Wire Expression2:Examples

From GMod Wiki

This section is for example expressions. To make this section as useful as possible, please try to keep your examples simple and targeted to specific areas of Expression 2. For instance, a good example would show you how to make a basic GPS using Expression 2's vectors. This isn't a place for you to share your contraptions, you can do that here (http://www.wiremod.com/forum/finished-contraptions/) .

Guidelines:

- When creating a new example make sure to follow the wiki format everyone else is using.
  - Put it in the correct complexity category.
  - Include (by <your name>) at the end of the title.
  - Include a description of what it does.
  - If you have more than one example that fits a difficulty, insert them one after another, it keeps things organized.
  - Use lua tags for your E2 code. The syntax highlighting isn't perfect, but it helps a lot with comments.
- Coding
  - Please make sure the variable names you use describe what they are for, for example, use *WeldLatch* instead of *WL* for your variable name.
  - Follow the code format set in Expression 2: Formatting. Use 4 spaces for each indentation (comments are exempt).
  - Try to line up series of comments so they are easier to separate from the code.
- This is a wiki, so expect your expressions to be modified by other editors.

Contents

Simple

RPM Counter (by Beer)

Description: Very simple expression for determining RPM using angVel(). For this example, Wheel:entity is a PHX wheel linked to an Entity Marker. Depending on the prop you use, yaw may not be what you need. Try pitch and roll if you have trouble.

Code

```
@name RPM
@inputs Wheel:entity
@outputs RPM
@trigger none
interval(200)
RPM = abs(Wheel:angVel():yaw()/360*60)
```

GPS (by Fishface60)

Description: This is a simple expression to get used to the new functionality, it would be suitable practice for a new user. The Interval is required for the GPS to update its values, as there are no inputs to change for it to recalculate. It will update ten times per second, if you need more or less you can change the interval.

Code

```
@name GPS (updated by Syranide)
@outputs X Y Z Pos:vector
runOnTick(1)
Pos=entity():pos()
X=Pos:x()
Y=Pos:y()
Z=Pos:z()
```

Speedometer (by integer)

Description: This will output the speed of the chip in Knots. This is achieved by getting the entity's velocity, then getting its length, which in the case of velocity is the speed. After this it is converted to Knots, if you prefer to use a different unit then change the knots to the desired unit.

Code

```
@name Speedometer
@outputs Knots
interval(1000)
Knots=toUnit("knots", entity():vel():length())
```

Detonator (by Syranide)

Description: After the button is pressed then the timer starts, making the output 1 after 1 second, then it resets itself after 100ms.

Code

```
@name Detonator
@inputs Button
@outputs WtfBbqKaboom
if (~Button & Button) {
    timer("boom", 1000)
}
if (clk("boom")) {
    WtfBbqKaboom = 1
    timer("reset", 100)
}
if (clk("reset")) {
    WtfBbqKaboom = 0
}
```

Chat example (by Matte)

Description: Simple expression showing how to set a value in the E2 via chat commands. It takes what the owner said, explodes it into an array, and checks if the first string equals "/light". If it does, it sets the Light-variable equal to the second variable. For instance, "/light 2" would set Light to 2.

Code

```
@name Chat
@outputs Light
runOnChat(1)
LastSaid=owner():lastSaid():explode(" ")
if (chatClk(owner())&(LastSaid[1,string]:lower() == "/light")) {
    Light = LastSaid[2,string]:toNumber()
```

```
}
```

Beacon Sensor (by Hitman271)

Description: A piece of simple code to show more entity interface. The TargetE input goes to the entity output of a target finder.

Code

```
@name Exp2 Beacon Sensor
@inputs TargetE:entity
@outputs Distance Bearing Elevation
interval(20)

EE = entity() #Expression Entity
TV = TargetE:pos() #Target Vector

Distance = EE:pos():distance(TV)
Bearing = EE:bearing(TV)
Elevation = EE:elevation(TV)
```

String Example (by Hitman271)

Description: Always remember to declare the variable type in the input and output fields when in doubt. Watch this code in the debugger. Tar would goto the entity output on a target finder.

Code

```
@name String Example
@inputs Tar:entity
@outputs PosX PosY PosZ Name Health
@outputs Pos:vector Type:string

interval(20)

Pos    = Tar:pos()
Type   = Tar:type()
Health = Tar:health()

PosX = Pos:x()
PosY = Pos:y()
PosZ = Pos:z()

if (Tar:isNPC()) {
    if     (Type == "npc_citizen")   {Name = 1}
    elseif (Type == "npc_combine_s") {Name = 2}
    elseif (Type == "npc_zombie")    {Name = 3}
    elseif (Type == "antlion")       {Name = 4}
    else                             {Name = 5}
}
```

String Explode (by Hitman271)

Description: Another simple code to show how to separate a string easily. This code prints to chat so no need for debugging

Code

```
@name String Explode! by Hitman271
@persist I

String = "Hello World , Goodbye World"
```

```
SaidFor = String:explode(" ") #This created an array

if (I<= SaidFor:count()) {
    I++
    print( SaidFor[ I, string ] )
    interval(10)
} else {
    I = 0
    interval(10000)
}
```

WireLink Example (by Hitman271)

Description: A wirelink example. The wirelink is located in the wire section of the wire menu, or right by the normal wire stool.

You will need; A target finder, A holo emitter, A holo grid, A constant value of 1, and this code of course. Right-click the emitter, then left-click the grid to connect them. On the grid's input of USEGPS, wire it to the constant value of 1.

With the wirelink stool, left-click the emitter.

Set the target finder to players or npcs.

Then wire the E2 Holo[WIRELINK] input to the emitter. Then the E2 Player[ENTITY] input to the target finder's entity output.

With the wirelink, MUCH less wiring is required as the emitter's input have been untouched to demonstrate.

Code

```
@name Wirelink Example
@inputs Holo:wirelink Player:entity

interval(20)

if (Holo & Player:isPlayer()) {
    Holo["Vector", vector] = Player:shootPos()
    Holo["Active", number] = 1
    Holo["FadeRate",number] = 100
}
```

Vector and Entity wirelink testing (by fishface60)

Description: These are the expressions I used to test whether the then new wirelink entity and vector functions were working, they serve as basic examples of how to use them.

WireLink[S,vector]

```
@name vector test
@inputs GPS:wirelink
@outputs GPSString:string
interval(20)
GPSString = GPS["Vector", vector]:toString()
```

WireLink[S,vector] = V

```
@name setVec test
@inputs Thruster:wirelink
@outputs ThrustVec:vector
interval(20)
ThrustVec = vec(random(), random(), random())
Thruster["Vector",vector] = ThrustVec
```

WireLink[S,entity]

```
@name entity Test
@inputs Marker:wirelink DirectEntity:entity
@outputs ID Type:string Model:string Owner:string Name:string
WLE = Marker["Entity",entity]
DE = DirectEntity
if (WLE) {
    ID=WLE:id()
    Type=WLE:type()
    Model=WLE:model()
    Owner=WLE:owner()
    Name=WLE:name()
}
if (DE) {
    ID=DE:id()
    Type=DE:type()
    Model=DE:model()
    Owner=DE:owner()
    Name=DE:name()
}
```

WireLink[S,entity] = E

```
@name setEntity test
@inputs Chip:wirelink Marker:entity
Chip["DirectEntity",entity] = Marker
```

Basic Player Targetting (by jman31415)

Description: Can target Players with chat by typing !target followed by the player's name.

Code

```
@name PlayerTarget
@inputs
@outputs Target:entity
@persist
runOnTick(1)
runOnChat(1)
LastSaid=owner():lastSaid():explode(" ")
if (chatClk(owner())&LastSaid[1,string]=="!target")
    {Target=findPlayerByName(LastSaid[2,string])}
#the above 2 lines of code basicly say that if the owner
#said something and the first word of that something
#is !target then the chip would take the second word and find the
#player with that name.
```

Thanks I would like to give a huge amout of credit to Matte for his example and credit to Shoffing for introducing me to chat controlled functions.

Follower (by Chinoto)

Description: Follows an entity, orientation does not affect it.

Code

```
@name Follower
@inputs TE:entity
@outputs LX LY LZ
interval(20)
EE=entity() #EE is to make the code shorter
DV=(TE:shootPos()+vec(0,0,50)-EE:pos())*10-EE:vel()
#Finds the world XYZ (vector) difference between the target and E2
#vel() is the equivalent of delta except its fixed at per second rather than difference between executions
LX=DV:dot(EE:right())
LY=DV:dot(EE:forward())
LZ=DV:dot(EE:up())
#EE:right gives a unit vector to the right of the expression
#dot() takes two vectors and multiplies their components then adds the answers together or Dot=X1*X2+Y1*Y2+Z1*Z2
```

Door (by Chinoto)

Description: This expression can accept 8 inputs for opening/closing a door ("If any variable is 1" then door opens/closes) and will automatically close after 10 seconds. If wanted you can also add a weld latch so then the door can't be opened through brute force, just wire length to the hydraulic controller and the weld latch to the expression.

Code

```
@name Door
@inputs B1 B2 B3 B4 B5 B6 B7 B8 Length
@outputs Hydr WeldLatch
@persist Trigger Active Timer
@persist Interval Rate Delay Open Closed
if (first()) {
    Interval=100
    Rate    =200*(Interval/1000)
    Delay   =10
    Open    =100
    Closed  =1
}
interval(Interval)
Timer=(Active ? Timer+0.1 : 0)      #If Active is 1 then increment by 0.1 else reset to 0
Trigger=B1|B2|B3|B4|B5|B6|B7|B8     #If any variable is 1 then return 1 else 0
if (($Trigger&Trigger) | (Timer>Delay)) {Active=!Active}
#If (Trigger is 1 and has changed) or (Timer is greater than delay) then {make Active equal NOT Active} #Toggles between 1 and 0
Target=(Active ? Open : Closed)     #If Active is 1 then return Open else Closed
Hydr+=clamp(Target-Hydr,-Rate,Rate) #Subtract Hydr from Target, then clamp it between -Rate and Rate, then add it to Hydr (It's a smoother)
WeldLatch=abs(Length-Hydr)<5        #If the difference between Length and Hydr is less than 5, then enable the weld latch
```

Chinoto Vokro 06:03, 2 April 2009 (GMT-6)

Table-RAM (by Magos Mechanicus)

Description: An example of the use of tables, this expression saves and outputs number data based on input in much the same way a RAM chip does. You control what data to save or retrieve with AddrRead and AddrWrite, and cause it to save the input Data in the address AddrWrite with the (edge-triggered) Clk input.
Note that unlike an ordinary RAM chip, you are not restricted to any particular number range for addressing, though the table will take up a lot of memory with many values saved.

(Syranide: I'd just like to point out that this method is fine, but it should not be used for as a proper memory because every single number will consume a large amount of space, more than 10 bytes each. So do not use it if you will be saving alot of data.)

Code

```
@name RAM table example
@inputs AddrRead AddrWrite Data Clk
@outputs Out
@persist Table:table
if (Clk & ~Clk) {Table[toString(AddrWrite),number] = Data}
Out=Table[toString(AddrRead),number]
```

Array-RAM (By GUN)

Description: Array are new to e2 and are made to be used as internal ram, its basically EXATLY the same as table ram, but its much more easy to code and use

Code

```
@name Ram array example
@inputs AddrRead AddrWrite Data Clk Reset
@outputs Out
@persist Ram:array
if (~Clk & Clk)     {Ram[AddrWrite,number] = Data}
if (~Reset & Reset) {Ram=array()}
Out=Ram[AddrRead,number]
```

See, its basically the same as table ram. Except that you don't have to transfer it to a string to write it to the ram. But on the other hand, you can also use arrays to store vectors or strings as well. But be aware, no matter which you use, you're limited to 1048576 cells or 1 MB(THAT'S A LOT!!). You will most likely NEVER use anything close to that. And with arrays you can also Reset the entire "ram" or array with the Ram=array():clone()function, so that's a new thing :P have fun :) -GUN

Gyroscope (by GabbaGubbel)

Description: Its just to take the Expression2 like a Gyroscope.

Code

```
@name Gyroscope by GabbaGubbel (updated by Syranide)
@outputs Pitch Yaw Roll
runOnTick(1)
Ang=entity():angles()
Pitch=Ang:pitch()
Yaw=Ang:yaw()
Roll=Ang:roll()
```

--GabbaGubbel 06:11, 11 January 2009 (GMT-6)

Writing/reading global variables (By ZeikJT/Divran)

New syntax (By Divran)

Description: These two E2s communicate using the 'new' system.

Code

```
#Expression nr 1
@inputs Button1 Button2
@persist G:gtable
if (first()) {G = gTable("mygroup",0)} #Get the gtable from a group, and make it non-shared (non-shared means only YOUR E2s can read & write on this ta
```

```
if (Button1) {G["button",number] = 1}  #If the first button is pressed, set the number to 1.
if (Button2) {G["button",number] = 2}  #If the second button is pressed, set the number to 2.
```

```
#Expression nr 2
@inputs Check
@persist G:gtable
if (first()) {G = gTable("mygroup",0)} #Get the same gtable as the first E2
if (Check) { #When you press the check button...
    print("The last pressed button is:", G["button",number],"!") #Tell me which button was pressed on the first E2
}
```

Old syntax (By ZeikJT)

Description: Example code using two expression2 chips to communicate wirelessly using a custom groupset.

Code

```
@name Global Writer
@inputs Input
interval(100)
gSetGroup("Wireless1")
gSetNum("A"Input)
```

```
@name Global Reader
@outputs Output
interval(100)
gSetGroup("Wireless1")
Output=gGetNum("A")
```

--ZeikJT 17:45, 11 January 2009 (GMT-6)

Using Built-In Ranger (by ZeikJT)

Description: Example code using and expression2 chip to do ranger calculations.

Code

```
@name Ranger Example
@outputs Distance Position:vector Hit Entity:entity
@persist T:ranger
interval(100)
T = ranger(500)
Distance = T:distance()
Position = T:position()
Hit = T:hit()
Entity = T:entity()
```

-ZeikJT 12:43, 20 January 2009 (GMT-6)

Built in ranger and wire colorer example (???)

Description: Using expression 2 to color an object between a built in ranger just like the wire colorer. It also resets the entities color when its not between the built in ranger.

Code

```
@name E2 Colorer and Ranger
@persist Last:entity Col:vector
interval(100)
```

```
Ranger = ranger(300):entity()
if (Ranger != Last) {
    Last:setColor(Col)
    Col = Ranger:getColor()
    Ranger:setColor(255,0,0)
    Last = Ranger
}
```

Using Hints (by hurricaaane)

hint(S,N) is a simple way to display strings onto your screen, including for debugging pruposes. It will appear like messages "Undone Wire Gate" appear. You can also spawn a notice popup when the expression is spawned using if (first()) .

Usages in a vehicle could be to display a notice to the passengers about the usage of the vehicle.

Code

```
@name HintPopup Test
@inputs EntMarkChair:entity
@persist Time Active
interval(1000)
if (first()) {hint("Hi! Wire EntMarkChair to an Entity Marker linked to a chair.",5.5)}

Active=!(!EntMarkChair:driver())

if ( Active & ($Active != 0)) {
    hint("Someone entered the chair!",3)
}

if (!Active & ($Active != 0)) {
    hint("Someone left the chair!",3)
    Time = 0
}

if (Active) {
    Time++
    EntMarkChair:hintDriver("You're in a " + EntMarkChair:type() + " for " + Time + " seconds",3)
}
```

Holo Emitters (by Entoros)

Description: These are two expressions that both do something with holoemitters; the first creates a sphere, the second follows you around in a cool helix.

Code

```
@name Holo Sphere
@inputs Holo:wirelink
@persist Timer

interval(20)
Timer += 0.02

ZCos = cos(Timer*5.625)
Holo["X",number] = sin(Timer*360)*ZCos)*20
Holo["Y",number] = cos(Timer*360)*ZCos)*20
Holo["Z",number] = sin(Timer*5.625))*20
Holo["FadeRate",number] = 1
Holo["Active",number] = 1

@name Holo Helix
@inputs Holo:wirelink Holo2:wirelink Active
@persist Timer
```

```
interval(20)
Timer += 0.02
PlayerPos = owner():pos()

Holo["X",number] = sin(Timer*180)*20 + PlayerPos:x()
Holo["Y",number] = cos(Timer*180)*20 + PlayerPos:y()
Holo["Z",number] = cosr(Timer)*35 + 35 + PlayerPos:z()

Holo2["X",number] = sin(Timer*180 + 180)*20 + PlayerPos:x()
Holo2["Y",number] = cos(Timer*180 + 180)*20 + PlayerPos:y()
Holo2["Z",number] = cosr(Timer)*35 + 35 + PlayerPos:z()

Holo["FadeRate",number] = 50
Holo["Active",number] = Active
Holo2["FadeRate",number] = 50
Holo2["Active",number] = Active
```

Chip Buddy (by Marshmellow)

Description: This is simply a chip that floats above your head and follows you around. All you have to do is spawn it and click on it with your physgun to make it work. I made it for those of you who are trying to learn vector basics. E-mail me at justizzle550@gmail.com if you have any questions.

Code

```
@name Chip Buddy
runOnTick(1)
EE=entity() #Expression Entity
applyForce(((owner():shootPos()+vec(0,0,50)-EE:pos())*10-EE:vel())*EE:mass())
#the +vec(0,0,50) part is so that it floats above your head instead hitting it.
```

Find player by steamID or partial steamID (by Venompapa)

Description: Use "for" loop function to find players by full or partial steamID.

Code

```
@name Find Player by SteamID
@inputs Find
@outputs Ply:entity
@persist N

if (Find & findCanQuery()) {
    Ply = noentity()
    N = findByClass("player") #Get all players and N = how many player finded.

    for (I=1, N) {
        if (findResult(I):steamID():find( "PARTIAL_STEAMID")) {
            Ply = findResult(I)
            break
        }
    }
}
```

Checking a material on a prop (by Godd)

Description:A code that allows you to check the material of a prop by placing the e2 on the prop

Code

```
E=entity():isWeldedTo()
print(E:getMaterial())
```

Counter Example (by Slaiiz)

Description: This is a not-so-hard-to-understand code that shows how to make a switcher using any triggerable thing.

Code

```
@name Counter Example
@outputs Color:vector
@persist Click Counter Count Modes:array
runOnTick(1) #Start here at each tick

Click = owner():keyAttack2() #Triggered when you press Mouse2

if (first()) {  #Triggers only once when spawning the chip.
    #Strings (1 String = 1 Value)
    Modes[1,string] = "Mode: Slow"
    Modes[2,string] = "Mode: Normal"
    Modes[3,string] = "Mode: Fast"

    Counter = 1            #Reset Counter to 1
    Count = Modes:count()  #How many String values stored

    #Values to be used with Strings
    Modes[4,vector] = vec(255,0,0)
    Modes[5,vector] = vec(255,255,0)
    Modes[6,vector] = vec(0,255,0)

    Color = Modes[Counter + Count,vector]  #We add Count to find Values

    entity():setColor(Color) #Set the color of the entity = chip
    print(4,Modes[1,string]) #You can change 4 to 1,2 or 3

    #1,2 prints to console
    #3 prints to chat
    #4 prints as a white string on the middle of the screen
}

if (Click & $Click) {   #Triggers once per click
    Counter++  #Increment Counter to get the next mode
    if (Counter > 3) {Counter = 1}  #Reset Counter to 1 if greater

    Color = Modes[Counter + Count,vector] #We add Count to find Values

    entity():setColor(Color) #Set the color of the entity = chip
    print(4,Modes[Counter,string]) #You can change 4 to 1,2 or 3

    #1,2 prints to console
    #3 prints to chat
    #4 prints as a white string on the middle of the screen
}
```

Advanced

Advanced Player Targetting (by Jackorobot)

Description: This is an code that targets people easier then Wodden's Code. Usage: type in chat: "/target [name|partial name]"

Code

```
@name Player Targetting
```

```
@persist Target:entity TargetS:string LastSaidL:string

runOnChat(1) #makes it run on chat

LastSaidL=owner():lastSaid():lower() #makes lastSaid lower-case

if (LastSaidL:sub(1,8)=="/target ") {
    TargetS=LastSaidL:sub(9,LastSaidL:length())
} #gets the targets name/partial name

if (TargetS!="") {
    Target=findPlayerByName(TargetS)
} #searches the player
#Minor Fix - Bayangan (12/07/2010)
print("Target: "+Target:name()+"!") #prints the player's name in chat window
```

Player targeting via Chat (by Wodden)

Description: With this handy little expression you can target players, by typing !target followed by the name.

Code

```
@name Player Chat Target
@outputs Target:entity
@persist Targ
runOnChat(1)
if (first()) {Targ = 0}
if (chatClk() & owner():lastSaid() == "!target") {
    Targ = !Targ
    if     ( Targ) { hint("Targeting enabled" , 2) }
    elseif (!Targ) { hint("Targeting disabled", 2) }
}
if (chatClk() & Targ == 1 & (owner():lastSaid() != "!target")) {
    Target = findPlayerByName(owner():lastSaid())
    Targ = 0
    hint("Target: " + Target:name() + "", 2)
}
```

Velocity Stabilisation (by fishface60)

Description: The purpose of this expression is to allow thrusters placed anywhere on a contraption to right the orientation. To make it work on your contraption you may need to reassign the forward, right and up vectors dependant on which way you like up. Also, some Values may need to be negated, and for it to run smoothly you may have to multiply by some constants. To use, create an entity marker and link it to the prop you want to use for the base, in my case a vehicle pod. Then spawn a vector thruster and use the wirelink tool on it to create the wirelink output. After this just wire it up. Remember that you need one of these gates per vector thruster, it doesn't have to be located near the thruster though, it can be closer to the control circuits to make it easier to wire, or it could be on a control board, which could be recommended to reduce lag.

Code

```
@name Angular Velocity Stabilisation
@inputs Thruster:wirelink Base:entity PitchInput YawInput RollInput
interval(20)
AngVel = Base:angVel()
CentreVector = (Base:pos() - Thruster:entity():pos()):normalized()
PitchVector = CentreVector:cross(Base:right())
YawVector = CentreVector:cross(Base:up())
RollVector = CentreVector:cross(Base:forward())
Thrust = PitchVector*(AngVel:pitch() + PitchInput)
Thrust += YawVector*(AngVel:yaw() + YawInput)
Thrust += RollVector*(AngVel:roll() + RollInput)
```

```
Thruster["Vector",vector] = Thrust
```

Missile (by Exitium)

Description: As the name suggests, this is my homing missile expression. It needs the following:

- Vector thruster. - Thruster. - Target finder. - Button or some other input to fire it. - And a missile!

Simple to wire:

- Target Finders entity -> TargetE. - Target Finders 1 -> HasTarget. - Button or other input -> Fire.

- CoordsV -> Vector thrusters vector input. - Go -> Thrusters A and Vector thrusters Mul.

The other crap is for a multi stage, multi warhead missile that uses weld latches, but that parts still work in progress.

Code

```
@name Homing Missile Advanced
@inputs TargetE:entity HasTarget WantToDupe Fire
@outputs CoordsV:vector Latch Launch Target Go
@persist Track

CurPosV = entity():pos()
TargetV = TargetE:pos()

if (Fire == 1) {
    Go = 1
    timer("delay", 2000)
} else {
    Track = 0
}

if (clk("delay") | Track) {
    Track = 1
    CoordsV = CurPosV - TargetV
}

Target = HasTarget

if (HasTarget & (TargetV:distance(CurPosV) <= 3500)) {
    Latch = 0
    Launch = 1
} else {
    Latch = 1
    Launch = 0
}

if (WantToDupe) {Latch = 0}
```

Gyroscopic Stabilization (by chinoto)

Description: Using force functions to make a expression resist movement and gravity, and align itself with ang(0,0,0).

Code

```
@name Force Test
runOnTick(1)
E1=entity(),EE=(E1:isWeldedTo() ? E1:isWeldedTo() : E1)
#Decides whether to use the expression or attached entity for force calculation
EA=-EE:angles()*15-EE:angVel()*2 #The angle we intend to force with
EV=EE:massCenter(),ER=EE:right(),EF=EE:forward(),EU=EE:up() #Set up stuff to make manipulation easier
```

```
EE:applyForce((vec(0,0,9.015)-EE:vel())*EE:mass()) #Prevents it from falling by defying gravity and movement
Lev=EE:inertia():length() #Determines how much "Leverage" is needed to rotate the entity at a reasonable rate

EE:applyOffsetForce( EU*EA:pitch(),EV-EF*Lev) #Rotates the entity according to the angle EA
EE:applyOffsetForce(-EU*EA:pitch(),EV+EF*Lev) #Imagine a rod coming out of the entity's axes with a length of Lev
EE:applyOffsetForce( ER*EA:yaw()  ,EV-EF*Lev) #And at the end of each rod is a thruster pushing
EE:applyOffsetForce(-ER*EA:yaw()  ,EV+EF*Lev) #up or down depending on the angle given
EE:applyOffsetForce( EU*EA:roll() ,EV-ER*Lev)
EE:applyOffsetForce(-EU*EA:roll() ,EV+ER*Lev)
#EE:applyAngForce(EA*Lev) #This can replace the above 6 lines
```

Code Because of complaints from Jimlad about it being hard to understand and getting people into the "bad" habit of using short variables, I made an alternate with longer variable names.

```
@name Force Test
runOnTick(1)
Temp=entity(),Entity=(Temp:getConstraints():count() > 0 ? Temp:isWeldedTo() : Temp)

Angle=-Entity:angles()*15-Entity:angVel()*2
Center=Entity:massCenter(),Right=Entity:right(),Forward=Entity:forward(),Up=Entity:up()
Entity:applyForce((vec(0,0,9.015)-Entity:vel())*Entity:mass())
Leverage=Entity:inertia():length()

Entity:applyOffsetForce( Up   *Angle:pitch(),Center-Forward*Leverage)
Entity:applyOffsetForce(-Up   *Angle:pitch(),Center+Forward*Leverage)
Entity:applyOffsetForce( Right*Angle:yaw()  ,Center-Forward*Leverage)
Entity:applyOffsetForce(-Right*Angle:yaw()  ,Center+Forward*Leverage)
Entity:applyOffsetForce( Up   *Angle:roll() ,Center-Right  *Leverage)
Entity:applyOffsetForce(-Up   *Angle:roll() ,Center+Right  *Leverage)
#Entity:applyAngForce(Angle*Leverage) #This can replace the above 6 lines
```

Chinoto Vokro 12:22, 14 February 2009 (GMT-6), Fixed by Materie 15:09, 6 September 2009 (GMT+1)

Force Circle (by chinoto)

Description: Uses force functions and a while loop to make a bunch of entities orbit.

Code

```
@name Force Circle
@outputs Count Run Radius
@persist Props:array
runOnTick(1)
if (Run) {
    Owner=owner()
    if (findCanQuery()) {
        findIncludePlayerProps(Owner) #The functions only work on the owner's props, so we only use them on the owner's props
        findByModel("models/props_junk/PopCan01a.mdl") #Pick a model
        Props=findToArray() #Save what we found
        Count=Props:count() #How many entities we found
    }

    if (tickClk()) {   #Set stuff up first to use less ops
        Cur=curtime()  #Just an easy way of getting an incrementing value
        Radius=max(Count*50/pi()/2,100) #Creates a radius with atleast 50 units between each prop
        #Pos=mix(Owner:pos(),Owner:shootPos(),(sin(Cur*45)+1)/2)   #Moves the center position from the feet of the owner to the head and back
        Pos=entity():pos() #A sun
        Orbit1=vec(Radius/3,0,10):rotate(0,Cur*90,0)   #A planet orbitting the sun
        Index=0
        while (Index<Count) {
            Index++,TE=Props[Index,entity] #TE=Target Entity, yay less ops
            Orbit2=vec(Radius,0,10):rotate(0,-(360/Count*Index + Cur*45),0)    #Moons orbitting the planet
            TE:applyForce(((Pos+Orbit1+Orbit2-TE:massCenter())*10-TE:vel())*TE:mass()) #Standard formula
            TE:setColor(randint(128,255),randint(128,255),randint(128,255))    #Pretty colors
        }
```

```
    }

    if ($Count>0) {timer("trail",2000)} #If the amount of props has increased then set new trails
    if (clk("trail")) {
        Color=vec(255,255,255) #
        foreach (Index,Prop:entity=Props) {
            Prop:setTrails(100,0,1,"trails/physbeam",Color,255)
        }
    }
}
Run=(minquota()>100) #If there aren't more than 100 ops left, then the above code block will be skipped the next execution to prevent a shutdown
```

Defy Gravity (by chinoto)

Description: Uses force functions and while loops to make a bunch of entities defy gravity and to a small extent, velocity.

Code

```
@name Defy Gravity
@inputs Stop
@outputs Count1 Count2
@persist A1:array A2:array Find
runOnTick(1)
if (findCanQuery()) {
    Find=(Find+1)%2
    findClearBlackList(),findClearWhiteList()
    findIncludePlayerProps(owner())
    if (Find==0) {
        findByClass("prop_physics")
        A1=findToArray()
    }
    if (Find==1) {
        findByClass("sent_ball")
        A2=findToArray()
    }
}

if (!Stop) {
    Gravity=vec(0,0,9.015)
    Count1=A1:count()
    T=0
    while ((T<Count1)&(minquota()>100)) {
        T++
        TE=A1[T,entity]
        TE:applyForce((Gravity-TE:vel()/100)*TE:mass())
    }

    Count2=A2:count()
    T=0
    while ((T<Count2)&(minquota()>100)) {
        T++
        TE=A2[T,entity]
        TE:applyForce((Gravity-TE:vel()/100)*TE:mass())
    }
}
```

Radar Chip (Ranger and Wirelink) (By Coder0xff)

Description: Use the ranger commands to feed a digital screen with depth perception. All you need is this code in a chip and a digital screen. Shoot the digital screen once with the "Expression 2 - wirelink" stool. Go to the wire stool, and connect the LCD input of the chip with the [wirelink] output on the monitor. Use the physics gun to move the chip around and watch the display update. Make sure you follow the short instruction at the end of the code to take that block of code and copy paste it so there are 16 copies, total. It'll update much faster that way.

Code

```
@name RADAR Chip
@inputs Res Rate Zoom Mem:wirelink LCD:wirelink
@outputs Inc XO YO D X Y
@persist X Y MinDepth MaxDepth FindMin FindMax
#Rate is the number of times per second for full update
#For zoom, 1 = 90 degress, larger values equal small degress
#if it can't keep up then increase the value of Mult and
#copy paste the code block accordingly (to many and it just gets slower)

Mult = 16 #this is the number of copies of the scan code there is

if (Res  == 0) {Res = 32}
if (Rate == 0) {Rate = 5}
if (Zoom == 0) {Zoom = 1}
Inc = 2 / Res  #increment between each offset
interval(1000 / Rate * Mult / Res ^ 2 )

#Start of repeating code block
XO = (X * Inc - 1) / Zoom  #XO & YO are offsets
YO = (Y * Inc - 1) / Zoom
R = ranger(50000, XO, YO)
D = R:distance()
if (D < FindMin) {FindMin = D}
if (D > FindMax) {FindMax = D}
if (X < 32 & Y < 32) {LCD:writeCell(X + Y * 32, D - MinDepth / (MaxDepth - MinDepth) * 255)}
Mem:writeCell(X + Y * Res, D)
X++
if (X >= Res) {X = -1, Y++}
if (Y >= Res) {
    Y = -1
    MinDepth = FindMin
    MaxDepth = FindMax
    FindMin = D
    FindMax = D
}
#End of code block - repeat this for a total of 16 copies
```

Applying Entity Discovery: Thermometer (by Entoros)

Description: Uses Gwahir's Entity Discovery functions to create a thermometer of sorts. It detects whether entities near it are on fire, then changes the temperature accordingly.

- Note: this code is not perfect, but it's a good example.

Code

```
@name Thermometer
@outputs Temp
@persist FindNum NumEnts Dist Ent:entity TempChange FindNum2 Ent2:entity
interval(50)

findInSphere(entity():pos(),1000)
findClipToClass("prop_physics")
NumEnts = findSortByDistance(entity():pos())
if (FindNum > NumEnts) {FindNum = 1}

Ent = findResult(FindNum)
Dist = entity():pos():distance(Ent:pos())
if (Ent:isOnFire() & Dist <= 500) {
    if (FindNum2 > NumEnts) {FindNum2 = 1}
    Ent2 = findResult(FindNum2)
    if (Ent2:isOnFire() & Ent2:pos():distance(entity():pos()) < Dist) {Ent = Ent2}
    FindNum2 += 1
```

```
    if     (Dist > 150) {TempChange = (500 - Dist)/10}
    elseif (Dist > 100) {TempChange = (500 - Dist)/5}
    elseif (Dist >  40) {TempChange = (500 - Dist)/3}
    elseif (Dist <  40) {TempChange =  300 - Dist}
} else {
    FindNum +=1
    TempChange = 0
}

Temp = 72 + round(TempChange) - (entity():isUnderWater()*20+TempChange)
```

NPC pet (by Bobsymalone)

Description: This is a simple demonstration of the NPC control functions, being used to turn the nearest NPC into a pet. Wire a button or numpad input to Find and press it to find the nearest NPC.

Code

```
@name NPC pet
@inputs Find
@persist NPC:entity

Owner = owner()
OwnerPos = Owner:pos()

if (Find&~Find) {
    findByClass("npc_*")
    NPC = findClosest(OwnerPos)
    NPC:npcRelationship("player","neutral",900)
    NPC:npcRelationship(Owner,"like",999)
}

if (NPC) {
    interval(1000)
    DPos = NPC:pos() - OwnerPos
    WalkPos = OwnerPos + DPos:normalized()*100

    #This next bit stops the NPC from continually trying to run to spot it's standing in,
    #and also stops it being distracted while standing still

    if (DPos:length()>120) {
        NPC:npcGoRun(WalkPos)
    } else {
        NPC:npcStop()
    }
}
```

Introduction to applyForce() and applyAngForce() (by Mr. Scruff)

Description: A really baisic detailed introduction to applyForce() and applyAngForce(). This Expression 2 baisically makes whatever you spawn it on hover above your head and look where you look.

Code

```
@name ApplyForce() and ApplyAngForce() Tutorial
@persist Vec:vector Ang:angle
##-Start-##
if (first()) {   ##-We run these things ONLY when the E2 has just been spawned-##
    runOnTick(1)   ##-Make the E2 "refresh" every tick (66 times a second or about 15ms on most servers)-##
}

##-Abbreviations-##
E   = entity():isWeldedTo() ##-Define what the chip is welded to-##
```

```
O   = owner()  ##-Define who YOU are-##
Pos = O:pos()  ##-Define Where YOU are-##

##-Vectors-##
Goto = O:pos()+vec(0,0,100) ##-This is the vector for exactly 100 inches above your feet-##
Cur  = E:pos() ##-The current position of what the E2 is welded onto-##

Vec = Goto-Cur ##-This creates a local vector to tell which direction to move the plate, Its allways where you want it to be minus where it-##
##-currently is-##

E:applyForce((Vec + $Vec*5)*E:mass()*5) ##-Apply the necessary force relative to the objects mass. The $Vec*5 part stabilises it but you must
##-put it in the persists line at the top for it to work-##
##-I find E:mass()*5 works on most things but with smaller props you might have to adjust this-##


##-Angles-##
P = E:elevation(O:aimPos())##-This gets the elevation towards where you are looking-##
Y = E:bearing(O:aimPos())  ##-This gets you the bearing towards where you are looking-##
R = E:angles():roll()      ##-This gets you the roll angle of the prop to allow it to stay upright-##

Ang = -ang(P,Y,R)  ##-Composing the angle, ApplyAngForce() likes this part to be negative so you put a - infront of it-##

E:applyAngForce((Ang + $Ang*5)*E:mass()*5) ##-Apply The necessary angular force relative to the objects mass. Again the $Ang*5 part stabilises it-##
##-Again, you may find it necessary to change the multiplier-##
```

Apply Angle Force Turret (by Sax)

Description: It's pretty basic really, basically you wire the Vehicle:entity input to an entity marker that is linked to a pod and Activate is connected to whatever you want as "On". The turret has support for Camera controller.

Code Angle Calcs rewritten by chinoto

```
@name Apply Force Turret
@inputs Vehicle:entity Activate
@outputs CameraPos:vector Camera:vector
runOnTick(1)
EE = entity() #Expression Entity
EE:setMass(1000)
AimPos = Vehicle:driver():aimPos()

#Angle Calcs
Ang1 = ang(EE:elevation(AimPos), EE:bearing(AimPos), EE:angles():roll())
Ang  = (Activate ? Ang1 : EE:angles())
applyAngForce((-Ang*50-EE:angVel()*5)*EE:inertia():length())
#applyForce(-EE:vel()*EE:mass()) #prevent it from moving when uncommented

A = EE:pos()
CameraPos = A+vec(0,0,10)
Camera = AimPos-A
```

Text controlled props (by Shoffing)

Description: This expression can control props in many different ways using player input from chat.

Download Link

Simply click here and copy-paste into an expression 2 gate to use it. "Text Controlled Props" (https://docs.google.com/View?revision=_latest&docid=ddm9vh2p_27fn9p7ng5&hl=en)

Made by shoffing; cleaned, organized, and perfected by Chinoto.
Text controlled props is being continually worked on by both shoffing and chinoto, so check back to see if there are any updates.

Artillery (by Shoffing)

Description: This is meant to be a realistic sounding/feeling/looking artillery barrage. It uses soundPlay(),applyForce(),timers, and many more. Just put one on every prop you want to use (you need 1 per prop for the sounds), then say "/t [name]" (no brackets, not case sensitive) Then, when your ready, say "/fire" and all the props will fire at a random interval at a random point in a 250 unit radius of the targeted player. It's really cool, it shakes the ground too haha. If you're looking for effect, use this.

Code Go here to get it!!! It was too big again :/ "Artillery" (https://docs.google.com/Doc?id=ddm9vh2p_30fzgmd5d9)

Bone Example: Zombie (by Shoffing)

Description: This expression will turn any human-looking ragdoll into a zombie that follows you, arms outstretched. Go ahead and add your own sound effects, I just thought of doing that now. xD

Code

```
@name Zombie (by Shoffing)
@persist Base:vector Delay
interval(10)

E = entity():isWeldedTo()
OPos = owner():shootPos()
E:setMass(100)

#Sets up parts we're gonna use
RH = E:bone(7) #Right hand
LH = E:bone(5) #Left hand
Head = E:bone(10)
LF = E:bone(13) #Left foot
RF = E:bone(14) #Right Foot

if ((E:pos():distance(OPos) < 100) & !Delay) {
    soundPurge()
    timer("delay",750)
    Force = 10^9
    soundPlay(0,100,"npc/zombie/zo_attack2.wav")
    E:applyForce(Force * (OPos - E:pos()))
    Delay = 1
} else {
    Force = 3
    E:applyForce(E:vel() * -7)
}

if (clk("delay")) {Delay = 0}

#Extends teh arms
RH:applyForce(Force*(OPos - RH:pos()))
LH:applyForce(Force*(OPos - LH:pos()))

#To make it stand up, I pushed the feet down and the head up.
Base = vec(0,0,150)
LF:applyForce(-Base)
RF:applyForce(-Base)
Head:applyForce(Base*10)
```

Music Example (by Shoffing)

Description: Using lots of time and proportions, I was able to "convert" 2 octaves of the musical scale into soundPitch() changes for the sound "synth/tri.wav" (which happened to be 440 frequency)

Using these discoveries, I have created the love theme from Star Wars using timers, soundPitch(), and soundPlay(). Feel free to implement.

Code

```
@name Star Wars Love Theme
interval(10)

D3  =  33.37
E3  =  37.456
C3  =  59.4
C3s =  62.9
D3  =  66.7
D3s =  70.7
E3  =  74.9
F3  =  79.3
F3s =  84
G3  =  89
G3s =  94.3
A4  = 100
A4s = 105
B4  = 112.2
C4  = 118.925
C4s = 125.993
D4  = 133.484
E4  = 149.83
F4  = 158.740
G4  = 178.179

#Star Wars Love Theme
Tempo = 365.38

if (first()  ) {timer("00",Tempo)}
if (clk("00")) {soundPlay(0,Tempo*3  -50,"synth/tri.wav"),soundPitch(0,F4),timer("01",Tempo*3  )}
if (clk("01")) {soundPlay(0,Tempo    -50,"synth/tri.wav"),soundPitch(0,D4),timer("02",Tempo    )}

if (clk("02")) {soundPlay(0,Tempo*0.3-50,"synth/tri.wav"),soundPitch(0,G4),timer("03",Tempo*0.3)}
if (clk("03")) {soundPlay(0,Tempo*0.3-50,"synth/tri.wav"),soundPitch(0,F4),timer("04",Tempo*0.3)}
if (clk("04")) {soundPlay(0,Tempo*0.3-50,"synth/tri.wav"),soundPitch(0,E4),timer("05",Tempo*0.3)}
if (clk("05")) {soundPlay(0,Tempo*3  -50,"synth/tri.wav"),soundPitch(0,F4),timer("06",Tempo*3  )}
if (clk("06")) {soundPlay(0,Tempo    -50,"synth/tri.wav"),soundPitch(0,D4),timer("07",Tempo    )}

if (clk("07")) {soundPlay(0,Tempo*0.3-50,"synth/tri.wav"),soundPitch(0,F4),timer("08",Tempo*0.3)}
if (clk("08")) {soundPlay(0,Tempo*0.3-50,"synth/tri.wav"),soundPitch(0,E4),timer("09",Tempo*0.3)}
if (clk("09")) {soundPlay(0,Tempo*0.3-50,"synth/tri.wav"),soundPitch(0,D4),timer("10",Tempo*0.3)}
if (clk("10")) {soundPlay(0,Tempo*3  -50,"synth/tri.wav"),soundPitch(0,E4),timer("11",Tempo*3  )}
if (clk("11")) {soundPlay(0,Tempo    -50,"synth/tri.wav"),soundPitch(0,C4),timer("12",Tempo    )}

if (clk("12")) {soundPlay(0,Tempo*3  -50,"synth/tri.wav"),soundPitch(0,D4),timer("13",Tempo*3  )}
if (clk("13")) {soundPlay(0,Tempo    -50,"synth/tri.wav"),soundPitch(0,C4),timer("14",Tempo    )}
if (clk("14")) {soundPlay(0,Tempo*3  -50,"synth/tri.wav"),soundPitch(0,A4),timer("15",Tempo*3  )}
if (clk("15")) {soundPurge(),reset()}
```

HINT: Creating a 2D array with 1D arrays using strings (by Shoffing)

Description: Lets say you wanted to make a chess board and needed something to save the position of the pieces into.
You cant use just a normal array, right? Because you need 2 dimensions (x and y) to fill in the board.
My way of getting around this is to use strings in all the slots of an array, and checking their indexes
to get information from them. For example:

Code

```
@name 2D array (Chess example)
```

```
@persist Board:array
interval(10)

if (first()) {
    #Make empty board
    I = 0
    while(I <= 8) {
        #Every - is a blank spot
        Board[I,string] = "--------"
    }
}

#Lets place a knight on spot (6, 5)
Board[6,string]:index(5) = "K"
#Now the 6th slot in board reads "----K---"

#Your board looks like:
# --------
# --------
# --------
# --------
# --------
# ----K---
# --------
# --------

#To manipulate the pieces, you would just use string manipulation, subtraction, etc.
#Hopefully you understood my logic in doing this, if not email me at shoffing@gmail.com
```

Remote Trail Maker (by OmicroN)

Description: I thought that it would be neat to make chip that would allow you to set trails to any entity that YOU OWN just by looking at it and pressing a button, enjoy

Code

```
@name Remote Trail Maker
@inputs Set


E = owner():aimEntity()#This will set up our variables for Entity
Startsize = 40         #Let's set up our Trails starting size
Endsize   = 10         #Next we'll set up our ending size, so the trail can fade to a size
Length    =  5         #Now we set the time duration of trail, the higher the number the longer it stays

#Here's a list of possible materials to use for your Trail
Physbeam = "trails/physbeam"
Laser    = "trails/laser"
Smoke    = "trails/smoke"
Electric = "trails/electric"
Plasma   = "trails/plasma"
Tube     = "trails/tube"
Love     = "trails/love"
Lol      = "trails/lol"

#I like to use a sub-selector for variables like this
#-so I don't have to go edit the code
#Use Material = THE MATERIAL STRING YOU WANT TO USE
Material = Electric

#Now for our color, this will be in the form of a Vector
#Your first value will be Red, use range from 0 to 255
#Your second value will be Green, use range from 0 to 255
#Your third value will be Blue, use range from 0 to 255
Color = vec(255,255,0)
```

```
#Now we can set alpha, which is the level of transparency
#Alpha uses 0 to 255 range as well, 255 being solid visible
Alpha = 255

#This statement sets the application of the trail to the entity
#-that the player is aiming at.  So wire 'Set' to a 'Numpad_Input'
#-with the values set to 'OFF=0' and 'ON=1'
#When you hit Set the entity will have a trail attached to it
if (Set) {E:setTrails(Startsize,Endsize,Length,Material,Color,Alpha)}
```

World Orientation (by coder0xff)

Description: Creates cones and a sphere to indicate the positive world axes. They grow and shrink based on the owners distance so that they always fill about the same space on screen.

Code

```
@name World Orientation
interval(100)
EPos = entity():pos()
Scalar = 0.02 * owner():pos():distance(EPos)
Size = vec(1, 1, 1) * Scalar
if (first()) {
    holoCreate(0, EPos + vec(Scalar * 2, 0, 0), Size, ang(90, 0, 0),   vec(255, 0, 0))
    holoShadow(0, 0)
    holoModel(0, "cone")

    holoCreate(1, EPos + vec(0, Scalar * 2, 0), Size, ang(90, 0, -90), vec(0, 255, 0))
    holoShadow(1, 0)
    holoModel(1, "cone")

    holoCreate(2, EPos + vec(0, 0, Scalar * 2), Size, ang(0, 0, 0),    vec(0, 0, 255))
    holoShadow(2, 0)
    holoModel(2, "cone")

    holoCreate(3, EPos + vec(0, 0, 0),          Size, ang(0, 0, 0),    vec(255, 255, 255))
    holoShadow(3, 0)
    holoModel(3, "sphere")
}

holoPos(0, EPos + vec(Scalar * 2, 0, 0))
holoScaleUnits(0, Size)

holoPos(1, EPos + vec(0, Scalar * 2, 0))
holoScaleUnits(1, Size)

holoPos(2, EPos + vec(0, 0, Scalar * 2))
holoScaleUnits(2, Size)

holoPos(3, EPos + vec(0, 0, 0))
holoScaleUnits(3, Size)
```

Angular stabilization using applyTorque and quaternions (by Fizyk)

Description: Stabilizes an entity in an orientation given by an angle. If you want to make something facing a particular direction, get the direction as a vector V, use V:toAngle() and proceed with this code.

Code

```
@name Stabilization
@inputs E:entity A:angle

#E is the stabilized entity
```

```
#A is the target angle

runOnTick(1)

TarQ = quat(A) #Calculate quaternion for target orientation
CurQ = quat(E) #Calculate current orientation quaternion

#TarQ/CurQ is a quaternion representing the rotation that will rotate the object from current orientation to the target one.
Q = TarQ/CurQ

#applyTorque() works on intrinsic vectors! Get the rotation vector and transform it.
V = E:toLocal(rotationVector(Q)+E:pos())
#Alternatively, can use "V = transpose(matrix(E))*rotationVector(Q)"

#Apply torque. angVelVector() works like a delta term.
#Factors 150 and 12 can be adjusted to achieve best effect.
E:applyTorque((150*V - 12*E:angVelVector())*E:inertia())

#This is only to stop movement, but it won't make the object float.
#Disable gravity for the entity or write some code for floating if you want to see how this expression works.
E:applyForce(-E:vel()*E:mass())
```

Mass Prop Colorer (by RoflChoppa)

Description: This expression will turn all of your spawned props red as an example of "for loop" usage.

Code

```
@name Prop Color Changer
@persist OwnerProps:array

findIncludePlayerProps(owner())    #exclude everyone else's props from all future finds
findByClass("prop_physics")        #find all props, this will only find your props due to the Include function
OwnerProps=findToArray()           #give the array "OwnerProps" every prop found in the previous find
for (Index=1, OwnerProps:count()) { #run a for loop to color every entity in the array red
    #make a temporary variable, "prop", equal to the entity in the array that corresponds to the for loop's current index
    Prop=OwnerProps[Index,entity]

    #change the color of the entity that "prop" currently equals to red
    Prop:setColor(vec(255,0,0))
}
```

Hologram Clipping Example (by Soul less)

Description: An example of the new holoClip function, used to clip holograms.

Code

```
@name Haloguy's Hologram Clip Example
@inputs
@outputs
@persist Pos:vector Dist
@trigger
#Haloguy's Hologram Clip Example
interval(10)

if (first()) {
    #create plate 1
    holoCreate(1)
    holoScaleUnits(1,vec(50,50,1))

    #create plate 2
    holoCreate(2)
    holoScaleUnits(2,vec(50,50,1))
```

```
    #spawn the cube
    holoCreate(3)

    #turn on clipping for the cube
    holoClipEnabled(3,1)

    #assign a distance between the two plates
    Dist=25

    #position and angle the plates
    holoPos(1,entity():pos()+vec( Dist,0,0))
    holoAng(1,ang()+ang(90,0,0))
    holoPos(2,entity():pos()+vec(-Dist,0,0))
    holoAng(2,ang()+ang(90,0,0))
    Pos=entity():pos()
    holoAng(3,ang())
}

#position our cube hologram
holoPos(3,entity():pos())

#check if the X of our cube is greater then our initial X pos
if (holoEntity(3):pos():x()>Pos:x()) {
    #switch the clipping plane
    holoClip(3,holoEntity(1):pos(),holoEntity(1):up(),1)
}

#check if the X of our cube is less then our initial X pos
if (holoEntity(3):pos():x()<Pos:x()) {
    #switch the clipping plane
    holoClip(3,holoEntity(2):pos(),holoEntity(2):up()*-1,1)
}

#Is the cube between the two plates? If so then enable clipping
if (inrange(holoEntity(3):pos(),Pos-vec(Dist+10,25,25),Pos+vec(Dist+10,25,25))) {
    holoClipEnabled(3,1)
} else { #else disable clipping
    holoClipEnabled(3,0)
}

#Tips:
#You can only clip a hologram once at a time.
#When you execute holoClip, your assigning a plane in the world is the last argument is set to 1
#Imagine a flat plane
#holoClip(Index, Plane center pos, Plane Directional Vector, Global?)
#first we give it the index of the hologram we want to clip
#then we give it the center of the imaginary clipping plane
#next we give it a direction that the place will cut in
#finally, we ask if the plane is local to the chip or global to the world
#holoClipEnabled(Index, 1/0) will turn on clipping if true, or stop clipping if false
#When you use holoClip, remember that you only have to assign it once, the clipping plane will remain there until the chip is reloaded,delted, or you c



#Haloguy's Hologram Clip Example
```

Target Crusher (by I_am_Einstein, modified by Initrd.gz)

Description: A standalone Expression 2 Chip that crushes targets that the user designates through chat, in the format: "/crush <username>"

Code

```
@name Crusher
@persist Target:entity Active
runOnChat(1)
runOnTick(1)
```

```
Owner = owner()

ArrayChat=Owner:lastSaid():explode(" ")
if (chatClk(Owner)&ArrayChat:string(1)=="/crush") {
    Target = findPlayerByName(ArrayChat[2,string])
    if (Target) {Active = 1}
}
elseif (Active & findCanQuery()) {
    Pos = Target:pos()
    findInSphere(Pos,8000)
    findClipToClass("prop_physics")
    PropArray = findToArray()
    for(I=1, PropArray:count()) {
        Prop = PropArray[I,entity]
        Prop:applyForce(Prop:mass()*500*(Pos - Prop:pos()))
    }
    Active = 0
}

#I_am_Einstein's Crusher
```

WAN IP Fetcher (by Nitrousoxide)

Description: An Expression 2 chip that retrieves your WAN (Wide Area Network) IP Address and returns it in the form of a string or in separate octets (array and number form).

Code

```
@name WAN IP Fetcher
@inputs Refresh
@outputs IP:string Octets:array O1 O2 O3 O4
if (Refresh | first()) {
    httpRequest("http://www.whatismyip.com/automation/n09230945.asp")
    IP = httpData()

    Octets = IP:explode(".")
        O1=Octets[1,number]
        O2=Octets[2,number]
        O3=Octets[3,number]
        O4=Octets[4,number]
}
```

Using findRe & replaceRE (by OmicroN)

Description:http://www.wiremod.com/forum/wiremod-tutorials/17699-how-use-findre-replacere.html Visit the link to read about it, I'm putting the link here so people who read the wiki know about it as well.

Console Screen Selection Screen (by GlitchDetector (|SpB| FB GameMaker))

Description: A expression 2 code to make the famous selection screens on automatic gunshops.

Code

```
@name Basic Selection Screen
@inputs Screen:wirelink Up Down Enter
@persist Selection
@outputs S1 S2 S3
S1=0
```

```
S2=0
S3=0
Selection=(Selection+(Up-Down))%3

Screen:writeString("Selection 1",0,1,0,0+999*(Selection==0))
Screen:writeString("Selection 2",0,2,0,0+999*(Selection==1))
Screen:writeString("Selection 3",0,3,0,0+999*(Selection==2))

if (Enter) {
    S1=(Selection==0)
    S2=(Selection==1)
    S3=(Selection==2)
}
```

Have fun with it! Please also add me to steam! "glitchdetector" without quotes.

E2 Gravity Enabled Hoverdrive (by ILOVEPIE)

Description: An expression 2 code to make a Hoverdrive Controller that does not mess up your contraption's physics.

Code

```
@name HOVERDRIVE TELEPORTER CHIP
@inputs JUMP WORLD_X WORLD_Y WORLD_Z WORLDPOS:vector
@outputs
@persist X Y Z
@trigger JUMP
X = WORLD_X
X = WORLDPOS:x()
Y = WORLD_Y
Y = WORLDPOS:y()
Z = WORLD_Z
Z = WORLDPOS:z()
if  (vec(X,Y,Z):isInWorld()){
if ( JUMP == 1){
    I=1
    while(entity():isWeldedTo():isWeldedTo(I)!=noentity()){
    Relativepos = (entity():isWeldedTo():pos()-entity():isWeldedTo():isWeldedTo(I):pos())
    entity():isWeldedTo():isWeldedTo(I):setPos(vec(X,Y,Z)+Relativepos)
        I += 1
    }
    entity():isWeldedTo():setPos(vec(X,Y,Z))
}
}
```

See Also

- Gunnanmons Expression 2 Examples
- Expression 2 Documentation
- Expression 2 Wishlist
- Expression 2 Bugs
- Wiremod

Retrieved from "http://wiki.garrysmod.com/?title=Wire_Expression2:Examples"
Category: Wire Addon Tutorials

- This page was last modified on 12 December 2011, at 02:01.