Welcome to creating mini games with 3D Game studio A8.
Written by : René Pol 05-08-2017©

Welcome to issue number 8. Since you learned a lot of basics in the first 7 issues we will not discuss menu, buttons and start up screens. If you want to insert It all in this project you can always read back how.

Break out, Arkanoid, Krakout, Ahhh amazing how people can spend hours of breaking bricks with a bat and a ball. In this issue we will do the basic setup for such a game. Is it difficult ? Not at all.

The level is setup Bricks, a bat at the bottom and a ball to break those bricks.

The movement of the bat is very simple just left and right. Remember how to do that ?

```
//////////////////////Movement bat/////////////////////////////////////

action moving_bat()
{
set(my,SHADOW | METAL);
bat = me;
c_setminmax(me);
var my_speed =10;
while(1)
{
c_move(my, vector(0, my_speed * (key_cul - key_cur) * time_step, 0), nullvector, GLIDE | IGNORE_SPRITES | IGNORE_PASSABLE);
camera.x = bat.x -270;
camera.y = 0;
camera.z = bat.z + 250;
camera.tilt =-25;
wait(1);
}
}
```

That was easy. We made a pointer for the bat but also for the ball when it comes in play.

On game start we should see the ball in front of the bat before we can fire it. So we attach a ball model to it that is only shown when the ball is not fired. We create a variable for fire the ball.

```
var fire_ball =0;

STRING* bal_mdl = "bal.mdl";
function adjust_ball()
{
set (my, PASSABLE | BRIGHT);
ball = me;
while (1)
{
vec_set(my.x ,bat.x);
my.x = bat.x + 20;
my.z = you.min_z+15; // place the circle 35 quants above the entity's min_z value, play with 35
wait (1);
if(fire_ball ==1){
set(my,INVISIBLE);
}
if(fire_ball ==0){
reset(my,INVISIBLE);
}
}
}
```

Now all you have to do is call this function and create the ball in the bat_movement action.

```
action moving_bat()
{
set(my,SHADOW | METAL);
bat = me;
c_setminmax(me);
var my_speed =10;
ent_create(bal_mdl, bat.x, adjust_ball);
while(1)
{
c_move(my, vector(0, my_speed * (key_cul - key_cur) * time_step, 0), nullvector, GLIDE | IGNORE_SPRITES | IGNORE_PASSABLE);
camera.x = bat.x -270;
camera.y = 0;
camera.z = bat.z + 250;
camera.tilt =-25;
wait(1);
}
}
```

So now we have the ball and bat in place 😊 left and right the ball moves with the bat and is visible as long as it's not fired.

Let's add a simple score on screen like we did before.

```
////////////////////////////////scoring/////////////////////////////////////////

var score =0;
FONT*      fnt1_pan = "Supercell-Magic#28b";
PANEL* pan_score = {digits=170,50,"Score : %006.0f",fnt1_pan,1,score;
layer = 1;
flags = SHOW;green=255; blue=0; red=255;
}
```

Remember the bouncing enemies ? Well now it's a ball that needs to bounce so we use that basic script for the ball.

```
//////////////////////Bouncing balls//////////////////////////////////////

function ball_collides()
{
vec_to_angle(my.pan,bounce);
my.pan += 10 - random(20);
my.tilt = 0;
}

action bouncing_ball()
{
my.emask |= (ENABLE_BLOCK | ENABLE_ENTITY);
my.event = ball_collides;
my.pan = bat.pan;
my.skill1 = 200;
while(1)
{
c_move(my, vector(10 * time_step, 0, 0),nullvector, NULL | IGNORE_SPRITES | IGNORE_PASSABLE);
wait(1);
}
}
```

I use a skill for the ball so later we can use that for the bricks. It will be like this. A ball with skill1 set to 200 hits a brick. The brick recognises it and will react to it as we scripted.

So we will create a variable that shows the balls the player has during game.

```
var balls =3;

FONT*       fnt1_pan = "Supercell-Magic#28b";
PANEL* pan_balls = {digits=170,100,"Balls : %002.0f",fnt1_pan,1,balls;
layer = 1;
flags = SHOW;green=255; blue=0; red=255;
}
```

Now on key press space bar we will make the ball fire and the fired_ball variable will be set on 1. We create a function that checks how many balls are left and if the player can shoot one.

```
action moving_bat()
{
set(my,SHADOW | METAL);
bat = me;
c_setminmax(me);
var my_speed =10;
ent_create(bal_mdl, bat.x, adjust_ball);
on_space = shoot_ball;
while(1)
{
c_move(my, vector(0, my_speed * (key_cul - key_cur) * time_step, 0), nullvector, GLIDE | IGNORE_SPRITES | IGNORE_PASSABLE);
camera.x = bat.x -270;
camera.y = 0;
camera.z = bat.z + 250;
camera.tilt =-25;
wait(1);
}
}
```

The function that checks and shoots a ball.

```
function shoot_ball()
{
if(fire_ball ==0 && balls > 0){
fire_ball =1;
ent_create("bal.mdl", vector(ball.x,ball.y, 11),bouncing_ball);
wait(1);
}
}
```

So now we can shoot a ball that will bounce of the bat and keeps bouncing around in the level. When the ball leaves the area behind the bat it should disappear and one ball gets subtract from the balls variable. The fire_ball variable is set back to 0 so the ball on the bat will be shown again.

```
action bouncing_ball()
{
my.emask |= (ENABLE_BLOCK | ENABLE_ENTITY);
my.event = ball_collides;
my.pan = bat.pan;
my.skill1 = 200;
while(1)
{
c_move(my, vector(10 * time_step, 0, 0),nullvector, NULL | IGNORE_SPRITES | IGNORE_PASSABLE);
wait(1);
if(my.x < -230){
set(my,INVISIBLE | PASSABLE);
my.event = NULL;
wait(-1);
balls -=1;
ent_remove(me);
fire_ball =0;
break;
}
}
}
```

When the balls hits zero it's game over and a game over panel will appear. As the bat is always in play we make it happen in this action.

```
action moving_bat()
{
set(my,SHADOW | METAL);
bat = me;
c_setminmax(me);
var my_speed =10;
ent_create(bal_mdl, bat.x, adjust_ball);
on_space = shoot_ball;
while(1)
{
if(balls ==0){
fire_ball =1;
set(gameover_pan,SHOW);
}
c_move(my, vector(0, my_speed * (key_cul - key_cur) * time_step, 0), nullvector, GLIDE | IGNORE_SPRITES | IGNORE_PASSABLE);
camera.x = bat.x -270;
camera.y = 0;
camera.z = bat.z + 250;
camera.tilt =-25;
wait(1);
}
}
```

The game is getting shape. I created a new variable that checks how many bricks are in play.

```
var bricks_inplay;
```

We need this so when all bricks are destroyed the game knows we go to the next level.

Time to give our bricks a simple action.

```
//////////////////////////////////////////////brick actions//////////////////////////////////////////////////////

function brick_hit()
{
if(you.skill1 ==200){
my.skill1 =0;
}
}

action brick_one()
{
bricks_inplay +=1;
set(my,BRIGHT | SHADOW);
my.skill1 =100;
c_setminmax(me);
my.emask |= (ENABLE_ENTITY | ENABLE_IMPACT);
my.event = brick_hit;
while(1)
{
if(my.skill1 ==0){
set(my,PASSABLE |INVISIBLE);
my.event= NULL;
wait(1);
ent_remove(me);
bricks_inplay -=1;
score +=50;
break;
}
wait(1);
}
}
```

Every time the brick gets hit its skill turns into 0 and the brick will be removed and score is added. At start each brick adds 1 to the bricks_inplay variable when this variable hits 0 (Each time a brick is removed) its time to move on to a next level.

I added a new variable to show the level.

```
//////////////////////////////////////level variable//////////////////////////////
var level =1;
FONT*       fnt1_pan = "Supercell-Magic#28b";
PANEL* pan_level = {digits=700,100,"Level : %002.0f",fnt1_pan,1,level;
layer = 1;
flags = SHOW;green=255; blue=0; red=255;
}
```

We create a function that checks the bricks in play. If it's zero the ball will disappear and the level up panel will appear. 1 will be added to the level variable.

Here is the check_bricks function.

```
////////////////////////////////////check_bricksinplay//////////////////////
function check_bricks()
{
if(bricks_inplay ==0){
set(levelup_pan,SHOW);
}
}
```

To make the ball disappear when a level is done we add some lines to the ball action.

```
action bouncing_ball()
{
my.emask |= (ENABLE_BLOCK | ENABLE_ENTITY);
my.event = ball_collides;
my.pan = bat.pan;
my.skill1 = 200;
while(1)
{
c_move(my, vector(10 * time_step, 0, 0),nullvector, NULL | IGNORE_SPRITES | IGNORE_PASSABLE);
wait(1);
if(bricks_inplay ==0){
set(my,INVISIBLE | PASSABLE);
my.event = NULL;
wait(-1);
fire_ball = 0;
ent_remove(me);
break;
}
if(my.x < -230){
set(my,INVISIBLE | PASSABLE);
my.event = NULL;
wait(-1);
balls -=1;
ent_remove(me);
fire_ball =0;
break;
}
}
}
```

Wohooooo this game is almost done 😊 So when all bricks are removed the level goes up. I created a new level and called it Level002.

All left to do is to lead the next level so we can play one. We can do that in the check_bricks function.

```
function check_bricks()
{
if(bricks_inplay ==0){
level +=1;
set(levelup_pan,SHOW);
wait(-2);
if(level ==2){
level_load("level002.WMB");
reset(levelup_pan,SHOW);
}
}
}
```

This way you can create as many levels if you want.

We have bricks that get destroyed after one hit. It would be easy to make bricks that need to be hit 2 or more times. Here is how.

```
function brick_hit2()
{
if(you.skill1 ==200){
my.skill1 -=100;
}
}

action brick_two()
{
bricks_inplay +=1;
set(my,BRIGHT | SHADOW);
my.skill1 =200;
c_setminmax(me);
my.emask |= (ENABLE_ENTITY | ENABLE_IMPACT);
my.event = brick_hit2;
while(1)
{
if(my.skill1 ==0){
set(my,PASSABLE |INVISIBLE);
my.event= NULL;
bricks_inplay -=1;
wait(1);
ent_remove(me);
check_bricks();
score +=100;
break;
}
wait(1);
}
}
```

Simple right you can create different bricks this way.

I added some sound fx to the game. If you do not understand the working of sounds I suggest you read the first 7 issues.

Now to finish this game we will add an explosion visual. The explosion is in fact a sprite containing more pictures that play after each other. This way it works like a animation. Best thing is you can use this script in all your game projects once you know how to use it.

Here is the script that plays the explosion sprite.

```
function sprite_played()
{
set (my, PASSABLE |TRANSLUCENT);
my.scale_x = 0.2;
my.scale_y = my.scale_x;
my.ambient = 100;
my.roll = random(360);
my.alpha = 100;
while (my.frame < 8)
{
my.frame += 1 * time_step;
wait (1);
}
while (my.alpha > 0)
{
my.alpha -= 8 * time_step;
wait (1);
}
ent_remove (me);
}
```

All you have to do is to create the sprite where you want and give it the sprite_played function. Here is how it's done when the ball is removed.

```
if(my.x < -230){
snd_play(ballout_snd,100,0);
set(my,INVISIBLE | PASSABLE);
my.event = NULL;
ent_create("explosion+16.tga", vector(my.x,my.y,my.z), sprite_played);
wait(-1);
balls -=1;
ent_remove(me);
fire_ball =0;
break;
}
```

You can use it where and when ever you want. I also used It when a brick is completely destroyed.

From here it's up to you. Create new levels, make a menu use buttons and a mouse cursor if needed. The basic break out game is here and ready to be used. You can even insert the pause game function if you want remember that one ?

Happy game creations !

Rene Pol aka Realspawn.