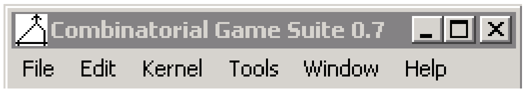
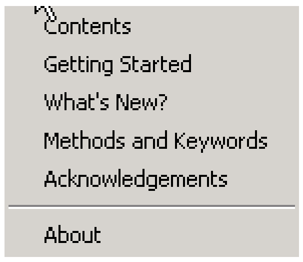
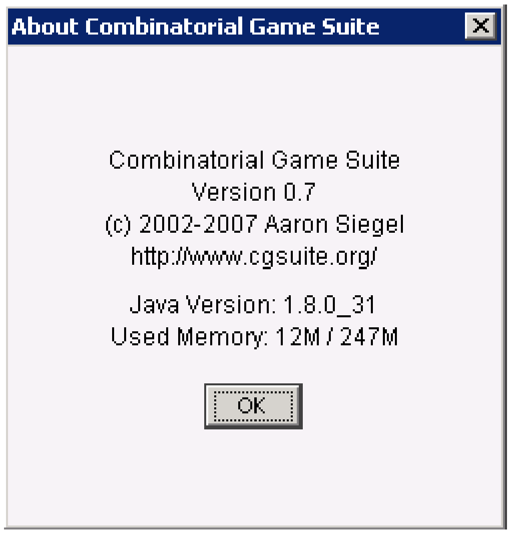
Poprvé s CGSuite



Nabídka Help

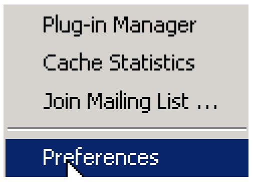



About Combinatorial Game Suite

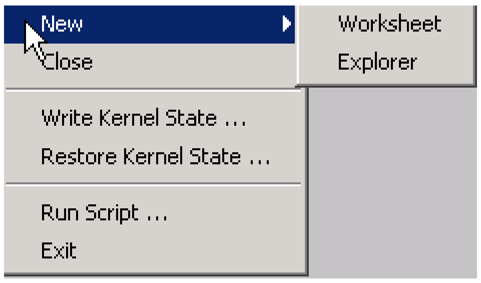
Combinatorial Game Suite
Version 0.7
(c) 2002-2007 Aaron Siegel
http://www.cgsuite.org/

Java Version: 1.8.0_31
Used Memory: 12M / 247M

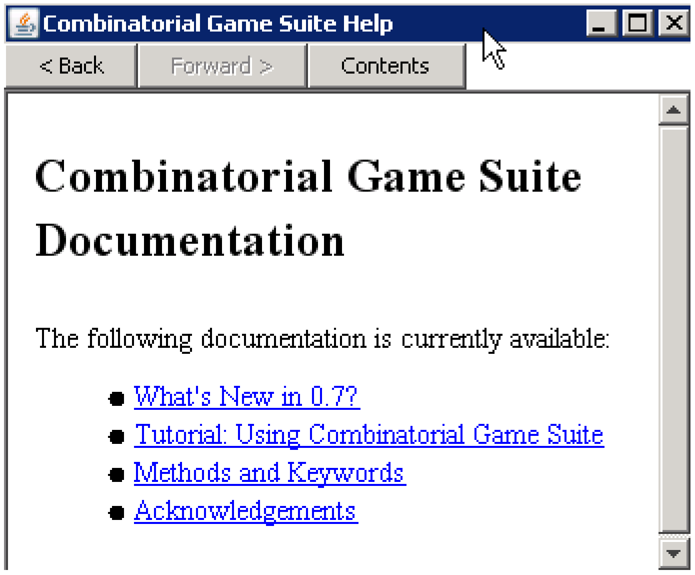
Nabídka Tools



Nabídka File



CGSuite Help

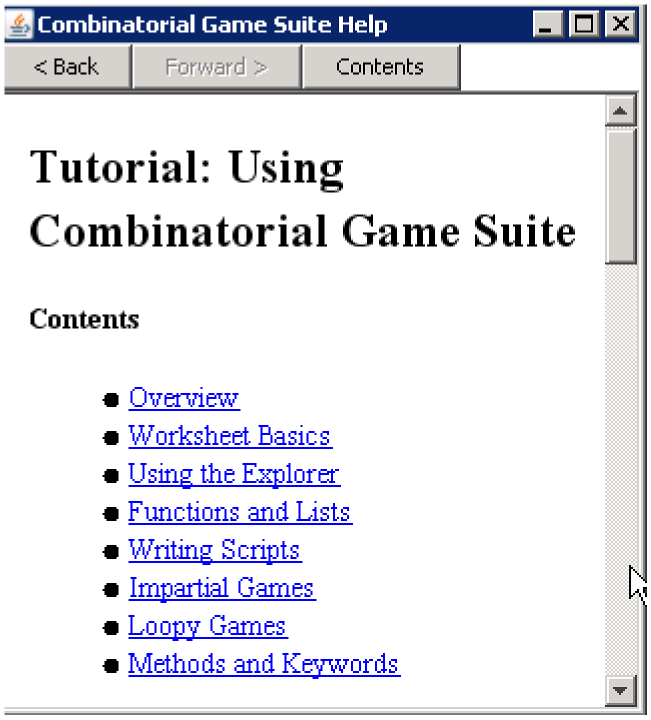


Combinatorial Game Suite Documentation

The following documentation is currently available:

- What's New in 0.7?
- Tutorial: Using Combinatorial Game Suite
- Methods and Keywords
- Acknowledgements

Tutorial



Tutorial: Using Combinatorial Game Suite

Contents

- Overview
- Worksheet Basics
- Using the Explorer
- Functions and Lists
- Writing Scripts
- Impartial Games
- Loopy Games
- Methods and Keywords

Tutorial: Using Combinatorial Game Suite

Contents
Overview
Worksheet Basics
Using the Explorer
Functions and Lists
Writing Scripts

Impartial Games
Loopy Games
Methods and Keywords

Overview
Welcome to Combinatorial Game Suite! This tutorial describes the main features of CGSuite and contains numerous examples and exercises.
The most fundamental part of Combinatorial Game Suite is the Worksheet, where you can type commands and perform calculations directly. If you're new to CGSuite, you should start with the Worksheet Basics section.
The Explorer provides a graphical editor for positions in various games (such as Amazons and Domineering). It also allows you to browse the game tree and search for good moves. The Using the Explorer section describes it in more detail.
Combinatorial Game Suite permits recursive definitions of functions - for example, to specify the value of a game in terms of the values of previous positions. That feature is documented extensively in a separate section of this tutorial, Functions and Lists.
A variety of scripting commands are also supported, such as if/then/else statements and for loops. Scripts in Combinatorial Game Suite are very similar to Maple programs; see the Writing Scripts section for more details.
Finally, there are separate tutorial segments for Impartial Games and Loopy Games.
Included with this tutorial is a comprehensive list of all Methods and Keywords available in CGSuite.
Finally, cgsuite incorporates an extensive plug-in system that can be used to add third-party support for new games. Plug-ins are discussed in a separate tutorial, How to Write a Plug-in.

Using the Explorer
Contents
Creating Game Positions
The Game Tree
Exercises

Creating Game Positions
The explorer is a graphical position editor and game tree browser. You can use the explorer to construct positions in various games, as well as to display options and move sequences in a tree.
To start the explorer, click File/New/Explorer. This will bring up a list of supported games. For this tutorial, select "Amazons Position" and click OK. You should see something similar to the following (the actual display may vary depending on your operating system):

The window is divided into three segments. In the top-left is the position being edited - currently an empty 2x5 Amazons board. In the bottom-left is a panel that displays information about positions being analyzed. We haven't created any positions yet, so no information is displayed there. The panel to the right is used to display game trees.
To edit the Amazons position, simply click on any of the empty squares on the board. This will cause a stone to appear there. Clicking again on the stone changes it to a black Amazon, and so on, cycling through all available possibilities. Use this method to place a black Amazon in the top-left square of the board and a white Amazon in the bottom-left square, so that you see the following:

Now click on the "Copy" button. This copies a text representation of the displayed Amazons position to the clipboard. Click on the Worksheet and select Edit/Paste. This inserts the text representation directly into the worksheet. You should see the following text appear in the Worksheet:
Amazons("l....","r....")
You can use this technique to create positions graphically in the explorer and copy them to the Worksheet for analysis.

The Game Tree
Click on the explorer again. The Amazons position should still be displayed. Now click on the "Add" button. This creates a node in the game tree panel and changes the displayed information to something

like the following:

If the position's canonical form doesn't appear, you can click the "Generate" button to display it.
To start exploring the options for each player, right-click on the node (the grey circle in the rightmost panel of the explorer), and choose "Expand Sensible Options" from the menu that appears. This displays a minimal set of good moves for each player. More precisely, it displays a minimal set S of options of G with the property: If Left has a winning move in G + H (for any H), then the only options from G that he needs to consider are those in S, and likewise for Right.
You'll see four nodes appear (two for each player). You can click on them to display the associated positions. For example:

Notice that the canonical forms of the sensible options correspond to G's canonical options. (This will not always be the case, however: some of the sensible options might reverse out.)
While this makes it easy to explore the good moves for each player, there are some situations when you might want to explore other options as well - for example, to determine why they are dominated. You can achieve this by editing the position manually and clicking "Add". Combinatorial Game Suite will automatically search the tree for an appropriate place to attach the new position. For example, if you make any legal move (for either player) from the position above and click "Add", the new position will appear as an option. You can then continue to explore the new option just as any other. Note that there are two ways to edit a position: Click on the squares to cycle through the available possibilities (as discussed above), or simply drag existing pieces around the board with the mouse.
To shrink or enlarge the board, simply right-click on it and select Add/Remove Row/Column.
Another feature of the explorer is worth noting. Consider the following situation: We are given a game H and its canonical form G. We'd like to know where each option of G comes from. More precisely, we'd like to find, for each option GL (say) of G, a reversible sequence of moves H, HL, HLR, HLRL, ..., HLRLR...L, terminating in a position whose canonical form equals GL. (Note that there is not necessarily a direct option HL with canonical form GL, since the relevant HL might reverse out.) If you right-click on a node in the tree and select "Expand Sensible Lines of Play", CGSuite will expand exactly one such sequence for each option of G.
Finally, you can open an explorer based on an existing game directly from the Worksheet by typing Edit(g) or Explore(g). For example:
Edit(DomineeringRectangle(4,4))

Worksheet Basics

Entering Games

You can type games directly into the worksheet. For example, click on the window labeled "Worksheet 1" and enter the following:
1/2vv*2
This represents the game 1/2 + double-down + *2. In general, you can enter combinations of numbers, ups and downs, and nimbers just as you'd expect. The symbol ^ is used for up, v for down, and * for nimbers. To enter, say, 5.up, you could type either ^^^^^ or ^5.
More complicated games can be entered using braces and slashes. For example:
{2||1|0,*}
Expressions containing slashes must be enclosed in braces. That is, to enter the game 1|0, you must type {1|0}. Ambiguous expressions, such as {1|0|-1}, will be rejected; you'd need to enter {1||0|-1} or {1|0||-1} instead.
Combinatorial Game Suite recognizes a wide variety of common games and displays them using standard shorthand notation (such as +2 for tiny-two). Here's a few to try (make sure to enter each on a separate line):
{0||0|-2} {0|v*} {1,1*|-1,-1*}

{0||0|-2} could also be entered as Tiny(2).
This tutorial contains many more examples of commands that you can enter into the worksheet. If you're interested in seeing the results, but you don't want to type everything in manually, you can copy-paste directly from this tutorial into the worksheet. Simply highlight the command with the mouse and select "Copy" from the Edit menu. Then click back on the Worksheet and select "Paste" from the Edit menu. Make sure to copy each command separately onto its own line of the Worksheet.

Operations on Games
You can enter sums and products directly. . is used for Norton product. For example:
{3|2} + {2|0} 4.{2||1|0} 1/8.^
There are nine comparison operators: == <= >= < > != <| |> <> (Respectively: equals, less than or equal to, greater than or equal to, less than, greater than, not equal to, less than or confused with, greater than or confused with, confused with.) For example, try entering:
^^ > * 0 <| ^*
Other common operations can be entered as method calls. Some examples to try:
Mean({3||2|1}) Temperature({3||2|1}) Plot(Thermograph({3||2*|1})) AtomicWeight({^^|vv}) Cool({3||2|1},1/2) Freeze({3||2|1}) Heat(*,3) Overheat(1/2,1,2) Dissociate({3||2|1}) OrdinalSum(*7,1)
Many more are available as well; see the glossary of methods for a complete list. Variable assignments are permitted, e.g.:
G := {3||2|1} Freeze(G)
And multiple statements can be strung together as a single command using semicolons, so the above could be rewritten on a single line as:
G := {3||2|1}; Freeze(G)
Note that output is generated only for the last command. If a command ends with a semicolon, no output will be generated at all.

Types and Canonical Forms
Every object in Combinatorial Game Suite has a type associated with it. Objects such as ^^, {3||2|1}, etc., are canonical games, but CGSuite also supports games that are not in canonical form. In this way, positions in a game such as Domineering are distinguished from their canonical forms.
For example, try entering the following:
H := DomineeringRectangle(4,4)
This assigns to H the Domineering position represented by an empty 4x4 grid. Note that Combinatorial Game Suite does not yet try to calculate the canonical form of H. The canonical form of H can be obtained by explicit user request, using the Solve method:
Solve(H)
Alternatively, you can type Canonicalize(H) or use the shorthand C(H). Of course we could have just typed:
C(DomineeringRectangle(4,4))
Several other games are included as examples, including Amazons, Clobber, Fox and Geese, Konane, and Toads and Frogs. Typically, positions can be entered as a sequence of strings, separated by commas; each string corresponds to a single row in the grid. For example, try entering:
A := Amazons("L..X.","R....")
As always, C(A) calculates the canonical form of A.
Entering actual game positions via the worksheet is somewhat cumbersome. A more intuitive interface is provided by the graphical explorer, which is described in the next section of this tutorial, Using the Explorer.

Functions and Lists
This section of the tutorial describes two powerful new tools that were added in version 0.4: functions and lists.
Contents

Functions

Combinatorial Game Suite permits user-defined functions. A function can be a simple mapping between objects or a more complicated recursive definition. As a simple example, the following function sets up the mapping n -> n.^:

```
f(n_) := n.^
```

The underscore _ indicates that the function is being defined for all values of n, rather than for a single specific value. After defining f, you can access its values by typing simply (for example):

```
f(2)
```

To construct a recursive definition, specify the general rule in conjunction with any number of base cases (typed in as separate statements). For example, the following is an alternative way to specify the mapping n -> n.^:

```
g(0) := 0;
g(n_) := {0|g(n-1)+*}
```

Try typing g(2) to check that it works. Notice that this is just the usual recursive definition of the canonical form of n.^.

Combinatorial Game Suite keeps track of all the previously computed values of a function. It also remembers whether a value was entered manually (as a base case) or computed automatically (using a recursive definition). If you change the function by entering new base cases, redefining old ones, or changing the recursive definition, then all of the calculated values will be cleared (but not the manually entered ones).

You can type clear(f) at any time to destroy any existing information about f.

Example: Partizan Subtraction Games

Impartial Games

This section describes CGSuite's specialized support for impartial games.

Methods and Keywords

This appendix lists all of the methods (built-in functions) and keywords currently available in Combinatorial Game Suite. G,H,S,T, etc. represent games.

Contents

General Games

Canonicalize(G)
The canonical form of G.
C(G)
Shorthand for Canonicalize(G).
LeftOptions(G)

The left options of G.
RightOptions(G)
The right options of G.
OrdinalSum(G,H)
The ordinal sum of G and H.

Canonical Games
Mean(G)
The mean value of G.
Temperature(G)
The temperature of G.
Thermograph(G)
The thermograph of G.   (Use Plot(Thermograph(G)) to plot it.)
Cool(G,t)
G cooled by the temperature t.
Chill(G)
G cooled by 1.
Freeze(G)
G cooled by its temperature.
Heat(G,T)
G heated by T.
Overheat(G,S,T)
G overheated from S to T.
Dissociate(G)
The Norton thermal dissociation of G.
LeftStop(G)
The left stop of G.
RightStop(G)
The right stop of G.
Tiny(G)
Tiny-G.
Miny(G)
Miny-G.
Pow(G,n)
The game Gn. (G must have form {0|H})
PowTo(G,n)
The game G->n. (G must have form {0|H})
Superstar (n1,n2,n3,...)
The superstar with exponent n1,n2,n3,...
LeftIncentives(G)
The canonical left incentives of G.
RightIncentives(G)
The canonical right incentives of G.
Incentives(G)
The canonical incentives of G.
Birthday(G)
The canonical birthday of G.
LeftCriticalTemps(G)
The left critical temperatures of G.
RightCriticalTemps(G)
The right critical temperatures of G.
IsAllSmall(G)
true if G is all small.
IsEven(G)
true if G is a (canonical or misere) even game.
IsInfinitesimal(G)
true if G is infinitesimal.

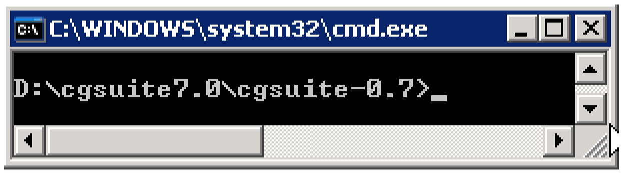
IsNimber(G)
true if G is a (canonical or misere) nimber.
IsNumber(G)
true if G is a number.
IsNumberish(G)
true if G is number-ish.
IsOdd(G)
true if G is a (canonical or misere) odd game.
AtomicWeight(G)
The atomic weight of G.
Rcf(G)
The reduced canonical form of G.
OrthodoxForm(G)
An orthodox form for G (obtained by eliminating unorthodox options for G and all of its followers).
ConwayProduct(G,H)
The Conway product of G and H.

Loopy Games

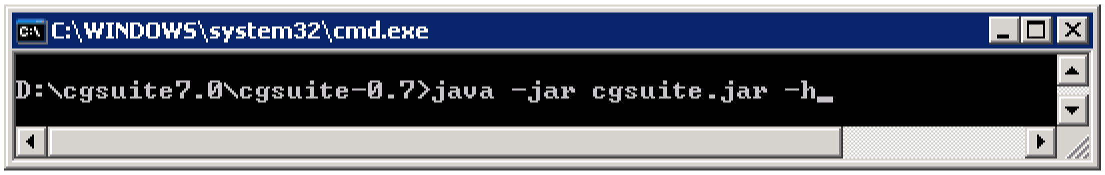
Impartial games

Specific Games
Amazons (S1,S2,S3,...)
The Amazons position with rows given by S1,S2,S3,... Each Sn should be a string consisting only of the following characters: L R . X (Example: Amazons("L...","R..."))
Clobber (S1,S2,S3,...)
The Clobber position with rows given by S1,S2,S3,... Each Sn should be a string consisting only of the following characters: L R .
Domineering (S1,S2,S3,...)
The Domineering position with rows given by S1,S2,S3,... Each Sn should be a string consisting only of the following characters: . X
DomineeringRectangle (m,n)
The empty m by n Domineering position.
Konane (S1,S2,S3,...)
The Konane position with rows given by S1,S2,S3,... Each Sn should be a string consisting only of the following characters: L R .
FoxAndGeeseTable (x1,y1,x2,y2,   x3,y3,x4,y4)
A table containing the values of all Fox and Geese positions with the specified coordinates for the geese. The coordinates are zero-based with (0,0) corresponding to the lower-left corner.   (Example: FoxAndGeeseTable(1,3,3,3,5,3,7,3))
ToadsAndFrogs(S)
The Toads and Frogs position given by S. S should be a string consisting only of the following three characters: T F .

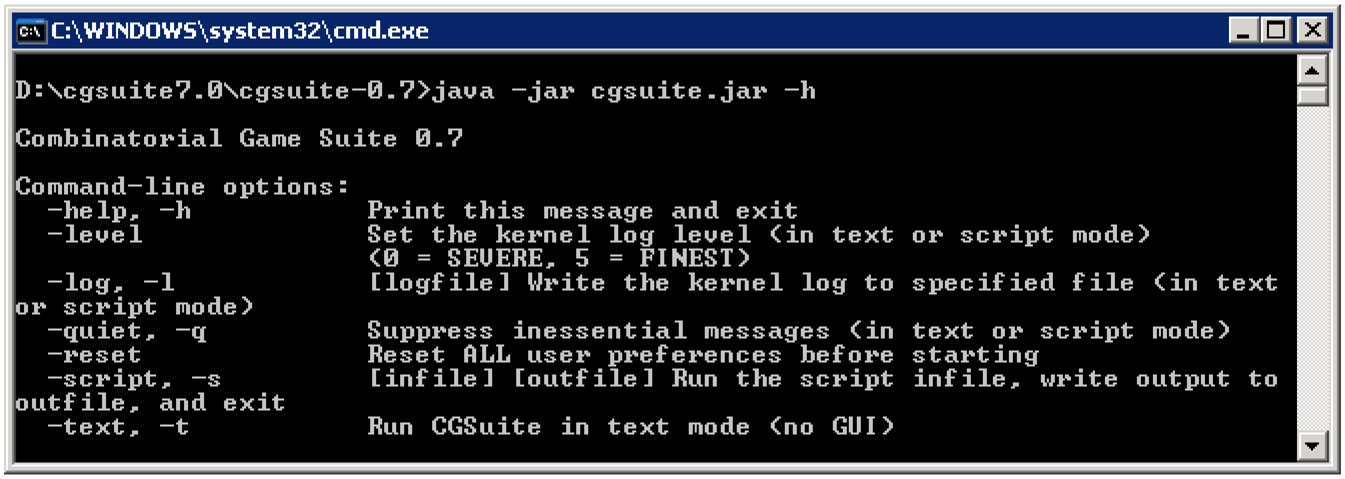
Keywords
The following are CGSuite keywords and therefore may not be used as variable names:
and break by clear continue defined do elif else end err false
fi for from if in is literally local log not od option
or out proc remember return seq tabulate then to true while
Also, variable names may not begin with "v"; it's reserved for down, double-down, etc.



```
D:\cgsuite7.0\cgsuite-0.7>_
```

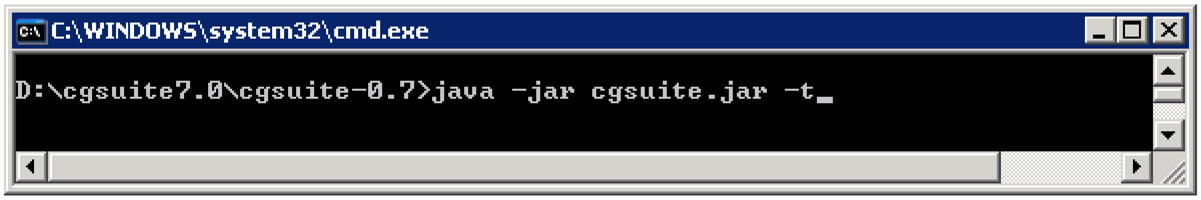


```
D:\cgsuite7.0\cgsuite-0.7>java -jar cgsuite.jar -h_
```

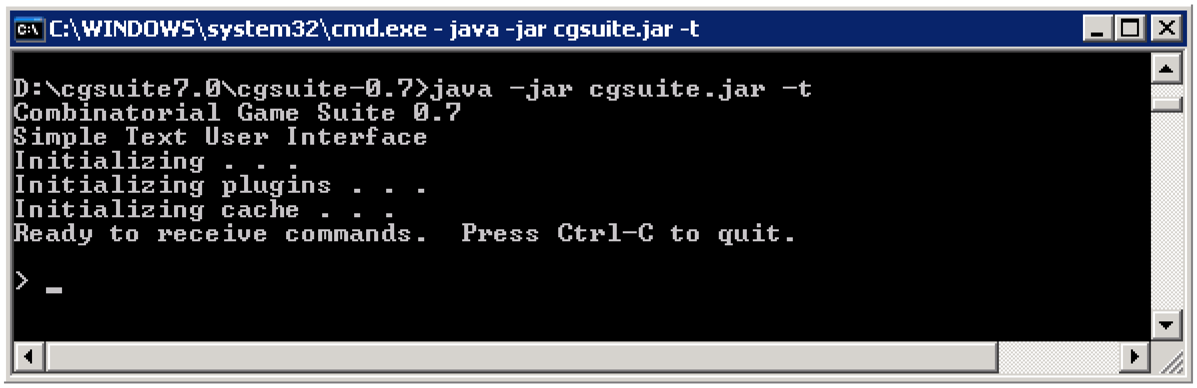


```
D:\cgsuite7.0\cgsuite-0.7>java -jar cgsuite.jar -h

Combinatorial Game Suite 0.7

Command-line options:
  -help, -h             Print this message and exit
  -level                Set the kernel log level (in text or script mode)
                        (0 = SEVERE, 5 = FINEST)
  -log, -l              [logfile] Write the kernel log to specified file (in text
or script mode)
  -quiet, -q            Suppress inessential messages (in text or script mode)
  -reset                Reset ALL user preferences before starting
  -script, -s           [infile] [outfile] Run the script infile, write output to
outfile, and exit
  -text, -t             Run CGSuite in text mode (no GUI)
```

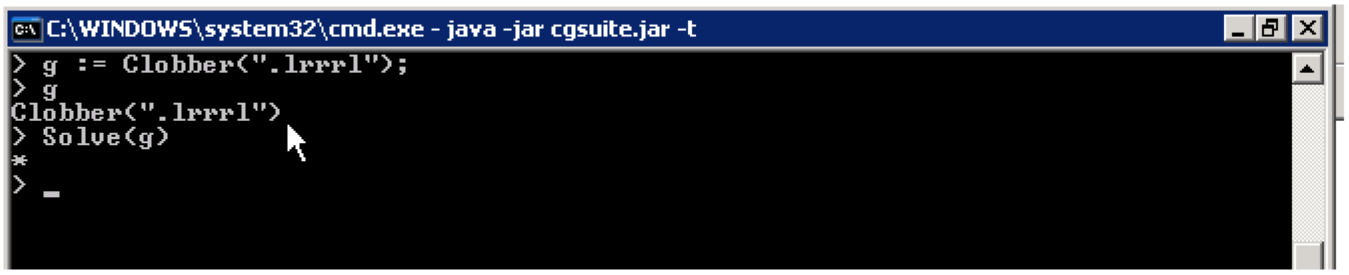
java -jar cgsuite.jar -s infile outfile



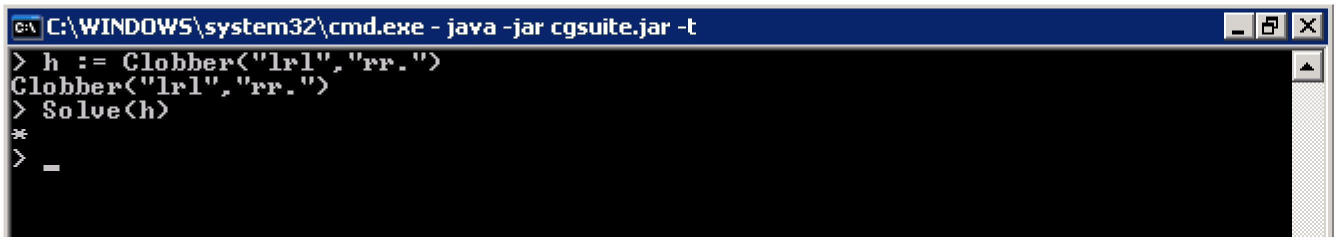
```
D:\cgsuite7.0\cgsuite-0.7>java -jar cgsuite.jar -t_
```



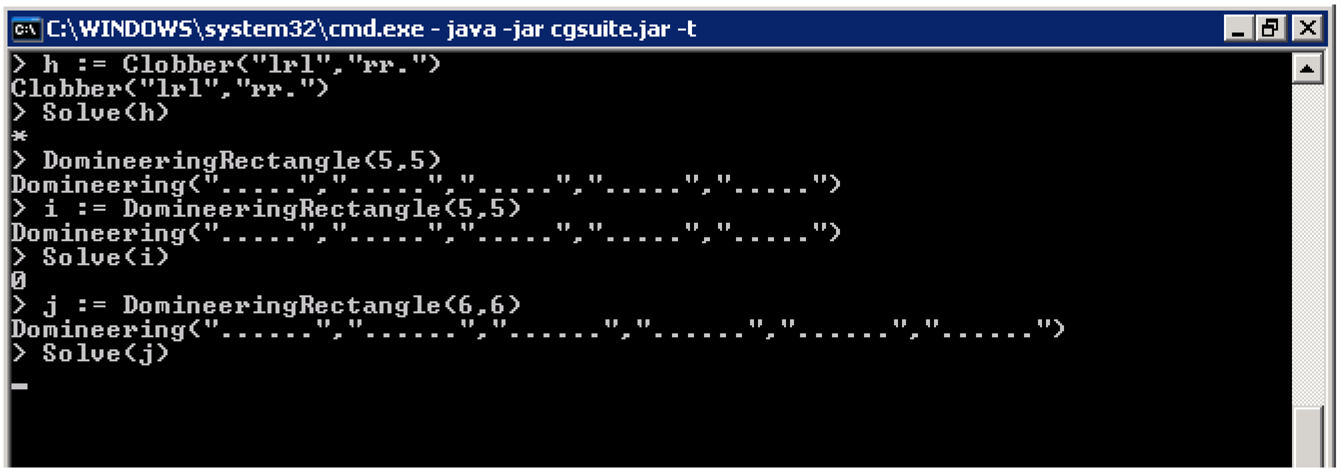
```
D:\cgsuite7.0\cgsuite-0.7>java -jar cgsuite.jar -t
Combinatorial Game Suite 0.7
Simple Text User Interface
Initializing . . .
Initializing plugins . . .
Initializing cache . . .
Ready to receive commands.  Press Ctrl-C to quit.

> _
```



```
C:\WINDOWS\system32\cmd.exe - java -jar cgsuite.jar -t
> g := Clobber(".lrrrl");
> g
Clobber(".lrrrl")
> Solve(g)
*
> _
```



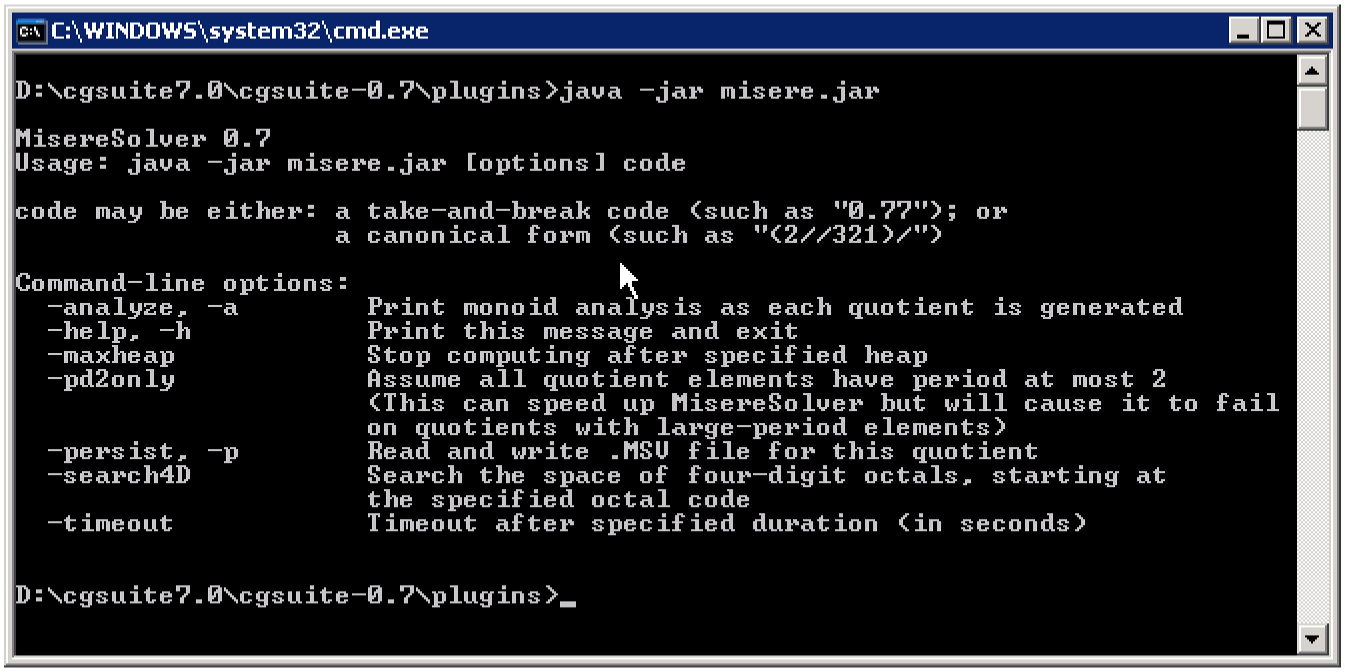
```
C:\WINDOWS\system32\cmd.exe - java -jar cgsuite.jar -t
> h := Clobber("lrl","rr.")
Clobber("lrl","rr.")
> Solve(h)
*
> _
```



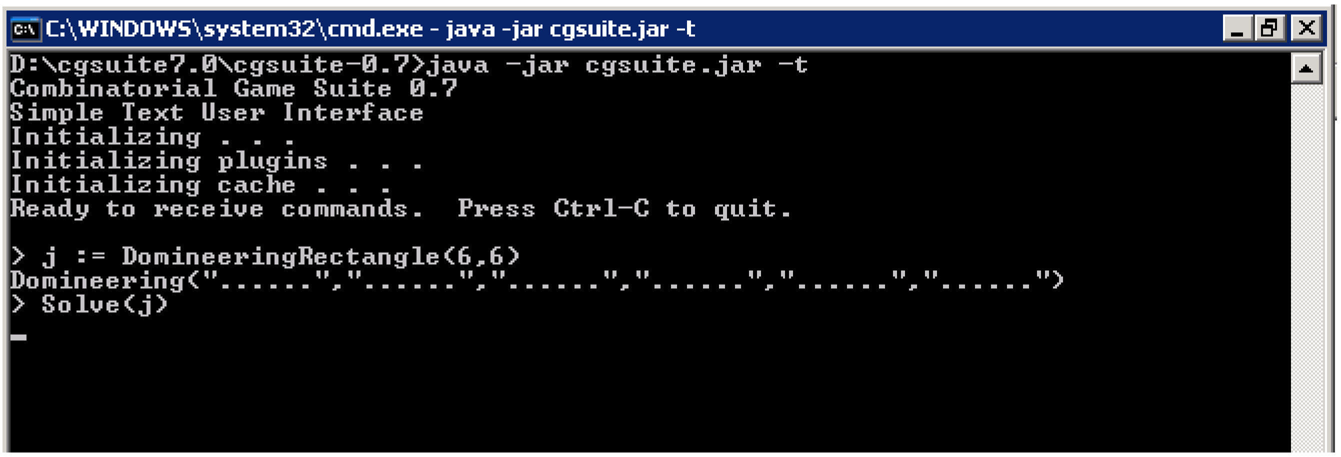
```
C:\WINDOWS\system32\cmd.exe - java -jar cgsuite.jar -t
> h := Clobber("lrl","rr.")
Clobber("lrl","rr.")
> Solve(h)
*
> DomineeringRectangle(5,5)
Domineering(".....",".....",".....",".....",".....")
> i := DomineeringRectangle(5,5)
Domineering(".....",".....",".....",".....",".....")
> Solve(i)
0
> j := DomineeringRectangle(6,6)
Domineering("......","......","......","......","......","......")
> Solve(j)
_
```

CTRL+K, resp. CTRL+C

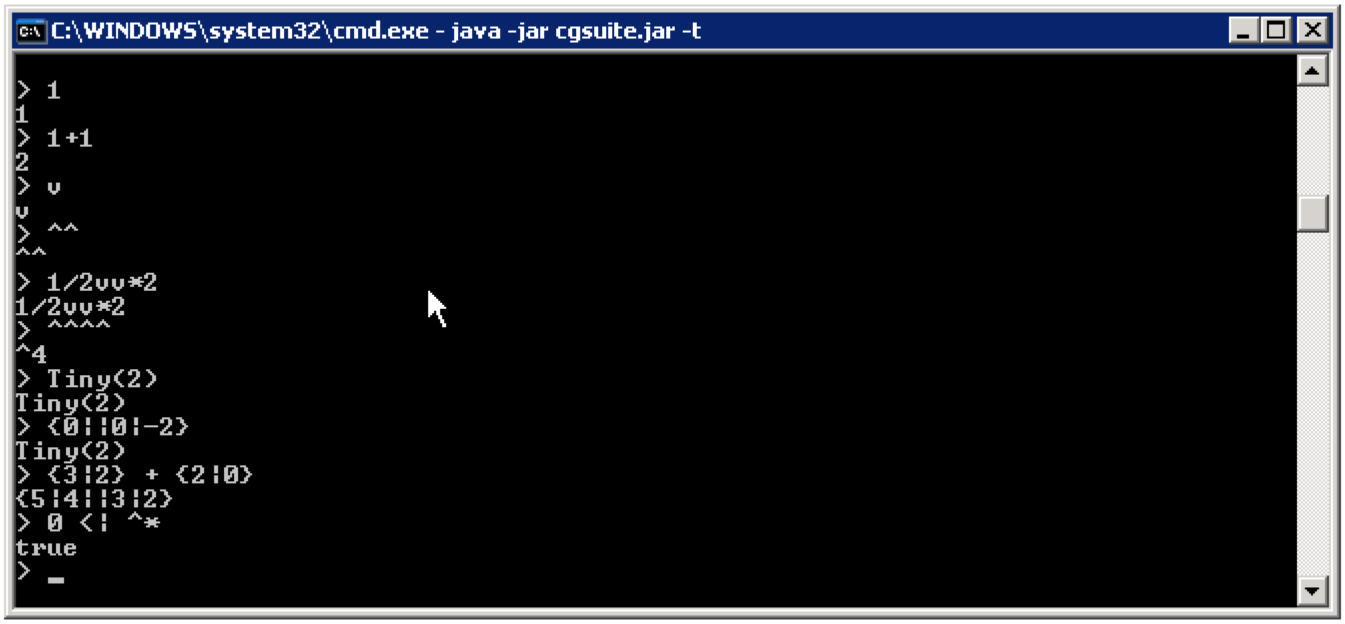
java -jar misere.jar



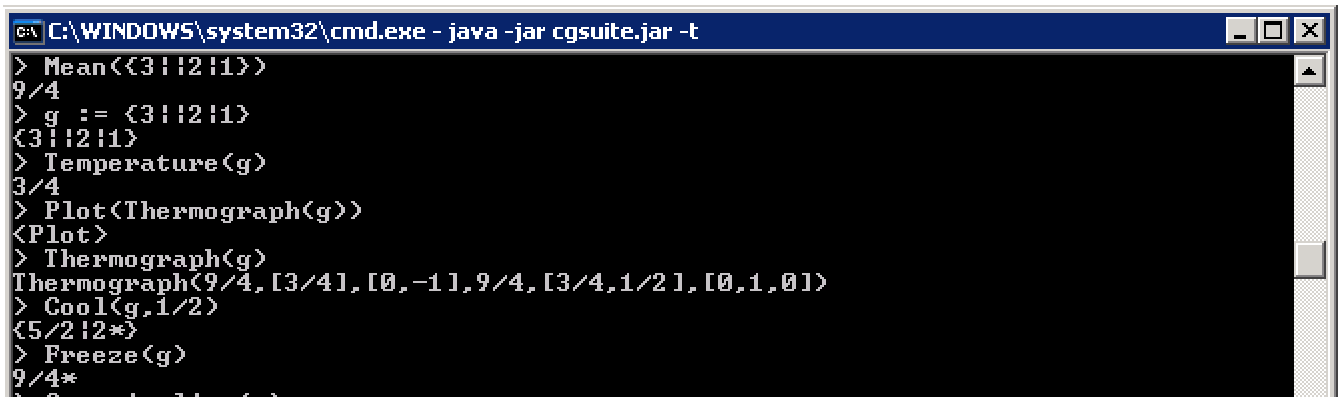
```
D:\cgsuite7.0\cgsuite-0.7\plugins>java -jar misere.jar

MisereSolver 0.7
Usage: java -jar misere.jar [options] code

code may be either: a take-and-break code (such as "0.77"); or
                    a canonical form (such as "(2//321)/")

Command-line options:
  -analyze, -a          Print monoid analysis as each quotient is generated
  -help, -h             Print this message and exit
  -maxheap              Stop computing after specified heap
  -pd2only              Assume all quotient elements have period at most 2
                        (This can speed up MisereSolver but will cause it to fail
                        on quotients with large-period elements)
  -persist, -p          Read and write .MSV file for this quotient
  -search4D             Search the space of four-digit octals, starting at
                        the specified octal code
  -timeout              Timeout after specified duration (in seconds)


D:\cgsuite7.0\cgsuite-0.7\plugins>_
```



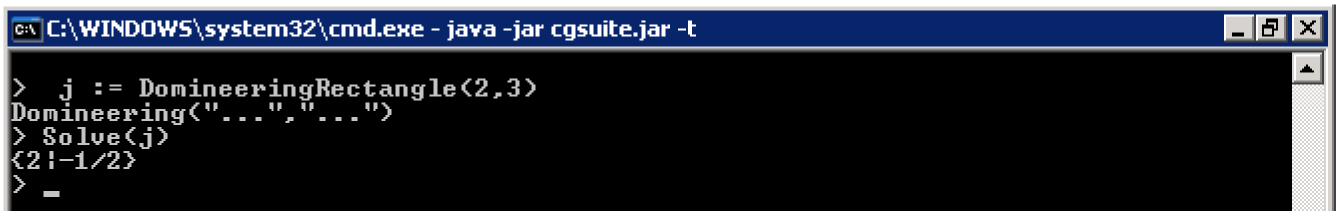
```
D:\cgsuite7.0\cgsuite-0.7>java -jar cgsuite.jar -t
Combinatorial Game Suite 0.7
Simple Text User Interface
Initializing . . .
Initializing plugins . . .
Initializing cache . . .
Ready to receive commands.  Press Ctrl-C to quit.

> j := DomineeringRectangle(6,6)
Domineering("......","......","......","......","......","......")
> Solve(j)
_
```



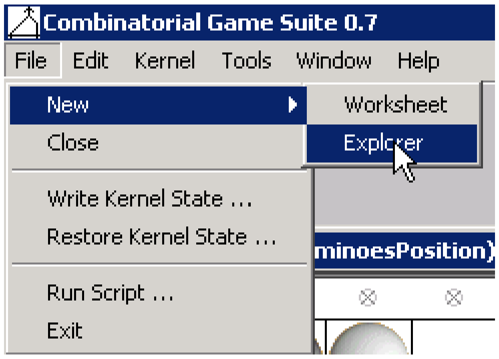
```
> 1
1
> 1+1
2
> v
v
> ^^
^^
> 1/2vv*2
1/2vv*2
> ^^^^
^4
> Tiny(2)
Tiny(2)
> {0||0|-2}
Tiny(2)
> {3|2} + {2|0}
{5|4||3|2}
> 0 <| ^*
true
> _
```

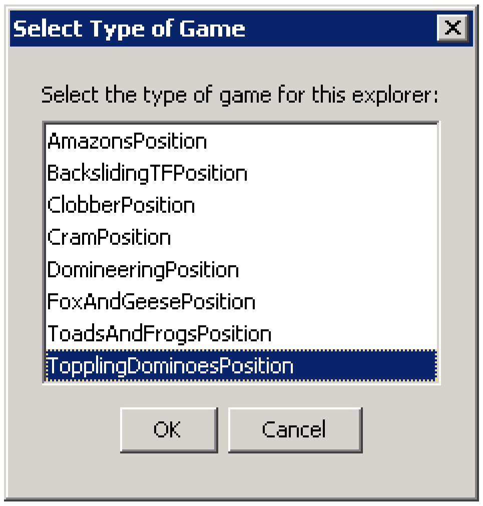


```
> Mean({3||2|1})
9/4
> g := {3||2|1}
{3||2|1}
> Temperature(g)
3/4
> Plot(Thermograph(g))
<Plot>
> Thermograph(g)
Thermograph(9/4,[3/4],[0,-1],9/4,[3/4,1/2],[0,1,0])
> Cool(g,1/2)
{5/2|2*}
> Freeze(g)
9/4*
```



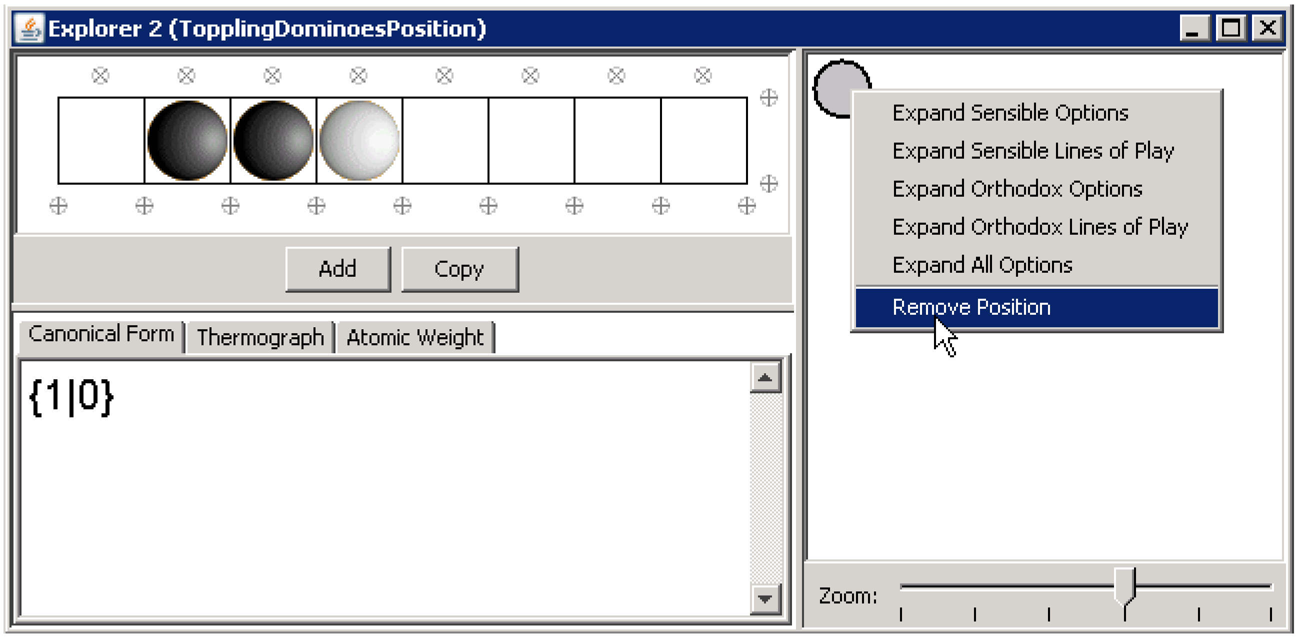
```
>  j := DomineeringRectangle(2,3)
Domineering("...","...")
> Solve(j)
{2|-1/2}
> _
```



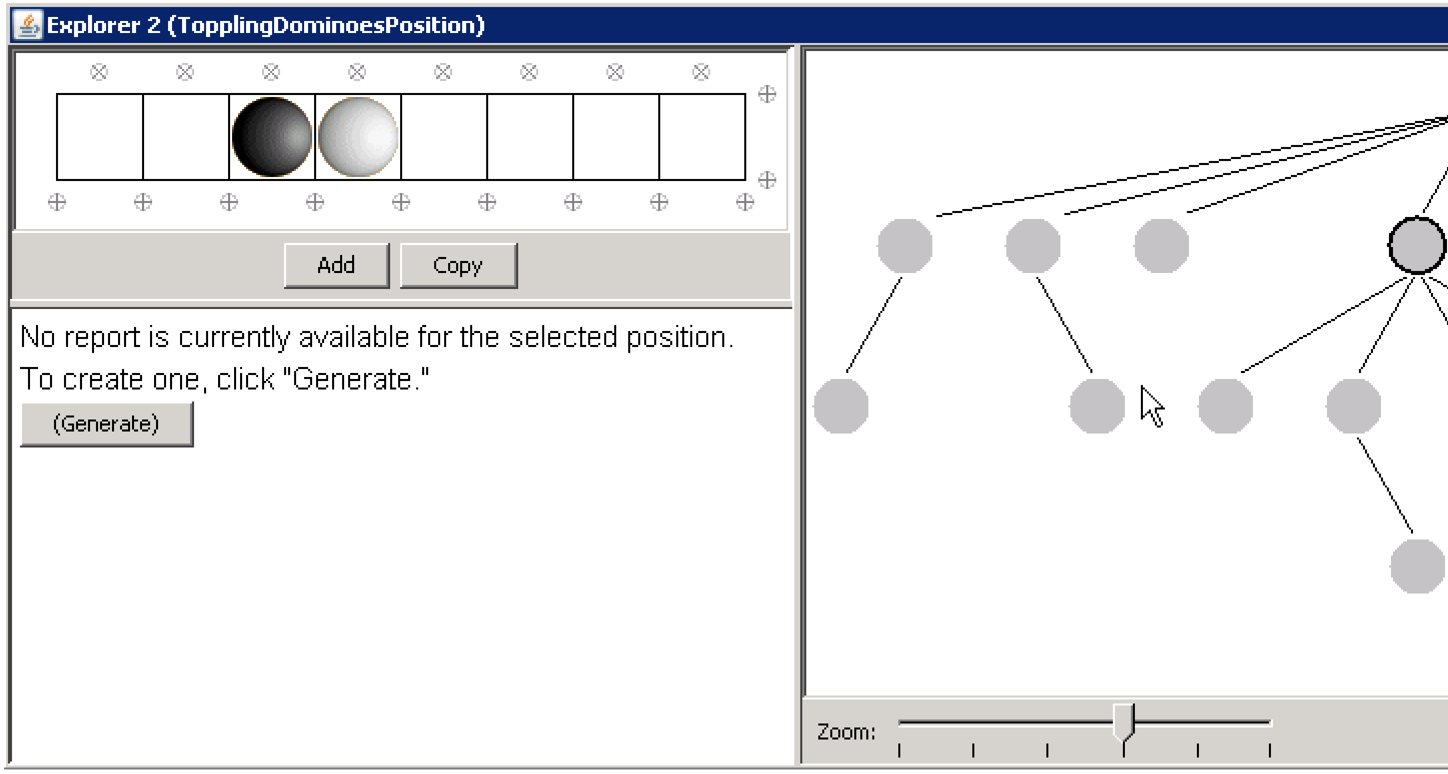

Select Type of Game

Select the type of game for this explorer:

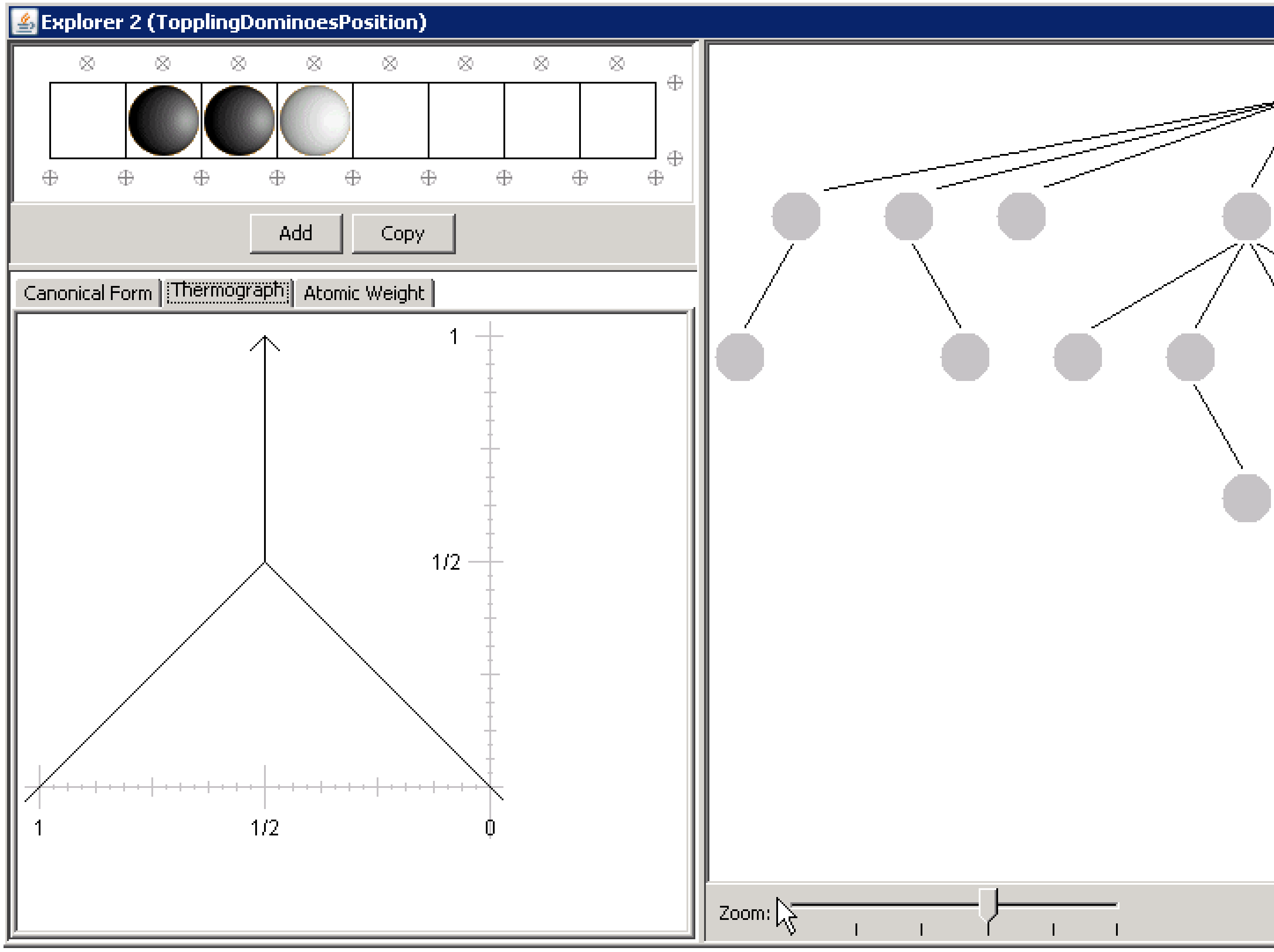
AmazonsPosition
BackslidingTFPosition
ClobberPosition
CramPosition
DomineeringPosition
FoxAndGeesePosition
ToadsAndFrogsPosition
TopplingDominoesPosition

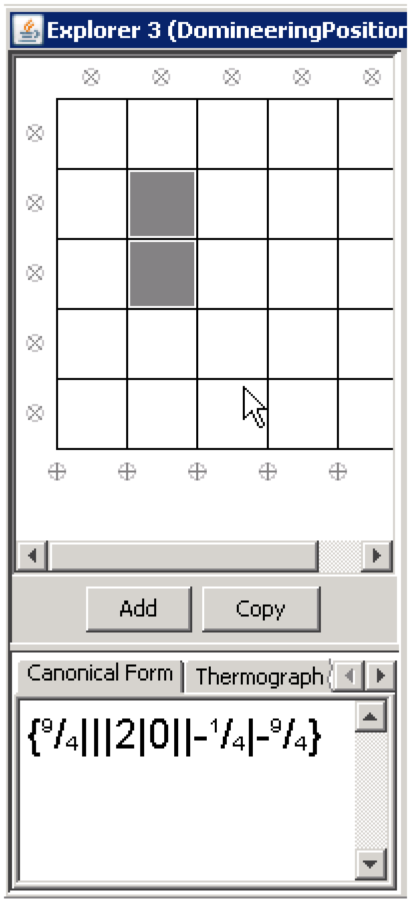

Explorer 2 (TopplingDominoesPosition)

{1|0}



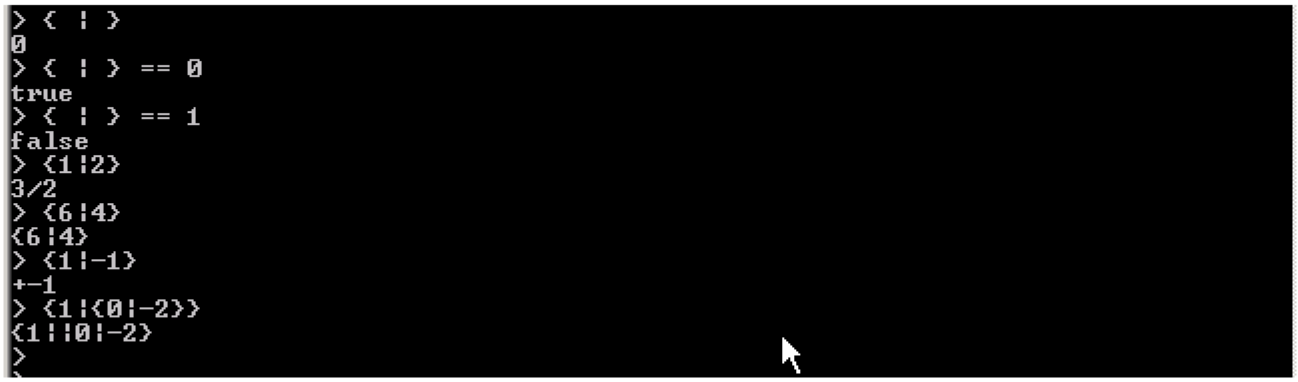
No report is currently available for the selected position.
To create one, click "Generate."


Explorer 2 (TopplingDominoesPosition)
Add
Copy
Canonical Form
Thermograph
Atomic Weight
1
1/2
1
1/2
0
Zoom:

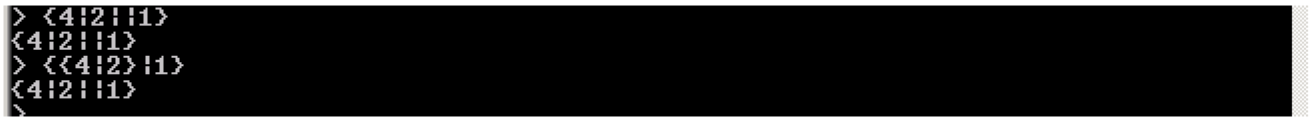


$\{^9/_4|||2|0||-^1/_4|-^9/_4\}$

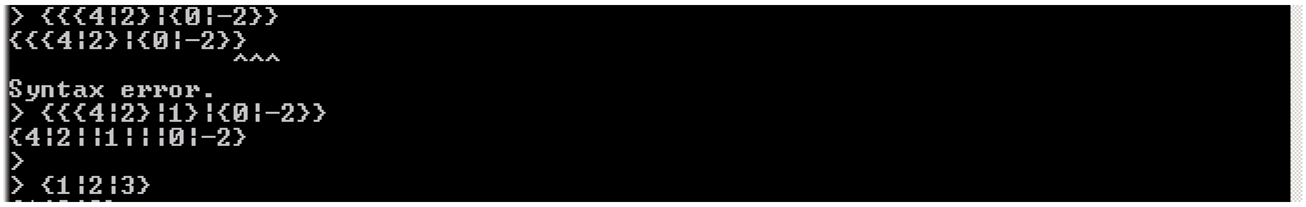
GUI/CLI



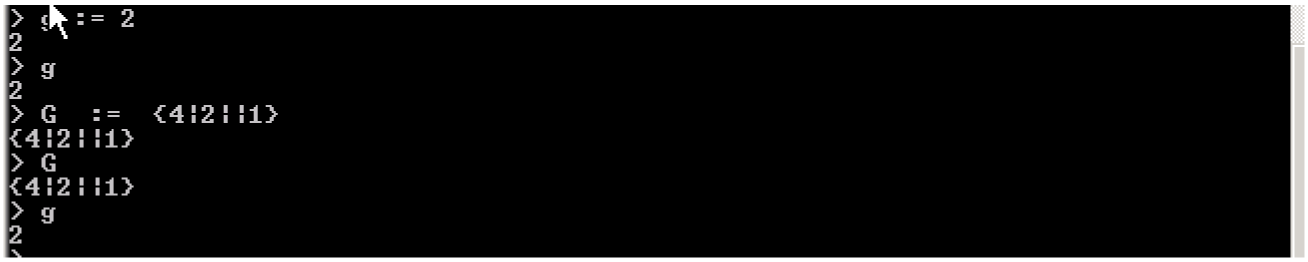
```
> { | }
0
> { | } == 0
true
> { | } == 1
false
> {1|2}
3/2
> {6|4}
{6|4}
> {1|-1}
+-1
> {1|{0|-2}}
{1||0|-2}
>
```



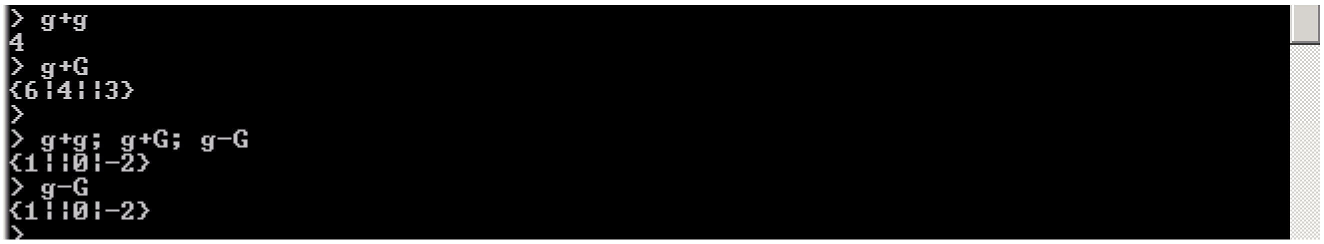
```
> {4|2||1}
{4|2||1}
> {{4|2}|1}
{4|2||1}
>
```



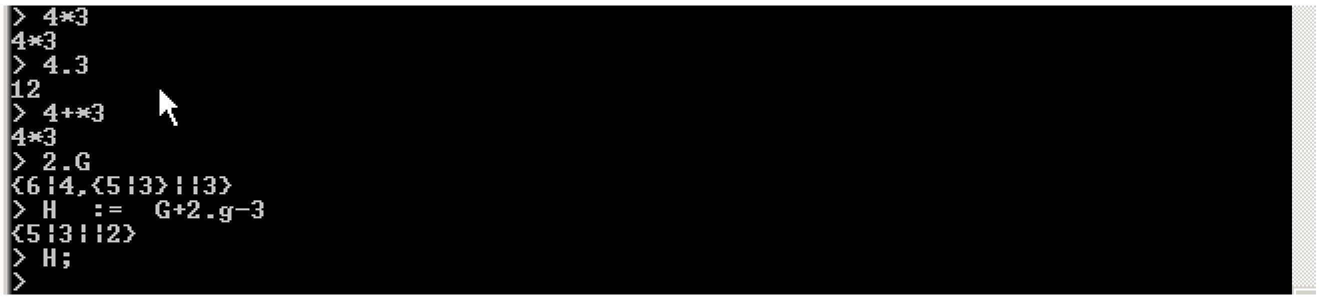
```
> {{{4¦2}¦{0¦-2}}
{{{4¦2}¦{0¦-2}}
              ^^^

Syntax error.
> {{{4¦2}¦1}¦{0¦-2}}
{4¦2¦¦1¦¦¦0¦-2}
>
> {1¦2¦3}
```



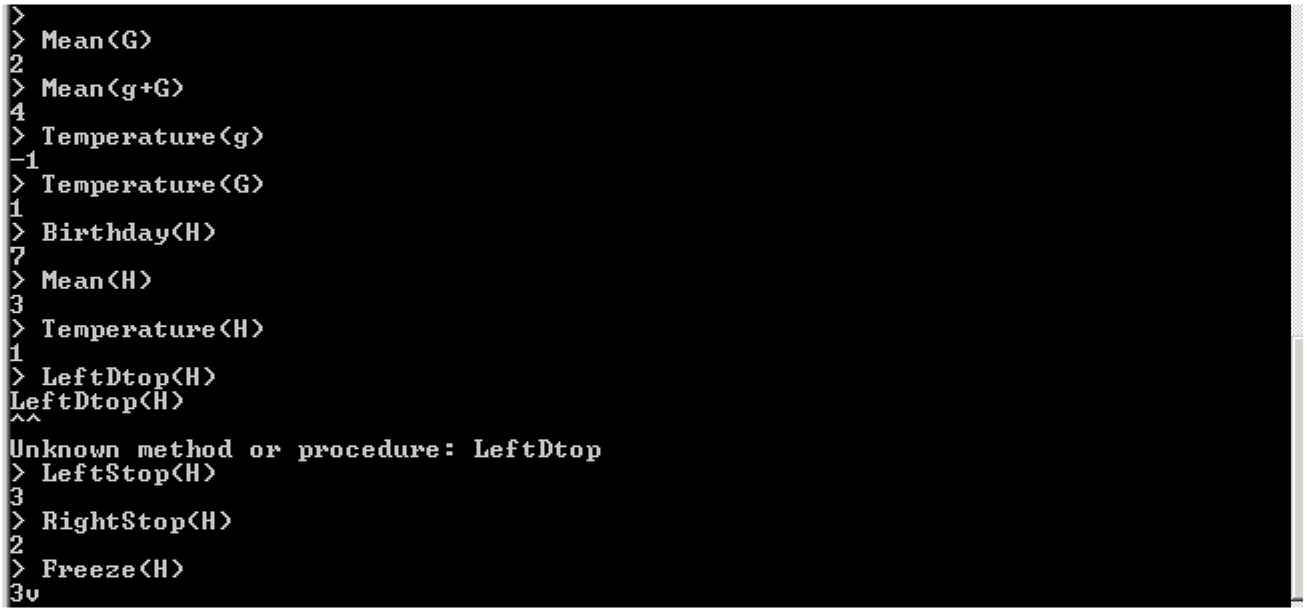
```
> g := 2
2
> g
2
> G  :=  {4¦2¦¦1}
{4¦2¦¦1}
> G
{4¦2¦¦1}
> g
2
>
```



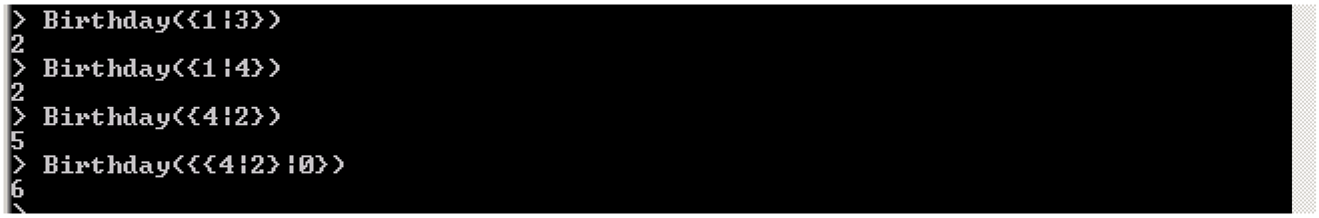
```
> g+g
4
> g+G
{6¦4¦¦3}
>
> g+g; g+G; g-G
{1¦¦0¦-2}
> g-G
{1¦¦0¦-2}
>
```



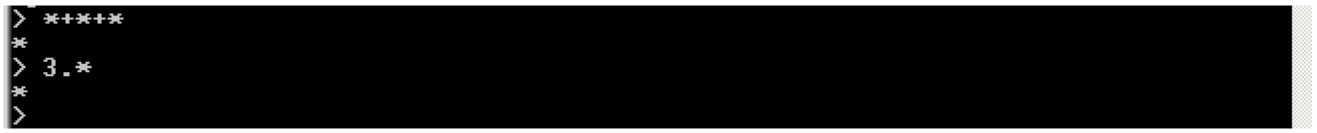
```
> 4*3
4*3
> 4.3
12
> 4+*3
4*3
> 2.G
{6¦4,{5¦3}¦¦3}
> H  :=  G+2.g-3
{5¦3¦¦2}
> H;
>
```



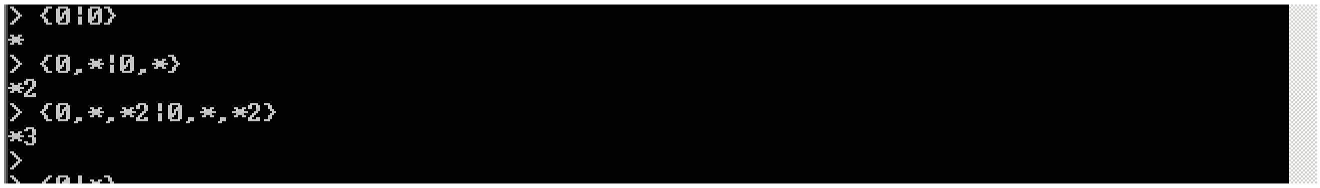
```
>
> Mean(G)
2
> Mean(g+G)
4
> Temperature(g)
-1
> Temperature(G)
1
> Birthday(H)
7
> Mean(H)
3
> Temperature(H)
1
> LeftDtop(H)
LeftDtop(H)
^^
Unknown method or procedure: LeftDtop
> LeftStop(H)
3
> RightStop(H)
2
> Freeze(H)
3v
```



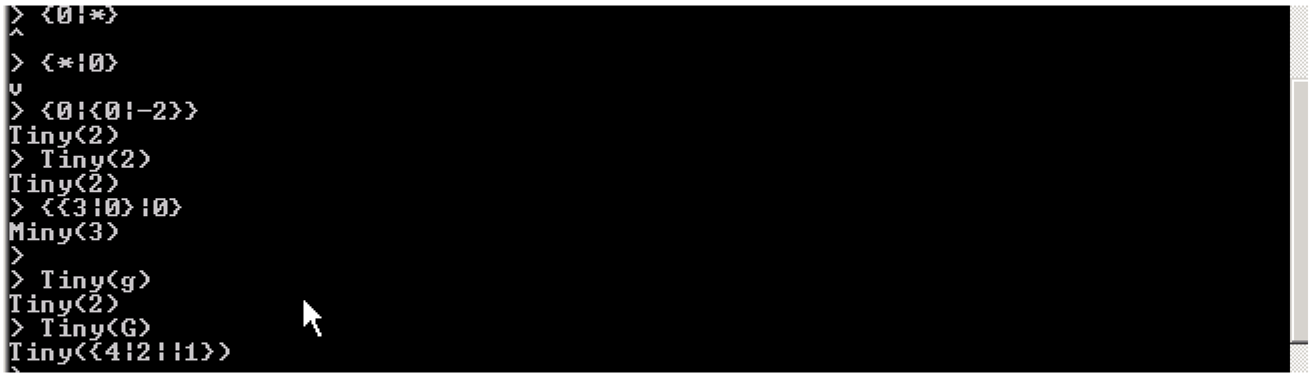
```
> Birthday({1|3})
2
> Birthday({1|4})
2
> Birthday({4|2})
5
> Birthday({{4|2}|0})
6
```



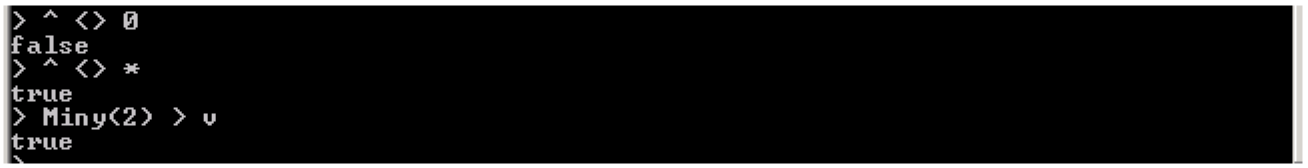
```
> *+*+*
*
> 3.*
*
>
```



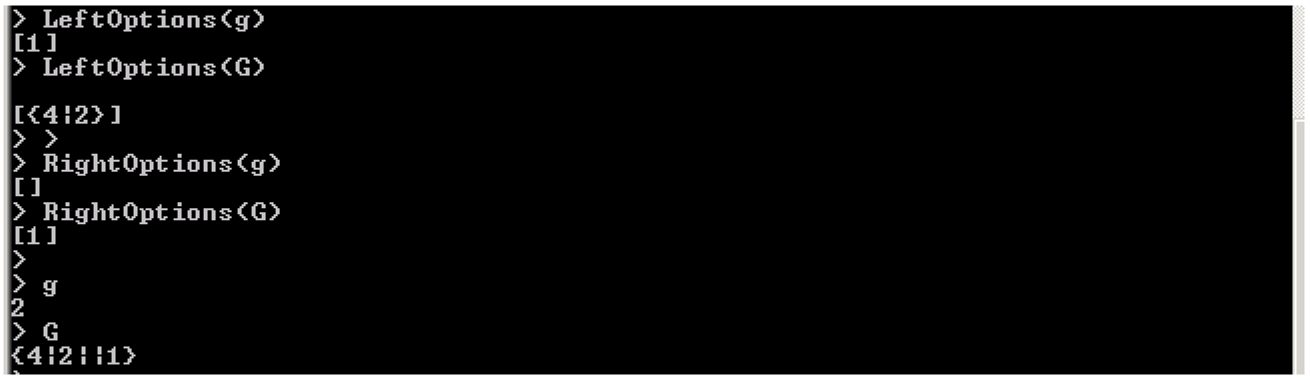
```
> {0|0}
*
> {0,*|0,*}
*2
> {0,*,*2|0,*,*2}
*3
>
```



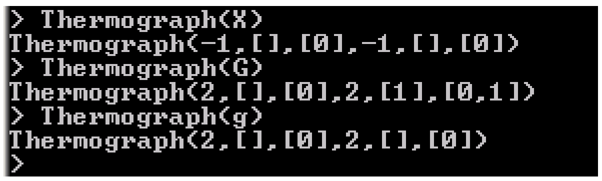
```
> {0|*}
^
> {*|0}
v
> {0|{0|-2}}
Tiny(2)
> Tiny(2)
Tiny(2)
> {{3|0}|0}
Miny(3)
>
> Tiny(g)
Tiny(2)
> Tiny(G)
Tiny({4|2||1})
```



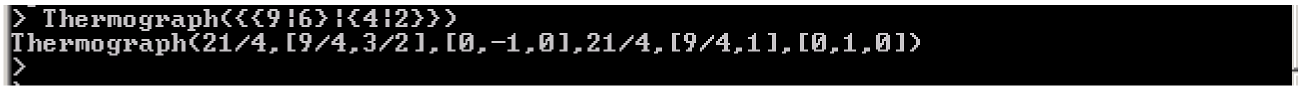
```
> ^ <> 0
false
> ^ <> *
true
> Miny(2) > v
true
```



```
> LeftOptions(g)
[1]
> LeftOptions(G)

[{4|2}]
> >
> RightOptions(g)
[]
> RightOptions(G)
[1]
>
> g
2
> G
{4|2||1}
```

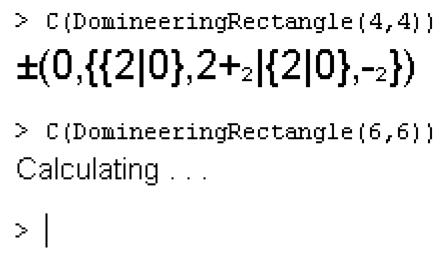


```
> Thermograph(X)
Thermograph(-1,[],[0],-1,[],[0])
> Thermograph(G)
Thermograph(2,[],[0],2,[1],[0,1])
> Thermograph(g)
Thermograph(2,[],[0],2,[],[0])
>
```



```
> Thermograph({{9|6}|{4|2}})
Thermograph(21/4,[9/4,3/2],[0,-1,0],21/4,[9/4,1],[0,1,0])
>
```

GUI



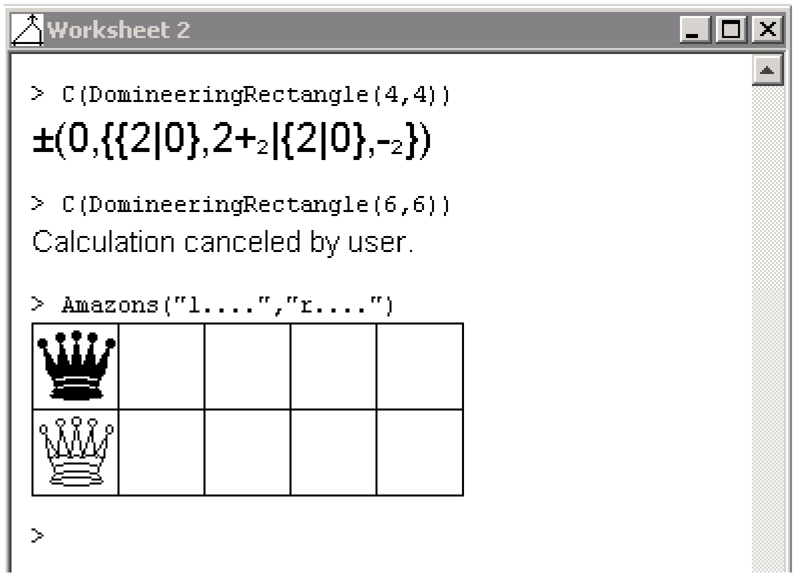
```
> C(DomineeringRectangle(4,4))
```

$\pm(0,\{\{2|0\},2+_2|\{2|0\},-_2\})$

```
> C(DomineeringRectangle(6,6))
```

Calculating . . .

```
> |
```



```
> C(DomineeringRectangle(4,4))
```

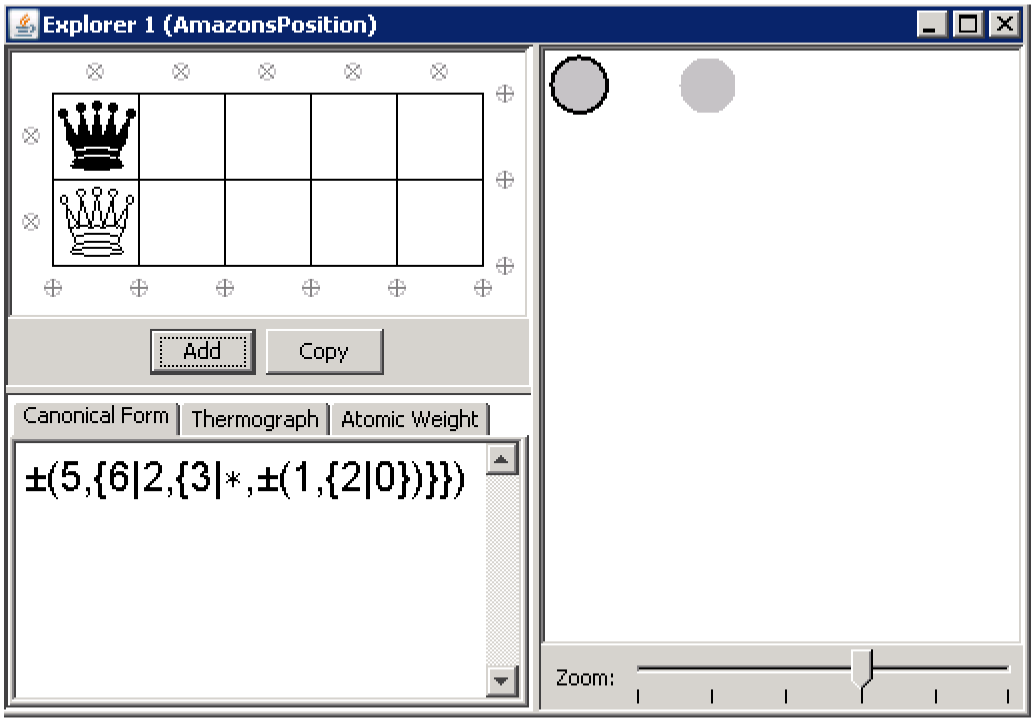
$\pm(0,\{\{2|0\},2+_2|\{2|0\},-_2\})$

```
> C(DomineeringRectangle(6,6))
```

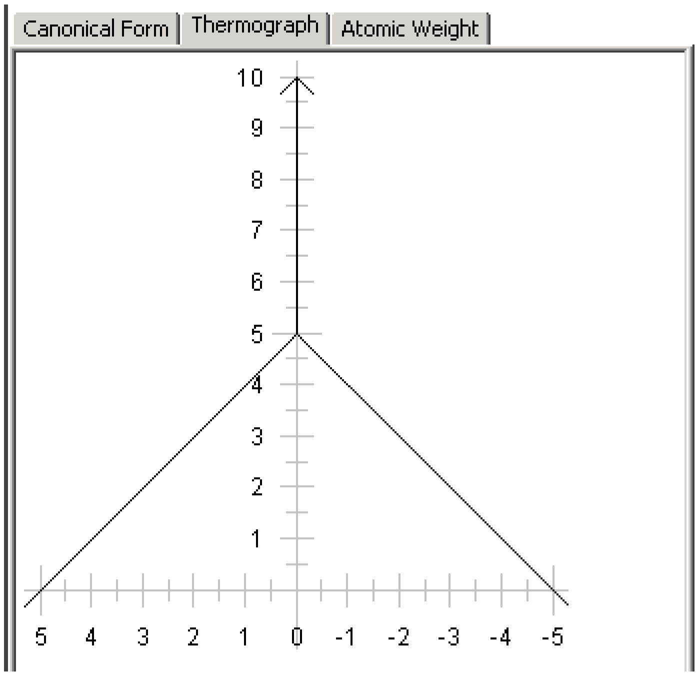
Calculation canceled by user.

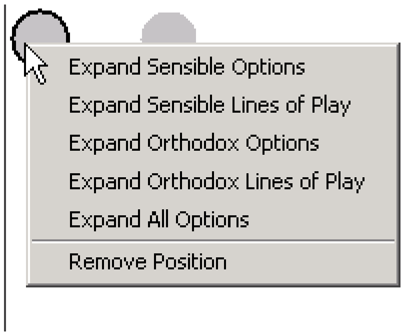
```
> Amazons("l....","r....")
```

```
>
```


Explorer 1 (AmazonsPosition)
Add
Copy
Canonical Form
Thermograph
Atomic Weight
Zoom:

±(5,{6|2,{3|*,±(1,{2|0})}})

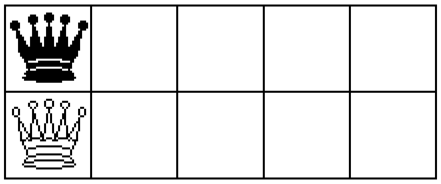

Canonical Form
Thermograph
Atomic Weight
10
9
8
7
6
5
4
3
2
1
5
4
3
2
1
0
-1
-2
-3
-4
-5


Expand Sensible Options
Expand Sensible Lines of Play
Expand Orthodox Options
Expand Orthodox Lines of Play
Expand All Options
Remove Position

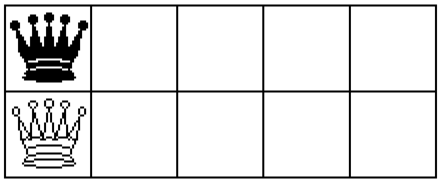
```
> C(A)
```

$\pm(5,\{6|2,\{3|*,\pm(1,\{2|0\})\}\})$

```
> A
```



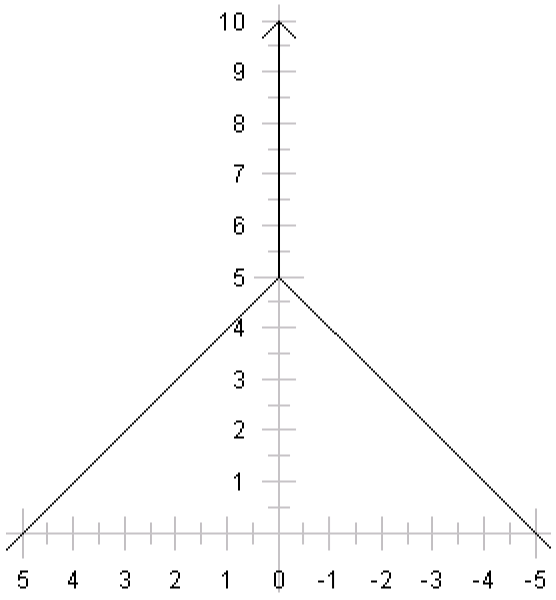
```
> A
```



```
> Thermograph(A)
```

Thermograph(0,[5],[0,-1],0,[5],[0,1])

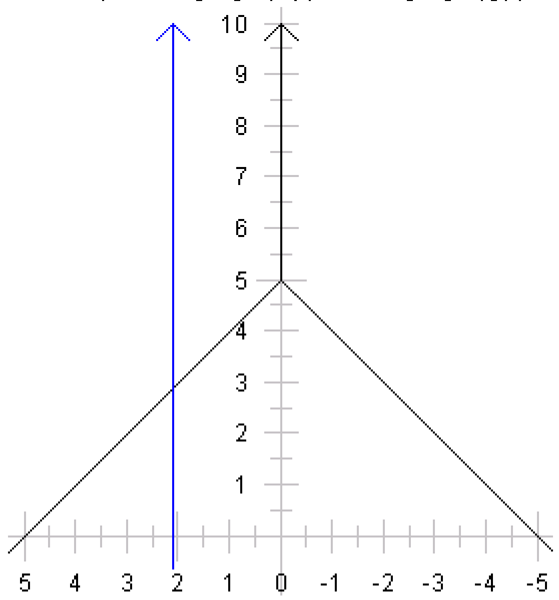
```
> Plot(Thermograph(A))
```



```
> g := {3|2||{4|2}}
```

2

```
> Plot(Thermograph(A),Thermograph(g))
```

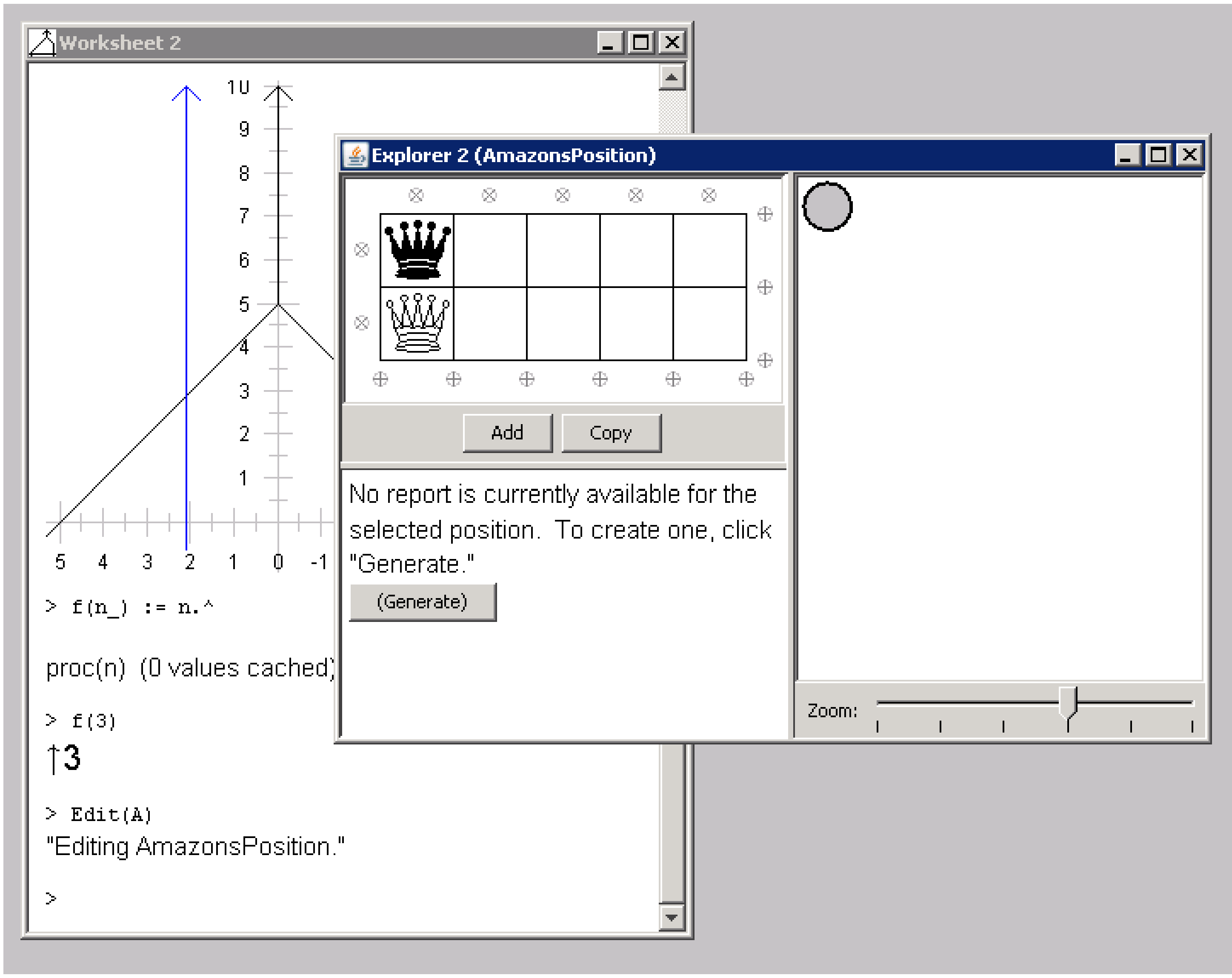


```
> f(n_) := n.^
```

proc(n) (0 values cached)

```
> f(3)
```

↑3



```
> f(n_) := n.^
```

proc(n) (0 values cached)

```
> f(3)
```

↑3

```
> Edit(A)
```

"Editing AmazonsPosition."

```
>
```

No report is currently available for the selected position. To create one, click "Generate."

```
> g(0) := 0;
  g(n_) := {0|g(n-1)+*}
```

proc(n) [0 → 0] (0 additional values cached)

```
> g(5)
```

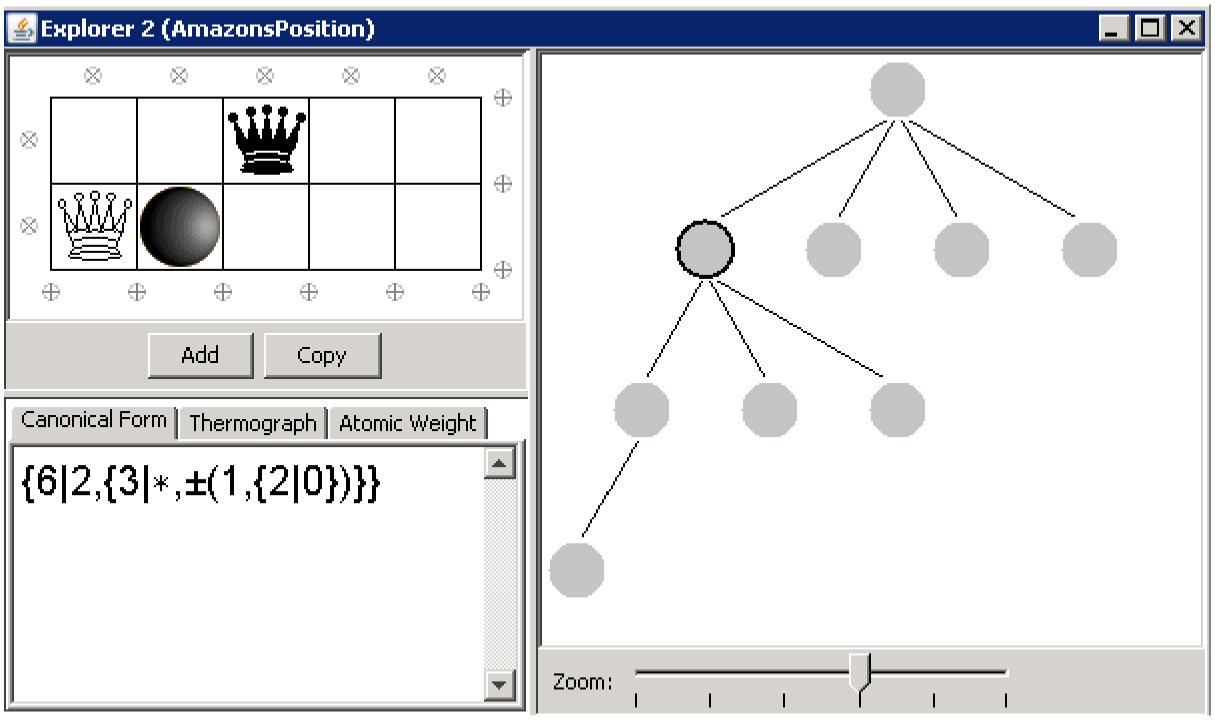
↑5

```
> v
```

↓

```
> ^
```

↑


Explorer 2 (AmazonsPosition)
Add
Copy
Canonical Form
Thermograph
Atomic Weight
Zoom:

{6|2,{3|∗,±(1,{2|0})}}