Serenity-Climate-Change Model

Adapted from:

- Tinker, R. and Wilensky, U. (2007). NetLogo Climate Change model. http://ccl.northwestern.edu/netlogo/models/ClimateChange. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

```
globals [
  sky-top      ;; y coordinate of top row of sky
  earth-top    ;; y coordinate of top row of earth
  temperature  ;; overall temperature
  slow         ;; slow down counter
]

breed [rays ray]     ;; packets of sunlight
breed [IRs IR]       ;; packets of infrared radiation
breed [heats heat]   ;; packets of heat energy
breed [CO2s CO2]     ;; packets of carbon dioxide

breed [clouds cloud]
clouds-own [cloud-speed cloud-id]

;;
;; Setup Procedures
;;

to setup
  clear-all
  set-default-shape rays "ray"
  set-default-shape IRs "ray"
  set-default-shape clouds "cloud"
  set-default-shape heats "dot"
  set-default-shape CO2s "CO2-molecule"
  setup-world
  set temperature 12
  reset-ticks
end

to setup-world
  set sky-top max-pycor - 5
  set earth-top 0
  ask patches [  ;; set colors for the different sections of the world
    if pycor > sky-top [  ;; space
      set pcolor scale-color white pycor 22 15
    ]
    if pycor <= sky-top and pycor > earth-top [ ;; sky
      set pcolor scale-color blue pycor -20 20
    ]
    if pycor < earth-top
      [ set pcolor red + 3 ] ;; earth
    if pycor = earth-top ;; earth surface
      [ update-albedo ]
  ]
end

;;
;; Runtime Procedures
;;

to go
  ask clouds [ fd cloud-speed ]  ; move clouds along
  run-sunshine   ;; step sunshine
  ;; if the albedo slider has moved update the color of the "earth surface" patches
  ask patches with [pycor = earth-top]
    [ update-albedo ]
  add-CO2;; New Code adds CO2 to the model based upon an input
  remove-CO2;; New Code removes CO2 from the model based upon an input
  run-heat  ;; step heat
  run-IR    ;; step IR
  run-CO2   ;; moves CO2 molecules
  tick
end

to update-albedo ;; patch procedure
  set pcolor scale-color green albedo 0 1
end

to add-cloud            ;; erase clouds and then create new ones, plus one
  let sky-height sky-top - earth-top
  ;; find a random altitude for the clouds but
  ;; make sure to keep it in the sky area
  let y earth-top + (random-float (sky-height - 4)) + 2
  ;; no clouds should have speed 0
  let speed (random-float 0.1) + 0.01
  let x random-xcor
  let id 0
  ;; we don't care what the cloud-id is as long as
  ;; all the turtles in this cluster have the same
  ;; id and it is unique among cloud clusters
  if any? clouds
  [ set id max [cloud-id] of clouds + 1 ]

  create-clouds 3 + random 20
  [
    set cloud-speed speed
    set cloud-id id
    ;; all the cloud turtles in each larger cloud should
    ;; be nearby but not directly on top of the others so
    ;; add a little wiggle room in the x and ycors
    setxy x + random 9 - 4
          ;; the clouds should generally be clustered around the
          ;; center with occasional larger variations
          y + 2.5 + random-float 2 - random-float 2
    set color white
    ;; varying size is also purely for visualization
    ;; since we're only doing patch-based collisions
    set size 2 + random 2
    set heading 90
  ]
end

to remove-cloud       ;; erase clouds and then create new ones, minus one
  if any? clouds [
    let doomed-id one-of remove-duplicates [cloud-id] of clouds
    ask clouds with [cloud-id = doomed-id]
      [ die ]
  ]
end

to run-sunshine
  ask rays [
    if not can-move? 0.3 [ die ]  ;; kill them off at the edge
    fd 0.3                        ;; otherwise keep moving
  ]
  create-sunshine  ;; start new sun rays from top
  reflect-rays-from-clouds  ;; check for reflection off clouds
  encounter-earth   ;; check for reflection off earth and absorption
end
```

```
to create-sunshine
  ;; don't necessarily create a ray each tick
  ;; as brightness gets higher make more
  if 10 * sun-brightness > random 50 [
    create-rays 1 [
      set heading 160
      set color yellow
      ;; rays only come from a small area
      ;; near the top of the world
      setxy (random 10) + min-pxcor max-pycor
    ]
  ]
end

to reflect-rays-from-clouds
 ask rays with [any? clouds-here] [   ;; if ray shares patch with a cloud
   set heading 180 - heading   ;; turn the ray around
 ]
end

to encounter-earth
  ask rays with [ycor <= earth-top] [
    ;; depending on the albedo either
    ;; the earth absorbs the heat or reflects it
    ifelse 100 * albedo > random 100
      [ set heading 180 - heading  ] ;; reflect
      [ rt random 45 - random 45 ;; absorb into the earth
        set color red - 2 + random 4
        set breed heats ]
  ]
end

to run-heat    ;; advances the heat energy turtles
  ;; the temperature is related to the number of heat turtles
  set temperature 0.99 * temperature + 0.01 * (12 + 0.1 * count heats)
  ask heats
  [
    let dist 0.5 * random-float 1
    ifelse can-move? dist
      [ fd dist ]
      [ set heading 180 - heading ] ;; if we're hitting the edge of the world, turn around
    if ycor >= earth-top [  ;; if heading back into sky
      ifelse temperature > 20 + random 40
              ;; heats only seep out of the earth from a small area
```

```
             ;; this makes the model look nice but it also contributes
             ;; to the rate at which heat can be lost
             and xcor > 0 and xcor < max-pxcor - 8
        [ set breed IRs                    ;; let some escape as IR
          set heading 20
          set color magenta ]
        [ set heading 100 + random 160 ] ;; return them to earth
    ]
  ]
end

to run-IR
  ask IRs [
    if not can-move? 0.3 [ die ]
    fd 0.3
    if ycor <= earth-top [   ;; convert to heat if we hit the earth's surface again
      set breed heats
      rt random 45
      lt random 45
      set color red - 2 + random 4
    ]
    if any? CO2s-here    ;; check for collision with CO2
      [ set heading 180 - heading ]
  ]
end

to add-CO2  ;; randomly adds CO2 molecules to atmosphere based upon a counter
  let sky-height sky-top - earth-top
  set slow (slow + 1)
  if slow = 50 [create-CO2s GreenHouseGas [
    set color green
    ;; pick a random position in the sky area
    setxy random-xcor
          earth-top + random-float sky-height
  ]]
  if slow = 50 [set slow 0]
end

to remove-CO2 ;; randomly remove 25 CO2 molecules
  repeat Absorption [
    if any? CO2s [
      ask one-of CO2s [ die ]
    ]
  ]
```

```
end

to run-CO2
  ask CO2s [
    rt random 51 - 25 ;; turn a bit
    let dist 0.05 + random-float 0.1
    if [shade-of? green pcolor] of patch-ahead dist [die]
    ;; keep the CO2 in the sky area
    if [not shade-of? blue pcolor] of patch-ahead dist
      [ set heading 180 - heading ]
    fd dist ;; move forward a bit
  ]
end


; Copyright 2007 Uri Wilensky.
; See Info tab for full copyright and license.
```