CLDAQ

For experiment coordinators

CREST, JST

Go IWAI

トークに先立って

- ユーザー様むけの話になります
  - プログラマー様むけの内容は別途、ご要望に応じていたします
  - 肝心なことは、あまり話しません
- 開発はほぼ終了しています
  - 最終バージョンは1.14.3（２００４年６月末）
  - 趣味で更新することはありえます

これはなんなのですか？

- CLDAQ - a Class Library for Data AcQuisition
  - シー・エル・ダキューとでも呼んでください
- データ収集システムの構築をサポートするクラスライブラリ
  - アプリケーションではないことに注意
- DAQのための最小セット(Micro KONOE)という位置づけ
  - ２００１年の夏ごろ、飲み会の最中、依頼を受ける
- KONOEでの経験に基づき、いわいがひとりでイチから作成
  - 美しく、OO（オー・オー）の規範となるようなコード

開発の目的

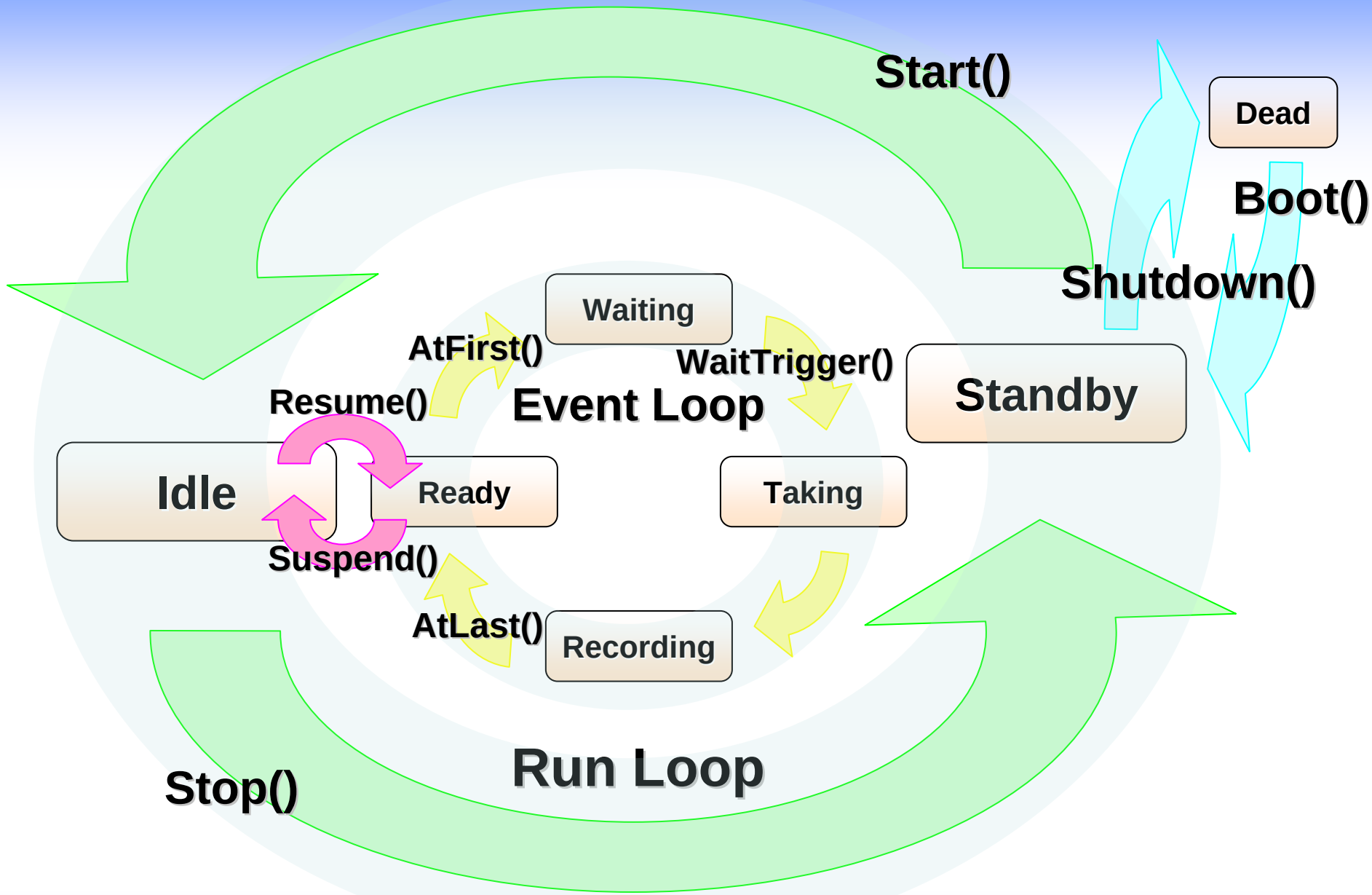
- データ収集システムを手っ取り早く構築
- メインのターゲットはＰＣ１台でやってる貧乏くさい環境の人
  - お金の余ってる大規模実験グループは外注する
  - たいていの実験（測定）はＰＣ１台でスローからファストまでコントロール可能
- オンライン分散の機能も一応提供
  - ちゃんと動くので、おまけという意味ではない

開発の方針

- C++のみで記述
  - ピュアでモノリシックなOOクラスライブラリ
- 無保証かつ、フリーなソフトウェア（GPL）
  - かつオープンソース
  - 何かのサポートを受けた研究ではなく、いわいの趣味の活動
  - http://cldaq.sourceforge.jp/　にて公開中
- 物理解析やデータの可視化は担当しない
- 余計なものは作らない、使わない
  - 世の中に良いものはいっぱいあるのに、素人が作るのはナンセンス
  - 変なスタイルにユーザーは混乱
- 良さそうなものには飛びついた
- 多くのライブラリに依存
  - Xlib, zlib, CLHEP, ROOT, curses, pthread

提供するもの（１）

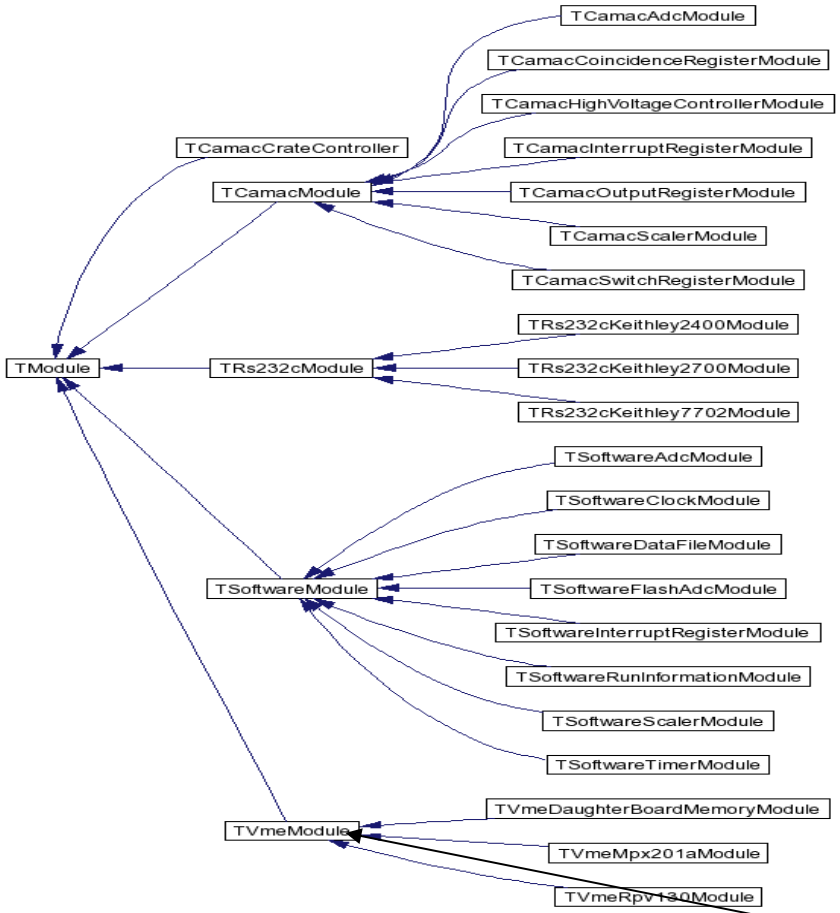
- ライブラリとクラスインターフェース、および、いくつかのスクリプト
  - アプリケーションは提供しないことに注意
- システムコントロールのためのフレームワーク（次ページで説明します）
  - バグの温床になりがち
  - 経験に乏しい人は書いていてつらい部分
  - ９個のフックと９個のステータス


Start()
Dead
Boot()
Shutdown()
Waiting
AtFirst()
WaitTrigger()
Standby
Resume()
Event Loop
Idle
Ready
Taking
Suspend()
AtLast()
Recording
Run Loop
Stop()

提供するもの（2）

- VME、CAMAC、RS232C、GPIB（廃止）のいくつかのモジュールへのアクセス
  - 次ページで説明します
- それらの代替モジュール
  - 乱数やGeant4などのシミュレーション結果を元にデータ発生させる
- 全てのモジュールを一元管理できるクレート
- モジュールをグループ単位で管理する仕組み

なんでも入る賢いクレート



プログラムから見れば、どんな規格の
モジュールだろうがメモリにしか見えない
多重度を持ったグルーピング機能

- ADC: TRG | RUN | EVENT
- TDC: TRG
- DV: EVENT | RUN

サポートされていないモジュールは
いわいにお願いする（対価：ビール一本）

TVmeKatoSpecialModule

提供するもの（３）

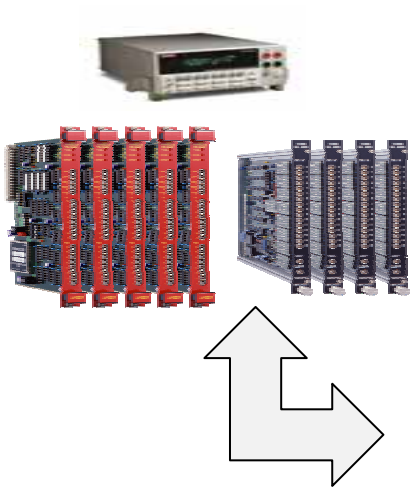
- タイマーやイベントによるスケジューリング
  - マンパワーの足りない実験にはありがたい機能
  - １０００イベント取得時、ランを終了させる
  - ３時間後にランを終了させる
  - 寝て起きたら、データが取れてる！

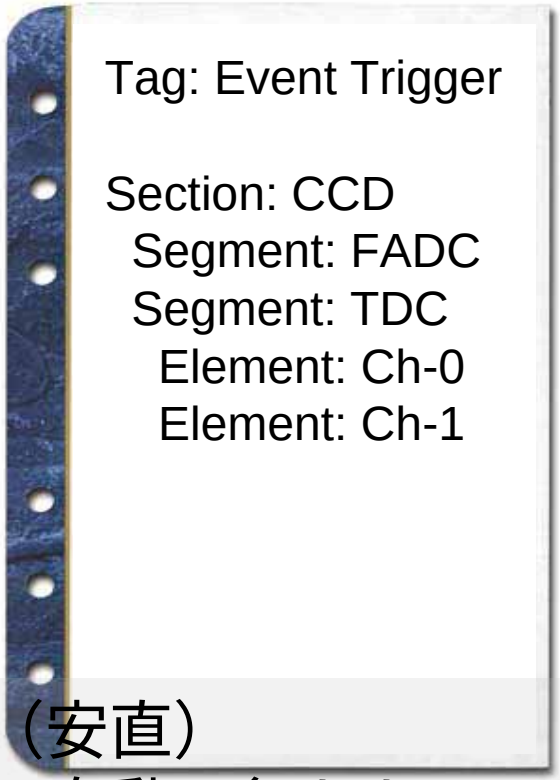
提供するもの（４）

- システムロギング
  - ロガーは作らなかった
    - たかがプロセス１個、動かすのに妙なお約束を強制するのは理不尽
  - システムロガーなんて、たいてい既に動いてるのでそれを利用したほうがいいに決まっている
  - モニタリングツールもたくさんある
- ostreamのように使えるログセンダーを提供
  - Temerg,Talert,Tcrit,Terror,Twarn,Tnotice,Tinfo,Tdebug
- ３つのフックで標準出力(Tcout)、標準エラー出力（Tcerr）、ログ（Tclog）を制御可能
  - CatchStandardOut(), CatchStandardError(), CatchLog()
  - デフォルトでは、std::cout、std::cerr、std::clogが使用されます

提供するもの（5）

- トリガーと読み出しリスト、発生データの関連づけの仕組み
  - ここはいわい自身、不満の残る実装であった

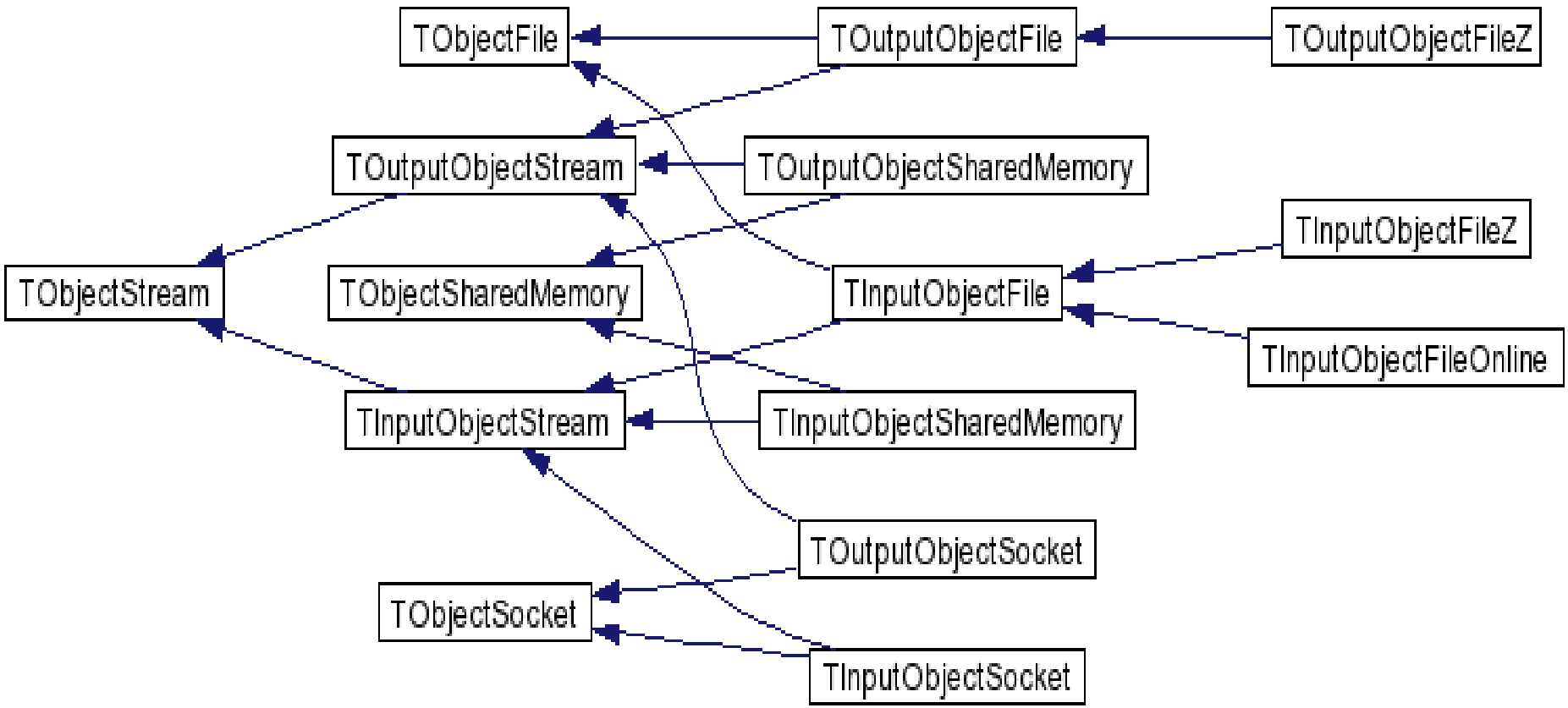




複数の読み出しリストからなる読み出し本（安直）
トリガーとリストのインデックス（関連）は自動で行われる

提供するもの（6）

- オブジェクト永続性
  - ファイル、共有メモリー、ソケットへの入出力機能
- 入力と出力が独立したプラグイン・フィルター

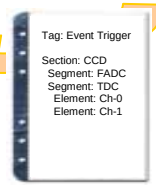


```
// コード中のどこかでイベント識別子の定義をひっそりとおこなう
enum RECORD_TAG {
  EVENT, BEGIN, END
};

// なんの略なのかさっぱりわからない識別子
enum SECTION_TAG {
  GC1, GC2, SC1, SC2, CN
};
```

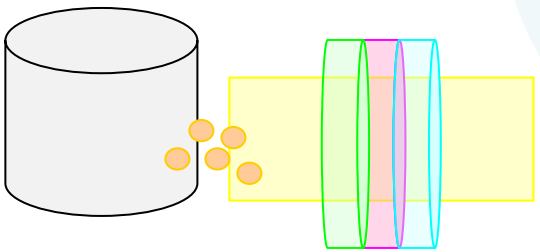
Brief Summary

残された疑問（次ページで話します）
たいていの場合、最大遅延発生箇所はここ

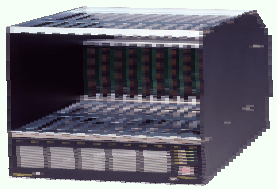
読み出し本



読み出しリスト

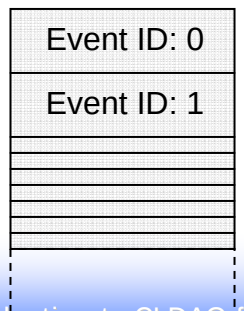


仮想化されたストリームとフィルター



いろいろ入ってるクレート

オブジェクト（データ）



push

ラン中であればいつでも興味のあるイベントを参照可能

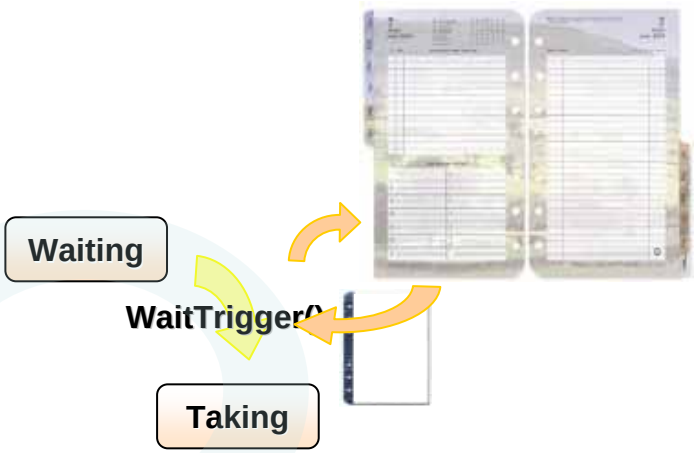
後悔：この部分は並列化のオプションも検討すべきだった

提供しないもの

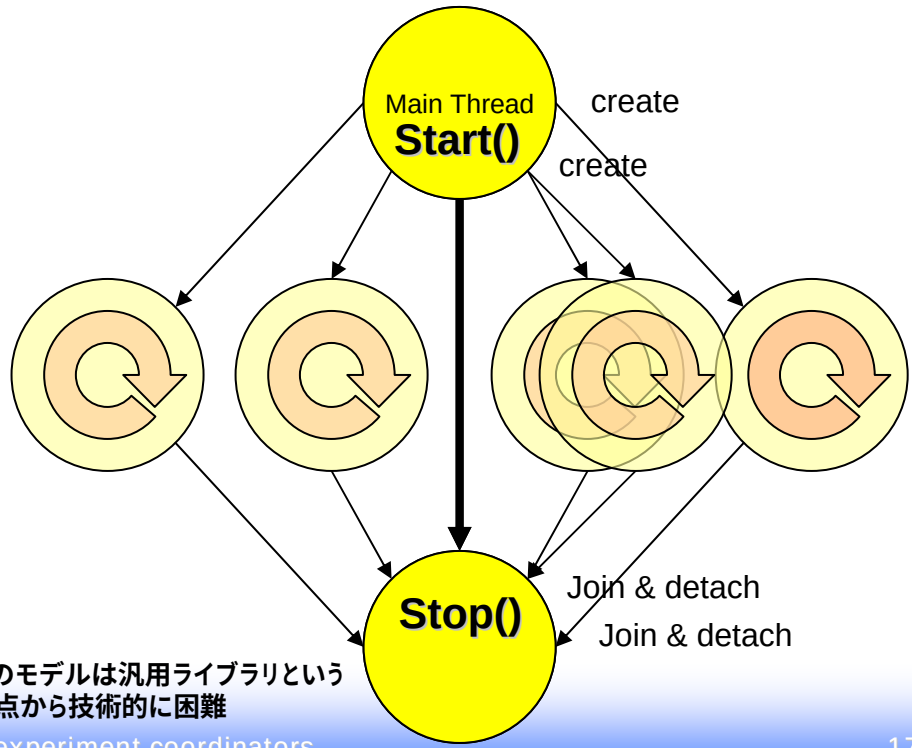
- 解析環境
  - あらゆる解析システムに対応可能なデザインであった
    - 肺炎になり頓挫して以来、放置
    - 興味が他に向いてしまい、やる気なし
- ロガー（収集側）とモニタリング
- イベントディスプレイ
- ランサマライザ
- ほとんどのモジュールへのアクセス機能

技術的な特徴

- 特にないが、POSIXスレッドを採用した
  - 単純なシングルプロセスモデルではイベントループ中（最大遅延発生箇所）にシステムコントロールが困難



Event Loop



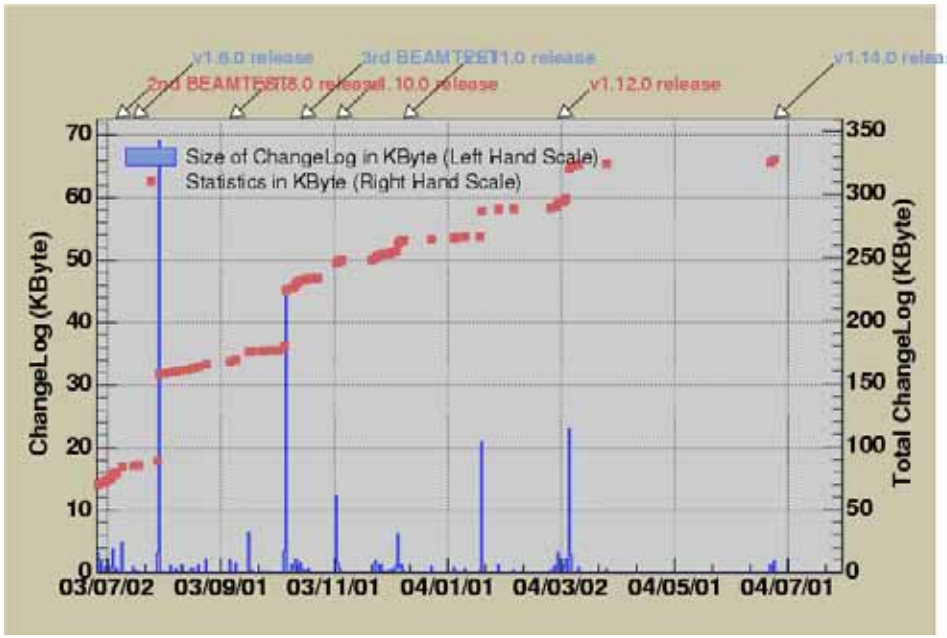
このモデルは汎用ライブラリという観点から技術的に困難

動作原理

- ＣＬＤＡＱの枠組みではいくつかのクラスの実装をプログラマーに半強制する
  - Boot(), Shutdown(), Start(), ...など
  - ラン中のプログラムの動作は実験者しか知りえないから
- ＣＬＤＡＱの枠組みの中で作られたプログラムはユーザー定義の実装を再現するだけ
  - それを便利にするためのいくつかのユーティリティは既に解説しました
  - 実装しなかったら、プログラムは何もしないだけ
- フックの数が足りない場合は自分でマネージメントクラスを実装する（これは大変）

開発の歴史と思い出

開発の頻度を可視化したもの



クラスライブラリの規模

２００２年秋の学会前：肺炎でいちはら病院に入院

２００２年１２月：１回目のビームテスト、CAMACサポート

２００３年：数度のビームテストを経てこのあたりでほぼ完成した

２００３～０４年：高速アクセスや言語バインディング

公開開始	２００３年5月
総アクセス数	４００００回以上
総ダウンロード数	５０００回以上
クラス数	～２００
ソースサイズ	～1MB、～４万行（アプリケーションを含めて１０万行）

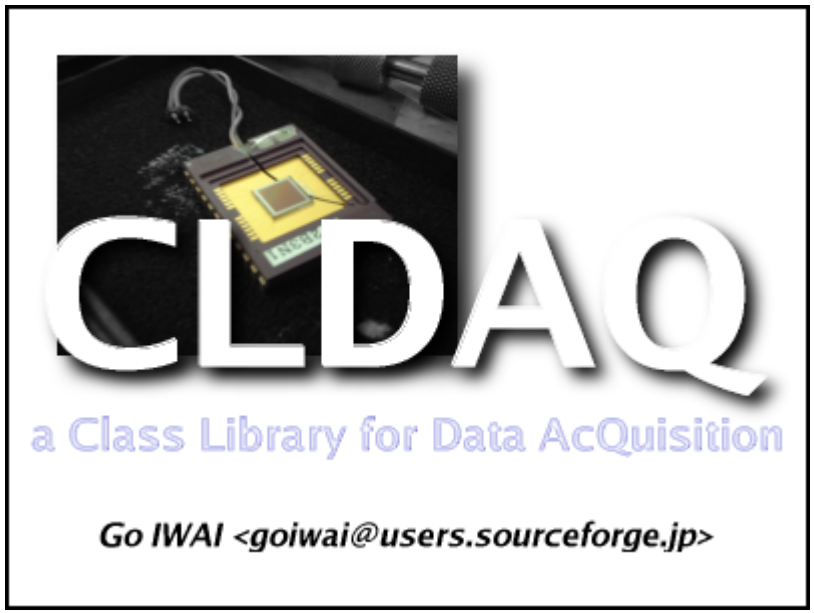
導入事例

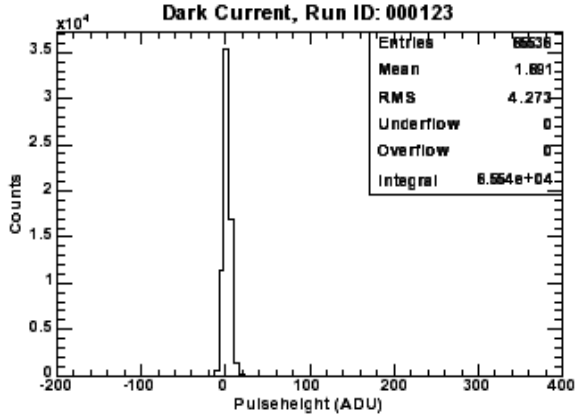
- 複数の種類のＣＣＤ読み出しに採用
  - ＶＭＥ規格のFlash ＡＤＣ（Pentland社製）
  - ＶＭＥ規格のIOレジスタ（林栄精器社製）
  - ＲＳ２３２Ｃ接続されたデジボル（ＫＥＩＴＨＬＥＹ社製）
- ビームエネルギー測定用にＣSI中の光量読み出しに採用
  - ＣＡＭＡＣ規格のいくつかのモジュール（ＡＤＣやスケーラー）
- ビームプロファイル・モニター用にPINホト・ダイオードの電流読み出しに採用
  - ＲＳ２３２Ｃ接続されたデジボル（ＫＥＩＴＨＬＥＹ社製）

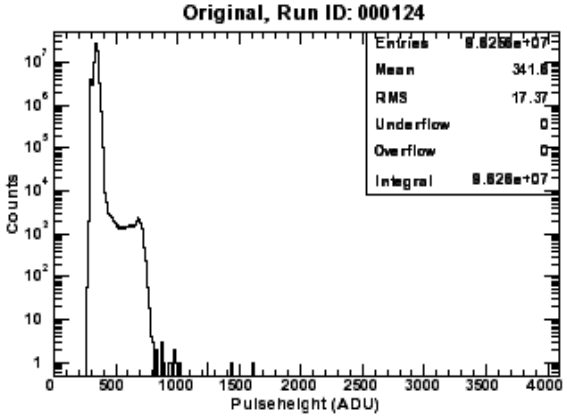
- ＫＥＫオンラインで試験導入（頓挫？）
- Ｋ２Ｋ（頓挫？）

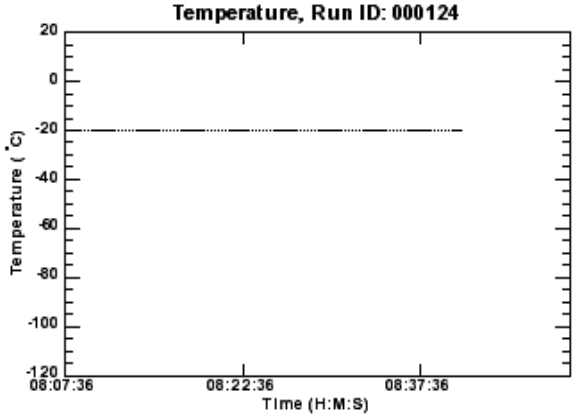
スクリーンショット

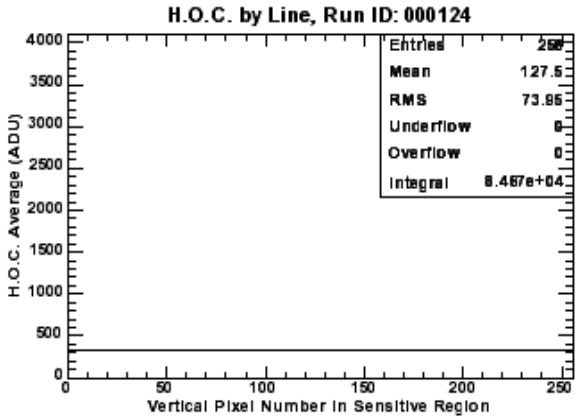
- CLDAQはレンダリング機能はもっていないので、ROOTに強依存しています
  - ROOTの開発チームに感謝しています

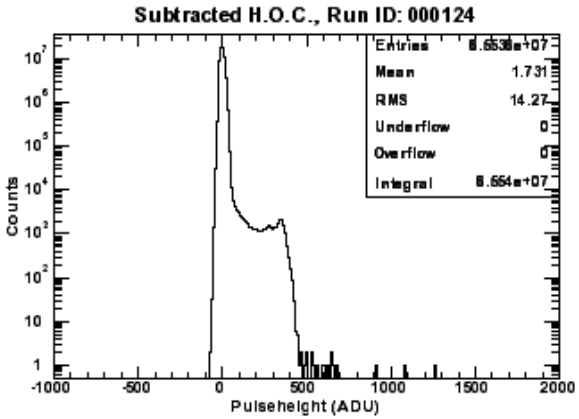


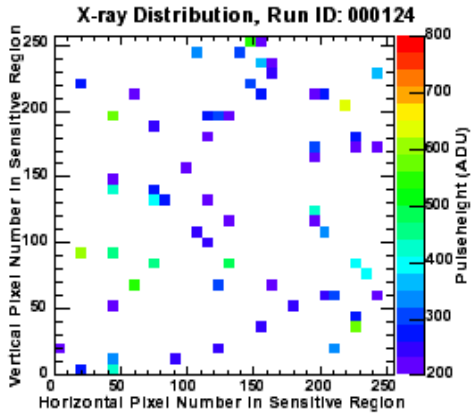


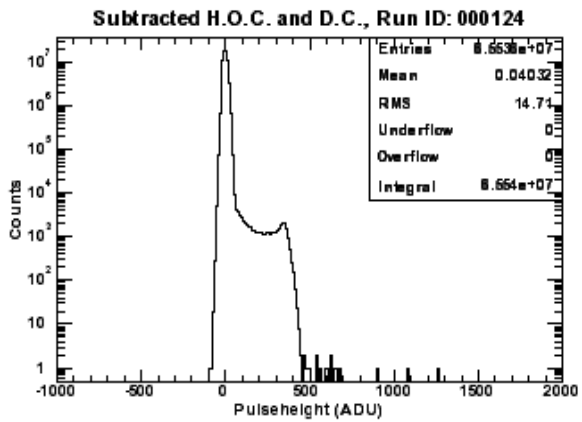


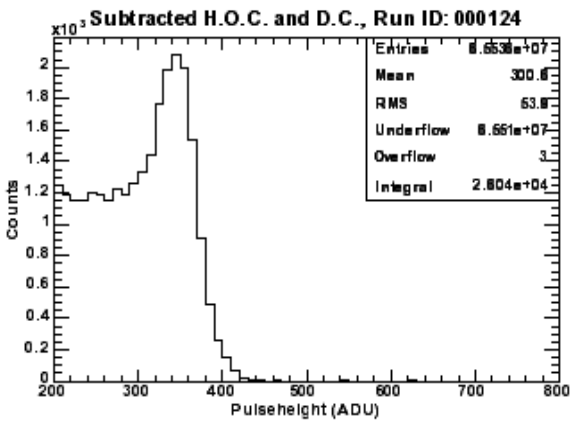












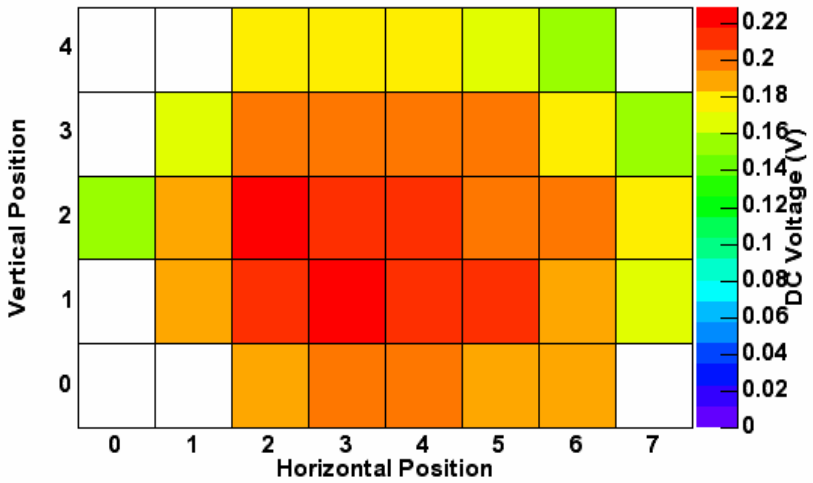
X-ray Peak (ADU): 345

Temperature (°C): -20.2

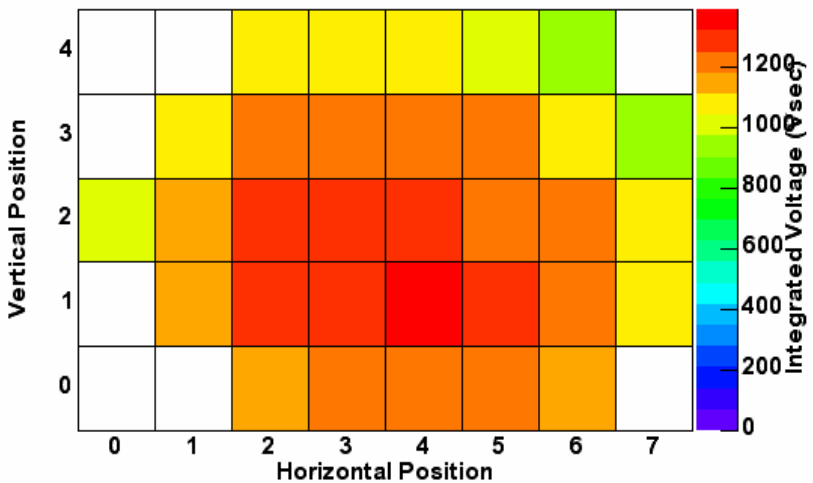
Max H.O.C. (ADU): 368.6

Min H.O.C. (ADU): 321.6

Beam Profile Monitor(Snapshot), Run ID: 000020



Beam Profile Monitor(Integrated), Run ID: 000020



CH00: 0.20395VDC, 7786sec, 1094.65Vsec, 2003/10/20 16:08:54
CH01: 0.201545VDC, 7787sec, 1135.18Vsec, 2003/10/20 16:08:55
CH02: 0.207247VDC, 7787sec, 1147.03Vsec, 2003/10/20 16:08:55
CH03: 0.194257VDC, 7787sec, 1070.84Vsec, 2003/10/20 16:08:55
CH04: 0.173163VDC, 7787sec, 999.965Vsec, 2003/10/20 16:08:55
CH05: 0.193782VDC, 7787sec, 1132.45Vsec, 2003/10/20 16:08:55
CH06: 0.225767VDC, 7787sec, 1241.11Vsec, 2003/10/20 16:08:55
CH07: 0.222328VDC, 7788sec, 1284.29Vsec, 2003/10/20 16:08:56
CH08: 0.22801VDC, 7788sec, 1269.13Vsec, 2003/10/20 16:08:56
CH09: 0.222136VDC, 7788sec, 1246.8Vsec, 2003/10/20 16:08:56
CH10: 0.203338VDC, 7788sec, 1146.9Vsec, 2003/10/20 16:08:56
CH11: 0.174042VDC, 7788sec, 992.816Vsec, 2003/10/20 16:08:56
CH12: 0.17969VDC, 7789sec, 1020.72Vsec, 2003/10/20 16:08:57
CH13: 0.212875VDC, 7789sec, 1189.71Vsec, 2003/10/20 16:08:57
CH14: 0.244835VDC, 7789sec, 1344.75Vsec, 2003/10/20 16:08:57
CH15: 0.243554VDC, 7789sec, 1327.14Vsec, 2003/10/20 16:08:57
CH16: 0.240972VDC, 7789sec, 1340.25Vsec, 2003/10/20 16:08:57
CH17: 0.220774VDC, 7790sec, 1254.35Vsec, 2003/10/20 16:08:58
CH18: 0.220983VDC, 7790sec, 1263.05Vsec, 2003/10/20 16:08:58
CH19: 0.202979VDC, 7790sec, 1135.88Vsec, 2003/10/20 16:08:58
CH20: 0.212744VDC, 7790sec, 1204.86Vsec, 2003/10/20 16:08:58
CH21: 0.240044VDC, 7790sec, 1318.9Vsec, 2003/10/20 16:08:58
CH22: 0.244166VDC, 7791sec, 1379.4Vsec, 2003/10/20 16:08:59
CH23: 0.236497VDC, 7791sec, 1388.09Vsec, 2003/10/20 16:08:59
CH24: 0.234726VDC, 7791sec, 1332.51Vsec, 2003/10/20 16:08:59
CH25: 0.217474VDC, 7791sec, 1270.23Vsec, 2003/10/20 16:08:59
CH26: 0.191345VDC, 7791sec, 1095.74Vsec, 2003/10/20 16:08:59
CH27: 0.206426VDC, 7792sec, 1234.81Vsec, 2003/10/20 16:09:00
CH28: 0.225157VDC, 7792sec, 1296.68Vsec, 2003/10/20 16:09:00
CH29: 0.224681VDC, 7792sec, 1273.38Vsec, 2003/10/20 16:09:00
CH30: 0.215701VDC, 7792sec, 1246.64Vsec, 2003/10/20 16:09:00
CH31: 0.213871VDC, 7792sec, 1178.4Vsec, 2003/10/20 16:09:00

CLDAQ Record Display

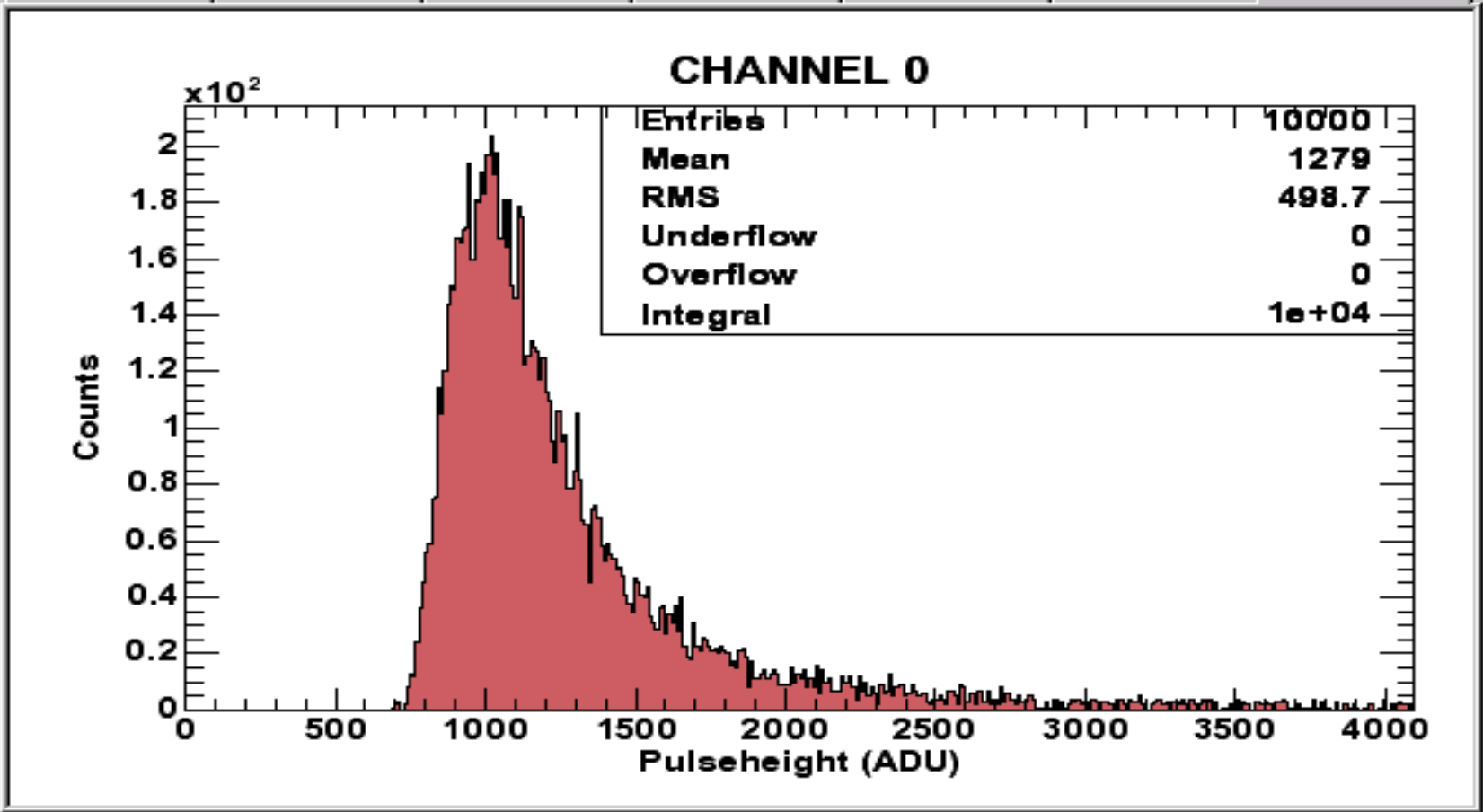
File Edit View

EVENT | TIMEOUT | RUN BEGIN | RUN END

INFO | CDC | CAL | CCD

SCALER | INTERRUPT | ADC 0 | ADC 1

CHANNEL 0 | CHANNEL 2 | CHANNEL 3 | CHANNEL 4 | CHANNEL 5 | CHANNEL 6 | CHANNEL 7

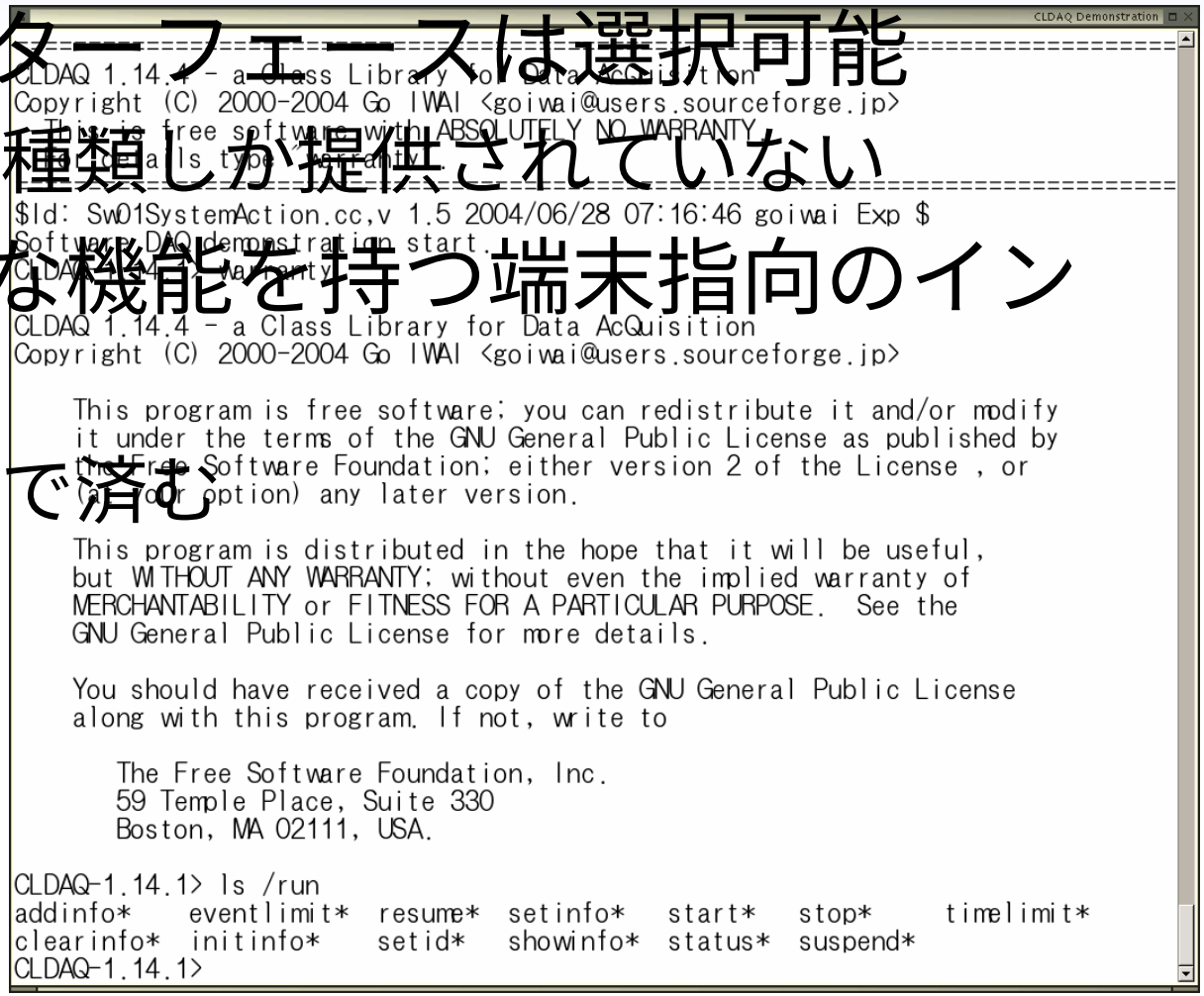


デモります

予定時間　５分

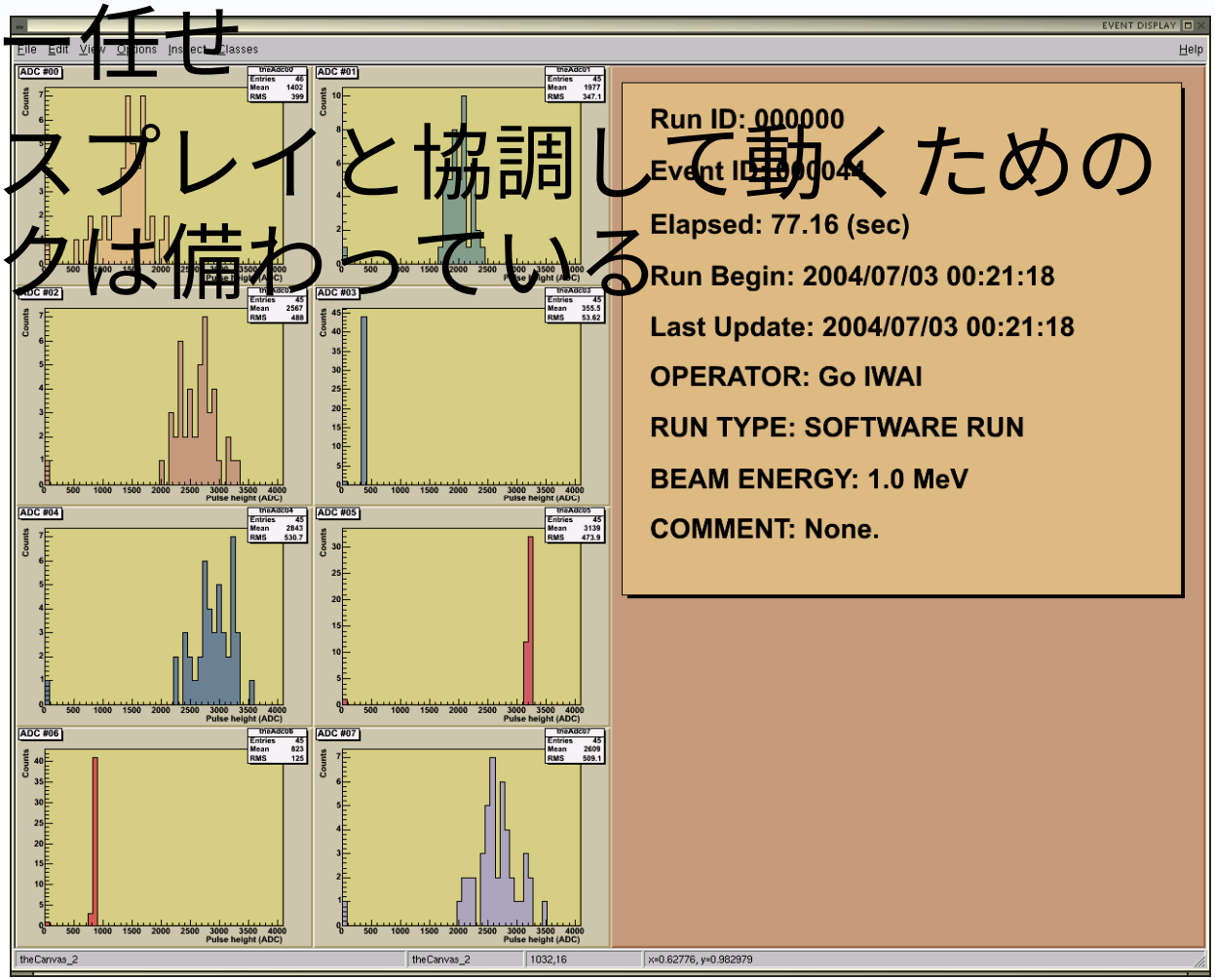
インターフェース

- ユーザーインターフェースは選択可能
  - 残念ながら１種類しか提供されていない
- シェルのような機能を持つ端末指向のインターフェース
  - たいていこれで済む



イベントディスプレイ

- 完全にユーザー任せ
- イベントディスプレイと協調して動くための
  フレームワークは備わっている



ランサマリー

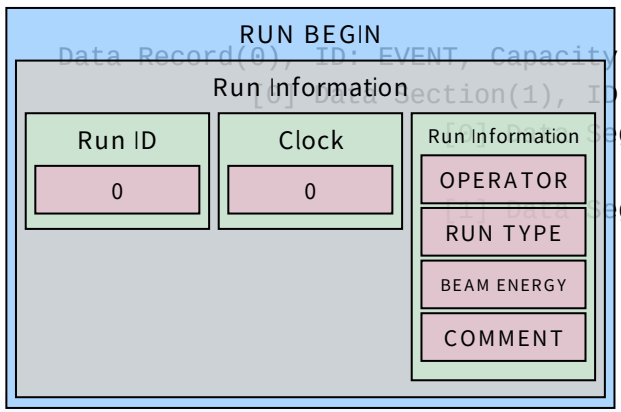
- 残念ながらあまりかっこよくない
- ハードコーディング多く不便

```
void g4view( const char* filename, int update )
{
  gROOT->Reset();
  if ( !gROOT->GetInterpreter()->IsLoaded("libIO.so") ) {
    gSystem->Load("libIO.so");
  }
  cv = new TCanvas( "cv", "Geant4 View" );
  hist = new TH1D( "hist", "Beam Energy Spectrum", 6, 0, 6 );
  hist -> Draw();

  file = new TInputObjectFileOnline( filename );
  TDataRecord record;
  Tint n = 0;
  while ( file -> Read( record ) ) {
    if ( record == "EVENT" ) {
      Tdouble ccdid, edep;
      record[0][3][0].StorePrimitive(ccdid);
      record[0][3][1].StorePrimitive(edep);
      hist -> Fill( ccdid, edep );
      if ( update > 0 && n % update == 0 ) {
        cv -> Modified();
        cv -> Update();
      }
      n ++;
    } else if ( record == "RUN END" ) {
      break;
    }
  }

  cv->Modified();
  cv->Update();
  cv->cd();
}
```

データの標準出力

- 単純に標準出力してもある程度までは理解できる
  - std::ostream << record << std::endl

```
Data Record(0), ID: RUN BEGIN, Capacity: 1, Entry: 1
            [0] Data Section(1), ID: Run Information, Capacity: 3, Entry: 3
                        [0] Data Segment(2), ID: Run ID, Capacity: 1, Entry: 1
                                    [0] Data Element(3), ID: 0, Type: 0/Tint(4b), Data(1): 0
                        [1] Data Segment(2), ID: Clock, Capacity: 1, Entry: 1
                                    [0] Data Element(3), ID: 0, Type: 1/Tstring, Data(1): 2004/07/03 00:22:26
                        [2] Data Segment(2), ID: Run Information, Capacity: 4, Entry: 4
                                    [0] Data Element(3), ID: OPERATOR, Type: 1/Tstring, Data(1): ***
                                    [1] Data Element(3), ID: RUN TYPE, Type: 1/Tstring, Data(1): ***
                                    [2] Data Element(3), ID: BEAM ENERGY, Type: 1/Tstring, Data(1): ***
                                    [3] Data Element(3), ID: COMMENT, Type: 1/Tstring, Data(1): ***
Data Record(0), ID: EVENT, Capacity: 1, Entry: 1
            [0] Data Section(1), ID: Event Data, Capacity: 5, Entry: 5
                        [0] Data Segment(2), ID: Event ID, Capacity: 1, Entry: 1
                                    [0] Data Element(3), ID: 0, Type: 0/Tint(4b), Data(1): 0
                        [1] Data Segment(2), ID: ADC, Capacity: 16, Entry: 16
                                    [0] Data Element(3), ID: 0, Type: 0/Tint(4b), Data(1): 0
                                    [1] Data Element(3), ID: 1, Type: 0/Tint(4b), Data(1): 0
                                    [2] Data Element(3), ID: 2, Type: 0/Tint(4b), Data(1): 0
                                    [3] Data Element(3), ID: 3, Type: 0/Tint(4b), Data(1): 0
                                    [4] Data Element(3), ID: 4, Type: 0/Tint(4b), Data(1): 0
```



デモの規模

- 用意したクラスは８つ（必須ではない）
- 実装も含めてせいぜい１０００行程度でほとんど全てが完結

```
% wc Sw01*.hh Sw01*.cc
      40      86    1002 Sw01CrateDefinition.hh
      43     102    1131 Sw01EventAction.hh
      48      89    1038 Sw01EventDisplay.hh
      41      92    1056 Sw01Geant4ViewCommand.hh
      42      90    1104 Sw01ReadoutBookDefinition.hh
      80     176    1969 Sw01RunAction.hh
     141     314    3240 Sw01RunSummary.hh
      55     113    1380 Sw01SystemAction.hh
      87     312    3378 Sw01CrateDefinition.cc
      91     259    2660 Sw01EventAction.cc
     162     518    4524 Sw01EventDisplay.cc
      54     164    1738 Sw01Geant4ViewCommand.cc
      73     262    2417 Sw01ReadoutBookDefinition.cc
     168     460    4718 Sw01RunAction.cc
      57     161    1803 Sw01RunSummary.cc
      58     141    1527 Sw01SystemAction.cc
    1240    3339   34685 total
```

デモの始まりから終わりまで

```
// ランマネージャーとインターフェース
TRunManager* manager = new TRunManager( new TTerminalUserInterface() );

// アクションクラスの登録
manager -> SetSystemAction( new Sw01SystemAction() );
manager -> SetRunAction( new Sw01RunAction() );
manager -> SetEventAction( new Sw01EventAction() );

// クレートと読み出しリストの登録
manager -> SetCrateDefinition( new Sw01CrateDefinition() );
manager -> SetReadoutBookDefinition( new Sw01ReadoutBookDefinition() );

// コマンドの登録
manager -> SetUserCommand( new Sw01Geant4ViewCommand() );

// DAQセッションの開始
manager -> SessionStart();

// ランマネージャーの解放
delete manager;
```

まとめ

- ＣＬＤＡＱはライブラリです
  - ヘッダファイルとオブジェクトモジュールを提供しますが、アプリケーションは提供しません
  - アプリケーション（ＤＡＱシステム）を作るためには少なくとも１０００行程度のC++でコードを書かなくてはなりません
  - 開発環境や開発を短期間に行うためのユーティリティ、およびＤＡＱフレームワークは提供しますが、アプリケーションそのものは提供しません
- 数回にわたるビームテスト、数千時間にわたるＣＣＤオペレーションに於いて、ただの一度も落ちて（異常終了）いません
  - 落ちないといって、落ちるプログラムはたくさんあるので、これは驚異的な成果です
- 公開開始から５０００回以上ダウンロードされ、プロジェクトホームページには４００００回以上のアクセスがありました
- １行も書かずに現場（東北大核理研）に臨んでも、翌朝にはちゃんとプログラムができてしまい、マシンタイム中のトラブルもなし
  - 開発期間短縮の証明

今後の予定

- 機能面ではもはやこれ以上増やすものはない
  - ややこしくなるというデメリットの方が大きい
- 設定ファイル（プレインテキスト）からオブジェクトを作成
  - 構文解析器が必要
  - 正規表現をC++からPerlライクに扱いたい（これは実はあるがまだバギー）
    - Boostの存在を早くから知っていれば必要なかった
- 理想なのはファイルを渡せば勝手に動くようなアプリケーション、いったい何のファイルにすればいいのか？
  - 悩ましい
- 言語バインディング
  - 現在はCINTのみ、しかも一部
  - Python, Perl, PHP&XMLのWeb Service
- ライブラリ開発からアプリケーション開発タームへと移行
  - より、人間に近いレイヤー
- Taking～Recordingは並列化のオプションを選択可能なデザインにすべき
- 他のプロセスとの協調動作の可能性
  - 現在は多重プロセス環境に置かれたときの通信ポートはデフォルトでは用意されていない

いろいろ教えてください

- アプリケーション構築にあたって、ＤＣＢＡサイドでの要求をまとめておく必要があります
- チャンネル数
- イベントサイズ
- トリガーレート
- システム構成
  - ＯＳ、コンパイラ、ＨＤＤ、ネットワーク、プリンタの有無、ファイヤーウォールの形状
- イベントの種類
- 解析はどうしたいのか？イメージをなるべく具体的に
  - ＳＲＢ等、ミドルウェアは必要か？
- 入力はどの程度あるのか？
- ディスクＩＯの制約はありそうか？
  - NFS:10MB/sec,　LOCAL:50MB/sec, GRID:∞

もっと教えてください

- ロジックがあればください
- エレキとドライバ
- ファイルサイズを小さくする必要があるか？
- 予想される年間の運転時間は？
- ＧＵＩはないけどいいのか？
- イベントディスプレイとして必要な絵
- ROOT, gcc, kernelの最新版を追いかけたいか？
- 制御、モニターの端末は複数存在するほうがよいか？
- C++プログラマーはいるのか？人数と経験値は？
- 「ＤＡＱが不調だ、今すぐ来て」と言われても行けないことがあるかもしれないけど、いいですか？