The Optimized Interactive Objects for CryEngine 1 ®

Author: Romochka
Keywords: Interactive Objects, physical reacting vegetation

Index

Prologue:

Probably to create the correct interactive objects possessing physics of a corpse. But such objects strongly influence game performance. There is other variant suitable for the adaptation of many interactive objects on maps. Even simultaneous activation of hundred such objects will not affect productivity of game. It is necessary to create the script of object and its model with a set of animations.

- Romochka

Step 1:

It is necessary to register object in a script "Scripts**ClassRegistry.lua**"

code:
```
{"","InteractiveObject", 209,"Others/InteractiveObject.lua"}, -- add line
```

Step 2:

To create a text file "**InteractiveObject.lua**" in a directory "**Others**"

Also registering in it functions:

code:
```
InteractiveObject = {}
function InteractiveObject:OnInit()
end
function InteractiveObject:OnReset()
end
function InteractiveObject:OnPropertyChange()
end
function InteractiveObject:OnEnter(sender)
end
function InteractiveObject:OnLeave(sender)
end
InteractiveObject.Empty = {}
InteractiveObject.Occupied = {}
InteractiveObject.Shooted = {}
InteractiveObject.Return = {}
function InteractiveObject:StartAnimationDEFAULT()
end
function InteractiveObject:StartAnimationTOUCH()
end
function InteractiveObject:StartAnimationSHOOT()
end
function InteractiveObject:OnSave(stm)
end
function InteractiveObject:OnLoad(stm)
end
function InteractiveObject:OnShutDown()
end
```

Step 3:

To register properties of object which can be changed in the Sandbox:

code:

```
InteractiveObject = {
type = "Trigger",
Properties = {
DimX = 1,
DimY = 1,
DimZ = 3,
object_Model = "",
AnimationDEFAULT = "",
AnimationTOUCH = "",
AnimationSHOOT = "",
Speed=1,
},
}
```

Step 4:

To register in function "**OnInit**" and "**OnReset**" all variables:

code:

```
function InteractiveObject:OnInit()
self.bLoop=1;
self.bPlaying=1;
self:TrackColliders(1);
self:OnPropertyChange();
self.vEntered=0;
self.vShoot=1;
self:RegisterState("Empty" );
self:RegisterState("Occupied" );
self:RegisterState("Shooted" );
self:RegisterState("Return" );
self:OnReset();
end
```

code:

```
function InteractiveObject:OnReset()
local Min = { x=-self.Properties.DimX/2, y=-self.Properties.DimY/2, z=-self.Properties.DimZ/2 };
local Max = { x=self.Properties.DimX/2, y=self.Properties.DimY/2, z=self.Properties.DimZ/2 };
self:SetBBox( Min, Max );
self.vEntered=0;
self.vShoot=1;
self:StartAnimationDEFAULT();
self:CreateStaticEntity( 100, 0 );
self: PhysicalizeCharacter( 100,0,0,0 );
self:GotoState("Empty" );
end
```

Step 5:

To integrate preset values of parameters:

code:

```
function InteractiveObject:OnPropertyChange()
if (self.Properties.object_Model == "" ) then
do return end;
end
if (self.ModelName ~= self.Properties.object_Model) then
self.ModelName = self.Properties.object_Model;
if (self:LoadCharacter(self.ModelName,0)) then
self.bCharacter = 1;
self: DrawCharacter(0,1);
else
self:LoadObject( self.ModelName,0,1 );
self.bCharacter = 0;
self: DrawObject(0,1);
end
end
if (self.bPlaying ~= self.bAnimPlaying or self.bLoop ~= self.bAnimLoop or
self.Properties.AnimationDEFAULT ~= self.AnimName) then
self.bAnimPlaying = self.bPlaying;
self.bAnimLoop = self.bLoop;
self.AnimName = self.Properties.AnimationDEFAULT;
if (self.bPlaying == 1) then
self:StartAnimation(0,self.Properties.AnimationDEFAULT,0,0,1,self.bLoop);
else
self:ResetAnimation(0);
end
end
self:SetAnimationSpeed(self.Properties.Speed)p
```

Step 6

To register two conditions of object:

code:

```
function InteractiveObject:OnEnter(sender)
if ((self.vEntered ~= 0)) then
return
end
self.vEntered=1;
BroadcastEvent(self,"Enter" );
end

function InteractiveObject:OnLeave(sender)
if (self.vEntered == 0) then
return
end
self.vEntered=0;
BroadcastEvent(self,"Leave" );
end
```

Step 7

To register all subconditions (states) of object:

code:

```
InteractiveObject.Empty = {
OnBeginState = function(self)
self.vEntered=0;
end,
OnEndState = function(self)
end,
OnEnterArea = function(self,entity,areaId)
self:StartAnimationTOUCH();
self:GotoState("Occupied" );
end,
OnDamage = function(self,hit)
if(self.vShoot == 1) then
self:StartAnimationSHOOT();
self.vShoot=0;
self:GotoState("Shooted" );
end
end,
}
```

code:

```
InteractiveObject.Occupied = {
OnBeginState = function(self)
self:OnEnter(self);
end,
OnEndState = function(self)
end,
OnLeaveArea = function(self,entity,areaId)
self:SetTimer(4000/self.Properties.Speed);
end,
OnDamage = function(self,hit)
if(self.vShoot == 1) then
self:StartAnimationSHOOT();
self.vShoot=0;
self:GotoState("Shooted" );
end
end,
OnTimer = function(self)
self:OnLeave(self);
self:GotoState("Empty" );
self:StartAnimationDEFAULT();
end,
}
```

code:

```
InteractiveObject.Shooted = {
OnBeginState = function(self)
self:SetTimer(500);
end,
OnEndState = function(self)
end,
OnEnterArea = function(self,entity,areaId)
self:StartAnimationTOUCH();
self:GotoState("Occupied" );
end,
OnTimer = function(self)
self.vShoot=1;
self:GotoState("Return" );
end,
}
```

code:

```
InteractiveObject.Return = {
OnBeginState = function(self)
self:SetTimer(3500/self.Properties.Speed);
end,
OnEndState = function(self)
end,
OnEnterArea = function(self,entity,areaId)
self:StartAnimationTOUCH();
self:GotoState("Occupied" );
end,
OnDamage = function(self,hit)
if(self.vShoot == 1) then
self:StartAnimationSHOOT();
self.vShoot=0;
self:GotoState("Shooted" );
end
end,
OnTimer = function(self)
self:GotoState("Empty" );
self:StartAnimationDEFAULT();
end,
}
```

Step 8:

Functions starting animations:

code:

```
function InteractiveObject:StartAnimationDEFAULT()
self:ResetAnimation(0);
self:StartAnimation(0,self.Properties.AnimationDEFAULT,0,0,1,self.bLoop);
self:SetAnimationSpeed(self.Properties.Speed);
end

function InteractiveObject:StartAnimationTOUCH()
self:ResetAnimation(0);
self:StartAnimation(0,self.Properties.AnimationTOUCH)
self:SetAnimationSpeed(self.Properties.Speed);
end

function InteractiveObject:StartAnimationSHOOT()
self:ResetAnimation(0);
self:StartAnimation(0,self.Properties.AnimationSHOOT)
self:SetAnimationSpeed(self.Properties.Speed);
end
```

Step 9:

Other functions:

code:

```
function InteractiveObject:OnSave(stm)
end

function InteractiveObject:OnLoad(stm)
end

function InteractiveObject:OnShutDown()
end
```

# EDIT #

Delays functions

AnimPlant.Occupied.OnLeaveArea.self:SetTimer (count/speed)

count - time through which animation "**TOUCH**" comes to the end and begins animation "**DEFAULT**"
speed - factor of speed for change of time of reproduction of animation (it is necessary that animation has had time to end before to be replaced with another)

AnimPlant.Shooted.OnBeginState.self:SetTimer (count)

count - time on which function is blocked (to avoid strangenesses in visualization)

AnimPlant.Return.OnBeginState.self:SetTimer (count/speed)

count - time through which works state " AnimPlant. Empty " (that at repeated activations of other functions animation has not got off, which else can play)
speed - factor of speed for change of time of reproduction of animation (it is necessary that animation has had time to end before to be replaced with another)

...
end script

Step 10:

To create model and three animations for it:
_default - plays at absence of influences
_shoot - simulates shooting
_touch - simulates a touch

To convert animations in a ".cga" format.

Step 11:

Naming model animations...

plant_howea.cga -- model name
plant_howea_wind.anm -- animation name

...and use in SandBox:

plant_howea.cga -- model name
howea_wind.anm -- animation name

Feature: In the editor the part of a animation name up to a sign "_" is not used.

- Download **THIS** video - 1.2 Mb

Happy modding!