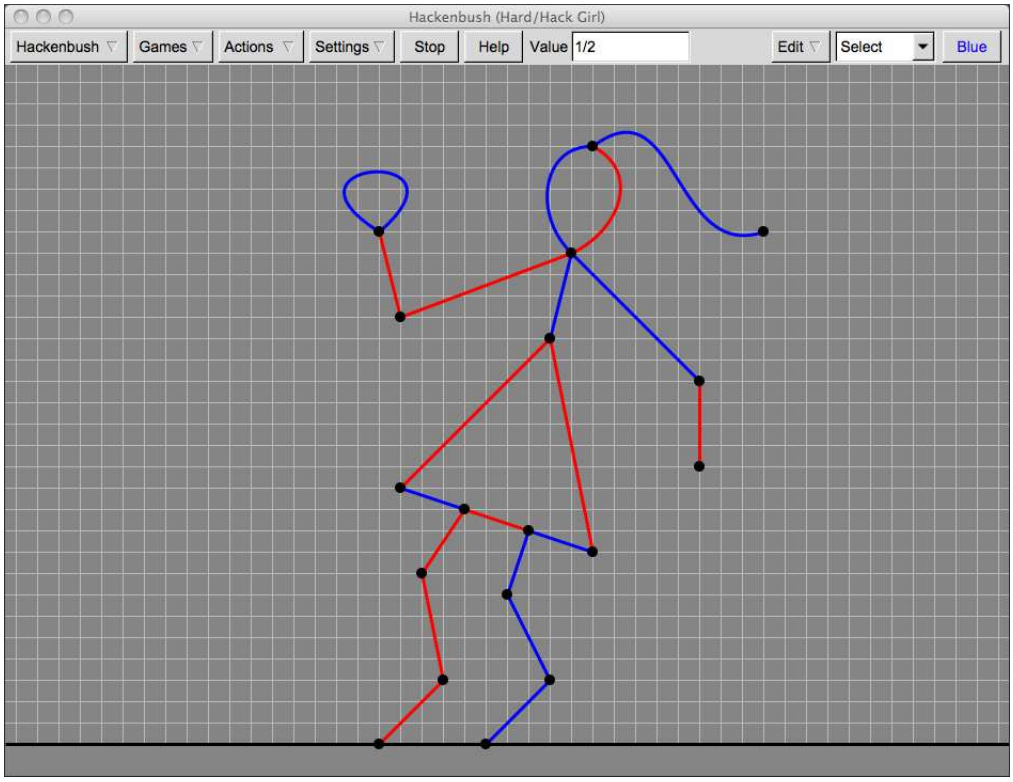
The **Hackenbush** User's Guide

Tom Davis
tomrdavis@earthlink.net
http://www.geometer.org
January 7, 2012



1 Introduction

The program **Hackenbush** lets you play red-blue hackenbush (a game invented by John Horton Conway) against the computer or against another human. It also allows you to edit and classify games. Features include:

- Play built-in games.
- Play games saved as files.
- Use the computer as a game board for use in human-human games.

- Enter your own games and analyze them according to (human and computer) difficulty.
- Search through alternative colorings of edge networks for colorings that are “interesting”.

The **Hackenbush** program’s user interface is based in part on the work of the FLTK project (http://www.fltk.org)—a darned good piece of software. The game analysis is via calculations with Surreal Numbers—a concept also invented by John Horton Conway and popularized by Donald Knuth. (See the bibliography near the end of this document.)

The software for this game (and perhaps other hackenbush-related goodies) is available here:

`http://www.geometer.org/Hackenbush`

2 Object of the Game

Players alternate cutting edge segments. The “Blue” player can only cut blue edges and the “Red” player, only red edges. After an edge is cut, it is eliminated as well as any other edges that are no longer connected to the ground via a series of other edges. (The black line near the bottom of the window is the ground.) If a player has no edges left to cut when it is his turn, he loses. (Thus, in an empty game, the first player to move would lose.)

Thus in the example program at the beginning of this document, if you are Blue and cut the girl’s hair, just the hair would disappear. But if you cut the girl’s “neck”, both arms and the head and hair would disappear.

3 Quick Overview

When playing a game, an edge is cut by clicking on it. By default, you are Blue, the machine is Red and Blue moves first. Each time you cut an edge it will blink for a couple of seconds and then it will be deleted as well as any others caused by your deletion. The machine will then choose an edge, that edge will blink, and it will then similarly disappear together with any dependent edges. Moves alternate until somebody wins.

3.1 Loading Games

If you load a game from the Games pulldown or run the tutorial, you are automatically playing the game. In tutorial mode, successive games are loaded automatically. If you are not in tutorial mode, you need to load games manually using the Games pulldown menu. There are some built-in games and there may be other game files available which have a `.hack` extension. Any games you create and save will have a `.hack` extension and will also be available.

There is a way for Blue (you, by default) to win every game in the Built-In Games choice under the Games pulldown.

4 More Playing Options

If you are in editing mode (creating your own position) and want to play the current position, click on the Play button. When you are playing a game, this button is relabeled Stop and you can press on it to stop the game if you wish to edit the game. Most other commands, like loading a new game, will stop the current game.

The Play/Stop button may also be labeled Manual if you are using **Hackenbush** as a game board in a contest between two humans.

You can undo your last move with the Take Back Move command under the Actions menu. Also under the Actions menu there is a Reset Game button that allows you to replay the current game from the start if you are in the middle of a game or have just finished it.

Under the Settings pulldown, you can toggle whether you or the machine plays blue, and whether you or the machine moves first. If you run into a game that you just don't seem to be able to win, you can switch sides and see how the machine wins it against you.

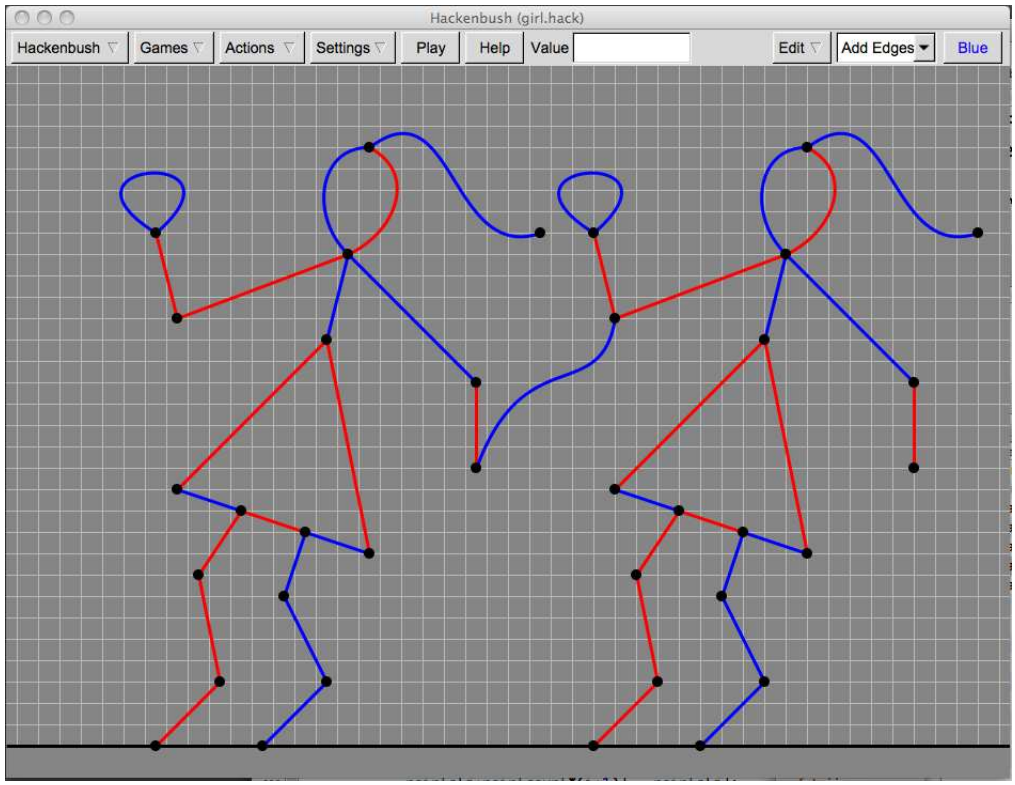
Also under the Settings pulldown, you can toggle whether you are in wizard mode or not. In wizard mode, the current surreal number that represents the current value of the game is displayed in the Value window. If it is positive, Blue has the advantage and can surely win with perfect play. If it is negative, the same is true for Red. If the game value is zero, then with perfect play, the player whose turn it is to move will lose.

Also, in wizard mode if it is your move and you make the cursor hover over one of your edges, the value of the game that would result if you cut that edge will be displayed next to the cursor. Thus in wizard mode you can "cheat" and play perfectly by hovering over each blue edge until you find the one that yields the largest positive number, or at least a zero.

(In finite games, as all the games in handled by **Hackenbush** are, all possible surreal numbers that can occur are positive, negative or zero rational numbers which have a denominator that is a power of 2.)

4.1 Complex Games

The **Hackenbush** program works as follows: when you start to play a game, either one that is built-in, one from a file, or one that you have entered yourself in editing mode, before it starts, it does a complete strategy analysis for the game. Almost all the built-in games are simple enough that this strategy calculation takes less than a second, but some positions are complex enough that a long time (and lots of computer memory) may be required. If the calculation is not complete after a short time, **Hackenbush** displays a message informing you of that, and updates that message approximately every second to show you how many positions it has analyzed. (Of course there's no indication of how many it will need to analyze for the complete game, but at least you can see that it is making progress.



In the figure of the hackenbush girl at the beginning of the article, a total of $10{,}447$ positions were analyzed to determine the complete strategy to play. But the figure above that consists of two copies of the girl "holding hands", for example, requires vastly more calculation to determine a complete strategy. As I'm tying this, I note that although the strategy calculation has not yet terminated, almost $80{,}000{,}000$ positions have been examined (and my computer, with 4 GB of memory, is grinding to a halt). This isn't surprising: if we just cut the connection between the hands there would be $10{,}447 \times 10{,}447$ positions to evaluate.

If you are interested in creating games that are difficult for a human to win (but that can be won), there is a better strategy than simply constructing incredibly complex interconnected games. See Section 10.

5 Manual Games

If you just want to use the program to play human versus human, load a game in the usual way (from a menu, file, or entering it manually) and then click on Play Manual Game under the Actions pulldown menu. Blue always starts and moves alternate until the end of the game. If you have a position where you'd like red to start, simply use the Swap Colors command in the Edit menu and red will have blue's position and vice-versa, so the blue player can effectively

always start.

Play in the usual way, where a single click on an edge of the appropriate color cuts that edge and all dependent edges. The name of the Play/Stop button will be changed to Manual and the text will alternate colors between red and blue to indicate whose move it is.

6 Multiple Straight Edges Connecting the Same Pair of Points

If multiple straight edges connect the same two points, the number of such edges is printed roughly in the middle of the edges that lie on top of each other. If some are red and some are blue, then the number of each type is listed with the blue number indicating the number of blue edges and the red number, the number of red edges. If there is more than one color, the edge is drawn with alternating red and blue dashes. If you're creating a game with multiple edges connecting a pair of points, it's probably best to make them curved so they don't lie on top of each other. See Section 8, below.

7 Creating or Editing Games

You can create any sort of game you like with any amount of advantage for either side, but since if you just flip the colors, the game becomes exactly the same thing for the opposite player, the convention **Hackenbush** uses is to always assume that a winnable position is winnable by Blue. It's easier to play these games, too, since by default, **Hackenbush** assumes that you, the human, are Blue and that it's your turn to move first.

If you are playing a game, you are in playing mode and cannot edit the position. To enter editing mode click on the Stop button (which may be labeled Manual if you are playing a manual game). If there is no Stop or Manual button but only a button labeled Play, then you are already in editing mode.

You can also use the Edit Game command under the Edit pulldown. This will load a position from a `.hack` file, but will not start playing it. This can be a huge advantage for games that require extensive calculations to determine the machine's proper strategy. (Generally such games have a lot of loops in them.) If you'd like to edit one of the built-in games, load it as if you were planning to play it and then click on the Stop button to enter editing mode.

You can edit the current position or clear the position with the Empty Game command in the Edit pulldown. Once you are satisfied with your editing, you can save the position as a `.hack` file using either the Save Game or the Save Game As command in the Edit pulldown menu.

When you are editing a position, you can be in various modes, selected using the Edit choice box:

- Select: Select things or drag points. The 'S' key is a shortcut to enter this mode. If an edge is selected, its color can be changed with the Red/Blue buttons or by typing the 'R' or 'B' key. If you use the Delete command in the Edit pull-down menu (shortcut: the 'D' key), the currently-selected edge or point will be deleted. You cannot delete a point with edges

connected to it. If you are in the mode where you can edit curves, then in select mode you can also click down on and drag the two control points for the line/curve.

- Add Points: In this mode every click makes a new point. If you click on the black line near the bottom of the window, you'll get a point on the ground. The 'P' key is a shortcut to enter this mode. Be sure that every edge is connected to the ground directly or indirectly when you create a hackenbush position or the position will be invalid. If you click on an existing edge in Add Points mode, you will have the option of splitting that edge into two with the new point between them, or just adding a point that is not part of the edge, or of canceling the addition.

- Add Edges: Click on two different points to create a straight edge connecting them. (If you wish to make a loop connecting a point to itself, see "Drawing and Editing Curves" in Section 8, below.) The 'E' key is a shortcut to enter this mode. The newly-created edges are of the color indicated by the Red/Blue button. Each time a new edge is created, it is selected (and everything else is de-selected). You can change the color of that selected edge by typing a 'R' or 'B' and you will remain in add-edges mode. Also, if you are in the mode where you can create curved edges then the newly-created edge will display two yellow control points, and you can drag them as long as the edge is selected.

- Set Color: In this mode, you can click repeatedly on edges to set them to the current color. The 'C' key is a shortcut to enter this mode.

The easiest way to draw a game is to click in the points, connect them with edges that are all the same color, and then change the color to enter the set-color mode. Then click on all those whose colors need to be changed.

Under the Edit menu, you can swap all the colors at once, if you wish. This is useful if you have a nice position but Red has the advantage and you'd like to create a game that Blue can always win.

As you edit a game, you can click on the Play button to pretend to play it. Look at the game value (in wizard mode) and then use the Stop button to stop it. You can tell how much of an advantage one player or the other has by looking at the game value. Usually, large positive values mean it's easy for Blue to win and large negative numbers mean the same for Red. But a number like $1/256$ means that Blue has a tiny advantage.

If the game is complex, with lots of loops, the calculation of the game value and machine's optimum strategy can take a LONG time. There is an Abort command in the Actions pulldown menu which will terminate the calculation. All editing information is lost, so if you suspect this may occur, save the game before you press the Play button. You can always reload it with the Edit Game command and then re-edit the position.

You can get some idea of the complexity of a game by using the Get Position Count command in the Edit pulldown menu. The number displayed is the number of positions the program needed to analyze to calculate its best strategy.

8 Drawing and Editing Curves

Under the Settings menu make sure that Edit Curves is checked. You can save this in your preferences (see Section 9) if you wish so you will always be able to edit edges to be curves. If curve editing is allowed, then when you are in Select mode and you select an edge or curve, not only is the selected item made darker and bold, but two additional control points are displayed in yellow. If you drag a control point, the edge/curve will be dragged appropriately. (For those of you who know a bit of math, the curves are cubic Bezier curves, and the two yellow dots are control points. They appear only during editing, not during game play.

So, to create a new curve connecting two existing points in Add Edges mode, create a straight edge connecting them, then click down on the yellow points and drag them. If you wish to change the shape of an existing curve, you must be in select mode, then select the edge, and then drag the control points. If you drag one of the endpoints of a curve, the two control points remain where they are, so you may have to reshape the curve by selecting it and dragging the control points appropriately.

If you are not in curve-editing mode, the two endpoints must be different. If you're allowed to edit curves, then clicking twice on the same point will create a Bezier loop at that point.

9 Preferences

Under the Hackenbush pulldown menu the Preferences option allows you to set a few **Hackenbush** preferences. Click on the boxes to turn them on or off and then save the resulting combination. Here are the preferences you can set:

- Wizard Mode: In this mode, the current value of the game is displayed in a Value box at top of the window. If it's positive, Blue has a certain win; if negative, then Red does. If it's zero, then with perfect play, the side whose move it is loses. Also in wizard mode if it's your move and you hover the cursor over an edge for a couple of seconds, the value of the resulting position if you remove that edge is displayed. This will show you how to play a perfect game from any position.
- Snap to Grid: If this item is checked, all new points and moved points will snap to a 20-pixel grid. The control points for (Bezier) curves are not so constrained.
- Skip Tutorial on Startup: Don't ask about running the tutorial when the game starts.
- Edit Curves: In this mode when you are editing a position, the control points for edges are displayed and can be dragged to make and modify curved edges.
- Stop Trivial Games: Use this mode when you're more comfortable with the game. Every game eventually gets to a point where no red edges can be eliminated by cutting any blue edge and similarly, no blue edge can be eliminated by cutting any red edge. At this point any idiot can play perfectly: If Blue has b edges and Red has r edges, then Blue has b moves left and Red, r moves. If $b - r$ is positive, Blue is certain to win. If it's negative, Red is sure to win. If it's zero, the player to move is certain to lose. In this mode, as

soon as a game reaches this trivial stage, either the computer resigns, or it claims the win, depending on the value of $b - r$ (and if it's zero, on whose move it is).

10 Creating Hard Games

A good way to create a difficult game is to compose it from a set of simpler games. One of the nicest features of hackenbush is that the value of a game composed of a number positions that do not interfere with each other is the sum of the values of those positions. This is pretty obvious if the positions consist of only blue or of only red edges. If Red can't cut anything that will eliminate a blue edge and Blue can't cut anything that will eliminate a red edge, then Blue has exactly as many moves available as there are blue edges and similarly for red. The advantage in such a game goes to the player with the most edges, so if you subtract the number of red edges from the number of blue edges, that number gives Blue's advantage (and if it's negative, Red's advantage).

What's nice is that the same is true for fractional game values. If we combine a position with value $-3/4$ (Red has an advantage and can always win this position if it stands alone) with one having value $+1$ (where Blue has a full move advantage), then the combination has value $-3/4 + 1 = +1/4$: a game Blue can always win. If, however, we add to that a third position having value $-1/2$ (a Red advantage), then the full game has value $-1/4$ and Red can always win.

When **Hackenbush** examines a game that is composed of positions that can be split, rather than analyze the entire game at once, it simply analyzes the individual positions and adds the resulting values. Thus rather than examining tens or hundreds of millions of sub-positions in the two-girl game illustrated in Section 4.1, if the two girls were not connected, the analysis would consist of something more like $2 \times 10,447$ positions.

Thus if you want to make a complex game that doesn't take the lifetime of the universe for the computer to analyze, one good strategy is to combine a bunch of difficult simple games, the sum of whose values gives a slight advantage for Blue (assuming you're trying to create a game that Blue can win).

10.1 Testing Colorings

Once you have a network of edges and points, **Hackenbush** provides a method to analyze possible colorings of this network to give you a rough idea of how difficult the resulting position is. Of course it's impossible to say how difficult a position is for a human to win, since every one of us thinks differently.

But **Hackenbush**'s strategy is not too bad, and it tends to eliminate colorings that are less interesting (difficult). The way it works is currently based on the following ideas:

- Given a position, **Hackenbush** counts the proportion of blue edges that are not connected to the ground either directly or via paths of other blue edges. The ones that are connected to the ground only via one or more red edges are vulnerable, so the more vulnerable edges there are, the tougher the position is. The ratio of vulnerable edges to total edges is a

fraction between 0 and 1. If the fraction is 0, the position is almost completely boring, since if there are n edges, Blue can make n moves. Of course this is *almost* completely boring: Imagine a position consisting of two blue edges connected to the ground, one of which has a red edge on top of it, and there is in addition one red edge connected to the ground. If it's Blue's move, he'd better cut the grounded edge with the red on top or he'll lose!

- Points with only one color of edge coming out from them are more boring than points supporting multi-colored edges. The ratio of multi-colored points to total points give a measure of this. (In its internal calculations, **Hackenbush** considers the ground to be a single point, so although your figure may seem to have a bunch of points on the ground, **Hackenbush** considers them all the same, and if at least one supports a red edge and another a blue, that single ground point will be considered to be multi-colored.)

- Perhaps most important is the proportion of ways Blue can screw up while playing. Again, this isn't a perfect solution, but usually during a game, the best initial plays are among the vulnerable edges so in for this calculation we count the number of edges which, if cut by Blue, result in a certain loss (where Red plays perfectly, as the machine does). This number, divided by the total number of edges, is again a measure of difficulty of a position.

The three ratios above are multiplied together to give an approximation of the difficulty of the coloring.

This difficulty is not absolute, especially among different networks, but when you wish to compare the difficulty of different colorings of a single network, the larger the product, the more interesting the position usually is.

10.2 Finding a Position's Difficulty

If you have a position colored in red and blue, you can use the command Test This Coloring under the Edit pulldown. The position will be analyzed as above, and the difficulty number will be presented to you. You can then manually modify the coloring and try again.

10.3 Looking for "Good" Colorings

Hackenbush can also take a position and try a lot of likely colorings on it, saving the ten best versions it finds. As the calculation proceeds, each time a better position is found than one of the top ten, the files are updated. If the position is complex and you don't have time (sometimes weeks) to wait for all the possibilities to be checked, you can interrupt the process at any time and at least you will have the top ten positions found so far.

To initiate this search, use the Find Good Colorings command in the Edit pulldown menu. You will be asked for a root file name and using the file browser you can navigate to the appropriate folder and add the root to that path. Suppose you want to put the files in the folder `/Users/fred/Desktop/hackfiles` and that you want all of them to have a root name of `foo`. Navigate to the correct folder (which you can create, if you need to, in the file browser) and

make sure that the name is `/Users/fred/Desktop/hackfiles/foo`. During the calculation, files will be created with names: `foo_0.hack`, `foo_1.hack`, ..., `foo_9.hack`. The first one, `foo_0.hack` has the best score (is rated most difficult), and the difficulties decrease from there, but if you've looked at a lot of colorings, number nine is probably pretty good, too.

In the header line of the files following the version information is an indication of the game value and the advantage Blue has in this particular situation. For example, the header:

```
Hackenbush(V.0000003)|0.356481|1/4|
```

indicates that the value of the game is 0.356481 and that Blue has an advantage of $1/4$.

One other important property of the positions that are generated in the files is that they are shifted almost as far to the left as possible (so that all the points and curve control points can still be selected and moved). This makes it much simpler to insert combinations of these generated patterns to make composed positions as we will discuss in the next section.

If you are making components of larger positions, then having complex positions with game value 0 is useful, since they can be mixed in with others to make games that are more complex. If you are simply looking for good positions for a particular network then you probably only want games with a non-zero value. Under the Settings pulldown you have the option of looking only for non-zero games. (By default, the search will include up to half of the games with zero value.)

10.4 Composing Positions

A nice way to compose complex games is to combine simpler ones. You can repeat a network a few times (2, 3 or 4, say) with different colorings so that the net game has a slight Blue advantage (assuming that's your goal). If we just combined games generated by the technique above, however, all will have a Blue advantage, so combining a number of them will give Blue an even larger advantage. But **Hackenbush** allows you to combine components either as they are (with a Blue advantage) or with all the colors flipped (with the result having a Red advantage).

So for example, if you have some components with values, say, $1/1$, $1/2$, $1/2$ and $1/4$, you might compose a position with the first and last inserted normally and the middle two inserted with the colors flipped. This will yield a combined game having a value of:

$$1/1 - 1/2 - 1/2 + 1/4 = +1/4 :$$

a game with a slight Blue advantage but consisting of positions that are at least somewhat complicated.

There may be a slight problem with this, of course: the games generated using techniques in the previous section are only supposed to be tough for Blue, so there's no guarantee that they are hard for Red to play, and that would mean that with the colors flipped they might be easy for Blue. One way around this (although it may be a little ugly aesthetically) is, for example, to combine all four games as-is (giving Blue an advantage of $2\ 1/4$ and then adding a two-edge stalk of red to the ground, giving Blue an overall edge of $1/4$, but that can only be achieved by excellent play on all four sub-positions.

The commands necessary to compose positions are these, all found under the Edit pulldown menu:

- New Empty Game: Clear the window, deleting all existing points and edges and putting **Hackenbush** into editing mode.
- Edit Game: Loads an existing game from a file into **Hackenbush** but does not start to play it. After the position is loaded you are still in editing mode. (If, for some reason, you want to edit one of the built-in games, load it from the Games menu and before you start to play, click on the Stop button to get into editing mode.
- Copy Game: This duplicates exactly what's on the screen, shifted left or right by an amount in pixels you specify. Points and control points cannot be shifted off the window.
- Shift Game: Shifts the picture on the screen left or right by the number of pixels you specify (using negative or positive shifts, respectively). You cannot shift points or control points off the visible screen.
- Insert Game: Inserts an additional game from a file and adds it to the position. You can manually enter a shift value in pixels, but if you just leave in the default value of "tight" your inserted data will be shifted so that it is 20 pixels to the right of the rightmost point or control point that is already on the screen.
- Insert Game Inverted: This is the same as Insert Game except that the inserted game will have all it's colors switched.
- Save Game and Save Game As: These commands save the current position of a `.hack` file.

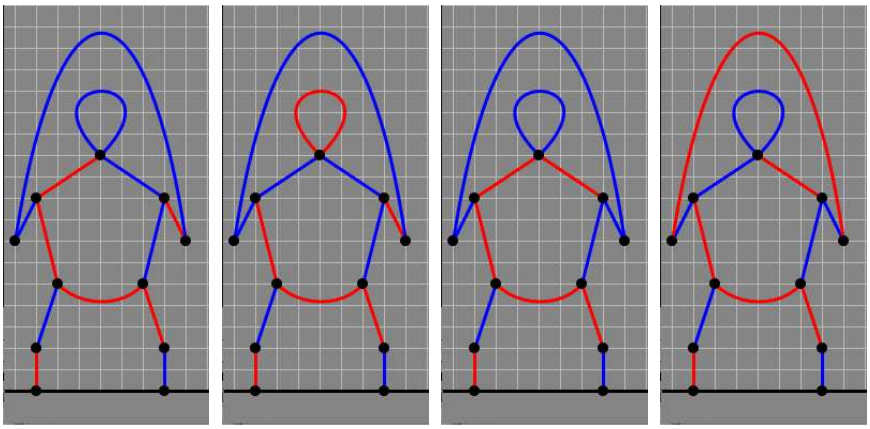
10.5 One Possible Strategy

My friends Greg Whitehead and Thane Plambeck have written a wonderful version of this program that runs on the iPad and can be loaded for free from the Apple store. Their version is called "HAKENBUSH" (no "C"), and it contains lots of good games. All of them come in series, where all the games in a series share a theme. For example, all the items may look like sea creatures or all like molecules, or something similar. If you wanted to do something like their sea-creature game, you could make a few relatively simple networks that all look like sea creatures, then search for good colorings of each, and then combine them to make games: maybe three sea stars or maybe a shark, some seaweed and a crab.

It does seem more fun to have games that have some pattern.

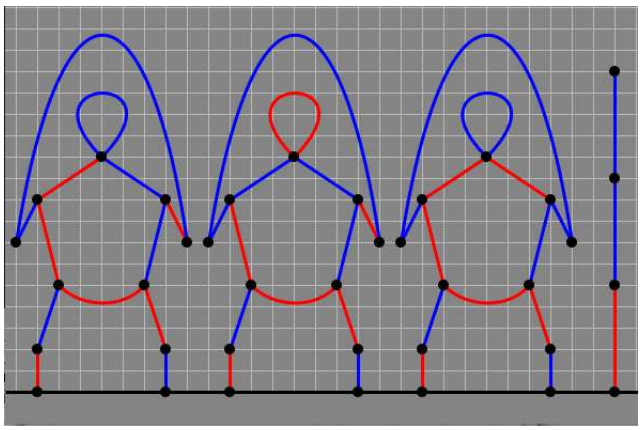
11 Example

I created a drawing of a person jumping rope, all in blue edges. I then searched for the best possible colorings, and obtained the four following versions, all having surreal game value of $1/8$:



Since all are worth $1/8$ I figured I could use the first three of them to make a game valued at $3/8$ and then add a simple three-edge stalk worth $-1/4$ to make the full game have value $1/4$, but it would be tough for Blue to win, since it would require good play on three difficult jump ropers.

Here's the result:



The latest version of **Hackenbush** also searches for up to 5 games with value zero, and those can also be included.

12 Limits

Currently **Hackenbush** is limited to games with 64 or fewer edges.

When you search for difficult colorings of a network, the network must have fewer than 25 edges or only a portion of the possible colorings will be tested. The searches are limited to combinations that have roughly the same number of red and blue edges. (By "roughly the same number" is meant at most off by three. For an odd number, like 25, positions with $12:13$ and $11:14$ red:blue or blue:red are checked. For an even number, like 24, combinations checked will be $12:12$ and $11:13$.) In reality, on large networks, you'll only have enough computer time to examine a few positions, since each coloring check requires a complete game analysis and

sometimes two. A full check of a position with 25 edges would require the analysis of $9{,}657{,}700$ positions.

It's not a true limit, but the analysis of large games with lots of loops can take an immense amount to time, and the only way to interrupt the analysis usually loses the position. If you are experimenting with complex positions, be sure to save it as a file before you attempt to Play it, so if you need to Abort, at least you can start where you were. On a position such as this, if it takes forever and requires an Abort, use the Edit Game command in the Edit pulldown to reload it to simplify it for further analysis. (If you try to load it using the Games menu, **Hackenbush** will begin the position analysis as soon as it has read the file.)

12.1 Keyboard Shortcuts During Editing

- A: Enter add-points mode. Each click adds a new point to the network. If you click on an edge (straight or curved) you have the option of breaking the edge at that point and inserting the new point to connect the pieces.
- B: Sets the currently-selected edge to blue.
- C: Enter repeat-set-color mode. In this mode, each time you click on an edge its color will be changed to the current color, indicated by the button at the top right of the screen.
- D: Deletes the currently-selected edge or free point, if there is one. A free point is a point with no edges connected to it.
- E: Enter add-edges mode. Click on two points to create an edge between them. Click on a single point twice (if you're allowed to create curves: see Section 8) to create a loop from this point to itself.
- R: Sets the currently-selected edge to red.
- S: Enter select mode: points and control points can be selected and dragged. Edges can be selected (making their control points visible).

13 File Format:

Except for the first line, the file is entirely integers, spaces and newlines. The file is a pure (ASCII) text file. Here's the format (ugly but simple):

Line 1: Hackenbush(V.0000003)[other stuff] Line 2: P = number of points Next P lines: x-coordinate y-coordinate isground Line P+3: L = number of edges Next L lines: point-1 x1 y1 x2 y2 point-2 color

The first line may have some additional information on it, indicated by [other stuff]. Right now the only useful [other stuff] that appears is an indication of the difficulty of the game and the surreal value of the game. For example the header:

```
Hackenbush(V.0000003)|0.356481|1/4|
```

indicates that the value of the game is 0.356481 and that Blue has an advantage of $1/4$.

The index of the first point in the list is 0, the last, P-1.

isground is 0 (off ground) or 1 (on ground)

point-1 and point-2 are point indices from 0 to P-1

If the edge is straight, x1, y1, x2 and y2 are -1.

If the edge is a curve, then (x1, y1) and (x2, y2) are the pair of control points between points 1 and 2 that describe a cubic Bezier spline. for color, 1=blue, 2=red

Valid coordinates are $0 < x < 940$, $0 < y \leq 636$

x goes from left to right, y from top to bottom. Grounded points MUST have a y-coordinate of 636.

Following is a sample file for a "lollypop". It consists of three points: point 0 on the ground, point 1 directly above it and point 2 directly above point 1. There is a straight red edge connecting points 0 and 1, then a slightly-curved blue edge connecting points 1 and 2, and finally a red loop connecting point 2 to itself:

```
Hackenbush(V.0000003)
3
430 636 1
430 536 0
430 436 0
3
0 1 -1 -1 -1 -1 2
1 2 402 504 448 471 1
2 2 272 293 597 295 2
```

You can paste the text between the lines of hyphens into a file called `lollypop.hack` and view the result in the program. You can also save any game you wish to a `.hack` file and view the results in a text editor.

The format is simple enough that you could also obviously generate suitable for use in **Hackenbush** games using a computer program.

14 Bibliography

- Conway, J. H. (1976), *On Numbers and Games.* Academic Press, London, New York, San Francisco.
  (Hackenbush is just a tiny part of this, and it also contains chapters introducing the surreal numbers.)
- Elwyn R. Berlekamp, John H. Conway, R. K. Guy (1982). *Winning Ways, Volume 1: Games in General.* Academic Press, London, New York, Paris, San Diego, San Francisco, São Paulo, Sydney, Tokyo, Toronto.
  (A very interesting book, covering lots of games, various versions of hackenbush included.)

- Knuth, D. E. (1974). *Surreal Numbers.* Addison-Wesley, Reading, Massachusetts, Menlo Park, California, London, Amsterdam, Don Mills, Ontario, Sydney.
  (A gentle introduction to surreal numbers. Lots of fun.)
- Davis, T. (2011). *Hackenbush.*
  `http://www.geometer.org/mathcircles/hackenbush.pdf`
  (The mathematics of hackenbush in particular.)
- Davis, T. (2012). *The **Hackenbush** User's Guide*.
  `http://www.geometer.org/Hackenbush/hackdoc.pdf`
  (Latest web version of this document.)