Parachute

by ®2-D2, modified by CRxTR with help from Ku2zoff Original site: http://www.planethalflife.com/hlprogramming/tutorial.asp?i=42

Tutorial type: Intermediate - A/F

Okay this tutorial covers how to create a parachute like the one in Firearms, but without an opened parachute model and without a HUD sprite message if it's opened. This tutorial could just be copied & pasted, but I want you to understand the stuff you copy!! Otherwise the tutorial is almost worthless.

Right, let's start: First open up «player.h» (we'll only work in the «server.dll» !!) and define the variables we'll need in the CBasePlayer class definition:

```
//parachute variables
BOOL m_fParachute; //the variable which checks if we have the parachute
BOOL m_fParaOpen; //the variable which checks if the parachute is opened
void ParaGlide( void ); //we need to define this function before we can
call/write it
```

Then open up «player.cpp» and write the ParaGlide function above the dead HEV stuff:

```
// by ®2-D2, modified by CRxTR with Ku2zoff
// fixed: some of the code are from the old HL SDK.
//        just re-appropriated the code for XashXT and
//        with Ku2zoff's advise, made it work.

void CBasePlayer::ParaGlide( void ) //the function itself
{
  if(!m_fParachute) //if we didn't pick up the chute ...
    return; // ...return

       if (m_fParaOpen && !FBitSet(pev->flags, FL_ONGROUND)) { // If you
deployed the chute and you are on the ground
       //ALERT( at_console, "You are gliding\n" );
               m_flFallVelocity = 50;
               pev->velocity.z = pev->velocity.z * 0.70;
       }
       else
   {
       m_fParaOpen = FALSE;
       //ALERT( at_console, "You are on the ground. Gliding disabled\n" );
       return;
   }
}
```

You can replace the m_flFallVelocity value with something lower, within 60 - 40 is fine. You can tamper the velocity code but keep it low still. You don't want to plummet to your death.

And in the CBasePlayer::PreThink function (still in «player.cpp») call the ParaGlide function (right there where the functions ItemPreFrame and WaterMove are called).

Now we have to define a command that calls the ParaGlide function. Do this in «client.cpp» in the ClientCommand function

```
else if ( FStrEq(pcmd, "parachute" ) ) //if the command "parachute" is called
{
  if ( GetClassPtr((CBasePlayer *)pev)->m_fParaOpen ) //again, the pointer to the player. this line checks if the parachute is opened
  {
    GetClassPtr((CBasePlayer *)pev)->m_fParaOpen = FALSE; //if so, the parachute shall close
  }
  else //else (if the parachute is closed)
  {
    GetClassPtr((CBasePlayer *)pev)->m_fParaOpen = TRUE; //if so, the parachute shall open
  }
}
```

Right, this opens the parachute if it's closed, and if it's opened it closes the parachute Okay, the parachute can now open and close but we can't pick it up yet. So, we have to add an «item_parachute» in «items.cpp»

```
class CItemParachute : public CItem //the parachute item class
{
  void Spawn( void ) //the spawn function
  {
    Precache( ); //calls the precache function
    SET_MODEL(ENT(pev), "models/w_longjump.mdl"); //sets the model lying in the world to the longjump model. I think it looks similar to a parachute. You can also use the TFC backpack model for example
    CItem::Spawn( ); //I think this declares the spawn function of the parachute as a public item spawn function
  }
  void Precache( void ) //The precache function we called in spawn
  {
    PRECACHE_MODEL ("models/w_longjump.mdl"); //this just precaches the model, otherwise you would get an error that it's not precached. You can't use unprecached models
  }
  BOOL MyTouch( CBasePlayer *pPlayer ) //MyTouch (if we pick the parachute up...)
  {
    if ( pPlayer->m_fParachute ) //if we have the parachute already...
```

```
    return FALSE; //don't let us pick it up again

    pPlayer->m_fParachute = TRUE; //...tell halflife we have the item
    EMIT_SOUND( pPlayer->edict(), CHAN_ITEM, "items/parachute_equip.wav", 1,
ATTN_NORM ); //parachute sound, if you want.

    MESSAGE_BEGIN( MSG_ONE, gmsgItemPickup, NULL, pPlayer->pev ); //this is
a message with which we tell the client that we have picked up an item
      WRITE_STRING( STRING(pev->classname) ); //it tells the game that it's
a parachute we just picked up
    MESSAGE_END(); //here ends the message we started above

    return TRUE; //return true to avoid errors
  }
};

LINK_ENTITY_TO_CLASS( item_parachute, CItemParachute ); //makes an entity
from the class we just wrote
```

Then precache the item in «weapons.cpp», compile it, start the game, give yourself the item, bind the command «parachute» to a key, jump from a high place in the map and press the key you just bound. If you press it again, the parachute will close and you will fall as fast as always

But if you die you still have the parachute, so let's remove it...in «player.cpp» in the player killed function. Just set m_fParachute to TRUE there.

You still need to add the parachute in the FGD though, if you want to utilize it for mapping with Hammer.

As you probably noticed, I don't tell you always where exactly to add the code. This is just, 'cuz I want you to understand the tutorial and these things I didn't tell you exactly, you normally have to know. If you don't understand with my tutorial or something, feel free to mail me (address at the top)

®2-D2 (Believe in Retrieve :)

Remarks:

- Using !FBitSet(pev→flags, FL_ONGROUND), you can detect if your player is on the ground. There are other flags out there that you can use.
- The code doesn't have a sprite, so audio cues are implemented here. You can though add sprites, with a little effort.