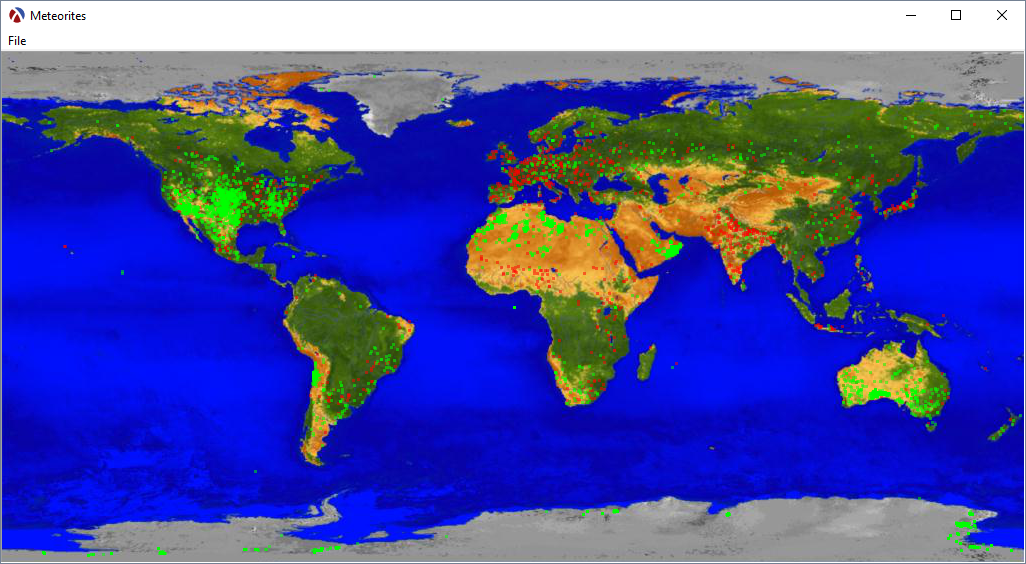
Jeremy Diamond PA3

For PA3 I opted for the meteorite map example.

**Part 1**

After installing racket and downloading the relevant files I began running the code as is. It yielded the following image.

This seems to be identical to the intended state of the code.

**Part 2**

I began from by attempting to draw the latitude and longitude lines. Modeling this off of the point drawing code provided, and making use of the [racket drawing documentation](https://docs.racket-lang.org/draw/index.html) , I came up with the following code in main.

;; Draw lines

(for ((lat (in-range -90.0 90.0 10.0)))

(define-values (x1 y1) (lat-long->x-y canvas lat -180))

(define-values (x2 y2) (lat-long->x-y canvas lat 180))

(send canvas-dc set-pen "gray" 1 'solid)

(send canvas-dc set-alpha 0.5)

(send canvas-dc draw-line x1 y1 x2 y2)

(yield))

(for ((long (in-range -180.0 180.0 10.0)))

(define-values (x1 y1) (lat-long->x-y canvas -90 long))

(define-values (x2 y2) (lat-long->x-y canvas 90 long))

(send canvas-dc set-pen "gray" 1 'solid)

(send canvas-dc set-alpha 0.5)

(send canvas-dc draw-line x1 y1 x2 y2)

(yield))

(define-values (x1 y1) (lat-long->x-y canvas -90 0))

(define-values (x2 y2) (lat-long->x-y canvas 90 0))

(send canvas-dc set-pen "gray" 3 'solid)

(send canvas-dc set-alpha 0.5)

(send canvas-dc draw-line x1 y1 x2 y2)

(yield)

(define-values (x3 y3) (lat-long->x-y canvas 0 -180))

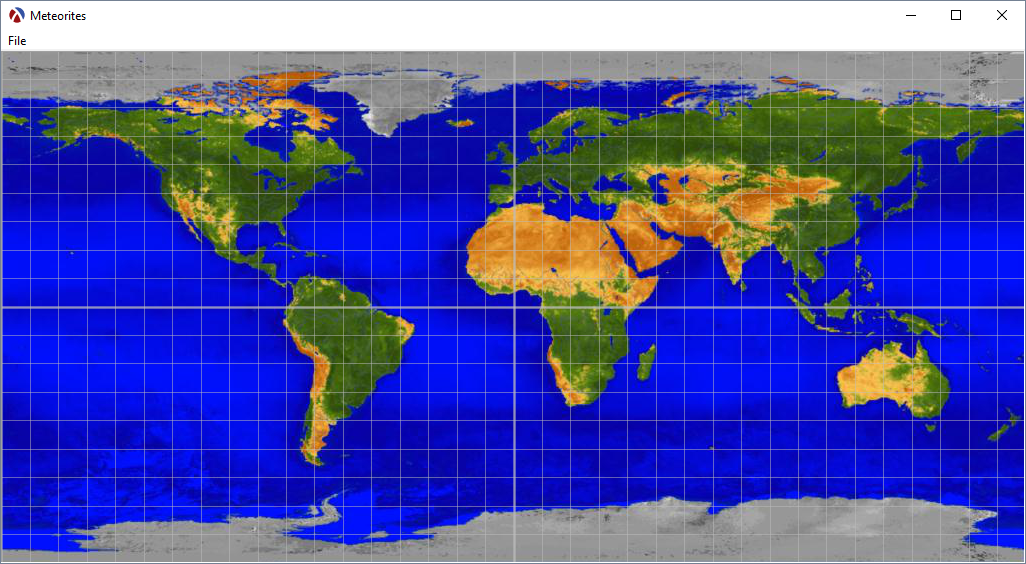
(define-values (x4 y4) (lat-long->x-y canvas 0 180))

(send canvas-dc set-pen "gray" 3 'solid)

(send canvas-dc set-alpha 0.5)

(send canvas-dc draw-line x3 y3 x4 y4)

(yield)



That code (removing the meteorite parsing for testing needs) generated the following image.

**Part 3**

To alter the size of the drawn object on a log scale I defined the following function. It first checks if the string “mass” can be converted into a number. If it passes then the log of that number is evaluated and if not it defaults to 3. If the number is 0 or the log of the number is less than 3 the function returns a 3. Otherwise it returns the log of the mass. Note all of the default 3’s were returned with the statement (+ 1 2). I couldn't figure out a better way to do this.

(define (mass->size mass)

(if (string->number mass)

(cond

[(= (string->number mass) 0) (+ 1 2)]

[(< (log (string->number mass)) 3) (+ 1 2)]

[else (log (string->number mass))]

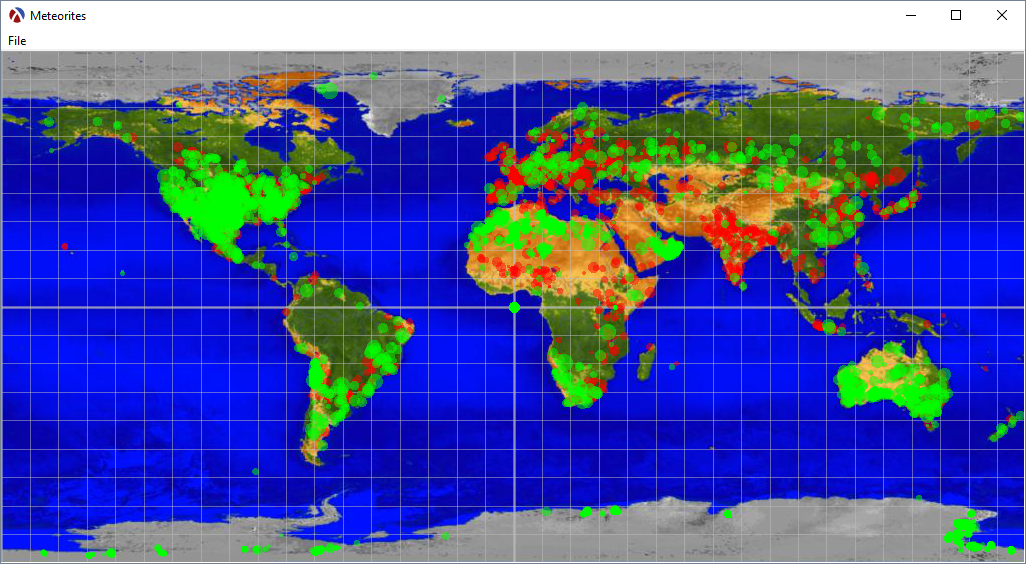
)

(+ 1 2)

)

)

Replacing the default size setting with a call to this function generated the following image.



Appendix

Diamond-meteorites.rkt

#lang racket/gui

(require csv-reading)

;;; Filenames

(define map-filename "BigEarth.jpg")

(define meteorites-filename "meteorite-landings.csv")

;;; Minimum and maximum latitude and longitude values

(define-values (lat-min lat-max) (values -90.0 90.0))

(define-values (long-min long-max) (values -180.0 180.0))

;;; Some data counts

(define fell-n (make-parameter 0))

(define found-n (make-parameter 0))

(define other-n (make-parameter 0))

(define invalid-n (make-parameter 0))

(define nomatch-n (make-parameter 0))

;;; (lat-long->x-y canvas lat long) -> (values real? real?)

;;; canvas : (is-a?/c canvas%

;;; lat : (real-in -90.0 90.0)

;;; long : (real-in -180.0 180.0)

;;; Returns the (x, y) coordinates corresponding to the given lat and long.

(define/contract (lat-long->x-y canvas lat long)

(-> (is-a?/c canvas%) (real-in -90.0 90.0) real? ; (real-in -180.0 180.0)

(values real? real?))

(define width (send canvas get-width))

(define height (send canvas get-height))

(values (\* width (/ (- long long-min) (- long-max long-min)))

(- height (\* height (/ (- lat lat-min) (- lat-max lat-min))) 1)))

;;; (fall->color fall) -> string?

;;; fall : string?

;;; Returns the color used to render a specified fall value. Also increments

;;; the data count dynamic variables.

(define/contract (fall->color fall)

(-> string? string?)

(case fall

(("Fell")

(fell-n (+ (fell-n) 1)) ; Increment fell count

"red")

(("Found")

(found-n (+ (found-n) 1)) ; Increment found count

"green")

(else

(other-n (+ (other-n) 1)) ; Increment other count

"white")))

(define (mass->size mass)

(if (string->number mass)

(cond

[(= (string->number mass) 0) (+ 1 2)]

[(< (log (string->number mass)) 3) (+ 1 2)]

[else (log (string->number mass))]

)

(+ 1 2)

)

)

;;; (main) -> any

(define (main)

;;; Initialize data count dynamic variables.

(parameterize ((fell-n 0)

(found-n 0)

(other-n 0)

(invalid-n 0)

(nomatch-n 0))

;; Get the device context for the canvas.

(define canvas-dc (send canvas get-dc))

;; Load the world map bitmap.

(define map (make-object bitmap% 1024 512))

(send map load-file map-filename)

(yield) ; Wait for the load to complete

(send canvas-dc draw-bitmap map 0 0)

;; Draw lines

(for ((lat (in-range -90.0 90.0 10.0)))

(define-values (x1 y1) (lat-long->x-y canvas lat -180))

(define-values (x2 y2) (lat-long->x-y canvas lat 180))

(send canvas-dc set-pen "gray" 1 'solid)

(send canvas-dc set-alpha 0.5)

(send canvas-dc draw-line x1 y1 x2 y2)

(yield))

(for ((long (in-range -180.0 180.0 10.0)))

(define-values (x1 y1) (lat-long->x-y canvas -90 long))

(define-values (x2 y2) (lat-long->x-y canvas 90 long))

(send canvas-dc set-pen "gray" 1 'solid)

(send canvas-dc set-alpha 0.5)

(send canvas-dc draw-line x1 y1 x2 y2)

(yield))

(define-values (x1 y1) (lat-long->x-y canvas -90 0))

(define-values (x2 y2) (lat-long->x-y canvas 90 0))

(send canvas-dc set-pen "gray" 3 'solid)

(send canvas-dc set-alpha 0.5)

(send canvas-dc draw-line x1 y1 x2 y2)

(yield)

(define-values (x3 y3) (lat-long->x-y canvas 0 -180))

(define-values (x4 y4) (lat-long->x-y canvas 0 180))

(send canvas-dc set-pen "gray" 3 'solid)

(send canvas-dc set-alpha 0.5)

(send canvas-dc draw-line x3 y3 x4 y4)

(yield)

;; Parse the meteorite landings file and skip the first row.

(define parsed-meteorite-landings

(csv->list (file->string meteorites-filename)))

(define meteroite-landings (cdr parsed-meteorite-landings))

(printf "There are ~s meteorite landings in file ~s.~n"

(length meteroite-landings) meteorites-filename)

;; Iterate over all the meteorite landings and put them on the map.

(for ((landing (in-list meteroite-landings)))

(match landing

((list name id nametype recclass mass fall year reclat reclong GeoLocation)

(define lat (string->number reclat))

(define long (string->number reclong))

(cond ((and lat long)

; (writeln (string->number mass))

(send canvas-dc set-pen (fall->color fall) (mass->size mass) 'solid)

(send canvas-dc set-alpha 0.5)

(define-values (x y) (lat-long->x-y canvas lat long))

(send canvas-dc draw-point x y)

(yield))

(else

(invalid-n (+ (invalid-n) 1)))))

(\_

(nomatch-n (+ (nomatch-n) 1))

(void))))

;; Print the data counts.

(printf "Fell = ~a~n" (fell-n))

(printf "Found = ~a~n" (found-n))

(printf "Other = ~a~n" (other-n))

(printf "Invalid = ~a~n" (invalid-n))

(printf "Nomatch = ~a~n" (nomatch-n))))

;;; Graphical Elements

(define frame

(instantiate frame%

("Meteorites")))

(define menu-bar

(instantiate menu-bar%

(frame)))

(define file-menu

(instantiate menu%

("&File" menu-bar)))

(define exit-menu-item

(instantiate menu-item%

("E&xit" file-menu)

(callback

(lambda (menu-item event)

(send frame show #f)))))

(define canvas

(instantiate canvas%

(frame)

(style '(border))

(min-width 1024)

(min-height 512)))

(send frame show #t)

(main)