

**First Semester 2018/2019 (A181)**

**Group B**

**Group Assignment:**

Application for 3D Coordinate Systems

**Student:**

Chai Sheng Loong 240378

Neoh Hai Liang 245807

Chia Wei Shin 246365

**Lecturer:**

Puan Noraziah Haji Man

**Submitted on:**

17 December 2018

**SQQM3034 Multivariate Calculus**

**Table of Content**

|  |  |  |
| --- | --- | --- |
| **No.** | **Title** | **Page Number** |
| 1.0 | Introduction | 1 |
| 2.0 | Rectangular Coordinate:  Imaginary Axis are Real | 2 – 3 |
| 3.0 | Cylindrical Coordinate:  HSB Colour Space Model | 4 – 5 |
| 4.0 | Spherical Coordinate:  Geographic Coordinate System | 6 – 8 |
| 5.0 | Conclusion | 9 |
|  | Reference | 10 |
|  | Appendix:  Source code for the *shinyapp* in R language | 11 – 15 |

**1.0 Introduction**

In Mathematics, any function with 2 variables (*x*, *y*) is able to plot into two-dimensional space with meaningful visualisation. There are various methods to plot the function in a plane, such as Cartesian Coordinate and Polar Coordinate. Based on Wikipedia (2018), Cartesian Coordinate is a coordinate system that locate each point of a function uniquely by specifying the distances from two perpendicular reference axes at the centre of origin. However, Polar Coordinate is a coordinate system which each point of a function is determined by specifying a direct distance from a reference pole and an angle from a reference direction.

In the three-dimensional space, it is only meaningful to visualise a particular function with 3 variables (*x*, *y*, *z*). There are also many types of coordinate system to plot the function, such as Rectangular Coordinate, Cylindrical Coordinate and Spherical Coordinate. In a simple word, Rectangular Coordinate System combines 2 Cartesian Plane to form a three-dimensional space with the distances from 3 perpendicular reference axes. In other hand, Cylindrical Coordinate System combines 1 Cartesian Plane and a Polar Plane, which form a three-dimensional space by specifying the distances from 2 perpendicular axes and an angle from the horizontal axis. Meanwhile, the Spherical Coordinate uses the combination of 2 polar planes, which a point is required to specify the angle from a vertical direction and the angle from horizontal direction and the direct distance from the pole.

Of course, these three-dimensional coordinate systems are interchangeable, the visualisation of the function is literally equivalent in these three coordinate systems. However, in term of Calculus, Polar Coordinate can simplify a very complicated function in Cartesian Coordinate, and integrate or differentiate in an easy manner.

Nowadays, there are many tools to visualise a three-dimensional graph, such as MATLAB, Mathematica, Maple and R programming language. In this project, we illustrate the 3D graph with using R programming language, because of the handy *rgl* package which are used to plot beautiful and interactive 3D graph. Besides, *shiny* package is also used to build the application in a Graphical User Interface (GUI) with interactive environment. The source code that we used to deploy the *shinyapp* in the following website is included the Appendix section:

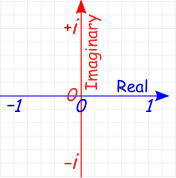
https://rexpert.shinyapps.io/Multivariate/

**2.0 Rectangular Coordinate: Imaginary Axis are Real**

Based on the information of Khan Academy, there are some quadratic equations that do not have any real number solutions in the study of mathematics. For example the quadratic equation, = 0, basically we will never be able to find a real number solution to that equation because it is impossible to square a real number and get a negative number. However, a solution to the equation = 0 does exist in a new number system called the complex number system or imaginary number, *i*.

Based on the information of Wolfram MathWorld, the imaginary number is the square root of , *i* = , which is commonly referred to as "*i*". Although there are two possible square roots of any number, the square roots of a negative number cannot be distinguished until one of the two is defined as the imaginary unit, at which point + *i* and - *i* can then be distinguished. For example, the square root of -16 is equal to 4 *i*. In short, imaginary number is very useful because we can solve things that need the square root of a negative number if we simply accepting that *i* exists.

Imaginary numbers are used in real-life applications, such as electricity, as well as quadratic equations. In quadratic planes, imaginary numbers show up in equations that don’t touch the x axis. Imaginary numbers become particularly useful in advanced calculus. Indeed, Imaginary numbers are particularly applicable in electricity, specifically alternating current (AC) electronics which AC electricity changes between positive and negative in a sine wave. Since combining AC currents can be very difficult because they may not match properly on the waves, therefore using imaginary currents and real numbers helps those working with AC electricity in order to do the calculations and avoid electrocution.



*Complex Plane*

In Mathematics, the complex plane is a geometric representation of the complex numbers established by the real axis and the perpendicular imaginary axis. Indeed, the complex plane is the plane of complex numbers spanned by the vectors 1 and *i*, where *i* is the imaginary number. Every complex number corresponds to a unique point in the complex plane and the line in the plane with *i* = 0 is also the real line. Every complex number can be expressed as a point in the complex plane as it is expressed in the form *a + bi* where *a* and *b* are real numbers where *a* described the real portion of the number and *b* describes the complex portion. Therefore, complex plane can used to identify the graph of the imaginary function from the quadratic equations, *y* =

*y* =

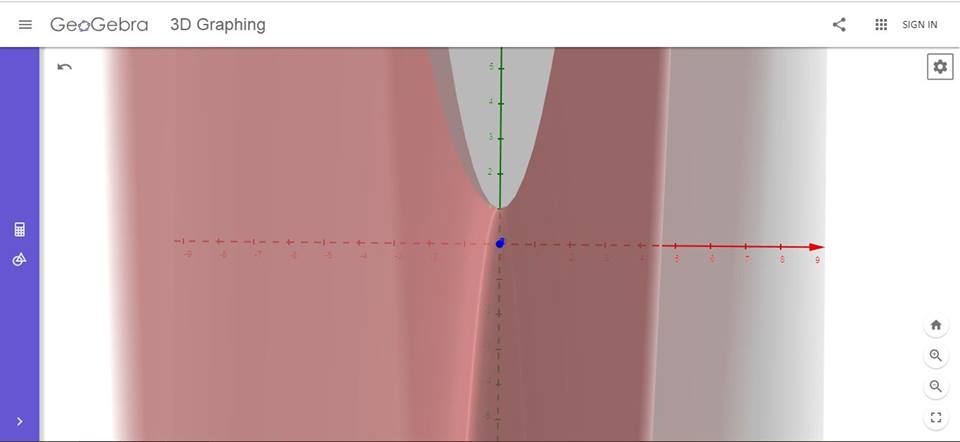
* *j*\**j* = (*ij*)(*ij*)

=

=

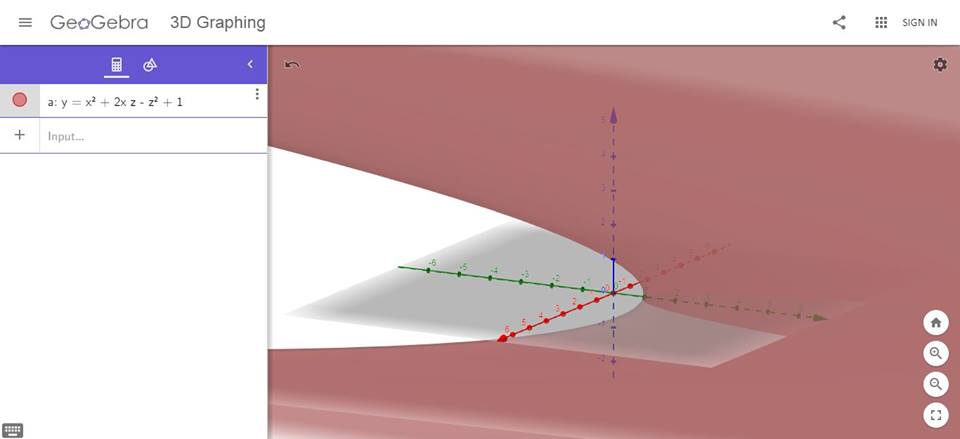
= + 2 *j*\**j* + 1

= + 2 + 1



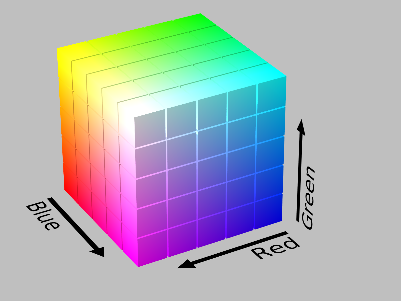
*This graph shows the curve of the quadratic equation, y =*

*This graph shows the hyperbolic paraboloid of the real-imaginary equation, y = + 2 + 1*



**3.0 Cylindrical Coordinate: HSB Colour Space Model**

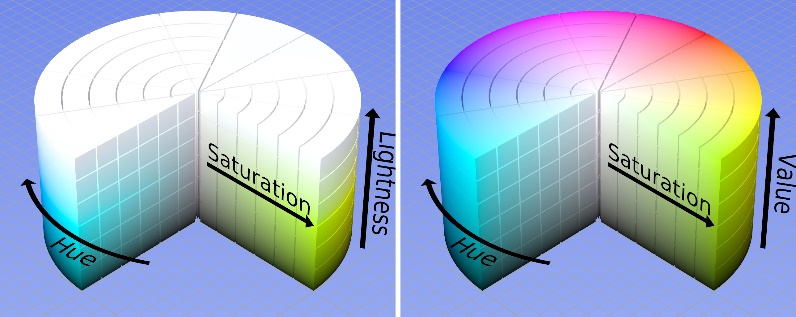
In this digital era, colour from the natural source can be captured to the digital devices, represent in the computer screen and print on a paper. However, within these processes, the selected colour is transformed via several colour space models, such as Red-Green-Blue (RGB) Colour Space Model, and Cyan-Magenta-Yellow-blacK (CMYK) Colour Space Model.



*Visualisation of RGB colour space model in Rectangular Coordinate System.*

RGB models is an additive colour model, which specifies the strength of Red, Green, and Blue colours, and project (add) those colours over the LED screen in the computer screen to present the desirable colour. While CMYK model is a subtractive colour model, which mix (subtract) the intensity specified from Cyan, Magenta, Yellow, and Black colours and print whatever the colour left in CMYK model.

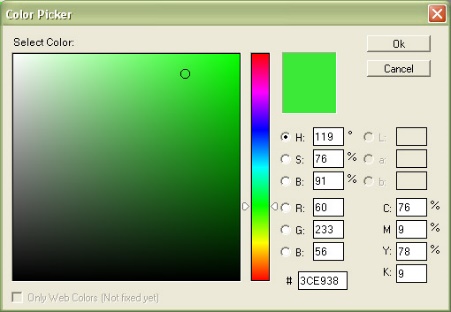
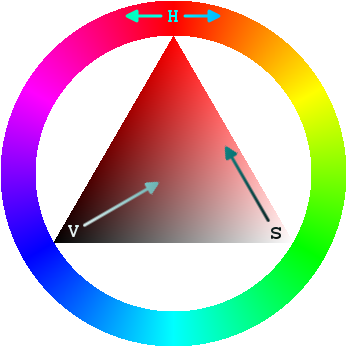
However, these models are not enough to for human to select the desirable colour from the screen in the “human way”. For example, human don’t analyse how much the amount of red, green, blue when choosing a colour. Thus, in 1938, M. Georges Valensi, a French telecommunication engineer, invented the Hue-Saturation-Lightness (HSL) colour space model, which allow the human to choose a colour and encode it into electronic devices in an intuitive manner. After 1970s, Alvy Ray Smith III, an American computer scientist, formally described the Hue-Saturation-Brightness (HSB) (as known as Hue-Saturation-Value) colour space model. This model was based more on how colours organised and conceptualised in human vision.



*Visualisation of HSL & HSB model in Cylindrical Coordinate System*

People can use HSB colour space model to select the colour in a very short time, because this model stores the colours’ attributes in the “human way”. For example, people usually recognise the colour by its hue (Which colour is this?), its saturation (How strong the colour is?), and its brightness (How bright the colour is?).

Thus, at the horizontal plane, the colour is presented in a circle with red primary colour at 0° or 360°, green primary colour at 120° and blue colour at 240°. Notice that these primary colours in RGB model are equally distributed through out of the circle. Besides, the Saturation of the colour is the radius from the centre of the circle, where the strength of the colour is increase as the horizontal distance from centre increases. The saturation takes value from 0% to 100%. In addition, the Brightness is measured as the vertical length in the cylinder model, which the colour’s brightness is increasing when the vertical coordinate of the colour is elevating. The value of brightness also bounded in 0% to 100%.



*Colour picker tools used in GIMP and Adobe Photoshop*

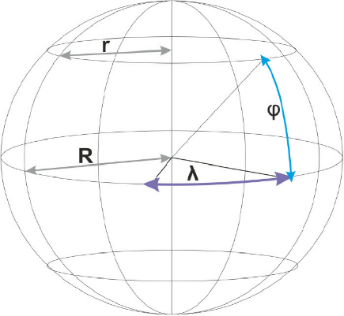
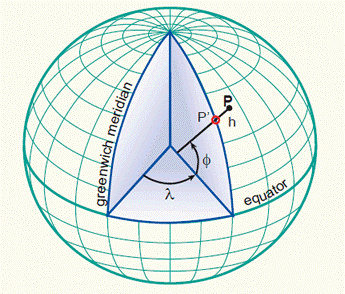
Since graphic designers prefer to select colour by using the HSB Colour Space model as an alternative of the RGB Colour Space model, hence many graphic design software provide HSB colour selection tools, such as Adobe’s series graphic editing applications, GIMP, Pixia and others. The implementation of HSB colour selection tools allow the artists choose the colours very quickly.

In a nutshell, the HSB colour space applies a bounded Cylindrical Coordinates to present the coordinates of every visible colour. The angle from the horizontal direction (*θ*) is represented by the hue of the colour, the distance from the pole (*r*) is represented by the saturation of the colour and the vertical elevation of the coordinates (*z*) is described by the brightness of the colour. Thus, by summing up all information, every colour in the HSB Colour Space model is uniquely plotted.

**4.0 Spherical Coordinate: Geographic Coordinate System**

Longitude and latitude are imaginary lines that circle the Earth. In geographic coordinate system, Latitude and Longitude are the units that represent the coordinates, which are used to exactly identify the location of a person or an object. These lines are located on maps and measured in degrees and each degree is divided into 60 minutes.

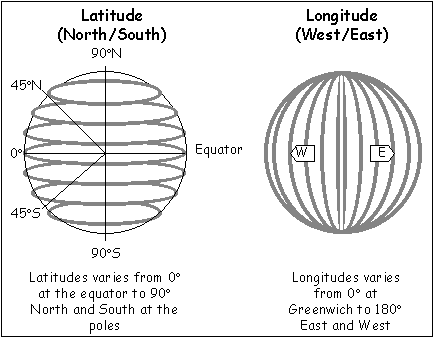
The latitude has the symbol of phi, *φ* and it indicates the angle between the straight line in the certain point and the equatorial plane. The latitude is starting from 0° and ending up with 90° to both sides of the equator, which formed of latitude Northern and Southern. The equator is the line with 0° latitude. The symbol of lambda *λ* in longitude is another angular coordinate describing the position of a point on a surface of earth. The longitude is defined as an angle pointing West or East from the Greenwich Meridian, which is also known as the Prime Meridian.



*Visualisation of the Geographic Coordinate System*

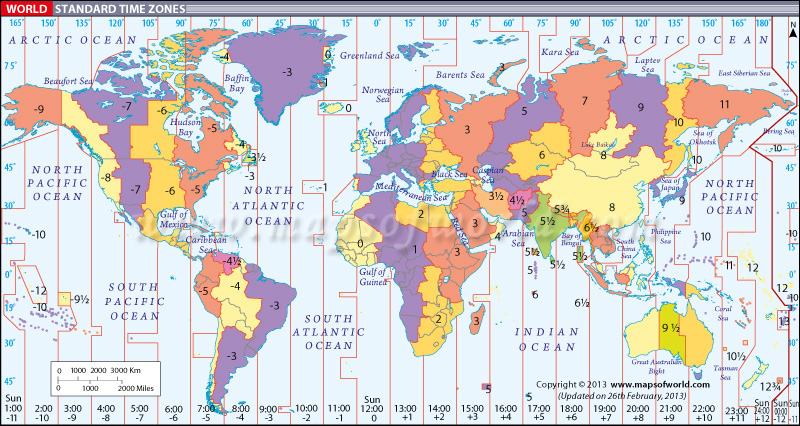
Latitude describes east-west lines that circle the Earth parallel to the equator. Any point along the equator has a latitude of 0°. While the equator is the longest of these lines, the farther you move to the poles, the shorter they are. The equator divides the earth into a northern and southern hemisphere.

Longitude shows the lines that run from north to south across the globe. Greenwich, England is the starting point for this system (0°). When England was leading in creating maps and navigating, prime meridian was chosen at a time. Locations west of prime meridian have a longitude of up to 179° west, however those to the east have a longitude of up to 179° east. At 180° from meridian is exactly on the opposite side of the earth, which is actually in the middle of the Pacific Ocean, which is known as the International Date Line. Tourists who travel across the International Date Line will experience the dateline change to the following day.



*The illustration of Latitudes and Longitudes*

Since the earth rotates around its axis once a day, it is divided into 24 different time zones. The starting point is from the prime meridian in Greenwich, also termed Greenwich Mean Time. Theoretically, every 15° to the east or west of Greenwich means approximately one hour. In practice, however, time zones do not go through the middle of countries. Therefore, each country will standardize their time zone. Hence, the larger countries that span thousands of km, for example Russia or the United States, have several time zones for several cities. On the other hand, the Chinese Communists changed the country to one time zone when they took over after World War II.



*Time zones distributed over countries on the world map*

Malaysian Standard Time or Malaysian Time with the short form of MYT is a standard time used in Malaysia. It is 8 hours ahead of Greenwich Mean Time. The local mean time in Kuala Lumpur was originally GMT+06:46:46. Peninsular Malaysia used this local mean time until the year of 1880, when they changed to Singapore mean time GMT+06:55:25. Between the end of the World War II and the formation of Malaysia on 16 September 1963, it was known as British Malayan Standard Time, which was a UTC +07:30 hour ahead of Greenwich Mean Time. At 2330 hrs local time of 31 December 1981, people in Peninsular Malaysia adjusted their clocks and watches ahead by 30 minutes to become 00:00 hours local time of 1 January 1982, to match the time in use in East Malaysia, which is UTC+08:00. SGT (Singapore) as follow on the same until now.

**5.0 Conclusion**

All in all, three-dimensional space can be present in many ways, such as Rectangular Coordinates, Cylindrical Coordinates, and Spherical Coordinate Systems. In our project, we presented the application of these coordinate systems. In Rectangular Coordinate System, the imaginary-real space can be visualised by the combination of complex (*x-j*) plane and the real (*x-y*) plane. The imaginary roots can be visualised by plotting this 3D space.

Besides, the HSB colour space model also can described in the Cylindrical Coordinate System. The attributes of the colours are stored as hue, saturation, and brightness, which can be plotted in a closed cylinder. In addition, any place on the Earth surface can be located through the Geographic Coordinate System, which is the application of Spherical Coordinate System.

As we know, these coordinate systems have its own application, hence it contributes the equal importance to every field of study, such as physics, mathematics, geography, and even colorology (a scientific study on the properties of colour). Hence, we should appreciate every knowledge that we learnt in the classroom and try to apply it on the problem in real situation. This will enhance our beautiful world with any new knowledge that we might contribute.

**Reference**

1. Halm, T. (2015, October 6). Polar, Spherical and Geographic Coordinates. Retrieved December 14, 2018, from https://vvvv.org/blog/polar-spherical-and-geographic-coordinates
2. Leonawicz, M. (2017, September 29). Examples using plot3D and rgl with Shiny. Retrieved December 15, 2018, from https://uasnap.shinyapps.io/plot3D/
3. Rosmanitz, K. (2015, August 24). Latitude, Longitude and Time Zones. Retrieved December 13, 2018, from https://www.english-online.at/geography/longitude-latitude/longitude-latitude-time-zones.htm
4. Russoft. (2007, March 01). Plotting x^2 in the complex axes. Retrieved December 15, 2018, from https://www.youtube.com/watch?v=Yk9RSdjlBPs
5. Welch Labs. (2015, August 28). Imaginary Numbers Are Real [Part 1: Introduction]. Retrieved December 13, 2018, from https://youtu.be/T647CGsuOVU
6. Wikipedia. (2018, November 28). HSL and HSV. Retrieved December 14, 2018, from https://en.wikipedia.org/wiki/HSL\_and\_HSV
7. Wikipedia. (2018, December 12). Geographic coordinate system. Retrieved December 13, 2018, from https://en.wikipedia.org/wiki/Geographic\_coordinate\_system
8. Wikipedia. (2018, November 24). Time in Malaysia. Retrieved December 13, 2018, from <https://en.wikipedia.org/wiki/Time_in_Malaysia>
9. Imaginary Unit. (n.d.). Wolfram MathWorld. Retrieved from: http://mathworld.wolfram.com/ImaginaryUnit.html
10. Intro to the imaginary numbers. (2015). Khan Academy. Retrieved from: https://www.khanacademy.org/math/algebra2/introduction-to-complex-numbers-algebra-2/the-imaginary-numbers-algebra-2/a/intro-to-the-imaginary-numbers

**Appendix: Source code for the *shinyapp* in R language**

library**(**shiny**)**

library**(**shinydashboard**)**

library**(**rgl**)**

library**(**lubridate**)**

header **<-** dashboardHeader**(**

title **=** textOutput**(**"title"**)**

**)**

sidebar **<-** dashboardSidebar**(**

sidebarMenu**(**

id **=** "tabs",

menuItem**(**"Rectangular Coordinate", tabName **=** "rec", icon **=** icon**(**"th"**))**,

menuItem**(**"Cylinder Coordinate", tabName **=** "cyl", icon **=** icon**(**"database"**))**,

menuItem**(**"Spherical Coordinate", tabName **=** "sph", icon **=** icon**(**"globe"**))**

**)**

**)**

body **<-** dashboardBody**(**

tabItems**(**

tabItem**(**"rec",

tags**$**style**(**".col-sm-3 {height: 68vh} .well {height: 100%}"**)**,

fluidRow**(**

valueBox**(**

tagList**(**tags**$**h3**(**

style **=** "font-style: italic;font-size: 50px;

text-align: center;font-family: \"Computer Modern\"",

HTML**(**"y = x<sup>2</sup> + 2xj - j<sup>2</sup> + 1"**)))**,

"", width **=** 12, color **=** "green"**)**

**)**,

sidebarLayout**(**

sidebarPanel**(**

width **=** 3,

titlePanel**(**"Sample View Point"**)**,

br**()**,br**()**,br**()**,

actionButton**(**"viewPointA", "A", width **=** "100%"**)**,

br**()**,br**()**,br**()**,

actionButton**(**"viewPointB", "B", width **=** "100%"**)**,

br**()**,br**()**,br**()**,

actionButton**(**"reset", "Reset", width **=** "100%"**)**

**)**,

mainPanel**(**

width **=** 9,

rglwidgetOutput**(**"surface", height **=** "68vh", width **=** "100%"**)**

**)**

**))**,

tabItem**(**"cyl",

fluidRow**(**

valueBoxOutput**(**"htmlColour", width **=** 12**)**

**)**,

fluidRow**(**

box**(**

width **=** 4,

height **=** "65vh",

sliderInput**(**"hue", "Hue:",min **=** 0, max **=** 360,value **=** 0**)**,

sliderInput**(**"sat", "Saturation:",min **=** 0, max **=** 100,value **=** 0**)**,

sliderInput**(**"bright", "Brightness:",min **=** 0, max **=** 100,value **=** 0**)**

**)**,

box**(**width **=** 8,

rglwidgetOutput**(**"plot", width **=** "100%", height **=** "62vh"**)**

**)**

**)**

**)**,

tabItem**(**"sph",

fluidRow**(**

valueBoxOutput**(**"coor", width **=** 12**)**

**)**,

sidebarLayout**(**

sidebarPanel**(**

tags**$**style**(**".col-sm-4 {height: 68vh} .well {height: 100%}"**)**,

tags**$**head**(**tags**$**style**(**HTML**(**"

.shiny-split-layout > div {overflow: visible;}"**)))**,

tags**$**style**(**'.mySpan { font-weight: bold; font-size: 20px; }'**)**,

tags**$**span**(**"Latitude", class **=** "mySpan"**)**,

splitLayout**(**

cellWidths **=** c**(**"65%", "35%"**)**,

textInput**(**"lat", "", placeholder **=** "Range: 0 - 90"**)**,

selectInput**(**"dirLat", "", choices **=** list**(**"N" **=** "N", "S" **=** "S"**))**

**)**,

tags**$**span**(**"Longitude", class **=** "mySpan"**)**,

splitLayout**(**

cellWidths **=** c**(**"65%", "35%"**)**,

textInput**(**"long", "", placeholder **=** "Range: 0 - 180"**)**,

selectInput**(**"dirLong", "", choices **=** list**(**"W" **=** "W", "E" **=** "E"**))**

**)**,

br**()**,br**()**,

splitLayout**(**

cellWidths **=** c**(**"65%", "35%"**)**,"",

actionButton**(**"submit", "GO!", width **=** "100%"**)**

**)**

**)**,

mainPanel**(**

rglwidgetOutput**(**"globe", width **=** "100%", height **=** "68vh"**)**

**)**

**)**

**)**

**)**

**)**

server **<-** **function(**input, output**)** **{**

output**$**title **<-** renderText**({**

switch **(**

input**$**tabs,

"rec" **=** "Imaginary Axis",

"cyl" **=** "HSB Colour Space",

"sph" **=** "Globe Coordinate"**)**

**})**

# Rectangular Coordinate -----------------------------------------

val **<-** reactiveValues**(**theta **=** 60,phi **=** 10**)**

observeEvent**(**input**$**viewPointA, **{**

val**$**theta **<-** 0

val**$**phi **<-** **-**90

**})**

observeEvent**(**input**$**viewPointB, **{**

val**$**theta **<-** 275

val**$**phi **<-** 5

**})**

observeEvent**(**input**$**reset, **{**

val**$**theta **<-** 60

val**$**phi **<-** 10

**})**

output**$**surface **<-** renderRglwidget**({**

open3d**(**useNULL **=** T**)**

rgl.viewpoint**(**theta **=** val**$**theta, phi **=** val**$**phi, zoom **=** 1**)**

rgl.bg**(**color **=** "white"**)**

long **<-** 3

x **<-** seq**(-**long, long, length.out **=** 50**)**

y **<-** seq**(-**long, long, length.out **=** 50**)**

z **<-** outer**(**x,y, **function(**x,y**)** **{**

x**^**2 **+** 2**\***x**\***y **-** y**^**2 **+** 1

**})**

persp3d**(**x,y,z, alpha **=** 0.6, xlab **=** "x", ylab **=** "j", zlab **=** "y", axes **=** F**)**

box3d**()**

lim **<-** **function(**x**){**c**(-**max**(**abs**(**x**))**, max**(**abs**(**x**)))** **\*** 1.1**}**

xlim **<-** lim**(**x**)**; ylim **<-** lim**(**y**)**; zlim **<-** lim**(**z**)**

rgl.lines**(**xlim, c**(**0, 0**)**, c**(**0, 0**)**, color **=** "red"**)**

rgl.lines**(**c**(**0, 0**)**, ylim, c**(**0, 0**)**, color **=** "blue"**)**

rgl.lines**(**c**(**0, 0**)**, c**(**0, 0**)**, zlim, color **=** "green"**)**

rglwidget**()**

**})**

# Cylindrical Coordinate -----------------------------------------

output**$**htmlColour **<-** renderValueBox**({**

theta **<-** input**$**hue **\*** pi **/** 180

radius **<-** input**$**sat **/** 100

high **<-** input**$**bright **/** 100

colr **<-** hsv**(**theta **/** **(**2**\***pi**)**, radius, high**)**

# Select black font colour when the condition met

hb **<-** theta **>** 5 **\*** pi **/** 18 **&** theta **<** pi

sb **<-** radius **<** 0.5

bb **<-** high **>** 0.7

fontColor **<-** ifelse**(**bb **&** **(**hb **|** **(!**hb **&** sb**))**,"#000000", "#FFFFFF"**)**

tgs **<-** paste**(**"font-weight: bold;font-size: 50px;

text-align: center;color: "

, fontColor**)**

display **<-** paste0**(**"HSB(", input**$**hue, "&deg;, ",

input**$**sat, ", " , input**$**bright, ")"**)**

sty **<-** paste0**(**".small-box.bg-black { background-color: ",

colr, "!important; }"**)**

valueBox**(**

tagList**(**tags**$**h3**(**style **=** tgs, HTML**(**display**))**,tags**$**style**(**sty**))**,

"", width **=** 12, color **=** "black"**)**

**})**

output**$**plot **<-** renderRglwidget**({**

open3d**(**useNULL **=** T**)**

rgl.viewpoint**(**theta **=** 0, phi **=** **-**80, zoom **=** 1**)**

rgl.bg**(**color **=** "white"**)**

theta **<-** input**$**hue **\*** pi **/** 180

radius **<-** input**$**sat **/** 100

high **<-** input**$**bright **/** 100

xcoor **<-** radius **\*** cos**(**theta**)**

ycoor **<-** radius **\*** sin**(**theta**)**

zcoor **<-** high **\*** 5

sphColour **<-** hsv**(**theta **/** **(**2**\***pi**)**, radius, high**)**

x **<-** rep**(**0,6**)**

y **<-** rep**(**0,6**)**

z **<-** seq**(**0,5**)**

cyl **<-** cylinder3d**(**cbind**(**x,y,z**)**, sides **=** 20**)**

plot3d**(**cyl, type **=** "wire", box **=** F, axes **=** F,

xlab **=** "", ylab **=** "", zlab **=** ""**)**

spheres3d**(**xcoor, ycoor, zcoor, radius **=** 0.1, color **=** sphColour**)**

rglwidget**()**

**})**

# Spherical Coordinate -----------------------------------------

val1 **<-** reactiveValues**(**

lat **=** 0,

long **=** 0,

dirLat **=** "N",

dirLong **=** "W"

**)**

observeEvent**(**input**$**submit, **{**

val1**$**lat **<-** ifelse**(!**is.na**(**as.numeric**(**input**$**lat**))**, as.numeric**(**input**$**lat**)**, 0**)**

val1**$**dirLat **<-** input**$**dirLat

val1**$**long **<-** ifelse**(!**is.na**(**as.numeric**(**input**$**long**))**,

as.numeric**(**input**$**long**)**, 0**)**

val1**$**dirLong **<-** input**$**dirLong

**})**

output**$**coor **<-** renderValueBox**({**

greenwichTime **<-** as.POSIXct**(**Sys.Date**())** **+** 4**\***60**\***60

currentTime **<-** switch **(**

val1**$**dirLong,

"E" **=** greenwichTime **+** val1**$**long **\*** 4 **\*** 60,

"W" **=** greenwichTime **-** val1**$**long **\*** 4 **\*** 60

**)**

currentTime **<-** ifelse**(**currentTime **==** Sys.Date**()** **|**

currentTime **==** Sys.Date**()** **+** 1,

"00:00",

gsub**(**"^(.\*\\s)|(:00)$", "", currentTime**))**

clock24 **<-** as.numeric**(**gsub**(**":.\*$", "", currentTime**))**

dayNight12 **<-** ifelse**(**clock24 **>=** 12, "pm", "am"**)**

clock12 **<-** ifelse**(**clock24 **>** 12, clock24 **-** 12, clock24**)**

currentTime **<-** paste0**(**clock12, gsub**(**"^.\*:", ":", currentTime**)**,

" ", dayNight12**)**

bg **<-** "#000000"

font **<-** "#FFFFFF"

**if(**clock24 **<** 7**)** **{**

# night

bg **<-** "#0a1340"

font **<-** "#6e73ae"

**}** **else** **if** **(**clock24 **<** 11**)** **{**

# morning

bg **<-** "#1895fd"

font **<-** "#a8fffd"

**}** **else** **if** **(**clock24 **<** 14**)** **{**

# noon

bg **<-**"#ffd61b"

font **<-** "#fcffb5"

**}** **else** **if** **(**clock24 **<** 17**)** **{**

# afternoon

bg **<-** "#f8aa27"

font **<-** "#fff8b6"

**}** **else** **if** **(**clock24 **<** 19**)** **{**

# evening

bg **<-** "#4d136d"

font **<-** "#d2a8ff"

**}** **else** **if** **(**clock24 **<** 24**)** **{**

# night

bg **<-** "#0a1340"

font **<-** "#6e73ae"

**}**

display **<-** HTML**(**paste0**(**val1**$**lat, "&deg; ", val1**$**dirLat, " ", val1**$**long,

"&deg; ", val1**$**dirLong, ", ", currentTime**))**

style **<-** paste0**(**"font-weight: bold;font-size: 50px;

text-align: center;color: ", font**)**

sty **<-** paste0**(**".small-box.bg-aqua { background-color: ",

bg, "!important; }"**)**

valueBox**(**tagList**(**tags**$**h3**(**style **=** style, display**)**, tags**$**style**(**sty**))**,

"", width **=** 12**)**

**})**

output**$**globe **<-** renderRglwidget**({**

lat **<-** matrix**(**seq**(**90, **-**90, len **=** 50**)\***pi**/**180, 50, 50, byrow **=** **TRUE)**

long **<-** matrix**(**seq**(-**180, 180, len **=** 50**)\***pi**/**180, 50, 50**)**

r **<-** 6378.1 # radius of Earth in km

x **<-** r**\***cos**(**lat**)\***cos**(**long**)**

y **<-** r**\***cos**(**lat**)\***sin**(**long**)**

z **<-** r**\***sin**(**lat**)**

open3d**(**useNULL **=** T**)**

rgl.viewpoint**(**theta **=** 23.5,phi **=** **-**70, zoom **=** 0.8**)**

persp3d**(**x, y, z, col **=** "white",

texture **=** system.file**(**"textures/world.png", package **=** "rgl"**)**,

type **=** "wire",specular **=** "black", axes **=** F, box **=** F,

xlab **=** "", ylab **=** "", zlab **=** "",

normal\_x **=** x, normal\_y **=** y, normal\_z **=** z**)**

longitude **<-** val1**$**long # [-180,180]

latitude **<-** val1**$**lat # [-90, 90]

directionLat **<-** val1**$**dirLat

directionLong **<-** val1**$**dirLong

phi **<-** switch **(**directionLat,

"N" **=** **(**90 **-** latitude**)** **\*** pi **/** 180,

"S" **=** **(**90 **+** latitude**)** **\*** pi **/** 180

**)**

rLat **<-** r **\*** sin**(**phi**)**

elevation **<-** switch **(**directionLat,

"N" **=** sqrt**(**r**^**2 **-** rLat**^**2**)**,

"S" **=** **-**sqrt**(**r**^**2 **-** rLat**^**2**)**

**)**

th **<-** seq**(**0, 2**\***pi, len **=** 201**)**

xr **<-** rLat **\*** cos**(**th**)**

yr **<-** rLat **\*** sin**(**th**)**

lines3d**(**xr, yr, elevation, col **=** "red", lwd **=** 2**)**

theta **<-** switch **(**directionLong,

"E" **=** longitude **\*** pi **/** 180,

"W" **=** **-(**longitude **\*** pi **/** 180**))**

fi **<-** seq**(-**pi**/**2, pi**/**2, len **=** 201**)**

xr1 **<-** r **\*** cos**(**fi**)** **\*** cos**(**theta**)**

yr1 **<-** r **\*** cos**(**fi**)** **\*** sin**(**theta**)**

zr1 **<-** r **\*** sin**(**fi**)**

lines3d**(**xr1, yr1, zr1, col **=** "red", lwd **=** 2**)**

rglwidget**()**

**})**

**}**

shinyApp**(**

ui **=** dashboardPage**(**header, sidebar, body**)**,

server **=** server

**)**