1 Introduction

1.1 Background

R can provide more statistical graph. The core graphics system can been divided in to two main packages. The first package is the graphics package. It is older and it provides the original GRZ graphics system from S. It is fast and it can provide traditional graphics. The newer package is the grid package. It is actually slower but is has more power on flexibility and additional features than the graphics package.

The graph is drawn by using grid can been editing in many more ways than the graph been drawn on the basic graphics package. However, there is a new package, called gridGrahics, which allows us to convert the plot been draw by graphics package to grid graphics. So that the plot can been manipulated by any function on the grid package in order to provide more statistical plot.

The main function on the gridGrahics is called `grid.echo()`, which allows any plot been drawn in the current graphics device convert to the grid graphics. The following codes provide a quick example of the demonstration. We generate 25 random number of x and y. First, we draw a scatter plot by using the function plot() from the basic graphic package, then we redraw it by using the grid.echo() from the gridGraphic package with grid.

```
> x = runif(25)
> y = runif(25)
> plot(x,y, pch = 16)
> grid.echo()
```

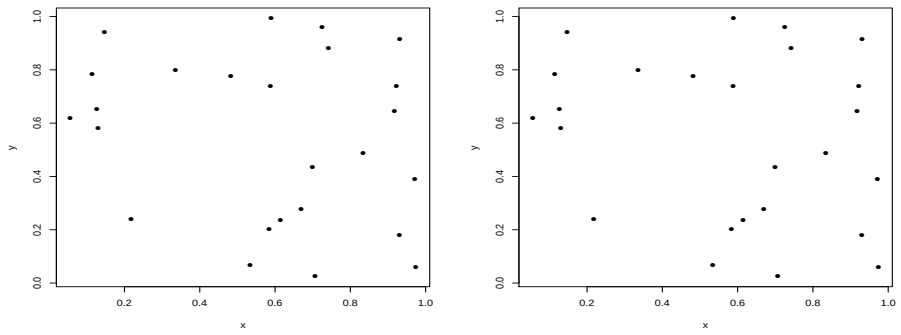

Figure 1: The left plot drawn by using plot(), the Right plot is redraw it by using `grid.echo()` on grid graphic system, overall, they are identical to each other

1.2 The problem

The `grid.echo()` can replicate most of plot that drawn by the graphics package. However, there are few functions on the graphics package that cannot replicate. One will the the `persp()` which for drawing 3-dimemtional surfaces on the basic graphics.

```
> x = seq(-10,10,length = 100)
> y = seq(-10,10,length = 100)
> f <- function(x, y) { r <- sqrt(x^2+y^2); 10 * sin(r)/r }
> z <- outer(x, y, f)
> z[is.na(z)] <- 1
> persp(x, y, z, theta = 20,phi = 30, expand = 0.5, box = TRUE)
> grid.echo()
Warning message:
In FUN(X[[i]], ...) : gridGraphics cannot emulate persp()
```

Figure 2: an example shows that the `grid.echo()` cannot replicate the `persp()` on grid

1.3 Aim of the project

This project has following aims: 1. replicate the persp plot on grid by translate the C code directly. 2. rewrite the translated R code in a more effifiency way that R behaved. 3. ...

2 Methodology

2.1 standalone

Drawing a 3-dimemsion object on a 2-dimemsion plane requires a transformation. The most important information is the trasnformation matrix. However, the `persp()` will return the transformation matrix for adding to the perspecive plots.

```
> x = y = seq(-10,10,length = 60)
> f = function(x, y) { r <- sqrt(x^2+y^2); 10 * sin(r)/r }
> z = outer(x, y, f); z[is.na(z)] <- 1
> trans <- persp(x, y, z, theta=120, phi = 20, expand = 0.5)
> trans
```

```
              [,1]        [,2]        [,3]        [,4]
[1,] -5.000000e-02 -0.02961981  0.08137977 -0.08137977
[2,]  8.660254e-02 -0.01710101  0.04698463 -0.04698463
[3,] -4.390837e-18  0.07781011  0.02832056 -0.02832056
[4,]  1.697562e-17 -0.30082519 -2.84154222  3.84154222
```

with this transformation matrix, we can now transform a 3-dimemsion points into 2-dimemsion points by multifying the 4D coordinates (x, y, z, 1) of points to the transformation matrix. Note that there is a function called `trans3d()` can do this computation easily.

```
> trans3d(1,1,1, trans)
```

```
$x
[1] 0.009933232

$y
[1] -0.07320118
```

The code will transform a 3-D point (1,1,1) into 2-D by using the transformation matrix.

Drawing a perspective plot is more likely that drawing a surface over the x-y plane. The surface can been decomposed as a finite number of polygons. These polygons will the the following properties: 1. Each polygons are Convex Polygon. (all its interior anges will less than 180 degrees.)

2. Each polygons will have 4 points.

The number of polygons required to been drawn can be calculated as follows: Suppose given the length of x, y is n and m, the function that defined the surface is f, i.e z = f(x, y). Then the length of z is n * m. The number of polygons to been drawn is (n - 1) * (m - 1) in total.

Suppose we have a set of coordinates x and y. where the length of x is n, the length of y is m, f is a surface function of x and y, then firstible we compute all possible combination of x and y, then evaluative the z coordinates with the function f by using the R function outer() so that we have a all 3-D points. The next step is transform the 3-D points into 2-D points by multifying a given transformation matrix. Then instead connecting the points line by line, we draw (n - 1) * (m - 1) number of polygons such that each polygons used 4 set of points.

There an issue that it can cause the problems, which is the drawing order. It is not a good idea to draw the points without figure out that the ordering of