

Gotowe klasy - [UT'99] RJumpPad - do zastapienia kickera by [Raven](#) dnia: 19 Kwiecien 2008, 23:17

Opis

Ten actors jest inspirowany JumpPadem z UT2004. Po zdefiniowaniu **JumpTarget** kierunek i sila wybicia zostana obliczone automatycznie. Konieczne bedzie takze zmiana wartosci **JumpZModifier** w celu zwiekszenia wysokosci wybicia. Skompilowany skrypt mozna sciagnac [stad](#).

Zmienne

Widoczne

bool bDebugJump: pozwala na zmiane wartosci JumpZModifier w trakcie gry (tylko tryb offline)

Actor JumpTarget: cel wybicia (np. Actor'MyLevel.LiftExit0')

float JumpZModifier: modyfikator wybicia na osi Z. Czasami konieczne jest wpisanie tu wartosci aby wybicie bylo odpowiednie

sound JumpSound: dzwiek wybicia

name JumpClasses: klasa mogaca korzystac z JumpPada (wliczna sa w to klasy potomne

Ukryte

vector RealJumoVelocity: prawdziwe, obliczone, wybicie.


```
//~~~~~
// Copyright 2005-2008 Dead Cow Studios. All Rights Reserved.
// ~~~~~
// Coder: Raven
// ~~~~~
// Changed Kicker.
class RJumpPad extends Triggers;

var(Advanced) bool bDebugJump;
var() Actor    JumpTarget;
var() float    JumpZModifier;
var() sound    JumpSound;
var() name     JumpClasses;
var() bool     bRandomize;
var vector RealJumoVelocity;

replication
{
    reliable if( Role == ROLE_Authority )
        CalculateJumpVelocity;
}

simulated function PostBeginPlay()
{
    if(JumpTarget == none) Destroy();
    RealJumoVelocity = CalculateJumpVelocity();
}

simulated function Touch( actor Other )
{
    local Actor A;

    if (!Other.IsA(JumpClasses)) return;
    PendingTouch = Other.PendingTouch;
    Other.PendingTouch = self;
    if( Event != " )
        foreach AllActors( class 'Actor', A, Event )
            A.Trigger( Other, Other.Instigator );
}

function vector CalculateJumpVelocity()
{
    local vector XYDir, JumpVelocity;
    local float ZDiff, Time, GravityZ;

    GravityZ = Region.Zone.ZoneGravity.Z;

    XYDir = JumpTarget.Location - Location;
```

```

ZDiff = XYDir.Z;
Time = 2.5f + JumpZModifier * Sqrt(Abs(ZDiff/GravityZ));
JumpVelocity = XYDir/Time;
JumpVelocity = XYDir;
JumpVelocity.Z = ZDiff + JumpZModifier;

return JumpVelocity;
}

simulated function PostTouch( actor Other )
{
    local bool bWasFalling;
    local vector Push;
    local float PMag;

    if(bDebugJump)
    {
        RealJumoVelocity = CalculateJumpVelocity();
        BroadcastMessage("X="@RealJumoVelocity.X@" Y="@RealJumoVelocity.Y@" Z="
@RealJumoVelocity.Z);
    }

    bWasFalling = ( Other.Physics == PHYS_Falling );
    if ( bRandomize )
    {
        PMag = VSize(RealJumoVelocity);
        Push = PMag * Normal(RealJumoVelocity + .5 * PMag * VRand());
    }
    else
        Push = RealJumoVelocity;
    if ( Other.IsA('Bot') )
    {
        if ( bWasFalling )
            Bot(Other).bJumpOffPawn = true;
        Bot(Other).SetFall();
    }
    if(JumpSound != none) PlaySound(JumpSound);
    Other.SetPhysics(PHYS_Falling);
    Other.Velocity = Push;
}

defaultproperties
{
    JumpClasses=Pawn
}

```