```
Program generate_moon
   use, intrinsic :: iso_fortran_env, only: int32, real64
   implicit none

   integer(int32), parameter :: width = 1024
   integer(int32), parameter :: height = 768
   real(real64), parameter   :: pi = 3.14159265358979323846_real64

   character(len=32) :: loc_name(2)
   real(real64)      :: lat(2), lon(2), elev(2)
   integer(int32)    :: i

   loc_name(1) = "MINGA"
   lat(1) = 48.173333_real64
   lon(1) = 11.635000_real64
   elev(1) = 515.0_real64

   loc_name(2) = "GAGGERS"
   lat(2) = 48.333333_real64
   lon(2) = 11.250000_real64
   elev(2) = 500.0_real64

   do i = 1, 2
      call render_moon_image(trim(loc_name(i)), lat(i), lon(i), elev(i))
   end do

contains

   subroutine render_moon_image(location_name, latitude, longitude, elevation)
      character(len=*), intent(in) :: location_name
      real(real64), intent(in)     :: latitude, longitude, elevation

      character(len=64)   :: filename_ppm
      character(len=64)   :: filename_raw_png
      character(len=64)   :: filename_final_png
      character(len=4096) :: sys_cmd
      character(len=8)    :: dt_date, dt_time
      character(len=5)    :: dt_zone
      character(len=2048) :: overlay_text, overlay_text_sun
      character(len=128)  :: line_loc, line_zeit, line_phase
      character(len=128)  :: line_entf, line_az, line_alt
      character(len=128)  :: line_sun_az, line_sun_alt
      integer(int32)      :: img(width, height, 3)
      integer(int32)      :: x, y, cx, cy, radius, fov_radius
      real(real64)        :: dx, dy_up, dist_fov, noise, brightness, wave
```

```
46
47        real(real64)      :: alt, az, phase_pct, dist_km
48        real(real64)      :: angle_sun_deg, angle_sun_rad
49        real(real64)      :: az_sun, alt_sun, sun_dx_deg, sun_dy_deg
50        real(real64)      :: lunar_eclipse_factor, sun_px_x, sun_px_y
51        real(real64)      :: dist_sun, dist_moon, sep_sun_moon_px, sun_coverage
52        real(real64)      :: sky_bright, red_factor, dimming, corona_factor
53        integer(int32)    :: sky_r, sky_g, sky_b
54        real(real64)      :: u_sun_x, u_sun_y, u_perp_x, u_perp_y
55        real(real64)      :: x_prime, y_prime, z_prime, k_phase
56        real(real64)      :: sin_phi, cos_phi, dot_light
57        real(real64)      :: scale_deg_per_px, alt_pixel
58        character(len=32) :: phase_str
59        integer           :: year, month, day, hour, minute, second
60
61        filename_final_png = lowercase(location_name)//".png"
62        filename_raw_png = lowercase(location_name)//"_raw.png"
63        filename_ppm = lowercase(location_name)//".ppm"
64
65        ! Systemzeit und Zeitzone auslesen
66        call date_and_time(date=dt_date, time=dt_time, zone=dt_zone)
67        read (dt_date(1:4), '(I4)') year
68        read (dt_date(5:6), '(I2)') month
69        read (dt_date(7:8), '(I2)') day
70        read (dt_time(1:2), '(I2)') hour
71        read (dt_time(3:4), '(I2)') minute
72        read (dt_time(5:6), '(I2)') second
73
74        ! year  = 1999
75        ! month = 8
76        ! day   = 11
77        ! hour  = 12
78
79        ! Datum und Uhrzeit exakt an Python uebergeben
80        write (sys_cmd, '(A,F9.6,A,F9.6,A,F6.1,A,I4.4,A,I2.2,A,I2.2,A,I2.2,A,I2.2,A,I2.2,A,A,A)') &
81           "python3 /var/www/html/moon/get_moon.py ", &
82           latitude, " ", longitude, " ", elevation, " ", &
83           year, " ", month, " ", day, " ", &
84           hour, " ", minute, " ", second, " ", dt_zone, &
85           " > moon_tmp.txt"
86        call execute_command_line(trim(sys_cmd))
87
88        open (unit=20, file="moon_tmp.txt", status="old", action="read")
89        read (20, *) az, alt, dist_km, phase_pct, phase_str, &
90           angle_sun_deg, az_sun, alt_sun, sun_dx_deg, sun_dy_deg, &
91           lunar_eclipse_factor
```

```
      close (20)
      call execute_command_line("rm moon_tmp.txt")

      angle_sun_rad = angle_sun_deg*pi/180.0_real64
      u_sun_x = sin(angle_sun_rad)
      u_sun_y = cos(angle_sun_rad)
      u_perp_x = cos(angle_sun_rad)
      u_perp_y = -sin(angle_sun_rad)

      k_phase = max(0.001_real64, min(0.999_real64, phase_pct/100.0_real64))
      cos_phi = 2.0_real64*k_phase - 1.0_real64
      sin_phi = sqrt(1.0_real64 - cos_phi**2)

      ! Massstab: Sichtfeld-Durchmesser = 1.0 Grad (560 Pixel)
      ! Mond/Sonne Radius = 140 Pixel (0.5 Grad Durchmesser)
      fov_radius = 280
      radius = 140
      scale_deg_per_px = 1.0_real64/(2.0_real64*real(fov_radius, real64))

      ! Position der Sonne im Bild (in Pixeln relativ zum Zentrum)
      sun_px_x = sun_dx_deg*(2.0_real64*real(fov_radius, real64))
      sun_px_y = sun_dy_deg*(2.0_real64*real(fov_radius, real64))

      ! Sonnenfinsternis-Abdeckung berechnen
      sep_sun_moon_px = sqrt(sun_px_x**2 + sun_px_y**2)
      if (sep_sun_moon_px < 280.0_real64) then
         sun_coverage = max(0.0_real64, min(1.0_real64, 1.0_real64 - sep_sun_moon_px/280.0_real64))
      else
         sun_coverage = 0.0_real64
      end if

      img = 0
      cx = width/2
      cy = height/2 + 30

      do y = 1, height
         dy_up = real(cy - y, real64)
         alt_pixel = alt + dy_up*scale_deg_per_px

         do x = 1, width
            dx = real(x - cx, real64)
            dist_fov = sqrt(dx*dx + dy_up*dy_up)

            if (dist_fov <= real(fov_radius, real64)) then
               ! Wasserflaeche bei Alt < 0
               if (alt_pixel < 0.0_real64) then
```

```
                    wave = sin(dx*0.1_real64)*cos(dy_up*0.2_real64)*4.0_real64
                    img(x, y, 1) = int(min(255.0_real64, max(0.0_real64, 8.0_real64 + wave)))
                    img(x, y, 2) = int(min(255.0_real64, max(0.0_real64, 28.0_real64 + wave*1.5_real64)))
                    img(x, y, 3) = int(min(255.0_real64, max(0.0_real64, 55.0_real64 + wave*2.0_real64)))
                 else
                    ! Himmelsfarbe je nach Sonnenstand und Sonnenfinsternis
                    if (alt_sun > 0.0_real64) then
                       sky_bright = min(1.0_real64, max(0.0_real64, alt_sun/15.0_real64))*(1.0_real64 - 0.9_real64*sun_coverage)
                       sky_r = int(12.0_real64 + sky_bright*80.0_real64)
                       sky_g = int(15.0_real64 + sky_bright*130.0_real64)
                       sky_b = int(22.0_real64 + sky_bright*200.0_real64)
                    else
                       sky_r = 12
                       sky_g = 15
                       sky_b = 22
                    end if

                    img(x, y, 1) = sky_r
                    img(x, y, 2) = sky_g
                    img(x, y, 3) = sky_b

                    ! Distanzen zu Sonne und Mond
                    dist_moon = dist_fov
                    dist_sun = sqrt((dx - sun_px_x)**2 + (dy_up - sun_px_y)**2)

                    ! 1. SONNENSCHEIBE / KORONA RENDERN
                    if (dist_sun <= real(radius, real64)) then
                       img(x, y, 1) = 255
                       img(x, y, 2) = 245
                       img(x, y, 3) = 210
                    else if (sun_coverage > 0.7_real64 .and. dist_sun <= real(radius, real64) + 60.0_real64) then
                       ! Korona bei hoher/totaler Finsternis
                       corona_factor = (1.0_real64 - (dist_sun - real(radius, real64))/60.0_real64)*sun_coverage
                       img(x, y, 1) = int(min(255.0_real64, sky_r + corona_factor*180.0_real64))
                       img(x, y, 2) = int(min(255.0_real64, sky_g + corona_factor*180.0_real64))
                       img(x, y, 3) = int(min(255.0_real64, sky_b + corona_factor*200.0_real64))
                    end if

                    ! 2. MONDSCHEIBE VOR DIE SONNE / NORMALE DARSTELLUNG RENDERN
                    if (dist_moon <= real(radius, real64)) then
                       y_prime = dx*u_sun_x + dy_up*u_sun_y
                       x_prime = dx*u_perp_x + dy_up*u_perp_y

                       if (x_prime**2 + y_prime**2 <= real(radius, real64)**2) then
                          z_prime = sqrt(real(radius, real64)**2 - x_prime**2 - y_prime**2)
                          dot_light = (y_prime*sin_phi + z_prime*cos_phi)/real(radius, real64)
```

```
  184
  185                        if (dot_light > 0.0_real64) then
  186                           noise = sin(dx*0.15_real64)*cos(dy_up*0.15_real64) + sin(dx*0.05_real64 + dy_up*0.08_real64)
  187                           brightness = (160.0_real64 + noise*40.0_real64)*sqrt(dot_light)
  188
  189                           ! Mondfinsternis (Blutmond-Einfaerbung)
  190                           if (lunar_eclipse_factor > 0.05_real64) then
  191                              red_factor = min(1.0_real64, lunar_eclipse_factor)
  192                              dimming = 1.0_real64 - 0.7_real64*red_factor
  193                              img(x, y, 1) = int(min(255.0_real64, max(0.0_real64, &
  194                                                                  brightness*dimming*(1.0_real64 + 0.5_real64*red_factor))
))
  195                              img(x, y, 2) = int(min(255.0_real64, max(0.0_real64, &
  196                                                                  brightness*dimming*(1.0_real64 - 0.6_real64*red_factor))
))
  197                              img(x, y, 3) = int(min(240.0_real64, max(0.0_real64, &
  198                                                                  brightness*dimming*(1.0_real64 - 0.8_real64*red_factor))
))
  199                           else
  200                              img(x, y, 1) = int(min(255.0_real64, max(0.0_real64, brightness)))
  201                              img(x, y, 2) = int(min(255.0_real64, max(0.0_real64, brightness)))
  202                              img(x, y, 3) = int(min(240.0_real64, max(0.0_real64, brightness*0.95_real64)))
  203                           end if
  204                        else
  205                           ! Schattenseite des Mondes
  206                           if (sun_coverage > 0.5_real64) then
  207                              ! Bei Sonnenfinsternis ist die Mondvorderseite eine dunkle Silhouette
  208                              img(x, y, 1) = 5
  209                              img(x, y, 2) = 5
  210                              img(x, y, 3) = 8
  211                           else if (lunar_eclipse_factor > 0.5_real64) then
  212                              ! Totale Mondfinsternis Kernschatten-Rot
  213                              img(x, y, 1) = 45
  214                              img(x, y, 2) = 12
  215                              img(x, y, 3) = 8
  216                           else
  217                              ! Normaler Erdschein
  218                              img(x, y, 1) = 18
  219                              img(x, y, 2) = 21
  220                              img(x, y, 3) = 30
  221                           end if
  222                        end if
  223                     end if
  224                  end if
  225               end if
  226            end if
```

```
227         end do
228      end do
229
230      open (unit=10, file=filename_ppm, status='replace', action='write', form='formatted')
231      write (10, '(A)') 'P3'
232      write (10, '(I0, 1X, I0)') width, height
233      write (10, '(A)') '255'
234      do y = 1, height
235         do x = 1, width
236            write (10, '(I0, 1X, I0, 1X, I0)') img(x, y, 1), img(x, y, 2), img(x, y, 3)
237         end do
238      end do
239      close (10)
240
241      sys_cmd = "pnmtopng "//trim(filename_ppm)//" > "//trim(filename_raw_png)//" && rm "//trim(filename_ppm)
242      call execute_command_line(sys_cmd)
243
244      write (line_loc, '(A,A,A,F9.6,A,F9.6,A)') &
245         "STANDORT: ", trim(location_name), " (", latitude, " N, ", longitude, " E)"
246
247      write (line_zeit, '(A,I2.2,A,I2.2,A,I2.2,A,I2.2,A,I2.2,A,I4)') &
248         "ZEIT: ", hour, ":", minute, ":", second, " CET, ", day, ".", month, ".", year
249
250      write (line_phase, '(A,A,A,F5.1,A)') &
251         "MONDPHASE: ", trim(phase_str), " (", phase_pct, "%)"
252
253      write (line_entf, '(A,I6,A)') &
254         "ENTF.: ", int(dist_km), " km"
255
256      write (line_az, '(A,F5.1,A)') &
257         "AZIMUT: ", az, " deg"
258
259      write (line_alt, '(A,F5.1,A)') &
260         "HOEHE: ", alt, " deg"
261
262      overlay_text = "SIMULATION: MOND-ANSICHT ("//trim(location_name)//")"//char(10)//char(10)// &
263                     trim(line_loc)//char(10)// &
264                     trim(line_zeit)//char(10)// &
265                     trim(line_phase)//char(10)// &
266                     trim(line_entf)//char(10)// &
267                     trim(line_az)//char(10)// &
268                     trim(line_alt)//char(10)// &
269                     "FIELD: 1.0 deg"
270
271      write (line_sun_az, '(A,F5.1,A)') "AZIMUT: ", az_sun, " deg"
272      write (line_sun_alt, '(A,F5.1,A)') "HOEHE:  ", alt_sun, " deg"
```

```
273
274       overlay_text_sun = "SONNE:"//char(10)// &
275                          trim(line_sun_az)//char(10)// &
276                          trim(line_sun_alt)
277
278       sys_cmd = "convert "//trim(filename_raw_png)// &
279                 " -font /usr/share/fonts/truetype/dejavu/DejaVuSansMono.ttf"// &
280                 " -pointsize 16 -fill white"// &
281                 " -annotate +30+40 '"//trim(overlay_text)//"'"// &
282                 " -annotate +30+570 '"//trim(overlay_text_sun)//"' "// &
283                 trim(filename_final_png)//" && rm "//trim(filename_raw_png)
284
285       call execute_command_line(sys_cmd)
286
287    end subroutine render_moon_image
288
289    pure function lowercase(str) result(res)
290       character(len=*), intent(in) :: str
291       character(len=len(str))      :: res
292       integer                      :: j, code
293       do j = 1, len(str)
294          code = iachar(str(j:j))
295          if (code >= 65 .and. code <= 90) then
296             res(j:j) = achar(code + 32)
297          else
298             res(j:j) = str(j:j)
299          end if
300       end do
301    end function lowercase
302
303 end program generate_moon
```