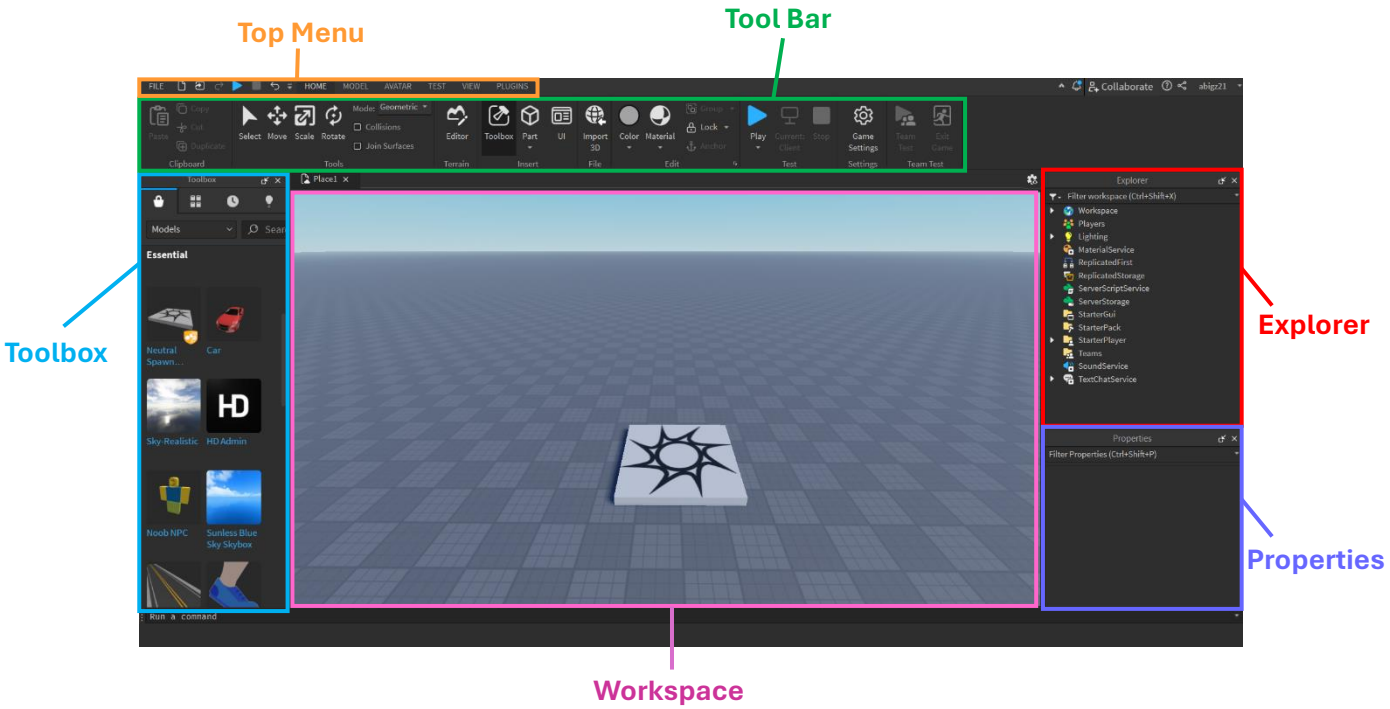
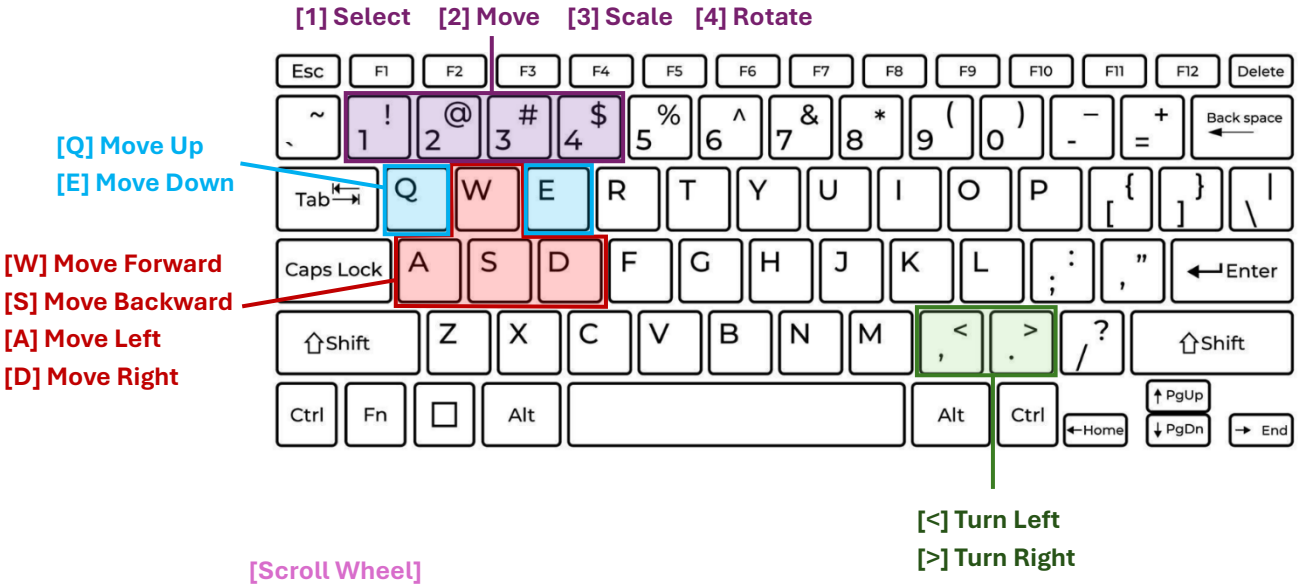
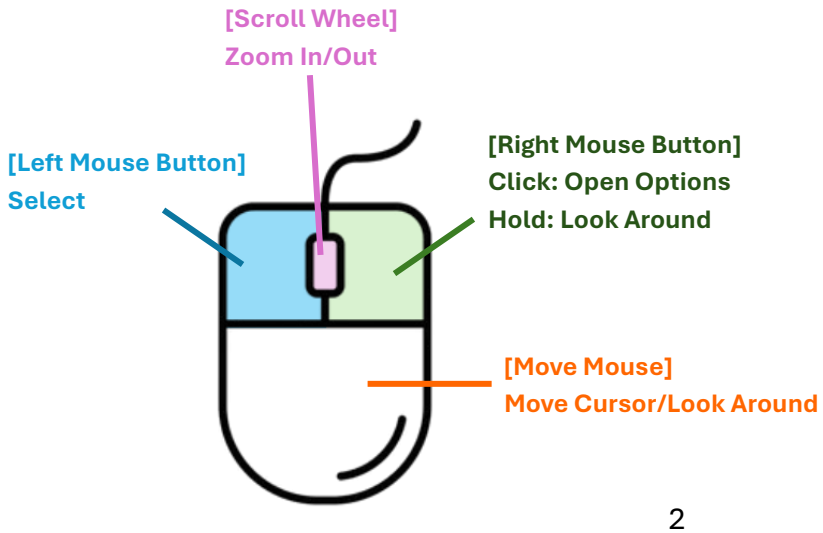
Champ Code Academy

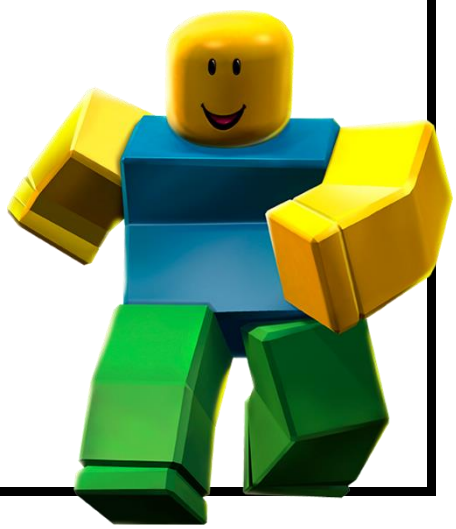
Musical Tiles

Roblox Studio Controls









Lesson Overview

Game Objective:

- Add music to tiles that produce sounds when stepped on!
- Discover and experiment with a variety of sound effects.
- Add a fun twist by making the tiles light up in different colors when stepped on!

Game Extras:

- Personalize your tiles with cool decals and designs.
- Create your own piano tiles, add music and decals.

Learning Objectives

- Understand how to add and manage sound effects in Roblox parts.
- Learn to use the Roblox Studio Toolbox to find assets like sounds and decals.
- Apply the concept of debounce to control repeated actions.

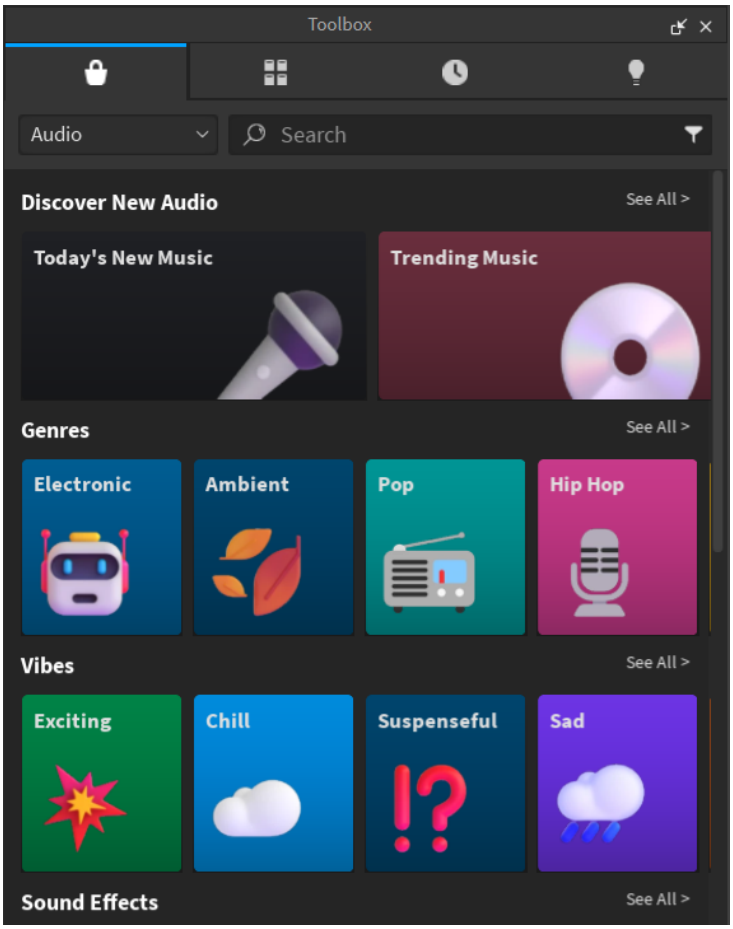
Lecture – What are Sound Effects?

Sound Effects

Sound effects, also known as **SFX**, are audio clips added to your game to make it more interactive and fun! For example, you can use sound effects for footsteps, explosions, or even music.

Toolbox

The **Toolbox** is a library in Roblox Studio where you can find ready-made assets like audio, models, and decals. Instead of creating everything from scratch, you can browse and pick items to add to your game quickly.

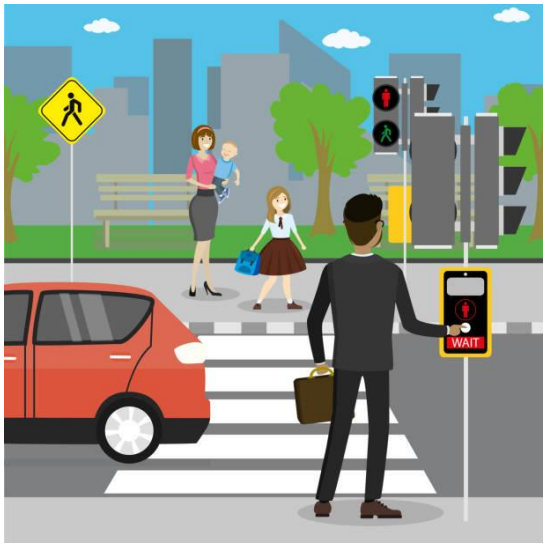


Lecture – What is Debounce?

Debounce

Debounce is a programming concept that stops an action from happening too many times in a row.

An example is **pressing the crosswalk button**. If you press it repeatedly, the button will only register your press **once** until the traffic signal changes. After the signal has changed, you can press it again, but it won't register a new press until then. The debounce concept ensures that the action (changing the light) only happens after a short wait.



Debounce Syntax

```
local debounce = false

local function onEvent()
    if not debounce then
        debounce = true
        -- Action code here
        wait(seconds)
        debounce = false
    end
end
```

Creates a variable called "**debounce**" that stores a **Boolean value**.

Checks if debounce is **false**.

If it is, it will set **debounce** to **true** temporarily to avoid the action from being triggered and will run the code.

A short pause is added before resetting the **debounce** back to **false**, meaning the action can be triggered again.

1) What is an example of using sound effects?

a. Giving the player a jump boost
b. The player's score increasing
c. The player's health points changing
d. The sound of footsteps when a player walks

2) What is the Toolbox used for?

a. A place to store your game's code
b. A place to find items, models, and sounds to use in your game
c. A tool that will help players win in the game
d. A place where you publish your finished game

3) What does debounce do in programming?

a. Makes sure an action happens only once
b. Makes sure the character can only move in one direction
c. Prevents an action from happening too many times quickly
d. Prevents the player to do an action forever

4) What does this code do?

```
if not debounce then
    debounce = true
      -- Action code here
    wait(seconds)
    debounce = false
```

a. It stops the action from happening too quickly by waiting for a few seconds

b. It makes the action happen instantly without delay

c. It changes the action everytime it is triggered

d. It makes the action happen only once, no matter what

Sound Effects

- Audio clips added to your game to make it more interactive and fun

ToolBox

- A library in Roblox Studio where you can find ready-made assets like audio, models, and decals

Debounce

- A programming concept that stops an action from happening too many times in a row.

CODES LEARNT:

Debounce Syntax

```
if not debounce then
    debounce = true
    -- Action code here
    wait(seconds)
    debounce = false
end
```

The code checks if **debounce** is **false**.

If it is, **debounce** is temporarily set to **true** to prevent the action from triggering again.

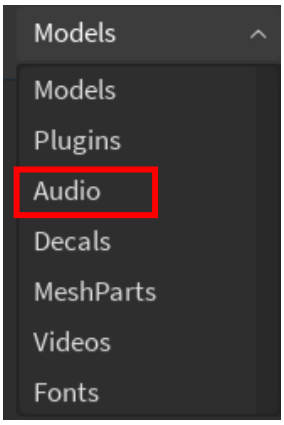
After a short pause, **debounce** is reset to **false**, allowing the action to happen again.

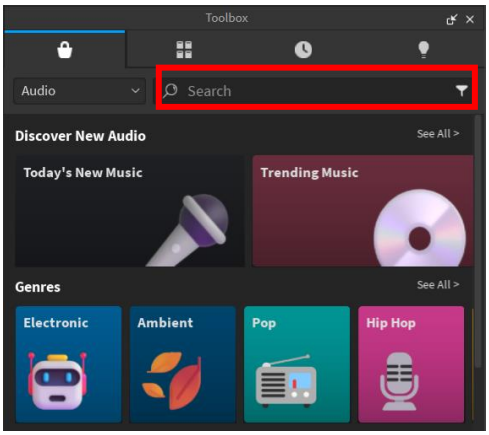
In-Class Activity 1 – Making it Musical!

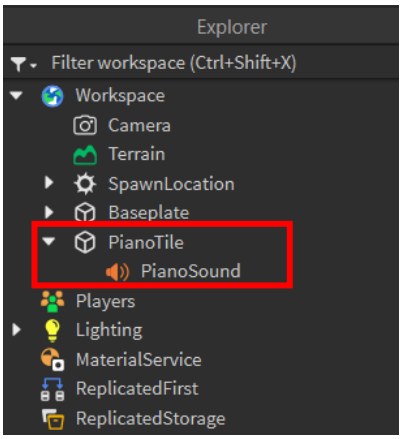
Step 1: Adding Sound Effects

To add a sound effect, open **Toolbox** and go to **Audio**.

Then, choose your desired sound effect and insert it **inside the part**! You can also use the search bar to look up the sound you want. You can also rename the sound when added.







Your Turn!

Open **Musical Tiles.rbxl** file and set up the game by following the steps above!

Step 2: Declaring variables

Variables store and hold information such as numbers and words.

We need to declare three variables: a part variable, sound variable and debounce variable.

```
1 local part = script.Parent
2 local sound = part:WaitForChild("SoundName")
3 local debounce = false
```

1. "**part**" variable will store the object that the script is attached to.

2. "**sound**" variable is used to reference the sound that will play when the part is touched. **WaitForChild("SoundName")** ensures that the script will wait for the sound object to be fully loaded before using it.

3. "**debounce**" variable stores a Boolean value used to prevent the sound from playing repeatedly in quick succession.

Step 3: Creating a Function

A **Function** is a reusable block of code that performs a specific task when called.

The first function that we need is to check if a player has touched the tiles.

```
local function getPlayerFromTouch(touched)
```

The keyword "**touched**" is a parameter of the function named "**getPlayerFromTouch**":

To check, the code should look like this:

```
local character = touched.Parent
```

Step 4: Playing the Sound

Every time the player steps on the tile, the code to play will be triggered.

First, the debounce is set to true to prevent the code from running multiple times simultaneously.

```
debounce = true
```

Then, the sound will play.

```
sound:Play()
```

After the sound has played, the code will pause for the duration of the sound, plus an additional 3 seconds. This ensures that the sound won't play again before it has finished. After that, debounce will be set back to false so the sound can play again.

```
wait (sound.TimeLength + 3)
debounce = false
```

In-Class Activity 1 – Making it Musical!

Your Turn!

Open "**TileScript1**" from PianoTile1 and fill in the blanks with the correct code.

BLANK #	HINT
1	What is the code that gets the object that the script is attached to?
2	What method is used to wait for the sound to be loaded?
3	What is the name of your sound?
4	What boolean value should be assigned to debounce to tell that no action has occurred yet?
5	How do you check if the part has been touched by the player?
6	Set debounce to true
7	Code to play the sound
8	Get the time length of the sound
9	Set debounce to false

Overall Code

```lua
-- 1. What is the code that gets the object that the script is attached to?
local part = script.Parent

-- 2. What method is used to wait for the sound to be loaded?
-- 3. What is the name of your sound?
local sound = part:WaitForChild("PianoSound") -- Ensure sound exists

-- 4. What boolean value should be assigned to debounce to tell that no action has occurred yet?
local debounce = false


local function getPlayerFromTouch(touched)

    -- 5. How do you check if the part has been touched by the player?
    local characterTouched = touched.Parent

    if characterTouched and characterTouched:FindFirstChild("Humanoid") then
        return game.Players:GetPlayerFromCharacter(characterTouched)
    end
    return nil
end

part.Touched:Connect(function(hit)

    local player = getPlayerFromTouch(hit)
    if player and not debounce then

        -- 6. Set debounce to true
        debounce = true

        -- 7. Code to play the sound
        sound:Play()

        -- 8. Get the time length of the sound
        wait(sound.TimeLength + 3)

        -- 9. Set debounce to false
        debounce = false

    end
end)
```

Additional Activity 1 – Decorating Your Tiles

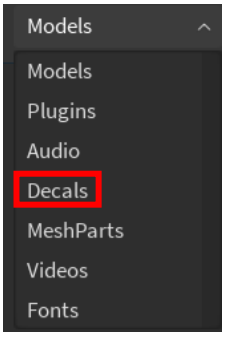
Let’s add fun images and icons on our tiles!

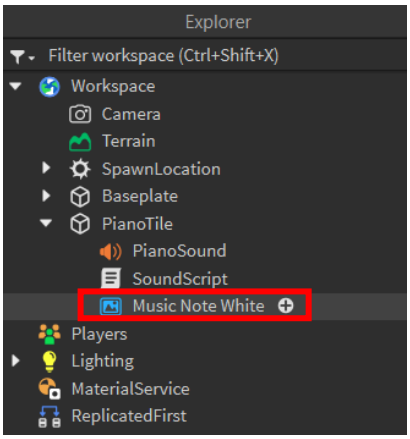
Step 1: Choosing a Decal

To choose a decal, head over to **Toolbox** and go to **Decals**.

Click on the design that you like or use the **search bar** to look up your preferred decal.

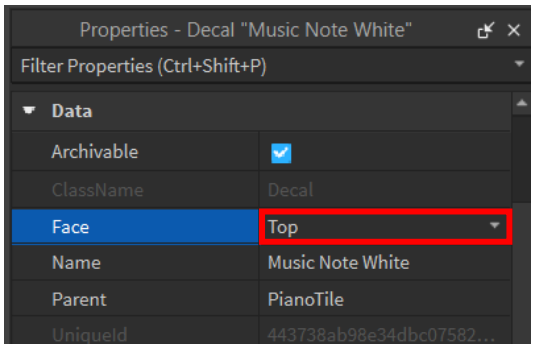
When inserting a decal, make sure it under the part where you want it to appear.





Step 2: Decal Properties

To place the decal on the top side of the part, select the decal, go to **Properties**, find the "**Face**" option, and set it to "**Top**".



And there you go! You've successfully added a decal to your part.

Additional Activity 2 – Colors Changing Tiles

Step 1: Creating a Function

math.random generates a random number within a specified range.

First, create a function called "**getRandomColor**" that chooses a random color:

```
local function getRandomColor()
    return Color3.fromRGB(math.random(0, 255), math.random(0, 255), math.random(0, 255))
end
```

Step 2: Calling the Function

Call the function right after the **sound:Play()** code, so that the color change happens immediately after the sound starts playing.

```
part.Color = getRandomColor()
```

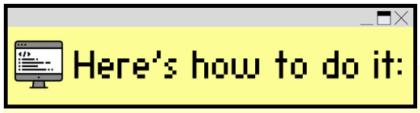
Your Turn!

Let's update your script to make the part change color every time the sound plays.

1. Add the function that generates a random color using **math.random**. Place it **before** the **part.Touched** function.

2. Call the function to change the part's color after the sound has played.

Additional Activity 2 – Colors Changing Tiles



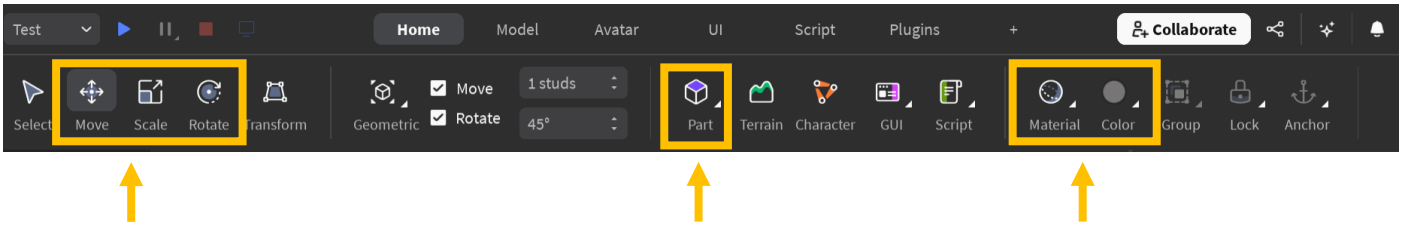

```lua
-- 1. What is the code that gets the object that the script is attached to?
local part = script.Parent

-- 2. What method is used to wait for the sound to be loaded?
-- 3. What is the name of your sound?
local sound = part:WaitForChild("PianoSound") -- Ensure sound exists

-- 4. What boolean value should be assigned to debounce to tell that no action has occurred yet?
local debounce = false


local function getPlayerFromTouch(touched)

    -- 5. How do you check if the part has been touched by the player?
    local characterTouched = touched.Parent

    if characterTouched and characterTouched:FindFirstChild("Humanoid") then
        return game.Players:GetPlayerFromCharacter(characterTouched)
    end
    return nil
end

local function getRandomColor()
    return Color3.fromRGB(math.random(0, 255), math.random(0, 255), math.random(0, 255))
end

part.Touched:Connect(function(hit)

    local player = getPlayerFromTouch(hit)
    if player and not debounce then

        -- 6. Set debounce to true
        debounce = true

        -- 7. Code to play the sound
        sound:Play()

        part.Color = getRandomColor()

        -- 8. Get the time length of the sound
        wait(sound.TimeLength + 3)

        -- 9. Set debounce to false
        debounce = false

    end
end)
```

Take Home Activity – Create Your own Musical Tiles!

Now let's add more tiles to your game!

Step 1: Adding Parts

Go ahead to your ribbon panel on the top of your screen to add, change, and design your parts!



Move: Move the part

Scale: Change the Size

Rotate: Change the tilt

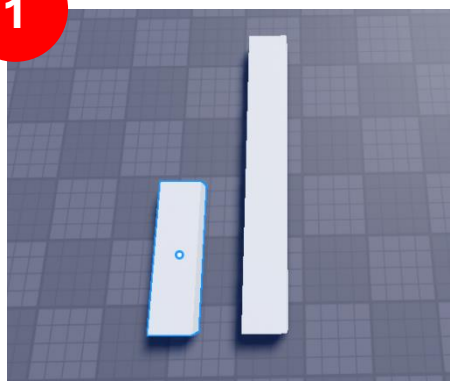
Create a part

Change the **material** / **colour**

Step 2: Union

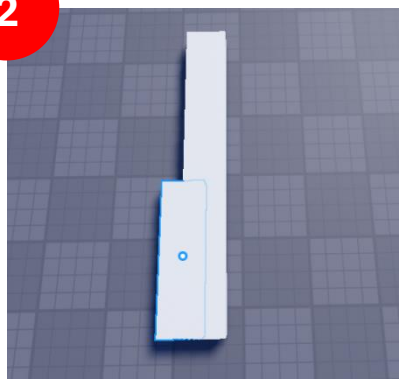
You can use "union" to merge parts together to make a "complex" looking part

1



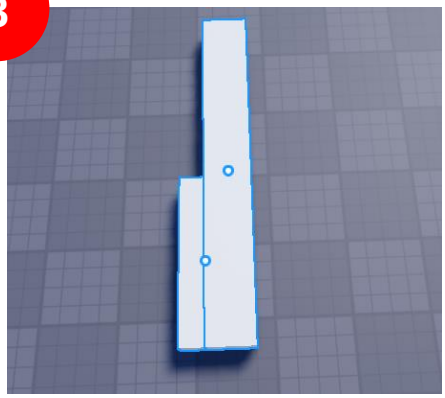
Add 2 or more parts

2



Move the parts so that they are touching each other

3

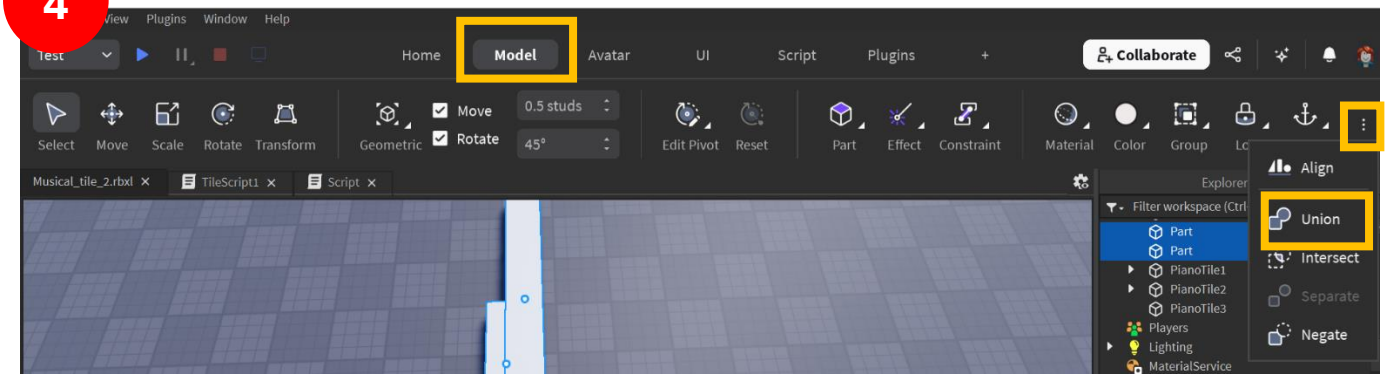


Select both parts

(Either left click and drag to select, or hold the shift button while left clicking desired parts)

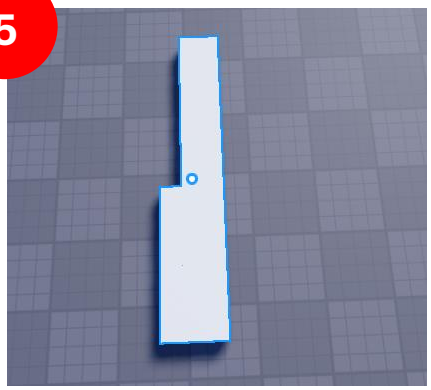
Take Home Activity – Create Your own Musical Tiles!

4



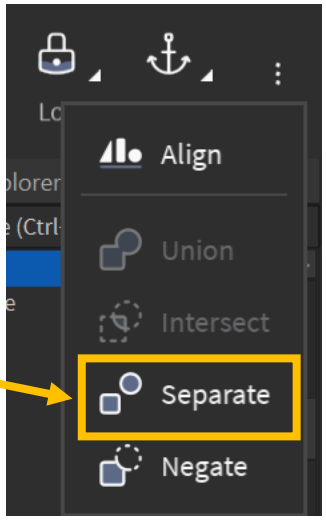
1) Go to the "Model" tab at the top of your screen
2) Press the 3 dots button ⋮ on the far right
3) Press union to merge the selected parts

5



Finished part!

(Hint: If you want to undo, you can press "Separate" under the same 3 dots button ⋮)



Step 3: Adding Sound and Code to Your Tiles

Once you're happy with your tiles, you can use the same codes in previous activities to make your own musical tile game!