If you follow the #rstats news on Twitter, you have most likely seen the impressive 3D maps produced with Tyler Morgan-Wall’s [rayshader](https://www.rayshader.com/) package. I’ve been wanting to try it for a long time and I finally found the time to do it last week. As a limnologist living on the shores of Lake Geneva, I didn’t hesitate for long about the location of my first map, which, in the excitement of the moment, I immediately published on twitter.

Finally I found time to dive into [@tylermorganwall](https://twitter.com/tylermorganwall?ref_src=twsrc%5Etfw) rayshader [#rstats](https://twitter.com/hashtag/rstats?src=hash&ref_src=twsrc%5Etfw) ! To start, here is a 3D rendering of Lake Geneva and my workplace [@UmrCarrtel](https://twitter.com/UmrCarrtel?ref_src=twsrc%5Etfw). Stay tuned, there's a blog post and more exciting stuffs to come! [pic.twitter.com/3VpPcYIdO8](https://t.co/3VpPcYIdO8)

— Francois Keck (@FrancoisKeck) [September 26, 2019](https://twitter.com/FrancoisKeck/status/1177284292025733122?ref_src=twsrc%5Etfw)

In this tweet I was promising to go deeper into the subject. That’s the purpose of this post.

First, we load some packages, functions and data. I have prepared a geotiff file for the region of interest (the surroundings of Lake Geneva). The elevation data were extracted from the [SRTM 30 m dataset](http://dwtkns.com/srtm30m/) (thanks [@tylermorganwall](https://twitter.com/tylermorganwall) for the advice!) and I have completed the DEM with a low-resolution model of the lake bathymetry. There is a very high-resolution bathymetry available for lake Geneva, but sadly not released in open data. Finally, we will also need a simple shapefile of the lake shoreline.

library(raster)

library(magrittr)

library(stringr)

library(rgl)

library(rayshader)

library(magick)

library(ggplot2)

source("<https://gist.github.com/fkeck/4820db83b9ff2fbf1f7fe901563ddc82/raw/>")

# Some useful functions by Will Bishop from his post on rayshader

# <https://wcmbishop.github.io/rayshader-demo/>

source("<https://raw.github.com/wcmbishop/rayshader-demo/master/R/rayshader-gif.R>")

mnt <- raster("combined\_leman.tif")

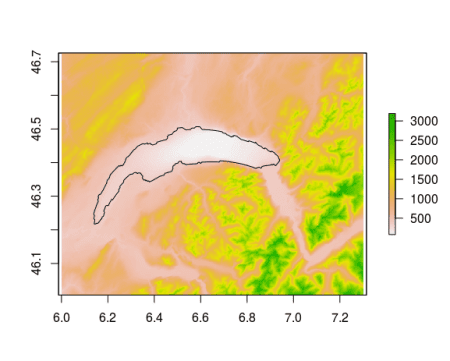
mnt <- aggregate(mnt, 2) # Simplification factor

ccv <- rgdal::readOGR("ccv.shp")

plot(mnt)

plot(ccv, add = T)

export\_folder <- "/home/francois/Documents/rayshader"



Now, we are ready to render our first 3D scene! The rayshader API makes the task extremely easy. Starting from the elevation matrix, we successively add shadow layers and print the result on screen with OpenGL.

#### Static rendering ####

elmat <- matrix(raster::extract(mnt, raster::extent(mnt), buffer = 1000),

nrow = ncol(mnt), ncol = nrow(mnt))

z\_scale <- 40

ambmat <- ambient\_shade(elmat)

raysha <- ray\_shade(elmat, zscale = z\_scale, maxsearch = 300)

hillshade <- elmat %>%

sphere\_shade(texture = "imhof1", sunangle = 35) %>%

add\_shadow(raysha, 0.5) %>%

add\_shadow(ambmat, 0.5)

plot\_3d(hillshade, elmat, zscale = z\_scale, windowsize = c(1400, 620))

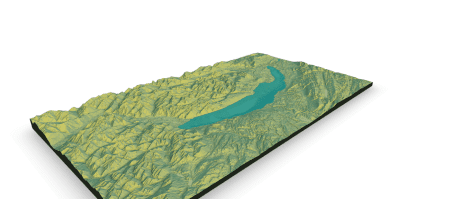
render\_camera(theta = 135, phi = 25, zoom = 0.4, fov = 75)

render\_water(elmat, zscale = z\_scale, wateralpha = 0.7,

watercolor = "turquoise4", waterdepth = 372,

waterlinecolor = "lightblue", waterlinealpha = 0.5)

render\_snapshot(file.path(export\_folder, "static\_leman.png"))



Ok, now that we have 3D, what about adding a 4th dimension with time? I have to admit that it was the animations made with rayshader that impressed me the most in the first place. The coolest thing was probably [this video of Monterey Bay](https://www.tylermw.com/wp-content/uploads/2018/07/montbayaboveweb2.mp4) showing water level progressively increasing and decreasing in an infinite, hypnotizing loop. I definitely needed to have the same thing done with the lake I see every day by my window.

#### Water filling animation ####

list.files(export\_folder, pattern = "render\_", full.names = TRUE) %>%

file.remove()

n\_frames <- 300

thetas <- transition\_values(from = -45, to = -225, steps = n\_frames)

water\_level <- transition\_values(from = 372, to = 0, steps = n\_frames)

plot\_3d(hillshade, elmat, zscale = z\_scale, windowsize = c(1400, 620))

for(i in 1:n\_frames){

file\_img\_i <- file.path(export\_folder, paste0("render\_", sprintf("%04d", i), ".png"))

render\_camera(theta = thetas[i], phi = 25, zoom = 0.4, fov = 75)

render\_water(elmat, zscale = z\_scale, wateralpha = 0.7,

watercolor = "turquoise4", waterdepth = water\_level[i])

render\_snapshot(file\_img\_i)

}

paste0('cd ', export\_folder, ' && ffmpeg -y -r 25 -i render\_%04d.png',

' -c:v libx264 -vf "fps=25,format=yuv420p" filling\_movie.mp4') %>%

system()

<http://www.pieceofk.fr/wp-content/uploads/2019/10/filling_movie.mp4>

The camera movement reinforces the 3D perception but I am a little disappointed by the « filling » of the lake. The low resolution of the bathymetry gives a poor result here.

So far, I have simply adapted bits of code found here and there. Now, it is time to try something a bit more exciting. For that I will need new data. My colleague and friend Frédéric Soulignac has developed a [3D hydrodynamic model](https://www.researchgate.net/publication/325853327_Using_3D_modeling_and_remote_sensing_capabilities_for_a_better_understanding_of_spatio-temporal_heterogeneities_of_phytoplankton_abundance_in_large_lakes) for the lake temperature and he is kind enough to share some of his simulation outputs for my little experiments. My idea is to project the simulated surface temperatures onto the lake surface. Fred provided several days of data, so I have the opportunity to make a nice animation.

Technically, I use a height-field surface shape added to the 3D scene with the rgl.surface function. The surface is flat, the height is fixed at the altitude of the lake (372m), locations are colored according to the temperature data and the height of the points outside the lake is set to NA, so that they are not drawn. It is also important to set the lit argument on FALSE to disable the calculation of light effects on this object. To complete the graph, we will add some elements: the date and time, a color scale bar and a temperature histogram. I didn’t try to integrate these elements directly into the 3D scene. Instead, I added them after the rendering, using the magic drawing functions of the [magick](https://cran.r-project.org/web/packages/magick/vignettes/intro.html) package.

#### Temperature animation ####

# Load the temperature data. These are raster objects imported from Matlab.

# We need to adjust projection and resolution to fit with mnt

load("/home/francois/Documents/fred\_temp.RData")

temp\_interp\_crop <- lapply(temp\_interp\_list, function(x) {

projectRaster(x, mnt) %>% extend(mnt)

})

# We prepare the color palette

all\_temps <- do.call(rbind, temp\_list)$temperature

pal <- viridisLite::viridis(100)

pal <- viridisLite::inferno(100)

pal\_cut <- seq(min(all\_temps), max(all\_temps), by = (max(all\_temps) - min(all\_temps))/100)

names(pal) <- levels(cut(all\_temps, pal\_cut))

# Color matrix for the surface

colmat\_list <- lapply(temp\_interp\_crop, function(temp\_interp\_crop){

colmat <- matrix(raster::extract(temp\_interp\_crop, raster::extent(temp\_interp\_crop), buffer = 1000),

ncol = nrow(temp\_interp\_crop), nrow = ncol(temp\_interp\_crop))

colmat <- matrix(pal[cut(colmat, pal\_cut)], nrow = nrow(colmat), ncol = ncol(colmat))

return(colmat)

})

# Color height for the surface

z\_height <- matrix(372, nrow = nrow(colmat\_list[[1]]), ncol = ncol(colmat\_list[[1]]))

z\_height[which([is.na](http://is.na)(colmat\_list[[1]]))] <- NA

n\_frames <- 500

thetas <- transition\_values(from = -45, to = -225, steps = n\_frames)

zoom <- transition\_values(from = 0.6, to = 0.3, steps = n\_frames)

temp\_idx <- transition\_values(from = 1, to = length(colmat\_list),

steps = n\_frames, type = "lin", one\_way = TRUE) %>% round(0)

# We load a color scale bar picture in memory

img\_scale <- image\_graph(width = 195, height = 50)

par(mar = c(3, 0, 0, 0))

col\_bar\_h(pal, min(all\_temps), max(all\_temps), n.tick = 2)

dev.off()

plot\_3d(hillshade, elmat, zscale = z\_scale, windowsize = c(1400, 620))

list.files(export\_folder, pattern = "render\_", full.names = TRUE) %>%

file.remove()

for(i in seq\_len(n\_frames)){

file\_img\_i <- file.path(export\_folder, paste0("render\_", sprintf("%04d", i), ".png"))

render\_camera(theta = thetas[i], phi = 25, zoom = zoom[i], fov = 75)

rgl.surface((1:nrow(colmat\_list[[temp\_idx[i]]])),

(1:ncol(colmat\_list[[temp\_idx[i]]])) - ncol(colmat\_list[[temp\_idx[i]]]),

z\_height/z\_scale,

color = colmat\_list[[temp\_idx[i]]], back = "lines", lit = FALSE)

render\_snapshot2(file\_img\_i)

rgl.pop()

img\_histo <- image\_graph(width = 300\*0.8, height = 150\*0.8)

print(

temp\_list[[temp\_idx[i]]] %>%

ggplot() +

geom\_histogram(aes(temperature, fill = cut(temperature, pal\_cut)), bins = 30) +

geom\_hline(yintercept = 0, size = 0.7) +

scale\_color\_manual(values = pal) +

scale\_fill\_manual(values = pal) +

theme\_void() +

theme(legend.position = "none") +

xlim(16, 24)

)

dev.off()

img <- image\_read(file\_img\_i)

img <- image\_composite(img, img\_scale, offset = "+1160+50")

img <- image\_composite(img, img\_histo, offset = "+20+10")

img <- image\_draw(img, xlim = c(0, image\_info(img)[["width"]]), ylim = c(0, image\_info(img)[["height"]]))

text(1260, 580, expression(paste("Temperature (", degree, "C)")), cex = 1.2)

text(625, 565, str\_split(temp\_date\_list[[temp\_idx[i]]], " ", simplify = TRUE)[1], cex = 2, adj = 0.5)

text(775, 565, str\_split(temp\_date\_list[[temp\_idx[i]]], " ", simplify = TRUE)[2], cex = 2, adj = 0.5)

dev.off()

image\_write(img, path = file\_img\_i)

print(i)

}

paste0('cd ', export\_folder, ' && ffmpeg -y -r 25 -i render\_%04d.png',

' -c:v libx264 -vf "fps=25,format=yuv420p" temp\_movie2.mp4') %>%

system()

<http://www.pieceofk.fr/wp-content/uploads/2019/10/temp_movie_sq-1.mp4>

[HD version](http://www.pieceofk.fr/wp-content/uploads/2019/10/temp_movie2.mp4)