Object-Oriented Framework For Teaching Introductory Programming

Master Thesis

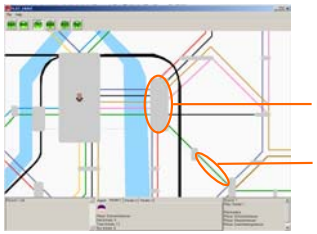
Rolf Bruderer
bruderer@computerscience.ch

Supervising Professor:
Prof. Dr. Bertrand Meyer

Supervising Assistants:
Michela Pedroni, Till G. Bay

TRAFFIC

Library for Modeling Transportation Networks



Previous Visualization:
CANVAS containing **DRAWABLE_OBJECTs**
the user can zoom and scroll.

ESDL

Eiffel Simple Directmedia Layer

- **Eiffel Wrapper for SDL: Multimedia Library in C**
- **Object-oriented library to easily develop interactive graphical applications, like little games**
- **"Programming in the large" games, developed by students in summer 2004**
  http://se.inf.ethz.ch/download/games
- **Goal:**
  Using and Extending ESDL for TRAFFIC-Visualization

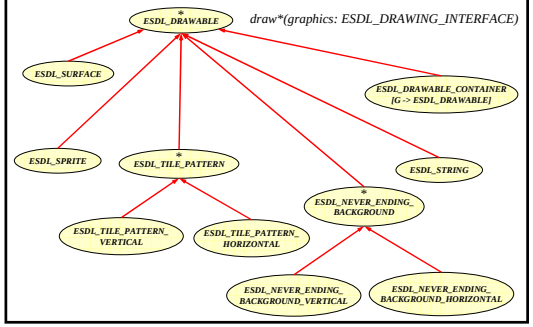
ESDL Extensions

What is new …

- **Drawing Primitives and Figure Classes: Lines, Circles, Rectangles, Polygons**
- **Clipping**
- **Coordinate Transformations: Translation and Scaling**
- **Container to Zoom and Scroll**
- **Keyboard and Mouse Events**
- **Simple Base Classes for ESDL Applications**

ESDL_DRAWABLE

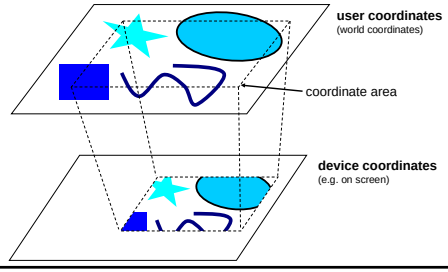
Drawable Objects



ESDL_DRAWING_INTERFACE

Drawing Commands and Coordinates (1)

Drawing Commands drawing relative to a **Coordinate System:**



ESDL_DRAWING_INTERFACE

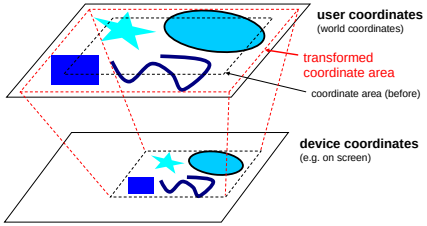
Drawing Commands and Coordinates (2)

```
coordinate_area: ORTHOGONAL_RECTANGLE
        -- Coordinates inside which all drawing primitives
        -- can draw, coordinates outside this area will be clipped

drawing_color: ESDL_COLOR
        -- Color used to draw

line_width: DOUBLE
        -- Line width used to draw lines

draw_polyline (points: DS_LINEAR [VECTOR_2D])
        -- Draw line segments between subsequent points in `points'.
    require
        at_least_two_points: points /= Void
                             and then points.count >= 2

fill_rectangle (a_rectangle: ORTHOGONAL_RECTANGLE)
        -- Draw filled rectangle `a_rectangle' in `drawing_color'.
    require
        a_rectangle_not_void: a_rectangle /= Void
```

ESDL_DRAWING_INTERFACE

Coordinate Transformations: Scaling and Translating

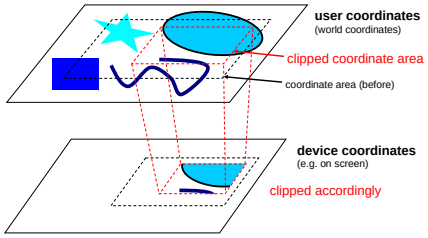
```
transform_coordinates (an_area: ORTHOGONAL_RECTANGLE)

translate_coordinates (a_distance: VECTOR_2D)
```



ESDL_DRAWING_INTERFACE

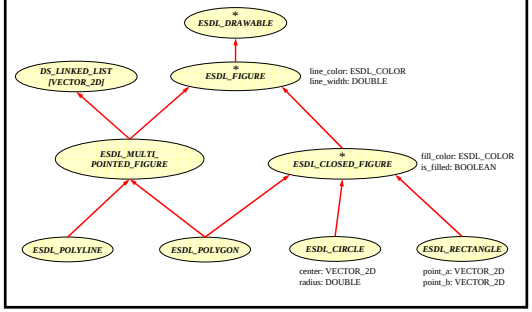
Clipping

```
clip_coordinates (an_area: ORTHOGONAL_RECTANGLE)
```



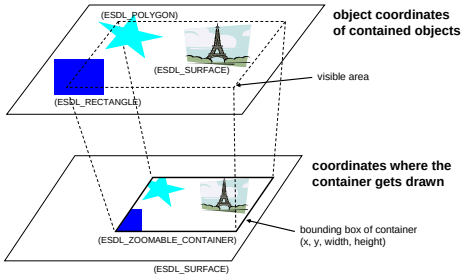
ESDL Figure Classes

Rectangles, Circles, Polygons and Polylines



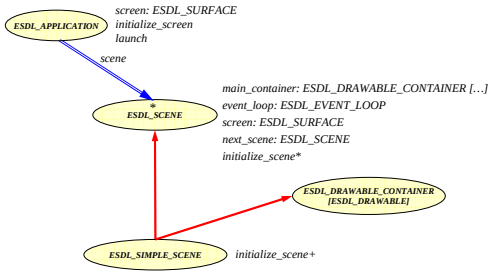
ESDL_ZOOMABLE_CONTAINER

Scrolling and Zooming its Content



Base Classes for Applications

Overview



Base Classes for Applications

Use of ESDL_APPLICATION

```
class MY_APPLICATION
inherit ESDL_APPLICATION
create make_and_launch
feature -- Initialization
      make_and_launch is
                  -- Create and execute application.
            local
                  first_scene: ...
            do
                  -- Initialize the screen.
                  initialize_screen
                  -- Build first scene to run.
                  -- ...
                  -- Set scene and launch.
                  set_scene (first_scene)
                  launch
            end
end
```

Base Classes for Applications

Example – Simple Scene with Zoomable Container

```
make_and_launch is
      -- Create and execute application.
    local
      the_scene: ESDL_SIMPLE_SCENE
      zoom_container: ESDL_ZOOMABLE_CONTAINER
      image: ESDL_SURFACE
    do
      -- Initialize the screen.
      initialize_screen
      -- Build the_scene.
      create the_scene.make
      create image.make_from_image ("pics/world.gif")
      the_scene.extend (image)
      create zoom_container.make (image.width, image.height)
      the_scene.extend (zoom_container)
      zoom_container.set_x_y (0, image.height)
      zoom_container.extend (image)
      zoom_container.zoom (2.0)
      -- Set scene and launch it.
      set_scene (the_scene)
      launch
    end
```

Base Classes for Applications

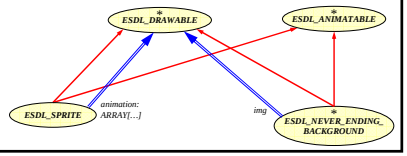
Example – Interactive Scene with Key Event Handling

```
class KEY_SCENE
inherit
  ESDL_SCENE
    redefine
      handle_key_down_event
    end
  ESDL_KEY_CONSTANTS
feature

  handle_key_down_event (a_keyboard_event: ESDL_KEYBOARD_EVENT) is
      -- Handle key down event.
    do
      if a_keyboard_event.key = sdlk_escape then
        event_loop.stop
      end
      if a_keyboard_event.key = sdlk_up then
        zoom_container.zoom (1.25)
      end
    end
```

Animation

ESDL_ANIMATABLE

```
deferred class
  ESDL_ANIMATABLE

feature -- Basic operations

  go_to_time (a_time: INTEGER) is
      -- Change Current object state to
      -- state at a_time (in milliseconds).
    deferred
    end

end
```



Animation

Animation Support in ESDL_SCENE

```
start_animating (an_animatable: ESDL_ANIMATABLE)
        -- Subscribe an_animatable to be animated
        -- when Current is running.

stop_animating (an_animatable: ESDL_ANIMATABLE)
        -- Unsubscribe an_animatable from beeing animated
        -- when Current is running.

animate
        -- Let all subscribed animatable objects
        -- perform their animation.
        -- (Calls go_to_time of subscribed animatable objects
        --  with current time tick)

animation_event: EVENT_TYPE [TUPLE [INTEGER]]
        -- Animation event, allows animatable objects
        -- to perform animation (i.e. moving themselves)
        -- before they get drawen.
```

Animation

Example - ESDL_SPRITE in an ESDL_SCENE

```
class MY_SCENE

inherit ESDL_SCENE

feature -- Initialization

  initialize_scene is
      -- Initialize the scene.
    local
      my_animation: ESDL_ANIMATION
      my_sprite: ESDL_SPRITE
    do
      -- Add some simple sprite animation.
      create my_animation.make_from_file ("./pics/alien.anim")
      create my_sprite.make (my_animation)
      my_sprite.set_x_y (500, 100)
      main_container.extend (my_sprite)
      start_animating (my_sprite)
    end
end
```

Subscribe for Animation !

Mouse Events

Over ESDL_DRAWABLEs

```
mouse_button_down_event: EVENT_TYPE [TUPLE []]
   -- Mouse button down event,
   -- gets published when the mouse button
   -- is pressed over Current,
   -- an ESDL_MOUSEBUTTON_EVENT is passed as argument

mouse_button_up_event: EVENT_TYPE [TUPLE []]
   -- Mouse button up event,
   -- gets published when the mouse button
   -- is released over Current,
   -- an ESDL_MOUSEBUTTON_EVENT is passed as argument

mouse_motion_event: EVENT_TYPE [TUPLE []]
   -- Mouse motion event,
   -- gets published when the mouse button is moved over Current,
   -- an ESDL_MOUSEMOTION_EVENT is passed as argument
```

Mouse Events

In ESDL_DRAWABLE_CONTAINER & ESDL_ZOOMABLE_CONTAINER

```
   mouse_button_down_on_item_event: EVENT_TYPE [TUPLE [G]]
          -- Mouse button down on item event,
          -- gets published when the mouse button event
          -- is caught by an item inside Current,
          -- item is passed as first argument to subscribers,
          -- an ESDL_MOUSEBUTTON_EVENT is passed
          -- as optional second argument

   mouse_button_up_on_item_event: EVENT_TYPE [TUPLE [G]]
          -- Mouse button up on item event,
          -- ...

   mouse_motion_on_item_event: EVENT_TYPE [TUPLE [G]]
          -- Mouse motion on item event,
          -- gets published when the mouse motion event
          -- is caught by an item inside Current,
          -- item is passed as first argument to subscribers,
          -- an ESDL_MOUSEBUTTON_EVENT is passed
          -- as optional second argument
```

THE END