Create Your First Civil & Safe Game on Roblox

Teacher Lesson Plans

Date created: Mar 20, 2025
Date updated: May 5, 2025
Author: Roblox Corporation
Intended audience: Primary, secondary, and informal educators
Approximate instructional time: 500 minutes
Roblox Studio version: 0.665.0.6650685

Introduction and Context

Game-based learning research finds that people learn more effectively when allowed to deepen their knowledge through hands-on experiences, problem-solving, and collaboration.

Education has been an integral part of our DNA since Roblox was founded, and we are excited to offer a growing number of educational games (https://www.roblox.com/charts#/sortName/Learn_And_Explore) and limitless creation opportunities on Roblox.

Creating games on Roblox is a powerful way to learn coding and design, but it also involves interacting with a global community. It is essential that students learn not only *how* to build games but also *how* to be safe, civil, and responsible digital citizens.

The following teacher lesson plans ("modules") help to guide teachers through the process of teaching their students how to create their first game on Roblox. When students create on Roblox, they can demonstrate their understanding of academic content, express themselves, explore their creativity, and collaborate and build with friends. The modules integrate key aspects of online safety, civility, privacy, and responsible creation throughout the game development process, drawing on resources from Roblox and online safety organizations.

Unit Description and Required Resources

This ten-module course, with more than 400 minutes of content and hands-on activities, concentrates on transferable game design and engineering skills using Roblox Studio and the coding language "Lua". The first six modules introduce students to the basics of Roblox Studio and introductory coding principles as well safety and civility principles necessary for creating games and experiences that respect the Roblox community. The last four modules are designed as group projects that mimic real-world working environments. Throughout the course, there will be tie-ins to various careers and how the demonstrated skills are used in real life.

Two additional stand-alone extension lessons are included for students who progress through content faster than others.

International Society of Technology in Education (ISTE) standards (link to ISTE student standards): Innovative Designer 4a, 4c, 4d, Creative Communicator 6b, 6d

Unit objectives	• Manipulate 3D parts to create an obstacle course. • Design and playtest an experience, focusing on a fun user experience. • Create scripts using variables and loops for in-game visual effects.
Skills and concepts	• Variable - A placeholder for information in code. • String - A variable that can store whole sentences, written in quotations. • Loop - A set of code that repeats until told otherwise.
Prep	• Make sure each student has a Roblox account and knows their login information. • Print out handouts and prep presentations.
Materials	• Windows or Mac Computer with Roblox Studio installed • Handout: Roblox Studio Cheatsheet - Includes common commands and hotkeys • Handout: Intro to Coding Cheatsheet - Common coding patterns • Handout: Project Feedback - Worksheet for peer feedback

Additional Unit Resources

- Technical Quick Start Guide - Minimum specs and account creation.
- Safety and Civility Resources - The Roblox Safety and Civility website has an abundance of resources to help parents, educators, and students understand how to promote and maintain a safe and civil environment on Roblox.
- Educator Onboarding - This site is designed to onboard teachers before they teach Roblox experience creation in class. While it is not a prerequisite to this Unit, the activities in Educator Onboarding provide ample context on experience creation and the Roblox platform.

Module 1: Introduction to Roblox Studio and Account Security (45 Minutes)

Description

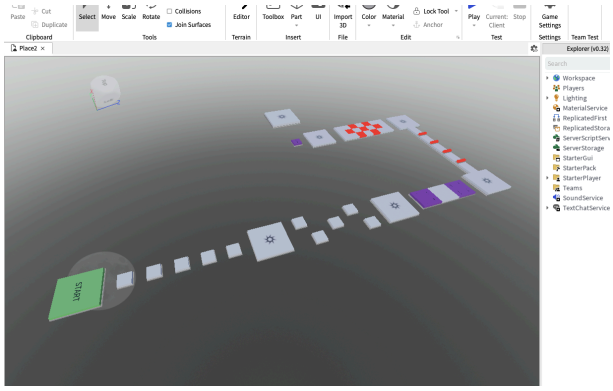
Get students familiar with Roblox and Roblox Studio (“Studio”) by having them play an onboarding game which teaches basic camera controls. Have students modify an experience by template using Studio's artificial intelligence (AI) Assistant.

Objectives

- Open the obby template
- Learn movement controls
- Use Assistant to create a “GlitterPart”
- Save and update a project
- Understand why learning to code is important

Optional Resources

- Studio UI Reference
- Studio Hotkeys



The Classic Obby Template

Introduction (5 minutes)

1. Introduce the course
    a. Ask students if they know what Roblox is:
        i. Roblox is the ultimate virtual universe that lets you create, share experiences with friends, and be anything you can imagine.
        ii. Or, Roblox is an online gaming platform and creation system where users can create, share, and play games and virtual experiences created by other users.
    b. Ask students if anyone knows what an obby is?
        i. A: “Obby” is short for “obstacle course”. Obbies are jumping puzzles where you must figure out how to jump from Bridge to Bridge without falling. They're one of the most popular types of games on Roblox
    c. Ask students if they know what Roblox Studio is:
        i. A: Roblox Studio is the creation tool used by all experiences on the Bridge. It is available for free so everyone can create and share new Roblox experiences.

d. Students will be coding and designing their own obby (obstacle course) that others can play using Roblox Studio.

2. Introduce Lesson
    a. This first lesson will show you an example of an obby
    b. For students who have never used Roblox before, this is also a chance to learn how the player controls work.
    c. At the end of the lesson, you will use Studio's built in AI to make changes to the example obby, you will also see why you still need to learn about how code works even with AI.

Guided Practice (35 minutes)

Logging In and Opening Roblox Studio (5 Minutes)

To use Roblox Studio, you will need a free Roblox account. If you do not have an account, create one at roblox.com. For tips on how to secure your account, see Keep Your Account Safe. Here are some basic account security tips:

- Strong Passwords: Emphasize creating unique, complex passwords for Roblox that are different from passwords used on other sites. Discuss why reusing passwords is risky.
- Email/Phone Verification: Explain the importance of adding and verifying a parent's or guardian's email address or phone number to the account for password recovery and security notifications.
- 2-Step Verification (2SV): Strongly recommend enabling 2SV using an authenticator app, email, or security keys as an extra layer of protection.
- Recognizing Scams: Educate students about common scams, such as offers for free Robux, requests for login information, or links to unfamiliar websites. Stress that Roblox employees will *never* ask for passwords or personal information. Teach them to identify legitimate Roblox sites and communication channels.
    - Roblox employees will never ask for your password — Report anyone who asks using the Report Abuse feature.
    - there is no such thing as free Robux — Never trust players or sites who say they have a way to get free Robux!
    - Check out this Roblox game to have your students learn more about online safety and civility: Google's Be Internet Awesome World (https://www.roblox.com/games/17756790122/Google-Be-Internet-Awesome-World)

- Logging Out: Advise students to log out of their accounts, especially when using shared computers.

System Requirements

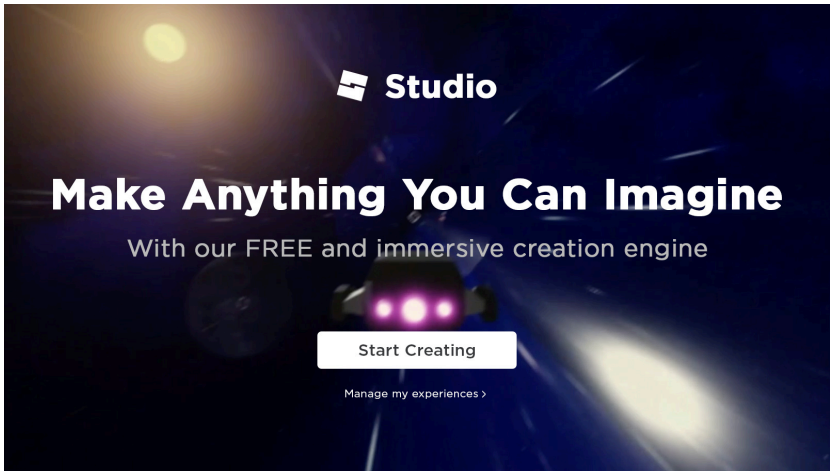
- A PC or Mac
- Internet Connection
- 2 button mouse

Installing Studio

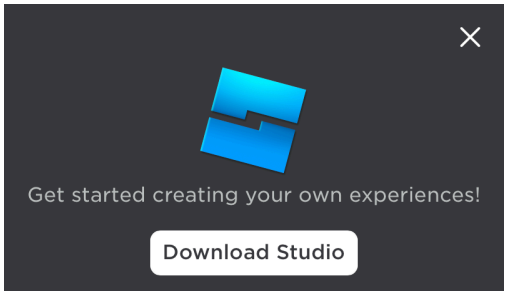
Before beginning, make sure you have Roblox Studio installed on your Mac or PC.

To install Studio:

1. Navigate to https://roblox.com/create.
2. Click the Start Creating button. A pop-up dialog displays.

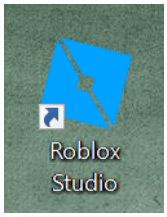


3. Click the Download Studio button.



4. Find the Studio installer in your browser's download history and double-click the file.

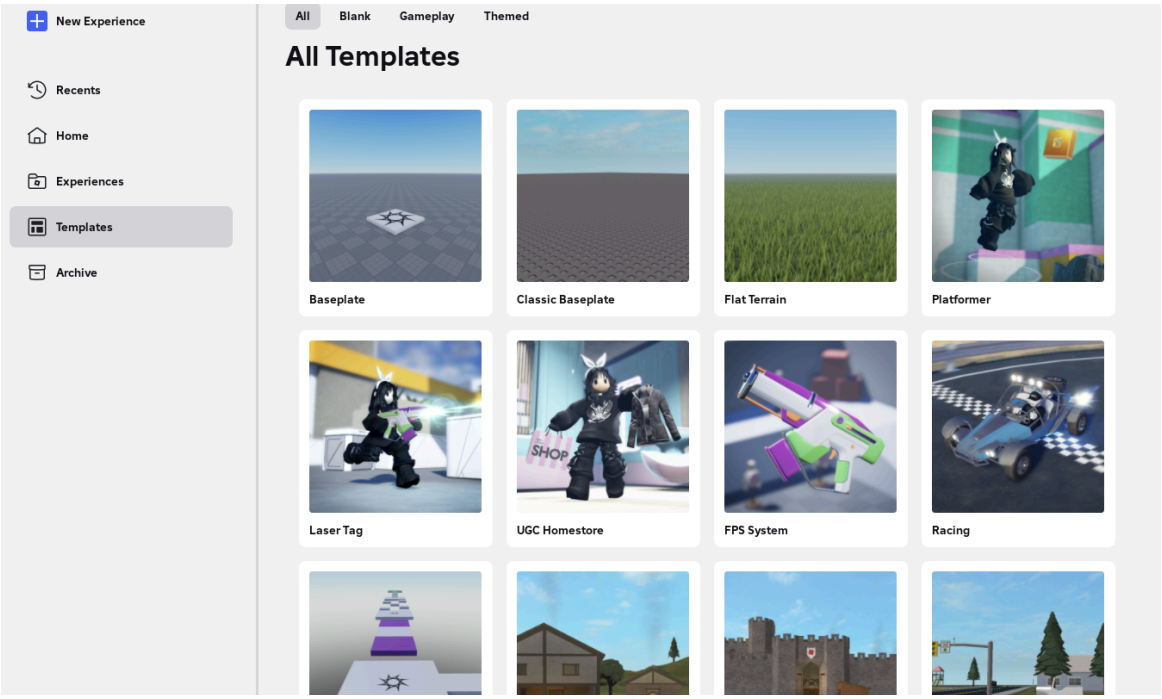
- On Windows, the file is RobloxStudio.exe.
- On macOS, the file is RobloxStudio.dmg.



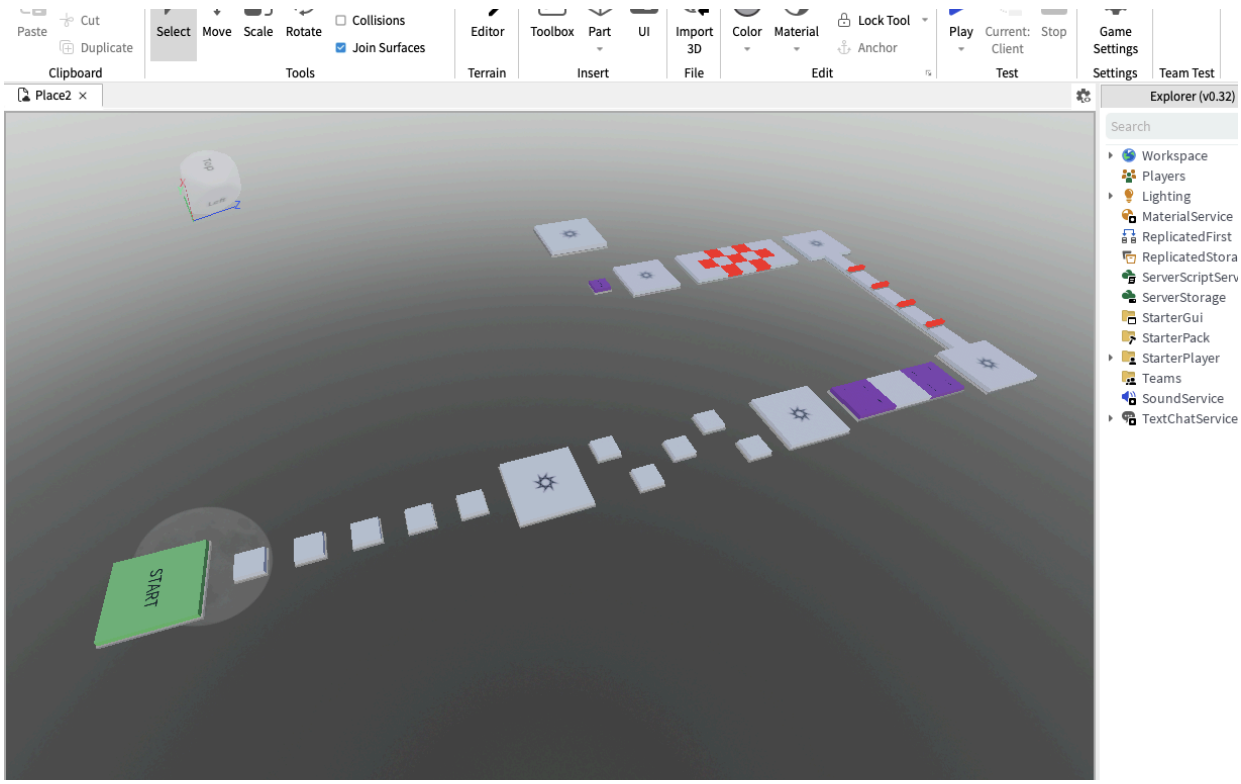
Opening the Classic Obby Templates (10 Minutes)

Templates are pre-built projects you can use as a starting place for your own experiences. you will open the Obby template and learn how to control your camera and avatar.

1. Click **Templates**.
2. Scroll down and click on the **Classic Obby** template



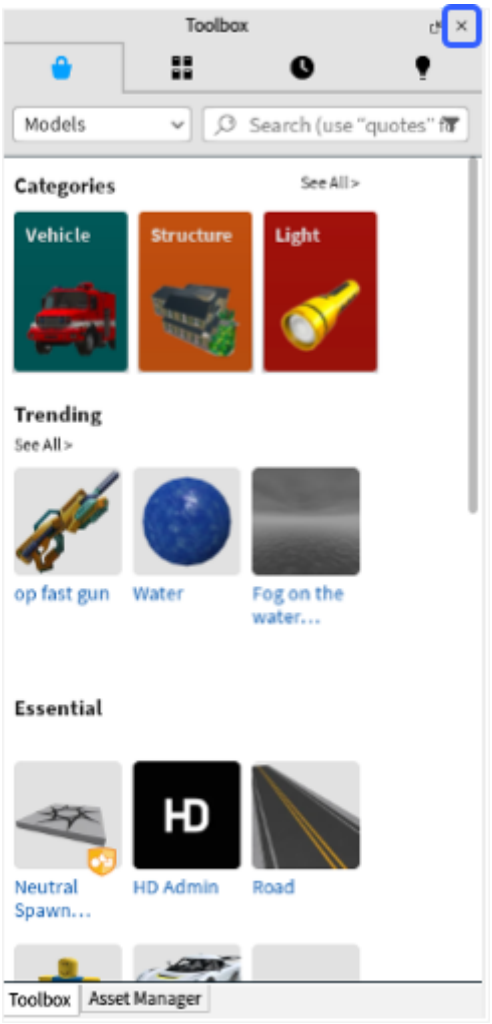
When Roblox Studio opens, it will look like this.



Customize Your Workspace

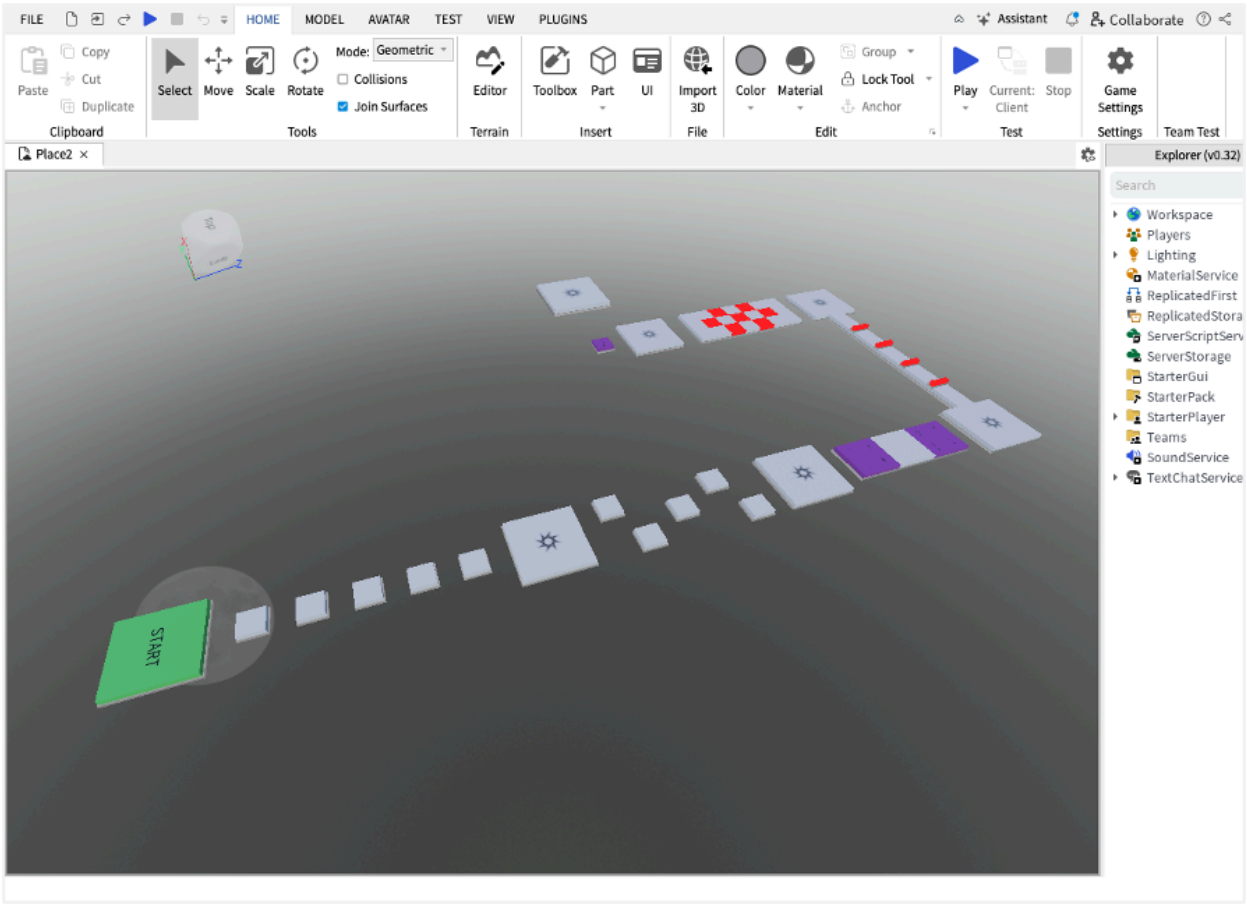
Give yourself more room to see by closing out everything on the left hand side. You can reopen those windows later.

1. Click the **X** on the right-side of the window.



Tip: Reopening Windows
If you want to reopen a Window you closed, go to the **VIEW** tab and click on the name of the window you want to open.

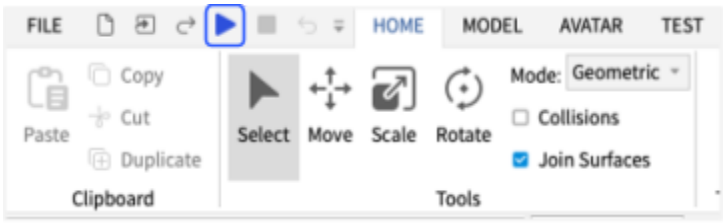
Studio should now look like this:



Playing the Classic Obby Template (2-3 minutes)

Lets see what this template Obby looks like. For those who have never played Roblox before, this is also a chance to learn how to control your character.

- Press the **Play** button in the top-left corner.



Teaching Tip
Let students play 2-3 minutes. This will give your students an idea of what they are working towards without having to go online. Let students know from the start how much time they have and give a one minute notice before time's up.

In-game Controls

Practice using the following controls to move within the Obby.

- Use **WASD** or the arrow keys to move your character.
- Use **Spacebar** to jump.
- Hold the **right mouse-button** to look around.

Stop Playtesting

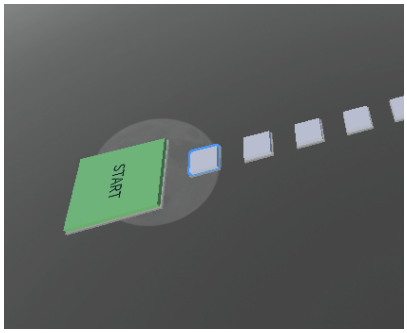
- To **stop** playtesting, press the red stop button.

Using Assistant to Make Changes (10 Minutes)

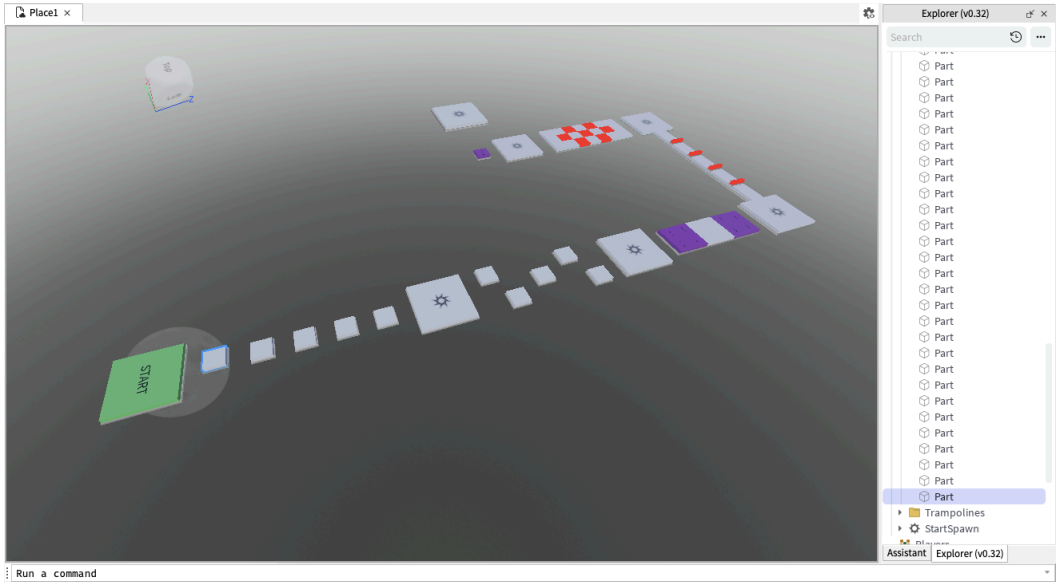
Roblox Studio has a built-in AI (artificial intelligence) named “Assistant” that can help you create and code your game. Think of Assistant as a friend that is eager to be helpful, but does not always understand what you want. This section will show you how to make requests using Assistant and how learning to code makes you better at using Assistant.

For your first request, you will make a part sparkle when a player jumps on it. Start by renaming a part to make it easier for Assistant to work with.

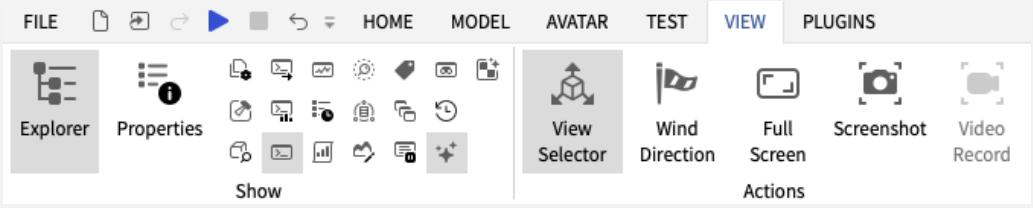
1. Click the part closest to the green starting point. It will be highlighted in blue when selected.



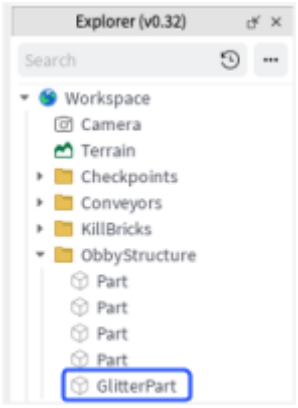
2. On the right side of the screen, called “Explorer”, you will see the part you selected highlighted in blue.



If you do not see Explorer, go to the **VIEW** tab and click on the name of the window you want to open. In this case click “Explorer.”



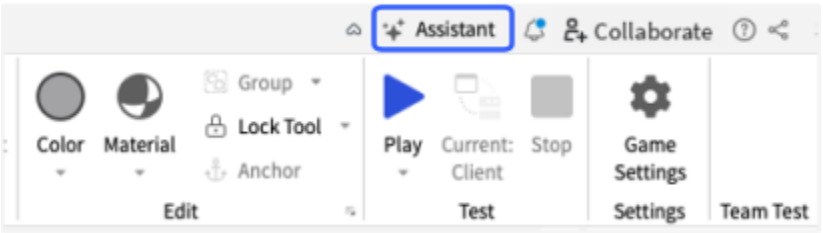
3. Right-click the highlighted part and select **Rename**.
4. Type **GlitterPart** and press **Enter**.



Opening Assistant

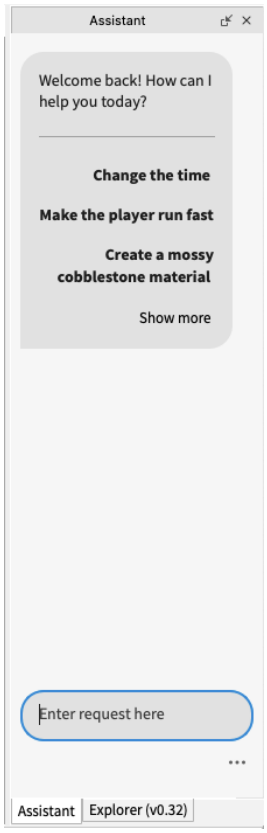
To access Assistant from Studio:

1. Click the **Assistant** button in the top-right corner.



Assistant will open bottom right. When first opened, it can be narrow and hard to see everything. Drag the edge of the window to make it a little larger.

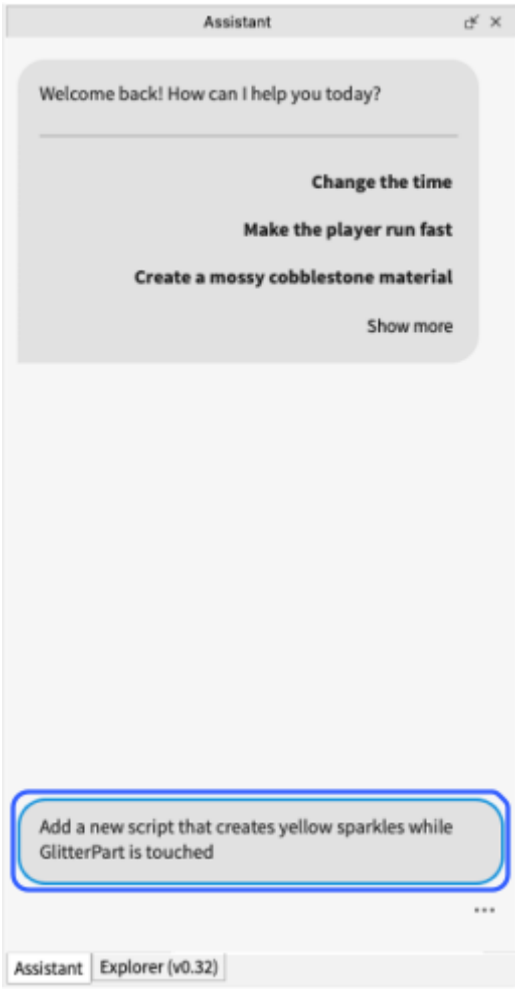
Before



After

2. Where it says "Enter request here", type: *Add a new script that creates yellow sparkles while GlitterPart is touched*.

   Warning:You can use a different color, but using different words might change the results.

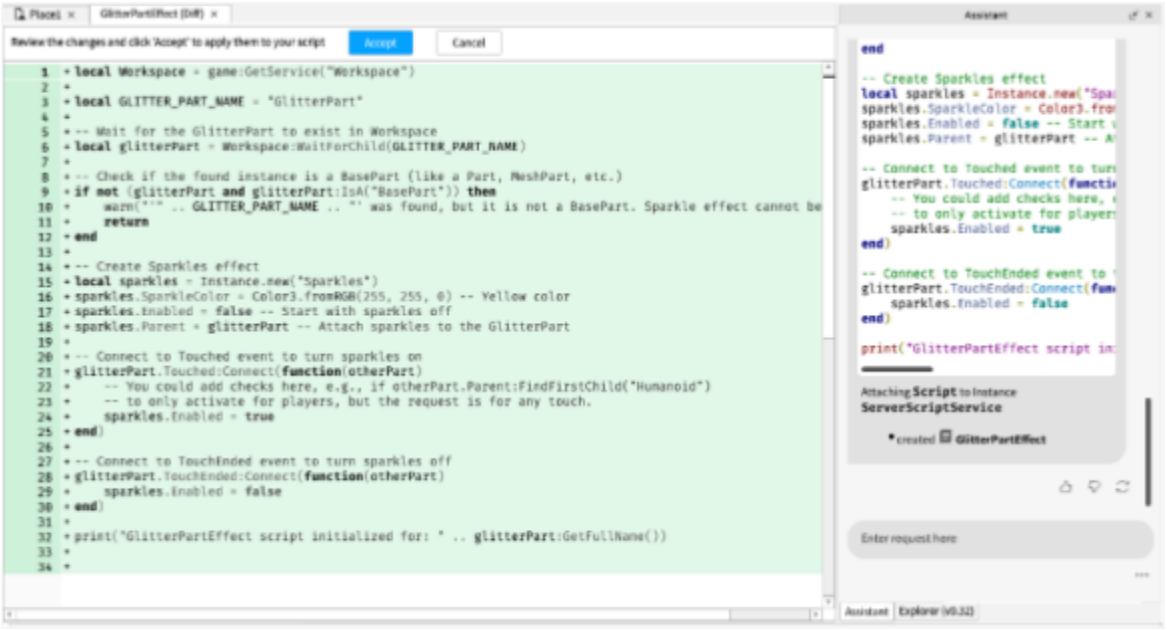


Teaching Note
This Assistant request is written in English to produce a specific and reliable result. Slight changes in grammar and word choice may cause different outcomes.

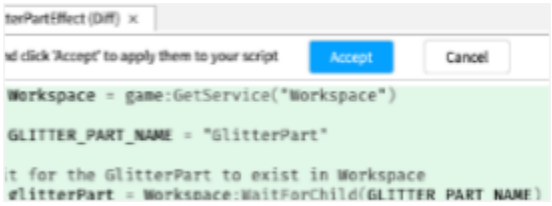
When using Studio in other languages, carefully test phrases to make sure they work as intended.

3. Click **Enter**. Assistant will run the request, and a window with the suggested code similar to below should appear.

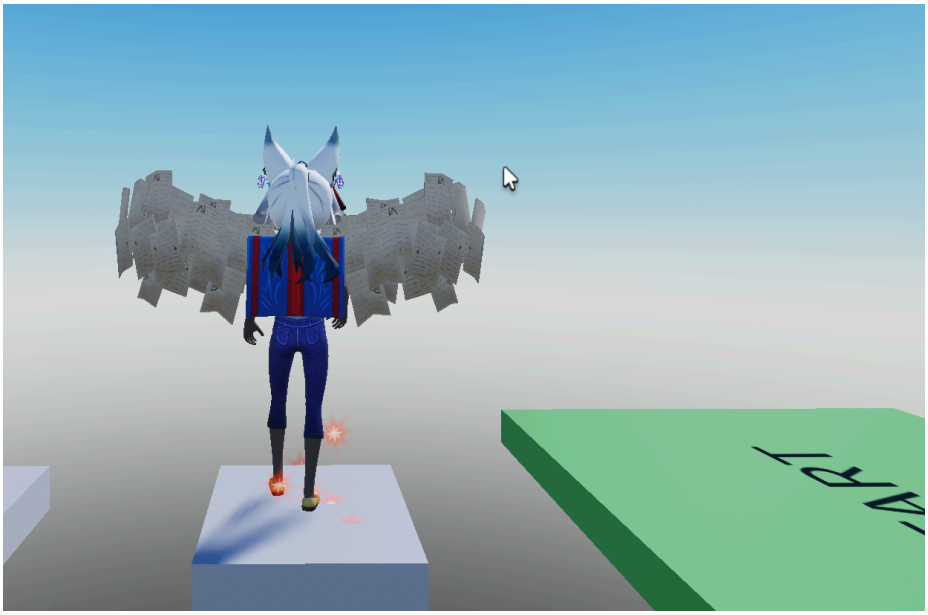
   Assistant is constantly learning, so it might not always produce the same results for the exact same request.



4. As you learn more about coding, you can better judge if the Assistant produced code that works as you intended.
   - For now, click **Accept**.



5. Playtest the game to check the results. Yellow sparkles should appear when you jump on GlitterPart.



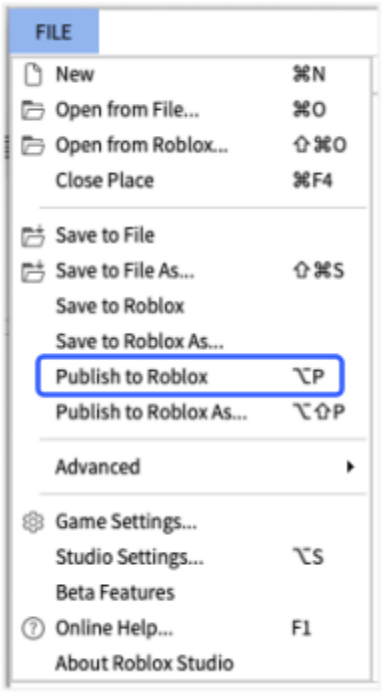
6. Remember to stop your playtest.

Save Your Work (2 minutes)

Before moving to the next section, take a moment to save the project. Publishing saves your game to your Roblox account, but it does not make it public yet. Once published, a project can be edited from any computer. Note: “Save to File” saves your project to your local computer. This is good for backups.

It is a good idea to publish every ten minutes while you are working or after a big change.

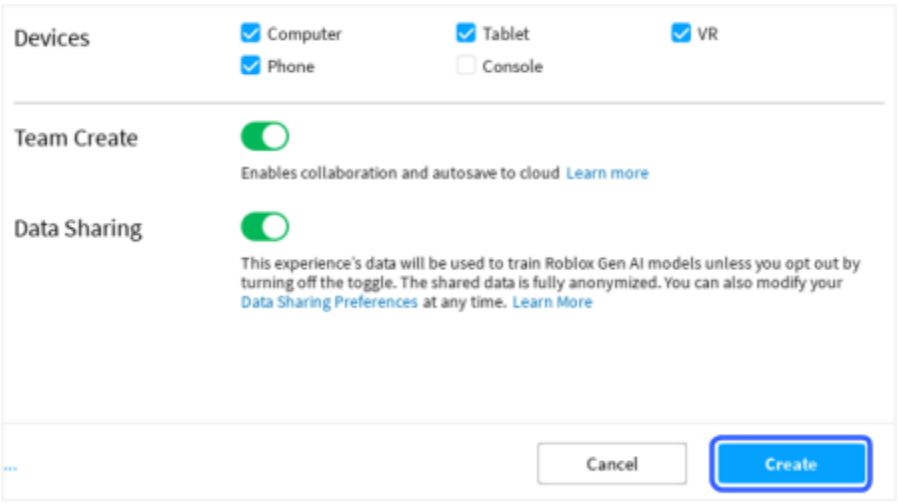
1. Select **File** (top left of Studio) and click **Publish to Roblox** to open the publishing window.



If the Publish button is grayed out, make sure you have stopped the playtest.

2. Enter a **name** and an optional **description**.

3. When happy with the name and description of your project (you can always go back and change it), click **Create**.



Next time you want to save your work, go to **File → Publish to Roblox** or use the hotkey (Alt+P).

Getting Better at Working with Assistant (5 Minutes)

There is more than one right way to make any particular request, but every word counts. If you make a request and do not get the results you want, rephrase the request and try again.

Below are examples of how slightly different word choices can produce different results.

Original: Add a new script that creates yellow sparkles while GlitterPart is touched.

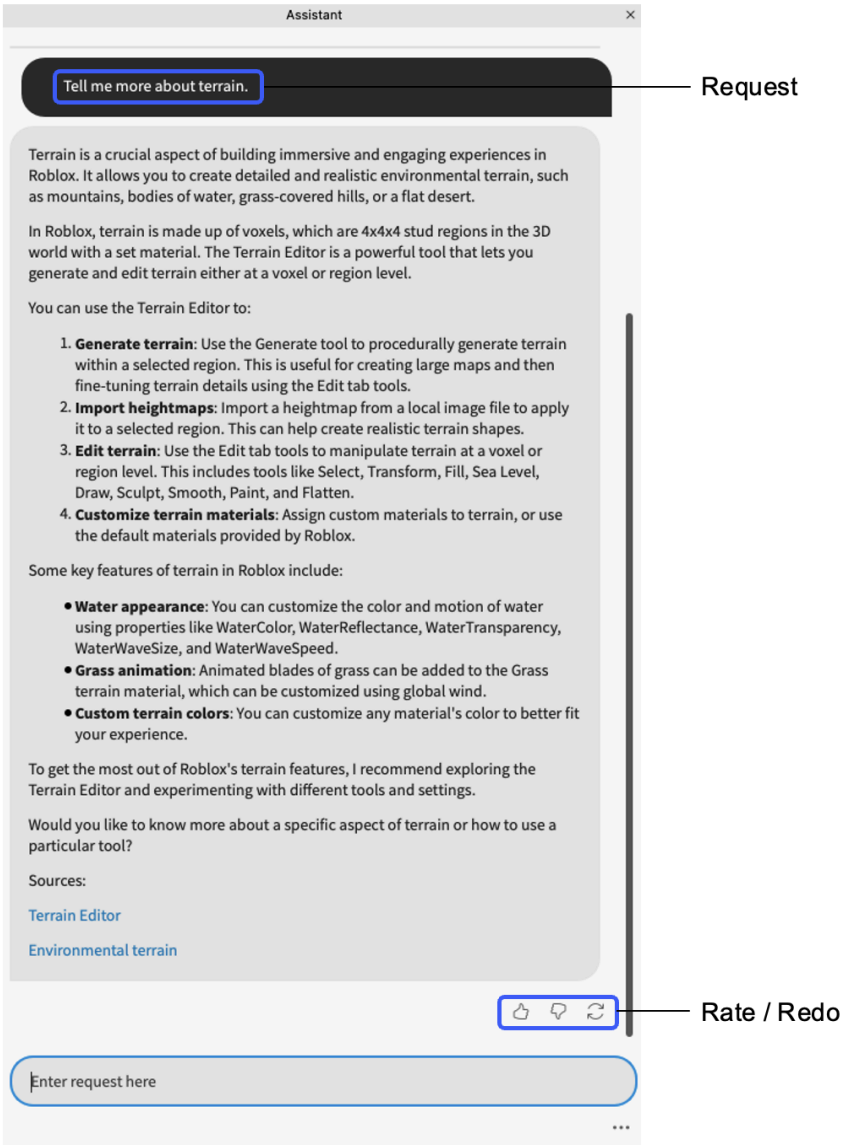
Request Variation	Possible Results
~~`Add a new script that`~~ `creates yellow sparkles while GlitterPart is touched.`	Assistant might show you the code, but not make a new script. It might also add the code to a random existing script.
`Make purple sparkles when the part named GlitterPart is stepped on.`	Assistant might make sparkles before GlitterPart is stepped on, or make only a few sparkles.
`Make GlitterPart sparkle`	Instead of adding a script, Assistant will just add a particle emitter (the object which makes sparkles) to GlitterPart. Which can be good if you want it to sparkle at all times.

Undoing Bad Results

- If you do not like Assistant's results, press Ctrl/Cmd + Z to undo them.
- If Assistant made a script:
    - You may click on the script's name in the Assistant Window. This will highlight it in Explorer.

- Press BackSpace or Delete to remove the unwanted script and start fresh.

- Assistant is always learning how to better help you. You can thumbs up or down to tell it if it did a good job or not.



Learn Coding Logic to Create Better Assistant Requests

Like many AI systems, Assistant is a helpful tool, but you must always check the results of your request to make sure they generate the outcomes you intended. The fewer details you provide, the more unpredictable the results. We will try an example.

1. Make sure you have stopped the playtest. It is always important to click the red stop square after a playtest.

2. Request: “Make parts sparkle when you touch them.” What do you expect to happen when you playtest again?

Teaching Note
Students will likely guess that it will make all the parts sparkle when touched. However the phrase “Make parts sparkle when you touch them” is unlikely to work that way.

3. If Assistant asks you to accept new code, accept it.
4. **Playtest** and compare what you think would happen with what actually happens.

Although Assistant is constantly learning and getting better, it sometimes produces results that are other than you intended. Possible outcomes for “Make parts sparkle when you touch them”:

- The resultant script may not work at all
- Only a single random part may work
- Parts may sparkle before they are touched

Get better results by learning to speak like computers do. Code is the language computers speak, and the more you can speak in the computer's language, the easier it will be for Assistant to help you. The following request uses words from several coding concepts to get a more reliable result:

- Try the following request: “Add a new script. For all objects in the workspace, if a player touches a part then make the part sparkle”

Recap (5 minutes)

Discussion Questions:

- Why is it still important to learn how code works if AI is so useful?
    - Answers: AI does not always generate the code for actions you intended. Learning to code can help you talk to AI and generate better results.
- What types of jobs do you think there are at gaming companies?
    - Answers: Engineers, artists, musicians, organizers (project managers), accountants, moderators. People who train AI.

Lesson Recap:

- Today we learned about:
    - Roblox, Roblox Studio, how to test games, and why you need to learn more about coding even with helpful AI like Assistant.
- Next lesson we will:
    - Build our own obby from the beginning
    - Learn about user testing

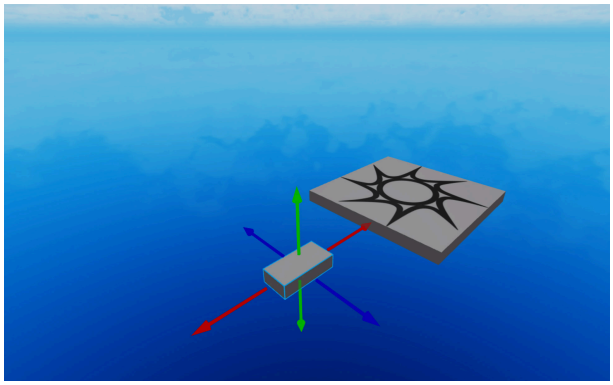
Module 2: Create Your Own Obby and Understand Roblox Community Standards (55 Minutes)

Description

Cover the basics of creating objects within Roblox Studio after understanding the Roblox Community Standards. Students will practice working in a 3D space to create the first level of an obby. While doing so students will be introduced to the Explorer and learn how to move, rotate, and scale parts. As they create their game, they will be asked to playtest their work and consider the end-user's experience. Students will also learn basic keyboard and mouse commands to control a character.

Objectives

- Understand Roblox Community Standards
- Open a new template
- Learn Studio camera controls
- Create new parts
- Scale, move, and rotate parts into position
- Consider the end user
- Save and update projects



Creating an obby from the beginning

Optional Resources

- Studio UI Reference
- Studio Hotkeys

Lesson Introduction (2 minutes)

This lesson will go through the steps of creating a new Roblox experience, which begins with an understanding of the standards that govern the Roblox community. Students will learn how to work with parts and materials while creating an obby, and use Roblox's generative AI tool, Assistant, to help make their game look unique.

- Teacher: "Who remembers what 'obby' means?"
- Students: "Obstacle course"

Guided Practice (43 minutes)

Understanding Roblox Community Standards

As students create, they are part of the Roblox community. Introduce the concept of Community Standards as the rules that help keep everyone safe and ensure a positive environment.

- Key Principles: Discuss the core principles: Safety (especially child protection), Respect, Fairness, and Appropriate Content.
- Respect and Civility: Define what respectful interaction looks like on Roblox, both in games and in communication. Discuss the importance of kindness and treating others as they would like to be treated.
- Prohibited Behavior: Clearly outline behaviors that are not allowed, such as bullying, harassment, hate speech, discrimination, sharing personal information without consent, cheating, scamming, and infringing on intellectual property.
- Intellectual Property: Briefly explain what intellectual property is (creations like games, art, music) and why it's important to respect the creations of others. Students should only use assets they have permission to use.
- Discussion/Activity: Have students brainstorm examples of positive and negative interactions they might see on Roblox. Discuss how their game design can encourage positive interactions.

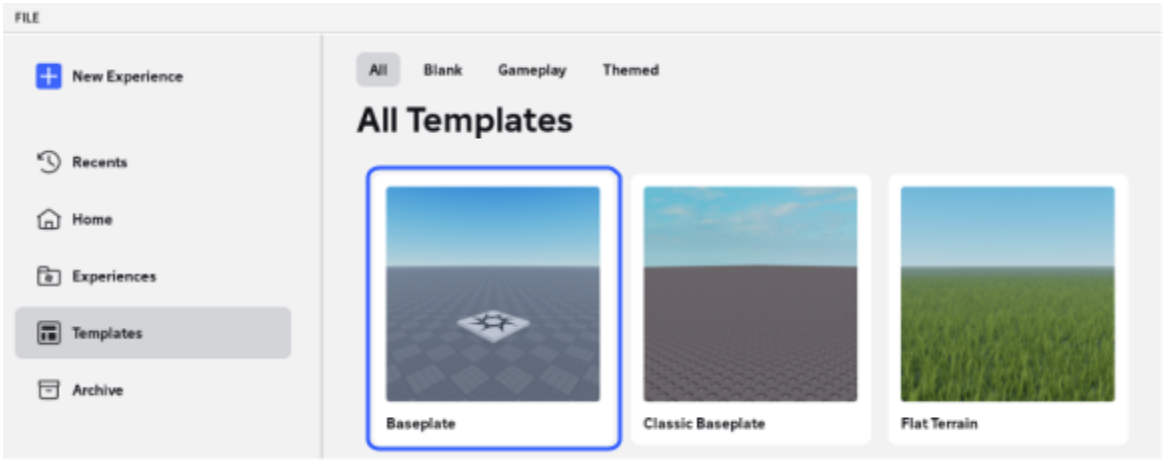
Teaching Tip
A fun way to help students better understand how to be kind online is to play the "Kindness Kingdom" mini-game in Google's Be Internet Awesome World. Students can play together or individually, and the mini-game helps students to Better understand kind interactions online.

Creating a New Project (5 minutes)

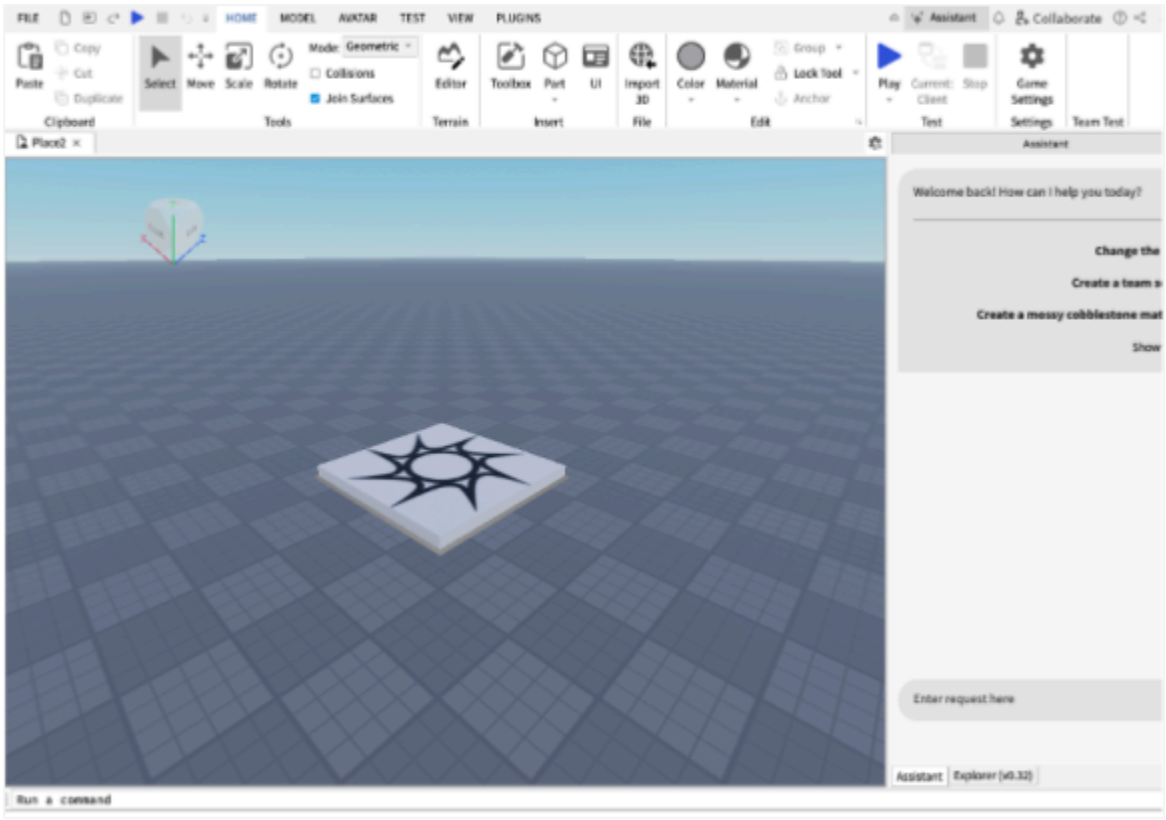
Now that you know what an obby is, you can begin creating your own. Start from a **brand-new empty project file**.

1. Click the **X** to close out of the Obby template if you have not already.
2. Open Studio.

3. Click **Templates**, then select the first **Baseplate** template.



4. Studio should look like this:

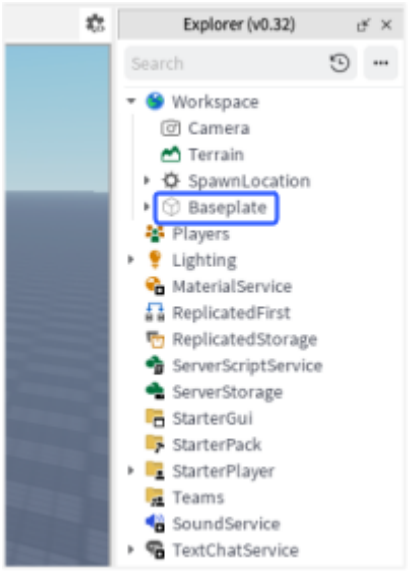


Delete the Baseplate

If you start building your obby above the baseplate, your player will fall harmlessly onto the baseplate rather than dying when they miss their jump. You will need a completely empty world to start building your obby.

To delete the baseplate:

1. Click the arrow next to **Workspace** in the Explorer window.
2. Select **Baseplate**.



3. Press **Delete** on the keyboard.

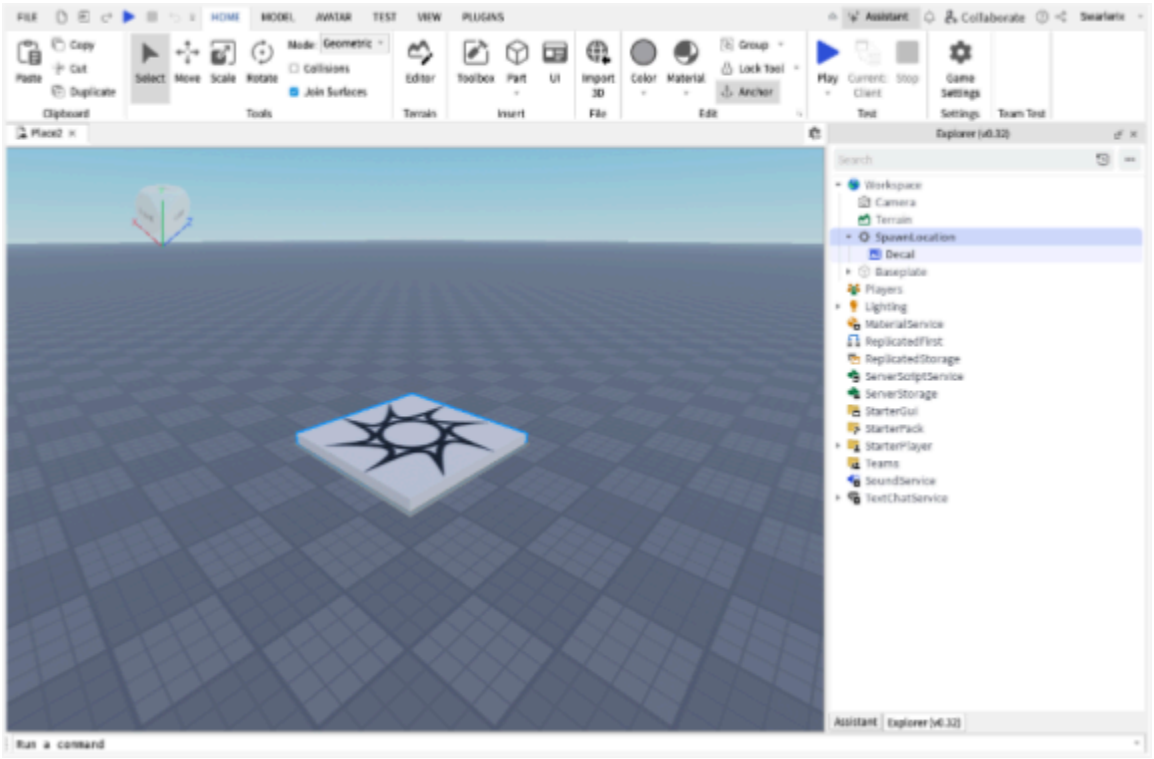
Teaching Note
By starting with a blank space, students better learn how to manipulate camera And creation tools by adding and placing blocks.

Focus on the SpawnLocation

Where a player appears in the world at the start of the game or after dying is called the **SpawnLocation**. This is where your player will start the obby. Learn how to get a better view of the SpawnLocation.

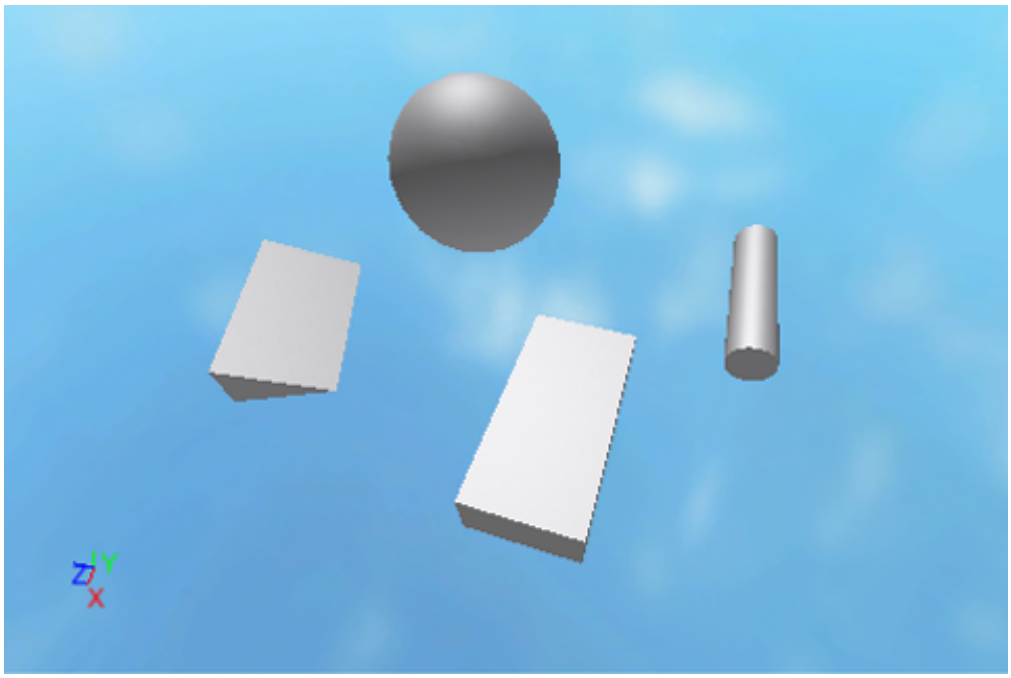
To focus the camera on the SpawnLocation:

- Select **SpawnLocation** in the Explorer.
- Press the **F** key to focus the camera on the selected part.



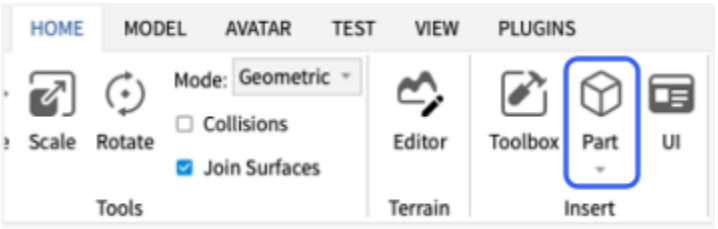
Adding and Moving Parts (10 minutes)

Parts are the building blocks of your game. You can use them to build environments and models for your game.

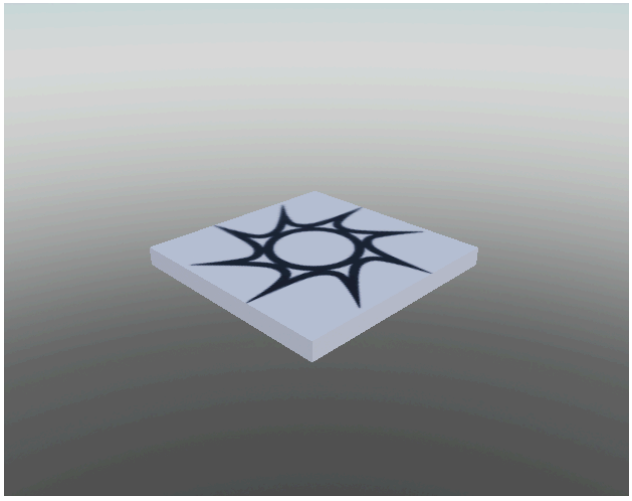


Add A Part

- In the **Home** or **Model** tab, click **Part**.



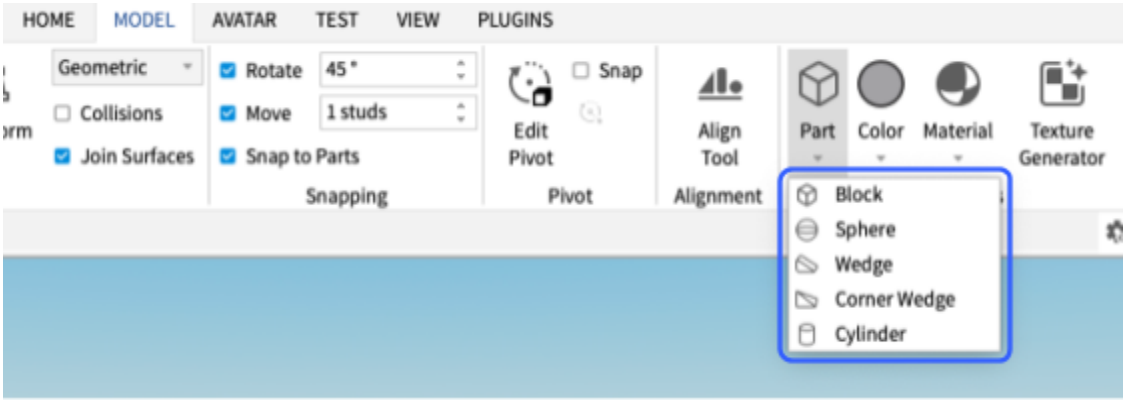
A part will appear at the exact center of your camera view. If you want more control over where the part appears, zoom in your camera with the right mouse button and center it on where you want the part to appear.



Change the Part Type

There are four different kinds of parts; Block, Sphere, Wedge, and Cylinder. You can change which part type the part button creates.

- Use the drop-down menu beneath **Part** and select a new part type to create.

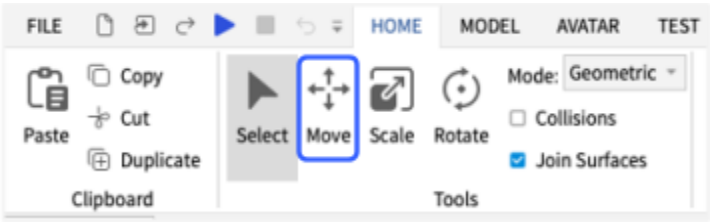


Move the Part

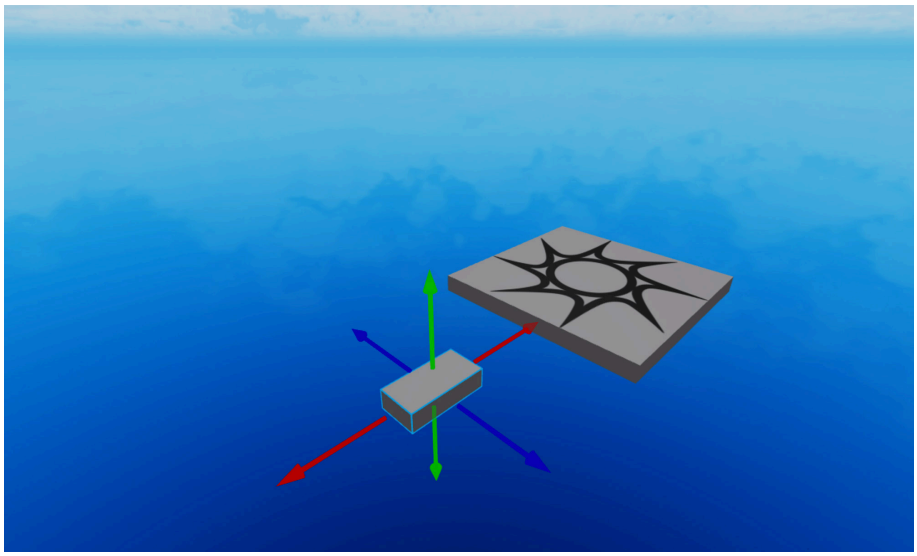
An obby usually starts with a simple jumping puzzle. As a good game designer, you want to make it easy for new players to get started. If you make it too hard right away, players may just quit instead of continuing to play.

1. Click on the part.
2. Use the camera controls or press **F** to focus on a better view.

3. Click the **Move** tool.



4. Drag the arrows to move the part.



5. Use the camera controls to check your part's position from different angles. If the camera does not move, try clicking the main camera window.

Camera Controls:

Action	Control
Move	W A S D
Rotate	Hold the right mouse button and look around
Zoom	Use the scroll wheel on the mouse
Focus	Press **F** to focus the camera on a specific part

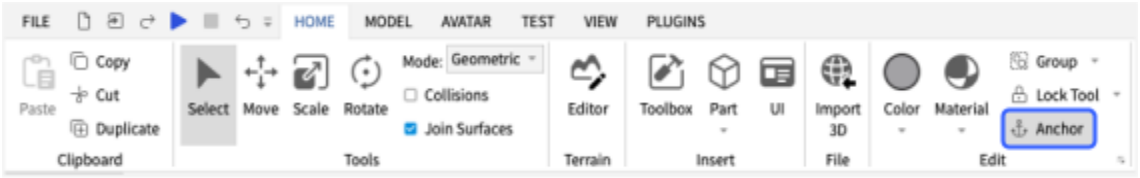
Anchoring Parts In Place (3 minutes)

If you test your project at this point, you will notice any parts you have added (other than the SpawnLocation) will fall. **Anchoring** will stop parts from falling. They will even stay in place when players and other objects bump into them.

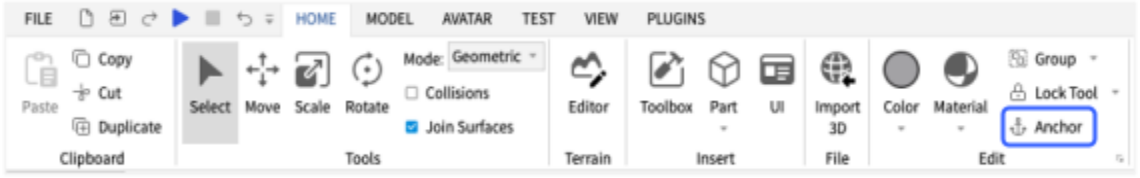
To anchor parts:

1. Select the part you would like to anchor.
2. Select the part you would like to anchor.
3. In the top tool bar, find and click the Anchor icon.

Anchored (is highlighted)



Not Anchored (not highlighted)



4. **Save and Playtest your project**. Your new part should be where you placed it. If you do not see the new part, stop the playtest and make sure it is anchored as shown.

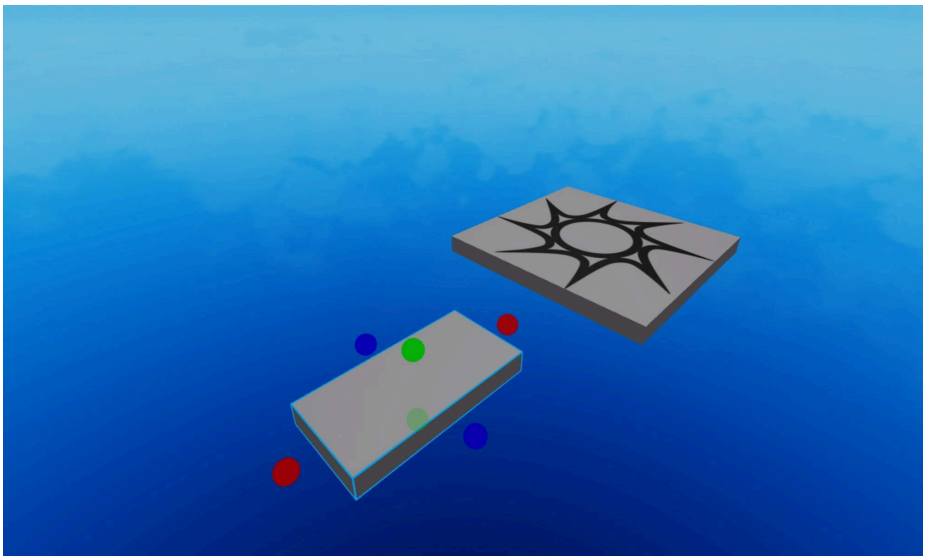
Scaling and Rotating Parts (5 minutes)

An entire level of the same size and shape of parts would not be exciting to players. By scaling and rotating parts, you can create different parts that will add variety to your obby.

To change the size of your part:

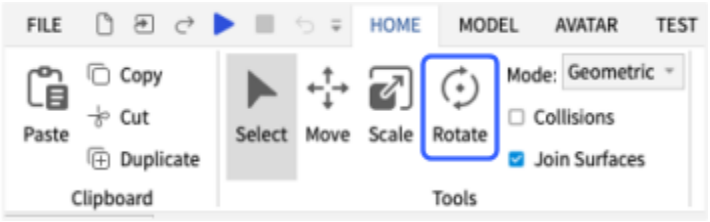
1. In the Home tab, click the **Scale** button.

2. Drag the spheres to resize the block.

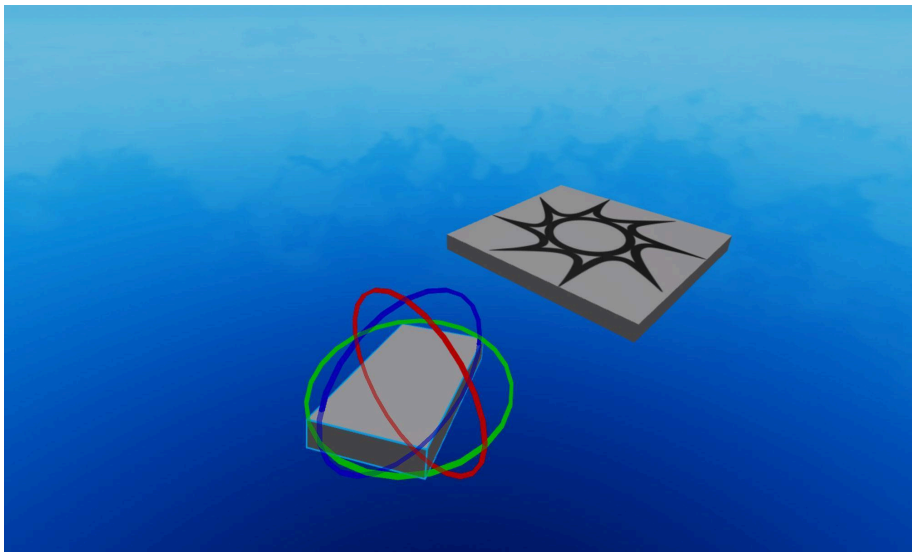


To rotate a part:

1. Click the **Rotate** button.



2. Drag the sphere to rotate the part.



If you are having trouble rotating or moving a part, make sure it does not intersect (collide) with another part. Make sure **Collision** is off.

Add Another Jump (5 minutes)

This is the beginning of your game, so you do not want the jumps to be difficult yet. You want the player to think your game is fun and keep playing. As you build, experiment with rotation, scaling, and snapping amounts and turning snap off.

- Add two additional parts to your game
- Save and playtest your game

Do not forget to anchor
Did your parts fall down? Do not forget to **Anchor** them!

Saving Projects (5 minutes)

Whenever you are working on a game, **save it every 10 minutes**. That way if there is a sudden event like an alien invasion or a power outage, at least you have not lost much work.

Save your work (Recap)

1. In Roblox Studio, click **FILE** > Publish to Roblox
2. Name your file **FirstnameObby_01** or with a meaningful title so you can find it later

> **Teaching Note**
> Before moving on, confirm every student in your class has the file named and saved appropriately. You can walk around the class if it is small enough, or have students check each other's work. Reiterating good file saving practices early will prevent upset students with lost work later.

Save Different Versions

Whenever you make new versions of your save file, change the number at the end of your filename. This way if something goes wrong with your file you can revert to a version where everything works as it did.

Even professional game designers lose work once in a while. Be prepared and it will be easy to get going again.

Opening Your Files

1. Open Roblox Studio
2. Click Experiences to find all the games you have saved to Roblox.

Finding Lost Files

Studio will save a temporary backup of your work every so often. If for some reason you can not find your file, go to:

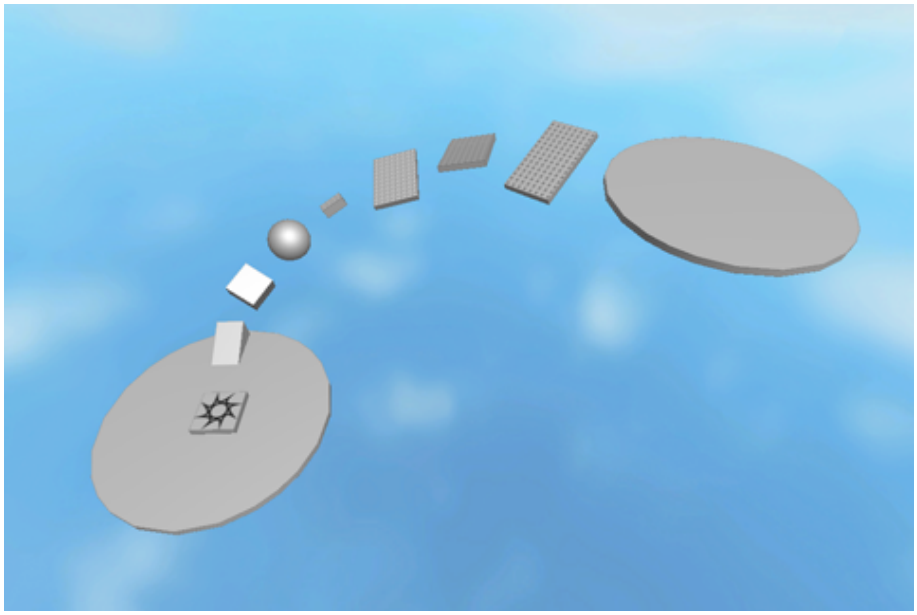
1. FILE > Advanced > Recent.
2. Open your file and resave it someplace you will be able to find it.

Independent Practice (10 minutes)

Add Starting and Rest Zones

The starting area is the first thing a player sees when they land in your game. Starting areas are places to introduce new players to your world, and begin setting themes for your game. Starting areas can be as simple as one big part to create a floor, or as fancy as you would like.

1. Add 4-5 parts of different sizes and shapes to create a jumping puzzle.
2. Create a starting area by placing a larger part beneath your spawnLocation.
3. Create a floor at the end of your first set of jumps to act as an endpoint for that level and to give players a place to rest.
4. Optional: Use Assistant to make the parts change color, or sparkle when stepped on.



Reminders

- As you create more parts, remember to view your obby course from multiple angles. Parts might not line up the way you think if you are only viewing from one direction.
- If any of the parts fall into space, you probably forgot to anchor them.

Example Roblox game file: DesigningAnObby_FinishingYourLevel_End.rbxl

Recap (2 minutes)

Discussion Questions:

- Why is it important to check your work?
- What happens if you do not anchor a part?
- How might new players be different from experienced players when they play your game?
- If you join a game, and it is way too hard, how do you feel? Would you stay or leave?

    - Remembering that everyone has different skill levels is important. Game designers create user personas, or pretend people with differing levels of experience, and try to think about how those different pretend people might feel throughout their games.

Lesson Recap

- Today we learned about:
    - How to create parts, the importance of testing our work, and why you do not want to make the game so hard only you can beat it.

- Next lesson we will:
    - Learn how to change the look of the obby
    - Use Assistant to change the time of day

Module 3: Creating a Theme (45 minutes)

Description

Change the visual stylings of the obby using both the standard Studio tools and Assistant.

Objectives

- Add Color and Materials
- Provide effective peer feedback
- Understand and develop a theme for the obby
- Use Assistant to create new materials

Optional Resources

- Studio UI Reference
- Optional Handout: Brainstorming
- Handout: Roblox Studio Cheatsheet
- Handout: Project Feedback
- Presentation: Project Feedback

Add colors and materials to parts

Lesson Introduction (5 minutes)

This lesson will review the steps of creating a new Roblox experience, learning how to work with parts and materials while creating an obby, and using Assistant to help make your game look unique. It is important to consider content maturity labels for Roblox experiences given that creators must complete a content maturity label questionnaire when publishing any experience on Roblox.

Content Maturity Labels

Creators have a responsibility to understand and apply Content Maturity Labels to their Experiences.

Source:

Action	Control
Minimal	May contain occasional mild violence, light unrealistic blood, and/or occasional mild fear.
Mild	May contain repeated mild violence, heavy unrealistic blood, mild crude humor, and/or repeated mild fear.
Moderate	May contain moderate violence, light realistic blood, moderate crude humor, unplayable gambling content, and/or moderate fear.
Restricted	May contain strong violence, heavy realistic blood, moderate crude humor, romantic themes, unplayable gambling content, the presence of alcohol, strong language, and/or moderate fear. These experiences are only available to 17+ users who verified their ages by completing ID verification.

https://en.help.roblox.com/hc/en-us/articles/8862768451604-Content-Maturity-Labels

Teaching Tip
It is important for students to understand content maturity labels as they create their experiences on Roblox. In the process of publishing their game, creators must respond to a content maturity labels questionnaire that helps Roblox to properly classify their game. Their responses contribute to the content maturity label applied to the experience, which in turn impacts the users on Roblox who may access the experience. For example, users under age 9 can only access “Minimal” or “Mild” content by default and can access “Moderate” content only with parental consent. Learn more about the create content maturity label questionnaire here: https://create.roblox.com/docs/production/promotion/content-maturity.

- Content Maturity Labels: Introduce the different maturity levels: Minimal, Mild, Moderate, and Restricted (17+). Explain what types of content (violence, blood, crude humor, romantic themes, language, fear, gambling, alcohol) are associated with each level.

- Creator Responsibility: Explain that creators are required to accurately label their experiences based on the content they include. This helps users and parents make informed decisions about what experiences are suitable.
- Designing Responsibly: Discuss how students can make conscious choices about the content in their games to align with their target audience and the Community Standards. For example, if creating a game for younger players, they should avoid content that would fall under Moderate or Restricted labels.
- Activity: Have students consider the game they are building and discuss what content maturity level it would likely receive based on the elements they plan to include.

Creating a Theme

- What is a theme?

A theme is more than simply colors for a game. A theme creates the setting and mood for your game, which often starts with a color scheme and then builds in other objects. What colors would you use to create the following settings:

- A happy day at the park
- A volcano about to blow up
- A unicorn ranch on the moon

In addition to color, professional game developers use props, lighting, narrative, and sound to create a theme for their games.

Guided Practice (20 minutes)

Choose Your Theme

What would you like the theme for your obby to be?

Color the parts you have used in your game so far to match it. For instance, if your obby is space themed, all of the parts might be colored and shaped like asteroids. If it is unicorn themed, maybe the parts are purple, pink, and blue. Whenever you make choices, think about how they can relate to your theme. can not think of a theme? Write down the following:

Examples:

- If you could visit anywhere, where would it be?
- Your favorite book or movie

- Your favorite sport

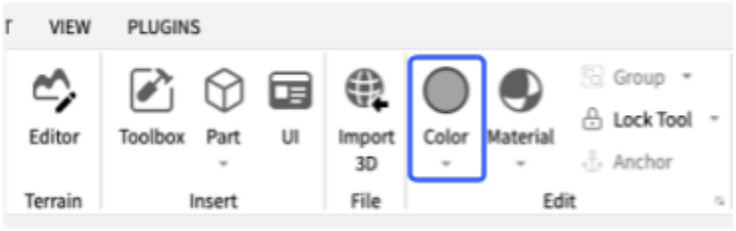
Any of those things can be a theme for your obby!

Plan Your Theme

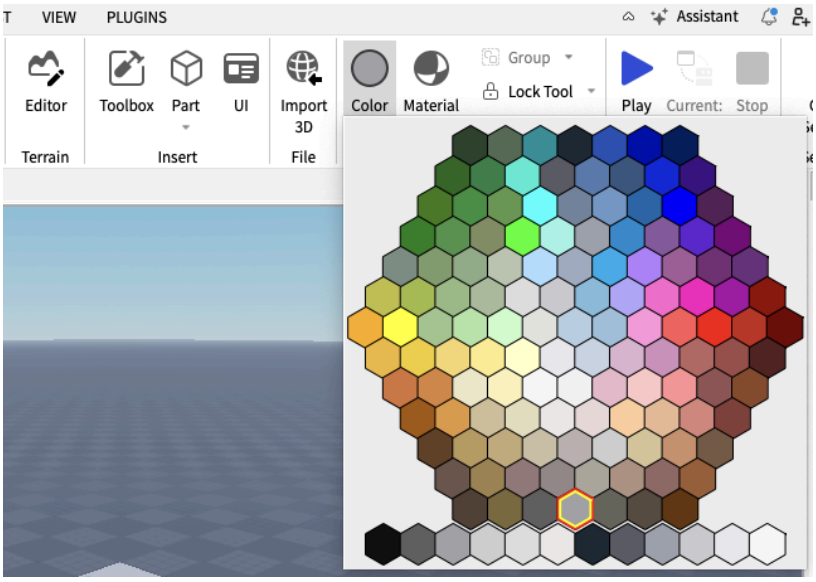
- Write down your theme
- Write down colors and part shapes that match your theme

Changing How Parts Look

1. Select the part
2. In the **Model** tab, click the arrow beneath **Color**



3. Select a color

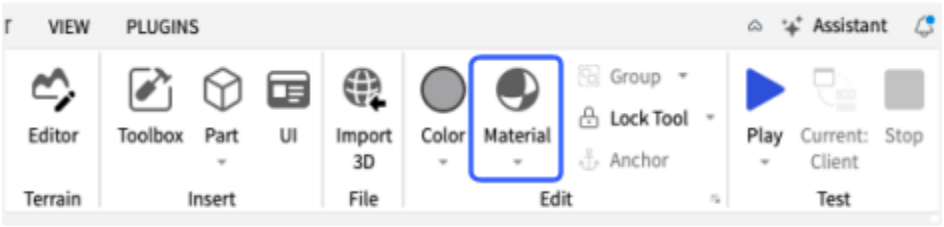


4. Ask your classmates for their advice on how to make your colors match your theme.

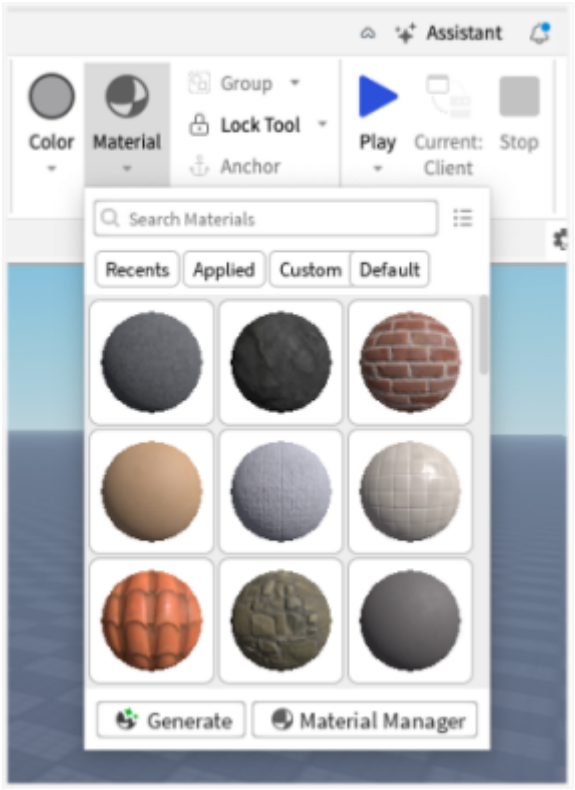
Teaching Tip
Give students 3-5 minutes to complete this activity. Ask students how their color choices contribute to the theme for their game.

Change the Material of Parts

1. Select a part.
2. Click the arrow beneath **Material** to select a material you would like to use



3. Once you have a material selected, you can click the color square to assign it to any selected part



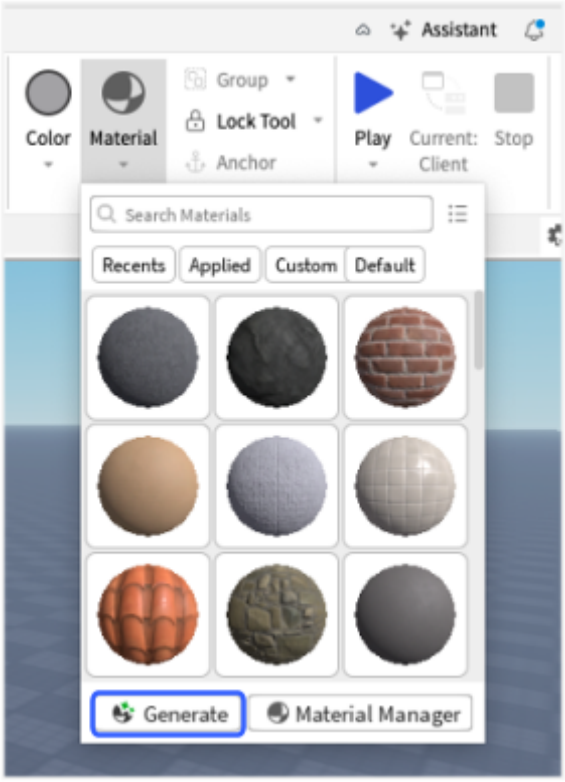
Teaching Tip
Give students a minute or two to change the material of a few parts.

The Material Generator

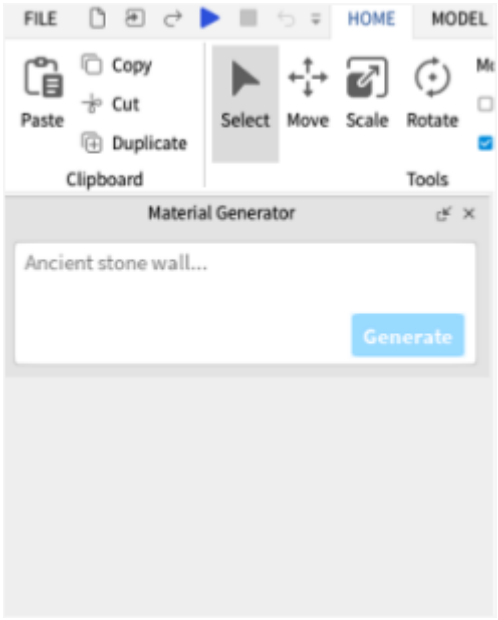
The **Material Generator** is AI designed to create material variants from text entries. Using it, you can type any phrase and select **Generate** to see results within seconds. Once you find a satisfying result, you can instantly save it as a new custom material.

Generate materials

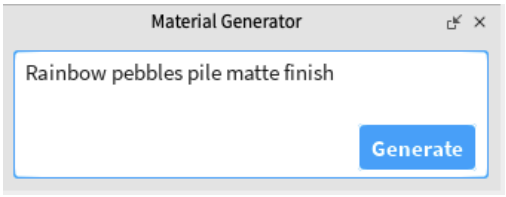
1. Select the part you want to apply a material to.
2. From the Material Picker dropdown, click **Generate**.



A new window will open on the left side.

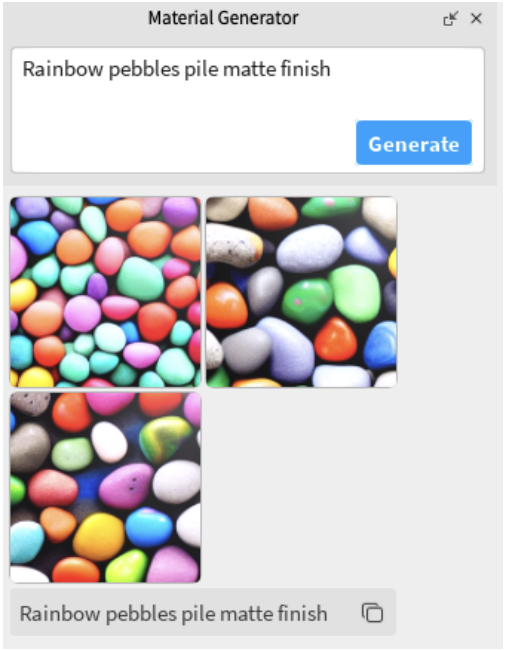


3. In the text box at the top of the window, describe the material you would like to see and then click the Generate button.

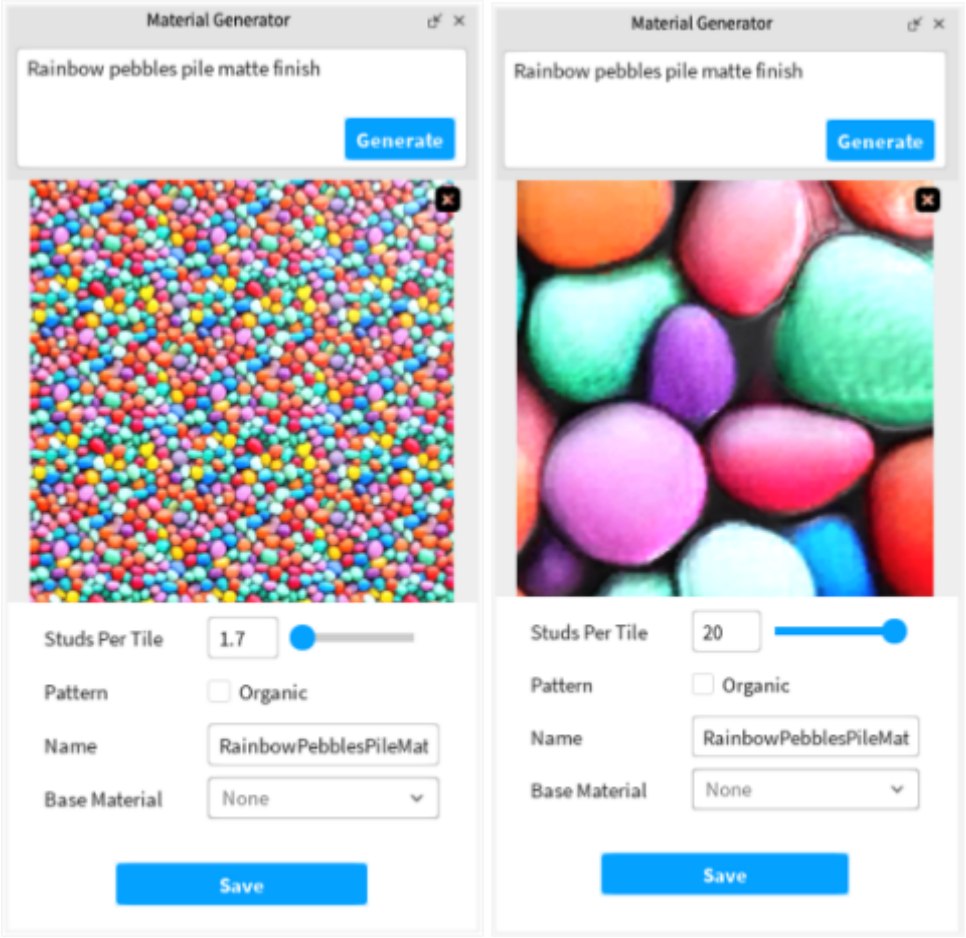


4. Every click of Generate yields different results, even with the exact same keywords.

   Example Results : “Rainbow Pebbles Pile Matte Finish”



5. Adjust the **Studs Per Tile** slider to control the size of the repeating texture. Additionally, test out the **Organic** toggle which makes materials appear less "repetitive" by randomizing the output.



6. Select the **Base Material**. If unsure, select something plain like “plastic” or “ground.”
7. Click **Save**.

Best practices

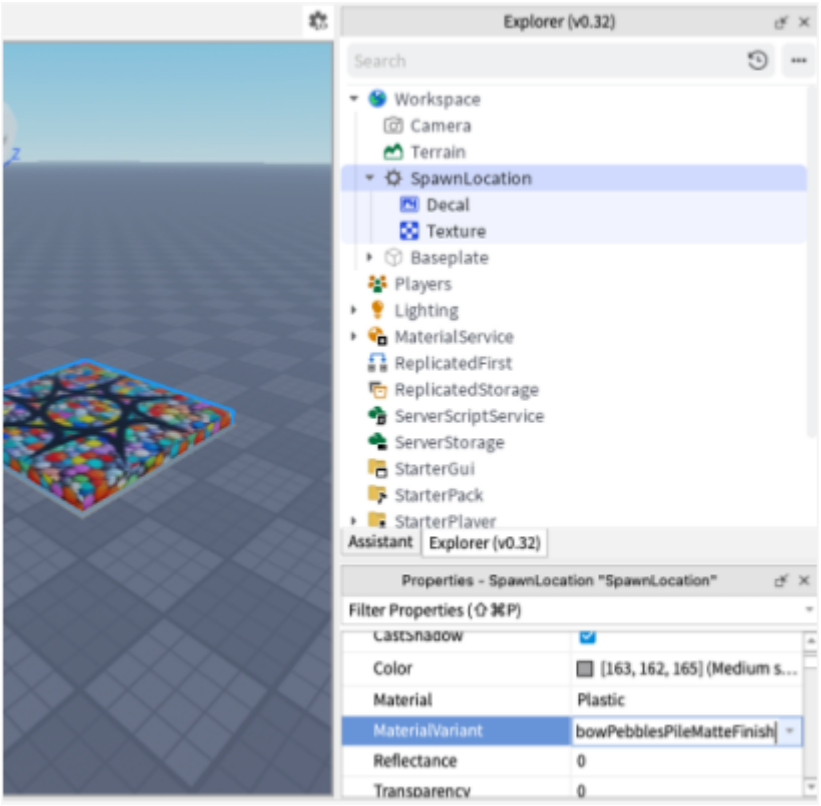
Generating satisfying materials can be an iterative process requiring a longer list of descriptors to help focus in on the material you want. Here are some tips:

- For close-up patterns, try using terms like "close up," "top down," and "texture."
- For simpler repeating patterns, try using terms like "simple," "pattern," "symmetrical," and "flat."
- For more control, add stylistic terms like "photorealistic," "cartoon," or "hand-drawn."

- For the ability to change colors, try including terms like "grayscale" which will allow you to tint the material afterwards.

Removing or Switching the Material Variant

1. Select the **Part** you want to remove it from in the Explorer window
2. In the Properties window, click MaterialVariant and select <None>



Tips for Working With Parts

Duplicating Parts

Using the same part in multiple places will save you time and strengthen your obby's theme. Duplicating parts and slightly changing them or scaling them also saves the game creator time.

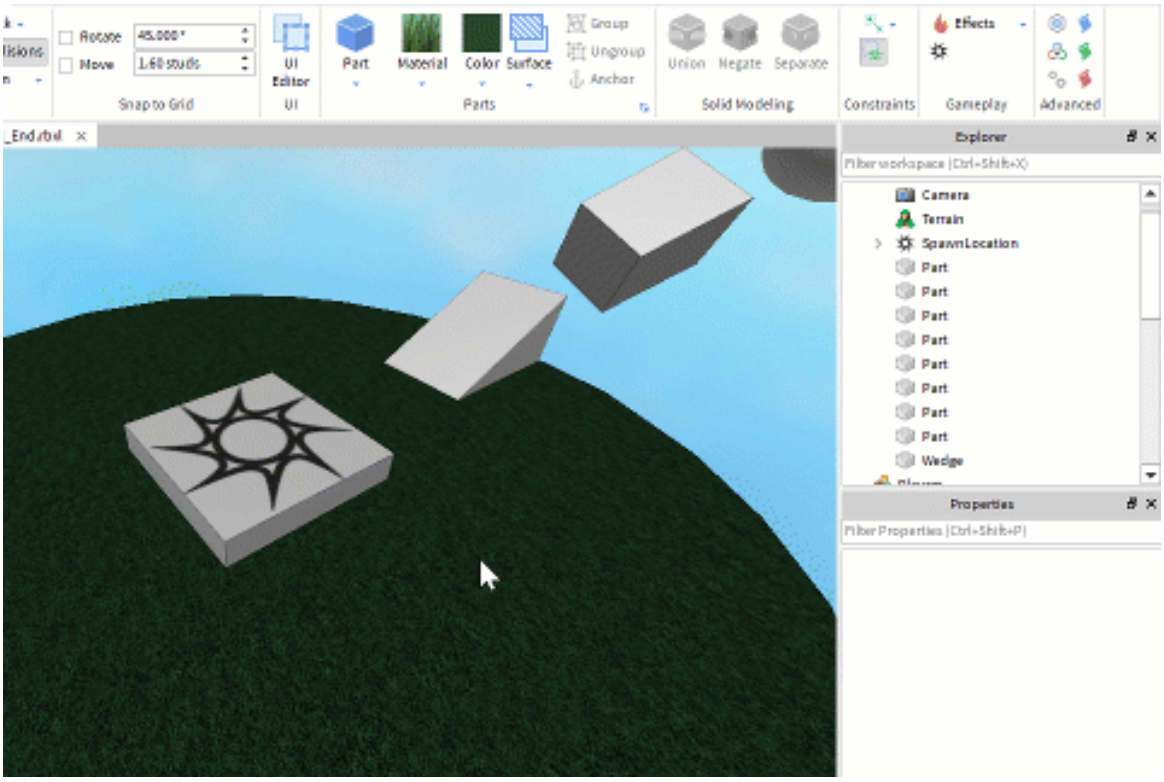
To duplicate a part:

- Select the part and press **Ctrl+D**.

Selecting Multiple Parts

To quickly change an aspect of multiple parts at once, you can select multiple parts.

- Press Ctrl while selecting parts in the Explorer or camera view.



Grouping

If there are multiple parts that you find yourself selecting often, you can organize them into groups. Any change to the group will be made to every part within the group. To create a group:

- Select all of the parts you would like to group together
- Click the **Group** button or press Ctrl+G

To Ungroup parts:

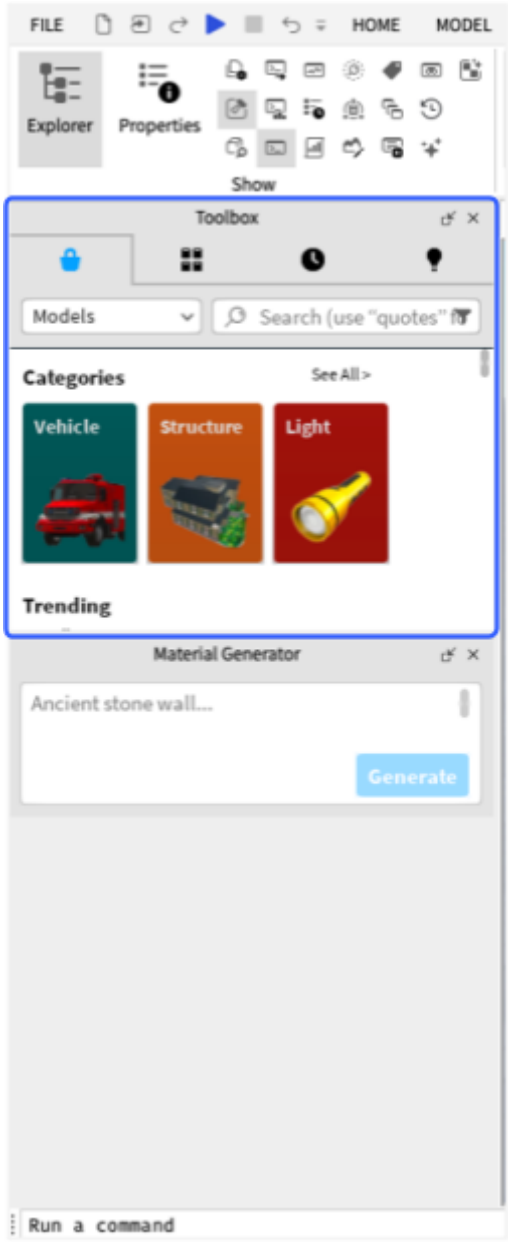
- Click the **Ungroup** button or press Ctrl+U

Change the Skybox

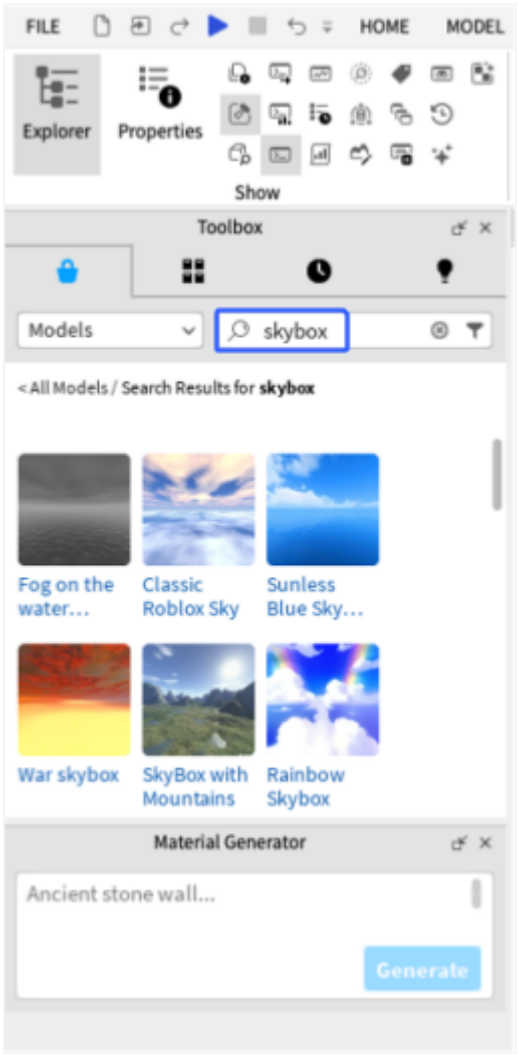
The Toolbox contains a selection of models, images, meshes, audio, plugins, videos, and fonts made by Roblox or Roblox community members. It also includes any creations you have personally published or those which were published by groups you belong to.

1. Select the View tab, then click the hammer icon to open **Toolbox**.

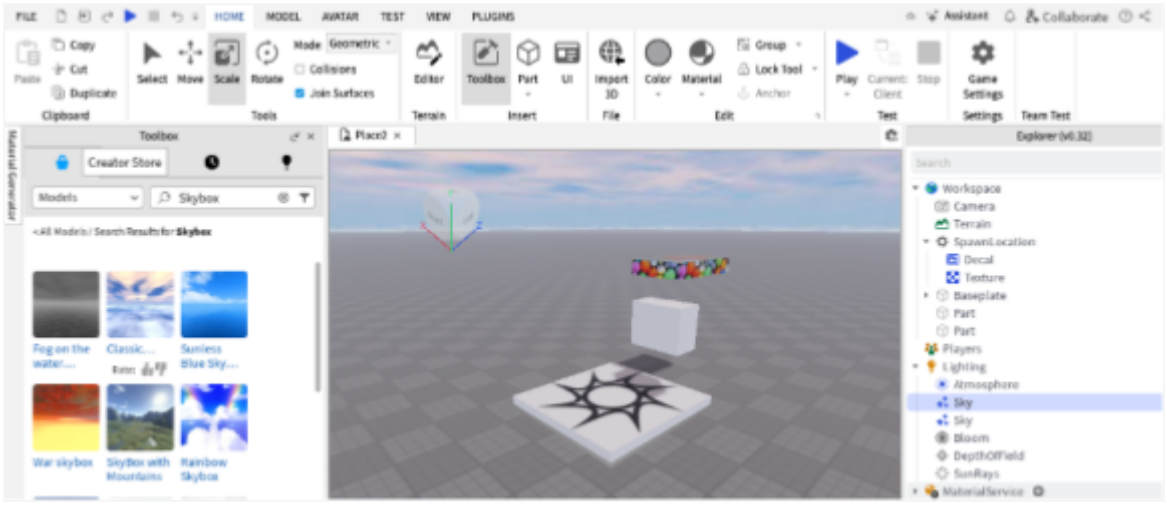
2. The **Toolbox** will open on the left side of the screen, above the **Material Generator**.



3. In the search bar, type "Skybox".



4. Click a tile to apply it to your world's sky.



Peer Playtest (15 minutes)

When engineers or game designers make a new application or game, they always make sure to have other people test it and provide their opinions on how to make it better.

> **Teaching Tip**
> Students must either publish their game and change **File → Game Settings → Permissions** to **Public** for anyone to playtest their game, or set permissions to **Friends** to only allow their Roblox friends to playtest the game.

Leaving Constructive Feedback
[Project feedback]

- Open Presentation: Project Feedback and pass out Handout: Project Feedback to each student. This
    - Explain the process of playtesting and getting feedback in slides 2-3.
    - Connect to digital safety and civility (Roblox Safety & Civility corporate website) - ask students what are some ways they can practice being a good digital citizen while sharing and receiving feedback (ex: Be open-minded with different experiences, do not pressure others to change their projects, etc).

- Have students play each other's project for about five minutes, then write down feedback for their peers.

Teaching Tip
In the last five minutes, make sure students can play each other's projects.

- If students attend class in-person, they can just swap seats.
- If students are remote, have students make their games public so others can play by sharing links

Lesson Recap (5 minutes)

Lesson Recap:

- Reconvene as a class and have 2-3 students share a piece of feedback they received.
- Have students reflect on making one change in their experience and write it down on the handout.
    - Have students form pairs and share the change they will make.
- Have students connect by giving feedback outside the classroom (online games, sports, talking with family). Remember that good online behavior is just like real life behavior. Ask these questions to start:
    - What are good or bad examples of giving someone feedback? Why?
    - What are good or bad examples of accepting feedback? How should you react or respond well when receiving feedback?

Next Lesson we will:

- Learn to write your own code to create a bridge that disappears and reappears.

Module 4: Introduction to Scripting (55 minutes)

Description

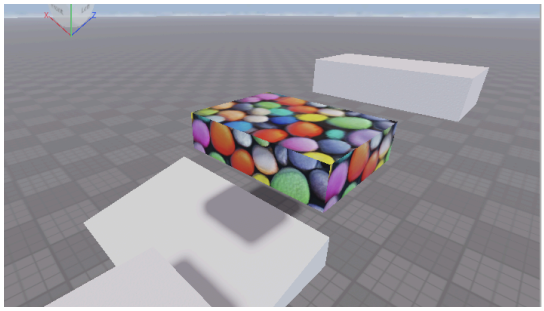
This module begins with non-technical information about how to protect information in their games as well as how to monitor and report any inappropriate behavior in their games. It then transitions to the more technical skills of learning to change objects in their games using code. Students will use variables, functions, and loops to create a part that vanishes and reappears over time.

Objectives

- Add scripts to parts
- Use variables and loops to make changes to parts
- Create and call a function
- Use Assistant to check code

Optional Resources

- Handout: Intro to Coding Cheatsheet



Use code to make a disappearing bridge

Lesson Introduction (5 minutes)

In this lesson, you will learn how to respect the privacy of users with your game design as well as how to maintain a safe and appropriate environment in your game. You will add a script to parts to make a bridge appear and disappear. You can use this bridge to span a gap, challenging users to time their jumps carefully to get to the other side.

Guided Practice (45 minutes)

Protecting Information in Your Game

When designing interactive elements, consider how to handle player information.

- Personal Information: Reiterate that students should never ask players for personal information (full name, address, school, phone number) within their game or through Roblox chat.
- In-Game Data: Discuss what kind of data their game might collect (e.g., score, time, items collected) and that this data should not be tied to real-world identities.
- Chat Safety: Explain that Roblox filters chat, but students should still be mindful of

what they say and share. For younger users, parental controls may limit chat options.
- Discussion: Talk about scenarios where someone might ask for personal information online and why it's important to say no and tell a trusted adult.

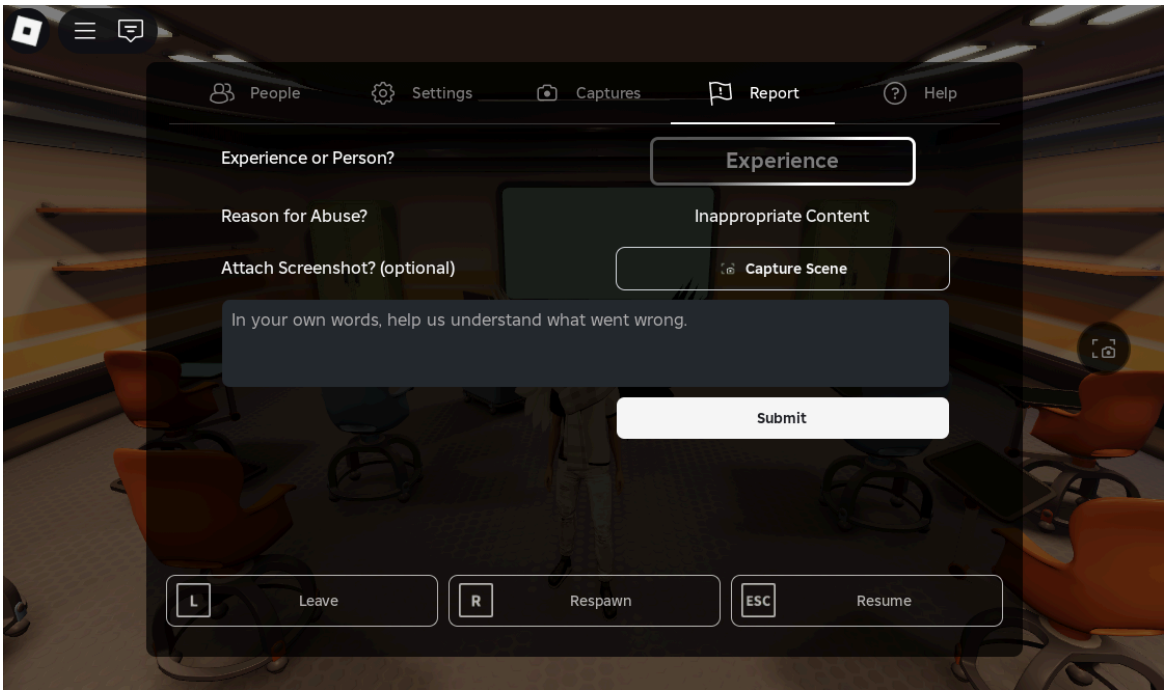
Teaching Tip
While you are introducing privacy controls as part of the student's game creation process, it is important to help students understand that both their games and any game they play on Roblox should never ask for personal information. Having the students play the "Mindful Mountain" mini-game within the Google Be Internet Awesome World experience on Roblox will help students to better understand what types of information they can safely share with different audiences online.

Reporting Inappropriate Behavior

Empower students to take action if they encounter content or behavior that violates Community Standards.

- Why Report: Explain that reporting helps Roblox moderators keep the platform safe for everyone. It's not "tattling" but contributing to a positive community.

- How to Report: Demonstrate how to use the "Report" feature within an experience, in chat, or on a user's profile. Emphasize reporting the *specific* content or behavior that is inappropriate.



- Blocking and Muting: Show students how to block or mute (support link) other users they don't want to interact with. Explain that blocking prevents communication and interaction.
- Discussion: Discuss when it's important to also tell a trusted adult (parent, teacher).

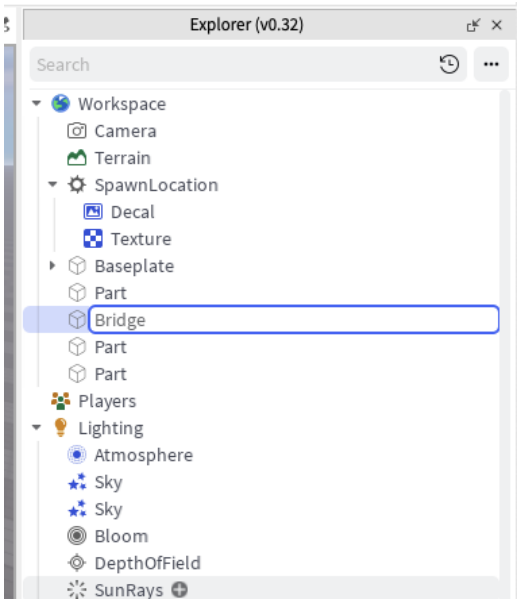
Teaching Tip
Reporting rule violations on Roblox is the responsibility of every user. It is important to reiterate the importance of maintaining a safe and civil environment on Roblox. It is important to make students feel comfortable reporting any perceived violation of the Community Standards even if they are not certain if what they are reporting is a clear violation. Their report will contribute to maintaining Roblox as a safe and civil environment.

Introduction to Scripting

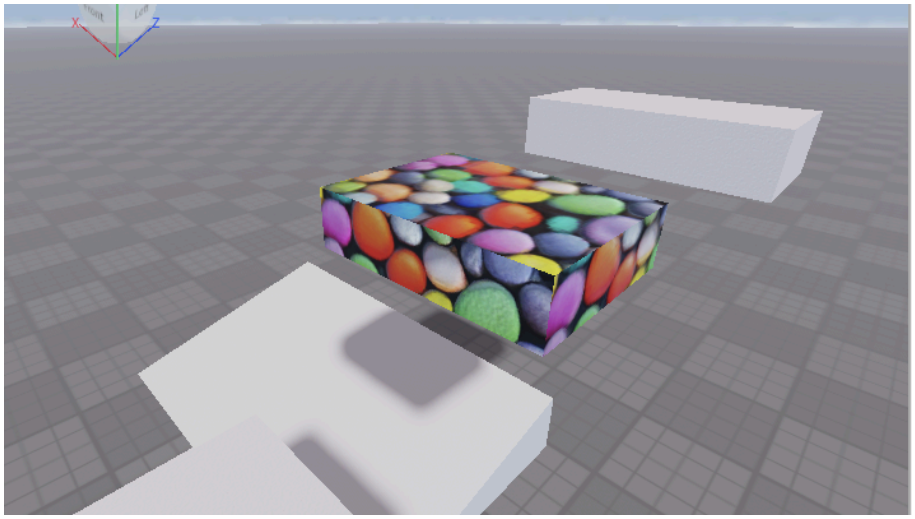
Set the Scene

First, you need a Part to act as the Bridge. You do not need a complicated world aside from the Bridge — you just need a gap that your users can not easily jump across.

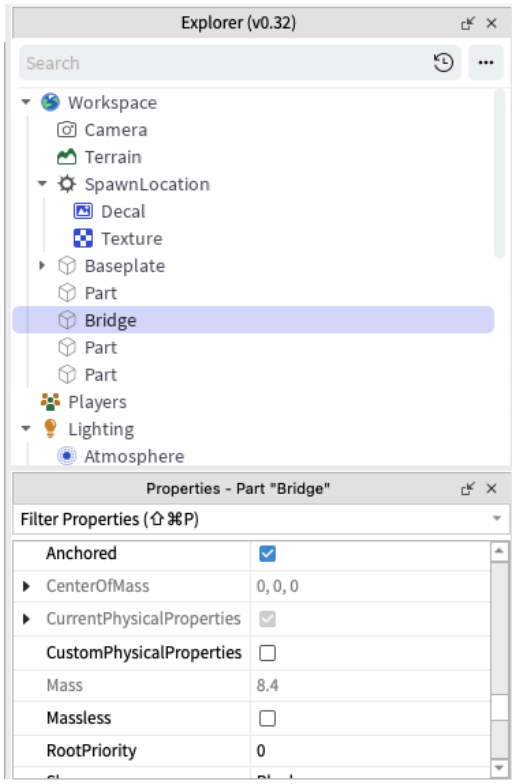
1. Insert a **Part** and rename it `Bridge`.



2. Make the bridge large enough for a user to jump on and place it somewhere you can easily test it.



3. Set the **Anchored** property to **true** in the Properties window. Your Bridge falls down if it is not anchored.

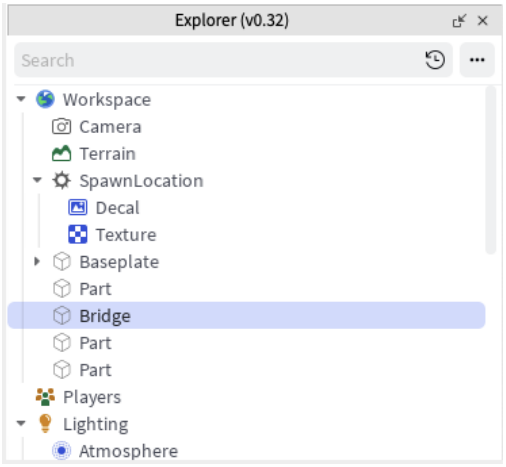


Insert A Script

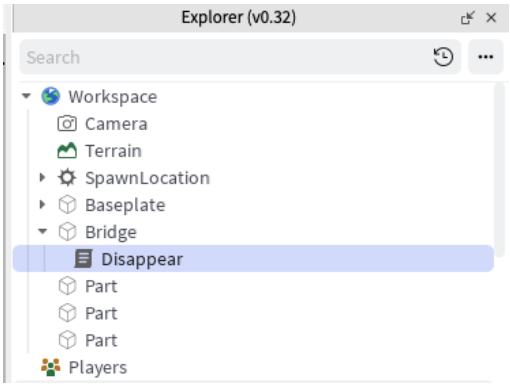
Coding is the process of creating instructions for computers to follow. Just like people use different languages, such as English and Spanish, so do programs. Roblox uses the coding language Lua.

In Roblox Studio, lines of Lua code are held in scripts. These scripts give the game sets of instructions on how to give players health points, create a rain of fireballs, or anything else imaginable.

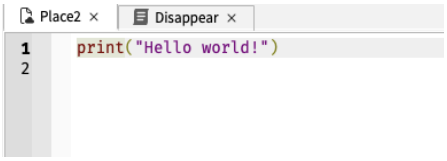
Hover over the Bridge part in the **Explorer** window and click the **+** button. From the dropdown select Script.



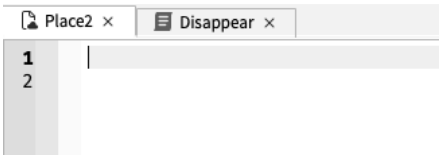
Right-click Script, and rename it **Disappear**.



You should now see:



Delete the default code inside.

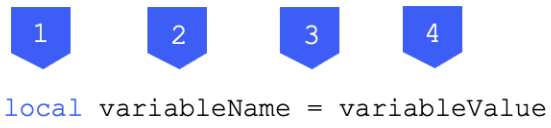


Remember to rename parts and scripts as soon as you create them so you do not lose track of things in the Explorer.

Parts of a Variable

A variable is a **name** associated with a **value**. Once a variable is created, it can be used again and again. You can change the value as needed.

In Luau, the programming language used by Roblox, a variable is created as follows:



1. `local` means that the variable is only going to be used in the block of the script where it is declared.
2. Names for variables are typically written in **"camel case"**. This is lowercase with every word following the first being capitalized, just**L**ike**T**his. There are no spaces between words in variable names.
3. The = sign is used to set the variable's value.
4. The initial value of the variable.

Create Your Own Variable

- In the script, type `local Bridge = script.Parent`.

This creates a **variable** for the Bridge called Bridge, where the **value** is script.Parent.

This variable will now look for the object above the script in Explorer so you can make changes to it.

Use Functions to Create Instructions

A function is a named block of code that helps you organize your code and use it in multiple places without writing it again.

In the code examples below, we use comments to help you better understand each part of the function. A comment starts with two dashes and does not impact the code.

Example: `-- This is a comment`

```
local function functionName() -- You will replace functionName() with
the name of the function. For example here it would be disappear()

      -- List of instructions
end

functionName() -- Tell the code to run, and change this to
disappear() as well.
```

Functions look like:

The first new line **declares** the function — it indicates the start of the function and names it as disappear. The code for a function goes between the first line and end.

The parentheses are for including additional information as needed. Time to make the Bridge disappear. It is always best to group code for achieving a specific action into a **function**.

1. Type `local function disappear()` and press Enter to autocomplete the function.

```
local Bridge = script.Parent -- This is from above and tells the
function what to make disappear.

local function disappear()

end
```

2. **Use Assistant to explain your script.** For common function names like disappear, Assistant will suggest code to you. That will be helpful when you know a little more, but for now you want to learn how the code works so that you better understand code generated by Assistant. Instead of asking Assistant to generate code for you, try asking Assistant to explain the script above.

Teaching Tip
Ask the students why using code they do not understand might be a bad idea.
Possible Answers:

- It might not do what they want it to do.
- If the code breaks, they may not know how to fix it.

Part Properties

When the Bridge disappears, it must be invisible and users must fall through it — but you cannot simply destroy the Bridge since it must reappear later.

Parts have various properties that can be used here. You can see a part's properties if you select it and look at the **Properties** window.

A part can be made invisible by changing the Transparency property. Transparency can be a value between 0 and 1, where 1 is fully invisible.

The **CanCollide** property determines if other parts (and users) can pass right through the part. If you set it to **false**, users will fall through the Bridge.

Make the Bridge Invisible and Untouchable

Just like script.Parent, properties are accessed using a **dot**. Values are assigned using an equals sign.

1. In the disappear function, set the **CanCollide** property of the Bridge to **false**.

```
local Bridge = script.Parent


local function disappear()

   Bridge.CanCollide = false

end
```

2. On the line following, set the Transparency property to 1.

```
local Bridge = script.Parent


local function disappear()

   Bridge.CanCollide = false

   Bridge.Transparency = 1

end
```

You may notice that Studio automatically indents your code inside a function. Always make sure to indent your code like this — it helps indicate where the function begins and ends, which makes your code more readable.

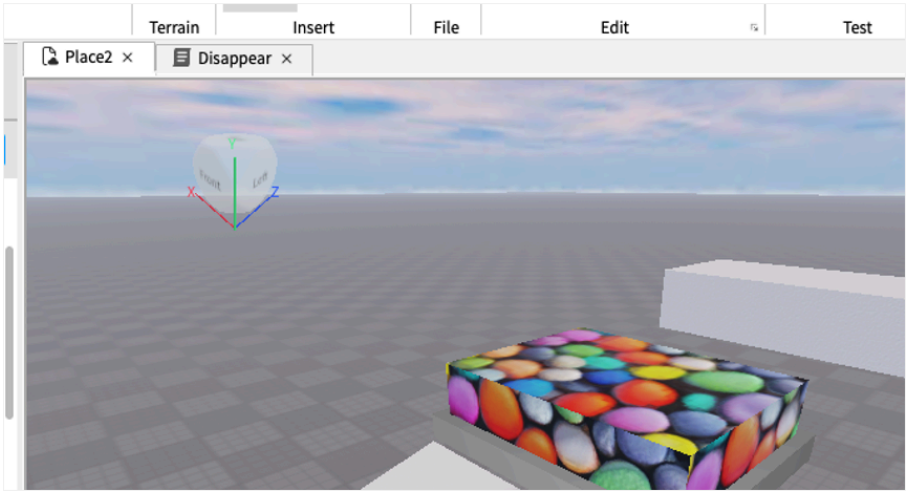
Call the function

Once you have declared a function, you can run it by writing its name with parentheses next to it. For example, disappear() will run the disappear function. This is known as **calling** a function.

1. Call the disappear function at the end of the script.

```
local Bridge = script.Parent

local function disappear()
   Bridge.CanCollide = false
   Bridge.Transparency = 1
end

disappear()
```

2. Press the Play button to test code. If your code works, the Bridge should have disappeared by the time the user spawns into the game.
3. Go back to the Script editor by clicking the tab named Disappear



Appear function

You can easily make the Bridge reappear by writing a function that does the exact opposite of the disappear function.

1. Delete the disappear() line from the script.
2. Declare a new function called appear.

3. In the function body, set the CanCollide property to true and the Transparency property to 0.

```
local Bridge = script.Parent

local function disappear()

   Bridge.CanCollide = false

   Bridge.Transparency = 1

end


local function appear()

   Bridge.CanCollide = true

   Bridge.Transparency = 0

end

disappear()
```

4. At the bottom, use a premade function called task.wait() to wait 5 seconds and then call appear()

```
local Bridge = script.Parent

local function disappear()

   Bridge.CanCollide = false

   Bridge.Transparency = 1

end


local function appear()

   Bridge.CanCollide = true

   Bridge.Transparency = 0

end

disappear()
task.wait(5.0)
appear()
```

5. Test your game, and the bridge should reappear after five seconds.

Loop code

Instead of just running the code once, use a loop to make the Bridge constantly disappear and reappear with a few seconds between each change.

1. Delete the last three lines of code.

```
local Bridge = script.Parent

local function disappear()

   Bridge.CanCollide = false

   Bridge.Transparency = 1

end


local function appear()

   Bridge.CanCollide = true

   Bridge.Transparency = 0

end

-- Delete below
disappear()
task.wait(5.0)
appear()
```

A while loop runs the code inside it for as long as the **statement** after while remains true. This particular loop needs to run forever, so the statement should just be true.

2. At the end of your script. Type while true do and press Enter to autocomplete the loop.

```
local Bridge = script.Parent

local function disappear()
   Bridge.CanCollide = false
   Bridge.Transparency = 1
end

local function appear()
   Bridge.CanCollide = true
   Bridge.Transparency = 0
end

while true do


end
```

3. Inside of the while true do loop:
    a. wait 3 seconds
    b. Call disappear()
    c. wait three more seconds
    d. Call appear

Final code

```
local Bridge = script.Parent

local function disappear()

   Bridge.CanCollide = false

   Bridge.Transparency = 1

end


local function appear()

   Bridge.CanCollide = true

   Bridge.Transparency = 0

end


while true do

  task.wait(3)

  disappear()

  task.wait(3)

  appear()

end
```

4. Test your code.

Lesson Recap (5 minutes)

Discussion Questions

- How many people made mistakes while they were coding?
- How did you fix your mistakes?
- When do you think you stop making mistakes as a professional engineer?

Lesson Recap

- Variables - Hold information

- Functions - A set of instructions that can be re-used
- Loops - Repeat code until they are stopped

In the next module we will:

- Learn how to make realistic nature scenes
- Create signs that players can read

Module 5: Creating an Environment (50 minutes)

Description

Create a realistic environment for the obby.

Objectives

- Add Color and Materials
- Provide effective peer feedback
- Understand and develop a theme for the obby.
- Use Assistant to help decorate.
- Create Signs

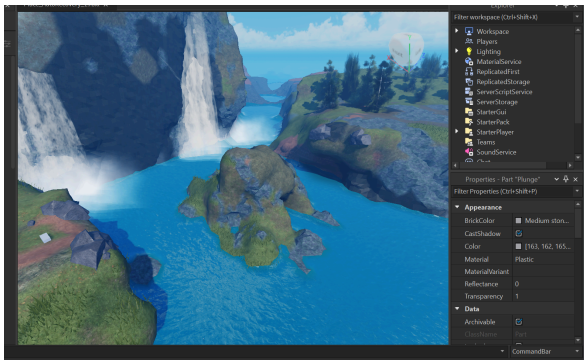


Image Description

Optional Resources

- Studio UI Reference
- Optional: Theme Worksheet
- Studio Cheatsheet

Lesson Introduction (1 minute)

In Roblox game development, an environment refers to the virtual setting or surroundings in which the gameplay occurs. It includes all elements that define the look, feel, and atmosphere of the game's world.

Studio provides developers with a suite of built-in tools to craft and adjust their environments. These tools include terrain editing, part placement, lighting settings, scripting, and asset importing.

We will use Studio's terrain tools to help you create realistic landscapes like oceans, mountains, rivers, and canyons.

Guided Instruction (30 minutes)

Designing for Everyone: Accessibility and Inclusion

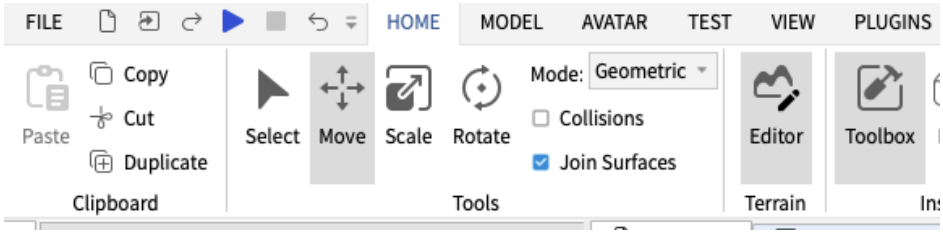
Creating a great game and a winning theme means making it enjoyable and accessible for a wide range of players, including those with disabilities.

- What is Accessibility? Explain accessibility in the context of game design – making games usable by people with different abilities (e.g., visual impairments, hearing impairments, motor difficulties, cognitive differences).
- Why is Accessibility Important? Discuss how designing with accessibility in mind makes games better for everyone and ensures that more people can enjoy their creations.
- Accessibility Features in Games: Brainstorm or research common accessibility features in games (e.g., adjustable text size, color contrast options, remappable controls, volume controls for different sounds, visual cues for sound-only events, clear instructions).
- Inclusion: Discuss creating a welcoming and inclusive environment within their game where all players feel safe and respected, regardless of their background or identity. This ties back to Community Standards.
- Activity/Project Idea: Challenge students to identify one accessibility feature they could add to their game or design a simple game concept that focuses on accessibility or inclusion.

Teaching Tip
Learn more about Roblox accessibility guidelines here: https://create.roblox.com/docs/production/publishing/accessibility.

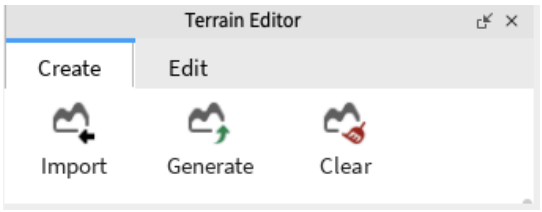
Creating your Theme: Opening the Terrain Tools

- The terrain tools can be opened from the Home tab.

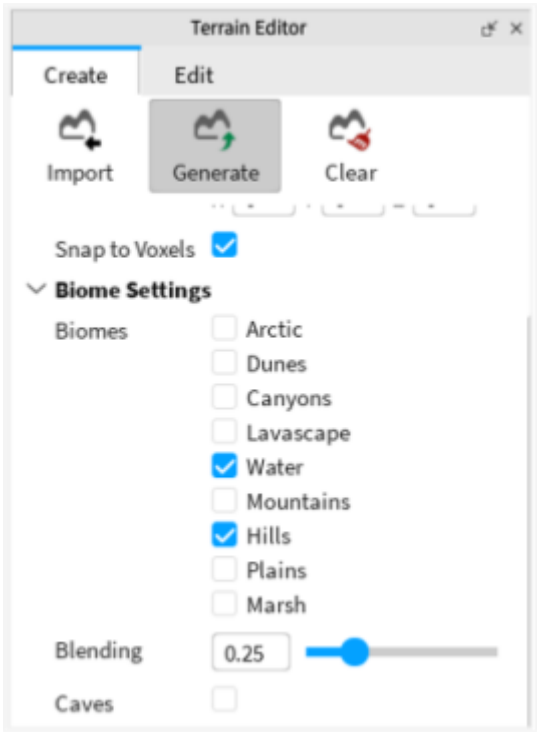


Create Terrain

1. In the **Terrain Editor** window, click **Generate**.

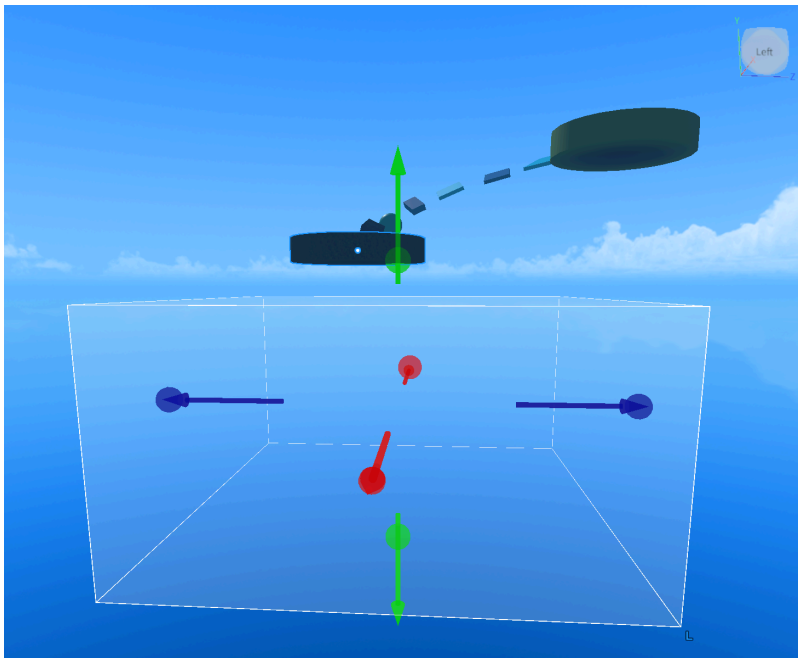


2. Under Biomes, select two types of environments you would like to see.

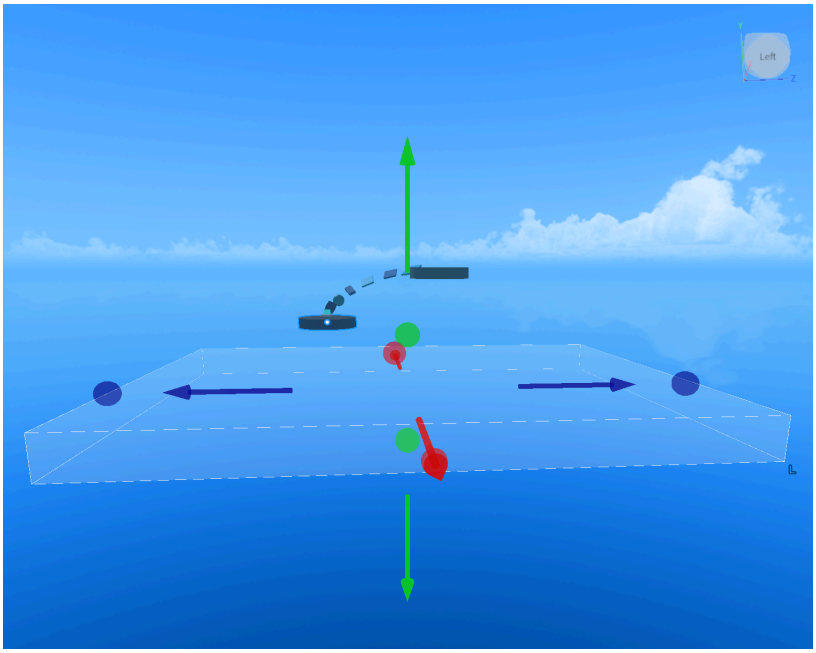


3. A white square will appear. If you cannot see it:
    a. Press F to center your camera.

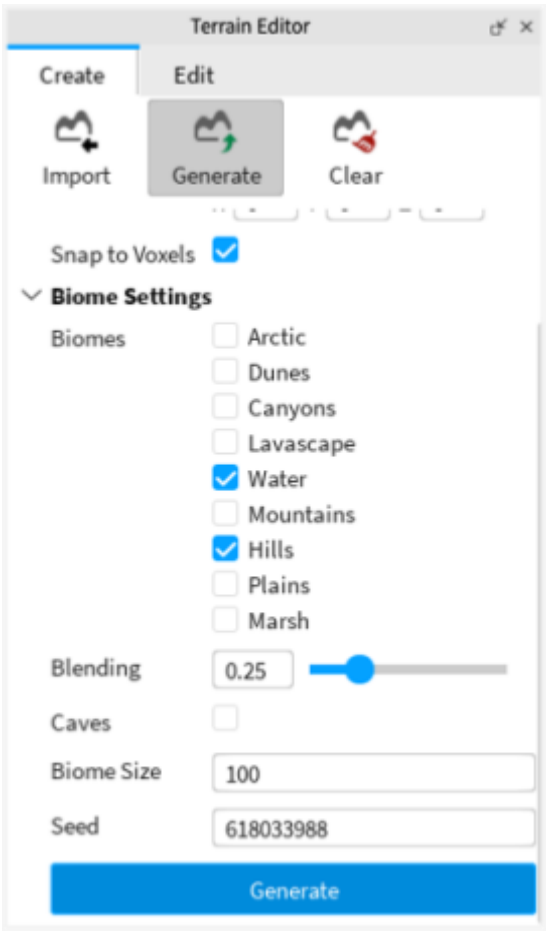
b. Use your scroll wheel to zoom out until you can see your whole obby.



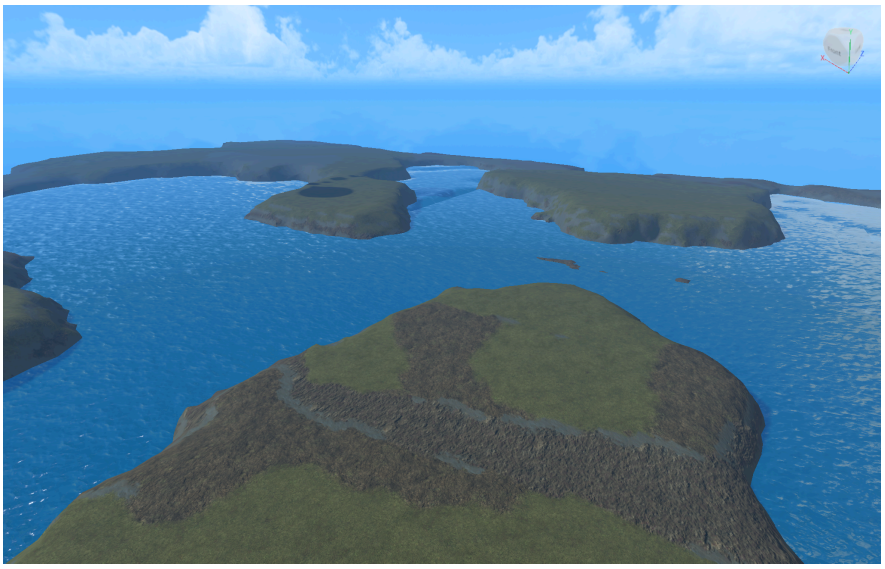
4. Use the circles to scale the 3D rectangle (prism) to the size of the terrain you want. Make sure your obby stays above the 3D rectangle.



5. Change biome size to 25. A value of 25 provides a balance between detail and overall biome structure.



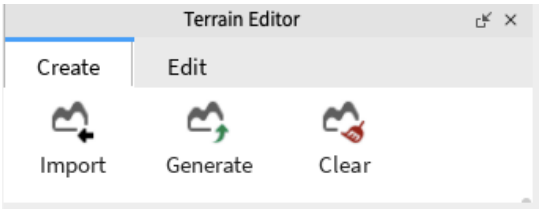
6. Click **Generate** when ready. If you do not like the results, undo and try again.



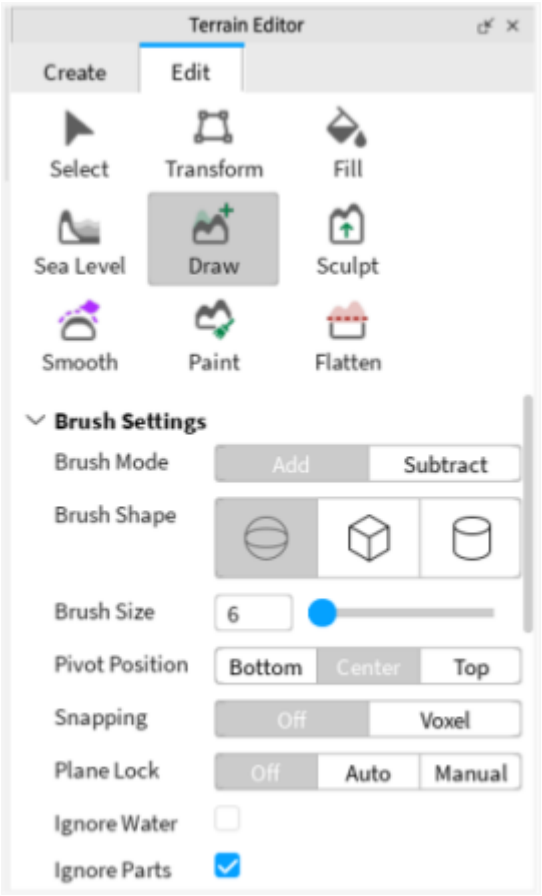
Editing the Terrain

You can make terrain changes with the Edit tools.

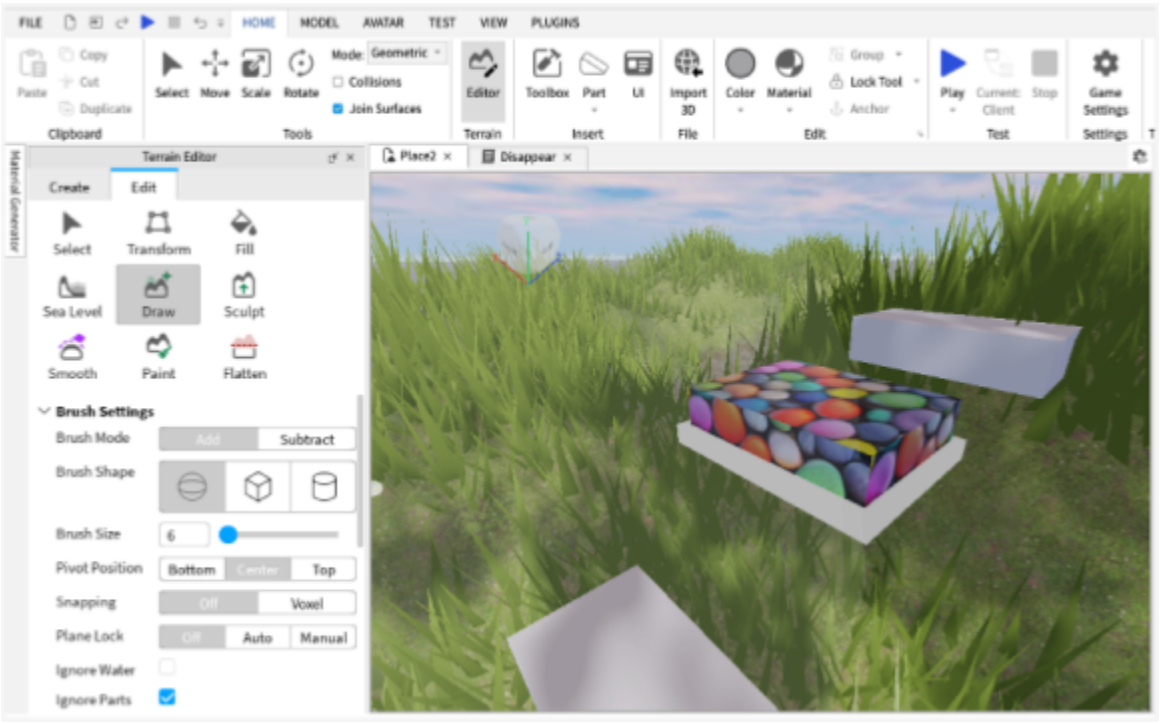
1. Click the **Edit** tab.



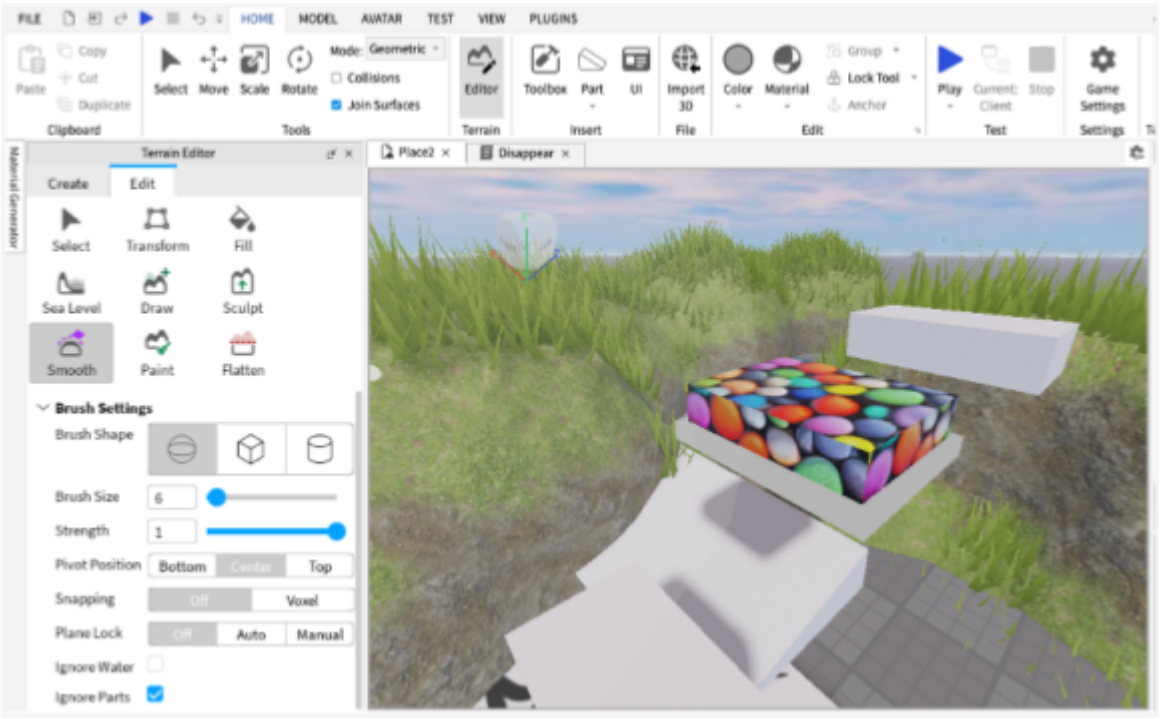
2. Select the **Draw** tool.



3. Zoom in on the terrain and use **Add** to add hills, and **Subtract** to take away land.



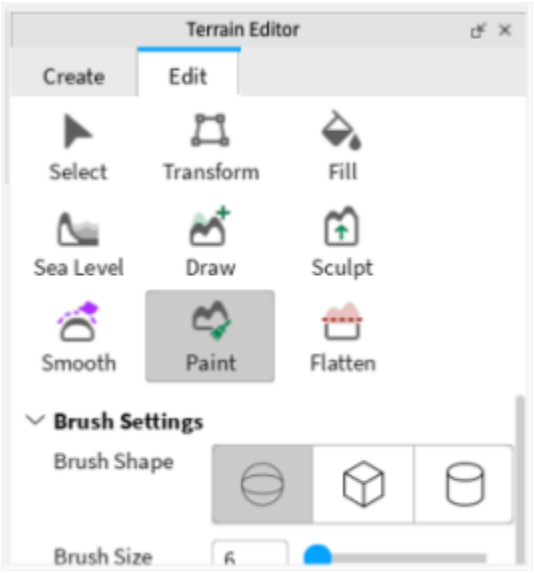
4. Select the **Smooth** tool to even out odd bumps using the brush on the island's surface to smooth out rough areas.



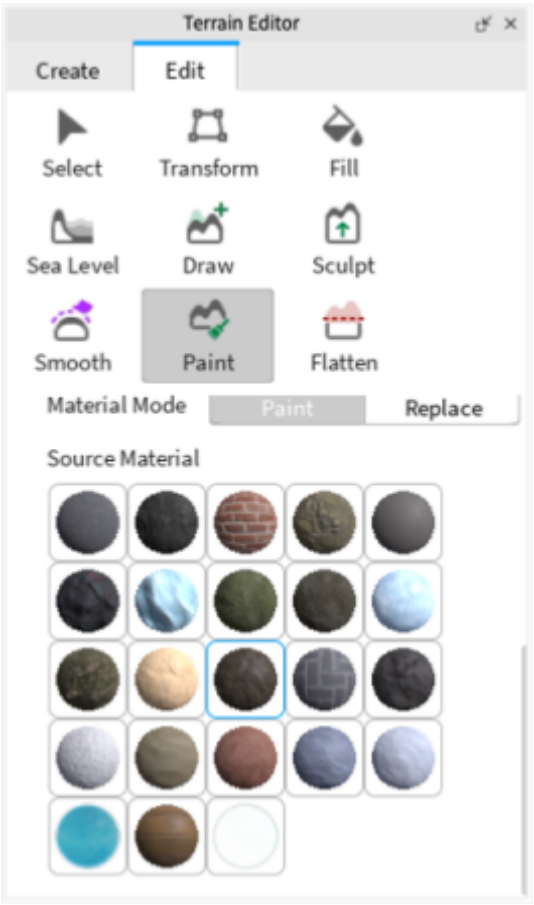
Painting Terrain

To create a more natural looking island, you can use the **Paint** tool to change the type of terrain.

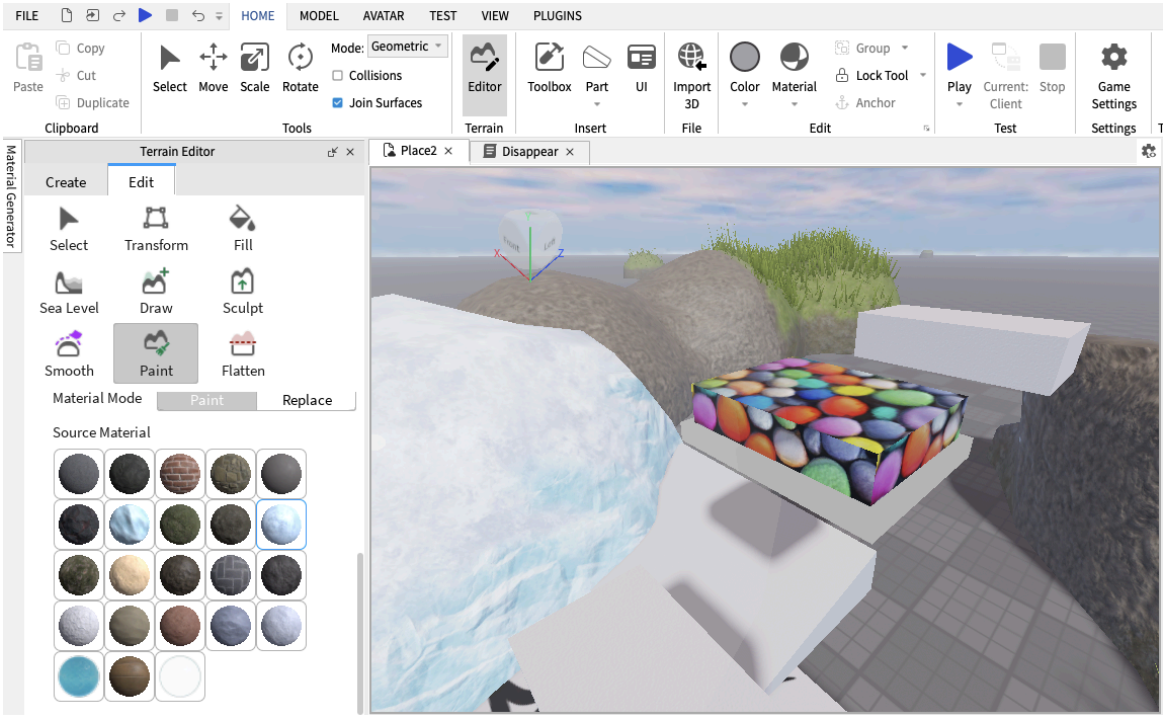
1. Select the **Paint** tool.



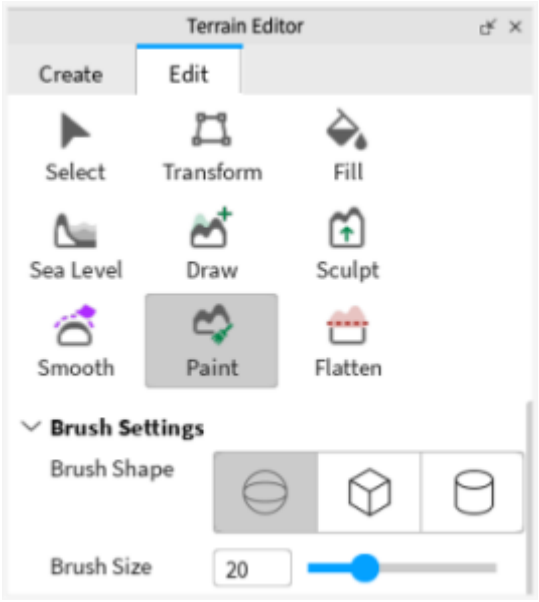
2. Select the material you wish to use in painting the terrain.



3. Use different types of materials to add more variety to your terrain. In this example we have added lava to a small mountain.



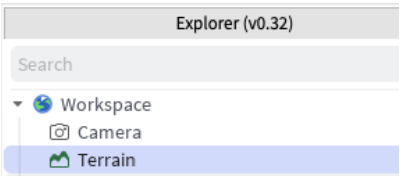
4. Change the size of the brush with the slider value at the bottom of the menu to allow you to paint large areas or small details.



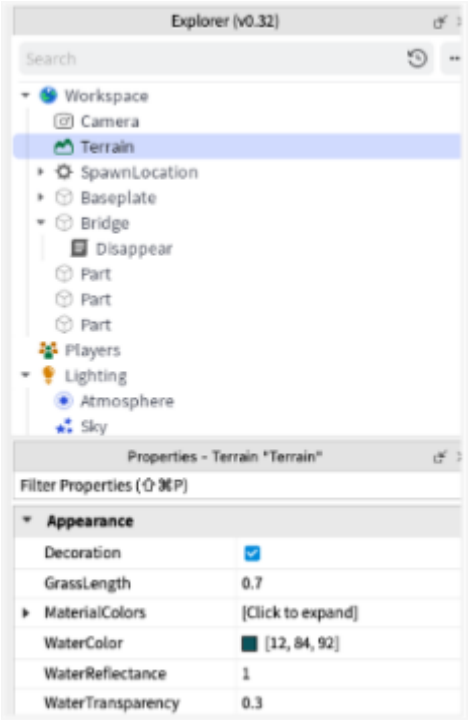
Customizing Appearance

Each terrain **material** has a default color, but these can be customized to fit a specific theme like an arctic wasteland or desert. You can also enable **decorations** like moving grass.

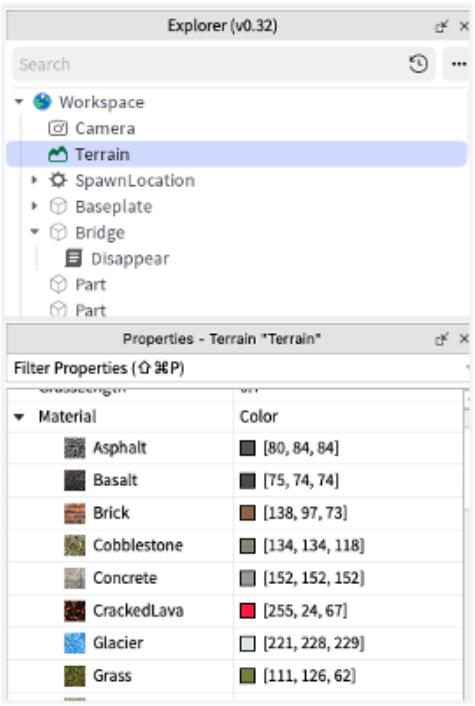
1. In the Explorer window, select the **Terrain** object within **Workspace**.



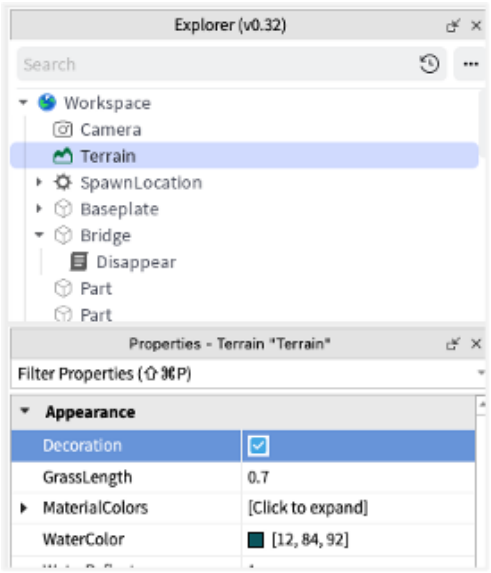
2. In the Properties window, expand the **MaterialColors** branch.



3. Click the squares to choose a new color for that material.



4. Finally, all terrain of the **Grass** material can be decorated with animated blades of grass by toggling on **Decoration**.



Assistant - Make Players Respawn if They Fall (5 minutes)

Now when you playtest the obby, you fall harmlessly to the ground.

1. See if you can Assistant to make it so players restart at the spawn point if they fall.
2. Write down what requests you try, and if they worked or not.

Teaching Tip
Wait a couple of minutes, and then ask students:

- Did anyone type in a successful request?
- What requests didn't work? What happened?
- What requests did work?

Code Concept review
Make the following Assistant request: “Add a new script with a function named respawn that makes players respawn if they touch terrain.”

Type in one of the successful requests and see if students can identify any of the following:

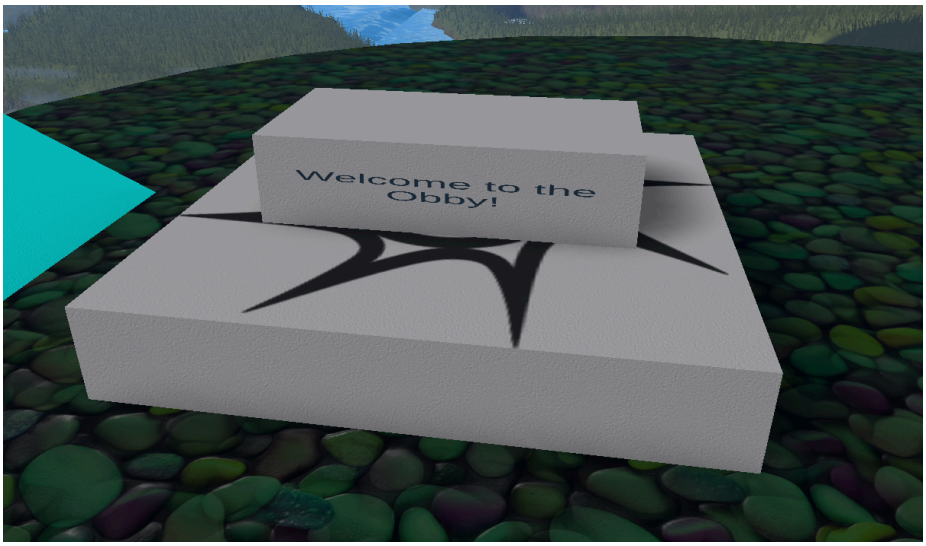
- Variables
- Properties
- Functions

Adding Props (10 minutes)

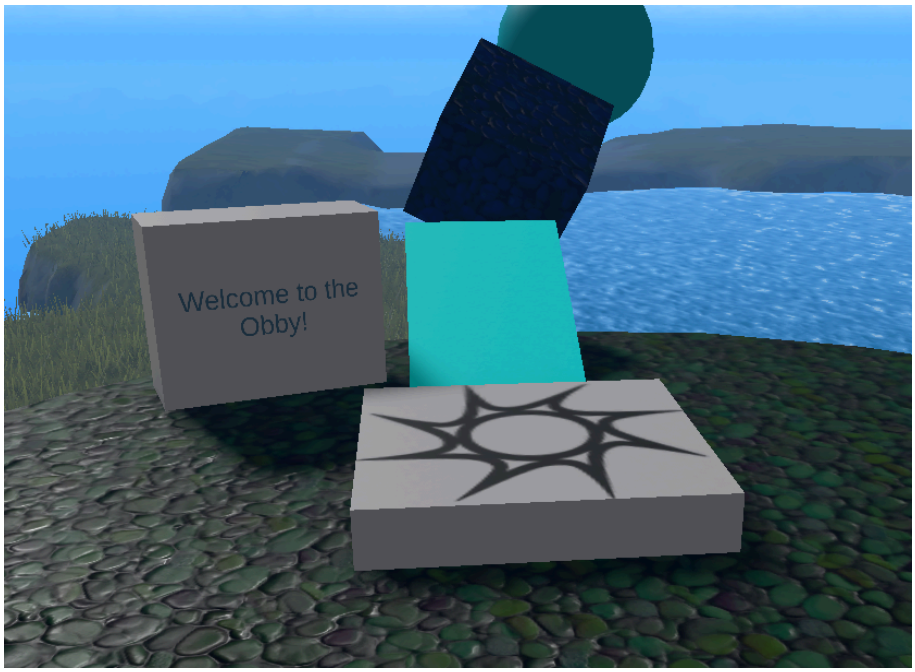
To add a label to the face of a part:

1. In Explorer, select **SpawnLocation** and Press **F** to quickly find the start of your obby.
2. Add a new part named “Sign”
3. Ask Assistant: Add a textlabel to Sign

4. Look at Sign from all sides and find the text

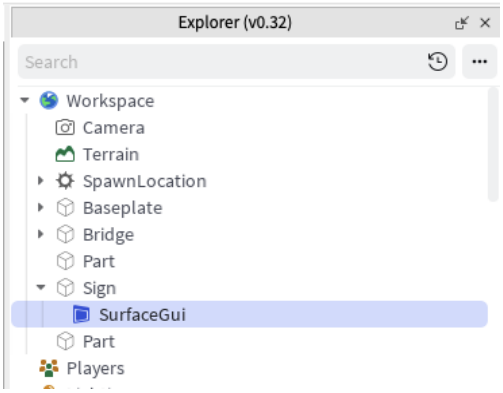


5. Scale and Rotate Sign so that it faces players when they spawn.

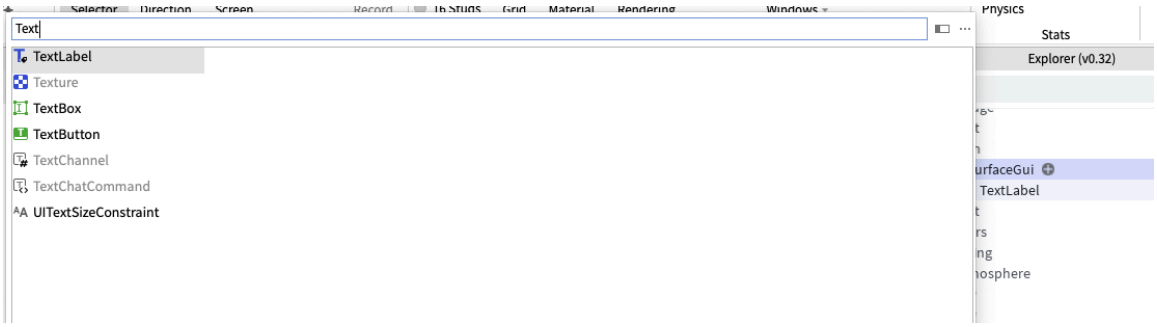


Change the Text

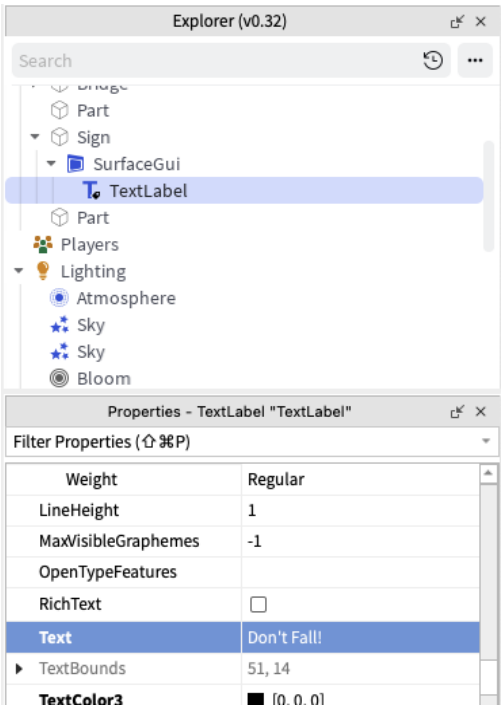
1. Under Sign, you will find an object called **SurfaceGui**. Click the arrow to expand it. You should see an object named **TextLabel**.



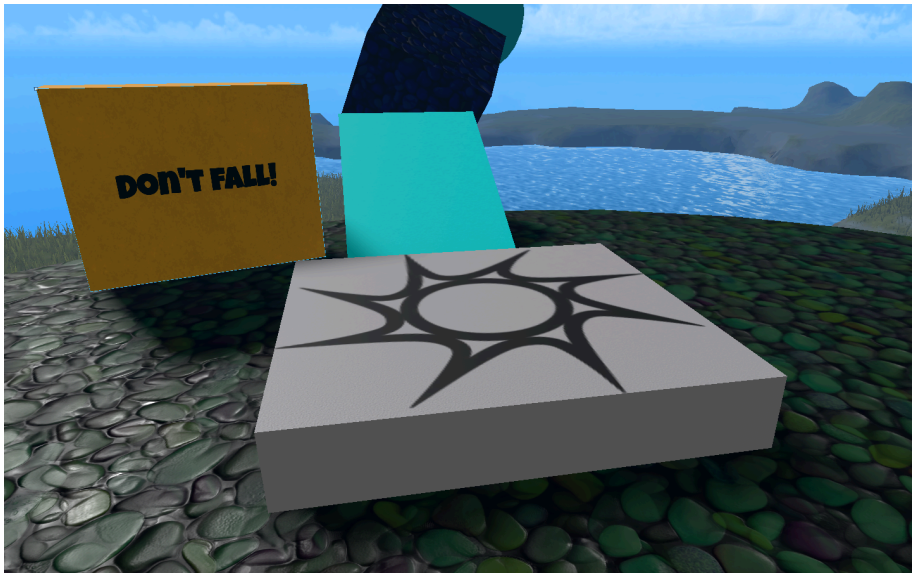
2. Select **TextLabel** to see all its Properties. Scroll down and find Text.



3. Click the field next to text to change the words on the sign.



4. Experiment with the other properties and customize the sign how you would like.



Lesson Recap (2-5 minutes)

Discussion Questions

- What are different environments in the world?
- What's a biome? (kids often know this because of Minecraft)

Next Lesson We Will:

- Code from scratch and learn new coding principles.

Module 6: Coding If - Then Statements and For Loops (60 minutes)

Description

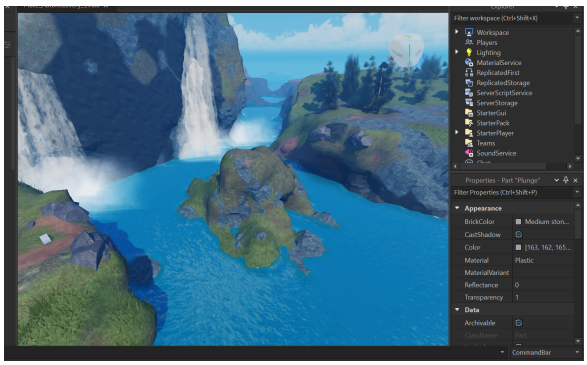
Using if/then statements and loops to modify parts.

Objectives

- Add Color and Materials
- Provide effective peer feedback
- Understand and develop a theme for the obby.
- Use Assistant to create new materials.

Optional Resources

- Studio UI Reference
- Optional: Theme Worksheet
- Studio Cheatsheet



Example of a themed environment

Lesson Introduction (7 minutes)

Introduction to if statements

In experiences, there are often many cause-and-effect relationships. For example:

- If a player scores 10 points, they win the game.
- If a player earns a power-up, they can run faster.
- If a player says "Happy Birthday" in chat, then confetti rains.

Scripts use conditional statements to handle these types of situations. **Conditional statements** are lines of code that will only run if a certain set of conditions are true.

One type of conditional statement is an **if/then statement**. This statement checks to see *if* something is *true* and if it is true, *then* it will run a specific set of instructions.

In Lua, the syntax pattern for if statements looks like this:

```
if "something happens" then

    -- Make something else happen

end
```

Teaching Tip
Have students create a list of some if/then statements they would like to use in their obbies.

In this lesson we will:

- Share information about parent controls
- Revisit variables and functions
- Practice using if/then statements
- Create a trap part
- Introduce For Loops

Guided Practice (45 minutes)

Sharing with Parents and Understanding Parental Controls

Educators can play a vital role in informing parents and caregivers about Roblox safety features.

- Open Communication: Encourage students to talk to their parents about the games they are creating and playing.
- Parental Controls: Provide information to share with parents about the range of parental controls available:
    - Account Linking: How parents can link their account to their child's to manage settings.
    - Content Restrictions: Setting maturity levels for playable experiences.
    - Communication Settings: Limiting who can chat with their child (e.g., Friends only, No one).
    - Spending Controls: Setting monthly limits on Robux purchases.
    - Screen Time: Monitoring and setting limits on time spent on the platform.

    - Friends List Management: Viewing and managing who their child is friends with, including blocking users.
    - Blocked Experiences: Blocking specific games.
- Parent/Caregiver Guide: Share resources like the "Parent & Caregiver Guide to Roblox" and "A Guide for Parents: Emotional Safety for Teens on Roblox" which offer advice on understanding Roblox, using controls, and talking to teens about emotional well-being online, including cyberbullying.
- Discussion: Facilitate a discussion with students about the importance of involving their parents in their online activities and using the available safety tools.

Teaching Tip
When educating parents and students about Roblox's parental controls, it is important to understand and communicate how parent and child accounts link together on Roblox. Learn more about parent controls here: https://en.help.roblox.com/hc/en-us/articles/30428321333140-Parents-How-to Link-Your-Child-s-Account.

If/Then Statements

When you create a new script, it includes a print function at the top of the script editor by default. Print functions display text on the screen. The print function is one of the first functions many people learn, and you will use it often. This code will make "Hello world!" appear in a special window called "Output."

```
print("Hello world!")
```

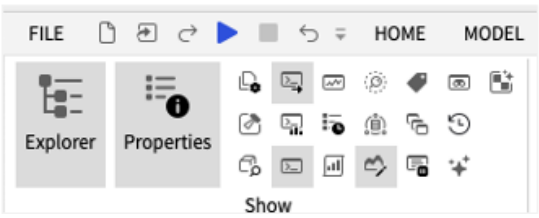
To find the script next time you open up Roblox Studio, click on the name of the script above the game editor, or double-click the script's name in Explorer.

Test output

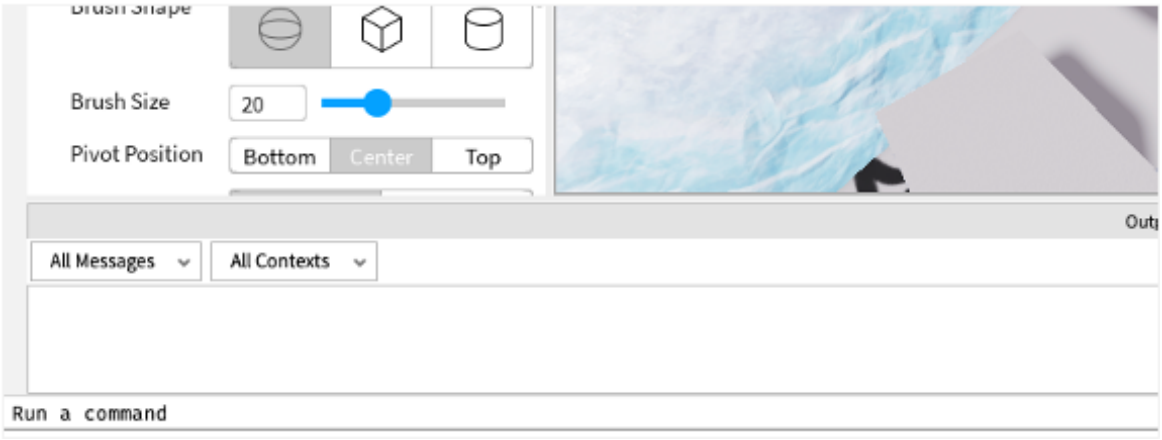
You can see the result of running the default code with the Output window. **The Output window** is typically at the bottom of Roblox Studio, but if you have never used it before, you may need to enable it.

You will need the Output window throughout this module, so now is a good time to enable it if you have not already.

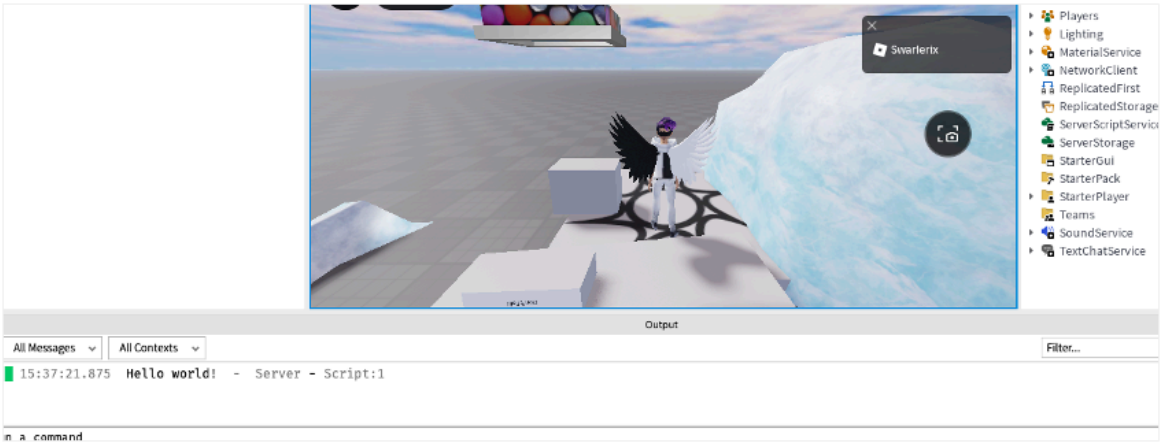
1. Select the **View** menu tab.
2. Click **Output**.



The window will appear at the bottom of Roblox Studio.



3. To test the script, click **Play**. Hello world! will show up in the Output window.



4. Click **Stop** to end the playtest. You can now return to the Script tab.

If statement practice

These steps show how to create a script that changes a part's color if a statement is true.

1. Create a new part named LieDetector.
2. Add a script to the LieDetector part named: "FactCheckerScript."

Comments

You may have noticed some of the scripts Assistant created for you included notes about the script.

```
-- This is a comment. Comments make your code easier to understand.
```

1. Add your own comment to the top of the script.

```
-- Changes the part if a condition is true
```

2. Add local factChecker = script.Parent

```
-- Changes the part if a condition is true

local factChecker = script.Parent
```

> **Teaching Tip**
> Encourage students to add comments to their code to help them remember what the code generates and to make it easier to troubleshoot when things do not work as expected.

Format "if" statements

Conditions come in many forms but are often simple statements like math equations.

For example, ***if*** 1+1 equals 2, ***then*** run some code.

Like ordinary math equations, conditionals can use operators such as plus + or less than < to evaluate statements.

One particular operator to be aware of is `==`; it stands for "is equal to." So the statement `2 + 2 == 4` can be read as "two plus two is equal to four". Be very careful not to mix it up with `=`, which assigns new values to objects like variables.

1. In the script, type `if then`, and press Enter to autocomplete the conditional.

   The keyword `then` will be underlined because the code is incomplete.

```
-- Changes the part if a condition is true

local factChecker = script.Parent

if then

    -- empty code

end
```

2. After the keyword `if`, type a true statement such as `3 + 3 == 6`.

```
if 3 + 3 == 6 then

-- empty code

end
```

3. Within the conditional statement, reference the part you named LieDetector and change the part's Color property to green.

```
if 3 + 3 == 6 then

      factChecker.Color = Color3.fromRGB(0, 255, 0)

end
```

   Indenting the lines of code within the conditional makes it easier to see where the code starts and stops.

4. Test your code. If three plus three is equal to six, the part will turn green.

Check a false condition

Now, purposely change the statement to see what happens when the math equation is false.

1. In the if statement, change the equation to something inaccurate, such as 3 + 3 >= 10.

```
if  3 + 3 >= 10 then

      factChecker.Color = Color3.fromRGB(0, 255, 0)

end
```

2. Test your code now. The part should not turn green for a false statement.

Math operators

The table below lists some common Lua operators. More information about operators can be found on Luau Operators.

Symbol	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division

Comparison operators

- == Is equal to.
- ~= Is not equal to.
- < or > are used for less or greater than, respectively.
- <= or >= are used for less/greater than or equal to, respectively.

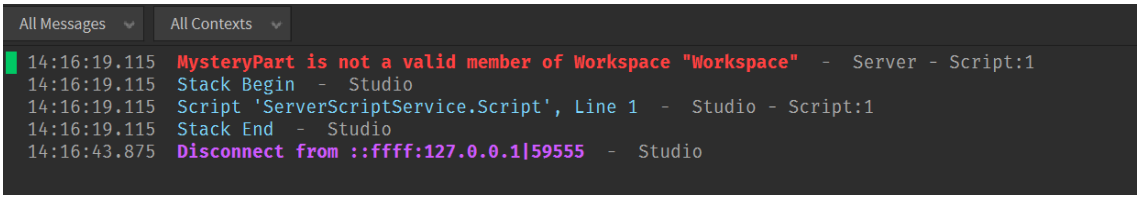
Using If Statements to Check Your Code

Conditional statements are also used to evaluate the status of properties and variables. The following steps check whether a variable was successfully assigned a value.

1. **Delete all of the code** and copy the following snippet into the script.

```
local mysteryPart = workspace.MysteryPart

-- Evaluates as true if MysteryPart was successfully assigned

if mysteryPart then

	mysteryPart.Color = Color3.fromRGB(0, 255, 0)

end
```

2. Test it, and a new error appears in Output. The error happened because Studio could not find MysteryPart inside of Workspace.



3. Now fix the script. Change the first line to local mysteryPart = workspace.LieDetector

```
local mysteryPart = workspace.LieDetector

-- Evaluates as true if MysteryPart was successfully assigned

if mysteryPart then

	mysteryPart.Color = Color3.fromRGB(0, 255, 0)

end
```

4. Test the code, and if MysteryPart has the default transparency of 0, it will become ghostly while LieDetector turns green.

Output Window vs. Assistant

Assistant can also take requests such as "Fix this script", but it can provide extra information, or your request could generate an outcome that you did not intend.

For this reason, it can often be faster to check for errors in the Output Window.

Checking for Multiple Conditions

What if you want a line of code to only run if multiple conditions are true? Perhaps you want to make sure that a player found and pressed multiple buttons before they can go through a door.

To check multiple conditions, use the keyword and to separate each condition.

Code Example

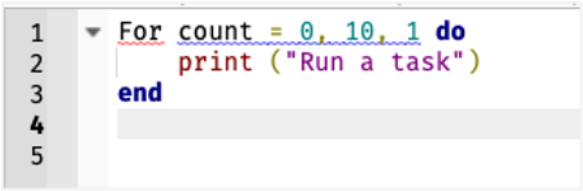
```
-- Checks to see if both 3+3 is equal to 6 and if 2 is greater than 1
-- Note that only the first condition uses ==
if 3+3 == 6 and 2 > 1 then
      print("both are true")
end
```

For Loops (10 minutes)

There are different ways to make code run again and again. If you want the code to only run a certain amount of times, use a “for loop.” This section will cover the logic behind for loops and demonstrate some practical examples, such as coding a countdown.

For loops use three values to control how many times they run: a control variable, an end value, and an increment value.

Starting from the value of the control variable, the for loops will either count up or down each time it runs code inside the loop until it passes the end value. Positive increment values count up, and negative increment values count down.



Teaching Tip

Different computer languages might use other terminology with for loops. For instance, in C# or JavaScript, the end value might be called the condition.

Code a countdown

To see how a for loop works, use these steps to code a for loop that starts at 10 and counts down to 0, one number at a time. Every time the loop runs, it will print the current value inside the control variable.

1. In ServerScriptService, create a new script named PracticeLoop. In the script, start by typing the keyword for.

```
for
```

2. Create a control variable named count and set a starting value of 10.

```
for count = 10
```

3. Set the end value to 0, by typing , 0. Make sure to include a comma to separate the values.

```
for count = 10, 0
```

4. Create an increment value of -1 by typing , -1. After the loop has finished its action, it will add the increment value to the control variable, count. Because the increment is negative, it will be subtracted when added to the control variable.

Teaching Tip
Negative numbers may be a challenging or new concept to younger students. Negative numbers can simply be expressed as "subtraction". In this example, students start with "10" and take away "1" for each loop of the code.

```
for count = 10, 0, -1
```

5. A for loop does not explicitly need an increment. Without one, the loop will, by default, add 1 after each loop.
    a. To finish the for loop, type do and press Enter to add end. Any code typed between do and end will run each time the loop repeats.

```
for count = 10, 0, -1 do

end
```

6. Inside the loop, create a countdown by printing the value of the control variable, count, and then delay the script with a wait function.

```
for count = 10, 0, -1 do

	-- Prints the current number the for loop is on

	print(count)

	-- Wait 1 second

task.wait(1)

end
```

7. Run the project and watch the Output window to see the for loop.

Troubleshooting tips

At this point, if the loop does not work as intended, try one of the following:

- Check that you have two commas separating the numbers in your for loop. Having extra or missing commas will make the loop not start.
- If the for loop prints all at once, make sure that there is a wait function that uses at least 1 second of wait time.

How For Loops Work

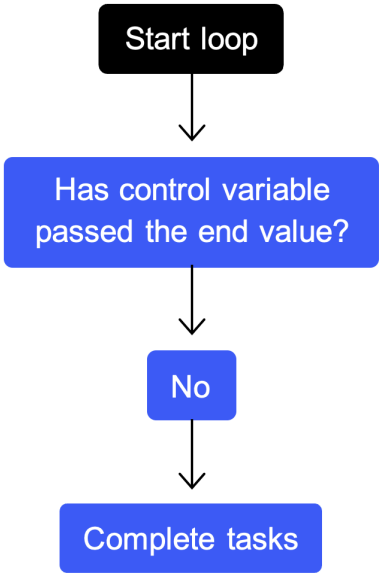
Notice that the loop will print out the current value of count each time it goes through an iteration.

An **iteration** is the complete process of checking the control value, running code, and updating the increment value. Because the control variable starts at 0 and has to go past 10, the loop will go through 11 iterations before stopping.

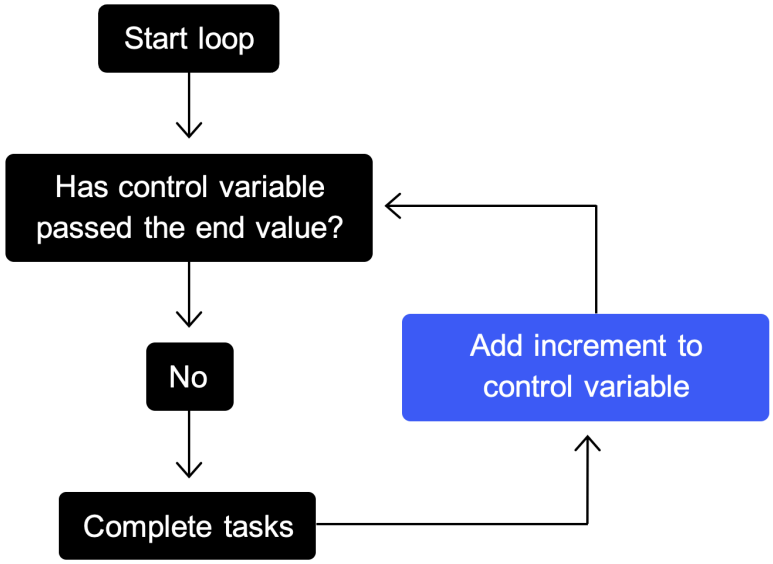
Steps in a for loop

To understand for loops, it helps to see a flow chart diagram showing the logic of how they progress.

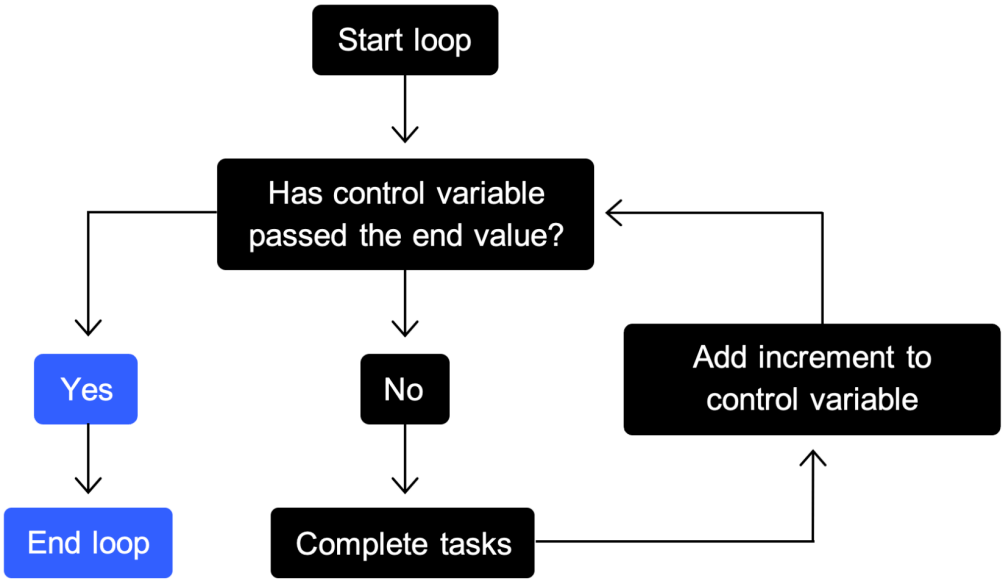
- First, the for loop compares the control variable with the end value.



- After running the code, the increment value is added to the control variable. The loop then checks the control variable and starts over.



- Once the control variable passes the end value, the loop will stop. For example, if a loop has an end value of 10, once the control variable has passed 10, the for loop will stop.



Different for loop examples

Changing the three values of a for loop will change how the loop functions. Below are different examples of for loops with different start, end, and increment values. Try putting them into scripts and see what happens.

Count up by one

```
for count = 0, 5, 1 do

   print(count)

task.wait(1)

end
```

Count up in even numbers

```
for count = 0, 10, 2 do

print(count)

task.wait(1)

end
```

If for loops do not run at all

If the control variable starts out beyond the end value, like in the example below, the for loop will not run at all.

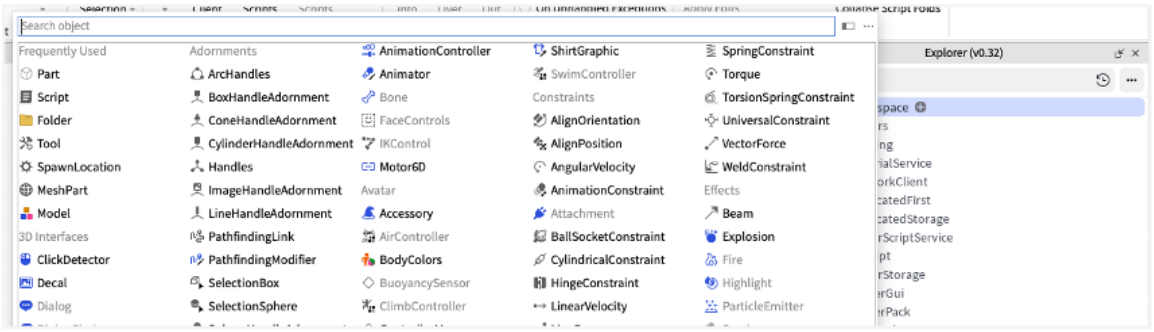
```
for count = 10, 0, 1 do

print(count)

task.wait(1)

end
```

In this case, the for loop is counting up and checking if count is greater than 0. When the for loop makes its first check, it sees that 10 is greater than 0, and so it will stop the loop without printing anything.

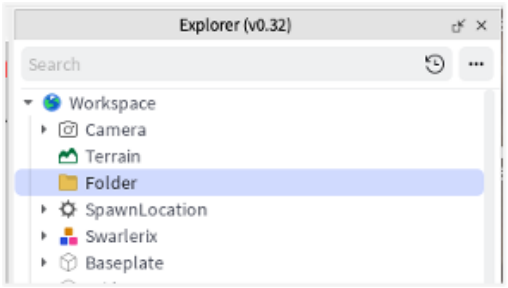
Use For Loops to Modify Groups of Parts

Now use for loops to change parts within a folder using Assistant.

1. Hover over Workspace and click the + button.



2. Select Folder



3. Rename the folder Wrong
4. Right-click Wrong and select Insert Part. Duplicate (Ctrl/Cmd +D) the part twice. Move the parts next to the spawn where they will be easy to see.

Using Assistant

You can now use your knowledge of loops to make a request of Assistant. Since this script will affect multiple parts, you will specifically ask Assistant to create a new script in ServerScriptService.

1. Make the following request:

```
In ServerScriptService, add a new script named AnswerManager.  For
```

```
all children in Wrong, If a player touches the child then make
players respawn
```

2. Playtest and see if all parts make the player respawn.

Example of Returned Code

```lua
local WrongFolder = game.Workspace:WaitForChild("Wrong")

local function respawn(character)
    local player = game.Players:GetPlayerFromCharacter(character)
    if player then
        player:LoadCharacter()
    end
end

for _, child in WrongFolder:GetChildren() do
    if child:IsA("BasePart") then
        child.Touched:Connect(function(hit)
            local character = hit.Parent
            local humanoid = character:FindFirstChild("Humanoid")
            if humanoid then
                respawn(character)
            end
        end)
    end
end
```

Lesson Recap (3 minutes)

Discussion Questions:

- What's an example of an if/then statement?
- What's something you can do with a for loop compared to a while loop?

In the next lesson we will:

- Use everything you have learned to make a quiz obby where players have to answer questions by jumping on the platform with the correct answer.
- Divide into groups to create shared projects.
- Each project will need to have a specific theme

Module 7: Working as a Group (50 minutes)

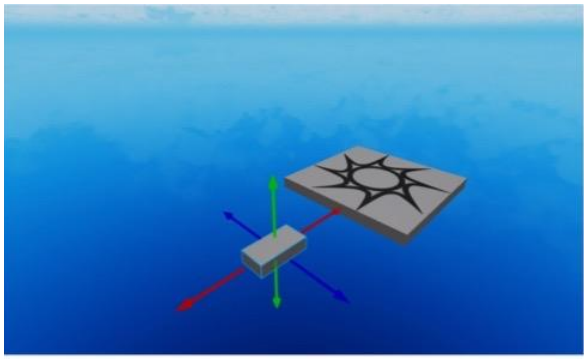
Description
Introduce the final project and break up into groups to plan and work.

Objectives

- Use Team Create to create a shared project
- Mimic a preproduction phase by creating a plan for the final project

Optional Resources

- Studio UI Reference
- Studio Hotkeys



Create a plan for the final project

Lesson Introduction (15 minutes)

1. Divide your class into groups of 2 - 3 students.
2. Once you have each group together, have the students designate the team lead.
3. The team lead will be responsible for inviting everyone else into the project. They will be group project owners.
4. The team lead should invite all group members as “friends” on Roblox. This is important for using the Team Create feature in Studio.

Teaching Tip
When using "Team Create" in Studio, the team lead must invite the group members to be their "friend" on Roblox to co-create. Here is more about Roblox friends. The team lead will own the project and control the permissions and settings for the project.

If you desire shared ownership of the project, you may create a Roblox Community and assign roles to members that manage their permissions for experience development and access. It is important to note that while Team Create is free, creating communities currently costs 100 Robux. You can learn more about Robux here: https://en.help.roblox.com/hc/en-us/articles/203313200-Ways-to-Get-Robux.

Project Requirements: Trivia Themed Obbies in a Safe and Civil Environment

- In trivia obbies, players see a question on a sign and must jump to the part which displays the correct answer.
- For this project, teams must create a trivia themed obby over the next few classes which:
    - Has an easy to identify theme such as:
        - Ocean conservation
        - Tips for tourists visiting your town
        - A favorite national park
    - Has at least 3 - 5 questions for players to answer
    - Has at least 4 - 5 types of player/object interactions such as:
        - if a player touches a part, then:
            - The part will turn green
            - The part will disappear
            - The player will be forced to respawn
    - Feedback from at least 3 people not on your team who reviewed both your game design concept and performed at least one playtest each.

In addition to the existing requirements, the project must also demonstrate an understanding of Roblox safety and civility. Students should:

- Integrate Safety/Civility Theme: Consider incorporating a safety or civility theme into the trivia game obby's concept or design (e.g., questions about recognizing

scams, ways to promote kindness, or questions about respectful interactions).
- Design for Positive Interaction: Include elements in their game design that encourage positive player interaction and discourage negative behavior (e.g., clear instructions, easy-to-understand goals, cooperative elements).
- Content Maturity Consideration: Be prepared to explain what content maturity level their game would fall under and why, based on Roblox guidelines.
- Accessibility Feature (Optional but Recommended): Attempt to include at least one simple accessibility feature in their game (e.g., large text for instructions, high-contrast colors for important elements).
- Presentation Inclusion: In their presentation, students should specifically address:
    - How their game aligns with Roblox Community Standards.
    - Any safety or civility themes or features included in their game.
    - What content maturity level their game is suitable for.
    - Any accessibility considerations made during development.

Guided Practice (10 Minutes)

Form into teams

1. Have each team gather around their team leader. The team leader will go through the steps with the remaining team members helping.
2. Designate a team name
3. Create a new baseplate template
4. Publish the game to Roblox
5. Name the game [TEAMNAME] Game and click Save. The description can be filled out later.
6. Click Collaborate
7. To add team members:
    a. Search for a Roblox username
    b. Make sure access is set to edit
    c. Click Save when done

Join the Shared File

You can find the game you have been invited to by:

- Opening Roblox Studio
- Clicking “Experiences”
- Clicking “Shared with Me”
- When everyone is in, you will be able to see their Roblox usernames within the space, and what they may have selected.

Independent Practice (10 minutes)

Create a brief game design

- As a group, decide what you want the theme of your obby to be and write down:
  - The theme of your obby
  - 3 - 5 trivia questions you want to include
  - At least 4 different if/then statements you want to use in your obby
  - Draw and describe your obby will look like
- Begin working on the environment of your obby.

Lesson Recap (10 minutes)

Discussion Questions:

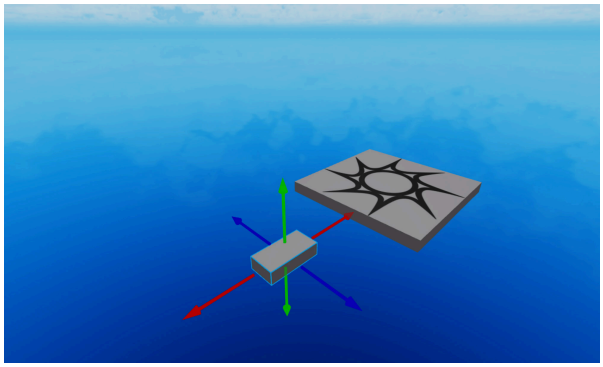
- What was easier about building your obby as a team than it was when you built your first obby individually?
- What was challenging about building a game with a team?
- How did considering safety and civility influence your game design choices?
- What was the most challenging aspect of ensuring your game was safe and respectful for players?
- What is one thing you learned about online safety or civility on Roblox that you didn't know before?
- How can game developers help create a more positive and safe environment on Roblox?

Module 8: Work Session (45 minutes)

Description
Work on final projects and test games from other groups.

Objectives

- Work independently on group projects.



Optional Resources

- Studio UI Reference
- Studio Hotkeys

Lesson Introduction (2 minutes)

- Work on creating as much of the game as possible
- Sources for learning more:
    - Use Assistant in Roblox's Creator Hub to learn more about any game building topic: https://create.roblox.com/docs/assistant
    - Play educational games on Roblox for further inspiration: https://www.roblox.com/charts#/sortName/Learn_And_Explore

Independent Practice (40 minutes)

- 20 minutes - Work on games
- 5 minutes - Play another game and give feedback. If you see something that you really like, ask how they did it.
- 15 minutes - Work on games

Lesson Recap (3 minutes)

- Discussion Topics
    - Did anyone get any really helpful feedback today?
    - Did anyone get any ideas from playing someone else's game?

- Next we will:
    - Customize the game page

Module 9: Game Detail Page (45 minutes)

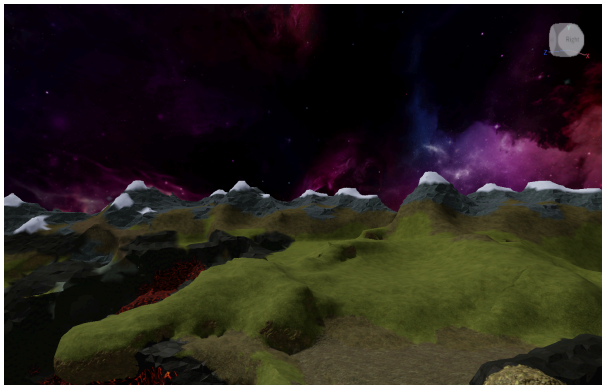
Description
Learn how to customize the game page in order to attract more players

Objectives

- Create a game Icon
- Create a game thumbnail
- Add a description for the game

Optional Resources

- Studio UI Reference
- Studio Hotkeys



Design Marketing Materials for your game

Lesson Introduction (1 minute)

The game page is important because it communicates what your game is about and helps players decide if they want to play it.

In this lesson you will:

- Continue to work on your group project
- Learn how to create custom thumbnails and game icons

Guided Practice (10 Minutes)

Name and Description

First, decide what to name your game and how you will describe it to players.

Experience Title
Up to 50 characters. Using 1 or 2 emojis is acceptable on the Roblox platform. Short statements can also be used to announce new features

Example Title	Your Title
My Aquarium Trivia Game [🪸 New Tropical Fish 🪸]	

Description

Ideally include just a few sentences and bullets of features or recent updates. Visit the game pages of popular educational games to see how these games use their game page: https://www.roblox.com/charts#/sortName/Learn_And_Explore.

⚠ Once you create the and submit your description, it is automatically sent to Roblox for review.

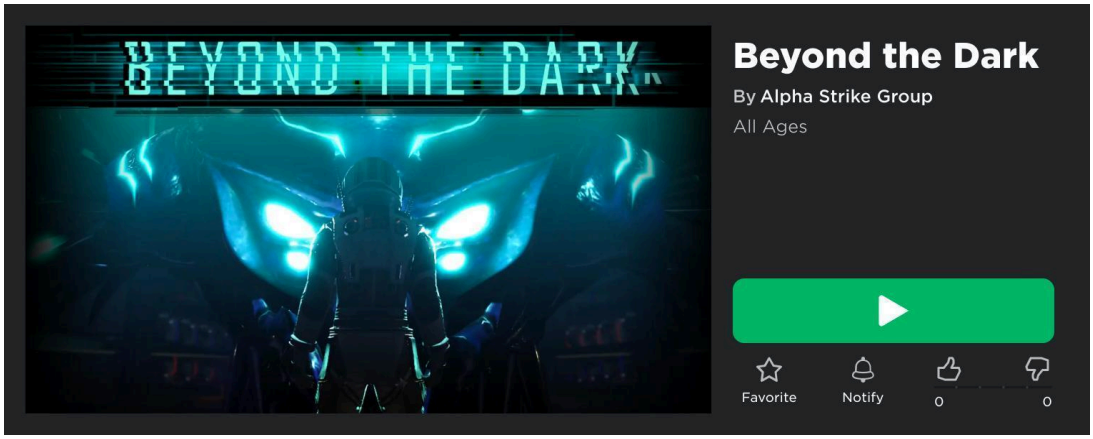
> **Teaching Tip**
> We advise teachers to review game descriptions prior to student submission. The game description provides an excellent opportunity for helping students craft fun and engaging language to describe their game.

Students may update their descriptions as often as they would like. What follows is an example of a description for an aquarium simulator game. You can add your own description in the column provided.

Example Description	Your Description
Start with a tiny desktop aquarium and raise your own fish by answering questions about tropical under sea environments. If you keep your fish happy with their favorite foods, decorations, and water parameters, two fish may become more fish! Eventually you may end up with a whole fish store! 🪸 Set up your tank and decorate it with plants 🐠 Purchase your first fishie friends 🧪 Keep the water just right. Not too hot, not too cold, and just the right minerals! 🐡Sell baby fish to buy improved tanks, decorations, and rare fish	

Thumbnails

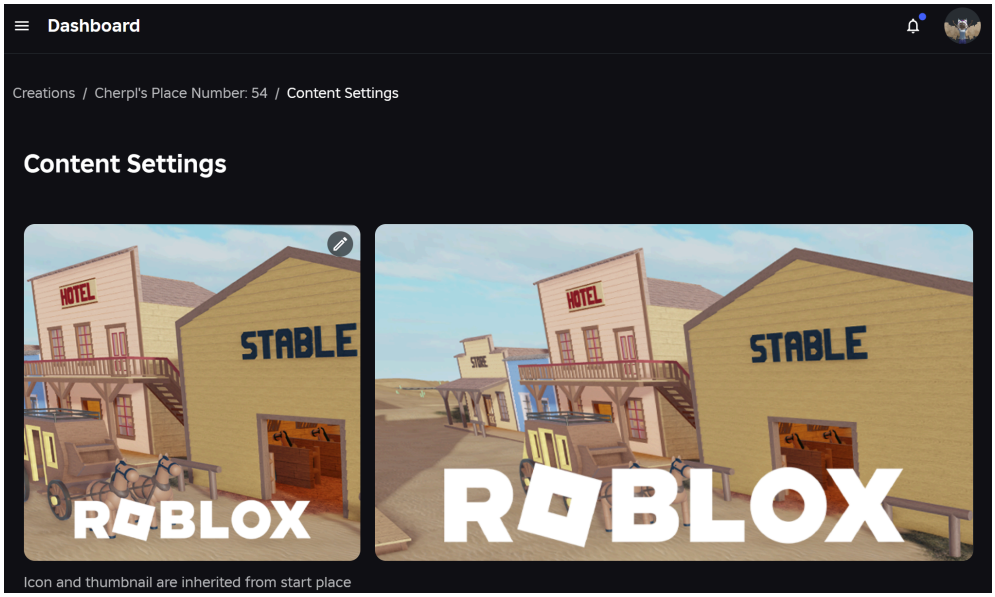
Thumbnails appear on Roblox's Home page and at the top of your experience's detail page, allowing you to showcase your experience's features and announce updates or events.



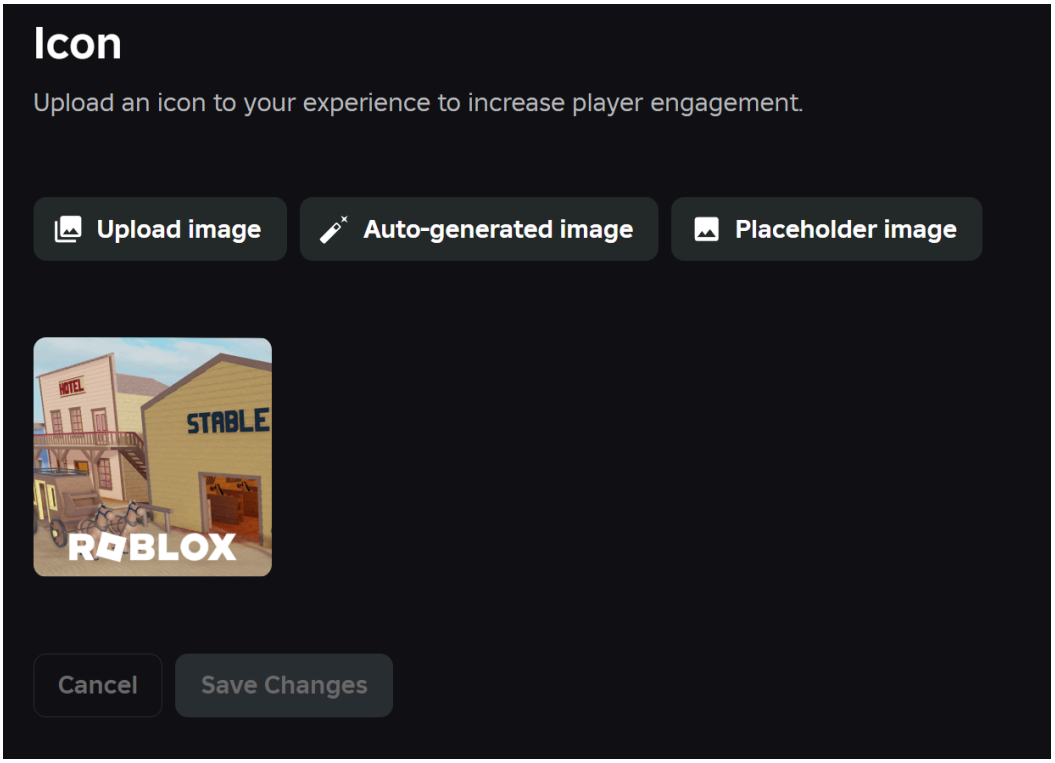
To access your thumbnails:

1. In Studio, ensure your game has been published.
2. Go to File > Game Settings

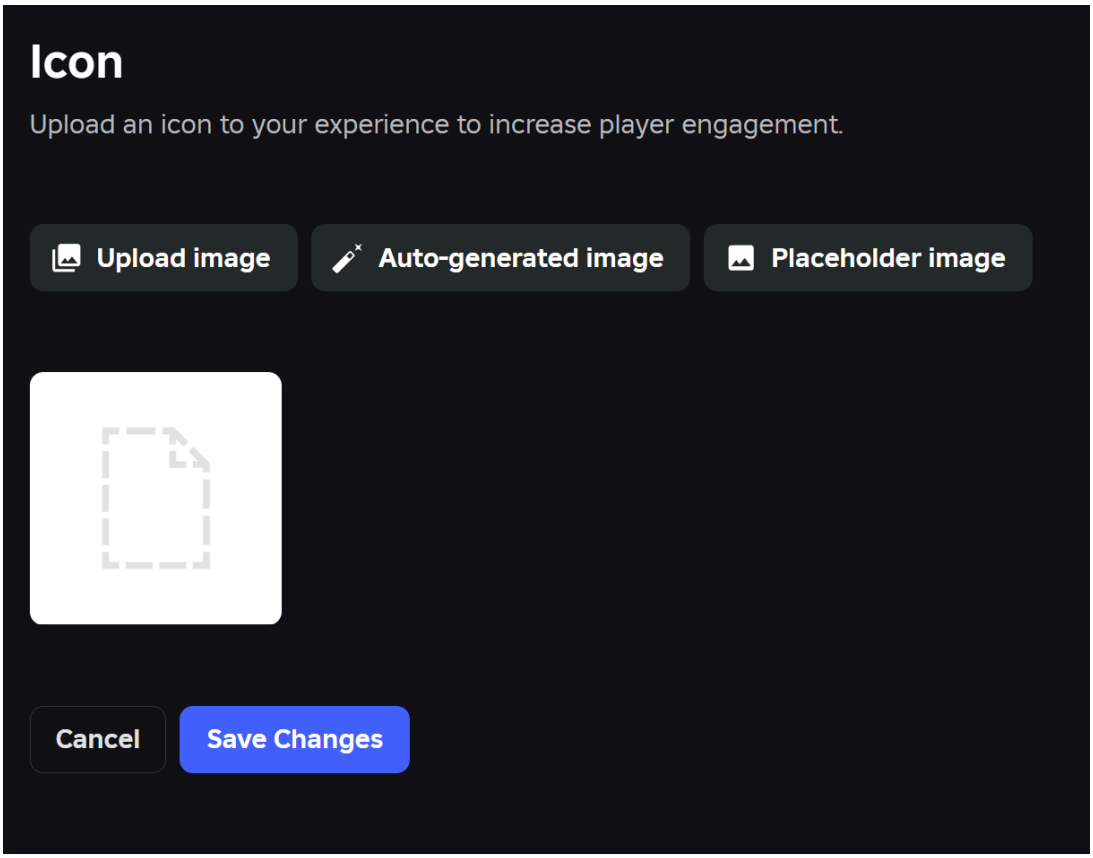
3. Click Manage on Creator Hub. Here you can customize all the information on your game's page.
4. Create a new Icon and Thumbnail by hovering in the top right corner, until you see the pencil icon. Click the Icon.



5. You can upload your own images, use screenshots, or autogenerate images. Click Autogenerated Image.

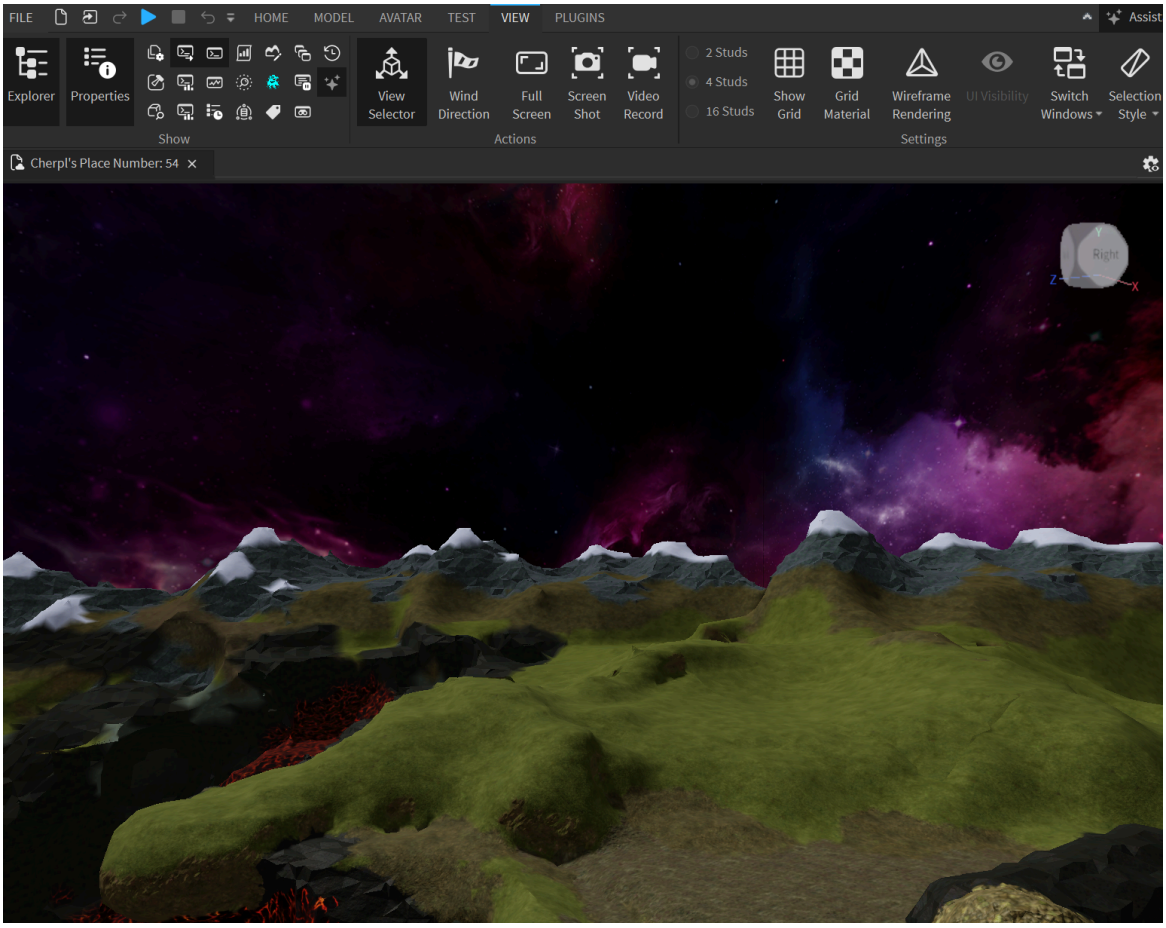


6. A screenshot from your last camera view will be taken. Click Save Changes and check the page later.

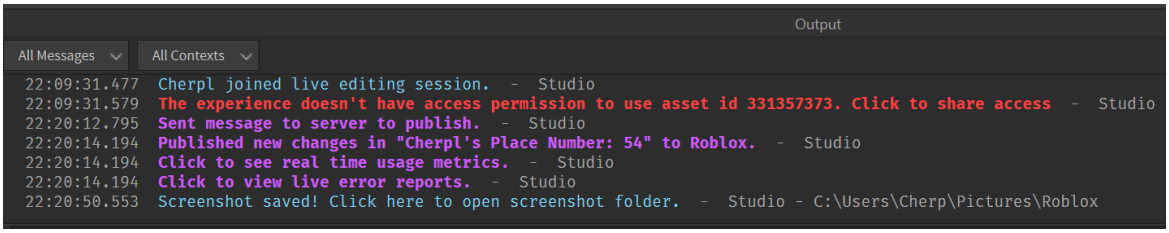


You can also take screenshots and Video from Roblox Studio:

- In the View tab, Select ScreenShot or Video Record.



- Click the link that appears in the Output Window to open the Screenshot folder.



- Drag the desired image someplace where you can easily find it, like the Desktop.
- Click the pencil at the top-right of the Thumbnail.
- Then select Upload Image. Select your image.

Independent Practice (10 Minutes)

- Continue to work on your game
- Make sure to ask someone from another group to test your game and provide feedback on what they liked best, and what they think would make the game better.

Lesson Recap (5 Minutes)

Discussion Questions

- Why do you think a thumbnail image for your game is important?
- How could a game's description make you want to play that game?

Teaching Tip
Some students may complete their work before others. The following resources can serve as extension or supplementary activities for all students:

- **Thumbnail personalization walkthrough and Q&A**: https://youtu.be/5NvGzKVyKxg This video provides excellent information on how to create a thumbnail and get people excited about your game.
- **Learn and Explore Chart**: https://www.roblox.com/charts#/sortName/Learn_And_Explore The games and experiences in this category provide a number of engaging examples of how best to design a game detail page.

Next we will:

- Work on presentations
- Present and test each game

Module 10: Polish and Present (50 minutes)

Description

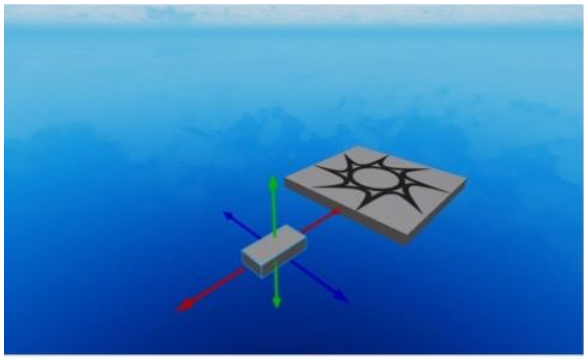
Present the planning, code, and final results of each group project.

Objectives

- Present group projects

Optional Resources

- Studio UI Reference
- Studio Hotkeys



Example of a group project

Lesson Introduction (5 minutes)

Teaching Tip

If your class is too large to allow everyone to present, have two or three groups present to each other instead.

It is important to remind the students that game developers who listen to their players' feedback and update their games often have the happiest game players. Games that continually update and improve are often the games that are highly respected in the Roblox community.

Independent Practice (10 minutes)

Create a presentation or use your game detail page and a live playthrough to present your game to the class.

Decide who in the group will talk about each of the following:

- The theme of your obby
- 3 - 5 trivia questions included
- At least 4 different if/then or for/each statements used in your obby

- What other code was used to create the obby
- Original drawings and plans
- Feedback you received about your obby

Present (30 minutes)

Students should specifically address the key requirements of the project:

- The project must have:
    - an easy to identify theme such as:
        - Ocean conservation
        - Tips for tourists visiting your town
        - A favorite national park
    - at least 3 - 5 questions for players to answer
    - at least 4 - 5 types of player/object interactions such as:
        - if a player touches a part, then:
            - The part will turn green
            - The part will disappear
            - The player will be forced to respawn
    - feedback from at least 3 people not on your team who reviewed both your game design concept and performed at least one playtest each.

Lesson Recap (5 Minutes)

- Questions for reflection:
    - What was the most fun aspect of creating your obby?
    - What was the biggest challenge you faced while planning and building your obby?
    - What will you build next on Roblox?