(a)

# 1 2015 Paper 4 Question 3

https://www.cl.cam.ac.uk/teaching/exams/pastpapers/y2015p4q3.pdf

Consider the transformations used in the construction and rendering of a three dimensional model on a screen.

(a) List the three principal transformations used in the processing pipeline and explain their roles.

- Translation

  To move the position of objects in the scene. This can be simply to change the location – ie make an object sit on a plane or to move it to the right – or to place a given object either fully or partially behind another object. Having translations in matrices applied to multiple instances of one object can move them to differeent places and make them distinct.

- Scaling

  To change the size of objects in the scene. This makes objects smaller or larger and enables the same base object to be used for different things (ie you can start off with a single instance of a cube, duplicate and resize them and then the cubes can be used completely differently).

- Rotation

  To change the orientation of objects in the scene. Perhaps to attach an object to another or simply to make the scene look better.

(b) Why is it convenient to represent the transformations as matrices?

You can combine multiple operations together by multiplying matrices together. This means that complicated series of transformations can be pre-calculated and then applied to a large number of points without recalculatiion – and since the matrices are precalculated and do not need to be changed during rendering, the multiplication can be performed in parallel. This is very efficient.

Matrices provide a standard format for storage and transformations. Without them there is no easy way to represent transformations, rotations and scalings in the same format. Matrices also standardise transformations across dimensions – there is no constraint on the dimensionality of a matrix or the transformations which it can be applied to. This means that 2D transformations, 3D transformations and beyond can be standardised.

Matrices are a simple comcept to visualise and understand. Understanding how transformations work makes it much easier to actually apply them.

GPU's are optimised for matrix multiplication. This means that performing transformations using matrices results in faster rendering than other methods.

(c) What are homogeneous coordinates? Explain how they are used in modelling these transformations as matrices.

Homogeneous coordinates are 4-dimensional coordinates $(x', y', z', w)$ where $w$ is a scaling factor. This "scaling" factor allows transformations to be expressed as a matrix without preventing translations and scalings.

Homogenous coordinates allow for $\Theta(1)$ time scalings (provided that you are scaling all the coordinates by the same amount) – since you can divide the value of $w$ by a scaling factor and that will scale the coordinates by the same scaling factor.

The conversion from homogeneous coordinates $(x', y', z', w)$ to 3D coordinates $(x, y, z)$ is $x = \frac{x'}{w}$, $y = \frac{y'}{w}$ and $z = \frac{z'}{w}$. The conversion takes $\Theta(1)$ time – however since matrix multiplication is $\Theta(n^3)$ (naïvely) with respect to the dimensionality of a square $n \times n$ matrix, this means that matrix multiplication with homogeneous

coordinates takes approximately twice as long as with standard 3D coordinates. This is generally more efficient, since matrix transformations are calculated once and then applied to many hundreds or thousands of coordinates. It is far more efficient to take the overhead associated with adding an extra dimension in the matrices than to apply for example $M_0$, $T_0$, $M_1$, $T_1$, $M_2$ to 1000 different points (where $M_i$ is a matrix and $T_i$ is a translation).

(d) Derive the matrix to represent a perspective transformation for a viewer at the origin of a point in three dimensions to a point on a screen in the plane $z = d$.

This transformation will require a translation. Translations in matrices require the operations to be done in homogeneous coordinates. The general 3D coordinates of a point on the plane $z = d$ are $(x_0, y_0, d)$. The matrix to do this translation is given below:

$$\begin{pmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

## (b) 2 2010 Paper 4 Question 4

https://www.cl.cam.ac.uk/teaching/exams/pastpapers/y2010p4q4.pdf

(a) Homogeneous coordinates are often used to represent transformations in 3D:

$$\begin{bmatrix} x'_H \\ y'_H \\ z'_H \\ w'_H \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \\ c_1 & c_2 & c_3 & d \end{bmatrix} \begin{bmatrix} x_H \\ y_H \\ z_H \\ w_H \end{bmatrix}$$

(i) Explain how to convert standard 3D coordinates, $(x, y, z)$, to homogeneous coordinates, and how to convert homogeneous coordinates to standard 3D coordinates.

To convert standard 3D coordinates $(x, y, z)$ to homogeneous coordinates $(x', y', z', w)$, create an arbitrary (non-zero) variable $w$ (often initially 1 for convenience). Then define $x' = w \cdot x$, $y' = w \cdot y$ and $z' = w \cdot z$.

To convert homogeneous coordinates $(x', y', z', w)$ to standard 3D coordinates $(x, y, z)$, define $x = \frac{x'}{w}$, $y = \frac{y'}{w}$ and $z = \frac{z'}{w}$.

(ii) Describe the types of transformations provided by each of the four blocks of coefficients in the matrix ($a_{11} \dots a_{33}$ $b_1 \dots b_3, c_1 \dots c_3$ and $d$).

$a$'s provide rotations and scaling (when the scaling factor for $x$, $y$, $z$ is different), $b$'s provide translations, $c$'s provide scalings dependant where the scaling factor is dependant on $(x, y, z)$ and $d$ provides a universal constant scaling factor – ie if you want to scale everything by 2 then you can multiply $w$ by $\frac{1}{2}$.

- a's do rotations and scalings. b's do translations, d does scalings too (but only if you scale all the components by the same amount. c does some sort of scaling dependant on xyz.

(iii) Explain what transformation is produced by each of the following matrices:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & p & -p(1+r) \\ 0 & 1 & q & -q(1+r) \\ 0 & 0 & 1+r & -r(1+r) \\ 0 & 0 & 1 & -r \end{bmatrix}$$

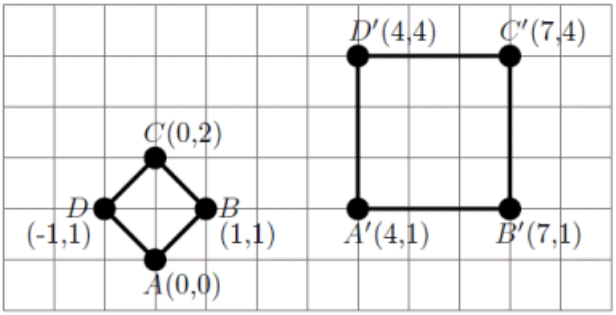
The left matrix scales the coordinates $(x, y, z)$ by a factor of $\frac{1}{z}$ changing them to $(\frac{x}{z}, \frac{y}{z}, 1)$.

Let the right matrix be notated by $M$.

$$
\begin{aligned}
M &= \begin{pmatrix} 1 & 0 & p & -p\cdot(1+r) \\ 0 & 1 & q & -q\cdot(1+r) \\ 0 & 0 & 1+r & -r\cdot(1+r) \\ 0 & 0 & 1 & -r \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{1+r} & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & p & -p\cdot(1+r) \\ 0 & 1 & q & -q\cdot(1+r) \\ 0 & 0 & 1+r & -r\cdot(1+r) \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{1+r} & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & p & 0 \\ 0 & 1 & q & 0 \\ 0 & 0 & 1+r & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & -p \\ 0 & 1 & 0 & -q \\ 0 & 0 & 1 & -r \\ 0 & 0 & 0 & 1 \end{pmatrix}
\end{aligned} \tag{1}
$$

So the matrix $M$ is equal to a scaling of the $z$ coordinate of $\frac{1}{1+r}$. Let $z' = \frac{1}{1+r}$. Then a shear parallel to the $z$ axis. Call these new coordinates $(x'', y'', z'')$. $z'' = \frac{(1+r)\cdot z}{1+r} = z$. Then there is a translation of $(-p, -q, -r)$. So the matrix $M$ represents a scaling proportional to $\frac{1}{1+r}$, then a shear parallel to the $z$ axis and a translation.

(b) Consider the following figure:




(i) Give a matrix, or product of matrices, that will transform the square $ABCD$ into the rectangle $A'B'C'D'$.

To transform the square $ABCD$ into $A'B'C'D'$, we first have to rotate it $\frac{\pi}{4}^c$ clockwise, then scale it by a factor of $\frac{3}{\sqrt{2}}$ and translate it by $(4, 1)$. Since this sequence involves a translation, to have a single matrix to represent this transformation we must use homogeneous coordinates and adding a variable $w$.

Let $T$ be the resultant transformation.

$$
\begin{aligned}
T &= \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{\sqrt{2}}{3} \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 4 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \\
T &= \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & \frac{\sqrt{2}}{3} \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 4 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \\
T &= \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 4 \cdot \frac{\sqrt{2}}{3} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{3} \\ 0 & 0 & \frac{\sqrt{2}}{3} \end{pmatrix}
\end{aligned} \tag{2}
$$

(ii) Show what happens if the same transformation is applied to $A'B'C'D'$.

If we apply the same transformation to $A'B'C'D'$, we do not get the desired transformation since our original transformation rotates and scales around the origin – and $A'B'C'D'$ is not at the origin.

Applying the matrix transformation $T$ to $A'B'C'D'$ gives the coordinates: $A'' = (11.5, -3.5)$, $B'' = (16, -8)$, $C'' = (20.5, -3.5)$ and $D'' = (16, 1)$

(c) Explain what a MIPmap is, how to create one, why one would want to use one, where one would be used, and how one is used.

A MIPmap is a way of storing a texture map where you store $k$ levels of resolution of the image – with the $0^{\text{th}}$ level being the normal resolution and the $(k+1)^{\text{th}}$ level having a quarter the resolution of the $k^{\text{th}}$ level. This allows very quick and accurate downsampling (since the results have essentially been precomputed). It also only takes a third more space that the normal texture map.

Downsampling must happen when the resolution of the texture map for an object at a point is greater than the resolution of the point that the texture map is for. This can either happen at the edge of a shape when the surface is almost parallel to the view vector $v$ or when the texture map is a far higher resolution than the shape it is being mapped onto. In the first case, there are often only a few pixels which need to be downsampled. Although it can be incredibly innefficient to do this (downsampling by a factor of 10 requires sampling 10 pixels with conventional methods – but only two pixels with a MIPmap [technically 1 but more likely 2]), often in the first situation there are not many pixels which have to be downsampled and so this is tolerable. However, in the second situation, every pixel on the object's surface must be downsampled. This means that it takes 10x longer to render the objects texture. Using a MIPmap means that it will only require 2 samples and so is 5x faster in this situation.

You downsample on a MIPmap by finding the value $k = \log_2 D$ where D is the width of the pixel to be downsampled (where the pixel is not square any reasonable method can be taken to estimate it's width – say the arithmetic mean or geometric mean of the sides – or simply the longest side). Then linearly interpolate the texture between the $\lfloor k \rfloor^{\text{th}}$ and $\lceil k \rceil^{\text{th}}$ level of the MIPmap. A quicker (but less accurate) approximation to this would be to take a single sample from the nearest level.