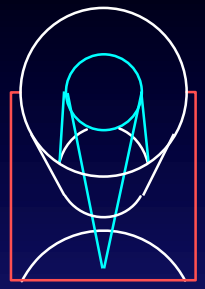
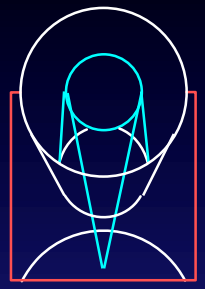


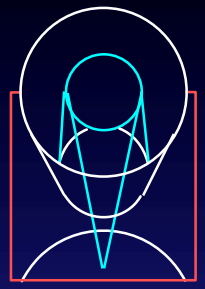
Orbit Planner Development Plan

December 18, 2000



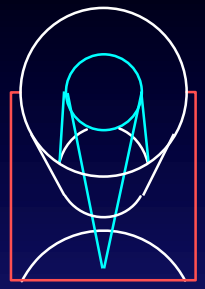
Orbit Planner Development Plan

- High level goals
- Basic functionality
- Initial Design Issues
- Implementation Plan



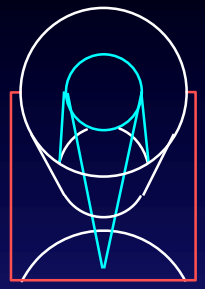
High Level Goals

- Release usable, but unofficial orbit planner by January, 2002 (cycle 11)
- Release official orbit planner by January, 2003 (cycle 12)



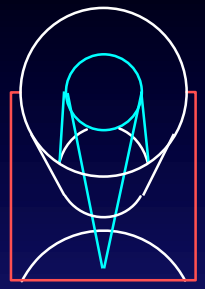
Basic Functionality

- Plan HST orbits in APT GUI environment
  - Share exposure/target data with other APT tools
  - Allow user input to Trans
  - Display output from Trans



Initial Design Issues

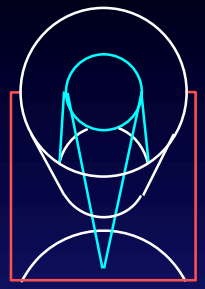
- Where will the Trans processes run?
  - Options:
    - On the local machine
    - On a remote server
- What sort of interface will be used to communicate with Trans?
  - Options:
    - TCP/IP sockets
    - CORBA
    - Other…



Where will Trans run?

Plan:

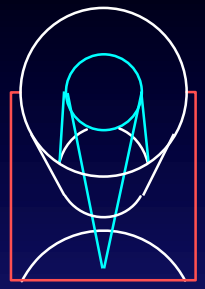
- Prefer running locally, to optimize performance
- For cycle 11 release, run locally on Solaris, and include Windows support if time allows.
  - Not supporting remote servers minimizes development/maintenance costs
  - Recognizes trend towards cheaper, faster computers
  - Gives us more time to make decisions about other platforms and remote servers for cycle 12
- We may support remote Trans server for cycle 12, but encourage running Trans locally.
  - Limiting server use limits our maintenance effort



Communications Interface – Sockets vs. CORBA

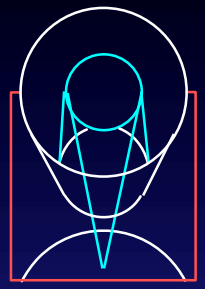
CORBA implements low level communications

- Saves us development/maintenance time
  - Don’t have to build initial comm. infrastructure
  - Less to do for routine maintenance and enhancements
  - Interface is more mature than what we would write
- Gives higher risk due to dependence on 3rd party software
  - Low level problems harder to debug and fix



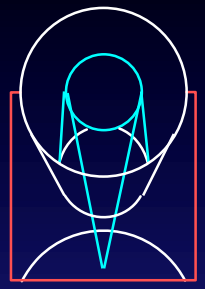
Communications Interface – Choice

- Use CORBA
  - Speed up development
  - Have more mature, general purpose interface
- Reduce project risk by exercising interface early
  - Leaves time to recognize problems; adopt alternate solutions



Implementation Plan: Manage Risks

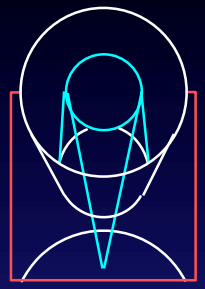
- Dependence on Trans development
- CORBA low-level comm. problems
- CORBA implementation issues
- Platform dependencies
- Unfriendly GUI
- Missing release deadlines



Manage Trans Development Risks

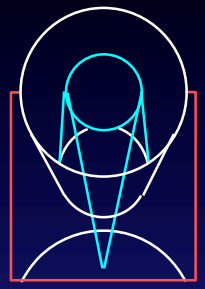
The Orbit Planner Project will **FAIL** without needed Trans development.

- Work with PDT to determine essential functionality for cycles 11 and 12
- Plan development of those functions
- Allow time to rework slow features



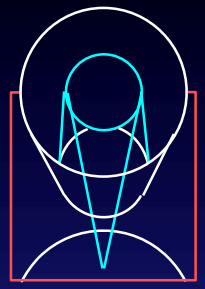
Sample of Trans features likely needed by orbit planner (and not found in Qwik-Trans prototype).

- Buffer management
- ACS auto-parallels
- During orbit packing, account properly for Alignment overheads, ovratalstart and ovratalend
- Constraint sequencing all input
- WFPC2 exp time adjustments



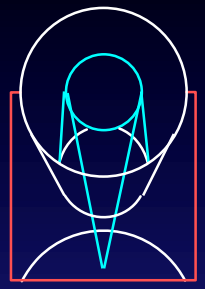
Sample of Trans features likely needed in orbit planner (found in QT, but implemented too crudely)

- Bypass old Trans code
- “Search engine” design/infrastructure
- Exposure ordering
- Merging
- Exposure offsets within alignments
- Some kind of constraint ordering
- Recognize coord. parallels before merging
- Isolated efficiency improvements



Manage CORBA Risks

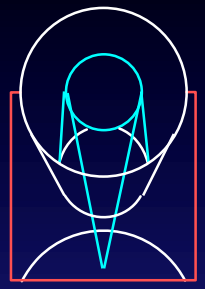
- Build an I/F using Java/CORBA/Qwik-Trans (robust enough to do legitimate testing)
  - Unexpected implementation issues discovered here
  - Can be done without GUI
  - NOT throwaway code
- Run automated tests such as the QT test suite
  - Most communication problems would be discovered very early in the project.
- Test responses to simulated comm. failures



Manage CORBA Risks

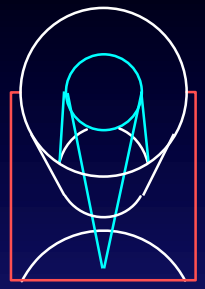
Dealing with problems found in testing:

1. Limit time spent dealing with 3rd party CORBA software problems.
2. When time limit exceeded, drop CORBA and use socket solution.
3. Socket implementation would displace other work, probably support for new platforms.



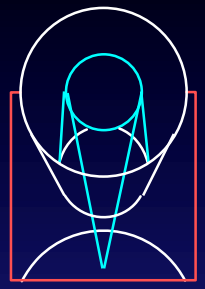
Manage Platform Dependency Risks

- Revise Trans/QT to build on a PC
- Run QT regression tests on PC



Manage GUI Quality

- Work with user groups to define key features and interface
- Develop usable subsets of features, so they can be evaluated before the whole product is ready



Manage Deadline Risks

- Developing subsets of each type of major feature early will identify problem areas before they become schedule breakers
- Functionality of cycle 11 release is flexible
- Avoid unnecessary process overhead
- Periodically review progress towards goals

Orbit Planner Development Schedule

ID	Task Name	Duration	Start
1	**GUI - Development Cycle**	**235 days**	**Mon 12/11/00**
2	Comm. I/F Development	8 wks	Mon 12/11/00
3	Comm. I/F Testing	6 wks	Mon 1/22/01
4	Buy Lisp for Windows	2 wks	Mon 12/11/00
5	Evaluate Lisp on Windows	6 wks	Mon 2/26/01
6	Plan TransVERSE Functionality	6 wks	Mon 12/11/00
7	Evaluate existing orbit planner code/GL	10 wks	Mon 12/11/00
8	Requirements analysis	16 wks	Mon 12/11/00
9	Initial GUI Development	13 wks	Mon 4/2/01
10	GUI Rework	18 wks	Mon 7/2/01
11	TransVERSE Rework	18 wks	Mon 7/2/01
12	Testing - Cycle 11	9 wks	Mon 11/5/01
13	Release - Cycle 11	0 days	Fri 1/4/02
14	Fill in Gaps; Support new TransVERSE featu	43 wks	Mon 1/7/02
15	Testing for Cycle 12	9 wks	Mon 11/4/02
16	Release - Cycle 12	0 days	Fri 1/3/03