Advanced FPS Kit

Team Games Package

Installing the CTF Example

This document will walk you through the necessary engine and TorqueScript changes to get the AFPSK CTF Example to properly function. If you have already installed a prior portion of the AFPSK that involves engine C++ changes related to the ShapeBase class (Team Numbers, Radar Pack), you will not need to perform the C++ step of the installation. If you are “clever” enough with TS you can bypass the C++ step and install your own method in TS to avert the C++ step if you’d like to.

C++ Installation:

Please make the following edits to ShapeBase.h

First, locate the block of code where mDamage is defined, it will be right after a protected definition, add the mTeam definition above it:

```
protected:
   S32 mTeam; //Phantom139: Added
   /// @name Damage
   /// @{
   F32 mDamage;
```

Next, scroll down to `enum ShapeBaseMasks {` and add a new mask to the list like so:

```
   enum ShapeBaseMasks {
      NameMask        = Parent::NextFreeMask,
      TeamMask        = Parent::NextFreeMask << 1,
      DamageMask      = Parent::NextFreeMask << 2,
      NoWarpMask      = Parent::NextFreeMask << 3,
      CloakMask       = Parent::NextFreeMask << 4,
      ShieldMask      = Parent::NextFreeMask << 5,
      InvincibleMask  = Parent::NextFreeMask << 6,
      SkinMask        = Parent::NextFreeMask << 7,
      MeshHiddenMask  = Parent::NextFreeMask << 8,
      SoundMaskN      = Parent::NextFreeMask << 9,       ///< Extends +
MaxSoundThreads bits
      ThreadMaskN     = SoundMaskN  << MaxSoundThreads,  ///< Extends +
MaxScriptThreads bits
      ImageMaskN      = ThreadMaskN << MaxScriptThreads, ///< Extends +
MaxMountedImage bits
      NextFreeMask    = ImageMaskN  << MaxMountedImages
   };
```

Then, scroll near the bottom of the ShapeBase class definition and you will see the following definitions:

```
F32 getLiquidHeight() { return mLiquidHeight; }
virtual WaterObject* getCurrentWaterObject();
```

```
void setCurrentWaterObject( WaterObject *obj );
virtual F32 getMass() const { return mMass; }
```

After these, add these two lines:

```
virtual S32 getTeam() { return mTeam; }
void setTeam(S32 newt) { mTeam = newt; setMaskBits(TeamMask); }
```

Save the file, then open and make the following edits to ShapeBase.cpp

First, scroll down to the ShapeBase constructor (It's located around line 900 with the format: ShapeBase::ShapeBase() : Stuff(), . . .

Inside the long list of definitions you will see this variable definition:

```
damageDir( 0.0f, 0.0f, 1.0f ),
mShapeBaseMount( NULL ),
mMass( 1.0f ),
mOneOverMass( 1.0f ),
```

Add a mTeam definition to the list:

```
damageDir( 0.0f, 0.0f, 1.0f ),
mShapeBaseMount( NULL ),
mMass( 1.0f ),
mTeam( 0 ), //< Add This
```

Next scroll down to ShapeBase::onNewDataBlock and add a mTeam definition to the list of pre-defined values as so:

```
//
mTeam = 0;
mEnergy = 0;
mDamage = 0;
mDamageState = Enabled;
```

Next, scroll way down in the file to: `U32 ShapeBase::packUpdate(NetConnection *con, U32 mask, BitStream *stream)` and make the following changes exactly as they appear here. Making a mistake here will lead to in-game crashing and access violations so you need to ensure everything here matches.

Near the bottom you will see the following "if" statement:

```
if (stream->writeFlag(mask & NameMask)) {
```

Above this if statement, add the following code block:

```
if (stream->writeFlag(mask & TeamMask)) {
   // Phantom: team
   stream->write( mTeam );
   //Phantom139: Done
}
```

Then scroll down to the unpack update function and find this if statement:

if (stream->readFlag()) { // NameMask

Then add the following code block above it.

```
if (stream->readFlag()) { // TeamMask
   //Phantom: Added Team
   stream->read(&mTeam);
   //Phantom139: End
}
```

Lastly, add the following to the bottom of the file:

```
//Phantom: Added Team Methods:
DefineEngineMethod( ShapeBase, getTeam, S32, (),,
"@brief Get the current team number used by this shape.\n\n"
"@return the team number\n\n") {
   return object->getTeam();
}
DefineEngineMethod( ShapeBase, setTeam, void, (S32 newTeam),,
"@brief Set the current team number of this shape.\n\n"
"@return void\n\n") {
   object->setTeam(newTeam);
}
```

Save and compile. This adds internal team numbers to the engine which will be used by the CTF example, and can also be used for numerous applications beyond game modes.

<u>TS Installation:</u>

To install the CTF Game mode into Torque, you will need to adjust a few aspects of your game behavior, and add a few lines of code to a few places.

First, inside your player datablocks, add the following definition to the list of maximum items:

```
maxInv[CTFFlag] = 1;
```

Then in player.cs (server-side), scroll to function PlayerData::damage and at the bottom of the function, modify this line:

```
%client.onDeath(%sourceObject, %sourceClient, %damageType, %location);
```

To Read as so:

```
Game.onDeath(%obj, %client, %sourceObject, %sourceClient, %damageType,
%location);
```

Next, in GameCore.cs and GameDM.cs as well as any custom game definitions change all instances of:

```
X::onDeath(%game, %client, %sourceObject, %sourceClient, %damageType,
%damLoc)
```

To:

```
X::onDeath(%game, %player, %client, %sourceObject, %sourceClient,
%damageType, %damLoc)
```

Then in GameCore.cs locate function GameCore::SpawnPlayer and after:

```
   if (%player.isMethod("setShapeName"))
      %player.setShapeName(%client.playerName);
```

Add:

```
   if(%client.team !$= "") {
      %player.setTeam(%client.team);
   }
```

Then at the bottom of GameCore.cs, add this new function (If you haven't read the doc. for team games regarding the team assign function):

```
function GameConnection::AssignTeam(%client, %game, %isTeamGame) {
   if(%isTeamGame) {
      // Get the count on both teams.
      for (%clientIndex = 0; %clientIndex < ClientGroup.getCount();
%clientIndex++) {
         %cl = ClientGroup.getObject(%clientIndex);
         %teamCount[%cl.team]++;
      }
      // Phantom: To-Do: Check for bot teams counts and add here as well.
      // Assign teams.
      if(%cl.team $= "") {
         //If the player hasn't already picked a team (via menu, we'll assign
one)
         if(%teamCount[2] > %teamCount[1]) {
            %client.team = 1;
         }
         else {
            %client.team = 2;
         }
      }
   }
   else {
      for(%i = 1; %game.teamUsed[%i] != 0; %i++) {} //Run a simple loop to
get the last index.
```

```
      %game.teamUsed[%i] = 1;
      %client.team = %i;
   }
}
```

Lastly, plug in the new gameCTF.cs file and execute it on the server side. You will need to set the game to CTF if you want to play the example game mode.

Setting Up a CTF Map:

To set up a CTF map you need to have two elements. First, you need to have a SimSet named TeamDropPoints containing the spawn-spheres for team 1 & 2, you will need to add the team field to the sphere manually (Dynamic Fields at the bottom of the object inspector).

The other requirement for a CTF Map is two waypoint (or marker) objects named: Team1Flag and Team2Flag respectively, you can set the team of these objects to -1 (To prevent radar interaction), the team’s flag object will spawn at these locations.

The sample Chinatown_Mist map is provided to show you how this works.