

checkForGrowth (Internal)

```
on exitFrame
  go to the frame
  pass
end
```

---

sign (Internal)

```
global gPlantList
```

```
on mouseUp
  put "works"
  set the locZ of sprite(30) = 500
  play sprite(30)
  repeat with x = 1 to count(gPlantList)
    puppetSprite getSpriteNum(getAt(gPlantList, x)), 0
  end repeat
  initializeAll
end
```

```
on mouseWithin
  cursor 280
end
```

```
on mouseLeave
  cursor -1
end
```

## Determine Prop Properties (Internal)

```

-- Determine Prop Properties
-- written by Ellen Strain, April 3, 2001
-- this script is dropped on to any prop that will interact with the plants
-- this includes the watering can, the fertilizer, the pesticide, the temperature, and
the rainstorm
-- this script is curenly incomplete

on getBehaviorDescription me
  return \
    "DETERMINE PROP PROPERTIES" & RETURN & RETURN & \
    "Allows developer to set up the basic characteristics of an object to be used in the
garden"
end

on getBehaviorToolTip me
  return \
    "Determine Prop Properties." & \
    "Set the quadrant characteristic that is affected."
end

-- PROPERTIES AND GLOBALS--

global gDepletionIncrement
global gSpriteNum
property pPriorIncrement
global gOccupied
global gPlantList, gPlantMember
global gQuadrantList, gCurrentQuadrant

-- author-defined parameters--

property pRandomORdrag -- whether the user has to drag it or if it is randomly triggered
like weather conditions
property pQuadrantORoverall -- whether it affects a single quadrant or the whole garden
(like a rainstorm)
property pAffectsWhat -- which plant characteristic it affects, e.g. ideal water level,
etc.
property pEffectStrength -- how much effect it has on this particular characteristic
property pLocationV -- initial vertical location
property pLocationH -- initial horizontal location
property pOriginalSpriteNum -- initial sprite number of prop
property pAddORsubtract
-- EVENT HANDLERS --

-- IF DRAGGED
-- look for plant that this prop is affecting
-- make some change to the overall level

-- set initial starting point
on beginSprite
  set gRandomEvent = 0
  set gAttack = 0
  set gAttackQuadrant = 0
  set the frame of sprite the currentSpriteNum = 1
  stop sprite the currentSpriteNum
  set the visible of sprite the currentSpriteNum = TRUE
  set the locV of sprite the currentSpriteNum = pLocationV
  set the locH of sprite the currentSpriteNum = pLocationH
end

```

```

-- cursor change
on mouseWithin
  if gOccupied = 0 then
    cursor 260
  end if
end

on mouseLeave
  cursor -1
end

-- put prop in highest sprite channel
on mouseDown
  if gOccupied = 0 then
    set gPlantMember = 0
    set the frame of sprite the currentSpriteNum = 1
    stop sprite the currentSpriteNum
    set the locZ of sprite the currentSpriteNum = (gSpriteNum + 1)
    set the moveableSprite of sprite the currentSpriteNum = TRUE
    updateStage
  end if
end

on exitFrame me
  if gOccupied = the currentSpriteNum then
    if the frame of sprite the currentSpriteNum < the frameCount of the member of sprite
the currentSpriteNum then
      updateStage
    else
      repeat with y = 1 to count(gPlantList)
        -- if any are in the current quadrant...
        if sprite the currentSpriteNum intersects getSpriteNum(getAt(gPlantList, y)) then
          -- then make the change
          changeProperty(getAt(gPlantList, y)) (pAffectsWhat, pAddORsubtract,
pEffectStrength)
        end if
      end repeat
      stop sprite the currentSpriteNum
      set gOccupied = 0
      set the frame of sprite the currentSpriteNum = 1
      set the locZ of sprite the currentSpriteNum = pOriginalSpriteNum
      set the locV of sprite the currentSpriteNum = pLocationV
      set the locH of sprite the currentSpriteNum = pLocationH
      updateStage
    end if
  end if
  pass
end

-- play animation
on mouseUp
  if gOccupied = 0 then
    if the mouseV > 190 then
      -- begin animation
      play sprite the currentSpriteNum
      -- wait until the animation is done playing
      -- set the locV of sprite the currentSpriteNum = the locV of sprite the
currentSpriteNum - 50
      -- set the locH of sprite the currentSpriteNum = the locH of sprite the

```

```
currentSpriteNum + 35
```

```
    set gOccupied = the currentSpriteNum
    -- scroll through the plant list
```

```
else
    set the locZ of sprite the currentSpriteNum = pOriginalSpriteNum
    set the locV of sprite the currentSpriteNum = pLocationV
    set the locH of sprite the currentSpriteNum = pLocationH
    updateStage
end if
end if
end
```

```
-- IF RANDOMLY TRIGGERED
-- check for gDepletionIncrement in order to trigger a random event, if this is a
randomly triggered object
-- make a change to the overall level
```

```
-- AUTHOR-DEFINED PARAMETERS
```

```
on getPropertyDescriptionList me
    set choiceList1 = ["randomly triggered", "user dragged"]
    set choiceList2 = ["quadrant", "overall"]
    set choiceList3 = ["fertilizer", "water", "pesticide"]
    descriptionList = [:]
    setaProp descriptionList, #pRandomORdrag, [#default:"user dragged", #format:#string,
#comment:"Is this randomly triggered? If not, it will be dragged", #range: choiceList1]
    setaProp descriptionList, #pQuadrantORoverall, [#default:"quadrant", #format:#string,
#comment:"Does this affect a quadrant or the whole garden?", #range: choiceList2]
    setaProp descriptionList, #pAffectsWhat, [#default:"water", #format:#string,
#comment:"What characteristic of the plant does this affect?", #range: choiceList3]
    setaProp descriptionList, #pEffectStrength, [#default:1, #format:#integer, #comment:"How
much effect does it have on the plants?", #range: [#min:1, #max:10]]
    setaProp descriptionList, #pLocationV, [#default:1, #format:#integer, #comment:"What is
the vertical starting point?", #range: [#min:1, #max:480]]
    setaProp descriptionList, #pLocationH, [#default:1, #format:#integer, #comment:"What is
the horizontal starting point?", #range: [#min:1, #max:600]]
    setaProp descriptionList, #pOriginalSpriteNum, [#default:1, #format:#integer,
#comment:"What is the initial sprite number?", #range: [#min:1, #max:40]]
    setaProp descriptionList, #pAddORsubtract, [#default:1, #format:#boolean, #comment:
"Select true if this object contributes to the levels"]
    return descriptionList
end getPropertyDescriptionList
```

---

on prepareMovie (Internal)

```
global gPlantList, gQuadrantList
global gPlantMember
global gSpriteNum
global gOverallWaterLevel
global gOverallFertilizerLevel
global gOverallPesticideLevel
global gOverallTemperature
global gResilience
global gIdealWL
global gIdealFL
global gIdealTemp
global gIdealPL
global gOccupied
global gRandomEvent, gRandomEventList
global gAttackQuadrant
global gStopKudzu
```

```
global gDepletionFlag
global gRandomFlag
```

```
on prepareMovie
  set the trace = 0
  -- preload all of the flash members
  gQuadrantList = []
  preloadMember "cactus1Member"
  preloadMember "cactus2Member"
  preloadMember "roseMember"
  preloadMember "daisy1Member"
  preloadMember "daisy2Member"
  preloadMember "daisy3Member"
  preloadMember "daisy4Member"
  initializeAll
  set the locV of sprite(28) = 15
  set the locH of sprite(28) = 491
  set the locZ of sprite(28) = 28
  set the locV of sprite(29) = 50
  set the locZ of sprite(29) = 29
  set the locV of sprite(32) = 100
  set the locH of sprite(32) = 341
  set the locZ of sprite(32) = 32
end
```

```
on initializeAll
  set gDepletionFlag = 0
  set gRandomFlag = 0
  set gRandomEventList = [#grasshopper, #grasshopper, #butterfly, #butterfly, #bees,
#bees]
  set gOccupied = 0
  set gPlantList = [] -- this list will keep track of what has been planted
  set gPlantMember = 0 -- a mouse click in the soil will not cause planting if this
equals zero
  set gSpriteNum = 100 -- this is the first sprite to puppet in the assigning of plants
to sprites
  startTimer -- a timer starts with the start of the movie
  set gOverallWaterLevel = 3 -- the initial level of water in the soil on a scale of 1 to
10
  set gOverallFertilizerLevel = 1 -- the initial level of fertilizer in the soil (1 to 10
```

```

scale)
  set gOverallPesticideLevel = 1 -- the initial level of pesticide on a scale of 1 to 10
  set gOverallTemperature = 5 -- the initial temperature on a scale of 1 to 10

end

on exitFrame

-- this handler regularly checks to see if a certain period of time has passed
if the timer > 60 * 20 then
  -- if 30 seconds have passed, then the timer is restarted
  startTimer
  -- random event is triggered every 20 seconds * eventTrigger weight
  -- all plants regularly check to see if a depletionIncrement has passed by
  -- comparing their record of the last gDepletionIncrement with the current one
  -- (their record = pPriorIncrement)

  -- the variable increases by one each twenty seconds
  -- with the passage of each twenty seconds the overall levels are taken down a notch
  --   set gOverallWaterLevel = gOverallWaterLevel - .2
  --   set gOverallFertilizerLevel = gOverallFertilizerLevel - .1
  --   set gOverallPesticideLevel = gOverallPesticideLevel - .1
  -- in the message window, you can watch these levels decreasing over time
  --   put "Amount of Pesticide:" & gOverallPesticideLevel
  --   put "Amount of Water:" & gOverallWaterLevel
  --   put "Amount of Fertilizer:" & gOverallFertilizerLevel
  set gDepletionFlag = gDepletionFlag + 1
  put "gDepletionFlag:" & gDepletionFlag
  if gDepletionFlag = 3 then
    set gDepletionFlag = 0
    put "gDepletionFlag is now:" & gDepletionFlag
    repeat with x = 1 to count(gPlantList)
      case (sprite(getSpriteNum(getAt(gPlantList, x))).member.name) of

        ("cactus1member") :
          put the frame of sprite(getSpriteNum(getAt(gPlantList, x)))
          if the frame of sprite(getSpriteNum(getAt(gPlantList, x))) = 125 then
            put the frame of sprite(getSpriteNum(getAt(gPlantList, x)))
            getDepletion(getAt(gPlantList, x), 3, 3, 1)
          else
            nothing
          end if

        ("cactus2member") :
          put the frame of sprite(getSpriteNum(getAt(gPlantList, x)))
          if the frame of sprite(getSpriteNum(getAt(gPlantList, x))) = 124 then
            put the frame of sprite(getSpriteNum(getAt(gPlantList, x)))
            getDepletion(getAt(gPlantList, x), 3, 3, 1)
          else
            nothing
          end if

        ("cactus3member") :
          put the frame of sprite(getSpriteNum(getAt(gPlantList, x)))
          if the frame of sprite(getSpriteNum(getAt(gPlantList, x))) = 125 then
            put the frame of sprite(getSpriteNum(getAt(gPlantList, x)))
            getDepletion(getAt(gPlantList, x), 3, 3, 1)
          else
            nothing
          end if
    end repeat
  end if
end if

```

```

("daisy1member") :
    if the frame of sprite(getSpriteNum(getAt(gPlantList, x))) = 54 then
        getDepletion(getAt(gPlantList, x), 3, 3, 1)
    else
        nothing
    end if

("daisy2member") :
    if the frame of sprite(getSpriteNum(getAt(gPlantList, x))) = 54 then
        getDepletion(getAt(gPlantList, x), 3, 3, 1)
    else
        nothing
    end if

("daisy3member") :
    if the frame of sprite(getSpriteNum(getAt(gPlantList, x))) = 54 then
        getDepletion(getAt(gPlantList, x), 3, 3, 1)
    else
        nothing
    end if

("daisy4member") :
    if the frame of sprite(getSpriteNum(getAt(gPlantList, x))) = 54 then
        getDepletion(getAt(gPlantList, x), 3, 3, 1)
    else
        nothing
    end if

("roseMember") :
    if the frame of sprite(getSpriteNum(getAt(gPlantList, x))) = 84 then
        getDepletion(getAt(gPlantList, x), 3, 3, 1)
    else
        nothing
    end if
end case
end repeat

else
    nothing
end if

set gRandomFlag = gRandomFlag + 1
put "gRandomFlag:" & gRandomFlag
if gRandomFlag = 3 then

    triggerRandomEvent
    set gRandomFlag = 0
else
    nothing
end if

--      -- kudzu
--      case ( gStopKudzu ) of
--          (1) :
--              set the locZ of sprite(2) = 500
--              play sprite(2)
--

```

```
--      repeat with s = 1 to count(gPlantList)
--
--      getDepletion(getAt(gPlantList, s), 30, 30)
--
--      end repeat
--
--      set gStopKudzu = 2
--      (2) :
--      initializeAll
--      otherwise
--      nothing
--      end case

end if

-- send bugs back off stage
if the frame of sprite(28) = the framecount of the member of sprite(28) then
  stop sprite(28)
  set the frame of sprite(28) = 1
  set the locV of sprite(28) = 15
  set the locH of sprite(28) = 491
  set the locZ of sprite(28) = 28
else
  nothing
end if

if the frame of sprite(29) = the framecount of the member of sprite(29) then
  stop sprite(29)
  set the frame of sprite(29) = 1
  set the locV of sprite(29) = 50
  set the locZ of sprite(29) = 29
else
  nothing
end if

if the frame of sprite(32) = the framecount of the member of sprite(32) then
  stop sprite(32)
  set the frame of sprite(32) = 1
  set the locV of sprite(32) = 100
  set the locH of sprite(32) = 300
  set the locZ of sprite(32) = 32
else
  nothing
end if

-- depletion

pass
end

on triggerRandomEvent

-- choose random event
set x = random(6)
set gRandomEvent = getAt(gRandomEventList, x)
put gRandomEvent
```

```

-- choose quadrant affected
set y = random(18)
put y
case (gRandomEvent) of

  (#grasshopper) :
    set gAttackQuadrant = getAt(gQuadrantList, y)
    put gAttackQuadrant
    --      bee animation moved into place

  repeat with z = 1 to count(gPlantList)
    --      if any are in the current quadrant...
    if gAttackQuadrant = (getQuadrant(getAt(gPlantList, z))) then
      put getPesticideLevel(getat(gPlantList, z))
      if getPesticideLevel(getat(gPlantList, z)) <= 1 then
        put getPesticideLevel(getat(gPlantList, z))
        --      then make the change
        set the locV of sprite(28) = the locV of sprite(gAttackQuadrant) - 170
        set the locH of sprite(28) = the locH of sprite(gAttackQuadrant)
        set the locZ of sprite(28) = 500
        play sprite(28)
        getDepletion(getAt(gPlantList, z), 30, 30, 0)

        -- other plants here
      end if
    else
      set the locZ of sprite(28) = 500
      play sprite(28)
    end if
  end repeat

  ---- butterfly
  (#butterfly) :
    repeat with z = 1 to count(gPlantList)
      --      if any are in the current quadrant...
      if gAttackQuadrant = (getQuadrant(getAt(gPlantList, z))) then
        put getPesticideLevel(getat(gPlantList, z))

        if getPesticideLevel(getat(gPlantList, z)) <= 1 then
          set p = random(4)
          case ( p ) of
            ( 1 ) :
              set the locZ of sprite(29) = 500
              set the locV of sprite(29) = 100
              play sprite(29)
            ( 2 ) :
              set the locZ of sprite(29) = 500
              set the locV of sprite(29) = 200
              play sprite(29)
            ( 3 ) :
              set the locZ of sprite(29) = 499
              set the locV of sprite(29) = 300
              play sprite(29)
            ( 4 ) :
              set the locZ of sprite(29) = 500
              set the locV of sprite(29) = 400
              play sprite(29)
          otherwise
            nothing
        end if
      end repeat
    end repeat

```

```

        end case

        end if
    end if
end repeat

(#bees) :
put "start bees"
set gAttackQuadrant = getAt(gQuadrantList, y)
put gAttackQuadrant

firstSprite = count(gPlantList)
repeat with z = 1 to firstSprite

    --          if any are in the current quadrant...
    if gAttackQuadrant = (getQuadrant(getAt(gPlantList, z))) then

        if getPesticideLevel(getat(gPlantList, z)) <= 1 then
            --          then make the change

            oldPlantMember = gPlantMember
            gPlantMember = sprite(100 + z).member.name
            plantSeed (gAttackQuadrant, sprite(100 + z).LocV + 30, sprite(100 + z).LocH +
30)
            plantSeed (gAttackQuadrant, sprite(100 + z).LocV + 30, sprite(100 + z).LocH -
30)

            gPlantMember = oldPlantMember
            --          bee animation moved into place
            set the locV of sprite(32) = the locV of sprite(gAttackQuadrant) + 25
            set the locH of sprite(32) = the locH of sprite(gAttackQuadrant) - 180
            set the locZ of sprite(32) = 500
            play sprite(32)
            firstSprite = z

            -- other plants here
            else
            end if
        else
            set the locZ of sprite(32) = 500
            play sprite(32)
        end if
    end repeat
otherwise
    nothing
end case

--      (#kudzu) :
--      set the visible of sprite(2) = TRUE
--      startTimer
--      set gStopKudzu = 1

end

-- this handler assigns the flower a sprite number and assigns quadrant
on plantSeed quadNum, whichLocV, whichLocH

```

```
-- check to see if the mouse is below skyline

if gPlantMember <> 0 then
  -- make sure a seed packet has first been clicked
  -- when a seed packet is clicked gPlantMember will no longer be zero
  set gSpriteNum = gSpriteNum + 1
  -- the sprite that will be puppeted to contain this new plant should be one higher
  than the last one puppeted
  append(gPlantList, (new (script "parentplant", gPlantMember, gSpriteNum, gResilience,
gIdealWL, gIdealFL, gIdealTemp, gIdealPL, QuadNum, whichLocV, whichLocH)))
  -- all of the globals are passed on to the generalized script that instantiates any
  new plant
  changeGrowthRate(getAt(gPlantList, count(gPlantList)))
  put gPlantList
  -- see how Director references child objects (see the message window)
end if

end
```

## Determine Plant Properties (Internal)

```

-- Determine Plant Properties
-- this script gets attached to the seed packet sprites

on getBehaviorDescription me
  return \
    "DETERMINE PLANT PROPERTIES" & RETURN & RETURN & \
    "Allows developer to set up the basic characteristics of a type of plant. " & \
    "This particular version allows the developer to determine a plant's resilience. " & \
    \
    "Resilience is determined on a scale of 1 to 20. " & \
    "Setting the resilience determines how quickly the plant grows unassisted. " & \
    "The resilience property sets the frame rate of the flash member representing the
plant. " & \
    "Be sure the member name of the sprite to which this behavior is attached can be
concatenated " & \
    "with the word member to generate the name of the Flash cast member representing the
plant."
end

on getBehaviorToolTip me
  return \
    "Determine Plant Properties." & \
    "Set the plants resilience, which will determine how quickly it grows unassisted."
end

-- PROPERTIES AND GLOBALS--

global gPlantMember -- the variable used to determine what type of plant will be planted
when the ground is clicked
global gResilience -- the variable that passes along the resilience of the plant
global gPlantList
global gSpriteNum
global gIdealWL
global gIdealFL
global gIdealTemp
global gIdealPL
global gQuadrant -- quadrant the plant is planted in

property pCurrentGrowthRate
property pPriorIncrement
property pQuadrant

-- author-defined parameters--

property pResilience -- this variable stays localized on an individual seed packet
property pIdealWaterLevel
property pIdealFertilizerLevel
property pIdealTemperature
property pIdealPesticideLevel

-- EVENT HANDLERS --

on mouseDown me -- the global variables for passing along information about the plant to
be planted are established when the user clicks on the seed packet
  case (the name of the member of sprite the currentSpriteNum) of
    ("daisy"):
      set gPlantMember = the name of the member of sprite the currentSpriteNum &
random(4) & "member"

```

```
("cactus"):
    set gPlantMember = the name of the member of sprite the currentSpriteNum &
random(3) & "member"
    otherwise:
        set gPlantMember = the name of the member of sprite the currentSpriteNum & "member"
    end case

-- the Flash cast member representing the plant should be called the name of the seed
packet member plus "member"
set gIdealWL = pIdealWaterLevel
set gIdealFL = pIdealFertilizerLevel
set gIdealTemp = pIdealTemperature
set gIdealPL = pIdealPesticideLevel
set gResilience = pResilience
-- the user determined parameter is turned into a global for passing along when the
user clicks the ground in order to plant the seed
-- these are passed on to the child object
end

-- cursor change
on mouseWithin
    cursor 280
end

on mouseLeave
    cursor -1
end

-- AUTHOR-DEFINED PARAMETERS

on getPropertyDescriptionList me
    descriptionList = [:]
    set aProp descriptionList, #pResilience, [#default: 10, #format: #integer,
#comment: "Unassisted Growth Rate", #range: [#min: 1, #max: 20]]
    set aProp descriptionList, #pIdealWaterLevel, [#default: 5, #format: #integer,
#comment: "Amount of water for ideal growth", #range: [#min: 1, #max: 10]]
    set aProp descriptionList, #pIdealFertilizerLevel, [#default: 5, #format: #integer,
#comment: "Amount of fertilizer for ideal growth", #range: [#min: 1, #max: 10]]
    set aProp descriptionList, #pIdealPesticideLevel, [#default: 5, #format: #integer,
#comment: "Amount of pesticide for ideal growth", #range: [#min: 1, #max: 10]]
    set aProp descriptionList, #pIdealTemperature, [#default: 5, #format: #integer,
#comment: "Best temperature for plant growth", #range: [#min: 1, #max: 10]]
    return descriptionList
end getPropertyDescriptionList
```

---

Determine Quadrant Properties (Internal)

```
-- Determine Quadrant Properties
-- written by Patrick Quattlebaum, April 10, 2001
-- this script gets attached to the quadrant sprites

on getBehaviorDescription me
    return \
        "DETERMINE QUADRANT PROPERTIES" & RETURN & RETURN & \
        "Allows developer to set up the basic characteristics of a quadrant. " & \
        "Resilience is determined on a scale of 1 to 20. " & \
        "Be sure the member name of the sprite to which this behavior is attached can be
        concatenated " & \
        "with the word member to generate the name of the Flash cast member representing random
        event."
end

on getBehaviorToolTip me
    return \
        "Determine Quadrant Properties." & \
        "Set the quadrant's number."
end

-- PROPERTIES AND GLOBALS--

global gPlantList, gQuadrantList
global gPlantMember
global gSpriteNum
global gResilience
global gIdealWL
global gIdealFL
global gIdealTemp
global gIdealPL
global gCurrentQuadrant

-- author-defined parameters--

-- EVENT HANDLERS --

-- plant seed

on beginSprite me
    add(gQuadrantList, the currentSpriteNum)
end

on mouseUp me
    if the mouseV > 190 then
        quadNum=the currentSpriteNum
        whichLocV= the mouseV
        whichLocH= the mouseH
        plantSeed (quadNum, whichLocV, whichLocH)
    end if
end

on mouseWithin
    set gCurrentQuadrant = the currentSpriteNum
```

end

-- AUTHOR-DEFINED PARAMETERS

```
on getPropertyDescriptionList me
descriptionList = []
setaProp descriptionList, #pQuadrant, [#default: 1, #format: #integer, #comment: "Quadrant
number", #range: [#min: 1, #max: 18]]
return descriptionList
end getPropertyDescriptionList
```

---

parentplant (Internal)

```

property pResilience
property pSprite
property pCurrentGrowthRate
property pIdealWaterLevel
property pIdealFertilizerLevel
property pIdealTemperature
property pIdealPesticideLevel
property pLocalWaterLevel
property pLocalFertilizerLevel
property pLocalPesticideLevel
property pQuadrant

```

```

global gPlantList
global gOverallWaterLevel
global gOverallFertilizerLevel
global gOverallPesticideLevel
global gOverallTemperature

```

```

-- this script is used to instantiate any new plant
-- parameters are sent to this script from the ground sprite, which receives the
seed-planting mouse click
-- the parameters are sent with the assistance of global variables that allow the basic
plant properties to
-- be ported from the seed packet sprite (which stores these values as properties) and
transferred by way
-- of the new script "parentPlant" parameters which you see following on new me below

```

```

on new me, whichMember, whichSprite, whichResilience, whichIdealWL, whichIdealFL,
whichIdealTemp, whichIdealPL, whichQuadrant, whichLocV, whichLoCH

```

```

-- this script primarily takes this information sent via parameters and sets the child
object's properties

```

```

-- to these values
puppetSprite whichSprite, 1
-- this sprite is not actually visible on the score but in all ways it acts as if it
were like any other

```

```

-- sprite that you had directly placed on the score
set the member of sprite whichSprite = member whichMember
-- plant the seed in the same spot as the mouse click
set the locV of sprite whichSprite = whichLocV - 30
set the locH of sprite whichSprite = whichLoCH + 5
set the ink of sprite whichSprite = 36

```

```

-- background transparent
set pResilience = whichResilience
set pSprite = whichSprite
set pIdealWaterLevel = whichIdealWL
set pIdealFertilizerLevel = whichIdealFL
set pIdealTemperature = whichIdealTemp
set pIdealPesticideLevel = whichIdealPL
set pQuadrant = whichQuadrant

```

```

-- the initial local levels are set at the same as the overall levels
set pLocalWaterLevel = gOverallWaterLevel
set pLocalFertilizerLevel = gOverallFertilizerLevel
set pLocalPesticideLevel = gOverallPesticideLevel
set pCurrentGrowthRate = pResilience

```

```

-- stops the flash movie in frame one. begins playing after watered (prop script).
-- this assures that when gDepletionIncrement increases by one, pPriorIncrement can be
compared to it
-- to see if one of these periods of time has passed
updateStage

```

```

return me
end

on getDepletion me, minusWater, minusFertilizer, minusPesticide
-- this handler is called from a handler below which is called from the frame script
using on enterFrame
-- so that it is regularly checked (i.e., check once every 15th of a second, or
whatever the frame rate is)
-- has one of these time periods passed? if so, gDepletionIncrement should be higher
than
-- pPriorIncrement which is this object's record of the last gDepletionIncrement
-- deplete all local levels by one and reset pPriorIncrement to gDepletionIncrement
set pLocalWaterLevel = pLocalWaterLevel - minusWater
set pLocalFertilizerLevel = pLocalFertilizerLevel - minusFertilizer
set pLocalPesticideLevel = pLocalPesticideLevel - minusPesticide
put pLocalWaterLevel
put "The new water level for the" && the name of the member of sprite pSprite && "in
quadrant" && pQuadrant && "is" && pLocalWaterLevel
put pLocalPesticideLevel
put "The new pesticide level for the" && the name of the member of sprite pSprite &&
"in quadrant" && pQuadrant && "is" && pLocalPesticideLevel
put pLocalFertilizerLevel
put "The new fertilizer level for the" && the name of the member of sprite pSprite &&
"in quadrant" && pQuadrant && "is" && pLocalFertilizerLevel

-- the message window will display the resulting new growth rate
changeGrowthRate me
end

on getQuadrant me
return pQuadrant
end

on getPesticideLevel me
return pLocalPesticideLevel
end

on getResilience me
-- properties can not be passed outside of the child object or sprite
-- however, information can be sent as long as it is sent by value rather than by
variable
-- this handler allows just this -- for a piece of information to be sent by value
rather than by variable
return pResilience
end

on getSpriteNum me
-- properties can not be passed outside of the child object or sprite
-- however, information can be sent as long as it is sent by value rather than by
variable
-- this handler allows just this -- for a piece of information to be sent by value
rather than by variable
return pSprite
end

on changeProperty me, whichProperty, addORsubtract, howMuch
case (whichProperty) of
"fertilizer":
if addORsubtract = 1 then

```

```

    set pLocalFertilizerLevel = pLocalFertilizerLevel + howMuch
    --      set gOverallFertilizerLevel = gOverallFertilizerLevel + howMuch
else
    set pLocalFertilizerLevel = pLocalFertilizerLevel - howMuch
    --      set gOverallFertilizerLevel = gOverallFertilizerLevel - howMuch
end if
put gOverallFertilizerLevel
put "The new fertilizer level for the" && the name of the member of sprite pSprite
&& "in quadrant" && pQuadrant && "is" && pLocalFertilizerLevel
put "The ideal level of fertilizer for this flower is" && pIdealFertilizerLevel
"water":
if addORsubtract = 1 then
    set pLocalWaterLevel = pLocalWaterLevel + howMuch
    --      set gOverallWaterLevel = gOverallWaterLevel + howMuch
else
    set pLocalWaterLevel = pLocalWaterLevel - howMuch
    --      set gOverallWaterLevel = gOverallWaterLevel - howMuch
end if
put "The new water level for the" && the name of the member of sprite pSprite &&
"in quadrant" && pQuadrant && "is" && pLocalWaterLevel
put "The ideal level of water for this flower is" && pIdealWaterLevel
"pesticide":
if addORsubtract = 1 then
    set pLocalPesticideLevel = pLocalPesticideLevel + howMuch
    --      set gOverallPesticideLevel = gOverallPesticideLevel + howMuch
else
    set pLocalPesticideLevel = pLocalPesticideLevel - howMuch
    --      set gOverallPesticideLevel = gOverallPesticideLevel - howMuch
end if
put "The new pesticide level for the" && the name of the member of sprite pSprite
&& "in quadrant" && pQuadrant && "is" && pLocalPesticideLevel
put "The ideal level of pesticide for this flower is" && pIdealPesticideLevel
end case
if the frame of sprite pSprite = 1 then
    sprite(pSprite).play()
else
    nothing
end if
changeGrowthRate me

end

on changeGrowthRate me
    calculateGrowthRate me
    --      set the playBackMode of sprite pSprite = #fixed
    --      this assures that the flash member does not play at the frame rate that it was
    originally
    --      created with, but rather looks to "the fixedRate" to determine its frame rate
    --      getSpriteNum is a handler on the child object that allows the proper information to
    be
    --      sent by value (rather than by variable, since properties have a limited domain --
    in
    --      other words, they can't leave their own sprite or child object)

    set the playBackMode of sprite pSprite = #fixed
    if pCurrentGrowthRate < 0 then
        --      check for a negative growth rate
        put sprite(pSprite).member.name
        case (sprite(pSprite).member.name) of

            ("cactus1member") :

```

```
if sprite(pSprite).frame > 125 then
  nothing
  put "works"
else
  sprite(pSprite).frame = 250 - sprite(pSprite).frame
  play sprite(pSprite)
  set the fixedRate of sprite pSprite = abs(pCurrentGrowthRate)
end if
("cactus2member") :
if sprite(pSprite).frame > 124 then
  nothing
  put "works"
else
  sprite(pSprite).frame = 250 - sprite(pSprite).frame
  play sprite(pSprite)
  set the fixedRate of sprite pSprite = abs(pCurrentGrowthRate)
end if
("cactus3member") :
if sprite(pSprite).frame > 125 then
  nothing
  put "works"
else
  sprite(pSprite).frame = 250 - sprite(pSprite).frame
  play sprite(pSprite)
  set the fixedRate of sprite pSprite = abs(pCurrentGrowthRate)
end if
("daisy1member") :
if sprite(pSprite).frame > 54 then
  nothing
else
  sprite(pSprite).frame = 109 - sprite(pSprite).frame
  play sprite(pSprite)
  set the fixedRate of sprite pSprite = abs(pCurrentGrowthRate)
end if
("daisy2member") :
if sprite(pSprite).frame > 54 then
  nothing
else
  sprite(pSprite).frame = 109 - sprite(pSprite).frame
  play sprite(pSprite)
  set the fixedRate of sprite pSprite = abs(pCurrentGrowthRate)
end if
("daisy3member") :
if sprite(pSprite).frame > 54 then
  nothing
else
  sprite(pSprite).frame = 109 - sprite(pSprite).frame
  play sprite(pSprite)
  set the fixedRate of sprite pSprite = abs(pCurrentGrowthRate)
end if
("daisy4member") :
if sprite(pSprite).frame > 54 then
  nothing
else
  sprite(pSprite).frame = 109 - sprite(pSprite).frame
  play sprite(pSprite)
  set the fixedRate of sprite pSprite = abs(pCurrentGrowthRate)
end if
("rosemember") :
if sprite(pSprite).frame > 84 then
  nothing
```

```
    else
      sprite(pSprite).frame = 170 - sprite(pSprite).frame
      play sprite(pSprite)
      set the fixedRate of sprite pSprite = abs(pCurrentGrowthRate)
    end if
  otherwise
    nothing
end case

else
  -- reset the frame rate of the appropriate flash member to the new growth rate

  case (sprite(pSprite).member.name) of

    ("cactus1member") :
      if sprite(pSprite).frame < 126 then
        nothing
      else
        sprite(pSprite).frame = 250 - sprite(pSprite).frame
        play sprite(pSprite)
        set the fixedRate of sprite pSprite = abs(pCurrentGrowthRate)
      end if
    ("cactus2member") :
      if sprite(pSprite).frame < 125 then
        nothing
      else
        sprite(pSprite).frame = 250 - sprite(pSprite).frame
        play sprite(pSprite)
        set the fixedRate of sprite pSprite = abs(pCurrentGrowthRate)
      end if
    ("cactus3member") :
      if sprite(pSprite).frame < 126 then
        nothing
      else
        sprite(pSprite).frame = 250 - sprite(pSprite).frame
        play sprite(pSprite)
        set the fixedRate of sprite pSprite = abs(pCurrentGrowthRate)
      end if
    ("daisy1member") :
      if sprite(pSprite).frame < 55 then
        nothing
      else
        sprite(pSprite).frame = 109 - sprite(pSprite).frame
        play sprite(pSprite)
        set the fixedRate of sprite pSprite = abs(pCurrentGrowthRate)
      end if
    ("daisy2member") :
      if sprite(pSprite).frame < 55 then
        nothing
      else
        sprite(pSprite).frame = 109 - sprite(pSprite).frame
        play sprite(pSprite)
        set the fixedRate of sprite pSprite = abs(pCurrentGrowthRate)
      end if
    ("daisy3member") :
      if sprite(pSprite).frame < 55 then
        nothing
      else
        sprite(pSprite).frame = 109 - sprite(pSprite).frame
        play sprite(pSprite)
        set the fixedRate of sprite pSprite = abs(pCurrentGrowthRate)
      end if
  end case
end if
end case
```

```

    end if
    ("daisy4member") :
    if sprite(pSprite).frame < 55 then
        nothing
    else
        sprite(pSprite).frame = 109 - sprite(pSprite).frame
        play sprite(pSprite)
        set the fixedRate of sprite pSprite = abs(pCurrentGrowthRate)
    end if
    ("rosemember") :
    if sprite(pSprite).frame < 85 then
        nothing
    else
        sprite(pSprite).frame = 170 - sprite(pSprite).frame
        play sprite(pSprite)
        set the fixedRate of sprite pSprite = abs(pCurrentGrowthRate)
    end if
    otherwise
        nothing
    end case
    set the fixedRate of sprite pSprite = abs(pCurrentGrowthRate)
end if
put "The new growth rate of" && the name of the member of sprite pSprite && "in
quadrant" && pQuadrant && "is" && pCurrentGrowthRate
checkforDeath
end

on calculateGrowthRate me
-- first all of the individual levels are calculated
-- the overall levels are taken into account but have only a small effect (thus, the .1
multiplier)
-- for both overall and local levels, the absolute value of the difference between the
level and the plant's ideal level is calculated
set fertilizerDifference = abs(pLocalFertilizerLevel - pIdealFertilizerLevel)
set waterDifference = abs(pLocalWaterLevel - pIdealWaterLevel)
set tempDifference = abs(gOverallTemperature - pIdealTemperature)
-- for temperature, there is only an overall level, no local level
set pesticideDifference = abs(pLocalPesticideLevel - pIdealPesticideLevel)
-- all of the differences from the ideal are added up
-- the multiplier here (.2) is a sensitivity factor that determines how much effect the
levels have on the growth rate
set totalDifference = integer(.2 * (fertilizerDifference + waterDifference))
-- + pesticideDifference
-- this number is subtracted from the plant's original growth rate (its default growth
rate, i. e., its resilience)
set pCurrentGrowthRate = pResilience - totalDifference
end

on checkForDeath whichSprite
-- this script not yet written
-- check to see if the flash member is on its last frame
-- if so remove child object from memory, un-puppetsprite after making it invisible
-- then abort
end

```

