

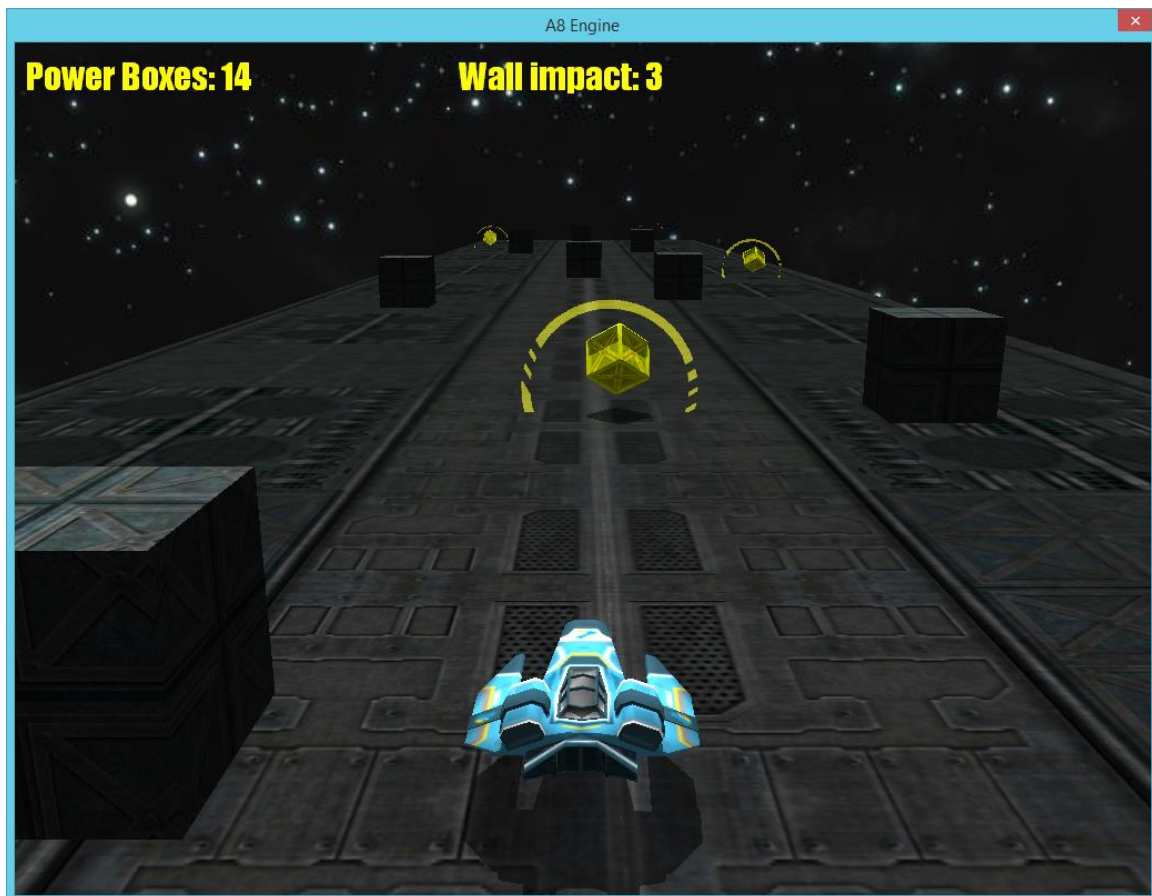
Welcome back happy coders.

I promised you by this time you should be able to create a small simple game. So I decided to create a small game that uses all the stuff you learned in the previous lessons with a little extra code that I will explain. Now from this game you should be able to expand. For example you could decide to make the ship move up and down or later make it fire rockets ☺ it's up to your own creativity. Let me know if you added something to this game or use it as template it's always nice to see what people can do with a little help right ? All new codes are marked in yellow so it is easy to recognize.

Ok so Realspawn what is this little game you cooked up?

At start you fly in with your space ship. You can move it left and right as the movement will make you move forward all the time (Remember our movement lessons) You have to pick up 14 power boxes that are at random places in the level. Blocks are in your way if you run into it you will explode and its game over.

At the end of the level you need to pick up the red box if your miss it, it will be game over and your mission has failed. Your mission is also failed if you did not pick up all the power boxes. So race the track pickup all 14 powerboxes take the red box and be an excellent pilot. Yes you can play the game right away but let's see step by step what I did and how all works shall we. (Alright if you want play it first)



Had fun? Let's see how this game was made and how the programming of it does its job.

The first lines of our script should be familiar for you know. It contains the paths to our subfolders and holds the template scripts that come with 3D gamestudio.

```
////////////////////the standard templates scripts of 3d game studio
#include <acknex.h>
#include <default.c>
#include <mtlFX.c>
////////////////////
#define PRAGMA_PATH "3dmodels" ///<<< we direct the engine to our made work folders
#define PRAGMA_PATH "graphics"
#define PRAGMA_PATH "scripts"
#define PRAGMA_PATH "sounds"
```

The next 2 lines contain a function name and an action name. We had to name these above the main otherwise the engine would not know what to do and you would get an error like: function not defined or action not found. We will find the full action and function later in our code.

```
action ship_movement();///<<< we have to place it here otherwise the engine would not know this action exists
function restart_game(); ///<<< we have to place it here otherwise the engine would not know this function exists
```

We use an in game song during this game and since we need to control this media we need a variable. This explains the next line in our coding right?

```
var soundtrack_handle;///<<< remember how to control audio ? so a variable is needed.
```

Then we see the defining of the 2 sound fx files we use. This so we will be able to make these files play whenever we want during game.

```
SOUND* ammo_snd = "ammo.wav";/// this defines a sound we will use for the pickup
SOUND* explo_snd = "explosion.wav";/// this defines a sound we will use for the explosion
```

Remember what pointers are and how we use them? Well here is the one for our ship again 😊 Every time we need our ship to react or do something it will be called by its name viper.

```
ENTITY* viper;///<<< we make the player model unique and recognizable
```

Next is something new so pay attention. Our hero is in outer space. Instead of putting our level in one big hollow cube we will use a sky cube(space cube) picture. Through our coding it will fold around our level so it looks were in space. Here is how we do it.

```
ENTITY* sky = ///<<< we define the sky entity
{
type = "spacecube1+6.bmp"; ///<<< the picture we will use for the sky
flags2 = SKY | CUBE | SHOW;///<<< the sky picture will show as a cube folding around your level without the need
of putting it in a hollow cube.
}
```

So the engine will use the spacecube 1+6.bmp picture forms a cube with it and shows it. So that's all to create the great outdoors or space. You learned again something new and useful here.

So the goal is to pick up 14 power boxes. This means we need a variable that starts on 14 and subtract every time we pick up a powerbox. When we run into a block we should explode after 3 times so here also we need a variable starting at 3 and subtract 1 every time we have a block impact. Here are the 2 variables made:

```
var powerbox_count = 14; ///<<the name of our variable where we store the powerbox number and we set it on
14 at start. We need to catch 14 powerboxes
FONT* fnt_pan = "Impact#40"; ///<< The true type font we want to use size
PANEL* pan_powerbox_count = {digits=10,10,"Power Boxes: %00.0f",fnt_pan,1,powerbox_count; //the panel
where the number is shown on
layer = 20; //the layer of the panel
flags = SHOW;green=255; blue=0; red=255; // the color of the font that is used. This is yellow.
}
var wallhit_count = 3;///<< the name of our impact variable 3 times impact will be game over
FONT* fnt_pan = "Impact#40"; ///<< The true type font we want to use size
PANEL* pan_wallhit_count = {digits=400,10,"Wall impact: %00.0f",fnt_pan,1,wallhit_count; //the panel
where the number is shown on
layer = 20; //the layer of the panel
flags = SHOW;green=255; blue=0; red=255; // the color of the font that is used. This is yellow.
}
```

I created 2 pictures that show if the mission was failed or succeeded. So we need to define them right? You should know how.

```
BMAP* complete_map = "missioncompleted.pcx"; ///<<< this map will show if the mission was a succes (on pickup
red box end level).
PANEL* complete_pan =
{
bmap = complete_map; //
pos_x = 360;///<<< the position on the x axis
pos_y = 380;///<<< the position on the y axis
}
BMAP* failed_map = "missionfailed.pcx"; ///<<< this map will show if the mission was a failure. (on pickup red box
end level).
PANEL* failed_pan =
{
bmap = failed_map;
pos_x = 360;///<<< the position on the x axis
pos_y = 280;///<<< the position on the y axis
}
```

Next on our list is the main function this one is important. It contains all instructions how the game should start. Camera_clip is new for you. It means that the horizon will show the closer you get. It saves memory if you do not need to see the full level at once. So this way it builds up when your flying to the horizon. Also you see stencil shadow is used. Note this won't work in the free or standard edition of 3D gamestudio A8. Stencil shadow makes a shadow with the same shape of the model itself. We see that the music is started and on pressing the R key we will be able to restart the game (it calls the restart_game function). The model is created and loaded on game start at his start position. Then the resolution is chosen and if the game should be played in full screen or in a window.

```
function main()
{
video_aspect = 1.333; ///<<< enforce 4:3 mode
video_mode = 8; ///<<< resolution
video_screen = 2; ///<<< play in window
soundtrack_handle = media_loop("starcrossed.wav", NULL, 100); ///<<< we use our variable to start and loop the music
level_load("workshop08.WMB"); ///<<< loads the level we made
camera.clip_far = 6000; ///<<< so the horizon will show when we get closer
shadow_stencil = 2; ///<<< will not work in free version of standard
on_r = restart_game; ///<< when pressed r it will start the restart function we made
ent_create("ship.mdl", vector(-2006, 0, 11), ship_movement); ///<<< create the ship at start position and give it the movement action at game start
}
```

You noticed on R we will call the restart function. Here it is. It resets variables to their original start, resets visible panels, stops the music and then makes the main function start all over again so the game will start again.

```
function restart_game() ///<<< our restart function remember we named it also above the main so it will be recognized by the engine
{
media_stop(soundtrack_handle); ///<<< we use again our variable this time to stop the music as it will start in the main again
powerbox_count = 14; ///<<< we reset the powerbox variable to 14 again
reset(failed_pan, SHOW | OVERLAY); ///<<< we hide this panel
reset(complete_pan, SHOW | OVERLAY); ///<<< we hide this panel
wallhit_count = 3; /// we reset the wallhit_count variable to 3 again
main(); ///<<< we go to main so the game fully restarts again
}
```

During an explosion a sprite will be created and it will play all its animation frames. After playing it, it will be removed. Here is the code for the sprite.

```
function sprite_played()///<<<<the function we create for the explosion
{
set (my, PASSABLE | BRIGHT | TRANSLUCENT);///<<<< set the sprite passable bright en transparant
my.scale_x = 1.5; // we scale it down to 0.15 ///<<<<< we scale the original sprite
my.scale_y = my.scale_x; // on both axis ///<<<< we scale the original sprite
my.ambient = 100; // and we give it an ambient of 100 ///<<<< we give it some ambient
my.roll = random(360); // we set a random roll angle ///<<<< the sprite rolls random ways
my.alpha = 100; // but we set it to be completely opaque for now ///<<< we set the transparency
while (my.frame < 16) // go through all the animation frames ///<<<<al sprite frames are 16 frames
{
my.frame += 2 * time_step; // animation speed ///<<< plays the sprite from 1 to 16
wait (1);
}
while (my.alpha > 0) // now fade the last frame quickly
{
my.alpha -= 2 * time_step; // 50 = fading speed ///<<<< make the sprite fades out
wait (1);
}
ent_remove (me); ///<<<< remove the sprite after played full
}
```

Let's see the next lines. It's all about the movement of our ship. Something new is used here. The word clamp. Clamp basically sets boundaries for your model so you can decide how far it can fly to left or right but you can also use it on the other axes like up and down. We use also a camera view you can always change it to some view you like better. If you look close you see that when the ship is over a certain place on the y-axis it shows the mission failed panel. This is because the ship has missed picking up the red box and moves further into space.

```
action ship_movement() ///<<<< the name of the ships action.
{
c_setminmax(me); ///<<<< set box for collision
viper = me; ///<<<< Remember the pointer we made it unique right ?
set(my,SHADOW); ///<<< turns on the shadow
while (1) ///<<<< after this we tell what the ship should do.
{
c_move (my, vector(45 * time_step, 0, 0), nullvector, GLIDE | IGNORE_PASSABLE); ///<<<<the full move instruction
the ship will keep moving forward it can move through passable objects
if (key_cul) // the player has pressed the left cursor key then.....
{
my.y += 35*time_step;///<<<< move to the left on the Y axis
if (my.roll > -20)///<<< roll the ship model til it is bigger then -20
{
my.roll -= 5 * time_step;///<<<< decide how fast the roll should go
```

```

}
}
else ///<<< if left cursor key is no longer pressed
{
if (my.roll < 0) ///<<< roll the ship model til it is smaller then 0
{
my.roll += 5 * time_step; ///<<< decide how fast to roll back
}
}
if (key_cur) // the player has pressed the right cursor key then....
{
my.y -= 25*time_step; ///<<< move to the right on the Y axis
if (my.roll < 20)///<<< roll the ship model til it is bigger smaller then 20
{
my.roll += 5 * time_step;///<<< decide how fast the roll should go
}
}
else///<<< if left cursor key is no longer pressed
{
if (my.roll > 0)///<<< roll the ship model til it is smaller then 0
{
my.roll -= 5 * time_step;///<<< decide how fast the roll should go
}
}
if (my.x > 23700){ ///<<< when our ship is over the distance of 23700 on the x axis remember > = bigger then
set(failed_pan,SHOW | OVERLAY);///<<<show failed panel
}
my.y = clamp(my.y, -400, 400);///<<< this keeps the ship between -400 and 400 on the y axis so it can't fly of the
track here is clamp used to do this
camera.x = my.x-200;///<<< Distance camera x axis from the model. Behind the model
camera.y = my.y; ///<<< We do nothing with the y axis unless you want to
camera.z = my.z + 110; ///<<< Distance height z axis from the model
camera.tilt = -15; ///<<< make the camera look down on the model a bit
wait(1);
}
}

```

I created the ring sprites. It's just for eye candy. The next action makes them simply rotate.

```

action rotator(){ ///<< name of the action we give to our ring sprites so the rotate
set(my,PASSABLE | TRANSLUCENT | BRIGHT); ///<<< makes the sprite passeble so it won't get stuck sets it
transparant and bright
my.ambient = 200;///<<<give it some ambient
while(1)///<<< after this we tell what the ring sprite what it should do.
{
set(my,PASSABLE); ///<<< makes the sprite passeble so it won't get stuck
my.roll += 7* time_step;///<<< rotate continuesly at speed 7
wait(1);
}
}

```

As you learned how to create actions we will need an action for the picking up powerboxes and subtract 1 point from the variable. Here is how we do that.

```
action pickup_bonus() ///<<<<the name of our action that we give to the powerboxes
{
    set(my,SHADOW | PASSABLE | BRIGHT);// set the model passable and give it shadow and make it bright
    wait(1); //wait one frame
    c_setminmax(my);// set proper collision BBOX
    while(!viper) //wait till our player is created the ship_movement action
    {
        wait(1);// wait one frame
    }
    while(vec_dist(my.x,viper.x) >50)// the distance between the viper and the pickup object
    {
        my.roll += 5* time_step;///<<<< rotate continously at speed 5
        my.tilt += 5* time_step;///<<<< tilting continously at speed 5
        wait(1);/// wait one frame
    }
    snd_play (ammo_snd, 100, 0);///<<<< play the sound file we defined when the model is picked up
    ent_remove(my);// we remove the pickup model
    powerbox_count -= 1; ///<<<<subtract 1 point off our variable so eacht time we get one it's registered in our
    variable as found
}
```

When the red box is picked up it should take a look if you have failed or completed the mission. Here is the action we give to the red last pickup box. In both cases the ship is removed but it is our powerbox variable that determines if you failed or completed the mission.

```

action endlevel_bonus() ///<<<the name of our action get the red box to end the level and see if you made the
score
{
set(my,SHADOW | PASSABLE | BRIGHT);// set the model passable and give it shadow and make it bright
wait(1); //wait one frame
c_setminmax(my);// set proper collision BBOX
my.ambient = 200;///<< give it some abient
my.lightrange = 500;///<< make it light up and have a radius of light
while(!viper) //wait till the player is created the ship_movement_action
{
wait(1);// wait one frame
}
while(vec_dist(my.x,viper.x) >50)// the distance between the viper and the pickup object
{
my.roll += 5* time_step;///<<< rotate continuesly at speed 5
my.tilt += 5* time_step;///<<< tilting continuesly at speed 5
wait(1);/// wait one frame
}
snd_play (ammo_snd, 100, 0);///"< play the sound file we defined
ent_remove(my);// we remove the pickup model
if(powerbox_count ==0){///<<< if all boxes are found
set(complete_pan,SHOW | OVERLAY);///<<< show the complete panel
ent_remove(viper);///<<<then we remove our player model
}
else///<<< if not then
if(powerbox_count > 0){ ///<<<if box count is bigger then zero
set(failed_pan,SHOW | OVERLAY);///<<< show failed panel
ent_remove(viper); ///<<< we also remove the player model
}
}
}

```

The final piece of our code handles the impact of the ship to the blocks placed on the level. When hits, it creates our explosion sprite that uses the sprite_played function and it will remove the block or ship when needed. Also it subtracts 1 from our hit wall impact variable. Let's finish this up!

```

function entity_event_impact();///<<<place it here otherwise the engine won't recognise the function
action test_impact()/////<<< name of the action we give the block wall parts
{
my.emask |= ENABLE_IMPACT;///<<< make it sensible for impact
my.event = entity_event_impact; /// <<< if there is impact goto the entity event_impact function
}
function entity_event_impact()///// <<<< Name of the function
{
if (event_type == EVENT_IMPACT)/////<<< if there is impact
{
wallhit_count -=1; ///<< subtract one point from the wallhit_count variable
}
}

```



```

snd_play (explo_snd, 100, 0);////< play the sound file we defined when the model is hit
ent_create("explo2+16.tga", vector(my.x,my.y,my.z), sprite_played);////<<< creates the explosion sprite at block
wall place and give it the sprite play function
ent_remove(me);////<<< remove the exploded wall block
if (wallhit_count <=0){wait(1); ////<<< when our ship is over the distance of 23700 on the x axis remember > =
bigger then
set(failed_pan,SHOW | OVERLAY);////<<<show failed panel cause it did not pick up the red box right ?
ent_remove(viper);/////<<<<remove the player
}
}
}

```

And there it is your little game in 270 lines of code. Use this to test things out changes the variables, movement, use your own sprites and models. This little game is good enough to expand from see if you can make the ship move up and down to and use the clamp command.

This is it for the first 8 workshops. Before I'll upload any new one's some time will pass. Note that I learned while I was writing this so I am just as far in understanding programming as you while your reading this.

I hope you learned something and enjoyed my work so far. If you did mail me or send me a PM at the forum as it's allways good to see and meet the people that are helped by my work.

Till next time

René Pol aka Realspawn

Realspawn@live.nl