



Elektrobit

EB GUIDE Studio

用户指南

版本 6.9.0.200120181101



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.

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

Copyright 2019, Elektrobit Automotive GmbH.

目录

1. 关于本文档	18
1.1. 用户文档的目标群体	18
1.1.1. 目标群体：建模员	18
1.1.2. 目标群体：扩展开发人员	19
1.2. 用户文档的结构	19
1.3. 命名约定	20
1.4. 路径约定	20
2. 安全和正确使用	21
2.1. 预期用途	21
2.2. 可能的误用	21
3. 支持	22
4. EB GUIDE 简介	23
4.1. EB GUIDE product line	23
4.2. EB GUIDE Studio	23
4.2.1. 对 HMI 行为进行建模	23
4.2.2. 对 HMI 外观进行建模	24
4.2.3. 处理数据	24
4.2.4. 模拟 EB GUIDE 模型	24
4.2.5. 导出 EB GUIDE 模型	24
4.3. EB GUIDE TF	25
4.4. EB GUIDE aware	25
5. 教程：入门	27
5.1. 启动 EB GUIDE	27
5.2. 创建项目	28
5.3. 对 HMI 行为进行建模	29
5.4. 对 HMI 外观进行建模	32
5.5. 启动模拟运行	35
6. 背景信息	36
6.1. 3D 图形	36
6.1.1. 受支持的 3D 图形格式	36
6.1.2. 3D 图形文件的设置	36
6.1.3. 导入 3D 图形文件	37
6.2. 动画	38
6.2.1. 控件动画	39
6.2.2. 视图转变的动画	39
6.2.3. 脚本曲线	40
6.3. 抗锯齿	40
6.4. 应用程序和模型之间的应用程序编程接口	40
6.5. 通信环境	41

6.6. 图形用户界面元素	41
6.6.1. EB GUIDE Studio 的图形用户界面	41
6.6.1.1. 项目中心	41
6.6.1.1.1. 导航区	42
6.6.1.1.2. 内容区域	43
6.6.1.2. 项目编辑器	43
6.6.1.2.1. 导航元素	44
6.6.1.2.2. “概要”元素	45
6.6.1.2.3. 工具箱元素	45
6.6.1.2.4. 属性元素	46
6.6.1.2.5. 内容区域	47
6.6.1.2.6. “事件”元素	48
6.6.1.2.7. “数据池”元素	48
6.6.1.2.8. “资产”元素	49
6.6.1.2.9. “命名空间”元素	49
6.6.1.2.10. 命令区	50
6.6.1.2.11. 问题元素	51
6.6.1.2.12. VTA 元素	51
6.6.1.2.13. 模板元素	51
6.6.2. EB GUIDE Monitor 的图形用户界面	51
6.6.3. 可对接元素	53
6.7. 数据池	54
6.7.1. 概念	54
6.7.2. 数据池项	54
6.7.3. 窗口化列表	54
6.8. EB GUIDE 模型和 EB GUIDE 项目	55
6.8.1. 存储格式	56
6.8.2. EB GUIDE 项目的验证标准	57
6.8.2.1. 打开 EB GUIDE 项目时进行验证	58
6.8.2.2. 使用“问题”元素进行验证	58
6.9. EB GUIDE Monitor	58
6.10. 事件处理	59
6.10.1. 事件系统	59
6.10.2. 事件	59
6.11. 扩展	62
6.11.1. EB GUIDE Studio 扩展	62
6.11.2. EB GUIDE GTF 扩展	62
6.11.3. EB GUIDE Monitor 扩展	62
6.12. 伽马校正渲染	63
6.12.1. 概念	63
6.12.2. EB GUIDE Studio 中的伽马校正	64
6.13. 基于图像的光照	64

6.13.1. IBLGenerator, 文件格式和导入	65
6.13.2. 使用 OpenGL 渲染器的 IBL 限制	65
6.14. 语言	65
6.14.1. EB GUIDE Studio 中的显示语言	65
6.14.2. EB GUIDE 模型中的语言	66
6.14.3. 导出和导入语言依赖项文本	66
6.15. 命名空间	67
6.16. 模型接口	68
6.16.1. 数据池项导入	68
6.16.2. 事件导入	68
6.16.3. 事件组导入	69
6.16.4. 命名空间导入	69
6.17. Photoshop 文件格式支持	69
6.18. 资源管理	70
6.18.1. 字体	71
6.18.1.1. 位图字体	71
6.18.1.2. 多字体支持	71
6.18.2. 3D 图形的基于图像的光照	72
6.18.3. 图像	72
6.18.3.1. 9-patch 图像	72
6.18.4. 3D 图形的网格	73
6.19. 脚本语言 EB GUIDE 脚本	74
6.19.1. 应用程序的功能和区域	74
6.19.2. 前缀和识别符	74
6.19.3. 注释	75
6.19.4. 类型	75
6.19.5. 表达式	76
6.19.6. 常量和引用	76
6.19.7. 算术和逻辑表达式	77
6.19.8. 左值和右值	78
6.19.9. 局部变量	78
6.19.10. While 循环	79
6.19.11. If-then-else	80
6.19.12. 外部函数调用	81
6.19.13. 数据池访问	82
6.19.14. 控件属性	83
6.19.15. 列表	84
6.19.16. 事件	84
6.19.17. 字符串格式设置	86
6.19.18. 标准库	86
6.20. 脚本化的值	87
6.21. 外观	89

6.22. 状态机和状态	90
6.22.1. 状态机	90
6.22.1.1. 触觉状态机	90
6.22.1.2. 逻辑状态机	90
6.22.1.3. 动态状态机	90
6.22.2. 状态	91
6.22.2.1. 组合状态	91
6.22.2.2. 视图状态	92
6.22.2.3. 初始状态	92
6.22.2.4. 结束状态	93
6.22.2.5. 选择状态	94
6.22.2.6. 历史状态	95
6.22.3. 转换	98
6.22.4. 状态机的执行	101
6.22.5. 与 UML 标记相比的 EB GUIDE 标记	105
6.22.5.1. 受支持的元素	105
6.22.5.2. 不受支持的元素	105
6.22.5.3. 偏差	105
6.23. 触摸输入	106
6.23.1. 非路径动作	106
6.23.2. 路径动作	106
6.23.3. 输入处理和操作动作	107
6.23.4. 多点触摸输入	107
6.24. 控件	108
6.24.1. 视图	108
6.24.2. 控件类别	109
6.24.3. 控件属性	111
6.24.4. 控件模板	112
6.24.5. 控件功能	112
6.24.5.1. 焦点控件功能类别	113
6.24.5.2. 列表管理控件功能类别	115
7. 对 HMI 行为进行建模	116
7.1. 对状态机进行建模	116
7.1.1. 添加状态机	116
7.1.2. 添加动态状态机	116
7.1.3. 为状态机定义进入动作	117
7.1.4. 为状态机定义退出动作	117
7.1.5. 删除状态机	118
7.2. 对状态进行建模	118
7.2.1. 添加状态	118
7.2.2. 向组合状态添加状态	119
7.2.3. 添加选择状态	120

7.2.4. 为状态定义进入动作	121
7.2.5. 为状态定义退出动作	122
7.2.6. 从状态机中删除模型元素	123
7.3. 通过转换连接状态	123
7.3.1. 在两种状态之间添加转换	123
7.3.2. 移动转换	124
7.3.3. 为转换定义触发器	125
7.3.4. 向转换添加条件	126
7.3.5. 向转换添加操作	127
7.3.6. 向状态添加内部转换	128
8. 对 HMI 外观进行建模	130
8.1. 使用控件	130
8.1.1. 添加视图	130
8.1.2. 向视图添加基本控件	131
8.1.2.1. 添加矩形	131
8.1.2.2. 添加椭圆	131
8.1.2.2.1. 编辑椭圆	131
8.1.2.3. 添加图像	132
8.1.2.4. 添加一个标签	134
8.1.2.5. 添加容器	135
8.1.2.6. 添加实例化器	135
8.1.2.7. 添加动画	137
8.1.2.8. 添加带有脚本曲线的动画	139
8.1.2.9. 添加 alpha 掩码	140
8.1.3. 向一个 3D 控件到视图	141
8.1.3.1. 向视图添加场景图	141
8.1.4. 导入 .psd 文件到视图中	142
8.1.5. 从 .psd 文件中提取图像	143
8.1.6. 导入 IBL 文件	144
8.1.7. 从视图中删除控件	145
8.2. 使用控件属性	145
8.2.1. 确定控件的位置	145
8.2.2. 调整控件的大小	146
8.2.3. 控件属性之间的链接	147
8.2.4. 将控件属性链接到数据池项	149
8.2.5. 向控件添加用户定义的属性	150
8.2.5.1. 添加用户定义的以下类型的属性 Function (): bool	151
8.2.6. 重命名用户定义的属性	152
8.2.7. 编辑类型列表属性	153
8.2.8. 管理控件的顺序和可视性	154
8.3. 按控件功能扩展控件	155
8.3.1. 添加控件功能	155

8.3.2. 删除控件功能	157
8.4. 更改字体设置	158
8.4.1. 更改标签的字体	158
8.4.2. 更改行间距	159
8.4.2.1. 更改默认行间距	160
8.4.2.2. 更改多行的行间距	161
8.4.3. 管理多字体支持	161
8.5. 使用语言支持	164
8.5.1. 向 EB GUIDE 模型中添加语言	164
8.5.2. 向数据池项添加语言支持	165
8.5.3. 删除语言	165
8.6. 使用语言支持	166
8.6.1. 向 EB GUIDE 模型中添加外观	166
8.6.2. 向数据池项添加外观支持	167
8.6.3. 外观间转换	168
8.6.4. 删除外观	168
8.7. 给视图转变制作动画	169
8.7.1. 添加入口动画	169
8.7.2. 添加更改动画	170
8.7.3. 重新排列动画	170
8.8. 重复使用控件	171
8.8.1. 添加模板	171
8.8.2. 定义模板接口	172
8.8.3. 使用模板	173
8.8.4. 删除模板	173
8.9. 启用抗锯齿	174
8.9.1. 全局启用抗锯齿	174
8.9.2. 为场景图启用抗锯齿	174
9. 处理数据	176
9.1. 添加事件	176
9.2. 向事件添加参数	176
9.3. 确定事件位置	177
9.4. 将键映射到事件	178
9.5. 删除事件	179
9.6. 添加数据池项	179
9.7. 编辑列表类型的数据池项	180
9.8. 将属性转换为脚本化的值	181
9.9. 建立外部通信	182
9.10. 数据池项之间的链接	183
9.11. 删除数据池项	184
9.12. 将模型元素添加到模型接口	185
9.12.1. 将事件添加到模型接口	185

9.12.2. 将数据池项添加到模型接口	186
9.13. 使用命名空间	186
9.13.1. 添加命名空间	186
9.13.2. 将模型元素添加到命名空间	187
9.13.3. 在命名空间之间移动模型元素	187
9.13.4. 删除命名空间	188
10. 处理项目	189
10.1. 创建项目	189
10.2. 打开项目	189
10.2.1. 从文件资源管理器中打开项目	190
10.2.2. 在 EB GUIDE Studio 中打开项目	190
10.3. 重命名模型元素	191
10.4. 验证和模拟运行 EB GUIDE 模型	191
10.4.1. 验证 EB GUIDE 模型	192
10.4.1.1. 在 EB GUIDE Studio 验证 EB GUIDE 模型	192
10.4.1.2. 使用命令行验证 EB GUIDE 模型	193
10.4.2. 启动和停止模拟运行	193
10.5. 导出 EB GUIDE 模型	193
10.5.1. 使用 EB GUIDE Studio 导出 EB GUIDE 模型	193
10.5.2. 使用命令行导出 EB GUIDE 模型	194
10.6. 更改 EB GUIDE Studio 的显示语言	194
10.7. 配置文件	195
10.7.1. 添加配置文件	195
10.7.2. 添加库	196
10.7.3. 配置场景	198
10.8. 导出和导入语言依赖项文本	199
10.8.1. 导出语言依赖项文本	199
10.8.2. 导入语言依赖项文本	200
10.8.2.1. 使用 EB GUIDE Studio 导入语言依赖项文本	200
10.8.2.2. 使用命令行导入语言依赖项文本	201
10.9. 使用模型接口	202
10.9.1. 创建模型接口	202
10.9.2. 导出模型接口	202
10.9.3. 导入模型接口	203
10.9.4. 更新导入的模型接口	204
10.9.5. 删除模型接口	204
11. 使用 EB GUIDE Monitor	206
11.1. 作为独立程序启动 EB GUIDE Monitor	206
11.2. 配置 EB GUIDE Monitor	207
11.3. 加载配置到 EB GUIDE Monitor	209
11.4. 在 EB GUIDE Monitor 中触发事件	210
11.5. 使用 EB GUIDE Monitor 更改数据池项的值	211

11.6. 在 EB GUIDE Monitor 中使用脚本	211
11.6.1. 为 EB GUIDE Monitor 编写脚本文件	211
11.6.2. 在 EB GUIDE Monitor 中启动脚本	216
11.7. 导出和导入观察列表	217
12. 扩展 EB GUIDE Studio	219
12.1. 概念	219
12.1.1. 依赖注入	219
12.1.2. EB GUIDE 模型扩展	219
12.1.3. EB GUIDE Studio 用户界面扩展	221
12.2. 创建扩展项目	224
12.3. 禁用程序集的复制	225
12.4. 运行扩展	225
13. 最佳实践	227
13.1. 最佳实践：处理脚本化的值	227
14. 教程	228
14.1. 教程：添加动态状态机	228
14.2. 教程：使用 EB GUIDE 脚本 对按钮行为进行建模	236
14.3. 教程：对路径动作进行建模	243
14.4. 教程：使用动态内容创建 列表	246
14.5. 教程：使椭圆移动经过屏幕	252
14.6. 教程：向数据池项添加语言依赖项文本	255
14.7. 教程：使用 3D 图形	259
14.8. 教程：正确渲染伽马	263
14.9. 教程：使用视图切换动画	266
14.10. 教程：使用脚本曲线制作动画	274
14.11. 教程：创建水平进度条	279
15. 参考	284
15.1. 命令行选项	284
15.1.1. 命令行选项 Studio.Console.exe	284
15.1.2. 命令行选项 Monitor.Console.exe	285
15.2. 数据池项	285
15.3. 数据类型	286
15.3.1. 布尔值	286
15.3.2. 颜色	286
15.3.3. 条件脚本	286
15.3.4. 浮点值	287
15.3.5. 字体	288
15.3.6. Function () : bool	288
15.3.7. lbl	288
15.3.8. 图像	288
15.3.9. 整数	289
15.3.10. 网格	289

15.3.11. 字符串	290
15.3.12. 列表	290
15.4. EB GUIDE 脚本	291
15.4.1. EB GUIDE 脚本 关键字	291
15.4.2. EB GUIDE 脚本 运算符优先顺序	292
15.4.3. EB GUIDE 脚本 标准库	293
15.4.3.1. EB GUIDE 脚本 函数 A - B	293
15.4.3.1.1. abs	293
15.4.3.1.2. absf	293
15.4.3.1.3. acosf	293
15.4.3.1.4. animation_before	294
15.4.3.1.5. animation_beyond	294
15.4.3.1.6. animation_cancel	294
15.4.3.1.7. animation_cancel_end	295
15.4.3.1.8. animation_cancel_reset	295
15.4.3.1.9. animation_pause	295
15.4.3.1.10. animation_play	295
15.4.3.1.11. animation_reverse	296
15.4.3.1.12. animation_running	296
15.4.3.1.13. animation_set_time	296
15.4.3.1.14. asinf	296
15.4.3.1.15. atan2f	297
15.4.3.1.16. atan2i	297
15.4.3.1.17. atanf	297
15.4.3.1.18. bool2string	298
15.4.3.2. EB GUIDE 脚本 函数 C - H	298
15.4.3.2.1. ceil	298
15.4.3.2.2. changeDynamicStateMachinePriority	298
15.4.3.2.3. character2unicode	299
15.4.3.2.4. clampf	299
15.4.3.2.5. clampi	299
15.4.3.2.6. clearAllDynamicStateMachines	299
15.4.3.2.7. color2string	300
15.4.3.2.8. cosf	300
15.4.3.2.9. deg2rad	300
15.4.3.2.10. expf	301
15.4.3.2.11. float2string	301
15.4.3.2.12. floor	301
15.4.3.2.13. fmod	301

15.4.3.2.14. focusMoveTo	302
15.4.3.2.15. focusNext	302
15.4.3.2.16. focusPrevious	302
15.4.3.2.17. format_float	302
15.4.3.2.18. format_int	303
15.4.3.2.19. frac	304
15.4.3.2.20. getAllLanguages	304
15.4.3.2.21. getAllSkins	305
15.4.3.2.22. getConfigItem	305
15.4.3.2.23. getFontAscender	305
15.4.3.2.24. getFontDescender	306
15.4.3.2.25. getFontLineGap	306
15.4.3.2.26. getImageHeight	306
15.4.3.2.27. getImageWidth	306
15.4.3.2.28. getLabelTextHeight	307
15.4.3.2.29. getLabelTextWidth	307
15.4.3.2.30. getLanguage	307
15.4.3.2.31. getLanguageName	308
15.4.3.2.32. getLanguageTag	308
15.4.3.2.33. getLineCount	308
15.4.3.2.34. getLineHeight	308
15.4.3.2.35. getProductString	309
15.4.3.2.36. getSkin	309
15.4.3.2.37. getSkinName	309
15.4.3.2.38. getTextHeight	309
15.4.3.2.39. getTextLength	310
15.4.3.2.40. getTextWidth	310
15.4.3.2.41. getVersionString	311
15.4.3.2.42. has_list_window	311
15.4.3.2.43. hsba2color	311
15.4.3.3. EB GUIDE 脚本 函数 I - R	312
15.4.3.3.1. int2float	312
15.4.3.3.2. int2string	312
15.4.3.3.3. isDynamicStateMachineActive	312
15.4.3.3.4. isWidgetOnActiveStatemachine	313
15.4.3.3.5. language	313
15.4.3.3.6. lerp	313
15.4.3.3.7. localtime_day	314
15.4.3.3.8. localtime_hour	314

15.4.3.3.9. localtime_minute	314
15.4.3.3.10. localtime_month	314
15.4.3.3.11. localtime_second	315
15.4.3.3.12. localtime_weekday	315
15.4.3.3.13. localtime_year	315
15.4.3.3.14. log10f	315
15.4.3.3.15. logf	316
15.4.3.3.16. maxf	316
15.4.3.3.17. maxi	316
15.4.3.3.18. minf	317
15.4.3.3.19. mini	317
15.4.3.3.20. nearbyint	317
15.4.3.3.21. popDynamicStateMachine	317
15.4.3.3.22. powf	318
15.4.3.3.23. pushDynamicStateMachine	318
15.4.3.3.24. rad2deg	318
15.4.3.3.25. rand	318
15.4.3.3.26. rgba2color	319
15.4.3.3.27. round	319
15.4.3.4. EB GUIDE 脚本 函数 S - W	319
15.4.3.4.1. saturate	319
15.4.3.4.2. seed_rand	320
15.4.3.4.3. setLanguage	320
15.4.3.4.4. setSkin	320
15.4.3.4.5. shutdown	321
15.4.3.4.6. sinf	321
15.4.3.4.7. skin	321
15.4.3.4.8. smoothstep	321
15.4.3.4.9. sqrtf	322
15.4.3.4.10. string2float	322
15.4.3.4.11. string2int	323
15.4.3.4.12. string2string	323
15.4.3.4.13. substring	323
15.4.3.4.14. system_time	324
15.4.3.4.15. system_time_ms	324
15.4.3.4.16. tanf	324
15.4.3.4.17. trace_dp	324
15.4.3.4.18. trace_string	325
15.4.3.4.19. transformToScreenX	325

15.4.3.4.20. transformToScreenY	325
15.4.3.4.21. transformToWidgetX	326
15.4.3.4.22. transformToWidgetY	326
15.4.3.4.23. trunc	326
15.4.3.4.24. widgetGetChildCount	327
15.5. 事件	327
15.5.1. 键事件的十进制代码	327
15.6. 按钮和图标	329
15.7. 场景	334
15.8. 键盘快捷键	336
15.9. 控件	338
15.9.1. 视图	338
15.9.2. 基本控件	339
15.9.2.1. Alpha 掩码	339
15.9.2.2. 动画	340
15.9.2.2.1. 常量曲线	341
15.9.2.2.2. 快速启动曲线	341
15.9.2.2.3. 慢速启动曲线	342
15.9.2.2.4. 二次曲线	342
15.9.2.2.5. 正弦曲线	343
15.9.2.2.6. 脚本曲线	343
15.9.2.2.7. 线性曲线	344
15.9.2.2.8. 线性插值曲线	345
15.9.2.3. 容器	345
15.9.2.4. 椭圆	345
15.9.2.5. 图像	346
15.9.2.6. 实例化器	347
15.9.2.7. 标签	347
15.9.2.8. 矩形	348
15.9.3. 3D 控件	348
15.9.3.1. 环境光	348
15.9.3.2. 摄像头	349
15.9.3.3. 定向光	349
15.9.3.4. 基于图像的光	349
15.9.3.5. 材料	350
15.9.3.6. 网格	350
15.9.3.7. PBR GGX 材料	351
15.9.3.8. PBR Phong 材料	352
15.9.3.9. 点光	353
15.9.3.10. 场景图	353
15.9.3.11. 场景图节点	354
15.9.3.12. 聚光	354

15.10. 控件功能	355
15.10.1. 公共	355
15.10.1.1. 子代可视性选择	355
15.10.1.2. 已启用	356
15.10.1.3. 已聚焦	356
15.10.1.4. 字体指标	356
15.10.1.5. 多行	357
15.10.1.6. 已按下	357
15.10.1.7. 已选中	358
15.10.1.8. 选择群组	358
15.10.1.9. 旋转	359
15.10.1.10. 文本切断	359
15.10.1.11. 已触摸	360
15.10.2. 效果	361
15.10.2.1. 边框	361
15.10.2.2. 着色	362
15.10.2.3. 笔划	362
15.10.3. 焦点	362
15.10.3.1. 自动焦点	363
15.10.3.2. 用户定义焦点	363
15.10.4. 操作动作	364
15.10.4.1. 滑动操作动作	364
15.10.4.2. 按住操作动作	365
15.10.4.3. 长按操作动作	365
15.10.4.4. 路径动作	366
15.10.4.4.1. 操作动作 ID	367
15.10.4.5. 二指缩放操作动作	368
15.10.4.6. 旋转操作动作	369
15.10.5. 输入处理	370
15.10.5.1. 操作动作	370
15.10.5.2. 按下的键	370
15.10.5.3. 放开的键	370
15.10.5.4. 键状态已变化	371
15.10.5.5. Unicode 键	371
15.10.5.6. 移入	372
15.10.5.7. 移出	372
15.10.5.8. 移过	373
15.10.5.9. 可移动的	373
15.10.5.10. 旋转	374
15.10.5.11. 触摸丢失	374
15.10.5.12. 触摸移动	375
15.10.5.13. 触摸已按下	375

15.10.5.14. 触摸已放开	376
15.10.5.15. 触摸状态已变化	376
15.10.6. 布局	377
15.10.6.1. 绝对布局	377
15.10.6.2. 方框布局	378
15.10.6.3. 流式布局	379
15.10.6.4. 栅格布局	380
15.10.6.5. 布局留白	380
15.10.6.6. 列表布局	381
15.10.6.7. 缩放模式	382
15.10.7. 列表管理	383
15.10.7.1. 行索引	383
15.10.7.2. 列表索引	383
15.10.7.3. 模板索引	383
15.10.7.4. Viewport	384
15.10.8. 3D	384
15.10.8.1. 抗锯齿模式	384
15.10.8.2. 摄像头视见区	385
15.10.8.3. 透明涂层	385
15.10.8.4. 环境纹理	385
15.10.8.5. 弥散纹理	386
15.10.8.6. 放射纹理	387
15.10.8.7. 光地图纹理	388
15.10.8.8. 金属纹理	389
15.10.8.9. 正常贴图纹理	390
15.10.8.10. 不透明纹理	391
15.10.8.11. 反射纹理	392
15.10.8.12. 粗糙纹理	393
15.10.8.13. 光亮纹理	394
15.10.8.14. 镜面纹理	395
15.10.8.15. Texture coordinate transformation	396
15.10.8.16. 色调映射	397
15.10.8.17. 摄像头光晕	398
15.10.8.18. 景深	399
15.10.8.19. 屏幕空间环境光遮挡	399
15.10.9. 变换	401
15.10.9.1. 中枢	401
15.10.9.2. 旋转	402
15.10.9.3. 缩放	402
15.10.9.4. 剪切	402
15.10.9.5. 转换	403
16. EB GUIDE Studio 的安装	404



16.1. 背景信息	404
16.1.1. 限制	404
16.1.2. 系统要求	404
16.2. 下载 EB GUIDE	405
16.3. 安装 EB GUIDE	405
16.4. 卸载 EB GUIDE	406
术语表	407
索引	412

1. 关于本文档

1.1. 用户文档的目标群体

本章将告诉您 EB GUIDE 项目中涉及的目标群体以及他们通常执行的的任务的相关信息。

您可以对任务进行分类并查找与自己相关的文档。

存在下列角色：

- ▶ [节1.1.1](#)
- ▶ [节1.1.2](#)

1.1.1. 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 特性
- ▶ 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）

1.1.2. 目标群体：扩展开发人员

可能存在无法通过简单地对 EB GUIDE 模型进行建模或添加特定于客户的应用程序来提供的缺失功能。需要新控件或特定渲染器时，可能会出现这种情况。

扩展开发人员执行下列任务：

- ▶ 通过 [章3](#) 与 EB GUIDE 开发团队的成员进行沟通，确定是否已有问题的解决方案
- ▶ 处理框架并开发新特性、EB GUIDE Studio 扩展或 EB GUIDE GTF 扩展
- ▶ 为下列项的附加模块编写代码：
 - ▶ 现有 EB GUIDE GTF 模块，如控件或着色器
 - ▶ 现有 EB GUIDE Studio 扩展，例如附加工具栏按钮

扩展开发人员具备以下方面的渊博知识：

- ▶ EB GUIDE 接口
- ▶ 与中央模块的交互
- ▶ 框架数据的结构

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。
- ▶ 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. 命名约定

在 EB GUIDE 文档中，使用以下文件夹名称：

- ▶ EB GUIDE 的安装文件夹称为 \$GUIDE_INSTALL_PATH。

例如：

C:/Program Files/Elektrobit/EB GUIDE Studio 6.9

- ▶ EB GUIDE SDK 平台的文件夹称为 \$GTF_INSTALL_PATH。名称模式为 \$GTF_INSTALL_PATH/platform/<platform name>。

例如：

C:/Program Files/Elektrobit/EB GUIDE Studio 6.9/platform/win64

- ▶ 保存 EB GUIDE 项目的文件夹称为 \$GUIDE_PROJECT_PATH。

例如：

C:/Users/[user name]/Documents/EB GUIDE 6.9/projects/

- ▶ 导出 EB GUIDE 模型的文件夹称作 \$EXPORT_PATH。

例如：

C:/Documents/Projects/My_exported_model

1.4. 路径约定

在 Windows 10 中 EB GUIDE Studio 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



Windows 7 中的长路径名

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. 安全和正确使用

2.1. 预期用途

- ▶ EB GUIDE Studio 和 EB GUIDE GTF 旨在用于娱乐信息头单元、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 并非用于需要 DO-178B、SIL 或 A-SIL 之类的特定认证的安全相关系统。

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. EB GUIDE 简介

EB GUIDE 可以在人机接口 (HMI) 的开发过程中为用户提供帮助。EB GUIDE 产品线提供用于图形用户界面的工具和平台。EB GUIDE 产品线旨在用于娱乐信息头单元、组合仪表及所选行业应用。主要用例包括大规模生产、说明和原型开发。

4.1. EB GUIDE product line

EB GUIDE product line 包含下列软件部件：

- ▶ EB GUIDE Studio
- ▶ EB GUIDE TF
- ▶ EB GUIDE aware

EB GUIDE Studio 是 PC 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。

EB GUIDE aware 是一个软件框架，可用于创建增强现实解决方案以增强驾驶体验。

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。

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。

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。可以将事件同时发送给 HMI 和应用程序，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。

EB GUIDE GTF (EB GUIDE Graphics Target Framework) 是执行图形 HMI 的运行时间环境。

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 使用下列平台抽象层：

- ▶ 操作系统抽象层

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 支持适用于不同平台的各种渲染器。

4.4. EB GUIDE arware

EB GUIDE arware 是一个软件框架，可用于创建增强现实解决方案，以提高驾驶安全性，建立驾驶员对车辆的信任并增强驾驶体验。EB GUIDE arware 使用汽车的 GPS（全球定位系统）和传感器来获取和识别汽车环境中的物体。使用 EB GUIDE arware，您可以在平视显示器上的视野内实时唤起驾驶员注意这些对象。

EB GUIDE arware 可以处理以下信息：

- ▶ 车辆传感器提供的信息，例如来自 ADAS 电子控制单元的对象数据
- ▶ 地图和向导信息提供者提供的信息，例如 ADASISv3 数据
- ▶ 由 EB GUIDE arware 计算的信息来自上述来源，例如通过融合地图和传感器数据
- ▶ 车载信息娱乐系统通过 EB GUIDE Studio 界面提供的信息

EB GUIDE arware 包含两个主要软件部分：

- ▶ 数据融合可创建车辆环境的虚拟模型，关联来自不同传感器的测量结果，并外推以补偿延迟。
- ▶ 可视化子系统决定显示哪些用户界面元素，计算这些元素在何处显示，并使用主机 OS 图形子系统呈现它们。



可以使用 EB GUIDE Studio 和 EB GUIDE GTF 扩展和自定义 EB GUIDE aware 的可视化子系统。

EB GUIDE aware 必须集成到特定于车辆的应用程序中。该应用程序负责将传入的数据消息转换为可由 EB GUIDE aware 处理的标准化格式。

5. 教程：入门

注意



默认窗口布局

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. 启动 EB GUIDE



启动 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 将启动。将显示项目中心。

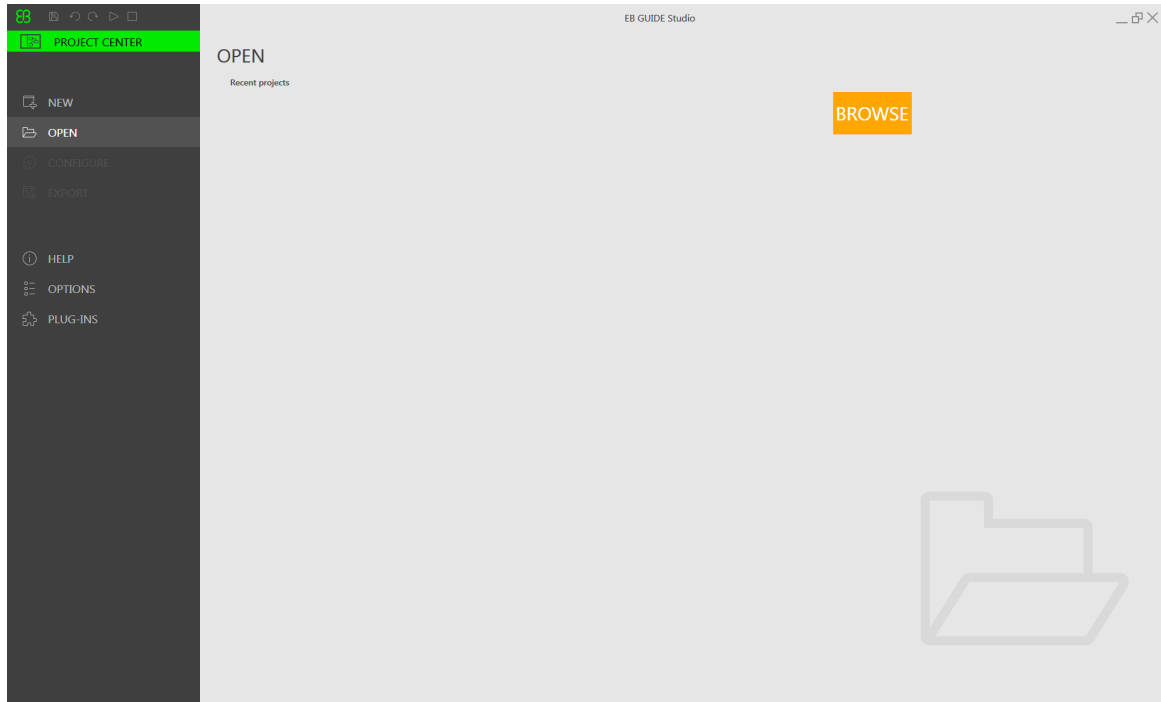


图 5.1. 项目中心

5.2. 创建项目



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 文件夹。

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。

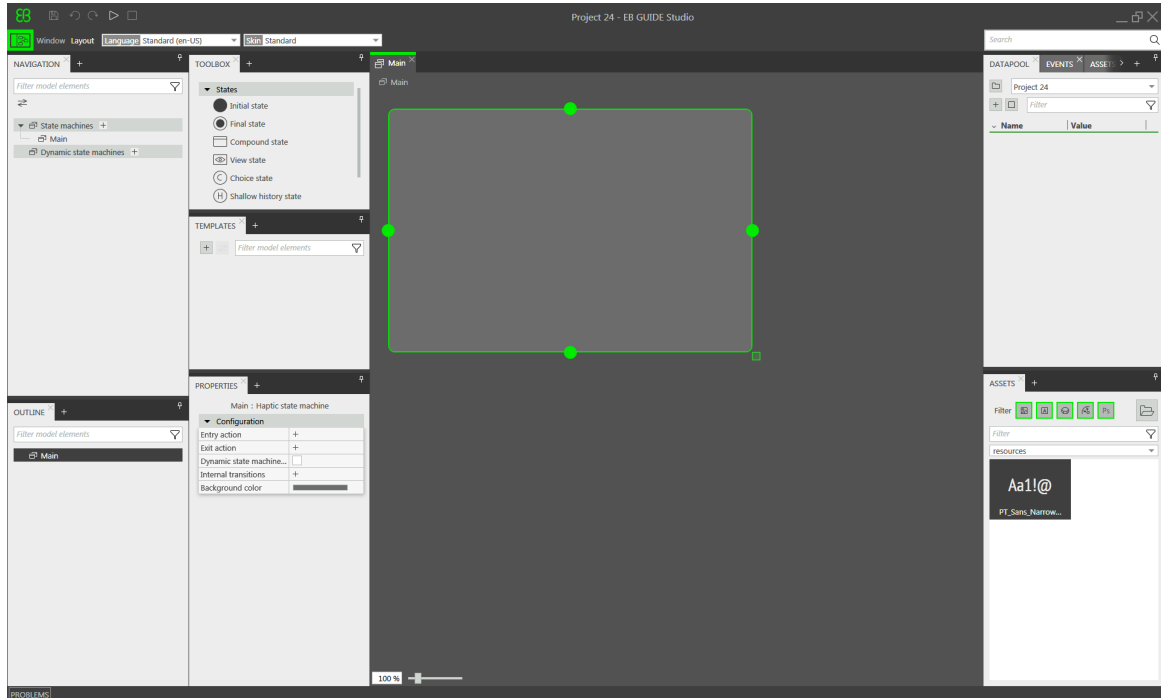


图 5.2. 具有“主要”状态机的项目编辑器

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”拖动到状态机中。

视图将随 View state 1 添加至 EB GUIDE 模型。

步骤 2

重复步骤 1。

View state 2 已添加。

步骤 3

将初始状态从“工具箱”拖动到状态机中。

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。

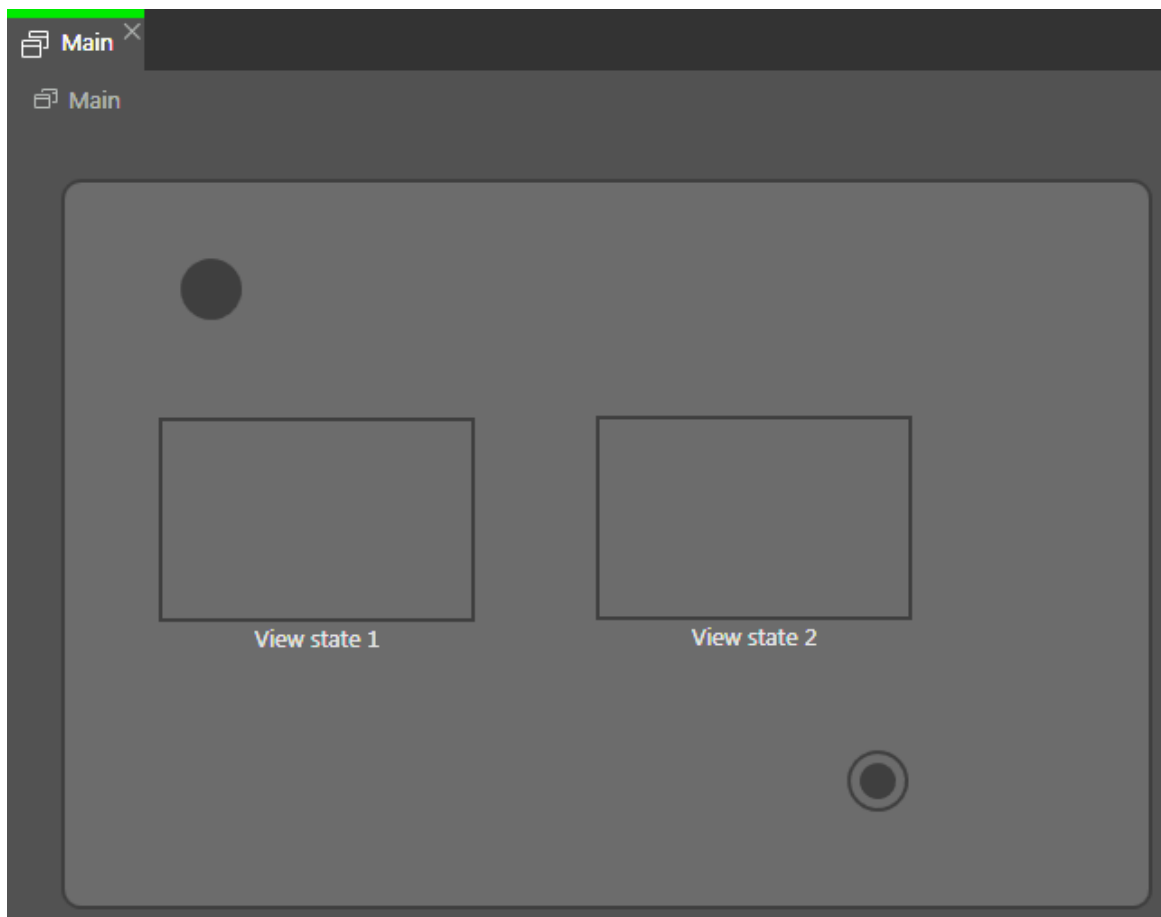


图 5.3. 具有状态的项目编辑器



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

将鼠标拖动至目标状态 View state 1。

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 View state 1 和 View state 2 之间添加转换。

选择 View state 1 并将 View state 2 作为目标状态重复步骤 2 - 4。

步骤 6

选择 View state 1 和 View state 2 之间的转换。

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”。

称为 Event 1 的事件将创建并作为转换触发器进行添加。无论何时触发 Event 1，都将执行转换。

步骤 8

在 View state 2 和结束状态之间添加转换。

选择 View state 2 并将结束状态作为目标状态重复步骤 2 - 4。

添加新的事件 Event 2 作为触发器。

此时，您的状态机将类似于下图：

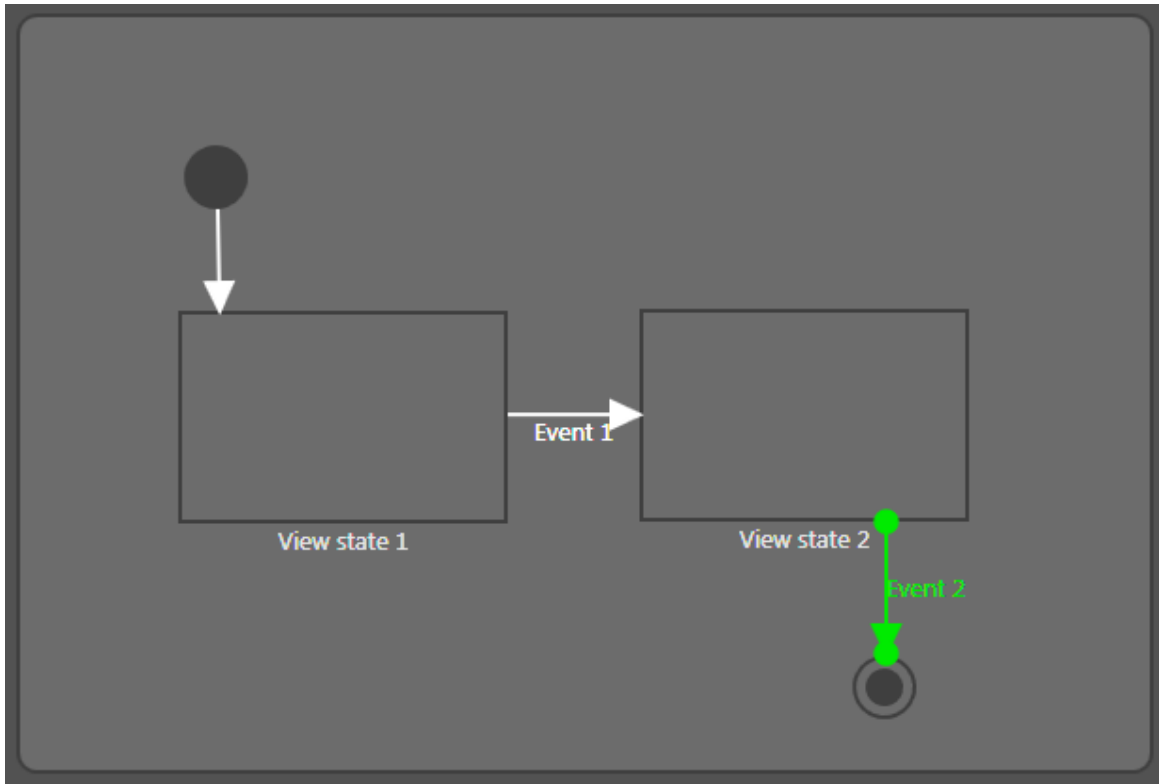


图 5.4. 通过包含事件的转换链接的状态

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：

- View state 1 已添加至模型。

步骤 1

双击 View state 1。

内容区域将显示 View 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 View 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。

矩形在模拟模式下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
}
```

如果在模拟模式下触摸了矩形，则将在 500 毫秒后触发 Event 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 View 2。

内容区域将显示 View 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 EB GUIDE 脚本 的控件属性

步骤 12

点击“接受”。

如果在模拟模式下触摸了矩形，则将在 500 毫秒后触发 Event 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之前在 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 View 1。点击红色矩形会将屏幕更改为 View 2。这是因为点击触发了 Event 1，而 Event 1 执行从 View state 1 到 View state 2 的转换。

然后，将显示 View 2。在 View 2 中点击蓝色矩形将终止状态机。这是因为点击触发了 Event 2，而 Event 2 执行从 View state 2 到结束状态的转换。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

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 可以实现在 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 2.0 或更高版本的渲染器才能显示 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

提示



场景图的伽马校正

gamma 允许调整场景图的亮度输出以匹配显示器或显示设备的亮度响应，以获得最佳视觉效果。该值必须大于 0.0 并且默认设置为 2.-2，这适用于大多数显示器。

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
- ▶ 基于图像的光源
- ▶ 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曲线来导入这些动画。此曲线应用于浮点、整数或颜色类型的动画属性。无法以与应用其他动画曲线相同的方式应用线性键值插值曲线。此曲线仅用于导入 3D 图形的动画。

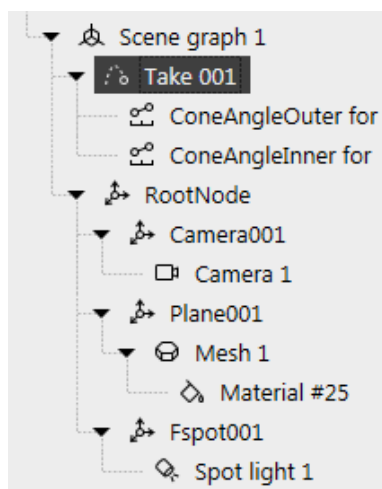


图 6.1. “导航”元素中显示的场景图示例

注意



限制

注意以下几点：

- ▶ 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。
- ▶ 如果在 EB GUIDE Studio 中，网格添加了多个材料，则仅渲染最顶层的材料控件。
- ▶ 导入从 Blender 导出的 .fbx 文件时，自发光颜色被设定为 (0, 0, 0)，并且场景中的所有光都将更改为强度为 1.0 的定向灯光。这样做的原因是因为 Blender 不会导出必要的材料和灯光信息。

导入 3D 图形文件后，在文件夹 \$GUIDE_PROJECT_PATH/<project name>/resources 中创建子文件夹。子文件夹根据导入的 .fbx 文件命名。此外，创建的日期和时间添加到子文件夹的名称。



例 6.1.

导入文件夹的命名

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 的子文件夹。

子文件夹中包含以下要素：

- ▶ 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6.24](#)、[节15.9.3](#)和[节15.10.8](#)。

有关说明，请参见[节8.1.3.1](#)和[节14.7](#)。

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。您可以将动画功能应用于控件属性、数据池项甚至颜色。您可以在“工具箱”元素的“基本控件”类别中找到动画控件。

6.2.1. 控件动画

对控件进行动画化意味着逐渐更改控件属性值。使用动画，则可以在视图中移动控件，更改控件的大小或逐渐更改控件的颜色。

动画由曲线定义。曲线具有 target 值。target 值可以是控件属性或数据池项。曲线描述了 target 值基于时间的变化。存在不同类型的曲线，您可以选择适合要创建的动画效果的曲线类型。例如，有常量曲线、线性插值曲线或正弦曲线。每个动画可能有一条或多条与之关联的曲线。

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 target，则这些曲线将同时并行覆盖此 target 属性的值。

有关动画和曲线属性，请参见 [节15.9.2.2](#)。

有关说明，请参见 [节8.1.2.7](#)。

要创建自己的自定义曲线，请参见 [节15.9.2.2.6](#)。有关更多背景信息，请参见 [节6.2.3](#)。

6.2.2. 视图转变的动画

给视图切换制作动画是指为进入或退出视图状态定义移动动画或消失动画。视图更改会触发此类动画。

您可以为视图状态和视图模板定义视图切换动画。每次重新使用视图模板后，实例将继承模板的视图切换动画。

视图切换动画有多种类型。

表 6.1. 动画类型

动画类型	说明
入口动画	进入带动画的视图状态时播放此动画。动画只能操作已添加视图的控件属性和控件功能属性。
退出动画	退出带动画的视图状态时播放此动画。动画只能操作已添加视图的控件属性和控件功能属性。

动画类型	说明
更改动画	视图状态更改时播放此动画。动画可操作源视图和目标视图的属性。源视图是添加动画的视图状态。目标视图可以是另一个视图状态或视图模板。
弹出出现动画	仅适用于视图模板和动态状态机视图。当相应的动态状态机被激活（已推送动态状态机）时播放此动画。动画可操作已添加视图的属性和控件属性。
弹出消失动画	仅适用于视图模板和动态状态机视图。当相应的动态状态机退出（已弹出动态状态机）时播放此动画。动画可操作已添加视图的属性和控件属性。

有关视图模板的动画属性，请参见[节15.9.1](#)。

有关说明，请参见[节8.7](#)。

6.2.3. 脚本曲线

默认动画曲线已经提供了大量自定义选项。另外，EB GUIDE Studio 允许您使用脚本曲线功能定义自己的动画曲线。通过此功能，可以使用 EB GUIDE 脚本 定义自己的曲线。

为了更好地理解脚本曲线动画，请在 EB GUIDE Monitor 中获取 v:diff 和 v:t_anim。有关说明，请参见[节8.1.2.8](#)。有关教程，请参见[节14.10](#)。

6.3. 抗锯齿

在 EB GUIDE Studio 中，可以为整个场景或每个场景图分别启用抗锯齿功能。因此，您可以全局启用或禁用抗锯齿，同时可以分别为场景图启用和配置抗锯齿以覆盖全局配置。

抗锯齿设置依赖于硬件。如果从硬件方面无法进行所需的设置，则控制台日志将显示错误消息和有关不支持内容的信息。

要考虑到，抗锯齿的地图分辨率越高，渲染结果质量越好。然而请注意，抗锯齿会减弱渲染性能，特别是在目标设备上。因此，起初不采用抗锯齿，如果性能良好，尝试设置两倍或四倍抗锯齿。如果高倍抗锯齿对质量影响不明显，则使用较低的设置。还应考虑到，抗锯齿带来的改进在具有高地图分辨率的小型显示器上的意义不大。

有关说明，请参见[节8.9](#)。

6.4. 应用程序和模型之间的应用程序编程接口

EB GUIDE 会抽取应用程序与 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`

6.5. 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。

通信环境由项目配置中的唯一名称及自动分配的数值 ID (0...255) 标识。

有关说明，请参见[节9.9](#)。

6.6. 图形用户界面元素

6.6.1. EB GUIDE Studio 的图形用户界面

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。

6.6.1.1. 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。

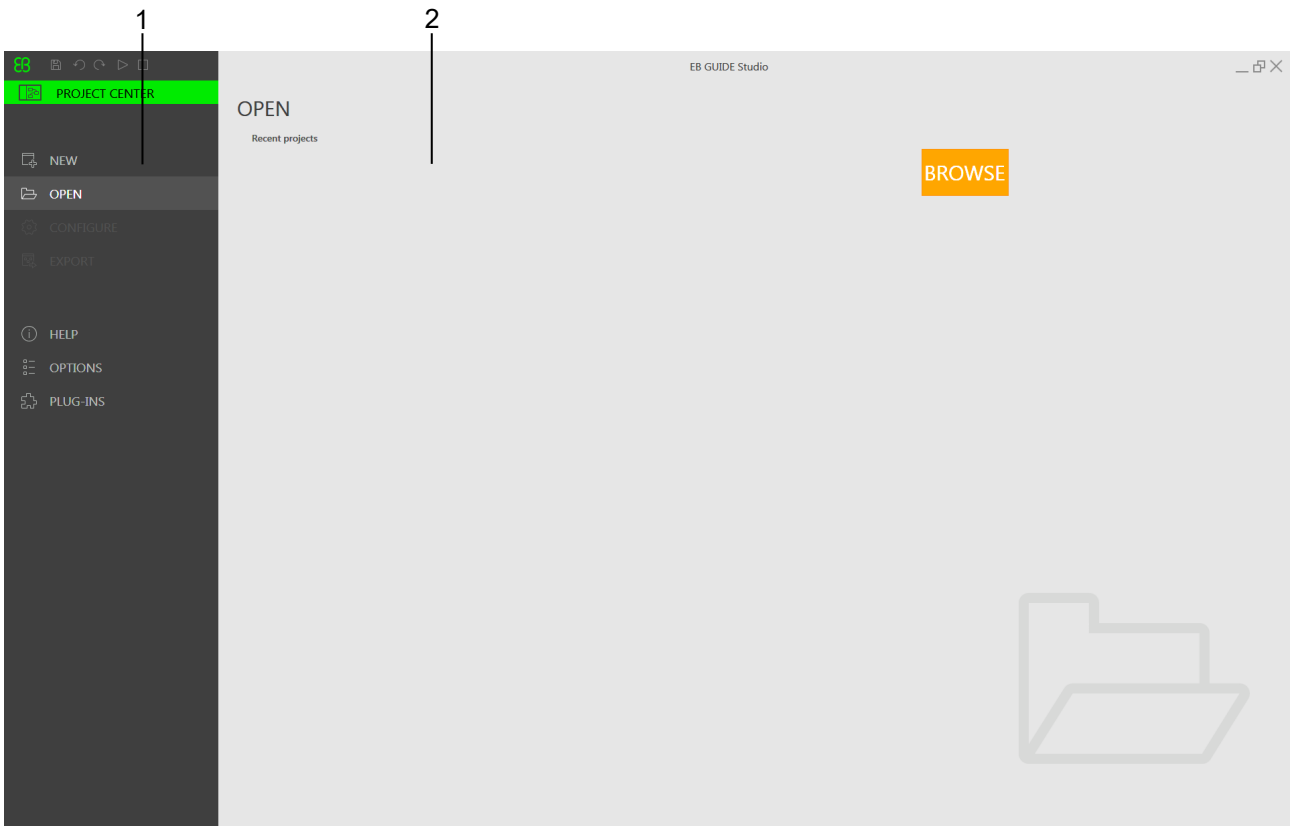


图 6.2. 包含导航区 (1) 和内容区域 (2) 的项目中心

6.6.1.1.1. 导航区

项目中心的导航区包含一些功能标签。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。

在标签中找到以下功能和设置：

“新建”

在“新建”标签中，可以创建新项目。

“打开”

在“打开”标签中，可以打开现有项目。

“配置”

在“配置”标签中，可以配置设置，例如“配置文件”、“外观”等。

“导出”

在“导出”标签中，可以导出 EB GUIDE 模型。

“帮助”

在“帮助”标签中，可以找到用户文档的链接。

“选项”

在“选项”标签中，可以切换 EB GUIDE Studio 的用户界面语言。

“插件”

在“插件”标签中，列出了所有已加载的插件。

6.6.1.1.2. 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，选中某个文件夹保存项目或定义 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。

6.6.1.2. 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。

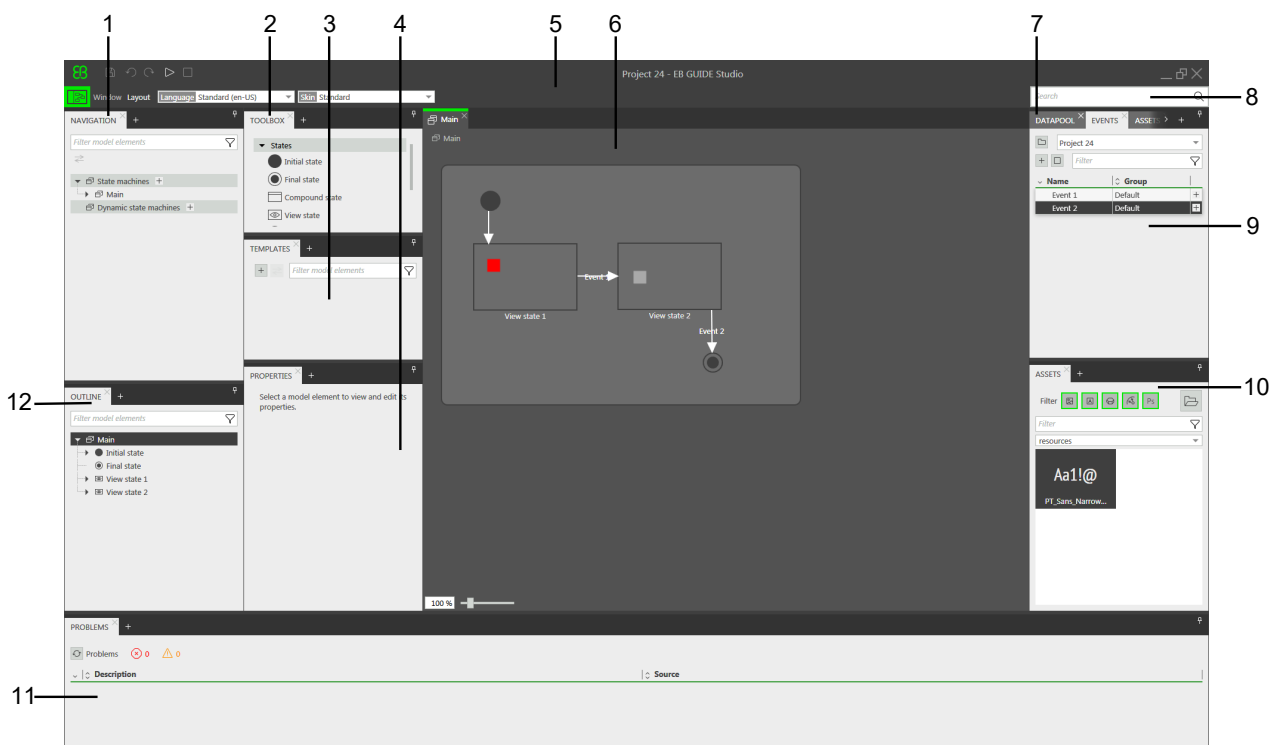


图 6.3. 项目编辑器及其区域和元素

(1) “导航”元素



- (2) “工具箱”元素
- (3) “模板”元素
- (4) “属性”元素
- (5) 命令区
- (6) 内容区域
- (7) “数据池”元素
- (8) 搜索框
- (9) “事件”元素
- (10) “资产”元素
- (11) “问题”元素
- (12) “大纲”元素

6.6.1.2.1. 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，例如状态机和动态状态机。您可以使用拖放操作添加“工具箱”中的元素，如控件和动画。

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。

点击元素中的模型元素，并按 **F3** 开始参考搜索：这样将打开 搜索结果窗口，并列出了 EB GUIDE 模型中出现的所有选择的模型元素。

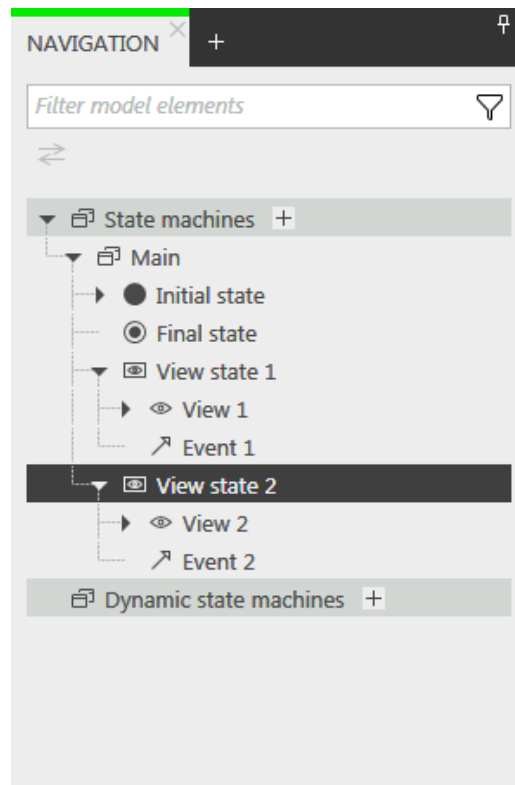


图 6.4. 项目编辑器中的“导航”元素

6.6.1.2.2. “概要”元素

“概述”元素列出了当前在内容区域中显示的结构和模型元素。

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。

点击元素中的模型元素，并按 **F3** 开始参考搜索：这样将打开 搜索结果窗口，并列出了 EB GUIDE 模型中出现的所有选择的模型元素。

6.6.1.2.3. 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。

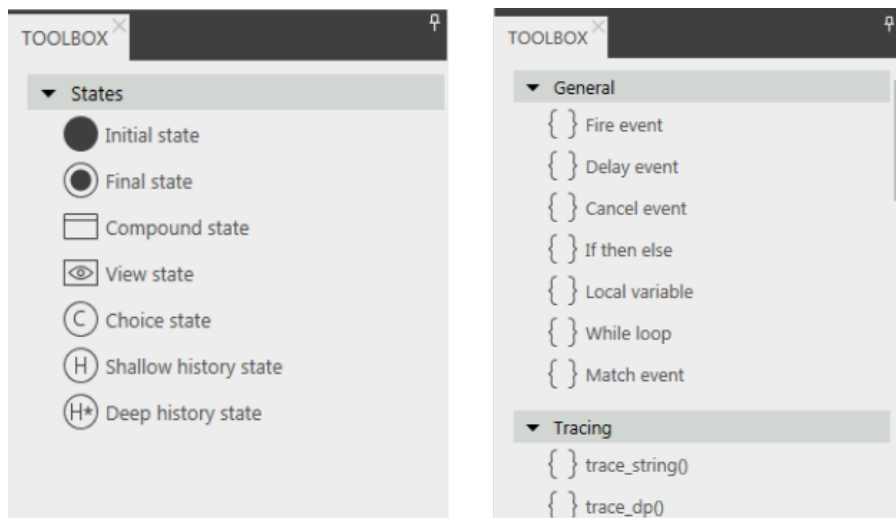


图 6.5. 项目编辑器中的工具箱

6.6.1.2.4. 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，并列出现所有选择的属性。

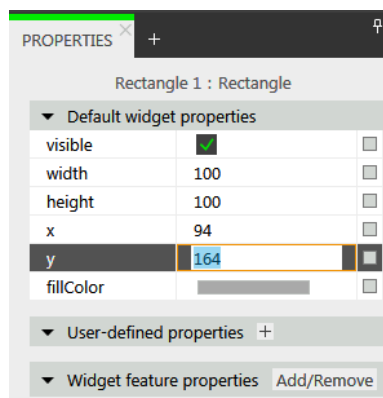


图 6.6. 显示控件属性的“属性”元素

6.6.1.2.5. 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。

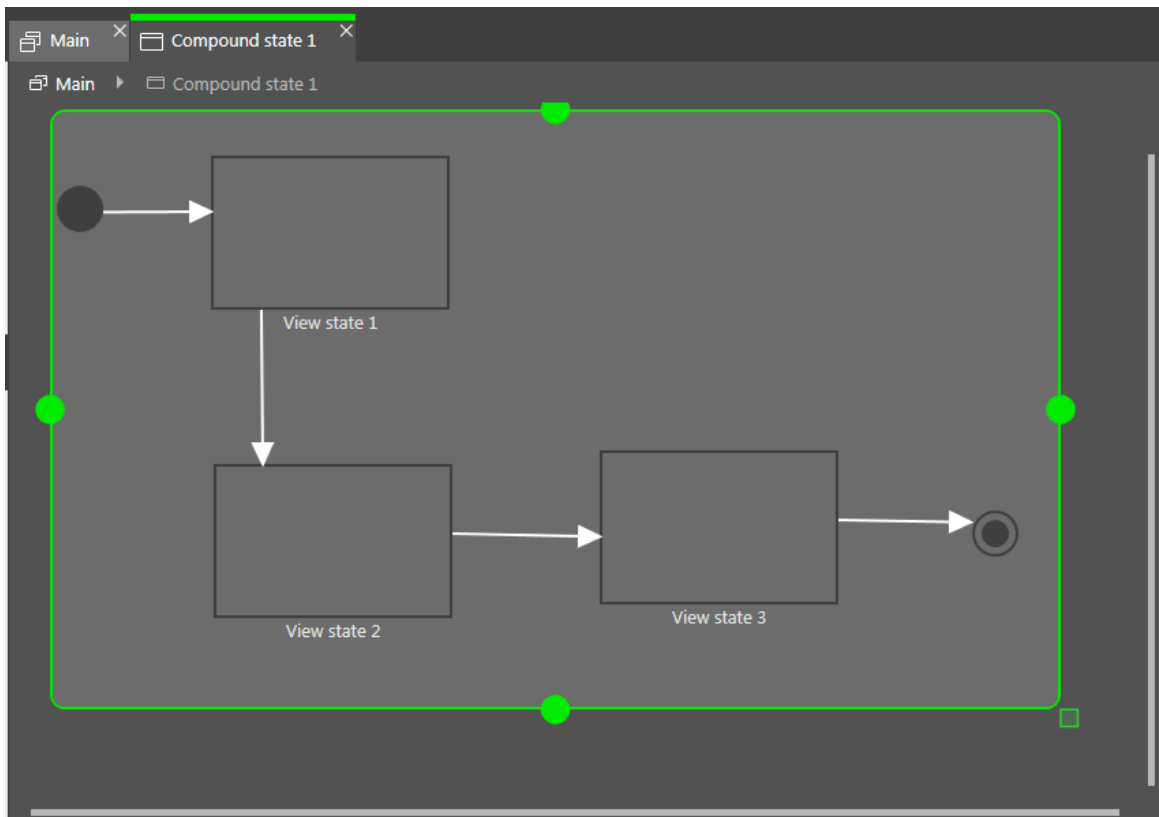


图 6.7. 项目编辑器中的内容区域

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。

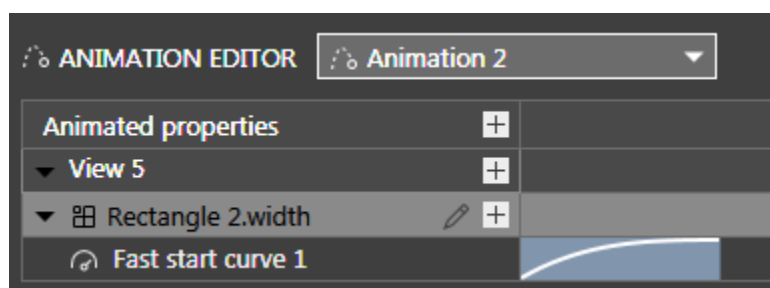


图 6.8. 动画编辑器

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，并列出了 EB GUIDE 模型中所选状态或控件的所有实例。

6.6.1.2.6. “事件”元素

这里，您可添加事件到选定的命名空间中，编辑事件中列表的属性，如“名称”、“分组”、“参数名称”和“类型”。

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。

点击元素中的模型元素，并按 **F3** 开始参考搜索：这样将打开 搜索结果窗口，并列出了 **EB GUIDE** 模型中出现的所有选择的模型元素。

提示



模型元素的多个选择

要选择多个元素，请按住 **Ctrl**，同时点击要选择的元素。也可以按住 **Shift** 并点击相应元素或使用 **Up arrow** 或 **Down arrow** 键。

在“事件”元素中，可以在以下视图选项中切换：

- ▶ 只显示选定命名空间中的事件。
- ▶ 显示选定命名空间及其子命名空间中的事件。
- ▶ 显示所有命名空间中的事件。

如果在 **EB GUIDE** 模型中定义或导入了多个模型接口，则可以根据模型元素所属的模型接口对其进行分组 或取消分组。模型元素按以下顺序分组：

1. 属于一个模型接口的元素
2. 属于多个模型接口的元素
3. 属于导入的模型接口的元素
4. 不属于任何模型接口的元素

6.6.1.2.7. “数据池”元素

这里，您可以将数据池项添加到选定的命名空间和编辑属性，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。

点击元素中的模型元素，并按 **F3** 开始参考搜索：这样将打开 搜索结果窗口，并列出了 **EB GUIDE** 模型中出现的所有选择的模型元素。

提示**模型元素的多个选择**

要选择多个元素，请按住 **Ctrl**，同时点击要选择的元素。也可以按住 **Shift** 并点击相应元素或使用 **Up arrow** 或 **Down arrow** 键。

在“数据池”元素中，可以在以下视图选项中切换：

- ▶ 只显示选定命名空间中的数据池项。
- ▶ 显示选定命名空间及其子命名空间中的数据池项。
- ▶ 显示所有命名空间中的数据池项。

如果在 **EB GUIDE** 模型中定义或导入了多个模型接口，则可以根据模型元素所属的模型接口对其进行分组 或取消分组。模型元素按以下顺序分组：

1. 属于一个模型接口的元素
2. 属于多个模型接口的元素
3. 属于导入的模型接口的元素
4. 不属于任何模型接口的元素

6.6.1.2.8. “资产”元素

这里，您可以添加资源，如图像、字体、.ebmesh、.psd 和 .ebibl 文件。所有资源文件位于 `$GUIDE_PROJECT_PATH/<project name>/resources` 文件夹及其子文件夹显示在元素的预览区域中。

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。

6.6.1.2.9. “命名空间”元素

在这里，您可以创建、移动和删除命名空间。默认情况下已添加了根命名空间。

注意




默认布局

“命名空间”元素并不在默认布局中。要将其打开，选择菜单“布局”>“命名空间”。

6.6.1.2.10. 命令区

在命令区中，您可以找到：


- ▶  按钮，可以打开项目中心
- ▶ 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。
或者，可以选中模型元素并按 F3。搜索结果窗口将打开并显示搜索结果。
- ▶ 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。要刷新搜索结果，点击 。

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 6.2. 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父状态。
视图	命中列表还显示找到的视图的父状态。
模板	命中列表还显示找到的模板的父状态和父控件。

类别	说明
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。
视图切换动画	预览显示视图切换动画所属的视图。

6.6.1.2.11. 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。

6.6.1.2.12. VTA 元素

在“VTA”（视图切换动画）元素中，可以编辑视图状态或视图模板的视图切换动画。可以选择不同的动画类型。所有相关的动画类型，例如条件为 **true** 的每个动画类型的首个动画类型将同时启动。

“VTA”元素并不会显示在默认布局中。要打开“VTA”元素，请选择“布局”菜单中的“VTA（视图切换动画）”。

6.6.1.2.13. 模板元素

在“模板”元素中，可以创建控件模板。当您想在 EB GUIDE 模型中重用控件时，模板很有用。

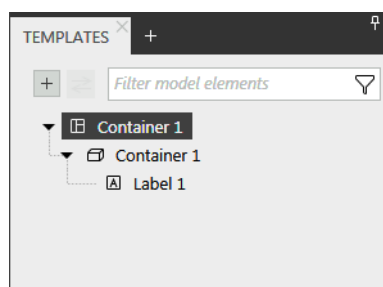


图 6.9. 项目编辑器中的“模板”元素

6.6.2. EB GUIDE Monitor 的图形用户界面

根据项目的需要，您可以在 EB GUIDE Monitor 中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。

注意



默认窗口布局

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 Monitor 窗口设置为默认布局，方法是选择“布局”>“重置为默认布局”。

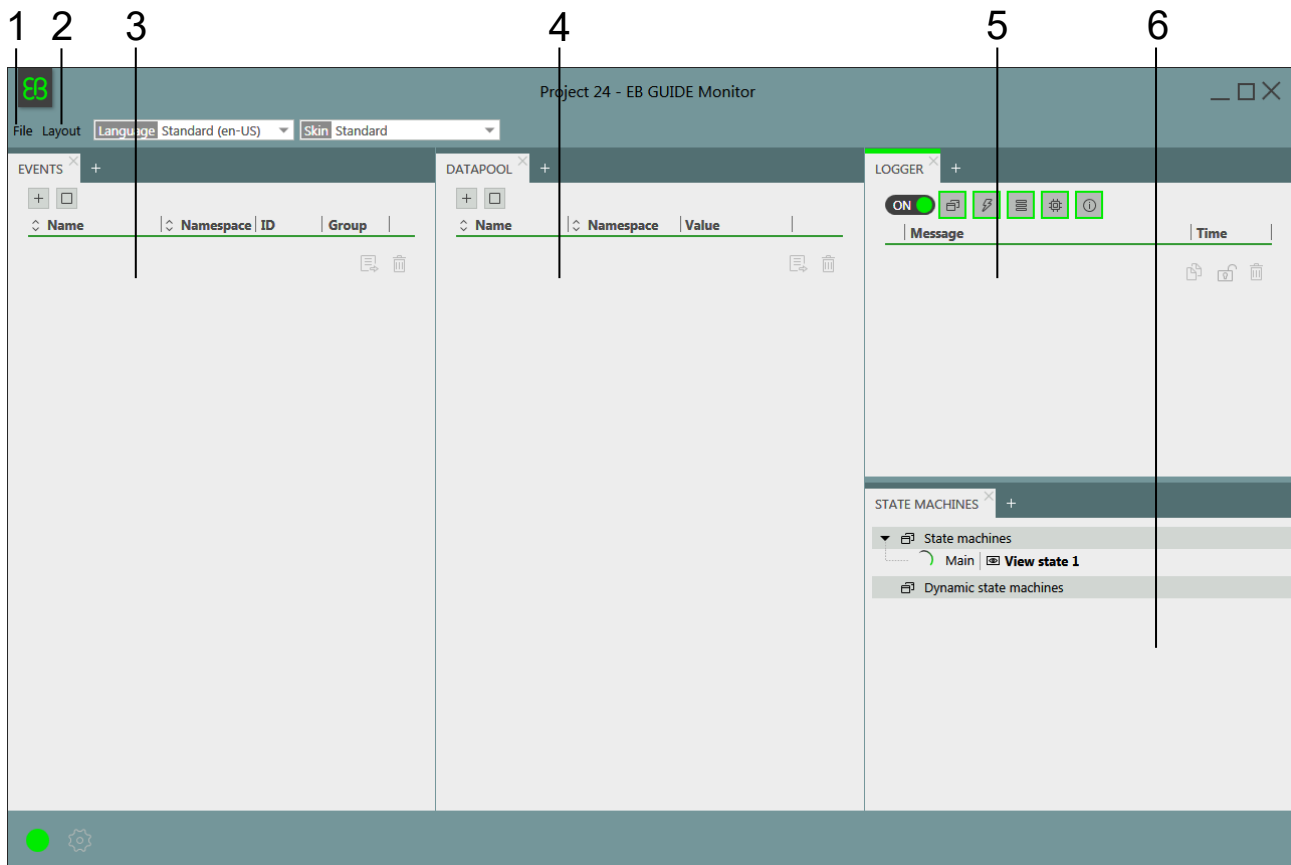


图 6.10. 使用默认布局的 EB GUIDE Monitor

- (1) “文件”菜单
- (2) “布局”菜单
- (3) “事件”元素
- (4) “数据池”元素
- (5) “记录器”元素
- (6) “状态机”元素

EB GUIDE Monitor 包括以下元素：

- ▶ 在“事件”元素中，您可以添加和触发事件动作。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，您可以添加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按钮。要更改自动滚动功能，点击 或 。
- ▶ 在“状态机”元素中，将显示当前活动的状态和状态机。
- ▶ 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”。

在命令区，可以用下拉列表框来改变语言和外观。

有关 EB GUIDE Monitor 的更多信息，请参见[节6.9](#)。

有关说明，请参见[章11](#)。

6.6.3. 可对接元素

您可以将项目编辑器的所有元素在 EB GUIDE Studio 和 EB GUIDE Monitor 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编辑器的任何部分，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。

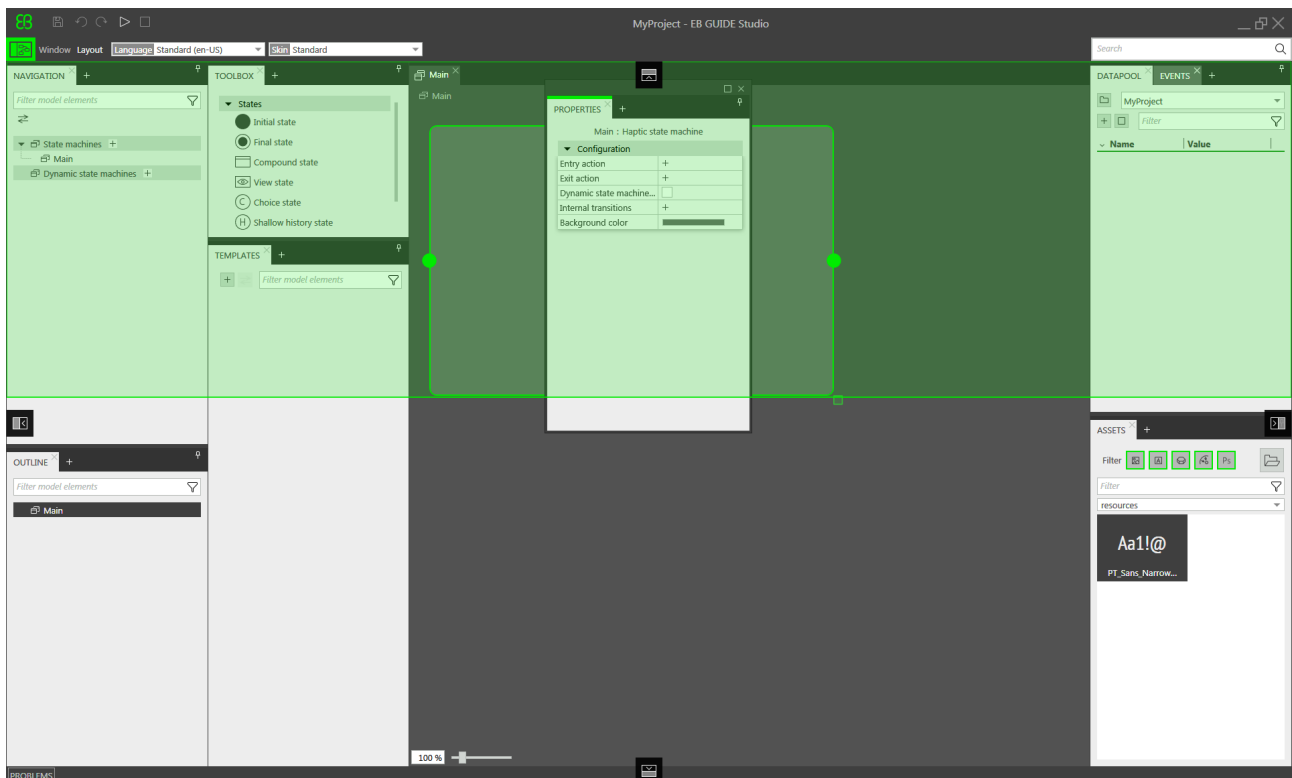


图 6.11. 对接控制和实时预览

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。
-

6.7. 数据池

6.7.1. 概念

在执行期间，模型将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。

6.7.2. 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9.6](#)。

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[节9.9](#)。

6.7.3. 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，文件夹中的所有 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。

6.8. 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 模型以及在目标设备上运行 EB GUIDE 所需的各种设置。导出的 EB GUIDE 模型可以通过定义的模型元素在目标设备上相互通信。您可以在模型接口中定义这些模型元素。

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

6.8.1. 存储格式

EB GUIDE 项目存储在 EB GUIDE 专有文件格式的多个文件中。文件格式由两个文件扩展名表示：

- ▶ .ebguide 用于 EB GUIDE 项目文件
- ▶ .gdata 用于所有其他项目文件

EB GUIDE Studio 存储格式由以下伪 EBNF 语法定义：

```
INT = [0-9]+ ;

HEXINT = '0' ( 'x' | 'X' ) [a-fA-F0-9]+

FLOAT = <as represented in the C# specification> ;

STRING = " " ; //escape characters are supported as specified in MSDN

SUFFIX = [a-zA-Z_-][a-zA-Z0-9_-]* ;

COLOR = [a-fA-F0-9]{8} ;

IDENTIFIER = ( '_' | [a-zA-Z] ) ( [a-zA-Z] | [0-9] | '_' | '$' | '.' ) * ;

file = header object ;

header = 'EBGUIDE' INT '.' INT '.' INT '.' INT SUFFIX ';' ;

object = type '(' objectId ')' '{ propertyList }' ;

type = identifier [ '<' type { ',' type } '>' ] ;

property = identifier ':' value ;

value = bool
      | int
      | float
      | string
      | color
      | object
      | externalObject
      | nullObject
      | objectReference
      | propertyReference
      | list ;

string : STRING { '\ ' STRING } ;
```




```
int = [ '+' | '-' ] INT
      | HEXINT ;

color = '#' COLOR ;

float = [ '+' | '-' ] FLOAT ;

bool = 'true' | 'false' ;

externalObject = '(' objectId ')' ;

nullObject = type '(' 'none' ')' ;

objectReference = '@' objectId '(' type ')' ;

propertyReference = identifier '@' objectId '(' type ')' ;

list = type '[' [ value { ',' value } ] ']' ;

identifier = IDENTIFIER | STRING ;

objectId = GUID ; //encoded as hex digits in
              //the XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX format
```

EB GUIDE 项目文件夹 `$GUIDE_PROJECT_PATH/<project name>/` 包含以下内容：

- ▶ 含与以下内容相关的文件的文件夹：
 - ▶ 配置
 - ▶ 模型接口
 - ▶ 命名空间
 - ▶ 状态机
 - ▶ 模板
 - ▶ 视图切换动画
- ▶ 具有项目特定资源的 `resource` 文件夹。有关更多信息，请参见[节6.18](#)。
- ▶ 包含脚本环境、数据池、事件系统、语言和外观数据的 `.gdata` 文件。
- ▶ `.txt` 文件，包含有关 `.xliff` 文件、`.psd` 文件和 `.fbx` 文件的加载错误、迁移或导入消息的信息。

6.8.2. EB GUIDE 项目的验证标准

EB GUIDE Studio 为 EB GUIDE 项目执行两种类型的验证检查。



6.8.2.1. 打开 EB GUIDE 项目时进行验证

当您打开 EB GUIDE 项目时，EB GUIDE Studio 执行多个结构验证，示例如下：

- ▶ .ebguide 项目文件不存在，或者多个 .ebguide 文件位于同一文件夹中
- ▶ 对象 ID 重复
- ▶ EB GUIDE 项目中缺少子对象
- ▶ 有重复的属性名称
- ▶ 列表项的值不一致
- ▶ .gdata 文件中的 EB GUIDE Studio 版本号与 .ebguide 文件中的 EB GUIDE Studio 版本号不对应
- ▶ 引用了未知类型

如果满足其中一个条件，则无法打开 EB GUIDE 项目，并在 `$GUIDE_PROJECT_PATH/<project name>/<project name>_LoadingErrorLog.txt` 中创建项目文件中包含错误类型和此错误位置的日志文件。

有关如何打开 EB GUIDE 项目的说明，请参见[节10.2](#)。

6.8.2.2. 使用“问题”元素进行验证

当 EB GUIDE 项目已经打开时，您可以在“问题”元素中验证 EB GUIDE 模型。例如，错误如下：

- ▶ EB GUIDE 脚本 使用无效
- ▶ 缺少默认转换
- ▶ 缺少链接项的目标

如果发现任何错误，则无法模拟运行和导出 EB GUIDE 模型。

有关如何验证 EB GUIDE 模型的说明，请参见[节10.4](#)。

6.9. EB GUIDE Monitor

在模拟运行过程中，EB GUIDE 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。

在 EB GUIDE 模拟模拟期间，在 EB GUIDE Studio 中自动启动 EB GUIDE Monitor。如果要控制导出的 EB GUIDE 模型，还可以将 EB GUIDE Monitor 用作独立应用程序。

有关 EB GUIDE Monitor 图形用户界面的更多信息，请参见[节6.6.2](#)。

有关说明，请参见[章11](#)。

对于 EB GUIDE Monitor API，请参见 `$GUIDE_INSTALL_PATH/doc/monitor/monitor_api.chm`。

您可以通过创建自定义扩展来增强 EB GUIDE Monitor 的其他功能。以下现成的 EB GUIDE 扩展示例显示如何创建 EB GUIDE Monitor 扩展：

- ▶ MonitorRemoteViewPlugin
- ▶ MonitorUiExtension
- ▶ MonitorUiExtensionEvents
- ▶ MonitorUiExtensionDatapool
- ▶ MonitorUiExtensionTargetViewer

从 EB GUIDE 微型网站下载 EB GUIDE 扩展示例：<https://www.elektrobit.com/ebguide/examples/>。有关说明，请参见随附的 EB GUIDE Studio Tutorial Using EB GUIDE Studio examples.pdf 文件。

6.10. 事件处理

6.10.1. 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。

6.10.2. 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 GTF 用于发送和接收事件。

注意



重复事件 ID

事件组中的事件 ID 必须是唯一的。当同时导入多个模型接口时，如果重复的事件 ID 属于不同的模型接口但在同一事件组内，则会发生验证错误。由于您无法在 EB GUIDE Studio 中更改导入事件的事件 ID，请恢复导入，在源模型中更改事件 ID，然后再次导出和导入。建议事先为所有 EB GUIDE 模型定义事件 ID 范围。



例 6.4. 事件用途

在具有旋转按钮或一组硬键（例如向左、向右、向上、向下、输入）的 HMI 中，可能不清楚用户接下来应该与哪个元素交互。因此，这些系统通常突出显示当前活动的显示元素。例如，YES 按钮标记为带有彩色边框的活动按钮。在 EB GUIDE Studio 中，您可以使用“已聚焦”控件功能对此突出显示功能进行建



模。当前已聚焦的元素，即 focused 属性设置为 true，也是活动的元素。并且，形成聚焦路径的此元素的父元素也是活动的。如果已聚焦元素不能处理键或旋转输入，则沿着聚焦路径向后，即朝向根元素的方向处理输入。如果聚焦路径中的一个元素处理输入，则该焦点被视为已处理。

在使用触摸输入的 HMI 中，交互是通过特定位置处的元素完成的。例如，当您接触摸屏上的 YES 按钮时，输入不会有歧义。其原因在于，根据显示器上已按下的位置，系统知道使用 YES 按钮完成了交互。

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 6.3. 允许的事件组和 ID

事件群组	识别码	详细信息																		
默认值	2	内部群组，即仅运行场景的核心接收事件																		
键输入事件	10	您可以配置以下参数：																		
		<table border="1"> <thead> <tr> <th>参数</th> <th>类型</th> <th>详细信息</th> </tr> </thead> <tbody> <tr> <td>display</td> <td>整数</td> <td>应该接收输入事件的场景 ID</td> </tr> <tr> <td>status</td> <td>整数</td> <td>0: 键按下 1: 键释放 2: Unicode 键</td> </tr> </tbody> </table>	参数	类型	详细信息	display	整数	应该接收输入事件的场景 ID	status	整数	0: 键按下 1: 键释放 2: Unicode 键									
		参数	类型	详细信息																
		display	整数	应该接收输入事件的场景 ID																
status	整数	0: 键按下 1: 键释放 2: Unicode 键																		
触摸输入事件	11	您可以配置以下参数：																		
		<table border="1"> <thead> <tr> <th>参数</th> <th>类型</th> <th>详细信息</th> </tr> </thead> <tbody> <tr> <td>display</td> <td>整数</td> <td>应该接收输入事件的场景 ID</td> </tr> <tr> <td>status</td> <td>整数</td> <td>0: 触摸按下 1: 触摸移动 2: 触摸释放 3: 邻近移动 4: 触摸新建 5: 触摸消失</td> </tr> <tr> <td>x</td> <td>整数</td> <td>触摸事件的 x 坐标</td> </tr> <tr> <td>y</td> <td>整数</td> <td>触摸事件的 y 坐标</td> </tr> <tr> <td>fingerId</td> <td>整数</td> <td>跟踪多个并行触摸位置以支持多点触控的数字</td> </tr> </tbody> </table>	参数	类型	详细信息	display	整数	应该接收输入事件的场景 ID	status	整数	0: 触摸按下 1: 触摸移动 2: 触摸释放 3: 邻近移动 4: 触摸新建 5: 触摸消失	x	整数	触摸事件的 x 坐标	y	整数	触摸事件的 y 坐标	fingerId	整数	跟踪多个并行触摸位置以支持多点触控的数字
		参数	类型	详细信息																
		display	整数	应该接收输入事件的场景 ID																
		status	整数	0: 触摸按下 1: 触摸移动 2: 触摸释放 3: 邻近移动 4: 触摸新建 5: 触摸消失																
		x	整数	触摸事件的 x 坐标																
		y	整数	触摸事件的 y 坐标																
fingerId	整数	跟踪多个并行触摸位置以支持多点触控的数字																		
旋转输入事件	12	您可以配置以下参数：																		

事件群组	识别码	详细信息		
		参数	类型	详细信息
		display	整数	应该接收输入事件的场景 ID
		increment	整数	增量值
系统通知事件	13	用于系统事件，例如在 Android 上的屏幕旋转，或如关闭等生命周期管理		

下图显示了如何在 EB GUIDE Studio 中对触摸、键和旋转事件进行建模。

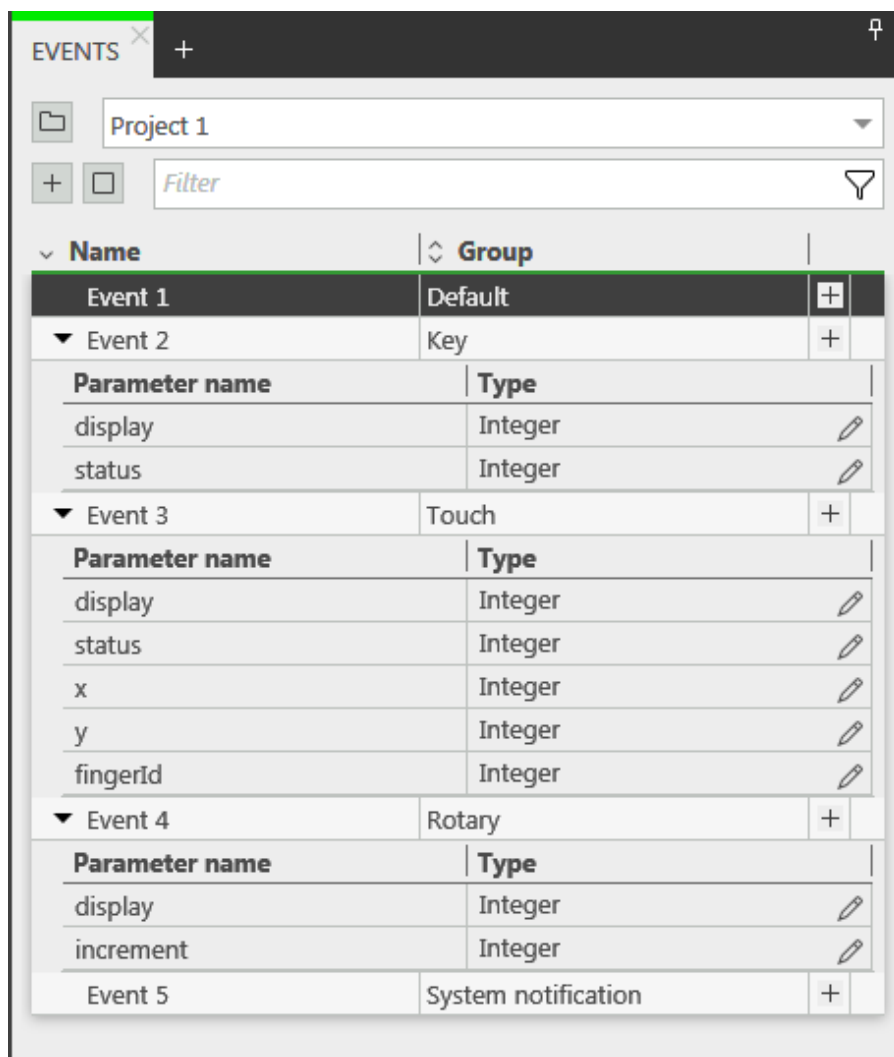


图 6.12. 事件组和事件 ID 示例

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](#)

有关参考内容，请参见 [节15.5](#)。

6.11. 扩展

6.11.1. 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 扩展是自定义 EB GUIDE 模型元素或自定义用户界面元素。有关说明和更多信息，请参见[章12](#)。

6.11.2. 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：`$GUIDE_PROJECT_PATH/<project name>/resources/target`

有关更新信息和说明，请参见 EB GUIDE GTF 用户指南。

您可以通过创建自定义的 EB GUIDE GTF 扩展来自定义 EB GUIDE 模型的外观和行为。一系列现成的 EB GUIDE 示例显示如何创建自己的 EB GUIDE GTF 扩展。从 EB GUIDE 微型网站下载 EB GUIDE 扩展示例：<https://www.elektrobit.com/ebguide/examples/>。有关说明，请参见随附的 EB GUIDE Studio Tutorial Using EB GUIDE Studio examples.pdf 文件。

有关更多类和接口的信息，请参见 EB GUIDE GTF API 文档。

6.11.3. EB GUIDE Monitor 扩展

EB GUIDE Monitor 扩展为 EB GUIDE Monitor 提供附加功能。

典型的 EB GUIDE Monitor 扩展包括：

- ▶ EB GUIDE Monitor 的附加元素
- ▶ 模拟运行期间创建屏幕截图的扩展

您可以创建自己的自定义扩展。一系列现成的 EB GUIDE 示例显示如何创建自己的 EB GUIDE Monitor 扩展。从 EB GUIDE 微型网站下载 EB GUIDE 扩展示例：<https://www.elektrobit.com/ebguide/examples/>。有关说明，请参见随附的 EB GUIDE Studio Tutorial Using EB GUIDE Studio examples.pdf 文件。

有关更多类和接口的信息，请参见 EB GUIDE Monitor API 文档。

6.12. 伽马校正渲染

6.12.1. 概念

伽马校正正在渲染管道中发挥着重要作用。伽马校正影响屏幕上的颜色重现以及图像颜色存储。伽马表达了颜色值和屏幕上的感知强度之间的关系，如需伽马示例，请参见 [图6.13](#)。

人类视觉系统 (HVS) 表现出类似的行为。它对暗图像区域中的亮度差异比对亮图像区域中的亮度差异更敏感。常见的 8 位图像格式 (JPEG、PNG) 利用这一事实并在使用非线性传递函数的 sRGB 色彩空间中存储颜色，以便提高图像暗区域的精度。这会影响到 3D 照明计算以及纹理化 alpha 混合，因为它们依赖于线性纹理颜色输入。为此，EB GUIDE 使用伽马校正渲染以消除 [图6.14](#) 中显示的这些影响。

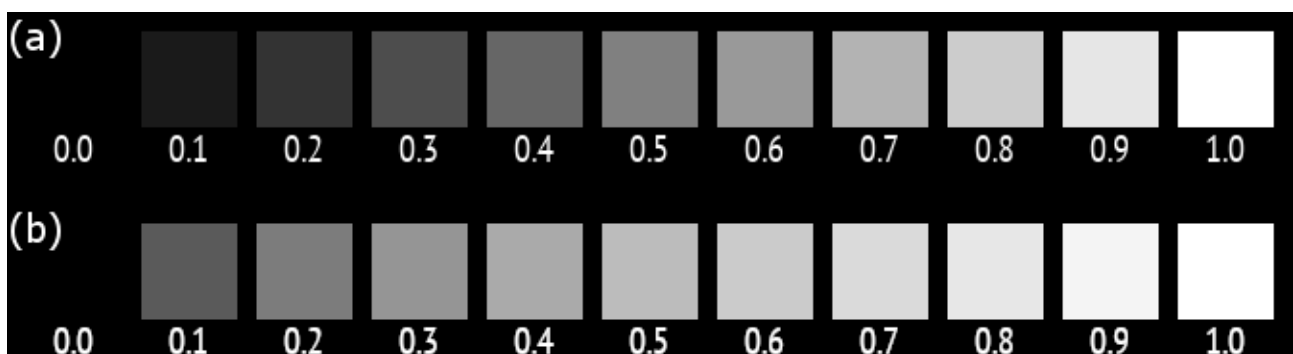


图 6.13. 伽马示例

(a) 彩色方块下方的值表示灰度等级。由于颜色值和显示亮度之间的非线性关系，对于正确校准的监视器，在 0.7 灰度级处大约达到 50% 的亮度。请注意，尽管屏幕具有非线性亮度响应，但相对差异在视觉上是均匀的。

(b) 在显示之前已经过伽马编码的颜色值。由于编码伽玛取消了显示器的伽玛，因此在 0.5 灰度级时达到 50% 的亮度。

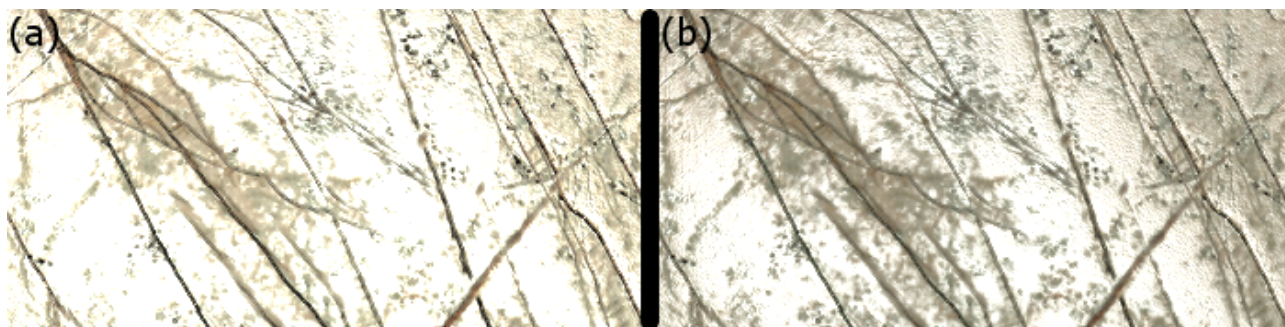


图 6.14. sRGB 纹理示例

(a) sRGB 纹理在照明计算中未经校正，且输出未经伽马校正。照明过度饱和，细节被冲刷。

(b) sRGB 纹理在照明之前被线性化，且结果经过伽马校正。细节和表面结构变得可见。

6.12.2. EB GUIDE Studio 中的伽马校正

在 EB GUIDE Studio 中，伽马校正渲染需要您按如下方式配置渲染管道：

- ▶ 输出配置控制显示器本身的伽玛编码。
- ▶ 输入配置告诉 EB GUIDE Studio 哪些图像和纹理资源应视为 sRGB 图像，以便渲染管道正确地线性化它们以进行渲染操作。

若要配置输入编码，必须为每个使用的图像或纹理配置。请注意，图像格式本身不提供有关 sRGB 编码的信息。需要事先获取此信息。有关说明，请参见 [节14.8](#)

6.13. 基于图像的光照

基于图像的光 (IBL) 是一种可以将图像用作 3D 对象的光的技术。在 EB GUIDE Studio 中，IBL 与基于图像的光控件一起应用。您可以将此控件应用与场景图节点。每个场景图不能有多个基于图像的光。如果添加的数量超过一个，则只有层级中的第一个基于图像的光可在场景中使用。



图 6.15. 基于图像的照明示例。左：一个陶瓷 PBR GGX 材料的茶壶，由三光源照亮。中间：使用基于图像的照明 (IBL)，茶壶由虚拟环境照亮，陶瓷 PBR GGX 材料看起来逼真。右：另外，可用纹理在空间上改变材料参数。

有关 `ibl` 数据类型的更多信息，请参见 [节15.3.7](#)。有关基于图像的光控件的更多信息，请参见 [节15.9.3.4](#)。

6.13.1. IBLGenerator，文件格式和导入

存储照明信息需要支持高动态范围图像数据的图像格式。EB GUIDE Studio 支持两种 IBL 格式：

- ▶ 可移植浮动地图 (.pfm)
- ▶ RGBE (.hdr)

对于 RGBE 格式，EB GUIDE 不支持 XYZ 色域。仅支持 -Y +X 方向。

要在 EB GUIDE Studio 使用这些 IBL 文件，您需要将它们转化成 .ebibl 格式。使用 IBLGenerator 来完成。IBLGenerator 是通过命令行进行控制的，并包含在 \$GUIDE_INSTALL_PATH\tools 的安装中。有关说明，请参见 [节8.1.6](#)。

IBL 文件可以包含立方体、球体或经纬度参数化的图像。在 IBLGenerator 中，您可以选择参数化类型。要查看 IBLGenerator 提供的所有选项，请在命令行界面中键入：`IBLGenerator.exe -h`

6.13.2. 使用 OpenGL 渲染器的 IBL 限制

OpenGL 3 渲染器始终支持 IBL。但是在使用 OpenGL 渲染器时，OpenGL ES 2.0 驱动器必须支持以下 OpenGL 扩展。如果不支持以下扩展之一，则忽略基于图像的光控件：

- ▶ GL_EXT_shader_texture_lod
- ▶ GL_EXT_texture_rg
- ▶ GL_OES_texture_float
- ▶ GL_OES_texture_half_float

6.14. 语言

6.14.1. EB GUIDE Studio 中的显示语言

EB GUIDE Studio 为图形用户界面提供了不同的显示语言。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](#)。

6.14.2. 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 Studio 提供的。

在项目配置中为 EB GUIDE 模型添加语言。然后，您可以导出文本，将其发送给本地化服务提供商，并将翻译后的文本导回到您的 EB GUIDE 模型中。

可以为所有数据池项目类型添加语言支持，从而使您的 EB GUIDE 模型能够以不同语言显示文本。数据池项为每种语言定义值。可以在运行时刻更改已导出的 EB GUIDE 模型的语言。

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。

有关更多信息，请参见 [节8.5.1](#) 和 [节6.14.3](#)。



例 6.5.

用于多语言用户界面的语言依赖文本

在项目配置中，添加了三种语言：英语、德语和法语。现在，您可以在更改用户界面的语言时为更改的标签建模。为此，将标签的 text 属性链接到数据池项目，其英文值为 Welcome，德语值为 Willkommen，法语值为 Bienvenue。

有关说明，请参见 [节14.6](#)。

6.14.3. 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、翻译和导入所有语言依赖项文本。文本将导出为 .xliff 文件。.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。.xliff 可以发送给您的本地化服务提供商，并可由任何翻译工具理解。

翻译后，您将翻译的 .xliff 文件导回到 EB GUIDE 模型中。

有关说明，请参见 [节10.8](#)。

.xliff 文件的结构如下所示：

- ▶ 标头包含有关源语言和目标语言的元数据：
 - ▶ source-language 和 target-language 标签遵循用于表示语言名称的 ISO 639 标准和用于表示国家/地区代码的 ISO 3166-1 标准。

- ▶ 为每个项目和语言对创建一个独特的字母数字 `sourcelanguageid` 和 `targetlanguageid`。这些 ID 可防止从其他项目或目标语言无意中导入 `.xliff` 文件。
- ▶ `trans-unit` 元素包含可本地化的数据。每个 `trans-unit` 元素包含用于存储源文本的 `source` 元素和用于存储翻译文本的 `target` 元素。将新语言添加到 EB GUIDE 模型后，`target` 元素将填充源语言。因此，在导出 `.xliff` 文件时，尚未翻译的所有 `target` 元素都显示源语言。

6.15. 命名空间

在 EB GUIDE Studio 中，可以使用命名空间创建模型元素组，例如数据源项和事件。这些组通常具有已定义的功能。每个命名空间都为模型元素创建命名范围，因此不同命名空间中的模型元素可以具有相同的名称。

每个模型元素都完全属于一个命名空间。

根命名空间是默认命名空间，既不能删除也不能重命名。根命名空间与 EB GUIDE 项目具有相同的名称。所有其他命名空间都是从此命名空间派生的。在以下情况中，模型元素始终添加到默认命名空间：

- ▶ 如果从上下文菜单中选择“添加链接到数据源项”并创建新的数据源项
- ▶ 如果在“触发器”组合框中为转化或内部转换创建事件

可以在命名空间之间移动模型元素。

注意



移动模型元素

如果将模型元素从一个命名空间移动到另一个命名空间，并且目标命名空间已包含具有相同名称的元素，则移动操作不会成功，并显示错误消息。



例 6.6. 命名空间树

在图 6.16 中，显示有命名空间树的示例。`myProject` 命名空间是默认命名空间，也是 EB GUIDE 项目的名称。一些命名空间嵌套在其他命名空间中。

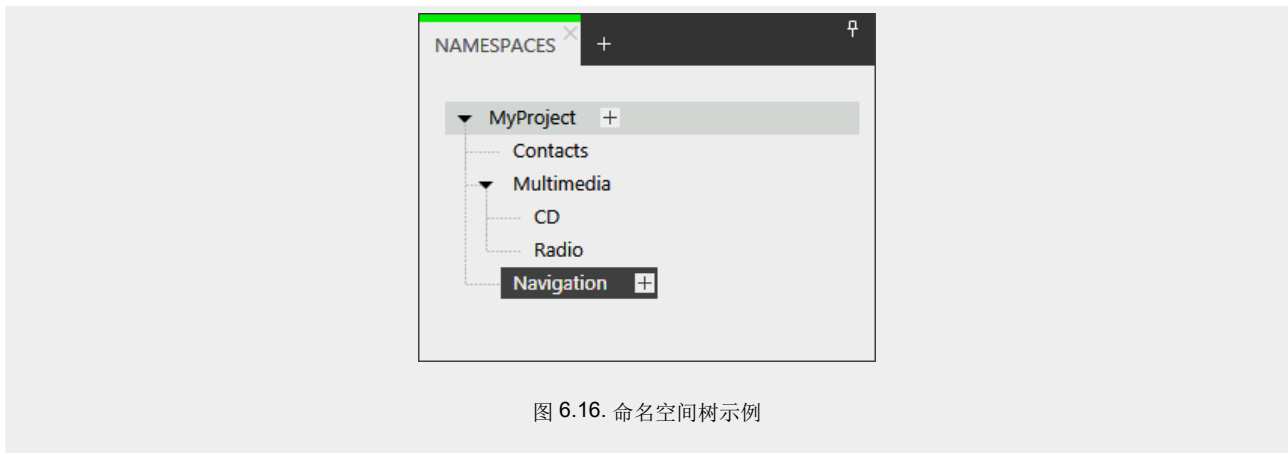


图 6.16. 命名空间树示例

有关更多信息，请参见[节6.6](#)。

有关说明，请参见[节9.13](#)。

6.16. 模型接口


使用 EB GUIDE 可以具有由多个 EB GUIDE 模型组成的 HMI，这些模型可以分别进行开发、测试、维护和运行。为此，您可以导出 EB GUIDE 模型的一个或多个接口。然后这些接口可以导入到其他 EB GUIDE 模型。您可以导入来自不同 EB GUIDE 模型的多个模型接口。

从根本上讲，这些接口由事件和数据池项组成。事件和数据池项使模型之间的通信成为可能。可以定义组成接口的事件和数据池项。

每个模型都有一个空的默认模型接口。但是您可以自己创建和定义模型接口。导出和导入通过使用 .json 文件完成。要考虑到，导入接口的 EB GUIDE 模型不能更改已导入的接口。

有关如何创建接口、如何向模型接口添加事件和数据池项以及如何导入和导出模型接口的说明，请参见[节10.9](#)。

6.16.1. 数据池项导入

EB GUIDE 模型可以读取或写入已导入模型接口的数据池项值，但不能重命名或删除这些数据池项。导入的数据池项没有  按钮。导出期间，将从每个数据池项中删除所有链接、语言或外观支持。如果数据池项具有脚本化的值，则在导出期间将其转换为纯值。如果数据池项不属于源模型中的任何命名空间，则在导入期间将创建一个以相应的模型接口命名的命名空间，并将此数据池项添加到其中。

6.16.2. 事件导入

EB GUIDE 模型可以触发已导入模型接口的事件，但无法修改、重命名或删除这些事件。导入的事件没有+按钮。

注意**重复事件 ID**

事件组中的事件 ID 必须是唯一的。当同时导入多个模型接口时，如果重复的事件 ID 属于不同的模型接口但在同一事件组内，则会发生验证错误。由于您无法在 EB GUIDE Studio 中更改导入事件的事件 ID，请恢复导入，在源模型中更改事件 ID，然后再次导出和导入。建议事先为所有 EB GUIDE 模型定义事件 ID 范围。

如果事件不属于源模型中的任何命名空间，则在导入期间将创建一个以相应的模型接口命名的命名空间，并将此事件添加到其中。

6.16.3. 事件组导入

导入具有事件组的模型接口时，事件组的所有权与导入接口的模型共享。因此，事件组对模型接口有特殊的处理：

- ▶ 不能为导入的模型接口中的事件更改事件组。
- ▶ 无法删除被至少一个作为导入模型接口一部分的事件所使用的事件组。
- ▶ 删除模型接口时，不会删除使用该接口导入的事件组。
- ▶ 当您使用重命名的事件组更新并重新导入模型接口时，事件组不会在要导入接口的模型中重命名。
- ▶ 当导入具有与现有事件组匹配的 ID 的事件组时，将会合并这些事件组。

例如：在 EB GUIDE 模型中，存在 ID 为 65536 的事件组“A”。导入具有相同 ID 65536 的事件组“B”的模型接口后，导入的事件组中的所有事件都将添加到 ID 为 65536 的事件组“A”中。

6.16.4. 命名空间导入

如果导入的模型接口具有属于特定命名空间的事件或数据池项，则这些名称空间也将导入。这些命名空间为只读。这意味着以下限制适用：

- ▶ 不能更改导入的命名空间的名称和内容，即数据池项或事件。
- ▶ 不能导入的命名空间。
- ▶ 不能向导入的命名空间添加子命名空间。
- ▶ 不能将任何数据池项或事件移动到导入的命名空间。

6.17. Photoshop 文件格式支持



EB GUIDE Studio 支持所有常用 .psd 文件格式。支持的色域为 8 位、16 位 和 32 位 RGB，以及 CMYK。您可以直接导入 .psd 文件，也可以从 .psd 文件中提取图像。.psb 文件不受支持。

导入

.psd 文件中的元素将直接放入模型中，并创建控件树。控件树由容器、图像和标签组成，这些容器、图像和标签是从 .psd 文件的图层派生的。有关说明，请参见[节8.1.4](#)。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。
- ▶ .psd 文件中的文本图层导入为标签。导入后，您可以在控件树中看到相应的标签。
- ▶ .psd 文件中的图像图层导入为图像。
- ▶ .psd 文件中的组图层导入为容器。容器以组图层命名。容器可以包含图像、标签或其他容器。

提取

创建一个子文件夹，其中包含 .psd 文件中的图像，但您正在处理的 EB GUIDE 模型不会更改。有关说明，请参见[节8.1.5](#)。

限制

EB GUIDE Studio 不支持 Photoshop 文件格式的以下功能：

- ▶ 图层效果、滤镜和纹理
- ▶ RGB 或 CMYK 以外的颜色模型
- ▶ 蒙版
- ▶ 应用于图层的多个蒙版（图层蒙版和矢量蒙版）
- ▶ 文本样式和字体
- ▶ 仅使用颜色通道

6.18. 资源管理

资源是并非在 EB GUIDE 中创建但项目所需要的内容。在资源文件夹中查找 EB GUIDE 项目的资源。

资源文件夹位于 `$GUIDE_PROJECT_PATH/<project name>/resources`。

EB GUIDE 支持以下类型的资源文件：

1. 字体
2. .ebibl 3D 图形的文件格式
3. 图像
4. 3D 图形的网格
5. .psd 文件格式

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



6.18.1. 字体

要在项目中使用字体，请将字体添加至 `$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8.4](#)。

6.18.1.1. 位图字体

EB GUIDE Studio 版本 3.0 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` 文件夹中，为您的 .fnt 位图字体和使用第三方工具生成的 .png 纹理文件创建子文件夹。EB GUIDE Studio 期望在 .fnt 文件所在的文件夹中找到 .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文件夹。

6.18.1.2. 多字体支持

在 EB GUIDE Studio 中，您可以使用多字体支持创建自己的字体组合。例如，如果您选择的字体未提供所有必需的字符，则此功能很有用。在这种情况下，您可以使用其他字体的字符替换缺少的字符。

可以将多字体支持添加到以下模型元素：

- ▶ 类型字体的属性和字体列表条目
- ▶ 用户定义的类型字体属性和字体列表条目
- ▶ 类型字体数据池项或字体列表条目

您可以定义要为其使用特定字体的 Unicode 字符范围，如下所示：

- ▶ 使用单个 **Unicode** 字符，例如 0000。
- ▶ 使用逗号分隔的多个 **Unicode** 字符，例如 0000, 0001。
- ▶ 使用一系列 **Unicode** 字符，例如 0000–FFFF。
- ▶ 使用逗号分隔的几个范围，例如 0000–0022, 0045–0055。

使用十六进制数字格式指定字符。

字体应在 `$GUIDE_PROJECT_PATH/resources` 中可用。

添加多字体支持时，会自动添加默认的多字体值。您不能删除默认的多字体值，也不能编辑其优先级和范围。但是，您可以编辑默认值的大小和字体。

PRIO	FONT	RANGE	
0	18 arialbd.ttf	0000-00020	×
1	25 fireflysung.ttf	00FF	×
2	12 arial.ttf	FFFF	×
Def.	30 PT_Sans_Na...	0-FFFFFFFF	

图 6.17. 添加了多字体支持的类型字体属性的示例

有关如何使用多字体支持的说明，请参见[节8.4.3](#)。

6.18.2. 3D 图形的基于图像的光照

可以在 **EB GUIDE Studio** 中使用基于图像的光照。外部命令行工具 **IBLGenerator** 将 `.pfm` 或 `.hdr` 文件作为输入数据，并创建表示 **IBL** 资源的 `.ebibl` 文件。IBL 资源由基于图像的光控件的 `ibl` 属性使用。

有关如何获取 `.ebibl` 文件的说明，请参见[节8.1.6](#)。

有关背景信息，请参见[节6.13](#)。

6.18.3. 图像

要在项目中使用图像，请将图像添加至 `$GUIDE_PROJECT_PATH/<project name>/resources`。如果您从另一个文件夹中选择图像，则该图像将复制到上述项目文件夹中。

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](#)。

6.18.3.1. 9-patch 图像

通过 9-patch 图像方法，EB GUIDE Studio 可以支持包含附加元信息的图像。9-Patch 图像是可伸缩的 .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。已标记的区域会进行缩放。标记不会显示在 EB GUIDE Studio 中。

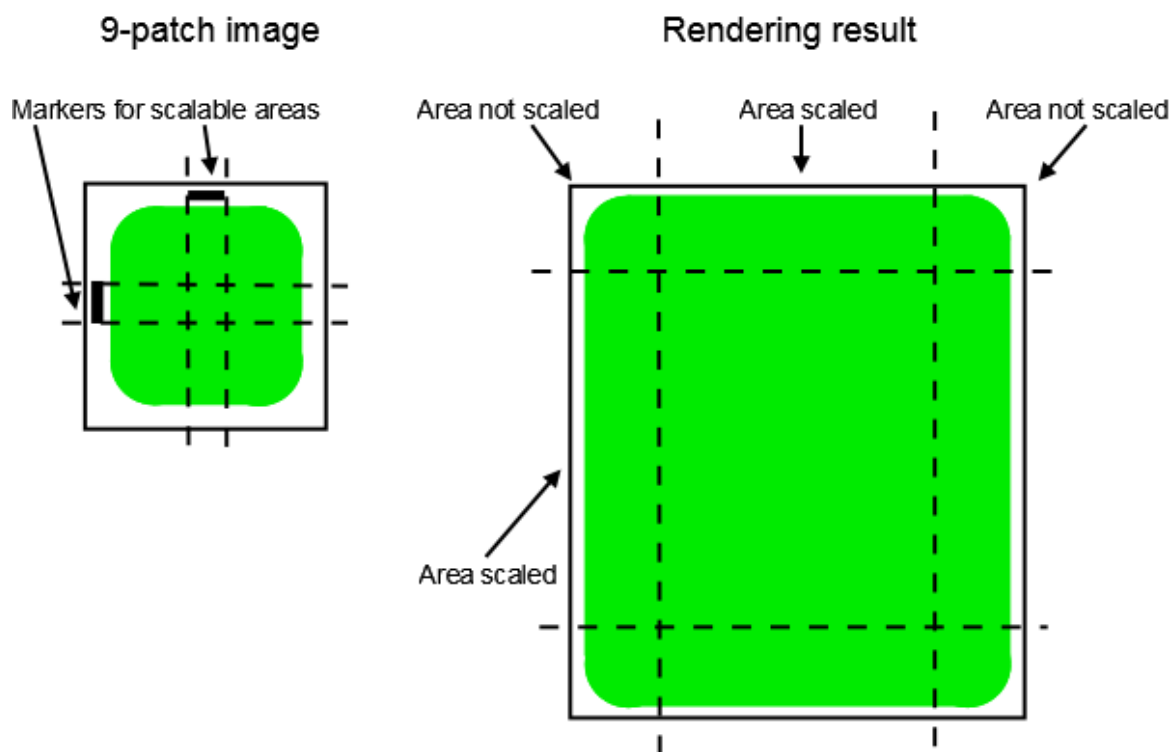


图 6.18. 9-patch 示例

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 2.0 或更高版本的渲染器。
- ▶ 9-patch 处理仅适用于 .png 图像。
- ▶ 对于 9-patch 图像，*.9.png 扩展名是必须的。
- ▶ 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](#)。

6.18.4. 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` 中找到子文件夹。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](#)。

6.19. 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。

6.19.1. 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

6.19.2. 前缀和识别符

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。为了实现此操作，EB GUIDE 脚本 中的每个识别符（即对象的名称）都必须具有前缀。前缀定义对象的类型，后跟冒号。

前缀集在 EB GUIDE 脚本 中是固定的，您无法引入新的前缀。存在下列前缀：

- ▶ 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 6.7.****EB GUIDE 脚本 中的识别符**

```
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!";
```

6.19.3. 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 6.8.****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。

6.19.4. 类型

EB GUIDE 脚本 是强类型和静态类型编程语言。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。

6.19.5. 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 6.9.
对整数值求值

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

6.19.6. 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 6.10.

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

```

6.19.7. 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 6.11. 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
```

6.19.8. 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。

6.19.9. 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。变量具有前缀 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 6.12.

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;
    }
}
```

6.19.10. 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 6.13.
while 循环的用法

```
// Assume dp:whaleInSight is of type bool
while( ! dp:whaleInSight )
{
    dp:whaleInSight = f:lookAtHorizon();
}
```

6.19.11. 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。

注意



if condition 表达式

如果 **if** 表达式包含由 **&&** 或 **||** 串联的几个子条件，则与某些其他编程语言不同，**EB GUIDE** 脚本 将评估所有子条件。这意味着，如果子条件为 **false**，因此整个条件为 **false**，则仍将评估所有子条件。

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 6.14.**

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 {}
```

6.19.12. 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 6.15.**

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);
```

6.19.13. 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。具有前缀 dp: 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。识别符可以仅包括数据池项的名称, 或者, 如果数据池项不在默认命名空间中, 则命名空间的名称后面跟着数据池项的名称。

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 6.16. 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。

6.19.14. 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 6.17. 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 6.18. 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;  
    }
```

6.19.15. 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 6.19. 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。

6.19.16. 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 6.20.

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 6.21.

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 6.22.

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 {}

```

识别符可以仅包括事件名称，或者，如果事件不在默认命名空间中，则命名空间的名称后面跟着事件名称。

如果 EB GUIDE 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 6.23. 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;
}

```

6.19.17. 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 6.24. 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);

```

6.19.18. 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15.4.3](#)。

6.20. 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9.8](#)。

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

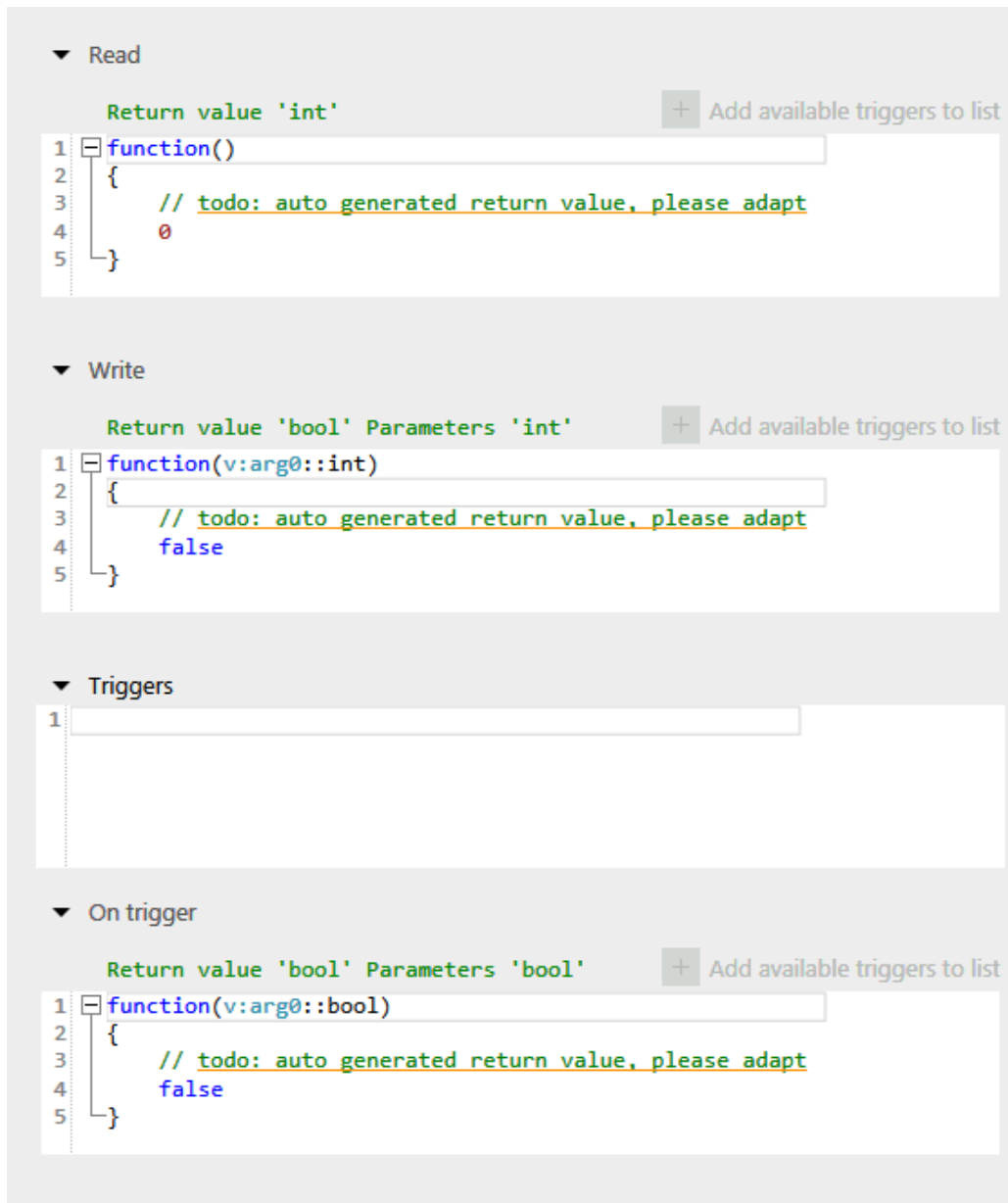


图 6.19. EB GUIDE Studio 中的 EB GUIDE 脚本 编辑器

- ▶ 当读取脚本化的值属性时，会调用“读取”脚本。如果属性为列表类型，则参数包括列表索引。
“读取”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“写入”脚本。
新属性值是“写入”脚本的参数。如果属性为列表类型，则参数包括列表索引。
“写入”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”脚本包含用于触发运行“触发器”脚本的事件、数据池项和控件属性列表。

点击“添加可用的触发器到列表”会将相应脚本中突出显示的所有触发器添加到触发器脚本中。

- ▶ “触发器”脚本在初始化时，或在事件触发器或属性更新后调用。

“触发器”脚本的参数指示了运行脚本的原因。运行脚本的原因可能是初始化操作，或“触发器”列表中的触发器之一。

“触发器”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

- ▶ “长度”脚本仅适用于列表类型的属性。

“长度”脚本的返回值表示列表的当前长度。

6.21. 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。

有关说明，请参见 [节8.6](#)。

6.22. 状态机和状态

注意



更改状态机和状态的背景颜色

在 EB GUIDE Studio 中，您可以更改以下内容的背景颜色：

- ▶ 状态机
- ▶ 视图状态
- ▶ 组合状态

要更改背景颜色，请从“属性”元素中的 Background color 下拉列表框中选择一种颜色。

6.22.1. 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，您可以创建以下类型的状态机。

6.22.1.1. 触觉状态机

触觉状态机允许指定 GUI。

6.22.1.2. 逻辑状态机

逻辑状态机允许在没有图形用户界面的情况下指定一些逻辑。

6.22.1.3. 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[节14.1](#)。

6.22.2. 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

6.22.2.1. 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。

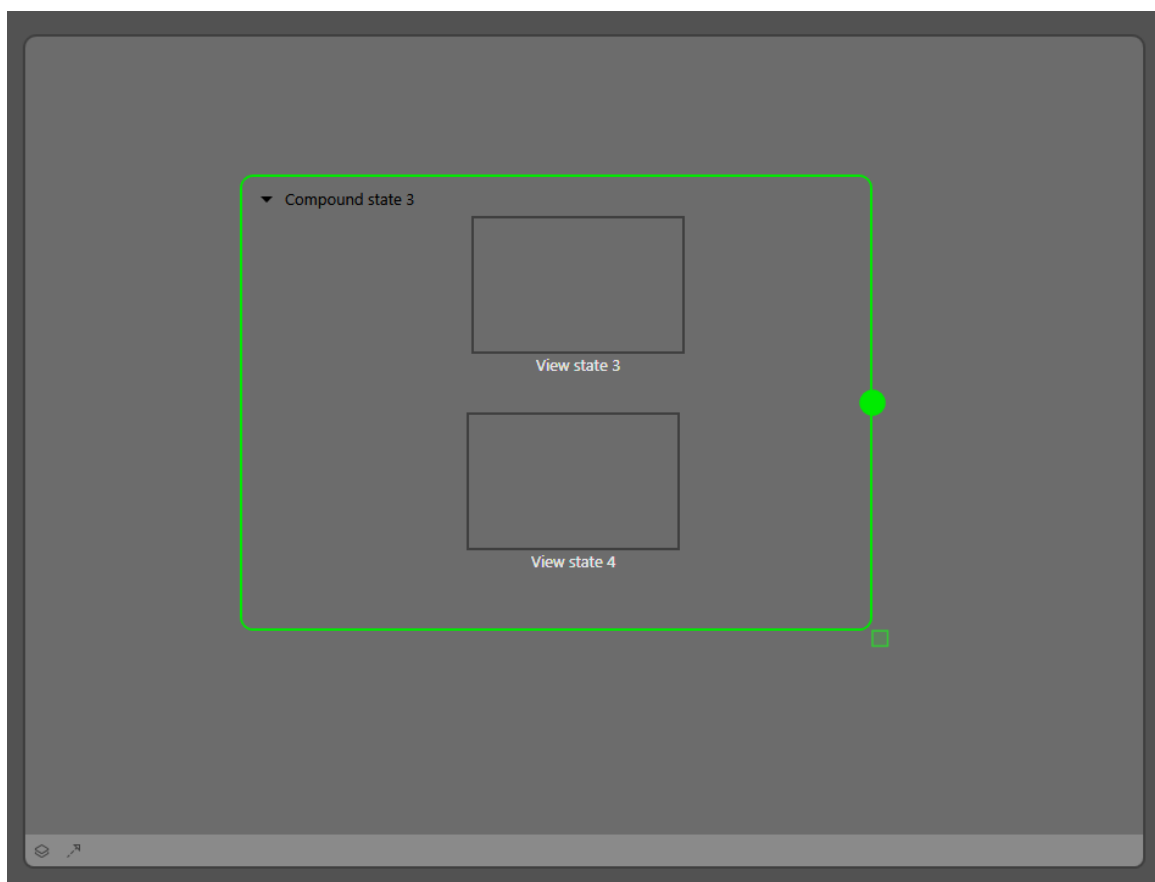


图 6.20. 组合状态

在“导航”元素中，状态层次结构显示为树结构。

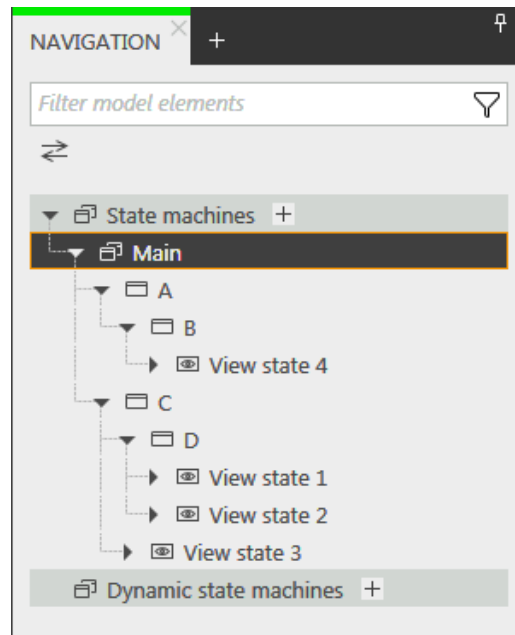


图 6.21. 以树形式显示的状态层次结构

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。

6.22.2.2. 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。

6.22.2.3. 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

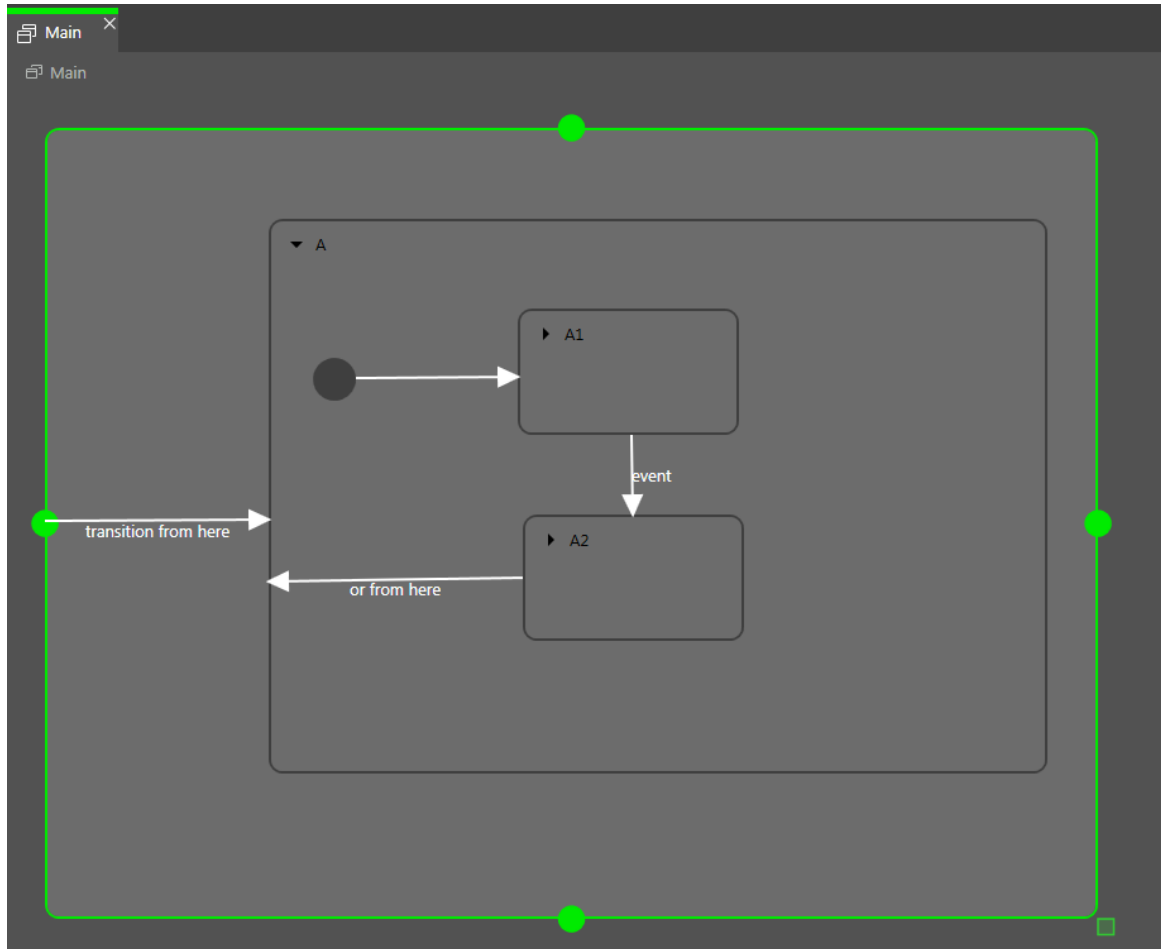


图 6.22. 初始状态示例

组合状态只能有一个初始状态。

6.22.2.4. 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。

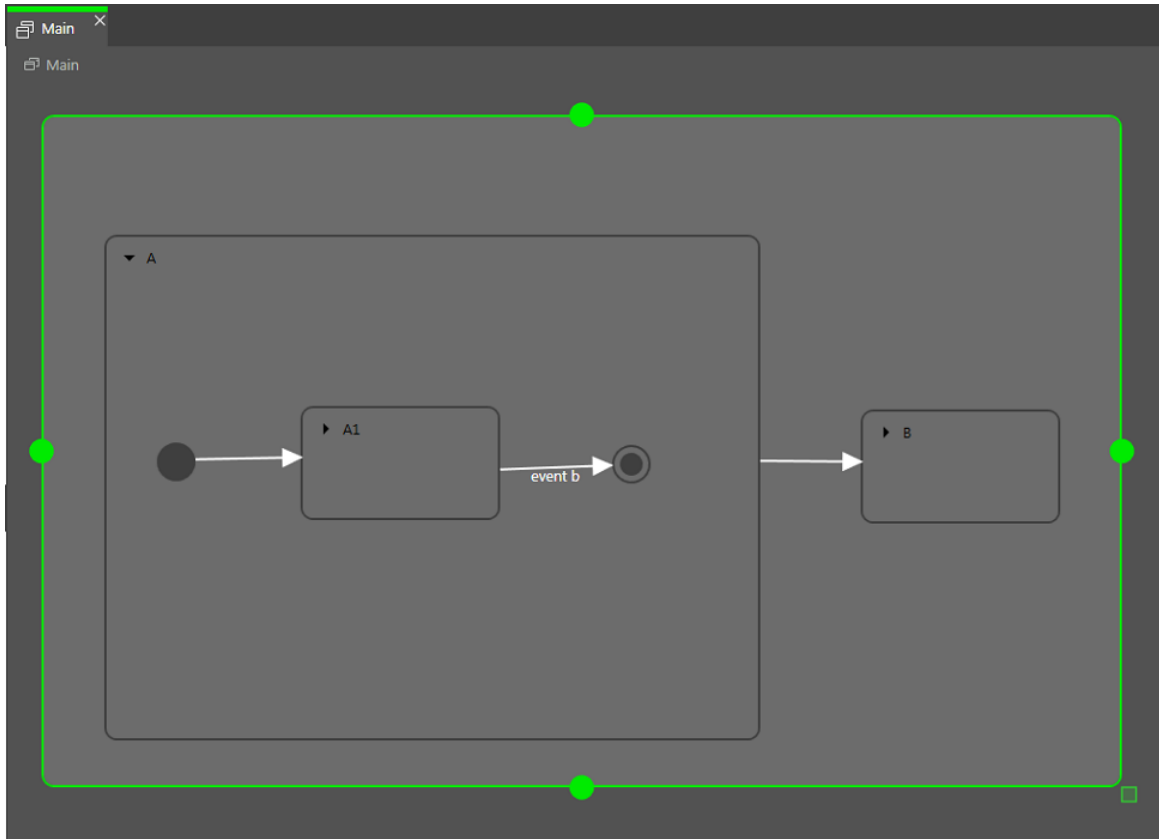


图 6.23. 组合状态中结束状态的用法

组合状态只能有一个结束状态。

6.22.2.5. 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。

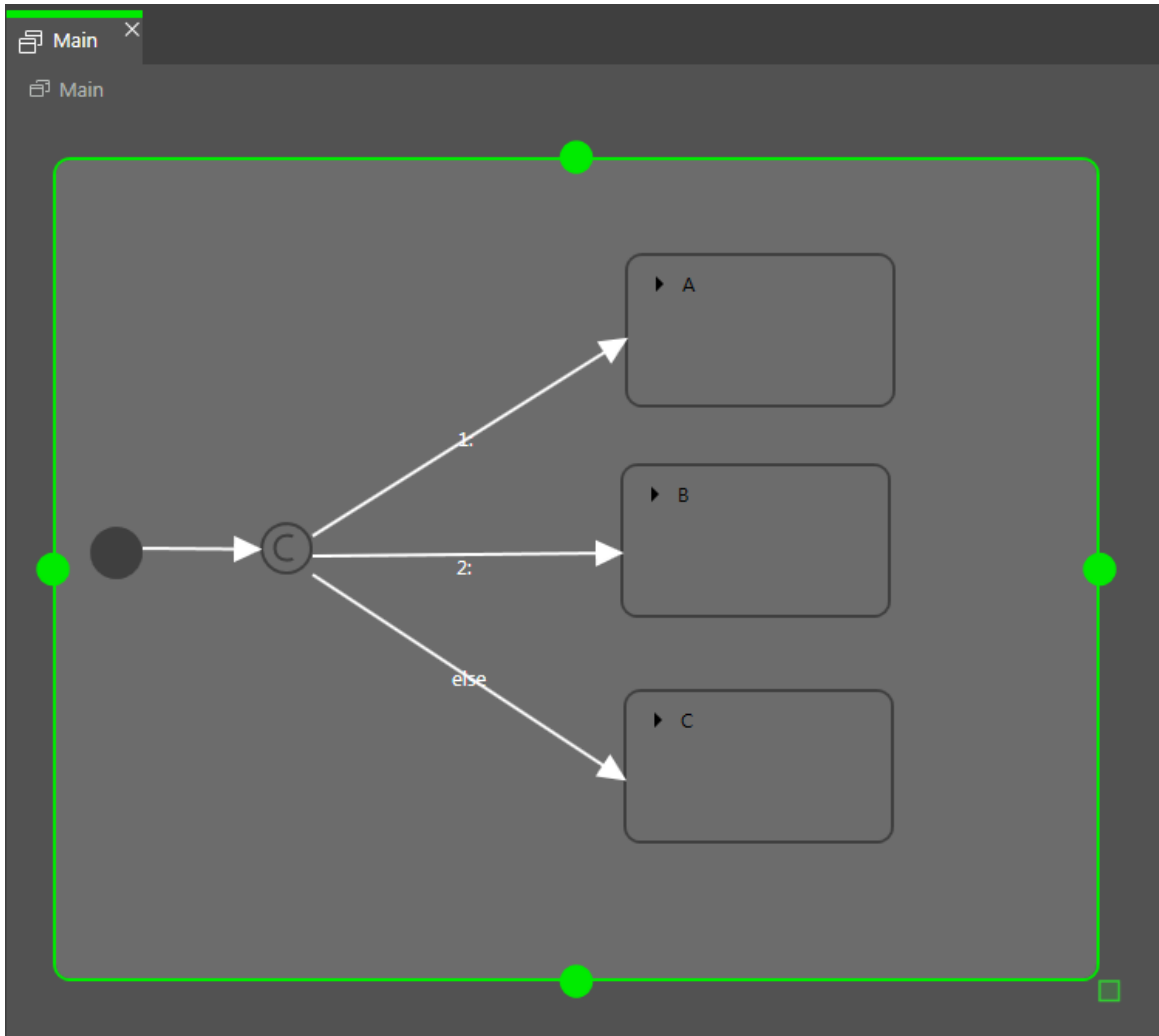


图 6.24. 具有传入和传出转换的选择状态

6.22.2.6. 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。

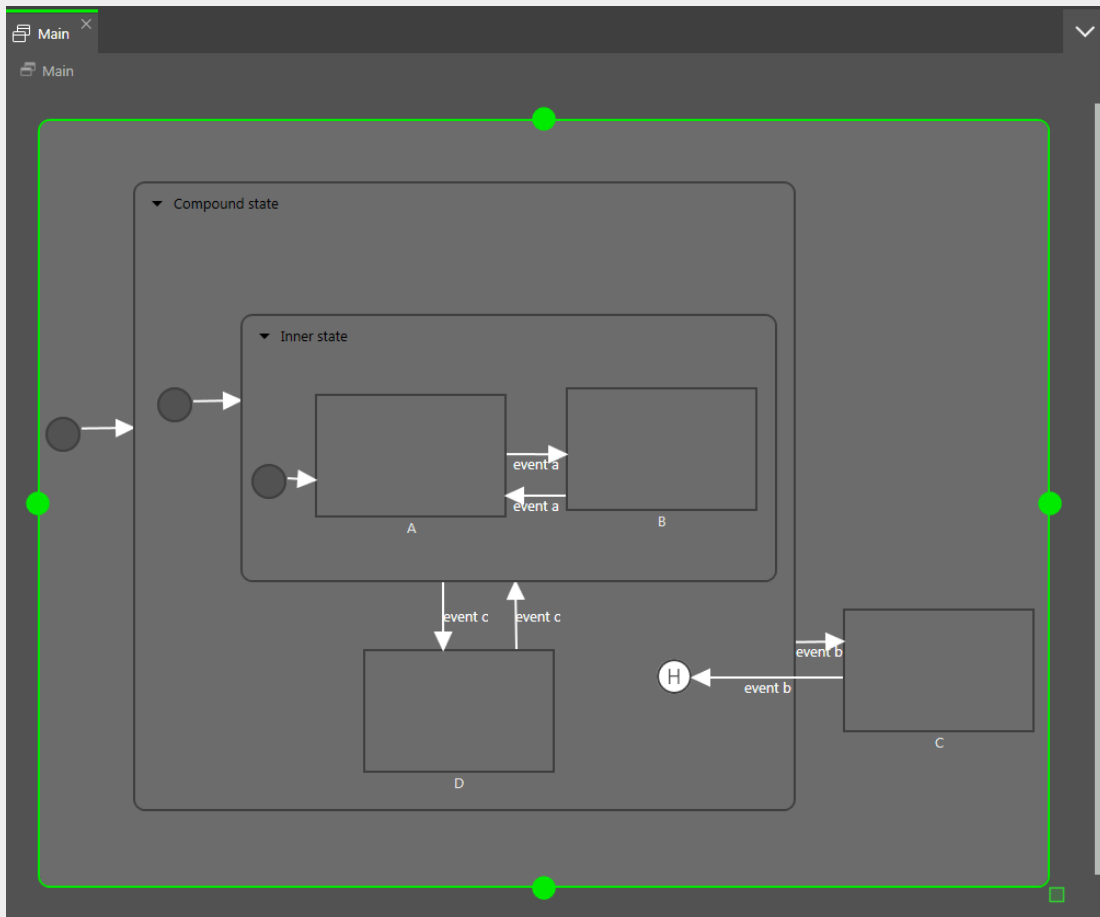


图 6.25. 浅度历史状态

- ▶ 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。

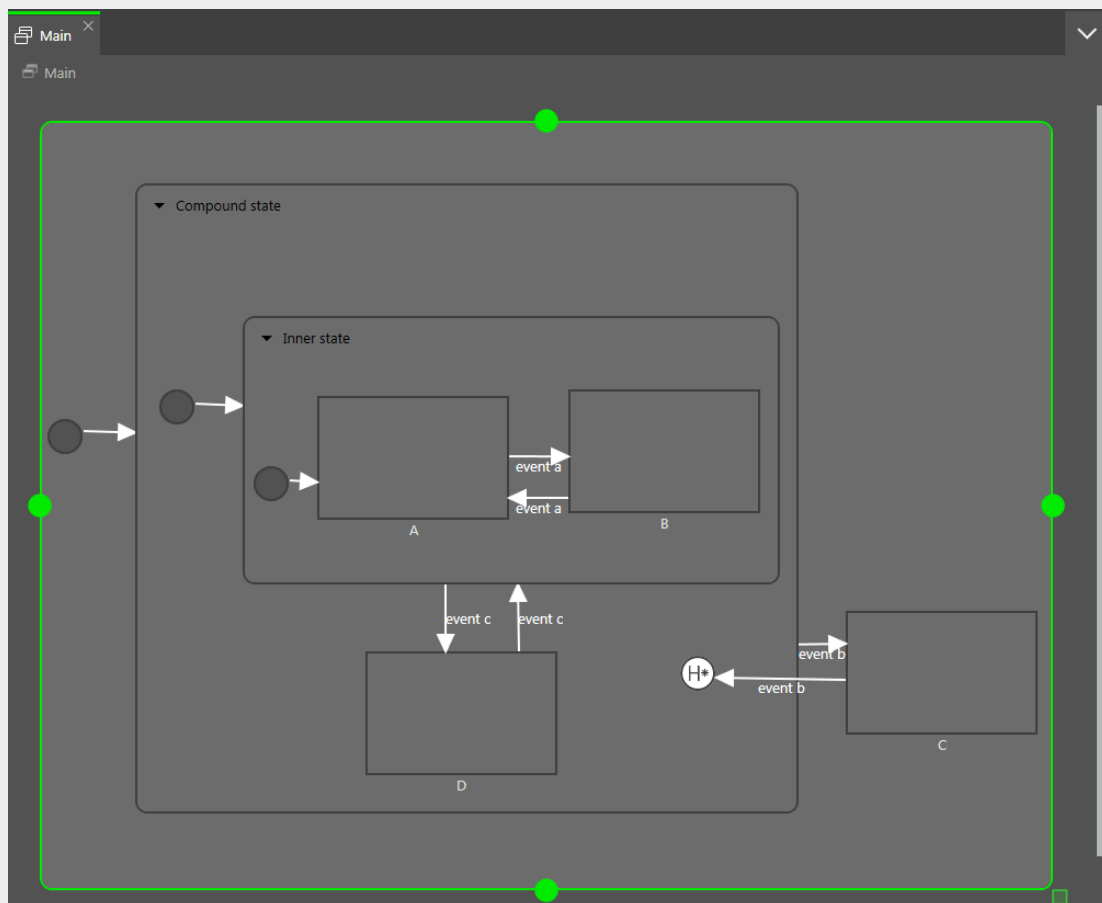


图 6.26. 深度历史状态

▶ 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。

6.22.3. 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

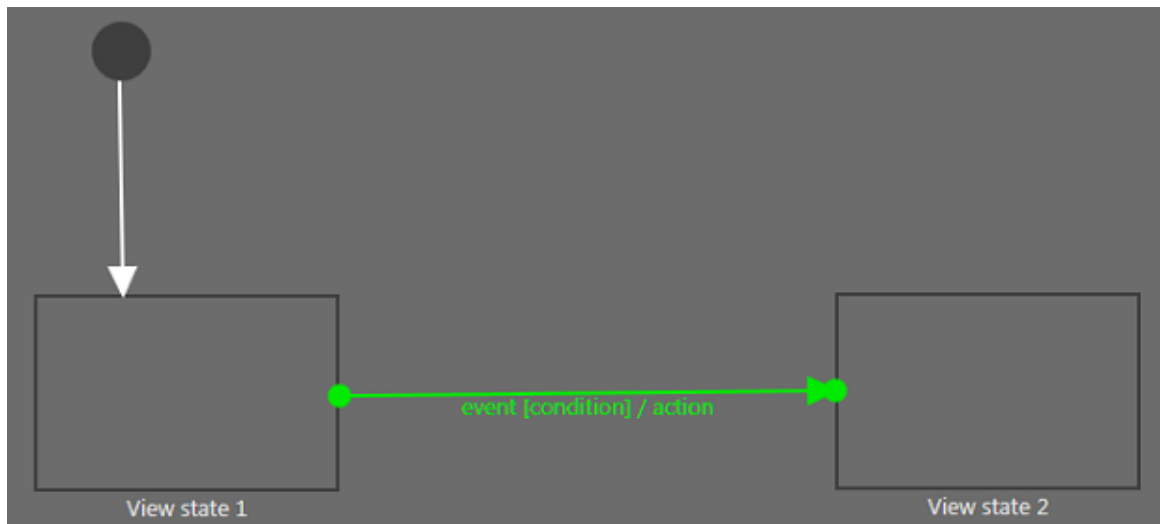


图 6.27. 转换

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

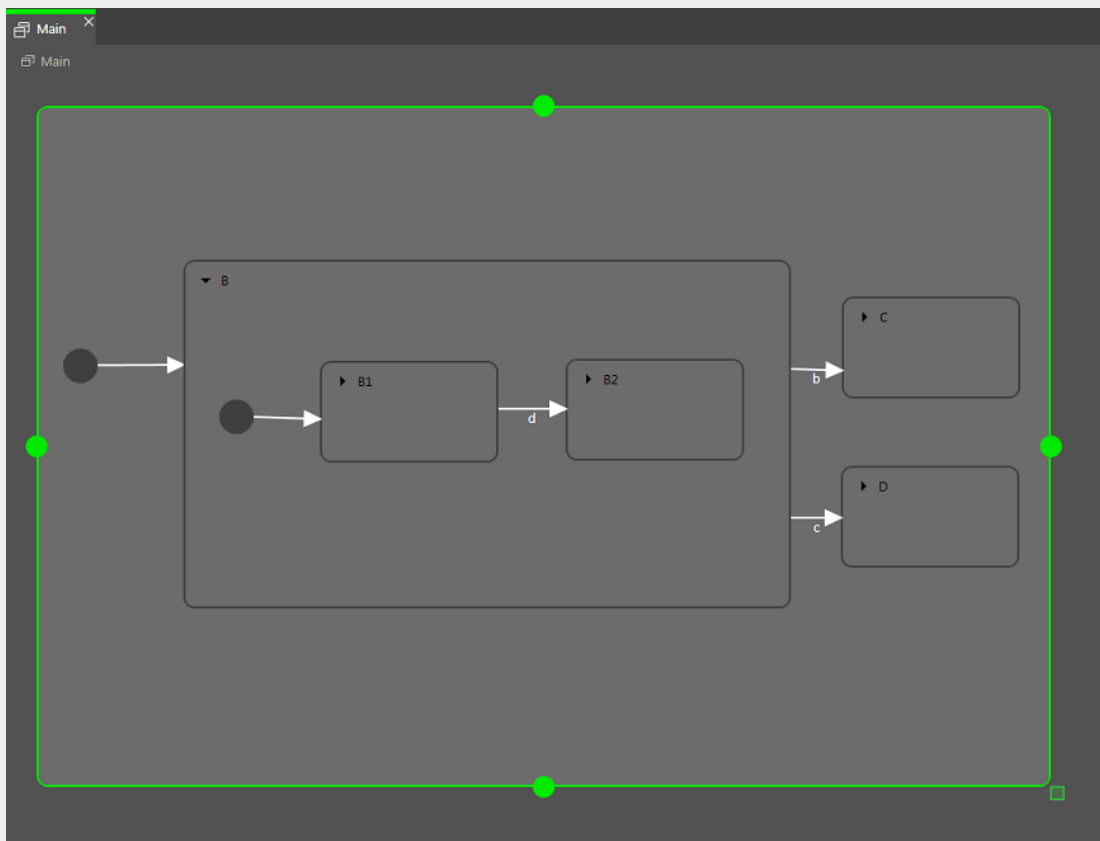


图 6.28. 转换继承

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

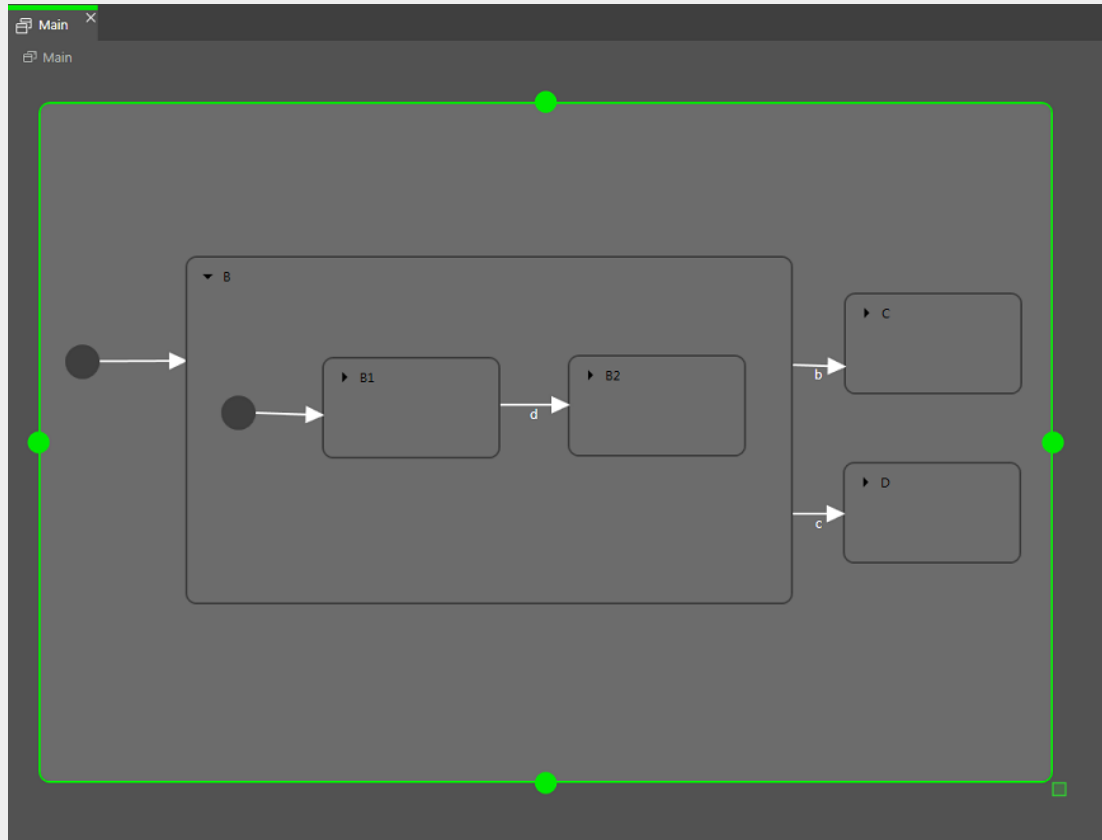


图 6.29. 转换覆盖

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。

6.22.4. 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

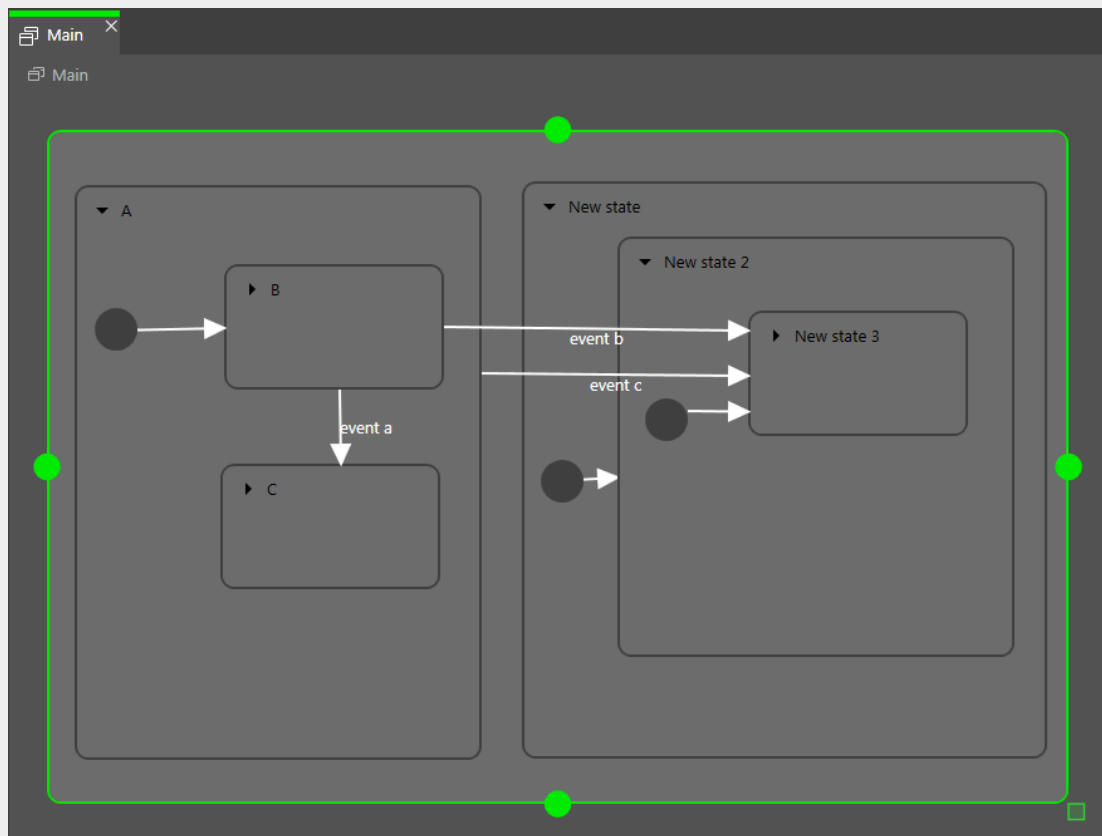


图 6.30. 执行转换

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

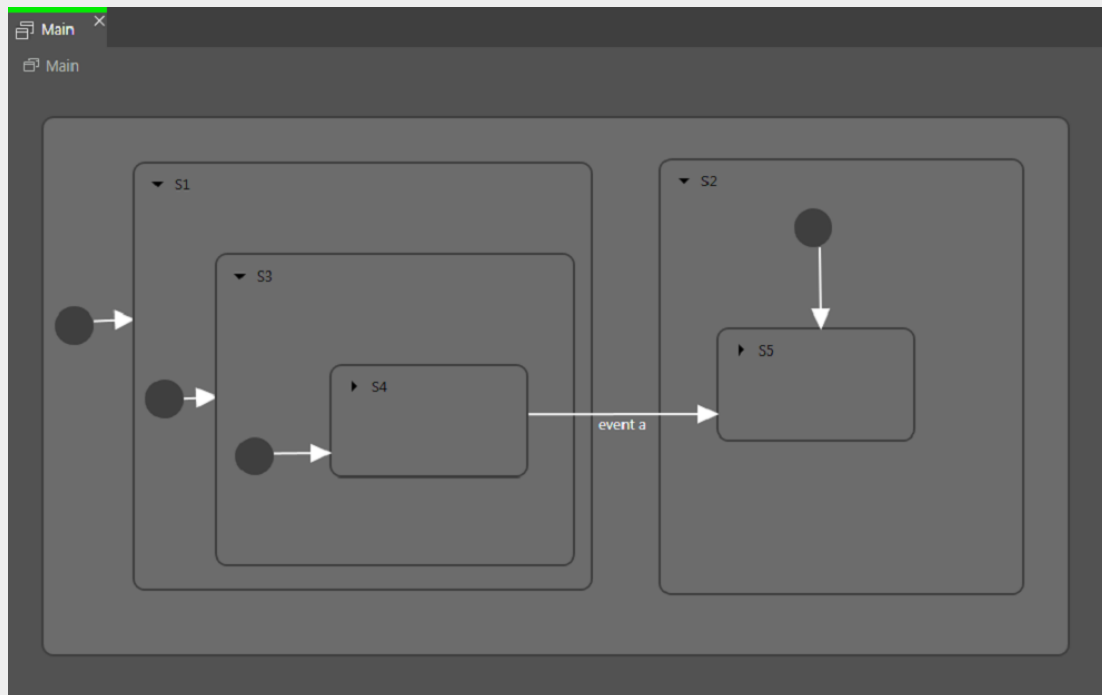


图 6.31. 执行转换

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

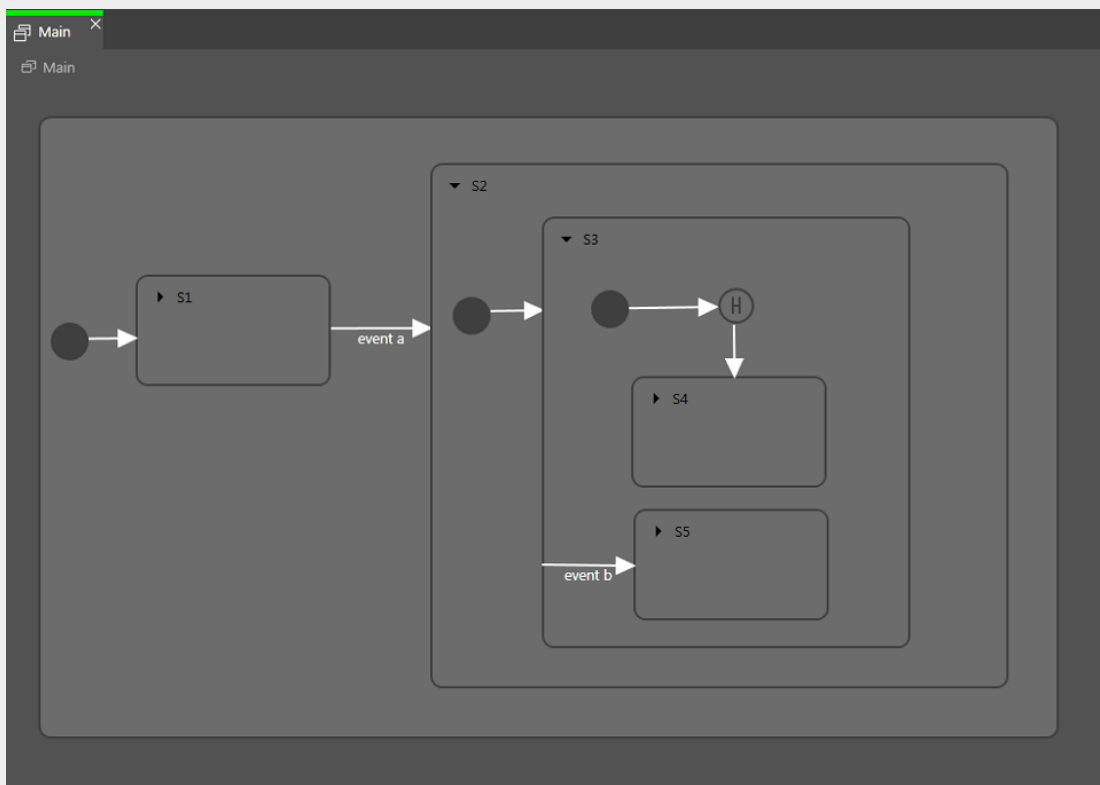


图 6.32. 执行转换

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`。

6.22.5. 与 UML 标记相比的 EB GUIDE 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。

6.22.5.1. 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：

EB GUIDE 中的名称	UML 2.5 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 ^a

^aEB GUIDE 不区分外部和局部转换。

6.22.5.2. 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

6.22.5.3. 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	EB GUIDE 的变通方案
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。

6.23. 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

6.23.1. 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。

6.23.2. 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14.3](#)。

6.23.3. 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。

6.23.4. 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。

6.24. 控件

控件是组成 EB GUIDE 模型的基本图形元素。

可以自定义控件。编辑某个控件的属性可以使该控件满足个人需要。以下元素是触摸或移动时的示例属性：

- ▶ 大小
- ▶ 颜色
- ▶ 布局
- ▶ 行为

可以合并控件。在较小的构建块之外创建了复杂结构。例如，可以组成以下元素的按钮：

- ▶ 椭圆
- ▶ 图像
- ▶ 标签
- ▶ 矩形

可以嵌套控件。在控件层级结构中，这些下级控件称为子代控件，上级控件称为父代控件。

6.24.1. 视图

视图是每个场景中最高的控件。建模时，以下元素位于视图中：

- ▶ 基本控件
- ▶ 3D 控件
- ▶ 动画
- ▶ 控件模板

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，要获得特写视图或查看更多视图，可以增大或减小视图的大小。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%。

注意



对齐主图像中的元素

在 EB GUIDE Studio 中，要均匀对齐如基本控件和 3D 控件等元素，可以将主图像添加到视图中。要将主图像添加到视图中，请点击视图底部的 。要隐藏主图像，选中或清除复选框。如果关闭视图，需要再次添加主图像。

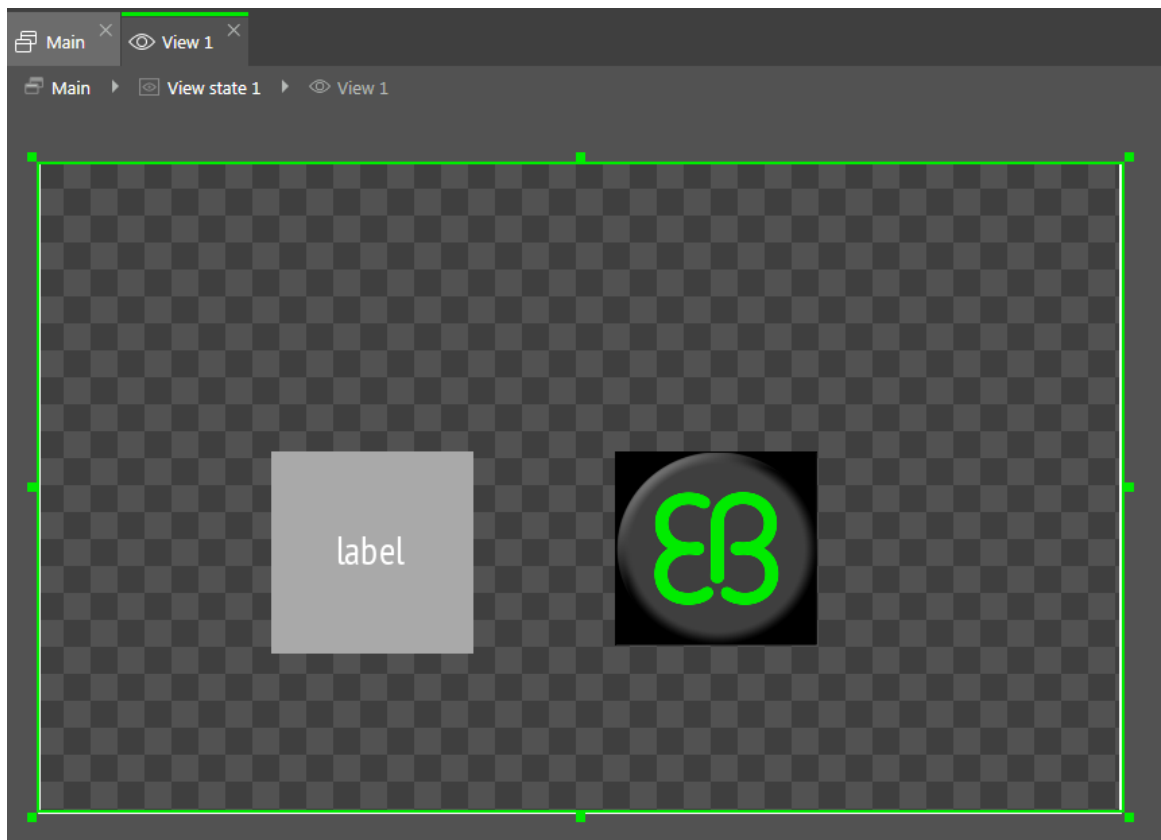


图 6.33. 包含矩形、标签和图像的视图

6.24.2. 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

以下元素是基本控件：

- ▶ Alpha 掩码
- ▶ 动画
- ▶ 容器
- ▶ 椭圆
- ▶ 图像
- ▶ 实例化器
- ▶ 标签
- ▶ 矩形
- ▶ 3D 控件

要显示 3D 图形，使用包含在 "3D 控件" 类别中的控件。"3D 控件" 是以下元素：

- ▶ 环境光
- ▶ 摄像头
- ▶ 定向光
- ▶ 基于图像的光
- ▶ 材料
- ▶ 网格
- ▶ PBR GGX 材料
- ▶ PBR Phong 材料
- ▶ 点光
- ▶ 场景图
- ▶ 场景图节点
- ▶ 聚光

注意**受支持的渲染器**

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

-
- ▶ 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

"定制控件" 类别包含定制的控件，因此只有将定制的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/examples/>。

有关说明，请参见[节8.1](#)。

6.24.3. 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。

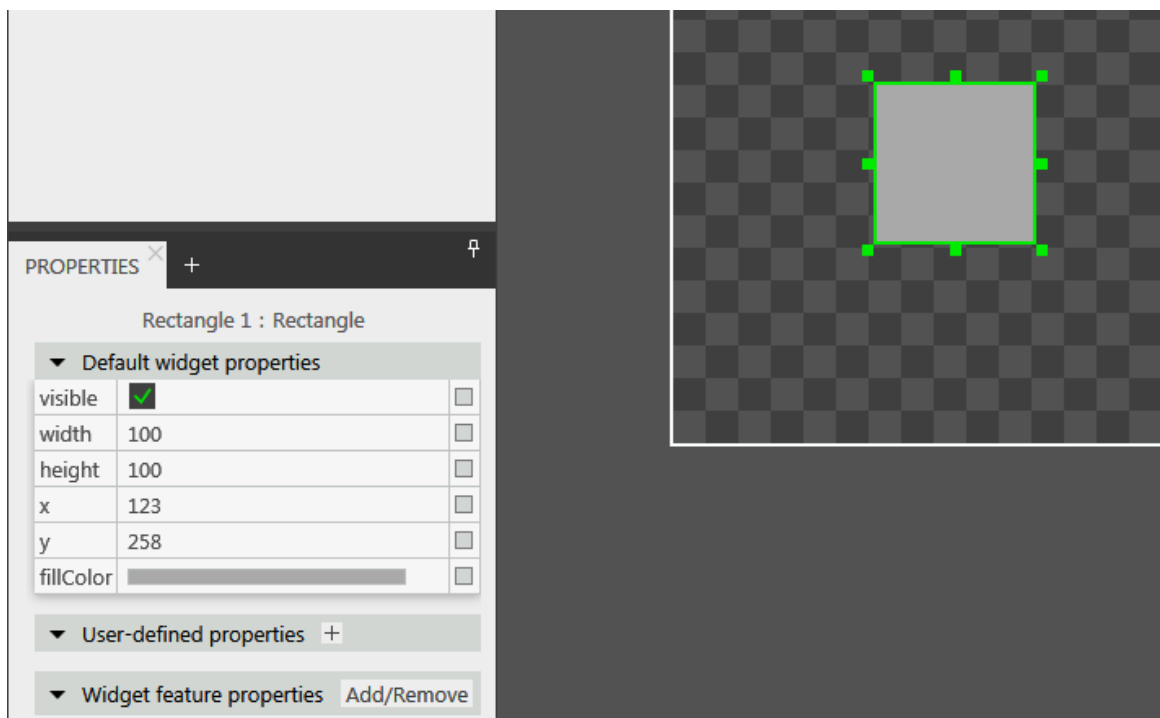


图 6.34. 矩形控件及其属性

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15.9](#)。
- ▶ 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。`touchPolicy` 属性定义触摸的方式，`touchBehavior` 属性定义触摸的区域。

6.24.4. 控件模板

通过使用控件模板，可以定义可在 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。因此通过使用控件模板，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。

注意



模板的父代控件

不能将动画控件用作模板的父代控件。



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 已添加至模板接口。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8.8](#)。

6.24.5. 控件功能

可以使用控件功能在其功能中扩展控件和控件模板。控件功能预定义控件属性。控件功能按类别分组。

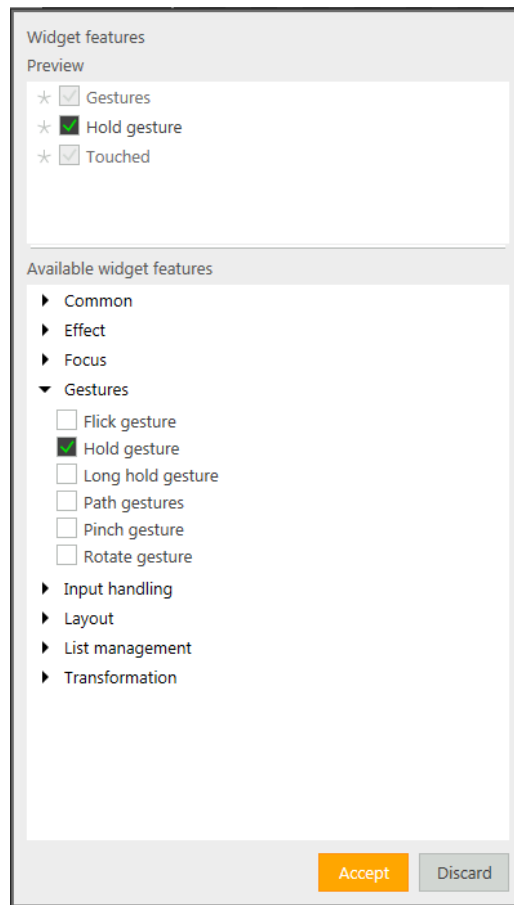


图 6.35. 控件功能

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。
- ▶ 可以将控件功能限制为特定类型的控件。
- ▶ 在运行时刻不能激活或禁用控件功能。

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15.10](#)。

6.24.5.1. 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。

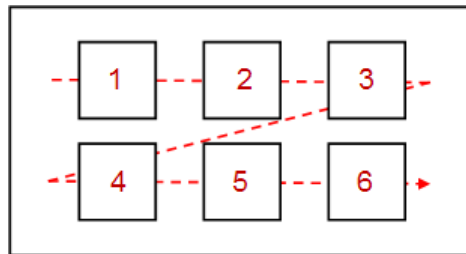


图 6.36. “自动焦点”控件功能的属性

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。

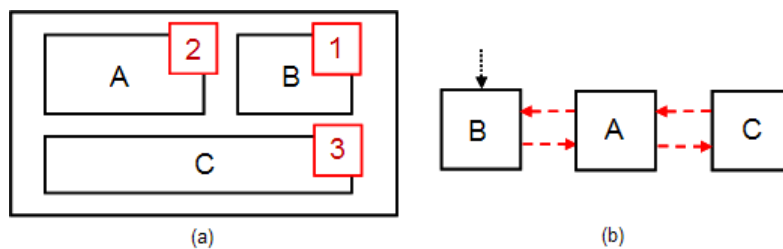


图 6.37. “用户定义焦点”控件功能的属性

在 [图6.37](#) 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。可以嵌套聚焦层次结构。

6.24.5.2. 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14.4](#)。

“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，要显示一个条目或一组条目，您需要多个数据列表。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

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14.1](#)。

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 **+**。

将打开脚本编辑器。

步骤 3

使用 EB GUIDE 脚本 输入操作。

有关背景信息，请参见 [节6.19](#)。

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“退出动作”属性，并点击 。

将打开脚本编辑器。

步骤 3

使用 EB GUIDE 脚本 输入操作。

有关背景信息，请参见[节6.19](#)。

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**。要跳到状态，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。

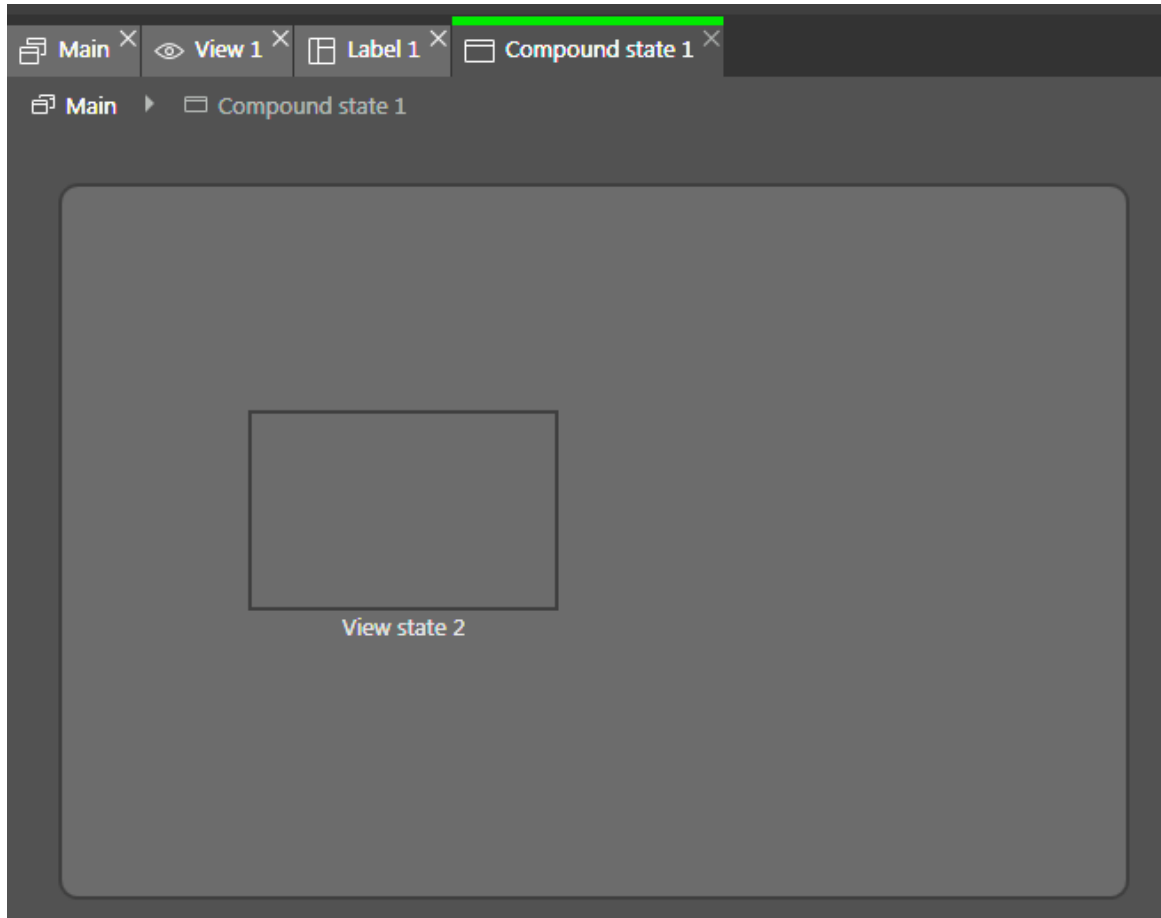


图 7.1. 具有嵌套视图状态的组合状态

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

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 3

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 4

添加自选择状态的传出转换。

步骤 5

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。

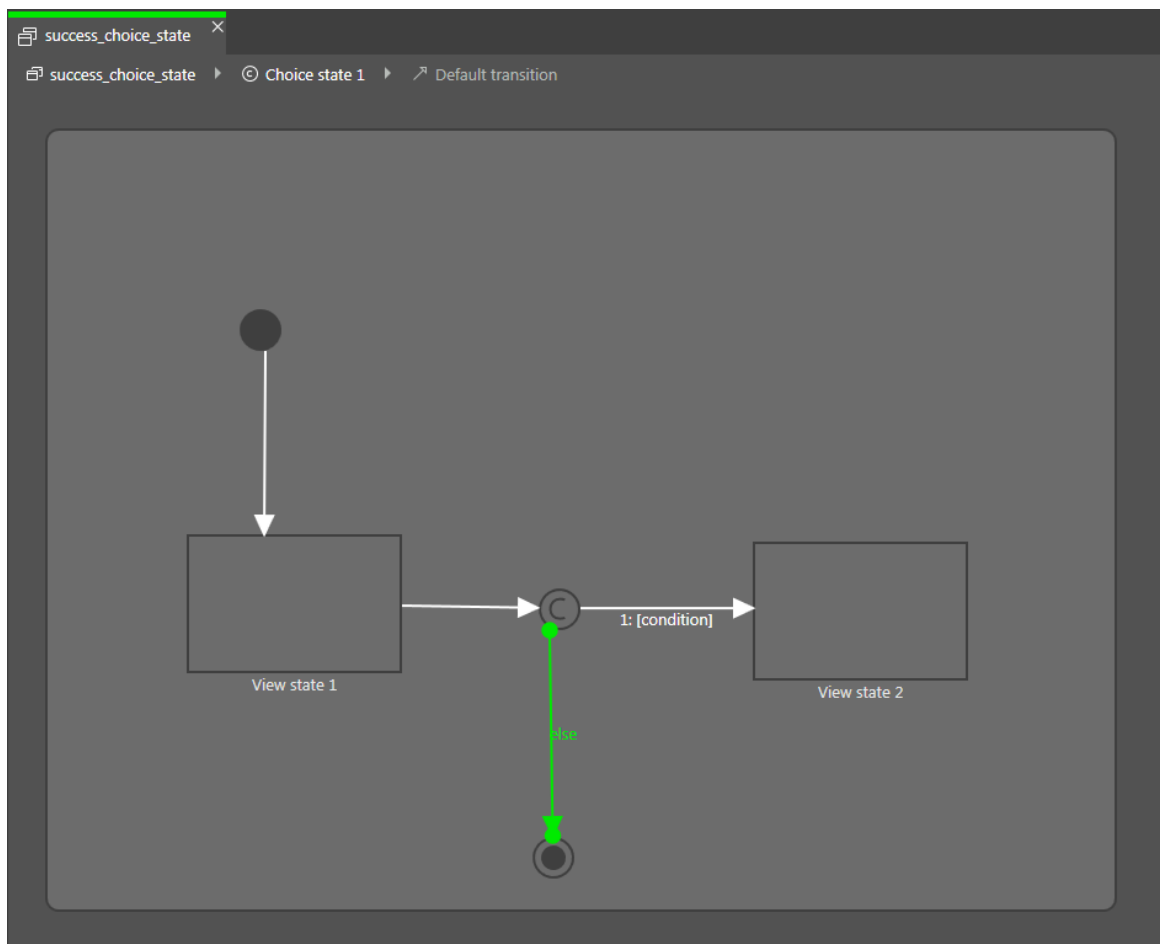


图 7.2. 选择状态及其选择转换

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⁺。

将打开脚本编辑器。

步骤 3

使用 EB GUIDE 脚本 输入操作。

有关背景信息，请参见[节6.19](#)。

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⁺。

将打开脚本编辑器。

步骤 3

使用 EB GUIDE 脚本 输入操作。

有关背景信息，请参见[节6.19](#)。

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。

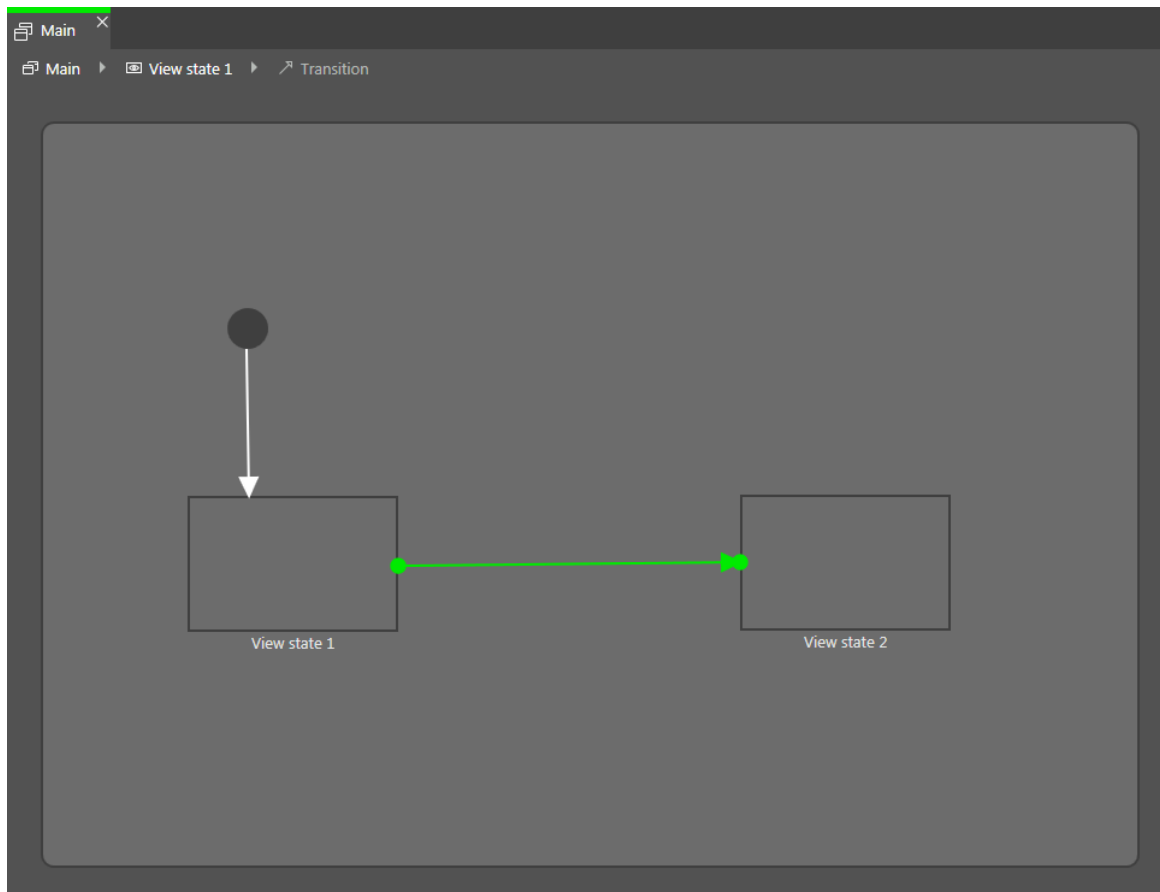


图 7.3. 转换

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。

提示



像样条线转化

可像样条线一样形成转化。要更改转化的形式，请点击转化线并按住鼠标键，同时拖动鼠标。

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，点击“触发器”旁边的框。

步骤 3

选中事件。

要创建新事件，请输入名称，并点击“添加事件”。

事件将添加为转换触发器。

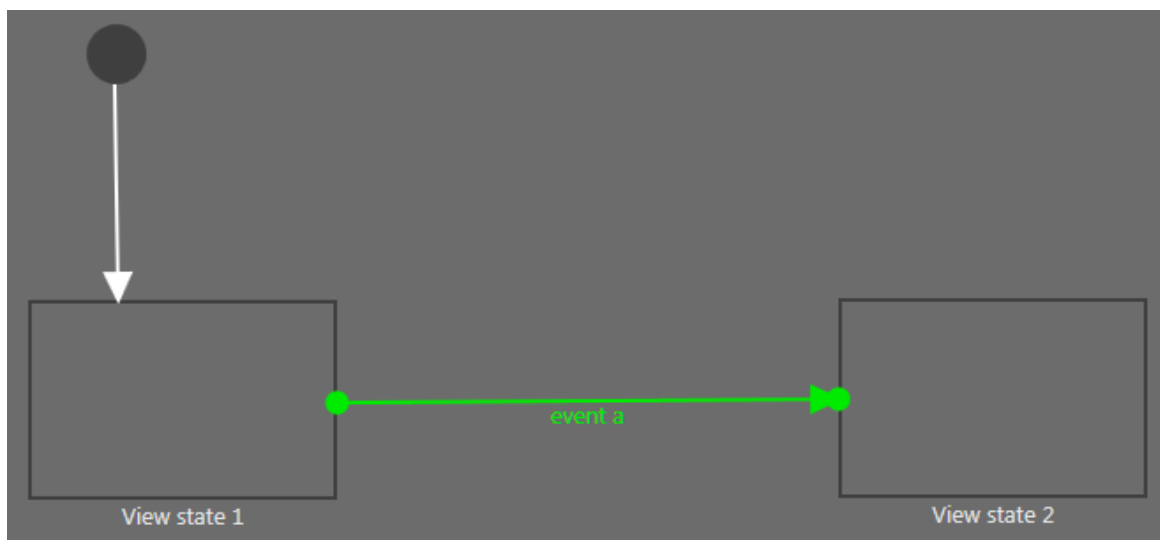


图 7.4. 具有触发器的转换

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 **+**。

将打开脚本编辑器。

步骤 3

使用 EB GUIDE 脚本 输入条件。

有关背景信息，请参见[节6.19](#)。

步骤 4

点击“接受”。

条件将添加至转换。

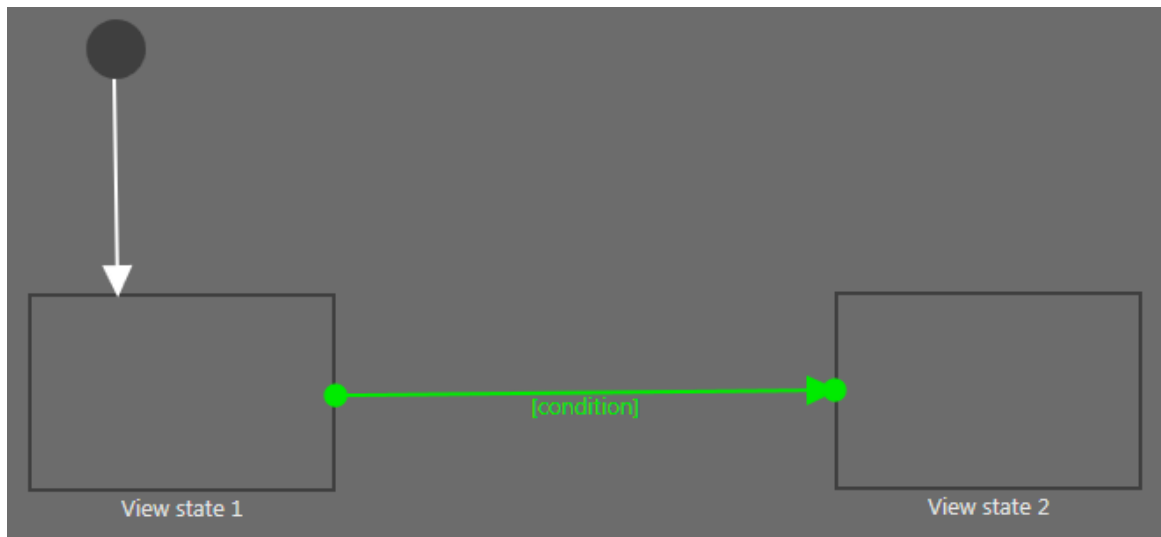


图 7.5. 具有条件的转换

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 **+**。

将打开脚本编辑器。

步骤 3

使用 EB GUIDE 脚本 输入操作。

有关背景信息，请参见 [节 6.19](#)。

步骤 4

点击“接受”。

操作将添加至转换。

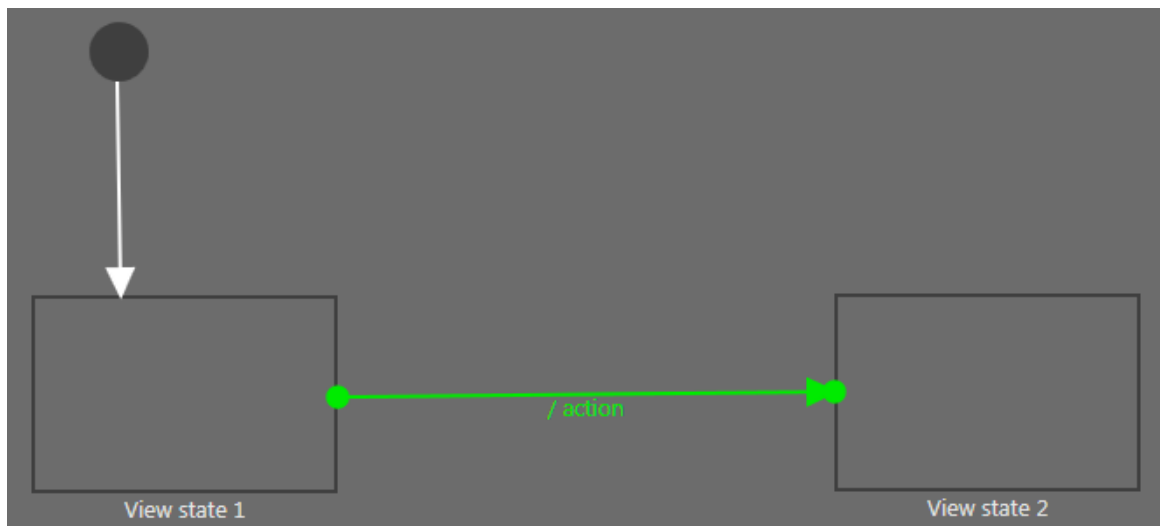


图 7.6. 具有操作的转换

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“内部转换”，并点击 **+**。



| 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

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 模型中查找特定视图或控件，请在搜索框中输入视图或控件的名称或使用 **Ctrl+F**。要跳到视图或控件，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”拖动到状态机中。

视图将随视图状态添加至 EB GUIDE 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15.9.2](#)。

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

在 arcWidth 文本框中输入 0 至 50 之间的宽度。

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。有关受支持的文件类型，请参见[6.18.3](#)。
- 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6.18.3](#)。
- 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6.18.3.1](#)。
- 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**注意****字符替换**

当您将文本输入到标签的 text 属性时，将替换以下字符：

- ▶ 序列 \\ \\ 替换为 \。
- ▶ 序列 \\n 替换为 \n。
- ▶ 如果文字显示在一行中，\n 替换为 空格字符。

**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。

有关更多信息，请参见[节8.4](#)。

**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6.18.1](#)。
- 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。

有关更多信息，请参见[节8.4](#)。

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

选择 lineMapping 并点击 。

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14.4](#)。

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15.9.2.2](#)。

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。

步骤 5

点击“接受”。

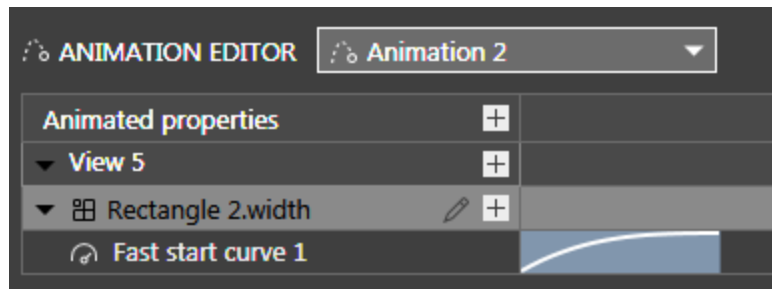


图 8.1. “动画”编辑器和示例曲线

步骤 6

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[第8.2.5](#)。

步骤 7

在条件脚本旁边，选择“值”列并点击 `{}`。

将打开 EB GUIDE 脚本 编辑器。

步骤 8

在“触发器”部分输入下列 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“触发器”脚本中的名称。

步骤 9

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。

步骤 10

要更改曲线行为，请在“属性”元素中编辑动画或曲线属性。

“动画”编辑器显示曲线行为的预览。

有关具体动画示例，请参见[第14.5](#)。

8.1.2.8. 添加带有脚本曲线的动画



获取脚本曲线的输出

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

将该视图重命名为 FirstView 并打开它。

步骤 2

将动画从“工具箱”拖动到 FirstView 中。

步骤 3

在“动画编辑器”中，在“动画属性”旁边点击 **+**，然后点击 FirstView。

菜单将展开。

步骤 4

在 FirstView 下，依次选择 y 属性和“脚本曲线”。

步骤 5

点击“接受”。

动画将添加到“动画编辑器”中。

步骤 6

将新动画重命名为 ScriptCurveMonitoring。

步骤 7

在“属性”元素中，转至“默认控件属性”并在 curve 属性旁边点击 **{}**。随即显示 EB GUIDE 脚本 编辑器。

输入下列 EB GUIDE 脚本:

```
function(v:diff::int, v:t_anim::int)
{
f:trace_string("Diff : "+ f:int2string(v:diff) + " t_anim: " + f:int2string(v:t_anim))
    0::int
}
```

点击“接受”。

步骤 8

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 9

在菜单中，选择 Conditional script。

向动画添加用户定义的 Conditional script 类型的属性。将其重命名为 PlayAnimation。

步骤 10

在 PlayAnimation 属性旁边，点击 `{}`。

这将显示 EB GUIDE 脚本 编辑器。

步骤 11

在“触发器”部分输入下列 EB GUIDE 脚本：

```
function(v:arg0::bool)
{
f:animation_play(v:this)
false
}
```




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 。

在 EB GUIDE Monitor 中，观察“记录器”元素。v:diff 显示动画每 16 毫秒运行一次。v:t_anim 显示自开始以来动画在此时运行的时间。参见下方图片。

```
🔔 traceString 'Diff : 0 t_anim: 0'           12:21:07.779
🔔 traceString 'Diff : 16 t_anim: 16'        12:21:07.779
🔔 traceString 'Diff : 16 t_anim: 32'        12:21:07.779
🔔 traceString 'Diff : 17 t_anim: 49'        12:21:07.779
🔔 traceString 'Diff : 19 t_anim: 68'        12:21:07.779
🔔 traceString 'Diff : 16 t_anim: 84'        12:21:07.779
🔔 traceString 'Diff : 16 t_anim: 100'       12:21:07.779
```

图 8.2. EB GUIDE Monitor 消息

8.1.2.9. 添加 alpha 掩码



添加 alpha 掩码

有关 alpha 掩码的详细信息，请参见[节15.9.2.1](#)。

先决条件：

- \$GUIDE_PROJECT_PATH/<project name>/resources 文件夹包含一个图像。
- 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。使用 OpenGL ES 2.0 或更高版本的渲染器支持 .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。

Alpha 掩码功能不适用于 9-patch 图像。9-patch 图像的处理方式与 PNG 和 JPEG 文件格式的处理方式相同。

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 图形文件的文件夹。

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，其中场景图作为父代节点。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. 导入 .psd 文件到视图中



添加 .psd 文件到视图中

有关背景信息，请参见[节6.17](#)。

先决条件：

- 在 \$GUIDE_PROJECT_PATH/<project name>/resources 或子文件夹中，.psd 可用。
- 内容区域将显示视图。

步骤 1

在“资产”元素中，选择相应的文件夹。

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，则“导航”元素显示新的控件树。顶部元素是以 .psd 文件命名的容器。

在 \$GUIDE_PROJECT_PATH/<project name>/resources 文件夹中，创建了一个子目录，包含着所有提取的图像。

注意**多行**

如果文本图层包含多行的文本，则需要添加控件功能“多行”并相应地编辑属性。有关更多信息，请参见 [节8.3.1](#) 和 [节15.10.1.5](#)。

8.1.5. 从 .psd 文件中提取图像



从 .psd 文件中提取图像

从 .psd 文件中提取图像而不是导入时，不会创建任何控件树。有关背景信息，请参见 [节6.17](#)。

先决条件：

- 在 \$GUIDE_PROJECT_PATH/<project name>/resources 或子文件夹中，.psd 可用。

步骤 1

在“资产”元素中，选择相应的文件夹。

步骤 2

右键点击 .psd 文件，并选择“从 .psd 文件中提取图像”。

显示导入状态信息。

步骤 3

点击“确定”。

在 \$GUIDE_PROJECT_PATH/<project name>/resources 文件夹中，创建了一个子目录，包含着所有提取的图像。子文件夹根据 .psd 文件命名。

8.1.6. 导入 IBL 文件

有关背景信息，请参见[节6.13](#)。



导入 IBL 文件

要导入 IBL 文件，它们必须先转化成 .ebibl 格式。这是使用 IBLGenerator 完成的

先决条件：

- 将创建 EB GUIDE 项目。
- .pfm 或 .hdr 格式的 IBL 文件。
- 您有管理员权限。
- 场景图控件将添加至 EB GUIDE 模型。
- 将导入 3D 文件。

步骤 1

以管理员身份打开命令行提示。

步骤 2

导航至 IBLGenerator 安装路径。它位于 \$GUIDE_INSTALL_PATH\tools\IBLGenerator，tools 文件夹中的 EB GUIDE Studio 安装目录。

步骤 3

输入命令以将文件转化成 .ebibl 格式。可能看起来像这样：

```
IBLGenerator.exe -i yourfile.hdr -o yourfile.ebibl -p latlong -q 1
```

- ▶ -i: 输入文件名
- ▶ -o: 输出文件名
- ▶ -p: 参数化类型。其他类型为立方体和球体。
- ▶ -q: 质量水平。质量水平为 1 表示低质量，10 为最高质量。最高质量水平需要更多的处理时间。

.ebibl 文件放在您提供的文件夹中。

提示



IBLGenerator 帮助

要查看 IBLGenerator 的选项列表，请使用参数 -h 运行它： IBLGenerator.exe -h

步骤 4

将 .ebibl 文件复制到 EB GUIDE 模型的资源文件夹中。现在，您可以在场景图节点中使用 .ebibl 文件。

步骤 5

在 EB GUIDE Studio 中，在“工具箱”元素中，从“3D 控件”中将基于图像的光拖动到场景图节点中。

步骤 6

在“属性”元素中，在 ibl 属性旁边选择 .ebibl 文件。

IBL 文件导入已完成

提示



IBL 的最佳效果

为获得最佳效果，请调整基于图像的光的属性，并使用 PBR GGX 材料或 PBR Phong 材料。

要仅使用基于图像的光照照亮场景，请禁用所有其他光源。

8.1.7. 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，右键点击某个控件。

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。

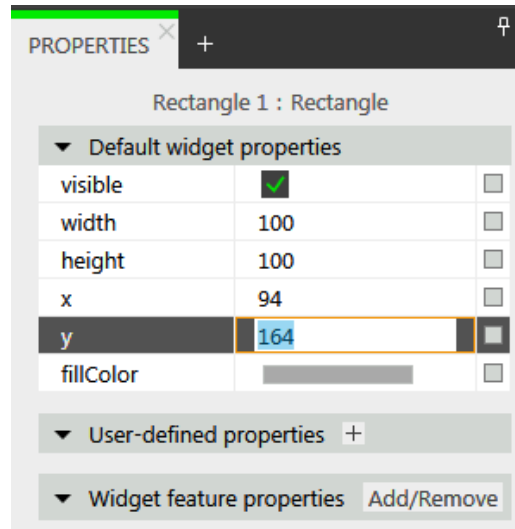


图 8.3. 矩形的属性

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。

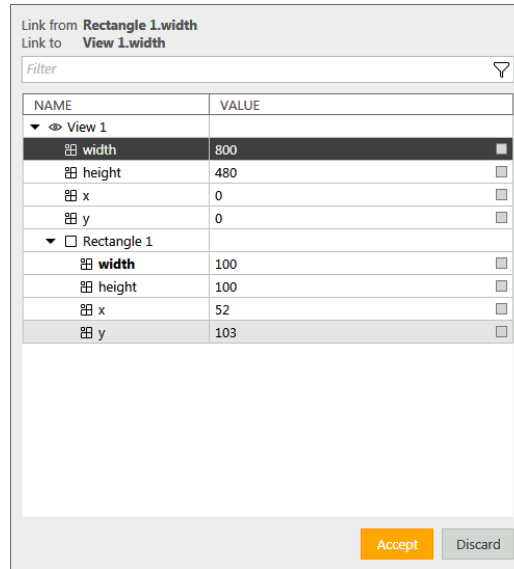


图 8.4. 控件属性之间的链接

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 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”。

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

在数据池项旁边，选择“值”列并点击 `{}`。

将打开 EB GUIDE 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，右键点击属性名称。

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。

8.2.7. 编辑类型列表属性



编辑类型列表属性


有关类型列表属性的更多信息，请参见[节15.3.12](#)。

有关资源管理的更多信息，请参见[节6.18](#)和[节6.6](#)。

先决条件：

- EB GUIDE 模型包含视图状态。
- 视图状态包含控件。
- 控件具有类型列表的属性。

步骤 1

在“属性”元素中，选择类型列表属性，并在旁边点击  按钮。

将打开“编辑”对话框。

步骤 2

要添加新条目到该列表，点击“添加”按钮。

添加了新行。

步骤 3

要编辑值，在表格的“值”列中点击相应的行。

注意**“资产”元素**

如果您的属性是字体列表、ib1 列表、图像列表或网格列表类型，则“编辑”对话框具有附加的“资产”元素。您可以将资产从“资产”元素拖放到“值”列中的相应行。

步骤 4

编辑完列表后，点击“接受”按钮。

对话框将关闭。

8.2.8. 管理控件的顺序和可视性

在 EB GUIDE Studio 中，您可以通过多种方式定义控件的顺序、图层或可视性。可能性如下，是否可以应用取决于您的用例。

如果只想更改视图中哪个控件相对于其他控件位于顶部，请使用“导航”树。在“导航”树中，控件的顺序决定了哪个控件将显示在顶部。如果控件位于同一分支上，则树中位置较高的控件将显示在下方。如果控件位于不同的分支上，则更高级别的控件将显示在顶部。

如果想要有一个显示在其他所有内容之上的弹出窗口，请使用动态状态机。有关说明，请参见[节14.1](#)。

如果您有根据用户选择的内容而更改的复杂菜单，请使用“子代可视性选择”控件功能。使用此控件功能，您可以控制控件的子代控件的可视性。它会覆盖受影响控件的 Visibility 属性。此功能有两种使用方法：

- ▶ 使单个子代控件可见。对于此用例，您只需要控件树中的控件索引。有关说明，请参见[使单个子代控件可见](#)。
- ▶ 使多个子代控件可见。对于此用例，您需要定义子代控件组并提供一个组的 Value。有关说明，请参见[使多个子代控件可见](#)。



使单个子代控件可见

有关控件功能的更多信息，请参见[节8.3](#)。

先决条件：

- EB GUIDE 项目将在 EB GUIDE Studio 中打开。
- EB GUIDE 模型包含控件。

步骤 1

将“子代可视性选择”控件功能添加到父代控件。

步骤 2

在“导航”元素中，查看子代控件所在位置。0 在第一的位置。

步骤 3

在“属性”元素中的 containerIndex 文本框中输入子代控件的位置。

此子代控件现在是唯一可见的子代控件。



使多个子代控件可见

有关控件功能的更多信息，请参见[节8.3](#)。

先决条件:

- EB GUIDE 项目将在 EB GUIDE Studio 中打开。
- EB GUIDE 模型包含三个以上控件。

步骤 1

将“子代可视性选择”控件功能添加到父代控件。

步骤 2

在“属性”元素中，选择 containerMapping 并点击 。

将打开编辑器。

步骤 3

定义控件组和子代控件之间的映射。

“索引”列包含子代控件索引。在“值”列中，输入想要将子代控件映射至的组。

步骤 4

点击“接受”。

步骤 5

在 containerIndex 文本框中，输入应该可见的组的 Value。

该控件组现在可视。没有映射到该组的控件不可视。

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。

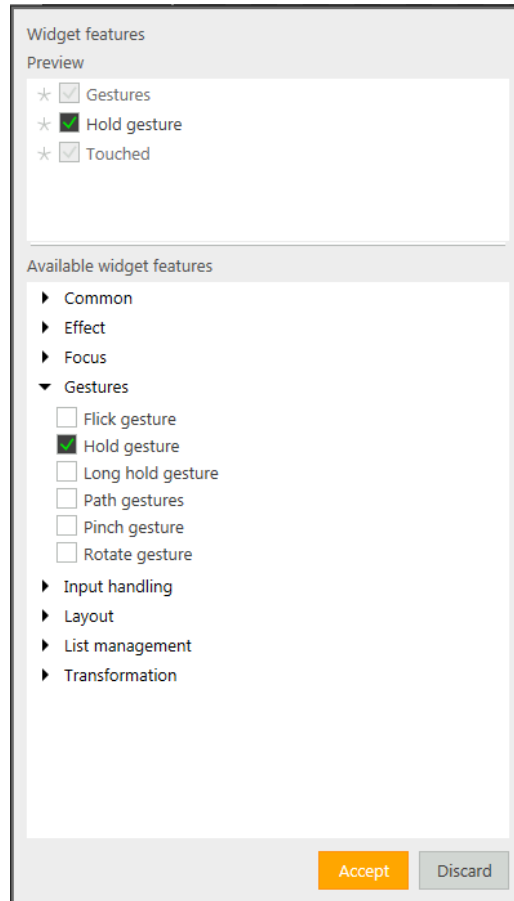


图 8.5. 控件功能对话框

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15.10](#)。

有关教程，请参见下列内容：

- ▶ [节14.3](#)
- ▶ [节14.4](#)
- ▶ [节14.2](#)

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。

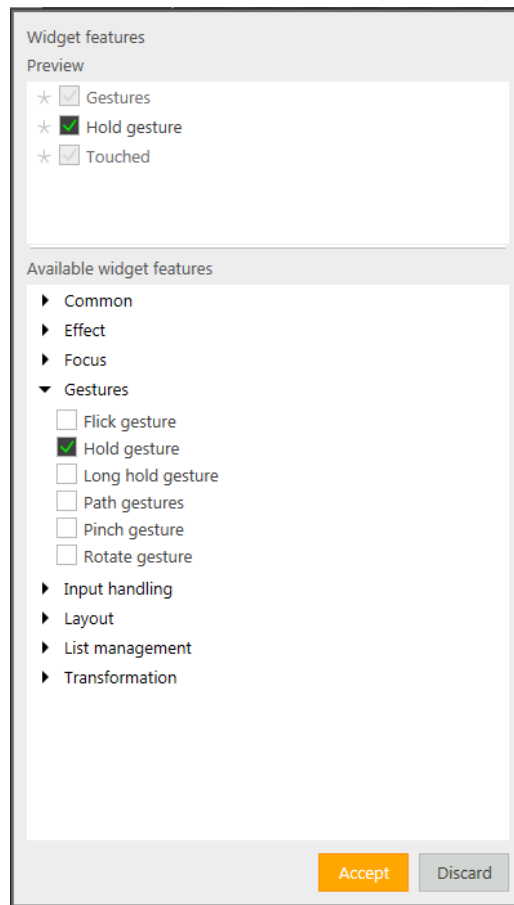


图 8.6. 控件功能对话框

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. 更改字体设置

8.4.1. 更改标签的字体

注意



文本高度、行高和行间距的计算

下图显示如何在 EB GUIDE Studio 中计算文本高度、行高和行间距。更改字体设置时请考虑这一点。有关更多信息，请参见[节8.4.2](#)。

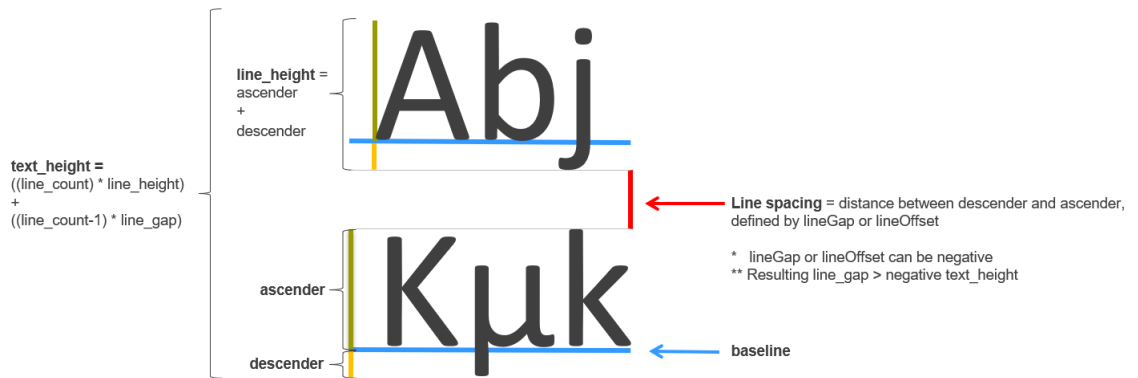


图 8.7. 文本高度、行高和行间距的计算



更改标签的字体

先决条件:

- 文字文件位于 $\$GUIDE_PROJECT_PATH/<project\ name>/resources$ 中。有关受支持的文件类型，请参见[节6.18.1](#)。
- 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。

有关如何使用多字体支持的说明，请参见[节8.4.3](#)。

8.4.2. 更改行间距

每种字体都有默认定义的行间距。您可以使用“字体指标”控件功能中的 `lineGap` 属性更改每个标签的间距。当“多行”控件功能也添加到此标签时，您还可以使用 `lineOffset` 属性定义行间距。可以为同一标签设置两个属性。请注意，在这种情况下，两个属性的设置可能会相互抵消。

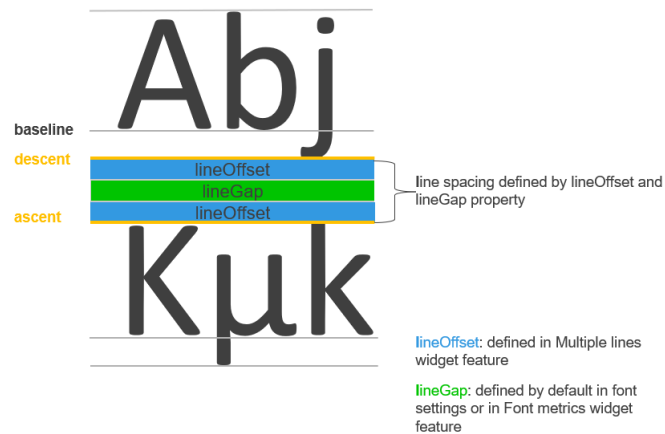


图 8.8. 行间距由 `lineOffset` 定义和 `lineGap`

8.4.2.1. 更改默认行间距

每种字体都有默认定义的行间距。您可以使用“字体指标”控件功能中的 `lineGap` 属性更改每个标签的行间距。

注意

与 `lineOffset` 属性的交互



请注意，当“多行”控件功能也添加到此标签时，行间距也可以通过 `lineOffset` 属性在此处更改，并且这两个属性可能相互抵消。



在“字体指标”控件功能中更改 `lineGap` 属性

下列说明显示如何更改字体的默认行间距。

先决条件：

- 视图包含标签。
- 存在类型字体的属性或数据池项。

步骤 1

选择标签。

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“常用”类别，然后选择“字体指标”。

步骤 4

点击“接受”。

相关控件功能属性将添加至字体，并显示在“属性”元素中。

步骤 5

在“属性”元素中，选择 lineGap 属性并更改值。

8.4.2.2. 更改多行的行间距



在“多行”控件功能中更改 lineOffset 属性

“多行”控件功能启用了标签中的换行符。下列说明显示如何更改多行的行间距。

注意



与 lineGap 属性的交互

请注意，更改 lineOffset 属性会影响“字体指标”控件功能 lineGap 属性中定义的行间距，并且这两个属性可能会相互抵消。

先决条件：

- 该视图包含一个标签，并向其中添加了“多行”控件功能。
- 存在类型字体的属性或数据池项。

步骤 1

选择标签。

步骤 2

在“属性”元素中，转至“多行”控件功能。

步骤 3

更改 lineOffset 属性的值。

8.4.3. 管理多字体支持

有关更多信息，请参见[节6.18.1.2](#)。



为类型字体添加多字体支持

先决条件:


- \$GUIDE_PROJECT_PATH/resources 中有多种字体可用。
- 存在类型字体的属性或数据池项。此属性不是脚本化的值。

步骤 1

要添加多字体支持，请执行以下操作：

- ▶ 如果想要添加多字体支持到控件属性，请转至“属性”元素。
- ▶ 如果想要添加多字体支持到数据池项，请转至“数据池”元素。

步骤 2

在属性或数据池项旁边，点击  按钮。


菜单将展开。

步骤 3

在菜单中，点击“添加多字体支持”。

表格显示在属性或数据池项下方。该表有一个默认的多字体值。

步骤 4

要添加新的多字体值，请点击  按钮。

添加了新行，并根据默认字体填充值。

步骤 5

在该行中，您可以编辑和定义以下内容：

- ▶ 在“优先级”列中，定义字体条目的优先级。
- ▶ 在“字体”列中，定义大小和字体。
- ▶ 在“范围”列中，定义受所选字体影响的 **Unicode** 字符。


步骤 6

重复步骤 2 到 5，直到所有必需的字体的值都已添加。

提示



移除多字体支持

要移除多字体支持，请再次点击  按钮。在打开的菜单中，点击“移除多字体支持”。



为类型字体列表添加多字体支持

有关如歌使用类型列表属性的说明，请参见[节8.2.7](#)。

先决条件：


- \$GUIDE_PROJECT_PATH/resources 中有多种字体可用。
- 存在类型字体列表的属性或数据池项，并具有至少一个字体列表条目。

步骤 1

要添加多字体支持，请执行以下操作：


- ▶ 如果想要添加多字体支持到控件属性，请转至“属性”元素。
- ▶ 如果想要添加多字体支持到数据池项，请转至“数据池”元素。

步骤 2

选择属性或数据池项，并在其旁边点击  按钮。

将打开“编辑”对话框。

步骤 3

在类型字体的条目旁边，点击  按钮。


菜单将展开。

步骤 4

在菜单中，点击“添加多字体支持”。

在属性或数据池项下面添加一个表。该表有一个默认的多字体值。

步骤 5

要添加新的多字体值，请点击  按钮。

添加了新行，并根据默认字体填充值。

步骤 6

在该行中，您可以编辑和定义以下内容：

- ▶ 在“优先级”列中，定义字体条目的优先级。
- ▶ 在“字体”列中，定义大小和字体。
- ▶ 在“范围”列中，定义受所选字体影响的 **Unicode** 字符。


步骤 7

重复步骤 3 到 6，直到所有必需的字体值都已添加。

提示



移除多字体支持

要移除多字体支持，请再次点击  按钮。在打开的菜单中，点击“移除多字体支持”。

步骤 8

编辑完列表条目后，点击“接受”按钮。

对话框将关闭。

8.5. 使用语言支持

要在运行期间更改 EB GUIDE 模型的语言，请添加语言支持和语言相关文本。

有关 EB GUIDE 中语言的更多信息，请参见[图6.14.2](#)。

8.5.1. 向 EB GUIDE 模型中添加语言

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

以下说明展示了如何将语言添加到 EB GUIDE 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” > “语言”。

将显示可用语言。列表中的第一个语言始终是默认语言。

步骤 3

在内容区域中，点击“添加”。

语言已添加至表。该语言将使用标准语言设置作为初始值。

步骤 4

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。

语言已添加至 EB GUIDE 模型。在项目编辑器中，可以在命令区域的“语言”下拉列表框中选择新语言。

在运行期间，您可以在语言之间切换，以查看不同数据池值的效果。有关更多信息，请参见[节14.6](#)。

8.5.2. 向数据池项添加语言支持



向数据池项添加语言支持

以下说明介绍如何向 EB GUIDE 模型的数据池项添加语言支持。


先决条件：

- EB GUIDE 模型包含数据池项。
- 至少需要将两种语言添加至 EB GUIDE 模型。

步骤 1

在项目编辑器中，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“添加语言支持”。

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语言支持，现在可为每个语言定义不同值。

向数据池项添加语言支持现在，可以为此数据池项定义与语言相关的值。有关更多信息，请参见[节14.6](#)。

8.5.3. 删除语言

注意**默认语言**

列表中的第一个语言始终是默认语言，并且无法删除。



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。

8.6. 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。

有关外观的更多信息，请参见 [第6.21](#)。

8.6.1. 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

输入该外观的名称。

新外观已添加至 EB GUIDE 模型。在项目编辑器中，可以在命令区域的“外观”下拉列表框中选择新外观。

8.6.2. 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。

8.6.3. 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。

8.6.4. 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。

8.7. 给视图转变制作动画

8.7.1. 添加入口动画



添加入口动画

下列说明将指导您完成向视图状态添加入口动画的过程。此说明同样适用于退出动画、弹出出现动画和弹出消失动画。有关更多信息，请参见 [节6.2.2](#) 和 [节15.9.1](#)。

先决条件：

- 视图状态和视图添加至 EB GUIDE 模型。
- 打开 "VTA" 元素。

步骤 1

在 "VTA" 元素中，点击 **+**。

步骤 2

在上下文菜单中，选中“入口动画”。

“动画”编辑器将打开。

“属性”元素显示已添加入口动画的属性。

步骤 3

为依赖视图中所有可用的控件属性制作动画。

在“动画”编辑器中定义入口动画。

8.7.2. 添加更改动画



添加更改动画

下列说明将指导您完成向视图状态或视图模板添加更改动画的过程。

先决条件:

- 打开 "VTA" 元素。
- "主要" 状态机包含两个试图状态。

步骤 1

在"主要"状态机中选择 View state 1。

步骤 2

在 "VTA" 元素中, 点击 **+**。

步骤 3

在上下文菜单中, 选中"更改动画"。

对话框将打开。

步骤 4

选中 View 2。

步骤 5

点击"接受"。


目标视图的名称显示在更改动画名称旁边。

"动画"编辑器将打开。

"属性"元素显示已添加更改动画的属性。

开始视图显示在项目编辑器中。

步骤 6

要编辑目标视图, 请在 "VTA" 元素中点击 。

步骤 7

要添加动画属性, 请点击"动画"编辑器中的 **+** 以添加各自的类别。

为依赖视图中所有可用的控件属性制作动画。

8.7.3. 重新排列动画



重新排列动画

下列说明将指导您完成重新排列视图状态或视图模板的动画的过程。此说明同样适用于退出动画、更改动画、弹出出现动画和弹出消失动画。

先决条件：

- 打开 "VTA" 元素。
- 已添加多个入口动画。

步骤 1

在 "VTA" 元素中，选择要先启动的入口动画旁边的优先次序文本框。

步骤 2

将入口动画的值更改为 0。

入口动画的顺序已更改。首先启动已编辑的入口动画，所有后续的条目以一递增。

值 0 表示首先播放的动画。后续值表示评估动画以后续顺序播放。仅播放具有最高优先级和满足条件的入口动画。

8.8. 重复使用控件

有关模板的更多信息，请参见 [第6.24.4](#)。

8.8.1. 添加模板



添加模板

步骤 1

在"模板"元素中，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的名称或使用 **Ctrl+F**。要跳到模板，请在命中列表中双击它。

8.8.2. 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。

8.8.3. 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”。

8.8.4. 删除模板



删除模板

步骤 1

在“模板”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。

8.9. 启用抗锯齿

有关背景信息，请参见[节6.3](#)。

8.9.1. 全局启用抗锯齿



全局启用抗锯齿

先决条件：

- 有一个 EB GUIDE 模型。

步骤 1

在“项目中心”中选择“配置” > “配置文件”。

“配置文件”菜单打开。

步骤 2

在“场景”选项卡的 antiAliasing 下拉列表框中，选择要为场景设置的抗锯齿模式。

现在为整个 EB GUIDE 模型配置了抗锯齿模式。

8.9.2. 为场景图启用抗锯齿



全局启用抗锯齿

先决条件：

- EB GUIDE 模型包含场景图。

步骤 1

选择场景图。



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

展开“3D”类别，选择“抗锯齿模式”控件功能，然后点击“接受”。

在“属性”元素中，显示“抗锯齿模式”控件功能。

步骤 4

从 antiAliasing 下拉列表框中，选择该场景图所需的模式。

此场景图的抗锯齿模式已配置。

9. 处理数据

注意



默认窗口布局

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，从命名空间下拉列表框中选择要添加事件的命名空间。

步骤 2

点击 **+**。

事件已添加至表中。

步骤 3

对事件进行重命名。

步骤 4

要更改事件 Id，请转至属性元素，并在“属性”元素，并在 Event ID 文本框中输入 ID。

提示



复制和查找事件

或者，您可以使用上下文菜单或 **Ctrl+C** 和 **Ctrl+V** 复制和粘贴现有事件。为防止重复，粘贴的事件具有与复制的事件不同的事件 ID。

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**。要跳到事件，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 使用 ID 在运行时间发送和接收事件。



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 EB GUIDE TF 的事件位置

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. 将键映射到事件

若要触发事件动作，可以按键盘上的键，或者例如按目标设备上的按键或使用旋转按钮。

为了对键盘按键做出反应，模型需要为这些键事件定义映射。

EB GUIDE GTF 在 C++ 头文件中为每个键定义一个数字代码。有关代码编号，请参见 `$INSTALL_PATH$/platform/win64/include/gtf/displayfactory/inputmapper/KeyConstants.h`。



将键映射到事件

有关事件处理的更多信息，请参见 [节6.10](#) 和 [节15.5](#)。

先决条件:

- 事件组“键” (ID 10) 已添加到“项目中心” > “配置” > “事件组”。
- 已添加事件。

步骤 1

在 KeyConstant.h 文件中查找要映射的键的十六进制代码编号。

步骤 2

计算十进制代码编号。

步骤 3

在“事件”元素中，在“分组”列中选择“键” (ID 10)。

步骤 4

在“属性”元素中，在 Event ID 文本框中输入计算得出的十进制代码编号。

所选的键现在已被映射到事件。



例 9.1.
将 **F1** 键映射到事件

F1 的内部十六进制代码编号为 12。

12 的十进制代码编号为 18。

转至“属性”元素并在 Event ID 文本框中输入 18。

F1 键现在已被映射到事件。

注意



EB GUIDE Monitor 中的不可见事件

事件本身未显示在 **EB GUIDE Monitor** 中，但由事件触发的 **EB GUIDE** 脚本 会做出反应。

有关十进制代码编号的更多信息，请参见[节15.5.1](#)。

9.5. 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，选择相应的命名空间。

步骤 2

右键点击事件并从上下文菜单中选择“删除”。

事件将被删除。

9.6. 添加数据池项



添加数据池项

步骤 1

在“数据池”元素中，从命名空间下拉列表框中选择要添加数据池项的命名空间。

步骤 2

点击 **+**。

菜单将展开。

步骤 3

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 4

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** 来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**。要跳到数据池项，请在命中列表中双击它。

9.7. 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

选择 Value 列，并点击 。

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。

9.8. 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。
- 数据池项不依赖于外观。
- 数据池项未链接。
- 数据池项不具有多字体支持。

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


在数据池项旁边，选择“值”列并点击 。

将打开 EB GUIDE 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。

9.9. 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”。

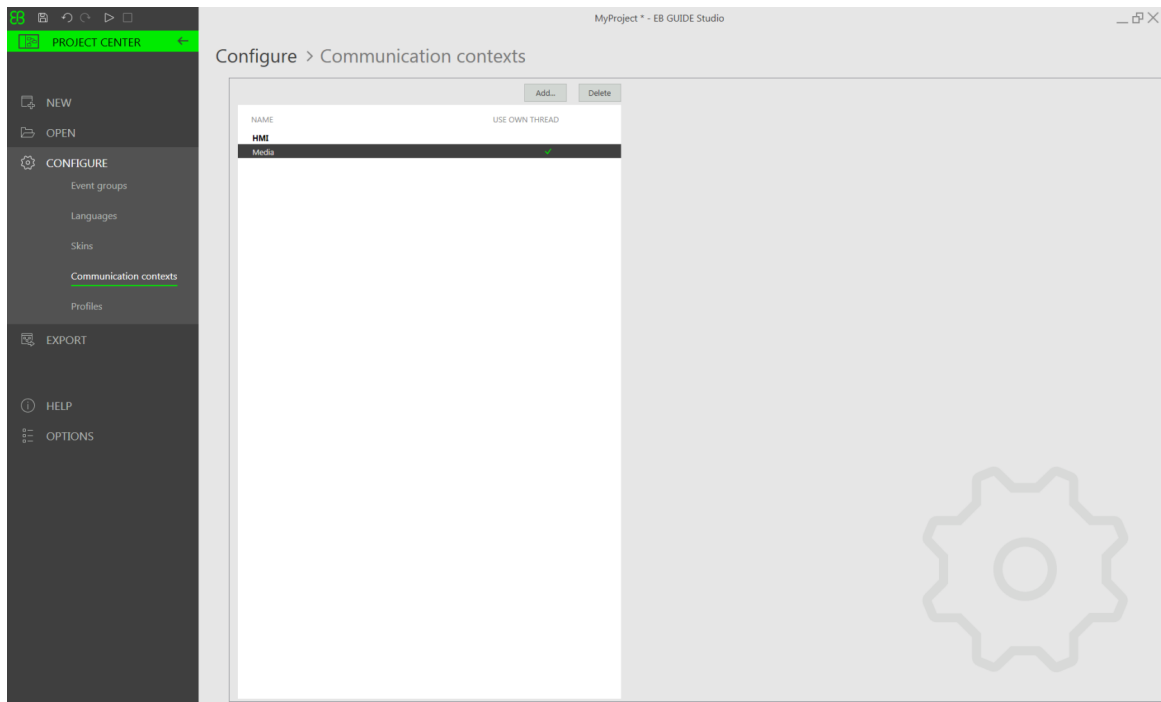


图 9.1. 通信环境 Media。

9.10. 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。
- 数据池项不依赖于外观。
- 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”。

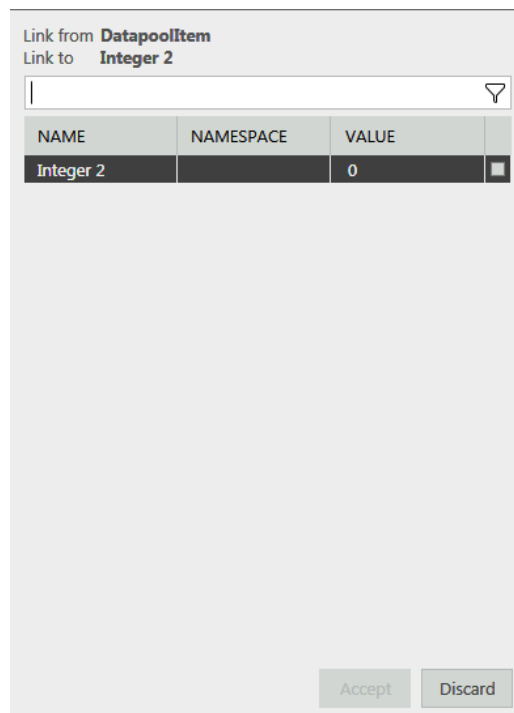



图 9.2. 数据池项之间的链接

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。

9.11. 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，选择相应的命名空间。

步骤 2

右键单击数据池项并从上下文菜单中选择“删除”。

数据池项将被删除。

9.12. 将模型元素添加到模型接口

有关模型接口的更多信息，请参见[第6.16](#)。

有关如何使用模型接口的说明，请参见[第10.9](#)。

9.12.1. 将事件添加到模型接口



将事件添加到模型接口

先决条件：

- EB GUIDE 模型包含事件。

步骤 1

转至“事件”元素。

步骤 2

如果只有一个默认模型接口，右键单击事件并选择“添加至模型接口”。

如果有多个模型接口，右键单击事件，选择“模型接口”，然后再选择应该包含此事件的模型接口。

现在，该事件已包含在模型接口中。事件左侧的彩色栏条表示已将其添加到模型接口。

9.12.2. 将数据池项添加到模型接口

注意



条件脚本

无法将条件脚本类型的数据池项添加到任何模型接口。



将数据池项添加到模型接口

先决条件:

- EB GUIDE 模型包含数据池项。

步骤 1

转至“数据池”元素。

步骤 2

如果只有一个默认模型接口，右键点击数据池项目并选择“添加至模型接口”。

如果有多个模型接口，右键点击项，选择“模型接口”，然后再选择应该包含此项的模型接口。

现在，该数据池项已包含在模型接口中。数据池项左侧的彩色栏条表示已将其添加到模型接口。

9.13. 使用命名空间

有关命名空间的更多信息，请参见 [节6.6](#) 和 [节6.15](#)。

9.13.1. 添加命名空间



添加命名空间

可以添加新的命名空间作为根命名空间或任何现有命名空间的子级。

步骤 1

由于“命名空间”元素在 EB GUIDE 的默认窗口布局中不可用，请执行以下操作：

在命令区中点击“布局”，然后选择“命名空间”。

显示“命名空间”元素。

步骤 2

在“命名空间”元素中，点击 **+**。

命名空间已添加至树。

步骤 3

对命名空间重命名

提示



移动命名空间

要移动命名空间，将其拖动到根命名空间或其他命名空间。注意命名空间的命名约定以避免命名冲突。

9.13.2. 将模型元素添加到命名空间

有关如何将事件添加到命名空间的说明，请参阅[节9.1](#)。

有关如何将数据源项添加到命名空间的说明，请参阅[节9.6](#)。

9.13.3. 在命名空间之间移动模型元素



在命名空间之间移动模型元素

先决条件：

- 将模型元素、事件或数据源项添加到命名空间。
- 至少存在两个命名空间。

步骤 1

要移动事件，请转到“事件”元素。

要移动数据源项，请转到“数据源”元素。

步骤 2

选择相应的命名空间，然后右键点击模型元素。

步骤 3

从上下文菜单中，选中“移动至命名空间...”。

对话框将打开。

步骤 4

选择目标命名空间并点击“接受”。

模型元素已移动至目标命名空间。

注意**移动模型元素**

或者，您可以将模型元素拖动到另一个名称空间。

9.13.4. 删除命名空间

警告**删除命名空间**

删除命名空间时，还会删除此命名空间包含的所有模型元素。

注意**根命名空间**

根命名空间无法删除。

**删除命名空间**

先决条件：

- 命名空间已添加至 EB GUIDE 模型。

步骤 1

在“命名空间”元素中，右键点击命名空间。

步骤 2

在上下文菜单中，点击“删除”。

该命名空间将被删除。

10. 处理项目

注意



默认窗口布局

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

注意



无效项目

如果 EB GUIDE 项目无效，EB GUIDE Studio 无法将其打开。将显示一条错误消息，并在以下位置创建包含错误说明的日志文件 `$GUIDE_PROJECT_PATH/<project name>/<project name>_LoadingErrorLog.txt`。

有关更多信息，请参见[节6.8.2](#)。

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 项目。

步骤 1

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

步骤 2

双击 EB GUIDE 项目文件。

项目将在 EB GUIDE Studio 中打开。

10.2.2. 在 EB GUIDE Studio 中打开项目



在 EB GUIDE Studio 中打开项目

先决条件：

- 将创建 EB GUIDE 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 项目文件。EB GUIDE 项目文件的文件扩展名为 .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、状态或状态机），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. 在 EB GUIDE Studio 验证 EB GUIDE 模型



在 EB GUIDE Studio 验证 EB GUIDE 模型

在“问题”元素中，EB GUIDE 显示了以下内容：


- ▶  错误
- ▶  警告

有关更多信息，请参见[节6.8.2](#)。

步骤 1

要展开“问题”元素，请点击“问题”。

步骤 2

在“问题”元素中，点击 。

将显示错误和警告列表。

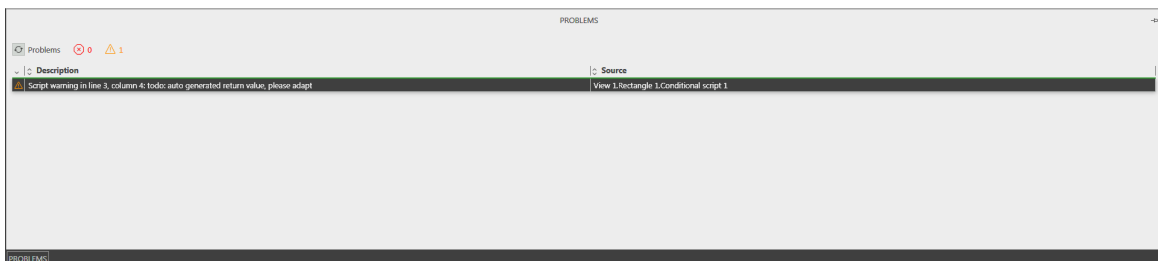


图 10.1. 问题元素

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。

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 EB GUIDE Studio 导出 EB GUIDE 模型



使用 EB GUIDE Studio 导出 EB GUIDE 模型

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 使得为一个 EB GUIDE 模型创建不同配置文件成为可能。

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 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。将 EB GUIDE TF 分为可执行的文件和一组库，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 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>/bin` 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

单击“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。

有关示例，请参见 EB GUIDE GTF 用户指南的参考部分。

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 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。

有关示例，请参见 EB GUIDE GTF 用户指南的参考部分。

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 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15.7](#)。

10.8. 导出和导入语言依赖项文本

通过 EB GUIDE 可采用用户首选的语言显示文本。为实现此操作，请将语言支持添加到 EB GUIDE 模型。然后，您可以将依赖于语言的文本导出到 .xliff 文件，翻译好再将它们导回到模型中。

注意



项目和语言特定 ID

为每个项目和语言对创建一个独特的字母数字 sourcelanguageid 和 targetlanguageid。这些 ID 可防止从其他项目或目标语言无意中导入 .xliff 文件。此外，每个数据池项都会接收每种语言唯一的字母数字 ID。

在 EB GUIDE Studio 模型中更改语言或数据池项目，并导出文本进行翻译时，由于特定 ID，翻译后的文本仍可以分配到正确的数据池项目和语言。参见导入日志文件查看是否正确分配了所有数据池项。

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。

有关更多信息，请参见[节10.4.1.1](#)。



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其传递给本地化服务提供商。

先决条件：

- 要翻译的语言已添加至 EB GUIDE 模型。有关更多信息，请参见[节8.5.1](#)。
- 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。有关信息，请参见[节8.5.2](#)。
- EB GUIDE 模型没有错误和警告。

步骤 1

单击 。

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文件夹。

步骤 6

点击“选择文件夹”。

结果：导出开始。文件将保存在所选文件夹中。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

导出文件的结构和内容



- ▶ 对于您在项目中心中选择的每个语言，将导出单独的 .xliff 文件。
- ▶ 源语言是默认语言。因此，如果文本尚未翻译，target-language 元素始终包含源文本。

有关 EB GUIDE 模型中语言依赖项文本和 .xliff 文件的更多信息，请参见[节6.14](#)。

10.8.2. 导入语言依赖项文本

10.8.2.1. 使用 EB GUIDE Studio 导入语言依赖项文本



使用 EB GUIDE Studio 导入语言依赖项文本

先决条件：

- 至少有一个翻译的 .xliff 文件可用于所选的 EB GUIDE 模型。
- 要翻译的数据池项和目标语言仍然存在。

- 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

点击“导入”。

对话框将打开。

步骤 4

选择存储已翻译的 .xliff 文件的文件夹。

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”。

现在，所有具有语言支持的数据池项都显示相应的语言依赖项文本。参见日志文件了解更多有关导入的信息。

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：

- 至少有一个翻译的 .xliff 文件可用于所选的 EB GUIDE 模型。
- 发送到翻译的数据池项仍然存在。
- 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 -l <language file> -o "$GUIDE_PROJECT_PATH/project_name.ebguide"`。

如果导入成功，EB GUIDE 模型被更改。如果导入失败，EB GUIDE 模型不更改。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. 使用模型接口

以下各部分中的说明将指导您完成向模型接口添加数据池项和事件以及导出和导入接口的过程。有关背景信息，请参见[节6.16](#)。

10.9.1. 创建模型接口



创建模型接口

先决条件：

- 将创建 EB GUIDE 项目。

步骤 1

点击 。


将打开项目中心。

步骤 2

在导航区中，点击“配置”>“模型接口”。

列出了当前项目的所有模型接口。

步骤 3

点击 。

使用默认名称创建用户定义的模型接口。

步骤 4

重命名模型接口。

步骤 5

模型接口旁边有一个彩色栏条。此栏条也位于事件或数据池项旁边。用于表示它们所属的模型接口。要更改此栏条的颜色，右键点击接口，然后选择“选择颜色”。

10.9.2. 导出模型接口



导出模型接口

先决条件:

- 将创建 **EB GUIDE** 项目。
- 模型接口将添加至 **EB GUIDE** 项目。
- 事件或数据池项将添加至模型接口。有关如何将模型元素添加到模型接口的说明, 请参见[节9.12](#)。

步骤 1

点击 。

将打开项目中心。

步骤 2

在导航区中, 点击“配置” > “模型接口”。

列出了当前项目的所有模型接口。

步骤 3

选择要导出的模型接口。

步骤 4

点击 。

随即打开一个资源管理器窗口。

步骤 5

将带有接口信息的 .json 文件保存在您选择的位置。

您已导出模型接口。现在可以使用此 .json 文件将模型接口导入至另一个 **EB GUIDE** 模型。

10.9.3. 导入模型接口



导入模型接口

先决条件:

- 创建了两个 **EB GUIDE** 项目。
- 从第一个项目中导出模型接口, 并创建了一个带有接口信息的 .json 文件。
- 第二个项目在 **EB GUIDE Studio** 中打开。

步骤 1

点击 。

将打开项目中心。

步骤 2

在导航区中，点击“配置” > “模型接口”。

将打开“模型接口”菜单。

步骤 3

要导入模型接口，请点击 。

随即打开资源管理器窗口。

步骤 4

导航至 .json 的位置，选择文件，然后点击“打开”。

接口已导入至项目中。在“模型接口”菜单中，可以看到接口和每个接口提供多少个数据池项和事件。

注意



重复事件 ID

事件组中的事件 ID 必须是唯一的。当同时导入多个模型接口时，如果重复的事件 ID 属于不同的模型接口但在同一事件组内，则会发生验证错误。由于您无法在 EB GUIDE Studio 中更改导入事件的事件 ID，请恢复导入，在源模型中更改事件 ID，然后再次导出和导入。建议事先为所有 EB GUIDE 模型定义事件 ID 范围。

10.9.4. 更新导入的模型接口

要更新导入的模型接口，在源 EB GUIDE 模型中更改，然后导出更新后的模型接口，并再次将其导入。

有关如何导出和导入模型接口的更多信息，请参见[节10.9.2](#)和[节10.9.3](#)。

10.9.5. 删除模型接口

注意



默认模型接口

无法删除默认模型接口。



删除模型接口

先决条件:

- EB GUIDE 模型具有用户定义的或导入的模型接口。

步骤 1

点击 。

将打开项目中心。

步骤 2

在导航区中，点击“配置” > “模型接口”。

列出了当前项目的所有模型接口。

步骤 3

右键点击要删除的模型接口，然后在上下文菜单中点击“删除”。

如果此模型接口是导入的，则从 EB GUIDE 模型中删除该模型接口和所有包含的模型元素。

如果此模型接口是创建的，则只从 EB GUIDE 模型中删除该模型接口。添加到此模型接口的所有模型元素仍然存在。

11. 使用 EB GUIDE Monitor

有关 EB GUIDE Monitor 的更多信息，请参见[节6.9](#)和[节6.6.2](#)。

注意



默认窗口布局

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 Monitor 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. 作为独立程序启动 EB GUIDE Monitor

在模拟运行 EB GUIDE 模型过程中，EB GUIDE Monitor 在 EB GUIDE Studio 中自动启动。但您也可以将 EB GUIDE Monitor 作为独立应用程序启动。



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

双击 Monitor.exe

EB GUIDE Monitor 将启动。



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 Monitor.exe

EB GUIDE Monitor 将启动。

11.2. 配置 EB GUIDE Monitor



连接 EB GUIDE Monitor

先决条件:

- EB GUIDE Monitor 已启动。
- EB GUIDE 模型正在运行。

步骤 1

要将连接设置更改为 EB GUIDE GTF, 点击

步骤 2

在“主机”中输入主机名, 在“端口”中输入端口地址。

步骤 3

点击

EB GUIDE Monitor 已连接且状态按钮为绿色:

提示



要连接 EB GUIDE Monitor, 点击

EB GUIDE Monitor 现已断开连接且状态按钮为红色:



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”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。

注意



EB GUIDE Monitor 继承 **EB GUIDE Studio** 的语言

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。EB GUIDE Monitor 的显示语言与 EB GUIDE Studio 相同。



重置 EB GUIDE Monitor 窗口的大小

EB GUIDE Monitor 窗口的大小及其在屏幕上的位置都针对每个单独的 EB GUIDE 项目分别存储在 C:\<user>\AppData\Local\Temp\eb_guide_simulation_export\<project>。

先决条件：

- EB GUIDE Monitor 已启动，且其原始位置和窗口大小已操作。

步骤 1

要将大小和位置重置为默认值，请删除 C:\<user>\AppData\Local\Temp\eb_guide_simulation_export\<project> 中的 monitor_layout.xml 和 monitor_settings.xml。

步骤 2

重新启动模拟，或者，如果 EB GUIDE Monitor 作为独立应用程序启动，则重新启动 EB GUIDE Monitor。

将使用默认大小和位置值创建新 monitor_layout.xml 和 monitor_settings.xml 文件。



编辑日志消息数

在 \$INSTALL_PATH\tools\monitor\Monitor.exe.config 配置文件中，可以定义 EB GUIDE Monitor 在记录器达到其内存限制时的行为。

步骤 1

打开 \$INSTALL_PATH\tools\monitor\Monitor.exe.config 配置文件。

步骤 2

要定义“记录器”元素中显示的条目数，更改 limit 值。

步骤 3

要定义达到限制时要删除的条目数，更改 removeCount 值。

步骤 4

启动 EB GUIDE Monitor。

EB GUIDE Monitor 使用来自自己更改的配置文件的新设置。

11.3. 加载配置到 EB GUIDE Monitor



将配置文件加载到 EB GUIDE Monitor

先决条件:

- EB GUIDE Monitor 作为独立应用程序启动。
- EB GUIDE 模型将导出至 \$EXPORT_PATH。
- 在 \$EXPORT_PATH 中, monitor.cfg 配置文件已创建。

步骤 1

选择“文件” > “加载配置”。

对话框将打开。

步骤 2

转至 \$EXPORT_PATH 并选中 monitor.cfg 配置文件。

步骤 3

点击“打开”。

项目的配置已加载到 EB GUIDE Monitor 中。



将最近的配置文件加载到 EB GUIDE Monitor

先决条件:

- EB GUIDE Monitor 作为独立应用程序启动。
- 最近已使用一个或多个配置文件。

步骤 1

选择“文件” > “最近配置文件”。

对话框将打开。

步骤 2

转至相应的位置并选择一个配置文件。

配置文件将加载到 EB GUIDE Monitor。

注意



EB GUIDE Monitor 与 **EB GUIDE GTF** 断开连接

在加载新配置文件之前，EB GUIDE Monitor 会自动与当前 EB GUIDE GTF 断开连接。

EB GUIDE Monitor 重新连接并加载新配置文件。

11.4. 在 EB GUIDE Monitor 中触发事件



在 EB GUIDE Monitor 中触发事件

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 的“事件”元素中，点击 **+**。


对话框将打开。

步骤 2

选择要触发的事件，并点击“接受”。

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

如果事件有参数，请执行以下操作：


步骤 4.1

单击  以展开参数。

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。

11.5. 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 的“数据池”元素中，点击 **+**。

对话框将打开。

步骤 2

选择数据池项并点击“接受”。

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。

11.6. 在 EB GUIDE Monitor 中使用脚本

11.6.1. 为 EB GUIDE Monitor 编写脚本文件

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 11.1. **EB GUIDE Monitor** 脚本文件示例

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"); //⑥
    await monitor.Write(string.Format("State {0} left",
        leftState.Name));

    var enteredState = await monitor.WaitForStateEnter
        ("Main", "State 2"); //⑦
    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"); //⑧
    await monitor.Write(string.Format("State Machine {0} started",
        startedStateMachine.Name));

    var stoppedStateMachine = await monitor.WaitForStateMachineStop
        ("Dynamic state machine 1"); //⑨
    await monitor.Write(string.Format("State Machine {0} stopped",
        stoppedStateMachine.Name));
}
}

public class Advanced
{
    public async Task CaptureScreenshot(IMonitorContext monitor) //⑩
    {
        // 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) //⑪
    {
        // 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");
}
}

namespace Elektrobot.Guide.Monitor.Scripting //12
{
    using Elektrobot.Guide.Monitor.Scripting.MonitorContext;
    using System.Threading;
    using System.Threading.Tasks;
    using Elektrobot.Guide.Monitor.Model.Event;
    using Elektrobot.Guide.Monitor.Model.Value;

    public class MonitorScripts
    {
        public async Task FireEventInNamespace(IMonitorContext monitor)
        {
            string[] namespacePath = { "Foo_namespace", "go_to_view2" };
            var identifier = new QualifiedIdentifier(namespacePath);

            await monitor.FireEvent(identifier);
        }

        public async Task FireEventInNestedNamespace(IMonitorContext monitor)
        {
            string[] namespacePath =
                { "Foo_namespace", "sub_namespace_under_foo", "go_to_view4" };
            var identifier = new QualifiedIdentifier(namespacePath);

            await monitor.FireEvent(identifier);
        }

        public async Task FireEventInRootNamespace(IMonitorContext monitor)
        {
            await monitor.FireEvent("go_to_view3");
        }
    }
}
```

```
}  
}  
}
```

- 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
- ⓬ 使用命名空间的示例

11.6.2. 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11.6.1](#)。

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。

11.7. 导出和导入观察列表

用于项目的事件和数据池项存储在观察列表中。要将项目用于其他项，您可以将观察列表导出为 .xml 文件，然后将其导入到您的新项目。



导出所有观察列表

先决条件：

- EB GUIDE Monitor 已启动。
- 已使用存储在“数据池”或“事件”选项卡中的项设置了 EB GUIDE 模型。

步骤 1

要导出所有观察列表，选择“文件” > “导出所有观察列表”。

对话框将打开。

步骤 2

选择目标文件夹并输入文件名。

所有数据池项和事件将导出。



导出单个观察列表

先决条件：

- EB GUIDE Monitor 已启动。
- 已经使用存储在“数据池”或“事件”元素中的项设置好 EB GUIDE 模型。

步骤 1

转到包含想要导出的项目的“数据池”或“事件”元素。

步骤 2

要保存此标签的项列表，请点击“导出”。

对话框将打开。

选择目标文件夹并输入文件名。

该元素的数据池项或事件将导出。



导入观察列表

先决条件:

- EB GUIDE Monitor 已启动。
- 导出的观察列表已经可用。

步骤 1

要导入观察视图，请选择“文件” > “导入观察列表”。

对话框将打开。

步骤 2

选择一个要导入的观察列表文件。

数据池项或事件将在新的“数据池”或“事件”元素中打开。

注意



布局未导入

只会导入数据池项和事件，不会导入布局。

新打开的“数据池”和“事件”元素将使用默认布局。

12. 扩展 EB GUIDE Studio

本章提供了有关为 EB GUIDE Studio 创建扩展所需了解的概念信息，以及说明和示例。如果您在实施扩展时遇到问题，请联系我们的支持。参见[章3](#)。

12.1. 概念

12.1.1. 依赖注入

EB GUIDE Studio 的构建考虑了依赖注入。为了管理和解决依赖关系，EB GUIDE Studio 使用了 .NET Framework 的托管扩展框架 (MEF)。

根据属性编程模型注册和注入依赖。扩展可以通过导出接口实现来提供新功能，并且可以通过导入接口来使用现有的 EB GUIDE Studio 功能。

要将依赖导入到类中，请将 `ImportingConstructor` 属性添加到构造函数中，然后将所需的依赖添加为构造函数参数。当请求您的类实例时，MEF 将尝试满足所有依赖关系并调用标记的构造函数。

请考虑到，MEF 仅满足自身导出的类的依赖关系。要导出类，将 `Export` 属性添加到类。



例 12.1. Export 属性的示例

下列示例以一般方式展示了 `ImportingConstructor` 和 `Export` 属性的用法。导出 `IFooService` 接口的实现，该实现又导入对 `IBarService` 接口的依赖。

```
[Export(typeof(IFooService))]  
internal class MyFooService : IFooService  
{  
    [ImportingConstructor]  
    public MyFooService(IBarService barService) {}  
}
```

有关托管扩展框架的详细概述，请参见<https://docs.microsoft.com/en-us/dotnet/framework/mef/>。有关属性编程模型的更多详细信息，请参见<https://docs.microsoft.com/en-us/dotnet/framework/mef/attributed-programming-model-overview-mef/>。

12.1.2. EB GUIDE 模型扩展

为了确保模型的一致性，必须按顺序执行所有修改。这是通过在任务计划程序上计划所有修改来实现的，该任务计划程序依次执行动作。除此之外，对模型的所有修改都必须在会话中执行。会话有两个目的：

- ▶ 将修改分组到单个变更集中。结果，可以在一个步骤中撤消在一个会话中执行的所有操作。
- ▶ 告诉底层存储哪些元素已更改。这意味着仅将在会话中执行的更改实际存储到文件系统。

警告



数据丢失

如果不使用会话来修改 EB GUIDE 模型，则可能会损坏模型并导致数据丢失。

EB GUIDE Studio 提供一个 API 来简化任务计划和会话的处理。

- ▶ `ITaskSchedulerProvider` 提供对必须执行所有模型修改的任务计划程序的访问。
- ▶ `IEventService` 接口提供了用于创建和修改模型中事件的方法。
- ▶ `ExecuteModelAction` 是 `ITaskSchedulerProvider` 接口中的扩展方法。它在正确的任务计划程序上计划修改并创建会话。此扩展方法的第二个参数是执行实际模型修改的委托。该会话是在调用委托之前创建的，并在委托执行后自动提交。这意味着作为用户，您仅需调用 `ExecuteModelAction` 并指定修改模型的方式，而无需自己进行任务计划和会话处理。



例 12.2.

将扩展应用于 EB GUIDE 模型

下列示例显示了如何更改前面的示例以将扩展应用到 EB GUIDE 模型。前面的示例已更改，因此现在可以导入 `ITaskSchedulerProvider` 和 `IEventService` 依赖。

```
[Export(typeof(IFooService))]
internal class MyFooService : IFooService
{
    private readonly ITaskSchedulerProvider _schedulerProvider;
    private readonly IEventService _eventService

    [ImportingConstructor]
    public MyFooService(
        ITaskSchedulerProvider schedulerProvider,
        IEventService eventService)
    {
        _schedulerProvider = schedulerProvider;
        _eventService = eventService;
    }

    public async Task ModifyModel(IProjectContext projectContext)
    {
        await _schedulerProvider.ExecuteModelAction(
```

```
projectContext,
session => _eventService.CreateEvent(
    session,
    projectContext,
    projectContext.Project.RootNamespace,
    "My Event");
}
}
```

如上所述，此示例还将 `IEventService` 接口作为依赖导入。强烈建议您使用现有的服务接口来修改模型，而不是直接修改模型元素。

如果没有适合您需求的现有服务，则需要遵循一些规则以确保模型的一致性。会话由 `IWriteSession` 接口表示。您必须在会话上进行正确的调用才能将更改存储在模型中。

- ▶ 如果可能，请先构建模型元素树，然后再将其添加到模型中。这样，可以避免不必要的模型更新通知，从而提高性能。
- ▶ 必须保存新创建的元素。如果创建了整个元素树，请使用 `SaveHierarchy` 方法。它会自动保存所有子元素。根据经验，请始终对新创建的元素调用 `SaveHierarchy`。
- ▶ 使用 `Save` 方法保存修改后的元素。避免在现有元素上使用 `SaveHierarchy` 方法，因为这会导致更改集中出现许多不必要的条目，从而导致性能下降。
- ▶ 使用 `Delete` 方法删除已移除的元素。与其直接调用此方法，不如导入 `IModelElementService` 接口并使用其 `DeleteElements` 方法。此方法以递归方式删除元素及其所有子元素。
- ▶ 不要显式调用 `Commit` 方法。调用 `Commit` 由上面提到的 `ExecuteModelAction` 扩展方法处理。

12.1.3. EB GUIDE Studio 用户界面扩展

EB GUIDE Studio 使用 Windows Presentation Foundation (WPF) 作为其用户界面框架。有关 WPF 的详细信息，请参见 <https://docs.microsoft.com/en-us/dotnet/framework/wpf/>。此外，EB GUIDE Studio 的用户界面层在构建时考虑了 Model-View-ViewModel (MVVM) 模式。这意味着，在大多数情况下必须提供视图模型和视图实现以扩展 EB GUIDE Studio 用户界面。对于诸如菜单之类的通用用户界面元素，存在现有的视图，但是对于自定义用户界面元素，则需要视图实现。

自定义视图实现通过使用 MEF 导出来提供。应该与 WPF 一起使用的视图模型必须遵循某些约定才能正常工作。视图模型应实现 `INotifyPropertyChanged` 接口。WPF 的数据绑定引擎使用此接口将视图模型的更改反映到视图。如果未在自定义视图模型中实现此接口，则绑定会仅将初始值传输到视图层，并且不会传播任何更新。为了简化自定义视图模型的创建，EB GUIDE Studio 提供了一个名为 `ViewModel` 的基类，实现了 `INotifyPropertyChanged` 接口。



例 12.3.

自定义视图模型的实现

下列示例显示了一个视图模型，该模型将其 `Text` 属性的更改传播到视图。它还具有可以绑定到视图的 `command` 属性。例如，每次执行命令时，通过单击绑定按钮将更改 `Text` 属性，并更新视图。

```
internal class MyViewModel : ViewModel
{
    private string _text;

    public string Text
    {
        get => _text;
        set => SetProperty(ref _text, value);
    }

    public ICommand DoSomethingCommand { get; }

    public MyViewModel()
    {
        Text = "Initial text";
        DoSomethingCommand = new DelegateCommand(DoSomething);
    }

    private void DoSomething()
    {
        Text = "Did something";
    }
}
```

通过为视图模型创建 **WPF** `DataTemplate` 来提供视图。`DataTemplate` 是在资源字典中的 **XAML** 中定义的。您可以通过导出 `IResourceProvider` 接口的实现来提供自定义资源字典。



例 12.4.

`DataTemplate` 用于自定义视图模型

下列代码段显示了如何为自定义视图模型提供 `DataTemplate`。资源字典是在程序集 `MyAssembly` 中名为 `Resources.xaml` 的文件中定义的。资源提供者实现将 `URI` 返回到 **XAML** 文件。

```
[Export(typeof(IResourceProvider))]
internal class MyResourceProvider : IResourceProvider
{
    public IEnumerable<Uri> GetResourceUris()
    {
```

```
var uri = new Uri(  
    @"MyAssembly;Component/Resources.xaml",  
    UriKind.Relative);  
  
return new[] { uri };  
}  
}
```

**例 12.5.**

ResourceDictionary

下列代码段显示了如何在 XAML 中使用 DataTemplate 创建 ResourceDictionary。

```
<ResourceDictionary  
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  
    xmlns:mynamespace="clr-namespace:MyNamespace">  
    <DataTemplate  
        DataType="{x:Type mynamespace:MyViewModel}">  
        <mynamespace:MyView />  
    </DataTemplate>  
</ResourceDictionary>
```

IResourceProvider 返回的 URI 必须遵循 Pack URI 语法，以便 WPF 可以找到并加载它们。有关 Pack URI 语法的更多信息，请参见<https://docs.microsoft.com/en-us/dotnet/framework/wpf/app-development/pack-uris-in-wpf>。

IResourceProvider 实现提供的资源字典在应用程序级别加载。这意味着字典中的所有资源都是全局可用的。为避免使用显式资源密钥命名与现有资源冲突，应在资源密钥之前加上扩展名。例如，特殊按钮的自定义按钮样式可以这样命名：MyPlugin.MySpecialButton。

从 EB GUIDE 微型网站下载 EB GUIDE 扩展示例：<https://www.elektrobit.com/ebguide/examples/>。有关说明，请参见随附的 EB GUIDE Studio Tutorial Using EB GUIDE Studio examples.pdf 文件。

12.2. 创建扩展项目



创建 Visual Studio 项目

先决条件:

- EB GUIDE Studio 安装文件夹是可写的。C:\Program Files\Elektrobit 下的默认安装文件夹受 Windows 保护。对于扩展开发，请使用不受写保护的安装目录。
- 已安装 Visual Studio 2017 或更高版本。
- 已安装 .NET Framework 4.7 SDK。

步骤 1

打开 Visual Studio。

步骤 2

创建新项目。

步骤 3

在“新建项目”对话框中，选择一个项目模板：

对于核心扩展，请选择 Class Library (.NET Framework)。

对于用户界面扩展，请选择 WPF User Control Library (.NET Framework)。

步骤 4

输入扩展项目的名称，然后单击“确定”。

将创建项目。

步骤 5

在“解决方案资源管理器”中右键单击该项目，然后选择“管理 NuGet 包...”。

步骤 6

单击“浏览”，然后搜索 Elektrobit.Guide.Studio。

步骤 7

对于核心扩展，请安装 Elektrobit.Guide.Studio.Core 包。

对于用户界面扩展，请安装 Elektrobit.Guide.Studio.Ui 包。

现在，您的项目已经设置好，可以进行编码了。

12.3. 禁用程序集的复制



禁用程序集的复制

构建项目时，默认情况下，引用的 NuGet 包中的所有程序集都会复制到您的输出文件夹中。对于 EB GUIDE 提供的程序集，NuGet 包不是必需的，因为这些程序集已经是 EB GUIDE 安装的一部分，将用于执行扩展。您可以按照以下步骤调整 NuGet 包参考以禁用程序集的复制。

先决条件：

- 使用 Visual Studio 创建了一个插件项目。

步骤 1

在文本编辑器中打开扩展项目的项目文件 (.csproj)。

步骤 2

找到 EB GUIDE NuGet 包的 PackageReference 条目。

步骤 3

添加 IncludeAssets 属性并将其设置为值 compile。

步骤 4

保存项目文件并在 Visual Studio 中重新加载项目。

现在，PackageReference 条目应类似于以下示例片段：

```
<PackageReference Include="Elektrobit.Guide.Studio.Ui">  
  <Version>6.9.0</Version>  
  <IncludeAssets>compile</IncludeAssets>  
</PackageReference>
```

12.4. 运行扩展



运行扩展

先决条件：

- EB GUIDE Studio 安装文件夹是可写的。C:\Program Files\Elektrobit 下的默认安装文件夹受 Windows 保护。对于扩展开发，请使用不受写保护的安装目录。
- 您的扩展项目被设置为解决方案的启动项目。

**步骤 1**

打开扩展项目的项目设置，然后转至“调试”标签。

步骤 2

从 **EB GUIDE Studio** 安装目录中选择 Start external program 和 Studio.exe。

步骤 3

转至“构建事件”选项卡，然后输入以下后构建脚本：

```
copy /Y $(TargetPath) <extension dir>  
copy /Y $(TargetDir)$(TargetName).pdb <extension dir>
```

成功构建后，此脚本会将您的扩展复制到 **EB GUIDE Studio** 插件文件夹中。

步骤 4

将 <extension dir> 替换为 **EB GUIDE Studio** 安装的正确路径。

对于核心扩展，请使用 \$GUIDE_INSTALL_PATH\studio\lib\core。

对于用户界面扩展，请使用 \$GUIDE_INSTALL_PATH\studio\lib\ui。

现在，您可以从 **Visual Studio** 运行扩展。

13. 最佳实践

本章中的主题按字母顺序排序。

注意



默认窗口布局

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”。

13.1. 最佳实践：处理脚本化的值

每次读取属性时，转换为脚本值的数据池项和控件的属性都会导致运行 EB GUIDE 脚本。在某些用例中，要最大限度地减少 EB GUIDE 脚本运行次数并提高性能，请执行以下操作：

1. 如果您有脚本值，请重置属性类型：使用未转换为脚本值但具有其纯值的属性。有关更多信息，请参见 [节 9.8](#) 和 [节 8.2.5](#)。
2. 要计算和设置当前值，请添加 Conditional script 类型的用户定义属性。请考虑只有在必要时才执行此操作，例如，在初始化或输入属性更改时。

14. 教程

注意



默认窗口布局

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”。

14.1. 教程：添加动态状态机

注意



默认窗口布局

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”拖入视图中。

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。

您已向视图中添加两个矩形。您已更改矩形的外观。

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。

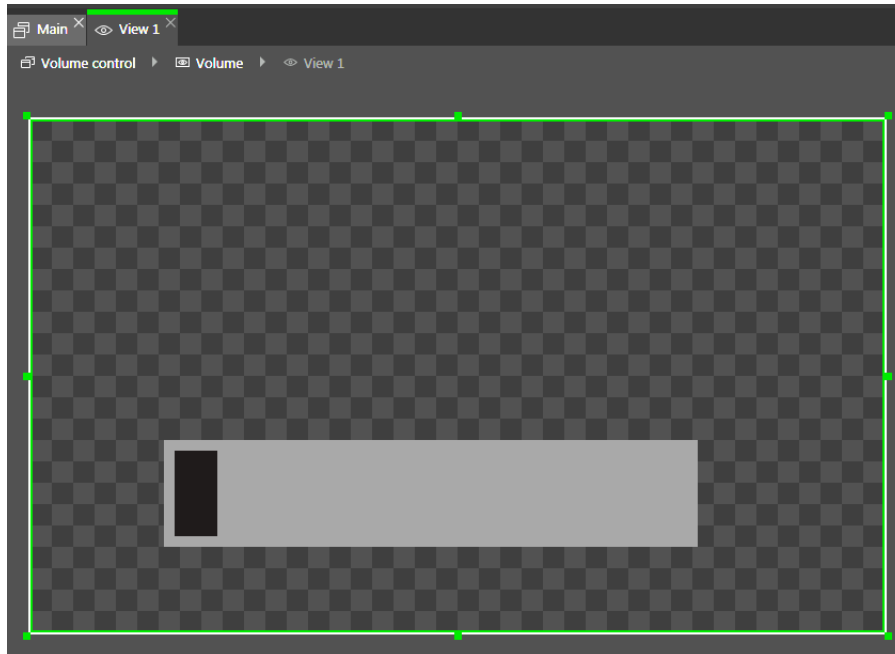


图 14.1. 包含两个矩形的 View 1 外观

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 **+**。

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，搜索 Volume up 事件并双击它。

步骤 4.3

在“操作”属性旁边，点击 **+**。

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，搜索 Volume down 事件并双击它。

步骤 5.3

在“操作”属性旁边，点击 **+**。

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，搜索 Close volume control 事件并双击它。

步骤 6.3

在“操作”属性旁边，点击 **+**。

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。

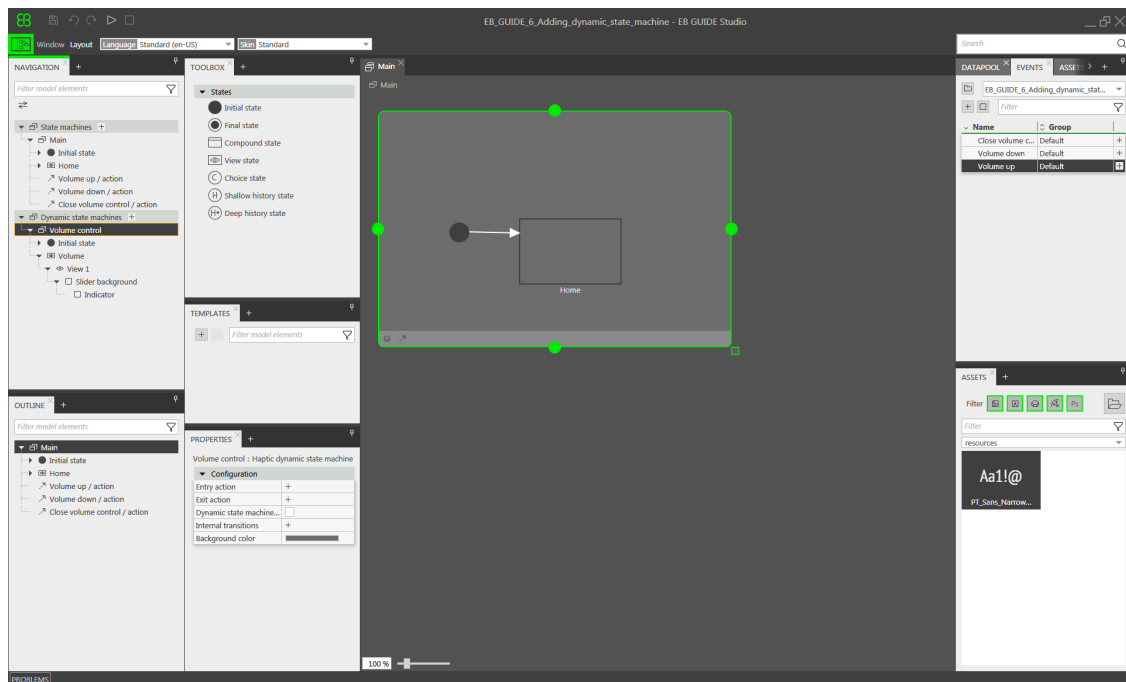


图 14.2. 包含所有模型元素的 EB GUIDE 模型



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，点击 **+**，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。

14.2. 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

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。

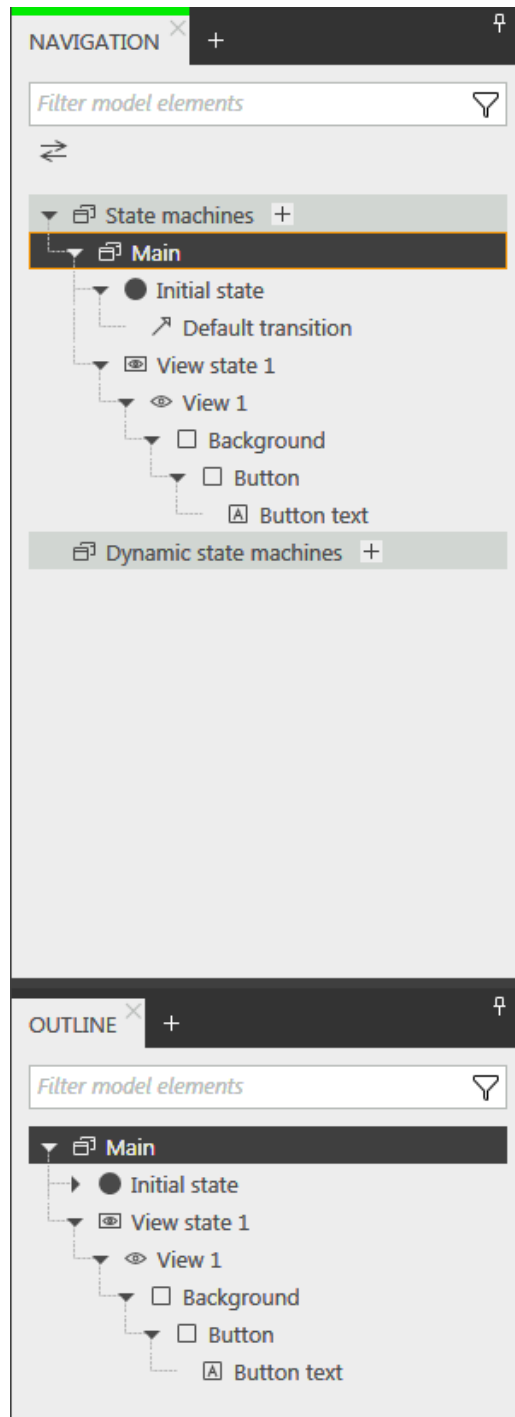


图 14.3. 控件层次结构



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

在 touchPressed 属性旁边，选择“值”列并点击 `{}`。

将打开 EB GUIDE 脚本 编辑器。

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，选择 center(1)。


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 14.4. 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。

14.3. 教程：对路径动作进行建模

注意



默认窗口布局

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

在 onPath 属性旁边，选择“值”列并点击 {} 。

将打开 EB GUIDE 脚本 编辑器。

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

在 onPathStart 属性旁边，选择“值”列并点击 {} 。

将打开 EB GUIDE 脚本 编辑器。

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

在 onPathNotRecognized 属性旁边，选择“值”列并点击 {} 。

将打开 EB GUIDE 脚本 编辑器。

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。

14.4. 教程：使用动态内容创建 列表

注意**默认窗口布局**

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

选择“值”列，并点击  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。

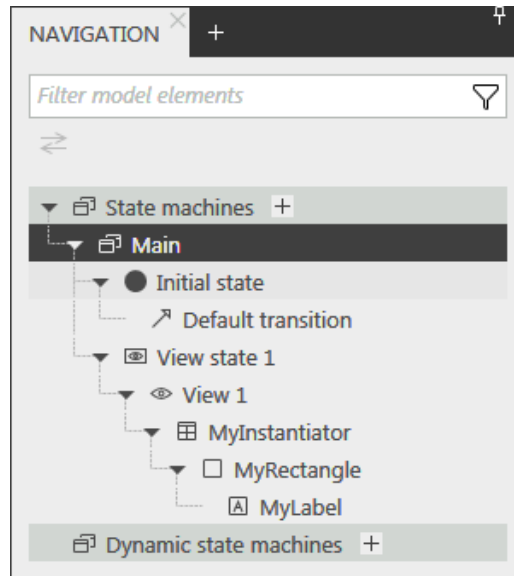


图 14.5. 包含实例化器的控件层次结构



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

在属性旁边，选择“值”列并点击 `{}`。

将打开 EB GUIDE 脚本 编辑器。

步骤 5.4

在“触发器”下，输入 dp:MyStringList。

步骤 5.5

输入以下“触发器”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。

步骤 6.2

点击“接受”。

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 6.3

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 vertical (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，请选择 MyLabel 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，选择 center(1)。

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

在 setText 属性旁边，选择“值”列并点击 **{}**。

将打开 EB GUIDE 脚本 编辑器。

步骤 5.5

在“触发器”下，输入 v:this.lineIndex 和 dp:MyStringList。

步骤 5.6

输入以下“触发器”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。

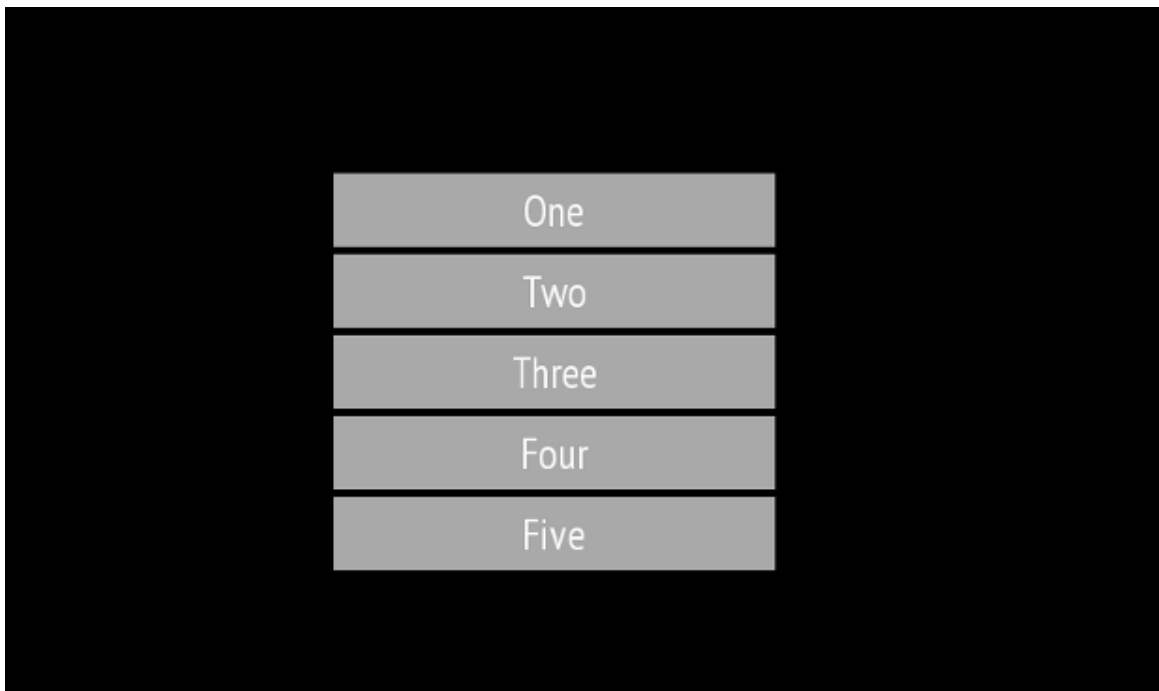


图 14.6. 使用实例化器创建的列表

14.5. 教程：使椭圆移动经过屏幕

注意



默认窗口布局

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。

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

在 startAnimation 属性旁边，选择“值”列并点击 {}。

将打开 EB GUIDE 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。
- 内容区域将显示视图 View 1 视图。

步骤 1

转至“动画编辑器”。在“动画属性”旁边，点击 + 并选择 View 1。


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。

14.6. 教程：向数据池项添加语言依赖项文本

注意



默认窗口布局

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”。

通过 **EB GUIDE** 可采用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。
- 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。
- 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

在 `touchPressed` 属性旁边，选择“值”列并点击 `{}`。

将打开 EB GUIDE 脚本 编辑器。

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:setLanguage(l:German, true)
        dp:SelectedLanguage = 1
    }
    else if (dp:SelectedLanguage == 1) // German selected
    {
        f:setLanguage(l:French, true)
        dp:SelectedLanguage = 2
    }
    else if (dp:SelectedLanguage == 2) // French selected
    {
        f:setLanguage(l:Standard, true)
        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。

14.7. 教程：使用 3D 图形

注意



默认窗口布局

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 2.0 或更高版本的渲染器才能显示 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 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”拖入视图中。

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 图形文件的文件夹。

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>` 文件夹包含 .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。

14.8. 教程：正确渲染伽马

注意**默认窗口布局**

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”。

在 EB GUIDE Studio 中可对以下内容执行伽马校正：

- ▶ 显示器
- ▶ 图像
- ▶ 纹理

下列说明展示了如何配置伽马校正。为获得最佳效果，请按提供的顺序执行步骤。

大约持续时间：**15** 分钟。

**为显示器配置伽马编码**

下列步骤指导您完成设置 EB GUIDE Studio 以将经过伽马编码的值输出到显示屏上的过程。

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。

步骤 1

点击 。

将打开项目中心。

步骤 2

在导航区，单击“配置” > “配置文件”。

步骤 3

对于每个可识别 sRGB 的配置文件，请配置如下：

步骤 3.1

在内容区域中，点击“场景”标签。

步骤 3.2

从 colorMode 下拉列表框中选择如下：

- ▶ 如果渲染管道使用 GPU 的硬件 sRGB 支持，请选择 32-bit sRGB (4)
- ▶ 如果渲染硬件不支持 sRGB，请选择 32-bit sRGB (Emulated) (5)。

要在编辑模式应用更改，请重启 EB GUIDE Studio。

注意



渲染硬件

OpenGL 3 渲染器始终具有硬件 sRGB 支持。如果硬件通过适当的 OpenGL ES 扩展支持，则使用 OpenGL ES 2.0 API 的 OpenGL 渲染器仅使用硬件 sRGB 支持。这是自动检测的。如果您的 OpenGL ES 2.0 硬件不支持 sRGB，则渲染器会自动回退到 32-bit sRGB (Emulated) (5)，它使用片段着色器进行转换。请注意，硬件 sRGB 模式不适用于所有系统，即使这些系统支持 OpenGL ES 3.0。在这种情况下，切换到 32-bit sRGB (Emulated) (5)。



为图像配置伽马编码

先决条件:

- 您已完成前述说明。
- 资源文件夹中有可用的图像文件。

步骤 1

转至项目编辑器并双击视图。将图像从“工具箱”拖入视图中。

步骤 2

在“属性”元素中，从 image 下拉列表框选择一个图像文件。

图像可能太亮。

这是因为场景属性中配置的 sRGB 颜色模式现在将伽马校正应用到已经经过伽马校正的图像上。

步骤 3

要将图像配置为经过伽马编码，请在“属性”元素中选择 sRGB。

图像现在在混合操作中将正确地显示并处理。



为纹理配置伽马编码

先决条件:

- 您已完成前述说明。
- 具有至少一个纹理化 3D 对象的 3D 文件（例如 .fbx 文件）可用且具有弥散纹理。

步骤 1

将场景图从“工具箱”拖入视图中。

步骤 2

在“属性”元素中，单击“导入文件”并选择一个 3D 文件。

步骤 3

在“导航”元素中找到导入的场景图。在场景图结构内选择使用弥散纹理的材料控件。

步骤 4

在“属性”元素中，点击“控件功能属性”中的“添加/删除”。

步骤 5

从“3D”类别中，选择“弥散纹理”。

步骤 6

选择 diffuseSRGB 属性。

纹理被视为经伽马编码的图像，并且在用于照明计算之前已经过线性化。

14.9. 教程：使用视图切换动画

注意**默认窗口布局**

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”。

视图切换动画 (VTA) 是在从一个视图移动到另一个视图时完成的动画。下列说明将指导您完成创建这些动画的过程。您将创建一个模型，其中包含更改视图时播放的视图和动画。将创建以下元素：

- ▶ 两种视图状态
- ▶ 导航元素，例如按钮和标签
- ▶ 触发视图状态更改的事件
- ▶ 转换到其他状态时播放的动画

大约持续时间：30 分钟。

**创建第一个视图状态****创建第一个视图和按钮**

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“导航”元素中，将视图状态重命名为 FirstState 并将视图重命名为 FirstView。



步骤 2

打开 FirstView。

步骤 3

从“工具箱”元素中，将矩形拖动到 FirstView 并将其重命名为 RectNextView。

此矩形用于触发转换的按钮。

步骤 4

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 5

在“可用控件功能”下，展开“效果”类别并选择“边框”。

步骤 6

点击“接受”。

“属性”元素显示相关的控件功能属性。

步骤 7

在“属性”元素中，输入以下内容：

- ▶ 在 width 文本框中，输入 220。
- ▶ 在 height 文本框中，输入 70。
- ▶ 在 x 文本框中，输入 290。
- ▶ 在 y 文本框中，输入 150。
- ▶ 将 fillColor 设置为空。
- ▶ 在 borderThickness 文本框中，输入 2。
- ▶ 将 borderColor 设置为白色。

步骤 8

从“工具箱”元素中，将标签拖动到“导航”元素并将其添加为 FirstView 的子代控件。

步骤 9

将标签重命名为 LabelNextView。

步骤 10

在“属性”元素中，输入以下内容：

- ▶ 在 text 文本框中，输入 Go to the next view。
- ▶ 在 font 文本框中，输入 25。
- ▶ 将 horizontalAlign 设置为居中。

步骤 11

将标签的尺寸链接到矩形的尺寸。链接以下属性：

- ▶ 将 LabelNextView 的 width 属性链接到 RectNextView 的 width。

- ▶ 将 LabelNextView 的 height 属性链接到 RectNextView 的 height。
- ▶ 将 LabelNextView 的 x 属性链接到 RectNextView 的 x。
- ▶ 将 LabelNextView 的 y 属性链接到 RectNextView 的 y。



图 14.7. 带按钮的 FirstView



创建第二个视图状态

第二个视图包含按钮。通过复制和重命名已创建的元素来创建此视图。

先决条件：

- 您已完成前述说明。

步骤 1

选择“主要”标签”

步骤 2

复制并粘贴 FirstState 状态。

步骤 3

在“导航”元素中，找到您创建的新状态并重命名以下控件：

- ▶ 将状态重命名为 SecondState。
- ▶ 将视图重命名为 SecondView。
- ▶ 将 RectNextView 重命名为 RectGoBack。
- ▶ 将 LabelNextView 重命名为 LabelGoBack。

步骤 4

双击 LabelGoBack，并在 text 文本框中输入 Go back。

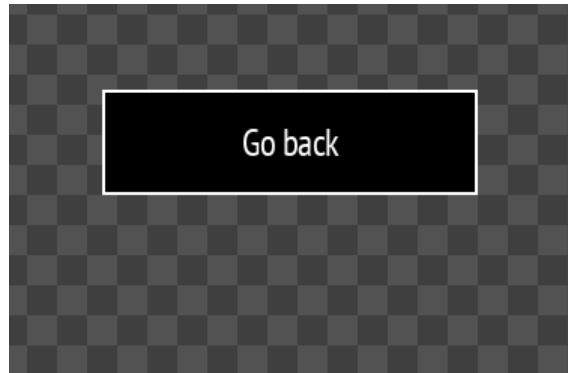


图 14.8. 带按钮的 SecondView



创建转换和事件

先决条件:

- 您已完成前述说明。

步骤 1

双击“主要”状态机

步骤 2

创建从“主要”状态机的边缘到两个视图状态的转换。

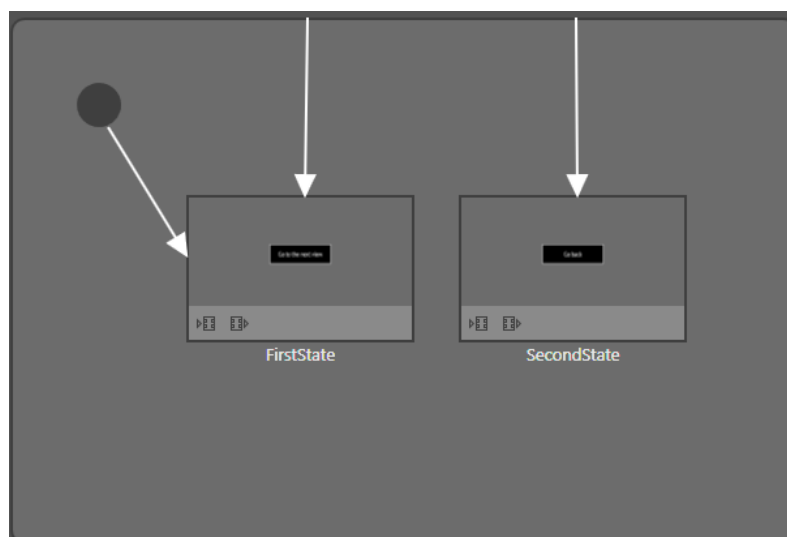


图 14.9. 具有转换的 Main 状态机

步骤 3

选择从“主要”状态机到 FirstState 的转换。

步骤 4

在“属性”元素中，在“触发器”组合框中输入 `goToFirstState` 并点击“添加事件”。

创建了新事件。

步骤 5

选择转换为 `SecondState`。

步骤 6

在“属性”元素中，在“触发器”组合框中输入 `goToSecondState` 并点击“添加事件”。

创建了新事件。



连接按钮和事件

现在，您可以定义以下行为：点击按钮时，将触发到另一个状态的转换。为此，请使用 **EB GUIDE** 脚本。

先决条件：

- 您已完成前述说明。

步骤 1

在“导航”元素中，双击 `LabelNextView`。

步骤 1.1

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 1.2

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”。

步骤 1.3

点击“接受”。

相关的控件功能属性，将添加至“属性”元素。

步骤 1.4

在 `touchShortReleased` 属性旁边，选择“值”列并点击 `{}`。

将打开 **EB GUIDE** 脚本 编辑器。

步骤 1.5

输入以下点击按钮时触发 `goToSecondState` 事件的 **EB GUIDE** 脚本：

```
function(v:touchId::int, v:x::int, v:y::int, v:fingerId::int)
{
    fire ev:goToSecondState()
    false
}
```

步骤 1.6

点击“接受”。

步骤 2

在“导航”元素中，双击 LabelGoBack。

步骤 2.1

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 2.2

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”。

步骤 2.3

点击“接受”。

相关的控件功能属性，将添加至“属性”元素。

步骤 2.4

在 "touchShortReleased" 属性旁边，选择“值”列并点击 {} 。

将打开 EB GUIDE 脚本 编辑器。

步骤 2.5

输入以下点击按钮时触发 goToFirstState 事件的 EB GUIDE 脚本：

```
function(v:touchId::int, v:x::int, v:y::int, v:fingerId::int)
{
    fire ev:goToFirstState()
    false
}
```

步骤 2.6

点击“接受”。



创建 FirstView 进入动画

启用 "VTA" 元素并创建一个从右侧移入按钮的动画。要创建动画，您需要定义动画的属性、动画的持续时间、启动位置以及结束位置。在本教程中，仅使用 x 属性。

先决条件：

- 您已完成前述说明。

步骤 1

"VTA" 元素在默认布局中并不可见。需要启用它。

在命令区点击“布局” > "VTA"。

显示 "VTA" 元素。

步骤 2

在“导航”元素中，双击 FirstState。

步骤 3

在“VTA”元素中点击 **+** 并选择“进入动画”。

显示“进入动画”表。

“动画编辑器”将显示在内容区域下方。

步骤 4

点击 **+** 并选择“退出动画”。

显示“退出动画”表。

步骤 5

在“动画编辑器”的下拉列表框中选择 Entry animation 1。

步骤 6

点击 **+** 并选择“目标：”FirstView。这将显示“动画属性”对话框。

步骤 7

依次点击 RectNextView、x、“快速启动曲线”和“接受”。

新的动画添加到“动画属性”列表。

步骤 8

在“属性”元素中输入以下内容：

- ▶ 在 start 文本框中，输入 900。
- ▶ 在 end 文本框中，输入 290。

当您开始模拟运行时，您可以看到按钮移入。



创建 FirstView 退出动画

创建一个将按钮从右侧移出的动画。

先决条件：

- 您已完成前述说明。

步骤 1

在“动画编辑器”的下拉列表框中选择 Exit animation 1。

步骤 2

点击 **+** 并选择“源：”FirstView。这将显示“动画属性”对话框。

步骤 3

依次点击 RectNextView、x、“快速启动曲线”和“接受”。

新的动画添加到“动画属性”列表。

步骤 4

在“属性”元素中输入以下内容：

- ▶ 在 duration 文本框中，输入 500。
- ▶ 在 start 文本框中，输入 290。
- ▶ 在 end 文本框中，输入 800。



创建 SecondView 进入动画

创建一个将按钮从右侧移入的动画。

先决条件：

- 您已完成前述说明。

步骤 1

在“导航”元素中，双击 SecondView。

SecondView 将显示在内容区域中。

步骤 2

在“VTA”标签中，点击 **+**

步骤 3

添加“进入动画”和“退出动画”。

“动画编辑器”将显示在内容区域下方。

步骤 4

在“属性”元素的下拉列表框中选择 Entry animation 2。

步骤 5

点击 **+** 并选择“目标”：SecondView。这将显示“动画属性”对话框。

步骤 6

依次点击 RectGoBack、x、“快速启动曲线”和“接受”。

新的动画添加到“动画属性”列表。

步骤 7

在“属性”元素中输入以下内容：

- ▶ 在 start 文本框中，输入 900。
- ▶ 在 end 文本框中，输入 290。



创建 SecondView 退出动画

创建一个将按钮从右侧移出的动画。

先决条件：

- 您已完成前述说明。

步骤 1

在“属性”元素的下拉列表框中选择 Exit animation 2。

步骤 2

点击 **+** 并选择“源”： “SecondView。这将显示“动画属性”对话框。

步骤 3

依次点击 RectGoBack、x、“快速启动曲线”和“接受”。

新的动画添加到“动画属性”列表。

步骤 4

在“属性”元素中输入以下内容：

- ▶ 在 duration 文本框中，输入 500。
- ▶ 在 start 文本框中，输入 290。
- ▶ 在 end 文本框中，输入 800。




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 。

点击 Go to the next view。视图更改并播放动画。

点击 Go back。视图将更改回第一个视图并播放动画。

14.10. 教程：使用脚本曲线制作动画

注意



默认窗口布局

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”。

如果要为动画定义自己的曲线，请使用脚本曲线。当其他动画曲线不合适或您只想定义自定义曲线时，可能需要定义自己的曲线。在本教程中，将创建一个包含两个动画的两个脚本曲线的简单模型。将创建以下元素：

- ▶ 视图状态
- ▶ 两个矩形控件
- ▶ 两个带有脚本曲线的动画控件，用于为矩形控件的位置设置动画

这会产生具有两个矩形的模型。一个矩形向下移动。另一方移向一边。

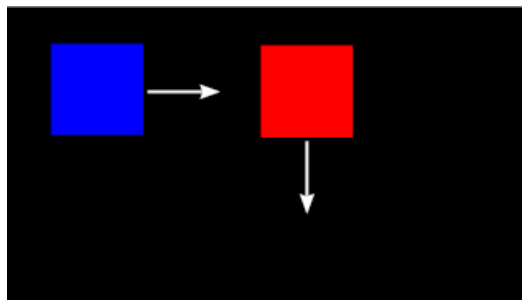


图 14.10. 矩形及其运动方向

大约持续时间：15 分钟。



创建第一个脚本曲线

先决条件：

- “主要”状态机包含初始状态和称为 FirstState 的视图状态，以及称为 FirstView 的视图。
- 初始状态具有到 FirstState 的转化。
- 内容区域将显示视图 FirstView 视图。

步骤 1

从“工具箱”元素中，将矩形拖动到视图中并将其重命名为 BlueRectangle

步骤 2

在“属性”元素中，将 fillColor 属性设置为蓝色。

步骤 3

从“工具箱”元素中，将动画拖动到视图中并将其重命名为 MoveAnimation。



步骤 4

在“数据池”元素中，添加一个类型为 Float 的数据池项并将其重命名为 xFloat。

步骤 5

在“导航”元素中，选择 BlueRectangle。

步骤 6

在“属性”元素中，转至“用户定义的属性”类别，并点击 **+**。

打开一个菜单。

步骤 7

在菜单中，选择“条件脚本”。

Conditional script 1 已添加到“用户定义的属性”。

步骤 8

将 Conditional script 1 重命名为 StartBlueAnimation。

步骤 9

在 StartBlueAnimation 旁边，点击 `{}`。

将打开 EB GUIDE 脚本 编辑器。

步骤 10

输入以下脚本：

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

步骤 11

在“导航”元素中，选择 FirstView。

步骤 12

在“动画编辑器”中，在“动画属性”旁边点击 **+** 并选择 FirstView。

这将打开“动画属性”对话框。

步骤 13

在 BlueRectangle 下，依次选择 x 属性和 Script curve

步骤 14

点击“接受”。

Script curve 1 将添加到“动画编辑器”中。

步骤 15

将 Script curve 1 重命名为 BlueCurve。

步骤 16

在“属性”元素中，在 curve 属性旁边点击 `{}`。

将打开 EB GUIDE 脚本 编辑器。

步骤 17

输入以下脚本：

```
function(v:diff::int, v:t_anim::int)
{
  dp:xFloat+=0.2
  f:floor(dp:xFloat*dp:xFloat)
}
```



创建第二个脚本曲线

先决条件：

- 您已完成先前的说明。

步骤 1

从“工具箱”元素中，将矩形拖动到 “FirstView” 并将其重命名为 RedRectangle。

步骤 2

在“属性”元素中，将 fillColor 属性设置为红色。

步骤 3

在“数据池”元素中，添加一个类型为 Integer 的数据池项并将其重命名为 1_diff。

步骤 4

添加另一个类型为 Integer 的数据池项并将其重命名为 2t_anim。

步骤 5

选中 RedRectangle。

步骤 6

在“属性”元素中，转至“用户定义的属性”类别，点击 **+** 并添加条件脚本类型的属性。

Conditional script 2 已添加。

步骤 7

将 Conditional script 2 重命名为 StartRedAnimation。

步骤 8

在 StartRedAnimation 旁边，点击 **{}**。

将打开 EB GUIDE 脚本 编辑器。

步骤 9

输入以下脚本：

```
function(v:arg0::bool)
{
  f:animation_play(v:this->^->“MoveAnimation”)
}
```

步骤 10

在“导航”元素中，选择 FirstView。

步骤 11

在“动画编辑器”中，在“动画属性”旁边点击 **+** 并选择 FirstView。

这将打开“动画属性”对话框。

步骤 12

在 RedRectangle 下，依次选择 y 属性和 Script curve。

步骤 13

点击“接受”。

Script curve 2 将添加到“动画编辑器”中。

步骤 14

将 Script curve 2 重命名为 RedCurve。

步骤 15

在“属性”元素中，在 curve 属性旁边点击 **{}**。

将打开 EB GUIDE 脚本编辑器。

步骤 16

输入以下脚本：

```
function(v:diff::int, v:t_anim::int)
{
  dp:"1_diff"=v:diff
  dp:"2t_anim"=v:t_anim
  v:t_anim/2::int
}
```



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 **▶**。

动画在模拟开始时播放。

14.11. 教程：创建水平进度条

注意



默认窗口布局

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对进度条进行建模的过程，如下所示。

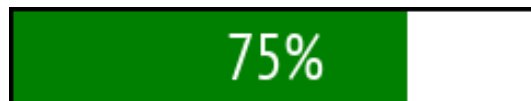


图 14.11. 进度条

您也可以在控件模板库中查看进度条模板。参见<https://www.elektrobit.com/ebguide/examples/>。

大约持续时间：10 分钟



添加控件

下列说明将指导您完成为进度条添加控件的过程。

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

在“模板”元素中，点击 **+** 并点击 Container。

创建了一个包含容器的模板。

步骤 2

将模板重命名为 T_ProgressBar。

步骤 3

将容器重命名为 ProgressBar_Container。

步骤 4

将矩形拖动到容器中，并将其重命名为 Background_Rectangle。

步骤 5

将另一个矩形拖动到容器中，并将其重命名为 Progress_Rectangle。

此矩形显示操作进度。

步骤 6

将标签拖动到容器中，并将其重命名为 Percentage_Text。



输入进度条属性

下列说明将指导您完成配置属性和向控件添加脚本的过程。

先决条件：



- 您已完成前述说明。

步骤 1


在“模板”元素中，选择 ProgressBar_Container。

步骤 2

将属性 width, height, x, y 添加到模板接口。

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

在“属性”元素中，转至“用户定义的属性”类别，点击  并选择 Integer。

向容器添加用户定义的 Integer 类型的属性。

步骤 4

将该属性重命名为 progress。

步骤 5


将 progress 添加至模板接口

步骤 6

在“模板”元素中，选择 Background_Rectangle。

步骤 7

将 width 链接到 ProgressBar_Container。

要将属性链接到其他属性，请在“属性”元素中点击属性旁的  按钮。在菜单中，点击“添加链接到控件属性”。

对话框将打开。

步骤 8

在对话框中，选择 ProgressBar_Container 的 width 属性并点击“接受”。

步骤 9

将 Background_Rectangle 的 height 属性链接到 ProgressBar_Container 的 height 属性。

步骤 10

在“模板”元素中，选择 Progress_Rectangle。

**步骤 11**

将 height 属性链接至 ProgressBar_Container 的 height 属性。

步骤 12

将 fillColor 设置为绿色。

步骤 13

在 width 属性旁边，点击  并选择 Convert to script。

width 属性将宽度定义为 ProgressBar_Container 宽度的百分比。

步骤 14

点击 {}。

将打开 EB GUIDE 脚本 编辑器。

步骤 15

在“读取”部分输入下列 EB GUIDE 脚本：

```
function()
{
  v:this->^.width * v:this->^.progress / 100
}
```

此脚本将 progress 属性的值除以 100。

步骤 16

点击“添加可用的触发器到列表”。

添加了 width 和 progress 的两个触发器。

步骤 17

在“模板”元素中，选择 Percentage_Text。

步骤 18

将 width 和 height 属性链接到 ProgressBar_Container 的宽度和高度。

步骤 19

将 horizontalAlign 设置为 center (1)。

步骤 20

将 text 属性转换为脚本。

文本将显示容器宽度的百分比。

步骤 21

点击 {}。

将打开 EB GUIDE 脚本 编辑器。

步骤 22

在“读取”部分中，输入下列脚本：

```
function()
{
```

```
f:int2string(v:this->.progress) + "%"  
}
```

此脚本将百分比值转换为字符串，并在百分比数字后添加 % 字符。

步骤 23

点击“添加可用的触发器到列表”。

添加了 progress 的触发器。

步骤 24

将模板中所有控件的 x 和 y 属性设置为 0。

步骤 25

在“导航”元素中，双击视图。

步骤 26

将 T_ProgressBar 文件从“工具箱”元素拖入内容区域中。

模板将添加至视图。现在，您可以向其中添加动画以显示操作的动态进度。



动画化进度

下列说明将指导您完成进度条动画化的过程，以便更好地查看更改百分比值时发生的情况。

先决条件：

- 您已完成前述说明。
- 内容区域将显示视图。

步骤 1

将动画拖动到视图中。

步骤 2

将动画重命名为 Loading_Animation。

步骤 3

在“属性”元素中，转至“用户定义的属性”类别，点击 **+** 并选择 Conditional script。

步骤 4

将条件脚本重命名为 animateProgress。

步骤 5

在条件脚本属性旁边，点击 **{}**。

将打开 EB GUIDE 脚本 编辑器。

步骤 6

在“触发器”部分中，输入下列脚本：

```
function(v:arg0::bool)
```

```
{  
  f:animation_play(v:this)  
  false  
}
```

步骤 7

在“导航”元素中，双击 Loading_Animation 以打开“动画编辑器”。

步骤 8

在“动画编辑器”中，在“动画属性”旁边点击 **+** 并选择 View 1。

这将打开“动画属性”对话框。

步骤 9

在 T_ProgressBar 1 下，依次选择 progress 属性和 Linear interpolation curve。点击“接受”。

步骤 10

在“属性”面板中，将 end 属性设置为 100。

当进度指示器达到 100% 时，进度动画将停止。




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 。

15. 参考

以下章节为您提供列表和表（例如，参数、属性和识别符等）。

有关 EB GUIDE GTF 的特定参数、属性和识别符，请参见 EB GUIDE GTF 用户指南。

15.1. 命令行选项

15.1.1. 命令行选项 Studio.Console.exe

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不会提示错误消息。

命令行的一般语法如下：

```
Studio.Console.exe <option> "project_name.ebguide"
```

表 15.1. 命令行选项 Studio.Console.exe

选项	说明
-c <logfile dir>	验证一个 EB GUIDE 模型，并将日志文件写入指定文件夹 logfile dir
-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>	使用在导出期间指定为 profile 的配置文件。



例 15.1. 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](#)

15.1.2. 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不会提示错误消息。

命令行的一般语法如下：

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

表 15.2. 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（对于中文简体）。
-o	打开配置文件 monitor.cfg
-s	运行已定义脚本中的所有方法



例 15.2. 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[章11](#)。

15.2. 数据池项

表 15.3. 数据池项的属性

属性名称	说明
Value	数据池项的初始值

15.3. 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。

15.3.1. 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15.3.12](#)。

15.3.2. 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15.3.12](#)。

15.3.3. 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：



- ▶ 在“触发器”部分，可以选择用于触发运行“触发器”脚本的事件、数据池项和控件属性。
- ▶ 在“触发器”部分，可以添加在初始化、事件触发或者数据池项或控件属性的值更新后调用的 EB GUIDE 脚本。

“触发器” EB GUIDE 脚本的参数指示了运行脚本的原因。

arg0 指的是 EB GUIDE 脚本是在初始化期间还是由触发器执行的。参考如下：

- ▶ 如果 EB GUIDE 脚本是在初始化期间执行的，arg0 为 true。
- ▶ 如果 EB GUIDE 脚本是由触发器执行的，arg0 为 false。

“触发器” EB GUIDE 脚本的返回值用于控制属性的更改通知。

“触发器” EB GUIDE 脚本的返回值规定 EB GUIDE 脚本是否必须生成通知。参考如下：

- ▶ 如果返回值是 true，则生成通知。
- ▶ 如果返回值是 false，则不生成通知。

为了能够执行“触发器”脚本，要在以下情况下满足条件：

- ▶ 在初始化时，例如在 EB GUIDE 模型启动期间的数据池项，或在视图创建期间的控件属性。
- ▶ 在处理触发器脚本中的事件时。每次匹配事件都会执行一次 EB GUIDE 脚本。
- ▶ 在处理触发器脚本中的一个或多个项目的数据池通知时。可以一次处理多个通知。
- ▶ 在处理触发器脚本中的一个或多个控件属性的通知时。可以一次处理多个通知。

15.3.4. 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 (>=)
- ▶ 小于 (<)
- ▶ 小于或等于 (<=)
- ▶ 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15.3.12](#)。

15.3.5. 字体

要向 EB GUIDE 项目中添加字体，请将字体文件复制到：`$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15.3.12](#)。

15.3.6. Function () : bool

通过 Function () : bool，您可以创建自己的函数。

此数据类型的可用操作是对所有属性的读取/运行操作。

15.3.7. ibl

ibl 是存储由 IBLGenerator 生成的照明信息的数据格式。

要向 EB GUIDE 项目中添加 ibl，请将 .ebibl 文件复制到：`$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 ibl 属性。有关列表的详细信息，请参见[节15.3.12](#)。

15.3.8. 图像

要向 EB GUIDE 项目中添加图像，请将图像文件复制到：`$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15.3.12](#)。

15.3.9. 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15.3.12](#)。

15.3.10. 网格

网格定义 3D 对象的形状。

要向 EB GUIDE 项目中添加网格，请将 .ebmesh 文件复制到：`$GUIDE_PROJECT_PATH/<project name>/resources`

可用运算如下所示：

- ▶ 赋值（可写属性）(=)



可以在列表中存储网格属性。有关列表的详细信息，请参见[节15.3.12](#)。

15.3.11. 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15.3.12](#)。

15.3.12. 列表

EB GUIDE 支持具有相同数据类型的值列表。

以下列表类型可用：

- ▶ 布尔值列表
- ▶ 颜色列表
- ▶ 浮点值列表
- ▶ 字体列表
- ▶ `lbl` 列表
- ▶ 图像列表
- ▶ 整数列表
- ▶ 网格列表
- ▶ 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：([])

15.4. EB GUIDE 脚本

15.4.1. 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它们用作识别符，则必须用引号将它们括起来。

关键字	说明
cancel_fire	取消使用 fire_delayed 触发的事件。
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。用于 f:setLanguage(l:English, true)。
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
fire_delayed	在指定时间后触发事件。以毫秒为单位指定时间。
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

关键字	说明
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
s:	随后是外观。用于 f:setSkin(mySkin, true)。
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

15.4.2. 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 15.4. EB GUIDE 脚本 运算符优先顺序

运算符	结合性
(()), ({}), ([])	无
(->)	左
(.)	无
(::)	左
length	无
(&)	右
(!), (-) 一元减号	右
(*), (/), (%)	左
(!), (&)	左
(<), (>), (<=), (>=)	左
(!=), (==), (=Aa=)	左
(&&)	左



运算符	结合性
()	左
(=), (+=), (--), (=>)	右
(,)	右
(;)	左

15.4.3. EB GUIDE 脚本 标准库

以下章节说明了所有 EB GUIDE 脚本 函数。

15.4.3.1. EB GUIDE 脚本 函数 A - B

15.4.3.1.1. abs

此函数用于返回整数 x 的绝对值。

表 15.5. 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

15.4.3.1.2. absf

此函数用于返回浮点数 x 的绝对值。

表 15.6. 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

15.4.3.1.3. acosf

此函数用于返回反余弦 x 的主值。

表 15.7. 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

15.4.3.1.4. 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 15.8. 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。

15.4.3.1.5. 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 15.9. 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。

15.4.3.1.6. 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 15.10. 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。



15.4.3.1.7. 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 15.11. 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。

15.4.3.1.8. 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 15.12. 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。

15.4.3.1.9. animation_pause

此函数用于暂停动画。

表 15.13. 参数 animation_pause

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。

15.4.3.1.10. animation_play

此函数用于启动或继续播放动画。

表 15.14. 参数 animation_play

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，则表示动画尚未播放。

15.4.3.1.11. animation_reverse

此函数用于回放动画。

表 15.15. 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。

15.4.3.1.12. animation_running

此函数用于检查动画当前是否正在播放。

表 15.16. 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。

15.4.3.1.13. 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 15.17. 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。

15.4.3.1.14. asinf

此函数用于计算反正弦 x 的主值。



表 15.18. 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

15.4.3.1.15. 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 15.19. 参数 atan2f

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

15.4.3.1.16. 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 15.20. 参数 atan2i

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

15.4.3.1.17. atanf

此函数用于计算反正切 x 的主值。

表 15.21. 参数 atanf

参数	类型	说明
x	浮点值	返回其反正切的数字



参数	类型	说明
<return>	浮点值	返回值

15.4.3.1.18. bool2string

此函数用于将布尔变量转换为字符串 true 或 false。

表 15.22. 参数 bool2string

参数	类型	说明
x	布尔值	要转换为字符串的值
<return>	字符串	true 用于当 x 为 true 时，否则返回 false

15.4.3.2. EB GUIDE 脚本 函数 C - H

15.4.3.2.1. ceil

此函数用于返回不小于参数的最小整数值。

表 15.23. 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

15.4.3.2.2. changeDynamicStateMachinePriority

此函数用于更改动态状态机的优先级。

表 15.24. 参数 changeDynamicStateMachinePriority

参数	类型	说明
stack	弹出堆栈 ID	动态状态机列表
sm	状态机 ID	动态状态机
priority	整数	动态状态机在列表中的优先级。注意，数字越大意味着优先级越高。



15.4.3.2.3. character2unicode

此函数用于返回字符串中第一个字符的 **Unicode** 值。

表 15.25. 参数 character2unicode

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

15.4.3.2.4. clampf

此函数用于将浮点值钳夹到定义的范围 $[x_{min}, x_{max}]$ ，即函数计算 $\max(x_{min}, \min(x_{max}, x))$ 。

表 15.26. 参数 clampf

参数	类型	说明
x	浮点值	要钳夹的值
xmin	浮点值	最小范围
xmax	浮点值	最大范围
<return>	浮点值	x 值已钳夹到 $[x_{min}, x_{max}]$ 范围

15.4.3.2.5. clampi

此函数用于将整数值钳夹到定义的范围 $[x_{min}, x_{max}]$ ，即函数计算 $\max(x_{min}, \min(x_{max}, x))$ 。

表 15.27. 参数 clampi

参数	类型	说明
x	int	要钳夹的值
xmin	int	最小范围
xmax	int	最大范围
<return>	int	x 值已钳夹到 $[x_{min}, x_{max}]$ 范围

15.4.3.2.6. 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 15.28. 参数 clearAllDynamicStateMachines

参数	类型	说明
state		包含动态状态机列表的状态

15.4.3.2.7. color2string

此函数用于将颜色转换为 8 个十六进制值。

表 15.29. 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。

15.4.3.2.8. cosf

此函数用于返回 x 的余弦，其中 x 以半径指定。

表 15.30. 参数 cosf

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

15.4.3.2.9. deg2rad

此函数用于将角度从度转换为半径。

表 15.31. 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



15.4.3.2.10. expf

此函数用于返回值 e （自然对数的底）的 x 次幂。

表 15.32. 参数 expf

参数	类型	说明
x	浮点值	指数
<return>	浮点值	返回值

15.4.3.2.11. float2string

此函数用于将简单浮点值转换为字符串。

表 15.33. 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

15.4.3.2.12. floor

此函数用于返回不大于参数值的最大整数值。

表 15.34. 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

15.4.3.2.13. fmod

此函数用于计算浮点值除法 x/y 的余数。

表 15.35. 参数 fmod

参数	类型	说明
x	浮点值	浮点分子
y	浮点值	浮点分母



参数	类型	说明
<return>	浮点值	除法的余数 x/y

15.4.3.2.14. focusMoveTo

此函数用于强制焦点管理器将焦点传递给专用的可聚焦元素。

表 15.36. 参数 focusMoveTo

参数	类型	说明
widget	控件	焦点移动的控件。
<return>	空	

15.4.3.2.15. focusNext

此函数用于强制焦点管理器将焦点传递给下一个可聚焦的元素。

表 15.37. 参数 focusNext

参数	类型	说明
<return>	空	

15.4.3.2.16. 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 15.38. 参数 focusPrevious

参数	类型	说明
<return>	空	

15.4.3.2.17. format_float

此函数用于设置浮点值的格式。

表 15.39. 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

参数	类型	说明
		<ul style="list-style-type: none"> ▶ 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。 ▶ 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。 ▶ 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。
useDotAsDelimiter	布尔值	用于定义分隔符符号。 可能的值： <ul style="list-style-type: none"> ▶ true：使用点作为分隔符。 ▶ false：使用逗号作为分隔符。
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。

15.4.3.2.18. format_int

此函数用于设置整数值的格式。

表 15.40. 参数 format_int

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 <ul style="list-style-type: none"> ▶ 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"> ▶ 精度：可选的十进制数字，用于指定打印的数字的最小数目。 ▶ 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。
value	int	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。

15.4.3.2.19. frac

此函数用于计算浮点值的小数部分。返回值位于区间 [0, 1]。例如，对于参数值 $x=1.5$ 或 $x=-1.-5$ ，函数返回 0.-5。

表 15.41. 参数 frac

参数	类型	说明
x	浮点值	浮点值
<return>	浮点值	浮点值的小数部分。

15.4.3.2.20. getAllLanguages

该函数用来自核心或模型范围的语言 UID 列表填充数据池项。

表 15.42. 参数 getAllLanguages

参数	类型	说明
itemId	dp_id	存储语言 UID 的数据池项 ID。数据池项类型必须是字符串列表。
isCoreScope	布尔值	指定范围。 可能的值： <ul style="list-style-type: none"> ▶ true: 核心范围 ▶ false: 模型范围



参数	类型	说明
<return>	空	

15.4.3.2.21. getAllSkins

该函数用来自核心或模型范围的外观 UID 列表填充数据池项。

表 15.43. 参数 getAllSkins

参数	类型	说明
itemId	dp_id	存储外观 UID 的数据池项 ID。数据池项类型必须是字符串列表。
isCoreScope	布尔值	指定范围。 <ul style="list-style-type: none"> ▶ true: 核心范围 ▶ false: 模型范围
<return>	空	

15.4.3.2.22. getConfigItem

此函数使用配置项值填充数据池项。

表 15.44. 参数 getConfigItem

参数	类型	说明
itemId	dp_id	要存储配置项的数据池 ID
name	字符串	配置项名称
<return>	布尔值	如果用配置项值成功填充数据池项，则为 true。

15.4.3.2.23. getFontAscender

此函数用于返回作为参数传递的字体的升部。

表 15.45. 参数 getFontAscender

参数	类型	说明
x	字体	要评估的字体 请注意，如果您添加了多字体支持，则仅评估默认字体。
<return>	整数	字体的升部

15.4.3.2.24. getFontDescender

此函数用于返回作为参数传递的字体的降部。

表 15.46. 参数 getFontDescender

参数	类型	说明
x	字体	要评估的字体 请注意，如果您添加了多字体支持，则仅评估默认字体。
<return>	整数	字体的降部

15.4.3.2.25. getFontLineGap

此函数用于返回作为参数传递的字体的线间距。

表 15.47. 参数 getFontLineGap

参数	类型	说明
x	字体	要评估的字体 请注意，如果您添加了多字体支持，则仅评估默认字体。
<return>	整数	字体的线间距

15.4.3.2.26. getImageHeight

此函数用于返回作为参数传递的图像的高度（以像素为单位）。

表 15.48. 参数 getImageHeight

参数	类型	说明
x	图像控件	要对其求值的控件
<return>	整数	图像的高度（以像素为单位）

15.4.3.2.27. getImageWidth

此函数用于返回作为参数传递的图像的宽度（以像素为单位）。

表 15.49. 参数 getImageWidth

参数	类型	说明
x	图像控件	要评估的控件



参数	类型	说明
<return>	整数	图像的宽度（以像素为单位）

15.4.3.2.28. getLabelTextHeight

此函数用于返回标签文本的总高度（以像素为单位）。使用以下公式计算总高度：

$$\text{total_height} = \text{line_height} * \text{line_count} + \text{line_spacing} * (\text{line_count} - 1)$$

`line_spacing` 是计算“多行”控件功能的字体 `lineGap` 属性和 `lineOffset` 属性之和。字体 `lineGap` 和 `lineOffset` 属性都可以是负数。

表 15.50. 参数 getLabelTextHeight

参数	类型	说明
widget	标签控件	要评估的控件
<return>	整数	文本的高度（以像素为单位）

15.4.3.2.29. getLabelTextWidth

此函数用于返回标签文本最长行的宽度（以像素为单位）。

表 15.51. 参数 getLabelTextWidth

参数	类型	说明
widget	标签控件	要对其求值的控件
<return>	整数	文本最长行的宽度（以像素为单位）

15.4.3.2.30. getLanguage

该函数从核心或模型范围返回当前语言。

表 15.52. 参数 getLanguage

参数	类型	说明
isCoreScope	布尔值	指定范围。 <ul style="list-style-type: none"> ▶ true: 核心范围 ▶ false: 模型范围
<return>	字符串	语言的 UID。



15.4.3.2.31. getLanguageName

该函数返回指定语言 UID 的名称。

表 15.53. 参数 getLanguageName

参数	类型	说明
languageUid	字符串	请求名称的语言。
<return>	字符串	语言的名称。

15.4.3.2.32. getLanguageTag

该函数返回指定语言 UID 的标记。

表 15.54. 参数 getLanguageTag

参数	类型	说明
languageUid	字符串	请求标记的语言。
<return>	字符串	语言的标记。

15.4.3.2.33. getLineCount

此函数返回标签文本中的行数。

表 15.55. 参数 getLineCount

参数	类型	说明
widget	标签控件	要评估的控件
<return>	整数	文本的行数

15.4.3.2.34. getLineHeight

此函数用于返回用作为参数传递的字体编写的行的高度。

表 15.56. 参数 getLineHeight

参数	类型	说明
x	字体	要评估的字体 请注意，如果您添加了多字体支持，则仅评估默认字体。



参数	类型	说明
<return>	整数	用特定字体编写的行的高度

15.4.3.2.35. getProductString

此函数用于返回一个产品名称为 EB GUIDE GTF 的字符串。

表 15.57. 参数 getProductString

参数	类型	说明
<return>	字符串	产品名称

15.4.3.2.36. getSkin

该函数从核心或模型范围返回当前外观。

表 15.58. 参数 getSkin

参数	类型	说明
isCoreScope	布尔值	指定范围。 <ul style="list-style-type: none"> ▶ true: 核心范围 ▶ false: 模型范围
<return>	字符串	外观的 UID。

15.4.3.2.37. getSkinName

该函数返回指定外观 UID 的名称。

表 15.59. 参数 getSkinName

参数	类型	说明
skinUid	字符串	需要名称的外观。
<return>	字符串	外观的名称。

15.4.3.2.38. getTextHeight

此函数用于返回关于其字体资源的文本高度。高度表示字体升部和降部的总和。

表 15.60. 参数 `getTextHeight`

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	文本的高度 如果字体大小为 0 或负数，函数返回 0。

注意

`getTextHeight`

此函数始终假设文本只有一行来计算高度。

15.4.3.2.39. `getTextLength`

此函数用于返回文本中的字符数。

表 15.61. 参数 `getTextLength`

参数	类型	说明
<code>text</code>	字符串	要对其求值的文本
<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 `getTextLength` 返回 7。**15.4.3.2.40. `getTextWidth`**

此函数用于返回关于其字体资源的文本宽度。

表 15.62. 参数 `getTextWidth`

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

参数	类型	说明
		如果字体大小为 0 或负数，函数返回 0。

注意 此函数始终假设文本只有一行来计算宽度。



15.4.3.2.41. getVersionString

此函数用于返回一个版本号为 EB GUIDE GTF 的字符串。

表 15.63. 参数 getVersionString

参数	类型	说明
<return>	字符串	版本字符串

15.4.3.2.42. 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 15.64. 参数 has_list_window

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

15.4.3.2.43. hsba2color

此函数用于将 HSB/HSV 颜色转换为 EB GUIDE GTF 颜色。

表 15.65. 参数 hsba2color

参数	类型	说明
hue	整数	颜色值以度为单位，介于 0 到 360 之间



参数	类型	说明
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 EB GUIDE GTF 颜色

15.4.3.3. EB GUIDE 脚本 函数 I - R

15.4.3.3.1. int2float

此函数用于返回已转换为浮点值的整数值。

表 15.66. 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

15.4.3.3.2. int2string

此函数用于将简单整数转换为字符串。

表 15.67. 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

15.4.3.3.3. isDynamicStateMachineActive

该功能检查动态状态机列表中是否包含动态状态机。

表 15.68. 参数 isDynamicStateMachineActive

参数	类型	说明
stack	弹出堆栈 ID	动态状态机列表



参数	类型	说明
sm	状态机 ID	动态状态机

15.4.3.3.4. isWidgetOnActiveStatemachine

此函数用于检查控件是否属于活动的状态机。

表 15.69. 参数 isWidgetOnActiveStatemachine

参数	类型	说明
widget	控件	要评估的控件
<return>	布尔值	如果控件属于活动的状态机，则为 true

15.4.3.3.5. language

已弃用：改用 setLanguage。

此函数用于切换核心范围的语言。该操作是同步执行的，但是 EB GUIDE 模型对此更改进行异步响应。

表 15.70. 参数 language

参数	类型	说明
language	字符串	要切换为的语言，例如 <code>f:setLanguage(1:German, true)</code>
<return>	空	

15.4.3.3.6. lerp

此函数使用公式计算两个值 x 和 y 的线性插值 $(1-s) * x + s * y$

表 15.71. 参数 lerp

参数	类型	说明
x	浮点值	第一个值
y	浮点值	第二个值
s	浮点值	x 和 y 值之间的线性插值
<return>	浮点值	返回线性插值 $(1-s) * x + s * y$



15.4.3.3.7. 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 15.72. 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

15.4.3.3.8. 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 15.73. 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

15.4.3.3.9. 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 15.74. 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

15.4.3.3.10. 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 15.75. 参数 localtime_month

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	提取的月份

15.4.3.3.11. localtime_second

此函数用于从系统时间值的本地时间中提取秒数。

表 15.76. 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

15.4.3.3.12. 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 15.77. 参数 localtime_weekday

参数	类型	说明
time	整数	系统时间返回的时间戳
<return>	整数	提取的工作日

15.4.3.3.13. localtime_year

此函数用于从系统时间值的本地时间中提取年份。

表 15.78. 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

15.4.3.3.14. log10f

此函数用于返回以 10 为底的 x 的对数。

表 15.79. 参数 log10f

参数	类型	说明
x	浮点值	参数
<return>	浮点值	返回值

15.4.3.3.15. logf

此函数用于返回 x 的自然对数。

表 15.80. 参数 logf

参数	类型	说明
x	浮点值	参数
<return>	浮点值	返回值

15.4.3.3.16. maxf

此函数用于计算两个浮点值的最大值。

表 15.81. 参数 maxf

参数	类型	说明
x	浮点值	第一个值
y	浮点值	第二个值
<return>	浮点值	x 和 y 的最大值

15.4.3.3.17. maxi

此函数用于计算两个整数值的最大值。

表 15.82. 参数 maxi

参数	类型	说明
x	int	第一个值
y	int	第二个值
<return>	int	x 和 y 的最大值 y



15.4.3.3.18. minf

此函数用于计算两个浮点值的最小值。

表 15.83. 参数 minf

参数	类型	说明
x	浮点值	第一个值
y	浮点值	第二个值
<return>	浮点值	x 和 y 的最小值 y

15.4.3.3.19. mini

此函数用于计算两个整数值的最小值。

表 15.84. 参数 mini

参数	类型	说明
x	int	第一个值
y	int	第二个值
<return>	int	x 和 y 的最小值 y

15.4.3.3.20. nearbyint

此函数用于将值舍入为最接近的整数。

表 15.85. 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

15.4.3.3.21. popDynamicStateMachine

此函数用于从动态状态机列表中删除动态状态机。

表 15.86. 参数 popDynamicStateMachine

参数	类型	说明
stack	弹出堆栈 ID	动态状态机列表



参数	类型	说明
sm	状态机 ID	动态状态机

15.4.3.3.22. powf

此函数用于返回值 x 的 y 次幂。

表 15.87. 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

15.4.3.3.23. pushDynamicStateMachine

此函数用于在动态状态机列表中插入动态状态机。

表 15.88. 参数 pushDynamicStateMachine

参数	类型	说明
stack	弹出堆栈 ID	动态状态机列表
sm	状态机 ID	动态状态机
priority	整数	动态状态机在列表中的优先级。注意，数字越大意味着优先级越高。

15.4.3.3.24. rad2deg

此函数用于将角度从半径转换为度。

表 15.89. 参数 rad2deg

参数	类型	说明
x	浮点值	参数
<return>	浮点值	返回值

15.4.3.3.25. rand

此函数用于获取介于 0 到 $2^{31}-1$ 之间的随机值。

表 15.90. 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

15.4.3.3.26. rgba2color

此函数用于将 RGB 色域转换为 EB GUIDE GTF 颜色。

表 15.91. 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
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 EB GUIDE GTF 颜色，并应用了 alpha 值

15.4.3.3.27. 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 15.92. 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

15.4.3.4. EB GUIDE 脚本 函数 S - W

15.4.3.4.1. saturate

此函数将浮点值钳夹到 [0, 1] 范围，即函数计算 $\max(0, \min(1, x))$ 并充当 `clampf(0, 1, x)` 的简写标记 `clampf(0, 1, x)`



表 15.93. 参数 saturate

参数	类型	说明
x	浮点值	要钳夹的值
<return>	浮点值	x 值已钳夹到 [0, 1] 范围

15.4.3.4.2. seed_rand

此函数用于设置随机数字生成器的种子。

表 15.94. 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	

15.4.3.4.3. setLanguage

此函数用于切换核心或模型范围的语言。该操作是同步执行的，但是模型对此更改进行异步响应。

表 15.95. 参数 setLanguage

参数	类型	说明
languageUid	字符串	要切换为的语言，例如 f:language(l:German)。
isCoreScope	布尔值	指定范围。 <ul style="list-style-type: none"> ▶ true: 核心范围 ▶ false: 模型范围
<return>	空	

15.4.3.4.4. setSkin

此函数用于切换核心或模型范围的语言。该操作是同步执行的，但是模型对此更改进行异步响应。

表 15.96. 参数 setSkin

参数	类型	说明
skinUid	字符串	要切换为的外观，例如 f:setSkin(x, y)。



参数	类型	说明
isCoreScope	布尔值	指定范围。 <ul style="list-style-type: none"> ▶ true: 核心范围 ▶ false: 模型范围
<return>	空	

15.4.3.4.5. shutdown

此函数用于请求框架关闭程序。

15.4.3.4.6. sinf

此函数用于返回 x 的正弦，其中 x 以半径指定。

表 15.97. 参数 sinf

参数	类型	说明
x	浮点值	参数
<return>	浮点值	返回值

15.4.3.4.7. skin

已弃用：改用 setSkin。

此函数用于切换核心范围的外观。该操作是同步执行的，但是模型对此更改进行异步响应。

表 15.98. 参数 skin

参数	类型	说明
skin	字符串	要切换为的外观，例如 <code>f:setSkin(x,y)</code>
<return>	空	

15.4.3.4.8. smoothstep

此函数用于计算平滑 **hermite** 插值 $3z^2 - 2z^3$ with $z = (x - x_{min}) / (x_{max} - x_{min})$ ，其在 $[x_{min}, x_{max}]$ 范围内，否则为 0。此函数返回位于区间 $[0, 1]$ 的值。



表 15.99. 参数 smoothstep

参数	类型	说明
xmin	浮点值	xmin 值
xmax	浮点值	xmax 值
x	浮点值	要内插的值
<return>	浮点值	返回 hermite 插值 $3z^2 - 2z^3$ with $z = (x-xmin) / (xmax-xmin)$

15.4.3.4.9. sqrtf

此函数用于返回 x 的非负平方根。

表 15.100. 参数 sqrtf

参数	类型	说明
x	浮点值	参数
<return>	浮点值	返回值

15.4.3.4.10. 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 15.101. 参数 string2float

参数	类型	说明
str	字符串	字符串值



参数	类型	说明
<return>	浮点值	返回值

15.4.3.4.11. 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 15.102. 参数 string2int

参数	类型	说明
str	字符串	字符串值
<return>	整数	返回值

15.4.3.4.12. string2string

此函数用于将字符串截断为给定数量的字符。

表 15.103. 参数 string2string

参数	类型	说明
str	字符串	要截断的字符串
len	整数	字符串的最大长度
<return>	字符串	已截断的字符串

15.4.3.4.13. 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)` 返回 abc。
- ▶ `substring("abc", 0, -2)` 返回 ab。
- ▶ `substring("abcd", 1, 3)` 返回 bc。

表 15.104. 参数 substring

参数	类型	说明
str	字符串	输入字符串



参数	类型	说明
startIndex	整数	结果字符串的第一个字符索引
endIndex	整数	不是结果组成部分的第一个字符索引
<return>	字符串	语言字符串

15.4.3.4.14. 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 15.105. 参数 system_time

参数	类型	说明
<return>	整数	以秒为单位的系统时间

15.4.3.4.15. system_time_ms

此函数用于获取以毫秒为单位的当前系统时间。

表 15.106. 参数 system_time_ms

参数	类型	说明
<return>	整数	以毫秒为单位的系统时间

15.4.3.4.16. tanf

此函数用于返回 x 的正切，其中 x 以半径指定。

表 15.107. 参数 tanf

参数	类型	说明
x	浮点值	参数
<return>	浮点值	返回值

15.4.3.4.17. 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 15.108. 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	

15.4.3.4.18. trace_string

此函数用于将字符串写入跟踪日志和连接日志。

表 15.109. 参数 trace_string

参数	类型	说明
str	字符串	要跟踪的文本
<return>	空	

15.4.3.4.19. 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 15.110. 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

15.4.3.4.20. 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 15.111. 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 x 位置
localY	整数	本地坐标的 y 位置



参数	类型	说明
<return>	整数	屏幕坐标的 y 位置

15.4.3.4.21. 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 15.112. 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 x 位置
screenY	整数	屏幕坐标的 y 位置
<return>	整数	本地坐标的 x 位置

15.4.3.4.22. 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 15.113. 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 x 位置
screenY	整数	屏幕坐标的 y 位置
<return>	整数	本地坐标的 y 位置

15.4.3.4.23. trunc

此函数用于将值舍入为最接近的整数值，始终朝向零。

表 15.114. 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



15.4.3.4.24. widgetGetChildCount

此函数用于获取指定控件的子代控件数。

表 15.115. 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

15.5. 事件

表 15.116. 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。

15.5.1. 键事件的十进制代码

表 15.117. 数字盘键的十进制代码

数字盘键	十进制代码
0	5
1	6
2	7
3	8
4	9
5	10
6	11
7	12
8	13
9	14

表 15.118. 功能键的十进制代码

功能键	十进制代码
F1	18
F2	19
F3	20
F4	21
F5	22
F6	23
F7	24
F8	25
F9	26
F10	27
F11	28
F12	29

表 15.119. ASCII 键的十进制代码

ASCII 键	十进制代码
空格	32
a	97
b	98
c	99
d	100
e	101
f	102
g	103
h	104
i	105
j	106
k	107
l	108
m	109
n	110

ASCII 键	十进制代码
o	111
p	112
q	113
r	114
s	115
t	116
u	117
v	118
w	119
x	120
y	121
z	122

15.6. 按钮和图标

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

表 15.120. 常规图标













常规图标	说明
	撤销
	重做
	保存
	验证项目。
	启动模拟运行。
	停止模拟运行。
	打开和关闭项目中心。
	添加元素，例如事件、数据池项或状态机。
	添加元素，例如事件、数据池项或状态机。

表 15.121. 项目中心图标

项目中心图标	说明
	表示可以在其中创建新项目的选项卡。
	表示可以在其中打开现有项目的选项卡。
	表示可以在其中配置多个项目选项的选项卡，例如模型接口、事件组或语言和外观。
	表示可以在其中创建 EB GUIDE 模型导出的选项卡。
	表示可以在其中访问用户文档的选项卡。
	表示可以在其中更改用户界面语言的选项卡。
	表示可以在其中查看已加载哪些插件的选项卡。
	导入项目接口。
	导出项目接口。

表 15.122. 项目编辑器图标

项目编辑器图标	说明
	隐藏元素。
	显示已隐藏的元素。
	同步内容区域和“导航”或“模板”元素。
	<p>打开与属性相关的上下文菜单。</p> <p>下面列出的按钮颜色表示以下内容：</p> <ul style="list-style-type: none">  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。
	打开 EB GUIDE 脚本 编辑器。
	打开命名空间选项或显示已选择的命名空间。






















项目编辑器图标	说明
	显示所有命名空间。
	包括子命名空间。
	打开设置。
	表示数据池项。
	表示转换。
	表示动态状态机列表。
	表示入口动画。
	表示退出动画。
	表示内部转换。
	表示模板。
	跳至此控件的模板。
	筛选搜索结果或列表元素。
Ps	表示 Photoshop 文件。
	表示控件功能属性中有更改的内容。
	表示触发器列表。
	触发按模型接口分组。

表 15.123. 状态图标

状态图标	说明
	表示选择状态。
	表示深度历史状态。
	表示结束状态。
	表示初始状态。
	表示状态机。
	表示浅度历史状态。


状态图标	说明
	表示视图状态。

表 15.124. 基本控件图标
















基本控件图标	说明
	表示 alpha 蒙板。
	表示容器。
	表示自定义控件。
	表示椭圆。
	表示图像。
	表示实例化器。
	表示标签。
	表示矩形。

表 15.125. 动画图标

动画图标	说明
	表示动画控件。
	表示带有常量曲线的动画。
	表示带有快速启动曲线的动画。
	表示带有线性曲线的动画。
	表示带有线性插值曲线的动画。
	表示带有二次曲线的动画。
	表示带有脚本曲线的动画。










动画图标	说明
	表示带有正弦曲线的动画。
	表示带有慢速启动曲线的动画。
	表示更改动画。
	表示入口动画。
	表示退出动画。
	展开动画编辑器。
	最小化动画编辑器。
	表示弹出出现动画。
	表示弹出消失动画。

表 15.126. 3D 控件图标











3D 控件图标	说明
	表示环境光。
	表示摄像头。
	表示定向光。
	表示基于图像的光。
	表示材料。
	表示网格。
	表示点光。
	表示场景图。
	表示场景图节点。
	表示聚光。

表 15.127. “问题”元素图标

















“问题”元素图标	说明
	验证模式。
	表示问题。
	表示一条信息。
	表示操作成功。
	表示警告。
	表示断开的链接。

表 15.128. EB GUIDE Monitor 图标

EB GUIDE Monitor 图标	说明
	触发事件或表示已触发了事件。
	表示已触发关键事件。
	指示是否建立了与主机的连接。
	打开连接配置。
	打开和关闭日志的自动滚动。
	复制所有日志消息。
	删除日志消息。
	导出观察列表。
	表示日志消息。
	触发按模型接口分组。

15.7. 场景

表 15.129. 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的宽度

属性名称	说明
x	触觉状态机的视图在其中于目标设备上渲染的区域的 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的 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
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： ▶ 32-bit (1): RGBA8888 ▶ 16-bit (2): RGB565 ▶ 24-bit (3): RGB888 ▶ 32-bit sRGB (4): 此值使用 GPU 硬件支持。 如果您希望获得对图像控件或对“弥散纹理”控件功能的 sRGB 支持，请使用此值。 ▶ 32-bit sRGB (Emulated) (5): 仅当 32-bit sRGB 未生成正确结果时，才使用此值。
antiAliasing	可能的值： ▶ Off (0): 无抗锯齿 ▶ MSAA 2x (1): 2x 抗锯齿 ▶ MSAA 4x (2): 4x 抗锯齿 另请参见 节6.3 。
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 Windows 的系统中打开另一个窗口进行模拟。

属性名称	说明
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"> ▶ Off (0)：不显示 FPS ▶ On screen (1)：在屏幕上显示 FPS ▶ Console (2)：在控制台中显示 FPS ▶ Console & on screen (3)：在屏幕和控制台中显示 FPS ▶ On screen (large text) (4) ▶ Console & on screen (large text) (5)
Renderer	定义场景的渲染器。 可能的值： <ul style="list-style-type: none"> ▶ OpenGLRenderer ▶ OpenGL3Renderer

注意



在场景配置中使用 sceneID

在场景配置中使用相同 sceneID 时，多个状态机同时对输入处理作出反应。

为避免这种情况并实现只有一个状态机对输入处理作出反应，请为场景配置中的各个状态机分配不同的 sceneID 值。

15.8. 键盘快捷键

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

表 15.130. 键盘快捷键

键盘快捷键	说明
Ctrl+A	选择所有元素
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

键盘快捷键	说明
Enter	在表中，确认输入的值并转至下一个单元格。
Ctrl+Enter	在触发器筛选框中，添加新事件。 在表中，确认输入的值并留在该单元格。
Ctrl+Shift+Insert	在“命名空间”元素中，添加新的命名空间作为现有命名空间的子级。
Alt+F4	关闭活动窗口
Shift+F1	打开 EB GUIDE TF 的用户文档
F1	打开 EB GUIDE Studio 的用户文档
F2	重命名所选元素
Shift+F2	重命名所选元素，以及所选元素的位置，如 EB GUIDE 脚本。适用于数据池项和事件。
F3	在 EB GUIDE 模型中查找所选元素的所有参考
F4	跳至原始模板
F5	启动模拟运行
F6	验证
Del	删除所选元素。
-	在树中，折叠所选元素。
+	在树中，展开所选元素。
*	在树中，展开所选元素和此元素的所有子项。
Up/Down/Left/Right	在内容区域中，将所选状态或控件向上、向下、向左或向右移动一个像素。 在表中，检查元素。
Ctrl 并点击鼠标左键 Shift 并点击鼠标左键或者向上箭头 或向下箭头键	选择多个元素。
Ctrl 并旋转滚轮按钮 Ctrl++ Ctrl+- Ctrl+0	在内容区域中，放大和缩小或将内容区域的缩放比例重置为 100%。

15.9. 控件

15.9.1. 视图

表 15.131. 视图的属性

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	控件的 x 坐标
y	控件的 y 坐标

视图状态和视图模板有视图切换动画的附加属性。视图切换动画适用于入口动画、退出动画、更改动画、弹出出现动画和弹出消失动画。

表 15.132. 视图切换动画属性

属性名称	说明
enabled	定义是否运行动画。
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"> ▶ 0: 动画播放完成 ▶ 1: 动画已取消，由……触发 f:animation_cancel ▶ 2: 控件由于视图转化而损坏 ▶ 3: 动画跳至最后一步，由……触发 f:animation_cancel_end

属性名称	说明
	▶ 4: 动画跳至其第一步, 然后取消播放, 由……触发 <code>f:animation_cancel_reset</code>

15.9.2. 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。

15.9.2.1. 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 15.133. alpha 掩码的属性

属性名称	说明
<code>visible</code>	如果为 <code>true</code> , 则控件及其子代控件可见

属性名称	说明
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
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"> ▶ original size (0) ▶ fit to size (1) ▶ keep aspect ratio (2)

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。使用 **OpenGL ES 2.0** 或更高版本的渲染器支持 **.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。

Alpha 掩码功能不适用于 **9-patch** 图像。**9-patch** 图像的处理方式与 **PNG** 和 **JPEG** 文件格式的处理方式相同。

15.9.2.2. 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 15.134. 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	<p>如果为 true，将反复/双向运行动画。</p> <p>如果为 false，将沿一个方向/单向运行动画。</p> <p>重复次数在 repeat 属性中定义。</p>

属性名称	说明
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"> ▶ 0: 动画播放完成 ▶ 1: 动画已取消，由……触发 f:animation_cancel ▶ 2: 控件由于视图转化而损坏 ▶ 3: 动画跳至最后一步，由……触发 f:animation_cancel_end ▶ 4: 动画跳至其第一步，然后取消播放，由……触发 f:animation_cancel_reset

15.9.2.2.1. 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 15.135. 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

15.9.2.2.2. 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 15.136. 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
start	初始值
end	最终值
target	向其分配生成的值的目标属性

15.9.2.2.3. 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 15.137. 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

15.9.2.2.4. 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 15.138. 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
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

15.9.2.2.5. 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 15.139. 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

15.9.2.2.6. 脚本曲线

脚本曲线是您可以通过 **EB GUIDE** 脚本 自己定义的曲线。如果您希望获得其他曲线无法实现的动画或您自己的自定义动画，请使用脚本曲线。如果想要为控件的移动设置自定义轨迹，则此曲线特别有用。脚本曲线可用于整数、布尔值、浮点和颜色类型。

表 15.140. 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）
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	在 EB GUIDE 脚本 中定义您的曲线函数。提供两个参数： <ul style="list-style-type: none"> ▶ diff: 自上次运行以来的毫秒数。在动画开始时，diff 为 0。 ▶ t_anim: 自动画开始以来的毫秒数。
target	向其分配生成的值的目标属性

15.9.2.2.7. 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 15.141. 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

15.9.2.2.8. 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 15.142. 线性插值曲线的属性

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
target	向其分配生成的值的目标属性

15.9.2.3. 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 15.143. 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

15.9.2.4. 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 15.144. 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
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）

15.9.2.5. 图像

图像用于将图片放入视图中。

表 15.145. 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。OpenGL ES 2.0 或更高版本的渲染器支持 .png 文件和 .jpg 文件。

15.9.2.6. 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 15.146. 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）
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

15.9.2.7. 标签

标签用于将文本放入视图中。

注意



字符替换

当您将文本输入到标签的 text 属性时，将替换以下字符：

- ▶ 序列 \\ 替换为 \。
- ▶ 序列 \\n 替换为 \n。
- ▶ 如果文字显示在一行中，\n 替换为 空格字符。

表 15.147. 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）

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
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

15.9.2.8. 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 15.148. 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）
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

15.9.3. 3D 控件

15.9.3.1. 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、PBR GGX 材料和 PBR Phong 材料的 ambient 颜色属性。

表 15.149. 环境光的属性

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，下限值 0.0 表示无环境光

15.9.3.2. 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 15.150. 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。在第三方 3D 建模软件中创建 3D 模型时，会定义测量单位。
farPlane	视图方向上摄像头的最远距离，可以看到场景。在第三方 3D 建模软件中创建 3D 模型时，会定义测量单位。
fieldOfView	摄像头的垂直视角，以度为单位，最大值为 180
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。 如果投影类型为正交，将使用 fieldOfView 角度计算视景物。

15.9.3.3. 定向光

从一个方向照亮场景的定向光。

表 15.151. 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，下限值 0.0 表示无定向光

15.9.3.4. 基于图像的光

基于图像的光是使用存储在 .pfm 或 .hdr 文件中的现实世界的照明信息来照亮场景的光。 .pfm 或 .hdr 文件用作 IBLGenerator 创建 .ebibl 文件的输入数据。

表 15.152. 基于图像的光的属性

属性名称	说明
enabled	如果为 true, 则启用控件
ibl	手动创建的 IBL 文件 .ebibl
intensity	光的强度, 0.0 表示无基于图像的光

15.9.3.5. 材料

材料使用 Phong 反射模型定义网格表面的视觉外观。

表 15.153. 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。如果添加了“弥散纹理”控件功能, 则此属性不起作用。
emissive	对象的自照明颜色。如果添加了“发光纹理”控件功能, 则此属性不起作用。
shininess	光亮因素 请注意, 仅有 0.0 和 1.0 之间的值, 例如 0.3, 是有效的。 当使用“光亮纹理”控件功能时, 将忽略 shininess 属性。
specular	具有光泽表面的物体反射的颜色。如果添加了“镜面纹理”控件功能或 shininess 属性设置为 0.0, 则属性 specular 不起作用。
opacity	不透明值 请注意, 仅有 0.0 和 1.0 之间的值, 例如 0.3, 是有效的。

15.9.3.6. 网格

网格定义 3D 对象的形状。

表 15.154. 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

属性名称	说明
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)

15.9.3.7. 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 15.155. 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。如果添加了“环境纹理”控件功能，则此属性不起作用。
diffuse	物体被纯白光照射亮时在所有方向上均匀反射的颜色。如果添加了“弥散纹理”控件功能，则此属性不起作用。
emissive	对象的自照明颜色。如果添加了“发光纹理”控件功能，则此属性不起作用。
specular	具有光泽表面的物体反射的颜色。如果添加了“镜面纹理”控件功能或 shininess 属性设置为 0.0，则属性 specular 不起作用。
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。

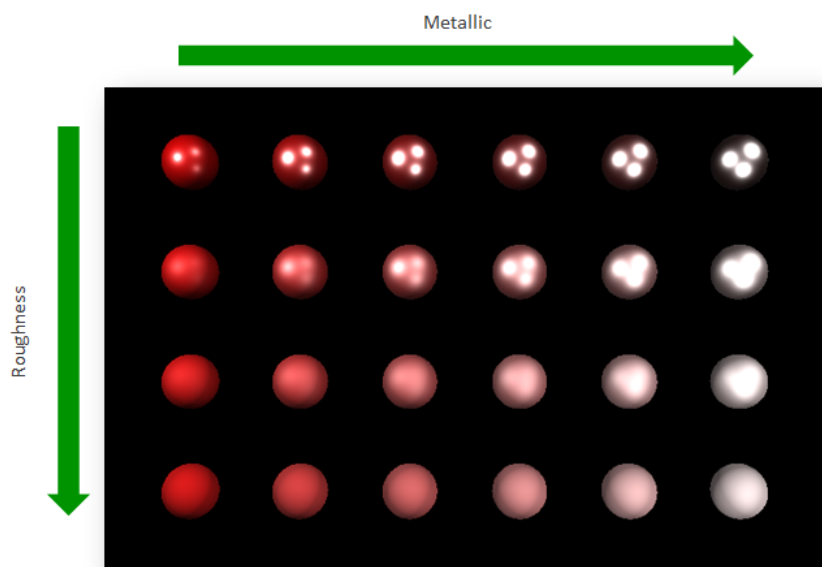


图 15.1. 基于物理的材料示例

15.9.3.8. 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 15.156. 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。如果添加了“环境纹理”控件功能，则此属性不起作用。
diffuse	物体被纯白光照亮时在所有方向上均匀反射的颜色。如果添加了“弥散纹理”控件功能，则此属性不起作用。
emissive	对象的自照明颜色。如果添加了“发光纹理”控件功能，则此属性不起作用。
shininess	光亮因素
specular	具有光泽表面的物体反射的颜色。如果添加了“镜面纹理”控件功能或 shininess 属性设置为 0.0，则属性 specular 不起作用。
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。

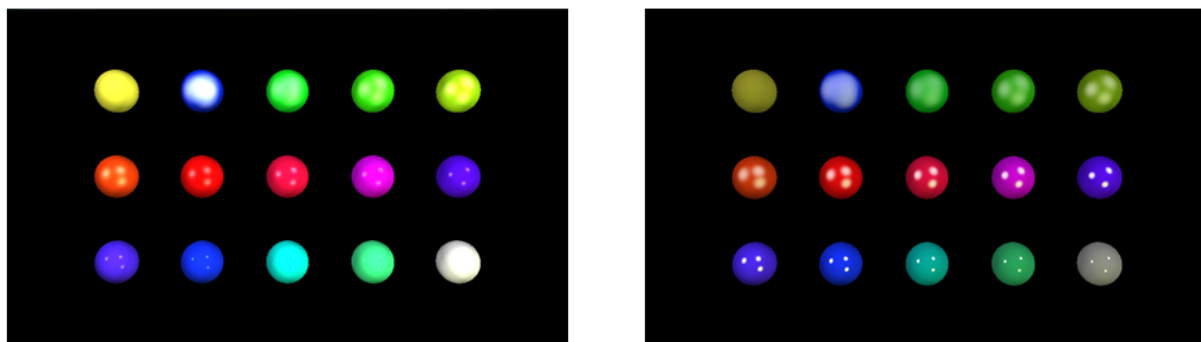


图 15.2. 非标准化材料（左）和标准化材料（右）的示例

15.9.3.9. 点光

点光为场景添加像灯泡一样向所有方向发光的光。

表 15.157. 点光的属性

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，下限值 0.0 表示无点光，上限值取决于衰减因数
attenuationConstant	光强度随着距离的增加而减弱的常数因数。0.0 值表示不使用该因数。
attenuationLinear	光强度随着距离的增加而减弱的线性因数。0.0 值表示不使用该因数。
attenuationQuadratic	光强度随着距离的增加而减弱的二次因数。0.0 值表示不使用该因数。

15.9.3.10. 场景图

场景图用于将 3D 对象放入视图中。

表 15.158. 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



属性名称	说明
gamma	校正场景图的亮度输出。默认值设置为 2.-2。

15.9.3.11. 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
- ▶ 基于图像的光
- ▶ 网格
- ▶ 点光
- ▶ 聚光

表 15.159. 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
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

15.9.3.12. 聚光

聚光添加将照明限制为影响锥的光。

表 15.160. 聚光的属性

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，下限值 0.0 表示无聚光，上限值取决于衰减因数
attenuationConstant	光强度随着距离的增加而减弱的常数因数
attenuationLinear	光强度随着距离的增加而减弱的线性因数
attenuationQuadratic	光强度随着距离的增加而减弱的二次因数
coneAngleInner	灯的内锥角，以度为单位，最大值为 180
coneAngleOuter	灯的外锥角，以度为单位，最大值为 180

15.10. 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。

15.10.1. 公共

15.10.1.1. 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。您可以将单个控件定义为可见，也可以将子代控件组同时定义为可见。要定义组，请将子代控件的索引映射到相同的组值。

表 15.161. “子代可视性选择”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
containerIndex	<p>控制子代控件的可视性。</p> <p>如果未填充 containerMapping，则 containerIndex 使单个子代控件可见。可见的子代控件按控件树中的顺序标识。最高的子代具有 containerIndex 0，下一个 containerIndex 1 等等。</p> <p>如果已填充 containerMapping，则 containerIndex 指一组子代控件。在 containerMapping 中定义组。</p>	输入
containerMapping	<p>使用此属性可以创建子代控件组。“索引”列标识子代控件。“值”列定义组。</p> <p>行数必须匹配子代控件数。否则，不使用映射。</p>	输入

15.10.1.2. 已启用

“已启用”控件功能用于向控件中添加 `enabled` 属性。

表 15.162. “已启用”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
<code>enabled</code>	如果为 <code>true</code> ，则控件会对触摸和按键输入作出反应	输入

15.10.1.3. 已聚焦

“已聚焦”控件功能使控件可以具有输入焦点。

表 15.163. “已聚焦”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
<code>focusable</code>	定义控件是否获得了焦点。可能的值： <ul style="list-style-type: none"> ▶ <code>not focusable</code> (0) ▶ <code>only by touch</code> (1) ▶ <code>only by key</code> (2) ▶ <code>focusable</code> (3) 	输入
<code>focused</code>	如果为 <code>true</code> ，则控件获得了焦点	x

15.10.1.4. 字体指标

使用“字体指标”控件功能，可以更改在标签中使用的字体设置。

有关如何更改 `lineGap` 的说明，请参见[节8.4.2](#)。

限制：

- ▶ “字体指标”控件功能只对标签控件可用。

表 15.164. “字体指标”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
<code>ascender</code>	在字体的基线上方延伸的字母一部分。	输入

属性名称	说明	由 EB GUIDE GTF 设置
descender	在字体的基线下方延伸的字母一部分。	输入
lineGap	每种字体默认包含的行间距。正值增加间距，负值减小间距。	输入

15.10.1.5. 多行

“多行”控件功能用于为标签启用换行符。根据为标签控件设置的 width 属性，在单词或字符之间设置换行符。要标记一行的结尾，还可以使用硬换行符 \n。

限制：

- ▶ “多行”控件功能仅对标签控件可用。

表 15.165. “多行”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
lineOffset	行间距的大小。正值增加间距，负值减小间距。 当 lineOffset 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，lineOffset 定义为 -50 时。	输入
maxLineCount	可见行数的最大值 0 = 无限制	输入

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15.4.3.2.33](#)。

注意



字符替换

当您将文本输入到标签的 text 属性时，将替换以下字符：

- ▶ 序列 \\ \\ 替换为 \。
- ▶ 序列 \\n 替换为 \n。
- ▶ 如果文字显示在一行中，\n 替换为 空格字符。

15.10.1.6. 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 15.166. “已按下”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
pressed	如果为 true，则表示某个键在控件获得焦点的同时被按下	x

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。

15.10.1.7. 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 15.167. “已选中”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
selected	<p>如果为 true，则控件处于选中状态。</p> <p>添加“选择群组”控件功能时，它将评估 buttonID 和 buttonValue 是否相同。如果为 true，则按钮处于选中状态。</p>	<p>输入</p> <p>当添加“选择群组”控件功能时，EB GUIDE GTF 会自动设置属性。</p>

15.10.1.8. 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 15.168. “选择群组”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
buttonId	用于标识按钮群组中的某个按钮的 ID	输入
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。	输入

15.10.1.9. 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 15.169. “旋转”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
currentValue	当前旋转值	x
maxValue	currentValue 属性的最大值	输入
minValue	currentValue 属性的最小值	输入
incValueTrigger	如果为 true，则 currentValue 属性增大 1	输入
incValueReaction	对 currentValue 属性增大的动作反应	输入
decValueTrigger	如果为 true，则当前值减小 1	输入
decValueReaction	对 currentValue 属性减小的动作反应	输入
steps	用于计算 currentValue 属性增大或减小的步骤数	输入
valueWrapAround	可能的值： <ul style="list-style-type: none"> ▶ true: currentValue 属性在反向边框处继续，前提是超过了 minValue 或 maxValue。 ▶ false: currentValue 属性不会减小/增大，前提是超过了 minValue 或 maxValue。 	输入

15.10.1.10. 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 15.170. “文本切断”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
truncationPolicy	<p>对于单行文本，truncationPolicy 属性定义了切断位置。可能的值：</p> <ul style="list-style-type: none"> ▶ leading (0)：文本将在文本开头进行替换 ▶ trailing (1)：文本将在文本结尾进行替换 <p>对于多行文本，truncationPolicy 属性定义了文本替换位置。可能的值：</p> <ul style="list-style-type: none"> ▶ 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。 ▶ 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。 	输入
truncationSymbol	替代所替换的文本部分显示的字符串	输入

15.10.1.11. 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 15.171. “已触摸”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
touchable	如果为 true，则控件会对触摸输入作出反应	输入
touched	如果为 true，则表示控件当前处于被触摸状态	x
touchPolicy	<p>定义如何处理跨控件边界的触摸和移动。可能的值：</p> <ul style="list-style-type: none"> ▶ 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。 ▶ 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。 ▶ Press then react on contact (2)：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。 	输入

属性名称	说明	由 EB GUIDE GTF 设置
touchBehavior	定义触摸评估。可能的值： <ul style="list-style-type: none"> ▶ 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。 ▶ 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。 图像中 alpha 透明度或像素内字母为 O 或 A 的透明像素是无法触摸的。 请注意 Visible pixels 值对标签不起作用。	输入

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。

15.10.2. 效果

15.10.2.1. 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 15.172. “边框”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
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	

15.10.2.2. 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 15.3.

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 15.173. “着色”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
colorationEnabled	如果为 true ，则将使用着色	
colorationColor	用于着色的颜色	

15.10.2.3. 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 15.174. “笔划”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
strokeEnabled	如果为 true ，则将使用笔划	
strokeThickness	边框的粗细（以像素为单位）	
strokeColor	用于渲染边框的颜色	

15.10.3. 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。

15.10.3.1. 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 15.175. “自动焦点”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
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"> ▶ stop at hierarchy (0) ▶ wrap within hierarchy level (1) ▶ step up in hierarchy (2) 	
focusedIndex	当前获得焦点的子代控件作为第 n 个可获得焦点的子代控件的索引	x
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。	

15.10.3.2. 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 15.176. “用户定义焦点”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
focusNext	用于将焦点分配给下一个子代控件的触发器	

属性名称	说明	由 EB GUIDE GTF 设置
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"> ▶ 已定义：将使用用户定义的控件顺序 ▶ 未定义：将改为使用默认的控件顺序 <p>每个子代控件都需要“已集中”控件功能，否则不会对控件进行焦点管理。示例：<code>focusOrder=1 0 2</code> 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"> ▶ stop at hierarchy level (0) ▶ wrap within hierarchy level (1) ▶ step up in hierarchy (2) 	
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。	X
initFocus	初始化时获得焦点的子代控件的索引	

15.10.4. 操作动作

15.10.4.1. 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 15.177. “滑动操作动作”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
onGestureFlick	<p>在识别操作动作后触发的动作反应</p> <p>动作反应参数：</p>	X

属性名称	说明	由 EB GUIDE GTF 设置
	<ul style="list-style-type: none"> ▶ 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 ▶ directionX: 操作动作的方向矢量的 X 部分 ▶ directionY: 操作动作的方向矢量的 Y 部分 	
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	

15.10.4.2. 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 15.178. “按住操作动作”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
onGestureHold	<p>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。</p> <p>动作反应参数：</p> <ul style="list-style-type: none"> ▶ x: 接触位置的 X 坐标 ▶ y: 接触位置的 Y 坐标 	X
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）	

15.10.4.3. 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 15.179. “长按操作动作”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
onGestureLongHold	<p>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</p> <p>动作反应参数：</p> <ul style="list-style-type: none"> ▶ x: 接触位置的 x 坐标 ▶ y: 接触位置的 y 坐标 	x
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）	

15.10.4.4. 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 15.180. “路径动作”控件功能的属性






属性名称	说明	由 EB GUIDE GTF 设置
onPath	<p>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：</p> <ul style="list-style-type: none"> ▶ gestureId: 匹配路径的 ID 	x
onPathStart	接触移出最小框后触发的动作反应（pathMinXBox, pathMinYBox）。	x
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。	x



属性名称	说明	由 EB GUIDE GTF 设置
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）	

15.10.4.4.1. 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 15.181. EB GUIDE 中提供的路径动作样本配置

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

识别码	形状	说明
5		从左到右的波浪形状
6		从右到左的波浪形状

15.10.4.5. 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 15.182. “二指缩放操作动作”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
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"> ▶ ratio: 当前接触距离与初始接触距离之比 ▶ 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 	x
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"> ▶ ratio: 当前接触距离与初始接触距离之比 ▶ centerX: 两点接触之间当前中心点的 X 坐标 	x

属性名称	说明	由 EB GUIDE GTF 设置
	<ul style="list-style-type: none"> centerY: 两点接触之间当前中心点的 Y 坐标 	
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"> ratio: 当前接触距离与初始接触距离之比 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 	X
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）。	

15.10.4.6. 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 15.183. “旋转操作动作”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
onGestureRotateStart	在识别操作动作的开始部分后触发的动作反应	X
onGestureRotateUpdate	所识别的角度或中心点变化时触发的动作反应	X
onGestureRotateEnd	在完成操作动作后触发的动作反应	X
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

15.10.5. 输入处理

15.10.5.1. 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。

15.10.5.2. 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 15.184. “按下的键”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
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"> ▶ keyId: 已处理键的 ID 	X

15.10.5.3. 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 15.185. “放开的键”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
keyShortReleased	控件对放开某个键的动作反应 动作反应参数： ▶ keyId: 已处理键的 ID	X

15.10.5.4. 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 15.186. “键状态已变化”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
keyStatusChanged	控件对按下或放开某个键的动作反应 动作反应参数： ▶ keyId: 已处理键的 ID ▶ status: 状态更改的数字 ID	X

15.10.5.5. 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 15.187. “Unicode 键”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
keyUnicode	控件对 Unicode 键输入的动作反应 动作反应参数:	X

属性名称	说明	由 EB GUIDE GTF 设置
	▶ keyId: 已处理键的 ID	

15.10.5.6. 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 15.188. “移入”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
moveIn	控件对移入其边界的动作反应 动作反应参数: <ul style="list-style-type: none"> ▶ touchId: 用户已点击或释放的触摸屏的 ID ▶ x: X 坐标 ▶ y: Y 坐标 ▶ fingerId: 移过控件的接触点的 ID 	X

15.10.5.7. 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 15.189. “移出”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
moveOut	控件对移出其边界的动作反应 动作反应参数: <ul style="list-style-type: none"> ▶ touchId: 用户已点击或释放的触摸屏的 ID 	X

属性名称	说明	由 EB GUIDE GTF 设置
	<ul style="list-style-type: none"> ▶ x: X 坐标 ▶ y: Y 坐标 ▶ fingerId: 移过控件的接触点的 ID 	

15.10.5.8. 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 15.190. “移过”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
moveOver	控件对其边界内的动作反应 动作反应参数: <ul style="list-style-type: none"> ▶ touchId: 用户已点击或释放的触摸屏的 ID ▶ x: X 坐标 ▶ y: Y 坐标 ▶ fingerId: 移过控件的接触点的 ID 	X

15.10.5.9. 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 15.191. “可移动的”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
moveDirection	控件的移动方向。可能的值:	

属性名称	说明	由 EB GUIDE GTF 设置
	<ul style="list-style-type: none"> ▶ horizontal (0) ▶ vertical (1) ▶ free (2) 	

15.10.5.10. 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 15.192. “旋转”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
rotaryReaction	<p>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。</p> <p>动作反应参数:</p> <ul style="list-style-type: none"> ▶ rotaryId: 整数 ID ▶ increment: 发送了即将发生的事件时旋转输入变化的单位数 	X

15.10.5.11. 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。
- ▶ 如果添加“触摸丢失”，请在“已触摸”控件功能的 touchPolicy 下拉列表框中选择 Press and grab。

触摸丢失不适用于其他触摸策略

表 15.193. “触摸丢失”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
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"> ▶ touchId: 用户已点击或释放的触摸屏的 ID ▶ x: X 坐标 ▶ y: Y 坐标 ▶ fingerId: 移过控件的接触点的 ID 	X

15.10.5.12. 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 15.194. “触摸移动”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
touchMoved	控件对触摸和移动的动作反应 动作反应参数： <ul style="list-style-type: none"> ▶ touchId: 用户已点击或释放的触摸屏的 ID ▶ x: X 坐标 ▶ y: Y 坐标 ▶ fingerId: 移过控件的接触点的 ID 	X

15.10.5.13. 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 15.195. “触摸已按下”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
touchPressed	控件对按下的动作反应 动作反应参数： <ul style="list-style-type: none"> ▶ touchId: 用户已点击或释放的触摸屏的 ID ▶ x: X 坐标 ▶ y: Y 坐标 ▶ fingerId: 移过控件的接触点的 ID 	X

15.10.5.14. 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 15.196. “触摸已放开”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
touchShortReleased	控件对放开的动作反应 动作反应参数： <ul style="list-style-type: none"> ▶ touchId: 用户已点击或释放的触摸屏的 ID ▶ x: X 坐标 ▶ y: Y 坐标 ▶ fingerId: 移过控件的接触点的 ID 	X

15.10.5.15. 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 15.197. “触摸状态已变化”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
touchStatusChanged	<p>控件对其触摸状态变化的动作反应</p> <p>动作反应参数:</p> <ul style="list-style-type: none"> ▶ touchId: 用户已点击或释放的触摸屏的 ID ▶ x: X 坐标 ▶ y: Y 坐标 ▶ touchStatus: 触摸类型的 ID <p>可能的值:</p> <ul style="list-style-type: none"> ▶ 0: 新接触 ▶ 1: 触摸按下 ▶ 2: 触摸移动 ▶ 3: 触摸已放开 ▶ 4: 无触摸的移动 ▶ 5: 触摸消失 <ul style="list-style-type: none"> ▶ fingerId: 移过控件的接触点的 ID 	X

15.10.6. 布局

15.10.6.1. 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 15.198. “绝对布局”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
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。	

15.10.6.2. 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 15.199. “方框布局”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
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"> ▶ horizontal (0) 	

属性名称	说明	由 EB GUIDE GTF 设置
	▶ vertical (1)	

15.10.6.3. 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 15.200. “流式布局”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
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"> ▶ horizontal (0) ▶ vertical (1) 	
horizontalChildAlign	子代控件的水平对齐方式。可能的值： <ul style="list-style-type: none"> ▶ leading (0)：子代控件位于左侧。 ▶ center (1)：子代控件位于中间。 ▶ trailing (2)：子代控件位于右侧。 	
verticalChildAlign	子代控件的垂直对齐方式。可能的值： <ul style="list-style-type: none"> ▶ center (0)：子代控件位于中间。 ▶ top (1)：子代控件位于顶部 	

属性名称	说明	由 EB GUIDE GTF 设置
	▶ bottom (2): 子控件位于底部。	

15.10.6.4. 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 15.201. “栅格布局”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
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	

15.10.6.5. 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 15.202. “布局留白”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
leftMargin	左侧边框的留白	
topMargin	顶部边框的留白	
rightMargin	右侧边框的留白	

属性名称	说明	由 EB GUIDE GTF 设置
bottomMargin	底部边框的留白	

15.10.6.6. 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（以像素为单位）。

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15.10.7.2](#)。

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 15.203. “列表布局”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
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"> ▶ horizontal (0) ▶ vertical (1) 	
scrollOffset	滚动列表的像素数	X
scrollOffsetRebase	如果 scrollOffsetRebase 属性发生更改，则当前 scrollOffset 将转换为 scrollIndex。其余的偏移量将写入 scrollOffset 属性。	
firstListIndex	第一个可见列表元素的列表索引，由控件功能定义	X
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。	X
scrollValue	当前滚动值（以像素为单位）	X

属性名称	说明	由 EB GUIDE GTF 设置
scrollValueMax	最大滚动值（以像素为单位），该值被映射到列表结尾	
scrollValueMin	最小滚动值（以像素为单位），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。	x
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"> ▶ true: scrollValue 属性在反向边框处继续，前提是超过了 scrollValueMin 或 scrollValueMax。 ▶ false: scrollValue 属性不会减小/增大，前提是超过了 scrollValueMin 或 scrollValueMax。 	

15.10.6.7. 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 15.204. “缩放模式”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
scaleMode	图片的缩放模式。可能的值： <ul style="list-style-type: none"> ▶ 0 = original size ▶ 1 = fit to size ▶ 2 = keep aspect ratio 	

15.10.7. 列表管理

15.10.7.1. 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 15.205. “行索引”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
lineIndex	表中当前行的索引	x

15.10.7.2. 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 15.206. “列表索引”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
listIndex	列表中当前控件的索引	x

15.10.7.3. 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 15.207. “模板索引”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
lineTemplateIndex	使用的行模板的索引	x

15.10.7.4. 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 15.208. "Viewport" 控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
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	

15.10.8. 3D

"3D" 类别中的控件功能仅适用于 3D 控件。

15.10.8.1. 抗锯齿模式

表 15.209. "抗锯齿模式"控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
antiAliasing	<p>定义场景图的抗锯齿模式。这将覆盖场景的配置。要使用场景中配置的不同值，请使用 Global (5)。</p> <ul style="list-style-type: none"> ▶ Off (0) ▶ MSAA 2x (1) ▶ MSAA 4x (2) ▶ MSAA 8x (3) ▶ FXAA (4) ▶ Global (5) 	

15.10.8.2. 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 15.210. “摄像头视见区”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
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）	

15.10.8.3. 透明涂层

“透明涂层”控件功能添加了一个反射图层来模拟多层表面。

限制：

- ▶ “透明涂层”控件功能可用于 PBR GGX 材料和 PBR Phong 材料。

表 15.211. “透明涂层”控件功能的属性

属性名称	说明
clearCoatStrength	透明涂层的强度。可能的值在 0.0 到 1.0 之间。
clearCoatRoughness	透明涂层的感观粗糙度。可能的值在 0.0 到 1.0 之间。

15.10.8.4. 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。
- ▶ 当添加“环境纹理”时，将忽略 ambient 属性。

表 15.212. “环境纹理”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
ambientTexture	纹理的文件名	
ambientTextureAddressModeU	<p>u 方向的纹理的地址模式。可能的值：</p> <ul style="list-style-type: none"> ▶ repeat (0)：当在纹理边界之外访问时，重复纹理。也称为 wrap 或 tile ▶ clamp (1)：当在纹理边界之外访问时，使用纹理边缘处的像素。 	
ambientTextureAddressModeV	<p>v 方向的纹理的地址模式。可能的值：</p> <ul style="list-style-type: none"> ▶ repeat (0)：当在纹理边界之外访问时，重复纹理。也称为 wrap 或 tile ▶ clamp (1)：当在纹理边界之外访问时，使用纹理边缘处的像素。 	
ambientFilterMode	<p>纹理的筛选模式。可能的值：</p> <ul style="list-style-type: none"> ▶ 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。 ▶ 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。 ▶ trilinear (2)：更贵，但比线性筛选效果更好。 	
diffuseSRGB	<p>如果启用了此属性，将使用 sRGB 色域渲染 ambientTexture 中选择的纹理。</p> <p>请注意，要使用 sRGB 功能，请打开项目中心并转至“配置”>“配置文件”，为 colorMode 属性选择 32-bit sRGB (4) 或 32-bit sRGB (Emulated) (5)。</p>	

15.10.8.5. 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。

- ▶ 当添加“弥散纹理”时，将忽略 diffuse 属性。

表 15.213. “弥散纹理”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
diffuseTexture	纹理的文件名	
diffuseTextureAddressModeU	u 方向的纹理的地址模式。可能的值： <ul style="list-style-type: none"> ▶ repeat (0)：当在纹理边界之外访问时，重复纹理。也称为 wrap 或 tile ▶ clamp (1)：当在纹理边界之外访问时，使用纹理边缘处的像素。 	
diffuseTextureAddressModeV	v 方向的纹理的地址模式。可能的值： <ul style="list-style-type: none"> ▶ repeat (0)：当在纹理边界之外访问时，重复纹理。也称为 wrap 或 tile ▶ clamp (1)：当在纹理边界之外访问时，使用纹理边缘处的像素。 	
diffuseFilterMode	纹理的筛选模式。可能的值： <ul style="list-style-type: none"> ▶ 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。 ▶ 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。 ▶ trilinear (2)：更贵，但比线性筛选效果更好。 	
diffuseSRGB	如果启用了此属性，将使用 sRGB 色域渲染 diffuseTexture 中选择的纹理。 请注意，要使用 sRGB 功能，请打开项目中心并转至“配置” > “配置文件”，为 colorMode 属性选择 32-bit sRGB (4) 或 32-bit sRGB (Emulated) (5)。	

15.10.8.6. 放射纹理

“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。
- ▶ 当添加“发光纹理”时，将忽略 emissive 属性。

表 15.214. “发光纹理”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
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"> ▶ repeat (0)：当在纹理边界之外访问时，重复纹理。也称为 wrap 或 tile ▶ clamp (1)：当在纹理边界之外访问时，使用纹理边缘处的像素。 	
emissiveTextureAddressModeV	<p>v 方向的纹理的地址模式。可能的值：</p> <ul style="list-style-type: none"> ▶ repeat (0)：当在纹理边界之外访问时，重复纹理。也称为 wrap 或 tile ▶ clamp (1)：当在纹理边界之外访问时，使用纹理边缘处的像素。 	
emissiveFilterMode	<p>纹理的筛选模式。可能的值：</p> <ul style="list-style-type: none"> ▶ 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。 ▶ 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。 ▶ trilinear (2)：更贵，但比线性筛选效果更好。 	
diffuseSRGB	<p>如果启用了此属性，将使用 sRGB 色域渲染 emissiveTexture 中选择的纹理。</p> <p>请注意，要使用 sRGB 功能，请打开项目中心并转至“配置”>“配置文件”，为 colorMode 属性选择 32-bit sRGB (4) 或 32-bit sRGB (Emulated) (5)。</p>	

15.10.8.7. 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 15.215. “光地图纹理”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
lightMapTexture	纹理的文件名	
lightMapTextureAddressModeU	<p>u 方向的纹理的地址模式。可能的值:</p> <ul style="list-style-type: none"> ▶ repeat (0): 当在纹理边界之外访问时, 重复纹理。也称为 wrap 或 tile ▶ clamp (1): 当在纹理边界之外访问时, 使用纹理边缘处的像素。 	
lightMapTextureAddressModeV	<p>v 方向的纹理的地址模式。可能的值:</p> <ul style="list-style-type: none"> ▶ repeat (0): 当在纹理边界之外访问时, 重复纹理。也称为 wrap 或 tile ▶ clamp (1): 当在纹理边界之外访问时, 使用纹理边缘处的像素。 	
lightMapFilterMode	<p>纹理的筛选模式。可能的值:</p> <ul style="list-style-type: none"> ▶ 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。 ▶ 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。 ▶ trilinear (2): 更贵, 但比线性筛选效果更好。 	

15.10.8.8. 金属纹理

“金属”控件功能将扩展配置值添加到材料。纹理控制 PBR GGX 材料和 PBR Phong 材料控件的金属参数。

限制:

- ▶ “金属纹理”控件功能可用于 PBR GGX 材料和 PBR Phong 材料。
- ▶ “金属纹理”是灰度图像。对于 RGB 颜色图像, 仅使用红色通道。
- ▶ 当添加“金属纹理”时, 将忽略 metallic 属性。

表 15.216. “金属纹理”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
metallicTexture	纹理的文件名	
metallicMinFactor	作为浮点数来内插纹理值的最小金属参数	
metallicMaxFactor	作为浮点数来内插纹理值的最大金属参数	
metallicTextureAddressModeU	<p>u 方向的纹理的地址模式。可能的值：</p> <ul style="list-style-type: none"> ▶ repeat (0)：当在纹理边界之外访问时，重复纹理。也称为 wrap 或 tile ▶ clamp (1)：当在纹理边界之外访问时，使用纹理边缘处的像素。 	
metallicTextureAddressModeV	<p>v 方向的纹理的地址模式。可能的值：</p> <ul style="list-style-type: none"> ▶ repeat (0)：当在纹理边界之外访问时，重复纹理。也称为 wrap 或 tile ▶ clamp (1)：当在纹理边界之外访问时，使用纹理边缘处的像素。 	
metallicFilterMode	<p>纹理的筛选模式。可能的值：</p> <ul style="list-style-type: none"> ▶ 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。 ▶ 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。 ▶ trilinear (2)：更贵，但比线性筛选效果更好。 	

15.10.8.9. 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 15.217. “正常贴图”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
normalMapTexture	纹理的文件名	

属性名称	说明	由 EB GUIDE GTF 设置
normalMapTextureAddressModeU	<p>u 方向的纹理的地址模式。可能的值：</p> <ul style="list-style-type: none"> ▶ repeat (0)：当在纹理边界之外访问时，重复纹理。也称为 wrap 或 tile ▶ clamp (1)：当在纹理边界之外访问时，使用纹理边缘处的像素。 	
normalMapTextureAddressModeV	<p>v 方向的纹理的地址模式。可能的值：</p> <ul style="list-style-type: none"> ▶ repeat (0)：当在纹理边界之外访问时，重复纹理。也称为 wrap 或 tile ▶ clamp (1)：当在纹理边界之外访问时，使用纹理边缘处的像素。 	
normalMapFilterMode	<p>纹理的筛选模式。可能的值：</p> <ul style="list-style-type: none"> ▶ 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。 ▶ 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。 ▶ trilinear (2)：更贵，但比线性筛选效果更好。 	

15.10.8.10. 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 15.218. “不透明纹理”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
opaqueTexture	纹理的文件名	
opaqueTextureAddressModeU	<p>u 方向的纹理的地址模式。可能的值：</p> <ul style="list-style-type: none"> ▶ repeat (0)：当在纹理边界之外访问时，重复纹理。也称为 wrap 或 tile 	

属性名称	说明	由 EB GUIDE GTF 设置
	<ul style="list-style-type: none"> ▶ clamp (1): 当在纹理边界之外访问时, 使用纹理边缘处的像素。 	
opaqueTextureAddressModeV	v 方向的纹理的地址模式。可能的值: <ul style="list-style-type: none"> ▶ repeat (0): 当在纹理边界之外访问时, 重复纹理。也称为 wrap 或 tile ▶ clamp (1): 当在纹理边界之外访问时, 使用纹理边缘处的像素。 	
opaqueFilterMode	纹理的筛选模式。可能的值: <ul style="list-style-type: none"> ▶ 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。 ▶ 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。 ▶ trilinear (2): 更贵, 但比线性筛选效果更好。 	

15.10.8.11. 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 15.219. “反射纹理”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
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	
reflectionFilterMode	纹理的筛选模式。可能的值:	

属性名称	说明	由 EB GUIDE GTF 设置
	<ul style="list-style-type: none"> ▶ 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。 ▶ 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。 ▶ trilinear (2): 更贵, 但比线性筛选效果更好。 	

注意



“反射纹理”控件功能

仅当为以下所有属性选择了图像文件时, EB GUIDE Studio 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和二次方形状必须相同。

15.10.8.12. 粗糙纹理

“粗糙纹理”控件功能将扩展配置值添加到材料。纹理控制 PBR GGX 材料控件的粗糙参数。

限制:

- ▶ “粗糙纹理”控件功能可用于 PBR GGX 材料。
- ▶ “粗糙纹理”是灰度图像。对于 RGB 颜色图像, 仅使用红色通道。
- ▶ 当“粗糙纹理”活动时, 将忽略 roughness 属性。

表 15.220. “粗糙纹理”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
roughnessTexture	纹理的文件名	
roughnessMinFactor	作为浮点数来内插纹理值的最小粗糙参数	
roughnessMaxFactor	作为浮点数来内插纹理值的最大粗糙参数	

属性名称	说明	由 EB GUIDE GTF 设置
roughnessTextureAddressModeU	u 方向的纹理的地址模式。可能的值： <ul style="list-style-type: none"> ▶ repeat (0)：当在纹理边界之外访问时，重复纹理。也称为 wrap 或 tile ▶ clamp (1)：当在纹理边界之外访问时，使用纹理边缘处的像素。 	
roughnessTextureAddressModeV	v 方向的纹理的地址模式。可能的值： <ul style="list-style-type: none"> ▶ repeat (0)：当在纹理边界之外访问时，重复纹理。也称为 wrap 或 tile ▶ clamp (1)：当在纹理边界之外访问时，使用纹理边缘处的像素。 	
roughnessFilterMode	纹理的筛选模式。可能的值： <ul style="list-style-type: none"> ▶ 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。 ▶ 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。 ▶ trilinear (2)：更贵，但比线性筛选效果更好。 	

15.10.8.13. 光亮纹理

“光亮纹理”控件功能将扩展配置值添加到材料。纹理通过将纹理值与标量 `shininess` 属性相乘来调节光亮强度。

限制：

- ▶ “光亮纹理”控件功能可用于材料和 **PBR Phong** 材料。
- ▶ “光亮纹理”是灰度图像。对于 **RGB** 颜色图像，仅使用红色通道。
- ▶ 当使用“光亮纹理”控件功能时，将忽略 `shininess` 属性。

表 15.221. “光亮纹理”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
shininessTexture	纹理的文件名	

属性名称	说明	由 EB GUIDE GTF 设置
shininessMinFactor	作为浮点数来内插纹理值的最小光亮参数	
shininessMaxFactor	作为浮点数来内插纹理值的最大光亮参数	
shininessTextureAddressModeU	u 方向的纹理的地址模式。可能的值： <ul style="list-style-type: none"> ▶ repeat (0)：当在纹理边界之外访问时，重复纹理。也称为 wrap 或 tile ▶ clamp (1)：当在纹理边界之外访问时，使用纹理边缘处的像素。 	
shininessTextureAddressModeV	v 方向的纹理的地址模式。可能的值： <ul style="list-style-type: none"> ▶ repeat (0)：当在纹理边界之外访问时，重复纹理。也称为 wrap 或 tile ▶ clamp (1)：当在纹理边界之外访问时，使用纹理边缘处的像素。 	
shininessFilterMode	纹理的筛选模式。可能的值： <ul style="list-style-type: none"> ▶ 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。 ▶ 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。 ▶ trilinear (2)：更贵，但比线性筛选效果更好。 	

15.10.8.14. 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。
- ▶ 当添加“镜面纹理”时，将忽略 specular 属性。

表 15.222. “镜面纹理”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
specularTexture	纹理的文件名	

属性名称	说明	由 EB GUIDE GTF 设置
specularTextureAddressModeU	<p>u 方向的纹理的地址模式。可能的值：</p> <ul style="list-style-type: none"> ▶ repeat (0)：当在纹理边界之外访问时，重复纹理。也称为 wrap 或 tile ▶ clamp (1)：当在纹理边界之外访问时，使用纹理边缘处的像素。 	
specularTextureAddressModeV	<p>v 方向的纹理的地址模式。可能的值：</p> <ul style="list-style-type: none"> ▶ repeat (0)：当在纹理边界之外访问时，重复纹理。也称为 wrap 或 tile ▶ clamp (1)：当在纹理边界之外访问时，使用纹理边缘处的像素。 	
specularFilterMode	<p>纹理的筛选模式。可能的值：</p> <ul style="list-style-type: none"> ▶ 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。 ▶ 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。 ▶ trilinear (2)：更贵，但比线性筛选效果更好。 	
diffuseSRGB	<p>如果启用了此属性，将使用 sRGB 色域渲染 specularTexture 中选择的纹理。</p> <p>请注意，要使用 sRGB 功能，请打开项目中心并转至“配置” > “配置文件”，为 colorMode 属性选择 32-bit sRGB (4) 或 32-bit sRGB (Emulated) (5)。</p>	

15.10.8.15. Texture coordinate transformation

"Texture coordinate transformation" 控件功能可以修改材料纹理的坐标。此功能可用于材料、PBR Phong 材料和 PBR GGX 材料。





表 15.223. "Texture coordinate transformation" 控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
uOffset	定义 u 方向上纹理坐标的偏移量	

属性名称	说明	由 EB GUIDE GTF 设置
vOffset	定义 v 方向上纹理坐标的偏移量	
uScale	定义 u 方向上纹理坐标的缩放	
vScale	定义 v 方向上纹理坐标的缩放	



例 15.4. "Texture coordinate transformation" 的示例

uOffset: 1.0	0.5	1.0	1.0	1.0
vOffset: 1.0	1.0	0.5	1.0	1.0
uScale: 1.0	1.0	1.0	0.5	3.0
vScale: 1.0	1.0	1.0	0.5	3.0
				
				

15.10.8.16. 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。¹

表 15.224. “色调映射”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
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。	输入

¹2002 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)。

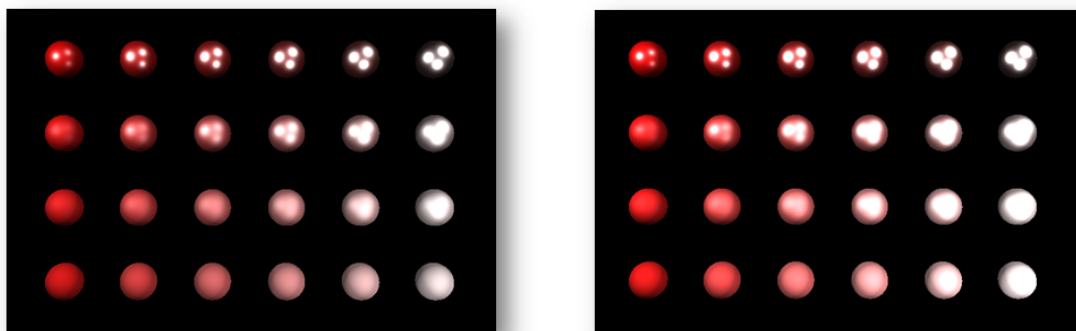


图 15.3. 不带色调映射的图像（左）和带色调映射的图像（右）的示例

15.10.8.17. 摄像头光晕

“摄像头光晕”控件功能是一种后期处理效果。它会产生从图像明亮区域的边界延伸的条纹，从而造成极亮光线覆盖捕捉场景的摄像机或眼睛的错觉。

限制：

- ▶ “摄像头光晕”控件功能仅可用于摄像头。

表 15.225. “摄像头光晕”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
enabled	如果为 true，则将光晕效果应用于摄像头。	
threshold	强度（亮度）值，定义受影响区域之间的划分。可能的值在 0.0 到 1.0 之间。 如果将“色调映射”控件功能添加到父场景图，则 threshold 可以大于 1.0。	输入
strength	光晕效果的强度。	输入
radius	光晕半径。	输入

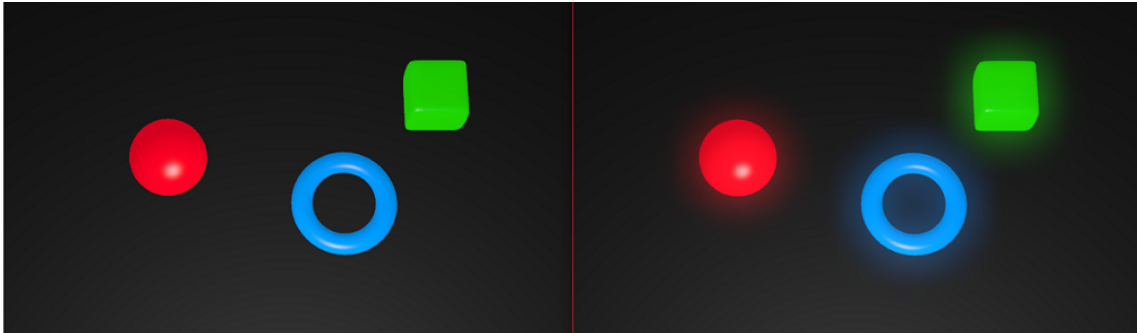


图 15.4. 没有光晕效果（左）和有光晕效果（右）的对象示例

15.10.8.18. 景深

“景深”控件功能是一种后期处理效果。它可以仅在特定距离的物体上聚焦，从而模拟摄像机镜头的聚焦属性。离摄像头较近或较远的物体会显得模糊。

限制：

- ▶ “景深”控件功能仅可用于摄像头。

表 15.226. “景深”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
enabled	如果为 true，则将景深效果应用于摄像头。	
focusDistance	在世界空间中距摄像头位置的距离。它应该在摄像头的 nearPlane 和 farPlane 属性之间。	输入
fstop	定义焦点区域的大小。	输入

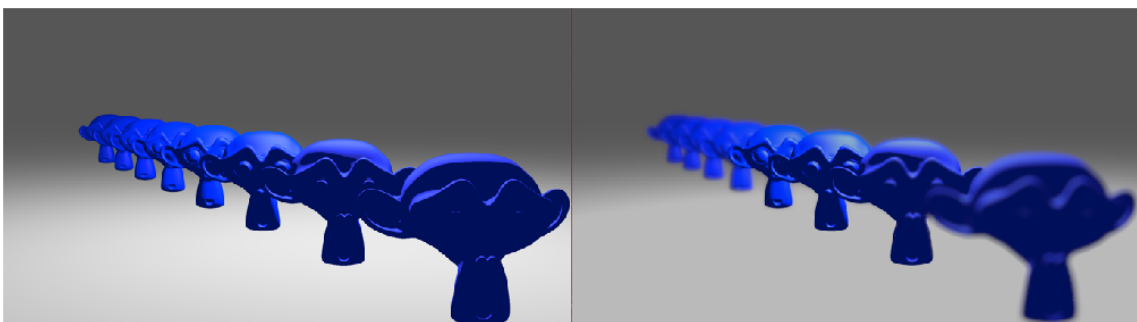


图 15.5. 禁用景深效果（左）和启用景深效果（右）的对象示例

15.10.8.19. 屏幕空间环境光遮挡

“屏幕空间环境光遮挡”控件功能是一种后期处理效果。它通过计算环境光将如何影响场景，提供了间接照明的近似值。

限制：

- ▶ “屏幕空间环境光遮挡”控件功能仅可用于场景图。

表 15.227. “屏幕空间环境光遮挡”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
radius	定义围绕兴趣点的示例球体半径，将在其中使用 samplesCount。值是介于 0.0 和 1.0 之间的正值。	
fallOff	定义最小距离的内球，从可遮挡的点开始。值是介于 0.0 和 1.0 之间的正值。值应该低于 radius 的值。	
samplesCount	样本球体中的样本数。数字越高，遮挡的质量越高。值介于 8 到 64 个样本之间。	

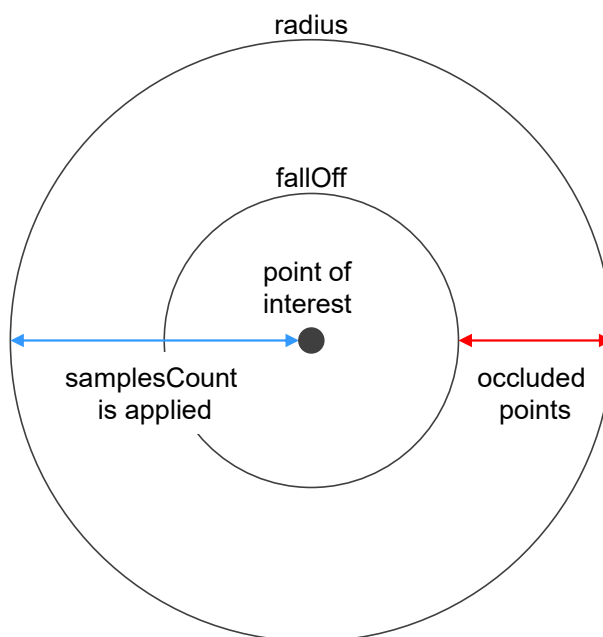


图 15.6. 样本球体和“屏幕空间环境光遮挡”控件功能的属性

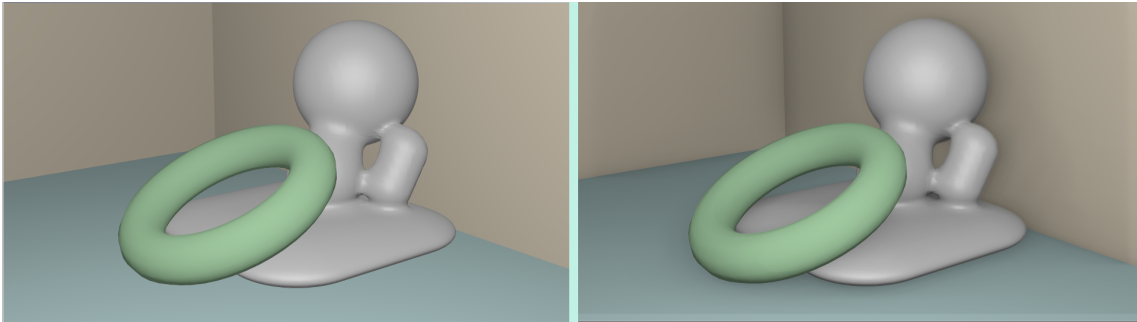


图 15.7. 禁用（左）和开启（右）“屏幕空间环境光遮挡”的对象示例

15.10.9. 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

15.10.9.1. 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 15.228. “中枢”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
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	

属性名称	说明	由 EB GUIDE GTF 设置
pivotZ	Z 轴上相当于父代控件的中枢点（如果控件为场景图）	

15.10.9.2. 旋转

“旋转”控件功能用于旋转控件及其子树。

表 15.229. “旋转”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
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。	
rotationAngleY	Y 轴上的旋转角度。此属性仅影响场景图。	
rotationAngleZ	Z 轴上的旋转角度	

15.10.9.3. 缩放

“缩放”控件特性用于缩放控件及其子树。

表 15.230. “缩放”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
scalingEnabled	定义是否使用缩放	
scalingX	X 轴上的缩放（以百分比为单位）	
scalingY	Y 轴上的缩放（以百分比为单位）	
scalingZ	Z 轴上的缩放（如果控件是场景图）	

15.10.9.4. 剪切

“剪切”控件功能用于改变控件及其子树的形状。

表 15.231. “剪切”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
shearingEnabled	定义是否使用剪切	



属性名称	说明	由 EB GUIDE GTF 设置
shearingXbyY	X 轴对 Y 轴的剪切	
shearingXbyZ	X 轴对 Z 轴的剪切（如果控件是场景图）	
shearingYbyX	Y 轴对 X 轴的剪切	
shearingYbyZ	Y 轴对 Z 轴的剪切（如果控件是场景图）	
shearingZbyX	Z 轴对 X 轴的剪切（如果控件是场景图）	
shearingZbyY	Z 轴对 Y 轴的剪切（如果控件是场景图）	

15.10.9.5. 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 15.232. “转换”控件功能的属性

属性名称	说明	由 EB GUIDE GTF 设置
translationEnabled	定义是否使用转换	
translationX	X 轴上的转换	
translationY	Y 轴上的转换	
translationZ	Z 轴上的转换（如果控件是场景图）	

16. EB GUIDE Studio 的安装

16.1. 背景信息

16.1.1. 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



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。

16.1.2. 系统要求

遵循以下设置：

表 16.1. EB GUIDE Studio 的建议设置

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（64 位），Windows 10（64 位）
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

表 16.2. EB GUIDE SDK 的建议设置

开发环境 (IDE)	Microsoft Visual Studio 2013 或更新版本
文件集成	CMake

16.2. 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 是您要从中下载 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/>。

16.3. 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

选择安装文件夹。

默认安装文件夹是 C:/Program Files/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。

16.4. 卸载 EB GUIDE



卸载 EB GUIDE

注意



永久删除 **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

ADAS 电子控制单元 先进的驾驶员辅助系统电子控制单元

基于 ISO 26262 硬件设计和软件体系结构的开放式可扩展平台，托管来自 Autoliv、客户或第三方的算法。

ADASIS 先进的驾驶员辅助系统接口规格

API 应用程序编程接口

应用程序 在 EB GUIDE 脚本环境中，应用程序是通过例如事件系统和数据池在 EB GUIDE GTF 运行时与一个或多个 EB GUIDE 模型交互的计算机软件。例如，应用程序是诸如媒体播放器之类的娱乐软件，诸如电话之类的通信软件等。参见应用程序接口。

外表 在 EB GUIDE 中，外表是指在 EB GUIDE GTF 运行时应用的 EB GUIDE 模型的外观相关修改。存在两种类型的外表：可用于为 EB GUIDE 模型定义不同外形的外观，以及语言。

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 arware	EB GUIDE arware 是一个软件框架，可用于创建增强现实解决方案以增强驾驶体验。
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。
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 模型和将此 HMI 模型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 项目	EB GUIDE 项目包含一个 EB GUIDE 模型以及在目标设备上运行 EB GUIDE 模型所需的设置。
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。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 的运行时间环境。它包含 EB GUIDE GTF。它是运行 EB GUIDE 模型所必需的。
扩展	在 EB GUIDE 中，扩展是任何 EB GUIDE 产品的补充。扩展是库（.dll 或 .so 文件）形式的插件，可为 EB GUIDE Studio、EB GUIDE GTF 或 EB GUIDE Monitor 添加某些功能。例如，这样的功能可以是数据导出器或附加的控件功能。

G

GL	显卡程序库
GPS	全球定位系统
GUI	图形用户界面

H

HMI	人机接口
HMI 模型	HMI 模型是定义 HMI 的外观和行为的所有元素的总和。它是使用 HMI 软件工具创建的。

I

IPC	进程间通信
IBL	基于图像的光照
IBLGenerator	IBLGenerator 是用于处理环境照明信息的工具。

L

库	库是 EB GUIDE 中使用的预编译软件部件、子例程或程序的集合。EB GUIDE 项目所必需的库在项目中心定义。支持两种文件类型：.dll 和 .so。
---	--

M

MEF	托管扩展框架。参见 https://docs.microsoft.com/en-us/dotnet/framework/mef/ 。
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。
模型接口	定义的模型元素集，用于在目标设备上导出的 EB GUIDE 模型之间进行通信。
多字体支持	作为单个字体的不同字符范围的多字体聚合。
MVC	模型视图控制器

N

命名空间	在 EB GUIDE Studio 中，可以使用命名空间创建模型元素组，例如数据源项和事件。这些组通常具有已定义的功能。每个命名空间都为模型元素创建命名范围，因此不同命名空间中的模型元素可以具有相同的名称。
------	---

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。

RomFS

只读存储器文件系统

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

转化

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。

VTA 视图切换动画

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。

WPF Windows Presentation Foundation。参见<https://docs.microsoft.com/en-us/dotnet/framework/wpf/>。

索引

符号

- .psd 文件格式, 142
- 3D 图形, 36, 72, 73, 407
 - 受支持的格式, 36, 72, 73
 - 基于图像的光照, 72
 - 导入, 37, 259
 - 添加, 141
 - 网格, 73
- 3D 对象, 36
- 3D 控件, 36, 72, 73, 109
 - 参考, 348
- 不透明纹理
 - 参考, 391
- 中枢
 - 参考, 401
- 事件, 59
 - EB GUIDE 脚本 函数, 84
 - 事件ID, 59
 - 事件群组, 59
 - 关键事件, 327
 - 删除, 179
 - 参考, 327
 - 在 EB GUIDE Monitor 中触发, 210
 - 复制, 176
 - 映射键, 178
 - 添加, 176
 - 添加参数, 176
 - 粘贴, 176
- 事件系统, 59
- 事件组
 - 添加, 177
- 二指缩放操作动作
 - 参考, 368
- 二次曲线
 - 参考, 342
- 人机接口, 409
- 伽马
 - 渲染, 263
- 伽马校正

- 概念, 63
- 渲染, 63
- 元素
 - 取消对接, 53
 - 对接, 53
- 光亮纹理
 - 参考, 394
- 光地图纹理
 - 参考, 388
- 入口动画, 169
 - 参考, 338
- 全球定位系统, 408
- 共享库, 410
- 内容区域
 - 项目中心, 43
 - 项目编辑器, 47
- 内部转化
 - 添加, 128
- 列表
 - 创建, 246
 - 数据类型, 290
 - 编辑, 180
- 列表布局
 - 参考, 381
- 列表索引
 - 参考, 383
- 剪切
 - 参考, 402
- 动作
 - 添加, 127
- 动态状态机
 - 添加, 116, 228
- 动画, 38, 169
 - 入口动画, 39, 169
 - 参考, 340
 - 更改动画, 39
 - 添加, 137, 252, 275
 - 添加更改动画, 170
 - 退出动画, 39, 169
 - 重新排列动画, 170
- 反射纹理
 - 参考, 392

- 发光纹理
 - 参考, 387
- 取消对接
 - 元素, 53
- 只读存储器文件系统, 410
- 可移动的
 - 参考, 373
- 可视性, 355 (见 控件可视性)
- 可视性组, 355
- 命令区
 - 项目编辑器, 50
- 命令行, 201, 284
 - EB GUIDE Monitor 选项, 285
 - EB GUIDE Studio 的选项, 284
 - 导出, 194
 - 语法, 284
 - 验证, 193
- 命令行解释器 (见 命令行)
- 命名空间, 67, 409
 - 删除, 188
 - 添加, 186
 - 添加模型元素, 187
 - 移动模型元素, 187
 - 重命名, 186
- 图像
 - 9-patch, 72
 - 参考, 346
 - 受支持的格式, 72
 - 数据类型, 288
 - 添加, 132
- 图标
 - 用户界面, 329
- 场景
 - 参考, 334
 - 配置, 198, 334
- 场景图, 36, 72, 73
 - 参考, 353
 - 添加, 141
 - 自定义, 259
- 场景图节点
 - 参考, 354
- 基于图像的光, 409
 - 参考, 349
- 基于图像的光照, 64, 72
 - IBLGenerator, 64
 - 导入, 144
 - 渲染, 65
- 基本控件, 109
 - 参考, 339
- 复制
 - 事件, 176
 - 数据池项, 180
- 外表, 407
 - 外观, 89
 - 语言, 65
- 外观
 - 切换, 168
 - 删除, 168
 - 支持, 89
 - 添加, 166
- 多字体支持, 71, 409
 - 为类型列表字体添加, 162
 - 添加, 161
- 多点采样, 40
- 多行
 - 参考, 357
- 子代可视性选择
 - 参考, 355
- 字体, 71
 - opentype 字体, 71
 - truetype 字体, 71
 - 位图字体, 71
 - 多字体支持, 71
 - 数据类型, 288
 - 更改, 158
- 字体指标
 - 参考, 356
- 字体设置
 - 更改, 158
- 字符串
 - 数据类型, 290
- 定向光
 - 参考, 349
- 实例化器

- 使用, 246
- 参考, 347
- 添加, 135
- 行模板, 135, 347
- 容器
 - 参考, 345
 - 添加, 135
- 对接
 - 元素, 53
- 导入
 - 语言依赖项文本, 200
- 导出
 - 语言依赖项文本, 199
- 导航元素
 - 项目编辑器, 44
- 导航区
 - 项目中心, 42
- 屏幕空间环境光遮挡
 - 参考, 399
- 属性元素
 - 命令区, 46
 - 最佳实践, 227
 - 项目编辑器, 46
- 工具箱 (见 工具箱元素)
- 工具箱元素
 - 项目编辑器, 46
- 已启用
 - 参考, 356
- 已按下
 - 参考, 357
- 已聚焦
 - 参考, 356
- 已触摸
 - 参考, 360
- 已选中
 - 参考, 358
- 布尔值
 - 数据类型, 286
- 布尔值列表
 - 数据类型, 286
- 布局留白
 - 参考, 380
- 常量曲线
 - 参考, 341
- 库 (见 扩展)
 - 添加, 196
- 应用程序, 407
- 应用程序编程接口, 40, 407
- 建模员, 18
 - 必要知识, 18
- 弥散纹理
 - 参考, 386
- 弹出出现动画
 - 参考, 338
- 弹出消失动画
 - 参考, 338
- 快速启动曲线
 - 参考, 341
- 慢速启动曲线
 - 参考, 342
- 手指 ID, 107
- 扩展, 62, 408
 - EB GUIDE Studio, 219
 - EB GUIDE 模型, 219
 - EB GUIDE 用户界面, 221
- 扩展开发人员, 19
 - 必要知识, 19
- 抗锯齿, 40
- 按下的键
 - 参考, 370
- 按住操作动作
 - 参考, 365
- 按钮
 - 用户界面, 329
- 控件, 108, 411
 - 3D 控件, 109
 - 位置, 145
 - 分组, 135
 - 删除, 145
 - 基本, 109
 - 控件功能, 111, 112
 - 控件属性, 111
 - 控件模板, 112
 - 模板, 109

- 添加, 131
- 管理可视性, 154
- 类型, 108
- 自定义控件, 109
- 视图, 108
- 调整大小, 146
- 链接到控件属性, 147
- 链接到数据池项, 149
- 控件功能, 111, 111, 112
 - 列表管理, 115
 - 删除, 157
 - 添加, 155
 - 添加路径动作, 243
 - 焦点, 113
- 控件可视性, 154
 - 单个子代控件, 154
 - 多个子代控件, 154
- 控件属性, 111
 - 使用 EB GUIDE 脚本 访问, 83
 - 控件功能属性, 111
 - 模板, 112
 - 添加用户定义的属性, 150
 - 用户定义的属性, 111, 150
 - 类型列表, 153
 - 链接到控件属性, 147
 - 链接到数据池项, 149
 - 默认属性, 111
- 控件模板 (见 模板)
- 控制台 (见 命令行)
- 插件 (见 扩展)
- 摄像头
 - 参考, 349
- 摄像头光晕
 - 参考, 398
- 摄像头视见区
 - 参考, 385
- 操作动作, 106
 - 参考, 364, 370
 - 多点触摸输入, 107
 - 手指 ID, 107
 - 路径动作, 106
 - 非路径动作, 106
- 操作动作 ID
 - 参考, 367
- 操作系统, 409
- 放开的键
 - 参考, 370
- 效果
 - 控件功能, 361
- 数据池, 54, 407
 - 使用 EB GUIDE 脚本 访问, 82
 - 数据池项, 54
 - 概念, 54
 - 窗口化列表, 54
- 数据池项, 407
 - 删除, 184
 - 参考, 285
 - 在 EB GUIDE Monitor 中更改, 211
 - 复制, 180
 - 导入, 200
 - 导出, 199
 - 添加, 179
 - 添加语言支持, 255
 - 粘贴, 180
 - 编辑列表, 180
 - 链接, 183
- 数据类型
 - Function (): bool, 288
 - ibl, 288
 - 列表, 290
 - 参考, 286
 - 图像, 288
 - 字体, 288
 - 字符串, 290
 - 布尔值, 286
 - 布尔值列表, 286
 - 整数, 289
 - 条件脚本, 286
 - 浮点值, 287
 - 网格, 289
 - 网格列表, 290
 - 颜色, 286
- 整数
 - 数据类型, 289

- 文本切断
 - 参考, 359
- 方框布局
 - 参考, 378
- 旋转
 - 参考, 359, 374, 402
- 旋转操作动作
 - 参考, 369
- 显卡程序库, 408
- 显示
 - 配置, 198
- 景深
 - 参考, 399
- 更改动画, 170
 - 参考, 338
- 最佳实践
 - 条件脚本, 227
- 材料
 - PBR GGX 材料, 351, 410
 - PBR Phong 材料, 352, 410
 - 参考, 350, 351, 352
- 条件
 - 添加, 126
- 条件脚本
 - 数据类型, 286
- 栅格布局
 - 参考, 380
- 标签
 - 参考, 347
 - 添加, 134
- 椭圆
 - 参考, 345
- 模型元素, 55, 409
 - 全局重命名, 191
 - 删除, 123
- 模型接口, 55, 68, 409
 - 事件, 68
 - 事群组, 69
 - 创建, 202
 - 删除, 204
 - 命名空间, 69
 - 导入, 203
 - 导出, 202
 - 数据池项, 68
- 模型视图控制器, 409
- 模拟, 193
- 模板, 112
 - 使用, 173
 - 删除, 173
 - 控件模板接口, 112
 - 模板接口, 172
 - 添加, 171
- 模板元素
 - 项目编辑器, 51
- 模板接口
 - 删除属性, 172
 - 添加属性, 172
- 模板索引
 - 参考, 383
- 正常贴图纹理
 - 参考, 390
- 正弦曲线
 - 参考, 343
- 流式布局
 - 参考, 379
- 浮点值
 - 数据类型, 287
- 添加入口动画, 169
- 渲染
 - 伽马校正, 63
- 渲染器
 - 配置, 198
- 滑动操作动作
 - 参考, 364
- 点光
 - 参考, 353
- 状态, 91, 101, 232, 410
 - 使用转化, 123
 - 初始状态, 92
 - 删除, 123
 - 历史状态, 95
 - 浅度历史状态, 95
 - 深度历史状态, 95
 - 添加, 118

- 添加到组合状态, 119
- 添加选择状态, 120
- 组合状态, 91
- 结束状态, 93
- 视图状态, 92
- 进入动作, 121
- 连接, 123
- 退出动作, 122
- 选择状态, 94
- 状态机, 90, 410
 - UML 2.5 标记, 105
 - 与 UML 相比, 105
 - 为转化添加动作, 127
 - 为转化添加条件, 126
 - 使用触发器进行转化, 125
 - 使用转化, 123
 - 内含状态机, 105
 - 删除, 118
 - 动态状态机, 90
 - 添加, 116
 - 添加内部转化, 128
 - 添加动态状态机, 116
 - 状态, 91, 101
 - 触觉状态机, 90
 - 转化, 98, 101
 - 运行, 101
 - 进入动作, 117
 - 退出动作, 117
 - 逻辑状态机, 90
- 环境光
 - 参考, 348
- 环境纹理
 - 参考, 385
- 用户定义焦点
 - 参考, 363
- 用户定义的属性
 - 添加, 150
 - 添加 `Function (): bool`, 151
 - 重命名, 152
- 用户界面, 41, 408
 - 问题元素, 191
- 着色
 - 参考, 362
- 矩形
 - 参考, 348
- 移入
 - 参考, 372
- 移出
 - 参考, 372
- 移过
 - 参考, 373
- 窗口化列表, 55
- 笔划
 - 参考, 362
- 粗糙纹理
 - 参考, 393
- 粘贴
 - 事件, 176
 - 数据池项, 180
- 线性插值整数曲线
 - 添加, 253
- 线性插值曲线
 - 参考, 345
- 线性曲线
 - 参考, 344
- 绝对布局
 - 参考, 377
- 缩放
 - 参考, 402
- 缩放模式
 - 参考, 382
- 网格, 73
 - 参考, 350
 - 数据类型, 289
- 网格列表
 - 数据类型, 290
- 聚光
 - 参考, 354
- 脚本化的值, 87
 - 最佳实践, 227
 - 转换, 181
- 脚本曲线, 40
 - 参考, 343
 - 添加, 274

- 自动焦点
 - 参考, 363
- 自动隐藏, 53
- 色调映射
 - 参考, 397
- 行索引
 - 参考, 383
- 行间距
 - 更改, 159
- 观察列表 (见 EB GUIDE Monitor)
- 视图, 108, 410
 - 参考, 338
 - 添加, 130
- 视图切换动画, 169, 411
 - 添加, 266
- 视图模板
 - 参考, 338, 338
- 视图状态
 - 参考, 338
- 触发器
 - 定义, 125
- 触摸丢失
 - 参考, 374
- 触摸已按下
 - 参考, 375
- 触摸已放开
 - 参考, 376
- 触摸操作动作 (见 操作动作)
- 触摸状态已变化
 - 参考, 376
- 触摸移动
 - 参考, 375
- 触摸输入 (见 操作动作)
- 语言, 164
 - 删除, 165
 - 导入, 66
 - 导出, 66
 - 更改, 194, 255
 - 添加, 164
 - 语言支持, 65
- 语言依赖项文本, 255
 - 导入, 200
 - 导出, 199
- 资源, 70, 410
 - .psd 文件格式, 69
 - 3D 图形, 72, 73
 - Photoshop 文件格式, 69
 - PSD, 69
 - 图像, 72
 - 基于图像的光照, 72
 - 字体, 71
 - 网格, 73
- 资源管理 (见 资源)
- 路径动作
 - 参考, 366, 367
 - 添加, 243
- 转化, 98, 101, 410
 - 使用触发器, 125
 - 添加, 123
 - 添加内部转化, 128
 - 添加动作, 127
 - 添加条件, 126
 - 移动, 124
 - 类型, 100
- 转换
 - 参考, 403
- 边框
 - 参考, 361
- 进入动作 (见 状态)
- 进度条
 - 建模, 279
- 进程间通信, 409
- 退出动作 (见 状态)
- 退出动画, 169
 - 参考, 338
- 选择群组
 - 参考, 358
- 透明涂层
 - 参考, 385
- 通信环境, 407
 - 写入器应用程序, 41
 - 添加, 182
 - 读取器应用程序, 41
- 配置文件, 195, 410

- 克隆, 195
- 添加, 195
- 金属纹理
 - 参考, 389
- 链接
 - 控件属性到控件属性, 147
 - 控件属性到数据池项, 149
 - 数据池项, 183
- 键状态已变化
 - 参考, 371
- 键盘快捷键 (见 键盘快捷键)
 - 用户界面, 336
- 键盘键, 178
- 镜面纹理
 - 参考, 395
- 长按操作动作
 - 参考, 365
- 问题元素
 - 项目编辑器, 51
- 项目
 - 创建, 189
 - 导出, 193
 - 打开, 189
 - 模拟, 193
 - 验证, 191
- 项目中心, 42, 410
 - 内容区域, 43
 - 导航区, 42
- 项目编辑器, 43, 410
 - VTA 元素, 51
 - 内容区域, 47
 - 命令区, 50
 - 导航元素, 44
 - 工具箱, 46
 - 工具箱元素, 45
 - 模板元素, 51
 - 问题元素, 51
- 颜色
 - 数据类型, 286

A

ADAS 电子控制单元, 407

ADASIS, 407

alpha 掩码

- 参考, 339
- 添加, 140

API (见 应用程序编程接口)

E

EB GUIDE arware, 407

EB GUIDE GTF, 408

EB GUIDE GTF SDK, 408

EB GUIDE GTF 扩展, 62, 408

EB GUIDE Monitor, 58

- monitor.cfg, 209
- 元素, 51, 53
- 加载配置, 209
- 启动, 206
- 启动脚本, 216
- 命令行, 206
- 复制数据池项, 211
- 导入观察列表, 217
- 导出观察列表, 217
- 更改语言, 207
- 标签, 51
- 模拟, 193
- 独立, 206
- 脚本示例, 211
- 触发事件, 210
- 连接, 207
- 配置, 207

EB GUIDE Monitor 扩展, 62, 408

EB GUIDE product line, 408

EB GUIDE SDK, 408

EB GUIDE Studio, 408

EB GUIDE Studio SDK; , 408

EB GUIDE Studio 扩展, 62, 408

EB GUIDE TF, 408

EB GUIDE 扩展 (见 扩展)

EB GUIDE 模型, 55, 408

- .gdata, 56
- 使用命令行导出, 194
- 存储格式, 56
- 导出, 193

- 模型元素, 55
- 模拟, 193
- 验证, 57, 58, 191
- EB GUIDE 脚本, 74, 408
 - if-then-else, 80
 - todo 注释, 75
 - While 循环, 79
 - 事件, 84
 - 使用函数, 236
 - 关键字, 291
 - 函数, 293
 - 列表, 84
 - 前缀, 74
 - 参考, 291, 292, 293
 - 右值, 78
 - 外部函数调用, 81
 - 字符串格式设置, 86
 - 局部变量, 78
 - 左值, 78
 - 控件属性, 83
 - 数据池访问, 82
 - 最佳实践, 227
 - 标准库, 86
 - 注释, 75
 - 类型, 75
 - 脚本化的值, 87
 - 表达式, 76
 - 识别符, 74
 - 转换, 181
 - 运算符, 292
- EB GUIDE 项目, 55, 408
- F**
 - Function () : bool
 - 数据类型, 288
- G**
 - GUI (见 用户界面)
- H**
 - HMI 模型, 409
- I**
 - IBL (见 基于图像的光照)
 - ibl
 - 数据类型, 288
 - IBLGenerator, 64, 409
 - IPC, 409
- L**
 - lineGap, 159
 - lineOffset, 159
- P**
 - Photoshop 文件格式, 69
 - 导入, 69
 - 提取, 69
 - PSD (见 Photoshop 文件格式)
- S**
 - shell (见 命令行)
- U**
 - UI (见 用户界面)
 - Unicode 键
 - 参考, 371
- V**
 - Viewport
 - 参考, 384
 - VTA (见 视图切换动画)
 - VTA 元素
 - 项目编辑器, 51