
MORGAN KAUFMANN

You may download `GA Sandbox` at:
http://www.geometricalgebra.net/sandbox.html
or on the handout CD in directory `/gasandbox`

GA Sandbox Examples

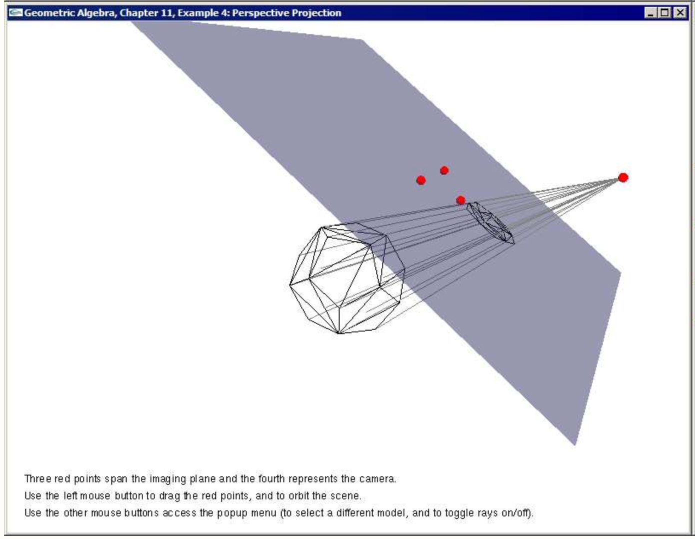


Geometric Algebra, Chapter 2, Example 2: Hidden Surface Removal (SOLUTION)



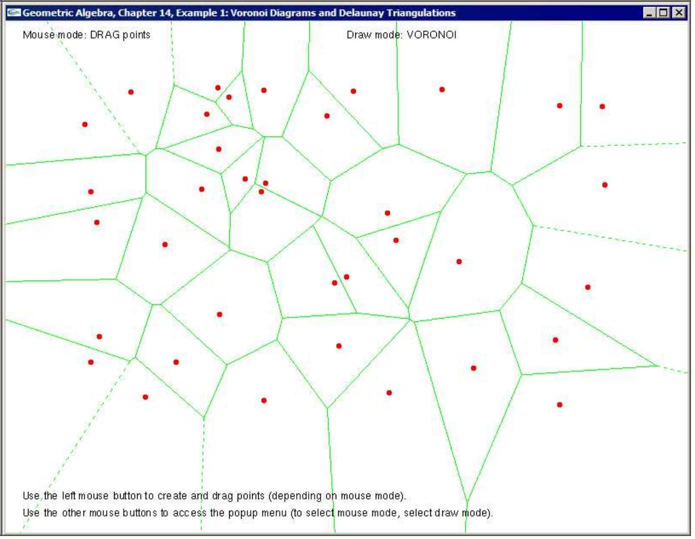
Geometric Algebra, Chapter 7, Example 5: Fractals (SOLUTION)

Use mouse buttons to translate, modify and zoom.
Press 1 .. 9 to set number of iterations.



Geometric Algebra, Chapter 11, Example 4: Perspective Projection

Three red points span the imaging plane and the fourth represents the camera.
Use the left mouse button to drag the red points, and to orbit the scene.
Use the other mouse buttons access the popup menu (to select a different model, and to toggle rays on/off).



Geometric Algebra, Chapter 14, Example 1: Voronoi Diagrams and Delaunay Triangulations

Mouse mode: DRAG points

Draw mode: VORONOI

Use the left mouse button to create and drag points (depending on mouse mode).
Use the other mouse buttons to access the popup menu (to select mouse mode, select draw mode).

Building the Source Code

Linux:

- Install libraries (`GLUT`, `ANTLR`, `OpenCV`, `FLTK`).
- Extract `GA Sandbox` package.
- `./configure`
- `make`

Building the Source Code

Linux:

- Install libraries (`GLUT`, `ANTLR`, `OpenCV`, `FLTK`).
- Extract `GA Sandbox` package.
- `./configure`
- `make`

Windows / Visual Studio:

- Install libraries (or extract precompiled libraries from zip).
- Extract `GA Sandbox` package.
- Open project and build it.

GA Sandbox Contents

Contents of `GA Sandbox`:

UNIVERSITEIT
VAN
AMSTERDAM

GA Sandbox Contents

Contents of `GA Sandbox`:

- `libgasandbox`: algebras + drawing + utility + analyzing multivectors.

GA Sandbox Contents

Contents of `GA Sandbox`:

- `libgasandbox`: algebras + drawing + utility + analyzing multivectors.
- `libgasandboxparse`: parsing multivector strings (requires `ANTLR` library).

UNIVERSITEIT
VAN
AMSTERDAM

GA Sandbox Contents

Contents of `GA Sandbox`:

- `libgasandbox`: algebras + drawing + utility + analyzing multivectors.
- `libgasandboxparse`: parsing multivector strings (requires `ANTLR` library).
- `chapX/exY`: nearly programming 50 examples.

UNIVERSITEIT
VAN
AMSTERDAM

GA Sandbox Contents

Contents of `GA Sandbox`:

- `libgasandbox`: algebras + drawing + utility + analyzing multivectors.
- `libgasandboxparse`: parsing multivector strings (requires `ANTLR` library).
- `chapX/exY`: nearly programming 50 examples.
- (library `QHull` for computing convex hulls).

libgasandbox 1/2

`libgasandbox` contains 5 algebra implementations: `e2ga`, `e3ga`, `h3ga`, `c2ga`, `c3ga`.

libgasandbox 1/2

`libgasandbox` contains 5 algebra implementations: `e2ga`, `e3ga`, `h3ga`, `c2ga`, `c3ga`.

Algebra implementations are in `xxga.cpp` and `xxga.h`: generated by `Gaigen 2` from `xxga.gs2` files.

The `xxga.cpp` and `xxga.h` files provide:

libgasandbox 1/2

`libgasandbox` contains 5 algebra implementations: `e2ga`, `e3ga`, `h3ga`, `c2ga`, `c3ga`.

Algebra implementations are in `xxga.cpp` and `xxga.h`: generated by `Gaigen 2` from `xxga.gs2` files.

The `xxga.cpp` and `xxga.h` files provide:

- General multivector class `mv`.

libgasandbox 1/2

`libgasandbox` contains 5 algebra implementations: `e2ga`, `e3ga`, `h3ga`, `c2ga`, `c3ga`.

Algebra implementations are in `xxga.cpp` and `xxga.h`: generated by `Gaigen 2` from `xxga.gs2` files.

The `xxga.cpp` and `xxga.h` files provide:

- General multivector class `mv`.
- Specialized classes like `vector`, `bivector`, `rotor`.

libgasandbox 1/2

`libgasandbox` contains 5 algebra implementations: `e2ga`, `e3ga`, `h3ga`, `c2ga`, `c3ga`.

Algebra implementations are in `xxga.cpp` and `xxga.h`: generated by `Gaigen 2` from `xxga.gs2` files.

The `xxga.cpp` and `xxga.h` files provide:

- General multivector class `mv`.
- Specialized classes like `vector`, `bivector`, `rotor`.
- Basis vectors, like `e1`, `e2`, `e3`.

libgasandbox 1/2

`libgasandbox` contains 5 algebra implementations: `e2ga`, `e3ga`, `h3ga`, `c2ga`, `c3ga`.

Algebra implementations are in `xxga.cpp` and `xxga.h`: generated by `Gaigen 2` from `xxga.gs2` files.

The `xxga.cpp` and `xxga.h` files provide:

- General multivector class `mv`.
- Specialized classes like `vector`, `bivector`, `rotor`.
- Basis vectors, like `e1`, `e2`, `e3`.
- Other constants, like `I`.

libgasandbox 1/2

`libgasandbox` contains 5 algebra implementations: `e2ga`, `e3ga`, `h3ga`, `c2ga`, `c3ga`.

Algebra implementations are in `xxga.cpp` and `xxga.h`: generated by `Gaigen 2` from `xxga.gs2` files.

The `xxga.cpp` and `xxga.h` files provide:

- General multivector class `mv`.
- Specialized classes like `vector`, `bivector`, `rotor`.
- Basis vectors, like `e1`, `e2`, `e3`.
- Other constants, like `I`.
- Basic functions: products, inversion, dualization, . . .

libgasandbox 1/2

libgasandbox contains 5 algebra implementations: e2ga, e3ga, h3ga, c2ga, c3ga.

Algebra implementations are in xxga.cpp and xxga.h: generated by Gaigen 2 from xxga.gs2 files.

The xxga.cpp and xxga.h files provide:

- General multivector class mv.
- Specialized classes like vector, bivector, rotor.
- Basis vectors, like e1, e2, e3.
- Other constants, like I.
- Basic functions: products, inversion, dualization, . . .
- Operators:

  outer product: $\wedge$

  geometric product: *

  left contraction (an inner product): $<<$

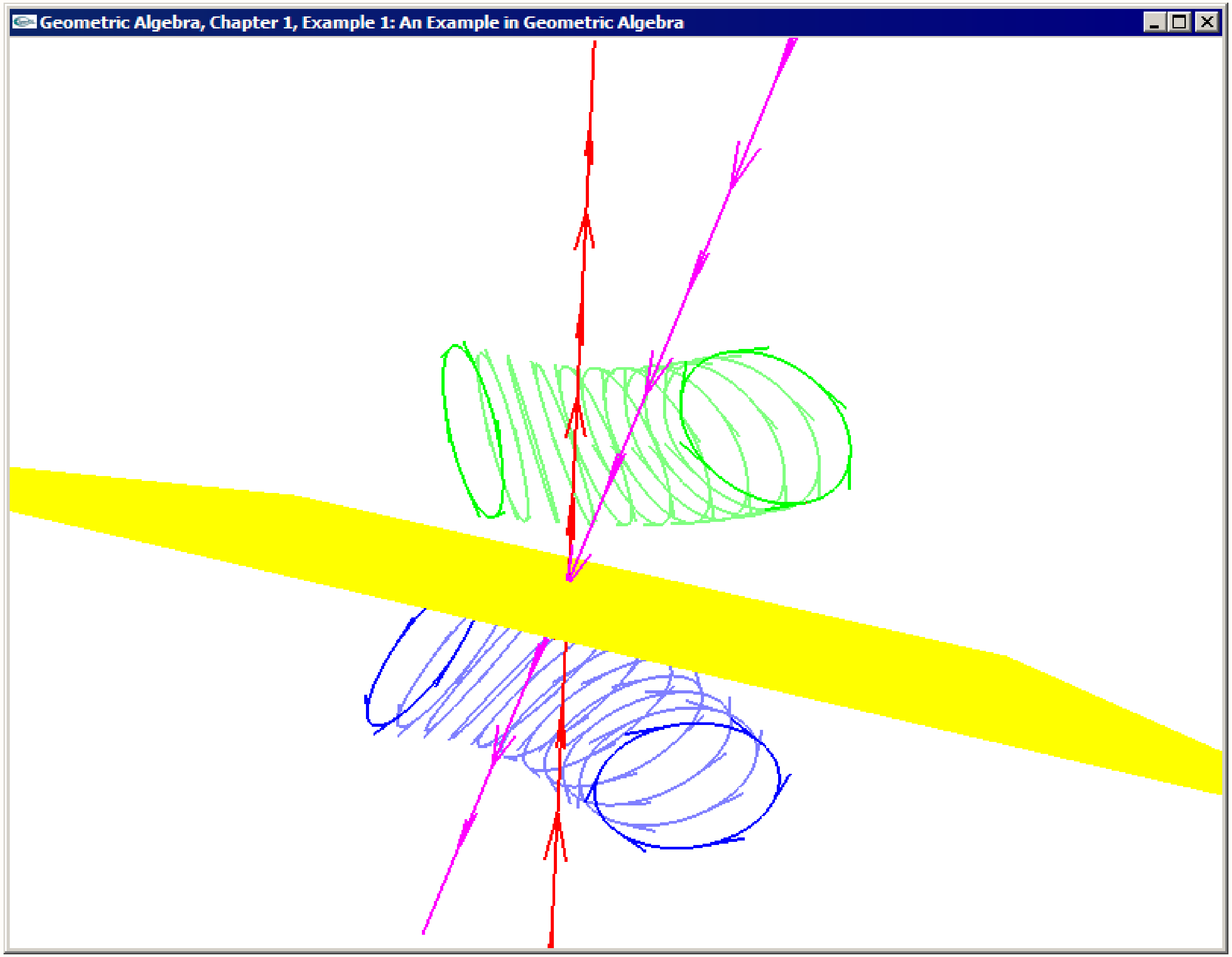
libgasandbox 2/2

Other files in `libgasandbox`:

`xxga_util.cpp`: utility functions.
`xxga_draw.cpp`: `OpenGL` drawing functions.
`mv_analyze_xxga.cpp`: multivector ‘analysis’ functions.

UNIVERSITEIT
VAN
AMSTERDAM

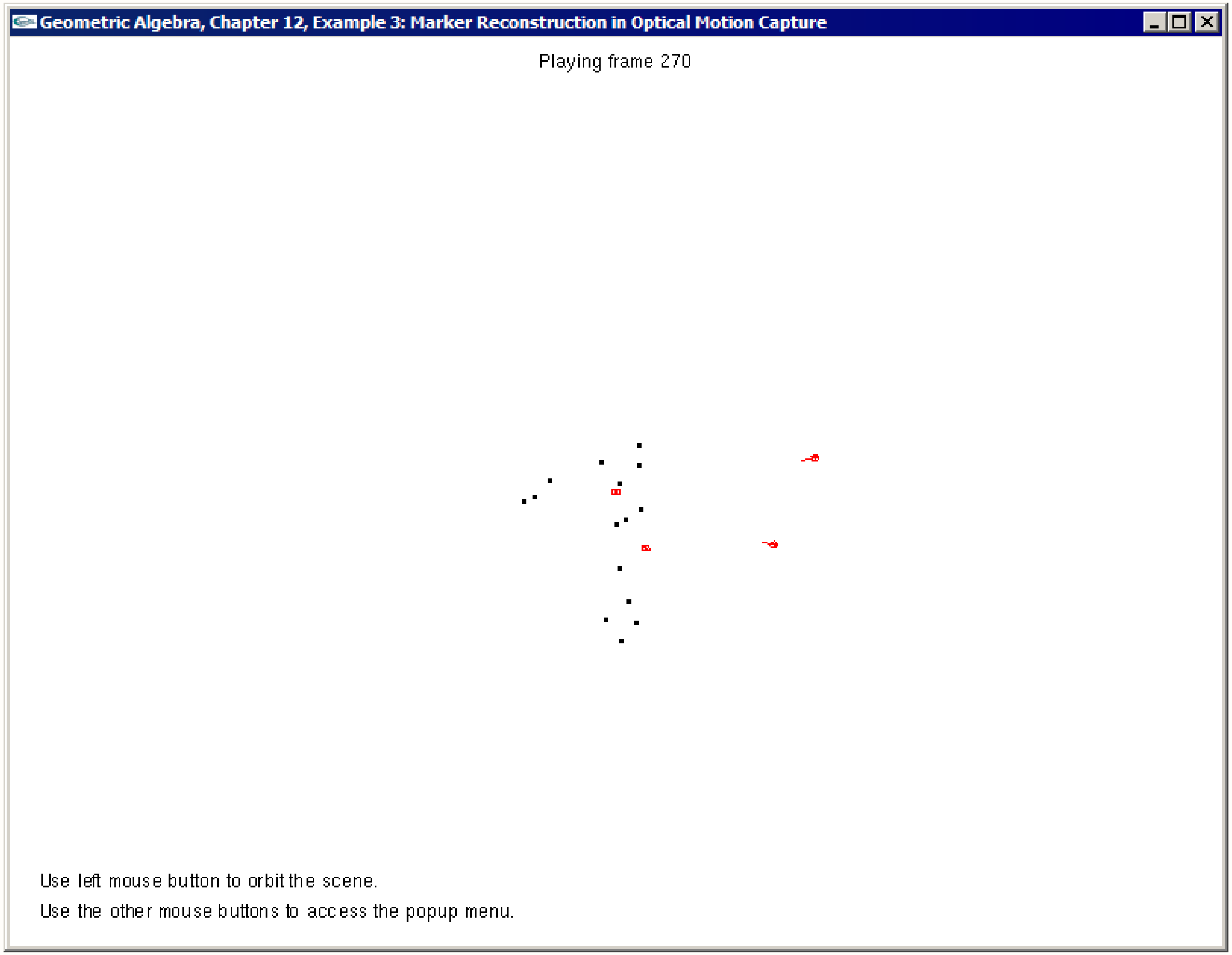
Chapter 1 Example Code



Geometric Algebra, Chapter 1, Example 1: An Example in Geometric Algebra

UNIVERSITEIT
VAN
AMSTERDAM

Optical Motion Capture Example Code



Geometric Algebra, Chapter 12, Example 3: Marker Reconstruction in Optical Motion Capture

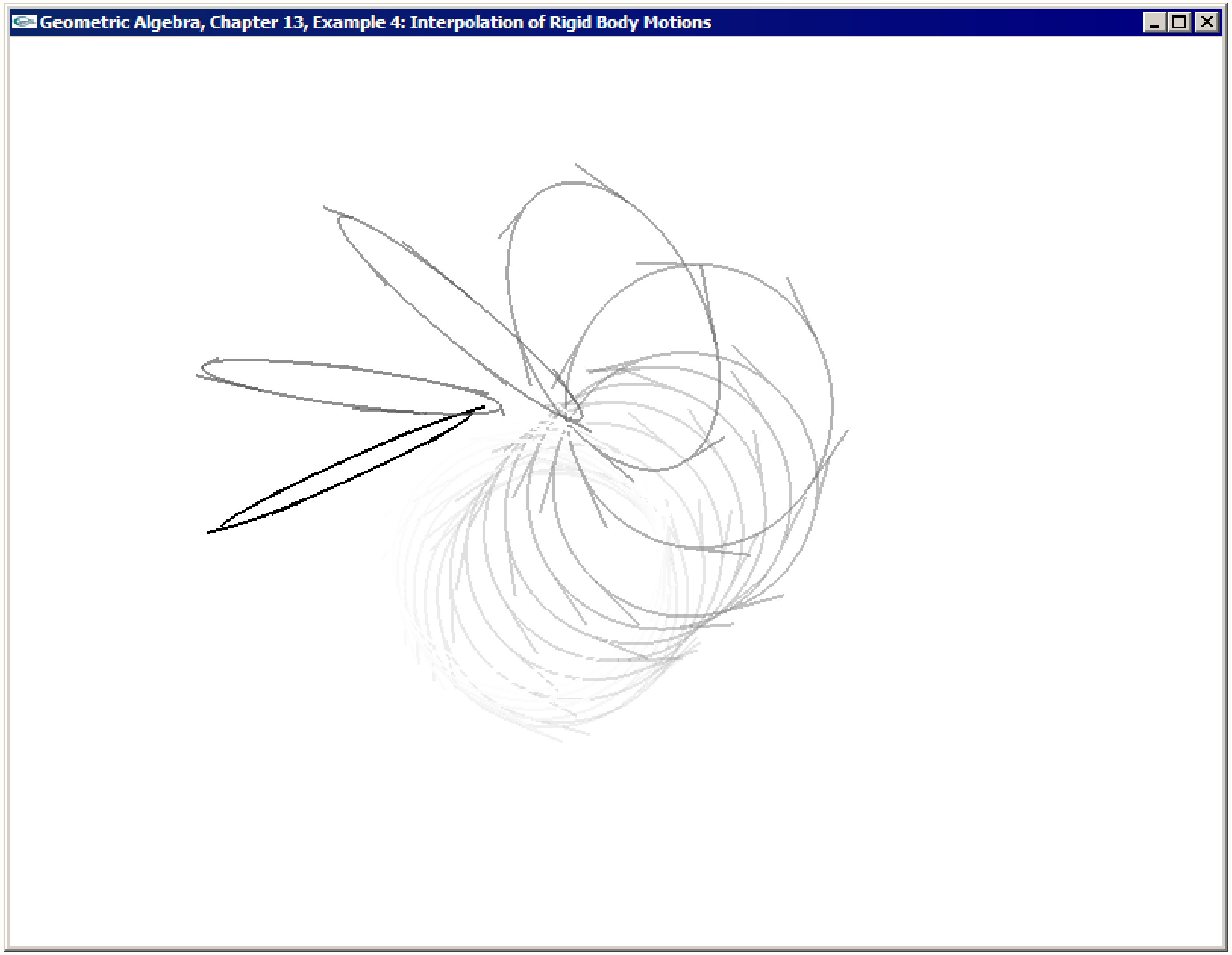
Playing frame 270

Use left mouse button to orbit the scene.

Use the other mouse buttons to access the popup menu.

UNIVERSITEIT
VAN
AMSTERDAM

Translation / Rotation Interpolation Code



Geometric Algebra, Chapter 13, Example 4: Interpolation of Rigid Body Motions

Part II: Gaigen 2

Part II: `Gaigen 2`

Gaigen 2 Overview 1/2

`Gaigen 2` is a code generator.

Gaigen 2 Overview 1/2

`Gaigen 2` is a code generator.

Input: specification of a geometric algebra + optional profile.

Gaigen 2 Overview 1/2

`Gaigen 2` is a code generator.

Input: specification of a geometric algebra + optional profile.

Output: efficient C++ or Java source code.

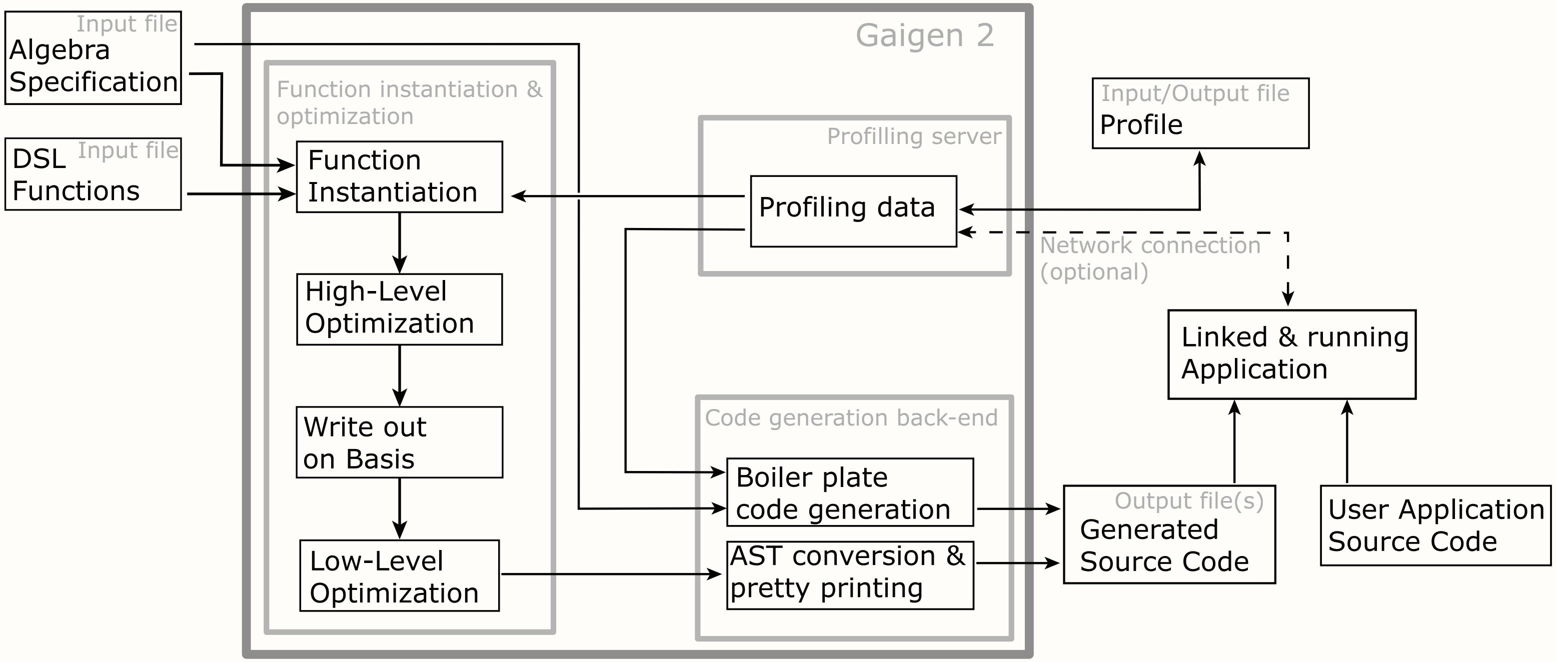
UNIVERSITEIT
VAN
AMSTERDAM

Gaigen 2 Overview 1/2

`Gaigen 2` is a code generator.

Input: specification of a geometric algebra + optional profile.

Output: efficient C++ or Java source code.



Gaigen 2 Overview 2/2

A word of caution: `Gaigen 2` is not a well-polished tool.
Easiest way to start ‘using’ it is `GA Sandbox`.

Gaigen 2 Overview 2/2

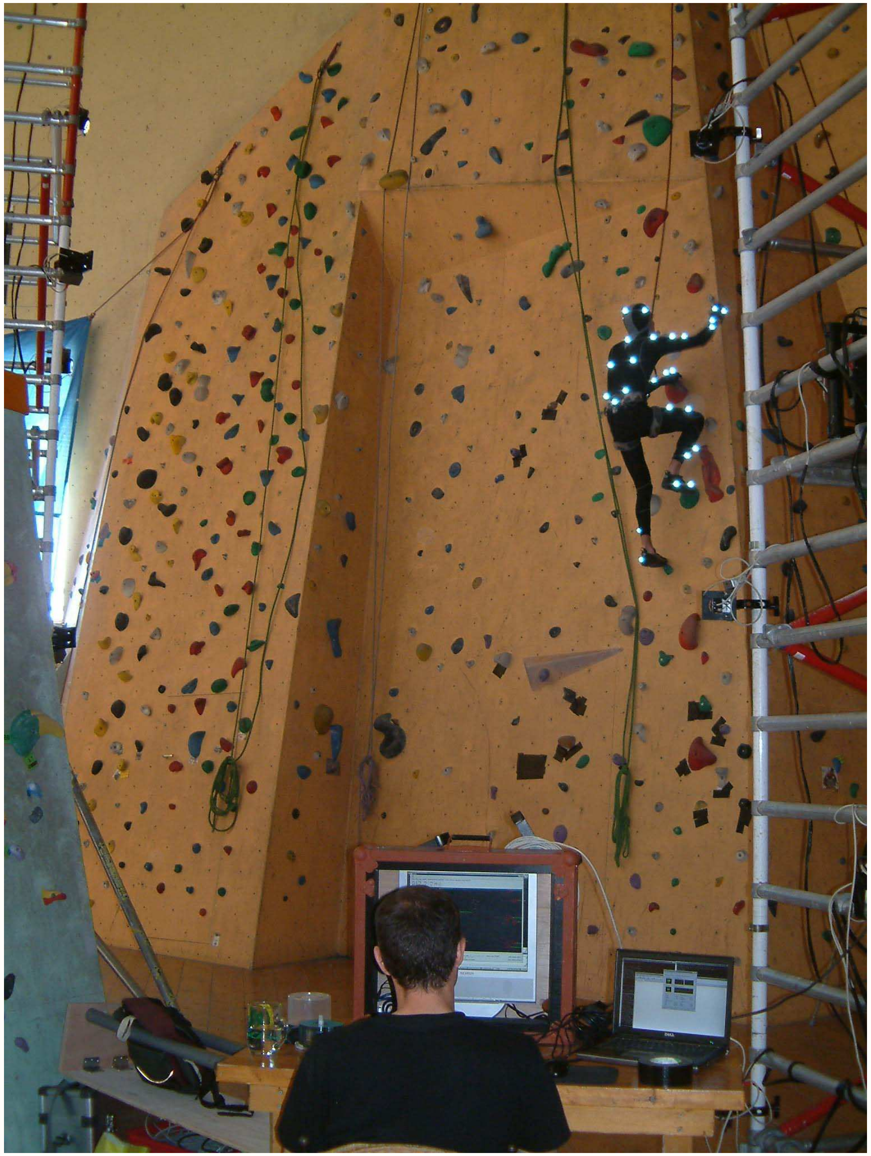
A word of caution: `Gaigen 2` is not a well-polished tool. Easiest way to start ‘using’ it is `GA Sandbox`.

But: personally I used it successfully for several larger projects:

- Set of simple ray tracers.
- `GA Sandbox`.
- Optical motion capture system.

UNIVERSITEIT
VAN
AMSTERDAM

Gaigen 2 used for Motion Capture



UNIVERSITEIT
VAN
AMSTERDAM

Ray tracer benchmarks

model	implementation	rendering time
3D LA	standard	1.00×
4D LA	standard	1.22×
3D GA	Gaigen 2	0.98×
4D GA	Gaigen 2	1.2×
5D GA	Gaigen 2	1.26×
3D GA	Gaigen 1	2.56×
4D GA	Gaigen 1	2.97×
5D GA	Gaigen 1	5.71×
3D GA	CLU	78×
5D GA	CLU	178×

Multivectors Representation in Gaigen 2

Like many other GA implementations, `Gaigen 2` represents multivectors as a sum of basis blades.

Multivectors Representation in Gaigen 2

Like many other GA implementations, `Gaigen 2` represents multivectors as a sum of basis blades.

Example of basis for 3-D space:

$$\{\ \underbrace{1}_{\textit{grade } 0}\ ,\ \underbrace{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3}_{\textit{grade } 1}\ ,\ \underbrace{\mathbf{e}_1 \wedge \mathbf{e}_2,\ \mathbf{e}_2 \wedge \mathbf{e}_3,\ \mathbf{e}_1 \wedge \mathbf{e}_3}_{\textit{grade } 2}\ ,\ \underbrace{\mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3}_{\textit{grade } 3}\}.$$

Multivectors Representation in Gaigen 2

Like many other GA implementations, `Gaigen 2` represents multivectors as a sum of basis blades.

Example of basis for 3-D space:

$$\{ \underbrace{1}_{\textit{grade 0}} , \underbrace{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3}_{\textit{grade 1}} , \underbrace{\mathbf{e}_1 \wedge \mathbf{e}_2,\ \mathbf{e}_2 \wedge \mathbf{e}_3,\ \mathbf{e}_1 \wedge \mathbf{e}_3}_{\textit{grade 2}} , \underbrace{\mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3}_{\textit{grade 3}} \}.$$

For example, a 3D-rotor:

$$\mathbf{R} = -0.30 - 0.04\,\mathbf{e}_2 \wedge \mathbf{e}_3 + 0.86\,\mathbf{e}_3 \wedge \mathbf{e}_1 - 0.68\,\mathbf{e}_1 \wedge \mathbf{e}_2$$

UNIVERSITEIT VAN AMSTERDAM

Multivectors Representation in Gaigen 2

Like many other GA implementations, `Gaigen 2` represents multivectors as a sum of basis blades.

Example of basis for 3-D space:

$$\{\underbrace{1}_{\textit{grade 0}}, \underbrace{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3}_{\textit{grade 1}}, \underbrace{\mathbf{e}_1 \wedge \mathbf{e}_2,\ \mathbf{e}_2 \wedge \mathbf{e}_3,\ \mathbf{e}_1 \wedge \mathbf{e}_3}_{\textit{grade 2}}, \underbrace{\mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3}_{\textit{grade 3}}\}.$$

For example, a 3D-rotor:

$$\mathbf{R} = -0.30 - 0.04\,\mathbf{e}_2 \wedge \mathbf{e}_3 + 0.86\,\mathbf{e}_3 \wedge \mathbf{e}_1 - 0.68\,\mathbf{e}_1 \wedge \mathbf{e}_2$$

In general:
n-dimensional geometric algebra $\longrightarrow$ 2^n multivector coordinates.

But in real-world usage, many of those coordinates are zero!

Ideas behind Gaigen 2

Observation in 2001: GA implementations are slow.

Ideas behind Gaigen 2

Observation in 2001: GA implementations are slow.

Reasons:

- The multivector is too general.
    - Wastes memory / processing time.
    - Solution: specialized multivector types.
    - A compromise between mathematical elegance and computational efficiency.

UNIVERSITEIT
VAN
AMSTERDAM

Ideas behind Gaigen 2

Observation in 2001: GA implementations are slow.

Reasons:

- The multivector is too general.
  - Wastes memory / processing time.
  - Solution: specialized multivector types.
  - A compromise between mathematical elegance and computational efficiency.
- Functions over multivectors are slow.
  - Custom functions for each type of argument.

UNIVERSITEIT VAN AMSTERDAM

Specialized multivectors

`Gaigen 2` generates classes for specialized multivectors.

Specialized multivectors

`Gaigen 2` generates classes for specialized multivectors.

Examples of specialized multivector specifications:

```
specialization: vectorE3GA(e1, e2, e3);
```

UNIVERSITEIT VAN AMSTERDAM

Specialized multivectors

`Gaigen 2` generates classes for specialized multivectors.

Examples of specialized multivector specifications:

```
specialization: vectorE3GA(e1, e2, e3);

specialization: line(e1^e2^einf, e1^e3^einf, e2^e3^einf,
                       e1^e0^einf, e2^e0^einf, e3^e0^einf);
```

Specialized multivectors

`Gaigen 2` generates classes for specialized multivectors.

Examples of specialized multivector specifications:

```
specialization: vectorE3GA(e1, e2, e3);

specialization: line(e1^e2^einf, e1^e3^einf, e2^e3^einf,
                     e1^e0^einf, e2^e0^einf, e3^e0^einf);

specialization: normalizedPoint(e0 = 1, e1, e2, e3, einf);
```

Specialized multivectors

`Gaigen 2` generates classes for specialized multivectors.

Examples of specialized multivector specifications:

```
specialization: vectorE3GA(e1, e2, e3);

specialization: line(e1^e2^einf, e1^e3^einf, e2^e3^einf,
                     e1^e0^einf, e2^e0^einf, e3^e0^einf);

specialization: normalizedPoint(e0 = 1, e1, e2, e3, einf);
```

All basis blades which are *not* listed in the specification are assumed to be constant 0. Memory is only allocated for non-constant coordinates.

UNIVERSITEIT
VAN
AMSTERDAM

Specializing Functions

Example of generated code: equation $\mathbf{P} = \mathbf{C} \cdot \mathbf{S}$.

Specializing Functions

Example of generated code: equation $\mathbf{P} = \mathbf{C} \cdot \mathbf{S}$.

```
inline pointPair innerProduct(const dualCircle& C, const sphere& S) {
    return pointPair(
        -C.c[2] * S.c[2] + C.c[4] * S.c[4] - C.c[1] * S.c[1],
        C.c[0] * S.c[1] - C.c[2] * S.c[3] + C.c[5] * S.c[4],
        C.c[1] * S.c[3] + C.c[0] * S.c[2] + C.c[3] * S.c[4],
        C.c[9] * S.c[1] + C.c[8] * S.c[4] - C.c[2] * S.c[0],
        C.c[6] * S.c[4] + C.c[9] * S.c[3] - C.c[0] * S.c[0],
        -C.c[1] * S.c[0] - C.c[9] * S.c[2] + C.c[7] * S.c[4],
        -C.c[4] * S.c[0] + C.c[8] * S.c[2] + C.c[7] * S.c[1],
        -C.c[6] * S.c[1] - C.c[5] * S.c[0] + C.c[8] * S.c[3],
        -C.c[6] * S.c[2] - C.c[3] * S.c[0] - C.c[7] * S.c[3],
        C.c[5] * S.c[2] - C.c[4] * S.c[3] - C.c[3] * S.c[1]);
}
```

Gaigen 2 Demo

- Installing.
- Writing a specification for an algebra.
- Generating the code.
- Walkthrough of default generated code.
- Profiling.
- The profile.
- Walkthrough of optimized code.

Conclusion / Discussion

Gaigen 2: efficient, usable for big projects, but a bit rough on the edges.

Conclusion / Discussion

Gaigen 2: efficient, usable for big projects, but a bit rough on the edges.

Possibly will develop Gaigen 2.5: simpler, cleaner, more pragmatic.

Conclusion / Discussion

Gaigen 2: efficient, usable for big projects, but a bit rough on the edges.

Possibly will develop Gaigen 2.5: simpler, cleaner, more pragmatic.

Is code generation always required for maximum efficiency? (Dietmar Hildenbrand is also moving in that direction)