3D rendering with PHP

Die neue PEAR-Klasse Image_3D bietet die Moeglichkeit nur mit PHP5 3d-Grafiken zu rendern

Der Vortragende

Kore Nordmann

- Student der Informatik an der Universität Dortmund
- System Developer bei der ezSystems GmbH in Dortmund
- PHP-Entwickler seit 2001

Sinn

Wo ist der Sinn eines 3D-Renderes in PHP

- Proof of Concept
- Datenvisualisierung

Überblick

- Möglichkeiten
  - Objekte & Lichter
  - Renderer
  - Treiber
- API & Beispiele
  - Aufbau einer Szene
  - Eigenes Objekt erstellen
  - Eigenen Treiber schreiben
- Geschwindigkeit
- Ausblick

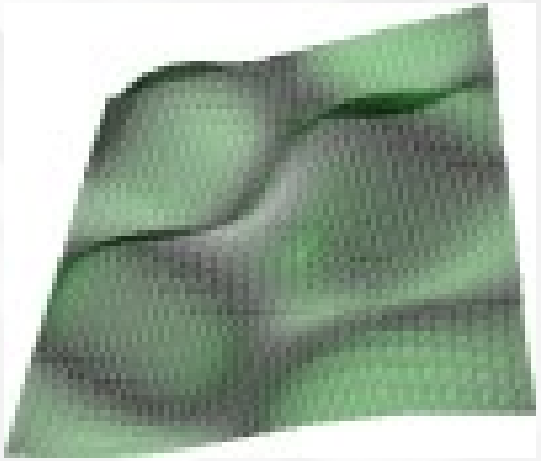
Objekte & Lichter (1)

- Objekte sind alle zeichenbaren Elemente einer Szene
  - Implementieren Image_3D_Interface_Paintable
  - Gruppen von Polygonen (Punkten), um gemeinsame Transformationen zu vereinfachen
  - Objekthierarchien / -bäume sind möglich

Objekte & Lichter (2)

Beispiele für Objekte in Image_3D

- Würfel
- Kugel
- Karten
- 3ds-Importer

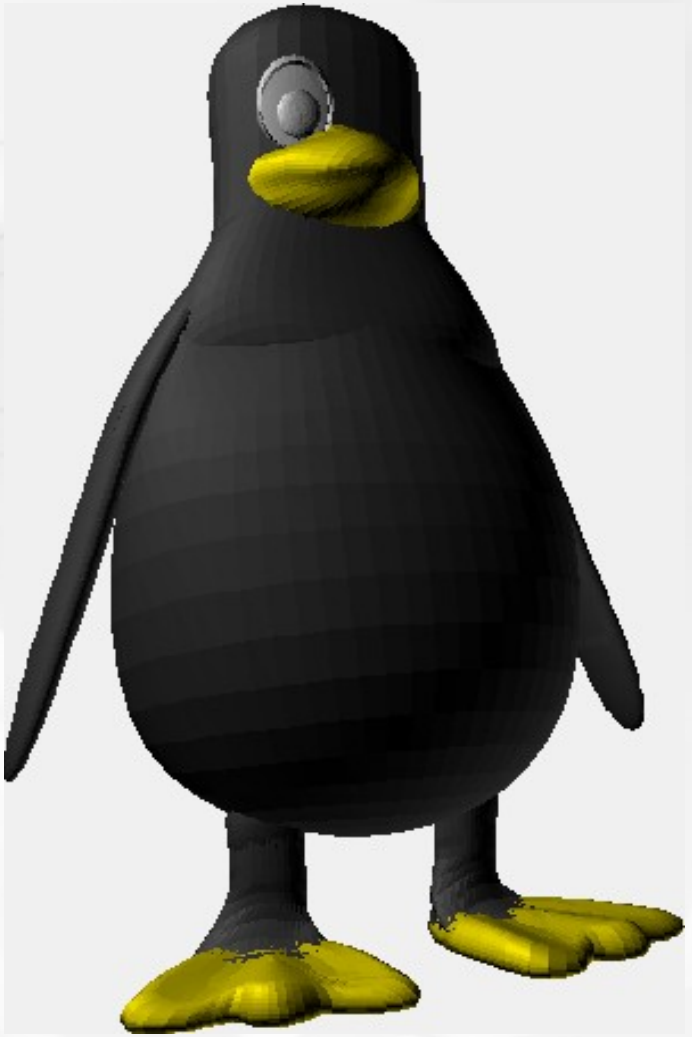


Objekte & Lichter (3)

- Image_3D_Interface_Paintable
  - setColor(Image_3D_Color $color)
    - Setzt die Farbe für alle Polygone eines Objektes
  - transform(Image_3D_Matrix $matrix)
    - Führt Transformationen auf allen Punkten der Polygone eines Objekte durch

Objekte & Lichter (4)

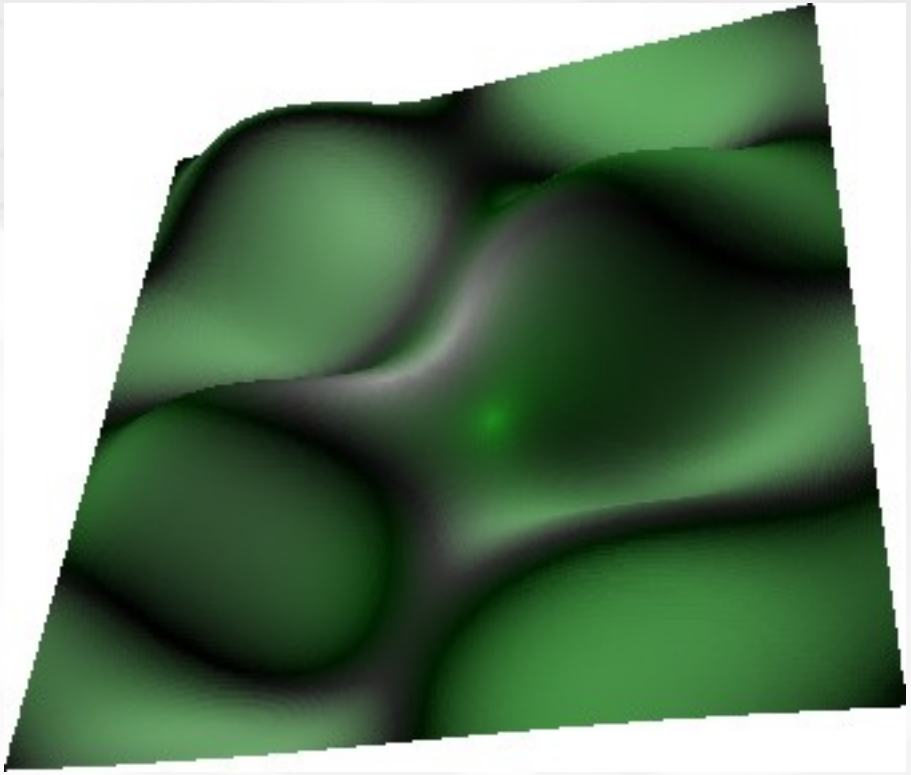
- Objektbäume
  - Setzen von Eigenschaften für Teilobjekte
  - Transformationen auf das ganze Objekt



Objekte & Lichter (5)

Beliebig viele Lichtquellen werden unterstützt

- Punktlichtquellen
- Farbige Lichter



Objekte & Lichter (6)

- Image_3d_Polygon und Image_3d_Point implementieren Image_3D_Interface_Enlightenable
  - getColor()
  - getNormale()
  - getPosition()

- Image_3d_Light berechnet aus Position des Lichtes, des Objektes und des Normalenvektors des Objektes die Lichteinstrahlung

Objekte & Lichter (7)

Der Renderer berechnet für jedes Polygon für alle Lichter die resultierende Farbe

```
foreach ($this->_polygones as $polygon)
   $polygon->calculateColor($this->_lights);
```

Kompelxität der Berechnung O(n*m)

n = Anzahl der Objekte

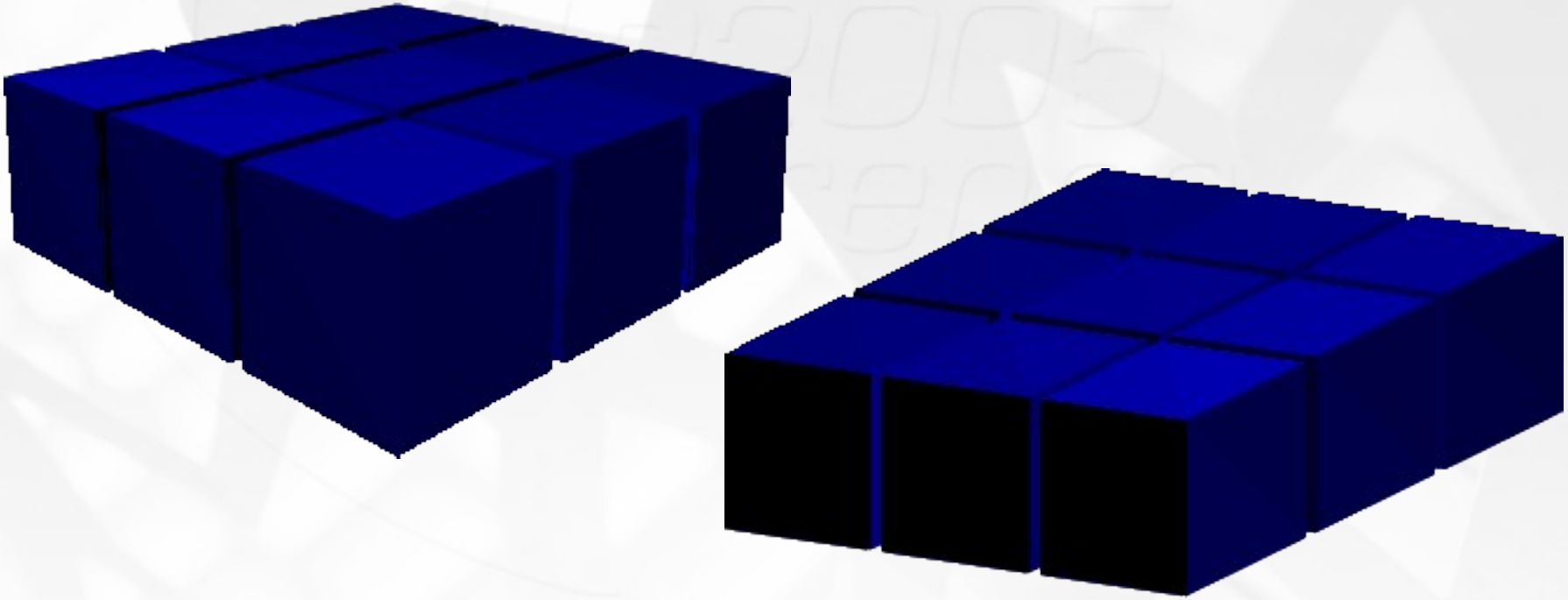
m = Anzahl der Lichter

Renderer (1)

- Erstellen aus Polygonen, Lichtern die fertige Szene
  - Berechnung der Farben abhängig vom eingestelltem Shading
    - None, Flat, Gouraud
- Implementieren die Funktion zur Berechnung der 2d-Abbildung der 3d-Koordinaten
  - Perspektivisch
  - Isometrisch

Renderer (2)

Perspektivisch <-> Isometrisch



Treiber (1)

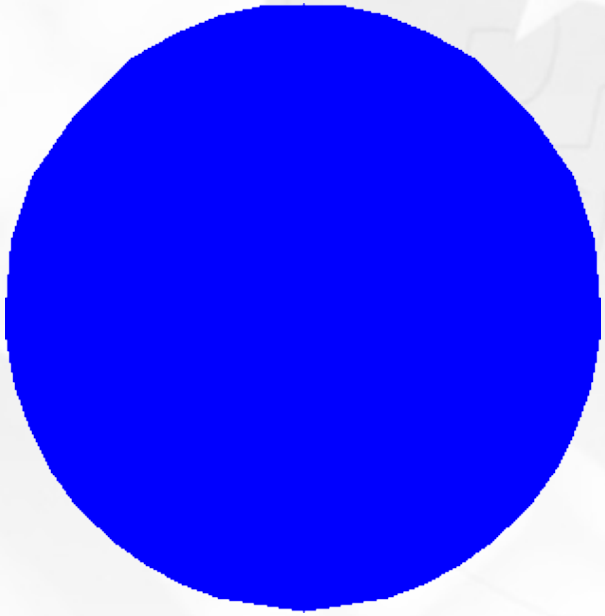
- Alle Polygone werden dem Treiber uebergeben, der diese rendert
  - Zwei verschiedene Methoden zum Zeichnen der Polygone
    - drawPolygon(Image_3D_Polygon $polygon)
    - drawGradientPolygon(Image_3D_Polygon $polygon)

Treiber (2)

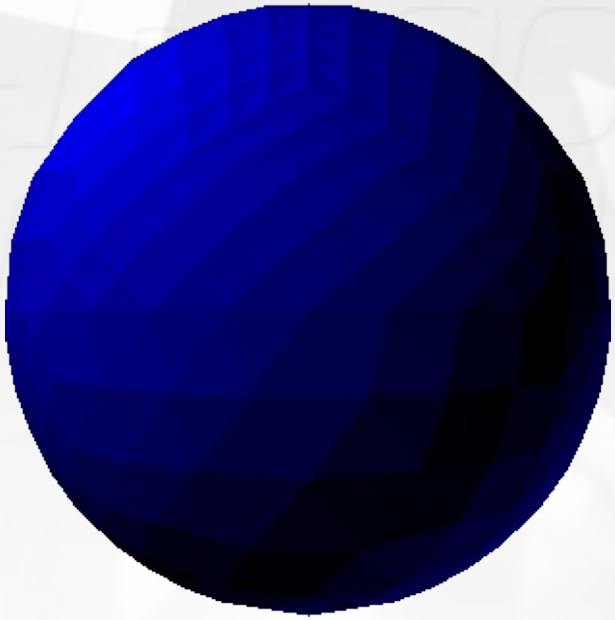
- GD-Treiber
  - Benutzt die gdlib
  - Kein ZBuffer, kein Gouraud-Shading
- SVG
  - Keine PHP-Extensions noetig
  - Gauroud-Shading
  - Kein ZBuffer

Treiber (3)

None



Flat

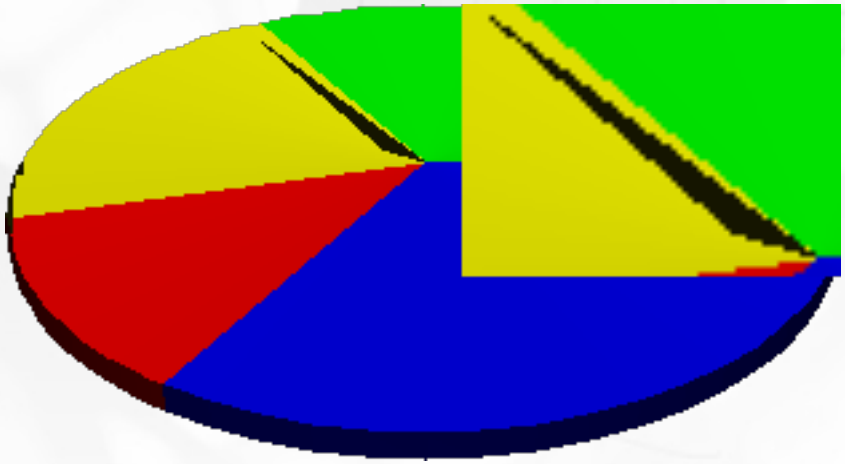


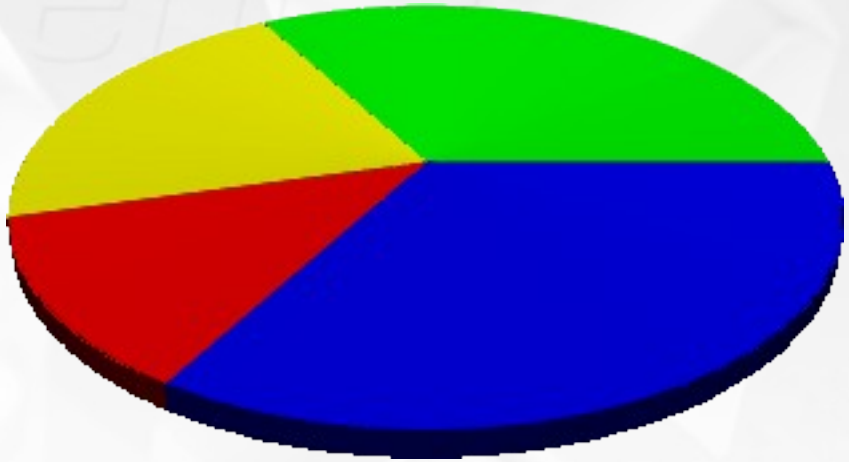
Gouraud

Treiber (4)

ZBuffer

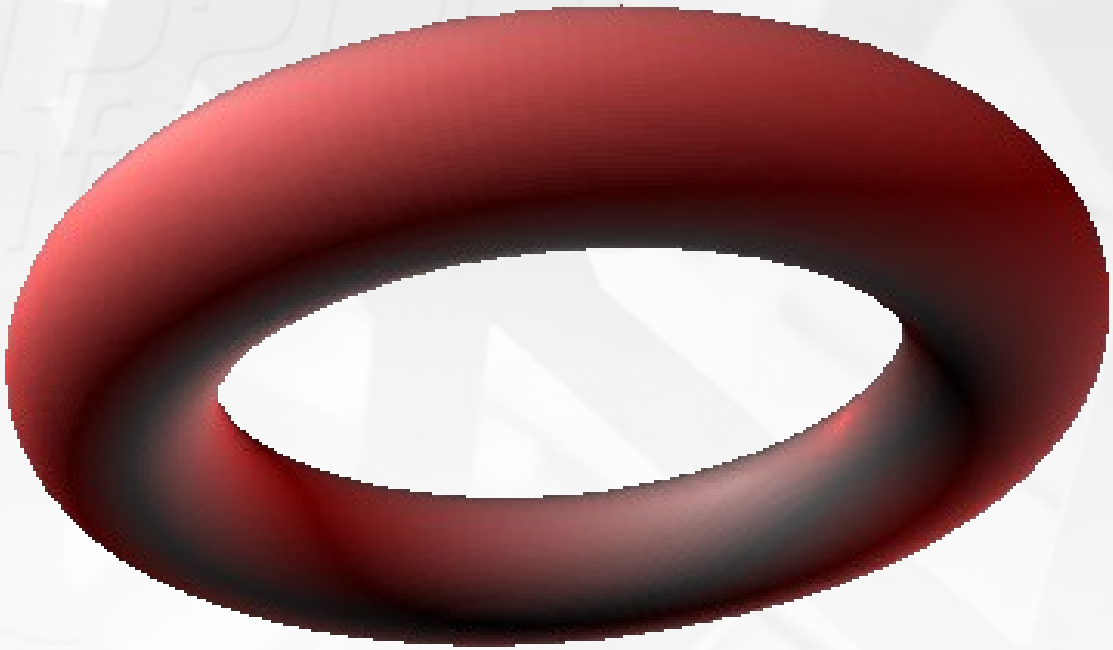
- Treiber mit z-buffering
- Langsamer, aber keine falschen Polygonreinfolgen





ASCII & SVGRotate

Beispiele



Aufbau einer Szene (1)

Initialisierung der Welt

```
require_once('Image/3D.php');

// Erstellen der Welt
$world = new Image_3D();
$world->setColor(new Image_3D_Color(255, 255, 255));

// Erstes Licht einfuegen
$light = $world->createLight(-2000, -2000, -2000);
$light->setColor(new Image_3D_Color(255, 255, 255));

$redLight = $world->createLight(90, 0, 50);
$redLight->setColor(new Image_3D_Color(255, 0, 0));
```

Aufbau einer Szene (2)

Erstellung der Objekte

```
$sphere = $world->createObject('sphere', array('r' => 60,
   'detail' => 5));
$sphere->setColor(new Image_3D_Color(150, 150, 150));
$sphere->transform($world->createMatrix('Move', array(50, 30,
   0)));


$text = $world->createObject('3ds', 'models/Image_3D.3ds');
$text->setColor(new Image_3D_Color(255, 255, 255, 180));
$text->transform($world->createMatrix('Rotation', array(90, 0,
   0)));
$text->transform($world->createMatrix('Scale', array(5, 5,
   5)));
$text->transform($world->createMatrix('Move', array(0, -40,
   0)));
```

Aufbau einer Szene (3)

```
$world->createRenderer('perspectively');
$world->createDriver('SVG');

$world->render(400, 200, 'example.svg');
```

Eigenes Objekt (1)

- Objekte definieren sich durch eine Menge von Polygonen
- Noch nicht existierendes Objekt: Kegel

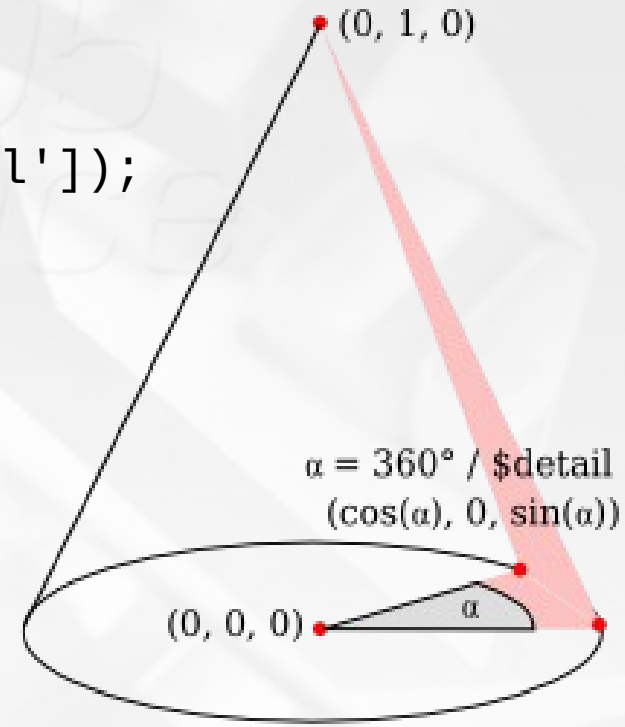
```
require_once('Image/3D/Paintable/Object.php');
class Image_3D_Object_Cone extends Image_3D_Object {
    public function __construct($parameter) {
        parent::__construct();
    }
}
```

- Cone.php in das Verzeichnis user/object/

Eigenes Objekt (2)

Wie kommen die Polygone aufgrund der Parameter zustande?

```
$radius = 1;
$height = 1;
$detail = max(3, (int) $parameter['detail']);
```

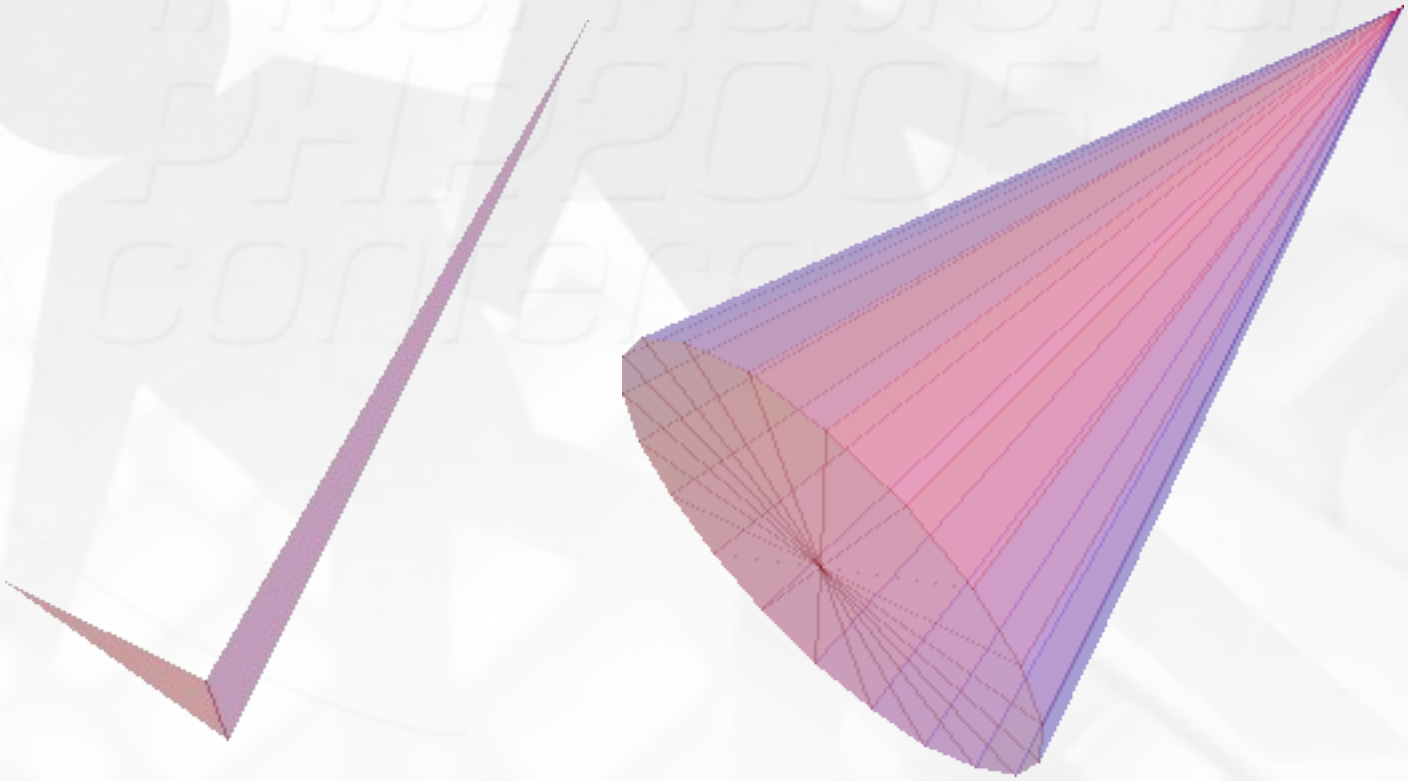


Eigenes Objekt (3)

```
$top = new Image_3D_Point(0, $height, 0);
$bottom = new Image_3D_Point(0, 0, 0);

$last = new Image_3D_Point(1, 0, 0);
$points[] = $last;
for ($i = 1; $i <  $detail; ++$i) {
    $actual = new Image_3D_Point(cos($i / $detail * 2 * pi()), 0, sin($i /
       $detail * 2 * pi()));
    $points[] = $actual;

    $this->_addPolygon(new Image_3D_Polygon($top, $last, $actual));
    $this->_addPolygon(new Image_3D_Polygon($bottom, $last, $actual));
    $last = $actual;
}

$this->_addPolygon(new Image_3D_Polygon($top, $last, $points[0]));
$this->_addPolygon(new Image_3D_Polygon($bottom, $last, $points[0]));
```

Eigenes Objekt (4)

Einzelne Segmente ergeben den Kegel



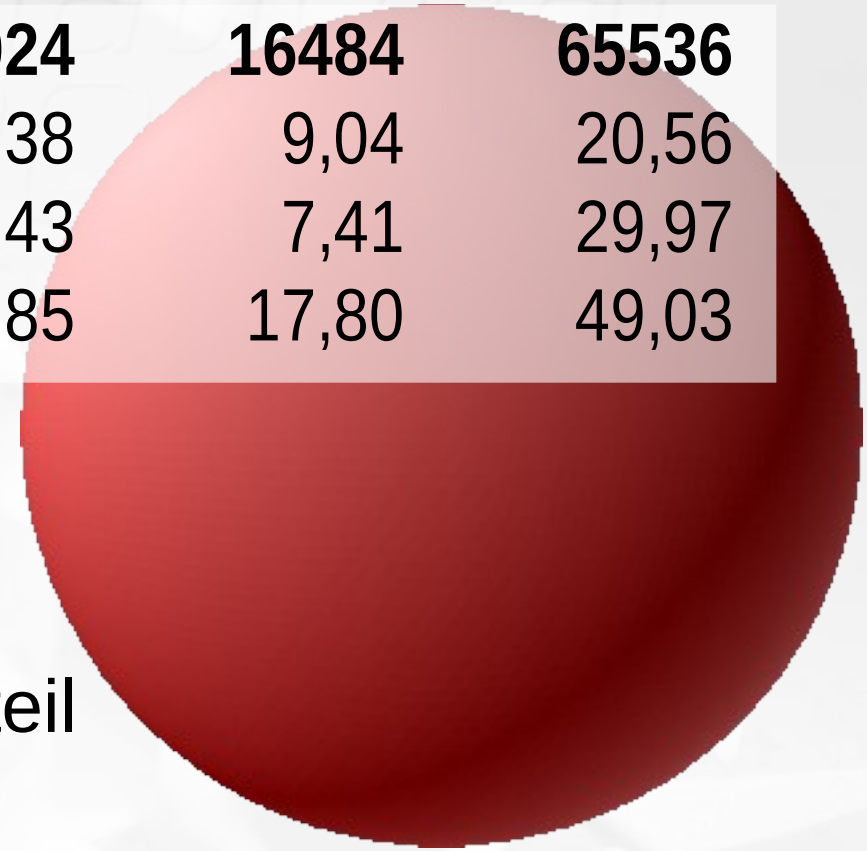
Eigener Treiber

- Treiber beliebig erweiterbar
  - Polygonbasierte Treiber
    - ImageMagick / GraphicsMagick
    - ...
  - Pixelbasierte Treiber
    - Vorlage: ASCII-Treiber
    - HTML
    - ...

Geschwindigkeit

Geschwindkeit der Treiber bei einem Licht

	256	1024	16484	65536
GD	0,13	0,38	9,04	20,56
SVG	0,10	0,43	7,41	29,97
Zbuffer	3,41	4,85	17,80	49,03



PHP 5.1 RC3
(80% Geschwindikeitsvorteil gegenüber PH 5.04)

Ausblick

- Anbindung an Image_Graph
  - Zeichnen von 3D-Graphen
- Warten auf Ideen Image_3D sinnvoll einzusetzen

Ende

Informationen:

http://pearadise.net/view/package/pear.php.net/Image_3D/

Release von Version 0.3 am 7.11

- Added class documentation
- Improved speed
- Added Driver for ASCII-output

Herzlichen Dank fürs Zuhöhren