Теория этого дела

В игре существует такой объект как "zone\_teleport", но если мы его создадим через create, то он будет выглядеть и переливаться как настоящий телепорт, но телепортировать нас куда-либо увы не сможет. Связано это с тем что, у аномалий ( а телепорт это такая разновидность аномалии ) параметры задаются хитромудро, через all.spawn. Вобщем штатным способом телепорты без полнофункционального редактора карт не получить (хотя возможно я не прав уже после написания этого текста появились идеи как это сделать через all.spawn). Значит на нашу долю остаються способы "не штатные" ;) Реализуем самый простой. Будем считать что у нас на карте есть квадрат с заданными координатами при попадании в который актера должно переместить в точку с другими координатами. Для этого будем периодически проверять координаты актера, и если он в квадрате - перемещяем. Вот в краце принцип действия нашего "самодельного" телепорта.

Реализация

В каталоге gamedata\scripts\ Создадим файл bind\_mteleport.script с логикой работы нашего телепорта.

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- \*\* Imp \*\*

-- \*\* Биндер самодельных телепортов \*\*

-- \*\* Поддерживает работу самопальных телепортов \*\*

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

local teleport\_binders ={} -- Список телепортов

function abs\_comp(a,b)

-- Служебная функция вычисления разности

if( a < b) then

return (b - a)

else

return (a - b)

end

end

function teleportate(x,y,z)

-- Функция телепортации

local a = vector()

-- Задаем координаты

a.x = x

a.y = y

a.z = z

-- Сама телепортация

db.actor:set\_actor\_position(a)

-- Звуковое сопровождение

local snd\_obj = xr\_sound.get\_safe\_sound\_object([[affects\tinnitus3a]])

snd\_obj:play\_no\_feedback(db.actor, sound\_object.s2d, 0, vector(), 1.0)

end

function actor\_update(delta)

local i,v,acter\_poz,s

-- Получим позицию актера (что-бы каждый раз не запрашивать)

acter\_poz = db.actor:position()

-- Проверяем наши телепорты

for i, v in pairs(teleport\_binders) do

s = v.parametrs

local obj = level.object\_by\_id( i )

if obj ~= nil then

-- Наш телепорт в онлайне проверяем дальше

if s.teleporte ~= nil and s.teleporte ~= false then

-- Телепорт запущен

if ( time\_global() <= s.time ) then

-- Если время отведенное на показ спецэфектов

-- прошло, производим телепортацию

teleportate(s.poz\_x,s.poz\_y,s.poz\_z)

if s.rotate ~= nil then

db.actor:set\_actor\_direction(s.rotate)

end

s.teleporte = false

end

return

end

-- Пороверим не забрел-ли актер в наш телепорт

if (abs\_comp(s.x, acter\_poz.x)< v.parametrs.radius and

abs\_comp(s.z, acter\_poz.z)< v.parametrs.radius and

abs\_comp(s.y, acter\_poz.y)< v.parametrs.z\_radius) then

-- Актер в зоне действия телепорта, запустим телепорт

s["teleporte"] = true

s["time"] = time\_global() + 500

-- Запускаем спецэфекты телепортации

level.add\_pp\_effector ("teleport.ppe", 2006, false)

end

end

end

end

function bind( obj )

obj:bind\_object( restrictor\_teleport( obj ) )

end

----------------------------------------------------------------------------------------------------

class "restrictor\_teleport" ( object\_binder )

function restrictor\_teleport:\_\_init(obj, char\_ini) super(obj)

end

function restrictor\_teleport:net\_spawn(data)

local char\_ini = system\_ini()

-- Если это телепорт то занесем его в специальный список телепортов

if self.teleport == true then

teleport\_binders[self.object:id()] = self

-- Заполним таблицу параметров

self["parametrs"] = {}

if char\_ini:line\_exist(self.section, "radius") then

self.parametrs["radius"] = tonumber(char\_ini:r\_string(self.section, "radius"))

else

self.parametrs["radius"] = 2 -- Дефолтный радиус по xy

end

if char\_ini:line\_exist(self.section, "z\_radius") then

self.parametrs["z\_radius"] = tonumber(char\_ini:r\_string(self.section, "z\_radius"))

else

self.parametrs["z\_radius"] = self.parametrs["radius"] -- если радиус высоты не задан то задаем равным радиусу xy

end

-- Запомним позицию что-бы каждый раз не считать

local s\_obj = alife():object(self.object:id())

self.parametrs["x"] = tonumber(s\_obj.position.x);

self.parametrs["y"] = tonumber(s\_obj.position.y);

self.parametrs["z"] = tonumber(s\_obj.position.z);

-- Запомним координаты куда телепортимся

self.parametrs["poz\_x"] = tonumber(char\_ini:r\_string(self.section, "poz\_x"))

self.parametrs["poz\_y"] = tonumber(char\_ini:r\_string(self.section, "poz\_y"))

self.parametrs["poz\_z"] = tonumber(char\_ini:r\_string(self.section, "poz\_z"))

if char\_ini:line\_exist(self.section, "rotate") then

self.parametrs["rotate"] = tonumber(char\_ini:r\_string(self.section, "rotate"))

end

end

return true

end

function restrictor\_teleport:net\_destroy()

-- Удаляем наш телепорт

teleport\_binders[self.object:id()] = nil

self.parametrs = nil

object\_binder.net\_destroy(self)

end

function restrictor\_teleport:reload(section)

local char\_ini = system\_ini()

self.section = section

-- Если это телепорт то

if char\_ini ~= nil and char\_ini:line\_exist(self.section, "teleport") then

self["teleport"] = true

end

end

Для постоянного обновления нужно прицепить функцию actor\_update() к биндеру актера, для чего в файле bind\_stalker.script найдем функцию:

function actor\_binder:update(delta)

В ней найдем вызов обновления рестрикторов bind\_restrictor.actor\_update(delta) под которым вставим строку с вызовом нашей функции обновления: bind\_mteleport.actor\_update(delta)

Все с программной частью закончили, теперь задаем данные телепорта.

В каталоге gamedata\config\misc открываем файл zone\_teleport.ltx и в конце файла добавляем следующие строки описывающие конкретный телепорт:

[m\_teleport\_1]:zone\_teleport

teleport = standart

script\_binding = bind\_mteleport.bind

;Параметры нашего телепорта

radius = 2

;Высота захвата телепорта

z\_radius = 2

;Куда телепортируемся (телепортация всегда идет в пределах карты)

poz\_x = 22.78

poz\_y = 20.35

poz\_z = 659.24

; Угол зрения при появлении. Если параметра нет то не меняется.

rotate = 1.5

Параметры нашего телепорта:

radius - на самом деле не радиус, а половина длинны стороны нашего квадрата (в начале я хотел сделать его кругом, но посчитал, что лучше не тратить процессорное врямя по пусту). Центром квадрата является точка респавна телепорта.

z\_radius - высота нашего телепорта.

poz\_x, poz\_y, poz\_z - координаты точки телепортации.

rotate - Угол поворота после телепортации от оси X (я не разбирался в каких единицах задается, но 1.5 примерно равно 90 градусов). Если параметр удалить то будет оставатья угол под которым актер вошел в телепорт.

Использование

Теперь с помощью create создадим наш телепорт: Пример:

local obj

local a = vector()

a.x = -244.55

a.y = -19.46

a.z = -125.42

obj = alife():create("m\_teleport\_1",a,12829,8,65535)

Создаст телепорт возле выхода из бункера Сидоровича. Наш телепорт перебрасывает игрока на вышку блокпоста (перед выходом с уровня).

Все! Вот тут вы можете взять готовый мод с двумя телепортами