Forum: Moteur de jeu GameBlender et alternatives
Topic: Portal BGE - Version 1.8 - 8 niveaux fonctionnels
Subject: Re: Portal BGE

PostÃ© par: gui36fr
Contribution le : 23/10/2010 13:33:01

J'ai plus ou moins fini le morceau de script et je l'ai testé : ça marche sans erreurs !

J'ai par contre laisser la force du joueur en sortie sur un plan, car le joueur passais à travers le sol quand la force était vers le bas. Il ne tombe donc plus lors du passage.

Voilà le script :
import des fonctions utiles from cmath import phase from math import cos, sin x, y, z = 0, 1, 2 # i est l'imaginaire pur unitaire qui sera utilise pour les rotations complexes i = complex(0, 1) # nor_1 est le vecteur normal du portail d'entree # nor_2 est le vecteur normal du portail de sortie # rot est la matrice d'orientation de obj (avant teleportation) # force est la force applique a obj (avant teleportation) nor_1 = rotOwn[z] nor_2 = rotPortail[z] rot = obj.orientation force = obj.worldLinearVelocity # si les portails ne sont pas a l'horizontale, on effectu la reorientation : if -0.9<nor_1[z]<0.9 and -0.9<nor_2[z]<0.9: # definition des modules des projetes sur un plan horizontal mod_1 = (nor_1[x]**2 + nor_1[y]**2)**0.5 mod_2 = (nor_2[x]**2 + nor_2[y]**2)**0.5 mod_force = (force[x]**2 + force[y]**2)**0.5 ######## REORIENTATION DE OBJ ######## # calcul de phi, l'angle entre l'axe y de obj par rapport au portail d'entree phi = phase(complex(-rot[y][x], -rot[y][y])) - phase(complex(nor_1[x]/mod_1, nor_1[y]/mod_1)) # calcul de angle, l'angle entre l'axe y futur de obj par rapport au portail de sortie angle = phase(complex(nor_2[x]/mod_2, nor_2[y]/mod_2)) - phi # calcul des axes d'orientations x et y de objet (forme complexe) axey = complex(cos(angle), sin(angle)) axex = -i*axey # reorientation de obj obj.orientation = [[axex.real, axex.imag, 0.], [axey.real, axey.imag, 0.], [0., 0., 1.]] ######## CALCUL DE LA FORCE APPLIQUEE A OBJ ######## # calcul de phi, l'angle du projete de la force appliquee sur obj par rapport au portail d'entree phi = phase(complex(-force[x]/mod_force, -force[y]/mod_force)) - phase(complex(nor_1[x]/mod_1, nor_1[y]/mod_1)) # calcul de angle, l'angle de la force future appliquee sur obj par rapport au portail de sortie angle = phase(complex(nor_2[x]/mod_2, nor_2[y]/mod_2)) - phi # application de la force sur obj obj.worldLinearVelocity = [cos(angle)*mod_force, sin(angle)*mod_force, 0.]

Bon blend