





Session Description

What we will cover in this session

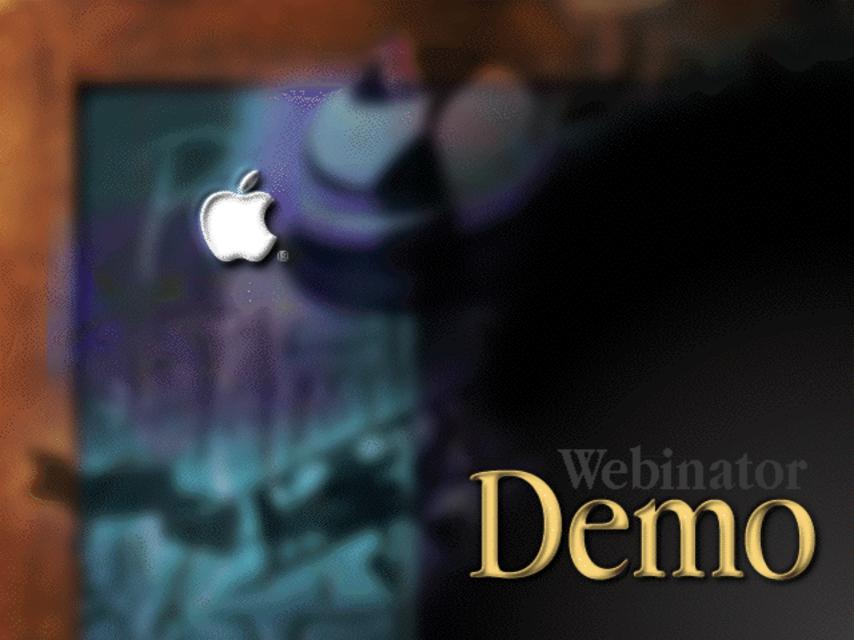
- Details on some new features
 - Networking
 - Platinum appearance
 - Threading
- And some words on
 - Transition to Release 13





Impress your co-workers with little effort

- Designed for networking newbies
- Designed for networking gurus
- Abstracted API
- Flexible object-oriented subsystem





Webinator — ctor

```
CWebinatorApp::CWebinatorApp()
     TApplication('?????', '?????'),
      fRecievingData(0)
  fConnections
   = TH new CMultiTCPConnectionManager AC;
  CReactor AC* aReactor
      = fConnections->GetReactor();
  aReactor->AquireIdleTimeEngine(
                  TH new TIdleTimeEngine);
  MTCPConnectionEventReceiver AC::
    RegisterForEventsFrom(fConnections);
  fConnections->ListenTo(
              CInternetAddress AC::kHTTP);
```

Receiving Data — Preparation



Receiving Data

```
void CWebinatorApp::DoReceiveData(
    CTCPConnection AC* inSession,
    CNetworkTransaction AC* inRcvr
 CFSSpec AC spec; CTextParser AC parser;
 CTextParser AC::CToken token =
     parser.Start(
          inRcvr->GetStream().GetReadPositionPtr());
 if(token == "GET")
    token = parser.GetNextToken();
    if(token == "/")
      spec.SetName(CStr63 AC("index.html"));
    else
      spec.SetName( CStr63 AC(token+1));
      // Advance over length byte
  inSession->Send( TH new FileSender AC(spec,
                   CFileSender AC::kDataFork ));
  inSession->Send(TH new CDisconnect AC );
```



For simple tasks

- Simple API inspired by Java
- Example: CTCPSocket_AC
- Blocking, synchronous calls
- Thread savvy



For complex tasks

- Asynchronous API
- Polymorphic network abstractions
- Manage multiple connections or sessions simultaneously
- Helper classes for managing single or multiple connections or sessions
- Queueable net transactions



Platform, framework and provider agnostic

- Chooses provider at run-time
 - MacTCP
 - OpenTransport
 - WinSock
 - BSD Sockets
- Not tied to the Macintosh platform
- ...we'd prefer that you use MacApp!



Uses modern C++'isms

- Auto pointers
- Reference counting auto pointers
- Templates
- Native exceptions
- Several Design Patterns
- Modern casting operators



Stream-based I/O

- Streams
- Sources/Sinks
- All compatible with MStreamable_AC



Features available today

- The "plumbing"
- Internet aware



Today: The "Plumbing"

- Asynchronous event handling
- Service handlers
- Service handler managers
- Streams, sources and sinks
- Buffer objects
- Threading savvy



Today: The "Plumbing" (cont.)

- Transaction classes for sending and receiving files and data
- Text parsing and byte iterators
- Timers
- Abstracted addressing
- Strategy classes



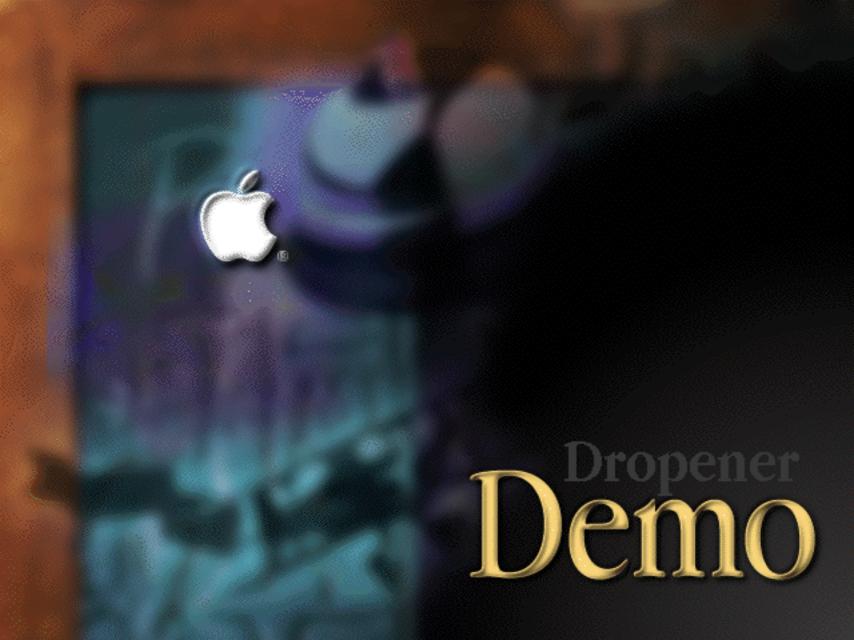
Today: Internet aware

- Internet DNR services
- TCP connections
- OpenTransport/MacTCP



Future features

- WinSock
- BSD Sockets
- AppleTalk
- UDP
- TCP Multicasting
- FTP, SMTP, POP, &c.
- Supporting views in MacApp
- Let us know what you'd like





Receive Data — Preparation

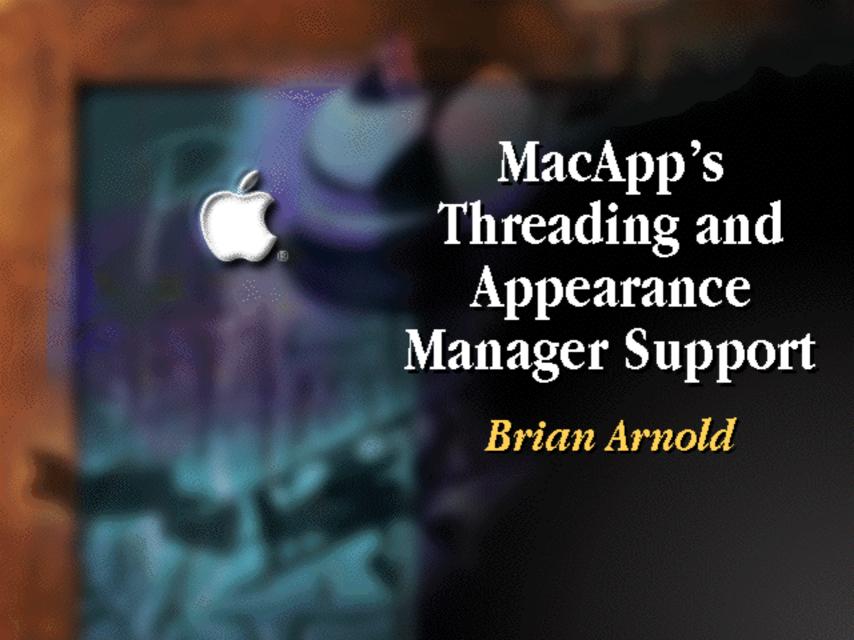


Receive Data

```
if(inTransaction->IsComplete())
 CFileReceiver AC* fileReceiver
      = dynamic cast<CFileReceiver AC*>
                              (inTransaction);
  ::ThrowIfNULL_AC(fileReceiver);
 TFile* file = new TFile;
  file->Specify(fileReceiver->GetFileSpec());
  HandleReceivedFile(file);
```

Sending Data

```
TemplateIterator AC<TFile*> iter(aFileList);
TFile * aFile = iter.First();
while (aFile)
  CRefCountingPtr AC<CServiceHandler AC>
      newConnection = fConnections->ConnectTo(
         CInternetAddress AC(remoteAddressStr,
                             remotePort));
  CFileSender AC* sender
    = TH new CFileSender AC(aFile->GetFileSpec(),
                    CFileSender AC::kEntireFile);
  newConnection->Send(sender);
  aFile = iter.Next();
```





Appearance Manager

Platinum Appearance

- ACS appearance classes
- New MacApp views
- Ad Lib templates





Overview

- Multitasking within an application
- Improved responsiveness
- Synchronization of shared resources



Quick and easy threaded applications

- Simplified CommonPoint threads
- CThread base class
- Convenience classes
 - CThreadRunner
 - CWhileSynchronized and CMonitorLock
- Platform agnostic
- Flexible object-oriented subsystem
- Thread Manager, MP, Win32 support



CTbread

- Base class for your threads
- Implements
 - Run(), and
 - A bool for shutdown



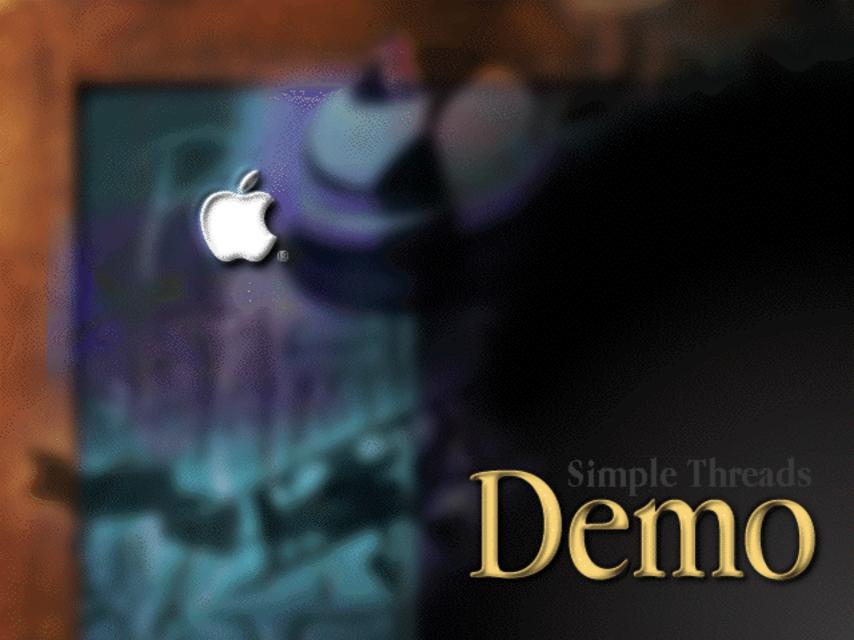
CTbreadRunner

- Convenience class
- Accepts thread and runs it
- Provides
 - State inquiry
 - Shutdown request
 - Join (wait for) its thread



CWhileSynchronized and CMonitorLock

- Stack-based, accepts CMonitorLock
- Acquires lock during ctor
- Releases lock in dtor





Pardon the Interruption

Contextual menus

- Expose selection content to plug-ins
- Data Detectors plug-ins
 - Maps content and patterns to actions



```
// Thread implementation
class TSimpleThread : public CThread AC
public:
   TSimpleThread(TSimpleThreadView*
                        itsView = NULL);
   virtual ~TSimpleThread();
   virtual void Run();
private:
   TSimpleThreadView* fView;
```



```
// Thread implementation
void TSimpleThread::Run()
   int iter = 0;
   if (fView != NULL)
      while (!ShutdownRequested())
         fView->ThreadedDraw(iter);
         Yield();
```





FracAppMP



FracAppMP

```
bool CFracAppEngine::CalcCity()
    // Scoped to unlock as soon as able.
    // CalcCity is called from within
    // multiple threads, so synchronize
    // when updating the current location.
    CWhileSynchronized AC entry(fLock);
    fCurrentLocation.v++;
```



ACS's Threading Suite

Threads, MacApp and the Future

- Use threads with any framework
 - On any platform
- Threaded idle and reactor patterns
- Threaded MacApp event handling
- Rhapsody thread support





ACS's TidyHeap Tools

Diagnostic tool

- Suite of C++ classes and operators
- Tracks and verifies memory blocks
- Reports undeleted allocations
- Flexible and extensible



ACS's TidyHeap Tools

Diagnostic tool

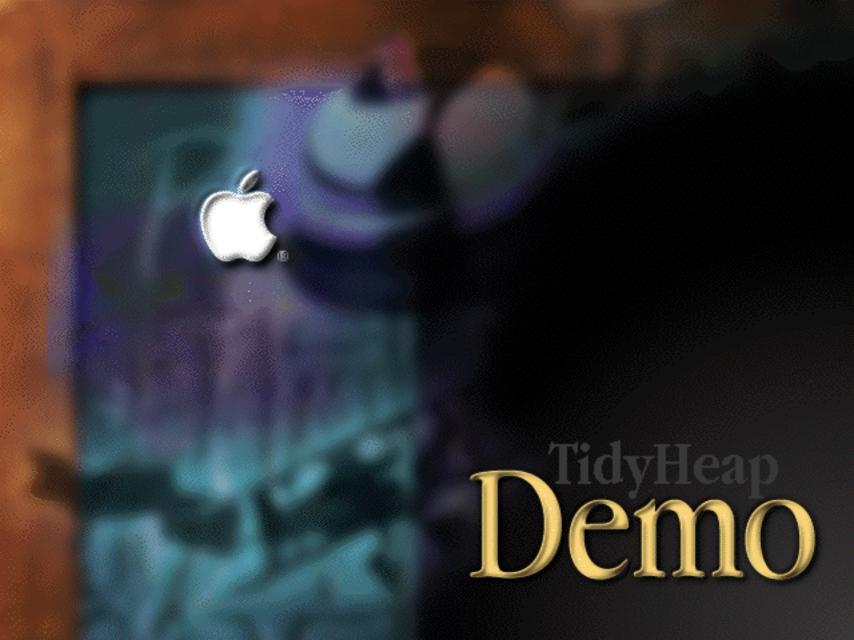
- Detects double deletions
- Garbage-in action class
- Garbage-out action class
- Force new to fail
- Verify selected or all blocks
- More...



ACS's TidyHeap Tools

Support Tools

- TidyHeap Director
- NewDropper





Transitioning to Release 13

Here are some thoughts

- One or two weeks typical
- MacApp 3.3 or R12 as a base
- Manual intervention required
- Conversion scripts and tools



Transitioning to Release 13

Fundamental changes

- I-methods -> ctors
- Free -> dtors
- Fail -> Throw/Catch
- ACS naming convention
- Scope qualification

