```
#include "AUDIO_FMODAudioSystem.h"
#include "AUDIO_Core.h"

////CONSTRUCTOR
CCARAudioCore::CCARAudioCore(ECARCoreNames coreName, bool
isActive)
    :CCARBaseCore(coreName, isActive)
{
    epAudioMainInterface = new CCARMainInterface(eSound);
    m_pAdoSystem = 0;
    m_bIsLoaded = false;
}
/
*================================================================
============================*/
////DESTURCTOR
CCARAudioCore::~CCARAudioCore()
{
    SAFE_DELETE(m_pAdoSystem);
    ShutDown();
}
/
*================================================================
============================*/
////DECODE MESSAGES
void CCARAudioCore::DecodeMessage()
{
    if(m_bIsLoaded)
    {
        for(m_tdmPkgMap::const_iterator cIter =
epAudioMainInterface->GetMsg()->m_mArgList.begin(); cIter !=
epAudioMainInterface->GetMsg()->m_mArgList.end(); cIter++)
        {
            switch(cIter->second->GetType())
            {
            case eLoadSound2D:
                {
                    STCARLoadSound2D* temp =
dynamic_cast<STCARLoadSound2D*>(cIter->second);

                    STCAR_2D_SOUNDDESC desc = {temp->id, temp-
>pan, temp->volume, temp->priority, temp->loopCount, temp-
>pause, temp->fileName.c_str(),temp->type};

                    LoadSound2D(desc);

                    break;
                }
            case eLoadSound3D:
```

```
                {
                    STCARLoadSound3D* temp =
dynamic_cast<STCARLoadSound3D*>(cIter->second);


                    STCAR_3DSOUNDCONE sCone = {temp-
>soundConeExteriorVolume, temp->soundConeExteriorAngle, temp-
>soundConeInteriorAngle, temp->soundConeHeading};
                    STCAR_3D_SOUNDDESC desc = {temp->id, temp-
>volume, temp->priority, temp->loopCount, temp->pause, temp-
>size, temp->spread, temp->fileName.c_str(), temp->position,
temp->velocity, temp->type, sCone};

                    LoadSound3D(desc);

                    break;
                }
            case eUpdateEars:
                {
                    STCARUpdateEars* temp =
dynamic_cast<STCARUpdateEars*>(cIter->second);

                    STCAR_EARS ears = {temp->up, temp->heading,
temp->position, temp->velocity};

                    UpdateEars(ears);

                    break;
                }
            case eUnloadSound:
                {
                    STCARUnloadSound* temp =
dynamic_cast<STCARUnloadSound*>(cIter->second);

                    STCAR_AUDIOKEY key = {temp->id, temp->type};

                    UnloadSound(key);

                    break;
                }
            case ePlaySound:
                {
                    STCARPlaySound* temp =
dynamic_cast<STCARPlaySound*>(cIter->second);

                    STCAR_AUDIOKEY key = {temp->id, temp->type};

                    PlaySound(key);
                    break;
```

```
                }
            case ePauseSound:
                {
                    STCARPauseSound* temp =
dynamic_cast<STCARPauseSound*>(cIter->second);

                    STCAR_AUDIOKEY key = {temp->id, temp->type};

                    PauseSound(key);

                    break;
                }
            case eUnpauseSound:
                {
                    STCARUnpauseSound* temp =
dynamic_cast<STCARUnpauseSound*>(cIter->second);

                    STCAR_AUDIOKEY key = {temp->id, temp->type};

                    UnpauseSound(key);

                    break;
                }
            case eGetSoundPan:
                {
                    STCARGetSoundPan* temp =
dynamic_cast<STCARGetSoundPan*>(cIter->second);

                    STCAR_AUDIOKEY key = {temp->id, temp->type};

                    GetSoundPan(key);

                    break;
                }
            case eGetSoundVolume:
                {
                    STCARGetSoundVolume* temp =
dynamic_cast<STCARGetSoundVolume*>(cIter->second);

                    STCAR_AUDIOKEY key = {temp->id, temp->type};

                    GetSoundVolume(key);

                    break;
                }
            case eGetSoundPriority:
                {
                    STCARGetSoundPriority* temp =
dynamic_cast<STCARGetSoundPriority*>(cIter->second);
```

```
                    STCAR_AUDIOKEY key = {temp->id, temp->type};

                    GetSoundPriority(key);

                    break;
                }
            case eGetSoundLoopCount:
                {
                    STCARGetSoundLoopCount* temp =
dynamic_cast<STCARGetSoundLoopCount*>(cIter->second);

                    STCAR_AUDIOKEY key = {temp->id, temp->type};

                    GetSoundLoopCount(key);

                    break;
                }
            case eGetSoundState:
                {
                    STCARGetSoundState* temp =
dynamic_cast<STCARGetSoundState*>(cIter->second);

                    STCAR_AUDIOKEY key = {temp->id, temp->type};

                    GetSoundState(key);

                    break;
                }
            case eGetSoundSize:
                {
                    STCARGetSoundSize* temp =
dynamic_cast<STCARGetSoundSize*>(cIter->second);

                    STCAR_AUDIOKEY key = {temp->id, temp->type};

                    GetSoundSize(key);

                    break;
                }
            case eGetSoundSpread:
                {
                    STCARGetSoundSpread* temp =
dynamic_cast<STCARGetSoundSpread*>(cIter->second);

                    STCAR_AUDIOKEY key = {temp->id, temp->type};

                    GetSoundSpread(key);
```

```
                    break;
                }
            case eGetSoundName:
                {
                    STCARGetSoundName* temp =
dynamic_cast<STCARGetSoundName*>(cIter->second);

                    STCAR_AUDIOKEY key = {temp->id, temp->type};

                    GetSoundName(key);

                    break;
                }
            case eGetEars:
                {
                    GetEars();

                    break;
                }
            case eGetSoundPosition:
                {
                    STCARGetSoundPosition* temp =
dynamic_cast<STCARGetSoundPosition*>(cIter->second);

                    STCAR_AUDIOKEY key = {temp->id, temp->type};

                    GetSoundPosition(key);

                    break;
                }
            case eGetSoundVelocity:
                {
                    STCARGetSoundVelocity* temp =
dynamic_cast<STCARGetSoundVelocity*>(cIter->second);

                    STCAR_AUDIOKEY key = {temp->id, temp->type};

                    GetSoundVelocity(key);

                    break;
                }
            case eGetSoundCone:
                {
                    STCARGetSoundCone* temp =
dynamic_cast<STCARGetSoundCone*>(cIter->second);

                    STCAR_AUDIOKEY key = {temp->id, temp->type};

                    GetSoundCone(key);
```

```
                    break;
                }
            case eSetSoundPan:
                {
                    STCARSetSoundPan* temp =
dynamic_cast<STCARSetSoundPan*>(cIter->second);

                    STCAR_AUDIOKEY key = {temp->id, temp->type};

                    SetSoundPan(key, temp->pan);

                    break;
                }
            case eSetSoundVolume:
                {
                    STCARSetSoundVolume* temp =
dynamic_cast<STCARSetSoundVolume*>(cIter->second);

                    STCAR_AUDIOKEY key = {temp->id, temp->type};

                    SetSoundVolume(key, temp->volume);

                    break;
                }
            case eSetSoundPriority:
                {
                    STCARSetSoundPriority* temp =
dynamic_cast<STCARSetSoundPriority*>(cIter->second);

                    STCAR_AUDIOKEY key = {temp->id, temp->type};

                    SetSoundPriority(key, temp->priority);

                    break;
                }
            case eSetSoundLoopCount:
                {
                    STCARSetSoundLoopCount* temp =
dynamic_cast<STCARSetSoundLoopCount*>(cIter->second);

                    STCAR_AUDIOKEY key = {temp->id, temp->type};

                    SetSoundLoopCount(key, temp->loopCount);

                    break;
                }
            case eSetSoundSize:
                {
```

```
                    STCARSetSoundSize* temp =
dynamic_cast<STCARSetSoundSize*>(cIter->second);

                    STCAR_AUDIOKEY key = {temp->id, temp->type};

                    SetSoundSize(key, temp->size);

                    break;
                }
            case eSetSoundSpread:
                {
                    STCARSetSoundSpread* temp =
dynamic_cast<STCARSetSoundSpread*>(cIter->second);

                    STCAR_AUDIOKEY key = {temp->id, temp->type};

                    SetSoundSpread(key, temp->spread);

                    break;
                }
            case eSetSoundPosition:
                {
                    STCARSetSoundPosition* temp =
dynamic_cast<STCARSetSoundPosition*>(cIter->second);

                    STCAR_AUDIOKEY key = {temp->id, temp->type};

                    SetSoundPosition(key, temp->position);

                    break;
                }
            case eSetSoundVelocity:
                {
                    STCARSetSoundVelocity* temp =
dynamic_cast<STCARSetSoundVelocity*>(cIter->second);

                    STCAR_AUDIOKEY key = {temp->id, temp->type};

                    SetSoundVelocity(key, temp->velocity);

                    break;
                }
            case eSetSoundCone:
                {
                    STCARSetSoundCone* temp =
dynamic_cast<STCARSetSoundCone*>(cIter->second);

                    STCAR_AUDIOKEY key = {temp->id, temp->type};
                    STCAR_3DSOUNDCONE sCone = {temp-
```

```
>exteriorVolume, temp->exteriorAngle, temp->interiorAngle, temp-
>heading};

                    SetSoundCone(key, sCone);

                    break;
                }
            }
        }
    }
}
/
*==============================================================
============================*/
////START UP
void CCARAudioCore::IgnitionSwitch()
{
    m_pAdoSystem = new CCARFMODAudioSystem();
    m_pAdoSystem->StartUp();
    m_bIsLoaded = true;

    //TESTING DELETE ME
    //STCAR_AUDIOKEY key;
    //STCAR_2D_SOUNDDESC desc = {0, 0, 100, 0, 0, false, "FMOD/
POH.mp3", SOUND_2DLONG};
    //key = m_pAdoSystem->LoadSound2D(desc);
    //m_pAdoSystem->PlaySoundA(key);
}
/
*==============================================================
============================*/
////UPDATE
void CCARAudioCore::UpdateCore(float dt)
{
    m_pAdoSystem->Update(0.005f);
    epAudioMainInterface->CheckInbox(eSound);
}
/
*==============================================================
============================*/
////SHUT DOWN
void CCARAudioCore::ShutDown()
{
    m_pAdoSystem->ShutDown();
}
/
*==============================================================
============================*/
////LOAD 2D SOUND
```

```
STCAR_AUDIOKEY CCARAudioCore::LoadSound2D(STCAR_2D_SOUNDDESC
desc)
{
    return m_pAdoSystem->LoadSound2D(desc);
}
/
*================================================================
============================*/
////LOAD 3D SOUND
STCAR_AUDIOKEY CCARAudioCore::LoadSound3D(STCAR_3D_SOUNDDESC
desc)
{
    return m_pAdoSystem->LoadSound3D(desc);
}
/
*================================================================
============================*/
////UPDATE EARS
void CCARAudioCore::UpdateEars(STCAR_EARS ears)
{
    m_pAdoSystem->UpdateEars(ears);
}
/
*================================================================
============================*/
////UNLOAD SOUND
void CCARAudioCore::UnloadSound(STCAR_AUDIOKEY key)
{
    m_pAdoSystem->UnloadSound(key);
}
/
*================================================================
============================*/
////PLAY SOUND
void CCARAudioCore::PlaySound(STCAR_AUDIOKEY key)
{
    m_pAdoSystem->PlaySound(key);
}
/
*================================================================
============================*/
////PAUSE SOUND
void CCARAudioCore::PauseSound(STCAR_AUDIOKEY key)
{
    m_pAdoSystem->PauseSound(key);
}
/
*================================================================
============================*/
```

```
////UNPAUSE SOUND
void CCARAudioCore::UnpauseSound(STCAR_AUDIOKEY key)
{
    m_pAdoSystem->UnpauseSound(key);
}
/
*================================================================
============================*/
////GET SOUND PAN
int CCARAudioCore::GetSoundPan(STCAR_AUDIOKEY key)
{
    return m_pAdoSystem->GetSoundPan(key);
}
/
*================================================================
============================*/
////GET SOUND VOLUME
int CCARAudioCore::GetSoundVolume(STCAR_AUDIOKEY key)
{
    return m_pAdoSystem->GetSoundVolume(key);
}
/
*================================================================
============================*/
////GET SOUND PRIORITY
int CCARAudioCore::GetSoundPriority(STCAR_AUDIOKEY key)
{
    return m_pAdoSystem->GetSoundPriority(key);
}
/
*================================================================
============================*/
////GET SOUND LOOP COUNT
int CCARAudioCore::GetSoundLoopCount(STCAR_AUDIOKEY key)
{
    return m_pAdoSystem->GetSoundLoopCount(key);
}
/
*================================================================
============================*/
////GET SOUND STATE
bool CCARAudioCore::GetSoundState(STCAR_AUDIOKEY key)
{
    return m_pAdoSystem->GetSoundState(key);
}
/
*================================================================
============================*/
////GET SOUND SIZE
```

```
float CCARAudioCore::GetSoundSize(STCAR_AUDIOKEY key)
{
    return m_pAdoSystem->GetSoundSize(key);
}
/
*============================================================
============================*/
////GET SOUND SPREAD
float CCARAudioCore::GetSoundSpread(STCAR_AUDIOKEY key)
{
    return m_pAdoSystem->GetSoundSpread(key);
}
/
*============================================================
============================*/
////GET SOUND NAME
char* CCARAudioCore::GetSoundName(STCAR_AUDIOKEY key)
{
    return m_pAdoSystem->GetSoundName(key);
}
/
*============================================================
============================*/
////GET EARS
STCAR_EARS CCARAudioCore::GetEars()
{
    return m_pAdoSystem->GetEars();
}
/
*============================================================
============================*/
////GET SOUND POSITION
STCAR_VECTOR CCARAudioCore::GetSoundPosition(STCAR_AUDIOKEY key)
{
    return m_pAdoSystem->GetSoundPosition(key);
}
/
*============================================================
============================*/
////GET SOUND VELOCITY
STCAR_VECTOR CCARAudioCore::GetSoundVelocity(STCAR_AUDIOKEY key)
{
    return m_pAdoSystem->GetSoundVelocity(key);
}
/
*============================================================
============================*/
////GET SOUND CONE
STCAR_3DSOUNDCONE CCARAudioCore::GetSoundCone(STCAR_AUDIOKEY
```

```
key)
{
    return m_pAdoSystem->GetSoundCone(key);
}
/
*================================================================
============================*/
////GET SOUND DESCRIPTION 2D
STCAR_2D_SOUNDDESC CCARAudioCore::GetSoundDesc2D(STCAR_AUDIOKEY
key)
{
    return m_pAdoSystem->GetSoundDesc2D(key);
}
/
*================================================================
============================*/
////GET SOUND DESCRIPTION 3D
STCAR_3D_SOUNDDESC CCARAudioCore::GetSoundDesc3D(STCAR_AUDIOKEY
key)
{
    return m_pAdoSystem->GetSoundDesc3D(key);
}
/
*================================================================
============================*/
////
void CCARAudioCore::SetSoundPan(STCAR_AUDIOKEY key, int pan)
{
    m_pAdoSystem->SetSoundPan(key, pan);
}
/
*================================================================
============================*/
////
void CCARAudioCore::SetSoundVolume(STCAR_AUDIOKEY key, int
volume)
{
    m_pAdoSystem->SetSoundVolume(key, volume);
}
/
*================================================================
============================*/
////
void CCARAudioCore::SetSoundPriority(STCAR_AUDIOKEY key, int
priority)
{
    m_pAdoSystem->SetSoundPriority(key,priority);
}
/
```

```
*==============================================================
============================*/
////
void CCARAudioCore::SetSoundLoopCount(STCAR_AUDIOKEY key, int
loopCount)
{
    m_pAdoSystem->SetSoundLoopCount(key, loopCount);
}
/
*==============================================================
============================*/
////
void CCARAudioCore::SetSoundSize(STCAR_AUDIOKEY key, float size)
{
    m_pAdoSystem->SetSoundSize(key, size);
}
/
*==============================================================
============================*/
////
void CCARAudioCore::SetSoundSpread(STCAR_AUDIOKEY key, float
angle)
{
    m_pAdoSystem->SetSoundSpread(key, angle);
}
/
*==============================================================
============================*/
////
void CCARAudioCore::SetSoundPosition(STCAR_AUDIOKEY
key,STCAR_VECTOR position)
{
    m_pAdoSystem->SetSoundPosition(key,position);
}
/
*==============================================================
============================*/
////
void CCARAudioCore::SetSoundVelocity(STCAR_AUDIOKEY
key,STCAR_VECTOR velocity)
{
    m_pAdoSystem->SetSoundVelocity(key,velocity);
}
/
*==============================================================
============================*/
////
void CCARAudioCore::SetSoundCone(STCAR_AUDIOKEY
key,STCAR_3DSOUNDCONE soundCone)
```

```
{
    m_pAdoSystem->SetSoundCone(key,soundCone);
}
/
*============================================================
============================*/
```