PLANET GENERATOR

September 26, 2021

Contents

1 INTRODUCTION:

This is version 3.0 and contains many enhancements over previous versions.

Simply put, this program will map any picture (any resolution, any depth upto to the memory limit) around a sphere (hence forth referred to as a planet). The planet is then shaded based on the location of a light source. The mapping is done by equating constant latitude and longitude lines around the planet with constant horizontal and vertical lines

on the map. The planet view is computed by applying a transformation based on inputted planet rotation angles. The result can be saved as a 24 bit image at any resolution.

The method used to shade the planet involves taking the cosine of the angle between the normal vector on the surface of the planet and the light source vector and multiplying each color component with the result.

Both the planet rotation and the light source position are specified in terms of rotations about the three axes, x, y, and z. Counterclockwise rotation is positive and clockwise is negative. These angles can be specified in any order. The resulting rotations are very dependant on this order of input.

Any number of planets (or moons, as discussed later) may be generated in one image. Also, any number of pictures may be placed in the image. Both planets and pictures may be placed anywhere on the screen by specifing the center location.

An option for generating moons has been added to allow for logical placement of planets with respect to each other. If a moon is specified then its location is determined from a distance from the planet center and the rotation of the respective planet.

2 RUNNING PLANET:

To run from the Workbench double click on the planet icon. To run from a CLI window just type:

-> [run] planet

2.1 Embedded REXX Macro Language

This version of Planet registers itself as an embedded Rexx interpreter which will process commands not directly understood by the Rexx macro processor. This effectivily adds all the capabilities of the Rexx macro language to Planet program macros.

3 THE COMMANDS:

3.1 map(FILENAME, FULL/HALF, OVERLAP, TRANSPARENCY, THRESHOLD)

The FILENAME argument specifies a file containing the map to be used for generating the planet. The file specified must be a portable bit map (PBM). If the file name begins with a pipe ('|') then the standard output of the specified process will be read as input.

By default, for planets, the map is wrapped around half the sphere and duplicated on the other half (in reverse). If you would like the map to be wrapped around the full sphere then specify FULL.

The OVERLAP is the amount of pixels to overlap at the "seam". This overlap region will be color interpolated and will result in a smearing of color in this "seam" region. The overall effect is to reduce the apparent "seam" when the edges of the map do not exactly match.

TRANSPARENCY is the degree of transparency of the image. A value of 0 will make the image totally opaque (i.e. it will overwrite everything) except where the map is black (this provides for a masking ability). This in effect will make the map image appear as if it is in the foreground. However, as this value approaches 255 the overall map image will become more transparent with 255 being fully transparent (fairly useless).

The THRESHOLD is a six digit hex value for color threshold (rrggbb) where any color in the map falling below this threshold will be treated as transparent.

This command can be placed anywhere before the GENERATE command.

An example of this command is:

```
map(|jpegtopnm map/brown-ring.jpg, full, 0, 100, 010101)
```

3.2 image

The effect of this command is to take the specified map image and size it to fit the currently set resolution (set by 'view').

This command can be placed anywhere before the GENERATE command.

3.3 planet(RADIUS, DISTANCE, ATMO, ANGLE(s),...)

The RADIUS argument indicates the size of the planet. The DISTANCE argument specifies the distance between the planet and the viewpoint.

The radius and distance are specified in "boxels". A boxel is defined as a cubic pixel (or the three dimensional equivilent of a pixel). In the case of this program a boxel is further defined as a unit such that at a distance equal to the $z_{resolution}$ (see the reset command above) each displayed boxel is exactly one screen pixel in height. The further away a boxel is from the viewpoint the smaller it will appear on the screen. For example, if the $z_{resolution}$ is set to 1000 and the planet being generated had a radius of 200 boxels, with the viewpoint set at a distance of 1000 boxels the planet would appear to fill the screen from top to bottom. The radius and distance are related to the displayed radius by the following equation.

$$displayed\,radius = \frac{radius * z_{resolution}}{distance}$$

The width of a boxel is assumed to be exactly the same as its height. Since this is not true of pixels there will not be the same corresponce between boxel and pixel width as there is between the heights. The relationship between boxel and pixel width (with the viewpoint set at the $z_{resolution}$) is the following:

$$boxel\,width = (pixel\,width) * \frac{x_{aspect}}{y_{aspect}}$$

where the x and y aspect ratios are selected based on the set x and y resolutions and the display modes. The reason for doing this is to insure that the generated planet would come as close as possible to a true sphere.

The above discussion only serves to highlight the relationship between distance, radius and the displayed radius. The only effect the distance argument has is on the displayed radius, therefore, if a specific radius is required then set the distance equal to the $z_{resolution}$. However, distance does become important if any moons are generated as will be discussed below.

The ATMO argument specifies the width of the atmosphere - a transition between the planet surface and empty (transparent) space.

The planet rotation angles are input by first specifing the axis of rotation (X, Y, or Z) then the angle. These angles may be specified in any order. An angle represents a rotation referenced to a rotated or local coordinate system, therefore, the order that these angles are input greatly influences the outcome of the total rotation. A positive angle indicates a counterclockwise rotation.

Any planet generated will overwrite any existing image. Therefore, planets will always appear to be in the foreground if generated last.

An example of this command is:

```
planet(140, 1000, 5, x20, z30, y-10)
```

3.4 moon(RADIUS, DISTANCE, ANGLE(s),...)

The RADIUS argument indicates the size of the moon. The DISTANCE argument specifies the distance between the moon and a previously specified planet. The moon is always placed the specified distance along the z-axis for the planet. Therefore, the rotation of the corresponding planet will have a big effect on the placement of the moon.

The radius and distance are specified in "boxels" (discussed under the 'planet' command).

The moon rotation angles are input by first specifing the axis of rotation (X, Y, or Z) then the angle. These angles may be specified in any order. An angle represents a rotation referenced to a rotated or local coordinate system, therefore, the order that these angles are input greatly influences the outcome of the total rotation. A positive angle indicates a counterclockwise rotation. Any angles specified are in addition to the angle of rotation specified for the associated planet (in other words initially the moon will have the same rotation as the planet).

Depending on the rotation of the associated planet the moon will either appear on the front side or back side of the planet. The program does not keep track of the various objects that may be on the screen, therefore, an assumption was made that if the moon appears in front of the planet then it will be placed in front of any existing image. If the moon appears behind the planet then it will be placed behind any existing image. Therefore, if a moon is to be generated it is suggested that the planet and accompanying moon be generated before any image or other planets are generated.

An example of this command is:

```
moon(30, 200, y20)
```

3.5 light(AMBIENT, ANGLE(s),...)

With this command the light source can be placed by specifying rotation angles around the associated planet. If no light source rotation angles are specified then the light would appear to come from the positive z-axis direction.

The AMBIENT argument is a six digit hex value for an ambient color (rrggbb) which will be applied to all shaded areas giving the appearance of an ambient lighting effect.

The light source rotation angles are input by first specifing the axis of rotation (X, Y, or Z) then the angle. These angles may be specified in any order. An angle represents a rotation referenced to a rotated or local coordinate system, therefore, the order that these angles are input greatly influences the outcome of the total rotation. A positive angle indicates a counterclockwise rotation.

This light source rotation remains in effect until reset or another planet is generated. Therefore, if a moon is subsequently generated the light source will appear in the same location (unless respecified or the moon is rotated further).

This command should be placed after any 'planet' command, otherwise can be placed anywhere.

An example of this command is:

```
light(202020, x15, z20, y20)
```

3.6 ring(RADIUS1, RADIUS2, SHADE)

Rings are created by "wrapping" the currently specified map image around a ring bounded by the specified radii. A shadow is created based on the location of the light source and radius of the planet or moon associated with the ring.

The radii are specified in units of boxels (as described above). These radii are distances from the center of the planet to the inside radius (RADIUS1) and outside radius (RADIUS2) of the ring.

The SHADE argument specifies the degree of brightness to be applied to the ring plane. A value of 0 will make the rings invisible (somewhat impractical) and a value of 100 will make the rings totally bright. Of course any value between 0 and 100 may be specified resulting in various degrees of brightness.

This command should be placed after any 'planet' or 'moon' command.

An example of this command is:

```
ring(160, 200, 40)
```

3.7 center(X, Y)

This command allows placement of the planet (and moon) or image anywhere on the screen. If not specified or no values are given then the default (0,0) will be used.

This command can be placed anywhere before the 'generate' command and will remain in effect until reset.

An example of this command is:

```
center(100, 300)
```

3.8 view(XRES, YRES, DANG, ASPECT)

This command defines the view resolution, shift angle, and aspect ratio. The resolution (XRES, YRES) defines the screen resolution of the generated image. The shift angle (DANG), if specified as a positive non-zero value, will result in two side-by-side images (doubling the x resolution of the generated image) with each image generated using a screen vertical (y-axis) angular shift. The left image is generated with a shift of -DANG/2 and the right image with a shift of +DANG/2. The resulting stereoscopic image pair may be viewed using a VR headset (e.g., Google Cardboard). Note that the generate command DEPTH parameter will result in an additional screen shift giving the appearance of an out-of-screen depth shift for the generated image. A zero value (the default value) for DANG will result in a single image generated.

An example of this command is:

```
view(640, 400)
```

3.9 save(FILENAME)

This command will save a generated image to a file.

The FILENAME argument specifies a file to save the resulting image to. The file output will be in portable bit map (PBM) format. If the file name begins with a pipe ('|') then the output will be provided as a standard input stream to the specified process.

An example of this command is:

```
save(|pnmtojpeg -quality 90 > output.jpg)
```

3.10 generate(DEPTH)

This command will begin the generation of the planet, moon, ring or image.

This command must be placed at the end of a sequence of commands (but before exit of course). Each generate command will place the specified planet, moon, or image on the existing picture unless a 'reset' command is issued at the beginning of the sequence. If the DEPTH parameter is specified with a non-zero value the resulting image will be shifted. Note that the perceived depth as a result will only be apparent if a 3D image is generated (i.e., the view command DANG parameter is specified as a non-zero value). The left image is generated with a shift of -DEPTH/2 and the right image with a shift of +DEPTH/2.

4 Example

```
/* macro for Planet */
view 1600 1200
/* define map composed of stars */
'map(|ppmforge -night -stars 200 -saturation 250 -xsize 1600 -ysize 1200)'
/* generate backdrop image */
image
generate
/* define planet map */
'map(|jpegtopnm map/blue-map.jpg,full,40)'
light 202020 'z-25' y45 x35
/* generate planet */
planet 140 800 6 'z-20' y45 x15
generate
/* define ring map */
'map(|jpegtopnm map/brown-ring.jpg,full,0,100,010101)'
light 404040 'z-25' y45 x35
/* generate ring */
ring 220 340 100
generate
'save(|pnmtojpeg -quality 90 > blue.jpg)'
exit
```