Electro

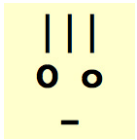
305900
2007년 가을학기
10/18/2007
박경신

Installing Electro

- www.evl.uic.edu/rlk/electro
- Download the latest executable (for Windows 2K/XP) electro-1042-w32.zip
- Download the source code from subversion repository https://s.snth.net/svn/rlk/electro/trunk
  or check out the source code using a SVN client
  - svn co https://s.snth.net/svn/rlk/electro/trunk electro
- Download the data package if necessary
  - Galaxy data electro-galaxy.zip - The contents of this package must be placed in electro/examples/data.
- Download the examples for beginners
  - Electro intro package electro-intro.zip

Electro

- Electro is an application development environment designed for use on both cluster-driven tiled displays and desktop systems
- Electro is based on the MPI process model and is bound to the Lua programming language.
- Electro supports Linux, Windows 2K/XP, and Mac OS X.



Electro Getting Started

- Command Line Arguments
  - All arguments with the extension .lua are taken to be Lua scripts to be executed in order
  - -f <Lua script> execute the named Lua script
  - -m grab the mouse pointer
  - -a disable audio playback
  - -l <log file> log all console output to the named file
  - -p <port> use the given TCP/IP port
  - -t <key> use the given key value to access 'trackd' tracker
  - -c <key> use the given key value to access 'trackd' controller
  - -h <index> use the given 'trackd' sensor index as head tracker
  - -H <catalog file> <galaxy file> process the given Hipparcos steller catalog into an Electro galaxy file
  - -T <catalog file> <galaxy file> process the given Tycho steller catalog into an Electro galaxy file

Electro 예제

- Fifteen Puzzle - www.evl.uic.edu/rlk/electro/example.html

Fifteen.lua

```
image_file = "venus.jpg“
view_x = 0
view_y = 0
view_w = 0
view_h = 0
camera  = nil
numbers = { }
sprites = { }
curr_i  = 4
curr_j  = 4
time    = 0
random  = false
```

Orthogonal camera

16 sprite objects
numbers[i][j] 4x4 스프라이트 테이블

current location of blank spot

Total amout of time passed
do_timer function is randomizing the puzzle

Fifteen.lua

```
function move_sprite(sprite, i, j)
   E.set_entity_position(sprite, view_x + view_w * (j - 1) / 4 + view_w / 8,
                         view_y + view_h * (4 - i) / 4 + view_h / 8, 0)
end
```

Given row i and column j,
Function move_sprite() positions the tile object sprite at the correct location on screen.

Fifteen.lua

```
function solved()
   local k = 1
   for i = 1, 4 do
      for j = 1, 4 do
         if numbers[i][j] == k then
            k = k + 1
         else
            return false
         end -- end of if
      end -- end of for
   end -- end of for
   return true
end – end of function
```

Function solved() indicates when the puzzle is complete. When solved, the tile numbers will read in order from left to right an top to bottom.

Fifteen.lua

```
function move_piece(di, dj)
    local next_i = curr_i + di
    local next_j = curr_j + dj
    if 1 <= next_i and next_i <= 4 and -- Confirm that the move is valid.
      1 <= next_j and next_j <= 4 then
          temp = numbers[curr_i][curr_j]
          numbers[curr_i][curr_j] = numbers[next_i][next_j] – Swap moving piece with blank
          numbers[next_i][next_j] = temp
          move_sprite(sprites[numbers[curr_i][curr_j]], curr_i, curr_j) -- Set the new locations
          move_sprite(sprites[numbers[next_i][next_j]], next_i, next_j)
          curr_i = next_i
          curr_j = next_j
          E.set_entity_flags(sprites[16], E.entity_flag_hidden, not solved()) -- Set the visibility
          return true
    else
          return false
    end
end
```


Fifteen.lua

```
function move_random()
    while true do
        local d = math.random(-1, 1)
        if math.random(0, 1) == 0 then
            if move_piece(d, 0) then
                return
            end
        else
            if move_piece(0, d) then
                return
            end
        end
    end
end
```


Fifteen.lua

```
function do_timer(dt)
  time = time + dt

  if time > 0.25 then
    move_random()
    time = 0
   return true
  end

  return false
end
```


Fifteen.lua

```
function do_keyboard(k, s)
  if s then
    if k == E.key_left  then move_piece( 0,  1) end  --  if arrow key is pressed
    if k == E.key_right then move_piece( 0, -1) end
    if k == E.key_up    then move_piece( 1,  0) end
    if k == E.key_down  then move_piece(-1,  0) end
    if k == E.key_space then -- If autoplay has been switched
      random = not random
      E.enable_timer(random)
    end
    if k == E.key_F12 then  -- Jump to the demo selector on F12.
      E.nuke()
      E.chdir("..")
      dofile("demo.lua")
    end
  end
  return true
end
```

Fifteen.lua

```
function do_start()
   view_x, view_y, view_w, view_h = E.get_display_union()

   camera = E.create_camera(E.camera_type_orthogonal)
   image  = E.create_image(image_file)
   brush  = E.create_brush()

   E.set_background(0, 0, 0)
   E.set_brush_image(brush, image)
   E.set_brush_flags(brush, E.brush_flag_unlit, true)
```

do_start()

Fifteen.lua

```
   -- Add 16 sprites and scale each to 1/16th the size of the display.
   for i = 1, 16 do
      sprites[i] = E.create_sprite(brush)

      local x0, y0, z0, x1, y1, z1 = E.get_entity_bound(sprites[1])

            W = x1 - x0
            H = y1 - y0

      E.parent_entity(sprites[i], camera)
      E.set_entity_scale(sprites[i], 0.25 * view_w / (x1 - x0),
                                     0.25 * view_h / (y1 - y0), 1.0)
   end
```

do_start()

Fifteen.lua

```
   -- Initialize an array that maps rows and columns onto puzzle pieces.
   numbers[1] = {  1,  2,  3,  4 }
   numbers[2] = {  5,  6,  7,  8 }
   numbers[3] = {  9, 10, 11, 12 }
   numbers[4] = { 13, 14, 15, 16 }
   -- Assign 1/16th of the image to each sprite.
   for i = 1, 4 do
      for j = 1, 4 do
         local x0 = W * (j - 1) / 4
         local x1 = W * (j - 0) / 4
         local y0 = H * (4 - i) / 4
         local y1 = H * (5 - i) / 4
         move_sprite(sprites[numbers[i][j]], i, j)
         E.set_sprite_range(sprites[numbers[i][j]], x0, x1, y0, y1)
      end
   end
```

do_start()

Fifteen.lua

```
   -- Scramble the tiles.
   for i = 1, 500 do
      move_random()
   end

   return true
end

--------------------------------------------------------------------------------

do_start()
```

do_start()

Electro API

- Electro API is encapsulated in a Lua namespace "E".
  - E.set_background(0.0, 0.5, 1.0)
- Entities
- Images
- Brushes
- Sound
- Console
- Configuration
- System
- Miscellaneous

Entity

- Entities
  - The entity base class handles all of the common properties and behaviors of scene graph elements, 3D transformation, scene-hierarchy
  - The class hierarchy is follows:

0. Entity
    1. Object
    2. Sprite
    3. String
    4. Camera
    5. Light
    6. Pivot
    7. Galaxy

Entity

- object = create_object(filename)
  - Load an OBJ format 3D object and return an Electro object entity
- sprite = create_sprite(brush)
  - Create a sprite entity using the given brush
  - Default size of the sprite object is equal to the size of the brush image
- string = create_string(text)
  - Create a string object using the current typeface and the given text
- camera = create_camera(type)
  - Create a camera of the given type
    - camera_type_orthogonal
    - camera_type_perspective

Entity

- light = create_light(type)
  - Create a light source of the given type
    - light_type_positional
    - light_type_directional
- pivot = create_pivot()
  - Create a pivot entity
- galaxy = create_galaxy(filename, brush)
  - Load a galaxy definition from the named file and return a galaxy entity
  - A galaxy entity is a static 3D collection of particles, organized into a BSP structure for efficient rendering
- entity = create_clone(entity)
  - Duplicate the given entity

Entity Hierarchy Management

- parent_entity(entity, parent)
  - Include entity as a child of parent
- delete_entity(entity)
  - Remove entity from the scene hierarchy
- parent = get_entity_parent(entity)
  - Return the parent of entity or nil
- child = get_entity_child(entity, n)
  - Return the nth child of entity or nil

Entity Transformation

- set_entity_position(entity, x, y, z)
- set_entity_scale(entity, x, y, z)
  - Position & scale is given in the entity's parent coordinates
- set_entity_rotation(entity, x, y, z)
  - The angles of rotation in degrees about the x, y, z axes
- set_entity_bound(entity, minx, miny, minz, maxx, maxy, maxz)
  - Axis-aligned bounding box of the entity
- set_entity_tracking(entity, sensor, mode)
  - tracking_mode_local
  - tracking_mode_world
- move_entity(entity, dx, dy, dz)
- turn_entity(entity, ax, ay, az)
  - Rotate about the x, y, z axes

Entity Rendering Properties & Query

- set_entity_alpha(entity,alpha)
  - Specify the transparency of the entity
- x,y,z = get_entity_position(entity)
  - Return the position (x,y,z) of the entity in local coordinates
- x,y,z = get_entity_x_vector(entity)
- x,y,z = get_entity_y_vector(entity)
- x,y,z = get_entity_z_vector(entity)
  - Return the vector pointing to the right/up/back from the entity
- minx,miny,minz,maxx,maxy,maxz = get_entity_bound(entity)
  - Return the minimun and maximum x, y, z values of the entity
- a = get_entity_alpha(entity)

Entity Flags

- set_entity_flags(entity,flags,value)
  - Set or clear flags on the given entity
  - entity_flag_wireframe – render wireframe
  - entity_flag_hidden – hidden (i.e. no rendering)
  - entity_flag_billboard – drawing entity facing the camera
  - entity_flag_bounded – render only when its bounding box inside view frustum
  - entity_flag_track_pos – update with its position by trackd
  - entity_flag_track_rot
  - entity_flag_left_eye – for stereo rendering
  - entity_flag_right_eye
  - entity_flag_visible_body – visual debugging for physical entity
  - entity_flag_visible_geom
- value = get_entity_flags(entity,flags)
  - Return true if flags are set on the entity

Collision Detection and Response

- Electro uses the Open Dynamics Engine (ODE)
  - A rigid body is composed "geoms", "bodies", and "joints".
  - A geom is solid: sphere, box, cylinder, plane. Geoms define the collection characteristics of an object.
  - A body is a compound solid: a rigid arrangement of geoms. Bodies define the physics charactertices of an object.
  - A joint is a non-rigid connection between bodies. Joints define the behaviors of articulated objects

Collision Detection and Response

- set_entity_body_type(entity, type)
- set_entity_geom_type(entity, type, …)
  - geom_type_none
  - geom_type_box, x, y, z
  - geom_type_sphere, r
  - geom_type_capsule, r, l – cylinder with spherical end caps
  - geom_type_ray, px, py, pz, vx, vy, vz
  - geom_type_plane, a, b, c, d
- set_entity_joint_type(entity1, entity2, type)
  - joint_type_ball
  - joint_type_hinge
  - joint_type_slider
  - joint_type_universal
  - joint_type_hinge_2

Collision Detection and Response

- set_entity_body_attr(entity, attr, value)
  - body_attr_gravity – the force of gravisy will act on this body
- set_entity_geom_attr(entity, attr, type)
  - geom_attr_category
  - geom_attr_collider – collider bit uses the category bit to determine if two geoms should be tested for collision
  - geom_attr_response – response bit uses the category bit to determine if two colliding geoms should respond physically in any way
  - geom_attr_callback – callback bit uses the category bit to determine if two colliding geoms should be reported to the application
  - geom_attr_mass
  - geom_attr_bounce – the coefficient of restitution or bounciness of an entity

Collision Detection and Response

- set_entity_geom_attr(entity, attr, type)
  - geom_attr_friction –the coefficient of friction
  - geom_attr_soft_erp – error reduction parameter and constraint force mixing value
  - geom_attr_soft_cfm
- set_entity_joint_attr(entity1, entity2, attr, value)
  - joint_attr_anchor – the position (x,y,z) of a ball, hinge, hinger2, or universal joint
  - joint_attr_axis_1 – the primary axis vector (x,y,z) of a hinge, slider, hinge2, or universal joint
  - joint_attr_axis_2 – the secondary axis vector (x,y,z) of a hinge2 or universal joint
  - joint_attr_lo_stop, joint_attr_hi_stop, joint_attr_lo_stop_2, joint_attr_hi_stop_2 – the minimum and maximum values of the primary and secondary joint attributes

Collision Detection and Response

- set_entity_joint_attr(entity1, entity2, attr, value)
  - joint_attr_velocity, joint_attr_force_max, joint_attr_velocity_2, joint_attr_foce_max_2 – the velocity and force applied by a joint motor
  - joint_attr_bounce, joint_attr_cfm, joint_attr_bounce_2, joint_attr_cfm_2 – the bounciness and CFM of the joint between stops
  - joint_attr_stop_erp, joint_attr_stop_erp_2, joint_attr_stop_cfm, joint_attr_stop_cfm_2 – the ERP and CFM of the joint's lo_stop and hi_stop
  - joint_attr_susp_erp, joint_attr_susp_cfm – the ERP and CFM of a suspension joint

Collision Detection and Response

- value = get_entity_body_attr(entity, attr)
  - body_attr_center – the center of mass of a body
- value = get_entity_geom_attr(entity, attr)
- value = get_entity_joint_attr(entity1, entity2, , attr)
  - joint_attr_value
  - joint_attr_rate_1
  - joint_attr_rate_2
- Direct force application
  - add_entity_force(entity, x, y, z) – apply force to entity with the magnitude and direction of vector (x, y, z)
  - add_entity_torque(entity, x, y, z) – apply torque to entity with the magnitude and axis of vector (x, y, z)

Object

- iM = create_mesh(object,[brush])
  - Add a new mesh to object and return its index
- iV = create_vert(object,[vx, vy, vz, [nx, ny, nz, [tu, tv]]])
  - Add a new vertex to object and return its index
- iF = create_face(object, iM, iV, jV, kV)
  - Add a new triangular face to mesh iM of object defined by the vertices (iV, jV, kV)
- iE = create_edge(object, iM, iV, jV)
  - Add a new mesh to object and return its index
- delete_mesh(object, iM)
- delete_vert(object, iV)
- delete_face(object, iM, iF)
- delete_edge(object, iM, iE)

Object

- set_mesh(object, iM, brush)
  - Set the brush used to render mesh iM of object
- set_vert(object, iV, [vx, vy, vz, [nx, ny, nz, [tu, tv]]])
  - Set the vector values of vertex iV
- set_face(object, iM, iF, iV, jV, kV)
  - Specify the set of vertices defining face iF
- set_edge(object, iM, iE, iV, jV)
  - Specify the set of vertices defining edge iE

Object

- brush = get_mesh(object, iM)
  - Return the brush used by a mesh
- vx, vy, vz, nx, ny, nz, tu, tv = get_vert(object, iV)
  - Return the position, normal, texture coordinate of a vertex
- iV, jV, kV = get_face(object, iM, iF)
  - Return the set of vertex indices that define a face
- iV, jV = get_edge(object, iM, iE)
  - Return the set of vertex indices that define an edge
- n = get_mesh_count(object)
- n = get_vert_count(object)
- n = get_face_count(object, iM)
- n = get_edge_count(object, iM)

Sprite & String

- set_sprite_brush(sprite, brush)
  - Set the brush used to display sprite
- set_sprite_range(sprite, s0, s1, t0, t1)
  - Set the texture coordinate rectangle used by sprite
  - By default a sprite uses its brush's entire image – tex coord (0, 1, 0, 1)
- set_string_text(string, text)
  - Set the text displayed by the given string object
- set_string_fill(string, brush)
  - Set the fill brush of the string entity
- set_string_line(string, brush)
  - Set the outline brush of the string entity

Camera

- x,y,z = get_camera_vector(camera, displayx, displayy)
  - Return the normalized world-space vector corresponding to the display position (displayx, displayy)
- set_camera_stereo(camera, mode, Lx, Ly, Lz, Rx, Ry, Rz)
  - Set the stereo options on the given camera
  - (Lx, Ly, Lz) and (Rx, Ry, Rz) arguments given the offsets from the camera to the user's left and right eyes
  - stereo_mode_none
  - stereo_mode_tile – stereo rendering on tracked Geowall-type display
  - stereo_mode_quad – enables quad_buffered stereo viewing
  - stereo_mode_red_blue – enables red-blue anaglyphic stereo
  - stereo_mode_varrier_11 – enables Varrier stereo viewing using the 1-line-pass/1-scene-pass Varrier alorithm
  - Stereo_mode_varrier_33 – enables Varrier stereo viewing using the 3-line-pass/3-scene-pass Varrier sub-pixel rendering algorithm

Camera

- set_camera_offset(camera, x, y, z)
  - Set the offset of a camera's view
- set_camera_range(camera, near, far)
  - Set the near and far clipping planes for a camera
- set_camera_image(camera, image, left, right, bottom, top)
  - Set an image to be used as off-screen render target for camera
  - The left, right, bottom, right arguments define the projection to be applied

Light, Galaxy

- set_light_color(light, r, g, b)
  - Set the diffuse color of a light source
- set_galaxy_magnitude(galaxy)
  - Set the magnitude multiplier of a galaxy
- iS = get_star_index(galaxy, entity)
  - Return the star "being pointed at" by the given entity
- x,y,z = get_star_position(galaxy, iS)
  - Return the 3D position of the star at index iS of galaxy

Images

- image = create_image(filename)
  - Load a static image from the named PNG or JPEG file
- image = create_image(filename-x, filename+x, filename-y, filename+y, filename-z, filename+z)
  - Load a set of cube map images
- image = create_image(pattern, width, height, bytes, frame0, framen, pulldownn, pulldownd)
  - Create a flip-book animated image
- image = create_image(port) - create a video stream image
- image = create_image(width, height, bytes)
  - Create a blank image using of the given size and depth
- delete_image(image)
- r,g,b,a = get_image_pixel(image, x, y)
- w,h = get_image_size(image)

Brushes

- Brushes describe the appearance of the surface of objects, strings, and sprites
- brush = create_brush()
  - Create a new brush object with the default material properties
- delete_brush(brush)
- set_brush_color(brush, dr, dg, db, da, [sr, sg, sb, sa, [ar, ag, ab, aa, [e]]])
  - Set the material color properties of brush
- set_brush_image(brush, image, [n])
  - Sekect image for use as the texture map of brush

Brushes

- set_brush_flags(brush, flags, value)
  - Set or clear flags on the given brush
  - brush_flag_diffuse, brush_flag_specular, blush_flag_ambient, brush_flag_shin – material color properties
  - brush_flag_unlit – surfaces with an unlit brush are drawn with all lighting disabled
  - brush_flag_transparent – surfaces with a transparent brush are drawn to the color buffer
  - brush_flag_env_map_0, brush_flag_env_map_1, brush_flag_env_map_2, brush_flag_env_map_3 – environment map is dynamically applied to a surfaced based upon the surface normal and view direction
  - brush_flag_sky_map_0, brush_flag_sky_map_1, brush_flag_sky_map_2, brush_flag_sky_map_3 – for sky map

Brushes

- set_brush_flags(brush, flags, value)
  - Set or clear flags on the given brush
  - brush_flag_diffuse, brush_flag_specular, blush_flag_ambient, brush_flag_shin – material color properties
  - brush_flag_unlit – surfaces with an unlit brush are drawn with all lighting disabled
  - brush_flag_transparent – surfaces with a transparent brush are drawn to the color buffer
  - brush_flag_env_map_0, brush_flag_env_map_1, brush_flag_env_map_2, brush_flag_env_map_3 – environment map is dynamically applied to a surfaced based upon the surface normal and view direction
  - brush_flag_sky_map_0, brush_flag_sky_map_1, brush_flag_sky_map_2, brush_flag_sky_map_3 –sky map

Brushes

- set_brush_frag_shader(brush, flags, value)
  - Specify a fragment shader to be applied to brush
- set_brush_vert_shader(brush, file)
  - Specify a vertex shader to be applied to brush
- set_brush_uniform_sampler(brush, name, T)
  - Specify a uniform sampler to be applied to brush
- set_brush_uniform_vector(brush, name, x, [y, [z, [w]]])
  - Specify a uniform vector to be applied to brush
- set_brush_uniform_matrix(brush, name, […])
  - Specify a uniform matrix to be applied to brush

Brushes

- set_brush_frag_prog(brush, file)
  - Specify a fragment program to be applied to brush
- set_brush_vert_prog(brush, file)
  - Specify a vertex program to be applied to brush
- set_brush_frag_param(brush, name, T)
  - Specify a local parameter to be accessible to brush's fragment program
- set_brush_vert_param(brush, name, x, [y, [z, [w]]])
  - Specify a local parameter to be accessible to brush's vertex program
- set_brush_line_width(brush, name, […])
  - Specify the width in pixels of all lines drawn using this brush

Sound

- sound = load_sound(filename)
  - Open a reference to the named Ogg Vorbis file
- free_sound(sound)
- play_sound(sound)
- loop_sound(sound)
- stop_sound(sound)
- set_sound_amplitude(sound, amplitude)
- set_sound_frequency(sound, frequency)
- set_sound_receiver(entity, distance)
  - Specify entity as the receiver
- set_sound_emitter(sound, entity)
  - Associate sound with entity

Console

- print_console(…)
  - Print all string and number arguments to the console
- clear_console()
  - Clear the console to blank
- close_console()
  - Remove the console from the screen
- color_console(r, g, b)
  - Set the console text color to (r, g, b)

Configuration

- host = add_host(name, x, y, w, h)
  - Add a host, returning a host index
- set_host_flags(host, flags, value)
  - Set or clear host flags on host
  - host_flag_full
  - host_flag_stereo
  - host_flag_framed

Tiles

- tile = add_tile(host, x, y, w, h)
  - Add a tile to a host, returning a tile index
- set_tile_position(tile, ox, oy, oz, rx, ry, rz, ux, uy, uz)
  - Set the world-space position of tile
- set_tile_viewport(tile, x, y, w, h)
  - Set the pixel area of tile
- set_tile_mirror(tile, x, y z, d)
  - Specify a world-space plane of reflection to be applied to the view position vector when rendering to tile

Tiles

- set_tile_flags(tile, flags, value)
  - Set or clear tile flags on tile
  - tile_flag_mirror
  - tile_flag_flip_x
  - tile_flag_flip_y
  - tile_flag_left_eye
  - tile_flag_right_eye
- set_tile_linescreen(tile, pitch, angle, thickness, shift, cycle)
- x,y,w,h = get_display_union()
  - Return the total pixel size of the display
- minx,miny,minz,maxx,maxy,maxz = get_display_bound()
  - Return the axis-aligned bounding box of the entire display
- set_tracker_transform(index, m0, m1, … m15)

System API

- exit()
  - Exit Electro cleanly
- chdir(path)
  - Change the current working directory to path
- pushdir(path)
  - Push the current working directory to the directory stack and change to path
- popdir()
  - Pop the previous current working directory from the directory stack and change there

Miscellaneous API

- set_background(Tr, Tg, Tb, [Br, Bg, Bb])
  - Set the background color of the display
  - If (tr, tg, tb) and (br, bg, bb) is specified, the background will be drawn with a top-to-bottom gradient
- set_typeface(filename, [epsilon, outline])
  - Set the typeface to be used for all subsequently created string entities
- enable_timer(enabled)
  - Enable or disable the "do_timer" callback
- state = get_modifier(modifier)
  - Return the current state of a keyboard modifier
  - key_modifier_shift, key_modifier_control, key_modifier_alt
- x,y = get_joystick(number, [axisx, axisy])
  - Return the current value of a pair of axes of joystick number

Callbacks

- do_point(dx, dy)
  - invoked whenever the mouse point moves within the Electro main window
- do_click(b, s)
  - invoked whenever a mouse button is pressed or released
- do_timer(dt)
  - invoked regularly while idling is enabled
- do_frame()
  - invoked immediately before a frame is rendered
- do_keyboard(k, s)
  - invoked whenever a key is pressed or released
- do_joystick(n, b, s)
  - invoked whenever a joystick is pressed or released
- do_contact(entityA, entityB, px, py, pz, nx, ny, nz, d)

Reference

- http://www.evl.uic.edu/rlk/electro/
- http://www.evl.uic.edu/rlk/electro/electro.html
- http://www.evl.uic.edu/rlk/electro/example.html