A Library of Goo for the 5620

In the following summaries, all coordinates are screen or bitmap coordinates. The library is `/usr/jerq/lib/libjj.a` which can be accessed by including a `-ljj` in your compilation.

3d

3d: scaled integer three dimensional geometry

```
#define ONE 16384
#define XAXIS   0
#define YAXIS   1
#define ZAXIS   2
typedef short fract; /* fixed point scaled by ONE */
typedef fract matrix[4][4];
typedef short coord[4];
typedef struct point3
{
    fract x, y, z, w;
} Hcoord;
```

These are the data structures for a 3 dimensional geometry package. `Hcoord` are homogeneous coordinates. Author (for all the 3d stuff): Tom Duff.

Clip

Clip: clipped jerq primitives

```
Clip(r) Rectangle r;
Cbitblt(sb, sr, db, dp, fc) Bitmap *sb, *db; Rectangle sr; Point dp;
Crectf(bp, r, fc) Bitmap *bp; Rectangle r;
Point Cstring(f, s, bp, p, fc) Font *f; char *s; Bitmap *bp; Point p;
Ctexture(bp, r, t, fc) Bitmap *bp; Rectangle r; Texture *t;
```

`Clip` sets the clipping rectangle for the routines `Cbitblt`, `Crectf`, `Cstring` and `Ctexture`. These routines implicitly clip to the layer as well. `Cstring` returns the same result as `string`, that is, the result is not clipped. Author: Andrew Hume.

NMenu

NMenu: extended menu

```
typedef struct NMitem {
    char*text;       /* text in menu */
    char*help;       /* optional help text */
    struct NMenu *next;  /* cascade to */
    void(*dfn)(), (*bfn)(), (*hfn)();
} NMitem;
typedef struct NMenu {
    NMitem  *item;          /* string array, ending with text=0 */
    NMitem  *(*generator)(); /* used if item == 0 */
    /* and some other private data */
} NMenu;
```

A `NMenu` is used by `mhit` described below. Null or missing values should be zero. The help text should be less than 40 characters or so. The generator function is called with an integer argument. The end of the menu is indicated by a zero `text` field. The function `dfn` is called just before the new menu is invoked. The function `bfn` is called just after this menu is finished. The function `hfn` is invoked just after the selection is made and before `bfn`. All three functions are called with one argument, a pointer to the `NMitem` concerned. `Mhit` returns a pointer to the NMitem that was

selected or (NMitem*)0 The above definitions are available in /usr/jerq/include/menu.h.

confirm

confirm: confirm a decision

```
confirm(but)
```

confirm waits for all buttons to be released, changes the cursor to an icon indicating which button to push for an affirmative reponse, waits for a button push and returns the equality between the button push and the argument but. Author: Adrian Freed.

icos

icos, isin, isqrt: accurate scaled functions

```
fract icos(x) angle(x);
fract isin(x) angle x;
long isqrt(x) long x;
```

icos and isin return a scaled value of their (scaled) argument. See 3d above.

init3

init3: initialise the 3d geometry world

```
init3(v, d, s) Bitmap *v;
```

init3 initialises the matrix stack (see push3below). v is the Bitmap for the image. d/s is the distance to the screen whose half-width is 1.

kbdstr

kbdstr: enter a string from the keyboard

```
kbdstr(s) char *s;
```

kbdstr displays the given text in a box 300 pixels long and allows the user to type after the prompt. The erase and kill characters are BS and @ respectively. Author: Andrew Hume.

mhit

mhit: extended analog to menuhit

```
NMitem *mhit(m, but, 0) NMenu *m;
```

mhit supports cascading menus with help. Menus may be a static list of MNitem or a generating function. mhit may be called with a button argument of 2 or 3 only. Button 1 causes the help message associated with the currently highlighted entry to be displayed. The menu items (and help messages) scroll. Author: Andrew Hume.

polyture

polyture: texture a polygon

```
polyture(bp, pts, n, t, c) Bitmap *bp; Point *pts; Texture *t; Code c;
```

Polyture textures with mode c the polygon bounded by and including the n vertices specified by pts. The list should be closed, i.e. pts[0]==pts[n-1]. Author: Andrew Hume.

push3

push3, pop3: manipulate the 3d matrix stack

```
push3()
pop()
```

`push3` pushes a copy of the top of the matrix stack onto the matrix stack. `pop3` throws away the top of the matrix stack.

rot3

rot3, rosc3: rotate a matrix

```
rot3(theta, axis) angle theta;
rotsc3(s, c, axis) short s, c;
```

These routines rotate the top of the matrix stack about the specified axis. `rot3(theta, axis)` is identical to `rotsc3(isin(theta), icos(theta), axis)`.

todo

3d stuff to be done

```
Hcoord hcoord(x,y,z,w) fract x, y, z, w;
scale3(p) Hcoord p;
tran3(p) Hcoord p;
ident(m) matrix m;
xform3(m) matrix m;
long dot(a, b) Hcoord a, b;
Hcoord unitize(x) Hcoord(x);
Hcoord cross(a, b) Hcoord a, b;
look3(e, l, u) Hcoord e, l, u;
move3(p) Hcoord p;
line3(p) Hcoord p;
```

These have yet to be documented.