Arm One *C-language*
Little Man One *DOS*
Calculations

THE CONTROL SYSTEMS IN ARM VERSION 1, ORIGINAL DOS VERSION

This program simulates a human arm reaching out to touch a target the user can move in three dimensions. The arm has three degrees of freedom (two at the shoulder and one at the elbow). The position of the "fingertip" is ray-traced to form two retinal images in which both the target and fingertip positions appear. These images are used to derive x, y, and distance signals, which are controlled by a visual system that varies the reference signals entering the three kinesthetic higher-order control systems.

INTRODUCTION

The routines that run the arm model are all based on integer arithmetic, for speed. In some routines, protection against dividing a very large number by a very small number was omitted for the same reason. As a result, the model will occasionally halt on a "division by zero" error. This occurs mostly when the fingertip is very close to the eyes and a rapid lateral or vertical movement of the target occurs, with rather loose settings of the second-level integration factors. Just restart the program. This model is not as skillful as a real human being!

Those who are used to models in which output is calculated may be taken aback by the casual treatment of sensitive calculations such as time-integrations, and by other approximations that frequently occur. In any output computation involving time integrations, the normal approach would require elaborate methods for achieving accurate integrations and setting initial conditions, for the smallest errors will accumulate over time. If, as in this model, there are thousands of calculations per second, and they are done with an accuracy of at best 0.1 per cent, and the model is allowed to run for many thousands of iterations, the normal expectation would be that the model's behavior would rapidly drift off in meaningless directions. In this control-system model where all calculations involve closed causal loops, no such effects occur. The model will run indefinitely without any cumulative errors.

The calculations to follow are given in the programming language Pascal (Borland International's Turbo Pascal 5.0). It is assumed that the reader understands the notation of this language, can infer its meaning, or will refer to a book on this language for enlightenment. One hint: the term "div" indicates integer division. The meaning of the program segments is described in English. It is also assumed that the reader either has some basic acquaintance with control theory or is willing to skip sections that are unfamiliar. See *Some basic concepts of Perceptual Control Theory* on page 9.

In all control-system calculations, the error signal may be used in the positive or the negative sense; this is determined by details of the calculations in the external feedback loop. Adjusting the overall loop gain to be negative was easiest to accomplish in this way.

I: THE KINESTHETIC CONTROL SYSTEMS

There are three kinesthetic control systems, which are assumed to sense and control joint angles. The lower-order systems which make the muscles produce the requested joint angle are assumed to be optimally damped, and are not modeled. A summary of variable definitions is shown on page 8.

Each of these systems senses its appropriate joint angle, compares the sensor signal with the reference signal given from higher-order systems, and produces an output change in joint angle that is driven by a "slowed proportional" output signal. The slowed-proportional signal is computed by multiplying the error signal by a gain factor, then letting the actual output signal change a specified fraction of the way from its previous value toward the new value. The outputs of all three systems are linked to the arm by the procedure Link1to0, below.

In trigonometric calculations, there are 4096 angle units per circle. All trigonometric functions are taken from precomputed tables of integer values, scaled to a maximum value of 4096.

Kinesthetic Azimuth control (horizontal angle)

This procedure calculates the output signal of the kinesthetic azimuth control system, which swings the arm horizontally about a vertical axis passing through the right shoulder. The reference signal, r1b, is a global variable and is set by a higher-order system. The perceived input azimuth "az" is set by the procedure that links the first-order output signals to the environment, discussed below. The variable i1b is a long integer (32-bit) temporary variable; the error signal is multiplied by 100 and by the gain (g1b), so that the computed next value of output is 100 times its actual size. This permits dividing by slowing factors (s1b) of up to 100 without losing precision. The result is divided by 100 to become the real output signal (o1b).

```
procedure azimuth1;
begin
 p1b := az;
  {kinesthetic perception of lateral shoulder-
   finger line's angle}
 e1b := r1b - p1b;
  {e = error. r1b comes from a higher level}
 i1b := i1b + (longint(100)*g1b*e1b - i1b) div
   s1b;
 if i1b > lim1b then i1b := lim1b;
  {set signal limits}
 if i1b < -lim1b then i1b := -lim1b;
 o1b := i1b div 100;
end;
```

Kinesthetic Elevation control

This procedure calculates the output of the system that controls the angular elevation of the fingertip above/below horizontal. T1 is the sensed and actual angle of the upper arm relative to horizontal. T2 is the angle between an extension of the upper arm and the lower arm. Because the two parts of the arm are assumed equal in length, the angle at the shoulder from the upper-arm axis to the shoulder-finger line is just half the external angle at the elbow, or T2/2. Thus the sensed elevation perception, p1a, is perceptually computed from the sensed joint angles T1 and T2: T1 + T2/2. The remainder of the procedure is identical in form to the azimuth control system; its reference signal r1a comes from a higher-order system.

```
procedure vertical1;
begin
 p1a := t1 + (t2 div 2);
  {kinesthetic perception of vertical shoulder-
   finger angle}
 e1a := r1a - p1a;
  {error = reference - perception}
 i1a := i1a + (longint(100) * g1a * e1a - i1a)
   div s1a;
 if i1a > lim1a then i1a := lim1a;
 if i1a < -(lim1a + lim1a) then i1a := -(lim1a
   + lim1a);
 o1a := i1a div 100;
  {output signal}
end;
```

Kinesthetic Distance control

"Kinesthetic distance" is actually the elbow angle T2 (subtracted from 180 degrees); the actual distance from shoulder to fingertip is given by 2*Ra*cos(T2/2), where Ra is the radius of the upper arm (equal to the lower arm). In Pascal, multiplication is always indicated by an asterisk (*). When the elbow angle changes, the elevation control system is disturbed, but it alters the upper-arm angle (T1) to compensate for the disturbance due to the change in T2. Thus when the distance control system changes the elbow angle, the fingertip moves straight out from the shoulder, the elevation angle of the fingertip remaining constant. This is accomplished without any explicit coordination between elevation and distance control and is a natural consequence of feedback control.

```
procedure distance1;
begin
 p1c := 2048 - t2;
  {kinesthetic perception of shoulder-finger dis-
    tance: inner angle at elbow. 2048 is equivalent
    to 180 degrees.}
 e1c := r1c - p1c;
 i1c := i1c - (longint(100)*g1c*e1c - i1c) div
    s1c;
 if i1c > lim1c then i1c := lim1c;
 if i1c < 0 then i1c := 0;
 o1c := i1c div 100;
end;
```

Linking first-order systems to the physical arm.

Here the loops are closed for each of the kinesthetic control systems. The output of the elevation (vertical angle) system, for example, becomes T1, the angle at the upper arm. If we were modeling the still-lower-order systems, this is where the outputs of the kinesthetic joint-angle control systems would become reference signals for lower-order force-control systems, and where the physical dynamics of the arm would be introduced.

This linking procedure also links the "gaze" control systems to their environmental effect: gazex is the azimuth (horizontal) head angle, and gazey is the elevation (vertical) head angle.

```
procedure link1to0;
begin
 t1 := o1a;     {vertical}
 az := o1b;     {azimuth}
 t2 := o1c;     {distance}
 gazex := o1d;  {gaze x}
 gazey := o1e;  {gaze y}
end;
```

II: THE HEAD ANGLE (VISUAL TRACKING) CONTROL SYSTEMS

These two systems use visual information about the x and y positions of the target relative to the line of sight from the right eye only. It would be just as easy to compute this information as the average as seen by both eyes, which would permit tracking to continue with one eye closed. These systems keep the right eye looking directly at the target, by turning and tilting the head.

The visual information is derived in a way described below under "Environment calculations." The relevant information is the deviation of the target, in angle, from the line of sight, in the x (horizontal) and y (vertical) directions.

The target position is set independently by the user of this program, from the keyboard in ARMKEY.EXE or from a joystick in ARM.EXE.

Vertical visual (head) tracking

The variable TRy is the angular Target deviation in the y direction from the line of sight of the Right eye. The output function is a pure time-integrator: the output signal accumulates according to the size and direction of the error signal. The reference signal is set to zero, meaning that this system attempts to bring the y target position to the line of sight (by moving the head and thus the line of sight). If the reference signal were nonzero, the head would move to keep the target above or below the line of sight by the specified amount. The output signal becomes the head vertical angle, "gazey," in the Link1to0 procedure above. Gazey is the angle that is used (after adjustment for running off the ends of the trigonometric tables) in the Environment Calculations section to compute the vertical angle between the line of sight and the direction to the target, TRy. The lower-order muscle systems that make physical head angle follow the output signal are assumed.

```
procedure gaze1y;
begin
 r1e := 0;
 p1e := TRy;
  {perceived y target deviation from gaze angle}
 e1e := r1e - p1e;
 i1e := i1e - e1e;
  {time-integration (with negative sign)}
 if i1e > lim1e then i1e := lim1e;
 if i1e < -lim1e then i1e := -lim1e;
 o1e := i1e div g1e;
end;
```

Horizontal visual (head) tracking

The variable TRx is the angular Target deviation in the x direction from the line of sight of the Right eye. The horizontal tracking system works exactly as the vertical system does. Its output signal, o1d, becomes the head horizontal angle through specifying "gazex" in the Link1to0 procedure above. Gazex becomes the angle used in the Environment Calculations section to compute the horizontal angle between the line of sight and the direction to the target, TRx.

```
procedure gaze1x;
begin
 r1d := 0;
 p1d := TRx;
  {perceived x target deviation from gaze angle}
 e1d := r1d - p1d;
 i1d := i1d - e1d;
  {time-integration}
 if i1d > lim1d then i1d := lim1d;
 if i1d < -lim1d then i1d := -lim1d;
 o1d := i1d div g1d;
end;
```

III: LATERAL AND VERTICAL VISUAL ARM CONTROL

The higher-level systems that move the finger sideways and vertically on the basis of visual information sense the deviation of the finger from the target as seen by the right eye. The average deviation seen by both eyes could also be used, but by using the right eye we imitate the effect of "dominance" (and speed the calculations).

These two systems are conceptually of a higher order than the visual head control systems. The visual head control systems use only information about the target angular position relative to the center of vision of the right eye; this is an "absolute" position in the eye's framework, and depends on the direction of gaze. The systems we now consider sense an invariant in the eye's framework, namely, the difference in position between two objects in the same field of vision, the finger and the target. This angular relative distance is invariant (to a first approximation) with respect to the absolute position of the images on the retina, and hence is unaffected (nearly) by the head's movements. The lateral visual control system is organized to bring the horizontal component of finger-target angular separation to zero by moving the arm in azimuth (swinging it horizontally); the vertical visual control system is organized to bring the vertical component of finger-target angular separation to zero by moving the elevation angle of the line extending from shoulder to fingertip.

Neither of the two physical arm movements is exactly in the direction needed to correct either visual error; as a result, correcting one visual error disturbs the other visual control system. Nevertheless, the two systems maintain their own perceptual variables in the reference state, zero. There are no output computations that compensate for the "wrong" directions of movement; this is accomplished entirely by visual feedback effects.

If non-zero reference signals (r2a and r2b) were used, the finger would track the target, but at a fixed distance vertically and horizontally from it.

Horizontal visual target-finger control

This system perceives the difference in target and finger angles in the x direction. It integrates the error signal computed relative to a reference signal of zero. The output signal that results, o2b, becomes the reference signal for the kinesthetic azimuth control system; the link occurs in the procedure Link2to1, below.

The perceptual signal is adjusted for "size constancy". If the Finger Left-Right angle difference FLRd is greater than 10 angle units (it is, over the normal range of distances), the Target signal is computed as the produce of the Target x displacement in the Right eye, TRx, multiplied by a constant and divided by the angle difference FLRd. That is nearly the same as multiplying the angular deviation by the perceived target distance. The perceived Finger displacement in x seen by the Right eye, FRx, is also multiplied by the distance in this way. The perceptual signal p2b is the difference between these two quantities.

The function "dmd" is a "double-multiply-divide" assembly-language macro that multiplies the first two arguments to give a 32-bit intermediate result, which is then divided by the third argument to give an integer (16-bit) result. This preserves accuracy when multiplying an integer by a ratio of integers.

```
procedure azimuth2;
begin
 r2b := 0;
 if FLRd > 10 then
 p2b := dmd(TRx,90,TLRD) - dmd(FRx,90,FLRd);
 {perceived finger -target azimuth x constant x
   distance}
 e2b := r2b - p2b;
 if e2b > errlim then e2b := errlim
 else if e2b < -errlim then e2b := -errlim;
 i2b := i2b - e2b;
 if i2b > lim2b then i2b := lim2b;
 if i2b < -lim2b then i2b := -lim2b;
 o2b := i2b div g2b;
end;
```

Vertical visual target-finger control

This system works exactly as above, but perceives the vertical angular separation between finger and target. Its output, o2a, becomes the reference signal for the kinesthetic vertical position control system. The function "dmd" is explained above.

```
procedure vertical2;
begin
 r2a := 0;
  {reference target-finger vertical angle}
 if FLRd > 10 then
 p2a := dmd(TRy,90,TLRd) - dmd(FRy,90,FLRd);
 {perceived target-finger vertical angle x con-
   stant x distance}
 e2a :=  r2a - p2a;
 if e2a > errlim then e2a := errlim
 else if e2a < -errlim then e2a := -errlim;
 i2a := i2a - e2a;
 if i2a > (lim2a + lim2a) then i2a := lim2a +
   lim2a;
 if i2a < -lim2a then i2a := -lim2a;
 o2a := i2a div g2a;
end;
```

IV: VISUAL DISTANCE CONTROL

The final control system uses relative target-finger angular separations from both eyes, information that is invariant with respect to location of the images on the retina, to a first approximation.

The visual distance control system uses the visual parallax from binocular viewing of both Finger and Target. Only information about deviations in the x (horizontal) direction is used. Visual distance is the reciprocal of the angular difference between left-eye and right-eye images of the same object. For the finger, the angles relative to the direction of gaze are FLx and FRx, and for the target, TLx and TRx. The information obtained from each eye contains the angle between finger and target: TLx – FLx, or TRx – FRx. As mentioned, only the latter is used for lateral visual control of the finger relative to the target. If each relative angle is represented by a signal, then subtracting the left-eye signal from the right-eye signal is equivalent to computing (TRx – FRx) – (TLx – FLx), which by simple transposition is (TRx – TLx) – (FRx – FLx). Thus we get the difference between parallaxes of the finger and target, without requiring that the eye be looking directly at either object. This difference in parallaxes corresponds to the relative distance in depth between finger and target (in a nonlinear way). When this difference is zero, the finger is at the same distance as the target.

Depth information can be obtained without visual tracking by the head, if the objects are somewhere in the visual field. Visual tracking has the effect of keeping the baseline between the eyes nearly perpendicular to the line of sight to the target, which changes the "horopter" of classical visual research (surface of equal apparent distances) almost to a sphere. Visual tracking was ignored in the classical analysis. With visual head tracking, distance control works better over a wider lateral range of angles because the effective baseline does not change (except during transient errors).

To simplify computations, the difference in target angle is computed outside this procedure from the basic angle information about target and finger as seen by each eye. The method of subtracting the target-to-finger angle in one eye from the target-to-finger angle in the other eye would be exactly equivalent. In this procedure, TLRd is the Target Left-Right deviation in angle, and FLRd is the same deviation for the finger.

The quantity REx – LEx is the baseline distance between the eyes.

FLRd is defined as FLx – FRx, and TLRd as TLx – FLx. Distance is obtained by multiplying the eye separation, (REx – LEx) by a constant and dividing the product by FLRd or TLRd.

As mentioned above, the function "dmd" is a "double-multiply-divide" assembly-language macro that multiplies the first two arguments to give a 32-bit intermediate result, which is then divided by the third argument to give an integer (16-bit) result.

Note that the perceptual signal p2c is the difference between perceived target distance and perceived finger distance. It is therefore a perception of relationship. This control system also works properly if we define the perceptual signal as p2c := TLRD – FLRd (i.e., the reciprocal of distance). The sense of the error signal has to be reversed. This redefines the variable as a "closeness" variable—the larger either angle-difference is, the closer the object is. With this definition, the loop gain rises rapidly as the finger approaches the eyes, so performance is less uniform over the whole range of distances. But the nonlinearity has no other effect: when the perception is zero (matching the zero reference signal), the finger is at the same perceived distance as the target in either case.

```
procedure distance2;
var u,v: integer;
begin
 r2c := 0;
 u := dmd(REx - LEx,1024,FLRd);
  {finger distance}
 v := dmd(REx - LEx,1024,TLRd);
  {target distance}
 p2c := (v-u);
  {relative distance}
 e2c := r2c - p2c;
 if e2c > errlim then e2c := errlim
 else if e2c < -errlim then e2c := -errlim;
 i2c := i2c - e2c;
  {compute total integral first}
 if i2c > lim2c then i2c := lim2c;
 if i2c < 1 then i2c := 1;
 o2c := i2c div g2c;
  {reduce by integration factor}
end;
```

Linking to Level 1.

The output signals of the higher-order control systems become the reference signals for the kinesthetic control systems in the following procedure, which closes all the remaining loops except for the environmental part:

```
procedure link2to1;
begin
 r1a := o2a;    {vertical angle}
 r1b := o2b;    {azimuth}
 r1c := o2c;    {distance}
end;
```

VI: ENVIRONMENTAL CALCULATIONS

By far the most complex computations in this model concern the physical properties of the body and its environment.

When the figure is viewed from its right side, the body is placed 200 units to the left of the origin and the positive z coordinate extends to the right. The positive y coordinate extends upward, and the positive x coordinate extends toward the viewer.

Before the "Environment" procedure is entered, the gaze angles that have been computed as output signals from the two head-angle control systems are used to rotate the head and eyes in space, first about the x axis and then about the y axis. Using the other sequence of rotations would tilt the head sideways. The result is a new set of eye positions, used below, and a stored polygon of x-y coordinates for drawing the head and eyes. The Left Eye x-coordinate after rotation is symbolized LExr, and so on.

Computing fingertip position

The kinesthetic control systems make perceived joint angles match their given reference signals, but know nothing of where that places the elbow or the fingertip. From the given angles and the radius of the upper and lower arm (identical), we must first calculate where the fingertip will be in space. The origin of this space is a point 200 units distant (in the "z" coordinate) from the place where the neck of the figure meets the body (coordinates 0,0,–200, in x,y, and z). The shoulder around which the upper arm swivels is located at x = 80, y = 0, and z = –200. Note that these calculations are purely physical and are independent of the behavior-model.

In the following, SH is shoulder position and FI is finger position. Expressions like sine[x] and cosine[x] do not refer to the built-in trigonometric functions of Pascal (note spelling and use of brackets rather than parentheses), but to precomputed tables; "x" would be an index into the table, where index values can run from –2048 (–180 degrees) to 2047 (+179.91 degrees). Readers converting to the "C" language will have to compensate for these negative indices, which "C" cannot use.

"Scale" is a constant equal to 4096, and represents the one-way peak value of sines or cosines in the stored tables.

```
procedure environment;
begin

 t3 := t1 + t2 div 2; {angle, horiz to shoulder-
   finger line}

{Find x,y,z coordinates of fingertip}

 rf :=  dmd(ra + ra,cosine[t2 div 2],scale);
  {radius, shoulder-finger}

 FIy := SHy + dmd(rf,sine[t3],scale);
  {y coordinate of finger}
 temp := dmd(rf,cosine[t3],scale);
 FIx := SHx + dmd(temp,sine[az],scale);
  {x coordinate of finger}
 FIz := SHz + dmd(temp,cosine[az],scale);
  {z coordinate of finger}
 if FIz < LEz + 25 then FIz := LEz + 25;
  {keep finger in bounds}
if FIz < REz + 25 then FIz := REz + 25;

{Find x,y,z coordinates of elbow}

 ELy := SHy + dmd(ra,sine[t1],scale);
 temp := dmd(ra,cosine[t1],scale);
 ELx := SHx + dmd(temp,sine[az],scale);
 ELz := SHz + dmd(temp,cosine[az],scale);

   {Fill in 3-dimensional polygon for arm
   points}
 arm3[1].x3d := SHx; arm3[1].y3d := SHy; arm3[1].
   z3d := SHz;
 arm3[2].x3d := ELx; arm3[2].y3d := ELy; arm3[2].
   z3d := ELz;
 arm3[3].x3d := FIx; arm3[3].y3d := FIy; arm3[3].
   z3d := FIz;
```

The final steps above fill in an array containing a series of x,y,z coordinates: the shoulder, the elbow, and the arm positions. This is later rotated according to the viewing angle, then projected into two-dimensional x-y coordinates for display.

The environment procedure then continues:

Calculating target, finger angles relative to gaze

The first step is to compute x,y,x coordinates of the target and finger position relative to each eye. TXL, for example, means Target X coordinate relative to Left eye.

A notation like LExr means Left Eye, x coordinate, rotated, calculated as explained above before entering this procedure.

```
{Calculate temporary variables for computing eye
   angles}

 TXL := TAx - LExr; TXR := TAx - RExr;
 TYL := TAy - LEyr; TYR := TAy - REyr;
 TZL := TAz - LEzr; TZR := TAz - REzr;

 FXL := FIx - LExr; FXR := FIx - RExr;
 FYL := FIy - LEyr; FYR := FIy - REyr;
 FZL := FIz - LEzr; FZR := FIz - REzr;
```

In the following, the procedure “rotateimage” is entered with the x,y,z coordinates of an object, target or finger as seen by one eye, and the angles of rotation theta2 and phi2, which are simply copies of the two gaze angles, gazex and gazey (compensated for overrunning the limits of the sine and cosine tables). The first two arguments are passed “by reference” and indicate the returned values. These values are the angles in x and y of the object relative to the direction of gaze. The angles thus represent the displacement of the objects from the center of the field of view for each eye. This routine makes use of the rotated positions of the eyes, which are global variables.

The final two statements compute the parallax angles of finger and target from the angle variables just calculated.

```
  {Find angles, finger and target, right eye}

 rotateimage(TRx,TRy,TXR,TYR,TZR,theta2,phi2);
 rotateimage(FRx,FRy,FXR,FYR,FZR,theta2,phi2);

 {Find angles, finger and target, left eye}

 rotateimage(TLx,TLy,TXL,TYL,TZL,theta2,phi2);
 rotateimage(FLx,FLy,FXL,FYL,FZL,theta2,phi2);

 {Find depth angles, eyes to target and finger,
   radians}

 FLRd := FRx - FLx;
 TLRd := TRx - TLx;

 end;   {of environment procedure}
```

VII: COMMENTS ON THE MODEL

The discussions above have reviewed the main calculations for each of the eight control systems, and for the environmental part of the control loops. The Registered User version of this demonstration contains all the source code (Borland International's Turbo Pascal 5.0) including the special Units used. Most of this code is occupied with doing the rotations, calculating positions for plotting, initializing, and allowing for editing of parameters.

The plots that are displayed on the screen show the horizontal angle, vertical angle, and distance-differential between the finger and the target, as seen by the modeled person.

It would be possible to assign two separate control systems to each eye, while retaining control of the head as a whole. The eyes could then converge on the target (or the finger), and track the target independently of the head. This would lead to some interesting studies of the effect of relative loop gains in the head and eye control systems. The extra plotting required, as well as the extra rotation and control-system calculations, however, would slow the display. In later versions of this program, eye convergence will be introduced—although to preserve speed, much of the program might have to be converted to assembler language, or to a somewhat faster language such as "C".

A further refinement of the model will involve introducing the lower-order control systems that convert joint-angle position error signals into commands to force-control systems. These systems will need the physical properties of the arm to be calculated—mass and moments of inertia. This will undoubtedly slow the display considerably, but will be worth doing just to see what problems arise. This would increase the number of levels in the model to about five, and with eye-convergence included, the number of control systems to about 15.

VARIABLE DEFINITIONS

```
halfside,   {half of one side of box}
FIx,   {x position of finger}
FIy,   {y position of finger}
FIz,   {z position of finger}
TAx,   {x position of target}
TAy,   {y position of target}
TAz,   {z position of target}
REx,   {right eye x coordinate}
REy,   {right eye y coordinate}
REz,   {right eye z coordinate}
LEx,   {left eye x coordinate}
LEy,   {left eye y coordinate}
LEz,   {left eye z coordinate}
REx r, {right eye rotated x coordinate}
REyr,  {right eye rotated y coordinate}
REzr,  {right eye rotated z coordinate}
LExr,  {left eye rotated x coordinate}
LEyr,  {left eye rotated y coordinate}
LEzr,  {left eye rotated z coordinate}
ELx,   {x coordinate of elbow}
ELy,   {y coordinate of elbow}
ELz,   {z coordinate of elbow}
SHx,   {x coordinate of shoulder}
SHy,   {y coordinate of shoulder}
SHz,   {z coordinate of shoulder}
ra,    {length of upper and lower arms}
t1,    {angle from nadir to upper arm}
t2,    {angle, projection of upper arm to lower arm}
t3,    {angle from shoulder to finger}
az,    {azimuth angle of arm}
TLRd,  {depth angle to target}
FLRd,  {depth angle to finger}
TRx,   {azimuth angle, right eye to target}
TLx,   {azimuth angle, left  eye to target}
FRx,   {azimuth angle, right eye to finger}
FLx,   {azimuth angle, left  eye to finger}
TRy,   {vertical angle, right eye to target}
TLy,   {vertical angle, left  eye to target}
FRy,   {vertical angle, right eye to finger}
FLy,   {vertical angle, left  eye to finger}
TXL,TYL,TZL,FXL,FYL,FZL,
 {positions relative to eyes}
TXR,TYR,TZR,FXR,FYR,FZR: integer;
p1a,p1b,p1c,p1d,p1e,e1a,e1b, e1c,e1d,e1e,o1a,o1b
  ,o1c,o1d, o1e,r1a,r1b,r1c,r1d,r1e: integer;
 {lower order control}
p2a,p2b,p2c,e2a,e2b,e2c, o2a,o2b,o2c,r2a,r2b,r2c:
  integer;
 {higher order control}
g1a,g1b,g1c,g1d,g1e,s1a,s1b,s1c,s1d,s1e,
 {gain constants}
g2a,g2b,g2c: integer;
```

SOME BASIC CONCEPTS OF PERCEPTUAL CONTROL THEORY

A control system is an organization that

1 Senses some external variable through an Input Function that represents one environmental attribute as a variable perceptual signal;
2 Compares the perceptual signal against a given reference signal to generate an error signal; and
3 Operates on the error signal with an Output Function to generate an output signal.

The output signal may serve as the reference signal for lower-order control systems (in a hierarchy of control), or at the lowest level of control may actuate muscles that move the limbs or apply forces to the environment.

The effects of the output signal coming from a control system alter the world outside it. Those effects, in turn, alter the variable that is being sensed, thus altering the perceptual signal. If the control system is properly organized, its output effects will always tend to bring the perceptual signal closer in value to the reference signal, reducing the error signal that is driving the action. Disturbances that affect the input to the control system cause error signal changes that make the action of the system strongly oppose the effects of the disturbance. Thus control systems control their own inputs relative to an intended state set by the reference signal.

The "loop gain" of a control system determines how energetically it will oppose disturbances and how accurately it will maintain its perceptual signal in a match with its reference signal. The main determinant of loop gain is the sensitivity of the Output Function to error signals. The more output that a given error signal can produce through the Output Function, the smaller the final error signal will be.

The main thesis of perceptual control theory is that organisms are control systems at every level of organization. All behavior is employed to control the perceptions that behavior affects. There is no such thing as "responding to a stimulus." Stimuli are actually disturbances of controlled perceptions.