T3D Micro: Force Field - Source Changes

This document will guide you through the process of installing the necessary C++ engine changes to allow the force field to properly perform. This mainly comes in the form of adding new instances to the existing ShapeBase class to allow for internal declarations of teams and shape icon instances. If you already have installed the AFPSK Radar pack, you do not need to perform this installation. Certain pieces of this installation involve parts that have been done in AFPSK, check this document and your prior AFPSK installations to see if some of the work (involving mTeam) can be omitted.

Open up Engine/Source/T3D/ShapeBase.h and find the following class declaration:

```
class ShapeBase : public GameBase, public ISceneLight
```

Scroll down to this block of code located in the protected: section

```
/// @name Render settings
/// @{
TSShapeInstance* mShapeInstance;
Convex * mConvexList;
NetStringHandle mSkinNameHandle;
String mAppliedSkinName;
NetStringHandle mShapeNameHandle;
```

Right below the mShapeNameHandle line add the following:

```
//Phantom139: AFPSK Added
NetStringHandle mShapeIconHandle;
//Phantom139: End
```

Following that, there will be a short public: section followed by a second protected: section with the first instance being the following:

```
/// @name Damage
/// @{
F32 mDamage;
```

Above that right after protected: add the following so it shows like so:

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

Now, Scroll down to the next public: section until you see the following definitions:

```
/// @name Name & Skin tags
/// @{
void setShapeName(const char*);
const char* getShapeName();
void setSkinName(const char*);
const char* getSkinName();
/// @}
```

After that, add the following code block:

```
//Phantom139: AFPSK Added
void setShapeIcon(const char*);
const char* getShapeIcon();
//Phantom139: End
```

Next, scroll down to near the end of the ShapeBase definition to these lines:

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

And add the following after:

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

Lastly, scroll down to the bottom of the file and on the line prior to the #endif add the following code block:

```
//Phantom139: AFPSK Added
inline const char* ShapeBase::getShapeIcon() {
return mShapeIconHandle.getString();
}
//Phantom139: end
```

Save the file, then open ShapeBase.cpp. Scroll down to the ShapeBase Constructor (Around line 900-ish) that reads:

```
ShapeBase::ShapeBase() : stuff() …
```

Then there will be a large list of pre-defined values. After you see:

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

Add the following line so it reads:

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

```
mOneOverMass( 1.0f ),
```

Now scroll down to: `bool ShapeBase::onNewDataBlock( GameBaseData *dptr, bool reload )`, and you will see the following near the bottom of this function:

```
mEnergy = 0;
mDamage = 0;
mDamageState = Enabled; . . .
```

Add a similar mTeam definition above so it now reads:

```
//
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)) {
```

Inside there will be one stream pack definition. Update the if block to read the following, the area around it should now be the following:

```
else
stream->writeFlag(mFadeVal == 1.0f);
}
if (stream->writeFlag(mask & NameMask)) {
// Phantom: team
stream->write( mTeam );
con->packNetStringHandleU(stream, mShapeNameHandle);
//Phantom139: Added
con->packNetStringHandleU(stream, mShapeIconHandle);
//Phantom139: Done
}
if (stream->writeFlag(mask & ShieldMask)) {
```

Next, scroll down to `void ShapeBase::unpackUpdate(NetConnection *con, BitStream *stream)`, and make the following adjustment near the bottom:
You will see a code segment that appears like so:

```
else
mFadeVal = F32(stream->readFlag());
}
if (stream->readFlag()) { // NameMask
```

We are going to adjust the if block refering to NameMask to read the following so your code area there should now read:

```
else
mFadeVal = F32(stream->readFlag());
}
if (stream->readFlag()) { // NameMask
//Phantom: Added Team
stream->read(&mTeam);
mShapeNameHandle = con->unpackNetStringHandleU(stream);
//Phantom139: Added
mShapeIconHandle = con->unpackNetStringHandleU(stream);
//Phantom139: End
}
if(stream->readFlag()) // ShieldMask
{
```

Now, scroll down until you see `void ShapeBase::setShapeName(const char* name)` and above this function add the following code:

```
//Phantom139: AFPSK Added
void ShapeBase::setShapeIcon(const char* name) {
if (!isGhost()) {
if (name[0] != '\0') {
// Use tags for better network performance
// Should be a tag, but we'll convert to one if it isn't.
if (name[0] == StringTagPrefixByte) {
mShapeIconHandle = NetStringHandle(U32(dAtoi(name + 1)));
}
else {
mShapeIconHandle = NetStringHandle(name);
}
}
else {
mShapeIconHandle = NetStringHandle();
}
setMaskBits(NameMask);
}
}
//Phantom139: End
```

Now, scroll down and you will be in the engine function definitions, scroll down to:

```
DefineEngineMethod( ShapeBase, getShapeName, const char*, (),,
"@brief Get the name of the shape.\n\n"
"@note This is the name of the shape object that is sent to the client, "
"not the DTS or DAE model filename.\n"
"@return the name of the shape\n\n"
"@see setShapeName()\n")
{
return object->getShapeName();
}
```

After this definition add the following declarations:

```
//Phantom139: AFPSK Added
DefineEngineMethod( ShapeBase, setShapeIcon, void, ( const char* name ),,
"@brief Set the icon name of this shape.\n\n"
"@note This is used by the Advanced Radar GUI, use \t to separate special
cases (IE: Players).\n"
"@param name new icon name for the shape\n\n"
"@see getShapeIcon()\n")
{
object->setShapeIcon( name );
}
DefineEngineMethod( ShapeBase, getShapeIcon, const char*, (),,
"@brief Get the name of the shape.\n\n"
"@note This is used by the Advanced Radar GUI, use \t to separate special
cases (IE: Players).\n"
"@return the icon name of the shape\n\n"
"@see setShapeIcon()\n")
{
return object->getShapeIcon();
}
//Phantom139: End
```

Then at the bottom of the file add the following:

```
//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 your file and rebuild your solution. A quick note regarding the team field is that it makes use of the NameMask under pack and unpack so essentially the only way to update the team is the call setTeam() before using setShapeName() or setShapeIcon().

Someone with some C++ experience could change this behavior, but for applications of this pack, it is not necessary to be located elsewhere.