Visualization Toolkit Extreme Testing

A Production Release Every Day

Bill Lorensen
Jim Miller

GE Corporate Research and Development
Niskayuna, NY

lorensen@crd.ge.com
millerjv@crd.ge.com

Outline

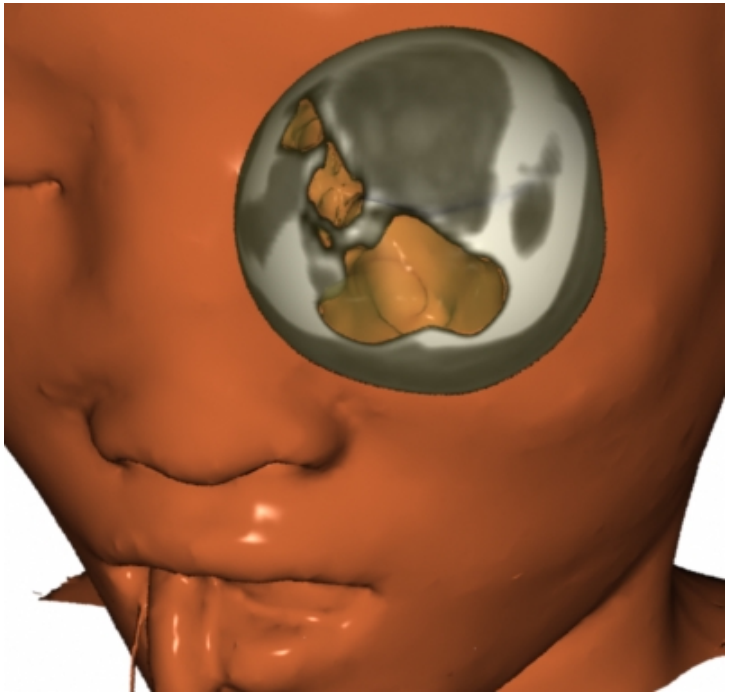
The Visualization Toolkit

Motivation

Software Process

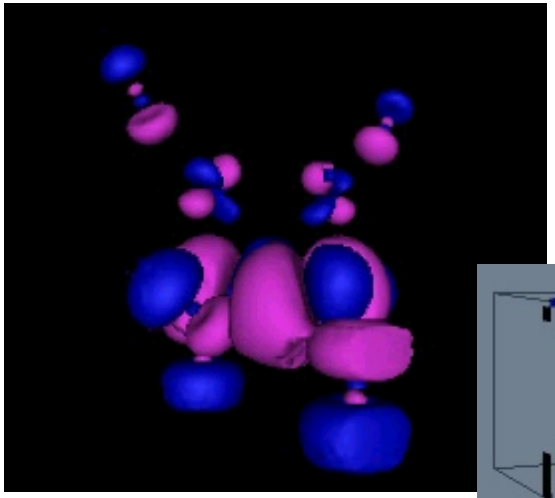
Automated Testing

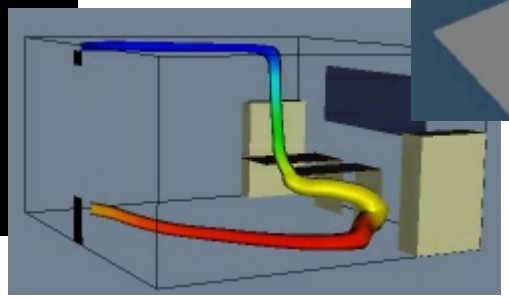
Discussion

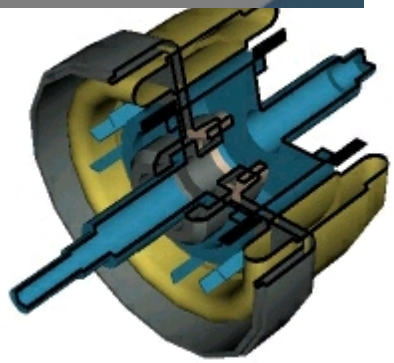


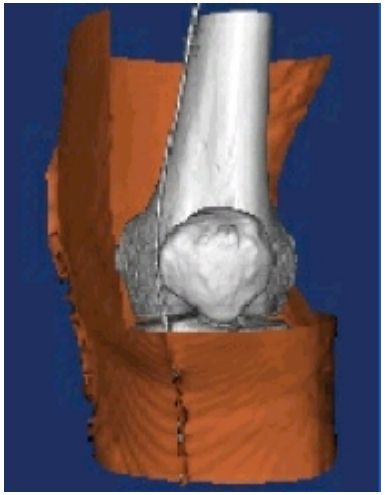
The Visualization Toolkit

Open source toolkit for scientific visualization, computer graphics, and image processing

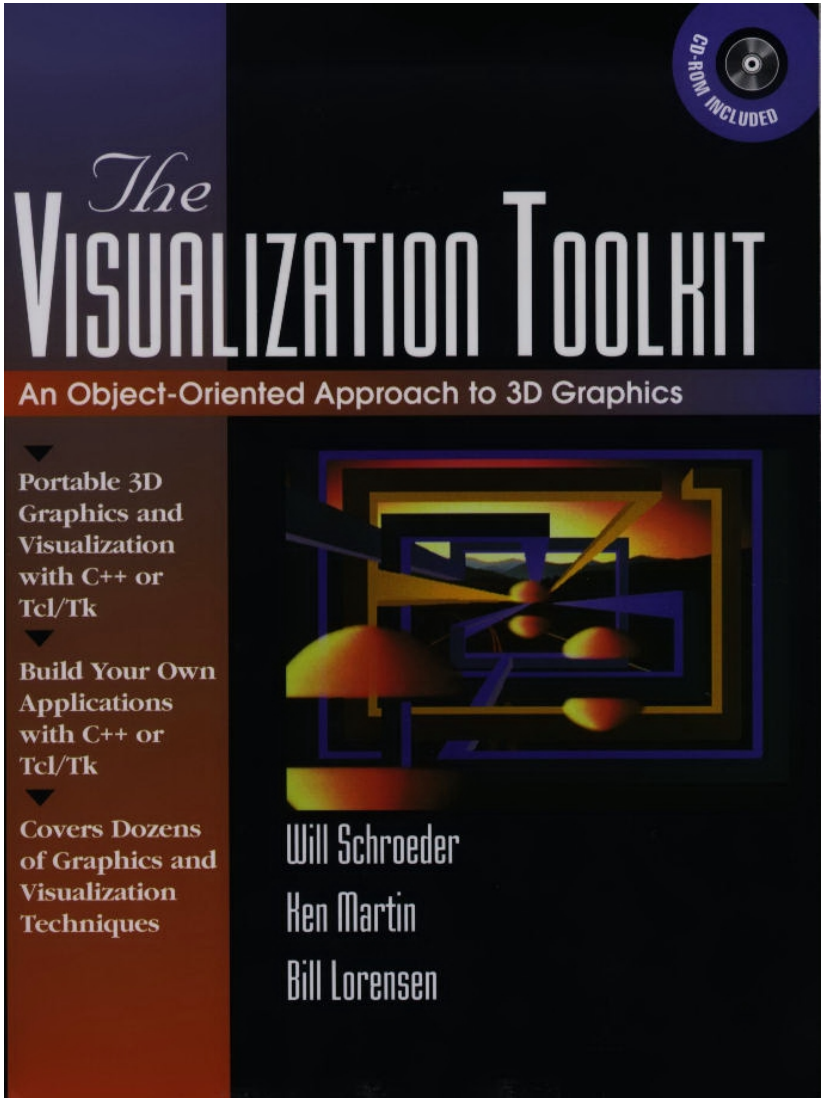




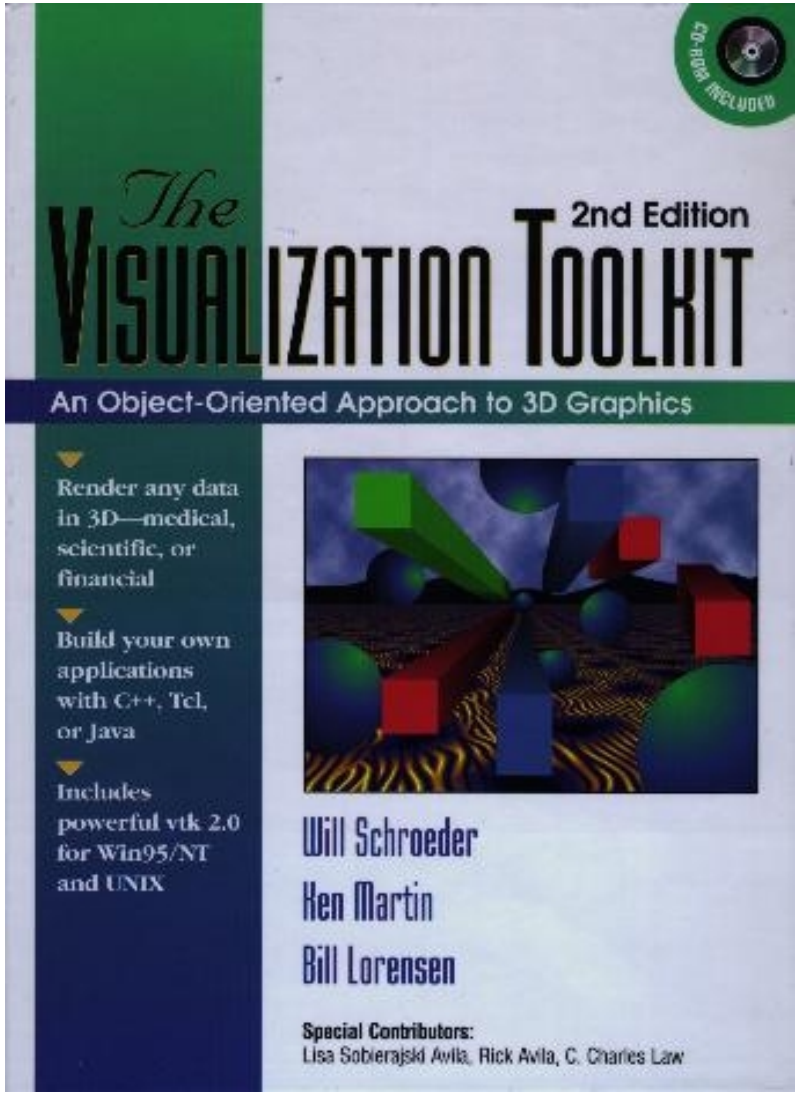




vtk1.0 1995



vtk2.0 1997



Visualization Toolkit

Open Source

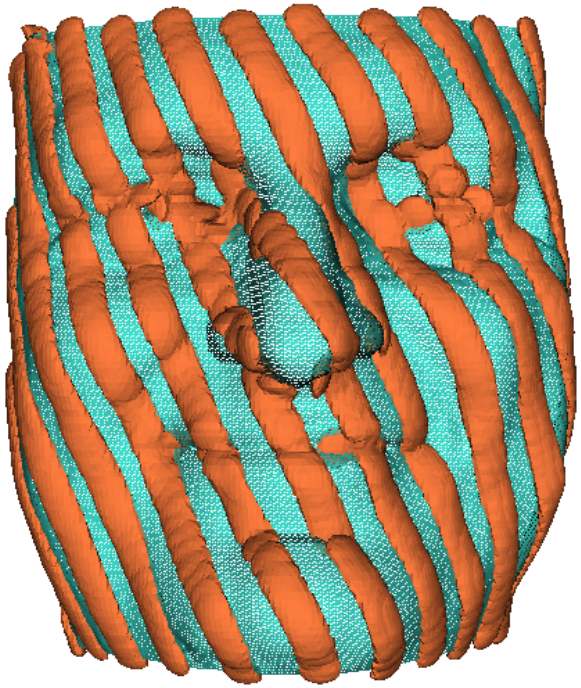
500 C++ classes

240,000 Lines of Code

- 100,000 executable

20+ developers

6 years of development



"We don't sell VTK,
we sell what we do with VTK."

Visualization Toolkit

Internal Customers

- GE Medical Systems
  - Platform software and applications, 1996
  - Volume visualization on the scanner, 1999
- GE Aircraft Engines
  - Engineering productivity tools, 1996-1998
- Lockheed Martin
  - Composite inspection system for JSF, 1998-2000
- DARPA
  - Virtual Endoscopy, 1996-1998
- FBI
  - Facial Reconstruction, 1999-2000
- Engineering Animation
  - Swept Surfaces, Path Planning 1998-2000

Visualization Toolkit

External "customers"

- Visual Interfaces Inc.
- Principia Mathematica Inc.
- Numerical Objects Inc.
- Brigham and Womens Hospital
- Rensselaer Polytechnic Institute
- NCSA
- Los Alamos National Lab, Argonne, Livermore

- 900 people on mailing list
- 12 Universities using VTK textbook

Testing Motivation

Nature of our business

- People working on projects for many different customers
- Leading edge algorithm and software development
- Development cycle does not fit "nicely" into a standard release schedule
  - Partly because VTK is not our "product"

Software is dynamic.

- Code worked when written, may not work now

Need "release on demand"

- Requires continuous software quality assurance

Testing Constraints

We only have 15 active developers, spread across many projects and sites

- Can't afford separate SQA division

We don't have dedicated testing hardware

- Testing cannot hinder project work

We are "algorithm developers" not "software engineers"

- Testing must be automated and concise

Testing Design Goals

Frequent testing

- Identify defects as soon as they are introduced
- Too hard to find cause if not done frequently

Minimally invasive to daily activities

Automated testing

Automated report generation/summaries

- Must be concise yet informative

Track testing results over time

Testing Terminology

Regression Test

White Box vs Black Box Test

Smoke Test

Regression Test

What

- Test code and data that reproduces an expected result
- Usually hand crafted (white box testing)

Why

- Assure that changes made to software do not introduce new defects

When

- Throughout the life cycle

Smoke Test

What

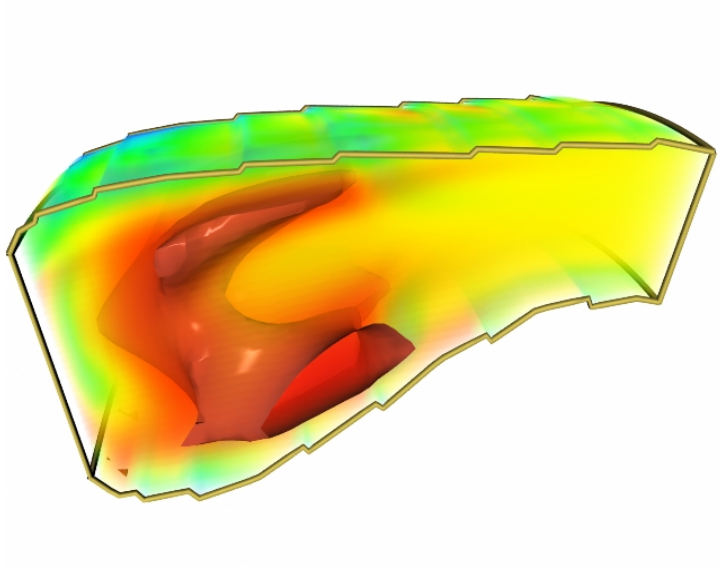
- A quick sanity check
- A single compile, link and run

Why

- Show minimum capability

When

- Anytime a change is made



White Box vs Black Box

White Box

- exercise procedural control points
- requires knowledge of individual classes/methods
- hand crafted
- logic driven

Black Box

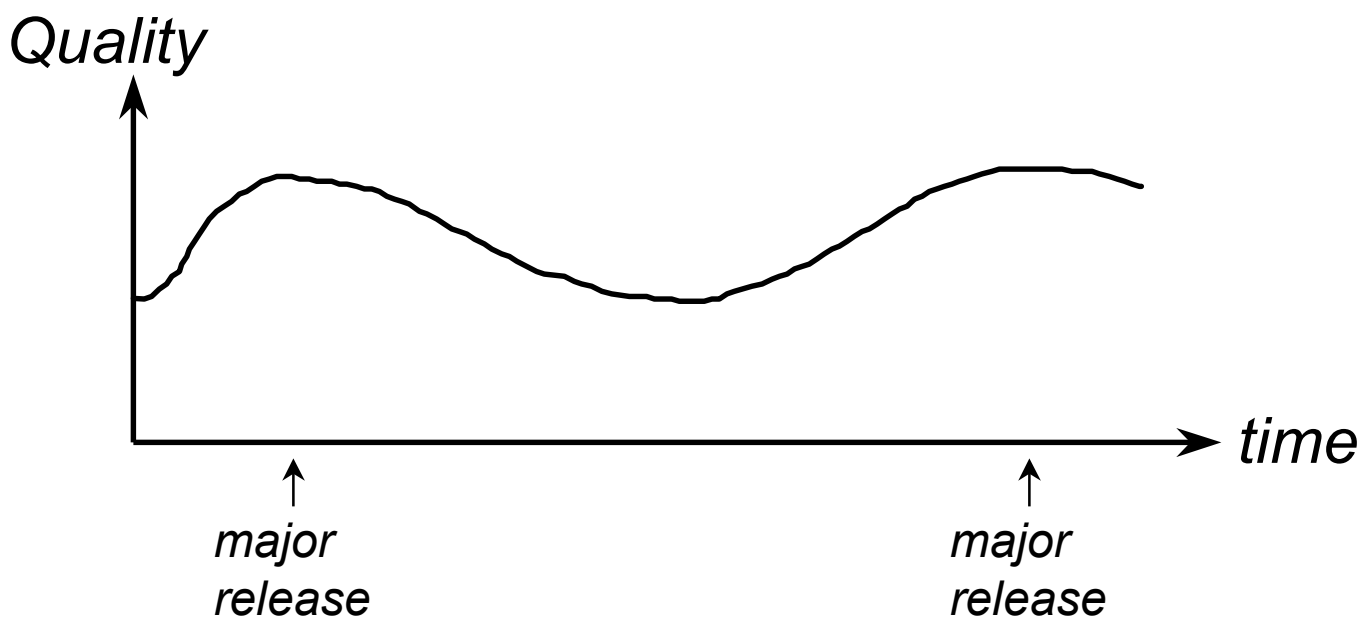
- automatically generated
- limited knowledge of class details
- test cases derived from specs
- data driven

Prior to 1998

100 regression tests

Performed manually and infrequently

- just before a major release (6 months)



Company-wide Quality Initiative, 1998 Brought 14 "Green Belt" Projects

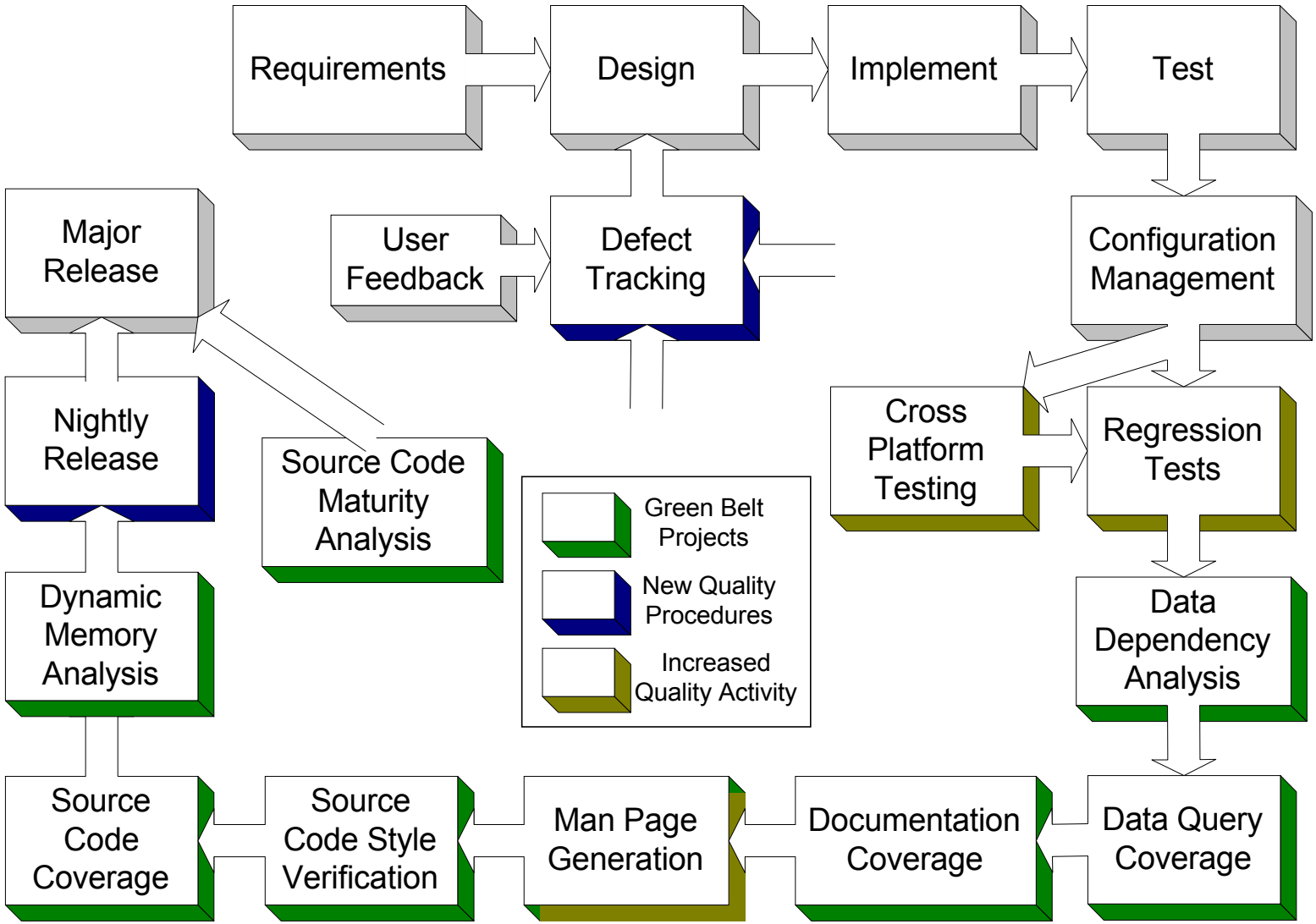
Measure, Analyze, Improve, Control (MAIC) applied to various aspects of the software

- Regression Tests
- Dynamic memory analysis
- Static code analysis
- Coverage analysis

Each of these projects yielded measurable (albeit temporary) improvement.

Next step was to apply these tests frequently and automatically

Visualization Toolkit (VTK) Software Process



Opportunities and Defects

A Day in the Life of vtk Quality

The day starts at 8pm (EST)

- ***Determine what has changed in the system***
- ***Update the testing system's version of the software***
- ***On 11 different system configurations, we***
  - ***Build the software***
  - ***Run over 500 regression tests***
- ***Dynamic memory analysis***
- ***Coverage analysis***
- ***Check on coding style and documentation***

8:01pm - Change Log

Summarize the changes that occurred since yesterday

This tightens the “cause and effect” cycle

There were 15 files changed since the last dashboard.

U common/vtkVersion.h
revision 1.521
date: 2000/05/24 20:08:58; author: millerjv; state: Exp; lines: +3 -3
Automated commit to force versioning.
==========
U common/vtkFileOutputWindow.cxx
revision 1.1
date: 2000/05/24 09:06:51; author: turek; state: Exp;
ENH: Captures debug/warning/error messages in a log file instead of on the console.
==========

8:05pm - Compile on all platforms

Summarize errors and warnings

Platform	Build VTK	
	Errors	Warnings
irix6	0	286
irix6n32	2	2419
solaris	0	16273

Keep complete log of errors and warnings
Provide link to logs

Link to abstracts of Warning/Error Logs

VTK Warning Summary for BuildLog

Warning (BuildLog line 373)

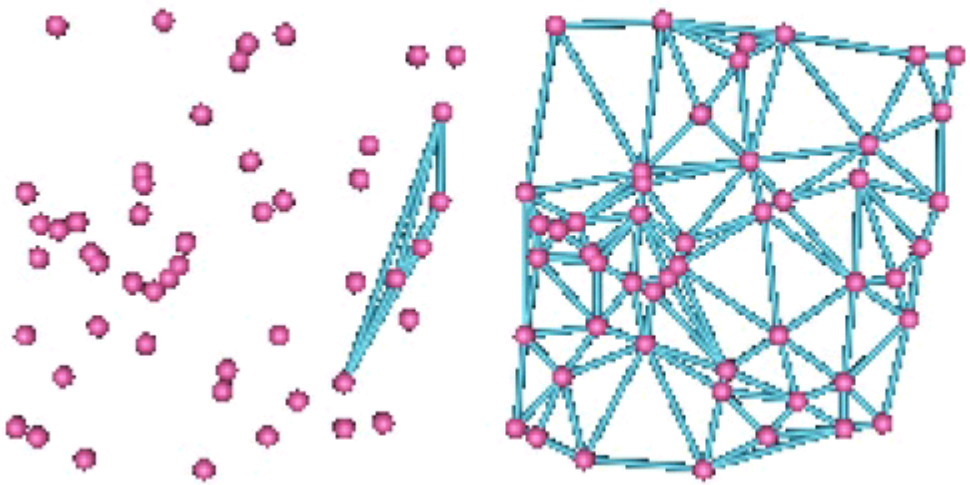
```
gmake[1]: Leaving directory `/tmp_mnt/projects/vtk/production/vtk-irix65/common'
cd common;     gmake -k all
gmake[1]: Entering directory `/tmp_mnt/projects/vtk/production/vtk-irix65/common'
"vtkCoordinate.cxx", line 436: warning(3665): variable "RefValue" is used before its value is set
      val[0] += RefValue[0];
               ^
line 54: warning(3482): class "vtkFileOutputWindow"  has no accessible constructors
  class VTK_EXPORT vtkFileOutputWindow : public vtkOutputWindow
```

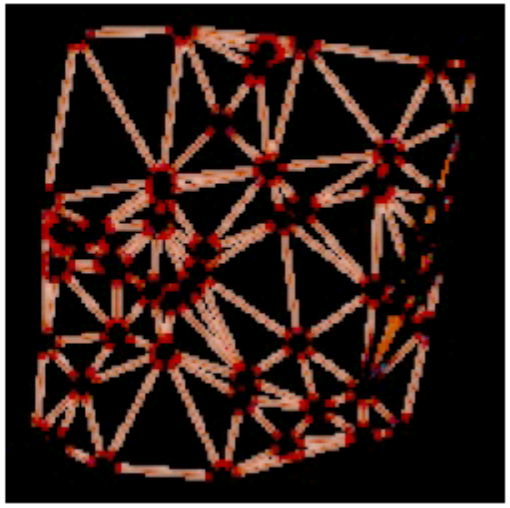
9:00pm - Image Regression Tests

Each regression test produces an image that is compared against a "gold standard image"

vtk regression summary for graphics: 171 passed and 15 failed

DelMesh.tcl took 0.312 seconds but failed with an error of 835.484:





Delaunay2D.tcl took 0.406 seconds but failed with an error of 28524.1 Delaunay2DAlpha.tcl took 0.469 seconds but failed with an error of 1314.08:

11:00pm - Other Regression Tests

Other tests produce text output that is compared against a baseline file

```
17,18c17,18
< World(0, 0, 0) -> Display(49, 49)
< World(0, 0, 0) -> LocalDisplay(49, 50)
---
> World(0, 0, 0) -> Display(50, 50)
> World(0, 0, 0) -> LocalDisplay(50, 49)
30c30
< Display(50, 50, 0) -> World(2.70654e05, 2.70654e-05, 0.99)
---
> Display(50, 50, 0) -> World(0, 0, 0.99)
44c44
< Normalized Display(0.5, 0.5, 0) -> World(2.70654e05, 2.70654e-05, 0.99)
---
> Normalized Display(0.5, 0.5, 0) -> World(0, 0, 0.99)
```

Regression Tests

Compare images

Quantify defects

Produce summary

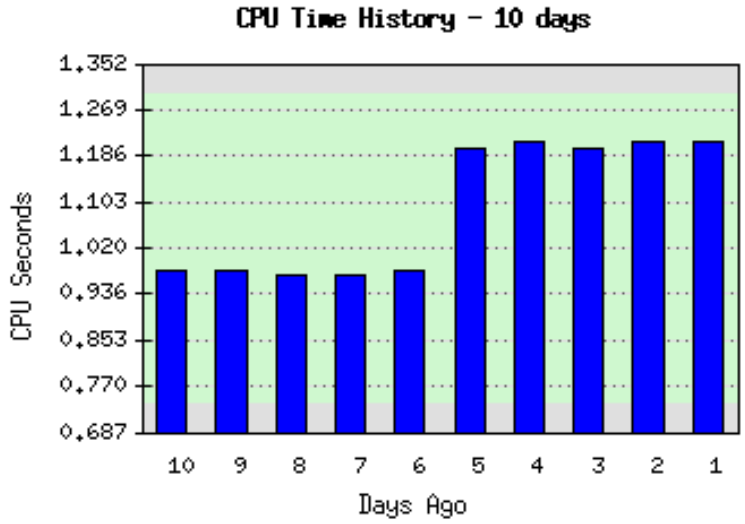
Platform	Build VTK		Test Tcl				Test Cxx	
	Errors	Warnings	Passed	Failed	Faster	Slower	Passed	Failed
irix6	0	286	355	0	7	4	44	2
irix6n32	2	2419	355	0	1	0	46	0
solaris	0	16273	341	2	3	1	41	0
solarisCoverage	0	10	353	2	0	8	45	0
WinNT	Catastrophic failure.		Catastrophic failure.					
hp	0	4955	0	396	0	0	0	630
linux	0	10	Catastrophic failure.					
solaris26	0	0	310	9	0	0		

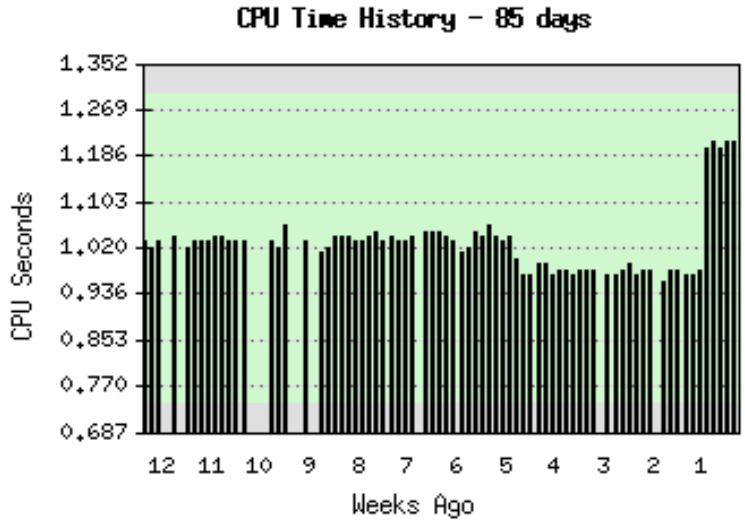
Regression Tests

Record CPU time and compare to history

"Defect" is test that takes "statistically" more time than usual

VisQuad.tcl - 1.3794 wall, 1.2100 cpu, 0.5398 imgdiff, Passed Warning: Recent Time Increase:





1:00am - Coverage Analysis

Identify which lines of code are tested

- *"If it isn't tested, it doesn't work."*

Summarize coverage across VTK, across each kit, and across individual files.

Coverage results: 72.70%, tested: 52088, untested: 19562, total: 71650

vtkActor2D.cxx	78.12%
vtkActor2DCollection.cxx	84.21%
vtkAttributeData.cxx	88.51%
vtkBitArray.cxx	69.23%
vtkByteSwap.cxx	57.14%
vtkCell.cxx	80.87%
vtkCellArray.cxx	81.25%
vtkCellData.cxx	5.26%
vtkCellLinks.cxx	64.18%
vtkCellTypes.cxx	90.48%
vtkCharArray.cxx	58.41%
vtkCollection.cxx	80.00%
vtkContourValues.cxx	91.49%
vtkCoordinate.cxx	36.27%

Defect: Coverage below spec limit

1:00am - Style

Check coding conventions

Detect code that may cause a future defect

- Someone other than the original author may be modifying the code

Summarize overall style and style per file

Style test results: 53 defective files out of 972 files, 757 defects out of 104876 opportunities.

Style Rule	Opportunities	Defects
Class name prefix "vtk"	507	0
Class name alphanumerics	507	0
Instance variable begins with uppercase	2222	15
Protected member data	2222	153
Member function begins with uppercase	8590	0
Dereferencing member data	33641	249
Dereferencing member function	24656	85
Braces around "if", "else", "for", ...	20721	188

1:00am - Memory Analysis

Check for memory leaks, array bound writes, uninitialized memory reads, etc.

Purify Total Critical Defects: 0. (Purify last run on Jan 6 06:11)

```
VTK Dynamic Code Analysis for Thursday January 07, 1999 at 07:35 AM
==================================
Total Memory Leaked: 0 bytes
Number of Leaks    : 0
Number of ABRs     : 0
Number of ABWs     : 0
Number of CORs     : 0
Number of FMMs     : 0
Number of FUMs     : 0
Number of FMRs     : 0
Number of FMWs     : 0
Number of MAFs     : 0
Number of UMCs     : 213
Number of UMRs     : 144

Purify Total Critical Defects: 0
```

7:00 am - Tests complete

Construct top level Web-dashboard

Package and export testing

Package and export the software for world wide distribution

- Zipped source
- Linux rpm's
- Windows install

VTK Dashboard for Wed May 24 20:03:30 EDT 2000

There were 15 files changed since the last dashboard. Changes for the month.

Platform	Build VTK		Tcl Image Tests				Cxx Image Tests		Other Tcl Tests		Other Cxx Tests	
	Errors	Warnings	Pass	Fail	Fast	Slow	Pass	Fail	Pass	Fail	Pass	Fail
irix65	0	18	478	5	3	26	51	0	6	1	7	1
irix6n32	0	16	480	3	0	3	51	0	6	1	7	1
irix6x64	0	18	480	3	0	2	51	0	6	1	7	1
solaris	1	20	446	7	0	7	45	0	6	1	7	1
solarisCoverage	1	3	467	10	0	9	50	0	6	1	7	1
WinNT	0	3	473	10	0	0						
hp	2	0	475	8	0	3	49	2	6	1	7	1
linuxRH52	0	5	476	5	0	3	51	0	6	1	7	1
solaris26m6	0	7	478	5	0	13	51	0	6	1	7	1
solaris26	0	0	436	33	0	0						

7:00 am - A Good Day

A Good Day

There were 32 files changed since the last dashboard. Changes for the month.

Platform	Build VTK		Test Tcl				Test Cxx	
	Errors	Warnings	Passed	Failed	Faster	Slower	Passed	Failed
irix65	0	22	416	0	3	5	49	0
irix6n32	0	5210	416	0	0	0	49	0
irix6x64	0	445	416	0	0	0	49	0
solaris	6	19815	394	2	0	13	44	0
solaris27	0	4064	416	0	8	3	49	0
solarisCoverage	0	679	412	2	0	5	49	0
WinNT	0	2	414	2	0	0		
hp	0	11729	413	3	1	2	48	1
linuxRH52	0	298	415	1	0	4	49	0
solaris26m6	0	4070	408	8	0	3	47	2
solaris26	0	0	402	1	0	0		

Key:	Internal Test System	External Test System	System or Network Error	Build/Test Successful	Build/Test Error	Timing Change

Real work begins at 7:05am

7:00am - A Bad Day

Platform	Build VTK		Test Tcl				Test Cxx	
	Errors	Warnings	Passed	Failed	Faster	Slower	Passed	Failed
irix6	0	286	355	0	7	4	44	2
irix6n32	2	2419	355	0	1	0	46	0
solaris	0	16273	341	2	3	1	41	0
solarisCoverage	0	10	353	2	0	8	45	0
WinNT	Catastrophic failure.		Catastrophic failure.					
hp	0	4955	0	396	0	0	0	630
linux	0	10	Catastrophic failure.					
solaris26	0	0	310	9	0	0		

Key:	Internal Test System	External Test System	System or Network Error	Build/Test Successful	Build/Test Error	Timing Change

We are “prisoners of quality”...

In search of a new software engineering process


COMPUTER
Innovative Technology for Computer Professionals
October 1999
Stumbling toward a Privatized DNS
Software Engineering in the Small
MEMS Move the World
The Wireless Web
Computer Society
http://computer.org
Code of Ethics for Software Engineers, p. 84

Cover Feature

Embracing Change with Extreme Programming

Extreme Programming turns the conventional software process sideways. Rather than planning, analyzing, and designing for the far-flung future, XP programmers do all of these activities—a little at a time—throughout development.

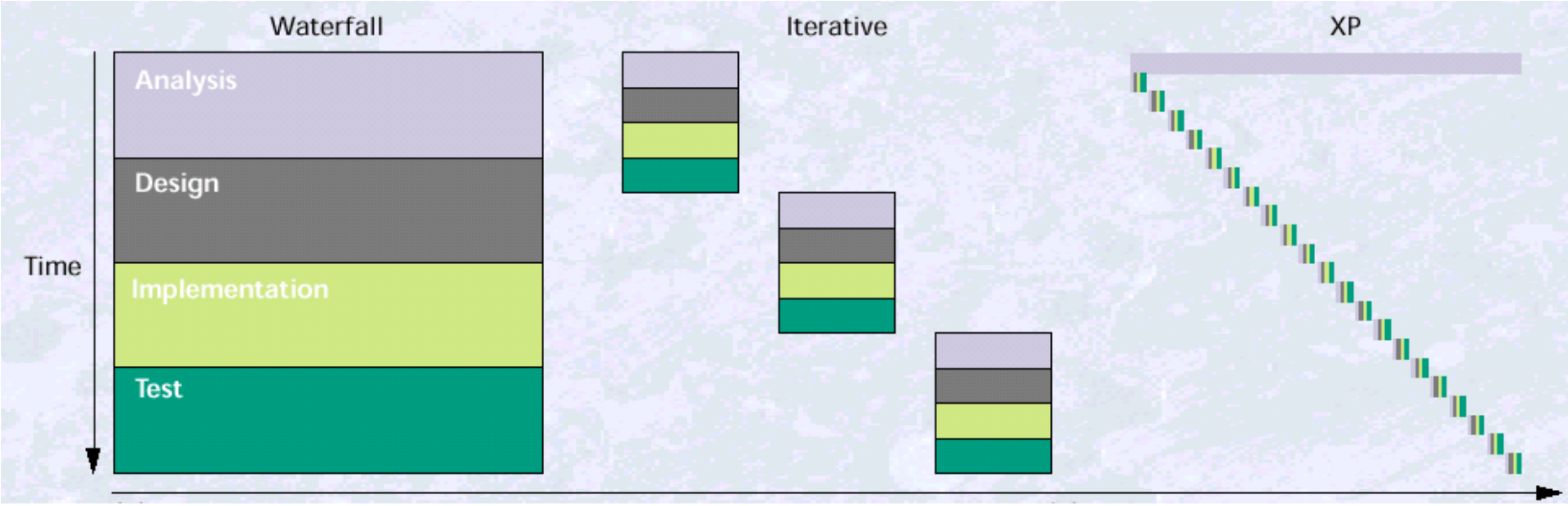
Kent Beck
First Class Software

In the beginning was the waterfall (Figure 1a). We would get the users to tell us once and for all exactly what they wanted. We would design the system that would deliver those features. We would code it. We would test to make sure the features were delivered. All would be well.

shows, the waterfall begat iterations.

The waterfall model didn't just appear. It was a rational reaction to the shocking measurement that the cost of changing a piece of software rose dramatically over time. If that's true, then you want to make the biggest, most far-reaching decisions as early in the

Extreme Programming



Extreme Testing

Short software engineering life cycle

- Design, implement, test

The Software should ALWAYS work

Find and fix defects in hours not weeks

- Bring SQA inside the development cycle
- Break the cycle of letting users find bugs

Automate everything

All developers are responsible for testing

The Importance of Early Testing

Testing early and often is critical to high quality software

Retain measurements to assess progress and measure productivity

Present results in concise, informative ways

Know and Show the status of the system at any time

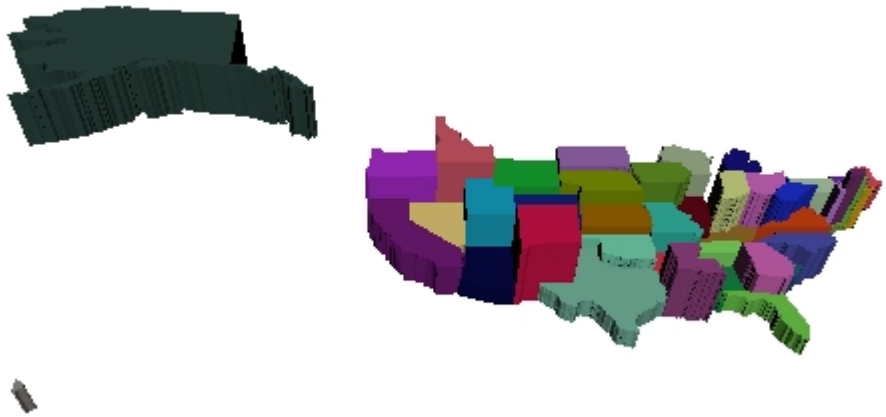
Our customers expect it to be the way we work

http://tinderbox.mozilla.org/

Build Time	Guilty	Mac Debug Clbr	Mac Opt Clbr	Win32 VC6.0Clbr	Win32 VC6.0Dep	harpoon Linux Dep	shrike Linux Clbr	speedracer SunOS/sparc 5.6 Dep
Click time to see changes since then	Click name to see what they did							
05/25 13:18		L C	L C	L C	L C	L C Lk:4K Bl:23M	L C	L C
13:17				L C				
13:09	mj				L C			
05/25 12:58	blizzard kin waterson					L C Lk:4K Bl:23M		L C
12:54					L C			
12:49								
12:38	jgaunt leaf					L C Lk:4K Bl:23M		L C
12:37					L			
12:36								
12:34								
12:27						L Lk:4K Bl:22M		
12:22	buster							L
12:12						L Lk:4K Bl:23M		
12:09				L	L			
12:08						L Lk:4K Bl:23M		
12:05								
05/25 11:51							L C	L

Why a Continuous Build?

- The nightly test is critical to assessing the state of the system
- A single compilation or load error can jeopardize an entire night of testing



Extreme Testing

VTK Continuous Build Dashboard

Build	Time	Files changed	Build	Test	Who
1184	Thu May 25 15:43:40 EDT 2000		Waiting		
1183	Thu May 25 15:34:43 EDT 2000	4	0 errors	4 passed	martink
1182	Thu May 25 09:53:29 EDT 2000	1	0 errors	4 passed	martink
1181	Thu May 25 08:13:12 EDT 2000	2	0 errors	4 passed	turek
1180	Wed May 24 22:17:59 EDT 2000	1	0 errors	4 passed	millerjv
1179	Wed May 24 17:37:48 EDT 2000	1	0 errors	4 passed	martink
1178	Wed May 24 17:22:48 EDT 2000	6	0 errors	4 passed	martink
1177	Wed May 24 11:18:05 EDT 2000	3	0 errors	4 passed	will
1176	Wed May 24 09:08:34 EDT 2000	3	0 errors	4 passed	turek
1175	Wed May 24 08:26:28 EDT 2000	1	0 errors	4 passed	lorensen

Someone Broke The Build!

1175	Wed May 24 08:26:28 EDT 2000	1	0 errors	4 passed	lorensen
1174	Tue May 23 22:36:24 EDT 2000	1	0 errors	4 passed	millerjv
1173	Tue May 23 11:56:57 EDT 2000	1	0 errors	4 passed	martink
1172	Tue May 23 08:54:18 EDT 2000	1	0 errors	4 passed	will
1171	Mon May 22 16:53:13 EDT 2000	1	0 errors	4 passed	dgobbi
1170	Mon May 22 07:45:16 EDT 2000	1	0 errors	4 passed	lawcc
1169	Sun May 21 22:06:31 EDT 2000	5	0 errors	Java failed	lawcc will
1168	Sun May 21 21:47:21 EDT 2000	1	0 errors	4 passed	millerjv
1167	Sun May 21 19:07:20 EDT 2000	2	0 errors	4 passed	will
1166	Sun May 21 17:46:46 EDT 2000	1	0 errors	4 passed	biddi
1165	Sun May 21 15:57:12 EDT 2000	2	0 errors	4 passed	dgobbi
1164	Sun May 21 14:07:46 EDT 2000	4	0 errors	4 passed	dgobbi

How do we use the system?

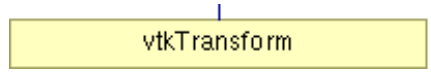
- **Track the effects of major changes**
- Identify what needs to be changed
- Portability leads to quality
- Navigate software features
- Build and test on future OS releases
- Test new 3rd party software (e.g. OpenGL) for compliance

Has truly "Changed the way we work."

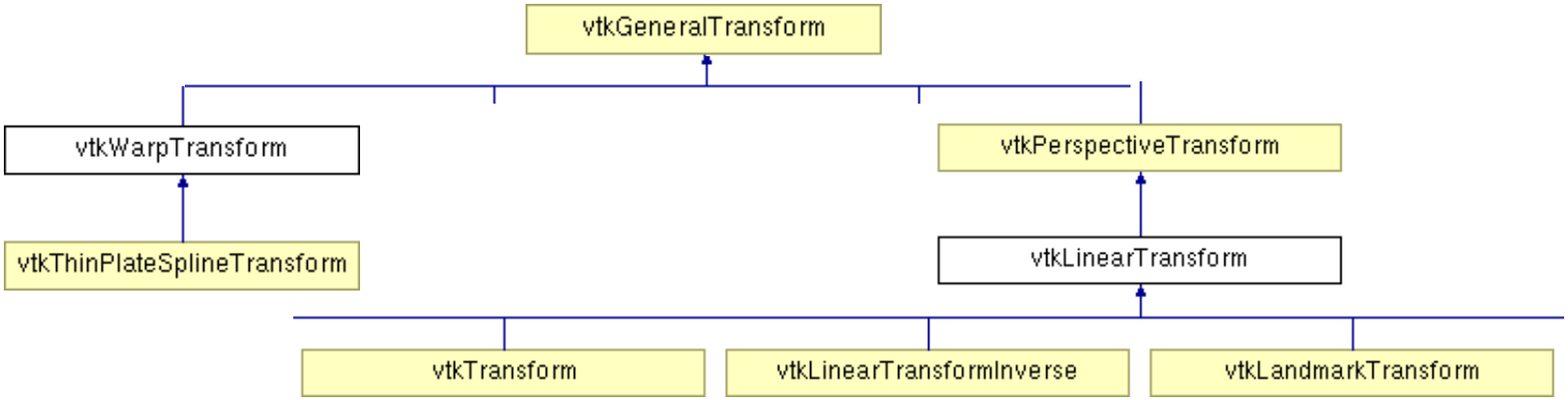
Case Study

New developer to our Open Source community proposed a new class hierarchy for vtk Transformations

Old



New



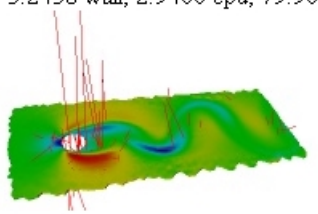
The first checkin

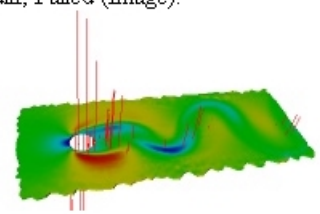
There were 42 files changed since the last dashboard. Changes for the month.

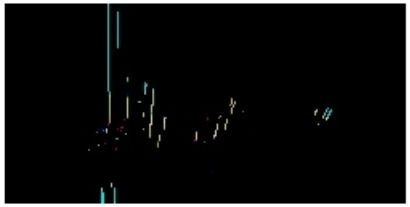
Platform	Build VTK		Tcl Image Tests				Cxx Image Tests		Other Tcl Tests		Other Cxx Tests	
	Errors	Warnings	Pass	Fail	Fast	Slow	Pass	Fail	Pass	Fail	Pass	Fail
irix65	0	1	Catastrophic failure.									
irix6n32	0	0	457	2	3	2	49	0	5	2	8	0
irix6x64	0	1	457	2	2	2	49	0	5	2	8	0
solaris	0	11	425	4	1	16	43	0	5	2	8	0
solarisCoverage	0	2	373	6	1	10	48	0	5	2	8	0
WinNT	Catastrophic failure.		0	0	0	0						
hp	2	0	454	5	1	2	48	1	5	2	8	0
linuxRH52	0	0	452	5	2	3	49	0	5	2	7	1
solaris26m6	0	0	450	9	4	7	48	1	5	2	8	0
solaris26	0	0	440	19	0	0						

vtk regression summary for contrib: 22 passed and 1 failed, with 0 slower, 2 faster, and 0 new tests.

cellDerivs.tcl - 5.2458 wall, 2.9400 cpu, 79.9647 imgdiff, Failed (Image):







The day after

There were 8 files changed since the last dashboard. Changes for the month.

Platform	Build VTK		Tcl Image Tests				Cxx Image Tests		Other Tcl Tests		Other Cxx Tests	
	Errors	Warnings	Pass	Fail	Fast	Slow	Pass	Fail	Pass	Fail	Pass	Fail
irix65	0	0	460	0	2	13	49	0	5	2	8	0
irix6n32	0	0	460	0	3	1	49	0	5	2	8	0
irix6x64	0	0	460	0	3	6	48	1	5	2	8	0
solaris	0	2	428	2	1	17	43	0	5	2	8	0
solarisCoverage	0	0	449	5	0	10	48	0	5	2	8	0
WinNT	Catastrophic failure.		Catastrophic failure.									
hp	2	0	457	3	2	1	48	1	5	2	8	0
linuxRH52	0	0	457	3	2	9	49	0	5	2	7	1
solaris26m6	0	0	Catastrophic failure.									
solaris26	0	0	440	19	0	0						

U graphics/vtkTransformPolyDataFilter.cxx

revision 1.13

date: 2000/03/03 15:36:48; author: lorensen; state: Exp; lines: +4 -4

ERR: Logic was confused for transforming Cell Normals and Cell Vectors.

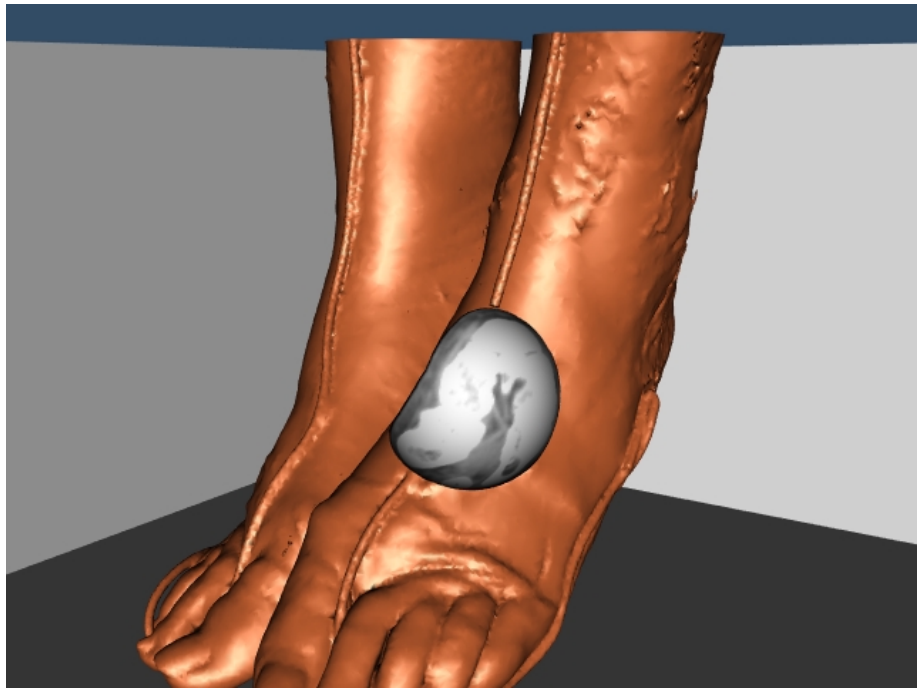
How do we use the system?

- Track the effects of major changes
- Identify what needs to be changed
- Portability leads to quality
- Navigate software features
- Build and test on future OS releases
- Test new 3rd party software (e.g. OpenGL) for compliance

Has truly "Changed the way we work."

Lessons Learned

- Frequent testing
- Automation
- Summarization
- Navigation
- Cross platform



What's next?

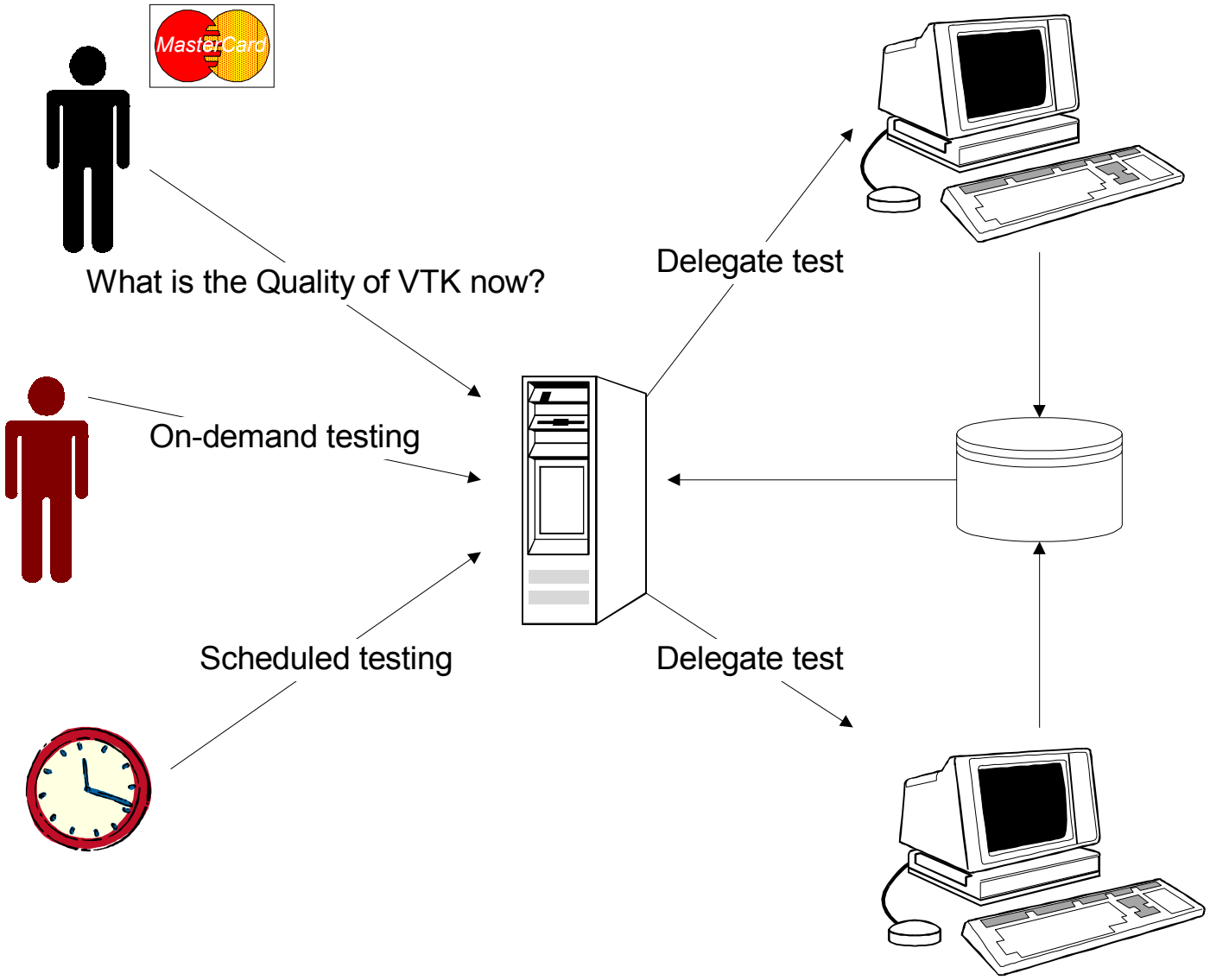
Current system

- uses: csh, tcsh, tclsh, awk, grep, make, gmake, cvs, rcp, ssh, scp, ...
- is "vtk-centric"
- monolithic control structure best suited for daily testing of the complete system
- dashboard summarizes state as of 6 am

New system

- Client-server model
- On demand dashboards summarizing the most recent results

Testing Architecture



Alternate Presentation of Test Results

Platform	Build VTK		Tcl Image Tests		Cxx Image Tests		Other Tcl Tests		Other Cxx Tests	
	Errors	Warnings	Pass	Fail	Pass	Fail	Pass	Fail	Pass	Fail
hp	1(2)	0	470	4						
irix65	0	1	473	1	51	1	6(7)	1(0)	8(7)	0
irix6n32	0	1	473	1						
irix6x64	0	1	472	2	50	2	6(7)	1(0)	8(7)	0
linuxRH52	0	0	469(470)	4(3)	51	2	5(7)	2(0)	8(7)	0
solaris	0	2	440	4	44	2	5(7)	2(0)	8(7)	0
solaris26	0	0								
solaris26m6	0	0	465(466)	9(8)	50	2	5(7)	2(0)	8(7)	0
solarisCoverage	0	2(0)	460(471)	4(3)						
WinNT	0	0	26(80)	1(4)						

Summary

Automated, nightly build and test on mulitple hardware and OS configurations

A daily dashboard presents the current state of the system

All users and customers can see the state of the system

All developers can run tests

Developers are expected to repair defects before the next nightly build and test

vtk Dashboards on the Web

www.visualizationtoolkit.org/vtk/quality/

VTK Dashboard for Wed May 24 20:03:30 EDT 2000

There were 15 files changed since the last dashboard. Changes for the month.

Platform	Build VTK		Tcl Image Tests				Cxx Image Tests		Other Tcl Tests		Other Cxx Tests	
	Errors	Warnings	Pass	Fail	Fast	Slow	Pass	Fail	Pass	Fail	Pass	Fail
irix65	0	18	478	5	3	26	51	0	6	1	7	1
irix6n32	0	16	480	3	0	3	51	0	6	1	7	1
irix6x64	0	18	480	3	0	2	51	0	6	1	7	1
solaris	1	20	446	7	0	7	45	0	6	1	7	1
solarisCoverage	1	3	467	10	0	9	50	0	6	1	7	1
WinNT	0	3	473	10	0	0						
hp	2	0	475	8	0	3	49	2	6	1	7	1
linuxRH52	0	5	476	5	0	3	51	0	6	1	7	1
solaris26m6	0	7	478	5	0	13	51	0	6	1	7	1
solaris26	0	0	436	33	0	0						

MostRecentResults/Dashboard.html

VTK Continuous Build Dashboard

Build	Time	Files changed	Build	Test	Who
1184	Thu May 25 15:43:40 EDT 2000		Waiting		
1183	Thu May 25 15:34:43 EDT 2000	4	0 errors	4 passed	martink
1182	Thu May 25 09:53:29 EDT 2000	1	0 errors	4 passed	martink
1181	Thu May 25 08:13:12 EDT 2000	2	0 errors	4 passed	turek
1180	Wed May 24 22:17:59 EDT 2000	1	0 errors	4 passed	millerjv
1179	Wed May 24 17:37:48 EDT 2000	1	0 errors	4 passed	martink
1178	Wed May 24 17:22:48 EDT 2000	6	0 errors	4 passed	martink
1177	Wed May 24 11:18:05 EDT 2000	3	0 errors	4 passed	will
1176	Wed May 24 09:08:34 EDT 2000	3	0 errors	4 passed	turek
1175	Wed May 24 08:26:28 EDT 2000	1	0 errors	4 passed	lorensen

ContinuousResults/solaris/ContinuousResults.html

Visualization Toolkit Extreme Testing

A Production Release Every Day

Bill Lorensen
Jim Miller

GE Corporate Research and Development
Niskayuna, NY

lorensen@crd.ge.com
millerjv@crd.ge.com