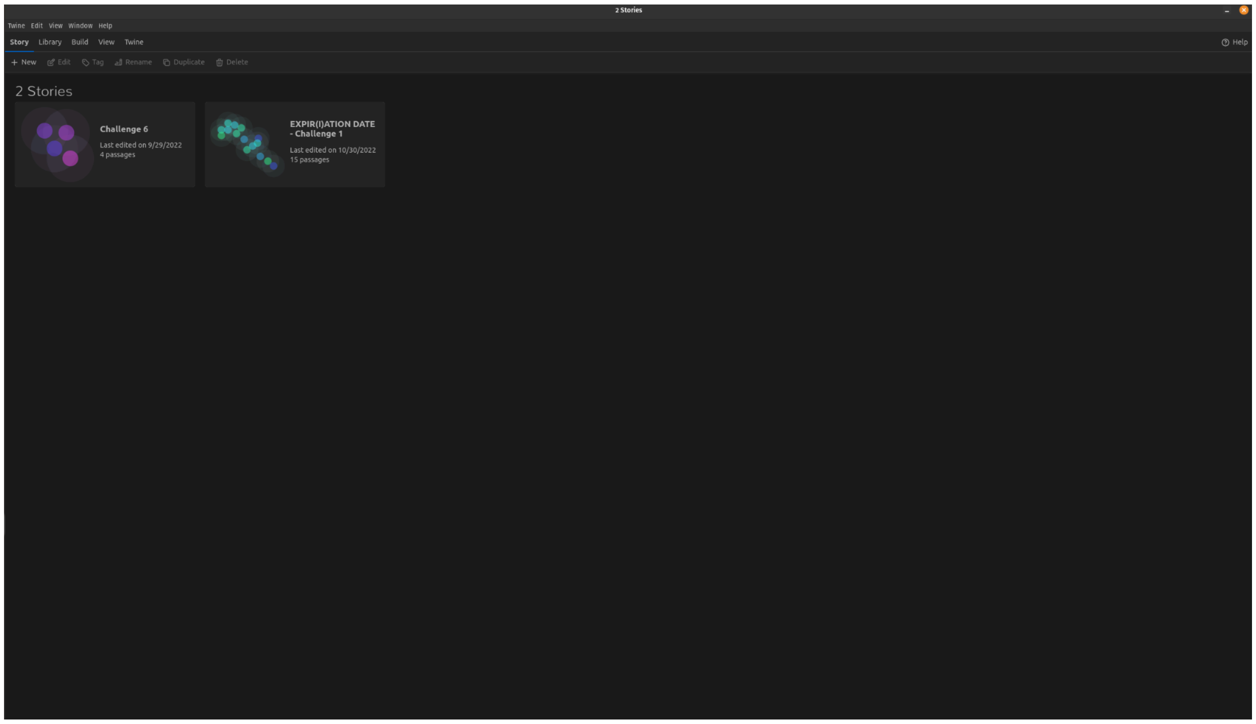
Learnus+ - User guide

OVERVIEW

The following document describes how the user can create a game using Twine - a free tool for telling interactive, nonlinear stories and then upload it on the Learnus+ platform. This user guide is for all types of users, no matter whether they have any previous experience with Twine or not.

STARTING WITH TWINE

The user should navigate to the following website: Twine. It's the official website for Twine. There are a few different story formats for Twine, and in this project we use one of them called SugarCube. For the purpose of the project we would use the Desktop app, so the user should click on the 'Download desktop app' button. After successfully installing it, the user can see the following screen:


2 Stories
Twine Edit View Window Help
Story Library Build View Twine
Help
New Edit Tag Rename Duplicate Delete
2 Stories
Challenge 6
Last edited on 9/29/2022
4 passages
EXPIR(I)ATION DATE - Challenge 1
Last edited on 10/30/2022
15 passages

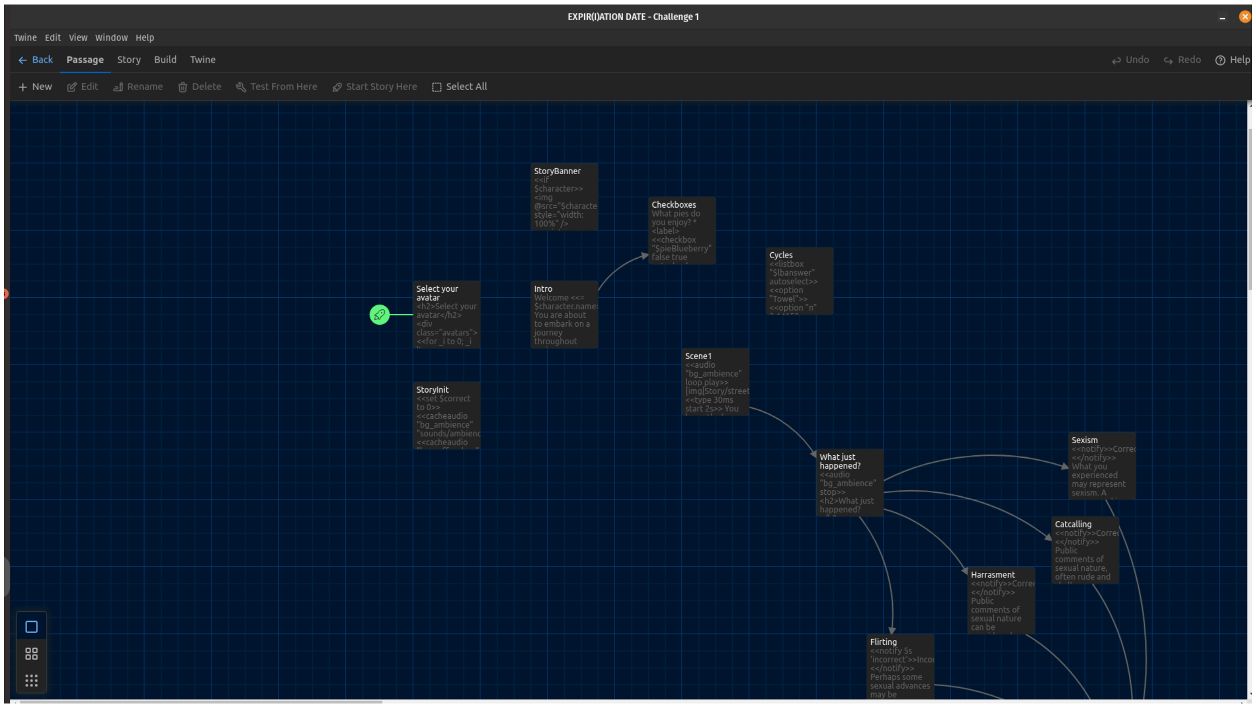
Screenshot 1:

On this screen, the user can create a new story, open and edit an existing one or duplicate and delete a story.

CREATING A NEW GAME

By clicking on the '+ New' link in the menu, the user can create a new game. The user is then navigated to a new window where he/she can see the actual space for creating the game, as in screenshot 2:

This screen is called A Story Map. The Story Map screen shows the visual structure of a story. Each passage in it is represented by a card (the gray rectangle), and links between them are shown as lines with arrows. Along with this user guide, we will develop a simple game with different features so we can show the users what can be accomplished with Twine and how. The game can also be downloaded from the button below this document, on the Learnus+ platform and can be used as a reference for creating your own games.


EXPIR(I)ATION DATE - Challenge 1
Twine Edit View Window Help
Back Passage Story Build Twine
Undo Redo Help
+ New Edit Rename Delete Test From Here Start Story Here Select All
StoryBanner
Checkboxes
Cycles
Select your avatar
Intro
StoryInit
Scene1
What just happened?
Sexism
Catcalling
Harrasment
Flirting

Adding JavaScript

Before you start creating the passages, there are some initial codes that should be added in the story in order to work and look properly. JavaScript is a modern programming language that is used for this purpose. Open the Samples folder and navigate to *Story -> JavaScript*, and then copy the JavaScript code from there and paste it in the same location in your game.

Adding Stylesheets

After adding the JavaScript code snippet, the user should also add a code for CSS. CSS is a simple language we use to style elements. For the purpose of this project, we prepared some initial css code that will be needed in order to create a simple looking game. By navigating from *Story -> Stylesheet*, copy the code and paste it in the same location in your game.

FOLDER GAME STRUCTURE

We propose a simple and clean folder structure for the game. The same folder structure is used for the sample game. It consists of the following folders:

- *Audio* - This is the folder for audio and sounds. All the audio sounds should be placed here
- *Characters* - This is the folder for avatars' images. All the images of the avatars should be placed here
- *Story* - all other images used in the game should be placed here
- *index.html* - this is the main file for the game and should be named exactly like this. By opening it in a browser, the user can see the output of the game.

Please be careful with naming the files and the folders - they are case sensitive. Filenames and folder names should all be in lower-case letters, with no spaces between words; use underscore (_) if necessary. Make sure you also write the file ending (the letters after the ".", for example .mp3, .jpg etc) correctly!

CREATING AND EDITING PASSAGES

After the JavaScript and CSS codes are added, we can create the passages. A passage is represented by a gray rectangle and it's used to describe a scene in the game. A passage can be created by clicking on the *New* button, edited, renamed and deleted by clicking on the correct buttons (Edit, Rename, Delete).

STORYINIT - SETTING VARIABLES

Used for pre-story-start initialization tasks, like variable initialization (happens at the beginning of story initialization). A variable is a value that can change, depending on conditions or on information passed on the program. A variable in Twine is used followed by the $ sign.

StoryInit generates no output.

```
<<set $correct to 0>>
<<cacheaudio "bg_ambience" "audio/ambience.mp3">>
<<cacheaudio "bg_coffeeshop" "audio/coffeeshop.mp3">>

<<character 'robi' 'Characters/Robi.jpeg'>>
<<character 'bob' 'Characters/Bob.jpeg'>>

<<set $characters to [
      {
      name: "Robi",
        image: "Characters/Robi.jpeg"
    },
      {
      name: "Doris",
        image: "Characters/Doris.jpeg"
    },
      {
      name: "Bob",
        image: "Characters/Bob.jpeg"
    },
      {
      name: "Gina",
        image: "Characters/Gina.jpeg"
    },
      {
      name: "Denis",
        image: "Characters/Denis.jpeg"
    },
]>>
<<set $sentences to [
      {
      slug: 'compliment',
        text: 'That was a compliment!'
    },
      {
      slug: 'relax',
        text: 'Relax, nothing happened.'
    },
      {
      slug: 'report',
        text: 'You should report this.'
    },
    {
      slug: 'wear',
        text: 'What did you wear?'
    },
    {
      slug: 'provocative',
        text: 'Aren't you too provocative?'
    },
    {
      slug: 'france',
        text: 'Did you know that in France you can get a fine of 3500 EUR
for catcalling?'
    },
    {
      slug: 'outrageous',
        text: 'Outrageous! I'm so sorry for you.'
    },
    {
        slug: 'exaggerating',
        text: 'You're exaggerating, probably the person wanted to ask you
out.'
    },
    {
      slug: 'imhere',
        text: 'I'm here for you.'
    },
    {
      slug: 'jeans',
        text: 'I told you you shouldn't wear those jeans.'
    },
    {
      slug: 'whatyouneed',
        text: 'Tell me what you need.'
    },
    {
      slug: 'same',
        text: 'Same thing happened to me yesterday!'
    },
    {
      slug: 'chocolate',
        text: 'Have some chocolate.'
    }
]>>

<<set $correct to 0>>
<<newmeter 'correct' 0.5>><</newmeter>>
```

In this code snippet, all variables used through the game are initialized. The user should copy this, paste it in their StoryInit passage and make the following changes:

- Change the values of the names and the images for the characters
- Change the values of the slugs and texts with your versions for the set of sentences
- Change or add your own characters, audio files, etc.

STORYBANNER

The Story banner is a special kind of passage. It is used to populate the story's banner area in the UI bar (sidebar menu). In our case, we use the Story Banner to show the avatar (if it's chosen) and for printing the user's correct points. Use the following code snippet to create a Story Banner, the same as in the samples:

```
<<if $character>>
      <img @src="$character.image" style="width: 100%" />
    <<print $character.name>>
<</if>>
Points
<<print $correct>>
<<set _correctPercentage to ($correct + 5) / 10>>
<<showmeter 'correct' _correctPercentage>>
```

Note that _correctPercentage needs to be a decimal number between 0 and 1 where 0 will display an empty bar and 1 will make a full bar. In this case, 0 points will display a half-full bar, -5 points will display an empty bar, and +5 points will be a full bar. Modify the formula according to your requirements.

SELECTING AN AVATAR

Copy this code snippet and paste it in the passage where the user should make the selection of the avatar who he/she will be using in the game. If the user has followed all the explanations from above and used the same variable names in his/hers game, he/she doesn't have to change anything in this code in order to create a character selection, just copy and paste the code.

```
<div class="avatars">
  <<for _i to 0; _i lt $characters.length; _i++>>
    <<capture _i>>
      <a data-passage="Intro" data-setter="$character to $characters[_i]">
          <img @src="$characters[_i].image">
          <<print $characters[_i].name>>
      </a>
    <</capture>>
  <</for>>
</div>
```

LINKS

Passage links are represented with a solid line ending in an arrow in the story map. A passage can also link to itself, but its usage is not very common.

Adding links or connecting the passages between themselves can be done using the following command:

```
[[A link->Passage 1]]
```

Writing this command, makes a link to a passage, named Passage 1, and the text that is displayed onscreen is ‘A link’. What the user has to do here is just copy the code from above and change the texts with suitable ones for his/hers purposes.

If your link also needs to do something else, like changing a variable, you can easily achieve this by using the following syntax:

```
[[A link->Passage 1][$correct to $correct - 1]]
```

This example reduces the value of $correct by 1

ADDING AUDIO

Adding audio, as well as other multimedia files can be done by placing these files in the same folder as the game. The naming convention we propose and it’s used in the Samples folder we provided is:

- Use *audio* folder for the sounds you would like to have in the game, as explained in the Folder structure section
- Preload required audio in the StoryInit block
- Use the following command to automatically start an audio:

```
<<audio "bg_coffeeshop" loop play>>
```

Also, the following buttons can be used for more audio controls:

```
<<button "Play">><<audio "bg_ambience" play>><</button>>
<<button "Pause">><<audio "bg_ambience" pause>><</button>>
```

```
<<button "Stop">><<audio "bg_ambience" stop>><</button>>
```

The first button plays the audio, the second one pauses it and the last one is used for stopping the audio. Just copy the code snippet and paste it in the passage where you would like to use it.

ADDING IMAGES

Adding images, as well as other multimedia files can be done by placing these files in the Images folder, part of the main game folder (explained in the Section Project Structure above). Then, use the following code snippet to add an image in a passage:

```
[img[Story/coffeeshop.jpg]]
```

LINK APPENDS

Creates a single-use link that deactivates itself and appends its contents to its link text when clicked. Here is an example of using the link append macro:

```
<<linkappend "Harrasment">>
Definition: any unwanted behaviour, physical or verbal (or even suggested),
that makes a person feel uncomfortable, humiliated, or mentally distressed.
[[This is it!|Harrasment][$correct to $correct + 1]]
<</linkappend>>
```

All you have to do here is change the values of *'Harrasment'* and the other text inside the <<linkappend>> brackets.

At first this displays a link that says "Harrassment". When clicked, the link disappears, and the entire content (Definition: any unwanted….), including the link at the end that enables you to lock down your answer and get points.

PRINTING VARIABLES

In order to print a variable (like the character's name), just copy the following code snippet and paste it:

```
<<= $character.name>>
```

This prints out the number of user's points:

```
<<= $correct >>
```

SELF-TYPING TEXT

Another Twine feature implemented in the sample game is the self-typing text. This can be done using the following code snippet:

```
<<type 30ms start 2s>>
Insert text here
<</type>>
```

The text should be added between the opening and the closing <type> blocks.

DIALOG BOX

Dialog boxes can be used by simply copying and pasting the following code snippet in your code. You should insert the text inside the brackets.

```
<<robi>>Hey there!<</robi>>
```

LISTBOXES

Cycles demonstrates how to create a 'cycling' effect of different choices through clicking on them. The user has to choose only one option from the given ones. This is an example of a

```
<<listbox "$lbanswer" autoselect>>
      <<option "Towel">>
      <<option "π" 3.14159>>
      <<option 42>>
      <<option 5>>
      <<option "∞" Infinity>>
<</listbox>>
```

In order to use this component, the user should copy the code snippet and paste it. Then only the value in the *option* brackets should be changed (e.g 42, 5 etc).

CHECKBOXES

Twine also offers using checkboxes and the user can select one or more choices. Here is an example of using checkboxes:

```
What pies do you enjoy?
*<label><<checkbox "$pieBlueberry" false true autocheck>>Blueberry?</label>
*<label><<checkbox "$pieCherry" false true autocheck>> Cherry?</label>
```

```
*<label><<checkbox "$pieCoconutCream" false true autocheck>>Coconut cream?</label>
```

To add a checkboxes option in the game, the user should take the following steps:

1. Copy the code from above
2. Change the value of the variable inside the checkbox block, for example, instead of $pieBlueberry - you should add your option. Do not forget the $ before the variable name!
3. Change the value of the option, for example, instead of Blueberry, type something you want to have as a checkbox option.

So, all in all, you should change 2 values - the variable inside the checkbox code block and the actual value inside the label code block.

NOTIFY MACRO

The <<notify>> macro displays everything between its tags in a small message box on the top right of the screen that slides in and slides out after a brief delay. Example of how it can be used for a correct statement:

```
<<notify>>Correct!<</notify>>
```

And for an incorrect one, please use the following one:

```
<<notify 5s 'incorrect'>>Incorrect!<</notify>>
```

CAPTURE MACRO

The capture macro captures story $variables and temporary _variables, creating localized versions of their values within the macro body. It's used in combination with other macros, as explained in the 'Selecting an avatar' section.

FADE-IN MACRO

This is a simple macro set that causes the text between its tags to fade in or out over a period of time specified by the user, with an optional delay. This is the code snippet you need to create a fade-in macro in your game:

```
<<fadein 3s>>
Actions go here
```

```
<</fadein>>
```

SHOWMETER MACRO

This is a simple, custom macro, developed for creating and working with dynamic, animated meters. It's useful for creating things like health bars, progress meters and visual timers. In our case, it's used for showing the percentage of correct answers, using the following command:

```
<<showmeter 'correct' _correctPercentage>>
```

BUILDING THE GAME

After completing the game, the user should build a version of it. He/she can do it by navigating to Build-> Publish to file from the menu, and build it in the location where all the other assets are located. **The file needs to be named index.html to properly work on the platform. All the files should be selected and put into a zip file. index.html must be at the root of this zip file.**

UPLOADING A GAME

After completing the game, the user can upload it on the Learnus+ platform. He/she can do it by taking the following steps:

1. Navigate to the 'Create your own game' menu item
2. On this page the user can see a form. He/she should fill in the following fields:
    - Title of the game
    - Short excerpt of the game
    - Description of the game
    - Attach featured image
    - Upload the .zip folder of the game

After clicking on the 'Upload game' button, the game is sent for review to our administrators. If the administrators accept the game, it will be then listed along with the other games on the Homepage.

BADGES

The creator of the game should send data that the game is finished. This can be done using 2 very simple steps:

1. At the end of the *Story JavaScript* file, the following code should be added:

```
window.sendData = () => {
  window.top.postMessage("gamecomplete", "*");
  return "";
}
```

2. At the end of the last passage, the following line should be added:

```
<<= sendData() >>
```

This example is added in the Samples folder.