SmartDM Guide for developers

The_Cowboy

July 15, 2013

In this article I will describe the basic structure of mod SmartDM and the ability of SmartDM to support custom Scoreboards. This is one of the classic examples of Object Oriented Programming where one can avoid adding unnecessary dependencies amongst packages.

SmartDM consists of two packages

- SmartDM.u (I will refer this as core file)
- SmartDMScoreboard.u (most probably, you will be modifying this package)

and two corresponding .ini files

- SmartDM.ini (for server-side settings)
- SmartDMScoreboard.ini (for client-side settings).

1 Working

The package SmartDM.u contains classes which calculate various statistics of players (on server), do keybindings, modify serverinfo, various in-game messages and replicationinfo of the game and players. ReplicationInfo classes are responsible for transferring the information related to game (what gametype is being played, tickrate and other stuff) and players (player statistics, netspeed etc).

The main class is SmartDM. It initializes the various server variables for the mod. In function

```
function PreBeginPlay()
{
    local Mutator M;
    local class<scoreboard> SmartScoreBoard;

    super.PreBeginPlay();

    Log( "Searching for Smart Scoreboard..." , 'SmartDM' );
    SmartScoreBoard = class<scoreboard>( DynamicLoadObject(
    ScoreBoardType ,class'Class' ) );

    SDMGame.NormalScoreBoardClass = Level.Game.ScoreBoardType;
    if ( SmartScoreBoard != none )
    {
       Log( "SUCCESS - A Smart Scoreboard is found" , 'SmartDM' );
       Log( "SmartDM "$Version$" will use "$SmartScoreboard$" as its Scoreboard" , 'SmartDM' );
       Level.Game.ScoreBoardType = SmartScoreBoard;//class'SmartDMScoreBoard';
       SDMGame.SmartScoreBoardClass = SmartScoreBoard;
    }
    else
    {
       Log( "WANRING!! Smart ScoreBoard not found ", 'SmartDM' );
       Log( "WANRING!! Using epic's default Scoreboard ", 'SmartDM' );
    }

    Log( "Original Scoreboard determined as" @ SDMGame.NormalScoreBoardClass , 'SmartDM' );

 ...
```

Line 9 dynamically loads a class with name ScoreBoardType (configured in SmartDM.ini which points to a SmartScoreboard in package SmartDMScoreboard). Since the function searches the mentioned class in loaded packages, no dependency is required. Once loaded, this object is assigned to Level.Game.ScoreBoardType.

Congratulations! We have changed the default scoreboard to our favourite smartscoreboard. But we want both the scoreboards to be displayed (via keybind) so better store the default scoreboard to some variable. Line 11 does the job. The variable SDMGame.NormalScoreBoardClass is, then, replicated to client through SmartDMGameReplicationInfo.

Now let us see how SmartDM opens the uwindow (present in other package) on the client machine without creating dependency to that package. In SmartDM there is a function which intercepts the mutate commands sent to server [1]

```
function Mutate( string MutateString, PlayerPawn Sender )
{
...
}
```

When it sees that some player wants to open SmartDM's client window, it calls the function

```
function OpenMenuWindow(PlayerPawn Sender)
{
   local SmartDMPlayerReplicationInfo sPRI;
```

[1] Note that this execution is being done serverside.

```

    sPRI = SDMGame.GetStats( Sender );
    sPRI.OpenMenuWindow();
}
```

In server machine the instance of SmartDMPlayerReplicationInfo associated to the player (who sent mutate command) is searched and the function OpenMenuWindow() of that instance is executed.

In SmartDMPlayerReplicationInfo, one can find the replication block

```
1 replication
  {
3       ...

5       // Toggle stats functions + some stuff
        reliable if ( Role == ROLE_Authority )
7               ToggleStats, ShowStats, Version, OpenMenuWindow;

9       ...
  }
```

Line 6 tells the engine to execute the functions (and replicate the parameters from server) on client. The function OpenMenuWindow() though called on server, will execute on the client of the player who wants to open the window.

So now we should assume that we are sitting in client's machine and think what functions and variables we can utilize (because client is dumb and only little number of variables have meaningful purpose). We know that SmartDMScoreboard.u package is already loaded at client (because we had ServerPackages = SmartDMScoreboard).

We know that class SmartDMScoreBoard is derived from class Actor. If we call a default function written in Actor and override the same function in SmartDMScoreBoard with code to open a window, our purpose will be solved. But we should be careful to use only that function of Actor which is not supposed to be used for SmartDMScoreBoard[2] by the game.

Fortunately there are some functions in Actor which, I am sure, are never used for Actor like SmartDMScoreBoard. For example

```
function SetDefaultDisplayProperties()
{
  Style = Default.Style;
  texture = Default.Texture;
  bUnlit = Default.bUnlit;
  bMeshEnviromap = Default.bMeshEnviromap;
}
```

[2]since SmartDMScoreBoard is also an Actor

So let's use this function to open the uwindow by overriding it. Open class SmartDMScoreBoard in package SmartDMScoreBoard.u and browse for the function

```
function SetDefaultDisplayProperties()
{
    local SmartDMMenuWRI SWRI;
    local SmartDMMenuWRI A;

    // check if window already open
    foreach AllActors(class'SmartDMMenuWRI', A) // check all
     existing WRIs
    {
        if (PlayerOwner == A.Owner)
        {
            A.CloseWindow();
            A.Destroy();
        }
    }
    Log("In scoreboard :)");
    SWRI = Spawn(class'SmartDMMenuWRI',PlayerOwner,,PlayerOwner.
     Location);

    if (SWRI == None)
        Log("Oops! Nothing spawned. Do something!!");
}
```

Since class SmartDMMenuWRI is present the package, there should be no problem spawning it.

Finally let's see how to call this function from instance of SmartDMPlayerReplicationInfo [3].

```
simulated function OpenMenuWindow()
{

     local Actor A;

     foreach AllActors(Class'Actor',A)
     {
      if ( A.IsA('SmartDMScoreBoard') )
      {
        A.SetDefaultDisplayProperties();
        break;
      }
     }

     if(A != none)
       A.SetDefaultDisplayProperties();
     else Log("Found nothing");

}
```

[3]present in SmartDM.u

Now remember that this function is being executed in the machine of the client. Thus, instance all the actors[4] are searched and if an instance is found[5] then the function SetDefaultDisplayProperties() is called.

There are few points worth mentioning

- function IsA('SmartDMScoreBoard') is a native bool function which takes name input variable (which in this case is SmartDMScoreBoard) and compares the name variable with the classname of the instance. Spits true if they match.
- if match is found, line A.SetDefaultDisplayProperties() calls the function defined in Actor but overridden in SmartDMScoreBoard according to our needs

In both the cases, no dependency is required.

But there is one drawback. The instance of SmartDMScoreBoard is spawned only when the player presses F1 to see the scores. Before pressing F1 there is no instance of this SmartDMScoreBoard at client's machine. Thus uwindow can not be spawned by calling A.SetDefaultDisplayProperties().

Our uwindow will open only after the player has seen scores atleast once.

2 Developing

For creating/modifying the scoreboard, one must edit file SmartDMScoreBoard.uc because instance with this name is searched by SmartDM. Now the scoreboard might have different clientside configurable settings (which can be stored in SmartDMScoreboard.ini).

One can make these settings available in SmartDMMenuWindow. Since the window is spawned clientside, it can directly manipulate the variables of SmartDMScoreBoard with immediate effect.

One can also integrate the ability of uwindow, to get notification[6] from user, with the scoreboard to implement features like scrolling.

[4]which are spawned at client machine
[5]which belongs to SmartDMScoreBoard
[6]via function Notify(UWindowDialogControl C, byte E)