```
--[=[ Start of LUA multiline comment at level 1, one '=' in --[=[ ;-)

   radius functions and global variables defined below:
   radius_vers                      tested: 245, 256, 260
   radius_city_sizes                volatile array of biggest city sizes
   radius_info()                    console + chat window printf output
   radius_magic()                   restore city size table after reload
   radius_tutorial_msg()            msg 1..7 (turn started, city growth)
   radius_turn_started()            welcome message for turn 0 etc.
   radius_city_growth()             message for size 2,5,8,11,14,17
   radius_partisans()               inspire partisans if enabled
   radius_city_lost()               radius + number of partisans
   radius_city_transferred()        2.6.x: calls radius_city_lost()
   radius_city_destroyed()          make ruins + radius_partisans()
   radius_set_label()               label terrain x surrounded by x
   radius_map_generated()           check topology (WRAPX|WRAPY|...)
   radius_hut_type()                find unit by role for hut tile
   radius_hut_gold()                get gold
   radius_hut_barb()                barbarians or abandoned village
   radius_hut_city()                get nomads      or gold
   radius_hut_merc()                get mercenaries or gold
   radius_hut_tech()                get technology  or gold
   radius_hut_enter()               hut chances 30+30+20+10+10
--]=]
-- radius startup -----------------------------------------------------

radius_vers       = 245              -- check 2.6.x incompatibilities
radius_city_sizes = {}               -- max. seen city size per player

function radius_info( msg, ... )     -- command echo green #006400
   msg = "radius: " .. msg
   log.verbose( msg, ... )           -- verbose log output (-d 3)
   print( string.format( msg, ... )) -- stdout (standalone server)
   msg = "[c fg=\"#006400\"]" .. msg .. "[/c]"
   notify.event( nil, nil, E.CHAT_MSG, msg, ... )
end

radius_info( "%s script.lua for Freeciv %d", _VERSION, radius_vers )

-- magic called after map generated or game reloaded in all handlers ---

function radius_magic( )
   local init = radius_city_sizes[ 0 ]

   if not init then                  -- nil is FALSE, 0 is TRUE
      init = 0
      radius_city_sizes[ 0 ] = 0     -- tag city sizes as initialized

      if radius_vers < 260 then      -- disable some default handlers
         signal.remove( "hut_enter", "default_hut_enter_callback" )
         signal.remove( "city_lost", "default_make_partisans_callback" )
      else
         signal.remove( "hut_enter", "_deflua_hut_enter_callback" )
         signal.remove( "city_transferred",
                                "_deflua_make_partisans_callback" )
      end

for p in players_iterate() do        -- restore max. human city sizes
   local b = 0                       -- biggest city size 0 for p.id

   for pc in p:cities_iterate()  do
      if b < pc.size then
         b = pc.size                 -- update biggest size for p.id
      end
   end

   radius_city_sizes[ p.id ] = b
   if init < b then
      init = b                       -- update biggest seen city size
   end
end

if init == 0 then                    -- found no city, assume start
   init = fc_version()
   init = string.sub( init, 17, 17 ) .. string.sub( init, 19, 19 )
   init = tonumber( init ) * 10  -- version x.y.z to number xy0

   if radius_vers < init or init + 9 < radius_vers then
      log.fatal( "radius: expected version %d, got %d",
         radius_vers, init )         -- fatal / assert have no effect
   end

   init = server.setting.get( "topology" )
```

```
         if not string.find( init, "WRAPX|WRAPY" ) then
            radius_info( _("expected WRAPX|WRAPY|..., got %s"), init )
         end                           -- unsuited topology warning
      else                             -- info if a game was reloaded
         radius_info( _("game loaded, biggest city size %d."), init )
      end
   end                                 -- no result interpreted as NIL
end

-- radius tutorial -----------------------------------------------------

function radius_tutorial_msg( n )
   local msgs = {                                        -- 1 @1st turn:
      _("Your settlers and other units might get killed when entering\
 a hut, but your \n workers and leader have nothing to fear. \n Your\
 first research project Masonry will allow you to build a Palace and\
 \n get a Capital (not necessarily in your first city) as soon as\
 possible."),                                          -- 2 @size 2:
      _("The radius of workable city tiles will be increased at size\
 4, 9, 16, and so \n on.  Build a Temple and a Marketplace when you\
 have researched Ceremonial \n Burial and Currency."),   -- 3 @size 5:
      _("At city size 7 you should have a Cathedral or a Colosseum\
 (more expensive) \n to make your citizens happy when you switch your\
 Government to a Republic."),                          -- 4 @size 8:
      _("You need an Aqueduct (requires Construction) for a city\
 growing over size 8."),                               -- 5 @size 11:
      _("You need a Sewer System (requires Sanitation) for a city\
 growing over size 12."),                              -- 6 @size 14:
      _("Big cities pollute the environment.  Plan your research to\
 build Mass Transit and a Recycling Center."),         -- 7 @size 17:
      _("If you ever get a city with size 100 you'll win the game ;-)") }

   return msgs[ n ]
end

function radius_turn_started( turn, year )
   local magic = radius_magic( )
   local ends  = tonumber( server.setting.get( "endturn" ))

   if       turn < 1 then
      notify.all( radius_tutorial_msg( 1 ))
   elseif   turn == 252 then           -- (200 + 52) * 20 until 1040 CE
      radius_info( _("10 years per turn until 1770 CE."))
   elseif   turn == 325 then           -- 325 == 252 + 73 until 1770 CE
      radius_info( _("1 year per turn after 1770 CE."))
   elseif   turn + 33 == ends then     -- helpful early endturn warning
      notify.all( "endturn = %d", ends )
   end                                 -- endturn is a literal, no i18n
end

function radius_city_growth( city, size )
   local magic = radius_magic( )
   local owner = city.owner
   local osize = radius_city_sizes[ owner.id ]

   if osize == nil   then
      osize = 0
   end
   if osize < size   then              -- note new biggest city size
      radius_city_sizes[ owner.id ] = size

      for n = 2, 7   do
         if osize < 3 * n - 4 and 3 * n - 4 <= size   then
            notify.event( owner, city.tile,
                          E.SCRIPT,    -- E.BEGINNER_HELP after 2.6.x
                          radius_tutorial_msg( n ))
            return
         end
      end
      if size > 99   then
         owner:victory()               -- shows "scenario victory to..."
      end
   end
end

signal.connect( "turn_started", "radius_turn_started" )
signal.connect( "city_growth",  "radius_city_growth"  )

-- 1..18 partisans -----------------------------------------------------

function radius_partisans( city, loser, winner, troops, radius )
   if city:inspire_partisans( loser ) > 0 then
```

```
      city.tile:place_partisans( loser, troops, radius )

      notify.event(  loser, city.tile, E.CITY_LOST,
         _("The loss of %s has inspired partisans!"), city.name )
      notify.event( winner, city.tile, E.UNIT_WIN_ATT,
         _("The loss of %s has inspired partisans!"), city.name )
   end
end

function radius_city_lost( city, loser, winner )
   local magic = radius_magic( )
   local radius = city:map_sq_radius()

   for s = 9, 1, -1  do                -- testing a basic Lua "fornum"
      if s * s <= city.size   then
         return radius_partisans( city, loser, winner, 2 * s, radius )
      end
   end
end

function radius_city_transferred( city, loser, winner, reason )
   if reason == "conquest" then
      radius_city_lost( city, loser, winner )
   end                                 -- call also checks magic(), this
end                                    -- kludge isn't consequent, sorry

if radius_vers < 260                   -- 2.6.x reason "conquest"
   then  signal.connect( "city_lost",        "radius_city_lost" )
   else  signal.connect( "city_transferred", "radius_city_transferred" )
end

-- make Ruins at the location of a destroyed city ----------------------

function radius_city_destroyed( city, loser, destroyer )
   local magic = radius_magic( )

   if radius_vers < 260                -- Ruins have no owner (2.5: nil)
      then  city.tile:create_base(  "Ruins", nil )
      else  city.tile:create_extra( "Ruins" )
   end
   if destroyer and loser ~= destroyer then
      radius_partisans( city, loser, destroyer, 1, 1 )
   end                                 -- one partisan if enemy action
end

signal.connect( "city_destroyed", "radius_city_destroyed" )

-- try to create a map label -------------------------------------------
-- vague idea for 2.6: create extra huts at any labelled land tiles

function radius_set_label( name, label, want )
   local same  = false                 -- test all adjacent tiles
   local last  = 0                     -- count candidates

   for tile in whole_map_iterate() do  -- check all tiles once
      if tile.terrain:rule_name() == name then
         same = true                   -- check adjacent tiles
         for near in tile:square_iterate( 1 )   do
            same = same and near.terrain:rule_name() == name
         end
         if same  then
            last = last + 1            -- count candidate
            if last == want   then     -- label selected tile
               tile:set_label( label )
               return
            end
         end
      end
   end
   if last > 0 then                    -- select candidate
      radius_set_label( name, label, random( 1, last ))
      radius_info( "set %s label", label )
   else                                -- bad luck (or land terrain)
      radius_info( "found no %s",  label )
   end
end

function radius_map_generated( )
   local magic = radius_magic( )

   radius_set_label( "Deep Ocean" , _("Deep Trench")  , 0 )
-- radius_set_label( "Desert"     , _("Scorched Spot"), 0 )
   radius_set_label( "Glacier"    , _("Frozen Lake")  , 0 )
```

```
   radius_set_label( "Mountains"  , _("Highest Peak") , 0 )
end

if radius_vers < 250                   -- 2.4.x had no map_generated
   then  radius_info( "experimental Freeciv 2.5.x to 2.4.x backport" )
   else  signal.connect( "map_generated", "radius_map_generated" )
end

-- randomly choose a hut event ---------------------------------------
--[=[     data/default/default.lua vs. radius/script.lua
   Gold 1*25 + 3*50 + 1*100: 5/12, now 30%, Gold 33/66/99
   (average 275/12 = 22.91)                 (average 198/10 = 19.8)
   Technology    or Gold 25: 3/12, now 30%, Gold 33/66/99 if impossible
   Mercenaries   or Gold 25: 2/12, now 20%, Gold       11 if impossible
   City, Nomades or Gold 25: 1/12, now 10%, always Nomades or bug bounty
   Barbarians              : 1/12, now 10%, protected Workers and Leader
--]=]

function radius_hut_type( owner, tile, role )
   local type = find.role_unit_type( role, owner )

   if type and type:can_exist_at_tile( tile )
      then  return type
      else  return nil
   end
end

function radius_hut_gold( owner, tile, gold )
   owner:change_gold( gold )           -- no i18n plural, at least 10
   notify.event( owner, tile, E.HUT_GOLD, _("You found %d gold."), gold )
end

function radius_hut_barb( owner, tile, type )
   if       server.setting.get( "barbarians" ) == "DISABLED"   or
            tile:city_exists_within_max_city_map( true )       or
            type:has_flag( "Airbase" )                         or
            type:has_flag( "Gameloss" )                        then
      notify.event( owner, tile, E.HUT_BARB_CITY_NEAR,
         _("An abandoned village is here."))
   elseif   tile:unleash_barbarians()  then
      notify.event( owner, tile, E.HUT_BARB,
         _("You have unleashed a horde of barbarians!")  )
   else
      notify.event( owner, tile, E.HUT_BARB_KILLED,
         _("Your %s has been killed by barbarians!"),
         type:name_translation()                          )
   end
end

function radius_hut_city( owner, tile, home )
   if radius_vers < 260
      then  type = radius_hut_type( owner, tile, "Cities" )
      else  type = radius_hut_type( owner, tile, "CitiesStartUnit" )
   end
   if type  then
      owner:create_unit( tile, type, 0, home, -1 )
      notify.event( owner, tile, E.HUT_SETTLER,
         _("Friendly nomads are impressed by you, and join you."))
   else
      radius_hut_gold( owner, tile, 77 )
      radius_info( "radius_hut_city(): tile unsuited for nomads" )
   end                                 -- bug bounty is rather poor
end

function radius_hut_merc( owner, tile, home )
   local type = radius_hut_type( owner, tile, "HutTech" )

   if not type then
      type = radius_hut_type( nil, tile, "Hut" )
   end
   if type  then
      owner:create_unit( tile, type, 0, home, -1 )
      notify.event( owner, tile, E.HUT_MERC,
         _("A band of friendly mercenaries joins your cause."))
   else
      radius_hut_gold( owner, tile, 11 )
   end
end

function radius_hut_tech( owner, tile, gold )
   local who   = owner.nation:plural_translation()
   local what  = nil
   local tech  = nil
```

```lua
   if radius_vers < 260
      then  tech = owner:give_technology( nil, "hut" )
      else  tech = owner:give_tech( nil, -1, false, "hut" )
   end
   if tech  then
      what = tech:name_translation()   -- stick to default.lua strings:

      notify.event( owner, tile, E.HUT_TECH,
         _("You found %s in ancient scrolls of wisdom."), what )

      if radius_vers < 260 then
         notify.embassies( owner, tile, E.HUT_TECH,
            _("The %s have acquired %s from ancient scrolls of wisdom."),
            who, what )
      else
         notify.research( owner, false, E.TECH_GAIN,
            _("The %s found %s in ancient scrolls of wisdom for you."),
            who, what )                -- added "The " to default string
         notify.research_embassies( owner, E.TECH_EMBASSY,
            _("The %s have acquired %s from ancient scrolls of wisdom."),
            who, what )
      end
   else
      radius_hut_gold( owner, tile, gold )
   end
end
```

```lua
function radius_hut_enter( unit )
   local magic = radius_magic( )
   local owner = unit.owner
   local tile  = unit.tile
   local luck  = random( 0, 9 )

   if       luck <= 2  then            -- 30%           Gold 33/66/99:
      radius_hut_gold( owner, tile, 33 * ( luck + 1 ))
   elseif   luck <= 5  then            -- 30% Tech.  or Gold 33/66/99:
      radius_hut_tech( owner, tile, 33 * ( luck - 2 ))
   elseif   luck <= 7  then            -- 20% Mercenaries  or Gold 11:
      radius_hut_merc( owner, tile, unit:get_homecity())
   elseif   luck == 8  then            -- 10% Nomades      or Gold 77:
      radius_hut_city( owner, tile, unit:get_homecity())
   else                                -- 10% Barbarians or abandoned:
      radius_hut_barb( owner, tile, unit.utype )
   end                                 --  0% City (E.HUT_CITY unused)
end

signal.connect( "hut_enter", "radius_hut_enter" )
```