PERSPEX MACHINE II: VISUALISATION

Perspex Machine II: Visualisation

James A.D.W. Anderson*
Computer Science, The University of Reading, England

Abstract

We review the perspex machine and improve it by reducing its halting conditions to one condition. We also introduce a data structure, called the "access column," that can accelerate a wide class of perspex programs. We show how the perspex can be visualised as a tetrahedron, artificial neuron, computer program, and as a geometrical transformation. We discuss the temporal properties of the perspex machine, dissolve the famous time travel paradox, and present a hypothetical time machine. Finally, we discuss some mental properties and show how the perspex machine solves the mind-body problem and, specifically, how it provides one physical explanation for the occurrence of paradigm shifts.

Keywords: perspex machine, artificial neurons, solid modelling, mind-body problem, paradigm shifts, time travel paradox, time machine.

1. Introduction

The perspex machine has a very broad aim. It is intended to solve the mind-body problem and, in particular, to support practical advances in computing, by showing how the perspex can be both a mind and a body. This ambition is set out in the perspex thesis, which presupposes the materialistic thesis that everything that exists is physical: *The perspex machine can model all physical things, including mind, to arbitrary accuracy and, conversely, all physical things, including mind, instantiate a perspex machine.* In this paper we review the development of the perspex machine, make some technical improvements to it, show how it can be visualised, and set out the basis for the bold claims in the perspex thesis. It is intended that future papers will concentrate on technical aspects of the machine – as they affect computer science, AI, philosophy, and physics – without restating the arguments for the thesis. However, at the time of this paper's publication, further discussion of the perspex thesis, and software modelling the perspex machine, will be available via the author's web sites*,[8].

The *perspex machine* performs perspective transformations and was introduced in[5] by unifying the Turing machine with projective geometry. This was done by showing how four, specific, perspective transformations can carry out the four operations of the Unlimited Register Machine (URM), and how these transformations can be laid out in space as the program and registers of the URM. The URM is equivalent to a Turing machine so the perspex machine can carry out all Turing operations. However, the perspex machine can be defined to operate in a continuous space, in which case it can perform operations on general real numbers that are not accessible to a Turing machine. This establishes the theoretical superiority of visualisation over symbolic modes of thought, and establishes vision geometry as the strongest basis for mathematics. It follows that the continuous perspex machine cannot be fully described in words, which is why this paper contains so many figures illustrating the machine's operation, with an invitation, for the reader, to visualise its dynamics.

Whilst[5] gave a clear description and examples of the operation of the perspex machine, the discussion of halting conditions was unsatisfactory. We remedy this here by specifying that the machine halts only when it executes the nullity perspex, H, as defined in this paper. In practical machines it is often convenient to store H, as a constant, at $(0, 0, 0, 0)$ or at the point at nullity, (Φ, Φ, Φ, Φ), to provide a reliable source of halting instructions. The point at nullity was introduced in[2] and formalised in[4], though this formalisation omitted a sign convention that is supplied here, in the section *Erratum*. The number nullity, $\Phi = 0/0$, lies off the real number line. Consequently a real numbered perspex machine that jumps to a point with any co-ordinate nullity leaves Euclidean space, and also the perspective space in which projective geometry takes place, whence it cannot return. Similarly, if a perspex machine jumps to any point with co-ordinate infinity, $\infty = 1/0$,

* J.A.D.W.Anderson@reading.ac.uk, http://www.reading.ac.uk/~sssander
Computer Science, The University of Reading, Reading, Berkshire, England, RG6 6AY.

it cannot return to Euclidean space. In[4] infinity was defined as the point at the positive extreme of the rational number line and is extended here to the positive extreme of the real number line. In projective geometry, mappings within the plane at infinity are permitted so it is reasonable to allow the perspex machine to operate within the plane at infinity and, by analogy, to operate in the nullity subspaces, including the point at nullity. This is discussed in the section *Perspex Instruction*. The transrational numbers, introduced in[4], being the rational numbers, together with infinity and nullity, support total, rational, trigonometric functions. These are useful for describing numerically exact rotations in a digital computer.

The perspex, or perspective simplex, can exist in various physical forms: as a 4×4 matrix with column vectors x, y, z, and t; as a 3D simplex, or tetrahedron, viewed in perspective; as a general linear and perspective transformation; as an artificial neuron; and as a computer instruction. Many of these forms are illustrated in the section *Visualisation,* with the remainder illustrated elsewhere in this paper. A data structure, called the "access column," that can accelerate a wide class of perspex programs is also presented in this section.

Taking the t axis as the time axis leads to an interesting model of spacetime. In this model perspexes can instruct the machine to read and write anywhere in spacetime, but perspexes in canonical form cannot instruct control to jump backwards in time, though some un-normalised perspexes can. Thus, the perspex machine typically undergoes a forward motion in time. In[5] it was proposed that physical time naturally oscillates, corresponding to un-normalised perspexes, but that random events ratchet time into a forward direction, corresponding to normalised perspexes. This gave rise to a proposal to construct a time machine at a microscopic scale[5]. A simpler design is presented here in the section *Time*, along with a resolution of the famous time travel paradox. Dissolving this paradox makes it easier to accept that time machines might be physically possible.

In the section *Paradigm Shifts* it is shown that rational approximations to a continuous function generally oscillate in their accuracy with increasing periods. In other words, rational approximations, and by implication, symbolic systems, such as mathematics and the scientific literature, undergo paradigm shifts. Thus, one cause of the physically mental phenomenon of paradigm shifts is explained by the physical properties of physical numbers affecting a physical machine.

The phenomenon of visual consciousness was defined in terms of mathematical mappings in[1] and was generalised to a perspex definition of consciousness and other mental properties in[3]. Further suggestions for perspex representations of mental properties are given in the section *Mental Properties*.

2. Perspex Instruction

The perspex machine operates in a 4D space, called "perspex" or "program" space[5], that contains perspexes at every point. A perspex is a 4×4 matrix with column vectors x, y, z, and t. The machine executes the perspex at a point as a generic instruction, see also Eqn 5 in[5]:

$$\vec{x}\vec{y} \rightarrow \vec{z}\,;\ \text{jump}(\vec{z}_{11}, t) \qquad \text{(Eqn 1)}$$

This reads the perspexes at locations x and y, multiplies them together and writes the product, reduced to canonical form, into the location z. It then examines the top left element, $\vec{z}_{11}$, of the product and constructs a relative jump from the current location using the components of t. See Eqn 6 in[5]: if $\vec{z}_{11} < 0$ it jumps by t_1 along the x-axis, otherwise if $\vec{z}_{11} = 0$ it jumps by t_2 along the y-axis, otherwise if $\vec{z}_{11} > 0$ it jumps by t_3 along the z-axis. In every case it jumps by t_4 along the t axis. Thus, the machine starts at some point and control jumps from point to point until a halting condition is encountered. We now define that the perspex machine halts only when it executes the halting instruction, H, in (Eqn 3).

We generalise the perspex machine so that it operates on all real numbers, augmented with Φ and ∞. The generalisation from rational numbers to real numbers is obtained by writing an irrational number n as $n/1$ and carrying through the syntactic analysis of transrational numbers in[4]. The set of real numbers, augmented with Φ and ∞, may then be called "transreal" numbers by analogy with the transrational numbers.

We redefine the canonical form of a perspex A, (Eqn 2) in[5], to account for the strictly transreal numbers Φ and ∞:

$$kA \equiv A \text{ with } k = \begin{cases} 1/a_{44}, & a_{44} \neq 0, \Phi, \infty \\ 1, & \text{otherwise} \end{cases} \qquad \text{(Eqn 2)}$$

Hence, a perspex in canonical form kA has element $a_{44} = t_4 \in \{0, 1, \Phi, \infty\}$. All jumps are relative so a machine in Euclidean space stays in Euclidean space when it jumps by a real component, but if it jumps to infinity it cannot jump back to Euclidean space because $\infty + a = \infty$ for all real a and $\infty + \infty = \Phi$ and $\infty + \Phi = \Phi$. Similarly, if it jumps to nullity it cannot jump back to Euclidean space or infinity because $\Phi + a = \Phi$ for all real, infinity, and nullity a. As has been said, projective geometry allows operations on points at infinity so operations at infinity are allowed in the perspex machine. By also allowing operations at nullity we permit a thread of processing to perform housekeeping functions after operating in Euclidean and perspective space and before halting.

We now redefine the universal halting perspex, H, see (Eqn 4) in[5], as:

$$H = \begin{bmatrix} \Phi & \Phi & \Phi & \Phi \\ \Phi & \Phi & \Phi & \Phi \\ \Phi & \Phi & \Phi & \Phi \\ \Phi & \Phi & \Phi & \Phi \end{bmatrix} \qquad \text{(Eqn 3)}$$

H is the default value of every point in perspex space so the machine will execute H when it jumps to any uninstantiated point. This leads to a convenient defensive programming policy. Setting all unwanted jump components to nullity forces the machine to jump into a nullity subspace on a jump error and, if this subspace is uninstantiated, the machine subsequently halts by executing H. Conversely, using 0 as the "don't care" value of a jump invites the bug where the machine cycles infinitely by jumping a zero distance from a point to itself without limit.

Physically random events are, arguably, acausal, but acausal events in Euclidean and perspective space can be modelled by reading or writing to a point with some co-ordinate nullity. Thus, a perspex can appear acausally in Euclidean and perspective space by reading it from nullity, and the history of a perspex in real or perspective space can be saved acausally by writing it to nullity. In other words, physically acausal events can be modelled in Euclidean and perspective space by holding data and/or processing in the nullity subspaces. This modelling is, however, entirely causal within the perspex machine. The perspex instruction (Eqn 1) is the only operation in the perspex machine so it is the machine's causality. Having such a simple causality simplifies discussion of the general irreversibility of time and of time travel within the perspex machine.

3. Visualisation

The perspex can be visualised as an artificial neuron, see Figure 1. The perspexes are stored at locations L_1 and L_2. The large spheres, centred on L_1 and L_2, denote the neurons' bodies. The large disc indicates the boundary between one time on the left and another time on the right. The perspex at L_1, on the left, reads the perspexes at locations $x^{(1)}$ and $y^{(1)}$ into its body. This is shown by the afferent dendrites, mottled cylinders, that run from the neuron's body at L_1 to the synapses at $x^{(1)}$ and $y^{(1)}$. The synapses are shown as small spheres centred on $x^{(1)}$ and $y^{(1)}$. By convention the nullity perspexes, H, at the synapses and elsewhere, are not shown. H is the default value for a point in space and would obliterate the image if it were drawn everywhere. In later figures the synapses are shown as hemisphere capped cylinders. The sphere more distant from the neuron's body is centred on the location of the synaptic perspex. The inner sphere is centred on the axon one neuron body radius toward the body. This has the side effect that it appears as a, roughly, hemispherical synapse on the surface of the synaptic neuron's body, if this is drawn. Having read the perspexes $x^{(1)}$ and $y^{(1)}$ into its body, the neuron then multiplies them together, reduces the result to canonical form, and writes the result to $z^{(1)}$ via the efferent dendrite. The neuron then examines the top-left element of the resultant neuron/perspex and jumps, via a transferent dendrite, to one of $t_1^{(1)}$, $t_2^{(1)}$, $t_3^{(1)}$, or $t_4^{(1)}$.

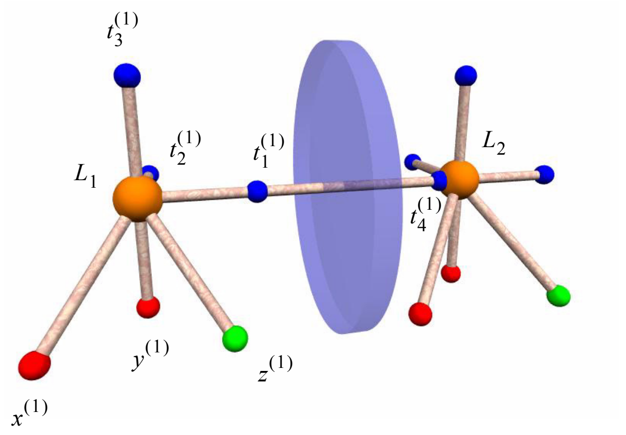

Figure 1: Perspex neuron over time.

Figure 1 was drawn in the ray-tracing package PovRay[6] using hand written code. The later figures showing neurons were generated by a back-end to a rational perspex machine implemented in Pop11[7], though the viewpoint, clipping, or lighting have been adjusted by hand in some cases. All annotation is applied to the figures by hand. Every substantive piece of code demonstrated here will be available via the author's web sites[8] by the time this paper is published.

Visualising perspexes as neurons makes it much easier to visualise the operation of a perspex program, rather than viewing the program, say, as a list of perspex instructions set out in arbitrary order. It is important to recall, however, that the perspex is a perspective transformation[5] so it can be illustrated and analysed as a pencil of rays in projective geometry. One is free to choose a physical instantiation of the perspex that best suits the task at hand.

Figure 2 shows a cube of a standard size and orientation described by perspex neurons. The description uses only the afferent and efferent dendrites, not the transferent ones. Figure 3 shows the neurons in Figure 2 drawn as tetrahedra with the neuron's location and its *x, y,* and *z* vectors as vertices. Figure 4 shows an embryonic neural program that grows into the program in Figure 5 when it is executed. Figure 6 shows a close up of the rotated cube in Figure 5. Figure 7 shows the neurons in Figure 6 drawn as tetrahedra. It can be seen that the program in Figure 5 has rotated the standard cube by comparing Figure 3 with Figure 7. The differences in size amongst the figures is due mainly to a difference in the view point computed by the back-end program, though the view point has been slightly modified by hand in some cases.

Figures 2, 3, 6, and 7 are drawn in a 3D Euclidean space at an instant in time. Figures 4 and 5 are drawn as a contiguous sequence of 3D Euclidean spaces each at successive integral times. Each co-ordinate, c, in 3D space at an instant in time is transformed as $c \to (\mathrm{arctanq}(c))/2$ so as to map the infinite extent of Euclidean space onto a unit cube. The function *arctanq* is a numerically exact, total, transrational version[4] of the standard real function *arctan.* The unit cubes are laid out contiguously on the *x*-axis and are rendered by PovRay. PovRay is a 3D ray tracer.

The program in Figure 4 is arranged as a fibre of neurons running from the top-left to the bottom-right. The first neuron is a variable that holds the current transformation to be applied to a neural model of a standard cube. This variable is initialised to identity. The second neuron is a numerically exact rotation that is used to increment the transformation held in the first neuron. The next four neurons describe a standard cube. The next four neurons are a drawing program that reads the standard cube and writes it into the $t = 1$ hyperplane. The next neuron is the entry point to the fibre. It increments the current transformation, in the first neuron, by the rotation, in the second neuron. It then passes control to the first of four neurons that form a rotation program. The rotation program applies the current transformation, in the first neuron, to the drawing program and writes the result into the growth site. The last neuron in the rotation program passes control to the first neuron in the newly grown drawing program, shown in Figure 5. This drawing program reads the standard cube, applies the current transformation to it, and writes the result as the rotated cube. Thus, the rotation of the cube is obtained by rotating a drawing program and the data it is applied to, *not* by rotating the cube data alone. The dead code in Figure 5 is not drawn, as if it had been explicitly killed by writing H into each unwanted location.

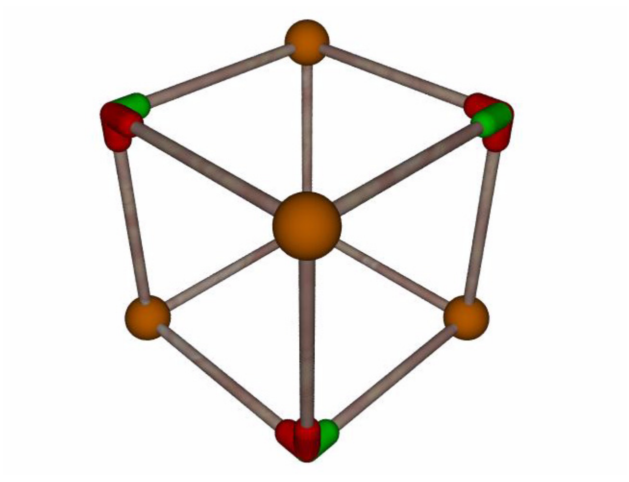

Figure 2: Perspexes as neurons describing a standard cube.

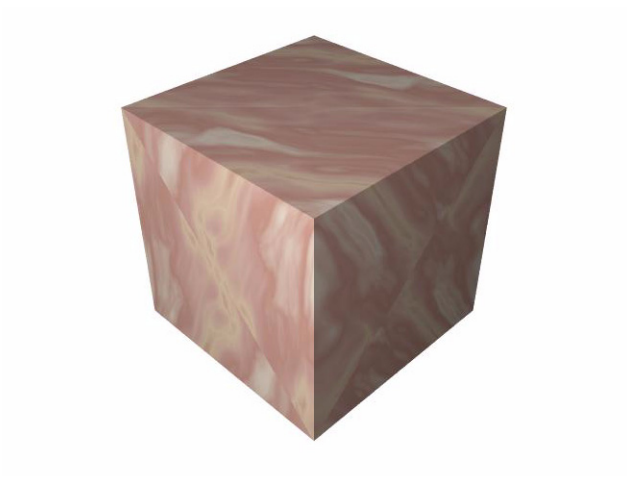

Figure 3: Perspexes as tetrahedra describing the standard cube.

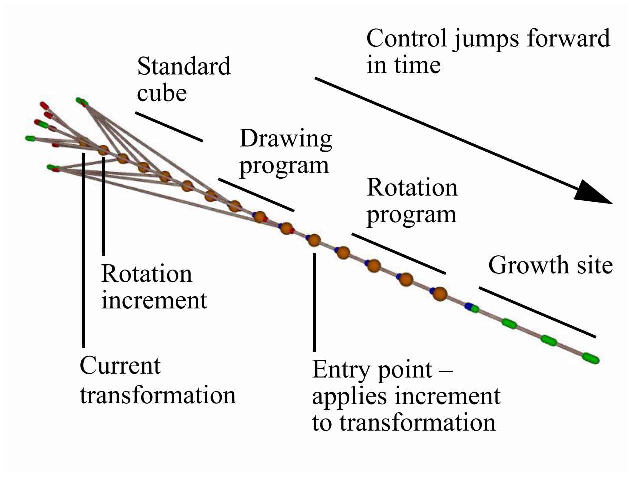

Figure 4: Perspexes as neurons being a self modifying program.

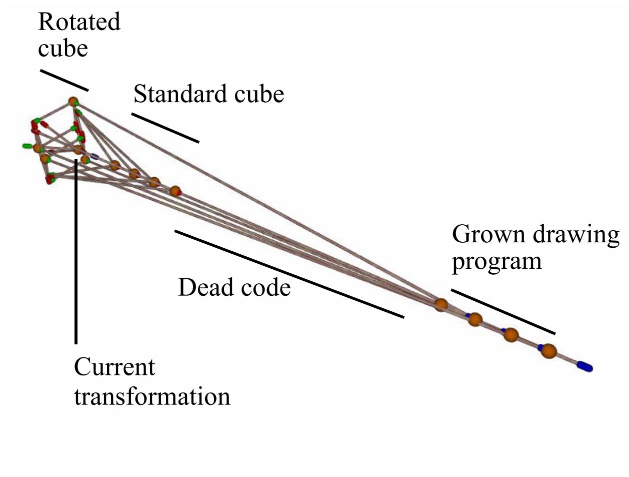

Figure 5: Perspexes as neurons being the modified program.

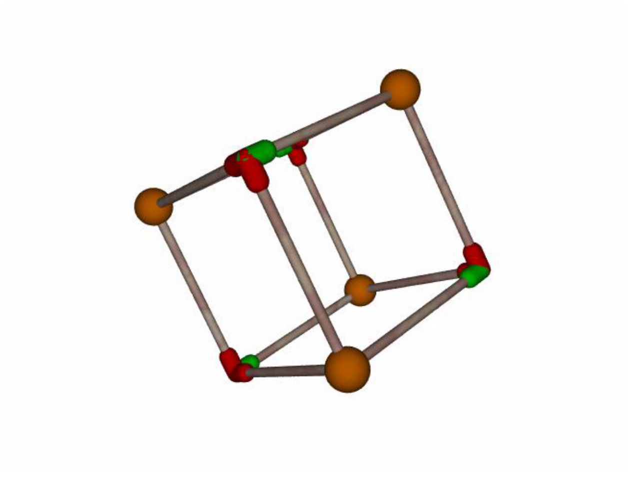

Figure 6: Perspexes as neurons being the rotated cube.

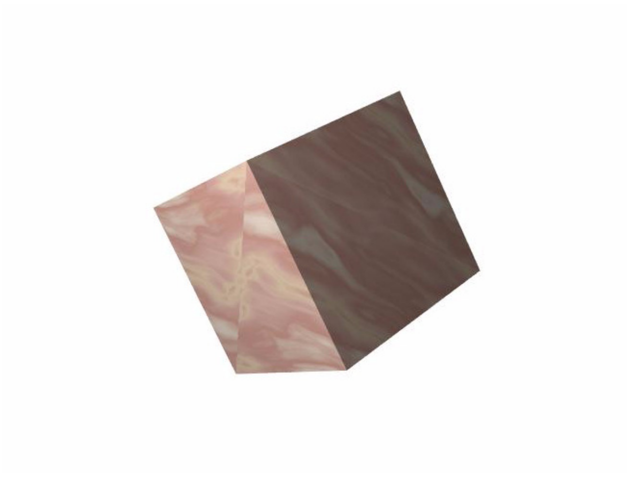

Figure 7: Perspexes as tetrahedra being the rotated cube.

It might seem perverse to implement a drawing program that applies a transformation both to itself and to the data it is applied to. This was done for both a theoretical and a practical reason. It demonstrates the theoretical point that perspex programs are objects like neurons, tetrahedra, or transformations. It makes sense to apply geometrical transformations to each type of object, though the effect is generally homomorphic, not isomorphic, amongst the different instantiations of the perspex. The practical advantage of writing a self-modifying program in this way is its conciseness. There are just 15 neurons shown in Figure 4 and another 5 that act as a bootstrap program to move control from the entry point of perspex space at $\{1, 1, 1, 0\}$ to the entry point of the program at $\{0, 0, 0, 12\}$. These five neurons demonstrate an "operating system" task of moving control from one program to another.

A program made up of 4×4 matrices stored at 4×1 locations can be written as a collection of 4×5 matrices using the following encoding:

$$\begin{bmatrix} x_1 & y_1 & z_1 & t_1 \\ x_2 & y_2 & z_2 & t_2 \\ x_3 & y_3 & z_3 & t_3 \\ x_4 & y_4 & z_x & t_4 \end{bmatrix} \rightarrow \begin{bmatrix} l_1 \\ l_2 \\ l_3 \\ l_4 \end{bmatrix} \quad \text{is encoded as} \quad \begin{bmatrix} x_1 & y_1 & z_1 & t_1 & l_1 \\ x_2 & y_2 & z_2 & t_2 & l_2 \\ x_3 & y_3 & z_3 & t_3 & l_3 \\ x_4 & y_4 & z_x & t_4 & l_4 \end{bmatrix} \qquad \text{(Eqn 4)}$$

Thus, the perspexes shown in Figure 4 are, in order along the fibre, that is, from top-left to bottom-right of the figure:

$$\begin{bmatrix} 1&0&0&0&0 \\ 0&1&0&0&0 \\ 0&0&1&0&0 \\ 0&0&0&1&2 \end{bmatrix}, \begin{bmatrix} \text{cosq}(1/2) & \text{sinq}(1/2) & 0&0&0 \\ \text{-sinq}(1/2) & \text{cosq}(1/2) & 0&0&0 \\ 0&0&1&0&0 \\ 0&0&0&1&3 \end{bmatrix}, \begin{bmatrix} -1&1&1&0&0 \\ -1&1&-1&0&0 \\ -1&-1&1&0&0 \\ 1&1&1&0&4 \end{bmatrix}, \begin{bmatrix} 1&-1&-1&0&0 \\ 1&-1&1&0&0 \\ -1&-1&1&0&0 \\ 1&1&1&0&5 \end{bmatrix}, \begin{bmatrix} -1&1&1&0&0 \\ 1&-1&1&0&0 \\ 1&1&-1&0&0 \\ 1&1&1&0&6 \end{bmatrix}, \begin{bmatrix} 1&-1&-1&0&0 \\ -1&1&-1&0&0 \\ 1&1&-1&0&0 \\ 1&1&1&0&7 \end{bmatrix}, \begin{bmatrix} 0&0&1&0&0 \\ 0&0&-1&0&0 \\ 0&0&-1&0&0 \\ 2&4&1&1&8 \end{bmatrix},$$

$$\begin{bmatrix} 0&0&-1&0&0 \\ 0&0&1&0&0 \\ 0&0&-1&0&0 \\ 2&5&1&1&9 \end{bmatrix}, \begin{bmatrix} 0&0&1&0&0 \\ 0&0&1&0&0 \\ 0&0&1&0&0 \\ 2&6&1&1&10 \end{bmatrix}, \begin{bmatrix} 0&0&-1&0&0 \\ 0&0&-1&0&0 \\ 0&0&1&0&0 \\ 2&7&1&1&11 \end{bmatrix}, \begin{bmatrix} 0&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \\ 3&2&2&1&12 \end{bmatrix}, \begin{bmatrix} 0&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \\ 2&8&17&1&13 \end{bmatrix}, \begin{bmatrix} 0&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \\ 2&9&18&1&14 \end{bmatrix}, \begin{bmatrix} 0&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \\ 2&10&19&1&15 \end{bmatrix}, \begin{bmatrix} 0&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \\ 2&11&20&1&16 \end{bmatrix}$$

And the bootstrap program, which accesses the program root and spine, described below, is entered at $(1, 1, 1, 0)$:

$$\begin{bmatrix} -1&0&0&0&3 \\ 0&-1&0&0&0 \\ 0&0&-1&0&0 \\ 0&0&0&1&0 \end{bmatrix}, \begin{bmatrix} 1&0&2&0&1 \\ 0&1&0&0&1 \\ 0&0&0&1&1 \\ 0&0&0&0&0 \end{bmatrix}, \begin{bmatrix} 1&3&2&-1&1 \\ 0&0&0&0&1 \\ 0&0&0&0&2 \\ 0&0&0&0&0 \end{bmatrix}, \begin{bmatrix} 3&3&2&0&0 \\ 3&3&0&-1&1 \\ 3&3&0&0&2 \\ 0&0&0&0&0 \end{bmatrix}, \begin{bmatrix} 1&0&2&0&0 \\ 0&1&0&0&0 \\ 0&0&0&-2&2 \\ 0&0&0&12&0 \end{bmatrix}$$

A perspex machine may be entered at any point, or any number of points, including all of the points in a segment of the real number line, so the choice to enter the bootstrap program at $(1, 1, 1, 0)$ is arbitrary. This starting point arises from a particular implementation of a perspex machine that is laid out as follows. H is stored at $(0, 0, 0, 0)$. This point is known as the "program seed" and is declared read only in the implementation. The identity perspex is stored at $(1, 0, 0, 0)$, $(0, 1, 0, 0)$, and $(0, 0, 1, 0)$. These locations are called the "program root" and are declared read only. The program root is useful for implementing primitive bootstrap programs, say, by using the identity perspex machine described in[5], or the slightly more sophisticated bootstrap program above. Built-in structures are assigned to the fibre $(k, k, k, 0)$ with integer $k \in Z^+$. This fibre is called the "program spine." The built-in structures handle file input and output, and numerical tasks that would be too tedious to carry out with explicit perspexes, such as the construction and destruction of transrational numbers, respectively, from and to integers. The program spine also contains constant perspexes, such as the zero perspex at $(3, 3, 3, 0)$, and an access column. The perspex machine is entered by the host operating system transferring control to the first perspex, $(1, 1, 1, 0)$, in the spine. No other part of the perspex machine accesses peripheral devices, though executing H passes control back to the host operating system unless this exit is trapped and is used to re-start the perspex machine[5].

The proof[5] unifying projective geometry with the perspex machine relied on operations that change only one element of a homogeneous matrix. Thus, a perspex machine can simulate matrix algebra by using one perspex for every element of a matrix. This is wasteful. It is more efficient to provide a built-in structure that allows the arbitrary accessing of elements within a single perspex so that one perspex can represent up to a 4×4 partition of a matrix. This is what the access column provides. The access column has $2^{16} = 65,536$ perspexes laid out consecutively along the program spine. The first perspex is at a location known as the 0th location and the last perspex is at a location known as the 65,535th location, though the actual locations at which the perspexes are stored in the spine is arbitrary. A sixteen-bit binary number, with bits from the 0th bit to the 15th bit, then encodes both a location within the access column and within the perspex matrix:

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \end{bmatrix} \qquad \text{(Eqn 5)}$$

The access column is used by writing and reading perspexes into and from it. When a perspex p is written into the n'th location, the location number n is examined. If the k'th bit of n is clear then the element from the k'th mask position (Eqn 5) is read from p into the same position in the 0th perspex, otherwise, if the k'th bit of n is set, the element from the k'th mask position is read from p into the same position in the 65,535th perspex. On a read from the n'th perspex the direction of copying is reversed, so that elements are copied from the 0th and the 65,535th perspex into the n'th perspex as the k'th bit of n is, respectively, clear or set. Thus, perspexes can be constructed that contain any combination of the elements from the 0th and the 65,535th perspexes. This provides a very efficient way to access partitions of a perspex. It allows many perspex programs to be accelerated by folding several computations into one.

The access column is of great practical utility, but there does not seem to be any simple way to implement it as a sequence of perspective transformations. Hence, a perspex machine with an access column, or for that matter with read or write only locations, or built-in structures, is cumbersome to analyse using only projective geometry. But this is just to say that access columns, read/write policies, and built-in structures are useful because they perform theoretically cumbersome tasks in a simple way.

This completes the current specification of the perspex machine and describes one particular implementation of a serial perspex machine that contains a program seed, root, and spine. We now consider more abstract properties of the machine.

4. Time

The well known time travel paradox involves a time traveller travelling back in time and killing his grandfather, thereby preventing the time traveller from being born. This paradox is meant to show that time travel is impossible. But, like all logical paradoxes, it relies on a deterministic universe. It is arguable whether quantum events in our universe are deterministically pseudo random or genuinely random. If they are genuinely random they dissolve the time travel paradox, removing this philosophical objection to time travel. It then becomes a matter of considerable scientific interest to discover whether quantum events are genuinely random. One way to try to discover this is to attempt to construct a time machine that exploits genuine randomness in a universe that is hypothesised to contain particles that move arbitrarily in time. Such a machine was proposed in[5] and a simpler machine is proposed below.

It is rather difficult to analyse a paradox that includes terms so complex as a homicidal man and biological reproduction. It is much simpler to consider the transmission of a single bit of information backwards in time, inside a computer, and to measure the consequences of this in terms of a programmed outcome.

Suppose that there is a room with a light in it that is switched on and off by a computer, that the computer is equipped with a light meter, and that the computer has a circuit in it that can send one bit of information backwards in time by sending a single photon backwards in time. Suppose, further, that the computer is programmed to establish a temporal paradox as shown in Figure 8.

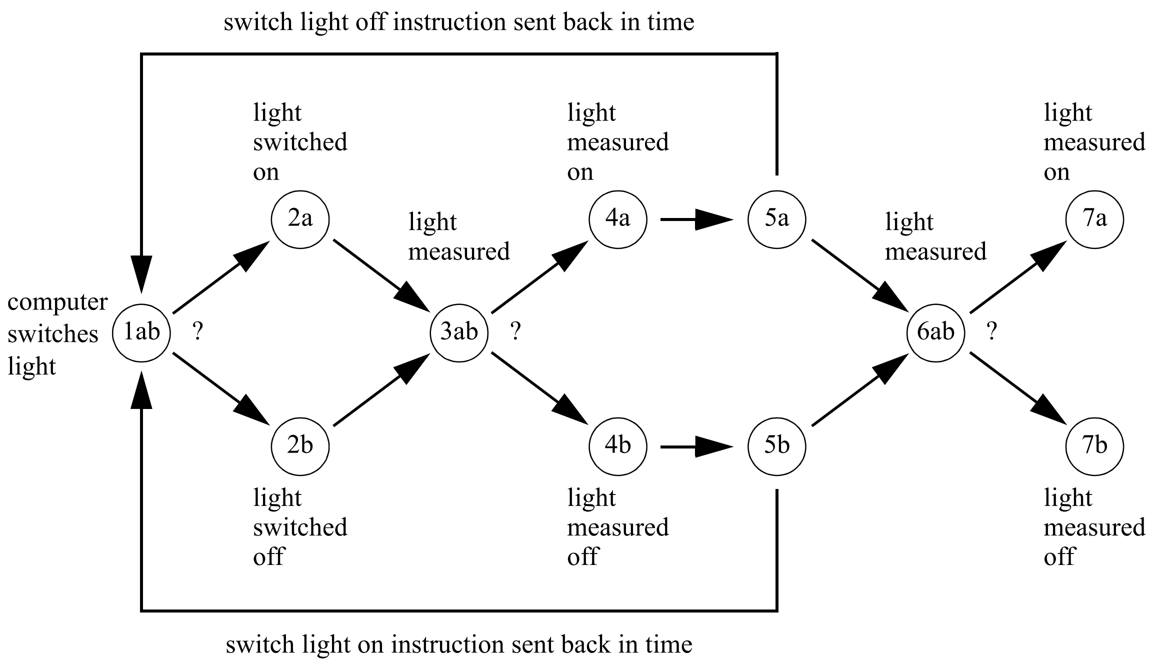

Figure 8: Time travel paradox.

The experiment starts at state *1a* where the computer switches the light on or off. Suppose it switches the light on. The universe, comprising the room, light, light meter, and computer, is in state *2a.* The light in the room is measured at *3a.* If the measurement indicates that the light is on then the universe is in state *4a.* At *5a* a signal is sent back in time instructing the computer at *1b* to turn the light off. If the computer switches the light off then the universe is in state *2b.* The light in the room is measured at *3b.* If the measurement indicates that the light is off then the universe is in state *4b.* At *5b* a signal is sent back in time instructing the computer at *1a* to turn the light on. This establishes the paradoxical time loop *(1a-2a-3a-4a-5a-1b-2b-3b-4b-5b)…* which cycles without limit. Alternatively, if the experiment starts with the light switched off then there is a phase shift in the loop to *(1b-2b-3b-4b-5b-1a-2a-3a-4a-5a)…* One can suppose that the single time travelling photon, carrying one bit of information, is in the superposition of states *1a-2a-3a-4a-5a* and *1b-2b-3b-4b-5b.* In a deterministic universe the photon stays in superposition and cannot escape the region of spacetime between *1ab* and *5ab.* In a quantum universe the superposition of states is unmeasurable so the time travel paradox establishes that no measurable consequence arises from this kind of time travel, in other words, time travel is impossible, in practical terms, as expected from the time travel paradox. But suppose that the universe is non-deterministic, then the superposition of paradoxical states can collapse in several ways. For example, at *1a* the computer switches the light on, then the universe is in state *2a.* The light is measured at *3a* and found to be on at *4a.* The computer then transmits a signal from *5a* to turn the light off at *1b,* but a random signalling error causes it, as before, to enter state *1a* where it instructs the light to be switched on, as before, at *2a.* The light is measured, as before, at *3a,* and found to be on, as before, at *4a.* The signal to switch the light off is transmitted, as before, at *5a.* The part of the computer that is not the time travelling photon measures the light level at *6a* and, we suppose, the light is measured to be on at *7a.* Thus there is no paradox and no superposition of states. So far as measurable spacetime is concerned the history of the universe is *1a-2a-3a-4a-5a-6a-7a.* A similar argument can bring about the history *1b-2b-3b-4b-5b-6b-7b.* Thus, the time travel paradox is resolved if a suitable random event takes place, but a suitable random event does take place in a time loop with a probability tending to one, because[5], if an event is genuinely random it is unaffected by earlier or later outcomes, which is to say that it is acausal and occurs at an instant of time. Hence, on every cycle of a paradoxical time loop a random event gets a chance to occur and, as the loop cycles indefinitely often, the probability of some random event collapsing the superposition of states tends to one. Thus, the time travel paradox hardly ever holds in a non-deterministic universe and cannot be a general prohibition against time travel.

It only remains to claim that time travel in the macro universe of homicidal men results from time travel at the quantum level. One can then construct an argument that some random accident prevented the “time traveller” from being born, or else the time traveller was born, but suffered a random accident preventing him from bringing about his own death. Thus, the time travel paradox is dissolved, as anticipated in[5].

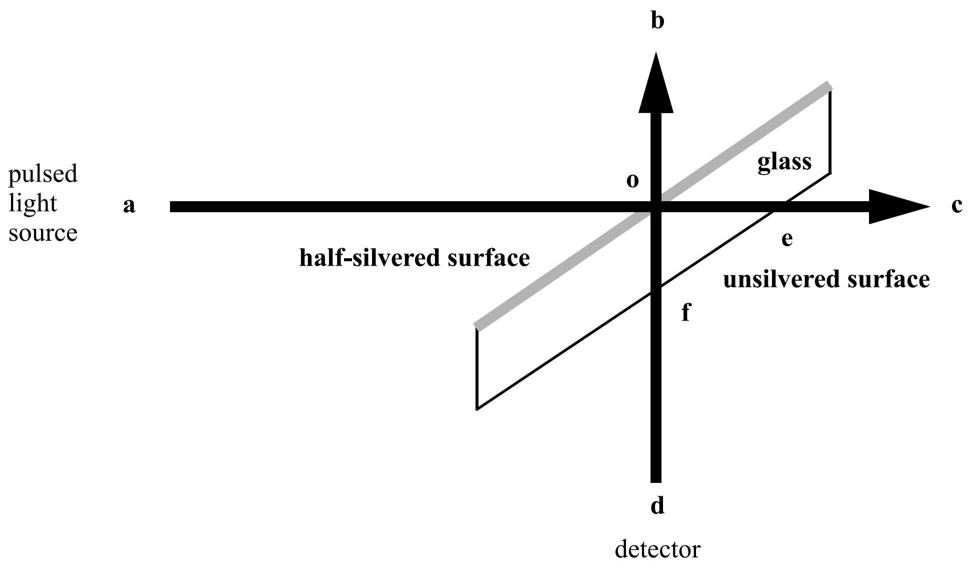

Figure 9: Time machine.

The figure above shows a very simple time machine involving nothing more than: a pulsed light source at *a;* a beam splitter with a glass body that has a half-silvered surface at *o* and an unsilvered surface at *e* and *f;* and a detector at *d.* In forward time light is emitted at *a* and passes straight through the beam splitter to *c* or else is reflected at *o* and travels to *b.* Suppose that light passing from *o* to *e,* in forward time, is retarded by a phase ϕ_1 and a ray of light reflected at *o* from *a* to *b,* in forward time, is phase shifted by ϕ_2, depending on the polarising properties of the surface of the mirror. If time were to reverse in a deterministic universe then photons at *b* and *c* would exactly retrace their paths to *a,* but in a non-deterministic universe some of the photons at *b* will pass through the beam splitter to *d,* and some of the photons at *c* will reflect at *o* and go to *d.* Thus, in a deterministic universe there are no photons at *d,* but in a non-deterministic universe there are photons at *d* that have undergone a systematic phase shift of $-\phi_1 \pm \phi_2$, relative to the phase along the ray in the vacuum, as calculated next.

Step	Effect	Phase Difference
ao		
oo	reflection in forward time	ϕ_2
ob		
bb	time reversal	
bo		
of	retardation in backward time	$-\phi_1$
fd		
Nett Effect		$-\phi_1 + \phi_2$

Step	Effect	Phase Difference
ao		
oe	retardation in forward time	ϕ_1
ec		
cc	time reversal	
ce		
eo	retardation in backward time	$-\phi_1$
oo	reflection in backward time	$-\phi_2$
of	retardation in backward time	$-\phi_1$
fd		
Nett Effect		$-\phi_1 - \phi_2$

5. Paradigm Shifts

If a perspex machine, or a Turing computable subset of it, produces a rational approximation to a continuous function at increasing precisions then it will, in general, produce a close approximation that is followed by increasing numbers of less close approximations. In other words, the accuracy of approximation goes through a shift where a less precise approximation is more accurate than many subsequent, more precise, approximations. This numerical property is analogous to the case in science where a radical theory is produced in a paradigm shift, is made more precise during a period of conventional science, but without radically increasing its accuracy, until a radically more accurate theory is produced in a paradigm shift. The numerical shifts in the tightness of a bound are called "paradigm shifts" here because they are likely to cause scientific paradigm shifts in any machine sophisticated enough to develop symbolic theories that describe a continuous property of the universe, or a property at a discrete resolution finer than its theoretical symbols. For example, suppose the word "mass" is physical symbols in a perspex machine and is related to objects in the universe by the way the machine moves. The word "mass" might initially mean a volume, but come to be specified by weight as the machine produces a more accurate measure of the effort needed to move a "mass" of things of varying density. As further accuracy is obtained the word "mass" might come to mean inertial mass, as the machine experiences the varying weight of things under different gravitational conditions. Whatever quantities the machine uses they will go through numerical paradigm shifts, any of which might cause a scientific paradigm shift in the perspex machine's symbolic processing.

Without loss of generality, consider the bounded segment of the real number line $[0, 1]$. This is shown in Figure 10 as the circumference of a circle drawn clockwise from 0 to 1. (As rational bounds on this segment are symmetrical about $1/2$ the circular figure has the advantage of illustrating the symmetry.)

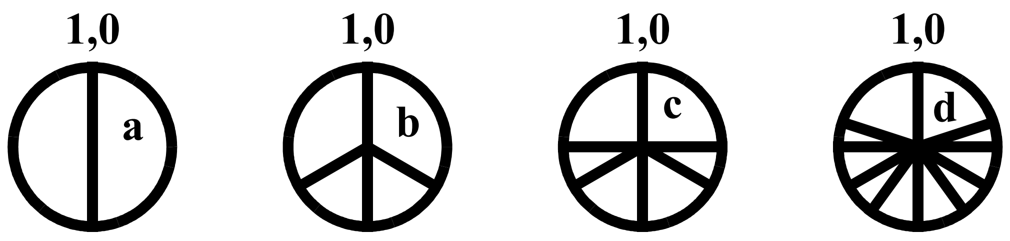

Figure 10: Successive rational bounds – walnut cake.

Suppose that measurements are made by making a cut from the centre of the circle to the circumference. In the first generation 2 cuts are made, dividing the circle into 2 equal parts of size *a.* In the second generation 3 cuts are made, dividing the circle into 3 equal parts of size *b,* in addition to some parts surviving from earlier cuts. There are just 4 sectors in the circle *b,* not $2 + 3 = 5$, because of the common cuts at the position $2/2 = 3/3 = 1$. In the third generation 4 cuts are made, but there are just 6 sectors, not $2 + 3 + 4 = 9$, because of the additional common cuts at $4/4 = 1$ and $2/4 = 1/2$. This process continues without limit. At each stage the potential number of cuts is given by the arithmetic sequence $2, 3, 4, \ldots, n$, but all of the repeated cuts are removed from the sequence.

Developing a formula for all repeated cuts would be equivalent to developing a formula for the prime numbers. A more tractable approach is to find polynomial bounds on the sequence, which we do next in a simple presentation of a more general theorem we call the "walnut cake theorem." We examine the case of measurements that lie in a sector bounded both above and below by cuts. The number of cuts on the line segment $[0, 1]$ grows arithmetically, so there are order $\mathrm{O}n^2$ cuts up to precision n. It follows that adding $n + 1$ cuts in the next term of the sequence is not sufficient to subdivide all of the sectors so, in general, a measurement at precision n will not be improved until many new cuts have been made. That is, a measurement remains the most tightly bounded for many terms before going through a paradigm shift. We now seek a sharp bound on when this paradigm shift occurs for new measurements taken at a single precision.

Lemma 1: Upper bound. If all segments at precisions $2, 3, \ldots, n$ have been constructed, then measurements at the single precision n^2 are more than sufficient to subdivide each segment, tightening the bounds everywhere.

Proof 1: The smallest constructed sector is of size:

$$1/(n-1) - 1/n = 1/(n^2 - n)$$

Therefore, segmenting at the single precision $1/n^2$ is more than enough to subdivide every sector constructed up to precision n. In other words, given measurements up to precision n, a paradigm shift occurs before a measurement at precision n^2.

Lemma 2: Lower bound. If all segments at precisions $2, 3, \ldots, n$ have been constructed then measurements at a single precision of order $\mathrm{O}(n^2/8)$ are *not* sufficient to subdivide each segment, i.e., they do *not* tighten the bounds everywhere.

Proof 2: Consider the sequence:

$$a_1 = 2, a_2 = 3, \ldots, a_k = k+1, \ldots, a_{n-1} = n \text{ where}$$

$$a_k = \begin{cases} n/2, n \text{ even} \\ (n-1)/2, n \text{ odd} \end{cases}$$

All cuts at precisions $a_i = a_1, a_2, \ldots, a_k$ are duplicated by cuts at double this precision $2a_i = a_{k+1}, a_{k+2}, \ldots, a_{n-1}$. And there is at least one cut at $1/n$ that is not duplicated. Hence, by the sum of the arithmetic sequence, there are more than:

$$\sum_{i=2}^{k} a_i = \begin{cases} (n^2-4)/8, n \text{ even} \\ (n^2-2n-3)/8, n \text{ odd} \end{cases} \text{ cuts.}$$

Therefore a lower bound is of order $\mathrm{O}(n^2/8)$.

Both the upper and lower bound are of order $\mathrm{O}n^2$ so we accept n^2, as given by Lemma 1, as a sharp upper-bound on the single precision that will tighten the bounds on any and all measurements at precisions $2, 3, \ldots, n$.

The above proof deals with Turing computable functions that are bounded both above and below. It can be relaxed to partial sequences of measurements, and is readily extended to give the probability of a paradigm shift in Turing semi-computable functions that are bounded only above or else below. In every case the $\mathrm{O}n^2$ result holds.

6. Mental Properties

The perspex thesis is a materialistic thesis in which all mental properties are hypothesised to be physical. The phenomenon of consciousness was set out in terms of perspexes in[3], based on an earlier, more abstract, definition[1]. The definitions, below, set out a pan-psychic model of mind grounded in perspexes. In most cases these are a clarification of earlier definitions, but the definition of intelligence is new. It is hypothesised that all forms of intelligence involve establishing a symmetry between a knowledge representation in a machine and objects that may be in the world. In the definition of intelligence, $f(a) = a$ establishes a symmetry and $g(f(a)) = g(a)$ allows a perceptual function g, in the machine, to relate an internal knowledge representation, $f(a)$, to a, possibly, external object a. Further, it is hypothesised that symmetry in knowledge representations manifests as efficient processing and economy of explanation, the hallmarks of intelligence. It is further hypothesised that morality involves the symmetry of dealing "fairly," i.e. symmetrically, so that, as defined, morality is identical to intelligence.

action $\vec{x}\vec{y} \to \vec{z}$; $\mathrm{jump}(\vec{z}_{11}, t)$.
afferent The afferent vectors are x and y. The afferent perspexes are $\vec{x}$ and $\vec{y}$. They are called afferent by analogy with an afferent nerve that brings information in from the body to the brain. Compare with *efferent* and *transferent.*
agent All of the *wills* in a *body* that set the *body* into *motion.*
body A collection of perspexes.
causality See *action.*
consciousness A partial, bi-directional mapping between perspexes.

efferent The efferent vector is z. The efferent perspex is $\vec{z}$. They are called efferent by analogy with an efferent nerve that takes information outward from the brain to the body. Compare with *afferent* and *transferent.*

feeling *Consciousness caused* by a collection of *afferences.*

intelligence $g(f(a)) = g(a)$ where a is a collection of *perspexes;* and f, g, and "=" are functions implemented in perspexes.

mind A collection of *actions.*

morality See *intelligence.*

motion $\vec{x}\vec{y} \to \vec{z}$.

selection $\mathrm{jump}(\vec{z}_{11}, t)$.

transferent Relating to any, or all four, control paths used by $\mathrm{jump}(\vec{z}_{11}, t)$ that transfer control from one perspex to another. Compare with *afferent* and *efferent.*

will *Conscious selection* of *action.*

will, free An *agent's will* is *free* if it is not *willed* by another *agent.*

7. Conclusion

The perspex machine is a super-Turing machine that arose from the unification of the Turing machine and projective geometry. It is a continuous machine so, we suppose, it can model all physical things, including minds, to arbitrary accuracy. If so, this establishes an isomorphism between the perspex machine and all physical things so it is correct to say that the physical universe instantiates a perspex machine. This thesis, in itself, solves the mind-body problem by holding that the perspex machine is a physical thing that is also a mind. Furthermore, we have begun a more detailed account of how the machine describes minds and bodies. We have shown how the machine can model physical things, such as bodies composed of tetrahedra moving through space and time, without time travel paradoxes. We have shown how theories, expressed in any physical format, are limited by the physically numerical properties of space so that they undergo paradigm shifts when describing properties at a finer resolution than themselves. We claim to have identified a physical basis for intelligence and morality (fair dealing). Future work will concentrate on technical aspect of the perspex machine.

8. Erratum

The sign convention in equation 18 of paper[4] is not explicit. The convention is in two parts. Firstly, the integer square root is signed. That is, the positive or negative root $\sqrt{x}$ is chosen so that $\mathrm{sgn}(\sqrt{x}) = \mathrm{sgn}(x)$. Secondly, the radius r is non-negative. Consequently the sign of the denominators p and q of r/p and r/q is chosen so that $\mathrm{sgn}(p) = \mathrm{sgn}(r/p)$ and $\mathrm{sgn}(q) = \mathrm{sgn}(r/q)$.

References

1 J.A.D.W. Anderson "Visual Conviction" *Proceedings of the Fifth Alvey Vision Conference* pp. 301-303 (Sept. 1989).

2 J.A.D.W. Anderson "Representing Geometrical Knowledge" *Phil. Trans. Roy. Soc. Lond.* series B, vol. 352, no. 1358, pp. 1129-1139, Aug. 1997.

3 J.A.D.W. Anderson "Robot Free Will" *ECAI 2002 Proceedings of the 15th European Conference on Artificial Intelligence* Lyon, France, ed. F. van Harmelan, pp. 559-563, 2002.

4 J.A.D.W. Anderson, "Exact Numerical Computation of the Rational General Linear Transformations" in *Vision Geometry XI* Longin Jan Latecki, David M. Mount, Angela Y. Wu, Editors, Proceedings of the SPIE Vol. 4794, 22-28 (2002). (See the erratum in the current paper.)

5 J.A.D.W. Anderson, "Perspex Machine" in *Vision Geometry XI* Longin Jan Latecki, David M. Mount, Angela Y. Wu, Editors, Proceedings of the SPIE Vol. 4794, 10-21 (2002).

6 PovRay is a publicly available ray-tracing package. See http://www.povray.org

7 Pop11 is a publicly available AI language. See http://www.poplog.org

8 The author's web sites are http://www.reading.ac.uk/~sssander and http://www.bookofparagon.btinternet.co.uk