Signal Taste01 wird geändert, Verschiebung

```
--[[
Steuern Sie komplexe Abläufe Ihrer Anlage mit Hilfe der
Skriptsprache Lua:
  - Drücken Sie STRG + Leertaste, um eine Übersicht aller
    Funktionen und Variablen zu erhalten
  - Drücken Sie F1 zum Öffnen der Online-Hilfe mit weiteren
    Informationen und zusätzlichen Hilfestellungen
--]]
if $("Taste ▼01").state == 0 then
  -- Verschiebung x=x+a, y=y+b,z=z+c.
  -- Geben Sie ein Dollarzeichen ein und wählen Sie das Objekt.
  -- Übersetzen ist eine spezifische Verschiebung.
  -- Wenn du transformation:translate eingibst, erhältst du weitere Informationen.

  -- translate(a,b,c) verschiebt das Objekt von (x,y,z) nach (x+a,y+b,z+c)
  -- Beachte, dass du hier in Metern arbeitest, also benutze den Maßstab und berechne neu.
  -- In LUA müssen Sie den Dezimalpunkt "." verwenden, nicht das Komma ",".

  -- Beispiel im Maßstab HO 1/87 x=x+300,y=y+300, z=z+0, und 0,3 * 87 = 26,1.
  -- Natürlich kann man das hier auch in Lua berechnen.

   $("Loc 01").transformation:translate(26.1,26.1,0)

  -- Bitte beachten Sie auch die Varianten translateX, translateY und translateZ.
else
  -- Verschiebung zurück in die erste Position.
  $("Loc 01").transformation:translate(-26.1,-26.1,0)
end
```

Signal Taste02 wird geändert, Rotation

```
if $("Taste ▼02").state == 0 then
  -- Rotation um die z-Achse z=z+r (Radiale).
  -- Geben Sie ein Dollarzeichen ein und wählen Sie das Objekt.
  -- RotateZ ist eine spezielle Transformation.
  -- Wenn Sie transformation:rotateZ eingeben, erhalten Sie weitere Informationen.

  -- Berücksichtigen Sie, dass Sie hier in Radialen arbeiten, nicht in Grad.
  -- In LUA müssen Sie den Dezimalpunkt "." verwenden, nicht das Komma ",".

  -- Beispiel Drehen über 33,45°. 180° = Pi Radiale.
  -- Sie können math.pi für den Wert von Pi anstelle von 3,1415 verwenden.
  -- Sie können math.rad verwenden, um Grad in Radiale zu konvertieren.

  local AngleRad = math.rad(33.45)

  -- oder
  -- local AngleRad = 33.45 * math.pi / 180

   $("Container 01").transformation:rotateZ(AngleRad)
```

```
  -- Bitte beachten Sie die Varianten rotateX rotateY.

  -- ACHTUNG ACHTUNG ACHTUNG, verwenden Sie NICHT direkt transformation.rotation mit einer Quaternion ! Es sei denn, Sie sind ein Experte in Quaternionen-Mathematik, was ich nicht bin!
  -- Dieser Mißbrauch kann sehr unangenehme Auswirkungen haben, daher ist eine direkte Verwendung nur für Entwickler und Insider.
  -- JEDOCH werden wir in zwei weiteren Beispielen erklären, wie man direkt um einen bestimmten Winkel drehen kann, ohne eine Quaternion einzugeben.


  -- Sie können math.rad verwenden, um Grad in Radiale umzuwandeln
  $("LabelRadials").text = math.rad(33.45) .. " Rad"
  -- Sie können math.deg verwenden, um Radiale in Grad zu konvertieren
  $("LabelDegrees").text = math.deg(AngleRad) .. " °"
else
  -- Rotation zurück um die z-Achse.
  -- Drehung zurück über z.
  -- Local AngleRad = -33.45 * math.pi / 180
  local AngleRad = math.rad(-33.45)
   $("Container 01").transformation:rotateZ(AngleRad)

  $("LabelRadials").text = "0 Rad"
  $("LabelDegrees").text = "0 °"
end
```

Signal Taste03 wird geändert, Objektposition anfahren

```
if $("Taste ▼03").state == 0 then
  -- Objekt an Position 02 verschieben.
  -- Transformation.position gibt die Position eines Objekts an.
  -- Sie können auch über transformation.position eine Position setzen.
  -- Drücken Sie das Dollarzeichen und wählen Sie das Objekt.
  -- Wenn Sie position nach der "transformation." eingeben, erhalten Sie weitere Informationen zur Position.

  -- Beispiel: Der Container wird an der Position des Labels platziert.

  local p =  $("Lbl Position02").transformation.position
  $("Container 02").transformation.position = p

  -- Wenn Sie möchten, dass sich der Container zusammen mit dem Etikett bewegt, können Sie dies mit dem grafischen EV wie folgt erreichen

  $("Container 02").link = $("Lbl Position02")
else
  -- Entkoppeln Sie das Objekt zuerst;

  $("Container 02").link = $("")

  -- Objekt auf Position 01 verschieben.
  local p =  $("Lbl Position01").transformation.position
```

```
  $("Container 02").transformation.position = p
end
```

Signal Taste04 wird geändert, direkt auf x,y,z fahren (globale Koordinaten)

```
if $("Taste ▼04").state == 0 then
  -- Verschieben Sie das Objekt direkt an eine x,y,z-Position.
  -- Sie gehen direkt zu einer neuen Position über die globalen Koordinaten x,y,z.
  -- Geben Sie ein Dollarzeichen ein und wählen Sie das Objekt aus.
  -- Wenn Sie transformation.position eingeben, erhalten Sie weitere Informationen.

  -- Beachten Sie, dass Sie hier im Skript in Metern arbeiten, verwenden Sie also den Maßstab.
  -- Wir verwenden die {x=a,y=b,z=c} Tabelle aus Lua.
  -- Beispiel x=300,y=-20,z=0 0,3 *87 = 26.1 -0,02 *87 = 1.74 0.
  -- In LUA muss man den Dezimalpunkt "." verwenden, nicht das Komma ",".
  -- Wir lassen es LUA berechnen

  a=(300/1000)*87
  b=(-20/1000)*87
  c=0

  local p = { x=a, y=b, z=c }
  $("Tree 01").transformation.position = p
else
-- Zurück zu x, y, z der vorherigen Position.
  a=(-40/1000)*87
  b=(130/1000)*87
  c=0

  -- Gehen Sie zurück zur ursprünglichen Position.
  local p = { x=a, y=b, z=c }
  $("Tree 01").transformation.position = p
end
```

Signal Taste05 wird geändert, Rotation gleich einem anderen Objekt setzen

```
if $("Taste ▼05").state == 0 then
  -- Stellen Sie die Drehung des blauen Containers gleich der des grünen Containers ein.
  -- Geben Sie das "Dollar"-Zeichen ein, um das Objekt auszuwählen.
  -- Hier erfassen wir die Drehung eines Objekts, das sich in der richtigen Drehposition für uns
befindet.
  -- Dann geben wir diese Drehung an ein anderes Objekt weiter.
  -- Wenn du transformation.rotation eingibst, erhältst du mehr Informationen.

  -- r übernimmt die Rotation von Container 05.
  -- Container 04 übernimmt die Rotation von r.
  local r = $("Container 05").transformation.rotation
  $("Container 04").transformation.rotation = r
else
  -- Die Drehung des blauen Containers wird der des braunen Containers angeglichen.
  local r = $("Container 03").transformation.rotation
  $("Container 04").transformation.rotation = r
```

```
end
```

Signal Taste06 wird geändert, Objekt auf ein anderes Objekt legen

```
if $("Taste ▼06").state == 0 then
  -- Ein Objekt auf ein anderes Objekt legen.
  -- Wir führen hier eine weitere Methode ein.
  -- Wir können die Größe eines Objekts im Skript ermitteln.

  -- Geben Sie ein Dollarzeichen ein und wählen Sie das Objekt
  -- Wenn Sie . nach dem Objekt eintippen und nach Objekten suchen, können Sie die Größe dieses Objekts herausfinden
  -- Die Größe ist eine LUA-Tabelle im Vektorformat (x,y,z), sie gibt die Abmessungen des Objekts an.

  local si = $("Container 07").size
  -- Wir nehmen die Höhe z des anderen Objekts, weil wir auf dieses draufgehen wollen.
  local h = si["z"]
  -- Wenn Sie sie benötigen, können Sie die x- und y-Werte auf ähnliche Weise ermitteln.

  -- Kombination mit früheren Methoden.
  -- Wir nehmen die Position und die Drehung von dem Objekt, zu dem wir gehen.  ( siehe vorherige Erklärungen )

  local p = $("Container 07").transformation.position
  local r = $("Container 07").transformation.rotation

  -- geben Sie dem Objekt die richtige Position und Drehung.     ( siehe vorherige Erklärungen )
  $("Container 06").transformation.position = p
  $("Container 06").transformation.rotation = r

  -- Ändern Sie nur die z-Position, um auf das andere Objekt zu gelangen.
  -- Verschiebe die z-Position nach z+h ( siehe vorherige Erklärungen )

  $("Container 06").transformation:translate(0,0,h)
else
  -- Das Objekt wieder an die vorherige Position, das Etikett, setzen.
  -- Setze das Objekt an die Position des Etiketts. ( zurück, wo es herkam )


  local p =  $("Lbl Position03").transformation.position
  local r =  $("Lbl Position03").transformation.rotation
  $("Container 06").transformation.position = p
  $("Container 06").transformation.rotation  = r
end
```

Signal Taste07 wird geändert, Rotation zurücksetzen

```
if $("Taste ▼07").state == 0 then
 if $("Taste ▼07").state == 0 then
  -- Setzt die Drehung eines Objekts auf 0°,0°,0°.
```

```
  -- Es ist sehr nützlich, wenn ein Objekt gedreht wird, um es wieder in die Standard-Rotation 0°,0°,0° zu bringen.
  -- Das ist genau das, was resetRotation() tut.
  -- Geben Sie ein Dollarzeichen ein und wählen Sie das Objekt.

  -- ResetRotation ist eine spezielle Transformation
  -- Wenn du transformation:resetRotation eingibst, siehst du einige zusätzliche Informationen.

  $("Container 08").transformation:resetRotation()
else
  -- Setzen Sie die Rotation wieder auf die des braunen Containers 07.
  -- Container 08 erhält die gleiche Drehung wie Container07.
  $("Container 08").transformation.rotation = $("Container 07").transformation.rotation
end
```

Signal Taste08 wird geändert, direkt auf ein bestimmtes z rotieren

```
if signal.state == 0 then
  -- Drehen Sie direkt auf 33,45°.
  --- Dazu setzen wir zunächst die Drehung auf 0,0,0 zurück.
  $("Container09").transformation:resetRotation()

  ----------------------------------------------------------------------------------------
  -- Lesen Sie die zusätzlichen Informationen über rotateZ vorher in einem anderen Beispiel.
  -- Wie z.B. die Verwendung des Dollarzeichens für die Objektauswahl, Umwandlung Grad - Radiale, nur das "." für das Komma "," verwenden.
  -- Wenn Sie transformation:rotateZ nach der "transformation:" eingeben, erhalten Sie weitere Informationen über rotateZ.


  local z = math.rad(33.45)
  -- Dann verwenden wir das, was wir zuvor gelernt haben, um über einen bestimmten Wert zu rotieren.
  $("Container09").transformation:rotateZ(z)
  -- Und fertig.

  -- Studieren Sie auch rotateX und rotateY, sie funktionieren ähnlich.
else
  -- Drehen Sie direkt auf -45°.
  --- Dazu setzen wir die Drehung zunächst auf 0,0,0 zurück.
  $("Container09").transformation:resetRotation()
  -- Lua arbeitet mit Radialen, also wandeln wir die -45° in Radiale um.
  -- Achten Sie darauf, dass Lua mit "." als Dezimalpunkt arbeitet, nicht mit unserem üblichen ",".
  local z = math.rad(-45)
  -- Dann verwenden wir das, was wir zuvor gelernt haben, um über einen bestimmten Wert zu rotieren
  $("Container09").transformation:rotateZ(z)
  -- Und fertig.
end
```

Signal Taste 09 wird geändert, viele Objekte in ihre vorherige Position /

Drehung bringen

```
if signal.state == 0 then
  -- Zuvor wird eine Liste Objekte manuell ausgefüllt, indem die Objekte markiert und diese in das Modul Liste Objekte eingefügt werden.

  local t = $("Events").variables["Objects"]
  for i, Iter in ipairs(t) do
    -- Für jedes Objekt aus der Liste geben Sie ihm seine gespeicherte Position/Drehung zurück.

    -- Sie befinden sich in den Variablen ObjRotation/ObjPosition.

    -- Gib einem Objekt die vorherige Position und Rotation aus den Variablen.
    -- Iter ist hier ein Objekt aus der Iteration.
    -- Die Transformationsposition/Rotation eines Objekts wird auf die vorherige Position/Drehung dieses Objekts gesetzt und in Variablen gespeichert.
    -- Jedes Objekt als vier Variablen: Tabellen mit der vorherigen Position und Rotation.

    -- Iter.transformation.position ist eine Tabelle mit den Werten von x, y, z.
    -- Iter.transformation.rotation ist eine Tabelle mit den Werten x,y,z und w als Quaternion.  Wir brauchen nicht zu wissen, was eine Quaternion ist.
    -- Es ist nur eine Möglichkeit, die Drehung eines Objekts zu identifizieren.

    -- Wenn Sie transformation.position eingeben, erhalten Sie weitere Informationen.
    -- Wenn Sie transformation.rotation eingeben, erhalten Sie weitere Informationen.

    -- Setzt die aktuelle Position/Drehung auf die vorherige Position/Drehung, die in den Variablen ObjPosition/ObjRotation des Objekts gespeichert ist.
    Iter.transformation.position = Iter.variables["ObjPosition"]
    Iter.transformation.rotation  = Iter.variables["ObjRotation"]

    -- Bitte experimentieren Sie nicht mit den Quaternionenwerten. Wir verwenden hier die korrekte Drehung, das ist alles, was wir wollten.
  Ende
else
  -- Vorher wird eine Liste Objekte manuell gefüllt, indem man die Autos auswählt und diese in das Modul Liste Objekte einfügt.

  local t = $("Events").variables["Objects"]
  for i, Iter in ipairs(t) do
    -- Für jedes Objekt aus der Liste wird die gespeicherte Position/Drehung zurückgegeben.

    -- Sie befinden sich in Variablen MessedUpPosition/MessedupRotation.

    -- Gib einem Auto die vorherige Position und Drehung aus Variablen.
    -- Iter ist hier ein Auto aus der Iteration.
    -- Die Transformationsposition/Drehung eines Objekts wird auf die vorherige Position/Drehung dieses Objekts gesetzt, gespeichert in Variablen.
    -- Jedes Auto als vier Variablen: Tabellen mit der vorherigen Position und Rotation.

    -- Iter.transformation.position ist eine Tabelle mit den Werten von x, y, z.
    -- Iter.transformation.rotation ist eine Tabelle mit den Werten x,y,z und w als Quaternion.   Wir
```

```
brauchen nicht zu wissen, was eine Quaternion ist.
   -- Es ist nur eine Möglichkeit, die Drehung eines Objekts zu identifizieren.

   -- Wenn Sie transformation.position eingeben, erhalten Sie mehr Informationen.
   -- Wenn Sie transformation.rotation eingeben, erhalten Sie weitere Informationen.

   -- Setzt die aktuelle Position/Drehung auf die vorherige Position/Drehung, die in den Variablen
MessedUpPosition/MessedUpRotation des Objekts gespeichert ist.

   Iter.transformation.position = Iter.variables["MessedUpPosition"]
   Iter.transformation.rotation  = Iter.variables["MessedUpRotation"]


   -- Bitte experimentieren Sie nicht mit den Quaternionenwerten. Wir verwenden hier die korrekte
Drehung, das ist alles, was wir wollten.
  end
end
```

Signal Taste 10 is changed, saves the current position/rotation from some objects

```
if not deferredCall then
  if signal.state == 0 then
   -- Vorher wird eine Liste Objekte manuell gefüllt, indem man die Autos auswählt und diese in das
Modul Liste Objekte einfügt.

   local t = $("Events").variables["Objects"]
   for i, Iter in ipairs(t) do
     -- Für jedes Objekt seine Position/Drehung speichern.

     -- Lege sie in Variablen ab MessedUpPosition/MessedupRotation.

     -- Speichere für ein Objekt seine Position und Rotation in Variablen.
     -- Iter ist hier ein Objekt aus der Iteration.
     -- Jedes Objekt als vier Tabellen mit der vorherigen Position und Drehung in den Variablen.

     -- Iter.transformation.position ist eine Tabelle mit den Werten von x, y, z.
     -- Iter.transformation.rotation ist eine Tabelle mit den Werten x,y,z,w als Quaternion.  Wir
brauchen nicht zu wissen, was ein Quaternion ist.
     -- Es ist nur eine Möglichkeit, die Drehung eines Objekts zu identifizieren.

     -- Wenn Sie transformation.position eingeben, erhalten Sie weitere Informationen.
     -- Wenn Sie transformation.rotation eingeben, erhalten Sie weitere Informationen.

     -- Setzen Sie die Positions-/Drehungsvariablen des Objekts auf die in
Iter.transformation.position/rotation gefundenen Werte.
     -- Speichern Sie in den Variablen MessedUpPosition/MessedUpRotation.

     Iter.variables["MessedUpPosition"] = Iter.transformation.position
     Iter.variables["MessedUpRotation"] = Iter.transformation.rotation

     -- Bitte experimentieren Sie nicht mit den Quaternionenwerten. Wir verwenden hier die korrekte
Drehung, das ist alles, was wir wollten.
```

```
    end

    defer(0.5, "Deferred execution")
  end
elseif deferredCall == "Deferred execution" then
  signal.state = 1
end
```

TEST

```
--[[
Steuern Sie komplexe Abläufe Ihrer Anlage mit Hilfe der
Skriptsprache Lua:
  - Drücken Sie STRG + Leertaste, um eine Übersicht aller
    Funktionen und Variablen zu erhalten
  - Drücken Sie F1 zum Öffnen der Online-Hilfe mit weiteren
    Informationen und zusätzlichen Hilfestellungen
--]]
if not deferredCall then
  if signal.state == 0 then
    -- Vorher wird eine Liste Objekte manuell gefüllt, indem man die Objekte markiert und diese in das Modul Liste Objekte einfügt

    local t = $("Events").variables["Objects"]
    for i, Iter in ipairs(t) do
      -- Für jedes Objekt seine Position/Drehung speichern.

      -- Speichere sie in Variablen ObjRotation/ObjPosition.

      -- Speichere für ein Objekt seine Position und Rotation in Variablen.
      -- Iter ist hier ein Objekt aus der Iteration.
      -- Jedes Objekt hat vier Tabellen mit der vorherigen Position/Drehung in den Variablen.

      -- Iter.transformation.position ist eine Tabelle mit den Werten von x, y, z.
      -- Iter.transformation.rotation ist eine Tabelle mit den Werten x,y,z,w als Quaternion. Wir brauchen nicht zu wissen, was ein Quaternion ist.
      -- Es ist nur eine Möglichkeit, die Drehung eines Objekts zu identifizieren.

      -- Wenn Sie transformation.position eingeben, erhalten Sie weitere Informationen.
      -- Wenn Sie transformation.rotation eingeben, erhalten Sie weitere Informationen.

      -- Setzt die Positions-/Rotationsvariablen des Objekts auf die in Iter.transformation.position/rotation gefundenen Werte
      -- Gespeichert in den Variablen ObjPosition/ObjRotation des Objekts.

      Iter.variables["ObjPosition"] = Iter.transformation.position
      Iter.variables["ObjRotation"] = Iter.transformation.rotation

      -- Bitte experimentieren Sie nicht mit den Quaternionenwerten. Wir verwenden hier die korrekte Drehung, das ist alles, was wir wollten.
    end
```

```
    defer(0.5, "Deferred execution")
  end
elseif deferredCall == "Deferred execution" then
  signal.state = 1
end
```