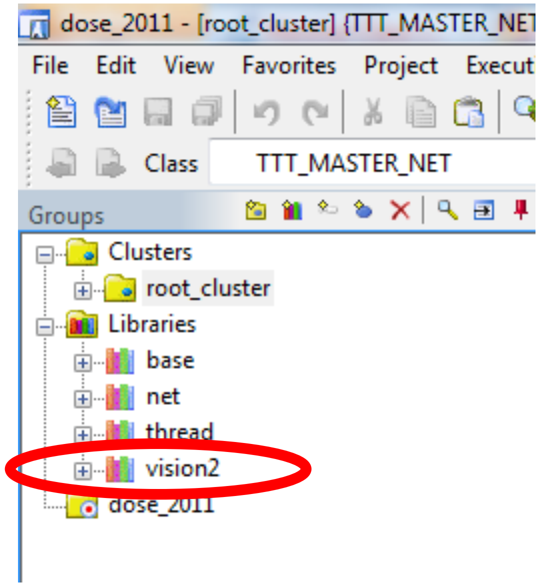
Introduction to Eiffel Vision 2

Christian Estler

Distributed Software Engineering Lab 2015

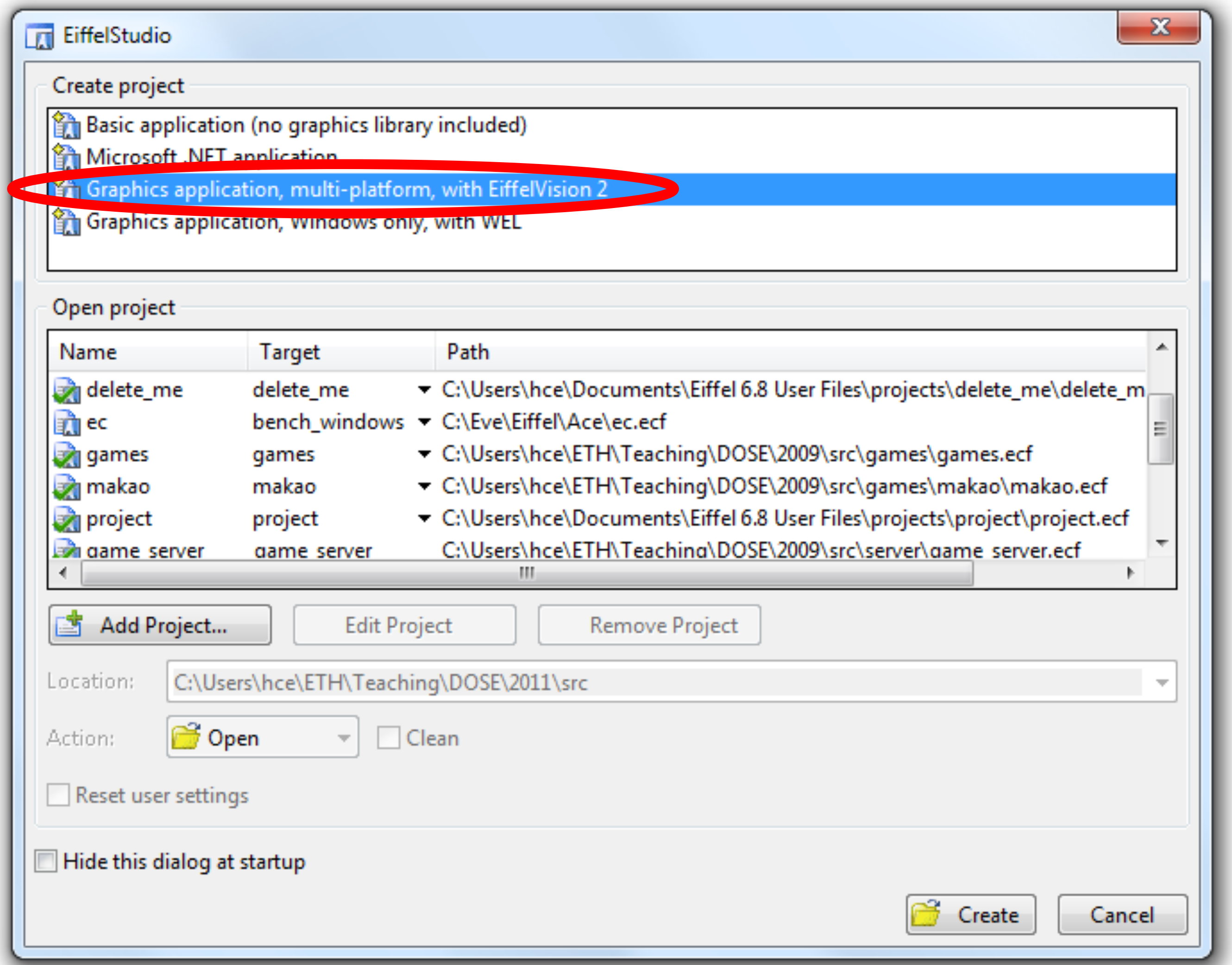
What it‘s about...

- Eiffel Vision 2 is a UI library
- Allows for building platform independent graphical UIs
    - Windows
    - Linux
    - Mac OS



A first EV2 Application (1)

Create a new project...



EiffelStudio

Create project

- Basic application (no graphics library included)
- Microsoft .NET application
- [✓ Graphics application, multi-platform, with EiffelVision 2]
- Graphics application, Windows only, with WEL

Open project

Name	Target	Path
delete_me	delete_me	C:\Users\hce\Documents\Eiffel 6.8 User Files\projects\delete_me\delete_m
ec	bench_windows	C:\Eve\Eiffel\Ace\ec.ecf
games	games	C:\Users\hce\ETH\Teaching\DOSE\2009\src\games\games.ecf
makao	makao	C:\Users\hce\ETH\Teaching\DOSE\2009\src\games\makao\makao.ecf
project	project	C:\Users\hce\Documents\Eiffel 6.8 User Files\projects\project\project.ecf
game_server	game_server	C:\Users\hce\ETH\Teaching\DOSE\2009\src\server\game_server.ecf

Add Project... | Edit Project | Remove Project

Location: C:\Users\hce\ETH\Teaching\DOSE\2011\src

Action: Open ☐ Clean

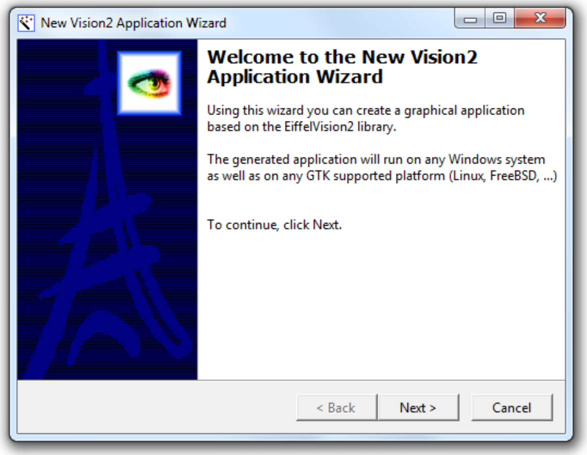
☐ Reset user settings

☐ Hide this dialog at startup

Create | Cancel

A first EV2 Application (2)

Step through the wizard...



New Vision2 Application Wizard

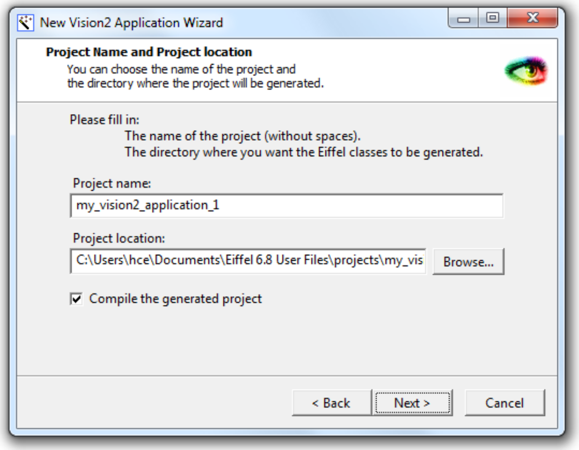
Welcome to the New Vision2 Application Wizard

Using this wizard you can create a graphical application based on the EiffelVision2 library.

The generated application will run on any Windows system as well as on any GTK supported platform (Linux, FreeBSD, ...)

To continue, click Next.

< Back | Next > | Cancel



New Vision2 Application Wizard

Project Name and Project location

You can choose the name of the project and the directory where the project will be generated.

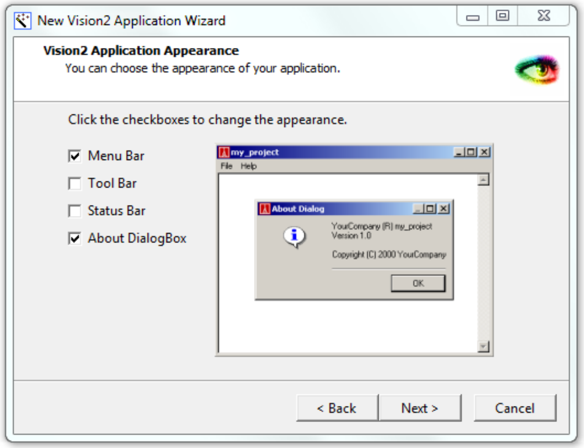
Please fill in:
The name of the project (without spaces).
The directory where you want the Eiffel classes to be generated.

Project name: my_vision2_application_1

Project location: C:\Users\hce\Documents\Eiffel 6.8 User Files\projects\my_vis | Browse...

☑ Compile the generated project

< Back | Next > | Cancel



New Vision2 Application Wizard

Vision2 Application Appearance

You can choose the appearance of your application.

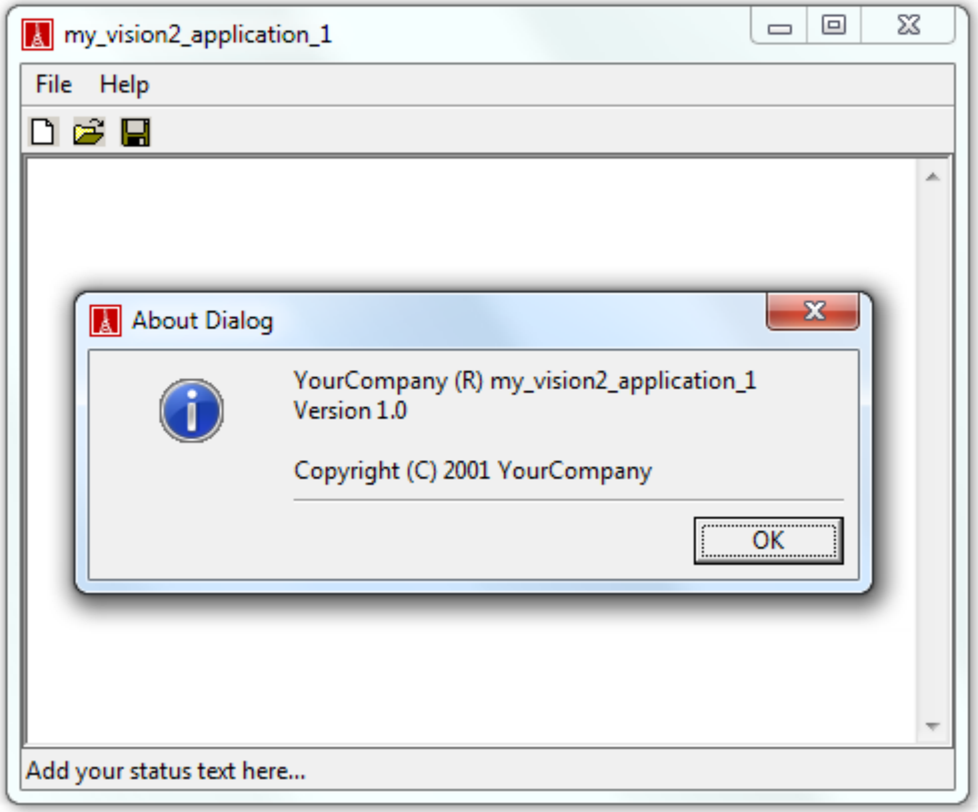
Click the checkboxes to change the appearance.

☑ Menu Bar
☐ Tool Bar
☐ Status Bar
☑ About DialogBox

< Back | Next > | Cancel

A first EV2 Application (3)

Compile and run...



YourCompany (R) my_vision2_application_1
Version 1.0

Copyright (C) 2001 YourCompany

Add your status text here...

Note: EiffelStudio uses Vision2 for its GUI

Structure of an EV2 UI

- Start with a main window, i.e. inherit from
  - class EV_WINDOW or
  - class EV_TITLED_WINDOW or
  - .... (some other WINDOW class you want)

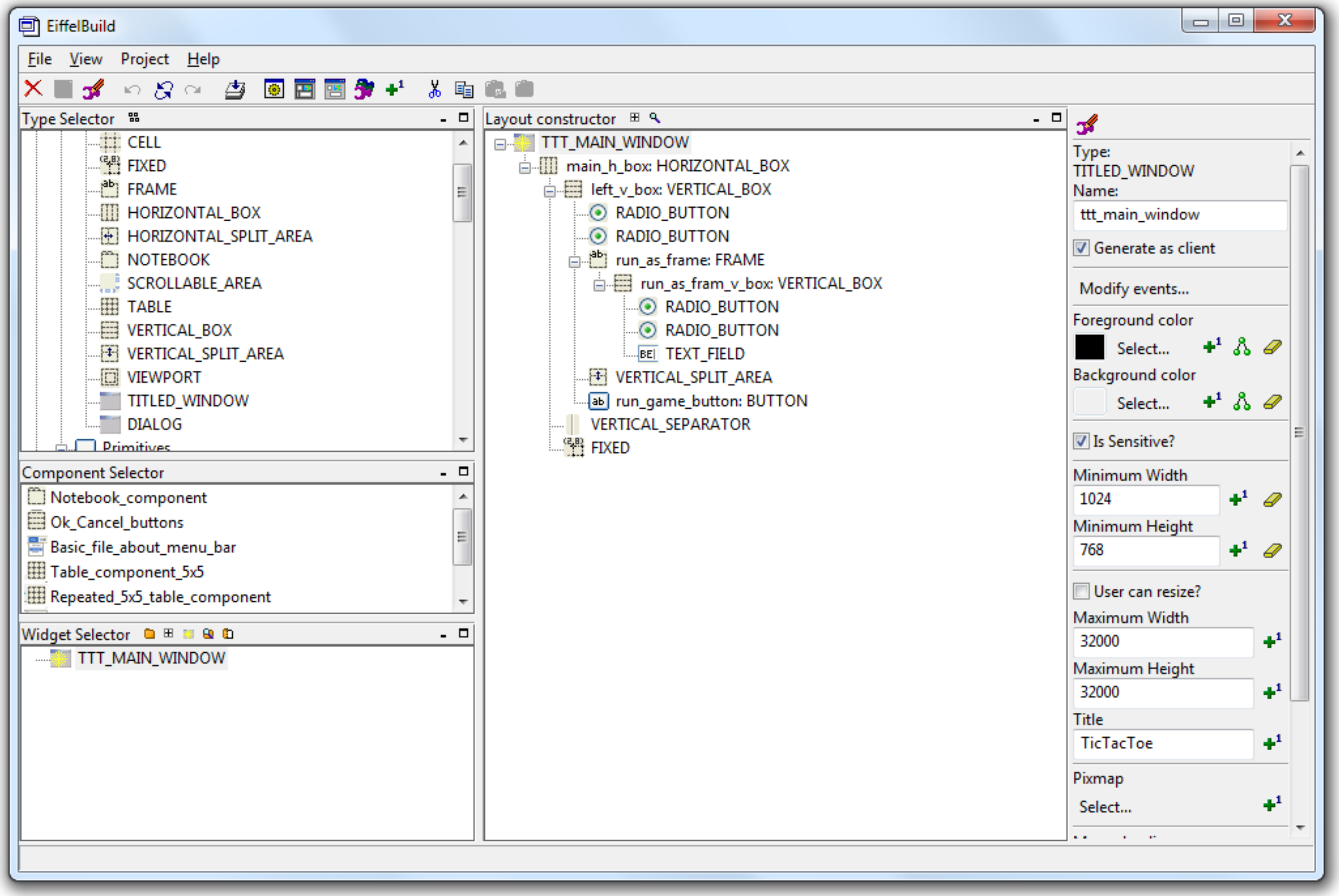
- Extend window with
  - **Containers**
    - To define layouts, e.g. Vertical, Horizontal; or other Windows…
  - **Primitives**
    - E.g. Buttons, Labels, Text Fields, ...
  - **Items**
    - E.g. Menu bar, Tool bar, Tool bar button, …

both are Widgets

How to get an overview of EV?

- Your Eiffel installation comes with the **EiffelBuild** application
- EiffelBuild is a simple GUI Builder
  - Lets you pick&drop the tree structure of an UI
  - Generates code
- You can use it to get familiar with EV2

EiffelBuild



EiffelBuild

File View Project Help

Type Selector

- CELL
- FIXED
- FRAME
- HORIZONTAL_BOX
- HORIZONTAL_SPLIT_AREA
- NOTEBOOK
- SCROLLABLE_AREA
- TABLE
- VERTICAL_BOX
- VERTICAL_SPLIT_AREA
- VIEWPORT
- TITLED_WINDOW
- DIALOG
- Primitives

Component Selector

- Notebook_component
- Ok_Cancel_buttons
- Basic_file_about_menu_bar
- Table_component_5x5
- Repeated_5x5_table_component

Widget Selector

- TTT_MAIN_WINDOW

Layout constructor

```
TTT_MAIN_WINDOW
  main_h_box: HORIZONTAL_BOX
    left_v_box: VERTICAL_BOX
      RADIO_BUTTON
      RADIO_BUTTON
      run_as_frame: FRAME
        run_as_fram_v_box: VERTICAL_BOX
          RADIO_BUTTON
          RADIO_BUTTON
          TEXT_FIELD
      VERTICAL_SPLIT_AREA
      run_game_button: BUTTON
    VERTICAL_SEPARATOR
    FIXED
```

Type:
TITLED_WINDOW

Name: ttt_main_window

[✓] Generate as client

Modify events...

Foreground color: Select...

Background color: Select...

[✓] Is Sensitive?

Minimum Width: 1024

Minimum Height: 768

[ ] User can resize?

Maximum Width: 32000

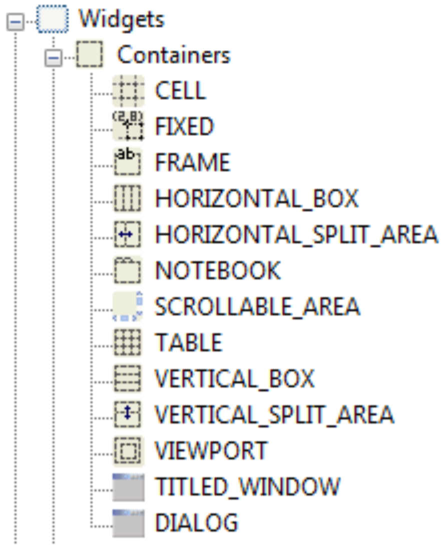
Maximum Height: 32000

Title: TicTacToe

Pixmap: Select...

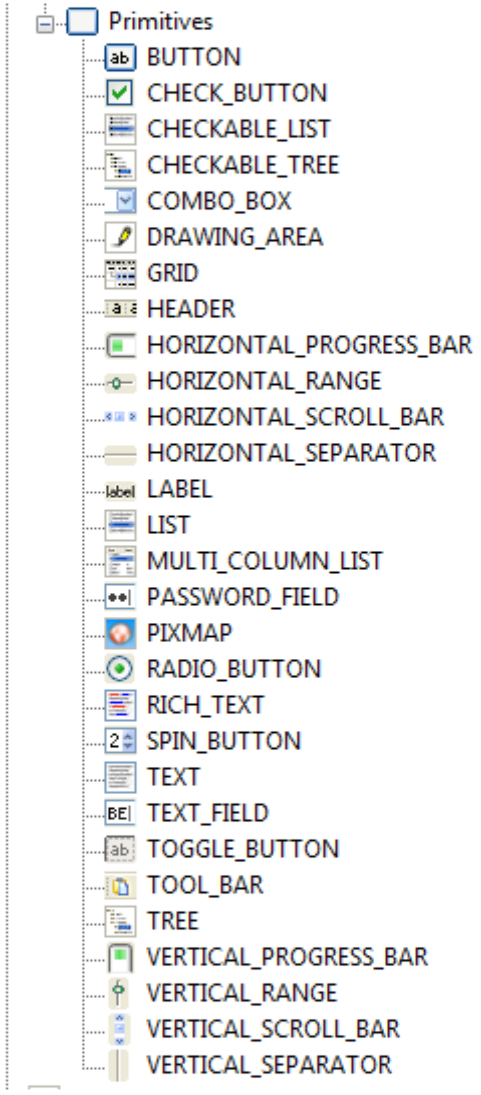
EiffelBuild: Containers, Primitives & Items

Containers



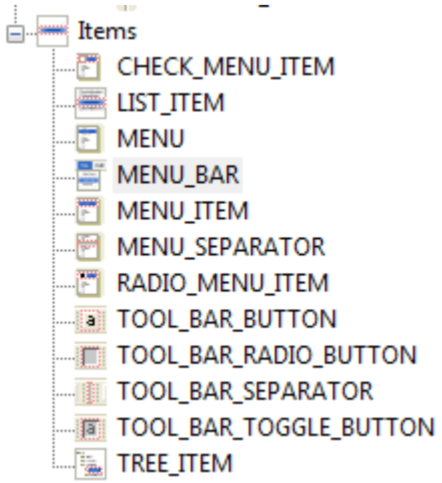
- Widgets
  - Containers
    - CELL
    - FIXED
    - FRAME
    - HORIZONTAL_BOX
    - HORIZONTAL_SPLIT_AREA
    - NOTEBOOK
    - SCROLLABLE_AREA
    - TABLE
    - VERTICAL_BOX
    - VERTICAL_SPLIT_AREA
    - VIEWPORT
    - TITLED_WINDOW
    - DIALOG

Primitives



- Primitives
  - BUTTON
  - CHECK_BUTTON
  - CHECKABLE_LIST
  - CHECKABLE_TREE
  - COMBO_BOX
  - DRAWING_AREA
  - GRID
  - HEADER
  - HORIZONTAL_PROGRESS_BAR
  - HORIZONTAL_RANGE
  - HORIZONTAL_SCROLL_BAR
  - HORIZONTAL_SEPARATOR
  - LABEL
  - LIST
  - MULTI_COLUMN_LIST
  - PASSWORD_FIELD
  - PIXMAP
  - RADIO_BUTTON
  - RICH_TEXT
  - SPIN_BUTTON
  - TEXT
  - TEXT_FIELD
  - TOGGLE_BUTTON
  - TOOL_BAR
  - TREE
  - VERTICAL_PROGRESS_BAR
  - VERTICAL_RANGE
  - VERTICAL_SCROLL_BAR
  - VERTICAL_SEPARATOR

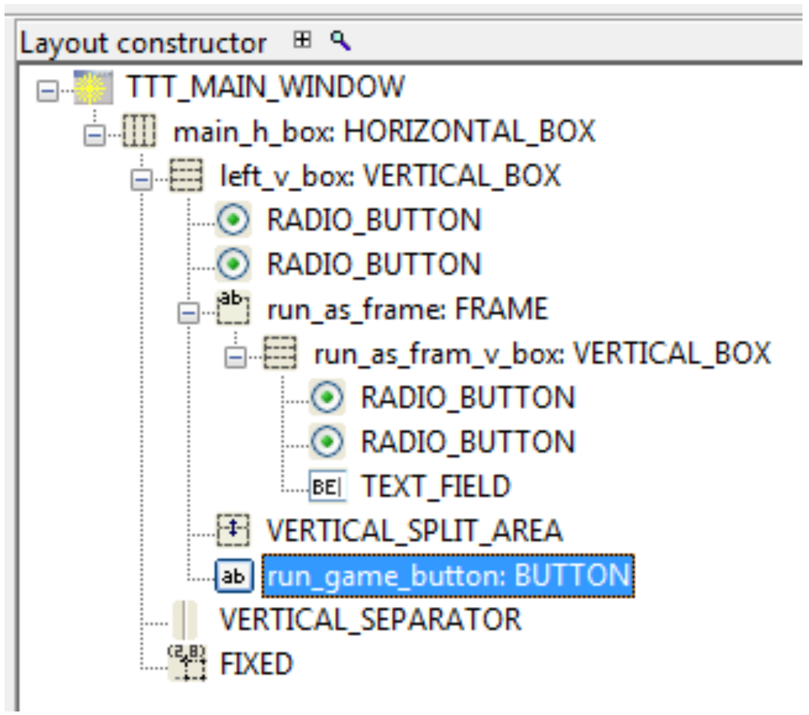
Items



- Items
  - CHECK_MENU_ITEM
  - LIST_ITEM
  - MENU
  - MENU_BAR
  - MENU_ITEM
  - MENU_SEPARATOR
  - RADIO_MENU_ITEM
  - TOOL_BAR_BUTTON
  - TOOL_BAR_RADIO_BUTTON
  - TOOL_BAR_SEPARATOR
  - TOOL_BAR_TOGGLE_BUTTON
  - TREE_ITEM

EiffelBuild

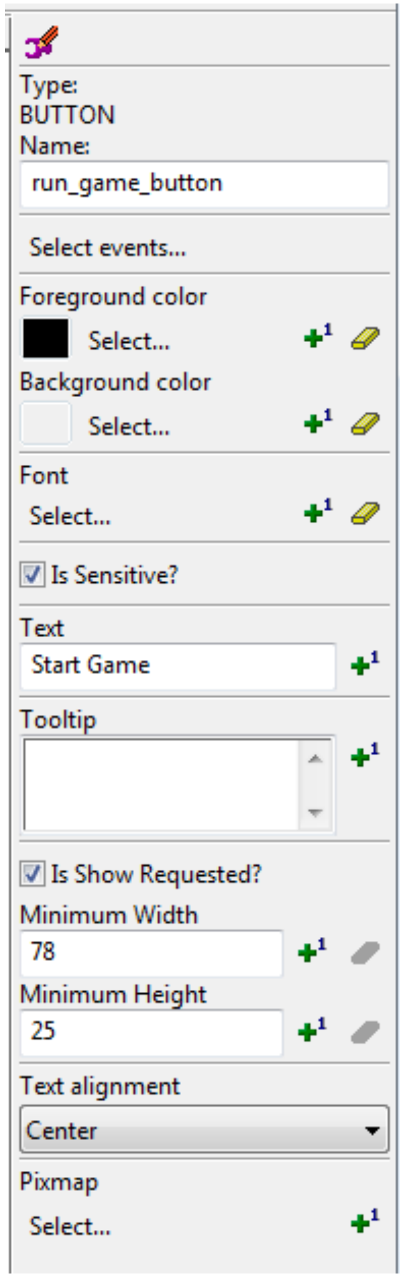
Design tree structure



Layout constructor

- TTT_MAIN_WINDOW
  - main_h_box: HORIZONTAL_BOX
    - left_v_box: VERTICAL_BOX
      - RADIO_BUTTON
      - RADIO_BUTTON
      - run_as_frame: FRAME
        - run_as_fram_v_box: VERTICAL_BOX
          - RADIO_BUTTON
          - RADIO_BUTTON
          - TEXT_FIELD
      - VERTICAL_SPLIT_AREA
      - run_game_button: BUTTON
    - VERTICAL_SEPARATOR
    - FIXED

Set properties of elements



Type:
BUTTON
Name:
run_game_button

Select events...

Foreground color
Select...

Background color
Select...

Font
Select...

☑ Is Sensitive?

Text
Start Game

Tooltip

☑ Is Show Requested?

Minimum Width
78

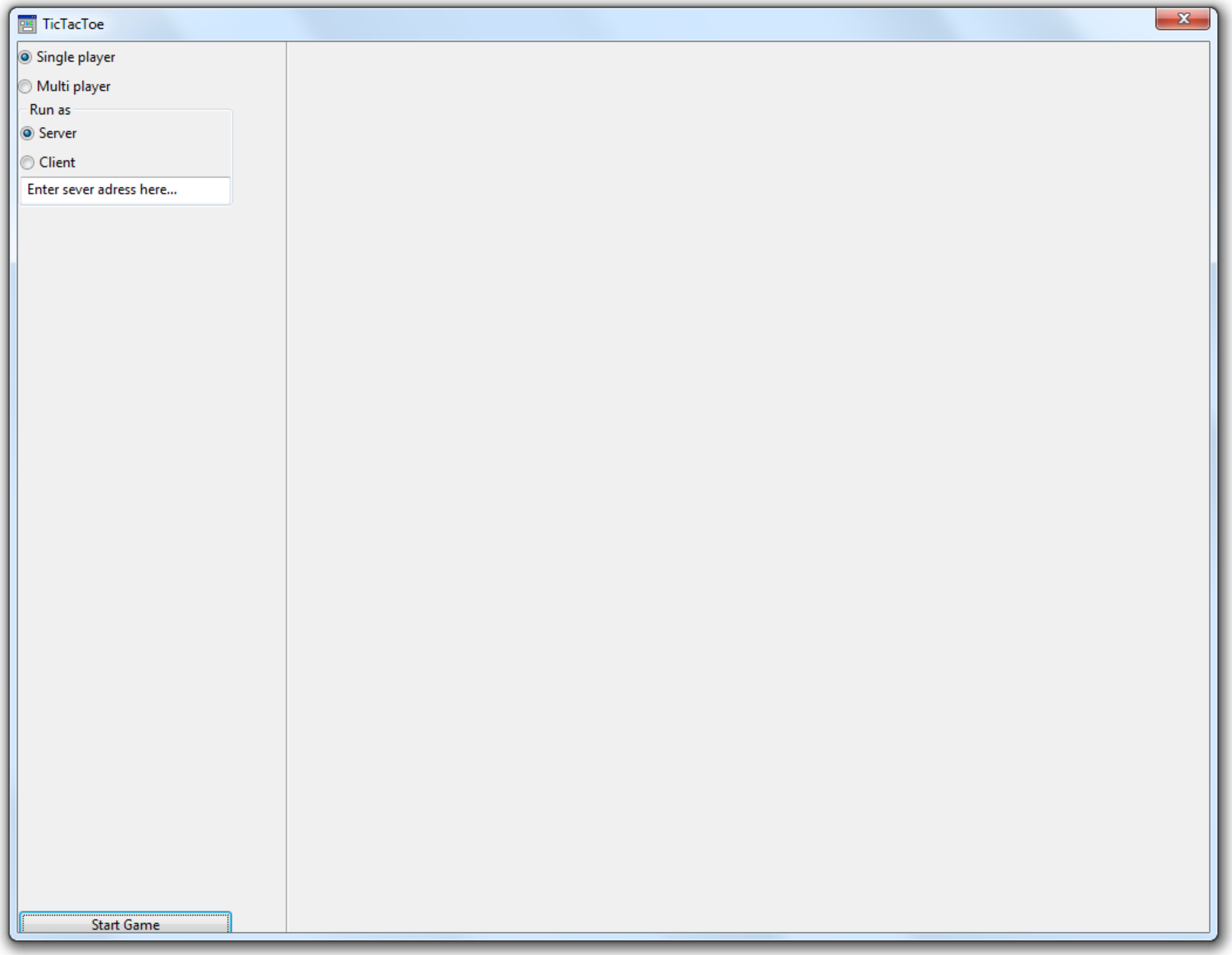
Minimum Height
25

Text alignment
Center

Pixmap
Select...

EiffelBuild

Preview your UI



EV2 for Games

- How to use EV2 for (board) games?

- Two main approaches
    - Use containers with background images and EV_PIXMAP
    - Use EV_MODEL to draw on a EV_DRAWING_AREA

EV2 Game based on containers (1)

- Use lots of containers with
  - Subcontainers with
  - Subsubcontainers with
  - Subsubcontainers with
  - ...

- Containers typically have
  - Fixed or relative positions
  - background images (*.bmp, *.png)

Tip: only have on PIXMAP per container. Add additional images “on top” via pixmap.draw_pixmap

How to build a game with containers?

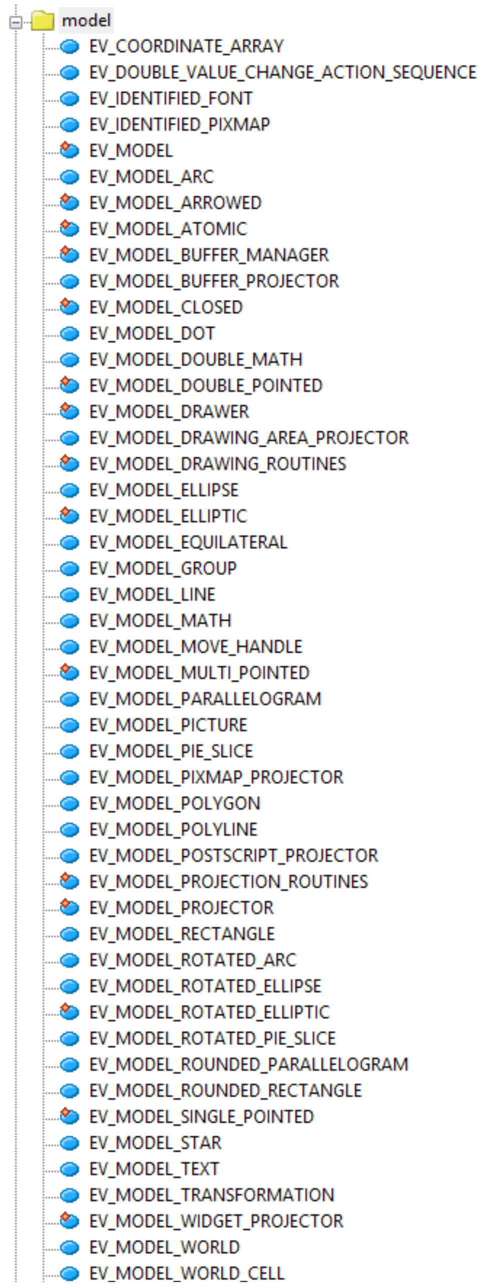
- Main idea
    - Main container has background image
    - Other containers have pictures representing the game state
    - Images change on Mouse/Keyboard events

- This works fine if you don’t need a lot transparency or dragging of elements over the board game

- If you need to move elements or want to “draw” instead of using images, use MODEL

EV2 Game based on Model

- Eiffel Vision has MODEL classes
- Can be used to draw elements on a drawing area
  - Line, ellipse, pie-slice, polygon, rectangle, text, star…
- Model classes work better with transparent pictures



model
- EV_COORDINATE_ARRAY
- EV_DOUBLE_VALUE_CHANGE_ACTION_SEQUENCE
- EV_IDENTIFIED_FONT
- EV_IDENTIFIED_PIXMAP
- EV_MODEL
- EV_MODEL_ARC
- EV_MODEL_ARROWED
- EV_MODEL_ATOMIC
- EV_MODEL_BUFFER_MANAGER
- EV_MODEL_BUFFER_PROJECTOR
- EV_MODEL_CLOSED
- EV_MODEL_DOT
- EV_MODEL_DOUBLE_MATH
- EV_MODEL_DOUBLE_POINTED
- EV_MODEL_DRAWER
- EV_MODEL_DRAWING_AREA_PROJECTOR
- EV_MODEL_DRAWING_ROUTINES
- EV_MODEL_ELLIPSE
- EV_MODEL_ELLIPTIC
- EV_MODEL_EQUILATERAL
- EV_MODEL_GROUP
- EV_MODEL_LINE
- EV_MODEL_MATH
- EV_MODEL_MOVE_HANDLE
- EV_MODEL_MULTI_POINTED
- EV_MODEL_PARALLELOGRAM
- EV_MODEL_PICTURE
- EV_MODEL_PIE_SLICE
- EV_MODEL_PIXMAP_PROJECTOR
- EV_MODEL_POLYGON
- EV_MODEL_POLYLINE
- EV_MODEL_POSTSCRIPT_PROJECTOR
- EV_MODEL_PROJECTION_ROUTINES
- EV_MODEL_PROJECTOR
- EV_MODEL_RECTANGLE
- EV_MODEL_ROTATED_ARC
- EV_MODEL_ROTATED_ELLIPSE
- EV_MODEL_ROTATED_ELLIPTIC
- EV_MODEL_ROTATED_PIE_SLICE
- EV_MODEL_ROUNDED_PARALLELOGRAM
- EV_MODEL_ROUNDED_RECTANGLE
- EV_MODEL_SINGLE_POINTED
- EV_MODEL_STAR
- EV_MODEL_TEXT
- EV_MODEL_TRANSFORMATION
- EV_MODEL_WIDGET_PROJECTOR
- EV_MODEL_WORLD
- EV_MODEL_WORLD_CELL

How to get started with EV Model

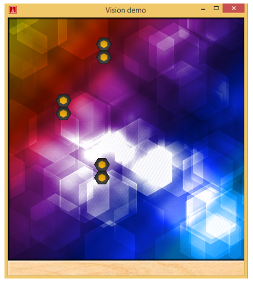
1. Start from a usual EV application

2. Add an *area: EV_DRAWING_AREA* to a container

3. Create objects
   - *projector: EV_MODEL_DRAWING_AREA_PROJECTOR*
   - *world: EV_MODEL_WORLD*

4. Add the *world* and the *area* to the projector

Demo: https://github.com/DOSE-ETH/eiffel_vision_demo



Your turn...

We recommend you

- Build a first EV2 application (EiffelStudio default one)
  and take a look at the code

- Play around with EiffelBuild quickly learn about available widgets

- Take a look at the provided example

- Take a look at:
  http://docs.eiffel.com/book/solutions/eiffelvision-2