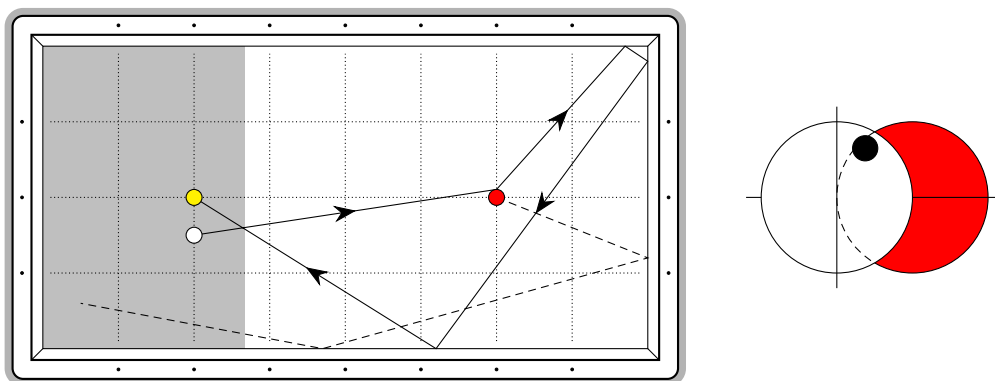


The carom-billiards package

Anthony Saint-Criq ([satplotlib](#))

July 7, 2026



1 Introduction

The [carom-billiards](#) package provides a new environment together with a collection of TikZ macros designed to facilitate the creation of French billiards table diagrams.

French billiards (also known as “carom billiards”) is played on a pocketless table with only three balls. The objective is to strike the cue ball (either white or yellow, one for each player) so that it subsequently contacts the other two balls. Successfully doing so scores a point and entitles the player to continue their turn until a point is no longer scored.

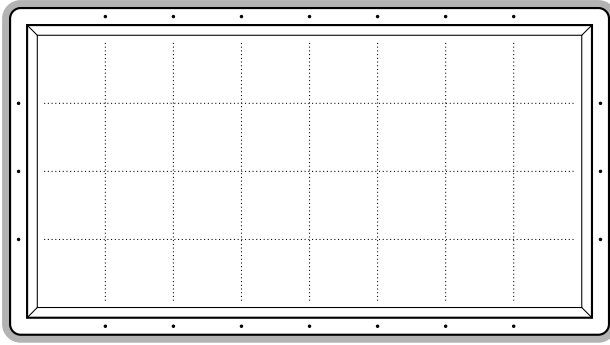
The cue ball may be struck off-center to impart spin (commonly referred to as an “effect”). Likewise, the point of contact on the object ball plays a significant role in determining the resulting trajectory.

This package provides tools for indicating ball positions on the table, as well as the intended point of impact on the cue ball and the desired aiming point on the object ball.

The author is grateful to the \TeX hackers from Mathcord: Valentin Dao ([xnkaa](#)) and [qigamma](#) for reviewing portions of the code, [γloud](#) for their guidance and explanations, and [xnkaa](#) again for assistance in sanitizing the source code.

2 The [bTable](#) Environment

The package defines a [bTable](#) environment, which provides the drawing area representing a French billiards table, and defines some commands available for placing and annotating objects on it.



```
\begin{bTable} [<key=value>]
  % inner code
\end{bTable}
```

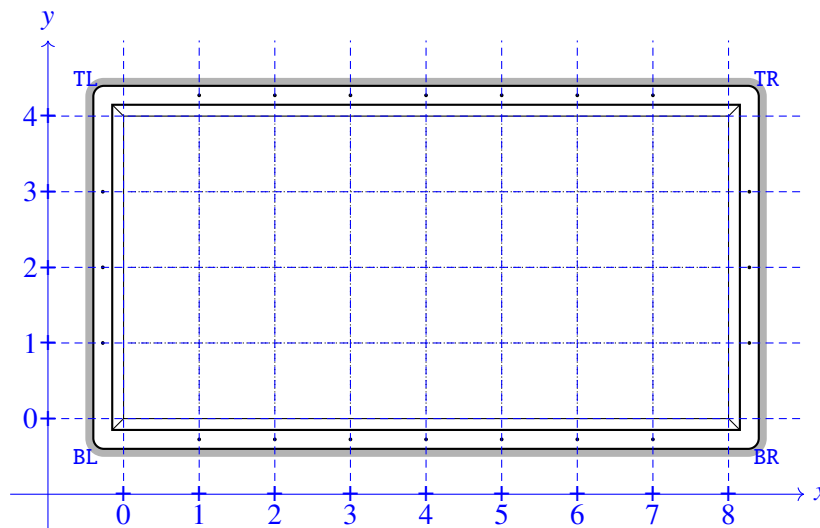
The optional argument consists of a comma-separated list of `key=value` pairs controlling various aspects of the diagram.

centering This key controls whether the table is horizontally centered with respect to the page layout. Its value defaults to `false`.

portrait This boolean key selects portrait orientation for the table. When set to `true`, the table is typeset in portrait mode. When set to `false` (default), the standard orientation is used.

scale This key is equivalent to TikZ's `scale` key and follows the same semantics. It sets a scaling factor applied to the entire table. The value must be a positive floating-point number. A value of 1 corresponds to the default size, while values greater than 1 enlarge the table and values between 0 and 1 reduce its size. Negative values are not supported.

guidelines This boolean key controls whether auxiliary construction guidelines are displayed on the table. When set to `true`, the guidelines used for construction and alignment purposes are drawn. When set to `false` (default), these guidelines are suppressed. These are *not* intended for printing.



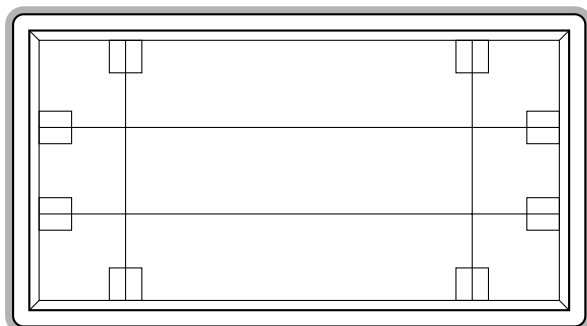
The guidelines exhibit a coordinate system, which will be described in the next section.

ballsize This key takes a dimension and defaults to `3pt`. It sets the radius of the balls on the table and updates the `\br` macro accordingly (see the next section). Note that the `scale` key *does* affect the ball size: `ballsize=3pt` does not produce a 6pt-wide element in the document unless `scale=1`.

gridlines This key toggles whether the dotted gridlines should be shown on the table. The default value is `true`.

cadre These are boolean keys, both defaulting to `false`, which control the display of some auxiliary markings on the table. The **cadre** key enables the display of the “lignes de cadre” (for the 42/2 game), while the **ancres** key enables the display of the “ancres” (for the 47/2 game). Typically, the ancres are not displayed without the cadre being enabled.

mouches This boolean key controls the display of the side markings (“mouches”) on the table. When set to `true` (default), the markings are drawn; when set to `false`, they are suppressed.



```
\begin{bTable}[gridlines=false,
cadre,ancres,mouches=false]
% inner code
\end{bTable}
```

The default value for each key is summarized in the following table. As is standard for boolean keys, the value may be omitted; in that case, specifying `[key]` is equivalent to `[key=true]`. The value `false` must, however, be stated explicitly, i.e., `[key=false]` cannot be abbreviated.

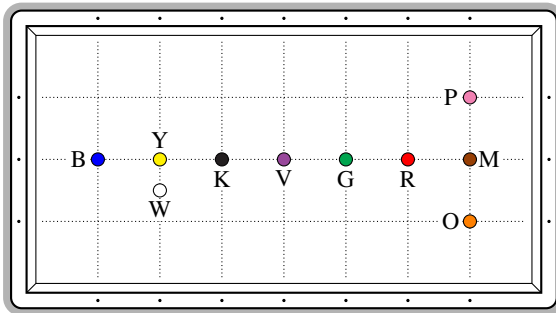
Key	Type	Default value
<code>centering</code>	Boolean	<code>false</code>
<code>portrait</code>	Boolean	<code>false</code>
<code>scale</code>	Float	<code>1.0</code>
<code>guidelines</code>	Boolean	<code>false</code>
<code>ballsize</code>	Dimension	<code>3pt</code>
<code>gridlines</code>	Boolean	<code>true</code>
<code>cadre</code>	Boolean	<code>false</code>
<code>ancres</code>	Boolean	<code>false</code>
<code>mouches</code>	Boolean	<code>true</code>

3 Commands Available Inside the `bTable` Environment

A French billiards table is a rectangle composed of two squares. To draw elements on the table, a coordinate system is used, which can be displayed using the `guidelines` key. Traditionally, the “mouches” are numbered from 1 to either 3 or 7, depending on the side of the table. This is reflected by using coordinates ranging from 0 to 8 along the x -axis, and from 0 to 4 along the y -axis.

The package defines many commands that are **scoped** to the `bTable` environment.

`\WBall` These commands respectively place the white, red, yellow, blue¹, green, black, orange, pink,
`\RBall` violet and maroon balls on the table at the specified coordinates.
`\YBall` Note that no warning is issued if a ball is placed outside the rectangle delimited by the coordinates
`\BBall` (0, 0) and (8, 4).
`\GBall`
`\KBall` It is possible not to use any of each ball's colour-specific macro, as well as to include the same
`\OBall` colour of ball multiple times (although this may have undesired consequences; see later).
`\PBall` These macros expect the coordinates of a point, followed by a semicolon (following standard
`\VBall` TikZ conventions).
`\MBall`



```

\begin{bTable}
  \RBall (6,2);   \VBall (4,2);
  \YBall (2,2);   \GBall (5,2);
  \WBall (2,1.5); \OBall (7,1);
  \BBall (1,2);   \PBall (7,3);
  \KBall (3,2);   \MBall (7,2);
\end{bTable}

```

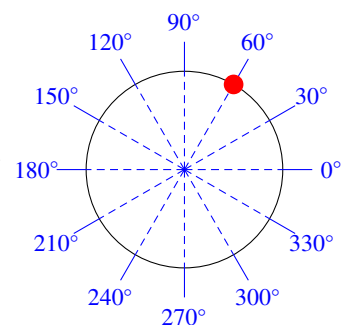
`\draw` These commands are used to draw trajectories on the board. The `\draw` command is based on
`\dash` the standard TikZ one, with additional handling to integrate with the package's layer system. The
`\dash` command is provided as a convenience alias for a densely dashed version of `\draw`.

Note that, regardless of where these macros are used in the environment's code, balls will always render *above* all path elements.

`\wpos` When the `\WBall` command is used, the `\wpos` macro stores the coordinates of the white ball.
`\rpos` This may subsequently be reused as a shorthand for the ball's position. The same behaviour
`\ypos` applies to the other ball colours provided by the package.
`[...]` The `\wpos` macro may be used before `\WBall` is called; similarly for the corresponding macros of the other colours. However, if no `\WBall` command has been issued, `\wpos` is not defined, and an Undefined control sequence error is raised. Furthermore, if `\WBall` is used multiple times, only the last specified position is stored in `\wpos`.

`\wbdry` In order to facilitate the drawing of trajectories involving collisions, the
`\rbdry` user may wish to target a point on the boundary of a ball. Consequently,
`\ybdry` when the `\RBall` command is used, the `\rbdry` macro is defined. It
`[...]` may be used within the syntax of a `\draw` command with a mandatory argument specifying the angle at which the boundary point is taken. For instance, `\rbdry[60]` denotes the point shown in red in the accompanying diagram.

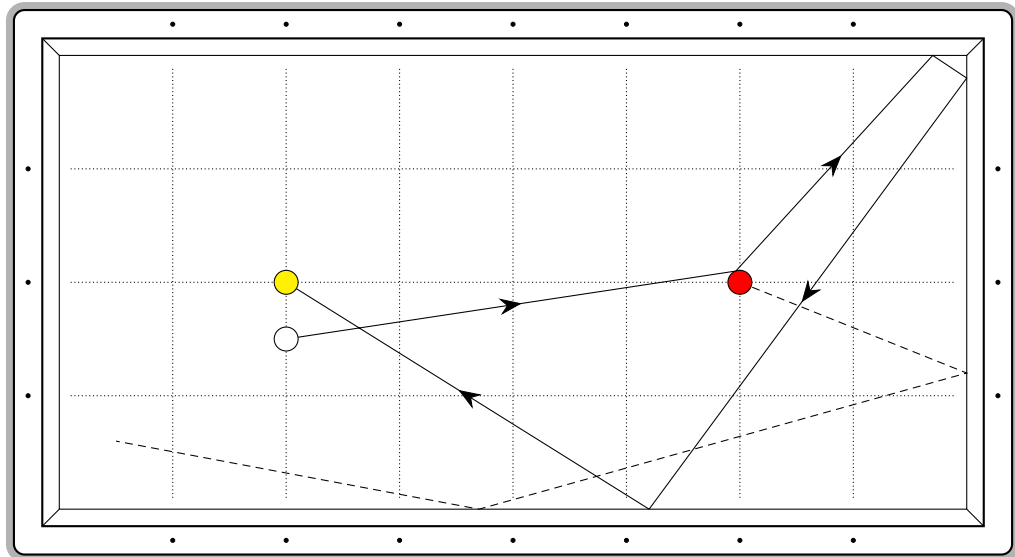
If no `\RBall` command has been issued, calling the `\rbdry` macro results in an Undefined control sequence error.



`\To` This may be used in any `\draw` command as a replacement for `--`, in order to display arrow marks along the middle of the path.

We now give an example of a diagram, where we display how to perform the game's opening shot.

¹There is a beginner-friendly variant of the game in which a fourth, blue ball is added; in this variant, striking two of the other three balls is sufficient to score a point.



```

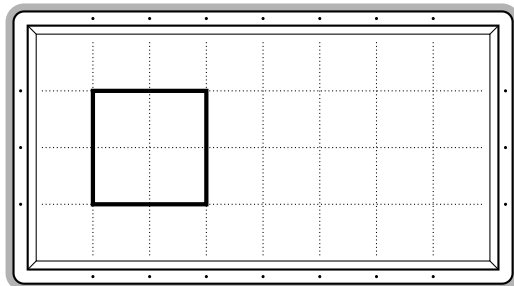
\begin{bTable}[centering,scale=1.5]
  \RBall (6,2);
  \YBall (2,2);
  \WBall (2,1.5);
  \draw \wpos \To \rbdry[110] \To (7.7,4) -- (8,3.8)
        \To (5.2,0) \To \ypos;
  \dash \rpos -- (8,1.2) -- (3.7,0) -- (.5,.6);
\end{bTable}

```

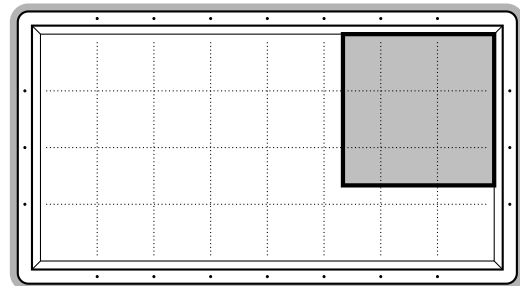
`\Box` These macros allow the user to delimit rectangular areas on the table, often indicating where a given set of balls should be located after a shot is played.

`\ShadedBox` These macros expect two point coordinates, separated by `--` and terminated by a semicolon, as is standard with TikZ commands.

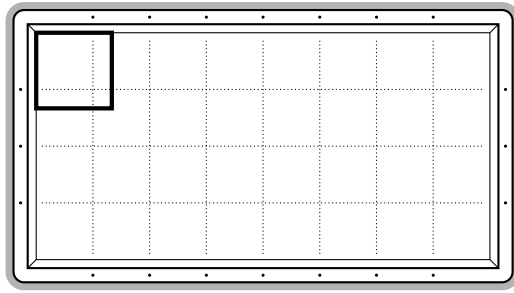
`\blBox` These are aliases for, respectively: `\Box (0,0)--(8/6,4/3)`, `\Box (8-8/6,0)--(8,4/3)`,
`\brBox` `\Box (0,4-4/3)--(8/6,4)` and `\Box (8-8/6,4-4/3)--(8,4)`. These aliases *do* expect a
`\tlBox` semicolon. They allow the user to draw square boxes whose side length is one third of the table
`\trBox` width, positioned in each of the four corners.



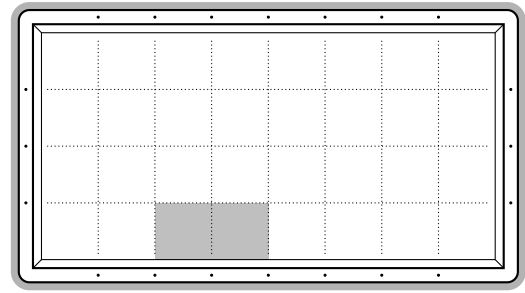
`\Box (1,1)--(3,3);`



`\ShadedBox (8-8/3,4-2*4/3)--(8,4);`

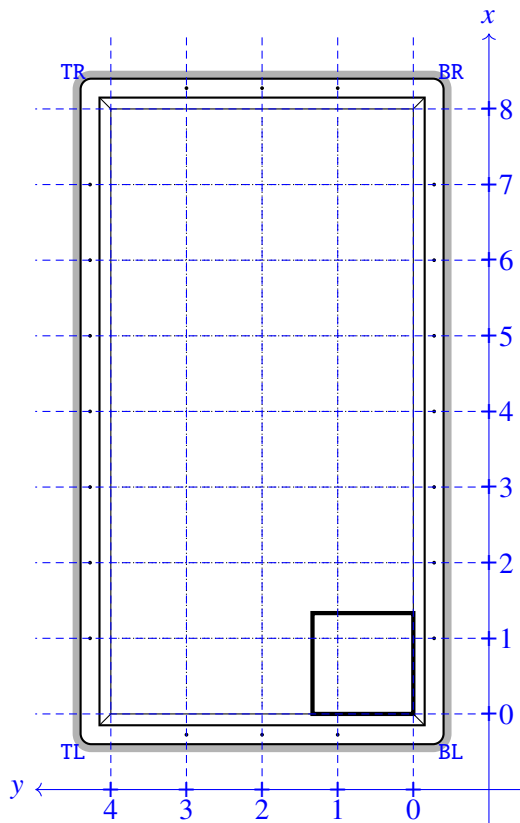


`\tlBox;`



`\ShadedRegion (2,0)--(4,1);`

In the case where the `portrait` key is set to `true`, `\blBox` is *not* placed in the bottom-left corner of the printed document. The `guidelines` reflect this by indicating the name of each corner.



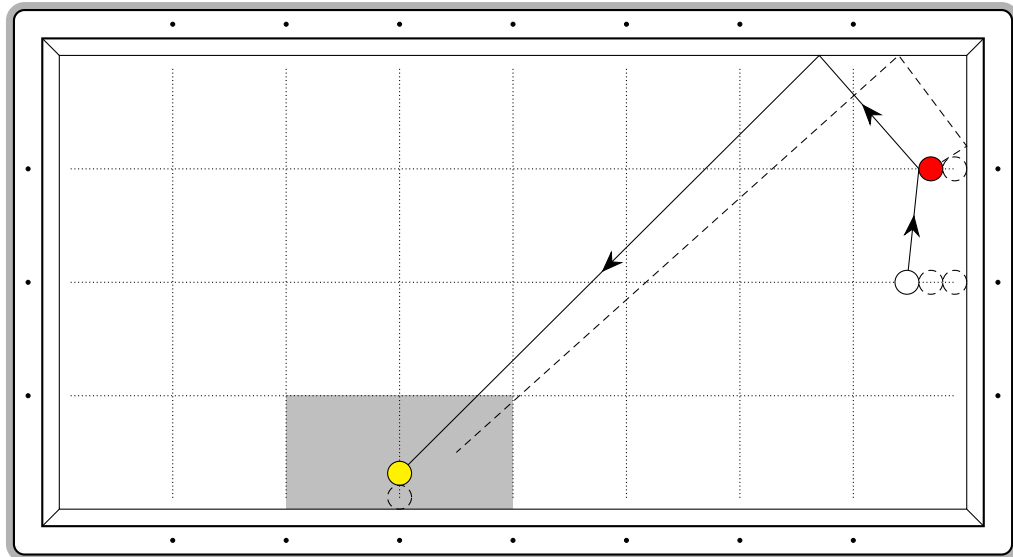
```
\begin{bTable}[portrait, guidelines]
  \blBox;
\end{bTable}
```

`\br` This macro stores the ball radius (its value is updated by the `ballsize` key) and may be used to position balls more accurately.

`\SBall` This command² displays a “fake ball”, which may be used to indicate how the actual balls should be physically positioned on the table. This command does **not** define associated `\spos` and `\sbdry` macros.

Below, we give an example of how and why these commands may be used.

²The name stands for “shadow ball”.



```

\begin{bTable}[scale=1.5,centering]
  \YBall (3,3*\br);
  \SBall (3,\br);
  \RBall (8-3*\br,3);
  \SBall (8-\br,3);
  \WBall (8-5*\br,2);
  \SBall (8-\br,2);
  \SBall (8-3*\br,2);
  \draw \wpos \To \rbdry[180] \To (6.7,4) \To \ypos;
  \dash \rpos -- (8,3.2) -- (7.4,4) -- (3.5,.5);
  \ShadedRegion (2,0) -- (4,1);
\end{bTable}

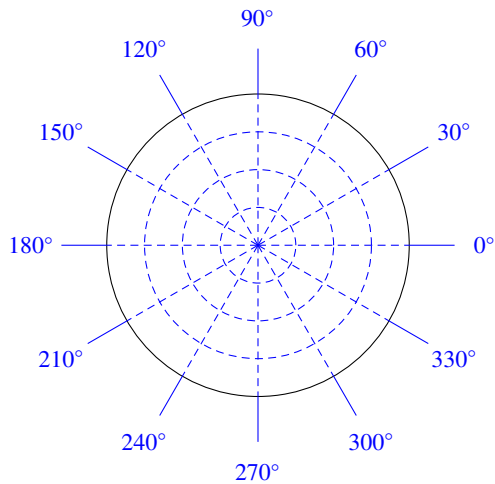
```

4 The `\Qty` Macro

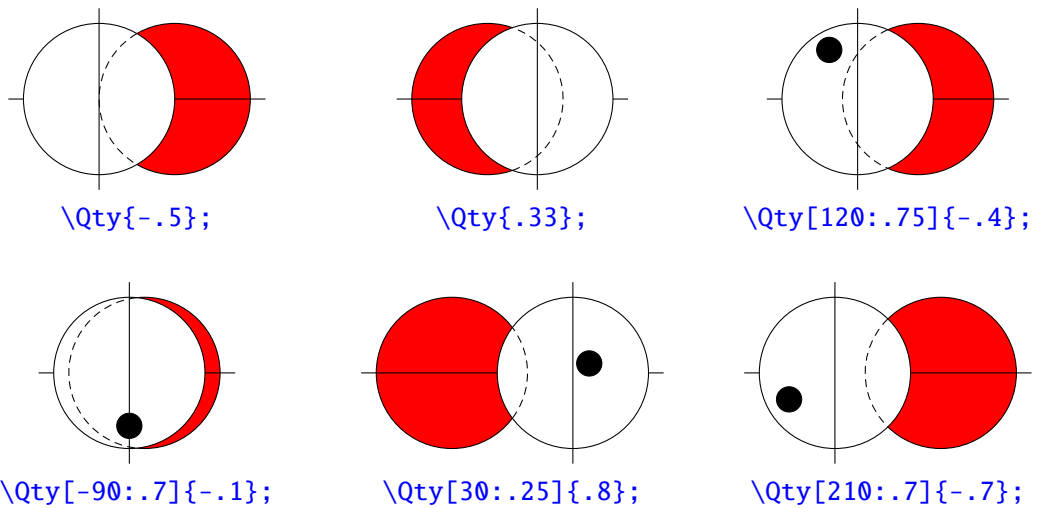
To indicate how the cue ball should be struck and how to aim at the object ball, the `\Qty` macro may be used. It can be used either inside or outside the `bTable` environment.

4.1 Inside the `bTable` Environment

This command takes an optional argument and a mandatory one. The mandatory argument is a real number between -1 and 1 , indicating the contact point on the object ball; it represents how thickly the ball is struck, with the sign determining whether the shot is taken from the left or the right. The optional argument has the form `<angle>:<radius>` and specifies, in polar coordinates (see the diagram below), the point at which the cue strikes the cue ball. The angle is set in *degrees*, and the radius is expected to be between 0 and 1 . The macro must be terminated with a semicolon, as is standard for many TikZ commands.



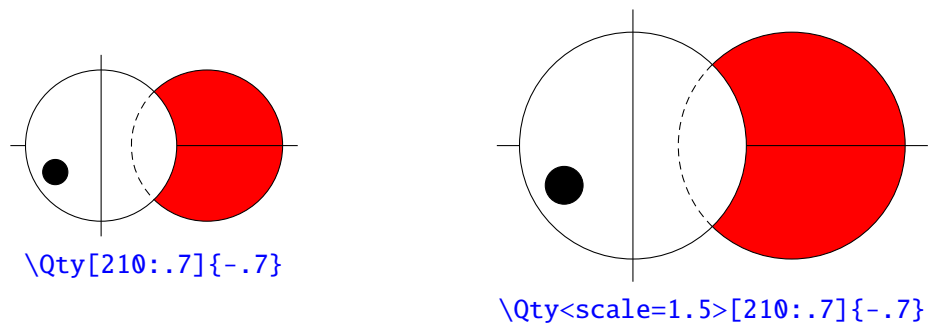
The environment's `guidelines` key may be used to display the above diagram as an overlay on the cue ball if the `\Qty` command is present. The environment's `scale` key also affects the render of this command. We give several examples below.

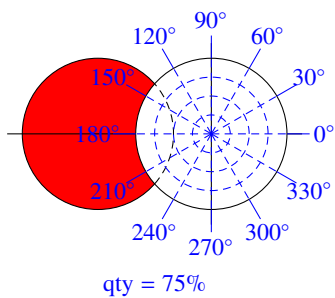


It is expected that only one `\Qty` command is present within a given `bTable` environment. However, if multiple instances are used, no error is raised, and the corresponding diagrams are simply stacked on top of one another.

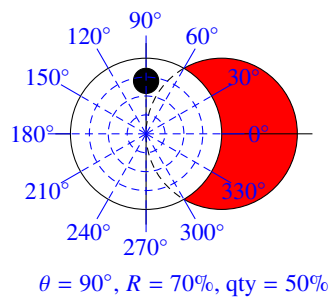
4.2 Outside the `bTable` Environment

The macro is still named `\Qty`, and its syntax remains unchanged, except that it now accepts the optional `scale` and `guidelines` key–value parameters enclosed in chevrons.





`\Qty<guidelines>{.75}`



`\Qty<guidelines>[90:.7]{-.5}`

5 TikZ Layering

No clipping path is applied to user-defined drawings; consequently, graphical elements may extend beyond the boundaries of the table.

The package renders elements in a fixed order: the table, the shaded areas, the user's `\draw` commands, the box outlines, and finally the balls. This drawing order determines the visual stacking of all elements.

Internally, every `\draw` command is stored in the `\l__billiards_drawcalls_tl` token list. Likewise, shaded areas, box outlines, and balls are stored in the `\l__billiards_shadedboxes_tl`, `\l__billiards_boxes_tl`, and `\l__billiards_balls_tl` token lists, respectively. The `\l__billiards_shadedboxes_tl` token list is processed before the grid lines are drawn.

The processing order is therefore as follows:

1. the table and the shaded areas;
2. the markings (the “mouches”, the “ancres”, the grid lines, etc.);
3. the `\draw` commands;
4. the box outlines;
5. the billiard balls;
6. the guidelines, if specified.

The output of the `\Qty` macro is processed before all items in the above list. Consequently, a `\ShadedRegion` command will be rendered on *above* the output produced by `\Qty`.

In the event of overlapping `\draw` commands, the corresponding drawings are rendered in the order in which the commands appear. The same principle applies to overlapping billiard balls.

6 Future Development

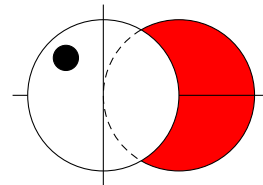
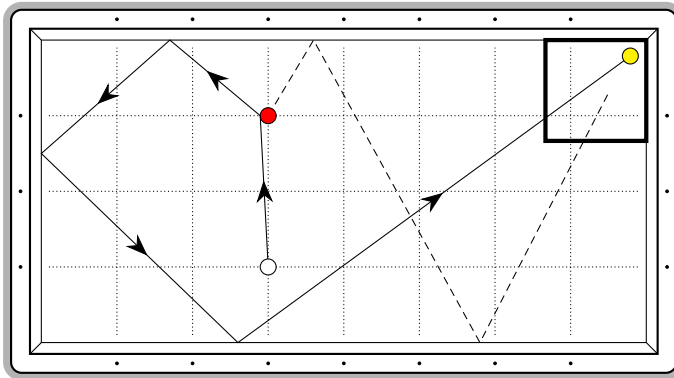
Possible areas for future development include, and are not limited to:

- support for the addition of user-defined ball colours;
- enhanced versatility of markings and annotations;
- support for easy handling of curved trajectories (used in “artistic billiards”).

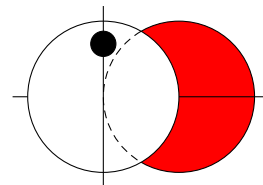
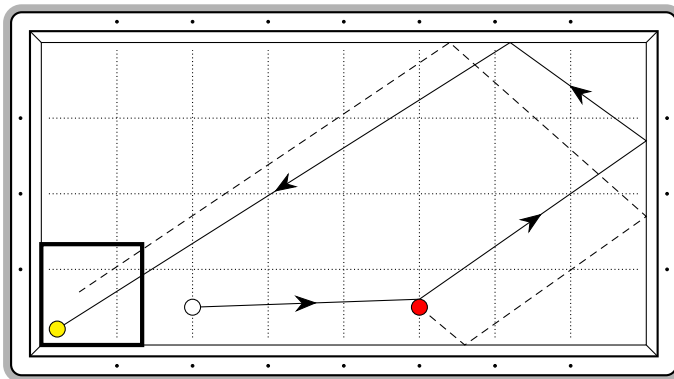
These features are under consideration and may be implemented in future releases.

7 Some Further Examples

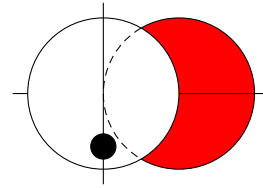
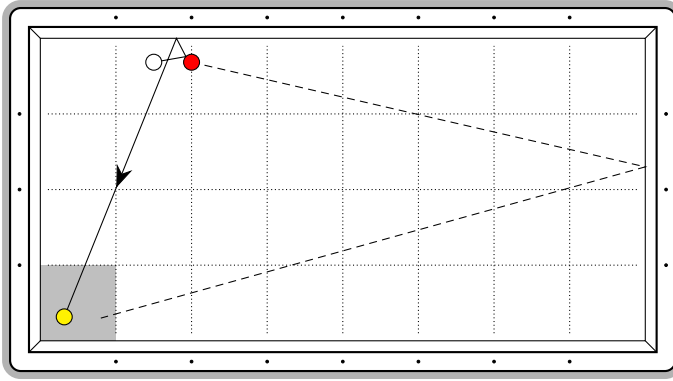
We present eight examples that illustrate a variety of use cases and showcase the possibilities offered by the package. Readers are warmly encouraged to try reproducing these shots on an actual billiard table.



```
\begin{bTable}
  \YBall (8-2*\br,4-2*\br);
  \RBall (3,3);
  \WBall (3,1);
  \trBox;
  \draw \wpos \To \rbdry[180] \To (1.7,4) \To (0,2.5) \To (2.6,0) \To \ypos;
  \Qty[135:.7]{-.5};
  \dash \rpos -- (3.6,4) -- (5.8,0) -- (7.5,3.3);
\end{bTable}
```



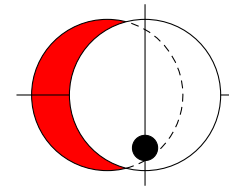
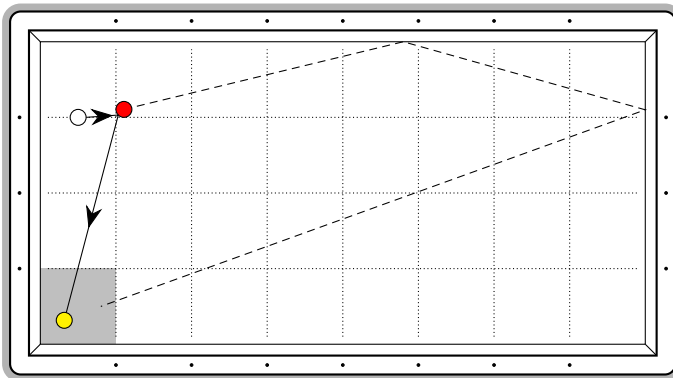
```
\begin{bTable}
  \YBall (2*\br,2*\br);
  \WBall (2,.5);
  \RBall (5,.5);
  \blBox;
  \draw \wpos \To \rbdry[90] \To (8,2.7) \To (6.2,4) \To \ypos;
  \dash \rpos -- (5.6,0) -- (8,1.7) -- (5.4,4) -- (.5,.7);
  \Qty[90:.7]{-.5};
\end{bTable}
```



```

\begin{bTable}
  \WBall (1.5,4-3*\br);
  \RBall (2,4-3*\br);
  \YBall (3*\br,3*\br);
  \draw \wpos -- \rbdry[135] -- (1.8,4) \To \ypos;
  \dash \rpos -- (8,2.3) -- (.8,.3);
  \ShadedRegion (0,0) -- (1,1);
  \Qty[-90:.7]{-.5};
\end{bTable}

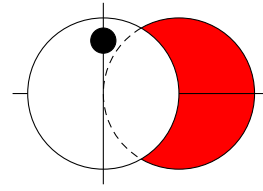
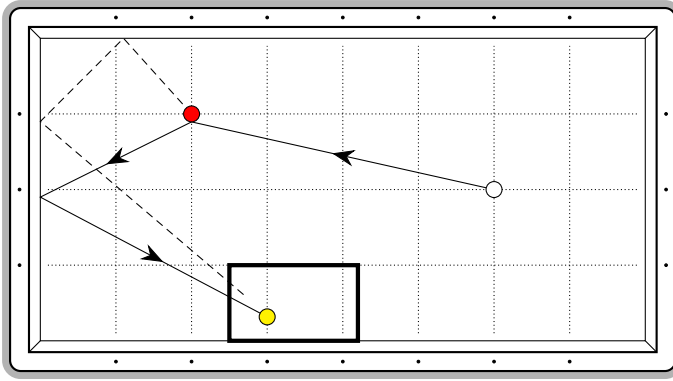
```



```

\begin{bTable}
  \WBall (.5,3);
  \RBall (1+\br,3+\br);
  \YBall (3*\br,3*\br);
  \draw \wbdry[0] \To \rbdry[-135] \To \ypos;
  \dash \rpos -- (4.8,4) -- (8,3.1) -- (.8,.5);
  \ShadedRegion (0,0) -- (1,1);
  \Qty[-90:.7]{.25};
\end{bTable}

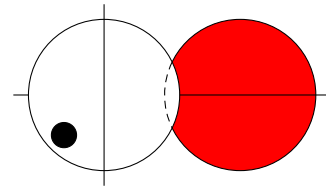
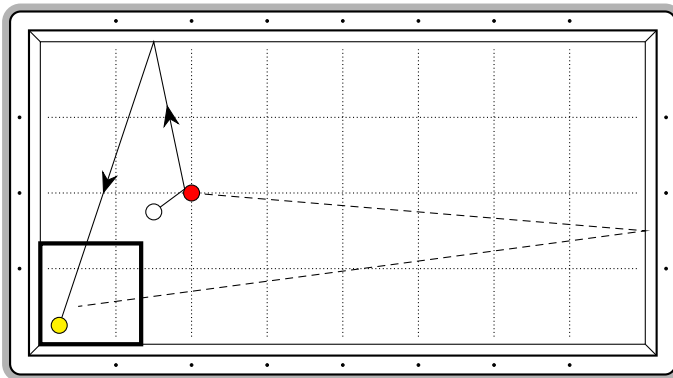
```



```

\begin{bTable}
  \WBall (6,2);
  \RBall (2,3);
  \YBall (3,3*\br);
  \draw \wpos \To \rbdry[-90] \To (0,1.9) \To \ypos;
  \dash \rpos -- (1.1,4) -- (0,2.9) -- (2.7,.6);
  \Box (2.5,0) -- (4.2,1);
  \Qty[90:.7]{-.5};
\end{bTable}

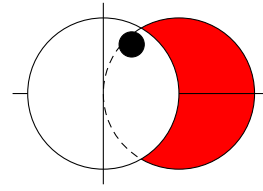
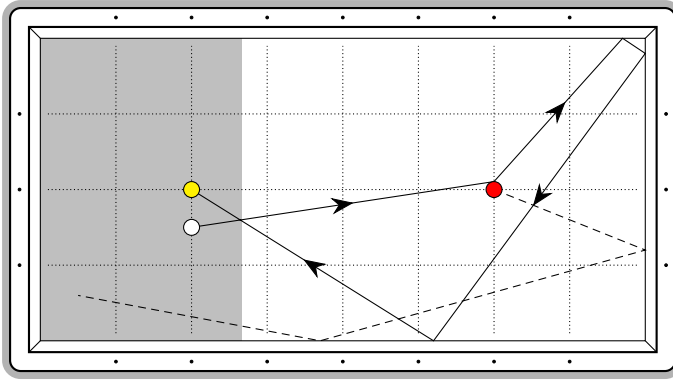
```



```

\begin{bTable}
  \RBall (2,2);
  \WBall (1.5,1.75);
  \YBall (.25,.25);
  \draw \wpos -- \rbdry[150] \To (1.5,4) \To (.25,.25);
  \dash \rpos -- (8,1.5) -- (.5,.5);
  \blBox;
  \Qty[-135:.75]{-.9};
\end{bTable}

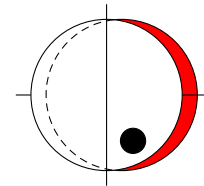
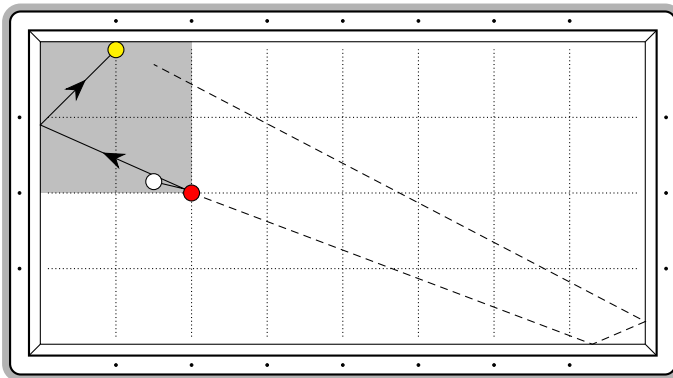
```



```

\begin{bTable}
  \RBall (6,2);
  \YBall (2,2.5);
  \WBall (2,1.5);
  \Qty[60:.75]{-.5};
  \draw \wpos \To \rbdry[90] \To (7.7,4) -- (8,3.8) \To (5.2,0) \To \ypos;
  \dash \rpos -- (8,1.2) -- (3.7,0) -- (.5,.6);
  \ShadedRegion (0,0) -- (8/3,4);
\end{bTable}

```



```

\begin{bTable}
  \YBall (1,4-\br);
  \RBall (2,2);
  \WBall (1.5,2.15);
  \Qty[300:.7]{-.1};
  \draw \wpos -- \rbdry[150] \To (0,2.9) \To \ypos;
  \ShadedRegion (0,2) -- (2,4);
  \dash \rpos -- (7.3,0) -- (8,.3) -- (1.5,3.7);
\end{bTable}

```