Roblox Supplemental 1 & 2

Table of Content

Roblox Supplemental 1

Chapter 1. Camera Keyboard Controls

A, Left Direction Key - move camera left

D, Right Direction Key - move camera right

W, Up Arrow Key - move camera forward

S, Down Arrow Key - move camera backwards

Q - move the camera down

E - move the camera up

i - zoom in

o - zoom out

Chapter 2. Authoring - Clipboard

Cut - select an object

Copy - create a copy or clone of the selected object

Paste - insert the copy or clone

Duplicate - create a copy of the selected object

Chapter 3. Ribbon Panel called "Model" (look up to the top of the navigation)

3.1. Object Manipulation -

Tools scale - change the object's

size rotate - rotate on an axis

move - position an object

3.2. Material

Change the material composition of the selected object

3.3. Color

Change the color of the selected object

3.4. Group via Ctrl G

Composition of parts into a group

3.5. Ungroup via Ctrl U

Break up the group composition and return to individual parts

3.6. Anchor
Place an object permanently in a location

Chapter 4. Terrain - used to create your own Roblox world

4.1. Erode
Make a depression at location of the mouse click

4.2. Grow
Make a hill or a mountain at location of the mouse click

4.3. Add
Insert biome at location of the mouse click

4.4. Subtract
Remove biome at the location of the mouse click

4.5. Paint
Paint the area, not the object. This allows us to paint snow on top of the mountain

4.6. Smooth

a. level a depression, hill or mountain. For example, if we have a depression, smoothing it brings the depression up OR, if we have a hill, smoothing it brings it down

This end result of using this feature is that the location of the mouse click becomes a flat plain

4.7. Generate

Select one or multiple biome and add them into the game

4.8. Region

Select the entire selection, group them together, and all changes apply to all of the parts that make up the group

Chapter 5. Home Menu --> Toolbox

Display premade models that can be included into the game

Chapter 6. Home Menu --> View --> Explorer

Displays the game hierarchy and lists all game objects within the game.

To find a part and its location in the game, go to the Explorer and click on the part. After being clicked on, the part is highlighted. Next, click on "f" to find the part and its specific location.

Chapter 7. Home Menu --> View --> Properties

Displays the name of property and property value of the selected game object

Challenge - Show Sensei

a. Create a snow capped mountain

b. zoom in, out; move camera left, right

c. rotate the camera

d. play game and have avatar jump

Chapter 8. Composition & Creation

Composition means to combine or merge together different entities. For us, we can compose objects by **using group and ungroup controls.**

To combine different objects together, click and hold down the left button of the mouse. Next, move the mouse cursor and you will see a box. Draw the mouse cursor so that it includes all of the entities that are to be combined are "within" the box

1. **all entities that are to be combined will have their outline lit up**
2. **hold down the "ctrl" button and then click on the "g" key of the keyboard. This will combine all of the entities together into one new object**
3. **to ungroup, hold down the "ctrl" button and then click on "u" to ungroup**

Composition is important because it allows us to create our own parts using the basic, primitive parts (wedge, sphere, cube, and cylinder)

By combining parts, we are now able to create custom components to make our game unique

Challenge - Show Sensei

a. create the Batcave, Avengers Fortress, or anything you want

Super Challenge - Show Sensei

a. create table and couch using composition of parts

b. group the parts together so that the entire table moves when it is being dragged

Chapter 9. Introduction to Programming

What is programming? Programming is writing characters in a structured and organized way that makes sense to the computer. The structure and organization creates an instruction and computers are machines that executes instructions

9.1. Insert A Part and Rename It

9.1.1. insert a cube into your game.

9.1.2. next, click on the cube and press F2 and then change it's name to "**myCube**".

9.2. Add A Script To Your Game

9.2.1. find the cube and left click on the mouse to select your part. To the right of the part, you will find a plus symbol (+). Click on this plus and type in "Script"

9.2.1.1. do not select "LocalScript", that is something else

9.3. Print Statement

Communicating with the computer is very important because communication can be used to help us debug.

Roblox has an output console that allows us to print messages. To open the output console, look up to the navigation bar and find the "**View**" ribbon. Click on this "**View**" ribbon. Next, find and click on the "Output" button to open the output console.

The print statement allows us to print any custom message to the Output console.

```
print( ”My favorite number is ” .. 5 )
```

The ".." from above is called the concatenation operator. It is used to combine two things together.

Chapter 10. Basic Roblox Programming

A game is composed of actions and these actions are used to manipulate property values. The actions are implemented via functions and proper values are implemented as storage elements.

10.1. Local Variables

We declare a local variable by using the reserve word "local" followed by the name of the variable. Next, we assign data to the variable using the assignment operator.

NOTE: the word "var" is not needed
NOTE: Roblox does not need a delimiter to end a line of code.

```
local myPart = 0
```

10.2. Function Definition and Function Call

10.2.1. A function definition in the Roblox programming language requires an "end". This tells the computer that the function definition has ended.
10.2.2. In most programming languages, we group together lines of code using curly braces { }. However, in Roblox, we do not need to use curly braces. Instead, the computer will search for the reserve word "end". We start a function definition by using

```
local function sayHello()

end
```

The beginning is the "local function sayHello()"
The reserve word "end" signals the end

Lines of code in between the beginning and end will be grouped together by the computer.

Challenge 1
Inside your function definition, write the code to print your name.

Function Call

In order to run the function, we now have to call its name. After the "end" statement of your function definition, call the name of your function.

```
sayHello()
```

Challenge 2
Below the function definition, call the name of the function and then press F5 to run the game. The output console should print your name.

Chapter 11. Passing Data To Function Definition

We can make our print statement unique by passing data to the function definition. To pass data to the function definition, we need to put data between the open and closed parenthesis.

In the example below, the parameter is "**Code Ninjas PV**". This data is put into a variable called **nameVar**. It is important to give data a name because doing so means that the function definition can access the data by referencing the name of the variable that holds the data. The function definition then accesses the parameter in the body of the function definition.

```
-- function definition
local function sayHello( nameVar )

     print( ”Hi, my name is ” .. nameVar )

end


-- function call
sayHello( ”Code Ninjas PV” )
```

Challenge
Write the code above but change the data that is passed to the function definition. We do not want to pass "Code Ninjas PV" to the function definition. Instead, replace it with your name.

Press F5 and check that the code does print your name in the output console.

Chapter 12. Looping

What if we want to execute a block of code forever? In this case, we use an infinite loop. The while loop can be used to ensure a block of code continues to run. This can be used to create a time delay. An infinite loop can be created by writing the following code

```
while true do


end
```

Chapter 13. Delay

What if we want to create a delay? Then we use the `wait()` function. *** **NOTE: always put a number inside the open and closed parenthesis. ***.** The time unit for a `wait()` function is seconds. For example, `wait(1)` means to wait 1 second before proceeding to the line of code after the `wait(1)`

For us, we will create a 1 second delay using `wait(1)`

Chapter 14. Random Number Generator

There are times we want to change the frequency of lights flashing. ** **NOTE** ** **Future challenges will be to create infinity stones flashing at different frequencies.**

We do not want lights to flash at the same frequencies. Instead, let's have different infinity stones flash at different frequencies.

To make things interesting, let's have infinity stones blink at random speeds. To prevent repeating numbers, we must create a seed. We can do this by writing

```
math.randomseed(tick())
```

Once our seed is set, we then use math.random(100) to select a single random number between 1 and 100

```
math.random(100)
```

Challenge - Running Numbers

Combine function definition, passing data to function definition, function call, variable, wait(), random number generator, print statement, and looping to create a script that prints out a random number every 2 seconds.

First, insert a sphere into your game. You can rename the sphere by clicking on the sphere and pressing F2. Rename the sphere "**mySphere**" and then add a script to the sphere

The script should ***<u>ONLY INCLUDE</u>*** the items listed below

a. at least 1 function definition. The name of the function definition is "**loopyRandom**"

b. the function definition must accept at least 1 parameter

c. must use at least 1 local variable

d. must use while true loop

e. must use wait(2)

f. must use random number seed and generator. The range of random values is between 1 and 50

g. must print out the random number to the output console

h. the function name is called when the game starts

Chapter 15. Hierarchy

A Roblox game is based on a hierarchy and this means that the entities within a Roblox game is organized as a tree structure. The root (or the top) node is the game itself and then comes the workspace. This means that we can use hierarchy to find our way to a part.

```
game.Workspace.mySphere;
```

Chapter 16. Event programming

In the previous challenge, the function "**loopyRandom**" will run when the Roblox game starts. This is important because there are situations where we want a function to run when the game starts.

However, what if we want to execute a function ***<u>ONLY</u>*** when something happens? This "something happens" is called an event and events are **VERY IMPORTANT** to game development.

One of the most basic event is called Touched. When we touch a part, like a wedge, we want to execute a function (an action). How do we do this?

In Roblox, we have to ***connect*** a function to an event. How can we connect a function to an event? We use the function Connect() - notice the open and closed parenthesis after "**connect**"

Remember the first part you inserted into your Roblox game? This was a cube that was renamed "**myCube**". Write the following code in your script that you gave to myCube. Afterwards, press F5 to play the game.

```
local myPart = game.Workspace.myCube;

-- function definition

local function sayHello( nameVar )

    print( "Hi, my name is " .. nameVar )

end

myPart.Touched:Connect( sayHello );
```

16.1. Error - Can't Pass Data To Function using Connect

When F5 is pressed and the game starts, touching the part named "**myCube**" might not work. The reason why is because the function definition "**sayHello**" takes in a parameter. However, we cannot pass a parameter to a function definition when using event programming. This means that the code below will not work.

```
local myPart = game.Workspace.myCube;

-- function definition

local function sayHello( nameVar )

    print( "Hi, my name is " .. nameVar )

end


myPart.Touched:Connect( sayHello( "spiderMan") );
```

Challenge 1 - Find Another Way

a. find a way to print out your name **ONLY WHEN "myCube"** is touched. This is a tricky problem. You might have to use a function within a function ...

Challenge 2 - Touch To Print Random Number

a. based on what you learned after completing Challenge 1, use it to complete Challenge 2

b. refer to the part called "**mySphere**"

c. go back to the script that was attached to "**mySphere**"

d. modify the code so that when your avatar touches "**mySphere**", this touch event will call the function "**loopyRandom**"

Chapter 17. Object Property

A property is something that describes a game object. In our example, we have a part that was renamed "**myCube**". Click on "**myCube**" and search for the "**Properties**" panel. You will see that these are the properties that describe myCube.

These properties of the object are storage elements. This means that we can read the content and also write data into them.

For example, our part called "**myCube**" can have a different color. We use our hierarchy and then reference the property name. ** **NOTE: it is important to recognize that the property name must be spelled exactly, including lower or uppercase. Else, the computer will be confused** **

17.1. BrickColor Property

Write the code below into your script for myCube:

```
game.Workspace.myCube.BrickColor = BrickColor.new( 155 );
```

Notice that BrickColor is capital B and C. The code above creates a new BrickColor and gives it the value of 155. We then take this new color and assign it into our existing myCube.

Press F5 to run the game. When does the BrickColor actually change? The color of "**myCube**" changes once the game starts. This is fine but we want an event to change the BrickColor.

17.2. One Part, Multiple Scripts

It is possible for a part to have multiple scripts. Add another script to "**myCube**" and prepare for your challenge. The challenge should be done in your second script that was attached to "**myCube**"

Challenge

a. Write the code that will change the BrickColor only when the Avatar touches "**myCube**"

Chapter 18. Animation

We can create animation by having the BrickColor change values. In order to see the change in color, we need to create a delay. In Roblox, a delay can be created using `wait(1)`.

We can also create flashing animation by using a while true loop. In Roblox, the syntax for a while true loop is written below. Notice that the word "**end**" signals the end of a block of code

```
while true do
end
```

Challenge - Animation

Modify your second script to do the following

a. keep the touch event

b. After your avatar "touches" the part named "**myCube**", the part should change color ***every*** 1 second **forever**

c. you might need to create a 2nd function definition in order to do the flashing animation ...

Chapter 19. Shortcut --> script.Parent

Using a hierarchy allows us to find the part. We start at the root node and work our way down. This is great organization but it forces us to type a long line of code just to refer to a part. A spelling mistake will cause the script to not run. There is a shortcut.

Assumption
1. a part has already been inserted into your game.
2. a script has been inserted into a part

Inside any script that has been attached to a part, we can refer to the part using

```
local myPart = script.Parent
```

In yellow, orange, and green belt, we use `$this` to refer to the currently selected object. In Roblox, `script.Parent` refers to the currently selected part. **We will use this as our shortcut**

Chapter 20. Property Of Humanoid (This is your avatar)

Roblox parts, like cube, cylinder, or sphere, have property values that describe it. The avatar that we control, called Humanoid, also has property values.

One important property value is called Health (**NOTE: the property name is spelled with a capital H**). This is important because we are now able to manipulate the health of the Humanoid.

20.1. One Part, Multiple Scripts

It is possible for a part to have multiple scripts. Add another script to myCube and write the code below. Remember that we need to have "end" or else the computer gets confused. Write the following code ***exactly.*** What do you think will happen when your avatar touches myCube?

```
-- using shortcut
local myPart = script.Parent

-- function definition
local function onPartTouch(otherPart)
    local partParent = otherPart.Parent
    local humanoid = partParent:FindFirstChildWhichIsA("Humanoid")
    if ( humanoid ) then
        -- Set player's health to 0
        humanoid.Health = 0
    end
end

-- event listener
-- game.Workspace.myCube.Touched:Connect(onPartTouch)
myPart.Touched:Connect(onPartTouch)
```

Explanation

```
local function onPartTouch(otherPart)
```

a. the code above is the heading of function definition. The function name is "**onPartTouch**".   The input to the function definition, which is named "**otherPart**", is placed their by Roblox.   The "**otherPart**" identifies the part that touched "**myCube**".

```
local humanoid = partParent:FindFirstChildWhichIsA(”Humanoid”)
```

a. remember that Roblox is based on a hierarchy. When a part touches "**myCube**", we want to find out whether or not the part is our Avatar. The code above uses the hierarchy to find the location of the part.

```
if ( humanoid ) then
```

a. after we find the part, we have to check if it is our Avatar. The "if" statement performs this check. If our Avatar did indeed touch "**myCube**", ONLY then do we manipulate the "**Health**" property of the Avatar.

20.2. Super Power
The avatar also has two properties that allow us to create a super hero game. These two properties are named **JumpPower** and **WalkSpeed** (remember lower and upper case do matter and spell exactly)

Challenge 1 - Super Speed & Super Jump
a.   Insert a wedge into your game and rename it "**superJump**". Insert a script into your wedge. When your avatar touches the wedge, the avatar's **JumpPower** property value will increase by 50 points. Also, the avatar’s **JumpHeight** property value will also increase by 50 points.

b.   Insert a cube into your game and rename it "**superSpeed**". Insert a script into your cube. When your avatar touches the cube, the avatar's **WalkSpeed** property will increase by 10 points

Challenge 2 - Health Pack and Hazard Pack
a. create a health pack for your avatar. This health pack is cube shape and will be the color red (the color red is easy to spot). When the avatar touches the health pack, then the avatar's health will ***increase*** by 10 points.

b. create a hazard pack for your avatar. This hazard pack is cube shape and will be the color light grey. When the avatar touches the hazard pack, then the avatar's health will ***decrease*** by 10 points.

c. for both health and hazard pack, the color of each pack is assigned by writing code inside a script.

Super Challenge 1 - Scavenger Hunt World
Recall that "**myCube**" is flashing at different colors and is the part that makes the avatar's health go down to 0. This is one of our infinity stone.

Create your own world using the Terrain Editor. Hide the infinity stone named "**myCube**" somewhere in your world (underwater, in the air, inside a cave, and etc). It is a scavenger hunt and so do not reveal the location of your infinity stone. Be creative when creating with your Scavenger Hunt World.

The goal is to find the first infinity stone named "myCube". If your world is a big world, then it will take some time to find the infinity stone. The wedge will give your avatar super jump and the cube will give your avatar super speed. These two super powers will shorten the time needed to find the infinity stone.

When you are done, have a fellow Ninja or Sensei play your game and try to find the infinity stone. Write down the name of the person who found it in the shortest time.

Super Challenge 2 - Multiple Infinity Stones
The part named "**myCube**" is one infinity stone. Now, incorporate multiple infinity stones into your game.

a. Each stone will flash at different colors and different frequencies. Write code to do this.

a.1. Recall that "**myCube**" only changes color when the Avatar "touches" it.

b. Each stone is of different shape and size (you can do this manually instead of with code)

c. Each stone will be a different part (can't have 6 infinity stones of the same cube shape)

d. Modify your wedge named "**superJump**" to be one of the infinity stones d.1. the Avatar gets super power when the Avatar touches the wedge

d.2. the flashing animation will be different. When the game starts, the wedge will flash at different frequencies and color - no touch event needed

e. Modify your cube named "**superSpeed**" to be one of the infinity stones

e.1. the Avatar gets super power when the Avatar touches the cube

e.2. the flashing animation will be different. When the game starts, the cube will flash at different frequencies and color - no touch event needed

f. For the other infinity stones, they will flash at different colors and frequencies when the game starts - no touch event is needed

Super Challenge 3 - Composite Function

a. write code so that **WalkSpeed** and **JumpPower** only has a duration of 5 seconds.

b. after 5 seconds has elapsed, the **WalkSpeed** and **JumpPower** return to their default values

Super Challenge 4 - Danger Ramp

Modify your Scavenger Hunt World so that there is a body of water. The only way to cross this body of water is to walk across a bridge. For this Super Challenge, the bridge will be a cube.

a. modify your Scavenger Hunt World to have a body of water and a hill on opposite ends of the body of water

b. insert a cube into the game and rename it "**dangerRamp**". Manually elongate the cube so that it stretches from one hill to another. This creates our bridge. Make the bridge wide enough so that your Avatar can run or walk across it. However, don't make it too wide or else the task stated below will be easily completed.

c. Insert a script into "**dangerRamp**" and use the short cut to refer to the currently selected object.

c.1. inside the script, write code to have the "**dangerRamp**" flash at different colors every .2 second. The transition from one color to another occurs every .2 seconds in an infinite loop c.2. for the value of your BrickColor, start at color code 1. After the .2 seconds are up, then add 1 to this color code. The color codes of a part can be found at

https://developer.roblox.com/en-us/articles/BrickColor-Codes

d. For this Super Challenge, we will use an "if" statement. The syntax for an "if" statement is as follows:

```
if true then

end
```

We can replace the "true" with any math expression. We can check the current value of the BrickColor by using am embedded property. Specifically,

```
BrickColor.Number -- this tells us the current color code
```

While the Avatar is walking across the "**dangerRamp**", if the ramp's BrickColor.Number is 24 or 49 or 124, then the Avatar's Health property is set to 0.

When the BrickColor.Number is greater than 140, then reset the BrickColor.Number back to 1;

e. the goal is to get across the "**dangerRamp**" safely.

This is essentially a guessing game. The goal is to run or walk across the dangerRamp safely. However, if you run too slow, then the brick color might set your health to 0.

Some players will try to guess the color and jump to avoid touching the ramp when its color is 24, 49, or 124.

Roblox Supplemental 2

Chapter 1. Introduction to Worm Hole

What is a worm hole? According to https://en.wikipedia.org/wiki/Wormhole, a wormhole is "a tunnel with two ends, each at separate points in spacetime. A wormhole can be used to travel long distances". A wormhole can also be thought of as a shortcut.

Why are worm holes useful? A wormhole allows us to travel a long distance in a short amount of time. What are some real world applications of worm holes? In Star Trek, a worm hole is used to travel from one point to another. Wormholes can be used to quickly transport items or human beings (ie. to give aid) and for instantaneous messaging.

In video games, wormholes are used as a portal to a new world. Each new world has its own challenges. Rather than having a player run from the player's current position to the destination, we will now have the player enter the wormhole and be teleported directly to a new world.

A wormhole can be used to get us to our homebase. Our homebase can hold our achievements, be a place where we can acquire our next task, or serve as a reference point.

Chapter 2.Teleportation by Manipulating CFRAME

In Roblox, we have the capability to create a wormhole and use it to teleport the Avatar to any location. This opens the door to many different ways to design a game.

In Roblox, the Avatar that we control is composed of many different parts (such as head, arms, and legs). We can use this to our advantage. We will focus on the upper torso. The upper torso has a property called CFrame, which stands for ***coordinate frame***. This CFrame is described by 3 numbers and the 3 numbers refer to the x, y, and z position.

By storing a new value into the CFrame of the upper torso, we are instantly placed to a new location. Look at the code below. Notice that we created a new CFrame and this new CFrame has inputs 78, 2.449, and -138.28. The numbers represent the x, y, and z coordinates. After we create and initialize the new CFrame, we must assign the new data to our existing upper torso.

The code to assign the upper torso to our new position is

```
hit.Parent.UpperTorso.CFrame = CFrame.new( 78, 2.449, -138.28 )
```

The code below is shorthand notation for a touch event. ***** NOTE *** This shorthand notation can be tricky to read. Separating the code into function definition and event will make it easier to read, test, and maintain.**

```
script.Parent.Touched:connect(function(hit)
     if hit.Parent:FindFirstChild( ”Humanoid” )then
         hit.Parent.UpperTorso.CFrame = CFrame.new( 78, 2.449, -138.28)

     end

end)
```

Challenge

a. insert a cylinder part into your game and rename it "**teleporter1**".

a.1. change the teleporter to be a light blue color. No coding needed. Instead, manually change the color to light blue

a.2. resize the teleporter so that it is a long, vertical cylinder.

b. insert a script into "**teleporter1**" and write the code that was previously discussed.

c. test the code and see if the avatar touching the cylinder results in the avatar being teleported to the location specified.

Super Challenge 1

a. rewrite the code that was given to you by splitting the code into function definition and event

b. for this challenge and beyond, do not use `game.Workspace`. Instead, use `script.Parent.`

Super Challenge 2

a. insert 2 more teleporters into your game. These two teleporters should have the name "**teleporter2**" and "**teleporter3**"

a.1. insert a script into both take the code that you rewrote in Super Challenge 1 and apply it to both "**teleporter2**" and "**teleporter3**"

b. each teleporter should teleport the avatar to a different location within the game.

Chapter 3. Your Own Obby Game

https://developer.roblox.com/en-us/articles/Creating-Your-First-Game

According to Roblox, an obby is an obstacle course. The purpose of an obby is to create platforms that are used by the avatar to get from the start point to the end point. The challenge is having to avoid the obstacle along the way. The word "obstacle" is part of the "obby" name.

Challenge - Obby

a. include multiple platforms in the sky. You can manually create the platforms by inserting 5 cubes and manually positioning them into the sky.

There must be at least 5 sky platforms. As the platforms are positioned higher into the sky, the size and length become thinner and shorter. For this challenge, you can set the size and height manually.

b. modify your **"teleporter 1"** so that when the player touches **"teleporter 1"**, the player is transported to the first platform

c. the goal is to jump from one platform to the next until the highest platform is reached. An infinity stone can be found at the highest platform

Super Challenge - Obby With Teleporter

Modify your game so that when the player falls from any of the platforms, the player is teleported to safety. In this case, safety is any point that is close to the first teleporter.

Chapter 4. Part Property - Size, Position, and Velocity

In our discussion about obby levels, we noticed that the platforms are stationary. They are static and do not move at all. It would be more fun, and difficult, if our game allowed the obby to "come alive". We can make the platforms of our obby come to life by manipulating a part's size, position, and velocity property values.

We can use a delay to make a part move from one place to another and the movement is in incremental steps

4.1. Vector3 Data Type

The Size, Position, and Velocity property values (***the names must be spelled with first letter as uppercase***) take a type of data called Vector3.

Vector3 is a type of data that has 3 inputs, x, y, and z. We can change the Size, Position, or Velocity of a part by giving the part a new Vector3.

For example, we can insert a new cube into our game and call it "***moving Platform***". Next, we can change its size via code by writing

```
local movingPlatform = script.Parent

movingPlatform.Size = Vector3.new( 20, 100, 10 )
```

Press F5 and see what happens. We have basically manipulated the size of a part by writing code.

We can now make our platforms of our obby dynamic by changing the position of our part.

Challenge - Dynamic Obby

a.	modify your platforms from Obby so that they can move from right to left. When the platforms reach a certain x position, have the platforms bounce in the opposite direction. The goal is to create platforms that move side to side.

For this challenge, you must create a delay by pausing .5 seconds in between changing the position of the platforms. Since we want the platforms to move side to side forever, this would require the use of an infinite loop.

We also want the platforms to start moving once the game starts.

b. the platforms must move at random speed. Before, each platform moved after every .5 seconds. We did this by using wait(.5).

Moving at random speeds means combining random and wait().

Super Challenge - Dynamic Obby 2

a. the speed of a platform is indicated by the color and its flashing frequency

glowing white	-	speed 15,	change color very fast
glowing red	-	speed 11,	change color fast
glowing orange	-	speed 8,	change color moderately fast
glowing yellow	-	speed 5,	change color slowly
glowing purple	-	speed 2,	change color very slow

Chapter 5. Cloning

We could make multiple copies manually but that is an inefficient use of our time. A better use of our time would be to create clones by writing code.

5.1. What is cloning?

Cloning means to duplicate or to make a copy of something. Cloning is very powerful because we can duplicate a part in code instead of duplicating manually by hand.

5.2. Real world application

Cloning simplifies game development by having one script be in charge of creating multiple copies

Pick a platform from your Obby level. Insert ***a second script*** into this platform. Type the code below ***exactly*** into the script of the chosen obby platform.

```
local myBlock = script.Parent
local child = myBlock:Clone()

--child.Parent = script.Parent
child.Parent = workspace

--child.BrickColor = BrickColor.new("Gold")
child.Color = Color3.new( 139, 324, 293 )

print("done!")
```

```
 local posSt     = 10
 local posOffset = 5
 wait( 5 )
 while true do
       posSt = posSt + posOffset
       print( "posSt: " .. posSt )
       child.Position = Vector3.new( posSt, 10, 10 )
       wait(1)
end
```

The code sample from above has several items to focus on.

1. we use the :Clone() function to make a duplicate of the chosen part

2. a clone must be linked to a parent. Remember that Roblox is based on a hierarchy. A clone must be included into that hierarchy. The code that includes the duplicate into the game hierarchy is

```
child.Parent = workspace
```

3. we used a while true loop and variables to incrementally change the Position of the part. Every second, we increment posSt by posOffset. This means that every second, we move 5 units in the x position.

We then use the new value of posSt to change the x position of the Vector3. This allows the platform to move side to side.

Challenge

a. use cloning to spawn one more sky platform. This will be the 6th sky platform of your obby.

a.1. position this clone to be higher than the 5th platform, thinner and shorter than the 5th platform.

b. the clone must move from side to side. The goal is to start at the first platform (remember, touch **"teleporter 1"** to get to the first platform) and jump your way up to the 6th, top most platform.

b.1. The challenge is that the infinity stone is on the platform that you cloned and this cloned platform is moving from side to side. If you fall down at any point, then you must start over again.

Super Challenge

a. modify your obby. We no longer want to manually create platforms. Instead, we will now use cloning to create 5 sky platforms.

b. write the code to have each duplicate bounce from side to side at random speeds, is of different size and length, and flashes at different frequencies

Chapter 6. Super Keys!!! Keyboard Control & Creating New Part

So far, we control a player using the mouse buttons, the space bar on the keyboard, and the following keyboard keys (up arrow, right arrow, down arrow, and left arrow). Now, we focus on keyboard buttons outside those we already used.

6.1. __Script__ is different from <u>LocalScript</u>

Up to this point, we have used Script on our game part. However, there is another type of script called LocalScript. A local script is included into a different location. This is a special location.

Look to the panel on the right side and search for the "**Explorer**" panel. If the "**Explorer**" panel is not shown then look to the top of the navigation panel. Find "**View**" and click on it. Next, click on "**Explorer**". The "**Explorer**" panel should appear on the right hand side.

Search for "**StarterPlayer**" and click on it to open its sub-folders. Next, search for "**StarterPlayerScripts**". Hover over "**StarterPlayerScripts**" and go to the right and click on the plus icon.

This time, we want "**LocalScript**". Type in "**LocalScript**" and click on it. Once you have an empty "**LocalScript**", type in the below code ***<u>exactly</u>***.

```
-- ====================================================

local UserInputService = game:GetService("UserInputService")


local function checkIfFPressed(input, gameProcessedEvent)

	--local fPressed = UserInputService:IsKeyDown(Enum.KeyCode.F)
	if input.UserInputType == Enum.UserInputType.Keyboard then

		local keyPressed = input.KeyCode
		--print("A key is being pushed down! Key: ",input.KeyCode)
```

```
                if( keyPressed == Enum.KeyCode.F )then

                        print( "saw that F was pressed" )

                        for i = 1, 5, 1 do

                                print( "i : " .. i )
                                local circle  = Instance.new( "Part", workspace )
                                circle.Name   = "FireBall"
                                circle.CFrame = CFrame.new( 100, 100, 50 )
                                circle.Color  = Color3.new( 200, 100, 230 )
                                circle.Shape  = Enum.PartType.Ball
                        end

                end


        end

        --
--      if gameProcessedEvent then
--              print("\tThe game engine internally observed this input!")
--      else
--              print("\tThe game engine did not internally observe this input!")
--      end

end


UserInputService.InputBegan:Connect(checkIfFPressed)
--UserInputService.InputChanged:Connect(checkIfFPressed)
--UserInputService.InputEnded:Connect(checkIfFPressed)
```

There are several items of the code that we will focus on.

`game:GetService("UserInputService")`

a. this is a service that allows us to detect user input

`local function checkIfFPressed(input, gameProcessedEvent)`

a. this is the head of the function definition.

```
if( keyPressed == Enum.KeyCode.F )then
```

a.	enumeration is equivalent to alias. Each keyboard button is assigned a number. We could check for this number in an "if" statement. But instead, we use alias to make code easer to read and understand.

Instead of checking for a number in an "if" statement, we now check for the "**Enum.KeyCode.F**"

```
for i = 1, 5, 1 do

     print( "i : " .. i )

     local circle = Instance.new( "Part", workspace )
     circle.Name   = "FireBall"
     circle.CFrame = CFrame.new( 100, 100, 50 )
     circle.Color = Color3.new( 200, 100, 230 )
     circle.Shape = Enum.PartType.Ball
end
```

a. this is our first introduction to "for" loops ==> `for i = 1, 5, 1 do`

The first item is initializing our variable and the starting value. The code named our variable i and it is assigned the start value of 1.

The second item is the end value, which is 5

The third item is the the increment amount. This means that we will increment i by 1 after every round

The end result is that 5 clones are created.

NOTE: if the third item is not written, then the default increment amount is 1

b. this is our first introduction to creating a new part ==>

```
local circle = Instance.new( "Part", workspace )
```

b.1. creating a part is ***very different*** from cloning a part.

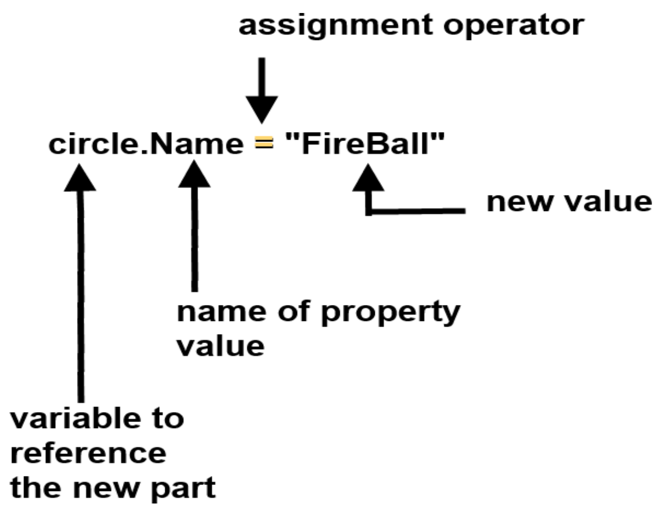
b.1.1. a clone retains all property values of the original copy.

b.1.2. creating an instance of a part means creating a blank part. This blank part is nameless and needs to be initialized. In the code above, we initialized its name, CFrame, Color, and Shape.

```
local circle = Instance.new( "Part", workspace )
circle.Name   = "FireBall"
```

The code above creates a blank part using `Instance.new()`. This new instance is given the name "Part" and is given the parent "workspace". Remember, each part must be included into the game hierarchy.

The new instance is stored into a variable called `circle`. If we want to change the property value of the part, we say `circle.Name = "FireBall"`.



Challenge

Press F5 to run the LocalScript. For this challenge, you must zoom out in order to see the effect.

Does the function `checkIfFPressed` execute immediately when the game starts? Why not?

When the function `checkIfFPressed` is executed, what happens?

Chapter 7. Hazard

At this point, we have learned several powerful, and useful, ways to develop our game. We can

1. clone a part
2. create a blank part
3. activate a script by pressing a key button on the keyboard

These allow us to create hazards on the fly. For us, a hazard can be defined as something or someone that can make it difficult for us to achieve our goal.

Remember the Obby that we created? It was a dynamic obby with platforms moving side to side at random speeds. Let's increase the difficulty.

Challenge

a. add code into your obby game so that clones are created in the sky and drop down on top of the platforms every random number of seconds. The clones are named "**skySpheres**".

a.1. the range of values for the random function is 1 second to 6 seconds.

b. after the random time has elapsed, we create more clones. The amount of clones created is also random. The range of values for the random function is between 5 and 15.

c. the goal now is to start by touching **"teleporter1"**, which will teleport you to the first platform. As you jump from one platform to another, clones that were created, called "**skySpheres**" fall from the sky. In order to get from one platform to the next, you must avoid the falling "**skySpheres**" or else the collision will cause you to fall off the platform.

Recall that a teleporter will catch your fall and teleport you back to safety. Recall that safety is any location that is close to the first teleporter, named "**teleporter1**"

Chapter 8. Animation Revisited

We have already done animation by making our infinity stones flash at different frequencies and at different colors. This was accomplished by adding a delay in between changing a property value. This was good but the animation was not a smooth transition from one state to another.

Roblox has a dedicated service that allows us to schedule animation and this service allows a smooth transition from one state to another. This service is called TweenService.

The ***first step*** is to get the service from Roblox and this is done using

```
local TweenService = game:GetService("TweenService")
```

> The above line of code retrieves a TweenService and stores a reference to it into a local variable called TweenService.

The ***second step*** is to specify the animation properties by defining the `TweenInfo` *** **NOTE The `TweenInfo` defines the animation sequence, NOT how a part will be animated. This is an important difference.** *** The `TweenInfo` will define the duration of the animation and if the animation should run once or multiple times.

According to Roblox, the TweenInfo takes inputs as parameters. **DO NOT WRITE THE CODE BELOW. The code below is just for study.**

```
TweenInfo.new
(
    number time = 1.0,
    Enum easingStyle = Enum.EasingStyle.Quad,
    Enum easingDirection = Enum.EasingDirection.Out,
    number repeatCount = 0,
    bool reverses = false,
    number delayTime = 0

)
```

Properties

`number TweenInfo.`**`Time`** – The duration of the tween animation. The unit is seconds.

`Enum TweenInfo.`**`EasingDirection`** – The direction in which the EasingStyle executes.

`number TweenInfo.`**`DelayTime`** – The amount of time that elapses before the tween animation starts again. The unit is seconds.

`number TweenInfo.`**`RepeatCount`** – The number of times the tween repeats after tweening once.

`Enum TweenInfo.`**`EasingStyle`** – The style in which the tween executes.

`bool TweenInfo.`**`Reverses`** – Whether or not the tween does the reverse once the inital tween is completed.

The ***third step*** is to define how the part will be animated and then create the Tween. Remember that animation is based on a transition from a start state to an end state. The third step is to define the end state.

1. For example, when the game begins a platform can have size 10 in the x position (the start state). The end state would be to have the platform change its size to 100 in the x position.

2. By Tweening this, the platform would grow by 10 in a smooth animation sequence.

```
Create ( Instance instance , TweenInfo tweenInfo , Dictionary propertyTable )
```

The first parameter specifies the part that we want to animate.

The second part describes the animation sequence, such as duration.

The third parameter specifies the property values that will be changed to create the animation.

1. this third parameter specifies the end state. Animation can be described as the transition from begin state to end state.

2. for example, if we want a sky platform to move side to side, the end state would specify the x position of our platform before it bounces back.

Task

a. insert a new cube into your game and call it "**tweenPlatform**". Next, insert a script into this new cube.

b. write the code below ***exactly*** into your script. After you are done typing in the code ***exactly***, then press F5

b.1. what is the start state of **"tweenPlatform"**?

b.2. what is the end state of **"tweenPlatform"**?

```
print(”Hello world!”)


--local myBlock = script.Parent

local part     = Instance.new(”Part”)
part.Position = Vector3.new(0, 30, 0)
part.Size     = Vector3.new( 20, 20, 20 )
part.Color    = Color3.new(1, 0, 0)

-- ============================================== anchored
-- true makes it float in air
-- false makes it grounded
-- ==============================================
part.Anchored = true
--part.Anchored = false

--part.Parent   = game.Workspace
part.Parent   = script.Parent


local TweenService = game:GetService(”TweenService”)
-- tween info
--     4, Time of 4 ==> the animation duration is 4 seconds
--
--
--    -1, RepeatCount of -1 ==> repeat forever
-- true, ReverseCount of true ==> toggles between start and end state
--    .5, DelayTime of .5 ==> wait .5 seconds and start the Tween again

local tweenInfo    = TweenInfo.new(
      4,
      Enum.EasingStyle.Linear,
      Enum.EasingDirection.Out,
      -1,
      true,
      .5

)

-- property table. This is the end state
 local goal   = {}
goal.Position = Vector3.new( 100, 30, 0 )
goal.Color    = Color3.new( 1, 1, 0 )

wait( 10 )
print( ”about to create TweenService” )
local tweenCon = TweenService:Create( part, tweenInfo, goal )
print( ”done with create TweenService” )

tweenCon:Play()
```

Super Challenge

a. looking at the code given above, only position and color are changed.

b. ***modify the goal*** so that multiple properties are animated.

b.1. ***modify the goal*** so that your "**tweenPlatform**" moves side to side

b.2. ***modify the goal*** so that your "**tweenPlatform**" changes shape