

COMBINATORIAL GAME BOARDS WITH TIKZ

SIMON RUBINSTEIN-SALZEDO

ABSTRACT. The `combgames-tikz` package draws combinatorial game boards using TikZ. It replaces the PostScript boards of `combgames` and adds further games and color variants. Symbols and game notation come from the separate `combgames-symbols` package, which is loaded automatically.

CONTENTS

1. Installation and a first example	2
2. Game Boards	2
2.1. Reading and sizing boards	2
2.2. Color versions	2
2.3. Domineering	3
2.4. Clobber and AntiClobber	3
2.5. Dots and Boxes	3
2.6. Strings and Coins	4
2.7. Chomp	4
2.8. Bridg-It	4
2.9. Amazons	4
2.10. Fox and Geese	5
2.11. Push, Shove, and their all-small versions	5
2.12. Toppling Dominoes	6
2.13. Hackenbush strings	6
2.14. General Hackenbush diagrams	6
2.15. Hex and boundary components	7
2.16. Maze and Maize	8
2.17. Christmas Lights' Fixture	9
2.18. Minesweeper	9
2.19. Toads and Frogs	10
2.20. Ski Jumps	10
3. Boards with named points and coordinates	10
3.1. Hare and Hounds	10
3.2. Nine Men's Morris	11
3.3. Phutball	11
4. Defining and customizing tokens	12
5. Compatibility with <code>combgames</code>	13
6. Troubleshooting	14

1. INSTALLATION AND A FIRST EXAMPLE

Install `combgames-symbols` as well as this package. For a local installation or an Overleaf project, place `combgames-tikz.sty` beside your document and add the files listed in the installation section of the `combgames-symbols` manual. All examples compile with pdfLaTeX, XeLaTeX, or LuaLaTeX.

```
\documentclass{article}
\usepackage{combgames-tikz}
\begin{document}
A game: $\combgame{\{1|0\}}$.
\ToadsAndFrogs{TTT.FFF}
\end{document}
```

The development versions used the name `cgt-board`. Replace it by `combgames-tikz`; board commands retain their names. For symbols and braces-and-slashes notation, see the `combgames-symbols` manual.

2. GAME BOARDS

There is also an extensible facility for typesetting grid-based games. To draw game boards, you must include the package `combgames-tikz`. The following examples illustrate usage for all of the built-in games.

2.1. Reading and sizing boards. Most grid commands take an optional drawing unit followed by the rows:

```
\GameName[.5cm]{rows}
```

The optional argument is a length, not a key such as `unit=.5cm`. Rows run from top to bottom and cells from left to right; separate rows with `\\`. Spaces and completely empty rows are ignored, including a final row separator. To leave a gap, use the game's explicit empty or missing-cell token. A dot does *not* have the same meaning in every game: the token descriptions below distinguish an empty square from a missing square. Unequal row lengths are allowed.

The length sets the coordinate unit. It is a square's side for square boards, a point-spacing unit for intersection boards, and a hexagon's width for HEX. Text inside cells does not necessarily scale with this length. The board commands can be used in text or mathematics; they are vertically centered on the math axis. The package draws positions, but does not check the rules or calculate moves.

2.2. Color versions. The following commands have a corresponding `Color` version with the same arguments:

```
\Push, \Shove, \AllSmallPush, \AllSmallShove, \TopplingDominoes,
\HackenbushString, \Hex, \ToadsAndFrogs, \SkiJumps, \HareAndHounds,
\NineMensMorris, \Phutball and \Minesweeper.
```

For example, use `\ToadsAndFrogsColor`. Color normally identifies Left with blue and Right with red. In PHUTBALL it colors the move route instead; the stones remain black and white. `\Minesweeper` is monochrome by default. `\MinesweeperColor` uses colored clues and a red flag. CHRISTMAS LIGHTS use their explicitly chosen colors.

The shared switch `\gridgamecolortrue` also enables color in PUSH/SHOVE, HACKENBUSH STRINGS, HEX, TOADS AND FROGS, SKI JUMPS, HARE AND HOUNDS, NINE MEN’S MORRIS, PHUTBALL, and MINESWEEPER; `\gridgamecolorfalse` reverses it. Use a group to keep such a setting local. The `Color` wrappers do this automatically. TOPPLING DOMINOES uses a separate color token map, so use its `Color` command.

2.3. Domineering. `\Domineering` defaults to `.35cm`. Use `o` for an unoccupied square, `x` for a shaded square, and `.` for a missing square. A horizontal domino is `<>`; a vertical domino has `n` in its upper cell and `v` immediately below. The upper-half token `^` is also supported, but `n` avoids TeX’s special treatment of consecutive carets and works reliably in arguments already read by another command. If using two carets, separate them with a space: `^ ^`.

```
\Domineering{ooo\oo\} = \combgame{
  \{\Domineering{noo\vo}, \Domineering{ono\ov}
  | \Domineering{<o\oo}, \Domineering{o<\oo},
  \Domineering{ooo\<>}\}}
```

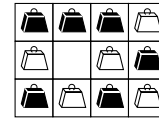


2.4. Clobber and AntiClobber. The original small and large CLOBBER commands are retained: `\clobber` defaults to `.3cm`, and `\Clobber` to `.5cm`. The tokens `X`, `x`, `L`, and `l` draw dark pieces; `O`, `o`, `R`, and `r` draw light pieces. A dot is an empty square.

```
\clobber{xxxo\o.ox\oxox}
```



```
\Clobber{xxxo\o.ox\oxox}
```



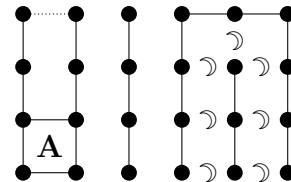
`\AntiClobber` uses the same tokens with inverted pieces, and defaults to `.3cm`. Its lowercase alias is `\anticlobber`.

```
\AntiClobber[.5cm]{XOXO\OXOX\XOXO}
```



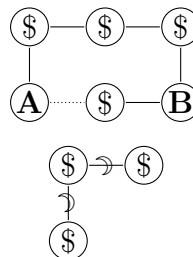
2.5. Dots and Boxes. `\DotsAndBoxes` defaults to `.35cm`. Dots and connecting edges occupy alternating positions in the input: `o` is a dot, `-` and `|` are solid edges, `+` and `:` are dotted horizontal and vertical edges, and `.` leaves a position blank. `A` and `B` label boxes; `*` adds a loony symbol. The following preserves the original annotated example.

```
\DotsAndBoxes{
  o+o.o.o-o-o\
  |.|.|.|.*.|\\
  o.o.o.o*o*o\
  |.|.|.|.|\\
  o-o.o.o*o*o\
  |A|.|.|.|.|\\
  o-o.o.o*o*o}
```



2.6. **Strings and Coins.** `\StringsAndCoins` defaults to `.35cm`. A coin is `o`, or `A/B` for a labeled coin. Use `-` and `|` for strings, `+` and `:` for dotted strings, and `.` for a blank position. The annotated string tokens `*` (horizontal) and `!` (vertical) include a loony mark. Coins and strings occupy alternating positions.

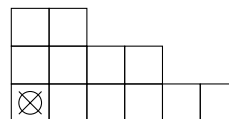
```
\StringsAndCoins[.5cm]{
  o-o-o\\
  |...|\\
  A+o-B}
```



```
\StringsAndCoins[.5cm]{o*o\\!...\\o..}
```

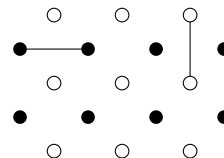
2.7. **Chomp.** `\Chomp` defaults to `.35cm`. Use `0` for a square, `P` (also `p` or `X`) for the poisoned square, and `.` for an absent square. Here the poison is at bottom left; the upper-right part of the chocolate bar has been eaten.

```
\Chomp[.5cm]{00...\\0000...\\P00000}
```



2.8. **Bridg-It.** `\Bridgit` defaults to `.35cm`. The two sets of intersections are `x` (filled) and `o` (open); `-` and `|` draw bridges. Dotted bridges are `+` and `:`; a dot leaves a position blank. The interleaved arrangement below includes a horizontal bridge for one player and a vertical bridge for the other.

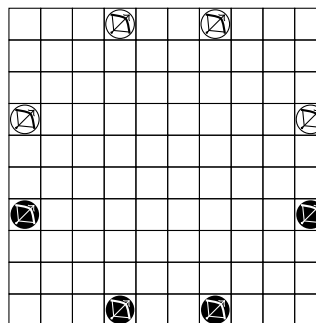
```
\Bridgit[.45cm]{
  .o.o.o.\\
  x-x.x|x\\
  .o.o.o.\\
  x.x.x.x\\
  .o.o.o.}
```



2.9. **Amazons.** `\Amazons` defaults to `.35cm`. `L` and `R` are the two players' amazons (lower-case works too), `o` is an empty *square*, and `x` is a blocked square containing an arrow. A dot omits a square entirely: it should not be used for the open cells of a full board.

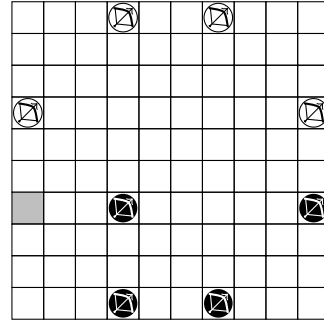
This is the usual 10-by-10 starting arrangement, with four amazons for each player. The top row is rank 10; columns run from a to j. Here Left occupies a4, d1, g1, j4, and Right occupies a7, d10, g10, j7.

```
\Amazons[.42cm]{
  oooRooRooo\\
  oooooooooo\\
  oooooooooo\\
  RooooooooR\\
  oooooooooo\\
  oooooooooo\\
  LooooooooL\\
  oooooooooo\\
  oooooooooo\\
  oooLooLooo}
```



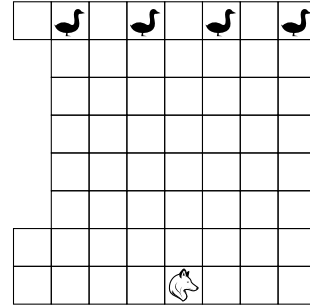
For a realistic blocked position, move Left's a4 amazon to d4 and shoot an arrow back to a4. Every other piece remains in its starting square.

```
\Amazons[.42cm]{
oooRooRooo\\
oooooooooooo\\
oooooooooooo\\
RoooooooooR\\
oooooooooooo\\
oooooooooooo\\
xooLoooooL\\
oooooooooooo\\
oooooooooooo\\
oooLooLooo}
```



2.10. **Fox and Geese.** `\FoxAndGeese` defaults to `.5cm`. Use `f` for the fox, `g` for a goose, `o` for an empty square, and `.` for a missing square. Uppercase `F`, `G`, and `O` are aliases.

```
\FoxAndGeese{
ogogogog\\
.ooooooo\\
.ooooooo\\
.ooooooo\\
.ooooooo\\
.ooooooo\\
.ooooooo\\
ooooooo\\
ooofoooo}
```



2.11. **Push, Shove, and their all-small versions.** `\Push`, `\Shove`, `\AllSmallPush`, and `\AllSmallShove` all default to `.5cm`. They use `L/X` for Left, `R/O` for Right, and `G` for a neutral piece; lowercase aliases work. The all-small versions use smaller piece artwork. These examples show both the monochrome and color forms.

```
\Push{LLRGLR}\quad\PushColor{LLRGLR}
```



```
\Shove{LRLLLG}\quad\ShoveColor{LRLLLG}
```



```
\AllSmallPush{LLRGLR}\quad\AllSmallPushColor{LLRGLR}
```



```
\AllSmallShove{LRLLLG}\quad\AllSmallShoveColor{LRLLLG}
```



In `PUSH`, `SHOVE`, and `ALL SMALL SHOVE`, a dot draws an empty track cell; in `ALL SMALL PUSH` it leaves a blank position. A literal `#` leaves a blank position in all four maps.

2.12. Toppling Dominoes. `\TopplingDominoes` and `\TopplingDominoesColor` default to .5cm. Use `x/X/L/l` for Left's domino, `o/O/R/r` for Right's domino, `g/G` for a neutral domino, and `.` for an empty ground position.

`\TopplingDominoes{xxogox}\quad`
`\TopplingDominoesColor{xxogox}`



2.13. Hackenbush strings. `\HackenbushString` defaults to .5cm. Each row is a horizontal string attached to ground at the left. Use `B` (or `L`) for blue, `R` for red, and `G` for green; lowercase aliases work. A dot leaves a gap. In monochrome, blue is solid, red is double, and green is gray. The spelling `\Hackenbushstring` is also accepted, including its `Color` form.

`\HackenbushString{BBR\RBG}\quad`
`\HackenbushStringColor{BBR\RBG}`



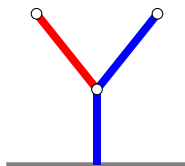
2.14. General Hackenbush diagrams. For more complicated HACKENBUSH positions and positions with cycles, use `TikZ` directly. The package supplies two styles: `hackenline` draws edges with a default thickness of 3pt, and `hackennode` draws white-filled, outlined circular vertices. Draw the edges first and the vertices afterward.

The following position has a blue stalk with red and blue branches. The gray horizontal line represents the ground.

```
\begin{tikzpicture}
  \coordinate (root) at (0,0);
  \coordinate (fork) at (0,1);
  \coordinate (left) at (-.8,2);
  \coordinate (right) at (.8,2);

  \draw[gray,line width=2pt] (-1.2,0)--(1.2,0);
  \draw[hackenline,blue] (root)--(fork);
  \draw[hackenline,red] (fork)--(left);
  \draw[hackenline,blue] (fork)--(right);

  \node[hackennode] at (fork) {};
  \node[hackennode] at (left) {};
  \node[hackennode] at (right) {};
\end{tikzpicture}
```



Cycles can be drawn in the same way. Here a green stalk supports a triangle with blue, red, and green edges.

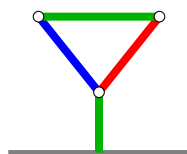
```

\begin{tikzpicture}
  \coordinate (root) at (0,0);
  \coordinate (base) at (0,.8);
  \coordinate (left) at (-.8,1.8);
  \coordinate (right) at (.8,1.8);

  \draw[gray,line width=2pt] (-1.2,0)--(1.2,0);
  \draw[hackenline,green!70!black] (root)--(base);
  \draw[hackenline,blue] (base)--(left);
  \draw[hackenline,red] (base)--(right);
  \draw[hackenline,green!70!black] (left)--(right);

  \node[hackennode] at (base) {};
  \node[hackennode] at (left) {};
  \node[hackennode] at (right) {};
\end{tikzpicture}

```



These diagrams use explicit TikZ colors. The shared board-color switch does not affect them. Coordinates, ground lines, and edge connections are supplied by the author.

To change the appearance locally, add options to the `tikzpicture`, for example:

```

\begin{tikzpicture}[
  hackenline/.append style={line width=1.5pt},
  hackennode/.append style={minimum size=3pt}]
  ...
\end{tikzpicture}

```

2.15. Hex and boundary components. `\Hex` and `\HexColor` have syntax

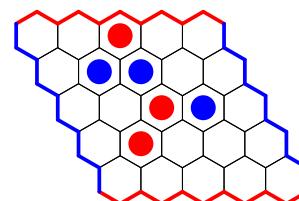
```
\Hex[unit][options]{rows}
```

with default unit `.5cm`. Use `o` (or `*`) for an empty hexagon, `X` for a Left stone with its cell outline, and `O` for a Right stone with its cell outline. `L` and `R` draw just the stones. In the color version these are blue and red. Aliases are `x` for `X`, and `l/r` for `L/R`. A dot omits a cell. Lowercase `o` always means an empty hexagon, not a Right stone.

```

\HexColor[.55cm][boundary=standard]{
  ooOoo\\
  oXXoo\\
  ooOXo\\
  oOooo\\
  ooooo}

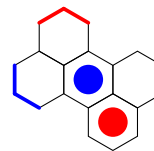
```



`boundary=standard` colors the top and bottom boundary red and the left and right boundary blue. `boundary=swapped` reverses them; `boundary=none` is the default. The shortcut requires equally long rows with no omitted cells.

For fragments, specify individual edges instead. An entry has the form `row/column/direction/color`, with row and column counted from 1 at the *top left of the input*. Directions are NE,E,SE,SW,W,NW; colors are `blue` and `red`. Edges must refer to existing cells. This marks only the selected boundary segments, rather than treating the fragment as an entire board.

```
\HexColor[.65cm][edges={
  1/2/NW/red,1/2/NE/red,
  2/1/W/blue,2/1/SW/blue}]{
  .o.\
  oXo.\
  .0.}
```



Explicit edges are drawn after the full-board shortcut, so they can also override selected edges. Boundary colors are explicit annotations and remain colored even with `\Hex`.

2.16. Maze and Maize. `\Maze` and `\Maize` default to `.3cm`. Cells are lightly shaded gray, with white seams separating neighboring cells where there is no wall. Actual walls are drawn in black.

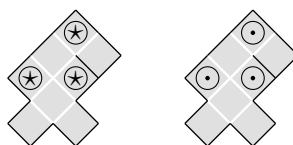
Both commands use the same wall encoding. Around a cell's diamond, assign weights 1 to left-top, 2 to top-right, 4 to right-bottom, and 8 to bottom-left. Add the selected weights and write the result as one hexadecimal digit 0-9, a-f. Uppercase A-P selects the same sixteen wall combinations *with a center marker*, in order. MAZE's marker is a circled star; MAIZE's is a circled dot.

For example, 5 selects the walls of weights 1 and 4, and F selects those same walls with a center marker. A shared wall may be specified by either neighboring cell; it need not be specified by both.

The diamond cells occupy alternating positions, like one color of a checkerboard. Use `.` for intervening positions and locations outside the board. Use 0 or o for an actual shaded cell with no walls or marker. Thus, dots and zeros are no longer interchangeable.

```
\Maze{..D\\.1.e\\J.E\\.0..\\d.e}\quad
```

```
\Maize{..D\\.1.e\\J.E\\.0..\\d.e}
```



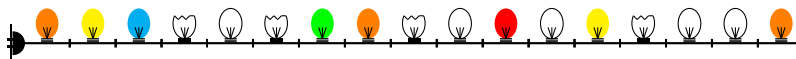
The tokens `*` and `x` give a shaded cell with a center marker and no walls. The tokens `/` and `|` select the southwest/northeast and northwest/southeast pairs of walls, respectively; they are equivalent to `a` and `5`.

The following atlas separates the sixteen wall patterns. Each pair of token names selects the plain walls or the corresponding marked cell. The final column in each group shows the marked Maize version.

Tokens	Walls	Maze	Maize	Tokens	Walls	Maze	Maize
0/A				8/I			
1/B				9/J			
2/C				a/K			
3/D				b/L			
4/E				c/M			
5/F				d/N			
6/G				e/O			
7/H				f/P			

2.17. Christmas Lights' Fixture. `\Xmas` takes an optional drawing unit (default `.35cm`) followed by a string of fixture characters. It does not use row separators. Letter case is ignored. The bulbs are C (cyan), G (green), O (orange), R (red), Y (yellow), W (unfilled), and B (broken). M draws a short gray rail with bars at both ends; N has a bar only at the right end. P draws a plug, normally placed at the beginning of the string. Unrecognized characters are skipped.

`\Xmas[.1cm]{POYCBWBGOWBRWYBWWO}`

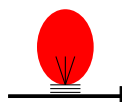


`\Xmas[.1cm]{PCOY}\quad\Xmas[.1cm]{OMBNR}`



Individual commands `\XmasB`, `\XmasC`, `\XmasG`, `\XmasM`, `\XmasN`, `\XmasO`, `\XmasP`, `\XmasR`, `\XmasW`, and `\XmasY` are available. `\UseXmas{R}` invokes the corresponding uppercase command. The individual commands use the default unit. For a single fixture at another size, use either `\Xmas[.25cm]{R}` or the single-token interface:

`\TokenBox[.25cm]{XMAS}{R}`



2.18. Minesweeper. `\Minesweeper` defaults to `.5cm`. A dot draws an unlabelled square; 0-8 draw clues; f is a flag. Tokens a, b, c print variables, while A, B, C print their barred forms.

```
\Minesweeper{
.....\\
.3110\\
.2f10\\
.1110\\
.aA00}
```

	3	1	1	0
	2	▶	1	0
	1	1	1	0
	a	$\bar{a}$	0	0

```
\MinesweeperColor{
.....\\
..ff..\\
112211\\
000000}
```

1	1	2	2	1	1
0	0	0	0	0	0

2.19. **Toads and Frogs.** `\ToadsAndFrogs` defaults to .6cm. T is a dark toad facing right, and F a light frog facing left. L and R are aliases; all four also accept lowercase. A dot or o is an empty square. Color versions use blue toads and red frogs.

```
\ToadsAndFrogs{TTT.FFF}\quad
```

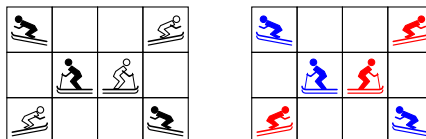
```
\ToadsAndFrogsColor{TTT.FFF}
```



2.20. **Ski Jumps.** `\SkiJumps` defaults to .6cm. L and R are jumpers, while l and r are slippers. Uppercase and lowercase therefore have different meanings here. Left's figures face left; Right's face right. A dot or o is an empty square. This diagram illustrates all four piece states.

```
\SkiJumps{L..R\\..lr.\\R..L}\quad
```

```
\SkiJumpsColor{L..R\\..lr.\\R..L}
```



3. BOARDS WITH NAMED POINTS AND COORDINATES

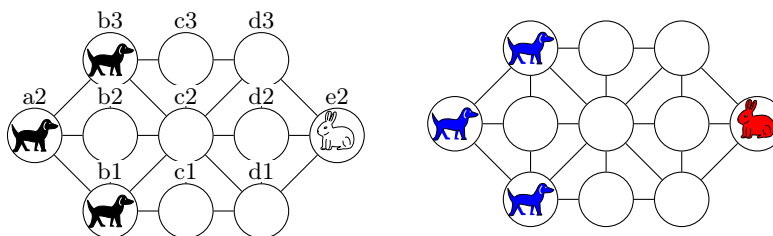
These commands draw their own board geometry. They take *two* mandatory arguments, even for an empty board. Supply {} for an empty list. The general form is `\Game[unit][options]{first list}{second list}`. To supply options, also supply the first optional argument. Every command in this section has a `Color` version with identical syntax.

3.1. **Hare and Hounds.** `\HareAndHounds` defaults to 1cm. The first list gives hound positions; the second gives the hare position. Positions are named a2, b1-b3, c1-c3, d1-d3, and e2; numbers increase upward. Use `labels` (or `labels=true`) to print the point names. The initial position from the manuscript is:

```
\HareAndHounds[1cm][labels]{a2,b1,b3}{e2}
```

```
\quad
```

```
\HareAndHoundsColor[1cm]{a2,b1,b3}{e2}
```



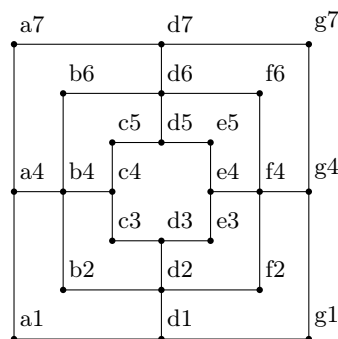
The board connections are built in. For a blank reference board, use:

`\HareAndHounds[1cm][labels]{}`

The hound and rabbit figurines are TikZ paths, so no external image files are needed. Unknown point names and duplicate occupied points produce errors.

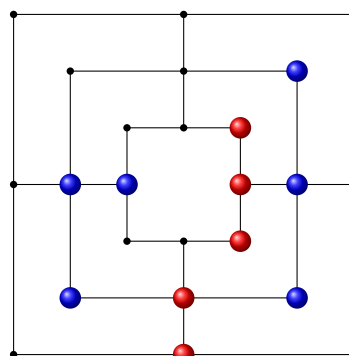
3.2. Nine Men's Morris. `\NineMensMorris` defaults to `.65cm`. The first list gives Left's pieces, the second Right's. Names follow a seven-by-seven coordinate frame: **a1** is bottom left and **g7** top right. Only the board's 24 marked points are valid; for example, **d4** is not a point. The `labels` option displays the names.

`\NineMensMorris[.65cm][labels]{}`



This position reproduces the manuscript's shaded blue and red balls.

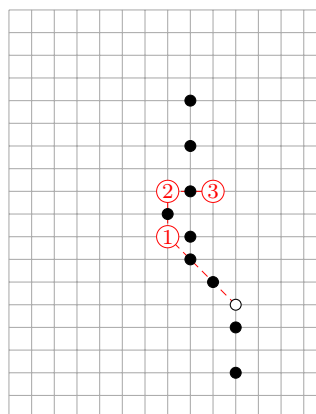
`\NineMensMorrisColor[.75cm]`
`{b2,f2,f6,f4,b4,c4}`
`{d1,d2,e3,e4,e5}`



Without `Color`, the balls are shaded black and white. The command draws the board and pieces; it does not track reserves, turns, mills, or captures.

3.3. Phutball. `\Phutball` defaults to `.35cm`. The first list gives black stones; the second gives the white ball (one coordinate, or an empty argument). Coordinates use **x/y**, with 1/1 at bottom left. The defaults are `columns=15,rows=19`. The example below is the manuscript's position, including its three-jump route.

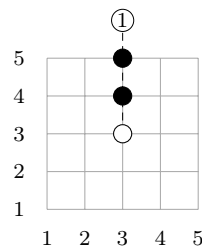
`\PhutballColor[.3cm]` [
`move={11/6,8/9,8/11,10/11}]`
`{11/3,11/5,10/7,9/8,9/9,`
`8/10,9/11,9/13,9/15}`
`{11/6}`



The move list starts at the ball and then lists each landing. A thin dashed route is drawn behind the stones so the stones remain intact. Landings are numbered by default; use `numbered=false` to suppress the numbers. `\PhutballColor` uses a red route, while `\Phutball` uses black. The key `move color=blue`, for example, overrides either default.

Use `labels` to display grid coordinates. A smaller board is useful for a local position, and move endpoints may lie beyond the grid to show a goal:

```
\Phutball[.5cm][columns=5,rows=5,
  labels,move={3/3,3/6}]
{3/4,3/5}{3/3}
```



Stone coordinates must be on the board, and occupied points cannot be duplicated. Move routes are annotations: the command does not validate jumps or remove the stones crossed. Supply the resulting position separately when illustrating a before-and-after move. Avoid repeated landing coordinates in one numbered route, since their labels would overlap.

4. DEFINING AND CUSTOMIZING TOKENS

The extensible grid facility now uses TikZ. A token definition has the form

```
\gridtoken{map}{character}{drawing code}
```

The drawing code runs in a local scope at that token's position. For a square board, draw inside the unit square from (0,0) to (1,1). Then call `\GridGame[unit]{map}{rows}`, or define a wrapper. Here is a complete example of a new token map:

```
\gridtoken{DEMO}{o}{\draw (0,0) rectangle (1,1);}
```

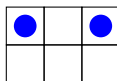
```
\gridtoken{DEMO}{L}{
```

```
\draw (0,0) rectangle (1,1);
```

```
\fill[blue] (.5,.5) circle[radius=.3];}
```

```
\gridtoken{DEMO}{.}{}
```

```
\GridGame[.5cm]{DEMO}{LoL\ooo}
```



For a reusable command, put the definitions in the preamble and add:

```
\NewDocumentCommand{\MyBoard}{0{.5cm} m}{
```

```
\GridGame[#1]{DEMO}{#2}}
```

Then `\MyBoard{LoL\ooo}` draws the same board. Redefining a token replaces its existing drawing, and definitions can be localized with a TeX group. Use one character per token. Unknown tokens in `\GridGame` generate a warning and leave that position blank.

Both `\GridGame` and `\TokenBox` default to a .35cm unit. `\TokenBox` draws a single registered token without the row parser:

```
\TokenBox[.6cm]{TAF}{T}
```



The built-in map names are shown here for customization.

Domineering	DomTok	Clobber	CLOB
AntiClobber	ACLOB	Dots and Boxes	DAB
Strings and Coins	SNC	Chomp	CHM
Bridg-It	BRG	Amazons	AMZ
Fox and Geese	FNG	Push / Shove	PUSH / SHOV
All-small Push / Shove	ASP / ASHV	Toppling Dominoes	TDOM / TDOMC
Hackenbush strings	HACK	Hex	HEX
Maze	MAZE	Maize	MAIZ
Toads and Frogs	TAF	Ski Jumps	SKI
Minesweeper	MINE	Christmas fixtures	XMAS

Color variants share the corresponding token map, except for TOPPLING DOMINOES, which uses TDOM for monochrome and TDOMC for color. Both MINESWEEPER variants use MINE.

\HareAndHounds, \NineMensMorris, and \Phutball use named board positions rather than token maps. Use their dedicated commands, including their Color variants.

CHRISTMAS LIGHTS' FIXTURES artwork spans more than a unit square. Use \Xmas for complete strings or \TokenBox for individual fixtures, rather than placing them in adjacent \GridGame cells.

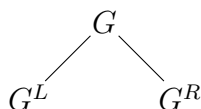
5. COMPATIBILITY WITH COMBGAMES

The original `combgames` manual, by Aaron N. Siegel and David Wolfe, provides the model for this manual's source-and-result examples. This rewrite retains the symbol and braces-and-slashes interfaces and supplies TikZ board renderers, including additional games and color variants.

It is not a drop-in implementation of every original facility. In particular, `\cgtree`, `thermoplot`, `\thermograph`, `\trajectory`, and the old `\newgridgame` interface are not implemented here. Nor are PSTricks key lists such as `unit=.5cm` accepted in place of a board length. Use `\gridtoken` and `\GridGame` for new boards. Loading `combgames` together with these packages may cause conflicting command definitions.

The `treenode` style provides unoutlined nodes for manually drawn game trees.

```
\begin{tikzpicture}
  \node[treenode] (g) at (0,1) {$G$};
  \node[treenode] (l) at (-1,0) {$G^L$};
  \node[treenode] (r) at (1,0) {$G^R$};
  \draw (g)--(l) (g)--(r);
\end{tikzpicture}
```



6. TROUBLESHOOTING

Adjacent vertical domineering pieces fail: Use `n` for the top half. TeX can interpret consecutive carets before the board parser sees them.

Literal bars fail in a macro or another argument: Inside `\combgame`, use the explicit slash commands described in the `combgames-symbols` manual.

Squares disappear: Check the board's token vocabulary. In `AMAZONS`, for example, `o` draws an empty square but `.` omits it. Blank input rows do not reserve vertical space.

An option or position is rejected: Check the command's argument order and coordinate convention. `HEX` edges count rows downward from the top; `PHUTBALL` coordinates count upward from the bottom. Named-point boards accept only the names shown by labels.

Email address: `simon@eulercircle.com`