Virtual Tutoring Previewer Documentation

Ricardo Pereira

Ricardo Silva

Ricardo Rodrigues

Introduction

This document offers documentation on the Virtual Tutoring Previewer (VTP). We'll start by exploring **Yarn,** its functionalities and specific usages in VTP, followed by an in-depth exploration of **Previewer**, which includes its menus and components and Basic and Advanced commands that offer more control over the tutors, yet they are optional and not essential to create a good scenario.

Along with this documentation, we recommend the reading of Yarn and YarnSpinner documentation, since we use these technologies you can find interesting things we currently don't explain in this documentation.

Yarn - https://github.com/InfiniteAmmoInc/Yarn

YarnSpinner - https://github.com/thesecretlab/YarnSpinner

Dialogue

We create dialogue by using a program named **Yarn**, this program allows you to easily create dialogue that can interpreted by certain games. It presents a simple to use interface, using nodes and connections to allow the creation and reading of dialogues. We'll discuss some of its components in this chapter along with specific behaviors in VTP.

Node

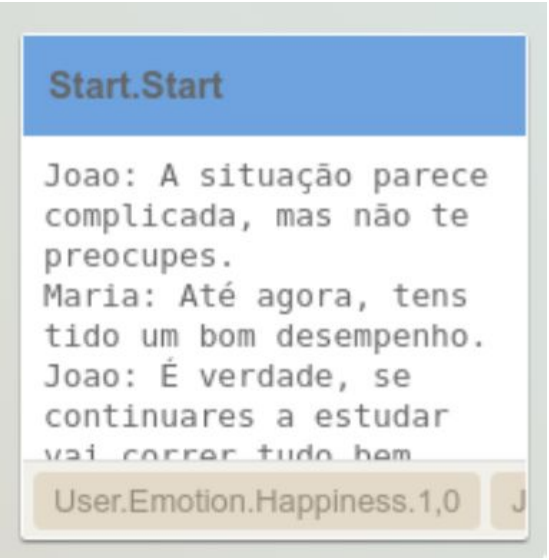


Fig. 1. Example of a dialogue node.

To create a node right-click in an open space inside Yarn, or press **+ Node** on the top right. You can edit it by double-clicking it. In it you can edit 3 main sections: title, tags and content. These are simple text strings that are interpreted in different ways.

Title

Title is an unique name that can be used to reference a node. Allowing a program to select where to start the dialogue or connect different nodes.

Tags

Tags are used to describe the node, for example define what subject the node discusses, because of this it highly depends on context. Tags are small text strings split by spaces. In the following example we show two tags, "Subject.Welcome" and "Emotion.Happy":

```
Subject.Welcome Emotion.Happy
```

In VTP tags are currently used to define the mood the virtual tutors are feeling. The accepted tags should be written in the following format:

```
[tutor name].Emotion.[mood name].[intensity]
```

[tutor name]: can be either "Maria" or "Joao", names of the virtual tutors.

[emotion name]: can be one of the accepted mood emotions in the following list:

- Neutral;
- Happiness;
- Sadness;
- Fear;
- Surprise.

[intensity]: number between 0 and 1 that determines the intensity of the emotion mood being felt, from low to high respectively. Fractional numbers should be written using commas, as such 0,7 (*and not 0.7)*.

For example a tag such as Maria.Emotion.Happiness.0,9 will make the tutor Maria be in a mood of happiness with a high intensity, while a tag such as Joao.Emotion.Fear.0,2 will make the tutor Joao be slightly frightened.

Content

Content is where one defines the dialogue and the options shown to the user. We'll use the figure below as an example.

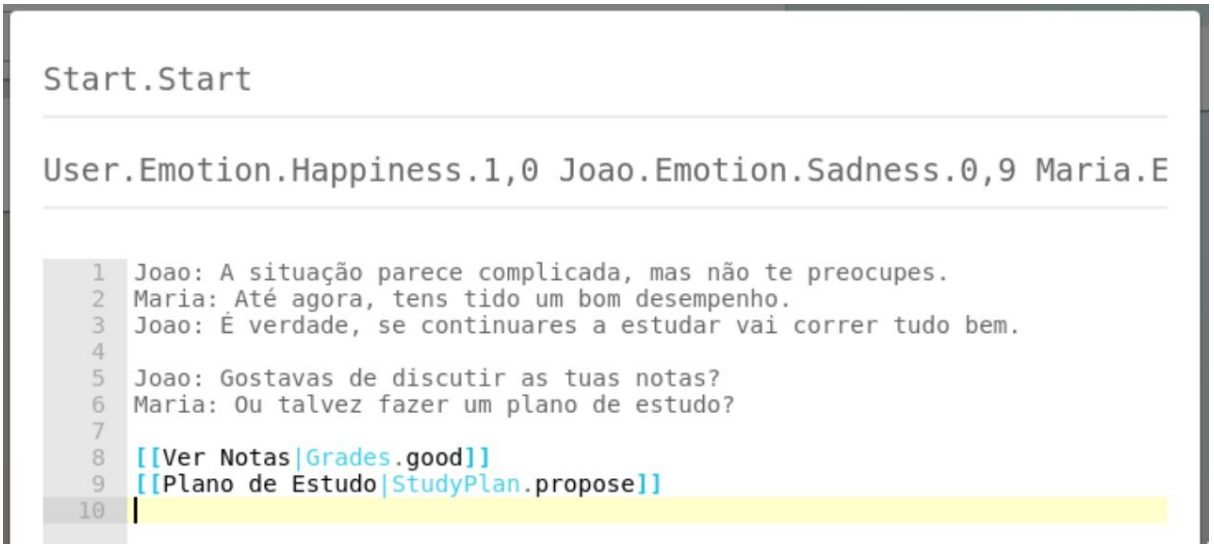


Fig. 2. The editor window for a dialogue node.

Dialogue Lines

Each line represents a line of dialogue (line 1 through 6). In VTP each dialogue line should inform what tutor is speaking and what they are saying, to do that simple state the name of the tutor, Maria or Joao, and the line he is saying, split by a colon:

Maria: Nice to meet you!

Spaces before the message (after the colon) and after the message are ignored.

Options

Options are choices presented to the user, based on the context of the interaction with the tutors. Think of them as possible answers to questions.

For an user to select an option it should be defined in the content. To present an option write a line with the a message, a vertical bar and a node that the option leads to, surrounded with double square brackets, like so:

```
[[Let's go for a walk|GoForAWalkNode]]
```

This option will be presented to the user with the sentence "Let's go for a walk" and if selected the dialogue will go to a node named "GoForAWalkNode".

To present several options to the user, one should write each option in separate, sequential lines (figure 2, lines 8 and 9).

There is a special option in VTP where, if the message is named BLANK, `[[BLANK|OtherNode]]`, it tell the program that if the user doesn't reply within a certain time, one minute, it should go to node "OtherNode". It is not currently possible to modify the time the program waits for an answer.

Commands

For more advanced users, there are also commands that can be defined using double less-than and greater-than signs, like so `<<commandName parameters>>`. In the following chapter we'll explore custom commands in VTP, additionally we recommend visiting the documentation of YarnSpinner (https://github.com/thesecretlab/YarnSpinner/blob/master/Documentation/YarnSpinner-Dialogue/Complex-Dialogue-Example.md).

Previewer

The presenter is an Unity application that allows a user to interact with the virtual tutors. In this chapter we'll describe the components and custom commands interpreted by this application.

File Chooser

First and foremost the application needs to know what file to read the dialogue from. When booting up, the application will present you with a file chooser to allow you to select a .yarn.txt file. An example file is already accompanying the application, named preview.yarn.txt. Feel free to edit and create other yarn files.

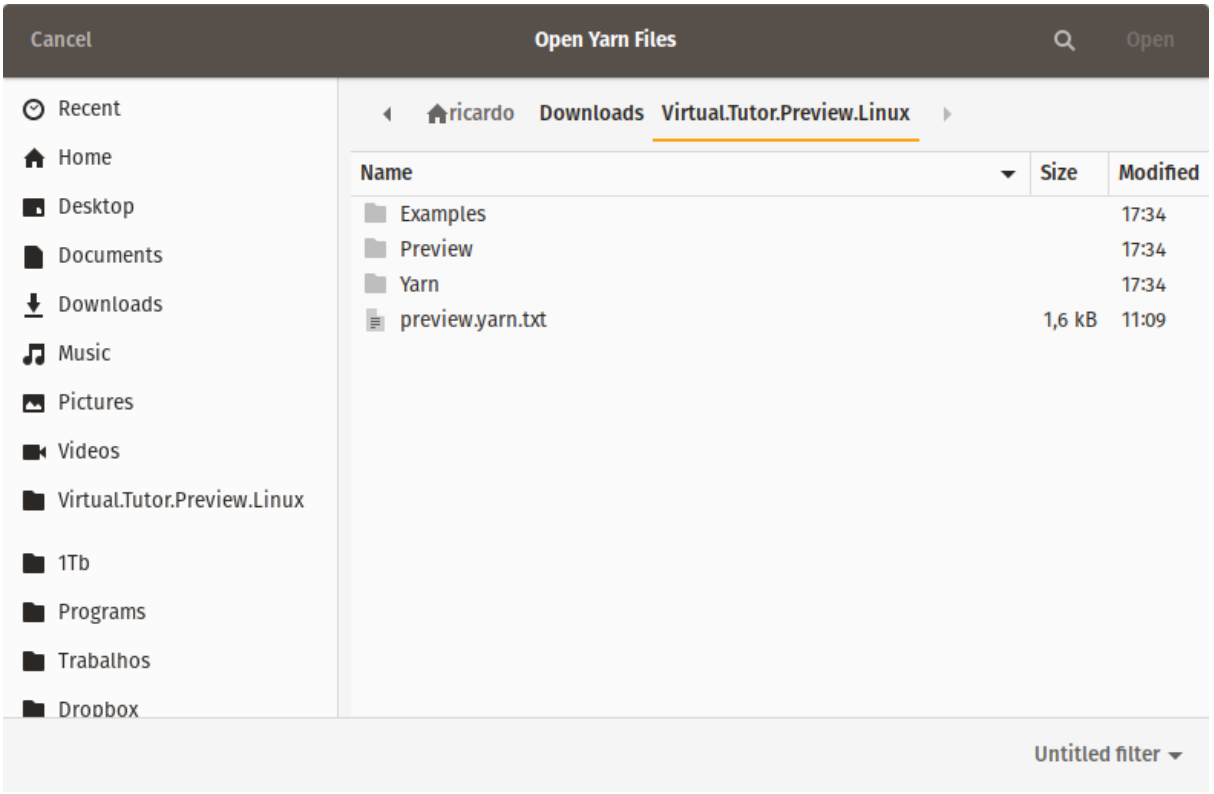


Fig. 3. File chooser will only show yarn files.

Start Menu

After starting the application, the start menu is presented along with the two tutors.

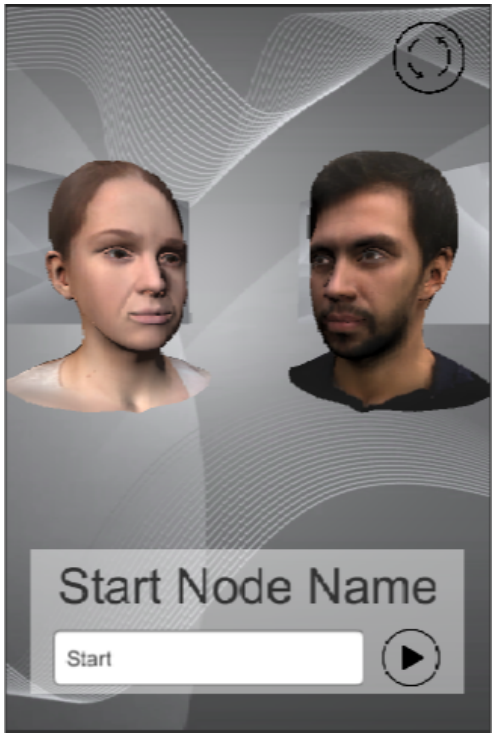


Fig. 4. The Start Menu of the VT Previewer.

The start menu is composed of two items, **Start Node Name** field and the **Restart** button. The former allows you to select what node you want the dialogue to start from, by naming a node in the field and, either pressing in the Play arrow button on the bottom right or entering the Return key, it starts the dialogue using the tutors.

The latter is presented at the top right corner with a two rotating arrows icon, when pressed it resets the experience and shows the **file chooser** again. It also remembers the name of the last node selected, simply press the Play button and test the modifications.

General Use Commands

VTP supports some general use commands.

The first is the exit command:

```
<<exit>>
```

This commands, as the name suggests, exits the application. You can consult the *ExitExample.yarn.txt* file, located in the folder “General Use Commands”, for an example dialog that showcases this command.

Wait command is also available:

<<wait [seconds]>>

It tells the application to wait for a given amount of time before running the next step. You can consult the *WaitExample.yarn.txt* file, located in the folder “General Use Commands”, for an example dialog that showcases this command.

To modify how much time the tutors wait for user reply. For that there is the following command:

<<SetOptionsDuration [seconds]>>

It modifies the value used by the application, when waiting for the user reply. This means, that the options will disappear after the given time. By default the value is -1, meaning there won’t be a timeout, where the tutors will always wait for the user response.

You can consult the *SetOptionsDurationExample.yarn.txt* file, located in the folder “General Use Commands”, for an example dialog that showcases this command.

Command	**Seconds**
SetOptionsDuration	seconds >= 2 seconds = -1

Sometimes it’s important to be able to modify how the tutors are feeling during a dialogue. Therefore a command to manipulate that is available, the Feel command:

<<Feel [Tutor] [Emotion] [Intensity] [Reason]>>

At any given moment, and depending on the situation, a tutor can be feeling one of a variety of different emotions. If the dialogue calls for an emotional change, it can be requested by issuing a "Feel" command to the tutor in question. This command will also force the background to change, in order to match the new desired emotion.

The accepted values for each property of the command are listed in the following table.

Command	**Tutor**	**Emotion**	**Intensity**	**Reason**
Feel	Maria Joao	Neutral Happiness Sadness Fear Surprise	0.0 ≤ … ≤ 1.0	None Challenge Effort Engagement Enjoyment Importance Performance

Note: One can achieve a "Neutral" behavior in two ways:

- By giving a value of 0.0 to the field "Intensity". In this case, it is irrelevant which "Emotion" option is used.
- By giving the option "Neutral", for the emotion. In this case, it is irrelevant which value is used for "Intensity".

Here are some usability examples of the "Feel" command.

Example	Description
<<Feel Maria Happiness 0.8 Challenge>>	Forces the tutor "Maria" to feel *very* happy, and changes the background to the challenge icon.
<<Feel Joao Sadness 0.5 None>>	Forces the tutor "Joao" to feel *moderately* sad, and changes the background to an abstract icon.

You can consult the *FeelExample.yarn.txt* file, located in the folder “General Use Commands”, for an example dialog that showcases this command.

Character Animation Commands

The character animation module controls the actions of the two virtual tutors.

A variety of commands can be sent, via the dialog creation system, to issue various requests to each of them.

The generic format of these requests is as follows.

Command	Type of request that will be sent to the animation module.
Target	Virtual tutor one wishes to send the command to.
Outcome	Desired outcome of the command.
Value	Optional decimal value, required only for certain types of commands.

The following sections will go over each of the accepted requests in greater detail, and the respective commands to trigger them.

Expressions

```
<<Express [Tutor] [Emotion] [Intensity]>>
```

While the *Feel* command is useful to change the mood of a tutor over a long period of time, sometimes it may also be useful to have them express brief “outbursts” of emotion, that only last for a few seconds.

Should the dialogue call for an expressive act of that nature, it can be requested by issuing an "Express" command to the tutor in question.

The accepted values for each property of the command are listed on the following table.

Command	Tutor	Emotion	Intensity
Express	Maria Joao	Neutral Happiness Sadness Fear Surprise Disgust Anger	0.0 ≤ … ≤ 1.0

Here are some usability examples of the "Express" command.

Example	Description
<<Express Maria Happiness 0.8>>	Tutor "Maria" will briefly display a big smile.
<<Express Joao Anger 0.5>>	Tutor "Joao" will briefly display an angry face.

You can consult the *ExpressExample.yarn.txt* file, located in the folder “Character Animation Commands”, for an example dialog that showcases this command.

Movements

Besides emotional displays, the virtual tutors can also perform a variety of movements, either by request, or when relevant events occur. If the dialogue calls for adicional movements, such as a head nod to agree with a particularly strong statement, or a gaze movement directed at the user, they can be requested by issuing the appropriate command to the tutor in question.

Note: The movement system is fully reactive, and will take care of any gazing, nodding and talking motions automatically as the dialog progresses. These manual commands should only be used as a way to request *additional* movements from the Tutors.

We will address the permitted movement commands, and present some usability examples, in the following subsections.

Nodding

<<Nod [Tutor] [State]>>

This command can be used to request a Tutor to initiate\terminate a nod action.

Note: Once requested, nod actions will continue to be performed until the appropriate, opposite command is issued. This means that, if a nod action for Maria is initiated via the <<Nod Maria Start>> command, then Maria will continue noding until the command <<Nod Maria Stop>> is issued.

Command	Tutor	State
Nod	Maria Joao	Start Stop

Here are some usability examples of these commands.

Example	Description
<<Nod Maria Start>>	Tutor "Maria" will begin nodding.
<<Nod Joao Stop>>	Tutor "Joao" will cease nodding.

You can consult the *NodExample.yarn.txt* file, located in the folder “Character Animation Commands”, for an example dialog that showcases this command.

Gazing

<<Gazeat [Tutor] [Target]>>; <<Gazeback [Tutor] [Target]>>

This command can be used to request a Tutor to initiate\terminate a gaze action.

Note: Any gaze action that is initiated will continue to be performed until the appropriate, opposite command is issued.

Command	Tutor	Target
Gazeat Gazeback	Maria Joao	Maria Joao User

Here are some usability examples of these commands.

Example	Description
<<Gazeat Maria Joao>>	Tutor "Maria" will gaze at tutor “Joao”
<<Gazeback Joao User>>	Tutor "Joao" will stop gazing at the user.

You can consult the *GazeExample.yarn.txt* file, located in the folder “Character Animation Commands”, for an example dialog that showcases this command.

Frequency and speed

<<[Action] [Tutor] [Parameter] [Value]>>

Every movement discussed previously can also have its frequency and speed adjusted. This affects, for example, how fast a Tutor speaks, and how frequently he will acknowledge the other Tutor by nodding (and how fast that nodding will be).

Note: These values are dependent of (and will vary depending on) the emotion each Tutor is currently feeling. This command allows them to also be altered at any time via the dialog system, if so desired.

Action	Tutor	Parameter	Value
Talk Nod Gazeat Gazeback	Maria Joao	Speed Frequency	0.0 ≤ … ≤ 2.0 (speed) 0.0 ≤ … ≤ 1.0 (frequency)

Here are some usability examples of these commands.

Example	Description
<<Gazeback Maria Speed 2.0>>	Doubles the speed at which tutor “Maria” gazes back from looking at others.
<<Talk Joao Speed 0.5>>	Halves the speed at which tutor "Joao" talks.
<<Nod Maria Frequency 0.5>>	Makes it so there is a 1 in 2 chance that “Maria” will nod her head when acknowledging what the other tutor is saying.
<<Gazeat Joao Frequency 0.25>>	Makes it so there is a 1 in 4 chance that “Joao” will gaze at the other tutor, when that tutor is speaking.

You can consult the *SpeedAndFrequencyExample.yarn.txt* file, located in the folder “Character Animation Commands”, for an example dialog that showcases this command.

Bubble System Commands

The bubble system module controls the static elements of the scene (backgrounds and balloons).

A variety of commands can be sent, via the dialog creation system, to issue various requests to each of them.

The commands have different input values, which will later be specified in detail in the following sections.

Text Style

Contrary to the other commands, text style is performed in an exclusive way. The strings within the VT application accept rich text, and for that reason the dialogue can be written with the corresponding tags.

The generic format of tags is as follows.

<attribute=value>	Overrides the attribute with that specific value.

Note: When it comes to color, the format used is |RRGGBBAA instead of #RRGGBBAA.

Here are some usability examples:

Example	Output	Description
<color=|FF0000FF>Hello.	Hello.	Sentence in red.
How are you?	**How are you?**	Sentence in bold.
Hello	***H**ello*	Sentence in italic, with the first letter in bold.

For more information, tags and examples check the TextMesh Pro Manual: http://digitalnativestudios.com/textmeshpro/docs/rich-text/

You can consult the *TextStyle.yarn.txt* file, located in the folder “Bubble System Commands”, for an example dialog that showcases this command.

Set Next Dialogue Data Command

<<SetNextDialogueData [Tutor] [Emotion] [Intensity] [showEffects effect [curve]] (*optional*) [hideEffects effect [curve]] (*optional*)>>

Balloons, like most elements in the scene, are defined based on the emotion and intensity that wants to be transmitted. A dialogue is triggered every time a new sentence appears in yarn, therefore the emotion needs to be set beforehand. It is usually defined with tags, however, it is possible to override the way the next dialogue balloon will be presented with a command.

There are optional parameters, which are the text animations to use, if given. Inside the text animation parameters, the curve to use is also optional in some effects. The exceptions will be detailed in a following section. If not given, the predefined animations will be used.

The showEffects refers to the text animations to be performed when the balloon is appearing, whilst the hideEffects refers to the text animations to be performed when the balloon is disappearing. Both of these parameters are optional.

The accepted values for each property of the command are listed on the following table.

Command	Tutor	Emotion	Intensity
SetNextDialogueData	Maria Joao	Neutral Happiness Sadness Anger Fear Disgust Surprise	0.0 ≤ … ≤ 1.0

Note: This command only affects the next line for a specific tutor.

Note: When it comes to text animations, the order matters, and the result may have inconsistencies.

Note: One can achieve a "Neutral" behavior in two ways:

- By giving a value of 0.0 to the field "Intensity". In this case, it is irrelevant which "Emotion" option is used.
- By giving the option "Neutral", for the emotion. In this case, it is irrelevant which value is used for "Intensity".

Here are some usability examples of the "SetNextDialogueData" command.

Example	Description
<<SetNextDialogueData Maria Happiness 0.8>>	Forces the next balloon by “Maria” to be very happy.
<<SetNextDialogueData Joao Sadness 0.5 showEffects FadeIn linearCurve>>	Forces the next balloon by “Joao” to be *moderately* sad. It also forces the show text animation to be a fade in, with a linear curve.

<<SetNextDialogueData Joao Sadness 0.5 hideEffects FadeOut linearCurve>>	Forces the next balloon by “Joao” to be *moderately* sad. It also forces the hide text animation to be a fade out, with a linear curve.
<<SetNextDialogueData Joao Sadness 0.5 showEffects FadeIn linearCurve Shake hideEffects FadeOut linearCurve>>	Forces the next balloon by “Joao” to be *moderately* sad. It also forces the show text animation to be a fade in with a linear curve and a shake, and the hide text animation to be a fade out, with a linear curve.

Note: A detailed list of all the text animations and curves can be found in the following command.

You can consult the *SetNextDialogueData.yarn.txt* file, located in the folder “Bubble System Commands”, for an example dialog that showcases this command.

Override Text Effects Command

<<OverrideTextEffects [Emotion] [Intensity] [showEffects effect [curve] (*optional*)] [hideEffects effect [curve] (*optional*)]>>

If needed, it is possible to override the text effects defined for a specific emotion and intensity.

There are optional parameters, which are the text animations to override. Inside the text animation parameters, the curve to use is also optional in some effects. The exceptions will be detailed in a following section. At least one type should appear in the command (showEffects or hideEffects).

The showEffects refers to the text animations to be performed when the balloon is appearing, whilst the hideEffects refers to the text animations to be performed when the balloon is disappearing.

The accepted values for each property of the command are listed on the following table.

Command	**Tutor**	**Emotion**	**Intensity**
OverrideTextEffects	Maria Joao	Default Neutral Happiness Sadness Anger Fear Disgust Surprise	0.0 ≤ … ≤ 1.0

Note: Default applies to the options balloons.

Here is a detailed list of all the effects that can be used:

Effect	Description	Requires Curve
None		No
Appear	Show text letter by letter	Yes
Blush	Change color over time	Yes
BlushCharacters	Change color over time, letter by letter	Yes
DeflectionFont	Diminish text font size over time	Yes
Erase	Erase text over time	Yes
FadeIn	Fade in text over time	Yes
FadeInCharacters	Fade in text letter by letter over time	Yes
FadeOut	Fade out text over time	Yes
FadeOutCharacters	Fade out text letter by letter over time	Yes
Flashing	Flashes text over time	Yes
Jitter	Jitters text over time	No
Palpitations	Palpitates text over time	Yes
Shake	Shakes text over time	No
ShakeCharacters	Shakes text letter by letter over time	No
Squash	Squashes text over time	Yes
SquashX	Squashes text over time, but only on x axis	Yes
SquashY	Squashes text over time, but only on y axis	Yes
Stretch	Stretches text over time	Yes
StretchX	Stretches text over time, but only on x axis	Yes
StretchY	Stretches text over time, but only on y axis	Yes
SwellingFont	Augment text font size, over time	Yes
Swing	Swings text over time	No
SwingCharacters	Swings text letter by letter, over time	No
Warp	Warps text over a curve	Yes
WarpCharacters	Warps each letter of text over a curve	Yes
Wave	Waves text over time	Yes

WaveCharacters	Waves text over time, letter by letter	Yes

Here is a detailed list of all the animation curves that can be used:

Animation Curve	Description
bellCurve	Bell curve from 0 to 1
linearCurve	Linear curve from 0 to 1
flashCurve	Custom curve to flash text
lowerBellCurve	Bell curve from -0.25 to 0.25
palpitationCurve	Custom curve to palpitate text

Here are some usability examples of the "OverrideTextEffects" command.

Example	Description
<<OverrideTextEffects Happiness 0.8 showEffects FadeIn linearCurve>>	Overrides the show text animation of happiness with an intensity of 0.8 to use a fade in with a linear curve.
<<OverrideTextEffects Happiness 0.8 hideEffects FadeOut linearCurve>>	Overrides the hide text animation of happiness with an intensity of 0.8 to use a fade out with a linear curve.
<<OverrideTextEffects Happiness 0.8 showEffects fadeIn linearCurve Shake hideEffects FadeOut linearCurve>>	Overrides the show text animation of happiness with an intensity of 0.8 to use a fade in with a linear curve and shake, and the hide text animation to use a fade out with a linear curve.

You can consult the *OverrideTextEffects.yarn.txt* file, located in the folder “Bubble System Commands”, for an example dialog that showcases this command.

Override Blush Color Command

```
<<OverrideBlushColor "Color">>
```

It is possible to override the blush color used by the Blush effect. Blush effect is the effect that changes the color of the sentence over time.

Note: Red is the color used by default.

The accepted values for each property of the command are listed on the following table.

Command	Color
OverrideBlushColor	Color in #RRGGBBAA format

Here are usability examples of the "OverrideBlushColor" command.

Example	Description
<<OverrideBlushColor #FF0000FF>>	Overrides the blush color with red.
<<OverrideBlushColor #00FF00FF>>	Overrides the blush color with blue.

You can consult the *OverrideBlushColor.yarn.txt* file, located in the folder “Bubble System Commands”, for an example dialog that showcases this command.

Set Mix Colors Command

<<SetMixColors [ShouldMix]>>

It is possible to set whether to mix colors or not, when defining the emotion colors. If disabled, the colors used in the backgrounds and balloons will always be the same, independent of the intensity (eg. happiness -> yellow). If enabled, the color will be mixed with white, returning a lighter color (eg. happiness 0.5 -> lighter yellow).

Note: This is enabled by default.

The accepted values for each property of the command are listed on the following table.

Command	ShouldMix
SetMixColors	0 (false) 1 (true)

Here are usability examples of the "SetMixColor" command.

Example	Description
<<SetMixColors 0>>	Disables color mixing.
<<SetMixColors 1>>	Enables color mixing.

You can consult the *SetMixColors.yarn.txt* file, located in the folder “Bubble System Commands”, for an example dialog that showcases this command.

Set Balloon Animation Blending Command

<<SetBalloonAnimationBlending [ShouldBlend]>>

It is possible to set whether to blend the balloon animations (either with intensity or multiple emotions). If disabled, the balloon animation of the highest emotion will be performed with an intensity of 1. If enabled, the animation will perform a blend

between the neutral animation and the corresponding emotion(s) the tutor has in that moment, each with its intensity (eg. happiness 0.3 and anger 0.6 will result in a blend between happiness with a weight of 0.3, anger with a weight of 0.6, and neutral with a weight of 0.1).

Note: This is disabled by default.

The accepted values for each property of the command are listed on the following table.

Command	ShouldBlend
SetBalloonAnimationBlending	0 (false) 1 (true)

Here are usability examples of the "SetBalloonAnimationBlending" command.

Example	Description
<<SetBalloonAnimationBlending 0>>	Disables balloon animation blending.
<<SetBalloonAnimationBlending 1>>	Enables balloon animation blending.

You can consult the *SetBalloonAnimationBlending.yarn.txt* file, located in the folder “Bubble System Commands”, for an example dialog that showcases this command.

Set Balloon Duration Command

<<SetBalloonDuration [Duration]>>

It is possible to set how long the balloon will be shown (its duration).

Note: By default, the duration is 5 seconds.

The accepted values for each property of the command are listed on the following table.

Command	Duration
SetBalloonDuration	>= 2

Here are usability examples of the "SetBalloonDuration" command.

Example	Description
<<SetBalloonDuration 5>>	Sets the balloon duration to 5 seconds.
<<SetBalloonDuration 2>>	Sets the balloon duration to 2 seconds.

You can consult the *SetBalloonDuration.yarn.txt* file, located in the folder “Bubble System Commands”, for an example dialog that showcases this command.

Set Background Duration Command

<<SetBackgroundDuration [Duration]>>

It is possible to set how long the background will take to transition to a different emotion or reason (its duration).

Note: By default, the duration is 5 seconds.

The accepted values for each property of the command are listed on the following table.

Command	Duration
SetBackgroundDuration	>= 2

Here are usability examples of the "SetBackgroundDuration" command.

Example	Description
<<SetBackgroundDuration 5>>	Sets the background transition duration to 5 seconds.
<<SetBackgroundDuration 2>>	Sets the background transition duration to 2 seconds.

You can consult the *SetBackgroundDuration.yarn.txt* file, located in the folder “Bubble System Commands”, for an example dialog that showcases this command.

Override Emotion Color Command

<<OverrideEmotionColor [Emotion] [Color]>>

It is possible to override the color defined for each emotion (eg. define happiness as red).

Note: This will change the color for both the backgrounds and the balloons.

The accepted values for each property of the command are listed on the following table.

Command	Emotion	Color
OverrideEmotionColor	Neutral Happiness Sadness	Color in #RRGGBBAA format

	Anger Fear Disgust Surprise	

Here are usability examples of the "OverrideEmotionColor" command.

Example	Description
<<OverrideEmotionColor Happiness #FF0000FF>>	Overrides happiness color to red.
<<OverrideEmotionColor Happiness #0000FFFF>>	Overrides happiness color to blue.

You can consult the *OverrideEmotionColor.yarn.txt* file, located in the folder “Bubble System Commands”, for an example dialog that showcases this command.

Set Force Text Update Command

<<SetForceTextUpdate [ForceUpdate]>>

When a text animation occurs, the last supposed value of the effect is forced, for consistency. This means that even if a FadeIn does not end within the duration specified, or the animation curve used does not end in 1, the alpha will be forced to the maximum value in the end. This allows the correct effect to be shown if slight errors happen.

However, It is possible to define not to update text after the duration completes. This can be useful, if different curves are being used. For example, a FadeIn with a bell curve would fade in and then fade out the alpha of the text. If ForceUpdate is enabled, the alpha will be forced to 1 in the end (the common fade in behaviour). If disabled, the alpha will remain with the value after the fadeOut.

Note: This is enabled by default.

The accepted values for each property of the command are listed on the following table.

Command	ForceUpdate
SetForceTextUpdate	0 (false) 1 (true)

Here are usability examples of the "SetForceTextUpdate" command.

Example	Description
<<SetForceTextUpdate 0>>	Disables force text update.

<<SetForceTextUpdate 1>>	Enables force text update.

You can consult the *SetForceTextUpdate.yarn.txt* file, located in the folder “Bubble System Commands”, for an example dialog that showcases this command.

Add Animation Curve Command

<<AddAnimationCurve [Name] [Time] [Value] [smooth Weight (*optional*)] …>>

If needed, it is possible to add more animation curves, which can then be used in text animations. For that, a custom name should be given, followed by at least two keyframes (time-value pairs). There is the possibility to smooth the tangents of the curve (optional) with a specific weight (should only be used in intermediate keyframes).

For more information on smooth tangents, check Unity documentation: https://docs.unity3d.com/ScriptReference/AnimationCurve.SmoothTangents.html

Note: The creation of an animation curve, curves the edges by default.

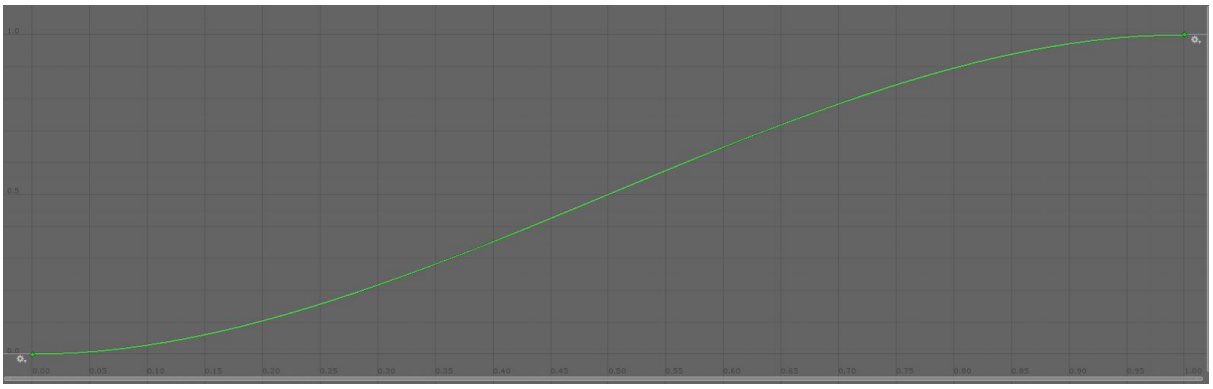
The accepted values for each property of the command are listed on the following table.

Command	Name	Time	Value	Weight
AddAnimationCurve	Custom name	X value	Y value	0.0 ≤ … ≤ 1.0

Here are usability examples of the "AddAnimationCurve" command.

Example	Description
<<AddAnimationCurve myCurve 0 0 1 1>>	Creates a new curve called “myCurve” with one point as 0 in the X axis and the Y axis, and a second point with 1 in the X axis and the Y axis (linear with edges curved).

Below is an image of the curve created with the previous command:



You can consult the *AddAnimationCurve.yarn.txt* file, located in the folder “Bubble System Commands”, for an example dialog that showcases this command.