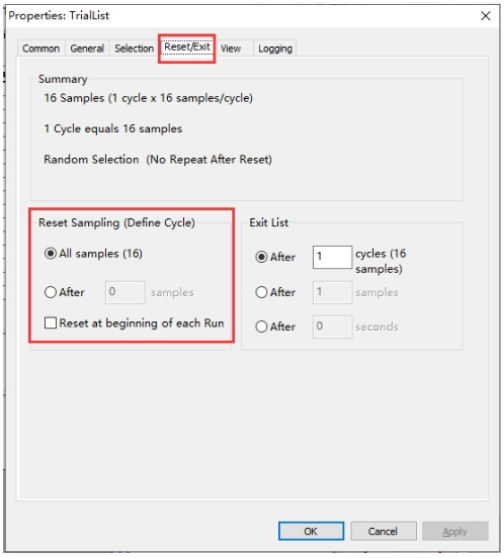
一、E-Prime 实验设计核心步骤

1. **画出实验流程图**：设计实验的逻辑流程。
2. **新建文件夹**：存储实验程序的所有相关文件。
3. **创建指导语**：在总过程（SessionProc）中创建指导语对象（Slide 或 TextDisplay）。
4. **使用 List 对象生成核心流程**：用 List 对象来指定和生成流程图中的核心实验试次序列。
5. **在 List 中输入刺激材料与属性**：在 List 对象中输入每个试次所需的刺激材料及其相关属性（如刺激词、颜色、正确反应键等）。
6. **创建核心实验过程**：在核心实验 Procedure 中创建各个组成部分（如注视点、刺激呈现屏、反馈屏），并设置呈现时间、反应方式、需记录的数据等。
7. **创建结束语**：在总过程（SessionProc）中创建实验结束语。
8. **运行、调试和修改程序**：反复运行调试，直至程序符合实验要求。
9. **区分练习与正式模块**：设置练习模块，并可设定标准（如正确率达 80%）自动进入正式实验。
10. **编译生成脚本文件**：最终编译程序，生成可独立运行的脚本文件（.es3）。

二、E-Prime 核心对象与功能

1. **Procedure（流程对象）**：用于组织实验试次的流程结构。
2. **TextDisplay（文本对象）**：用于呈现文本刺激。
3. **Slide 对象**：可整合文本、图片等多种刺激的呈现对象。
4. **ImageDisplay 对象**：用于呈现图片刺激。
5. **SoundOut 对象**：用于播放声音刺激。
    - **属性**：包括文件名（支持 WAV、MP3、WMA 格式）、缓冲区大小、播放模式、起始/停止偏移、是否循环等。
6. **List（列表对象）**：核心的实验设计工具，用于定义试次序列、刺激属性和水平。
    - **属性调用（引用）**：在对象属性中，使用 [属性名] 的格式调用 List 中定义的变量。
    - **嵌套（Nest）**：当需要随机组合不同类别的变量时（如数字和字母），可以使用嵌套 List。
    - **重新取样（Reset/Exit）**：可设置 List 的抽样方式（如随机、不重复）、循环次数等。



7. **FeedbackDisplay 对象**：用于向被试提供反馈（如正确/错误）。
    - **属性修改**：可以修改其文本内容等属性。
8. **InLine 对象**：用于插入 E-Basic 脚本代码，实现复杂逻辑控制。
    - **功能**：可设置变量、进行条件判断、控制流程跳转等。

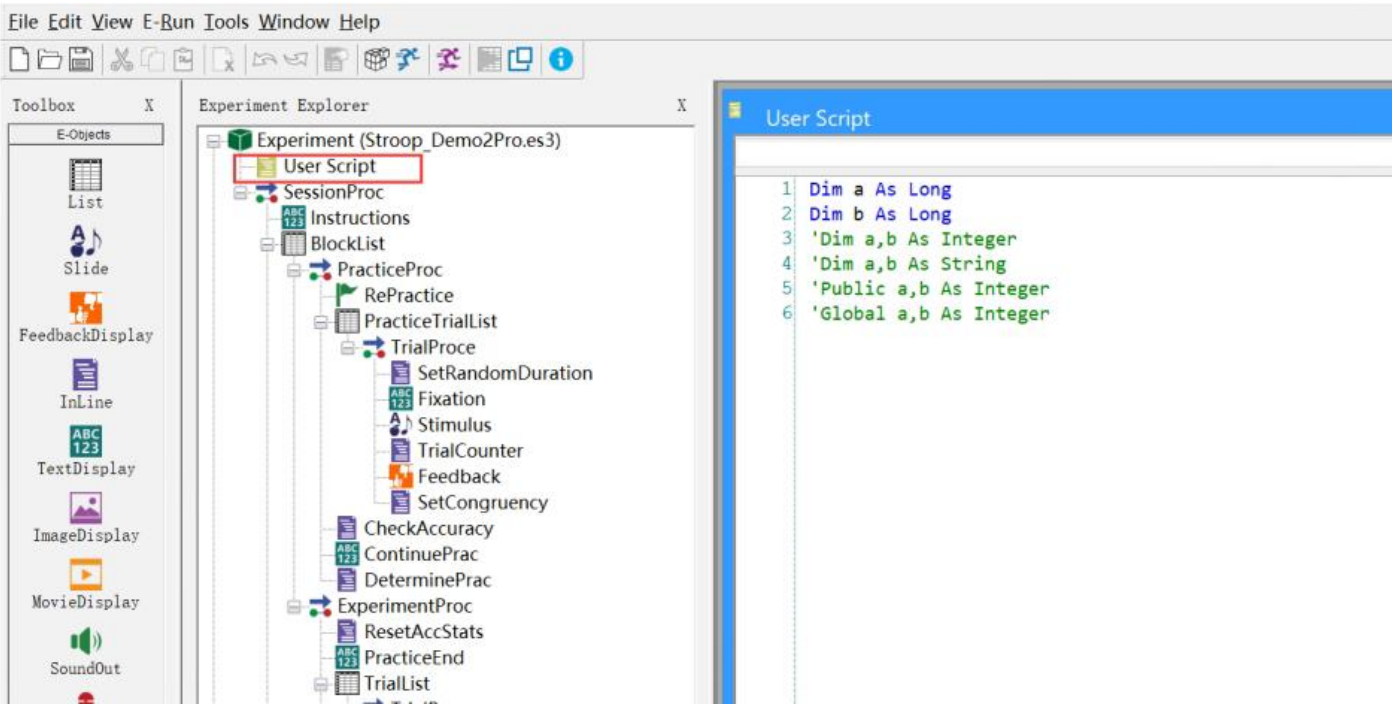
三、数据记录与收集

1. **数据记录设置**：在对象的属性中，将 Data Logging 设置为 Standard 以记录标准数据。
2. **关键记录字段**：
    - OnsetTime：对象启动的时间点（毫秒）。
    - ACC：记录按键反应是否正确（1/0）。
    - CRESP：正确的反应键。
    - RESP：被试实际按下的键。
    - RT：反应时（毫秒）。
    - RTTime：自实验开始到做出反应所经过的时间（毫秒）。
3. **启动信息收集（Startup Info）**：在实验开始时弹窗收集被试信息（如编号、姓名、年龄、性别等）。
4. **数据处理流程**：运行实验（E-Run）生成数据文件（.edat2 或.edat3），可使用 E-Merge 合并数据，再用 E-DataAid 进行分析。

四、变量、条件与流程控制

1. **变量定义**：
    - **原则**：先定义，后使用。格式如：Dim/Global/Public 变量名 As 变量类型。
    - **数据类型**：Integer（整型）、String（字符型）、Single（单精度小数）等。

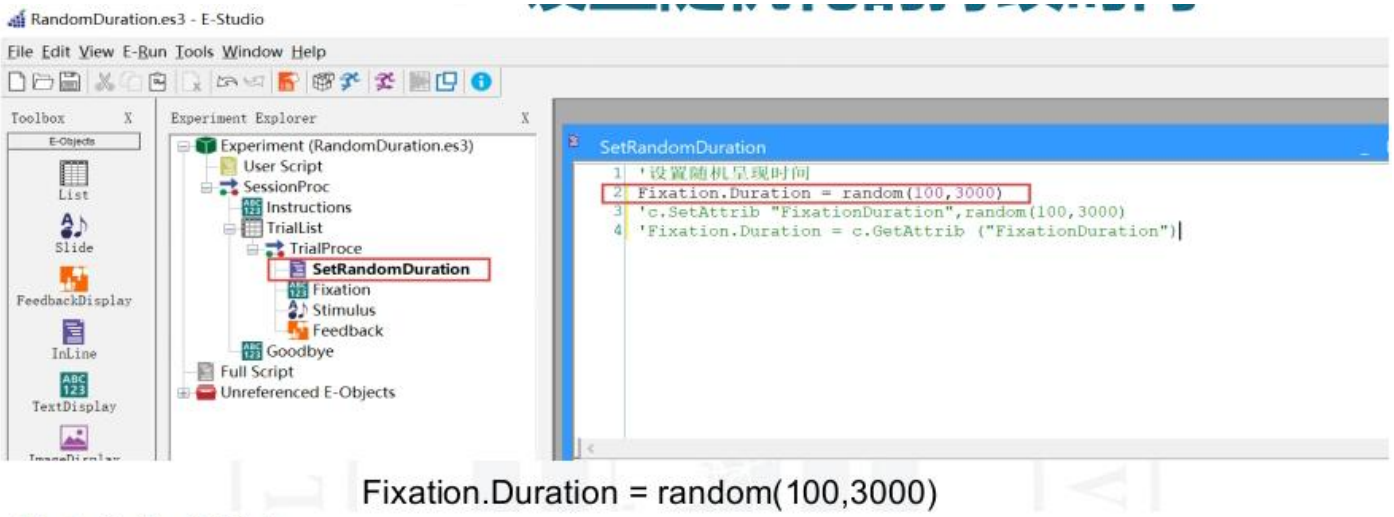
- **全局变量**：在 User Script 中定义，作用范围是整个实验。



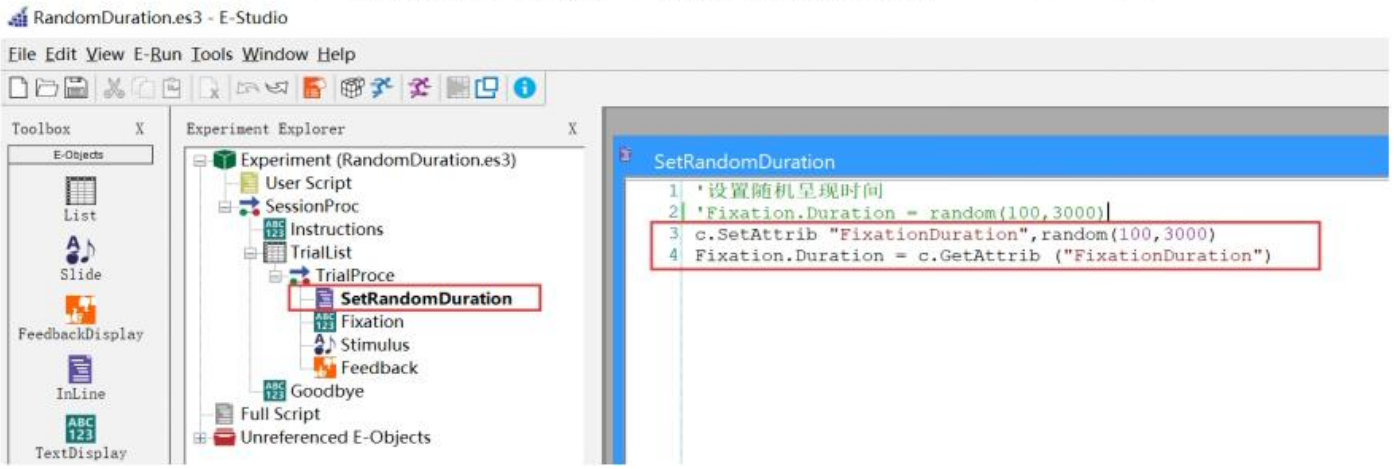
- **局部变量**：在某个 Procedure 的 InLine 中定义，仅在该 Procedure 内有效。

2. **属性赋值与调用**：

- 设置属性：c.SetAttrib “属性名”，值。



Fixation.Duration = random(100,3000)

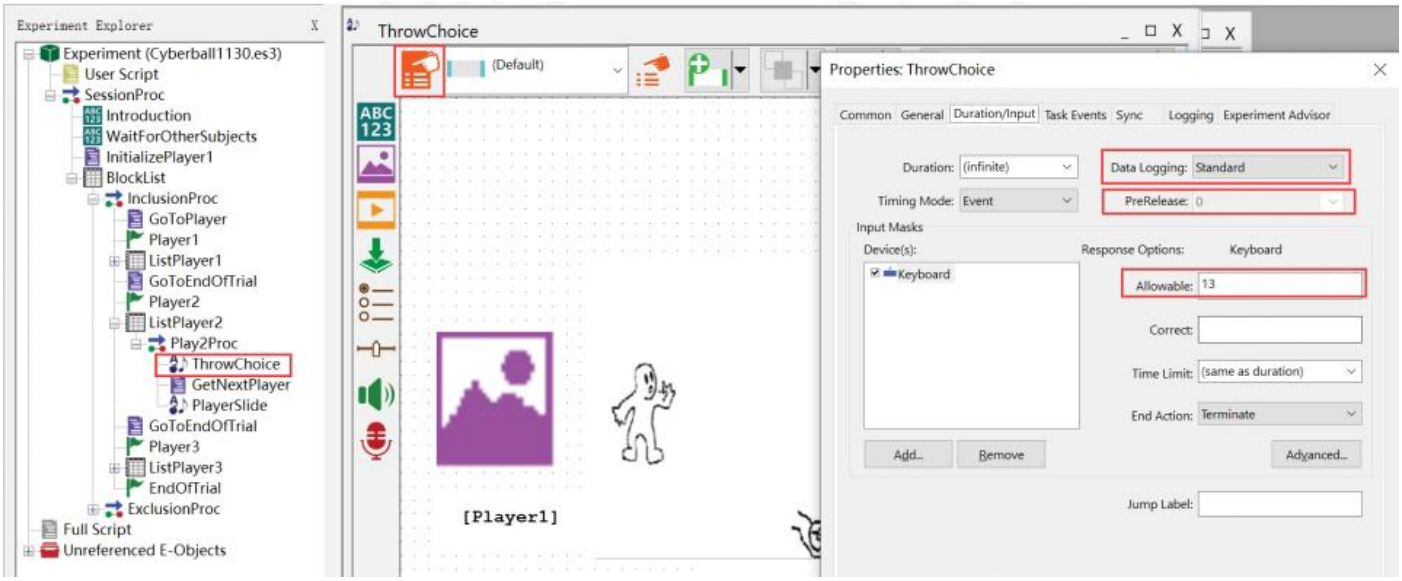


- 获取属性：变量 = c.GetAttrib(“属性名”)。

3. **随机化**：
    - 使用 random()函数，例如：Fixation.Duration = random(100, 3000) 或 c.SetAttrib “FixationDuration”, random(100,3000)。
4. **条件判断与流程控制**：
    - 使用 If...Then...ElseIf...Else...End If 结构进行条件判断。
    - 使用 GoTo Label 实现流程跳转。
    - 使用 Mod 函数实现周期性操作（如每 N 个试次休息一次）。
5. **根据反应给予不同反馈**：
    - 在 InLine 中判断 Stimulus.ACC（正确性）和 Stimulus.RT（反应时），然后通过 c.SetAttrib 设置一个属性（如”whichState”），在 Slide 对象中根据该属性的不同值激活不同的 State（如 Correct, Wrong, tooFast, tooSlow），以呈现对应的反馈信息。

五、特定实验范式的编程要点

1. **Stroop 实验**：
    - **因素与水平**：词（Word）、颜色（Color）、一致性（Congruent/Incongruent）。
    - **记录一致性**：在 List 中直接定义 Condition 属性，或在 InLine 中用脚本判断词与颜色是否匹配并赋值。
2. **整体-局部范式（Navon 任务）**：
    - 使用嵌套 List 来组合整体字母和局部字母。
    - 通过不同的 Procedure（如 LocalProc, GlobalProc）和指导语来控制被试判断整体还是局部。
3. **社会排斥范式（Cyberball 网络传球任务）**：
    - **核心结构**：包含接纳程序（InclusionProc）和排斥程序（ExclusionProc）。
    - **流程控制**：使用 InLine 和 Label，根据变量 CurrentPlayer（当前持球者）的值，跳转到对应玩家（Player1, Player2, Player3）的传球流程。
    - **视频刺激**：使用 SlideMovie 对象播放传球动画（如 1to2.wmv）。
    - **被试交互**：玩家 2（真被试）在 ThrowChoice 屏通过按键（如 1 或 3）选择传球对象。



- **传球比例**：通过设置 List 中不同传球目标试次的权重，控制接纳（如 70%传给被试）和排斥（如 10%传给被试）条件。

六、其他重要功能与技巧

1. **PreRelease（后台预加载）**：可设置为 0，以确保前一个对象结束后立即呈现下一个对象，减少加载延迟。
2. **练习模块自动化**：
   - 在练习模块末尾，通过 InLine 计算正确率（如 b/a）。
   - 使用条件判断（If b/a >= 0.8 Then），若达到标准则显示信息并进入正式实验；若未达到则提示重新练习。
3. **编译与运行**：最终通过 E-Studio 编译程序，生成可在 E-Run 中执行的脚本。

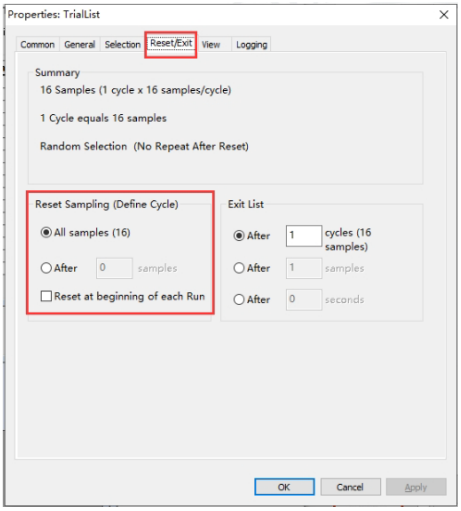
List 对象

一、List 对象的知识点

List 对象是 E-Prime 中用于定义和控制实验试次序列的核心工具。

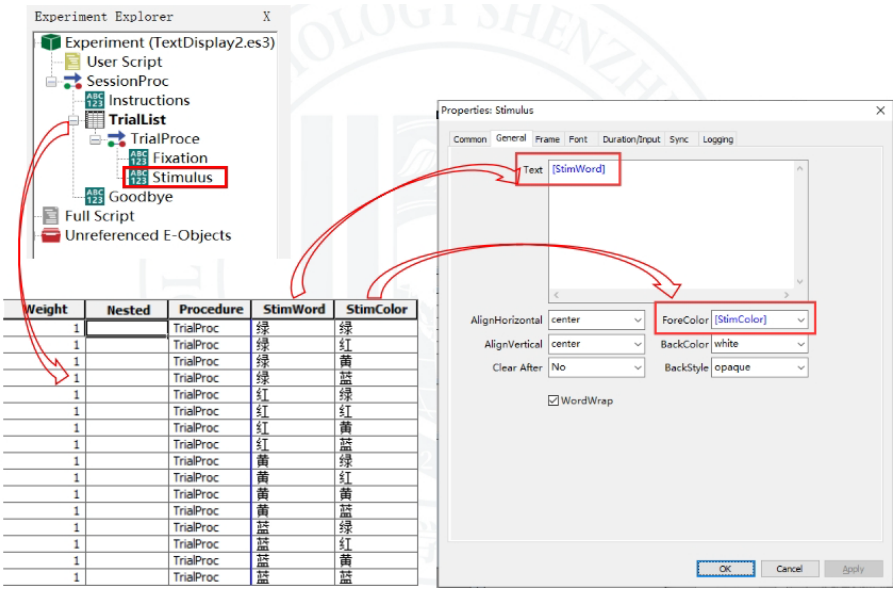
1. **核心功能**：
   - 用于**指定和生成**流程图中的核心实验过程。
   - 在 List 中输入**刺激材料及其相关属性**（例如，在 Stroop 实验中，输入 StimWord、StimColor、CorrectResp 等）。
2. **关键属性与设置**：
   - **呈现顺序（Order/Selection）**：可以设置试次的呈现顺序，例如**顺序呈现（Sequential）**或**随机呈现（Random）**。

- **重新取样（Reset/Exit）**：用于控制 List 中试次的抽样规则。可以设置“**Reset at beginning of each Run**”或在完成一定样本量后重置，以避免长序列中某些刺激出现次数过多或过少的问题。



- **权重（Weight）**：可以设置每个试次（行）的权重，以控制其出现的相对次数。
- **嵌套（Nest）**：当实验需要随机组合**不同类别（种类）的变量**时，可以使用嵌套 List。例如，在一个任务中，需要将数字和字母两类刺激随机组合呈现，就可以分别创建数字 List 和字母 List，然后将一个嵌套到另一个之中。

3. **属性调用（引用）**：
   - 在实验过程中的其他对象（如 TextDisplay、Slide）的属性设置中，可以通过 **[属性名]** 的格式来调用 List 中定义的变量值。例如，将 TextDisplay 对象的 Text 属性设置为 [StimWord]，将 ForeColor 属性设置为 [StimColor]。



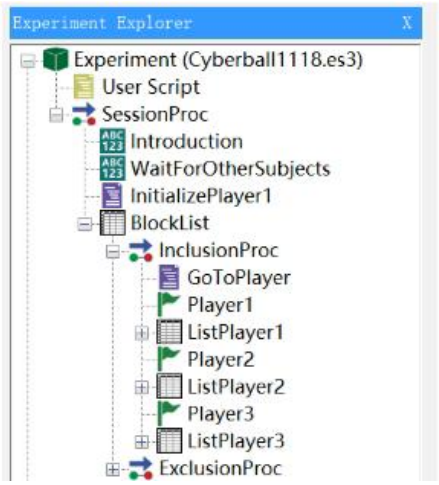
4. **复制 List**：
   - 文档提到了复制 List 时的两种选项，这关系到对象之间的关联性：
     - 选择“**是**”：独立复制 List 及其子对象，修改其中一个不会影响另一个。

- 选择“**否**”：独立复制 List，但关联复制其中的子对象，修改一个 List 中的子对象会影响另一个。

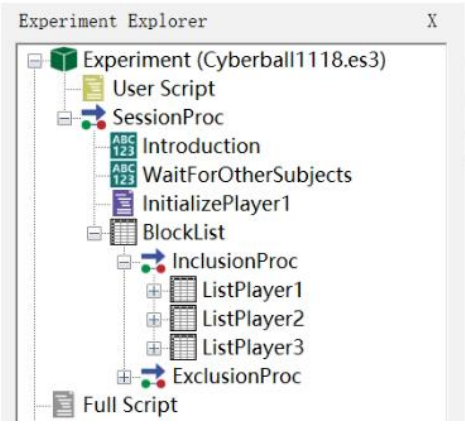
二、Label 的知识点

Label 在文档中主要出现在 **Cyberball（网络传球任务）** 的实验设计示例中，用于**流程控制**。

1. **主要用途**：与 **Inline 对象**结合使用，实现**条件跳转**，控制实验流程的执行路径。
2. **应用实例（Cyberball 任务）**：
   - 在任务中，需要根据变量 CurrentPlayer（当前持球者）的值，决定程序跳转到哪个传球流程。
   - 在 Inline 对象中编写条件判断语句（如 If...Then...ElseIf...），根据 CurrentPlayer 的值，使用 **GoTo** 语句跳转到对应的 Label（例如 Player1, Player2, Player3）。



   - 每个 Label（如 Player1）后面会跟着一个对应的 List 对象（如 ListPlayer1），用于播放该玩家传球的视频。



   - 在每次播放完一个传球视频（即完成一次传球试次）后，会通过 Inline 中的 GoTo 语句跳转到一个名为 **EndOfTrial** 的 Label，从而结束当前试次，进入列表中的下一个试次。

Inline 对象语句的编写方法

一、Inline 语句的核心功能

Inline 对象用于插入 E-Basic 脚本代码，以实现程序无法通过图形界面直接设置的复杂逻辑和控制。其主要功能包括：

1. **设置变量与属性**：定义变量，或为对象属性动态赋值。
2. **条件判断**：根据特定条件（如反应正确与否、反应快慢、试次编号等）执行不同的操作。
3. **流程控制**：控制程序执行的流程，例如跳转到指定的 Label。
4. **进行计算**：执行算术运算、逻辑判断等。

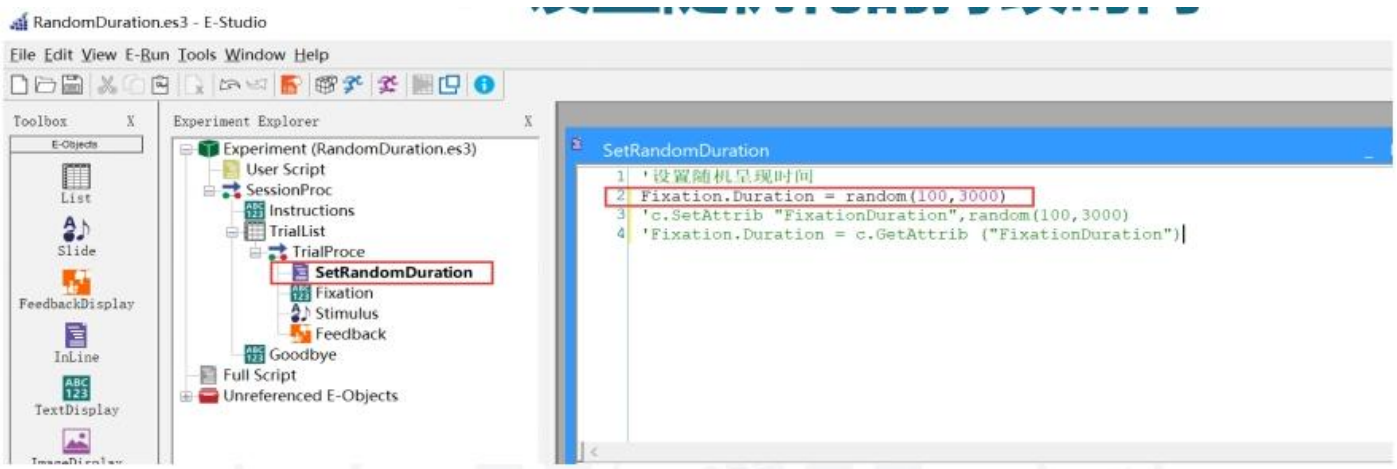
二、Inline 语句的编写方法与示例

文档中展示了以下几种核心的编写方法：

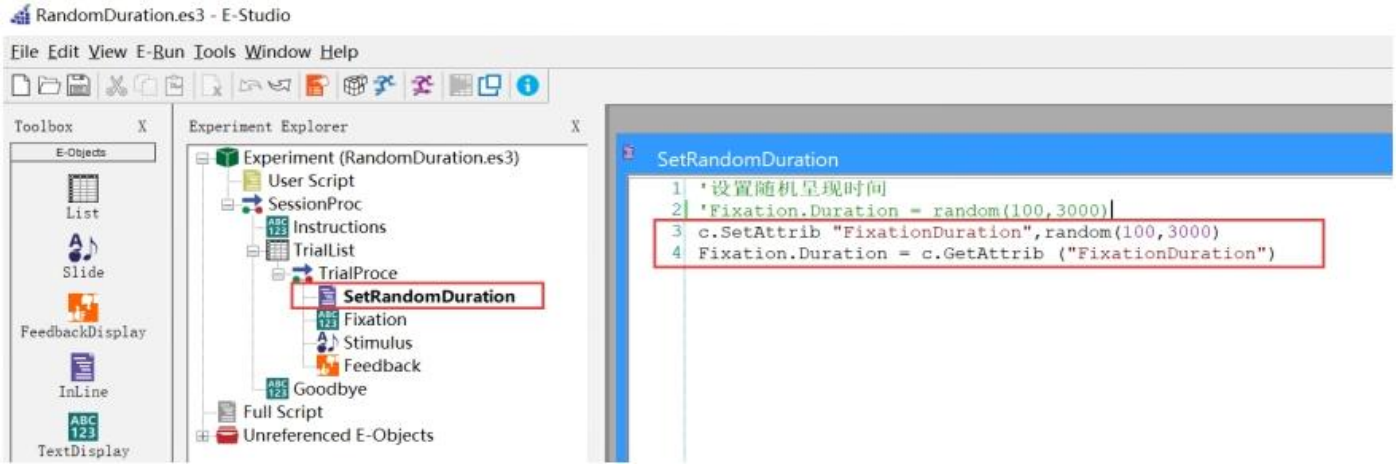
1. 设置随机化的持续时间

用于在试次中动态生成一个随机时间。

- **方法一：直接为对象属性赋值**
- Fixation.Duration = random(100,3000)
    - **说明**：将注视点（Fixation）的呈现时间设置为在 100 毫秒到 3000 毫秒之间随机取值。



Fixation.Duration = random(100,3000)



- **方法二：通过 c 对象设置并获取 List 属性**
- c.SetAttrib “FixationDuration”, random(100,3000)
- Fixation.Duration = c.GetAttrib (“FixationDuration”)
    - **说明**：首先生成一个随机数，并将其赋值给当前试次（c 代表当前 List 行）的一个自定义属性“FixationDuration”。然后，从该属性中读取这个值，并赋给 Fixation 对象的 Duration 属性。这种方法允许在实验的其他地方通过[FixationDuration]来引用这个随机值。

2. 条件判断与流程控制（使用 If…Then… 和 GoTo）

这是实现复杂实验逻辑的关键，文档中在 **Stroop 实验**和 **Cyberball 任务**中均有应用。

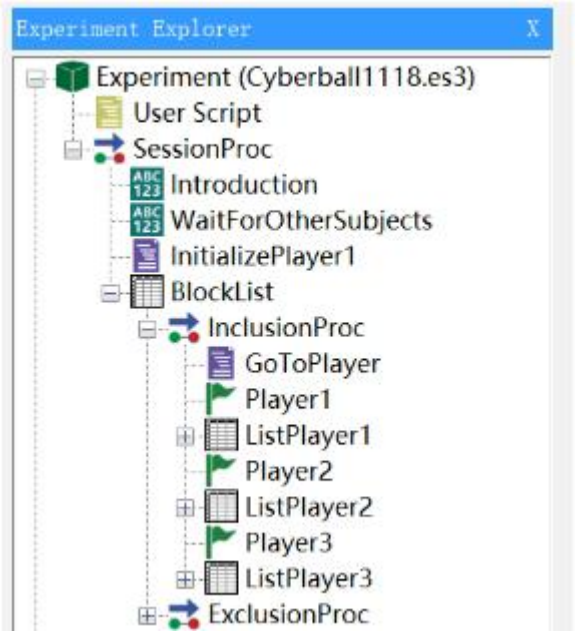
基本条件判断结构:

- If 条件 1 Then
- 	‘执行操作 1
- ElseIf 条件 2 Then
- 	‘执行操作 2
- Else
- 	‘执行其他操作
- End If

示例：根据变量值跳转到不同流程（Cyberball 任务）

- If CurrentPlayer = 1 Then
- 	GoTo Player1
- ElseIf CurrentPlayer = 2 Then
- 	GoTo Player2
- ElseIf CurrentPlayer = 3 Then
- 	GoTo Player3
- End If

说明：根据变量 CurrentPlayer（当前持球者）的值，决定程序流程跳转到哪个对应的 Label（Player1, Player2, Player3），以播放不同玩家的传球视频。



示例：计算正确率并判断（练习模块）

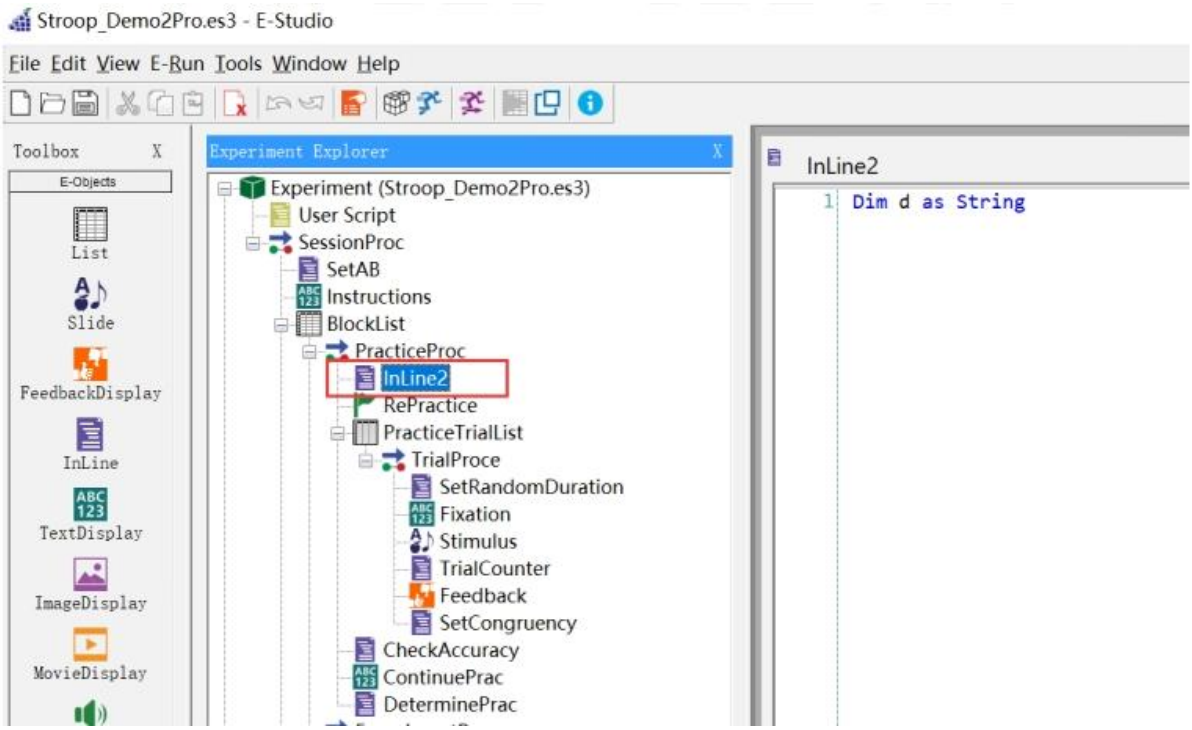
- a = a + 1
- If Stimulus.ACC = 1 Then
- b = b + 1
- End If
- '...（后续代码可能判断 b/a 是否达到 0.8，以决定是否结束练习）
    - **说明**：a 累计总试次数，b 累计正确反应试次数。通过判断刺激对象（Stimulus）的 ACC 属性是否为 1（正确），来更新 b。

示例：周期性操作（每 N 个试次后休息）

- '假设在某个 Inline 中，`trialNum`记录了当前是第几个试次
- If trialNum Mod 30 = 0 Then
- '执行休息操作，例如显示休息提示屏
- End If
    - **说明**：使用 Mod（求余）函数。如果 trialNum 除以 30 的余数为 0，意味着每完成 30 个试次就触发一次休息。

3. 定义局部变量

- **方法**：在某个 Procedure 内的 Inline 对象中，使用 Dim 语句定义。

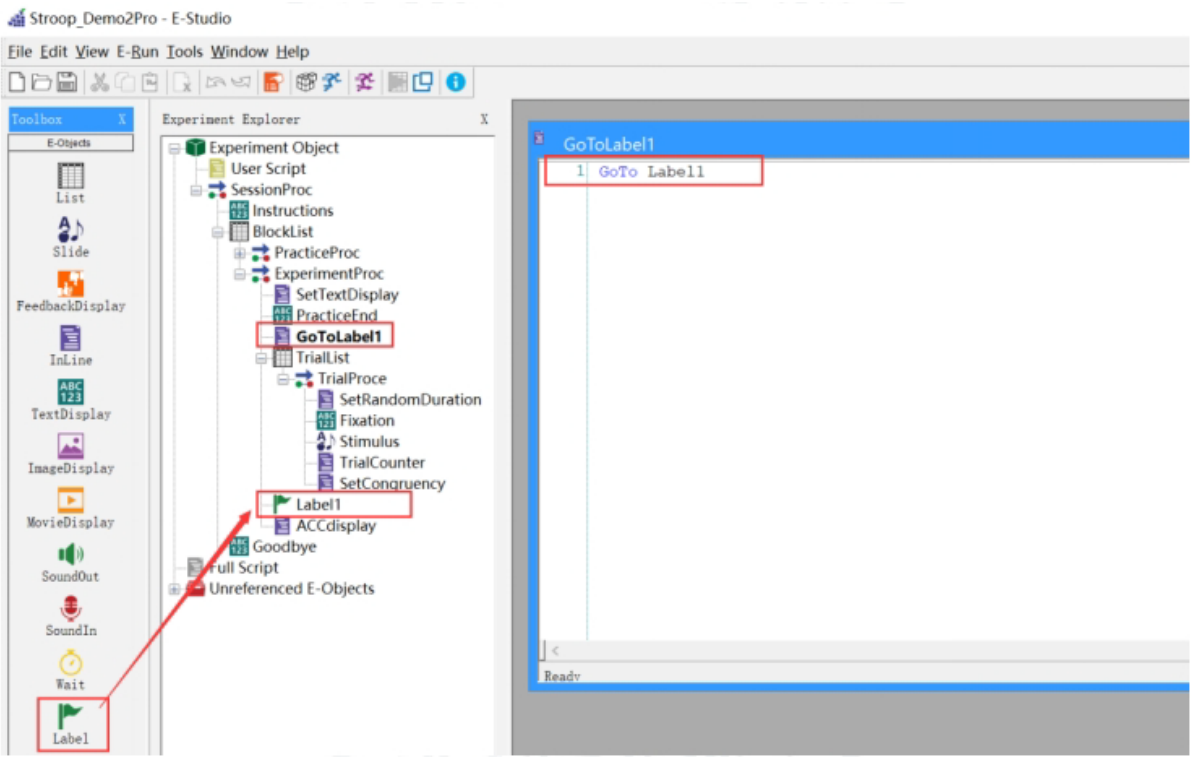


Dim d as String

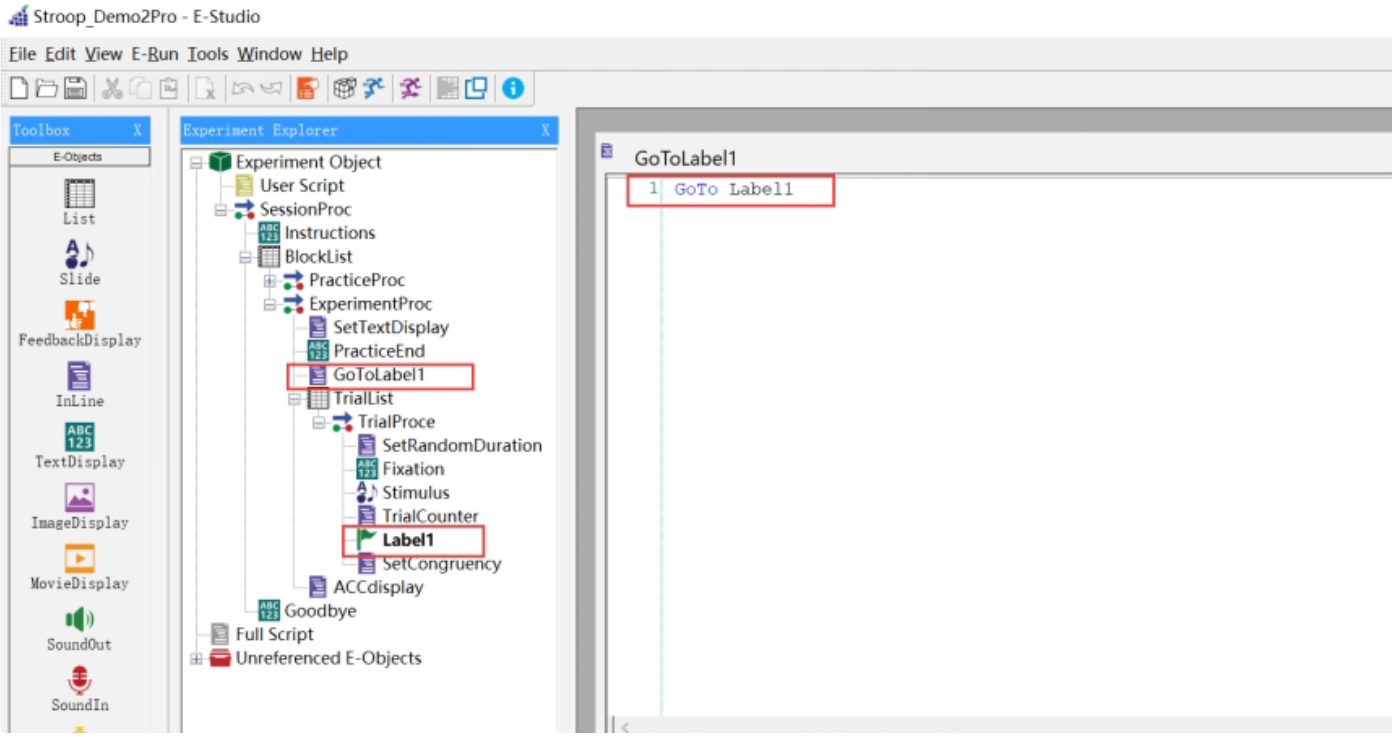
- **说明**：定义了一个名为 d 的字符串（String）类型变量。**文档明确指出**，此类局部变量只能在其被定义的 Procedure 中使用，该 Procedure 结束后变量将被丢弃，不能用于其他 Procedure。

4. 流程跳转的注意事项

Inline 中的 GoTo 语句只能跳转到与其处于同一个 Procedure 内的 Label。



如果 Label 位于另一个 Procedure 中，则无法直接跳转。



三、Inline 对象的重要关联设置

在提供反馈的流程中，文档提到一个关键设置：

- 为了保证反馈能立即呈现，在 FeedbackDisplay 对象**之前**的那个对象（通常是刺激呈现屏）的属性

中，应将其 **PreRelease** 设置为 **0**。这可以避免因预加载造成的延迟。

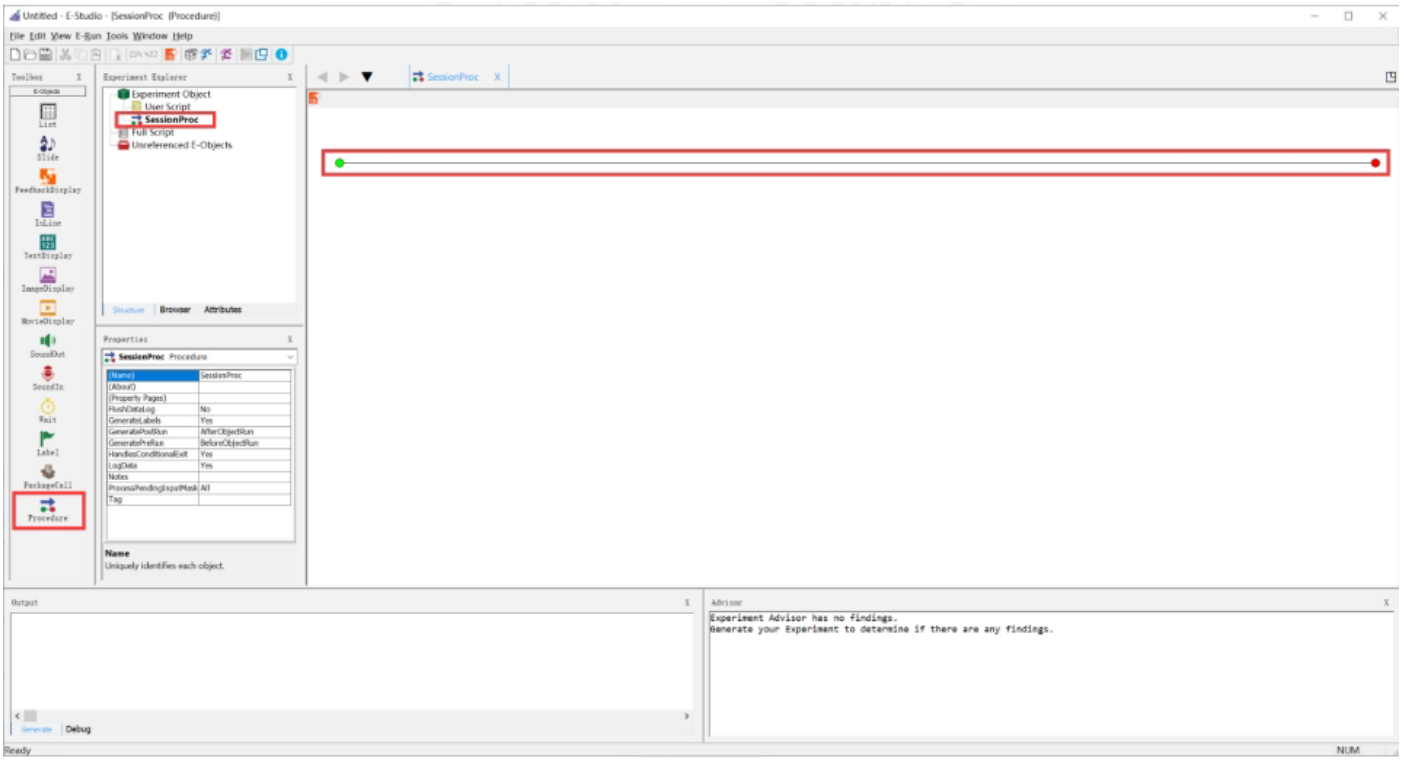
E-Prime 中插入图片

1. 准备图片文件

确保您的图片文件是 E-Prime 支持的格式。支持的格式包括：**.bmp、.jpg、.jpeg、.gif、.png 等**。

2. 使用 ImageDisplay 对象插入图片

在 E-Studio 的“工具箱”（Toolbox）中，找到并拖拽 **ImageDisplay 对象** 到实验流程（Procedure）中。



3. 设置图片属性

选中刚放入的 ImageDisplay 对象，在其属性窗口中进行关键设置：

- **文件名（Filename）**：点击右侧的“...”按钮，选择您已准备好的图片文件。
- **拉伸（Stretch）**：这个属性控制图片如何适应显示框。
    - 如果选择 **“No”**，则只调整图片框（第二层）的大小，图片本身保持原始尺寸。您需要手动调整“Size”中的 Width（宽度）和 Height（高度）百分比来适配。
    - 如果选择 **“Yes”**，图片会自动拉伸以填满整个图片框。您可以选择“Stretch Mode”（拉伸模式），例如“Both”（同时调整宽高）。
- **大小与位置（Size & Position）**：在“Size”和“Position”选项卡中，可以精确设置图片框在屏幕上的大小（用百分比或像素）和位置（如 X: center, Y: center）。

4. 调整显示层级

文档指出，像 ImageDisplay 这类对象具有**三层显示结构**：

- **最底层**：整个屏幕背景。
- **第二层**："内容框"（即图片框），您在上面步骤中调整的大小就是这一层。
- **最上层**：具体的图片内容本身。 通过调整第二层"框"的大小和上层的"拉伸"属性，可以控制图片最终的显示效果。

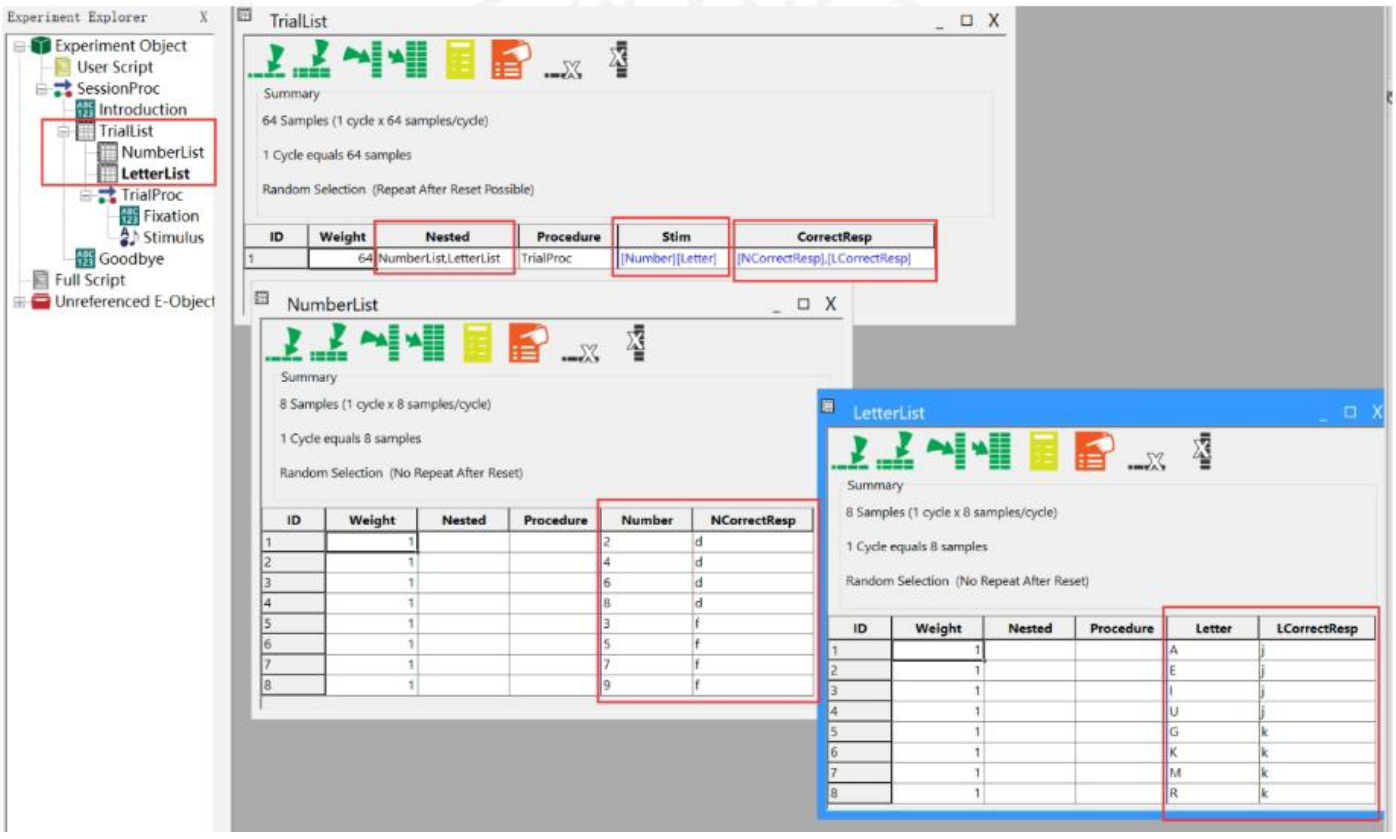
总结：插入自定义图片的核心是使用 **ImageDisplay 对象**，并在其属性中指定 **Filename**，然后通过 **Stretch** 和 **Size** 属性调整其显示方式。请确保图片格式正确，且文件路径能被 E-Prime 程序访问。

嵌套 List

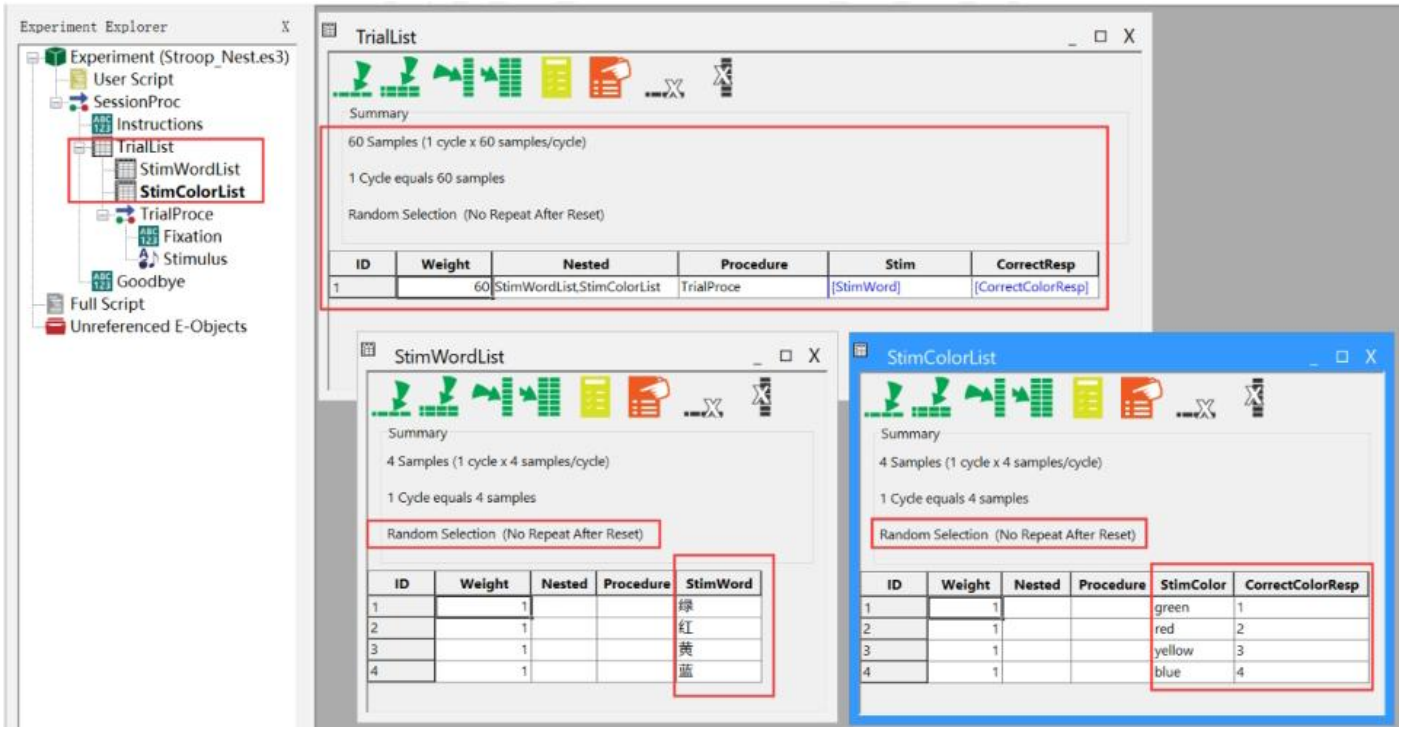
一、核心用途

当实验的每个试次都需要**随机组合来自不同类别（或种类）的变量**时，使用嵌套 List。

- **典型场景**：在"数字-字母任务"中，每个试次需要随机呈现一个数字**和**一个字母。数字和字母分别属于不同的变量类别。
- **例子**：
    1. **数字-字母任务**：需要将 NumberList（数字列表）和 LetterList（字母列表）嵌套。



2. **Stroop 任务**：需要将 StimWordList（词义列表）和 StimColorList（颜色列表）嵌套，以实现词义和颜色的随机组合。



二、操作步骤

1. **创建子 List**：为每一类变量创建一个独立的 List。
    - 例如，创建 NumberList，其中包含 Number（数字）和 NCorrectResp（数字对应的正确反应键）等属性列。
    - 创建 LetterList，其中包含 Letter（字母）和 LCorrectResp（字母对应的正确反应键）等属性列。
2. **创建主 List 并设置嵌套**：创建一个用于运行实验流程的主 List（例如 TrialList）。
    - 在主 List 的属性窗口中，找到 **“Nested”** 列。
    - 点击该列的单元格，通过弹出的对话框，将需要随机组合的**子 List（如 NumberList 和 LetterList）添加进来**。多个子 List 之间用逗号分隔。
3. **在实验过程中引用属性**：在实验的 Procedure 中（例如刺激呈现对象），通过 **[属性名]** 的格式调用来自不同子 List 的属性。
    - 例如，在 TextDisplay 对象的 Text 属性中，可以设置为 [Number][Letter]，以同时显示来自两个 List 的变量。
    - 在需要设置正确反应键的地方，可以设置为 [NCorrectResp],[LCorrectResp]。

三、嵌套效果

- 主 List（TrialList）运行时，会从每个被嵌套的子 List 中**各随机抽取一行**，组合成一个试次。
- 文档中数字-字母任务的例子显示，TrialList 的样本量（Samples）为 64，这正好是 NumberList（8 个样本）和 LetterList（8 个样本）所有可能组合的数量（8 x 8 = 64），确保了所有配对都能出现。

总结：嵌套 List 的核心作用是**高效地实现多类别变量的完全随机化组合**。使用方法为：分别创建各类变量的

子 List，在主 List 的“Nested”属性中嵌入这些子 List，然后在实验对象中通过[属性名]调用即可。

Stroop 实验详细操作

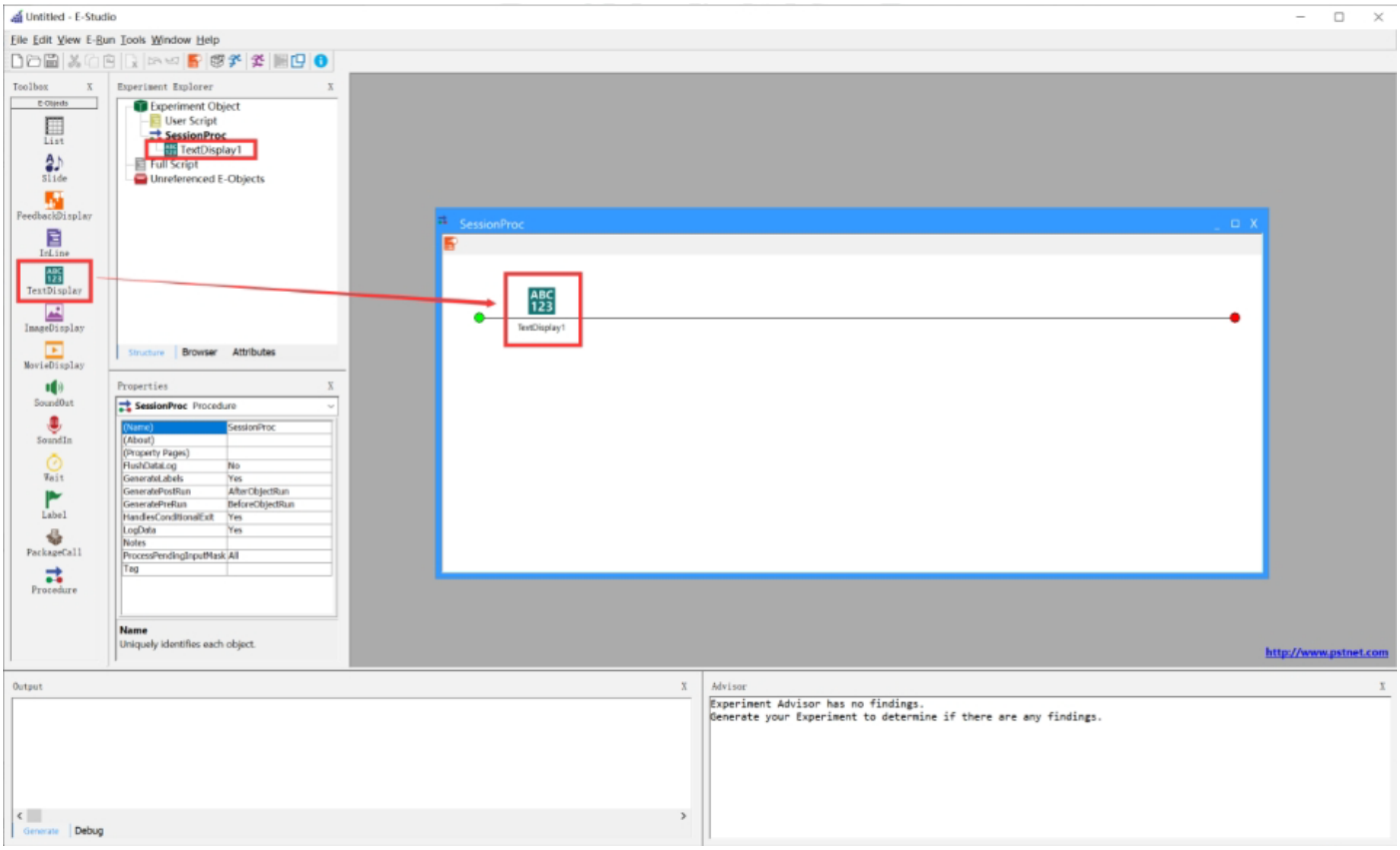
一、实验结构与核心设计

Stroop 实验的核心是呈现颜色与词义不一致的文字（如用红色显示“绿”字），要求被试忽略词义，只对字的颜色进行按键反应。实验通常包含**指导语、练习阶段和正式实验阶段**。练习阶段要求正确率达到一定标准（如 80%）方可进入正式实验。

二、详细操作步骤

步骤 1：创建实验结构

1. **创建 SessionProc**：在 Experiment Explorer 中，这是实验的总流程。



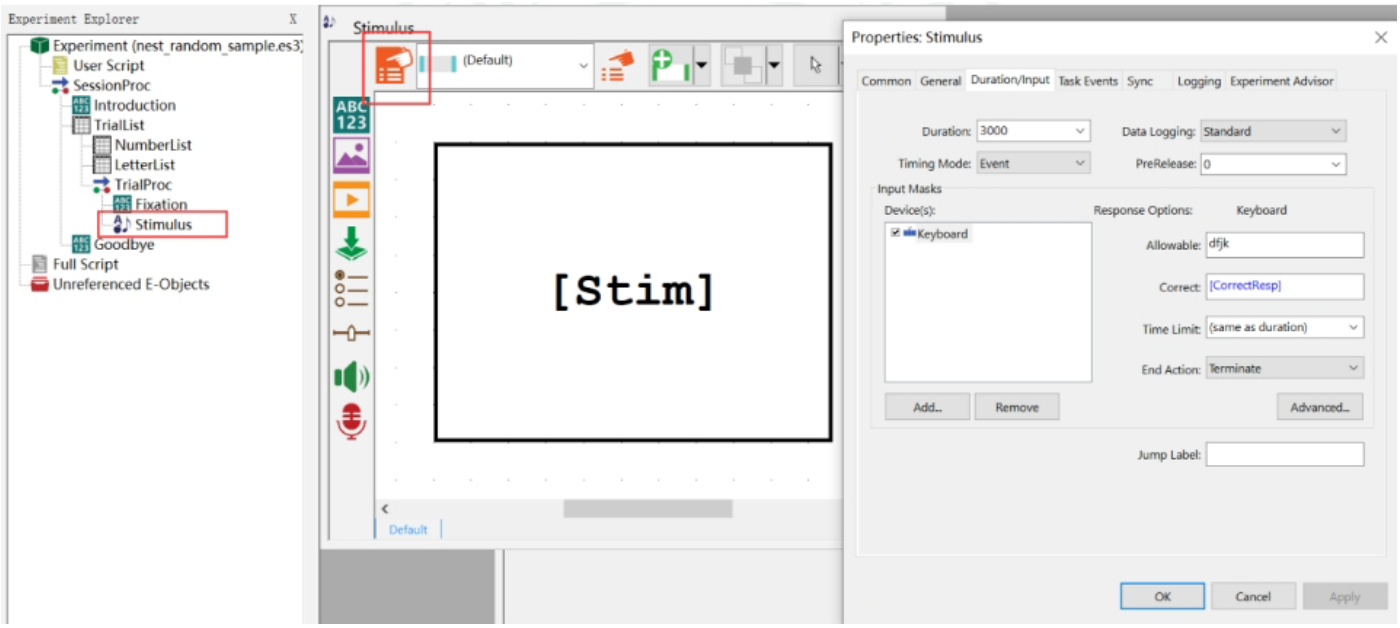
2. **添加指导语**：在 SessionProc 中插入一个 TextDisplay 或 Slide 对象，输入实验指导语（例如：“如果字是绿色，请按 1 键…”）。
3. **创建练习和正式实验流程**：
    - 创建 PracticeProc（练习流程）和 ExperimentProc（正式实验流程）。
    - 在每个 Procedure 中，构建试次序列：通常包括 Fixation（注视点）、Stimulus（刺激呈现屏）和 Feedback（反馈屏，**仅练习阶段需要**）。

步骤 2：创建并设置 List 对象（定义实验材料）

1. **创建材料列表**：为练习和正式实验分别创建 List 对象（如 PracticeTrialList 和 TrialList）。
2. **定义属性列**：在 List 中，至少需要定义以下属性列：
    - StimWord：刺激词（如：红、绿、黄、蓝）。
    - StimColor：词的颜色（如：red, green, yellow, blue）。
    - CorrectResp：该试次对应的正确反应键（如：当 StimColor 为“red”时，CorrectResp 为“2”）。
3. **填入材料**：在 List 的各行中填入具体的词、颜色和正确按键。
4. **设置运行参数**：在 List 的 Properties 中，设置 Selection（如：Random，随机）、Samples（试次数）和 Reset/Exit 规则。

步骤 3：在刺激对象中引用 List 属性

1. 在 Stimulus（一个 TextDisplay 对象）的属性窗口中：
    - **Text 属性**：设置为 [StimWord]。这将显示 List 中当前行的词。
    - **ForeColor 属性**：设置为 [StimColor]。这将把词的颜色设置为 List 中当前行定义的颜色。
    - **Duration/Input 设置**：在 Input Masks 中添加键盘反应，并将 Correct 设置为 [CorrectResp]，以便系统自动判断对错。



步骤 4：实现练习阶段的自动判断与跳转（使用 Inline 对象）

这是实现“正确率达 80%进入正式实验”逻辑的关键。

1. **在练习流程末尾添加 Inline 对象**：例如，在 PracticeProc 中，Feedback 对象之后添加一个名为 CheckAccuracy 的 Inline。
2. **编写条件判断脚本**：在该 Inline 中写入 E-Basic 代码。
3. If Feedback.AccStats.Mean >= 0.8 Then
4. 　　‘正确率达标，进入正式实验

5. ‘可以在此设置提示文本，然后跳转到正式实验流程

6. ‘例如：GoTo Label_StartExp

7. Else

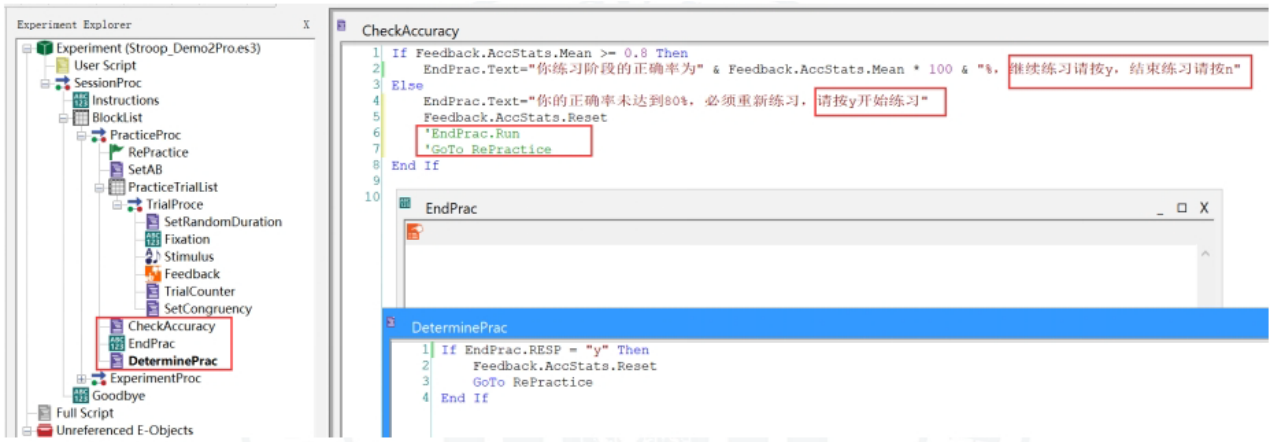
8. ‘正确率未达标，重置统计数据并返回练习开始

9. Feedback.AccStats.Reset

10. GoTo Label_RePractice

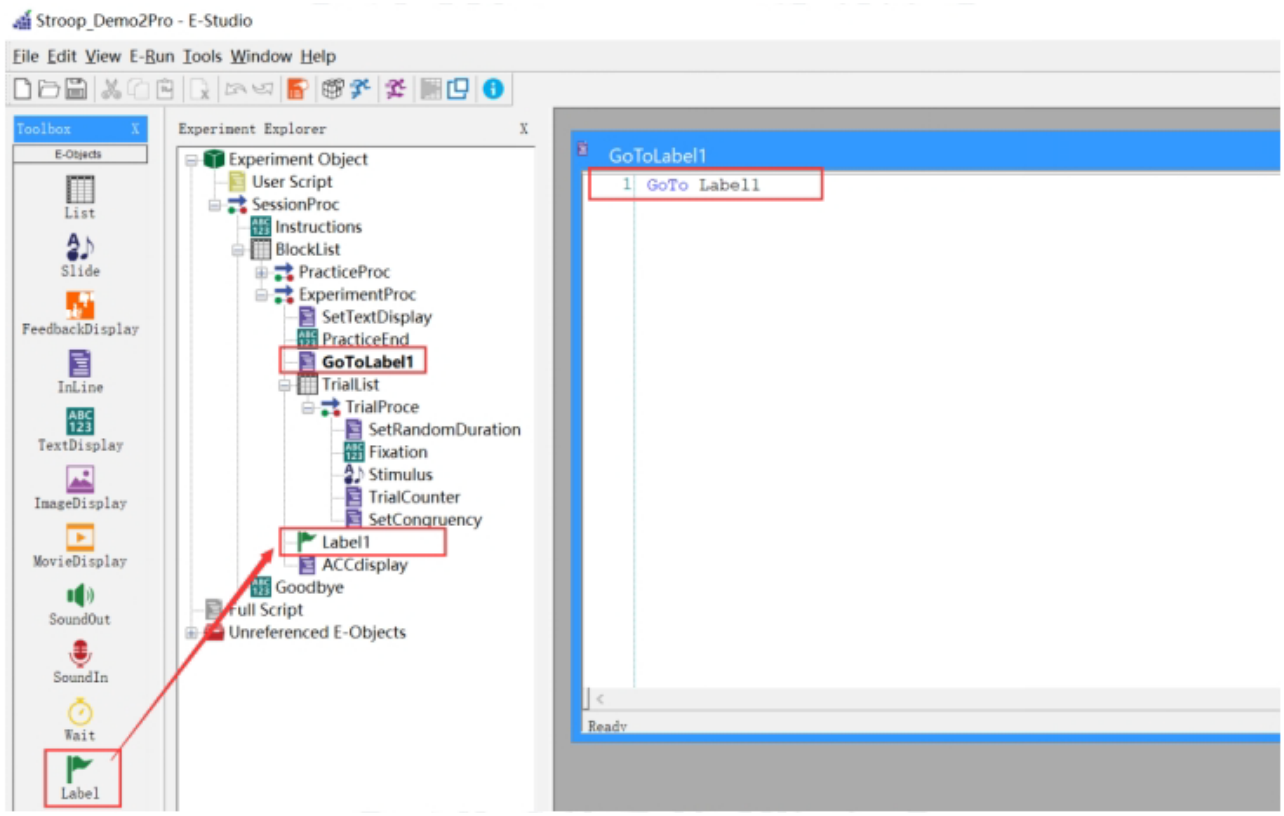
11. End If

- Feedback.AccStats.Mean 可获取当前练习阶段的平均正确率。




```
If Feedback.AccStats.Mean >= 0.8 Then
```

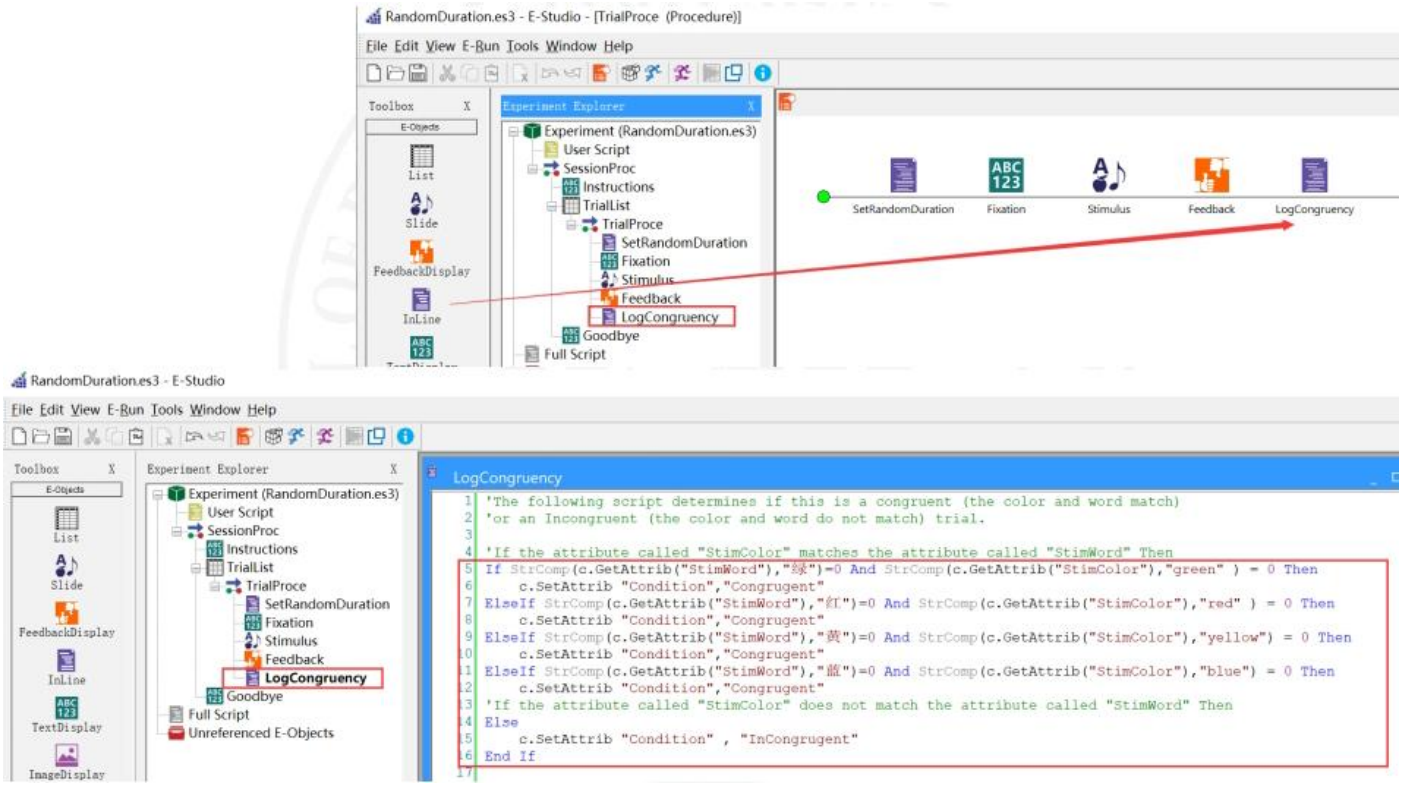
- Feedback.AccStats.Reset 用于清空之前的练习数据，确保重新练习时统计从零开始。
- **重要规则**：GoTo 语句只能跳转到**同一 Procedure 内**定义的 Label 对象。



步骤 5：记录词义-颜色的一致性条件

需要在每个试次中记录当前刺激是“一致”(Congruent)还是“不一致”(Incongruent)。

1. **在刺激呈现前添加 Inline 对象**：例如，在 Stimulus 对象之前添加一个名为 SetCongruency 的 Inline。
2. **编写判断脚本**：比较 StimWord 和 StimColor 的属性。



If c.GetAttrib(“StimWord”) = “绿” And c.GetAttrib(“StimColor”) = “green” Then

c.SetAttrib “Condition”, “Congruent”

ElseIf ‘…（判断其他一致情况，如红-red，黄-yellow，蓝-blue）

Else

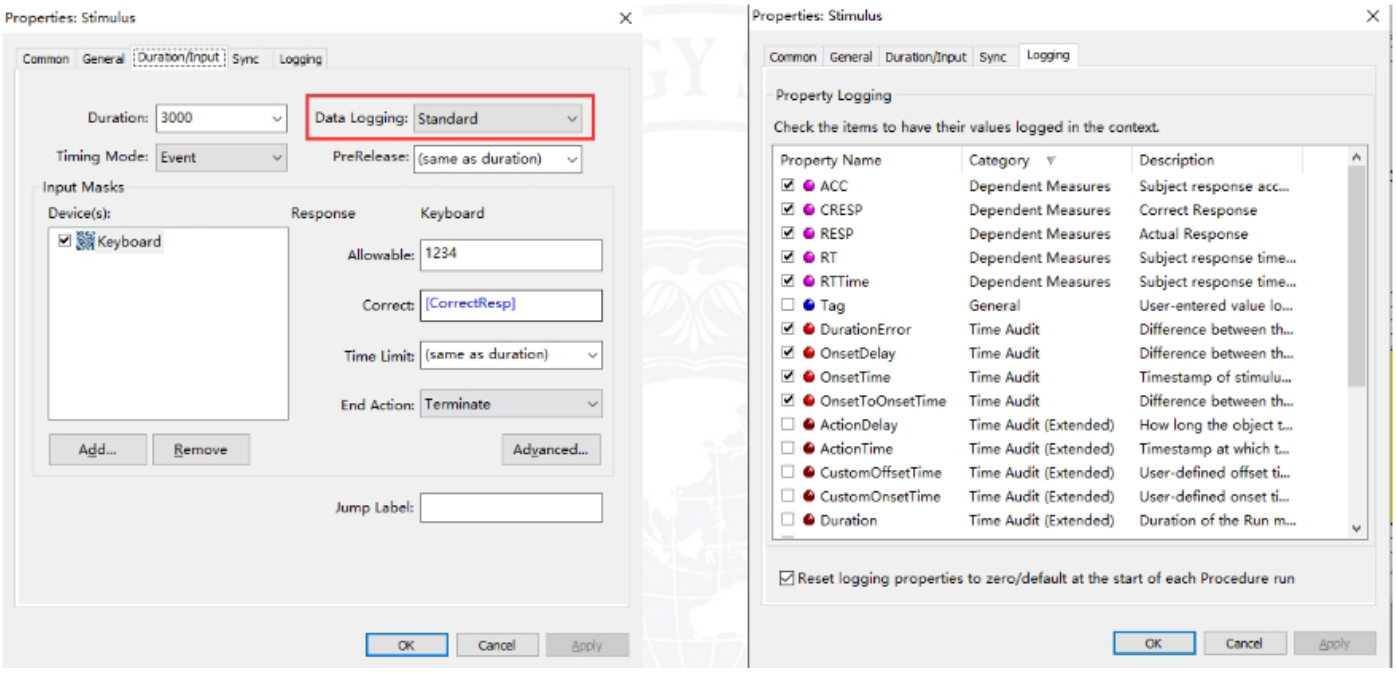
c.SetAttrib “Condition”, “Incongruent”

End If

- 这样会在数据文件中生成一列 Condition，标记每个试次的一致性。

步骤 6：数据记录与反馈设置

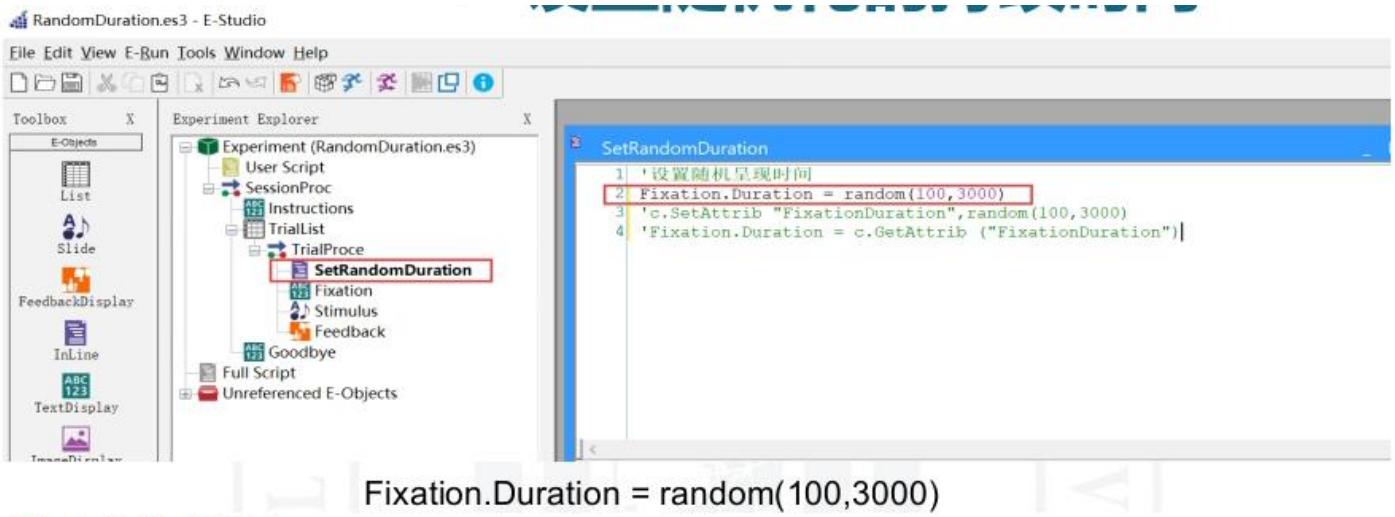
1. **确保数据记录**：在 Stimulus 对象的属性中，将 Data Logging 设置为 Standard。



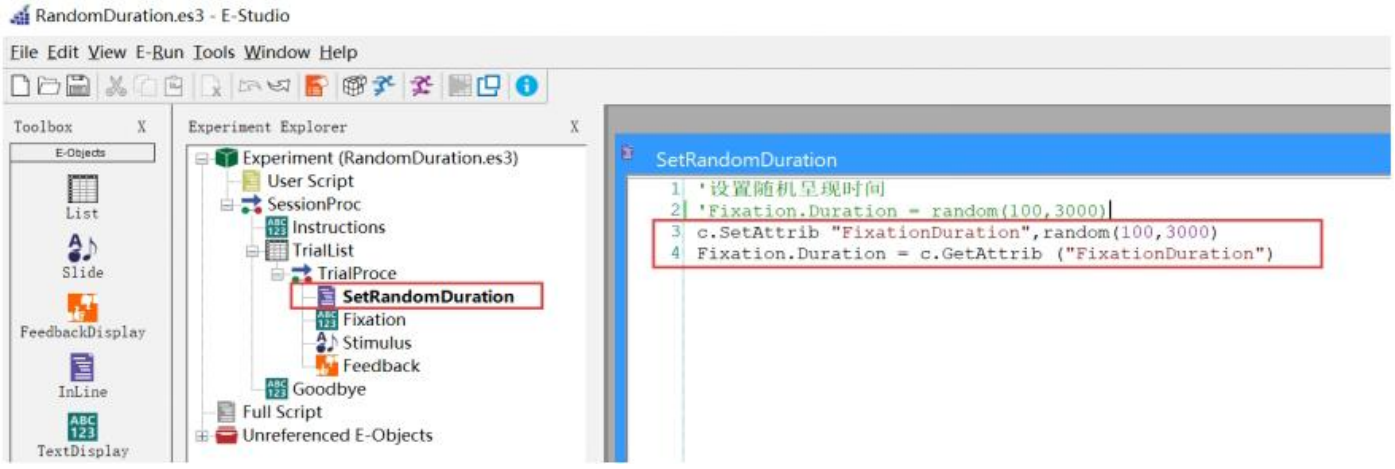
2. **设置反馈（仅练习阶段）**：在练习流程的 FeedbackDisplay 对象中，可以设置其 Text 属性为“正确率：[Feedback.AccStats.Mean]”等，给被试实时反馈。

步骤 7：其他高级设置（根据文档可选）

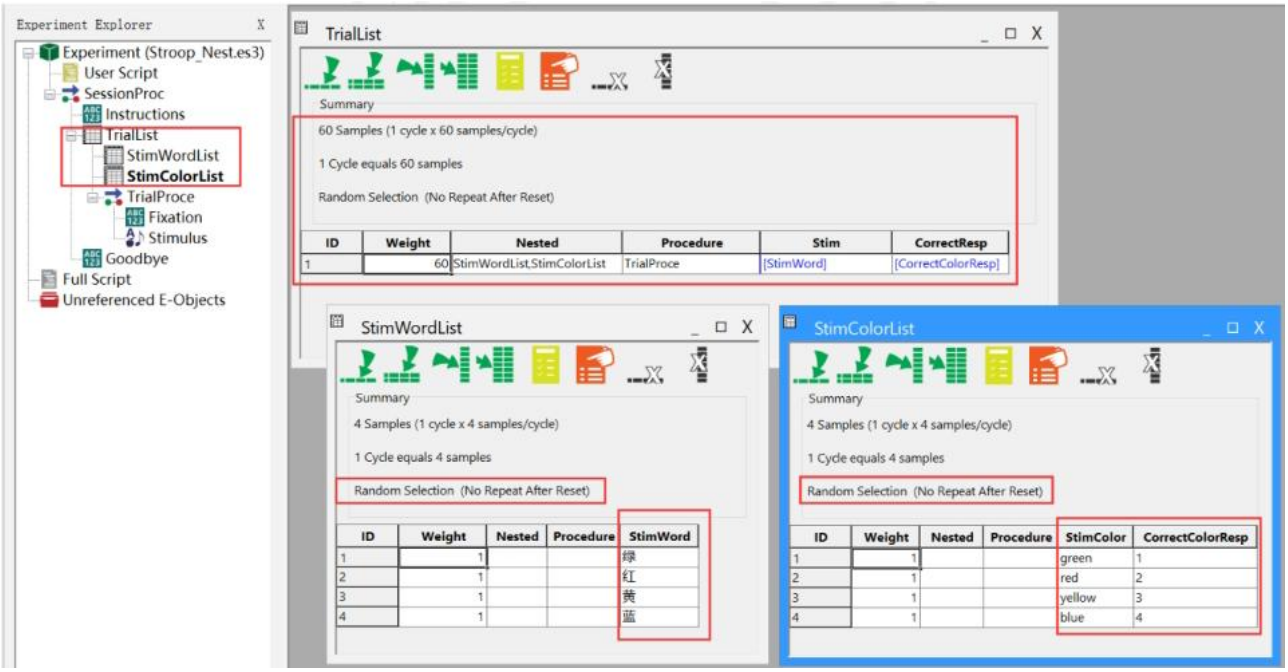
1. **设置随机化持续时间**：在 Fixation 前添加 Inline，写入 Fixation.Duration = random(100, 3000)，使注视点呈现时间在 100-3000 毫秒间随机。



Fixation.Duration = random(100,3000)

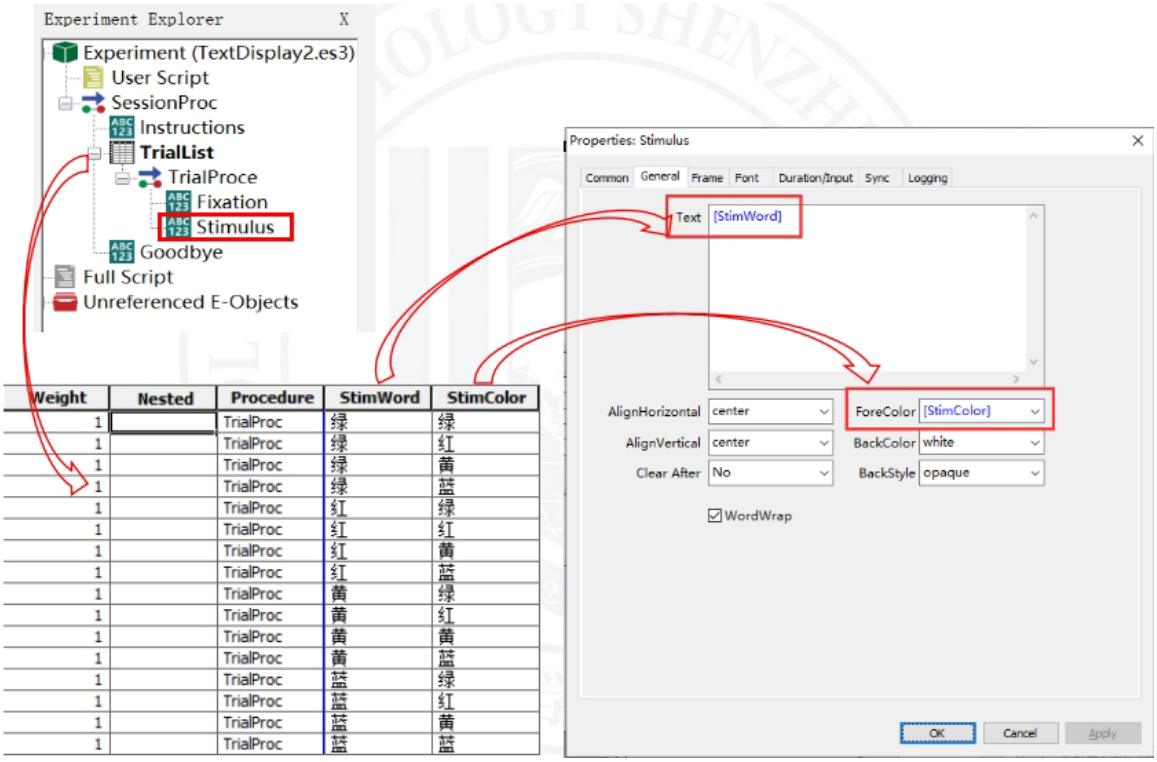


2. **使用嵌套 List**：如果希望词和颜色完全随机组合，可以创建两个独立的 List（StimWordList 和 StimColorList），然后在主 TrialList 的 Nested 属性中填入这两个 List 的名字（用逗号分隔）。在 Stimulus 中仍用[StimWord]和[StimColor]引用属性。



3. **优化刺激加载**：为了保证 Feedback 能立即呈现，应将 Feedback **之前**的那个对象（通常是 Stimulus）的 PreRelease 属性设置为 0，防止其预加载造成延迟。

总结：构建 Stroop 实验的核心步骤是：**搭建流程结构 -> 在 List 中定义材料 -> 在刺激对象中引用属性 -> 用 Inline 实现条件跳转和逻辑判断 -> 设置数据记录**。重点在于理解 List 的属性调用和 Inline 的流程控制逻辑。

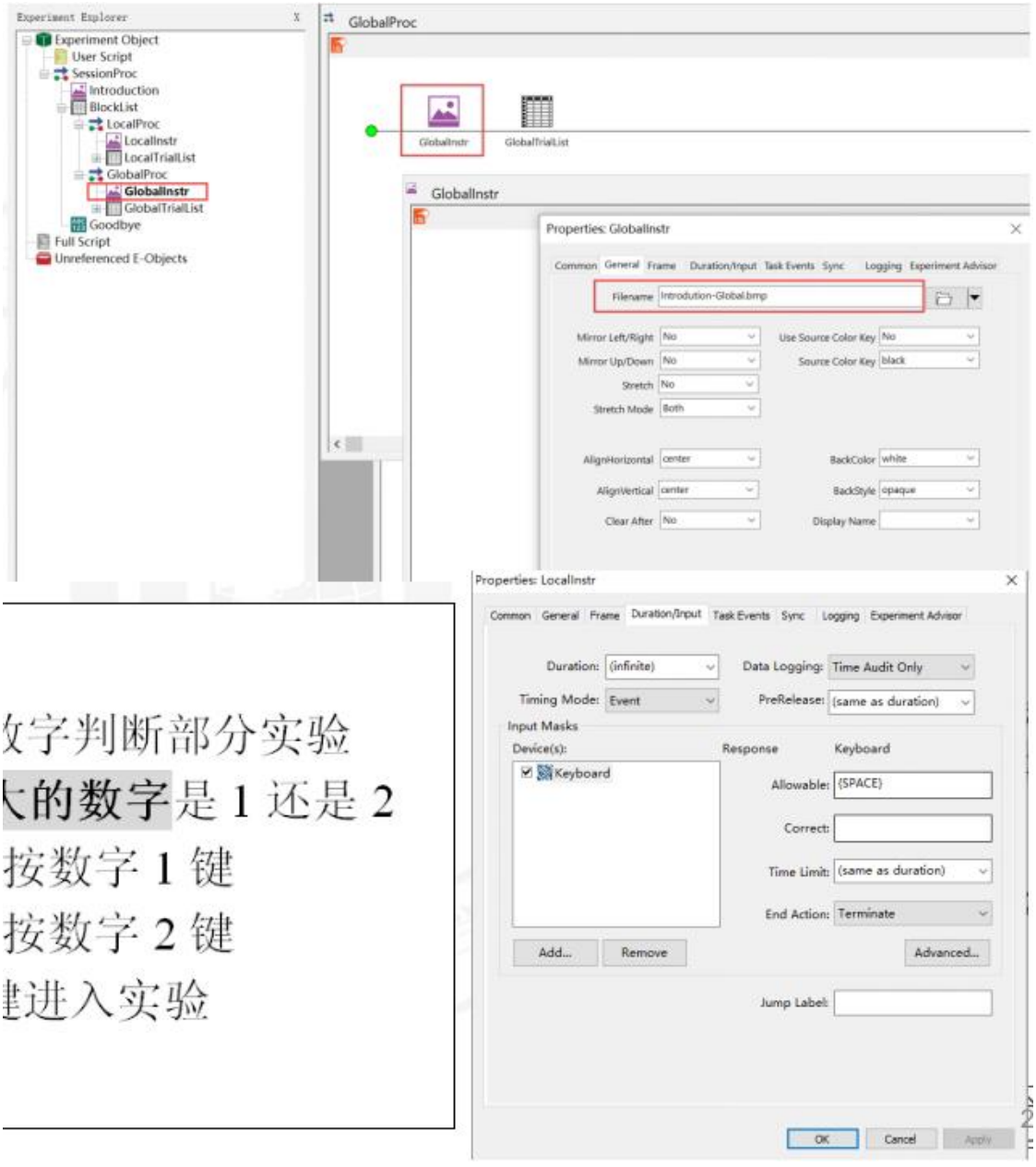


构建整体-局部范式（Navon 任务）详细操作

一、实验核心设计

该范式用于研究整体与局部信息加工的优先级。实验中，刺激是由许多**小字母（局部）** 组成的一个**大字母（整体）**。

- **自变量 1：一致性**：整体字母与局部字母的关系。
  - **一致**：整体字母与局部字母相同（例如，由许多小“H”组成的大“H”）。
  - **不一致**：整体字母与局部字母不同（例如，由许多小“S”组成的大“H”）。
- **自变量 2：注意指向**：要求被试报告的目标。
  - **注意整体**：判断大字母是“1”还是“2”。
  - **注意局部**：判断小字母是“1”还是“2”。



- **因变量**：被试的反应时和正确率。

二、详细操作步骤

步骤 1：创建实验结构

在 E-Studio 的 Experiment Explorer 中创建如下结构：

1. **SessionProc**：实验总流程。
2. **Introduction**：指导语界面（可使用 TextDisplay 或 ImageDisplay 呈现图片格式的指导语）。
3. **BlockList**：**关键对象**，用于控制实验的"注意指向"条件。它包含两行，分别指向两个不同的流程（Procedure）：
    - LocalProc （局部判断任务流程）
    - GlobalProc （整体判断任务流程） 在 BlockList 的属性中，将 Selection 设置为 Random 以实现任务顺序的随机化。
4. **LocalProc 与 GlobalProc**：分别对应"判断局部"和"判断整体"的任务流程。每个流程内部结构相似：
    - LocalInstr / GlobalInstr：该任务块开始前的具体指导语（如："现在进行局部数字判断部分实验…"）。
    - LocalTrialList / GlobalTrialList：定义该任务块下所有试次材料的 List。
    - LocalTrialProc / GlobalTrialProc：定义单个试次流程的 Procedure。
5. **Goodbye**：结束语界面。

步骤 2：准备实验材料与 List 设置

1. **准备刺激图片**：根据实验设计（整体字母为 1 或 2，局部字母为 1 或 2，一致或不一致），生成对应的图片文件。文档中示例的命名格式为 [Global]-[Local].bmp (例如：1-2.bmp 表示整体为"1"，局部为"2"，属于不一致条件)。
2. **配置 TrialList**：分别为局部任务(LocalTrialList)和整体任务(GlobalTrialList)创建 List。
    - 在 List 中定义必要的属性列，例如：
        - StimulusFile：刺激图片文件名，如 [Global]-[Local].bmp。
        - Global：整体字母（1 或 2）。
        - Local：局部字母（1 或 2）。
        - Consistent：一致性条件（Consistent 或 Inconsistent）。
        - CorrectResp：根据当前任务是判断整体还是局部，填入对应的正确反应键（如"1"或"2"）。
    - 在 List 中填入所有实验条件组合对应的行。

步骤 3：构建单个试次流程 (TrialProc)

在 LocalTrialProc 或 GlobalTrialProc 中，构建如下序列：

1. **SoundOut1**：可选的提示音对象，播放一段声音提示试次开始。
2. **Blank**：空屏，作为刺激出现前的间隔。
3. **Stimulus**：**核心刺激呈现对象**。通常使用 ImageDisplay。
    - 在 Filename 属性中，设置为 [StimulusFile]，以调用 List 中定义的图片。
    - 在 Duration/Input 中，设置一个时间限制（如 3000ms），并添加键盘输入掩码（Keyboard），将 Allowable 设置为 1 和 2，Correct 设置为 [CorrectResp]。
    - 确保 Data Logging 为 Standard，以记录反应时和正确率。

步骤 4：实现分条件反馈（可选但推荐用于练习）

如果需要根据被试反应的正确性和反应时给予详细反馈（如“反应太快”、“正确”、“太慢”、“错误”）：

1. **添加反馈判断对象**：在 Stimulus 对象后，插入一个 Inline 对象（例如命名为 SetFeedback）。
2. **编写反馈逻辑脚本**：在 SetFeedback 的 Script 窗口中，根据 Stimulus.ACC (正确性) 和 Stimulus.RT (反应时) 设置一个自定义属性（如 whichState），用于决定显示哪种反馈。
3. If Stimulus.ACC = 1 Then
4. 	If Stimulus.RT > 2500 Then
5. 		c.SetAttrib “whichState”, “tooSlow”
6. 	ElseIf Stimulus.RT >= 300 Then
7. 		c.SetAttrib “whichState”, “Correct”
8. 	Else
9. 		c.SetAttrib “whichState”, “tooFast”
10. 	End If
11. Else
12. 	c.SetAttrib “whichState”, “Wrong”
13. End If
14. c.SetAttrib “myRTTime”, Stimulus.RT
15. **添加反馈呈现对象**：在 Inline 后插入一个 Slide 对象（例如命名为 RespInfo）。
    - 在 Slide 中创建多个状态（States），分别命名为 Correct， tooFast， tooSlow， Wrong。
    - 在每个状态的文本中，可以嵌入属性引用，如 “反应正确，反应时间为 [myRTTime] 毫秒”。
    - 在 Slide 的 General 属性中，将 Active State 设置为 [whichState]。这样，系统会根据 Inline 中设置的 whichState 值，自动激活对应的反馈界面。

步骤 5：数据记录

确保在 Stimulus 对象的属性中启用了数据记录。实验运行后，数据文件将记录关键变量，如：

- Subject：被试编号。
- Stimulus.RT / Stimulus.ACC：反应时和正确率。
- Global / Local：整体和局部字母。
- Consistent：一致性条件。
- Procedure：来自哪个流程（可区分是局部还是整体任务）。

步骤 6：运行与数据合并

1. **生成数据**：在 E-Studio 中，通过 Tools -> Generate 生成实验程序。在运行界面输入被试编号进行测试。
2. **合并数据**：实验完成后，使用 E-Prime 的 E-Merge 工具，选择所有被试的 .edat2 数据文件进行合并，得到一个可用于统计分析的总数据文件。

总结：构建 Navon 整体-局部范式实验的核心是：**利用 BlockList 控制“注意指向”自变量 -> 在 TrialList 中定义“一致性”自变量和材料 -> 在 TrialProc 中用 ImageDisplay 呈现组合刺激 -> 通过 Inline 和 Slide 实现复杂反馈 -> 确保关键变量被记录**。

社会排斥范式（Cyberball 网络传球任务）

一、实验核心设计

Cyberball 任务用于研究社会排斥。在程序中，模拟三名玩家（1 号、2 号、3 号）进行传球游戏。

- **玩家 2**：真实的被试。
- **玩家 1 和玩家 3**：由程序控制的虚拟玩家。
- **核心变量**：使用一个全局变量 CurrentPlayer（整数型）来**实时追踪球在谁手中**。
- **两种实验条件**：
    - **接纳条件 (InclusionProc)**：虚拟玩家（1 号和 3 号）会以较高比例（如 50%）将球传给真被试（2 号）。
    - **排斥条件 (ExclusionProc)**：虚拟玩家（1 号和 3 号）会以极低比例（如 10%）将球传给真被试（2 号），他们之间互相传球。

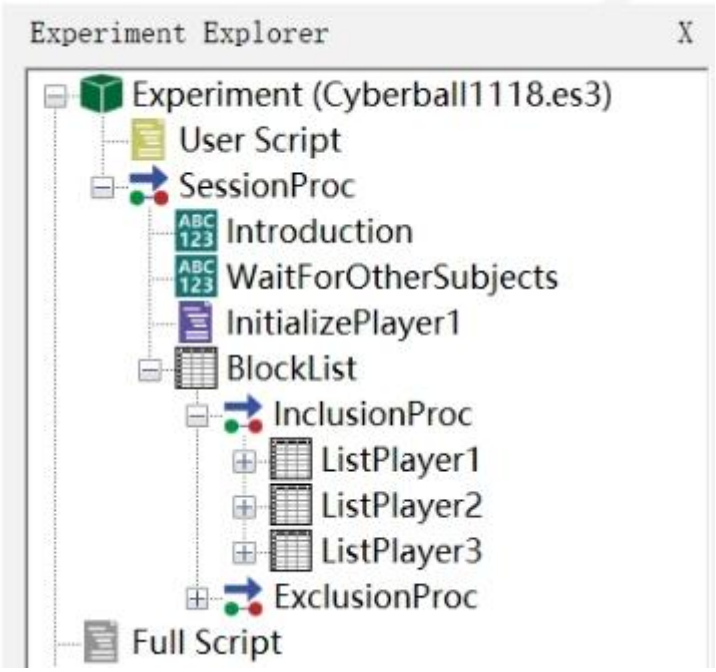
二、详细操作流程与步骤

整个实验程序围绕“**判断当前持球者 -> 执行对应传球动作 -> 更新持球者**”这一核心循环构建。

步骤 1：创建实验整体结构

在 Experiment Explorer 中创建如下结构：

1. **SessionProc**：实验的总流程。
2. **Introduction**：指导语界面（TextDisplay 或 Slide)。指导被试（2 号玩家）按“1”键传给左边玩家，按“3”键传给右边玩家。
3. **WaitForOtherSubjects**：一个简单的等待界面（如空屏 2000ms)，营造等待其他玩家的氛围。
4. **InitializePlayer1**：一个 Inline 对象。在此写入脚本 CurrentPlayer = 1，**将球的初始持有者设置为 1 号玩家**。
5. **BlockList**：一个 List 对象。这是控制实验条件的核心。
    - 在 List 中设置两行，Procedure 列分别指向 InclusionProc（接纳流程）和 ExclusionProc（排斥流程)。
    - 将 Selection 属性设置为 Random，以随机呈现接纳和排斥条件。
6. **InclusionProc** 与 **ExclusionProc**：分别对应“接纳”和“排斥”条件的实验流程。两者的内部结构**完全一致**，但其中 ListPlayer1 和 ListPlayer3 中设定的传球概率不同（详见步骤 3)。



7. **Full Script**：在 User Script 中声明全局变量 Dim CurrentPlayer As Integer。

步骤 2：构建接纳/排斥条件的内部流程（以 InclusionProc 为例)

在 InclusionProc 这个 Procedure 中，按顺序放置以下对象：

1. **GoToPlayer**：一个 Inline 对象。其核心作用是**根据 CurrentPlayer 的值，跳转到对应玩家的传球流程**。
2. If CurrentPlayer = 1 Then
3. GoTo Player1
4. ElseIf CurrentPlayer = 2 Then

5. GoTo Player2
6. ElseIf CurrentPlayer = 3 Then
7. GoTo Player3
8. End If
9. **Player1, Player2, Player3**：这三个是 Label 对象，作为上述 Inline 跳转的目标。
10. **ListPlayer1, ListPlayer2, ListPlayer3**：分别放置在对应 Label 下方的 List 对象，用于定义该玩家的行为。
11. **EndOfTrial**：一个 Label 对象，标志一个传球试次结束。在每个玩家 List 的流程最后，都需要一个 Inline 对象包含 GoTo EndOfTrial 语句，以结束当前试次并返回 BlockList 开始下一个试次。

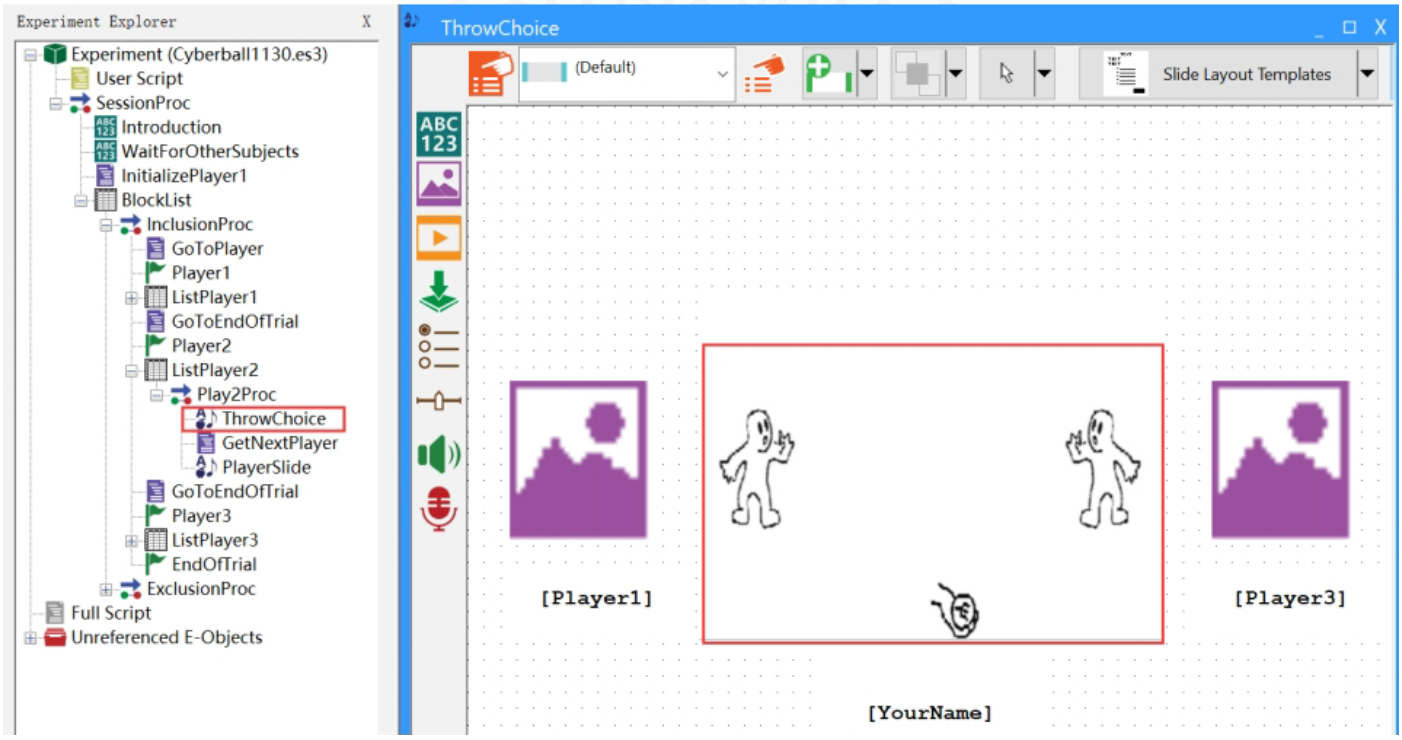
步骤 3：配置各玩家的 List 与传球逻辑

每个玩家的 List 决定了其行为模式。

- **ListPlayer1 (虚拟玩家 1)**:
  - **属性**：至少包含一列 TargetPlayer，其值为 2 或 3，代表本次传球的目标。
  - **设置**：在 InclusionProc 中，Weight 值可设为 5 和 5，实现 50%传给 2 号（被试），50%传给 3 号。在 ExclusionProc 中，Weight 值可设为 1 和 9，实现 10%传给 2 号，90%传给 3 号。
  - **流程 (Play1Proc)**:
    1. PrepareToThrow1 (Inline)：根据 TargetPlayer 生成传球视频文件名。
    2. If CurrentPlayer = 1 Then
    3. c.SetAttrib “MovieName”, “1to” & c.GetAttrib(“TargetPlayer”) & “.wmv”
    4. End If

5. PlayerSlide (SlideMovie 对象)：在 Movie 属性中，将 Filename 设置为 [MovieName]，以播放对应的传球动画（如 1to2.wmv）。
6. SendNextPlayer (Inline)：**关键步骤**。更新全局变量，将球权移交给目标玩家。
7. CurrentPlayer = CInt(c.GetAttrib("TargetPlayer"))

- **ListPlayer2 (真被试)**:
  - **流程 (Play2Proc)**:
    1. ThrowChoice (Slide)：呈现一张**静止的、球在被试手中的图片**（如 2Ball.JPG），等待被试按键。

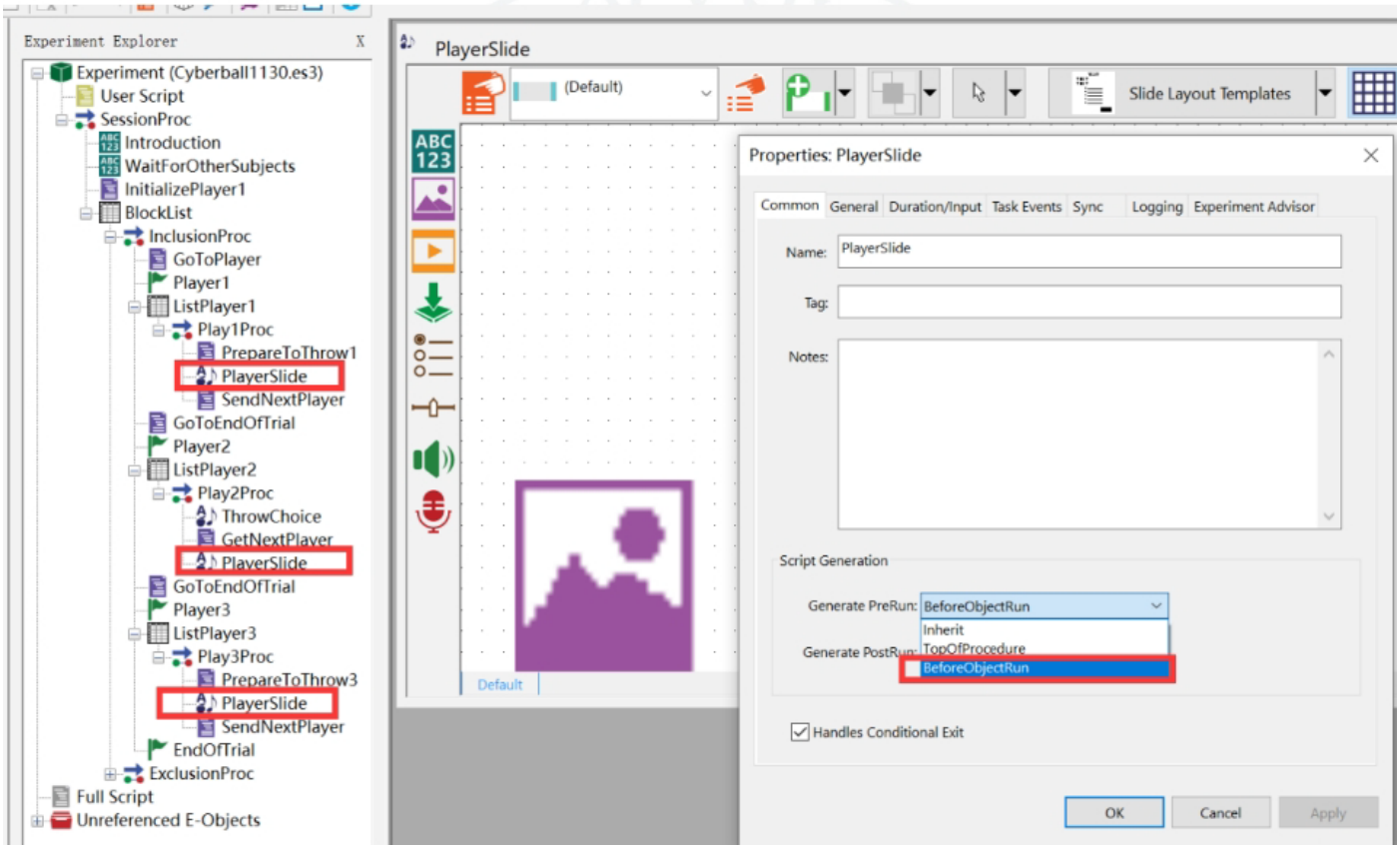


       - 在 Input Mask 中，设置 Allowable 为 1 和 3。
       - 被试按"1"表示传给 1 号玩家，按"3"表示传给 3 号玩家。
    2. GetNextPlayer (Inline)：根据被试的按键反应，设置本次传球的目标。
    3. If ThrowChoice.Resp = "1" Then
    4. c.SetAttrib "TargetPlayer", 1
    5. ElseIf ThrowChoice.Resp = "3" Then
    6. c.SetAttrib "TargetPlayer", 3
    7. End If
    8. PlayerSlide (SlideMovie)：同样，根据 TargetPlayer 播放对应的传球动画（如 2to1.wmv）。
    9. SendNextPlayer (Inline)：与玩家 1 相同，更新 CurrentPlayer。

- **ListPlayer3 (虚拟玩家 3)**:
    - 配置逻辑与 ListPlayer1 完全对称，视频文件名为 3to1.wmv 和 3to2.wmv。其传球比例设置与玩家 1 相同。

步骤 4：设置关键对象属性

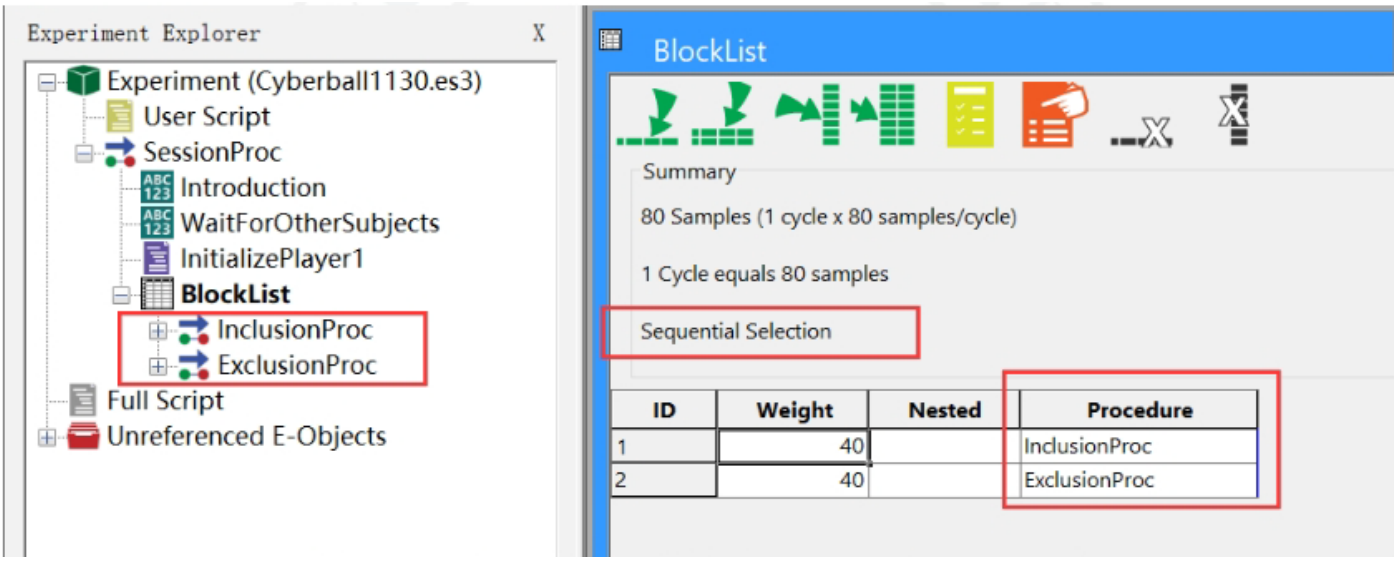
1. **PlayerSlide (SlideMovie 对象)**:
    - 将 Duration 设置为 (infinite)。
    - 在 Movie 帧的 Filename 属性中，填入 [MovieName]。
    - 在 Script Generation 选项卡中，将 Generate PreRun 设置为 BeforeObjectRun。**这是确保视频文件能根据 Inline 脚本动态加载的关键设置**。



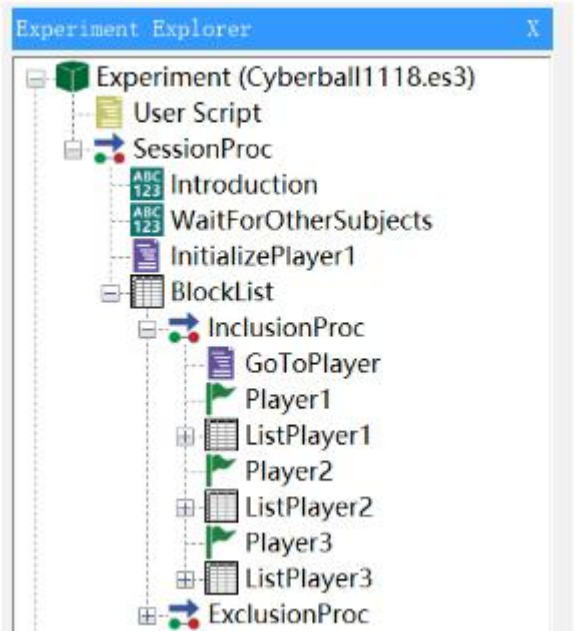
2. **BlockList**：根据练习要求，设置 InclusionProc 和 ExclusionProc 各运行 20 次（可通过 Samples 或 Cycles 控制）。

三、流程总结

1. 实验开始，球初始在**玩家 1** (CurrentPlayer=1)。
2. 进入 BlockList，随机进入 InclusionProc 或 ExclusionProc。



3. 在流程中，GoToPlayer 根据 CurrentPlayer 值跳转到**玩家 1** 的流程。
4. **玩家 1** 的 ListPlayer1 随机决定本次传球给谁（TargetPlayer），通过 Inline 生成视频文件名并播放，播放后更新 CurrentPlayer 为目标玩家编号。
5. 本次试次结束(GoTo EndOfTrial)，流程返回 BlockList，开始下一个试次。
6. 下一个试次开始，GoToPlayer 再次判断 CurrentPlayer 值。若此时 CurrentPlayer=2，则跳转到**玩家 2（真被试）**的流程。



7. **玩家 2** 的界面等待被试按键选择传球目标，记录反应，播放相应视频，并更新 CurrentPlayer。
8. 如此循环，直到完成 BlockList 中设定的所有传球次数。

核心：整个实验通过**一个全局变量(CurrentPlayer)** 和 **GoToPlayer 这个 Inline** 来串联三个玩家的行为，模拟出连续的传球互动。接纳与排斥条件的区别，仅在于**虚拟玩家（1 号和 3 号）的 List 中，传球给被试（2 号）的试次所占的权重(Weight)**不同。