

Writing PHP Extensions

php[tek] 2024

Derick Rethans

derick@php.net — @derickr@phpc.social

<https://derickrethans.nl/talks/phpext-tek24>

About Me

Derick Rethans

- I'm European, living in London
- ~~PHP 7.4 Release Manager~~
- **PHP Foundation**
- **Xdebug** & PHP's Date/Time support
- I ♥ 🌐 maps, I ♥ 🍺 beer, I ♥ 🥃 whisky
- mastodon: @derickr@phpc.social



The diagram consists of several horizontal bars of different colors and lengths. At the top, there is a long gray bar. Below it, a shorter gray bar is followed by three small blue squares. To the right of these, there is a blue structure resembling a capital letter 'I'. In the middle, a long gray bar is followed by a blue 'I' structure. A horizontal light gray bar spans the width of the diagram, with the word 'Agenda' centered on it. Below this bar, there is another shorter gray bar followed by three small blue squares, and at the bottom, a long blue bar. The blue 'I' structures are positioned to the right of the gray bars, suggesting a sequence or flow.

Agenda

Questions

- What are extensions?
- Why would you write an extension?
- How do you write an extension?



What are extensions?

- Additional functions and classes
- Written in C
- Either compiled in, or a shared object
- Bundled with PHP, distributed through PECL, or distributed separately



Why write an extension?

- Wrap existing library
- Not everything can be done in userland
- Speed



Wrapping Libraries

- Database drivers: **libmongoc**, **libmysql**, ...
- Image processing: **GD**, **Cairo**, **ImageMagick**, ...
- Image recognition: **OpenCV**, **Dlib**, ...

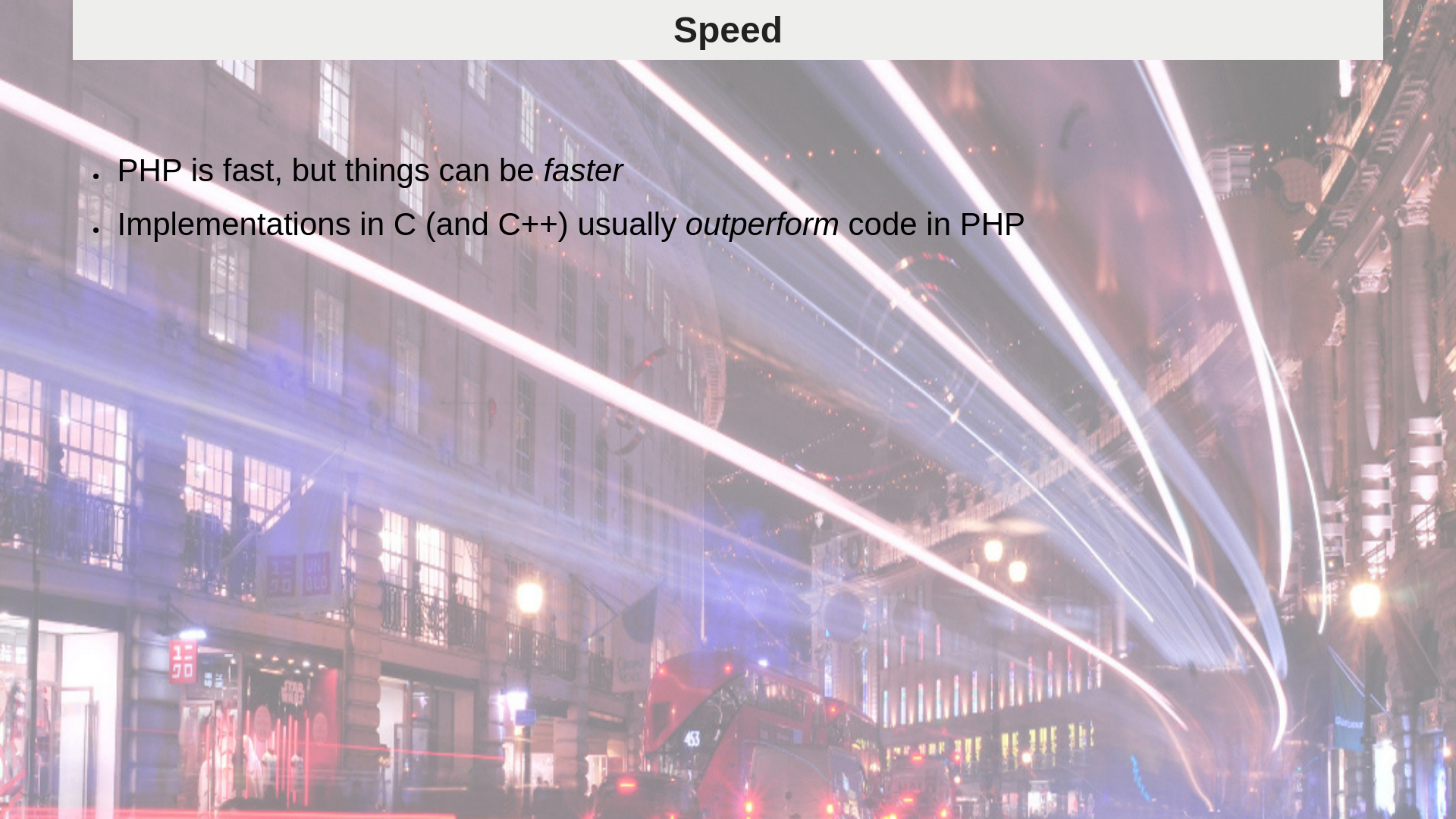
Impossible in userland

- Information or resources shared between requests: **persistent connections, PHP objects**
- Peeking into internals: **vld, Xdebug, opcache**



Speed

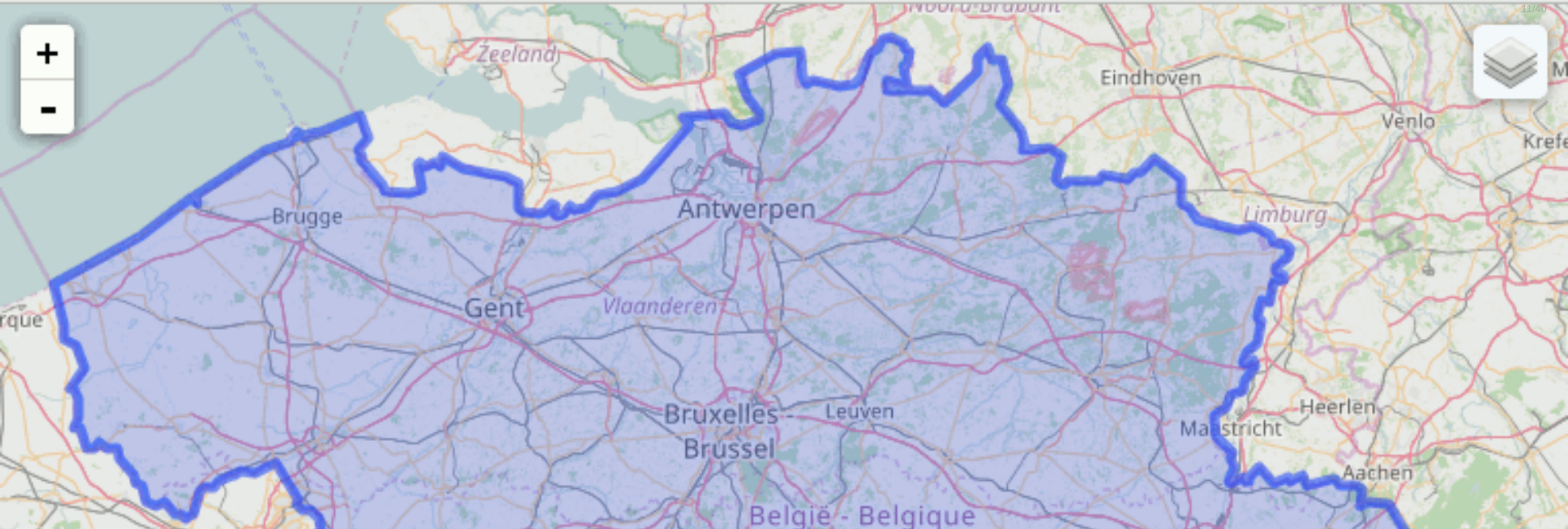
- PHP is fast, but things can be *faster*
- Implementations in C (and C++) usually *outperform* code in PHP



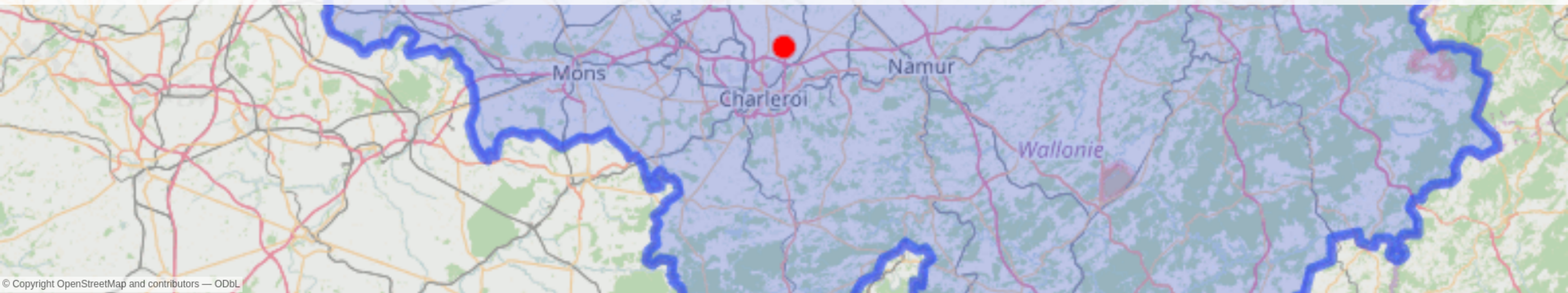
Why not implement something in an extension?

- It takes a lot more time to write
- It takes a lot more time to debug
- It takes a lot more time to maintain





Oh, Belgium!



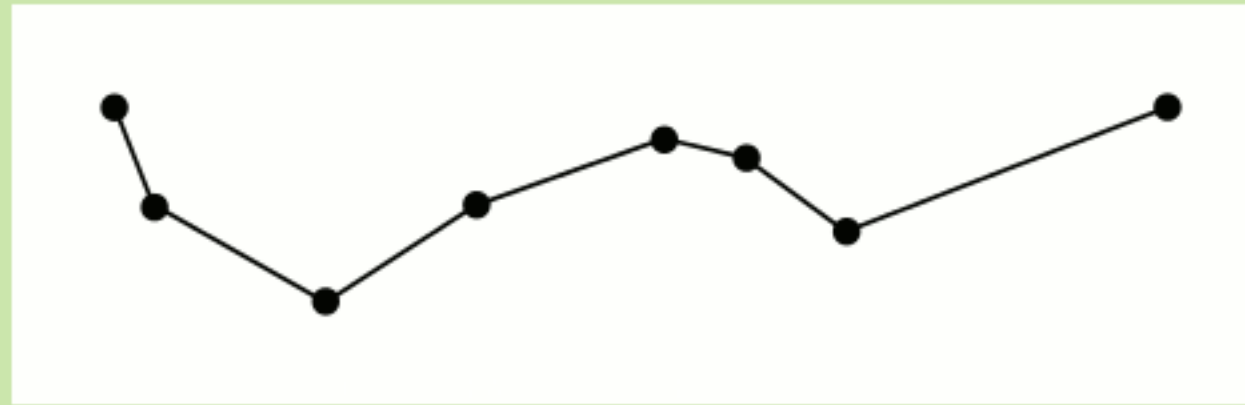
Polygon Simplification

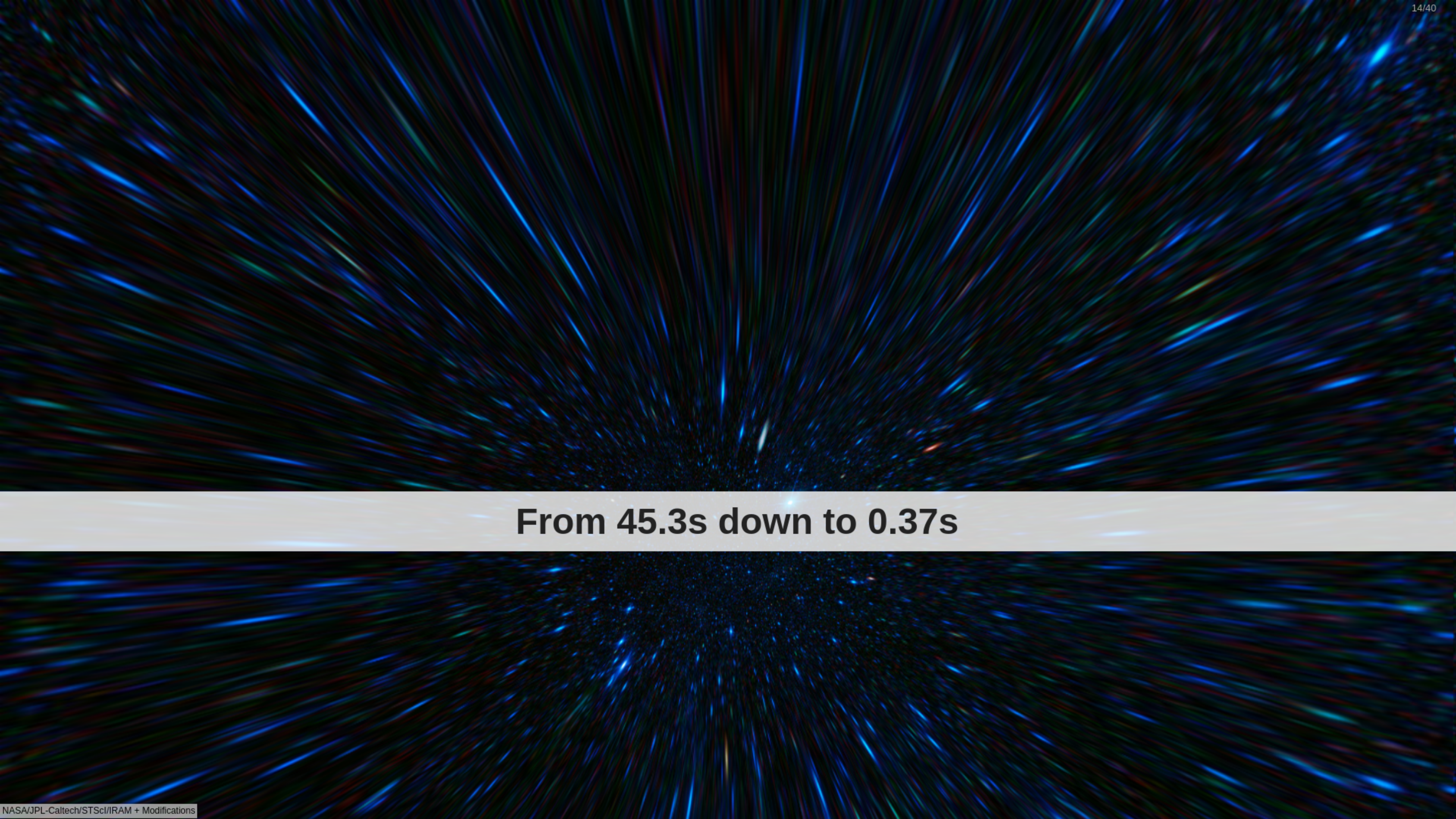
```
[
  "geometry": {
    "type": "Polygon",
    "coordinates": [
      [
        [ 3.0275144577026, 50.770740509033 ],
        [ 3.0138611793518, 50.770278930664 ],
        [ 2.9991388320923, 50.763305664062 ],
        [ 2.9890556335449, 50.761665344238 ],
        [ 2.9853610992432, 50.756999969482 ],
        [ 2.9746110439301, 50.751445770264 ],
        [ 2.9703054428101, 50.751277923584 ],
        [ 2.9549167156219, 50.750751495361 ],
        [ 2.9511666297913, 50.750610351562 ],
        [ 2.9386110305786, 50.742416381836 ],
        [ 2.9403610229492, 50.73405456543 ],
        [ 2.936666727066, 50.729389190674 ],
        [ 2.9306666851044, 50.72785949707 ],
        [ 2.9279444217682, 50.723888397217 ],
        [ 2.9258055686951, 50.712223052979 ],
        [ 2.9244999885559, 50.704971313477 ],
        [ 2.9100832939148, 50.702026367188 ],
        [ 2.9036111831665, 50.700389862061 ],
        [ 2.9013612270355, 50.699806213379 ],
        [ 2.8941388130188, 50.701332092285 ],
        [ 2.88047224617, 50.705207006826 ],

```


Ramer–Douglas–Peucker algorithm

- Recursive algorithm for line/polygon simplification
- Computationally intensive
- 100 runs with epsilon of 0.01:
 - PHP function: **45.3s**
 - C implementation in extension: **0.37s**





From 45.3s down to 0.37s

What makes up an Extension?

- `config.m4 / config.w32`
configure options and build instructions
- `php_extension.h`
header files and glue
- `extension.c`
extension definition and function implementation
- `lib/algorithm.c`
C implementation of algorithms
- `tests/*.phpt`
Tests written in PHP

Build Instructions

config.m4

Comments:

```
dn1 config.m4 for extension geospatial
```

Configure option definition:

```
PHP_ARG_ENABLE(geospatial, whether to enable geospatial support,  
[ --enable-geospatial          Enable geospatial support])
```

Build instructions:

```
if test "$PHP_GEOSPATIAL" != "no"; then  
    PHP_NEW_EXTENSION(geospatial, geospatial.c geo_array.c, $ext_shared)  
fi
```


Build Instructions

config.w32

Comments:

```
// vim:ft=javascript
```

Configure option definition:

```
ARG_ENABLE("geospatial", "enable geospatial support", "no");
```

Build instructions:

```
if (PHP_GEOSPATIAL != "no") {  
    EXTENSION("geospatial", "geospatial.c geo_array.c");  
}
```


Building the Extension

- `phpize`
Brings in headers and creates `configure` script
- `./configure`
Runs system checks, and creates `Makefile`
- `make`
Builds the extension

Header File

Guard:

```
#ifndef PHP_GEOSPATIAL_H  
#define PHP_GEOSPATIAL_H
```

```
#define PHP_GEOSPATIAL_VERSION "0.1.1-dev"
```

Module definition:

```
extern zend_module_entry geospatial_module_entry;  
#define phpext_geospatial_ptr &geospatial_module_entry
```

Module initialiser declarations:

```
PHP_MINIT_FUNCTION(geospatial);  
PHP_MININFO_FUNCTION(geospatial);
```

Forward declarations:

```
PHP_FUNCTION(rdp_simplify);  
  
#endif /* PHP_GEOSPATIAL_H */
```


Extension Set-Up

Headers and Function Definition

Header inclusions:

```
#include "php.h"  
#include "php_ini.h"  
#include "ext/standard/info.h"  
  
#include "php_geospatial.h"  
#include "php_geospatial_arginfo.h"  
  
#include "Zend/zend_exceptions.h"  
#include "ext/spl/spl_exceptions.h"
```


Extension Set-Up

Function Definitions

php_geospatial.stub.php

```
<?php
/** @generate-function-entries */
function rdp_simplify(array $points, float $epsilon): array {}
```

Generates php_geospatial_arginfo.h:

```
/* This is a generated file, edit the .stub.php file instead.
 * Stub hash: e128dccdbd9aa407d95c5b62e8000cb12d0bffa3 */

ZEND_BEGIN_ARG_WITH_RETURN_TYPE_INFO_EX(arginfo_rdp_simplify, 0, 2, IS_ARRAY, 0)
    ZEND_ARG_TYPE_INFO(0, points, IS_ARRAY, 0)
    ZEND_ARG_TYPE_INFO(0, epsilon, IS_DOUBLE, 0)
ZEND_END_ARG_INFO()

ZEND_FUNCTION(rdp_simplify);

static const zend_function_entry ext_functions[] = {
    ZEND_FE(rdp_simplify, arginfo_rdp_simplify)
    ZEND_FE_END
};
```


Extension Set-Up

Module Definition

Module (Extension) definition:

```
zend_module_entry geospatial_module_entry = {  
#if ZEND_MODULE_API_NO >= 20010901  
    STANDARD_MODULE_HEADER,  
#endif  
    "geospatial",  
    ext_functions,  
    PHP_MINIT(geospatial),  
    NULL,  
    NULL,  
    NULL,  
    PHP_MINFO(geospatial),  
#if ZEND_MODULE_API_NO >= 20010901  
    PHP_GEOSPATIAL_VERSION,  
#endif  
    STANDARD_MODULE_PROPERTIES  
};
```

Handle Shared Object Loading:

```
#ifdef COMPILE_DL_GEOSPATIAL  
ZEND_GET_MODULE(geospatial)  
#endif
```


Extension Set-Up

Auxiliary Init and Info sections

Module INITialisation

```
PHP_MINIT_FUNCTION(geospatial)
{
    REGISTER_DOUBLE_CONSTANT("GEO_DIAMETER", GEO_DIAMETER, CONST_CS|CONST_PERSISTENT);
    return SUCCESS;
}
```

Module INFO

```
PHP_MINFO_FUNCTION(geospatial)
{
    php_info_print_table_start();
    php_info_print_table_header(2, "Geospatial functions", "enabled");
    php_info_print_table_row(2, "Version", PHP_GEOSPATIAL_VERSION);
    php_info_print_table_end();
}
```


Implementing the function

- Parse Input Parameters
- Convert Data into Right Format
- Run Algorithm
- Return Result

$$R_{\mu\nu} - \frac{1}{2}Rg_{\mu\nu} = 8\pi GT_{\mu\nu}$$

Parsing Input Parameters

```
PHP_FUNCTION(rdp_simplify)
{
    zval      *points_array;
    double     epsilon;

    ZEND_PARSE_PARAMETERS_START(2, 2)
        Z_PARAM_ ARRAY(points_array)
        Z_PARAM_ DOUBLE(epsilon)
    ZEND_PARSE_PARAMETERS_END();
}
```

- A different macro for each different data type
- Matched with a C data type
- Tests for type matches
- Can deal with optional arguments too

zval?

PHP's representation of a value

- Contains the type: `Z_TYPE_P(points_array)`
- Contains the value: `Z_ARRVAL_P(points_array)`

`IS_DOUBLE`, `IS_STRING`, `IS_ARRAY`, ...

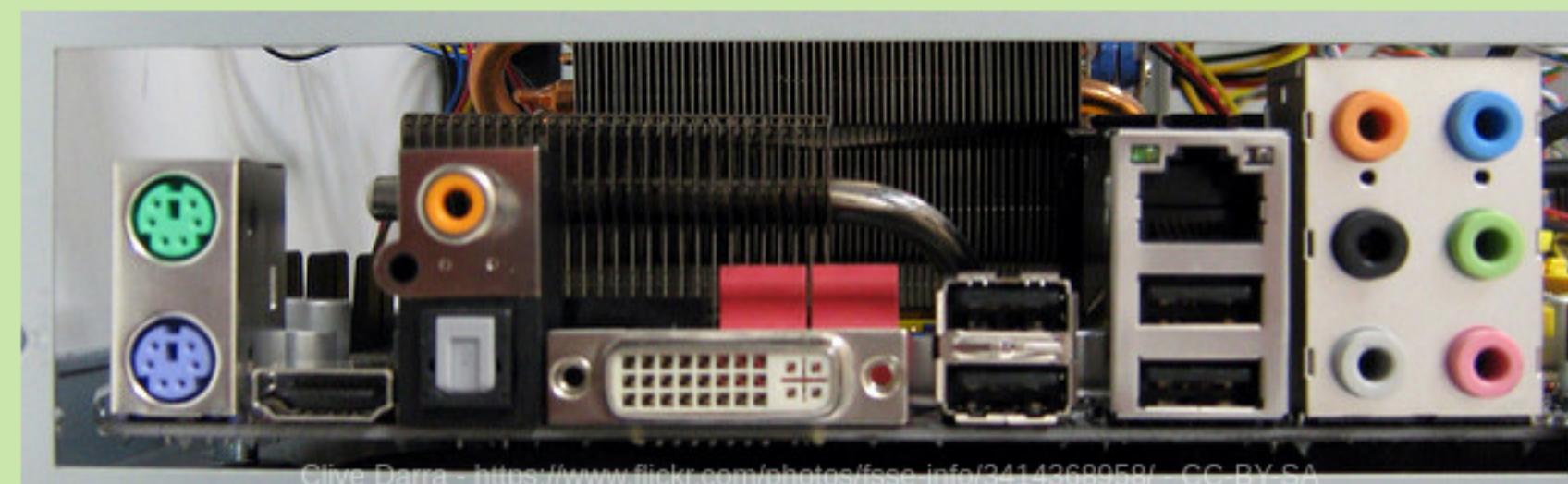
`Z_DVAL_P(lat)`, `Z_STRVAL_P(type)`, ...

```
if (Z_TYPE_P(points_array) != IS_ARRAY) {  
    return;  
}
```


Parsing Input Parameters

macro	variable	C data-type
Z_PARAM_ARRAY	array	zval*
Z_PARAM_ARRAY_HT	array	HashTable
Z_PARAM_BOOL	boolean	zend_bool
Z_PARAM_DOUBLE	double	double
Z_PARAM_LONG	long	zend_long
Z_PARAM_OBJECT_OF_CLASS	object of specific type	zval*, zend_class_entry
Z_PARAM_PATH	path	zend_string*
Z_PARAM_STRING	string	char*, size_t
Z_PARAM_STR	string	zend_string*

https://wiki.php.net/rfc/fast_zpp



Converting from PHP variables

```
points = geo_hashtable_to_array(points_array TSRMLS_CC);  
if (!points) {  
    return;  
}
```


Converting from PHP variables

```

geo_array *geo_hashtable_to_array(zval *array TSRMLS_DC)
{
    geo_array *tmp;
    int element_count, i = 0;
    zval *entry;
    double lon, lat;

    element_count = zend_hash_num_elements(Z_ARRVAL_P(array));
    tmp = geo_array_ctor(element_count);

    ZEND_HASH_FOREACH_VAL(Z_ARRVAL_P(array), entry) {
        if (!parse_point_pair(entry, &lon, &lat TSRMLS_CC)) {
            goto failure;
        }

        tmp->x[i] = lon;
        tmp->y[i] = lat;
        tmp->status[i] = 1;

        i++;
    } ZEND_HASH_FOREACH_END();

    return tmp;

```

failure:

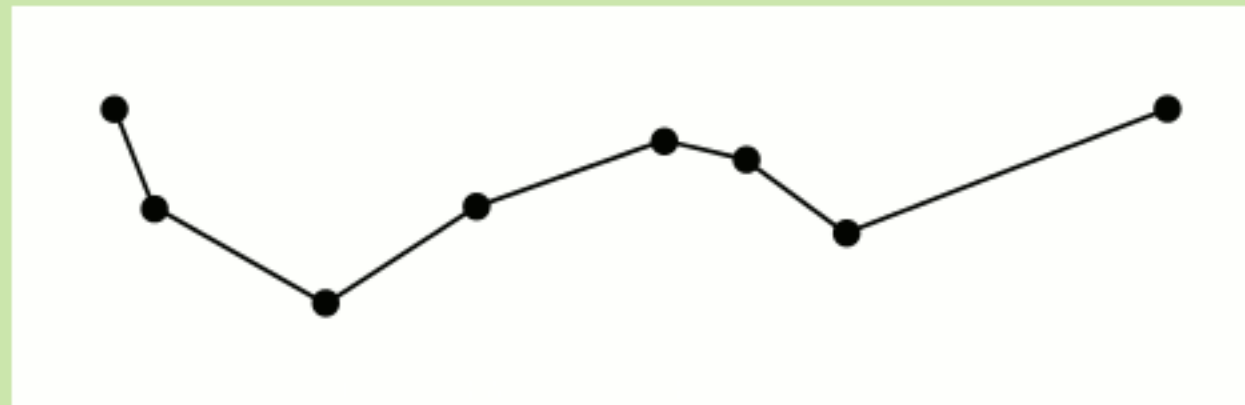
```

    geo_array_dtor(tmp);

```


Run Algorithm

```
rdp_simplify(points, epsilon, 0, points->count - 1);
```



RDP (PHP)

```

static private function simplifyInternal( &$points, $epsilon, $start, $end )
{
    $firstPoint = $points[$start];
    $lastPoint = $points[$end];
    $index = -1;
    $dist = 0;

    if ( $end - $start < 2 ) {
        return;
    }

    for ( $i = $start + 1; $i < $end; $i++ ) {
        if ( !isset( $points[$i] ) ) {
            continue;
        }

        $cDist = self::findPerpendicularDistance( $points[ $i ], $firstPoint, $lastPoint );

        if ( $cDist > $dist ) {
            $dist = $cDist;
            $index = $i;
        }
    }

    if ( $dist > $epsilon ) {
        self::simplifyInternal( $points, $epsilon, $start, $index );
    }
}

```


RDP (C)

```

void rdp_simplify(geo_array *points, double epsilon, int start, int end)
{
    double firstX = points->x[start], firstY = points->y[start];
    double lastX = points->x[end], lastY = points->y[end];
    int    index = -1, i;
    double dist  = 0.0, current_dist;

    if (end - start < 2) {
        return;
    }

    for (i = start + 1; i < end; i++) {
        if (!points->status[i]) {
            continue;
        }

        current_dist = rdp_find_perpendicular_distable(points->x[i], points->y[i],

        if (current_dist > dist) {
            dist = current_dist;
            index = i;
        }
    }

    if (dist > epsilon) {
        rdp_simplify(points, epsilon, start, index);
    }
}

```

Return Result

```
PHP_FUNCTION(rdp_simplify)
{
...
    zval      pair;
...
    array_init(return_value);

    for (i = 0; i < points->count; i++) {
        if (points->status[i]) {
            array_init(&pair);
            add_next_index_double(&pair, points->x[i]);
            add_next_index_double(&pair, points->y[i]);
            add_next_index_zval(return_value, &pair);
        }
    }

    geo_array_dtor(points);
}
```


Returning Scalar Values

Set Return Value:

```
RETVAL_FALSE;
```

```
RETVAL_STRING(trace_fname);
```

Set Value and Return Immediately:

```
RETURN_BOOL(!mongoc_cursor_is_alive(intern->cursor));
```

```
RETURN_LONG(zend_memory_usage(0));
```

```
RETURN_STRING(i->function.function);
```

Memory Management

- If you allocate something, you need to free it
- PHP Memory Management helpers: `emalloc`, `efree`
- To prevent PHP from cleaning up input arguments:

```
Z_ADDREF(zval);
```

- But make sure to remove the reference later:

```
zval_ptr_dtor(&zval);
```


Tests

- Go into the `tests/` directory
- `phpt` format
- Run with `make test` in top source directory

Example Test File

tests/rdp-belgium.phpt

--TEST--

Test for rdp_simplify

--FILE--

<?php

\$json = json_decode(file_get_contents(dirname(__FILE__) . '/rdp-belgium.json'

\$points = \$json[0]->geometry->coordinates[0];

\$result = RDP_Simplify(\$points, 0.001);

var_dump(count(\$result));

\$result = RDP_Simplify(\$points, 0.01);

var_dump(count(\$result));

?>

--EXPECT--

int(1029)

int(261)

- Sections: TEST, INI, FILE, EXPECT, EXPECTF
- Simple PHP script

Debugging Tools

`gdb`

- allows the inspection of programs when they're running
- `export USE_ZEND_ALLOC=0`

`valgrind`

- tells you about memory corruption and allocation issues
- `export ZEND_DONT_UNLOAD_MODULES=1`



Any Queries?

Let's Get Practical

Project Directory:

```
mkdir tutorial-geospatial && cd tutorial-geospatial && git init
```

Skeleton:

```
php ~/dev/php/php-src.git/ext/ext_skel.php  
  --ext geospatial --author "Your Name" --dir . --onlyunix
```

Algorithms:

```
wget https://drck.me/files/dump/rdp.tgz -O /tmp/rdp.tgz
```

```
mkdir lib && cd lib && tar xvzf /tmp/rdp.tgz && cd -
```

Test File:

```
https://drck.me/files/dump/geojson-belgium.json
```

PDF with Code as Diffs:

```
https://drck.me/files/dump/tutorial-geospatial.pdf
```

GIT repository:

```
https://github.com/derickr/tutorial-geospatial
```



Slides & Resources

Contact

derick@php.net—@derickr