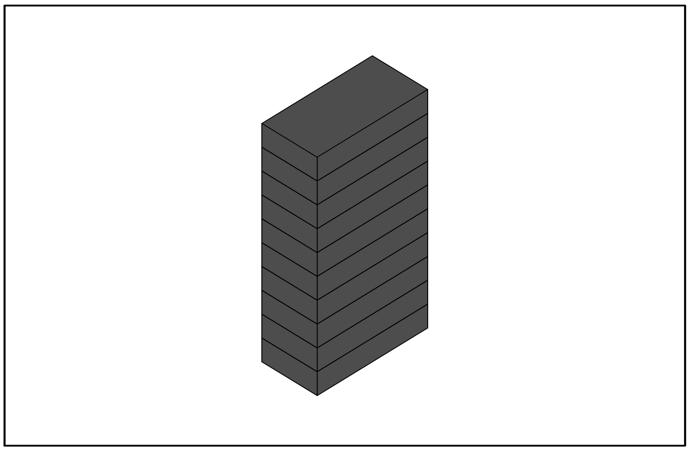
1 Making it Move - an Introduction to Animations

In previous chapters we have learned how to make lists of Maple plot structures and then to display them all together in the same frame as a single picture. There is a simple option to "plots[display]" called "insequence" which if set to "true" displays the pictures not all together but one at a time. This is all it takes to make movies. Lets start simply, by animating the stacking of a small pile of bricks. In Chapter 4 we made stacks of bricks using a word BL defined as follows:

```
BL := proc (shift)
local v1, v2, v3, v4, v5, v6, v7, v8, front, back, left, right,
bottom, top, box;
v1:=[0, 0, 0]+shift; v2 := [8, 0, 0]+shift; v3 := [8, 4, 0]+shift;
v4 := [0, 4, 0]+shift; v5 := [0, 0, 2]+shift; v6 := [8, 0, 2]+shift;
v7 := [8, 4, 2]+shift; v8 := [0, 4, 2]+shift;
front := [v1, v2, v3, v4]; back := [v5, v6, v7, v8];
left := [v4, v8, v5, v1]; right := [v3, v2, v6, v7];
bottom := [v3, v4, v8, v7]; top := [v1, v2, v6, v5];
box := [front, back, left, right, top, bottom];
plots[polygonplot3d](box,color=red,style=patch, scaling=constrained )
end:

bricklist:=[seq(BL([0,0,2*n]),n=0..9)]:

plots[display](bricklist);
```



Now lets try that last command with "insequence" set equal to "true". After executing the command "click" anywhere on the display to open a new "toolbar" and click on the large solid triangular icon or pull-down the "animation" menu which appears and select "play".

```
plots[display](bricklist,insequence=true);
```

Suppose we **want a brick to "slide"** along the x-axis. We simply generate a sequence of bricks occupying intermediate positions and display it "insequence =true". For example we could define

```
slider:= seq( BL([t/5,0,0]),t=0..20):
```

```
plots[display]([slider],insequence=true,
axes=boxed);
```

The brick would look better if it had a "road" to slide along. that just means we need a colored polygon "under" it.

```
road:=plots[polygonplot3d]([[-1,-1,0],[20,-1,
0],[20,5,0],[-1,5,0]],
color=blue,scaling=constrained,style=patch):
```

Now we just add the road to each frame:

```
slider:= seq(
plots[display]([road,BL([t/5,0,0])]),t=0..20):
plots[display]([slider],insequence=true);
```

Maybe we would like to see a brick "hop up and down". All we need to do is:

```
hopper:=seq( BL([0,0, 3*(t mod 2)]),t=0..10):
```

```
plots[display]([hopper],insequence=true);
```

Maybe we would like to have it simply sit there and alternate its colors. Recall that we have a word to make colored bricks

```
CBP := proc (shift,clr )
local v1, v2, v3, v4, v5, v6, v7, v8, front, back, left, right,
bottom, top, box;
v1 := [0, 0, 0]+shift; v2 := [8, 0, 0]+shift; v3 := [8, 4, 0]+shift;
v4 := [0, 4, 0]+shift; v5 := [0, 0, 2]+shift; v6 := [8, 0, 2]+shift;
v7 := [8, 4, 2]+shift; v8 := [0, 4, 2]+shift;
front := [v1, v2, v3, v4]; back := [v5, v6, v7, v8]; left := [v4, v8,
v5, v1];
right := [v3, v2, v6, v7]; bottom := [v3, v4, v8, v7]; top := [v1, v2,
v6, v5];
box := [front, back, left, right, top, bottom];
plots[polygonplot3d](box,color=clr,style=patch, scaling=constrained )
end:
```

This time we **make the list to be animated** the "movie", as we go. Recall that it is usually best to build lists by buiding sequences and then converting them to lists with brackets "[]". We start with Maple's empty list "NULL"

```
movie:=NULL:
for i from 1 to 10 do
if (i mod 2) = 0 then frame:=CBP([0,0,0],red)
else frame:=CBP([0,0,0],blue) fi;
movie:=movie,frame:
od:

plots[display](movie,insequence=true);
```

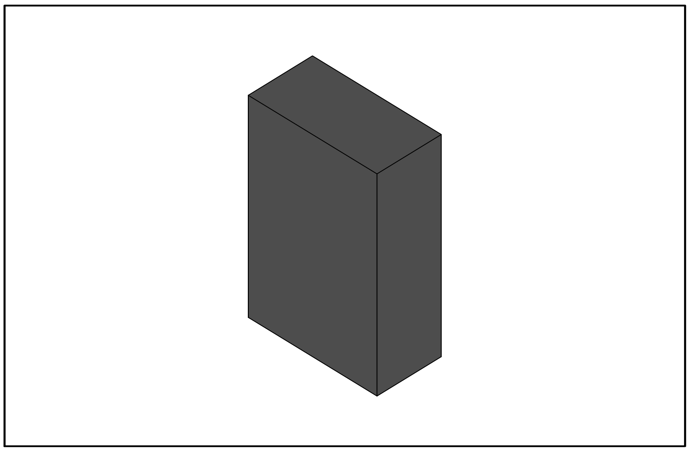
1.0.1 *The above illustrates a useful model for making animations We use an empty list called something like "MOVIE" and a word like "FRAME" to represent a typical frame of the movie. We have a sequence of Maple commands whose output is a plot structure and use a "do-loop" to construct the animation sequence one frame at a time:*

MOVIE:=NULL;
for INDEX from START to STOP do
{sequence of Maple commands which produces a plot structure "FRAME"}
MOVIE:=MOVIE,FRAME;
od:
plots[display]([MOVIE], insequence=true);

We can further illustrate this by making a movie of a brick moving "end-over-end". What we need to do is alternate horizontal and vertical bricks and account for the different translation distances. We need a vertical brick layer which we might as well call "VBL"

```
VBL := proc (shift,clr )
local v1, v2, v3, v4, v5, v6, v7, v8, front, back, left, right,
bottom, top, box;
v1 := [0, 0, 0]+shift; v2 := [0, 0, 8]+shift; v3 := [0, 4, 8]+shift;
v4 := [0, 4, 0]+shift; v5 := [2, 0, 0]+shift; v6 := [2, 0, 8]+shift;
v7 := [2, 4, 8]+shift; v8 := [2, 4, 0]+shift;
front := [v1, v2, v3, v4]; back := [v5, v6, v7, v8];
left := [v4, v8, v5, v1]; right := [v3, v2, v6, v7];
bottom := [v3, v4, v8, v7]; top := [v1, v2, v6, v5];
box := [front, back, left, right, top, bottom];
plots[polygonplot3d](box,color=clr,style=patch, scaling=constrained )
end:

VBL([0,0,0],red);
```



Lets put a vertical brick down first and alternate horizontal and vertical after that. This means that, starting with 0 the even bricks will be vertical and the odd ones horizontal. We need to move over 2 inches after a vertical brick and eight after a horizontal

```
movie:=NULL: frame:=NULL: tmp:=[0,0,0]:
for i from 0 to 9 do
if (i mod 2) = 0 then frame:=VBL(tmp,red):  tmp:=tmp+[2,0,0];
else frame:=CBP(tmp,red):  tmp:=tmp+[8,0,0]:  fi;
movie:=movie,frame:
od:

plots[display]([movie],insequence=true);print
level:=0;
```

Suppose we would like to see a stack of bricks growing as each successive brick is added. What we need then is a new list whose first element is the picture of the first brick, whose second is a picture of the first and second bricks, whose third is the first three bricks, ... etc. Then, when we display this as an animation we will see the stack of bricks appear to grow. There are several ways to do this. First, if the list already exists it is easy to do it by hand. Lets make a list of the first three frames of the movie of this stack of bricks. An extremely useful bit of Maple notation first:

NOTATION: If "L" is a Maple list then the "nth" element of "L" is "L[n]"—

For instance "L[1]" is the first element, "L[2]" is the second, etc.

Recall that "plots[display]" wants a list of displays as input.

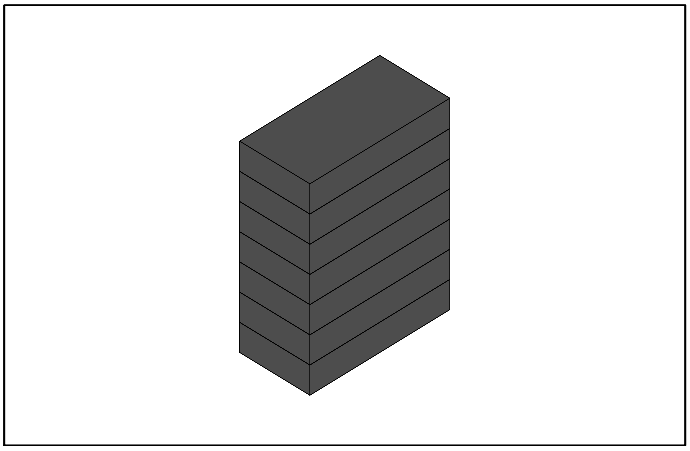
```
movielist:=[bricklist[1],
plots[display]([bricklist[1], bricklist[2]]),
plots[display]([bricklist[1],bricklist[2], bricklist[3]]) ]:

plots[display](movielist,insequence=true);
```

Obviously construction of this list would be tedious to do by hand if it were of any length. However, we know how to make the computer do that sort of thing. However before we are able to make the computer do anything we have to know how to do it at least in principle by hand. We first ask ourselves: "How do I build the seventh "frame" of the movie?" The answer is "Make a list in the same order of the first seven pictures in "bricklist" and feed it to "plots[display]". Thats easy - we do it the same way as we made the original stack of ten.

```
stack7:=plots[display]([seq(
bricklist[j],j=1..7)]):

stack7;
```

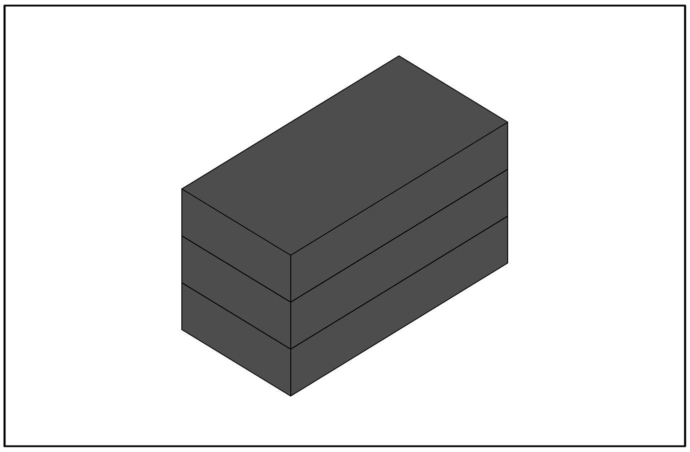


We can make a Maple word which makes the nth stack

```
stackn:=proc(n)
plots[display]([seq( bricklist[j],j=1..n)])
end;
```

$$stackn := \mathrm{proc}(n)\ plots_{display}([\mathrm{seq}(bricklist_j,\ j = 1..n)])\ \mathrm{end}$$

```
stackn(3);
```



Now we can easily make our movie

```
movie:=[seq(stackn(t),t=1..10)]:

movie:=[op(movie),seq(movie[11-i],i=1..10)]:

plots[display](movie,insequence=true);
```

In the previous example we were dealing with an extant list of known length. What if we didn't know the length of "bricklist". We might ask Maple to display it by entering its name and try to count the entries. However it might be very long with each entry very complicated. There is a simple solution If "L" is a list then **nops (L);** returns the length of "L".

```
nops(bricklist);

nops(movie);
```

Thus this sequence works and would work even if we didn't know the length of "L"

```
movie:=[seq(stackn(t),t=1..nops(bricklist))]:

plots[display](movie,insequence=true);
```

Usually we don't encounter a list we want to animate but rather we have a process we want to illustrate and we make a "movie" by assembling the final list - the one which will produce the animation we want from the outset. We don't usually make the analog of "bricklist" but go straight to "movie". Suppose we want to make a 7-frame brick stacking movie. Recall that "NULL" is the expression sequence with no elements.

```
movie:= NULL ;
frame:=NULL;
```

```
for p from 1 to 7 do
tmp:= BL([0,0,2*p]);
frame:=plots[display]([frame,tmp]);
movie:=movie,frame;
od:

plots[display]([movie],insequence=true);
```

This illustrates a basic pattern one can use to animate a "growing" structure. We start with an empty sequence "MOVIE" and an empty sequence "FRAME".

```
MOVIE:=NULL;
FRAME:= NULL;
for INDEX from START to STOP do
‘[sequence of Maple commands which generates a plot structure TMP]‘;
FRAME:=plots[display]([FRAME,TMP]);
MOVIE:=MOVIE, FRAME ;
od:

plots[display]([MOVIE],insequence=true);
```

We can illustrate by making a movie which stacks alternate colored red and blue bricks, using the "CBP" word from above.

```
movie:=NULL: frame:=NULL:
for q from 1 to 10 do
if (q mod 2 = 0) then tmp:=CBP([0,0,2*q], red);
else tmp:= CBP([0,0,2*q], blue) fi;
frame:=plots[display]([frame,tmp]);
movie:=movie,frame;
od:

plots[display]([movie],insequence=true);
```

1.1 Exercises on Movie Making

1.1.1 Exercise: *The following makes a movie that counts from 0 to 10.*

```
MOV:=NULL;
for k from 0 to 10 do
txt:=cat( k,‘ ‘,‘and‘,‘ ‘,‘counting‘):
MOV:=MOV, plots[textplot]([0,0,txt], axes=none, font=[TIMES,
BOLD,28]): od:

plots[display]([MOV],insequence=true);
```

1.1.2 *a. Modify the above to count to 50.*

1.1.3 *b. The following will make a sequence which reverses MOV*

```
VOM:=seq(MOV[11-i],i=0..10):
```

1.1.4 *Put the sequences MOV and VOM together into a movie (by combining the two sequences into one) which counts up to 10 then back down to 0.*

1.1.5 *c, Make a movie that counts from 3 to 9 then down to -5.*

1.1.6 Exercise (Movie Splicing)*:*

1.1.7 *a. Modify the following movie from the chapter text to make 10 frame movie of bricks stacking. Call it "tenbricks".*

```
sevenbricks:= NULL ;
frame:=NULL;
for p from 1 to 7 do
tmp:= BL([0,0,2*p]);
frame:=plots[display]([frame,tmp]);
sevenbricks:=sevenbricks,frame;
od:

plots[display]([movie],insequence=true);
```

1.1.8 *b. Make a movie called "OnetoTen" which counts from 1 to to by making a sequence of all but the first frame of the movie "MOV" in the previous problem.*

1.1.9 *c. Splice the two movies together by making a 10-frame sequence "MergedMovie" in which the ith element is plots[display[(MOV[i], OnetoTen[i]);*

1.1.10 *d. When "MergedMovie" is played the text meets the bricks. Look at the code for "MOV" and determine the computer is being told to place the text. Modify the location of the text so that it sits below the brick stack with the stack approximately over the center of the text.*

1.1.11 Exercise: *As in the chapter text the following makes a movie of alternate red, white, and blue bricks stacking.*

```
movie:=NULL: frame:=NULL:
for q from 1 to 10 do
if (q mod 2 = 0) then tmp:=CBP([0,0,2*q], red);
else tmp:= CBP([0,0,2*q], blue) fi;
frame:=plots[display]([frame,tmp]);
movie:=movie,frame;
od:
```

1.1.12 *If one selects the 'continuous' option in the animation menu the movie will cycle when you play it. The motion is not smooth however, because the movie is played from beginning to end over and over. One would like to have the bricks stack up and then "unstack". Do this by making a movie which is twice as long as this one, having the bricks unstack in the last half of the movie so that when you play it continuously, the motion is smooth. this can be done without making any more images.*

1.1.13 Exercise: *Make the hopping brick hop along a yellow road. Let each hop cover one brick length with three frames of the brick "in the air" beween frames in which it is "on the road".*

1.1.14 *Exercise: Make the brick move end over end along a yellow road.*

1.1.15 Exercise: *Make a pair of bricks, a blue one and a red one move in a straight line with each one "leap frogging" the other.*