



Tcl/Tk

민 인학 (inhak.min@gmail.com)

한국 Tcl/Tk 커뮤니티

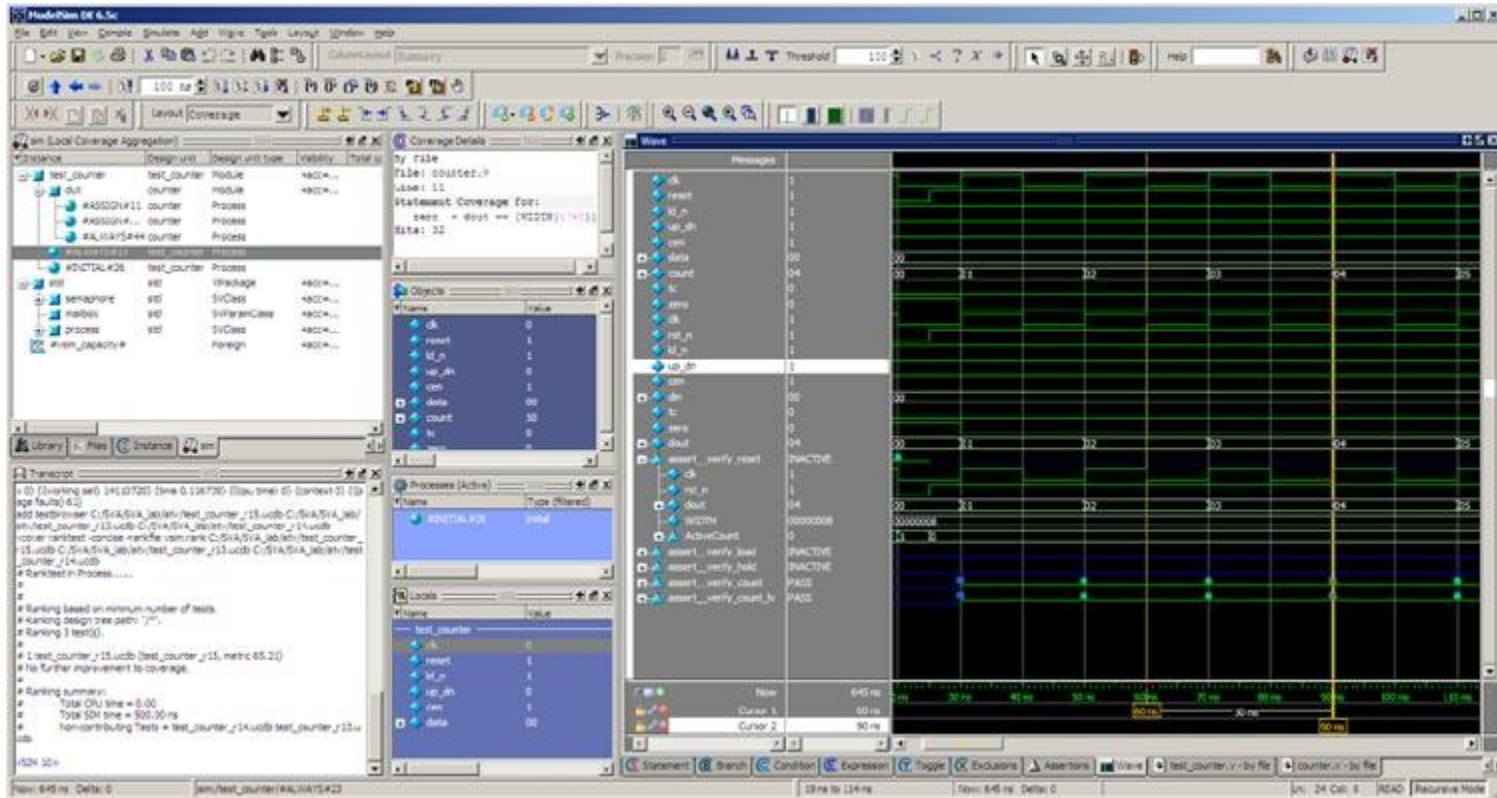
Tcl 역사

- 1988년도에 존 오스터하우트(John Ousterhout) 에 의해 개발시작
- 1991년 Tcl 첫번째 릴리즈
- 1992년 Tk 첫번째 릴리즈
- 1994~1998년 선(Sun) 에서 유지보수
 - 이 시기에 맥 오에스/윈도우즈 지원시작
 - 국제화지원/실행속도개선/
UTF-8채용/고성능의 정규 표현식 지원...
- 2000년10월부터 오픈소스로 전환

Tcl/Tk 특징

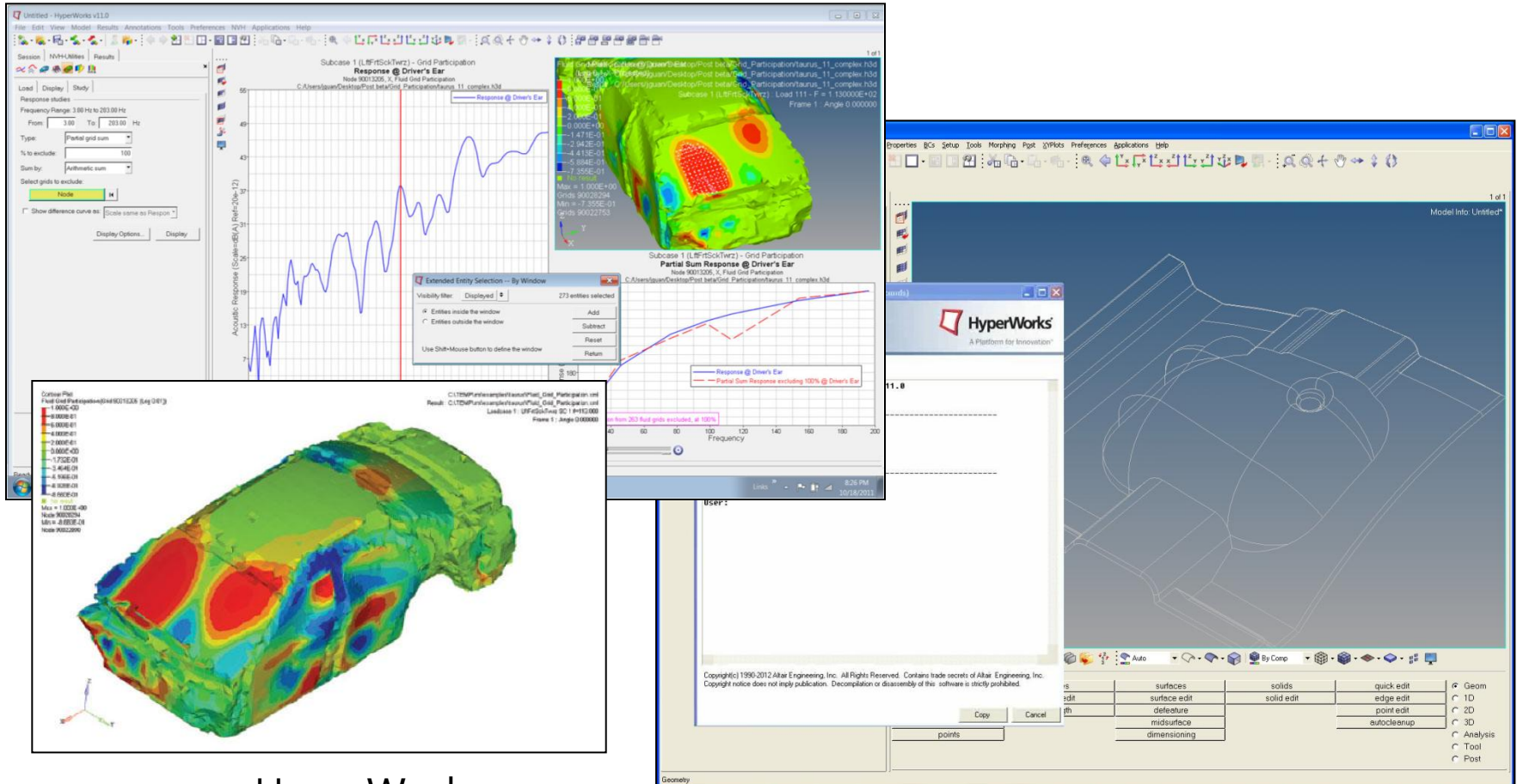
- high-level scripting language
 - 모티프나 윈도우즈 코드보다 작다.
- interpreted
 - 직접 실행이 가능하며, 컴파일링/링킹 과정이 필요없다.
- extensible
 - Tcl이나 C로 확장가능하다.
- embeddable
 - C에서 Tcl 터프리터를 호출할 수 있다.
- most platforms
 - 유닉스/리눅스/맥/윈도우즈 지원.
- Autoloading
 - 자동 라이브러리 로딩.
- free
 - 오픈소스.
 - 로열티없음.

Tcl/Tk Application



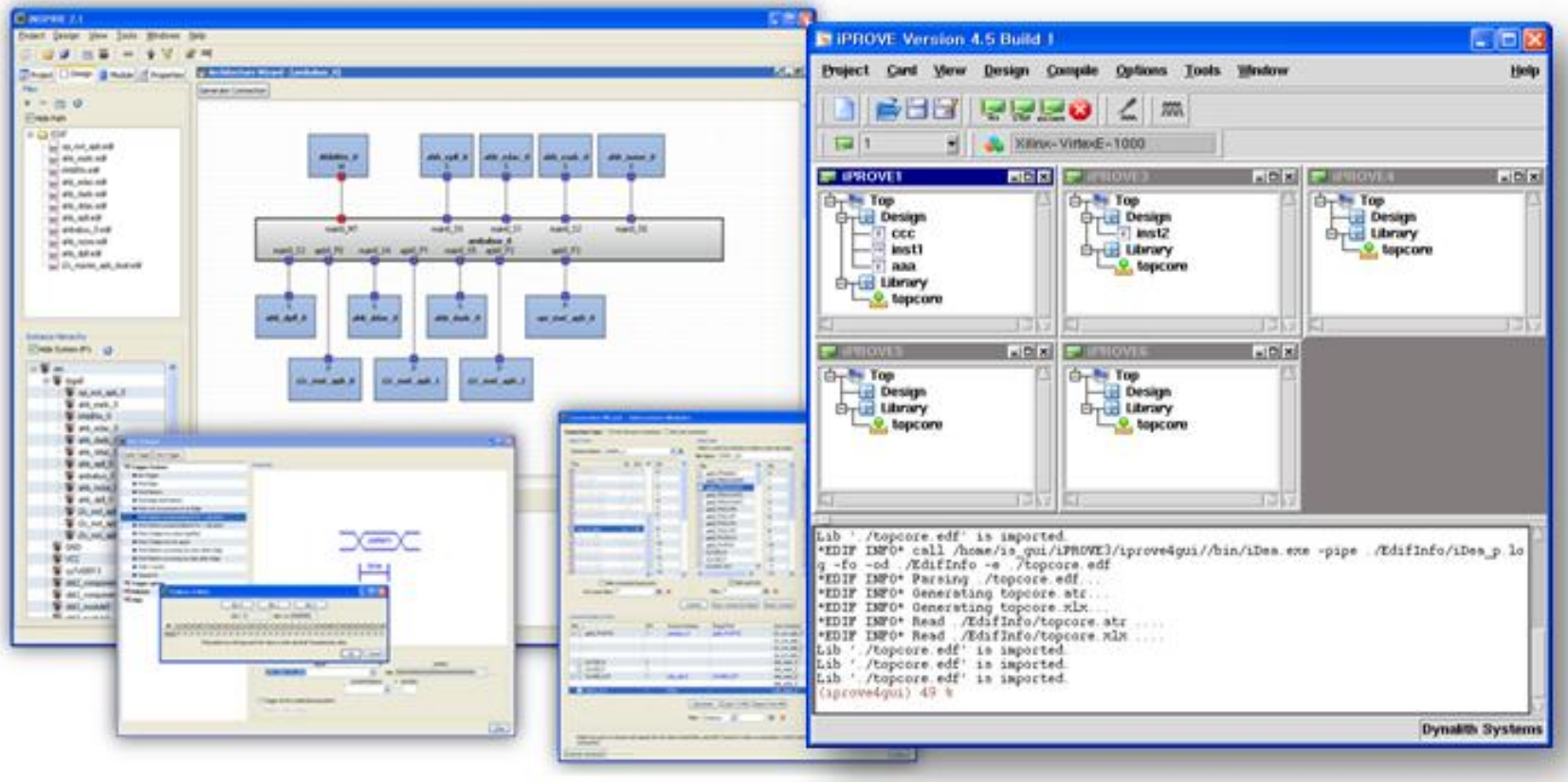
- Modelsim
 - Tcl/Tk 8.4 버전 사용됨

Tcl/Tk Application



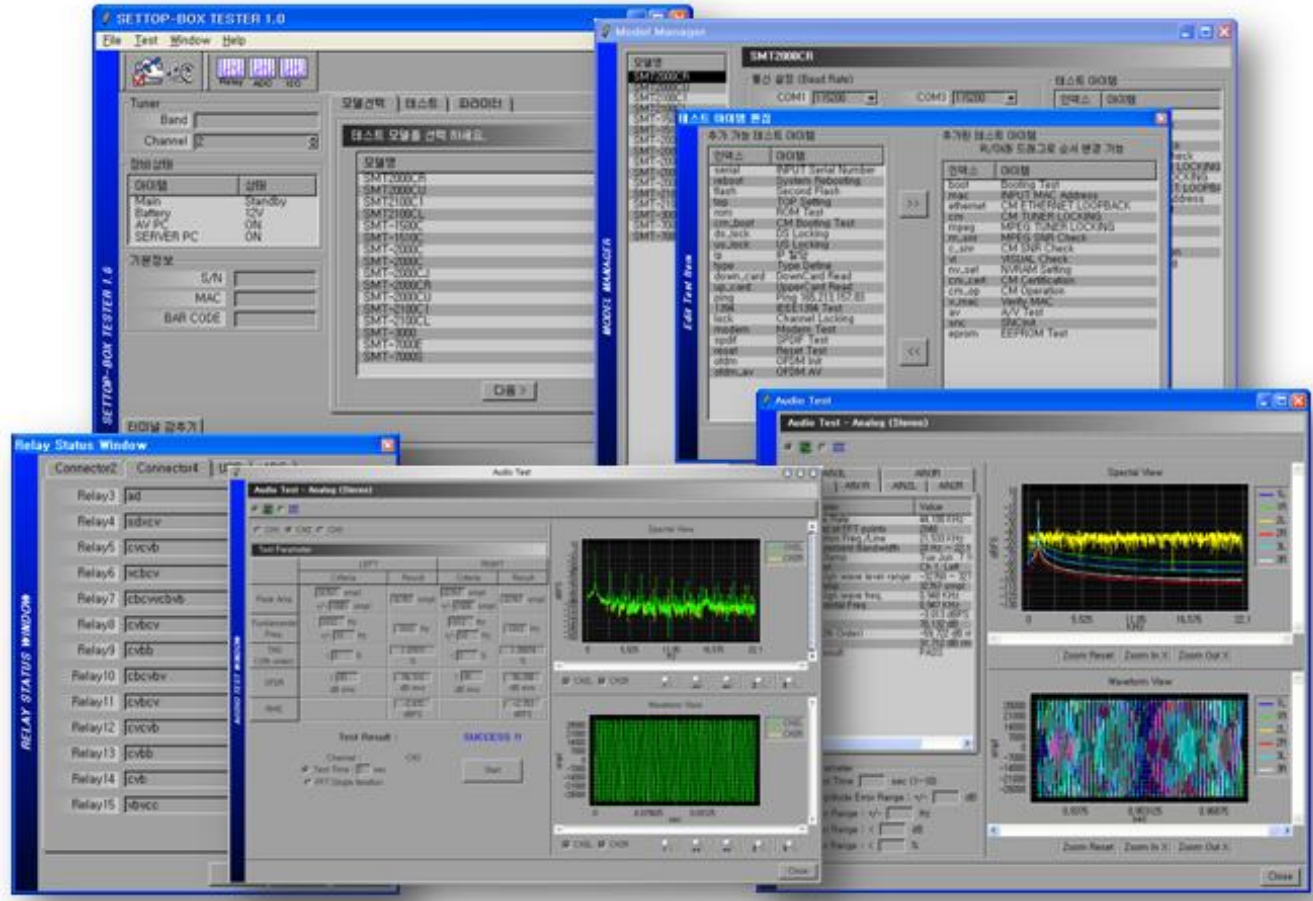
- HyperWorks
 - World Best CAE Total Solution

Tcl/Tk Application



- INSPIRE, iPROVE
 - Tcl/Tk 8.4 버전 사용됨

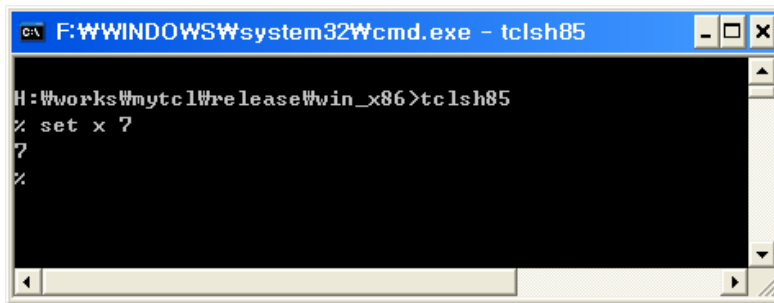
Tcl/Tk Application



- STB Tester
 - Tcl/Tk 8.4 버전 사용됨

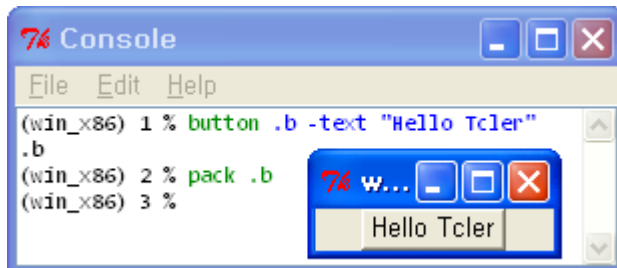
Tcl/Tk 사용방식

- tclsh 콘솔 어플



```
cmd F:\WINDOWS\system32\cmd.exe - tclsh85
H:\works\mytcl\release\win_x86>tclsh85
% set x 7
7
%
```

- wish 윈도우 어플



```
7% Console
File Edit Help
(win_x86) 1 % button .b -text "Hello Tcler"
.b
(win_x86) 2 % pack .b
(win_x86) 3 %
```

The screenshot shows a console window with a menu bar (File, Edit, Help) and a list of commands. The output shows a button widget being created and packed. A small window titled '7% w...' is overlaid on the console, displaying a button with the text 'Hello Tcler'.

Tcl/Tk 사용방식

- C/C++ 언어의 embedded 어플

```
#include <tcl.h>
```

```
int main(int argc, char *argv[]) {
```

```
    Tcl_interp *interp = Tcl_CreateInterp();
```

```
    code = Tcl_EvalFile(interp, argv[1]);
```

```
    if (*interp->result != 0) {
```

```
        printf("%s\n", interp->result);
```

```
    }
```

```
}
```

Tcl/Tk 사용방식

- 전통적인 유닉스 스크립트

```
#!/usr/local/bin/wish -f  
puts stdout "Hello, world!"
```

기본 문법

- 명령어(커맨드)의 인자를 스페이스로 구분지어 나열
 - `command arg1 arg2 arg3 ...`
- 한 행에 여러 개의 명령어를 나열할 땐 ";"으로 구분
 - `command arg1 arg2 arg3 ... ; command arg1 arg2 arg3 ...`

기본 문법

- 한 행이 길어진다면 백(역) 슬래쉬 “\”로 이어 쓴다.
 - `command arg1 \`
`arg2 arg3 ...`

Hello World

- `puts stdout {Hello World}`
 - Hello World
 - `puts stderr {Hello World}`
 - Hello World
 - `puts {Hello World}`
 - Hello World
 - `puts stdout "Hello World"`
 - Hello World
 - `puts stderr "Hello World"`
 - Hello World
 - `puts "Hello World"`
 - Hello World
- ✓ `puts` 커맨드로 출력한다.
 - ✓ 스페이스(공백) 문자를 포함하고 싶다면 "... "나 {...} 사용
 - ✓ `stdout`은 표준 메시지 출력
 - ✓ Standard output
 - ✓ `stderr`은 표준 에러 메시지 출력
 - ✓ Standard error

변수

- `set var 123`
 - `123`
- `set var`
 - `123`
- `set var abc`
 - `abc`
- `puts stdout $var`
 - `abc`
- `puts ${var}def`
 - `abcdef`
- `unset var`

- ✓ 변수에는 타입이 없다.
 - ✓ C언어에서의 `int`, `char`, `float`같은 타입은 없다.
- ✓ `set` 커맨드로 변수에 값을 대입.
- ✓ `set` 커맨드로 값을 생략 시 변수 값 참조.
- ✓ 변수명의 처음에 `$`을 붙여주면 값을 참조.
- ✓ `unset` 메모리에 변수 삭제.

산술연산

- `expr 1 / 0`
 - divide by zero
 - `expr 10 + 0x10 + 010`
 - 34
 - `expr 2.0 * asin(1.0)`
 - 3.14159265359
 - `set i 1`
 - 1
 - `incr i`
 - 2
- ✓ `expr` 커맨드 사용
 - ✓ 16진수,8진수도 대입 가능
 - ✓ 16진수 `0x10` → 10진수로 16
 - ✓ 8진수 `010` → 10진수로 8
 - ✓ 다양한 수학함수 지원
 - ✓ 난수(random) 지원

커맨드의 치환 (Substitution)

- `set x [expr 10 * 2]`
 - 20
- `puts $x`
 - 20
- `set a 10; set b [set a]`
 - 10

✓ [...] 는 커맨드의 실행 결과로 치환

더블 쿼테이션 "... " 과 중괄호{...}

- `set var 123`
 - `123`
- `puts "result = $var"`
 - `result = 123`
- `puts {result = $var}`
 - `result = $var`
- `puts "result = \ $var"`
 - `result = $var`
- `set var 123`
 - `123`
- `puts "result = [pwd]"`
 - `result = C:/`
- `puts {result = [pwd]}`
 - `result = [pwd]`
- `puts "result = \ [var]"`
 - `result = [var]`

- ✓ "... "와 {...}는 여러 개의 문자열을 하나로 묶어줌.
- ✓ 더블 쿼테이션 내에서는 변수와 커맨드가 치환이 됨
- ✓ 중괄호 내에서는 변수와 커맨드가 치환이 안됨
- ✓ 백 슬래시는 '[' 와 '\$' 의 치환을 무효화

format 서식

- scan "123.456" "%d.%d" a b
 - 2
 - set a
 - 123
 - set b
 - 456
 - format "%d.%d" \$a \$b
 - 123.456
- ✓ C언어의 scanf, printf 와 동일한 서식 지원
 - ✓ scan은 서식에 맞춰 문자열 분리
 - ✓ format은 서식에 맞게 문자열 작성

procedure(함수)

보통의 인자

```
set foo 3
```

→3

```
proc add {a b} {  
    global foo  
    return [expr $a + $b + $foo]  
}
```

```
add 1 2
```

→6

- ✓ proc 커맨드로 0개 이상의 인자를 갖는 함수를 정의
- ✓ 함수 내에서 선언된 변수는 함수 내에서만 참조가능
- ✓ 다른 영역의 변수를 참조하고자 한다면 global을 사용

procedure(함수)

참조형 인자와 기본 인자

```
proc plus {a {b 1}} {  
    upvar $a r  
    set r [expr $r + $b]  
    return $r  
}
```

```
set foo 1
```

```
→1
```

```
plus foo
```

```
→2
```

```
plus foo 2
```

```
→4
```

- ✓ 인자의 기본 값을 가질 수 있다.
- ✓ 인자에 변수의 값이 아닌 변수의 이름 자체를 넣고자 한다면 upvar를 사용한다. (call by name)

procedure(함수)

가변 인자

✓ 가변인자는 args를 사용한다.

```
proc sum {args} {  
    set s 0  
    foreach i $args {  
        set s [expr $s + $i]  
    }  
    return $s  
}
```

sum 1 2

→ 3

sum 1 2 3

→ 6

namespace

```
namespace eval Hello {  
  variable var {Hello World}  
  proc print {} {  
    variable var  
    puts $var  
  }  
}
```

```
namespace eval Hello2 {  
  variable var {Hello World}  
  proc print {} {  
    variable var  
    puts $var  
  }  
}
```

- ✓ namespace는 패키지 사이의 심볼(함수, 변수)을 충돌을 피하기 위한 기능

```
puts $Hello::var  
→Hello World  
Hello::print  
→Hello World
```


comment(주석)

주석 1

주석 2 ₩

주석 2 계속

puts {Hello World} ; #주석 3

→Hello World

✓ # 시작하는 행은 주석을 의미

✓ 스크립트 기술도중에는 ";" # 를 사용

if, elseif, else

```
set var 일  
→ 일  
if {$var == "일"} {  
  puts 1  
}
```

→ 1

```
set var 0  
→ 0  
if {$var == "일"} {  
  puts 1  
} else {  
  puts 2  
}
```

→ 2

- ✓ 지정된 조건 일때 수행한다.
- ✓ else는 생략 가능하다.

```
set var 삼  
→ 삼  
if {$var == "일"} {  
  puts 1  
} elseif {$var == "0"} {  
  puts 2  
} else {  
  puts 3  
}
```

→ 3

for

```
for {set i 1} {$i <= 3} {incr i}
{
    puts $i
}
```

→1

→2

→3

```
for {set i 1} {$i <= 5} {incr i}
{
    if {$i < 3} {
        continue
    }
    puts $i
}
```

→3

→4

→5

- ✓ 지정된 횟수만큼 loop를 수행한다.
- ✓ break는 loop를 빠져나간다.
- ✓ continue는 다음 loop을 수행한다.

```
for {set i 1} {$i <= 5} {incr i} {
    if {$i > 3} {
        break
    }
    puts $i
}
```

→1

→2

→3

while

```
set i 3  
→3  
while {$i != 0} {  
  puts stdout $i  
  incr i -1  
}  
→3  
→2  
→1
```

- ✓ 조건을 만족할 때까지 loop를 수행한다.
- ✓ break는 loop를 빠져나간다.
- ✓ continue는 다음 loop을 수행한다.

foreach

```
foreach i {A B C} {  
  puts stdout $i  
}
```

→A

→B

→C

```
foreach {i j} {A B C D E F} {  
  puts stdout "$i $j"  
}
```

→ A B

→ C D

→ E F

- ✓ 주어진 리스트만큼 loop를 수행한다.
- ✓ break는 loop를 빠져나간다.
- ✓ continue는 다음 loop을 수행한다.

switch

```
set fruit "사과"
```

```
→ 사과
```

```
switch $fruit {  
  사과    {puts 1000원}  
  귤     {puts 500원}  
  바나나 {puts 200원}  
  default {puts 모름}  
}
```

```
→ 1000원
```

- ✓ 만족 조건만 수행한다.
- ✓ default는 맞는 조건이 없을때 수행

catch

catch {expr 1+2} var

→0

catch {expr 1*_} var

→1

catch {expr 1/0} var

→1

catch {error bug} var

→1

- ✓ 에러를 catch(감지)한다.
- ✓ 에러발생시 1, 아닐시 0을 리턴

```
% expr 1*_  
invalid character "_"  
in expression "1*_"  
  (parsing expression "1*_  
  invoked from within  
"expr 1*_"  
  (file "a.tcl" line 1)
```


array 1/2

set ary(사과) 1000원

→1000원

set ary(오렌지) 500원

→500원

set ary(바나나) 2000원

→2000원

parray ary

→ary(사과) = 1000원

→ary(오렌지) = 500원

→ary(바나나) = 2000원

puts \$ary(사과)

→1000원

✓ 요소들의 모음

✓ 근본적으로 1차원 array만 가능하나
키가 문자열인 것을 이용하여 2차원
이상도 표현 가능

✓ parray는 array의 요소를 출력

✓ 키 = 값

set ary(0,0) 0

set ary(0,1) 1

set ary(0,2) 2

set ary(1,0) 0

set ary(1,1) 1

set ary(1,2) 2

array 2/2

array names ary

→바나나 오렌지 사과

array size ary

→3

array exists ary

→1

array set ary {

사과 1000원

오렌지 500원

바나나 2000원

}

array get ary

→바나나 2000원 오렌지 500원 사과 1000원

array unset ary

→ary 삭제됨

unset ary(오렌지)

→오렌지 요소 삭제됨

이 외에도 다양한 커맨드가 있으니
Tcl 공식 문서를 참고할 것

string 1/2

```
set str abc
if {[string equal $str "abc"]} {
    puts 같음
}
→같음
```

```
set str abc
if {[string match "ab*" $str]} {
    puts 같음
}
→같음
```

- ✓ 문자열 조작 커맨드
- ✓ 문자열 비교시 string 커맨드를 사용하는 것이 안전하다.

string 2/2

string length "abc"

→3

string range "abcdef" 0 2

→abc

string tolower "ABC"

→abc

string toupper "abc"

→ABC

string index "abc" 1

→b

*이 외에도 다양한 커맨드가 있으니
Tcl 공식 문서를 참고할것*

append

```
set str1 abc  
append str1 def ghi  
set str1  
→ abcdefghi
```

```
append str2 123 456  
set str2  
→ 123456
```

✓ 문자열 추가 커맨드

list

set var "사과 오렌지 바나나"

→사과 오렌지 바나나

set var {사과 오렌지 바나나}

→사과 오렌지 바나나

set var [list 사과 오렌지 바나나]

→사과 오렌지 바나나

set var [concat "사과 오렌지 바나나"]

→사과 오렌지 바나나

set var [concat {사과 오렌지 바나나}]

→사과 오렌지 바나나

set var {사과 오렌지 바나나}

→사과 오렌지 바나나

lappend var 멜론 딸기

→사과 오렌지 바나나 멜론 딸기

- ✓ 쌍 따옴표나 중괄호를 사용하여 공백으로 구분된 문자열의 집합
- ✓ 인자의 문자열로부터 리스트 생성
- ✓ 인자의 문자열로부터 공백을 기준으로 리스트 생성
- ✓ 리스트에 요소 추가
- ✓ 지정 변수 없을시 새로운 리스트 생성

list

set var "사과 오렌지 바나나"

llength \$var

→3

lindex \$var 1

→오렌지

lrange \$var 0 1

→사과 오렌지

linsert \$var 1 메론 딸기

→사과 메론 딸기 오렌지 바나나

lset \$var 0 키위

→키위 오렌지 바나나

✓ 리스트의 요소 개수 리턴

✓ 지정된 인덱스의 요소 리턴

✓ 지정된 영역의 요소를 리턴

✓ 지정된 위치에 요소를 삽입

✓ 지정된 위치에 요소를 변경

list

set var "사과 오렌지 바나나"

lsearch \$var 오렌지

→1

✓ 리스트의 요소를 검색

lsort "사과 오렌지 바나나"

→바나나 사과 오렌지

✓ 리스트의 요소를 정렬

set var {사과:오렌지:바나나:사과}

→사과:오렌지:바나나:사과

✓ 문자열을 분리하여 리스트로 리턴

split \$var :

→사과 귤 바나나 사과

*이 외에도 다양한 커맨드가 있으니
Tcl 공식 문서를 참고할것*

file i/o access

```
set fd [open "sample.txt" w]
puts $fd "hello world 1"
puts $fd "hello world 2"
close $fd
```

✓ 파일 생성

```
set fd [open "sample.txt" r]
set txt [read $fd]
puts $txt
close $fd
```

✓ 파일 읽음

이 외에도 파일 i/o 관련하여 다양한 커맨드를 제공하고 있으니 직접 찾아볼것.

file

```
proc filechk {file1 file2} {  
    set time1 [file mtime $file1]  
    set time2 [file mtime $file2]  
    if {$time1 > $time2} {  
        return "$file1 이 최근것입니다."  
    } elseif {$time1 < $time2} {  
        return "$file2 이 최근것입니다."  
    } else {  
        return "같습니다."  
    }  
}
```

```
filechk a.txt b.txt
```

- ✓ file 커맨드는 file에 관련된 다양한 기능을 제공하며
- ✓ 파일의 상태를 체크하는 기능도 제공한다.
- ✓ mtime은 modify time.

이 외에도 file에 관련된 다양한 커맨드가 있으니 Tcl 공식 문서를 참고할것

package

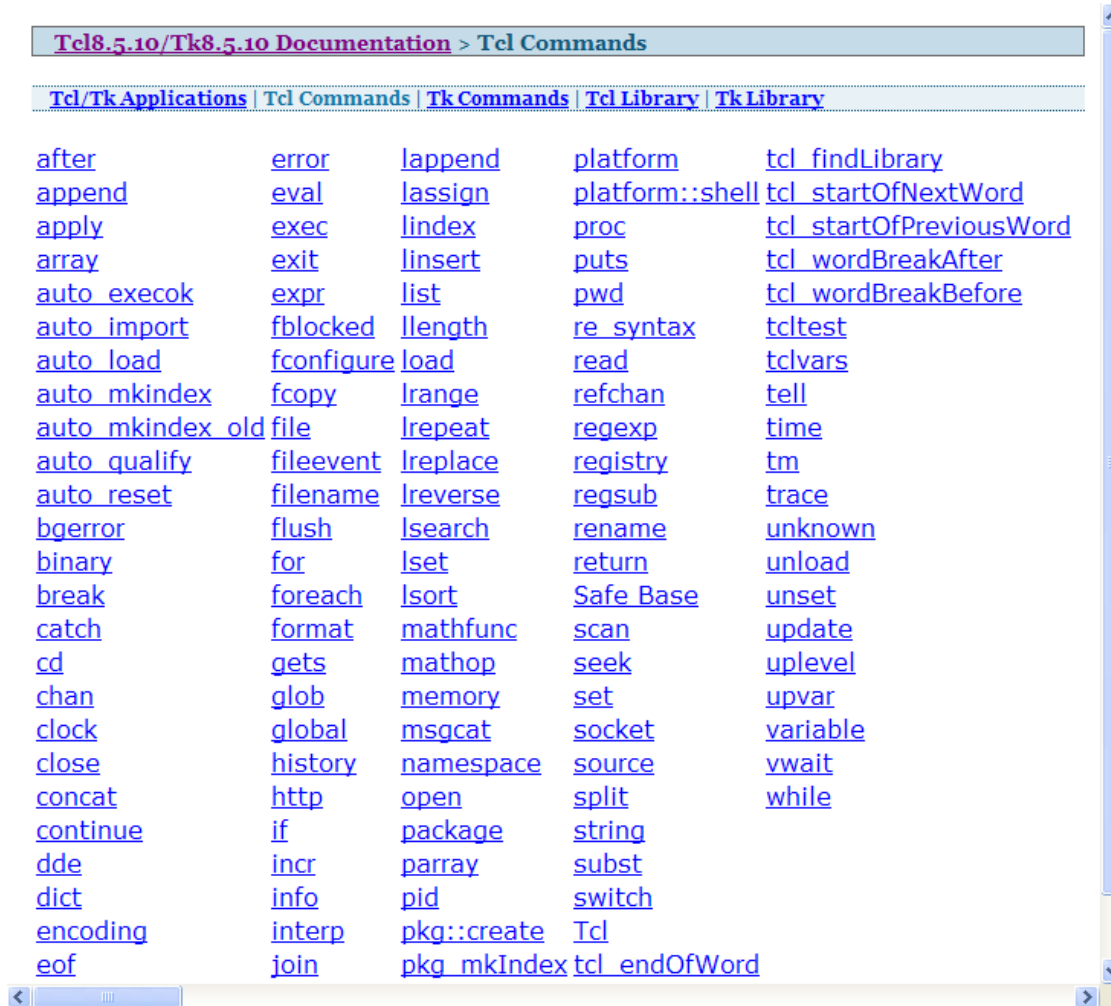
package require tcl3d
package require vtk
package require Bwidget

Speed Tables	0	8 weeks 3 days 전 admin	없음
pdf4tcl 0.7	0	10 weeks 1 day 전 admin	없음
BLT 2.4z for Tcl/Tk8.5	2	3 years 42 weeks 전 admin	16 weeks 2 days 전 admin
tile 0.8.4	0	50 weeks 1 day 전 admin	없음
BLT 2.5	0	1 year 1 week 전 admin	없음
signal 1.4	0	1 year 3 weeks 전 admin	없음
mysqtdcl 3.05	0	1 year 13 weeks 전 admin	없음
tile 0.8.3	0	1 year 13 weeks 전 admin	없음
tcl-mmap 1.1	0	1 year 13 weeks 전 admin	없음
TkDND 2.2	0	1 year 13 weeks 전 admin	없음
BWidget 1.9.2	0	1 year 13 weeks 전 admin	없음
Itcl / Itk 4.0b4	0	1 year 13 weeks 전 admin	없음
Iwidgets 4.0.1	0	1 year 13 weeks 전 admin	없음
TkTreeCtrl 2.2.10	0	1 year 13 weeks 전 admin	없음
tablelist 5.1	0	1 year 13 weeks 전 admin	없음
TkRibbon 1.0 a1	0	1 year 31 weeks 전 admin	없음

- ✓ package는 Tcl의 기본 기능에서 플러그인 형식으로 확장하는 개념으로 보통 확장패키지라 부름
- ✓ 순수 Tcl이나, C/C++로 확장 패키지를 작성함.
- ✓ package require는 확장 패키지를 인터프리터로 적재시키는 커맨드
- ✓ 확장 패키지들은 tcltk.co.kr 에서 다운 받을 수 있음.

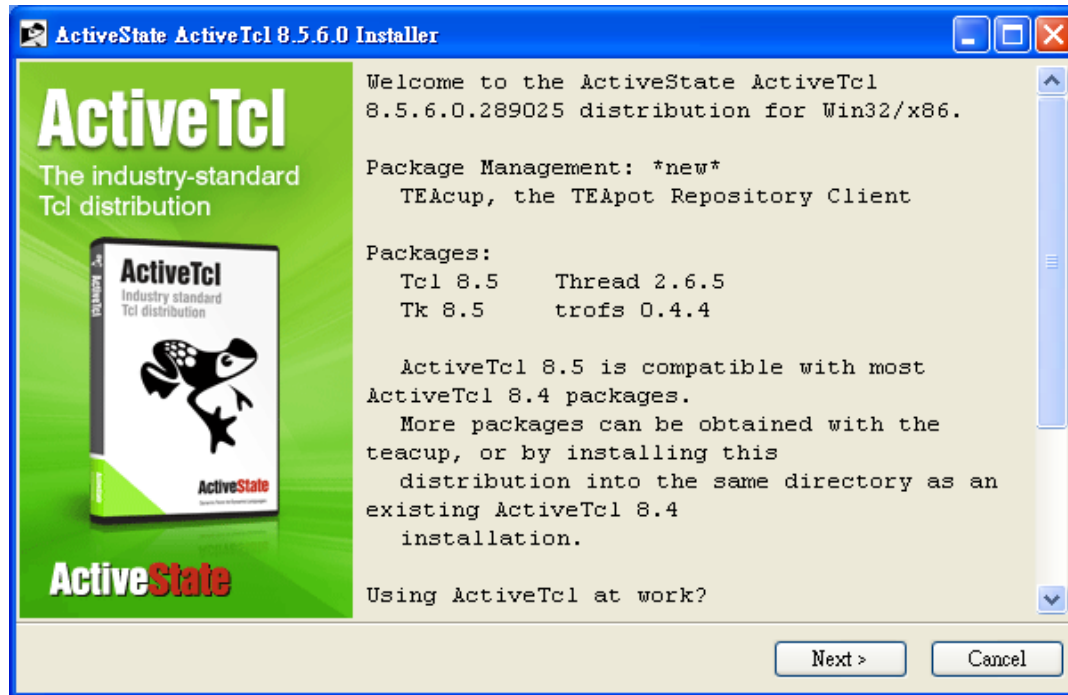


Tcl commands



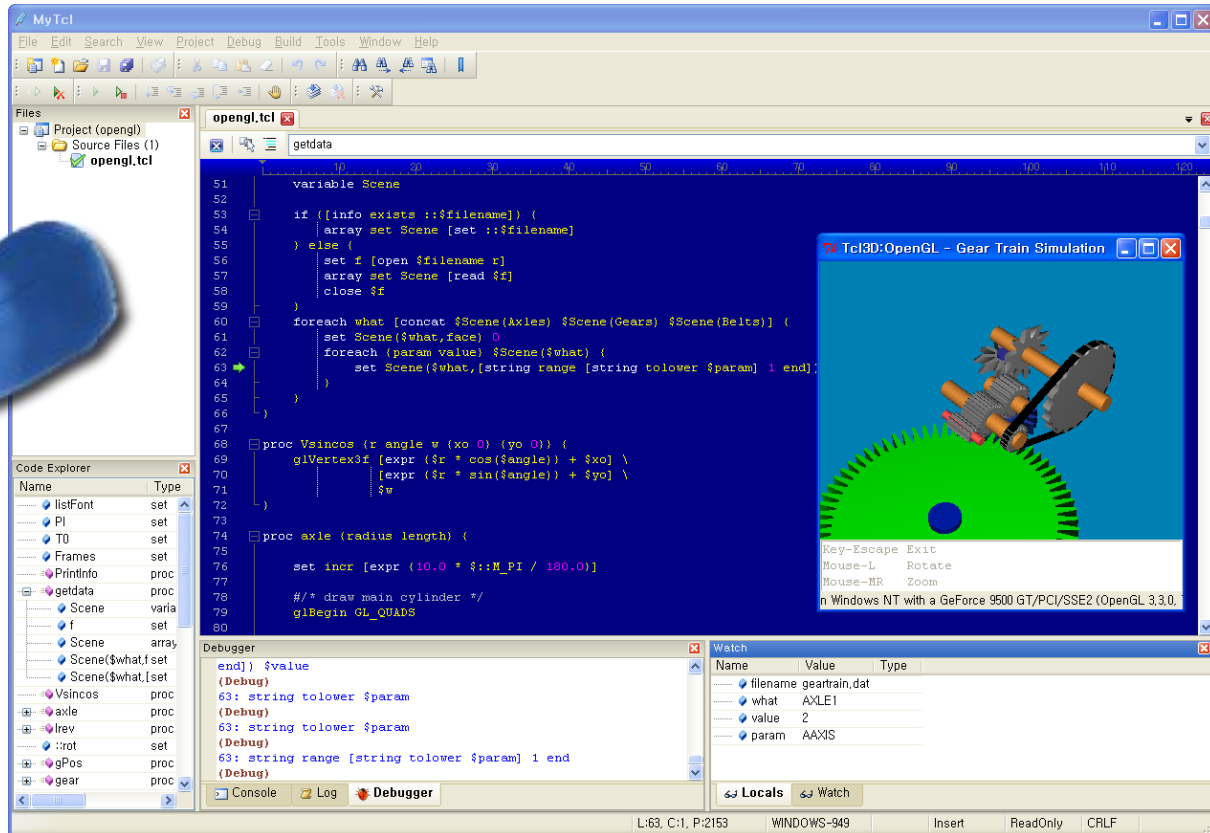
지금까지 알아본 것 외에도 다양한 커맨드를
제공하고 있으니 반드시 확인해 볼것

Tcl/Tk 개발환경



- **ActiveTcl (www.activestate.com/activetcl)**
 - ActiveState사에서 제공되는 Tcl/Tk 인터프리터
 - Tclsh, Wish만 제공됨
 - 다양한 플랫폼에서 컴파일된 Tcl/Tk 인터프리터 제공
 - Windows, Linux, Mac OS X, Solaris, AIX and HP-UX
 - 무료

Tcl/Tk 개발환경



- MyTcl (<http://mytcl.tcltk.co.kr>)
 - 개인적으로 시작한 Tcl/Tk 통합 개발환경
 - 인터프리터/ 에디터/ 코드 브라우저/ 디버거/ 빌더 제공
 - 유용하고 다양한 확장 패키지 제공
 - 현재 윈도우 설치 파일만 제공/ 차후 리눅스 버전 계획 있음.
 - 무료

감사합니다