TSG

Theoretical Science Group

理論科学グループ

部報 208 号
— うれしはずかし夏合宿号 —

目　次

夏休みの予定 ー涼しいお部屋で...ゲームしよ?ー

夏合宿のお知らせ

菅原

詳細

今年の TSG 夏合宿の計画は現在のところ次のようになっております。変更あるいは追加事項は逐次電子メールといぬにてお伝えします。

日程　8 月 3 日 (日) 〜 8 月 5 日 (火)
場所　千葉県白子 ホテル東天光
　　TEL: 0475-33-3100, 最寄り駅: JR 茂原駅
費用　20,000 円 (宿泊その他) + 6000 円 (交通費)
交通　鉄道
集合　8 月 3 日 (日) 午後 2 時 東京駅 銀の鈴広場 銀の鈴前
解散　8 月 5 日 (火) 到着次第 東京駅にて

費用

宿泊その他にかかる費用は前払いとなります。駒場生より順次、担当の菅原に払ってください。領収書は発行する予定です。

7 月中にどうしても駒場に来られないなどで、費用が払えない場合は当日に支払うことになりますが、その分コンパ委の負担が大きくなるので、できるだけ早く払うようにしてください。

また、**現地で昼食は用意されない**ので**昼食は各自で取るよう**お願いします。

旅程

東京駅に 8 月 3 日 (日) の午後 2 時頃集合し、茂原駅まで JR 外房線。茂原駅からホテルまでバスを利用することを予定しています。注[1]注[2]

[1] 二日めから参加するなど、何らかの事情で単独で茂原まで行く方はあらかじめ僕に到着時刻などを伝えて下さるようお願いします

[2] JR 外房線の時刻表を自ら調べたいという方は、外房線が 7/19 〜 8/17 まで特別ダイヤになるという点に注意して下さい

参加してくれる方々

現在のところ当方で把握しているのは以下の方々です (敬称略)

参加決定

経澤 須磨 多賀 西澤 大岩 鴨島 高野 松村 富樫
金子 木原 植原 坂東 菅原 保原 馬本
斉藤 (文彦) 坂尾 大久保 長谷川

予定があえば参加

丹下

おべんとイベント たのしいな

- 2 日目 (8 月 4 日) の午後にテニスコートを借りるので、テニスをしたい方はラケットとシューズを用意して下さい。ボールは僕が持参します[3]。
- 海岸まで徒歩 5 分と近いので、水着などを用意すると良いでしょう。
- その他のイベントについては現在検討中です[4]。

質問・意見などは以下まで。

TSG コンパ委員 菅原 豊
TEL: 0466-50-6182
E-Mail: g640535@komaba.ecc.u-tokyo.ac.jp
INU00049 (いぬ x BBS)

[3]ニューボールではありません:P
[4]編注:コンパ委員に任せきらないでいろいろ提案してあげると良いでしょう

夏休み 3x5 開館日程

アンブレラ委員

開館日程

月	火	水	木	金	土	日
8/4 −−	5 −−	6 −−	7 植原	8 −−	9 丹下	10 −−
11 −−	12 坂本	13 保原	14 金子	15 −−	16 菅原	17 −−
18 −−	19 坂東	20 保原	21 菅原	22 −−	23 坂東	24 −−
25 −−	26 丹下	27 植原	28 坂本	29 −−	30 金子	31 −−

時間　11:00 〜 16:00 (時間延長は状況に合わせて適宜可能)

−−　責任者不在。都合により予約はできない。

概説

夏休みの間は学生会館がほとんど利用できないこと等から、1 年生も気軽に (2 年生に予約無しに) 利用できる 3x5 環境を提供するために[1]、予め 3x5 に責任者となる 2 年生を送り込んでおくことにしました。

上記の表に責任者の記述がある日の 11:00 に神泉駅改札口前に来れば、TSGer は誰でも 3x5 を利用することが可能です。

コンピュータはあるけど一人では何をしていいのかわからなかったり、数人が集まらないとできない作業や、誰かに見せたいものを作ったりした時に来たらきっといいことがあるでしょう。つまり普段の 305 的存在と思ってください。

具体的に何をするかについては、以前部報に書いた規約に従う限り[2]自由です。

[1] 大袈裟だな:D

[2] つまり「アンブレラに迷惑かけちゃダメよ」ってこと

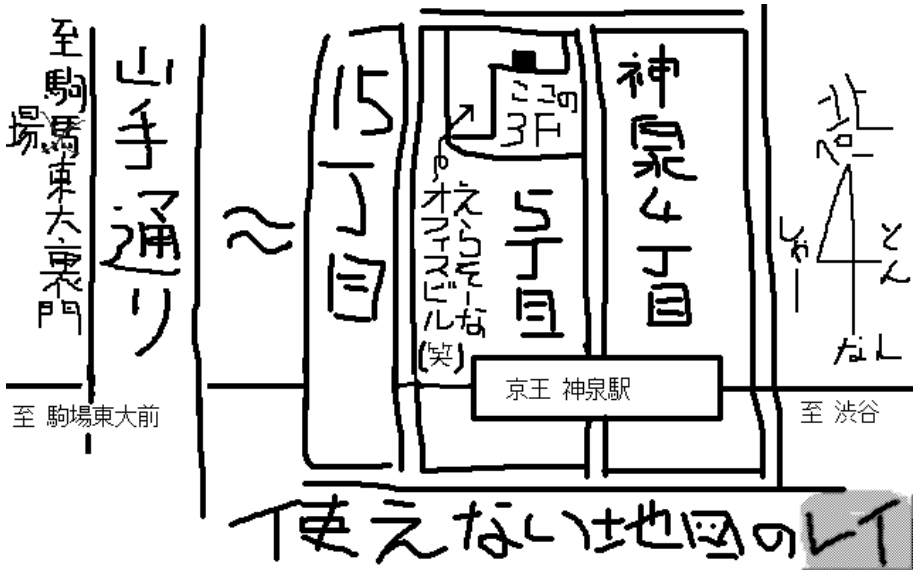
夏休み 3x5 開館日程

海や山や地球に遊びに行く計画をみんなで立てるもよし、夏学期なんにもできなかった人もここで遅れを取り戻してみたり、(無いと思うけど) 駒祭プログラムを作り始めるもよし、TSG の名を有名にするべく壮大なプロジェクトを立ちあげるもよし、**カップラーメン**を極めるのも良いでしょう。

ただし、釘をさすようですが、おざわさんから以下のようなコメントを頂いています。

- **私語が耳についた場合**
  その時点のアンブレラの緊迫度に合わせてうまくやって下さい。地声の大きい人は注意 :-)。
- **アンブレラ社内にある一切のことについて**
  すべて個人のむねの内にとどめるようにしてください。他言が判明した時点で何らかのペナルティをかします。現在かなり秘密なプロジェクトが進行中なので、この点はくれぐれもげんじゅーによろぴく。

また、突然利用できなくなった場合は 11:00 神泉駅改札前で担当責任者がその旨告知します。この場合、本郷の ReACT なり、海へ遊びに行くことになったりするかもしれませんが、御了承ください。



駒祭へむけた準備

うえ

実際の作業はいつから始まるのか?

駒祭へ向けた日々

今年も暑い夏がやってきました[1]
夏が終われば秋
秋といえば当然

駒場祭

です。

ここで、駒場祭までの道のりを記しておきましょう。

7 月 2・3 日	第 1 回企画代表者会議
7 月 30 日	午後 7 時 企画申込書締め切り
8 月 19 日	部屋割り発表
9 月 4 日	部屋割り異議申請締め切り
9 月 9 日	第 2 回企画代表者会議・プログラム原稿締め切り
10 月 18 日	第 3 回企画代表者会議
11 月 1 日	第 4 回企画代表者会議
11 月 15 日	第 5 回企画代表者会議
11 月 22,23,24 日	駒場祭

企画を決める

例年 TSG は 1 号館の 106 教室をまるごと占領してきました。

1 部屋とれると展示がやりやすいだけでなく、いろいろと愉快なことができるのですが、その詳細については駒場祭のお楽しみです。> 1 年生な人々[2]

[1] 暑すぎる、異常。
[2] 今年はウテナか?

しかし 1 部屋とるためには登録する企画がかなり必要となります。
多すぎてもいろいろと面倒なので、今年も例年どうり 4 つでいくことにしました。

1. ネットワーク対戦 RPG
2. 理論科学シミュレーション
3. 人ごみの動きのシミュレーション
4. 1 年の何か

を登録するつもりです。

問題点

さて、ここで問題があります。つまり

1 年の企画が立っていない (爆)

一応金を稼ぐ企画をやる、と決まったのですが...
金銭が絡む企画の場合きちんと登録しておかないとまずいのは明白なので、さっさと決めなくてはなりません。しょうがないので現在しょっちゅう部室に顔を出す 1 年生に決めてもらいましょう。締め切りもテスト終了直後なだけに、今のうちに決めておかないとまずい。この部報が出る頃には決まっているのか?

駒祭に向けて何か作ろう!

とりあえず登録しておく企画は 4 つだけですが、その他の企画ももちろん歓迎しますし、必要です。
現在の 2 年生にはどうせ何か作るでしょうから特に言いませんが、一年生の人には

何か作ってみよう

と強く呼びかけます。自分の興味のある物ならなんでもあり。ゲームでもシェルでもカーネルでも CG でも音楽でも同人誌でも[3] 何でも構いません。
現在プログラムが組めないからといって何もやらないのはもったいないですよ。駒場祭に向けて作っている間に必要な知識などはやりながら身につければいいわけです。もしも企画が完成しなかったとしてもそれはそれで構わないでしょうし[4]。

[3] ??
[4] 編注:ねぇレイちゃん

分科会活動報告

DirectX 分科会 DirectSound の使い方

1997 年 7 月 5 日

わたる

はじめに

前回の記事では DirectDraw の使い方を説明しましたので、今回は DirectSound を解説することにします。

DirectSound とは、Windows マシンに搭載されている PCM 音源を制御するためのインターフェイスです。単純に wave ファイルを再生するだけでなく、音量や周波数の制御はもちろん、音の合成も簡単にできます。また、DirectSound3D を用いれば、音源の 3 次元的な移動を表現でき、音量の変化やドップラー効果などの処理を自動的にしてくれます。これらの特殊効果は、CPU の力技の計算により実現されますが、もし PCM 音源がそれらの機能をハードウェア的にサポートしている場合は[1]、自動的にそちらが使用されます。

DirectSound の初期化

DirectSound オブジェクトの生成

DirectSound を使うには、他の DirectX コンポーネントと同様に、まず DirectSound にアクセスするためのインターフェイスを手に入れる必要があります。それが DirectSound オブジェクトです。

```
#include <windows.h>
#include <dsound.h>
(中略)
  LPDIRECTSOUND lpDS;
  HRESULT hr = DirectSoundCreate(NULL, &lpDS, NULL);
  if(hr) err("DirectSound が使用できません。");
```

[1] Total3D や MonsterSound などが対応している。(未確認)

DirectSound オブジェクトは、DirectSoundCreate() によって生成します。パソコンが複数の PCM 音源を装備しているかもしれないので、第 1 引数にドライバの GUID を与えることにより、音源を指定できます。しかし、そのようなマシンは現時点では非常にまれですから、ここでは NULL を与えてデフォルトの音源を使用するようにしています。

DirectSoundCreate() のプロトタイプ宣言は、dsound.h の中にあります。実体は dsound.lib です。インクルード、リンクして下さい。

err() は、解説をわかりやすくするために私が作った関数です。引数に与えられたメッセージを表示して、プログラムを強制終了させる機能があります。実際のプログラミングでは、hr の値をチェックして、どのようなエラーが発生したのか、ちゃんと調べて下さい。

```
hr = lpDS->SetCooperativeLevel(hWnd, DSSCL_NORMAL);
if(hr) err("DirectSound の協調レベルを設定できません。");
```

次に DirectSound の協調レベルを設定します。協調レベルとは、プログラムが PCM 音源をどの程度、排他的・独占的に使うか定めるものです。通常は DSSCL_NORMAL を指定して下さい。プログラムがアクティブな状態の時にだけ音が鳴ります。

プライマリバッファの生成

DirectSound には、PCM データを蓄えるためのバッファが 2 種類あります。プライマリバッファとセカンダリバッファです。プライマリバッファには、実際にスピーカへ出る波形データを置きます。プライマリバッファは 1 つしかありません。セカンダリバッファには、wave ファイルの波形データを格納します。セカンダリバッファはメモリがある限り、いくつでも作れます。

PCM を再生するときは、セカンダリバッファのデータを必要に応じて変換しながら、プライマリバッファにすでにあるデータと合成するわけです。必要な計算は全て DirectSound のシステムがやってくれます。

プライマリバッファは DirectSound の初期化時に作ってしまって下さい。

```
LPDIRECTSOUNDBUFFER lpPrimaryBuffer;
DSBUFFERDESC dsbd;
ZeroMemory(&dsbd, sizeof(dsbd));
dsbd.dwSize  = sizeof(dsbd);
dsbd.dwFlags = DSBCAPS_PRIMARYBUFFER | DSBCAPS_CTRL3D;
hr = lpDS->CreateSoundBuffer(&dsbd, &lpPrimaryBuffer, NULL);
if(hr) err("DirectSound のプライマリバッファを作れません。");
```

波形データを格納するバッファは、プライマリ・セカンダリの両方とも IDirectSound::CreateSoundBuffer で生成します。第 1 引数に与える DSBUFFERDESC 構造体の dwFlags で、バッファの種類を指定するのです。この構造体は必ずゼロクリアし、dwSize にその大きさを代入する必要があります[2]。DirectSound3D を利用したい場合は、プライマリバッファ作成時に DSBCAPS_CTRL3D も指定する必要があります。

これで DirectSound の初期化は完了しました。

[2] DirectDraw などと同様、dwSize によってバージョン判定をしているためです。

PCM の再生

wave ファイルの読み込み

PCM を再生するには、波形データを格納するセカンダリバッファを生成する必要があります。しかし、セカンダリバッファを生成するには、あらかじめ PCM フォーマットの種類とデータの大きさを調べておかなければなりません。

ここでは Windows の標準的な PCM のファイル形式である.wav の読み込み方法を解説します。なんと、DirectSound には.wav を読み込んでくれる API がありません。自力で.wav を解析する必要があります。

以下のソースは samle.wav というファイルを読み込んで、&wave_file[0x2c] に波形データの開始アドレスを、buffer_size にデータの大きさを、*lpWaveFormatEx にフォーマットの種類を、格納するプログラムです。

```
//セカンダリバッファを作成するのに必要な情報
unsigned char *wave_file;
int file_size;
DWORD buffer_size;
LPWAVEFORMATEX lpWaveFormatEx;

//.wav の読み込み
FILE *file = fopen("sample.wav","rb");
if(file == NULL) err("%s を読み込めません。");
file_size = _filelength(_fileno(file));
wave_file = (unsigned char *)malloc(file_size);
if(!wave_file) err("malloc() 失敗。");
fread(wave_file, 1, file_size, file);
fclose(file);

//本当に WAVE ファイルかチェック
if( *((DWORD *)&wave_file[0x08]) != *((DWORD *)"WAVE") ){
  free(wave_file);
  err("WAVE ファイルではありません。");
}

//PCM の情報を取得
lpWaveFormatEx = (LPWAVEFORMATEX)&wave_file[0x14];
int offset = 0;
while( *((DWORD *)&wave_file[offset]) != *((DWORD *)"data") )
  offset++;
offset += 4;
buffer_size = *((DWORD *)&wave_file[offset]);
```

ひどいことをいろいろとやっているソースですが (笑)、とりあえず動きます。.wav ファイルはフォーマットが何種類かあるので（PCM フォーマットのことではありません）、自分でもっとましなルーチンを作るときは気を付けて下さい。

セカンダリバッファの生成

.wav を解析して得た情報を元に、セカンダリバッファを生成します。

```
DSBUFFERDESC dsbd;
ZeroMemory(&dsbd, sizeof(dsbd));
dsbd.dwSize        = sizeof(dsbd);
dsbd.dwFlags       = DSBCAPS_STATIC | DSBCAPS_CTRLDEFAULT;
dsbd.dwBufferBytes = buffer_size;
dsbd.lpwfxFormat   = lpWaveFormatEx;

LPDIRECTSOUNDBUFFER lpSecondaryBuffer;
hr = lpDS->CreateSoundBuffer(&dsbd, &lpSecondaryBuffer, NULL);
if(hr) err("DirectSound のセカンダリバッファを作れません。");
```

dwFlags の DSBCAPS_STATIC を付けないと、PCM の再生後にバッファが自動的に解放されるため、一度だけしか再生できなくなります。

WAVEFORMATEX 構造体を正しく記述できれば、.wav 以外のファイルの PCM でもセカンダリバッファを作れます。この構造体の詳細は SDK Help で調べて下さい。

波形データをセット

セカンダリバッファを生成しただけでは、まだ PCM の波形データはバッファに格納されていません。プログラムで転送してやる必要があります。

```
LPVOID lpBuf1, lpBuf2;
DWORD  dwBuf1, dwBuf2;
hr = lpSecondaryBuffer->Lock(0, dsbd.dwBufferBytes,
                        &lpBuf1, &dwBuf1, &lpBuf2, &dwBuf2, 0);
if(hr) err("lpSecondaryBuffer->Lock() に失敗。");

CopyMemory(lpBuf1, &wave_file[0x2c], dwBuf1);
if(dwBuf2 != 0)
  CopyMemory(lpBuf2, &wave_file[0x2c] +dwBuf1, dwBuf2);

hr = lpSecondaryBuffer->Unlock(lpBuf1, dwBuf1, lpBuf2, dwBuf2);
if(hr) err("lpSecondaryBuffer->Unlock() に失敗。");
free(wave_file);
```

セカンダリバッファは一種のグローバルメモリ、共有メモリです。アクセスするには他のスレッドの排他制御をする必要があります。それが IDirectSoundBuffer::Lock です。Lock() は波形データの格納先をポインタで指定してきます。PCM が再生中のときは、バッファが 2 箇所に分割されていることがあります。その場合はポインタを二つ返してきます。

バッファのアドレスを取得できたら、CopyMemory() で波形データを転送して下さい。.wav を読み込んだメモリの解放も忘れずに。

再生

セカンダリバッファができてしまえば、後は簡単です。

```
hr = lpSecondaryBuffer->Play(0,0,0);
```

これで再生できます。複数のセカンダリバッファを同時に Play() すれば、自動的に合成されます。再生をループさせることもできます。

```
hr = lpSecondaryBuffer->Play(0,0,DSBPLAY_LOOPING);
```

これだけす。

再生するにときに気を付けなければならないことが、一つだけあります。PCM のバッファは、プライマリ・セカンダリとも、DirectDrawSurface のように lost することがあるのです。プログラムが非アクティブな状態になると、Windows はメモリを他のプログラムに割り当てるために、巨大な PCM データを破棄してしまうのです。バッファが破棄されてしまった場合、Play() はエラーコードに DSERR_BUFFERLOST を返してきます。IDirectSoundBuffer::Restore() でメモリは再確保できますが、波形データは復元されません。.wav ファイルなどから読み込み直す必要があります。

再生を停止

PCM の再生を停止するには、Stop() を呼び出します。

```
hr = lpSecondaryBuffer->Stop();
```

これにより直ちに停止されるとは限りません。プライマリバッファにすでにコピーされてしまったデータは、そのまま再生されてしまいます。

再生・停止状態の判定

PCM が再生中か停止状態なのかも、取得できます。

```
DWORD status;
hr = lpSecondaryBuffer->GetStatus(&status);
if(hr) err("lpSecondaryBuffer->GetStatus() に失敗。");
if(status & DSBSTATUS_PLAYING){
  //再生中。
} else {
  //停止中。
}
```

同じ PCM を多重再生する

同じ PCM を合成して再生するには、Play() を何度実行しても無駄です。再生中のセカンダリバッファで Play() を呼び出すと、PCM の再生は一度停止し、波形データの最初から再生を開始します。DirectSound では、PCM の合成したい数だけセカンダリバッファが必要なのです。しかし、同じ波形データをいくつも持つのは、メモリの大変な浪費になります。そこで、波形データを共有したセカンダリバッファを作成する方法が用意されています。

```
hr = lpDS->DuplicateSoundBuffer(lpSecondaryBuffer,
                &lpSecondaryBuffer2);
if(hr) err("DirectSound のセカンダリバッファをコピーできません。");
```

IDirectSound::DuplicateSoundBuffer で複製されたセカンダリバッファは、元のセカンダリバッファと波形データを格納するメモリを共有しています。

終わりに

以上のように、DirectSound の扱い方は非常に簡単です。処理も全然重くないので、ぜひ自作のソフトで使ってみて下さい。

DirectSound3D も非常に簡単なので、ついでに解説を書こうかと思っていたのですが、Direct3D との連携を説明しないと意味がないので、今回はやめることにします。基本的には、DirectSoundBuffer から QueryInterface して、座標や速度、向きをを指定してやるだけなのです。しかし実際に使うときは、それらの要素は時事刻々と変化するので、更新方法などに Direct3D のテクニックがいろいろと必要になります。紹介するのは Direct3D の後にしておきましょう。

理論科学分科会 第 1 次中間報告

うえ

経緯

「理論科学分科会」なる怪しげな分科会が設立されたのは今年の 4 月です。ずっと前からある由緒ある分科会だと思っていた人もいるようですが、間違えなきように。(笑)

で、基本的に本 (偏微分方程式の差分解法) を輪読した訳ですが、どうも TSG の分科会と言うより 305 にある某サークルのゼミという感が強かったような。

今振り返ると、この本の内容は、ひたすら「数値計算の方法」を書いているものなので、やってそれほど楽しいゼミではありませんでした[1]。

おまけに、やはり 1 年生にはかなりハードな内容だったようです。

そのせいか、参加する 1 年生の数は単調減少。(0 にはなりませんでしたが...)

今後

試験が近いので夏学期の活動はこれで終わりますが、冬学期の予定はまだ立っていません。(そもそも続くかどうかも疑問ですが)

とりあえず数値計算に興味のある人は駒場祭で展示するプログラムを組んで欲しいです。

冬学期に行うゼミについては、2 年生の中からいろいろと意見が出てはいるのですが (力学系の理論とか)、何かやりたい分野がある 1 年生がいるならば、知らせて下さい。基本的にジャンルは問いません[2]。

[1] 必要なのは認めますが

[2] といっても数学か物理かでしょうが

情処分科会 レイストーム分科会

6 月 11 日、高田馬場にて発動

しのぶ

其の壱

山手線の高田馬場駅付近に大きなゲームセンターを発見。ビルの 6 階で、全然混んでいない上、かなり広い。ぷよぷよ 2(通) やリッジレーサーなど、305 でよく動いているゲームがたくさんある。しかもたいていのゲームが 50 円。10 円でできるゲームセンターが存在することを考えるとそんなに安くはないが、やっぱり安い。うれしくなってレイストームをした。大きな画面でできるのと、普通の画面の 2 種類があった。どちらも 50 円。もちろん大画面の方で行なうべきだ。

その他のゲームとしては、セーラームーンクイズや電車で GO などが目についた。(電車で GO の方は 200 円) しかし、50 円でできるゲームの機械には、「緊急に値下げ」などという貼紙が貼られてやや怪しかった。しかし考えてみたらリッジレーサーもレイストームも部室に行けばただでできる。あらまあ。というわけでこの前行ってからもう 10 日ほどここには足を運んでいないが、とりあえず広さと安さの点ではかなり群を抜いているので、暇なら 1 度行ってみるのも良いと思う。

其の弐

池袋駅の東口を出てサンシャイン 60 のほうに歩いていくと東急ハンズの向かいにセガの大きなゲームセンターがあります。6 階だてぐらいで、上の方の階にプリクラばかり集めた所があります。私は 4 月にここでピングー[1]のプリクラをとりました。人気のあるプリクラの後ろには待ち合いイスが置かれて若い女性がえんえんと列を作っていました。

しかし、そろそろ世間のプリクラ熱はさめ始めているのでしょうか? 個人的にはパソコンとデジカメで作ったプリクラをみてみたいのですが...[2]。

[1] そういうペンギンのキャラがある
[2] 実家の母は最近デジカメを買って遊んでいるらしい

其の参 (付け足し)

私の家にはプレステもサターンも存在しないのですが、このたびプレステを借り受ける約束をとりつけることができました。6 月 27 日 (金曜日) 入手予定。極パロやバイオハザードがあるらしいし、おまけにネジコンもついてリッジレーサーもできるのです。うふ。

情処分科会 こんぴゅーたと SS と PS と TRPG

あるいは『雑記』

HiMaJin=坂尾 要祐 (ItaO)

僕が TSG に入ってもう 3 か月たちました。いやー、早いですねぇ。

今回 (情処分科会) はここで何か記事を書けという事なので、これまでの自分が TSG でこれまでやってきたこととか、その他てきとーに自分が興味を持ってることとかについて書こうと思います。

TSG のお家芸のコンピュータのことについて...

実は、あんまりやってないんです。できるようになったことと言えば、メールの取り扱いとか LaTeX の初歩とか C++ の初歩とか Pascal の初歩とか...要は初歩しかできてないんです。これはカップ麺ばかり食べてばかりないでもう少し精進しないといけませんねぇ。

我が家のメインマシンであるところのサターンについて

はやくパソコン買わないとなー (^^;;。

わくわく 7[1]はやっぱり面白いです。システムは某カプコンの某バンパイア・シリーズに近いけど、ノリはこっちの方が好きです。必殺技とかわくわくコンボ[2]とかも簡単に出せるというのもいいし。

何といってもキャラがよいです。特に、最近はダンディ J[3]が気に入ってます。SE の音質が悪いのがちょっと心残りですね。ちょっと動きも遅いし。でも、BGM も全曲アーケード版からアレンジがかかっているし、全キャラのエンディングに声が入ってるし、個人的にはかなりお勧めのソフトです。

[1] サンソフトのわくわくでほえほえなノリの格闘ゲーム
[2] 某バンパイア・シリーズのチェーンコンボみたいなもん
[3] 38 歳の探険家。筋肉系ながら、すごく渋いキャラ。クロコダイル・インディではない。

あと、サターンといえばあとはサイバーボッツ[4]。零豪鬼[5]はなんともいえないです。というか強い。1 度お試し下さい。

部室でバリバリに活躍している PS について

いやー、面白いですよ、エースコンバット 2 は。あれはかっこいいです。(でもまだ EXPART モードの操縦ができない... こんなとこでもまだ初心者か (泣)。レイストームもまだ下手だし) あと、どーでもいいことかもしれないけど、AZITO ほしい! 今までに『地下秘密基地経営シミュレーション』などという熱いジャンルのゲームが在っただろうか? いや、ない (反語)! あ、でもこれでもこれシュミレーションゲームだし、セーブに 15 ブロック使うし、マウスあったほうがいいゲームだから多分部室には入らないんでしょうね。残念。

なんとなく TRPG のことについて

それにしても私利私欲に走りまくった記事になってますね、これ。

いやー『天羅万象』[6] カッコいい! 『トーグ』[7]面白い! そして、祝!『It came from late late late show』[8](略称『レレレ』) 日本語版発売! (欲しい!)

いやー、最近いい TRPG たくさんやれてうれしいなぁ。あとはお金さえあればいいんですけどねぇ。(切実)

なんか、中身がない上に、非常に読みづらい記事になってしまいました (脚注とか括弧とか異様に多いし)。これからはもうちょっとパソコンをつかえるようになってまともなことが書けるようになるといいなぁ。(トホホ)

[4]カプコンの高機動ロボット格闘ゲーム

[5]サイバーボッツに登場する、豪鬼 (ストリートファイターシリーズの名物キャラ) 型ロボット。ちなみにこのゲームに出てくるのは零豪鬼の初号機らしい。

[6]諸行無情で熱血でサイバーなハイパーオリエンタル RPG

[7]6 つの異世界からの侵略者と活劇するスーパーヒーローハイパーメタジャンル RPG

[8]趣味と悪ノリの B 級俗悪深夜テレビ映画製作 RPG

UNIX 分科会活動報告その 1

菅原

はじめに

最近僕の手によって、勝手に UNIX 分科会なるものが発足しました。主な活動内容は X ライブラリプログラミングや、UNIX システムコールプログラミング等です。たまに PI の計算速度競争[1]などもしています。

今回は文化会活動報告ということで、X ライブラリを用いたプログラムを一つ紹介したいと思います。

簡易グラフィック通信プログラム xmouse

今回紹介するプログラムは、xmouse[2]という、X ライブラリを利用して、簡単なグラフィック通信を行なうものです。ソースは ~g640535/c/Project/xmouse.c として保存してあるので、もの好きな方は参照して下さい。

実行するには、通信する相手のディスプレイ番号を調べて、さらに相手に xhost を実行してもらってから

```
xmouse -d ディスプレイ番号
例) xmouse -dxt132
```

とします。駒場での使用に限定してつくったので、通常ディスプレイ番号にくっつく "komaba.ecc.u-tokyo.ac.jp:0.0" は省略できます。というか、省略しないとエラーになります。-d オプションを省略すると自分の画面に二つのウィンドウが現れます。

実行すると自分と相手の画面にウィンドウが現れ、そこの中でマウスをドラッグすると図が描けます。上のツールバーで色や筆の太さを変えられます。相手の画面にも同じ図が描かれるので、一種の通信ができます。終了するにはボタン 3(右ボタン) をクリックします[3]。

[1] ガウスの公式で現在 30 秒で 12000 桁

[2] このネーミングの裏には、最初マウスのテストのために作ったプログラムが拡張され、いつの間にかグラフィック通信プログラムになってしまったという経緯があります

[3] ツールバーまで作っておきながらこの仕様は一体...

プログラムリスト

以下が xmouse のソースリストです。

```
 1 #include <stdio.h>
 2 #include <X11/Xlib.h>
 3 #include <X11/Xutil.h>
 4
 5 #define  Maxcolor 8
 6 /*cc -o xmouse xmouse.c -I/usr/local/X11R6/include -lX11 */
 7
 8 main(argc, argv)
 9 int argc;
10 char **argv;
11 {
12  Display  *d[2];
13  Window   w[2], cw[2];
14  GC       gc[2][2];
15  XEvent   e[2];
16  XColor   scrdf,extdf;
17
18  char     dn[100];
19  int      i,j, r[2], c[2], bton[2];
20  unsigned long pixels[2][Maxcolor];
21  static char *colors[] = {
22   "black", "blue", "red", "magenta", "green", "cyan", "yellow", "white"
23  };
24  strcpy ( dn, "" );
25  if ( argc > 1 )
26  {
27   if ( ( strncmp ( argv[1], "-d" ,2 ) ) == 0 )
28   {
29    strcpy ( dn, &(argv[1][2]) );
30    strcat ( dn, ".komaba.ecc.u-tokyo.ac.jp:0.0" );
31   }
32  }
33
34  d[0] = XOpenDisplay( NULL );  d[1] = XOpenDisplay ( dn );
35
36  for ( i = 0; i < 2; i++ )
37  {
38   w[i] = XCreateSimpleWindow( d[i] , RootWindow( d[i], 0 ),
39           0, 0, 800, 600, 1,
40           BlackPixel ( d[i], 0 ), WhitePixel ( d[i], 0 ) );
41
42   XMapWindow ( d[i], w[i] );
43
44   XMapWindow ( d[i], cw[i] = XCreateSimpleWindow ( d[i], w[i],
45                0, 0, 600, 60, 1,BlackPixel ( d[i], 0 ),
46                WhitePixel ( d[i], 0 ) ) );
47   XFlush ( d[i] );
48   gc[0][i] = XCreateGC ( d[i], w[i], 0, NULL );
49   gc[1][i] = XCreateGC ( d[i], w[i], 0, NULL );
50
51   for ( j = 0; j < Maxcolor ; j++)
52   {
53    XAllocNamedColor ( d[i], DefaultColormap ( d[i], 0 ),
54     colors[j], &scrdf, &extdf );
55    pixels[i][j] = scrdf.pixel;
56   }
57
58   XSelectInput ( d[i], w[i], ButtonPressMask | ButtonReleaseMask |
59    PointerMotionMask | ExposureMask );
60   XFlush ( d[i] );
61   c[i] = 0 ; r[i] = 10; bton[i] = 0;
62
```

```
 63  }
 64  XStoreName ( d[0], w[0], "xmouse parent");
 65  XStoreName ( d[1], w[1], "xmouse child");
 66
 67  XSetForeground ( d[0], gc[0][0], pixels[0][c[0]] );
 68  XSetForeground ( d[0], gc[1][0], pixels[0][c[1]] );
 69  XSetForeground ( d[1], gc[0][1], pixels[1][c[0]] );
 70  XSetForeground ( d[1], gc[1][1], pixels[1][c[1]] );
 71
 72  XSync ( d[0] ); XSync ( d[1] );
 73  while ( 1 )
 74   for ( i = 0; i < 2; i++ )
 75   {
 76    if ( XCheckMaskEvent
 77          ( d[i], ButtonPressMask | ButtonReleaseMask |
 78            PointerMotionMask| ExposureMask , &e[i] ) )
 79    {
 80     switch ( e[i].type ){
 81     case Expose:
 82      PensizeDisplay ( d[i], cw[i], r[i] );
 83      PencolorDisplay ( d[i], cw[i], c[i], pixels[i] );
 84      break;
 85     case ButtonPress:
 86      if ( e[i].xbutton.y < 60 )
 87      {
 88       if ( ( e[i].xbutton.x > 200 ) &&
 89            ( e[i].xbutton.x < 200 + Maxcolor*40) )
 90       {
 91        c[i] = ( e[i].xbutton.x - 200 ) / 40;
 92        XSetForeground ( d[0], gc[i][0],
 93         pixels[0][c[i]] );
 94        XSetForeground ( d[1], gc[i][1],
 95         pixels[1][c[i]] );
 96        PencolorDisplay ( d[i], cw[i], c[i] ,
 97         pixels[i] );
 98       }
 99       if  ( e[i].xbutton.x < 180)
100       {
101        r[i] = ( e[i].xbutton.x ) / 60;
102        r[i] = r[i] * 15 + 10;
103        PensizeDisplay ( d[i], cw[i], r[i] );
104       }
105      }
106      else
107       switch ( e[i].xbutton.button )
108       {
109        case Button3:
110         exit ( 0 );
111        default:
112         bton[i] = 1;
113         DrawCircle ( d, w, gc[i], e[i], r[i] );
114       }
115      break;
116     case MotionNotify:
117      if(bton[i])
118       DrawCircle ( d, w, gc[i], e[i], r[i] );
119      break;
120     case ButtonRelease:
121       bton[i] = 0;
122     }
123    }
124   }
125 }
126
127 PensizeDisplay ( d, w, r )
128 Display *d;
129 Window   w;
130 int r;
131 {
132  int i;
```

```
133  XClearArea ( d, w, 0, 0, 175, 60, 0 );
134  for ( i = 10 ; i < 46 ; i += 15 )
135  {
136   XFillArc ( d, w, DefaultGC ( d, 0 ),
137    (i - 10) * 4 + (25 - i/2), 30 - i/2, i, i, 0, 360*64 );
138   if ( r == i )
139    XDrawRectangle ( d, w, DefaultGC ( d, 0 ),
140     (i - 10) * 4, 5, 50, 50 );
141  }
142  XFlush ( d );
143 }
144
145 PencolorDisplay ( d, w, c, p )
146 Display *d;
147 Window   w;
148 int c;
149 unsigned long p[];
150 {
151  int i;
152  GC gc;
153
154  XClearArea ( d, w, 200, 0, 200 + 40*Maxcolor, 60, 0 );
155  gc = XCreateGC ( d, w, 0, NULL );
156  for ( i=0 ; i < Maxcolor ; i++)
157  {
158   XSetForeground ( d, gc, p[i] );
159   XFillRectangle ( d, w, gc, i*40 + 205, 10, 30, 30 );
160   XDrawRectangle ( d, w, DefaultGC ( d, 0 ), i*40 + 205,
161    10, 30, 30 );
162  if ( c == i )
163   XDrawRectangle ( d, w, DefaultGC ( d, 0 ), i*40 + 200,
164    5, 40, 40 );
165  }
166  XFlush ( d );
167 }
168
169 DrawCircle ( d, w, gc, e, r )
170 Display *d[];
171 Window   w[];
172 GC      gc[];
173 XEvent     e;
174 int        r;
175 {
176  XFillArc ( d[0], w[0], gc[0],
177   e.xbutton.x - r/2, e.xbutton.y - r/2,
178   r, r, 0, 360 * 64 );
179  XFillArc ( d[1], w[1], gc[1],
180   e.xbutton.x - r/2, e.xbutton.y - r/2,
181   r, r, 0, 360 * 64 );
182 }
```

プログラム解説

まずこのプログラムのもっとも重要な点は、実際には通信をどこでもしていないことです。かわりに、相手と自分の二つのディスプレイを一つのプログラムで制御することで目的を達成しています。こうすると、処理が重くなる、親側でしかプロセスが走っていないので画像の保存などの機能を付け加えるのに不便である等の不都合が生じるので、本格的なソフトに仕上げようとするには、プロセス分割をした上で socket により通信するなどの工夫が必要となりま

す。しかし、簡単に済ませるにはこの方法が有効です。

それから、UNIX では他のウィンドウなどによって表示が乱された時はプログラム側で再描画を行なわねばなりませんが、このプログラムではツールバーの再描画だけで、描かれた図の再描画は行ないません。完璧に再描画するにはサーバーのバッキングストア機能を使うかあるいはピックスマップを使うという方法がありますが、今回は割愛します。次回があれば、説明したいと思います。

行番号をあげて解説します。

21-22 プログラムで使う色名を文字型配列に入れています。

24-32 実行時に -d オプションが指定された時は dn にそのディスプレイ番号を、それ以外の時は NULL 文字を入れます。

34 自分と相手の、二つのディスプレイを開きます。d[0] が自分用、d[1] が相手用です。

36 以下の、ウィンドウを開くための一連の操作を自分と相手と 2 回行ないます。

38-46 800x600 の大きさのウィンドウを開き、その中に 600x60 の大きさのツールバーを作ります。

47 今まで実行した X リクエストをサーバーに強制的に送ります。

48-49 一人の人が二つのウィンドウに書き込むので、描画属性を保存する変数は一人につき二つ必要です。

51-56 pixels[][] に、”black” から”white” までに対応する色番号を入れます。色番号は各ディスプレイ毎に用意せねばなりません。

58-59 イベントのうち、ボタンの押下 (ButtonPress)、ボタンの解放 (ButtonRelease)、マウスカーソルの移動 (MotionNotify)、ウィンドウの出現 (Expose) を関知するようにします。

61 筆の太さ r を 10 に、色 c を 0(“black”) にします。

64-65 ウィンドウのバーに名前を表示します。

67-70 描画属性を実際に変数 c に対応させます。

72 今まで実行した X リクエストをウェイト付でサーバーに強制的に送ります。

73 以下がイベント無限ループとなっています。

76 現在どのイベントが起きているかを調べています。XCheckMaskEvent は、イベントが来ていなくてもブロッキングされずに、単に False を返します。今回のように二つのディスプレイを扱わなければならない時は有効ですが、システムの負荷は大きくなってしまいます。

80-84 ウィンドウが出現した時、ツールバーを再描画します。

85-105 ボタンが押された時、カーソルがツールバー内にあれば色や筆の太さの変更をします。

106-114 ボタンが押された時カーソルが描画領域の中にあれば、DrawCircle で円をかき、同時にボタンが押されたことを記憶するため bton[i] = 1 とします。

116-119 カーソルが動いた時、bton[i] の値を参照し、ボタンが押されているなら円を描き

ます。

120-121 ボタンがはなされた時、そのことを bton[i] = 0 として記憶します。

127-143 現在の筆の太さをツールバーに表示する関数です。

145-167 現在の筆の色をツールバーに表示する関数です。

169-182 円を描く関数です。

終りに

UNIX 分科会に興味のある方は気軽に参加してみて下さい。現在の活動人数は 2 名ほどで、活動方式は、気が向いた時に情報処理棟におもむき有用なものから使えないものまでまでさまざまなプログラムを作り、もし tsgwho で仲間を見つけたら write でプログラムを紹介しあうといった感じです。

UNIX 分科会活動報告その 2

丹下

はじめに

UNIX でネットワークプログラミングしようということで、チャットのプログラム (C++) を使って簡単に解説していきます。システムコール等の詳細については時間の都合上省かせていただきましたので予め御了承下さい。

プログラムと解説

ネットワークで良く知られている手法としてサーバー・クライアントモデルというのがありますが、ここでもそれにならって話をすすめていきます。まず、サーバーについて見ていきましょう。サーバーしなければいけないことは大体次の 3 つに分けられます。

1. クライアントからの接続を受け付ける。
2. クライアントからの書き込みを他のユーザーに伝える。
3. クライアントが接続を切ったあとの後処理をする。

1 については今回はストリーム型の接続 (接続しっぱなし) を使用するのでクライアントごとに子プロセスを生成して無限ループで随時接続を受け付けます。そこで問題となるのが 2 をどうやって実現するかということです。具体的にはプロセス間通信の方法を考えなければいけないのですが、ここで C++の機能を利用して以下のようなクラスを導入することにしましょう。

`<manager.h>`

```
 1  class manager
 2  {
 3  private:
 4    int current,pipefd[LIST_MAX][2];
 5    bool isNULL[LIST_MAX];
 6
 7  public:
 8    manager()
 9    {
10      for (int i=0;i < LIST_MAX;i++)
11      {
12        pipe(pipefd[i]);
13        isNULL[i] = true;
14      }
15      current = 0;
16    }
17
18    void del(int ID)
19    {
20      isNULL[ID] = true;
21    }
22
23    int getID(void)
24    {
25      for (int i=0;;i++)
26      {
27        if (i > LIST_MAX)
            throw "Too many users. Connection refused.\n";
28        if (isNULL[i])
29        {
30          current = i;
31          isNULL[i] = false;
32          return i;
33        }
34      }
35    }
36
37    void cast(const char *buf,size_t count)
38    {
39      for (int i=0;i < LIST_MAX;i++)
40        if (!isNULL[i])
41           write(pipefd[i][1],buf,count);
42    }
43
44    void lwrite(int ID,const char *buf,size_t count)
45    {
46      if (!isNULL[ID])
47         write(pipefd[ID][1],buf,count);
48    }
49
50    int lread(int ID,char *buf,size_t count)
51    {
52      if (isNULL[ID]) return 0;
53      return read(pipefd[ID][0],buf,count);
54    }
55  };
```

この manager というクラスは、接続してきたクライアントごとに ID を発行し (23 行目の getID)、接続しているすべてのクライアントにデータを書き込んだり (37 行目の cast)、それぞれのクライアントごとの読み書き (44 行目の lwrite と 50 行目の lread) を行ないます。クライアントが接続を切ったら ID を削除します (18 行目の del)。これによって前述の 3 つの機能が実装されました。以下がサーバープログラムの全貌です。

<server.cc>

```
 1  #include <iostream.h>
 2  #include <string.h>
```

```
 3  #include <sys/types.h>
 4  #include <sys/socket.h>
 5  #include <netinet/in.h>
 6  #include <arpa/inet.h>
 7  #include <unistd.h>
 8  #include <sys/ipc.h>
 9  #include <sys/shm.h>
10
11  #include "chat.h"
12  #include "manager.h"
13
14  #ifdef __SUN__    // SunOS ではなぜか宣言しなければいけない関数群
15  extern "C" {
16    void bzero(void *,int);
17    int socket(int,int,int);
18    int connect(int,struct sockaddr *,int);
19    int bind(int,struct sockaddr *,int);
20    int listen(int,int);
21    char* shmat(int,char *,int);
22    int shmget(key_t,int,int);
23    int accept(int,struct sockaddr *,int *);
24  }
25  #endif
26
27  initfd(int &fd)  // ファイルディスクリプタを初期化
28  {
29    struct sockaddr_in sadr;
30    fd = socket(AF_INET,SOCK_STREAM,TCP);
31    bzero((char *) &sadr,sizeof(sadr));
32    sadr.sin_family      = AF_INET;
33    sadr.sin_addr.s_addr = htonl(INADDR_ANY);
34    sadr.sin_port        = htons(TCPPORT);
35    bind(fd,(struct sockaddr *) &sadr,sizeof(sadr));
36    listen(fd,MAXC);
37  }
38
39  template<class type>
40  void share (type* &pm)  // 共有メモリ用関数
41  {
42    pm= (type *)shmat(
43        shmget(IPC_PRIVATE,sizeof(*pm),0666|IPC_CREAT),
44        (char *)0,SHM_RND);
45  }
46
47  main (int argc,char *argv[])
48  {
49    int fd;
50    initfd(fd);
51
52    manager *pm;
53    share(pm);     // manager のインスタンスを共有メモリに割り付け
54    pm->manager();
55
56    int ns,cl,nfd;
57    UINT uClient;
58    char name[MAXL],buf[MAXL],log[MAXL];
59    struct sockaddr_in sadr;
60    cl = sizeof(sadr);
61
62    for(;;)
63    {
64      nfd = accept(fd,(struct sockaddr *) &sadr,&cl); // 接続待ち
65
66      try {
67         uClient = pm->getID();  // ID 発行
68      } catch(char *err) {
69         cout << "Coonection refused." << endl;
70         write(nfd,err,MAXL);
```

```
 71          close(nfd);
 72          continue;
 73      }
 74
 75      ns = read(nfd,name,MAXL);
 76      name[ns] = '\0';
 77      cout << name << " logged in." << endl;
 78
 79      strcpy(log,name);
 80      strcat(log," logged in.\n");
 81
 82      pm->cast(log,MAXL);  // ログインを知らせる
 83
 84      if (fork() == 0)  // 読み込み専用プロセス
 85      {
 86        close(fd);
 87
 88        for(;;)
 89        {
 90          ns = read(nfd,buf,MAXL); // ソケットから読み込む
 91          if (ns == 0) break;
 92          if (ns < 0) continue;
 93          buf[ns] = '\0';
 94
 95          if (!strcmp(buf,QUIT_SYMBOL)) break;
 96
 97          strcpy(log,name);
 98          strcat(log,"\t> ");
 99          strcat(log,buf);
100
101          cout << uClient << ": " << log << flush;
102          pm->cast(log,MAXL);  // 他のクライアントへ送信
103          sleep(1);
104        }
105
106        pm->lwrite(uClient,EXIT_SYMBOL,MAXL);
107        pm->del(uClient);   // ID 削除
108        close(nfd);
109
110        strcat(name," logged out.\n");
111
112        pm->cast(name,MAXL);
113
114        cout << uClient << ": write closed." << endl;
115        exit(0);
116      }
117
118      if (fork() == 0)  // 書き込み専用プロセス
119      {
120        close(fd);
121
122        for(;;)
123        {
124          ns = pm->lread(uClient,buf,MAXL);  // パイプから読み込む
125
126          if (ns == 0) break;
127          if (ns < 0) continue;
128          buf[ns]='\0';
129
130          if (!strcmp(buf,EXIT_SYMBOL)) break;
131          write(nfd,buf,MAXL);  // クライアントへ送信
132          sleep(1);
133        }
134        close(nfd);
135
136        cout << uClient << ": read closed." << endl;
137        exit(0);
138      }
139
140      close(nfd);
```

```
141    }
142  }
```

結局、プロセス間通信の方法としてはパイプと共有メモリを組み合わせて使っています。後者は、<39-45 行目> というテンプレート関数を定義しておけば、任意のデータ型を共有できて便利です。ただし、データはポインターで操作しなければいけませんし、クラスを指定した場合自動的にコンストラクターが呼び出されないので、54 行目のように自分で呼び出す必要があります。次にクライアントを見ていきましょう。

<client.cc>

```
 1  #include <iostream.h>
 2  #include <stdio.h>
 3  #include <string.h>
 4  #include <sys/types.h>
 5  #include <sys/socket.h>
 6  #include <netinet/in.h>
 7  #include <arpa/inet.h>
 8  #include <unistd.h>
 9  #include <netdb.h>
10  #include <sys/wait.h>
11  #include <signal.h>
12  #include "chat.h"
13
14  #ifdef __SUN__    // SunOS ではなぜか宣言しなければいけない関数群 2
15  extern "C" {
16    int atoi(char *);
17    void bzero(void *,int);
18    int socket(int,int,int);
19    int connect(int,struct sockaddr *,int);
20  }
21  #endif
22
23  char* getIP(char dn[])  // ドメインから IP アドレスを割出す
24  {
25    char **pp;
26    struct hostent *phost;
27    struct in_addr *paddr;
28    phost = gethostbyname(dn);
29
30    switch(phost->h_addrtype)
31    {
32    case AF_INET:
33      pp = phost->h_addr_list;
34      if ((paddr = (struct in_addr *) *pp) != NULL)
35         return inet_ntoa(*paddr);
36      break;
37
38    default:
39      throw "Unknown address type";
40      break;
41    }
42  }
43
44  initfd(int &fd,char IP[],int CHATPORT = TCPPORT)
45  {                          // ファイルディスクリプタを初期化
46    struct sockaddr_in sadr;
47    fd = socket(AF_INET,SOCK_STREAM,TCP);
48    bzero((char *) &sadr,sizeof(sadr));
49    sadr.sin_family = AF_INET;
50    sadr.sin_addr.s_addr = inet_addr(IP);
51    sadr.sin_port = htons(CHATPORT);
52    connect(fd, (struct sockaddr *) &sadr,sizeof(sadr));
53    cout << "READY " << CHATPORT << endl;
54  }
55
56  main (int argc,char *argv[])
```

```
 57  {
 58    int fd,n,cpid;
 59    char buf[MAXL];
 60
 61    try
 62    {
 63      if (argc == 3) initfd(fd,getIP(argv[1]),atoi(argv[2]));
 64      else initfd(fd,getIP(argv[1]));
 65    } catch(char *err) {
 66      cerr << err << endl;
 67      exit(-1);
 68    }
 69
 70    if ((argc == 2) || (atoi(argv[2]) == TCPPORT))
 71    {
 72      cout << "input yourname:";cin >> buf;  // 名前を入力
 73      write(fd,buf,strlen(buf));
 74    }
 75
 76    if ((cpid = fork()) == 0)  // キー入力を受け付けるプロセス
 77    {
 78      for (;;)
 79      {
 80        fgets(buf,MAXL,stdin);
 81        if (buf[0] == '\n') continue;
 82        write(fd,buf,strlen(buf));
 83        if (!strcmp(buf,QUIT_SYMBOL)) break;
 84      }
 85      close(fd);
 86      exit(0);
 87    }
 88
 89    for (;;)   // サーバーから送られてくるデータを表示
 90    {
 91      n = read(fd,buf,MAXL);
 92      if (n == 0) break;
 93      if (n < 0) continue;
 94      buf[n]='\0';
 95
 96      cout << buf << flush;
 97      sleep(1);
 98    }
 99
100    cerr << "Connection refused." << endl;
101    kill(cpid,SIGTERM);
102    wait(0);          // 子プロセスを消して待つ
103    close(fd);
104  }
```

クライアントプログラムは、指定されたサーバーに接続し、標準入力から得られたデータを送信しつつサーバーから受信したデータを表示します。残ったヘッダーファイルを示します。

<chat.h>

```
 1  #define __SUN__     // SunOS でコンパイルする際に指定
 2  #define TCP 6
 3  #define TCPPORT 2015  // ポート番号は 2000 より大きい数を指定
 4  #define LIST_MAX 12   // 同時に参加できる最大人数
 5  #define MAXL 512
 6  #define MAXC 5
 7  #define QUIT_SYMBOL "exit\n"
 8  #define EXIT_SYMBOL "\a::EXIT::"
 9
10  typedef unsigned int UINT;
```

最後に Makefile を作ります。

<Makefile>

```
 1  # Makefile
 2
 3  CC = g++ -fhandle-exceptions  // 例外ハンドラのオプションを付ける
 4
 5  all::
 6          make server
 7          make client
 8
 9  server: server.cc chat.h manager.h
10          $(CC) -o server server.cc
11
12  client: client.cc chat.h
13          $(CC) -o client client.cc
```

プログラムソースは以上です。コンパイルするときは、manager.h , server.cc ,client.cc, chat.h, Makefile を同じディレクトリに置いて make を実行して下さい。

プログラムの使用法

適当なワークステーションを選んで server プログラムを走らせます。別のシェルウインドー上で client を実行します。その際、server を走らせたワークステーションのドメインを指定します。

例) ecc-as50 で走らせた場合

```
> client ecc-as50.komaba.ecc.u-tokyo.ac.jp
```

または

```
> client ecc-as50
```

あとはプログラムの指示に従って名前を入力すれば OK です。終了したいときは exit と入力します。kterm 上で kinput2 を使用するなどして漢字入力することも出来ます。

一般記事

部報の書き方

部報マクロの使い方

編集ちょ

初めに部報ありき

まず真っ先に、

何故部報を作らなくちゃいけないのか

この重大にして難解な問題にぶち当たりますが、私やひいては TSG 自体の存在自体を揺るがしてしまいかねないこの問題は先送りすることにしましょう[1]。

さて、部報はどのようにして作られているのでしょうか?

去年の大掃除の時に過去の遺産が多数発掘され、中には

手書き

のものや、なぜか

青い

ものまで出てきましたが、私の記憶が確かならばここ 2 年間は pLaTeX という素晴らしい組み版システムを使用して作成されています。

pLaTeX で部報を作るには...

さて、この pLaTeX というシステム。これで部報を作ろうとした日にゃいろんな問題が浮上してきます。

1. 配布状態のままのクラスでは使い物にならない
2. 執筆者が皆 LaTeX に習熟しているわけがない

[1] 部報のネタに困るようなことになったら、それは TSG 終焉の時なのでしょうね

実際にはもっとたくさんの細かい問題がありますが、編集長としてまずぶち当たる問題になるのがこの 2 点です。

クラスについて

pLaTeX の配布状態のパッケージには jarticle, jreport, jbook の 3 種類のクラスが付いてきます[2]。これらは基本的にオリジナルの英語版を日本語対応にしたものです。しかし、これらのクラスをそのまま使って部報を作ろうとすると

英語臭い偉くたいそうな論文調部報

が出来上がってしまいます。また、部報レベル (数十ページ) の規模に対応するクラスが無いことも問題です。

そこで、しかたなく独自のクラスを作る[3] ことになるです。この際、余計なことにばかりこだわって

誰にも理解してくれない存在[4]

になってしまうこともあるので、この点には今後の編集ちょは気をつけるべきでしょう:D

恐れを知らない執筆者

なにあともあれ pLaTeX で部報を作成できる環境は整いました。しかし、まだ油断できません。原稿執筆者 (TSGer) はコンピュータヲであることであることは多いですが、TeX ヲとは限らないのです。あるときは .tex でありながら

全角文字 や \\ のオンパレード

だったり、あるときは

アスキーグラフ[5]華やかな plain TEXT

を送ってきたり、手書きだったりします。

しかし、こんなことで文句を言ってはいけません。締切を守って原稿を送ってきてくれるだけでそれは素晴らしいことなのです。送られた原稿を取り合えず 'xtr -ch < original > filtered' と全角を半角にして、その後 LaTeX のタグ付けをちまちましている内に夜が明けます。

ところが今年は少し事情が違いました。

TSG としては数年振りの快挙

分科会が夏学期中続いた

[2] 縦書きは除く
[3] 受け継ぐだけのバカもいる:D
[4] 私さえたまに TeX ヲ呼ばわりされる
[5] 文字で作った絵のこと

のです。更に、駒場 ecc に pLaTeX をインストールさせたこともあってこの機を逃すわけにはいかないとばかりに部報マクロを汎用化して、私の担当していた「情報教育棟活用分科会」でこのマクロを使用して部報を書く方法を公開したのでした。実際、この部報にはその時の課題である「1 年生による初めての原稿」が載っているはずです。

このマクロは少々野心的で、みんながこのマクロを使って理想的かつ奇麗な原稿を書いてくれるならば編集ちょは何も考えずに make じゃなくて、

platex 一発!!

で部報が出来てしまうのです。

では、次章以降でこのマクロの使い方を述べることにしましょう。

部報開発キット - Buho Development Kit - その使い方

インストール

部報開発キット (以下 BDK) は現在のところ以下のファイルからなります。

```
TSGBUHO  CLS         5,767 97-07-25    0:00
TSGBH10  CLO         2,524 97-07-25    0:00
TSGBUHO  STY           522 97-07-25    0:00
TYMMAC   STY         2,830 97-10-30    0:00
J1SHAPE  FD          2,449 97-03-20    0:00
OMSCMTT  FD            789 97-03-08    0:00
```

これらのファイルを pLaTeX の検索パスの通ったところに置きます。DOS 版 pTeX-2.1.4 ならば TEXDIR/texmf/web2c/texmf.cnf を編集して TEXDIR/texmf/tex/bdk/ を作成し、そこに入れると良いでしょう。面倒なら TEXDIR/texmf/tex/platex2e/ へ放り込んでも構いません。なお、TEXDIR/texmf/ls-R を作成して検索を高速化してあるなら、当然これも更新するのを忘れないでください[6]。

インストール作業としてはこれだけですが、実は前提としているパッケージ (ascgrp) があるので、これがない場合は `http://www.ascii.co.jp/pb/ptex/` などから別途入手してください。

マクロの使い方

実際の使用方法は実に

簡単

です。

[6] djgpp 版 GNU の ls や UNIX の ls なら ls -1R > ls-R, DOS 版 pTeX 2.1.4 付属 ls-r なら ls-r > ls-R

```
\documentclass[b5paper]{tsgbuho}
\usepackage{tsgbuho}
\usepackage{tymmmac}

\title{タイトル}
\subtitle{サブタイトル}
\author{著者名}

\begin{document}
\makrtitle
本文 (pLaTeX の文法そのまま)
\end{document}
```

とするだけで OK です。ただ、現在のところ author の引数には \thanks 等は使えませんので注意してください。

このマクロは編集ちょが実際に編集に使用するマクロそのままである為、見た感じも tsgbuho.sty で当てたパッチ部分を除き、全く同一になるはずです。これで作成する側も実際の出力イメージと見比べながら作成できるようになったのでだいぶ楽ですよね?!

これで説明を終わります...って、それじゃああんまりなんで「部報」と言う立場の上で LaTeX を使う時の注意を上げてみましょう。

ソースリスト 地球資源を守るため、ソースへのポインタ (在処) を示し、できるだけ必要な部分だけを取り出しましょう。また、どうしても長く引用する場合は、

```
{\small
\begin{quote}
\begin{verbatim}

ソースリスト

\end{verbatim}

\end{quote}}
```

のようにすると良いでしょう。

アスキーグラフ 自分で書いてみるとわかりますが、顔文字を初めとするアスキーグラフはとても LaTeX で実現しようとすると面倒なものです。また、そうでなくても多用はあまり好ましいものでは無いので控えましょう。

脚注 脚注の多用は美しくありません。絶対にやめましょう。

画像 私の環境では、PaintShopPro もしくは MG がコンバートできる画像なら OK ですが、所詮白黒しかでないので自分でうまく減色してから送る方が良いでしょう。

全角 TeX 環境では previewer 上ではいざ知らず、全角英数字よりも ANK の方が美しく印刷されます。全角英数字は避けましょう。

マクロのしくみ

あぁ、私はなぜこんな章を作ったのでしょう? ユーザはしくみなんか知らなくても使えりゃいいじゃないですか。でもそれなのになぜこの章はあるのでしょう? 枚数が足りないから? いいえ、違います。それはこの問いに対する本当の答え

ヲだから

から逃れるための言い訳に過ぎません:D

前置きはさておき、

説明しましょう

といっても全てを書くわけにはいかないので、全体の概要と一部分の詳説を書くことにします。

各ファイルの役割

tsgbuho.cls, tsgbh10.clo

`\documentclass{tsgbuho}` の部分で読み込まれるファイルです。

jbook クラスをオーバーライドして部報に適したクラス設定に変更する、このマクロの要です。実際これらファイルさえあれば部報の印刷は可能です。先代編集長おおいわさんの kaihoh.sty を参考にしたのでよく似てます:D [7]

tsgbuho.sty

`\usepackage{tsgbuho}` の部分で読み込まれるファイルです。

本来 tsgbuho.cls は本の形として定義されたものなので、記事ひとつだけで使用するには不足する定義類などといくつかのパッチ等が記述されています。

以前の BDK を用いた記事のファイル中で `\begin{document}` の代わりに`\article{}{}{}` という変なものを使っていて不思議に思われていたかもしれませんが、このファイル中で `\article{}{}{}` も実際には `\begin{document}` であるとのことが記述されています。

tymmac.sty, j1shape.fd, omscmtt.fd

`\usepackage{tymmac}` の部分で読み込まれるファイルです。

私 (すーゆー/TYM) のオリジナルのよく利用する定義類を書いてあるものです。本質的なものではありません。が、組み込まないと pLaTeX が「フォントが無い」とかうるさいし、`\ruby` なども使えるようになるので組み込んでおきましょう。

[7] いちおー全てのコードは書き直したつもりだけどね

platex 一発の理由

実際の部報では、記事ごとのファイルを \input を用いてインクルードすることで一つの部報に仕上げています。つまり、

```
\documentclass[b5paper]{tsgbuho}
\usepackage{tymmac}
%%
%% 中略
%%
\begin{document}

\input{art1.tex}
\input{art2.tex}
\input{art3.tex}

\end{document}
```

ということをしています。art1.tex は BDK を用いて作成されて送られてきた理想的な記事のファイル

そのまんま

です。

ここまで来て「どうしてそんなことが出来るんだ?」と思わなくてはなりません。なぜなら、記事ファイル中には余計な文字列 \documentclass 等が含まれており、そのままインクルード (\input) してはエラーとなるからです。

すなわちここですべき事は、\doucmentclass や \usepackage というものを人畜無害な \relax という命令にオーバーライドして無効化することです。例えば \foo という命令を \relax と定義する (もしくは定義しなおす) には \def\foo\relax とすれば良いのです。

しかし、これでうまくいくでしょうか? 例えば \def\documentclass\relax としたとしましょう。ここで \documentclass{tsgbuho} がどう解釈されるでしょう?

解釈前 : \documentclass{tsgbuho}
解釈後 : {tsgbuho}

つまり、文章中に tsgbuho と表示されるようになってしまうのです。ならば、引数つきの書式 \def\foo#1\relax を使ってみてはどうでしょう? 確かに \documentclass{tsgbuho} といった記述ならいいのですが、\documentclass には \documentclass[b5paper]{tsgbuho} といった書式もあって対処できません。これは \usepackage でも同様です。

お手上げか? しかし、これで引き下がったら

TEX ヲの名が廃る

のでそんなことはできない... さてどうしたものか...

そして \UltraRelax へ...

はてさて、言語仕様の貧弱な言語ほど遊んでいて楽しいものはありません。いろいろ眺めているとありました、ありました、解決法が。鍵は `\@ifnextchar[{A}{B}` です。これは次の文字が '[' ならば A、違ったら B、という処理をするものです。

そろそろ疲れてきたので、種明かしをすることにしましょう。次の定義を眺めてください。

```
\def\RelaxA[#1]#2{\RelaxD}
\def\RelaxB#1{\RelaxD}
\def\RelaxC[#1]{\relax}
\def\RelaxD{\@ifnextchar[{\RelaxC}{\relax}}
\def\UltraRelax{\@ifnextchar[{\RelaxA}{\RelaxB}}
```

これでわかりましたね、この `\UltraRelax` を使用して

```
\def\documentclass{\UltraRelax}
\def\documentstyle{\UltraRelax}
\def\usepackage{\UltraRelax}
\def\author{\UltraRelax}
\def\title{\UltraRelax}
\def\date{\UltraRelax}
\def\maketitle{\relax}
```

とでもしてやれば、見事問題が解決しました。`\(^o^)/` 解らないですか? ゆっくりたどればじきにわかりますよ。大したことはやっていませんから。

終わりに

いやぁ疲れた。久しぶりなもんだから、文章を書くのがこんなに疲れるものだったてのをわすれてました。

しかし、たった 12KB ぐらいのドキュメントでくたばるようになったってのは軟弱になった証拠ですねぇ。中学の頃、友人数人とでやってたサークル (とゆーかなんとゆーか) では 30KB ぐらいの (くだらない) 記述を毎週書いてた気がするんですが... もう遠い昔ですかね。

当初は某 256 倍シリーズみたいなものをめざしていたんですけど、

書いているうちに眠くなってネタが思い付かなくなった

なんてょぁ過ぎて口が裂けても言えませんわ、おほほほほ。

まぁ、

TSGer 皆チョベリ TEX たれ

とはいいませんけど、

\asc256 マクロ

```
\def\asc256#1{\begin{center}{\normalfont\Large #1}\end{center}}
```

が何を意味してるかぐらいはわかってほしいなぁ、みたいなぁ。

なお、動作チェックは LaTeX 2ε<96/12/01> と pLaTeX 2ε<97/02/01>+2 という組み合わせでしか行っていません。執筆段階では LaTeX 2ε<97/06/28> と pLaTeX 2ε<97/07/02>+1 というものも有るようですが、まだ固まったようでは無いのでまだ導入を見送っています。が、恐らく動作するでしょう。

ホントに最後になりましたが、参考になるかわかりませんが BDK とそのサンプル (原稿のソース) は**いぬ x BBS filer original** に置いていますので、興味のある方はどうぞ。

<e-mail> tym@tky.threewebnet.or.jp
g640770@komaba.ecc.u-tokyo.ac.jp
Author '97 編集ちょ / すーゆー / TYM

STL の欠点と VisualC++のバグ

Nishi

本来ならば STL 入門の原稿が出来上がるはずでしたが、STL を使ったときに現れる、VisualC++5.0 のヘッダファイルのバグに原稿を書く時間を奪われてしまったので、それについて書いてみようかと思います。

STL の欠点

STL の紹介をする前に STL の欠点を書くのもなんか変ですが、とりあえず書いてみることにしましょう。実際に私が使ってみたところ、大きな欠点は次の 4 が挙げられると思います。

1. インスタンスがやたらとできて、実行ファイルの大きさが大きくなる。
2. インスタンス生成時のエラー、警告がテンプレート宣言部分で生じる。
3. デバッガが使いにくくなる。
4. デバッグシンボルがやたらと長くなり、警告がでる。(VisualC++)

1) はテンプレートを使う以上、ある程度やむをえないところではありますが、コンテナに限って言えば、普通の使い方の範囲では、STL のコンテナは、MFC のコンテナを使うよりも、はるかにましでしょう。:)

2) ですが、これは STL のような複雑なテンプレートを使っている場合深刻です。例えば STL の場合 `template<class T> class vector<T>`というコンテナクラスがあるのですが、これを使う際には、`T` に対する `operator==`と `operator<`が定義されている必要があります。もし、こ

れを定義し忘れていると、STL のインクルードファイル側でエラーが大量に出ます。クラス定義が複雑になるほど、エラーの原因の特定が難しくなります。はっきりいって STL はこれではまります (^^;

3) も大きな問題です。STL では演算子のオーバロードを多用しているため、コンテナに格納されているオブジェクトをデバッガで参照するのが結構面倒です。例えば STL の vector クラスは配列のようなアクセスが可能で、`vector<int> v(20);` と定義されている場合、ソースコード上では `v[10] = 0;` というアクセスが可能なのですが、`operator[]` はオーバライドされていて、その結果はコードに依存したものですから、デバッガの watch コマンドで `v[10]` と入力してもエラーがでるか、正しくないデータが表示されるかのいずれかです。これを見るために、STL のコンテナクラス内部を見なくてはなりませんが、もちろん STL 内部の互換性は保証されていません。確かに現時点では大半の処理系で Hewlett-Packard の実装が使われてはいますが、データを参照するのが大変であることはに変わりありません。

4) ははっきり言って VisualC++の仕様上の問題です。VC++ではデバッグシンボル名は 255 以下である必要があるのですが、STL のコンテナを使うと、この制限を越え、警告が出ることがしばしばあります。

ちなみに VisualC++付属の STL サンプルでは警告が出ないように`#pragma` が書いてあったりします (^^; きっと M$の人は M$C/C++以外の処理系を知らないのでしょう (ぉ

VisualC++5.0 のバグ

VisualC++5.0 では C++標準ライブラリのサポートがありますが、STL の `vector<complex<double> > z;` のような、`complex<double>` (あるいは、`complex<float>`, `complex<long double>`) をオブジェクトとするコンテナを宣言すると、STL のヘッダファイル内で、C2300 エラー (デストラクタがない) が発生します。

注) ここでは特に触れていませんが、`complex<T>`を STL コンテナのオブジェクトとする場合、`complex<T>`に対する `operator<`が定義されている必要があります。

Microsoft の複素数ライブラリの実装においては、`template <class T> class complex<T>` は、`T` が float, float, long double のときの専用版が用意されます。というのも、それ以外の型のときは、実際の計算が全て double 精度で行われるからです。これは実装の問題であり、このこと自体に問題はありません。しかし、`complex` ファイルでの宣言が、

```
template <class _Ty> complex{
    complex<Ty> sin(complex<Ty>&);
    complex<Ty> cos(complex<Ty>&);
    ...
};
class complex<double> {
```

```
    typedef double _Ty;
    complex<Ty> sin(complex<Ty>&);
    complex<Ty> cos(complex<Ty>&);
    ...
};
```

のようになっているため、STL ヘッダファイル内の

```
template<class _Ty> inline
void _Destory(_Ty *_P) { *_P->~_Ty;};
```

という部分に引っかってしまうのです。これは、complex<double>内で_Ty が typedef されているため、_Ty は complex<double>にとっては実宣言であり、それが、関数テンプレート _Destroy の仮引数_Ty より強いものとして、処理されてしまうからのようです。この動作が言語仕様的に正しいのかどうかは私にはわかりませんが、解決方法自体は簡単で<complex>を

```
class complex<double> {
    complex<double> sin(complex<double>&);
    complex<double> cos(complex<double>&);
    ...
};
```

のように書き換え、typedef を行わないようにすればいいだけのことです。

とはいえ、これはコンパイラ自体のバグなのでしょうか? それともヘッダファイルのバグなのでしょうか?

ともかく私はこれに 3 時間以上もはまり、STL 入門なんて書いている暇がなくなってしまいました (^^;

編集後記

最近の出来事

1. CD-R 共同購入ほぼ決定 (限定じゃんけんにより、全責任は Bandai へ)
2. もののけえば企画 成功なるか?
3. 夏休み 3x5 開館日程制定へ 夏休みは 3x5 で涼もう
4. 学友会予算、TSG またもや文代サークル最高額

今回の部報で落ちた原稿 (T^T)

1. 和井田 AT 機自作にまつわるえとせとら
2. すーゆー MSCDEX ReadLong で CDDA 吸い出し
3. すーゆー THE END OF FRIENDS, へや/いどころを 君に (C) たなかぁ
4. すーゆー わたるさんの lib98(djgpp2.00) を djgpp2.01 へ移植

この部報は今までで最も難産でした。部報は、ふと思い付きで作るもんじゃないな。終わり。

次回部報は 9 月試験終了後にだします。というわけで締切は 8 月末ごろかな? 夏合宿にいくコンパ委員 (指名:れい) は詳細なレポート (b5 5 枚以上) を提出のこと。正式な締切は決まりしだい いぬ x に書く予定。

表紙は... 初めは You know ? みたいな感じの絵を張ってたけど、映画オフもやったことだしこれにしましょう。

理論科学グループ　部報 208 号

1997 年 7 月 20 日 発行 / 1997 年 11 月 3 日 改訂第 2 版

発行者　植原 洋介

編集者　坂本 崇裕

発行所　理論科学グループ

〒 153　東京都目黒区駒場 3–8–1

東京大学教養学部内学生会館 305

Telephone: 03–5454–4343

理論科学グループ部報 第 208 号
— うれしはずかし夏合宿号 —
1997 年 7 月 20 日

THEORETICAL SCIENCE GROUP