Just create a game on the flow.

Like the title says this is what I am going to do. Nothing planned just toy around with found scripts and create something that actually works. No deadlines, No hurry just all in the name of fun. Along the way I will expand by adding stuff so at the end (no matter how far away) I will end up with a simple yet funny game ☺. I will do every day I can a little more so we will see where it all will lead to. Take your time to test it and study it all. If I can do it, well so can you. I will do my best to explain things as clear as possible. Keep in mind I am no way a professional coder but I do know how to make things work. Maybe not the best way but hey at least it works and I enjoy it.We begin the game from scratch. I created a work folder called Just a game and added the subfolders: Sounds/Graphics/3Dmodels. I made a simple platform floor and placed a character model on it that has many animations. From here I will start the game creation journey and you are free to join me. I also placed a simple position camera to oversee all of the level.

So basically this is the script we have now :

```
#define PRAGMA_PATH "3Dmodels"
#define PRAGMA_PATH "Graphics"
#define PRAGMA_PATH "Sounds"
function main()
{
video_set(1024,768,32,0);
level_load("justagame.WMB");
}
```

Day one.

I created a work folder and the paths to the game stuff, made the game start in a resolution 1024x768 have 32 depth and start in a window screen. The first thing I am going to do is make our character walk and use it’s walking and standing animations. So when not walking around it will use the standing animation and when walking it will use the walking animations. So what I do is create an easy player movement action script. I will place it under the main function. The cursor keys will be used to move the character in the direction I want it to go.

```
action player_movement()
{
set(my,SHADOW);
var run_percentage = 0;
var stand_percentage = 0;
while(1)
{
if(key_cul && my.pan != 90){ my.pan = 90; }
if(key_cur && my.pan != 270){ my.pan = 270; }
if(key_cuu && my.pan != 0){ my.pan = 0; }
if(key_cud && my.pan != 180){ my.pan = 180; }
result = c_move(me, nullvector, vector(15 * (key_cuu - key_cud) * time_step, 15 * (key_cul - key_cur) * time_step, 0), GLIDE |
IGNORE_PASSABLE | IGNORE_SPRITES);
if(result > 0)
{
run_percentage %= 100;
run_percentage += 15 * time_step;
ent_animate(my, "run", run_percentage, ANM_CYCLE);
}
else{
stand_percentage %= 100;
stand_percentage += 5 * time_step;
ent_animate(my, "stand", stand_percentage, ANM_CYCLE);
}
```

```
wait(1);
}
}
```

I'll assign this player_movement action to our model and run the level. Now I can use the cursor keys to move freely around in my level. You know when I did this for the first time it gave me really a boost to go on. Cause moving a model around is a good start for any game. It does not matter if it's an animated or static model but the fact you can script it to move is just pure fun. You see in the c_move instruction that it uses GLIDE (So it will glide along walls), IGNORE_PASSABLE (So it will go right through passable models) and IGNORE_SPRITES (it will go through any sprite in the level).

It is important to try out these so called flags and use them the right way.

Also I've created a pointer for our player : ENTITY* sid;

in the action player_movement we see this pointer in the line : sid = me;

This way I can add/call all functions or actions that are connected to the player model. (sid).
This so called pointer makes the model have its unique name so it separates itself from all other models used in your game.

So what should I do next ? I know I will create a score on screen ☺ Don't know how we will use it later but hey it's fun to do and we can use the same technique in future projects.

I will use a variable (Kind of container to store stuff in) and call it score. then use a true type font and give it a place on the screen.

I place this script on top of the main :

```
var score = 0;

FONT*      fnt1_pan = "Ariel#22b";
PANEL* pan_score = {digits=10,5,"Score: %00.0f",fnt1_pan,1,score;
layer = 1;
flags = SHOW;
}
```

Layer 1 is the layer on screen. Any higher layer will be placed on top of this one, Any lower layer will be under this one. This way you can stack layers. The flags **SHOW** makes it visible on screen.

I'll run the level and there you have the score on screen.

Now if you want the score text to be in another color you have to add one line that holds a color code. In the flags script we add this.

```
flags = SHOW;green=255; blue=0; red=255;
```

In this case the text will show up in yellow. Green,blue and red form together one color.
Want to know all color settings ? Have a look at :

http://www.rapidtables.com/web/color/RGB_Color.htm

Here you can choose any color you want. Just try it.

So what will I be doing next ? Mmmm, I know I'll make it like Mario ☺ Sid should collect coins.
So I've created a coin model. let's script an action for it so Sid can pick it up.

Place this script under the player movement action.

```
action coins()
{
set(my,PASSABLE | SHADOW | METAL | BRIGHT);
c_setminmax(my);
while(vec_dist(my.x,sid.x) >20)
{
my.roll -= 4* time_step;
my.tilt += 4* time_step;
my.ambient = 200;
my.lightrange = 20;
wait(1);
}
ent_remove(me);
}
```

When the distance between the coin and the pointer (sid) is bigger (>) then 20 it will roll/tilt the coin model and it will have ambient and light added to it. When the distance is smaller (<) to the pointer (sid) the coin model gets removed. Outside the loop I scripted that with the ent_remove(me) line.

So I will place a few coin models in the level and assign them the coin action. Now I'll run it. Sid is able to pick up all the coins simple by running into them. Yes this is used in almost any game where you can pick up stuff. Platform games but even race games, where you can pick up bonus weapons when you drive into them.

Cool start for this first day. It all looks so simple but you see that many lines of script are needed to make it all come to life.

For today I will do one more thing. A sound should play when the coin is picked up and add 10 points to the score. First I need to define the sound file I am using so we add this line above the main :

```
SOUND* coin_snd = "coin.wav";
```

This sound file is stored in the subfolder Sounds I defined on top of the script.
To make the sound play I'll add this line in the action coins script right in top of the ent_remove(me) line. So on removal the sound will play.

```
snd_play(coin_snd, 100, 0);
```

So you see on removal the sound will play. If we I want to add 10 points to the scorei'll will make it happen on this point. I'll simply add one more line on top of the ent_remove(me) line.

```
score +=10;
```

Score is the name of the variable used to hold the score the plus means it will add something to it, in this case 10. So when you would use a minus (-) it would subs tract points.

The full coins script should look like this :

```
action coins()
{
set(my,PASSABLE | SHADOW | METAL | BRIGHT);
c_setminmax(my);
while(vec_dist(my.x,sid.x) >20)
my.roll -= 4* time_step;
my.tilt += 4* time_step;
my.ambient = 200;
my.lightrange = 20;
wait(1);
}
snd_play(coin_snd, 100, 0);
score +=10;
ent_remove(me);
}
```

The c_setminmax(my); is used so it uses the full models box for collision. In this case it's passable so it does not matter. The coin is picked up by using distance to our pointer. (The sid model)

I'll place some more coins in the level and run it. Sid is now able to collect coins and getting points for it. Also the coin sound is playing every time a coin is picked up.

I have now made a very simple Mario game where the player has to collect coins ☺ to earn points and is able to walk around.

Enough for today. You have enough to experiment with. Change sounds, Speed of the character, use different animations. I enjoy this and I hope you do to ☺

Day two.

Let’s see what can I do today. I know I’ll make sure that in this game the player has 3 lives. When all lives are lost it is game over. Ehmmm so what will I need for all of this ?

- A game over message picture. (For if all lives are lost)
- A variable that stores the lives.
- A loose life action/function that makes the player disappear and gets back at its start position.
- A hurt player action/function so sid is able to lose a life.

You see how something simple yet needs a lot of things to be made. Well here we go.

I will start by creating the variable for the lives number. I’ll set it so it starts on 3 lives. Remember the score variable ? well this lives variable works exactly the same. Place this script under your score variable.

```
var lives = 3;
FONT*       fnt1_pan = "Ariel#22b";
PANEL* pan_lives = {digits=300,5,"Lives: %00.0f",fnt1_pan,1,lives;
layer = 1;
flags = SHOW;green=255; blue=0; red=255;
}
```

Now if I run the level it will show the score and lives variables on screen. I told you these variables are used a lot and come in handy.

So when sid has lost 3 lives a game over message should display on screen. I created this panel and stored it into the graphics folder. If I want to use it I’ll have to define it in our script.

Place this script right under the variables.

```
BMAP* gameover_map = "gameover.png";

PANEL* gameover_pan =
{
bmap = gameover_map;
pos_x = 370;
pos_y = 300;
layer = 1;
}
```

So I defined the game over panel. It has layer one and is placed on x 370 and y 300.

Next I’ll need an object that will hurt Sid and make it loose a life. We use the pickup coin script and duplicate it but we call the action now Hurt_sid and add some changes to it. I will make it so Sid will disappear and loses one life then gets teleported back at its starting point. The hurt Sid coin will be removed also and replaced by a new one. When Sid loses all lives the game over message will show.

Place this script under the coin script.

```
action hurt_sid()
{
set(my,PASSABLE | SHADOW | METAL | BRIGHT);
c_setminmax(my);
while(vec_dist(my.x,sid.x) >20)
{
my.roll -= 4* time_step;
my.tilt += 4* time_step;
my.ambient = 200;
my.lightrange = 20;
wait(1);
}
set(my,INVISIBLE|PASSABLE);
ent_remove(sid);
lives -=1;
wait(-2);
if(lives >=1){
ent_create("Sid.mdl", vector(0,0,48), player_movement);
ent_create("hurtcoin.mdl", vector(my.x,my.y,my.z),hurt_sid);
ent_remove(me);
}
else
{

set(gameover_pan,SHOW);
}
}
```

Look close at this script. When Sid is in range Sid will be removed by using its pointer. ent_remove(Sid). The coin is also removed (ent_remove(me). Then a life is subtracted
Lives -=1;

Next is important : If the lives variable is bigger or equal (>=) to 1 we will create the Sid model and its action together with a new hurt_coin and its action. The original coin is removed.

If the lives variable is smaller (<) then 1 we make only the game over panel show by using set(gameover_pan,SHOW);

So there you have it. Our game hero has now 3 lives to use. On zero it's game over ☺ Those variables are handy in use right ?

While I did this I was thinking mmmm you see also in games the use of pictures for lives instead of the variable numbers of lives. Will I be able to create something like that myself ?

Well I am not sure if I am up to the task but it never hurts to try. I know how to define a panel (Graphic) so I made 1 lives panel and define them on screen like this.

```
BMAP* lives01_map = "lives.png";

PANEL* lives01_pan =
{
bmap = lives01_map; // compass_map as panel background
pos_x = 800;
pos_y = 10;
layer = 1;
flags = SHOW;
}


BMAP* lives02_map = "lives.png";

PANEL* lives02_pan =
{
bmap = lives02_map; // compass_map as panel background
pos_x = 870;
pos_y = 10;
layer = 1;
flags = SHOW;
}

BMAP* lives03_map = "lives.png";

PANEL* lives03_pan =
{
bmap = lives03_map; // compass_map as panel background
pos_x = 940;
pos_y = 10;
layer = 1;
flags = SHOW;
}
```

When I run the level know it will show 3 Sid pictures in the screen ☺ Each picture stands for one life. So when you lose a life one picture should be removed ☺

I'll do that by changing some things in our hurt_sid action.

I added these lines to the script :

```
if(lives ==2){
reset(lives03_pan,SHOW);
}
if(lives ==1){
reset(lives02_pan,SHOW);
}
if(lives ==0){
reset(lives01_pan,SHOW);
}
```

So it now checks the lives variable and makes sure that when 1 live is lost the right picture disappears of the screen by removing it. Set means show a panel on screen. Reset means remove it from screen (Make it invisible).

I'll run the level and all works ☺ So I know now how to use variables for lives or use graphics.
You can even now make the variable lives invisible so the graphics will tell the user how many lives he has ☺ just remove the SHOW flags of this variable. Easy done.

You can use this any way you like. It's easy know to make an action that gives you an extra life and show it as variable number or as graphic. So the chapter handling lives I made through.

The last thing for today is : can I make the game restart easy when it's game over. When I test the game I need to be able to restart it whenever I reach game over right ?

I will create a variable for this and place it at top of the script :

```
var enable_gamestart =1;
```

I set this variable on one you will see why later.

Then we create a restart game function. Place this script above your main.

```
function begin_game()
{
if(enable_gamestart ==0){
reset(gameover_pan,SHOW);
level_load("justagame.WMB");
score =0;
lives =3;
set(lives01_pan,SHOW);
set(lives02_pan,SHOW);
set(lives03_pan,SHOW);
enable_gamestart =1;
wait(1);
}
}
```

See what it does ?

When the variable enable_game is equal (==) to 0 it will reset the game over panel, reloads the level , set score back to 0 and lives back to 3. Also the 3 lives panels are set on visible again. Then when game begins the variable is set back to 1 so the restart won't work until its game over.

Now to call this function we will add this line to our main function :

```
on_space = begin_game;
```

So on the press of space bar it will call the function begin_game. Since we have set our variable enable_game to 1 it is needed to set it on 0 cause the function will only work when this variable is 0. So when game over we will put it back to 0 by adding one line to the hurt_sid script.

```
enable_gamestart =0;
```

That's it. Play the game lose 3 lives and it is game over. Restart now by pressing the spacebar.

As I am writing this doc while I am creating this game sometimes I might forget to explain stuff. In this case we made the Sid model move but I did not explain fully how the script works so I'll do that right now by having a look at the player_movement script line by line.

```
action player_movement()     <<<< This is the name of the action you assign in WED.
{
set(my,SHADOW);   <<<< We give the model shadow
var run_percentage = 0; <<<< we use a variable for the run (walk) animation and set it on 0
var stand_percentage = 0; <<<< we use a variable for the stand (stand) animation and set it on 0
while(1) <<<<<  we create a while to create a loop so in this while stuff happens until the loop is broken.
{
if(key_cul && my.pan != 90){ my.pan = 90; }       <<< when user presses cursor left key the model turns 90
                                                      degree direction.
if(key_cur && my.pan != 270){ my.pan = 270; }  <<< when user presses cursor right key the model turns 270
                                                      degree direction.
if(key_cuu && my.pan != 0){ my.pan = 0; }         <<< when user presses cursor up key the model turns 0
                                                      degree direction.
if(key_cud && my.pan != 180){ my.pan = 180; }  <<< when user presses cursor down  key the model turns 180
                                                      degree direction.



result = c_move(me, nullvector, vector(15 * (key_cuu - key_cud) * time_step, 15 * (key_cul - key_cur) *
time_step, 0), GLIDE |  IGNORE_PASSABLE | IGNORE_SPRITES);
```

c_move is the movement instruction, Nullvector is the place it starts from, 15* is the speed it moves it uses the keys cursor up down left and right. Time_step means it will run the same speed on any pc (Frames) The flages are explained.

```
if(result > 0)                                      <<< when result is bigger then 0 (so keys are used here)
{
run_percentage %= 100;                                      <<< total variable percentage of the run animation
run_percentage += 15 * time_step;                            <<<< 15 is the speed of the run animation
ent_animate(my, "run", run_percentage, ANM_CYCLE);          <<<<< make the model play its run animation
                                                                   in a cycle (ANM_CYCLE). If you place NULL
                                                                  instead of ANM_CYCLE the animation will
                                                                  only play once.
}
else                                                <<< if keys are not used so when result is 0
{
stand_percentage %= 100;                            <<< total variable percentage of the stand animation.
stand_percentage += 5 * time_step;                  <<<<  5 is the speed of the run animation

ent_animate(my, "stand", stand_percentage, ANM_CYCLE);    <<<<< make the model play its stand animation
                                                                   in a cycle (ANM_CYCLE). If you place NULL
                                                                  instead of ANM_CYCLE the animation will
                                                                  only play once.

}
wait(1);                                            <<< wait one frame.
}
}
```

So this is how movement works. Well it is one of the ways to use it. Make sure you understand how this works so you can apply it to other models. In med you can open up a model an see it’s animations. Each series of animation has a name (If not you need to rename the frames) In this case the series of animations run and stand are used and work perfect for this game in the making.

Day three.

Well we did pretty much ☺ and our game is really coming to life. For the next things I added walls around so the player is locked inside. I made a door opening and placed a door model. It's a block made in wed so we call it a map entity. (WMB file)

I am thinking if I get a key this door should open for me. So I will need a key model to be picked up and a variable for the amount of keys Sid has. Then when sid picks up the key the door should open for him when he's close.

Creating the variable is easy. Remember how ? We just duplicate our score variable and change it into keys. Ad this script under your other variables.

```
var keys = 0;

FONT*       fnt1_pan = "Ariel#22b";
PANEL* pan_keys = {digits=300,5,"Keys: %00.0f",fnt1_pan,1,keys;
layer = 1;
flags = SHOW;green=255; blue=0; red=255;
}
```

We set the keys at 0 cause no keys are found yet ☺

Now we need an action to assign to the door model that will only work when the key number is higher than 0. Here is how. Place this script under the action Hurt_sid script.

This script will make a door slide open to the left (on x axis) when the variable keys is bigger or equal (>=) then 1. Only then the door will open as it reacts to the distance between the pointer SID and the door model.

```
action doorslide_left()
{
set(my,METAL | SHADOW);
c_setminmax(me);
var init_x;
init_x = my.x;
while (!sid) {wait (1);}
while (1)
{
while (vec_dist (sid.x, my.x) > 100) {wait (1);}

if(keys >=1){
snd_play (door_snd, 100, 0);
keys -=1;
while (my.x > init_x -200)
{
my.x -= 20 * time_step;
wait (1);
}
}
while (vec_dist (sid.x, my.x) < 500) {wait (1);}
snd_play (door_snd, 100, 0);
while (my.x < init_x)
{
my.x += 20 * time_step;
wait (1);
}
my.x = init_x;

}
}
```

You see that the action starts by remembering the start position of the door (init.x)
Then when the distance is bigger than (>) 100 the door will slide open on its X axis for a distance of -200 This only works is the keys variable is bigger or equal (>=) to 1. If else the door will stay closed.
If door has opened and distance player is smaller (<)then 500 the door will close. While opening the door a key is subtracted.

Now the final piece is creating a pickup key action that adds 1 to the keys when picked up.
Place this script under the door script.

```
action pickup_keys()
{
set(my,PASSABLE | SHADOW | METAL | BRIGHT);
c_setminmax(my);
while(vec_dist(my.x,sid.x) >20)
{
my.roll -= 4* time_step;
my.tilt += 4* time_step;
my.ambient = 200;
my.lightrange = 20;
wait(1);
}

keys +=1;
ent_remove(me);
}
```

It's basically the same script as the coin but it gives no score but sets a key number.

Assign the doorslide_left action to the door and give the key model the pickup_keys action.
Run your level. If you walk directly to the door it won't open. Pick up a key and try again voila it opens for you ☺

The door code you can easy change into a door that opens vertical (z) or slides forward/backward (y).
Try to change it. Have you noticed that the door uses a sound ?

```
snd_play (door_snd, 100, 0);
```

Remember in order to use sounds they need to be defined on top of the script like in this case :

```
SOUND* door_snd = "door.wav";
```

You can add sound playing to the pickup_keys action easy, just have a look how I did it with the coin pickup.

Enough for today.

Day four.

The thing that annoyed me was that my sid model walks out of the screen when the doors opens. How come ? Well the answer is simple I use a fixed camera point (position) I remove it from wed and give the Sid model its own camera view. I want it to be isometric/third person view so I added the camera script into the player_movement script by adding these lines.

```
camera.tilt = -25;                <<< look down
camera.x = sid.x-800;             <<< Distance from player
camera.y = sid.y;                 <<< stay with the Y of sid pointer
camera.z = sid.z + 450;           <<< Distance up in the air.
```

Your total player_movement script should be like this :

```
ENTITY* sid;

action player_movement()
{
c_setminmax(my);
set(my,SHADOW);
sid = me;
var run_percentage = 0;
var stand_percentage = 0;

while(1)
{
if(key_cul && my.pan != 90){ my.pan = 90; }
if(key_cur && my.pan != 270){ my.pan = 270; }
if(key_cuu && my.pan != 0){ my.pan = 0; }
if(key_cud && my.pan != 180){ my.pan = 180; }

result = c_move(me, nullvector, vector(15 * (key_cuu - key_cud) * time_step, 15 * (key_cul - key_cur) * time_step, 0), GLIDE |
IGNORE_PASSABLE | IGNORE_SPRITES);
if(result > 0)
{
run_percentage %= 100;
run_percentage += 15 * time_step;
ent_animate(my, "run", run_percentage, ANM_CYCLE);
}
else{
stand_percentage %= 100;
stand_percentage += 5 * time_step;
ent_animate(my, "stand", stand_percentage, ANM_CYCLE);
}
wait(1);

camera.tilt = -25; // look down at the player, play with this value
camera.x = sid.x-800; // 250 = distance between the player and the camera, play with this value
camera.y = sid.y; // use the same value here
camera.z = sid.z + 450; // place the camera above the player, play with this value

}
}
```

Run the game and now the camera is always following Sid wherever he goes.
If you toy with the camera settings you can make it into any view you want. First person/Top view and more. So know we know how to give a model a camera view. We come along way by taking it all step by step. The game is beginning to look like a real adventure ☺

A question I hear many times is about teleporting. You know the character moves from one place to another without the need of walking the distance. I wonder can I create that ? Well let’s try do this. I made a teleport model and placed 2 of them in the level. So if player touches this model it should teleport to the second teleporter and vice versa.

First I create 2 pointers for the teleporters. Add all of the following script at the bottom of your script.

```
ENTITY* beamer1;
ENTITY* beamer2;
```

Then I create a function for the beamer 1 and an action to assign to this first teleport model.

```
function beam_me1()             <<< Name of the function for teleporter 01.
{
my.event = NULL;                <<< No other event will run
wait (1);                       <<<wait one frame
set (beamer2, PASSABLE);        <<< make beamer 2 passable so it won’t react when we get there
vec_set(sid.x, beamer2.x);      <<< teleport the sid model to beamer 2
my.event = beam_me1;            <<< On touch achtivate the beam_me1 function
}
```

This part should be clear. Now we make an action for the teleport model 01.

```
action portal1()                                    <<< the name of our nr 01 teleporter action
{
beamer1 = my;                                       <<<Use the pointer
while (!sid) {wait (1);}
my.emask |= (ENABLE_IMPACT | ENABLE_ENTITY);        <<<This emask makes the model sensitive for impact and entities.
while (1)                                           <<<The while to create a loop
{
my.pan +=5*time_step;                               <<< Make the model rotate continuesly
if (vec_dist(sid.x, my.x) < 10)                     <<< When sid pointer distance is smaller then 10 to the teleport model
{
set (my, PASSABLE);                                 <<< make the teleport model passable
my.event = NULL;                                    <<< don't allow this beamer to teleport it back
while (vec_dist (my.x, sid.x) < 100) {wait (1);}    <<< When sid pointer is smaller then 100 from teleport model we wait
                                                          one frame}

else                                                <<< If all above is not happenning
{
reset (my, PASSABLE);                               <<<make model not passable again
my.event = beam_me1;                                <<< start beam_me1 event.
}
wait (1);                                           <<<wait one frame
}
}
```

So now for the second model we use a duplicate of the function and action only this one uses beam_me2 for teleporting to beamer 1

```
function beam_me2()
{
my.event = NULL; // don't trigger several events
wait (1);
set (beamer1, PASSABLE); // don't allow beamer1 to teleport the player back to beamer2
vec_set(sid.x, beamer1.x);
my.event = beam_me2;
}
action portal2() // attach this action to the second portal
{
beamer2 = my;
while (!sid) {wait (1);}
my.emask |= (ENABLE_IMPACT | ENABLE_ENTITY);
while (1)
{
if (vec_dist(sid.x, my.x) < 10) // the player was just beamed here?
{
set (my, PASSABLE);
my.event = NULL; // then don't allow this beamer to teleport it back
// wait until the player has moved away from this beamer
while (vec_dist (my.x, sid.x) < 100) {wait (1);}
}
else
{
reset (my, PASSABLE);
my.event = beam_me2;
}
wait (1);
}
}
```

Now give the first teleport model the Portal1 action and the second the Portal2. Run your level and yes I’ve created a real working teleporter. You can make the teleport going slower by using wait(-5) this will wait 5 seconds before it gets to the other side. Also you should know how to add sound as you used sound for the door to ☺ This way you can as many teleport portals as you want cool isn’t it ?

Well this was enough for one day.

Day five.

For today I will try to create a very simple enemy. I will create a bouncing ball that when it hits the player makes it lose a life and gets teleported back to its starting position. So nothing too hard for a start I am keeping this simple.

The first thing I do is make the Sid model be sensitive for impact. When sid is hit by impact a function will be started.

I added 2 lines of script before the while in the player_movement action.

```
my.emask |= (ENABLE_IMPACT);   <<<<the object is sensitive to block and entity collisions
my.event = hit_byball;<<<< if the model is hit start the hit_byball function.
```

Then I create the hit_byball function wich we place on top of the player_movement action :

```
function hit_byball()   <<<< name of the function
{
set(my,PASSABLE | INVISIBLE);  <<< set the model passable and invisible
my.event = NULL; <<< end the models action
wait(1);<<< wait one frame
ent_remove(sid);<<<<< remove the model
lives -=1;<<<<<< substract a life
```

Then we repeat what we already used with the hurt_sid action

```
if(lives ==2){
reset(lives03_pan,SHOW);
}
if(lives ==1){
reset(lives02_pan,SHOW);
}
if(lives ==0){
reset(lives01_pan,SHOW);
}
wait(-1);
if(lives >=1){
ent_create("Sid.mdl", vector(0,0,48), player_movement);


}
else
{

set(gameover_pan,SHOW);
enable_gamestart =0;

}
}
```

So now it’s time to make this enemy that bounces around hitting the sid model.

```
function bounce_off()   <<< name of the function activate when the ball hits something
{
vec_to_angle(my.pan, bounce);  <<<< change the direction when hitting something
my.pan += 5 - random(10); <<<random pan direction
my.tilt = 0; <<<< keeps the ball on the ground
}

action my_enemyball() <<< name of the action for the bouncing enemy
{
my.emask |= (ENABLE_BLOCK | ENABLE_ENTITY); <<<<the object is sensitive to block and entity collisions
my.event = bounce_off; <<<< When ball hits something activate the bounce_off function

while(1) <<<< we create a loop
{
c_move(me, vector(35 * time_step, 0, 0), nullvector, 0); // move ahead, in the direction given by the pan angle
```

make the ball move straight foreward with a speed of 35.

```
wait(1); <<<< wait one frame
}
}
```

Ok now give the ball model the my_enemyball action and run the game.
The ball will bounce of walls and objects. If Sid is hit by impact Sid will lose a life.
So now we have some simple yet effective enemies. Make sure the ball is not in a room with
A teleporter entity as the teleporter reacts on impact to.

In a lot of games they use countdown timers ☺ so I will make one to. If time is left when player
Reached the end of the level the left over time will be added to the score.

Here is how we make a countdown timer. We will place this script on top of all.

```
var countdown_timer = 100;    <<<< timer variablethis timer will run for 100 seconds
FONT* arial_font = "Arial#48b"; <<<< were using the arial font
PANEL* timer_pan = <<< create panel for the timer
{
layer = 15;  <<< the layer remember this ?
digits(900, 700, 4 ,arial_font, 1, countdown_timer); <<<<< position of the panel
flags = SHOW; <<<< make it visible during game
}
function countdown() <<<<< the function that starts the counter
{
while (1)<<<<< create a loop
{
countdown_timer -= time_step / 16; <<< counting down per 1 second
countdown_timer = maxv(countdown_timer, 0); <<< don't allow the timer to go below zero
wait (1);
}
}
```

At game start we make the counter start counting so in our main we add :

```
countdown();
```

When we restart the game we simply set the counter variable back to 100

```
countdown_timer = 100;
```

Run it and you will see the counter running. On restart it is set back on 100. Yay now we can
Earn extra points by finishing the level fast ☺ Pfew enough done for today.

Day six

So what will we do now. I will create a second level and call it level002.wmb
Then we create a model that on touch does this :

Add left over time to the score.
Add one to the stage number.
Load the second level.

Mmmm so we will need a new variable called stage. This is so easy now to do right ?

Place this script with the other variables.

```
var stage = 1;

FONT*	fnt1_pan = "Ariel#22b";
PANEL* pan_stage = {digits=300,50,"Stage: %00.0f",fnt1_pan,1,stage;
layer = 1;
flags = SHOW;green=255; blue=0; red=255;
}
```

I created also a next stage panel that will show on level switch. We define it like we did before.

```
BMAP* next_map = "next.png";

PANEL* next_pan =
{
		bmap = next_map;
		pos_x = 400;
		pos_y = 300;
		layer = 1;
}
```

I created a sprite with the text exit on it. This will be the sprite that if Sid touches it, it will end the level does some scoring ad 1 to stage and loads the next level.

Here is the action for it. Place it at the bottom of your script.

```
action exit_level() <<< name of the action
{
set(my,PASSABLE | SHADOW | METAL | BRIGHT); <<<< make it passable have shadow  metal look and bright.
c_setminmax(my); <<< us box for collision
while(vec_dist(my.x,sid.x) >100) <<<< if sid pointer distance is bigger then 100 from model exit
{
my.pan += 14* time_step;  <<<< rotate pan continously
my.ambient = 200; <<<< give ambient
my.lightrange = 20; << give little light
wait(1); <<< wait one frame
}
ent_remove(sid); <<<< if distance is smaller

stage +=1;  <<<< ad one to stage variable
proc_kill2(countdown,NULL); <<< stop the countdown function
set(next_pan,SHOW); <<<< set the next panel visible
wait(-2); <<< wait 2 seconds
if(stage ==2){<<<< if stage is equal to 2
reset(next_pan,SHOW); <<<<make next panel invisible
score = score + countdown_timer; <<<<< the left over seconds are added to the score
countdown_timer = 100; <<< timer is set on 100 again
ent_remove(me); <<< the model is removed.
level_load("level002.WMB"); <<<< the level 2 is loaded
countdown(); Countdown starts again from 100
}
```

Place the sprite somewhere in your level and give it the exit_level action.

Wammo ! there we go. Once the player hits this sign the scoring is done, the next stage panel message is showing and the new level is loaded.

Important is that the start position in level 2 is different than in level one so we need to add a few lines of script to the function hit_byball and action hurt_sid

```
if(stage ==1){

ent_create("Sid.mdl", vector(0,0,48), player_movement);
ent_create("hurtcoin.mdl", vector(my.x,my.y,my.z),hurt_sid);
ent_remove(me);
}
if(stage ==2)
ent_create("Sid.mdl", vector(1397,-4,377), player_movement);
}
```

All will be working now ☺ You can use a full screen panel for the next level so you won't see the actual level change. You can even make it play a sound or music file before traveling to the next Level.

So what do we have made so far ? quite a lot.

Walking player – Camera view – Bouncing enemy – Timer – Picking up stuff – Teleporting – Lives count in graphics – Using sound – Doors & keys…. It's amazing while I am creating I realize we have basically already made a real game. Sure it needs eye candy and more stuff but we can add it so easy now since we have all the basics working ☺

Enough for today I will continue next time.

DAY seven.

Today I will spend some time at eye candy. The levels background is now just blue so I will give it a nice sky. We will use a sky cube picture for this. By script this picture will fold around our level so wherever you look you will see the blue blue sky. Place the following script on top of your full script.

```
ENTITY* sky =  <<< the skybox picture pointer
{
		type = "skysunset+6.tga";  <<<< the skycube picture it uses it is stored in the graphic folder
		flags2 = SKY | CUBE | SHOW; <<< set as sky,cube and show it.
}
```

Save the script and run it. That was sooooooooo easy now we have a cool blue sky. You can use any skycube picture you want.

Sometimes I amaze myself hahaha. So now for some more nice eye candy. I have this self-made magic animate sprite it is stored in the graphic folder called magic22+8.png

Now it will be cool that whenever Sid is reappearing or disappearing this animated sprite will show.

Here is how we do it. First I Need to create a function that will make the sprite appear play all its frames and then disappear.

Place this code on top of you script :

```
function sprite_played()  <<<< name of the function
{
set (my, PASSABLE |TRANSLUCENT); <<<< make it passable and transparant
my.scale_x = 1; <<< set the scale of the sprite
my.scale_y = my.scale_x; <<< set scale x and y
my.ambient = 100; <<<< give it ambient
my.roll = random(360); <<<< make it roll random from 360 degrees
my.alpha = 100; <<< set alpha (Transparancy on 100 fully visible)
while (my.frame < 8) <<<< when frame is smaller then 8
{
my.frame += 1 * time_step; <<<play all frames
wait (1); <<<<< wait one frame
}
while (my.alpha > 0) <<<< if transparency alpha is bigger then 0
{
my.alpha -= 4 * time_step; <<<<< make alpha smaller with the speed of 4 (gets more transparent.
wait (1);<<<< wait one frame
}
ent_remove (me); <<<<< remove the sprite
}
```

So we made the sprite function let’s see if we can make it show every time Sid appears. Outside the while (on top) in the player movement action we add :

```
ent_create("magic22+8.png", vector(my.x,my.y,my.z), sprite_played);
```

It creates the sprite and uses the sprite_played function just one time ☺ so every time Sid appears it will be used. If you want to use it when Sid disappears you simple add this same line to the scripts where Sid is removed.

So place it in the Hurt_sid coin action and the hit_byball function. Make sure you use the Sid pointer so the sprite is created at this position.

```
ent_create("magic22+8.png", vector(sid.x,sid.y,sid.z), sprite_played);
```

I really enjoy this and while I am creating a game out of nothing I learn a lot of programming along the way. Who says learning is no fun ?

I want more eye candy so next on my to do list is create a particle trail for the bouncing enemy and I want some cool fire at some places to show. The fire is not really needed but hey it looks cool.

For the particle trail put this script on top of your script :

```
function my_p_event(PARTICLE*p)  <<<< name of the particle event
{
p.alpha -= p.skill_a*time_step; <<< alpha using skill a
if(p.alpha <=0)p.lifespan =0; <<<<< when alpha is smaller or equal to 0 the lifespan is 0
if(p.lifespan <25 && p.lifespan >15) <<<<<if lifespa is smaller then 25 but bigger then 15
p.green-=26.5*time_step; <<<< remove the green at speed 26.5

}

function my_particle(PARTICLE* p) <<<<< name of the function
{
p.red = 255; <<<< red color code 255
p.blue =0; <<<<<< blue set on 0
p.green =165; <<<<< green 165 (will be removed in the P_event)
set(p, MOVE | BRIGHT | TRANSLUCENT);<<<<make particle move bright and transparant
p.alpha = 100; <<<< set full visibility
p.size = 10; <<<< set size
p.lifespan = 10; <<<< set particle life span (How long will it be there)
p.gravity = 0.06;<<<<  give it a little gravity
p.skill_a = 2; <<<< set skill at 2
p.event = my_p_event; <<<< activate and use the p_event

}
```

So to attach the particle to the ball we have to add a few lines of script to the my_enemyball action.

First outside the loop we create a vec tor

```
VECTOR vec_temp;
```

Then in the loop above the c_move line we add these lines.

```
vec_set(vec_temp,vector(0,random(5)-2.5,0)); <<<< sets random and uses the vector
vec_rotate(vec_temp,my.pan); <<<< rotates the particle with the emitter pan vector
effect(my_particle,1,my.x,vec_temp); <<<< set the effect and attach it to the model.
```

So if I run the game now the ball should have a nice particle trail. How about that it actually works great !!!

Till now I programmed 577 lines of script to create all of this. Sids adventure is really getting shape.

Last thing for today is to create a nice fire particle just for the fun of it. Let’s see if I can pull that off.

First we create the particle fading function and the fire effect function. place it on top of your script :

```
BMAP* fire_tga = "fire.tga"; <<<< defining the picture used for the particle.


function fade_fire(PARTICLE *p) <<<< function name for fading particles.
{
p.alpha -= 4 * time_step;  <<<<<< 4 sets the fire particle fade out speed
if (p.alpha < 0) <<<<< when alpha is smaller then 0
p.lifespan = 0; <<<<lifespan is set to 0
}



function fire_effect(PARTICLE *p)
{
p->vel_x = 4 - random(8); // the fire particles have a random speed
p->vel_y = 4 - random(8); // on the x adn y axis
p->vel_z = 2 + random(5); // and a positive, random speed on the z axis
p.alpha = 20 + random(30); // initial alpha value
p.bmap = fire_tga; // fire bitmap
p.size = 45; // sets the size of the flame particles
p.flags |= (BRIGHT | MOVE);
p.event = fade_fire; // fire particle event
}
```

Now we create an action that we can give to any model so that model will create a nice fire.

Place it at the bottom of your script:

```
action making_fire() <<<< name of the action
{
set(my,INVISIBLE| PASSABLE); <<< make the model invisible and passable
VECTOR temp; <<< create a vector to create the fire from
while(1) <<<< making a loop
{
effect(fire_effect,1,my.x,temp); <<< set the fire effect to the vector made.
my.pan +=5*time_step;<<<< make the model rotate
wait(1);<<<< wait 1 frame
}
}
```

Assign the making_fire action to any model you want.

So now i run the game and yes i created fire ☺ Nice flames and good eye candy right ?

Well enough is enough for today I learned a lot about particles and animated sprites. You can use this to your own models/sprites en effects.

Day eight.

Good morning I just woke up an hour ago so time to work on the just create a game on the flow. We came a long way and have all the basic stuff for our game working. Time to add some new stuff.

Our player has no gravity. I create a kind of ramp block in the level. It does walk up to it but when it walks of it just hangs in the air and is not falling back to the ground. I want to keep all as simple as possible so me and you are always able to look things back up in this document.

We will give the player some simple gravity coding (Thx to Malice) it will tilt, roll the model when walking up and down slopes and makes sure the model always fall down to the ground. We have to make some modifications to our player_movement action for this.

Outside the loop we place these 3 lines :

```
ANGLE temp_angles; / <<<< create a temp for angles
my.skill20 =0;  <<<<<<<SCAN RESULT FOR GRAVITY
my.skill21=0;  <<<<< GRAVITY SPEED (how fast you get back to the ground.
```

Then in top of the while we place one line of code :

```
my.skill20=c_trace(my.x, vector(my.x,my.y,my.z-500), IGNORE_ME); <<<<c_trace will check gravity minus 500 so it will go down walking of
                                                                   a block that's heigher.
```

The next lines of script we place before the c_movement instruction

```
temp_angles.tilt = 0;   <<<< use temp for tilt model
temp_angles.roll = 0; <<<<< use temp for roll model
temp_angles.pan = -my.pan; <<<< temp angle will be the models pan
vec_rotate(normal, temp_angles);<<<<< rotate all temp angles when needed
temp_angles.tilt = -asin(normal.x); <<<<set tilt
temp_angles.roll = -asin(normal.y);<<<<set roll
my.tilt += 0.1 * ang(temp_angles.tilt - my.tilt); <<<<< play with 0.1 - it sets the angle adjusting speed
my.roll += 0.1 * ang(temp_angles.roll - my.roll); <<<<< play with 0.1 - it sets the angle adjusting speed

if(my.skill20 >60) <<<<< if result gravity is bigger then 60
my.skill21=-50*time_step; <<<<< make it fall down at a speed of 50
else <<<<<if not then
my.skill21=0;<<<<< gravity is 0 the model is on the ground.
```

Then we add one last thing to the c_move line

```
result = c_move(me, nullvector, vector(15 * (key_cuu - key_cud) * time_step, 15 * (key_cul - key_cur) * time_step,my.skill21), GLIDE |
IGNORE_PASSABLE | IGNORE_SPRITES | ENABLE_BLOCK);
if(result > 0)
```

Can you see what it is ? we added my,skill21 so during movement it will check the gravity speed.

If I run my level now I can walk up and down the ramp without any problems ☺ when I walk of a height block I will fall back to the ground. Yes I did it I added some simple gravity.

So next thing I want to learn is the use of elevators where the player stands on a platform then goes up to a certain point, wait and go down again. I found this template elevator code that works however the player won't go down with it unless it jumps on the platform while it's going down.

I'll ask in the forum for a solution but if you place this script at the bottom of you script and assign a model or map entity the elevator action it works. In wed you can use the skills to set height speed and waiting time.

```
#define elevator_height skill1
#define elevator_speed skill2
#define elevator_wait skill3
#define elevator_roaming skill4

function t_elevator_start();
function t_elevator_stop();

function elevator_waits()
{
my.skill70 = 0;
while (my.skill70 < my.elevator_wait)
{
my.skill70 += 50*time_step;
wait (1);
}
my.skill70 = 99;
}

// skill1: elevator_height 200
// skill2: elevator_speed 2
// skill3: elevator_wait 2

action elevator()
{
c_setminmax(me);
var init_z = my.z ;
var waiting_time;
var start_once = 0;
var stop_once = 0;
while (!sid) {wait (1);}
while(sid)
{
if (vec_dist (sid.x, my.x) < 100)
{
if (my.z < my.elevator_height)
{
stop_once = 0;
start_once += 1;
if (start_once == 1)
{

}

sid.z += my.elevator_speed * time_step;
my.z += my.elevator_speed * time_step;
}
else
{
start_once = 0;
stop_once += 1;
if (stop_once == 1)
{

}
waiting_time = 0;
while (waiting_time < my.elevator_wait)
```

```
{
waiting_time += time_step;
wait (1);
}
}
}
else
{
start_once = 0;
while (my.z > init_z)
{
start_once += 1;
if (start_once == 1)
{

}
my.z -= my.elevator_speed * time_step;
if (vec_dist (sid.x, my.x) < 100)
{
sid.z -= my.elevator_speed * time_step;
}
wait (1);
}
stop_once += 1;
if (stop_once == 1)
{

}
my.z = init_z;
}
wait (1);
}
}
```

Take a good look at this script See how we use the pointer SID so it gets activated by it. Gravity will make sure that if we leave the platform in open air we fall down to the ground. If someone gives me a good solution to make it perfect I will get back to this elevator stuff.

For now we did a lot and were almost done with creating a full game.

Day nine

What to do what to do. First I will add a nice tune to the game. Music makes a game always seem to come more to life doesn't it ? I create a music file ingame.wav It is very simple to use during game.

First on top of your script you need to place a variable

```
var soundtrack_handle;
```

Then in the main we place this line that will loop the music file and so play it ☺

```
soundtrack_handle = media_loop("ingame.wav", NULL, 80);
```

There now the game has some nice music with it. It gives it something extra.

I am thinking to give Sid some magical powers where he can blast magic balls that can hurt enemies. Let’s see how should I do this.

So this magic ball when its shot it should go the direction that Sid is facing so I will first create the magic ball action.

```
action magic_ball()  <<< name of the action
{
set(my,PASSABLE); <<< begin passable
my.emask |= (ENABLE_IMPACT | ENABLE_BLOCK | ENABLE_ENTITY);<<< sensitive for al mentioned.
my.event = ball_impact; <<<<< on impact use this event
my.pan = sid.pan; <<< move same pan as sid pointer
while(1)<<<<< create a loop
{
c_move(me, vector(65 * time_step, 0, 0), nullvector, IGNORE_ME); <<<<< move straight
wait(1);<<< wait one frame
if(vec_dist(my.x,sid.x) >40){ <<<< if distance between magic_ball and sid pointer is bigger than 40
reset(my,PASSABLE); <<<< reset passable
}
}
}
```

Now above this action we create the ball_impact function

```
function ball_impact() <<<<<< name of function
{
set(my,PASSABLE|INVISIBLE); <<<< set invisible and passable
my.event = NULL; <<<< kill its action
wait(1);<<<<< wait one frame
ent_remove(me); <<<< remove model
}
```

So now we need a function that say on press ctrl the Sid models shoots a ball. Place it under the action magic_ball script.

In the player movement script we add one simple line :

```
on_ctrl =magic_spell;
```

Yeahaaaaaaa sid can now shoot magic balls every time you hit ctrl ☺

So I learned how to blast/throw and shoot stuff now. That’s handy for all your future projects right ?

Spend enough time today tomorrow is another day.

Day ten

Today I am planning to wrap stuff up. So you can make a game in 10 days just like me ☺

First I will add some eye candy to the bullet impact. I want an animated sprite of an explosion to play when it hits something. So I duplicated our sprite_played code and called it sprite_played2

```
function sprite_played2()
{

set (my, PASSABLE |TRANSLUCENT);
my.scale_x = 1;
my.scale_y = my.scale_x;
my.ambient = 100;
my.roll = random(360);
my.alpha = 100;
while (my.frame < 16)
{
my.frame += 1 * time_step;
wait (1);
}
while (my.alpha > 0)
{
my.alpha -= 4 * time_step;
wait (1);
}

ent_remove (me);
}
```

Basically it's the same script accept I changed one important thing. The sprite I use has 16 frames instead of 8 so I changed the while.

Then in the ball_impact script we add one simple line that creates the explosion sprite that will use the sprite_played2 function.

```
ent_create("Explosion+16.tga", vector(my.x,my.y,my.z), sprite_played2);
```

And it's done I added the IGNORE_SPRITES flag to the magic_ball movement otherwise the balls would be stopped by the explosion sprites. Now on impact the balls explode ☺ easy to add sound to if you want feel free.

To limit the amount of magic balls we create a variable. Oh man this is now so easy:

```
var spells = 10;

FONT*	fnt1_pan = "Ariel#22b";
PANEL* pan_spells = {digits=10,50,"Spells: %00.0f",fnt1_pan,1,spells;
layer = 1;
flags = SHOW;green=255; blue=0; red=255;
}
```

We begin with 10 magic balls ☺ every time a ball is used we subtract one from this variable. If smaller than 1 no more balls to shoot. Simple change of the shoot ball function.

```
function magic_spell()
{
if(spells >=1){

ent_create("ball2.mdl", vector(sid.x,sid.y,sid.z), magic_ball);
spells -=1;
}
}
```

Now when the spell variable is under 1 the player can't fire anymore magic_balls

So what's next ? I know I Know picking up magic potion that will give you extra magic balls ☺
This should not be a problem ☺ Place the script at the bottom of your script

```
action pickup_magic()
{
set(my,PASSABLE | SHADOW | METAL | BRIGHT);
c_setminmax(my);
while(vec_dist(my.x,sid.x) >100)
{
my.roll -= 4* time_step;
my.tilt += 4* time_step;
my.ambient = 200;
my.lightrange = 20;
wait(1);
}
spells +=10;
ent_remove(me);

}
```

Since i understand how to make a character pickup stuff it became so easy to make more stuff to pick up and use by using variables for the items. See if you can make some more yourself.

Ok so it will be cool if the magic balls actually make stuff explode ? so we need an action to assign to any model so when it gets hit by the magic ball it will go kaboomie. Place this script at the bottom of your script.

```
function explode_now() <<< name of the function
{
if(you.skill100 == my.skill100){ <<<< if its hit by an object with the same skill
set(my,PASSABLE|INVISIBLE); <<<< set passable and invisible
my.event = NULL;<<< kill total function
ent_create("Explosion+16.tga", vector(my.x,my.y,my.z), sprite_played2); <<<<create and use sprite animation
wait(-0.1);
ent_create("Explosion+16.tga", vector(my.x,my.y,my.z), sprite_played2);
wait(1);
ent_remove(me); <<< remove model.
}
}
```

```
action exploding_stuff() <<< Name of the explosion action
{
set(my,SHADOW | METAL); <<<set shadow and metal look
my.skill100=100; <<<< skill100
my.emask |= (ENABLE_IMPACT);<<<make sensitive for impact
my.event = explode_now; <<<<on impact use function explode now
}
```

So all left to do is make sure the magic_balls have the same skill so add this line outside the loop.

```
my.skill100 = 100;
```

So the magic balls have the same skills as the explosive objects. Try it you will see you can destroy all models with the same skill added to it.

So in 10 days I made a full game ☺ well the basic game that is. You can use your imagination to add stuff like sounds eye candy more enemies and so on.

Now if I want to wrap all up and share with you I need to solve 2 more things that annoy me.

- The player has to pan right while moving diagonal. (on press cuu and cul for example)
- The elevator should go down to with player on it without the need to jump or leave the platform

Only then when all works perfect it is ready to be used by you all. So I will dig through all the AUMS, forum and resources there are to figure this out. (need to add an extra day then) Also I'll try to contact some 3d game designer friends to see if they can help me out.

Day eleven

So I need to fix that diagonal walking and making sure that the player is facing in the right direction. We have to redo part of the player_movement code and add a new function to it. ***A big thank you to Arrovs from the forum that helped me out on this for me tough problem.***

First I need to use a variable for the key pressing. Place this script on top of the movement action.

```
var is_input_pressed()
{
if(key_cul != 0 || key_cuu != 0 || key_cur != 0 || key_cud != 0)
{
return 1;
}
return 0;
}
```

Then we make a function that will rotate the player always in the good direction

```
//Allows to rotate player

function movement_input(ENTITY* my)
{
if(is_input_pressed())
{
var side_direction = 0;
var up_direction = 0;
if(key_cul){side_direction = 1;}
if(key_cur){side_direction = 2;}
if(key_cuu){up_direction = 1;}
if(key_cud){up_direction = 2;}
//finalizing angles
//horizontal movement
if(up_direction == 0)
{
if(side_direction == 1)
{
my.pan = 90;
}
if(side_direction == 2)
{
my.pan = 270;
}
}
//vertical movement
if(side_direction == 0)
{
if(up_direction == 1)
{
my.pan = 0;
}
if(up_direction == 2)
{
my.pan = 180;
}
}
//diagonal movement
```

```
if(up_direction != 0 && side_direction != 0)
{
if(side_direction == 1 && up_direction == 1)
{
my.pan = 45;
}
if(side_direction == 2 && up_direction == 1)
{
my.pan = 315;
}
if(side_direction == 1 && up_direction == 2)
{
my.pan = 135;
}
if(side_direction == 2 && up_direction == 2)
{
my.pan = 225;
}
}
}
}
```

All left now is to remove the pan settings in player_movement code and call the function we made. So your player_movement code should be like this now :

```
action player_movement()
{
 my.skill100 =200;
 c_setminmax(my);
 set(my,SHADOW);
 sid = me;
 ANGLE temp_angles; // make this a local ANGLE if you use several identical entities
 my.skill20 =0; //SCAN RESULT FOR GRAVITY
 my.skill21=0;  // GRAVITY SPEED
 var run_percentage = 0;
 var stand_percentage = 0;
 my.emask |= (ENABLE_IMPACT); // the object is sensitive to block and entity collisions
 my.event = hit_byball;
 ent_create("magic22+8.png", vector(my.x,my.y,my.z), sprite_played);
 on_ctrl =magic_spell;
 while(1)
 {
		movement_input(my);

  my.skill20=c_trace(my.x, vector(my.x,my.y,my.z-500), IGNORE_ME);



  temp_angles.tilt = 0;
  temp_angles.roll = 0;
  temp_angles.pan = -my.pan;
  vec_rotate(normal, temp_angles);
  temp_angles.tilt = -asin(normal.x);
  temp_angles.roll = -asin(normal.y);
  my.tilt += 0.1 * ang(temp_angles.tilt - my.tilt); // play with 0.1 - it sets the angle adjusting speed
  my.roll += 0.1 * ang(temp_angles.roll - my.roll); // play with 0.1 - it sets the angle adjusting speed

  if(my.skill20 >60) //CHANGED THIS TO 60
  my.skill21=-50*time_step;
  else
  my.skill21=0;
 result = c_move(me, nullvector, vector(15 * (key_cuu - key_cud) * time_step, 15 * (key_cul - key_cur) * time_step,my.skill21), GLIDE |
IGNORE_PASSABLE | IGNORE_SPRITES | ENABLE_BLOCK);
  if(result > 0)
  {
   run_percentage %= 100;
   run_percentage += 15 * time_step;
   ent_animate(my, "run", run_percentage, ANM_CYCLE);
  }
  else{
   stand_percentage %= 100;
   stand_percentage += 5 * time_step;
```

```
   ent_animate(my, "stand", stand_percentage, ANM_CYCLE);
  }
  wait(1);

  camera.tilt = -25;
  camera.x = sid.x-800;
  camera.y = sid.y;
  camera.z = sid.z + 450;

 }
}
```

If I run the level now all is working perfect ☺ ***We all love Arrovs*** now for his good help on this one. See with help of our friends we will get there were we want.

For the final problem I got great help from forum user Dico. Remember that elevator we used ? Well delete that old script we will use Dico's new and better working one.

Place this script at the bottom of your script :

```
function add_force_to(ENTITY* ent)
{
if(ent)
{
if(vec_dist(my.x,ent.x) < my.max_x * 2)
{
c_move(ent, nullvector, vector(0, 0,my.skill2), GLIDE | IGNORE_PASSABLE | IGNORE_SPRITES );
}
}
}



action basic_elevator()
{
var speed_elev = 5;
var hight_elev = 200;
var wait_time_in_second = 5;
set(me,POLYGON);
my.skill1 = 1;
while(1)
{
var my_z = my.z + hight_elev;
while(my.z < my_z)
{
my.skill2 = speed_elev * time_step;
c_move(me, nullvector, vector(0, 0,my.skill2), GLIDE | IGNORE_PASSABLE | IGNORE_SPRITES );
add_force_to(sid);
wait(1);
}
wait(-wait_time_in_second);
my_z = my.z - hight_elev;
while(my.z > my_z)
{
my.skill2 = -speed_elev * time_step;
c_move(me, nullvector, vector(0, 0,my.skill2), GLIDE | IGNORE_PASSABLE | IGNORE_SPRITES );
add_force_to(sid);
wait(1);
}
wait(-wait_time_in_second);
wait(1);
}
}
```

Now since our hero is sensitive for impact we have to prevent the elevator of destroying Sid ☺
So we add one line to the hurt_byball event.

```
if(you.skill1 != 1)
```

This makes sure that the Impact of the elevator will not destroy our hero. Assing this action to the elevator platform and there you go ☺ up and down without any problems. This script can be easy used to create all kinds of moving platforms.

Were at the end of this game and its document. With these basics you already capable in making more games. If you want to receive updates (When I find new better ways or adding stuff) let me know by dropping me a mail.

You will receive then a mail whenever something is updated. Hell you can even drop some ideas. Keep in mind that I am never sure if I can do it or not I’ll just try along the way and in this case we all have fixed.

I hope you enjoyed this journey cause I did for Sure.

See you all later.

René Pol.