Scripting the Lighthouse (scripting light, compund objects and simple animation)

Hi !

For my new map I needed a lighthouse. It is a nightmap, and I thought it important that the light-beacon really enlights the scene, throws a flare and rotates. So here some basics of my model first.

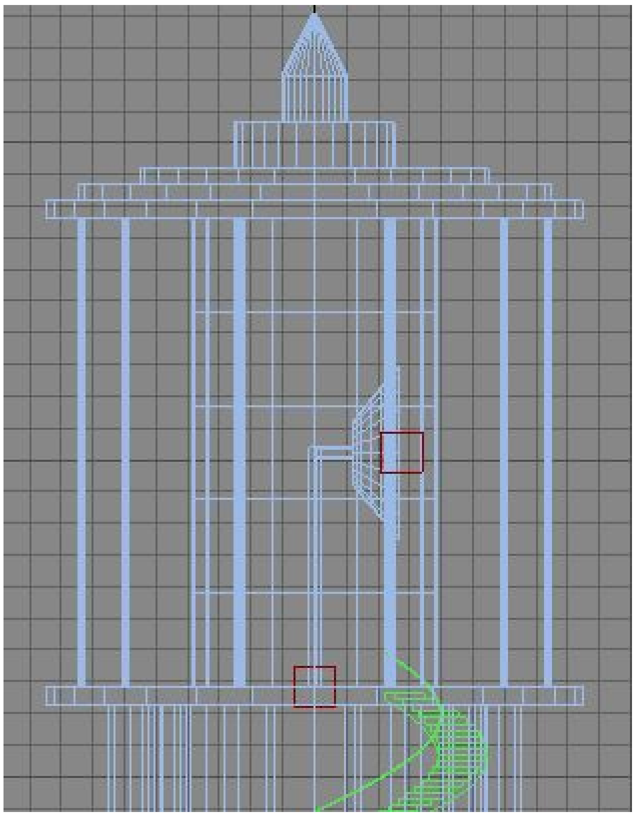
Please understand that I do not upload my models yet, this tutorial shall only cover some general scripting topics. Just make models yourself...

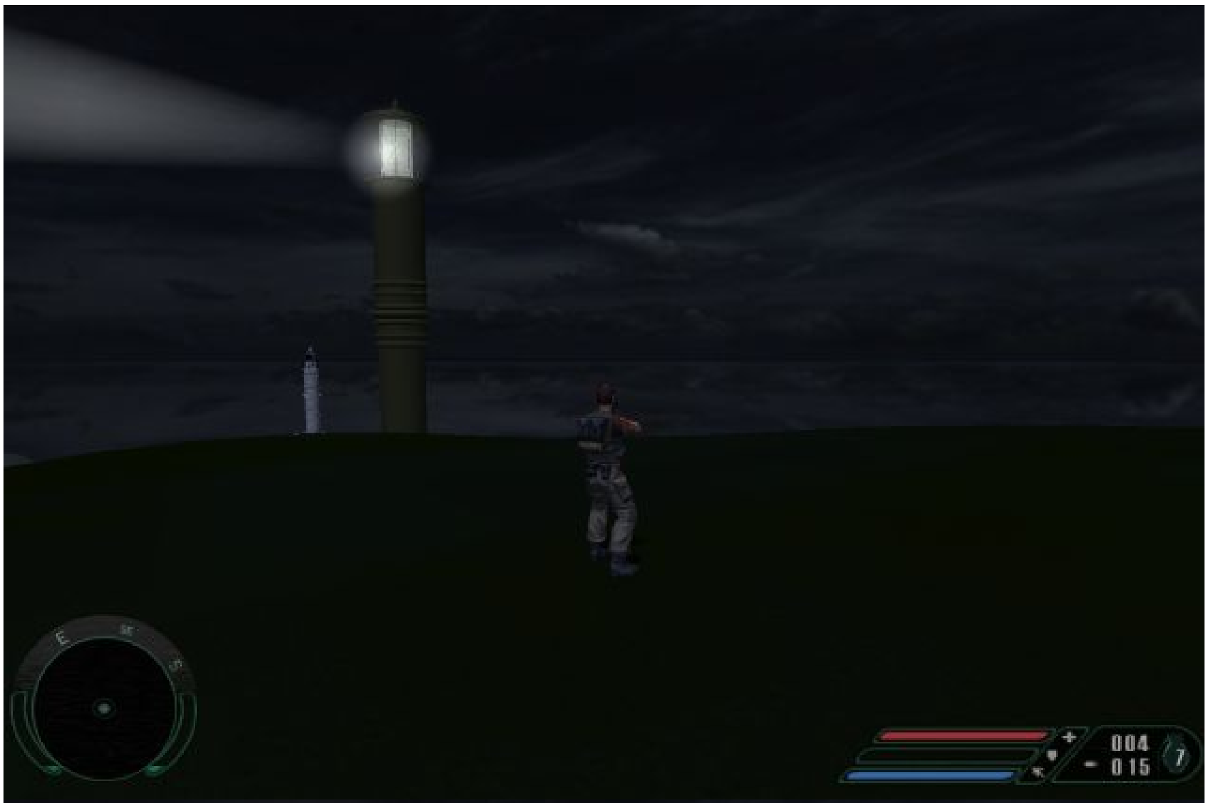
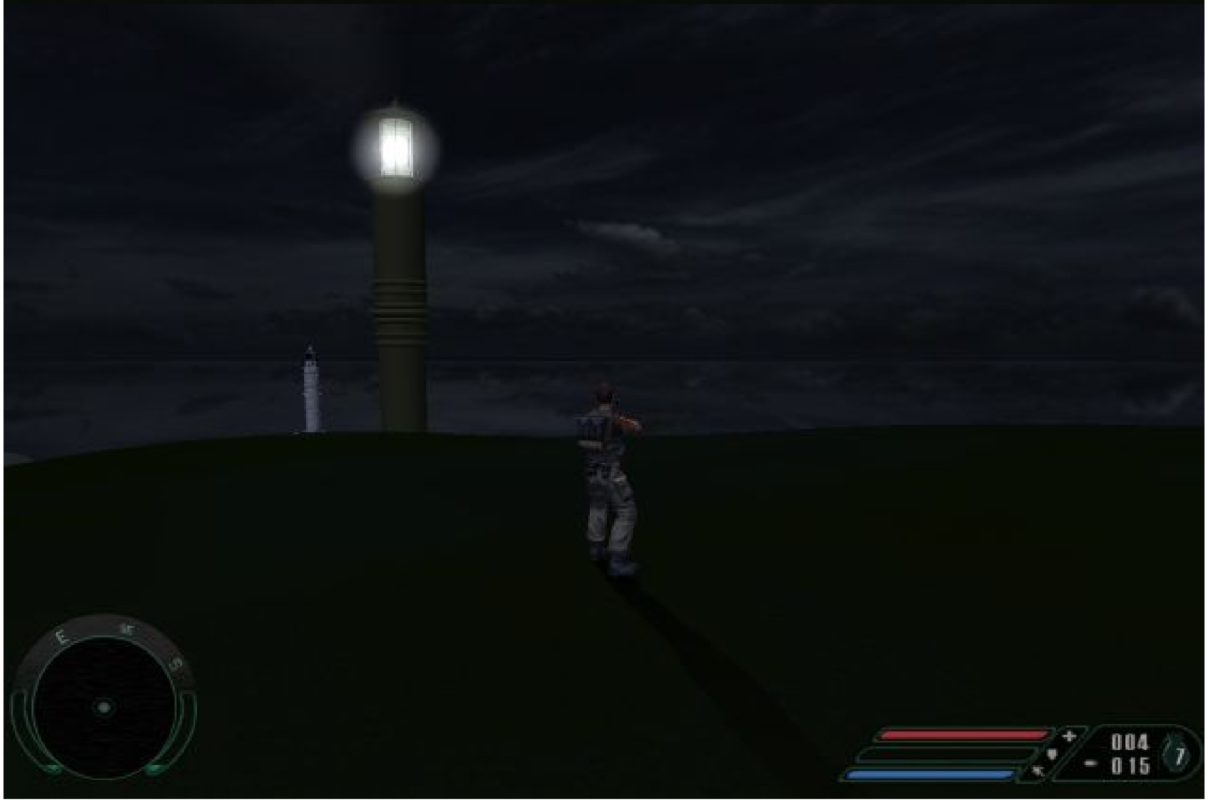
The model "lighthouse.cgf" is the building itself. The model has two helpers (3dsmax dummy objects) named "helper_light" and "helper_lamp". It is important to place them correctly and to export them into the cgf.

The model "lamp.cgf" is a lamp cone (attached to a cylinder). This lamp will turn.

helper_lamp is placed at the position of the lamp (the pivot of the lamp shall be placed here later on)
helper_light is the initial position of the light source.

Here some pics to show what I mean. In the first you see the lighthouse-top with the lamp inside (it is inside a glass cylinder). The box on the bottom of the lamp is "helper_lamp", the box to the right is "helper_light". The house and helpers get exported to "lighthouse.cgf", the lamp to "lamp.cgf".




E
SE
S
004
015
7

E
SE
S
004
015
7

E
SE
S
004
015
7

E
SE
S
005
020
7

The Scripting.

First of all add a "lighthouse" entity to classreg.lua. I won't go into the details of this, because others have done tutorials about editing classreg.

code:

```
	{"",		"Lighthouse",	254, "Lighthouse.lua"},
```

Now let's create farcry/scripts/default/entities/lighthouse.lua. At first the Properties (not all are in use yet, and many of them are copy&paste of dynamiclight.lua and automaticelevator.lua).

ATTENTION: Please take a look at the light shader property. It is "LightBeamTower". This is our own light shader, which we have to configure later on.

code:

```
Lighthouse = {
	Properties = {
		fileModel = "Objects/lighthouse.cgf",
		fileLamp = "Objects/lamp.cgf",
		bRotate = 1,

		WarnLight=
		{
			bHasWarnLight = 1,
			fLightRadius=10000,
			clrLightDiffuse={254, 255, 255},
			clrLightSpecular={255, 255, 255},
			LightAngles = {
				x=0,
				y=0,
				z=0,
			},
			fLightRotSpeed = 50,

			bProjectInAllDirs=0,
			ProjectorFov=90,
			texture_ProjectorTexture="Textures/projector.jpg",
			shader_lightShader="LightBeamTower",
			bAffectsThisAreaOnly=0,
			bUsedInRealTime=1,
			bFakeLight=0,
			bHeatSource=0,
			bIgnoreTerrain=0,
			Optimization=0,
			bDot3Type=1,
			bFakeRadiosity=0,
			vector_LightDir={x=0,y=90,z=0},
			bFixedLightDir={x=0,y=90,z=0},
			bOcclusion=1.0

		},
	},
}

```

Now the functions: OnInit is the constructor. We call the init-Fn OnReset and then set EnableUpdate(1), for our rotating lamp has to be updated regularly.

code:

```
function Lighthouse:OnInit()
        self:OnReset();
        self:EnableUpdate(1);
end
```

OnReset is very interesting: Here we put parts together. We use "LoadObject" to load a cgf, DrawObject to display it. SetPos will position the cgf. To get the correct position, we use our helpers: GetHelperPos returns the helper position. Attention on Parameter 2. This is a boolean. GetHelperPos("helper", 0) gives world-Coordinate, GetHelperPos("helper", 1) gives reltive coordinates within the CGF (what we need). Then we load a Projector-Texture (the one from the elevator) and init the light (this will not show it, just initialize!)

Please note that you give your CGFs an integer object-ID when using LoadObject. This ID will be used as a reference in later calls.

code:

```
function Lighthouse:OnReset()
        self:LoadObject( self.Properties.fileModel, 0, 0 );
        self:CreateStaticEntity(10,-2);
        self:DrawObject( 0, 1 );

        local helperPos=self:GetHelperPos("helper_lamp", 1);
        self:LoadObject( self.Properties.fileLamp, 1, 0);
        self:SetObjectPos( 1 , helperPos );
        self:DrawObject( 1, 1 );

        self.proj_tex_id =
System:LoadTexture(self.Properties.WarnLight.texture_ProjectorTexture,1);

        self:InitDynamicLight(self.Properties.WarnLight.texture_ProjectorT
exture,
                self.Properties.WarnLight.shader_lightShader,
                self.Properties.WarnLight.bProjectInAllDirs,0);

end
```

Finally we have to make parts rotate. We use some simple trigonometry to position the light. OnUpdate gets a time-Value as parameter (time since last update). From that we calculate a rotation angle. We can directly rotate the lamp. The light has to be positioned: At first getting th radius (planar distance from helper_light to helper_land). We ask Pythagoras for that.

Then we rotate the light around the LAMP Helper (the center of the tower), shift the calculated radius, but use the height of the light-helper. We use cos() and sin() to move the light accordingly:

code:

```
function Lighthouse:OnUpdate(dt)
        Pos={x=0, y=0, z=0};
        local lampPos=new(self:GetHelperPos("helper_lamp", 0));
-- new is necessary !!
        local lightPos = new(self:GetHelperPos("helper_light", 0));
-- new is necessary !!
        local rx = lightPos.x - lampPos.x; -- x-distance
        local ry = lightPos.y - lampPos.y; -- y-distance
        local radius = sqrt(rx*rx + ry*ry); -- Pythagoras
        Pos.x = lampPos.x; -- Lamp turns around lamp x,y, but light z
        Pos.y = lampPos.y;
        Pos.z = lightPos.z;

        -- calculate turn angle from dt (see fan.lua)
        self.Properties.WarnLight.LightAngles.z =
self.Properties.WarnLight.LightAngles.z +
self.Properties.WarnLight.fLightRotSpeed * dt;

        if (self.Properties.WarnLight.LightAngles.z>=360) then
                self.Properties.WarnLight.LightAngles.z =
self.Properties.WarnLight.LightAngles.z-360;
        end
        if (self.Properties.WarnLight.LightAngles.z<0) then
                self.Properties.WarnLight.LightAngles.z =
self.Properties.WarnLight.LightAngles.z+360;
        end

        -- now shift the position of the light source.
        Pos.x = Pos.x +
radius*cos(rad(self.Properties.WarnLight.LightAngles.z));
        Pos.y = Pos.y +
radius*sin(rad(self.Properties.WarnLight.LightAngles.z));

        -- now render the light ! This is one huge function call.

        self:AddDynamicLight(   Pos,
                        self.Properties.WarnLight.fLightRadius,
                        self.Properties.WarnLight.clrLightDiffuse[1]*100,
                        self.Properties.WarnLight.clrLightDiffuse[2]*100,
                        self.Properties.WarnLight.clrLightDiffuse[3]*100,
                        1,
                        self.Properties.WarnLight.clrLightSpecular[1]*100,
                        self.Properties.WarnLight.clrLightSpecular[2]*100,
                        self.Properties.WarnLight.clrLightSpecular[3]*100,
                        0,
                        0,
                        0, -- not used
                        self.Properties.WarnLight.LightAngles,
                        self.Properties.WarnLight.ProjectorFov,
                        self.proj_tex_id,
                        self.Properties.WarnLight.bAffectsThisAreaOnly,
                        self.Properties.WarnLight.bUsedInRealTime,
                        self.Properties.WarnLight.bHeatSource,
                        self.Properties.WarnLight.bFakeLight,
                        self.Properties.bIgnoreTerrain,
                        self.Properties.Optimization,
                        self.Properties.bDot3Type,
                        self.Properties.bFakeRadiosity,
                        self.Properties.vector_LightDir,
                        self.Properties.bFixedLightDir,
                        self.Properties.bOcclusion
        );

-- And now turn the lamp. This is simple !! 1 is the Object-ID for our
lamp.cgf
        self:SetObjectAngles(1, self.Properties.WarnLight.LightAngles);
end
```

Now we need an own light-shader for really decent looking light. Go to FCDATA\shaders.pak and take a close look on "lights.csl" (a text file).

Add our shader "LighBeamTower". It will have two effects: One really big, transparent light beam and a bright lens glow. Tweak parameters as liked.

code:

```
Shader 'LightBeamTower'
(
  Params
  {
    Sort = SeeThrough
  }
  ClientEffect 'Corona'
  {
    RGBStyle = FromLight
    Scale = 0.9
    DistFactor=0.5
    Map =lens/lens2
  }
  ClientEffect 'Beam'
  {
    Model = objects/indoor/lights/beam.cgf
    LightStyle = 0
        StartRadius = 5
        EndRadius = 60
        Length = 500
        StartColor (1 1 1 0.2f)
        EndColor (1 1 1 -0.1)
  }
  Layer '0'
  (
    Map = $None
  )
)


```

Done. This thing is neat (I really love it). It dynamically enlights the scene (and the light-cone reflects in the water). If you have any questions or suggestions, please ask.

Regards.