POxL

Willkommen in der Welt der **POxL**. POxL ist ein Kunstbegriff, zusammengesetzt aus den Wörtern Pixel und Box. Pixel bewegen sich auf der Ebene und beinhalten Farbinformationen und Transparenz-Angaben, die in ihrer Gesamtheit ein Bild darstellen. POxL erheben sich in die dritte Ebene, so dass jedes Pixel zu einer kleinen Box wird. Diese Boxen haben Inhalte, die über die eingeschränkten Möglichkeiten von Pixeln weit hinausgehen.

Als Inhalt stelle man sich mehrere Bilder vor, die übereinander liegen und jedes für sich einen Zustand im Augenblick der Betrachtung darstellt. Ein Zustand beschreibt einen Status, der mit Bild, Ton, Bewegung und Fähigkeiten der Kommunikation ausgedrückt werden kann. Um diese Status zu überwachen und zu steuern, beinhaltet jedes POxL kleine Programme, die mit einer besonderen Programmiersprache geschrieben werden. Bezeichnen wir diese Sprache einmal mit POxL-Programming-Language, kurz: PPL. Über die PPL wirst du später in diesem Buch mehr erfahren.

Eine Besonderheit der POxL ist die Fähigkeit, mit anderen zu kommunizieren. Das bedeutet, sie haben Schnittstellen, mit denen sie untereinander verbunden werden können. Zudem kann ihnen eine 'Freiraum-Ausbreitung' in Form von Pings zugeordnet werden, oder aber auch eine 'Field'-Fähigkeit, um diese Ausbreitungen zu empfangen.

So wie viele Pixel zu einem Bild werden, werden mehrere Poxel zu einem Experiment.

Das Ziel

Ziel des Programms ist es, Gestaltungsräume zu schaffen, die durch die Kreativität der Nutzer in vielfältiger Weise ausgefüllt werden können – POxL-Welten. Als POxL ist alles vorstellbar, was sich bewegt oder in irgendeiner Abhängigkeit zu etwas Anderem steht. Diese Dinge müssen zeichnerisch gestaltet werden, was den Umgang mit Grafikprogrammen oder Bildverarbeitungsprogrammen fördert. Die Beziehungen der Dinge zueinander müssen definiert werden, was wiederum eine analytische Auseinandersetzung mit den Situationen erfordert und schließlich die Programmierung der Verhaltensweisen, was die logische Verarbeitung von Algorithmen beinhaltet. Dabei kommt es nicht auf das Erlernen einer HOL-Programmiersprache an, sondern nur auf die wesentlichen Schritte bei der Verarbeitung von Daten von einer Eingabe über die Prozessierung hin zur Ausgabe.

Das Buch

Was das Buch beschreibt

An Beispielen werden die Möglichkeiten der Erstellung und Gestaltung von POxLn aufgezeigt. Diese werden dann, wiederum an einem Beispiel aufgezeigt, zu einem Experiment zusammengestellt, das dann gestartet werden kann. Experimente allgemein können 'ablaufen', ähnlich einem Video.

Im letzten Teil des Buches ist die Programmiersprache für POxL beschrieben.

Die Bilder

Grundlagen für die POxL sind Bilder, welche die einzelnen Status dieser aufzeigen. Die einzelnen Bilder können mit jedem beliebigen Zeichenprogramm erstellt werden, das Bilder im PNG-Format mit Tranzparenz bearbeiten und speichern kann. Die Anzahl der Bilder für ein POxL ist beliebig, aus Gründen der Handhabbarkeit, wird jedoch empfohlen, diese auf das notwendige Maß zu reduzieren. Alle Bilder für ein POxL müssen die gleiche Größe haben, damit bei der späteren Ausführung keine Verschiebungen in der Darstellung sichtbar werden, es sei denn, sie sind gewollt.

Das Programm

Voraussetzungen

In diesem Abschnitt wird das Programm 'POxL *welten*' vorgestellt und die notwendigen Arbeitsschritte werden beschrieben. Das Programm kannst du im Internet unter -tbd- downloaden.

Das Programm ist in PHP geschrieben und du musst einen PHP-Server als Laufzeit-Umgebung auf deinem Rechner installiert haben, um das Programm nutzen zu können. Eine solche Umgebung findest du als meist kostenloses Programm im Internet und kannst dieses direkt auf deinen Rechner laden und installieren. Bei der Suche nach einem solchen Programm helfen dir die Suchwörter 'WAMP', 'LAMP' oder 'MAMP' aus einer Suchmaschine heraus.

Die Laufzeit-Umgebung bildet die Funktionen eines Host-Rechners auf deinem eigenen, lokalen Rechner ab und wird daher als Localhost bezeichnet. Sie enthält in der Regel einen Ordner 'www', der als Raum für den Server dient. In diesem Ordner legst du einen neuen Ordner für die POxL-Welten an, z.B. 'POxL' und kopierst den Inhalt des Download-Files 'poxelwelten.zip' in diesen Ordner. Vorher musst du das Paket natürlich entzippen.

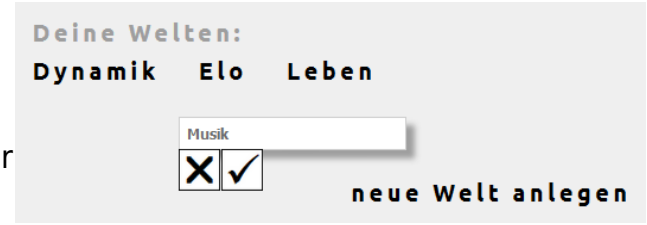
Nach Start des Servers wird dein Browserfenster geöffnet und das Programm dann unter 'POxL' (oder dem Namen den du für den Ordner bestimmt hast) angeboten. Einfach einen Doppelklick und schon geht’s los.

Wenn das Programm startet, solltest du das folgende Bild auf deinem Bildschirm haben.

Im Gastverzeichnis findest du vier Welten, in denen du dir alles genau anschauen kannst.

Programmbeispiel

Schritt 1: Anlegen einer neuen Welt

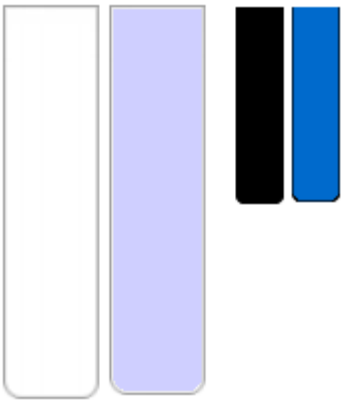


Mit einem Klick auf 'neue Welt anlegen' öffnet sich ein kleines Eingabepad, in dem du eine neue Welt bezeichnen und mit einem Klick auf ☑ anlegen kannst. Bei einer neuen Anmeldung wirst du die neue Welt dann in der Liste sehen.

Schritt 2: Bilder übernehmen

POxL leben von Bildern. Hast du dir Gedanken über deine neue Welt gemacht, entwickelst du am besten alle Bilder die du benötigst in einem beliebigen Ordner. Du kannst jedes Zeichenprogramm nehmen, das Bilder im PNG-Format bearbeiten und speichern kann. Wichtig ist, dass die Bilder eine Transparenz um das eigentliche Symbol haben, da dies eine Voraussetzung ist für die Annäherungserkennung von POxLn.

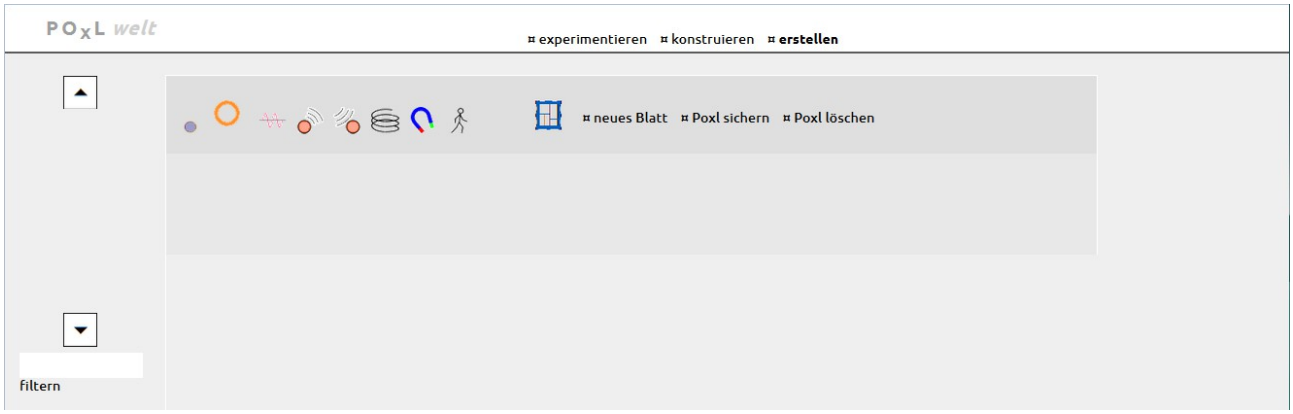
Die Bilder kannst du dann später bei der Erstellung von POxLn ins System hochladen.



Im Beispiel werden für eine Klaviatur vier Bilder benötigt, je zwei für die 'vollen' und die 'halben' Tasten. Bild 1 stellt jeweils den Ruhezustand und Bild 2 den gedrückten Zustand dar.

Sie dienen als Statusbilder für 2 POxL, die als Grundlage für eine Klaviatur als Experiment dienen sollen.

Die Bilder müssen für die kontrollierte Nutzung vom Programm aus in einem festgelegten Ordner 'uploads' liegen. Dies ist wichtig, da das Programm auch auf einem Server im Netz liegen kann und dann nur die Möglichkeit besteht, auf einen untergeordneten Ordner des Programms zugreifen zu können. Alle erforderlichen Bilder müssen somit 'hochgeladen' werden. Bei der Erstellung der POxL kann dies über einen besonderen Knopf erfolgen, mit dem du in die Bildauswahl gelangst. Dort kannst du auch deine Bilder aus dem Erstellungsordner in den Ordner 'uploads' laden. Zuerst also in den Bereich 'erstellen' wechseln.



Dann mit einem Klick auf den Bild-Knopf in der Mitte oben drücken, um die Bildauswahl zu öffnen. Zuerst wählst du 'Durchsuchen', um dann dein Bild in deinem Erstellungsordner auszuwählen; der Filename wird dir rechts neben dem Knopf zur Kontrolle angezeigt. Anschließend drückst auf 'hochladen' und der Filename erscheint neben der Taste. Jetzt kannst du den nächsten File wählen.

In unseren Beispiel wären das die vier Einzelbilder für die Tasten.

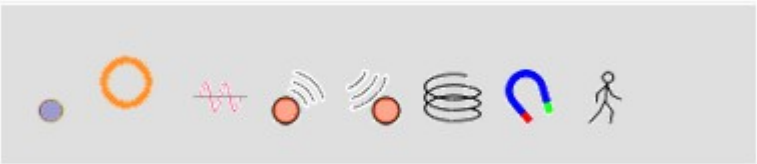
Bei der POxL-Erstellung kannst du dann über die Auswahlliste auf die Bilder zugreifen.

Schritt 3: POxL erstellen

POxL werden nur unter dem Menüpunkt 'erstellen' für den späteren, allgemeinen Gebrauch aufgebaut und gepflegt. Der Weg zu einem POxL ist einfach und immer der gleiche:

- Zuerst wählst du ein Bild aus deinem Kontingent. Fehl das Bild, wird immer ein Default-Bild als Grundlage genommen und du musst es dann später berichtigen. Welches Bild aktuell an oberster Stelle steht siehst du, wenn du die Bildauswahl öffnest.
- Ein neues Blatt anlegen mit einem Klick auf 'neues Blatt'. Dein POxL wird dann in der Auswahlliste links erscheinen.
- Für die weiteren Eingaben zu deinem POxL wählst du es aus der Liste erneut aus.
- Jetzt fügst du die weiteren Bilder für die einzelnen Status des POxLs hinzu.
- Dann erfolgt die Ergänzung mit den Eigenschaften, die nachfolgend für unser Beispiel aufgezeigt sind.

Ein POxL kann unterschiedliche Eigenschaften haben, die in der Kopfleiste zur Auswahl angeboten werden.

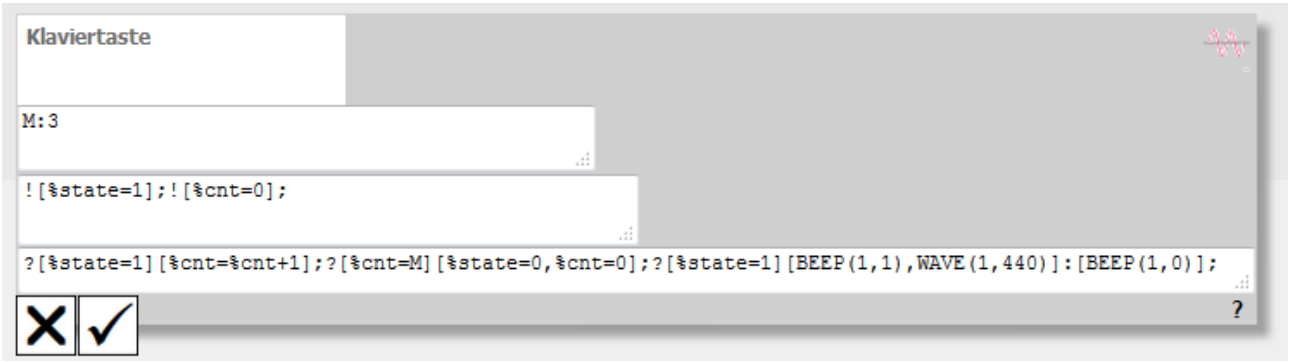


Du klickst auf ein Symbol und die entsprechende Eigenschaft wird dem POxL zugeordnet. Die beiden Punkte links sind die wichtigsten.

Der erste Punkt (Verbinder) bildet eine Schnittstelle zum POxL und kann mehrfach vorhanden sein. Jede einzelne Schnittstelle wird individuell gesetzt und als Input oder Output definiert. An dieser Stelle ist das aber noch nicht so wichtig, die endgültige Bestimmung erfährt eine Schnittstelle erst bei der Erstellung eines Experiments.

Punkt zwei stellt einen Sensor dar. Es können mehrere Sensoren auf ein POxL gelegt werden, die im POxL-Programm später zum Einsatz kommen. Das Beispiel der Klaviertasten zeigt einen Sensorpunkt , um die Taste im Experiment auch spielen zu können. Mit einem Klick auf den angelegten Sensorpunkt öffnet sich ein Eingabepad, in dem du die Position des Sensors auf dem POxL bestimmen kannst.

Klicks du auf die Markierung öffnet sich ein besonders Pad: Das POxL-Pad.



Hier bestimmst du das Leben eines POxLs.

Die Felder zeigen oben den Namen des POxLs. Wähle hier einen eindeutigen, aussagekräftigen Begriff, unter dem du später dein POxL wiederfinden möchtest.

Darunter ein Feld für die Deklaration von Variablen, die du in den Programmen verwenden möchtest. Nur die hier deklarierten Variablen stehen später für die Programme zur Verfügung.

Im dritten Feld steht das Programm für ein Event, also einen Klick auf einen Sensor. In unserem Beispiel wird bei einem Tastendruck der Status auf 1 gesetzt und ein Zähler auf 0.

Im untersten Feld steht das Programm, das bei der Ausführung eines Experiments periodisch aufgerufen wird. Im Beispiel für eine Taste lautet dies: Ist der Status 1, erhöhe den Zähler. Hat der Zähler den festgelegten Wert M, setze den Status auf 0 und lösche den Zähler. Ist der Status 1 aktiviere den Oszillator und setze die Frequenz auf 440, sonst deaktiviere ihn.

In der Ecke oben rechts siehst du alle Eigenschaften, die dem POxL zugeordnet wurden. Für die Taste hier im Beispiel ist es ein Oszillator.

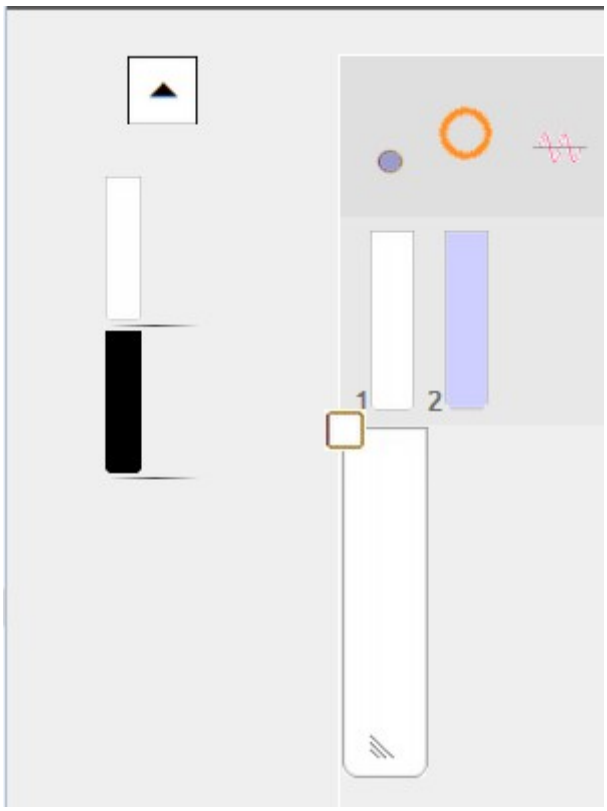
Über die Programmierung erfährst du mehr im Kapitel über die PPL. Hier werden auch die Funktionen wie BEEP und WAVE erklärt.

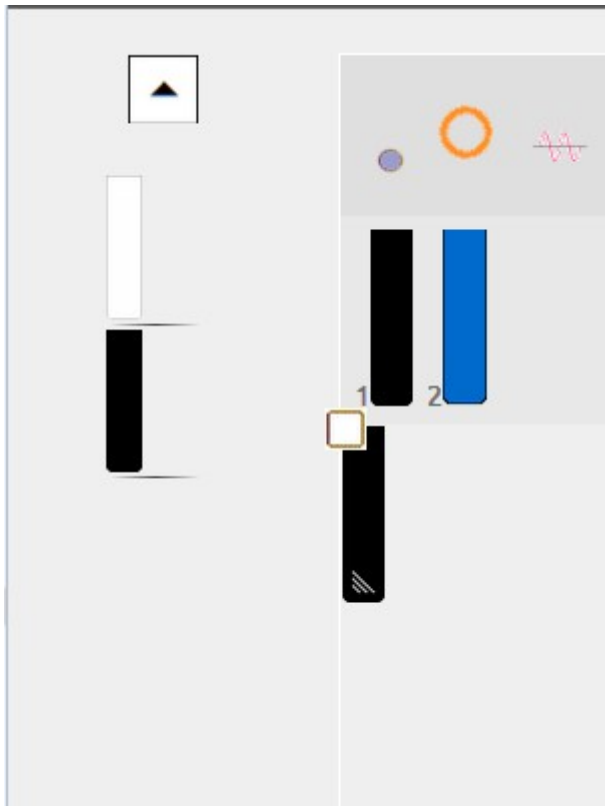
Aus der Liste der Eigenschaften oben fehlen noch der Ping als Sender oder Emitter und sein Gegenstück der Empfänger oder Blip, der für eine Feld-Funktion steht.

Dann kommt eine Abstoß- oder Bump-Eigenschaft sowie das Gegenstück eine Anziehungs- oder Magnet-Eigenschaft.

Die letzte Eigenschaft ist der Walk oder auch Move. Mit ihr kannst du ein POxL in Bewegung versetzen.

Wichtig: Die POxL werden auf dem Server gehalten, also irgendwo im Netz. Um Änderungen dauerhaft zu erhalten, müssen diese immer gesichert werden!





Das Ergebnis ist ein POxL 'Klaviertaste' für die vollen Töne. Nach dem gleichen Muster bauen wir nun noch ein neues POxL für die halben Töne, so dass unsere PoxL-Liste zwei Einträge zur Auswahl hat: Links die Taste für die vollen und rechts die Taste für die halben Töne.

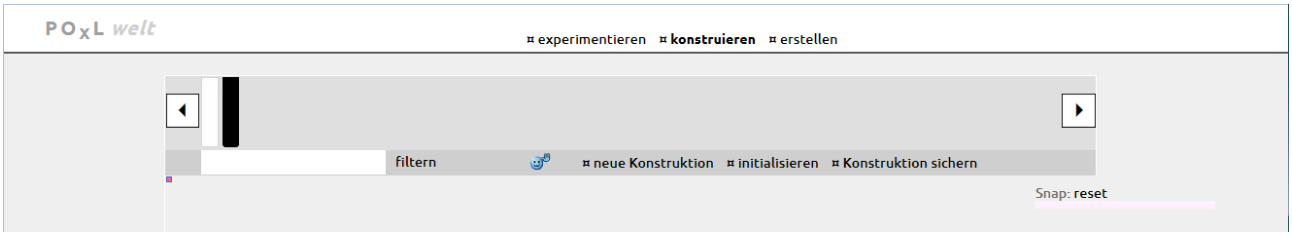
Damit wären die Grundlagen geschaffen für unser erstes Experiment, ein Tasten-Panel. Hierzu wechseln wir ins Arbeitsfeld 'konstruieren'.

Schritt 4: Experiment konstruieren

Ein Experiment besteht aus einem oder mehreren POxLn, die unabhängig voneinander agieren, oder über Schnittstellen miteinander kommunizieren und sich gegenseitig beeinflussen.

In unserem Beispiel sind die POxL eigenständig, bilden aber in ihrer Gesamtheit wiederum ein Ganzes, ein Tasten-Panel.

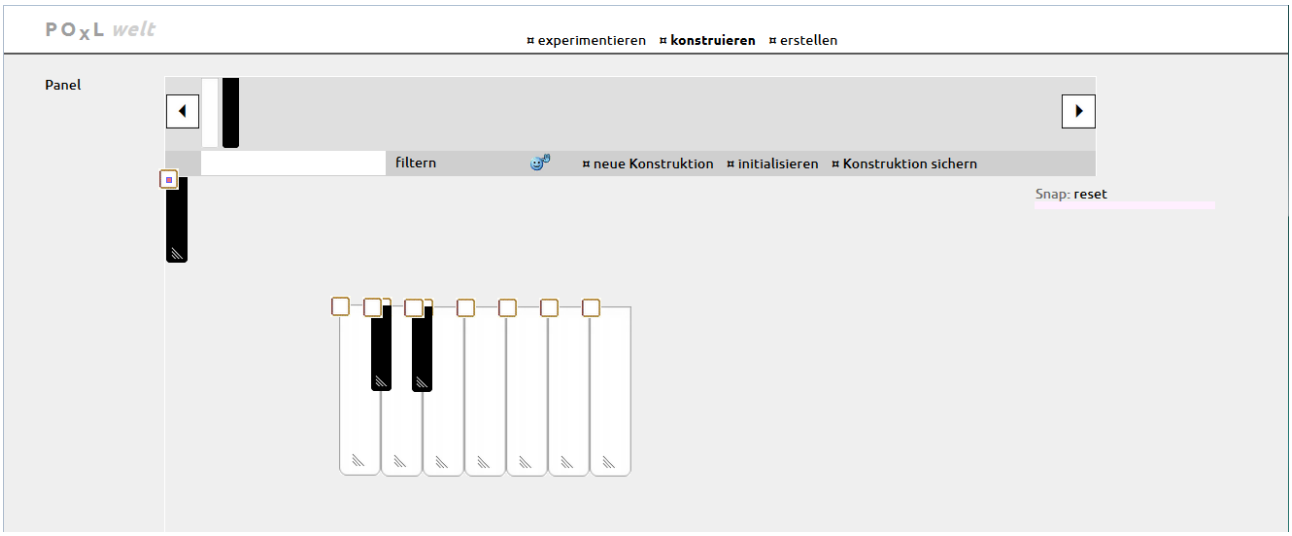
In der Konstruktion wird aus dem Pool der POxL ein Experiment aufgebaut und die POxL in ihren Eigenschaften so geändert, dass sie im Experiment miteinander funktionieren.



Im oberen Bereich werden alle POxL aus der gewählten Welt für die Auswahl angeboten. In unserem Beispiel sind dies die Tasten für volle und halbe Töne.

Wir starten mit einem Klick auf 'neue Konstruktion', wodurch sich ein Eingabepad zeigt, in das wir einen Namen für unser Experiment eingeben. Hier muss ein kurzes, aussagekräftiges Wort eingetragen werden, das später in der Auswahlliste der Experimente erscheint.

Um das Tastenpanel aufzubauen, müssen nun zuerst die vollen und dann die halben Töne eingebaut werden. Dies ist in diesem Fall wichtig, damit die halben, schwarzen Tasten auf den vollen, weißen liegen.

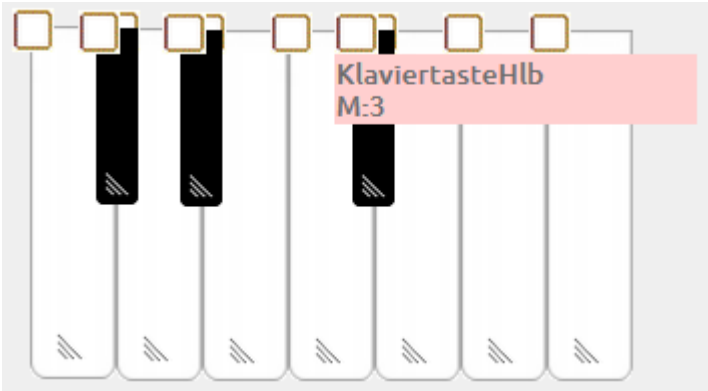


Im Beispiel sind die weißen Tasten bereits komplett aufgebaut und bereits zwei schwarze Tasten gesetzt. Um das Panel zu vervollständigen wird jetzt der Weg zur nächsten Taste beschrieben.

Um eine Taste auf dem Feld zu erzeugen, also ein neues POxL einzufügen, klickst du auf das POxL in der Auswahl oben. Im Feld wird dann das POxL oben links in der Ecke dargestellt und steht für weitere Aktivitäten bereit.

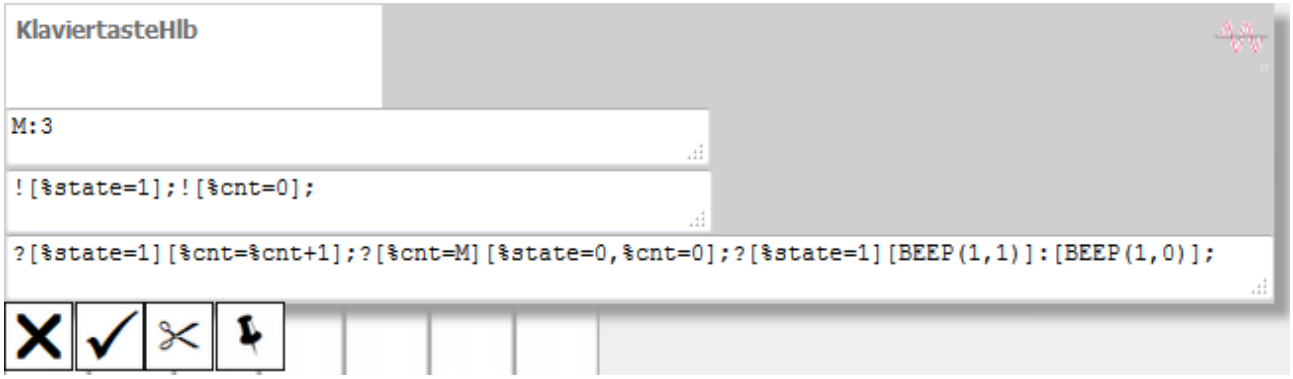
Der erste Schritt ist jetzt die Positionierung, damit das POxL seinen Arbeitsplatz

bekommt und du die Übersicht behältst. Nach Änderungen musst du die Konstruktion immer wieder sichern, damit deine neuen Einträge auch auf dem Server gespeichert werden!



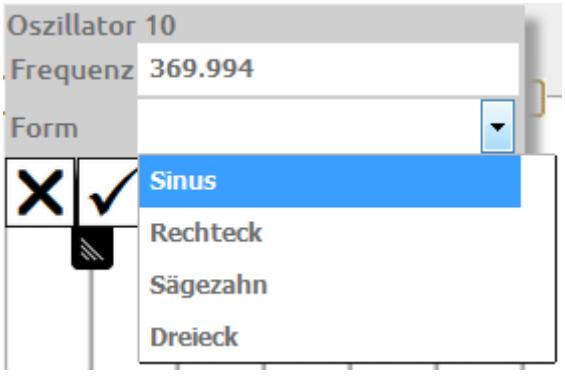
Hat das POxL seine gesicherte Position, kannst du an sein Innenleben gehen. Mit einem Klick auf das Kontrollkästchen öffnet sich das Bearbeitungspad. Die Grundeinstellungen sind aus der POxL-Erstellung bereits vorhanden und hier kannst du jetzt alle Anpassungen vornehmen, dass sich das POxL in das Experiment einfügt und entsprechend richtig arbeitet.

In unserem Beispiel sind an den Programmen keine Änderungen erforderlich, da die POxL in ihren Grundeinstellungen bereits so eingerichtet wurden, wie im folgenden Eingabepad zu sehen.



Möchtest du die Spieldauer noch verlängern, änderst du einfach die Zählgrenze oben M:3.

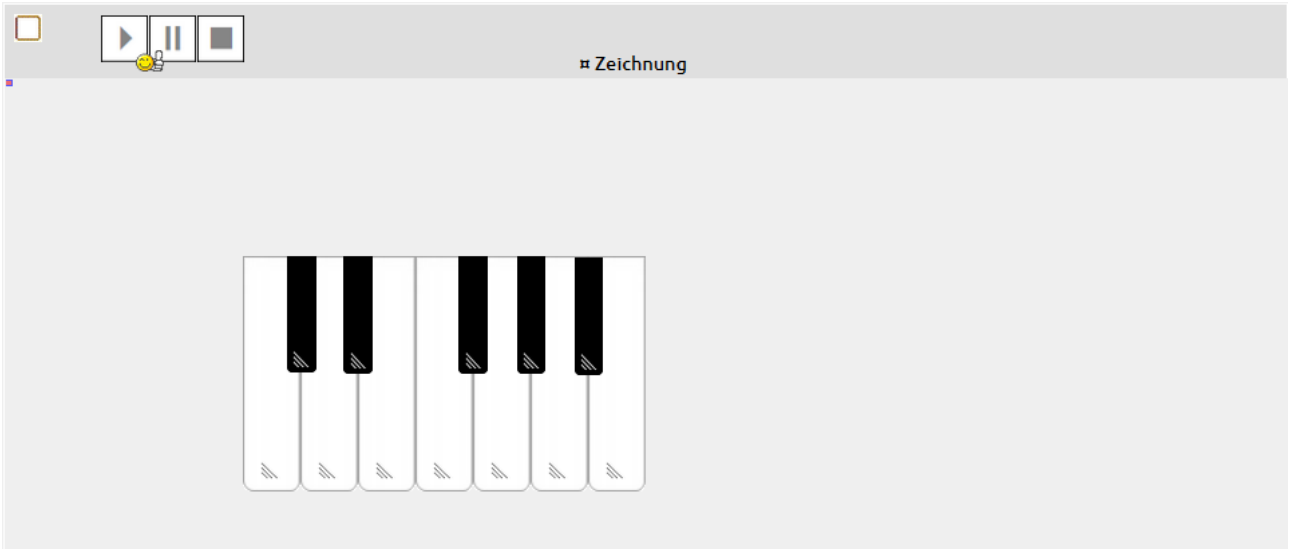
Was aber unbedingt erfolgen muss, ist die Anpassung des Oszillators an die Tastenposition im Panel. Mit einem Klick auf das Oszillator-Symbol oben rechts erscheint hier wieder das entsprechende Pad, um die Eingaben vornehmen zu können. Hier kannst du jetzt die Frequenz eingeben und die Hüllform wählen.



Jetzt bleibt nur noch die Taste in der Position zu fixieren und das erreichst du mit einem einfachen Klick auf den Pin-Knopf (siehe zuvor).

Nicht vergessen, die Konstruktion zu sichern!

Auf die gleiche Weise müssen jetzt noch die beiden fehlenden Tasten ergänzt werden und unser Beispiel ist fertig.



Das Experiment steht nun zur Nutzung bereit. Dabei handelt es sich um ein einfaches Konstrukt, das aber bewusst so gewählt wurde, um die Wege aufzuzeigen. Die Beispiele in den verschiedenen Welten zeigen dir viele Möglichkeiten und geben dir alle Freiheiten diese zu erforschen.

Eigenschaften

Die Eigenschaften, die ein POxL haben kann, sind vom Programm vorgegeben und werden entsprechend auch vom Programm behandelt. Die Inhalte der Eigenschaften werden aber vom POxL-Programm festgelegt und können in den unterschiedlichen Situationen auch unterschiedlich bewertet und behandelt werden.

Verdeutlichen lässt sich dies am Beispiel eines Pings, der einen Emitter darstellt. Die Eigenschaft ist ein Ping, was ein POxL aber aussendet ist ausschließlich von der Bewertung auf der Empfängerseite abhängig. Es kann eine konkreter Wert sein, es kann ein Signal als Schalter sein oder ein Licht, wie im Beispiel der Fotozelle im Experiment Elo'ABC.

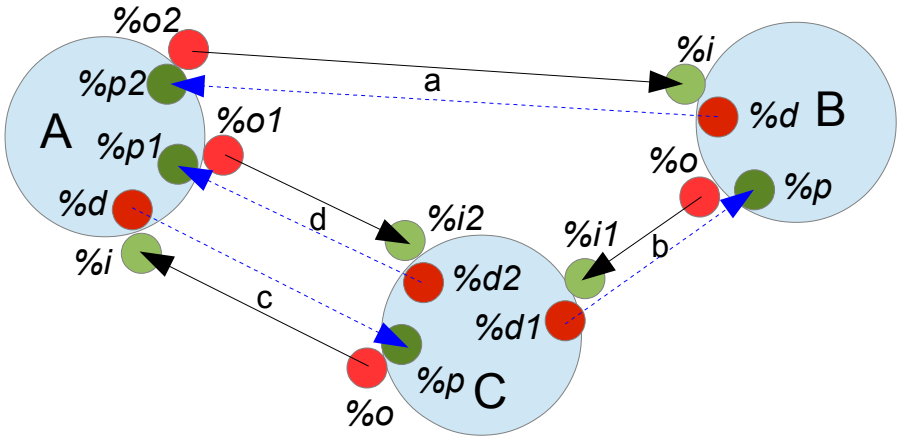
Die Liste der Eigenschaften:

- Ein **Schnittstellenpunkt**. Schnittstellenpunkte dienen als Ein- bzw. Ausgabepunkte für ein POxL und können mehrfach vorkommen. Sie müssen dann mit einer laufenden Nummer ergänzt werden. Eingabepunkte werden mit `%i` und Ausgabepunkte mit `%o` gekennzeichnet, die so direkt in den POxL-Programmen gesetzt und gelesen werden können. Bezeichnen wir eine Verbindung zwischen einem Eingabepunkt und einem Ausgabepunkt als Kanal, so ist der Datenfluss von einem Ausgabepunkt zu einem Eingabepunkt, also von %o nach %i. Automatisch wird für einen Kanal immer ein Duplexer angelegt, der es ermöglicht, Informationen als Ergebnis einer Aktion in die Gegenrichtung zu senden. Die Ansprache in den Programmen ist mit %d für einen Duplex und %p für einen Polling-Point möglich. Die Nummerierung ist dann immer die gleiche wie bei den zugehörigen Datenkanälen. Dem Eingabepunkt %i1 ist z.B. immer der Duplexer %d1 zugeordnet und dem Ausgabepunkt %o4 immer der Polling-Speicher %p4.

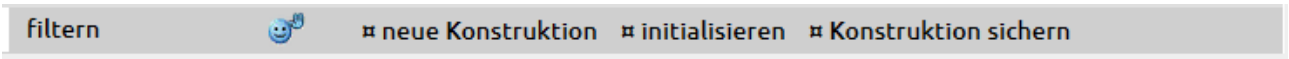
Als Beispiel dient das Experiment Elo'Duplextest, in dem drei Schalter auf 'ein' stehen

müssen, damit die Lampe leuchtet. Jeder einzelne Schalter stellt nur durch; die Information, dass alle drei Schalter eingeschaltet sind, wird über den Duplex ermittelt.

Die nachfolgende Skizze stellt ein Beispiel einer Gruppe von 3 POxLn (A,B und C) und deren Verbindungen vor. Die schwarzen Linien/Pfeile stellen die einzelnen Kanäle (a,b,c und d) dar mit ihren Duplexern, den blauen, gestrichelten Linien/Pfeilen. Die Bezeichner (i,o,d und p) der Schnittstellenpunkte zeigen hier auch die Adressaten für die Programmierung in diesem Fall.



Bei der Konstruktion eines Experiments müssen alle Schnittstellenpunkte nach dem Einbringen der POxL neu durchnummeriert werden, um die Verbindungen der Punkte adressieren zu können. Dies geschieht mit einem einfachen Klick auf 'initialisieren'.



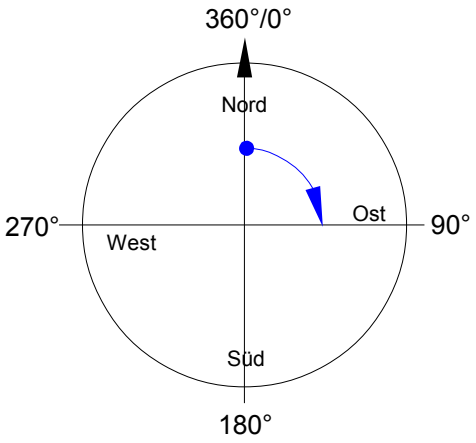
Ein **Sensorpunkt**. Diese dienen als Triggerpunkte für die Bedienung eines POxLs mit der Maus. Beim Überstreichen mit dem Mauszeiger wird zur Kenntlichmachung ein Ring ○ über den Punkt gelegt, der Trigger wird dann mit einem Mausklick ausgelöst. Du kannst einen Sensor aber auch als Berührungssensor definieren, dann wird dieser mit einem ausgefüllten Kreis ● markiert und beim Überstreichen ausgelöst. Sensorpunkte können mehrfach vorkommen, ihre Kennung ist `%e`. Die Definition erfolgt über den Typ 0:Klick, 1:Touch.

Ein **Oszillator**. Oszillatoren erzeugen einen Ton mit einer programmierbaren Frequenz und wählbaren Hüllkurve. Über das Programm lässt sich die Frequenz einstellen und die Oszilatoren beliebig ein- bzw. ausschalten.

An Hüllkurven stehen Sinus, Rechteck, Dreieck und Sägezahn zur Auswahl.

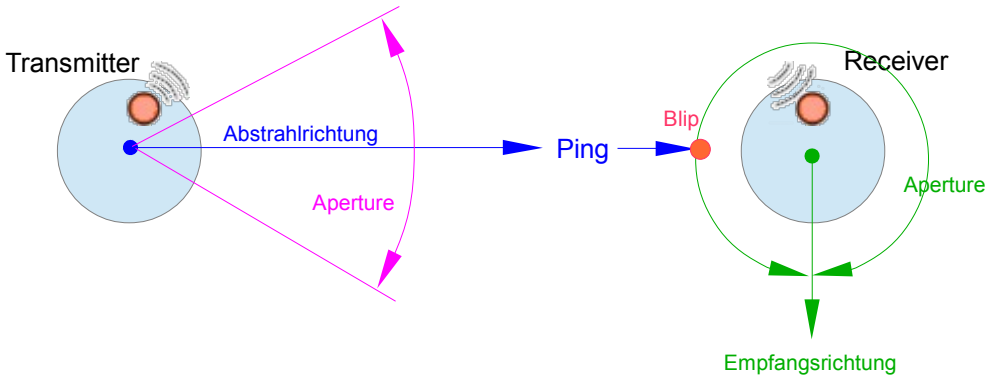
POxL können sich bewegen

Jedem POxL kann die Eigenschaft 'Walk' zugeordnet werden. Damit ist es möglich, einem POxL Kurs und Fahrt zu geben, und es auf dem Schirm in Bewegung zu setzen. Der Kurs ist rechtweisend, d.h. 0° ist immer oben und die Gradangabe erfolgt im Uhrzeigersinn.



POxL können über Freiraum kommunizieren

Für die Kommunikation der POxL untereinander lassen sich diesen Transmitter (Sender) und Receiver (Empfänger) zuordnen. Die Transmitter strahlen Pings aus, die von den Receivern anderer POxL empfangen werden können. Diese Funktion ist eine Feld-Funktion des POxLs und führt dort zu einem Blip.



Transmitter und Receiver haben eine Abstrahl- bzw. Empfangsrichtung und auch jeweils einen Öffnungswinkel (Aperture). Bei einem Rundum-Feld beträgt dieser 360°. Die Funktionen sind PING() für eine Abstrahlung und LOOK() für einen Empfang.

POxL können sich berühren

Bewegen sich POxL in der Ebene, können andere POxL mit Berührungssensoren ausgestattet werden, die eine solche wahrnehmen und bearbeiten können. Ein Berührungssensor wird durch eine Feder gekennzeichnet. Eine Berührung ist ein Bump.

Die Umkehr ist eine magnetische Wirkung, die mit einem Magneten gekennzeichnet ist und ein POxL anzieht.

Hintergrundbilder

Um Experimente in eine, dem Thema angepasste Umgebung zu setzten, können dem Experiment verschiedene Hintergrundbilder zugeordnet werden.

Mit einem Klick auf das Pixel-Feld (Mitte oben) auf der Konstruktionsseite, wird ein Pad geöffnet, um Bilder auszuwählen und den Mode zu bestimmen, wie das Bild verarbeitet werden soll. So lässt sich wie im Beispiel Leben'segeln, mit einer kleinen Wellen-Kachel ein ganzer See erzeugen. Die Bilder werden rückwärts gestaffelt, somit muss das unterste Bild als letztes eingegeben werden.

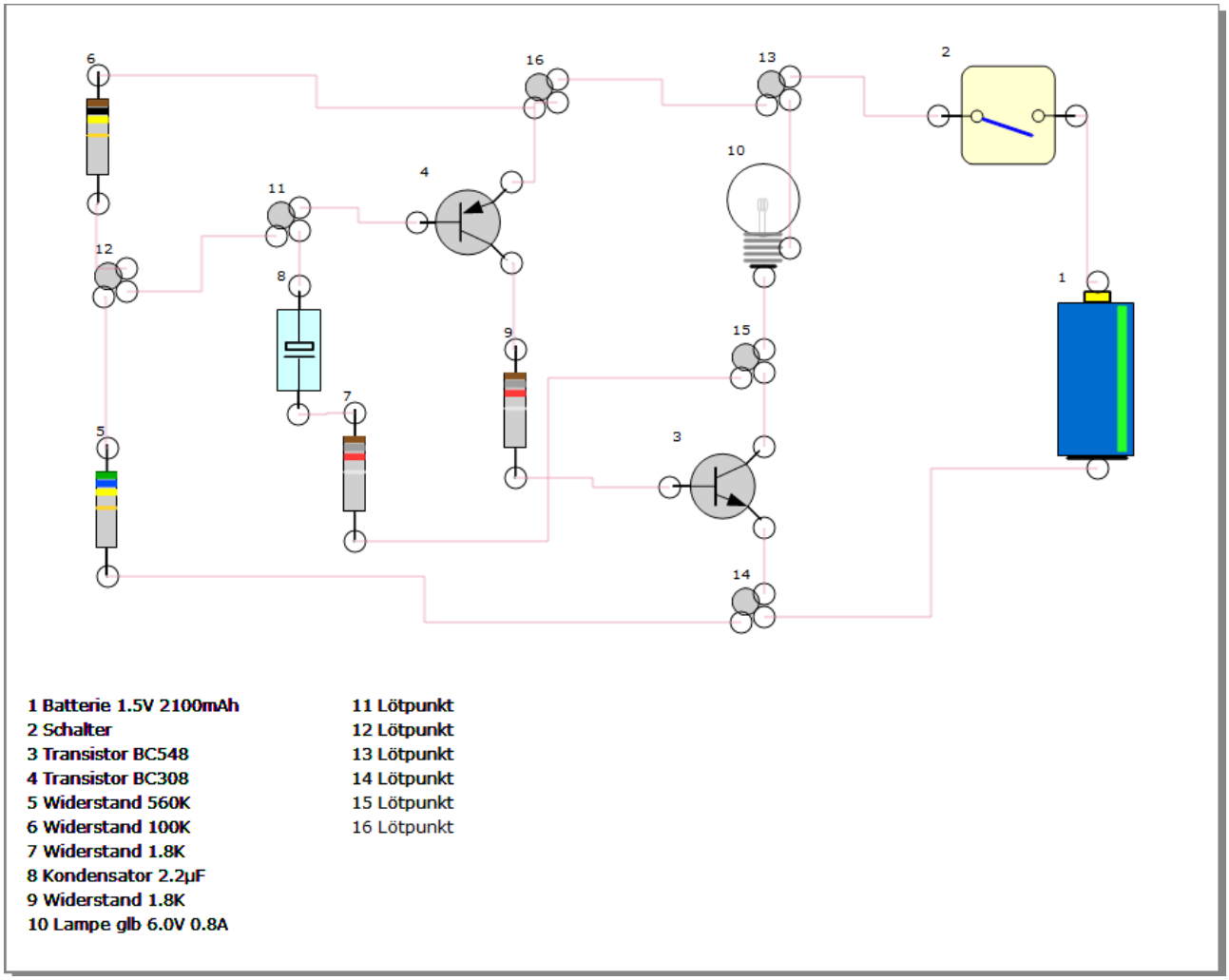
Das Icon daneben ermöglicht dir in der Konstruktion bei Sensorpunkten von der Ausführung in die Editierung zu wechseln.

Zeichnungen

Von allen Experimenten lassen sich mit einem einfachen Klick Zeichnungen im SVG-Format erstellen, die in einem Browser dargestellt werden können und mit der jeweiligen Druckfunktion (Druckvorschau mit Rand und Titeleinstellugen) entsprechend ausdrucken lassen.

Bei Experimenten aus dem elektronischen Bereich können so Schaltungen gedruckt werden, die dann durch echte Hardware realisiert werden können. Haben die POxL entsprechend Namen, die die Bauteile beschreiben, erhält man auch direkt eine exakte Stückliste auf der Zeichnung.

Die nachfolgende Zeichnung ist ein Abdruck des Experiments Elo'Blinklicht mit einer kompletten Stückliste, so dass sich das Experiment auch als echte Schaltung bauen lässt.



Experimente als Gast

POxL *welten* hat einen Gast-Zugang mit einigen Experimenten, die als Beispiele für die Anwendung der verschiedenen Funktionen von POxLn dienen sollen. Nachfolgend sind die Experimente aufgeführt, mit Hinweisen auf Besonderheiten.

Die interessanteste Welt dürfte die Elektronik (Elo) darstellen, in der einige Experimente zum Verständnis und der Funktionsweise von Schaltungen beitragen. Hier wird auch die Möglichkeit der Zeichnungserstellung bedeutsam.

Elo'ABC

Hier sind verschieden kleine Schaltkreise aufgezeigt, die Töne auf verschiedene Weise erzeugen. Im Fall oben links, erfolgt die Steuerung über einen Lichtsensor, der je nach Einfallswinkel und Abstand das Signal ertönen lässt. Bewegt man die Lampe entsprechend, ist dies gut zu erkennen.

Elo'Blinklicht

Diese Schaltung ist eine echte Schaltung, die auch durch entsprechende Hardware nachgebaut werden kann. Das Experiment soll lediglich die Funktionsweise der Schaltung aufzeigen. Die POxL haben hier die korrekten Bezeichner, so dass mit der Zeichnung ein echter Schaltplan entsteht.

Elo'Duplextest

Duplexer sind ein integraler Bestandteil der Kommunikation zwischen zwei PoxL-Anschlüssen. Die Funktionsweise der Duplexer soll durch dieses Experiment aufgezeigt werden. Jeder einzelne Schalter kann ein- bzw. ausgeschaltet werden und gibt dementsprechend den Durchgang frei oder sperrt ihn. Erst durch die Rückmeldung der einzelnen Schalterstände kann die Gesamtheit des Durchgangs festgestellt werden und die Lampe leuchtet nur, wenn alle drei Schalter eingeschaltet sind.

Elo'ODER

Das Experiment zeigt die Wirkungsweise einer ODER bzw. NICHT-ODER-Schaltung.

Elo'UND

Das Experiment zeigt die Wirkungsweise einer UND bzw. NICHT-UND-Schaltung.

Elo'messen

Bei diesem Experiment soll die Möglichkeit einer Displaysteuerung gezeigt werden und der Einfluss von Veränderungen in den Widerständen einer Schaltung.

Elo'Stromkreis

Ein einfaches Experiment zur Erläuterung eines Stromkreises.

Elo'Wechselschalter

Ein einfaches Experiment zur Erläuterung eines Wechselschalters.

Leben'segeln

Dieses Experiment soll die Möglichkeiten der erweiterten Kommunikation zeigen.

Das Experiment zeigt ein Segelboot und die Möglichkeiten der Steuerung über eine Pinne und zwei Schoten. Auf der rechten Seite gibt es ein Icon für den Wind, der in den Stärken 0 bis 2 geschaltet werden kann.

Die Schoten BB und StB geben bei Bedienung der Sensoren jeweils einen PING ab, der sowohl beim Boot, als auch bei der jeweils anderen Schot durch einen LOOK erfasste wird und entsprechend reagiert. Die Pinne meldet ebenfalls einen PING vom Typ 'Sensor', der aber nur auf das Boot wirkt und dort ausgewertet wird. Das Boot selbst hat einen Sender (PING) vom Typ 'zeigen', der sich als Segel darstellt. Die Sensoren werden mit ihren Indices aus der Sensorliste gemeldet und können somit nach einem LOOK gezielt bedient werden.

Leben'Uhrzeit

Dieses Experiment soll die Möglichkeit vom Ein- bzw. Ausschalten einzelner POxL zeigen. Beim Überfahren der einzelnen Bereiche des Zifferblatts werden die sprachlich gebräuchlichen Zeitansagen durch zusätzliche POxL dargestellt.

Leben'Obstgarten

Das Experiment zeigt verschieden POxL als Obst, die auf dem Schirm beliebig verschoben werden können. Der kleine Roboter überfährt die Rasenfläche und scannt nach Obst. Kommt eine Frucht in seinen Erfassungsbereich, wird der Name in seinem Display dargestellt. Bei der Erfassung wird jeweils das nächstliegende Früchtchen beachtet.

Dynamic'Ball

Hier bewegt sich ein Ball zwichen vier Wänden, die sich beliebig verschieben lassen. Der Ball prüft ständig eine Annäherung und spring dann bei Brührung einer Wand zurück. Die Aufprallkraft wird bei jeder Berührung gemindert, bis der Ball irgendwann stillsteht.

Dynamic'Ball2

Der Ball fällt bei diesem Experiment auf den Boden, weil er von diesem angezogen wird. Der Boden besitzt hier die Magnet-Eigenschaft. Im Zusammenspiel zwischen Ball und Boden nimmt die Aufprallkraft stetig ab bis der Ball zur Ruhe kommt.

Musik'Panel

In diesem Experiment ist ein Klavierpanel gezeigt, bei dem jede Taste, volle und halbe Töne, ein eigenständiges POxL ist, die alle mit einem Oszilator ausgestattet sind. Die Besonderheit ist hierbei, dass die POxL fixiert sind und sich nicht, wie üblich, begegen lassen.

Musik'Twinkel

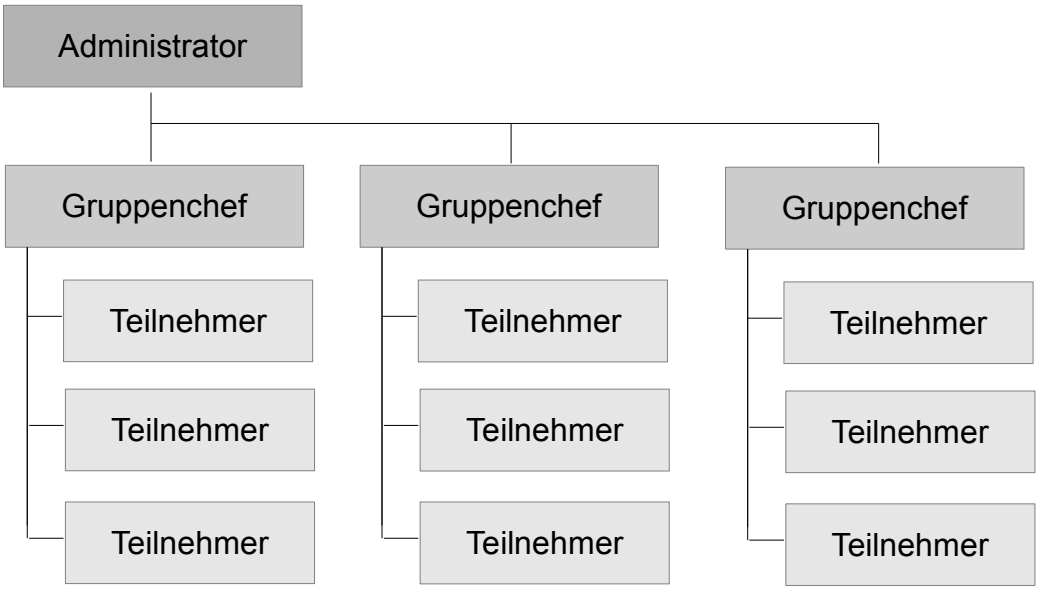
Ein einfaches Experiment, das die Anwendung von Zufallszahlen, Tönen und Positionierungen zeigen soll.

Administration

Nutzergruppen

POxL *welten* hat einen administrativen Zweig, um einzelne Nutzer und Nutzergruppen zu erstellen und zu verwalten. Mit einem Klick auf 'Tutor' in der Kopzeile der Einstiegsseite komm man in diesen Bereich.

Hier ist es erforderlich, sich als Administrator anzumelden, da für den Zugang besondere Rechte erforderlich sind. Die Rechte sind hirarchisch angeordnet und für den

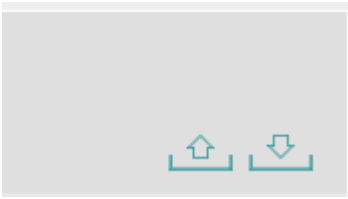


Einstieg braucht man Admin- bzw. Gruppenchef-Rechte (Tutor/Manager).

Ein Administrator kann Gruppenchefs einrichten und verwalten. Gruppenchefs wiederum können sich eine Gruppe mit Teilnehmern zusammenstellen und diese managen. So lassen sich Ausbildungsgruppen oder auch Klassen aufbauen, die als Grundlage für die Verteilung (Import / Export) von POxLn und/oder Experimenten dienen. Gruppenmitglieder können untereinander Daten tauschen.

Nutzer

Jeder User hat einen Briefkasten für Import und Export und eine gesonderte Ablage für den Transfer von Daten zwischen seinen Welten.



Bei einem Klick auf den jeweiligen Knopf für Import (rechts) und Export (links) öffnet sich jeweils ein entsprechendes Pad mit den angepassten Möglichkeiten eines Transfers.

Backup und Retrieve

Jeder Nutzer kann seine Welten in einem Backup-Ordner sichern und wieder zurückholen. Dies ist wichtig, falls bei der Erstellung von POxL oder Experimenten etwas schief geht. Darum sollte vor einem Backup auch immer geprüft werden, ob alle Experimente funktionieren. Nur dann besteht auch die Sicherheit, im Fehlerfall auch wieder funktionierende Experimente zurückzuholen.

Die Programmiersprache für POxL

Diese Programmiersprache ist speziell für die Schnittstellen und die Statuskontrolle von POxLn konzipiert und sehr einfach. Es gibt wenige Bedingungen aber besondere Vorgaben, die für die richtige Programmausführung beachtet werden müssen.

Die Programme werden ohne Whitespaces (Leerzeichen, Tabs usw.) in einer Endloszeile (ohne Zeilenumbruch) geschrieben. Es besteht aus einzelnen Statements, die jeweils durch ein Semikolon voneinander getrennt sind.

Es gibt drei Arten von Statements: Anweisungen, Fragen und Switches. Jedes Statement wird durch einen Kenner markiert und in eckige Klammern eingebettet. Fragen und Switches enthalten im Statement alle zugehörigen Anweisungen. Anweisungen können in einer Pipe, also hintereinander, liegen und werden dann durch Hochkommata voneinander getrennt.

Fragen können ein Else-Statement haben, es wird mit einem : (Doppelpunkt) markiert.

Bei Switches wird der Case-Wert (einstellig) direkt vor die Anweisung gesetzt. Der Switch kann auch ein Default-Statement haben, es trägt den Kenner #. Mit einem SKIP lassen sich Statements überspringen.

Die allgemeine Form:

Anweisungen:	![s's];
Fragen:	?[t][s]:[s];
Switches:	*[t]c[s]c[s]c[s]#[s];
Sprünge:	SKIP (n);
	Legende: s – Statement, t – Term, c – Case, n - Nummer

Variablen: Freie Variablen werden durch einfache, große Buchstaben definiert A, B, C ... Z und mit einem Initialwert versehen: A:0,P:3.14

Die Wertezuweisung im Programm erfolgt durch eine einfache Anweisung: A=4, Z=0.33.

Reservierte Variablen haben als Kenner immer ein %. Die Bezeichner sind reserviert und werden immer klein geschrieben. Es gibt mehrere Arten: Input %i, Output %o, Event %e, Status %state und die Zeiteinheit %timetic. Zudem die Duplex-Variablen %d und %p.

Input und Output können mehrfach vorkommen, sie werden dann durch eine laufende Nummer unterschieden: %i, %i1, %i2, ..., %o, %o1, %o2, ... usw.

Beispiele:

Anweisungen und Fragen:

```
![C=C-%i];![%o=C];?[C<1.7][%state=1];?[C<1.3][%state=2];?[C<0.9]
[%state=3];?[C<0.5][%state=4];
```

Frage mit Else:

```
?[%state=0][%state=%e]:[%state=0];
```

Switch mit zwei Cases und einem Default-Statement:

```
*[%state]2[%o1=%i'%o2=0]1[%o1=0'%o2=%i]#[%o1=0'%o2=0];
```

Funktionen

Die PPL bietet einige Funktionen, die auf die Bedürfnisse der POxL abgestimmt sind und die Kommunikation der POxL untereinander ermöglichen. Die Bezeichner haben immer vier Buchstaben.

Mathematische Funktionen:

ABSV Liefert den Absolutwert des übergebenen Wertes.
Form: E=ABSV(R)
Param: Rationaler Wert.

ACOS Gibt den Winkel zum Cosinuswert zurück.
Form: E=ACOS(C)
Param: Cosinus des Winkels in Grad.

ASIN Gibt den Winkel zum Sinuswert zurück.
Form: E=ASIN(S)
Param: Sinus des Winkels in Grad.

ATAN Gibt den Winkel zum Tangenswert zurück.
Form: E=ATAN(T)
Param: Tangens des Winkels in Grad.

COSA Gibt den Cosinuswert zum Winkel zurück.
Form: E=COSA(W)
Param: Winkel in Grad.

FRMT Gibt einen formatierten Wert zurück.

Form: E=FRMT(W,N)

Param: W: Wert

N: Anzahl Nachkommastellen

MAXV Gibt den Maximumwert der übergebenen Werte zurück.

Form: E=MAXV(A,B)

Param: 2 Werte von denen der Maximalwert bestimmt werden soll.

MINV Gibt den Minimumwert der übergebenen Werte zurück.

Form: E=MINV(A,B)

Param: 2 Werte von denen der Minimalwert bestimmt werden soll.

RAND Gibt eine Zufallszahl zurück (ganze Zahl).

Form: E=RAND(W)

Param: Wert aus dem die Zufallszahl bestimmt wird (Maxwert).

SINA Gibt den Sinuswert zum Winkel zurück.

Form: E=SINA(W)

Param: Winkel in Grad.

SQRT Gibt den Quadratwurzelwert zur Eingabe zurück.

Form: E=SQRT(W)

Param: Wert zu dem die Quadratwurzel bestimmt werden soll.

TANA Gibt den Tangenswert zum Winkel zurück.

Form: E=TANA(W)

Param: Winkel in Grad.

Aktionen:

BEEP Aktiviert einen Oszillator mit den voreingestellen Werten. Für die Einstellung muss WAVE benutzt werden.

Form: BEEP(X,S)

Param: X: Index des Oszilators <1,2..>.

S: Schalter ON/OFF <0,1>

Beispiele:

gast'Elo'ABC, gast'Musik'Panel, gast'Musik'Twinkel,

BUMP Prüft die Umgebung auf eine Berührung.

Form: BUMP(P)

Param: P: Prüfart <0,..>

0: Berührung prüfen. Rückgabe ist das Prüfergebnis. 0: nein, 1: ja

sonst: Rückgabewert ist der Bumpwinkel.

Beispiele:

gast'Dynamic'Ball, gast'Dynamic'Ball2

DISP Stellt einen definierten Inhalt in einem Display dar.

Form: DISP(X,T,L)

Param: X: Index des Displays <1,2..>.

T: Wertauswahl <0,1,...>

0: Literal

1: Titel des POxLs

sonst: Titel eines anderen POxLs (Blips)

L: Darzustellendes Literal (Zahlenwert)

Beispiele:

gast'Elo'messen, gast'Leben'Obstgarten,

GETP GetParameter: Liefert den Wert einer Ping-Variablen.

Form: GETP(X,V)

Param: X: Index des Blips <1,2..>.

V: Variable <1,2,3>

1: Richtung

2: Öffnung

3: Leistung

4: x-Pos

5: y-Pos

Beispiele:

gast'Leben'segeln

LOOK Prüft die Umbegung auf Ausstahlungen und gibt der erfassten Maximalwert zurück. Ist kein Typ (T) angegeben und der Ping-Typ ist 'Sensor', wird die Sensornummer zurückgegeben, sonst die aktuelle Leistung.

Form: LOOK(X,T), LOOK(X)

Param: X: Index des Blips <1,2..>.

T: Wert

1: Log-Wert der max. Leistung

2: Max. Leistung

3: Leistung

4: x-Pos

5: y-Pos

Beispiele:

gast'Elo'ABC, gast'Leben'segeln

PCRS POxLCourse: Liest oder setzt den Kurs für ein POxL. Die Angabe des Kurses ist in Grad und rechtweisend d.h. 0 Grad ist immer oben und die Gradangabe erfolgt im Uhrzeigersinn.

Form: PCRS(T)

Param: T: Transferrichtung <0,1,..>

0: Wert lesen

1: Wert schreiben

sonst: Ausfallwinkel setzen (Bumpwinkel ist Einfall)

Beispiele:

gast'Leben'segeln, gast'Eleben'Obstgarten

PING Erzeugt eine Ausstahlung. Die Ausstrahlung wird bestimmt durch den Ping-Typ. Generiert einen vorher definierten Ping. Für Änderungen an den Parametern muss die Funktion PUTP genutzt werden.

'aktiv' : Wie bei einer Radar-Ausstrahlung erfolgt hier die Rückgabe eines Echos. Es wird der Blip mit dem geringsten Abstand zum Pinger zurückgegeben.

'Sensor': Setzt die Sensornummer als Ausstrahlung.

'zeigen': Generiert eine Linie aus Punkten in der Länge der angegebenen Wertes.

'warten': Sensor kann ein- bzw. ausgeschaltet werden.

Form: PING(X,S), PING(X)

Param: X: Index des Pings <1,2..>.

S: on/off <0,1> (wenn Typ='warten').

Beispiele:

gast'Elo'ABC, gast'Leben'segeln

PPOS POxLPosition: Setzt oder liest die Position eines POxLs. Ist der Typ > 1, wird die Position mit den beiden Parametern P (x) und W (y) gesetzt.

Form: PPOS(T,P,W), PPOS(T,X,Y)

Param: T: Transferrichtung <0,1,..>

0: Wert lesen

1: Wert schreiben

sonst: Pos schreiben x und y

P: Positionswert

0: x

1: y

W:Wert

Beispiele:

gast'Musik'Twinkel

PSPD POxLSpeed: Liest oder setzt die Gechwindigkeit für ein POxL.

Form: PSPD(T)

Param: T: Transferrichtung <0,1,..>

0: Wert lesen

1: Wert schreiben

sonst: Wert reduzieren (Bumpwert ist Prozentwert)

Beispiele:

gast'Leben'segeln

PUTP PutPing: Setzt den Wert einer Ping-Variablen.

Form: PUTP(X,V,W)

Param: X: Index des Pings <1,2..>.

V: Variable <1,2,3>

1: Richtung

2: Öffnung

3: Leistung

4: x-Pos

5: y-Pos

W: Wert

Beispiele:

gast'Leben'segeln

SKIP Überspringt eine Anzahl von Statements.

Form: SKIP(N)

Param: N: Anzahl der zu überspringenden Statements <1,2..>.

WAVE Stellt die Frequenz für einen Oszilator ein. Die Schaltung des Oszilators erfolg mit BEEP.

Form: WAVE(X,F)

Param: X: Index des Oszilators <1,2..>.

F: Frequenz

Beispiele:

gast'Elo'ABC, gast'Musik'Panel, gast'Musik'Twinkel

Noch ein Tipp: Um den Wert von Systemvariablen zu prüfen, empfiehlt es sich, eine Nutzervariable zu definieren und dieser an der zu überprüfenden Stelle die Systemvariable zuzuweisen. Die Prüfvariable wird dann im Snap-Pad (bei der Konstruktion oben rechts) beim Überfahren der PoxL-Marke angezeigt werden.