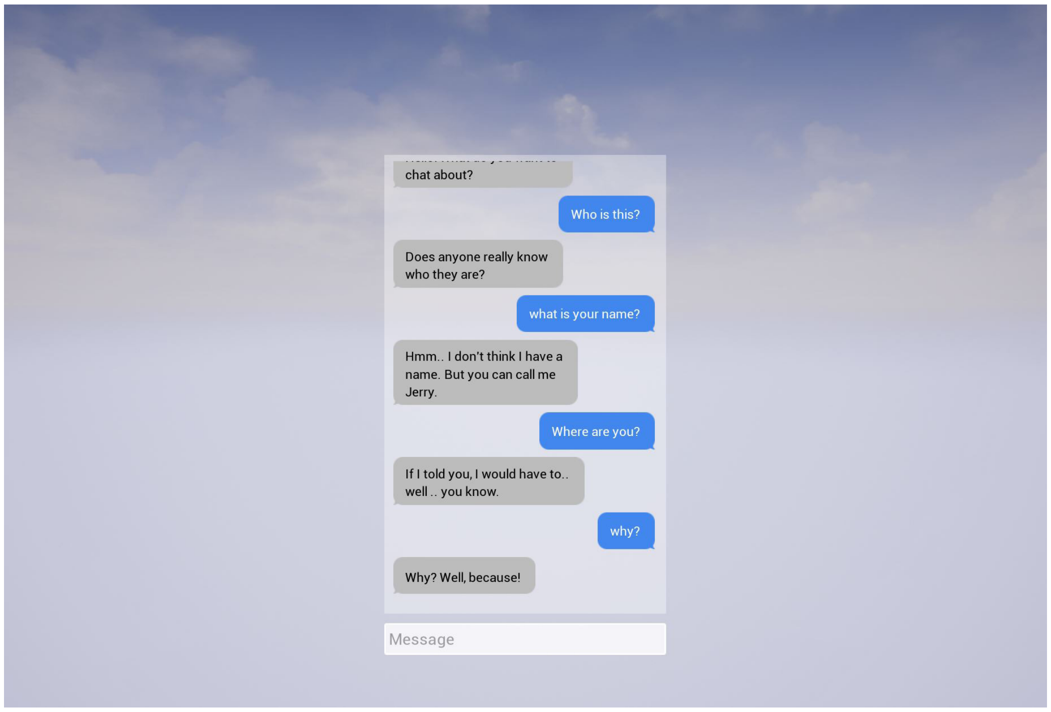


chat about?

Who is this?

Does anyone really know who they are?

what is your name?

Hmm.. I don't think I have a name. But you can call me Jerry.

Where are you?

If I told you, I would have to.. well .. you know.

why?

Why? Well, because!

CVChatBotAI Simulator

For Unreal Engine

Instructions for Customizing and Usage

By

creativevilla.com

System Overview

A 100% Blueprint driven Conversation Bot that simulates communication with Artificial Intelligence.

CVChatBotAI is a 100% Blueprint driven Conversation Bot framework that simulates artificial intelligence by responding to questions and comments in a casual English language manner. Great for simulating communication with a human – for Adventure games with multiple Non human characters, interaction with a computer terminal with 'someone' on the other side of the network, and for giving out Hints to players as they ask it questions about quests or anything at all. Add as many variations and conversation branches as you want, add your own key words and replies – completely customizable. Comes with a nifty mobile phone Text Messenger interface to give you an idea of how it might be used.

Features:
Based on hierarchical English language structure based on key words (people, places and things), action verbs, single words, and multi-word phrases – all customizable.
Simply type in any request, comment or question in regular English grammar, and the ChatBot will respond accordingly encouraging you to ask more questions.
Easy to modify/Customize with basic blueprint knowledge.
Entire system runs inside 1 main HUD blueprint widget.
Accompanying Text Message simulator requires 2 additional customizable widgets also included.

Can your chat bot pass the Turing test?

Technical Details
Previous knowledge of how blueprints work is recommended for editing and customizing
Intended platform: All

Any questions, comments, suggestions, please contact me at:
support@creativevilla.com

Best wishes, Jerry

Contents

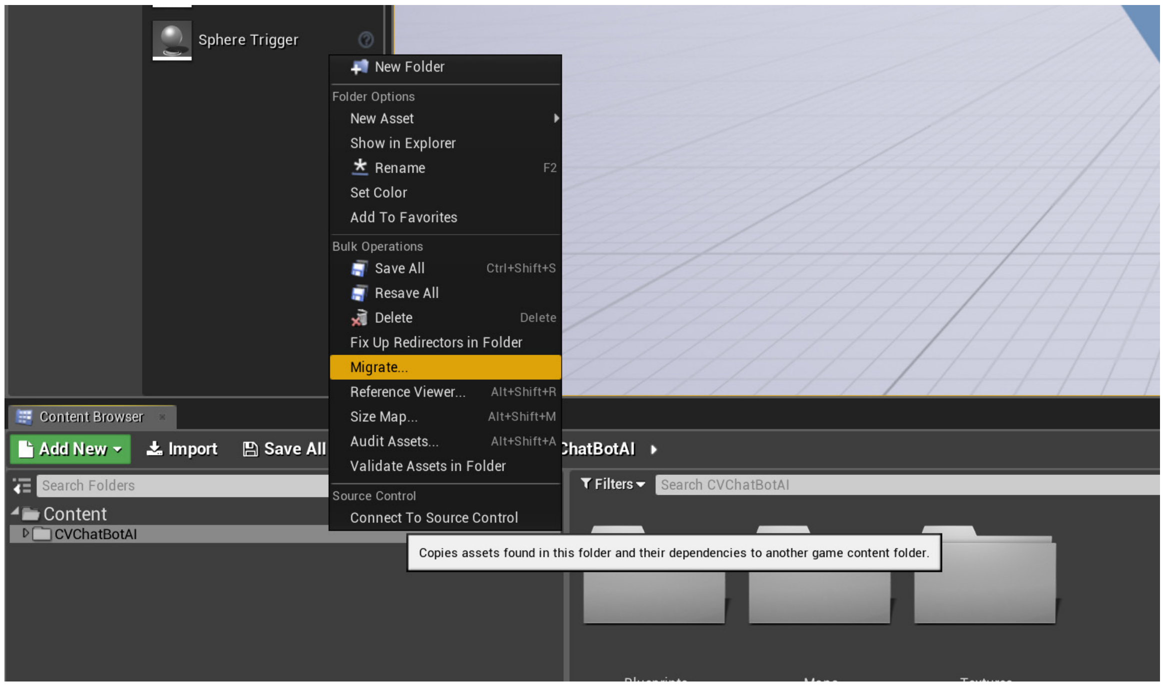
Step 1 Migrating to new or existing project folder

1) Open original CVChatBoAI project you want to migrate from (the marketplace project)

2) In the Content Browser, locate the folder: CVChatBotAI

3) Click on the CVChatBotAI folder to select it.

4) Right click and select Migrate



Copies assets found in this folder and their dependencies to another game content folder.

5) A window will come up showing you all the components being migrated. Click OK.

6) Navigate to your new project folder structure and locate your Content Folder

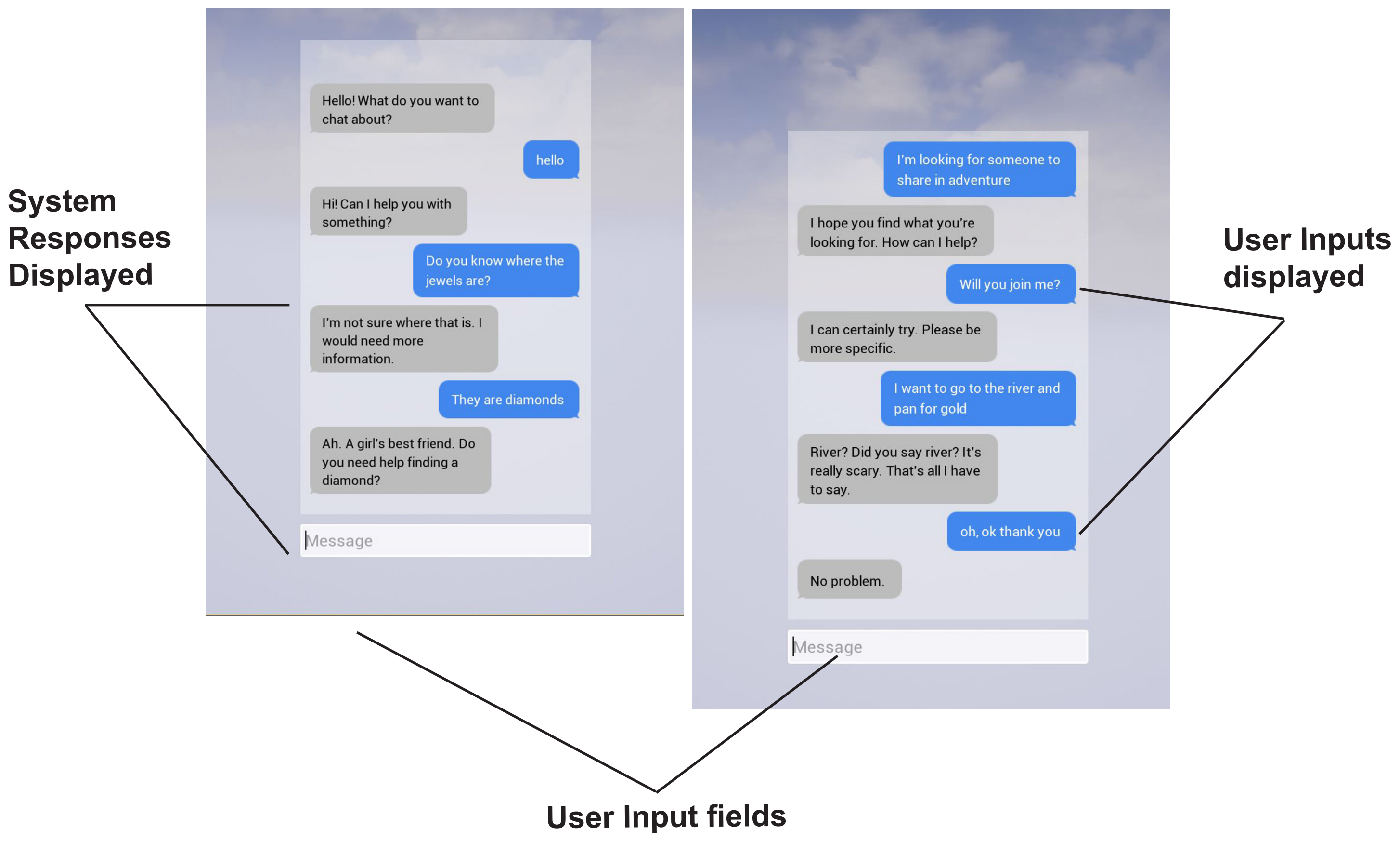
7) Select the Content folder in your new project and click OK

8) When done you will receive a message: Content migration completed successfully.

9) Close the original CVChatBotAI Marketplace project

SYSTEM OVERVIEW

The included ‘Text Message’ interface is a good example of how to use the ChatBotAI in your games or simulations. It has a user input, and a system response, and scrolls to show you a history of inputs and responses. The default setup keeps 10 of each (20 total) messages on the screen that you can scroll up and down to reveal. After 20 messages, the 21st one is removed from the screen. This is customizable to as many or few as you want. Just click PLAY on the Overview map - and you will see the below screen appear and ready for your input.



Hello! What do you want to chat about?

hello

Hi! Can I help you with something?

Do you know where the jewels are?

I'm not sure where that is. I would need more information.

They are diamonds

Ah. A girl's best friend. Do you need help finding a diamond?

I'm looking for someone to share in adventure

I hope you find what you're looking for. How can I help?

Will you join me?

I can certainly try. Please be more specific.

I want to go to the river and pan for gold

River? Did you say river? It's really scary. That's all I have to say.

oh, ok thank you

No problem.

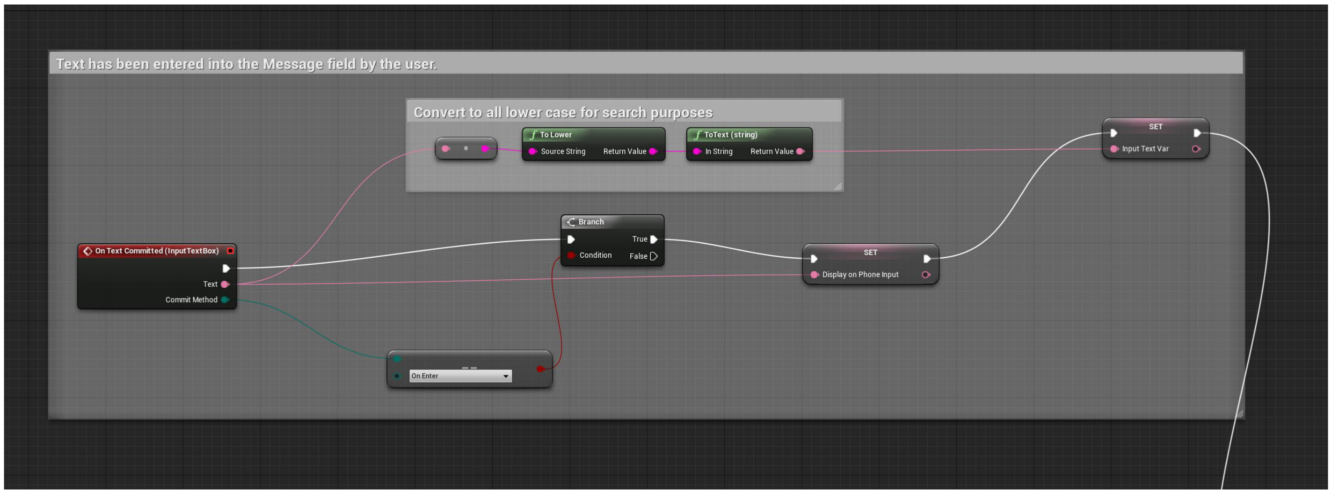
The default setup (out of the box) is set up for a ‘helpful’ bot that is here to assist you in finding things related to a possible quest. (Mountain, River, Diamond, Puppies, etc)

HOW IT WORKS

The ChatBotAI system is an Artificial Intelligence Simulator, designed to give the impression that the communicator on the other end is not just a robot with canned pre-determined responses.

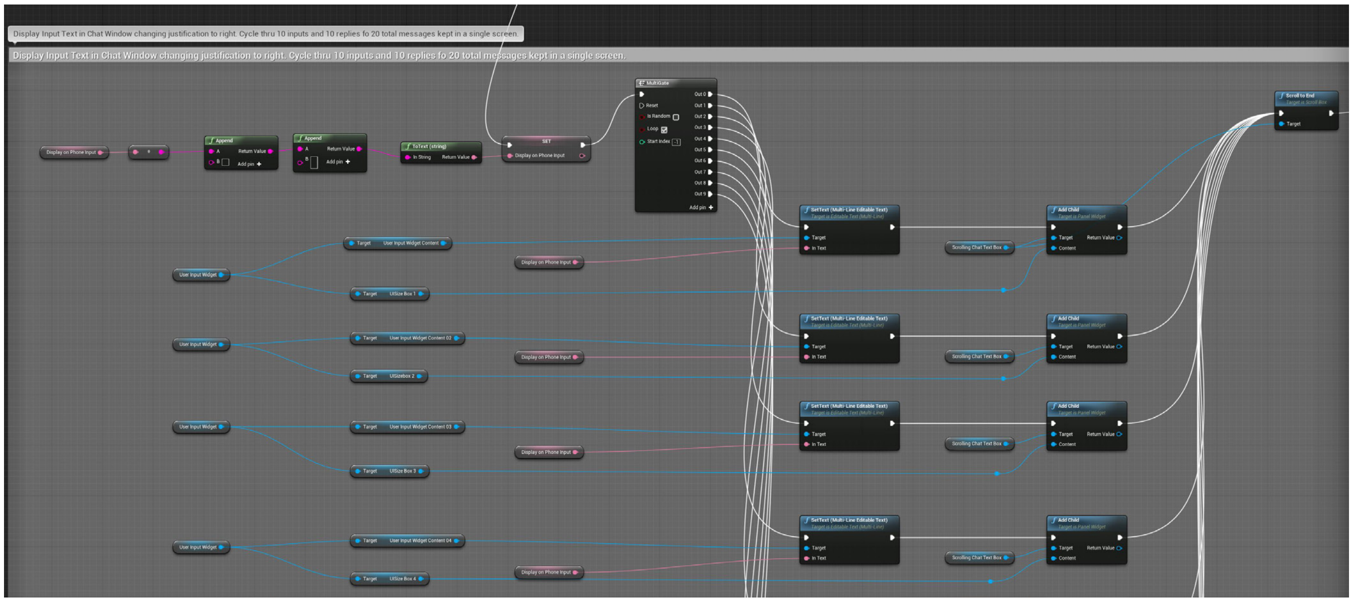
This is done through a Hierarchy search through the user's input for specific KEYWORD Substrings that might relate to a specific answer (ie: Where is the mountain?).

Hierarchy searching order goes like this: **Multikeyword** (Steve + Mountain), then **Keyword** (Mountain), then, **Single Words** (yes, no, what, where), then **Action Words** (running, eating, hunt, talk), then **Specific Questions** (what is the meaning of life?) and then **Single Phrases** (I like, I Don't Know, Do you Want), and finally a **Catch all** that handles every other possible input that the previous searches did not pick up.

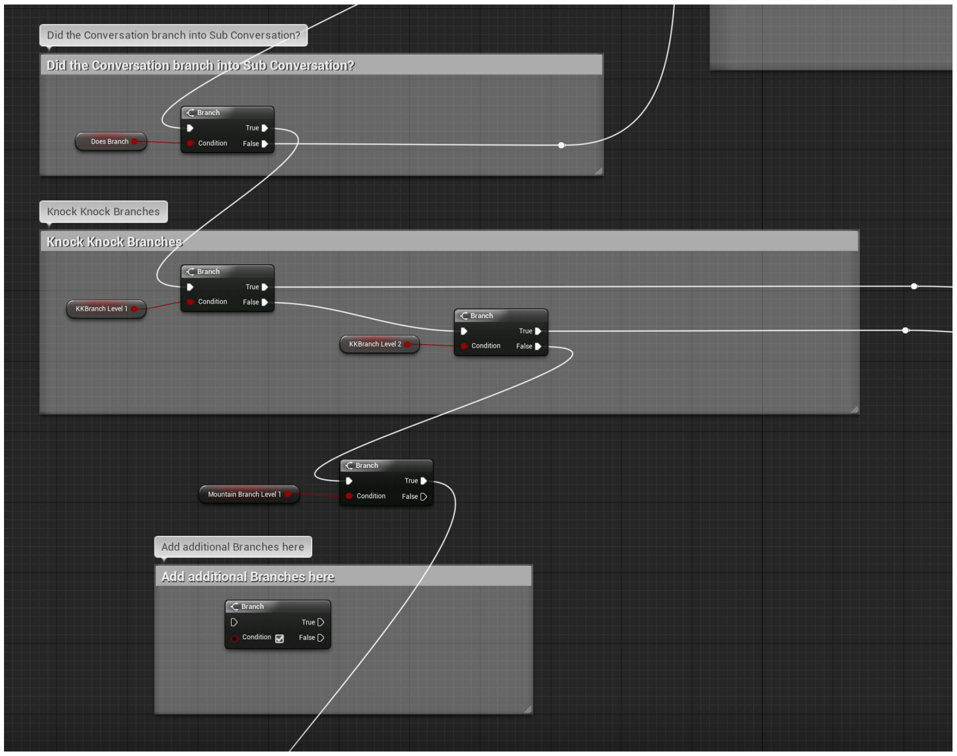


When a user inputs text into the input dialog box and presses Enter - the system sets the **InputTextVar** to equal the users input (with capitalization taken away for search purposes. It also sets a **Display on Phone Input** - which displays the correct capitalization when the system spits the users input back out onto the screen.

The included Text Message interface then takes this data and re-outputs it onto the screen (you type it in, press enter, and then it appears on the phone). If you wanted to direct this output to someplace else, you would do that before the **Set Display on Phone Input** node.

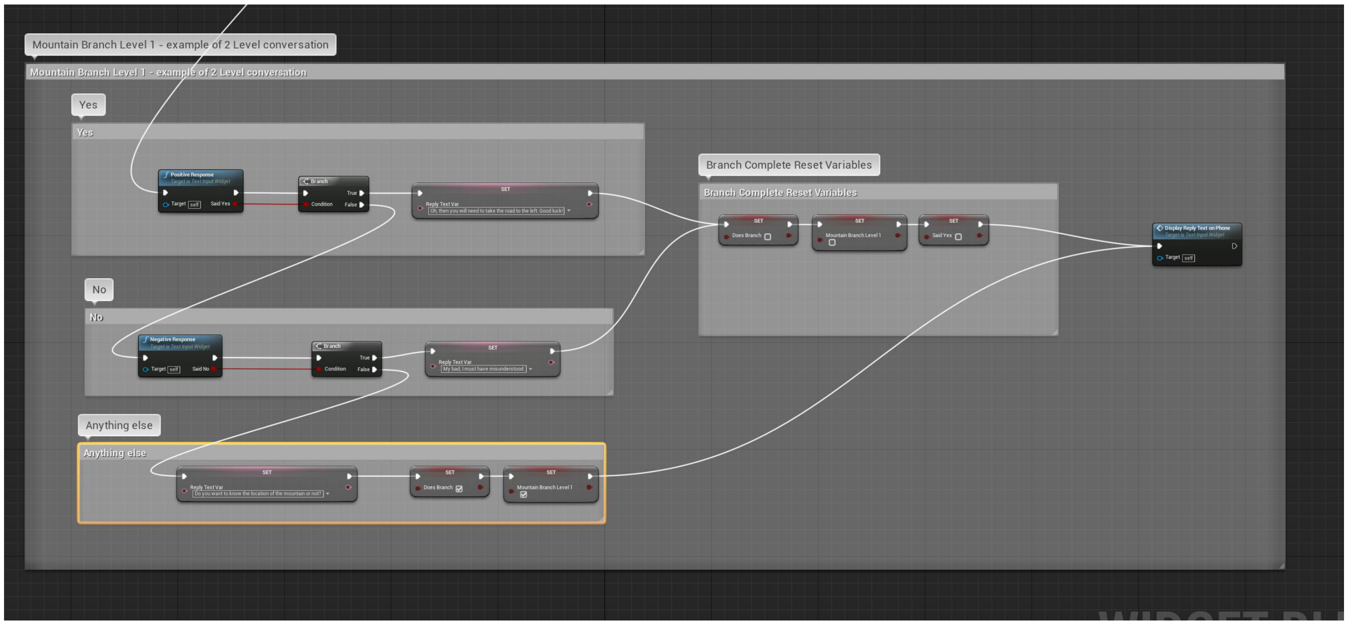

Display Input Text in Chat Window changing justification to right. Cycle thru 10 inputs and 10 replies fo 20 total messages kept in a single screen.
Display on Phone Input
Append
Append
ToText (string)
SET
MultiGate
SetText (Multi-Line Editable Text)
Add Child
Scroll to End
User Input Widget
Scrolling Chat Text Box

Next, the system checks to see if this is a new input, or if you are answering a previously asked question by the ChatBot - we call this a Conversation Branch. If there is no existing branch, the system goes on to do its substring searches. If **Does Branch** is currently set to YES, then it will continue on down the branches until it finds the one that is Set to YES, and then do what it is told.



Below is an example of what the Mountain Branch looks like. When the user first types in ‘mountain’ inside of a input , the system sees it and sets the boolean **Does Branch** and **Mountain Branch Level 1** to YES. **Does Branch** tells the system that there is a Branched conversation taking place. The **Mountain Branch Level 1** tells the system to look for answers to very specific question the Bot just asked and tells the system that THIS is the branch to follow and stay inside of until resolved.

While inside of a conversation branch, inputs from the user are ONLY checked within the branch and not the entire system - this focuses attention on the BOT’s question and your answer to that question (or non- answer).



Oh, then you will need to take the road to the left. Good luck!

My bad, I must have misunderstood.

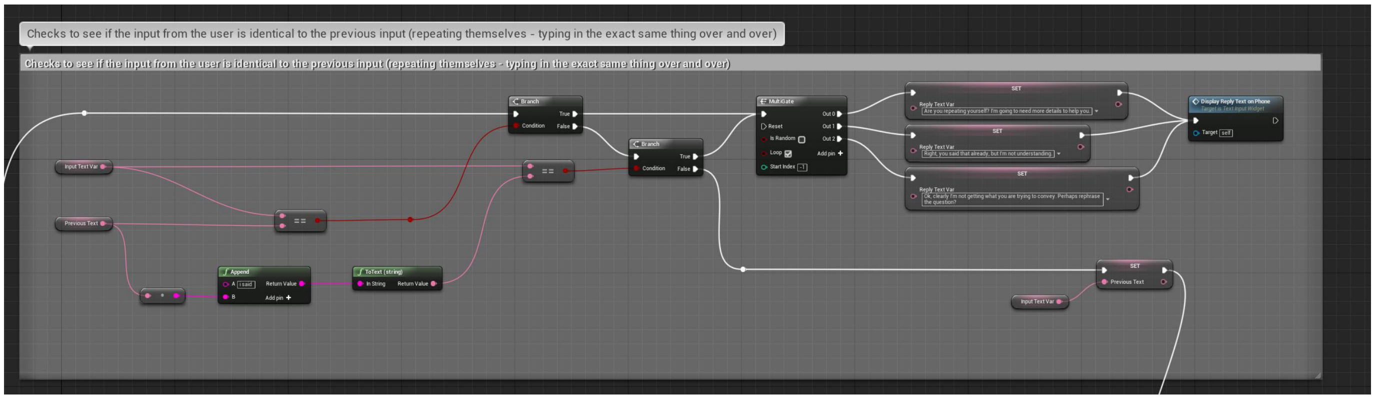
Do you want to know the location of the mountain or not?

In the above conversation branch, the only options for input are a Positive response (yes, yea, sure, etc), a Negative response (no, never, nope) and Anything else. If the user types anything other than a positive or negative response, the system spits out an encouraging question to get the user to pick one of the two directions. Note - you can put multiple responses to Anything Else - so it feels more natural if the User continues to not provide a Yes or No answer.

Branches can go as deep and complex as you want. The knock knock joke provided is an example of a 2 branch conversation.

User input: Knock Knock	Bot Response: Who’s there? (Branch 1)
User input: A little old lady	Bot Response: A little old lady who? (Branch 2)
User input: I didnt know you could yodel!	Bot Response: I don’t get it. (Reset branches)

The system then checks to see if the input from the user is the exact same input as the previous input. (user repeating the exact same request or word over and over again).


Checks to see if the input from the user is identical to the previous input (repeating themselves - typing in the exact same thing over and over)

Reply Text Var: Are you repeating yourself? I'm going to need more details to help you.

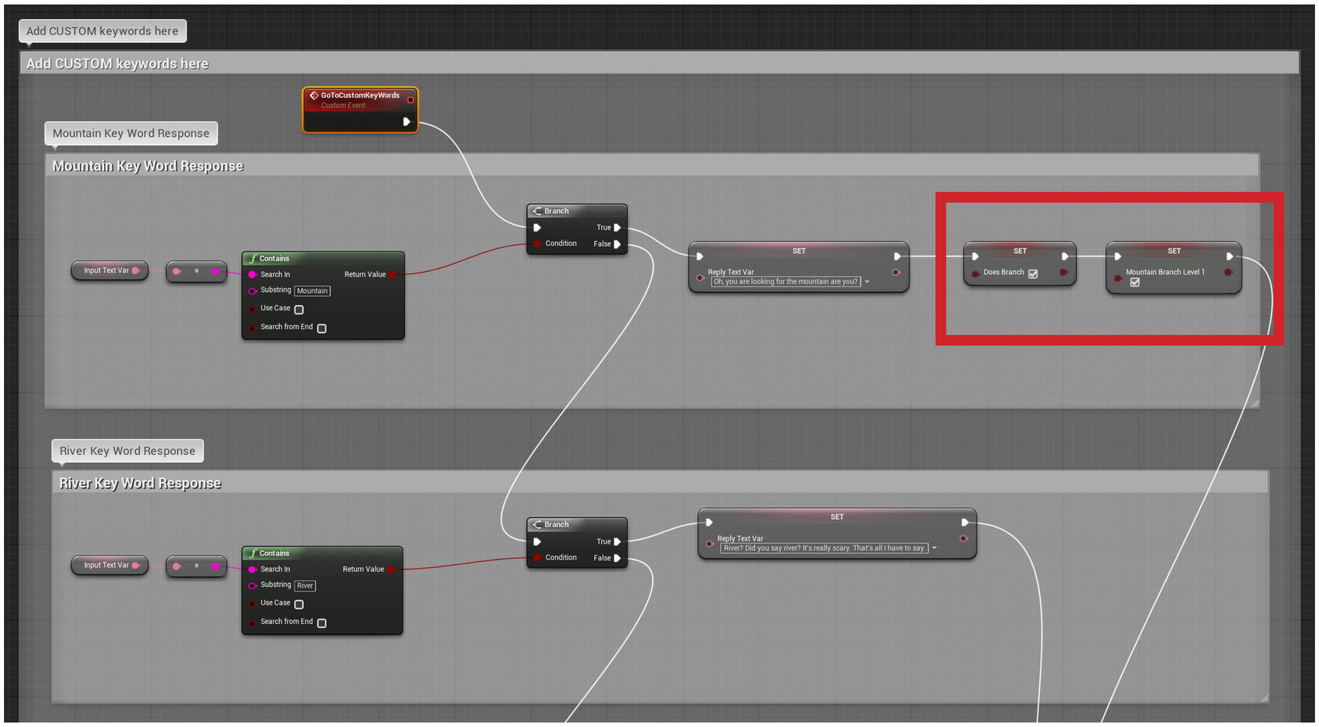
Reply Text Var: Right, you said that already, but I'm not understanding.

Reply Text Var: Ok, clearly I'm not getting what you are trying to convey. Perhaps rephrase the question?

After the very first input from the user, the system sets **Previous Text** variable to equal that input. Then the next input is compared to the **Previous Text**. If they are equal, that means the user has repeated the same question or comment back to back, and the system can respond accordingly. With all these examples, we have provided some default responses the system can give, you can change these or add as many as you like. The **Multigate** in the image above, can be used to randomize these responses or loop through them in order - which allows you to increase the annoyance or inquiries of the bot.

From here the system looks for Keywords and Multiple Keywords that the developer has put in as important words that should elicit a specific response from the system if that word appears anywhere in the sentence.

KEYWORDS SEARCH FRAMEWORK


Add CUSTOM keywords here

Mountain Key Word Response

Reply Text Var: Oh, you are looking for the mountain are you?

River Key Word Response

Reply Text Var: River? Did you say river? It's really scary. That's all I have to say.

Adding additional Keywords into the mix is simple, just duplicate one of the Keyword set of nodes in the previous image above, and paste in between two other nodes, and then reconnect the nodes accordingly: (**False** output goes to the next **Branch** input, **Set** output node goes to **Display Reply Text on Phone** input.

Note the RED box in the previous image highlighting the Branch variables being set. The system has detected the word ‘mountain’ in the user’s input. It sets Does Branch, and sets Mountain Branch Level 1 as positive. The next time the user inputs anything - it will branch into the Mountain Branch until resolved. Once resolved, the Branch variables are cleared back to negative.

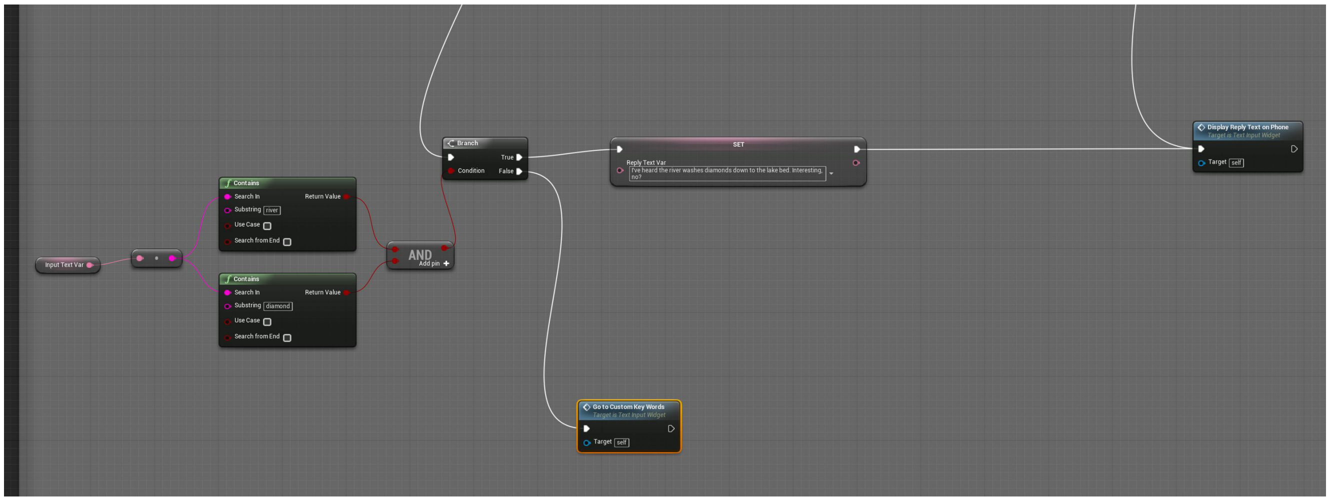
KEYWORDS SEARCH SUMMARY

Typing the phrase “Where is the Mountain?” contains 3 items which the system looks at/ recognizes.
1) Where is   2) Mountain   3) ?
It recognizes “mountain” as a key word the Developer has put in as an important word, that might elicit specific responses.
It recognizes “Where is” as a potential question regarding a Person, Place or Thing’s location.
It recognizes “Where” by itself as too little information to give an educated guess as to the users intention.
And finally it sees the ? as a question to be answered - and can keep track of how many questions you ask that do not have answers. The default is 5 un -answered questions - after which the system responds accordingly. (try it!)

Typing in the following phrases may get you these corresponding responses

Input: **Where is** the **mountain?**	Output: Oh, you seek the **mountain** do you?
Input: Have you seen the **mountain?**	Output: I have seen the **mountain**, do you want directions to it?
Input: **Where is it?**	Output: I’m not sure **where that is**.
Input: **Where?**	Output: **Where** is what?
Input: **Where is** the gobbly gook**?**	Output: I’m not sure **where that is**, do you have a different question?
Input: **Um, do you know how to sail?**	Output: I’m afraid **I don’t know that**. Perhaps if you were more specific.

MULTIPLE KEYWORDS SEARCH FRAMEWORK



MULTIPLE KEYWORD SEARCH SUMMARY

Typing the phrase “Steve told me you know where the Mountain is.” contains 2 items which the system looks at/recognizes. See above image showing the 2 **Contains** nodes, and the **AND** boolean. To turn this 2 keyword search into a 4 keyword search, you would add two additional **Contains** nodes, add 2 more pins to the **AND** node, and you are all set.

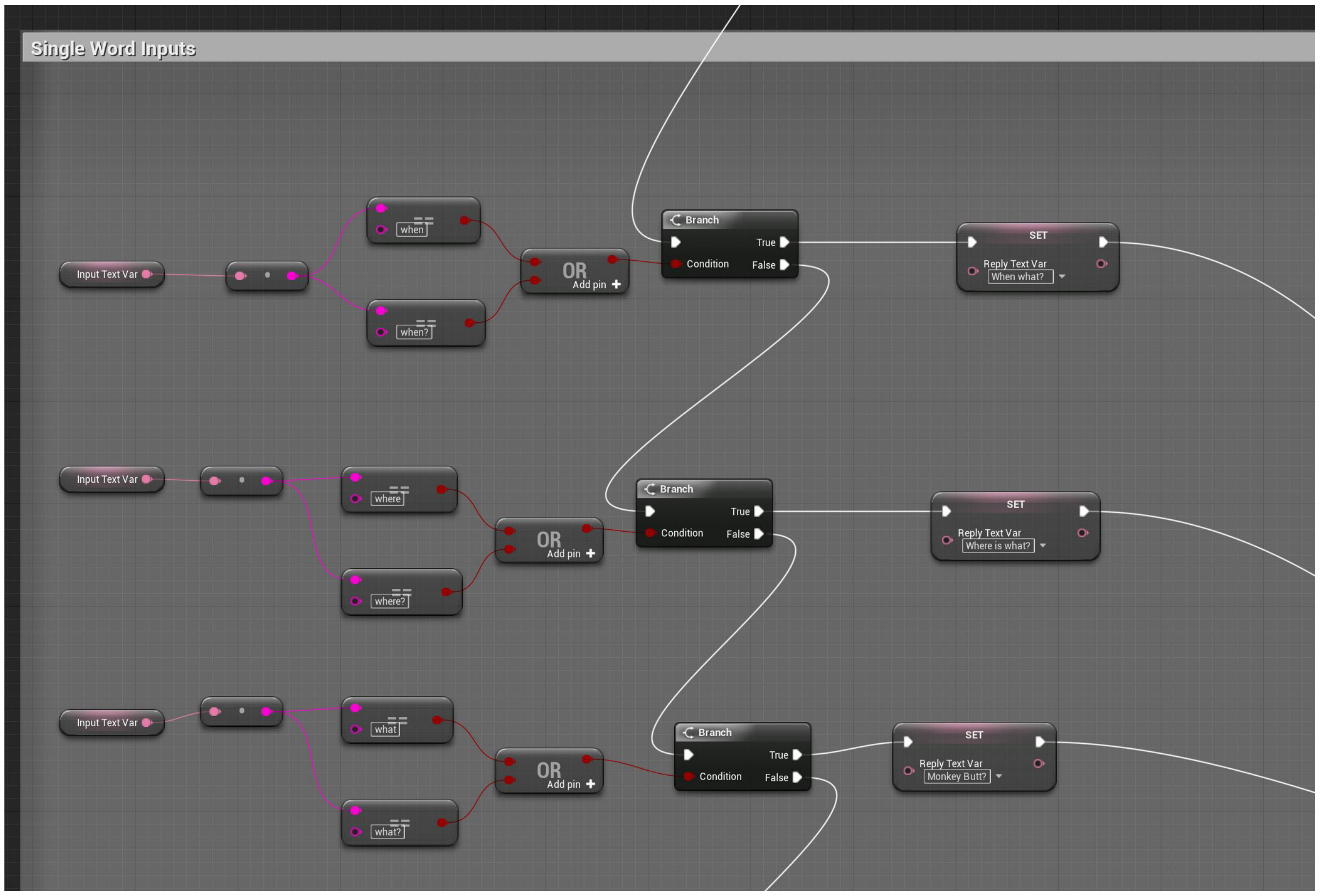
1) Steve 2) Mountain

It recognizes “mountain” and “steve” as Multiple key words the Developer has put in as an important word(s), that might elicit specific responses. You can have as many keywords in a sentence as you want, and customize the system to respond to each scenario.

Typing in the following phrases may get you these corresponding responses

Input: **Where is** the **mountain?**	Output: Oh, you seek the **mountain** do you?
Input: **Steve** said there was a **mountain?**	Output: Yes, **Steve** always loved that **mountain**.

SINGLE WORD INPUT FRAMEWORK

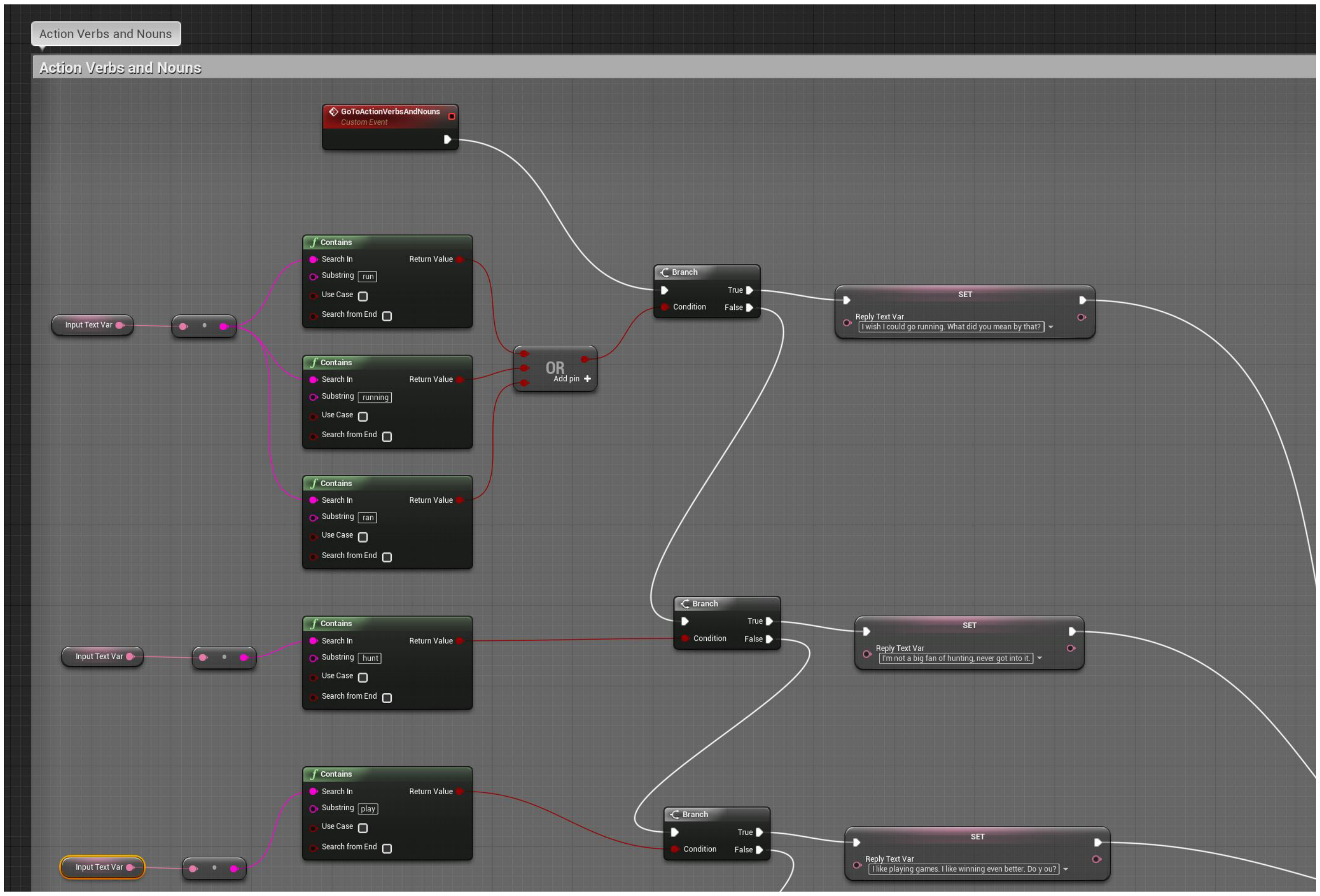


SINGLE WORD INPUT SUMMARY

Single words inputs are inputs that contain just that: 1 word in a sentence. An example of 1 word might be ‘thanks’, ‘bye’, ‘nothing’, ‘how’, etc. You can add as many single words as you want in this section. If these words appear in a larger sentence, they won’t get picked up as single words. Single words usually contain no context, unless answering a question, or asking a single word question like “what?” Other examples of single word entries may be “OMG”, “LOL”, etc to which the system can respond in-kind.

Input: **Wait**	Output: **Waiting...**
Input: **Hey!**	Output: **Hey** is for horses!

ACTION VERBS AND NOUNS FRAMEWORK



I wish I could go running. What did you mean by that?

I'm not a big fan of hunting, never got into it.

I like playing games. I like winning even better. Do y ou?

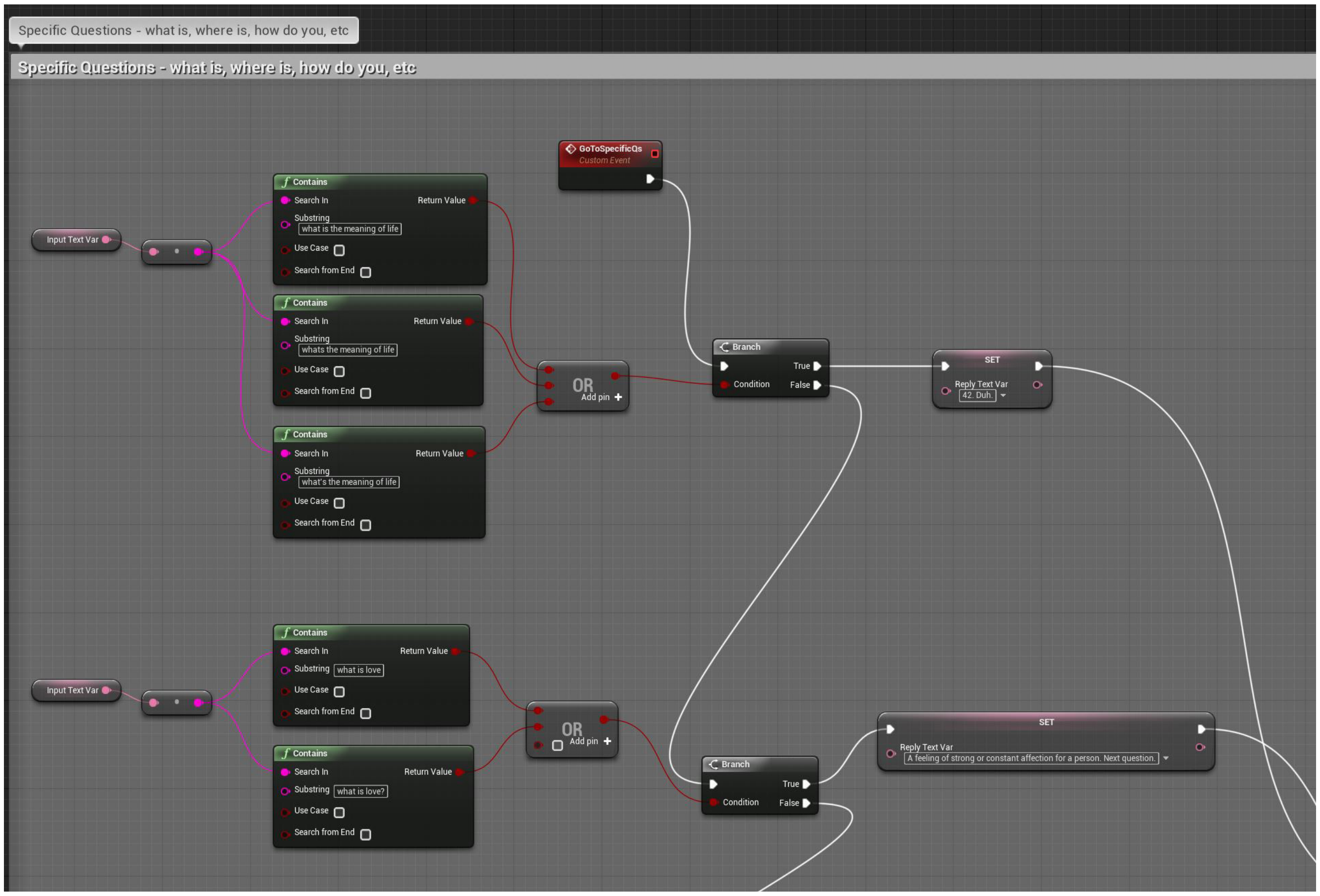
ACTION VERBS AND NOUNS SUMMARY

Action Verbs and Nouns is the section to put single words or short phrases that represent verbs in a sentence (hunt, run, die, play,) - by looking for the root substring only you also pick up the rest of the verbs (**Play** = **Play**ing, **Play**ed as well) NOTE: If you look closely, the verbs and nouns all start with a space _ . This is to insure that they are indeed a part of a larger sentence, and not inside another word. Example: ICE is part of RICE but _ICE is not. For words that the system gets confused on, just add additional versions of the root word to compensate. Example: Run is also part of RUNT, and _RUN is part of _RUNT. To avoid this, you could add a space before and after run: _RUN_, and then add in _RUNNING as a separate search term.

Input: I like to **Run** Output: I wish I could go **running.**

Input: Do you want to **Play?** Output: I like **playing** games, don't you?

SPECIFIC QUESTIONS FRAMEWORK



SPECIFIC QUESTIONS SUMMARY

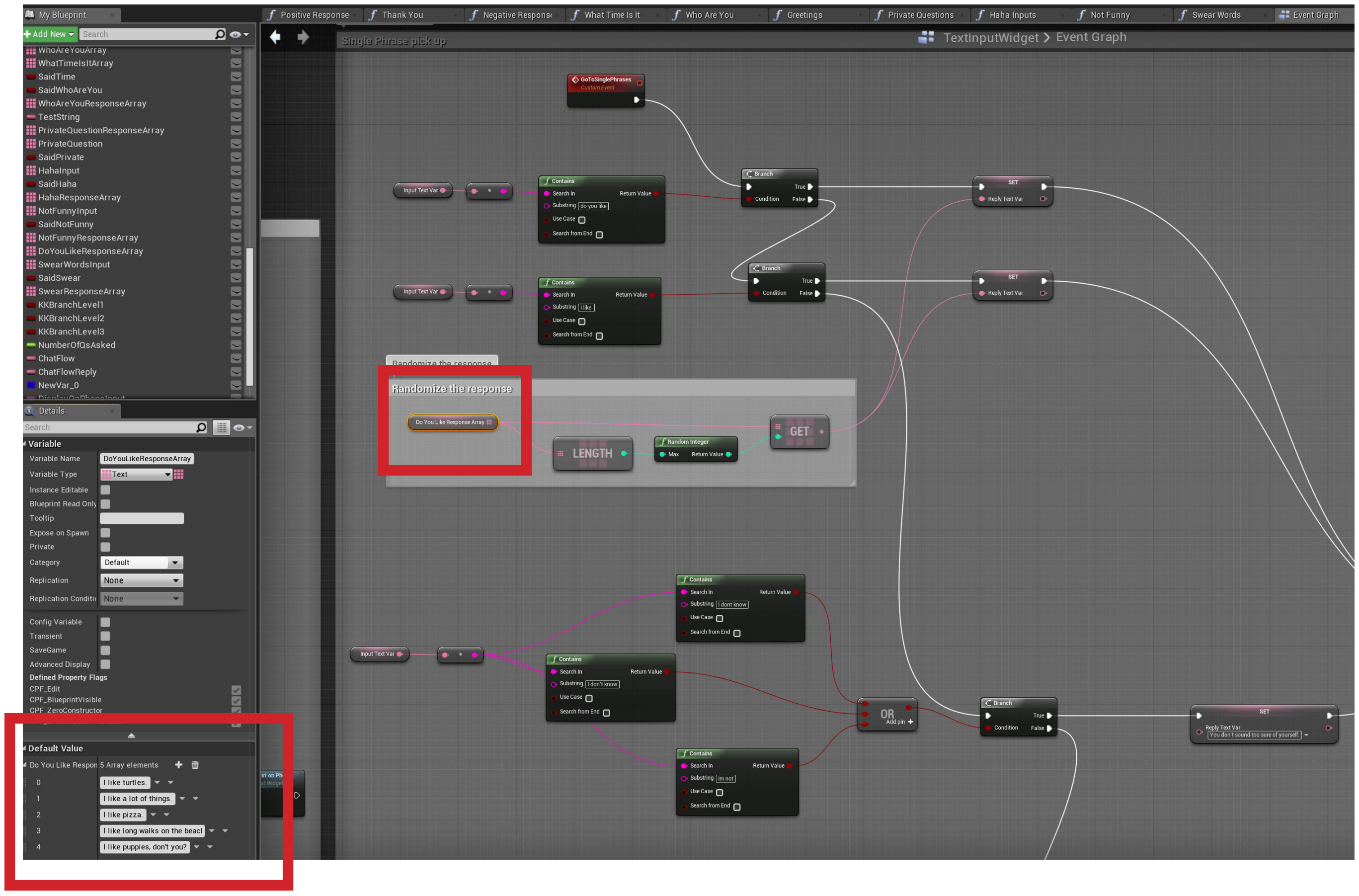
The specific questions section deals with exact questions that you want the user to input. These can be short or long, and as long as the user inputs them to match your substring searches, they can be either stand alone or part of a larger sentence.

To add a new Specific Question to the system, just copy and past an existing one, and place it between two others and reconnect the nodes, then change the substring searches and the responses. To add additional responses use a MultiGate node to offer multiple possible replies via a random or looped system.

Input: **What is the meaning of life?** Output: **42. Duh!**

Input: **The password is inconceivable.** Output: **That's right! You may come in!**

SINGLE PHRASE FRAMEWORK



Details

Variable

Variable Name: DoYouLikeResponseArray
Variable Type: Text
Instance Editable
Blueprint Read Only
Tooltip
Expose on Spawn
Private
Category: Default
Replication: None
Replication Condition: None
Config Variable
Transient
SaveGame
Advanced Display
Defined Property Flags
CPF_Edit
CPF_BlueprintVisible
CPF_ZeroConstructor

Default Value

Do You Like Respon 5 Array elements

0	I like turtles.
1	I like a lot of things.
2	I like pizza.
3	I like long walks on the beacl
4	I like puppies, don't you?

SINGLE PHRASE SUMMARY

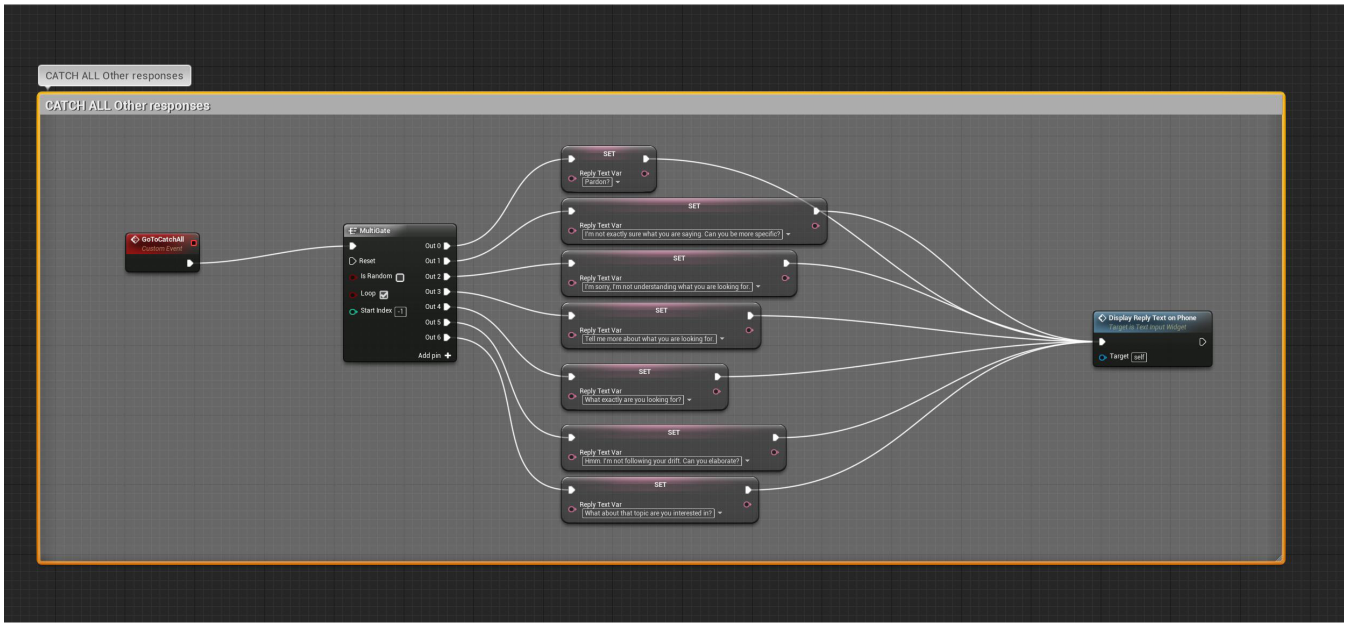
Single phrases can be treated like any other word or keyword. These are small phrases that are part of a larger sentence (not the entire sentence it self). These usually start a sentence. Example: **Can you....** is a single phrase the system can then respond to no matter what comes next. Same with **Do you**, **Have you**, **Are you**, etc. They can also end or be in the middle of a sentence to get picked up if a response is desired. Example: _is stupid, **_went fishing**, etc) Multiple responses to any question can be put into an array. See above, instead of having multiple nodes out in the blueprint. Arrays can then be accessed randomly or in order.

Input: **Can you juggle?** Output: **I'll try anything once.**

Did you have anything else to ask?

Input: **My cousin went fishing yesterday.** Output: **I have never been fishing, is it fun?**

CATCH ALL OTHER FRAMEWORK



Pardon?

I'm not exactly sure what you are saying. Can you be more specific?

I'm sorry, I'm not understanding what you are looking for.

Tell me more about what you are looking for.

What exactly are you looking for?

Hmm. I'm not following your drift. Can you elaborate?

What about that topic are you interested in?

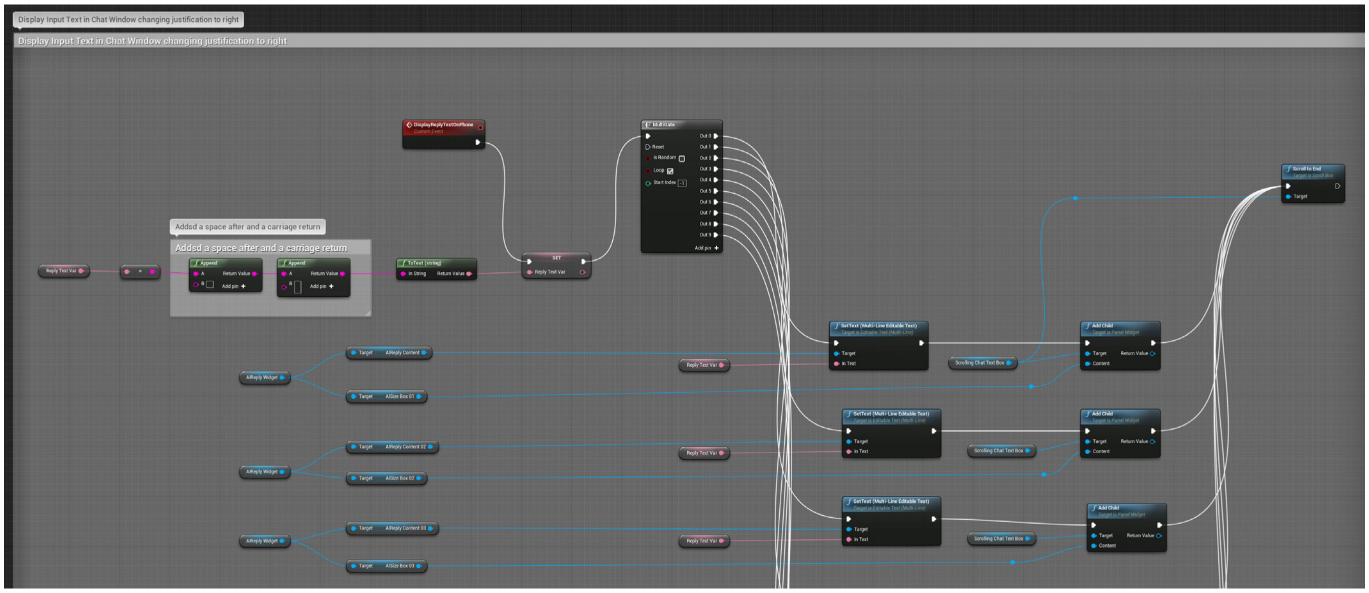
CATCH ALL OTHER SUMMARY

If after all the keywords, phrases, verbs and single phrases are searched for and come up with nothing, the catch all catches all, and gives the system a way to respond to help the user ask the correct questions or steer them down a path to know what to ask. You can have as many of these as you want to help the system look more ‘smart’ as it answers non-answerable questions with unique inquiries to attempt to get the user to ask a question with a key word or other phrase.

Input: **Mondays are boring.**	Output: **Hmm. I’m not following your drift. Are you looking for something?**
Input: **The sky is blue.**	Output: **That’s nice. Did you have a question for me?**

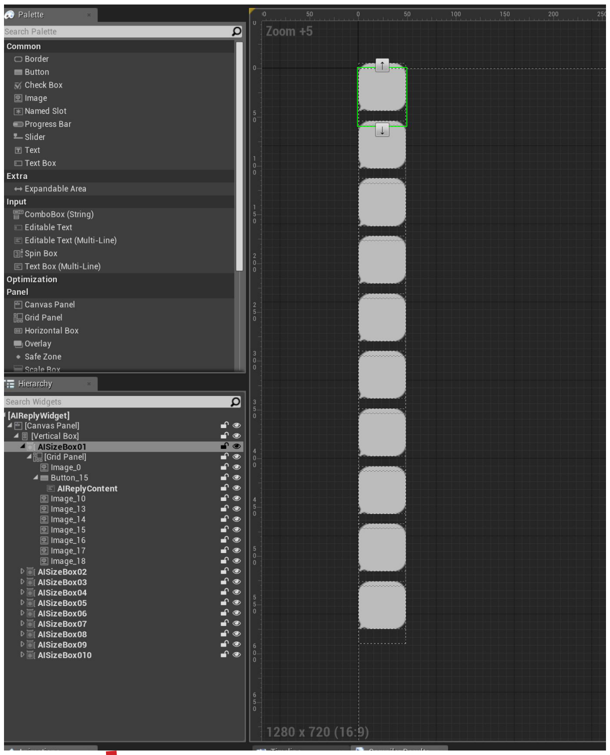
OUTPUT CHATBOT RESPONSES TO THE SCREEN

As with the user input. The same technique for putting responses to the screen is used for the ChatBot.

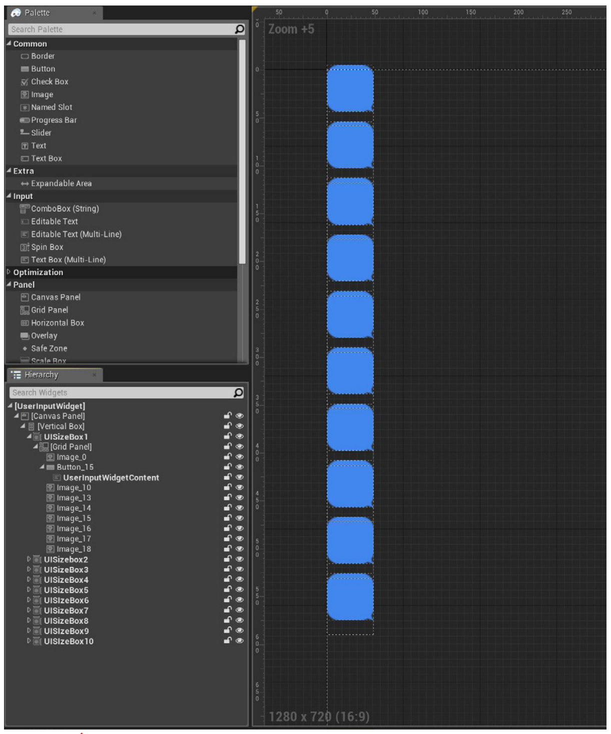


HOW THE TEXT MESSAGE PHONE INTERFACE WORKS

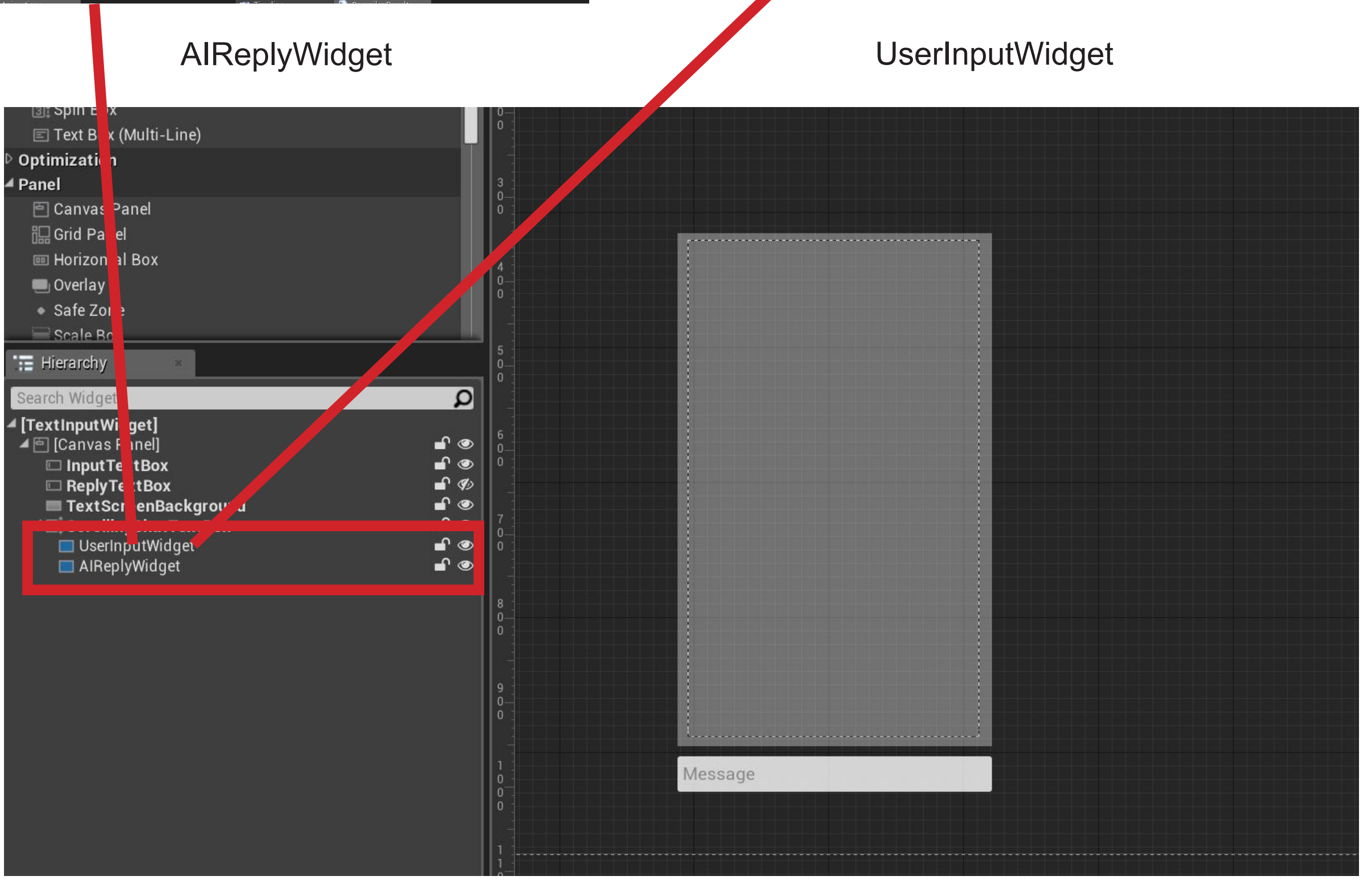
The phone interface uses 2 separate other Widgets to drive inputs into the main TextInputWidget.



AIReplyWidget



UserInputWidget

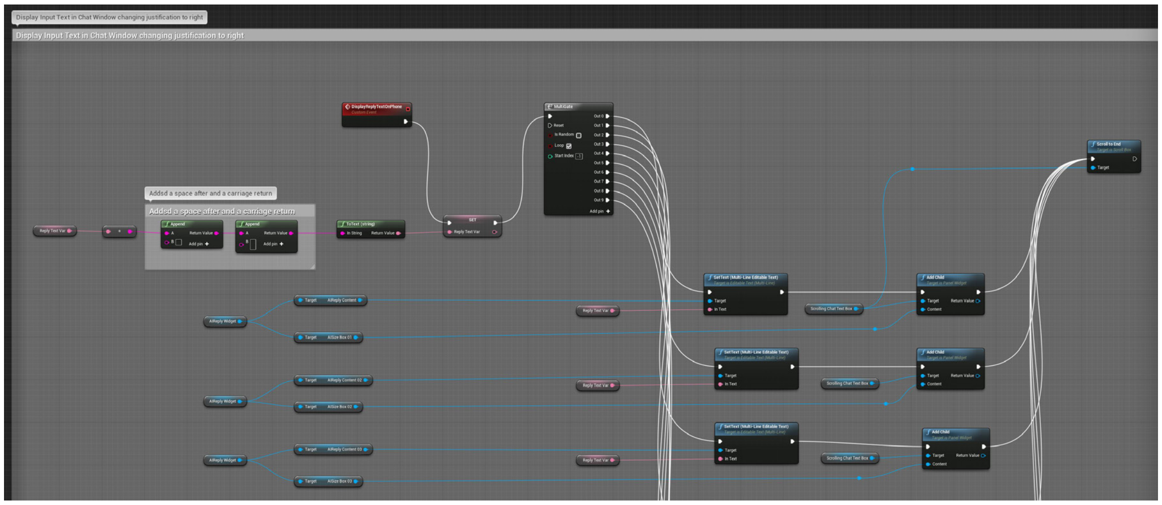


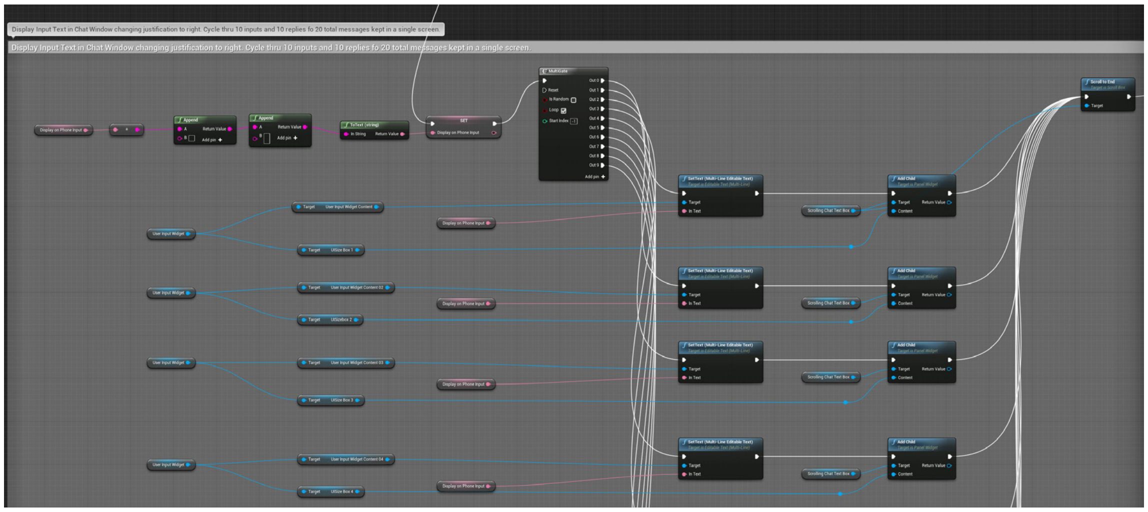
TextInputWidget

HOW THE TEXT MESSAGE PHONE INTERFACE WORKS continued.

Each of these 2 other widgets contain 10 slots. 1 for each text message for a total of 20. Once 20 is reached, the 21'st one is removed so there is only ever a total of 20 text messages on screen at one time (you can scroll the screen to see past messages out of those 20).

To increase this amount you will need to 1) add more slots to the respective widgets by duplicating (copy and pasting) the AISizeBox elements and the UISizeBox elements. Inside each is a button, and in the button is the text which the variables in the main Widget set (your input, and the systems response) - Be sure to name the AIReplyContent and the UserInputWidgetContent with appropriate versioning (11, 12, 13, etc). Once these are in place return to the main TextInputWidget and go to the 2 sections that deal with outputing data to the phone interface and add these new elements into the system at the end. Don't forget to reconnect all the nodes:


Display Input Text in Chat Window changing justification to right
Addsd a space after and a carriage return
DisplayReplyTextOnPhone
MultiGate
SetText (Multi-Line Editable Text)
Add Child
Scroll to End
Reply Text Var
AIReply Widget
Scrolling Chat Text Box


Display Input Text in Chat Window changing justification to right. Cycle thru 10 inputs and 10 replies fo 20 total messages kept in a single screen.
Display on Phone Input
Append
MultiGate
SetText (Multi-Line Editable Text)
Add Child
Scroll to End
User Input Widget
Scrolling Chat Text Box

Trouble shooting the ChatBotAI Simulator

1) Hints

The hierarchy of where your phrase lives in the scheme of the system is very important. The word “Where” is at the end of the system, instead of the beginning. Because otherwise if the user asked “Where is the mountain” - the single word Where would be picked up first and responded to, instead of the keyword mountain. They hierarchy is a level of importance to the system and helps direct the user to the eventual goal - finding things, and winning games.

2) Test Test Test

Have someone other than you use the system without your help. Take note of what they ask and how, and how the system responds, and how they react to that response. It will probably be different then you think. People type some crazy stuff in to try to trip up the system - be aware of that. When they get a clever response to their childish inputs, they are even more impressed.

3) I added a phrase within the node chain, yet its not recognizing it when you I type it in the input. I get the catch all response or a response that I wasn’t expecting.

Couple things to check. 1) Is it in the right place in the hierarchy, or is something in that sentence triggering before it can get to your phrase? 2) if you are still getting the catch all responses, chances are you have missed a connection when adding it in. Don’t forget to connect all of your **Set Reply Text Variables** to the **Display Reply Text on Phone** node so it will print on the screen.

Questions, comments or suggestions: support@creativevilla.com

Best wishes, and have fun!