```
// This file is an example of how one would write the initial
// frame of the animation. This file would be read by POVAnim and
// output as a series of files. Timothy A. Grubb - WallySoft - Dec 97


#include "colors.inc"
#include "textures.inc"
#include "stones.inc"

#declare SPHERE1_TRANS = <0,0,0>
#declare SPHERE1_ROTATE = <0,0,0>
#declare SPHERE2_TRANS = <-10,0,0>
#declare SPHERE2_ROTATE = <0,0,0>
#declare SPHERE3_TRANS = <10,0,0>
#declare SPHERE3_ROTATE = <0,0,0>
#declare SPHERE4_TRANS = <0,6,0>
#declare SPHERE4_ROTATE = <0,0,0>
#declare SPHERE5_TRANS = <8,4,6>
#declare SPHERE5_ROTATE = <0,0,0>
#declare CAM_ROTATE = <0,0,0>
#declare CAM_TRANS = <0,0,-10>
#declare PLANEX_ROTATE = <0,0,0>
#declare PLANEY_ROTATE = <0,0,0>
#declare PLANEZ_ROTATE = <0,0,0>

new vector

frame you

coordinates.
camera { location <0,0,0>
         look_at <0,0,.1>
values for your
         rotate CAM_ROTATE
from the file
         translate CAM_TRANS
to the .pad
}
be a couple

code that

is a
```

These are variables that can be used in POV they are user defined. They enable POVAnim to change the values over time (Frames).

All that is happening here is I declare the Initial positions of the SPHERE , CAM, and PLANE objects. Please note that the names that I used are not important. I could have used BITE_ME for SPHERE1_TRANS if I had wanted. When POVAnim reads this file it will place all the vector based variables from #declare statements (ie. The ones you see here) in a selectable listbox. From the listbox you select a variable and then click on the ADD button. The Add button pops up a dialog that allows you to enter (x , y, z) coordinates and what new vector you want your objects to be at those coordinates.

After you have set all of the values for your objects you will choose SAVE AS from the file menu. NOTE: You can save your work to the .pad format if you want but it's gonna be a couple weeks before I get a bug out of the code that reloads that file. The .paf format is a

```
                                                proposed (by me) animation scene
file desc.
                                                for POV. In order to write the file
                                                series for animation you need to
choose the
sky_sphere                                      .pov format. This will write out
the number
{                                               of files that equals what you have
entered for
  pigment                                       frames per second * number of
seconds.
  {                                             See easy.pov for a bare bones
example.
    gradient y
    color_map { [0.0 color blue 0.6] [1.0 color rgb 1] }
  }
}


// An infinite planar surface
// plane {<A, B, C>, D } where: A*x + B*y + C*z = D
plane
{
  z, // <X Y Z> unit surface normal, vector points "away from surface"
  10.0 // distance from the origin in the direction of the surface normal

// texture pigment/normal pattern
// cube checker pattern, alternates color1 and color2
pigment {
  checker
    color Black
    color White
    }
    finish { Shiny }
    rotate PLANEZ_ROTATE

}

// An infinite planar surface
// plane {<A, B, C>, D } where: A*x + B*y + C*z = D
plane
{
  y, // <X Y Z> unit surface normal, vector points "away from surface"
  -1 // distance from the origin in the direction of the surface normal

// texture pigment/normal pattern
// cube checker pattern, alternates color1 and color2
pigment {
  checker
    color Black
    color White
    }
    finish { Shiny }
    rotate PLANEY_ROTATE

}
```

```
// An infinite planar surface
// plane {<A, B, C>, D } where: A*x + B*y + C*z = D
plane
{
  x, // <X Y Z> unit surface normal, vector points "away from surface"
  10.0 // distance from the origin in the direction of the surface normal

// texture pigment/normal pattern
// cube checker pattern, alternates color1 and color2
pigment {
  checker
    color Black
    color White
    }
    finish { Shiny }
    rotate PLANEX_ROTATE

}

// An infinite planar surface
// plane {<A, B, C>, D } where: A*x + B*y + C*z = D
plane
{
  x, // <X Y Z> unit surface normal, vector points "away from surface"
  -10.0 // distance from the origin in the direction of the surface normal

// texture pigment/normal pattern
// cube checker pattern, alternates color1 and color2
pigment {
  checker
    color Black
    color White
    }
    finish { Shiny }
  rotate PLANEX_ROTATE
}


light_source { <1,1,-2> color White }

sphere { <0,0,0>, 1   texture { T_Stone21 Shiny }
  rotate SPHERE1_ROTATE translate SPHERE1_TRANS
}
sphere { <0,0,0>, 1   texture { T_Stone20 Shiny }
  rotate SPHERE2_ROTATE translate SPHERE2_TRANS
}
sphere { <0,0,0>, 1   texture { T_Stone19 Shiny }
  rotate SPHERE3_ROTATE translate SPHERE3_TRANS
}
sphere { <0,0,0>, 1   texture { T_Stone18 Shiny }
  rotate SPHERE3_ROTATE translate SPHERE3_TRANS
}
sphere { <0,0,0>, 1   texture { T_Stone17 Shiny }
  rotate SPHERE4_ROTATE translate SPHERE4_TRANS
  }
sphere { <0,0,0>, 1   texture { T_Stone16 Shiny }
  rotate SPHERE5_ROTATE translate SPHERE5_TRANS
```

```
}
```