Spatial Skills App Documentation

Table of Contents

1.0 Introduction

This documentation covers the Spatial Skills app developed for Glasgow University.

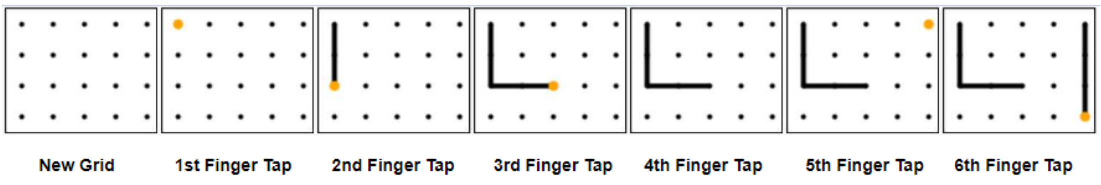
The app has been developed to allow users to develop their spatial skills by completing a series of drawing exercises. The exercises are comprised of orthographic and isometric projection, mental rotation and reflection.

The app has been created using HTML, CSS and JavaScript and can be deployed using the PhoneGap Build technology from Adobe. The app is available to be deployed to both Apple and Android devices.

2.0 Using the App

2.1 Drawing in the app

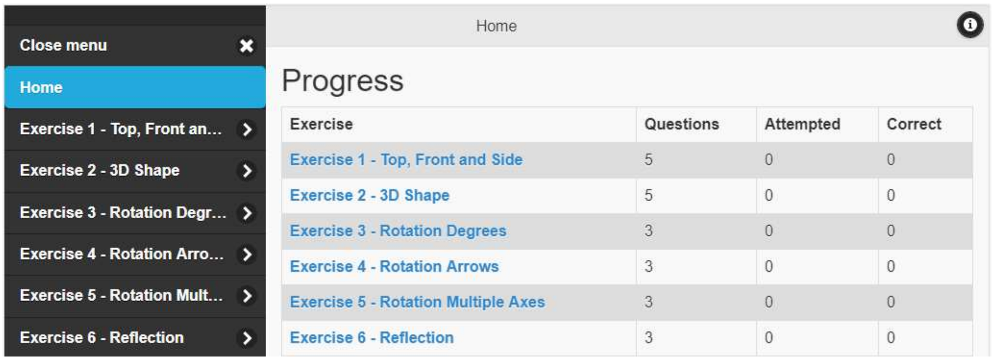
The app allows the user to draw lines by touching the screen as shown below:



The first finger tap on a black dot would display an orange dot, indicating that the next (second) finger tap would draw a line from the orange dot to this tap location. The orange dot would now exist at the second tap location, indicating that the next (third) finger tap would draw a line from this new location and so on. To deactivate the orange dot and begin drawing lines elsewhere, a single tap on the orange dot would deactivate, removing the orange dot from view (fourth), and allowing a line to begin being drawn from another location (fifth and sixth).

2.2 Home Page

The home page shows the user their current progress through the exercises:


Close menu
Home
Exercise 1 - Top, Front an...
Exercise 2 - 3D Shape
Exercise 3 - Rotation Degr...
Exercise 4 - Rotation Arro...
Exercise 5 - Rotation Mult...
Exercise 6 - Reflection
Home

Progress

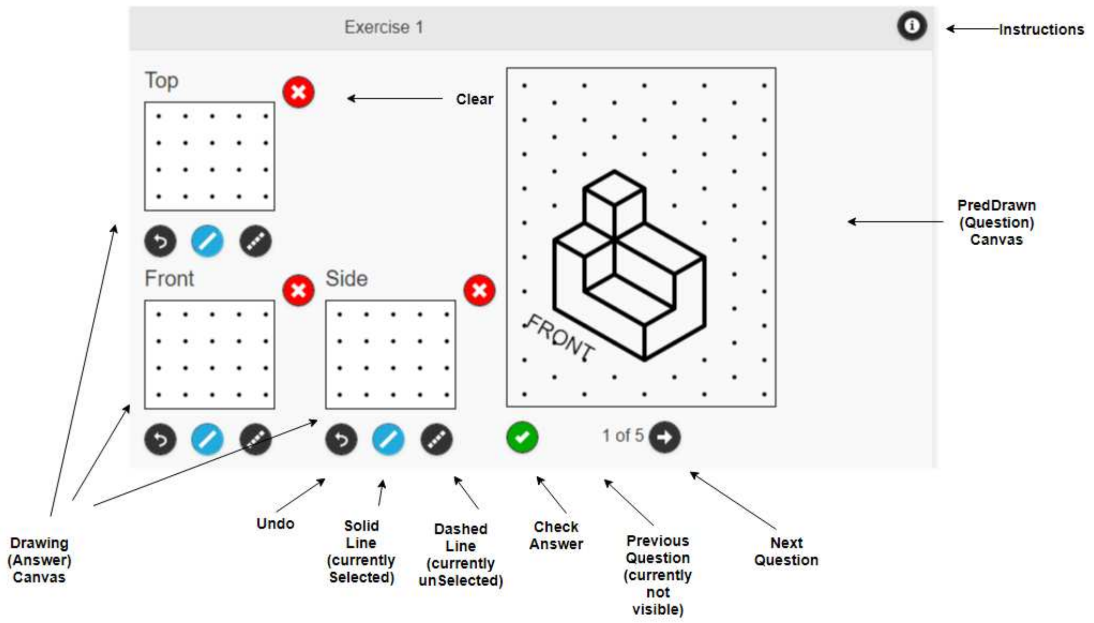
Exercise	Questions	Attempted	Correct
Exercise 1 - Top, Front and Side	5	0	0
Exercise 2 - 3D Shape	5	0	0
Exercise 3 - Rotation Degrees	3	0	0
Exercise 4 - Rotation Arrows	3	0	0
Exercise 5 - Rotation Multiple Axes	3	0	0
Exercise 6 - Reflection	3	0	0

Users can access the different exercises by clicking the appropriate link in the table or left-hand menu. The menu is collapsible on a smaller screen such as a phone.

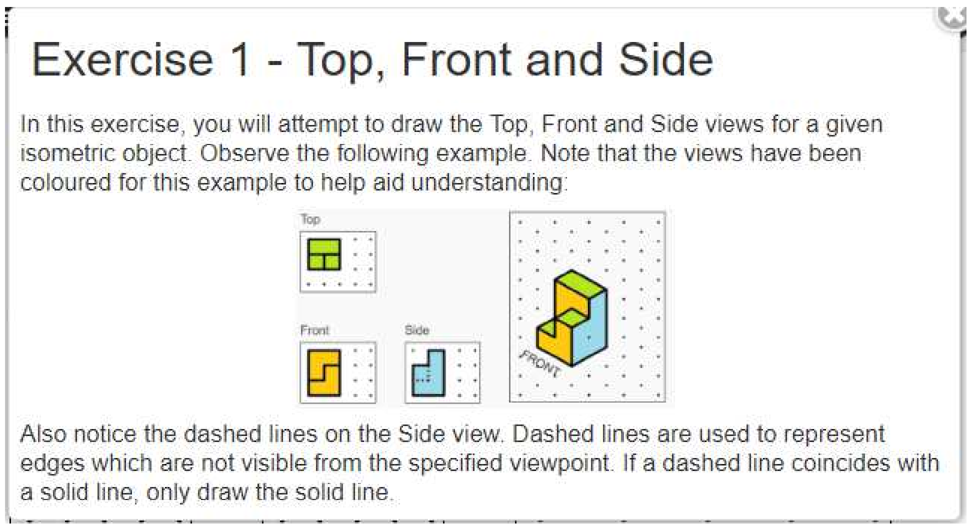
2.3 Exercises

2.3.1 Exercise 1 – Top, Front and Side Views

In exercise 1, the user has to draw the top, front and side orthographic views for a give 3D isometric shape:


Exercise 1
Instructions
Top
Clear
Front
Side
PredDrawn (Question) Canvas
FRONT
1 of 5
Drawing (Answer) Canvas
Undo
Solid Line (currently Selected)
Dashed Line (currently unSelected)
Check Answer
Previous Question (currently not visible)
Next Question

The instructions for how to carry out the exercise are given in a popup, accessed by pressing the instructions icon at the top right of the screen:



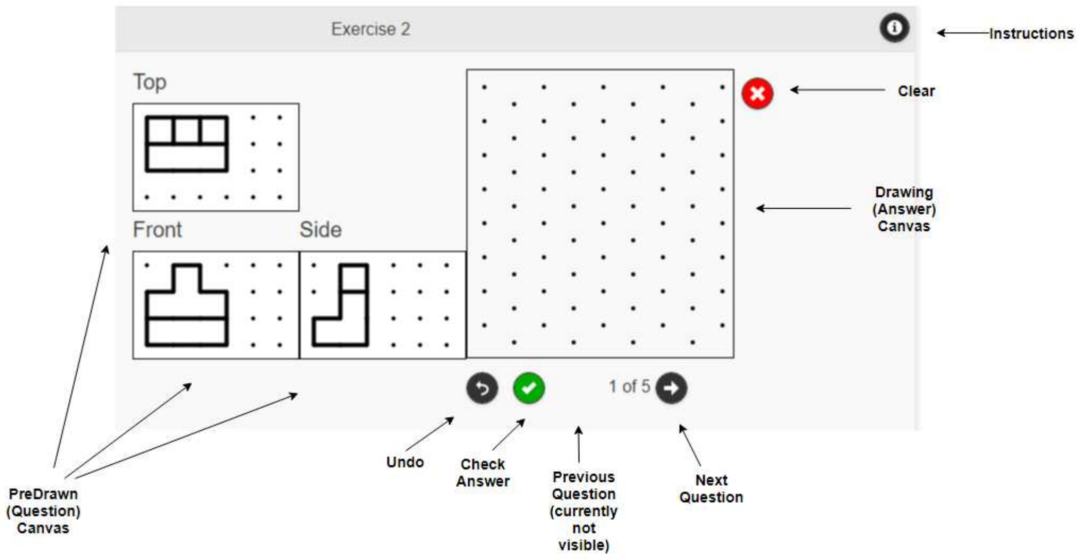
Exercise 1 - Top, Front and Side

In this exercise, you will attempt to draw the Top, Front and Side views for a given isometric object. Observe the following example. Note that the views have been coloured for this example to help aid understanding:

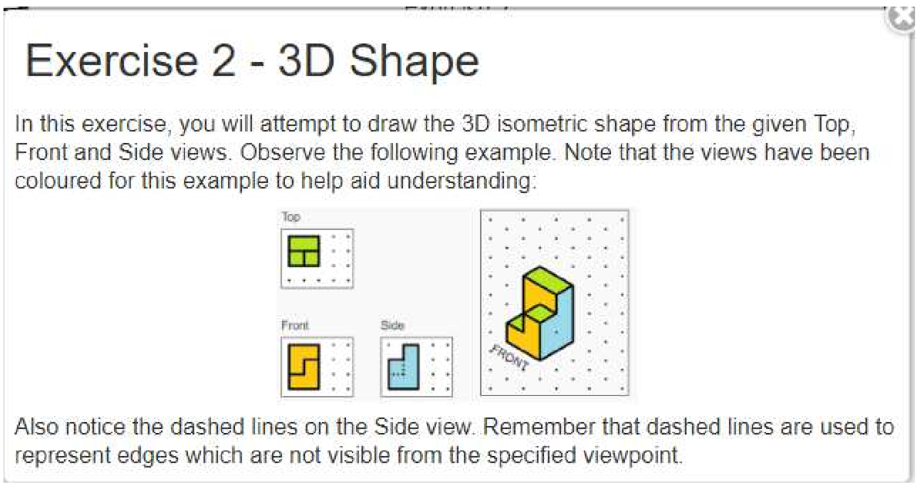
Also notice the dashed lines on the Side view. Dashed lines are used to represent edges which are not visible from the specified viewpoint. If a dashed line coincides with a solid line, only draw the solid line.

2.3.2 Exercise 2 – 3D Shape

In exercise 2, the user has to draw the 3D isometric shape from the given top, front and side orthographic views:



The instructions for how to carry out the exercise are given in a popup, accessed by pressing the instructions icon at the top right of the screen:



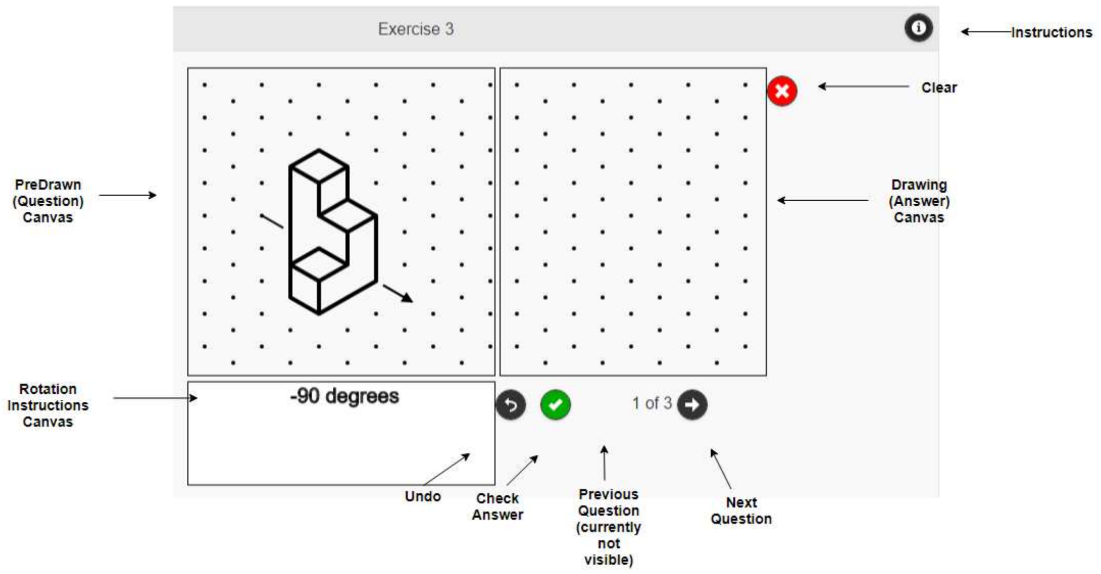
Exercise 2 - 3D Shape

In this exercise, you will attempt to draw the 3D isometric shape from the given Top, Front and Side views. Observe the following example. Note that the views have been coloured for this example to help aid understanding:

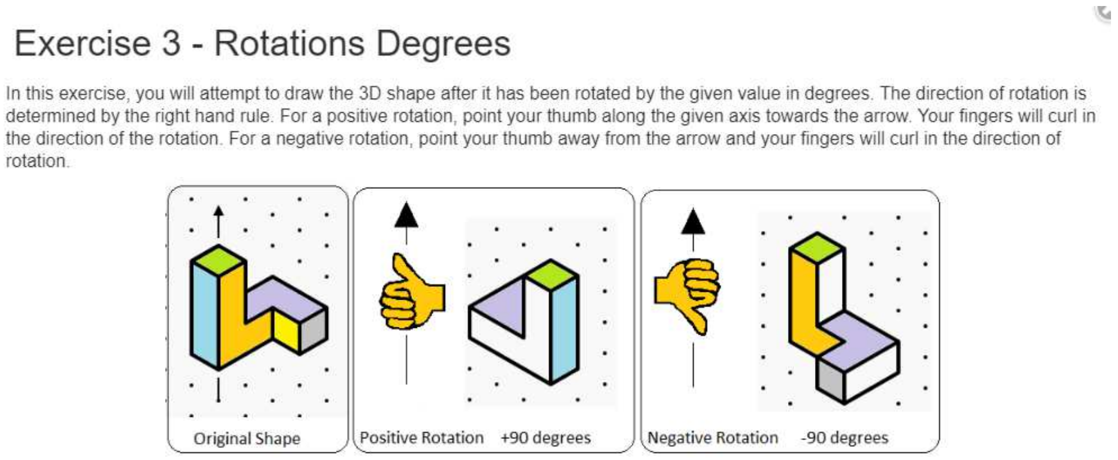
Also notice the dashed lines on the Side view. Remember that dashed lines are used to represent edges which are not visible from the specified viewpoint.

2.3.3 Exercise 3 – Rotate (degrees)

In exercise 3, the user is asked to draw the given 3D isometric shape, as it would be viewed after it has been rotated according to the instructions (degrees) in the Rotation Canvas:



The instructions for how to carry out the exercise are given in a popup, accessed by pressing the instructions icon at the top right of the screen:

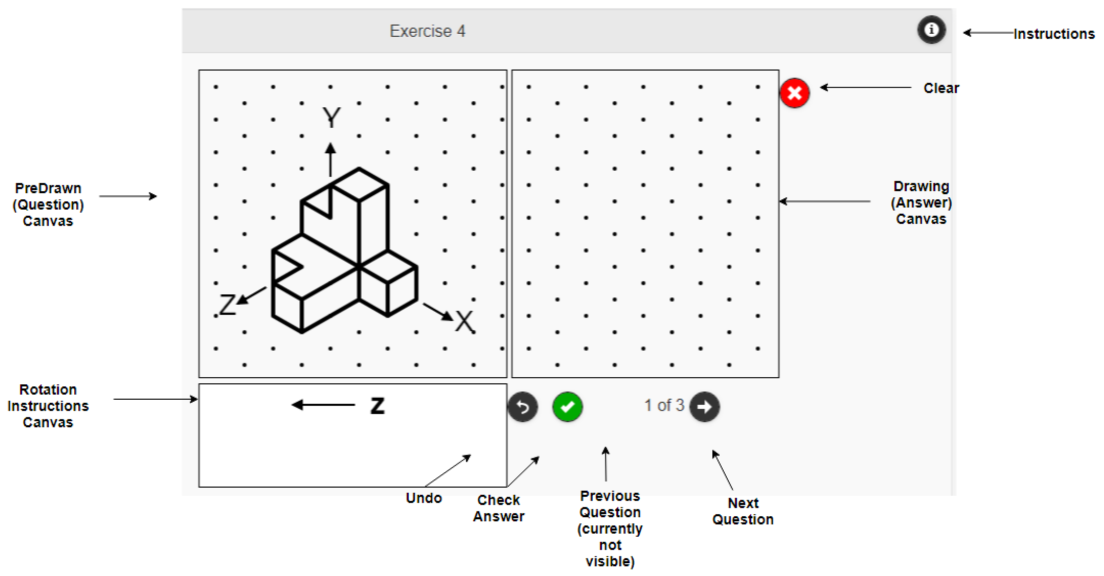


Exercise 3 - Rotations Degrees

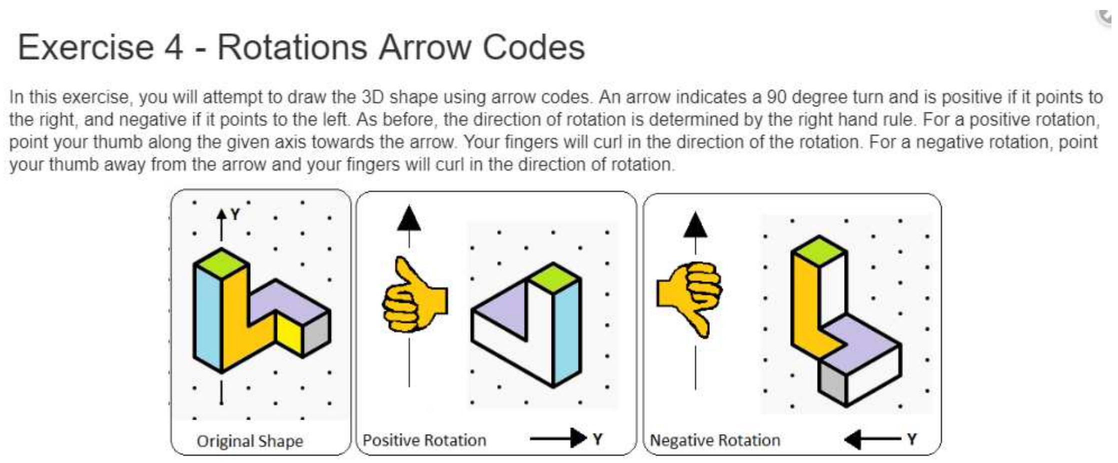
In this exercise, you will attempt to draw the 3D shape after it has been rotated by the given value in degrees. The direction of rotation is determined by the right hand rule. For a positive rotation, point your thumb along the given axis towards the arrow. Your fingers will curl in the direction of the rotation. For a negative rotation, point your thumb away from the arrow and your fingers will curl in the direction of rotation.

2.3.4 Exercise 4 – Rotate (arrow codes)

In exercise 4, the user is asked to draw the given 3D isometric shape, as it would be viewed after it has been rotated according to the instructions (arrow codes) in the Rotation Canvas:


Exercise 4
Instructions
Clear
PreDrawn (Question) Canvas
Drawing (Answer) Canvas
Y
Z
X
Rotation Instructions Canvas
Z
1 of 3
Undo
Check Answer
Previous Question (currently not visible)
Next Question

The instructions for how to carry out the exercise are given in a popup, accessed by pressing the instructions icon at the top right of the screen:

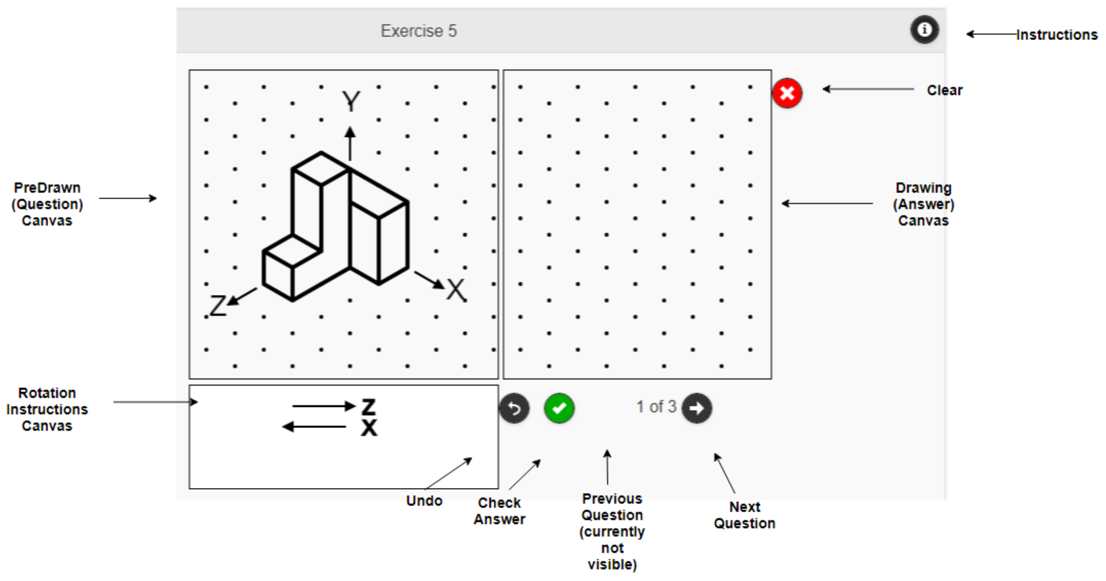


Exercise 4 - Rotations Arrow Codes

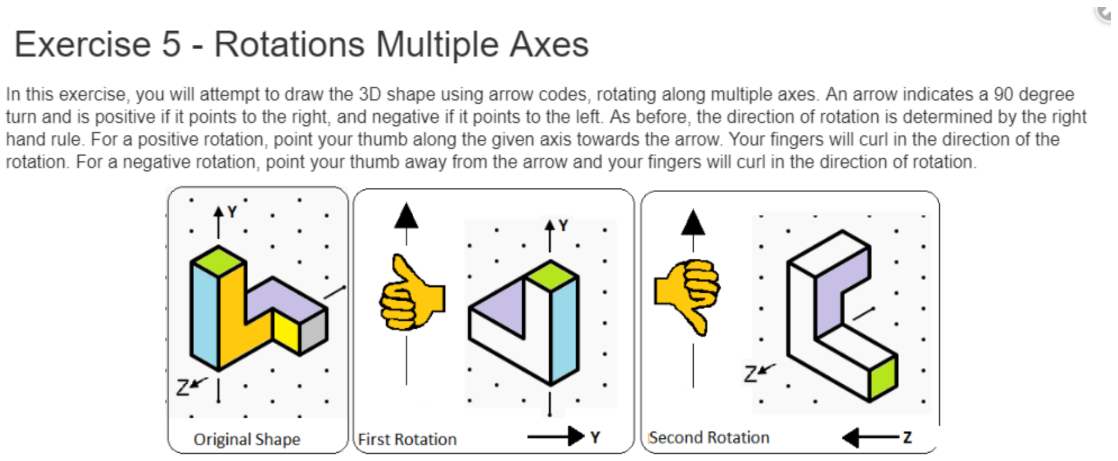
In this exercise, you will attempt to draw the 3D shape using arrow codes. An arrow indicates a 90 degree turn and is positive if it points to the right, and negative if it points to the left. As before, the direction of rotation is determined by the right hand rule. For a positive rotation, point your thumb along the given axis towards the arrow. Your fingers will curl in the direction of the rotation. For a negative rotation, point your thumb away from the arrow and your fingers will curl in the direction of rotation.

2.3.5 Exercise 5 – Rotate (Multiple Axes)

In exercise 5, the user is asked to draw the given 3D isometric shape, as it would be viewed after it has been rotated around multiple axes according to the instructions (arrow codes) in the Rotation Canvas:


Exercise 5
Instructions
Clear
PreDrawn (Question) Canvas
Drawing (Answer) Canvas
Y
X
Z
Rotation Instructions Canvas
Z
X
1 of 3
Undo
Check Answer
Previous Question (currently not visible)
Next Question

The instructions for how to carry out the exercise are given in a popup, accessed by pressing the instructions icon at the top right of the screen:



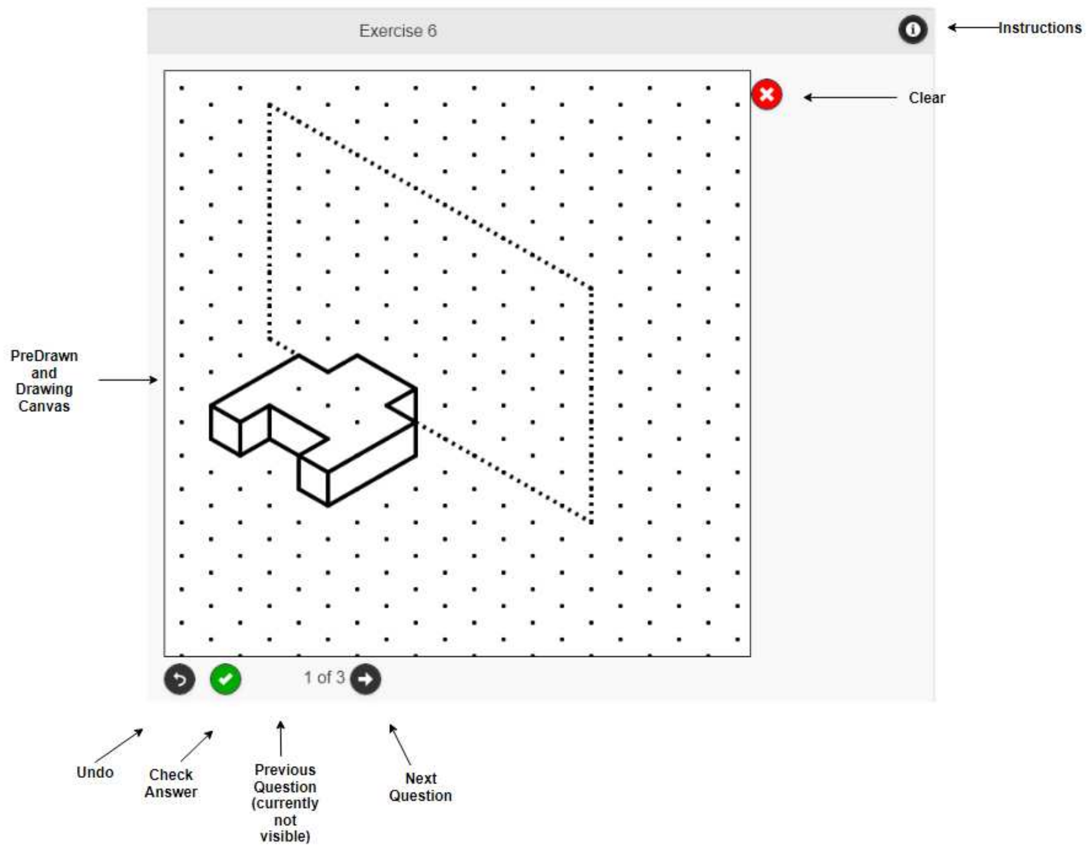
Exercise 5 - Rotations Multiple Axes

In this exercise, you will attempt to draw the 3D shape using arrow codes, rotating along multiple axes. An arrow indicates a 90 degree turn and is positive if it points to the right, and negative if it points to the left. As before, the direction of rotation is determined by the right hand rule. For a positive rotation, point your thumb along the given axis towards the arrow. Your fingers will curl in the direction of the rotation. For a negative rotation, point your thumb away from the arrow and your fingers will curl in the direction of rotation.

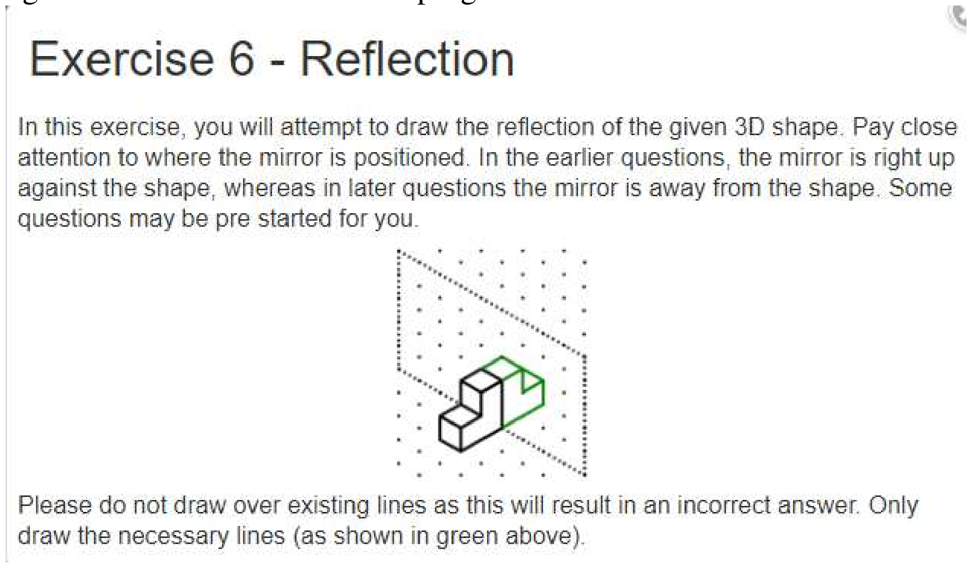
2.3.6 Exercise 6 – Reflection

In exercise 6, the user is asked to draw the reflection on a given shape:

The instructions


Exercise 6
Instructions
Clear
PreDrawn and Drawing Canvas
1 of 3
Undo
Check Answer
Previous Question (currently not visible)
Next Question

The instructions for how to carry out the exercise are given in a popup, accessed by pressing the instructions icon at the top right of the screen:



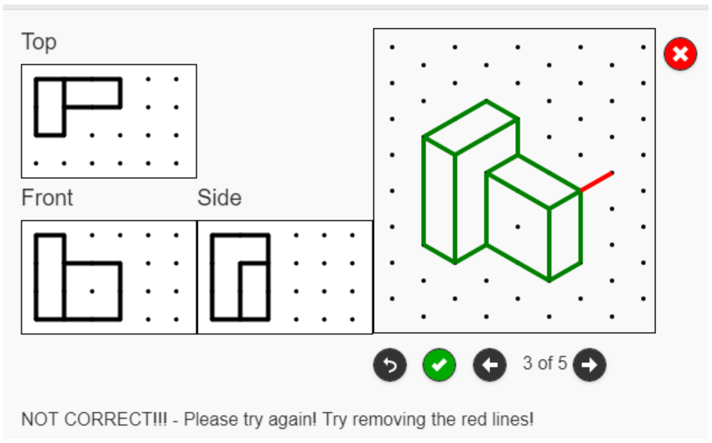
Exercise 6 - Reflection

In this exercise, you will attempt to draw the reflection of the given 3D shape. Pay close attention to where the mirror is positioned. In the earlier questions, the mirror is right up against the shape, whereas in later questions the mirror is away from the shape. Some questions may be pre started for you.

Please do not draw over existing lines as this will result in an incorrect answer. Only draw the necessary lines (as shown in green above).

2.4 Checking an Answer

The app allows the checking of an answer in order to provide feedback to the user. The following example shows the feedback after a user has pressed the green check button:



NOT CORRECT!!! - Please try again! Try removing the red lines!

As can be seen above, the user has got the 3D isometric almost correct. The green lines show all the correct lines. Lines shown in red are any lines which are incorrect.

There is also a message displayed to the user to either inform them:

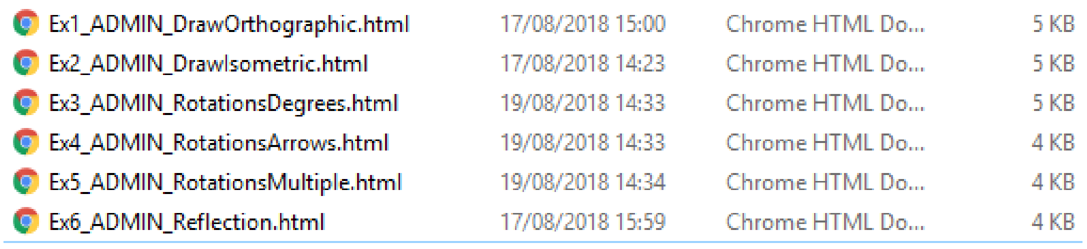
- The answer is correct
- The answer requires more lines
- The answer requires lines to be removed (shown in red)
- The answer requires thought regarding dashed lines

Once a user has pressed the check button for a question, that question is marked as attempted and this will be reflected in the Home Page. If the question is correct this will also be reflected in the Home Page totals.

3.0 Admin Pages

3.1 Using Admin Pages

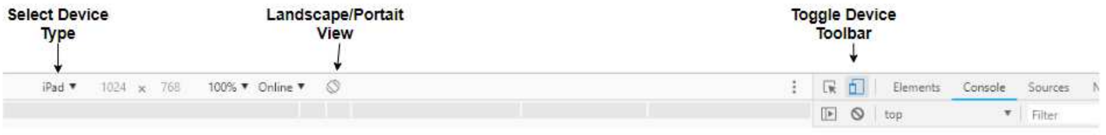
There are 6 admin pages, one for each exercise:



Ex1_ADMIN_DrawOrthographic.html	17/08/2018 15:00	Chrome HTML Do...	5 KB
Ex2_ADMIN_DrawIsometric.html	17/08/2018 14:23	Chrome HTML Do...	5 KB
Ex3_ADMIN_RotationsDegrees.html	19/08/2018 14:33	Chrome HTML Do...	5 KB
Ex4_ADMIN_RotationsArrows.html	19/08/2018 14:33	Chrome HTML Do...	4 KB
Ex5_ADMIN_RotationsMultiple.html	19/08/2018 14:34	Chrome HTML Do...	4 KB
Ex6_ADMIN_Reflection.html	17/08/2018 15:59	Chrome HTML Do...	4 KB

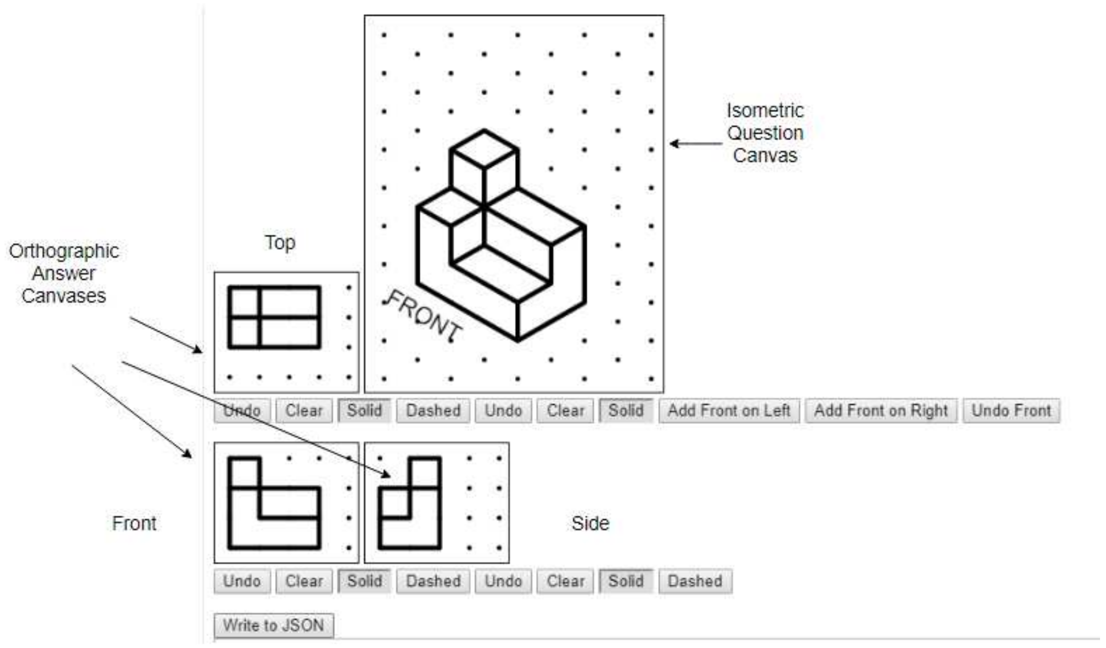
Each admin page is set up to reflect the questions for that particular exercise in the app. You have to draw the predrawn shapes for the question as well as the answer. The answer will not be displayed to the user but will be stored as the correct answer for the question in order that the user can check to see if their answer is correct.

In order to interact with the admin pages on a PC we must first simulate using the app on a touch device, such as an iPad for example. Launch the desired admin page in Google Chrome, Open Chrome Developer Tools (Ctrl + Shift + I), open the 'Device Toolbar' if not already open and select iPad from the device toolbar, rotating the iPad view to Landscape if necessary:



Each exercise has its own intricacies, such as axes, or the text "FRONT" written on the canvas. Therefore the admin pages have different buttons to reflect this. Below is an example of each admin page with a complete question having been created for each. In order to actually save and add questions to the app, please read the "Saving Questions from Admin Pages to JSON" section.

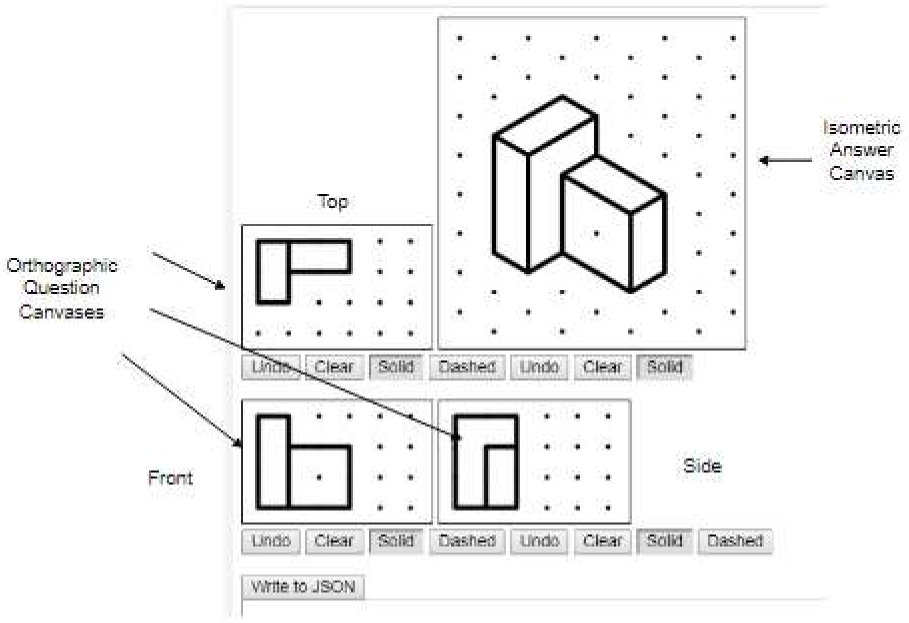
3.1.1 Exercise 1 – Admin Page



Each Canvas has its own Undo and Clear buttons to undo any lines drawn. The Top, Front and Side canvases also have an option to draw dashed lines as this is necessary to represent edges which are not visible from the specified view.

Additionally, the isometric canvas must have the text "FRONT" to indicate from where the shape should be viewed from. Clicking the button 'Add Front on Left' and then clicking on the canvas will add the text "FRONT" at the location of the click. The 'Add Front on Right' button can be used in a similar way.

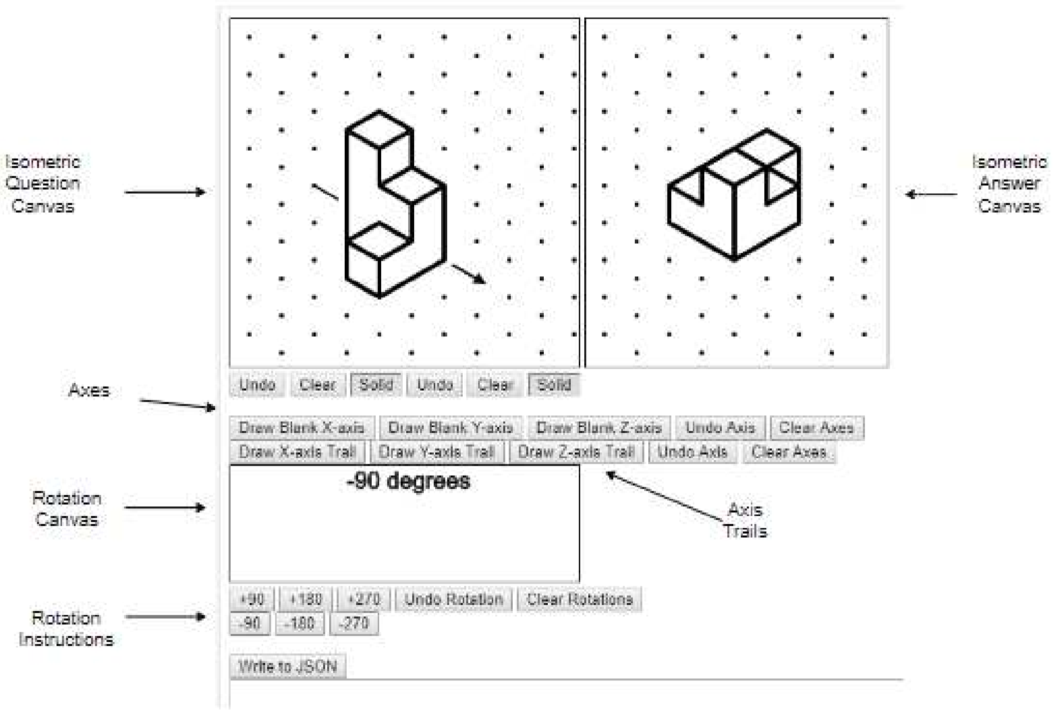
3.1.2 Exercise 2 – Admin Page



Each Canvas has its own Undo and Clear buttons to undo any lines drawn. The Top, Front and Side canvases also have an option to draw dashed lines as this is necessary to represent edges which are not visible from the specified view

There is no need to add text to any of the canvases for this question.

3.1.3 Exercise 3 – Admin Page

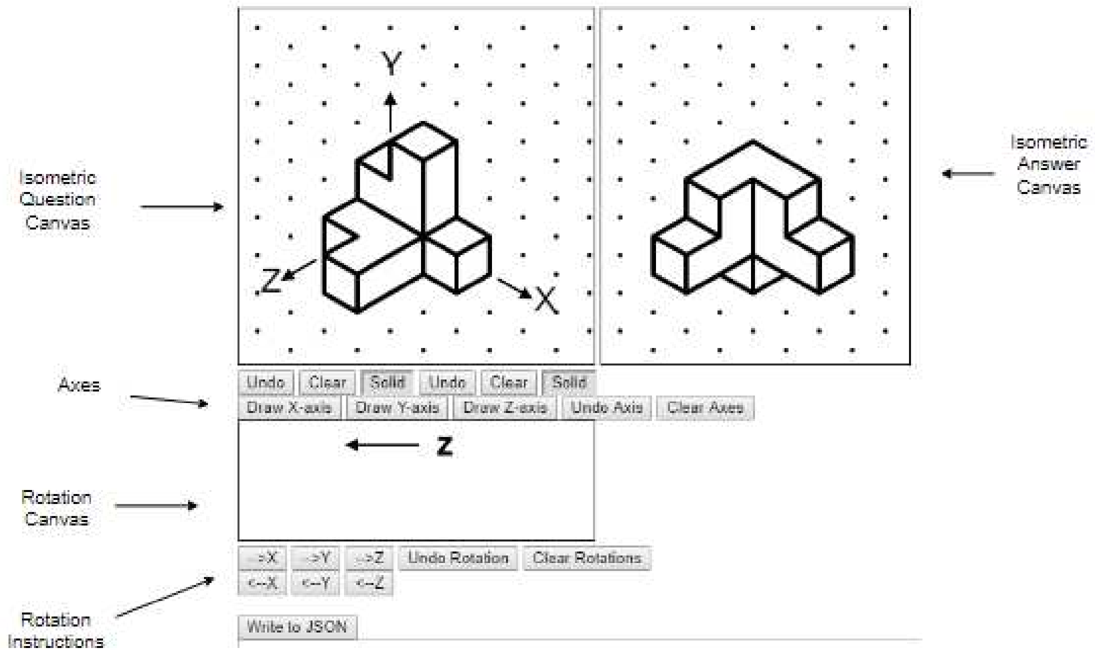


Each Canvas has its own Undo and Clear buttons to undo any lines drawn.

There are also x, y and z axis buttons to add an axis arrow to the question canvas. To do this, click one of the axis buttons, e.g. ‘Draw Blank X-axis’, then click on the canvas to draw an axis arrow which will point away from the click location. Similarly, there are corresponding buttons to enable the drawing of axis trails on the question canvas.

There is also a rotation canvas which allows rotation instructions to be shown to the user. Click the rotation instruction buttons to add the corresponding values to the rotation canvas.

3.1.4 Exercise 4 – Admin Page

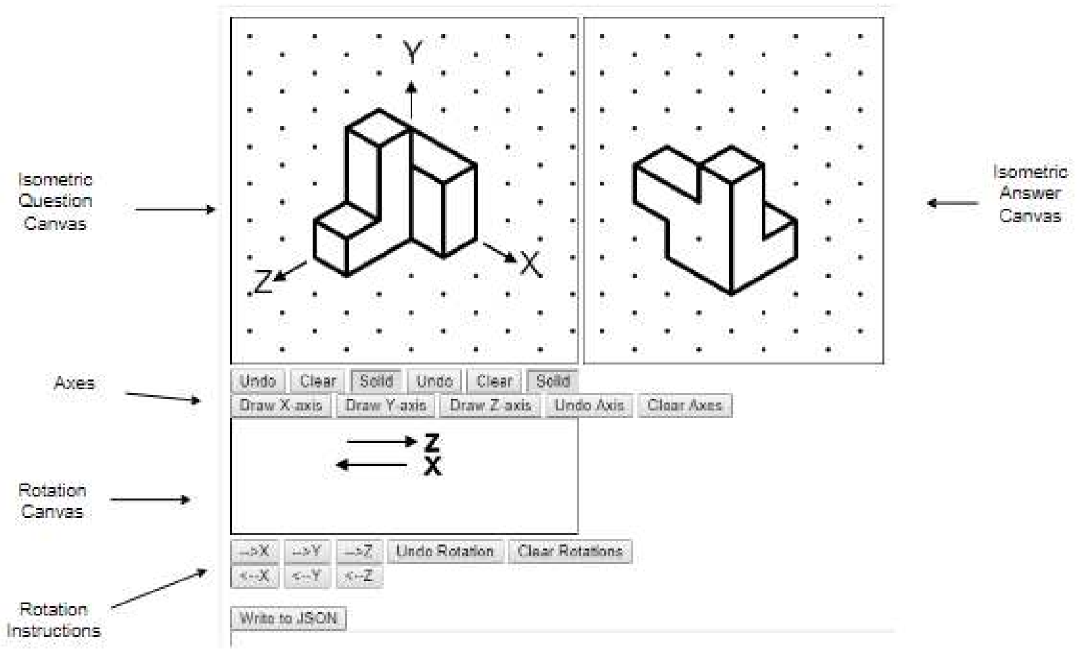


Each Canvas has its own Undo and Clear buttons to undo any lines drawn.

There are also x, y and z axis buttons to add an axis arrow to the question canvas. To do this, click one of the axis buttons, e.g. ‘Draw Blank X-axis’, then click on the canvas to draw an axis arrow which will point away from the click location.

There is also a rotation canvas which allows rotation instructions to be shown to the user. Click the rotation instruction buttons to add the corresponding values to the rotation canvas.

3.1.5 Exercise 5 – Admin Page

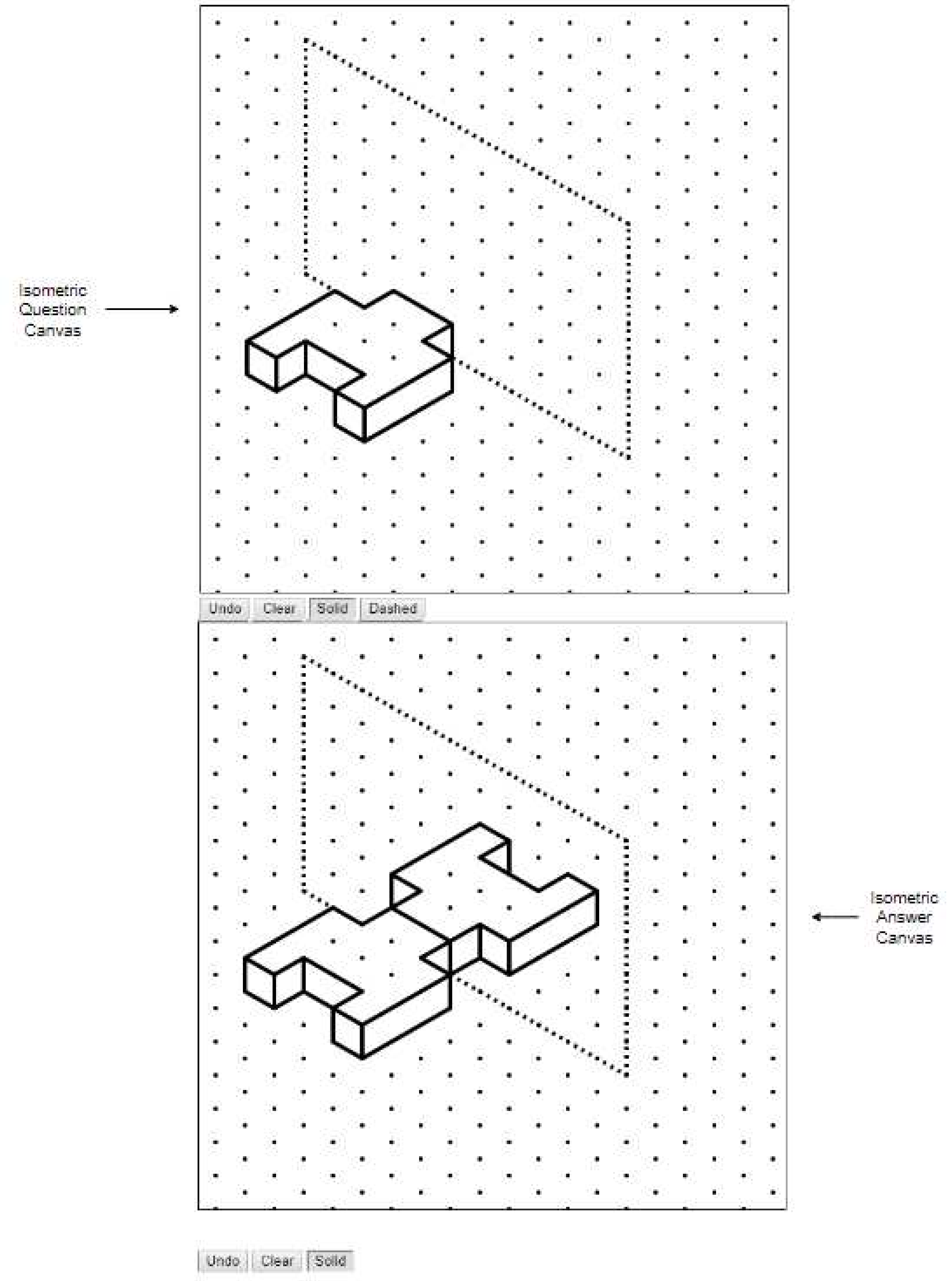


Each Canvas has its own Undo and Clear buttons to undo any lines drawn.

There are also x, y and z axis buttons to add an axis arrow to the question canvas. To do this, click one of the axis buttons, e.g. 'Draw Blank X-axis', then click on the canvas to draw an axis arrow which will point away from the click location.

There is also a rotation canvas which allows rotation instructions to be shown to the user. Click the rotation instruction buttons to add the corresponding values to the rotation canvas.

3.1.6 Exercise 6 – Admin Page

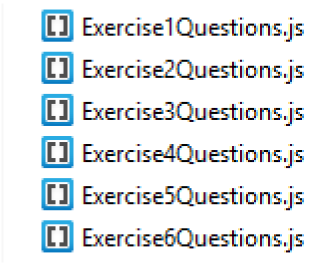


Each Canvas has its own Undo and Clear buttons to undo any lines drawn.

Drawing lines on the question canvas will create duplicate lines on the answer canvas. This is to aid drawing the answer lines and will not actually be part of the answer.

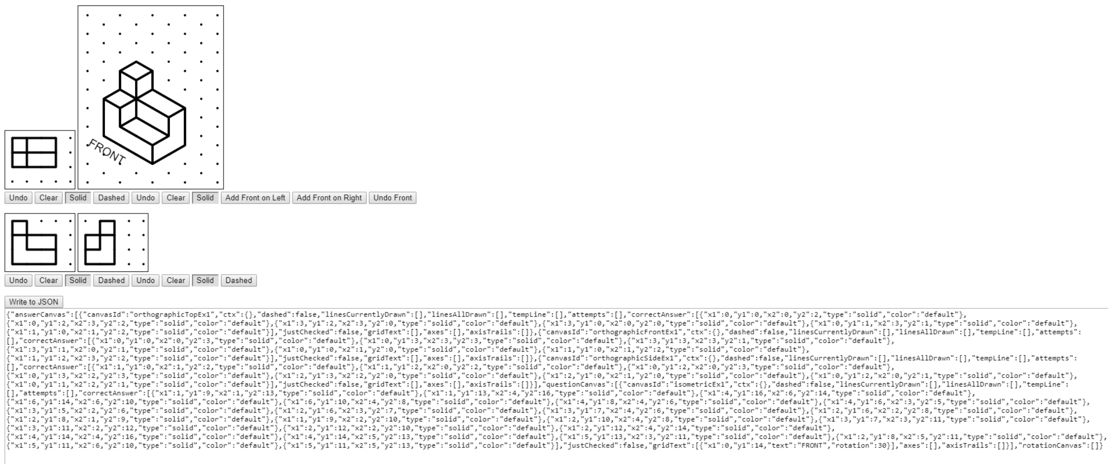
3.2 Saving a Question from Admin Pages with JSON

Every admin html page has a button with the text ‘Write to JSON’. This is because the questions are stored as JSON (JavaScript Object Notation) in the /js folder:



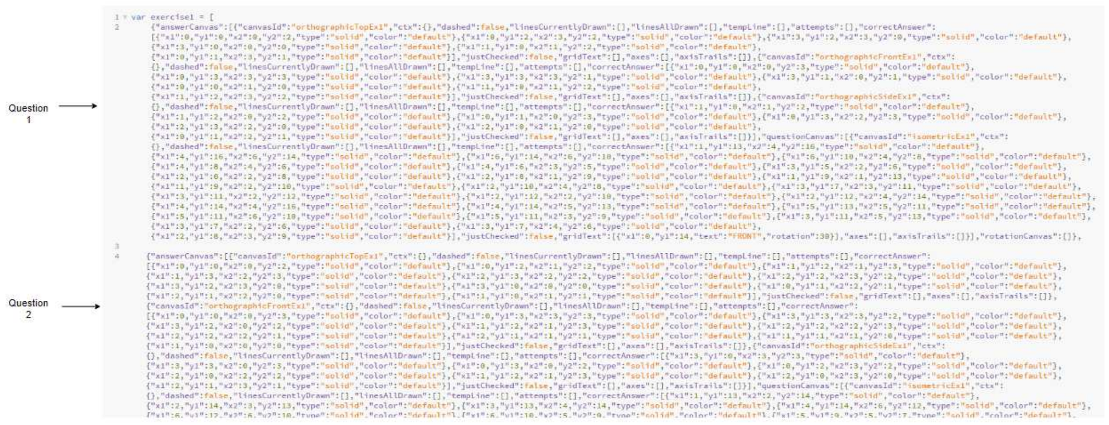
Exercise1Questions.js
Exercise2Questions.js
Exercise3Questions.js
Exercise4Questions.js
Exercise5Questions.js
Exercise6Questions.js

In order to add a newly created question, click the ‘Write To JSON’ button to display the JSON data, for example with the Exercise 1 admin page:



```
{"answerCanvas":[{"canvasId":"orthographicTopEx1","ctx":{},"dashed":false,"linesCurrentlyDrawn":[],"linesAllDrawn":[],"tempLine":[],"attempts":[],"correctAnswer":[{"x1":0,"y1":0,"x2":0,"y2":2,"type":"solid","color":"default"},
{"x1":0,"y1":2,"x2":3,"y2":2,"type":"solid","color":"default"},{"x1":3,"y1":2,"x2":3,"y2":0,"type":"solid","color":"default"},{"x1":3,"y1":0,"x2":0,"y2":0,"type":"solid","color":"default"},{"x1":0,"y1":1,"x2":3,"y2":1,"type":"solid","color":"default"},
{"x1":1,"y1":0,"x2":1,"y2":2,"type":"solid","color":"default"}],"justChecked":false,"gridText":[],"axes":[],"axisTrails":[]},{"canvasId":"orthographicFrontEx1","ctx":{},"dashed":false,"linesCurrentlyDrawn":[],"linesAllDrawn":[],"tempLine":[],"attempts":
[],"correctAnswer":[{"x1":0,"y1":0,"x2":0,"y2":3,"type":"solid","color":"default"},{"x1":0,"y1":3,"x2":3,"y2":3,"type":"solid","color":"default"},{"x1":3,"y1":3,"x2":3,"y2":1,"type":"solid","color":"default"},
{"x1":3,"y1":1,"x2":0,"y2":1,"type":"solid","color":"default"},{"x1":0,"y1":0,"x2":1,"y2":0,"type":"solid","color":"default"},{"x1":1,"y1":0,"x2":1,"y2":2,"type":"solid","color":"default"},
{"x1":1,"y1":2,"x2":3,"y2":2,"type":"solid","color":"default"}],"justChecked":false,"gridText":[],"axes":[],"axisTrails":[]},{"canvasId":"orthographicSideEx1","ctx":{},"dashed":false,"linesCurrentlyDrawn":[],"linesAllDrawn":[],"tempLine":[],"attempts":
[],"correctAnswer":[{"x1":1,"y1":0,"x2":1,"y2":2,"type":"solid","color":"default"},{"x1":1,"y1":2,"x2":0,"y2":2,"type":"solid","color":"default"},{"x1":0,"y1":2,"x2":0,"y2":3,"type":"solid","color":"default"},
{"x1":0,"y1":3,"x2":2,"y2":3,"type":"solid","color":"default"},{"x1":2,"y1":3,"x2":2,"y2":0,"type":"solid","color":"default"},{"x1":2,"y1":0,"x2":1,"y2":0,"type":"solid","color":"default"},{"x1":0,"y1":2,"x2":0,"y2":1,"type":"solid","color":"default"},
{"x1":0,"y1":1,"x2":2,"y2":1,"type":"solid","color":"default"}],"justChecked":false,"gridText":[],"axes":[],"axisTrails":[]}],"questionCanvas":[{"canvasId":"isometricEx1","ctx":{},"dashed":false,"linesCurrentlyDrawn":[],"linesAllDrawn":[],"tempLine":
[],"attempts":[],"correctAnswer":[{"x1":1,"y1":9,"x2":1,"y2":13,"type":"solid","color":"default"},{"x1":1,"y1":13,"x2":4,"y2":16,"type":"solid","color":"default"},{"x1":4,"y1":16,"x2":6,"y2":14,"type":"solid","color":"default"},
{"x1":6,"y1":14,"x2":6,"y2":10,"type":"solid","color":"default"},{"x1":6,"y1":10,"x2":4,"y2":8,"type":"solid","color":"default"},{"x1":4,"y1":8,"x2":4,"y2":6,"type":"solid","color":"default"},{"x1":4,"y1":6,"x2":3,"y2":5,"type":"solid","color":"default"},
{"x1":3,"y1":5,"x2":2,"y2":6,"type":"solid","color":"default"},{"x1":2,"y1":6,"x2":3,"y2":7,"type":"solid","color":"default"},{"x1":3,"y1":7,"x2":4,"y2":6,"type":"solid","color":"default"},{"x1":2,"y1":6,"x2":2,"y2":8,"type":"solid","color":"default"},
{"x1":2,"y1":8,"x2":1,"y2":9,"type":"solid","color":"default"},{"x1":1,"y1":9,"x2":2,"y2":10,"type":"solid","color":"default"},{"x1":2,"y1":10,"x2":4,"y2":8,"type":"solid","color":"default"},{"x1":3,"y1":7,"x2":3,"y2":11,"type":"solid","color":"default"},
{"x1":3,"y1":11,"x2":2,"y2":12,"type":"solid","color":"default"},{"x1":2,"y1":12,"x2":2,"y2":10,"type":"solid","color":"default"},{"x1":2,"y1":12,"x2":4,"y2":14,"type":"solid","color":"default"},
{"x1":4,"y1":14,"x2":4,"y2":16,"type":"solid","color":"default"},{"x1":4,"y1":14,"x2":5,"y2":13,"type":"solid","color":"default"},{"x1":5,"y1":13,"x2":3,"y2":11,"type":"solid","color":"default"},{"x1":2,"y1":8,"x2":5,"y2":11,"type":"solid","color":"default"},
{"x1":5,"y1":11,"x2":6,"y2":10,"type":"solid","color":"default"},{"x1":5,"y1":11,"x2":5,"y2":13,"type":"solid","color":"default"}],"justChecked":false,"gridText":[{"x1":0,"y1":14,"text":"FRONT","rotation":30}],"axes":[],"axisTrails":[]}],"rotationCanvas":[]}
```

Exit the Chrome Developer Tools and copy the JSON text. Open the relevant exercise question JavaScript file, e.g. Exercise1Questions.js, and paste in the question at the end of the exercise array (e.g. exercise1) inside the file, making sure that questions are separated by commas:



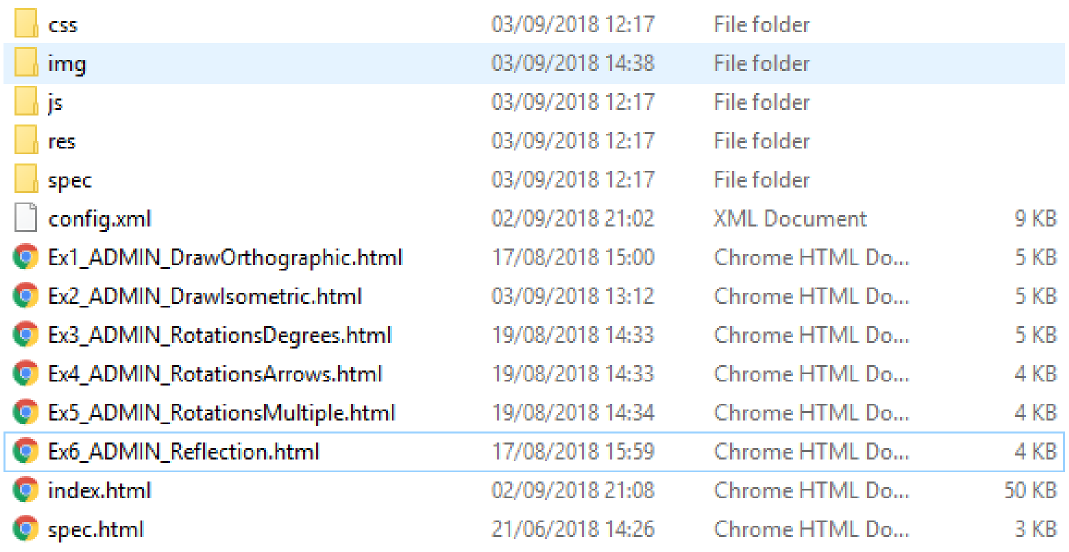
```
var exercise1 = [
    {"answerCanvas":[{"canvasId":"orthographicTopEx1","ctx":{},"dashed":false,"linesCurrentlyDrawn":[],"linesAllDrawn":[],"tempLine":[],"attempts":[],"correctAnswer":
    [{"x1":0,"y1":0,"x2":0,"y2":2,"type":"solid","color":"default"},{"x1":0,"y1":2,"x2":3,"y2":2,"type":"solid","color":"default"},{"x1":3,"y1":2,"x2":3,"y2":0,"type":"solid","color":"default"},
    {"x1":3,"y1":0,"x2":0,"y2":0,"type":"solid","color":"default"},{"x1":1,"y1":0,"x2":1,"y2":2,"type":"solid","color":"default"},
    {"x1":0,"y1":1,"x2":3,"y2":1,"type":"solid","color":"default"}],"justChecked":false,"gridText":[],"axes":[],"axisTrails":[]},{"canvasId":"orthographicFrontEx1","ctx":
    {},"dashed":false,"linesCurrentlyDrawn":[],"linesAllDrawn":[],"tempLine":[],"attempts":[],"correctAnswer":[{"x1":0,"y1":0,"x2":0,"y2":3,"type":"solid","color":"default"},
    {"x1":0,"y1":3,"x2":3,"y2":3,"type":"solid","color":"default"},{"x1":3,"y1":3,"x2":3,"y2":1,"type":"solid","color":"default"},{"x1":3,"y1":1,"x2":0,"y2":1,"type":"solid","color":"default"},
    {"x1":0,"y1":0,"x2":1,"y2":0,"type":"solid","color":"default"},{"x1":1,"y1":0,"x2":1,"y2":2,"type":"solid","color":"default"},
    {"x1":1,"y1":2,"x2":3,"y2":2,"type":"solid","color":"default"}],"justChecked":false,"gridText":[],"axes":[],"axisTrails":[]},{"canvasId":"orthographicSideEx1","ctx":
    {},"dashed":false,"linesCurrentlyDrawn":[],"linesAllDrawn":[],"tempLine":[],"attempts":[],"correctAnswer":[{"x1":1,"y1":0,"x2":1,"y2":2,"type":"solid","color":"default"},
    {"x1":1,"y1":2,"x2":0,"y2":2,"type":"solid","color":"default"},{"x1":0,"y1":1,"x2":0,"y2":3,"type":"solid","color":"default"},{"x1":0,"y1":3,"x2":2,"y2":3,"type":"solid","color":"default"},
    {"x1":2,"y1":3,"x2":2,"y2":0,"type":"solid","color":"default"},{"x1":2,"y1":0,"x2":1,"y2":0,"type":"solid","color":"default"},
    {"x1":0,"y1":1,"x2":2,"y2":1,"type":"solid","color":"default"}],"justChecked":false,"gridText":[],"axes":[],"axisTrails":[]}],"questionCanvas":[{"canvasId":"isometricEx1","ctx":
    {},"dashed":false,"linesCurrentlyDrawn":[],"linesAllDrawn":[],"tempLine":[],"attempts":[],"correctAnswer":[{"x1":1,"y1":13,"x2":4,"y2":16,"type":"solid","color":"default"},
    {"x1":4,"y1":16,"x2":6,"y2":14,"type":"solid","color":"default"},{"x1":6,"y1":14,"x2":6,"y2":10,"type":"solid","color":"default"},{"x1":6,"y1":10,"x2":4,"y2":8,"type":"solid","color":"default"},
    {"x1":4,"y1":8,"x2":4,"y2":6,"type":"solid","color":"default"},{"x1":4,"y1":6,"x2":3,"y2":5,"type":"solid","color":"default"},{"x1":3,"y1":5,"x2":2,"y2":6,"type":"solid","color":"default"},
    {"x1":2,"y1":6,"x2":2,"y2":8,"type":"solid","color":"default"},{"x1":2,"y1":8,"x2":1,"y2":9,"type":"solid","color":"default"},{"x1":1,"y1":9,"x2":1,"y2":13,"type":"solid","color":"default"},
    {"x1":1,"y1":9,"x2":2,"y2":10,"type":"solid","color":"default"},{"x1":2,"y1":10,"x2":4,"y2":8,"type":"solid","color":"default"},{"x1":3,"y1":7,"x2":3,"y2":11,"type":"solid","color":"default"},
    {"x1":3,"y1":11,"x2":2,"y2":12,"type":"solid","color":"default"},{"x1":2,"y1":12,"x2":2,"y2":10,"type":"solid","color":"default"},{"x1":2,"y1":12,"x2":4,"y2":14,"type":"solid","color":"default"},
    {"x1":4,"y1":14,"x2":4,"y2":16,"type":"solid","color":"default"},{"x1":4,"y1":14,"x2":5,"y2":13,"type":"solid","color":"default"},{"x1":5,"y1":13,"x2":5,"y2":11,"type":"solid","color":"default"},
    {"x1":5,"y1":11,"x2":6,"y2":10,"type":"solid","color":"default"},{"x1":5,"y1":11,"x2":3,"y2":9,"type":"solid","color":"default"},{"x1":3,"y1":11,"x2":5,"y2":13,"type":"solid","color":"default"},
    {"x1":3,"y1":7,"x2":2,"y2":6,"type":"solid","color":"default"},{"x1":3,"y1":7,"x2":4,"y2":6,"type":"solid","color":"default"},
    {"x1":2,"y1":8,"x2":3,"y2":9,"type":"solid","color":"default"}],"justChecked":false,"gridText":[{"x1":0,"y1":14,"text":"FRONT","rotation":30}],"axes":[],"axisTrails":[]}],"rotationCanvas":[]},

   {"answerCanvas":[{"canvasId":"orthographicTopEx1","ctx":{},"dashed":false,"linesCurrentlyDrawn":[],"linesAllDrawn":[],"tempLine":[],"attempts":[],"correctAnswer":
   [{"x1":0,"y1":0,"x2":0,"y2":2,"type":"solid","color":"default"},{"x1":0,"y1":2,"x2":1,"y2":2,"type":"solid","color":"default"},{"x1":1,"y1":2,"x2":1,"y2":3,"type":"solid","color":"default"},
   {"x1":1,"y1":3,"x2":2,"y2":3,"type":"solid","color":"default"},{"x1":2,"y1":3,"x2":2,"y2":2,"type":"solid","color":"default"},{"x1":2,"y1":2,"x2":3,"y2":2,"type":"solid","color":"default"},
   {"x1":3,"y1":2,"x2":3,"y2":0,"type":"solid","color":"default"},{"x1":3,"y1":0,"x2":0,"y2":0,"type":"solid","color":"default"},{"x1":0,"y1":1,"x2":2,"y2":1,"type":"solid","color":"default"},
   {"x1":2,"y1":1,"x2":2,"y2":0,"type":"solid","color":"default"},{"x1":1,"y1":0,"x2":1,"y2":1,"type":"solid","color":"default"}],"justChecked":false,"gridText":[],"axes":[],"axisTrails":[]},
   {"canvasId":"orthographicFrontEx1","ctx":{},"dashed":false,"linesCurrentlyDrawn":[],"linesAllDrawn":[],"tempLine":[],"attempts":[],"correctAnswer":
   [{"x1":0,"y1":0,"x2":0,"y2":3,"type":"solid","color":"default"},{"x1":0,"y1":3,"x2":3,"y2":3,"type":"solid","color":"default"},{"x1":3,"y1":3,"x2":3,"y2":2,"type":"solid","color":"default"},
   {"x1":3,"y1":2,"x2":0,"y2":2,"type":"solid","color":"default"},{"x1":1,"y1":2,"x2":1,"y2":3,"type":"solid","color":"default"},{"x1":2,"y1":2,"x2":2,"y2":3,"type":"solid","color":"default"},
   {"x1":2,"y1":2,"x2":2,"y2":1,"type":"solid","color":"default"},{"x1":2,"y1":1,"x2":1,"y2":1,"type":"solid","color":"default"},{"x1":1,"y1":1,"x2":1,"y2":0,"type":"solid","color":"default"},
   {"x1":1,"y1":0,"x2":0,"y2":0,"type":"solid","color":"default"}],"justChecked":false,"gridText":[],"axes":[],"axisTrails":[]},{"canvasId":"orthographicSideEx1","ctx":
   {},"dashed":false,"linesCurrentlyDrawn":[],"linesAllDrawn":[],"tempLine":[],"attempts":[],"correctAnswer":[{"x1":3,"y1":0,"x2":3,"y2":3,"type":"solid","color":"default"},
   {"x1":3,"y1":3,"x2":0,"y2":3,"type":"solid","color":"default"},{"x1":0,"y1":3,"x2":0,"y2":2,"type":"solid","color":"default"},{"x1":0,"y1":2,"x2":3,"y2":2,"type":"solid","color":"default"},
   {"x1":2,"y1":0,"x2":2,"y2":2,"type":"solid","color":"default"},{"x1":1,"y1":2,"x2":1,"y2":3,"type":"solid","color":"default"},{"x1":2,"y1":0,"x2":3,"y2":0,"type":"solid","color":"default"},
   {"x1":2,"y1":1,"x2":3,"y2":1,"type":"solid","color":"default"}],"justChecked":false,"gridText":[],"axes":[],"axisTrails":[]}],"questionCanvas":[{"canvasId":"isometricEx1","ctx":
   {},"dashed":false,"linesCurrentlyDrawn":[],"linesAllDrawn":[],"tempLine":[],"attempts":[],"correctAnswer":[{"x1":1,"y1":13,"x2":2,"y2":14,"type":"solid","color":"default"},
   {"x1":2,"y1":14,"x2":3,"y2":13,"type":"solid","color":"default"},{"x1":3,"y1":13,"x2":4,"y2":14,"type":"solid","color":"default"},{"x1":4,"y1":14,"x2":6,"y2":12,"type":"solid","color":"default"},
   {"x1":6,"y1":12,"x2":6,"y2":10,"type":"solid","color":"default"},{"x1":6,"y1":10,"x2":5,"y2":9,"type":"solid","color":"default"},{"x1":5,"y1":9,"x2":5,"y2":7,"type":"solid","color":"default"},
```

To remove a question, just delete it from the array. At present there is no way to modify an existing question so just create a new question and replace.

4.0 File Structure

4.1 www Folder

The code for the app is located within the www folder. The following files and folders exist here:



css	03/09/2018 12:17	File folder	
img	03/09/2018 14:38	File folder	
js	03/09/2018 12:17	File folder	
res	03/09/2018 12:17	File folder	
spec	03/09/2018 12:17	File folder	
config.xml	02/09/2018 21:02	XML Document	9 KB
Ex1_ADMIN_DrawOrthographic.html	17/08/2018 15:00	Chrome HTML Do...	5 KB
Ex2_ADMIN_DrawIsometric.html	03/09/2018 13:12	Chrome HTML Do...	5 KB
Ex3_ADMIN_RotationsDegrees.html	19/08/2018 14:33	Chrome HTML Do...	5 KB
Ex4_ADMIN_RotationsArrows.html	19/08/2018 14:33	Chrome HTML Do...	4 KB
Ex5_ADMIN_RotationsMultiple.html	19/08/2018 14:34	Chrome HTML Do...	4 KB
Ex6_ADMIN_Reflection.html	17/08/2018 15:59	Chrome HTML Do...	4 KB
index.html	02/09/2018 21:08	Chrome HTML Do...	50 KB
spec.html	21/06/2018 14:26	Chrome HTML Do...	3 KB

css folder – css files used by the app are stored here.
img folder – images used by the app, e.g. in pop ups, are stored here
js folder – JavaScript files are stored here
res – resources folder currently stores default app icons
spec – files to help with app events such as deviceready

config.xml – file used to specify name of app and any plugins
Ex<number>_ADMIN_<desc>.html – html pages used to create questions
index.html – html for the app

4.2 js Folder – JavaScript Scripts, Functions and variables

This folder contains the JavaScript logic for the functioning of the application. Below is a summary of the important files, why they are necessary and the important functions contained within them.

4.2.1 AddTouch.js

Functionality used by the app to enable touch on canvases, with separate touch functions for Isometric and Orthographic canvases.

enableTouch(canvas) – function used to add touchstart eventListener on the isometric canvas passed in as a parameter

disableTouch(canvas) – function used to remove touchstart eventListener on the isometric canvas passed in as a parameter

enableTouchOrth(canvas) – function used to add touchstart eventListener on the orthographic canvas passed in as a parameter

disableTouchOrth(canvas) – function used to remove touchstart eventListener on the orthographic canvas passed in as a parameter

4.2.2 AddTouch_Admin.js

Functionality used by the Admin html pages to enable touch on canvases, with separate touch functions for Isometric and Orthographic canvases.

enableTouch(canvas) - function used to add touchstart eventListener on the isometric canvas passed in as a parameter

disableTouch(canvas) – function used to remove touchstart eventListener on the isometric canvas passed in as a parameter

handleStart(evt) – function used to deal with "touchstart" event i.e. respond to a touch on an isometric canvas

enableTouchOrth(canvas) – function used to add touchstart eventListener on the orthographic canvas passed in as a parameter

disableTouchOrth(canvas) – function used to remove touchstart eventListener on the orthographic canvas passed in as a parameter

handleStartOrth(evt) – function used to deal with "touchstart" event i.e. respond to a touch on an orthographic canvas

enableTouchText(canvas, text, rotation) – function which removes touchstart eventListener on an isometric canvas and allows the admin to add text to a canvas by adding a new touchstart eventListener. The canvas parameter specifies the isometric canvas, the text parameter specifies the text e.g. "FRONT" and the rotation is the angle at which the text is drawn on the canvas

disableTouchText(canvas, text, rotation) – function to remove touchstart eventListener which adds text to the isometric canvas and adds new touchstart eventListener by calling enableTouch(canvas). The canvas parameter specifies the isometric canvas, the text parameter specifies the text e.g. "FRONT" and the rotation is the angle at which the text is drawn on the canvas.

enableTouchAxes(canvas, axis, axisLabel) – function which removes touchstart eventListener on an isometric canvas and allows the admin to add axis to a canvas by adding a new touchstart eventListener. The canvas parameter specifies the isometric canvas, the axis parameter specifies the axis e.g. X and the axisLabel is the text accompanying the axis e.g. ‘X’ or ‘blank’.

disableTouchAxes(canvas, axis, axisLabel) – function to remove touchstart which adds axis to the isometric canvas and adds a new touchstart eventListener by calling enableTouch(canvas). The canvas parameter specifies the isometric canvas, the axis parameter specifies the axis e.g. X, and the axisLabel is the text accompanying the axis e.g. ‘X’ or ‘blank’.

enableTouchTrails(canvas, axis) – function which removes touchstart eventListener on an isometric canvas to allow the admin to add axis trail to a canvas by adding a new touchstart eventListener. The canvas parameter specifies the isometric canvas, the axis parameter specifies the axis e.g. X.

disableTouchTrails(canvas, axis) – function to remove touchstart eventListener which adds axis to the isometric canvas and adds a new touchstart eventListener by calling enableTouch(canvas). The canvas parameter specifies the isometric canvas, the axis parameter specifies the axis e.g. X.

enableTouchMirror(canvas, mirrorCanvasObject) – function which adds touchstart eventlistener on the isometric canvas passed in as a parameter to allow lines to be drawn on two canvases. The mirrorCanvasObject parameter is another canvas on which lines drawn on the canvas parameter will be replicated.

disableTouchMirror(canvas, axis) – function to remove touchstart eventListener which allows lines drawn on canvas to be replicated on the mirrorCanvasObject. The canvas parameter is the canvas which is drawn on. The mirrorCanvasObject is the canvas where lines will be replicated on.

handleStartText(evt, text, rotation) – function to handle touchstart event by calling the addGridText function. The evt parameter is the touchstart event, the text parameter is the text which will be added to the canvas and the rotation parameter is the angle at which the text will be drawn onto the canvas.

handleStartAxes(evt, axis, axisLabel) – function to handle touchstart event by calling the addGridAxis function. The evt parameter is the touchstart event, the axis parameter is the axis to be added to the grid e.g. X, and the axisLabel determines the label given to the axis e.g. ‘X’.

handleStartTrails(evt, axis) – function to handle touchstart event by calling the addGridTrails function. The evt parameter is the touchstart event and the axis parameter is the axis on which the trail should be drawn.

handleStartMirror(evt, mirrorCanvasObject) – function to handle touchstart event by calling addPoint function for current canvas and canvas to be mirrored on.

4.2.3 AlphabeticRotation.js

AlphabeticRotation(direction, axis) – Prototype for the AlphabeticRotation object. This is required for arrow code rotation information displayed in the rotation canvas in rotation questions. The direction parameter denotes whether the rotation is 'positive' or 'negative' and the axis parameter denotes the axis X, Y or Z.

4.2.4 Axis.js

Axis(x1, y1, axis, axisLabel) - Prototype for Axis object. This is required for axes in rotation questions. Parameters x1 and y1 note the start coordinates of the axis on the canvas, the axis paarmeter is X, Y or Z the axisLabel parameter is the text used to label the axis, either 'X', 'Y', 'Z' or 'blank'

4.2.5 AxisTrail.js

AxisTrail(x1, y1, axis) - Prototype for AxisTrail object. This is required for axes in rotation questions. Parameters x1 and y1 note the start coordinates of the axis trail on the canvas, the axis parameter is X, Y or Z.

4.2.6 CanvasObject.js

CanvasObject(canvasId) - Prototype for canvasObject object. This is required for question and answer canvases. There are some very important properties of the CanvasObject:

canvasId – the html id tag of the canvas
dashed – Boolean used to determine whether dashed lines or solid lines are being drawn on the canvas currently.
linesCurrentlyDrawn - an array of all Line objects currently drawn on the canvas.
tempLine – used to help prepare next Line object drawn.
linesAllDrawn – an array of all Line objects drawn on the canvas, even deleted ones.
attempts – an array of user attempts
correctAnswer – an array which holds the correct answer made up of lines if this is an answer canvas, or an array which holds the predrawn shape for a question if this is a question canvas.
correct – a Boolean used to mark question as correct
gridText – used to store GridText object with text to be drawn on canvas e.g. “FRONT”.
axes – used to store Axis objects which store any axes to be drawn on canvas.
axisTrails – used to store AxisTrail objects which store axis trails to be drawn on canvas.

4.2.7 Exercise.js

Exercise(name) – Prototype for the Exercise object. This is required to represent an exercise in the app. There are some important properties of the Exercise object:

name – this is the name of the exercise
questions – this is an array used to store Question objects which hold question, answer and rotation canvasObjects.
currentQuestion – numeric value to keep track of current Question
numRight – numeric value to keep track of number of correct Questions
numWrong – numeric value used to keep track of incorrectQuestions

4.2.8 Exercise1Questions.js

exercise1 – an array which holds Question objects for exercise 1 as JSON objects

4.2.9 Exercise2Questions.js

exercise2 – an array which holds Question objects for exercise 2 as JSON objects

4.2.10 Exercise3Questions.js

exercise3 – an array which holds Question objects for exercise 3 as JSON objects

4.2.11 Exercise4Questions.js

exercise4 – an array which holds Question objects for exercise 4 as JSON objects

4.2.12 Exercise5Questions.js

exercise5 – an array which holds Question objects for exercise 5 as JSON objects

4.2.13 Exercise6Questions.js

exercise6 – an array which holds Question objects for exercise 6 as JSON objects

4.2.14 ExercisesSetup.js

setUpExercises() – function which sets up each Exercise object, sets up Question 1 by calling setupQuestion(exercise) and adds to an exercises array.

4.2.15 GridText.js

GridText(x1, y1, text, rotation) – Prototype for GridText object. This is needed for any text that has to be displayed on the canvas e.g. 'FRONT'. Parameters x1 and y1 note the start coordinates of the text on the canvas, the text parameter is used to store the text that will be displayed and the rotation parameter is used to indicate the rotation in degrees of the text on the canvas

4.2.16 HomePageLoad.js

Script used before Home page in app loads. Loops through all Questions in all Exercises and tallies up attempted and correct Questions before adding them to the table on this page.

4.2.17 Index.js

Script used to bind events that are required during the lifecycle of the app, for example ‘deviceready’.

4.2.18 IsometricAnswers.js

Script with functionality which helps to create and check answers for isometric grids.

createAnswerFromArray(canvasObject, inputArray) – function used to create a correctAnswer array of Line objects within the given canvasObject parameter, based on the given inputArray parameter. The canvasObject is an isometric canvasObject and the inputArray parameter is an array with the form [x1, y1, x2, y2, type], where x and y cords are integers and type is ‘solid’ or ‘dashed’.

checkAnswer(canvasObject, answerId) – function to check that current drawing matches answer. canvasObject parameter is the canvas on which the check is being made, and which there is a stored correctAnswer. The answerId parameter is the html id tag indicating the element which should display feedback to the user. Uses breakUpAllLines function and findBottomLeftPoint function.

findBottomLeftPoint(lineArray) – function used to find the bottom left point of an array of Line objects. This is the point from which a check can then be made to compare the lines a user has drawn with a stored answer.

breakUpAllLines(lineArray) – function which takes an array of Line objects and breaks it up into the smallest lines possible. This is necessary in order to compare identical lines which may have been drawn differently e.g. many small lines could make the same line as one big line.

breakUpLine(line) – function used to take in a Line object and breakup the line into smaller Line objects if necessary and return any new Line objects in an array.

4.2.19 IsometricDistances.js

Script which contains an initial value in pixels for the distance between dots vertically and diagonally in an isometric grid, and from which other distances are automatically figured out. This script also contains named values associated with axes and axis trails such as line widths, arrow lengths, colours etc.

lengthOfSide(hypotenuse, side) – function used to work out the length of one side of a triangle, based on hypotenuse and side parameters.

lengthOfHypotenuse(side1, side2) – function used to work out the length of the hypotenuse of a triangle, based on the side 1 and side 2 parameters

findClosestCoord(x, y) – function used to find and return the closest dot coordinate based on the true touch coorinate parameters x and y. Makes use of the findXCoord and findYCoord functions.

findXCoord(x) – function used to find the simple x coordinate based on the true x coordinate parameter passed in.

findYCoord(y) – function used to find the simple y coordinate based upon the true y coordinate parameter passed in.

findTrueXCoord(x) – function used to find the true x coordinate based upon the simple x coordinate passed in as a parameter.

findTrueYCoord(y) – function used to find the true y coordinate based upon the simple y coordinate passed in as a parameter.

drawDots(canvasObject) – function used to draw dots on the canvas represented by the canvasObject passed in as a parameter. Clears the canvas before this however.

clearCanvas(canvasObject) – function to clear a canvas represented by the canvasObject parameter passed in.

addPoint(x, y, canvasObject) – function to add touch to grid and possibly draw a new line.

drawACircle(canvasObject, x, y, radius, color) – function used to draw a circle on the canvas represented by the canvasObject parameter passed in. Used by drawDots function and addPoint function to represent touch point to user. x and y parameters are used to specify centre of circle, radius parameter specifies the radius in pixels and color represents the color of the circle to be drawn.

drawLines(canvasObject, lineArray) – function used to draw all Line objects in the lineArray parameter, on the canvas represented by the canvasObject parameter. Makes used of the drawALine function. Also has to call the drawText, drawAxes and drawTrails functions as the canvas is refreshed everytime it is drawn upon.

drawALine(ctx, line) – function used to draw the Line object represented by the line parameter, onto the canvas context represented by the ctx parameter.

undoLine(canvasObject) – function used to undo the last Line object drawn on the canvas represented by the given canvasObject parameter.

clearLines(canvasObject) – function used to clear all Line objects drawn on the canvas represented by the given canvasObject parameter.

addGridText(canvasObject, x, y, text, rotation) – function to add text to a canvasObject. The parameters x and y represent the coordinates at which the text shall be added, the text parameter represents the String of text and the rotation parameter represents the rotation of the text in radians.

drawText(canvasObject) – function used to actually draw text based on gridText stored within given canvasObject parameter.

undoText(canvasObject) – function to undo the last text drawn on canvas represented by the canvasObject parameter.

addNumericRotation(rotationCanvas, degrees) – function used to add numeric rotation instruction to a rotationCanvas. The parameter degrees represents the value of the numeric rotation. Calls drawNumericRotations function to draw instruction.

undoNumericRotation(rotationCanvas, degrees) – function to undo a numeric rotation instruction for the canvas represented by the rotationCanvas parameter.

clearNumericRotations(rotationCanvas) – function to clear numeric rotation instructions for the canvas represented by the rotationCanvas parameter.

clearRotationCanvas(rotationCanvas) – function to visibly clear the canvas represented by the rotationCanvas parameter.

drawNumericRotation(rotationCanvas) – function to draw the rotation instructions for a given canvasObject represented by the rotationCanvas parameter.

addAlphabeticRotation(rotationCanvas) – function to add an alphabetic rotation instruction to a canvasObject represented by the rotationCanvas parameter.

undoAlphabeticRotation(rotationCanvas) – function to undo an alphabetic rotation instruction for the canvasObject represented by the rotationCanvas parameter.

clearAlphabeticRotations(rotationCanvas) – function to clear alphabetic rotation instructions for a canvasObject represented by the rotationCanvas parameter.

drawAlphabeticRotations(rotationCanvas) – function to draw the rotation instructions for the canvas represented by the canvasObject which is itself represented by the rotationCanvas parameter.

drawNegativeAlphabeticArrow(rotationCanvas, alphabeticRotation, width, height) – function to draw a negative arrow and alphabeticRotation on a canvas represented by the rotationCanvas parameter. Parameters width and height are used to position arrow on canvas.

drawPositiveAlphabeticArrow(rotationCanvas, alphabeticRotation, width, height) – function to draw a positive arrow and alphabeticRotation on a canvas represented by the rotationCanvas parameter. Parameters width and height are used to position arrow on canvas.

addGridAxis(canvasObject, x, y, axis, axisLabel) – function used to add axis to canvasObject represented by canvasObject parameter. Parameters x and y are the simple coordinates at which the axis will be drawn from, axis is the axis which is represented i.e. X, Y or Z and axisLabel is the label which will be drawn next to the axis.

drawAxes(canvasObject) – function to draw all axes on the canvas represented by the canvasObject parameter.

undoAxis(canvasObject) – function to undo an axis on the canvas represented by the canvasObject parameter.

clearAxes(canvasObject) – function to clear all axes on canvas represented by the canvasObject parameter.

addGridTrails(canvasObject, x, y, axis) – function to add axis trail information to canvasObject represented by canvasObject parameter. Parameters x and y represent the simple coordinates that the trail is drawn from and the parameter axis represents the axis along which the trail sits.

drawTrails(canvasObject) – function to draw axis trails on the canvas represented by the canvasObject parameter.

undoTrail(canvasObject) – function to undo an axis trail on the canvas represented by the canvasObject parameter.

clearTrails(canvasObject) – function to clear all axis trails on the canvas represented by the canvasObject parameter.

4.2.20 Line.js

Line(x1, y1, x2, y2) - Prototype for Line object. This is required to store the start and end x and y coordinates of a line drawn on a canvas. The parameters x1 and y1 note the start of the line on the canvas and x2 and y2 note the end of the line on the canvas. The parameter type is used to note whether the line is ‘solid’ or ‘dashed’ and color is used to specify denote what colour the line should be. Color is set to default as this allows colour to be changed elsewhere due to named value in drawing functions.

compareLines(line1, line2) – function used to compare whether two Line objects are equal based on x1, y1, x2 and y coordinates.

reverseLine(line) – function used to reverse the x1, y1 and x2, y2 pairs, effectively reversing the direction of the line. This is useful for comparisons between lines used by other functions.

4.2.21 LineProperties.js

This script holds many values together which are named and referenced by functions throughout the code. This makes this script a good place to easily change the values associated with these names rather than changing values within functions themselves. For example there are line width, line color, dot width, dot color, text style, text color, dash style, rotation instruction size and many more properties which can be changed.

4.2.22 Login.js

A sample script to simulate login and logout functionality. It would be expected that a more sophisticated login and logout functionality would be added at a later date.

4.2.23 Messages.js

A script which is used to store the feedback messages to the user after checking an answer. The messages are stored within named values in order that they can be changed easily if required.

4.2.24 NumericRotations.js

NumericRotation(degrees) - Prototype for NumericRotation object. This is required for rotation degree information displayed in rotation questions.

4.2.25 OrthographicAnswers.js

Script with functionality which helps to create and check answers for orthographic grids.

createAnswerFromArrayOrth(canvasObject, inputArray) – function used to create a correctAnswer array of Line objects within the given canvasObject parameter, based on the given inputArray parameter. The canvasObject is an orthographic canvasObject and the inputArray parameter is an array with the form [x1, y1, x2, y2, type], where x and y cords are integers and type is 'solid' or 'dashed'.

checkAnswerOrth(canvasObject, answerId) – function to check that current drawing matches answer. canvasObject parameter is the canvas on which the check is being made, and which there is a stored correctAnswer. The answerId parameter is the html id tag indicating the element which should display feedback to the user. Uses breakUpAllLinesOrth function and findBottomLeftPointOrth function.

findBottomLeftPointOrth(lineArray) – function used to find the bottom left point of an array of Line objects. This is the point from which a check can then be made to compare the lines a user has drawn with a stored answer.

breakUpAllLinesOrth(lineArray) – function which takes an array of Line objects and breaks it up into the smallest lines possible. This is necessary in order to compare identical lines which may have been drawn differently e.g. many small lines could make the same line as one big line.

breakUpLineOrth(line) – function used to take in a Line object and breakup the line into smaller Line objects if necessary and return any new Line objects in an array.

4.2.26 OrthographicDistances.js

Script which contains the horizontal and vertical distance in pixels between dots on an orthographic grid, represented by the variable distBetweenDotsOrth.

findClosestCoordOrth(x, y) – function to determine the closest dot coordinate based on the true touch x and y coordinates represented by the parameters x and y.

findXCoordOrth(x) – function to determine the simple x coordinate based on the true x coordinate parameter passed in.

findYCoordOrth(y) – function to determine the simple y coordinate based on the true y coordinate parameter passed in.

findTrueXCoordOrth(x) – function to determine the true x coordinate based on the simple x coordinate passed in as a parameter.

findTrueYCoordOrth(y) – function to find the true y coordinate based upon the simple y coordinate passed in as a parameter.

drawDotsOrth(canvasObject) – function to draw orthographic dots on the canvas represented by the canvasObject passed in as a parameter.

clearCanvasOrth(canvasObject) – function to clear the canvas represented by the canvasObject passed in as a parameter.

addPointOrth(x, y, canvasObject) – function to add touch to canvas represented by canvasObject passed in as a parameter, possibly resulting in a new line being drawn.

drawACircleOrth(canvasObject, x, y, radius, color) – function to draw a circle, used to represent a user touch and used by drawDotsOrth function.

drawLinesOrth(canvasObject, lineArray) – function to draw all Line objects in the given lineArray parameter on the canvas represented by the canvasObject parameter.

drawALineOrth(ctx, line) – function to draw a line on the canvas context ctx based on the Line object represented by the line parameter.

undoLineOrth(canvasObject) – function to undo a line drawn on the canvas represented by the canvasObject parameter passed in.

clearLinesOrth(canvasObject) – function to clear all lines drawn on the canvas represented by the canvasObject passed in as a parameter.

4.2.27 Question.js

Question() - Prototype for Question object. The answerCanvas property is an array which stores one or more canvases that the user will draw an answer on. The questionCanvas property is an array which stores one or more canvases which have a predrawn shape. The rotationCanvas property is an array which stores a rotation canvas with rotation instructions.

4.2.28 RotationCanvas.js

Function RotationCanvas(canvasId) – Prototype for RotationsCanvas object. This is needed to display rotation instructions. The canvasId property is the id tag of the rotation canvas. The ctx property is the context of the rotation canvas. The numericRotations property is an array of any numeric rotation instructions and the alphabeticRotations property is an array of any alphabetic rotation instructions.

4.2.29 Startup_Ex1_Admin.js

Script which creates a Question object, to which question and answer canvases are added to help create the corresponding admin html page for Exercise 1.

4.2.30 Startup_Ex2_Admin.js

Script which creates a Question object, to which question and answer canvases are added to help create the corresponding admin html page for Exercise 2.

4.2.31 Startup_Ex3_Admin.js

Script which creates a Question object, to which question, answer and rotation canvases are added to help create the corresponding admin html page for Exercise 3.

4.2.32 Startup_Ex4_Admin.js

Script which creates a Question object, to which question, answer and rotation canvases are added to help create the corresponding admin html page for Exercise 4.

4.2.33 Startup_Ex5_Admin.js

Script which creates a Question object, to which question, answer and rotation canvases are added to help create the corresponding admin html page for Exercise 5.

4.2.34 Startup_Ex6_Admin.js

Script which creates a Question object, to which question and answer canvases are added to help create the corresponding admin html page for Exercise 6.

4.2.35 Startup2_All.js

Basic script used to call the setUpExercises function present in ExerciseSetup.js.

4.2.36 Startup3_DrawIsometricFromOrthographic.js

Script used to setup Exercise 2, button functionality and the ability to load Next and Previous questions.

4.2.37 Startup3_DrawOrthographicFromIsometric.js

Script used to setup Exercise 1, button functionality and the ability to load Next and Previous questions.

4.2.38 Startup3_DrawSingleIsometricReflection.js

Script used to setup Exercise 6, button functionality and the ability to load Next and Previous questions.

4.2.39 Startup3_DrawSingleIsometricRotationArrowsMultipleAxes.js

Script used to setup Exercise 5, button functionality and the ability to load Next and Previous questions.

4.2.40 Startup3_DrawSingleIsometricRotationArrowsSingleAxis.js

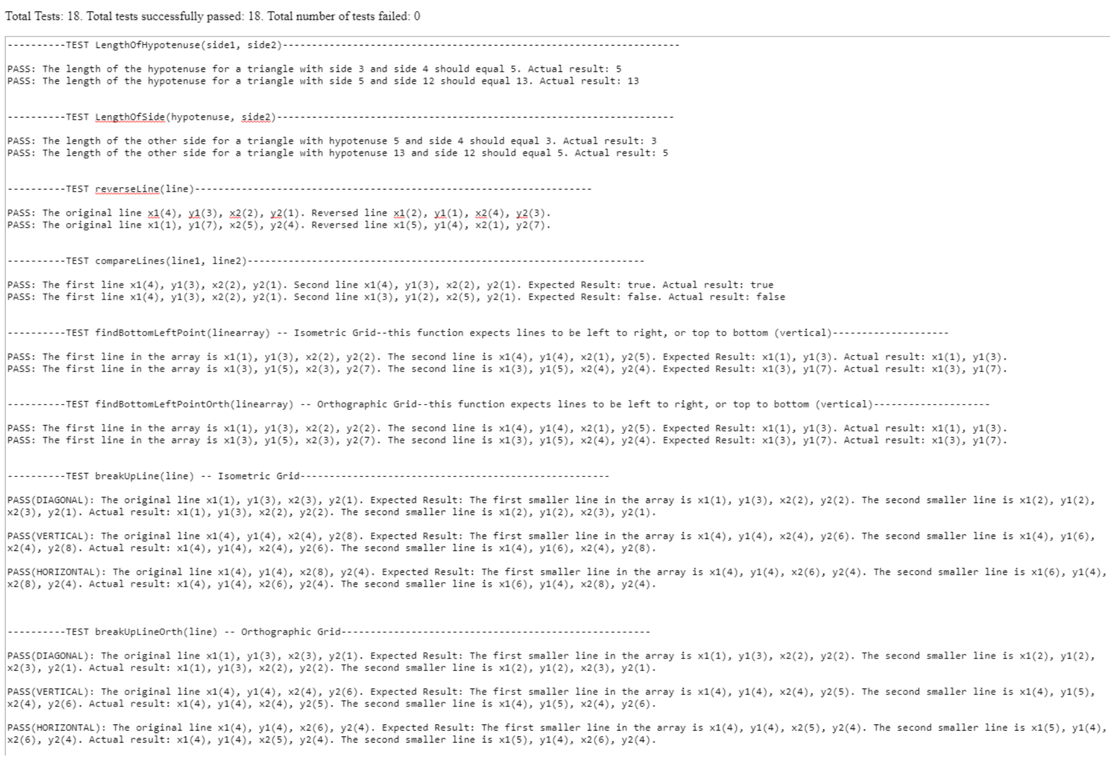
Script used to setup Exercise 4, button functionality and the ability to load Next and Previous questions.

4.2.41 Startup3_DrawSingleIsometricRotationDegreesSingleAxis.js

Script used to setup Exercise 3, button functionality and the ability to load Next and Previous questions.

5.0 Testing

Tests of critical functionality have been created in the Test.html file. Opening this file will run the critical tests as shown below:



```
Total Tests: 18. Total tests successfully passed: 18. Total number of tests failed: 0

----------TEST LengthOfHypotenuse(side1, side2)-------------------------------------------------------------------

PASS: The length of the hypotenuse for a triangle with side 3 and side 4 should equal 5. Actual result: 5
PASS: The length of the hypotenuse for a triangle with side 5 and side 12 should equal 13. Actual result: 13


----------TEST LengthOfSide(hypotenuse, side2)--------------------------------------------------------------------

PASS: The length of the other side for a triangle with hypotenuse 5 and side 4 should equal 3. Actual result: 3
PASS: The length of the other side for a triangle with hypotenuse 13 and side 12 should equal 5. Actual result: 5


----------TEST reverseLine(line)------------------------------------------------------------------

PASS: The original line x1(4), y1(3), x2(2), y2(1). Reversed line x1(2), y1(1), x2(4), y2(3).
PASS: The original line x1(1), y1(7), x2(5), y2(4). Reversed line x1(5), y1(4), x2(1), y2(7).


----------TEST compareLines(line1, line2)-------------------------------------------------------------------

PASS: The first line x1(4), y1(3), x2(2), y2(1). Second line x1(4), y1(3), x2(2), y2(1). Expected Result: true. Actual result: true
PASS: The first line x1(4), y1(3), x2(2), y2(1). Second line x1(3), y1(2), x2(5), y2(1). Expected Result: false. Actual result: false


----------TEST findBottomLeftPoint(linearray) -- Isometric Grid--this function expects lines to be left to right, or top to bottom (vertical)--------------------

PASS: The first line in the array is x1(1), y1(3), x2(2), y2(2). The second line is x1(4), y1(4), x2(1), y2(5). Expected Result: x1(1), y1(3). Actual result: x1(1), y1(3).
PASS: The first line in the array is x1(3), y1(5), x2(3), y2(7). The second line is x1(3), y1(5), x2(4), y2(4). Expected Result: x1(3), y1(7). Actual result: x1(3), y1(7).


----------TEST findBottomLeftPointOrth(linearray) -- Orthographic Grid--this function expects lines to be left to right, or top to bottom (vertical)--------------------

PASS: The first line in the array is x1(1), y1(3), x2(2), y2(2). The second line is x1(4), y1(4), x2(1), y2(5). Expected Result: x1(1), y1(3). Actual result: x1(1), y1(3).
PASS: The first line in the array is x1(3), y1(5), x2(3), y2(7). The second line is x1(3), y1(5), x2(4), y2(4). Expected Result: x1(3), y1(7). Actual result: x1(3), y1(7).


----------TEST breakUpLine(line) -- Isometric Grid-----------------------------------------------------

PASS(DIAGONAL): The original line x1(1), y1(3), x2(3), y2(1). Expected Result: The first smaller line in the array is x1(1), y1(3), x2(2), y2(2). The second smaller line is x1(2), y1(2),
x2(3), y2(1). Actual result: x1(1), y1(3), x2(2), y2(2). The second smaller line is x1(2), y1(2), x2(3), y2(1).

PASS(VERTICAL): The original line x1(4), y1(4), x2(4), y2(8). Expected Result: The first smaller line in the array is x1(4), y1(4), x2(4), y2(6). The second smaller line is x1(4), y1(6),
x2(4), y2(8). Actual result: x1(4), y1(4), x2(4), y2(6). The second smaller line is x1(4), y1(6), x2(4), y2(8).

PASS(HORIZONTAL): The original line x1(4), y1(4), x2(8), y2(4). Expected Result: The first smaller line in the array is x1(4), y1(4), x2(6), y2(4). The second smaller line is x1(6), y1(4),
x2(8), y2(4). Actual result: x1(4), y1(4), x2(6), y2(4). The second smaller line is x1(6), y1(4), x2(8), y2(4).



----------TEST breakUpLineOrth(line) -- Orthographic Grid-----------------------------------------------------

PASS(DIAGONAL): The original line x1(1), y1(3), x2(3), y2(1). Expected Result: The first smaller line in the array is x1(1), y1(3), x2(2), y2(2). The second smaller line is x1(2), y1(2),
x2(3), y2(1). Actual result: x1(1), y1(3), x2(2), y2(2). The second smaller line is x1(2), y1(2), x2(3), y2(1).

PASS(VERTICAL): The original line x1(4), y1(4), x2(4), y2(6). Expected Result: The first smaller line in the array is x1(4), y1(4), x2(4), y2(5). The second smaller line is x1(4), y1(5),
x2(4), y2(6). Actual result: x1(4), y1(4), x2(4), y2(5). The second smaller line is x1(4), y1(5), x2(4), y2(6).

PASS(HORIZONTAL): The original line x1(4), y1(4), x2(6), y2(4). Expected Result: The first smaller line in the array is x1(4), y1(4), x2(5), y2(4). The second smaller line is x1(5), y1(4),
x2(6), y2(4). Actual result: x1(4), y1(4), x2(5), y2(4). The second smaller line is x1(5), y1(4), x2(6), y2(4).
```

These tests cover triangle functions used in touch detection as well as functions which help to compare lines that have been drawn by a user. These must all “PASS” for the app to work correctly. Any test which has a “FAIL” must be investigated.

6.0 PhoneGap Build

PhoneGap Build website: https://build.phonegap.com/

A free account can be set up which allows one app to be hosted on PhoneGap Build, with payment allowing up to 25 separate apps.

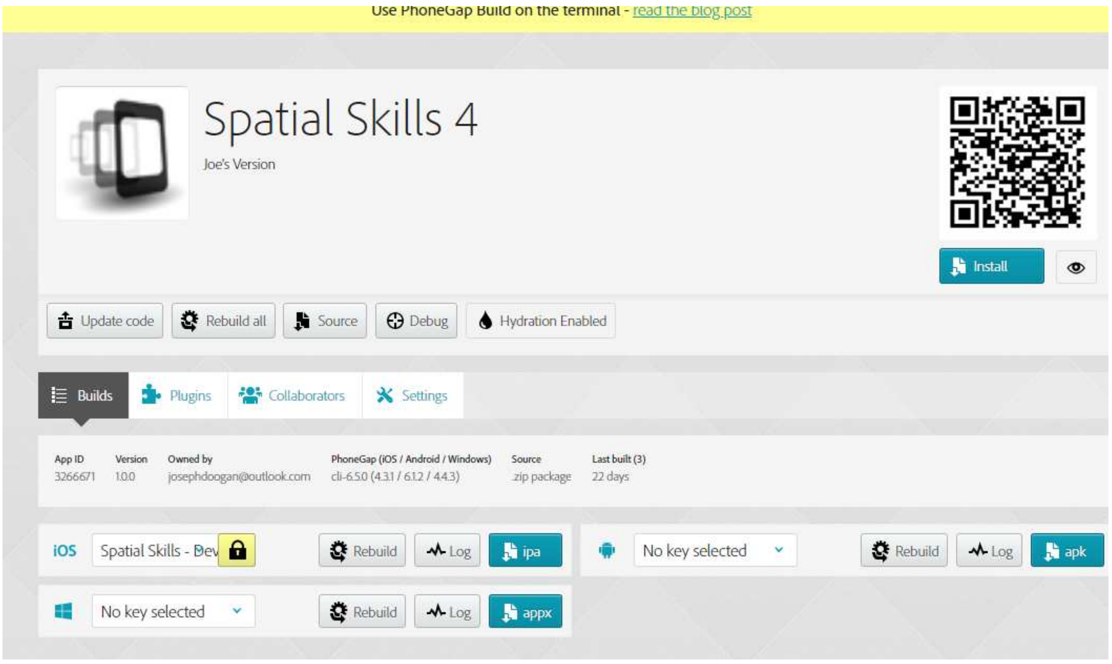
6.1 Upload App to PhoneGap Build

With PhoneGap Build, an app can be uploaded and pushed to multiple devices running Apple iOS, Android or Windows.

The www folder can be compressed as a .zip file and uploaded to the PhoneGap Build website using the Update Code option.

6.2 Installing the app on devices

For an Android device, you can scan the QR code and this will allow the app to be installed.


Use PhoneGap Build on the terminal - read the blog post

Spatial Skills 4

Joe's Version

App ID	Version	Owned by	PhoneGap (iOS / Android / Windows)	Source	Last built (3)
3266671	1.0.0	josephdoogan@outlook.com	cli-6.5.0 (4.3.1 / 6.1.2 / 4.4.3)	.zip package	22 days

For an Apple device, you will have to use the Apple Developer website (https://developer.apple.com/) to create a mobile provisioning certificate and add specific devices to this certificate based on their UDID (Universal Device ID – this can be found by connecting the device to a Mac or PC, opening iTunes and clicking on the Serial Number).

6.3 Updating an app

When you make changes to your code, compress the new version of the www folder and upload this by clicking the ‘Update Code’ button. If ‘Hydration’ is enabled, the update will be pushed automatically to devices.

6.4 Hydration

Enable hydration for the app to allow updates to be pushed to devices automatically when an update is added to PhoneGap Build. Devices will be informed next time the app is launched that “Hydra has detected an update”.

The new version of the app will be downloaded when ‘Update and Restart’ is selected. A message will appear to show an error regarding cordova.js after download has completed. This can be ignored by selecting ‘Go Back’ and then ‘Ignore for Now’.