Fluctuations of Rainfall and its Effect on an Agricultural Environment

New Mexico

Supercomputing challenge

Final Report

April 6, 2011

Team 118

Volcano Vista High School

Team members

Joseph Miera

Joe Fosse

Teacher

Janet Penevolpe

Table of Contents

Advances in science in the field of agriculture have brought many useful things that have improved yield and quality of crops: high-tech machinery to help grow and harvest better, fertilizers to increase yield and a healthy crop, crop dusters and pesticides to rid the field of invasive intruders. A problem remains: How much can a few simple fluctuations of rainfall effect an agricultural environment and those making their living off of it? According to the studies of this project, it appears it does have a greatly significant effect. The project is based off of captured data from a simulation of a wheat farm. All the significant variables are recorded as the simulation plays out, including the irrigated water usage of the farmer, rainfall, amount of [wheat] bushel return, profit, revenue (gross profit, no costs deducted), and costs. The simulation loops through the days of a season, mimicking the basic events and conditions that go into a day. It takes into account several simulated conditions including: a range of possible chances of rain, a range of possible rainfall amounts, cost of irrigated water, field size, grain price, and amount the farmer waters weekly (if they need to water). These variables can be user-defined as of the final edits to the program. Patterns and trends can be noted as conditions are changed, trends such as the probable outcomes of a year with less rain than normal. In other words, the user can define specific conditions and the program will predict the possible outcome(s) of that season. It can also do predictions spanning over many seasons, or it can re-simulate one season over and over. And if the user does not define conditions, the program can randomly generate conditions and run with those. The power of the project is immense, even though this is only its first year of being. Many things have been and can be noticed from the data generated by the program, including trends relating to field size and revenue (and thus profit), trends relating to rainfall and water usage and costs (and thus profit),

and other minor trends. The power of the program is immense, and our plans for it in the future will even more greatly increase the scope and power of it. In the future, for the next challenges, we plan to make the program be able to access online databases and thus suggest the likely conditions for the specified year. We plan to make the program not be solely dependent on wheat and be able to use any crop specified. Also, we plan to make the program suggest the crop that would do best in specific years. But, if one is focusing on the present, it is still an amazingly powerful program, because even though the default conditions are based off an area here in New Mexico, it is adaptable to any area. This means one could apply predictions and ranges of possible data to themselves even if they do not live in New Mexico. With this tool, it could potentially help farmers maximize their yields, communities maximize their farming industry, and thus help the economy and help individuals with the abundance of things we need. We see this as a probable tool that doesn't alter nature, but allows us to maximize its potential. Maybe farmers can't change the rainfall, but with this program they can take all the data and information available to make the best decision, and can make the most of what they get.

Introduction:

This project is based on the problem of fluctuations of rainfall affecting an agricultural environment. This really translates to be: a lot of simulating and researching. The simulation created was, in turn, based off of the research of conditions and trends of wheat, and, for default conditions, the Lower Mesilla Valley here in New Mexico. It loops through the days of a season continuously, with the user set or preset conditions. With these conditions, it mocks the actual major events and happenings in a day. These events include rainfall, irrigation, and, at the end of the season, tally of revenue, costs, profit, and bushels gained are calculated. With this simulation, it outputs data that is a prediction of what may happen based on the conditions. If one approximately knows the current conditions (such as how much they water their farm each week, about how much the range of rain amounts are, and such), the program will predict their final income and how much their costs might be and how much their revenue might be. This problem is significant to me and to everyone else, because if we know there is going to be a drought based on patterns, we can approximate how much it would hurt individual farmers. This would affect the market in which I buy my food. Thus, if we can approximate how much farmers would get hurt, one could probably approximate the change in food prices. This is important to me, because it would be nice to know if I am going to have to pay twice as much for bread next year, so I can prepare the finances needed to pay for my own costs when the time comes. Most people probably agree with that. With something as big as agriculture, it affects the entire world.

Method:

The program takes certain conditions, and runs through a simulated season with them, and returns an outcome that may be similar to what would happen in real life. It will loop again and again, and depending on your settings, will predict an outcome over many seasons, or the many possible outcomes of one season.

The process of the project:

1. If it is the first day of the "season" all graphs and variables are established. If it is the first day of the first season, or the user has clicked the "reset each season" box, permanent graphs and variables are established.
2. Then it loops through each day:
    a. Starts by checking the chance of raining. A value in between the max. and min. of percent chance of rain is randomly generated. That becomes the percent chance of rain that day.
    b. It then randomly generates an integer between 0 and 100, and if the value is less than or equal to the percent chance of rain, it rains.
    c. If it rains, a randomly generated number between the min. and max. possible amount of rain becomes the amount of rain in inches.
    d. If it has rained more than the amount the farmer waters each week, the farmer does not water. If it has not rained enough, the farmer will water.
    e. If the farmer waters, he puts the same amount of water (in inches) in the field as he does every week. It never differs in that respect.

3. It loops through half a years worth of days, then reaches the season's end.
4. At the season's end, water used determines the cost of that season, and the predicted amount of bushels is returned. The amount of bushels and bushel price determine revenue. Profit is calculated by the difference of revenue and cost.
5. If told to do so, the graphs save their data to the local folder.
6. It then loops through another season.

As of this year, conditions can be randomly generated, or manually inputted. The program cannot retrieve predicted conditions (i.e. the range of percent chance of rain that is possible to have each day, the minimum and maximum that is possible to rain each day, etc.) on its own from the internet, but as of this year can only retrieve conditions manually. We plan to remedy this for next year's competition. The project uses wheat as a basis, and uses its water needs, and bushel price in the calculations. But also by next year we plan to have the project be able to use other plants as a basis, and be able to return a recommended crop for specific conditions.

Results and Conclusions:

There are several things that can be found from the data generated from our project. For example, our team found that when it rains less, the revenue and profit gained are significantly less. This is most likely because of several reasons. First of all, when it rains less, the farmer has to water more. If the farmer has to water more himself, and there is less that comes naturally (and for free), the overall costs are greater in the end. Also, when there's less water in the field overall, less bushels of wheat can be produced and sold. This means there would also be a drop in revenue. Another thing that can be pulled from the data is that field size affects revenue and cost greatly. If one of the field's attributes is that it is only one acre, the profit will only compile by increments in the hundreds range, but if the size attribute of the field is fifty acres, then the farmer's profit will accumulate by the thousands (in either the positive or negative direction). In other words, the larger the field, the greater possibility of attaining an enormous profit, but it is also a greater risk of going into an enormous amount of debt. Other trends that can be noted include minor ones such as the fact that the amount the farmer waters overall can help determine how severe the drought is (e.g. If they water a lot, it was a severe drought with little rain.)

See Appendix B for graphic representations

Acknowledgements:

Thanks to our wonderful teacher Miss Penevolpe, who has guided and helped us in Volcano Vista's first year of Supercomputing. Thanks to everyone involved and for parent support. We could not have done it with out all the help we have received from all these great people.

Works Cited:

Kowalski, Chuck. "Wheat - Planting and Harvest Seasons for Wheat Crops." *Commodities - About Commodities and Futures Trading*. About.com. Web. 06 Apr. 2011. <http://commodities.about.com/od/researchcommodities/a/wheat-seasons.htm>.

Bauder, James W. "Wheat Irrigation." *MSU Extension Water Quality Program*. 12 May 2005. Web. 06 Apr. 2011. <http://waterquality.montana.edu/docs/irrigation/wheatirrigation.shtml>.

Chang, C. W. "Chemical Properties of Alkali Soils in Mesilla Valley, New M... : Soil Science."*LWW Journals*. Web. 06 Apr. 2011. <http://journals.lww.com/soilsci/Citation/1953/03000/Chemical_Properties_of_Alkali_Soils_in_Mesilla.8.aspx>.

"Requirements for Growing Wheat | Garden Guides." *Garden Guides, Your Guide to Everything Gardening*. Web. 18 Feb. 2011. <http://www.gardenguides.com/124101-requirements-growing-wheat.html>.

An Excerpt from "Wheat Irrigation", EM 3048, December, 1976. Washington State University Extension Service.

"Physical Characteristics of the Site." *Co.dona-ana.nm.us*. Web. 18 Feb. 2011. http://www.co.dona-ana.nm.us/superfund/docs/ris/RISsection3.pdf

Herrera, Esteban A., and Ted Sammis. "Water Management in Pecan Orchards." *Docstoc – Documents, Templates, Forms, Ebooks, Papers & Presentations*. Web. 18 Feb. 2011. <http://www.docstoc.com/docs/68291777/Water-Management-in-Pecan-Orchards>.

"Price of Wheat." *Wheat Prices, Wheat Quote*. Web. 18 Feb. 2011. <http://www.quotewheat.com/>.

"Cost of Water." *Fairfax Water*. Web. 18 Feb. 2011. <http://www.fcwa.org/story_of_water/html/costs.htm>.

Appendix A:

Rainfall_in_an_agricultural_setting.exe (main (master) program)

Code from objMain, main handler of events and controls the flow of the program

Information about object: objMain

Sprite: <no sprite>
Solid: false
Visible: true
Depth: 0
Persistent: false
Parent: <no parent>
Mask: <same as sprite>

Create Event:
execute code:

```
// http://commodities.about.com/od/researchcommodities/a/wheat-seasons.htm
// Growing season (two seasons per year taking up all year)

//http://waterquality.montana.edu/docs/irrigation/wheatirrigation.shtml
//Total seasonal requirement is about 18-21" inches for these two crops, depending upon the location within the state and seasonal weather variations.

//located in the lower Mesilla valley?
//http://journals.lww.com/soilsci/Citation/1953/03000/Chemical_Properties_of_Alkali_Soils_in_Mesilla.8.aspx
//Average Low Temperature, 27.9° F. Rainy Days, 13%. Average Annual Rainfall, 9.2 inches ...

globalvar FIRSTLOOP, LOOPCOUNT, DAYCOUNT, SEASONCOUNT, TEMPERATURE, Potential, YIELD;
globalvar WATER, tempWater, TOTALWATER, R, I, GROUNDWATER;//WATER is total amt. of water in field, temp"" is the same but respective to the day
globalvar LOW,HIGH,CHANCERAIN,Daylength,TOTALDAYS;//lowest amt. of rainfall possible, highest possible, % chance of rain
LOW = 0.5; HIGH = 1.5; CHANCERAIN = 13; LCHANCE = 0; HCHANCE = 13;//HChance and LChance determine the possible range of percent chance of rain each day
Daylength = 4;
I = 1;

TOTALWATER = 0;

FIRSTLOOP = 1;
LOOPCOUNT = 0;
```

DAYCOUNT = 0; TOTALDAYS = 0;
SEASONCOUNT = 0;

TEMPERATURE = 75;//Wheat grows best in temperatures between 70 and 75 degrees F. The minimum temperature that wheat can withstand during its growth cycle is about 40 degrees F.
//Wheat does not grow well if temperatures exceed 95 degrees F. Temperatures below 40 degrees F during seed germination will result in lower germination rates. Temperatures higher than 95 degrees F during maturation will result in lower yields.
//From "http://www.gardenguides.com/124101-requirements-growing-wheat.html"

//

/*SM = 1.5;*/ R = 0; I = 0; Potential = 0;//Potential is measure of "Life" or how well a field is doing by how much it can return

/*From "http://waterquality.montana.edu/docs/irrigation/wheatirrigation.shtml"

A simple way to calculate either the potential yield or seasonal water requirements is by use of models, such as the following:

Estimated yield (in bushels/acre) = 5.8 (SM + R/I - 4.1) bushels/acre where:

SM = soil moisture (inches)
R = rainfall (inches)
I = irrigation (inches)

For instance, if plant available soil moisture is 6 inches, rainfall is 3 inches, and irrigation is 6 inches, estimated yield in bushels/acre is = 5.8 (6 + 3 + 6 - 4.1) = 63 bu/acre.

Admittedly, this provides only an estimate. More refined models are available to distinguish between winter wheat and spring wheat. However, for practical planning purposes, models like the above provide a good approximation.

*Excerpted from "Wheat Irrigation", EM 3048, December, 1976. Washington State University Extension Service.
*/
//

//some really good info "http://www.co.dona-ana.nm.us/superfund/docs/ris/RISsection3.pdf"
//Potential evaporation and transpiration greatly exceeds rainfall. Potential evaporation rates measured in an evaporation pan average about 97 inches per year
//Most rain is limited to brief, sometimes intense
//thunderstorms, with more than half of the annual precipitation falling during the period July through
//September

execute code:

```
//http://www.docstoc.com/docs/68291777/Water-Management-in-Pecan-Orchards
//Water Moisture about 1.5 in./foot assuming 3 ft root zone
//
//
//

execute code:

globalvar SEASONWATER,DROUGHT,RR,WR,SEASONDAYCOUNT,WI;
globalvar severity;
DROUGHT = 0;//whether there is a drought or not
SEASONWATER = 0;
RR = 0;//temporary rain of a day, R is seasonal rain
WR = 0;//temporary rain of the week
WI = 0;//Weekly irrigation count
SEASONDAYCOUNT = 0;//how many days in a season
severity = 0;

execute code:

globalvar WATERGRAPH,MONEYNGRAPH,MONEYGGRAPH,MONEYLGRAPH;
WATERGRAPH = objMain;//The graph to keep track of water
MONEYNGRAPH = objMain;//Graph tracking net profit
MONEYGGRAPH = objMain;//graph tracking gross profit
MONEYLGRAPH = objMain;//graph tracking money debts

execute code:

globalvar FIRSTSEASON;//whether it is the first season of the simulation
FIRSTSEASON = 1;
globalvar COLORARR;
COLORARR[0] = c_blue;

globalvar INDICATORG;//Indicator button graph
INDICATORG = self;

//an acre is 6,272,640 inches squared
globalvar Profit,Debit;
Profit = 0; Debit = 0;
NetProfit = 0;

execute code:

globalvar
default_run,usrPercentmax,usrPercentlow,usrWeeklyI,usrRainhigh,usrRainlow,usrGrainprice,usrWaterc
ost,usrFieldsize;
```

```
default_run = 1;//whether there are NOT user values to use instead of the default random ones
usrPercentmax = HCHANCE;//user vals for highest possible chance of rain per day
usrPercentlow = LCHANCE;//user vals for lowest possible chance of rain per day
usrRainhigh = HIGH;//high amt. it can rain when it does rain
usrRainlow = LOW;//low amt. it can rain when it does rain
usrWeeklyI = WEEKLY;//weekly amt. farmer irrigates
usrGrainprice = 5.5;//profit per bushel
usrWatercost = 1.5;//water cost per thousand gallons
usrFieldsize = 1;//Size of the field
```

Other Event: Game Start:

execute code:

```
/*if (!file_exists("BatchRASSCC01.bat"))
 {
 var fff,str;
 fff = file_text_open_write("BatchRASSCC01.bat");
 str = "UserVarGetSlave.exe "+string(usrPercentmax)+" "+string(usrPercentlow)+" "+string(usrWeeklyI)+"
"+string(usrRainhigh)+" "+string(usrRainlow)+" "+string(usrGrainprice)+" "+string(usrWatercost)+"
"+string(usrFieldsize);
 file_text_write_string(fff,str);
 file_text_close(fff);
 }
else
 {
 var ans;
 ans = show_question("BatchRASSCC01.bat already exists.  This is a file the program needs to overwrite
to operate correctly.  If this Batch file is important please hit 'no' and change its directory.  If this is not
important, hit 'yes'.  If program was shutdown unexpectedly, this is a common error and just hit 'yes'.");
 if (ans)//if they say yes, overwrite file
  {
  var fff,str;
  fff = file_text_open_write("BatchRASSCC01.bat");
  str = "UserVarGetSlave.exe "+string(usrPercentmax)+" "+string(usrPercentlow)+"
"+string(usrWeeklyI)+" "+string(usrRainhigh)+" "+string(usrRainlow)+" "+string(usrGrainprice)+"
"+string(usrWatercost)+" "+string(usrFieldsize);
  file_text_write_string(fff,str);
  file_text_close(fff);
  }
 }*/
```

Other Event: Game End:

execute code:

file_delete("BatchRASSCC01.bat");

Draw Event:

execute script scrDrought with arguments (0,0,0,0,0)
execute script scrTempMain with arguments (0,0,0,0,0)

ScrDrought(): script that handles drought intervals at the beginning of each season

```
if (SEASONCOUNT == 1) || (SEASONCOUNT > 1 && DROUGHT)//whether to cause a drought or not, second season will be drought, first will not, rest are random
 {
 if (SEASONDAYCOUNT == 1)//if first day of drought
  {
  DROUGHT = 1;//confirm drought
  //var severity;
  severity = choose(1,2,3);//how severe is the drought, 3 is the worst
  switch (severity)
   {
   case 1:
   LOW = 0; HIGH = 1;
   HCHANCE = 8;
   break;

   case 2:
   LOW = 0; HIGH = .5;
   HCHANCE = 4;
   break;

   case 3:
   LOW = 0; HIGH = 0;
   HCHANCE = 0;
   break;
   }
  }

 }

if (!DROUGHT) && (SEASONDAYCOUNT == 1)
 {
 LOW = 0.5; HIGH = 1.5; LCHANCE = 0; HCHANCE = 13;
 }

var /*severity,*/aaa,ccc; //severity = 0;
aaa = draw_get_alpha();
ccc = draw_get_color();
draw_set_alpha(1);
draw_set_color(c_black);

//draw_text(8,room_height-32,string(DROUGHT)+","+string(severity));//draw drought info

draw_set_alpha(aaa);
draw_set_color(ccc);
```

Information about object: objField (the field that is seen in the middle of the Window)

Sprite: <no sprite>
Solid: false
Visible: true
Depth: 100
Persistent: false
Parent: <no parent>
Mask: <same as sprite>

Create Event:
execute code:

```
LIGHT = .5;
Draw_light = LIGHT;//the alpha the orange color is actually drawn with
```

Draw Event:
execute code:

```
draw_set_color(c_black);
//draw_rectangle(96,96,room_width-64,room_height-64,0);
draw_rectangle(152,8,882-8,room_height-152,0);
draw_set_color(c_orange);
draw_set_alpha(Draw_light);
draw_rectangle(152,8,882-8,room_height-152,0);

if (Draw_light > LIGHT)
 {
 Draw_light -= .075;
 }
else
 {
 Draw_light = LIGHT;
 }
draw_set_alpha(1);
draw_set_color(c_green);
draw_text(152,8,"The Wheat Field");
```

Information about object: objRain (the rain that is drawn when it does rain)

Sprite: sprRain
Solid: false
Visible: true
Depth: 50
Persistent: false
Parent: <no parent>
Mask: <same as sprite>

Create Event:
execute code:

```
x = objField.x;
y = objField.y;
image_alpha = .2;
deathcount = 4;
```

Other Event: Animation End:
execute code:

```
if (deathcount > 0)
 {
 deathcount -= 1;
 }
else
 {
 instance_destroy();
 }
```

Draw Event:
execute code:

```
draw_sprite(sprite_index,image_index,152,8);//draw on field
```

Information about object: objIndicator (colored box at top of window, indicates the kind of drought or whether user-inputted data is being run)

Sprite: <no sprite>
Solid: false
Visible: true
Depth: -100
Persistent: false
Parent: <no parent>
Mask: <same as sprite>

Create Event:
execute code:

```
haveset = 0;
my_col = c_blue;
```

End Step Event:
execute code:

```
if (!haveset)
 {
 if (instance_exists(objGraph))
  {
  objGraph.dotcol = my_col;
  haveset = 1;
  }
 }
```

Draw Event:
execute code:

```
var aaa,ccc,str;
ccc = draw_get_color();//backup color
aaa = draw_get_alpha();//backup alpha
draw_set_color(c_black);
str = "Drought Severity";
draw_text((room_width/2)-(string_width(str)/2),0,str);
if (default_run)
 {
 if (!DROUGHT)
  {
  LOW = 0.5; HIGH = 1.5; LCHANCE = 0; HCHANCE = 13;//set correct values

  draw_set_color(c_blue);
  my_col = c_blue;
  draw_rectangle((room_width/2)-
64,string_height(str),(room_width/2)+64,string_height(str)+16,0);//draw indicator
```

```
 draw_set_color(c_white);
 draw_text((room_width/2)-64,string_height(str),"None");
 }
else
 {
 switch (severity)
  {
  case 1:
  draw_set_color(c_yellow);
  my_col = c_yellow;
  draw_rectangle((room_width/2)-
64,string_height(str),(room_width/2)+64,string_height(str)+16,0);//draw indicator
  draw_set_color(c_black);
  draw_text((room_width/2)-64,string_height(str),"Mild");
  break;

  case 2:
  draw_set_color(c_orange);
  my_col = c_orange;
  draw_rectangle((room_width/2)-
64,string_height(str),(room_width/2)+64,string_height(str)+16,0);//draw indicator
  draw_set_color(c_black);
  draw_text((room_width/2)-64,string_height(str),"Medium");
  break;

  case 3:
  draw_set_color(c_red);
  my_col = c_red;
  draw_rectangle((room_width/2)-
64,string_height(str),(room_width/2)+64,string_height(str)+16,0);//draw indicator
  draw_set_color(c_white);
  draw_text((room_width/2)-64,string_height(str),"Severe");
  break;
  }
 }

}
else//if running user-variables, say so
 {
 draw_set_color(c_dkgray);
 my_col = c_black;
 draw_rectangle((room_width/2)-
64,string_height(str),(room_width/2)+64,string_height(str)+16,0);//draw indicator
 draw_set_color(c_white);
 draw_text((room_width/2)-64,string_height(str),"User-Defined");
 }
```

```
draw_set_color(ccc);//restore old color
draw_set_alpha(aaa);//restore old alpha
```

Information about object: objGraph (The graphs that keep track of all data)

Sprite: <no sprite>
Solid: false
Visible: true
Depth: 0
Persistent: false
Parent: <no parent>
Mask: <same as sprite>

Create Event:
execute code:

```
if (!variable_local_exists("XArray")) {XArray[0] = 0;}//XArray
if (!variable_local_exists("YArray")) {YArray[0] = 0;}//YArray arrays holding pixels
if (!variable_local_exists("XC")) {XC = 0;}//XC current dot
if (!variable_local_exists("YC")) {YC = 0;}//YC current dot
if (!variable_local_exists("ind")) {ind = 0;}//ind
//if (!variable_local_exists("")) {}//XWidthNum
if (!variable_local_exists("YHeightNum")) {YHeightNum = 10;}//YHeightNum
if (!variable_local_exists("pxlHeight")) {pxlHeight = 96;}//pxlHeight
if (!variable_local_exists("pxlWidth")) {pxlWidth = room_width;}//pxlWidth
if (!variable_local_exists("dotcol")) {dotcol = c_black;}//color to draw graph in
if (!variable_local_exists("bckcol")) {bckcol = c_white;}//color to draw background with
if (!variable_local_exists("squish")) {squish = 1;}//squish ??
if (!variable_local_exists("GName")) {GName = "Graph";}//Graph name

if (!variable_local_exists("Permanent")) {Permanent = 0;}

if (!variable_local_exists("XGreatest")) {XGreatest = 0;}
if (!variable_local_exists("YGreatest")) {YGreatest = 0;}
if (!variable_local_exists("XLowest")) {XLowest = 0;}
if (!variable_local_exists("YLowest")) {YLowest = 0;}

if (!variable_local_exists("NewCoord")) {NewCoord = 0;}//whether there are new coordinates or not
if (!variable_local_exists("NewX")) {NewX = 0;}//next x to be graphed
if (!variable_local_exists("NewY")) {NewY = 0;}//next y to be graphed

if (!variable_local_exists("tpText")) {tpText = 0;}
xlast = x;//variable that stores
ylast = y;
save_at_end = 0;
```

Destroy Event:
execute code:

```
if (save_at_end)
 {
 var fff,str;
  str = "";

  fff = file_text_open_write(uid+".txt");//clear any data previously in file
  file_text_write_string(fff,"");
  file_text_close(fff);
  for (i=0;i<ind;i+=1)//write data to file
   {
   str = string(XArray[i])+" "+string(YArray[i]);//

   fff = file_text_open_append(uid+".txt");
   file_text_write_string(fff,str);
   file_text_close(fff);

   fff = file_text_open_append(uid+".txt");
   file_text_writeln(fff);
   file_text_close(fff);
   }//end writing data loop
 instance_create(mouse_x,mouse_y,objSaved);//show "Saved" message
 save_at_end = 0;
 }
```

End Step Event:

execute code:

```
if (NewCoord)//if there is another coordinate detected
 {
 XArray[ind] = NewX; XC = NewX;//add new coordinates to array
 YArray[ind] = NewY; YC = NewY;//set to Current coordinate

 if (XC > XGreatest) {XGreatest = XC;}
 if (YC > YGreatest) {YGreatest = YC;}//if bigger than biggest number, change record

 if (XC < XLowest) {XLowest = XC;}
 if (YC < YLowest) {YLowest = YC;}//if smaller than smallest number, change record

 NewX = 0;
 NewY = 0;
 ind += 1;
 NewCoord = 0;
 }
```

Draw Event:
execute code:

```
var i,xpoint,ypoint,aaa,ccc;

dotcol = objIndicator.my_col;

objMouse.image_index = 0;
aaa = draw_get_alpha();//save backup
ccc = draw_get_color();
draw_set_color(bckcol);
draw_rectangle(x,y-16,x+string_width(GName),y+string_height(GName)-16,1);
if (tpText = 0)
{
draw_rectangle(x,y,x+pxlWidth,y+pxlHeight,0);//draw background

draw_set_color(c_white);
draw_text(x,y-16,GName);

draw_set_color(ccc);
draw_set_alpha(aaa);

draw_set_color(c_gray);
//draw_set_alpha(.3);//draw bar dividing graph
if ((YGreatest - YLowest) <= pxlHeight)//if not too many gray lines draw horizontal lines
 {
 for (i=YLowest;i<YGreatest;i+=1)
  {
  draw_line(x,y+(i*(pxlHeight/(YGreatest - YLowest))),x+pxlWidth,y+(i*(pxlHeight/(YGreatest-YLowest))));
  }
 }
draw_set_color(c_black);
draw_text(x,y,string(YGreatest));
draw_text(x,y+pxlHeight-16,YLowest);

for (i=0;i<ind;i+=1)//draw line graph in loop
 {
 aaa = draw_get_alpha();//save backup
 ccc = draw_get_color();

 if (XGreatest > 0 && YGreatest > 0) || (/*XLowest < 0 || */YLowest < 0)
  {
  xpoint = (pxlWidth/(XGreatest - XLowest))*XArray[i];//set pixel coordinates to draw on
  ypoint = (pxlHeight/(YGreatest - YLowest))*YArray[i];

  draw_set_alpha(.3);//draw bar dividing graph
  draw_set_color(c_gray);
```

```
 draw_line(x+xpoint,y,x+xpoint,y+pxlHeight);

 draw_set_alpha(1);

 if (i > 0)//connect dots with line
  {
  var lnwidth;
  draw_set_color(dotcol);
  if (Permanent)
   {
   draw_set_color(COLORARR[i-1]); lnwidth = 2;
   }
  else {lnwidth = 1;}
  xlast = (pxlWidth/(XGreatest - XLowest))*XArray[i-1];
  ylast = (pxlHeight/(YGreatest - YLowest))*YArray[i-1];
  draw_line_width(x+xlast,(y+pxlHeight)-ylast,x+xpoint,(y+pxlHeight)-ypoint,lnwidth);//draw connecting
line
  }

 draw_point(x+xpoint,(y+pxlHeight)-ypoint/*,dotcol*/);//draw point
 }
}
}
else
{
draw_set_color(c_white);
/*draw_line(x,y,x,y-string_height(GName));//draw box around title
draw_line(x+string_width(GName),y,x+string_width(GName),y-string_height(GName));
draw_line(x,y-string_height(GName),x+string_width(GName),y-string_height(GName));*/
draw_rectangle(x,y-16,x+string_width(GName),y+string_height(GName)-16,1);
draw_text(x,y-16,GName);
draw_set_font(fntLarge);
draw_set_color(bckcol);
draw_rectangle(x,y+8,x+string_width(string(YC)),y+string_height(string(YC)),0)
if (YC > 0) {draw_set_color(c_blue);}
else {draw_set_color(c_red);}
draw_text(x,y+8,string(YC));
draw_set_font(-1);
pxlWidth = string_width(GName)
if (string_width(string(YC)) > pxlWidth)
 {pxlWidth = string_width(string(YC))}
pxlHeight = string_height(GName)+string_height(string(YC))+8;

}
//--------------------If click, create .txt file that can be imported into excel
if (mouse_x > x) && (mouse_x < x+pxlWidth) && (mouse_y > y) && (mouse_y < y+pxlHeight)
{
```

```
 draw_set_alpha(.8);
 draw_set_color(c_blue);
 draw_rectangle(x-4,y-4,x+pxlWidth+4,y+pxlHeight+4,1);//draw background
 var mmm; mmm = 1;
 if (mouse_check_button_pressed(mb_right))
  {
  var strTemp;
  if (!save_at_end) {strTemp = "Save Data at Season's end";}
  else {strTemp = "Cancel Save at end"}
  mmm = show_menu/*_pos*/(/*mouse_x,mouse_y,*/"Save Snapshot of Data|"+strTemp,-1);
  if (mmm == 1)//toggle save at end
   {
   if (!save_at_end) {save_at_end = 1;}
   else {save_at_end = 0;}
   }
  }
 if (mouse_check_button_pressed(mb_left)) || (mmm == 0)//if left click
  {//--begin file writing process
  var fff,str;
  str = "";

  fff = file_text_open_write(uid+".txt");//clear any data previously in file
  file_text_write_string(fff,"");
  file_text_close(fff);

  objMouse.image_index = 1;
  for (i=-1;i<ind;i+=1)//write data to file
   {
   if (i==-1 && uid = "Drought_Record") {str = "0 is the best, if greater, then it is worse; 5 indicates user
variables being run"}
   else//if not drought recording graph or not first loop
    {
    if (i>-1)
     {str = string(XArray[i])+" "+string(YArray[i]);}
    else {str = "/s/s/s/s/s/"}
    }

   if (str != "/s/s/s/s/s/")//only write to file if either: is drought recorder (can be on the "-1st" loop) |or| if
not drought recorder and not on "-1st" loop
    {
    fff = file_text_open_append(uid+".txt");
    file_text_write_string(fff,str);
    file_text_close(fff);

    fff = file_text_open_append(uid+".txt");
    file_text_writeln(fff);
    file_text_close(fff);
```

```
   }
  }//end writing data loop
 instance_create(mouse_x,mouse_y,objSaved);//show "Saved" message
 }//--end process
}
//------------------end click code----------------------------------------
draw_set_alpha(aaa);//restore backup of color settings
draw_set_color(ccc);
```

Information about object: objSaved (object alerting a particular graph has saved its data)

Sprite: <no sprite>
Solid: false
Visible: true
Depth: 0
Persistent: false
Parent: <no parent>
Mask: <same as sprite>

Create Event:
execute code:

```
my_alpha = 1;
```

Draw Event:
execute code:

```
var aaa,ccc;
aaa = draw_get_alpha();//restore point
ccc = draw_get_color();

draw_set_color(c_white);
draw_rectangle(x,y,x+string_width("Saved Graph Data"),y+string_height("Saved Graph Data"),0);
draw_set_color(c_red);
draw_set_alpha(my_alpha);
draw_text(x,y,"Saved Graph Data");
y -= 4;
my_alpha -= .1;

draw_set_color(ccc);//restore defaults
draw_set_alpha(aaa);

if (my_alpha == 0)
```

```
{
instance_destroy();
}
```

Information about object: objHelp (the help button)

Sprite: sprHelp
Solid: false
Visible: true
Depth: 0
Persistent: false
Parent: <no parent>
Mask: <same as sprite>

Create Event:
execute code:

```
image_speed = 0;
my_imgind = 0;
```

Draw Event:
execute code:

```
if (position_meeting(mouse_x,mouse_y,self))
 {
 if (mouse_check_button_pressed(mb_any))//if clicked, set right image and show help
  {
  my_imgind = 2;
  show_info();
  }
 else//if not clicked, just make darker
  {my_imgind = 1;}
 }
else {my_imgind = 0;}//set to regular image when mouse isn't over

draw_sprite(sprite_index,my_imgind,x,y);//draw self with correct images
```

Information about object: objMouse (mouse that is in window)

Sprite: sprMouse
Solid: false
Visible: true
Depth: -1000
Persistent: false
Parent: <no parent>
Mask: <same as sprite>

Create Event:
execute code:

```
image_speed = 0;
image_index = 0;
counter = 0;
yesno = 0;//boolean controlling speed of arrows on sides
```

Mouse Event for Glob Left Button:
execute code:

```
if (position_meeting(mouse_x,mouse_y,objGraph))
 {
 image_index = 1;
 }
```

Draw Event:
execute code:

```
x = mouse_x;
y = mouse_y;
draw_sprite(sprite_index,image_index,x,y);
```

execute code:

```
var ccc;
ccc = draw_get_color();
draw_set_color(c_dkgray);
draw_line(view_wview,0,view_wview,view_hview);
draw_set_color(ccc);
```

execute code:

```
if yesno
 {
```

```
 if (counter < 4)
  {counter += 1;}
 else {counter = 0;}
 yesno = 0;
 }
else
 {yesno = 1;}

var aaa,midx,temp;
midx = (view_xview+view_xview+view_wview)/2;
aaa = draw_get_alpha();
temp = abs(midx - x)/((view_xview+view_wview)/2)
//draw_text(x+32,y,temp);
//draw_set_alpha(temp);
if (view_xview > 0)
 {
 //draw_sprite_ext(sprPointerL,counter,view_xview,y,1,1,0,c_white,temp);
 draw_sprite(sprPointerL,counter,view_xview,room_height-32);
 }
if (view_xview+view_wview < room_width)
 {
 //draw_sprite_ext(sprPointer,counter,view_xview+view_wview-48,y,1,1,0,c_white,temp);
 draw_sprite(sprPointer,counter,view_xview+view_wview-48,room_height-32);
 }
draw_set_alpha(aaa);
```

Information about object: objEnterInfo (button that executes variable grabber prgm)

Sprite: sprEnterInfo
Solid: false
Visible: true
Depth: 0
Persistent: false
Parent: <no parent>
Mask: <same as sprite>

Create Event:
execute code:

```
my_imgind = 0;
```

Draw Event:
execute code:

```
if (position_meeting(mouse_x,mouse_y,self))
 {
 if (mouse_check_button_pressed(mb_any))//if clicked, set right image and show help
  {
  my_imgind = 2;
  //execute "click" code here
  if (default_run)
  {
  if (secure_mode)//if in secure mode, give an alert
   {
   //show_message("This function is not available in secure mode. Please turn secure mode off to access
the programs full potential.");
   wd_message_simple("This function is not available in secure mode. Please turn secure mode off to
access the programs full potential.");
   }
  else//execute user variable getting program if not in secure mode
   {
   if (file_exists("UserVarGetSlave.exe"))//if variable getting program exists
    {
    /*if (!file_exists("BatchRASSCC01.bat"))//write batch file to execute prgm
     {*/
     var fff,str;
     fff = file_text_open_write("BatchRASSCC01.bat");
     str = "UserVarGetSlave.exe "+string(usrPercentmax)+" "+string(usrPercentlow)+"
"+string(usrWeeklyI)+" "+string(usrRainhigh)+" "+string(usrRainlow)+" "+string(usrGrainprice)+"
"+string(usrWatercost)+" "+string(usrFieldsize);
     file_text_write_string(fff,str);
     file_text_close(fff);
     /*}
    else
     {
     var ans;
     ans = show_question("BatchRASSCC01.bat already exists.  This is a file the program needs to
overwrite to operate correctly.  If this Batch file is important please hit 'no' and change its directory.  If
this is not important, hit 'yes'.  If program was shutdown unexpectedly, this is a common error and just
hit 'yes'.");
     if (ans)//if they say yes, overwrite file
      {
      var fff,str;
      fff = file_text_open_write("BatchRASSCC01.bat");
      str = "UserVarGetSlave.exe "+string(usrPercentmax)+" "+string(usrPercentlow)+"
"+string(usrWeeklyI)+" "+string(usrRainhigh)+" "+string(usrRainlow)+" "+string(usrGrainprice)+"
"+string(usrWatercost)+" "+string(usrFieldsize);
      file_text_write_string(fff,str);
      file_text_close(fff);
      }
     }*/
```

```
   var argTemp;

//usrPercentmax,usrPercentlow,usrWeeklyI,usrRainhigh,usrRainlow,usrGrainprice,usrWatercost,usrFiel
dsize
   //argTemp =
string(usrPercentmax)+"|"+string(usrPercentlow)+"|"+string(usrWeeklyI)+"|"+string(usrRainhigh)+"|"+s
tring(usrRainlow)+"|"+string(usrGrainprice)+"|"+string(usrWatercost)+"|"+string(usrFieldsize);
   //show_message(argTemp);//debug thing, get rid of after fixed
   //execute_program("UserVarGetSlave.exe",string(usrPercentmax) | string(usrPercentlow) |
string(usrWeeklyI) | string(usrRainhigh) | string(usrRainlow) | string(usrGrainprice) |
string(usrWatercost) | string(usrFieldsize)/*argTemp*/,1);//launch variable-obtaining prgm; pause until
it ends
   execute_program("BatchRASSCC01.bat","",1)

   if (file_exists("VarGetExe.UVGS"))//if code with variables to set exists, execute the code
    {
    execute_file("VarGetExe.UVGS");
    //file_delete("VarGetExe.UVGS")
    }
   else//if file with variables does not exist
    {
    //show_message("Error 503: No file containing variables exists.  If secure mode is on, please turn it
off.")
     wd_message_set_text("Error 503: No file containing variables exists.  If secure mode is on, please
turn it off.")
     wd_message_show(wd_mk_error,wd_mb_ok,wd_mb_none,wd_mb_none);
    }
   }
  else//if (var-getting) executable does not exist
   {
   //show_message("Error 504: No executable to obtain variables exists.  If secure mode is on, please
turn it off.")
   wd_message_set_text("Error 504: No executable to obtain variables exists.  If secure mode is on,
please turn it off.")
   wd_message_show(wd_mk_error,wd_mb_ok,wd_mb_none,wd_mb_none);
   }
  }
 }
 else {default_run = 1; scrDrought();}
 }
else//if not clicked, just make darker
 {my_imgind = 1;}
}
else {my_imgind = 0;}//set to regular image when mouse isn't over

draw_sprite(sprite_index,my_imgind,x,y);//draw self with correct images
```

```
/*if (!default_run)
 {
 var ccc;
 ccc = draw_get_color();
 draw_set_color(c_white);
 draw_rectangle(x,y-16,x+sprite_width,y-1,0);//draw rectangle background
 draw_set_color(c_black);
 draw_text(x,y-16,"Running User variables");
 draw_set_color(ccc);
 }*/
```

---

scrTempMain(): (Main script ran by objMain)

```
if (LOOPCOUNT == Daylength)//if delay between days is over
{
 if (SEASONDAYCOUNT == 1)
  {
  objIndicator.haveset = 0;
  WATERGRAPH = instance_create(0,room_height-320,objGraph);
  WATERGRAPH.uid = "Water_field_Graph";//used in file naming
  WATERGRAPH.GName = "Water in the field (in.)/Day";
  WATERGRAPH.pxlWidth = 540;
  //WATERGRAPH.image_alpha = .6;

  RAINGRAPH = instance_create(0,room_height-208,objGraph);
  RAINGRAPH.uid = "Rain_Graph";
  RAINGRAPH.GName = "Rain (in.)/Day";
  RAINGRAPH.pxlWidth = 540;

  IRRGRAPH = instance_create(0,room_height-96,objGraph);
  IRRGRAPH.uid = "Irrigation_Graph";
  IRRGRAPH.GName = "Irrigated Water (in.)/Day";
  IRRGRAPH.pxlWidth = 540;
  //now create total graphs
  if (FIRSTSEASON)
   {
   SWATERGRAPH = instance_create(542,room_height-448,objGraph);
   SWATERGRAPH.uid = "Season_Water_Graph";
   SWATERGRAPH.Permanent = 1;
   SWATERGRAPH.GName = "Total Water/Season";
   SWATERGRAPH.pxlWidth = 340;

   UWATERGRAPH = instance_create(542,room_height-336,objGraph);
   UWATERGRAPH.uid = "Season_Irrigation_Graph";
   UWATERGRAPH.Permanent = 1;
   UWATERGRAPH.GName = "Total Irrigated Water/Season";
   UWATERGRAPH.pxlWidth = 340;

   BUSHELGRAPH = instance_create(542,room_height-224,objGraph);
   BUSHELGRAPH.uid = "Bushel_Graph";
   BUSHELGRAPH.Permanent = 1;
   BUSHELGRAPH.GName = "Total number of bushels gained/Season";
   BUSHELGRAPH.pxlWidth = 340;
   BUSHELGRAPH.pxlHeight = 224;
   //------------------------------Set first values to zero---------------//
   SWATERGRAPH.NewX = 0;
   SWATERGRAPH.NewY = 0;
```

```
SWATERGRAPH.NewCoord = 1;

UWATERGRAPH.NewX = 0;
UWATERGRAPH.NewY = 0;
UWATERGRAPH.NewCoord = 1;

BUSHELGRAPH.NewX = 0;
BUSHELGRAPH.NewY = 0;
BUSHELGRAPH.NewCoord = 1;

MONEYGGRAPH = instance_create(882,room_height-224,objGraph);
MONEYGGRAPH.uid = "Revenue_Graph";
MONEYGGRAPH.Permanent = 1;
MONEYGGRAPH.GName = "Total revenue ($) per season";
MONEYGGRAPH.pxlWidth = room_width - MONEYGGRAPH.x;
MONEYGGRAPH.pxlHeight = 224;

MONEYLGRAPH = instance_create(882,room_height-464,objGraph);
MONEYLGRAPH.uid = "Money_Costs";
MONEYLGRAPH.Permanent = 1;
MONEYLGRAPH.GName = "Total costs ($) per season";
MONEYLGRAPH.pxlWidth = room_width - MONEYLGRAPH.x;
MONEYLGRAPH.pxlHeight = 224;

MONEYNGRAPH = instance_create(882,16,objGraph);
//show_message("Yay");
MONEYNGRAPH.tpText = 1;
MONEYNGRAPH.uid = "Net_Profit_Graph";
MONEYNGRAPH.Permanent = 1;
MONEYNGRAPH.GName = "Profit ($)";
MONEYNGRAPH.pxlWidth = room_width - MONEYNGRAPH.x;
MONEYNGRAPH.pxlHeight = room_height-496;

//inserted here
MONEYGGRAPH.NewX = 0;//Gross Profit graph
MONEYGGRAPH.NewY = 0;
MONEYGGRAPH.NewCoord = 1;

MONEYLGRAPH.NewX = 0;//Debit graph
MONEYLGRAPH.NewY = 0;
MONEYLGRAPH.NewCoord = 1;

MONEYNGRAPH.NewX = 0;//Net Profit graph
MONEYNGRAPH.NewY = 0;
MONEYNGRAPH.NewCoord = 1;

//-------special button graph to save Drought sequence
```

```
  INDICATORG = instance_create(0,room_height-380,objGraph);
  INDICATORG.Permanent = 1;
  INDICATORG.uid = "Drought_Record";//used in file naming
  INDICATORG.GName = "Drought Record on scale from 0-3";
  INDICATORG.pxlWidth = 32;
  INDICATORG.pxlHeight = 32;

  instance_create(32,56,objHelp);//create help button
  instance_create(objHelp.x+objHelp.sprite_width+16,objHelp.y,objEnterInfo);//create enter data object
below help button
  }
 R = 0;
 RR = 0;
 WR = 0;
 WI = 0;
 I = 0;
 SEASONWATER = 0;
 WATER = 0;

 INDICATORG.NewX = SEASONCOUNT;
 if (default_run) {INDICATORG.NewY = severity;}
 else {INDICATORG.NewY = 5;}//5 indicates user variable being run
 INDICATORG.NewCoord = 1;
 }
if (SEASONDAYCOUNT < 182)//if season still going on
 {
 SEASONDAYCOUNT += 1;
 //In a typical Day:
    var GRAPHI;//whether to graph irrigation level
    GRAPHI = 1;

    WATER = 0;
    //Rain--------------------
    ////Average Low Temperature, 27.9° F. Rainy Days, 13%. Average Rainfall, 9.2 inches ...
    var israin;
    if (default_run)
     {CHANCERAIN = irandom_range(LCHANCE, HCHANCE);}
    else
     {CHANCERAIN = irandom_range(usrPercentlow, usrPercentmax);}
    israin = random_range(0,100);
    if (israin < CHANCERAIN)//test percent chance of rain
     {
     //effect_create_above(ef_rain,0,0,4,c_dkgray);//Yay! it's raining!
     if (default_run)
      {RR = irandom_range(LOW,HIGH+1);} //how much rain in inches
     else {RR = irandom_range(usrRainlow,usrRainhigh+1);} //how much rain in inches
     WATER += RR;//add rain to water count in field
```

```
 R += RR;
 instance_create(objField.x,objField.y,objRain);//make it look like it rained
 WR += RR;
 }
else {RR = 0;WR += RR;}
//End of Rain----------


//water field----------
   //NEEDS is amt. in inches field needs per season
   //WEEKLY is (NEEDS*7)/182 (amt. watered per week w/o rain)
   if (DAYCOUNT == 0)//if first day of week
    {
    WI = 0;
    if (default_run)
     {
     ///////
      if (WR <= WEEKLY)
       {
       WI = (WEEKLY-WR);//Weekly is the amt. of water farmer needs to water, Water field amt.
needed (taking into account rain)
       WATER += WI;//add the farmer's addition to the amount of water in the field

       GRAPHI = 0;
       if (instance_exists(objGraph))
        {
        IRRGRAPH.NewX = SEASONDAYCOUNT;
        IRRGRAPH.NewY = WI;
        IRRGRAPH.NewCoord = 1;
        }
       I += WI; WI = 0;
       }//overall Irrigation add from weekly Irrigation
      ////////
      }
     else
      {
     ///////
      if (WR <= usrWeeklyI)
       {
       WI = (usrWeeklyI-WR);//Weekly is the amt. of water farmer needs to water, Water field amt.
needed (taking into account rain)
       //WI *= usrFieldsize;
       WATER += WI;//add the farmer's addition to the amount of water in the field

       GRAPHI = 0;
       if (instance_exists(objGraph))
        {
```

```
          IRRGRAPH.NewX = SEASONDAYCOUNT;
          IRRGRAPH.NewY = WI;
          IRRGRAPH.NewCoord = 1;
          }
         I += WI; WI = 0;
         }//overall Irrigation add from weekly Irrigation
        ////////
        }
       WR = 0;
       }
    //end of watering field-----


    //Draw?
    //End of Draw?

    TOTALWATER += WATER;
    SEASONWATER += WATER;
    //Potential = 5.8*(SM + R + I - 4.1);
    if (WATER > 0)
     {
     objField.LIGHT = (WEEKLY/(WATER+(WEEKLY*1.5)));//change field color to match how much water
was put in it.
     }
    else
     {
     objField.LIGHT = 1;//if completely dry, just plain orange
     }

  if (instance_exists(objGraph))
  {
  WATERGRAPH.NewX = SEASONDAYCOUNT;
  WATERGRAPH.NewY = WATER;
  WATERGRAPH.NewCoord = 1;

  RAINGRAPH.NewX = SEASONDAYCOUNT;
  RAINGRAPH.NewY = RR;
  RAINGRAPH.NewCoord = 1;

  if (GRAPHI)
   {
   IRRGRAPH.NewX = SEASONDAYCOUNT;
   IRRGRAPH.NewY = WI;
   IRRGRAPH.NewCoord = 1;
   }
  }
```

```
 LOOPCOUNT = 0;//reset number controlling how long till the next day

 if (DAYCOUNT < 6) {DAYCOUNT += 1;}//control day of the week
 else {DAYCOUNT = 0;}
 TOTALDAYS += 1;//the total amount of days farm has been working
 draw_set_color(c_white);
 draw_set_alpha(1);
 draw_text(32,32,/*string(SEASONWATER)+","+string(WATER)+*/"Total days:"+string(TOTALDAYS));

 //End of day
 }
else//if season is over
 {
 COLORARR[SEASONCOUNT] = objIndicator.my_col;//save drought severity to array
 objIndicator.haveset = 0;
 //calculate water loss, grain profit
 if (default_run)
  {
  Potential = 5.8*(SM + SEASONWATER - 4.1);
  }
 else
  {
  Potential = 5.8*(SM + SEASONWATER - 4.1)*usrFieldsize;
  }

 //calculate money gain and loss
 if (default_run)
  {
  Profit = Potential * 5.5;//Profit gained
  Debit = ((6272640 * I)/231)*(1.5/1000);//inches in acre squared * inches of water irrigated /
conversion to gallons * conversion to money
  NetProfit += (Profit-Debit);
  }
 else
  {
  Profit = Potential * 5.5;//Profit gained
  Debit = (((6272640 * usrFieldsize) * I)/231)*(1.5/1000);//inches in acre squared * inches of water
irrigated / conversion to gallons * conversion to money
  NetProfit += (Profit-Debit);
  }

 //show_message("Amt. of water field got: "+string(SEASONWATER)+"#Was gained in
"+string(SEASONDAYCOUNT)+" days#Water Loss: "+string(I));
 //show_message("You earned "+string(Potential)+" bushels of wheat#In season
"+string((SEASONCOUNT+1)));

 if (instance_exists(objGraph))//send data to graphs
```

```
 {
 SWATERGRAPH.NewX = SEASONCOUNT+1;//season water graph
 SWATERGRAPH.NewY = SEASONWATER;
 SWATERGRAPH.NewCoord = 1;

 UWATERGRAPH.NewX = SEASONCOUNT+1;//irrigated (used) water graph
 if (!default_run) {I *= usrFieldsize;}
 UWATERGRAPH.NewY = I;
 UWATERGRAPH.NewCoord = 1;

 BUSHELGRAPH.NewX = SEASONCOUNT+1;//amount of bushels graph
 BUSHELGRAPH.NewY = Potential;
 BUSHELGRAPH.NewCoord = 1;

 MONEYGGRAPH.NewX = SEASONCOUNT+1;//Gross Profit graph
 MONEYGGRAPH.NewY = Profit;
 MONEYGGRAPH.NewCoord = 1;

 MONEYLGRAPH.NewX = SEASONCOUNT+1;//Debit graph
 MONEYLGRAPH.NewY = Debit;
 MONEYLGRAPH.NewCoord = 1;

 MONEYNGRAPH.NewX = SEASONCOUNT+1;//Net Profit graph
 MONEYNGRAPH.NewY = NetProfit;
 MONEYNGRAPH.NewCoord = 1;
 }

SEASONDAYCOUNT = 0;
SEASONCOUNT += 1;//Add to number of growing seasons that have been on the farm
DAYCOUNT = 0;//reset number of days in growing season
DROUGHT = choose(0,1);//set DROUGHT to random
if (!DROUGHT)
 {
 severity = 0;
 LOW = .5;
 HIGH = 1.5;
 CHANCERAIN = 13;
 }//if it's not a drought, set drought severity to 0
else {scrDrought();}

with (objGraph)//destroy graphs
 {
 if (!Permanent)//if it's not the total graphs
  {
  instance_destroy();
  }
 else//if a permanent graph, check if needs to save data
```

```
   {
   if (save_at_end)
    {
    var fff,str;
    str = "";

    fff = file_text_open_write(uid+".txt");//clear any data previously in file
    file_text_write_string(fff,"");
    file_text_close(fff);
    for (i=0;i<ind;i+=1)//write data to file
     {
     str = string(XArray[i])+" "+string(YArray[i]);//

     fff = file_text_open_append(uid+".txt");
     file_text_write_string(fff,str);
     file_text_close(fff);

     fff = file_text_open_append(uid+".txt");
     file_text_writeln(fff);
     file_text_close(fff);
     }//end writing data loop
    instance_create(mouse_x,mouse_y,objSaved);//show "Saved" message
    save_at_end = 0;
    }
   }
  }

  FIRSTSEASON = 0;
  }
}
else//the steps that are not one of the "days"
 {
 LOOPCOUNT += 1;
 draw_set_color(c_white);
 draw_set_alpha(1);
 draw_text(32,32,/*string(TOTALWATER)+","+string(WATER)+*/"Total days:"+string(TOTALDAYS));
 }
```

UserVarGetSlave.exe (variable grabber program (slave program, used by main or master program))

scrMain(): (used by objMain)

```
if (firstrun)
 {

//usrPercentmax,usrPercentlow,usrWeeklyI,usrRainhigh,usrRainlow,usrGrainprice,usrWatercost,usrFiel
dsize
 var i,ii,iii,iv,v,vi,vii,viii;
 i = instance_create(32,32,clsButton);
 i.title = "Max. Percent Chance it can rain each day";
 i.number = 1;
 i.default_val = real(parameter_string(1));

 ii = instance_create(32,96,clsButton);
 ii.title = "Min. Percent Chance it can rain each day";
 ii.number = 2;
 ii.default_val = real(parameter_string(2));

 iii = instance_create(32,160,clsButton);
 iii.title = "Amount Farmer waters each week (inches/acre)";
 iii.number = 3;
 iii.default_val = real(parameter_string(3));

 iv = instance_create(32,224,clsButton);
 iv.title = "Max. amount it is possible to rain each time it rains (inches)";
 iv.number = 4;
 iv.default_val = real(parameter_string(4));

 v = instance_create(32,288,clsButton);
 v.title = "Min. amount it is possible to rain each time it rains (inches)";
 v.number = 5;
 v.default_val = real(parameter_string(5));

 vi = instance_create(32,352,clsButton);
 vi.title = "Grain Price per Bushel (dollars)";
 vi.number = 6;
 vi.default_val = real(parameter_string(6));

 globalvar pxlLeftWidth;//get x-coordinate right column should be placed
 pxlLeftWidth = string_width("Max. amount it is possible to rain each time it rains (inches)")+32;

 vii = instance_create(pxlLeftWidth,32,clsButton);
 vii.title = "Water price per Thousand Gallons (dollars/thousand gallons)";
```

```
 vii.number = 7;
 vii.default_val = real(parameter_string(7));

 viii = instance_create(pxlLeftWidth,96,clsButton);
 viii.title = "Field size (acres)";
 viii.number = 8;
 viii.default_val = real(parameter_string(8));
 }


firstrun = 0;
```

Information about object: objMain

Sprite: <no sprite>
Solid: false
Visible: true
Depth: 0
Persistent: false
Parent: <no parent>
Mask: <same as sprite>

Create Event:
execute code:

```
//usrPercentmax,usrPercentlow,usrWeeklyI,usrRainhigh,usrRainlow,usrGrainprice,usrWatercost,usrFiel
dsize
globalvar myParam,firstrun;
for (i=1;i<=parameter_count();i+=1)
 {
 myParam[i] = real(parameter_string(i));
 }
firstrun = 1;
```

Other Event: Game End:
execute code:

```
var fff,tempCode;
tempCode = "usrPercentmax = "+string(myParam[1])+"; usrPercentlow = "+string(myParam[2])+";
usrWeeklyI = "+string(myParam[3])+"; usrRainhigh = "+string(myParam[4])+"; usrRainlow =
"+string(myParam[5])+"; usrGrainprice = "+string(myParam[6])+"; usrWatercost =
"+string(myParam[7])+"; usrFieldsize = "+string(myParam[8])+"; default_run = 0;";//set string to write to
file for "mother" to execute; sets correct variables in mother
fff = file_text_open_write("VarGetExe.UVGS");//open file for writing, creating it if necessary
```

```
file_text_write_string(fff,tempCode);
file_text_close(fff);
```

Draw Event:
execute code:

```
scrMain();
var str,i;
str = "";
for (i=1;i<=parameter_count();i+=1)
 {
 str += " "+parameter_string(i);
 }
draw_text(x,y,str);
```

Information about object: clsButton

Sprite: sprButton
Solid: false
Visible: true
Depth: 0
Persistent: false
Parent: <no parent>
Mask: <same as sprite>

Create Event:
execute code:

```
//if (!variable_local_exists("default_val")) {default_val = 0;}
if (!variable_local_exists("title")) {title = "";}//just a title saying what variable is being changed
checked = 0;
//new_val = default_val;
set = 0;
```

Begin Step Event:
execute code:

```
if (!set)//if haven't checked for number setting order and parameter, do so
 {
 //the commented code is error causing, delete soon...
 /*if (!variable_local_exists("number"))
  {
  number = 1;//just a title saying what variable is being changed
  }
```

```
 default_val = myParam[number];*/
 new_val = default_val;

 set = 1;//set checker for "" (see top line) to "has been set"
 }
```

Draw Event:

execute code:

```
if (!set)//if haven't checked for number setting order and parameter, do so
 {

 if (!variable_local_exists("number"))
  {
  number = 1;//just a title saying what variable is being changed
  }
 default_val = myParam[number];
 new_val = default_val;

 set = 1;//set checker for "" (see top line) to "has been set"
 }
```

execute code:

```
draw_text(x,y-16,title);//draw title above button
if (mouse_check_button_pressed(mb_any)) && (position_meeting(mouse_x,mouse_y,self))//if user
clicks on button
 {
 checked = !checked;
 if (checked)
  {
  var str,ttt;
  str = "0";//string which will be converted to a number to be the new value
  ttt = 0;//amt. it looped through getting a number

  do//get string to be converted to a number, repeat until all attributes are valid
   {
   if (ttt > 0)
    {
    //show_message("Must be greater than 0!");
    wd_message_set_text("New Value must be greater than 0");
    wd_message_show(wd_mk_warning,wd_mb_ok,wd_mb_none,wd_mb_none);//show warning
message
    }
   //str = get_string("Enter a number greater than 0 for "+title,string(new_val));
```

```
   str = wd_input_box("New Value","Enter a number greater than or equal to 0 for:
"+title,string(new_val));
   ttt += 1;
   } until (real(str) >= 0);

  new_val = real(str);
  myParam[number] = new_val;
  }
 }
draw_sprite(sprite_index,image_index,x,y);//draw self
if (checked)
 {
 draw_sprite(sprCheck,0,x,y);//if checked, draw check mark
 draw_text(x,y+sprite_height,string(new_val));
 }
else
 {
 myParam[number] = default_val;
 draw_text(x,y+sprite_height,string(default_val));
 }
```

Appendix B: Graphs of data