

ChatGPT

Share



10/22/2024

You said:

what coding platform does roblox use?\

ChatGPT said:

ChatGPT

Roblox uses a programming language called **Lua** for scripting. Lua is lightweight and easy to learn, making it accessible for both beginners and experienced developers. In Roblox, Lua scripts are used to control game mechanics, create interactions, and manage various elements within the game. The Roblox Studio is the primary development environment where users can create and edit their games, utilizing Lua for scripting.

4o mini

You said:

Do you believe a game like this is achieve-able: Obby Race Color-Made on release and ready to be released Color-shortly after release, probably 1 week after Color-3rd update Color-Developing, before release, important skills to know We make 10 or more maps that players compete to finish to win tokens On release we only use 6 of the maps so that we have 2 updates worth of content for later Coins can be used in shop so players can buy cosmetics and rank up Shoes and trails will be implemented in the game to give players a boost in speed, jump, and dash power These stats will flow nicely until the final 6 where we will make it so one of the 3 stats given will outweigh all the others, this will make play styles different and players will be able to feel more invested in the game With every rank comes special items players can buy to give them a head start or power up Players who are rank 20 or up are instant TP'ed to a pro lobby That can be another update about a month or so Pro lobbies give more coins per win Win's give 100 gold 1st 100 gold 2nd 70 gold 3rd 50 gold Losers 30 gold Rank 0, everyone starts with it Rank 1, 5 wins and 200 gold every rank after that is 2X the last rank up Ex, {rank 2, 10 wins and 400 gold}{rank 3, 20 wins 800 gold}{rank 4, 40 wins 1600 gold} We can balance that once we get to like rank 50 which will be max stat wise, but players will be able to rank up beyond 50 for titles and cosmetic rewards and be able to be on Leader board So were not good enough scripters and builders for this but cosmetics can have stats or something in the future 2x win events every weekend Updates every positive weekend/every 2nd weekend Saturday at 3 pm Key things to build/script/learn Shop system UI Obbies Cosmetics Rank system Win system Gold system Game pass system Updating Server differentiation (rank 19 and lower in one server— rank 20 and above in another) Player stats scaling (Jump, speed) Bug fixing/prevention... Global announcement system (maybe) Robux items Activate spawn points (99 RS) Extra jump power (399 RS) Faster rank ups, about 30% less needed (599 RS, stacks with VIP) VIP (daily coin chest, rainbow title, 20% less to rank up)(499 RS Stacks with faster

rank ups) Coin packs 100 coins (99 RS) 500 coins (399 RS) 1000 coins (799 RS) 2300 coins(1299 RS)

ChatGPT said:

ChatGPT

Creating a game like **Obby Race** on Roblox is definitely achievable, especially with the right planning and execution. Your concept has a solid foundation, and here are some insights on the different components of your plan:

Key Features and Mechanics

1. **Map Design:** Starting with six maps and planning for future updates is a smart approach. This allows you to focus on quality while ensuring content availability post-launch. Consider using varied themes and challenges in each obby to keep players engaged.
2. **Token and Cosmetic System:** Implementing a token system where players earn coins from winning races is a great motivation for competition. Offering cosmetics with various stats adds depth, making players feel more invested in their choices.
3. **Stat System:** The idea of different stats (speed, jump, dash) affecting gameplay and play styles is excellent. Make sure to balance these carefully to avoid making one stat overwhelmingly powerful.
4. **Ranking System:** The progression system you've outlined is straightforward and encourages players to keep competing. Just be sure to adjust the requirements for ranks beyond 50 to maintain engagement and not create frustration.
5. **Pro Lobby:** This adds an element of exclusivity and encourages players to strive for higher ranks, which can enhance long-term engagement.

Development Considerations

1. **Scripting and Building Skills:** If your team feels unsure about their scripting or building skills, consider:
 - **Learning Lua:** Many resources and tutorials are available for learning Lua scripting specifically for Roblox.
 - **Using Free Models:** You can find community-created models and scripts in the Roblox library to speed up development, but ensure you understand how they work for effective integration.
2. **Key Systems to Develop:**
 - **Shop System:** Design a simple and intuitive UI for players to purchase cosmetics and items.
 - **UI/UX:** Focus on clean and appealing interfaces, especially for menus and leaderboards.
 - **Obby Challenges:** Make sure your obby courses are fun, balanced, and have a good learning curve.

- **Rank and Win System:** Implement systems that accurately track player wins and rank progression.
- 3. **Bug Fixing and Testing:** Continuous testing is essential. Make sure to have a closed beta phase before the full launch to identify bugs and gather feedback.
- 4. **Server Management:** Implement server differentiation for ranks to ensure competitive fairness.
- 5. **Event Planning:** Organizing 2x win events and regular updates will keep the community engaged and give players something to look forward to.

Community Engagement

- **Feedback Loop:** Encourage players to give feedback on updates and gameplay, as this can provide valuable insights and help refine the game.
- **Promotions and Events:** Plan seasonal or special events to maintain player interest.

Final Thoughts

Your vision for **Obby Race** is well thought out and could attract a player base if executed effectively. Focus on iterative development—launch with a solid core game and build from there based on player feedback and your team's capabilities. This approach will help ensure that you maintain quality while expanding the game. Good luck!

4o mini

You said:

could you possibly make a code that would make this whole entire game?

ChatGPT said:

ChatGPT

Creating a full game like **Obby Race** would require a considerable amount of code and design work, far beyond what can be provided in a single response. However, I can outline the essential components and provide snippets of Lua code for key features that you can build upon. Here's a breakdown of the core components you'll need to implement, along with some example scripts to get you started.

Core Components

1. **Map Creation**
 - Design your obby maps using Roblox Studio.
2. **Token and Coin System**
 - Track player wins and coins earned.
3. **Ranking System**
 - Manage player ranks and requirements for ranking up.
4. **Shop System**

- Allow players to purchase cosmetics and upgrades.
- 5. **UI System**
 - Display player stats, coins, and ranks.
- 6. **Pro Lobby System**
 - Separate servers for different rank groups.
- 7. **Event System**
 - Implement double coin events and regular updates.

Example Code Snippets

Here's some example Lua code to get you started on different parts of your game:

1. Token and Coin System

This script tracks wins and coins for each player.

lua

Copy code

```
-- Server Script to track player wins and coins
local Players = game:GetService("Players")

Players.PlayerAdded:Connect(function(player)
    player:SetAttribute("Wins", 0)
    player:SetAttribute("Coins", 0)
end)

function addWin(player)
    local currentWins = player:GetAttribute("Wins")
    player:SetAttribute("Wins", currentWins + 1)

    -- Award coins based on rank
    local rankCoins = getCoinsForRank(player)
    local currentCoins = player:GetAttribute("Coins")
    player:SetAttribute("Coins", currentCoins + rankCoins)
end

function getCoinsForRank(player)
    local rank = player:GetAttribute("Rank")
    if rank == 0 then return 30
    elseif rank == 1 then return 100
    elseif rank == 2 then return 70
```

```
elseif rank == 3 then return 50
else return 30 -- Default for losers
end
end
```

2. Ranking System

This code snippet defines how players rank up.

lua

Copy code

```
-- Server Script to handle ranking
function rankUp(player)
    local wins = player:GetAttribute("Wins")
    local rank = player:GetAttribute("Rank")
    local rankRequirements = {0, 5, 10, 20, 40} -- Add more as needed

    if wins >= rankRequirements[rank + 1] then
        player:SetAttribute("Rank", rank + 1)
        -- Notify player about rank up
        player:SendNotification("Congratulations! You've ranked up to
Rank " .. (rank + 1))
    end
end

-- Call this function after a player wins
addWin(player)
rankUp(player)
```

3. Shop System

This snippet implements a basic shop.

lua

Copy code

```
-- Server Script for shop system
local ShopItems = {
    {Name = "Extra Jump", Cost = 399},
    {Name = "Speed Boost", Cost = 199},
```

```

        -- Add more items as needed
    }

function purchaseItem(player, itemName)
    local coins = player:GetAttribute("Coins")

    for _, item in pairs(ShopItems) do
        if item.Name == itemName and coins >= item.Cost then
            player:SetAttribute("Coins", coins - item.Cost)
            -- Give the item to the player
            giveItemToPlayer(player, itemName)
            return "Purchase successful!"
        end
    end
    return "Not enough coins or item not found."
end

function giveItemToPlayer(player, itemName)
    -- Logic to add item to player's inventory
    print(player.Name .. " purchased " .. itemName)
end

```

4. UI System

Create a simple UI for showing stats.

lua

Copy code

```

-- Local Script for UI
local player = game.Players.LocalPlayer
local PlayerGui = player:WaitForChild("PlayerGui")

local statsFrame = Instance.new("ScreenGui", PlayerGui)
local winsLabel = Instance.new("TextLabel", statsFrame)
winsLabel.Position = UDim2.new(0.1, 0, 0.1, 0)
winsLabel.Size = UDim2.new(0, 200, 0, 50)
winsLabel.Text = "Wins: " .. player:GetAttribute("Wins")

player:GetAttributeChangedSignal("Wins"):Connect(function()

```

```
winsLabel.Text = "Wins: " .. player:GetAttribute("Wins")
end)
```

Next Steps

1. **Building:** Design your obby maps in Roblox Studio, ensuring they are fun and challenging.
2. **Testing:** Continuously test your scripts to ensure everything works as expected.
3. **Iteration:** Gather feedback from players and make adjustments to gameplay, balancing, and UI.

Additional Resources

- **Roblox Developer Hub:** The official documentation contains detailed guides and tutorials on scripting and game design.
- **YouTube Tutorials:** There are many video tutorials available that can provide visual aids in learning how to create games on Roblox.

While these snippets provide a starting point, developing a full game will require additional coding, design, and refinement. Good luck with your project! If you have more specific features or components you want to implement, feel free to ask!

4o mini

You said:

lua makes it so your able to create physical objects through code correct?

ChatGPT said:

ChatGPT

Yes, Lua allows you to create and manipulate physical objects in Roblox through scripting. In Roblox, you can create parts, models, and other objects using code to dynamically generate and modify elements in your game world. Here's a brief overview of how this works:

Creating Objects with Lua in Roblox

Creating Parts: You can create parts (basic building blocks in Roblox) using the `Instance.new` function. This allows you to define properties such as size, color, and position.

lua

Copy code

```
-- Create a new part
local newPart = Instance.new("Part")
newPart.Size = Vector3.new(4, 1, 2) -- Width, Height, Depth
```

```
newPart.Position = Vector3.new(0, 10, 0) -- X, Y, Z position in the
game world
newPart.Color = Color3.new(1, 0, 0) -- RGB color (red in this case)
newPart.Anchored = true -- Prevents the part from falling
newPart.Parent = game.Workspace -- Add the part to the Workspace so it
appears in the game
```

1.

Creating Models: You can group multiple parts together to create a model. This is useful for complex objects like buildings or vehicles.

lua

Copy code

```
-- Create a model and add parts to it
local myModel = Instance.new("Model")
myModel.Name = "MyModel"
myModel.Parent = game.Workspace

-- Create and add parts to the model
local part1 = Instance.new("Part")
part1.Size = Vector3.new(4, 1, 4)
part1.Position = Vector3.new(0, 0, 0)
part1.Parent = myModel -- Adding to the model

local part2 = Instance.new("Part")
part2.Size = Vector3.new(2, 2, 2)
part2.Position = Vector3.new(0, 2, 0)
part2.Parent = myModel -- Adding to the model
```

2.

Manipulating Existing Objects: You can also manipulate existing objects in the game by referencing them and changing their properties.

lua

Copy code

```
-- Change the color of an existing part
local part = game.Workspace:FindFirstChild("MyPart") -- Assuming
"MyPart" exists
if part then
    part.Color = Color3.new(0, 0, 1) -- Change to blue
end
```


3.

Physical Properties: You can adjust physical properties like mass, friction, and bounciness to affect how objects interact in the game.

lua

Copy code

```
local part = Instance.new("Part")
part.Size = Vector3.new(4, 1, 4)
part.Position = Vector3.new(0, 10, 0)
part.Anchored = false -- Make it fall under gravity
part.Material = Enum.Material.Bouncy -- Make it bouncy
part.Parent = game.Workspace
```

4.

Key Concepts

- **Instances:** Everything in Roblox is an instance of a class (like `Part`, `Model`, etc.), and you use `Instance.new` to create them.
- **Properties:** Each object has properties you can modify, such as size, position, color, and physical properties.
- **Parenting:** Objects must be added to the game hierarchy (usually as children of `Workspace`) to appear in the game.

Conclusion

With Lua scripting in Roblox, you have powerful tools to create and manage physical objects in your game, making it possible to build dynamic and interactive experiences. If you have specific scenarios or objects in mind that you want to create through code, feel free to ask!

4o mini

10/24/2024

You said:

design one of the obbies that could be in my game based on the contents in the long paragraph text from the beginning

ChatGPT said:

ChatGPT



Here's a visual concept of the obby race, designed with floating platforms, obstacles, and vibrant colors as described. It captures the dynamic and challenging course players would navigate, with different platforms, jumps, and in-game boosts. Let me know if you'd like adjustments or further ideas for the design!

You said:

can you make a code that will give me platforms like the ones in the image?

ChatGPT said:

ChatGPT

Certainly! Below is a Lua script that can create basic platforms similar to the ones in the image, with different colors, sizes, and movement for variety. You can add this script to Roblox Studio to generate floating platforms for your obby:

Lua Code to Create Floating Platforms

lua

Copy code

```
-- Function to create a platform
function createPlatform(position, size, color, canMove, moveDirection,
moveDistance, moveSpeed)
    local platform = Instance.new("Part")
    platform.Size = size
    platform.Position = position
```

```

platform.Color = color
platform.Anchored = true -- Keeps the platform in place
platform.Parent = game.Workspace

-- If the platform should move, add movement behavior
if canMove then
    local moveTween = game:GetService("TweenService")
    local startPosition = platform.Position
    local endPosition = startPosition + moveDirection *
moveDistance

    -- Tween settings for movement
    local tweenInfo = TweenInfo.new(moveSpeed,
Enum.EasingStyle.Linear, Enum.EasingDirection.InOut, -1, true)

    -- Create the movement tween
    local moveTweenProperties = {Position = endPosition}
    local platformTween = moveTween:Create(platform, tweenInfo,
moveTweenProperties)
    platformTween:Play() -- Start the movement
end

return platform
end

-- Function to create multiple platforms for the obby
function createObbyPlatforms()
    -- Table of colors for platforms
    local colors = {
        Color3.new(1, 0, 0), -- Red
        Color3.new(0, 0, 1), -- Blue
        Color3.new(0, 1, 0), -- Green
        Color3.new(1, 1, 0)  -- Yellow
    }

    -- Create static and moving platforms
    createPlatform(Vector3.new(0, 10, 0), Vector3.new(10, 1, 10),
colors[1], false)

```

```

        createPlatform(Vector3.new(20, 10, 0), Vector3.new(10, 1, 10),
colors[2], true, Vector3.new(0, 0, 1), 10, 3) -- Moving platform
        createPlatform(Vector3.new(40, 10, 0), Vector3.new(8, 1, 8),
colors[3], false)
        createPlatform(Vector3.new(60, 10, 0), Vector3.new(12, 1, 12),
colors[4], true, Vector3.new(0, 0, 1), 15, 2) -- Moving platform
end

-- Call the function to create the obby
createObbyPlatforms()

```

Key Parts of the Script:

- **Static Platforms:** Platforms that remain anchored and don't move.
- **Moving Platforms:** Platforms that move back and forth in a specific direction using `TweenService`.
- **Parameters:**
 - **Position:** The position of the platform in the game world.
 - **Size:** The size of the platform (`Vector3.new(width, height, depth)`).
 - **Color:** The color of the platform (`Color3.new(r, g, b)`).
 - **canMove:** Whether the platform moves or not.
 - **moveDirection:** The direction the platform moves in (`Vector3.new(0, 0, 1)` for forward movement).
 - **moveDistance:** How far the platform moves.
 - **moveSpeed:** Speed of movement.

How to Use It:

1. **Add the Script:** Copy and paste the above script into a **Script** or **LocalScript** in Roblox Studio.
2. **Run the Game:** When you play the game, platforms will be generated at the specified positions. Two platforms will move, and two will remain static.
3. **Customize:** Adjust the position, size, color, and movement parameters to fit your obby design.

This should give you a good starting point for creating platforms like the ones in the image. You can extend it by adding more platforms and adjusting the behavior as needed!

4o

You said:

make a script for running, jumping, and dashing in lua

ChatGPT said:

ChatGPT

Here's a Lua script you can use in Roblox Studio to implement **running**, **jumping**, and **dashing** mechanics for players. This script will modify the player's movement behavior and add keybinds for these actions.

Running, Jumping, and Dashing Script

You can place this script inside a **LocalScript** within **StarterPlayerScripts**.

lua

Copy code

```
-- Variables
local player = game.Players.LocalPlayer
local character = player.Character or player.CharacterAdded:Wait()
local humanoid = character:WaitForChild("Humanoid")
local userInputService = game:GetService("UserInputService")
local runSpeed = 32 -- Default walking speed is 16
local dashDistance = 50 -- Distance covered when dashing
local dashCooldown = 1 -- Time in seconds before dash can be used again
local isDashing = false -- Whether the player is currently dashing
local canDash = true -- If the player can dash
local dashKey = Enum.KeyCode.LeftShift -- Key to dash
local runKey = Enum.KeyCode.LeftControl -- Key to run
local normalWalkSpeed = 16 -- Default walk speed
local jumpPower = 50 -- Set higher jump power for more height

-- Enable running when runKey is pressed
userInputService.InputBegan:Connect(function(input)
    if input.KeyCode == runKey then
        humanoid.WalkSpeed = runSpeed
    end
end)

-- Return to normal speed when runKey is released
userInputService.InputEnded:Connect(function(input)
    if input.KeyCode == runKey then
        humanoid.WalkSpeed = normalWalkSpeed
    end
end)
```

```

        end
    end)

    -- Dash mechanic
    userInputService.InputBegan:Connect(function(input)
        if input.KeyCode == dashKey and canDash and not isDashing then
            isDashing = true
            canDash = false

            -- Get the current direction the player is moving
            local moveDirection = humanoid.MoveDirection
            if moveDirection.Magnitude > 0 then
                -- Make the player dash in the current direction
                local dashVector = moveDirection * dashDistance
                local newPosition = character.PrimaryPart.Position +
dashVector
                humanoid:MoveTo(newPosition)
            end

            -- Dash cooldown
            wait(dashCooldown)
            canDash = true
            isDashing = false
        end
    end)

    -- Adjust jump power for enhanced jumping
    humanoid.JumpPower = jumpPower

    -- Debug function for showing when a dash is active
    humanoid.DashingChanged:Connect(function(isDashing)
        if isDashing then
            print("Player is dashing!")
        else
            print("Player is not dashing!")
        end
    end)
end)

```

Key Mechanics

1. **Running:**
 - **Key:** Left Control (`runKey`).
 - **Effect:** Increases the player's walk speed from `16` (default) to `32`.
2. **Jumping:**
 - The default jump in Roblox can be modified by changing the humanoid's `JumpPower`. In this case, it's set to `50` for a higher jump.
3. **Dashing:**
 - **Key:** Left Shift (`dashKey`).
 - **Effect:** Allows the player to dash forward in the direction they're moving. The dash covers a distance of `50` studs and has a cooldown of 1 second.
 - The `dashDistance`, `dashCooldown`, and key binding can be adjusted as needed.

How It Works:

- **Running:** The script listens for when the player presses and releases the run key (Left Control) and adjusts their walk speed.
- **Jumping:** The jump height is set higher by adjusting the humanoid's `JumpPower`.
- **Dashing:** When the dash key (Left Shift) is pressed, the script checks if dashing is available (to prevent spamming), then moves the player in the direction they are facing using `humanoid.MoveTo()`.

Usage:

1. **Place this script:** Insert it into a **LocalScript** inside the `StarterPlayerScripts` folder in Roblox Studio.
2. **Adjust as needed:** You can tweak the dash distance, speed, cooldown, and jump power to match your game mechanics.

This setup gives players enhanced movement options, which should work well for an obby game with dynamic and fast-paced gameplay!

4o

You said:

add a double jump feature

ChatGPT said:

ChatGPT

To add a **double jump** feature alongside the existing running, jumping, and dashing mechanics, we can modify the script so that the player can jump twice before touching the ground again.

Here's an updated version of the script that includes **double jumping**:

Running, Jumping, Dashing, and Double Jump Script

lua

Copy code

```
-- Variables
local player = game.Players.LocalPlayer
local character = player.Character or player.CharacterAdded:Wait()
local humanoid = character:WaitForChild("Humanoid")
local userInputService = game:GetService("UserInputService")
local runSpeed = 32 -- Default running speed
local dashDistance = 50 -- Distance covered when dashing
local dashCooldown = 1 -- Time in seconds before dash can be used
again
local isDashing = false -- Whether the player is currently dashing
local canDash = true -- If the player can dash
local dashKey = Enum.KeyCode.LeftShift -- Key to dash
local runKey = Enum.KeyCode.LeftControl -- Key to run
local normalWalkSpeed = 16 -- Default walk speed
local jumpPower = 50 -- Regular jump power
local canDoubleJump = true -- Whether the player can double jump
local hasDoubleJumped = false -- Whether the player has already double
jumped
local jumpCount = 0 -- Tracks how many jumps the player has made
local maxJumps = 2 -- Maximum jumps (1 regular + 1 double jump)

-- Enable running when runKey is pressed
userInputService.InputBegan:Connect(function(input)
    if input.KeyCode == runKey then
        humanoid.WalkSpeed = runSpeed
    end
end)

-- Return to normal speed when runKey is released
userInputService.InputEnded:Connect(function(input)
    if input.KeyCode == runKey then
        humanoid.WalkSpeed = normalWalkSpeed
    end
end)
```



```

-- Dash mechanic
userInputService.InputBegan:Connect(function(input)
    if input.KeyCode == dashKey and canDash and not isDashing then
        isDashing = true
        canDash = false

        -- Get the current direction the player is moving
        local moveDirection = humanoid.MoveDirection
        if moveDirection.Magnitude > 0 then
            -- Make the player dash in the current direction
            local dashVector = moveDirection * dashDistance
            local newPosition = character.PrimaryPart.Position +
dashVector
            humanoid:MoveTo(newPosition)
        end

        -- Dash cooldown
        wait(dashCooldown)
        canDash = true
        isDashing = false
    end
end)

-- Adjust jump power for enhanced jumping
humanoid.JumpPower = jumpPower

-- Double Jump mechanic
local function onJumpRequest()
    if jumpCount < maxJumps then
        humanoid:ChangeState(Enum.HumanoidStateType.Jumping)
        jumpCount = jumpCount + 1
    end
end

-- Reset jump count when player lands
humanoid.StateChanged:Connect(function(_, newState)
    if newState == Enum.HumanoidStateType.Landed then

```

```

        jumpCount = 0
        hasDoubleJumped = false
    end
end)

-- Detect when the jump input is pressed
userInputService.JumpRequest:Connect(onJumpRequest)

-- Debug function for showing when a dash is active
humanoid.DashingChanged:Connect(function(isDashing)
    if isDashing then
        print("Player is dashing!")
    else
        print("Player is not dashing!")
    end
end)
end)

```

New Feature: Double Jump

1. Double Jump Logic:

- Players can now jump twice before they need to land.
- The jump count is tracked using the variable `jumpCount`, and it resets when the player touches the ground.
- The `maxJumps` variable is set to `2`, allowing one regular jump and one double jump.

2. Jump Request Event:

- The `UserInputService.JumpRequest` event listens for when the player attempts to jump, and if they haven't exceeded their jump limit, they can jump again (i.e., double jump).

3. Reset on Landing:

- When the player touches the ground (`Landed` state), the jump count resets, allowing the player to double jump again on their next jump.

How to Use:

1. **Place the Script:** Add the script to a **LocalScript** inside the `StarterPlayerScripts` folder.
2. **Test:** When you run the game, the player should now be able to run, dash, jump, and double jump.

Tuning:

- **maxJumps = 2**: You can modify this variable to allow for triple jumps or more if you like.
- **jumpPower**: Adjust this to increase or decrease the height of jumps, including the double jump.

This script should now give you a running, dashing, and double jumping system for your game!

40

ChatGPT can make mistakes. Check important info.

?