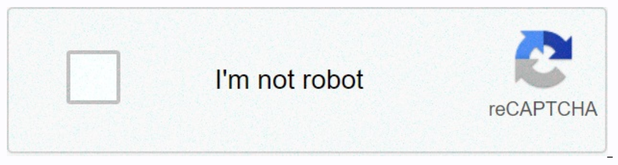

Continue

Roblox studio how to make a group door

5 min In some games, it's desirable to have doors or compulsors that certain players can pass from others cannot. For example, numerous combat games will spawn players from each team in a safe area that other teams can't access (so they don't attack the moment they join or spawn). There are several ways to accomplish this, but this tutorial uses Article/Collision Filtering|collision filtering to create threshold that only certain players can pass through. Create the threshold for this simple example, the thresholds are basic block parts of space called BlueDoor and Reddoor. Materials are Enum/Materials | Glass and The BazPart / Transparency | Transparency set to 0.5. Group collisions since these doors exist in the gaming world at the beginning, we can set up the collision group immediately. In ServerScriptService, create a Script and paste the following code into it: Now we need to add spawning players to a suitable collision group. There are several ways to assign individual players to a group, including: Give play a color based on their items/ Play Spawns and Teams|flow teams. Check if the player has earned a special item/Special Badges Special Items|badges or purchase items/Games Pass One Tumor Purchases |pass game that allows them to access a restricted area. For the simplicity of this tutorial, we'll just give all incoming players to the blue team and give them access to the blue door, but not the red door. Collisions Groups like and Gates, Let's first set up collision group play by adding the following lines to the script: When adding the players to collision bands, it's critical to remember that all parts of the characters must be allowed to pass through the door - head, torso, arms, feet, feet, etc. We can do this with a function that finds all BasePart objects, the base classes of both parties | Party with MeshPart | MeshParts that make up the avatar play: When span characters, body parts are assembled over the course of about one second, so this loop functions in the first children and uses the Example / DescendantAdded | DesandanAdded events are detected when other parties are added. This ensures that all character parts receive the collision group's placement collision. With this function in place, we can detect the Players/PlayerAdded events that then listen for the Player/CharacterrAdded | CharacterAdded events and run our setCollisionGroup() functioning with group collision bluePlayers. Lastly, we just need to define the collision groups that are not collisions with each other – especially, blue players should not collision with blue doors, and red players should not collision with red doors. Uh oh! Your browser does not appear to support embedded videos! Here is a direct link to the video instead. Tags: So we all know about these classic bands only doors that all players who touch them, should not be in the relevant group. This is 2020! As scripters, we have some incredible mechanics at our disposal to create some well developed goods that carry out the same goal. And that's what we're going to do. In this thread, we will use the Collision section of the Physical Services to create a sway threshold that is always group-only. Something that I personally haven't seen much. Understanding collisions to begin with, we need to find ourselves around how Roblox deals with collisions. Firstly, we're more professional than the standard CanCollide property attached with the BasePart class. Below you can see a video by Roblox demonstrating some of the collisions we'll be working with: Notice how only some of the spheres pass through the block. We'll use this exact logic to create our team only doors, sphere replacement for player characters and blocks for our doors. If you are familiar with collisions on Roblox (using the Physevise), please familiarity yourself with this article before we continue: Let's attack how we can stop some players going through a door and how to leave some of. Well, we can create a single block that stops all players and manually puts the collisons off for play in the right group. So our algorithm looks something like this: Luckily, Roblox takes care of this for us through the use of the Collisons grouping. So all of our developers need to make it tell Roblox what characters they leave in, and who doesn't. Now we have the basic locale of how our script will work, we can start to deal with things like data structure and planning exactly how our algorithm will work – considering all the cases we can think of: two or more doors per group, the player of more than one applicable group etc. Set out exactly what we need to start up the collisions for the doors: and set out exactly what we need to start up the collision for the players: We are now ready to develop our doors, know exactly what we need. Set up our doors as always, before we can start scripted we need to make sure our Explorer is in order and has everything we need. I will go ahead and affirm official door model created by Roblox with our balanced font. If you'd like to create this door yourself, you can see the following videos created by Roblox: Watch Video 1. Making a door: vertical Hinge 2. Make a threshold: Limit 3. Have a door: Closed and springs so we have our doors to the studios now. Explorers I Now let's make our edits. We need to consider many cases making our script truly robust, so let's say we publish us so anyone can use it. We'll need some configuration then - the only editable property other developers really need to change will be the GroupId. Explorers me Finally, let's create this part that will stop all players from walking through the doors. By making this part transparent and anchored, we can ensure that only whitelisted players can travel to this by scripting the collisions as per our algorithm above. My explorers For incorporating the case into multiple doors, it is necessary to organize the doors like them that they have a common parent in the Explorer. To make things clean, I created a folder called TeamDoors which also allows us to easily iterate through all the doors of one for loop. My Explorers Scripting Finally we can start writing our code. We will firstly attack up our collision groups for our doors; check whether the group assigning to the gate already has a group collision created before another one is made. -- Service Roblox service local physicalSevis = game: GetService(PhysicsErvice) – Check to see if a Group collision already exists Function CollisionGroupExist(id) Local CollisionGroups = PhysicalSevice: GetCollisionGroups() For i = 1, #CollisionGroups Done If CollisionGroups[i].name = GroupDoor_ . ID Then return true End Return False End – Initialise Group Only Collision Threshold for _, Door to Pair(workspace: FindFirstChild(TeamDoors, true): GetChildren()) perform local setup = Door.Configuration if not CollisionGroupExists(Door.Configuration.GroupId.Value) Then PhysicalSeviservis: CreateCollisionGroup(GroupDoor_ . Configuration.GroupId.Value) Ends Next, we need to put the player characters to have their own collision group: - Array to store all groups that have a local group-door ID ={} - Initialise Play Collision Game. Players.PlayerAdded:Connect(Function (Player) for i, The pair id(id) is done if play: IsInGroup(IDs[i]) Then Player.Characteradded:Connect(Function(Character) To _ , Part of Pair(Character: GetDescendants()) do if Part: IsA(BasePart) Then PhysicsErvice: SetPartCollisionGroup(Part, Players_ . Id[i]) ending) You'll notice that we just added a player's character to an existing collision group without his first initiative, this is because we need to make an editor of the original function: for _, Door in pair (workspace: FindFirFirChild(TeamDoors, true): GetChildren()) perform local configuration = Door.Configuration if not CollisionGroupExists(Door.Configuration.GroupId.Value) Then PhysicalSeviservis: CreateCollisionGroup(GroupDoor_ . Configuration.GroupId.Value) PhysicsService: CreateCollisionGroup(Players_ . Configuration.GroupId.Value) PhysicalSevis: CollisionGroupSetCollidable(GroupDoor_. Configuration.GroupId.Value, Player_. Configuration.GroupId.Value, false) End PhysicalSevice : SetPartCollisionGroup(Pot.CollisonsHandler, .. Configuration.GroupId.Value) Finish This Conclusion! In less than 50 lines of rope, we created a truly sleepy and modern take on the classic group-only mechanical doors. Feel free to add any and all features like killing the player when they are not allowed in or teleporte pads or anything you may think of! I hope you enjoyed this tutorial, -Tom Full Code - Script by ThomasMGardiner – Roblox local service PhysicalService = game: GetService (PhysicalService) - System variable lost local ID = {} - Check to see if a Group Collision already exists function CollisionGroupExists(id) local CollisionsGroups = PhysicalService: GetCollisionGroups() for i = 1, #CollisionsGroups done if CollisionsGroups[i].name = GroupDoor_.. ID Then return true end return false end – Initialise Group Only Collision Doors for Me, Threshold(workspace: FindFirstChild(TeamDoors, true): GetChildren()) Do-Initialise Door-Play Local Collision Configuration = Door.Configuration If Not CollisionGroupExist(Door.Configuration.GroupId.Value) Then IDs[i] = Configuration.GroupId.Value Physicservice : CreateCollisionGroup(GroupDoor_ . Configuration.GroupId.Value) PhysicsService: CreateCollisionGroup(Players_ . Configuration.GroupId.Value) PhysicalSevis: CollisionGroupSetCollidable(GroupDoor_. Configuration.GroupId.Value, Players_. Configuration.GroupId.Value, false) End game. PhysicalService: SetPartCollisionGroup(Door.TeamHandler, GroupDoor_.. Configuration.GroupId.Value) End - Initialise Play collision game. Players.PlayerAdded:Connect(Function (Player) for i, The pair id(id) is done if Player:IsInGroup(IDs[i]) Then Player.Characteradded:Connect(Function(Character) Print(hi) To _, Part of Pair(Character: GetDescendants()) do if Part: IsA(BasePart) Then PhysicsErvice: SetPartCollisionGroup(Part, Players_ . Id[i]) Ending) EndIng) End)