Advanced FPS Kit

AI Lite Pack

AI Installation & Documentation

I. C++ Installation

If you have not installed a prior portion of the AFPSK that installs the team field on the ShapeBase object, you will need to make the following changes.

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.

II. AI Documentation

The Advanced AI object is a brand new form based on the thinking of the AIPlayer class. The AI itself doesn't base off of the AIPlayer. The class bases off of Player and uses functioning similar to the AIPlayer while expanding the functioning to use new code.

Applications of AdvancedAIPlayer are exactly the same to AIPlayer, therefore AI-Lite can be directly applied to other AI Solutions without too much hastle. Below is a list of the TS Functions that are similar and different from the original AIPlayer. Here are the new functions:

- AdvancedAIPlayer::getClosestObject(U32 typeMask, bool viewReq): This function provides two methods for obtaining the closest object of a certain typemask. viewReq defaults to "false", and setting this to true will enforce a requirement that the object be in a line of sight with the player.

- AdvancedAIPlayer::getClosestEnemyObject(U32 typeMask, bool viewReq): This function uses the exact same logic as getClosestObject(), however it also checks to see if the closest object is using a team number that is different from the AI.
- AdvancedAIPlayer::hasLOSTo(Object o): This function checks to see if the AI has a line of sight to Object o.
- AdvancedAIPlayer::setCurrentNavMesh(S32 meshID): This function sets the AI to use a recast navigation mesh on the map. When you call this function, the AI will automatically create a navPath object if it needs one. This function requires that you send the object ID of the NavMesh, and not the name, therefore you need to use the nameToID() function to obtain the ID.
- AdvancedAIPlayer::setMoveDestination(Point3F target, bool slowdown): This function behaves exactly the same as the original AIPlayer::setMoveDestination() function, however it automatically plans a recast path out to target and then the AI will follow the path.
- AdvancedAIPlayer::setMoveDestination_old(Point3F target, bool slowdown): This function is included for T3D AI Packs that use their own pathfinding solutions. This code will use the original AIPlayer's setMoveDestination code and will perform similar logic to the code, allowing you to use this if you need it.

All of the remaining functions the AdvancedAIPlayer uses are exactly the same to the original function names, and no changes are applied to them. In order to use the new navmesh functions, your T3D solution must have recast enabled on it, which can be enabled by using:

includeModule('navigation');

In your projects: project.conf file before generating the project for usage in MSVS.

Included in this pack is a set of TS AI Functions that I use in my FPS Demo Project. They are widely based on Steve Acaster's Recast AI Deathmatch resource, however they have been adjusted to use the new AI Player's functions. For owners of the Radar Pack and the FPS Design pack, the AI is built in with radar functions, and custom class creation functions to allow the AI to use custom loadouts and appear on your radars by default.