Creating a Music and Text-Based Game Discord Bot with Go

Josue Torres-Fonseca: josuetorresfonse@u.boisestate.edu
Joseph Moore: josephmoore@u.boisestate.edu
Andres Guzman: andresguzman@u.boisestate.edu
Jared Rackley: jaredrackley@u.boisestate.edu

November 30, 2020

Abstract

A main function of Discord is the ability to create bots that can perform various functions. This functionality makes Discord more usable and allows those who enjoy using the platform to contribute to its development and growth. For this project, the Go programming language was used to make a program coined DJGopher. DJGopher is a Discord bot that is meant to be used in various servers, which allow friends/family to play text-based games such as hangman, trivia, and connect 4 while also listening to a variety of music in a voice channel.

1 Introduction

Go is very different from many other programming languages partly because it was made by Google to promote scalability. While most programming languages nowadays do not handle concurrency and multi-threaded programs well Go was built to support these features. Go also compiles very quickly and converts into binary code which helps it take up as little space as possible.

One of the primary reasons we chose to use Go was that it is a language that has a lot of potential, efficiency and speed, while at the same time being relatively easy to use. Things like a garbage collector and the declare-and-initialize construct ':=' (that automatically types newly declared and initialized variables for you) make coding in Go a lot simpler than a language like C that requires memory management and explicitly typing for each new variable that is created. This ease of use made it much easier and faster to get straight into developing our project in the limited time frame that we had.

For our project we decided to create a Discord bot that would allow users in a specific channel to use commands to both play music or play text based games in the chat. As previously mentioned, speed was an important factor as we wanted the bot to be able to give responses in the chat as quickly as possible. This would

make the bot seem more responsive and easy to use for the user base. For these reasons, we thought that this project would be an excellent endeavour since we could certainly keep on using this bot in the future for our own Discord servers and continue development on it even after the official project had concluded. Overall, the combination of the Go language and a Discord bot seemed like a great idea for us to pursue.

2 The Go Programming Language

Go supports two different types of assignment methods. It can infer the type with a ":=" assignment, but otherwise you must declare the type. The type is declared after the variable, which helps demonstrate the types in Go more clearly. Go handles naming like most other languages. By convention, Go encourages camel case. However, it does not enforce this so variables can be named anything. Go is a statically scoped language, which means everything can be bound at compile time. This decision most likely came from the influence of C.

One of the great features Go has, that is different than other languages, is how it implements arrays. Instead of implementing arrays like most other languages, Go most commonly uses slices. Slices don't store any data, but they reference an underlying array, and any actions taken on the slice affect the underlying array. The official Go tutorial describes slices as a dynamically-sized, flexible view into the elements of an array [2]. While Go does officially have arrays, they are not used very often.

The Go programming language sports a lot of features that are standard in C like pointers and structs. However, Go also takes many things from higher-level languages like Java. For example, Go has its own garbage collection system and the language has implementable interface while also being modular and having the ability to easily import packages. Go handles concurrency easily with concurrency primitives such as goroutines and channels which help concurrency problems be resolved easily [3].

There are downsides to the Go language, specifically around errors. We kept getting an error when we had an unused import or variable. This error is exactly what it sounds like, an error for when an import or variable goes unused. Our thoughts were that this was a way to optimize the code Go was importing, but it still meant that every single time we would run our code, we got an error. The team was patient as we understood the value in the Go programming.

Go has many features that would make it a desirable language for future projects. Its solutions to concurrency make it optimal for building a web server. This language also has goroutines which are like threads. These make it desirable for multi-threaded programs. These features also make Go good for large-scale projects. Furthermore, Go was made and is maintained by Google, which means it is going to be maintained reliably.

Overall, Go is a very versatile language that was made to solve the problems relating to multi-threading and capacity of current languages. It does a great

job with this, and also can be used widely for any type of project. We would most likely pick Go specifically if we were going to program a web server, a program with high volumes of inflow and outflow of data, or a program that requires multithreading.

3 Project Description

As previously mentioned our overall project idea was to create a Discord bot that could both play music for users and allow them to play simple games with each other in the chat channels. We started the process by first trying to get an actual Discord bot working in one of our testing channels. To do this, we looked up a github package made by bwmarrin called "discordgo[1]." This gave us the ability to start making interactions with the Discord API without having to get into the nitty gritty details of it so that we could focus on other critical aspects of the project.

Once we were able to get a simple bot working, we began development on the three games that we chose to include for this project: hangman, trivia, and connect four. Once we had gotten the basic functionality of these games working, we started working on the DJ (musical) portion of our bot. We had initially planned to have the DJ use YouTube to play music until we discovered that this would have violated Google's terms of service. Because of this, we decided to alter our approach by simply having the bot play pre-installed music in playlists that we organized ourselves.

With the finished product we now have a Discord bot that anyone can put on their server that allows them to interact with other friends on the channel by playing little mini-games and listening to music together through the use of simple commands and in chat interactions.

4 Program Description

The DJGopher program takes as input various commands which allow the user to play text-based games or listen to pre-installed music in a Discord voice channel. These commands are shown in Figure 1.

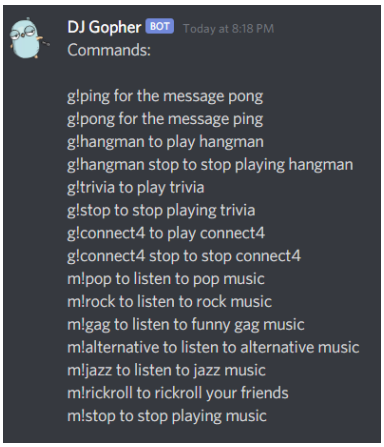

Figure 1: Input commands to use DJGopher Functionality

The games that can be played include hangman, trivia, and connect 4 and the music categories users can listen to include pop, rock, gag, alternative, and jazz. These also includes other commands such as g!ping, g!pong, m!rickroll, g!hangman stop and m!stop that contain extra features or helper functions to allow a user to perform actions such as play a new hangman game or stop music from playing.

All of the music functionality works by allowing users to type in commands such as m!pop. Once this command is used the bot then searches for the folder containing pop music and loads the audio file into a byte array. The bot then parses through every voice channel within the Discord server the bot is in. As it does this it searches for the channel which contains the user who called the bot using the m!pop command. The bot then joins this channel and plays the audio files through processing of the byte array containing the audio file data. The bot works the same for any other type of music. This is shown in Figure 2.

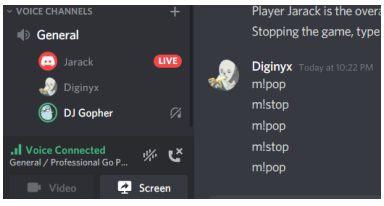

Figure 2: Music functionality

The hangman game is a text-based game which allows a user to guess a word/phrase one character at time. The number of characters is demonstrated using dashes and the number of words is demonstrated by using a space between

groups of dashes. If they guess a character correctly then the position of those characters in the phrase is shown through the replacement of certain dashes with the character guessed. If they get guess a character incorrectly then the display of the hanged man is updated to incrementally draw his body parts and then the player is shown the new hanged man display and the players current phrase with all correctly guessed characters. Overall, the players have 7 incorrect attempts until they lose the game and the hanged man is completely drawn. This is shown below in Figure 3.

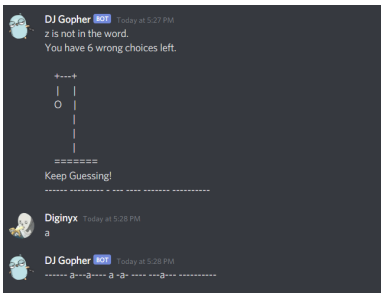

Figure 3: Hangman game

The trivia game on the other hand is very simple. This game requires at least 2 players and allows up to 6 players and then various trivia questions are asked with numerical answers. Whoever had the right answer wins, however, if nobody had the right answer then whoever had the closest answer would win and then the next question would be asked. This goes on for five rounds and whoever won the most rounds wins the game. The end of a game is shown in Figure 4.

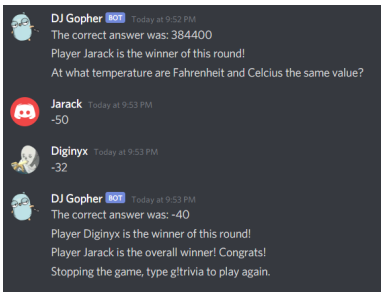

Figure 4: Trivia game

The last game connect 4 works exactly like the commonly known board game connect 4. 2 players are presented with a 6x6 board of side-by-side circles where columns 1 through 6 are labeled. Each player needs to pick a column to

drop a piece and this goes on back and forth between players until one of the players connects 4 pieces horizontally, vertically or diagonally. The player who successfully does this wins the game. A win is shown in Figure 5.

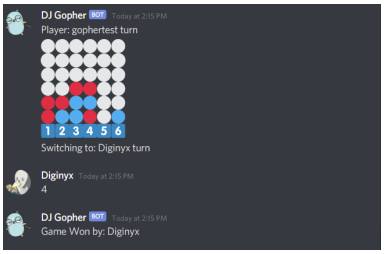

Figure 5: Connect 4 game

5 Results

In order to test the functionality of our Discord bot two methods were used. First, we tested our bot ourselves by using all of the functionality among ourselves as much as possible and by reviewing each others code. For hangman, we all played three games of it together and tried to find bugs through edge cases such as if guessing more than one character would crash the bot, as we only allowed it to guess one character at a time. We also made sure that if more than one user guessed characters that the bot would register that and would move on to the next turn, whether they were correct or wrong, through self testing hangman nothing was found. We then went on to self test trivia with four of us. The edge cases we explored include trying to opt-in more people than the user said would play or having other users who did not join the game try to play. No bugs or issues were found in trivia through this method. On connect four the same was done exploring edge cases such as what if the board was full or what if a player took a turn if it was not their turn. Through this testing we found that the game did not recognize a full board and it would allow other players to play even if they did not join the game or if they were in the game and it was not their turn. We fixed these issues in the game. The music functionality of the bot was harder to test as there were not really very many edge cases that could be tested from it. The only edge cases we could think of is trying to play other music while music is already playing or trying to play different types of music requested from two different users. We found from these edge cases that it would load the audio files and then layer them to play at the same time. This issue was fixed using flags to check if other music was playing and then required the user to stop the current music playing before they could play other music.

The second way we tested our bot was buy having friends/family use our bot and fill out a very simple survey. The survey consisted of these four questions: was it fun?, easy to use?, any bugs/issues?, and other comments? The first two

questions had the choices strongly disagree, disagree, neutral, agree and strongly agree while the last two questions allowed them to type in long answers so we may gather their suggestions/comments on the bot. We were able to get 7 different people to test the bot and do the survey while we watched them use the bot only to help them with simple Discord issues not related to the bot and/or gather information from watching them use it. 71.4% of users agreed that the bot was fun to use while 28.6% of users strongly agreed that the bot was fun to use. The same results occurred concerning if the bot was easy to use as 71.4% of users agreed that the bot was easy to use while 28.6% of users strongly agreed that the bot was fun to use. Through this evaluation and survey we were able to find several bugs. This included an issue where the connect four game would not allow users to drop a piece on the top row of the board and an issue where the bot would switch players when it should not such as if the user did try to put a piece when the column was full. These issues in connect four were fixed. In hangman, when playing with certain phrases such as "Life's too short" it would allow the user to guess all of the letters however it would not allow them to guess special characters/punctuation such as the apostrophe in "Life's too short". This issue is more complicated than expected and will be fixed in the future. In the trivia game and music functionality no issues were found. Instead, other issues not related to a certain functionality as we could see would crash the bot however some of these we were not able to identify where the issue came from. Other than bugs/issues users also gave us comments to make it more useable such as making the commands shorter, not making the commands case sensitive, and having all games reset automatically when the game ends rather than making the user do it themselves.

6 Future Work

The Discord bot is very flexible and can *GO* in many different directions such as new games, additional music support, and image scraper capabilities. Looking at the top Discord bots, there a few different popular game ideas in common. Pokemon bots as well as most RPG bots seem to fare well with Discords text based environment. There was an idea for a social RPG text-based game early on in development, however most of the time spent on that would be working on the games options and not the GO language and therefore it was scrapped. In its current state the bot can only play music that is already preinstalled. A good addition would be to integrate Spotify or Soundcloud into the Discord bot. The bot would have a queue, playlist, and other common media controls. If Youtube's terms of service change in the future, adding Youtube functionality would be great. There are many popular bots where typing in a keyword pulls up that respective image. The most common are Memebots, however we can use any images and give the users options to create an upload there own keyword image pair. Lastly, we discovered that the bot does not work well on multiple Discord servers therefore we would like to eventually fix this so the bot has its own instance on multiple servers.

References

[1] bwmarrin. *discordgo*. URL: https://github.com/bwmarrin/discordgo.

[2] Google. *A Tour of Go*. URL: https://tour.golang.org/list.

[3] Vivian Kumar. *My 5 favourite features of Go and how to use them*. URL: https://making.pusher.com/my-5-favourite-features-of-go-and-how-to-use-them/.