extensions [ GIS ]

globals [

Yield Yield-A Yield-C EC EC-A EC-C CS Water Water-A Water-C Biodiversity LS ESS-type-list D-1 D-2 D-3 CSA output_file actors duration month-num

]

breed [farmers farmer]

farmers-own [user-id]

patches-own [

r b g label-1 label-2 label-3 city-ID S-yield S-ec S-cs S-w S-bio S-ESS D-yield D-yield-A D-yield-C D-ec D-ec-A D-ec-C D-cs D-cs-B D-w D-w-A D-w-C

D-bio D-bio-B D-A D-B D-C D-N D-all D-yield-A-ID D-yield-C-ID D-ec-A-ID D-ec-C-ID D-cs-B-ID D-w-A-ID D-w-C-ID D-bio-B-ID D-A-ID D-B-ID D-C-ID

G-ESS G-A G-B G-C Conf L g2 n f K-1 K-2 K-3 K-4 K-5 K-6 K-7 K-8 K-9 K-10 K-11 K-12 K-13 K-14 K-15 d_x d_y d_z d_l d_m wt-1 wt-2 wt-3 wt-4 wt-5 wt-6

wt-7 wt-8 wt-9 wt-10 wt-11 wt-12 wt-13 wt-14 wt-15 r_x r_y r_z r_l r_m MO-data-effort MO-data-utility MO-data-utility-elasticity efficiency norm_efficiency

maximum-parameters parameter-list q norm_q efforts values elasticity norm-C norm-a norm-a-tot norm-C-tot a-share C-share q_x q_y q_z q_l q_m

C-1 C-2 C-3 C-4 C-5 C-6 C-7 C-8 C-9 C-10 C-11 C-12 C-13 C-14 C-15 a-1 a-2 a-3 a-4 a-5 a-6 a-7 a-8 a-9 a-10 a-11 a-12 a-13 a-14 a-15

b2-1 b2-2 b2-3 b2-4 b2-5 b2-6 b2-7 b2-8 b2-9 b2-10 b2-11 b2-12 b2-13 b2-14 b2-15 a b2 K C C-x C-y C-z C-l C-m K-0 C-fin C-soc C-phy C-hum C-cul

K-fin K-soc K-phy K-hum K-cul K0-1 K0-2 K0-3 K0-4 K0-5 K0-6 K0-7 K0-8 K0-9 K0-10 K0-11 K0-12 K0-13 K0-14 K0-15 K-plot C-plot

x_per_x x_per_y x_per_z x_per_l x_per_m c_x c_y c_z c_l c_m harv_x harv_y harv_z harv_l harv_m u_x u_y u_z u_l u_m u f_x f_y f_z f_l f_m

w1_x w2_x w3_x w1_y w2_y w3_y w1_z w2_z w3_z w1_l w2_l w3_l w1_m w2_m w3_m w_x w_y w_z w_l w_m sum_w_c_x sum_w_c_y sum_w_c_z sum_w_c_l sum_w_c_m

w1 w2 w3 w sum_w_c V V-fin V-soc V-phy V-hum V-cul V-1 V-2 V-3 V-4 V-5 V-6 V-7 V-8 V-9 V-10 V-11 V-12 V-13 V-14 V-15 V-plot C_max v_x v_y v_z v_l v_m

growth tot_harv Price P-fin P-soc P-phy P-hum P-cul P-1 P-2 P-3 P-4 P-5 P-6 P-7 P-8 P-9 P-10 P-11 P-12 P-13 P-14 P-15 Price-plot f_sum f_targ C_targ

r_x_targ r_y_targ r_z_targ r_l_targ r_m_targ phi_x phi_y phi_z phi_l phi_m rv_x rv_y rv_z rv_l rv_m sum_rv tmp tmp2

]

```
;; Future work: integrate qualitative scenarios evaluation; showing the supply-driven demand
adaptation dynamics; displaying capitals in space; include real demand data and perceived supply,
update supply data.

;;#############################################################################

to load-map

 if Resolution = "500 m"   [ set-patch-size 7  resize-world -33  33  -33  33 ] ;

 if Resolution = "1 km"    [ set-patch-size 14 resize-world -16  16  -16  16 ]

 if Resolution = "1 ha"    [ set-patch-size 1.4  resize-world -167 167 -167 167] ; 100m resolution

  clear-all

  ask patches [set pcolor 9]


;; all supply and demand maps have the projection of: WGS 1984 UTM Zone 33N


  if Region = "MOL" [

    set CSA gis:load-dataset "Spatial/CLC/MOL_prj.shp" ]

  if Region = "OPR" [

    set CSA gis:load-dataset "Spatial/CLC/OPR_prj.shp" ]

  if Region = "UMK" [

    set CSA gis:load-dataset "Spatial/CLC/UMK_prj.shp" ]


    gis:set-world-envelope gis:envelope-of (CSA)

    gis:set-drawing-color 4

    gis:draw CSA 1


    gis:apply-coverage CSA "R" r

    gis:apply-coverage CSA "B" b

    gis:apply-coverage CSA "G" g

    gis:apply-coverage CSA "CLC2000__2" label-1

    gis:apply-coverage CSA "CLC2000__3" label-2

    gis:apply-coverage CSA "CLC2000__4" label-3

    gis:apply-coverage CSA "OBJECTID" city-ID
```

```
if Region = "MOL" [
  set Yield       gis:load-dataset "Demand/Yield_prj.shp"
  set Yield-A     gis:load-dataset "Demand/Yield-A_prj.shp"
  set Yield-C     gis:load-dataset "Demand/Yield-C_prj.shp"
  set EC          gis:load-dataset "Demand/Erosion_prj.shp"
  set EC-A        gis:load-dataset "Demand/Erosion-A_prj.shp"
  set EC-C        gis:load-dataset "Demand/Erosion-C_prj.shp"
  set CS          gis:load-dataset "Demand/CS_prj.shp"
  set Water       gis:load-dataset "Demand/Water_prj.shp"
  set Water-A     gis:load-dataset "Demand/Water-A_prj.shp"
  set Water-C     gis:load-dataset "Demand/Water-C_prj.shp"
  set Biodiversity gis:load-dataset "Demand/Bio_prj.shp"

  gis:apply-raster (gis:load-dataset "Spatial/Supply/Yield_MOL_prj.asc")    S-yield
  gis:apply-raster (gis:load-dataset "Spatial/Supply/EC_water_MOL_prj.asc") S-ec
  gis:apply-raster (gis:load-dataset "Spatial/Supply/CS_MOL_prj.asc")       S-cs
  gis:apply-raster (gis:load-dataset "Spatial/Supply/Water_MOL_prj.asc")    S-w
  gis:apply-raster (gis:load-dataset "Spatial/Supply/Bio_MOL_prj.asc")      S-bio

  gis:apply-coverage Yield       "DEMAND" D-yield
  gis:apply-coverage Yield-A     "DEMAND" D-yield-A
  gis:apply-coverage Yield-C     "DEMAND" D-yield-C
  gis:apply-coverage EC          "DEMAND" D-ec
  gis:apply-coverage EC-A        "DEMAND" D-ec-A
  gis:apply-coverage EC-C        "DEMAND" D-ec-C
  gis:apply-coverage CS          "DEMAND" D-cs-B
  gis:apply-coverage CS          "DEMAND" D-cs
  gis:apply-coverage Water       "DEMAND" D-w
  gis:apply-coverage Water-A     "DEMAND" D-w-A
  gis:apply-coverage Water-C     "DEMAND" D-w-C
  gis:apply-coverage Biodiversity "DEMAND" D-bio-B
  gis:apply-coverage Biodiversity "DEMAND" D-bio
```

```
  gis:apply-coverage Yield-A      "ID" D-yield-A-ID

  gis:apply-coverage Yield-C      "ID" D-yield-C-ID

  gis:apply-coverage EC-A         "ID" D-ec-A-ID

  gis:apply-coverage EC-C         "ID" D-ec-C-ID

  gis:apply-coverage CS           "ID" D-cs-B-ID

  gis:apply-coverage Water-A      "ID" D-w-A-ID

  gis:apply-coverage Water-C      "ID" D-w-C-ID

  gis:apply-coverage Biodiversity "ID" D-bio-B-ID


  set LS patches with [ city-ID >= 0 ]

  set Interaction-scenario "Select" set Land-management "Select" set Plot-title "Select" set Actor
"Select"

  set ESS "Select" set Demand_Scenario "Select" set site "Select" set Intersecting-actors "Select" set
Map-display "Select" set Map-type "Select" set Interact "Select" set Capital "Select" set Month
"Select"

  set Demand 0 set Match-limit 1


  set ESS-type-list n-values 5       [ ?1 -> ?1 ]

  set D-1 n-values 5 [0.1]

  set D-2 n-values 5 [0.1]

  set D-3 n-values 5 [0.1]


  ask LS [

    set S-ESS n-values 5 [ ?1 -> ?1 ]

    set S-ESS replace-item 0 S-ESS (S-yield)

    set S-ESS replace-item 1 S-ESS (S-ec)

    set S-ESS replace-item 2 S-ESS (S-cs)

    set S-ESS replace-item 3 S-ESS (S-w)

    set S-ESS replace-item 4 S-ESS (S-bio)

    set tmp n-values 5 [0.1]

    set D-N n-values 5 [0.1]

    set G-ESS   n-values 5 [0]
```

```
  if item 0 S-ESS = -3.4028234663852888 or item 0 S-ESS = 0 [set S-ESS replace-item 0 S-ESS (0)]
  if item 1 S-ESS = -3.4028234663852888 or item 1 S-ESS = 0 [set S-ESS replace-item 1 S-ESS (0)]
  if item 2 S-ESS = -3.4028234663852888 or item 2 S-ESS = 0 [set S-ESS replace-item 2 S-ESS (0)]
  if item 3 S-ESS = -3.4028234663852888 or item 3 S-ESS = 0 [set S-ESS replace-item 3 S-ESS (0)]
  if not (item 4 S-ESS < 0 or item 4 S-ESS > 0 or item 4 S-ESS = 0) or item 4 S-ESS = 0 [set S-ESS
replace-item 4 S-ESS (5)]
    if  not (D-yield-A < 0 or D-yield-A > 0 or D-yield-A = 0) [set D-yield-A 0.1]
    if  not (D-yield-C < 0 or D-yield-C > 0 or D-yield-C = 0) [set D-yield-C 0.1]
    if  not (D-yield   < 0 or D-yield   > 0 or D-yield   = 0) [set D-yield 0.1]
    if  not (D-ec      < 0 or D-ec      > 0 or D-ec      = 0) [set D-ec 0.1]
    if  not (D-ec-A    < 0 or D-ec-A    > 0 or D-ec-A    = 0) [set D-ec-A 0.1]
    if  not (D-ec-C    < 0 or D-ec-C    > 0 or D-ec-C    = 0) [set D-ec-C 0.1]
    if  not (D-cs-B    < 0 or D-cs-B    > 0 or D-cs-B    = 0) [set D-cs-B 0.1]
    if  not (D-cs      < 0 or D-cs      > 0 or D-cs      = 0) [set D-cs 0.1]
    if  not (D-w       < 0 or D-w       > 0 or D-w       = 0) [set D-w 0.1]
    if  not (D-w-A     < 0 or D-w-A     > 0 or D-w-A     = 0) [set D-w-A 0.1]
    if  not (D-w-C     < 0 or D-w-C     > 0 or D-w-C     = 0) [set D-w-C 0.1]
    if  not (D-bio-B   < 0 or D-bio-B   > 0 or D-bio-B   = 0) [set D-bio-B 0.1]
    if  not (D-bio     < 0 or D-bio     > 0 or D-bio     = 0) [set D-bio 0.1]

    if  not (D-yield-A-ID < 0 or D-yield-A-ID > 0 or D-yield-A-ID = 0) [set D-yield-A-ID 0.1]
    if  not (D-yield-C-ID < 0 or D-yield-C-ID > 0 or D-yield-C-ID = 0) [set D-yield-C-ID 0.1]
    if  not (D-ec-A-ID    < 0 or D-ec-A-ID    > 0 or D-ec-A-ID    = 0) [set D-ec-A-ID 0.1]
    if  not (D-ec-C-ID    < 0 or D-ec-C-ID    > 0 or D-ec-C-ID    = 0) [set D-ec-C-ID 0.1]
    if  not (D-cs-B-ID    < 0 or D-cs-B-ID    > 0 or D-cs-B-ID    = 0) [set D-cs-B-ID 0.1]
    if  not (D-w-A-ID     < 0 or D-w-A-ID     > 0 or D-w-A-ID     = 0) [set D-w-A-ID 0.1]
    if  not (D-w-C-ID     < 0 or D-w-C-ID     > 0 or D-w-C-ID     = 0) [set D-w-C-ID 0.1]
    if  not (D-bio-B-ID   < 0 or D-bio-B-ID   > 0 or D-bio-B-ID   = 0) [set D-bio-B-ID 0.1]

    set D-A     n-values 5 [0.1]
    set D-B     n-values 5 [0.1]
```

```
set D-C     n-values 5 [0.1]
set D-all   n-values 5 [0.1]

set D-A-ID n-values 5 [0.1]
set D-B-ID n-values 5 [0.1]
set D-C-ID n-values 5 [0.1]

set D-A replace-item 0 D-A (D-yield-A)
set D-A replace-item 1 D-A (D-ec-A)
set D-A replace-item 3 D-A (D-w-A)

set D-B replace-item 2 D-B (D-cs-B)
set D-B replace-item 4 D-B (D-bio-B)

set D-C replace-item 0 D-C (D-yield-C)
set D-C replace-item 1 D-C (D-ec-C)
set D-C replace-item 3 D-C (D-w-C)

set D-all replace-item 0 D-all (D-yield)
set D-all replace-item 1 D-all (D-ec)
set D-all replace-item 2 D-all (D-cs)
set D-all replace-item 3 D-all (D-w)
set D-all replace-item 4 D-all (D-bio)

set D-A-ID replace-item 0 D-A-ID (D-yield-A-ID)
set D-A-ID replace-item 1 D-A-ID (D-ec-A-ID)
set D-A-ID replace-item 3 D-A-ID (D-w-A-ID)

set D-B-ID replace-item 2 D-B-ID (D-cs-B-ID)
set D-B-ID replace-item 4 D-B-ID (D-bio-B-ID)

set D-C-ID replace-item 0 D-C-ID (D-yield-C-ID)
```

```
    set D-C-ID replace-item 1 D-C-ID (D-ec-C-ID)
    set D-C-ID replace-item 3 D-C-ID (D-w-C-ID)
 ]]
end

;;#############################################################################
to Show-supply
 clear-drawing
 ask LS [
 set pcolor 9

  if ESS = "yield"          [set tmp2 precision item 0 S-ESS 2 set pcolor scale-color red    item 0 S-ESS 120 0  ]
  if ESS = "erosion_control" [set tmp2 precision item 1 S-ESS 2 set pcolor scale-color lime   item 1 S-ESS 120 0  ]
  if ESS = "carbon_seq"     [set tmp2 precision item 2 S-ESS 2 set pcolor scale-color yellow item 2 S-ESS 120 0  ]
  if ESS = "water"          [set tmp2 precision item 3 S-ESS 2 set pcolor scale-color blue   item 3 S-ESS 120 0  ]
  if ESS = "biodiversity"   [set tmp2 precision item 4 S-ESS 2 set pcolor scale-color orange item 4 S-ESS 120 0  ]
  ]
 if ESS = "yield"          [ clear-output output-print  "Source: BGR 2013  " output-print "(Die Bundesanstalt für Geowissenschaften und Rohstoffe)"]
 if ESS = "erosion_control" [ clear-output output-print  "Source: JRC ESDAC 2015" output-print "(JOINT RESEARCH CENTRE EUROPEAN SOIL DATA CENTRE) "]
 if ESS = "carbon_seq"     [ clear-output output-print  "Source: JRC ESDAC 2004" output-print "(JOINT RESEARCH CENTRE EUROPEAN SOIL DATA CENTRE) "]
 if ESS = "water"          [ clear-output output-print  "Source: BGR 2015" output-print "(Die Bundesanstalt für Geowissenschaften und Rohstoffe) "]
 if ESS = "biodiversity"   [ clear-output output-print  "Source: EEA 2012" output-print "(The European Environment Agency) "]
 end
;;#############################################################################
```

```
 to Show-demand
   clear-drawing
   ask LS [
   set pcolor 9
   foreach ESS-type-list [ ?1 ->
   if Actor = "A"         [set tmp replace-item ?1 tmp (item ?1 D-A  )]
   if Actor = "B"         [set tmp replace-item ?1 tmp (item ?1 D-B  )]
   if Actor = "C"         [set tmp replace-item ?1 tmp (item ?1 D-C  )]
   if Actor = "New-user"   [set tmp replace-item ?1 tmp (item ?1 D-N  )]
   if Actor = "all"       [set tmp replace-item ?1 tmp (item ?1 D-all)] ]
   display-ess

   if ESS = "mix" and item 0 tmp = max tmp [ set tmp2 precision item 0 tmp 2 set pcolor scale-color red    item 0 tmp 120 0 ]
   if ESS = "mix" and item 1 tmp = max tmp [ set tmp2 precision item 1 tmp 2 set pcolor scale-color lime   item 1 tmp 120 0 ]
   if ESS = "mix" and item 2 tmp = max tmp [ set tmp2 precision item 2 tmp 2 set pcolor scale-color yellow item 2 tmp 120 0 ]
   if ESS = "mix" and item 3 tmp = max tmp [ set tmp2 precision item 3 tmp 2 set pcolor scale-color blue   item 3 tmp 120 0 ]
   if ESS = "mix" and item 4 tmp = max tmp [ set tmp2 precision item 4 tmp 2 set pcolor scale-color orange item 4 tmp 120 0 ]

   ]

ask LS [ ; to capture the demand of the three sites of each actor from a patch variable to global variable
   foreach ESS-type-list [ ?1 ->
     if Actor = "A" [
    if [item ?1 D-A-ID] of one-of LS = 0 [set D-1 replace-item ?1 D-1 ([item ?1 tmp] of one-of LS with [item ?1 D-A-ID = 0])]
    if [item ?1 D-A-ID] of one-of LS = 1 [set D-2 replace-item ?1 D-2 ([item ?1 tmp] of one-of LS with [item ?1 D-A-ID = 1])]
    if [item ?1 D-A-ID] of one-of LS = 2 [set D-3 replace-item ?1 D-3 ([item ?1 tmp] of one-of LS with [item ?1 D-A-ID = 2])]]
```

```
    if Actor = "B" [

    if [item ?1 D-B-ID] of one-of LS = 0 [set D-1 replace-item ?1 D-1 ([item ?1 tmp] of one-of LS with [item ?1 D-B-ID = 0])]

    if [item ?1 D-B-ID] of one-of LS = 1 [set D-2 replace-item ?1 D-2 ([item ?1 tmp] of one-of LS with [item ?1 D-B-ID = 1])]

    if [item ?1 D-B-ID] of one-of LS = 2 [set D-3 replace-item ?1 D-3 ([item ?1 tmp] of one-of LS with [item ?1 D-B-ID = 2])]]

    if Actor = "C" [

    if [item ?1 D-C-ID] of one-of LS = 0 [set D-1 replace-item ?1 D-1 ([item ?1 tmp] of one-of LS with [item ?1 D-C-ID = 0])]

    if [item ?1 D-C-ID] of one-of LS = 1 [set D-2 replace-item ?1 D-2 ([item ?1 tmp] of one-of LS with [item ?1 D-C-ID = 1])]

  ifelse [item ?1 D-C-ID] of one-of LS = 2 [set D-3 replace-item ?1 D-3 ([item ?1 tmp] of one-of LS with [item ?1 D-C-ID = 2])][set D-3 replace-item ?1 D-3 (0.1)]]

   ]]


 if ESS = "yield" [

   clear-output

   output-print word "Site 1:" item 0 D-1 output-print word "Site 2:" item 0 D-2  output-print word "Site 3:" item 0 D-3]


if ESS = "erosion_control"  [

   clear-output

   output-print word "Site 1:" item 1 D-1 output-print word "Site 2:" item 1 D-2 output-print word "Site 3:" item 1 D-3]


if ESS = "carbon_seq" [

   clear-output

   output-print word "Site 1:" item 2 D-1 output-print word "Site 2:" item 2 D-2 output-print word "Site 3:" item 2 D-3]


if ESS = "water" [

   clear-output

   output-print word "Site 1:" item 3 D-1 output-print word "Site 2:" item 3 D-2 output-print word "Site 3:" item 3 D-3]
```

```
if ESS = "biodiversity" [

   clear-output

   output-print word "Site 1:" item 4 D-1 output-print word "Site 2:" item 4 D-2  output-print word "Site 3:" item 4 D-3]


end

;;#############################################################################################

  to Show-gap

   ask LS [

    set pcolor 9


    foreach ESS-type-list[ ?1 ->

    set G-ESS   replace-item ?1 G-ESS   (item ?1 S-ESS / item ?1 tmp  )


     if item ?1 G-ESS >= 10 or item ?1 G-ESS <= 0.01 or item ?1 S-ESS = 0 [set G-ESS   replace-item ?1 G-ESS   (0) ]


   ]

   if ESS = "yield"          [ set tmp2 precision item 0 G-ESS 2 set pcolor scale-color cyan    item 0 G-ESS 2 0.1]

   if ESS = "erosion_control" [ set tmp2 precision item 1 G-ESS 2 set pcolor scale-color cyan    item 1 G-ESS 2 0.1]

   if ESS = "carbon_seq"      [ set tmp2 precision item 2 G-ESS 2 set pcolor scale-color cyan    item 2 G-ESS 2 0.1]

   if ESS = "water"          [ set tmp2 precision item 3 G-ESS 2 set pcolor scale-color cyan    item 3 G-ESS 2 0.1]

   if ESS = "biodiversity"    [ set tmp2 precision item 4 G-ESS 2 set pcolor scale-color cyan    item 4 G-ESS 2 0.1]


    ]

 end

;;*****************************************************************************************
```

```
to Show-conflict


  ask LS [

    set pcolor white


    if (item 0 D-A > 0.1 and item 4 D-B > 0.1) or (item 0 D-C > 0.1 and item 4 D-B > 0.1) [

      set Conf abs (item 0 S-ESS - item 4 S-ESS)

      set tmp2 precision Conf 2              ;;****Added to export map

      set pcolor scale-color violet Conf 100 0]

  ]


end

;;*****************************************************************************

to Show-match

  ask LS [

    set pcolor 9

    foreach ESS-type-list [ ?1 ->

   set tmp replace-item ?1 tmp (0.1)

   if item ?1 S-ESS = item ?1 D-1 or item ?1 S-ESS > Match-limit * item ?1 D-1 [set tmp replace-item ?1 tmp (Match-limit * item ?1 D-1)]

   if item ?1 S-ESS = item ?1 D-2 or item ?1 S-ESS > Match-limit * item ?1 D-2 [set tmp replace-item ?1 tmp (Match-limit * item ?1 D-2)]

   if item ?1 S-ESS = item ?1 D-3 or item ?1 S-ESS > Match-limit * item ?1 D-3 [set tmp replace-item ?1 tmp (Match-limit * item ?1 D-3)]

   ]

   display-ess

  ]


end


;;*****************************************************************************
```

```
to Show-Pot-Demand
  Show-Demand

 ask LS [
   set pcolor 9
   foreach ESS-type-list [ ?1 ->

  if Demand_Scenario = "D-1"    and item ?1 tmp = 0.1   [set tmp replace-item ?1 tmp (item ?1 D-1)]
  if Demand_Scenario = "D-2"    and item ?1 tmp = 0.1   [set tmp replace-item ?1 tmp (item ?1 D-2)]
  if Demand_Scenario = "D-3"    and item ?1 tmp = 0.1   [set tmp replace-item ?1 tmp (item ?1 D-3)]
  if Demand_Scenario = "average" and item ?1 tmp = 0.1   [set tmp replace-item ?1 tmp ((item ?1 D-1
+ item ?1 D-2 + item ?1 D-3) / 3) ] ]

  display-ess
 ]

end

;;*****************************************************************************

to Show-intersect
  ask LS [
   set pcolor white
   set G-A filter[ ?1 -> ?1 > 0.1 ]  D-A
   set G-B filter[ ?1 -> ?1 > 0.1 ]  D-B
   set G-C filter[ ?1 -> ?1 > 0.1 ]  D-C

  if Intersecting-actors = "A-B"   and not (empty? G-A) and  not (empty? G-B)                [set pcolor
black   ]
  if Intersecting-actors = "A-C"   and not (empty? G-A) and  not (empty? G-C)                [set pcolor
black ]
  if Intersecting-actors = "B-C"   and not (empty? G-B) and  not (empty? G-C)                [set pcolor
black]
```

```
   if Intersecting-actors = "A-B-C" and not (empty? G-A) and  not (empty? G-B) and  not (empty? G-C)
[set pcolor black  ]


 ]
end
;;#############################################################################################
to display-cell-info

 if mouse-down? [ask patch mouse-xcor mouse-ycor [
   clear-output


   output-print word "CLC-1: " label-1
   output-print word "CLC-2: " label-2
   output-print word "CLC-3: " label-3
   output-print "  "
if Region = "MOL" [
   output-print "Supply of ESS (% of potential): "
   output-print "  "
   output-print word "Yield:            " precision item 0 S-ESS 2
   output-print word "Erosion control:   " precision item 1 S-ESS 2
   output-print word "Carbon capture:    " precision item 2 S-ESS 2
   output-print word "Water availability:" precision item 3 S-ESS 2
   output-print word "Biodiversity:      " precision item 4 S-ESS 2
   output-print "  "
   output-print "Demand of ESS (% of potential): "
   output-print "Actor/ ESS (Yld|EC|CS|Wa|Biod):"
   output-print word " A:       "  D-A
   output-print word " B:       "  D-B
   output-print word " C:       "  D-C
   output-print word " New-user: "  D-N
   output-print word " all:     "  D-all
   if ESS = "yield" and Actor = "A" and D-yield-a-id != 0.1 [
```

```
    output-print word " Site-ID:      "  (D-yield-a-id + 1)]
   if ESS = "erosion_control" and Actor = "A" and D-ec-a-id != 0.1 [
    output-print word " Site-ID:      "  (D-ec-a-id + 1)]
   if ESS = "water" and Actor = "A" and D-w-a-id != 0.1 [
    output-print word " Site-ID:      "  (D-w-a-id + 1)]
   if ESS = "carbon_seq" and Actor = "B" and D-cs-b-id != 0.1 [
    output-print word " Site-ID:      "  (D-cs-b-id + 1)]
   if ESS = "biodiversity" and Actor = "B" and D-bio-b-id != 0.1 [
    output-print word " Site-ID:      "  (D-bio-b-id + 1)]
   if ESS = "yield" and Actor = "C" and D-yield-c-id != 0.1 [
    output-print word " Site-ID:      "  (D-yield-c-id + 1)]
   if ESS = "erosion_control" and Actor = "C" and D-ec-c-id != 0.1 [
    output-print word " Site-ID:      "  (D-ec-c-id + 1)]
   if ESS = "water" and Actor = "C" and D-w-c-id != 0.1 [
    output-print word " Site-ID:      "  (D-w-c-id + 1)]
   output-print "  "
   output-print word "S/D gap of actor (ratio): " Actor
   output-print "(Yld|EC|CS|Wa|Biod):"
    set G-ESS   n-values 5 [0]
    foreach ESS-type-list[ ?1 ->
    set G-ESS   replace-item ?1 G-ESS   precision (item ?1 S-ESS / item ?1 tmp  ) 2
    if item ?1 G-ESS >= 10 [set G-ESS   replace-item ?1 G-ESS   (0)]
    ]
   output-print word "" G-ESS
   output-print word "Risk of conflicts (%):"precision Conf 2
   chart-1
   chart-2
  ]]]
end

to chart-1
  if mouse-down? [ask patch mouse-xcor mouse-ycor [
```

```
   set-current-plot "Supply Analysis"
   plot-pen-reset
   set-plot-pen-color red
   set-plot-x-range 0 5
   set-plot-pen-interval 1
   plot (item 0 S-ESS)
   set-plot-pen-color green
   plot (item 1 S-ESS)
   set-plot-pen-color yellow
   plot (item 2 S-ESS)
   set-plot-pen-color blue
   plot (item 3 S-ESS)
   set-plot-pen-color orange
   plot (item 4 S-ESS)
   set-plot-pen-color red
   ]]
end

to chart-2
  if mouse-down? [ask patch mouse-xcor mouse-ycor [
   set-current-plot "Supply-Demand Gap"
   plot-pen-reset
   set-plot-x-range 0 2
   set-plot-pen-interval 1

   if ESS = "yield"         [set-plot-pen-color green plot (item 0 S-ESS) set-plot-pen-color red plot (item 0 tmp) set-plot-pen-color green]
   if ESS = "erosion_control"[set-plot-pen-color green plot (item 1 S-ESS) set-plot-pen-color red plot (item 1 tmp) set-plot-pen-color green]
   if ESS = "carbon_seq"     [set-plot-pen-color green plot (item 2 S-ESS) set-plot-pen-color red plot (item 2 tmp) set-plot-pen-color green]
   if ESS = "water"          [set-plot-pen-color green plot (item 3 S-ESS) set-plot-pen-color red plot (item 3 tmp) set-plot-pen-color green]
```

```
   if ESS = "biodiversity"   [set-plot-pen-color green plot (item 4 S-ESS) set-plot-pen-color red plot
(item 4 tmp) set-plot-pen-color green]


]]

end

;;#############################################################################################

to Show-map ; it shows different background maps that can be used for stating the demand for a
new actor

  ask patches [set pcolor 9 ]

  if Map-type = "CLC"      [ gis:draw CSA 1 ask LS [set pcolor rgb r b g ]]

  if Map-type = "Basic"     [clear-drawing import-drawing "Spatial/Basic.png" ]

  if Map-type = "Satellite" [clear-drawing import-drawing "Spatial/Satellite.png" ]

end


to add-demand


  if mouse-down? [clear-drawing ask patch mouse-xcor mouse-ycor [

     foreach ESS-type-list[

   if ESS = "yield"          [ set D-N replace-item 0 D-N Demand set pcolor scale-color red    item 0 D-N
120 0 ]

   if ESS = "erosion_control" [ set D-N replace-item 1 D-N Demand set pcolor scale-color lime   item 1
D-N 120 0 ]

   if ESS = "carbon_seq"      [ set D-N replace-item 2 D-N Demand set pcolor scale-color yellow item 2
D-N 120 0 ]

   if ESS = "water"          [ set D-N replace-item 3 D-N Demand set pcolor scale-color blue   item 3 D-N
120 0 ]

   if ESS = "biodiversity"    [ set D-N replace-item 4 D-N Demand set pcolor scale-color orange item 4
D-N 120 0 ]

  ]]]

  if not (mouse-down?) [

  if Map-type = "Basic" [import-drawing "Spatial/Basic.png"]

  if Map-type = "Satellite" [import-drawing "Spatial/Satellite.png"]


  ]
```

```
end
to Clear-demand

  if mouse-down? [clear-drawing ask patch mouse-xcor mouse-ycor [
     foreach ESS-type-list[
    if ESS = "yield"          [ set D-N replace-item 0 D-N 0.1 set pcolor scale-color red    item 0 D-N 120 0 ]
    if ESS = "erosion_control" [ set D-N replace-item 1 D-N 0.1 set pcolor scale-color lime   item 1 D-N
120 0 ]
    if ESS = "carbon_seq"      [ set D-N replace-item 2 D-N 0.1 set pcolor scale-color yellow item 2 D-N
120 0 ]
    if ESS = "water"          [ set D-N replace-item 3 D-N 0.1 set pcolor scale-color blue   item 3 D-N 120 0
]
    if ESS = "biodiversity"    [ set D-N replace-item 4 D-N 0.1 set pcolor scale-color orange item 4 D-N
120 0 ]
  ]]]
  if not (mouse-down?) [
  if Map-type = "Basic"     [import-drawing "Spatial/Basic.png"]
  if Map-type = "Satellite" [import-drawing "Spatial/Satellite.png"]
]
end
;;#############################################################################################
to test-gis-output
  if Map-display = "Demand" [
  set output_file gis:patch-dataset tmp2
  gis:store-dataset output_file "Output/maps/Demand"]

  if Map-display = "Supply" [
  set output_file gis:patch-dataset tmp2
  gis:store-dataset output_file "Output/maps/Supply"]

  if Map-display = "Gap" [
  set output_file gis:patch-dataset tmp2
```

```
 gis:store-dataset output_file "Output/maps/Gap"]

 if Map-display = "Conflict" [
 set output_file gis:patch-dataset tmp2
 gis:store-dataset output_file "Output/maps/Conflict"]
end
;;#############################################################################
to setup

 clear-all-plots
 clear-turtles
 set Py precision (1 - Px - Pz) 1
 set Px precision (1 - Py - Pz) 1
 set Pz precision (1 - Px - Py) 1

set actors    n-values 3 [?1 -> ?1]
ask LS [
set L 100       ; carrying capacity
set g2 0.002      ; restoration rate
set n  0.0002     ; rate of natural damage
set f  0.9      ; fraction of capital

 ;    set parameters [; values of parameters for each actor
   ;financial capital
 set K-1   [100000  36000          300000] ;Income
 set K-2   [60000            26000  100000] ;Expenditure
 set K-3   [40000            10000  100000] ;Savings
   ;social capital
 set K-4   [100     100      100   ] ;Social-insurance
 set K-5   [100     100      100   ] ;Health-insurance
 set K-6   [100     0          100   ] ;Agricultural-insurance
   ;physical capital
```

```
 set K-7   [5000    2000    100   ] ;Equipment and tools (Inv.)
 set K-8   [100     10        10    ] ;Internet Network
 set K-9   [50000           5000      2000  ] ;Vehicles/transportation means (Inv.)
    ;human capital
 set K-10  [5         3         1     ] ;Health
 set K-11  [5         3         1     ] ;Nutrition
 set K-12  [3         2         1     ] ;Skills
    ;cultural capital
 set K-13  [0         5         4     ] ;Celebrations
 set K-14  [80       80        30    ] ;Beliefs
 set K-15  [10       80        90    ] ;Traditions

    ;natural capital
set d_x     [0.1     0.1    0.1] ; demand for yield ************************************
set d_y     [0.1     0.1    0.1] ; demand for  EC**************************************
set d_z     [0.1     0.1    0.1] ; demand for  CS **************************************
set d_l     [0.1     0.1    0.1] ; demand for  water **********************************
set d_m     [0.1     0.1    0.1] ; demand for  bio *************************************

  foreach actors [

    set d_x replace-item 0 d_x (d-yield-a)
    set d_x replace-item 2 d_x (d-yield-c)
    set d_y replace-item 0 d_y (d-ec-a)
    set d_y replace-item 2 d_y (d-ec-c)
    set d_z replace-item 1 d_z (d-cs-b)
    set d_l replace-item 0 d_l (d-w-a)
    set d_l replace-item 2 d_l (d-w-c)
    set d_m replace-item 1 d_m (d-bio-b)

   ]
```

```
set wt-1  [0.058823529    0.010309278    0.012987013]
set wt-2  [0.102941176    0.103092784    0.064935065]
set wt-3  [0.058823529    0.051546392    0.12987013]

set wt-4  [0.058823529    0.103092784    0.103896104]
set wt-5  [0.088235294    0.051546392    0.051948052]
set wt-6  [0.029411765    0.051546392    0.12987013]

set wt-7  [0.044117647    0.051546392    0.025974026]
set wt-8  [0.058823529    0.06185567     0.038961039]
set wt-9  [0.147058824    0.092783505    0.038961039]

set wt-10 [0.073529412    0.041237113    0.090909091]
set wt-11 [0.102941176    0.103092784    0.012987013]
set wt-12 [0.029411765    0.082474227    0.090909091]

set wt-13 [0.014705882    0.06185567     0.077922078]
set wt-14 [0.088235294    0.072164948    0.012987013]
set wt-15 [0.044117647    0.06185567     0.116883117]


;weights of natural capital
set r_x     [0.33    0.09    0.35] ; preference of yield
set r_y     [0.29    0.09    0.23] ; preference of EC
set r_z     [0.04    0.41    0.04] ; preference of CS
set r_l     [0.25    0.05    0.31] ; preference of water
set r_m     [0.09    0.36    0.07] ; preference of bio

 ; Management options data regarding efforts, utilities and efficiency
 set MO-data-effort  [  ; costs C differs with managemen options
```

; Income Expenditure Savings Social-insurance Health-insurance Agricultural-insurance Equipment and tools (Inv.) Internet Networ Vehicles/transportation means (Inv.) Health Nutrition Skills Celebrations Beliefs Traditions

[ 0 10000 0 0 0 0 0 0 0 0.1 0 0 1 10 20 ]

[ 30000 20000 0 0 0 0 0 0 0 0 0 0 0 0 0 ]

[ 50000 40000 0 0 0 0 0 0 0 0 0 0 5 0 ]

]

; Still missing the interconnection between the variables

set MO-data-utility [ ; initial price per unit management option (in the fishery model, it is per unit fish harvested) a, impact of management options. It could be interpreted per ESS supply by dividing by the efficiency.

; a can be related to ESS directly so it shows the impact of each ESS on the below listed items istead of having it an impact of the mangement options so that MOs would only indicate the efficiency of increasing the supply.

; Income Expenditure Savings Social-insurance Health-insurance Agricultural-insurance Equipment and tools (Inv.) Internet Network Vehicles/transportation means (Inv.) Health Nutrition Skills Celebrations Beliefs Traditions

[ 200000 0 0 0.1 0 -10 30000 200 50000 0 1 1 0 10 0 ]

[ 0 0 0 0 0 -5 10000 100 0 0.25 0.25 0 0 10 0 ]

[ 10000 0 0 0.05 -5 -5 10000 0 0 0.35 0.35 0 0.25 5 5 ]

]

set MO-data-utility-elasticity [ ; price elasticity b: the application of a management option will increase the supply, thus the value gained will decrease with the increase of supply at a rate of b. Here it is taken with the increase of the application of

; the management option in terms of land area

; Income Expenditure Savings Social-insurance Health-insurance Agricultural-insurance Equipment and tools (Inv.) Internet Network Vehicles/transportation means (Inv.) Health Nutrition Skills Celebrations Beliefs Traditions

```
  [ 0.1 0 0 0.1 0 0.1 0.1
    0.1 0.1 0 0.1 0.1 0
0.1 0 ]

  [ 0 0 0 0 0 0.1 0.1
    0.1 0 0.1 0.1 0 0
    0.1 0 ]

  [ 0.1 0 0 0.1 0.1 0.1 0.1
    0 0 0.1 0.1 0 0.1
0.1 0.1 ]

 ]

 set efficiency [ ; q for each management options

  ; efficiency_yield  efficiency_ec  efficiency_cs  efficiency_water  efficiency_bio

  [ 0.04 0.08 0.04 0.06 0.12 ]

  [ -0.02 0.05 0.09 0.02 0.09 ]

  [ 0.02 0.04 0.02 0.01 0.04 ]

 ]


  set norm_efficiency [

   ; efficiency_yield  efficiency_ec  efficiency_cs  efficiency_water  efficiency_bio

  [ 0.118 0.235 0.118 0.176 0.353 ]

  [ -0.09 0.22 0.39 0.09 0.39 ]

  [ 0.154 0.308 0.154 0.077 0.307 ]

  ]

set maximum-parameters

  [ 1000000  500000  10000000  100  100  100  100000
300  200000  5  5  5  5  100  100  ]

let min-parameters 0


  set parameter-list n-values 15 [?1 -> ?1]

  set norm-C  n-values 15 [?1 -> ?1]

  set norm-a  n-values 15 [?1 -> ?1]

  set a-share n-values 15 [?1 -> ?1]

  set C-share n-values 15 [?1 -> ?1]
```

```
   if Land-management = "Crop rotation"  [set q item 0 efficiency  set norm_q item 0 norm_efficiency set efforts item 0 MO-data-effort set values item 0 MO-data-utility set elasticity item 0 MO-data-utility-elasticity]

   if Land-management = "Hedge"         [set q item 1 efficiency  set norm_q item 1 norm_efficiency set efforts item 1 MO-data-effort set values item 1 MO-data-utility set elasticity item 1 MO-data-utility-elasticity]; **********Achtung

   if Land-management = "Agroforestry"   [set q item 2 efficiency  set norm_q item 2 norm_efficiency set efforts item 2 MO-data-effort set values item 2 MO-data-utility set elasticity item 2 MO-data-utility-elasticity];**********Achtung

   if Land-management = "mix"           [set q ((Px * item 0 efficiency) + (Py * item 1 efficiency) + (Pz * item 2 efficiency)) set norm_q ((Px * item 0 norm_efficiency) + (Py * item 1 norm_efficiency) + (Pz * item 2 norm_efficiency))

                                        set efforts ((Px * item 0 MO-data-effort) + (Py * item 1 MO-data-effort) + (Pz * item 2 MO-data-effort)) set values ((Px * item 0 MO-data-utility) + (Py * item 1 MO-data-utility) + (Pz * item 2 MO-data-utility))

                                        set elasticity ((Px * item 0 MO-data-utility-elasticity) + (Py * item 1 MO-data-utility-elasticity) + (Pz * item 2 MO-data-utility-elasticity))]


   foreach parameter-list [ ?1 ->  ; to normalize efforts, initial value so that to have a value between 0.01 and 1

   set norm-C replace-item ?1 norm-C ((item ?1 efforts - min-parameters) / (item ?1 maximum-parameters - min-parameters) * 100) ; normalized efforts

   set norm-a replace-item ?1 norm-a ((item ?1 values  - min-parameters) / (item ?1 maximum-parameters - min-parameters) * 100) ; normalized values


   ]

   set norm-a-tot sum norm-a

   set norm-C-tot sum norm-C


   foreach parameter-list [ ?1 ->  ; to normalize efforts, initial value so that to have a value between 0.01 and 1


    set a-share replace-item ?1 a-share (item ?1 norm-a / norm-a-tot) ; the sum of a-share should be 1

    set C-share replace-item ?1 C-share (item ?1 norm-C / norm-C-tot)

   ]
```

```
   ;Actors     A      B     C
set q_x n-values 3 [item 0 q]  ; increase efficiency of yield
set q_y n-values 3 [item 1 q]  ; increase efficiency of EC
set q_z n-values 3 [item 2 q]  ; increase efficiency of CS
set q_l n-values 3 [item 3 q]  ; increase efficiency of water
set q_m n-values 3 [item 4 q]  ; increase efficiency of bio

   set a-1    n-values 5 [?1 -> ?1]
   set a-2    n-values 5 [?1 -> ?1]
   set a-3    n-values 5 [?1 -> ?1]
   set a-4    n-values 5 [?1 -> ?1]
   set a-5    n-values 5 [?1 -> ?1]
   set a-6    n-values 5 [?1 -> ?1]
   set a-7    n-values 5 [?1 -> ?1]
   set a-8    n-values 5 [?1 -> ?1]
   set a-9    n-values 5 [?1 -> ?1]
   set a-10   n-values 5 [?1 -> ?1]
   set a-11   n-values 5 [?1 -> ?1]
   set a-12   n-values 5 [?1 -> ?1]
   set a-13   n-values 5 [?1 -> ?1]
   set a-14   n-values 5 [?1 -> ?1]
   set a-15   n-values 5 [?1 -> ?1]
   set a      n-values 5 [?1 -> ?1]

   set b2-1   n-values 5 [?1 -> ?1]
   set b2-2   n-values 5 [?1 -> ?1]
   set b2-3   n-values 5 [?1 -> ?1]
   set b2-4   n-values 5 [?1 -> ?1]
   set b2-5   n-values 5 [?1 -> ?1]
   set b2-6   n-values 5 [?1 -> ?1]
   set b2-7   n-values 5 [?1 -> ?1]
```

set b2-8 n-values 5 [?1 -> ?1]

set b2-9 n-values 5 [?1 -> ?1]

set b2-10 n-values 5 [?1 -> ?1]

set b2-11 n-values 5 [?1 -> ?1]

set b2-12 n-values 5 [?1 -> ?1]

set b2-13 n-values 5 [?1 -> ?1]

set b2-14 n-values 5 [?1 -> ?1]

set b2-15 n-values 5 [?1 -> ?1]

set b2 n-values 5 [?1 -> ?1]

set K n-values 3 [?1 -> ?1]

set K-0 n-values 3 [?1 -> ?1]

set C n-values 3 [?1 -> ?1]

set K-fin n-values 3 [?1 -> ?1]

set K-soc n-values 3 [?1 -> ?1]

set K-phy n-values 3 [?1 -> ?1]

set K-hum n-values 3 [?1 -> ?1]

set K-cul n-values 3 [?1 -> ?1]

set K0-1 n-values 3 [?1 -> ?1]

set K0-2 n-values 3 [?1 -> ?1]

set K0-3 n-values 3 [?1 -> ?1]

set K0-4 n-values 3 [?1 -> ?1]

set K0-5 n-values 3 [?1 -> ?1]

set K0-6 n-values 3 [?1 -> ?1]

set K0-7 n-values 3 [?1 -> ?1]

set K0-8 n-values 3 [?1 -> ?1]

set K0-9 n-values 3 [?1 -> ?1]

set K0-10 n-values 3 [?1 -> ?1]

set K0-11 n-values 3 [?1 -> ?1]

set K0-12 n-values 3 [?1 -> ?1]

set K0-13 n-values 3 [?1 -> ?1]

set K0-14 n-values 3 [?1 -> ?1]

set K0-15 n-values 3 [?1 -> ?1]

set P-1 n-values 5 [?1 -> ?1]

set P-2 n-values 5 [?1 -> ?1]

set P-3 n-values 5 [?1 -> ?1]

set P-4 n-values 5 [?1 -> ?1]

set P-5 n-values 5 [?1 -> ?1]

set P-6 n-values 5 [?1 -> ?1]

set P-7 n-values 5 [?1 -> ?1]

set P-8 n-values 5 [?1 -> ?1]

set P-9 n-values 5 [?1 -> ?1]

set P-10 n-values 5 [?1 -> ?1]

set P-11 n-values 5 [?1 -> ?1]

set P-12 n-values 5 [?1 -> ?1]

set P-13 n-values 5 [?1 -> ?1]

set P-14 n-values 5 [?1 -> ?1]

set P-15 n-values 5 [?1 -> ?1]

set V-1 n-values 3 [?1 -> ?1]

set V-2 n-values 3 [?1 -> ?1]

set V-3 n-values 3 [?1 -> ?1]

set V-4 n-values 3 [?1 -> ?1]

set V-5 n-values 3 [?1 -> ?1]

set V-6 n-values 3 [?1 -> ?1]

set V-7 n-values 3 [?1 -> ?1]

set V-8 n-values 3 [?1 -> ?1]

set V-9 n-values 3 [?1 -> ?1]

set V-10 n-values 3 [?1 -> ?1]

set V-11 n-values 3 [?1 -> ?1]

set V-12 n-values 3 [?1 -> ?1]

set V-13 n-values 3 [?1 -> ?1]

set V-14 n-values 3 [?1 -> ?1]

set V-15 n-values 3 [?1 -> ?1]

```
set C-fin  n-values 3 [?1 -> ?1]
set C-soc  n-values 3 [?1 -> ?1]
set C-phy  n-values 3 [?1 -> ?1]
set C-hum  n-values 3 [?1 -> ?1]
set C-cul  n-values 3 [?1 -> ?1]

set C-x    n-values 3 [?1 -> ?1]
set C-y    n-values 3 [?1 -> ?1]
set C-z    n-values 3 [?1 -> ?1]
set C-l    n-values 3 [?1 -> ?1]
set C-m    n-values 3 [?1 -> ?1]

set V-fin n-values 3 [?1 -> ?1]
set V-soc n-values 3 [?1 -> ?1]
set V-phy n-values 3 [?1 -> ?1]
set V-hum n-values 3 [?1 -> ?1]
set V-cul n-values 3 [?1 -> ?1]

set P-fin n-values 5 [?1 -> ?1]
set P-soc n-values 5 [?1 -> ?1]
set P-phy n-values 5 [?1 -> ?1]
set P-hum n-values 5 [?1 -> ?1]
set P-cul n-values 5 [?1 -> ?1]

set C-1  n-values 3 [item 0  norm-C]
set C-2  n-values 3 [item 1  norm-C]
set C-3  n-values 3 [item 2  norm-C]
set C-4  n-values 3 [item 3  norm-C]
set C-5  n-values 3 [item 4  norm-C]
set C-6  n-values 3 [item 5  norm-C]
set C-7  n-values 3 [item 6  norm-C]
```

```
   set C-8  n-values 3 [item 7  norm-C]

   set C-9  n-values 3 [item 8  norm-C]

   set C-10 n-values 3 [item 9  norm-C]

   set C-11 n-values 3 [item 10 norm-C]

   set C-12 n-values 3 [item 11 norm-C]

   set C-13 n-values 3 [item 12 norm-C]

   set C-14 n-values 3 [item 13 norm-C]

   set C-15 n-values 3 [item 14 norm-C]


   foreach ESS-type-list [ ?1 ->


   set a-1  replace-item ?1 a-1   (item 0  norm-a * item ?1 norm_q)  set b2-1  replace-item ?1 b2-1
(item 0  elasticity * item ?1 norm_q)

   set a-2  replace-item ?1 a-2   (item 1  norm-a * item ?1 norm_q)  set b2-2  replace-item ?1 b2-2
(item 1  elasticity * item ?1 norm_q)

   set a-3  replace-item ?1 a-3   (item 2  norm-a * item ?1 norm_q)  set b2-3  replace-item ?1 b2-3
(item 2  elasticity * item ?1 norm_q)

   set a-4  replace-item ?1 a-4   (item 3  norm-a * item ?1 norm_q)  set b2-4  replace-item ?1 b2-4
(item 3  elasticity * item ?1 norm_q)

   set a-5  replace-item ?1 a-5   (item 4  norm-a * item ?1 norm_q)  set b2-5  replace-item ?1 b2-5
(item 4  elasticity * item ?1 norm_q)

   set a-6  replace-item ?1 a-6   (item 5  norm-a * item ?1 norm_q)  set b2-6  replace-item ?1 b2-6
(item 5  elasticity * item ?1 norm_q)

   set a-7  replace-item ?1 a-7   (item 6  norm-a * item ?1 norm_q)  set b2-7  replace-item ?1 b2-7
(item 6  elasticity * item ?1 norm_q)

   set a-8  replace-item ?1 a-8   (item 7  norm-a * item ?1 norm_q)  set b2-8  replace-item ?1 b2-8
(item 7  elasticity * item ?1 norm_q)

   set a-9  replace-item ?1 a-9   (item 8  norm-a * item ?1 norm_q)  set b2-9  replace-item ?1 b2-9
(item 8  elasticity * item ?1 norm_q)

   set a-10 replace-item ?1 a-10  (item 9  norm-a * item ?1 norm_q)  set b2-10 replace-item ?1 b2-10
(item 9  elasticity * item ?1 norm_q)

   set a-11 replace-item ?1 a-11  (item 10 norm-a * item ?1 norm_q)  set b2-11 replace-item ?1 b2-11
(item 10 elasticity * item ?1 norm_q)

   set a-12 replace-item ?1 a-12  (item 11 norm-a * item ?1 norm_q)  set b2-12 replace-item ?1 b2-12
(item 11 elasticity * item ?1 norm_q)
```

set a-13 replace-item ?1 a-13  (item 12 norm-a * item ?1 norm_q)  set b2-13 replace-item ?1 b2-13 (item 12 elasticity * item ?1 norm_q)

set a-14 replace-item ?1 a-14  (item 13 norm-a * item ?1 norm_q)  set b2-14 replace-item ?1 b2-14 (item 13 elasticity * item ?1 norm_q)

set a-15 replace-item ?1 a-15  (item 14 norm-a * item ?1 norm_q)  set b2-15 replace-item ?1 b2-15 (item 14 elasticity * item ?1 norm_q)

set a replace-item ?1 a  (item ?1 a-1 + item ?1 a-2 + item ?1 a-3 + item ?1 a-4 + item ?1 a-5 + item ?1 a-6 + item ?1 a-7 + item ?1 a-8 + item ?1 a-9 + item ?1 a-10 + item ?1 a-11 + item ?1 a-12 + item ?1 a-13 + item ?1 a-14 + item ?1 a-15)

set b2 replace-item ?1 b2 ((item ?1 b2-1 + item ?1 b2-2 + item ?1 b2-3 + item ?1 b2-4 + item ?1 b2-5 + item ?1 b2-6 + item ?1 b2-7 + item ?1 b2-8 + item ?1 b2-9 + item ?1 b2-10 + item ?1 b2-11 + item ?1 b2-12 + item ?1 b2-13 + item ?1 b2-14 + item ?1 b2-15) / 15)

]

foreach actors [ ?1 ->

set K-1  replace-item ?1 K-1  ((  item ?1 K-1  - min-parameters) / (item 0  maximum-parameters - min-parameters)  * 100)

set K-2  replace-item ?1 K-2  ((item 1 maximum-parameters - item ?1 K-2) / (item 1  maximum-parameters - min-parameters)  * 100) ; this should be reduced, so I normalized it in a reciprocal form

set K-3  replace-item ?1 K-3  ((Budget + item ?1 K-3  - min-parameters) / (item 2  maximum-parameters - min-parameters)  * 100)

set K-4  replace-item ?1 K-4  ((  item ?1 K-4  - min-parameters) / (item 3  maximum-parameters - min-parameters)  * 100)

set K-5  replace-item ?1 K-5  ((  item ?1 K-5  - min-parameters) / (item 4  maximum-parameters - min-parameters)  * 100)

set K-6  replace-item ?1 K-6  ((  item ?1 K-6  - min-parameters) / (item 5  maximum-parameters - min-parameters)  * 100)

set K-7  replace-item ?1 K-7  ((  item ?1 K-7  - min-parameters) / (item 6  maximum-parameters - min-parameters)  * 100)

set K-8  replace-item ?1 K-8  ((  item ?1 K-8  - min-parameters) / (item 7  maximum-parameters - min-parameters)  * 100)

set K-9  replace-item ?1 K-9  ((  item ?1 K-9  - min-parameters) / (item 8  maximum-parameters - min-parameters)  * 100)

set K-10 replace-item ?1 K-10 ((  item ?1 K-10 - min-parameters) / (item 9  maximum-parameters - min-parameters)  * 100)

set K-11 replace-item ?1 K-11 ((  item ?1 K-11 - min-parameters) / (item 10 maximum-parameters - min-parameters)  * 100)

set K-12 replace-item ?1 K-12 ((  item ?1 K-12 - min-parameters) / (item 11 maximum-parameters - min-parameters)  * 100)

set K-13 replace-item ?1 K-13 ((  item ?1 K-13 - min-parameters) / (item 12 maximum-parameters - min-parameters)  * 100)

set K-14 replace-item ?1 K-14 ((  item ?1 K-14 - min-parameters) / (item 13 maximum-parameters - min-parameters)  * 100)

set K-15 replace-item ?1 K-15 ((  item ?1 K-15 - min-parameters) / (item 14 maximum-parameters - min-parameters)  * 100)

if weighted-decision? [

set K-1  replace-item ?1 K-1  (item ?1 K-1 * item ?1 wt-1) set C-1 replace-item ?1 C-1  (item ?1 C-1 * item ?1 wt-1)

set K-2  replace-item ?1 K-2  (item ?1 K-2 * item ?1 wt-2) set C-2 replace-item ?1 C-2  (item ?1 C-2 * item ?1 wt-2)

set K-3  replace-item ?1 K-3  (item ?1 K-3 * item ?1 wt-3) set C-3 replace-item ?1 C-3  (item ?1 C-3 * item ?1 wt-3)

set K-4  replace-item ?1 K-4  (item ?1 K-4 * item ?1 wt-4) set C-4 replace-item ?1 C-4  (item ?1 C-4 * item ?1 wt-4)

set K-5  replace-item ?1 K-5  (item ?1 K-5 * item ?1 wt-5) set C-5 replace-item ?1 C-5  (item ?1 C-5 * item ?1 wt-5)

set K-6  replace-item ?1 K-6  (item ?1 K-6 * item ?1 wt-6) set C-6 replace-item ?1 C-6  (item ?1 C-6 * item ?1 wt-6)

set K-7  replace-item ?1 K-7  (item ?1 K-7 * item ?1 wt-7) set C-7 replace-item ?1 C-7  (item ?1 C-7 * item ?1 wt-7)

set K-8  replace-item ?1 K-8  (item ?1 K-8 * item ?1 wt-8) set C-8 replace-item ?1 C-8  (item ?1 C-8 * item ?1 wt-8)

set K-9  replace-item ?1 K-9  (item ?1 K-9 * item ?1 wt-9) set C-9 replace-item ?1 C-9  (item ?1 C-9 * item ?1 wt-9)

set K-10 replace-item ?1 K-10 (item ?1 K-10 * item ?1 wt-10) set C-10 replace-item ?1 C-10  (item ?1 C-10 * item ?1 wt-10)

set K-11 replace-item ?1 K-11 (item ?1 K-11 * item ?1 wt-11) set C-11 replace-item ?1 C-11  (item ?1 C-11 * item ?1 wt-11)

set K-12 replace-item ?1 K-12 (item ?1 K-12 * item ?1 wt-12) set C-12 replace-item ?1 C-12  (item ?1 C-12 * item ?1 wt-12)

set K-13 replace-item ?1 K-13 (item ?1 K-13 * item ?1 wt-13) set C-13 replace-item ?1 C-13  (item ?1 C-13 * item ?1 wt-13)

set K-14 replace-item ?1 K-14 (item ?1 K-14 * item ?1 wt-14) set C-14 replace-item ?1 C-14 (item ?1 C-14 * item ?1 wt-14)

set K-15 replace-item ?1 K-15 (item ?1 K-15 * item ?1 wt-15) set C-15 replace-item ?1 C-15 (item ?1 C-15 * item ?1 wt-15)

]

set K replace-item ?1 K ( item ?1 K-1 + item ?1 K-2 + item ?1 K-3 + item ?1 K-4 + item ?1 K-5 + item ?1 K-6 + item ?1 K-7 + item ?1 K-8 + item ?1 K-9 + item ?1 K-10 + item ?1 K-11 + item ?1 K-12 + item ?1 K-13 + item ?1 K-14 + item ?1 K-15)

set C replace-item ?1 C ( item ?1 C-1 + item ?1 C-2 + item ?1 C-3 + item ?1 C-4 + item ?1 C-5 + item ?1 C-6 + item ?1 C-7 + item ?1 C-8 + item ?1 C-9 + item ?1 C-10 + item ?1 C-11 + item ?1 C-12 + item ?1 C-13 + item ?1 C-14 + item ?1 C-15)

Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C)

Set C-y replace-item ?1 C-y (item ?1 r_y * item ?1 C)

Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C)

Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C)

Set C-m replace-item ?1 C-m (item ?1 r_m * item ?1 C)

set K-0 replace-item ?1 K-0 ( item ?1 K )

set C-fin replace-item ?1 C-fin (item ?1 C-1 + item ?1 C-2 + item ?1 C-3 ) set K-fin replace-item ?1 K-fin (item ?1 K-1 + item ?1 K-2 + item ?1 K-3 )

set C-soc replace-item ?1 C-soc (item ?1 C-4 + item ?1 C-5 + item ?1 C-6 ) set K-soc replace-item ?1 K-soc (item ?1 K-4 + item ?1 K-5 + item ?1 K-6 )

set C-phy replace-item ?1 C-phy (item ?1 C-7 + item ?1 C-8 + item ?1 C-9 ) set K-phy replace-item ?1 K-phy (item ?1 K-7 + item ?1 K-8 + item ?1 K-9 )

set C-hum replace-item ?1 C-hum (item ?1 C-10 + item ?1 C-11 + item ?1 C-12) set K-hum replace-item ?1 K-hum (item ?1 K-10 + item ?1 K-11 + item ?1 K-12)

set C-cul replace-item ?1 C-cul (item ?1 C-13 + item ?1 C-14 + item ?1 C-15) set K-cul replace-item ?1 K-cul (item ?1 K-13 + item ?1 K-14 + item ?1 K-15)

```
    set K0-1  replace-item ?1 K0-1  (item ?1 K-1  / item ?1 K-0)
    set K0-2  replace-item ?1 K0-2  (item ?1 K-2  / item ?1 K-0)
    set K0-3  replace-item ?1 K0-3  (item ?1 K-3  / item ?1 K-0)
    set K0-4  replace-item ?1 K0-4  (item ?1 K-4  / item ?1 K-0)
    set K0-5  replace-item ?1 K0-5  (item ?1 K-5  / item ?1 K-0)
    set K0-6  replace-item ?1 K0-6  (item ?1 K-6  / item ?1 K-0)
    set K0-7  replace-item ?1 K0-7  (item ?1 K-7  / item ?1 K-0)
    set K0-8  replace-item ?1 K0-8  (item ?1 K-8  / item ?1 K-0)
    set K0-9  replace-item ?1 K0-9  (item ?1 K-9  / item ?1 K-0)
    set K0-10 replace-item ?1 K0-10 (item ?1 K-10 / item ?1 K-0)
    set K0-11 replace-item ?1 K0-11 (item ?1 K-11 / item ?1 K-0)
    set K0-12 replace-item ?1 K0-12 (item ?1 K-12 / item ?1 K-0)
    set K0-13 replace-item ?1 K0-13 (item ?1 K-13 / item ?1 K-0)
    set K0-14 replace-item ?1 K0-14 (item ?1 K-14 / item ?1 K-0)
    set K0-15 replace-item ?1 K0-15 (item ?1 K-15 / item ?1 K-0)

set K-plot K
set C-plot C

  if capital  = "Financial capital"  [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-fin) Set C-y
replace-item ?1 C-y (item ?1 r_y * item ?1 C-fin) Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-
fin)
                                    Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-fin) Set C-m replace-item ?1
C-m (item ?1 r_m * item ?1 C-fin) set K-plot K-fin set C-plot C-fin]
  if capital  = "Social capital"    [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-soc) Set C-y
replace-item ?1 C-y (item ?1 r_y * item ?1 C-soc) Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-
soc)
                                    Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-soc) Set C-m replace-item
?1 C-m (item ?1 r_m * item ?1 C-soc) set K-plot K-soc set C-plot C-soc]
  if capital  = "Physical capital"  [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-phy) Set C-y
replace-item ?1 C-y (item ?1 r_y * item ?1 C-phy) Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-
phy)
                                    Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-phy) Set C-m replace-item
?1 C-m (item ?1 r_m * item ?1 C-phy) set K-plot K-phy set C-plot C-phy]
```

```
  if capital   = "Human capital"      [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-hum) Set C-y
replace-item ?1 C-y (item ?1 r_y * item ?1 C-hum) Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1
C-hum)

                                Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-hum) Set C-m replace-item
?1 C-m (item ?1 r_m * item ?1 C-hum) set K-plot K-hum set C-plot C-hum]

  if capital   = "Cultural capital"   [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-cul) Set C-y
replace-item ?1 C-y (item ?1 r_y * item ?1 C-cul) Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-
cul)

                                Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-cul) Set C-m replace-item ?1
C-m (item ?1 r_m * item ?1 C-cul) set K-plot K-cul set C-plot C-cul]


  if Indicator = "Income"        [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-1)  Set C-y replace-
item ?1 C-y (item ?1 r_y * item ?1 C-1)   Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-1)

                                Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-1)  Set C-m replace-item ?1 C-
m (item ?1 r_m * item ?1 C-1)   set K-plot K-1 set C-plot C-1 ]

  if Indicator = "Expenditure"   [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-2)  Set C-y
replace-item ?1 C-y (item ?1 r_y * item ?1 C-2)   Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-
2)

                                Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-2)  Set C-m replace-item ?1 C-
m (item ?1 r_m * item ?1 C-2)   set K-plot K-2 set C-plot C-2 ]

  if Indicator = "Savings"       [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-3)  Set C-y replace-
item ?1 C-y (item ?1 r_y * item ?1 C-3)   Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-3)

                                Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-3)  Set C-m replace-item ?1 C-
m (item ?1 r_m * item ?1 C-3)   set K-plot K-3 set C-plot C-3 ]

  if Indicator = "Soc_insurance" [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-4)  Set C-y
replace-item ?1 C-y (item ?1 r_y * item ?1 C-4)   Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-
4)

                                Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-4)  Set C-m replace-item ?1 C-
m (item ?1 r_m * item ?1 C-4)   set K-plot K-4 set C-plot C-4 ]

  if Indicator = "H_insurance"   [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-5)  Set C-y
replace-item ?1 C-y (item ?1 r_y * item ?1 C-5)   Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-
5)

                                Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-5)  Set C-m replace-item ?1 C-
m (item ?1 r_m * item ?1 C-5)   set K-plot K-5 set C-plot C-5 ]

  if Indicator = "Agr_insurance" [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-6)  Set C-y
replace-item ?1 C-y (item ?1 r_y * item ?1 C-6)   Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-
6)

                                Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-6)  Set C-m replace-item ?1 C-
m (item ?1 r_m * item ?1 C-6)   set K-plot K-6 set C-plot C-6 ]
```

```
  if Indicator = "Equipment"    [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-7)  Set C-y
replace-item ?1 C-y (item ?1 r_y * item ?1 C-7)   Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-
7)

                    Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-7)  Set C-m replace-item ?1 C-
m (item ?1 r_m * item ?1 C-7)   set K-plot K-7 set C-plot C-7 ]

  if Indicator = "Internet"     [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-8)  Set C-y replace-
item ?1 C-y (item ?1 r_y * item ?1 C-8)   Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-8)

                    Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-8)  Set C-m replace-item ?1 C-
m (item ?1 r_m * item ?1 C-8)   set K-plot K-8 set C-plot C-8 ]

  if Indicator = "Vehicles"     [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-9)  Set C-y replace-
item ?1 C-y (item ?1 r_y * item ?1 C-9)   Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-9)

                    Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-9)  Set C-m replace-item ?1 C-
m (item ?1 r_m * item ?1 C-9)   set K-plot K-9 set C-plot C-9 ]

  if Indicator = "Health"       [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-10) Set C-y replace-
item ?1 C-y (item ?1 r_y * item ?1 C-10)  Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-10)

                    Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-10) Set C-m replace-item ?1 C-
m (item ?1 r_m * item ?1 C-10)  set K-plot K-10 set C-plot C-10 ]

  if Indicator = "Nutrition"    [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-11) Set C-y
replace-item ?1 C-y (item ?1 r_y * item ?1 C-11)  Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-
11)

                    Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-11) Set C-m replace-item ?1 C-
m (item ?1 r_m * item ?1 C-11)  set K-plot K-11 set C-plot C-11 ]

  if Indicator = "Skills"       [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-12) Set C-y replace-
item ?1 C-y (item ?1 r_y * item ?1 C-12)  Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-12)

                    Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-12) Set C-m replace-item ?1 C-
m (item ?1 r_m * item ?1 C-12)  set K-plot K-12 set C-plot C-12 ]

  if Indicator = "Celebration"  [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-13) Set C-y
replace-item ?1 C-y (item ?1 r_y * item ?1 C-13)  Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-
13)

                    Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-13) Set C-m replace-item ?1 C-
m (item ?1 r_m * item ?1 C-13)  set K-plot K-13 set C-plot C-13 ]

  if Indicator = "Beliefs"      [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-14) Set C-y replace-
item ?1 C-y (item ?1 r_y * item ?1 C-14)  Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-14)

                    Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-14) Set C-m replace-item ?1 C-
m (item ?1 r_m * item ?1 C-14)  set K-plot K-14 set C-plot C-14 ]

  if Indicator = "Traditions"   [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-15) Set C-y
replace-item ?1 C-y (item ?1 r_y * item ?1 C-15)  Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-
15)
```

```
                    Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-15) Set C-m replace-item ?1 C-
m (item ?1 r_m * item ?1 C-15)  set K-plot K-15 set C-plot C-15 ]
  ]

set x_per_x  [50     50     50] ; perceived supply of yield
set x_per_y  [50     50     50] ; perceived supply of EC
set x_per_z  [50     50     50] ; perceived supply of CS
set x_per_l  [50     50     50] ; perceived supply of water
set x_per_m  [50     50     50] ; perceived supply of bio
  ; yield  EC  CS  water  bio

set growth     n-values 5 [?1 -> ?1]
set tot_harv   n-values 5 [?1 -> ?1] ; total added amount of different ESS by all actors
set Price      n-values 5 [?1 -> ?1] ; Price of each ESS
set f_targ     n-values 5 [?1 -> ?1]
set f_sum      n-values 5 [?1 -> ?1]
              ;   A   B   C

set harv_x     n-values 3 [?1 -> ?1] ; the added amount of yield by actor A, B and C
set harv_y     n-values 3 [?1 -> ?1]
set harv_z     n-values 3 [?1 -> ?1]
set harv_l     n-values 3 [?1 -> ?1]
set harv_m     n-values 3 [?1 -> ?1] ; the added amount of EC by actor A, B and C
set c_x        n-values 3 [?1 -> ?1] ; unit cost of adding yield by A, B and C
set c_y        n-values 3 [?1 -> ?1] ; unit cost of adding EC by A, B and C
set c_z        n-values 3 [?1 -> ?1] ; unit cost of adding EC by A, B and C
set c_l        n-values 3 [?1 -> ?1] ; unit cost of adding EC by A, B and C
set c_m        n-values 3 [?1 -> ?1] ; unit cost of adding EC by A, B and C
set u_x        n-values 3 [?1 -> ?1] ; intermediate to calculate the marginal value
set u_y        n-values 3 [?1 -> ?1]
set u_z        n-values 3 [?1 -> ?1]
set u_l        n-values 3 [?1 -> ?1]
```

```
set u_m        n-values 3 [?1 -> ?1]
set u          n-values 3 [?1 -> ?1] ; Benefits of each actor efforts
set w1_x       n-values 3 [?1 -> ?1] ; intermediate to calculate the marginal value
set w2_x       n-values 3 [?1 -> ?1]
set w3_x       n-values 3 [?1 -> ?1]
set w_x        n-values 3 [?1 -> ?1]
set sum_w_c_x  n-values 3 [?1 -> ?1]
set w1_y       n-values 3 [?1 -> ?1]
set w2_y       n-values 3 [?1 -> ?1]
set w3_y       n-values 3 [?1 -> ?1]
set w_y        n-values 3 [?1 -> ?1]
set sum_w_c_y  n-values 3 [?1 -> ?1]
set w1_z       n-values 3 [?1 -> ?1]
set w2_z       n-values 3 [?1 -> ?1]
set w3_z       n-values 3 [?1 -> ?1]
set w_z        n-values 3 [?1 -> ?1]
set sum_w_c_z  n-values 3 [?1 -> ?1]
set w1_l       n-values 3 [?1 -> ?1]
set w2_l       n-values 3 [?1 -> ?1]
set w3_l       n-values 3 [?1 -> ?1]
set w_l        n-values 3 [?1 -> ?1]
set sum_w_c_l  n-values 3 [?1 -> ?1]
set w1_m       n-values 3 [?1 -> ?1]
set w2_m       n-values 3 [?1 -> ?1]
set w3_m       n-values 3 [?1 -> ?1]
set w_m        n-values 3 [?1 -> ?1]
set sum_w_c_m  n-values 3 [?1 -> ?1]
set w1         n-values 3 [?1 -> ?1] ; Mutual coupling between efforts and values of actors
set w2         n-values 3 [?1 -> ?1]
set w3         n-values 3 [?1 -> ?1]
set w          n-values 3 [?1 -> ?1]
set sum_w_c    n-values 3 [?1 -> ?1]
```

```
set V          n-values 3 [?1 -> ?1] ; Value or profits
set C_max      n-values 3 [?1 -> ?1]
set C_targ     n-values 3 [?1 -> ?1] ;
set v_x        n-values 3 [?1 -> ?1] ; marginal value of x
set v_y        n-values 3 [?1 -> ?1] ; marginal value of y
set v_z        n-values 3 [?1 -> ?1] ; marginal value of y
set v_l        n-values 3 [?1 -> ?1] ; marginal value of y
set v_m        n-values 3 [?1 -> ?1] ; marginal value of y
set rv_x       n-values 3 [?1 -> ?1]
set rv_y       n-values 3 [?1 -> ?1]
set rv_z       n-values 3 [?1 -> ?1]
set rv_l       n-values 3 [?1 -> ?1]
set rv_m       n-values 3 [?1 -> ?1]
set sum_rv     n-values 3 [?1 -> ?1]
set r_x_targ   n-values 3 [?1 -> ?1]
set r_y_targ   n-values 3 [?1 -> ?1]
set r_z_targ   n-values 3 [?1 -> ?1]
set r_l_targ   n-values 3 [?1 -> ?1]
set r_m_targ   n-values 3 [?1 -> ?1]
set f_x        n-values 3 [?1 -> ?1] ; effective effort of each actor for ecosystem x
set f_y        n-values 3 [?1 -> ?1]
set f_z        n-values 3 [?1 -> ?1]
set f_l        n-values 3 [?1 -> ?1]
set f_m        n-values 3 [?1 -> ?1]
set phi_x      n-values 3 [?1 -> ?1] ; share of effective effort of each actor for ecosystem x
set phi_y      n-values 3 [?1 -> ?1]
set phi_z      n-values 3 [?1 -> ?1]
set phi_l      n-values 3 [?1 -> ?1]
set phi_m      n-values 3 [?1 -> ?1]
 ]
```

```
calc_intermediate

calc_derivative




foreach ESS-type-list [?1 ->

   ask LS [

  set growth   replace-item ?1 growth   ((g2 * item ?1 S-ESS * (1 - (item ?1 S-ESS / L))) - (n * item ?1 S-ESS))     ; delta s: the change of the supply

  set tot_harv replace-item 0 tot_harv (sum harv_x) ; total change of yield by all actors A, B and C

  set tot_harv replace-item 1 tot_harv (sum harv_y) ; total change of EC by all actors A, B and C

  set tot_harv replace-item 2 tot_harv (sum harv_z) ; total change of CS by all actors A, B and C

  set tot_harv replace-item 3 tot_harv (sum harv_l) ; total change of water by all actors A, B and C

  set tot_harv replace-item 4 tot_harv (sum harv_m) ; total change of biodiversity by all actors A, B and C


  set Price replace-item ?1 Price (item ?1 a - (item ?1 b2 * item ?1 tot_harv)) ; price of added yield and EC




   set P-fin replace-item ?1 P-fin (((item 0 a-share  + item 1 a-share  + item 2 a-share ) * item ?1 a) - (((item ?1 b2-1  + item ?1 b2-2  + item ?1 b2-3 ) / 3) * item ?1 tot_harv))

   set P-soc replace-item ?1 P-soc (((item 3 a-share  + item 4 a-share  + item 5 a-share ) * item ?1 a) - (((item ?1 b2-4  + item ?1 b2-5  + item ?1 b2-6 ) / 3) * item ?1 tot_harv))

   set P-phy replace-item ?1 P-phy (((item 6 a-share  + item 7 a-share  + item 8 a-share ) * item ?1 a) - (((item ?1 b2-7  + item ?1 b2-8  + item ?1 b2-9 ) / 3) * item ?1 tot_harv))

   set P-hum replace-item ?1 P-hum (((item 9 a-share  + item 10 a-share + item 11 a-share) * item ?1 a) - (((item ?1 b2-10 + item ?1 b2-11 + item ?1 b2-12) / 3) * item ?1 tot_harv))

   set P-cul replace-item ?1 P-cul (((item 12 a-share + item 13 a-share + item 14 a-share) * item ?1 a) - (((item ?1 b2-13 + item ?1 b2-4  + item ?1 b2-15) / 3) * item ?1 tot_harv))


   set P-1  replace-item ?1 P-1  ((item 0 a-share * item ?1 a) - (item ?1 b2-1 * item ?1 tot_harv))

   set P-2  replace-item ?1 P-2  ((item 1 a-share * item ?1 a) - (item ?1 b2-2 * item ?1 tot_harv))

   set P-3  replace-item ?1 P-3  ((item 2 a-share * item ?1 a) - (item ?1 b2-3 * item ?1 tot_harv))

   set P-4  replace-item ?1 P-4  ((item 3 a-share * item ?1 a) - (item ?1 b2-4 * item ?1 tot_harv))
```

```
set P-5  replace-item ?1 P-5  ((item 4 a-share * item ?1 a) - (item ?1 b2-5 * item ?1 tot_harv))
set P-6  replace-item ?1 P-6  ((item 5 a-share * item ?1 a) - (item ?1 b2-6 * item ?1 tot_harv))
set P-7  replace-item ?1 P-7  ((item 6 a-share * item ?1 a) - (item ?1 b2-7 * item ?1 tot_harv))
set P-8  replace-item ?1 P-8  ((item 7 a-share * item ?1 a) - (item ?1 b2-8 * item ?1 tot_harv))
set P-9  replace-item ?1 P-9  ((item 8 a-share * item ?1 a) - (item ?1 b2-9 * item ?1 tot_harv))
set P-10 replace-item ?1 P-10 ((item 9 a-share * item ?1 a) - (item ?1 b2-10 * item ?1 tot_harv))
set P-11 replace-item ?1 P-11 ((item 10 a-share * item ?1 a) - (item ?1 b2-11 * item ?1 tot_harv))
set P-12 replace-item ?1 P-12 ((item 11 a-share * item ?1 a) - (item ?1 b2-12 * item ?1 tot_harv))
set P-13 replace-item ?1 P-13 ((item 12 a-share * item ?1 a) - (item ?1 b2-13 * item ?1 tot_harv))
set P-14 replace-item ?1 P-14 ((item 13 a-share * item ?1 a) - (item ?1 b2-14 * item ?1 tot_harv))
set P-15 replace-item ?1 P-15 ((item 14 a-share * item ?1 a) - (item ?1 b2-15 * item ?1 tot_harv))

  set Price-plot Price

if capital   = "Financial capital"  [ set Price-plot P-fin]
if capital   = "Social capital"     [ set Price-plot P-soc]
if capital   = "Physical capital"   [ set Price-plot P-phy]
if capital   = "Human capital"      [ set Price-plot P-hum]
if capital   = "Cultural capital"   [ set Price-plot P-cul]

 if Indicator = "Income"        [ set Price-plot P-1 ]
 if Indicator = "Expenditure"   [ set Price-plot P-2 ]
 if Indicator = "Savings"       [ set Price-plot P-3 ]
 if Indicator = "Soc_insurance" [ set Price-plot P-4 ]
 if Indicator = "H_insurance"   [ set Price-plot P-5 ]
 if Indicator = "Agr_insurance" [ set Price-plot P-6 ]
 if Indicator = "Equipment"     [ set Price-plot P-7 ]
 if Indicator = "Internet"      [ set Price-plot P-8 ]
 if Indicator = "Vehicles"      [ set Price-plot P-9 ]
 if Indicator = "Health"        [ set Price-plot P-10 ]
 if Indicator = "Nutrition"     [ set Price-plot P-11 ]
 if Indicator = "Skills"        [ set Price-plot P-12 ]
```

```
   if Indicator = "Celebration"    [ set Price-plot P-13 ]

   if Indicator = "Beliefs"       [ set Price-plot P-14 ]

   if Indicator = "Traditions"    [ set Price-plot P-15 ]


     if Interaction-scenario = "cooperative-optimizing" [

  set f_sum replace-item 0 f_sum (sum f_x)

  set f_sum replace-item 1 f_sum (sum f_y)

  set f_sum replace-item 2 f_sum (sum f_z)

  set f_sum replace-item 3 f_sum (sum f_l)

  set f_sum replace-item 4 f_sum (sum f_m)



     set f_targ replace-item ?1 f_targ ( n - ( g2 * (1 - (item ?1 S-ESS / L))))


     ]

  ]]


  foreach actors [ ?1 ->

    ask LS [

      if Interaction-scenario = "Competitive-optimizing" [

   set C_targ replace-item ?1 C_targ ( ((item ?1 u - (item ?1 sum_w_c - item ?1 w ))) * item ?1 C / (2 *
item ?1 w * 1000))


  set r_x_targ replace-item ?1 r_x_targ (1 / item ?1 c_x ^ 2 * (item ?1 u_x - item ?1 sum_w_c_x - item
?1 w_x ) / (2 * item ?1 C * item 0 b2 * 10000000) )

  set r_y_targ replace-item ?1 r_y_targ (1 / item ?1 c_y ^ 2 * (item ?1 u_y - item ?1 sum_w_c_y - item
?1 w_y ) / (2 * item ?1 C * item 1 b2 * 10000000) )

  set r_z_targ replace-item ?1 r_z_targ (1 / item ?1 c_z ^ 2 * (item ?1 u_z - item ?1 sum_w_c_z - item
?1 w_z ) / (2 * item ?1 C * item 2 b2 * 10000000) )

  set r_l_targ replace-item ?1 r_l_targ (1 / item ?1 c_l ^ 2 * (item ?1 u_l - item ?1 sum_w_c_l - item ?1
w_l ) / (2 * item ?1 C * item 3 b2 * 10000000) )

  set r_m_targ replace-item ?1 r_m_targ (1 / item ?1 c_m ^ 2 * (item ?1 u_m - item ?1 sum_w_c_m -
item ?1 w_m ) / (2 * item ?1 C * item 4 b2 * 10000000) )

     ]
```

```
    if Interaction-scenario = "cooperative-optimizing" [
      ifelse item 0 f_sum = 0 [set phi_x replace-item ?1 phi_x (0)]
      [set phi_x replace-item ?1 phi_x (item ?1 f_x / item 0 f_sum)]
       ifelse item 1 f_sum = 0 [set phi_y replace-item ?1 phi_y (0)]
      [set phi_y replace-item ?1 phi_y (item ?1 f_y / item 1 f_sum)]
       ifelse item 2 f_sum = 0 [set phi_z replace-item ?1 phi_z (0)]
      [set phi_z replace-item ?1 phi_z (item ?1 f_z / item 2 f_sum)]
       ifelse item 3 f_sum = 0 [set phi_l replace-item ?1 phi_l (0)]
      [set phi_l replace-item ?1 phi_l (item ?1 f_l / item 3 f_sum)]
       ifelse item 4 f_sum = 0 [set phi_m replace-item ?1 phi_m (0)]
      [set phi_m replace-item ?1 phi_m (item ?1 f_m / item 4 f_sum)]

   ]
 ]
 ]

calc_value
 reset-ticks

end

;
##########################################################################################
to calc_intermediate

 foreach actors [?1 ->

    ask LS [
  ifelse item ?1 d_x = 0.1 or item 0 S-ESS = 0 or item 0 S-ESS >= item ?1 d_x [set c_x replace-item ?1
c_x (0.0001)]
    [set c_x replace-item ?1 c_x (  (item ?1 q_x * item 0 S-ESS * item 0 S-ESS / item ?1 x_per_x * (1 -
(item 0 S-ESS / item ?1 d_x))))] ; this is the inverse of c (unit effort) to avoid division by zero
```

```
  ifelse item ?1 d_y = 0.1 or item 1 S-ESS = 0 or item 1 S-ESS >= item ?1 d_y  [set c_y replace-item ?1 c_y (0.0001)]

    [set c_y replace-item ?1 c_y (  (item ?1 q_y * item 1 S-ESS * item 1 S-ESS / item ?1 x_per_y * (1 - (item 1 S-ESS / item ?1 d_y))))]

  ifelse item ?1 d_z = 0.1 or item 2 S-ESS = 0 or item 2 S-ESS >= item ?1 d_z  [set c_z replace-item ?1 c_z (0.0001)]

    [set c_z replace-item ?1 c_z (  (item ?1 q_z * item 2 S-ESS * item 2 S-ESS / item ?1 x_per_z * (1 - (item 2 S-ESS / item ?1 d_z))))]

  ifelse item ?1 d_l = 0.1 or item 3 S-ESS = 0 or item 3 S-ESS >= item ?1 d_l  [set c_l replace-item ?1 c_l (0.0001)]

    [set c_l replace-item ?1 c_l (  (item ?1 q_l * item 3 S-ESS * item 3 S-ESS / item ?1 x_per_l * (1 - (item 3 S-ESS / item ?1 d_l))))]

  ifelse item ?1 d_m = 0.1 or item 4 S-ESS = 0 or item 4 S-ESS >= item ?1 d_m  [set c_m replace-item ?1 c_m (0.0001)]

    [set c_m replace-item ?1 c_m (  (item ?1 q_m * item 4 S-ESS * item 4 S-ESS / item ?1 x_per_m * (1 - (item 4 S-ESS / item ?1 d_m))))]


 set harv_x replace-item ?1 harv_x (item ?1 C-x  * item ?1 c_x)

 set harv_y replace-item ?1 harv_y (item ?1 C-y  * item ?1 c_y)

 set harv_z replace-item ?1 harv_z (item ?1 C-z  * item ?1 c_z)

 set harv_l replace-item ?1 harv_l (item ?1 C-l  * item ?1 c_l)

 set harv_m replace-item ?1 harv_m (item ?1 C-m  * item ?1 c_m)


    set u_x replace-item ?1 u_x (item 0 a * item ?1 c_x)   ; This is used for the calculation of the first component of the marginal value for each actor

    set u_y replace-item ?1 u_y (item 1 a * item ?1 c_y)

    set u_z replace-item ?1 u_z (item 2 a * item ?1 c_z)

    set u_l replace-item ?1 u_l (item 3 a * item ?1 c_l)

    set u_m replace-item ?1 u_m (item 4 a * item ?1 c_m)

    set u replace-item ?1 u ((item ?1 u_x * item ?1 r_x) + (item ?1 u_y * item ?1 r_y) + (item ?1 u_z * item ?1 r_z) + (item ?1 u_l * item ?1 r_l) + (item ?1 u_m * item ?1 r_m) - 1)

                    ; this is used for the calculation of profit (first component) and the change of investment (last component)
```

```
   set f_x replace-item ?1 f_x (item ?1 q_x * item ?1 C-x )
   set f_y replace-item ?1 f_y (item ?1 q_y * item ?1 C-y )
   set f_z replace-item ?1 f_z (item ?1 q_z * item ?1 C-z )
   set f_l replace-item ?1 f_l (item ?1 q_l * item ?1 C-l )
   set f_m replace-item ?1 f_m (item ?1 q_m * item ?1 C-m )
 ]
]
end

to calc_derivative

foreach actors [?1 ->
    ask LS [
     ;This is used for the calculation of the marginal value v of ESS x for actor 1, the sum represents the middle component which is the impact of others efforts on self actor values
  set w1_x replace-item ?1 w1_x (item 0 b2 * item ?1 r_x * ( item ?1 c_x * item 0 c_x )  * item ?1 C)
  set w2_x replace-item ?1 w2_x (item 0 b2 * item ?1 r_x * ( item ?1 c_x * item 1 c_x )  * item ?1 C)
  set w3_x replace-item ?1 w3_x (item 0 b2 * item ?1 r_x * ( item ?1 c_x * item 2 c_x )  * item ?1 C)

  set w_x  replace-item ?1 w_x  (item 0 b2 * item ?1 r_x * ( item ?1 c_x * item ?1 c_x )  * item ?1 C)  ; This is used for the calculation of the marginal value of ESS x for all actor representing the impact of self effort.
 set sum_w_c_x replace-item 0 sum_w_c_x (sum w1_x)  ; the middle component for marginal value x each item for each actor
 set sum_w_c_x replace-item 1 sum_w_c_x (sum w2_x)
 set sum_w_c_x replace-item 2 sum_w_c_x (sum w3_x)

  set w1_y replace-item ?1 w1_y (item 1 b2 * item ?1 r_y * ( item ?1 c_y * item 0 c_y ) * item ?1 C)
  set w2_y replace-item ?1 w2_y (item 1 b2 * item ?1 r_y * ( item ?1 c_y * item 1 c_y ) * item ?1 C)
  set w3_y replace-item ?1 w3_y (item 1 b2 * item ?1 r_y * ( item ?1 c_y * item 2 c_y ) * item ?1 C)

  set w_y  replace-item ?1 w_y  (item 1 b2 * item ?1 r_y * ( item ?1 c_y * item ?1 c_y ) * item ?1 C)
```

```
set sum_w_c_y replace-item 0 sum_w_c_y (sum w1_y)
set sum_w_c_y replace-item 1 sum_w_c_y (sum w2_y)
set sum_w_c_y replace-item 2 sum_w_c_y (sum w3_y)

set w1_z replace-item ?1 w1_z (item 2 b2 * item ?1 r_z * ( item ?1 c_z * item 0 c_z ) * item ?1 C)
set w2_z replace-item ?1 w2_z (item 2 b2 * item ?1 r_z * ( item ?1 c_z * item 1 c_z ) * item ?1 C)
set w3_z replace-item ?1 w3_z (item 2 b2 * item ?1 r_z * ( item ?1 c_z * item 2 c_z ) * item ?1 C)

set w_z  replace-item ?1 w_z  (item 2 b2 * item ?1 r_z * ( item ?1 c_z * item ?1 c_z ) * item ?1 C)
set sum_w_c_z replace-item 0 sum_w_c_z (sum w1_z)
set sum_w_c_z replace-item 1 sum_w_c_z (sum w2_z)
set sum_w_c_z replace-item 2 sum_w_c_z (sum w3_z)

set w1_l replace-item ?1 w1_l (item 3 b2 * item ?1 r_l * ( item ?1 c_l * item 0 c_l ) * item ?1 C)
set w2_l replace-item ?1 w2_l (item 3 b2 * item ?1 r_l * ( item ?1 c_l * item 1 c_l ) * item ?1 C)
set w3_l replace-item ?1 w3_l (item 3 b2 * item ?1 r_l * ( item ?1 c_l * item 2 c_l ) * item ?1 C)

set w_l  replace-item ?1 w_l  (item 3 b2 * item ?1 r_l * ( item ?1 c_l * item ?1 c_l ) * item ?1 C)
set sum_w_c_l replace-item 0 sum_w_c_l (sum w1_l)
set sum_w_c_l replace-item 1 sum_w_c_l (sum w2_l)
set sum_w_c_l replace-item 2 sum_w_c_l (sum w3_l)

set w1_m replace-item ?1 w1_m (item 4 b2 * item ?1 r_m * ( item ?1 c_m * item 0 c_m ) * item ?1
C)
set w2_m replace-item ?1 w2_m (item 4 b2 * item ?1 r_m * ( item ?1 c_m * item 1 c_m ) * item ?1
C)
set w3_m replace-item ?1 w3_m (item 4 b2 * item ?1 r_m * ( item ?1 c_m * item 2 c_m ) * item ?1
C)

set w_m  replace-item ?1 w_m  (item 4 b2 * item ?1 r_m * ( item ?1 c_m * item ?1 c_m ) * item ?1
C)
set sum_w_c_m replace-item 0 sum_w_c_m (sum w1_m)
set sum_w_c_m replace-item 1 sum_w_c_m (sum w2_m)
```

```
 set sum_w_c_m replace-item 2 sum_w_c_m (sum w3_m)



 set w1 replace-item ?1 w1 ((item ?1 w1_x * item 0 r_x) + (item ?1 w1_y * item 0 r_y) + (item ?1 w1_z * item 0 r_z) + (item ?1 w1_l * item 0 r_l) + (item ?1 w1_m * item 0 r_m)) ; This is to be summed with other actors to claculate the the profit and the change in investments sum of services w_ij * C_j

 set w2 replace-item ?1 w2 ((item ?1 w2_x * item 1 r_x) + (item ?1 w2_y * item 1 r_y) + (item ?1 w2_z * item 1 r_z) + (item ?1 w2_l * item 1 r_l) + (item ?1 w2_m * item 1 r_m))

 set w3 replace-item ?1 w3 ((item ?1 w3_x * item 2 r_x) + (item ?1 w3_y * item 2 r_y) + (item ?1 w3_z * item 2 r_z) + (item ?1 w3_l * item 2 r_l) + (item ?1 w3_m * item 2 r_m))


 set w  replace-item ?1 w  ((item ?1 w_x  * item ?1 r_x) + (item ?1 w_y  * item ?1 r_y)  + (item ?1 w_z * item ?1 r_z)  + (item ?1 w_l  * item ?1 r_l)  + (item ?1 w_m  * item ?1 r_m)) ; this is used for the calculation of the last item of the change in investments.

 set sum_w_c replace-item 0 sum_w_c (sum w1) ; This is the middle component in the calculation of investment and the last component for the calculation of the profit.

 set sum_w_c replace-item 1 sum_w_c (sum w2)

 set sum_w_c replace-item 2 sum_w_c (sum w3)


 ]]

end


to calc_value


foreach actors [?1 ->

   ask LS [

    ;ifelse item ?1 u <= 0 [set V replace-item ?1 V (0)][

    ifelse Interaction-scenario = "cooperative-optimizing" [

      set V replace-item ?1 V ( (item 0 Price * item ?1 harv_x) + (item 1 Price * item ?1 harv_y) + (item 2 Price * item ?1 harv_z) + (item 3 Price * item ?1 harv_l) + (item 4 Price * item ?1 harv_m))

    ]

    [set V replace-item ?1 V ((item ?1 u - item ?1 sum_w_c) * item ?1 C)] ; Utility function
```

```
    set V-fin replace-item ?1 V-fin ((item 0 a-share  + item 1 a-share  + item 2 a-share  ) * item ?1 V)
    set V-soc replace-item ?1 V-soc ((item 3 a-share  + item 4 a-share  + item 5 a-share  ) * item ?1 V)
    set V-phy replace-item ?1 V-phy ((item 6 a-share  + item 7 a-share  + item 8 a-share  ) * item ?1 V)
    set V-hum replace-item ?1 V-hum ((item 9 a-share  + item 10 a-share + item 11 a-share ) * item ?1
V)
    set V-cul replace-item ?1 V-cul ((item 12 a-share + item 13 a-share + item 14 a-share ) * item ?1 V)


    set V-1 replace-item ?1 V-1 (item 0 a-share  * item ?1 V)
    set V-2 replace-item ?1 V-2 (item 1 a-share  * item ?1 V)
    set V-3 replace-item ?1 V-3 (item 2 a-share  * item ?1 V)
    set V-4 replace-item ?1 V-4 (item 3 a-share  * item ?1 V)
    set V-5 replace-item ?1 V-5 (item 4 a-share  * item ?1 V)
    set V-6 replace-item ?1 V-6 (item 5 a-share  * item ?1 V)
    set V-7 replace-item ?1 V-7 (item 6 a-share  * item ?1 V)
    set V-8 replace-item ?1 V-8 (item 7 a-share  * item ?1 V)
    set V-9 replace-item ?1 V-9 (item 8 a-share  * item ?1 V)
    set V-10 replace-item ?1 V-10 (item 9 a-share  * item ?1 V)
    set V-11 replace-item ?1 V-11 (item 10 a-share  * item ?1 V)
    set V-12 replace-item ?1 V-12 (item 11 a-share  * item ?1 V)
    set V-13 replace-item ?1 V-13 (item 12 a-share  * item ?1 V)
    set V-14 replace-item ?1 V-14 (item 13 a-share  * item ?1 V)
    set V-15 replace-item ?1 V-15 (item 14 a-share  * item ?1 V)



  set V-plot V


  if capital   = "Financial capital"  [ set V-plot V-fin]
  if capital   = "Social capital"     [ set V-plot V-soc]
  if capital   = "Physical capital"   [ set V-plot V-phy]
  if capital   = "Human capital"      [ set V-plot V-hum]
  if capital   = "Cultural capital"   [ set V-plot V-cul]
```

```
  if Indicator = "Income"        [ set V-plot V-1 ]
  if Indicator = "Expenditure"   [ set V-plot V-2 ]
  if Indicator = "Savings"       [ set V-plot V-3 ]
  if Indicator = "Soc_insurance" [ set V-plot V-4 ]
  if Indicator = "H_insurance"   [ set V-plot V-5 ]
  if Indicator = "Agr_insurance" [ set V-plot V-6 ]
  if Indicator = "Equipment"     [ set V-plot V-7 ]
  if Indicator = "Internet"      [ set V-plot V-8 ]
  if Indicator = "Vehicles"      [ set V-plot V-9 ]
  if Indicator = "Health"        [ set V-plot V-10 ]
  if Indicator = "Nutrition"     [ set V-plot V-11 ]
  if Indicator = "Skills"        [ set V-plot V-12 ]
  if Indicator = "Celebration"   [ set V-plot V-13 ]
  if Indicator = "Beliefs"       [ set V-plot V-14 ]
  if Indicator = "Traditions"    [ set V-plot V-15 ]


 set C_max replace-item ?1 C_max ( f * item ?1 K)
 if item ?1 C_max < 0 [
 set C_max replace-item ?1 C_max   ( 0 )]
 set v_x replace-item ?1 v_x ((item ?1 u_x - item ?1 sum_w_c_x - item ?1 w_x) * item ?1 C)
 set v_y replace-item ?1 v_y ((item ?1 u_y - item ?1 sum_w_c_y - item ?1 w_y) * item ?1 C)
 set v_z replace-item ?1 v_z ((item ?1 u_z - item ?1 sum_w_c_z - item ?1 w_z) * item ?1 C)
 set v_l replace-item ?1 v_l ((item ?1 u_l - item ?1 sum_w_c_l - item ?1 w_l) * item ?1 C)
 set v_m replace-item ?1 v_m ((item ?1 u_m - item ?1 sum_w_c_m - item ?1 w_m) * item ?1 C)
 ]]


end

;;#############################################################################
```

```
to go


  if Month = "January" [set month-num 1] if Month = "February" [set month-num 2] if Month =
"March" [set month-num 3] if Month = "April" [set month-num 4] if Month = "May" [set month-num
11]

  if Month = "June" [set month-num 5] if Month = "July" [set month-num 6] if Month = "August" [set
month-num 7] if Month = "September" [set month-num 8] if Month = "October" [set month-num 9]

  if Month = "Novemebr" [set month-num 10] if Month = "December" [set month-num 12]

 set duration ((year - 2020) * 12) + (month-num - 12)


 ; if ticks >= duration   [ stop ]

  if ticks >= 10   [ stop ] ; this version of the model has been developed and tested with values
changing per year for 10 years since the data included are not real data.


;;*****************************************************************************

  foreach actors [?1 ->

    ask LS [

    if Interaction-scenario = "Competitive-gradient" [

      set C   replace-item ?1 C   (item ?1 C + (kappa * item ?1 C * (item ?1 C_max - item ?1 C) * (item ?1
u - item ?1 sum_w_c - item ?1 w)))]

    if Interaction-scenario = "Competitive-optimizing" [


    set C replace-item ?1 C (item ?1 C + (kappa * (item ?1 C_targ - item ?1 C)))]


      if Interaction-scenario = "cooperative-optimizing" [

        set C-x replace-item ?1 C-x ( item ?1 C-x + (gama * (item 0 f_targ - item 0 f_sum) * item ?1 phi_x
/ item ?1 q_x))

        set C-y replace-item ?1 C-y ( item ?1 C-y + (gama * (item 1 f_targ - item 1 f_sum) * item ?1
phi_y / item ?1 q_y))

        set C-z replace-item ?1 C-z ( item ?1 C-z + (gama * (item 2 f_targ - item 2 f_sum) * item ?1 phi_z
/ item ?1 q_z))

        set C-l replace-item ?1 C-l ( item ?1 C-l + (gama * (item 3 f_targ - item 3 f_sum) * item ?1 phi_l /
item ?1 q_l))

        set C-m replace-item ?1 C-m ( item ?1 C-m + (gama * (item 4 f_targ - item 4 f_sum) * item ?1
phi_m / item ?1 q_m))
```

```
    set C replace-item ?1 C (item ?1 C-x + item ?1 C-y + item ?1 C-z + item ?1 C-l + item ?1 C-m) ]


  set rv_x replace-item ?1 rv_x (item ?1 r_x * item ?1 v_x)

  set rv_y replace-item ?1 rv_y (item ?1 r_y * item ?1 v_y)

  set rv_z replace-item ?1 rv_z (item ?1 r_z * item ?1 v_z)

  set rv_l replace-item ?1 rv_l (item ?1 r_l * item ?1 v_l)

  set rv_m replace-item ?1 rv_m (item ?1 r_m * item ?1 v_m)

  set sum_rv replace-item ?1 sum_rv (item ?1 rv_x + item ?1 rv_y + item ?1 rv_z + item ?1 rv_l + item
?1 rv_m)


  if Interaction-scenario = "Competitive-gradient" [

  set r_x replace-item ?1 r_x (item ?1 r_x + (alpha * item ?1 r_x * (item ?1 v_x - item ?1 sum_rv)))

  set r_y replace-item ?1 r_y (item ?1 r_y + (alpha * item ?1 r_y * (item ?1 v_y - item ?1 sum_rv)))

  set r_z replace-item ?1 r_z (item ?1 r_z + (alpha * item ?1 r_z * (item ?1 v_z - item ?1 sum_rv)))

  set r_l replace-item ?1 r_l (item ?1 r_l + (alpha * item ?1 r_l * (item ?1 v_l - item ?1 sum_rv)))

  set r_m replace-item ?1 r_m (item ?1 r_m + (alpha * item ?1 r_m * (item ?1 v_m - item ?1
sum_rv)))]

  if Interaction-scenario = "Competitive-optimizing" [


  set r_x replace-item ?1 r_x (item ?1 r_x + (alpha *  (item ?1 r_x_targ - item ?1 r_x)))

  set r_y replace-item ?1 r_y (item ?1 r_y + (alpha *  (item ?1 r_y_targ - item ?1 r_y)))

  set r_z replace-item ?1 r_z (item ?1 r_z + (alpha *  (item ?1 r_z_targ - item ?1 r_z)))

  set r_l replace-item ?1 r_l (item ?1 r_l + (alpha *  (item ?1 r_l_targ - item ?1 r_l)))

  set r_m replace-item ?1 r_m (item ?1 r_m + (alpha *  (item ?1 r_m_targ - item ?1 r_m)))

      if item ?1 r_x < 0 [set r_x replace-item ?1 r_x (0)] if item ?1 r_x > 1 [set r_x replace-item ?1 r_x
(1)]

      if item ?1 r_y < 0 [set r_y replace-item ?1 r_y (0)] if item ?1 r_y > 1 [set r_y replace-item ?1 r_y
(1)]

      if item ?1 r_z < 0 [set r_z replace-item ?1 r_z (0)] if item ?1 r_z > 1 [set r_z replace-item ?1 r_z
(1)]

      if item ?1 r_l < 0 [set r_l replace-item ?1 r_l (0)] if item ?1 r_l > 1 [set r_l replace-item ?1 r_l (1)]

      if item ?1 r_m < 0 [set r_m replace-item ?1 r_m (0)] if item ?1 r_m > 1 [set r_m replace-item ?1
r_m (1)]

    ]
```

```
    if Interaction-scenario = "cooperative-optimizing" [
 set r_x replace-item ?1 r_x (item ?1 C-x / item ?1 C)
 set r_y replace-item ?1 r_y (item ?1 C-y / item ?1 C)
 set r_z replace-item ?1 r_z (item ?1 C-z / item ?1 C)
 set r_l replace-item ?1 r_l (item ?1 C-l / item ?1 C)
 set r_m replace-item ?1 r_m (item ?1 C-m / item ?1 C)
    ]

 set K replace-item ?1 K (item ?1 K + item ?1 V)

   set C-fin replace-item ?1 C-fin  ((item 0  C-share + item 1  C-share + item 2 C-share ) * item ?1 C) set K-fin replace-item ?1 K-fin  ((item ?1 K0-1  + item ?1 K0-2  + item ?1 K0-3 )  * item ?1 K)
   set C-soc replace-item ?1 C-soc  ((item 3  C-share + item 4  C-share + item 5 C-share ) * item ?1 C) set K-soc replace-item ?1 K-soc  ((item ?1 K0-4  + item ?1 K0-5  + item ?1 K0-6 )  * item ?1 K)
   set C-phy replace-item ?1 C-phy  ((item 6  C-share + item 7  C-share + item 8 C-share ) * item ?1 C) set K-phy replace-item ?1 K-phy  ((item ?1 K0-7  + item ?1 K0-8  + item ?1 K0-9 )  * item ?1 K)
   set C-hum replace-item ?1 C-hum  ((item 9  C-share + item 10 C-share + item 11 C-share) * item ?1 C) set K-hum replace-item ?1 K-hum  ((item ?1 K0-10 + item ?1 K0-11 + item ?1 K0-12)  * item ?1 K)
   set C-cul replace-item ?1 C-cul  ((item 12 C-share + item 13 C-share + item 14 C-share) * item ?1 C) set K-cul replace-item ?1 K-cul  ((item ?1 K0-13 + item ?1 K0-14 + item ?1 K0-15)  * item ?1 K)

    set K-1 replace-item ?1 K-1 (item ?1 K0-1 * item ?1 K) set C-1 replace-item ?1 C-1 (item 0 C-share * item ?1 C)
    set K-2 replace-item ?1 K-2 (item ?1 K0-2 * item ?1 K) set C-2 replace-item ?1 C-2 (item 1 C-share * item ?1 C)
    set K-3 replace-item ?1 K-3 (item ?1 K0-3 * item ?1 K) set C-3 replace-item ?1 C-3 (item 2 C-share * item ?1 C)
    set K-4 replace-item ?1 K-4 (item ?1 K0-4 * item ?1 K) set C-4 replace-item ?1 C-4 (item 3 C-share * item ?1 C)
    set K-5 replace-item ?1 K-5 (item ?1 K0-5 * item ?1 K) set C-5 replace-item ?1 C-5 (item 4 C-share * item ?1 C)
    set K-6 replace-item ?1 K-6 (item ?1 K0-6 * item ?1 K) set C-6 replace-item ?1 C-6 (item 5 C-share * item ?1 C)
```

```
    set K-7 replace-item ?1 K-7 (item ?1 K0-7 * item ?1 K) set C-7 replace-item ?1 C-7 (item 6 C-share * item ?1 C)

    set K-8 replace-item ?1 K-8 (item ?1 K0-8 * item ?1 K) set C-8 replace-item ?1 C-8 (item 7 C-share * item ?1 C)

    set K-9 replace-item ?1 K-9 (item ?1 K0-9 * item ?1 K) set C-9 replace-item ?1 C-9 (item 8 C-share * item ?1 C)

    set K-10 replace-item ?1 K-10 (item ?1 K0-10 * item ?1 K) set C-10 replace-item ?1 C-10 (item 9 C-share * item ?1 C)

    set K-11 replace-item ?1 K-11 (item ?1 K0-11 * item ?1 K) set C-11 replace-item ?1 C-11 (item 10 C-share * item ?1 C)

    set K-12 replace-item ?1 K-12 (item ?1 K0-12 * item ?1 K) set C-12 replace-item ?1 C-12 (item 11 C-share * item ?1 C)

    set K-13 replace-item ?1 K-13 (item ?1 K0-13 * item ?1 K) set C-13 replace-item ?1 C-13 (item 12 C-share * item ?1 C)

    set K-14 replace-item ?1 K-14 (item ?1 K0-14 * item ?1 K) set C-14 replace-item ?1 C-14 (item 13 C-share * item ?1 C)

    set K-15 replace-item ?1 K-15 (item ?1 K0-15 * item ?1 K) set C-15 replace-item ?1 C-15 (item 14 C-share * item ?1 C)


  set K-plot K

  set C-plot C

   if capital   = "Financial capital"  [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-fin) Set C-y replace-item ?1 C-y (item ?1 r_y * item ?1 C-fin) Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-fin)

                                  Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-fin) Set C-m replace-item ?1 C-m (item ?1 r_m * item ?1 C-fin) set K-plot K-fin set C-plot C-fin]

   if capital   = "Social capital"     [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-soc) Set C-y replace-item ?1 C-y (item ?1 r_y * item ?1 C-soc) Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-soc)

                                  Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-soc) Set C-m replace-item ?1 C-m (item ?1 r_m * item ?1 C-soc) set K-plot K-soc set C-plot C-soc]

   if capital   = "Physical capital"   [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-phy) Set C-y replace-item ?1 C-y (item ?1 r_y * item ?1 C-phy) Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-phy)

                                  Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-phy) Set C-m replace-item ?1 C-m (item ?1 r_m * item ?1 C-phy) set K-plot K-phy set C-plot C-phy]

   if capital   = "Human capital"      [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-hum) Set C-y replace-item ?1 C-y (item ?1 r_y * item ?1 C-hum) Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-hum)
```

Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-hum) Set C-m replace-item ?1 C-m (item ?1 r_m * item ?1 C-hum) set K-plot K-hum set C-plot C-hum]

if capital = "Cultural capital" [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-cul) Set C-y replace-item ?1 C-y (item ?1 r_y * item ?1 C-cul) Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-cul)

Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-cul) Set C-m replace-item ?1 C-m (item ?1 r_m * item ?1 C-cul) set K-plot K-cul set C-plot C-cul]

if Indicator = "Income" [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-1) Set C-y replace-item ?1 C-y (item ?1 r_y * item ?1 C-1) Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-1)

Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-1) Set C-m replace-item ?1 C-m (item ?1 r_m * item ?1 C-1) set K-plot K-1 set C-plot C-1 ]

if Indicator = "Expenditure" [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-2) Set C-y replace-item ?1 C-y (item ?1 r_y * item ?1 C-2) Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-2)

Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-2) Set C-m replace-item ?1 C-m (item ?1 r_m * item ?1 C-2) set K-plot K-2 set C-plot C-2 ]

if Indicator = "Savings" [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-3) Set C-y replace-item ?1 C-y (item ?1 r_y * item ?1 C-3) Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-3)

Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-3) Set C-m replace-item ?1 C-m (item ?1 r_m * item ?1 C-3) set K-plot K-3 set C-plot C-3 ]

if Indicator = "Soc_insurance" [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-4) Set C-y replace-item ?1 C-y (item ?1 r_y * item ?1 C-4) Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-4)

Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-4) Set C-m replace-item ?1 C-m (item ?1 r_m * item ?1 C-4) set K-plot K-4 set C-plot C-4 ]

if Indicator = "H_insurance" [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-5) Set C-y replace-item ?1 C-y (item ?1 r_y * item ?1 C-5) Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-5)

Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-5) Set C-m replace-item ?1 C-m (item ?1 r_m * item ?1 C-5) set K-plot K-5 set C-plot C-5 ]

if Indicator = "Agr_insurance" [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-6) Set C-y replace-item ?1 C-y (item ?1 r_y * item ?1 C-6) Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-6)

Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-6) Set C-m replace-item ?1 C-m (item ?1 r_m * item ?1 C-6) set K-plot K-6 set C-plot C-6 ]

if Indicator = "Equipment" [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-7) Set C-y replace-item ?1 C-y (item ?1 r_y * item ?1 C-7) Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-7)

Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-7)  Set C-m replace-item ?1 C-m (item ?1 r_m * item ?1 C-7)  set K-plot K-7 set C-plot C-7 ]

if Indicator = "Internet"  [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-8)  Set C-y replace-item ?1 C-y (item ?1 r_y * item ?1 C-8)  Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-8)

Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-8)  Set C-m replace-item ?1 C-m (item ?1 r_m * item ?1 C-8)  set K-plot K-8 set C-plot C-8 ]

if Indicator = "Vehicles"  [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-9)  Set C-y replace-item ?1 C-y (item ?1 r_y * item ?1 C-9)  Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-9)

Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-9)  Set C-m replace-item ?1 C-m (item ?1 r_m * item ?1 C-9)  set K-plot K-9 set C-plot C-9 ]

if Indicator = "Health"  [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-10) Set C-y replace-item ?1 C-y (item ?1 r_y * item ?1 C-10)  Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-10)

Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-10) Set C-m replace-item ?1 C-m (item ?1 r_m * item ?1 C-10)  set K-plot K-10 set C-plot C-10 ]

if Indicator = "Nutrition"  [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-11) Set C-y replace-item ?1 C-y (item ?1 r_y * item ?1 C-11)  Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-11)

Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-11) Set C-m replace-item ?1 C-m (item ?1 r_m * item ?1 C-11)  set K-plot K-11 set C-plot C-11 ]

if Indicator = "Skills"  [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-12) Set C-y replace-item ?1 C-y (item ?1 r_y * item ?1 C-12)  Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-12)

Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-12) Set C-m replace-item ?1 C-m (item ?1 r_m * item ?1 C-12)  set K-plot K-12 set C-plot C-12 ]

if Indicator = "Celebration"  [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-13) Set C-y replace-item ?1 C-y (item ?1 r_y * item ?1 C-13)  Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-13)

Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-13) Set C-m replace-item ?1 C-m (item ?1 r_m * item ?1 C-13)  set K-plot K-13 set C-plot C-13 ]

if Indicator = "Beliefs"  [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-14) Set C-y replace-item ?1 C-y (item ?1 r_y * item ?1 C-14)  Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-14)

Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-14) Set C-m replace-item ?1 C-m (item ?1 r_m * item ?1 C-14)  set K-plot K-14 set C-plot C-14 ]

if Indicator = "Traditions"  [ Set C-x replace-item ?1 C-x (item ?1 r_x * item ?1 C-15) Set C-y replace-item ?1 C-y (item ?1 r_y * item ?1 C-15)  Set C-z replace-item ?1 C-z (item ?1 r_z * item ?1 C-15)

Set C-l replace-item ?1 C-l (item ?1 r_l * item ?1 C-15) Set C-m replace-item ?1 C-m (item ?1 r_m * item ?1 C-15)  set K-plot K-15 set C-plot C-15 ]

```
 ]]
 foreach ESS-type-list [?1 ->
   ask LS [
  ifelse item ?1 S-ESS < 1   [set S-ESS   replace-item ?1 S-ESS   (0)]
    [set S-ESS       replace-item ?1 S-ESS       (item ?1 S-ESS + item ?1 growth + item ?1 tot_harv)]
  set growth   replace-item ?1 growth   ((g2 * item ?1 S-ESS * (1 - (item ?1 S-ESS / L))) - (n * item ?1 S-ESS))

   if Interaction-scenario = "cooperative-optimizing" [

    set f_sum replace-item ?1 f_sum  (item ?1 f_sum + (gama * (item ?1 f_targ - item ?1 f_sum)))
      set f_targ replace-item ?1 f_targ ( n - ( g2 * (1 - (item ?1 S-ESS / L))))
   ]
    if item ?1 S-ESS > 100 [set S-ESS   replace-item ?1 S-ESS   (100)]
    ifelse item ?1 S-ESS < 1 [set G-ESS   replace-item ?1 G-ESS   (0) ]
    [set G-ESS   replace-item ?1 G-ESS   (item ?1 S-ESS / item ?1 tmp  )]
    if item ?1 G-ESS >= 10  [set G-ESS   replace-item ?1 G-ESS   (0) ]
    if item ?1 G-ESS <= 0.01 [set G-ESS   replace-item ?1 G-ESS   (0) ]
if ESS = "yield"          [ set tmp2 precision item 0 G-ESS 2 set pcolor scale-color cyan    item 0 G-ESS 2 0.1]
if ESS = "erosion_control" [ set tmp2 precision item 1 G-ESS 2 set pcolor scale-color cyan    item 1 G-ESS 2 0.1]
if ESS = "carbon_seq"     [ set tmp2 precision item 2 G-ESS 2 set pcolor scale-color cyan    item 2 G-ESS 2 0.1]
if ESS = "water"          [ set tmp2 precision item 3 G-ESS 2 set pcolor scale-color cyan    item 3 G-ESS 2 0.1]
if ESS = "biodiversity"    [ set tmp2 precision item 4 G-ESS 2 set pcolor scale-color cyan    item 4 G-ESS 2 0.1]

 ]]

calc_intermediate
calc_derivative
```

```
foreach ESS-type-list [?1 ->
   ask LS [
  set tot_harv replace-item 0 tot_harv (sum harv_x)
  set tot_harv replace-item 1 tot_harv (sum harv_y)
  set tot_harv replace-item 2 tot_harv (sum harv_z)
  set tot_harv replace-item 3 tot_harv (sum harv_l)
  set tot_harv replace-item 4 tot_harv (sum harv_m)
  set Price replace-item ?1 Price (item ?1 a - (item ?1 b2 * item ?1 tot_harv))

   set P-fin replace-item ?1 P-fin (((item 0 a-share  + item 1 a-share  + item 2 a-share ) * item ?1 a) -
(((item ?1 b2-1  + item ?1 b2-2  + item ?1 b2-3 ) / 3) * item ?1 tot_harv))
   set P-soc replace-item ?1 P-soc (((item 3 a-share  + item 4 a-share  + item 5 a-share ) * item ?1 a) -
(((item ?1 b2-4  + item ?1 b2-5  + item ?1 b2-6 ) / 3) * item ?1 tot_harv))
   set P-phy replace-item ?1 P-phy (((item 6 a-share  + item 7 a-share  + item 8 a-share ) * item ?1 a) -
(((item ?1 b2-7  + item ?1 b2-8  + item ?1 b2-9 ) / 3) * item ?1 tot_harv))
   set P-hum replace-item ?1 P-hum (((item 9 a-share  + item 10 a-share + item 11 a-share) * item ?1
a) - (((item ?1 b2-10 + item ?1 b2-11 + item ?1 b2-12) / 3) * item ?1 tot_harv))
   set P-cul replace-item ?1 P-cul (((item 12 a-share + item 13 a-share + item 14 a-share) * item ?1 a) -
(((item ?1 b2-13 + item ?1 b2-4  + item ?1 b2-15) / 3) * item ?1 tot_harv))

   set P-1  replace-item ?1 P-1  ((item 0 a-share * item ?1 a) - (item ?1 b2-1 * item ?1 tot_harv))
   set P-2  replace-item ?1 P-2  ((item 1 a-share * item ?1 a) - (item ?1 b2-2 * item ?1 tot_harv))
   set P-3  replace-item ?1 P-3  ((item 2 a-share * item ?1 a) - (item ?1 b2-3 * item ?1 tot_harv))
   set P-4  replace-item ?1 P-4  ((item 3 a-share * item ?1 a) - (item ?1 b2-4 * item ?1 tot_harv))
   set P-5  replace-item ?1 P-5  ((item 4 a-share * item ?1 a) - (item ?1 b2-5 * item ?1 tot_harv))
   set P-6  replace-item ?1 P-6  ((item 5 a-share * item ?1 a) - (item ?1 b2-6 * item ?1 tot_harv))
   set P-7  replace-item ?1 P-7  ((item 6 a-share * item ?1 a) - (item ?1 b2-7 * item ?1 tot_harv))
   set P-8  replace-item ?1 P-8  ((item 7 a-share * item ?1 a) - (item ?1 b2-8 * item ?1 tot_harv))
   set P-9  replace-item ?1 P-9  ((item 8 a-share * item ?1 a) - (item ?1 b2-9 * item ?1 tot_harv))
   set P-10 replace-item ?1 P-10 ((item 9 a-share * item ?1 a) - (item ?1 b2-10 * item ?1 tot_harv))
   set P-11 replace-item ?1 P-11 ((item 10 a-share * item ?1 a) - (item ?1 b2-11 * item ?1 tot_harv))
   set P-12 replace-item ?1 P-12 ((item 11 a-share * item ?1 a) - (item ?1 b2-12 * item ?1 tot_harv))
```

```
 set P-13 replace-item ?1 P-13 ((item 12 a-share * item ?1 a) - (item ?1 b2-13 * item ?1 tot_harv))
 set P-14 replace-item ?1 P-14 ((item 13 a-share * item ?1 a) - (item ?1 b2-14 * item ?1 tot_harv))
 set P-15 replace-item ?1 P-15 ((item 14 a-share * item ?1 a) - (item ?1 b2-15 * item ?1 tot_harv))

set Price-plot Price

if capital   = "Financial capital"  [ set Price-plot P-fin]
if capital   = "Social capital"     [ set Price-plot P-soc]
if capital   = "Physical capital"   [ set Price-plot P-phy]
if capital   = "Human capital"      [ set Price-plot P-hum]
if capital   = "Cultural capital"   [ set Price-plot P-cul]

 if Indicator = "Income"        [ set Price-plot P-1 ]
 if Indicator = "Expenditure"   [ set Price-plot P-2 ]
 if Indicator = "Savings"       [ set Price-plot P-3 ]
 if Indicator = "Soc_insurance" [ set Price-plot P-4 ]
 if Indicator = "H_insurance"   [ set Price-plot P-5 ]
 if Indicator = "Agr_insurance" [ set Price-plot P-6 ]
 if Indicator = "Equipment"     [ set Price-plot P-7 ]
 if Indicator = "Internet"      [ set Price-plot P-8 ]
 if Indicator = "Vehicles"      [ set Price-plot P-9 ]
 if Indicator = "Health"        [ set Price-plot P-10 ]
 if Indicator = "Nutrition"     [ set Price-plot P-11 ]
 if Indicator = "Skills"        [ set Price-plot P-12 ]
 if Indicator = "Celebration"   [ set Price-plot P-13 ]
 if Indicator = "Beliefs"       [ set Price-plot P-14 ]
 if Indicator = "Traditions"    [ set Price-plot P-15 ]

]]
foreach actors [?1 ->
  ask LS [
```

```
   if Interaction-scenario = "Competitive-optimizing" [


  set C_targ replace-item ?1 C_targ ( ((item ?1 u - (item ?1 sum_w_c - item ?1 w ))) * item ?1 C / (2 *
item ?1 w * 1000))


  set r_x_targ replace-item ?1 r_x_targ (1 / item ?1 c_x ^ 2 * (item ?1 u_x - item ?1 sum_w_c_x - item
?1 w_x ) / (2 * item ?1 C * item 0 b2 * 10000000) )

  set r_y_targ replace-item ?1 r_y_targ (1 / item ?1 c_y ^ 2 * (item ?1 u_y - item ?1 sum_w_c_y - item
?1 w_y ) / (2 * item ?1 C * item 1 b2 * 10000000) )

  set r_z_targ replace-item ?1 r_z_targ (1 / item ?1 c_z ^ 2 * (item ?1 u_z - item ?1 sum_w_c_z - item
?1 w_z ) / (2 * item ?1 C * item 2 b2 * 10000000) )

  set r_l_targ replace-item ?1 r_l_targ (1 / item ?1 c_l ^ 2 * (item ?1 u_l - item ?1 sum_w_c_l - item ?1
w_l ) / (2 * item ?1 C * item 3 b2 * 10000000) )

  set r_m_targ replace-item ?1 r_m_targ (1 / item ?1 c_m ^ 2 * (item ?1 u_m - item ?1 sum_w_c_m -
item ?1 w_m ) / (2 * item ?1 C * item 4 b2 * 10000000) )

  ]]]

calc_value


tick


end

;;#############################################################################

to display-ess

   if ESS = "yield"          [ set tmp2 precision item 0 tmp 2 set pcolor scale-color red    item 0 tmp 120 0
]

   if ESS = "erosion_control" [ set tmp2 precision item 1 tmp 2 set pcolor scale-color lime   item 1 tmp
120 0 ]

   if ESS = "carbon_seq"      [ set tmp2 precision item 2 tmp 2 set pcolor scale-color yellow item 2 tmp
120 0 ]

   if ESS = "water"          [ set tmp2 precision item 3 tmp 2 set pcolor scale-color blue   item 3 tmp 120
0 ]

   if ESS = "biodiversity"    [ set tmp2 precision item 4 tmp 2 set pcolor scale-color orange item 4 tmp
120 0 ]

end


;;#############################################################################
```

```
to invite-client
 listen-clients
 every 0.1
[
 display
]
end


to start-up
 hubnet-reset
end

to listen-clients
 ;; as long as there are more messages from the clients
 ;; keep processing them.
 while [ hubnet-message-waiting? ]
 [
  ;; get the first message in the queue
  hubnet-fetch-message
  ifelse hubnet-enter-message?
  [ create-new-farmer ]
  [
   ifelse hubnet-exit-message?
   [ ask farmers with [user-id = hubnet-message-source]
    [ die ] ]
      [ run-command hubnet-message-tag
    ask farmers with [user-id = hubnet-message-source]
      [move      hubnet-message-tag]
               ;interact  hubnet-message-tag]
  ]
 ]
```

```
  ]


  ask farmers with [user-id = hubnet-message-source] [
   hubnet-send hubnet-message-source "Value" [tmp2] of patch-here]

end

to create-new-farmer
  create-farmers 1

  [
    set shape "person farmer"
    set size 4
    set user-id hubnet-message-source
    set label user-id
    set label-color 115

  ]

end
to run-command [command] ; run observer commands
 ; if command = "load-map"     [load-map    stop]
  if command = "show-supply"  [show-supply stop]
  if command = "show-demand"  [show-demand stop]
  if command = "show-gap"     [show-gap    stop]
  if command = "Interact"     [set Interact hubnet-message stop]
  if command = "show-map"     [show-map stop]
  if command = "ESS"          [set ESS       hubnet-message stop]
  if command = "Actor"        [set Actor     hubnet-message stop]
  if command = "Management"   [set Land-management hubnet-message stop]
  if command = "Demand"       [set Demand    hubnet-message stop]
```

```
 if command = "Map-type"     [set Map-type   hubnet-message stop]

 if command = "View"        [


 ask patches with [ pxcor = (round item 0 hubnet-message) and

                 pycor = (round item 1 hubnet-message) ]

 [

   foreach ESS-type-list[

  if ESS = "yield"          [ set D-N replace-item 0 D-N Demand set pcolor scale-color red    item 0 D-N
120 0 ]

  if ESS = "erosion_control" [ set D-N replace-item 1 D-N Demand set pcolor scale-color lime   item 1
D-N 120 0 ]

  if ESS = "carbon_seq"      [ set D-N replace-item 2 D-N Demand set pcolor scale-color yellow item 2
D-N 120 0 ]

  if ESS = "water"          [ set D-N replace-item 3 D-N Demand set pcolor scale-color blue   item 3 D-N
120 0 ]

  if ESS = "biodiversity"    [ set D-N replace-item 4 D-N Demand set pcolor scale-color orange item 4
D-N 120 0 ]

 ]]]

end


to move [move-farmer]  ; run turtles command

 if move-farmer = "up"

 [ execute-move 0 stop]

 if move-farmer = "down"

 [ execute-move 180 stop]

 if move-farmer = "right"

 [ execute-move 90 stop]

 if move-farmer = "left"

 [ execute-move 270 stop]


end


to execute-move [new-heading]
```

```
 set heading new-heading
 fd 1
 chart-3
 chart-4
 if Interact = "Add"
 [add]
 if Interact = "Erase"
 [erase  hubnet-broadcast "Demand" 0 stop]

end

to Add     ; run patches command

 ask patch-here
 [
    foreach ESS-type-list[
   if ESS = "yield"          [ set D-N replace-item 0 D-N Demand set pcolor scale-color red    item 0 D-N 120 0 ]
   if ESS = "erosion_control" [ set D-N replace-item 1 D-N Demand set pcolor scale-color lime   item 1 D-N 120 0 ]
   if ESS = "carbon_seq"      [ set D-N replace-item 2 D-N Demand set pcolor scale-color yellow item 2 D-N 120 0 ]
   if ESS = "water"          [ set D-N replace-item 3 D-N Demand set pcolor scale-color blue   item 3 D-N 120 0 ]
   if ESS = "biodiversity"    [ set D-N replace-item 4 D-N Demand set pcolor scale-color orange item 4 D-N 120 0 ]
 ]];]
end

to erase     ; run patches command

 ask patch-here
 [
```

```
    foreach ESS-type-list[
   if ESS = "yield"          [ set D-N replace-item 0 D-N 0 set pcolor 9 ]
   if ESS = "erosion_control" [ set D-N replace-item 1 D-N 0 set pcolor 9 ]
   if ESS = "carbon_seq"      [ set D-N replace-item 2 D-N 0 set pcolor 9 ]
   if ESS = "water"          [ set D-N replace-item 3 D-N 0 set pcolor 9 ]
   if ESS = "biodiversity"    [ set D-N replace-item 4 D-N 0 set pcolor 9 ]
 ]];]
end

to chart-3

  set-current-plot "Supply Analysis"
   plot-pen-reset
   set-plot-pen-color red
   set-plot-x-range 0 5
   set-plot-pen-interval 1
   plot (item 0 S-ESS)
   set-plot-pen-color green
   plot (item 1 S-ESS)
   set-plot-pen-color yellow
   plot (item 2 S-ESS)
   set-plot-pen-color blue
   plot (item 3 S-ESS)
   set-plot-pen-color orange
   plot (item 4 S-ESS)
   set-plot-pen-color red
end

to chart-4
   set-current-plot "Supply-Demand Gap"
   plot-pen-reset
   set-plot-pen-color green
```

```
  set-plot-x-range 0 2

  set-plot-pen-interval 1

  if ESS = "yield"         [set-plot-pen-color green plot (item 0 S-ESS) set-plot-pen-color red plot (item 0 tmp) set-plot-pen-color green]

  if ESS = "erosion_control"[set-plot-pen-color green plot (item 1 S-ESS) set-plot-pen-color red plot (item 1 tmp) set-plot-pen-color green]

  if ESS = "carbon_seq"    [set-plot-pen-color green plot (item 2 S-ESS) set-plot-pen-color red plot (item 2 tmp) set-plot-pen-color green]

  if ESS = "water"         [set-plot-pen-color green plot (item 3 S-ESS) set-plot-pen-color red plot (item 3 tmp) set-plot-pen-color green]

  if ESS = "biodiversity"  [set-plot-pen-color green plot (item 4 S-ESS) set-plot-pen-color red plot (item 4 tmp) set-plot-pen-color green]

  set-plot-pen-color green

end
```