I'm not a bot

Introduction Enabling Text-to-Speech in Moon Base Alpha Syntax for Telling the Program What to Do Using Phonemes to Create Specific Sounds Modifying Sounds with Symbols Changing Pitch and Length with Syntax Character Limit and Cueing Up Songs Recording Game Audio and Walkthrough Process Creating Segments of An Anime Song Using Excel for Easier Typing and Macro Creation Figuring Out Pitches and Timings The Importance of Notes and Rhythm Determining Timing in Milliseconds Pitch Shifting Notes with Formulas Rough Version of the Song and Syntax Using an Audio Editor for Proper Timing Tweaking the Timing for Accurate Representation Recording and Tweaking the Song Final Tweaks and Completion Moon Base Alpha is a popular game that allows for text-to-speech functionality. Enabling this feature in the options menu will cause any text Typed in the chat to be Read out loud by the program. This article will provide a step-by-step guide on how to make Moon Base Alpha sing using text-to-speech, including the syntax for telling the program what to do, using phonemes to create specific sounds, modifying sounds with symbols, changing pitch and length with syntax, and more. 1. Introduction Moon Base Alpha is a game that offers a unique text-to-speech feature. By enabling this feature in the game's options menu, players can make the characters speak any text they Type in the chat. This opens up possibilities for creating songs, sound effects, and other audio experiences within the game. In this article, we will explore the process of making Moon Base Alpha sing using text-to-speech. 2. Enabling Text-to-Speech in Moon Base Alpha Before we dive into the process of making Moon Base Alpha sing, let's first ensure that the text-to-speech feature is enabled in the game. To do this, follow these steps: Launch Moon Base Alpha and navigate to the options menu. Look for the text-to-speech settings and make sure it is set to "enabled". Save the changes and exit the options menu. Now that the text-to-speech feature is enabled, any text You type in the chat will be read out loud by the game. 3. Syntax for Telling the Program What to Do To make Moon Base Alpha sing, we need to use a specific syntax to tell the program what to do. This syntax involves using brackets and apostrophes to encapsulate the text and provide instructions to the program. For example, if we want to make the game say "hello," we would write it as '[huh][hey][bull][oh]'. Each set of brackets represents a specific phoneme or sound. 4. Using Phonemes to Create Specific Sounds Phonemes are specific sounds that can be created using the syntax in Moon Base Alpha. These phonemes are represented within square brackets and are the building blocks of the text-to-speech system. For example, the phoneme for "hello" would be represented as '[huh][hey][bull][oh]'. By combining different phonemes, we can create a wide range of sounds and words in the game. 5. Modifying Sounds with Symbols In addition to using phonemes, we can also modify sounds further by using symbols within the syntax. For example, adding an apostrophe after a phoneme emphasizes that sound. So, if we want to emphasize the "oh" in "hello," we would write it as '[huh][hey][bull][oh']'. 6. Changing Pitch and Length with Syntax One of the exciting features of Moon Base Alpha's text-to-speech system is the ability to change the pitch and length of each spoken phoneme. By using syntax, we can modify the pitch and length to create a more dynamic and expressive audio experience. For example, adding a number after a phoneme can change its pitch, and adding a '-' or '+' symbol can decrease or increase its length. 7. Character Limit and Cueing Up Songs It is essential to note that Moon Base Alpha's chat has a character limit of 128. This means that we cannot copy and paste long songs directly into the game. However, we can cue up more songs by leaving out a closing bracket until the very end. This way, we can queue up longer segments of music by pasting them together one after another. 8. Recording Game Audio and Walkthrough Process To make Moon Base Alpha sing, we need to Record the game audio. This involves a process that requires meticulous Attention to Detail. Here is an overview of the required steps: Determine the pitches and timings of the song in text form. Copy and paste 128 characters at a time into the game. Record the game audio using a suitable recording software. Walk through the entire song repeatedly, recording each segment. Assemble the recorded segments using an audio editor, ensuring precise timing. In the following sections, we will explore each step of the process in more detail, providing tips and guidance along the way. 9. Creating Segments of An Anime Song Let's suppose we want to make Moon Base Alpha sing a segment of an anime song. To do this, we will need to break down the lyrics into segments of 128 characters, ensuring each segment has precise pitches and timings. 10. Using Excel for Easier Typing and Macro Creation Typing up all the sounds, lengths, and pitches can be challenging and time-consuming. However, using a tool like Microsoft Excel can make this process much easier. By typing the phonemes, pitches, and lengths into an Excel sheet, we can create a clear and organized structure for our text-to-speech composition. Additionally, macros can be created within Excel to automate the process further. 11. Figuring Out Pitches and Timings Determining the pitches and timings in the song is crucial for creating an accurate text-to-speech representation. By listening to the song and using tools like Reaper, we can determine the semitone values for each note. By considering the ratio of semitones, we can accurately pitch shift the notes in the text-to-speech system. 12. The Importance of Notes and Rhythm When making Moon Base Alpha sing, it is essential to consider the notes and rhythm of the original song. By counting the number of quarter notes or eighth notes in our heads, we can determine the basic rhythm and ensure that the text-to-speech representation aligns with the original song's structure. 13. Determining Timing in Milliseconds While Moon Base Alpha's text-to-speech system uses milliseconds for timing, each semitone can add a varying number of milliseconds to the beginning. By estimating the timing Based on the speed of the song, we can treat one eighth note as a specific value and use formulas in Excel to determine the timing for each segment of the song. 14. Pitch Shifting Notes with Formulas In Moon Base Alpha's text-to-speech system, the note C2 is represented as number one, and each semitone up or down follows a specific ratio. By using formulas in Excel, we can pitch shift notes accurately by calculating the required semitone adjustment for each note in the song. 15. Rough Version of the Song and Syntax After determining the pitches, timings, and creating segments in Excel, we can generate a rough version of the song using Moon Base Alpha's text-to-speech syntax. By copying and pasting the values from Excel into the game, we can hear an initial representation of the song. 16. Using an Audio Editor for Proper Timing The rough version of the song created in the previous step may have timing issues due to estimations and variations in timing between phonemes. To resolve these issues and ensure precise timing, we need to use an audio editor like Reaper. By splitting and moving audio clips, we can Align the segments and adjust timing accurately. 17. Tweaking the Timing for Accurate Representation In the audio editing process, it is crucial to split and move clips through the midsection of notes. This allows us to make small adjustments to the timing without introducing gaps or overlaps. By adding or subtracting milliseconds using time selection, we can achieve accurate timing in our text-to-speech composition. 18. Recording and Tweaking the Song After tweaking the timing in the audio editor, it is time to record the song again. Following the same process of segmenting, recording, and tweaking, we can refine the representation of the song in Moon Base Alpha. This process may require multiple iterations to ensure the best possible result. 19. Final Tweaks and Completion The final steps in making Moon Base Alpha sing involve additional tweaks and refinements. By repeating the recording and tweaking process, we can further improve the timing and quality of the text-to-speech composition. Once we achieve the desired result, our Moon Base Alpha composition is complete. Highlights Moon Base Alpha offers a unique text-to-speech feature for creating audio experiences. Enabling text-to-speech in the game opens up possibilities for making the characters sing. The process involves using a specific syntax, phonemes, and symbols to create sounds and modify them. Pitches, lengths, and timings can be determined using tools like Excel and audio editors. Recording, tweaking, and refining the song are essential steps in achieving an accurate text-to-speech representation. FAQ Q: Can I make Moon Base Alpha sing any song? A: Yes, as long as you can break down the lyrics into 128-character segments and determine the pitches and timings. Q: Are there any limitations to the text-to-speech feature in Moon Base Alpha? A: The chat has a character limit of 128, so longer songs need to be queued up by leaving out closing brackets until the end. Q: What software do I need to make Moon Base Alpha sing? A: You will need a text editor, Microsoft Excel for easier typing, an audio editor like Reaper for recording and tweaking, and the Moon Base Alpha game itself. Q: How long does it take to make Moon Base Alpha sing a song? A: The process can be time-consuming, as it involves typing, recording, and editing. The time required depends on the complexity of the song and your familiarity with the tools and techniques involved. This article is about the video game. For other uses, see Moonbase Alpha (disambiguation). 2010 video gameMoonbase AlphaDeveloper(s)Virtual HeroesArmy Game StudioPublisher(s)NASA Learning TechnologiesEngineUnreal Engine 3[1]Platform(s)Microsoft WindowsReleaseWW: July 6, 2010Genre(s)SimulationMode(s)Single-player, multiplayer Moonbase Alpha is a simulation video game based on potential Moon base programs. It was made by the Army Game Studio, developers of America's Army, and Virtual Heroes, Inc. in conjunction with NASA Learning Technologies. The game was released on July 6, 2010, as a free download on Steam.[2] At the Interservice/Industry Training, Simulation and Education Conference in 2010, the game won the top honors in the government category of the Serious Game Showcase & Challenge.[3] Moonbase Alpha remains available on Steam, and a modified version has been playable at an exhibit at the Museum of Science and Industry in Tampa, Florida since 2012.[4][5] Moonbase Alpha is a three-dimensional simulation game playable either as single-player or multi-player with up to six players.[6][7] The player controls an astronaut who must repair or replace equipment damaged by a nearby meteor impact in an outpost on the Moon within a limited amount of time.[7] The player uses a welding torch or a wrench; however, they can only carry one tool.[6] The player repairs equipment by doing mini-games; for instance, welding repairs are done by tracing circuit lines.[6] The size of the game map varies by the number of players in a server.[8] A time limit of 25 minutes is set to repair the outpost, although this time limit can be turned off.[6] Players can control a repair robot that carries a welding torch or a holding arm to make repairs in inaccessible areas.[6] Repair robots can only be controlled through a console within a range limit.[9] Players can also drive a lunar rover to travel across the map.[6] Moonbase Alpha is set in the year 2025. As a meteor impact damages an outpost near the Moon's South Pole, the player must take control of a member of the outpost's research team and repair the outpost to save the personnel on board before they asphyxiate within 25 minutes. These tasks include repairing vital components of the life support system, solar panel array and oxygen units, and can be accomplished with a wide variety of tools ranging from robotic repair units to the lunar rover. Moonbase Alpha was designed as a precursor to Astronaut: Moon, Mars and Beyond, NASA's massively multiplayer online game that was never released. The game intended to encourage interest in space exploration in young children. Because the game is meant to be a collaborative effort, the repair mission can be conducted by six players with an additional six observers. An online leaderboard is included, encouraging players to use teamwork to help repair the station faster and earn high rankings.[10] The game was designed using the Unreal Engine 3.[8] NASA announced in December 2009 that a demo of the game would be available on Steam in January 2010.[11][12] In June 2010, the game was announced for a release date in July 6, 2010.[13] Moonbase Alpha was released to mixed reception. Gameplanet gave the game a score of 6/10, feeling that it was too short and needed more diverse missions, but noted that it was still quite well-made for a free game.[9] Common Sense Media said the game "shows students practical challenges of space", but that it was held back by poor introductory tutorials.[14] Moonbase Alpha also spawned a meme surrounding the DECtalk text-to-speech functionality within its internal chat system, where users would spam certain nonsensical phrases and words into its chat rooms, parsed by the TTS system in a humorous manner.[15][16] ^ "Moonbase Alpha". NASA. Retrieved December 8, 2011. ^ "Moonbase Alpha on Steam". Steam. July 6, 2010. Archived from the original on March 5, 2021. Retrieved November 9, 2010. ^ "Virtual Heroes' Moonbase Alpha Wins Top Serious Gaming Honor at I/ITSEC". Archived from the original on April 3, 2011. Retrieved May 12, 2020. ^ "NASA Moonbase Alpha". Virtual Heroes. September 29, 2020. Archived from the original on July 14, 2021. Retrieved July 14, 2021. ^ "Explore MOSI". Museum of Science and Industry. July 1, 2016. Retrieved July 14, 2021. ^ a b c d e f Hsu, Jeremy (July 6, 2010). "Game review: Virtual lunar life on NASA's 'Moonbase'". NBC News. Archived from the original on February 4, 2023. Retrieved March 8, 2025. ^ a b Webster, Andrew (July 12, 2010). "NASA-funded game aims to make science more appealing". Ars Technica. Archived from the original on December 29, 2024. Retrieved March 8, 2025. ^ a b Pearson, Dan (July 7, 2010). "Moonbase Alpha". Eurogamer. Archived from the original on September 10, 2024. Retrieved March 8, 2025. ^ a b Cheer, Dan (July 8, 2010). "Moonbase Alpha review". Gameplanet. Archived from the original on November 16, 2018. Retrieved August 25, 2012. ^ "Moonbase Alpha". Solar System Exploration Research Virtual Institute. Retrieved July 14, 2021. ^ Bayer, Rubi (December 16, 2009). "NASA releases new information on their upcoming MMO". Massively. Archived from the original on November 8, 2010. Retrieved March 8, 2025. ^ Rossignol, Jim (December 17, 2009). "NASA MMO To Get "Demo" In January". Rock, Paper, Shotgun. Archived from the original on June 23, 2024. Retrieved March 8, 2025. ^ "NASA's Moonbase Alpha Game Release Date Announced". IGN. June 25, 2010. Archived from the original on November 21, 2018. Retrieved March 8, 2025. ^ "Moonbase Alpha Review for Teachers". Common Sense Education. March 5, 2014. Archived from the original on July 14, 2021. Retrieved July 14, 2021. ^ "NASA's latest game makes you a network admin... in space!". Geek.com. Archived from the original on June 14, 2018. Retrieved August 25, 2012. NASA's previous game release, Moonbase Alpha, [...] was perhaps best known for a text-to-speech feature in its in-game chat channel[...] ^ "Moonbase Alpha's Chat Function Is Still Funny After 14 Years". Jalopnik. July 17, 2024. Archived from the original on September 14, 2024. Retrieved September 14, 2024. Listen to this article (2 minutes) This audio file was created from a revision of this article dated 29 May 2024 (2024-05-29), and does not reflect subsequent edits.(Audio help · More spoken articles) NASA site Moonbase Alpha on Steam Retrieved from " [aa][aa][aa][aa][aa][aa][aa][aa][aa][aa] [aa][aa][aa][aa][aa][aa][aa][aa][aa][aa] [aa][aa][aa][aa][aa][aa][aa][aa][aa][aa] [naenaenaenaenaenaenaenaebaettmaenn] i dont know lol Ever want to make songs using the TTS in Moonbase Alpha? Now you can! Introduction Hi, I'm Tacticman. I've been playing Moonbase Alpha for a while now and I have a lot of experience with the text to speech. I've written some of the most popular scripts you hear as well as contributed to some. Some songs I've written include parts of Happy Birthday John Madden, all of HEYEAYEAYEAYEA, all of Still Alive from Portal, part of Imperial March from Star Wars, Taps, 2001 Space Odessey, as well as others. I've decided to reveal the secrets to how to successfully manipulate the text to speech to make whatever songs you want. Basic Steps Writing a song code in Moonbase Alpha is actually simpler than it seems. Usually, it will look something like this: [aa] We'll call this the "standard form" of song coding. This is the standard form for two reasons; the first reason is that this particular code lasts for one second. The second being that the pitch is at a middle C on a piano. If you're musically inclined that will make more sense. Here's a diagram of a piano to help demonstrate: All of the keys are labeled. Look for the C in the middle of the keyboard. That's where the number 25 will put you. I've also added numbers to each key to indicate what note you'll hit when you enter a certain number. Note: The TTS only plays a note up to number 37, which is a high C. After that, anything higher will just play a change in pitch. Now, let's change the number to 24, making the new code [aa] . You'll notice the pitch went down to a B. In music we call this a half step. Now change the code to 26. You'll notice it goes to a C sharp but not a D. Again, changing it one number only changes the pitch by a half step. If you wanted to hit a D you'd have to go up to 27, not 26. What about the first number? As mentioned before in this "standard form," this code is exactly 1 second long. The first number in ANY code indicates the length at which is it played in thousandths of seconds. Since this is 1000, it plays for 1 second. If it were 2500, it would be 2.5. If it were 500, it would be half a second, and so on. However, if you write a code that doesn't have a pitch indicator as we'll call it, it will play whatever phoneme you put for the indicated duration, but it will sound as if you just put it into the text normally. For example, trying typing [mmeh]. It's fun! tl;dr, the first number controls how long the note goes for and the second controls the pitch. Writing the code We've all seen people write elaborate songs in Moonbase Alpha. But you'll notice that they can't do it all in one chat message. Instead, they need to stretch it out over several messages. The chat can only hold 128 characters at the most. This means if you have a code longer than 128 characters, you'll have to break it into separate parts. This also means that you'll need to conserve as much space as possible to get the most characters in per message. It's a hassle to have to keep going back to wherever you're storing the codes and copying and pasting more messages. Let's try a simple code; going from standard form to [aa] . This is from middle C to E. Now, many song codes will make it look like this: [aa] [aa] There's a lot of extra space that can be cut down from that. Let's take it part by part to see what we can take out. First of all, we have to remember that every code MUST start and end in brackets. But this rule can be omitted when combining codes. I'll show you what I mean: And that's it. This method allows you to take lengthy codes and reduce them by several characters. Advanced Coding Now that we all know the basics behind how codes are written, let's look more in depth beyond the standard form. Let's say you want to do more than match the pitch of the song; you want to make the actual lyrics. You can use other text codes other than what's in the standard form with "aa". These can be hard to find, but it takes practice, trial and error, and above all, patience. However, only certain phonemes are able to be used. I CANNOT stress this enough. If an incorrect phoneme is used, you'll get the well known "Command Error in Phoneme" prompt. That being said, you can combine phonemes. I know it sounds ridiculous, but the TTS will recognize certain parts of a code and cut them up for you. Let's take an example: Let's say you wanted to say "so" in standard form. You'd probably try this: [so] You'll get the initial S sound but nothing else. This is MBA trying to break up the code into segments. The only problem with this code is the O part. Because there are several different pronounciations of the letter O, simply having the letter there isn't specific enough for Moonbase Alpha to make a sound out of it. You'll have to add a modifier to it. Modifiers are suffixes or prefixes added to a phoneme that specify its sound. In this case, in order to produce an "Oh" sound, we need to add a W. Try this code: [sow] It works! Now, what about adding sounds at the end of that? What if we wanted to say "Soul" instead? As we said before, the code MUST be contained within the brackets. You can put an l at the end of the code outside the brackets, but the TTS will end up literally saying "So-el" instead of "soul." So to do this, we must put it inside the brackets. Try this: [sowll] There, that worked. However, the L sound doesn't always work. Moonbase Alpha seems to have a problem with certain sounds such as L, H, J, and ironically, "uh." We'll get more into that later. Advanced Coding cont. As far as the problem phonemes go, there are a few easy fixes. You may have to add a modifier to it; if you wanted a "HA" sound, you couldn't just put "ha", you'd have to put "hxaa". Another one is adding a short time modifier to it. This generally isn't an issue but it can turn up. A simple fix is adding behind the phoneme to tell Moonbase Alpha to play that sound longer than the other parts of the code. You may find that you'll need to increase that number. Ok, what about connecting codes together that have advanced phonemes? Let's make a simple sentence out of the text to speech: I want cake. Start with the first sound. Remember, you may have to mispell things in order to use the working phonemes in the game. [ay] Next? [waant] and finally: [kayk] This might get a little confusing. You'll notice that in the second line BEFORE the time indicator, TWO LETTERS ARE REQUIRED TO WORK THE PHONEME. But for a suffix, ONLY ONE IS NEEDED. The second line is an exception to this because it is producing two separate sounds, the "n" and the "t" You'll see that in the third line, the K in the "kay" part also only has one letter. Prefixes and suffixes generally only require one letter to dictate a preceding/following sound. But as always, there are exceptions. THe biggest exceptions are the H and J sounds. Now let's put it together using the same method as before. [ay] [waant] [kayk] [ay waant kayk] [aywaantkayk] You'll notice that the "nt" and "kay" come in contact, but that is ok. Moonbase Alpha will sort it out; if it doesn't, simply add a time modifier after the suffix of the preceding code. It would look like this: [aywaantkayk] Another important thing to note: If you want to make a pause, add an underscore in place of a phoneme. This will make a silence for however long the time indicates. List of Known Phonemes + Codes Only certain phonemes will work in Moonbase Alpha. Here's a list of the most popular ones without prefixes: aa (ah sound) ah (ah sound) ae (ae as in "Apple", "Madden") ay (Eh) ao ("ohaa") aw (ahoh, will change depending on whatever follows) ar (arm, army) ey (ay as in Hey, bay) eh (eh as in Meh) iy (ee as in Bee) ih (hitch, myth) ir (ear) ix (same as ih) ow (Oh, boat) uh (uh) uw (You, goo) Special cases: jh – May be needed to make a "ja" sounded like John or Joy hx – Used to make a "Ha" sound as in Happy or Harry Add any other phonemes to the comments if you have them. Well that about sums it up. So, to sum it all up: 1: Everything has to be in the brackets 2: Everything has to be under 128 characters 3: Only use certain phonemes, and some prefixes Comment or message me with any questions you have. Enjoy some songs I wrote! Still Alive (Portal) [thihsswaassahtrayahmmff] [aymeykkeyngahnowthiyrr_hxuwgsuhkksehss] [ihtshxahrdtowowversteytmaysaedihsfae] [shaan_aepehchersayehns_weydowwaat] [weymahstbeykahzweykaenn] [fowrthaagahdaavahllaavahss] [ehksehpthaawaanshxowardehdd] [bahthersnowsehnskrayihnowverehvriymihs] [teykweyljhahstkiypawntrayihntihllwiyrahn] [awtahfkeykaendthaasayehnsgehtsdahnnaend] [yxuwmeykahniytgahnfowrthaapiypihllhxow] [arstihllahllayvv] HEYEAYEAYEAYEAYEA [sowaykkrayssuhmmttaymmswwehnnaymmllayihnn] [ihnnbehehddguhsstuwgehttowttwahttsih] [mahhxaeddaeddaymmffiyllihnnpehkkuwllyur] [sowaywweykkihnnthuhmornnihnaennddaysstehpp] [awttssahayddaendaytteykkuhddiyppbrehthae] [ndgehttrihllhxayaennaysay_aettehttawppuh] [ffmaalluhnngswahssgowihngawnn] [aenndayssae_hxeheyeheyeheyey] [hxeheyeheyeh_ayssehhxeheh] [wwahttsgowihnnawnn] Windows startup [dahdahdahdahdahdah] Star Spangled Banner [owowseykaenyuwsiy] [baythaadawnsurlliyllaytt] [waattsowprawdlliywiyhxeyll] [aettahtwayllaytsllaestgliyymmihng] [hxuwsbrawddstraypsaendbrayttstahrs] [thruwthaaperihllahsfaytt] [owverraempartswiywaach] [wersowgaellehntlliystriymmihng] [aendthaaraakehtsrehdglleyrr_thaabaamsberst] [tihngihner_geyvpruwffthruwthaanaytt] [thaetarfllaeggwaasstihllther] [owseydaosthaeaettstarspeynguhlldbaenner] [eryxehehttweyeyvv_forthaallaeaendawvv] [thaafriy_aendthaahxaomaavthaabreyvv] Indiana Jones Theme [dahdahdahdahdahdahdah] [dahdahdahdahdahdahdah] [dahdahdahdahdahdahdah]