Happy Hacking Keyboard のアニメーション

和田英一 †

Happy Hacking Keyboard の開発は 1995 年に始まった. いよいよ金型を作るという頃に, キーボードの 3 面図を PFU から貰った. もちろん 3 面図があれば, 大体の形は推測できるが, やはり見取り図を描いて感触もつかみたい. そこで Postscript でキーボードの見取り図を描くことにした. そこは計算機なので, 座標変換はお手のもので, 図はほどなく出来た. そのうち, キーの押し下げられている図も描けるのではないかとやってみると, これも簡単であった. 各キーについて図を用意し, それを順に表示するとキーが自動的に動くように見える. それが PFU の HHKB ホームページにあるキーボードアニメーションである. 今回はその図の描き方を説明したい.

How Happy Hacking Keyboard was animated

Eiiti Wada †

The development of the Happy Hacking Keyboard was launched in 1995. Before the prototype came out, a perspective picture was drawn using Post Script. A set of similar pictures were also prepared each with single key in the pressed down position and from which keyboard animation played by a pair of invisible hands was programmed.

1 キーボードアニメーション

Happy Hacking Keyboard のそもそもは, 1992 年 2 月の PFU Technical Review に「けん盤配列にも大いなる関心を」[1] という文を載せたことに始まる. この時は別に反応はなかったが, 3 年後に PFU の新海専務と会話中, この文に話題が及び, 文中に提案した alpha キーボード, その後考えた aleph キーボードの検討を PFU で始めることになった. 別のキーボードを改造したプロトタイプによる試用などを経, 96 年に製造が始まった. 発売は 96 年 12 月である.

私は一日も早く完成品が見たいので, PFU から貰った 3 面図を元に, 見取り図を描くことにした. Postscript で絵を描くのはかなり経験があったが, やってみるといろいろ大変であった. それでも次々と問題を解決し, 図-1 のようなものが描けた [2].

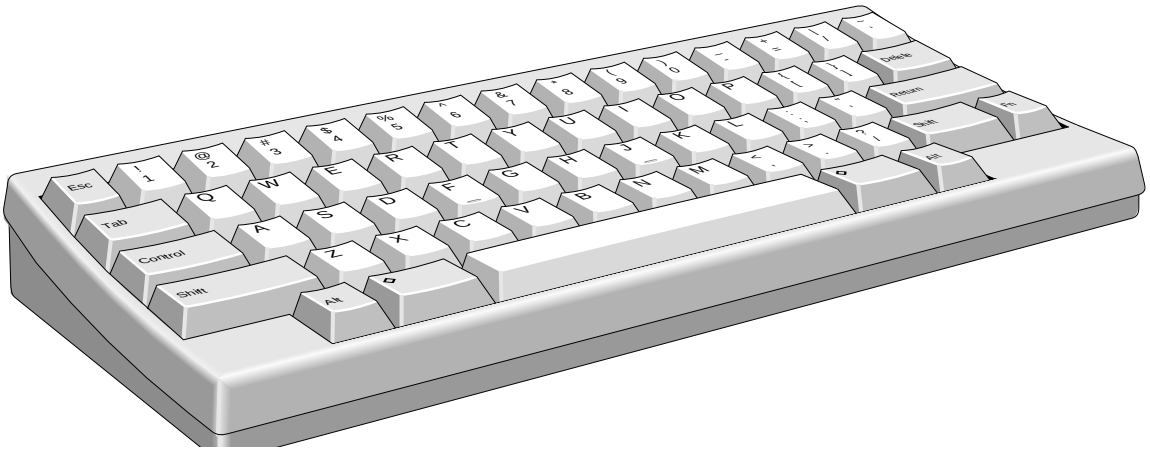

図-1 Happy Hacking Keyboard

この一つ一つのキーは図面から位置を読み, シリンドリカルスカルプチャの角度も合わせて描いているので, その位置を押し下げ距離 (3.8 ミリ) だけ下げて描けば, キーを押した図も描けるのではないかと考えた. さっそく実行してすべてのキーの各々の押し下げ位置の図を描いた. これを次々を連続表示すると面白いことにキーボードのアニメーションが見える. ちょうど自動ピアノのけん盤が見えない手で次々と押され, 演奏しているようなものである. 別にアッと驚くようなプログラムではないが, 描き方を以下に紹介する.

† インターネットイニシアティブ技術研究所 IIJ research laboratory

2　透視図のための座標変換

透視図では立体を平面に投影するので, 3 次元空間の座標 (x, y, z) を 2 次元の座標 (x', z') に変換しなければならない. その方針は以下の通りである (図-2).

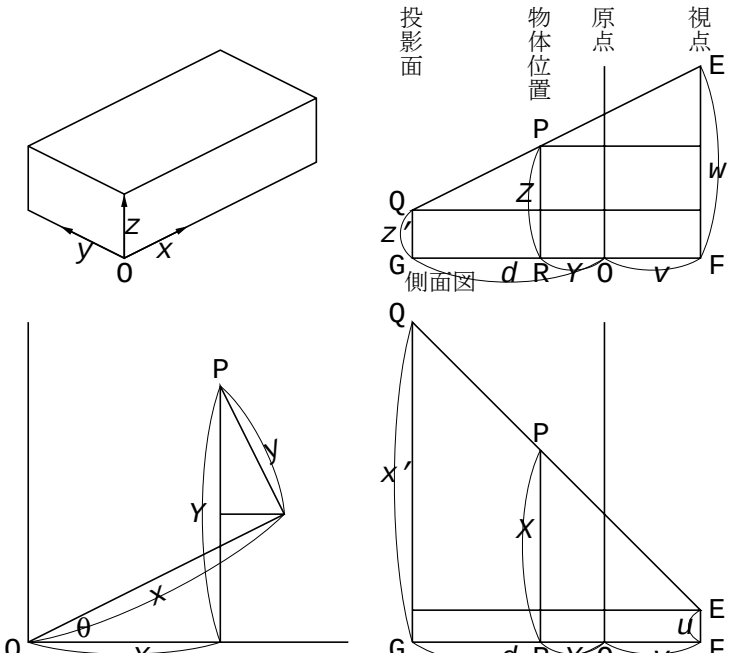

図-2 座標変換

図-2 の左上は物体にそって決めた座標軸を示す. この長方形の箱がほぼキーボードと思ってよい. これが向う側にある投影面に対し, 右手方向が奥に入るように角 θ だけ回転したとする. それを上から見たのが左下の図である. z 軸を中心にして回転するから, z 座標は変らない. 投影面と平行に手前に新しい X 座標を, それと垂直に奥に向かって Y 座標をとる. この座標系での X, Y, Z の値が

$$X = x\cos\theta - y\sin\theta, \quad Y = y\cos\theta + x\sin\theta, \quad Z = z$$

なのは見て分かる通りである.

次にこの点 P(X, Y, Z) を視点 E から見たとき, 投影面上に見える位置 Q(x', z') を右側の図で検討する. 右上は −X 方向から見た側面図, 右下は +Z 方向から見た平面図である. O は X, Y, Z の原点で, 投影面は X, Z 面から d だけ離れている. 正負のとりかたが多少いい加減だが, 視点 E は原点から v だけ手前, u だけ右, w の高さのところにある. P から X,Y 平面に下ろした垂線の足を R, Y 軸と投影面の交点を G, Y 軸と視点を含む投影面と平行な面との交点を F とする. 視点と P を結ぶ線と投影面の交点が Q である.

側面図では EF=w, OF=v, OG=d, PR=Z, PQ=Y, QG=z',

平面図では EF=u, OF=v, OG=d, PR=X, PQ=Y, QG=x'.

EP を斜辺とする直角三角形と EQ を斜辺とする直角三角形は相似だから,

$$\frac{x - z}{Y + v} = \frac{w - z'}{d + v}, \quad \therefore \ z' = w - \frac{d + v}{Y + v}(w - z)$$

$$\frac{X - u}{Y + v} = \frac{x' - u}{d + v}, \quad \therefore \ x' = \frac{d + v}{Y + v}(X - u) + u$$

x, y, z を貰い, x', z' を返す関数 `trans` は以下の通り. パラメータの値は試行錯誤で決めた.

```
% d distance of reference plane and projection plane
% v distance of eye and reference plane
% w eye height
```

```
/trans {10 dict begin /z exch def /y exch def /x exch def
/th 30 def /d 99700 def /u 0 def /v 1300 def /w 600 def
/X x th cos mul y th sin mul sub def
/Y x th sin mul y th cos mul add def %X Y rotated coordinate
v d add X u sub v Y add div mul u add 30 div %x'=(d+v)*(X-u)/(v+Y)+u
w v d add w z sub v Y add div mul sub 30 div %z'=w-(d+v)*(w-z)/(v+Y) end} def
```

この座標変換を用いてキーボードの外形を投影したものが, 図-3 である.

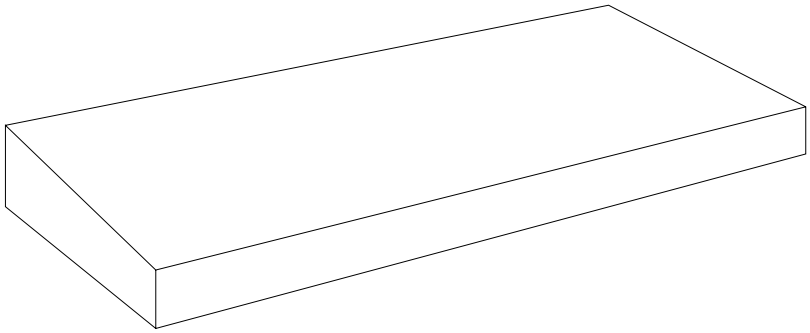

図-3 キーボードの外形

3 本体の描画

図-4, 図-5 でキーボードを描く手順を説明する. 基本は Z buffer 法と同様に遠くのものから描く.

0. 上面を塗る
1. 右奥の輪郭を描く
2. キーボードを置く位置を黒く塗る
3. [‘,~] のキー (E14) を描く
4. [\,|] のキー (E13) を描く
5. [=,+] のキー (E12) を描く
6. 同様にして E 段のキーを描く
7. 同様にして D 段のキーを描く
8. 同様にして C 段のキーを描く
9. 同様にして B 段のキーを描く
10. 同様にして A 段 Alt, ◇, スペースのキーを描く
11. キーの左端の狭い部分の表面を塗る
12. 左シフトキーの手前を塗る
13. スペースバー手前の輪郭を描く
14. スペースバーの手前を塗る
15. 下部ケースの正面を塗る
16. 正面と左面の交差部の曲面を塗る
17. 下部ケースの左面を塗る
18. 正面と左面の交差部の曲面を塗る
19. 正面と左面の交差部の光った部分を描く
20. 下部ケースの下の輪郭を描く
21. 上部ケースの正面を塗る
22. 正面, 左面の交差部の曲面を塗る
23. 上部ケースの左面を塗る

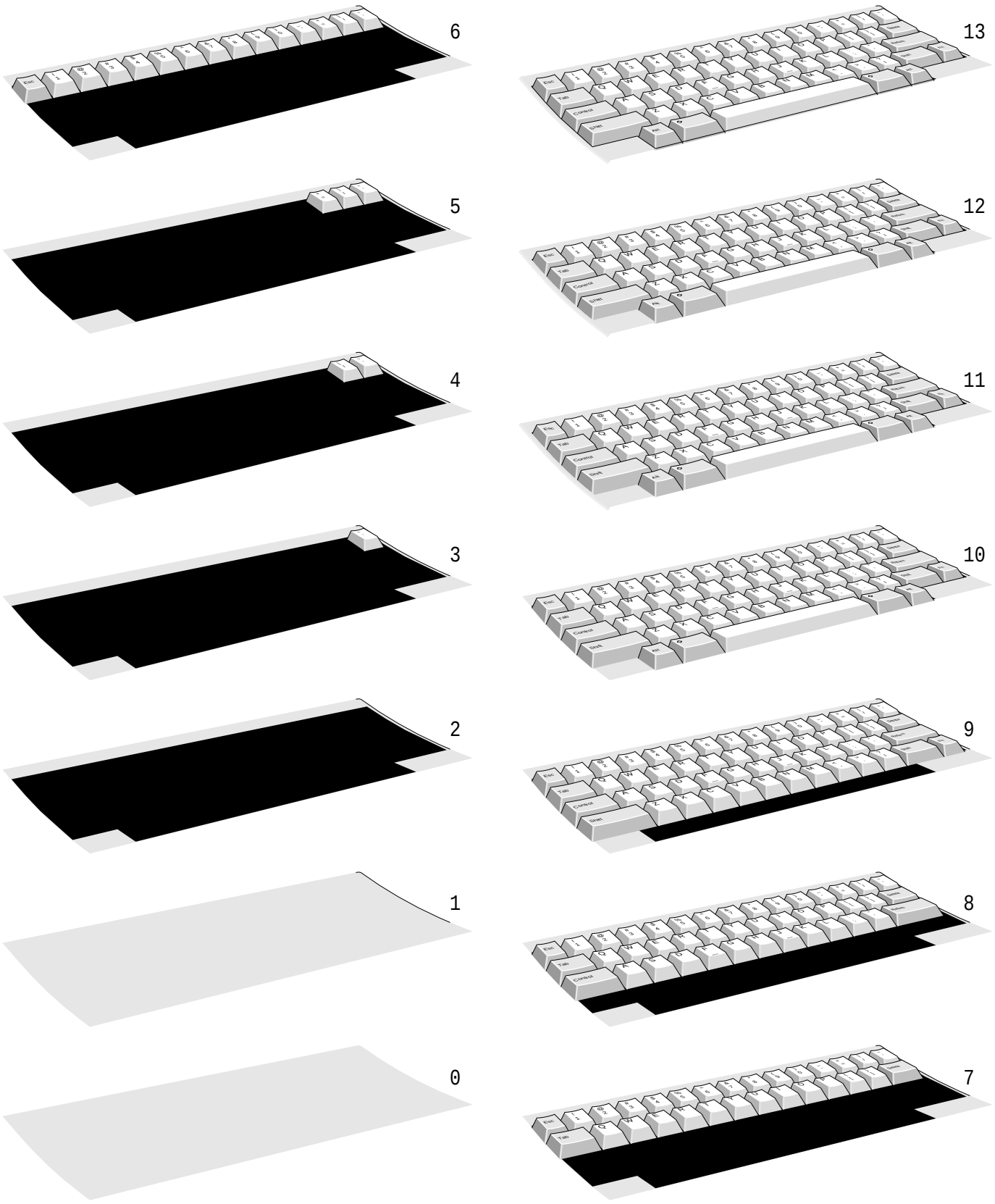

図-4 見取り図の組み立て-0

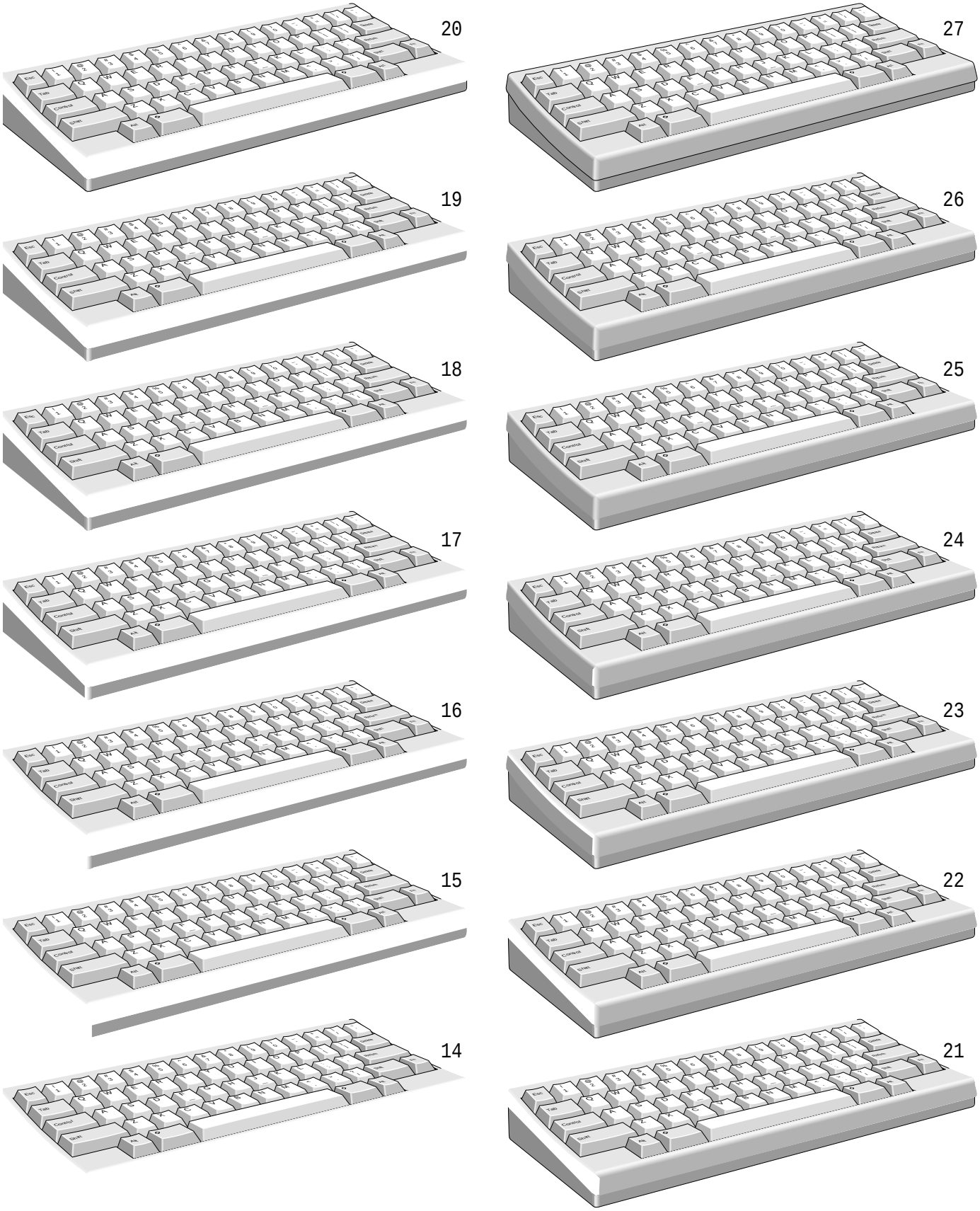

図-5 見取り図の組み立て-1

24. 左面, 上面の交差部の曲面を塗る
25. 正面, 左面の交差部の曲面を塗る
26. 上面, 正面, 左面の交差部の曲面を塗る
27. 上部ケースの輪郭を描く

27 番までで図-1 と同じものが完成する. なお曲面は gray の度合いを変えながら, 線を何本も並べて描いている.

4　キートップ

個々のキートップの描き方はキーの基本形状を決め, それを然るべき位置に然るべき傾きで置いて投影描図するだけである. まず全体を眺めると図-6 のようになる. スペースバー (これは中央が凹んでいない) 以外は長さには 1 単位のものから 2.25 単位のものまである. 長いものは中央部分を引き延ばして描くだけである.
6 単位 (space 1 個)
2.25 単位 (左 shift, return 2 個)
1.75 単位 (control, 右 shift 2 個)
1.5 単位 (tab,delete, 左 meta, 右 meta 4 個)
1 単位 (上記以外 51 個)
単に台形を描いてもよさそうだが, よく見るとキートップの左右の線は中ほどが凹んでいるので, そこは多少垂れ下がるように描くことにした.

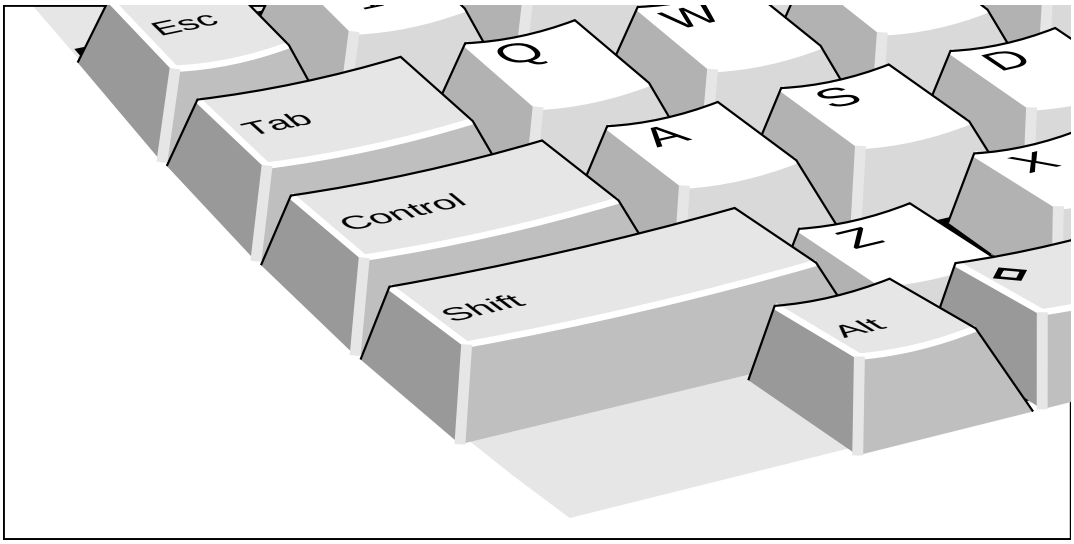

図-6 キートップ (10 番まで終わったとき, Z は押し下げ位置にある)

この図で左端に並ぶ機能キーの輪郭を見ると, シリンドリカルスカルプチャになっているのが分かる. 手前の Alt のキーをよく見ると, 右手前の斜線が特に長く, キーが左に倒れているようにも感じるが, 実際のキーボードを横からみるとそれもたしかにそのように見えるから, 座標変換がおかしいわけではない. 一種の錯覚であろう.
キートップの文字の描き方を図-7 左に示す. 菱形のようなのが, 投影されたキートップである. この投影面上の隅の座標を (x_0,y_0),(x_1,y_1),(x_2,y_2) とする. その時

```
/x0 40 def /y0 10 def
/x1 80 def /y1 20 def
```

```
/x2 10 def /y2 40 def
x0 y0 moveto x1 y1 lineto -30 30 rlineto x2 y2 lineto closepath stroke
gsave
[x1 x0 sub 40 div %[a
 y1 y0 sub 40 div % b
 x2 x0 sub 40 div % c
 y2 y0 sub 40 div % d
 x0 y0] concat % tx, ty] で CTM を作る
/Helvetica findfont 30 scalefont setfont
20 (A) stringwidth pop 2 div sub 10 moveto (A) show
grestore
```

のように CTM(current transformation matrix) を concat で置き換え, そこに文字を書くと, 図-7 左のように, ひしゃげたキートップに合った文字が現れる.

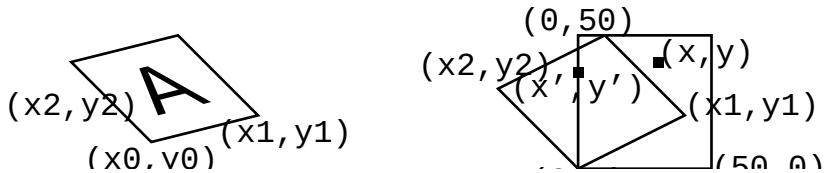

図-7 キートップに文字を書く.

この変換の理由を図-7 右で説明する. 正方形の領域 (0,0) (50,0) (0,50) (50,50) がひしゃげて (50,0) が (x_1, y_1) に, (0,50) が (x_2, y_2) に, 一般に (x, y) が (x', y') に移ったとする. この変換は一次変換なので,

$$x' = ax + by, \quad y' = cx + dy$$

と書ける. これに $x_1 = 50a + 0b, y_1 = 50c + 0d, x_2 = 0a + 50b, y_2 = 0c + 50d$ と既知の 2 点をいれて a, b, c, d を解く. Postscript の CTM はこうして出来た a, b, c, d と原点の移動距離 t_x, t_y の 6 要素で出来ているので, それを計算して利用しているだけである.

5 アニメの原画

前述のように, キーの描画の際, ある文字のキー位置を深さ方向に押し下げ距離 (3.8mm) だけ移動しておけば, 図-6 に一部を見るような, そのキーを押し下げた図が得られる. これを 60 個のキーすべてについて用意した. Postscript の白黒のイメージは 8 ビットの画素の列である. そこであるキーの押し下げの図とどれも押し下げられていない標準図を取り出し, ビットごとの排他的論理和をとる. これは押し下げられたキーに関係する領域だけで 1 になるから, それを囲む最小の長方形の領域 (起点とサイズ) が分かる. これを各キーについて計算し, 押し下げ状態と正常状態のその領域を切り出した.
この切り出しのプログラムは lisp で書いてある. 図-8 にキー H の図 (左が押し下げ位置 右が正常位置) を示す.

図-8 アプレットの画素

切り出した各領域を図示すると図-9 のようになる. 全体に右上がりになっているので, 幅の広いキーの領域は横だけでなく, 縦も大きい. 当然スペースバーは広い場所を取っている.

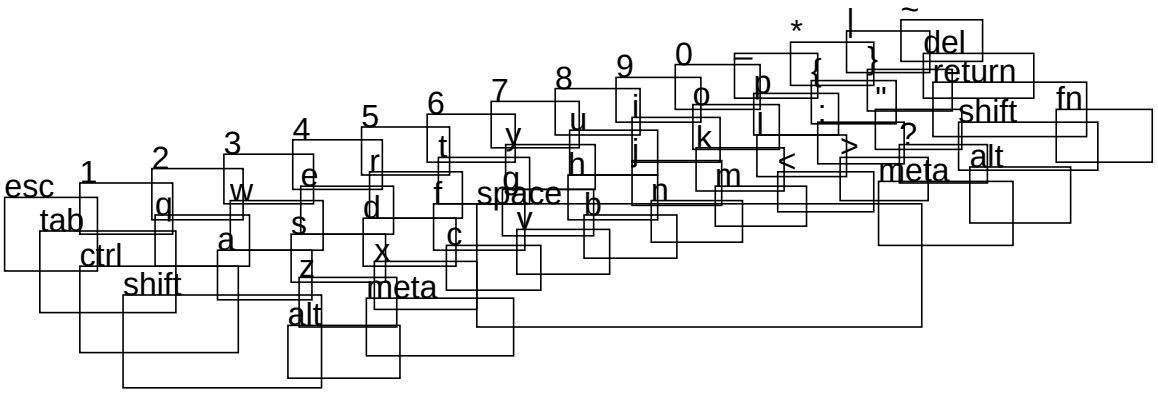

図-9 キーの差分の領域

PFU のホームページにあるアプレットは全体画像を表示し, 一定の時間毎に各キーの押し下げの図と正常の図を然るべき位置に表示するものである. シフトキーの場合は押し下げた後, シフトの続く限りそのまま表示し, シフトが終わると正常位置にもどす. この時の問題は例えば左シフトを押しながら A を押そうとすると, A の押し下げの図は左シフトは正常位置のものしかないので, うまく表示できない. そこはタイプの教科書にあるように, シフトされる文字が左にあるときは右シフトを押し, 右にあるときは左シフトを押す. そうすれば押し下げの図が重なることはない. PFU のアニメでは Happy Hacking Keyboard と繰り返すだけであった. H も K もシフトキーとは重ならないことが図-9 から確認でき, どちらのシフトキーを使おうと自由であった. (この問題が起きるのは, 図-6, 図-7 から分かるように A と Z である.)
他のキーと同時に押すものにはコントロールもあり, これは左にしかないからシフトのような解法はない. しかしこれはデモにもありそうでもないから, 深入りはしていない.
私はこのアニメの画素とその挿入位置を用意しただけで, アニメ化は PUF の白神さんにお願いした. PFU ではその後クイズなどに利用していた時期もある. いまは会社名を印字しているらしい.
Javascript で書き直した最新版に関してはデモで紹介する.

6 質疑応答

- Q(近山):一般の 3 次元モデルを作り, それで描くことは考えなかったか.
  A:一回切りの作業だし, 段々に作っていったので, そうはなっていない. むしろ描こうとする気持が入っている.
- Q(粕川):キーを置く位置を黒く塗っているが, どこかに現れているか.
  A: 左上, 右下に僅かに見える. キーを押したときに見えるかもしれない.
- Q(不明):キートップの文字は正確か.
  A: 図-7 の程度に相当いい加減である. 中央が凹んでいることもない.
- Q(宮原):知人にアルミの HHKB を作ったのがいる.
  A: ウェブページに出ていたので, 承知している.

参考文献

[1] 和田英一: けん盤配列にも大いなる関心を, PFU Technical Review, Vol.3, No.1,pp.1-15 (Feb 1992).

[2] 和田英一: 個人用小型キーボードへの長い道, bit,Vol.27,No.5,pp.4-12 1997 年 5 月.