Accessible Blockly Plugin

Authors: Rachael Bosley, Luna Meier

# Overview

## Goal

Allow for people with any level of vision to work with Blockly, whether they have no vision, full vision, or somewhere in between. The program must also stay user friendly so that children can use it effectively. In order to allow Blockly to be used by blind users, the entire interface must be hotkey accessible and easily readable by a screen reader.

## About Blockly

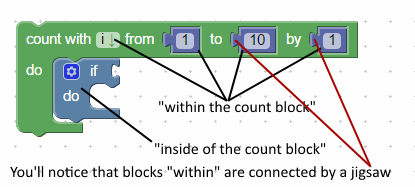
Blockly is a visual coding interface. It allows users to create code by placing blocks together. New blocks can be easily created by developers to enable more functionality within the interface. Blockly code can be exported to XML, Python, Javascript, PHP, and Dart. Basic Blockly does not have the ability to run code, so these exports are necessary to see the final product.

## Glossary

**User**: Person who is interacting with the finished version of the program, after everything has been put together.

**Developer**: Person or group responsible for integrating our plugin into their own Blockly implementation.

**Within vs. Inside of**: A block or field is considered “within” another block if it is an integral part of the block itself. A block is considered “inside of” another block if it is part of a statement that is run by a block. This concept is rather confusing, so see the picture below for a visual example:



From a coding standpoint, child blocks that are within a parent block output directly into the parent block, while child blocks that are inside of a parent block are code that is run according to the parent block.

## Features

### Navigating Between Blocks in the Scene

The user will be able to navigate throughout the scene with a series of hotkeys. When the user presses a hotkey the selected block should change, and the user will be informed of what they are now selecting. The user should be able to move to any block attached to the current block. Very often in Blockly the user will need to get to a child block of a parent block that is within the parent block. In order to allow the user to select all possible child blocks of a selected parent block, we have implemented a specific mode that allows you to cycle through all possible connections of the selected block. You press a hotkey, which allows you to use the hotkeys you were previously using to navigate through the scene to be used to select a specific connection instead. In addition, this mode will allow you to select any fields that are part of the block, ensuring that the user has full access to the block. Once the user has interacted with the block as they like, the program returns to the normal navigation mode, where pressing the navigation hotkeys allows you to move to attached blocks and inside of blocks.

*Luna please rewrite*

### Navigating Within a Block

*The section above needs to be edited and some needs to be put into this section-- Luna*

### Adding Blocks to the Scene

The user will be able to press a key to initiate menu mode. In this mode, you can select a category and then select a block from that category. Alternatively, you can click on a category and click on a block and drag it into the scene.

*Add anything else here*

### Moving Blocks around on the Scene

The user will be able to use hotkeys within a block to select a connection, select another block, and move the new block to that connection. This is already done with ability to drag with a mouse, but we need it to be able to be done with hotkeys as well.

### Tree Diagram Representing Scene

The user will have a box on the side of their screen that reads them their comments in a tree view. The tree view will be updated automatically, it will be displayed in the format of the blocks to keep the structures looking the same. All the blocks in the workspace have their own prefixes assigned to them and the comment tree view will display the block prefix followed by that blocks comment if it has a comment. They will also have an info box which tells more information about the block they are currently on in a larger font for a better understanding of where you are in the structure. You will be able to jump between a comment you have highlighted and a block in the workspace.

### Ability to Use both Hotkeys and Mouse

In order for this product to be accessible to blind people, everything must be accessible for use with hotkeys. People that are unable to see the screen cannot use a mouse, so they need to be able to use hotkeys instead. This does not mean that mouse control can be disabled, because this product also needs to be available to sighted people. Hotkeys cannot conflict with mouse control or other hotkeys.

### Accessibility for Screen Readers

The Accessible Blockly plugin is designed with the use of screen readers in mind. WAI-Aria attributes are utilized throughout the program to enhance the semantics of every item on the page. Users are able to navigate and interact with Blockly using verbose audio information. The user will always know their location on the page or within the workspace as well as how to interact with their current selection. The following screen readers have been tested to work with our plugin: Voiceover, NVDA, JAWS.

### Use of Hooks for Developer Ease

*Luna please write notes for this*

### Quick Reference Guide

The user will have access on each block to bring them to an accessible link to find information about the block. Each block will have a basic run-down of how it can be used and what for. There is also a page that provides information on the hotkeys.

## Goals

## List of Hotkeys:

Alt Shift C: toggle collapse/expand  
Alt Shift D: duplicate  
Alt Shift E: toggle enable/disable  
Alt Shift H: navigate to help page for selected block  
Alt Shift I: toggle inline  
Ctrl Y: redo  
Ctrl Z: undo  
Delete: delete a block  
Enter: leaves editing a field and saves the field  
Escape: exit current field or menu  
A: traverse out  
C: add a comment to selected block  
D: traverse in  
G: jump to associated block or comment  
R: jump to top of section  
S: traverse down  
W: traverse up

# Technical

## Libraries Used

We are using the Google Closure library for Javascript. This is necessary, because in order to keep our code up to Google coding standards, we need to use Closure.

## Coding Standards

Link to the coding standards can be found here: <http://google.github.io/styleguide/javascriptguide.xml>

# Future Plans

## QA Testing

## Building an educational module for students

# Additional Examples