Lua Scripting Soft Kollision für Statics setzen

Für manche Items ist die harte Kollision sinnvoll, wie für Türen, oder Pushables.

Wenn wir aber bedenken dass Lara auch vor Bäumen oder Pflanzen wie vor eine Wand rennt, sieht das schon wieder anders aus.

In TEN haben die Items alle standardmäßig die "Harte" Kollision.

Nun ändern wir das für Objekte ab, die eine "softe" Kollision haben sollen.

Ab TEN 1.0.3 können wir mit GetStaticsBySlot() alle Objekte eines Slots auswählen und benötigen daher nicht mehr alle Objekte einzeln.

Dazu kopiert ihr diese Funktion in euer Level Lua Script:

Code

```
function SetStaticsSoftCollision()
    local staticSlots = {
        0,2,4,5,6,7,11,12
    }

    for index, staticSlot in pairs(staticSlots) do
        local statics = GetStaticsBySlot(staticSlot)
        for index, static in pairs(statics) do
            static:SetSolid(false)
        end
    end
end
```

Alles anzeigen

Erklärung:

Ihr sucht also erstmal alle Slot ID's der Statics raus, die eine Soft Collision haben sollen und speichert dies in einem Table (in diesem Fall "staticSlots") ab.

Code

```
local staticSlots = {
    0,2,4,5,6,7,11,12
}
```

Dann werden alle Static ID Slots in einem Schleifendurchgang iteriert und während jeden Schleifendurchgangs werden alle Static des aktuell in der Schleife befindlichen Slots in einer neuen Variable "Statics" gespeichert ->

Code

```
local statics = GetStaticsBySlot(staticSlot)
```

Ein weiterer Schleifendurchgang setzt dann für alle Statics des aktuellen Slots die Soft Collision:

Code

```
for index, static in pairs(statics) do
    static:SetSolid(false)
end
```

Dann muss die Funktion noch getriggert werden. Ihr könnt das dann entweder über einen Volume Trigger machen, oder (in dem Fall vielleicht die sinnvollere Alternative)

dies über einen Level Event triggern - in dem Fall ergibt es Sinn dies hier zu triggern:

Code

```
LevelFuncs.OnStart = function()
    SetStaticsSoftCollision()
end
```

Dies führt dazu, dass eure Funktion

Code

```
SetStaticsSoftCollision()
```

dann einmal zu Anfang nach dem Start des Level ausgeführt wird.

Wollt ihr die Funktion aber trotzdem lieber via Volume Trigger ausführen, muss die Funktion als "LevelFuncs" definiert werden:

Code

```
LevelFuncs.SetStaticsSoftCollision = function(triggerer)
    ...
end
```

Das wars.

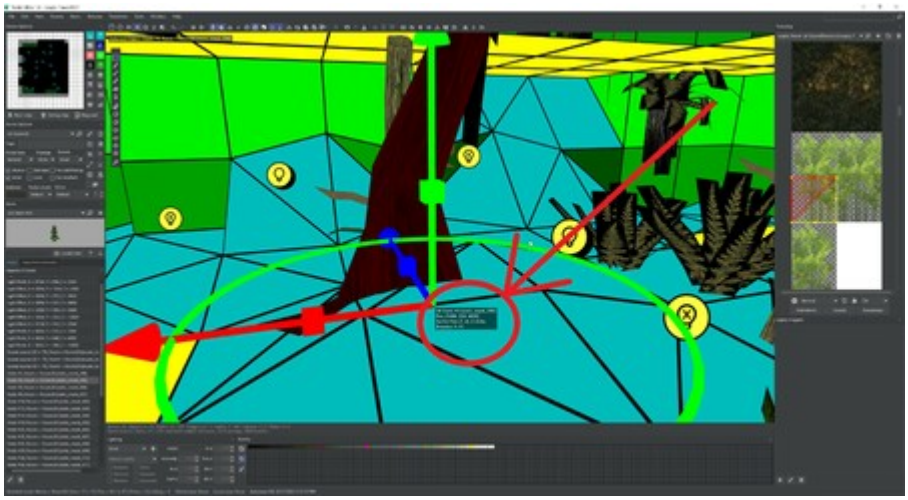
Alle Slots (in meinem Fall sind das 0,2,4,5,6,7,11,12) haben dann nach dem Level Start eine "Soft Collision".

Vor der Version 1.0.3 mussten wir alle Namen der Objekte manuell heraussuchen:

Dazu benötigen wir den "Namen" der Objekte. Finden könnt ihr das im TE.

Dazu markiert ihr das betreffende Objekt und in der ersten Zeile im schwarzen Kasten steht der Name in eckigen Klammern [].

In meinem Fall ist das "static_mesh_399":



Ihr könnt den Namen auch anpassen, damit das Script später lesbarer ist. Zum Beispiel "Baum 1" oder sowas in der Art.

Das geht mit Rechtsklick auf das Objekt und "Rename Object".

Nun kopiert ihr den Namen.

Solltet ihr mehrere ändern wollen, so kopiert ihr auch die anderen Namen. Ich habe noch 2 weitere: "static_mesh_433" und "static_mesh_398".

Im Level Lua Script setzt ihr bei der "OnStart" Function eine neue Funktion - wir nennen diese "SetStaticsSoftCollision" :

Code

```
LevelFuncs.OnStart                              =                              function()
    SetStaticsSoftCollision()
end
```

Damit wird SetStaticsSoftCollision nach dem Start des Levels aufgerufen.

Die Neue Funktion könnt ihr dann nach dem Ende eures Scriptes setzen.

Bei einem leeren Script ist das normalerweise diese Zeile:

Code

```
LevelFuncs.OnEnd= function() end
```

Die Funktion "SetStaticsSoftCollision" bekommt folgenden Inhalt:

Code

```
function                                                SetStaticsSoftCollision()

                                                                    local                   Soft
        "static_mesh_433",
        "static_mesh_399",
        "static_mesh_398"
    }

                          for      index,      value      in      ipairs(SoftCollisionStatics

        static:SetSolid(false)
    end
end
```

Alles anzeigen

Was macht die Funktion?

Code

```
local                   SoftCollisionStatics                    =                   {
    "static_mesh_433",
    "static_mesh_399",
    "static_mesh_398"
}
```

Erstellt einen lokalen "Table" (eine Gruppe von Werten) mit dem Namen "SoftCollisionStatics".

Die Werte werden kommasepariert hinzugefügt.

Code

```
for       index,       value       in       ipairs(SoftCollisionStatics)       do
                                                    local               static
    static:SetSolid(false)
end
```

Hier wird mit einer "Schleife" bzw. Loop gearbeitet. Es ist eine spezielle Schleife, denn diese wird NUR SO OFT ausgeführt, wie es Werte in dem Table "SoftCollisionStatics" gibt.

Ich hätte auch eine "normale" for oder while Schleife nehmen können, aber da hätte ich eine Anzahl an Schleifen Durchgängen angeben müssen. Somit müsste ich bei Anpassungen (hinzufügen oder entfernen von Objekten im Table "SoftCollisionStatics" auch immer die Schleife abändern. Als Tipp: Wenn ein Table durchgegangen werden soll (wie in diesem Fall) - nutzt immer die spezielle For Schleife.

Tables haben immer einen Index und ein Value. Sofern man kein "assoziatives" Table erstellt, sondern wie in unserem Fall ein einfaches durchnummeriertes Table Object, ist der Index immer fortlaufend eine Ziffer und das Value das dazugehörige "Element" (den Namen den wir vergeben haben, also)

Im index wird der "key" des Items des Table des aktuellen Durchgangs gespeichert (Index 1 - 3) - und im value eben das "Value" - in unserem Fall "static_mesh_433", "static_mesh_399", "static_mesh_398"

Innerhalb der Schleife wird nun über GetStaticByName(value) ein Static Object mit dem Namen geholt (welche in "value") steht und in der lokalen Variable "static" gespeichert.

Mit static:SetSolid(false) wird die "soft" Kollision gesetzt.

Und das wars. Alle Objekte die im Table definiert werden, bekommen nun eine "soft" Kollision.

Eventuell gibt es ja irgendwann mal die Möglichkeit Statics oder Moveable über den Slot (anstelle über den Namen) zu holen.

Dann bräuchte man sich keinen Table erstellen. Aber solange es das nicht gibt, muss man es so machen.

Hier noch das gesamte Script:

Code

```
----                            FILE:                            \Jungle_Feaver.lua

LevelFuncs.OnLoad                  =                  function()                  end
LevelFuncs.OnSave                  =                  function()                  end
LevelFuncs.OnStart                            =                            function()
    SetStaticsSoftCollision()
end
LevelFuncs.OnControlPhase               =              function()              end
LevelFuncs.OnEnd                  =                  function()                  end

function                                                SetStaticsSoftCollision()

                                                                    local                 Soft
        "static_mesh_433",
        "static_mesh_399",
        "static_mesh_398"
    }

                           for      index,      value      in      ipairs(SoftCollisionStatics

        static:SetSolid(false)
    end
end
```

Alles anzeigen