MirrorOS **BETA**

Developer Information

Last Updated: February 3rd, 2021

It is important to note that all features discussed may not be available.

Table Of Contents

Using the Guide

Welcome to MirrorOS! This guide is designed to help you get your program up and running on the MirrorOS platform. Don't worry; getting your program able to run on MirrorOS is much easier than anyone would expect. That's the beauty of MirrorOS; It's made for developers. To start, you need to know how to use the guide.

1. Important Information will be in red. Unimportant Information and filler words will be in black or light gray. Headings will be in sky blue. Steps or subheadings will be in forest green.
2. At the start of every section, there will be a brief synopsis of what the section covers, as shown above. That way you won't have to search extensively for what you need.
3. This guide is always in steps. It is important to follow the steps in order. Otherwise, you may end up with a program that doesn't work.
4. There may be footnotes[1] next to words, clarifying what these words mean, or adding information for reference. If you have a question, it's a good idea to check them. Footnotes are not always definitions, they may be tips and tricks for problems that occur often.

That's it! Hope all goes well on your way to a smooth visual interface.

[1] foot·note
/ˈfo͝otˌnōt/
(n) An ancillary piece of information printed at the bottom of a page.

Programming your Code

Let's get started! In this section, you'll learn how to code for MirrorOS, and what prerequisites are required for your code to work on MirrorOS.

1. Coding your application

To start, your program should be in an individual project, not in MirrorOS. We'll merge the two programs later. Then, make a copy of MirrorOS, which will come in handy later. After you do that, feel free to code whatever you want. It could be a program, a song, or a cute little animation or two. MirrorOS doesn't care, as long as your program meets the requirements below.

2. Requisites

To take advantage of the interface we offer, you need two blank spots in the lower right-hand corner of the screen, and the upper right-hand corner of the screen.[2] (Image 1) The only thing that should be in this area is your stage; this allows room for the Home Button and the Exit Button. The rest of the stage is for you to use. Your stages should be individual sprites; do not use the stage interface in scratch. This will make your program easier to integrate into MirrorOS.

[2] These areas should be square, right up to the corner of the screen, and measure 25 by 25 units.

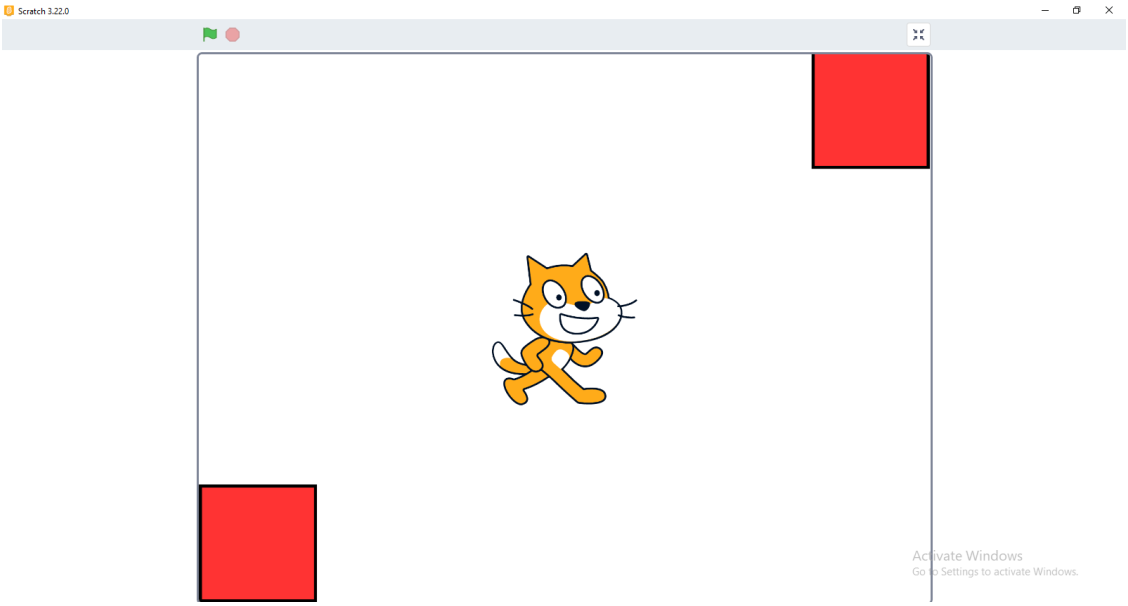

Image 1: The size of the areas that should be blank for MirrorOS

3. Notifications

If your program uses the MirrorOS notification system, you should create a variable called “Alert” with the text of your notification inside the variable. When you would like to show the alert, have your program send a broadcast called “Notify”. The way your code should be for the notification is shown below. (See Help; For Developers; Notification System) (Image 2)

Image 2: Code for Notifications

Pre-Integration

In this section, you'll learn what you need to do to integrate your program into MirrorOS when it's done. Pay attention! If you don't follow these instructions, you may have to start over.

When your program is finished, there are a few things you need to do before you can start the integration process.

1. Create an icon

Much like a youtube thumbnail, this is the first look at your program a user will get. Think of the little images on your computer's desktop, that's the style you're looking for. (Image 3) These icons should be simplistic, and easy to tell what they represent. Directly next to the icon, there should be white text no longer than 16 characters and should be sized appropriately, (see Image 3 for an example) using the default text font. Create this icon in scratch's sprite creator.

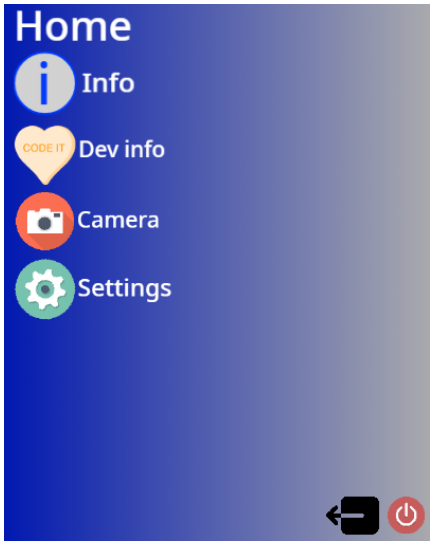

Image 3: Home Menu Icons

2. Systems Check

This is your last check of your code before you begin the integration process. Once you start the integration process, we recommend not editing your code until you complete integration,[3] as it may cause unnecessary confusion. Make sure everything is working properly, and that you have all the bugs worked out. Integration may cause undesired operation of your code if your code isn't completed, or you've only partially completed integration. After that, you're ready to go!

[3] Advanced users may do what we call an "integrated code," and code the project inside MirrorOS. We don't recommend doing this if you are coding your first project for MirrorOS or are unfamiliar with how scratch works, as there is a lot going on in MirrorOS and it may get confusing. However, a "integrated code" may give the user more control over their project, and gets rid of the nasty integration step.

Integration

In this section, you'll learn how to integrate your program into MirrorOS when it's done. This is the harder part, so read carefully.

1. Choose where to integrate

Before you integrate your program into MirrorOS, you need to select a version[4] of MirrorOS. There are many versions, one version you can use is a Modded Version, which is a version other developers have edited for their own purposes.[5] Another is a Vanilla Version, and that's just basic MirrorOS, without any pre-installed programs, that comes directly from SeadomGames Studios and is not affected by any other developer. These versions may load faster, so if you want to integrate multiple programs into MirrorOS that's the way to go. A Home Version of MirrorOS has some programs that you see in a normal operating system, such as a camera, text editor, calculator, and more. A Home Office Version of MirrorOS has many programs that are designed for work. This version takes longer to load, so we suggest you only use it if your program builds off of other programs in the Home Office Version, or you want your program to be in a Home Office Version.

2. Begin Integration

To start, open your project. Then open your backpack, located at the lower bottom left of your screen. Drag and drop each sprite in your project into your backpack, making sure each stage is a different costume in a sprite named "c/stages/yourprogramname/yourstagename." Sprites should be renamed to "c/sprites/yourprogramname/yourspritename." Then, open the MirrorOS project you have selected, click "remix" and then open your

[4] You can find the version title and the version number in the title of the MirrorOS project.
[5] Modded Versions of MirrorOS are not supported or maintained by Seadom Games and may cause undesired operation of your program.

backpack. In the MirrorOS project, open the HOME menu. Add your icon sprite to the program, then add the code below to the sprite: (Image 4) After that, drag your icon to a free space in the HOME menu.[6] You will have to disable HOMEHIDE; see Help; For Developers; Disabling HOMEHIDE. Your app icon may not use the desktop space, that is for system approved apps only.[7] Then, enable HOMEHIDE, close the home menu, and add your remaining sprites to the program.

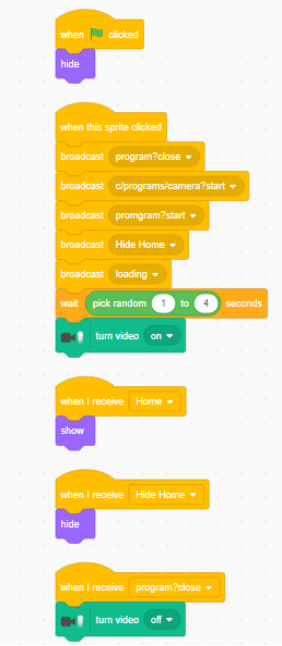

Image 4: This is an example from our camera app icon. While the program itself does not run in the icon, the icon does send out a broadcast, signaling other sprites that it's time to start.

[6] You will have to enable spacing view to properly space your icon; see Help; For Developers; Spacing View.

[7] Want your app to be system approved? Head over to our GIthub: https://github.com/SeadomGames/MirrorOS

3. Code Integration

For your application to work in MirrorOS, you will need to replace all “When Flag Clicked” blocks with a “When I receive” broadcast block with the start-up broadcast located in your MirrorOS icon. All broadcasts used by your program must be renamed to c/programs/yourprogramname?broadcastname to avoid accidentally activating another program's broadcast.[8] Your program cannot have any “Stop All” blocks, instead, replace them with three blocks: the first, a broadcast block broadcasting program?close,[7] the second, a “Hide” block, a “Stop All Scripts In this Sprite” block. Repeat with all sprites, except replace the broadcast block with a “When I receive program?close” block. All variables need to be renamed to c/{yourprogramname}yourvariablename” to avoid changing another program’s variable, same with lists.

4. Finished!

When you have completed the above steps, you’re done with integration. To make sure your program is integrated properly, test it thoroughly, making sure that all your broadcasts and sprites work as intended. Make sure to monitor major updates of MirrorOS that may impact your code and fix issues in your code that are caused by updates.[9] Enjoy premium interfaces and program stacking, and watch your code grow with the community.

[8] Replace yourprogramname with your program name, and broadcastname with your broadcast name.
[9] Updates can be tracked on our scratch page, or our github. https://github.com/SeadomGames/MirrorOS