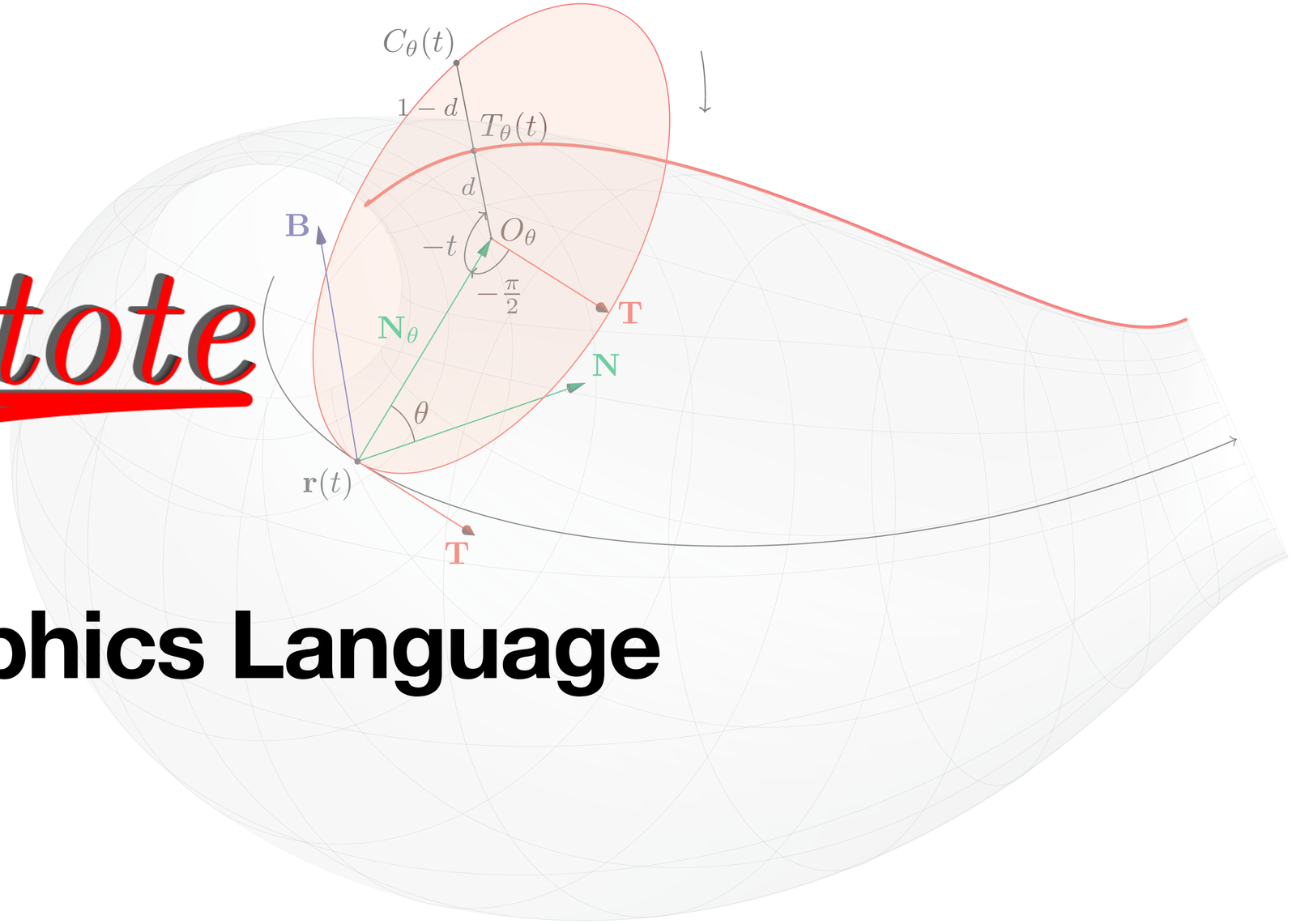
Asymptote



The Vector Graphics Language

이재호 2021년 11월 27일

Asymptote? Ti*k*Z?
비교

- Asymptote와 Ti*k*Z 모두 미려한 그림을 얻을 수 있음.
- Ti*k*Z는 TeX을 기반으로 만들어진 '언어'이고, Asymptote는 C/C++을 기반으로 만들어진 언어이다.
  - 따라서 Asymptote는 외부 프로그램으로 실행시켜야 한다. (TeXLive 배포판에 포함)
  - Asymptote는 fully-featured 프로그래밍 언어.
- Ti*k*Z는 2D가, Asymptote는 3D가 '홈 그라운드'.
  - 물론 각각 3D, 2D 그림을 그릴 수 있다.
  - 다만 Ti*k*Z는 3차원 상의 레이어들의 순서와 겹침을 그려낼 수 없음.

왜 Asymptote가 필요했는가

TikZ에서 Asymptote로

Figures from *Invariance of the Length and the Area of Cycloids* (Choi, 2020)

2019

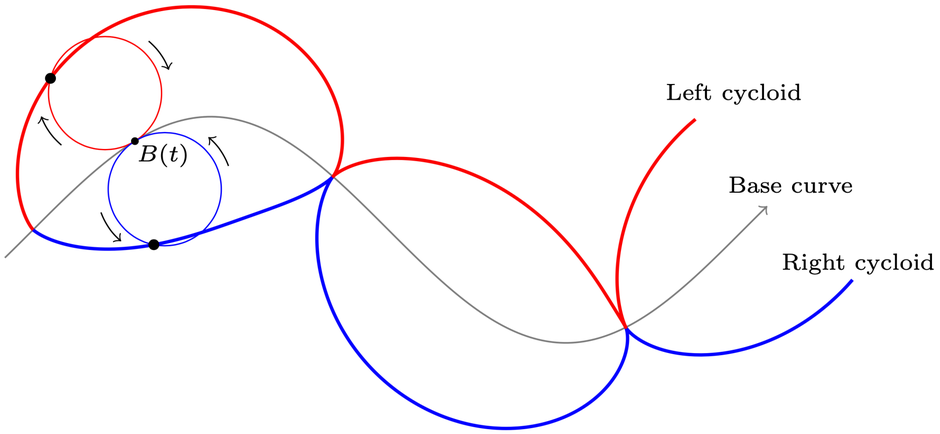


Figure 1. Cycloids along a base curve; left and right ones.

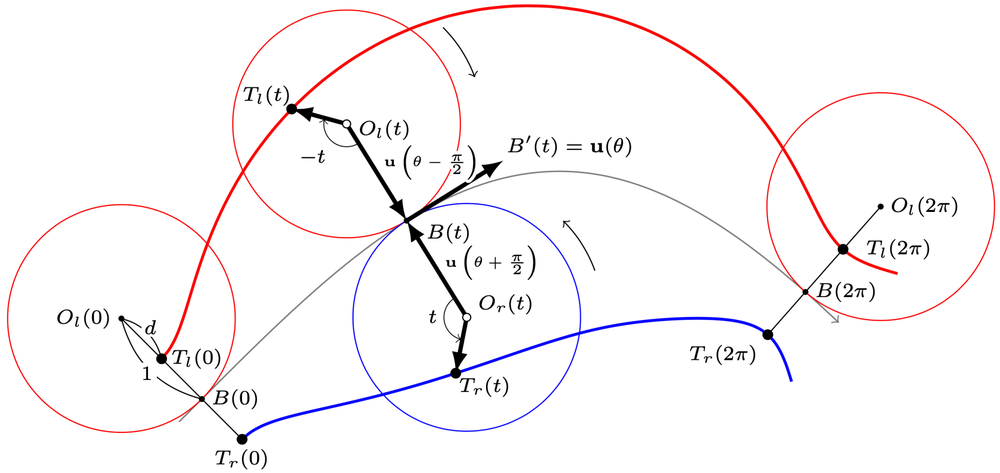


Figure 4. Parametrizations of trochoids: $\overrightarrow{O_lT_l}=d\mathbf{u}\left(\theta(t)-\frac{\pi}{2}-t\right)$ and $\overrightarrow{O_rT_r}=d\mathbf{u}\left(\theta(t)+\frac{\pi}{2}+t\right)$.

왜 Asymptote가 필요했는가

TikZ에서 Asymptote로

Figures from *Invariance of the Area and Volume of Cycloid Surfaces and Trochoid Surfaces* (Choi, 2022)

2020

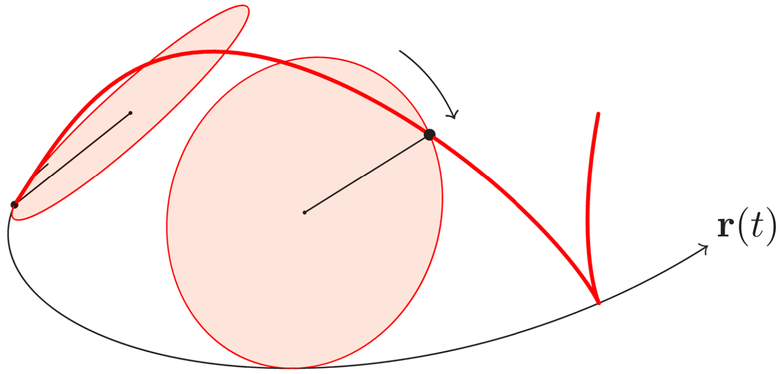


Figure 2. A cycloid along a space curve

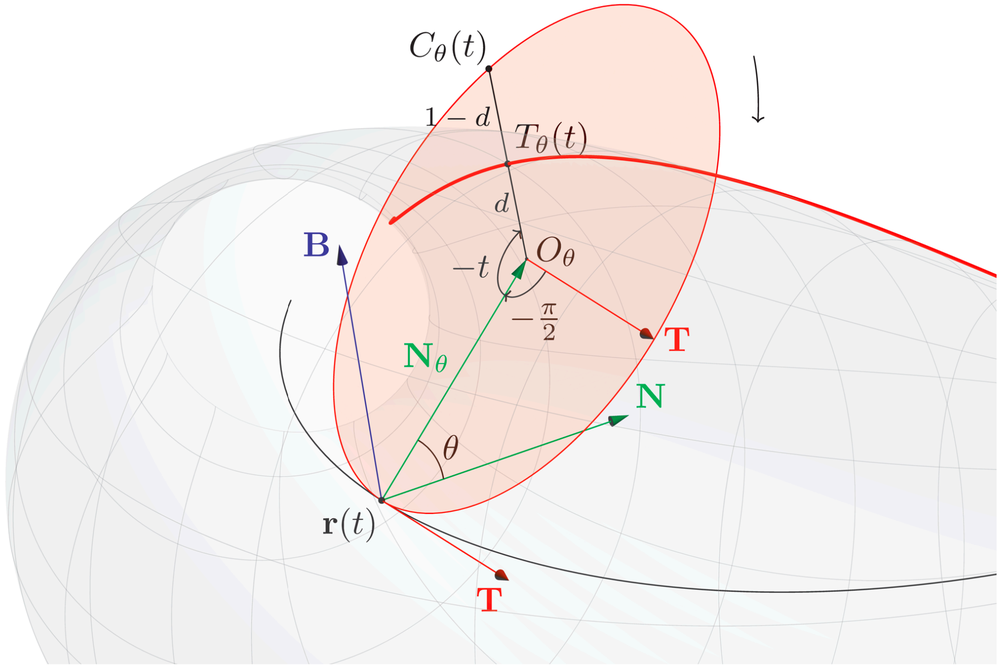


Figure 6. A paramerization $C_\theta(t)$ of the cycloid surface and a paramerization $T_\theta(t)$ of the trochoid surface. : The base curve is parametrized by the arclength as $\mathbf{r}(t)$. The vector $\mathbf{N}_\theta$ is a linear combination of $\mathbf{N}$ and $\mathbf{B}$. The vector $\overrightarrow{O_\theta C_\theta}$ is a linear combination of $\mathbf{T}$ and $\mathbf{N}_\theta$. The arclength along the boundary of the disk between $\mathbf{r}(t)$ and $C_\theta(t)$ is t. The distance between the center of the disk and the marked point is d. The point $T_\theta(t)$ is the internally dividing point of O_θ and $C_\theta(t)$ in the ratio $d : 1-d$.

왜 Asymptote가 필요했는가

TikZ에서 Asymptote로

Figures from *Invariance of the Area and Volume of Cycloid Surfaces and Trochoid Surfaces* (Choi, 2022)

2020

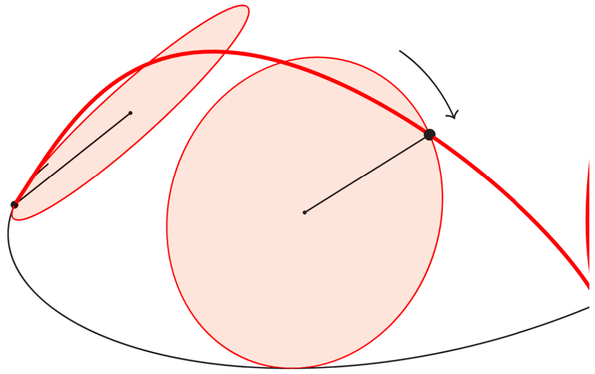
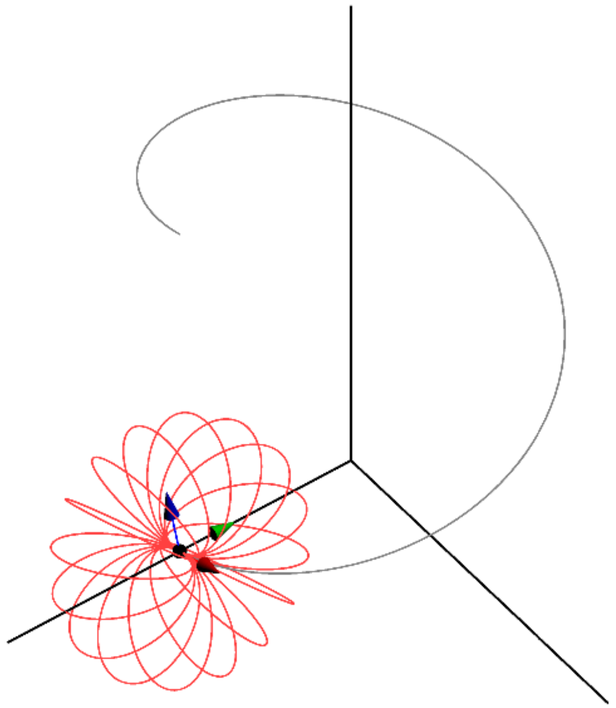
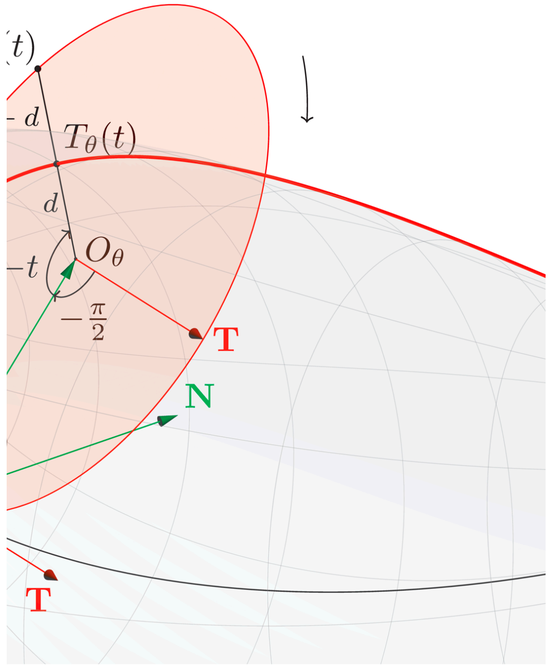


Figure 2. A cycloid along a space c

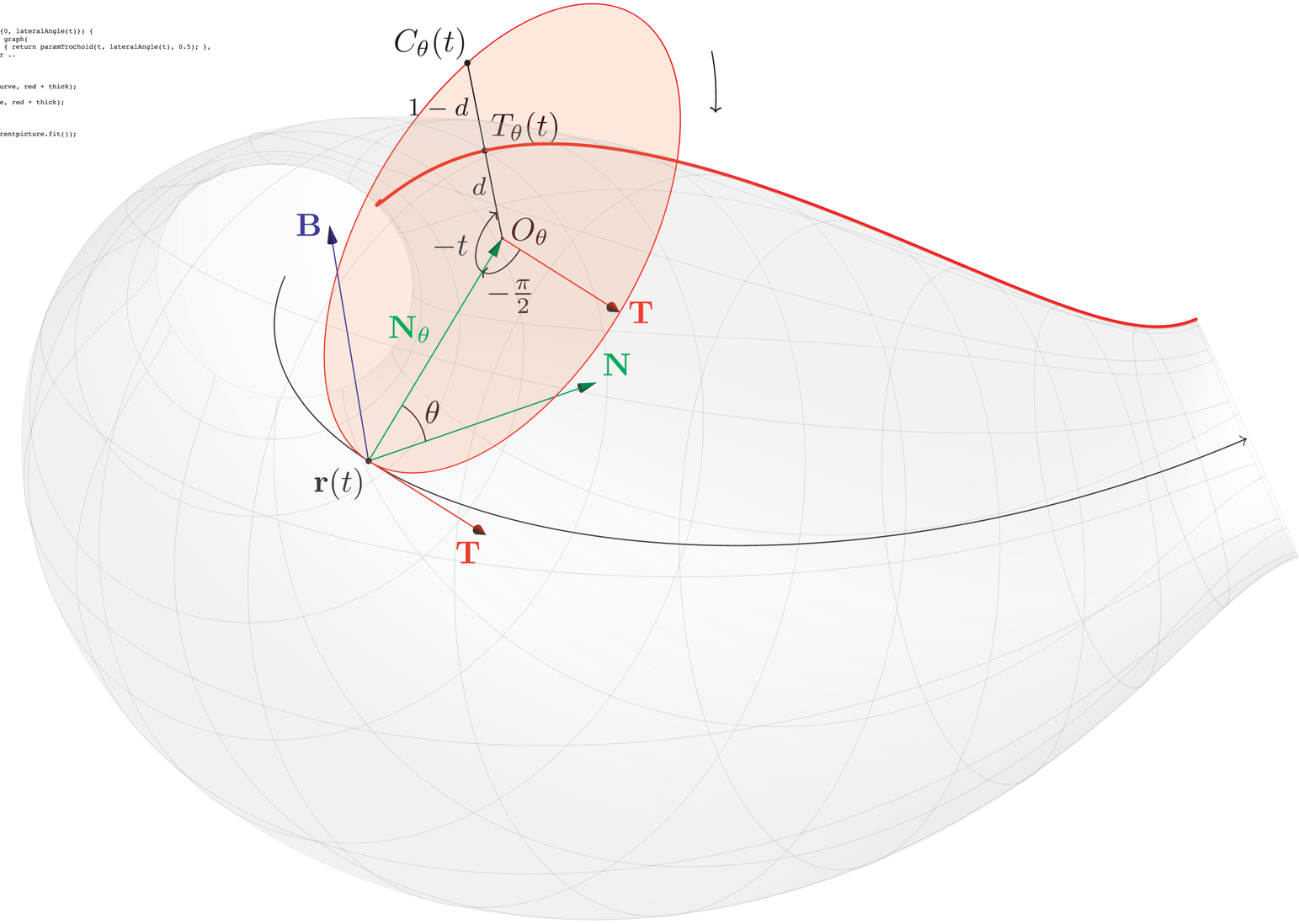




urface and a paramerization $T_\theta(t)$ of the trochoid surface.
s $\mathbf{r}(t)$. The vector $\mathbf{N}_\theta$ is a linear combination of $\mathbf{N}$ and $\mathbf{B}$.
$\mathbf{N}_\theta$. The arclength along the boundary of the disk between
er of the disk and the marked point is d. The point $T_\theta(t)$ is
ratio $d : 1 - d$.

```
////////////////////////////////////
/////////// Preamble //////////////
////////////////////////////////////

// Output settings
settings.outformat = "png";
settings.prc = false;
settings.render = 16;

// Font settings
defaultpen(fontsize(10pt));
// settings.tex="xelatex";
// texpreamble("\usepackage{fontspec}\setmainfont{Noto Serif CJK KR}");
texpreamble("\usepackage{bm}\renewcommand{\vec}[1]{\bm{\mathrm{#1}}}");

// Use 3D graphs
import graph3;

// Figure settings
size(14cm, 0);
currentprojection = orthographic(-1, 6, 5);

////////////////////////////////////
/////////// Mathematics ///////////
////////////////////////////////////

// Global constant representing an infinitestimal.
real epsilon = .0001;

// Returns the derivative f'(t) of a parametrized function f: R -> R^3.
triple derivative(triple f(real), real t, real dx=epsilon)
{
    return (f(t + dx) - f(t - dx)) / 2dx;
}

// Returns a vector starting from `start` with a direction `direction`.
path3 pos(triple start, triple direction)
{
    return shift(start) * (O -- direction);
}

// Base vector
triple base(real t)
{
    return (3cos(t / sqrt(10)), 3sin(t / sqrt(10)), t / sqrt(10));
}

// Tangent vector
triple tangent(real t)
{
    return unit(derivative(base, t));
}

// Normal vector
triple normal(real t)
{
    return unit(derivative(tangent, t));
}

// Binormal vector
triple binormal(real t)
{
    return cross(tangent(t), normal(t));
}

// N_theta vector
triple normal(real t, real theta)
{
    return cos(theta) * normal(t) + sin(theta) * binormal(t);
}

// O_theta (center of a circle)
triple center(real t, real theta)
{
    return base(t) + normal(t, theta);
}

// The rolling circle
path3 cyCircle(real t, real theta, real radius=1)
{
    triple n = cross(tangent(t), normal(t, theta));
    return circle(center(t, theta), radius, normal=n);
}

// Parametrized cycloidal surface
triple paramCycloid(real t, real theta)
{
    return center(t, theta) - sin(t) * tangent(t) - cos(t) * normal(t, theta);
}

triple paramCycloid(pair z)
{
    return paramCycloid(xpart(z), ypart(z));
}

triple principalCycloid(real t)
{
    return paramCycloid(t, 0);
}

// Parametrized trochoidal surface
triple paramTrochoid(real t, real theta, real d)
{
    return (1 - d) * center(t, theta) + d * paramCycloid(t, theta);
}

triple paramTrochoid(pair z, real d)
{
    return paramTrochoid(xpart(z), ypart(z), d);
}

triple cameraDirection(triple pt, projection p=currentprojection)
{
    if (p.infinity) {
        return unit(p.camera);
    } else {
        return unit(p.camera - pt);
    }
}

triple towardCamera(triple pt, real distance=1, projection p=currentprojection)
{
  return pt + distance * cameraDirection(pt, p);
}

////////////////////////////////////
/////////// Start drawing /////////
////////////////////////////////////
// t range
real tMin = 0;
real tMax = 2pi;

// real lateralAngle = pi / 8;
real lateralAngle(real t)
{
    return pi / 4;
    // return 2t;
}

pen extraThin = linewidth(.2pt);
pen thin = linewidth(.4pt);
pen thick = linewidth(1pt);
pen extraThick = linewidth(2pt);

path3 baseCurve = graph(base, tMin, tMax, operator ..);
draw(
    baseCurve,
    thin,
    arrow=Arrow3(TeXHead2)
);

real t = (2tMin + tMax) / 3;
// Current point on the base curve
triple curr = base(t);
dot(curr, extraThick, L=Label("$\vec r(t)$", align=SW));

// Start circle
// triple startCenter = center(tMin, lateralAngle(tMin));
// path3 startCircle = cyCircle(tMin, lateralAngle(tMin));
// dot(base(tMin), thick);
// dot(startCenter, thick);
// draw(startCircle, blue + thin);
// draw(surface(startCircle), blue + opacity(.1), light=nolight);

// triple startInter = paramTrochoid(tMin, lateralAngle(tMin), .5);
// dot(startInter, extraThick);
// draw(startInter -- base(tMin) -- startCenter, thin);

// End circle
// triple endCenter = center(tMax, 0);
// path3 endCircle = cyCircle(tMax, 0);
// dot(base(tMax), thick);
// dot(endCenter, thick);
// draw(endCircle, red + thin);
// draw(surface(endCircle), red + opacity(.1), light=nolight);
//
// triple endInter = paramTrochoid(tMax, 0, .5);
// triple endInter2 = paramTrochoid(tMax, lateralAngle(tMax), .5);
// dot(endInter, extraThick);
// dot(endInter2, extraThick);
// draw(paramTrochoid(tMax, lateralAngle(tMax), .5) -- base(tMax) -- endCenter, thin);

// The TNB frame
draw(
    pos(curr, tangent(t)),
    red + thin,
    arrow=Arrow3(),
    L=Label("$\vec T$", position=EndPoint, align=SW)
);
draw(
    pos(curr, normal(t)),
    green + thin,
    arrow=Arrow3(),
    L=Label("$\vec N$", position=EndPoint, align=NE)
);
draw(
    pos(curr, binormal(t)),
    blue + thin,
    arrow=Arrow3(),
    L=Label("$\vec B$", position=EndPoint, align=W)
);

// main red circle (principal)
triple redCenter = center(t, 0);
// path3 redCircle = cyCircle(t, 0);
// dot(redCenter, thick);
// draw(redCircle, red + thin);
// draw(surface(redCircle), red + opacity(.1), light=nolight);
//
// triple redInter = paramTrochoid(t, 0, .5);
// dot(redInter, extraThick);
// draw(redCenter -- redInter, thin, arrow=Arrow3());

// path3 redRollDir =
//     arc(curr, curr + (redCenter - curr) / 4, curr + (blueCenter  - curr) / 4);

// main blue circle
triple blueCenter = center(t, lateralAngle(t));
path3 blueCircle = cyCircle(t, lateralAngle(t));
dot(blueCenter, thick, L=Label("$O_\theta$", align=.3N + E));
draw(blueCircle, red + thin);
draw(surface(blueCircle), red + opacity(.1), light=nolight);

triple blueInter1 = paramTrochoid(t, lateralAngle(t), .5);
dot(blueInter1, extraThick, L=Label("$T_\theta(t)$", align=NE));
triple blueInter2 = paramCycloid(t, lateralAngle(t));
dot(blueInter2, extraThick, L=Label("$C_\theta(t)$", align=.5NW));
draw(
    blueCenter -- blueInter2,
    thin,
    L=Label("$1 - d$", fontsize(8), align=W, position=(MidPoint + EndPoint) / 2)
);
draw(
    blueCenter -- blueInter2,
    invisible,
    L=Label("$d$", black + fontsize(8), align=.5N + W, position=MidPoint / 2)
);

draw(
    curr -- blueCenter,
    green + thin,
    arrow=Arrow3(),
    // arrow=Arrow3(emissive(darkgray)),
    L=Label("$\vec N_\theta$", position=MidPoint, align=NW)
);

draw(
    blueCenter -- pos(blueCenter, tangent(t)),
    red + thin,
    arrow=Arrow3(),
    L=Label("$\vec T$", position=EndPoint, align=E)
);

// angle symbol
path3 angleArc =
    arc(curr, curr + (redCenter - curr) / 4, curr + (blueCenter  - curr) / 4);
draw(angleArc, thin);
label(
    "$\theta$",
    align=(NE + E) / 2,
    // position=towardCamera(midpoint(angleArc))
    position=midpoint(angleArc)
);

// rolling angle
triple rollingAngle(real t, real theta, real d, real phi)
{
    return center(t, theta)
        + d * cos(phi) * tangent(t) + d * sin(phi) * normal(t, theta);
}

var rollingAngleArc0 = graph(
    new triple (real theta) {
        return rollingAngle(t, lateralAngle(t), 1.2, theta);
    },
    pi / 3,
    pi / 4
);
draw(
    rollingAngleArc0,
    thin,
    arrow=Arrow3(TeXHead2),
    align=W
);

var rollingAngleArc = graph(
    new triple (real theta) {
        return rollingAngle(t, lateralAngle(t), .15, theta);
    },
    -pi / 2,
    -pi / 2 - t
);
draw(
    rollingAngleArc,
    thin,
    L=Label("$-t$"),
    arrow=Arrow3(TeXHead2),
    align=W
);

var rollingAngleArc2 = graph(
    new triple (real theta) {
        return rollingAngle(t, lateralAngle(t), .15, theta);
    },
    0,
    -pi / 2
);
draw(
    rollingAngleArc2,
    thin,
    L=Label("$-\frac\pi 2$"),
    arrow=Arrow3(TeXHead2),
    align=S + .3E
);


// Draw the trochoidal surface
nmesh = 12;
var trochoidalSurface = surface(
    new triple (pair z) { return paramTrochoid(xpart(z), ypart(z), 0.5); },
    // (tMin, -pi / 10), (tMax, lateralAngle(t) + pi / 10), Spline
    (tMin, 0), (tMax, 2pi), Spline
);
var surfacepen = material(
    white+opacity(.1),
    emissivepen=gray(.5)
);
var meshpen = gray + opacity(.1) + extraThin;
draw(
    trochoidalSurface,
    surfacepen=surfacepen,
    meshpen=meshpen
);

// Draw a trochoids
for (real ang : new real[] {0, lateralAngle(t)}) {
    path3 trochoidalCurve = graph(
        new triple (real t) { return paramTrochoid(t, lateralAngle(t), 0.5); },
        tMin, tMax, operator ..
    );

    if (ang == 0) {
        // draw(trochoidalCurve, red + thick);
    } else {
        draw(trochoidalCurve, red + thick);
    }
}

// shipout(scale(4.0) * currentpicture.fit());
```

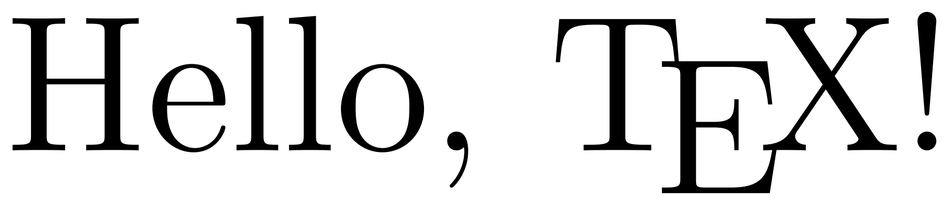
A figure from *Invariance of the Area and Volume of Cycloid Surfaces and Trochoid Surfaces* (Choi, 2022)



http://asymptote.ualberta.ca

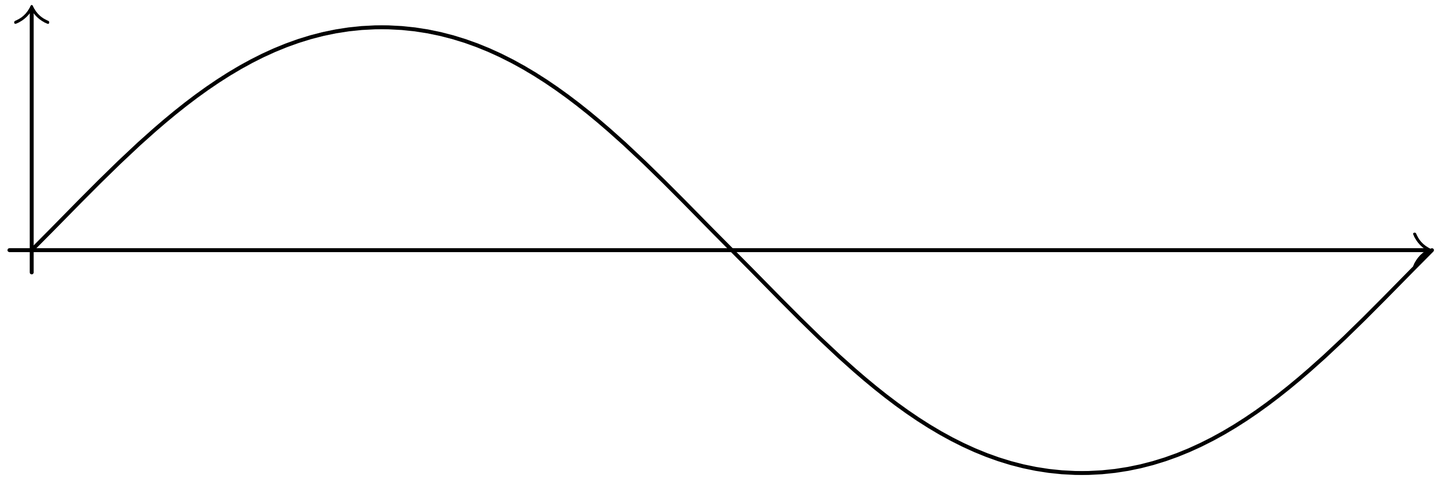
Hello, TeX!

```
// settings.outformat = "png";

// settings.outformat = "pdf";

label("Hello, \TeX!");
```



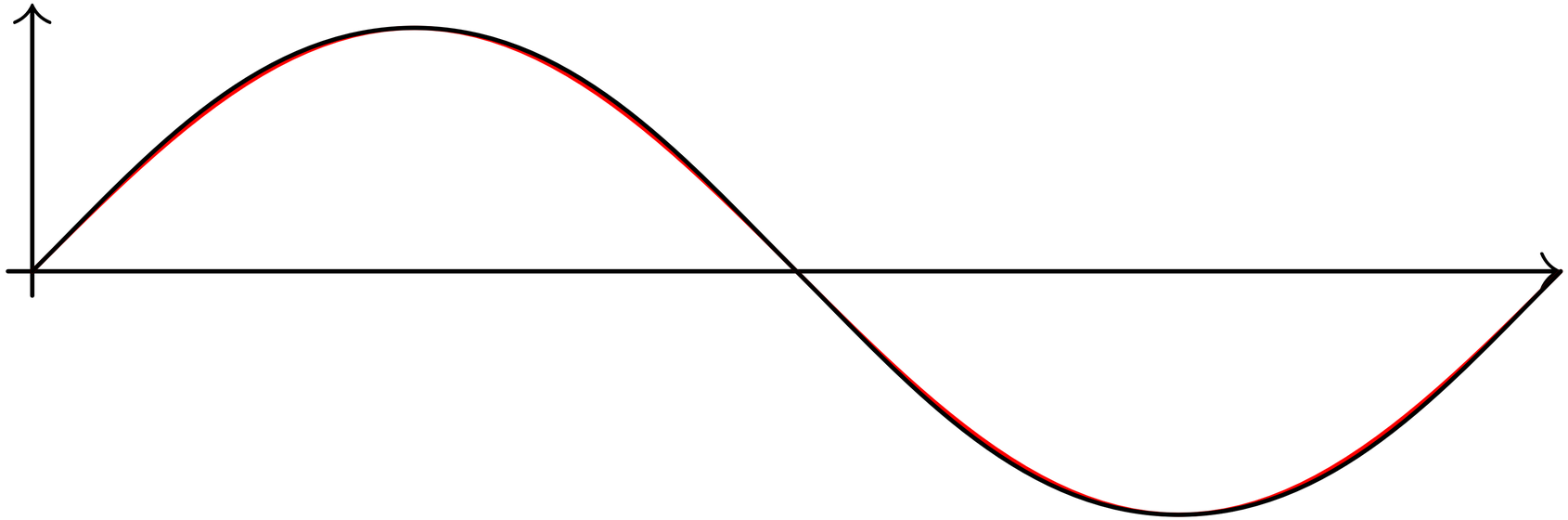
Maybe sine?

```
settings.outformat="pdf";

unitsize(1cm);

draw((-.1, 0) -- (2 * pi, 0), arrow=Arrow(TeXHead));

draw((0, -.1) -- (0, 1.1), arrow = Arrow(TeXHead));

draw(

    (0, 0){(1, 1)}

    .. {right}(pi / 2, 1)

    .. {(1, -1)}(pi, 0)

    .. {right}(3pi / 2, -1)

    .. {(1, 1)}(2pi, 0)

);
```



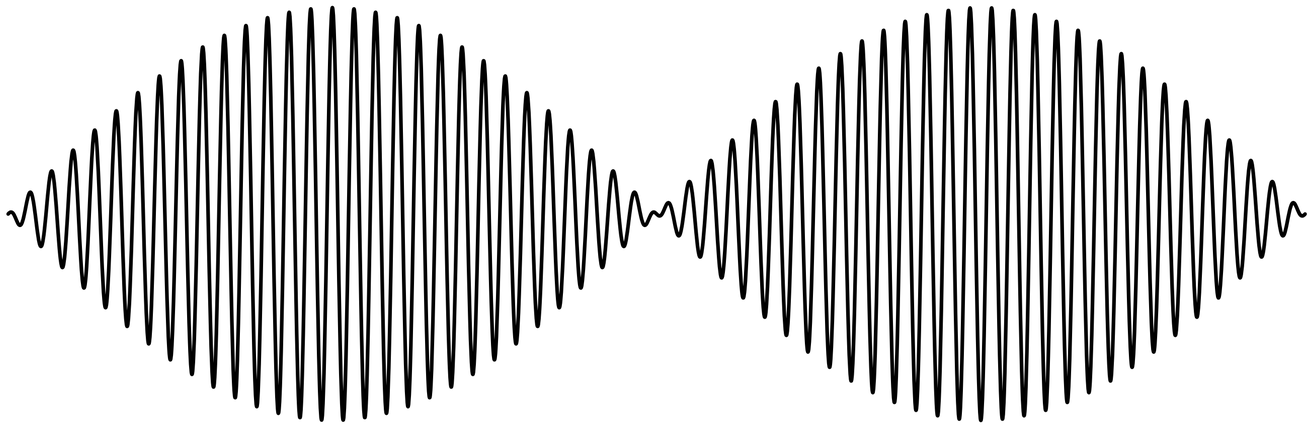
Almost sine!

```
settings.outformat="pdf";

unitsize(1cm);

draw((-.1, 0) -- (2 * pi, 0), arrow=Arrow(TeXHead));

draw((0, -.1) -- (0, 1.1), arrow = Arrow(TeXHead));

import graph;

path s = graph(sin, 0, 2pi);

draw(s, red);

draw(
    (0, 0){(1, 1)}
    .. {right}(pi / 2, 1)
    .. {(1, -1)}(pi, 0)
    .. {right}(3pi / 2, -1)
    .. {(1, 1)}(2pi, 0)
);
```



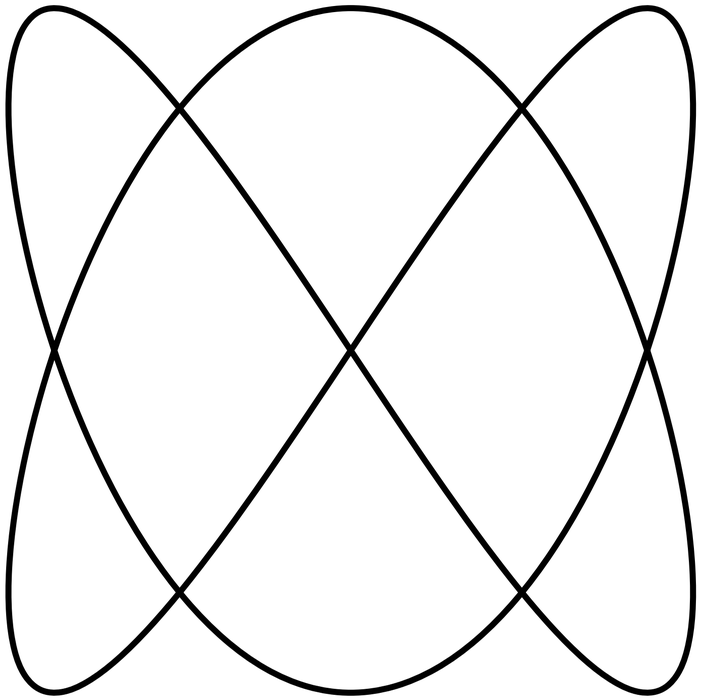
More curves--Defining functions

```
settings.outformat="pdf";

unitsize(1cm);

import graph;

real f(real x) {

    return sin(x) * cos(60x);

}

path s = graph(f, 0, 2pi, n=500, Hermite); // better than
operator..

draw(s);
```



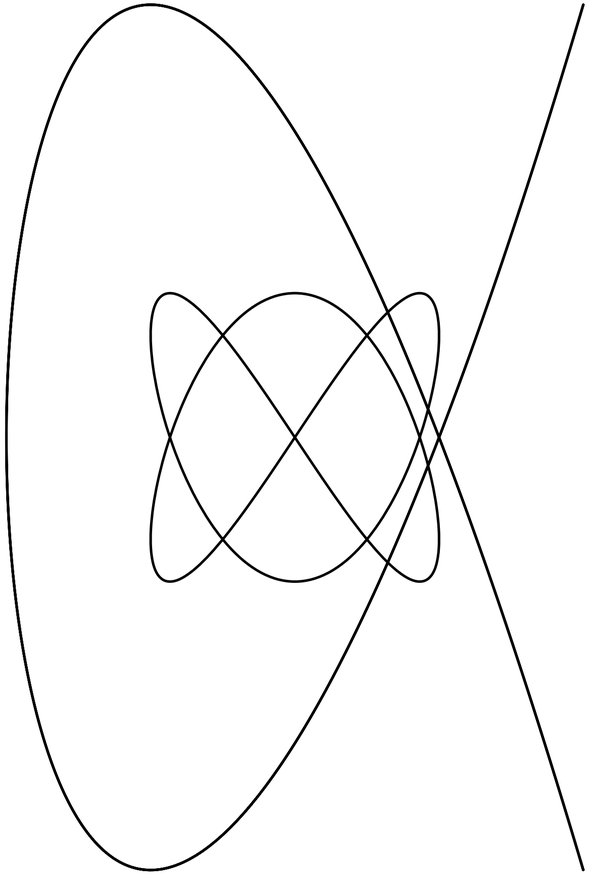
More curves--Parametric curves

```
settings.outformat="pdf";

unitsize(1cm);

import graph;

pair f(real t) {

    return (sin(2t), sin(3t));

}

path s = graph(f, 0, 2pi, operator..);

draw(s);
```



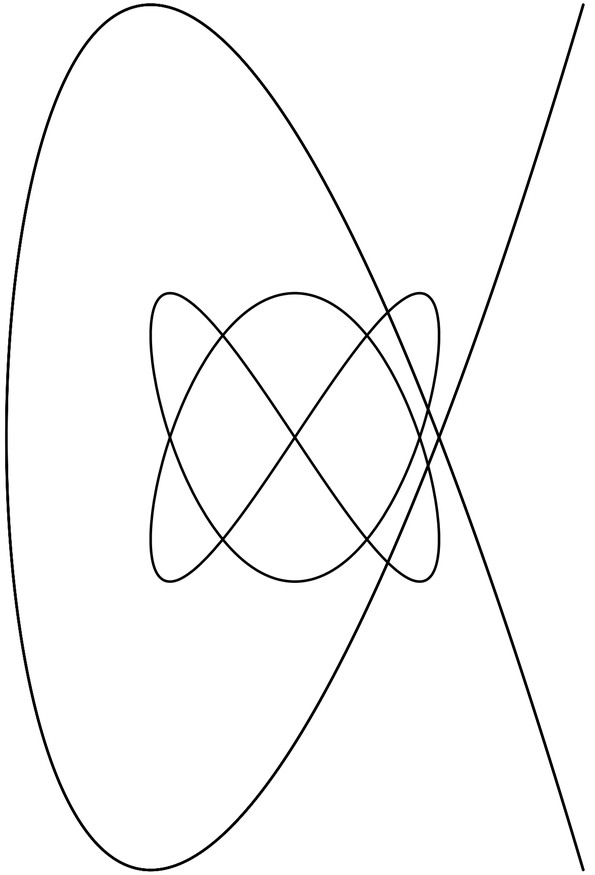
More curves--Some math

```
settings.outformat="pdf";

unitsize(1cm);

import graph;

// Global constant representing an infinitesimal.

real EPSILON = .0001;

// Returns the derivative f'(t) of a parametrized function f: R -> R^2.

pair derivative(pair f(real), real t, real dx=EPSILON) {

    return (f(t + dx) - f(t - dx)) / 2dx;

}

pair f(real t) {

    return (sin(2t), sin(3t));

}

pair df(real t) {

    return derivative(f, t);

}

draw(graph(f, 0, 2pi, operator..));

draw(graph(df, 0, 2pi, n=500, operator..));
```



Anonymous function

```
settings.outformat="pdf";

unitsize(1cm);

import graph;

// Global constant representing an infinitesimal.

real EPSILON = .0001;

// Returns the derivative f'(t) of a parametrized function f: R -> R^2.

pair derivative(pair f(real), real t, real dx=EPSILON) {

    return (f(t + dx) - f(t - dx)) / 2dx;

}

pair f(real t) {

    return (sin(2t), sin(3t));

}

draw(graph(f, 0, 2pi, operator..));

draw(graph(new pair (real t) { return derivative(f, t); }, 0, 2pi, n=500, operator..));
```

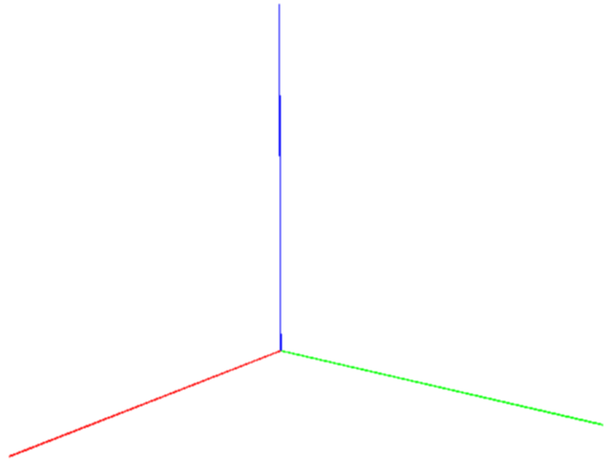


3D Sphere

```
settings.outformat = "pdf";

settings.prc = false;

size(5cm, 0);

import three;

draw(unitsphere);
```

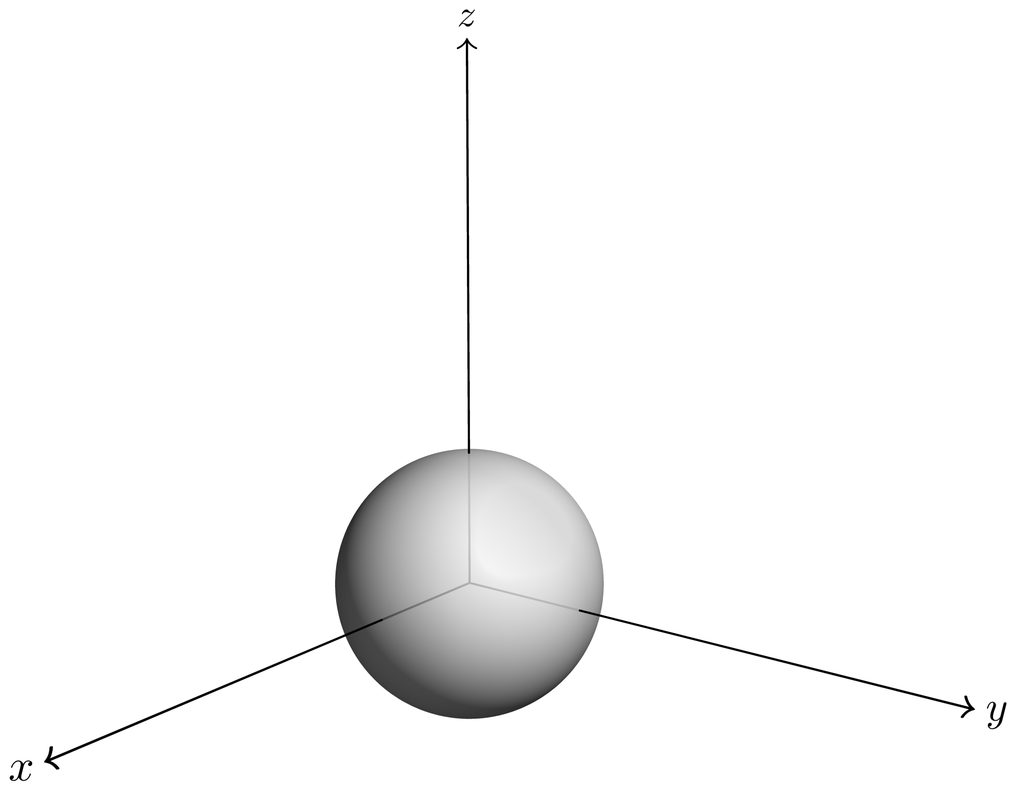
3D Axes

```
settings.outformat = "pdf";

settings.prc = false;

size(5cm,0);

import three;

draw((0, 0, 0)--(2, 0, 0), red);

draw(O--2Y, green);

draw(O--2Z, blue);
```



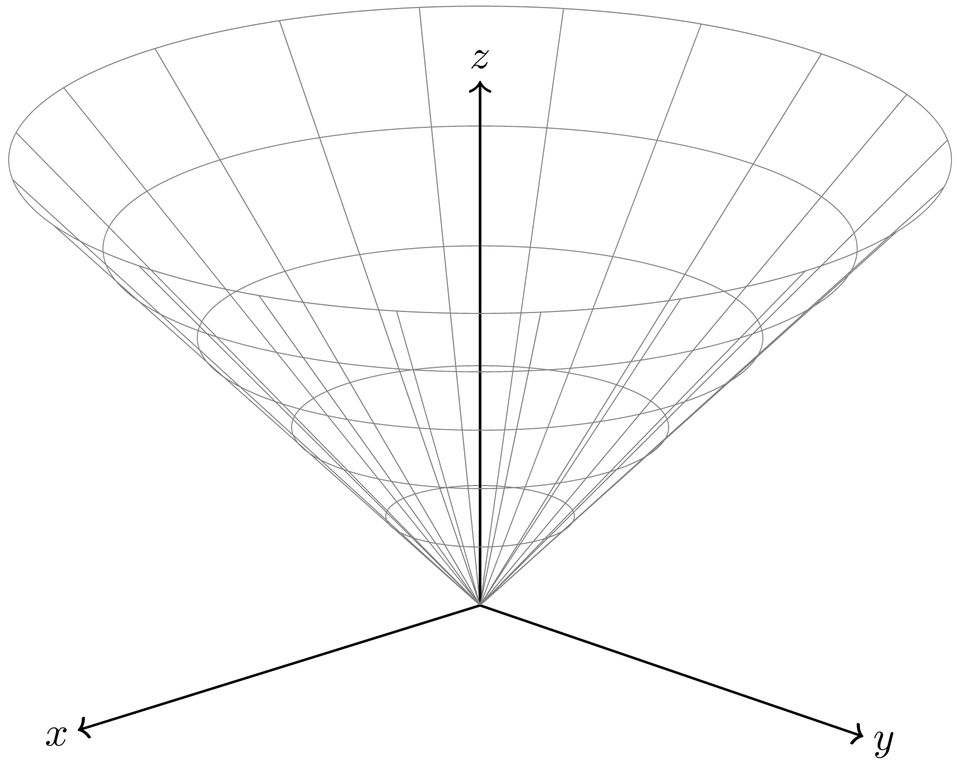
Pens

```
settings.outformat = "pdf";

settings.prc = false;

settings.render = 16;

// Font settings

defaultpen(fontsize(10pt));

import graph3;

size(8cm, 0);

// Pens

pen thin = linewidth(.4pt);

pen medium = linewidth(.6pt);

pen thick = linewidth(1pt);

// Axes

draw(O -- 4X, medium, L=Label("$x$", position=EndPoint), arrow=Arrow3(TeXHead2));

draw(O -- 4Y, medium, L=Label("$y$", position=EndPoint), arrow=Arrow3(TeXHead2));

draw(O -- 4Z, medium, L=Label("$z$", position=EndPoint), arrow=Arrow3(TeXHead2));

draw(unitsphere, 3, 3, surfacepen=material(white + opacity(.8), emissivepen=gray(.1)));
```

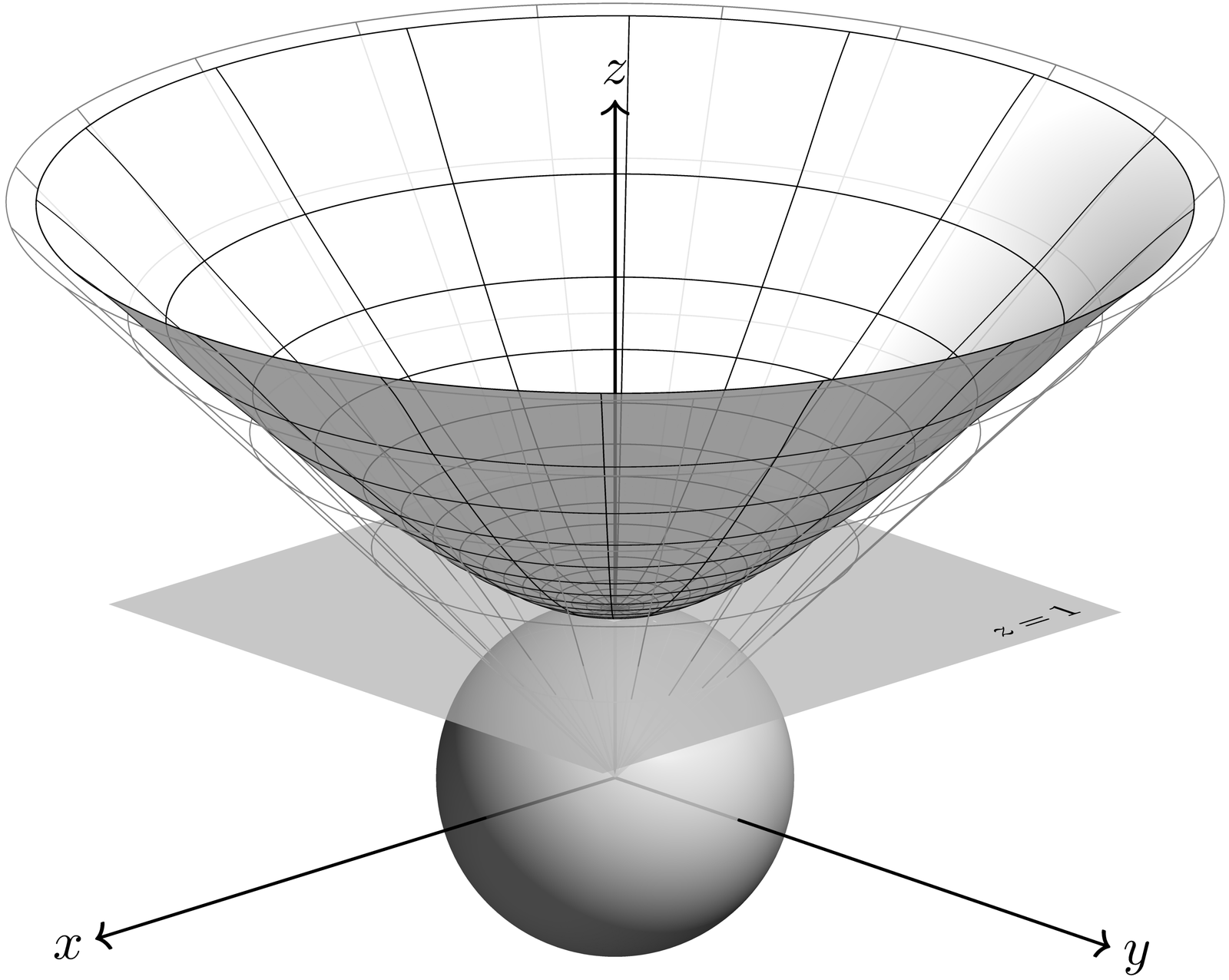


Transformations

```
settings.outformat = "pdf";
settings.prc = false;
settings.render = 16;
// Font settings
defaultpen(fontsize(10pt));
import graph3;
size(8cm, 0);
currentprojection = orthographic(2, 2.1, 1);
pen extraThin = linewidth(.25pt);
pen thin = linewidth(.4pt);
pen medium = linewidth(.6pt);
pen thick = linewidth(1pt);
pen extraThick = linewidth(2pt);
// Axes
draw(O -- 4X, medium, L=Label("$x$", position=EndPoint), arrow=Arrow3(TeXHead2));
draw(O -- 4Y, medium, L=Label("$y$", position=EndPoint), arrow=Arrow3(TeXHead2));
draw(O -- 4Z, medium, L=Label("$z$", position=EndPoint), arrow=Arrow3(TeXHead2));
// Pens
nmesh = 24;
var sub_mesh = gray + extraThin;
draw(scale3(3.4) * rotate(180, X) * shift(-Z) * unitcone, 5, 5, surfacepen=nullpen, meshpen=sub_mesh);
```



Surfaces



See `surfaces.asy`