Bodacious Battle Bots

TECHINICAL

MANUAL

Technical Manual

Table of Contents

1. Programming Language….......//fix page #.............................................................2

**Programming Language and Structure of Files**

Programming Language

The programming language used throughout all the skills is Javascript. Misty is capable of running only Javascript and C#, we chose to use Javascript since C# is still under development.

Structure of Files

The “root” directory is MistySkills. Inside there, there is a folder for every skill, and the name of the folder is the name of the skill.

Inside each folder of a skill, you will find at least three files. The source code, this file will have the .js extension. The JSON file, which will have the .json extension, this file holds the parameters needed for the skill, such as identifiers, variables, and instructions of how the robot should interact with that skill. Also, you will find a read.me, which gives a brief overview of the skill along with some details.

Some skills will have more than three files inside. These files are used during the skill and will only be images or audios.

Coding Practices

One of the most important coding practices involved in this project that is different than usual is the fact that if you want to use a function in a RegisterTimerEvent method, you will need to name as follow “ \_nameOfFuncion()” (underscore plus the name of your function).

Another important practice that in this project is different than others is how to declare global variables. In order to declare a global variable you will need to use misty.Set(), which takes three parameters, the first one is the name of the variable (string), the second one is the value, and the last one is if it should be kept after the skill is done. In order to access this variable, you will need to use misty.Get(), which requires only one argument, which is the name of the variable (string).

**Brief introduction to skills**

Dancing  
In this skill, Misty moves its arms and head. It also turns around and plays an audio.   
  
Detecting Objects  
Misty starts to walk while it detects objects. It plays and audio when the object is found.  
  
Text to Sound  
The robot gets a string and uses an API to speak the text in the desired language.  
  
Sound to Text  
The robot hears a phrase and uses an API to turn that spoken phrase into text in the desired language.  
  
Conversation  
Misty answers to some audio inputs based on previously coded interactions.

Translation  
The robot hears a phrase, transforms it to a string, then translates the string to another language, and then gets the string and transforms it in audio. All that using APIs.

Tic-tac-toe

Security Camera  
Misty starts streaming and it is possible to see the video streamed using a website.

Reaction  
Robot copies the arms movements that the person does, movements possible: arms up and down.

Face Recognition  
Misty recognizes faces and displays and audio each time it recognizes the person. The audio changes depending on the amount of interactions Misty had with that person.

Rock Paper Scissors  
The robot plays rock paper scissors.

Start with IP Address

This skill runs on startup. The only thing that it does is show the IP address of the robot on its screen once you tap on the robot’s chin.

**Special details of the skills code**

Dancing  
In this skill, Misty moves its arms, head. It also turns around and plays an audio.

We set Misty’s face to admiration, that we got from the Misty SDK website.

We created two functions to handle the arms movements. In these functions, we change the position of the arms.

In the head movement function, Misty turns its head left and right by us just changing the second parameter of the MoveHead() function.

In the turning function, Misty turns back and forth in a preset circle.

The most important part of this skill is handling the loops. Misty does not allow any kind of while loop inside her code. The way the robot uses loops is through registering an event to happen every X amount of time.

We used the RegisterTimerEvent to achieve that. This method needs 3 parameters: first, the name of the function (without the underscore) as a string, second, after how many milliseconds should it be triggered as an integer, and third, if it should repeat or not as a Boolean.

For this skill, there is nothing different in the JSON file from what the Misty SDK website generates.   
  
Detecting Objects  
  
  
Text to Sound  
Straight forward skill, transforms a string into an audio that Misty will play.

We have 3 functions speakTheText, \_Base64In, and \_speakTheText.

The speakTheText method takes only one parameter. This parameter is a string and should be the text that you want Misty to Speak. Inside this method we have an important setup to do in order to run this skill. We have to use JSON.stringify() to make a string out of the arguments we need to use. These arguments are passed to the Google API which then sends us the audio response.

The most important parameters under the arguments are ‘input’, which should be the only parameter used in this method, ‘languageCode’, which is the language of the input, and ‘audioEncoding’, which should always be “LINEAR16” for the rest of the code to work.

After setting up, you can then use SendExternalRequest to talk to the Google API.

The \_Base64In function is where the Google API talks to misty and saves the audio file, then plays it. This method needs to start with an underscore, because it is being called from SendExternalRequest and getting data from it.

The \_speakTheText function is only to allow the Google API to talk to misty, by registering an event that misty will be waiting before running the rest.

Our JSON file in this skill plays a very important part and have a different set up. Inside the “parameters”, there is a variable called "APIKEY\_Google" which is followed by a colon and then a string. This string is the key that you should get from google. Explanation on how to get this key is under “Installation and Requirements for This Project”.  
  
Sound to Text

Record an audio, then displays it on Misty’s screen.

There are 2 functions that we created \_voice\_record\_complete\_message and register\_voice\_record\_complete.

In order to accomplish this task, we used the built-in function CaptureSpeechGoogle. Which takes the following parameters: overwriteExisting(which should always be false), silenceTimeout(how long should Misty wait without hearing anything before it moves on), maxSpeechLength(how long should it record if the silenceTimeout does not stop it), requireKeyPhrase (if misty should wait for a “Hey, Misty” before it starts recording), captureFile(should always be true), speechRecognitionLanguage (the language that she is going to hear in language code), key (Google API key, that should be inside JSON file).

Since the Google API will return a lot of data, we need to specify what we want to use. For this, we used the register\_voice\_record\_complete function. Which add the proper return cases that we should look from the API, then register the event voice\_record\_complete\_message.

The \_voice\_record\_complete\_message is an event; therefore it needs to begin with an underscore. Inside, we just call built in functions to display the proper results.

Our JSON file in this skill plays a very important part and have a different set up. Inside the “parameters”, there is a variable called "APIKEY\_Google" which is followed by a colon and then a string. This string is the key that you should get from google. Explanation on how to get this key is under “Installation and Requirements for This Project”.  
  
  
Conversation

Translation  
This skill is an applied use of two skills defined before, Sound To Text and Text To Sound. The translation part is the middle of the skill. For understanding of the other parts used, please read the documentation for the other two skills.

The translation part uses only two functions: \_translatedData and translateText.

Our translateText function needs only one parameter, which is a string representing the text that you want to translate. Inside this function, you will need to set up the variable arguments. It is important to change the “source” and “target” based on the languages that you want to translate from and to. Then, we use the SendExternalRequest function to call the Google API, and set our last argument to the event in which the return values should be sent to, in this case \_translatedData.

The \_translatedData function will get the return value from the Google API as its only parameter. Then, it will save to a variable that can be accessed from other functions.

Our JSON file in this skill plays a very important part and have a different set up. Inside the “parameters”, there are three variables called: "APIKEY\_GoogleSTT", "APIKEY\_GoogleTTS", and "APIKEY\_GoogleTranslate” which are followed by a colon and then a string. These strings are the keys that you should get from google. Each key should be entered with the service that you want to access. Explanation on how to get this key is under “Installation and Requirements for This Project”.

Tic-tac-toe

Security Camera

Reaction  
Misty identifies a human, then it tries to do an estimative of which pose the person is, then reacts to that.

The most important part of this skill is the \_human\_pose\_estimation. All the interactions that the Robot will perform is inside this method. Everything else is setting up the robot and allowing it to a lot of math and statistics to perform the pose estimation. It is highly advised to not change anything that is not inside the \_human\_pose\_estimation method.

In this skill, the human body is defined as follow:

NOSE(0), LEFT\_EYE(1), RIGHT\_EYE(2), LEFT\_EAR(3), RIGHT\_EAR(4), LEFT\_SHOULDER(5), RIGHT\_SHOULDER(6), LEFT\_ELBOW(7), RIGHT\_ELBOW(8), LEFT\_WRIST(9), RIGHT\_WRIST(10), LEFT\_HIP(11), RIGHT\_HIP(12), LEFT\_KNEE(13), RIGHT\_KNEE(14), LEFT\_ANKLE(15), and RIGHT\_ANKLE(16).

The only parameter that the \_human\_pose\_estimation takes is a return value from an event. Then, we will store the key points from the human that misty was able to detect inside an array. From there, we try to estimate the pose based on the key points and define an action for misty based on what she saw.

In the JSON file, the only thing that needs to be changed from the generated template from the Misty SDK website is to add an argument called "SkillStorageLifetime" and set its value to "LongTerm". Since we will need to have access to global variables through the code.

Face Recognition

Rock Paper Scissors

Start with IP Address