



Elektrobit

# EB GUIDE Studio

用户手册

版本 6.6.0.142803



Elektrobit Automotive GmbH  
Am Wolfsmantel 46  
D-91058 Erlangen  
GERMANY

Phone: +49 9131 7701-0  
Fax: +49 9131 7701-6333  
<http://www.elektrobit.com>

## Legal notice

Confidential and proprietary information.

ALL RIGHTS RESERVED. No part of this publication may be copied in any form, by photocopy, microfilm, retrieval system, or by any other means now known or hereafter invented without the prior written permission of Elektrobit Automotive GmbH.

ProOSEK®, tresos®, and street director® are registered trademarks of Elektrobit Automotive GmbH.

All brand names, trademarks and registered trademarks are property of their rightful owners and are used only for description.

Copyright 2018, Elektrobit Automotive GmbH.

# 目录

1. 关于本文档 .....	15
1.1. 目标群体：建模员 .....	15
1.2. 用户文档的结构 .....	15
1.3. 印刷和样式约定 .....	16
1.4. 命名约定 .....	17
1.5. 路径约定 .....	18
2. 安全并正确使用 .....	19
2.1. 预期用途 .....	19
2.2. 可能的误用 .....	19
3. 支持 .....	20
4. 简介 EB GUIDE .....	21
4.1. 此 EB GUIDE product line .....	21
4.2. EB GUIDE Studio .....	21
4.2.1. 对 HMI 行为进行建模 .....	21
4.2.2. 对 HMI 外观进行建模 .....	22
4.2.3. 处理数据 .....	22
4.2.4. 模拟 EB GUIDE 模型 .....	22
4.2.5. 导出 EB GUIDE 模型 .....	23
4.3. EB GUIDE TF .....	23
5. 教程：入门 .....	24
5.1. 起始 EB GUIDE .....	24
5.2. 创建项目 .....	25
5.3. 对 HMI 行为进行建模 .....	26
5.4. 对 HMI 外观进行建模 .....	29
5.5. 启动模拟运行 .....	31
6. 背景信息 .....	33
6.1. 3D 图形 .....	33
6.1.1. 受支持的 3D 图形格式 .....	33
6.1.2. 3D 图形文件的设置 .....	33
6.1.3. 导入 3D 图形文件 .....	34
6.2. 动画 .....	35
6.2.1. 控件动画 .....	35
6.2.2. 视图转变的动画 .....	36
6.3. 应用程序和模型之间的应用程序编程接口 .....	36
6.4. 通信环境 .....	36
6.5. 图形用户界面元素 .....	37
6.5.1. 项目中心 .....	37
6.5.1.1. 导航区 .....	38
6.5.1.2. 内容区域 .....	38

6.5.2. 项目编辑器 .....	38
6.5.2.1. 导航元素 .....	39
6.5.2.2. “概要”元素 .....	40
6.5.2.3. 工具箱元素 .....	40
6.5.2.4. 属性元素 .....	41
6.5.2.5. 内容区域 .....	42
6.5.2.6. “事件”元素 .....	43
6.5.2.7. “数据池”元素 .....	44
6.5.2.8. “资产”元素 .....	44
6.5.2.9. 命令区 .....	44
6.5.2.10. 问题元素 .....	45
6.5.3. 可对接元素 .....	45
6.5.4. EB GUIDE Monitor .....	46
6.6. 数据池 .....	48
6.6.1. 概念 .....	48
6.6.2. 数据池项 .....	48
6.6.3. 窗口化列表 .....	49
6.7. EB GUIDE 模型和 EB GUIDE 项目 .....	49
6.8. 事件处理 .....	50
6.8.1. 事件系统 .....	50
6.8.2. 事件 .....	50
6.9. 扩展 .....	50
6.9.1. EB GUIDE Studio 扩展 .....	51
6.9.2. EB GUIDE GTF 扩展 .....	51
6.10. 语言 .....	51
6.10.1. EB GUIDE 中的显示语言 .....	51
6.10.2. EB GUIDE 模型中的语言 .....	51
6.10.3. 导出和导入语言依赖项文本 .....	52
6.11. 外观 .....	52
6.12. 资源管理 .....	53
6.12.1. 字体 .....	53
6.12.1.1. 位图字体 .....	53
6.12.2. 图像 .....	54
6.12.2.1. 9-patch 图像 .....	54
6.12.3. 3D 图形的网格 .....	55
6.12.4. .psd 文件格式 .....	55
6.13. 脚本语言 EB GUIDE 脚本 .....	55
6.13.1. 应用程序的功能和区域 .....	55
6.13.2. 命名空间和识别符 .....	56
6.13.3. 注释 .....	57
6.13.4. 类型 .....	57
6.13.5. 表达式 .....	58

6.13.6. 常量和引用 .....	58
6.13.7. 算术和逻辑表达式 .....	59
6.13.8. 左值和右值 .....	60
6.13.9. 局部变量 .....	60
6.13.10. While 循环 .....	61
6.13.11. If-then-else .....	62
6.13.12. 外部函数调用 .....	63
6.13.13. 数据池访问 .....	64
6.13.14. 控件属性 .....	64
6.13.15. 列表 .....	65
6.13.16. 事件 .....	66
6.13.17. 字符串格式设置 .....	68
6.13.18. 标准库 .....	68
6.14. 脚本化的值 .....	68
6.15. 键盘快捷键、按钮和图标 .....	70
6.15.1. 键盘快捷键 .....	70
6.15.2. 命令行选项 .....	71
6.15.2.1. Studio.Console.exe 的命令行选项 .....	71
6.15.2.2. 命令行选项 Monitor.Console.exe .....	72
6.15.3. 按钮 .....	72
6.15.4. 图标 .....	73
6.16. 状态机和状态 .....	74
6.16.1. 状态机 .....	74
6.16.1.1. 触觉状态机 .....	74
6.16.1.2. 逻辑状态机 .....	74
6.16.1.3. 动态状态机 .....	74
6.16.2. 状态 .....	75
6.16.2.1. 组合状态 .....	75
6.16.2.2. 视图状态 .....	76
6.16.2.3. 初始状态 .....	76
6.16.2.4. 结束状态 .....	77
6.16.2.5. 选择状态 .....	78
6.16.2.6. 历史状态 .....	79
6.16.3. 转换 .....	82
6.16.4. 状态机的执行 .....	86
6.16.5. EB GUIDE 与 UML 标记相比的 标记 .....	90
6.16.5.1. 受支持的元素 .....	90
6.16.5.2. 不受支持的元素 .....	90
6.16.5.3. 偏差 .....	91
6.17. 触摸输入 .....	91
6.17.1. 非路径动作 .....	91
6.17.2. 路径动作 .....	92

6.17.3. 输入处理和操作动作 .....	92
6.17.4. 多点触摸输入 .....	92
6.18. 控件 .....	93
6.18.1. 视图 .....	93
6.18.2. 控件类别 .....	94
6.18.3. 控件属性 .....	95
6.18.4. 控件模板 .....	96
6.18.5. 控件功能 .....	97
6.18.5.1. 焦点控件功能类别 .....	98
6.18.5.2. 列表管理控件功能类别 .....	99
7. 对 HMI 行为进行建模 .....	100
7.1. 对状态机进行建模 .....	100
7.1.1. 添加状态机 .....	100
7.1.2. 添加动态状态机 .....	100
7.1.3. 为状态机定义进入动作 .....	101
7.1.4. 为状态机定义退出动作 .....	101
7.1.5. 删除状态机 .....	102
7.2. 对状态进行建模 .....	102
7.2.1. 添加状态 .....	102
7.2.2. 向组合状态添加状态 .....	103
7.2.3. 添加选择状态 .....	104
7.2.4. 为状态定义进入动作 .....	105
7.2.5. 为状态定义退出动作 .....	106
7.2.6. 从状态机中删除模型元素 .....	106
7.3. 通过转换连接状态 .....	107
7.3.1. 在两种状态之间添加转换 .....	107
7.3.2. 移动转换 .....	108
7.3.3. 为转换定义触发器 .....	109
7.3.4. 向转换添加条件 .....	110
7.3.5. 向转换添加操作 .....	111
7.3.6. 向状态添加内部转换 .....	113
8. 对 HMI 外观进行建模 .....	114
8.1. 使用控件 .....	114
8.1.1. 添加视图 .....	114
8.1.2. 向视图添加基本控件 .....	115
8.1.2.1. 添加矩形 .....	115
8.1.2.2. 添加椭圆 .....	115
8.1.2.2.1. 编辑椭圆 .....	115
8.1.2.3. 添加图像 .....	116
8.1.2.4. 添加一个标签 .....	118
8.1.2.4.1. 更改标签的字体 .....	118
8.1.2.5. 添加容器 .....	119

8.1.2.6. 添加实例化器 .....	120
8.1.2.7. 添加动画 .....	122
8.1.2.8. 添加 alpha 掩码 .....	123
8.1.3. 向一个 3D 控件到视图 .....	124
8.1.3.1. 向视图添加场景图 .....	124
8.1.4. 添加 .psd 文件到视图中 .....	125
8.1.5. 从视图中删除控件 .....	126
8.2. 使用控件属性 .....	126
8.2.1. 确定控件的位置 .....	126
8.2.2. 调整控件的大小 .....	127
8.2.3. 控件属性之间的链接 .....	128
8.2.4. 将控件属性链接到数据池项 .....	130
8.2.5. 向控件添加用户定义的属性 .....	132
8.2.5.1. 添加用户定义的以下类型的属性 Function (): bool .....	133
8.2.6. 重命名用户定义的属性 .....	134
8.3. 按控件功能扩展控件 .....	135
8.3.1. 添加控件功能 .....	135
8.3.2. 删除控件功能 .....	137
8.4. 向 EB GUIDE 模型中添加语言 .....	138
8.4.1. 添加语言 .....	139
8.4.2. 删除语言 .....	140
8.5. 使用语言支持 .....	140
8.5.1. 向 EB GUIDE 模型中添加外观 .....	141
8.5.2. 向数据池项添加外观支持 .....	141
8.5.3. 外观间转换 .....	142
8.5.4. 删除外观 .....	142
8.6. 给视图转变制作动画 .....	143
8.6.1. 添加入口动画 .....	143
8.6.2. 添加退出动画 .....	144
8.7. 重复使用控件 .....	144
8.7.1. 添加模板 .....	144
8.7.2. 定义模板接口 .....	145
8.7.3. 使用模板 .....	146
8.7.4. 删除模板 .....	146
9. 处理数据 .....	148
9.1. 添加事件 .....	148
9.2. 向事件添加参数 .....	148
9.3. 确定事件位置 .....	149
9.4. 删除事件 .....	150
9.5. 添加数据池项 .....	150
9.6. 编辑列表类型的数据池项 .....	151
9.7. 将属性转换为脚本化的值 .....	151

9.8. 建立外部通信 .....	152
9.9. 数据池项之间的链接 .....	154
9.10. 删除数据池项 .....	155
10. 处理项目 .....	156
10.1. 创建项目 .....	156
10.2. 打开项目 .....	157
10.2.1. 从文件资源管理器中打开项目 .....	157
10.2.2. 在.....中打开项目 EB GUIDE Studio .....	157
10.3. 重命名模型元素 .....	158
10.4. 验证和模拟运行 EB GUIDE 模型 .....	158
10.4.1. 验证 EB GUIDE 模型 .....	158
10.4.1.1. 使用.....验证 EB GUIDE 模型 EB GUIDE Studio .....	158
10.4.1.2. 使用命令行验证 EB GUIDE 模型 .....	159
10.4.2. 启动和停止模拟运行 .....	160
10.5. 导出 EB GUIDE 模型 .....	160
10.5.1. 使用以下工具导出 EB GUIDE 模型 EB GUIDE Studio .....	160
10.5.2. 使用命令行导出 EB GUIDE 模型 .....	161
10.6. 更改 EB GUIDE Studio 的显示语言 .....	161
10.7. 配置文件 .....	162
10.7.1. 添加配置文件 .....	162
10.7.2. 添加库 .....	163
10.7.3. 配置场景 .....	165
10.8. 导出和导入语言依赖项文本 .....	165
10.8.1. 导出语言依赖项文本 .....	166
10.8.2. 导入语言依赖项文本 .....	167
10.8.2.1. 使用以下工具导入语言依赖项文本 EB GUIDE Studio .....	167
10.8.2.2. 使用命令行导入语言依赖项文本 .....	168
10.9. 使用 EB GUIDE Monitor .....	168
10.9.1. 在 EB GUIDE Monitor 触发事件 .....	168
10.9.2. 使用 EB GUIDE Monitor 更改数据池项的值 .....	169
10.9.3. 在 EB GUIDE Monitor 中启动脚本 .....	170
10.9.3.1. 编写脚本文件以作为 EB GUIDE Monitor .....	171
10.9.4. 作为独立程序启动 EB GUIDE Monitor .....	174
11. 教程 .....	176
11.1. 教程：添加动态状态机 .....	176
11.2. 教程：使用 EB GUIDE 脚本 对按钮行为进行建模 .....	184
11.3. 教程：对路径动作进行建模 .....	190
11.4. 教程：使用动态内容创建 列表 .....	193
11.5. 教程：使椭圆移动经过屏幕 .....	199
11.6. 教程：向数据池项添加语言依赖项文本 .....	202
11.7. 教程：使用 3D 图形 .....	206
12. 参考 .....	211



12.1. Android 事件 .....	211
12.2. 数据池项 .....	212
12.3. 数据类型 .....	212
12.3.1. 网格 .....	212
12.3.2. 布尔值 .....	212
12.3.3. 颜色 .....	213
12.3.4. 条件脚本 .....	213
12.3.5. 浮点值 .....	213
12.3.6. 字体 .....	214
12.3.7. 图像 .....	214
12.3.8. 整数 .....	214
12.3.9. 列表 .....	215
12.3.10. 字符串 .....	216
12.4. EB GUIDE 脚本 .....	216
12.4.1. EB GUIDE 脚本 关键字 .....	216
12.4.2. EB GUIDE 脚本 运算符优先顺序 .....	217
12.4.3. EB GUIDE 脚本 标准库 .....	218
12.4.3.1. EB GUIDE 脚本 函数 A .....	218
12.4.3.1.1. abs .....	218
12.4.3.1.2. absf .....	218
12.4.3.1.3. acosf .....	219
12.4.3.1.4. animation_before .....	219
12.4.3.1.5. animation_beyond .....	219
12.4.3.1.6. animation_cancel .....	220
12.4.3.1.7. animation_cancel_end .....	220
12.4.3.1.8. animation_cancel_reset .....	220
12.4.3.1.9. animation_pause .....	220
12.4.3.1.10. animation_play .....	221
12.4.3.1.11. animation_reverse .....	221
12.4.3.1.12. animation_running .....	221
12.4.3.1.13. animation_set_time .....	221
12.4.3.1.14. asinf .....	222
12.4.3.1.15. atan2f .....	222
12.4.3.1.16. atan2i .....	222
12.4.3.1.17. atanf .....	223
12.4.3.2. EB GUIDE 脚本 函数 C - H .....	223
12.4.3.2.1. ceil .....	223
12.4.3.2.2. changeDynamicStateMachinePriority .....	223
12.4.3.2.3. character2unicode .....	223
12.4.3.2.4. clearAllDynamicStateMachines .....	224

12.4.3.2.5. color2string .....	224
12.4.3.2.6. cosf .....	224
12.4.3.2.7. deg2rad .....	225
12.4.3.2.8. expf .....	225
12.4.3.2.9. float2string .....	225
12.4.3.2.10. floor .....	225
12.4.3.2.11. focusNext .....	226
12.4.3.2.12. focusPrevious .....	226
12.4.3.2.13. format_float .....	226
12.4.3.2.14. format_int .....	227
12.4.3.2.15. getLineCount .....	228
12.4.3.2.16. getTextHeight .....	228
12.4.3.2.17. getTextLength .....	228
12.4.3.2.18. getTextWidth .....	229
12.4.3.2.19. has_list_window .....	229
12.4.3.2.20. hsba2color .....	229
12.4.3.3. EB GUIDE 脚本 函数 I - R .....	230
12.4.3.3.1. int2float .....	230
12.4.3.3.2. int2string .....	230
12.4.3.3.3. isDynamicStateMachineActive .....	230
12.4.3.3.4. language .....	231
12.4.3.3.5. localtime_day .....	231
12.4.3.3.6. localtime_hour .....	231
12.4.3.3.7. localtime_minute .....	232
12.4.3.3.8. localtime_month .....	232
12.4.3.3.9. localtime_second .....	232
12.4.3.3.10. localtime_weekday .....	232
12.4.3.3.11. localtime_year .....	233
12.4.3.3.12. log10f .....	233
12.4.3.3.13. logf .....	233
12.4.3.3.14. nearbyint .....	233
12.4.3.3.15. popDynamicStateMachine .....	234
12.4.3.3.16. powf .....	234
12.4.3.3.17. pushDynamicStateMachine .....	234
12.4.3.3.18. rad2deg .....	235
12.4.3.3.19. rand .....	235
12.4.3.3.20. shutdown .....	235
12.4.3.3.21. rgba2color .....	235
12.4.3.3.22. round .....	236

12.4.3.4. EB GUIDE 脚本 函数 S - W .....	236
12.4.3.4.1. seed_rand .....	236
12.4.3.4.2. sinf .....	236
12.4.3.4.3. skin .....	237
12.4.3.4.4. sqrtf .....	237
12.4.3.4.5. string2float .....	237
12.4.3.4.6. string2int .....	238
12.4.3.4.7. string2string .....	238
12.4.3.4.8. substring .....	238
12.4.3.4.9. system_time .....	239
12.4.3.4.10. system_time_ms .....	239
12.4.3.4.11. tanf .....	239
12.4.3.4.12. trace_dp .....	240
12.4.3.4.13. trace_string .....	240
12.4.3.4.14. transformToScreenX .....	240
12.4.3.4.15. transformToScreenY .....	241
12.4.3.4.16. transformToWidgetX .....	241
12.4.3.4.17. transformToWidgetY .....	241
12.4.3.4.18. trunc .....	242
12.4.3.4.19. widgetGetChildCount .....	242
12.5. 事件 .....	242
12.6. model.json 配置文件 .....	242
12.6.1. 示例 model.json 位于 EB GUIDE Studio .....	248
12.7. platform.json 配置文件 .....	250
12.7.1. 示例 platform.json 位于 EB GUIDE Studio .....	252
12.8. 场景 .....	253
12.9. EB GUIDE GTF 支持的触摸屏类型 .....	255
12.10. 控件 .....	256
12.10.1. 视图 .....	256
12.10.2. 基本控件 .....	256
12.10.2.1. Alpha 掩码 .....	257
12.10.2.2. 动画 .....	258
12.10.2.2.1. 常量曲线 .....	259
12.10.2.2.2. 快速启动曲线 .....	259
12.10.2.2.3. 慢速启动曲线 .....	260
12.10.2.2.4. 二次曲线 .....	260
12.10.2.2.5. 正弦曲线 .....	261
12.10.2.2.6. 脚本曲线 .....	261
12.10.2.2.7. 线性曲线 .....	262
12.10.2.2.8. 线性插值曲线 .....	262

12.10.2.3. 容器 .....	263
12.10.2.4. 椭圆 .....	263
12.10.2.5. 图像 .....	264
12.10.2.6. 实例化器 .....	264
12.10.2.7. 标签 .....	265
12.10.2.8. 矩形 .....	265
12.10.3. 3D 控件 .....	266
12.10.3.1. 环境光 .....	266
12.10.3.2. 摄像头 .....	266
12.10.3.3. 定向光 .....	267
12.10.3.4. 材料 .....	267
12.10.3.5. 网格 .....	267
12.10.3.6. PBR GGX 材料 .....	268
12.10.3.7. PBR Phong 材料 .....	269
12.10.3.8. 点光 .....	269
12.10.3.9. 场景图 .....	270
12.10.3.10. 场景图节点 .....	270
12.10.3.11. 聚光 .....	271
12.11. 控件功能 .....	271
12.11.1. 公共 .....	271
12.11.1.1. 子代可视性选择 .....	271
12.11.1.2. 已启用 .....	272
12.11.1.3. 已聚焦 .....	272
12.11.1.4. 多行 .....	272
12.11.1.5. 已按下 .....	273
12.11.1.6. 已选中 .....	273
12.11.1.7. 选择群组 .....	274
12.11.1.8. 旋转 .....	274
12.11.1.9. 文本切断 .....	275
12.11.1.10. 已触摸 .....	275
12.11.2. 效果 .....	276
12.11.2.1. 边框 .....	276
12.11.2.2. 着色 .....	277
12.11.2.3. 笔划 .....	277
12.11.3. 焦点 .....	278
12.11.3.1. 自动焦点 .....	278
12.11.3.2. 用户定义焦点 .....	278
12.11.4. 操作动作 .....	279
12.11.4.1. 滑动操作动作 .....	279
12.11.4.2. 按住操作动作 .....	280
12.11.4.3. 长按操作动作 .....	280
12.11.4.4. 路径动作 .....	281

12.11.4.4.1. 操作动作 ID .....	281
12.11.4.5. 二指缩放操作动作 .....	282
12.11.4.6. 旋转操作动作 .....	283
12.11.5. 输入处理 .....	284
12.11.5.1. 操作动作 .....	284
12.11.5.2. 按下的键 .....	284
12.11.5.3. 放开的键 .....	285
12.11.5.4. 键状态已变化 .....	285
12.11.5.5. Unicode 键 .....	285
12.11.5.6. 移入 .....	286
12.11.5.7. 移出 .....	286
12.11.5.8. 移过 .....	287
12.11.5.9. 可移动的 .....	287
12.11.5.10. 旋转 .....	288
12.11.5.11. 触摸丢失 .....	288
12.11.5.12. 触摸移动 .....	288
12.11.5.13. 触摸已按下 .....	289
12.11.5.14. 触摸已放开 .....	289
12.11.5.15. 触摸状态已变化 .....	290
12.11.6. 布局 .....	291
12.11.6.1. 绝对布局 .....	291
12.11.6.2. 方框布局 .....	291
12.11.6.3. 流式布局 .....	292
12.11.6.4. 栅格布局 .....	293
12.11.6.5. 布局留白 .....	293
12.11.6.6. 列表布局 .....	294
12.11.6.7. 缩放模式 .....	295
12.11.7. 列表管理 .....	296
12.11.7.1. 行索引 .....	296
12.11.7.2. 列表索引 .....	296
12.11.7.3. 模板索引 .....	296
12.11.7.4. Viewport .....	296
12.11.8. 3D .....	297
12.11.8.1. 摄像头视见区 .....	297
12.11.8.2. 环境纹理 .....	297
12.11.8.3. 弥散纹理 .....	298
12.11.8.4. 发光纹理 .....	299
12.11.8.5. 光地图纹理 .....	300
12.11.8.6. 正常贴图纹理 .....	301
12.11.8.7. 不透明纹理 .....	301
12.11.8.8. 反射纹理 .....	302
12.11.8.9. 镜面纹理 .....	303



12.11.8.10. 色调映射 .....	304
12.11.9. 变换 .....	305
12.11.9.1. 中枢 .....	305
12.11.9.2. 旋转 .....	305
12.11.9.3. 缩放 .....	306
12.11.9.4. 剪切 .....	306
12.11.9.5. 转换 .....	306
13. 安装 EB GUIDE Studio .....	308
13.1. 背景信息 .....	308
13.1.1. 限制 .....	308
13.1.2. 系统要求 .....	308
13.2. 下载 EB GUIDE .....	309
13.3. 安装 EB GUIDE .....	309
13.4. 卸载 EB GUIDE .....	310
术语表 .....	312
索引 .....	315

# 1. 关于本文档

## 1.1. 目标群体：建模员

建模员使用 **EB GUIDE Studio** 创建人机接口 (HMI)。在 **EB GUIDE** 中，HMI 称为 **EB GUIDE** 模型。与应用程序的通信，是通过已确定的事件（使用事件机制）、通过数据池项（使用数据池）以及通过特定于用户的 **EB GUIDE** 脚本 函数进行的。

建模员执行以下任务：

- ▶ 使用控件和视图体系结构指定显示器上显示的图形元素
- ▶ 与设计人员及可用性专家进行沟通，以优化用户界面
- ▶ 使用状态机功能指定何时显示图形元素
- ▶ 定义元素如何对来自设备（如，控制面板或触摸屏）的输入作出反应
- ▶ 定义元素如何从硬件或软件应用程序（提供导航单元类服务）处接收信息
- ▶ 定义模型元素之间以及输入输出设备之间的接口

建模员具备以下方面的渊博知识：

- ▶ **EB GUIDE Studio** 特色
- ▶ UML 状态机概念
- ▶ 域的规范和要求
- ▶ 交换数据和 **EB GUIDE GTF** 通信机制
- ▶ 3D 图形的规范（如果项目中使用了 3D 图形）

## 1.2. 用户文档的结构

信息的结构如下所示：

- ▶ 背景信息  
背景信息向您介绍了特定的主题和重要事实。通过此信息，您可以执行相关指令。
- ▶ 应用程序演示  
这些指令将指导您逐步完成特定任务，并向您说明如何使用 **EB GUIDE**。可以通过标题中的现在分词 (ing) 识别指令，例如 **Starting EB GUIDE Studio**。
- ▶ 教程



教程是应用程序演示的扩展版本。它将指导您完成复制任务。标题以教程：开头，例如教程：创建按钮。

▶ 参考

参考提供了详细的技术参数和表。

▶ 演示

演示可以让您深入了解如何编写应用程序以及交互顺序。演示包含在 EB GUIDE GTF SDK 中。

## 1.3. 印刷和样式约定

本文档中使用的以下象形文字和信号词用于表示重要信息。

信号词警告，是指对成功配置至关重要的信息。

警告



问题来源和种类  
软件可能会发生什么？

问题的后果是什么？

用户如何避免这个问题？

信号词注意表示某一主题的重要信息。

注意



重要信息  
提供有关主题的重要信息。

信号词小贴士提供了有用的提示、技巧和快捷方式。

提示



有用提示  
提供有用提示

在整个文档中，您将看到以“粗体”，斜体或以等宽字体显示的单词和短语。

要确定这些约定的含义，请查阅以下的示例。

所有默认文本使用 **Arial Regular** 字体或 SimSun。

字体	说明	示例
双引号	用于 GUI 元素	从“项目”的下拉列表框中，选择 <b>Project_-A.</b> 。





字体	说明	示例
SimSun	用于文件名称、目录名称和章节名称	将脚本放入 function_name/abcdirectory。
SimSun	用于用户输入，编码和文件目录	<pre>CC_FILES_TO_BUILD =(PROJECT_PATH)/source/ network/can_node.c CC_FILES_TO_BUILD += \$(PROJECT_PATH)/source/network/can_ config.c</pre> <p>该模块调用 BswM_Dcm_RequestSessionMode() 函数。</p> <p>对于项目名称，输入 Project_Test。</p>
方括号 []	表示可选参数；用于具有可选参数的命令语法	insertBefore [<opt>]
花括号 {}	表示必需参数；用于具有必需参数的命令语法	insertBefore {<file>}
三个点 ...	表明有更多的参数；具有多个参数的命令语法	insertBefore [<opt>...]
竖线	表明所有可用参数；您可从中选择一个可用参数的命令语法	allowinvalidmarkup {on off}



此为逐步指令

只要看到带步骤跟踪的栏，就表示您看到的是逐步指令或基本知识。

先决条件：

- 此行列出了指令的必备条件。

步骤 1

用于完成任务的指令。

步骤 2

用于完成任务的指令。

步骤 3

用于完成任务的指令。

## 1.4. 命名约定

在 EB GUIDE 文档 中，使用以下目录名称：

- ▶ EB GUIDE 的安装目录称为 \$GUIDE\_INSTALL\_PATH。

例如：

C:/Program Files/Elektrobit/EB GUIDE Studio 6.6

- ▶ EB GUIDE SDK 平台的目录称为 \$GTF\_INSTALL\_PATH。名称模式为 \$GTF\_INSTALL\_PATH/platform/<platform name>。

例如：

C:/Program Files/Elektrobit/EB GUIDE Studio 6.6/platform/win32

- ▶ 保存 EB GUIDE 项目的目录称为 \$GUIDE\_PROJECT\_PATH。

例如：

C:/Users/[user name]/Documents/EB GUIDE 6.6/projects/

- ▶ 导出 EB GUIDE 模型的目录称作 \$EXPORT\_PATH。

## 1.5. 路径约定

EB GUIDE Studio 在 Windows 10 中支持处理超过 260 个字符的路径名。完整路径名可以超过 260 个字符，但路径中的单个文件名或目录名仍然限制为 248 个字符。

注意



以下系统中的长路径名 **Windows 7**

Windows 7 不支持处理长路径名。要使用长路径名，请在 Windows 10 上运行 EB GUIDE Studio。有关如何在 Windows 10 中启用长路径名的更多信息，请参见 Windows 10 文档。

## 2. 安全并正确使用

### 2.1. 预期用途

- ▶ EB GUIDE Studio 和 EB GUIDE GTF 旨在用于信息娱乐前端单元、组合仪表及所选行业应用的用户界面项目。
- ▶ 主要用例是量产、规范和原型使用，具体取决于许可证的范围。

### 2.2. 可能的误用

#### 警告



#### 可能的滥用和责任

您只能按照与预期用途相符的方式并按照适用的 许可条款和协议中的规定使用本软件。Elektrobit Automotive GmbH 对于任何不符合适用许可条款和协议规定的软件使用方式，不负有任何义务和责任。

- ▶ 不使用 Elektrobit Automotive GmbH 提供的 EB GUIDE product line，在 ISO 26262/A-SIL 所定义的安全相关系统中实现人机接口。
- ▶ EB GUIDE product line 并非用于特定认证的安全相关系统（需要 D0-178B、SIL 或 A-SIL 之类）。

不允许在此类环境中使用 EB GUIDE GTF。如果您不确定特定应用，请通过 [章3](#) 联系 Elektrobit Automotive GmbH 进行说明。



## 3. 支持

可以通过以下方式获取 EB GUIDE 支持。

► 对于社区版本：

请在我们的文章、博客和论坛中寻找综合信息。

► 对于企业版本：

请根据您的支持合同联系我们。

寻求支持时，请提供您的 EB GUIDE 安装版本的版本号。要寻找版本号，请转至项目中心并点击“帮助”。版本号位于对话框的右下角。

## 4. 简介 EB GUIDE

EB GUIDE 在人机接口 (HMI) 的开发过程中，可以为用户提供帮助。EB GUIDE 产品线为图形或语音用户界面提供工具和平台。EB GUIDE 产品线旨在用于信息娱乐系统、组合仪表及所选行业应用的项目中。主要用例包括量产、规格说明和原型开发。

### 4.1. 此 EB GUIDE product line

EB GUIDE product line 包含下列软件部件：

- ▶ EB GUIDE Studio
- ▶ EB GUIDE TF

EB GUIDE Studio 是您电脑上的建模工具。通过使用 EB GUIDE Studio，您可以将整个 HMI 功能作为中央控制元素进行建模，此元素用于提供对函数的用户访问。

EB GUIDE TF 运行 EB GUIDE Studio 中创建的 EB GUIDE 模型。EB GUIDE TF 可用于开发 PC 以及用于不同的嵌入式平台。

使用 EB GUIDE Studio 创建的 EB GUIDE 模型与 EB GUIDE TF 上运行的已导出 EB GUIDE 模型完全相互独立。它们相互进行交互，但不会相互阻止。

### 4.2. EB GUIDE Studio

#### 4.2.1. 对 HMI 行为进行建模

通过在状态机中放置状态并组合多个状态，指定 EB GUIDE 模型的动态行为。

状态机

状态机是一种决定性有限状态自动机，用于说明系统的动态行为。在 EB GUIDE Studio 中，提供了不同类型的状态机，例如触觉状态机。触觉状态机允许指定图形用户界面。

状态

状态由转换进行链接。转换是状态和触发器状态变更之间的连接。

## 4.2.2. 对 HMI 外观进行建模

在 EB GUIDE Studio 中定义 EB GUIDE 模型的图形用户界面和语音用户界面。

### 控件

为了创建图形用户界面，EB GUIDE Studio 提供了控件。控件是用于定义外观的模型元素。它们主要用于显示信息，例如文本标签或图像。控件还允许用户控制系统行为，例如按钮或滑块。多个控件组成一个称为视图的结构。

### 语音控件

为了创建语音用户界面，EB GUIDE Studio 提供了语音控件。语音空间用于指定语音对话的基本部分。语音识别作为用户输入，语音合成作为系统输出。提示语音控件，允许对通过文本转语音合成器 (TTS) 播放的文本进行建模。命令语音控件，允许对用于说明语音识别器所理解的内容的语法进行建模。相关语音控件通过模型元素分组在一起。此组称为讲话。

## 4.2.3. 处理数据

HMI 和应用程序之间的通信使用数据池和事件系统实现。

### 数据池

数据池是嵌入式数据库，保存需要显示的所有数据以及所有其他的内部信息。数据池项用于存储和交换数据。

### 事件系统

事件是临时触发器。可以将事件发送给双方，以指示发生了特定情况。

应用程序软件可以通过 API 访问事件和数据池。

## 4.2.4. 模拟 EB GUIDE 模型

在模拟运行期间，您可以使用 EB GUIDE Studio 测试您的 EB GUIDE 模型的功能。通过鼠标点击开始模拟，即可立即体验 EB GUIDE 模型的外观和感觉。

使用鼠标、键盘或触摸屏等输入设备与模拟进行互动。

您还可以使用 EB GUIDE Monitor 控制您的 EB GUIDE 模型，并进行以下操作：

- ▶ 通过更改数据池项目的值，来更改显示的数据
- ▶ 通过触发事件模拟用户输入
- ▶ 跟踪日志中的所有更改
- ▶ 启动脚本

您也可以使用 **EB GUIDE Monitor** 作为独立应用程序。

### 4.2.5. 导出 EB GUIDE 模型

要在目标设备上使用 **EB GUIDE** 模型，您需要从 **EB GUIDE Studio** 中导出 **EB GUIDE** 模型并将其转换为目标设备理解的格式。在导出过程中，所有相关数据将导出为一组 **ASCII** 文件。

## 4.3. EB GUIDE TF

**EB GUIDE TF** 包含运行 **EB GUIDE** 模型时所需的 **GtfStartup** 可执行文件和一组库。

根据在 **EB GUIDE Studio** 中选择的项目类型，您将运行：

- ▶ **EB GUIDE GTF**

**EB GUIDE Graphics Target Framework** 是执行图形 **HMI** 的运行环境。

- ▶ **EB GUIDE STF**

**EB GUIDE Speech Target Framework** 是执行 **HMI** 中的语音功能的运行环境。

**EB GUIDE TF** 的大部分程序代码与平台无关。可以很容易地将代码移植到新系统中。

只需通过交换 **EB GUIDE** 模型文件即可交换完整的 **HMI**。无需重新编译 **EB GUIDE TF**。只需从 **EB GUIDE Studio** 中重新导出更改后的 **EB GUIDE** 模型。

**EB GUIDE TF** 使用下列平台抽象：

- ▶ 操作系统抽象

操作系统 (**OS**) 的平台依赖关系由操作系统抽象层 (**GtfOSAL**) 封装。**EB GUIDE TF** 使用的操作系统功能有很多，如文件系统或 **TCP** 套接字。

- ▶ **GL** 抽象层

图形子系统的平台依赖关系由渲染器封装。**EB GUIDE** 模型包含元素属性，如几何形状和照明。已导出的 **EB GUIDE** 模型中包含的数据将传递给渲染器，进行处理并输出至数字图像。渲染器是硬件上实际图形系统的抽象层。**EB GUIDE TF** 对于不同平台，支持各种渲染器。

- ▶ 音频提取

语音用户界面需要访问音频硬件。音频提取提供对麦克风和扬声器的访问。**EB GUIDE STF** 实现语音识别和文本到语音输出。为了实现此目的，**EB GUIDE STF** 吸收了第三方语音引擎。

## 5. 教程：入门

### 注意



#### 默认窗口布局

本用户手册的所有说明和屏幕截图，都使用默认窗口布局。如果您想按照说明进行操作，建议您将 **EB GUIDE Studio** 窗口设置为默认 布局，方法是选择“布局” > “重置为默认布局”。

以下部分提供了关于使用 **EB GUIDE Studio** 进行 HMI 建模的简短概览。它向您说明了如何启动 **EB GUIDE Studio**、如何创建项目、如何对 **EB GUIDE** 模型的行为和外观进行建模以及如何模拟运行 **EB GUIDE** 模型。

大约持续时间：20 分钟。

### 5.1. 起始 EB GUIDE



#### 起始 EB GUIDE

先决条件：

- **EB GUIDE** 已安装。

#### 步骤 1

在 **Windows** 的“开始”菜单中，点击“所有程序”。

#### 步骤 2

在 "Elektrobit" 菜单中，点击您要启动的版本。

**EB GUIDE Studio** 启动将显示项目中心。



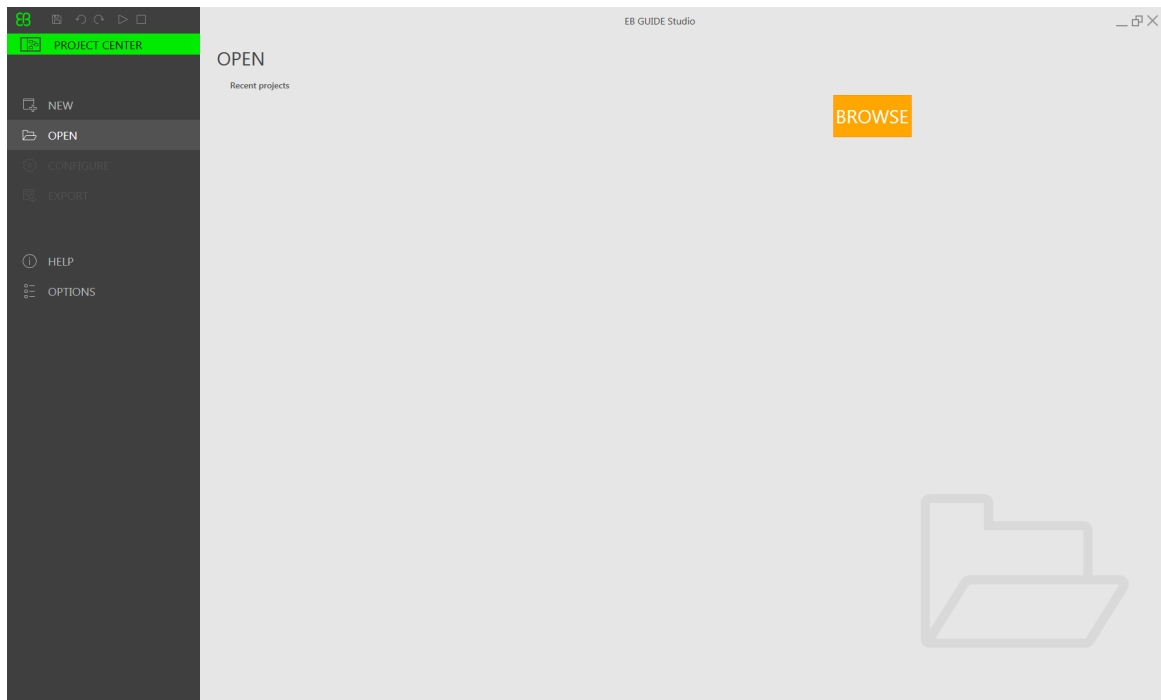


图 5.1. 项目中心

## 5.2. 创建项目



先决条件：

- EB GUIDE Studio 已启动
- 目录 C:/temp 已创建。

### 步骤 1

在项目中心的导航区，点击“新建”。

### 步骤 2

在内容区域中，选中作为“位置”的 C:/temp 目录。

### 步骤 3

输入项目名称 MyProject。

### 步骤 4

点击“创建”。

将创建项目。项目编辑器将打开并显示空白项目。

默认情况下，“主要”状态机已添加，并显示在内容区域中。

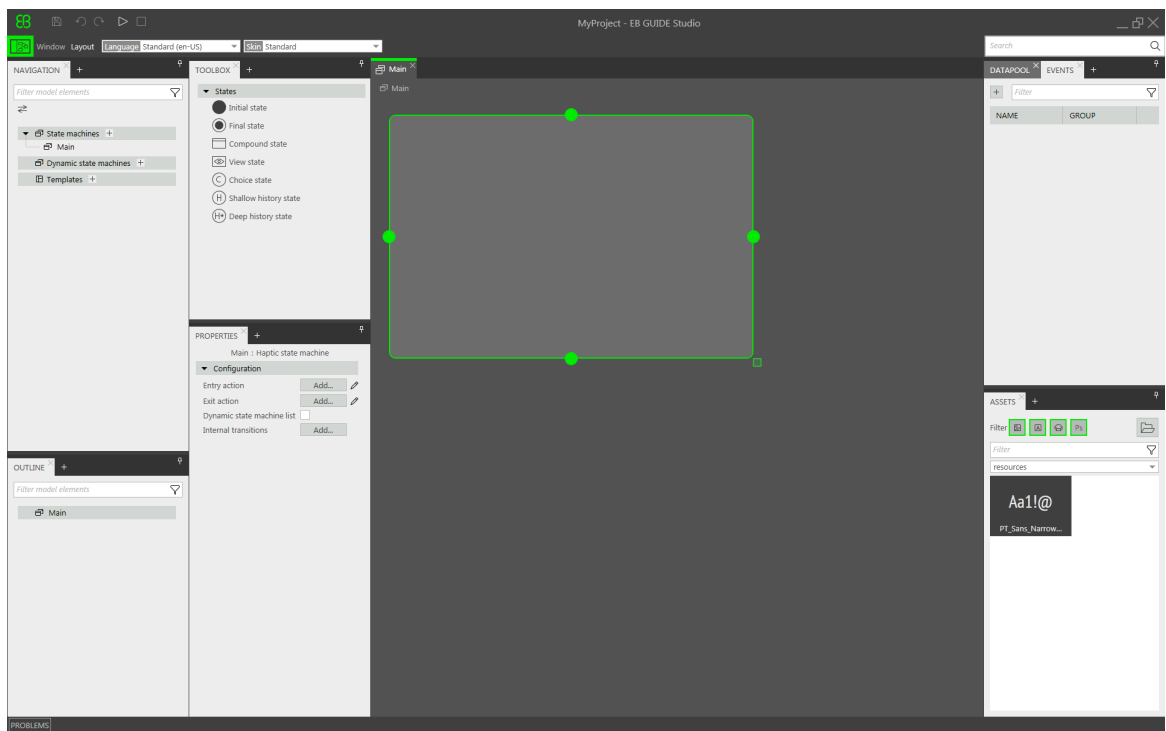


图 5.2. 具有“主要”状态机的项目编辑器

## 5.3. 对 HMI 行为进行建模

EB GUIDE 模型的行为由状态机进行定义。EB GUIDE 使用类似于 UML 的语法来完成此操作。

在以下部分中，您将了解如何对启动时显示已定义的视图并在按下某个按钮时切换为另一个视图的状态机。



### 向状态机添加状态

EB GUIDE 提供了各种状态。以下部分显示了三种不同状态。初始状态定义了状态机的起始点。默认情况下，视图状态显示某个视图。并且状态的结束状态将终止该状态机。

先决条件：

- 将创建项目 MyProject。
- 内容区域将显示“主要”状态机。

#### 步骤 1

将视图状态从“工具箱”拖入状态机中。

视图将随“视图状态 1”添加至 EB GUIDE 模型。

#### 步骤 2

重复步骤 1。

将添加“视图状态 2”。

#### 步骤 3

将初始状态从“工具箱”拖入状态机中。

#### 步骤 4

将结束状态从“工具箱”拖入状态机中。

您添加至“主要”状态机的四种状态，将在内容区域中显示为状态表，并在“导航”元素中显示为分层树视图。

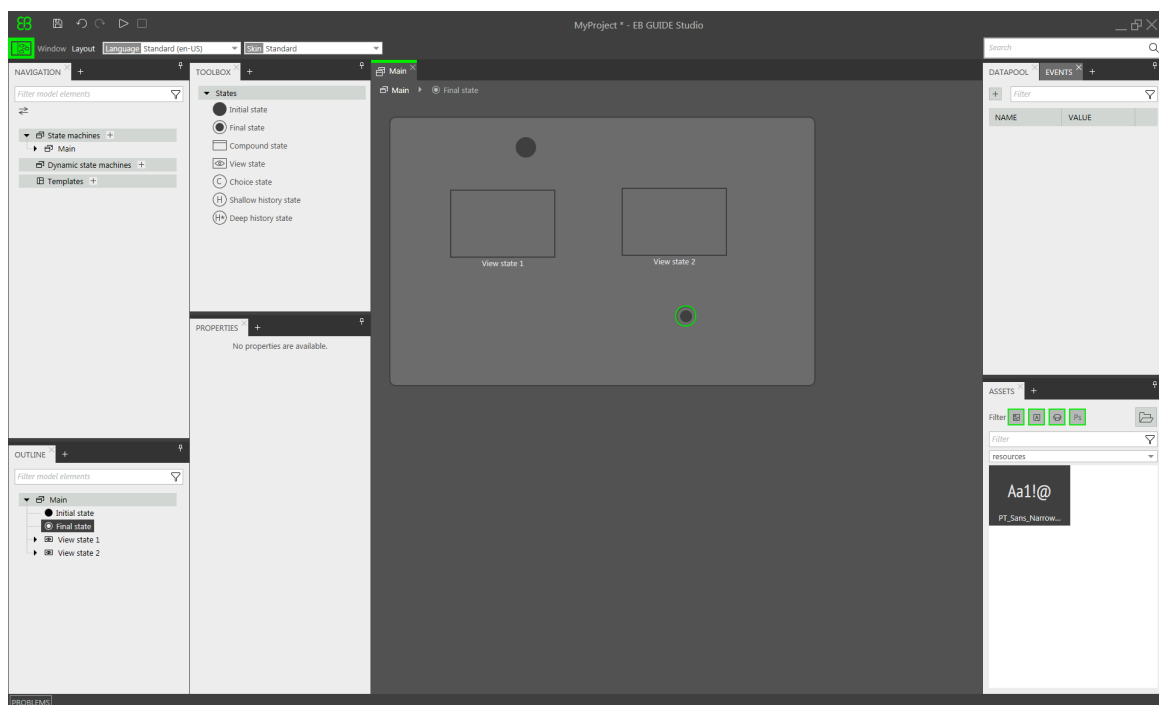


图 5.3. 具有状态的项目编辑器



### 添加转换

转换是状态和触发器状态变更之间的连接。有不同的转换类型。以下部分显示了默认转换以及事件触发的转换。

先决条件：

- 内容区域将显示“主要”状态机。
- “主要”状态机包含一个初始状态、两个视图状态和一个结束状态。

步骤 1

选中初始状态作为转换的源状态。

步骤 2

点击绿色拖动点并按住鼠标键。

步骤 3

将鼠标指针拖入目标状态（即，“视图状态 1”）中。

步骤 4

目标状态以绿色突出显示时，松开鼠标键。

转换将创建并显示为绿色箭头。

步骤 5

在“视图状态 1”和“视图状态 2”之间添加转化。

选择“视图状态 1”并重复步骤 2 - 4。

步骤 6

选择“视图状态 1”和“视图状态 2”之间的转化。

作为下一步，您需要将转换关联至事件。

步骤 7

转至“属性”元素，在“触发器”组合框中，输入 Event 1 并点击“添加事件”。

将创建称为“事件 1”的事件，并添加为转换触发器。无论何时触发“事件 1”，都将执行转化。

步骤 8

在“视图状态 2”和结束状态之间添加转化。

选择“视图状态 2”并重复步骤 2 - 4。

添加新事件“事件 2”作为触发器。

此时，您的状态机将类似于下图：

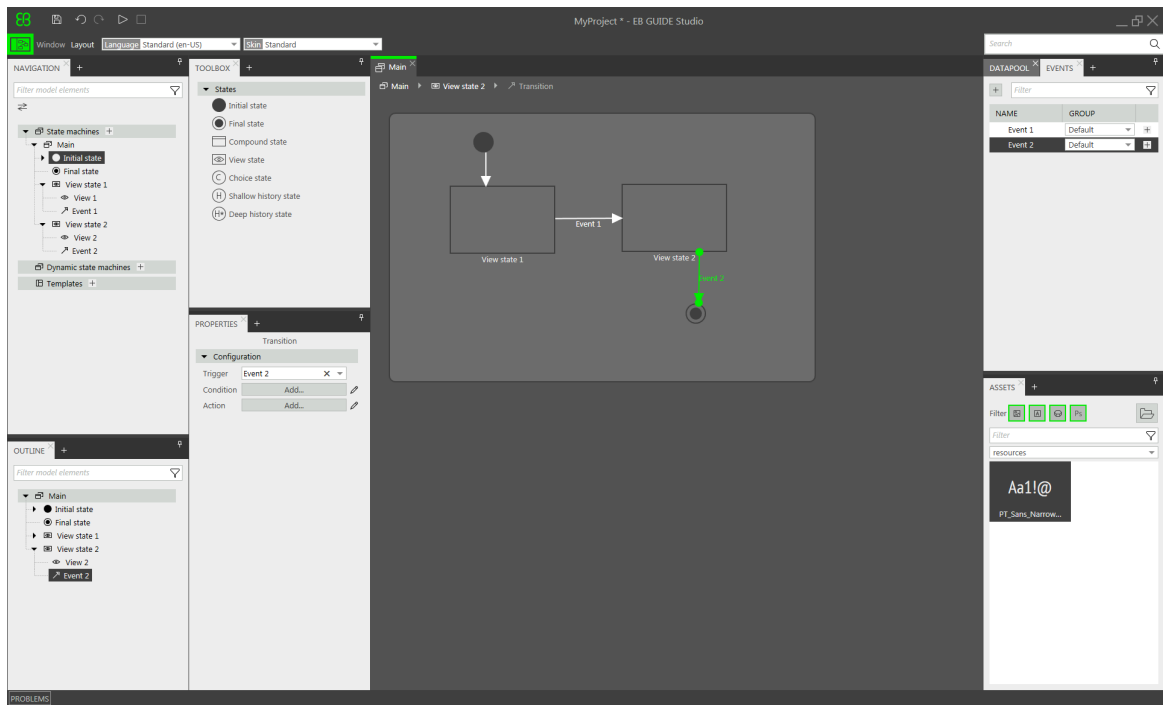


图 5.4. 通过包含事件的转换链接的状态

您已定义基本状态机的行为。

## 5.4. 对 HMI 外观进行建模

您在上述部分中创建的状态机包含两种视图状态。在以下部分中，您将了解如何对视图建模。



### 打开视图

先决条件：

- “视图状态 1”状态机将添加至模型中。

#### 步骤 1

双击“视图状态 1”。

内容区域将显示“视图 1”。



### 向视图添加按钮

通过 EB GUIDE Studio，您可以使用各种选项对视图外观进行建模。



为了提供示例，下一部分向您显示了如何将矩形添加至视图。矩形将对用户输入作出反应并因此用作按钮。

先决条件：

- 内容区域将显示“视图 1”。

#### 步骤 1

将矩形从“工具箱”拖入视图中。

#### 步骤 2

在“属性”元素中，转至“控件功能属性”类别，并点击“添加/删除”。

这将显示“控件功能”对话框。

#### 步骤 3

在“可用控件功能”下，展开“输入处理”类别并选择“已释放的触摸”。

点击“接受”。

相关的控件功能属性，将添加至“属性”元素。

#### 步骤 4

在“属性”元素中，从 touchPolicy 下拉列表框选择 Press then react。

矩形将对触摸输入作出反应。

#### 步骤 5

转至 touchShortReleased 属性，并点击“编辑”。

#### 步骤 6

输入下列 EB GUIDE 脚本：

```
function(v:touchId::int, v:x::int, v:y::int, v:fingerId::int)
{
    fire_delayed 500, ev:"Event 1"()
    true
}
```

如果触摸了矩形，则将在 500 毫秒后触发“事件 1”。

#### 步骤 7

点击“接受”。

#### 步骤 8

在“属性”元素中，对 fillColor 属性，选择红色。

#### 步骤 9

在“导航”元素中，双击“视图 2”。

内容区域将显示“视图 2”。

#### 步骤 10

重复步骤 1 至 5。

#### 步骤 11

输入下列 EB GUIDE 脚本：

```
function(v:touchId::int, v:x::int, v:y::int, v:fingerId::int)
{
    fire_delayed 500, ev:"Event 2"()
    true
}
```

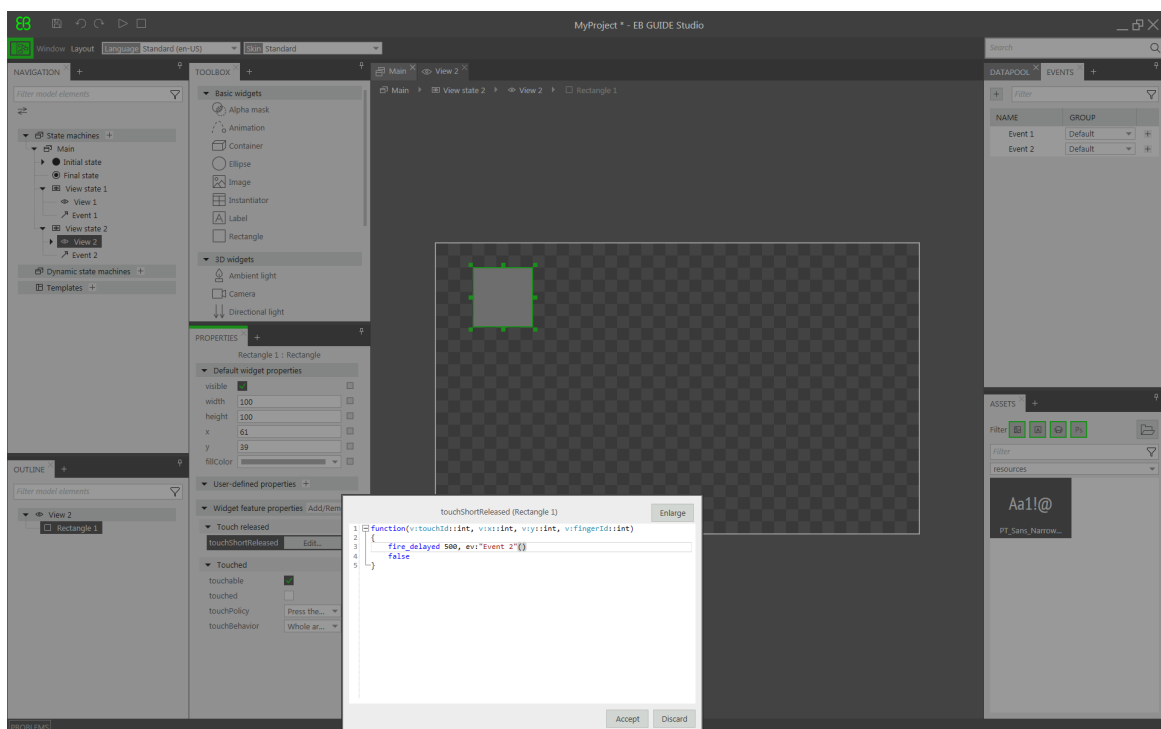


图 5.5. 具有.....的控件属性 EB GUIDE 脚本

#### 步骤 12

点击“接受”。

如果触摸了矩形，将在 500 毫秒后触发“事件 2”。

#### 步骤 13

在“属性”元素中，对 fillColor 属性，选择蓝色。


## 5.5. 启动模拟运行

EB GUIDE 允许您将模型导出至目标设备前，在 PC 上对其进行模拟运行。




#### 启动模拟运行

##### 步骤 1


要保存项目，请点击命令区的 。

##### 步骤 2

在命令区中，请点击 。

EB GUIDE 模型将启动，并显示已建模的行为和外观。

首先，将显示“视图 1”。点击红色矩形会将屏幕更改为“视图 2”。这是因为点击触发了“事件 1”，而“事件 1”将运行从“视图状态 1”到“视图状态 2”的转化。

然后，将显示“视图 2”。在“视图 2”中点击蓝色矩形将终止状态机。这是因为点击触发了“事件 2”，而“事件 2”将运行从“视图状态 2”到结束状态的转化。模拟运行窗口仍保持打开状态。要停止模拟运行，请点击 。



## 6. 背景信息

本章中的主题按字母顺序排序。

注意



默认窗口布局

本用户手册的所有说明和屏幕截图，都使用默认窗口布局。如果您想按照说明进行操作，建议您将 **EB GUIDE Studio** 窗口设置为默认布局，方法是选择“布局” > “重置为默认布局”。

### 6.1. 3D 图形

EB GUIDE Studio 可在 EB GUIDE 项目中使用 3D 图形。

#### 6.1.1. 受支持的 3D 图形格式

只有 OpenGL ES 版本 2.0 或更高版本和 DirectX 11 渲染器才能显示 3D 图形。受支持的 3D 图形格式为 COLLADA (.dae) 和 Filmbox (.fbx)。要获得最佳结果，请使用 Filmbox 格式。

#### 6.1.2. 3D 图形文件的设置

要使 3D 对象显示在 EB GUIDE Studio 的视图中，您需要通过下列选项来创建 3D 图形文件：

- ▶ 透视摄像头
- ▶ 至少一个对象（包含网格和至少一种材料）
- ▶ 至少一个光源

要创建 3D 图形文件，请使用第三方 3D 建模软件。

3D 图形支持下面列出的各种附加内容：

- ▶ 3D 对象及位置、法线、副法线、切线和一个纹理通道
- ▶ 定向光源
- ▶ 环境光源
- ▶ 点光源及常量、线性、二次和立方衰减
- ▶ 聚光光源及锥角、常量、线性、二次和立方衰减
- ▶ 对视野、近平面和远平面的透视摄像头支持

► 纹理：放射、漫射、高光、法线地图、不透明、反射方形及光地图

提示



设置 **3D** 图形文件

请注意，不透明地图需要有效的 **alpha** 通道。

### 6.1.3. 导入 3D 图形文件

要向视图添加 3D 图形，需要使用场景图形导入 3D 图形文件。在导入期间 EB GUIDE Studio 将 3D 图形文件转换为具有场景图作为父节点的控件树。对于 3D 图形文件的内容，例如摄像头、材料和网格，EB GUIDE Studio 创建相应的控件。如果导入的 3D 图形文件的 3D 场景中包含动画，EB GUIDE Studio 将使用线性键值插值整数曲线和线性键值插值浮点曲线来导入这些动画。

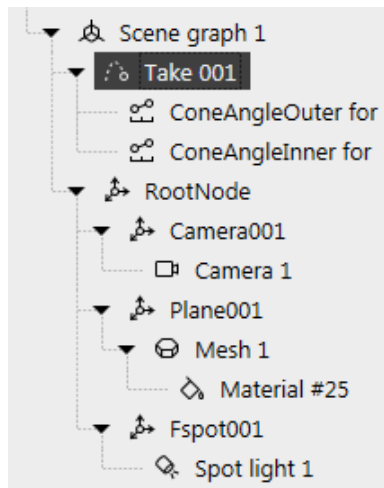


图 6.1. “导航”元素中显示的场景图示例

注意



限制

在 EB GUIDE Studio 中，每个网格只允许一种材料。如果 3D 图形的每个网格具有多种材料，则在导入期间 EB GUIDE Studio 为每种其他材料创建其他网格。

在导入 .fbx 文件过程中，仅创建一个默认材料控件。如果您的 3D 模型具有其他类型的材料，EB GUIDE Studio 仅添加一个默认材料，并且其属性将设置为默认值。在 EB GUIDE Studio 中，您可以使用 PBR Phong 材料和 PBR GGX 材料控件添加其他类型的材料。

导入 3D 图形文件后，在文件夹 \$GUIDE\_PROJECT\_PATH/<project name>/resources 中创建子目录。子目录根据导入的 .fbx 文件命名。此外，创建的日期和时间添加到子目录的名称。



例 6.1.

#### 导入目录的命名

3D 图形文件称为 car.fbx。在 EB GUIDE Studio 中导入 3D 图形文件后，您可以在 \$GUIDE\_PROJECT\_PATH/<project name>/resources 中找到名为 car\_20160102\_103029 的子目录。

子目录中包含以下要素：

- ▶ 网格作为 .ebmesh 文件
- ▶ 纹理作为 .png 或 .jpg 文件

要为您的 3D 图形使用更多纹理，请将纹理复制到 \$GUIDE\_PROJECT\_PATH/<project name>/resources。作为纹理使用 .png 或 .jpg 图像。

可以在一个场景图中导入多个 3D 图形。

导入后，您可以添加、修改或删除 3D 控件。

有关详情，请参见[节6.18](#)、[节12.10.3](#)和[节12.11.8](#)。

有关说明，请参见[节8.1.3.1](#)和[节11.7](#)。

## 6.2. 动画

动画给您的 EB GUIDE 模型带来运动效果和视觉效果。在 EB GUIDE 中，您可以对不同用例使用动画。您可以在一个视图中给控件制作动画，也可以制作一个动图到另一个动图的转变动画。

### 6.2.1. 控件动画

给控件制作动画是指沿着视图移动控件。动作由曲线定义。因此，“工具箱”中的“基本控件”类别包括名为动画的控件。对于每个动画，您可以添加一组曲线，例如常量曲线、线性插值曲线或正弦曲线。曲线具有 target 控件属性，并且描述了基于时间的 target 属性更改。

每个动画都有一条或多条与之关联的曲线。

此外，给控件制作动画还可以执行以下操作：

- ▶ 移动视图中的控件
- ▶ 更改控件的大小
- ▶ 逐渐更改控件的颜色

动画受 EB GUIDE 脚本 函数 f:animation\_play、f:animation\_pause、f:animation\_cancel 等控制。

提示



并发动画

在 EB GUIDE 中，动画是并发动画并且曲线将并行运行。这意味着，如果多个动画的曲线使用同一控件属性作为目标，则这些曲线将并行覆盖此 target 属性的值。

有关动画和曲线属性，请参见 [节12.10.2.2](#)。

有关说明，请参见 [节8.1.2.7](#)。

## 6.2.2. 视图转变的动画

给视图转变制作动画是指为进入或退出视图定义移动动画或消失动画。视图更改引发此类动画。

您可以为视图模板定义视图切换动画。每次重新使用视图模板后，实例将继承入口动画和退出动画。

视图切换动画有多种类型。例如，入口动画即从右侧移入或从底部移入。例如，退出动画即从顶部移出到底部。

有关视图模板的动画属性，请参见 [节12.10.1](#)。

有关说明，请参见 [节8.6](#)。

## 6.3. 应用程序和模型之间的应用程序编程接口

EB GUIDE 抽取应用程序与 EB GUIDE TF 在应用程序编程接口 (API) 中的所有通信数据。例如，应用程序可以为媒体播放器或导航。

API 由数据池项和事件定义。事件在 HMI 和应用程序之间进行发送。



### 例 6.2. API 的内容

- ▶ 发送到应用程序并包含应播放的轨道数的参数 track 的事件 START\_TRACK
- ▶ 已播放的轨道结束时从应用程序发送到 HMI 的事件 TRACK\_STOPPED
- ▶ 由应用程序写入的动态数据池项 MEDIA\_CURRENT\_TRACK
- ▶ 用于定义播放速度并由用户在 HMI 中设置的动态数据池项 MEDIA\_PLAY\_SPEED

## 6.4. 通信环境

通信环境说明了进行通信的环境。通信环境的示例为与 HMI 模型进行通信的媒体或导航应用程序。某个通信环境执行的更改对其他通信环境不可见，直到这些更改由写入器应用程序发布，并由读取器应用程序更新为止。

在项目配置中，通信环境由唯一名称及数字 ID (0...255) 标识。

有关说明，请参见[节9.8](#)。

## 6.5. 图形用户界面元素

EB GUIDE Studio 图形用户界面分为两个元素：项目中心和项目编辑器。在项目中心，您可以管理 EB GUIDE 项目，配置选项，以及导出 EB GUIDE 模型以复制到目标设备。在项目编辑器中，您可以对 HMI 外观和行为进行建模。

### 6.5.1. 项目中心

项目中心是启动 EB GUIDE Studio 后显示的第一个屏幕。所有与项目相关的功能位于项目中心内。项目中心包含两个部分：导航区和内容区域。

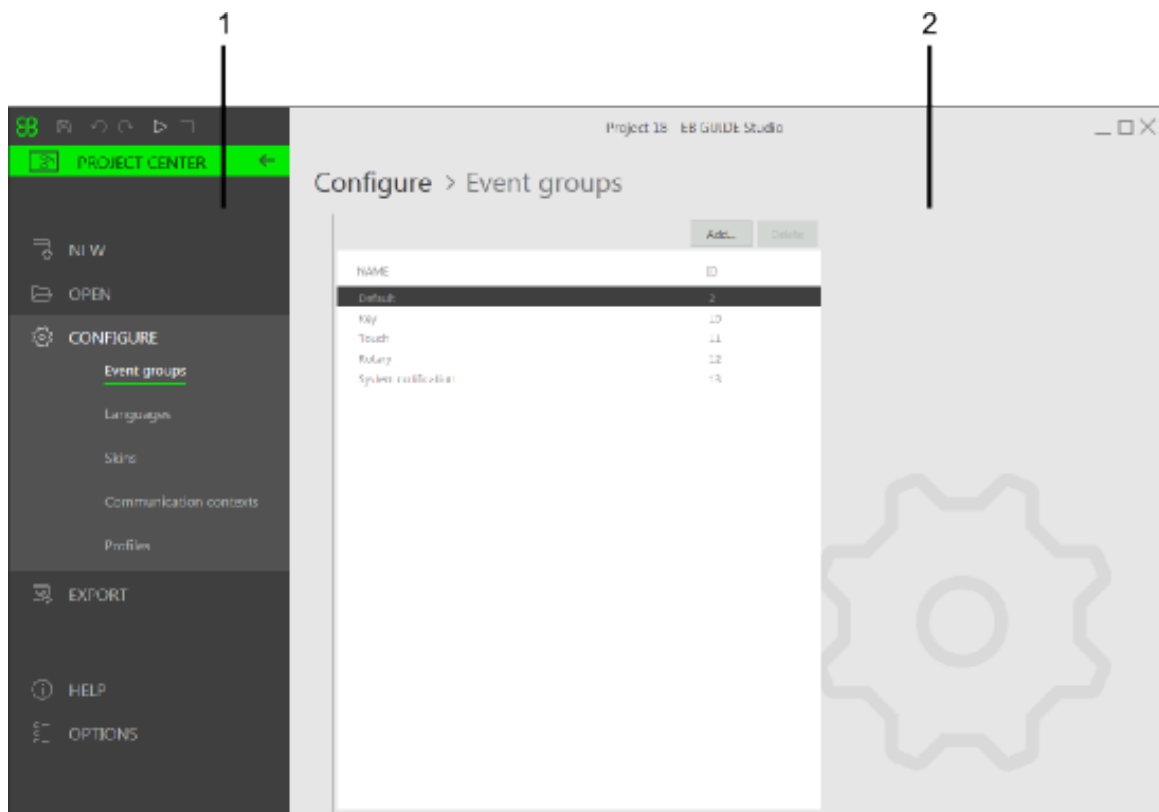


图 6.2. 包含导航区 (1) 和内容区域 (2) 的项目中心

### 6.5.1.1. 导航区

项目中心的导航区包含一些功能标签，例如“配置”或“导出”。您在导航区中点击某个标签，而内容区域将显示相应的功能和设置。

### 6.5.1.2. 内容区域

项目中心的内容区域是执行项目管理和配置的位置。例如，选择某个目录保存项目或定义 EB GUIDE 模型的启动行为。内容区域的外观依赖于在导航区中选中的标签。

## 6.5.2. 项目编辑器

创建项目后，将显示项目编辑器。在项目编辑器中，您可以对 HMI 的行为和外观进行建模：对状态机进行建模，创建视图并管理事件和数据池。项目编辑器包含以下区域和元素。项目编辑器的所有元素可以停靠或浮动，并放置在内容区域之外项目编辑器的任何位置。

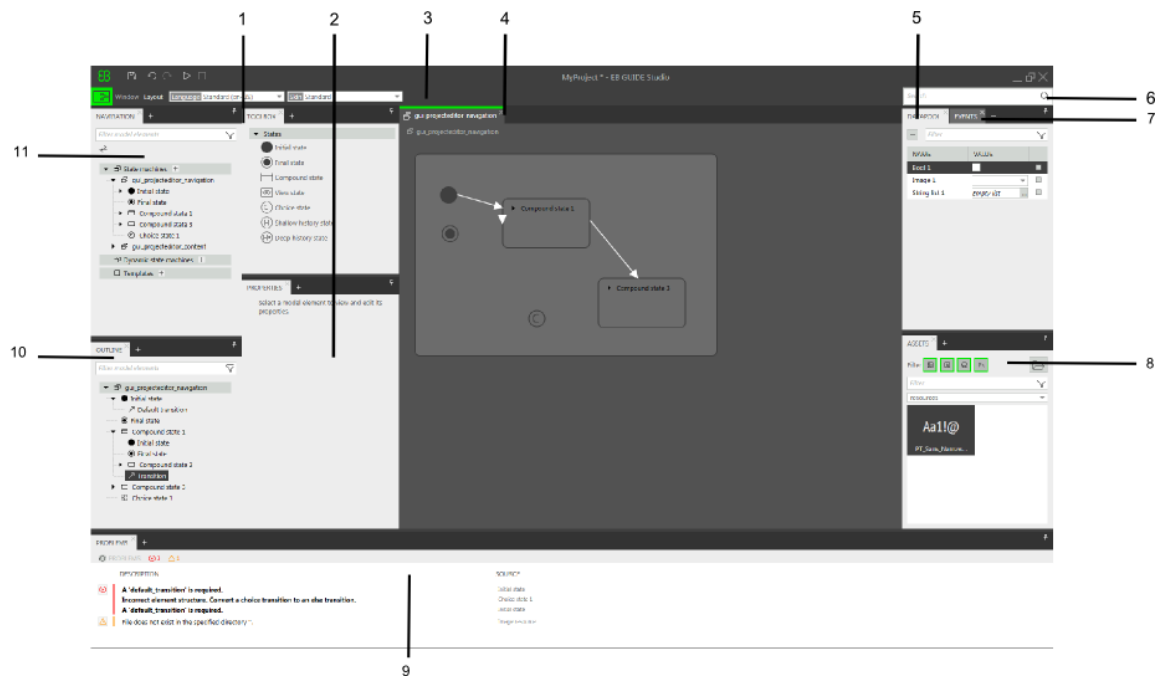


图 6.3. 项目编辑器及其区域和元素

1 “工具箱”元素

2 “属性”元素

3 命令区



4 内容区域

5 “数据池”元素

6 搜索框

7 “事件”元素

8 “资产”元素

9 “问题”元素

10 “大纲”元素

11 “导航”元素

### 6.5.2.1. 导航元素

“导航”元素将模型元素（如状态，视图，动画和 EB GUIDE 模型转化）显示为分层结构，并允许您浏览到任何元素。双击某个模型元素后，将在内容区域中显示该模型元素。

“导航”元素提供了关于 EB GUIDE 模型的所有图形和非图形元素的概览，并反映了状态机层次结构。

还可以在您的 EB GUIDE 模型中添加元素，例如状态机、动态状态机和模板。您可以使用拖放操作添加“工具箱”中的元素，如控件和动画。

在顶部，您可以找到能用于搜索导航区任何模型元素的筛选框。

点击“导航”元素中的模型元素，并按 F3 键开始参考搜索：将打开搜索结果窗口，并列出 EB GUIDE 模型中出现的所有选择的模型元素。

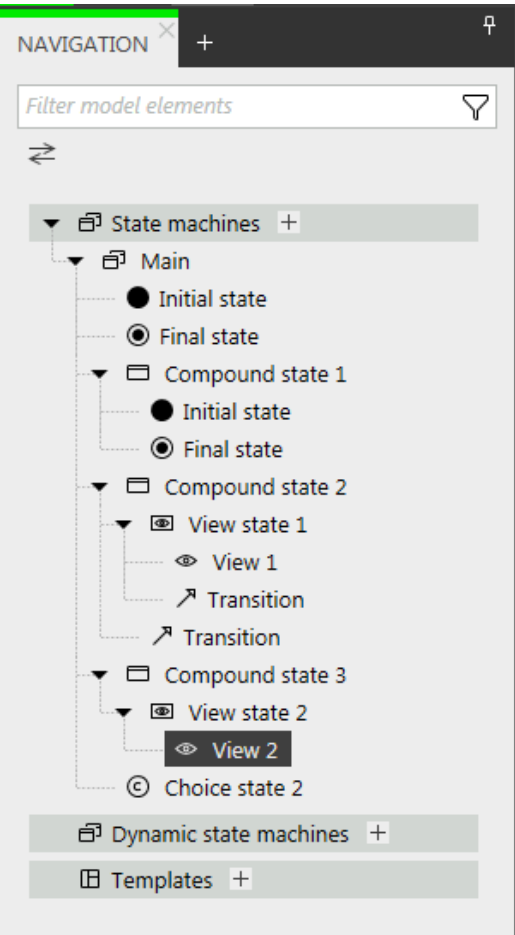


图 6.4. 项目编辑器中的“导航”元素

### 6.5.2.2. “概要”元素

仅显示“导航”元素中所选树部分的结构和模型元素，或者在内容区域显示的编辑器元素。

注意



筛选框

在组件顶部，您会找到用于搜索组件内任何元素的筛选框。

单击组件中的元素，并按 **F3** 键开始引用搜索：这样将打开 搜索结果窗口，并列出 **EB GUIDE** 模型中出现的所有选择的模型元素。

### 6.5.2.3. 工具箱元素

“工具箱”元素提供了进行建模所需的所有工具，也被称为“工具箱”。根据内容区域中显示的元素，“工具箱”提供一组不同的工具，可以拖动到内容区域或“导航”元素中。例如，“工具箱”可以包含以下内容：



- ▶ 如果内容区域显示某个状态机，则“工具箱”包含您可以添加至状态机的状态。
- ▶ 如果内容区域显示某个视图，则“工具箱”包含您可以在视图中排列的控件。
- ▶ 如果内容区域显示脚本化的值属性，则“工具箱”包含您可以插入的 EB GUIDE 脚本 函数。

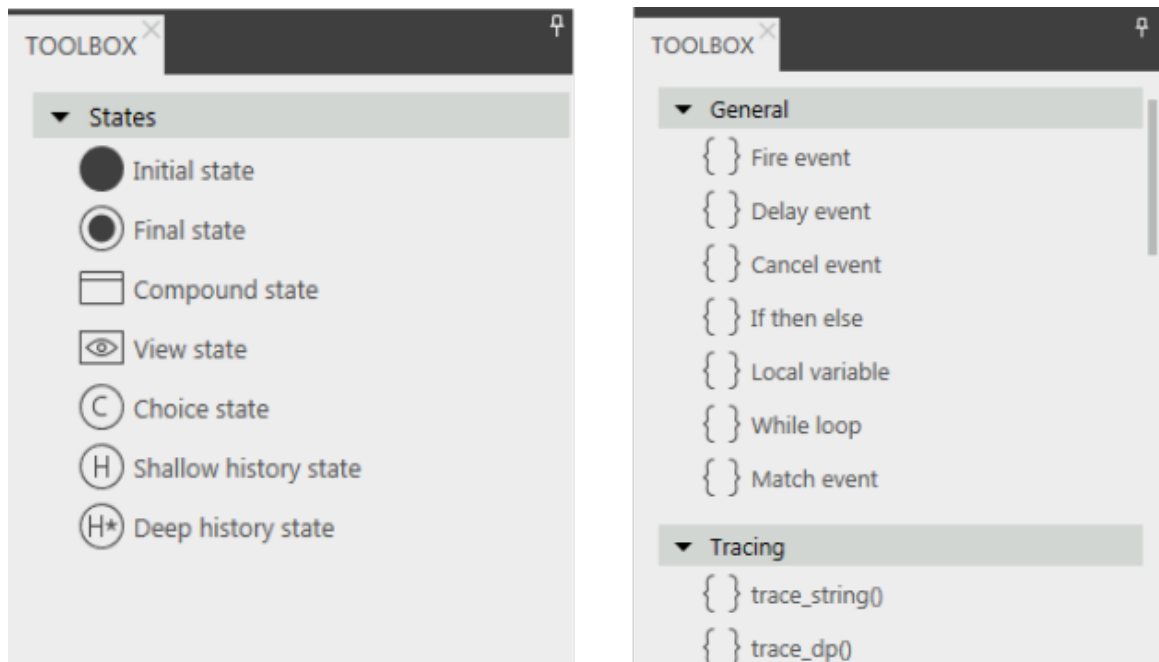


图 6.5. 项目编辑器中的工具箱

#### 6.5.2.4. 属性元素

“属性”元素显示所选模型元素的属性，例如控件或状态的属性。属性以类别分组，且可在“属性”元素中编辑。

点击属性，并按 **F3** 启动参考搜索：将打开搜索结果窗口，并列出 EB GUIDE 模型中出现的所有选择的属性。

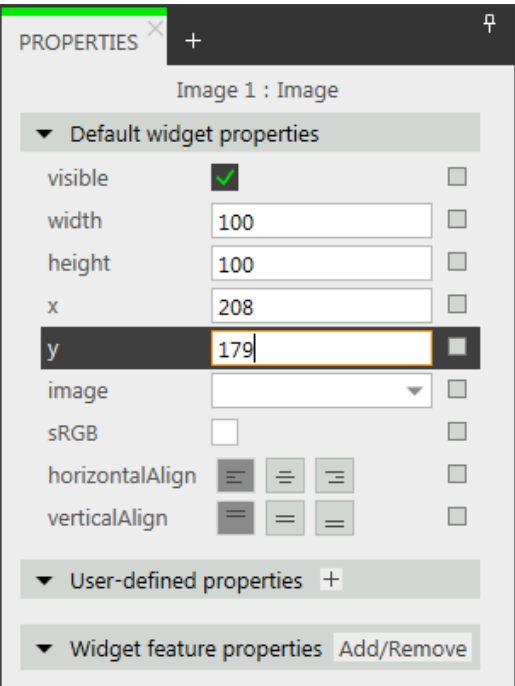


图 6.6. 显示控件属性的“属性”元素

### 6.5.2.5. 内容区域

内容区域中显示的内容，取决于“导航”元素中的所选内容。要编辑某个模型元素，请在“导航”元素中双击该模型元素，它将显示在内容区域中。例如，对状态机的状态进行建模，在视图中排列控件，或者在内容区域中编辑 EB GUIDE 脚本。

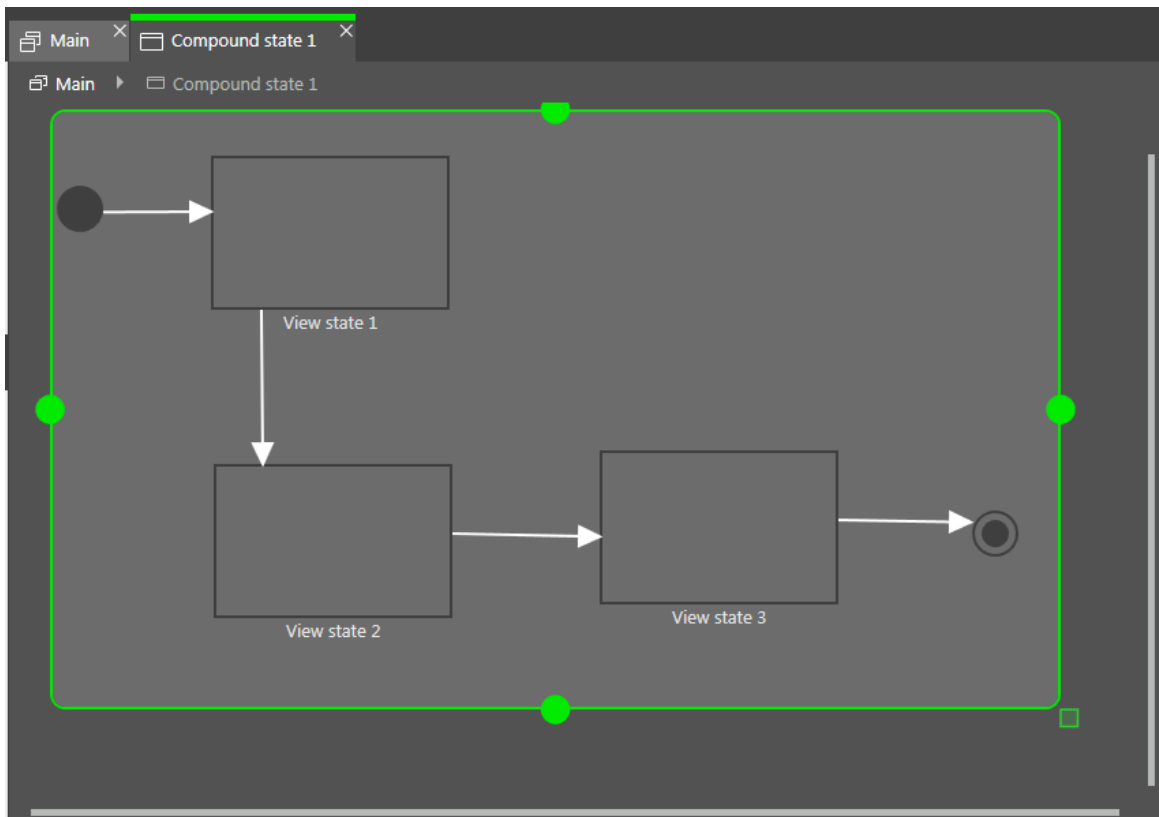


图 6.7. 项目编辑器中的内容区域

在内容区域中，如果您有已打开的视图并且该视图包含动画，则“动画”编辑器将打开。在“动画”编辑器中，您可以向控件属性添加曲线。您还可以通过移动预览中的手柄来编辑曲线的 delay 和 duration 属性。

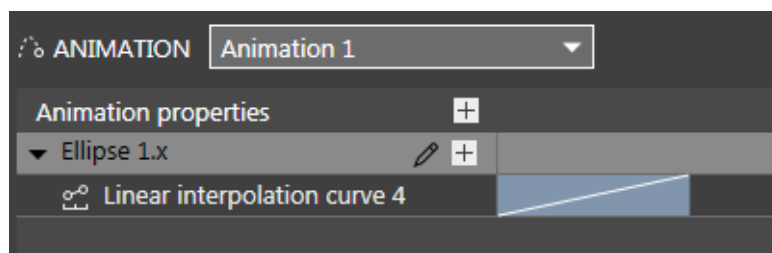


图 6.8. 动画编辑器

要开始参考搜索，请点击内容区域中的状态或控件并按 **F3** 键。搜索结果窗口将打开，并列出 EB GUIDE 模型中所选状态或控件的所有实例。

#### 6.5.2.6. “事件”元素

这里，您可添加时间到您的模型中，编辑事件中列表的属性，如“名称”、“分组”、“类型”和“参数名称”。

注意



筛选框

在组件顶部，您会找到用于搜索组件内任何元素的筛选框。

单击组件中的元素，并按 **F3** 键开始引用搜索：这样将打开 搜索结果窗口，并列出 **EB GUIDE** 模型中出现的所有选择的模型元素。

#### 6.5.2.7. “数据池”元素

这里，您可以添加“数据池”项和编辑属性，如“名称”和“值”。您还可以添加到数据池项目的链接，将值转换为脚本，并添加语言和外观支持。

注意



筛选框

在组件顶部，您会找到用于搜索组件内任何元素的筛选框。

单击组件中的元素，并按 **F3** 键开始引用搜索：这样将打开 搜索结果窗口，并列出 **EB GUIDE** 模型中出现的所有选择的模型元素。

#### 6.5.2.8. “资产”元素

这里，您可以添加资源，如图像、字体、.ebmesh 和 .psd 文件。位于 `$GUIDE_PROJECT_PATH/<project name>/resources` 目录及其子目录中的所有资源文件显示在元素的预览区域中。您可以使用拖放操作将资源添加到模型中。

注意




筛选框

在组件顶部，您会找到用于搜索组件内任何元素的筛选框。

单击组件中的元素，并按 **F3** 键开始引用搜索：这样将打开 搜索结果窗口，并列出 **EB GUIDE** 模型中出现的所有选择的模型元素。

#### 6.5.2.9. 命令区

在命令区中，您可以找到：

- ▶  按钮，可以打开项目编辑器
- ▶ 搜索框搜索模型的元素并跳转到
- ▶ 详细菜单

搜索框

可以使用搜索框找到模型元素。按以下步骤使用搜索框：



- ▶ 点击搜索框或使用 **Ctrl+F** 快捷键跳至搜索框。输入要搜索的模型元素的名称。
- ▶ 通过在命中列表中双击跳至模型元素。

搜索结果窗口的左侧部分列出了按类别分组的模型元素。使用上面的筛选按钮显示或隐藏类别。选择模型元素以获得预览或以只读模式查看模型元素的属性。

当关闭搜索结果窗口时，将保存最后一个搜索项、筛选设置和相应的命中列表，并在再次打开搜索结果窗口时显示。当模型元素在其间改变时，需要再次执行搜索。

搜索不区分大小写。

使用星号 \* 进行通配符搜索时，以下规则适用于：

- ▶ 搜索条目 **t** 返回所有包含 **t** 的元素名称。
- ▶ 搜索条目 **\*t** 返回所有名称以 **t** 结尾的元素。
- ▶ 搜索条目 **t\*** 返回所有名称以 **t** 开头的元素。

您可以搜索以下模型元素类别。

表 6.1. 搜索框中的类别

类别	说明
状态	命中列表还显示找到的状态的子代元素。
视图	命中列表还显示找到的视图的子代元素。
模板	命中列表还显示找到的模板的子代元素。
事件	预览显示事件的属性。
数据池项	预览将显示数据池项的属性。
脚本	预览显示包含文本的脚本的内容。找到的文本将突出显示。
属性	预览显示属性所属的控件。

### 6.5.2.10. 问题元素

在“问题”元素中，您可以检查模型是否有效。它显示当前打开的 **EB GUIDE** 模型，可能的错误和警告。要直接跳转到出现问题的部分，请双击描述。

### 6.5.3. 可对接元素

您可以将项目的所有元素对接为标签，或者取消对接为浮动元素。您可以将元素作为浮动元素拖动到项目中心的任何部分，但内容区域除外。

对接控制的箭头可帮助您选择对接位置，并且实时预览会显示布局的外观。

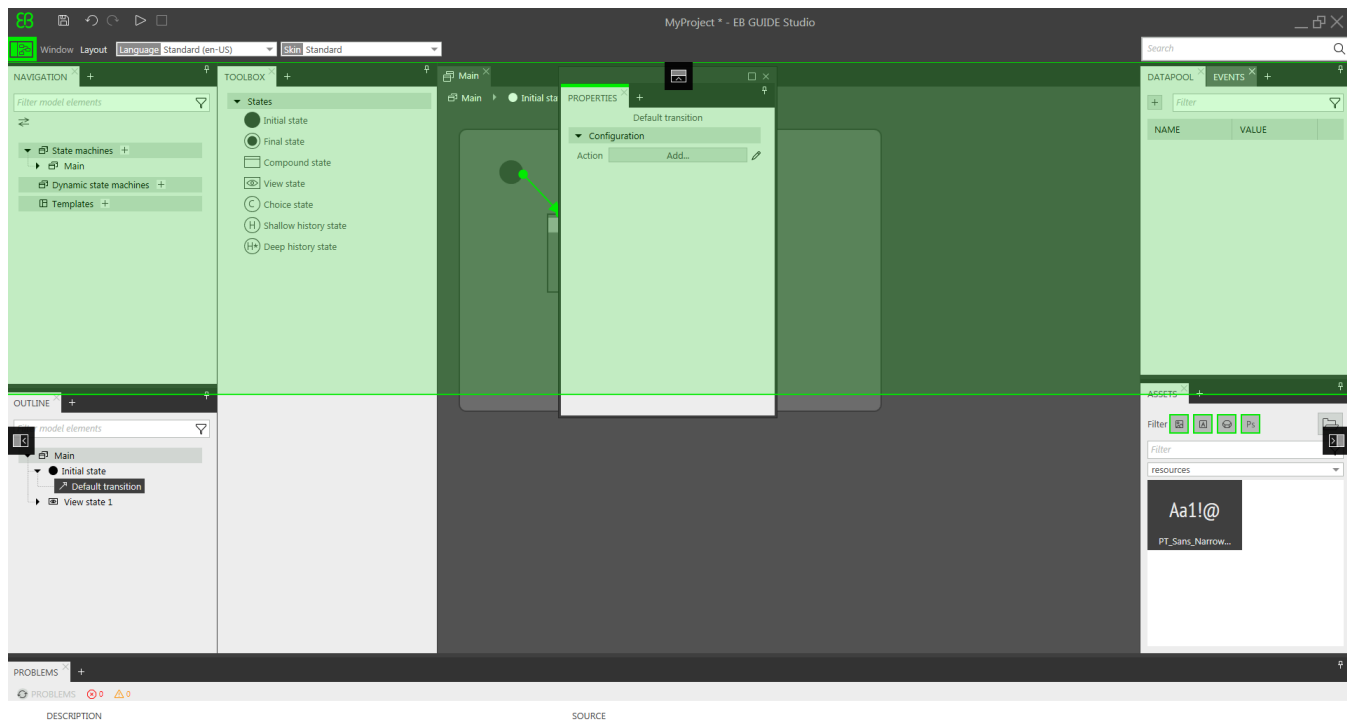


图 6.9. 对接控制和实时预览

注意



默认布局

要重置默认布局，请转至命令区并选择“布局”>“重置默认布局”。

注意



自动隐藏

要在项目编辑器中获取足够控件，您可以隐藏元素。

- ▶ 要隐藏元素或元素组，请点击针形标记。
- ▶ 要显示隐藏的元素，用鼠标在标签上回旋，并再次点击针型标记。

## 6.5.4. EB GUIDE Monitor

EB GUIDE 在模拟运行过程中，提供观察和控制该 EB GUIDE 模型的工具 EB GUIDE Monitor。EB GUIDE Monitor 包括与数据池、事件系统以及 EB GUIDE 模型的状态机通信的机制。

EB GUIDE Monitor 在 EB GUIDE 模拟期间，在 EB GUIDE Studio 中自动启动。您也可以使用 EB GUIDE Monitor 作为独立应用程序。

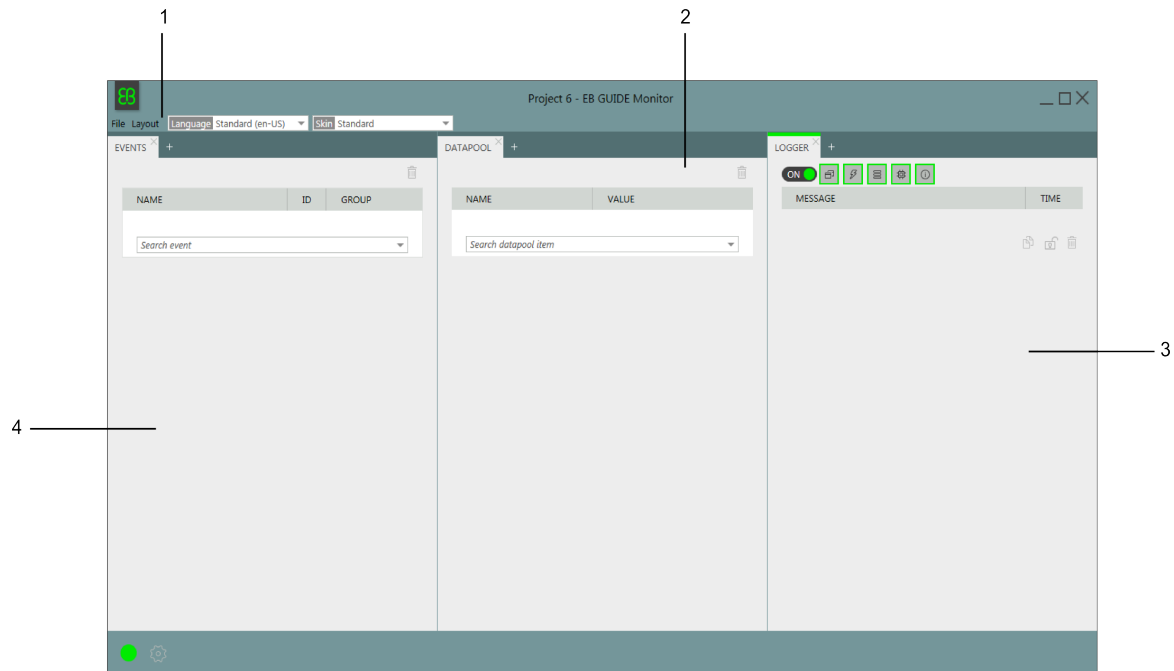


图 6.10. EB GUIDE Monitor 和默认布局

1 “布局”菜单

2 “数据池”元素

3 “记录器”元素

4 “事件”元素

EB GUIDE Monitor 包括以下元素。

- ▶ 在“事件”元素中，您可以搜索和触发事件。如果事件包含参数，您可以更改这些参数，然后触发此事件。
- ▶ 在“数据池”元素中，您可以搜索数据池项并更改它的值。
- ▶ 在“记录器”元素中，将跟踪所有更改、信息消息、错误和警告。在元素顶部，您可以找到用于筛选元素内条目的筛选框。要启用或禁用自动滚动功能，请在元素底部选择或取消选择自动滚动复选框。
- ▶ 在“脚本”元素中，您可以启动脚本并查看输出脚本消息。要注意的是，“脚本”元素不在默认布局中。要添加元素，请点击“布局”>“脚本”。
- ▶ 在“状态机”元素中，将显示当前活动的状态和状态机。请注意，“状态机”元素不在默认布局中。要添加元素，请点击“布局”>“状态机”。

根据项目的需要，您可以重新组织元素并添加新元素。也可以在 EB GUIDE Monitor 窗口内对接和取消对接元素。

在 EB GUIDE Monitor 窗口的左下角，找到连接状态的以下按钮。

按钮	状态
	EB GUIDE Monitor 已连接。 如您点击这个按钮，EB GUIDE Monitor 会断开连接。
	EB GUIDE Monitor 断开连接。 如果您点击此按钮，EB GUIDE Monitor 会进行连接。
	EB GUIDE Monitor 断开连接。 如果点击此按钮，您可以配置 EB GUIDE Monitor 的连接设置。

在命令区，可以用下拉列表框来改变语言和外观。

有关说明，请参见[节10.9](#)。

对于 EB GUIDE Monitor API，请参见 \$GUIDE\_INSTALL\_PATH/doc/monitor/monitor\_api.chm。

## 6.6. 数据池

### 6.6.1. 概念

在执行期间，模型与不同的应用程序进行通信。要启用通信，您的 EB GUIDE 模型必须提供接口。数据池是一个接口，它允许访问数据池项以交换数据。数据池项用于存储值并在 HMI 和应用程序之间进行通信。数据池项是在 EB GUIDE 模型中定义的。

### 6.6.2. 数据池项

数据池项是用于执行下列操作的模型元素：

- ▶ 将数据从应用程序发送到 HMI
- ▶ 将数据从 HMI 发送到应用程序
- ▶ 存储仅在 HMI 或应用程序中使用的数据

有关说明，请参见[节9.5](#)。

要传送通信，您可以使用写入器和读取器程序。

内部通信用于存储数据。用两个不同的应用程序可以建立外部通信。



有关说明，请参见[图9.8](#)。

### 6.6.3. 窗口化列表

EB GUIDE product line 支持窗口列表的概念。窗口化列表操作模式通常用于减少显示大型列表（例如，目录中的所有 MP3 标题）时耗用的内存。这些列表通常由一个应用程序（例如，媒体应用程序）提供，并且仅由另一个应用程序（例如，HMI）不完全显示。

写入器应用程序，定义了多个窗口的虚拟列表长度，这些窗口可能仅包含此列表的一部分。读取器应用程序，仅从窗口覆盖的位置读取数据。从其他位置读取数据将失败。在此用例中，读取器应用程序，必须将当前所需的列表部分告知写入器应用程序。例如，HMI 可以执行用于提供完整列表中当前光标位置的应用程序调用。



#### 例 6.3. 窗口化列表

音频播放器设备的 MP3 标题列表包含 1,000,000 个元素。HMI 必须以并行方式在三个不同的显示屏上显示此列表：信息娱乐系统显示屏、组合仪表显示屏以及平视显示屏。

可以分别控制每个显示屏，它们具有不同的显示行数并且在完整列表中具有不同的光标位置。

只要三个光标之一移动位置，HMI 就会通过异步方式通过事件将新位置发送给媒体应用程序。媒体应用程序提供了一个包含三个窗口的列表。三个窗口中的每个窗口都与三个显示屏之一相关联。光标移动后，窗口更新会稍微延迟。因此，建议使用可以覆盖特定显示屏所显示行周围的窗口位置和窗口大小。

## 6.7. EB GUIDE 模型和 EB GUIDE 项目

EB GUIDE 模型是说明 HMI 的外观和行为的所有元素的总和。它在 EB GUIDE Studio 中进行完整构建。您可以在 PC 上运行 EB GUIDE 模型。

要在目标设备上运行 EB GUIDE 模型，请导出 EB GUIDE 模型并将生成的二进制文件复制到目标设备。

EB GUIDE 项目包含一个 EB GUIDE 模型以及建模所需的各种设置。它包含特定于项目的选项、扩展、资源，并且对于图形项目，它还包含对触觉对话框的说明。

EB GUIDE 项目包含已在 EB GUIDE 模型中配置并链接的对象。这些对象称为 EB GUIDE 模型元素。EB GUIDE 模型元素的示例如下所示：

- ▶ 数据池项
- ▶ 事件
- ▶ 状态
- ▶ 状态机



- ▶ 控件
- ▶ 资源
- ▶ 语言

## 6.8. 事件处理

### 6.8.1. 事件系统

事件系统是用于在应用程序内或应用程序之间进行通信的异步机制。

EB GUIDE 事件系统会严格按事件发送顺序传送所有事件。没有用于将事件传送给不同订户的预定义顺序。

### 6.8.2. 事件

EB GUIDE 中的事件是具有唯一事件 ID 且属于某个事件组的模型元素。此事件 ID 由 EB GUIDE TF 用于发送和接收事件。

介于 0 到 65535 之间的事件组 ID 保留供 EB GUIDE product line 内部使用。例外情况为下表中列出的事件组。

表 6.2. 允许的事件组和 ID

事件群组	识别码
默认值	2
键输入事件	10
触摸输入事件	11
旋转输入事件	12
系统通知事件	13

其余的组 ID 范围可以供特定于客户的应用程序使用。

有关说明，请参见下列内容：

- ▶ [节9.1](#)
- ▶ [节9.3](#)

## 6.9. 扩展



### 6.9.1. EB GUIDE Studio 扩展

EB GUIDE Studio 拓展用于拓展 EB GUIDE Studio，且对所有 EB GUIDE 模型均有效。EB GUIDE Studio 扩展不影响 EB GUIDE GTF。

典型的 EB GUIDE Studio 扩展包括：

- ▶ 附加工具栏按钮
- ▶ 附加数据导出器

### 6.9.2. EB GUIDE GTF 扩展

EB GUIDE GTF 扩展是用于扩展 EB GUIDE GTF，在 EB GUIDE Studio 提供附加特性，但仅对一种 EB GUIDE 模型有效。EB GUIDE GTF 扩展以 EB GUIDE GTF 为基础。

典型的 EB GUIDE GTF 扩展包括：

- ▶ 新控件功能
- ▶ 新 EB GUIDE 脚本 函数

EB GUIDE GTF 扩展是动态链接库 (.dll) 或共享对象 (.so) 文件。

将 EB GUIDE GTF 扩展，包括其第三方库置于以下目录：

`$GUIDE_PROJECT_PATH/<project name>/resources/target`

## 6.10. 语言

### 6.10.1. EB GUIDE 中的显示语言

EB GUIDE Studio 为图形用户界面提供不同的显示语言。您可以在项目中心的“选项”标签中选择显示语言。

有关说明，请参见[节10.6](#)。

### 6.10.2. EB GUIDE 模型中的语言

大多数人机接口可采用用户首选的语言显示文本。此类语言管理也是由 EB GUIDE 提供的。您可以在项目配置中为 EB GUIDE 模型添加语言。

有关说明，请参见[节8.4.1](#)。



注意



无外观支持

为数据池项定义语言支持时，您无法为同一项目添加外观支持。

可以设置数据池项为语言依赖项。数据池项为每种语言定义值。要支持语言，请选择“语言支持”属性。



#### 例 6.4. 语言依赖项文本

在项目配置中，添加了三种语言：英语、德语和法语。数据池项有分别用英语、德语和法语的值 Welcome、Willkommen 和 Bienvenue。

有关说明，请参见 [节11.6](#)。

可以在运行时刻设置已导出的 EB GUIDE 模型的当前语言。

### 6.10.3. 导出和导入语言依赖项文本

使用 EB GUIDE Studio 中的导出和导入功能，可导出、编辑和导入所有语言依赖项文本。您可将文本导出至 .xliff 文件，并将该文件转发给翻译机构。.xliff（XML 本地化交换文件格式）是基于 XML 的格式，用于存储提取的文本，以及将数据从本地化过程中的一个步骤传送到另一个步骤。

完成翻译后，您可在 EB GUIDE Studio 中导入以对应语言显示且经过翻译的 .xliff 文件。

有关说明，请参见 [节10.8](#)。

## 6.11. 外观

外观允许您通过为相同的 EB GUIDE 模型定义不同的数据池值来定义不同的用户界面。这样，您可以为相同的 HMI 定义各种外观，例如夜间和日间模式下的外观。

在运行期间，您可以在外观之间切换，以查看不同数据池值的效果。

外观支持，仅适用于普通数据池值，不能用于脚本值或带链接的数据池项。

注意



无可语言支持

为数据池项目定义外观支持时，您无法向同一项目添加语言支持。

有关说明，请参见 [节8.5](#)。

## 6.12. 资源管理

资源是并非在 EB GUIDE 中创建但项目所需要的内容。在资源目录中查找 EB GUIDE Studio 项目的资源。

资源目录位于 `$GUIDE_PROJECT_PATH/<project name>/resources`。

EB GUIDE 支持以下类型的资源文件：

1. 字体
2. 图像
3. 3D 图形的网格
4. .psd 文件格式

要在项目中使用资源，请将资源文件添加至目录 `$GUIDE_PROJECT_PATH/<project name>/resources`。

### 6.12.1. 字体

要在项目中使用字体，请将字体添加至目录 `$GUIDE_PROJECT_PATH/<project name>/resources`。

受支持的字体类型为 **TrueType** 字体 (\*.ttf、\*.ttc)、OpenType 字体 (\*.otf) 和位图字体 (\*.fnt)。

有关说明，请参见[节8.1.2.4.1](#)。

#### 6.12.1.1. 位图字体

EB GUIDE Studio 在版本 3.0 中支持来自 **Angelcode** 的 \*.fnt 位图字体。要创建位图字体，请使用第三方字体生成器，例如 **Angelcode Bitmap Font Generator**。有关更多信息，请参见 <http://www.angelcode.com>。

确保所生成的字体具有下列设置：

- ▶ 所需字体大小已定义。
- ▶ 字符集为 Unicode。
- ▶ 字体描述符号是二进制的。
- ▶ 纹理以 8 位 .png 文件形式提供。

注意以下几点：

- ▶ 在 EB GUIDE Studio 中，您无法使用标签的 font 属性更改位图的字体大小。这意味着，您需要在生成 .fnt 字体时定义大小。
- ▶ “笔划”控件功能不适用于位图字体。如果您需要字体的特定边框，请在生成 .fnt 字体时定义它。

- ▶ 在 `$GUIDE_PROJECT_PATH/resources` 目录中，为您的 `.fnt` 位图字体和使用第三方工具生成的 `.png` 纹理文件创建子目录。EB GUIDE Studio 期望在 `.fnt` 文件所在的目录中找到 `.png` 文件。

如果您有多个位图字体，请为其中每个字体创建一个子目录。

## 6.12.2. 图像

要在项目中使用图像，请将图像添加至目录 `$GUIDE_PROJECT_PATH/<project name>/resources`。如果您从另一个目录中选择图像，则该图像将复制到上述目录中。

受支持的图像格式为可移植网络图形 (`*.png`)、JPEG (`*.jpg`) 以及 9-patch 图像 (`*.9.png`)。

有关说明，请参见[节8.1.2.3](#)。

### 6.12.2.1. 9-patch 图像

EB GUIDE Studio 通过 9-patch 图像方法，可以支持包含附加元信息的图像。9-Patch 图像是可伸缩的 `.9.png` 图像。9-patch 图像包含两个黑色标记，分别位于图像顶部和左侧。未标记的区域不会进行缩放。已标记区域将进行缩放。标记不会显示在 EB GUIDE Studio 中。

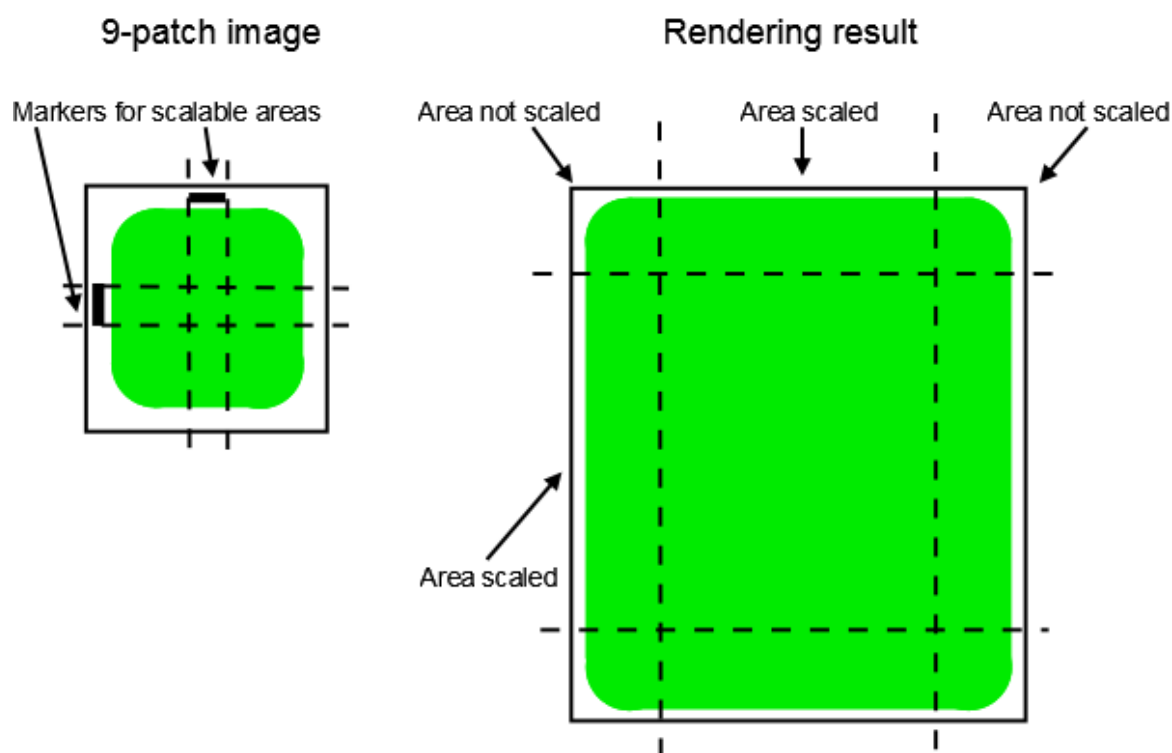


图 6.11. 9-patch 示例



在您处理 9-patch 图像时，请考虑以下几点：

- ▶ 9-patch 处理仅适用于 OpenGL ES 版本 2.0 或更高版本和 DirectX 渲染器。
- ▶ 9-patch 处理仅适用于 .png 图像。
- ▶ 对于 9-patch 图像，\*.9.png 扩展名是必需的。
- ▶ 可以在顶部和左侧指定一个或多个标记，也可以不指定任何标记。9-patch 定义还在图像的右侧和底部包含用于文本区域的标记。这些标记不在 EB GUIDE Studio 中进行求值。

有关说明，请参见[节8.1.2.3](#)。

### 6.12.3. 3D 图形的网格

可以在 EB GUIDE Studio 中导入 3D 图形。在 EB GUIDE Studio 中导入 3D 图形文件后，您可以在 \$GUIDE\_PROJECT\_PATH/<project name>/resources 中找到子目录。在 3D 图形文件中定义的网格导入为 .ebmesh 文件。有关详细信息，请参见[节6.1.3](#)。

有关说明，请参见[节8.1.3.1](#)。

### 6.12.4. .psd 文件格式

EB GUIDE Studio 支持 .psd 文件格式。EB GUIDE Studio 导入 .psd 文件后，会创建一个控件树。控件树由容器和图像组成，这些容器和图像是从 .psd 文件的图层导入过程中创建的。注意以下几点：

- ▶ 如果 .psd 文件中的图层设置为不可见，则相应容器或图像的 visible 属性旁边的复选框将被清除。
- ▶ 如果 .psd 文件中的图层有设置的透明度值，则在导入后将“着色”控件功能添加到相应的图像中。colorationColor 属性的 alpha 通道设置为与 .psd 文件有相同的透明度值。

有关说明，请参见[节8.1.4](#)。

## 6.13. 脚本语言 EB GUIDE 脚本

EB GUIDE 脚本 是 EB GUIDE 的内置脚本语言。本章说明了 EB GUIDE 脚本 语言功能、语法和用法。

### 6.13.1. 应用程序的功能和区域

您可以项目中的各个位置使用 EB GUIDE 脚本，例如：



- ▶ 控件属性中
- ▶ 状态机中，作为转换或状态的组成部分
- ▶ 数据池项中

在所有情况下，只有部分 **EB GUIDE** 脚本 功能可用。例如，只有在脚本是控件的组成部分的情况下，才允许访问局部控件属性。另一方面，始终允许访问数据池。

通过使用 **EB GUIDE** 脚本，您可以直接操作模型元素，例如以执行下列操作：

- ▶ 触发事件
- ▶ 编写数据池项
- ▶ 修改控件属性

## 6.13.2. 命名空间和识别符

在 **EB GUIDE** 中，可以向不同种类的对象提供完全相同的名称。例如，您可以同时将事件和数据池项命名为 Napoleon。**EB GUIDE** 脚本 命名空间可以实现此操作。在 **EB GUIDE** 脚本 中，必须为每个识别符（即，对象的名称）添加命名空间和冒号作为前缀。

命名空间集在 **EB GUIDE** 脚本 中是固定的，您无法引入新的命名空间。存在下列命名空间：

- ▶ ev: 事件
- ▶ dp: 数据池项
- ▶ f: 用户定义的操作（外部函数）
- ▶ v: 局部变量

例如，ev:Napoleon 指定名为 Napoleon 的事件，而 dp:Napoleon 指定名为 Napoleon 的数据池项。

不带命名空间前缀的识别符是字符串常量。

**EB GUIDE** 中的识别符包含许多字符，其中包括空格和标点。因此，在 **EB GUIDE** 脚本 中必须将识别符用引号引起来。如果识别符不包含特殊字符（例如，仅包含字母、数字和下划线的有效 **C** 识别符），则不必用引号将其引起来。



### 例 6.5. **EB GUIDE** 脚本 中的标识符

```
dp:some_text = foo; // foo is a string here
dp:some_text = "foo"; // this statement is identical to the one above
dp:some_text = v:foo; // foo is the name of a local variable
// of course you can quote identifiers, even if it is not strictly necessary
```



```
dp:some_text = v:"foo";
// again, a string constant
dp:some_text = "string with spaces, and -- punctuation!";
// identifiers can also contain special characters, but you have to quote them
dp:some_text = v:"identifier % $ with spaces @ and punctuation!";
```

### 6.13.3. 注释

EB GUIDE 脚本 具有两种注释：C 样式块注释和 C++ 样式行注释。块注释不得进行嵌套。



#### 例 6.6.

**EB GUIDE** 脚本 中的注释

```
/* this is a C style block comment */
// this is a C++ style line comment
```

对于包含字符串“todo”的每条 **EB GUIDE** 脚本 注释，**EB GUIDE Studio** 会在您验证项目时在“问题”元素中显示警告。使用此功能可以标记所有打开的任务并显示其概览。

注意



条件脚本的默认注释

默认情况下，类型为 Conditional script 的数据池项或属性包含以下注释：// todo: auto generated return value, please adapt。要清除警告，请在输入所需 **EB GUIDE** 脚本 代码后从注释中删除 todo 字符串。

### 6.13.4. 类型

**EB GUIDE** 脚本 是强类型和静态类型的编程语言。每个表达式都有意义明确的类型。提供意外类型将导致发生错误。

**EB GUIDE** 脚本 支持以下类型：

- ▶ 整数
- ▶ Unicode 字符串 (string)
- ▶ 带有参考计数的对象
- ▶ 上面列出的类型以及以下对象的类型定义：
  - ▶ 颜色（对于 32 位 RGBA 值为整数）
  - ▶ 布尔值



- ▶ 不同模型元素的 ID：数据池项、视图、状态机、弹出框（所有都具有整数类型）
- ▶ 空，也称为单元类型。此类型在函数式编程（例如 Haskell）中起一定作用。
- ▶ 控件和事件引用。这些是记录类型，您可以使用 dot 标记（C 或 Java 中熟知的标记）来访问这些记录类型的字段。您不能之间创建这里类型的新对象，它们是在适当的情况下自动创建的。

所有类型和类型定义相互都不兼容，并且没有类型转换。在成功编译脚本后，此特性可以确保类型安全。

### 6.13.5. 表达式

EB GUIDE 脚本 基于表达式。每个语言构造都是一个表达式。可以通过将较小的表达式与运算符组合在一起构建较大的表达式。

对表达式进行求值表示将它替换为它的值。



例 6.7.  
对整数值求值

```
1 + 2 // when this expression is evaluated, it yields the integer 3
```

### 6.13.6. 常量和引用

基本表达式是整数、颜色、布尔值和字符串常量以及对模型元素的引用。

空类型也具有值常数，此常数可以通过两种不同的但是语义相同的方法进行编写：

- ▶ 使用后跟右花括号的左花括号 {}
- ▶ 使用关键字 unit



例 6.8.  
常量的用法

```
"hello world" // a string constant
true          // one of the two boolean constants
ev:back       // the event named "back" of type event_id
dp:scrollIndex // the datapool item named "scrollIndex",
               // the type is whichever type the dp item has
5             // integer constants have a dummy type "integer constant"
5::int        // typecast your constants to a concrete type!
color:255,255,255,255 // the color constant for white in RGBA format
```

```
// the following are two ways to express the same
    if( true )
{
}
else
{
}

if( true )
    unit
else
    unit
```

### 6.13.7. 算术和逻辑表达式

EB GUIDE 脚本 支持以下算术表达式：

- ▶ 加 (+)、减 (-)、乘 (\*)、除 (/) 和按模 (%) 可应用于整数类型的表达式。
- ▶ 逻辑运算符或 (||)、与 (&&)、非 (!) 可以应用于布尔值类型的表达式。
- ▶ 可以使用以下的比较运算符，来比较整数和字符串：大于 (>)、小于 (<)、大于或等于 (>=)、小于或等于 (<=)。
- ▶ 可以用相等运算符来比较数据类型：等于 (==) 和 不等于 (!=)。

可以使用相等运算符 (=Aa=) 来比较字符串（不区分大小写）。

注意

相等运算符的适用性



事件和资源数据类型（例如，3D 图形、字体和图像）不支持相等运算符 (==) 和 (!=)。

- ▶ 可以使用 (+) 运算符来连接字符串。



例 6.9.

算术和逻辑表达式

```
10::int + 15::int // arithmetic expression of type int
dp:scrollIndex % 2 // arithmetic expression of type int,
// the concrete type depends on the type
// of dp:scrollIndex
"Morning Star" == "Evening Star" // type bool and value false (wait, what?)
"name" =Aa= "NAME" // type bool and value true
```

```
!true           // type bool, value false
!(0 == 1)       // type bool, value true
// as usual, parenthesis can be used to group expressions
((10 + dp:scrollIndex) >= 50) && (!dp:buttonClicked)
// string concatenation
"Napoleon thinks that " + "the moon is made of green cheese"
f:int2string(dp:speed) + " km/h" // another string concatenation
```

### 6.13.8. 左值和右值

EB GUIDE 脚本 中有两种表达式：l-values 和 r-values。左值具有地址，可以出现在赋值左侧。右值没有地址，并且始终不会出现在赋值左侧。

- ▶ 左值是数据池引用、局部控件属性和局部变量。
- ▶ 右值是事件参数和常量表达式，如字符串或整数常量。

### 6.13.9. 局部变量

let 表达式引入了局部变量。它包含一系列变量声明以及 in 表达式，变量在此表达式中可见。变量是左值，您可以在赋值左侧使用它们。变量具有命名空间 v:。let 表达式的语法如下所示：

```
let v:<identifier> = <expression> ;
  [ v:<identifier> = <expression> ; ]...
in
  <expression>
```

let 表达式的类型和值与 in 表达式的类型和值相同。

let 可以是嵌套表达式，外层 let 表达式的变量在内层表达式中仍然可见。



#### 例 6.10.

let 表达式的用法

```
// assign 5 to the datapool item "Napoleon"
let v:x = 5 in dp:Napoleon = v:x;

// define several variables at once
let v:morning_star = "Venus";
  v:evening_star = "Venus";
in
  v:morning_star == v:evening_star; // Aha!
```

```

let v:x = 5;
    v:y = 20 * dp:foo;
in
{
    // Of course you may have a sequence as the in expression,
    // but parenthesis or braces are required then.
    v:x = v:y * 10;
    dp:foo = v:x;
}
// Because let expression also have types and values, we can have them
// at the right hand side of assignments.
dp:x = let v:sum = dp:x + dp:y + dp:z
        in v:sum; // this is the result
                // of the let expression

// A nested let expression
let v:x = dp:x + dp:y;
v:a = 5;
in
{
    let v:z = v:x + v:a;
    in
    {
        dp:x = v:z;
    }
}

```

### 6.13.10. While 循环

while EB GUIDE 脚本 中循环的语法与 C 或 Java 中的循环类似，它们都包含条件表达式和 do 表达式。语法如下所示：

```
while (<condition expression> ) <do expression>
```

do 表达式重复进行求值，直至条件表达式产生 false 为止。condition expression 必须具有布尔值类型，而 do 表达式必须具有 void 类型。while 表达式的类型为 void 并且不得出现在赋值左侧或右侧。



#### 例 6.11.

while 循环的用法

```

// Assume dp:whaleInSight is of type bool
while( ! dp:whaleInSight )
{

```

```
dp:whaleInSight = f:lookAtHorizon();  
}
```

## 6.13.11. If-then-else

if-then-else EB GUIDE 脚本 中的行为类似于 C 和 Java 中的三元条件运算符 (?:)。

if-then-else 表达式包含以下子表达式:

- ▶ **condition** 表达式
- ▶ **then** 表达式
- ▶ **else** 表达式

语法如下所示:

```
if ( < condition expression> ) <then expression> else <else expression>
```

if-then-else 按照以下方式处理:

1. 首先, 对 **condition** 表达式进行求值。它必须具有布尔值类型。
2. 如果条件为 true, 则对 **then** 表达式进行求值。
3. 如果条件为 false, 则对 **else** 表达式进行求值。

if-then-else 自身是一个表达式。整个表达式的类型是 **then** 表达式和 **else** 表达式的类型, 必须完全相同。根据上述规则, if-then-else 表达式的值为 **then** 表达式的值, 或者为 **else** 表达式的值。

有一种特殊形式的 if-then-else, 在此表达式中, 您可以忽略 **else** 分支。此特殊形式具有类型 void, 并且不得用于从脚本中返回值。



### 例 6.12.

用途 if-then-else

```
// Assume dp:whaleInSight is of type bool  
// and dp:user is of type string.  
if( dp:whaleInSight && dp:user == "Captain Ahab" )  
{  
    dp:mode = "insane";  
}  
else  
{  
    dp:mode = "normal";  
}
```

```
// Because if-then-else is also an expression,
// we may simplify the previous example:
dp:mode = if( dp:whaleInSight && dp:user == "Captain Ahab" )
    "insane"
else
    "normal"

if ( <expression> ) <expression> // This is the reduced way of
    writing if-then-else
//It is an alternative to the following
if( <expression> ) { <expression> ; {} } else {}
```

### 6.13.12. 外部函数调用

您可以使用 **C** 编写的函数（即，所谓的外部函数）来扩展 **EB GUIDE** 脚本。

具有前缀 **f:** 的识别符是外部函数的名称。外部函数具有自变量列表和返回值，和在 **C** 中的用法相同。外部函数调用的语法如下所示：

```
f:<identifier> ( <expression> [ , <expression> ] ... )
```



**例 6.13.**  
调用外部函数

```
// write some text to the connection log
f:trace_string("hello world");
// display dp:some_index as the text of a label
v:this.text = f:int2string(dp:some_index);

// passing different parameters of matching type
f:int2string(v:this.x)
f:int2string(4)
f:int2string(dp:myInt)
f:int2string(v:myVar)

//passing parameters of different types
// starts an animation (parameter type GtfTypeRecord) from a script
// located in its parent widget
f:animation_play(v:this->Animation);

// checks the number of child widgets of a widget (parameter type widget)
f:widgetGetChildCount(v:this);
```

```
// traces debugging information about a datapool item (parameter type dp_id)
// to the connection log; uses the address of the datapool item as parameter
f:trace_dp(&dp:myFlag);
```

### 6.13.13. 数据池访问

以 EB GUIDE 脚本 编写的脚本可以读取和写入数据池项。以命名空间 `dp:` 为前缀的识别符称为数据池项表达式。其类型为 `datapool item of type X`，其中 **X** 是它引用的数据池条目的类型。

如果类型为 **X** 的数据池项出现在赋值左侧，并且类型为 **X** 的表达式出现在赋值右侧，则将写入数据池项的值。

如果数据池项出现在程序中的某个位置，而不是出现在赋值左侧，则将读取数据池项的值。



#### 例 6.14. 数据池值的赋值

```
// Assume intA to be of type int. Assign 10 to it.
dp:intA = 10;
// Assume strA to be of type string. Assign the string "blah" to it.
dp:strA = blah; // Yes, we can omit the quotes, remember?
dp:strA = 42; // Error: integer cannot be assigned to string

// Assign the value of the datapool item intB to intA.
// Both datapool items must have the same type.
dp:intA = dp:intB;
// Multiply the value of intB by two and assign it to intA.
dp:intA = 2 * dp:intB;
// Use the value of a datapool item in an if-clause.
if( dp:speed > 100 )
{
    // ...
}
```

以下运算符可以应用于数据池项：

- ▶ 引用运算符 (**&**) 可应用于数据池项。它引用数据池项的地址，而不是引用其值。在外部函数调用中，引用运算符用于传递类型为 `dp_id` 的参数。
- ▶ 重定向链接运算符 (**=>**) 更改了数据池项的链接目标。链接源只能是已链接的数据池项。

### 6.13.14. 控件属性



如果某个脚本是控件的组成部分，则它可以访问该控件的属性。EB GUIDE 脚本 创建了名为 `v:this` 的变量，用于使用点标记访问属性。

如果某个脚本附加到控件属性，则它是控件的组成部分，例如作为输入反应，如点击或按下按钮。



#### 例 6.15. 设置控件属性

```
// assume this script is part of a widget
v:this.x = 10; // if the widget has an x-coordinate

v:this.text = "hello world"; // if the widget is a label and has a text property
// assume testEvent has one integer parameter
fire ev:testEvent(v:this.x);
```

如果某个脚本是控件的组成部分，则它还可以访问控件树中其他控件的属性。

**go-to** 运算符 (`->`) 用于引用控件树中的其他控件。语法如下所示：

`<expression> -> <expression>`

左侧的表达式必须引用某个控件，而右侧的表达式必须为字符串，即子代控件的名称。要浏览到父代控件，请在右侧使用符号 `^`。整个 **go-to** 表达式引用了某个控件。

浏览控件树可能会影响运行时间性能。控件已指定给变量，以便高效地操作多个属性。



#### 例 6.16. 访问控件属性

```
v:this.x          // access the properties of the current widget
v:this->^.x        // access the x property of the parent widget
v:this->^->caption.text // access the text property of a label called caption,
                      // read: "go-to parent, go-to caption, text"

// Modify several properties of the caption.
// This way, the navigation to the caption is only performed once.
let v:cap = v:this->^->caption
in
{
  v:cap.textColor = color:0,0,0,255;
  v:cap.x += 1;
  v:cap.y += 1;
}
```

## 6.13.15. 列表

数据池项和控件属性可以存放列表。下标运算符 ([]) 用于访问列表元素。语法如下所示：

<expression> [ <expression> ]

第一个表达式必须求值为列表类型，而第二个表达式必须求值为整数值。如果列表的类型为 list A，则整个列表下标表达式必须具有类型 A。

如果列表下标表达式出现在赋值左侧，则将写入所参考的列表元素的值。

length 关键字将返回列表的元素数。如果将它放在列表表达式之前，则整个表达式必须具有整数类型。



例 6.17.  
列表

```
// Assume this widget is a label and dp:textList is a list of strings
v:this.text = dp:textList[3];

dp:textList[1] = v:this.text; // writing the value of the list element

v:this.width = length dp:textList; // checking the length of the list
dp:textList[length dp:textList - 1] = "the end is here";
```

在 EB GUIDE 脚本 中，当前不支持在列表中添加和删除元素。

尝试访问列表结尾之外的列表元素将立即停止执行脚本。请确保所有列表访问都在范围内。

## 6.13.16. 事件

EB GUIDE 脚本 提供了以下用于处理事件的表达式：

- fire 表达式用于发送事件。语法如下所示：

```
fire ev:<identifier> ( <parameter list> )
```

事件可以具有参数，但不是必需的。fire 表达式的参数列表，必须与已触发事件的参数相匹配。如果事件没有参数，则括号必须为空白。



例 6.18.  
使用 fire 表达式

```
fire ev:toggleView(); // the event "toggleView" has no parameters
fire ev:mouseClick(10, 20); // "mouseClick" has two integer parameters
fire ev:userNameEntered("Ishmael"); // string event parameter
```

- fire\_delayed 表达式用于在指定的时间延迟后发送事件。语法如下所示：

```
fire_delayed <time> , ev:<identifier> ( <parameter list> )
```

time 参数是整数值，用于指定延迟（以毫秒为单位）。



#### 例 6.19.

使用 fire\_delayed 表达式

```
fire_delayed 3000, ev:mouseClick(10, 20); // send the event "mouseClick"
//in 3 seconds.
```

- ▶ cancel\_fire 表达式用于取消已延迟的事件。语法如下所示：

```
cancel_fire ev:<identifier>
```

- ▶ match\_event 表达式用于检查脚本的执行是否已由某个事件触发。语法如下所示：

```
match_event v:<identifier> = ev:<identifier>
in
    <expression>
else
    <expression>
```

match\_event 表达式的类型是 in 表达式和 else 表达式的类型，必须完全相同。

有一种特殊形式的 match\_event 表达式，在此表达式中您可以忽略 else 分支。此特殊形式具有类型 void，并且不得用于从脚本中返回值。



#### 例 6.20.

使用 match\_event 表达式

```
match_event v:theEvent = ev:toggleView in
{
    // this code will be executed when the "toggleView" event
    // has triggered the script
    dp:infoText = "the view has been changed";
}
else {}

match_event ( <expression> ) in <expression> //special form
//without an else branch
//The special form is an alternative way to express the following
match_event ( <expression> ) in { <expression> ; {} } else {}
```

如果某个脚本已由带有参数的事件触发，则可以在 match\_event 表达式的 in 表达式中访问这些参数。在您访问 C 中某个结构的字段时，使用点标记读取参数。事件参数不可用在 else 表达式中。



#### 例 6.21.

#### 事件参数

```
// assume that "mouseClick" has two parameters: x and y
match_event v:event = ev:mouseClick in
{
    dp:rectX = v:event.x;
    dp:rectY = v:event.y;
}
```

### 6.13.17. 字符串格式设置

在 EB GUIDE 脚本 中，字符串格式设置通过在字符串中将连接运算符 (+) 与各种数据到字符串转换函数结合使用来完成。EB GUIDE 脚本 标准库自带 `int2string` 功能，用于简单的整数到字符串转换。



#### 例 6.22. 字符串格式设置

```
// Assume this widget is a label and has a text property.
// Further assume that the datapool item dp:time_hour and
// dp:time_minute hold the current time.
v:this.text = "the current time is: " + f:int2string(dp:time_hour)
            + ":" + f:int2string(dp:time_minute);
```

### 6.13.18. 标准库

EB GUIDE 脚本 随附了一个标准库，该库包含一组外部函数，示例如下所示：

- ▶ 字符串格式设置
- ▶ 语言管理
- ▶ 跟踪
- ▶ 时间和日期
- ▶ 随机数生成

有关详细信息，请参见[节12.4.3](#)。

## 6.14. 脚本化的值

脚本化的值是控件属性值或数据池项值的替代标记。控件或数据池项的此类属性使用其他模型元素来求取自己的值，或者针对事件或属性更新做出反应。脚本化的值采用 EB GUIDE 脚本 脚本语言编写。

EB GUIDE 中的属性可以转换为脚本化的值，也可以转换回纯值。

有关说明，请参见[节9.7](#)。

为编辑脚本化的值，EB GUIDE Studio 包含了一个分为不同类别的脚本编辑器。



图 6.12. EB GUIDE Studio 中的 EB GUIDE 脚本编辑器

- ▶ 当读取脚本化的值属性时，会调用 "Read" 脚本。如果属性为列表类型，则参数包括列表索引。

"Read" 脚本的返回值代表属性的当前值。

- ▶ 当写入脚本化的值属性时，会调用 "Write" 脚本。

新属性值是 "Write" 脚本的参数。如果属性为列表类型，则参数包括列表索引。

"Write" 脚本的返回值用于控制属性的更改通知。

- ▶ true: 触发更改通知
- ▶ false: 不触发更改通知
- ▶ "触发器"列表包含用于触发运行 "On trigger" 脚本的事件、数据池项和控件属性的列表。
- ▶ 在初始化时，或者在事件触发器或属性更新后，将调用 "On trigger" 脚本。



"On trigger" 脚本的参数指示了运行脚本的原因。初始化或"触发器"列表中的触发器之一可能会导致运行脚本。

"On trigger" 脚本的返回值用于控制属性的更改通知。

- ▶ true: 触发更改通知
- ▶ false: 不触发更改通知
- ▶ "Length" 脚本仅适用于列表类型的属性。

"Length" 脚本的返回值表示列表的当前长度。

## 6.15. 键盘快捷键、按钮和图标

### 6.15.1. 键盘快捷键

下表列出了 EB GUIDE Studio 中可用的键盘快捷键，并说明了它们的含义。

表 6.3. 键盘快捷键

键盘快捷键	说明
Ctrl+C	复制选择
Ctrl+F	跳转到搜索框
Ctrl+S	保存
Ctrl+V	粘贴复制的选择
Ctrl+Y	重做
Ctrl+Z	撤销
Alt+F4	关闭活动窗口
Shift+F1	打开 &tf; 的用户文档 EB GUIDE TF
F1	打开 &tf; 的用户文档 EB GUIDE Studio
Shift+F2	重命名"数据池"或"事件"元素中的所选元素，以及所选元素的位置，如 EB GUIDE 脚本。适用于数据池项和事件。
F2	重命名所选元素
F3	在 EB GUIDE 模型中查找所选元素的所有
F5	启动模拟运行
F6	验证



键盘快捷键	说明
Del	从“内容”区域或元素中，删除所选模型元素
-	折叠“导航”或“大纲”元素中的所选元素
* 和 +	展开“导航”或“概述”元素中的所选模型元素
Up/Down/Left/Right	将内容区域中的所选状态或控件向上、下、左或右移动一个像素

## 6.15.2. 命令行选项

### 6.15.2.1. Studio.Console.exe 的命令行选项

在 EB GUIDE Studio 中，下表列出了 Studio.Console.exe 可用的命令行选项，并说明了它们的含义。未定义的命令行选项将被忽略。

命令行的一般语法如下：

Studio.Console.exe <option> "project\_name.ebguide"

表 6.4. <filename>Studio.Console.exe</filename> 的命令行选项 Studio.Console.exe

选项	说明
-c <logfile dir>	验证一个 EB GUIDE 模型，并将配置文件写入 logfile dir 指定目录
-e <destination dir>	将 EB GUIDE 模型导出到目标目录 destination dir 使用命令行选项 -p，请参见下面的示例。
-h	显示帮助消息
-l <language file>	导入一个保存为 language file 的语言文件 (.xliff) 到 EB GUIDE 模型，并创建一个日志文件
-m	允许项目迁移
-o	打开项目文件
-p <profile>	在导出期间，使用 profile 的配置文件



#### 例 6.23. 命令行选项

命令行 Studio.Console.exe -e "C:/temp/exported\_project" -p "target\_profile" -o "project\_name.ebguide" 使用配置文件 target\_profile 将 project\_name.ebguide 导出至指定的目标目录 C:/temp/exported\_project。

有关说明，请参见下列内容：

- ▶ [节10.4.1.2](#)
- ▶ [节10.5.2](#)
- ▶ [节10.8.2.2](#)

### 6.15.2.2. 命令行选项 Monitor.Console.exe

在 EB GUIDE Monitor 中，下表列出了 Monitor.Console.exe 可用的命令行选项，并说明了它们的含义。未定义的命令行选项将被忽略。

命令行的一般语法如下：

Monitor.Console.exe <option> "monitor.cfg"

表 6.5. 命令行选项 Monitor.Console.exe

选项	说明
-c <host:port>	将 EB GUIDE 模型关联到正在运行的 EB GUIDE GTF 进程
-h	显示帮助消息
-l <language>	将 EB GUIDE Monitor 的语言设置为以下项之一： en （对于英语）， ja （对于日语）， ko （对于韩语）， zh-cn （对于中文）。
-o	打开配置文件 monitor.cfg
-s	运行已定义脚本中的所有方法



#### 例 6.24. 命令行选项

命令行 Monitor.Console.exe -l ko 用于将 EB GUIDE Monitor 的语言设置为韩语。

有关如何使用 EB GUIDE Monitor 的说明，请参见[节10.9](#)。

### 6.15.3. 按钮

下表列出了 EB GUIDE Studio 和 EB GUIDE Monitor 中使用的按钮，并说明了它们的含义。

表 6.6. EB GUIDE Studio 的按钮

按钮	说明
	撤销









按钮	说明
	重做
	保存
	验证项目
	启动模拟运行
	停止模拟运行
	打开项目中心
	打开附加编辑器
	对内容区域和“导航”元素进行同步
	添加事件、数据池项或状态机
	打开与属性相关的上下文菜单。  根据按钮颜色，指示下列内容：   属性为局部属性。   属性已链接到另一属性。   属性已链接到数据池项。   属性值等于模板值。
	触发事件

### 6.15.4. 图标

下表列出了 EB GUIDE Studio 中使用的图标，并说明了它们的含义。

表 6.7. EB GUIDE Studio 的图标

图标	说明
	表示视图模板的退出动画
	表示视图模板的进入动画
	表示状态机或状态的进入动作
	表示状态机或状态的退出动作
	打开用于删除进入或退出动作的上下文菜单
	表示已启用动态状态机列表



图标	说明
	表示模板
	表示转换
	表示内部转换
	控件模板： 表示属性将添加到控件模板接口

## 6.16. 状态机和状态

### 6.16.1. 状态机

状态机是一种决定性有限状态自动机，用于说明系统的动态行为。在 EB GUIDE 中，状态机由任意数目的分层排列状态以及状态之间的转化组成。

在 EB GUIDE 中，您可创建以下类型的状态机：

#### 6.16.1.1. 触觉状态机

触觉状态机允许指定 GUI。

#### 6.16.1.2. 逻辑状态机

逻辑状态机允许在没有 GUI 的情况下指定一些逻辑。

#### 6.16.1.3. 动态状态机

动态状态机与其他状态机并行运行。

动态状态机不会在系统启动时自动启动。动态状态机的启动和停止由另一个状态机发起。

有两种类型的动态状态机：

- ▶ 触觉动态状态机
- ▶ 逻辑动态状态机

有关说明，请参见[图11.1](#)。

## 6.16.2. 状态

EB GUIDE 使用状态概念。状态决定了状态机的状况和行为。状态由转换进行链接。转换是两种状态之间的连接，用于定义从源状态到目标状态的状态更改。

状态具有以下属性：

- ▶ 进入动作
- ▶ 退出动作
- ▶ 内部转换

### 6.16.2.1. 组合状态

组合状态可以将它包含的其他状态用作子状态。组合状态结构是分层结构，并且可能的子状态的数目任意。任何类型的状态都可以嵌套在组合状态中。



图 6.13. 组合状态

在“导航”元素中，状态层次结构显示为树结构。

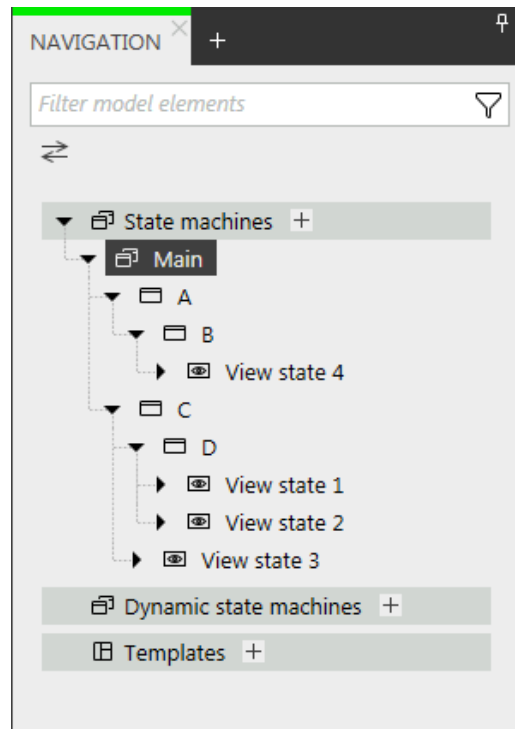


图 6.14. 以树形式显示的状态层次结构

组合状态可以具有任意数目的传入和传出转换以及任意数目的内部转换。子状态继承父状态的转换。

### 6.16.2.2. 视图状态

视图状态包含视图。视图表示特定于项目的 HMI 屏幕。在对应的视图状态为活动状态时，将显示视图。视图由作为用户与系统之间的接口的控件组成。

### 6.16.2.3. 初始状态

初始状态定义了状态机的起始点。初始状态具有指向第一个状态的默认传出转换。初始状态没有传入转换。

初始状态可用作组合状态的起始点，或者可用于通过下列方式进入组合状态：

- ▶ 到组合状态的转换，初始状态是必需的
- ▶ 到组合状态中的子状态的转换

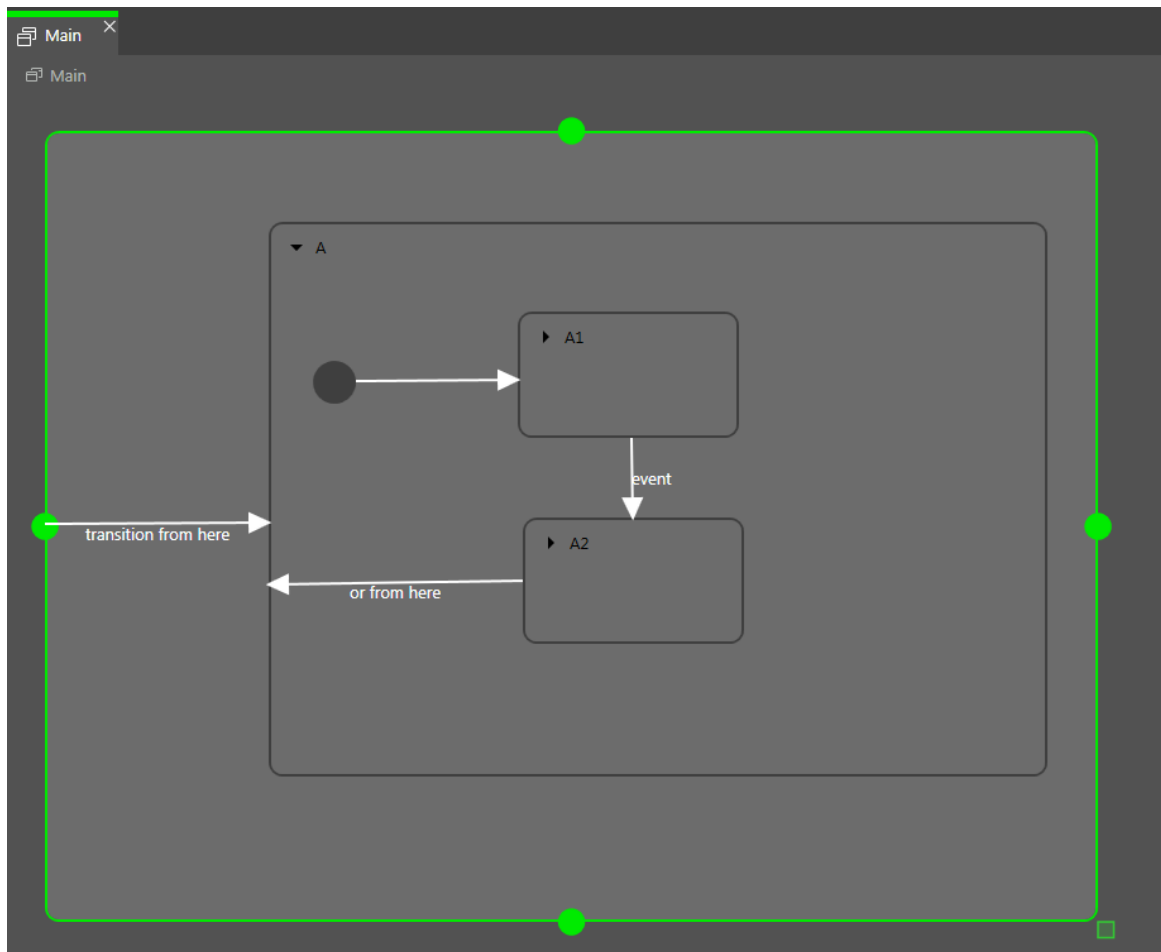


图 6.15. 初始状态示例

#### 6.16.2.4. 结束状态

结束状态用于退出组合状态。如果进入了状态机的结束状态，则状态机将终止。组合状态中的所有历史状态将被重置。结束状态没有任何传出转换。

组合状态只能有一个结束状态。结束状态由以下操作触发：

- ▶ 从子状态转换到组合状态外部（包含事件 **z** 的转换）
- ▶ 自组合状态的传出转换（包含事件 **y** 的转换）
- ▶ 到组合状态中的结束状态的转换（包含事件 **x** 的转换）

如果组合状态包含结束状态，则该组合状态必须具有传出转换。

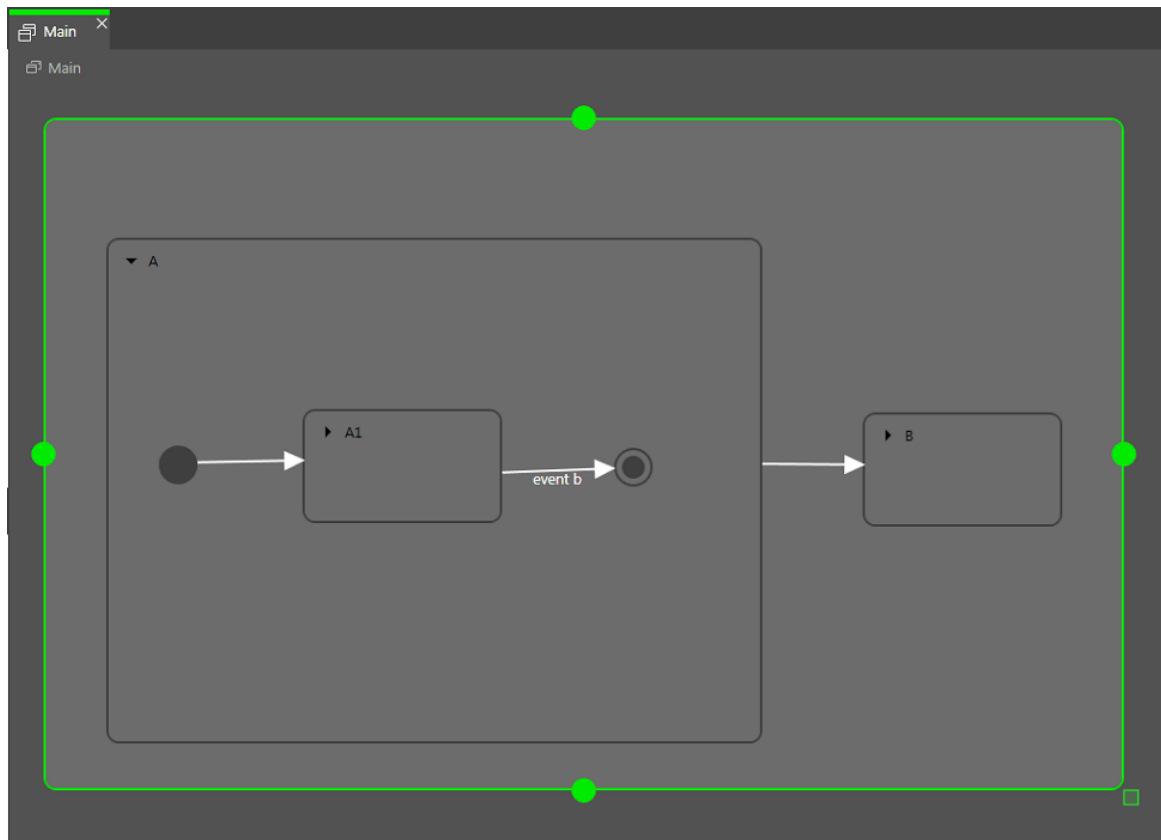


图 6.16. 组合状态中结束状态的用法

### 6.16.2.5. 选择状态

选择状态用于实现动态条件分支。根据条件触发事件时将使用此状态。选择状态是源状态和目标状态之间的连接。选择状态可以具有多个传入和传出转换。将为每个传出转换指定一个条件，并且只有在该条件求值为 true 的情况下，才会执行此转换。其中一个传出转换为 else 转换。如果所有其他条件求值为 false，那么将执行此转换。else 转换是必需的。

一些传出转换为 true 是可能的，因此必须定义对传出转换进行求值的顺序。

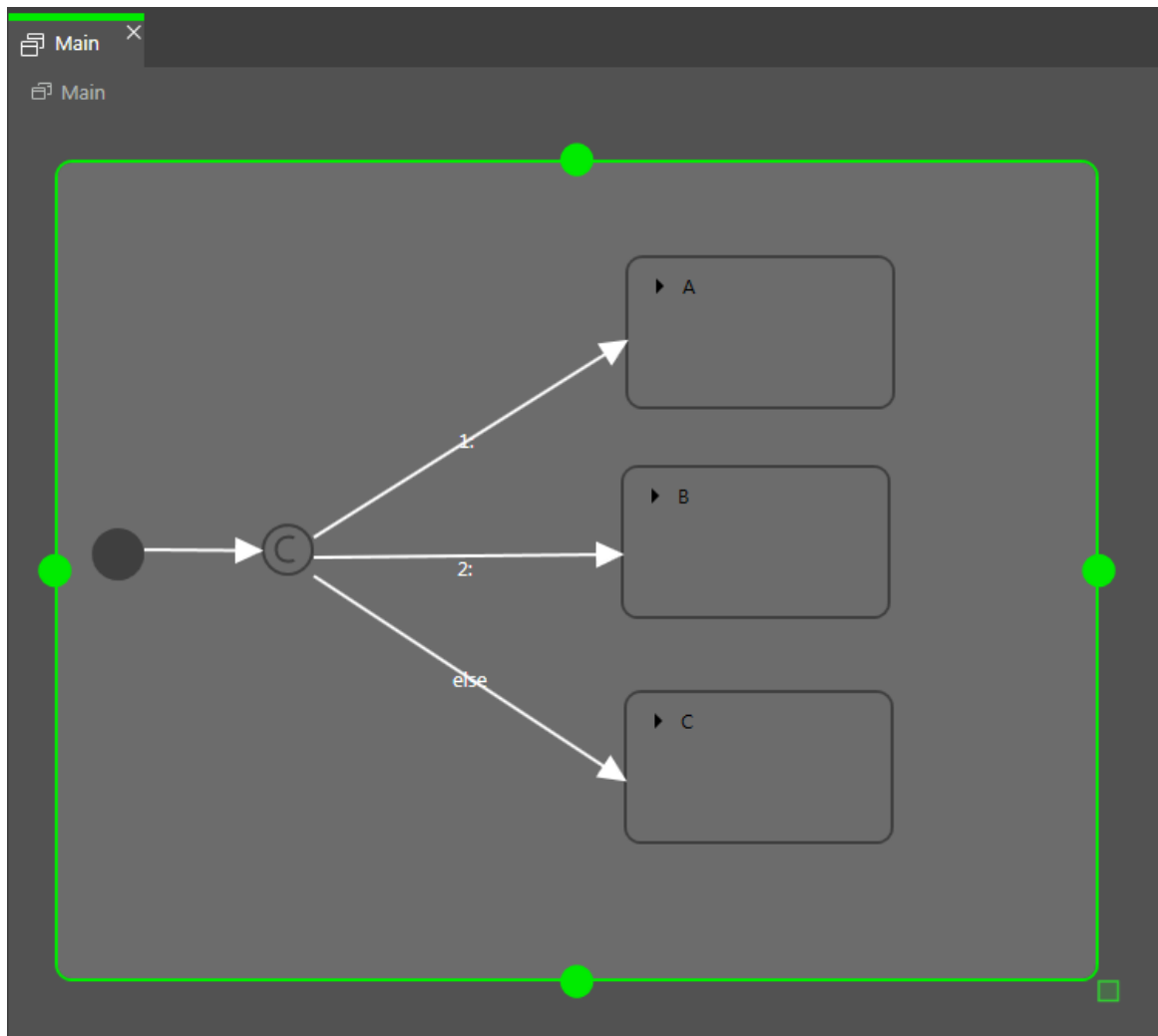


图 6.17. 具有传入和传出转换的选择状态

#### 6.16.2.6. 历史状态

EB GUIDE 支持两种类型的历史状态：

- ▶ 浅度历史状态存储最近的活动子状态：仅在退出组合状态之前活动的子状态。
- ▶ 深度历史状态仅在组合状态退出之前存储该组合状态及其完整的子层次结构。

首次进入历史状态的父状态时，将恢复上一个活动子状态。

浅度历史状态仅记忆退出组合状态之前上一个活动的状态。它不能记忆层次结构。

浅度历史状态将恢复组合状态内记录的上一个活动状态。它具有不带条件的默认外出转换，但可以具有多个传入转换。

首次进入组合状态时，浅度历史状态为空。进入空的浅度历史状态时，浅度历史状态默认转换将决定下一个状态。



例 6.25.  
浅度历史状态

可以按以下方式使用浅度历史状态。

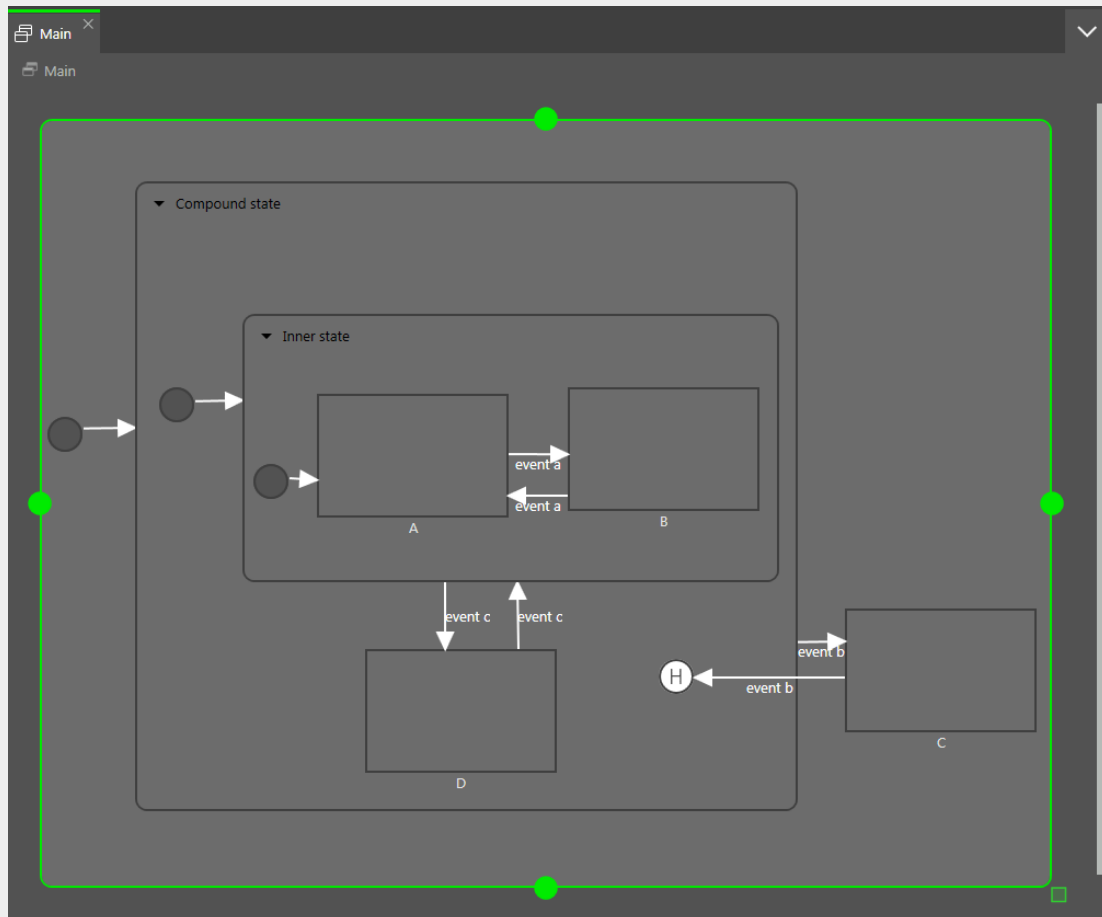


图 6.18. 浅度历史状态

- ▶ 案例 1：活动状态是 D。
  1. event b 已触发并且进入了状态 C。
  2. event b 已再次触发并且进入了浅度历史状态。
  3. 状态机从浅度历史状态进入状态 D，因为状态 D 是 Compound State 中的最后一个活动状态。
- ▶ 案例 2：活动状态是 B。
  1. event b 已触发并且进入了状态 C。
  2. event b 已再次触发并且进入了浅度历史状态。



3. 状态机从浅度历史状态进入了 Inner state，因为浅度历史状态可以记住最后一个活动的状态，但无法记住层次结构。
4. 进入 Inner state 导致进入状态 A。

深度历史状态能够保存分层历史记录。



#### 例 6.26. 深度历史状态

可以按以下方式使用深度历史状态。

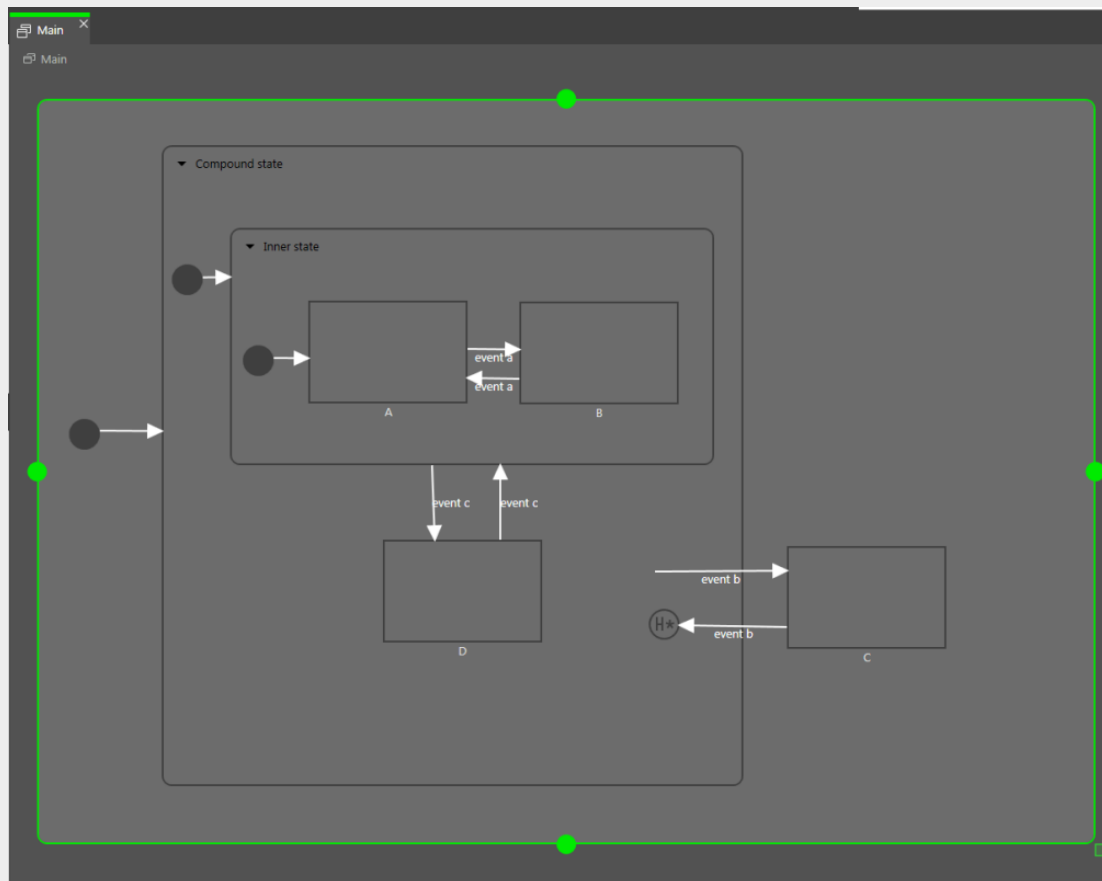


图 6.19. 深度历史状态

- ▶ 案例 1：活动状态是 D。
  1. event b 已触发并且进入了状态 C。
  2. event b 已再次触发并且进入了深度历史状态。
  3. 状态机从深度历史状态进入状态 D，因为状态 D 是 Compound State 中的最后一个活动状态。
- ▶ 案例 2：活动状态是 B。



1. event b 已触发并且进入了状态 C。
2. event b 已再次触发并且进入了深度历史状态。
3. 状态机从深度历史状态进入状态 B，因为状态 B 是最后一个活动状态，并且深度历史状态可以记住状态层次结构。

一个状态可以具有浅度历史状态或深度历史状态。您可以具有处于父状态的历史状态以及另一个处于子状态的历史状态。

### 6.16.3. 转换

转换是源状态和目标状态之间的定向关系。它使状态机从一个状态进入另一个状态。转换具有以下属性：

- ▶ 用于执行转换的触发器

触发器可以为事件或数据池项的更改。

- ▶ 必须求值为 true 才能执行转换的条件
- ▶ 随转换执行的操作

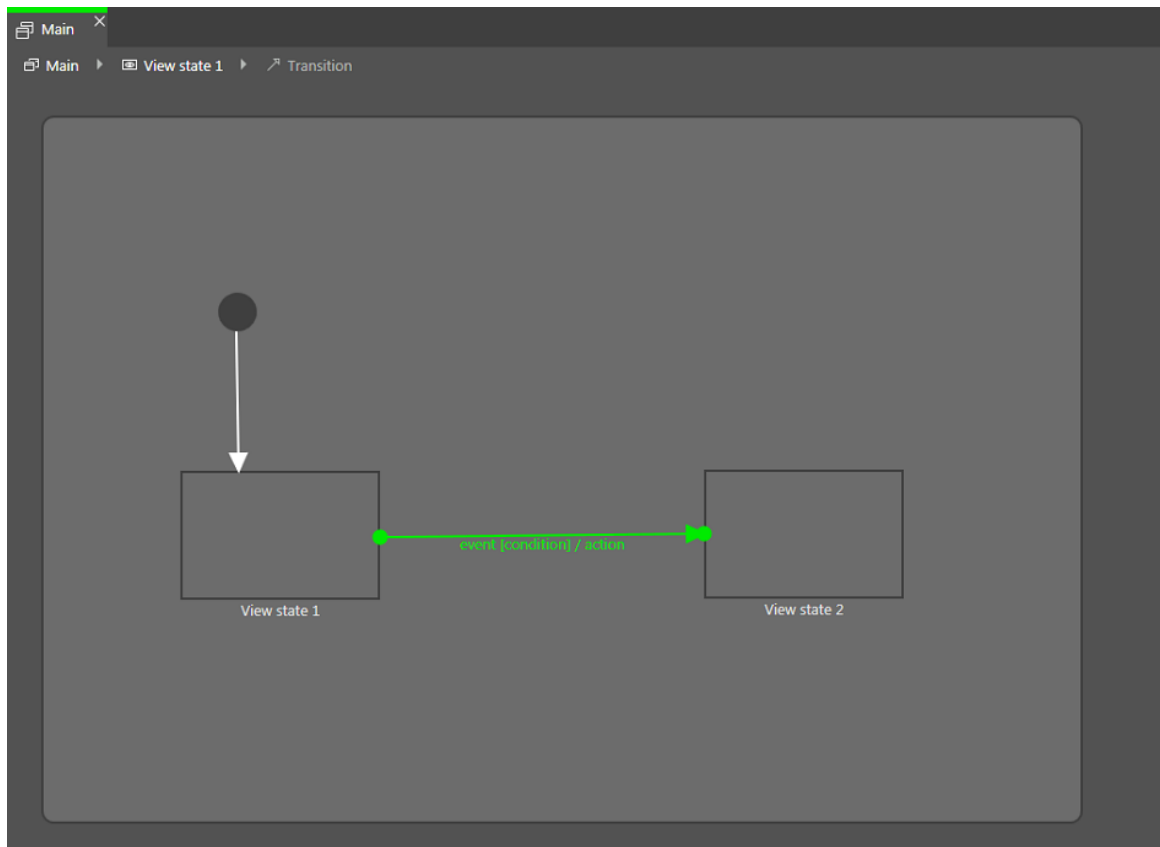


图 6.20. 转换

## 注意



## 转换具有确定性

即使对于具有不同条件的同一事件，也不可能出现自特定源状态的多个转换。如果假定状态机根据不同条件转至不同的目标状态，请使用选择状态。

状态将从其父状态处继承所有转换。如果多个状态与另一个状态共享相同的转换，则封闭组合状态可用于捆绑这些转换，并因此减少条件数。



## 例 6.27.

## 转换继承

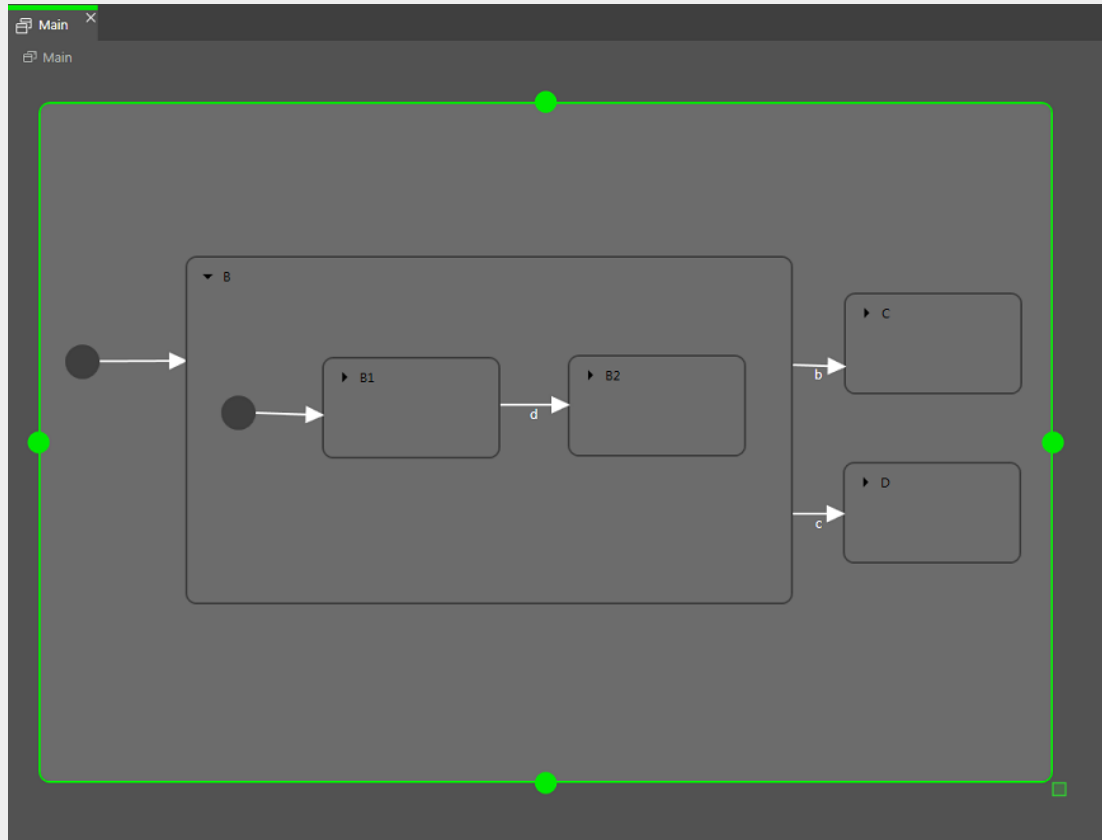


图 6.21. 转换继承

如果事件 b 是在状态机处于 State B1 时触发的，则将运行到 State C 的转化，因为子状态 State B1 和 State B2 将继承状态 State B 的转化。

如果自子状态的内部转换使用的事件与自父状态的外部转换相同，则转换继承将被覆盖。



例 6.28.

转换覆盖

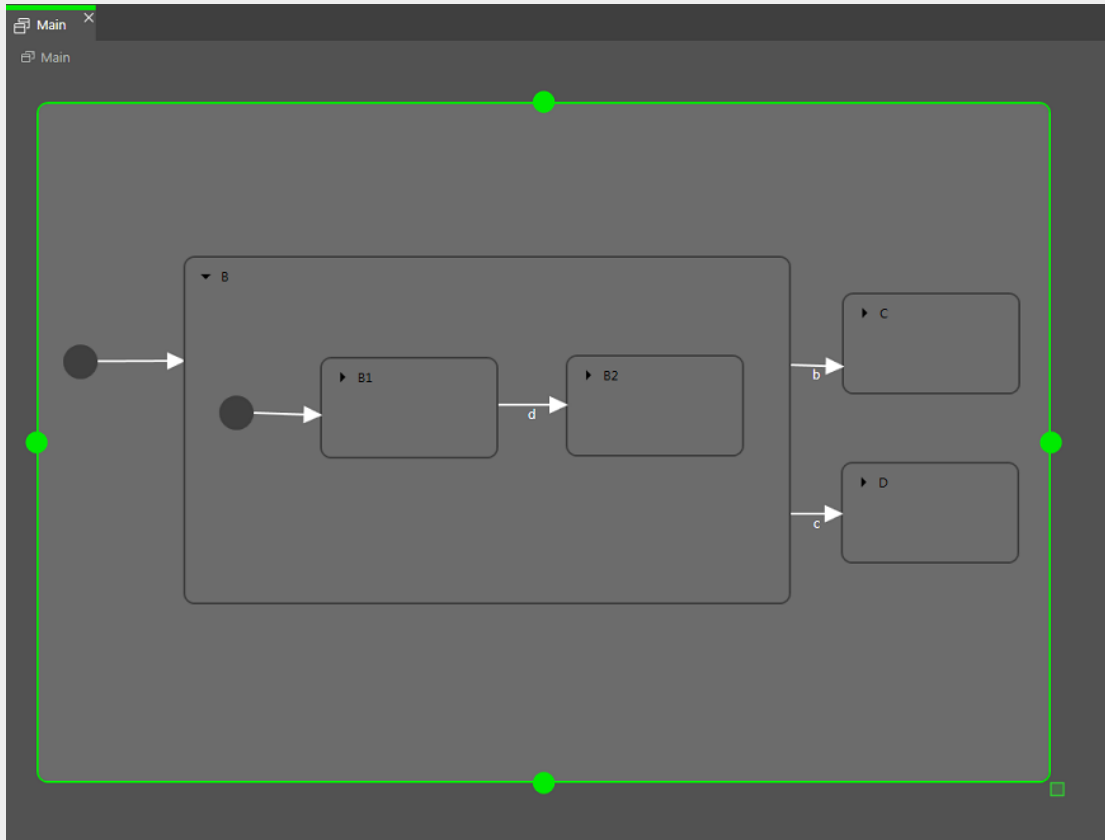


图 6.22. 转换覆盖

如果事件 d 是在状态机处于状态 State B 时触发的，则将执行到 State C 的转换。

如果事件 d 是在状态机处于状态 State B1 时触发的，则将执行到 State B2 的转换，而不会执行到 State C 的转换。由于两个转换同名，因此内层转换将覆盖外层转换。

注意



执行层次结构

在状态机中，使用同一事件的转换的执行层次结构始终由内而外。这意味着与外部转换相比，内部转换是首选。

有不同类型的转换。

► 默认转换

默认转换是自动触发的，而并非由任何事件或数据池项更新触发。它不包含任何条件，但是可以具有操作。它可以与初始状态、结束状态、选择状态及历史状态配合使用。

► 选择转换



选择转换是向其分配了条件的传出转换。其源状态为选择状态。选择转换由对其条件的求值触发。它们将触发某项操作。将执行第一个条件为 `true` 的选择转换。

► **Else 转换**

**else** 转换是选择转换的必需对应部分。每个选择状态需要具有一个 **else** 转换，后者将在所有其选择转换的条件求值为 `false` 时执行。.

► **内部转换**

内部转换是没有目标状态并因此不会更改活动状态的转换。内部转换的目的是对事件作出响应而不退出当前状态。它可以具有条件并触发某项操作。

对于处于某个状态的同一事件，可以具有多个内部转换。定义了执行顺序。

► **自转换**

自转换是源状态和目标状态相同的转换。与内部转换不同，自转换退出当前状态并重新进入，并因此执行其进入和退出动作。

## 6.16.4. 状态机的执行

执行状态机时，它在任何时候都只能有一个活动状态。状态机是由事件驱动的。

状态机循环如下所示：

1. 状态机通过进入其初始状态而启动。
2. 状态机等待传入事件。
  - a. 找到内部转换。
    - i. 在当前状态下启动并搜索第一个由当前事件触发并且条件为 `true` 的内部转换。如果找到了此类转换，那么将执行该转换。
    - ii. 如果未找到任何转换，则转至父状态并搜索第一个由当前事件触发且条件为 `true` 的内部转换。
    - iii. 如果未找到任何转换，则重复先前步骤，直至到达顶级状态为止。
  - b. 对内部转换进行处理。

执行内部转换将仅触发连接到此内部转换的动作。不会退出并重新进入状态。
  - c. 找到了转换。
    - i. 在当前状态下启动，并搜索由当前事件触发且条件为 `true` 的转换。如果找到了此类转换，那么将执行该转换。
    - ii. 如果未找到任何转换，则转至父状态并搜索转换。
    - iii. 重复先前步骤，直到找到第一个适合的转换为止。

d. 对转换进行处理。

执行转换会将状态机从一个状态更改为另一个状态。将退出源状态，并进入目标状态。

只有在触发了某个转换的对应事件，且条件取值为 true 的情况下，才会执行该转换。

转换可以退出并进入状态层次结构中的多个组合状态。在退出层叠和进入层叠之间，将执行转换的动作。

进入某个状态可能需要执行后续转换，例如进入组合状态需要执行初始状态转换作为后续转换。可以执行一系列后续转换。

3. 状态机在达到其结束状态后将停止。

如果转换跨越状态层次结构中的多个状态，则将执行退出和进入动作层叠。



例 6.29.  
执行转换

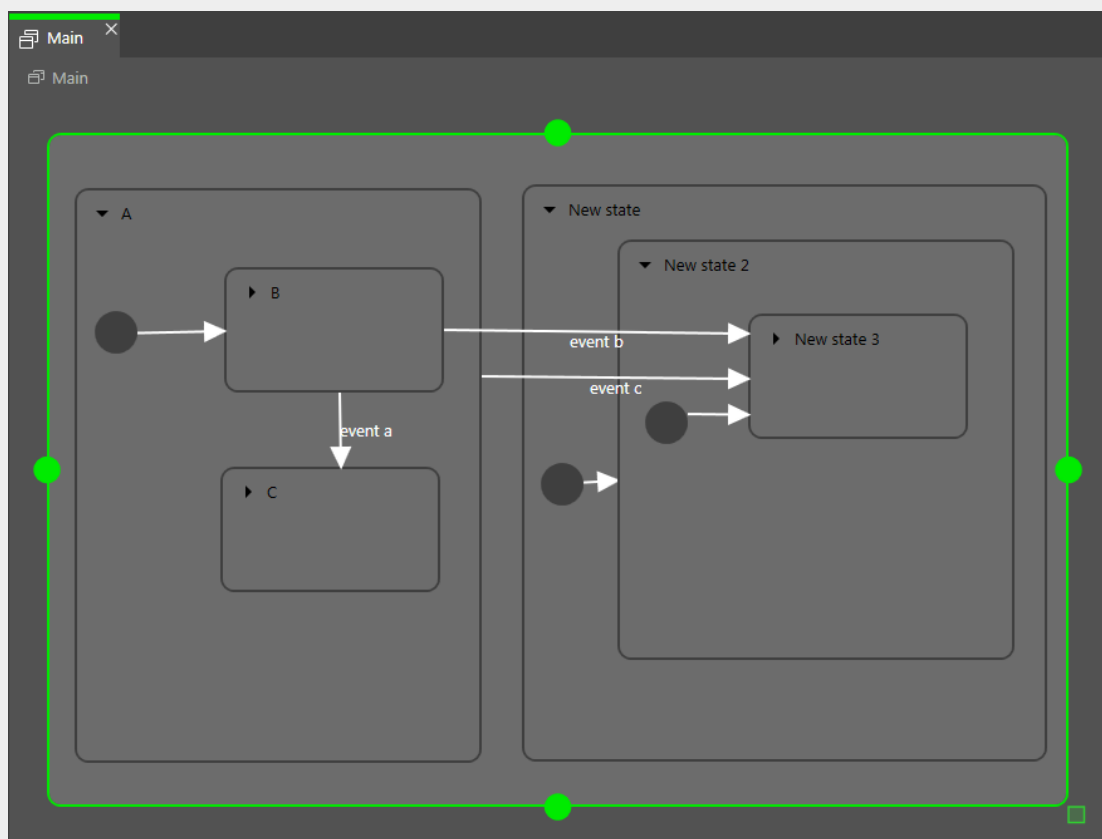


图 6.23. 执行转换

触发 event a 后，将发生下列情况：

1. 退出状态 B。

2. 进入状态 C。

触发 event b 后，将发生下列情况：

1. 退出状态 B。
2. 退出状态 A。
3. 进入状态 New state。
4. 进入状态 New state 2。
5. 进入状态 New state 3。

触发 event c 后，将发生下列情况：

1. 如果状态 B 或状态 C 为活动状态，则将退出状态 B 状态 C。
2. 退出状态 A。
3. 进入状态 New state。
4. 进入状态 New state 2。
5. 进入状态 New state 3。



例 6.30.  
执行转换

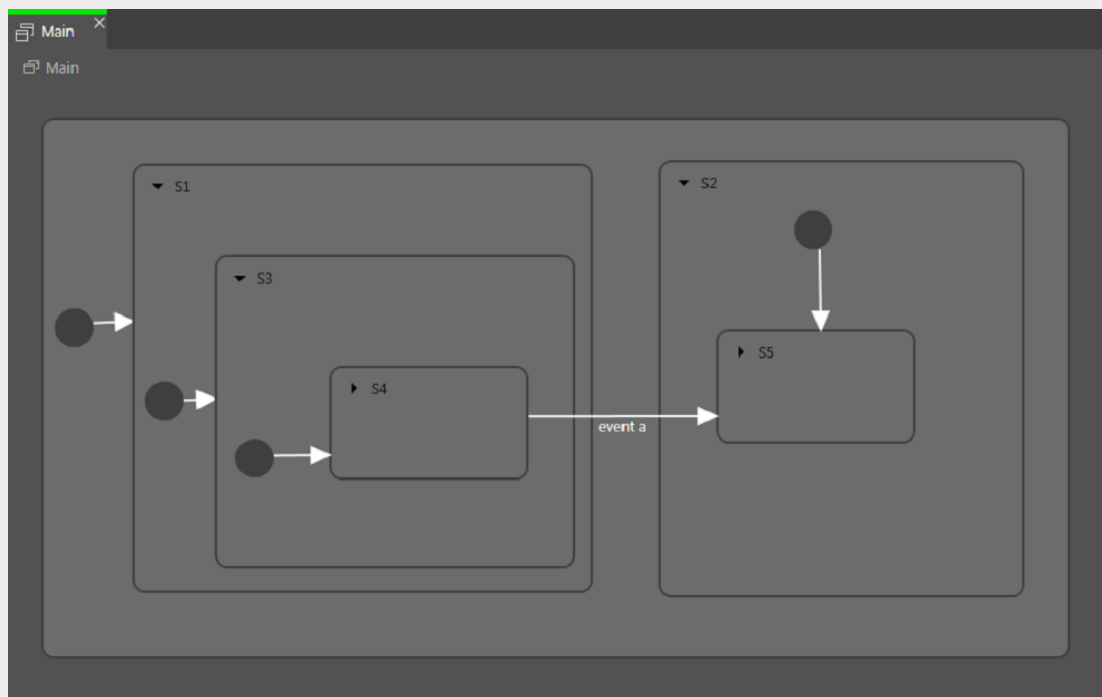


图 6.24. 执行转换



event a 触发转换时，将发生下列情况：

1. 退出状态 S4。
2. 退出状态 S3。
3. 退出状态 S1。
4. 进入状态 S2。
5. 进入状态 S5。



例 6.31.  
执行转换

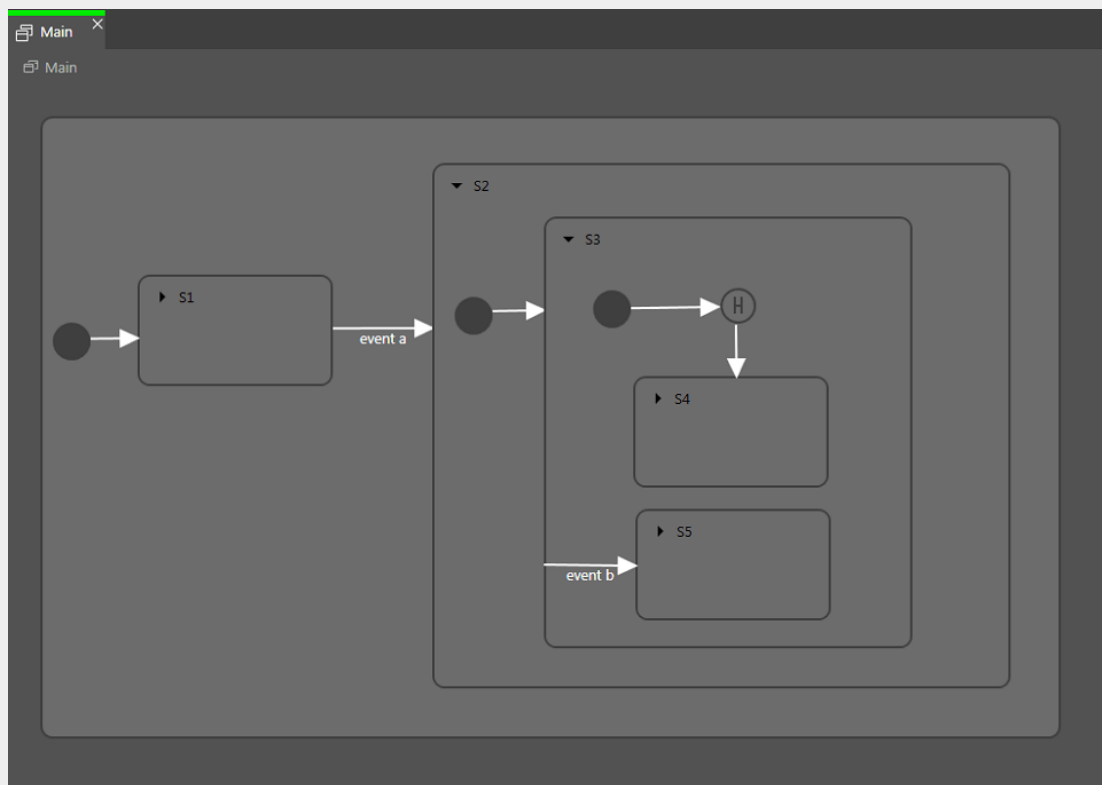


图 6.25. 执行转换

由 event a 触发的转换导致出现下列转换序列：

1. 状态机将进入状态 S2。
2. 默认转换导致进入状态 S3。
3. 下一个默认转换将进入浅度历史状态。
4. 浅度历史状态将恢复状态 S3 的最后一个活动状态，即状态 S4 或状态 S5。



对于每个步骤，将分别运行 `entry-exit-cascade`。

### 6.16.5. EB GUIDE 与 UML 标记相比的 标记

在此部分中，会将 EB GUIDE 标记与 Unified Modeling Language (UML) 2.5 标记进行比较。

#### 6.16.5.1. 受支持的元素

下表显示了 EB GUIDE 支持的所有 UML 2.5 元素。一些元素的名称不符合 UML 2.5 中的命名约定，但这些元素的功能是相同的：

在……的名称 <b>EB GUIDE</b>	<b>UML 2.5</b> 中的名称
初始状态	初始（伪状态）
结束状态	结束状态
组合状态	状态
选择状态	选择（伪状态）
深度历史状态	深度历史（伪状态）
浅度历史状态	浅度历史（伪状态）
内部转换	内部转换
转换	外部/局部转换 <sup>a</sup>

<sup>a</sup>EB GUIDE 不区分外部和局部转换。

#### 6.16.5.2. 不受支持的元素

以下 UML 2.5 元素在 EB GUIDE 中不受支持：

- ▶ 联接
- ▶ 分叉
- ▶ 接合
- ▶ 入口点
- ▶ 出口点
- ▶ 终止



### 6.16.5.3. 偏差

UML 2.5 标记中的一些元素未在 EB GUIDE 中实现。但是，可以使用 EB GUIDE 概念对这些元素的功能进行建模。

UML 2.5 中的概念	替代方法 <b>EB GUIDE</b>
并行状态	概念使用动态状态机进行实现。
每次转换的触发器数	概念使用 EB GUIDE 脚本 在数据池项或视图中进行实现。
转换中的时间触发器	概念使用 EB GUIDE 脚本 (fire_delayed) 在状态机、数据池项、转化或视图中进行实现。

## 6.17. 触摸输入

EB GUIDE 支持两种类型的触摸输入：触摸操作动作和多点触控输入。

每个触摸操作动作在 EB GUIDE Studio 中都表示为一个控件功能。启用控件功能会向控件中添加一组属性。

操作动作分为两个基本类型：

- ▶ 非路径动作
- ▶ 路径动作

### 6.17.1. 非路径动作

EB GUIDE 实现以下非路径动作：

- ▶ 滑动
- ▶ 二指缩放
- ▶ 旋转
- ▶ 按
- ▶ 长按

非路径动作包含多点触摸动作和单点触摸动作。多点触摸操作动作需要支持多点触摸输入的输入设备。单点触摸操作动作作用于任何受支持的输入设备。

每个操作动作的反应相互独立。如果启用了多个操作动作，则建模员负责确保 EB GUIDE 模型的行为一致。

## 6.17.2. 路径动作

路径动作是手指在触摸屏上画出的形状或者由一些其他输入设备输入的形状。为控件启用控件功能后，用户可以输入从该控件开始的形状。此形状必须超过路径动作识别器所考虑的可配置最低边界框。此形状与一组已知形状以及某个已识别的操作动作（如果找到了匹配项）进行比较。

有关说明，请参见[节11.3](#)。

## 6.17.3. 输入处理和操作动作

操作动作识别并行运行到正常输入处理。每个操作动作可以请求从正常输入处理中删除操作动作中涉及的接触。操作动作请求删除接触的时间取决于实际操作动作，并且对于某些操作动作，可以配置此时间。

接触删除仅与操作动作中涉及的手指相关。接触被删除后，正常输入处理会将其忽略，直到接收到针对接触的释放事件为止。在没有邻近感应支持的触摸屏上，这暗示接触被删除后不会触发任何其他触摸反应。

### 提示



#### 从正常输入处理中删除接触

请考虑包含用于操作动作的按钮和控件功能的窗口。操作动作中涉及接触时，它不应该导致触发与按钮关联的操作，即使释放按钮上的接触也是如此。

## 6.17.4. 多点触摸输入

EB GUIDE 能够处理多点触摸输入，前提是使用了兼容的多点触摸输入设备。

多点触摸是某个表面识别并跟踪输入设备上的多个接触点的能力。典型场景是多个手指接触触摸屏。

### ► 多点触摸事件处理

多点触摸事件使用触摸事件机制进行分派，来自鼠标和单点触摸触摸屏的事件以相同方式进行分派。唯一的区别在于，每次接触触发触摸反应的行为独立于所有其他接触。为了能够区别各个接触，向每个接触反应提供称为 `fingerid` 的参数。

### ► 手指 ID

为输入设备跟踪的每个接触分配了一个用于标识它的数字。此识别符称为 `fingerid` 并且对于每个输入设备都是唯一的。但是，可以稍后在同一值不再使用后将其分配给另一接触。

请考虑启用多点触摸输入后允许终端用户执行的额外触摸交互顺序。它们包括以下内容：



- ▶ 终端用户可以同时与界面的多个元素进行交互，例如在列表中滚动时按下某个按钮。
- ▶ 终端用户可以将多个手指放在单个控件上。

两个典型的情况是滚动和拖动此清单。可以通过使用 `fingerid` 正确处理它们。根据所需的行为，可能的方案包括以下内容：

- ▶ 仅允许按下控件的第一个手指执行滚动和/或拖动。
- ▶ 始终使用最后一个手指放在控件上执行滚动和/或拖动。可以略微修改对上述方法来轻松实现此目的。

## 6.18. 控件

控件是组成 EB GUIDE 模型的基本图形元素。

可以对控件进行自定义。编辑某个控件的属性可以使该控件满足个人需要。示例属性包括大小、颜色、布局或被触碰或移动时的行为。

可以对控件进行组合：在较小的构建块之外创建了复杂结构。例如，某个按钮可由一个矩形、一个图像以及一个标签构成。

可以对控件进行嵌套：在控件层次结构中，下级控件称为子代控件，而上级控件称为父代控件。

### 6.18.1. 视图

视图是每个场景中最高的控件。进行建模时，基本控件、3D 控件、动画和控件模板将放入视图中。每个视图都正好关联到一个视图状态。在没有视图状态的情况下，视图无法存在。

## 注意



## 更改视图的大小

在 EB GUIDE Studio 中，您可以增大或减小视图的大小，以便获得特写视图或看到更多内容。要放大和缩小，请使用滑块控件或点击视图底部的文本框。默认缩放比例为 100%。或者，使用 **Ctrl++** 可进行放大，使用 **Ctrl+-** 可进行缩小，并且使用 **Ctrl+0** 可将缩放比例重置为 100%。

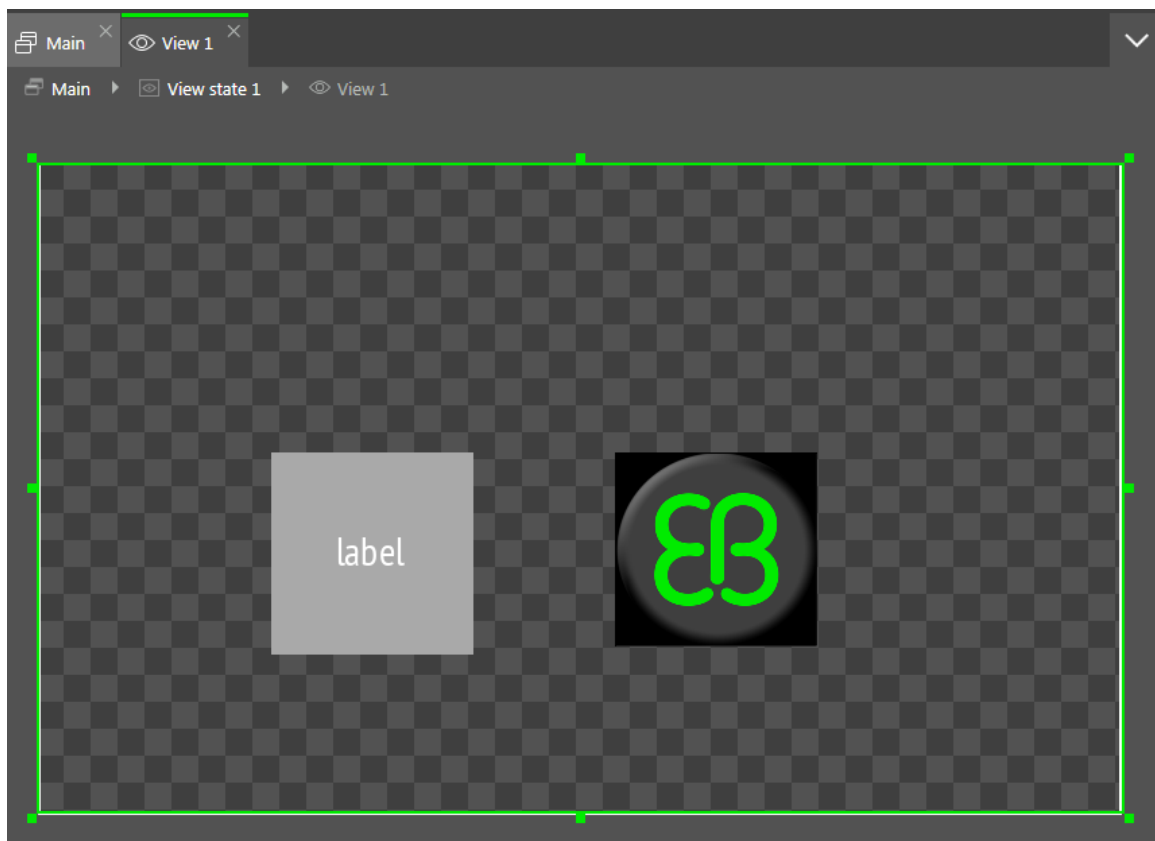


图 6.26. 包含矩形、标签和图像的视图

## 6.18.2. 控件类别

在“工具箱”中，控件按类别分组。以下为可用的类别。

► 基本控件

基本控件为 **alpha** 掩码、动画、容器、椭圆、图像、实例化器、标签和矩形。

► 3D 控件

“3D 控件”类别包含显示了 **3D** 图形的控件。3D 控件为场景图、场景图节点、材料、PBR **Phone** 材料、PBR **GGX** 材料、网格、摄像头、定向光、点光、聚光和环境光。

## 注意



## 受支持的渲染器

要显示 3D 图形，需要使用 OpenGL ES 2.0 版本或更高版本或 DirectX 11 渲染器。确保您的图形驱动程序与渲染器的版本兼容。

## ▶ 控件模板

“模板”类别包含控件模板。仅当定义了控件模板时它才可见。

## ▶ 定制控件

“定制元素”分类， 包含定制的窗口控件，因此只有将定制的窗口控件添加到项目时才可见。有关详细信息，请参见我们的网站 <https://www.elektrobit.com/ebguide/learn/resources/>。

有关说明，请参见[节8.1](#)。

### 6.18.3. 控件属性

控件由一组属性定义，这些属性用于指定该控件的外观和行为。“属性”元素显示当前关注的控件的属性，并允许编辑属性。

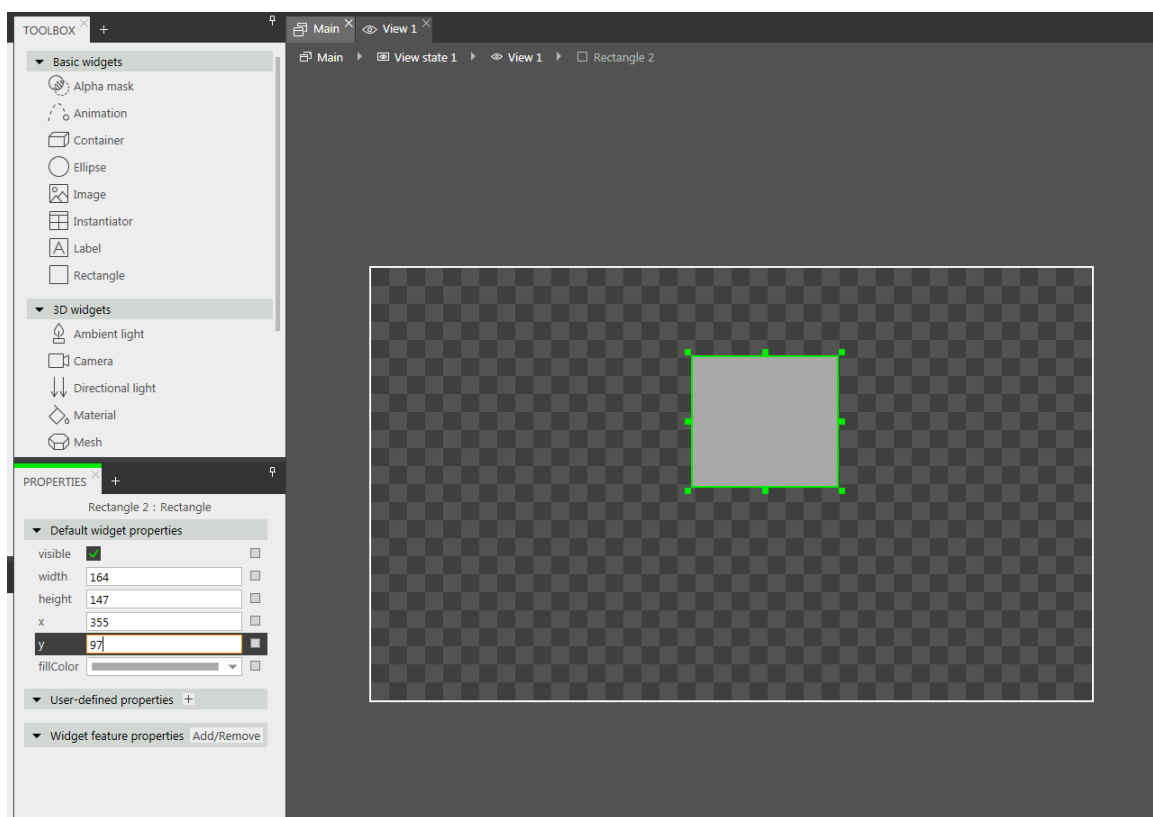


图 6.27. 矩形控件及其属性

有三种类型的控件属性：

- ▶ 默认控件属性与每个控件实例一起创建。有关所有控件的默认属性列表，请参见[节12.10](#)。
- ▶ 除了默认属性之外，用户定义的控件属性也是由建模员创建的。
- ▶ 当建模员向控件中添加控件功能时，控件功能属性由 **EB GUIDE Studio** 创建。控件功能属性按类别分组。控件功能为控件的外观和行为增加了更多功能。




**例 6.32.**  
“触摸”控件功能

“触摸”控件功能定义了当控件被触摸时是否响应以及如何响应。其添加四个控件。布尔属性 `touchable` 用于确定控件是否响应触摸输入。如果当前已触摸该控件，布尔属性 `touched` 在运行时间由 **EB GUIDE** 设置。两个整数属性 `touchPolicy` 和 `touchBehavior` 确定该控件如何响应触摸输入。

## 6.18.4. 控件模板

通过使用控件模板，可以定义能够在 **EB GUIDE** 模型中多次使用的自定义控件。您可以根据现有控件定义模板，也可以根据现有模板获得新模板。创建模板后，您可以根据需要修改模板，例如通过添加属性或控件功能来修改模板。通过使用控件模板，您可以构建包含复杂控件的库。

控件模板具有模板接口。模板接口包含控件实例中可见且可访问的模板属性。控件实例将继承其模板的接口的属性。所继承的属性称为模板属性。模板属性以  按钮标记。

更改模板属性的值时，该属性将变为局部属性。局部属性以  按钮标记。



**例 6.33.**  
控件模板及其实例的属性的关系

将控件模板 `Square` 添加至 **EB GUIDE** 模型。假设 `Square` 具有属性 `color`。`color` 添加至模板接口。假设 `color` 的值为 `red`。

您将控件模板 `Square` 的实例添加至视图。此实例名为 `BlueSquare`。

- ▶ `BlueSquare` 将继承值为 `red` 的 `color`。
- ▶ 在 `Square` 模板中，将 `color` 的值更改为 `green`。  
=> `BlueSquare` 中 `color` 的值也将更改为 `green`。
- ▶ => 在 `BlueSquare` 中，将 `color` 的值更改为 `blue`。  
在 `Square` 模板中，将 `color` 的值更改为 `yellow`。  
=> `BlueSquare` 中 `color` 的值仍为 `blue`。



有关说明，请参见[节8.7](#)。

## 6.18.5. 控件功能

可以使用控件功能拓展控件和控件模板的功能。控件功能预定义控件属性。控件功能按类别分组。

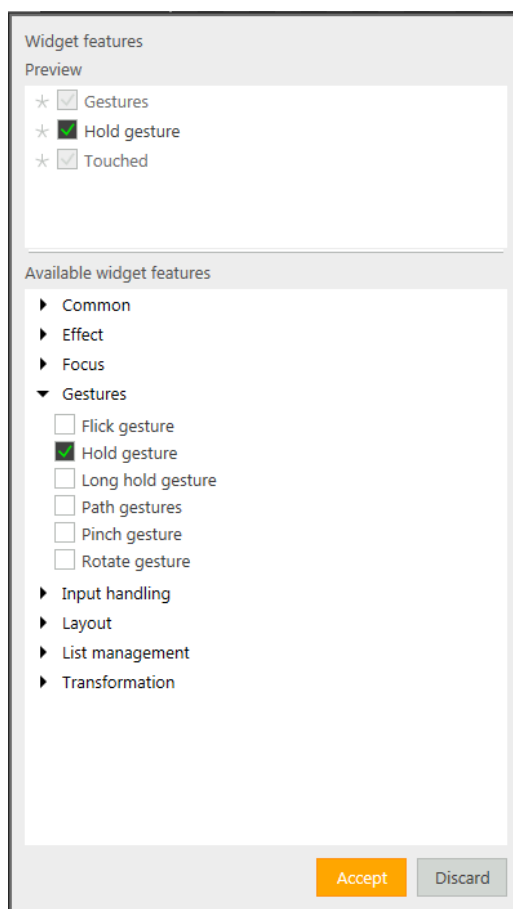


图 6.28. 控件功能

如果您向控件模板添加控件功能，则所创建的控件模板实例将继承已添加的控件功能。请注意，您不能将控件功能添加到控件模板实例或从模板创建的模板。

使用控件功能的限制如下：

- ▶ 控件功能未能继承层次结构。
- ▶ 每个控件功能不可多次添加控件。
- ▶ 一些控件功能是相互依存的。这意味着，要添加一个控件功能，您必须添加另一个，否则控件功能可能会相互排斥。
- ▶ 控件功能可以限制在特定类型的控件中。

► 控件功能在运行时无法激活或停用。

默认情况下，禁用所有控件功能。如果您需要一个指定的控件功能，您需要将它添加到控件中。

有关说明，请参见[节8.3](#)。有关所有控件功能的列表，请参见[节12.11](#)。

### 6.18.5.1. 焦点控件功能类别

在 EB GUIDE Studio 中，使用“聚焦”控件功能，可以为控件焦点管理建模。“自动聚焦”和“用户定义聚焦”。

以下两个焦点方向可用：

1. 前进方向：下一个可聚焦控件已聚焦。
2. 向后方向：前一个可聚焦控件已聚焦。

“自动聚焦”和“用户定义聚焦”控件功能，为如何处理向前方向聚焦提供了配置。对于后退方向，使用相同的聚焦顺序，但只能反向使用。

“聚焦”控件功能有以下特点：

“自动焦点”

在这个策略中，焦点是从左到右从顶行分配在可聚焦控件之间。通过小部件树的结构定义顺序。

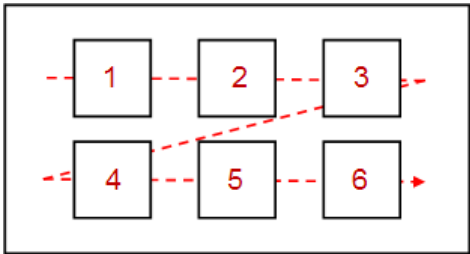


图 6.29. “自动焦点”控件功能的属性

无法跳过可获得焦点的子代控件。不可见控件，禁用 focused 属性的控件和无“聚焦”控件功能的控件，不视为可聚焦控件。因此，当确定当前聚焦的控件时，它们被跳过。

“用户定义焦点”

由于视图的复杂性，通过自动对焦策略的重点排序可能相当困难。在这种情况下，确定用户定义的焦点顺序是有用的。

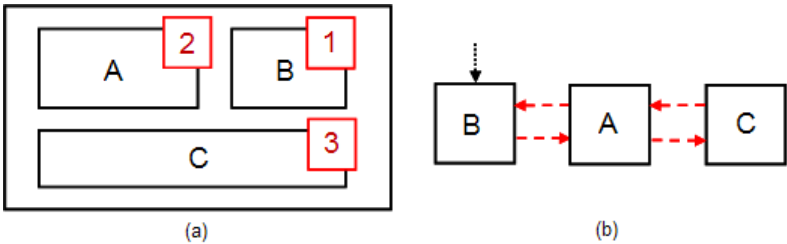


图 6.30. “用户定义焦点”控件功能的属性

在 [图6.30](#) 中，(a) 显示视图，而 (b) 显示焦点顺序。处理焦点更改的顺序，可能与小部件树结构不同。

当控件层次结构中的控件被标记为可聚焦时，它们是焦点层次结构的一部分。此焦点层次结构由可聚焦控件和焦点策略、“自动聚焦”控件功能或“用户定义的焦点”控件功能组成，定义了层次结构中焦点的处理方式。聚焦层次结构可以嵌套。

### 6.18.5.2. 列表管理控件功能类别

“行索引”和“模板索引”控件功能允许您将数据（例如图像，歌曲标题）连接到实例化器的动态创建的相应行模板中。

“行索引”

“行索引”控件功能，用来对实例化器控件进行自定义。“行索引”控件功能，定义了列表或表格中每行的唯一位置。



例 6.34.

“行索引”控件功能

如果要对列表建模，您可能希望列表的每个条目都有反映列表属性条目的特定值。要访问列表中的某个条目，行模板的实例需要知道是哪个实例的子节点。“行索引”控件功能添加了 `lineIndex` 属性。当实例化器创建行模板的实例时，它将使用值填充 `lineIndex`：索引首先以零开头。如果在实例化器中您有两个元素，第二个元素 `lineIndex` 取值为 1。

有关说明，请参见 [节11.4](#)。

“模板索引”

“模板索引”控件功能，允许复杂的数据抽象。对于非常复杂的列表或表，您需要多个数据列表可视化条目或一组条目。例如，具有混合图像和文本内容的表，需要图像列表和字符串列表。为了覆盖这种情况，“模板索引”控件功能提供了属性 `lineTemplateIndex`。



例 6.35.

“模板索引”控件功能

如果您使用的实例化器的属性 `lineMapping` 设定为 `0|1` 和属性 `numItems` 设定为 5，`lineTemplateIndex` 导致 `0|0|1|1|2`。



# 7. 对 HMI 行为进行建模

注意



默认窗口布局

本用户手册的所有说明和屏幕截图，都使用默认窗口布局。如果您想按照说明进行操作，建议您将 EB GUIDE Studio 窗口设置为默认 布局，方法是选择“布局” > “重置为默认布局”。

## 7.1. 对状态机进行建模

### 7.1.1. 添加状态机



添加状态机

#### 步骤 1

在“导航”元素中，转至“状态机”，然后点击 **+**。

菜单将展开。

#### 步骤 2

选择状态机的类型。

已添加所选类型的新状态机。

#### 步骤 3

重命名状态机。

### 7.1.2. 添加动态状态机

动态状态机与其他状态机并行运行，并且可在运行时刻启动（推送）和停止（弹出）。



添加动态状态机

例如，您可以使用动态状态机显示覆盖常规屏幕的错误消息。

先决条件：

- 已将一个状态机（视图状态或组合状态）添加到 EB GUIDE 模型。

#### 步骤 1

在“导航”元素中，转至“动态状态机”，然后点击 **+**。

菜单将展开。

#### 步骤 2

选择动态状态机的类型。

已添加所选类型的新动态状态机。

#### 步骤 3

在“导航”元素中，点击您希望与动态状态机并行运行的状态机、视图状态或组合状态。

#### 步骤 4

在“属性”元素中，选中 Dynamic state machine list 复选框。

完成上述步骤后，您可使用与动态状态机相关的 EB GUIDE 脚本 函数。

有关详细信息，请参见[节11.1](#)。

### 7.1.3. 为状态机定义进入动作



#### 为状态机定义进入动作

#### 步骤 1

选择状态机。

#### 步骤 2

在“属性”元素中，转至“进入动作”属性，并点击“添加”。

#### 步骤 3

使用 EB GUIDE 脚本 输入操作。

有关背景信息，请参见[节6.13](#)。

#### 步骤 4

点击“接受”。

已为状态机定义进入动作。

### 7.1.4. 为状态机定义退出动作



### 为状态机定义退出动作

#### 步骤 1

选择状态机。

#### 步骤 2

在“属性”元素中，转至 Exit action 属性，并点击“添加”。

#### 步骤 3

使用 EB GUIDE 脚本 输入操作。

有关背景信息，请参见[节6.13](#)。

#### 步骤 4

点击“接受”。

已为状态机定义退出动作。

## 7.1.5. 删除状态机



### 删除状态机

#### 步骤 1

在“导航”元素中，右键点击状态机。

#### 步骤 2

在上下文菜单中，点击“删除”。

状态机被删除。

## 7.2. 对状态进行建模

### 7.2.1. 添加状态



### 添加状态

先决条件：

- 内容区域将显示状态机。



步骤 1

将状态从“工具箱”拖入状态机中。

状态将添加至状态机。

注意



初始状态、结束状态和历史状态是唯一的  
您只能在每个组合状态中插入一个初始状态、结束状态和历史状态。

提示



复制和查找状态  
或者，您可以使用上下文菜单或 **Ctrl+C** 及 **Ctrl+V** 来复制并粘贴现有状态。  
  
要在 EB GUIDE 模型中查找特定状态，请在搜索框中输入状态的名称或使用 **Ctrl+F**。要跳过状态，请在命中列表中双击它。

## 7.2.2. 向组合状态添加状态



### 向组合状态添加状态

要创建状态层次结构，您可以将某个状态创建为另一个状态的子代。可以通过向组合状态中添加状态来完成此操作。

先决条件：

- 内容区域将显示状态机。
- 状态机包含组合状态。

步骤 1

在“导航”元素中，双击组合状态。

组合状态将在内容区域中扩展。

步骤 2

将状态从“工具箱”拖入组合状态中。

该状态将添加为组合状态的子状态。

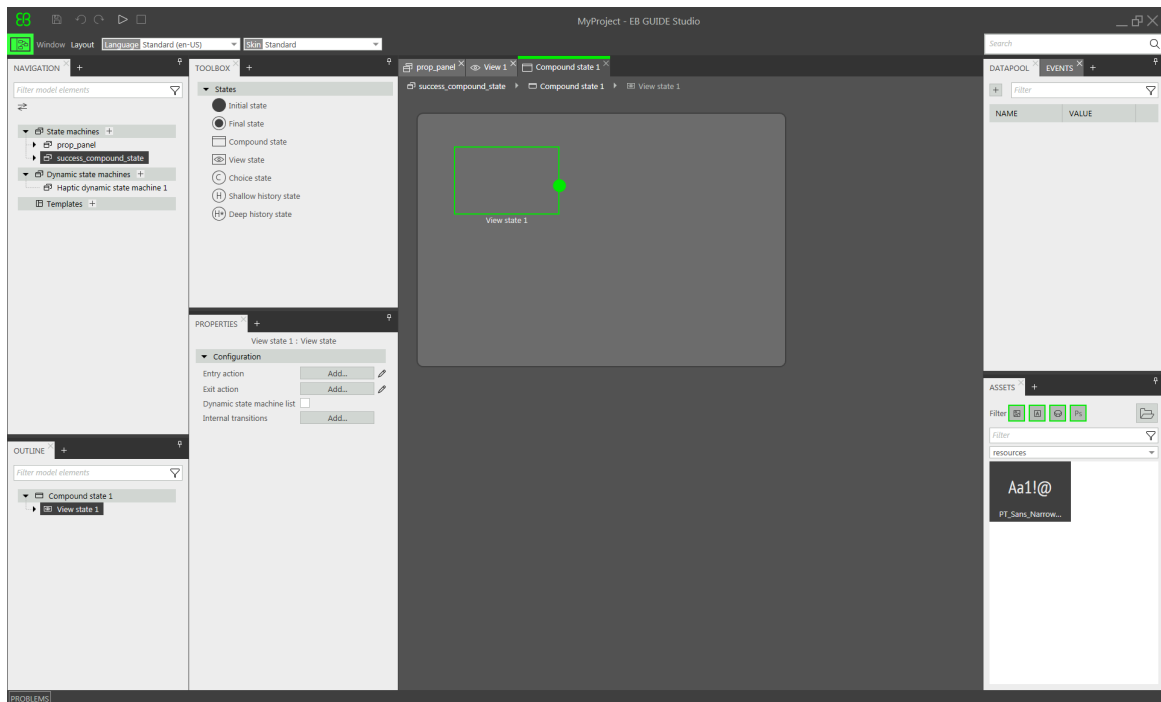


图 7.1. 具有嵌套视图状态的组合状态

### 7.2.3. 添加选择状态



#### 添加选择状态

先决条件：

- 内容区域将显示状态机。
- 状态机至少包含两种状态。

##### 步骤 1

将选择状态从“工具箱”拖入状态机中。

##### 步骤 2

添加自选择状态的传出转换。

##### 步骤 3

向传出转换添加条件。有关详细信息，请参见[节7.3.4](#)

为条件指定了优先级一。状态机进入选择状态时，将首先对具有优先级一的条件进行求值。

##### 步骤 4

要添加更多选择转换，请重复前两个步骤。



为新的选择转换指定的优先级低于先前创建的转换。

步骤 5

添加自选择状态的传出转换。

步骤 6

在“导航”元素中，右键点击转化。在上下文菜单中，点击“转换为其他”。

您已添加其他转换。当指定给传出选择转换的所有条件都求值为 false 时，将执行其他转换。

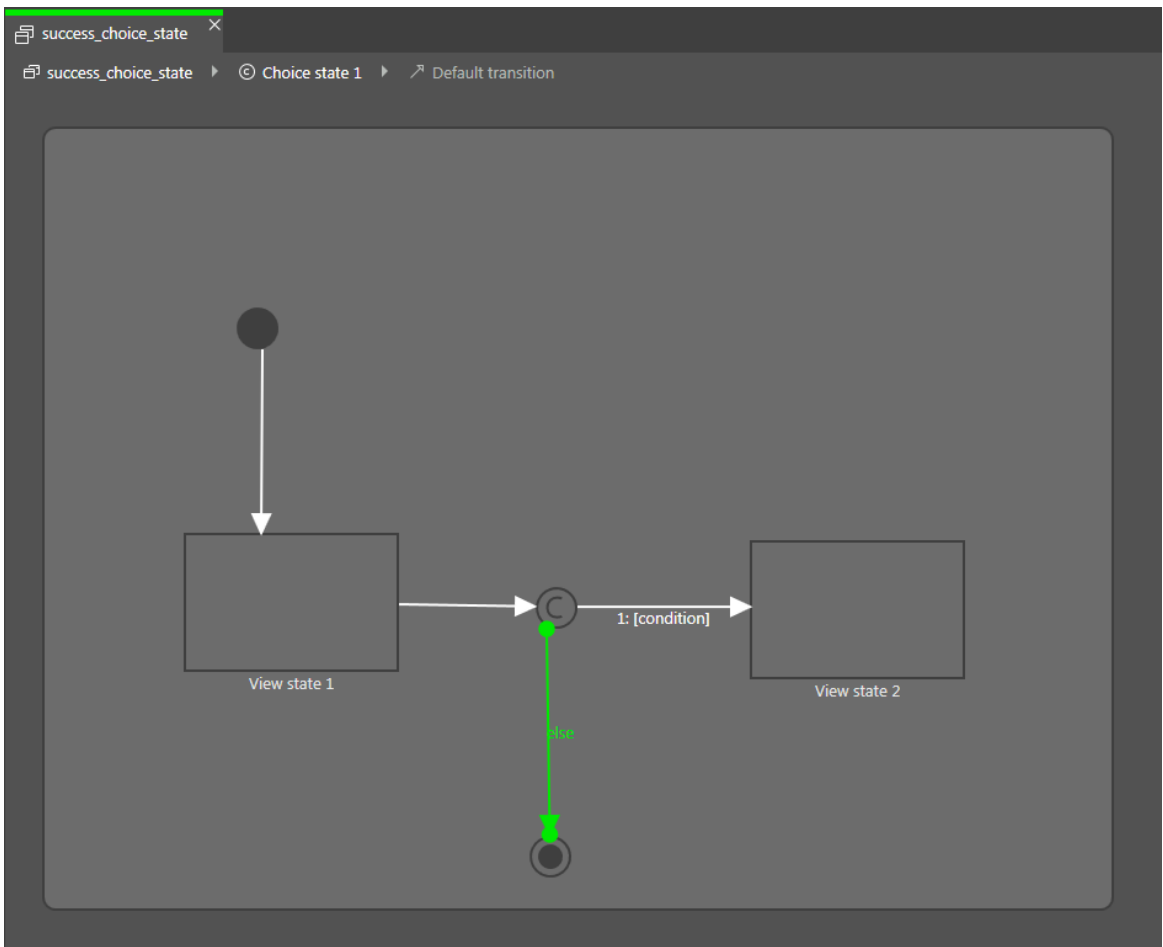


图 7.2. 选择状态及其选择转换

### 7.2.4. 为状态定义进入动作



为状态定义进入动作

对于视图状态和组合状态，您可以定义进入动作。进入动作将在每次进入状态时执行。

先决条件:

- 状态机包含视图状态或组合状态。

步骤 1

选中状态。

步骤 2

在“属性”元素中，转至 Entry action 属性，并点击“添加”。

步骤 3

使用 EB GUIDE 脚本 输入操作。

有关背景信息，请参见[节6.13](#)。

步骤 4

点击“接受”。

## 7.2.5. 为状态定义退出动作



### 为状态定义退出动作

对于视图状态和组合状态，您可以定义退出动作。退出动作将在每次退出状态时执行。

先决条件:

- 状态机包含视图状态或组合状态。

步骤 1

选中状态。

步骤 2

在“属性”元素中，转至 Exit action 属性，并点击“添加”。

步骤 3

使用 EB GUIDE 脚本 输入操作。

有关背景信息，请参见[节6.13](#)。

步骤 4

点击“接受”。

## 7.2.6. 从状态机中删除模型元素



#### 从状态机中删除模型元素

先决条件:

- 状态机至少包含一个模型元素。

##### 步骤 1

在“导航”元素中，右键点击模型元素。

##### 步骤 2

在上下文菜单中，点击“删除”。

模型元素将被删除。

## 7.3. 通过转换连接状态

### 7.3.1. 在两种状态之间添加转换



#### 在两种状态之间添加转换

通过转换，您可以将源状态连接到目标状态。

先决条件:

- 内容区域将显示状态机。
- 状态机至少包含两种状态。

##### 步骤 1

选中某个状态作为转换的源状态。

##### 步骤 2

点击绿色拖动点并按住鼠标键。

##### 步骤 3

将鼠标拖动至目标状态。

##### 步骤 4

目标状态以绿色突出显示时，松开鼠标键。

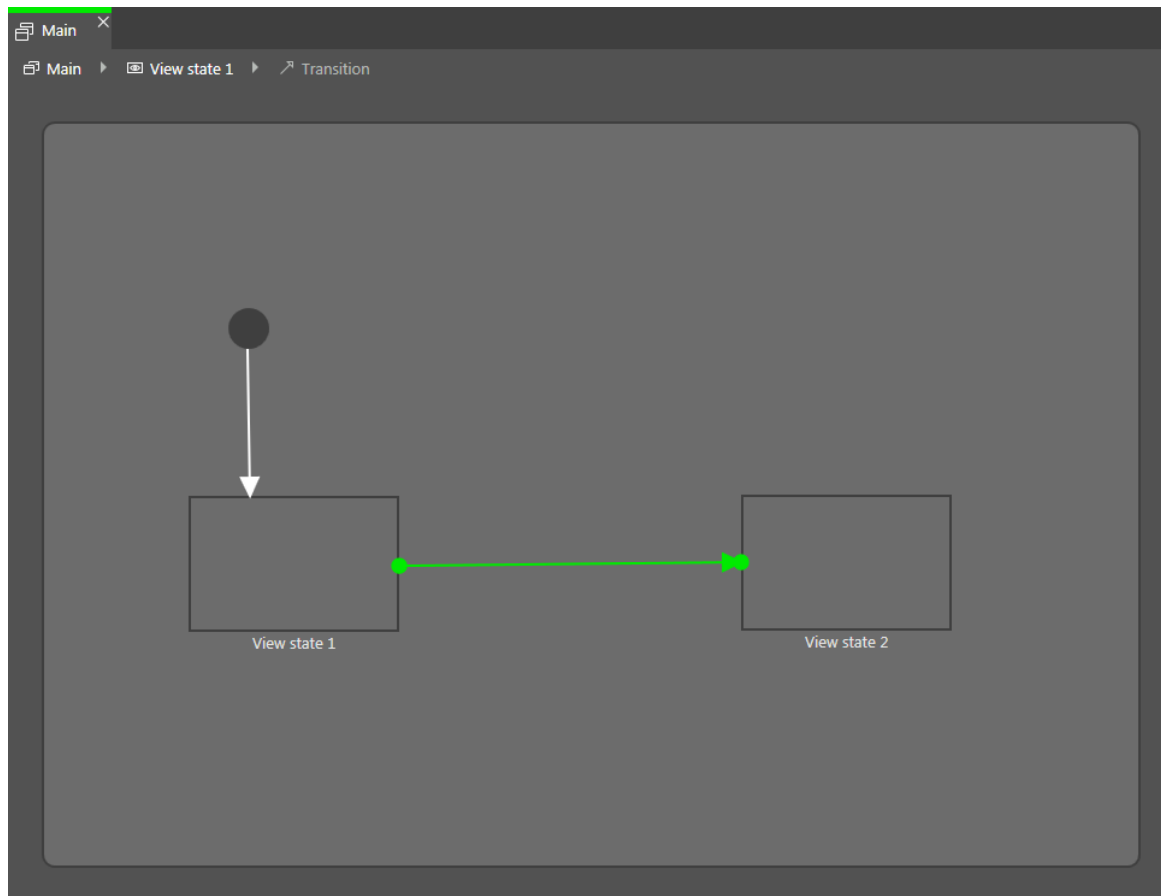


图 7.3. 转换

转换将添加并显示为绿色箭头。

提示



将转换连接到状态机

状态机是最顶层的组合状态。因此，您可以创建以状态机边界为目标或来源的转换。状态机中的所有状态都继承此类转换。

### 7.3.2. 移动转换



移动转换

通过移动转换的某个端点来移动转换。

先决条件：

- 内容区域将显示状态机。
- 状态机至少包含两种状态。
- 状态由转换进行连接。

#### 步骤 1

在内容区域中，点击某个转换。

将显示两个绿色拖动点。

#### 步骤 2

点击您要移动的拖动点并按下鼠标键。

#### 步骤 3

将鼠标拖动至另一个状态。

#### 步骤 4

状态以绿色突出显示时，松开鼠标键。

转换将进行移动。

### 7.3.3. 为转换定义触发器



#### 为转换定义触发器

对于转换，您可以定义触发该转换的事件。

先决条件：

- 状态机至少包含两种状态。
- 状态由转换进行连接。

#### 步骤 1

选中转换。

#### 步骤 2

在“属性”元素中，展开“触发器”组合框。

#### 步骤 3

选中事件。

#### 步骤 4

要创建新事件，请在“触发器”组合框中输入名称，并点击“添加事件”。

事件将添加为转换触发器。

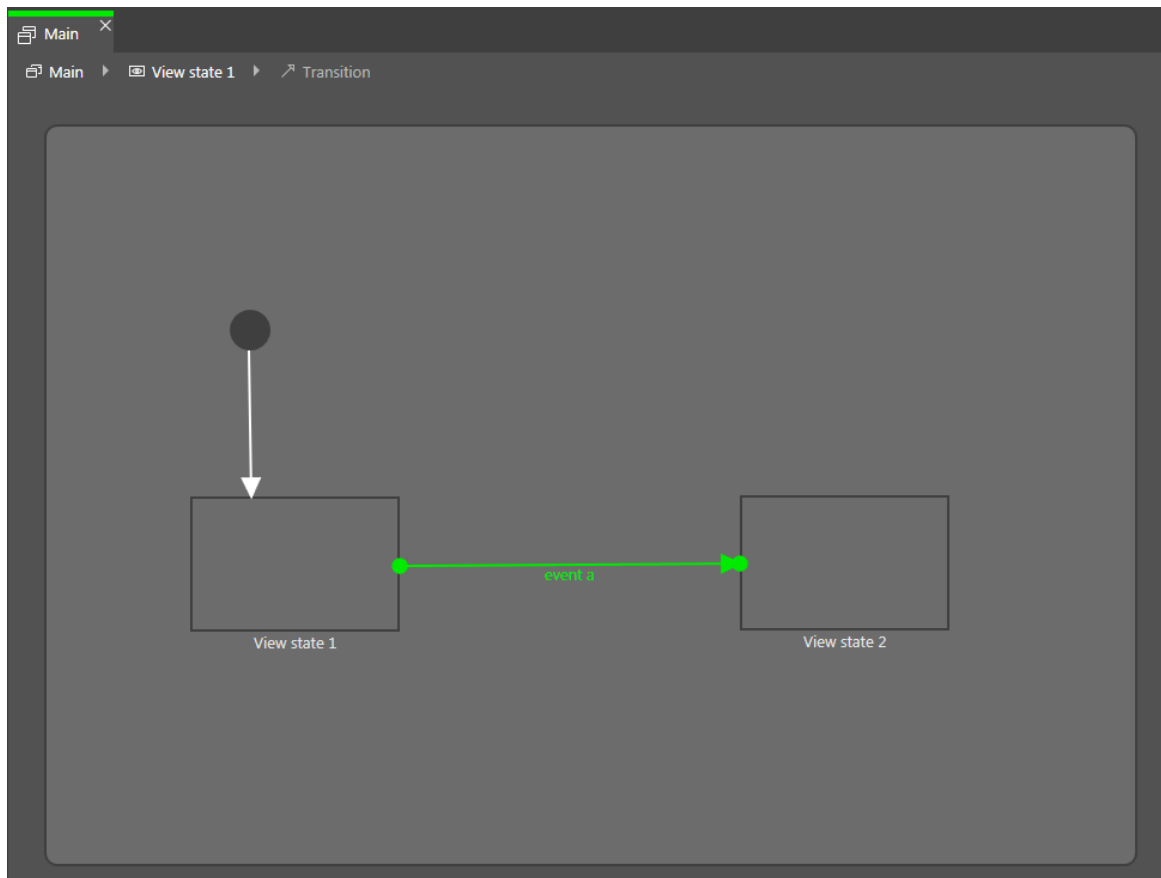


图 7.4. 具有触发器的转换

### 7.3.4. 向转换添加条件



#### 向转换添加条件

对于每个转换，您可以定义必须满足才能执行该转换的条件。

先决条件：

- 状态机至少包含两种状态。
- 状态由转换进行连接。

#### 步骤 1

选中转换。

#### 步骤 2

要向转换添加条件，请转至“属性”元素。在 Condition 属性旁边，点击“添加”。

**步骤 3**

使用 EB GUIDE 脚本 输入条件。

有关背景信息，请参见[节6.13](#)。

**步骤 4**

点击“接受”。

条件将添加至转换。

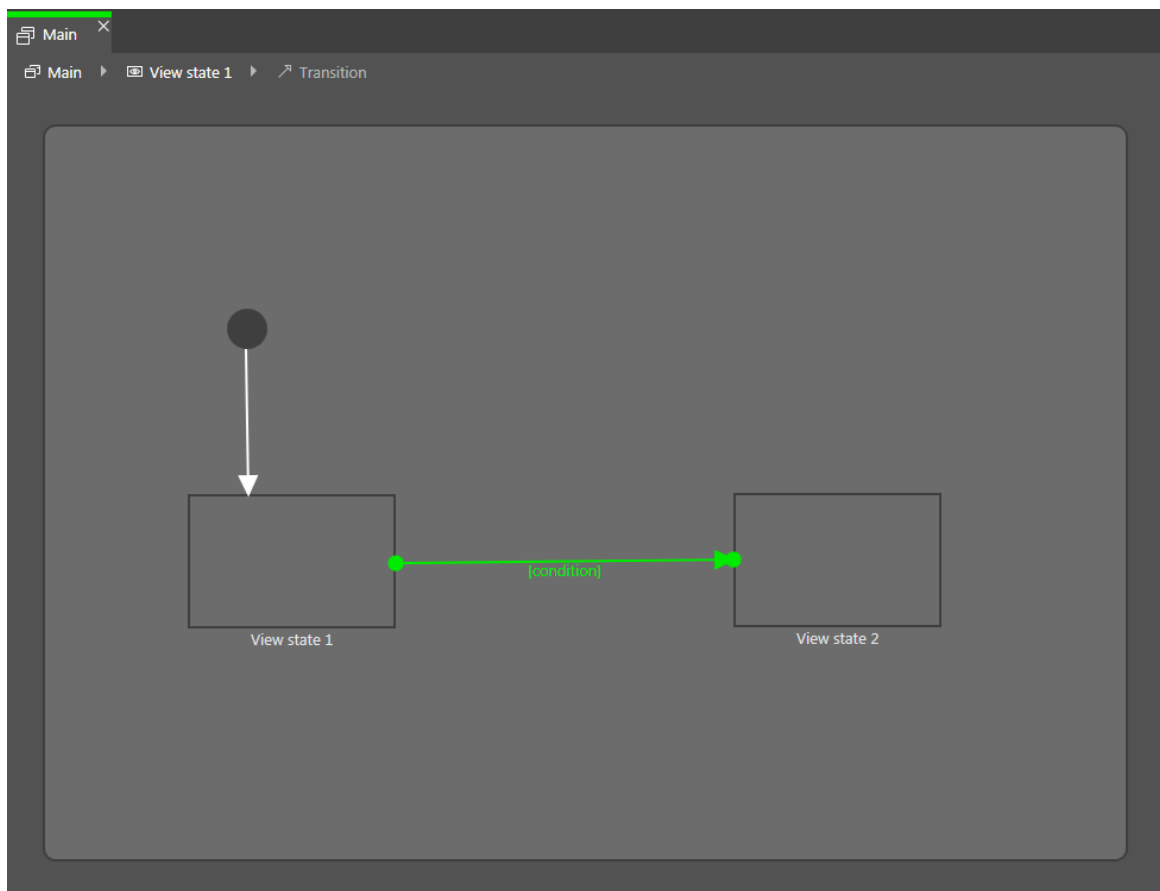


图 7.5. 具有条件的转换

### 7.3.5. 向转换添加操作



#### 向转换添加操作

对于每个转换，您可以定义随该转换执行的操作。

先决条件:

- 状态机至少包含两种状态。
- 状态由转换进行连接。

#### 步骤 1

选中转换。

#### 步骤 2

要向转换添加操作，请转至“属性”元素。在 Action 属性旁边，点击“添加”。

#### 步骤 3

使用 EB GUIDE 脚本 输入操作。

有关背景信息，请参见[节6.13](#)。

#### 步骤 4

点击“接受”。

操作将添加至转换。

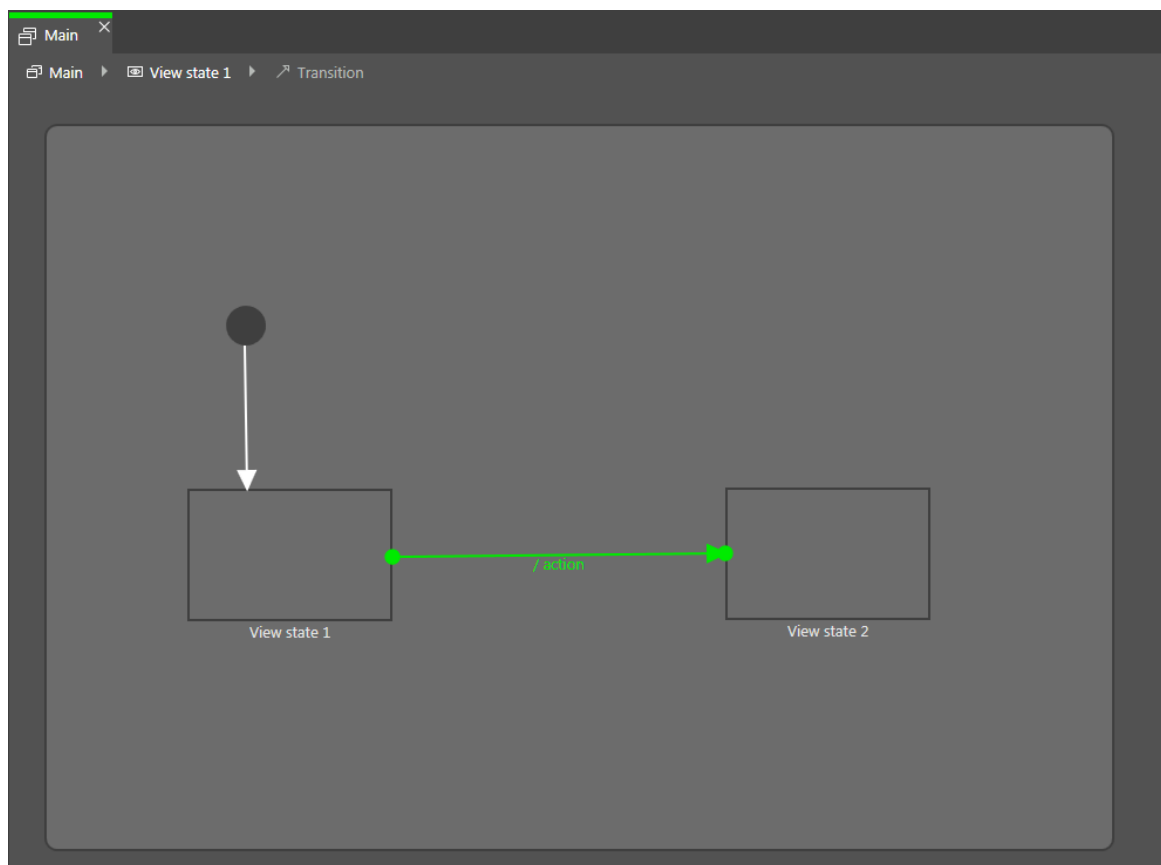


图 7.6. 具有操作的转换



## 7.3.6. 向状态添加内部转换



### 向状态添加内部转换

先决条件：

- 状态机包含一种状态。

#### 步骤 1

选中状态。

#### 步骤 2

在“属性”元素中，转至“内部转换”，并点击“添加”。

将向状态添加内部转换。内部转换在“导航”元素内可见。



## 8. 对 HMI 外观进行建模

注意



默认窗口布局

本用户手册的所有说明和屏幕截图，都使用默认窗口布局。如果您想按照说明进行操作，建议您将 EB GUIDE Studio 窗口设置为默认 布局，方法是选择“布局” > “重置为默认布局”。

### 8.1. 使用控件

提示



复制、查找视图和控件

您可以使用上下文菜单或 **Ctrl+C** 及 **Ctrl+V** 来复制并粘贴现有视图或控件。

要在 EB GUIDE 模型中查找特定控件，请在搜索框中输入视图或控件的名称或使用 **Ctrl+F**。要跳过视图或控件，请在命中列表中双击它。

#### 8.1.1. 添加视图



添加视图

先决条件：

- 内容区域将显示状态机。

步骤 1

将视图状态从“工具箱”拖入状态机中。

视图将随视图状态添加至模型。

步骤 2

在“导航”元素中，点击视图。

步骤 3

按 **F2** 键并重命名视图。

步骤 4

双击内容区域中的视图状态。

内容区域显示新的视图。

## 8.1.2. 向视图添加基本控件

关于基本控件的详细信息，请参见 [节12.10.2](#)。

### 8.1.2.1. 添加矩形



添加矩形

先决条件：

- 内容区域将显示视图。

#### 步骤 1

将矩形从“工具箱”拖入视图中。

矩形将添加至视图。

### 8.1.2.2. 添加椭圆



添加椭圆

先决条件：

- 内容区域将显示视图。

#### 步骤 1

将椭圆从“工具箱”拖入视图中。

控件将添加至视图。

#### 8.1.2.2.1. 编辑椭圆

您可以绘制一个椭圆的扇形，您可以更改椭圆的弧。



创建一个循环扇区

先决条件：

- 该视图包含椭圆。

**步骤 1**

单击椭圆并转至“属性”元素。

**步骤 2**

在 centralAngle 文本框中，输入扇区的角度。

**步骤 3**

在 sectorRotation 文本框中，输入扇区的方向。

您创建了一个循环扇区。

**创建圆弧**

先决条件：

- 该视图包含椭圆。

**步骤 1**

单击椭圆并转至“属性”元素。

**步骤 2**

要定义圆弧的宽度，请在 arcWidth 文本框中输入值。

您创建了一个圆弧。

### 8.1.2.3. 添加图像

**用“工具箱”添加图像**

先决条件：

- 图像文件位于 \$GUIDE\_PROJECT\_PATH/<project name>/resources 目录中。有关受支持的文件类型，请参见 [节6.12.2](#)。
- 内容区域将显示视图。

**步骤 1**

将图像从“工具箱”拖入视图中。

**步骤 2**

从“属性”元素的 image 组合框中选择一个图像。或者将另一个图像从“资产”元素拖入 image 下拉列表框中。

视图将显示该图像。



### 使用“资产”元素添加图像

先决条件:

- 图像文件位于 `$GUIDE_PROJECT_PATH/<project name>/resources` 目录中。有关受支持的文件类型，请参见[节6.12.2](#)。
- 内容区域将显示视图。

#### 步骤 1

将图像从“资产”元素拖入视图中。

视图将显示该图像。

#### 步骤 2

要更改图像文件，请转至“属性”元素并从 image 组合框中选择一个图像。或者将另一个图像从“资产”元素拖入 image 组合框中。

视图将显示该图像。



### 添加 9-patch 图像

先决条件:

- 9-Patch 图像文件位于 `$GUIDE_PROJECT_PATH/<project name>/resources` 目录中。有关 9-patch 图像的背景信息，请参见[节6.12.2.1](#)。
- 内容区域将显示视图。
- 图像将添加至 EB GUIDE 模型。

#### 步骤 1

选择图像，并转至“属性”元素。

#### 步骤 2

从 image 组合框中，选择 9-patch 图像。

#### 步骤 3

转至“控件功能属性”类别中，点击“添加/删除”。

这将显示“控件功能”对话框。

#### 步骤 4

在“可用控件功能”下，展开“布局”类别，然后选择“缩放模式”。

#### 步骤 5

点击“接受”。

相关控件功能属性将添加至图像，并显示在“属性”元素中。

#### 步骤 6

在“属性”元素中，对 `scaleMode` 属性选择 `fit to Size (=1)`。

#### 注意



#### 添加 9-patch 图像

如果您不添加“缩放模式”控件功能，或者对于 `scaleMode` 属性，您选择 `original Size (=0)` 或 `keep aspect ratio (=2)`，9-patch 图像被缩放到和正常 .png 图像一样。

### 8.1.2.4. 添加一个标签



#### 用“工具箱”添加标签

先决条件：

- 内容区域将显示视图。

#### 步骤 1

将标签从“工具箱”拖入视图中。

标签将添加至视图。标签有默认字体 `PT_Sans_Narrow.ttf`。



#### 用“资产”元素添加标签

先决条件：

- 文字文件位于 `$GUIDE_PROJECT_PATH/<project name>/resources` 目录中。有关受支持的文件类型，请参见 [节 6.12.1](#)。
- 内容区域将显示视图。

#### 步骤 1

将字体文件从“资产”元素拖入视图中。

视图将以所选字体显示标签。

#### 8.1.2.4.1. 更改标签的字体



## 更改标签的字体

先决条件:

- 文字文件位于 `$GUIDE_PROJECT_PATH/<project name>/resources` 目录中。有关受支持的文件类型，请参见[节6.12.1](#)。
- EB GUIDE 模型包含视图状态。
- 视图包含标签。

### 步骤 1

选中视图中的标签。

### 步骤 2

从“属性”元素的 font 组合框中选择一个字体。

或者将一个字体文件从“资产”元素拖入 font 组合框中。

视图将以新字体显示标签。请注意，如果您选择了 .fnt 位图字体，该字体的大小是固定的，并且您无法在标签的 font 属性中更改它。

### 注意



计算文本高度和线间距

下图显示如何在 EB GUIDE Studio 中计算文本高度、行高和线间距。改变字样、字号或标签的线间距时，应考虑到这个。

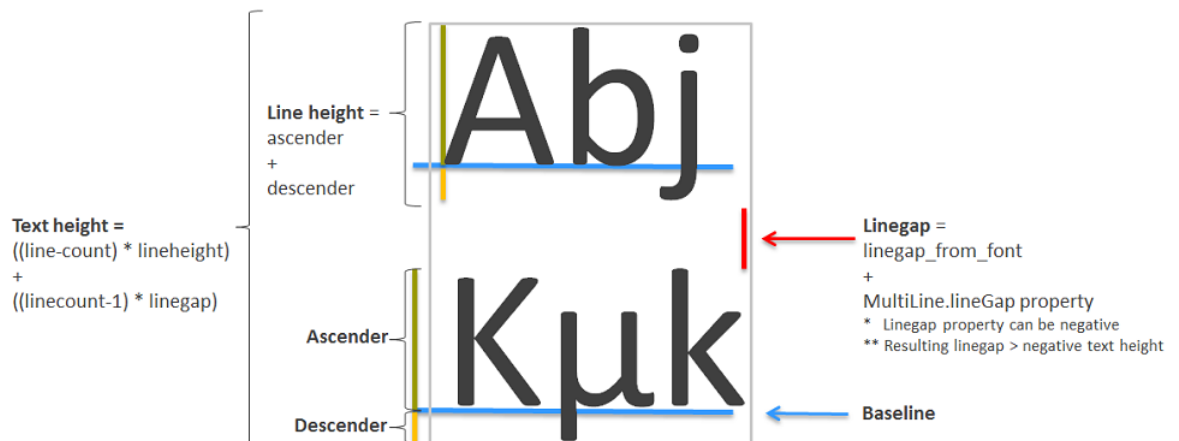


图 8.1. 文本高度、行高和线间距的计算

## 8.1.2.5. 添加容器



#### 添加容器

容器允许使用分组控件。

先决条件：

- 内容区域将显示视图。

##### 步骤 1

将容器从“工具箱”拖入视图中。

##### 步骤 2

在内容区域中，通过拖动容器的一个角来将其放大。

##### 步骤 3

将两个或两个以上控件从“工具箱”拖入容器中。

控件将作为容器的子代控件进行建模。移动容器会同时移动其子代控件。

### 8.1.2.6. 添加实例化器



#### 添加实例化器

先决条件：

- 内容区域将显示视图。

##### 步骤 1

将实例化器从“工具箱”拖入视图中。

##### 步骤 2

将控件从“工具箱”拖入实例化器中。

控件作为行模板。

##### 步骤 3

选择实例化器，并转至“属性”元素。

##### 步骤 3.1

在 numItems 属性中输入大于 1 的值。

##### 步骤 3.2

向实例化器中添加下列控件功能之一：

- ▶ “方框布局”
- ▶ “流式布局”



- ▶ “栅格布局”
- ▶ “列表布局”

有关详细信息，请参见[节8.3.1](#)。

在视图中，子代控件将按实例化器控件指定的布局，显示 numItems 属性所指定的次数。

#### 步骤 4


将控件从“工具箱”拖入实例化器中。

您添加第二个子代控件，作为第二个行模板。

#### 步骤 5

选择实例化器，并转至“属性”元素。

##### 步骤 5.1

在 lineMapping 属性旁边，点击 。

##### 步骤 5.2

点击“添加”按钮。

新条目已添加至表。

##### 步骤 5.3

在 Value 文本框中，输入 0。

##### 步骤 5.4

点击“添加”按钮。

新条目已添加至表。

##### 步骤 5.5

在 Value 文本框中，输入 1。

您定义了行模板实例化的顺序。



#### 例 8.1. 实例化顺序

lineMapping 属性定义了实例化顺序。比如，如果您输入值 1|0，实例化器实例化行模板 1 作为第一个自带空间，行模板 0 作为第二个子代模板。

lineMapping 属性被迭代应用。意味着，对于 numItems 属性，如果您输入 10，结果是顺序 1|0|1|0|1|0|1|0|1|0。

有关如何使用实例化器的详细示例，请参见[节11.4](#)。

注意



连接行模板的属性

以下是链接规则：

- ▶ 您不可链接行模板间的属性。
- ▶ 您不能从实例化器的外部链接到其线模板。
- ▶ 您可以从行模板链接到相应的实例化器。

### 8.1.2.7. 添加动画



添加动画

有关曲线的详细信息以及曲线属性的说明，请参见[节12.10.2.2](#)。

先决条件：

- 内容区域将显示视图。

步骤 1

将基本控件之一从“工具箱”拖入视图中。

步骤 2

将动画从“工具箱”拖入添加的控件中。

步骤 3

转至“动画”编辑器，并点击“动画属性”旁边的 **+**

菜单将展开。

步骤 4

在“动画属性”下，选择您要生成动画的属性并在“动画曲线”下选择相应的曲线。

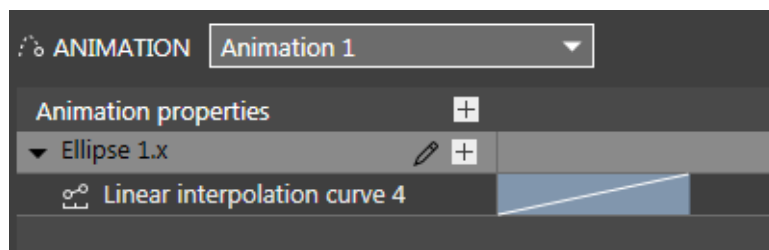


图 8.2. “动画”编辑器和示例曲线

步骤 5

选择基本控件并添加一个 Conditional script 类型的用户定义属性。有关详细信息，请参见[节8.2.5](#)。

#### 步骤 6

在属性名称旁的“属性”元素中点击“编辑”。

将打开脚本编辑器。

#### 步骤 7

输入下列 EB GUIDE 脚本：

```
function(v:arg0::bool)
{
  f:animation_play(v:this->"Animation 1")
}
```

“动画 1”是第一个添加的动画的默认名称。如果您在第二步中添加的动画有不同的名称，请替换“On trigger”脚本中的名称。

#### 步骤 8

启动模拟运行。

控件的链接属性将按照您所添加的曲线指定的方式逐渐更改。

作为后续步骤，您可以更改动画或曲线的属性。

有关具体动画示例，请参见[节11.5](#)。

### 8.1.2.8. 添加 alpha 掩码



#### 添加 alpha 掩码

有关 alpha 掩码的详细信息，请参见[节12.10.2.1](#)。

先决条件：

- \$GUIDE\_PROJECT\_PATH/<project name>/resources 目录包含一个图像。
- 内容区域将显示视图。

#### 步骤 1

将 alpha 掩码从“工具箱”拖入视图中。

#### 步骤 2

转至“属性”元素，并从 image 下拉列表框中选择一个图像。

## 注意



对于 **alpha** 掩码，支持的图像文件类型  
可用的图像格式取决于渲染器的实现。DirectX 11 和 OpenGL ES 版本 2.0 或更高版本支持 .png 文件和 .jpg 文件。RGB 图像会在用作 **alpha** 掩码之前转换为灰度图像。灰度图像将按原样使用。将忽略图像中的 **alpha** 通道。

您无法将 **alpha** 掩码与 9-patch 图像一起使用。

## 步骤 3

向 **alpha** 掩码中添加“工具箱”中的基本控件之一作为子控件。

**alpha** 通道（即，子控件的不透明度）使用 **alpha** 掩码进行控制。

## 8.1.3. 向一个 3D 控件到视图

### 8.1.3.1. 向视图添加场景图



#### 向视图添加场景图

有关限制和建议，请参见[节6.1.2](#)。

先决条件：

- 3D 图形文件可用。该文件包含摄像头、光源和对象（包含网格和至少一种材质）。有关受支持的 3D 图形文件格式，请参见[节6.1.1](#)。
- 内容区域将显示视图。

## 步骤 1

将场景图从“工具箱”拖入视图中。

视图显示空的边界框。

## 步骤 2

在“属性”元素中，点击“导入文件”。

对话框将打开。

## 步骤 3

转至储存 3D 图形文件的目录。

## 步骤 4

选择 3D 图形文件。

## 步骤 5

点击“打开”。

导入开始。对话框将打开。

#### 步骤 6

点击“确定”。

视图将显示该 3D 图形。“导航”元素显示导入的控件树，其中场景图作为父节点。如导入的 3D 场景具有动画，则添加线性键值插值整数或线性键值插补浮点曲线。注意：您不能修改 EB GUIDE Studio 中曲线的基本键值对。

#### 提示



#### 多重导入

可以在一个场景图中导入多个 3D 图形。

导入后，多个 3D 图形在彼此的顶部渲染。要单独显示 3D 对象，请使用 RootNode 的 visible 属性。

### 8.1.4. 添加 .psd 文件到视图中



#### 添加 .psd 文件到视图中

先决条件：

- 在 \$GUIDE\_PROJECT\_PATH/<project name>/resources 中，.psd 可用。有关背景信息，请参见[节6.12.4](#)。
- 内容区域将显示视图。

#### 步骤 1

在“资产”元素中，选择 resource 文件夹。

#### 步骤 2

将 .psd 文件从预览区域拖入内容区域中。

显示导入状态信息。

#### 步骤 3

点击“确定”。

如果导入成功，“导航”元素显示从 .psd 文件创建的控件树。控件树包含容器和图像，显示了 .psd 文件结构。在 \$GUIDE\_PROJECT\_PATH/<project name>/resources 目录中，生成一个包含所有解压图片的子目录。



#### 从 .psd 文件中提取图像

您可以从 .psd 文件中提取图像，而无需导入该文件。这意味着，控件树不是创建的。

先决条件:

- 在 `$GUIDE_PROJECT_PATH/<project name>/resources` 中, .psd 可用。有关背景信息, 请参见[节6.12.4](#)。
- 内容区域将显示视图。

#### 步骤 1

在“资产”元素中, 右键单击 .psd 文件, 并选择“从 .psd 文件中生成图像”。

在 `$GUIDE_PROJECT_PATH/<project name>/resources` 目录中, 生成一个包含所有解压图片的子目录。

## 8.1.5. 从视图中删除控件



### 从视图中删除控件

先决条件:

- EB GUIDE 模型包含控件。

#### 步骤 1

在“导航”元素中, 右键单击某个控件。

#### 步骤 2

在上下文菜单中, 点击“删除”。

该控件将被删除。

#### 提示



从内容区域中删除控件

还可以通过在内容区域中选中某个控件, 然后按删除键来删除该控件。

## 8.2. 使用控件属性

### 8.2.1. 确定控件的位置



### 确定控件的位置

确定控件的位置表示调整该控件的  $x$  和  $y$  属性。原点（ $x$  和  $y$  在此位置的值同时为 0）在父控件的左上角。

先决条件：

- 内容区域将显示视图。
- 视图包含控件。

#### 步骤 1

选中一个控件。

“属性”元素显示了所选控件的属性。

#### 步骤 2

要定义控件的  $x$  坐标，请在  $x$  文本框中输入值。

#### 步骤 3

要定义控件的  $y$  坐标，请在  $y$  文本框中输入值。

#### 步骤 4

在文本框外点击。

内容区域会将控件显示在所输入的位置。

#### 提示



#### 替代方法

要通过目视判断来定位控件，请在内容区域中选中该控件并使用鼠标移动它。

## 8.2.2. 调整控件的大小



### 调整控件的大小

先决条件：

- 内容区域将显示视图。
- 视图包含控件。

#### 步骤 1

选中一个控件。

“属性”元素显示了所选控件的属性。

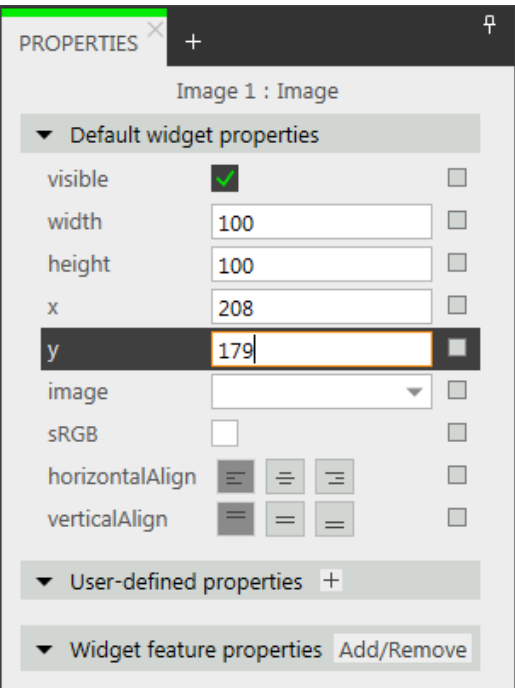


图 8.3. 图像的属性

- 步骤 2


要定义控件的高度，请在 height 文本框中输入值。
- 步骤 3

要定义控件的宽度，请在 width 文本框中输入值。
- 步骤 4

在文本框外点击。

内容区域会以所输入的大小显示控件。


注意



负值

请勿将负值用于 height 和 width 属性。EB GUIDE Studio 将负值视为 0，这意味着不会描述相关控件。

提示



替代方法

要通过目视判断来调整控件的大小，请在内容区域中选中该控件并使用鼠标拖动它的其中一个拐角。

### 8.2.3. 控件属性之间的链接





### 控件属性之间的链接

为了确保两个控件属性始终具有相同值，您可以链接这两个控件属性。例如，以下指令显示如何将矩形的 width 属性链接到视图的 width 属性。

您只能链接同一视图中的控件的属性

您不能链接到实例化器的子控件的属性。

先决条件：


- EB GUIDE 模型包含视图状态。
- 视图包含矩形。
- 矩形的 width 属性不是脚本化的值。

#### 步骤 1

点击该矩形。

“属性”元素显示该矩形的属性。

#### 步骤 2

在“属性”元素中，转至 width 属性，并点击属性旁的  按钮。

菜单将展开。

#### 步骤 3

在菜单中，点击“向控件属性中添加链接”。

对话框将打开。

#### 步骤 4

在此对话框中，转至视图并选择其 width 属性。

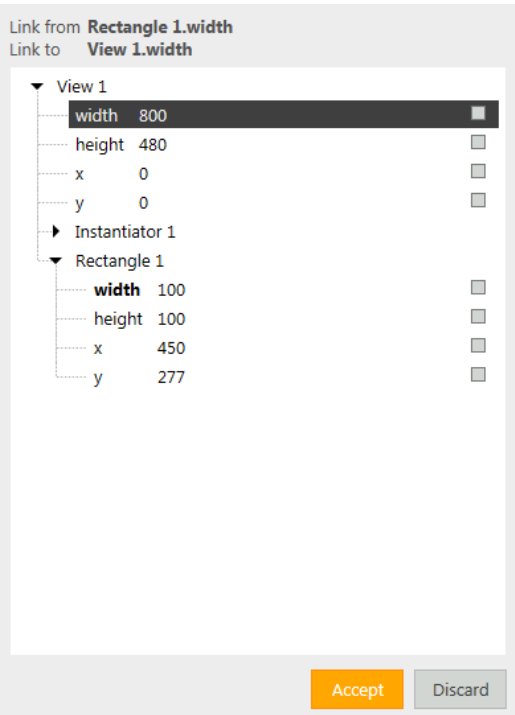





图 8.4. 控件属性之间的链接

步骤 5  
点击“接受”。

对话框将关闭。 按钮将显示在 width 属性旁边。此按钮表示，矩形控件的 width 属性现在已链接到视图的 width 属性。只要您更改视图的宽度，矩形的宽度就会发生更改，反之亦然。

注意



链接源和链接目标  
 按钮仅显示在链接源旁边。不会对链接目标显示该按钮。

提示



删除链接  
要删除链接，请再次点击  按钮。在打开的菜单中，点击“删除链接”。

### 8.2.4. 将控件属性链接到数据池项



### 将控件属性链接到数据池项

为了确保控件属性和数据池项始终具有相同值，您可以将控件属性链接到数据池项。例如，以下指令显示了如何将图像的 image 属性链接到新数据池项。

先决条件：


- EB GUIDE 模型包含视图状态。
- 该视图包含图像。
- 图像的 image 属性不是脚本化的值。

#### 步骤 1

点击该图像。

“属性”元素显示该图像的属性。

#### 步骤 2

在“属性”元素中，转至 image 属性，并点击属性旁的  按钮。

菜单将展开。

#### 步骤 3

在菜单中，点击“向数据池项中添加链接”。

对话框将打开。

#### 步骤 4

要添加新的数据池项，请在文本框中输入名称。

#### 步骤 5

点击“添加数据池项”。

#### 步骤 6

点击“接受”。

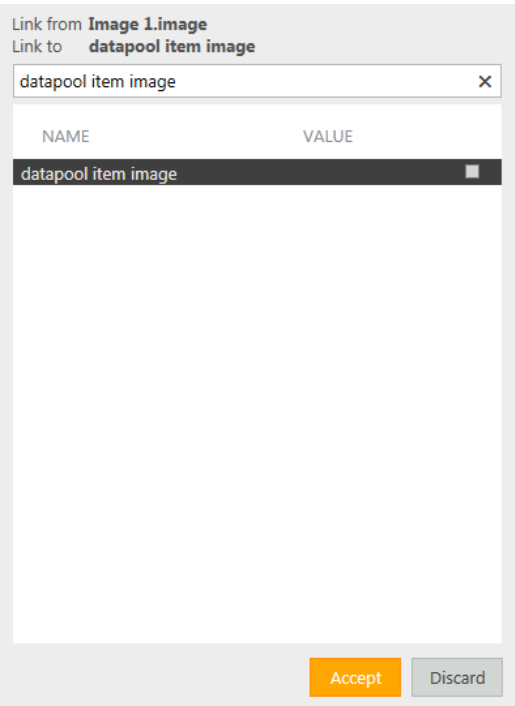


图 8.5. 链接到数据池项

已添加新的数据池项。

步骤 7

对话框将关闭。■ 按钮将显示在 image 属性旁边。此按钮表示 image 属性已链接到数据池项。只要您更改了图像，数据池项就会更改，反之亦然。

注意



链接源和链接目标

■ 按钮仅显示在链接源旁边。不会对链接目标显示该按钮。

提示



删除链接

要删除链接，请再次点击 ■ 按钮。在打开的菜单中，点击“删除链接”。

## 8.2.5. 向控件添加用户定义的属性



#### 向控件添加用户定义的属性

先决条件:

- EB GUIDE 模型包含视图状态。
- 视图包含控件。

##### 步骤 1

选中一个控件。

“属性”元素显示了所选控件的属性。

##### 步骤 2

在“属性”元素中，转至“用户定义的属性”类别，并点击 **+**。

菜单将展开。

##### 步骤 3

在菜单中，针对用户定义的属性点击类型。

已向控件添加所选类型的新控件属性。

##### 步骤 4

重命名该属性。

### 8.2.5.1. 添加用户定义的以下类型的属性 `Function () : bool`



#### 添加用户定义的以下类型的属性 `Function () : bool`

`Function () : bool` 类型的属性是一个函数，它没有参数并且将返回布尔值。您可以在 EB GUIDE 脚本中按照对后跟参数列表的控件属性进行寻址的方式调用此函数。

先决条件:

- EB GUIDE 模型包含视图状态。
- 视图包含控件。

##### 步骤 1

选中一个控件。

“属性”元素显示了所选控件的属性。

##### 步骤 2

在“属性”元素中，转至“用户定义的属性”类别，并点击 **+**。

菜单将展开。

#### 步骤 3

在菜单中，点击 `Function (): bool`。

已向控件添加 `Function (): bool` 类型的新控件属性。

#### 步骤 4

重命名该属性。

#### 步骤 5

点击该属性旁边的“编辑”。

将打开脚本编辑器。

#### 步骤 6

使用 **EB GUIDE** 脚本 定义新函数的行为。

#### 步骤 7

点击“接受”。



#### 例 8.2.

调用 ..... 类型的属性 `Function (): bool`

在 **EB GUIDE** 模型中，有一个名为 `Background color` 的矩形。已向其中添加了 `Function (): bool` 类型的属性。此属性名为 `change`。

在 **EB GUIDE** 模型中的任何 **EB GUIDE** 脚本 代码中，您可以在该属性中调用脚本，如下所示：

```
"Background color".change()
```

## 8.2.6. 重命名用户定义的属性



### 重命名用户定义的属性


先决条件：

- **EB GUIDE** 模型包含具有用户定义属性的控件。

#### 步骤 1

在“导航”元素中，选中具有用户定义属性的控件。

#### 步骤 2

在“属性”元素中，点击属性旁的  按钮。

菜单将展开。

**步骤 3**

在菜单中，点击“重命名”。

**步骤 4**

输入该属性的名称。

**步骤 5**

按 Enter 键。

## 8.3. 按控件功能扩展控件

控件功能为控件的外观和行为增加了更多功能。向控件添加控件功能表示添加一个或多个控件属性。所提供的控件功能依赖于控件的类型。

### 8.3.1. 添加控件功能



#### 添加控件功能

先决条件：

- EB GUIDE 模型包含控件。

**步骤 1**

在“导航”元素中，点击控件。

“属性”元素显示了所选控件的属性。

**步骤 2**

在“属性”元素中，转至“控件功能属性”类别，并点击“添加/删除”。

这将显示“控件功能”对话框。

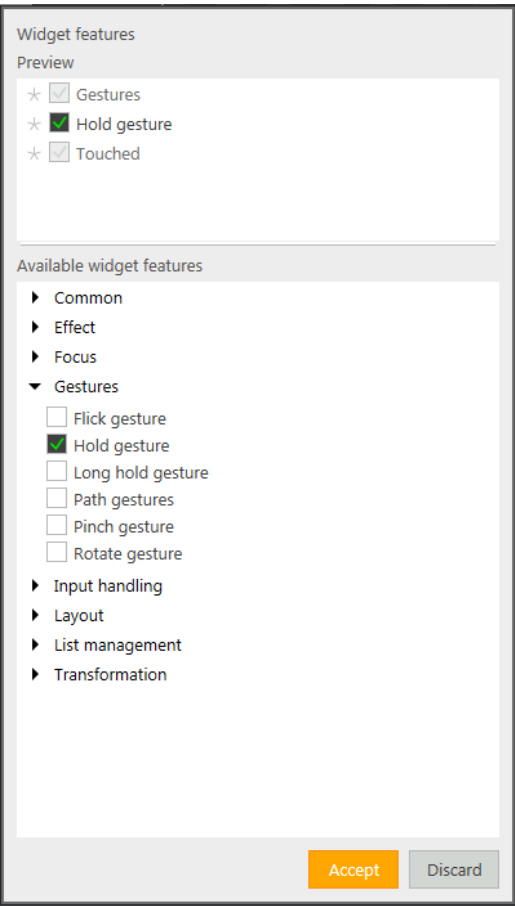


图 8.6. 控件功能对话框

步骤 3

在“可用控件功能”下，展开类别并选择您要添加的控件功能。

所选控件功能及自动随其激活的依赖项控件功能列示在“预览”下。

点击“接受”。

提示



控件功能之间的依赖关系

一些控件功能需要其他控件功能。因此，在一些情况下，如果您选择了某个控件功能，则其他控件功能将被自动选中。

例如，您要添加控件功能“可移动”。另外，还将自动添加控件功能“已触摸”和“触摸移动”。

有关按类别分组的控件功能的列表，请参见[节12.11](#)。

有关教程，请参见下列内容：

► [节11.3](#)



▶ [节11.4](#)

▶ [节11.2](#)

### 8.3.2. 删除控件功能



#### 删除控件功能

先决条件:

- EB GUIDE 模型包含控件。
- 至少将向控件中添加一个控件功能。

##### 步骤 1

在“导航”元素中，点击控件。

“属性”元素显示了所选控件的属性。

##### 步骤 2

在“属性”元素中，转至“控件功能属性”类别，并点击“添加/删除”。

这将显示“控件功能”对话框。

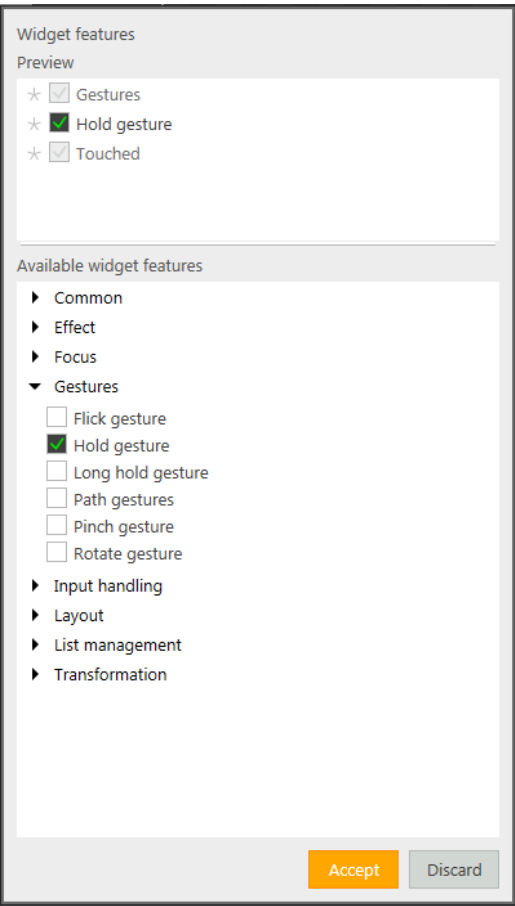


图 8.7. 控件功能对话框

**步骤 3**  
在“预览”下，清除您要删除的控件功能。  
  
点击“接受”。

相关控件功能属性将从“属性”元素中删除。

注意



删除具有依赖关系的控件功能  
不会自动删除由于存在依赖关系而自动添加的控件功能。无法直接删除它们。在清除子代控件功能之前，请清除父代控件功能。

## 8.4. 向 EB GUIDE 模型中添加语言

要在运行时刻启用语言支持，您需要将语言添加至 EB GUIDE 模型。



### 8.4.1. 添加语言

注意



无外观支持

为数据池项定义语言支持时，您无法为同一项目添加外观支持。



添加语言

列表中的第一个语言始终是默认语言，并且无法删除。如果您添加了某个语言，那么该语言将使用标准语言设置作为初始值。

步骤 1

单击 。

将打开项目中心。

步骤 2

在导航区中，单击“配置”>“语言”。

将显示可用语言。

步骤 3

在内容区域中，单击“添加”。

语言已添加至表。

步骤 4

按 F2 并输入该语言的名称。

步骤 5

从“语言”下拉列表框中选择语言。

步骤 6

从“国家/地区”下拉列表框中选择国家/地区。

您已添加语言。

有关如何在运行时刻更改语言的说明，请参见[节11.6](#)。

## 8.4.2. 删除语言




### 删除语言

先决条件:

- 至少向 EB GUIDE 模型中添加两种语言。

#### 步骤 1

点击 。

将打开项目中心。

#### 步骤 2

在导航区中，点击“配置”>“语言”。

将显示可用语言。

#### 步骤 3

在内容区域中，选择语言。

#### 步骤 4

在内容区域中，点击“删除”。

将从表中删除该语言。

## 8.5. 使用语言支持

通过外观支持，您可以为模型定义不同的数据池值。这样，您可以为同一型号定义不同的外观，例如夜间和日间模式。

### 8.5.1. 向 EB GUIDE 模型中添加外观

注意




无可用语言支持

为数据池项目定义外观支持时，您无法向同一项目添加语言支持。



向 EB GUIDE 模型中添加外观

#### 步骤 1

点击 。

将打开项目中心。

#### 步骤 2

在导航区中，点击“配置”>“外观”。

默认情况下，每个模型都添加了一个标准外观。

#### 步骤 3

在内容区域中，点击“添加”。

外观已添加至表。

#### 步骤 4

按 F2 并重命名外观。

新外观已添加至模型。在项目编辑器中，您可以在命令区域的下拉列表框中选择新的皮肤。

### 8.5.2. 向数据池项添加外观支持



向数据池项添加外观支持

要定义不同的数据池值，从而为您的 EB GUIDE 模型定义各种外观，您首先需要为数据池项添加外观支持。

先决条件：


- EB GUIDE 模型包含数据池项。
- 外观已添加至模型。

#### 步骤 1

在项目编辑器中，请转至“数据池”元素。




步骤 2

在数据池项的“值”属性旁，请点击  按钮。

菜单将展开。

步骤 3

在菜单中，点击“添加外观支持”。

对话框将关闭。在“值”属性旁，显示  按钮。它表示在此数据池项中添加了外观支持，现在可为每个外观定义不同值。

步骤 4

要为数据池项定义不同的值，请在“数据池”元素中选择数据池。

“属性”元素显示 EB GUIDE 模型中可用的所有外观的表。

步骤 5

为表中的每个外观定义一个值。

### 8.5.3. 外观间转换



#### 外观间转换

先决条件：

- EB GUIDE 模型包含数据池项。
- 外观已添加至模型。

步骤 1

在项目编辑器中，转至命令区。

步骤 2

从下拉列表框中选择渲染器。

内容区域显示具有对此外观有效的数据池值的模型。此外，模拟模式将显示具有特定外观值的模型。

### 8.5.4. 删除外观




#### 删除外观

先决条件：

- 外观已添加至模型。

#### 步骤 1

点击 。

将打开项目中心。

#### 步骤 2

在导航区中，点击“配置”>“外观”。

列出了当前项目的**所有外观**。

#### 步骤 3

选择要删除的外观，并点击“删除”。

从表中删除该外观。

## 8.6. 给视图转变制作动画

### 8.6.1. 添加入口动画



添加入口动画

要是视图以移动动画或消失动画显示，你可以向视图模板添加入口动画。

先决条件：

- 已添加视图模板。

#### 步骤 1

在“导航”元素中，点击视图模板。

#### 步骤 2

转至“属性”元素。

#### 步骤 3

要定义进入视图时播放的动画，选择“入口动画”复选框。

#### 步骤 4

从“转化类型”下拉列表框中，为视图转化选择类型。

#### 步骤 5

在“持续时间”文本框中，输入持续时间（毫秒）。

#### 步骤 6

选择“退出动画后播放”复选框。

结果：取自该视图模板的每一个视图都将以您定义的动画进入。使用“退出动画后播放”复选框，您定义了播放完上一个视图的退出动画后才能播放入口动画。

## 8.6.2. 添加退出动画



### 添加退出动画

要是视图以移动动画或消失动画消失，你可以向视图模板添加退出动画。

先决条件：

- 已添加视图模板。

#### 步骤 1

在“导航”元素中，点击视图模板。

#### 步骤 2

转至“属性”元素。

#### 步骤 3

要定义退出视图时播放的退出动画，请选择“退出动画”复选框。

#### 步骤 4

从“转化类型”下拉列表框中，为视图转化选择类型。

#### 步骤 5

在“持续时间”文本框中，输入持续时间（毫秒）。

#### 步骤 6

在“延迟”文本框中，输入延迟（毫秒）。

结果：取自该视图模板的每一个视图都将以您定义的动画退出。

## 8.7. 重复使用控件

### 8.7.1. 添加模板



### 添加模板

#### 步骤 1

在“导航”元素中，转至“模板”，然后点击 **+**。



菜单将展开。

步骤 2

在菜单中，点击模板的类型。

已添加所选类型的新模板。内容区域将显示该模板。

步骤 3

重命名该模板。

步骤 4

在“属性”元素中，编辑该模板的属性并定义模板接口。

提示



模板的模板

模板的类型可以是现有模板。EB GUIDE 因此允许根据模板来创建模板。

提示



复制和查找模板

或者，您可以使用上下文菜单或 **Ctrl+C** 及 **Ctrl+V** 来复制并粘贴现有模板。

要在 EB GUIDE 模型中查找特定模板，请在搜索框中输入模板名称或使用 **Ctrl+F**。要跳过模板，请在命中列表中双击它。

## 8.7.2. 定义模板接口



### 定义模板接口


先决条件：


- EB GUIDE 模型包含一个模板。

步骤 1

选择模板。


步骤 2

要向模板接口添加属性，请在“属性”元素中点击属性旁的  按钮。在菜单中，点击“添加至模板接口”。

 图标将显示在属性旁边。

步骤 3

要从模板接口中删除属性，请点击属性旁的  按钮。在菜单中，点击“从模板接口中删除”。

 图标不再显示在属性旁边。

注意



实例化器模板

对于实例化器的模板，无法将实例化器的子代控件的属性添加到模板接口中。

### 8.7.3. 使用模板



使用模板

先决条件：

- 内容区域将显示视图。
- “工具箱”中有一个可用控件模板。
- 控件模板中的模板接口中至少有一个属性。

步骤 1

将控件模板从“工具箱”拖入视图中。

模板的实例将添加至视图。“属性”元素显示属于模板接口的属性。

提示





定义模板接口


如果“属性”元素未显示模板实例的任何属性，则表示尚未向模板接口添加任何属性。定义模板接口以更改此情况。

步骤 2

在“属性”元素中，编辑模板实例的属性。

编辑属性后， 按钮更改为  按钮。

步骤 3

要将属性值重置为模板的值，请点击属性旁的  按钮。在菜单中，点击“重置为模板值”。

### 8.7.4. 删除模板



### 删除模板

#### 步骤 1

在“导航”元素中，右键点击模板。

#### 步骤 2

在上下文菜单中，点击“删除”。

模板将被删除。

## 9. 处理数据

注意



默认窗口布局

本用户手册的所有说明和屏幕截图，都使用默认窗口布局。如果您想按照说明进行操作，建议您将 EB GUIDE Studio 窗口设置为默认 布局，方法是选择“布局” > “重置为默认布局”。

### 9.1. 添加事件



添加事件

步骤 1

在“事件”元素中，点击 **+**。

事件已添加至表中。

步骤 2

对事件进行重命名。

提示



复制和查找事件

或者，您可以使用上下文菜单或 **Ctrl+C** 及 **Ctrl+V** 来复制并粘贴现有事件。为了避免出现重复，所粘贴的事件的事件 ID 不同于所复制的事件。

要在 EB GUIDE 模型中查找特定事件，请在搜索框中输入事件的名称或使用 **Ctrl+F**。要跳过事件，请在命中列表中双击它。

### 9.2. 向事件添加参数



向事件添加参数

先决条件：

- 添加事件至 EB GUIDE 模型。

#### 步骤 1

在“事件”元素中，点击某个事件。

#### 步骤 2

在事件表中，请点击事件旁的 **+**。

#### 步骤 3

从下拉列表框中，为参数选择类型。

已向事件添加所选类型的参数。

#### 步骤 4

对参数进行重命名。


## 9.3. 确定事件位置

事件 ID 和事件组 ID 用于确定事件位置。EB GUIDE TF 在运行时，使用 ID 发送和接收事件。



### 添加事件组

#### 步骤 1

点击 。

将打开项目中心。

#### 步骤 2

在导航区中，点击“配置”>“事件组”。

#### 步骤 3

在内容区域中，点击“添加”。

事件组已添加至表。

#### 步骤 4

对事件组进行重命名。

#### 步骤 5

要更改事件组 ID，双击该“ID”，然后输入数字。



### 确定……的事件位置 EB GUIDE TF

先决条件：

- 已添加事件组。
- 添加事件至 EB GUIDE 模型。

#### 步骤 1

在“事件”元素中，点击某个事件。

“属性”元素显示所选事件的属性。

#### 步骤 2

在 Event ID 文本框中插入 ID。

#### 步骤 3

转至“事件”元素，从 Group 的下拉列表框中选择一个事件组。

## 9.4. 删除事件



### 删除事件

先决条件：

- 添加事件至 EB GUIDE 模型。

#### 步骤 1

在“事件”元素中，右键点击事件。

#### 步骤 2

在上下文菜单中，点击“删除”。

事件将被删除。

## 9.5. 添加数据池项



### 添加数据池项

#### 步骤 1

在“数据池”元素中，点击 **+**。

菜单将展开。

#### 步骤 2

在菜单中，点击数据池项的类型。

已添加所选类型的新数据池项。数据池项已准备就绪，可供内部使用。

**步骤 3**

重命名该数据池项。

**提示**

复制和查找数据池项

或者，您可以使用上下文菜单或 **Ctrl+C** 及 **Ctrl+V** 复制并粘贴现有数据池项。

要在 **EB GUIDE** 模型中查找特定数据池项，请在搜索框中输入数据池项名称或使用 **Ctrl+F**。要跳过数据池项，请在命中列表中双击它。

## 9.6. 编辑列表类型的数据池项



### 编辑列表类型的数据池项


先决条件：

- 已添加列表类型的数据池项。

**步骤 1**

在“数据池”元素中，点击列表类型的数据池项。

**步骤 2**

在 Value 列，点击 .

将打开编辑器。

**步骤 3**

要向列表数据池项中添加项，请点击“添加”。

新条目已添加至表。

**步骤 4**

在 Value 文本框中输入新条目的值，或从组合框中选择值。

**步骤 5**

重复第三步和第四步，向列表中添加更多项。

**步骤 6**

点击“接受”。

列表内容将显示在 Value 列旁边。

## 9.7. 将属性转换为脚本化的值




### 将属性转换为脚本化的值

数据池项和控件的属性可以转换为脚本化的值，也可以转换回纯值。下列说明展示了数据池项值的转换过程。对于控件属性，该过程是相同的。

先决条件：

- 已添加数据池项。
- 数据池项不依赖于语言。
- 数据池项未链接。

#### 步骤 1

在“数据池”元素中，点击数据池项，并点击  按钮。

菜单将展开。

#### 步骤 2

在菜单中，点击“转换为脚本”。

数据池项已转换为脚本化的值。

#### 步骤 3


在 Value 列中，点击“编辑”。

将在内容区域打开脚本编辑器。

#### 步骤 4

编辑 EB GUIDE 脚本。

#### 步骤 5

要将数据池项转换回纯值，请点击  按钮。

菜单将展开。

#### 步骤 6

在菜单中，点击“转换为纯值”。

数据池项已转换为纯值。

## 9.8. 建立外部通信

要建立外部通信（例如，在 EB GUIDE 模型和应用程序之间建立通信），您需要将通信环境添加至 EB GUIDE 模型。






## 添加通信环境

通过使用通信环境，您能够进行通信。

### 步骤 1

点击 。

将打开项目中心。

### 步骤 2

在导航区中，点击“配置”>“通信环境”。

### 步骤 3

在内容区域中，点击“添加”。

通信环境已添加至表。

### 步骤 4

重命名通信环境，例如将其重命名为 Media。

### 步骤 5

要在自有线程中运行通信环境，请选择“使用自有线程”。

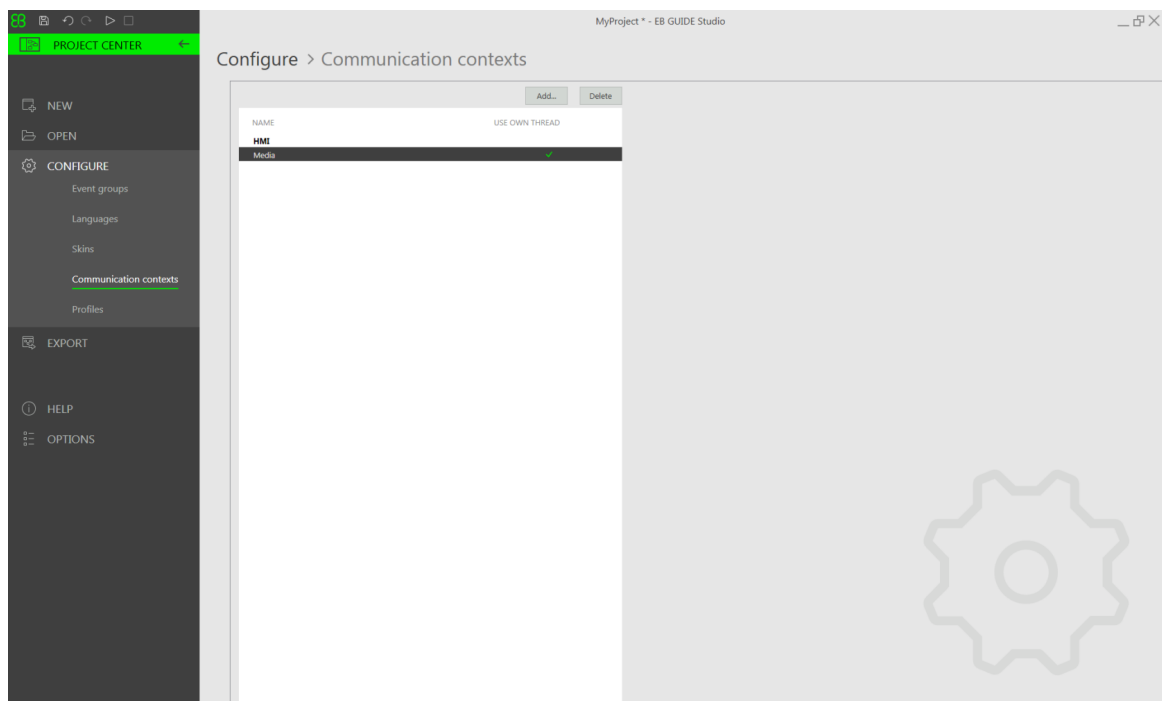


图 9.1. 通信环境 Media。

## 9.9. 数据池项之间的链接



### 数据池项之间的链接

先决条件：

- 已添加数据池项。
- 数据池项不依赖于语言。
- 数据池项不是脚本化的值。

#### 步骤 1

在“数据池”元素中，点击数据池项。

#### 步骤 2

点击  按钮。

菜单将展开。

#### 步骤 3

在菜单中，点击“向数据池项中添加链接”。

对话框将打开。

#### 步骤 4

要添加新的数据池项，请在文本框中输入名称。

#### 步骤 5

点击“添加数据池项”。

#### 步骤 6

点击“接受”。

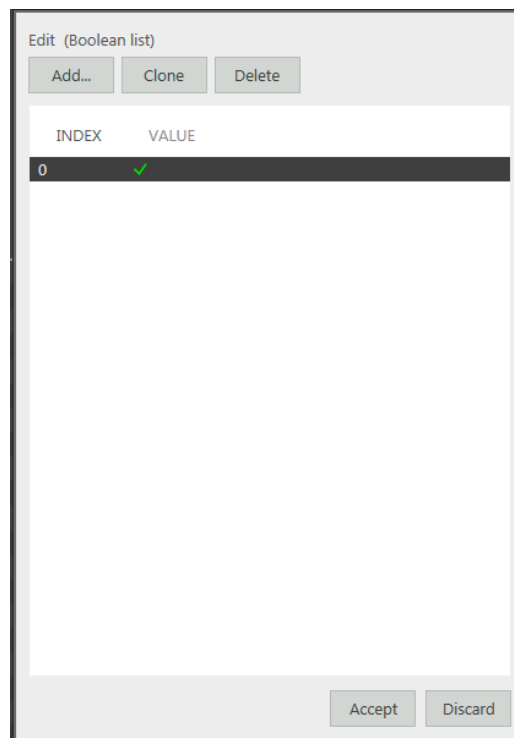



图 9.2. 数据池项之间的链接

对话框将关闭。在 Value 属性旁边，显示  按钮。此按钮表示 Value 属性已链接到数据池项。无论在何时某一数据池项改变，其他数据池项的值也会改变。

## 9.10. 删除数据池项



### 删除数据池项

先决条件：

- 已添加数据池项。

#### 步骤 1

在“数据池”元素中，右键点击数据池项。

#### 步骤 2

在上下文菜单中，点击“删除”。

数据池项将被删除。

## 10. 处理项目

注意



默认窗口布局


本用户手册的所有说明和屏幕截图，都使用默认窗口布局。如果您想按照说明进行操作，建议您将 **EB GUIDE Studio** 窗口设置为默认 布局，方法是选择“布局” > “重置为默认布局”。

### 10.1. 创建项目



创建项目

步骤 1

点击 。

将打开项目中心。

步骤 2

在导航区中，点击“新建”。

步骤 3

输入项目名称并选择一个位置。

步骤 4

点击“创建”。

将创建项目。项目编辑器将打开并显示新项目。

## 10.2. 打开项目

### 10.2.1. 从文件资源管理器中打开项目



#### 从文件资源管理器中打开项目

先决条件:

- 将创建 EB GUIDE Studio 项目。

#### 步骤 1

打开文件资源管理器，并选择您要打开的 EB GUIDE Studio 项目文件。EB GUIDE Studio 项目文件的扩展名为 .ebguide。

#### 步骤 2

双击 EB GUIDE Studio 项目文件。

项目将在 EB GUIDE Studio 中打开。

### 10.2.2. 在……中打开项目 EB GUIDE Studio



#### 在……中打开项目 EB GUIDE Studio

先决条件:

- 将创建 EB GUIDE Studio 项目。

#### 步骤 1

点击 。

将打开项目中心。

#### 步骤 2

在导航区中，点击“打开”标签。

#### 步骤 3

选择“最近项目”下面列出的项目，或点击“浏览”并选择您要打开的 EB GUIDE Studio 项目文件。EB GUIDE Studio 项目文件的扩展名为 .ebguide。

项目将在 EB GUIDE Studio 中打开。

## 10.3. 重命名模型元素



### 重命名模型元素

以下说明将指导您完成重命名模型元素（如状态、状态机、控件、转化、数据池项和事件）的过程。

先决条件：

- 模型元素将添加至 EB GUIDE 模型。

#### 步骤 1

要重命名模型元素，请执行下列操作：

- ▶ 要重命名模型元素（如控件、状态、状态机或转化），请在“导航”元素中，右键点击模型元素。
- ▶ 要重命名数据池项，请在“数据池”元素中，右键点击数据池项。
- ▶ 要重命名事件，请在“事件”元素中，右键点击数据池项。

上下文菜单打开。

#### 步骤 2

在上下文菜单中，点击以下任意一项：

- ▶ 要仅重命名所选模型元素，请选择“重命名”。
- ▶ 例如，要重命名所选模型元素及其在 EB GUIDE 模型中的条目，请在 EB GUIDE 脚本 中选择“重命名全局”。

## 10.4. 验证和模拟运行 EB GUIDE 模型

将 EB GUIDE 模型导出至目标设备之前，您需要解决错误并在 PC 上模拟运行此模型。

### 10.4.1. 验证 EB GUIDE 模型

#### 10.4.1.1. 使用……验证 EB GUIDE 模型 EB GUIDE Studio



### 使用……验证 EB GUIDE 模型 EB GUIDE Studio


在“问题”元素中，EB GUIDE 显示了以下内容：



▶  错误

▶  警告

步骤 1

在“问题”元素中，点击 。

将显示错误和警告数。

步骤 2

点击“问题”以展开“问题”元素。

将显示错误和警告列表。

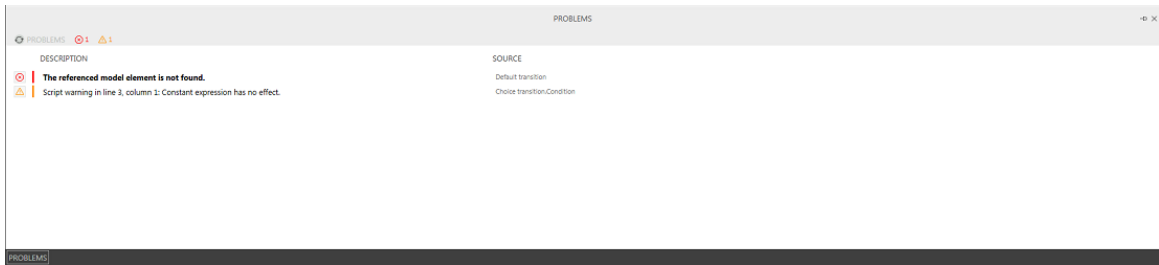


图 10.1. 问题元素

步骤 3


要浏览至问题来源，请双击对应的行。

将突出显示导致问题的元素。

步骤 4

解决该问题。

步骤 5

点击 .

您解决的问题不再列示在“问题”元素中。

步骤 6

要折叠“问题”元素，请再次点击“问题”。

如果没有错误，EB GUIDE 模型是有效的。如果出现警告，EB GUIDE 模型也是有效的。

### 10.4.1.2. 使用命令行验证 EB GUIDE 模型



#### 使用命令行验证 EB GUIDE 模型

步骤 1

使用命令行浏览至 \$GUIDE\_INSTALL\_PATH/Studio。

### 步骤 2

输入 `Studio.Console.exe -c "<logfile dir>/log.txt" -o "$GUIDE_PROJECT_PATH/project_name.-ebguide"`。


验证 EB GUIDE 模型后，结果保存至指定位置的配置文件中 <logfile dir>。

## 10.4.2. 启动和停止模拟运行



### 启动和停止模拟运行


#### 步骤 1

要启动模拟运行，请点击命令区的 。

模拟运行和 EB GUIDE Monitor 启动。模拟运行将使用它自己的配置启动。

要更改配置，请转至项目中心并点击“配置”>“配置文件”。

#### 步骤 2

要启动模拟运行，请点击命令区的 。

模拟运行和 EB GUIDE Monitor 停止。

## 10.5. 导出 EB GUIDE 模型

### 10.5.1. 使用以下工具导出 EB GUIDE 模型 EB GUIDE Studio




### 使用以下工具导出 EB GUIDE 模型 EB GUIDE Studio

要将 EB GUIDE 模型复制到目标设备，您需要用 EB GUIDE Studio 将其导出。

在每次导出 EB GUIDE 模型时，您都需要选择配置文件。

#### 步骤 1

点击 。

将打开项目中心。



#### 步骤 2

在导航区中，点击“导出”标签。

#### 步骤 3

从 Profile 下拉列表框中选择配置文件。

#### 步骤 4

点击“浏览”并选择将向其中导出二进制文件的位置。

#### 步骤 5

点击“选择文件夹”。

#### 步骤 6

点击“导出”。

二进制文件将导出至所选位置。

## 10.5.2. 使用命令行导出 EB GUIDE 模型



### 使用命令行导出 EB GUIDE 模型

先决条件：

- EB GUIDE 模型没有错误和警告。

#### 步骤 1

使用命令行浏览至 `$GUIDE_INSTALL_PATH/Studio`。

#### 步骤 2

输入 `Studio.Console.exe -e <destination dir> -p <profile> -o "$GUIDE_PROJECT_PATH/project_name.-ebguide"`。


EB GUIDE 模型和配置文件 `<profile>` 一起导出至所选位置 `<destination dir>`。

## 10.6. 更改 EB GUIDE Studio 的显示语言



### 更改 EB GUIDE Studio 的显示语言

#### 步骤 1

点击 。

将打开项目中心。

#### 步骤 2

在导航区中，点击“选项”标签。

#### 步骤 3

从“显示语言”下拉列表框中选择语言。

#### 步骤 4

重启 EB GUIDE Studio。

重启后，将以选定的语言显示图形用户界面。

## 10.7. 配置文件

EB GUIDE Studio 可以为 EB GUIDE 模型创建不同的配置文件。

您可以使用配置文件进行以下操作：

- ▶ 发送消息
- ▶ 配置内部和用户定义的库以加载
- ▶ 配置场景
- ▶ 配置渲染器

有两种默认配置文件：“编辑”和“模拟”。

### 10.7.1. 添加配置文件



#### 添加配置文件

要在 EB GUIDE Studio 中添加配置文件，请克隆现有的配置文件。

先决条件：

- 将打开 EB GUIDE Studio 项目。
- 将显示项目中心。

#### 步骤 1

在导航区，单击“配置” > “配置文件”。

#### 步骤 2

在内容区域，选中“模拟”配置文件。

#### 步骤 3

单击“克隆”。

配置文件已添加至表。该配置文件是默认配置文件“模拟”的克隆。

#### 步骤 4

在表中双击，然后将配置文件重命名为 MySimulation。

#### 步骤 5

选择“用于模型运行”。

MySimulation 配置文件用于在 PC 上模拟运行。

## 10.7.2. 添加库

EB GUIDE TF 的默认产品可运行于支持共享库的操作系统，例如 Windows 10、Linux 或 QNX。EB GUIDE TF 分为可执行文件和一组库，以适应大多数非常规的客户项目。

以下任务介绍了如何添加能与 EB GUIDE 模型交互并提供其他功能的用户定义库。



### 添加库：平台

此任务将显示如何添加可在当前平台上由所有 EB GUIDE 模型使用的库或多个库。

先决条件：

- 将打开 EB GUIDE Studio 项目。
- 将显示项目中心。
- 在导航区中，已选择“配置” > “配置文件”标签。
- 已添加配置文件 MySimulation。
- 在 \$GTF\_INSTALL\_PATH/platform/<platform name> 中，库 MyLibraryA 和 MyLibraryB 可用。

#### 步骤 1

在内容区域中，选中 MySimulation 配置文件。

#### 步骤 2

点击“平台”标签。

#### 步骤 3

输入以下代码：

```
{
  "gtf":
  {
    "core":
    {
      "pluginstoload": ["MyLibraryA", "MyLibraryB"]
    }
  }
}
```

```
}  
}  
}
```

添加库 MyLibraryA 和 MyLibraryB 到启动代码。

#### 注意



#### JSON 对象标记

如果您在 EB GUIDE Studio 中配置 platform.json，请使用 JSON 对象标记。

例如，请参见[节12.7.1](#)。

有关 JSON 格式的详细信息，请参见 <http://www.json.org>。



#### 添加库：模型

此任务将显示如何添加只能由当前 EB GUIDE 模型使用的一个或多个库。

先决条件：

- 将打开 EB GUIDE Studio 项目。
- 将显示项目中心。
- 在导航区中，已选择“配置” > “配置文件”标签。
- 已添加配置文件 MySimulation。
- 在 \$GUIDE\_PROJECT\_PATH/<project name>/resources 中，库 MyLibraryA 和 MyLibraryB 可用。

#### 步骤 1

在内容区域中，选中 MySimulation 配置文件。

#### 步骤 2

点击“模型”标签。

#### 步骤 3

输入以下代码：

```
{  
  "gtf":  
  {  
    "model":  
    {  
      "pluginstoload": ["resources/MyLibraryA", "resources/MyLibraryB"]  
    }  
  }  
}
```

添加库 MyLibraryA 和 MyLibraryB 到启动代码。

## 注意

**JSON 对象标记**

如果您在 EB GUIDE Studio 中配置 model.json，请使用 JSON 对象标记。

例如，请参见[节12.6.1](#)。

有关 JSON 格式的详细信息，请参见<http://www.json.org>。

### 10.7.3. 配置场景

在 EB GUIDE Studio 中，可为每个状态机配置场景。

由于下列原因之一，项目可拥有多个状态机：

- ▶ 将模型的逻辑分为不同的状态机
- ▶ 使用多个显示或图层



#### 配置场景

先决条件：

- 将打开 EB GUIDE Studio 项目。
- 将显示项目中心。
- 在导航区中，已选择“配置”>“配置文件”标签。

步骤 1

在内容区域中，点击“场景”标签。

步骤 2

从“状态机”下拉列表框中，选择主要显示的状态机，例如“主要”。

步骤 3

要设定窗口在台式机桌面上的初始位置，请为 x 和 y 输入值。

步骤 4

从“渲染器”下拉列表框中，选择渲染器。

步骤 5

调整更多属性。有关每个属性的信息，请参见[节12.8](#)。

## 10.8. 导出和导入语言依赖项文本

## 10.8.1. 导出语言依赖项文本

提示



验证 **EB GUIDE** 模型

要避免在导出和导入文本期间出现错误，请在开始前验证 **EB GUIDE** 模型。




### 导出语言依赖项文本

要以用户首选语言提供文本，您可以导出数据池项的所有语言依赖项文本并将这些文本传递给翻译人员。

先决条件：

- 已添加 String 或 String list 类型的数据池项。
- 该数据池项提供了语言支持。有关如何添加语言依赖项文本的信息，请参见 [节11.6](#)。
- 至少向 **EB GUIDE** 模型中添加两种语言。
- **EB GUIDE** 模型没有错误和警告。

#### 步骤 1

点击 。

将打开项目中心。

#### 步骤 2

在导航区中，点击“配置”>“语言”。

#### 步骤 3

在内容区域中，选择需要翻译的目标语言。

可以进行多项选择。

#### 步骤 4

点击“导出”。

对话框将打开。

#### 步骤 5

选择要向其中导出文件的目录。

#### 步骤 6

点击“选择文件夹”。

结果：导出开始。文件将保存在所选目录中。该文件具有语言依赖项首字母缩写词，并且格式为 .xliff。该文件包含源语言的值以及目标语言的值。

注意



对于每种语言将导出一个文件

对于您在项目中心中选择的每个语言，将导出单独的文件。

## 10.8.2. 导入语言依赖项文本

### 10.8.2.1. 使用以下工具导入语言依赖项文本 **EB GUIDE Studio**




使用以下工具导入语言依赖项文本 **EB GUIDE Studio**

先决条件：

- 已添加 String 或 String list 类型的数据池项。
- 该数据池项提供了语言支持。有关如何添加语言依赖项文本的信息，请参见 [节11.6](#)。
- 至少向 **EB GUIDE** 模型中添加两种语言。
- **EB GUIDE** 模型没有错误和警告。
- 至少有一个已翻译的 .xliff 文件可用。

#### 步骤 1

点击 。

将打开项目中心。

#### 步骤 2

在导航区中，点击“配置”>“语言”。

#### 步骤 3

点击“导入”。

对话框将打开。

#### 步骤 4

选择存储已翻译的 .xliff 文件的目录。

#### 步骤 5

选择已翻译的 .xliff 文件。

可以进行多项选择。

#### 步骤 6

点击“打开”。

导入开始。对话框将打开。

#### 步骤 7

点击“关闭”。

### 10.8.2.2. 使用命令行导入语言依赖项文本



#### 使用命令行导入语言依赖项文本

先决条件：

- 至少向 EB GUIDE 模型中添加两种语言。
- EB GUIDE 模型没有错误和警告。
- 提供经过翻译的 .xliff 语言文件。

#### 步骤 1

使用命令行浏览至 \$GUIDE\_INSTALL\_PATH/Studio。

#### 步骤 2

输入 `Studio.Console.exe -l <language file> -o "$GUIDE_PROJECT_PATH/project_name.ebguide"`。

如果导入成功，EB GUIDE 模型被保存。如果导入失败，EB GUIDE 模型不保存。两种情况下，都会生成日志文件。日期和时间戳会添加到日志文件名中。

## 10.9. 使用 EB GUIDE Monitor

### 10.9.1. 在 EB GUIDE Monitor 触发事件



#### 触发事件位于 EB GUIDE Monitor

先决条件：

- EB GUIDE 模型包含事件。



- 启动 EB GUIDE 模型的模拟运行。
- 启动了 EB GUIDE Monitor。

#### 步骤 1


在 EB GUIDE Monitor 中的“事件”元素中，搜索要在“搜索事件”搜索框中触发的事件。

#### 步骤 2

点击事件。

事件将添加至列表。


#### 步骤 3

要触发事件，请在事件旁的“事件”元素中点击 。

事件将触发。日志消息显示在“记录器”元素中。

#### 步骤 4


##### 步骤 4.1

如果事件有参数，请点击  展开参数。

##### 步骤 4.2

在“值”列中更改参数。

##### 步骤 4.3

要触发事件，请点击事件旁的 。

用更改的参数触发事件。日志消息显示在“记录器”元素中。

## 10.9.2. 使用 EB GUIDE Monitor 更改数据池项的值



### 在 EB GUIDE Monitor 中更改数据池项的值

先决条件：

- EB GUIDE 模型包含数据池项。
- 启动 EB GUIDE 模型的模拟运行。
- 启动了 EB GUIDE Monitor。

#### 步骤 1

在 EB GUIDE Monitor 中的“数据池”元素中，搜索“搜索数据池项”搜索框的数据池项。

#### 步骤 2

点击数据池项。

数据池项将添加至列表。

步骤 3

在“值”列中更改数据池项的值。

注意



支持的类型

可以更改以下数据类型的数据池项：

- ▶ 布尔值
- ▶ 颜色
- ▶ 整数
- ▶ 浮点值
- ▶ 字符串

数据池项的值将更改。日志消息显示在“记录器”元素中。

### 10.9.3. 在 EB GUIDE Monitor 中启动脚本



在 EB GUIDE Monitor 中启动脚本

先决条件：

- 启动 EB GUIDE 模型的模拟运行。
- 启动了 EB GUIDE Monitor。
- 带脚本的 .cs 文件或者 .dll 文件，在您的计算机上可用。有关脚本示例，请参见[节10.9.3.1](#)。

步骤 1

要打开“脚本”元素，请选择“布局”>“脚本”。

“脚本”元素以对接元素形式打开。

步骤 2

在“脚本”元素中，请点击“打开”按钮。

打开文件浏览器。

步骤 3

选择一个 .cs 文件或 .dll 文件，并点击“打开”。

文件中包含的所有适用方法和相应的类别，都列在了“脚本”表中。

步骤 4

选择一种方法，然后点击开始按钮。

该脚本已启动。在“脚本输出”区域中，将显示一条日志消息。

### 10.9.3.1. 编写脚本文件以作为 EB GUIDE Monitor

有关脚本方法的详细信息，请参见 `$GUIDE_INSTALL_PATH/doc/monitor/monitor_api.chm` 中的 EB GUIDE Monitor API。

下面是基本 EB GUIDE Monitor 脚本函数的示例。

注意



使用状态和状态机的方法

如果您的 EB GUIDE 模型具有多个同名的状态或状态机，请使用 uint ID。在 `$EXPORT_PATH/monitor.cfg` 中找到与您的项目相关的 uint ID。



例 10.1.

示例脚本文件关于 EB GUIDE Monitor

下面是示例脚本 `MonitorScriptSample.cs`。

```
namespace MyProject
{
    using System.Threading.Tasks;

    using System.Windows.Media; // necessary for Color type!

    using Elektrobit.Guide.Monitor.Scripting.MonitorContext;

    public class Basic
    {
        public async Task PrintMessage(IMonitorContext monitor) //❶
        {
            await monitor.Write("Hello World");
        }

        public async Task FireEvent(IMonitorContext monitor) //❷
        {
            await monitor.FireEvent("nextView");
        }
    }

    public class Events
    {
        public async Task FireEventWithParameter(IMonitorContext monitor)
        {
            await monitor.FireEvent("setBool", true);
        }

        public async Task WaitForEvent(IMonitorContext monitor) //❸
```

```

    {
        var ev = await monitor.WaitForEvent("nextView");
        await monitor.Write("Even occurred: " + ev.EventModel.Name);
    }

    public async Task WaitForEventWithParameters(IMonitorContext monitor)
    {
        var ev = await monitor.WaitForEvent("setBool");

        bool mv1 = ev["value"]; // read parameter via name
        bool mv2 = ev[0]; // read the parameter via index

        await monitor.Write("Parameter 'value' is: " + mv1);
        await monitor.Write("Parameter [0] is: " + mv2);
    }
}

public class Datapool
{
    public async Task WriteDpValue(IMonitorContext monitor) //❹
    {
        await monitor.WriteDatapool("Boolean 1", true);
    }

    public async Task ReadDatapoolValue(IMonitorContext monitor) //❺
    {
        bool boolValue = await monitor.ReadDatapool("Boolean 1");
        string stringValue = await monitor.ReadDatapool("String 1");
        int integerValue = await monitor.ReadDatapool("Integer 1");
        float floatValue = await monitor.ReadDatapool("Float 1");

        await monitor.Write("Boolean: " + boolValue);
        await monitor.Write("String: " + stringValue);
        await monitor.Write("Integer: " + integerValue);
        await monitor.Write("Float: " + floatValue);
    }

    public async Task ReadColor(IMonitorContext monitor)
    {
        Color colorValue = await monitor.ReadDatapool("Color 1");
        await monitor.Write("Boolean: " + colorValue);
    }
}

public class StateMachines
{
    public async Task WaitForStateChanges(IMonitorContext monitor)

```



```

    {
        var leftState = await monitor.WaitForStateExit
            ("Main", "State 1"); //❸
        await monitor.Write(string.Format("State {0} left",
            leftState.Name));

        var enteredState = await monitor.WaitForStateEnter
            ("Main", "State 2"); //❹
        await monitor.Write(string.Format("State {0} entered",
            enteredState.Name));
    }

    public async Task WaitForStateMachineChanges(IMonitorContext monitor)
    {
        var startedStateMachine = await monitor.WaitForStateMachineStart
            ("Dynamic state machine 1"); //❺
        await monitor.Write(string.Format("State Machine {0} started",
            startedStateMachine.Name));

        var stoppedStateMachine = await monitor.WaitForStateMachineStop
            ("Dynamic state machine 1"); //❻
        await monitor.Write(string.Format("State Machine {0} stopped",
            stoppedStateMachine.Name));
    }
}

public class Advanced
{
    public async Task CaptureScreenshot(IMonitorContext monitor) //❿
    {
        // make sure remote framebuffer is enabled in profile
        uint sceneId = 0;
        await monitor.CaptureScreenshot(sceneId, @"d:/image.png");
    }

    public async Task CountTo10(IMonitorContext monitor)
    {
        for (var i = 0; i < 10; i++)
        {
            await monitor.Write("Hello World: " + i);
            await Task.Delay(1000, monitor.CancellationToken);

            monitor.CancellationToken.ThrowIfCancellationRequested();
        }
    }

    public async Task WaitForEventWithTimeout(IMonitorContext monitor) //⓫

```

```

{
    // Disclaimer:
    // this is just one of many opportunities provided by
    // the .NET's "Task Parallel Library"

    var eventWaitTask = monitor.WaitForEvent("nextView");

    await Task.WhenAny(eventWaitTask, Task.Delay(5000));

    if (!eventWaitTask.IsCompleted || eventWaitTask.IsFaulted)
    {
        return;
    }

    await monitor.Write("event occurred");
}
}
}

```

- ❶ 打印输出消息的方法
- ❷ 触发事件的方法
- ❸ 等待事件的方法
- ❹ 写入数据池值的方法
- ❺ 读取数据池值的方法
- ❻ 等待直到进入该状态并进行报告的方法
- ❼ 等待直到退出该状态并进行报告的方法
- ❽ 等待直到该状态机启动并进行报告的方法
- ❾ 等待直到该状态机停止并进行报告的方法
- ❿ 捕获屏幕截图的方法
- ⓫ 等待存在超时的事件的方法

### 10.9.4. 作为独立程序启动 EB GUIDE Monitor

EB GUIDE Monitor 在模拟运行 EB GUIDE 模型过程中，在 EB GUIDE Studio 中自动启动。但是，您还可以在 \$GUIDE\_INSTALL\_PATH/tools/monitor 中或使用命令行启动 EB GUIDE Monitor 作为独立应用程序。



#### 用命令行启动 EB GUIDE Monitor

先决条件：

- EB GUIDE 已安装。

- EB GUIDE 模型将导出至 \$EXPORT\_PATH。

#### 步骤 1

在文件浏览器中，请导航至 \$GUIDE\_INSTALL\_PATH/tools/monitor。

#### 步骤 2

打开命令行并输入以下内容： `Monitor.exe -c <ip adress>:<port> -o <$EXPORT_PATH/monitor.cfg>`

### EB GUIDE Monitor 启动

#### 提示



重复使用预配置数据池项和事件

在 **EB GUIDE Monitor** 中，您可以配置数据池项和事件。已配置的值存储在 `C:/Users/<username>/AppData/Local/Temp/eb_guide_simulation_export/<guide_project>/monitor_settings.xml` 中。要重复使用预配置值，请将 `monitor_settings.xml` 复制到 \$EXPORT\_PATH。



### 更改 EB GUIDE Monitor 的显示语言

先决条件：

- EB GUIDE Monitor 作为独立应用程序启动。

#### 步骤 1

从“文件”>“显示语言”菜单中，选择语言。

#### 步骤 2

重启 EB GUIDE Monitor。

重启后，图形用户界面会以所选语言显示。

请注意，如果 EB GUIDE Monitor 在 EB GUIDE Studio 中启动，您无法更改图形用户界面的显示语言。这意味着 EB GUIDE Monitor 的显示语言与 EB GUIDE Studio 相同。



# 11. 教程

注意



默认窗口布局

本用户手册的所有说明和屏幕截图，都使用默认窗口布局。如果您想按照说明进行操作，建议您将 EB GUIDE Studio 窗口设置为默认 布局，方法是选择“布局” > “重置为默认布局”。

## 11.1. 教程：添加动态状态机

注意



默认窗口布局

本用户手册的所有说明和屏幕截图，都使用默认窗口布局。如果您想按照说明进行操作，建议您将 EB GUIDE Studio 窗口设置为默认 布局，方法是选择“布局” > “重置为默认布局”。

动态状态机允许在运行时间弹出。例如，您可以使用动态状态机显示覆盖常规显示的错误消息。

下列说明将指导您完成创建动态状态机的过程。这些说明将向您展示如何为用于音量控制的动态状态机建模。为获得最佳效果，请按提供的顺序执行下列步骤。

大约持续时间：20 分钟。



### 添加事件和数据池项

下列说明将指导您完成添加事件和数据池项的过程。这些事件稍后将用于更改音量。数据池项的用途是在稍后部分中更改图形元素的位置。

#### 步骤 1

转至“事件”元素，并点击 $\oplus$ 。

事件已添加至表中。

#### 步骤 2

将事件重命名为 Volume up。

#### 步骤 3

添加事件并将其重命名为 Volume down。

#### 步骤 4

添加事件并将其重命名为 Close volume control。

#### 步骤 5

转至“数据池”元素，并点击 $\oplus$ 。



菜单将展开。

#### 步骤 6

在菜单中，点击“整数”。

已添加 Integer 类型的数据池项。

#### 步骤 7

将数据池项重命名为 Volume indicator。

您已添加三个事件和一个数据池项。



### 添加动态状态机并对行为进行建模

下列说明将指导您完成添加动态状态机的过程。您建模的触觉动态状态机将用于控制音量。

先决条件：

- 您已完成前述说明。

#### 步骤 1

在“导航”元素中，转至“动态状态机”，然后点击 **+**。

菜单将展开。

#### 步骤 2

在菜单中，点击“触觉动态状态机”。

触觉动态状态机将添加并显示在内容区域中。

#### 步骤 3

将动态状态机重命名为 Volume control。

#### 步骤 4

将初始状态从“工具箱”拖动到动态状态机中。

#### 步骤 5

将视图状态从“工具箱”拖动到动态状态机中。

视图将随视图状态添加至 EB GUIDE 模型。

#### 步骤 6

在“导航”元素中，点击视图状态。

#### 步骤 7

按 **F2** 键，将视图状态重命名为 Volume。

#### 步骤 8

添加从初始状态到 Volume 视图状态的转换。



## 对滑块控件进行建模

下列说明将指导您完成对水平滑块控件指示器进行建模的过程。滑块控件指示器用于显示运行时间的音量。

滑块控件指示器包含两个矩形。一个矩形表示滑块控件的背景。另一个矩形指示音量。

先决条件：

- 您已完成前述说明。

### 步骤 1

在“导航”元素中，展开 Volume 视图状态。双击视图。

内容区域将显示视图。

### 步骤 2

将矩形从“工具箱”拖动到视图中。

### 步骤 3

在“导航”元素中，点击矩形并按 **F2** 键。

### 步骤 4

将矩形重命名为 Slider background。

### 步骤 5

要更改 Slider background 的外观，请点击矩形并转至“属性”元素。

#### 步骤 5.1

在 width 文本框中输入 500。

#### 步骤 5.2

在 x 文本框中输入 125。

#### 步骤 5.3

在 y 文本框中输入 300。

### 步骤 6

将矩形从“工具箱”拖动到“导航”元素中的 Slider background 中。

此矩形将添加为 Slider background 的子代控件。

### 步骤 7

在“导航”元素中，点击矩形并按 **F2** 键。

### 步骤 8

将矩形重命名为 Indicator。

### 步骤 9

要更改 Indicator 的外观，请点击矩形并转至“属性”元素。


#### 步骤 9.1

在 width 文本框中输入 40。

### 步骤 9.2

在 height 文本框中输入 80。

### 步骤 9.3

在 x 属性旁边，点击  按钮。

菜单将展开。

### 步骤 9.4

在菜单中，点击“向数据池项添加链接”。


对话框将打开。

### 步骤 9.5

从列表中，选择 Volume indicator 数据池项。

### 步骤 9.6

点击“接受”。

对话框将关闭。 按钮将显示在 x 属性旁边。x 和 Volume indicator 的值现在已建立链接。

### 步骤 9.7

在 y 文本框中输入 10。

### 步骤 9.8

为 fillColor 属性选择黑色。

您已向视图添加两个矩形。您已更改矩形的外观。

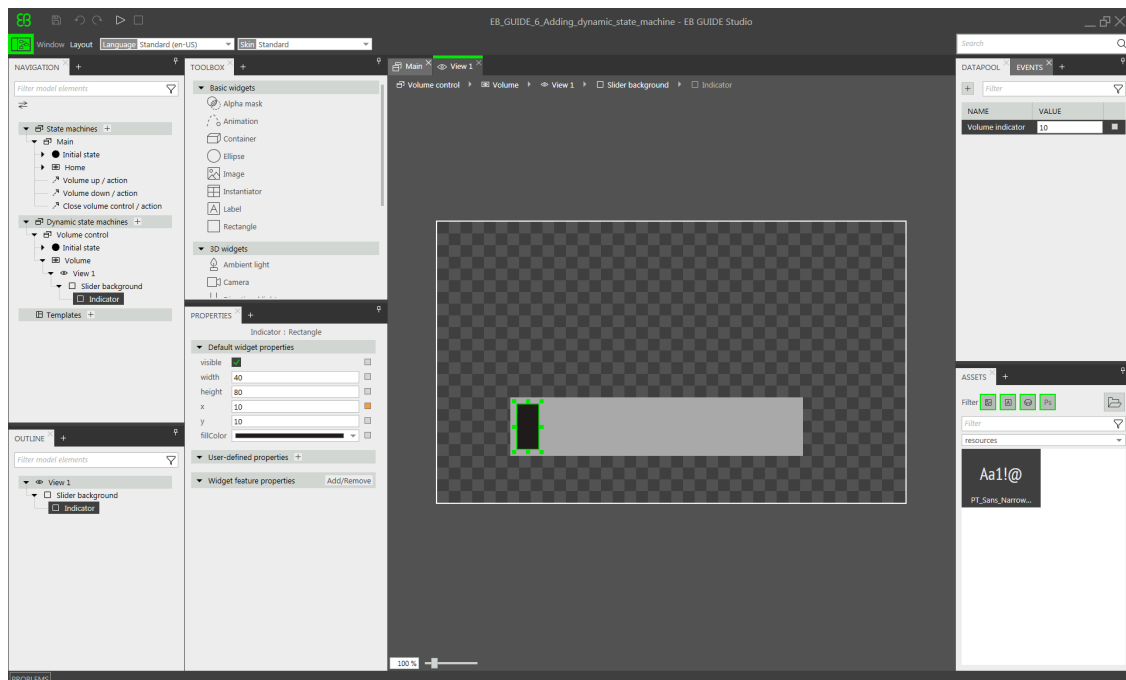


图 11.1. 包含两个矩形的“视图 1”外观

#### 步骤 10

在“数据池”元素中，点击数据池项 Volume indicator。

#### 步骤 11

在 Value 文本框中，输入 10。

在内容区域中，Indicator 矩形将会更改位置。

Volume indicator 数据池项，用于控制矩形 Indicator 的 x 位置。



### 向“主要”状态机添加状态

在下列说明中，您将向“主要”状态机添加一个初始状态和一个视图状态。您将使用视图状态来与其他状态机并行运行动态状态机。

先决条件：

- 您已完成前述说明。

#### 步骤 1

在“导航”元素中，双击“主要”。

“主要”状态机将显示在内容区域中。

#### 步骤 2

将初始状态从“工具箱”拖动到状态机中。

#### 步骤 3

将视图状态从“工具箱”拖动到状态机中。

视图将随视图状态添加至 EB GUIDE 模型。

#### 步骤 4

将视图状态重命名为 Home。

#### 步骤 5

在内容区域中，点击初始状态。

#### 步骤 6

添加从初始状态到 Home 视图状态的转换。

#### 步骤 7

在“导航”元素中，点击“主要”。

#### 步骤 8

在“属性”元素中，选中 Dynamic state machine list 复选框。

完成上述步骤后，您可以使用与动态状态机相关的 EB GUIDE 脚本 函数。

您已向“主要”状态机添加了一个初始状态和一个视图状态。触觉动态状态机将与“主要”状态机并行运行。



### 向“主要”状态机添加内部转换

在下列说明中，您将添加内部转换。您将在运行时间使用内部转换来启动（推送）和停止（弹出）动态状态机。

先决条件：

- 您已完成前述说明。

#### 步骤 1

在“导航”元素中，点击“主要”状态机。

#### 步骤 2

在“属性”元素中，转至“内部转换”，并点击“添加”。

已向状态机添加内部转换。内部转换在“导航”元素内可见。

#### 步骤 3

添加另外两个内部转换。

#### 步骤 4

在“导航”元素中，点击第一个内部转换。

##### 步骤 4.1

转至“属性”元素。

##### 步骤 4.2

在“触发器”组合框中，选择 Volume up。

##### 步骤 4.3

在“操作”属性旁边，点击“添加”。

##### 步骤 4.4

输入下列 EB GUIDE 脚本：

```
function()
{
    dp:"Volume indicator" = dp:"Volume indicator" + 20
    f:pushDynamicStateMachine(popup_stack:Main, sm:"Volume control", 0)
}
```

##### 步骤 4.5

点击“接受”。

操作将添加至转换。在“导航”元素中，将该内部转换重命名为 Volume up。

#### 步骤 5

在“导航”元素中，点击第二个内部转换。

步骤 5.1

转至“属性”元素。

步骤 5.2

在“触发器”组合框中，选择 Volume down。

步骤 5.3

在“操作”属性旁边，点击“添加”。

步骤 5.4

输入下列 EB GUIDE 脚本：

```
function()
{
    dp:"Volume indicator" = dp:"Volume indicator" - 20
    f:pushDynamicStateMachine(popup_stack:Main, sm:"Volume control", 0)
}
```

步骤 5.5

点击“接受”。

操作将添加至转换。在“导航”元素中，将该内部转换重命名为 Volume down。

步骤 6

在“导航”元素中，点击第三个内部转换。

步骤 6.1

转至“属性”元素。

步骤 6.2

在“触发器”组合框中，选择 Close volume control。

步骤 6.3

在“操作”属性旁边，点击“添加”。

步骤 6.4

输入下列 EB GUIDE 脚本：

```
function()
{
    f:popDynamicStateMachine(popup_stack:Main, sm:"Volume control")
}
```

步骤 6.5

点击“接受”。

操作将添加至转换。在“导航”元素中，将该内部转换重命名为 Close volume control。

您已添加三个用于启动和停止动态状态机的内部转换。而且，内部转换 Volume up 和 Volume down 将更改 Indicator 矩形控件的位置。

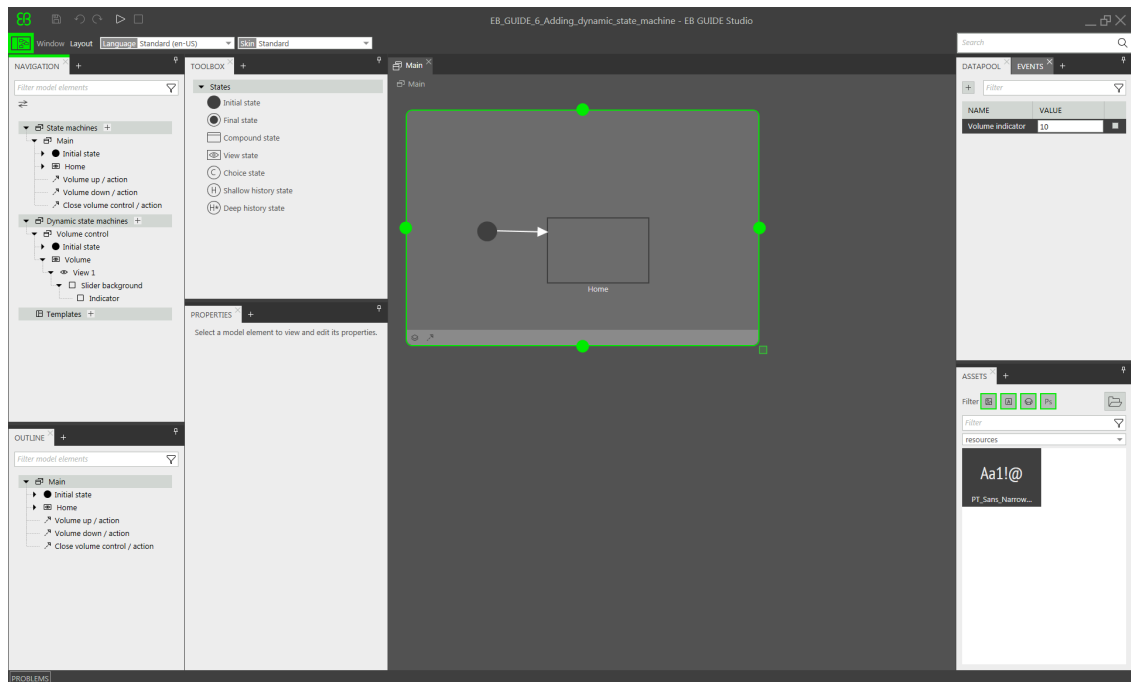


图 11.2. EB GUIDE 包含所有模型元素的 EB GUIDE 模型



## 启动模拟运行并测试 EB GUIDE 模型


先决条件：

- 您已完成前述说明。

要启动模拟运行，请点击命令区的 。

模拟运行和 EB GUIDE Monitor 启动。EB GUIDE 模型显示 Home 视图状态。

### 步骤 1

在 EB GUIDE Monitor 的“事件”元素中，选择 Volume up 事件并点击  来触发事件。

将启动动态状态机并显示滑块控件指示器。动态状态机将覆盖 Home 视图状态。

在触发事件 Volume up 或 Volume down 时，黑色 Indicator 矩形将会移动。如果触发事件 Close volume control，滑块控件将从视图中消失。

如果您向“主要”状态机添加更多状态，Volume control 动态状态机将会覆盖其他状态。

## 11.2. 教程：使用 EB GUIDE 脚本 对按钮行为进行建模

### 注意



#### 默认窗口布局

本用户手册的所有说明和屏幕截图，都使用默认窗口布局。如果您想按照说明进行操作，建议您将 EB GUIDE Studio 窗口设置为默认 布局，方法是选择“布局” > “重置为默认布局”。

使用 EB GUIDE 脚本，您可以表达属性值、操作或条件，并在运行时间对其进行求值。

下列说明将指导您完成使用 EB GUIDE 脚本 对按钮行为进行建模的过程。该按钮在被点击时会增大，在达到定义的最大大小后将收缩为原始大小。为获得最佳效果，请按提供的顺序执行步骤。

大约持续时间：10 分钟。



添加控件

先决条件：

- “主要”状态机包含初始状态和视图状态。
- 初始状态具有到视图状态的转换。
- 内容区域将显示视图。

#### 步骤 1

将矩形从“工具箱”拖入视图中。

#### 步骤 2

在“导航”元素中，点击该矩形，按 F2 键，并将该矩形重命名为 Background。

#### 步骤 3

将矩形从“工具箱”拖入“导航”元素中。将其置为 Background 矩形的子代控件。

#### 步骤 4

在“导航”元素中，点击新矩形，按 F2 键，并将该矩形重命名为 Button。

#### 步骤 5

将标签从“工具箱”拖入“导航”元素中。将标签置为 Button 矩形的子代控件。

#### 步骤 6

在“导航”元素中，点击该标签，按 F2 键，并将该标签重命名为 Button text。

现在，您的控件层次结构如下图所示。



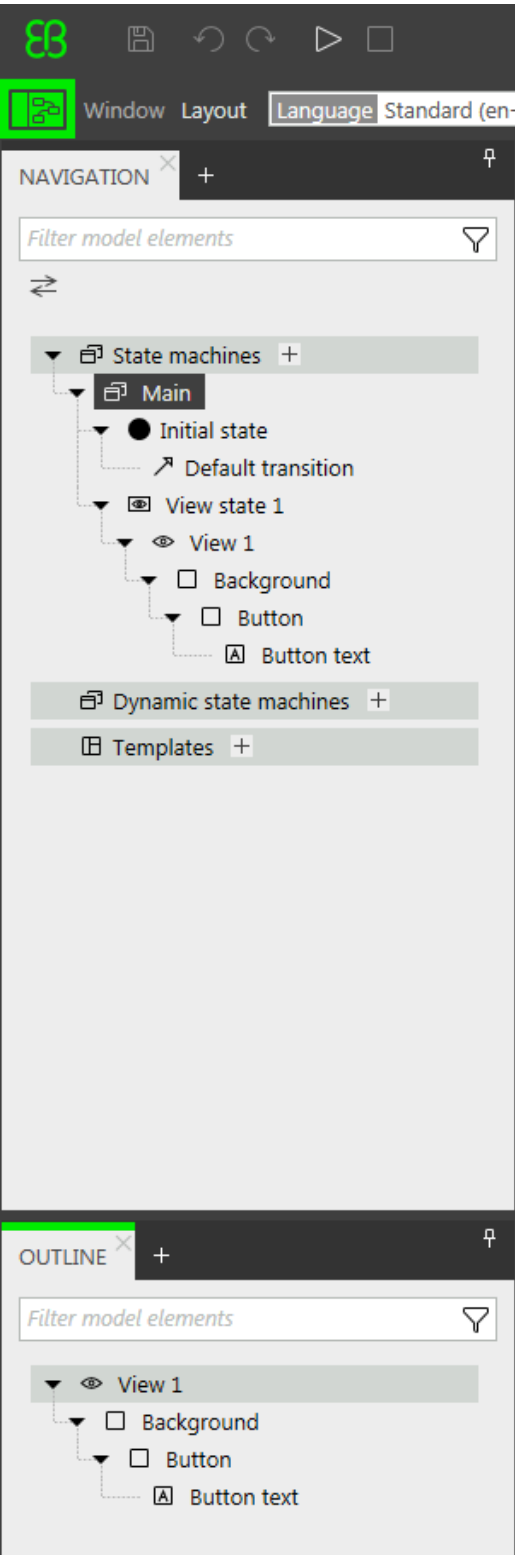


图 11.3. 控件层次结构



## 配置背景


先决条件:

- 您已完成前述说明。

步骤 1

在“导航”元素中，点击 Background 矩形，并转至“属性”元素。

步骤 2

在 width 属性旁边，点击  按钮。

菜单将展开。

步骤 3

在菜单中，点击“向控件属性中添加链接”。


对话框将打开。

步骤 4

在此对话框中，转至视图并选择其 width 属性。

步骤 5

点击“接受”。

对话框将关闭。 按钮将显示在 width 属性旁边。

步骤 6

将 Background 矩形的 height 属性链接至视图的 height 属性。

步骤 7

将 Background 矩形的 x 属性链接至视图的 x 属性。

步骤 8

将 Background 矩形的 y 属性链接至视图的 y 属性。

Background 矩形覆盖了视图的确切大小和位置。



## 定义最大按钮宽度

数据池项保存了按钮的最大宽度值。可以在运行时间对其进行更改。

先决条件:

- 您已完成前述说明。

步骤 1

在“数据池”元素中，点击 .

菜单将展开。

#### 步骤 2

在菜单中，点击“整数”。

已添加 Integer 类型的新数据池项。

#### 步骤 3

将数据池项重命名为 Maximum width。

#### 步骤 4

在 Value 文本框中，输入 400。



### 配置按钮

先决条件：

- 您已完成前述说明。

#### 步骤 1

在“导航”元素中，点击 Button 矩形，并转至“属性”元素。

##### 步骤 1.1

在 height 文本框中输入 50。

##### 步骤 1.2

在 x 文本框中输入 350。

##### 步骤 1.3

在 y 文本框中输入 215。

##### 步骤 1.4

为 fillColor 属性选择蓝色。

该按钮现在变为蓝色。

#### 步骤 2

在“控件功能属性”类别中，点击“添加/删除”。

这将显示“控件功能”对话框。

#### 步骤 3

在“可用控件功能”下，展开“输入处理”类别，然后选择“触摸已按下”控件功能。

#### 步骤 4

点击“接受”。

相关控件功能属性将添加至 Button 矩形中，并显示在“属性”元素中。

#### 步骤 5

点击 touchPressed 属性旁边的“编辑”。

### 步骤 6

使用下列代码替换现有的 EB GUIDE 脚本：

```
function(v:touchId::int, v:x::int, v:y::int, v:fingerId::int)
{
    if (v:this.width > dp:"Maximum width") // If the button has grown
        // beyond its maximum size...
    {
        // ...reset its dimensions to the default values.
        v:this.height = 50
        v:this.width = 100
        v:this.x = 350
        v:this.y = 215
    }
    else // Otherwise...
    {
        // ... increase button size...
        v:this.width += 80
        v:this.height += 40

        // ...and move the button to keep it centered.
        v:this.x -= 40
        v:this.y -= 20
    }
    false
}
```

### 步骤 7

点击“接受”。

您已配置 Button 矩形并编写了用于在运行时间更改 Button 矩形大小的 EB GUIDE 脚本。



### 配置按钮文本

先决条件：

- 您已完成前述说明。

### 步骤 1

在“导航”元素中，点击 Button text 标签，并转至“属性”元素。

### 步骤 2

在 text 文本框中输入 grow!。

### 步骤 3

将 Button text 标签的 width 属性链接至 Button 矩形的 width 属性。

步骤 4

将 Button text 标签的 height 属性链接至 Button 矩形的 height 属性。

步骤 5

在 x 文本框中输入 0。

步骤 6

在 y 文本框中输入 0。

步骤 7

在 horizontalAlign 属性旁边，点击 。

现在，Button text 标签和 Button 矩形的大小和位置都相等。




### 保存和测试 EB GUIDE 模型

先决条件：

- 您已完成前述说明。

步骤 1

要保存项目，请点击命令区的 。

步骤 2

要启动模拟运行，请点击命令区的 。

结果：

模型运行启动您所创建的 EB GUIDE 模型。它的行为如下。

1. 首先，它显示一个灰色屏幕，屏幕中央有一个蓝色按钮。屏幕内容如下所示。



图 11.4. 结果

2. 任何时候点击该按钮时，它会增大，但位置仍保持在屏幕中央。
3. 当按钮达到数据池项 Maximum width 的值时，它会收缩回原始大小和位置。

## 11.3. 教程：对路径动作进行建模

### 注意



#### 默认窗口布局

本用户手册的所有说明和屏幕截图，都使用默认窗口布局。如果您想按照说明进行操作，建议您将 **EB GUIDE Studio** 窗口设置为默认 布局，方法是选择“布局” > “重置为默认布局”。

路径动作是手指在触摸屏上画出的形状或者由一些其他输入设备输入的形状。

下列说明将指导您完成对路径动作进行建模的过程。

大约持续时间：10 分钟



### 添加控件和配置默认控件属性

先决条件：

- “主要”状态机包含初始状态和视图状态。
- 初始状态具有到视图状态的转换。
- 内容区域将显示视图。

#### 步骤 1

将矩形从“工具箱”拖入视图中。

#### 步骤 2

将标签从“工具箱”拖入导航区中。

此标签将添加为矩形的子控件。

“属性”元素显示标签的属性。

#### 步骤 3

在“属性”元素的 width 文本框中，输入 500。

#### 步骤 4

选择矩形。

“属性”元素显示该矩形的属性。

#### 步骤 5

在 width 文本框中输入 500。

#### 步骤 6

在“属性”元素中，转至 “fillColor” 并选择红色。

您已添加两个控件并配置了默认控件属性。



### 向矩形中添加控件功能

要使用户输入从控件开始的形状，请向矩形中添加“路径动作”控件功能。此形状与一组已知形状以及某个已识别的操作动作（如果找到了匹配项）进行比较。

先决条件：

- 您已完成前述说明。

#### 步骤 1

选择矩形。

“属性”元素显示该矩形的属性。



### 步骤 2

在“属性”元素中，转至“控件功能属性”，并点击“添加/删除”。

这将显示“控件功能”对话框。

### 步骤 3

在“可用控件功能”下，展开“操作动作”类别，然后选择 Path gestures。

“已触摸”控件功能自动处于选中状态，因为它是“动作”控件功能必需的。

### 步骤 4

点击“接受”。

相关控件功能属性将添加至矩形，并显示在“属性”元素中。

### 步骤 5

对于“路径动作”控件功能，请编辑下列属性：

#### 步骤 5.1

点击 onPath 属性旁边的“编辑”。

#### 步骤 5.2

输入下列 EB GUIDE 脚本：

```
function(v:gestureId::int)
{
    v:this->"Label 1".text = "recognized path gesture #"
    + f:int2string(v:gestureId);
}
```

#### 步骤 5.3

点击“接受”。

#### 步骤 5.4

点击 onPathStart 属性旁边的“编辑”。

#### 步骤 5.5

输入下列 EB GUIDE 脚本：

```
function()
{
    v:this->"Label 1".text = "path gesture start";
}
```

#### 步骤 5.6

点击“接受”。

#### 步骤 5.7

点击 onPathNotRecognized 属性旁边的“编辑”。

#### 步骤 5.8

输入下列 EB GUIDE 脚本：



```
function()
{
  v:this->"Label 1".text = "shape not recognized";
}
```

#### 步骤 5.9

点击“接受”。

#### 步骤 6

要启动模拟运行，请点击命令区的 ▶。

模拟运行和 EB GUIDE Monitor 启动。要查看反应，请使用鼠标在矩形内画一个形状。

## 11.4. 教程：使用动态内容创建 列表

### 注意



#### 默认窗口布局

本用户手册的所有说明和屏幕截图，都使用默认窗口布局。如果您想按照说明进行操作，建议您将 EB GUIDE Studio 窗口设置为默认 布局，方法是选择“布局” > “重置为默认布局”。

利用实例化器可以在运行时间动态创建列表。基于列表类型的数据池项，实例化器会按预定义的布局显示所有列表元素。如果数据池项的内容发生修改，实例化器的外观也会变化。

下列说明将指导您完成使用动态内容创建 列表的过程。每个列表元素包含一个带有标签的矩形。

大约持续时间：15 分钟。



### 添加数据池项

下列说明指导您完成添加类型 String list 的数据池项过程。数据池项为实例化器的每个列表元素都提供了一个值。如果数据池项的内容发生修改，实例化器的外观也会变化。

先决条件：

- “主要”状态机包含初始状态和视图状态。
- 初始状态具有到视图状态的转换。

#### 步骤 1

要显示列表中的内容，请添加类型为 String list 的数据池项。

在“数据池”元素中，点击 +。

菜单将展开。

#### 步骤 2


在菜单中，点击“字符串列表”。

已添加 String list 类型的新数据池项。

#### 步骤 3

将数据池项重命名为 MyStringList。

#### 步骤 4

点击  按钮。

将打开编辑器。

##### 步骤 4.1

点击“添加”。

新条目已添加至表。

##### 步骤 4.2

在 Value 文本框中输入 One。

##### 步骤 4.3

将值 Two、Three、Four 和 Five 添加到数据池项 MyStringList。

##### 步骤 4.4

点击“接受”。

您已添加一个类型 String list 的数据池项。该数据池项包含五个条目。

列表内容将显示在 Value 属性旁边。



添加控件

先决条件：

- 您已完成前述说明。

#### 步骤 1

要将控件添加到视图，请双击内容区域的视图状态。

视图将显示在内容区域中。

#### 步骤 2

在“导航”元素中，展开视图状态和视图。

#### 步骤 3

将实例化器从“工具箱”拖入视图中。将实例化器重命名为 MyInstantiator。

#### 步骤 4

将矩形从“工具箱”拖入实例化器中。将矩形重命名为 MyRectangle。

### 步骤 5

将标签从“工具箱”拖入导航区中。将标签重命名为 MyLabel。

此时的控件层次结构如下图所示。

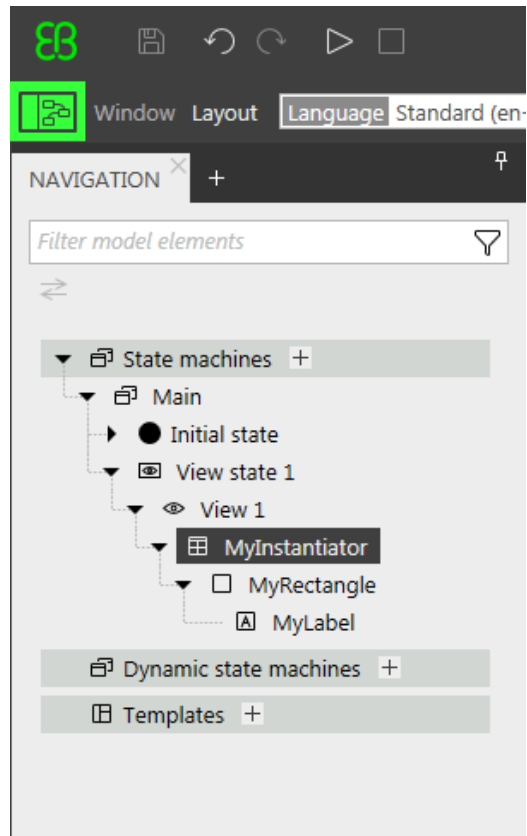


图 11.5. 包含实例化器的控件层次结构



## 配置实例化器

先决条件：

- 您已完成前述说明。

### 步骤 1

要更改 MyInstantiator 的属性，请选中实例化器并转至“属性”元素。

### 步骤 2

在 width 文本框和 height 文本框中，输入 300。

### 步骤 3

在 x 文本框中输入 250。



#### 步骤 4

在 y 文本框中输入 150。

#### 步骤 5

要动态计算列表长度，添加条件脚本。

在“用户定义的属性”类别中，点击 **+**。

菜单将展开。

##### 步骤 5.1

在菜单中，点击“条件脚本”。

##### 步骤 5.2

将该属性重命名为 calculateNumItems。

##### 步骤 5.3

在 calculateNumItems 属性旁边，点击“编辑”。

将在内容区域打开脚本编辑器。

##### 步骤 5.4

将 MyStringList 数据池项添加到“触发器”列表。

##### 步骤 5.5

输入以下 "On trigger" 脚本：

```
function(v:arg0::bool)
{
  v:this.numItems = length dp:MyStringList;
  false
}
```

根据 MyStringList 的内容，添加自动更改列表条目数量的脚本。

#### 步骤 6

要在实例化器中排列所有标签，向其添加布局。

在“控件功能属性”类别中，点击“添加/删除”。

这将显示“控件功能”对话框。

##### 步骤 6.1

在“可用控件功能”下，展开“布局”类别，然后选择“框布局”控件功能以并排排列标签。

相关控件功能属性将添加至实例化器并显示在“属性”元素中。

##### 步骤 6.2

点击“接受”。

##### 步骤 6.3

在 gap 文本框中输入 5，在每个列表元素之间设置 5 像素的间距。

##### 步骤 6.4

从 "layoutDirection" 下拉列表框中选择“垂直 (=1)”，以便在彼此之间排列标签。

您配置了实例化器，它定义了列表的视觉外观，并动态地调整列表项的数量。



#### 配置列表元素文本

先决条件：

- 您已完成前述说明。

##### 步骤 1

要更改标签的外观，请选中标签并转至“属性”元素。


##### 步骤 2

在 x 和 y 文本框中输入 0。

##### 步骤 3

将标签的 width 属性的链接添加到矩形的 width 属性中。

##### 步骤 3.1

在 width 属性旁边，点击  按钮。

菜单将展开。

##### 步骤 3.2

在菜单中，点击“向控件属性中添加链接”。


对话框将打开。

##### 步骤 3.3

在此对话框中，转至矩形并选择其 width 属性。

##### 步骤 3.4

点击“接受”。

对话框将关闭。 按钮将显示在 width 属性旁边。

##### 步骤 4

将标签的 height 属性的链接添加到矩形的 height 属性中。

##### 步骤 5

在 horizontalAlign 属性旁边，点击 。

您已更改标签的外观。标签现在在矩形中居中。



#### 配置列表元素

先决条件：

- 您已完成前述说明。

#### 步骤 1

要更改矩形的外观，请选中矩形并转至“属性”元素。

#### 步骤 2

为确保列表元素使用可用宽度，请将矩形的 width 属性的链接添加到实例化器的 width 属性中。

#### 步骤 3

在 height 文本框中输入 50。

#### 步骤 4

要为列表的每一行定义唯一的位置，需要添加“列表索引”控件功能。

##### 步骤 4.1

在“控件功能属性”类别中，点击“添加/删除”。

这将显示“控件功能”对话框。

##### 步骤 4.2

在“可用控件功能”下，展开“列表管理”类别，然后选择“行索引”控件功能。

lineIndex 属性将添加到矩形的属性。

#### 步骤 5

要使用 MyStringList 的内容填充列表的标签，请添加一个条件脚本。

##### 步骤 5.1

在“用户定义的属性”类别旁，点击 **+**。

菜单将展开。

##### 步骤 5.2

在菜单中，点击“条件脚本”。

##### 步骤 5.3

将该属性重命名为 setText。

##### 步骤 5.4

点击 setText 属性旁边的“编辑”。

将在内容区域打开脚本编辑器。

##### 步骤 5.5

将矩形的 lineIndex 属性和 MyStringList 数据池项添加到“触发器”列表中。

##### 步骤 5.6

输入以下 "On Trigger" 脚本：

```
function(v:arg0::bool)
{
    v:this->MyLabel.text=dp:MyStringList[v:this.lineIndex];
    false
}
```

您已更改矩形的外观。使用 setText 属性时，MyStringList 的标签自动以 MyStringList 的内容进行填充。



## 测试 EB GUIDE 模型

先决条件:

- 您已完成前述说明。

步骤 1

要启动模拟运行，请点击命令区的 ▶。

结果:

由于 MyStringList 包含五个数据池项，五个标记为一到五的矩形显示为垂直排列。

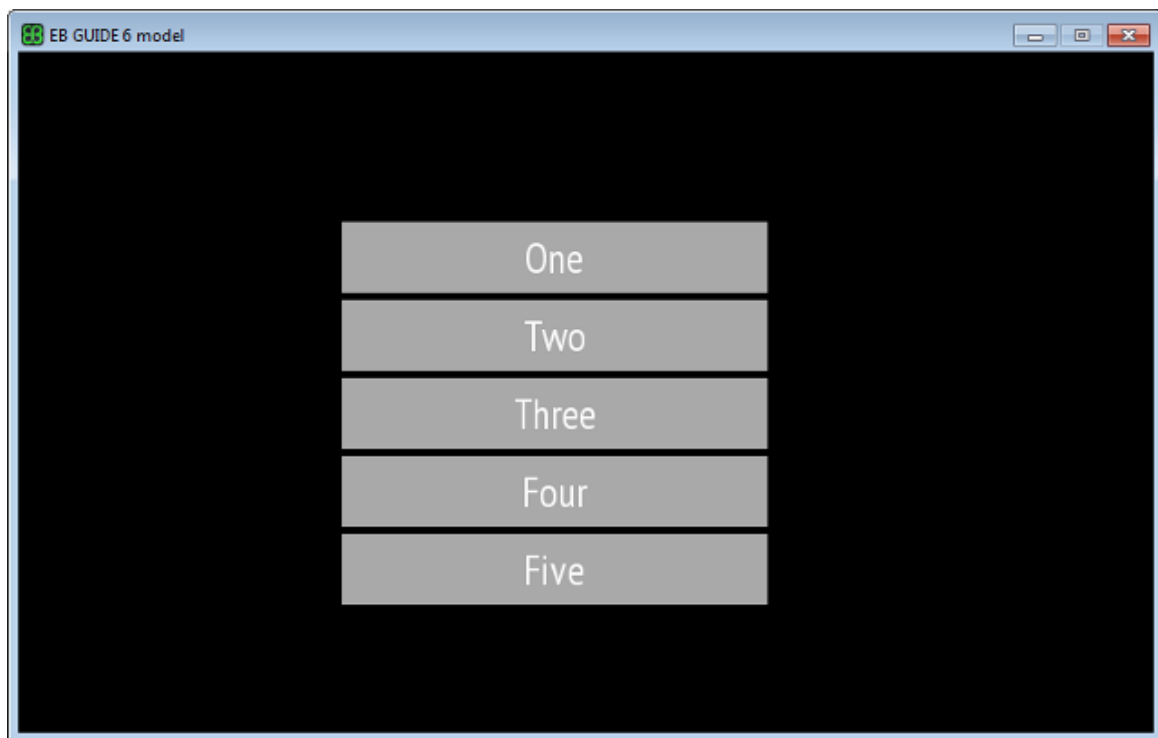


图 11.6. 使用实例化器创建的列表

## 11.5. 教程：使椭圆移动经过屏幕

注意



默认窗口布局

本用户手册的所有说明和屏幕截图，都使用默认窗口布局。如果您想按照说明进行操作，建议您将 EB GUIDE Studio 窗口设置为默认布局，方法是选择“布局”>“重置为默认布局”。

下列说明将指导您完成以动画方式显示椭圆，使其在模拟运行开始后连续移动经过屏幕的过程。

大约持续时间：**5 分钟**。



#### 添加控件

在下列步骤中，您将向视图添加三个控件，并组织这些控件的层次结构。

先决条件：

- 内容区域将显示“主要”状态机。
- “主要”状态机包含初始状态和视图状态。
- 初始状态具有到视图状态的转换

##### 步骤 1

在内容区域中，双击视图状态。

视图将显示在内容区域中。

##### 步骤 2

将椭圆从“工具箱”拖入视图中。

##### 步骤 3

将动画从“工具箱”拖入椭圆中。

##### 步骤 4

在“导航”元素中，点击动画并按 **F2** 键。将动画重命名为 MyAnimation。

##### 步骤 5

将线性插值整数控件，从“工具箱”拖入“导航”元素中，使其成为动画的子控件。

现在，如果启动模拟运行，视图中将会显示一个椭圆。该椭圆尚未移动。



#### 添加用户定义的以下类型的属性 Conditional script

下一步，向椭圆中添加用户定义的属性。使用条件脚本属性，在模拟运行启动动画时渲染该椭圆。

先决条件：

- 您已完成前述说明。

##### 步骤 1

选择该椭圆。

##### 步骤 2

在“属性”元素中，转至“用户定义的属性”类别，并点击 **+**。



菜单将展开。

#### 步骤 3

在菜单中，点击 Conditional script。

向椭圆添加用户定义的 Conditional script 类型的属性。

#### 步骤 4

将该属性重命名为 startAnimation。

#### 步骤 5

点击 startAnimation 属性旁边的“编辑”。

将在内容区域打开脚本编辑器。

#### 步骤 6

输入下列 EB GUIDE 脚本：

```
function(v:arg0::bool)
{
    f:animation_play(v:this->MyAnimation)
}
```



### 使动画可见

下列说明将指导您完成使动画可见的过程。

先决条件：

- 您已完成前述说明。

#### 步骤 1

转至“动画”编辑器，并点击“动画属性”旁边的 **+**。


菜单将展开。

#### 步骤 2

在 Ellipse 1 下，依次选择 x 属性和“线性插值曲线”。

#### 步骤 3

点击“接受”。

 按钮将显示在 target 属性旁。

#### 步骤 4

将 end 属性链接至视图的 width 属性。

完成上述设置后，当动画启动时，矩形的 x 属性，将从零变为视图的宽度。因此，椭圆将从视图的左侧边界移至右侧边界。


**步骤 5**

要使动画无限次重复播放，请在 repeat 属性中输入 0。

**步骤 6**

保存项目。

**步骤 7**

要启动模拟运行，请点击命令区的 。

结果：

椭圆将从视图的左侧连续移动至视图的右侧。

## 11.6. 教程：向数据池项添加语言依赖项文本

**注意****默认窗口布局**

本用户手册的所有说明和屏幕截图，都使用默认窗口布局。如果您想按照说明进行操作，建议您将 EB GUIDE Studio 窗口设置为默认 布局，方法是选择“布局” > “重置为默认布局”。

EB GUIDE 可以以用户首选的语言显示文本。下列说明展示了如何创建一个随英语、法语和德语用户界面更改的标签模型。

大约持续时间：15 分钟

**注意****语言依赖项的必备条件**

将语言支持添加到数据池的步骤如下：

- ▶ 如果其 Value 属性已与其他数据池项或控件属性相关联，将此链接移除。
- ▶ 如果其 Value 属性为脚本化的值，将属性转化成纯值。

**将控件属性链接到数据池项**

下列说明将指导您完成将标签的 text 属性链接至数据池项的过程。在运行时刻，显示文本将由数据池项提供。

先决条件：


- 将以下三种语言添加至 EB GUIDE 模型：英语、德语和法语。“语言 1”的名称设置为 German，而“语言 2”的名称设置为 French。
- 内容区域将显示“主要”状态机。

- 主要状态机包含初始状态和视图状态。
- 内容区域将显示视图。
- 初始状态具有到视图状态的转换。
- 视图包含标签。

#### 步骤 1

点击标签。

#### 步骤 2

在“属性”元素中，转至 text 属性，并点击属性旁的  按钮。

#### 步骤 3

在菜单中，点击“向数据池项中添加链接”。

对话框将打开。

#### 步骤 4

要添加新的数据池项，请在文本框中输入 Welcome\_text。

#### 步骤 5

点击“添加数据池项”。

#### 步骤 6

点击“接受”。

已添加数据池项 Welcome\_text。

在内容区域中，标签不再显示任何文本。



### 向数据池项输入语言依赖项文本

下列说明将指导您，完成向数据池项添加语言依赖项文本。对于每种语言，Value 属性具有不同的文本。

先决条件：

- 您已完成前述说明。

#### 步骤 1

在“数据池”元素中，点击数据池项 Welcome\_text。

#### 步骤 2

点击  按钮。

#### 步骤 3

在菜单中，点击“添加语言支持”。

在“属性”元素中，显示语言属性。

#### 步骤 4

在“数据池”元素的 Value 文本框中，输入 Welcome。

在内容区域中，标签将显示 Welcome。

#### 步骤 5

转至“属性”元素。

#### 步骤 6

在 German 文本框中，输入 Willkommen。

在 Language 框的左上角，将语言更改为 German。

在内容区域中，标签将显示 Willkommen。

#### 步骤 7

在 French 文本框中，输入 Bienvenue。

在 Language 框的左上角，将语言更改为 French。

在内容区域中，标签将显示 Bienvenue。

您已经加英语、德语和法语的语言支持，并且定义了语言依赖项标签。



### 在运行时间更改语言

以下说明指导您完成创建用于在运行时间更改语言的脚本。用户每次点击标签时，显示语言都会改变。

先决条件：

- 您已完成前述说明。

#### 步骤 1

在“数据池”元素中，点击 **+**。

菜单将展开。

#### 步骤 2

在菜单中，点击 Integer。

已添加 Integer 类型的数据池项。

#### 步骤 3

将数据池项重命名为 SelectedLanguage。

#### 步骤 4

在“导航”元素中，点击 Label 1 标签。

#### 步骤 5

在“属性”元素中，转至“控件功能属性”并点击“添加/删除”。

这将显示“控件功能”对话框。

#### 步骤 6

在“可用控件功能”下，展开“输入处理”类别，然后选择“触摸已按下”控件功能。

步骤 7

点击“接受”。

相关控件功能属性将添加至标签，并显示在“属性”元素中。

步骤 8

点击 touchPressed 属性旁边的“编辑”。

步骤 9

使用下列代码替换现有的 EB GUIDE 脚本：

```
function(v:touchId::int, v:x::int, v:y::int, v:fingerId::int)
{
    if (dp:SelectedLanguage == 0) // Standard selected
    {
        f:language(1:German)
        dp:SelectedLanguage = 1
    }
    else if (dp:SelectedLanguage == 1) // German selected
    {
        f:language(1:French)
        dp:SelectedLanguage = 2
    }
    else if (dp:SelectedLanguage == 2) // French selected
    {
        f:language(1:Standard)
        dp:SelectedLanguage = 0
    }
    false
}
```

步骤 10

点击“接受”。

您已配置标签并编写了用于在运行时刻更改标签语言的 EB GUIDE 脚本。

结果：

您已将 String 类型的数据池项添加至 EB GUIDE 模型。该数据池项具有不同的语言值。在英语中，该值为 Welcome。在德语中，该值为 Willkommen。在法语中，该值为 Bienvenue。该数据池项已链接到标签的 text 属性。每次更改 EB GUIDE 模型的语言时，该标签的文本也会发生更改。

## 11.7. 教程：使用 3D 图形

### 注意



#### 默认窗口布局

本用户手册的所有说明和屏幕截图，都使用默认窗口布局。如果您想按照说明进行操作，建议您将 **EB GUIDE Studio** 窗口设置为默认 布局，方法是选择“布局” > “重置为默认布局”。

EB GUIDE Studio 可以实现在 EB GUIDE 模型中使用 3D 图形。

下列说明将指导您完成将 3D 图形添加到 EB GUIDE 模型的过程。该说明显示如何导入 3D 图形以及如何使用控件特性修改导入的 3D 图形的外观。为获得最佳效果，请按提供的顺序执行下列步骤。

### 注意



#### 3D 图形

要创建 3D 图形文件，请使用第三方 3D 建模软件。

只有 OpenGL ES 版本 2.0或更高版本和 DirectX 11 渲染器才能显示 3D 图形。确保您的图形驱动程序与渲染器的版本兼容。受支持的 3D 图形格式为 COLLADA (.dae) 和 Filmbox (.fbx)。要获得最佳结果，请使用 Filmbox 格式。

为了能够将纹理应用于网格，3D 对象需要具有纹理坐标。要添加纹理坐标，请使用第三方 3D 建模软件。

大约持续时间：15 分钟。



#### 导入 3D 图形。

下列说明将指导您完成将 3D 图形导入到 EB GUIDE Studio 项目的过程。

先决条件：

- 内容区域将显示“主要”状态机。
- “主要”状态机包含初始状态和视图状态。
- 初始状态具有到视图状态的转换。
- 3D 图形文件可用。该文件包含摄像头、光源和对象（包含网格和至少一种材质）。

#### 步骤 1

在内容区域中，双击视图状态。

视图将显示在内容区域中。

#### 步骤 2

将场景图从“工具箱”拖动到视图中。

视图显示空的边界框。

#### 步骤 3

将场景图重命名为 My3DGraphic。

#### 步骤 4

在“属性”元素中，点击“导入文件”。

对话框将打开。

#### 步骤 5

转至储存 3D 图形文件的目录。

#### 步骤 6

选择 3D 图形文件。

#### 步骤 7

点击“打开”。

导入开始。将显示“成功导入”对话框。此处可以检查导入日志文件。

#### 步骤 8

点击“确定”。

视图将显示该 3D 图形。“导航”元素显示导入的控件树，其中场景图作为父代节点。My3DGraphic 包含 RootNode，具体取决于您的 3D 图形文件的内容，该节点至少有一个包含材料、摄像头和其他几个子控件的网格。



添加控件

下列说明将指导您完成向 3D 图形添加额外光源的过程。

先决条件：

- 您已完成前述说明。

#### 步骤 1

在“导航”元素中，展开 RootNode。

#### 步骤 2

将定向光从“工具箱”拖动到 RootNode。

您向 My3DGraphic 添加了定向光。您可以使用 RootNode 的变换属性来操作并变换此定向光。

#### 步骤 3

要添加光源，并将其与不同于 RootNode 场景图的默认控件属性放在一起，请执行以下操作：

##### 步骤 3.1

将场景图节点从“工具箱”拖动到 RootNode。

##### 步骤 3.2

将场景图节点重命名为 MyLight。

### 步骤 3.3

将定向光从“工具箱”拖动到 MyLight。

您向 My3DGraphic 添加了定向光。要更改定向光的放置，请更改 MyLight 的属性。



### 更改网格

先决条件：

- 您已完成前述说明。
- `$GUIDE_PROJECT_PATH/<project name>/resources/<3D graphic name>` 文件夹包含额外的 .ebmesh 文件。

### 步骤 1

在“导航”元素中，点击 Mesh 1 并转至“属性”元素。

### 步骤 2

在 mesh 组合框中，从前面提到的资源文件夹中选择 .ebmesh 文件。

视图将以新网格显示场景图。

### 步骤 3

或者，将 .ebmesh 文件从“资产”元素中拖入 mesh 下拉列表框。

视图将以新网格显示场景图。



### 更改纹理

下列说明将指导您完成向 3D 图形添加和修改纹理的过程。

先决条件：

- 您已完成前述说明。
- `$GUIDE_PROJECT_PATH/<project name>/resources/<3D graphic name>` 目录包含 .png 或 .jpg 图像文件。

### 步骤 1

在“导航”元素中，点击材料并转至“属性”元素。

### 步骤 2

在“控件功能属性”类别中，点击“添加/删除”。

这将显示“控件功能”对话框。

### 步骤 3

在“可用控件功能”下方，展开“3D”类别，然后选择“弥散纹理”等纹理控件功能。

### 步骤 4

点击“接受”。





相关控件功能属性将添加至材料，并显示在“属性”元素中。

步骤 5

从“属性”元素的 diffuseTexture 组合框中选择一个图像。

视图将以新纹理显示场景图。

注意



**"3D" 控件功能的用途**

这些说明可用于以下来自 "3D" 类别的控件功能：

- ▶ “环境纹理”
- ▶ “弥散纹理”
- ▶ “放射纹理”
- ▶ “光地图纹理”
- ▶ “正常贴图纹理”
- ▶ “不透明纹理”
- ▶ “反射纹理”
- ▶ “镜面纹理”



多次显示 3D 对象

以下说明指导您完成添加其他摄像机的过程，以便能够多次显示 3D 图形的 3D 对象。您将能够拥有相同对象的不同视点。

先决条件：

- 您已完成前述说明。

步骤 1

在“导航”元素中，点击 My3DGraphic 并转至“属性”元素。

步骤 2

在 width 文本框中输入 800，在 height 文本框中输入 480。

My3DGraphic 场景图有视图的大小。

步骤 3

在“导航”元素中，展开 RootNode 和 Camera001。

步骤 4

单击 Camera 1 并转至“属性”元素。

步骤 5

在“控件功能属性”类别中，点击“添加/删除”。



这将显示“控件功能”对话框。

#### 步骤 6

在“可用控件功能”下，展开 "3D" 类别，然后选择“摄像头视见区”。

#### 步骤 7

点击“接受”。

相关控件功能属性将添加至 Camera 1 并显示在“属性”元素中。

#### 步骤 8

将相机从“工具箱”拖动到场景图节点 Camera001。

您添加了第二个摄像头。

#### 步骤 9

单击 Camera 2 并转至“属性”元素。

#### 步骤 10

在 nearPlane、farPlane 和 fieldOfView 文本框中，输入 Camera 1 具有的相同的值。

Camera 1 和 Camera 2 都具有相同的查看位置。

#### 步骤 11

在“控件功能属性”类别中，点击“添加/删除”。

这将显示“控件功能”对话框。

#### 步骤 12

在“可用控件功能”下，展开 "3D" 类别，然后选择“摄像头视见区”。

#### 步骤 13

点击“接受”。

相关控件功能属性将添加至 Camera 2 并显示在“属性”元素中。

#### 步骤 14

在“属性”元素的 viewportX 和 viewportY 文本框中，输入 100。

在视图中，3D 对象以不同的 x 坐标和 y 坐标显示两次。



## 12. 参考

以下章节为您提供了列表和表（例如，参数、属性和识别符等）。

### 12.1. Android 事件

Android 事件属于 SystemNotifications 事件组并且具有事件组 ID 13。

表 12.1. Android 事件

事件 ID	名称	说明
1	RendererEnabled	<p>当 <b>Android</b> 生命周期管理停止或启动渲染器时，由应用程序发送</p> <p>参数：</p> <ul style="list-style-type: none"><li>▶ enabled: 如果为 true，则渲染器处于启用状态。如果为 false，则渲染器设置为睡眠模式。</li></ul>
2	setKeyboardVisibility	<p>如果要显示虚拟键盘，则由 <b>EB GUIDE</b> 模型发送</p> <p>参数：</p> <ul style="list-style-type: none"><li>▶ visibility: 如果为 true，则虚拟键盘可见。如果为 false，则它不可见。</li></ul>
3	onKeyboardVisibilityChanged	<p>如果显示了虚拟键盘，则由应用程序发送</p> <p>参数：</p> <ul style="list-style-type: none"><li>▶ visibility: 如果为 true，则虚拟键盘可见。如果为 false，则它不可见。</li></ul>
4	onLayoutChanged	<p>当屏幕的可见区域发生变化时，由应用程序发送</p> <p>参数（以像素为单位）：</p> <ul style="list-style-type: none"><li>▶ x: 屏幕可见区域左上角的 <b>x</b> 坐标</li><li>▶ y: 屏幕可见区域左上角的 <b>y</b> 坐标</li><li>▶ width: 屏幕可见区域的宽度</li></ul>



事件 ID	名称	说明
		▶ height: 屏幕可见区域的高度

## 12.2. 数据池项

表 12.2. 数据池项的属性

属性名称	说明
Value	数据池项的初始值

## 12.3. 数据类型

以下部分说明了 EB GUIDE 中的数据类型。您可以根据下面列出的类型添加用户定义属性和数据池项。

### 12.3.1. 网格

网格定义 3D 对象的形状。

可用运算如下所示：

- ▶ 赋值（可写属性）(=)

可以在列表中存储网格。有关列表的详细信息，请参见[节12.3.9](#)。

### 12.3.2. 布尔值

布尔值属性可以具有值 **true** 和 **false**。

可用运算如下所示：

- ▶ 等于 (==)
- ▶ 不等于 (!=)
- ▶ 非 (!)
- ▶ 与 (&&)
- ▶ 或 (||)
- ▶ 赋值（可写属性）(=)

可以在列表中存储布尔值属性。有关列表的详细信息，请参见[节12.3.9](#)。

### 12.3.3. 颜色

颜色以 **RGBA8888** 格式存储。

示例：不透明的红色为 **(255, 0, 0, 255)**。

可用运算如下所示：

- ▶ 等于 (==)
- ▶ 不等于 (!=)
- ▶ 赋值（可写属性）(=)

可以在列表中存储颜色属性。有关列表的详细信息，请参见[节12.3.9](#)。

### 12.3.4. 条件脚本

条件脚本用于对初始化和触发作出反应。在您编辑条件脚本时，内容区域分为以下几个部分。

- ▶ “触发器”组合框包含用于触发执行 “On trigger” 脚本的事件和数据池项的列表。
- ▶ 在初始化时，或者在事件触发或数据池项的值更新后，将调用 “On trigger” 脚本。

“On trigger” 脚本的参数指示了运行脚本的原因。

“On trigger” 脚本的返回值用于控制属性的更改通知。

如果为 **true**，则它将触发更改通知。

如果为 **false**，则它不触发更改通知。

### 12.3.5. 浮点值

浮点数数据类型表示单精度 **32 位 IEEE 754** 值。

可用运算如下所示：

- ▶ 等于 (==)
- ▶ 不等于 (!=)
- ▶ 大于 (>)

- ▶ 大于或等于 ( $\geq$ )
- ▶ 小于 ( $<$ )
- ▶ 小于或等于 ( $\leq$ )
- ▶ 加 (+)
- ▶ 减 (-)
- ▶ 乘 (\*)
- ▶ 除 (/)
- ▶ 赋值 (可写属性) (=)

可以在列表中存储浮点值属性。有关列表的详细信息，请参见[节12.3.9](#)。

### 12.3.6. 字体

要向 **EB GUIDE** 项目中添加字体，请复制以下目录中的字体文件： `$GUIDE_PROJECT_PATH/<project name>/resources`

可用运算如下所示：

- ▶ 赋值 (可写属性) (=)

可以在列表中存储字体属性。有关列表的详细信息，请参见[节12.3.9](#)。

### 12.3.7. 图像

要向 **EB GUIDE** 项目中添加图像，请复制以下目录中的图像文件： `$GUIDE_PROJECT_PATH/<project name>/resources`

可用运算如下所示：

- ▶ 赋值 (可写属性) (=)

可以在列表中存储图像属性。有关列表的详细信息，请参见[节12.3.9](#)。

### 12.3.8. 整数

**EB GUIDE** 支持带符号的 32 位整数。

可用运算如下所示：



- ▶ 等于 (==)
- ▶ 不等于 (!=)
- ▶ 大于 (>)
- ▶ 大于或等于 (>=)
- ▶ 小于 (<)
- ▶ 小于或等于 (<=)
- ▶ 加 (+)
- ▶ 减 (-)
- ▶ 乘 (\*)
- ▶ 除 (/)
- ▶ 模数 (%)
- ▶ 赋值（可写属性）(=)

可以在列表中存储整数属性。有关列表的详细信息，请参见[节12.3.9](#)。

### 12.3.9. 列表

EB GUIDE 支持一系列具有同一数据类型的值。

以下列表类型可用：

- ▶ 网格列表
- ▶ 布尔值列表
- ▶ 颜色列表
- ▶ 浮点值列表
- ▶ 字体列表
- ▶ 图像列表
- ▶ 整数列表
- ▶ 字符串列表

无法在列表中使用以下类型：

- ▶ 列表
- ▶ 属性参考
- ▶ 列表元素参考



可用运算如下所示：

- ▶ 长度：(length)
- ▶ 元素访问器：([])

## 12.3.10. 字符串

EB GUIDE 支持字符串，例如 `Hello world`。

可用运算如下所示：

- ▶ 等于（区分大小写）(==)
- ▶ 不等于（区分大小写）(!=)
- ▶ 等于（不区分大小写，仅在 ASCII 码范围内）(=Aa=)
- ▶ 大于 (>)
- ▶ 大于或等于 (>=)
- ▶ 小于 (<)
- ▶ 小于或等于 (<=)
- ▶ 串联 (+)
- ▶ 赋值（可写属性）(=)

可以在列表中存储字符串属性。有关列表的详细信息，请参见[节12.3.9](#)。

# 12.4. EB GUIDE 脚本

## 12.4.1. EB GUIDE 脚本 关键字

以下是 EB GUIDE 脚本 中的保留关键字的列表。如果您要在脚本中将这些关键字用作识别符，则必须用引号将它们括起来。

关键字	说明
color:	随后是颜色参数，例如 {0, 255, 255}。
dp:	后跟数据池项。
l:	最后是语言。





关键字	说明
else	if 条件已完成。以下块作为替代项执行。
ev:	随后是事件。
f:	后跟用户定义函数。
false	布尔文本值
fire	触发事件
if	后跟用于测试布尔表达式的语句。如果此表达式为 true，则将执行该语句。
in	是局部变量声明和变量的使用范围之间的分隔符  match_event 和let 配合使用。
function	声明函数
length	属性的长度
let	声明可以在范围内访问的局部变量
list	声明列表类型，例如整数列表
match_event	检查当前事件是否与预期事件对应，并声明变量（如：） let
popup_stack	定义了动态状态机的优先级的 动态状态机列表
sm:	随后是状态机
true	布尔文本值
unit	类型为空的值
v:	后跟局部变量。
while	只要条件为 true，就会重复执行语句

## 12.4.2. EB GUIDE 脚本 运算符优先顺序

以下是 EB GUIDE 脚本 中的运算符的列表及其优先顺序和结合性。运算符按照优先级降序从顶部排到底部。

表 12.3. EB GUIDE 脚本 运算符优先顺序

运算符	结合性
( ), { }	无
[ ]	无
(->)	左
(.)	无
::	左



运算符	结合性
length	无
(&)	右
(!), (-) 一元减号	右
(*), (/), (%)	左
(!), (&)	左
(<), (>), (<=), (>=)	左
(!=), (==), (=Aa=)	左
(&&)	左
(  )	左
(=), (+=), (--), (=>)	右
(,)	右
(;)	左

### 12.4.3. EB GUIDE 脚本 标准库

以下章节说明了所有 EB GUIDE 脚本 函数。

#### 12.4.3.1. EB GUIDE 脚本 函数 A

##### 12.4.3.1.1. abs

此函数用于返回整数 **x** 的绝对值。

表 12.4. 参数 abs

参数	类型	说明
x	整数	返回其绝对值的数字
<return>	整数	返回值

##### 12.4.3.1.2. absf

此函数用于返回浮点数 **x** 的绝对值。



表 12.5. 参数 absf

参数	类型	说明
x	浮点值	返回其绝对值的数字
<return>	浮点值	返回值

**12.4.3.1.3. acosf**

此函数用于返回反余弦 **x** 的主值。

表 12.6. 参数 acosf

参数	类型	说明
x	浮点值	返回其反余弦的数字
<return>	浮点值	返回值

**12.4.3.1.4. animation\_before**

此函数用于检查放映的动画是否过了给定的时间点。

表 12.7. 参数 animation\_before

参数	类型	说明
animation	GtfTypeRecord	要操作的动画
time	整数	时间点
<return>	布尔值	如果为 true，则动画未过该时间点。

**12.4.3.1.5. animation\_beyond**

此函数用于检查放映的动画是否过了给定的时间点。

表 12.8. 参数 animation\_beyond

参数	类型	说明
animation	GtfTypeRecord	要操作的动画
time	整数	时间点
<return>	布尔值	如果为 true，则动画已经过了该时间点。



**12.4.3.1.6. animation\_cancel**

此函数用于取消动画并使已编辑的属性处于当前状态。

表 12.9. 参数 animation\_cancel

参数	类型	说明
animation	GtfTypeRecord	要操作的动画
<return>	布尔值	如果为 true，则表示此函数执行成功。

**12.4.3.1.7. animation\_cancel\_end**

此函数用于取消动画并将已编辑的属性设置为结束状态（如果可能）。

表 12.10. 参数 animation\_cancel\_end

参数	类型	说明
animation	GtfTypeRecord	要操作的动画
<return>	布尔值	如果为 true，则表示此函数执行成功。

**12.4.3.1.8. animation\_cancel\_reset**

此函数用于取消动画并将已编辑的属性重置为初始状态（如果可能）。

表 12.11. 参数 animation\_cancel\_reset

参数	类型	说明
animation	GtfTypeRecord	要操作的动画
<return>	布尔值	如果为 true，则表示此函数执行成功。

**12.4.3.1.9. animation\_pause**

此函数用于暂停动画。

表 12.12. 参数 animation\_pause

参数	类型	说明
animation	GtfTypeRecord	要操作的动画



参数	类型	说明
<return>	布尔值	如果为 true，则表示此函数执行成功。

**12.4.3.1.10. animation\_play**

此函数用于启动或继续播放动画。

表 12.13. 参数 animation\_play

参数	类型	说明
animation	GtfTypeRecord	要操作的动画
<return>	布尔值	如果为 true，则表示动画尚未播放。

**12.4.3.1.11. animation\_reverse**

此函数用于回放动画。

表 12.14. 参数 animation\_reverse

参数	类型	说明
animation	GtfTypeRecord	要操作的动画
<return>	布尔值	如果为 true，则表示动画尚未播放。

**12.4.3.1.12. animation\_running**

此函数用于检查动画当前是否正在播放。

表 12.15. 参数 animation\_running

参数	类型	说明
animation	GtfTypeRecord	要操作的动画
<return>	布尔值	如果为 true，则表示动画正在播放。

**12.4.3.1.13. animation\_set\_time**

此函数用于设置动画的当前时间，可用于跳过或重放动画。



表 12.16. 参数 animation\_set\_time

参数	类型	说明
animation	GtfTypeRecord	要操作的动画
time	整数	时间
<return>	布尔值	如果为 true，则表示此函数执行成功。

**12.4.3.1.14. asinf**

此函数用于返回反正弦 **x** 的主值。

表 12.17. 参数 asinf

参数	类型	说明
x	浮点值	返回其反正弦的数字
<return>	浮点值	返回值

**12.4.3.1.15. atan2f**

通过使用两个参数的符号来确定结果的象限，此函数用于计算反正切 **y/x** 的主值。

表 12.18. 参数 atan2f

参数	类型	说明
y	浮点值	参数 <b>y</b>
x	浮点值	参数 <b>x</b>
<return>	浮点值	返回值

**12.4.3.1.16. atan2i**

通过使用两个参数的符号来确定结果的象限，此函数用于计算反正切 **y/x** 的主值。

表 12.19. 参数 atan2i

参数	类型	说明
y	整数	参数 <b>y</b>
x	整数	参数 <b>x</b>
<return>	浮点值	返回值



**12.4.3.1.17. atanf**

此函数用于返回反正切  $x$  的主值。

表 12.20. 参数 atanf

参数	类型	说明
x	浮点值	返回其反正切的数字
<return>	浮点值	返回值

**12.4.3.2. EB GUIDE 脚本 函数 C - H**

**12.4.3.2.1. ceil**

此函数用于返回不小于参数的最小整数值。

表 12.21. 参数 ceil

参数	类型	说明
value	浮点值	要舍入的值
<return>	整数	已舍入的值

**12.4.3.2.2. changeDynamicStateMachinePriority**

此函数用于更改动态状态机的优先级。

表 12.22. 参数 changeDynamicStateMachinePriority

参数	类型	说明
state		包含动态状态机列表的状态
sm	整数	动态状态机
priority	整数	动态状态机在列表中的优先级

**12.4.3.2.3. character2unicode**

此函数用于返回字符串中第一个字符的 **Unicode** 值。



表 12.23. 参数 character2unicode

参数	类型	说明
str	字符串	输入字符串
<return>	整数	以 <b>Unicode</b> 形式表示的字符  0（如果发生错误）

**12.4.3.2.4. clearAllDynamicStateMachines**

此函数用于从动态状态机列表中删除所有动态状态机。

表 12.24. 参数 clearAllDynamicStateMachines

参数	类型	说明
state		包含动态状态机列表的状态

**12.4.3.2.5. color2string**

此函数用于将颜色转换为 8 个十六进制值。

表 12.25. 参数 color2string

参数	类型	说明
value	颜色	要转换为字符串的颜色
<return>	字符串	格式化为十六进制数字字符串的颜色，以 # 为前缀

注意



格式化示例

返回的字符串的格式为 #RRGGBBAA，其中每两个数字表示每个颜色通道（即，红色、绿色、蓝色和 alpha）。

例如，不透明的纯红色将转换为 #ff0000ff，而半透明的纯绿色将转换为 #00ff007f。

**12.4.3.2.6. cosf**

此函数用于返回 x 的余弦，其中 x 以半径指定。

表 12.26. 参数 cosf

参数	类型	说明
x	浮点值	返回其余弦的数字





参数	类型	说明
<return>	浮点值	返回值

**12.4.3.2.7. deg2rad**

此函数用于将角度从度转换为半径。

表 12.27. 参数 deg2rad

参数	类型	说明
x	浮点值	从度转换为半径的角度
<return>	浮点值	返回值

**12.4.3.2.8. expf**

此函数用于返回值 e（自然对数的底）的 x 次幂。

表 12.28. 参数 expf

参数	类型	说明
x	浮点值	指数
<return>	浮点值	返回值

**12.4.3.2.9. float2string**

此函数用于将简单浮点值转换为字符串。

表 12.29. 参数 float2string

参数	类型	说明
value	浮点值	要转换为字符串的值
<return>	字符串	浮点值，格式化为字符串

**12.4.3.2.10. floor**

此函数用于返回不大于参数值的最大整数值。



表 12.30. 参数 floor

参数	类型	说明
value	浮点值	要舍入的值
<return>	整数	已舍入的值

**12.4.3.2.11. focusNext**

此函数用于强制焦点管理器将焦点传递给下一个可获得焦点的元素。

表 12.31. 参数 focusNext

参数	类型	说明
<return>	空	

**12.4.3.2.12. focusPrevious**

此函数用于强制焦点管理器将焦点返回给上一个可获得焦点的元素。

表 12.32. 参数 focusPrevious

参数	类型	说明
<return>	空	

**12.4.3.2.13. format\_float**

此函数用于设置浮点值的格式。

表 12.33. 参数 format\_float

参数	类型	说明
format	字符串	具有下列结构的字符串：  %[flags] [width] [.precision] 类型  ► 标记：一个或多个可选字符，用于控制输出对齐和符号、空格、前导零、小数点以及八进制前缀和十六进制前缀的输出。  ► 宽度：可选的十进制数字，用于指定输出的字符的最小数目。

参数	类型	说明
		<ul style="list-style-type: none"><li>▶ 精度：可选的十进制数字，用于指定有效数字的数目或者小数点字符后的数字数目。</li><li>▶ 类型：必需的转换说明符字符，用于确定关联的参数解释为字符、字符串、整数还是浮点数。</li></ul>
useDotAsDelimiter	布尔值	用于定义分隔符符号。  可能的值： <ul style="list-style-type: none"><li>▶ true：使用点作为分隔符。</li><li>▶ false：使用逗号作为分隔符。</li></ul>
value	浮点值	要格式化的数字

警告



遵守 **C++** 的 **printf** 规范

`format` 参数是根据 **C++** 的 **printf** 规范定义的。

使用不符合此规范的值可能会导致出现意外行为。

例如，`format_float` 的允许类型为 `f`、`a`、`g` 及 `e`，并且不允许出现 **1** 个以上的类型字符。

#### 12.4.3.2.14. `format_int`

此函数用于设置整数值的格式。

表 12.34. 参数 `format_int`

参数	类型	说明
<code>format</code>	字符串	具有下列结构的字符串：  <b>%[flags] [width] [.precision] 类型</b> <ul style="list-style-type: none"><li>▶ 标记：一个或多个可选字符，用于控制输出对齐和符号、空格、前导零、小数点以及八进制前缀和十六进制前缀的输出。</li><li>▶ 宽度：可选的十进制数字，用于指定输出的字符的最小数目。</li><li>▶ 精度：可选的十进制数字，用于指定打印的数字的最小数目。</li><li>▶ 类型：必需的转换说明符字符，用于确定关联的参数解释为字符、字符串、整数还是浮点数。</li></ul>
<code>value</code>	<code>int</code>	要格式化的数字

警告



遵守 C++ 的 **printf** 规范

format 参数是根据 C++ 的 **printf** 规范定义的。

使用不符合此规范的值可能会导致出现意外行为。

例如，format\_int 的允许类型为 d、i、o、x 及 u，并且不允许出现 1 个以上的类型字符。

#### 12.4.3.2.15. getLineCount

此函数返回控件中文本的行数。

表 12.35. 参数 getLineCount

参数	类型	说明
widget	控件	要对其求值的控件
<return>	整数	行数

#### 12.4.3.2.16. getTextHeight

此函数用于返回关于其字体资源的文本高度。

表 12.36. 参数 getTextHeight

参数	类型	说明
text	字符串	要对其求值的文本
font	字体	要对其求值的字体
<return>	整数	文本的高度  如果字体大小为 0 或负数，函数返回 0。

#### 12.4.3.2.17. getTextLength

此函数用于返回文本中的字符数。

表 12.37. 参数 getTextLength

参数	类型	说明
text	字符串	要对其求值的文本



参数	类型	说明
<return>	整数	文本中的字符数

注意



转义序列

EB GUIDE 脚本 不解析转义序列，如 `\n`，并对每个字符进行计数。比如，对于文本 `Label\n`，函数 `getLength` 返回 7。

#### 12.4.3.2.18. `getWidth`

此函数用于返回关于其字体资源的文本宽度。

表 12.38. 参数 `getWidth`

参数	类型	说明
<code>text</code>	字符串	要对其求值的文本
<code>font</code>	字体	要对其求值的字体
<return>	整数	文本的宽度  如果字体大小为 0 或负数，函数返回 0。

#### 12.4.3.2.19. `has_list_window`

此函数用于检查索引是否对类型列表的数据池项有效。对于窗口化列表，它还会检查索引是否至少位于一个窗口中。

表 12.39. 参数 `has_list_window`

参数	类型	说明
<code>itemId</code>	<code>dp_id</code>	类型列表的数据池项的 ID
<code>index</code>	整数	数据池项内的索引
<return>	布尔值	如果为 <code>true</code> ，则数据池项内的索引有效并且至少位于一个窗口中。

#### 12.4.3.2.20. `hsva2color`

此函数用于将 HSB/HSV 颜色转换为 GTF 颜色。



表 12.40. 参数 hsb2color

参数	类型	说明
hue	整数	颜色值以度为单位，介于 0 到 360 之间
saturation	整数	饱和度（以百分比为单位）
brightness	整数	亮度（以百分比为单位）
alpha	整数	alpha 值介于 0（完全透明）到 255（不透明）之间
<return>	颜色	应用了 alpha 值的最终 GTF 颜色

### 12.4.3.3. EB GUIDE 脚本 函数 I - R

#### 12.4.3.3.1. int2float

此函数用于返回已转换为浮点值的整数值。

表 12.41. 参数 int2float

参数	类型	说明
value	整数	要转换为浮点值的值
<return>	浮点值	整数值，已转换为浮点值

#### 12.4.3.3.2. int2string

此函数用于将简单整数转换为字符串。

表 12.42. 参数 int2string

参数	类型	说明
value	整数	要转换为字符串的值
<return>	字符串	转换为字符串的整数值以十进制计数法表示

#### 12.4.3.3.3. isDynamicStateMachineActive

此函数用于检查包含动态状态机列表的状态是否为活动的。



表 12.43. 参数 isDynamicStateMachineActive

参数	类型	说明
state		包含动态状态机列表的状态
sm	整数	动态状态机

**12.4.3.3.4. language**

此函数用于切换所有数据池项的语言。该操作以异步方式执行。

表 12.44. 参数 language

参数	类型	说明
language	languageType	要切换为的语言，例如 f:language(l:German)
<return>	空	

**12.4.3.3.5. localtime\_day**

此函数用于从系统时间值中提取本地时间日期 [1:31]。

表 12.45. 参数 localtime\_day

参数	类型	说明
time	整数	系统时间返回的时间戳
<return>	整数	提取的日期

**12.4.3.3.6. localtime\_hour**

此函数用于从系统时间值的本地时间中提取小时数。

表 12.46. 参数 localtime\_hour

参数	类型	说明
time	整数	系统时间返回的时间戳
<return>	整数	提取的小时数



**12.4.3.3.7. localtime\_minute**

此函数用于从系统时间值的本地时间中提取分钟数。

表 12.47. 参数 localtime\_minute

参数	类型	说明
time	整数	系统时间返回的时间戳
<return>	整数	提取的分钟数

**12.4.3.3.8. localtime\_month**

此函数用于从系统时间值的本地时间中提取月份 [0:11]。

表 12.48. 参数 localtime\_month

参数	类型	说明
time	整数	系统时间返回的时间戳
<return>	整数	提取的月份

**12.4.3.3.9. localtime\_second**

此函数用于从系统时间值的本地时间中提取秒数。

表 12.49. 参数 localtime\_second

参数	类型	说明
time	整数	系统时间返回的时间戳
<return>	整数	提取的秒数

**12.4.3.3.10. localtime\_weekday**

此函数用于从系统时间值的本地时间中提取工作日 [0:6]。0 表示星期日。

表 12.50. 参数 localtime\_weekday

参数	类型	说明
time	整数	系统时间返回的时间戳





参数	类型	说明
<return>	整数	提取的工作日

**12.4.3.3.11. localtime\_year**

此函数用于从系统时间值的本地时间中提取年份。

表 12.51. 参数 localtime\_year

参数	类型	说明
time	整数	系统时间返回的时间戳
<return>	整数	提取的年份

**12.4.3.3.12. log10f**

此函数用于返回以 10 为底的 x 的对数。

表 12.52. 参数 log10f

参数	类型	说明
x	浮点值	参数
<return>	浮点值	返回值

**12.4.3.3.13. logf**

此函数用于返回 x 的自然对数。

表 12.53. 参数 logf

参数	类型	说明
x	浮点值	参数
<return>	浮点值	返回值

**12.4.3.3.14. nearbyint**

此函数用于将值舍入为最接近的整数。



表 12.54. 参数 nearbyint

参数	类型	说明
value	浮点值	要舍入的值
<return>	整数	已舍入的值

**12.4.3.3.15. popDynamicStateMachine**

此函数用于删除优先级队列顶部的动态状态机。

表 12.55. 参数 popDynamicStateMachine

参数	类型	说明
state		包含动态状态机列表的状态
sm	整数	动态状态机

**12.4.3.3.16. powf**

此函数用于返回值  $x$  的  $y$  次幂。

表 12.56. 参数 powf

参数	类型	说明
x	浮点值	参数 $x$
y	浮点值	参数 $y$
<return>	浮点值	返回值

**12.4.3.3.17. pushDynamicStateMachine**

此函数用于在优先级队列中插入动态状态机。

表 12.57. 参数 pushDynamicStateMachine

参数	类型	说明
state		包含动态状态机列表的状态
sm	整数	动态状态机



参数	类型	说明
priority	整数	动态状态机在列表中的优先级

**12.4.3.3.18. rad2deg**

此函数用于将角度从半径转换为度。

表 12.58. 参数 rad2deg

参数	类型	说明
x	浮点值	参数
<return>	浮点值	返回值

**12.4.3.3.19. rand**

此函数用于获取介于 0 到  $2^{31}-1$  之间的随机值。

表 12.59. 参数 rand

参数	类型	说明
<return>	整数	介于 0 到 $2^{31}-1$ 之间的随机数

**12.4.3.3.20. shutdown**

该函数请求框架关闭程序。

**12.4.3.3.21. rgba2color**

此函数用于将 RGB 色域转换为 GTF 颜色。

表 12.60. 参数 rgba2color

参数	类型	说明
red	整数	红色坐标，介于 0 到 255 之间
green	整数	绿色坐标，介于 0 到 255 之间



参数	类型	说明
blue	整数	蓝色坐标，介于 0 到 255 之间
alpha	整数	alpha 值介于 0（完全透明）到 255（不透明）之间
<return>	颜色	颜色从 RGB 色域转换为 GTF 颜色，并应用了 alpha 值

**12.4.3.3.22. round**

此函数用于将值舍入为最接近的整数，但对于半数情况，舍入方向远离零。

表 12.61. 参数 round

参数	类型	说明
value	浮点值	要舍入的值
<return>	整数	已舍入的值

**12.4.3.4. EB GUIDE 脚本 函数 S - W**

**12.4.3.4.1. seed\_rand**

此函数用于设置随机数字生成器的种子。

表 12.62. 参数 seed\_rand

参数	类型	说明
seed	整数	此函数用于设置随机数字生成器的种子
<return>	空	

**12.4.3.4.2. sinf**

此函数用于返回 x 的正弦，其中 x 以半径指定。

表 12.63. 参数 sinf

参数	类型	说明
x	浮点值	参数



参数	类型	说明
<return>	浮点值	返回值

**12.4.3.4.3. skin**

此函数切换所有数据池项的外观。该操作以异步方式执行。

表 12.64. 参数 skin

参数	类型	说明
skin	skinType	要切换为的外观，例如 f:skin(s:Standard)
<return>	空	

**12.4.3.4.4. sqrtf**

此函数用于返回 **x** 的非负平方根。

表 12.65. 参数 sqrtf

参数	类型	说明
x	浮点值	参数
<return>	浮点值	返回值

**12.4.3.4.5. string2float**

此函数用于将字符串初始组成部分转换为浮点。

字符串初始组成部分的期望格式如下所示：

1. 可选前导空格
2. 可选加号 ('+') 或减号 ('-')
3. 下列其中一项：
  - ▶ 十进制数
  - ▶ 十六进制数
  - ▶ 无穷



► NAN（非数字）

表 12.66. 参数 string2float

参数	类型	说明
str	字符串	字符串值
<return>	浮点值	返回值

**12.4.3.4.6. string2int**

此函数用于将字符串初始组成部分转换为整数。如果输入超出了范围，则会将结果缩减至介于 **2147483647** 到 **-2147483648** 之间的范围。如果字符串不以数字开头，则此函数将返回 0。

表 12.67. 参数 string2int

参数	类型	说明
str	字符串	字符串值
<return>	整数	返回值

**12.4.3.4.7. string2string**

此函数用于对字符串进行格式化。

表 12.68. 参数 string2string

参数	类型	说明
str	字符串	要格式化的字符串
len	整数	字符串的最大长度
<return>	字符串	语言字符串

**12.4.3.4.8. substring**

此函数用于创建字符串的子串副本。支持负结尾索引。

示例：

► `substring("abc", 0, -1)` 将返回 “abc”。



► `substring("abc", 0, -2)` 将返回 “ab”。

表 12.69. 参数 `substring`

参数	类型	说明
<code>str</code>	字符串	输入字符串
<code>startIndex</code>	整数	结果字符串的第一个字符索引
<code>endIndex</code>	整数	不是结果组成部分的第一个字符索引
<code>&lt;return&gt;</code>	字符串	语言字符串

**12.4.3.4.9. `system_time`**

此函数用于获取以秒为单位的当前系统时间。结果将传递给 `localtime_*` 函数。

表 12.70. 参数 `system_time`

参数	类型	说明
<code>&lt;return&gt;</code>	整数	以秒为单位的系统时间

**12.4.3.4.10. `system_time_ms`**

此函数用于获取以毫秒为单位的当前系统时间。

表 12.71. 参数 `system_time_ms`

参数	类型	说明
<code>&lt;return&gt;</code>	整数	以毫秒为单位的系统时间

**12.4.3.4.11. `tanf`**

此函数用于返回 `x` 的正切，其中 `x` 以半径指定。

表 12.72. 参数 `tanf`

参数	类型	说明
<code>x</code>	浮点值	参数



参数	类型	说明
<return>	浮点值	返回值

**12.4.3.4.12. trace\_dp**

此函数用于将数据池项的相关调试信息写入跟踪日志和连接日志。

表 12.73. 参数 trace\_dp

参数	类型	说明
itemId	dp_id	要跟踪器调试信息的项的数据池 ID
<return>	空	

**12.4.3.4.13. trace\_string**

此函数用于将字符串写入跟踪日志和连接日志。

表 12.74. 参数 trace\_string

参数	类型	说明
str	字符串	要跟踪的文本
<return>	空	

**12.4.3.4.14. transformToScreenX**

此函数用于获取某个控件和本地坐标，并返回屏幕相对世界坐标系中的 x 位置。

表 12.75. 参数 transformToScreenX

参数	类型	说明
widget	控件	坐标与之相对的控件
localX	整数	本地坐标的 x 位置
localY	整数	本地坐标的 y 位置
<return>	整数	屏幕坐标的 x 位置





**12.4.3.4.15. transformToScreenY**

此函数用于获取某个控件和本地坐标，并返回屏幕相对世界坐标系中某个位置的 **Y** 位置。

表 12.76. 参数 transformToScreenY

参数	类型	说明
widget	控件	坐标与之相对的控件
localX	整数	本地坐标的 <b>x</b> 位置
localY	整数	本地坐标的 <b>y</b> 位置
<return>	整数	屏幕坐标的 <b>y</b> 位置

**12.4.3.4.16. transformToWidgetX**

此函数用于获取某个控件以及提供给触摸动作反应的屏幕坐标，并返回控件相对本地坐标系中的 **x** 位置。

表 12.77. 参数 transformToWidgetX

参数	类型	说明
widget	控件	坐标与之相对的控件
screenX	整数	屏幕坐标的 <b>x</b> 位置
screenY	整数	屏幕坐标的 <b>y</b> 位置
<return>	整数	本地坐标的 <b>x</b> 位置

**12.4.3.4.17. transformToWidgetY**

此函数用于获取某个控件以及提供给触摸动作反应的屏幕坐标，并返回控件相对本地坐标系中的 **y** 位置。

表 12.78. 参数 transformToWidgetY

参数	类型	说明
widget	控件	坐标与之相对的控件
screenX	整数	屏幕坐标的 <b>x</b> 位置
screenY	整数	屏幕坐标的 <b>y</b> 位置
<return>	整数	本地坐标的 <b>y</b> 位置



**12.4.3.4.18. trunc**

此函数用于将值舍入为最接近的整数值，始终朝向零。

表 12.79. 参数 trunc

参数	类型	说明
value	浮点值	要舍入的值
<return>	整数	已舍入的值

**12.4.3.4.19. widgetGetChildCount**

此函数用于获取指定控件的子代控件数。

表 12.80. 参数 widgetGetChildCount

参数	类型	说明
widget	控件	要获取其子代控件数的控件
<return>	整数	子代控件数

## 12.5. 事件

表 12.81. 事件的属性

属性名称	说明
Name	事件的名称
Event ID	EB GUIDE TF 用于发送和接收事件的数字值
Event group	事件组的名称 事件组具有 EB GUIDE TF 用于发送和接收事件的 ID。

## 12.6. model.json 配置文件

model.json 是一个 EB GUIDE TF 配置文件，其中包含与单个 EB GUIDE 模型相关的配置项。

model.json 文件是导出的 EB GUIDE 模型的一部分。

下表用作所有默认配置参数的文档。

注意



### JSON 对象标记

如果您在 EB GUIDE Studio 中配置 model.json，请使用 JSON 对象标记。

例如，请参见[节12.6.1](#)。

有关 JSON 格式的详细信息，请参见 <http://www.json.org>。

表 12.82. 公共

配置项	类型	说明	默认值
gtf.eventsystem.maxQueue	整数	时间序列的最大尺寸	0
gtf.model.traces	布尔值	启用跟踪 f:trace_string 脚本函数	true
gtf.model.identifier	字符串	EB GUIDE 模型的唯一识别符（等同于 EB GUIDE Studio 项目 UUID）	空白
gtf.model.identifier.short	整数	EB GUIDE 模型的短识别符	0xdeadbeaf

表 12.83. 文件和路径

配置项	类型	说明	默认值
gtf.model.path	字符串	EB GUIDE 模型的路径	None
gtf.model.config	字符串	EB GUIDE 模型配置的完整路径	<gtf.model.path>/model.json
gtf.datapool.descriptionFile	字符串	数据池说明文件的名称	datapool.gtf
gtf.model.files.sm	字符串	状态机说明文件的名称	model.bin
gtf.model.files.rm	字符串	资源说明文件的名称	resources.bin
gtf.model.files.views	字符串	视图说明文件的名称	views.bin
gtf.model.files.types	字符串	类型说明文件的名称	types.bin
gtf.model.pluginstoload	字符串列表	要加载的 EB GUIDE 模型插件的名称	空白字符串列表
gtf.eventsystem.mapFile	字符串	事件系统映射文件的名称	eventMap.gtf

选项 `gtf.model.coreNames` 是一个包含所有配置核心名称的字符串列表。下表包含每个核心的配置项。

表 12.84. 核心

配置项	类型	说明	默认值
<code>gtf.model.cores.&lt;corename&gt;.ownThread</code>	布尔值	指定核心是否使用自己的运行阈值	false
<code>gtf.model.cores.&lt;corename&gt;.id</code>	整数	指定核心上下文标识符	0

选项 `gtf.model.sceneNames` 是一个包含所有配置场景名称的字符串列表。对于每个场景，可以找到下表中的配置项。

表 12.85. 场景

配置项	类型	说明	默认值
<code>gtf.model.scenes.&lt;scenename&gt;.visible</code>	布尔值	确定场景的可见性	true
<code>gtf.model.scenes.&lt;scenename&gt;.width</code>	整数	场景宽度	800
<code>gtf.model.scenes.&lt;scenename&gt;.height</code>	整数	场景高度	480
<code>gtf.model.scenes.&lt;scenename&gt;.x</code>	整数	场景的起点坐标	0
<code>gtf.model.scenes.&lt;scenename&gt;.y</code>	整数	场景的起点坐标	0
<code>gtf.model.scenes.&lt;scenename&gt;.projectName</code>	字符串	工作项目名称	
<code>gtf.model.scenes.&lt;scenename&gt;.-windowCaption</code>	字符串	显示窗口名称文本	
<code>gtf.model.scenes.&lt;scenename&gt;.sceneId</code>	整数	场景识别符	0
<code>gtf.model.scenes.&lt;scenename&gt;.maxFPS</code>	整数	重绘率（FPS = 每秒帧数）对于无限制的重绘率，设置为 0。	60
<code>gtf.model.scenes.&lt;scenename&gt;.hwLayerId</code>	整数	指定核心上下文标识符	0
<code>gtf.model.scenes.&lt;scenename&gt;.colorMode</code>	整数	指定色彩模式： <ul style="list-style-type: none"> <li>▶ 1: 32 位 (RGBA8888)</li> <li>▶ 2: 16 位 (RGB565)</li> <li>▶ 3: 24 位 (RGB888)</li> <li>▶ 4: 32 位 sRGB</li> <li>▶ 5: 32 位 sRGB（模拟）</li> </ul>	1

配置项	类型	说明	默认值
gtf.model.scenes.<scenename>.-multisampling	整数	指定场景的多重采样 <ul style="list-style-type: none"> <li>▶ 0: 无多点采样</li> <li>▶ 1: <b>2x</b> 多点采样</li> <li>▶ 2: <b>4x</b> 多点采样</li> </ul>	0
gtf.model.scenes.<scenename>.-enableRemoteFramebuffer	布尔值	如果为true, 则将启用离屏缓冲区到模拟窗口的转换。	false
gtf.model.scenes.<scenename>.-showWindowFrame	布尔值	确定是否应显示渲染器窗口框	true
gtf.model.scenes.<scenename>.showWindow	布尔值	如果为 true, 则将在基于 <b>Windows</b> 的系统中打开另一个窗口进行模拟。	true
gtf.model.scenes.<scenename>.disableVsync	布尔值	如果为 true, 则将禁用渲染器的垂直同步。	false
gtf.model.scenes.<scenename>.showFPS	整数	可能的值: <ul style="list-style-type: none"> <li>▶ 0: 不显示 <b>FPS</b></li> <li>▶ 1: 在屏幕上显示 <b>FPS</b></li> <li>▶ 2: 在控制台中显示 <b>FPS</b></li> <li>▶ 3: 在屏幕和控制台中显示 <b>FPS</b></li> </ul>	0
gtf.model.scenes.<scenename>.renderer	字符串	要使用的渲染器的名称: DirectXRenderer OpenGLRenderer 或 OpenGL3Renderer	

表 12.86. 渲染常用

配置项	类型	说明	默认值
gtf.model.fontCache.width	整数	字体缓存地图集纹理的宽度	512
gtf.model.fontCache.height	整数	字体缓存地图集纹理的高度	512

配置项	类型	说明	默认值
gtf.model.fontCache.age	整数	字体缓存刷新操作必须完成前允许的最大允许年限	100
gtf.model.traversalStackSize	整数	渲染器遍历堆栈大小（以字节为单位）	32768

下表中的配置项属于一起的。这意味着，渲染器预期相同数量的项在三个列表中。在一个列表中索引的条目，属于其他列表中具有相同索引的条目。

表 12.87. 渲染器显示扩展

配置项	类型	说明	默认值
gtf.model.displayId	整数列表	场景的标识符	
gtf.model.maxCacheSize	整数列表	用于场景的最大纹理缓存	
gtf.model.driverName	字符串列表	OS 场景的特定驱动程序名称，例如 /dev/fb0	

下表中的配置项用于配置 TextEngine 元素。TextEngine 基于 FreeType 第三方库。下列参数将传递给 FreeType 实现。有关 FreeType 的详细信息，请参见 [https://www.freetype.org/freetype2/docs/reference/ft2-cache\\_subsystem.html](https://www.freetype.org/freetype2/docs/reference/ft2-cache_subsystem.html)。

由于 EB GUIDE TF 处理字体大小的方式，ft\_size 对象不会与 ft\_face 对象分开缓存。需要考虑 max\_sizes 的值可能会受目标平台硬件的限制。

表 12.88. TextEngine 配置项

配置项	类型	说明	默认值
gtf.model.textengine.replacementGlyph	整数	应该在当前字体中未找到专用字体字符的情况下使用的 Unicode 字符	0xffffd
gtf.model.textengine.maxFaces	整数	缓存字体的最大数量	0
gtf.model.textengine.maxSizes	整数	缓存字体大小的最大量	0
gtf.model.textengine.maxBytes	整数	可用作缓存的内存的最大量（字节）	0
gtf.model.textengine.enablePlainFileStream	布尔值	确定字体访问配置。如果为 true，将使用普通文件 I/O 访问权限。如果为 false，将使用	false



配置项	类型	说明	默认值
		ROM 映射文件访问权限。	

注意



位图字体的配置项

对于 .fnt 位图字体，您只能使用 replacementGlyph 配置项。不能将[表12.88](#)中的其他配置项用于位图字体。

注意



**ROM** 映射文件方法与普通文件 **I/O** 方法

ROM 映射文件方法通常提供更高的性能。但在 QNX 等系统上，它比普通文件 I/O 方法消耗更多的内存。普通文件 I/O 方法通常比 ROM 映射文件方法消耗更少的内存。但它可能导致性能下降。

选项 `gtf.model.touchDevicesNames` 是一个字符串列表，其中包含所有配置的触摸设备的名称。对于每个触摸设备，可以使用下表中列出的配置项。

表 12.89. 触摸设备

配置项	类型	说明	默认值
<code>gtf.model.touchDevices.&lt;deviceName&gt;.touchscreenType</code>	整数	定义触摸设备类型： <ul style="list-style-type: none"><li>▶ 0: Galaxy</li><li>▶ 1: imx WVGA</li><li>▶ 2: Mouse</li><li>▶ 3: General</li><li>▶ 4: Lilliput_889GL</li><li>▶ 5: General Multitouch</li><li>▶ 6: Lilliput with automatic calibration</li><li>▶ 7: GenericTouch Configuration</li></ul>	3
<code>gtf.model.touchDevices.&lt;deviceName&gt;.displayManagerId</code>	整数	指定设备有效的场景 ID	0
<code>gtf.model.touchDevices.&lt;deviceName&gt;.touchId</code>	整数	指定设备的 ID	0



配置项	类型	说明	默认值
gtf.model.touchDevices.<deviceName>.-minimalDistanceToMove	整数	触摸位置变化反应的阈值	0
gtf.model.touchDevices.<deviceName>.-touchMoveRepeatTimeout	整数	触摸位置更改通知间的延迟	0
gtf.model.touchDevices.<deviceName>.width	整数	可触摸设备区域的宽度	0
gtf.model.touchDevices.<deviceName>.-height	整数	可触摸设备区域的高度	0
gtf.model.touchDevices.<deviceName>.x_high	整数	可触摸设备区域的最大水平分辨率	0
gtf.model.touchDevices.<deviceName>.y_high	整数	可触摸设备区域的最大垂直分辨率	0
gtf.model.touchDevices.<deviceName>.x_low	整数	可触摸设备区域的最小水平分辨率	0
gtf.model.touchDevices.<deviceName>.y_low	整数	可触摸设备区域的最小垂直分辨率	0
gtf.model.touchDevices.<deviceName>.-devicePath	字符串	触摸用驱动的名称，例如 /dev/input0	

### 12.6.1. 示例 model.json 位于 EB GUIDE Studio



**例 12.1.**  
model.json 中的 **EB GUIDE Studio**

```
{
  "gtf": {
    "datapool": {
      "descriptionFile": "datapool.gtf"
    },
    "eventsystem": {
      "maxQueue": 0,
      "mapFile": "eventMap.gtf"
    },
    "model": {
      "coreNames": [
        "<core_1>"
      ]
    }
  }
}
```



```
],
"cores":{
  "<core_1>":{
    "ownThread":false,
    "id":0
  }
},
"touchDevicesNames":[
  "<device_1>"
],
"touchDevices":{
  "<device_1>":{
    "touchscreenType":3,
    "displayManagerId":0,
    "touchId":0,
    "minimalDistanceToMove":0,
    "touchMoveRepeatTimeout":0,
    "width":0,
    "height":0,
    "x_high":0,
    "y_high":0,
    "x_low":0,
    "y_low":0,
    "devicePath":""
  }
},
"displayId":[

],
"driverName":[

],
"fontCache":{
  "width":512,
  "height":512,
  "age":100
},
"maxCacheSize":[

],
"sceneNames":[
  "<scene_1>"
],
"scenes":{
  "<scene_1>":{
    "visible":true,
    "width":800,
```

```
    "height":480,
    "x":0,
    "y":0,
    "projectName":"<project_x>",
    "windowCaption":"<Displayed window name text>",
    "sceneId":0,
    "maxFPS":60,
    "hwLayerId":0,
    "colorMode":1,
    "multisampling":0,
    "enableRemoteFramebuffer":false,
    "showWindowFrame":true,
    "showWindow":true,
    "disableVsync":false,
    "showFPS":0,
    "renderer":"DirectXRenderer"
  }
},
"traces":true,
"traversalStackSize":32768,
"identifier":"",
"path":"<binary_folder>",
"config":"<gtf.model.path>/model.json",
"files":{
  "sm":"model.bin",
  "rm":"resources.bin",
  "views":"views.bin",
  "types":"types.bin"
},
"pluginstoload":[
]
}
}
```

## 12.7. platform.json 配置文件

platform.json 是一个 **EB GUIDE TF** 配置文件，它包含常见的和依赖平台的项目。

platform.json 文件是导出的 **EB GUIDE** 模型的一部分。

下表用作所有默认配置参数的文档。

注意



### JSON 对象标记

如果您在 EB GUIDE Studio 中配置 platform.json，请使用 JSON 对象标记。

例如，请参见[节12.7.1](#)。

有关 JSON 格式的详细信息，请参见<http://www.json.org>。

表 12.90. 平台配置

配置项	类型	说明	默认值
gtf.servicemapper.port	整数	服务接口（比如，EB GUIDE Monitor）	60000
gtf.core.pluginstoload	字符串列表	应加载的核心插件列表（相对于二进制文件夹或绝对路径）	None
gtf.launcher.editmode	布尔值	定义 EB GUIDE TF 是否在 EB GUIDE Studio 中运行。这是一个只读项。	false
gtf.platform.config	字符串	platform.json 文件的完整路径。这是一个只读项。	<binary_folder>/platform.json
gtf.framework.path	字符串	GtfStartup 可执行文件的路径。这是一个只读项。	<binary_folder>
gtf.diagnostic.memory.interval	整数	指定内存诊断的时间间隔。如果值为 0，则禁用诊断。	0
gtf.ipc.role	字符串	IPC 节点的作用可能值为 server 或者 client	server
gtf.ipc.discovery.network	字符串	将用于服务器 - 客户端发现机制的 IPv4 网络地址。在直接连接的情况下，这表示服务器的网络地址。	255.255.255.255
gtf.ipc.discovery.port	整数	将用于服务器 - 客户端发现机制的网络端口。在直接连接的情况下，必须等于服	4711

配置项	类型	说明	默认值
		务器配置中的 gtf.- servicemapper.port 项。	
gtf.ipc.datapool.config	字符串	包含应作为 IPC 通信组 成部分的数据池项的配 置文件	ipc_datapool.gtf
gtf.ipc.discovery.mode	字符串	用于连接服务器和客户 端的发现模式。可能 选项为：“broadcast” 、“multicast”和 “direct”。	broadcast
gtf.ipc.client.timeout	整数	客户端连接到服务器的 重试周期，以毫秒为单 位。	5000

### 12.7.1. 示例 platform.json 位于 EB GUIDE Studio



#### 例 12.2.

platform.json 中的 **EB GUIDE Studio**

```
{
  "gtf": {
    "core": {
      "pluginstoload": [
        "TfRuntime",
        "TfService",
        "TfGui",
        "TfGUIOpenGLS20",
        "TfGUIOpenGLS3",
        "TfGUIDirectX11"
      ]
    },
    "servicemapper": {
      "port": 60000
    },
    "launcher": {
      "editmode": true
    }
  },
}
```



```
"platform":{
  "config":"<binary_folder>/platform.json"
},
"framework":{
  "path":"<binary_folder>"
},
"diagnostic":{
  "memory":{
    "interval":0
  }
},
"ipc":{
  "role":"server",
  "discovery":{
    "network":"255.255.255.255",
    "port":4711,
    "mode":"broadcast"
  },
  "client":{
    "timeout":5000
  },
  "datapool": {
    "config": "ipc_datapool.gtf"
  }
}
}
```

## 12.8. 场景

表 12.91. 场景属性

属性名称	说明
height	触觉状态机的视图在其中于目标设备上渲染的区域的高度
width	触觉状态机的视图在其中于目标设备上渲染的区域宽度
x	触觉状态机的视图在其中于目标设备上渲染的区域 x 偏移量
y	触觉状态机的视图在其中于目标设备上渲染的区域 y 偏移量
visible	如果为 true，则状态机及其子代控件可见。
projectName	EB GUIDE 项目的名称

属性名称	说明
windowCaption	显示在窗口框架中的文本
sceneID	可以用于某种用途（例如，用于输入处理）的唯一场景识别符
maxFPS	重绘率（FPS = 每秒帧数）  对于无限制的重绘率，设置为 0。
hwLayerID	目标设备显示屏上硬件图层的 ID，映射到当前状态机
colorMode	可能的值：  <ul style="list-style-type: none"> <li>▶ 32-bit (=1): <b>RGBA8888</b></li> <li>▶ 16-bit (=2): <b>RGB565</b></li> <li>▶ 24-bit (=3): <b>RGB888</b></li> <li>▶ 32-bit sRGB (=4):</li> </ul> <p>此值使用 <b>GPU</b> 硬件支持。</p> <p>如果您希望获得对图像控件或对“弥散纹理”控件功能的 <b>sRGB</b> 支持，请使用此值。</p> <ul style="list-style-type: none"> <li>▶ 32-bit sRGB (Emulated) (=5):</li> </ul> <p>仅当 32-bit sRGB 未生成正确结果时，才使用此值。</p>
multisampling	可能的值：  <ul style="list-style-type: none"> <li>▶ Off (= 0): 无多点采样</li> <li>▶ 2x (=1): <b>2x</b> 多点采样</li> <li>▶ 4x (=2): <b>4x</b> 多点采样</li> </ul> <p>另请参见<a href="#">多点采样设置</a>。</p>
enableRemoteFramebuffer	如果为 true，则将启用离屏缓冲区到模拟窗口的转换
showWindowFrame	如果为 true，则模拟窗口中将显示某个帧。此帧允许抓取和移动窗口。
showWindow	如果为 true，则将在基于 <b>Windows</b> 的系统中打开另一个窗口进行模拟。
disableVSync	如果为 true，则将禁用渲染器的垂直同步。
showFPS	可能的值：  <ul style="list-style-type: none"> <li>▶ Off (=0): 不显示 <b>FPS</b></li> <li>▶ On screen (=1): 在屏幕上显示 <b>FPS</b></li> </ul>

属性名称	说明
	<div><div>▶ Console (=2)：在控制台中显示 FPS</div><div>▶ Console &amp; on screen (=3)：在屏幕和控制台中显示 FPS</div></div>
Renderer	<div>定义场景的渲染器。</div> <div>可能的值：</div> <div><div>▶ DirectXRenderer</div><div>▶ OpenGLRenderer</div><div>▶ OpenGL3Renderer</div></div>

提示



**多点采样设置**

多点采样的地图分辨率越高，渲染结果质量越好。然而请注意，多点采样减弱渲染性能，特别是在目标设备上。多点采样对高分辨率的小显示几乎不起作用。

起初不采用多点采样，如果性能良好，尝试设置两倍或四倍多点采样。如果高倍多点采样影响不大，则使用较低的设置。

提示



**多点采样设置依赖于硬件**

如果无法从硬件侧获取所需的多点采样设置，可以从日志文件中获取相关信息。

## 12.9. EB GUIDE GTF 支持的触摸屏类型

受支持类型取决于目标设备。

表 12.92. 支持的触摸屏类型 EB GUIDE GTF

值	说明	平台
0	Galaxy	Linux
1	IMX WVGA	Linux
2	连接到鼠标接口的触摸屏	全部
3	依赖于平台的常规触摸屏接口	全部
4	Lilliput 889GL	QNX
5	依赖于平台的常规多点触控触摸屏接口	Linux

## 12.10. 控件

### 12.10.1. 视图

表 12.93. 视图的属性

属性名称	说明
name	控件的名称
height	控件的高度（以像素为单位）
width	控件的宽度（以像素为单位）
visible	如果为 true，则控件及其子代控件可见
x	控件的 <b>x</b> 坐标
y	控件的 <b>y</b> 坐标

视图模板有视图切换动画的附加属性。进入视图时执行入口动画。

表 12.94. 入口动画的属性

属性名称	说明
Entry animation	若为真，视图模板的实例将有入口动画。
Transition type	进入动画的类型，例如“从左侧移入”、“从中心淡入”或“立即显示视图”。
Duration	入口动画的持续时间（以毫秒为单位）。
Delay	入口动画的延迟（以毫秒为单位）。
Play after exit animation	若为真，入口动画的开始时间取决于上一个退出动画的持续时间。

退出视图时执行退出动画。

表 12.95. 退出动画的属性

属性名称	说明
Exit animation	若为真，视图模板的实例将有退出动画。
Transition type	退出动画的类型，例如“移出到顶部”、“淡出到中央”或“立即隐藏视图”。
Duration	退出动画的持续时间（以毫秒为单位）。
Delay	退出动画的延迟（以毫秒为单位）。

### 12.10.2. 基本控件



有八个基本控件。

- ▶ Alpha 掩码
- ▶ 动画
- ▶ 容器
- ▶ 椭圆
- ▶ 图像
- ▶ 实例化器
- ▶ 标签
- ▶ 矩形

以下各个部分列出了基本控件的属性。

注意



唯一名称  
对具有同一父代控件的两个控件使用唯一名称。

注意



负值  
请勿将负值用于 height 和 width 属性。EB GUIDE Studio 将负值视为 0，这意味着不会描述相关控件。

### 12.10.2.1. Alpha 掩码

alpha 掩码是一个容器控件，用于通过图像控制其子代控件的 alpha 通道（即，不透明度）。

表 12.96. alpha 掩码的属性

属性名称	说明
visible	如果为 true，则控件及其子代控件可见
width	控件的宽度（以像素为单位）
height	控件的高度（以像素为单位）
x	控件相对于其父代控件的 x 坐标
y	控件相对于其父代控件的 y 坐标
enabled	如果为 true，alpha 掩码将应用于子代控件
image	用于控制 alpha 通道（即，子代控件的不透明度）的图像



属性名称	说明
horizontalAlign	标签边界内图像文件的水平对齐方式
verticalAlign	标签边界内图像文件的垂直对齐方式
scaleMode	图片的缩放模式。可能的值： <ul style="list-style-type: none"> <li>▶ original size (=0)</li> <li>▶ fit to size (=1)</li> <li>▶ keep aspect ratio (=2)</li> </ul>

注意



对于 **alpha** 掩码，支持的图像文件类型  
 可用的图像格式取决于渲染器的实现。DirectX 11 和 OpenGL ES 版本 2.0 或更高版本支持 .png 文件和 .jpg 文件。RGB 图像会在 用作 **alpha** 掩码之前转换为灰度图像。灰度图像将按原样使用。将忽略图像中的 **alpha** 通道。

您无法将 **alpha** 掩码与 9-patch 图像一起使用。

### 12.10.2.2. 动画

动画用于定义控件沿着某个视图的移动。要定义动画的外观，请在“动画”编辑器中添加曲线。

表 12.97. 动画的属性

属性名称	说明
enabled	定义是否播放动画
repeat	重复次数，0 表示无穷大数
alternating	如果为 true，将反复/双向运行动画。  如果为 false，将沿一个方向/单向运行动画。  重复次数在 repeat 属性中定义。
scale	动画时间的倍增因子
onPlay	动画开始或继续播放时执行的动作反应。参数：开始时间和播放方向（true 表示播放，false 表示回放）
onPause	动画暂停时执行的动作反应。参数：当前动画时间。
onTerminate	动画播放完成时执行的动作反应。第一个参数：动画时间。第二个参数：终止原因，按以下方式编码： <ul style="list-style-type: none"> <li>▶ 0: 动画播放完成</li> <li>▶ 1: 动画已取消，由……触发 f:animation_cancel</li> </ul>



属性名称	说明
	<ul style="list-style-type: none"> <li>▶ 2: 控件由于视图转化而损坏</li> <li>▶ 3: 动画跳至最后一步, 由……触发 f:animation_cancel_end</li> <li>▶ 4: 动画跳至其第一步, 然后取消播放, 由……触发 f:animation_cancel_reset</li> </ul>

### 12.10.2.2.1. 常量曲线

常量曲线用于根据已定义的延迟设置目标值。常量曲线可用于整数、布尔值、浮点和颜色类型。

表 12.98. 常量曲线的属性

属性名称	说明
enabled	定义是否播放动画
delay	相对于动画开始时间的延迟时间（以 ms 为单位）
duration	曲线段的持续时间（以 ms 为单位）
repeat	重复次数, 0 表示重复次数不受限制
alternating	定义是否重复播放动画
relative	定义是否将更新值应用于初始值
value	生成的常量值
target	向其分配生成的值的目标属性

### 12.10.2.2.2. 快速启动曲线

快速启动曲线会周期性地设置一个值, 该值在开始阶段快速增加, 但增加速度不断降低, 直到结束。快速启动曲线可用于整数、浮点和颜色类型。

表 12.99. 快速启动曲线的属性

属性名称	说明
enabled	定义是否播放动画
delay	相对于动画开始时间的延迟时间（以 ms 为单位）
duration	曲线段的持续时间（以 ms 为单位）
repeat	重复次数, 0 表示重复次数不受限制
alternating	定义是否重复播放动画
relative	定义是否将更新值应用于初始值

属性名称	说明
start	初始值
end	最终值
target	向其分配生成的值的目标属性

### 12.10.2.2.3. 慢速启动曲线

慢速启动曲线会周期性地设置一个值，该值在开始阶段缓慢增加，但增加速度不断上升，直到结束。慢速启动曲线可用于整数、浮点和颜色类型。

表 12.100. 慢速启动曲线的属性

属性名称	说明
enabled	定义是否播放动画
delay	相对于动画开始时间的延迟时间（以 ms 为单位）
duration	曲线段的持续时间（以 ms 为单位）
repeat	重复次数，0 表示重复次数不受限制
alternating	定义是否重复播放动画
relative	定义是否将更新值应用于初始值
start	初始值
end	最终值
target	向其分配生成的值的目标属性

### 12.10.2.2.4. 二次曲线

二次曲线会使用二次函数曲线来周期性地设置一个值。二次曲线可用于整数、浮点和颜色类型。

表 12.101. 二次曲线的属性

属性名称	说明
enabled	定义是否播放动画
delay	相对于动画开始时间的延迟时间（以 ms 为单位）
duration	曲线段的持续时间（以 ms 为单位）
repeat	重复次数，0 表示重复次数不受限制
alternating	定义是否重复播放动画
relative	定义是否将更新值应用于初始值



属性名称	说明
acceleration	曲线的加速度
velocity	计算结果的速度
constant	用于计算结果的常量值
target	向其分配生成的值的目标属性

#### 12.10.2.2.5. 正弦曲线

正弦曲线会使用正弦函数曲线来周期性地设置一个值。正弦曲线可用于整数、浮点和颜色类型。

表 12.102. 正弦曲线的属性

属性名称	说明
enabled	定义是否播放动画
delay	相对于动画开始时间的延迟时间（以 ms 为单位）
duration	曲线段的持续时间（以 ms 为单位）
repeat	重复次数，0 表示重复次数不受限制
alternating	定义是否重复播放动画
relative	定义是否将更新值应用于初始值
amplitude	正弦曲线的振幅
constant	用于计算结果的常量值
frequency	曲线的频率（以 Hz 为单位）
phase	角度相变（以弧度为单位）
target	向其分配生成的值的目标属性

#### 12.10.2.2.6. 脚本曲线

脚本曲线会使用 EB GUIDE 脚本 所说明的曲线设置一个值。脚本曲线可用于整数、布尔值、浮点和颜色类型。

表 12.103. 脚本曲线的属性

属性名称	说明
enabled	定义是否播放动画
delay	相对于动画开始时间的延迟时间（以 ms 为单位）
duration	曲线段的持续时间（以 ms 为单位）



属性名称	说明
repeat	重复次数，0 表示重复次数不受限制
alternating	定义是否重复播放动画
relative	定义是否将更新值应用于初始值
curve	生成的曲线函数
target	向其分配生成的值的目标属性

### 12.10.2.2.7. 线性曲线

线性曲线会使用线性增长曲线周期性地设置一个值。线性曲线可用于整数、浮点和颜色类型。

表 12.104. 线性曲线的属性

属性名称	说明
enabled	定义是否播放动画
delay	相对于动画开始时间的延迟时间（以 ms 为单位）
duration	曲线段的持续时间（以 ms 为单位）
repeat	重复次数，0 表示重复次数不受限制
alternating	定义是否重复播放动画
relative	定义是否将更新值应用于初始值
velocity	计算结果的速度
constant	用于计算结果的常量值
target	向其分配生成的值的目标属性

### 12.10.2.2.8. 线性插值曲线

线性插值曲线控件会使用线性插值曲线来周期性地设置一个值。线性插值曲线可用于整数、浮点和颜色类型。

注意



#### 线性键值插值曲线

在导入 3D 图形文件时，如导入的 3D 场景有动画，则创建线性键值插值整数曲线和线性键值插值浮点曲线。在 EB GUIDE Studio 中，这些曲线的底层键值对无法修改。

表 12.105. 线性插值曲线的属性

属性名称	说明
enabled	定义是否播放动画



属性名称	说明
delay	相对于动画开始时间的延迟时间（以 ms 为单位）
duration	曲线段的持续时间（以 ms 为单位）
repeat	重复次数，0 表示重复次数不受限制
alternating	定义是否重复播放动画
relative	定义是否将更新值应用于初始值
target	向其分配生成的值的目标属性

### 12.10.2.3. 容器

容器用于将多个控件保存为子代控件，从而对控件进行分组。

表 12.106. 容器的属性

属性名称	说明
height	控件的高度（以像素为单位）
width	控件的宽度（以像素为单位）
visible	如果为 true，则控件及其子代控件可见
x	控件相对于其父代控件的 x 坐标
y	控件相对于其父代控件的 y 坐标

### 12.10.2.4. 椭圆

在视图中，椭圆使用该控件的尺寸和坐标来绘制彩色的椭圆。控件也可用于绘制扇形或弧形。

表 12.107. 椭圆的属性

属性名称	说明
height	控件的高度（以像素为单位）
width	控件的宽度（以像素为单位）
visible	如果为 true，则控件及其子代控件可见
x	控件相对于其父代控件的 x 坐标
y	控件相对于其父代控件的 y 坐标
fillColor	用于填充椭圆的颜色
arcWidth	椭圆中弧形的宽度

属性名称	说明
centralAngle	定义椭圆的扇形角度（以度为单位）
sectorRotation	描述椭圆扇形旋转角度（以度为单位）

### 12.10.2.5. 图像

图像用于将图片放入视图中。

表 12.108. 图像的属性

属性名称	说明
height	控件的高度（以像素为单位）
width	控件的宽度（以像素为单位）
visible	如果为 true，则控件及其子代控件可见
x	控件相对于其父代控件的 x 坐标
y	控件相对于其父代控件的 y 坐标
image	控件显示的图像
sRGB	如果启用了此属性，将使用 sRGB 色域渲染 image 中选择的图像。  请注意，要使用 sRGB 功能，请在项目中心的“配置” > “配置文件”下，为 colorMode 属性选择 32-bit sRGB (=4) 或 32-bit sRGB (Emulated) (=5)。
horizontalAlign	标签边界内图像文件的水平对齐方式
verticalAlign	标签边界内图像文件的垂直对齐方式

注意



受支持的图像文件类型  
可用的图像格式取决于渲染器的实现。DirectX 11OpenGL ES 版本 2.0 或更高版本支持 .png 文件和 .jpg 文件。

### 12.10.2.6. 实例化器

实例化器用于在运行时间创建控件实例。您可以用实例化器对含动态或静态内容的列表或表格进行建模。实例化器的子代控件用作运行时间创建的列表或表的行模板。默认情况下，实例化程序仅实例化第一行模板。

表 12.109. 实例化器的属性

属性名称	说明
height	控件的高度（以像素为单位）





属性名称	说明
width	控件的宽度（以像素为单位）
visible	如果为 true，则控件及其子代控件可见
x	控件相对于其父代控件的 x 坐标
y	控件相对于其父代控件的 y 坐标
numItems	实例化的子代控件数。如果 numItems 是 0, 不创建子控件。
lineMapping	定义哪个子控件是哪行的行模板，即定义实例化的顺序

### 12.10.2.7. 标签

标签用于将文本放入视图中。

表 12.110. 标签的属性

属性名称	说明
height	控件的高度（以像素为单位）
width	控件的宽度（以像素为单位）
visible	如果为 true，则控件及其子代控件可见
x	控件相对于其父代控件的 x 坐标
y	控件相对于其父代控件的 y 坐标
text	标签所显示的文本。如果文本不适合控件区域，默认将其末端切断。
textColor	文本的显示颜色
font	文本的显示字体
horizontalAlign	标签边界内文本的水平对齐方式
verticalAlign	标签边界内文本的垂直对齐方式

### 12.10.2.8. 矩形

矩形用于在视图中绘制使用该控件的尺寸和坐标绘制彩色矩形。

表 12.111. 矩形的属性

属性名称	说明
height	控件的高度（以像素为单位）
width	控件的宽度（以像素为单位）



属性名称	说明
visible	如果为 true，则控件及其子代控件可见
x	控件相对于其父代控件的 x 坐标
y	控件相对于其父代控件的 y 坐标
fillColor	用于填充矩形的颜色

### 12.10.3. 3D 控件

#### 12.10.3.1. 环境光

环境光是一种均匀地照亮场景的光。环境光会影响材料控件的 ambient 颜色属性。

表 12.112. 环境光的属性

属性名称	说明
enabled	如果为 true，则启用控件
color	光的颜色
intensity	光的强度，0.0 表示无环境光

#### 12.10.3.2. 摄像头

摄像头从特定的视点定义场景的视图。使用多个摄像头从不同的视点显示场景。

表 12.113. 摄像头的属性

属性名称	说明
enabled	如果为 true，则启用控件
nearPlane	视图方向上摄像头的最近距离，在该距离处场景变得可见
farPlane	视图方向上摄像头的最远距离，可以看到场景
fieldOfView	摄像头的垂直视角（度）
projectionType	定义摄像头的投影类型。物体使用 perspective (=0) 或 orthographic (=1) 投影进行渲染。  注意：如果投影类型为正交，将使用 fieldOfView 角度计算视景体。



### 12.10.3.3. 定向光

从一个方向照亮场景的定向光。

表 12.114. 定向光的属性

属性名称	说明
enabled	如果为 true，则启用控件
color	光的颜色
intensity	光的强度，0.0 表示无定向光

### 12.10.3.4. 材料

材料使用 Phong 反射模型定义网格表面的视觉外观。

表 12.115. 材料的属性

属性名称	说明
ambient	物体被环境光照亮时反射的颜色。如果未向父场景图中添加环境光，此属性不起作用。
diffuse	物体被纯白光照亮时在所有方向上均匀反射的颜色
emissive	物体的自照明颜色
shininess	光亮因素
specular	具有光泽表面的物体反射的颜色
opacity	不透明值  请注意，仅有 0 和 1 之间的值，例如 0.3，是有效的。

### 12.10.3.5. 网格

网格定义 3D 对象的形状。

表 12.116. 网格的属性

属性名称	说明
visible	如果为 true，则控件及其子代控件可见
mesh	自动创建的网格文件 *.ebmesh
culling	定义是否从网格中挑选非三角形 (0)、仅正面三角形 (1) 或仅背面三角形 (2)

### 12.10.3.6. PBR GGX 材料

PBR GGX 材料使用物理上正确的 Cook-Torrance 模型定义网格表面的视觉外观。

表 12.117. PBR GGX 材料的属性

属性名称	说明
ambient	物体被环境光照亮时反射的颜色
diffuse	物体被纯白光照亮时在所有方向上均匀反射的颜色
emissive	物体的自照明颜色
specular	具有光泽表面的物体反射的颜色
metallic	金属表面性质的值  此值在漫射和镜面反射之间插入。  请注意，仅介于 0 与 1 之间的值有效，例如 0.3。
roughness	粗糙表面性质的值  此值用于控制表面的微观结构。  请注意，仅介于 0 与 1 之间的值有效，例如 0.3。
opacity	不透明值  请注意，仅介于 0 与 1 之间的值有效，例如 0.3。

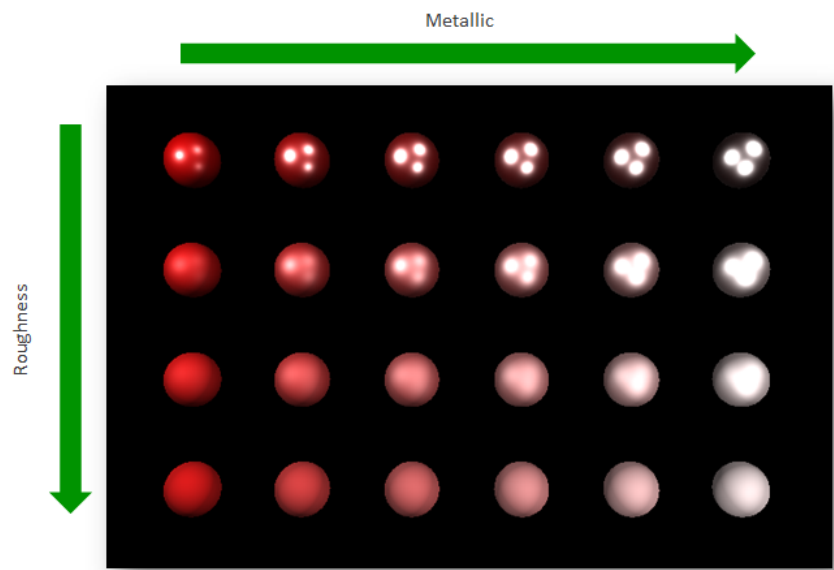


图 12.1. 基于物理的材料示例

### 12.10.3.7. PBR Phong 材料

PBR Phong 材料使用物理上正确的 Phong 反射模型定义网格表面的视觉外观。

表 12.118. PBR Phong 材料的属性

属性名称	说明
ambient	物体被环境光照亮时反射的颜色
diffuse	物体被纯白光照亮时在所有方向上均匀反射的颜色
emissive	对象的自照明颜色
shininess	光亮因素
specular	具有光泽表面的物体反射的颜色
metallic	金属表面性质的值  此值在漫射和镜面反射之间插入。  请注意，仅介于 0 与 1 之间的值有效，例如 0.3。
opacity	不透明值  请注意，仅介于 0 与 1 之间的值有效，例如 0.3。

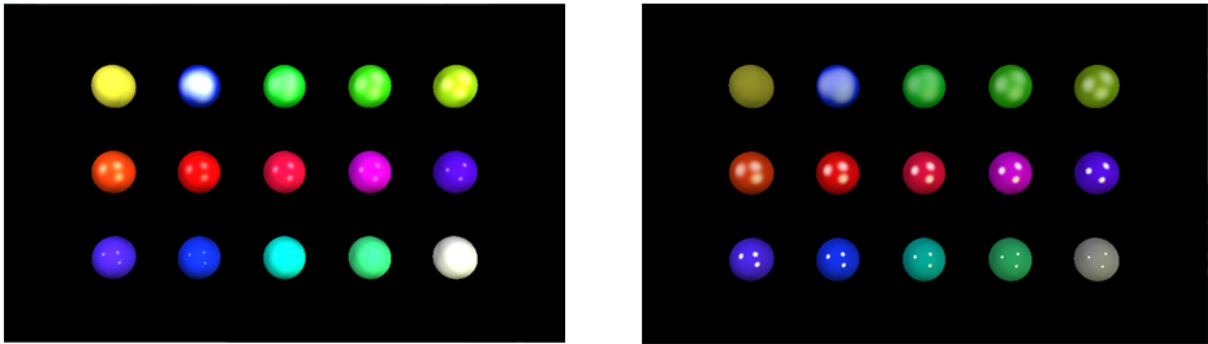


图 12.2. 非标准化材料（左）和标准化材料（右）的示例

### 12.10.3.8. 点光

点光为场景添加像灯泡一样向所有方向发光的光。

表 12.119. 点光的属性

属性名称	说明
enabled	如果为 true，则启用控件



属性名称	说明
color	光的颜色
intensity	光的强度，0.0 表示无点光
attenuationConstant	光随着距离的增加而减弱的常数因数
attenuationLinear	光线随着距离的增加而减弱的线性因数
attenuationQuadratic	光随着距离的增加而减弱的二次因数

### 12.10.3.9. 场景图

场景图用于将 3D 对象放入视图中。

表 12.120. 场景图的属性

属性名称	说明
visible	如果为 true，则控件及其子代控件可见
width	控件的宽度（以像素为单位）
height	控件的高度（以像素为单位）
x	控件相对于其父代控件的 x 坐标
y	控件相对于其父代控件的 y 坐标

### 12.10.3.10. 场景图节点

场景图节点是子节点，并且被添加到场景图或另一场景图节点。您可以使用场景图节点在 3D 场景中使用变换属性放置 3D 控件。可以将下列 3D 控件添加到场景图节点中：

- ▶ 摄像头
- ▶ 定向光
- ▶ 网格
- ▶ 点光
- ▶ 聚光

表 12.121. 场景图节点的属性

属性名称	说明
visible	如果为 true，则控件及其子代控件可见
rotationX	x 轴上的旋转
rotationY	y 轴上的旋转



属性名称	说明
rotationZ	Z 轴上的旋转
scalingX	X 轴上的缩放
scalingY	Y 轴上的缩放
scalingZ	Z 轴上的缩放
translationX	X 轴上的转换
translationY	Y 轴上的转换
translationZ	Z 轴上的转换

### 12.10.3.11. 聚光

聚光添加将照明限制为影响锥的光。

表 12.122. 聚光的属性

属性名称	说明
enabled	如果为 true，则启用控件
color	光的颜色
intensity	光的强度，0.0 表示无聚光
attenuationConstant	光随着距离的增加而减弱的常数因数
attenuationLinear	光线随着距离的增加而减弱的线性因数
attenuationQuadratic	光随着距离的增加而减弱的二次因数
coneAngleInner	光的内锥角
coneAngleOuter	光的外锥角

## 12.11. 控件功能

以下列表包含对所实现的所有控件功能的说明以及对如何在 EB GUIDE 模型中使用这些功能的简要说明。

### 12.11.1. 公共

#### 12.11.1.1. 子代可视性选择



“子代可视性选择”控件功能用于处理子代控件的可视性。一次只能看见一个子代控件的内容。

表 12.123. “子代可视性选择”控件功能的属性

属性名称	说明
containerIndex	父代控件的子代控件的索引
containerMapping	<p>如果设置了映射，则将重新根据容器的每个子代在 containerMapping 中的相应值确定其位置。</p> <p>如果未设置或未定义映射，或者长度与容器中的子代控件数不匹配，则不会使用映射。改为将控件树中的控件顺序用作其索引。最高的子代索引为 0，下一个子代索引为 1，等等。</p>

### 12.11.1.2. 已启用

“已启用”控件功能用于向控件中添加 enabled 属性。

表 12.124. “已启用”控件功能的属性

属性名称	说明
enabled	如果为 true，则控件会对触摸和按键输入作出反应

### 12.11.1.3. 已聚焦

“已聚焦”控件功能使控件可以具有输入焦点。

表 12.125. “已聚焦”控件功能的属性

属性名称	说明
focusable	<p>定义控件是否获得了焦点。可能的值：</p> <ul style="list-style-type: none"><li>▶ not focusable (=0)</li><li>▶ only by touch (=1)</li><li>▶ only by key (=2)</li><li>▶ focusable (=3)</li></ul>
focused	如果为 true，则控件获得了焦点

### 12.11.1.4. 多行

“多行”控件功能用于为标签启用换行符。



限制:

- ▶ “多行”控件功能仅对标签控件可用。

表 12.126. “多行”控件功能的属性

属性名称	说明
lineGap	行间间距的大小。负值将减小间距，而正值将增大间距。  当 line gap 太小（高负值）时，它不再有任何效果，并且文本在一行中呈现。这发生在设置字体样式为PT_Sans_Narrow，大小设置为 30，line gap 定义为-50时。
maxLineCount	可见行数的最大值0 = 无限制

提示



使用的行数

使用脚本函数 `getLineCount`，可以获取文本的行数。

有关这方面的更多信息，请参见 [节12.4.3.2.15](#)。

注意



字符替换

序列 '\ ' 将替换为 '\\ '。序列 '\ 'n' 将替换为 ' \n '。

如果标签大小增大，以便文本能够显示在一行内，则 '\n' 将替换为 ' '。

### 12.11.1.5. 已按下

“已按下”控件功能用于定义可以按下某个控件。

限制:

- ▶ 添加“已按下”控件功能，会自动添加“焦距”控件功能。

表 12.127. “已按下”控件功能的属性

属性名称	说明
pressed	如果为 true，则表示某个键在控件获得焦点的同时被按下

将“已触摸”控件功能与“触摸已按下”控件功能结合使用可以对按钮进行建模。

### 12.11.1.6. 已选中

“已选中”控件功能用于向控件中添加 `selected` 属性。它通常由应用程序或 HMI 建模器设置。任何其他软件元素都无法更改它。



表 12.128. “已选中”控件功能的属性

属性名称	说明
selected	如果为 true，则控件处于选中状态

### 12.11.1.7. 选择群组

“选择群组”控件功能用于对一组单选按钮进行建模。在列表中，每个单选按钮都具有“选择群组”控件功能和唯一的按钮 ID。

将数据池项用于 buttonValue 属性。将数据池项分配给单选按钮组中的所有控件。

选择和取消选择按钮群组中的某个控件可以通过用于设置 buttonValue 属性的应用程序完成。或者，可以通过触摸或键输入以及添加用于设置按钮值的条件来触发更改。

限制：

- ▶ 添加“选择群组”控件功能，会自动添加“选定”控件功能。

表 12.129. “选择群组”控件功能的属性

属性名称	说明
buttonId	用于标识按钮群组中的某个按钮的 ID
buttonValue	按钮的当前值。如果此值与 buttonId 相匹配，则按钮处于选中状态。
selected	对 buttonID 和 buttonValue 进行求值，确定它们是否相同。如果为 true，则按钮处于选中状态。

### 12.11.1.8. 旋转

“旋转”控件功能用于将控件变为旋转按钮。具有“旋转”控件功能的控件会通过更改内部值对增大和减小事件作出反应。“旋转”控件功能可用于创建比例尺、进度条或具有预览值的控件。

表 12.130. “旋转”控件功能的属性

属性名称	说明
currentValue	当前旋转值
maxValue	currentValue 属性的最大值
minValue	currentValue 属性的最小值
incValueTrigger	如果为 true，则 currentValue 属性增大 1
incValueReaction	对 currentValue 属性增大的动作反应



属性名称	说明
decValueTrigger	如果为 true，则当前值减小 1
decValueReaction	对 currentValue 属性减小的动作反应
steps	用于计算 currentValue 属性增大或减小的步骤数
valueWrapAround	可能的值： <ul style="list-style-type: none"><li>▶ true: currentValue 属性在反向边框处继续，前提是超过了 minValue 或 maxValue。</li><li>▶ false: currentValue 属性不会减小/增大，前提是超过了 minValue 或 maxValue。</li></ul>

### 12.11.1.9. 文本切断

“文本切断”控件功能，在 text 属性的内容无法容纳在控件区域时，切断该内容。控件功能启用不同于默认设置 trailing 的切断。

限制：

- ▶ “文本截断”控件功能支队标签控件可用。

表 12.131. “文本切断”控件功能的属性

属性名称	说明
truncationPolicy	对于单行文本，truncationPolicy 属性定义了切断位置。可能的值： <ul style="list-style-type: none"><li>▶ leading (=0)：文本将在文本开头进行替换</li><li>▶ trailing (=1)：文本将在文本结尾进行替换</li></ul> 对于多行文本，truncationPolicy 属性定义了文本替换位置。可能的值： <ul style="list-style-type: none"><li>▶ leading (=0)：将替换开头的行，而第一个可见行的文本将在文本开头切断。</li><li>▶ trailing (=1) 将替换结尾的行，且最后一个可见行的文本，将在文本结尾切断。</li></ul>
truncationSymbol	替代所替换的文本部分显示的字符串

### 12.11.1.10. 已触摸

“已触摸”控件功能使控件可以对触摸输入作出反应。

表 12.132. “已触摸”控件功能的属性

属性名称	说明
touchable	如果为 true，则控件会对触摸输入作出反应
touched	如果为 true，则表示控件当前处于被触摸状态
touchPolicy	定义如何处理跨控件边界的触摸和移动。可能的值： <ul style="list-style-type: none"><li>▶ Press then react (=0)：先按，然后控件将作出反应。移动和放开的通知仅在控件区域有效。</li><li>▶ Press and grab (=1)：按住以抓取接触点。接触点仍保持被抓取状态，即使它移出控件区域也是如此。</li><li>▶ Press then react on contact (=3)：即使接触点在控件边界外进入按下状态，后续移动和放开事件也将传递给控件。</li></ul>
touchBehavior	定义触摸评估。可能的值： <ul style="list-style-type: none"><li>▶ Whole area (=0)：为了识别已触摸的控件，渲染器将对控件的裁剪框进行求值。</li><li>▶ Visible pixels (=1)：为了识别已触摸的控件，渲染器将对已触摸像素所属的控件进行求值。</li></ul> <p>图像中 <b>alpha</b> 透明度或像素内字母为 <b>O</b> 或 <b>A</b> 的透明像素是无法触摸的。</p> <p>请注意 Visible pixels 值对标签不起作用。</p>

将“已触摸”控件功能与“已按下”控件功能结合使用可以对按钮进行建模。

#### 提示



#### 性能建议

如果您的项目中性能很重要，请将 touchBehavior 属性设置为 Whole area (=0)。EB GUIDE GTF 对 Whole area (=0) 求值比 Visible pixels (=1) 快。

## 12.11.2. 效果

### 12.11.2.1. 边框

“边框”控件功能用于向控件中添加可配置的边框。边框从控件边界开始并位于控件内。

限制：

► 控件功能可用于矩形。

表 12.133. “边框”控件功能的属性

属性名称	说明
borderThickness	边框的粗细（以像素为单位）
borderColor	用于渲染边框的颜色
borderStyle	用于渲染边框的样式

### 12.11.2.2. 着色

“着色”控件功能用于对控件及其控件子树进行着色。如果 **alpha** 值不为“不透明”，则它还会影响透明度。



#### 例 12.3. 着色控件功能的用法

对于介于 0.0 和 1.0 之间具有 **RGBA** 元素的所有颜色，“着色”控件功能中的算法会将控件的当前颜色值乘以 **colorationColor** 属性值。乘法按每个像素和元素方式完成。

使用不透明的蓝色对半透明的灰色着色会生成半透明的暗蓝色，如下所示：

$$(0.5, 0.5, 0.5, 0.5) * (0.0, 0.0, 1.0, 1.0) = (0.0, 0.0, 0.5, 0.5)$$

表 12.134. “着色”控件功能的属性

属性名称	说明
colorationEnabled	如果为 true，则将使用着色
colorationColor	用于着色的颜色

### 12.11.2.3. 笔划

“笔划”控件功能用于激活可配置的文本边框，即标签边框。

限制：

► 控件功能可用于标签。

表 12.135. “笔划”控件功能的属性

属性名称	说明
strokeEnabled	如果为 true，则将使用笔划
strokeThickness	边框的粗细（以像素为单位）



属性名称	说明
strokeColor	用于渲染边框的颜色

### 12.11.3. 焦点

“焦点”控件功能分类提供了和焦点管理相关的控件功能。

#### 12.11.3.1. 自动焦点

通过使用“自动焦点”控件功能，可以预先定义子代控件获得焦点的顺序。“自动焦点”控件功能用于检查有 focusable 属性的子代控件的控件子树。

布局中控件的顺序将用于计算焦点顺序。算法从左上角或右上角开始，具体取决于布局方向。

限制：

- ▶ “自动焦点”控件功能自动添加“已集中”控件功能。

表 12.136. “自动焦点”控件功能的属性

属性名称	说明
focusNext	焦点索引的递增条件
focusPrev	焦点索引的递减条件
focusFlow	焦点的行为会在层次结果内发生变化。可能的值： <ul style="list-style-type: none"><li>▶ stop at hierarchy (=0)</li><li>▶ wrap within hierarchy level (=1)</li><li>▶ step up in hierarchy (=2)</li></ul>
focusedIndex	当前获得焦点的子代控件作为第 <b>n</b> 个可获得焦点的子代控件的索引
initFocus	此索引用于定义初始化时获得焦点的子代控件。如果控件无法获得焦点，则将使用下一个可获得焦点的子代。

#### 12.11.3.2. 用户定义焦点

“用户定义焦点”控件功能用于为控件启用其他焦点功能。使用此功能的控件用于管理其控件子树的局部焦点层次结构。

限制：



- ▶ “用户定义焦点”控件功能可自动添加“已获得焦点的”控件功能。

表 12.137. “用户定义焦点”控件功能的属性

属性名称	说明
focusNext	用于将焦点分配给下一个子代控件的触发器
focusOrder	<p>通过 focusOrder 属性，可以在分配焦点时跳过子代控件。子代控件的 ID 对应于它在子树中的位置。默认情况下，将跳过无法获得焦点的子代控件。子代控件获得焦点的顺序如下：</p> <ul style="list-style-type: none"><li>▶ 已定义：将使用用户定义的控件顺序</li><li>▶ 未定义：将改为使用默认的控件顺序</li></ul> <p>每个子代控件都需要“已聚焦”控件功能，否则不会对控件进行焦点处理。示例：<b>focusOrder=1 0 2</b> 表示第二个控件首先获得焦点，然后第一个控件获得焦点，最后第三个控件获得焦点。</p>
focusPrev	用于将焦点分配给上一个子代的触发器
focusFlow	<p>焦点的行为会在层次结果内发生变化。可能的值：</p> <ul style="list-style-type: none"><li>▶ stop at hierarchy level (=0)</li><li>▶ wrap within hierarchy level (=1)</li><li>▶ step up in hierarchy (=2)</li></ul>
focusedIndex	索引用于定义子代控件在 focusOrder 列表中的位置。如果控件无法获得焦点，则将使用列表中的下一个子代。
initFocus	初始化时获得焦点的子代控件的索引

## 12.11.4. 操作动作

### 12.11.4.1. 滑动操作动作

快速轻触表面

限制：

- ▶ 添加“滑动操作动作”控件功能，自动添加“操作动作”和“已触摸”。

表 12.138. “滑动操作动作”控件功能的属性

属性名称	说明
onGestureFlick	在识别操作动作后触发的动作反应



属性名称	说明
	动作反应参数： <ul style="list-style-type: none"><li>▶ speed: 滑动操作动作的相对速度 以像素/ms 为单位的速度除以 flickMinLength/flickMaxTime</li><li>▶ directionX: 操作动作的方向矢量的 X 部分</li><li>▶ directionY: 操作动作的方向矢量的 Y 部分</li></ul>
flickMaxTime	在操作动作被识别为按住操作动作之前接触必须保持原位的最长时间（以毫秒为单位）
flickMinLength	被识别为滑动操作动作之前接触必须在表面上移动的最短距离

### 12.11.4.2. 按住操作动作

按住操作动作而不移动

限制:

- ▶ 添加“按住操作动作”控件功能，自动添加“操作动作”和“已触摸”控件功能。
- ▶ “按住操作动作”控件功能不会触发“触摸丢失”控件功能。

表 12.139. “按住操作动作”控件功能的属性

属性名称	说明
onGestureHold	在识别操作动作后触发的动作反应。每次接触仅触发一次动作反应：当 holdDuration 已到期并且接触仍然在初始触摸位置附近的小边界框内时。  动作反应参数： <ul style="list-style-type: none"><li>▶ x: 接触位置的 x 坐标</li><li>▶ y: 接触位置的 y 坐标</li></ul>
holdDuration	在操作动作被识别为按住操作动作之前接触必须保持原位的最短时间（以毫秒为单位）

### 12.11.4.3. 长按操作动作

长按操作动作而不移动

限制:





- ▶ 添加“长按操作动作”控件功能，自动添加“操作动作”和“已触摸”控件功能。
- ▶ “长按操作动作”控件功能不会触发“触摸丢失”控件功能。

表 12.140. “长按操作动作”控件功能的属性

属性名称	说明
onGestureLongHold	在识别操作动作后触发的动作反应。每次接触仅触发一次动作反应：当 longHoldDuration 已到期并且接触仍然在初始触摸位置附近的小边界框内时  动作反应参数： <ul style="list-style-type: none"><li>▶ x: 接触位置的 X 坐标</li><li>▶ y: 接触位置的 Y 坐标</li></ul>
longHoldDuration	在操作动作被识别为长按操作动作之前接触必须保持原位的最短时间（以毫秒为单位）

#### 12.11.4.4. 路径动作

一点接触所绘制的形状与一组已知形状进行比较。

限制：

- ▶ 添加“路径动作”控件功能，自动添加“操作动作”和“已触摸”控件功能。

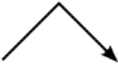
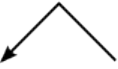





表 12.141. “路径动作”控件功能的属性

属性名称	说明
onPath	输入的形状匹配时触发的动作反应。仅当已触发 onPathStart 时，才会触发此动作反应。动作反应参数： <ul style="list-style-type: none"><li>▶ gestureId: 匹配路径的 ID</li></ul>
onPathStart	接触移出最小框后触发的动作反应（pathMinXBox, pathMinYBox）。
onPathNotRecognized	输入的形状不匹配时触发的动作反应。仅当已触发 onPathStart 时，才会触发此动作反应。
pathMinXBox	在路径动作识别器开始考虑输入之前接触必须移动的最短距离的 X 坐标（以像素为单位）
pathMinYBox	在路径动作识别器开始考虑输入之前接触必须移动的最短距离的 Y 坐标（以像素为单位）

##### 12.11.4.4.1. 操作动作 ID

操作动作识别符取决于路径动作识别器的配置。下表显示了 EB GUIDE 中提供的示例配置。

表 12.142. ....中提供的路径动作样本配置 EB GUIDE

识别码	形状	说明
0		从左到右的屋顶形状
1		从右到左的屋顶形状
2		从左到右的水平线
3		从右到左的水平线
4		勾选标记
5		从左到右的波浪形状
6		从右到左的波浪形状

**12.11.4.5. 二指缩放操作动作**



相向或相对移动的两点接触

限制:

- ▶ 添加“二指缩放操作动作”控件功能，自动添加“操作动作”和“已触摸”控件功能。

表 12.143. “二指缩放操作动作”控件功能的属性

属性名称	说明
onGesturePinchStart	在识别操作动作的开始部分后触发的动作反应。动作反应参数： <ul style="list-style-type: none"><li>▶ ratio: 当前接触距离与初始接触距离之比</li><li>▶ centerX: 两点接触之间当前中心点的 X 坐标</li><li>▶ centerY: 两点接触之间当前中心点的 Y 坐标</li></ul>
onGesturePinchUpdate	二指缩放比率或中心点变化时触发的动作反应动作反应参数： <ul style="list-style-type: none"><li>▶ ratio: 当前接触距离与初始接触距离之比</li><li>▶ centerX: 两点接触之间当前中心点的 X 坐标</li><li>▶ centerY: 两点接触之间当前中心点的 Y 坐标</li></ul>
onGesturePinchEnd	在完成操作动作后触发的动作反应动作反应参数： <ul style="list-style-type: none"><li>▶ ratio: 当前接触距离与初始接触距离之比</li><li>▶ centerX: 两点接触之间当前中心点的 X 坐标</li><li>▶ centerY: 两点接触之间当前中心点的 Y 坐标</li></ul>
pinchThreshold	在识别操作动作之前每点接触必须从其初始位置移动的最短距离（以像素为单位）。

12.11.4.6. 旋转操作动作

沿圆圈移动的两点接触

限制:

- ▶ 添加“旋转动作操作”控件功能，自动添加“操作动作”和“已触摸”控件功能。

表 12.144. “旋转操作动作”控件功能的属性

属性名称	说明
onGestureRotateStart	在识别操作动作的开始部分后触发的动作反应
onGestureRotateUpdate	所识别的角度或中心点变化时触发的动作反应



属性名称	说明
onGestureRotateEnd	在完成操作动作后触发的动作反应
rotateThreshold	在识别操作动作的开始部分之前每点接触必须从其初始位置移动的最短距离（以像素为单位）

onGestureRotateEnd, onGestureRotateStart, 和 onGestureRotateUpdate 的反应参数:

- ▶ **angle**: 涉及的两点接触的初始位置所指定的线与两点接触的当前位置所指定的线之间的角度。此角度按逆时针方向测量。
- ▶ **centerX**: 两点接触之间当前中心点的 **X** 坐标
- ▶ **centerY**: 两点接触之间当前中心点的 **Y** 坐标

## 12.11.5. 输入处理

### 12.11.5.1. 操作动作

“操作动作”控件功能使控件可以对触摸操作动作作出反应。

限制:

- ▶ 添加“操作动作”控件功能，自动添加“已触摸”控件功能。
- ▶ “操作动作”控件功能没有其他属性。

### 12.11.5.2. 按下的键

“按下的键”控件功能使控件可以对按下的键作出反应。

限制:

- ▶ 添加“按下的键”控件功能，自动添加“已按下”和“聚焦”控件功能。

表 12.145. “按下的键”控件功能的属性

属性名称	说明
keyPressed	控件对按下某个键的动作反应  动作反应参数: <ul style="list-style-type: none"><li>▶ <b>keyId</b>: 已处理键的 ID</li></ul>



### 12.11.5.3. 放开的键

“放开的键”控件功能使控件可以对放开的键作出反应。

限制:

- ▶ 添加“放开的键”控件功能，自动添加“已按下”和“聚焦”控件功能。

表 12.146. “放开的键”控件功能的属性

属性名称	说明
keyShortReleased	控件对放开某个键的动作反应  动作反应参数: <ul style="list-style-type: none"><li>▶ keyId: 已处理键的 ID</li></ul>

### 12.11.5.4. 键状态已变化

“键状态已变化”控件功能使控件可以对按下或放开的键作出反应。它用于定义对“短按”、“长”、“超长”及“连续”等键输入作出的反应。

限制:

- ▶ 添加“键状态已变化”控件功能，自动添加“已按下”和“聚焦”控件功能。

表 12.147. “键状态已变化”控件功能的属性

属性名称	说明
keyStatusChanged	控件对按下或放开某个键的动作反应  动作反应参数: <ul style="list-style-type: none"><li>▶ keyId: 已处理键的 ID</li><li>▶ status: 状态更改的数字 ID</li></ul>

### 12.11.5.5. Unicode 键

“Unicode 键”控件功能使控件可以对 Unicode 键输入作出反应。

限制:

- ▶ 添加“Unicode 键”控件功能，自动添加“已按下”和“聚焦”控件功能。



表 12.148. "Unicode 键"控件功能的属性

属性名称	说明
keyUnicode	<p>控件对 <b>Unicode</b> 键输入的动作反应</p> <p>动作反应参数:</p> <ul style="list-style-type: none"> <li>▶ keyId: 已处理键的 ID</li> </ul>

### 12.11.5.6. 移入

"移入"控件功能使控件可以对朝向其边界内的移动作出反应。

限制:

- ▶ 添加"移入"控件功能, 自动添加"已触摸"控件功能。

表 12.149. "移入"控件功能的属性

属性名称	说明
moveIn	<p>控件对移入其边界的动作反应</p> <p>动作反应参数:</p> <ul style="list-style-type: none"> <li>▶ touchId: 用户已点击或释放的触摸屏的 ID</li> <li>▶ x: x 坐标</li> <li>▶ y: y 坐标</li> <li>▶ fingerId: 移过控件的接触点的 ID</li> </ul>

### 12.11.5.7. 移出

"移出"控件功能使控件可以对朝向其边界外的移动作出反应。

限制:

- ▶ 添加"移出"控件功能, 自动添加"已触摸"控件功能。

表 12.150. "移出"控件功能的属性

属性名称	说明
moveOut	<p>控件对移出其边界的动作反应</p> <p>动作反应参数:</p> <ul style="list-style-type: none"> <li>▶ touchId: 用户已点击或释放的触摸屏的 ID</li> </ul>



属性名称	说明
	<ul style="list-style-type: none"> <li>▶ x: <b>x</b> 坐标</li> <li>▶ y: <b>y</b> 坐标</li> <li>▶ fingerId: 移过控件的接触点的 ID</li> </ul>

### 12.11.5.8. 移过

“移过”控件功能使控件可以对其边界内的移动作出反应。

限制:

- ▶ 添加“移过”控件功能，自动添加“已触摸”控件功能。

表 12.151. “移过”控件功能的属性

属性名称	说明
moveOver	<p>控件对其边界内的动作反应</p> <p>动作反应参数:</p> <ul style="list-style-type: none"> <li>▶ touchId: 用户已点击或释放的触摸屏的 ID</li> <li>▶ x: <b>x</b> 坐标</li> <li>▶ y: <b>y</b> 坐标</li> <li>▶ fingerId: 移过控件的接触点的 ID</li> </ul>

### 12.11.5.9. 可移动的

“可移动的”控件功能使控件可以通过触摸进行移动。

限制:

- ▶ 添加“可移动的”控件功能，自动添加“已触摸”和“触摸移动”控件功能。

表 12.152. “可移动的”控件功能的属性

属性名称	说明
moveDirection	<p>控件的移动方向。可能的值:</p> <ul style="list-style-type: none"> <li>▶ horizontal (=0)</li> <li>▶ vertical (=1)</li> <li>▶ free (=2)</li> </ul>



### 12.11.5.10. 旋转

“旋转”控件功能使控件可以对旋转作出反应。

限制：

- ▶ 添加“旋转”控件功能，自动添加“聚焦”控件功能。

表 12.153. “旋转”控件功能的属性

属性名称	说明
rotaryReaction	控件对旋转作出反应。如果为 true，则控件会对即将发生的旋转事件作出反应。  动作反应参数： <ul style="list-style-type: none"><li>▶ rotaryId：整数 ID</li><li>▶ increment：发送了即将发生的事件时旋转输入变化的单位数</li></ul>

### 12.11.5.11. 触摸丢失

“触摸丢失”控件功能使控件可以对抓取丢失作出反应。

接触是操作动作的组成部分或者在未松开的情况下离开触摸屏时，接触会消失。在这些情况下，不会执行 touchShortReleased 动作反应。

限制：

- ▶ 添加“触摸丢失”控件功能，自动添加“已触摸”控件功能。

表 12.154. “触摸丢失”控件功能的属性

属性名称	说明
onTouchGrabLost	控件对触摸接触丢失的动作反应  动作反应参数： <ul style="list-style-type: none"><li>▶ touchId：用户已点击或释放的触摸屏的 ID</li><li>▶ x：x 坐标</li><li>▶ y：y 坐标</li><li>▶ fingerId：移过控件的接触点的 ID</li></ul>

### 12.11.5.12. 触摸移动





“触摸移动”控件功能使控件可以对触摸和移动作出反应。

限制:

- ▶ 添加“触摸移动”控件功能，自动添加“已触摸”控件功能。

表 12.155. “触摸移动”控件功能的属性

属性名称	说明
touchMoved	<p>控件对触摸和移动的动作反应</p> <p>动作反应参数:</p> <ul style="list-style-type: none"><li>▶ touchId: 用户已点击或释放的触摸屏的 ID</li><li>▶ x: x 坐标</li><li>▶ y: y 坐标</li><li>▶ fingerId: 移过控件的接触点的 ID</li></ul>

### 12.11.5.13. 触摸已按下

“触摸已按下”控件功能使控件可以对按下作出反应。

限制:

- ▶ 添加“触摸已按下”控件功能，自动添加“已触摸”控件功能。

表 12.156. “触摸已按下”控件功能的属性

属性名称	说明
touchPressed	<p>控件对按下的动作反应</p> <p>动作反应参数:</p> <ul style="list-style-type: none"><li>▶ touchId: 用户已点击或释放的触摸屏的 ID</li><li>▶ x: x 坐标</li><li>▶ y: y 坐标</li><li>▶ fingerId: 移过控件的接触点的 ID</li></ul>

### 12.11.5.14. 触摸已放开

“触摸已放开”控件功能使控件可以对放开作出反应。

限制:



- ▶ 添加“触摸已放开”控件功能，自动添加“已触摸”控件功能。

表 12.157. “触摸已放开”控件功能的属性

属性名称	说明
touchShortReleased	控件对放开的动作反应  动作反应参数： <ul style="list-style-type: none"><li>▶ touchId: 用户已点击或释放的触摸屏的 ID</li><li>▶ x: x 坐标</li><li>▶ y: y 坐标</li><li>▶ fingerId: 移过控件的接触点的 ID</li></ul>

**12.11.5.15. 触摸状态已变化**

“触摸状态已变化”控件功能使控件可以对其触摸状态变化作出反应。

限制:

- ▶ 添加“触摸状态已改变”控件功能，自动添加“已触摸”控件功能。

表 12.158. “触摸状态已变化”控件功能的属性

属性名称	说明
touchStatusChanged	控件对其触摸状态变化的动作反应  动作反应参数： <ul style="list-style-type: none"><li>▶ touchId: 用户已点击或释放的触摸屏的 ID</li><li>▶ x: x 坐标</li><li>▶ y: y 坐标</li><li>▶ touchStatus: 触摸类型的 ID</li></ul> 可能的值： <ul style="list-style-type: none"><li>▶ 0: 新接触</li><li>▶ 1: 触摸按下</li><li>▶ 2: 触摸移动</li><li>▶ 3: 触摸已放开</li><li>▶ 4: 无触摸的移动</li></ul>



属性名称	说明
	<ul style="list-style-type: none"><li>▶ 5: 触摸过</li><li>▶ 6: 任何状态已变化</li><li>▶ fingerId: 移过控件的接触点的 ID</li></ul>

## 12.11.6. 布局

### 12.11.6.1. 绝对布局

父代控件的“绝对布局”控件功能用于定义子代控件的位置和大小。将忽略不可见的子代控件。添加的控件功能属性包含整数列表。每项列表元素均已被映射到一个子代控件。

限制:

- ▶ “绝对布局”控件功能不包含以下控件功能:
  - ▶ “方框布局”
  - ▶ “流式布局”
  - ▶ “栅格布局”
  - ▶ “列表布局”

表 12.159. “绝对布局”控件功能的属性

属性名称	说明
itemLeftOffset	用于存储子代控件与左侧边框的偏移量的整数列表。每项列表元素均已被映射到一个子代控件。
itemTopOffset	用于存储子代控件与顶部边框的偏移量的整数列表每项列表元素均已被映射到一个子代控件。
itemRightOffset	用于存储子代控件与右侧边框的偏移量的整数列表。每项列表元素均已被映射到一个子代控件。
itemBottomOffset	用于存储子代控件与底部边框的偏移量的整数列表每项列表元素均已被映射到一个子代控件。

### 12.11.6.2. 方框布局

“方框布局”控件功能用于定义每个子代控件的位置和大小。

子代控件的位置和大小属性由父代控件设置。计算中将忽略不可见的子代控件。



限制:

- ▶ “方框布局”控件功能不包含以下控件功能:
  - ▶ “绝对布局”
  - ▶ “流式布局”
  - ▶ “栅格布局”
  - ▶ “列表布局”

表 12.160. “方框布局”控件功能的属性

属性名称	说明
gap	两个子代控件之间的间距，取决于布局方向
layoutDirection	列表元素（即，子代控件）的放置方向。可能的值： <ul style="list-style-type: none"><li>▶ horizontal (=0)</li><li>▶ vertical (=1)</li></ul>

12.11.6.3. 流式布局

“流式布局”控件功能用于定义每个子代控件的位置和大小。

子代控件的位置和大小属性由父代控件设置。计算中将忽略不可见的子代控件。

限制:

- ▶ “流式布局”控件功能不包含以下控件功能:
  - ▶ “绝对布局”
  - ▶ “方框布局”
  - ▶ “栅格布局”
  - ▶ “列表布局”

表 12.161. “流式布局”控件功能的属性

属性名称	说明
horizontalGap	两个子代控件之间的水平间距
verticalGap	两个子代控件之间的垂直间距
layoutDirection	列表元素（即，子代控件）的放置方向。可能的值： <ul style="list-style-type: none"><li>▶ horizontal (=0)</li><li>▶ vertical (=1)</li></ul>



属性名称	说明
horizontalChildAlign	子代控件的水平对齐方式。可能的值： <ul style="list-style-type: none"><li>▶ leading (=0)：子代控件位于中间。</li><li>▶ center (=1)：子代控件位于顶部。</li><li>▶ trailing (=2)：子控件位于底部。</li></ul>
verticalChildAlign	子代控件的垂直对齐方式。可能的值： <ul style="list-style-type: none"><li>▶ center (=0)：子代控件位于中间。</li><li>▶ top (=1)：子代控件位于顶部</li><li>▶ bottom (=2)：子控件位于底部。</li></ul>

**12.11.6.4. 栅格布局**

“栅格布局”控件功能用于定义每个子代控件的位置和大小。

子代控件的位置和大小属性由父代控件设置。计算中将忽略不可见的子代控件。

限制：

- ▶ “栅格布局”控件功能不包含以下控件功能：
  - ▶ “绝对布局”
  - ▶ “方框布局”
  - ▶ “流式布局”
  - ▶ “列表布局”

表 12.162. “栅格布局”控件功能的属性

属性名称	说明
horizontalGap	两个子代控件之间的水平间距
verticalGap	两个子代控件之间的垂直间距
numRows	定义行数
numColumns	定义列数

**12.11.6.5. 布局留白**

“布局留白”控件功能用于向使用“流式布局”、“绝对布局”、“方框布局”或“栅格布局”控件功能的控件中添加可配置的留白。



表 12.163. “布局留白”控件功能的属性

属性名称	说明
leftMargin	左侧边框的留白
topMargin	顶部边框的留白
rightMargin	右侧边框的留白
bottomMargin	底部边框的留白

### 12.11.6.6. 列表布局

“列表布局”控件功能用于定义每个子代控件的位置和大小。

子代控件的位置属性和“列表索引”控件功能的 `listIndex` 属性由父代控件设置。

最适合与实例化器配合使用以创建子代控件。

有关“列表索引”控件功能的详细信息，请参见[节12.11.7.2](#)。

限制：

- ▶ “列表布局”旨在与实例化器一起使用。
- ▶ “列表布局”控件功能不包含以下控件功能：
  - ▶ “绝对布局”
  - ▶ “方框布局”
  - ▶ “流式布局”
  - ▶ “栅格布局”

表 12.164. “列表布局”控件功能的属性

属性名称	说明
layoutDirection	列表元素（即，子代控件）的放置方向。可能的值： <ul style="list-style-type: none"><li>▶ <code>horizontal (=0)</code></li><li>▶ <code>vertical (=1)</code></li></ul>
scrollOffset	滚动列表的像素数
scrollOffsetRebase	如果 <code>scrollOffsetRebase</code> 属性发生更改，则当前 <code>scrollOffset</code> 将转换为 <code>scrollIndex</code> 。其余的偏移量将写入 <code>scrollOffset</code> 属性。
firstListIndex	第一个可见列表元素的列表索引，由控件功能定义



属性名称	说明
scrollIndex	对其应用 scrollOffset 属性的基本列表索引。滚动从 scrollIndex 属性中指定的列表元素处开始。
scrollValue	当前滚动值
scrollValueMax	最大滚动值，该值被映射到列表结尾
scrollValueMin	最小滚动值，该值被映射到列表开头
bounceValue	只要 scrollOffset 属性的结果是有效滚动范围内的某个位置，bounceValue 属性就为零。它具有正值（前提是滚动位置超过了列表开头）和负值（如果滚动位置超过了列表结尾）。如果 bounceValue 已添加至 scrollOffset，则滚动范围将回到范围内。
bounceValueMax	scrollOffset 可移动到有效的滚动范围外的最大值。scrollOffset 如果用户尝试继续滚动，则会被截断。
segments	对于水平布局方向：行数 对于垂直布局方向：列数
listLength	列表元素数
wrapAround	可能的值： <ul style="list-style-type: none"><li>▶ true: scrollValue 属性在反向边框处继续，前提是超过了 scrollValueMin 或 scrollValueMax。</li><li>▶ false: scrollValue 属性不会减小/增大，前提是超过了 scrollValueMin 或 scrollValueMax。</li></ul>

12.11.6.7. 缩放模式

“缩放模式”控件功能定义图片的显示方式以及图片大小是否与控件大小不同。

限制：

- ▶ “缩放模式”控件功能只对控件图像有效。

表 12.165. “缩放模式”控件功能的属性

属性名称	说明
scaleMode	图片的缩放模式。可能的值： <ul style="list-style-type: none"><li>▶ 0 = original size</li><li>▶ 1 = fit to size</li><li>▶ 2 = keep aspect ratio</li></ul>

## 12.11.7. 列表管理

### 12.11.7.1. 行索引

“行索引”控件功能，定义了列表或表格中每行的唯一位置。

限制：

- ▶ “行索引”控件功能旨在与实例化器结合使用。

表 12.166. “行索引”控件功能的属性

属性名称	说明
lineIndex	表中当前行的索引

### 12.11.7.2. 列表索引

“列表索引”控件功能定义列表中控件的唯一位置。

限制：

- ▶ “列表索引”控件功能旨在与“列表布局”控件功能结合使用。

表 12.167. “列表索引”控件功能的属性

属性名称	说明
listIndex	列表中当前控件的索引

### 12.11.7.3. 模板索引

“模板索引”控件功能定义已用行模板的唯一位置。

限制：

- ▶ “模板索引”控件功能旨在与实例化器结合使用。

表 12.168. “模板索引”控件功能的属性

属性名称	说明
lineTemplateIndex	使用的行模板的索引

### 12.11.7.4. Viewport





"Viewport" 控件功能，用于修剪控件边框处的过大元素。

限制：

- ▶ "Viewport" 控件功能旨在与容器或列表结合使用。
- ▶ "Viewport" 控件功能，对以下模型元素产生了影响：
  - ▶ 将已添加 "Viewport" 控件的子控件缩减至控件尺寸内。
  - ▶ 将已添加 "Viewport" 控件缩减至其父代视图的尺寸内。

表 12.169. "Viewport" 控件功能的属性

属性名称	说明
xOffset	子代控件的已绘制区域中可见修剪的水平偏移量
yOffset	子代控件的已绘制区域中可见修剪的垂直偏移量

## 12.11.8. 3D

"3D" 类别中的控件功能仅适用于 3D 控件。

### 12.11.8.1. 摄像头视见区

"摄像头视见区"控件功能定义场景图中摄像头的绘图区域。

限制：

- ▶ "摄像头视见区"控件功能可用于摄像头。

表 12.170. "摄像头视见区"控件功能的属性

属性名称	说明
viewportX	场景图中视见区的 x 原点
viewportY	场景图中视见区的 y 原点
viewportWidth	视窗宽度（以像素为单位）
viewportHeight	视窗高度（以像素为单位）

### 12.11.8.2. 环境纹理

"环境纹理"控件功能将扩展配置值添加到材料。

限制:

- ▶ “环境纹理”控件功能可用于材料、PBR **Phong** 材料和 **PBR GGX** 材料。

表 12.171. “环境纹理”控件功能的属性

属性名称	说明
ambientTexture	纹理的文件名
ambientTextureAddressModeU	<b>u</b> 方向的纹理的地址模式。可能的值: <ul style="list-style-type: none"><li>▶ repeat (=0): 当在纹理边界之外访问时, 重复纹理。也称为 <b>wrap</b> 或 <b>tile</b></li><li>▶ clamp (=1): 当在纹理边界之外访问时, 使用纹理边缘处的像素。</li></ul>
ambientTextureAddressModeV	<b>v</b> 方向的纹理的地址模式。可能的值: <ul style="list-style-type: none"><li>▶ repeat (=0): 当在纹理边界之外访问时, 重复纹理。也称为 <b>wrap</b> 或 <b>tile</b></li><li>▶ clamp (=1): 当在纹理边界之外访问时, 使用纹理边缘处的像素。</li></ul>
ambientFilterMode	纹理的筛选模式。可能的值: <ul style="list-style-type: none"><li>▶ point (=0): 纹理根本不平滑。纹理最小时, 价格最低但容易出现混叠伪像。</li><li>▶ linear (=1): 也称为双线性过滤。在最小化时, 平滑纹理以减少混叠伪影。</li><li>▶ trilinear (=2): 更贵, 但比线性筛选效果更好。</li></ul>

### 12.11.8.3. 弥散纹理

“弥散纹理”控件功能将扩展配置值添加到材料。

限制:

- ▶ “弥散纹理”控件功能可用于材料、PBR **Phong** 材料和 **PBR GGX** 材料。

表 12.172. “弥散纹理”控件功能的属性

属性名称	说明
diffuseTexture	纹理的文件名
diffuseTextureAddressModeU	<b>u</b> 方向的纹理的地址模式。可能的值: <ul style="list-style-type: none"><li>▶ repeat (=0): 当在纹理边界之外访问时, 重复纹理。也称为 <b>wrap</b> 或 <b>tile</b></li></ul>

属性名称	说明
	<ul style="list-style-type: none"> <li>▶ clamp (=1): 当在纹理边界之外访问时, 使用纹理边缘处的像素。</li> </ul>
diffuseTextureAddressModeV	<p>V 方向的纹理的地址模式。可能的值:</p> <ul style="list-style-type: none"> <li>▶ repeat (=0): 当在纹理边界之外访问时, 重复纹理。也称为 <b>wrap</b> 或 <b>tile</b></li> <li>▶ clamp (=1): 当在纹理边界之外访问时, 使用纹理边缘处的像素。</li> </ul>
diffuseFilterMode	<p>纹理的筛选模式。可能的值:</p> <ul style="list-style-type: none"> <li>▶ point (=0): 纹理根本不平滑。纹理最小时, 价格最低但容易出现混叠伪像。</li> <li>▶ linear (=1): 也称为双线性过滤。在最小化时, 平滑纹理以减少混叠伪影。</li> <li>▶ trilinear (=2): 更贵, 但比线性筛选效果更好。</li> </ul>
diffuseSRGB	<p>如果启用了此属性, 将使用 <b>sRGB</b> 色域渲染 diffuseTexture 中选择的纹理。</p> <p>请注意, 要使用 <b>sRGB</b> 功能, 请在项目中心的“配置” &gt; “配置文件”下, 为 colorMode 属性选择 32-bit sRGB (=4) 或 32-bit sRGB (Emulated) (=5)。</p>

#### 12.11.8.4. 发光纹理

“发光纹理”控件功能将扩展配置值添加到材料。

限制:

- ▶ “发光纹理”控件功能可用于材料、PBR **Phong** 材料和 **PBR GGX** 材料。

表 12.173. “发光纹理”控件功能的属性

属性名称	说明
emissiveTexture	纹理的文件名
emissiveTextureAddressModeU	<p>U 方向的纹理的地址模式。可能的值:</p> <ul style="list-style-type: none"> <li>▶ repeat (=0): 当在纹理边界之外访问时, 重复纹理。也称为 <b>wrap</b> 或 <b>tile</b></li> <li>▶ clamp (=1): 当在纹理边界之外访问时, 使用纹理边缘处的像素。</li> </ul>
emissiveTextureAddressModeV	V 方向的纹理的地址模式。可能的值:

属性名称	说明
	<ul style="list-style-type: none"> <li>▶ repeat (=0): 当在纹理边界之外访问时, 重复纹理。也称为 <b>wrap</b> 或 <b>tile</b></li> <li>▶ clamp (=1): 当在纹理边界之外访问时, 使用纹理边缘处的像素。</li> </ul>
emissiveFilterMode	纹理的筛选模式。可能的值: <ul style="list-style-type: none"> <li>▶ point (=0): 纹理根本不平滑。纹理最小时, 价格最低但容易出现混叠伪像。</li> <li>▶ linear (=1): 也称为双线性过滤。在最小化时, 平滑纹理以减少混叠伪影。</li> <li>▶ trilinear (=2): 更贵, 但比线性筛选效果更好。</li> </ul>

### 12.11.8.5. 光地图纹理

“光地图纹理”控件功能将扩展配置值添加到材料。

限制:

- ▶ “光地图纹理”控件功能可用于材料、PBR Phong 材料和 PBR GGX 材料。

表 12.174. “光地图纹理”控件功能的属性

属性名称	说明
lightMapTexture	纹理的文件名
lightMapTextureAddressModeU	<b>u</b> 方向的纹理的地址模式。可能的值: <ul style="list-style-type: none"> <li>▶ repeat (=0): 当在纹理边界之外访问时, 重复纹理。也称为 <b>wrap</b> 或 <b>tile</b></li> <li>▶ clamp (=1): 当在纹理边界之外访问时, 使用纹理边缘处的像素。</li> </ul>
lightMapTextureAddressModeV	<b>v</b> 方向的纹理的地址模式。可能的值: <ul style="list-style-type: none"> <li>▶ repeat (=0): 当在纹理边界之外访问时, 重复纹理。也称为 <b>wrap</b> 或 <b>tile</b></li> <li>▶ clamp (=1): 当在纹理边界之外访问时, 使用纹理边缘处的像素。</li> </ul>
lightMapFilterMode	纹理的筛选模式。可能的值: <ul style="list-style-type: none"> <li>▶ point (=0): 纹理根本不平滑。纹理最小时, 价格最低但容易出现混叠伪像。</li> <li>▶ linear (=1): 也称为双线性过滤。在最小化时, 平滑纹理以减少混叠伪影。</li> </ul>



属性名称	说明
	▶ trilinear (=2): 更贵, 但比线性筛选效果更好。

### 12.11.8.6. 正常贴图纹理

“正常贴图”控件功能将扩展配置值添加到材料。

限制:

- ▶ “正常贴图纹理”控件功能可用于材料、PBR Phong 材料和 PBR GGX 材料。

表 12.175. “正常贴图”控件功能的属性

属性名称	说明
normalMapTexture	纹理的文件名
normalMapTextureAddressModeU	<b>u</b> 方向的纹理的地址模式。可能的值: <ul style="list-style-type: none"><li>▶ repeat (=0): 当在纹理边界之外访问时, 重复纹理。也称为 <b>wrap</b> 或 <b>tile</b></li><li>▶ clamp (=1): 当在纹理边界之外访问时, 使用纹理边缘处的像素。</li></ul>
normalMapTextureAddressModeV	<b>v</b> 方向的纹理的地址模式。可能的值: <ul style="list-style-type: none"><li>▶ repeat (=0): 当在纹理边界之外访问时, 重复纹理。也称为 <b>wrap</b> 或 <b>tile</b></li><li>▶ clamp (=1): 当在纹理边界之外访问时, 使用纹理边缘处的像素。</li></ul>
normalMapFilterMode	纹理的筛选模式。可能的值: <ul style="list-style-type: none"><li>▶ point (=0): 纹理根本不平滑。纹理最小时, 价格最低但容易出现混叠伪像。</li><li>▶ linear (=1): 也称为双线性过滤。在最小化时, 平滑纹理以减少混叠伪影。</li><li>▶ trilinear (=2): 更贵, 但比线性筛选效果更好。</li></ul>

### 12.11.8.7. 不透明纹理

“不透明纹理”控件功能将扩展配置值添加到材料。

限制:

- ▶ “不透明纹理”控件功能可用于材料、PBR Phong 材料和 PBR GGX 材料。



表 12.176. “不透明纹理”控件功能的属性

属性名称	说明
opaqueTexture	纹理的文件名
opaqueTextureAddressModeU	<b>u</b> 方向的纹理的地址模式。可能的值： <ul style="list-style-type: none"> <li>▶ <b>repeat (=0)</b>：当在纹理边界之外访问时，重复纹理。也称为 <b>wrap</b> 或 <b>tile</b></li> <li>▶ <b>clamp (=1)</b>：当在纹理边界之外访问时，使用纹理边缘处的像素。</li> </ul>
opaqueTextureAddressModeV	<b>v</b> 方向的纹理的地址模式。可能的值： <ul style="list-style-type: none"> <li>▶ <b>repeat (=0)</b>：当在纹理边界之外访问时，重复纹理。也称为 <b>wrap</b> 或 <b>tile</b></li> <li>▶ <b>clamp (=1)</b>：当在纹理边界之外访问时，使用纹理边缘处的像素。</li> </ul>
opaqueFilterMode	纹理的筛选模式。可能的值： <ul style="list-style-type: none"> <li>▶ <b>point (=0)</b>：纹理根本不平滑。纹理最小时，价格最低但容易出现混叠伪像。</li> <li>▶ <b>linear (=1)</b>：也称为双线性过滤。在最小化时，平滑纹理以减少混叠伪影。</li> <li>▶ <b>trilinear (=2)</b>：更贵，但比线性筛选效果更好。</li> </ul>

### 12.11.8.8. 反射纹理

“反射纹理”控件功能将扩展配置值添加到材料。

限制：

- ▶ “反射纹理”控件功能可用于材料、PBR **Phong** 材料和 **PBR GGX** 材料。

表 12.177. “反射纹理”控件功能的属性

属性名称	说明
reflectionTopTexture	纹理的文件名
reflectionBottomTexture	纹理的文件名
reflectionLeftTexture	纹理的文件名
reflectionRightTexture	纹理的文件名
reflectionFrontTexture	纹理的文件名
reflectionBackTexture	纹理的文件名

属性名称	说明
reflectionFilterMode	纹理的筛选模式。可能的值： <ul style="list-style-type: none"><li>▶ point (=0)：纹理根本不平滑。纹理最小时，价格最低但容易出现混叠伪像。</li><li>▶ linear (=1)：也称为双线性过滤。在最小化时，平滑纹理以减少混叠伪影。</li><li>▶ trilinear (=2)：更贵，但比线性筛选效果更好。</li></ul>

注意



“反射纹理”控件功能

EB GUIDE Studio 仅当为以下所有属性选择了图像文件时，才显示“反射纹理”控件功能：

- ▶ reflectionTopTexture
- ▶ reflectionBottomTexture
- ▶ reflectionLeftTexture
- ▶ reflectionRightTexture
- ▶ reflectionFrontTexture
- ▶ reflectionBackTexture

这些图像文件的大小必须相同。

### 12.11.8.9. 镜面纹理

“镜面纹理”控件功能将扩展配置值添加到材料。

限制：

- ▶ “镜面纹理”控件功能可用于材料、PBR Phong 材料和 PBR GGX 材料。

表 12.178. “镜面纹理”控件功能的属性

属性名称	说明
specularTexture	纹理的文件名
specularTextureAddressModeU	<b>u</b> 方向的纹理的地址模式。可能的值： <ul style="list-style-type: none"><li>▶ repeat (=0)：当在纹理边界之外访问时，重复纹理。也称为 <b>wrap</b> 或 <b>tile</b></li><li>▶ clamp (=1)：当在纹理边界之外访问时，使用纹理边缘处的像素。</li></ul>
specularTextureAddressModeV	<b>v</b> 方向的纹理的地址模式。可能的值：

属性名称	说明
	<ul style="list-style-type: none"><li>▶ repeat (=0)：当在纹理边界之外访问时，重复纹理。也称为 <b>wrap</b> 或 <b>tile</b></li><li>▶ clamp (=1)：当在纹理边界之外访问时，使用纹理边缘处的像素。</li></ul>
specularFilterMode	纹理的筛选模式。可能的值： <ul style="list-style-type: none"><li>▶ point (=0)：纹理根本不平滑。纹理最小时，价格最低但容易出现混叠伪像。</li><li>▶ linear (=1)：也称为双线性过滤。在最小化时，平滑纹理以减少混叠伪影。</li><li>▶ trilinear (=2)：更贵，但比线性筛选效果更好。</li></ul>

12.11.8.10. 色调映射

“色调映射”控件功能用于为场景图启用色调映射，即用于将亮度值映射到有限范围的方法。

限制：

- ▶ “色调映射”控件功能可用于场景图。

请注意，“色调映射”控件功能将实现 Erik Reinhard 及其他人描述的全局色调映射运算符。<sup>1</sup>

表 12.179. “色调映射”控件功能的属性

属性名称	说明
pureWhiteLuminance	映射到纯白色的最小亮度值。请注意，仅大于或等于 0 的值有效。

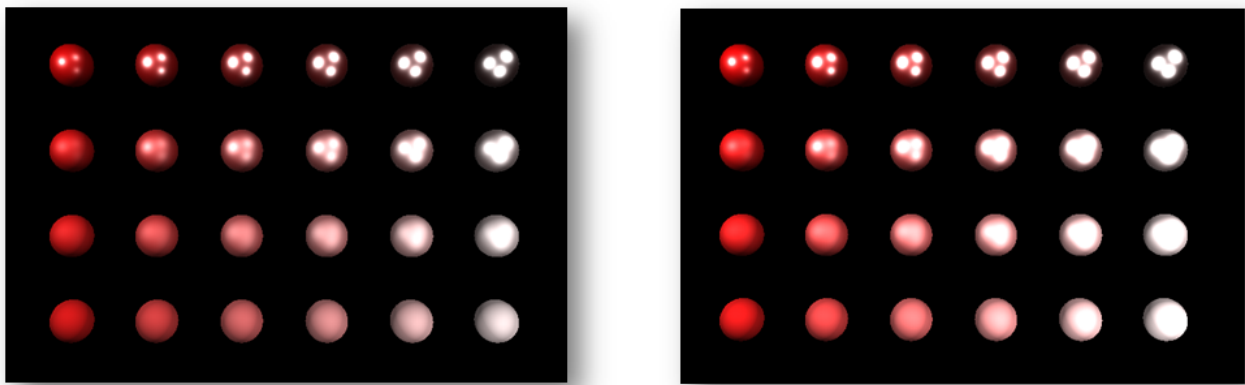


图 12.3. 不带色调映射的图像（左）和带色调映射的图像（右）的示例

<sup>1</sup>2002 年 “Proceedings of the 29th annual conference on Computer graphics and interactive techniques” 中第 267-276 页的 Photographic tone reproduction for digital images (Erik Reinhard 及其他人)。





## 12.11.9. 变换

“变换”分类的控件功能，修改控件的位置、形式和大小。

执行变换的顺序等于控件树中的顺序。如果多个变换适用于同一控件树层次结构级别的某个控件，则此顺序如下所示：

1. 转换
2. 剪切
3. 缩放
4. 围绕 z 轴旋转
5. 围绕 y 轴旋转
6. 围绕 x 轴旋转

### 12.11.9.1. 中枢

“中枢”控件功能用于定义适用于该控件的变换的中枢点。如果未配置任何中枢点，则默认中枢点位于 (0.0, 0.0, 0.-0)。

限制：

- 添加“中枢”控件特性，自动添加“旋转”、“缩放”和“剪切”控件特性。

表 12.180. “中枢”控件功能的属性

属性名称	说明
pivotX	x 轴上相当于父代控件的中枢点
pivotY	y 轴上相当于父代控件的中枢点
pivotZ	z 轴上相当于父代控件的中枢点（如果控件为场景图）

### 12.11.9.2. 旋转

“旋转”控件功能用于旋转控件及其子树。

表 12.181. “旋转”控件功能的属性

属性名称	说明
rotationEnabled	定义是否使用旋转
rotationAngleX	x 轴上的旋转角度。此属性仅影响场景图。



属性名称	说明
rotationAngleY	<b>Y</b> 轴上的旋转角度。此属性仅影响场景图。
rotationAngleZ	<b>Z</b> 轴上的旋转角度

### 12.11.9.3. 缩放

“缩放”控件特性用于缩放控件及其子树。

表 12.182. “缩放”控件功能的属性

属性名称	说明
scalingEnabled	定义是否使用缩放
scalingX	<b>x</b> 轴上的缩放（以百分比为单位）
scalingY	<b>y</b> 轴上的缩放（以百分比为单位）
scalingZ	<b>z</b> 轴上的缩放（如果控件是场景图）

### 12.11.9.4. 剪切

“剪切”控件功能用于改变控件树中控件的形状。

表 12.183. “剪切”控件功能的属性

属性名称	说明
shearingEnabled	定义是否使用剪切
shearingXbyY	<b>x</b> 轴对 <b>y</b> 轴的剪切量
shearingXbyZ	<b>x</b> 轴对 <b>z</b> 轴的剪切量（如果控件是场景图）
shearingYbyX	<b>y</b> 轴对 <b>x</b> 轴的剪切量
shearingYbyZ	<b>y</b> 轴对 <b>z</b> 轴的剪切量（如果控件是场景图）
shearingZbyX	<b>z</b> 轴对 <b>x</b> 轴的剪切量（如果控件是场景图）。
shearingZbyY	<b>z</b> 轴对 <b>y</b> 轴的剪切量（如果控件是场景图）

### 12.11.9.5. 转换

“转换”控件功能用于转换控件及其子树。它会沿 **x**、**y** 和 **z** 方向移动控件。



表 12.184. “转换”控件功能的属性

属性名称	说明
translationEnabled	定义是否使用转换
translationX	<b>x</b> 轴上的转换
translationY	<b>y</b> 轴上的转换
translationZ	<b>z</b> 轴上的转换（如果控件是场景图）

# 13. 安装 EB GUIDE Studio

## 13.1. 背景信息

### 13.1.1. 限制

注意



兼容性

EB GUIDE product line 6 与之前的所有主要版本都不兼容。

注意



**EB GUIDE Speech Extension**

EB GUIDE Speech Extension 许可为只在购买时才可启用的加载项产品。

注意



用户权限

要将 EB GUIDE 安装在 Windows 7 或 Windows 10 系统上，您需要管理员权限。

### 13.1.2. 系统要求

遵循以下设置：

表 13.1. 推荐 EB GUIDE Studio 设置

硬件	包含四核 CPU（速度不低于 2 GHz）和 8 GB RAM 的 PC
操作系统	Windows 7, Windows 10
屏幕分辨率	1920 x 1080 像素或更高 推荐使用两个单独的显示器
软件	Microsoft .NET Framework 4.7 DirectX 11



表 13.2. 推荐设置 EB GUIDE SDK

开发环境 (IDE)	Microsoft Visual Studio 2017 或更新版本
文件集成	CMake

## 13.2. 下载 EB GUIDE

要下载 EB GUIDE 社区版本，请转至 <https://www.elektrobit.com/ebguide/try-eb-guide/> 并按照说明进行操作。

要下载 EB GUIDE 企业版本，请转至 EB Command。

注意



激活您的帐户

订购产品后，您将会收到一封来自销售部的邮件。单击电子邮件中的链接。按照电子邮件中的指示，完成相应步骤以创建帐户，然后在浏览器中进行登录。

EB Command 是您将要下载 EB GUIDE product line 软件的服务器。关于如何从 EB Command 下载的说明，请参见 <https://www.elektrobit.com/support/downloading-from-eb-command/>。

## 13.3. 安装 EB GUIDE



### 安装 EB GUIDE

先决条件：

- 下载安装文件 studio\_setup.exe。
- 您在操作系统上具有管理员权限。

#### 步骤 1

双击安装文件 studio\_setup.exe。

对话框将打开。

#### 步骤 2

点击“是”。

这将打开“安装 - EB GUIDE Studio”对话框。

#### 步骤 3

接受产品协议，然后点击“下一步”。

#### 步骤 4

选中安装目录。

默认安装目录是 C:/Program Files (x86)/Elektrobit/EB GUIDE <version>。

#### 步骤 5

点击“下一步”。

此时将打开摘要对话框，其中显示了所有选定的安装设置。

#### 步骤 6

要确认使用所显示的设置进行安装，请点击“安装”。

安装开始。

#### 步骤 7

点击“完成”退出安装。

您已安装 EB GUIDE。

#### 提示



#### 多个安装版本

可以安装多个 EB GUIDE 版本。

## 13.4. 卸载 EB GUIDE



### 卸载 EB GUIDE

#### 注意



#### 永久移除 EB GUIDE

如果按照说明进行操作，您将从 PC 中永久移除 EB GUIDE。

先决条件：

- EB GUIDE 已安装。
- 您在操作系统上具有管理员权限。

#### 步骤 1

在 Windows 的“开始”菜单中，单击“所有程序”。

#### 步骤 2

在 "Elektrobit" 菜单中，单击您要卸载的版本。



步骤 3

在子菜单中，单击“卸载”。

# 术语表

## #

**3D 图形** 3D 图形是 3D 场景的虚拟图片。3D 场景是 3D 模型（网格或形状）、材料、光源和摄像头的集合。通过颜色、纹理以及虚拟照明下的行为，材料定义了 3D 模型的视觉外观。摄像头提供拍摄 3D 场景虚拟图片的视点。

## A

**API** 应用程序编程接口

## C

**通信环境** 通信环境说明了进行通信的环境。每个通信环境都由唯一的数字 ID 标识。

## D

**数据池** 数据池是 EB GUIDE 模型中的数据缓存，在运行时刻提供对数据池项的访问。它用于应用程序和 HMI 之间的数据交换。

**数据池项** 数据池项用于存储和交换数据。数据池中的每个项都包含通信指令。

## E

**EB GUIDE GTF** EB GUIDE GTF 是 EB GUIDE product line 的图形目标框架，并且是 EB GUIDE TF 的组成部分。EB GUIDE GTF 表示用于在目标设备上运行 EB GUIDE 模型的运行时间环境。

**EB GUIDE GTF SDK** EB GUIDE GTF SDK 是包含在 EB GUIDE GTF 中的开发环境。它是 EB GUIDE SDK 的子集。另一个子集是 EB GUIDE Studio SDK。

**EB GUIDE 模型** EB GUIDE 模型是对使用 EB GUIDE Studio 创建的 HMI 的说明。

**EB GUIDE product line** EB GUIDE product line 是软件库和工具的集合，这些软件库和工具是指定 HMI 模型并将其转换为嵌入式环境系统中运行的图形用户界面所需的。

**EB GUIDE 脚本** EB GUIDE 脚本是 EB GUIDE product line 的脚本语言。EB GUIDE 脚本实现了对数据池、模型元素（如，控件、状态机和系统事件）的访问。

**EB GUIDE SDK** EB GUIDE SDK 是 EB GUIDE 的产品元素。它是 EB GUIDE product line 的软件开发工具包。它包括 EB GUIDE Studio SDK 和 EB GUIDE GTF SDK。



EB GUIDE Studio	EB GUIDE Studio 是用于通过图形用户界面，进行建模和指定 HMI 的工具。
EB GUIDE Studio SDK	EB GUIDE Studio SDK 是与 EB GUIDE Studio 进行通信的应用程序编程接口（API）。它是 EB GUIDE SDK 的子集。另一个子集是 EB GUIDE GTF SDK。
EB GUIDE TF	EB GUIDE TF 是 EB GUIDE product line 的运行时间环境。它由 EB GUIDE GTF 和 EB GUIDE STF 组成。它是运行 EB GUIDE 模型所需的。

## G

GL	图形库
GUI	图形用户界面

## H

HMI	人机接口
-----	------

## L

库	库是 EB GUIDE Studio 中使用的一组资源。EB GUIDE 项目所必需的库在项目中心定义。
---	--

## M

模型元素	模型元素是 EB GUIDE 模型中的对象，例如状态、控件或数据池项。参见 EB GUIDE 模型。
------	--

## O

OS	操作系统
----	------

## P

PBR	基于物理的渲染
配置文件	在项目中心，配置文件是一组规范。在配置文件中定义项目的库、消息和场景。在 EB GUIDE 模型的导出过程中，配置文件中的数据写入到 model.json 配置文件。
项目中心	所有与项目相关的功能（例如配置文件和语言）都位于项目中心。
项目编辑器	在项目编辑器中，您可以对人机接口的行为和外观建模。

## R

**资源** 资源是作为 **EB GUIDE** 项目组成部分的数据包。资源示例包括字体、图像和网格。资源存储在 **EB GUIDE** 模型外部（例如存储在文件中），具体取决于操作系统。

## S

**共享库** 在准备执行程序时，可以加载共享库，而不是静态库。在 **Windows** 平台上，共享库称为动态链接库，并具有 `.dll` 文件扩展名。在 **Unix** 系统上，共享库称为共享对象，并具有 `.so` 文件扩展名。

**状态** 状态定义了状态机的状态。状态和状态转换在状态图中进行建模。

**状态机** 状态机是一组状态、这些状态之间的转换以及操作。状态机用于说明系统的动态行为。

## T

**转换** 转换定义了从一种状态到另一种状态的更改。转换通常由事件触发。

## U

**UI** 用户界面

## V

**视图** 视图是特定于项目的 **HMI** 屏幕的图形表示，并且与特定的状态机状态相关。视图由控件树组成。

## W

**控件** 控件是基本图形元素。控件用于与图形用户界面进行交互。

# 索引

## Symbols

.psd 文件格式, 125

3D 图形, 33, 55, 206, 312

受支持的格式, 33, 55

导入, 206

添加, 124

网格, 55

3D 对象, 33

3D 控件, 33, 55, 94

参考, 266

不透明纹理

参考, 301

中枢

参考, 305

事件, 50, 66

参考, 242

复制, 148

添加, 148

粘贴, 148

触发, 168

事件系统, 50

二指缩放操作动作

参考, 282

二次曲线

参考, 260

元素

取消对接, 45

对接, 45

光地图纹理

参考, 300

入口动画, 143

参考, 256

共享库, 314

内容区域

项目中心, 38

项目编辑器, 42

内部转换, 113

写入器应用程序, 36

列表, 151

创建, 193

数据类型, 215

列表布局

参考, 294

列表索引

参考, 296

剪切

参考, 306

动作

转换, 111

进入动作, 101

退出动作, 101

动态状态机

添加, 100, 176

动画, 35, 94, 122, 143, 199

入口动画, 36, 143

参考, 258

退出动画, 36, 143

反射纹理

参考, 302

发光纹理

参考, 299

取消对接

元素, 45

可移动的

参考, 287

命令区

项目编辑器, 44

命令行, 70, 159, 161, 168

图像

9-patch, 54

参考, 264

受支持的格式, 54

数据类型, 214

添加, 116

图标

用户界面, 70

场景图, 33, 55, 124, 206

参考, 270

添加, 124

纹理, 206

场景图节点

- 参考, 270
- 场景配置
  - 参考, 253
- 基本控件, 94
  - 参考, 256
- 复制
  - 事件, 148
  - 数据池项, 151
- 外观
  - 切换, 140
  - 删除, 140
  - 支持, 52
  - 添加, 140
- 多点触摸输入, 92
- 多点采样, 255
- 多行
  - 参考, 272
- 子代可视性选择
  - 参考, 271
- 字体, 53
  - opentype 字体, 53
  - truetype 字体, 53
  - 位图字体, 53, 53
  - 数据类型, 214
- 字符串
  - 数据类型, 216
- 定向光
  - 参考, 267
- 实例化器, 193
  - 参考, 264
  - 添加, 120
  - 行模板, 120, 264
- 容器
  - 参考, 263
  - 添加, 119
- 对接
  - 元素, 45
- 导入
  - 语言依赖项文本, 167
- 导出, 160
  - 语言依赖项文本, 165
- 导航元素
  - 项目编辑器, 39
- 导航区
  - 项目中心, 38
- 属性元素
  - 命令区, 41
  - 项目编辑器, 41
- 工具箱
  - 项目编辑器, 40
- 工具箱元素
  - 项目编辑器, 40
- 已启用
  - 参考, 272
- 已按下
  - 参考, 273
- 已聚焦
  - 参考, 272
- 已触摸
  - 参考, 275
- 已选中
  - 参考, 273
- 布尔值
  - 数据类型, 212
- 布尔值列表
  - 数据类型, 213
- 布局留白
  - 参考, 293
- 常量曲线
  - 参考, 259
- 库, 313
  - 添加, 163
- 应用程序编程接口, 36
- 弥散纹理
  - 参考, 298
- 快速启动曲线
  - 参考, 259
- 慢速启动曲线
  - 参考, 260
- 手指 ID, 92
- 按下的键
  - 参考, 284
- 按住操作动作
  - 参考, 280

- 按钮
  - 用户界面, 70
- 控件, 93, 314
  - 3D 控件, 94
  - 位置, 126
  - 删除, 126
  - 动画, 94
  - 基本, 94
  - 添加, 115
  - 组, 119
  - 调整大小, 127
- 控件功能, 95, 96, 97
  - 删除, 137
  - 添加, 135
  - 路径动作, 190
- 控件属性, 95
  - EB GUIDE 脚本, 64
  - 控件功能属性, 96
  - 控件模板, 96
  - 添加, 132
  - 用户定义的属性, 96, 132
  - 链接到控件属性, 128
  - 链接到数据池项, 130
  - 默认属性, 96
- 控件模板, 96, 144, 146
- 控件模板接口, 96
- 控制台 (见 命令行)
- 摄像头
  - 参考, 266
- 摄像头视见区
  - 参考, 297
- 操作动作, 91
  - 参考, 279, 284
  - 路径动作, 91
  - 非路径动作, 91
- 操作动作 ID
  - 参考, 281
- 放开的键
  - 参考, 285
- 效果
  - 控件功能, 276
- 数据池, 48, 312
- 数据池项, 48, 151, 312
  - 列表, 151
  - 参考, 212
  - 复制, 151
  - 导入, 167
  - 导出, 165
  - 更改, 169
  - 添加, 150
  - 窗口化列表, 49
  - 粘贴, 151
  - 语言支持, 202
  - 链接, 154
- 数据类型
  - 列表, 215
  - 图像, 214
  - 字体, 214
  - 字符串, 216
  - 布尔值, 212
  - 布尔值列表, 213
  - 整数, 214
  - 条件脚本, 213
  - 浮点值, 213
  - 网格, 212
  - 网格列表, 212
  - 颜色, 213
- 整数
  - 数据类型, 214
- 文本切断
  - 参考, 275
- 方框布局
  - 参考, 291
- 旋转
  - 参考, 274, 288, 305
- 旋转操作动作
  - 参考, 283
- 显示
  - 配置, 165
- 材料
  - PBR GGX 材料, 268, 313
  - PBR Phong 材料, 269, 313
  - 参考, 267, 268, 269
- 条件

- 转换, 110
- 条件脚本
  - 数据类型, 213
- 栅格布局
  - 参考, 293
- 标签, 118
  - 参考, 265
  - 字体, 118, 118
  - 添加, 118
- 椭圆
  - 参考, 263
- 模型元素, 49, 313
  - 删除, 106
- 模拟, 160
- 模板
  - 使用, 146
  - 创建, 144
  - 删除, 146
- 模板接口, 145
  - 删除属性, 145
  - 添加属性, 145
- 模板索引
  - 参考, 296
- 正常贴图纹理
  - 参考, 301
- 正弦曲线
  - 参考, 261
- 流式布局
  - 参考, 292
- 浮点值
  - 数据类型, 213
- 渲染器
  - 配置, 165
- 滑动操作动作
  - 参考, 279
- 点光
  - 参考, 269
- 状态, 75, 102, 103, 180, 314
  - 初始状态, 76
  - 历史状态, 79
  - 组合状态, 75
  - 结束状态, 77
  - 视图状态, 76
  - 转换, 107
  - 进入动作, 105
  - 退出动作, 106
  - 选择状态, 78
- 状态机, 74, 314
  - UML 2.5 标记, 90
  - 与 UML 相比, 90
  - 内含状态机, 74, 91
  - 删除, 102
  - 动态状态机, 74
  - 添加, 100
  - 状态, 75
  - 状态机的执行, 86
  - 触觉状态机, 74
  - 转换, 82
  - 逻辑状态机, 74
- 环境光
  - 参考, 266
- 环境纹理
  - 参考, 297
- 用户定义焦点
  - 参考, 278
- 用户定义的属性, 132
- 着色
  - 参考, 277
- 矩形
  - 参考, 265
- 移入
  - 参考, 286
- 移出
  - 参考, 286
- 移过
  - 参考, 287
- 窗口化列表
  - 数据池项, 49
- 笔划
  - 参考, 277
- 粘贴
  - 事件, 148
  - 数据池项, 151
- 线性插值曲线, 262

- 线性曲线, 262
- 组合状态, 103
- 绝对布局
  - 参考, 291
- 缩放
  - 参考, 306
- 缩放模式
  - 参考, 295
- 网格, 55
  - 参考, 267
  - 数据类型, 212
- 网格列表
  - 数据类型, 212
- 聚光
  - 参考, 271
- 脚本化的值, 68, 152
- 脚本曲线, 261
- 自动焦点
  - 参考, 278
- 自动隐藏, 45
- 色调映射
  - 参考, 304
- 行索引
  - 参考, 296
- 视图, 93, 314
  - 参考, 256
  - 添加, 114
- 视图模板
  - 参考, 256, 256
- 触发器
  - 转换, 109
- 触摸丢失
  - 参考, 288
- 触摸已按下
  - 参考, 289
- 触摸已放开
  - 参考, 289
- 触摸操作动作 (见 操作动作)
- 触摸状态已变化
  - 参考, 290
- 触摸移动
  - 参考, 288
- 触摸输入 (见 操作动作)
- 语言
  - 更改, 202
- 语言依赖项文本, 202
  - 导入, 167
  - 导出, 165
- 读取器应用程序, 36
- 资源, 314
  - .psd 文件格式, 55
  - 3D 图形, 55
  - 图像, 54
  - 字体, 53
  - 网格, 55
- 资源管理, 53
- 路径动作, 190
  - 参考, 281, 281
- 转换, 82, 107, 314
  - 内部, 113
  - 动作, 111
  - 参考, 306
  - 条件, 110
  - 添加, 107
  - 移动, 108
  - 触发器, 109
- 边框
  - 参考, 276
- 进入动作, 105
  - 状态机, 101
- 退出动作, 106
  - 状态机, 101
- 退出动画, 143
  - 参考, 256
- 选择状态, 104
- 选择群组
  - 参考, 274
- 通信环境, 36, 152, 312
- 配置
  - 显示, 165
- 配置文件, 162, 242, 250, 313
  - 克隆, 162
  - 添加, 162
- 重命名全局, 158

## 链接

控件属性, 129, 130

数据池项, 154

## 键状态已变化

参考, 285

## 键盘快捷键

用户界面, 70

## 镜面纹理

参考, 303

## 长按操作动作

参考, 280

## 问题元素, 158

项目编辑器, 45

## 项目中心, 37, 313

内容区域, 38

导航区, 38

## 项目编辑器, 38, 313

内容区域, 42

命令区, 44

导航元素, 39

工具箱, 40

工具箱元素, 40

问题元素, 45

## 颜色

数据类型, 213

**A**

## alpha 掩码, 123

参考, 257

API, 312 (见 应用程序编程接口)

**E**

EB GUIDE GTF, 312

EB GUIDE GTF SDK, 312

EB GUIDE

tor, 46, 160, 168, 168, 169, 170, 174

事件, 168

事件元素, 168

命令行, 174

数据池元素, 169

数据池项, 169

标签, 46

独立, 174

脚本元素, 170

EB GUIDE product line, 312

EB GUIDE SDK, 312

EB GUIDE Studio, 312

EB GUIDE Studio SDK;, 312

EB GUIDE TF, 312

EB GUIDE 扩展, 50

EB GUIDE 模型, 49, 312

模型元素, 49

EB GUIDE 脚本, 55, 151, 312

if-then-else, 62

While 循环, 61

事件, 66

列表, 65

右值, 60

命名空间, 56

外部函数调用, 63

字符串格式设置, 68

局部变量, 60

左值, 60

控件属性, 64

教程, 184

数据池访问, 64

标准库, 68

注释, 57

类型, 57

脚本化的值, 69

表达式, 58

识别符, 56

EB GUIDE 项目, 49

**G**

GL, 313

GUI, 313

**H**

HMI, 313

**L**

linear interpolation integer, 200

Moni-



**M**

model.json, 242

**O**

OS, 313

**P**

platform.json, 250

**T**

todo

EB GUIDE 脚本, 57

**U**

UI, 314

Unicode 键

参考, 285

**V**

Viewport

参考, 296

VTA, 143