# helper_modules.hps

#### Description_AddExt

```
int Description_AddExt(const tString &in asTextCat,
                       const tString &in asTextEntry,
                       float afTimeMul,
                       bool abPutInQueueIfShowing,
                       bool abConnectedToPrevious,
                       eDescEffect aEffect,
                       float afEffectAmount,
                       cColor aColor)
```

---

#### Description_Add

```
int Description_Add(const tString &in asTextCat,
                    const tString &in asTextEntry,
                    float afTimeMul,
                    bool abPutInQueueIfShowing)
```

---

#### Description_IsActive

```
bool Description_IsActive()
```

---

#### Description_GetCurrentId

```
int Description_GetCurrentId()
```

---

#### Description_SetForceFading

```
void Description_SetForceFading(bool abX)
```

---

#### TerrainParticles_Disable

```
void TerrainParticles_Disable()
```

Disables terrain particles, they have to be setup again to be started

---

#### TerrainParticles_SetupRange

```
void TerrainParticles_SetupRange(float afRangeMinStart,
                                 float afRangeMinEnd,
                                 float afRangeMaxStart,
                                 float afRangeMaxEnd)
```

Sets up the maximum and minimum distance that the particles are visible Particles also wrap around the maximum distance, coming out on the other side if they get to far away This should be called before spawning particles to get the best spread

- **afRangeMinStart**: particles closer than this value will not be visible
- **afRangeMinEnd**: particles between min start and min end will fade to fully visible
- **afRangeMaxStart**: particles further away then this value will start to fade out
- **afRangeMaxEnd**: particles further away then this will not be visible and will be spawned on the other side of the player

---

#### TerrainParticles_SetupParticles

```
void TerrainParticles_SetupParticles(const tString &in asMaterial,
                                     int alFixedAxisCount,
                                     float afFixedAxisSizeMin,
                                     float afFixedAxisSizeMax,
                                     cColor avFixedColorMin,
                                     cColor avFixedColorMax,
                                     int alPointSpriteCount,
                                     float afPointSpriteSizeMin,
                                     float afPointSpriteSizeMax,
                                     cColor avPointColorMin,
                                     cColor avPointColorMax)
```

Spawn particles that use the specified material There are two kind of particles. FixedAxis sprites follow the rotation of the surface. PointSprite always look at the player.

- **asMaterial**: the mat file used by the particles
- **alFixedAxisCount**: the number of FixedAxis particles to spawn
- **afFixedAxisSizeMin**: smallest size of a FixedAxis particle
- **afFixedAxisSizeMax**: largest size of a FixedAxis particle
- **avFixedColorMin**: random color bound of the FixedAxis particle
- **avFixedColorMax**: random color bound of the FixedAxis particle
- **alPointSpriteCount**: the number of PointSprite particles to spawn
- **afPointSpriteSizeMin**: smallest size of a PointSprite particle
- **afPointSpriteSizeMax**: largest size of a PointSprite particle

- **avPointColorMin**: random color bound of the PointSprite particle
- **avPointColorMax**: random color bound of the PointSprite particle

---

#### TerrainParticles_SetupForces

```
void TerrainParticles_SetupForces(const cVector3f &in avVelocityLimit,
                                  float afGravityAmount,
                                  float afRandomDirectionForce,
                                  float afRandomDirectionFrequency)
```

Setup forces that affect the particles

- **avVelocityLimit**: limit to velocity of the particles
- **afGravityAmount**: how much gravity should affect the particles, value between 0.0-1.0
- **afRandomDirectionForce**: the amount of random force that should be appled to the particles, the direction of this force is based on position and changes over time
- **afRandomDirectionFrequency**: how fast the direction of the random force changes

---

#### TerrainParticles_SetupWind

```
void TerrainParticles_SetupWind(const cVector3f &in avWindForce)
```

Setup wind force that is applied to all particles

- **avWindForce**: the force and direction of the wind, this is applied to all particles

---

#### DistortionEffect_AddInstance

```
void DistortionEffect_AddInstance(tID a_idEntity,
                                  float afMaxDistance,
                                  float afMinDistance,
                                  float afMaxAmount,
                                  eEasing aEasing=eEasing_Linear)
```

Adds the entity has an instance that will help create a distortion effect if the player is near enough.

- **a_idEntity**: The entity who this will be connected to.
- **afMaxDistance**: The distance at which the effect will start.
- **afMinDistance**: The distance when the effect will be at max
- **afMaxAmount**: The max amount the effect will have from this entity.
- **aEasing**: The easing function used to fade between max and min

---

#### DistortionEffect_RemoveInstance

```
void DistortionEffect_RemoveInstance(tID a_idEntity)
```

Removes an instance that will create the distortion effect.

- **a_idEntity**: The entity to break the connection with

---

#### LightFlash_Add

```
int LightFlash_Add(float afTime,
                   float afFadeInTime,
                   float afFadeOutTime,
                   const tString &in asTargetEntity,
                   const cColor &in aColor,
                   float afRadius,
                   float afBrightness)
```

Creates a flash of light at the postion of an entity. Returns id of the flash.

- **afTime**: The time that light light stays at full intensity
- **afFadeInTime**: How long it takes to fade in
- **afFadeOutTime**: How long it takes to fade out
- **asTargetEntity**: Entity at which it will be created.
- **aColor**: The color of the light.
- **afRadius**: The radius of the light (this will NOT be faded, it starts with this radius, and then just color fades in)
- **afBrightness**: The brightness of the light.

---

#### LightFlash_Add

```
int LightFlash_Add(float afTime,
                   float afFadeInTime,
                   float afFadeOutTime,
                   const cVector3f &in avTargetPos,
                   const cColor &in aColor,
                   float afRadius,
                   float afBrightness)
```

Creates a flash of light at the postion of an entity. Returns id of the flash.

- **afTime**: The time that light light stays at full intensity
- **afFadeInTime**: How long it takes to fade in
- **afFadeOutTime**: How long it takes to fade out

- **avTargetPos**: Position at which it will be created.
- **aColor**: The color of the light.
- **afRadius**: The radius of the light (this will NOT be faded, it starts with this radius, and then just color fades in)
- **afBrightness**: The brightness of the light.

---

#### ToolHandler_CanPickOrUseTool

```
bool ToolHandler_CanPickOrUseTool()
```

Checks if it is OK to use an item or picking one up. Basically just wanna make sure the tool handler (or something else) is not playing an animation (apart form idle)

---

#### ToolHandler_CanEquipToolAreaTool

```
bool ToolHandler_CanEquipToolAreaTool()
```

Meant for tool area. Does a check to see if it is OK to equip the tool in that area.

---

#### ToolHandler_GetPlayerCarriesHeavyTool

```
bool ToolHandler_GetPlayerCarriesHeavyTool()
```

Checks if the player carries a tool that is heavy

---

#### ToolHandler_IsActive

```
bool ToolHandler_IsActive()
```

Is active?

---

#### ToolHandler_SetActive

```
void ToolHandler_SetActive(bool abX)
```

---

#### AttackMeter_AddInstance

```
void AttackMeter_AddInstance(tID a_idEntity,
                             float afStartDistance,
                             float afEndDistance,
                             float afDamage,
                             bool abFatal=false,
                             bool abFastNonLookKill=false,
                             bool abInstantKnockOut=false)
```

Adds an entity as something that is about to damage the player.

- **a_idEntity**: ID for the entity
- **afStartDistance**: How close entity needs to be before the attack meter starts.
- **afEndDistance**: How close the entity needs to be for the player to take damange
- **afDamage**: How much damage will the attack take?
- **abFatal**: If true then this death is instantly fatal rather than allowing you to die 3 times.
- **abFastNonLookKill**: If true, the entity will damage quickly even if the player isn't looking at it.
- **abInstantKnockOut**: if true, the player will be knocked out on the first attack

---

#### AttackMeter_RemoveInstance

```
void AttackMeter_RemoveInstance(tID a_idEntity)
```

Removes an entity as something that is about to damage the player.

- **a_idEntity**: ID for the entity

---

#### AttackMeter_AutoDamage

```
void AttackMeter_AutoDamage(float afTime,
                            float afDamage,
                            bool abFatal)
```

This starts a timer that will damage the player when it is over.

- **afTime**: ID for the entity
- **afDamage**: Amount of damage dealt to the player
- **abFatal**: If true, the damage can kill the player

---

#### AttackMeter_IsActive

```
bool AttackMeter_IsActive()
```

Checks if any effects are active

---

#### AttackMeter_SetPlayerIsRecovering

```
void AttackMeter_SetPlayerIsRecovering(bool abX)
```

Manually set that the player is in recovery mode as if he has just been attacked. While true, most Agents refrain from attacking and player cannot get knocked down.

---

#### AttackMeter_GetPlayerIsRecovering

```
bool AttackMeter_GetPlayerIsRecovering()
```

Check if the attack meter has just damaged the player and the player currently have the effects caused by that.

---

#### AttackMeter_SetCustomDamageCallback

```
void AttackMeter_SetCustomDamageCallback(const tString &in asCallback)
```

This lets you do your own damage event. Nothing is done in the AttackHandler when damage is to occurs except calling this.

- **asCallback**: The callback function to be used. Syntax: void f(const tString &in asSource)

---

#### AttackMeter_SetAfterDamageCallback

```
void AttackMeter_SetAfterDamageCallback(const tString &in asCallback)
```

This sets a callback that is called after taken daming.

- **asCallback**: The callback function to be used. Syntax: void f(const tString& in asSource)

---

#### AttackMeter_SetAttackerTeleportPosition

```
void AttackMeter_SetAttackerTeleportPosition(const tString &in asPosition)
```

If the player blacks out when damaged the agent that hurt him will be telported to one of these postions.

---

#### AttackMeter_GetAttackerTeleportPosition

```
tString AttackMeter_GetAttackerTeleportPosition()
```

Gets the base name (with wildcards) that was set with AttackMeter_SetAttackerTeleportPosition

---

#### AttackMeter_BreakAttackDamageSequence

```
void AttackMeter_BreakAttackDamageSequence()
```

Only to be called in _Global_PlayerAfterDamageCallback of an Agent or in an AfterDamageCallback. It breaks the sequence to continue. The following things will be changed WorldAll volume to 0, player move & look speed, effect fade out, and PlayerIsRecovering is true.

---

#### AttackMeter_ContinueAttackDamageSequence

```
void AttackMeter_ContinueAttackDamageSequence()
```

Only to be called after _Global_PlayerAfterDamageCallback of an Agent or in an AfterDamageCallback if AttackMeter_BreakAttackDamageSequence has been called. This simple continue this

---

#### AttackMeter_GetTimeSinceLastKnockDown

```
float AttackMeter_GetTimeSinceLastKnockDown()
```

Time since player was last knocked down.

---

#### GameOver_Start

```
void GameOver_Start(const tString &in asDeathSource)
```

Starts the Game Over screen

---

#### GameOver_End

```
void GameOver_End()
```

Ends the Game Over screen

---

#### GameOver_GetTimeSinceLastGameOver

```
float GameOver_GetTimeSinceLastGameOver()
```

Ends the Game Over screen

---

#### GameOver_SetCustomGameOverScreenText

```
void GameOver_SetCustomGameOverScreenText(const tString &in asCat,
                                          const tString &in asEntry)
```

Sets the message that will be displayed before a checkpoint is loaded (after proper death):

---

#### Emotion_StartHeartbeat

```
int Emotion_StartHeartbeat(float afTimeBetweenBeats,
                           float afVolume,
                           int alPrio,
                           float afDuration=-1,
                           float afFadeInTime=3,
                           float afFadeOutTime=3)
```

Starts a new instance of heart beats

- **afTimeBetweenBeats**: The time between the beats
- **afVolume**: Volume of the heart beats (0-1)
- **alPrio**: The prio of instnace. Higher is played over low.er
- **afDuration**: How long the heart beats lasts, -1 means until stopped.
- **afFadeInTime**: The time it takes for this instance to fade in.
- **afFadeOutTime**: The time it takes to fade out, only used if no instance is active.

#### Emotion_StopHeartbeat

```
void Emotion_StopHeartbeat(int alId)
```

Stops an instance of heart beats.

- **alId**: The ID of the instance

---

#### Emotion_SetHeartbeatProperties

```
void Emotion_SetHeartbeatProperties(int alId,
                                    float afTimeBetweenBeats,
                                    float afVolume)
```

Changes the properties for heart beat instance

- **alId**: The ID of the instance
- **afTimeBetweenBeats**: Time between beats
- **afVolume**: Volume for the beat

---

#### Emotion_StartBackgroundBreath

```
int Emotion_StartBackgroundBreath(eBreathType aType,
                                  float afStrength,
                                  int alPrio,
                                  float afDuration=-1,
                                  float afFadeInSpeed=0.2f,
                                  float afFadeOutSpeed=0.2f)
```

Starts a new instance of background breathing. Returns ID of the instnace.

- **aType**: The type of breath.
- **afStrength**: The strength of the breaths. Must be 0 - 1.
- **alPrio**: The prio of this instance. Higher is played over lower.
- **afDuration**: How long the breathing lasts, -1 means until stopped.
- **afFadeInSpeed**: the speed (note NOT time) that used when fading to this instance. If <0, the FadeOut from a previous instance is used.
- **afFadeOutSpeed**: if going back to default breathing or the upcoming has<0 as FadeInSpeed, this it the speed (NOT time) used. Note that if another instance is taking over, its FadeInSpeed will be used instead.

#### Emotion_StopBackgroundBreath

```
void Emotion_StopBackgroundBreath(int alID)
```

Stops an instance of background breathing.

- **alID**: The ID of the instance

---

#### Emotion_SetBackgroundBreathMuted

```
void Emotion_SetBackgroundBreathMuted(int alID,
                                      bool abX)
```

Sets an instance of background breathing muted. Note that this muting only works if the is the top prio instnace and in that case no other instance will play either.

- **alID**: The ID of the instance
- **abX**: If the instance is muted. -1 = the default breathing.

---

#### Emotion_PlayEventBreath

```
void Emotion_PlayEventBreath(const tString &in asSound,
                             int alPrio=1)
```

Plays an event breathing sound.

- **asSound**: The name of the sound file/event to play.
- **alPrio**: The prio of the sound, if a a breaht is already playing having higher prio overrides. 0 is lowest prio and usually used for non-critical stuff (such as breathing when crawling), so mostly use 1 or higher.

---

#### Wake_SetAsleep

```
void Wake_SetAsleep(bool abAsleep)
```

Puts the player instantly to sleep (or awake)

- **abAsleep**: if true then the player is asleep, if false then awake

---

#### Wake_StartWakeup

```
void Wake_StartWakeup(float afTime)
```

Wakes up the player.

- **afTime**: time to take to wake up.

---

#### PauseMenu_Enabled

```
void PauseMenu_Enabled(bool abActive)
```

Sets if the pause menu is allowed to be used

---

#### PauseMenu_Show

```
void PauseMenu_Show(bool abActive)
```

Show the pause manu

---

#### MainMenu_Show

```
void MainMenu_Show(bool abX)
```

If main menu should be shown

---

#### MainMenu_IsShowing

```
bool MainMenu_IsShowing()
```

Return whether the Main Menu is active or not

---

#### MainMenu_GameOver

```
void MainMenu_GameOver()
```

Returning to the main menu when the game's complete

---

#### MainMenu_SetBGPhase

```
void MainMenu_SetBGPhase(eMainMenuPhase aPhase)
```

Change main menu BG state

---

#### MainMenu_SetIsE3Demo

```
void MainMenu_SetIsE3Demo()
```

---

#### LastOnSoma_SetText

```
void LastOnSoma_SetText(const tString &in asText)
```

---

#### Hint_ShowHint

```
void Hint_ShowHint(const tString &in asTextCat,
                   const tString &in asTextEntry,
                   bool abIsInputHint=false,
                   float afTimeMul=1.5f,
                   bool abAddAsGiven=true)
```

---

#### Hint_ShowHint_Hold

```
int Hint_ShowHint_Hold(const tString &in asTextCat,
                       const tString &in asTextEntry,
                       bool abIsInputHint=false,
                       float afTimeMul=1.5f,
                       bool abAddAsGiven=true)
```

---

#### Hint_ShowInfo

```
void Hint_ShowInfo(const tString &in asTextCat,
```

```
                    const tString &in asTextEntry,
                    bool abIsInputHint=false,
                    float afTimeMul=1.5f,
                    bool abAddAsGiven=true)
```

---

#### Hint_ShowInfo_Hold

```
int Hint_ShowInfo_Hold(const tString &in asTextCat,
                       const tString &in asTextEntry,
                       bool abIsInputHint=false,
                       float afTimeMul=1.5f,
                       bool abAddAsGiven=true)
```

---

#### Hint_ShowDanger

```
void Hint_ShowDanger(const tString &in asTextCat,
                     const tString &in asTextEntry,
                     bool abIsInputHint=false,
                     float afTimeMul=1.5f,
                     bool abAddAsGiven=true)
```

---

#### Hint_ShowDanger_Hold

```
int Hint_ShowDanger_Hold(const tString &in asTextCat,
                         const tString &in asTextEntry,
                         bool abIsInputHint=false,
                         float afTimeMul=1.5f,
                         bool abAddAsGiven=true)
```

---

#### Hint_ShowAlert

```
void Hint_ShowAlert(const tString &in asTextCat,
                    const tString &in asTextEntry,
                    bool abIsInputHint=false,
                    float afTimeMul=1.5f,
                    bool abAddAsGiven=true)
```

---

**Hint_ShowAlert_Hold**

```
int Hint_ShowAlert_Hold(const tString &in asTextCat,
                        const tString &in asTextEntry,
                        bool abIsInputHint=false,
                        float afTimeMul=1.5f,
                        bool abAddAsGiven=true)
```

---

**Hint_Show_Helper**

```
void Hint_Show_Helper(const tString &in asHeaderEntry,
                      const tString &in asTextCat,
                      const tString &in asTextEntry,
                      bool abIsInputHint,
                      const cColor &in aColor,
                      float afTimeMul,
                      bool abAddAsGiven)
```

---

**Hint_Show_Ext**

```
int Hint_Show_Ext(const tString &in asHeaderCat,
                  const tString &in asHeaderEntry,
                  const tString &in asTextCat,
                  const tString &in asTextEntry,
                  bool abIsInputHint,
                  const cColor &in aColor,
                  float afTimeMul,
                  bool abAddAsGiven)
```

---

**Hint_Release**

```
void Hint_Release(int alID)
```

---

**Hint_StopHint**

```
void Hint_StopHint()
```

---

#### Hint_AddAsGiven

```
void Hint_AddAsGiven(int alID)
```

---

#### Hint_AddAsGiven

```
void Hint_AddAsGiven(const tString &in asCat,
                     const tString &in asEntry)
```

---

#### Infection_CreateBorderTree

```
void Infection_CreateBorderTree(int alBorder,
                                float afOffset,
                                const cVector2f &in avSize,
                                float afSpeed)
```

Creates an infection tree at a border of the screen

---

#### Infection_ClearTrees

```
void Infection_ClearTrees()
```

Clears all created trees.

---

#### PlayerEnergy_GetMoveMul

```
float PlayerEnergy_GetMoveMul()
```

---

#### PlayerEnergy_GetLookMul

```
float PlayerEnergy_GetLookMul()
```

---

#### PlayerEnergy_SetFlowerSwallows

```
void PlayerEnergy_SetFlowerSwallows(bool abX)
```

---

#### PlayerEnergy_SetAllowCathComment

```
void PlayerEnergy_SetAllowCathComment(bool abX)
```

---

#### Inventory_AutoEnable

```
void Inventory_AutoEnable(float afTime)
```

This really means 'show the inventory!' but too late to rename now. 

---

#### Inventory_SetEnabled

```
void Inventory_SetEnabled(bool abEnabled)
```

Disables/enables the 'show inventory' key.

---

#### PlayerHands_SetHandModel

```
void PlayerHands_SetHandModel(const tString &in asFile)
```

---

#### PlayerHands_SetHandModel_Human

```
void PlayerHands_SetHandModel_Human()
```

---

#### PlayerHands_SetHandModel_Diving

```
void PlayerHands_SetHandModel_Diving()
```

---

**PlayerHands_SetHandModel_DeepSea**

```
void PlayerHands_SetHandModel_DeepSea()
```

---

**PlayerHands_SetHandModel_DeepSeaMutilated**

```
void PlayerHands_SetHandModel_DeepSeaMutilated()
```

---

**PlayerHands_PreloadHandModel**

```
void PlayerHands_PreloadHandModel(const tString &in asFile)
```

---

**PlayerHands_PreloadHandModel_Human**

```
void PlayerHands_PreloadHandModel_Human()
```

---

**PlayerHands_PreloadHandModel_Diving**

```
void PlayerHands_PreloadHandModel_Diving()
```

---

**PlayerHands_PreloadHandModel_DeepSea**

```
void PlayerHands_PreloadHandModel_DeepSea()
```

---

**PlayerHands_PreloadHandModel_DeepSeaMutilated**

```
void PlayerHands_PreloadHandModel_DeepSeaMutilated()
```

---

**PlayerHands_PlayAnimation**

```
void PlayerHands_PlayAnimation(const tString &in asAnim,
                               bool abLoop=false,
                               bool abFullScaleModel=false,
                               float afFadeTime=0.0f,
                               const tString &in asAttachedProp="",
                               bool abDisableWhenOver=true,
                               float afSpeed=1,
                               float afRelTimePos=-1)
```

---

#### PlayerHands_GetAnimationPlaying

```
bool PlayerHands_GetAnimationPlaying()
```

---

#### PlayerHands_SetActive

```
void PlayerHands_SetActive(bool abX)
```

---

#### PlayerHands_IsActive

```
bool PlayerHands_IsActive()
```

---

#### PlayerHands_SetUseCustomRotation

```
void PlayerHands_SetUseCustomRotation(bool abX)
```

---

#### PlayerHands_SetCustomRotation

```
void PlayerHands_SetCustomRotation(const cVector3f &in avRot)
```

---

#### PlayerHands_SetCustomRotationFromEntity

```
void PlayerHands_SetCustomRotationFromEntity(const tString &in asEntityName)
```

#### PlayerHands_SetUseCustomPosition

```
void PlayerHands_SetUseCustomPosition(bool abX)
```

#### PlayerHands_SetCustomPosition

```
void PlayerHands_SetCustomPosition(const cVector3f &in avPos,
                                   bool abUseBasicOffset)
```

#### PlayerHands_SetCustomPositionFromEntity

```
void PlayerHands_SetCustomPositionFromEntity(const tString &in asEntityName)
```

#### PlayerHands_GetBoneState

```
cBoneState PlayerHands_GetBoneState(const tString &in asBone)
```

#### PlayerHands_GetSocket

```
cNode3D PlayerHands_GetSocket(const tString &in asSocket)
```

#### PlayerHands_GetCurrentAnimationState

```
cAnimationState PlayerHands_GetCurrentAnimationState()
```

#### PlayerHands_AttachCameraToSocket

```
void PlayerHands_AttachCameraToSocket(const tString &in asBoneName,
                                      float afFadeTime,
                                      const tString &in asFadeOverCallback,
                                      bool abDisablePlayer=false,
```

```
                                        bool abAutoDetach=false,
                                        float afDetachTime=1.0f,
                                        const cVector3f &in avPosOffset=,
                                        const cVector3f &in avRotOffset=)
```

---

#### PlayerHands_DetachCameraFromSocket

```
void PlayerHands_DetachCameraFromSocket(float afFadeTime)
```

---

#### PlayerHands_SetAnimationOverCallback

```
void PlayerHands_SetAnimationOverCallback(const tString &in asFunction)
```

---

#### PlayerHands_SetVisible

```
void PlayerHands_SetVisible(bool abX)
```

---

#### PlayerHands_GetEntityName

```
tString PlayerHands_GetEntityName()
```

---

#### LoadScreen_SetBackground

```
void LoadScreen_SetBackground(const tString &in asTexture)
```

Sets the background texture used by the loading screen

---

#### LoadScreen_SetForceBackground

```
void LoadScreen_SetForceBackground(bool abX)
```

Sets if you want to force showing the background.

#### LoadScreen_SetIcon

```
void LoadScreen_SetIcon(const tString &in asFirstFrame,
                        int alFrameNum)
```

Sets the icon used by the loading screen when loading or saving

---

#### LoadScreen_SetUseSmallIcon

```
void LoadScreen_SetUseSmallIcon(bool abX)
```

If fullscreen icon should be used or if it should be shown in the bottom right corner

---

#### LoadScreen_IsVisible

```
bool LoadScreen_IsVisible()
```

---

#### LoadScreen_ShowLoadingIcon

```
void LoadScreen_ShowLoadingIcon(float afTimeMax=10.0f)
```

---

#### Credits_Start

```
void Credits_Start(const tString &in asCreditsFile,
                   const tString &in asCompleteCallback="",
                   float afRollSpeedStart=0.00f,
                   float afRollSpeedGoal=0.008f,
                   float afRollSpeedFadeTime=1.0f)
```

Starts the credits screen

---

#### Credits_SetRollSpeed

```
void Credits_SetRollSpeed(float afGoal,
                          float afFadeTime)
```

Changes rolling speed

---

#### Credits_Stop

```
void Credits_Stop(float afFadeTime=0.0f)
```

Dismisses the credits screen

---