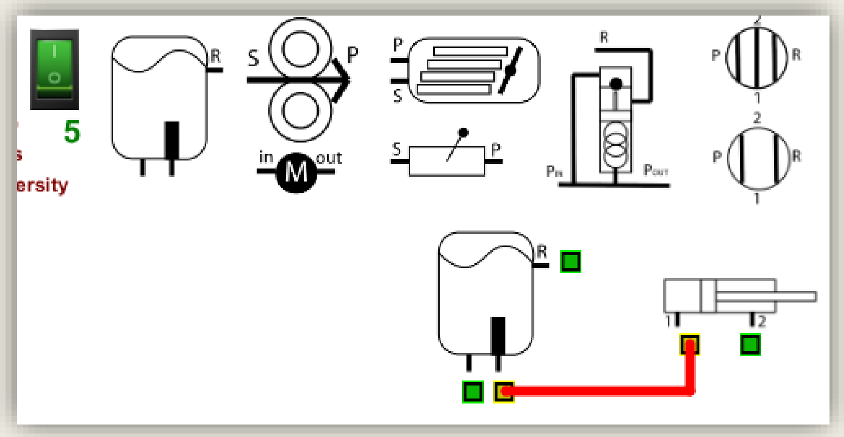
IMPROVED FEEDBACK MECHANISMS OF HYDRAULICS SANDBOX SIMULATOR

Dr. Karen Johnson and Martin Hebel
Aviation Technologies – Southern Illinois University

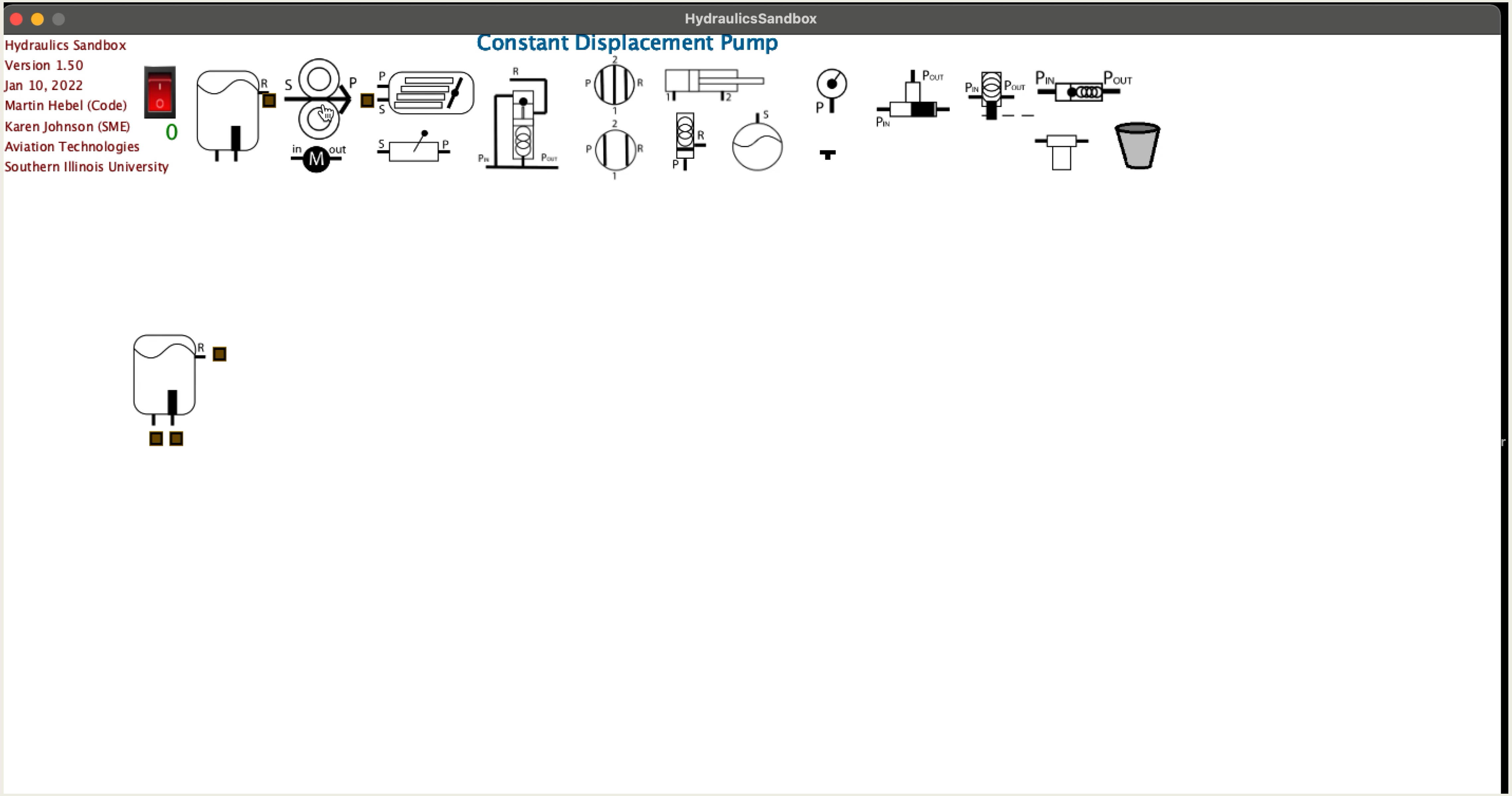
Hydraulics Sandbox

- Developed by the presenters
- Allows simulating the build of a hydraulics system with standard components
- The software checks for valid and invalid connections
- Hopefully will be beneficial in students' understanding



Quick Use Guide

- Download from:
  www.selmaware.com/sandbox
  PC & Linux installers, other options for MacOS
- Drag components to build area
- Click a port to connect hose - Right-click to cancel a hose
- Right-Click a port to delete a hose
- A Component must be disconnected to drag
- Drag components to trash to delete
- Turn on switch to check connection validity - Counter will update each time turned on


HydraulicsSandbox

Constant Displacement Pump

Hydraulics Sandbox
Version 1.50
Jan 10, 2022
Martin Hebel (Code)
Karen Johnson (SME)
Aviation Technologies
Southern Illinois University

Implementation

- Hydraulic Systems and Landing Gear Course (16 weeks)
  - *Section 1 (n=17) - concurrent sandbox*
  - *Section 2 (n=17) - terminal sandbox*
- Both sections given the same list of components each week to create their system
  - *Submitted screenshots of final build with timestamps and attempts*
- Unit tests (4 plus final exam)
  - *Drawing (on paper) a schematic with given list of components*
  - *MCQs related to component interactions within a system*
    - General and system specific

Findings

- **No statistically significant results**
- Only (very) minor differences in actual numbers of right/wrong on assessments
  - *Terminal group did slightly better on both schematics and MCQ*
- Overall more attempts made (weekly) by the concurrent group
  - *Started over rather than using the trash bin?*
- Overall more time logged (weekly) by the concurrent group
  - *More time going back to fix errors along the way?*
- Chalk this up to a pilot study of the software

THEORY OF CODE OPERATION

Hydraulics Sandbox Code

- Installers available for Windows and Linux.
  www.selmaware.com/sandbox

  The installers add an updated that can be used to check if a new version is available.

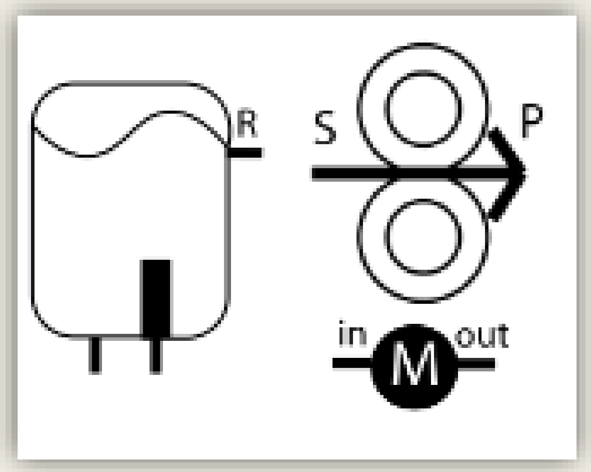
- A zip of the code is available under MacOS distribution – can be used on all platforms:

- Developed using the Processing 3 Java environment, which is free.

- Directions on the page explain how to make your own .exe build – open and export, done. Modify if you desire! Please do not publicly distribute.

- This is required for local MacOS distribution on flash drives as the exe did not pass Apple's Notarization checks for download use.

Component Objects



- There is a single component object.
- All the various components are created at load with a finite number of each.
- At creation of each, they are indexed in a certain range, such as the variable displacement pumps are 10 – 14. The also are assigned an image and component type, along with size information.

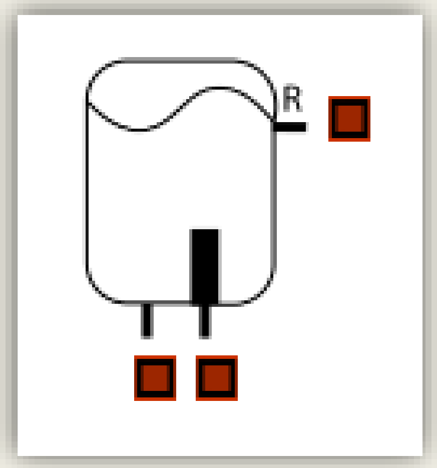
```
for(i=10; i<15;i++)          // create 5 variable pumps, #3
    component[i] = new  Components(350,30,100,i,3, "Variable Displacement Pump.png");
```

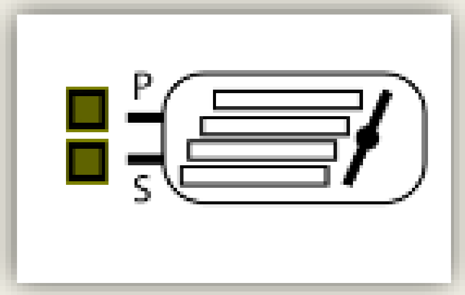
Port Objects

- Each component can have up to 4 connection ports.
- When placed in the build area, ports are added based on the Component ID.
- Each port number is indexed based on the index of the component:

  component index x 4 + 0
  component index x 4 + 1
  component index x 4 + 2
  component index x 4 + 3

- For the variable displacement pump (index 10), the ports will be 40, 41, 42 and 43 (if all 4 had been used).
- This allows quick identification of components from the port index number.



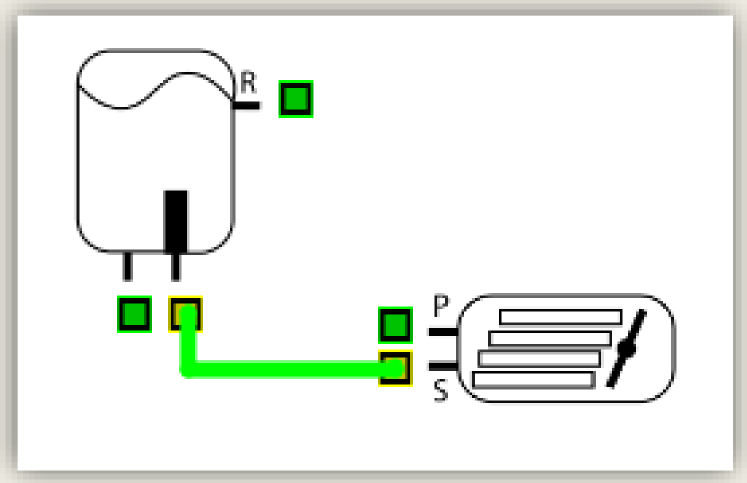


Hose Objects

- When a hose is placed, the beginning and end port index numbers are assigned to it.

- This allows easy identification of the component indexes the hose is connected to (40/4 = component index 10) and which ports by resolving 40, 41, 42, 43 to 0, 1, 2, 3.

- Having the component index, that component object can be polled to determine its type.



Connection Verification Rules

- Being able to resolve the port number and the component type, the rules check for 4 rule sets by checking each hose in sequence:
  - *Is there a valid connection?*
  - *Does it connect to itself some how?*
  - *Is there an invalid connection via tees?*
  - *Is there a valve/actuator agreement?*

```
for (int i=0;(i < numHoses); i++)            // go through each hose used
   {
    connCount = 0;
    if (hose[i].visible()) {               // if visible
      finalResult = checkHoses(i);                          // check for proper connections
      if (finalResult)  finalResult = test2Self(i);           // go ensure it doesn't connect to itself
      if (finalResult)  finalResult = checkBadTeeConnections(i);    // go run through not-allowed connection list
      if (finalResult)  finalResult = valves2Actuators(i);    // check valve/actuator agreement for both hoses
      setHose(i, finalResult);
     }
    }
```

Valid Connection Rules

- A valid connection rule checks component type and port to another component type and port for each hose.

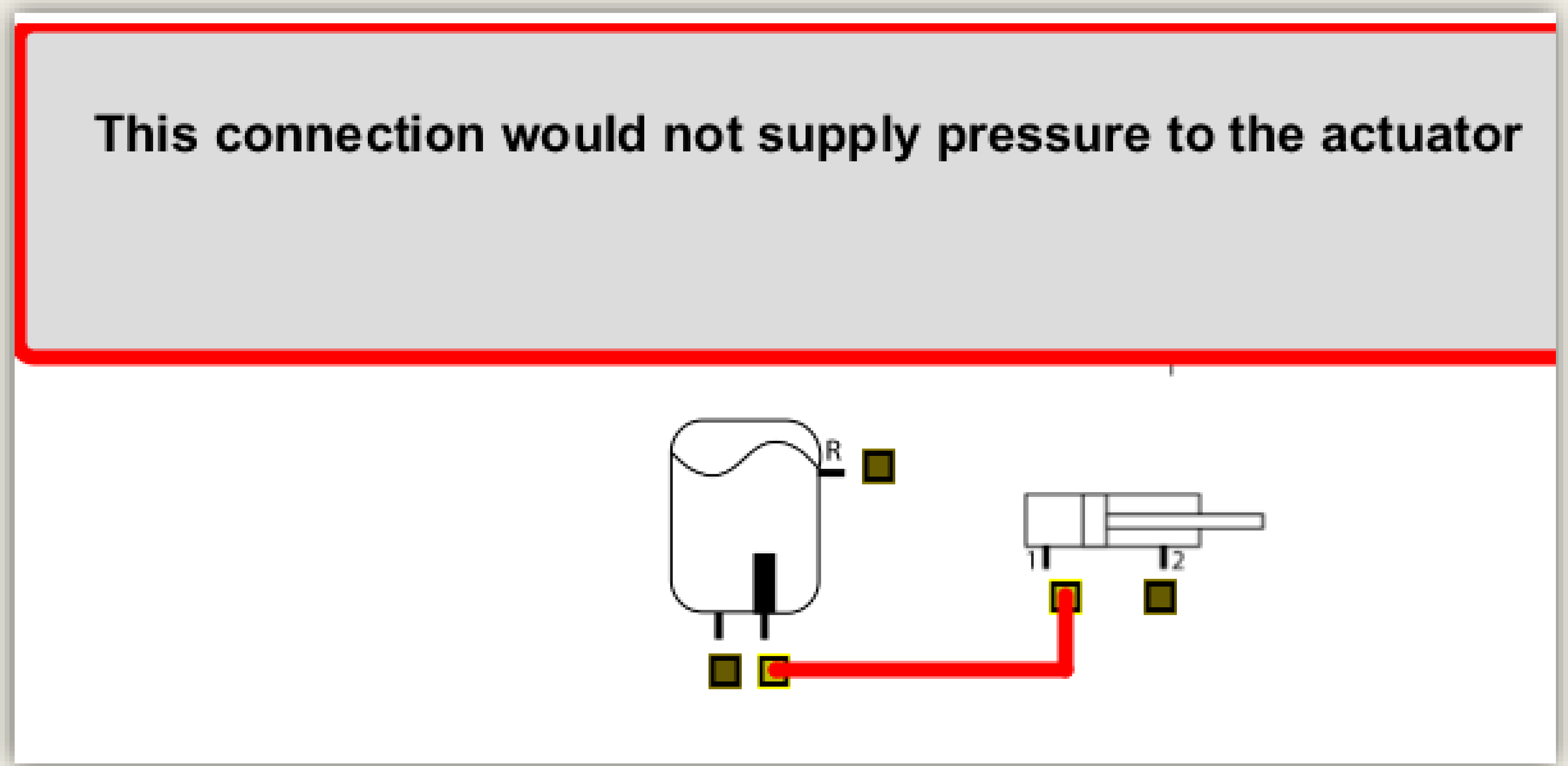
```
if (testHose(i,res,1,constDispPump,0))                    return true;
if (testHose(i,res,1,varDispPump,1))                      return true;
if (testHose(i,pressReg,2,closedCenterValve,0))           return true;
if (testHose(i,pressReg,2,closedCenterValve,1))           return true;
```

Invalid Connection Rules

- While a hose may check ok, a tee from it may form an invalid connection.
- Some invalid connections are checked to provide a feedback message to the user when they place the pointer over the connection.
- The port is not checked in all cases, just the component types.

```
if (testCompHose(i,res,actuator)) {
    hoseMsg[i][0]="This connection would not supply pressure to the
actuator";
    return false;
}
```

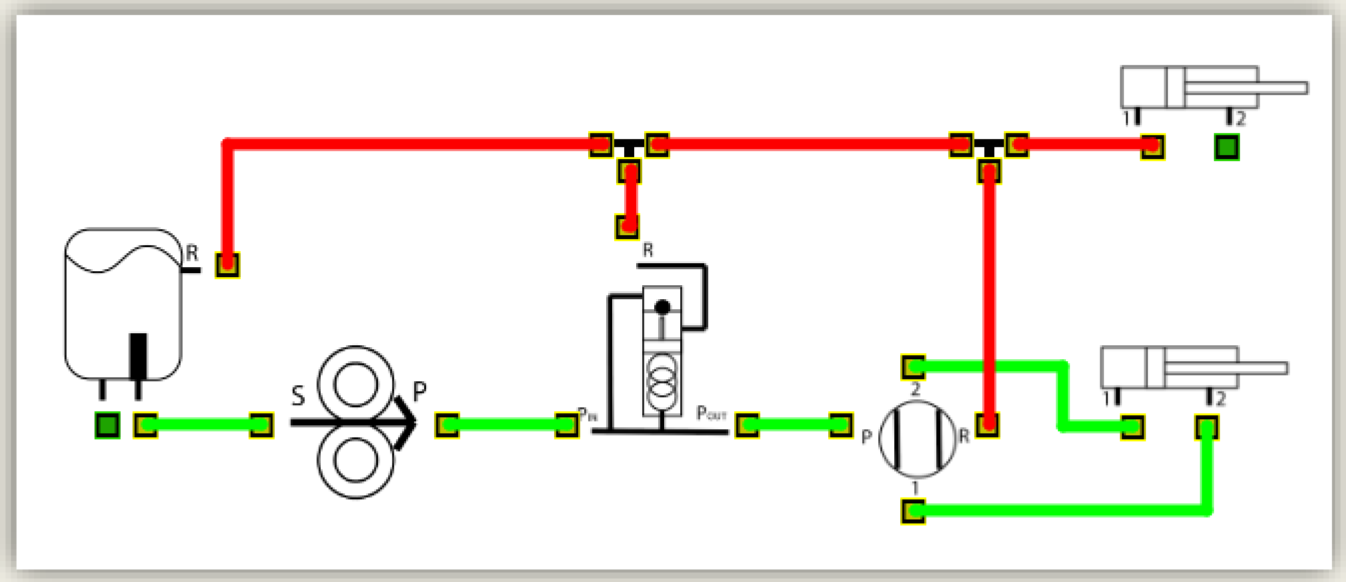
Example Feedback


This connection would not supply pressure to the actuator

Tee Checks

- While a single hose connection may be good, hoses from the tees, and subsequent tees and their hoses need to checked for validity.

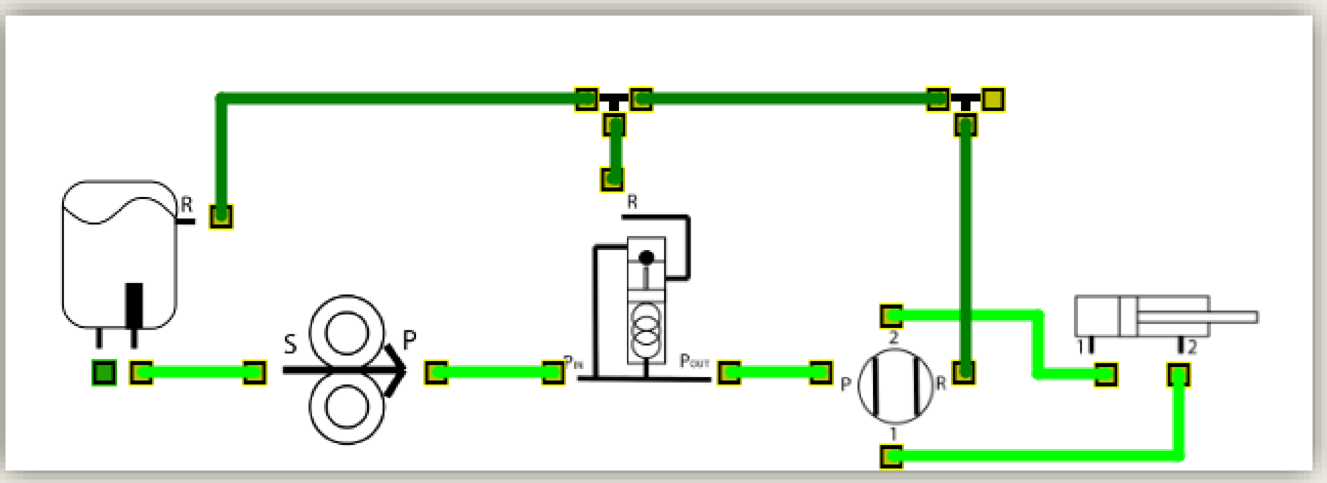
- This is done with recursive calls to trace a path through multiple tees.



Return Lines

- Hoses are checked to see if they connect to the reservoir return and displayed in dark green.

```
if (result){
     if (testHose(i,res,3))   // in return line, make dark green
       hose[i].finish(color(0,128,0));
     else
       hose[i].finish(color(0,255,0));    // good, normal green
   }
   else
     hose[i].finish(color(253,0,0));     //
```

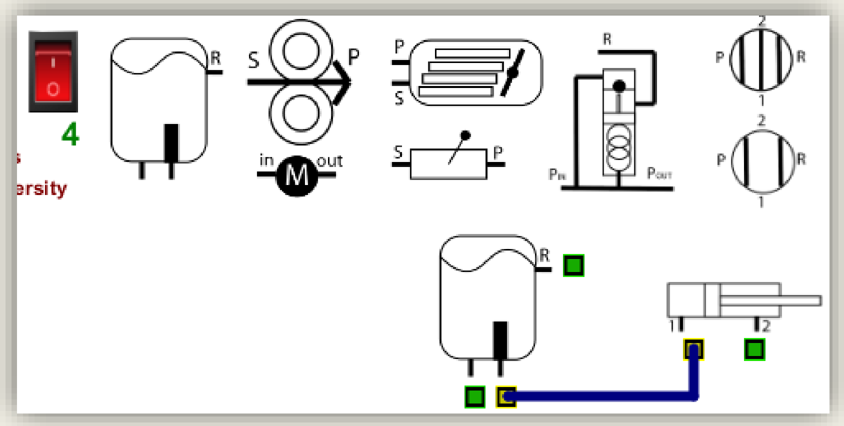


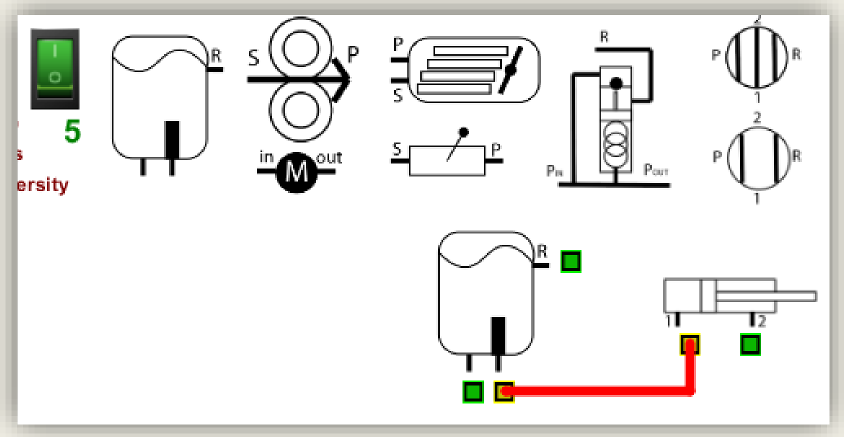
Enabling Checking

- When the toggle switch is off, hoses are not checked and will remain blue.

- When turned on, hoses are checked, and the counter is increased to allow verifying during a quiz situation.

  While on, any subsequent hoses placed will be checked.





Summary

- Final use notes
  - *There is NO saving/opening builds.*
  - *Do NOT press the escape key - It will close.*
  - *To start a new build, close and re-open or it may become sluggish and the parts bin may empty.*
- A Windows and Linux installer is available. Mac versions need to be ‘Exported’ using the source code for local manual distribution.
- Bugs may still exist depending on what the student does but is effective at helping them understand a hydraulics system build we hope.