INFOGR – Computer Graphics

J. Bikker - April-July 2015 - Lecture 5: "3D Engine Fundamentals"

Welcome!



efl + refr)) && (depth k HA

survive = SurvivalProbability(diff

at brdfPdf = EvaluateDiffuse(L, N)

efl * E * diffuse;

), N);

MAXDEPTH)

v = true;



Today's Agenda:

- Rendering Overview
- Matrices
- Transforms



```
), N );
efl * E * diffuse;
MAXDEPTH)
survive = SurvivalProbability( diff
adiance = SampleLight( &rand, I, II.
e.x + radiance.y + radiance.z) > 0) MR
v = true;
at brdfPdf = EvaluateDiffuse( L, N ) * Pu
st3 factor = diffuse * INVPI;
st weight = Mis2( directPdf, brdfPdf );
at cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (Fill)
/ive)
ot3 brdf = SampleDiffuse( diffuse, N, r1, r2, R, lp:
rvive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
```

its i (depth : Number it = nt / nt, dde ss2t = 1.0f = nnt set), N); i) st a = nt - nc, b = nt set st Tr = 1 - (R0 + (1 - R0) Fr) R = (0 * nnt - N * (R0) E * diffuse; = true; cfl + refr)) && (depth < NAUDITION N, N); refl * E * diffuse; = true; WXDEPTH) survive = SurvivalProbability(diffuse estimation - doing it properly, involved if; rediance = SampleLight(&rand, I, N, N) ext + radiance x + radiance x > 0) NAUDITION NAUDITION Survive = SurvivalProbability(diffuse ext + radiance x + radiance x > 0) NAUDITION NAUDITION Survive = SurvivalProbability(diffuse ext + radiance x + radiance x > 0) NAUDITION NAUDITION NAUDITION Survive = SurvivalProbability(diffuse ext + radiance x + radiance x > 0) NAUDITION NAUDITION NAUDITION Survive = SurvivalProbability(diffuse ext + radiance x + radiance x > 0) NAUDITION NAUDITI

Topics covered so far:

Lecture 1:

Field study

Lecture 2:

- Rasters
- Vectors
- Color representation

Lecture 3:

2D primitives





Rendering – Functional overview

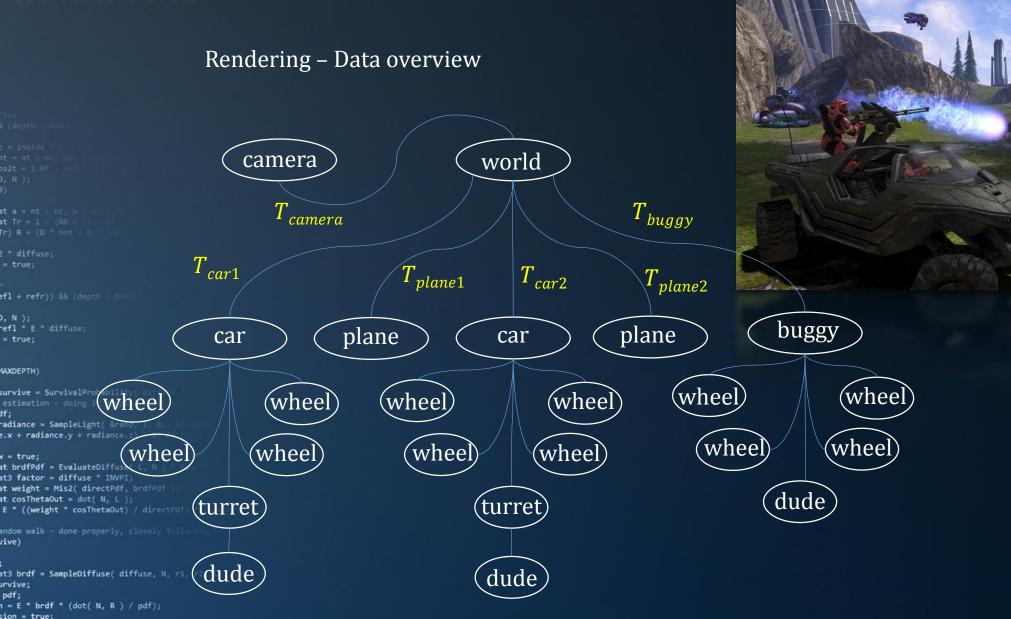
- 1. Transform: translating / rotating / scaling meshes
- 2. Project: calculating 2D screen positions
- 3. Rasterize: determining affected pixels
- 4. Shade: calculate color per affected pixel





efl * E * diffuse;





Rendering – Data overview

Objects are organized in a hierarchy: the *scenegraph*.

In this hierarchy, objects have translations and orientations relative to their parent node.

Relative translations and orientations are specified using matrices.

Mesh vertices are defined in a coordinate system known as *object space*.



E * ((weight * cosThetaOut) / directPdf)
andom walk - done properly, closely folio

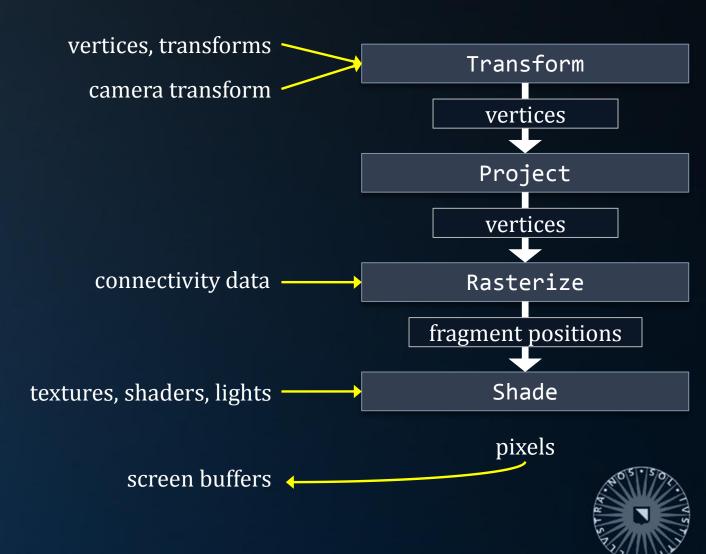
= E * brdf * (dot(N, R) / pdf);

ot3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, N

Rendering – Data overview

Transform takes our meshes from object space (3D) to camera space (3D).

Project takes the vertex data from camera space (3D) to screen space (2D).



), N); efl * E * diffuse; MAXDEPTH) survive = SurvivalProbability(diff adiance = SampleLight(&rand, I. e.x + radiance.y + radiance.z) > 0) at brdfPdf = EvaluateDiffuse(L. N) st weight = Mis2(directPdf, brdfPdf at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) andom walk - done properly, closely fello ot3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, No 1 = E * brdf * (dot(N, R) / pdf);

), N);

= true;

MAXDEPTH)

v = true;

efl * E * diffuse;

adiance = SampleLight(&rand, I.

e.x + radiance.y + radiance.z) >

at brdfPdf = EvaluateDiffuse(L st3 factor = diffuse * INVPI;

at cosThetaOut = dot(N, L);

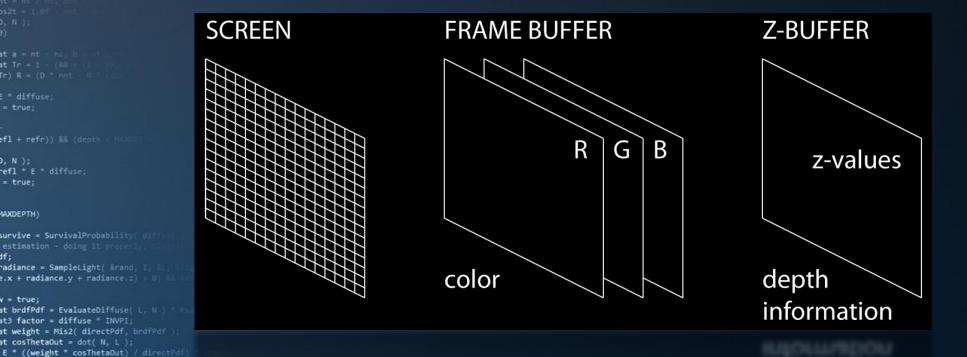
andom walk - done properly, closely fell-

= E * brdf * (dot(N, R) / pdf);

ot3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR)

Rendering – Data overview

The screen is represented by (at least) two buffers:





at3 brdf = SampleDiffuse(diffuse, N, r1, r2, RR, Lpd: urvive;

pdf; n = E * brdf * (dot(N, R) / pdf);

	Rendering – Components	
	Scenegraph	
	Culling	Lecture 7
	Vertex transform pipeline	
	Matrices to convert from one space to another	<i>Lecture 5</i>
·) R = (D * nnt - N * (dd * diffuse; • true;	Perspective	Lecture 6
fl + refr)) && (depth < MADDITION	Rasterization	
N); :fl * E * diffuse:	Interpolation	
true;	Clipping	Lecture 7
AXDEPTH)	Depth sorting: z-buffer	Lecture 7
rvive = SurvivalProbability(diffice estimation - doing it properly, f; diance = SampleLight(&rand, I, &t, &t)	Shading	
x + radiance.y + radiance.z) > 0) %% (contact) = true;	Light / material interaction	<i>P2</i>
<pre>t brdfpdf = EvaluateDiffuse(L, N)</pre>	Shadows / reflections / etc.	<i>P3</i>



Today's Agenda:

- Rendering Overview
- Matrices
- Transforms



```
), N );
efl * E * diffuse;
MAXDEPTH)
survive = SurvivalProbability( diff
adiance = SampleLight( &rand, I, II.
e.x + radiance.y + radiance.z) > 0) MR
v = true;
at brdfPdf = EvaluateDiffuse( L, N ) * Pu
st3 factor = diffuse * INVPI;
st weight = Mis2( directPdf, brdfPdf );
at cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (Fill)
/ive)
ot3 brdf = SampleDiffuse( diffuse, N, r1, r2, R, lp:
rvive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
```

Bases in \mathbb{R}^2 and \mathbb{R}^3

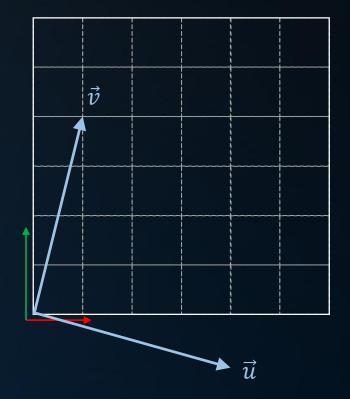
Recall:

- Two linearly independent vectors form a base.
- We can reach any point in using:

$$\vec{a} = \lambda_1 \vec{u} + \lambda_2 \vec{v}$$

- If \vec{u} and \vec{v} are perpendicular unit vectors, the base is orthonormal.
- The Cartesian coordinate system is an example of this, with $\vec{u} = (1,0)$ and $\vec{v} = (0,1)$.

By manipulating \vec{u} and \vec{v} , we can create a 'coordinate system' within a coordinate system.





```
v = true;
v = true;
st brdfPdf = EvaluateDiffuse( L, N ) * Page Sys
st3 factor = diffuse * INVPI;
st weight = Mis2( directPdf, brdfPdf );
st cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (radio
andom walk - done properly, closely following
vive)
;
st3 brdf = SampleDiffuse( diffuse, N, r1, r2, 4R, survive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;
```

), N);

= true;

MAXDEPTH)

efl * E * diffuse;

survive = SurvivalProbability

), N);

= true;

MAXDEPTH)

efl * E * diffuse;

survive = SurvivalProbability(diff

radiance = SampleLight(&rand, I, Al e.x + radiance.y + radiance.z) > 0)

at brdfPdf = EvaluateDiffuse(L, N

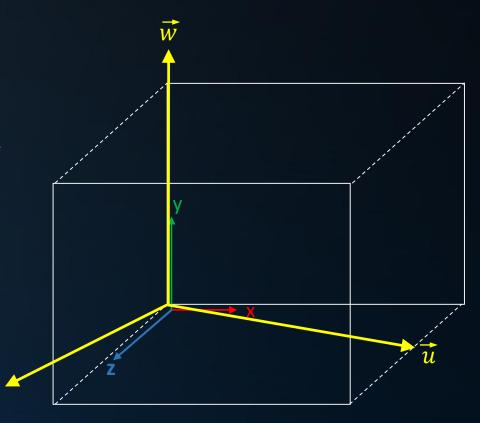
Bases in \mathbb{R}^2 and \mathbb{R}^3

This extends naturally to \mathbb{R}^3 :

Three vectors, \vec{u} , \vec{v} and \vec{w} allow us to reach any point in 3D space;

$$a = \lambda_1 \vec{u} + \lambda_2 \vec{v} + \lambda_3 \vec{w}$$

Again, manipulating \vec{u} , \vec{v} and \vec{w} changes where coordinates specified as $(\lambda_1, \lambda_2, \lambda_3)$ end up.





```
st weight = Mis2( directPdf, brdfPdf );
st cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (radian
sndom walk - done properly, closely fellowing
size)
st3 brdf = SampleDiffuse( diffuse, N, r1, r2, NR, Not
urvive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;
```

Matrices

A vector is an ordered set of *d* scalar values (i.e., a *d*-tuple):

$$\vec{v} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$$
 or (v_1, v_2, v_3) or ...

A $m \times n$ matrix is an array of $m \cdot n$ scalar values, sorted in m rows and n columns:

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

The elements a_{ij} are referred to as the *coefficients* of the matrix (or elements, entries). Note that here i is the row; j is the column.



```
efl * E * diffuse;
adiance = SampleLight( &rand, I.
st3 factor = diffuse * INVPI:
st weight = Mis2( directPdf, brdfPdf
st cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf)
andom walk - done properly, closely fell
ot3 brdf = SampleDiffuse( diffuse, N, r1, r2, ER,
```

1 = E * brdf * (dot(N, R) / pdf);

efl + refr)) && (depth

survive = SurvivalProbability(dif

radiance = SampleLight(&rand, I,
e.x + radiance.y + radiance.z) > 0)

E * ((weight * cosThetaOut) / directPdf andom walk - done properly, closely foll

1 = E * brdf * (dot(N, R) / pdf);

ot3 brdf = SampleDiffuse(diffuse, N, r1, r2, &R, &

st3 factor = diffuse " INVPI; st weight = Mis2(directPdf, brdfPdf st cosThetaOut = dot(N, L);

efl * E * diffuse;

), N);

= true;

MAXDEPTH)

Terminology – special matrices

- A diagonal matrix is a matrix for which all elements a_{ij} are zero if $i \neq j$.
- An *identity matrix* is a diagonal matrix where each element $a_{ii} = 1$.
- The zero matrix contains only zeroes.

$$A = \begin{pmatrix} 1.5 & 0 & 0 \\ 0 & 0.99 & 0 \\ 0 & 0 & 3.14 \end{pmatrix} \qquad A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \qquad A = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Before we continue, what *is* a matrix?

- Just a group of numbers;
- In graphics: often a representation of a coordinate system.



efl + refr)) && (depth K

survive = SurvivalProbability dif

radiance = SampleLight(&rand, I, L e.x + radiance.y + radiance.z) <u>> 0)</u>

at brdfPdf = EvaluateDiffuse(L, N)

efl * E * diffuse;

), N);

MAXDEPTH)

v = true;

Matrices - operations

Matrix addition is defined as:

$$A = B + C$$
, with: $c_{ij} = a_{ij} + b_{ij}$

Note that addition is only defined for matrices with the same dimensions.

Example:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} 2 & 2 \\ 4 & 4 \end{pmatrix} = \begin{pmatrix} 3 & 2 \\ 4 & 5 \end{pmatrix}$$

Subtraction works the same.

```
st3 factor = diffuse * INVPI;
st weight = Mis2( directPdf, brdfPdf );
st cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (redirectPdf) * (redirectP
```



Matrices - operations

Multiplying a matrix with a scalar is defined as follows:

$$A = \lambda B$$
, with: $a_{ij} = \lambda b_{ij}$

Example:

$$2\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

```
survive = SurvivalProbability( different estimation - doing it properly diff;
radiance = SampleLight( &rand, I, M, M) ex.x + radiance.y + radiance.z) > 0 M ex.x + radi
```

), N);

MAXDEPTH)

efl * E * diffuse;



fl + refr)) && (dept)

efl * E * diffuse;

Matrices - operations

Multiplying a matrix (dimensions $m_A \times n_A$) with another matrix (dimensions $m_B \times n_B$):

$$C = AB$$
, with: $c_{ij} = \sum_{k=1}^{n_A} a_{ik} b_{kj}$

Example:

$$\begin{pmatrix} 2 & 6 & 1 \\ 5 & 2 & 4 \end{pmatrix} \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix} = \begin{pmatrix} 17 & 44 \\ 21 & 54 \end{pmatrix}$$

Note the dimensions of the resulting matrix: $m_A \times n_B$.

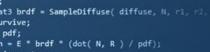
Matrix multiplication is only defined if $n_A = m_B$.

$$c_{11} = \sum_{k=1}^{2} a_{1k} b_{k1} = 2 * 1 + 6 * 2 + 1 * 3 = 17$$

$$c_{21} = \sum_{k=1}^{2} a_{2k} b_{k1} = 5 * 1 + 2 * 2 + 4 * 3 = 21$$

$$c_{12} = \sum_{k=1}^{2} a_{1k} b_{k2} = 2 * 4 + 6 * 5 + 1 * 6 = 44$$

$$c_{22} = \sum_{k=1}^{2} a_{2k} b_{k2} = 5 * 4 + 2 * 5 + 4 * 6 = 54$$



fl + refr)) && (dept

efl * E * diffuse;

survive = SurvivalProbability(dif

adiance = SampleLight(&rand, I

st weight = Mis2(directPdf, brdfPdf st cosThetaOut = dot(N, L);

= E * brdf * (dot(N, R) / pdf);

E * ((weight * cosThetaOut) / directPdf)
andom walk - done properly, closely follo

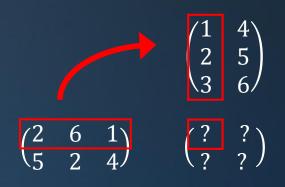
ot3 brdf = SampleDiffuse(diffuse, N, r1, r2, &R, &

= true;

MAXDEPTH)

Matrices - operations

Doing matrix multiplication manually:



Note that each cell in the resulting matrix is essentially the dot product of a row and a column.

Some properties:

Matrix multiplication is distributive over addition:

$$A(B+C) = AB + AC$$
$$(A+B)C = AC + BC$$

...and associative:

$$(AB)C = A(BC)$$

However, matrix multiplication is not commutative, i.e., in general:

$$AB \neq BA$$



efl + refr)) && (depth

survive = SurvivalProbability(diff

adiance = SampleLight(&rand, I. e.x + radiance.y + radiance.z)

at brdfPdf = EvaluateDiffuse(L, N st3 factor = diffuse * INVPI;

andom walk - done properly, closely fell-

1 = E * brdf * (dot(N, R) / pdf);

ot3 brdf = SampleDiffuse(diffuse, N, r1, r2, RR,);

at cosThetaOut = dot(N, L);

efl * E * diffuse;

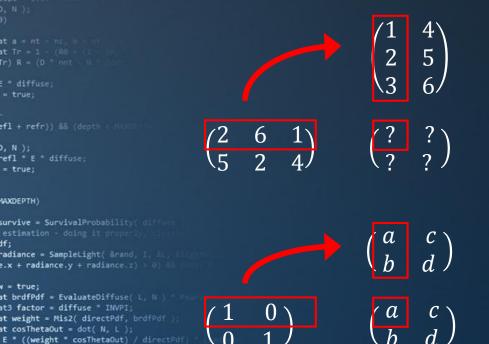
), N);

MAXDEPTH)

v = true;

Matrices - operations

Doing matrix multiplication manually:



Multiplying by the zero matrix yields the zero matrix:

$$0A = A0 = 0$$

Multiplying by the identity matrix yields the original matrix:

$$IA = AI = A$$



efl + refr)) && (depth of

survive = SurvivalProbability dif

radiance = SampleLight(&rand, I, LL. e.x + radiance.y + radiance.z) > 0) |

at brdfPdf = EvaluateDiffuse(L, N)

efl * E * diffuse;

), N);

MAXDEPTH)

v = true;

Matrices - operations

The *transpose* A^T of an $m \times n$ matrix is an $n \times m$ matrix that is obtained by interchanging rows and columns: a_{ij} becomes a_{ji} for all i, j:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \qquad A^T = \begin{pmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{pmatrix}$$

The transpose of the product of two matrices is:

$$(AB)^T = B^T A^T$$



```
st3 factor = diffuse * INVPI;
st weight = Mis2( directPdf, brdfPdf );
st cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (radio of the standown walk - done properly, closely following the standown walk - done properly following the standown walk - done properly followin
```

Matrices - operations

The *inverse* of a matrix *A* is a matrix *A*-1 such that

$$AA^{-1} = A^{-1}A = I$$

Note: only square matrix possibly have an inverse.

```
eff + refr)) && (depth < No.DITTO

(), N );
refl * E * diffuse;
= true;

MAXDEPTH)

survive = SurvivalProbability( diffuse
estimation - doing it properly, involved
estimation - doing it properly, invol
```



), N);

MAXDEPTH)

efl " E " diffuse;

survive = SurvivalProbability diff

Matrices - operations

We can multiply a *d*-dimensional vector by an $m \times d$ matrix:

$$\begin{pmatrix} a_{11} & \cdots & a_{1d} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{md} \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_d \end{pmatrix} = \begin{pmatrix} a_{11}v_1 + \cdots + a_{1d}v_d \\ \cdots & + \cdots + \cdots \\ a_{m1}v_1 + \cdots + a_{md}v_d \end{pmatrix}$$

Example: multiply a 3D vector by a 3x3 matrix:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a_{11}x + a_{12}y + a_{13}z \\ a_{21}x + a_{22}y + a_{23}z \\ a_{31}x + a_{32}y + a_{33}z \end{pmatrix}$$

Note:

This is the same as matrix concatenation; the vector is $simply an m \times 1 matrix$.



efl + refr)) && (depth <

survive = SurvivalProbability(dif

adiance = SampleLight(&rand, I.

st brdfPdf = EvaluateDiffuse(L, N) *
st3 factor = diffuse * INVPI;
st weight = Mis2(directPdf, brdfPdf)
st cosThetaOut = dot(N, L);

1 = E * brdf * (dot(N, R) / pdf);

E * ((weight * cosThetaOut) / directPdf)
andom walk - done properly, closely folio

ot3 brdf = SampleDiffuse(diffuse, N, r1, r2, R,);

efl * E * diffuse;

), N);

MAXDEPTH)

v = true;

Matrices - operations

We can multiply a *d*-dimensional vector by an $m \times d$ matrix:

$$\begin{pmatrix} a_{11} & \cdots & a_{1d} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{md} \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_d \end{pmatrix} = \begin{pmatrix} a_{11}v_1 + \cdots + a_{1d}v_d \\ \cdots + \cdots + \cdots \\ a_{m1}v_1 + \cdots + a_{md}v_d \end{pmatrix}$$

Example: multiply a 3D vector by a 3x3 matrix:

$$\begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} u_x x + v_x y + w_x z \\ u_y x + v_y y + w_y z \\ u_z x + v_z y + w_z z \end{pmatrix} = x\vec{u} + y\vec{v} + z\vec{w}$$

Note:

This is the same as matrix concatenation; the vector is $simply an m \times 1 matrix$.



efl * E * diffuse;

ot3 brdf = SampleDiffuse(diffuse, N, r1, r2, RR)

= E * brdf * (dot(N, R) / pdf);

MAXDEPTH)

Matrices – determinant

The determinant |A| of an $n \times n$ matrix A is the signed area or volume spanned by its column vectors.

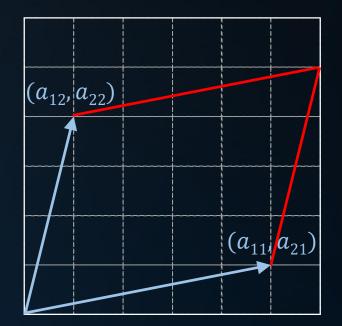
Example (in \mathbb{R}^2):

$$A = \begin{pmatrix} a_{11} \\ a_{21} \end{pmatrix} \begin{vmatrix} a_{12} \\ a_{22} \end{vmatrix} \qquad \det A = |A| = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}$$

In this case, the determinant is the oriented area of the *parallelogram* defined by the two column vectors.

The determinant is positive if the vectors are counterclockwise, or negative if they are clockwise. Therefore:

$$|\det |\overrightarrow{a1} \overrightarrow{a2}| = -\det |\overrightarrow{a2} \overrightarrow{a1}|$$





), N);

MAXDEPTH)

efl * E * diffuse;

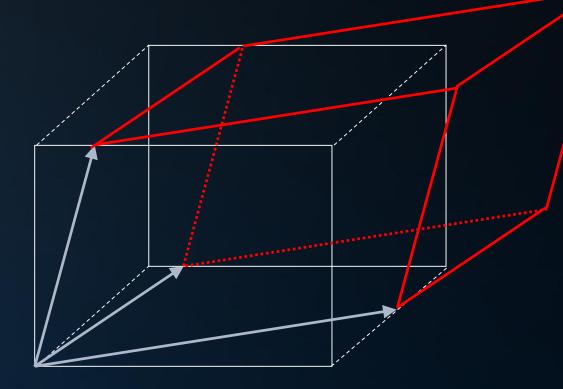
survive = SurvivalProbability dif

radiance = SampleLight(&rand, I, ... e.x + radiance.y + radiance.z) > 0) Matrices – determinant

The determinant |A| of an $n \times n$ matrix A is the signed volume spanned by its column vectors.

In \mathbb{R}^3 , the determinant is the oriented area of the parallelepiped defined by the three column vectors.

$$\det A = |A| = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$$







Matrices – determinant

Calculating determinants: Laplace's expansion.

The determinant of a matrix is the sum of the products of the elements of any row or column of the matrix with their *cofactors*.

The cofactor of an entry a_{ij} in an $n \times n$ matrix A is:

- The determinant of the $(n-1) \times (n-1)$ matrix A',
- that is obtained from A by removing the i-th row and j-th column,
- multiplied by -1^{i+j} .

Example:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

$$\boxed{a_{11}^c = \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} * (-1^2)}$$

$$\boxed{a_{12}^c} = \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} * (-1^3)$$

$$\boxed{a_{13}^c} = \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} * (-1^4)$$

$$|A| = a_{11} a_{11}^c + a_{12} a_{12}^c + a_{13} a_{13}^c$$



survive = SurvivalProbability(diffu estimation - doing it properly ff; radiance = SampleLight(&rand, I, &t e.x + radiance.y + radiance.z) > 0)

), N);

efl * E * diffuse;

v = true;
st brdfPdf = EvaluateDiffuse(L, N) * Po
st3 factor = diffuse * INVPI;
st weight = Mis2(directPdf, brdfPdf);
st cosThetaOut = dot(N, L);
E * ((weight * cosThetaOut) / directPdf)

;
pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, ER, E
prvive;
pdf;
n = E * brdf * (dot(N, R) / pdf);

andom walk - done properly, closely fello

efl + refr)) && (depth

adiance = SampleLight(&rand, I.

E * ((weight * cosThetaOut) / directPdf)

efl * E * diffuse;

Matrices – determinant

Full example for 3×3 matrix:

$$\begin{vmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{vmatrix} = 0 \begin{vmatrix} 4 & 5 \\ 7 & 8 \end{vmatrix} - 1 \begin{vmatrix} 3 & 5 \\ 6 & 8 \end{vmatrix} + 2 \begin{vmatrix} 3 & 4 \\ 6 & 7 \end{vmatrix}$$

$$\begin{vmatrix} 3 & 5 \\ 6 & 8 \end{vmatrix} = 3 * 8 * -1^2 + 5 * 6 * -1^3 = -6$$

$$\begin{vmatrix} 3 & 4 \\ 6 & 7 \end{vmatrix} = 3 * 7 * -1^2 + 4 * 6 * -1^3 = -3$$

$$0 - 1 * - 6 + 2 * - 3 = 0.$$

Generic approach for a for 3×3 matrix:

$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a \begin{vmatrix} e & f \\ h & i \end{vmatrix} - \cdots$$

$$= (aei + bfg + cdh) - (ceg + afh + bdi)$$

Rule of Sarrus for
$$2 \times 2$$
: $\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$



sndom walk - done properly, closely following
//ve)

;

st3 brdf = SampleDiffuse(diffuse, N, r1, r2, iR, ip)
urvive;
pdf;
n = E * brdf * (dot(N, R) / pdf);
sion = true;

efl + refr)) && (depth

survive = SurvivalProbability(diff

radiance = SampleLight(&rand, I, Al e.x + radiance.y + radiance.z) > 0)

efl * E * diffuse;

), N);

Matrices – adjoint

The *adjoint* (or *adjugate*) \tilde{A} of matrix A is the transpose of the cofactor matrix of A.

Example:

$$A = \begin{pmatrix} 2 & 5 \\ 1 & 3 \end{pmatrix} \implies C = \begin{pmatrix} 3 * (-1^2) & 1 * (-1^3) \\ 5 * (-1^3) & 2 * (-1^4) \end{pmatrix} = \begin{pmatrix} 3 & -1 \\ -5 & 2 \end{pmatrix}$$

$$adj(A) = C^T = \begin{pmatrix} 3 & -5 \\ -1 & 2 \end{pmatrix}.$$

The cofactor of an entry a_{ij} in an $n \times n$ matrix A is:

- The determinant of the $(n-1) \times (n-1)$ matrix A',
- that is obtained from *A* by removing the *i*-th row and *j*-th column,
- multiplied by -1^{i+j}.



```
st brdfPdf = EvaluateDiffuse( L, N, ) * Peace to
st3 factor = diffuse * INVPI;
st weight = Mis2( directPdf, brdfPdf );
st cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (Madian
sindom walk - done properly, closely following vive)

st brdf * SampleDiffuse( diffuse, N, r1, r2, RR, t)
strive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;
```

Matrices – inverse

The adjoint is used to calculate the inverse A^{-1} of a matrix A:

$$A^{-1} = \frac{\tilde{A}}{|A|}$$

```
), N );
efl * E * diffuse;
MAXDEPTH)
survive = SurvivalProbability( diff.
adiance = SampleLight( &rand, I. M.
e.x + radiance.y + radiance.z) > 0) [[
v = true;
at brdfPdf = EvaluateDiffuse( L, N ) * F
st3 factor = diffuse * INVPI;
at weight = Mis2( directPdf, brdfPdf );
at cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (Fill
andom walk - done properly, closely fellow
/ive)
ot3 brdf = SampleDiffuse( diffuse, N, r1, r2, R, s
= E * brdf * (dot( N, R ) / pdf);
```



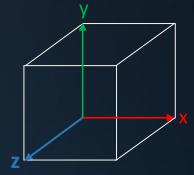
efl * E * diffuse;

radiance = SampleLight(&rand, I, ... e.x + radiance.y + radiance.z) > 0)

andom walk - done properly, closely for

Matrices - overview

$$A = \begin{pmatrix} a_{11} \\ a_{21} \\ a_{31} \end{pmatrix} \begin{bmatrix} a_{12} \\ a_{22} \\ a_{32} \end{bmatrix} \begin{bmatrix} a_{13} \\ a_{23} \\ a_{33} \end{bmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$



 $n \times m$: n rows, m columns

$$\det(A) = |A| = 1 = (aei + bfg + cdh) - (ceg + afh + bdi)$$

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

note:
$$\begin{vmatrix} -1 & 0 \\ 0 & 1 \end{vmatrix} = -1$$
, and: $\det |\overrightarrow{a1} \ \overrightarrow{a2}| = -\det |\overrightarrow{a2} \ \overrightarrow{a1}|$

cofactor
$$a_{11}^c = \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} * (-1^2)$$

Adjoint \tilde{A} of A is C^T ; inverse A^{-1} is $\frac{\tilde{A}}{|A|}$.



; st3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, Npd urvive; pdf; n = E * brdf * (dot(N, R) / pdf);

Today's Agenda:

- Rendering Overview
- Matrices
- Transforms



```
), N );
efl * E * diffuse;
MAXDEPTH)
survive = SurvivalProbability( diff
adiance = SampleLight( &rand, I, II.
e.x + radiance.y + radiance.z) > 0) MR
v = true;
at brdfPdf = EvaluateDiffuse( L, N ) * Pu
st3 factor = diffuse * INVPI;
st weight = Mis2( directPdf, brdfPdf );
at cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (Fill)
/ive)
ot3 brdf = SampleDiffuse( diffuse, N, r1, r2, R, lp:
rvive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
```

efl * E * diffuse;

survive = SurvivalProbability diff

at3 brdf = SampleDiffuse(diffuse, N, r1, r2, U

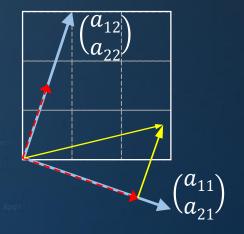
1 = E * brdf * (dot(N, R) / pdf);

Spaces - introduction

As we have seen before, we can multiply a matrix with a vector.

In 2D:
$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_{11}x + a_{12}y \\ a_{21}x + a_{22}y \end{pmatrix}$$
 $= x \begin{pmatrix} a_{11} \\ a_{21} \end{pmatrix} + y \begin{pmatrix} a_{12} \\ a_{22} \end{pmatrix}$

In 3D:
$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a_{11}x + a_{12}y + a_{13}z \\ a_{21}x + a_{22}y + a_{23}z \\ a_{31}x + a_{32}y + a_{33}z \end{pmatrix} = x \begin{pmatrix} a_{11} \\ a_{21} \\ a_{31} \end{pmatrix} + y \begin{pmatrix} a_{12} \\ a_{22} \\ a_{32} \end{pmatrix} + z \begin{pmatrix} a_{13} \\ a_{23} \\ a_{33} \end{pmatrix}$$



Geometric interpretation:

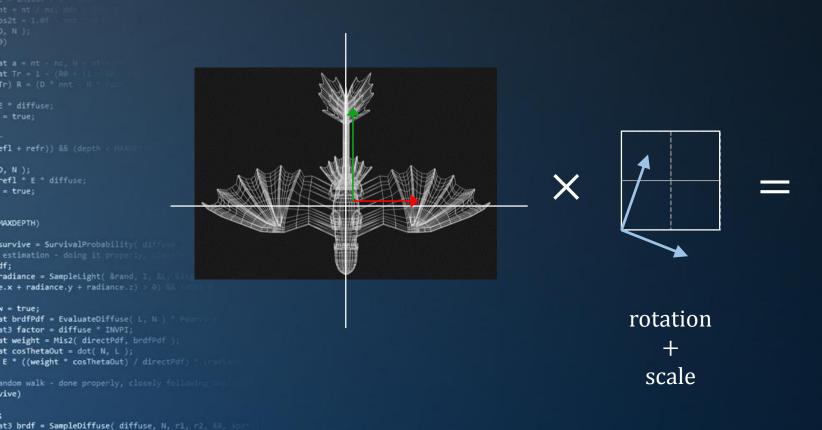
scalar multiplication of $\binom{a_{11}}{a_{21}}$ by x, plus scalar multiplication of $\binom{a_{11}}{a_{22}}$ by y yields *transformed point.*

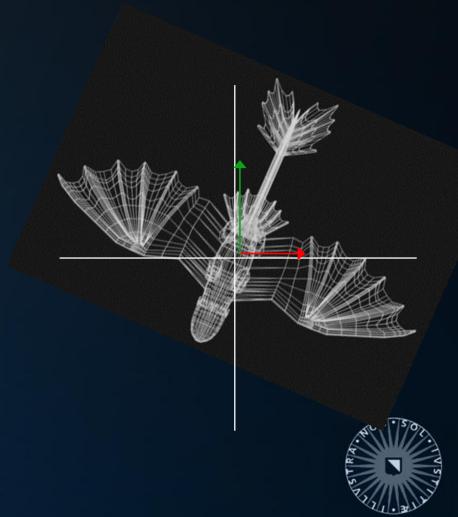


= E * brdf * (dot(N, R) / pdf);

Spaces – introduction

A matrix allows us to *transform* a coordinate system.





Spaces – scaling

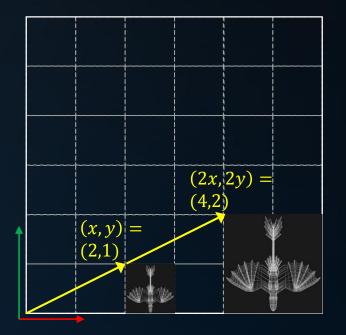
To scale by a factor 2 with respect to the origin, we apply the matrix

$$\begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

Applied to a vector, we get:

$$\begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 2x + 0y \\ 0x + 2y \end{pmatrix} = \begin{pmatrix} 2x \\ 2y \end{pmatrix}$$

This is called *uniform scaling*.





```
st brdfPdf = EvaluateDiffuse( L, N ) * Pourse
st3 factor = diffuse * INVPI;
st weight = Mis2( directPdf, brdfPdf );
st cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (rodling
sindom walk - done properly, closely following
vive)

st3 brdf = SampleDiffuse( diffuse, N, r1, r2, iR, ipper
urvive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
```

), N);

MAXDEPTH)

v = true;

efl * E * diffuse;

survive = SurvivalProbability(diff

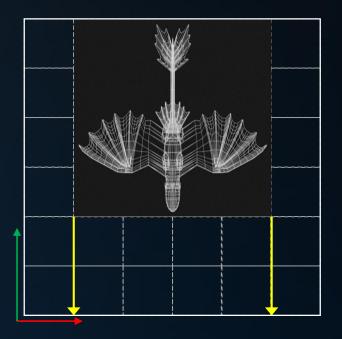
radiance = SampleLight(&rand, I, ... e.x + radiance.y + radiance.z) > 0)

Spaces – projection

If we set one of the a_{ii} to 0, we get an *orthographic projection*.

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

This is useful for projecting a shadow of the dragon on the x-axis, or to draw a 3D object on a 2D screen.



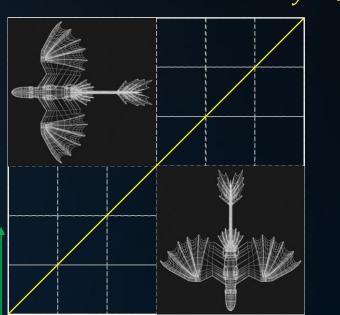


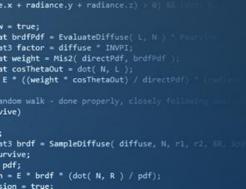
```
), N );
efl * E * diffuse;
= true;
MAXDEPTH)
survive = SurvivalProbability( diff
adiance = SampleLight( &rand, I
e.x + radiance.y + radiance.z) > 0)
v = true;
at brdfPdf = EvaluateDiffuse( L, N )
st3 factor = diffuse * INVPI;
at weight = Mis2( directPdf, brdfPdf )
at cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf)
andom walk - done properly, closely fello-
at3 brdf = SampleDiffuse( diffuse, N, r1, r2, R, N)
1 = E * brdf * (dot( N, R ) / pdf);
```

Spaces – reflection

We can construct a matrix that will swap x and y coordinates to get a reflection in the line y = x:

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0x + 1y \\ 1x + 0y \end{pmatrix} = \begin{pmatrix} y \\ x \end{pmatrix}$$





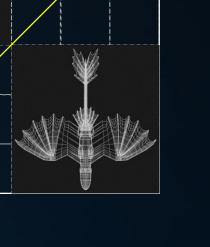
), N);

MAXDEPTH)

efl * E * diffuse;

survive = SurvivalProbability diff

adiance = SampleLight(&rand, I. A.

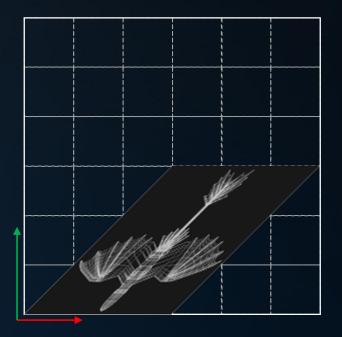


Spaces – shearing

Pushing things sideways:

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1x + 1y \\ 1y \end{pmatrix} = \begin{pmatrix} x + y \\ x \end{pmatrix}$$

This is called *shearing*.





```
), N );
efl * E * diffuse;
MAXDEPTH)
survive = SurvivalProbability diff
adiance = SampleLight( &rand, I. ...
e.x + radiance.y + radiance.z) > 0) [[]
v = true;
at brdfPdf = EvaluateDiffuse( L, N ) * F
st3 factor = diffuse * INVPI;
at weight = Mis2( directPdf, brdfPdf );
at cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (real
at3 brdf = SampleDiffuse( diffuse, N, r1, r2, R, lp:
= E * brdf * (dot( N, R ) / pdf);
```

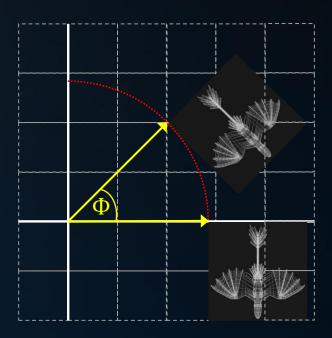
Spaces – rotation

To rotate counter-clockwise about the origin, we use the following matrix:

```
\begin{pmatrix}
\cos \emptyset & -\sin \emptyset \\
\sin \emptyset & \cos \emptyset
\end{pmatrix}
```

For clockwise rotation, we use

```
\begin{pmatrix} \cos \emptyset & \sin \emptyset \\ -\sin \emptyset & \cos \emptyset \end{pmatrix}
```





```
), N );
efl * E * diffuse;
= true;
MAXDEPTH)
survive = SurvivalProbability diff
adiance = SampleLight( &rand, I. M.
e.x + radiance.y + radiance.z) > 0)
v = true;
at brdfPdf = EvaluateDiffuse( L, N )
st3 factor = diffuse * INVPI;
st weight = Mis2( directPdf, brdfPdf )
at cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf)
andom walk - done properly, closely fello-
at3 brdf = SampleDiffuse( diffuse, N, r1, r2, UR, Up
= E * brdf * (dot( N, R ) / pdf);
```

Spaces – linear transformations

A function $T: \mathbb{R}^n \to \mathbb{R}^m$ is called a linear transformation, if it satisfies:

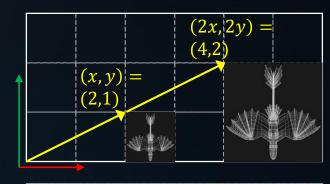
- 1. $T(\vec{u} + \vec{v}) = T(\vec{u}) + T(\vec{v})$ for all \vec{u} , $\vec{v} \in \mathbb{R}^n$.
- 2. $T(c\vec{v}) = cT(\vec{v})$ for all $\vec{v} \in \mathbb{R}^n$ and all scalars c.

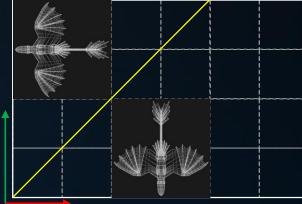
Linear transformations can be represented by matrices.

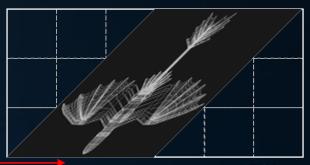
We can summarize both conditions into one equation:

$$T(c_1\vec{u} + c_2\vec{v}) = c_1T(\vec{u}) + c_2T(\vec{v})$$

for all $\vec{u}, \vec{v} \in \mathbb{R}^n$ and all scalars c_1, c_2 .









st3 factor = diffuse * INVPI; st weight = Mis2(directPdf, brdfPdf); st cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf andom walk - done properly, closely folivive)

efl " E " diffuse;

; ot3 brdf = SampleDiffuse(diffuse, N, r1, r2, UR, u prvive; pdf; n = E * brdf * (dot(N, R) / pdf);

Spaces – linear transformations

$$T(c_1\vec{u} + c_2\vec{v}) = c_1T(\vec{u}) + c_2T(\vec{v})$$

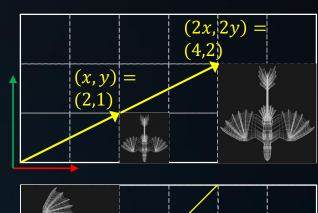
for all $\vec{u}, \vec{v} \in \mathbb{R}^n$ and all scalars c_1, c_2 .

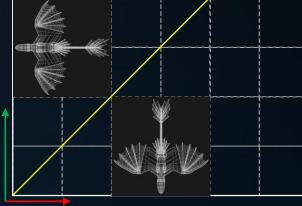
Remember Cartesian coordinates, where each vector \vec{w} can be expressed as a linear combination of base vectors \vec{u} and \vec{v} :

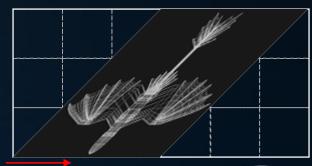
$$\vec{w} = \begin{pmatrix} x \\ y \end{pmatrix} = x \begin{pmatrix} 1 \\ 0 \end{pmatrix} + y \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

If we apply a linear transform T to this vector, we get

$$T\left(\binom{x}{y}\right) = T\left(x\binom{1}{0} + y\binom{0}{1}\right) = xT\left(\binom{1}{0}\right) + yT\left(\binom{0}{1}\right)$$









andom walk - done properly, closely fell

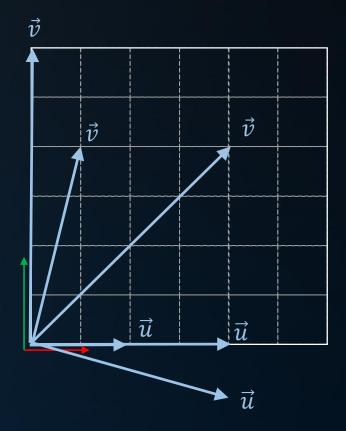
ot3 brdf = SampleDiffuse(diffuse, N, r1, r2, IR. 1 = E * brdf * (dot(N, R) / pdf);

Spaces – linear transformations

$$T(c_1\vec{u} + c_2\vec{v}) = c_1T(\vec{u}) + c_2T(\vec{v})$$

for all $\vec{u}, \vec{v} \in \mathbb{R}^n$ and all scalars c_1, c_2 .

Matrices are constructed conveniently using two base vectors.





```
), N );
efl * E * diffuse;
MAXDEPTH)
survive = SurvivalProbability( diff
adiance = SampleLight( &rand, I. M.
e.x + radiance.y + radiance.z) > 0) 6
v = true;
at brdfPdf = EvaluateDiffuse( L, N )
st3 factor = diffuse * INVPI;
at weight = Mis2( directPdf, brdfPdf )
at cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) *
andom walk - done properly, closely fello-
at3 brdf = SampleDiffuse( diffuse, N, r1, r2, UR, Up
= E * brdf * (dot( N, R ) / pdf);
```

efl + refr)) && (depth

survive = SurvivalProbability dif

adiance = SampleLight(&rand, I.

st weight = Mis2(directPdf, brdfPdf st cosThetaOut = dot(N, L);

1 = E * brdf * (dot(N, R) / pdf);

E * ((weight * cosThetaOut) / directPdf)
andom walk - done properly, closely follo

ot3 brdf = SampleDiffuse(diffuse, N, r1, r2, &R, &

efl * E * diffuse;

), N);

= true;

MAXDEPTH)

Spaces – transforming normals

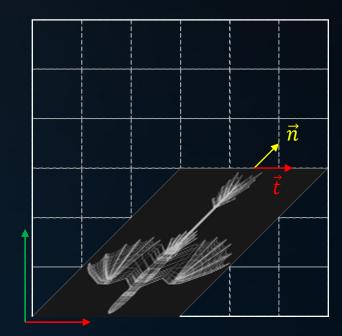
Unfortunately, normals are not always transformed correctly.

To transform a normal vector \vec{n} correctly under a given linear transformation A, we have to apply the matrix

$$(A^{-1})^T$$

Why?

Note: if the transform is orthonormal, $A^{-1} = A^{T}$; therefore $(A^{-1})^{T} = A$.





Spaces – transforming normals

We know that tangent vectors are transformed correctly: $A\vec{t} = \vec{t}_A$. But: $A\vec{n} \neq \vec{n}_A$. Goal: **find a matrix M that transforms** \vec{n} **correctly**, i.e. $M\vec{n} = \vec{n}_M$, where \vec{n}_M is the correct normal of the transformed surface.

Because the original normal vector \vec{n} is perpendicular to the original tangent vector \vec{t} , we know that $\vec{n}^T \vec{t} = 0$. This is the same as $\vec{n}^T I \vec{t} = 0$. Since $I = A^{-1}A$, this is the same as $\vec{n}^T (A^{-1}A) \vec{t} = 0$.

Because $A\vec{t}=t_A$ is the correctly transformed tangent vector, we have $\vec{n}^TA^{-1}\vec{t}_A=0$.

Because their scalar product is 0, $\vec{n}^T A^{-1}$ must be orthogonal to \vec{t}_A . So, the vector we are looking for must be: $\vec{n}_M^T = \vec{n}^T A^{-1}$ (which suggests $M = A^{-1}$).

Because of how matrix multiplication is defined, \vec{n}_M^T and \vec{n}^T are transposed vectors. We can rewrite this to $\vec{n}_M = (\vec{n}^T A^{-1})^T$. And finally, remember that $(AB)^T = B^T A^T$, which gets us $\vec{n}_M = (A^{-1})^T \vec{n}$.

sindom walk - done properly, closely following
//ive)

st3 brdf = SampleDiffuse(diffuse, N, r1, r2, 4R, 1997
urvive;
pdf;
n = E * brdf * (dot(N, R) / pdf);
sion = true:

efl + refr)) && (depth <)

survive = SurvivalProbability diff

adiance = SampleLight(&rand, I.

efl * E * diffuse;

), N);

= true;

MAXDEPTH)

Spaces – needful things

Three things left undiscussed:

- Reverting a transform
- Combining transforms
- Translation

Reverting a transform:

Invert the matrix.

Note: doesn't always work; e.g. the matrix for orthographic projection has no inverse.

Combining transforms:

Use matrix multiplication.

Note: matrix multiplication is not





Spaces – translation

Translation is not a linear transform.

With linear transforms, we get:

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_{11}x + a_{12}y \\ a_{21}x + a_{22}y \end{pmatrix}$$

But we need something like:

$$\binom{x}{y} = \binom{x + x_t}{y + y_t}$$

We can do this with a combination of linear transformations and translations called *affine transformations*.



```
fl + refr)) && (depth
), N );
efl * E * diffuse;
adiance = SampleLight( &rand, I.
e.x + radiance.y + radiance.z) > (
v = true;
at brdfPdf = EvaluateDiffuse( )
st3 factor = diffuse * INVPI;
st weight = Mis2( directPdf, brdfPdf
at cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf)
andom walk - done properly, closely fello
```

ot3 brdf = SampleDiffuse(diffuse, N, r1, r2, &R, &

1 = E * brdf * (dot(N, R) / pdf);

), N);

= true;

MAXDEPTH)

efl * E * diffuse;

survive = SurvivalProbability dif

andom walk - done properly, closely fell

1 = E * brdf * (dot(N, R) / pdf);

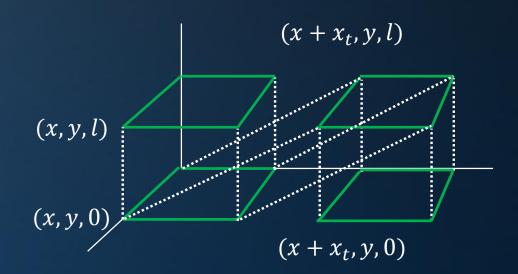
at3 brdf = SampleDiffuse(diffuse, N, r1, r2, in

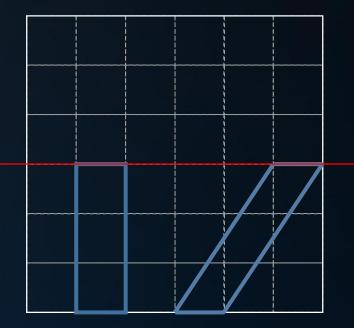
Spaces – translation

Observe: in 2D, shearing "pushes things sideways" (e.g., in the x direction), in a "fixed level" (the y value).

We are thus performing a translation in a 1D subspace (a line), using matrix multiplication in 2D.

In 3D, shearing leads to translation in a 2D subspace, i.e. a plane.







Spaces – translation

By adding a 3rd dimension to 2D space, we can use matrix multiplication to do translation.

$$M\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x + x_t \\ y + y_t \\ z \end{pmatrix}$$

But: what does matrix M look like? What about x_t and y_t ? And how do we deal with the third coordinate z?

```
v = true;
st brdfPdf = EvaluateDiffuse( L, N ) * Promote
st3 factor = diffuse * INVPI;
st weight = Mis2( directPdf, brdfPdf );
st cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (radian
sendom walk - done properly, closely following series)
st3 brdf * SampleDiffuse( diffuse, N, r1, r2, LR to
urvive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;
```

efl + refr)) && (depth

survive = SurvivalProbability diff

radiance = SampleLight(&rand, I, U. e.x + radiance.y + radiance.z) > 0)

efl * E * diffuse;

), N);

= true;

MAXDEPTH)



Spaces – translation

Shearing in 3D based on the z coordinate is a simple generalization of 2D shearing:

$$\begin{pmatrix} 1 & 0 & x_t \\ 0 & 1 & y_t \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1x + zxt + 0z \\ 1y + zyt + 0z \\ 0x + 0y + z \end{pmatrix} = \begin{pmatrix} x + x_t z \\ y + y_t z \\ z \end{pmatrix}$$

The final step is to set z to 1.

$$\begin{pmatrix} 1 & 0 & x_t \\ 0 & 1 & y_t \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x + x_t \\ y + y_t \\ 1 \end{pmatrix}$$

```
st cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (radio of standard of st
```

efl + refr)) && (depth <

e.x + radiance.y + radiance.z) > 0)

st brdfPdf = EvaluateDiffuse(L, N) st3 factor = diffuse = INVPI; st weight = Mis2(directPdf, brdfPdf);

efl * E * diffuse;

), N);

v = true;



efl * E * diffuse;

survive = SurvivalProbability(dif

adiance = SampleLight(&rand, I.

andom walk - done properly, closely fell-

1 = E * brdf * (dot(N, R) / pdf);

ot3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, N

Spaces – translation

Translations in 2D can be represented as shearing in 3D by looking at the plane z=1.

By representing our 2D points (x, y) by 3D vectors (x, y, 1), we can translate them about (x_t, y_t) by applying the following matrix:

$$\begin{pmatrix} 1 & 0 & x_t \\ 0 & 1 & y_t \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x + x_t \\ y + y_t \\ 1 \end{pmatrix}$$

That works for points. What about vectors? We use the following transform:

$$\begin{array}{lll} & \text{ we true;} \\ \text{st brdfPdf = EvaluateDiffuse(L, N) } & \text{ for } \begin{pmatrix} 1 & 0 & x_t \\ 0 & 1 & y_t \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} = \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} \\ \text{ st cosThetaOut = dot(N, L);} \\ \text{ E * ((weight * cosThetaOut) / directPdf)} \end{array}$$



Spaces – translation

Affine transformations (i.e., linear transformations and translations) can be done with matrix multiplication if we add *homogeneous coordinates*, i.e.

- A third coordinate z = 1 to each *point*: $p = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$
- A third coordinate z = 0 to each vector: $\vec{v} = \begin{pmatrix} x \\ y \\ 0 \end{pmatrix}$
- A third row (0 0 1) to each matrix.



```
st weight = Mis2( directPdf, brdfPdf );
st cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (rudion
sndom walk - done properly, closely following
sive)
;
st3 brdf = SampleDiffuse( diffuse, N, r1, r2, RR, rp
urvive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;
```

efl + refr)) && (depth < H

survive = SurvivalProbability(diff

radiance = SampleLight(&rand, I, Al e.x + radiance.y + radiance.z) > 0)

st brdfPdf = EvaluateDiffuse(L, N) st3 factor = diffuse " INVPI:

efl * E * diffuse;

MAXDEPTH)

v = true;

Spaces – translation

These concepts apply naturally to 3D, in which case we again add a homogeneous coordinate, i.e.

- A fourth coordinate w = 1 to each *point;*
- A fourth coordinate w = 0 to each vector;
- A fourth row (0 0 0 1) to each matrix.



efl + refr)) && (depth <

survive = SurvivalProbability(diff

radiance = SampleLight(&rand, I, U. e.x + radiance.y + radiance.z) > 0) |

efl * E * diffuse;

), N);

MAXDEPTH)



Today's Agenda:

- Rendering Overview
- Matrices
- Transforms



```
), N );
efl * E * diffuse;
MAXDEPTH)
survive = SurvivalProbability( diff
adiance = SampleLight( &rand, I, II.
e.x + radiance.y + radiance.z) > 0) MR
v = true;
at brdfPdf = EvaluateDiffuse( L, N ) * Pu
st3 factor = diffuse * INVPI;
st weight = Mis2( directPdf, brdfPdf );
at cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (Fill)
/ive)
ot3 brdf = SampleDiffuse( diffuse, N, r1, r2, R, lp:
rvive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
```

INFOGR – Computer Graphics

J. Bikker - April-July 2015 - Lecture 5: "3D Engine Fundamentals"

END of "3D Engine Fundamentals"

next lecture: "Transformations"

```
st cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (rudle
andom walk - done properly, closely following
vive)
;
st3 brdf = SampleDiffuse( diffuse, N, r1, r2, iR, is
rvive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true;
```

survive = SurvivalProbability diff

radiance = SampleLight(&rand, I, Al e.x + radiance.y + radiance.z) > 0)

ot brdfPdf = EvaluateDiffuse(L, N)

st3 factor = diffuse = INVPI;

st weight = Mis2(directPdf, brdfPdf

v = true;

