

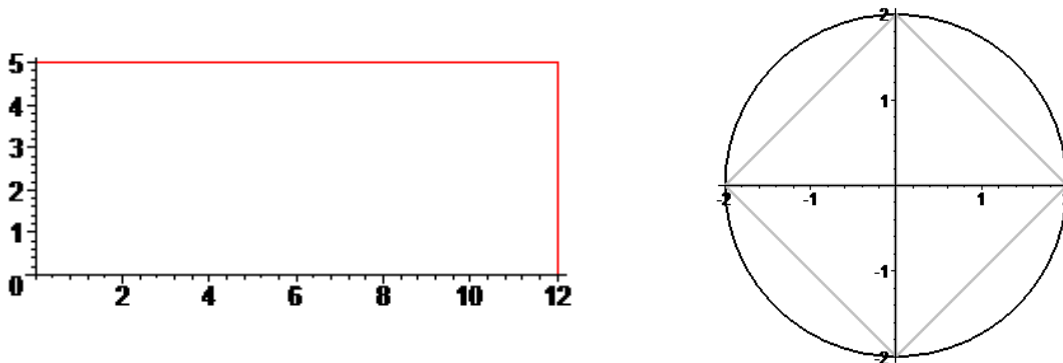
Making a Movie using Maple

© Philip B. Yasskin, Texas A&M Univ., 1997-2006

1. **The Story:** Decide the basic story to be in your movie and the objects to be in the pictures. If two parts of an object will move separately from each other, then they are two different objects. As an example, we will make a movie of a wagon moving across the screen. The objects are the wagon body and its wheels.
2. **Sketch the Objects:** Draw each object on graph paper. Beginners should only use:
 - a. Connected Straight Lines as in a dot-to-dot picture. Put each dot at a grid point.
 - b. Circles. Put the center at a grid point. You must know the radius of the circle.Advanced designers may also use (among others)

- a. Ellipse. Put the center at a grid point. You must know the horizontal and vertical radii of the ellipse.
- b. Arc of Circle. Put the center at a grid point. You must know the radius of the circle and the starting and finishing angles (in radians) measured counterclockwise from the right.
- c. Polygon. This is the same as a connected straight line except that the last point is automatically connected back to the first point. Put each dot at a grid point.

Here are the wagon body and a wheel:



3. **Origin and Scale:** Put a pair of axes on each object, one horizontal and one vertical. The point where the axes intersect is called the origin. Put a scale on each axis.
 - a. Put the origin at a convenient reference point in the object. Do not worry about the relative location of different objects since that will change in different frames of the movie.
 - b. If you plan to make an object rotate, put the origin at the pivot point. For example, if you plan to make a triangle rotate about one corner, put the origin at that corner.
 - c. Be sure the scales of different objects are reasonable. For example, a wagon should be bigger than its wheels.

For the wagon body, the origin is at the lower left corner and the dimensions are 12 by 5. For the wagon wheel, the origin is at the center and the radius is 2. This was done in the diagrams above.

4. **Begin your Maple file:** Launch Maple. Select File, New and Worksheet Mode. On the first line type `> restart; with(plots); with(plottools);` and execute it by pressing Enter. You will see the list of all the commands that are available in addition to the standard command `plot`. Many of these commands are 3 dimensional such as `sphere`. You should not use the 3 dimensional commands in your first movie. To see a help page on a command, type a `?` followed by the name of the command and press Enter.

5. **Saving, Quitting and Restarting:** To save your file, click on the **Save** icon or select **File** and **Save**. Browse to a folder where you can put your file. Type a name for your file. Click on **Save**.

Save Your File Frequently, because Maple can die!

Later times when you save, you will not have to enter a file name. If you want to change the name, use **File** and **Save As**.

When you need to quit for the day, you should save your file in a form which uses less hard disk space. The plots are huge. So you want to temporarily delete them. The most convenient way is to select **Edit**, **Remove Output** and **From Worksheet**. Then click on the **Save** icon. Then quit by clicking on the **X** or selecting **File** and **Exit**.

The next time you want to continue working on your file, launch Maple, and click on **File** and **Open**. Find your folder and click on your file. Then click **Open**.

If you removed the output before saving, there will not be any output. If you did not remove the output, it will still be on the screen, but: **CAUTION:** What you see on the screen is only on the screen and is not in Maple's memory. It only gets into Maple's memory when you execute a line by pressing **Enter**. To execute the entire worksheet and get it into memory, click on the **!!!** icon or select **Edit**, **Execute** and **Worksheet**. **CAUTION:** Maple executes the lines in the order they appear in the worksheet. If you jumped around in the worksheet when you created it, then they may not be executed in the order you intended.

6. **Plotting Objects:** Plot each object separately using the `plot` and `display` commands and the commands from the `plottools` package.

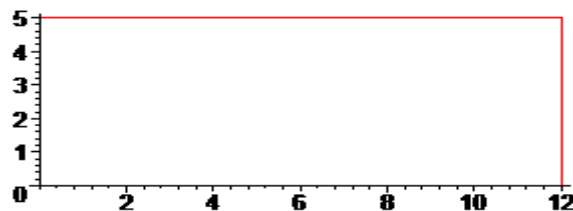
- Every command must end with a semi-colon (;) or a colon (:). The semi-colon will show the output, the colon will not.
- Extra spaces do not matter.

For example, to plot the body of the wagon, execute:

```
> body := plot( [[0,0], [12,0], [12,5], [0,5], [0,0]], color=red ):
```

- Each point has the form `[a,b]`. A list of points to be connected dot-to-dot must be enclosed in square brackets `[]` and separated by commas. This may be followed by a comma and a list of plot options (discussed below) like the choice of `color`. All of the arguments to the `plot` command (or any other command) must then be enclosed in parentheses `()`. You won't see any output because of the colon.
- This command stores the result in a memory location called `body`. To see the result, just type the name of the memory location:

```
> body;
```



Similarly to plot a wheel, first plot the straight lines (the spokes and the diamond). To see the plot, type its name.

```
> spokes := plot( [ [[-2,0], [2,0]], [[0,-2], [0,2]], [[2,0], [0,2],  
[-2,0], [0,-2], [2,0]] ], thickness=3, color=gray );  
> spokes;
```

- If several dot-to-dot lists are to be plotted, enclose them in square brackets and separate them by commas.
- The option `thickness=3` makes the lines thicker.

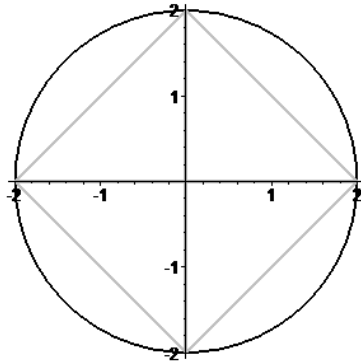
Next plot the rim using the `circle` command from the `plottools` package.

```
> rim := circle( [0,0], 2, thickness=2, color=black):
> display(rim);
```

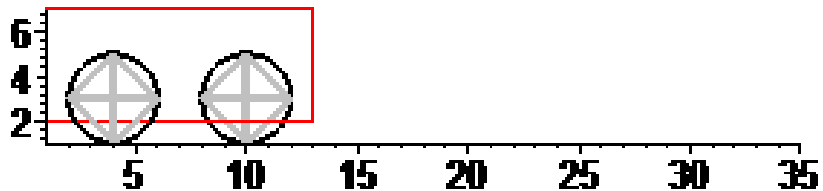
- The first argument to the `circle` command is the center, here $[0, 0]$. The second argument is the radius, here 2. These are followed by options.
- CAUTION: To see the result of a command in the `plottools` package, you must use the `display` command.

Finally, put the spokes and rim together using the `display` command. To see the result, type its name.

```
> wheel := display( [spokes, rim] ):
> wheel;
```



7. **The Frames:** You are now ready to make the frames of your movie. In our example, we want the first frame to look like:



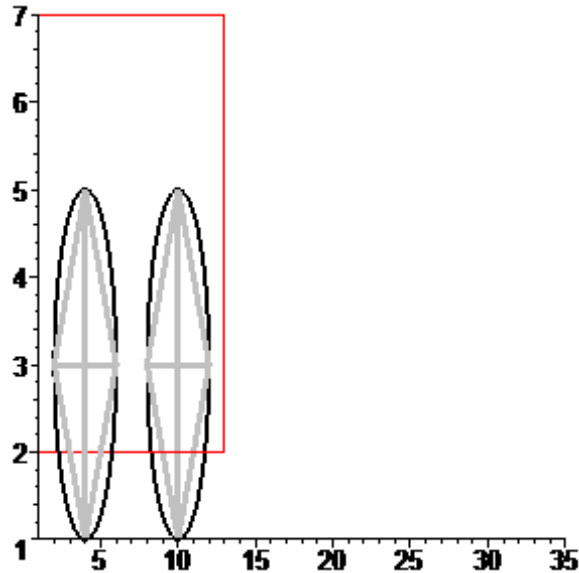
To put the objects into the frame, we use the `translate` command from the `plottools` package. This command moves the object left or right and up or down. For example, in our original plot of the body, the lower left corner on the body is at the origin $[0, 0]$. In the first frame, we want to move the lower left corner to $[1, 2]$. So, we enter the commands

```
> p1 := display(translate(body,1,2)):
> p1;
```

- Note: That's a *one* after the *p* not an *elle*.

Similarly, the center of the wheel is presently at the origin $[0, 0]$ and we want to add two wheels to the first frame with centers at $[4, 3]$ and $[10, 3]$. So we modify the above commands to read

```
> p1 := display({translate(body,1,2), translate(wheel,4,3),
translate(wheel,10,3)}):
> p1;
```



(Don't worry if the shape is distorted. We'll fix it later.) In each subsequent frame, let's move the wagon to the right by 2 units. To do this, the horizontal coordinate of the body and each wheel must increase by 2 in each frame. To reduce the amount of typing, copy and paste the "p1" command above onto a new line and press return. Repeat this 11 times. Then edit those 12 lines until they look like the following:

Do Not Retype All of This! Use Copy, Paste and Edit!

```
> p1 := display({translate(body,1,2), > translate(wheel,4,3),
translate(wheel,10,3)}):
> p2 := display({translate(body,3,2), > translate(wheel,6,3),
translate(wheel,12,3)}):
> p3 := display({translate(body,5,2), > translate(wheel,8,3),
translate(wheel,14,3)}):
      ⋮           ⋮           ⋮
> p12 := display({translate(body,23,2), > translate(wheel,26,3),
translate(wheel,32,3)}):
```

The 's mean to fill in the extra lines. Now put the cursor on the first line and press enter, 12 times. You have now made the frames of your movie.

- The Movie:** To view your movie, you must display your frames sequentially. This means one after another in time. To do this, you use the `display` command with the option `insequence=true` and you need square brackets around the list of plots to maintain the order of the frames. For the wagon, we have

```
> display( [ p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12 ],
insequence=true);
```

To run the movie, click in the plot and click on the **Loop** and **Play** buttons in the toolbar at the top. The **Loop** button is a circular arrow in one of the drop down menus; the **Play** button is a fat right arrow.

There are a couple of problems: The wagon is too tall and skinny. There is no space around the wagon. And we do not want the axes. We fix this by adding a couple more options to the `display` command:

```
> display( [ p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12 ],
insequence=true, scaling=constrained, view=[0..36, 0..9], axes=none);
```

9. **plot and display options:** There are many options which may be added to the end of the `plot` and `display` commands or any of the 2D commands in the `plottools` package. To see a full list of options, type `?plot, options` and press **Enter**. Here is a list of the most important ones:
- `color=...` specifies the color of the plot. To see the list of acceptable colors type `?color` and press **Enter**. Some commands color a line or a curve; other commands fill in a region with color. The `plot` command with dot-to-dot pictures, the `circle` command and the `ellipse` command in the `plottools` package just color the curve. To fill in a circle, use the `disk` command in the `plottools` package. To fill in a closed dot-to-dot curve or an ellipse, add the option `filled=true`, or use the `polygon` command in the `plottools` package. The main differences between dot-to-dot pictures drawn with `plot` and `polygon` are that `polygon` automatically connects its last point back to the first point and it automatically fills in the polygon. (CAUTION: Maple may not correctly color a polygon if it is not convex.)
 - `thickness=n` sets the thickness of the lines. `n` is a positive integer. The default is `n=1`.
 - `style=point` prevents dot-to-dot pictures from connecting the dots, plotting just the points. Then the option `symbol=S` specifies the symbol for points in the plot where `S` is one of `BOX`, `CROSS`, `CIRCLE`, `POINT`, and `DIAMOND`.
 - `scaling=constrained` guarantees that the horizontal and vertical scales are equal. Otherwise, Maple will rescale each axis to fit nicely on the page, thereby turning a circle into an ellipse.
 - `axes=none` eliminates the axes on the plots. You should not use this option until the very final step, because you will find the axes are useful in deciding where to put the objects in each frame.
 - `insequence=true` can only be used in the `display` command. Without this option, the `display` command will combine the various plots into a single plot. With this option, the `display` command will display the various plots in sequence as a movie.
 - `view=[a..b, c..d]` can only be used in the `display` command. It ensures that the horizontal axis runs from `a` to `b` and the vertical axis runs from `c` to `d`.

10. **Rotation:** Our movie is fine, but we really wanted the wheels to rotate. First, you need to know that Maple measures all angles in radians rather than degrees. Here is a small conversion table:

0°	15°	30°	45°	60°	75°	90°	180°	270°	360°
0 rad	$\frac{\pi}{12}$ rad	$\frac{\pi}{6}$ rad	$\frac{\pi}{4}$ rad	$\frac{\pi}{3}$ rad	$\frac{5\pi}{12}$ rad	$\frac{\pi}{2}$ rad	π rad	$\frac{3\pi}{2}$ rad	2π rad

● Note: π is entered as `Pi` with a capital P and a small i.

To rotate an object, we use the `rotate` command from the `plottools` package which rotates an object counterclockwise if the angle is positive and clockwise if the angle is negative. Since the wagon moves to the right, the wheels must rotate clockwise. We will rotate the wheels 30° in each frame. So to produce pictures of the wheel rotated 30° and 60° clockwise, we enter:

```
> wheel2 := rotate(wheel, -Pi/6):
> wheel3 := rotate(wheel, -Pi/3):
```

(You can specify a center of rotation at the end of the `rotate` command. See `?rotate`.)

We do not change frame `p1`. In the frame `p2` we replace `wheel` by `wheel2`. In the frame `p3` we replace `wheel` by `wheel3`. In the frame `p4` we want a 90° rotation, but looking at the wheel we see that this will not change the picture. So we do not change frame `p4`. Similarly, we replace `wheel` by `wheel2` in frames `p5`, `p8` and `p11`. And we replace `wheel` by `wheel3` in frames `p6`, `p9` and `p12`. So the commands to make the movie should look like

```

> p1:=display({translate(body,1,2), translate(wheel,4,3),
translate(wheel,10,3)}):
> p2:=display({translate(body,3,2), translate(wheel2,6,3),
translate(wheel2,12,3)}):
> p3:=display({translate(body,5,2), translate(wheel3,8,3),
translate(wheel3,14,3)}):
> p4:=display({translate(body,7,2), translate(wheel,10,3),
translate(wheel,16,3)}):
      ⋮      ⋮      ⋮
> p12:=display({translate(body,23,2), translate(wheel3,26,3),
translate(wheel3,32,3)}):
> display( [ p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p11,
p10, p9, p8, p7, p6, p5, p4, p3, p2 ], insequence=true,
scaling=constrained, view=[0..36, 0..9], axes=none);

```

- Other commands:** There are several other useful commands in the `plots` and `plottools` packages. To find out about these, read the help pages by typing a `?` followed by the name of the commands.

The `plots` package has a `textplot` command for adding text to the plot. Each `textplot` begins with a list of the horizontal and vertical positions for some text and the text enclosed in quotes. The `align` option says that the text should be placed ABOVE or BELOW and LEFT or RIGHT of the position with the default being centered. The `font` option sets the font and its size. The `display` command combines the text and a plot.

The `plottools` package has several additional shapes: `line`, `polygon`, `curve`, `hyperbola`, `disk`, `ellipse`, `arc`, `pieslice`, `ellipticArc`, `arrow`. The help pages are self-explanatory except for `arc`, `pieslice` and `ellipticArc`. These commands give a piece of a circle, disk or ellipse respectively, between two specified angles measured in radians. The zero angle is horizontal to the right. Positive angles are measured counterclockwise from zero. Negative angles are clockwise.

In addition, the `plottools` package also has commands to reflect and scale a previous plot as well as to translate and rotate them.

- Working at Home:** If you have a computer at home, you may be able to work at home.

If you have Maple at home, (Most of you do not.) you can email the file to yourself or drag the file onto a floppy disk or USB drive. Then open the file from Maple at home.

If you do not have Maple at home, you can still type into your file at home. You do this by exporting the file as **Maple Text**. From within Maple, click on **File** and **Export As**. From the "Files of type" drop down menu select **Maple Text**. Give the file a name and click **Save**. Email the file to yourself or drag the file onto a floppy disk or USB drive. At home you may edit the file using Notepad on Windows or SimpleText on MAC. You cannot execute anything or see any plots. When you come back to school, you can reopen your file from within Maple by clicking on **File** and **Open**. Set the file type to **Text**, select the file, click on **Open** and select **Maple Text**. Now you must execute each line and correct all of your typing mistakes. A problem arises in that everything executes as one group. To fix this, put your cursor at the beginning of each Maple command and press F3. This will split the execution group.

- The Internet:** Finally, you can save your movie as an animated gif file, so that you can include it on a web page. To do this, right click in the plot and select **Export As >> GIF**. Save it as name `.gif` View your movie by finding it in Windows Explorer and double clicking on it.