Self-assessment activities

It is fundamental to keep in mind that it is not by listening to a lecture that a student learns, but that it is by doing, by being active.

When learning online, self-assessment activities allow you to do just that, although they are not the only ones.

One of the main characteristics of online courses is that learners do not have direct contact with the teacher and can only interact with them in a much more limited way than face-to-face.

Accordingly, they can't question him when they have doubts or questions.

It is therefore fundamental, when designing online courses, to plan multiple self-assessment activities so that each learner can check his or her understanding of the concepts presented to him/her.

These activities should be offered after each new video or course page introducing new concepts. It is not necessary for the grades obtained to be memorized. It is probably even preferable that only the learner should be aware of them so that he or she does not fear that they will be used for his or her final assessment.

Here are four examples of self-assessment activities that are easy for a teacher to build if they have the right tools:

- Drag-and-drop image exercises allow learners to check if they can orient themselves in a sketch or graph. They work by simple drag-and-drop;

- Gap fill exercises with or without contextual images require a little more concentration on the part of the learner, but are very fruitful in ensuring that difficult concepts are well understood;

- Categorization exercises complement the previous two activities well: learners organize the elements into categories and thus learn to distinguish their characteristics;

- Finally, single or multiple-choice questions allow them to test their knowledge quite broadly, but they are not very user-friendly activities. They are also widely used for partial or final assessments of learners.

Tools for creating self-assessment activities

There are many environments for developing such activities, but most of them are paid and their deployment can cause various problems.

It therefore seemed justified to us to develop solutions that are easy to implement and free of rights, shared as open educational resources.

At the beginning of November 2023, we shared a set corresponding to the first three types of activity indicated above, using H5P technology, but this solution poses some deployment difficulties.

That's why we've prepared a second set, much lighter set, also developed in html 5.

Drag-and-drop exercises onto image

GUI

The screen for the drag-and-drop onto image exercise is as shown in Figure 1.

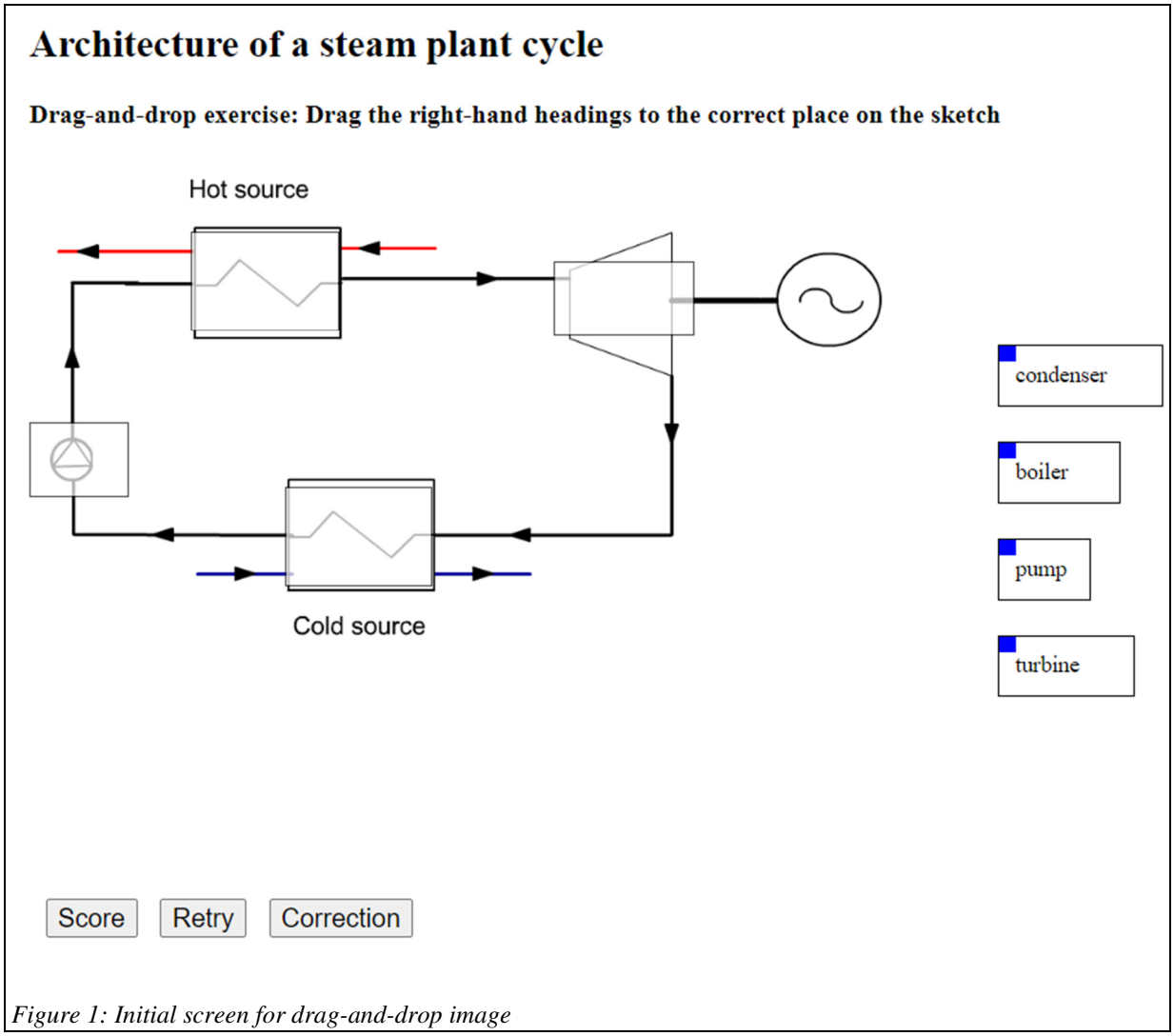

Figure 1: Initial screen for drag-and-drop image

The sketch of the installation appears on the left side of the screen, with some components topped by a semi-transparent rectangle defining a hotspot. On the right side of the screen are moving parts with the names of the different components, with a small blue rectangle at the top left.

These elements can be placed by dragging and dropping onto the hotspots in the background image, with the goal of the exercise being to properly match these areas and moving elements.

Three buttons allow you to get the score at any time, to restart the exercise or to know the correction.

Figure 2 shows the result when an error was made with the pump and turbine positions reversed.

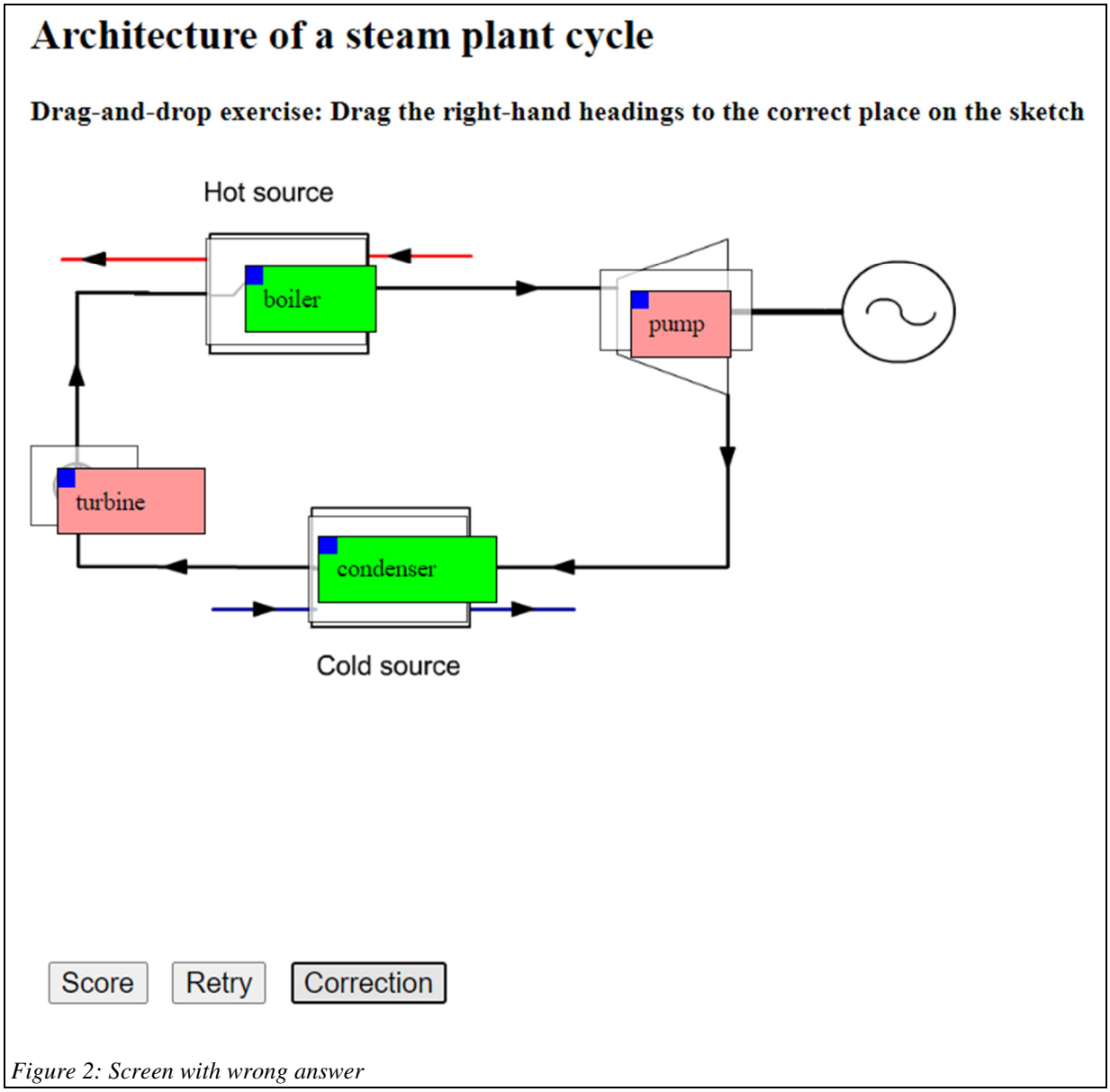

Figure 2: Screen with wrong answer

In order for the evaluation of the pairing of a component and its label to be carried out correctly, more than half of the latter must be located in the sensitive area (the target), and the small blue square must not extend towards the top of the box of the area.

HTML code analysis

The overall structure of the html file is given in Figure 3. To be able to handle drag-and-drop, it needs the jQuery libraries defined in the <head tag scripts>

The <style> tag, folded here, contains the CSS code of the page, which allows you to define its appearance, and in particular to position the targets.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <script src="https://code.jquery.com/jquery-3.6.4.min.js"></script>
  <script src="https://code.jquery.com/ui/1.12.1/jquery-ui.js"></script>
  <link rel="stylesheet" href="https://code.jquery.com/ui/1.12.1/themes/smoothness/jquery-ui.css">

  <style>
</head>
<body>

<div class="container">

<script>

</body>
</html>
```

Figure 3: Structure of the html file

The container division contains the various elements that appear on the screen, and the <script> tag is the JavaScript code that handles the operation of the page.

We will now explain how each of these blocks is constructed, knowing that only the first and second need to be modified to achieve a new drag-and-drop activity.

Tag <style>

This is the one that contains the CSS code for the file. Cascading Style Sheets (CSS) contain the code used to format a web page. If you want to change it, except for the positioning of targets, it is best to start by getting started with this language.

Targets are defined, as will be seen later, by identifiers obtained by adding their number to the word target. Their positions are defined in this block, as shown in Figure 4.

For example, target 3 above the pump in the diagram in Figure 1 is 248 pixels from the top of the page, 0 pixels from the left, 60 pixels wide, and 45 pixels high. For target 1 and 4, we just need to provide their positions, the heights and widths being those defined for the target set a little earlier in the code.

In practice, to find the right values of these parameters, the simplest solution is to open the html file in two ways:

- On the one hand, in an editor like NotePad++, which is well suited for this
- On the other hand, in an Internet browser

```
#target1 {
  top: 130px;
  left: 100px;
}

#target2 {
  top: 148px;
  left: 325px;
  width: 85px;
  height: 45px;
}

#target3 {
  top: 248px;
  left: 0px;
  width: 60px;
  height: 45px;
}

#target4 {
  top: 288px;
  left: 158px;
}
```

Figure 4: Target Positioning

In the browser, you can see the result obtained after changing the parameter values in NotePad++ and saving the file. In just a few minutes, you get the desired result.

Container division

It provides the links to the jpg or png file and specifies the names of the different labels, as shown in Figure 5.

```
113  <div class="container">
114    <div class="title">
115        <h2>Architecture of a steam plant cycle</h2>
116        <h4>Drag-and-drop exercise: Drag the right-hand headings to the correct place on the sketch</h4>
117    </div>
118
119    <img src="img/steam_plant.png" alt="Background Image" class="background-image" id="bgImage">
120
121    <div class="label target4" id="label4" data-correct-target="target4">condenser</div>
122    <div class="label target1" id="label1" data-correct-target="target1">boiler</div>
123    <div class="label target3" id="label3" data-correct-target="target3">pump</div>
124    <div class="label target2" id="label2" data-correct-target="target2">turbine</div>
125
126
127    <div class="target" id="target1"></div>
128    <div class="target" id="target2"></div>
129    <div class="target" id="target3"></div>
130    <div class="target" id="target4"></div>
131
132    <div id="buttons">
133      <button onclick="checkScore()">Score</button>
134      <button onclick="recommencer()">Retry</button>
135      <button onclick="corriger()">Correction</button>
136    </div>
137  </div>
```

Figure 5: Container division

Lines 115 and 116 indicate the title of the page and the instruction given to learners.

Line 119 defines the background image, which is placed in an "img" subfolder.

Lines 121 to 124 are used to set up labels, and lines 127 to 130 are used to set up targets.

Finally, lines 132 to 136 define the buttons and functions that are called when clicked.

The link between labels and targets is defined in lines 121 to 124. Let's consider the first one:

condenser

The id attribute indicates that its identifier is "label4", and the data-correct-target attribute indicates that the corresponding target is "target4". It is worded as "condenser."

The order in which these lines are placed in the code is the order in which they appear on the screen, as you can see from Figure 1.

You'll also notice that no labels are given to targets. If you want to specify one, insert it just before of the target line. This can be particularly useful during the positioning phase of the targets above the image, even if it means removing them once they are well placed.

To modify the html file for a new drag-and-drop exercise, you just need to change the link to the image line 118, then rename the labels, and finally position the targets.

You can easily change the number of targets and labels by adding or removing rows and adding the positioning parameters of the new targets to the <style> tag if there are any.

Tag <script>

It contains the JavaScript code that handles the operation of the page. If you want to change it, you should know that it is best to master this language well and then be very careful, otherwise mistakes can occur.

Editing the file to create a new drag-and-drop exercise

Let's recap the things that need to be changed to create a new drag-and-drop exercise:

1) Start by preparing your background image and update the link to it in the file

2) Determine the number of pairs (label, target) you need and modify the container division accordingly

3) Enter the labels of the different labels

4) Change the title of the activity and possibly the instruction if you wish

5) Perform the positioning of the targets by playing with the <style> tag settings, saving the file after each modification and refreshing the browser page where the activity is displayed

6) If you want, enrich the presentation by playing with CSS and adding elements to the page

Once these changes are made, your new activity should be up and running.

Categorization Exercises

We now present the categorization exercises because their html file is only a variant of the one that allows you to perform drag-and-drop.

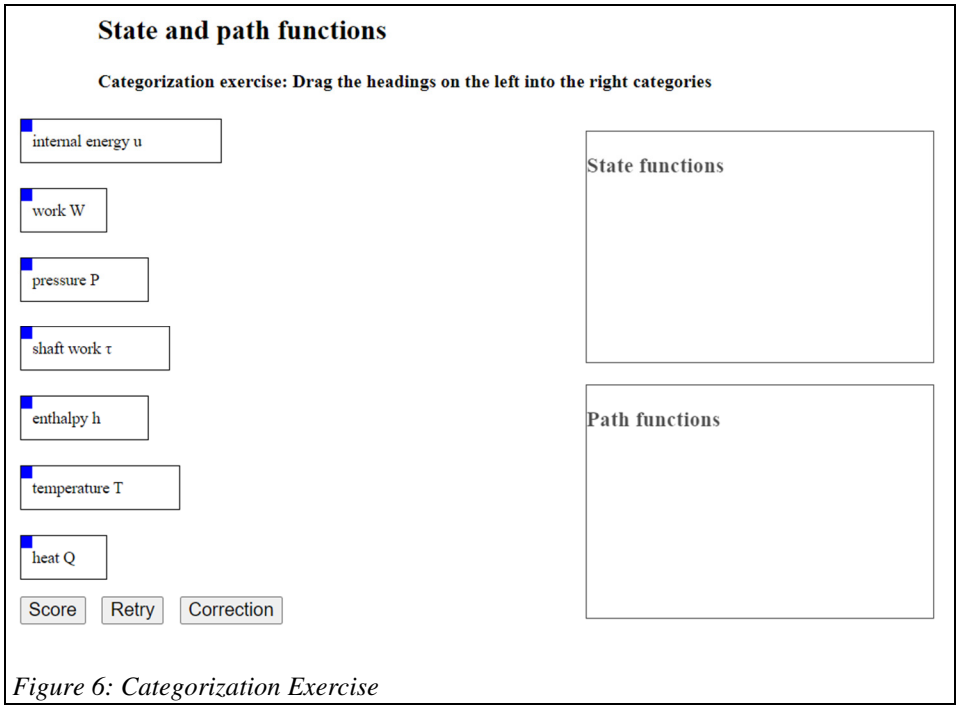

Figure 6: Categorization Exercise

Indeed, the latter is parameterized by indicating that a bijective relationship exists between targets and labels, while in categorization exercises targets can each receive several labels. Also, there is no need for a background image.

GUI

The categories appear on the right side of the screen in Figure 6. On the left are movable elements with the names of the different elements to be categorized, with a small blue rectangle at the top left.

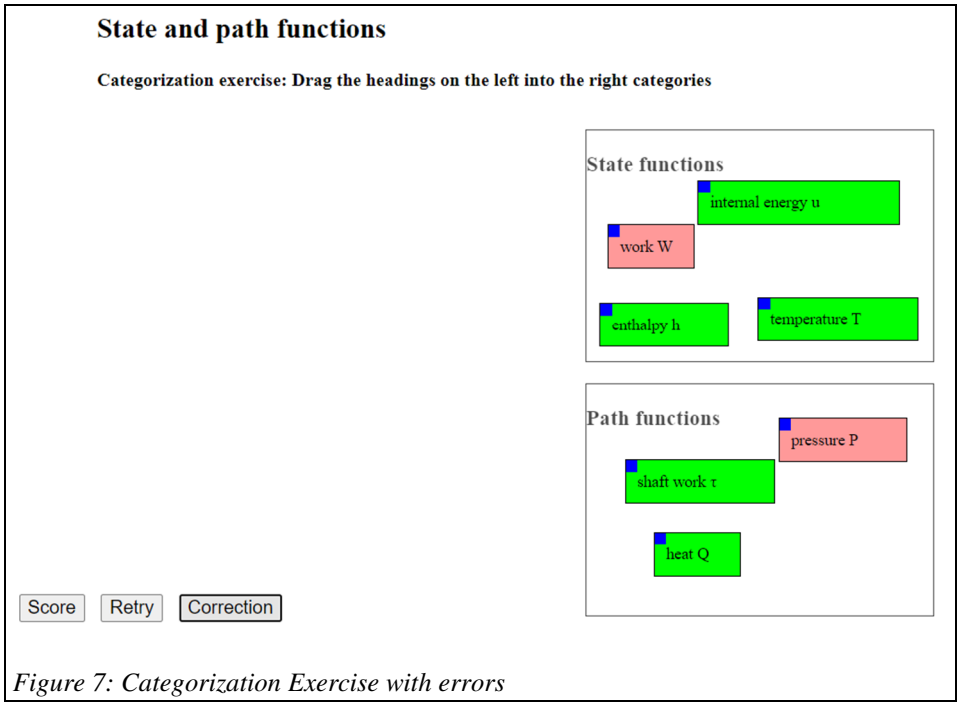

Figure 7: Categorization Exercise with errors

These items can be placed by dragging and dropping them into the large rectangles on the right, the goal of the exercise being to arrange them properly.

Three buttons allow you to get the score at any time, to restart the exercise or to know the correction.

Figure 7 shows the result obtained when an error was made, with the positions of the work W and the pressure P reversed.

In order for the evaluation of the pairing of a component and its label to be carried out correctly, more than half of the latter must be located in the sensitive area (the target), and the small blue square must not extend towards the top of the box of the area.

HTML code analysis

The overall structure of the html file is shown in Figure 8. To be able to handle drag-and-drop, it needs the jQuery libraries defined in the <head> tag scripts.

The <style> tag, folded here, contains the CSS code of the page, which allows you to define its appearance, and in particular to position the targets.

The container division contains the various elements that appear on the screen, and the tag <script> is the JavaScript code that handles the operation of the page.

We will now explain how each of these blocks is constructed, knowing that only the second needs to be modified to achieve a new categorization activity.

Figure 8: Structure of the html file

Tag <style>

This is the one that contains the CSS code for the file. Cascading Style Sheets (CSS) contain the code used to format a web page. If you want to change it, it's best to get started with the language.

Container division

It provides the links to the jpg or png file and specifies the names of the different labels and targets, as shown in Figure 9.

```
140  <div class="container">
141    <div class="title">
142        <h2>State and path functions</h2>
143        <h4>Categorization exercise: Drag the headings on the left into the right categories</h4>
144  </div>
145
146    <div class="label target1" id="label1" data-correct-target="target1">internal energy u</div>
147    <div class="label target2" id="label2" data-correct-target="target2">work W</div>
148    <div class="label target1" id="label3" data-correct-target="target1">pressure P</div>
149    <div class="label target2" id="label4" data-correct-target="target2">shaft work τ</div>
150    <div class="label target1" id="label5" data-correct-target="target1">enthalpy h</div>
151    <div class="label target1" id="label6" data-correct-target="target1">temperature T</div>
152     <div class="label target2" id="label7" data-correct-target="target2">heat Q</div>
153
154    <div class="target" id="target1"><h3>State functions</h3></div>
155    <div class="target" id="target2"><h3>Path functions</h3></div>
156
157    <div id="score">Score: 0</div>
158
159    <div id="buttons">
160      <button onclick="checkScore()">Score</button>
161      <button onclick="recommencer()">Retry</button>
162      <button onclick="corriger()">Correction</button>
163    </div>
164  </div>
```

Figure 9: Container Division

Lines 142 and 143 indicate the title of the page and the instruction given to learners.

Lines 146 to 152 are used to set up labels, and lines 154 and 155 are used to set up targets.

Finally, lines 160 to 162 define the buttons and functions that are called when clicked.

The link between labels and targets is defined in rows 146 to 152. Let's consider the first one:

internal energy u

The id attribute indicates that its identifier is "label1", and the data-correct-target attribute that the corresponding target is "target1". Its wording is "internal energy u".

The order in which these lines are placed in the code is the order in which they appear on the screen, as you can see from Figure 6.

You'll notice that both targets can receive multiple labels, which wasn't done in drag-and-drop exercises.

To modify the html file for a new categorization exercise, simply rename the labels and targets.

You can easily change the number of targets and labels by adding or removing rows and adding the positioning parameters of the new targets to the <style> tag if there are any.

Tag <script>

It contains the JavaScript code that handles the operation of the page. If you want to change it, you should know that it is best to master this language well and then be very careful, otherwise mistakes can occur.

Editing the file to create a new categorization exercise

Let's recap the things that need to be changed to create a new categorization exercise:

1) Determine the number of labels and targets you need and change the container division accordingly

2) Enter the labels of the different labels and targets

3) Change the title of the activity and possibly the instruction if you wish

4) If you want, enrich the presentation by playing with CSS and adding elements to the page

Once these changes are made, your new activity should be up and running.

Gap fill exercises

To specify where the gaps are located in a text, we have used the solution implemented in the corresponding H5P module: the gaps are identified by being inserted between two "*", as in this example: "Closed * systems do not * exchange matter with * the environment *, whereas open * systems do *."

In this sentence, the three gaps are: * systems do not *, * the environment *, and * systems do *.

GUI

The screen of the gap fill exercise is shown in Figure 10. The text is displayed normally, except that drop-down menus with a list of all missing sentence bits are displayed instead of gaps. The order of the contents of these drop-down menus varies randomly from gap to gap.

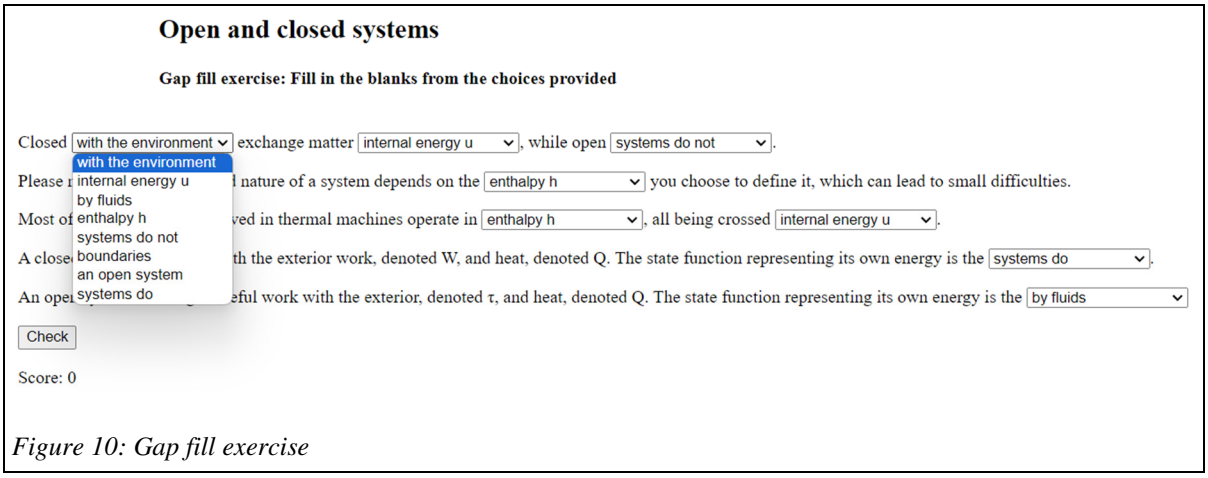

Figure 10: Gap fill exercise

A "Check" button allows you to get the score at any time and know the correction.

Figure 11 shows the result when errors were made, with the correct answers appearing in green and the wrong ones in red.

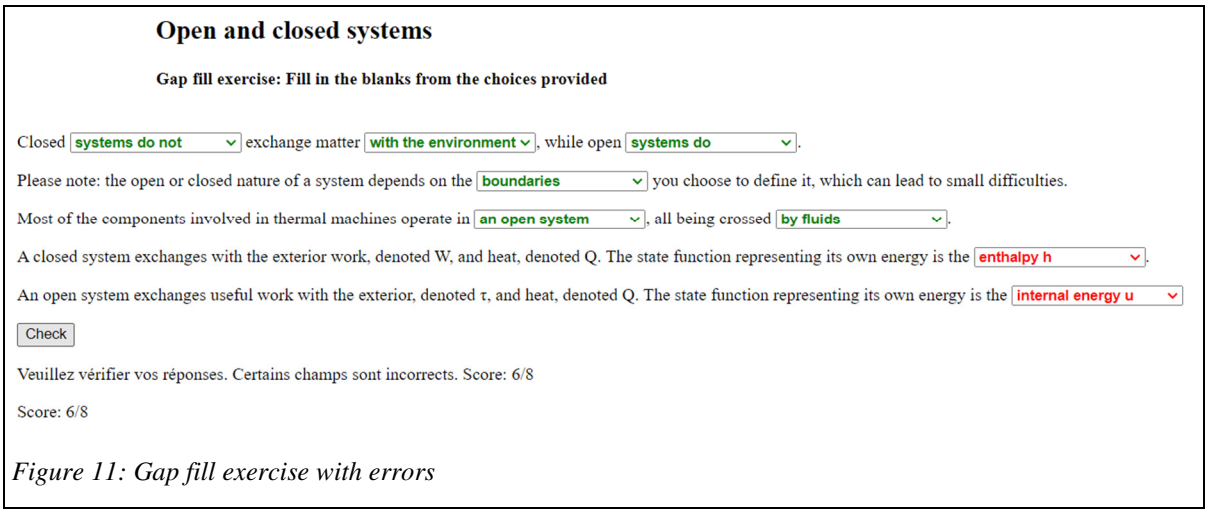

Figure 11: Gap fill exercise with errors

HTML code analysis

The overall structure of the html file is given in Figure 12.

The "title" and "main" divisions contain the various elements that appear on the screen, and the <script> tag is the JavaScript code that handles the operation of the page.

```
1   <!doctype html>
2   <html lang="en">
3    <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Gap Fill Exercise</title>
7     <style>
30   </head>
31   <body>
32    <div class="title">
33     <h2>Open and closed systems</h2>
34     <h4>Gap fill exercise: Fill in the blanks from the choices provided</h4>
35    </div>
36    <div class="main">
37     <form id="gapFillForm">
38      <p>Closed <select class="gapSelect"></select> exchange matter <select class="gapSelect"></select>, while open <select cl
39      <p></p>
40      <p>Please note: the open or closed nature of a system depends on the <select class="gapSelect"></select> you choose to c
41      <p></p>
42      <p>Most of the components involved in thermal machines operate in <select class="gapSelect"></select>, all being crossed
43      <p></p>
44      <p>A closed system exchanges with the exterior work, denoted W, and heat, denoted Q. The state function representing its
45      <p></p>
46      <p>An open system exchanges useful work with the exterior, denoted τ, and heat, denoted Q. The state function representi
47      <p></p><button type="button" onclick="verifier()">Check</button>
48     </form>
49     <p id="result"></p>
50     <p id="score">Score: 0</p>
51    </div>
52    <script>let score = 0;
123  </body>
124 </html>
```

Figure 12: Structure of the html file

Tag <style>

This is the one that contains the CSS code for the file. Cascading Style Sheets (CSS) contain the code used to format a web page. If you want to change it, it's best to get started with the language.

Division title

Lines 33 and 34 indicate the title of the page and the instruction given to the learners.

Division main

Lines 38 to 46 contain the text and its gaps, marked by:

<select class="gapSelect"></select>

Finally, line 41 defines the button and the function that is called when clicked.

Tag <script>

It contains the JavaScript code that handles the operation of the page. If you want to change it, you should know that it is best to master this language well and then be very careful, otherwise mistakes can occur.

Figure 13 shows a partial view of the JavaScript code.

Rows 61 to 68 define the reponsesExactes array, which, as the name suggests, contains all the correct answers, in the order in which the gaps appear.

Rows 101 and 102 show the same items, but in a different form, gathered in a table called reponsesPossibles, which is used to randomly populate the drop-down menus.

Note that both the "main" division and these parts of the JavaScript code can be deduced directly from the statement of the exercise in the form of text with "*", but that this work can be quite laborious if it is done by hand, with significant risks of errors.

That's why we've developed a small Java utility that allows you to easily build these parts of the html file while respecting the required syntax.

```
60    // Set of correct answers for each blank space
61    const reponsesExactes = [["systems do not"],
62  ["with the environment"],
63  ["systems do"],
64  ["boundaries"],
65  ["an open system"],
66  ["by fluids"],
67  ["internal energy u"],
68  "enthalpy h"
69  ];
70    total=reponsesExactes.length;
71
72   // Check answers
73    reponsesExactes.forEach((exactes, index) => {
86
87    // Show result
88    const toutesCorrectes = reponsesUtilisateur.every((reponse, index) => {
91
92    const resultatMessage = toutesCorrectes
93      ? `Congrats! All answers are correct. Score: ${score}/${total}`
94      : `Please check your answers. Some fields are incorrect. Score: ${score}/${total}`;
95
96    document.getElementById("result").textContent = resultatMessage;
97    document.getElementById("score").textContent = `Score: ${score}/${total}`;
98  }
99
100 // Pre-populate drop-down menus with all possible answers
101 const reponsesPossibles = ["systems do not","with the environment","systems do","boundaries","an open system","by fluids","i
102 ];
```

Figure 13: Partial view of the JavaScript code

HTML file builder for Gap fill exercises

Usage

This builder is an application that comes in the form of an executable Java archive (GapFillHtmlBuilder_En.jar file) containing all the libraries it needs.

On Windows and Mac, if it doesn't launch when you double-click on it, you don't have Java installed on your machine. Instructions on how to install it can be found at:
http://java.com/fr/download/index.jsp

On Linux, enter java -jar GapFillHtmlBuilder_En.jar from the command line, or create a launch shell suitable for your environment.

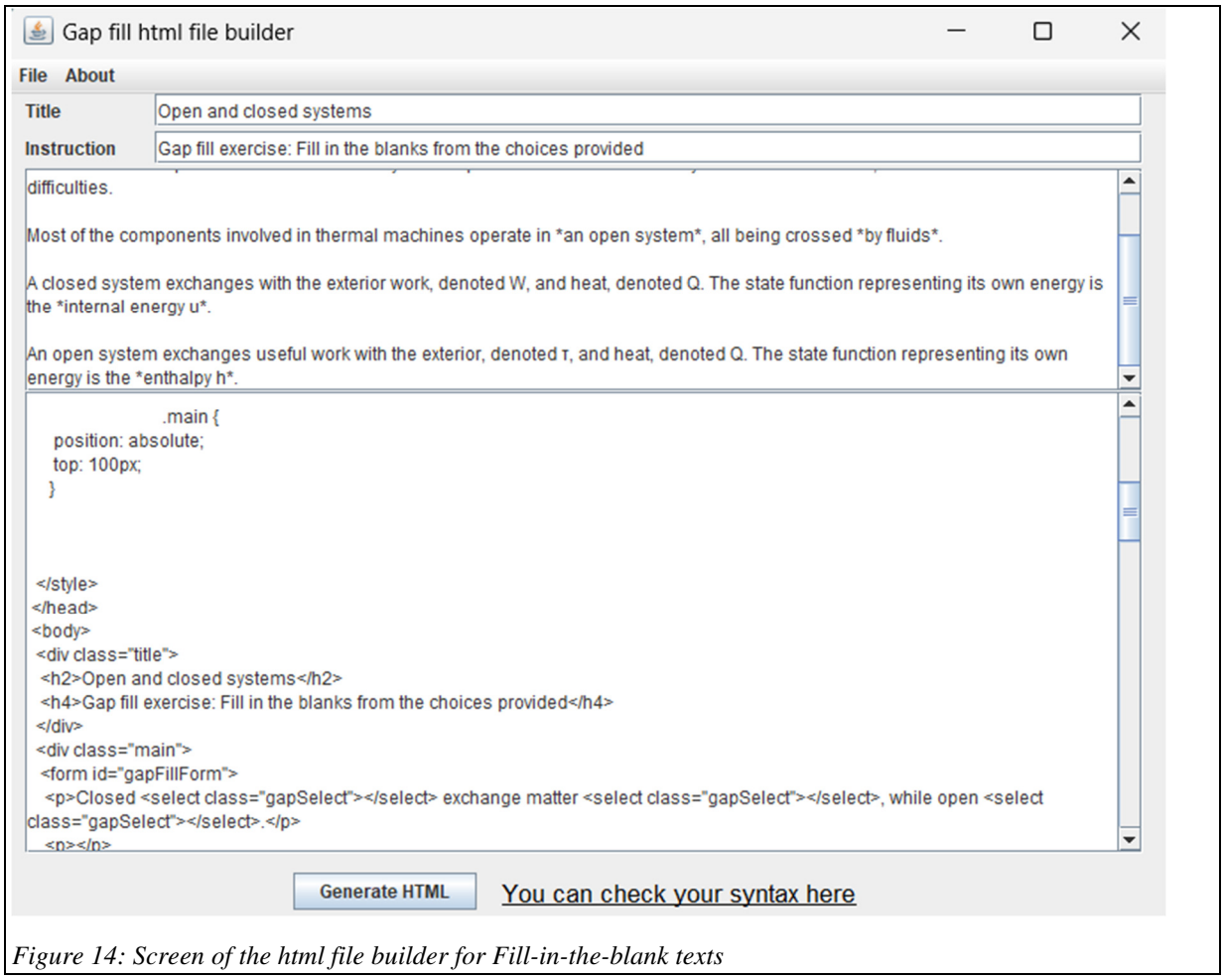

Figure 14: Screen of the html file builder for Fill-in-the-blank texts

This builder creates a new html file by simply editing the text with gaps. It retains the tag <style> and the parts of the other tags that are not dependent on the submitted text. It starts with an existing file that it looks for by default in the "distrib" folder.

GUI

Figure 14 shows the builder screen after opening an existing file.

From top to bottom, it contains two fields for the definition of the title and the instructions, a top text box where you can enter the statement of the exercise as text with "*", and a bottom text box where the html code of the exercise appears.

Finally, at the bottom of the screen there is a button to generate the html code and a hyperlink to check the syntax of the code if desired.

The menu allows you to open a file, save the one you're working with, open an "About" window, and exit the application.

When you open a gap fill exercises html file, its title and instructions are displayed at the top of the screen, and all of the html code is placed in the bottom text box.

All you have to do is enter the wording of the exercise in the form of text with "*" in the upper text box, change the title and the instructions, and you can generate the corresponding html code, which you can save under the name of your choice. You can also edit the html code yourself by hand to add what you want to the page.

Validation of the html code

If you have manually modified the html code, it is best to perform a syntax check before saving the file. This can be done by using the Nu Html Checker page to which the hyperlink at the bottom of the editing window points to.

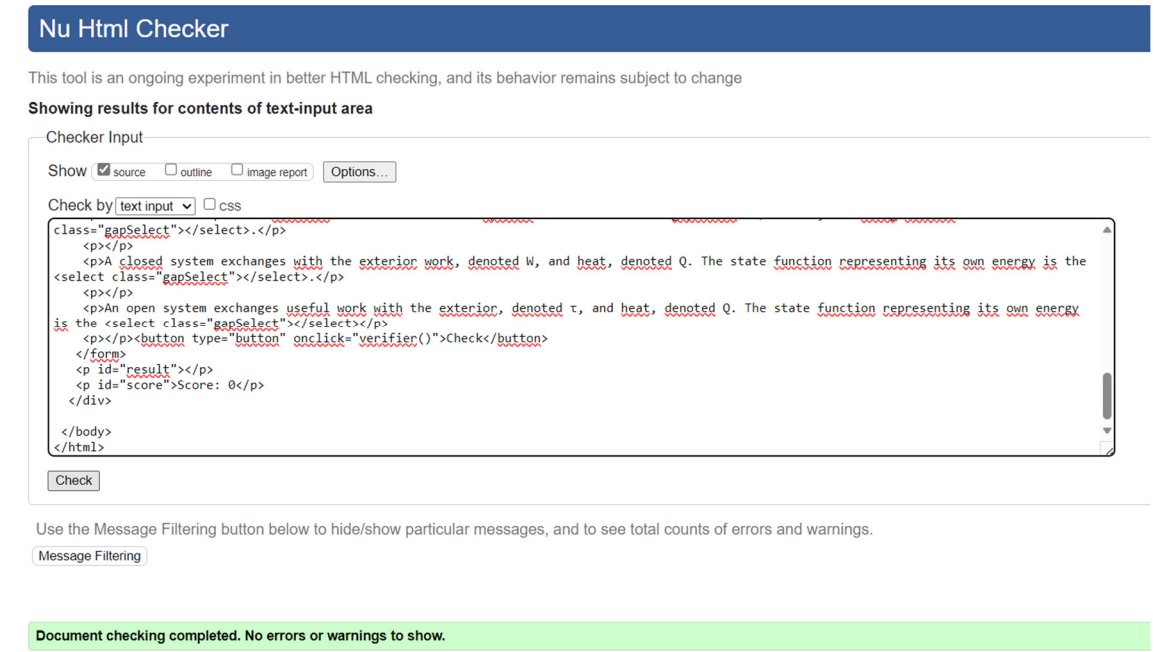

Figure 12: Checking the html code

Caution: Remove the <script> tag from the submitted code, otherwise the procedure will not succeed.