GOREBOX MODDING DOCUMENTATION
VERSION 15.0.0 BETA 4

Basic Commands

OpenConsole

Description: Opens the mod console for user interaction.

Usage: OpenConsole

CloseConsole

Description: Closes the mod console, ending user interaction.

Usage: CloseConsole

Physics Commands

AddRigidbody

Description: Adds a Rigidbody component to an object, enabling physics interactions. If the object already has a Rigidbody, it registers the existing one for script **Usage**.

Returns: true if a new Rigidbody was added, false if an existing one was found.

Usage: AddRigidbody

AddForce(x, y, z, mode)

Description: Applies a force to a registered physics object. The force's nature is determined by the specified mode.

Parameters:

x, y, z: Force vector components.

mode: Defines the ForceMode applied.

Usage: AddForce({"x": 0, "y": 0, "z": 0, "mode": 0})

Mode Explanation:

0 - Force: Adds a continuous force to the Rigidbody, considering its mass.

1 - Acceleration: Applies a continuous acceleration, ignoring mass.

2 - Impulse: Imparts an instantaneous force impulse, using mass.

3 - VelocityChange: Causes an instant velocity change, mass-independent.

RemoveRigidbody

Description: Removes the Rigidbody component from an object, effectively disabling physics interactions.

Returns: true if the Rigidbody was successfully removed, false if no Rigidbody was found.

Usage: RemoveRigidbody

🔗 Parenting & Movement

About Parenting

Parenting in GoreBox involves linking one object (the child) to another (the parent), causing the child to follow the parent's movements. This principle is fundamental in constructing complex objects and character rigs. Typically, only the parent object requires a Rigidbody, as it governs the movement of its children. Adding a Rigidbody to a child object is generally not recommended unless a joint is specifically attached, as it may lead to unpredictable physics interactions.

About Local Positioning

The local positioning of an object is always relative to its parent. Setting an object's local position and rotation to zero aligns it precisely with its parent.

SetParent(parent)

Description: Sets the parent of the current object to another specified, registered object.

Usage: SetParent({"name": "parentName"})

Position

Move(x, y, z)

Description: Moves the object in a specified global direction.

Usage: Move({"x": 0, "y": 0, "z": 0})

MoveLocal(x, y, z)

Description: Moves the object relative to its and its parent's rotation.

Usage: MoveLocal({"x": 0, "y": 0, "z": 0})

SetPosition(x, y, z)

Description: Sets the object's global position.

Usage: SetPosition({"x": 0, "y": 0, "z": 0})

SetLocalPosition(x, y, z)

Description: Sets the object's position relative to its parent's position.

Usage: SetLocalPosition({"x": 0, "y": 0, "z": 0})

Rotation

SetRotation(x, y, z)

Description: Sets the object's global rotation.

Usage: SetRotation({"x": 0, "y": 0, "z": 0})

SetLocalRotation(x, y, z)

Description: Sets the object's rotation relative to its parent's rotation.

Usage: SetLocalRotation({"x": 0, "y": 0, "z": 0})

Rotate(x, y, z)

Description: Rotates the object in a specified global direction.

Usage: Rotate({"x": 0, "y": 0, "z": 0})

RotateLocal(x, y, z)

Description: Rotates the object relative to its parent's rotation.

Usage: RotateLocal({"x": 0, "y": 0, "z": 0})

Scale

SetScale(x, y, z)

Description: Sets the object's scale, adjusted by its parent's scale.

Usage: SetScale({"x": 1, "y": 1, "z": 1})

AddScale(x, y, z)

Description: Modifies the object's scale relative to its parent's scale. Use negative values to reduce scale.

Usage: AddScale({"x": 1, "y": 1, "z": 1})

📎 Event Listeners

Important Setup Instructions

To enable event listeners and call events, the following loop must be placed at the end of your script. Events will only be triggered after entering this loop:

```
while true
  while _events.len > 0
    _nextEvent = _events.pull
    _nextEvent.invoke(_nextEvent.args)
  end while
  yield
end while
```

Examples of Event Listener

Adding a Listener

Description: Script will start calling player functions.

Usage: ListenForPlayer

Removing a Listener

Description: Script will stop calling player functions.

Usage: EndListenForPlayer

Adding a Function

Description: Executes code when the player takes damage, if "ListenForPlayer" is active. Accesses the damage key.

Example Function:

OnLocalTakeDamage = function (args)

```
    damageAmount = args.damage
    print "Ouch! I took " + damageAmount + " damage!"
    if damageAmount < 5 then
        print "Could have been worse."
    else
        print "Still hurts."
    end if
end function
```

Event Functions

ListenForPlayer / EndListenForPlayer

Enables functions related to the local player's actions:

OnLocalTakeDamage(damage): Triggered when the player takes damage. Provides the damage amount.

OnLocalDied: Triggered when the player dies.

OnLocalInfected: Triggered when the player is infected or cured.

OnLocalHealed: Triggered when the player is fully healed.

OnLocalKnockout: Triggered if the player gets knocked out.

OnLocalWakeUp: Triggered when the player wakes up from being knocked out.

OnLocalSpawned: Triggered when the player (re)spawns.

OnLocalRagdoll: Triggered when the player enters ragdoll mode.

OnLocalGetUp: Triggered when the player gets up from ragdoll state.

OnLocalEmote(emoteName): Triggered when the player uses an emote. Provides the emote name.

OnLocalEmoteEnd: Triggered when an emote is ended.

ListenForServer / EndListenForServer

Enables functions related to server activities:

OnPlayerJoined(player): Triggered when a player joins the server. Provides the player's name.

OnPlayerLeft(player): Triggered when a player leaves the server. Provides the player's name.

OnKilledPlayer(player): Triggered when the local player kills another player. Provides the victim's name.

ListenForChat / EndListenForChat

Enables chat-related functions:

OnChatMessage(sender, message): Triggered when a chat message is received. Provides the sender's username and the message.

ListenForInventory / EndListenForInventory

Enables functions related to inventory actions:

OnItemSelect(slot): Triggered when an item is selected or equipped. Provides the slot number.

ListenForInstantiation / EndListenForInstantiation

Enables functions related to object instantiation:

OnInstantiate(player, name): Triggered when an object is spawned. Provides the spawner's name and the object's name.

ListenForSceneChange / EndListenForSceneChange

Enables functions related to scene changes:

OnSceneChanged(sceneName): Triggered when a scene (map/level) is loaded. Provides the scene name.

ListenForInput / EndListenForInput

Enables functions related to player input:

OnKeyPress(key): Triggered when a key is pressed. Provides the key name.

OnKeyHold(key): Triggered while a key is held. Provides the key name.

OnKeyLift(key): Triggered when a key is released. Provides the key name.

Objects Management

SetTexture(name)

Description: Retrieves the renderer and sets its material's texture to a loaded texture by its name.

Usage: SetTexture({"name": "myTexture"})

InstantiateEmpty(name, x, y, z, rotX, rotY, rotZ, rotW)

Description: Creates an empty object at a specified position and rotation. This object is automatically registered for use in commands like Destroy or ExecuteScript.

Usage: InstantiateEmpty({"name": "emptyObject", "x": 0, "y": 0, "z": 0, "rotX": 0, "rotY": 0, "rotZ": 0, "rotW": 0})

InstantiateEmptyChild(name, x, y, z, rotX, rotY, rotZ, rotW)

Description: Creates an empty child object of this object at a specific position and rotation relative to this object. This object is automatically registered for use in commands like Destroy or ExecuteScript.

Usage: InstantiateEmptyChild({"name": "emptyChildObject", "x": 0, "y": 0, "z": 0, "rotX": 0, "rotY": 0, "rotZ": 0, "rotW": 0})

InstantiateModelChild(model, name, x, y, z, rotX, rotY, rotZ, rotW, scale, collisions, convex)

Description: Instantiates a loaded model into the scene as a child of this object, with a chosen name, set position, rotation, scale, relative to this object, and optional collisions (which can be convex). The model is automatically registered.

Usage: InstantiateModelChild({"model": "myModel.fbx", "name": "myModel", "x": 0, "y": 0, "z": 0, "rotX": 0, "rotY": 0, "rotZ": 0, "rotW": 0, "scale": 1, "collisions": true, "convex": true})

InstantiateModel(model, name, x, y, z, rotX, rotY, rotZ, rotW, scale, collisions, convex)

Description: Instantiates a loaded model into the scene with a chosen name at a set position, rotation, scale, and optional collisions (which can be convex). The model is automatically registered.

Usage: InstantiateModel({"model": "myModel.fbx", "name": "myModel", "x": 0, "y": 0, "z": 0, "rotX": 0, "rotY": 0, "rotZ": 0, "rotW": 0, "scale": 1, "collisions": true, "convex": true})

Instantiate(name, path, x, y, z, rotX, rotY, rotZ, rotW)

Description: Creates an object with a given name from the game files at a specific position and rotation. This object is automatically registered for use in commands like Destroy or ExecuteScript.

Usage: Instantiate({"name": "spawnedDoll", "path": "AI/GoreDoll", "x": 0, "y": 0, "z": 0, "rotX": 0, "rotY": 0, "rotZ": 0, "rotW": 0})

Note: To avoid difficulties with object identification and unintended duplicates, it's recommended to use unique names and, in certain cases, to have only the Host spawn objects.

ExecuteScript(target, script)

Description: Executes an existing user script on the registered target object.

Usage: ExecuteScript({"target": "spawnedDoll", "script": "aiScript"})

ExecuteScriptOnChildren(target, script)

Description: Executes an existing user script on all children of this object.

Usage: ExecuteScriptOnChildren({"target": "spawnedDoll", "script": "aiScript"})

Destroy(name)

Description: Destroys a registered object.

Usage: Destroy({"name": "spawnedDoll"})

Explanation: Deletes the object named "spawnedDoll", provided it matches the registered name.

DestroySelf

Description: Destroys the current object.

Usage: DestroySelf

💬 Chat Functionality

Custom Commands

With the chat API features, you can add custom commands to enhance player interaction. Messages sent by players that start with a "?" are recognized as custom commands. These messages won't be sent into the general chat but will be captured by OnChatMessage if ListenForChat is active.

Example **Usage** of Custom Commands:

```
OnChatMessage = function (args)
  if message == "?die" then
    PlayerKill
  end if
end function
```

SendChatMessage(sender, message)

Description: Sends a chat message to all players using a specified sender name and message. This command supports rich text formatting.

Usage: SendChatMessage({"sender": "Bank", "message": username + " currently has " + cash + "GB$"})

Explanation: Sends a message in the chat under the name "Bank" to everyone, indicating the username and cash amount of the player executing this code.

SendChatMessageLocal(sender, message)

Description: Sends a chat message only to the player executing the script, using a specified sender name and message. This command also supports rich text formatting.

Usage: SendChatMessageLocal({"sender": "Bank", "message": "You currently have " + cash + "GB$"})

Explanation: Sends a message in the chat under the name "Bank" to the player executing the script, informing them of their current cash amount.

SendErrorMessage(message)

Description: Sends a local error message, visible only to the player executing the script.

Usage: SendErrorMessage({"message": "This is an error message"})

Currency Management

GiveCurrency(amount)

Description: Awards the player a specified amount of GB$.

Usage: GiveCurrency({"amount": 500})

RemoveCurrency(amount)

Description: Deducts a specified amount of GB$ from the player's balance.

Usage: RemoveCurrency({"amount": 500})

DropCurrency(amount)

Description: Causes the player to drop a specified amount of GB$ in-game.

Usage: DropCurrency({"amount": 500})

Player Interaction

SetCanDropWeapons(state)

Description: Determines whether the player can drop weapons.

Usage: SetCanDropWeapons({"state": false})

PlayerHeal

Description: Fully restores the player's health.

Usage: PlayerHeal

PlayerKill

Description: Instantly kills the player.

Usage: PlayerKill

PlayerEmote(emote)

Description: Triggers a specified emote for the player.

Usage: PlayerEmote({"emote": "twerk"})

PlayerEndEmote

Description: Stops any ongoing emote performed by the player.

Usage: PlayerEndEmote

PlayerCripple

Description: Forces the player into a crouch position until death.

Usage: PlayerCripple

PlayerProtect(duration)

Description: Activates spawn protection for the player for a set duration (default is 4 seconds).

Usage: PlayerProtect({"duration": 4})

PlayerSetTeam(team)

Description: Assigns the player to a specified team, influencing AI interactions.

Usage: PlayerSetTeam({"team": "Maniac"})

Explanation: Sets the player's team to "Maniac", causing neutral NPCs to be aggressive and hostile NPCs to ally with the player.

Available Teams:

Civilian

Maniac

Mutant

PlayerRagdoll

Description: Triggers ragdoll physics for the player.

Usage: PlayerRagdoll

PlayerGetUp

Description: Allows the player to exit ragdoll mode, if possible.

Usage: PlayerGetUp

PlayerTeleport(x, y, z)

Description: Teleports the player to specified coordinates.

Usage: PlayerTeleport({"x": 0, "y": 0, "z": 0})

PlayerInfect(state)

Description: Infects or cures the player based on the provided state.

Usage: PlayerInfect({"state": true})

Inventory Management

ClearInventory

Description: Removes all items from the player's inventory.

Usage: ClearInventory

GiveItem(name)

Description: Grants the player a specified item.

Usage: GiveItem({"name": "P90"})

RemoveItem(name)

Description: Removes a specified item from the player's inventory.

Usage: RemoveItem({"name": "P90"})

✱ Logic Functions

StringContains(string, target)

Description: Returns true if the given string contains the specified target substring. Useful for keyword checks.

Usage: StringContains({"string": "your message or string", "target": "keyword to check for"})

Example:

```
if StringContains({"string": "hello, may I have some money?", "target": "money"}) then
    SendChatMessage({"sender": "Bank", "message": "Sure!"})
    GiveCurrency({"amount": 500})
end if
```

Explanation: In the example, the code checks if the message contains the word "money". If it does, it sends a message from "Bank" and gives the player 500GB$.

RandomInt(min, max)

Description: Generates a random integer between min and max. The upper limit is exclusive.

Usage: RandomInt({"min": 0, "max": 10})

Example:

```
x = RandomInt({"min": 0, "max": 10})
print x
if x == 9 then
    print "Lucky number!"
end if
```

RandomFloat(min, max)

Description: Generates a random float between min and max. The upper limit is exclusive.

Usage: RandomFloat({"min": 0, "max": 10})

Example:

```
x = RandomFloat({"min": 0, "max": 10})
print x
if x >= 9.9 then
    print "That had a 1% chance of happening!"
```

end if

HasRegistered(name)

Description: Checks if an object with the given name is registered.

Usage: HasRegistered({"name": "myObject"})

HasComponent(componentName)

Description: Verifies if this object has a specific component, returning true if it does.

Usage: HasComponent({"name": "AudioSource"})

🔊 Audio Management

AddAudioSource

Description: Attaches an AudioSource component to the object.

Usage: AddAudioSource

SetAudioLoop(state)

Description: Configures whether the audio should loop.

Usage: SetAudioLoop({"state": true})

SetAudioVolume(value)

Description: Adjusts the volume of the AudioSource.

Usage: SetAudioVolume({"value": 0.5})

SetAudioPitch(value)

Description: Sets the pitch of the AudioSource.

Usage: SetAudioPitch({"value": 1})

SetAudioSpatial(value)

Description: Determines the spatial blend of the AudioSource (0 for 2D Audio, 1 for Full 3D Audio).

Usage: SetAudioSpatial({"value": 1})

SetAudioMinDistance(value)

Description: Sets the minimum audible distance for the AudioSource.

Usage: SetAudioMinDistance({"value": 0})

SetAudioMaxDistance(value)

Description: Defines the maximum audible distance for the AudioSource.

Usage: SetAudioMaxDistance({"value": 30})

SetAudioClip(name)

Description: Assigns a loaded audio clip to the AudioSource.

Usage: SetAudioClip({"name": "theme.mp3"})

PlayAudioSource

Description: Initiates playback of the AudioSource.

Usage: PlayAudioSource

StopAudioSource

Description: Halts playback of the AudioSource.

Usage: StopAudioSource

RemoveAudioSource

Description: Detaches the AudioSource component from the object.

Usage: RemoveAudioSource

🕹 AI Integration

AddAI

Description: Adds an AI component to the object.

Usage: AddAI

AISetSpeed(value)

Description: Configures the movement speed of the AI.

Usage: AISetSpeed({"value": 5})

AIFollowTarget(target)

Description: Directs the AI to follow a specified registered object.

Usage: AIFollowTarget({"target": "player"})

RemoveAI

Description: Removes the AI component from the object.

Usage: RemoveAI

Transform Data

Overall Description

You can directly access or modify the following transform values of objects:

Data Access and Modification

You can directly access or modify the following data values, unless read only:

Usage Example:

cash = 300

x = cash // 'cash' can be substituted with any other value listed

Usage Example:

posX = 3

x = posX // *'posX' can be substituted with any other value listed below*

Position

posX: X position of this object.

posY: Y position of this object.

posZ: Z position of this object.

Local Position (Relative to Parent)

localPosX: Local X position, influenced by parent's position.

localPosY: Local Y position, influenced by parent's position.

localPosZ: Local Z position, influenced by parent's position.

Rotation

rotX: X rotation of this object.

rotY: Y rotation of this object.

rotZ: Z rotation of this object.

Local Rotation (Relative to Parent)

localRotX: Local X rotation, influenced by parent's rotation.

localRotY: Local Y rotation, influenced by parent's rotation.

localRotZ: Local Z rotation, influenced by parent's rotation.

Scale (Relative to Parent)

scaleX: X scale of this object.

scaleY: Y scale of this object.

scaleZ: Z scale of this object.

Child Count

childCount: Number of children this object has. (read only)

Game Data

- cash: The game currency (GB$).
- streak: The current killstreak.
- infected: Infected state of the player (true/false).
- isHost: Indicates if the player is the host (true/false).
- dead: Indicates if the player is dead.
- equippedWeapon: Currently equipped weapon (changing this value has no effect).
- name: Name of this object.
- username: Username of the player.
- System Data
- version: Current game version (read-only).
- path: Path to the GoreBox AppData folder (read-only).
- platform: Platform the mod is running on (read-only).
- targetFrameRate: Target frame rate for the game.

- gamemode: Current game mode (read-only).

🖫 Saving / Loading

Key and Value Explained

To save and load data, use a key to identify the value. For instance, save a score with the key "score", and then load it later using the same key.

SaveFloat(key, value)

Description: Securely saves a float value.

Usage: SaveFloat({"key": "hunger", "value": 34.2})

SaveInt(key, value)

Description: Securely saves an int value.

Usage: SaveInt({"key": "score", "value": 100})

SaveString(key, value)

Description: Securely saves a string.

Usage: SaveString({"key": "rank", "value": "chief"})

GetFloat(key, defaultValue)

Description: Loads a saved float value.

Usage: GetFloat({"key": "hunger", "defaultValue": 100})

GetInt(key, defaultValue)

Description: Loads a saved int value.

Usage: GetInt({"key": "score", "defaultValue": 0})

GetString(key, defaultValue)

Description: Loads a saved string.

Usage: GetString({"key": "rank", "defaultValue": "jobless"})

Advanced Functions

AddGamemode(modeName, isHidden, minPlayers, maxPlayers, singleplayer, multiplayer, hostSwitchFeed, playerConnectionFeed, killFeed, hostBadge, hostCommands, hostPerms, canDropWeapons, creator, noclip, invincibility, slowMo, infstamina, infammo, ignoredAI, actionCam, banDolls, banAiDolls, banExplosives, banBigExplosives, banRC, banVehicles, banNextbots, banEntities, banClothes, banMeds, banProps, banFood, banExplosiveWeapons, banHeavyWeapons, banLightWeapons, banMeleeWeapons)

Description: Adds a new gamemode with specified settings and permissions.

Usage: AddGamemode({"modeName": "GunGame", "isHidden": false, "minPlayers": 2, "maxPlayers": 8, "singleplayer": false, "multiplayer": true, "hostSwitchFeed": false, "playerConnectionFeed": true, "killFeed": true, "hostBadge": false, "hostCommands": false, "hostPerms": false, "canDropWeapons": false, "creator": false, "noclip": false, "invincibility": false, "slowMo": false, "infstamina": false, "infammo": true, "ignoredAI": false, "actionCam": false, "banDolls": true, "banAiDolls": true, "banExplosives": true, "banBigExplosives": true, "banRC": true, "banVehicles": true, "banNextbots": true, "banEntities": true, "banClothes": true, "banMeds": true, "banProps": true, "banFood": true, "banExplosiveWeapons": true, "banHeavyWeapons": true, "banLightWeapons": true, "banMeleeWeapons": true})

RegisterGameObject(name, registeredName)

Description: Finds an object with the target name, renames it to registeredName, and registers it.

Usage: RegisterGameObject({"name": "GoreDoll0291", "registeredName": "registeredGoreDoll"})

AddComponent(componentName)

Description: Adds the requested component to this object if it exists.

Usage: AddComponent({"componentName": "BoxCollider"})

RemoveComponent(componentName)

Description: Removes the requested component from this object if it exists.

Usage: RemoveComponent({"componentName": "BoxCollider"})

SetComponentValue(componentName, propertyName, value)

Description: Sets the value of a specified property in a component on this object.

Usage: SetComponentValue({"componentName": "RigidBody", "propertyName": "mass", "value": 50})

Explanation: Sets the object's RigidBody mass to 50.

GetComponentStringValue(componentName, propertyName)

Description: Retrieves the string value of a specified property in a component on this object.

Usage: GetComponentStringValue({"componentName": "TMP_Text", "propertyName": "text"})

GetComponentFloatValue(componentName, propertyName)

Description: Retrieves the float value of a specified property in a component on this object.

Usage: GetComponentFloatValue({"componentName": "PlayerMaster", "propertyName": "timePercent"})

GetComponentIntValue(componentName, propertyName)

Description: Retrieves the int value of a specified property in a component on this object.

Usage: GetComponentIntValue({"componentName": "GBWeapon", "propertyName": "MagSize"})