**Dialog System API**

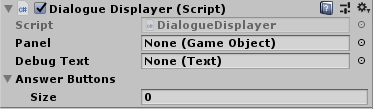
**Description:**

The dialog system is used to display dialog on the screen. This is done by reading the data from text files and displaying it to the screen. You will have to create your own dialog files and scenario files.

**Setup:**

You need to make a GameObject which has the script DialogueDisplayer. The Dialogue Displayer has a couple of public variables. (Also please note that you can only use 1 DialogueDisplayer in a scene.)

These public variables consist of:

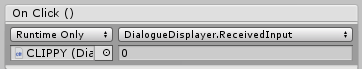


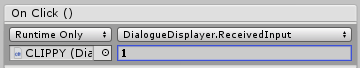
**-Debug Text**: The name isn’t all that good but use this variable to put in the text object that you want the dialog to display on.

**-Answer buttons:** these are all the buttons that you want to display the answers for questions on. (Important) Put all these buttons one 1 panel. The text on the buttons will display the answers for the questions. Also the answer buttons can be increased dynamically. If you only want 2 answer buttons than you can assign 2 buttons. But if you want to have 10 answers than assign 10.

**-Panel:** This is the panel you but all your buttons on. Drag this panel into the assigned slot.

Also on each button make a reference to DialogueDisplayer.ReceivedInput(int); And put in the numbers according to the answer you want it to be, if you want this button to display the first answer put in a zero, do you want it to display the second answer put in the number 1, and so on. This is important so please do this.





**Creating the dialog files:**

**Description:**

These are the files you have to make for the system to know what dialog to display. The system works with 2 separate files. A Scenario file and a Dialogue file.

**Setup:**

Please make folders in unity exactly like stated below. The Scenario folder will hold all the scenario files and the Script folder will hold all the dialogue files.

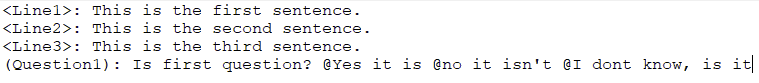


Inside the Scenario Folder please make a text file. Make sure the extension of the file is .sce and not .txt. otherwise it will not work. You can also give the file any name you want.

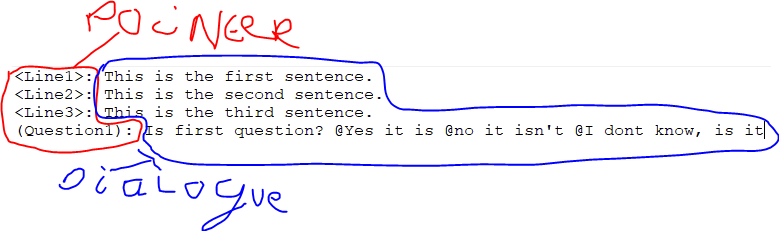
And in the scripts folder please make a text file with the extension .kdialog, again make sure it does NOT say .txt otherwise it will not work. You can also name this file however you like.

(if you have trouble opening these files please right click on the file and press open with and select your text editor of choice, if you don’t know what text editor to use please use notepad)

**Creating the dialog file:**

I will take you through all the steps but a very basic dialogue file will look a bit like this:  


As you can see the file consists of pointers and the dialogue lines itself. Pointers and dialogue lines always get separated by a semi colon (‘:’). It is important that you don’t use any colons in the dialogue lines itself since this may cause issues.



**Pointers:**

There are 2 different types of pointers. Ones used for normal sentences and ones used for questions they are distinguished by the use of <> and ().

For a normal sentence you use the <>. You can name your pointers by putting a name in between the <>. For example <Line1>. Make sure you never have 2 pointers with the same name otherwise the other pointer will never be found.

For a question use () instead of <>. The same rules apply here, you can put a name in between the (). For Example (Question1). Make sure you never have 2 pointers with the same name otherwise the other pointer will never be found.

Important!: after a pointer always put a colon. For example: <Line1>: or (Question1): . If you don’t do this the sentence will never be read. Always start a pointer on a new line of text.

**Dialogue lines:**

**Sentence:**

After you have made the pointer for a sentence you can write the sentence. You can put in anything except for a colon. (And even this will probably even work). Write the sentence in the same line as a pointer, and put it after the colon.

For example: <Line1>: This is the first line of dialogue.

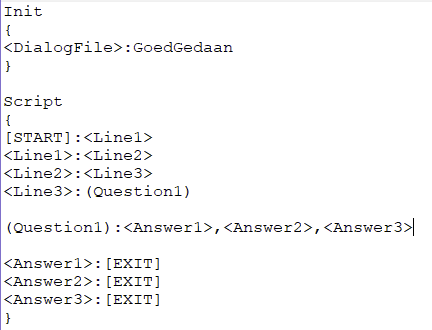
**Question:**

Questions work almost the same, after the colon put your question. This is like any other dialogue line. But for the different answers to the question please start each answer with a @. After the @ you can write the answer to the question like any other dialogue line.

For example: (Question1): Is this a question? @Yes it is @No it isn’t @I don’t know

**Scenario files:**

A basic scenario file will look a bit like this:



It may look complicated but it actually is quite simple really.

**Init:**

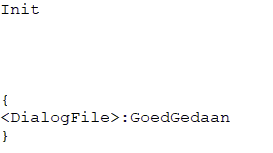
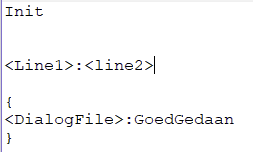
**Description:**

Init is used to specify what dialog file is going to be used. It will specify what lines will display after the correct pointers are called.

**Setup:**

In order to setup the Init please type ‘Init’ (this is case sensitive). In order for the Init to be useful you will have to put { } after the Init. This doesn’t necessarily have to be right behind the Init or directly underneath it as long as there isn’t anything else in between the Init and the { }.

For Example:

Is allowed ^ Is NOT allowed ^

**Specify the dialog file:**

**Description:**

You will have to specify the dialog file that you want to use, otherwise the system doesn’t know what lines to use.

**Setup:**

In order to specify the Dialog file, first write <DialogFile>: in between the { } right after the Init. After the <DialogFile>: please write the full name of the dialogue file without the .kdialog. Note that these names have to be exactly the same.

For example:

Init:

{

<DialogFile>:DialogFile1

}

**Script:**

**Description:**

With the script you will specify in what order the dialog lines you wrote in the dialog file will execute and display.

**Setup:**

In order to setup the script do the same as Init but instead of Init write ‘Script’ (this is case sensitive)

For example:

Script

{

}

The same rules as Init apply here.

**Setting up the script:**

**Description:**

The script only uses pointers and commands in order to specify how the dialog will execute.

**Setup:**

**Commands:**

There are only 2 commands

[START] and [EXIT]

[START] is used to specify where the dialogue will start. This is the line that gets called ones the dialogue starts. And will always be the first pointer.

[EXIT] is used to specify where the dialogue will stop.

**Pointers:**

The pointers are the same as in the .kdialog file. Please note that the pointers have to be written exactly the same as in the .kdialog file otherwise it will not work.

**How to make a script:**

You can split the script into 2 parts the left side and the right side. They get split in the middle with the colon. The left side is the side that gets called and will display the dialogue on the screen and the right side is used to specify what pointer will get executed next.

Know this setting up the script will be quite easy. I will explain how it works first without the use of questions.

To start write the command [START] and put a colon after that. After that put the first pointer you want to go to. It will look a bit like this:

[START]:<Line1>

In this example the next line you want to go to is <Line1>. So start a new line (this can be anywhere in between the { }) starting with <Line1> and specify what you want to do, do you want to go to a next line of dialogue or do you want to stop it. If you want to stop it simply write. (Note that <Line1> can be any pointer that you have made, as long as the name is the same as in the .kdialog file)

<Line1>:[EXIT]

If you want the dialog to continue please put another pointer after the colon, it will look like this.

[START]:<Line1>

<Line1>:<Line2>

<Line2>:<Line3>

<Line3>:[EXIT]

Now for implementing questions. Questions also have pointers so in order to point to a question first point to the question like this:

[START]:<Line1>

<Line1>:(Question1)

Now you have a pointer to a question. You can set the pointers to the answers of the question. It will look something like this

[START]:<Line1>

<Line1>:(Question1)

(Question1):<Answer1>,<Answer2>,<Answer3>

As you can see the question has multiple pointers separated by a comma ‘,’. In the dialog file you made answers with the @. Well the pointers point to the order the answers are written in, the first answer will point to <Answer1> and the second will point to <Answer2> and so on. You can also point to another question:

(Question1):<Answer1>,(Question2),Answer3>

After this you will only have to specify the correct pointers in the file it will look something like this:

[START]:<Line1>

<Line1>:(Question1)

(Question1):<Answer1>,<Answer2>,<Answer3>

<Answer1>:[EXIT]

<Answer2>:[EXIT]

<Answer3>:[EXIT]

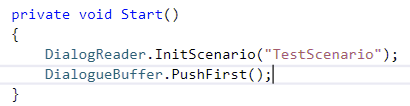
The exits can be chanced for any other pointer. At the end you will always have to put an [EXIT] though otherwise the dialogue won’t close properly.

**How to display dialogue from code:**

**Description:**

Using these methods dialogue will display on your screen after you setup the first part correctly.

**Setup:**



After the initial setup is done you can setup the dialogue like this. In order to put the dialogue into the buffer which has to be done first run the line:

DialogReader.InitScenario( /\* name of the scenario file as a string \*/);

Put in the exact name of the scenario file.

The moment you want to display the dialogue on the screen run the code DialogueBuffer.PushFirst(); the first dialogue in the buffer will display on the screen now.

|  |  |
| --- | --- |
| DialogReader.InitScenario(string); | This will setup the scenario into the dialogue buffer. After you have done this you can push the first dialogue and it will display. |
| DialogueBuffer.PushFirst(); | Will display the first dialogue in the buffer on the screen. |
| DialogueBuffer.DiscardFirst(); | Will remove the first dialogue from the buffer. |
| DialogueBuffer.DiscardAll(); | Will remove all the dialogue from the buffer. |
| DialogueDisplayer.textSpeed; (int) | ++ or – this for a quicker or slower text speed. Please not that if this goes under zero your text will never display and your dialog will never move on. |