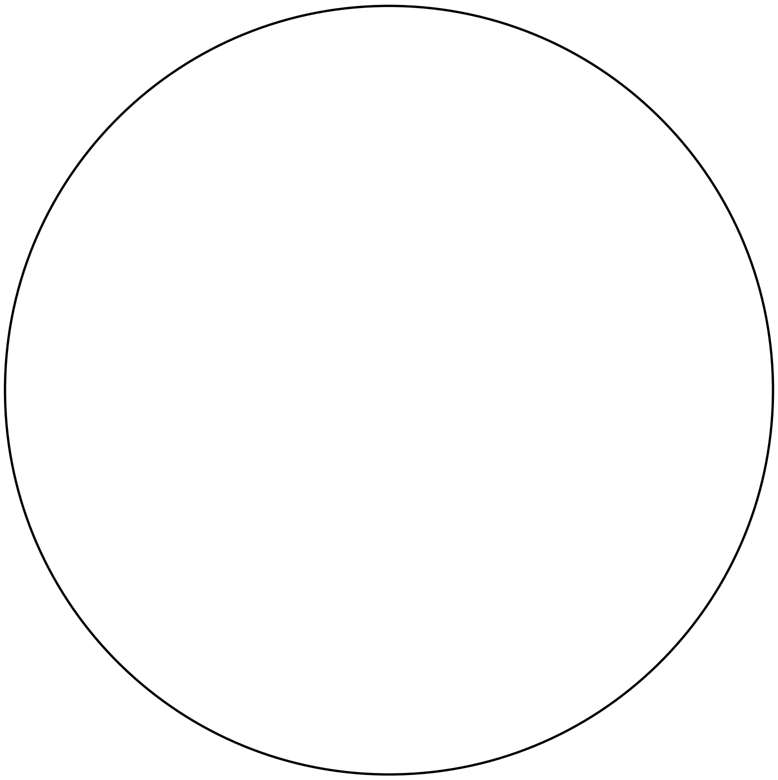


ROUTINE: **rotation-matrix**
ARGUMENTS: A $\mathbf{3}$-vector α and an angle θ
RETURNS: The matrix of the rotation by θ around α

ROUTINE: **vectorscale**
ARGUMENTS: A vector v and a scalar c
RETURNS: cv

ROUTINE: **affineconcat**
ARGUMENTS: Two affine transformations $[M_1\ v_1]$ and $[M_2\ v_2]$
RETURNS: The composition of these, in the same format

The three-dimensional mkpath package

Here are the routines defined in the file **`mkpath3d.inc`**. Roughly speaking, the entire package is concerned with building paths in $3D$.

ROUTINE: **mkpath3d**
ARGUMENTS: $\ell\ a\ f\ t_0\ t_f\ N$
RETURNS: Adds a sequence of Bezier curves to the current path

Here ℓ is the **location matrix**, of the form $[M\ v]$ where M is a 3×3 matrix and v a 3-vector. This is to be interpreted as a rigid affine transformation, describing the shifted position of the path to be drawn. The procedure f is called by name (with `/`). It has two arguments, an array of parameters and a value of t. Output from f is a three-dimensional parametrization in the format

$$[[x(t)\ y(t)\ z(t)][x'(t)\ y'(t)\ z(t)]]$$

ROUTINE: **mkpolypath3d**
ARGUMENTS: $\ell\ a\ f\ t_0\ t_f\ N$
RETURNS: Adds a sequence of line segments to the current path

Is to **`mkpath3d`** as **`mkpolypath`** is to **`mkpath`**.

ROUTINE: **mkpolygon3d**
ARGUMENTS: A location matrix ℓ, an array of $3D$ points p
RETURNS: Adds a sequence of line segments to the current path

The array looks like

$$[[x_0\ y_0\ z_0][x_1\ y_1\ z_1]\ldots[x_n\ y_n\ z_n]]$$

This can be used to draw a cube or faces of some other polyhedron, for example.

ROUTINE: **use-perspective**
ARGUMENTS: None
RETURNS: Sets up use of perspective for rendering $3D$ to $2D$

ROUTINE: **use-projection**
ARGUMENTS: None
RETURNS: Sets up use of projection

Exactly one of these must be used before any three-dimensional drawing is done. If you don't do this, you will get an error message about **`convert`** being undefined.

The mkpath package

Here are the routines defined in the file `mkpath.inc`. Roughly speaking, the entire package is concerned with building paths in $2D$.

ROUTINE: **mkgraph**
ARGUMENTS: $a\ f\ x_0\ x_f\ N$
RETURNS: Adds a sequence of Bezier curves to the current path

The path added has N Bezier curves, along the graph of $y = f(x)$ from x_0 to x_f. The initial array contains parameters passed to the routine f. It is the *name* of the routine f which is passed as an argument (flagged by `/` in `PostScript`). The routine f has a fixed interface: it has two arguments, an array of parameters and a single number x, and returns an array $[f(x)\ f'(x)]$. The array can be empty, but it must be used in the call to `mkgraph`, and it must be removed from the stack in the routine f even if it is not used.

To draw the graph of $y = 2x^4$ from $x = -1$ to $x = 1$ we might have (among other things)

```
/quartic {
3 dict begin
/x exch def
/pars exch def
/c pars 0 def
[
c x mul x mul x mul x mul
2 c mul x mul x mul x mul
]
end
} def

newpath
[2] /quartic -1 1 8 mkgraph
stroke
```

The number N is to be chosen by experience.

ROUTINE: **mkpath**
ARGUMENTS: $a\ f\ t_0\ t_f\ N$
RETURNS: Adds a sequence of Bezier curves to the current path

Here the routine f amounts to a parametrization of a path. It has arguments `[ ... ] t` as for `mkgraph`, and it is to return data of the form

$$[[x(t)\ y(t)][x'(t)\ y'(t)]]$$

This routine has a feature the graphing routine does not: it adds the path constructed to the current path by drawing a line from the current point if it exists to the start of the path being built. In this it acts like the command `arc` in `PostScript`.

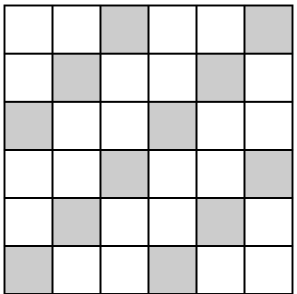
```
/lissajous {
4 dict begin
/t exch 180 mul 3.1416 div def
/pars exch def
/m pars 0 get
/n pars 1 get
[
[t m mul cos t n mul cos]
[t m mul sin neg t n mul sin neg]
```

```
]
end
} def

newpath
[2 3] /lissajous 0 6.28 8 mkpath
stroke
```

ROUTINE: **mkpolypath**
ARGUMENTS: $a\ f\ t_0\ t_f\ N$
RETURNS: Adds a sequence of line segments to the current path

This has exactly the same calling setup as **`mkpath`**, but ignores the derivative in order to build a sequence of straight segments from one position to the next. Its purpose is to help you debug your parametrization routine, to separate the derivative calculation from the position calculation.



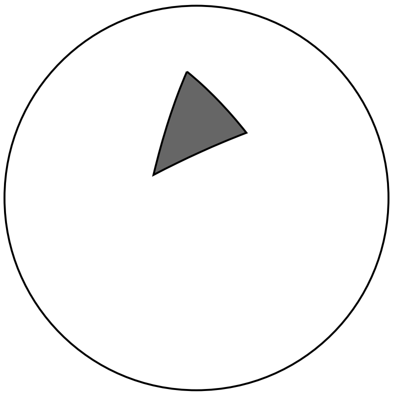
Mathematics 308—Fall 1996

Area on a sphere

These notes are concerned with two results about area on spheres. The first calculates the area of a triangle on the sphere in terms of the sum of its internal angles. The second uses this to show why the angles around a vertex on a convex polyhedron sum to less than 2π.

1. The area of spherical triangles

Suppose we are given a spherical triangle with interior angles α, β, γ. A few examples will suggest that the sum $\alpha + \beta + \gamma$ is always greater than π, which is what it would be if the triangle were in a plane.



In other words, if we call the **defect** of the triangle the quantity

$$D = \alpha + \beta + \gamma - \pi$$

then we expect that *D is greater than 0 for a spherical triangle*. There is in fact a simple interpretation of D which explains this phenomenon. Suppose that the sphere we are looking at has radius 1.

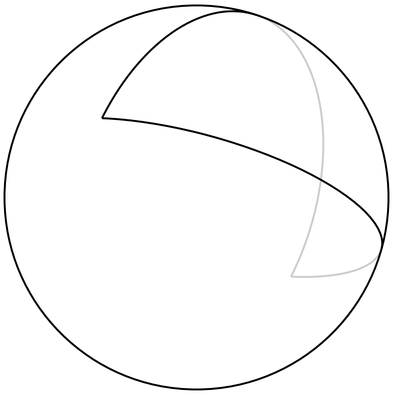
- *The defect of a spherical triangle on the unit sphere is equal to its area.*

This is a simple fact, and the proof is simple, too. The starting point, and in fact the only really tricky item, is that the area of the whole sphere is 4π. This step usually involves a limit process—i.e. calculus—and is in fact much more complicated than the rest of the argument.

The next step is to deduce from this a simple formula for the area of a sector, which is the region on the sphere made up of two great half-circles meeting in two antipodal points—for example the region on the Earth bounded by two longitudes and the two poles. The shape of a sector is completely determined by the interior angle at one end. If this angle is α, then the ratio of the area of the sector to that of the whole sphere is the same as the ratio of α to 2π. In other words, the area of a sector with interior angle α is

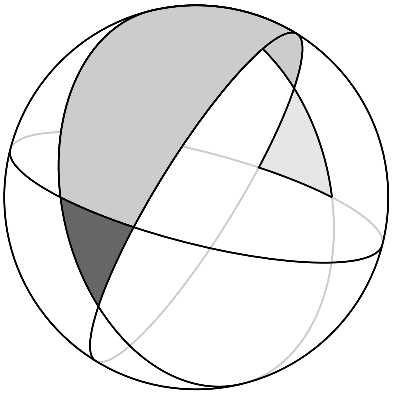
$$4\pi \, \frac{\alpha}{2\pi} = 2\alpha \ .$$

For example, a hemisphere is a sector with angle π, and has area 2π.



Let the triangle be Δ. Each corner of Δ is the interior angle of a sector, which is the union of Δ and a second region. Say the sector for angle α is Δ together with A, etc. Then as far as areas are concerned

$$\begin{aligned} A + \Delta &= 2\alpha \\ B + \Delta &= 2\beta \\ C + \Delta &= 2\gamma \end{aligned}$$



On the sphere, every region Ω has an antipodal region Ω'. This region has exactly the same shape as the original, and in particular the same area. The whole sphere is partitioned into the 8 regions

$$A,\ B,\ C,\ \Delta,\ A',\ B',\ C',\ \Delta'$$

so that in terms of areas

$$A + B + C + \Delta + A' + B' + C' + \Delta' = 4\pi$$

which since $A = A'$ etc means that

$$A + B + C + \Delta = 2\pi\ .$$

This is not quite obvious from the definition of this region, but it can be seen from the fact that Δ, A, B, and C' partition a hemisphere.

If we substitute the equations $A = 2\alpha - \Delta$ we see that

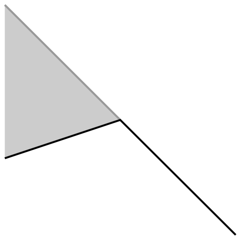
$$(2\alpha - \Delta) + (2\beta - \Delta) + (2\gamma - \Delta) + \Delta = 2(\alpha + \beta + \gamma - \Delta) = 2\pi$$

so that

$$\alpha + \beta + \gamma - \Delta = \pi$$
$$\alpha + \beta + \gamma - \pi = \Delta$$

which is what we wanted to prove.

The exterior angle at a vertex whose interior angle is α is $\pi - \alpha$, or the angle through which one turns in going clockwise around that vertex. It is also the angle through which we turn the normals to the sides when we go around the vertex. On the sphere, each side coming into a vertex lies on a spherical arc. The direction of traversal along the side determines an axis perpendicular to that arc according to the right-hand rule. The exterior angle is also the angle through which the axis turns as we pass from one side of the vertex to the other. We shall use this observation in the next section.



In the plane, when we travel around a triangle the total distance we turn is the sum of the exterior angles we pass and also 2π. In a spherical triangle the sum of the exterior angles is

$$(\pi - \alpha) + (\pi - \beta) + (\pi - \gamma) = 3\pi - (\alpha + \beta + \gamma) = 2\pi - \Delta\ .$$

This formulation remains valid for any simple polygon contained in a hemisphere. This can be seen by chopping the polygon up into triangles.

Exercise 1.1. *What is the correct formula if the sphere has radius R?*

Exercise 1.2. *The reasoning above assumes implicitly that the inside of Δ does not intersect its antipodal region Δ'. Why is that? Exhibit such a region and modify the proof to prove the theorem in that case also.*

2. The defect at the vertex of a convex polyhedron

Suppose that P is the vertex of a convex polyhedron and that touching it are faces $F_1, F_2, \ldots, F_n$ with angles α_i at P.

- *The sum of the α_i is less than 2π.*

This will follow from the result in the previous section! The defect

$$2\pi - \sum \alpha_i$$

will be interpreted as an area on the sphere.

Let ν_i be the unit normal to the face F_i, pointing out from it. The vector ν_i can be considered as a point on the unit sphere. Join each ν_i to ν_{i+1} by an arc on the sphere, with $\nu_{n+1} = \nu_1$. The angle between ν_i and ν_{i+1} is also the spherical distance between them, and the distance one turns the normal ν_i through in moving from F_i around onto F_{i+1}. The axis one makes this turn around is the edge leading out from P separating F_i from F_{i+1}. Note that this edge lies on both faces, and is by definition perpendicular to each of the normal vectors.

If Π is a plane through P with all of the convex polyhedron on one side, the sequence of points ν_i on the sphere will span a polygon contained in the hemisphere cut off by the plane parallel to Π through the origin.

Therefore we can apply the area formula from the previous section: the sum of the exterior angles at the vertices ν_i is less than 2π, and the defect is the area of the spherical polygon spanned by the ν_i. In order to conclude we need to know that the exterior angle at ν_i is the same as the angle α_i. But this follows from the observation that this angle is the angle between the axes defining the sides coming into that vertex.

The sphere package

Here are the routines defined in the file `sphere.inc`. This package is concerned with drawing on spheres.

ROUTINE:	**axis**
ARGUMENTS:	A rotation matrix A
RETURNS:	The axis of rotation

Returns $(Ae_0 - e_0) \times (A_1 - e_1)$, so doesn't work if this is 0, or in particulr if e_0 or e_1 is on the axis.

ROUTINE:	**great-circle**
ARGUMENTS:	$[w_0\ w_1]$ and t
RETURNS:	The point $\gamma(t)$ on the arc from w_0 to w_1.

This is used in `mkpath3d` to parametrize the spherical arc from w_0 to w_1. Assumes w_1 perpendicular to w_0.

ROUTINE:	**sphere-arc**
ARGUMENTS:	L, w_0, w_1, N
RETURNS:	Constructs the spherical path from w_0 to w_1.

I.e. the whole spherical path in N Bezier segments, on the unit sphere transformed by the 3D location matrix L.

ROUTINE:	**horizon**
ARGUMENTS:	axis
RETURNS:	The point where the great circle around the axis goes from visible to invisible.

ROUTINE:	**true-horizon**
ARGUMENTS:	R, w_0, w_1
RETURNS:	The point where the great circle from Rw_0 to Rw_1 from visible to invisible.

Assumes R a rotation matrix, w_0 visible.

ROUTINE:	**spherical-to-rect**
ARGUMENTS:	$[R\ \varphi\ \theta]$
RETURNS:	$[x\ y\ z]$

Converts spherical to rectangular coordinates.