Lime '90

K.I.T. COMPUTER CLUB

目　次

プロジェクト　CGA90

浜岡　裕樹

「CG（コンピューター・グラフィクス）でアニメを製作しよう」というのが、そもそもの動機でした。もちろん、コンピューター上でアニメをするなら、市販のソフトなど、出来あいのツールを使えば済むことですが、それではコンピューター部の意義に反する、必要なものは全て自分達で用意しよう。この考えのもとにCGA90は開始されました。

○スタッフ
- 近藤政雄
- 浜岡裕樹

○目的

パーソナル・コンピューター上でCG（表示原理はレイトレーシング）を用いたアニメーションを製作する。

○ハード構成
- PC－9801＋デジタルディスプレイ（8色）……物体の配置・動画の計算等
- ハードディスクドライブ（100MB程度）……計算させた動画データーの保存
- MSX2……画像のビデオ出力に用いる
- ビデオデッキ……コマ撮りに使用

○テーマ

今回は「パックマン」をCGで表現してみました。つまり、パックマンとモンスターの3D迷路でのチェイスをアニメで再現するのです。

○製作したツール

今回、CGA（CGアニメーション）を製作するに当たって、用意したツールは以下の通りです。但しMSX2上のものを除き、全てCで記述しコンパイルしました。

- MAP．EXE……各々の時刻における物体の配置をデーター化する。
- ANGLE．EXE……視点の軌跡をデーター化する。
- CGOUT．EXE……上記のデーターから1枚1枚の動画を生成する。
- G．EXE……MSX2に画像データーをRS－232C介して送信する。

etc.

各ツールで具体的にどのようなことをしているかというと、

1）MAP．EXE

パックマンの動きを基本単位として（ドットからドットまで＝１ステップ）、ステップ単位で、物体の配置データーをファイル（MAP．○○○、○○○＝０～９９９）に落してやります。なおこの際に、パックマンの動きはキーボード入力でトレースしてやり、モンスター達をパックマンに合わせて動かします。この時のパックマンの動きは、キーストロークデーターとして、保存・再生などが出来ます。

2）ANGLE．EXE

上記のMAPファイルから視点の位置・向きを計算して、AGLファイルと呼ばれるものを作成してやります。このファイルはテキスト形式で、１行に動画１枚の視点のデーターが記述されており、後述するCGOUT．EXEのパラメータの形式になっています。なお、このツールは今回のアニメ用に特に用意したもので、実際のアニメ製作における利用範囲はかなり限定されたものになっています。具体的にどのようなデーターを生成しているかというと、

・前半部分……パックマンから見た状態（の視点）
・中盤部分……モンスターから見た状態（の視点）
・後半部分……パックマンの上空を一定周期で旋回するような視点

となります。

3）CGOUT．EXE

今回のツール群の中でも中核に当たる部分で、動画データーを実際に計算しています。データーとして、MAPファイル・AGLファイル・PTN．Jを必要としますが、MAPファイルをそのまま動画１枚１枚に対応させると、動きが粗くなってしまい、滑らかさに欠けるので、MAPファイル中隣接するファイルの間で補完してやり（今回はファイル間を４枚に補完しています。）、その補完したデーターを用いています。CGの計算に用いているアルゴリズムは、レイ・トレーシングと呼ばれるもので、日本語に約すると、「光線（視線）追跡法」となり、近頃、映像表現の効果的な技法としてよく用いられています。詳細は前年度のｌｉｍｅを参照してください。

4）G．EXE

計算された画像データーをRS－２３２Cクロスケーブルを用いて送信してやります。

○技術的な説明

ここでは、CGAを製作するにあたって、問題とされたことを特に技術的な面に限って述べます。

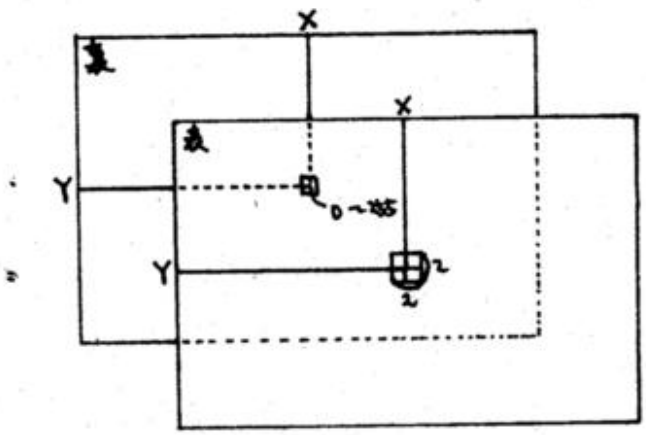
・レイトレーシング
レイトレーシングによって動画を計算する際に、問題となる点が2つあります。

1）データーをどのように表現するか
2）計算速度をいかにしてあげるか

まず、1）についてですが、動画は最終的にはＭＳＸ２上で２５６色の色で表現するのですが、ＰＣ－９８０１上では通常で最高１６色（今回のシステムでは8色）しか表現できず、それ以上の色を表現しようとすれば高価なフレームバッファを導入せねばならず、費用の点で問題となります。そこで、ＰＣ－９８０１上では６４０×４００ピクセルのグラフィック画面を２×２ピクセルを１単位として、３２０×３２０ピクセル２５６色のグラフィック環境を実現しています。さらに、ＰＣ－９８０１はグラフィック画面を２枚持っていますので、裏の画面を色データー（０～２５５）の保存用のバッファに使い、実際に、ＭＳＸ２へデーターを転送するときは、この裏画面のデーターを送ってやります。（図１）

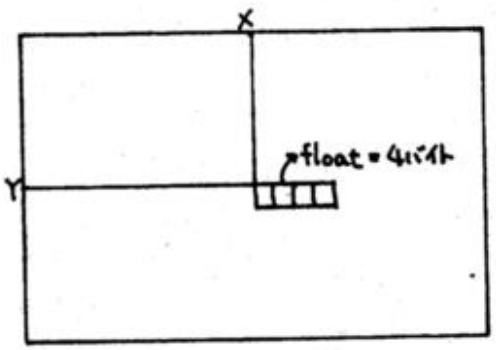
あと、今回のレイレーシングの原理上、１ピクセルごとに視点から物体までの距離を保存しておくバッファが必要なので、これをメインメモリ上にfloat のピクセル分の配列として確保しています。（図２）

さらに、物体のデーターをどのように表現するかという問題が残っていますが、これは純粋にレイトレーシング理論の問題なので、ここでは割愛させていただきます。



↑図1　VRAMの利用法

↓図2 ピクセルのバッファ



次に、2）について、これはレイトレーシングをやる上で必ず直面する問題で、長時間化する計算時間をいかに短縮化するかによって、全体の効率が決定します。そもそも、レイレーシングに時間がかかるかということは、大雑把にいうと、1ピクセルごとに視線と物体の交差を（物体ごとに）判定してやるために、結果として1ピクセルにつき、最低1回の実数演算を全ピクセルに対して行うことになり、演算総量は動画1枚につき、数万回から数十万回におよびます。コンピューターは実数演算を行うことが基本的に苦手なので（人間よりは遥かに高速に行ってくれますが）、1枚の動画完成するのに、数時間から2・3日、下手をすると1週間もかかってしまいます。

これではたまらないと、計算の高速化を計ることになるのですが、今回は以下の方針に基づいて行いました。

i）表示する物体を出来るだけ単純化する。
ii）無駄な計算をしない。

i）については、CGのテーマに「パックマン」を選んだことが、この方針を如実に物語っており、今回作品中には球と円柱しか登場しません。そもそも、アニメで複雑な物体を見たままに表現する必要はなく、出来るだけ単純な物体で構成すれば、充分に鑑賞に耐え得る作品が出来るのです。

ii）については、主に物体のクリッピングによって解決しています。クリッピングというのは、画面に表示されない部分の計算を省いてしまう処理のことなのですが、さらに1つ1つの物体について明らかに関係しないピクセルへの計算を省く処理も含めて行っています。

具体的にどのようにクリッピングを行っているかというと、まず、視界内に目的の物体が入っているかどうかを判定し、おおむね入っているものについてのみ、大雑把にスクリーンへその輪郭を投影し、その輪郭内のピクセルだけを走査するというふうにしています。（図3）

しかし、この処理にも問題点があります。クリッピングの際の処理を出来るだけ大雑把に行っているため、物体の表示が必ずしも正確に行われていないのです。具体的には、視点付近に存在する物体が急激に消滅したり、正しい形で表示されていなかったりしています。もちろんこれを回避することはできますが、それでは、本来のクリッピングによる処理の高速化が失われてしまうことになり、この辺りの兼合いが難しいものとなっています。

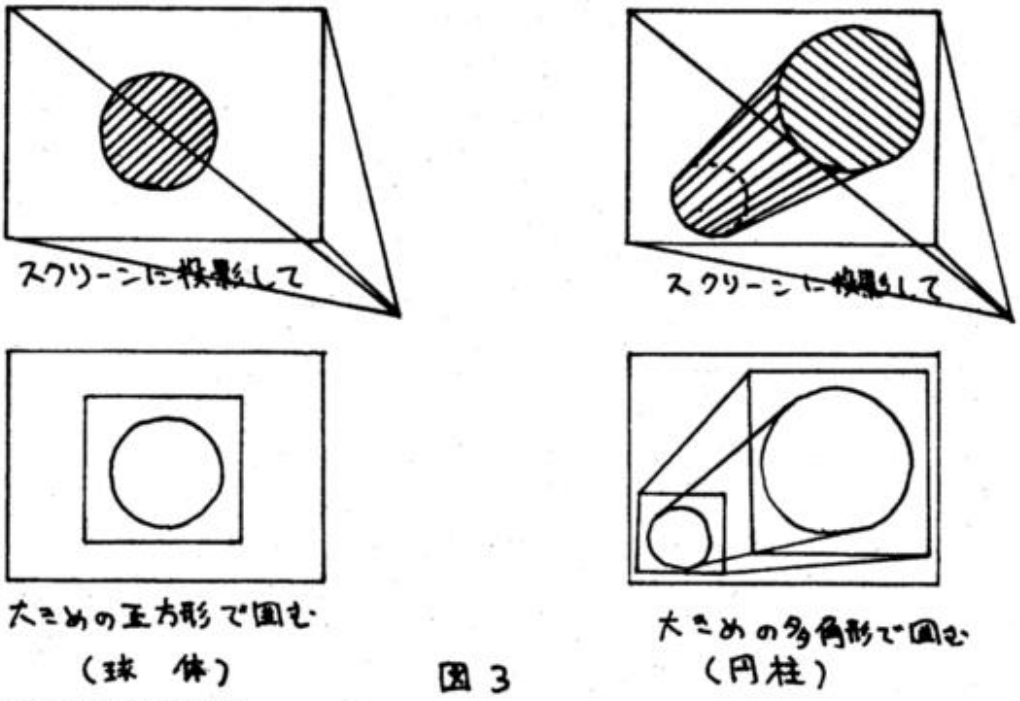


図３

○ＣＧＡ製作の実際

実際にＣＧＡを製作するに際して、以下のようなことがありました。

・いかに１画面当たりにかかる計算時間が短縮されたとは言え、１枚当たりにかかる計算時間は１０分はかかるので、かかる計算時間の総量は１００時間にもおよびます。で、これを回避するのに複数の９８を投入しました。（総数９台）

・コマ撮りに市販のビデオデッキを用いたために、１コマ間のスピードを細かく調整することができないので、出来上がりが少しぎくしゃくしてしまいました。

○製作を終わってみて

実質的な製作期間が短かったため、結局利用範囲の広いツールを作ることができませんでした。ただ、本来時間がかかるレイトレーシングを計算時間を飛躍的に短縮することができ、ＣＧに応用することができたことは、充分に価値のあることでしょう。これからの課題としては、より複雑な物体のレイトレーシングによる表現・ポリゴンによるＣＧの表現などがあるでしょう。

○ソフトの配布

今回作ったツール群を配布する予定です。詳細は同梱のドキュメントを参照して下さい。

○ＣＧＡの製作協力者

以下の方にはＣＧＡを製作するに当たって、多大なご協力賜わりました。最後にはなりましたが、この場をお借りして感謝の意を述べさせて頂きます。（以下敬称略）

Special thanks to...

○小長直治（アダモちゃん）	inアメ研	ＰＣ－２８６ＶＳ
○飯田晃紀（いいだだ）		ＰＣ－２８６ＶＳ＋ＤＡＳＨ９０Ｓ
○京本　猛（たける様）	inアメ研	ＢＧＭ作曲
○下迫　勇（さこちゃん）	inアメ研	ＢＧＭ録音、ＰＣ－９８０１ＲＸ
○伊藤秀史（いたうさん）	in二の丸荘	シベリアＳＴＵＤＩＯ　in　二の丸荘
○河口哲夫（かわぐちくん）	in二の丸荘	ＰＣ－２８６ＶＥ
○嶋田　順（ずんちゃん）	in二の丸荘	ＰＣ－３８６ｍ
○辻　邦彦（つじくん）	in二の丸荘	ＰＣ－２８６ＶＦ

（コンピューター部部員）

○近藤政雄	ＰＣ－２８６ＶＳ
○橋本圭介	ＰＣ－９８０１ＶＸ
○湖山文道	ＰＣ－２８６ＶＧ
○コンピューター部	ＰＣ－９８０１Ｖｍ２１

and others...

○最後の最後

今まであたかも私がＣＧＡを全般に渡って、やったようにかいてきましたが実はたいしたことはやっていないのでした。（ＭＡＰ．ＥＸＥを作ったのと、企画に参加したぐらいですね）。ひとえに近藤先輩の努力と協力者の皆様の暖かいご協力のおかげなのです。

さらに、この原稿も少々予備知識がないと何が書いてあるのかまるで判らないものになってしまいました。文書書述の才のなさを痛感する次第です。

とにかく、今回の経験を叩き台にして（というにはかなり大きな台ですけど）、来年の企画に活したいものです。

MP-80

エムピー はちじゅう

7.

ACT I　　プロローグ

file name is LIME.TXT

～～～ とある日の出来事 ～～～

とある日の夕暮れ時に、コンピュータ部に立ち寄った者がいた。その者はおもむろにコンセントを差し、御本尊（VM）をたち上げた。そしてMP－80の電源を入れ、あるファイルをプリントアウトをしようとするのだった。

MS－DOSでそのファイルをPRNへCOPYをすると、MP－80は元気よく動き出したのだった。その者は喜んだ。というのは、MP－80はケーブルの接触不良で未知の言葉を喋ることが多々あったからである。MP－80は順調にプリントアウトしていた。ところが、突然ドットを打ち始めたのである。それも意味不明のものである。ああ、MP－80は未知の世界へトリップしてしまった。その者は原因を究明し、発見した。打ち出そうとしたファイルは漢字が入っていたのだ。もちろん、MP－80は漢字がでない。

その者は漢字をプリントアウトしたかった、どうしてもしたかった。いや、したくなった。よってその瞬間からMP－80漢字出力への戦いが始まったのだった。

（出力例）

```
A>mp80
file name?1f5.bat
cheak
Cannot Open 1f5.bat
```

（使用例）

ACT II　　プロフィール

○ MP-80 の構成内容

- MP80.C
- PR_TAB.C
- PR_WIDE.C
- PRBIOS.H
- PRBIOS.C
- PRCODE.H

○ 使用方法

おもむろに
MP80とタイプする。

(例) 前ページ下)

○ 特長
- EPSON MP-80 で漢字も出力できる。

ACT III　　プログラム

MP80.C

```
int sjtoj( int k )
int getptn( unsigned int code ,BYTE ptn[] )
void cl_line()
int mp_outline( BYTE buf[] )
int mp_mkline( int h, unsigned int kstok[],
hit_key(){                              int j )
main( int argc, char *argv[] )
```

PR_TAB.C

```
void mp_taber( unsigned char line[300],
                                int tab_pich )
```

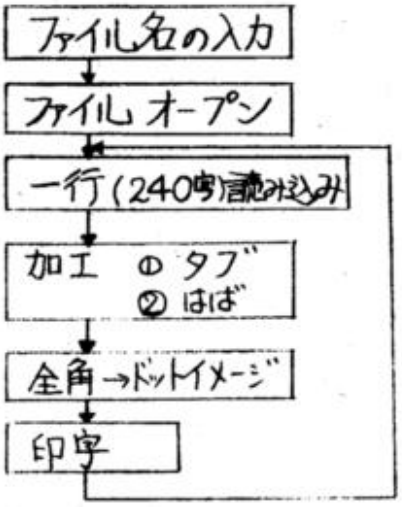
PR_WIDE.C

```
int mp_wide( BYTE buf[256] )
```

PRBIOS.C

```
int  pr_info();
int  pr_out(BYTE data);
void pr_outs(char *s);
int  pr_bios_int(BYTE ah,BYTE al);
void pr_init();
```

MP80 の流れ.



詳しくは
ソースファイルを
見て下さい。

ACT IV　　作成中のこと

○ 漢字のドットイメージによる印字方法について

MP-80は一行8ドット列で印字するので、16×16ドットの全角文字を印字するためには一行を2重印字しなければならない。(ドット出力は倍密度にて行う).

方法　1. フォントパターンをCRT BIOSより読み取る.

2. 奇数行と偶数行とに分け、それぞれのデータをline1[]、line2[]に格納する。

(注) line1[]は半角文字と全角文字の奇数行ドットを.
line2[]は全角文字の偶数行ドットで格納する。
初期化はline1[]は¥00 line2[]はスペースで行う。

3. line1[]を印字する。

4. 1/216 inch 紙送りを行なう。(ESC J 0x1)

5. line2[]を印字する。　(終)

○ プリンタBIOSによる出力への変更の原因について

最初はプリンタへの出力を fprintf(stdprn,"~");で行っていたが正しく印字させることができなかった。MS-DOS上で出力するためASCIIコードで80以上等のコードを出力すると漢字であると判断してしまうからだった。(DOSは漢字プリンタだと思っている。) 解決方法はDOSを通さず直接プリンタに出力することである。よってプリンタBIOSを使用することになり、PRBIOS.Cというファイルが作られることになった。

○ MP80による一行印字最大文字数と改行について

Ans文字の場合はプリンタの一行印字最大文字数になると自動的に改行するが、ドットイメージの場合は超過分を削除する。だから一行出力文字数をソフトにより調整しなければならない。又Ans文字と全角文字との幅の比が6:8であるので一行におけるそれぞれの割合により一行文字数を変化させねばならない。

方法 1. Ans文字は6ドット分、全角文字は8ドット分として、合計一行につき320ドット以内になるように一行の文字数を決定する。

2. 一行出力文字列を作成する。

関数 mp_wide()

終

冬といえば
年の終り
毛糸の季節

○ ESCキーでとまるようにしたわけ

MP80もほぼ完成し、テストプリントアウトをしていたころ。プリントアウト中はSTOPできないことがわかった。よってESCキーにより印字を止めることができるようにした。最初は出力時のBUSYのときにキーONのチェックをするようにして、KYBIOS.Cというファイル（キーボードBIOS）が作成されたが、kbhit()により一行出力後に検出する改良がなされ要を消すことになった。

ACT V　　課題

ここではMP80の今後の発展を願って改良すべき点、を述べる。

○ 出力ファイル名の入力方法の多様化

○ プリンタの各種機能を操作可能にする（LF. FF等）
これはコマンド入力により操作できるようにしたい。
HELP機能も必要になるだろう。

○ ミシン目スキップをさせる。

○ 出力ファイル名の印字の有無、

○ デバイスドライバ化

○ 一行印字最大文字数設定可能化、等.

ACT VI　　エピローグ

MP-80が他のプリンタにとってかわられるときは近いかもしれないが、それまではMP-80をとことん使ってあげたいものである。

あまり

あまり

あまりの頁

MIDI 超入門

Ou Runner

・MIDI とな？

MIDI とは Music Instrument Digital Interface の略で、電子楽器を接続するための世界統一規格であり現在のほとんどの電子楽器に装備されています。言って見れば楽器界の MS-DOS フォーマットといった所です。こいつを使って楽器と手持ちのパソコンを接続してやれば、コンピュータによってその楽器のほとんどすべてをコントロールすることができるようになります。(つまり、コンピュータをシーケンサとして使えるということ。) 多くの楽器を MIDI で接続することで、1人オーケストラや、1人バンドが簡単に楽しめます。また、パソコンの内蔵音源 (FM, SSG, etc.) にものたりない人も、MIDI によって、より質の高い音源を使えるようになります。内蔵音源と外部音源の同期を取ることによって、FM音源では力不足の音 (打楽器、オケヒット, etc) は、外部音源で、他の音は FM で、といったような使い方もできます。FM音源は打楽器に弱いと嘆いていた M君や、ハイハットやシンバルがスカスカだと悩んでいた S君もこれで

一気にストレス解消だ！

○MIDIの実際

MIDIを使って自分で楽器をコントロールするとき，最初にひっかかるのが，各楽器が各データに対してどんな動作をするのかということがわからないことです。MIDIを装備している楽器のマニュアルには必ずMIDIインプリメンテーションという詳細な表が付いているのですがこいつがくせ物で，その表の見方や実際の扱いなどが，たいていは全く書いていないのです。これではMIDIの素人が「さあMIDIで楽器をコントロールするぞ」と思っても，マニュアルをしばらくながめた後「どうやったらいいのかぜんぜんわかんねえよ！」と，途方にくれ，大抵はたいてミュージ郎を買うハメになってしまいがちです。しかしインプリメンテーションの見方さえわかればその楽器の性能をフルに引き出すことができるはずなのです。そこにはFM音源には無かった おいしい機能が山ほど（ちょっとオーバー）つまっているのです。インプリメンテーションには，

1) インプリメンテーションチャート

2) データフォーマットテーブル

の2つの表があるのですが、1)がわかれば、たいがいのことができるようになるので、誌面の都合上（原稿のメ切の都合上）ここでは1)について説明したいと思います。例として、下にRolandのMT-32のインプリメンテーションチャートを示します。

MT-32 MIDI インプリメンテーションチャート

ファンクション	送信	受信	備考
ベーシック 電源ON時		1－16	電源オフの状態も記憶されます。
チャンネル 設定可能		1－16	
モード 電源ON時		モード3	
モード メッセージ			
モード 代用	********		
ノート ナンバー 音域	* 0－127	0－127	
ノート ナンバー 音域	******	12－108	
ベロシティー ノートオン	*	○ V=1～127	
ベロシティー ノートオフ	*	×	
アフター タッチ キー別	*	×	
アフター タッチ チャンネル別	*	×	
ピッチベンダー	*	○（半音単位で0-24）	
コントロールチェンジ 1	*	○	モジュレーション
コントロールチェンジ 7	*	○	パートボリューム
コントロールチェンジ 10	*	○	パンポット
コントロールチェンジ 11	*	○	エクスプレッション
コントロールチェンジ 12	*	×	
コントロールチェンジ ：	*	×	
コントロールチェンジ 64	*	○	ホールド1
コントロールチェンジ ：	*	×	
コントロールチェンジ 121	*	○	リセットオールコントローラーズ
プログラムチェンジ 設定可能範囲	*	○ 0～127	
プログラムチェンジ 設定可能範囲		○ 0－127	
エクスクルーシブ	○×	○	
コモン ソングポジション	×	×	
コモン ソングセレクト	×	×	
コモン チューン	×	×	

リアル タイム	クロック コマンド	X X	X X	
その他	ローカル ON/OFF オールノートオフ アクティブセンシング リセット	Y X X X	Y O (123-127) O Y	
備考		*オーバーフローモードでは受信したメッセージはMIDIアウトからそり出される。		

モード1：オムニオン，ポリ　モード2：オムニオン，モノ　O：あり
モード3：オムニオフ，ポリ　モード4：オムニオフ，モノ　X：なし

○各項目の詳細

1. ベーシックチャンネル

楽器などを電源オンしたときに送受信できるMIDIチャンネルを表している。MIDI規格ではCh.1～16まであるがMT-32では1～10までしかサポートしていないので送ったDATAのチャンネルが11以上だと音がまったくからないので気を付けよう。MT-32では電源オン時にはch2-9にメロディパートとCh.10にリズムパートが必ず割り当てられる。

2. モード

MT-32を使う場合は常にモード3（オムニモードオン，ポリモード）なのであまり気にしなくていい。オムニモードオンのとき，すべてのMIDIチャンネルへのメッセージが受け付けられるがオフにすると あらかじめ指定

したチャンネルのメッセージのみ処理する。ポリ/モノは単音か和音かということ。

3. ノートナンバー

音程の演奏情報を送受信するときに使う。規格では音程は真ん中のド(C3)を $3C_H(60)$ として上下半音ずつ順番に1音ずつ、データが割り当てられている。ノートナンバー欄の上の数字は送受信できるデータ範囲、下は実際に発音される音域を表します。

4. ベロシティ

音の強弱を数値で表したもの

5. アフタータッチ

アフタータッチというのはキーボード特有の機能のひとつで、キーを押さえたあとさらにそのキーを強く押さえることで、弾いている音にビブラートをかけたり、音色を明るくしたり、ベロシティを変えたりすることができます。

6. ピッチベンダー

鳴っている音の音程(ピッチ)の上下を連続的に変化させる機能です。(ギターなどでよくやっているチョーキングというやつです)

7. コントロールチェンジ

各機種が個別に持つ機能をサポートするもの。

8. プログラムチェンジ

MIDIでは音色データのことをプログラムと言います。つまり音色切換ということ。

9. エクスクルーシブ

音色データ自身をMIDIコードを使って転送するものがエクスクルーシブメッセージと呼ばれ，表ではそれに対応しているかどうかが，示されている。自分で音色を作らない人にはあまり必要ない。

10. コモン

ソングポジションはシーケンサやリズムマシンで現在何小節目の第何拍にいるかを表し，ソングセレクトは何曲目のデータかを指定するもの。またチューンは音程を合わせるのに使う。

11. リアルタイム

省略

12. その他

省略

MIDIで送るデータの具体的なデータはデータフォーマットテーブルに示されているので，そちらを参照すれば以上の説明でだいたいのことはできるようになります。さらに深くハマりたい人はMIDIの入門書を買ってください。ではあなたにも良いMIDIライフを！

マイクロマウス教育日記（ソフト編）

ANKO

4月

さてことしもマイクロマウスを動かそうと思い、今年の改造はどうしようかと悩みはじめる。

5月

マウスは本当に動くのかどうか確かめたくて、部室内にマウスの迷路を入れて、マウスの実験をする。動作したが、同じプログラムなのに去年と動作が異なるような気がする。

6月

マウスはよく電池を喰うので、１Ａの専用電源をあたえてやる。さらにマウスの外形を去年からのアイデアである、円形に改造する。しばらくはこの改装のため時間を費やし、ほかのことをする暇がない。

7月

暇があると計算機センターに行っているのでマウスはほったらかしである。

8月

計算機センターにくるついでに部室によってマウスの御機嫌を伺う。

9月

マウスよりも自分の成績が大変。本格的にアイデアが出てくる。

10月

マウスのプログラミングの最中に専用電源は電圧が安定していないことに気づく。しかしプログラミングにはあまり関係がない。が、センサは電圧が不安定だと機嫌が悪くなる。

マウスの実験用テストコース”箱庭”（制作：う）ができる。さらに調整は進む。

11月

マウスの右折・左折ルーチンの完成を見る。

「去年はセンサを増設し基本的なソフトウェアを開発したが、右折・左折がうまくできず未完成であった。そこで、ことしは新しく右折・左折のルーチンを開発した。

コンピュータ部のマイクロマウスは駆動系が普通のＤＣモータのため９０°をモータの回転数などで調べることができない。したがって、センサだけが頼りなのである。

今までのルーチンでは、前方に壁があったときに左折する場合、まずその場で左旋回をする。そして右センサを基準にして、前壁をセンスして回転していく。この方法では、最初右や左に片寄っている場合右のセンサ位置の正確さがあまりない。このため左旋回の終了点を感知しなかったりする。

そこで今年は、左旋回をする場合にループによってある程度旋回させてやり、その後左センサによって旋回の終了点を認識させる。この方法によって左センサ位置が左通路の間に入っていることが確定する。それを利用して、左壁、もしくはいままで進んできた道の左壁の端点を左センサ３つでチェックする。

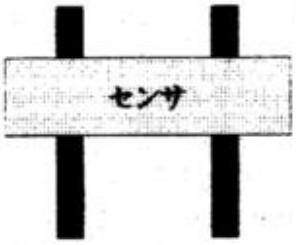
そうすれば、煩わしい前壁にセンサがかかる場合のチェックが不必要となり、後は左センサがセンスできなかった場合を考えるだけで済む。

このような場合は、右センサの中央で、左折後の右壁に整合したかどうかのチェックをつけておけば万全である。

ほぼこれで左折・右折は９０％くらいのできになった。」

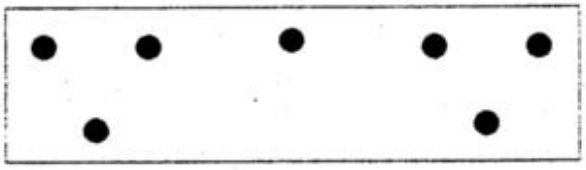
ということで、マウスの教育は施されるのだが、意志の疎通がうまく行かないようだ。これ以降のできは、学祭中のマウスの動きにてご覧ください。

マウスの動き

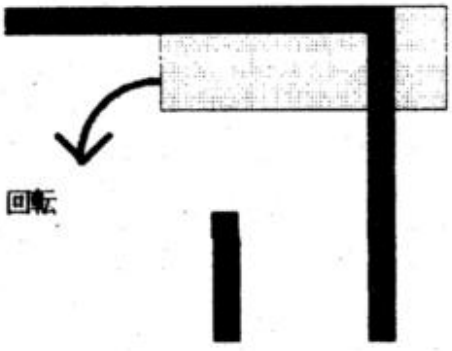


直進状態

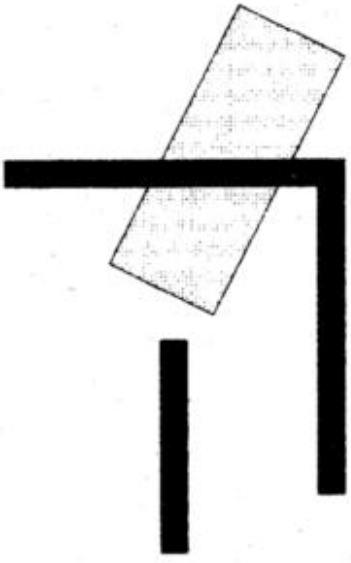
センサの状態 ○●○○○●○



センサ取り付け位置



左折（ループ前）



旋回ループ終了後

Ｗｉｔｈ　Ｘ－Ｔｏｏｌｋｉｔ

ａｎｋｏ

わたしは、今年の６月くらいから半年ほどＵＮＩＸ上にて、Ｘウィンドゥのプログラムを開発していた。その開発にあたって使用したものが、このＸ－Ｔｏｏｌｋｉｔである。このライブラリによって、わたしは多大なる影響を受けた。その一部をここに表そうと思う。

なお、実行環境は、
WS: NEC EWS4800
X-WINDOW V11 R3
Athena Widget set

Ｘ－Ｔｏｏｌｋｉｔは、Ｗｉｄｇｅｔというのもを基本とする。Ｗｉｄｇｅｔとは、ウィンドウとそれ付随するもののデータを含んだ型であると考えるとよい。

Ｗｉｄｇｅｔには、いくつかの種類がある。ウィンドゥを作るだけのものから、文字列をウィンドゥ内に表示するような高級なものまである。このような種類一つ一つをＷｉｄｇｅｔＣｌａｓｓといい、Ｗｉｄｇｅｔを作成するときにこれを指定する。

ＷｉｄｇｅｔＣｌａｓｓには、大きく分けて子Widgetを持てるComposite と子Widget を持てないPrimitiveに分かれる。ウィンドゥ内に文字を表示するなど表現力のあるものは、Primitive が多い。Composite は主に、子Widget の制御が多い。

まず、簡単なサンプルプログラムを示す。

```
/*  Label Widget  */

#include <X11/Intrinsic.h>
#include <X11/Label.h>

main(argc, argv)
int argc;
char *argv[];
{
  Widget toplevel, child;

  toplevel = XtInitialize("sample", "Test",
                          NULL, 0, &argc, argv);

  child = XtCreateManagedWidget(" Sample Widget ",
                                labelWidgetClass,
                                toplevel, NULL, 0);

  XtRealizeWidget(toplevel);
  XtMainLoop();

}
```

このプログラムは、簡単なメッセージを表示するウィンドゥを作成する。

まず最初に toplevel は、全ての親Ｗｉｄｇｅｔとなるもので、シェルウィジェットとよばれ、これを基準にしてほかのＷｉｄｇｅｔはつくられる。

このプログラムの場合、メッセージを表示しているのは、child がそのデータを持っており、labelWidgetClass はPrimitive なので、親Widget にはなれない。そのため、toplevel が、親Ｗｉｄｇｅｔとなっている。

XtInitialize() は、Ｘ－Ｔｏｏｌｋｉｔの初期化と同時に、シェルウィジェット（toplevel）をつくっている。

XtCreateManagedWidget() は、WidgetClass と 親Widget を指定してＷｉｄｇｅｔを作っている。ここで childが、labelWidgetClass であり、表示文字列は " Sample Widget "で、親Ｗｉｄｇｅｔは toplevel であることを示し、child を作成している。

XtRealizeWidget() は、toplevel とその全ての子Ｗｉｄｇｅｔを画面に表示する。

XtMainLoop() はイベントがおきるまで無限ループする関数である。

コンパイルは、

```
cc -lXaw -lXmu -lXt -lX11
```

で可能である。（日本語化は別ライブラリ）

さて、Ｗｉｄｇｅｔ の良い所は、オプションの指定の方法にある。例えば、ウィンドゥの大きさを変えたい場合は、

```
static Arg args[]={
  XtNwidth,400},
  XtNheight,100}
};

toplevel = XtInitialize("sample","Test",NULL,0,&
                        argc,argv);

child = XtCreateManagedWidget(" Sample Widget ",
                              labelWidgetClass,
                              toplevel,
                              args,
                              XtNumber(args));
```

とすれば、 child は４００×１００ピクセルの大きさのウィンドゥとなる。

XtNumber() は、配列の要素数を返す。このとき、 args の配列内のオプションの順番は任意であり、デフォルト値を変更するものだけ書き並べれば良い。

このように、オプションの指定がかなり楽に行える。さらに素晴らしいのは、XtNwidth などは全てマクロで書かれた文字列であり、これの様に定義することによってエラーを減らすこともできる。

最後に、私はこのライブラリを使用してからは完全にトップダウンでプログラムを作るようになった。完成されたライブラリを使えば、ボトムアップで作る必要はないと私は思う。全ては完成されたライブラリの有無にあり、それによってプログラミングスタイルは変化するのだと思う。

参考文献

X Toolkit プログラミング
トッパン
Computer Today 1990/9
サイエンス社
EWS－UX／Vシリーズ
Xツールキット説明書
NEC

編　集　後　記

今回 ライムの編集をまかせられて、
'あっ!!'と言う間に、"うおおおお。
明日、もう学祭かぁ。"という様
な時期になって、あせり、あせって、
やっと出来上がりました。いやー、
本当に、手伝ってくれたみなさん
のおかげです。ありがとうござ
いました。今後もライムをよろし
くかぁいがってやって下さい。

1990.11.22.THU
編集長
井上　守

lime....

1990　松ヶ崎祭

K.I.T　コンピュータ部