helper_map.hps

Helper functions for dealing with map related stuff

Map_GetTimeStamp

```
int Map_GetTimeStamp()
```

This gets the number of ticks (60/second is default) since start of map.

Map_GetElapsedTime

```
float Map_GetElapsedTime(int alTimeStamp)
```

Gets how long time it has elapsed since the time step was retrieved.

- **alTimeStamp**: value gotten from Map_GetTimeStamp

Map_SetDisplayNameEntry

```
void Map_SetDisplayNameEntry(const tString &in asNameEntry)
```

Sets the displayed name of the current map.

- **asNameEntry**: new name of the map.

Map_SetSkyBoxActive

```
void Map_SetSkyBoxActive(bool abX)
```

Activates or deactivates the skybox of the current map

- **abX**: true = activate - false = deactivate.

Map_SetSkyBoxTexture

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

Sets the texture of the skybox in the current map

- **asTexture**: file name of the texture to set.

Map_SetSkyBoxColor

```
void Map_SetSkyBoxColor(const cColor &in acColor)
```

Sets the color of the skybox in the current map

- **acColor**: color to set.

Map_SetFogActive

```
void Map_SetFogActive(bool abX)
```

Activates or deactivates fog in the current map

- **abX**: true = activate - false = deactivate.

Map_SetFogProperties

```
void Map_SetFogProperties(float afStart,
                          float afEnd,
                          float afFalloffExp,
                          const cColor &in acColor,
                          bool abCulling)
```

Sets the properties of the fog in the current map

- **afStart**: start distance of fog.
- **afEnd**: distance where the fog effect is maxed out.
- **afFalloffExp**: rate at which the fog effect will decrease over the distance between afStart and afEnd.
- **acColor**: color of fog.
- **abCulling**: true = objects hidden by fog are not rendered - false = fog has no effect on whether objects are rendered or not.

Map_SetSecondaryFogActive

```
void Map_SetSecondaryFogActive(bool abX)
```

Activates or deactivates the secondary fog

- **abX**: true = activate - false = deactivate.

Map_SetSecondaryFogProperties

```
void Map_SetSecondaryFogProperties(float afStart,
                                   float afEnd,
                                   float afFalloffExp,
                                   const cColor &in acColor)
```

Sets the properties of the secondary fog, both the primary and secondary fog will be applied, these are order independent

- **afStart**: start distance of fog.
- **afEnd**: distance where the fog effect is maxed out.
- **afFalloffExp**: rate at which the fog effect will decrease over the distance between afStart and afEnd.
- **acColor**: color of fog.

Map_FadeFogStart

```
void Map_FadeFogStart(float afTarget,
                      float afTime)
```

Map_FadeFogEnd

```
void Map_FadeFogEnd(float afTarget,
                    float afTime)
```

Map_FadeFogColor

```
void Map_FadeFogColor(const cColor &in aTarget,
                      float afTime)
```

Map_FadeFogFalloffExp

```
void Map_FadeFogFalloffExp(float afTarget,
                           float afTime)
```

Map_SetEnvironmentParticlesActive

```
void Map_SetEnvironmentParticlesActive(bool abState)
```

Map_GetEnvironmentParticles

```
array < cEnvironmentParticles @> Map_GetEnvironmentParticles(tString asName,
                                                             array<
cEnvironmentParticles @> &inout avParticles)
```

Get all env particles that match name

Map_SetEnvironmentParticleVisible

```
void Map_SetEnvironmentParticleVisible(tString asName,
                                       bool abVisible)
```

Toggle visiblity for specific layer

Map_SetEnvironmentParticleGravityVelocity

```
void Map_SetEnvironmentParticleGravityVelocity(tString asName,
                                               const cVector3f &in
avVelocity)
```

Sets the gravity of the env particles

Map_SetEnvironmentParticleRotateVelocity

```
void Map_SetEnvironmentParticleRotateVelocity(tString asName,
                                              const cVector3f &in
avVelocity)
```

Sets the rotation of the env particles

Map_SetEnvironmentParticleWindVelocity

```
void Map_SetEnvironmentParticleWindVelocity(tString asName,
                                            const cVector3f &in avVelocity)
```

Sets the wind of the env particles

Map_FadeEnvironmentParticleWindVelocity

```
void Map_FadeEnvironmentParticleWindVelocity(tString asName,
                                             const cVector3f &in avVelocity,
                                             float afTime)
```

Fade the wind velocity of all env particles over time

Map_FadeEnvironmentParticleColor

```
void Map_FadeEnvironmentParticleColor(tString asName,
                                      const cColor &in aColor,
                                      float afTime)
```

Fade the color of the env particle

Map_FadeEnvironmentParticleBrightness

```
void Map_FadeEnvironmentParticleBrightness(tString asName,
                                           float afBrightness,
                                           float afTime)
```

Fade the brightness of the env particle

Map_SetEnvironmentParticleColor

```
void Map_SetEnvironmentParticleColor(tString asName,
                                     const cColor &in aColor)
```

Fade the color of the env particle

Map_SetEnvironmentParticleFadeOut

```
void Map_SetEnvironmentParticleFadeOut(tString asName,
                                       float afStart,
                                       float afEnd)
```

Set the fade out settings for the environment particles.

Map_SetEnvironmentParticleFadeIn

```
void Map_SetEnvironmentParticleFadeIn(tString asName,
                                      float afStart,
                                      float afEnd)
```

Set the fade in settings for the environment particles.

Map_AddEnvironmentParticleClipArea

```
void Map_AddEnvironmentParticleClipArea(tString asName,
                                        tString asClipSourceEntity,
                                        bool abAdditive=true)
```

Sets the wind of the env particles

Map_RemoveEnvironmentParticleClipArea

```
void Map_RemoveEnvironmentParticleClipArea(tString asName,
```

```
                                                tString asClipSourceEntity)
```

Sets the wind of the env particles

Map_SetEnvironmentParticlesSpotLight

```
void Map_SetEnvironmentParticlesSpotLight(tString asName,
                                          cLightSpot @apLight,
                                          float afMul=1.0f)
```

Set flashlight to affect particles

Map_SetPlayerTerrainCollision

```
void Map_SetPlayerTerrainCollision(bool abX)
```

If player should collide with the terrain or not.

Map_SetEnvironmentParticleGravityVelocity

```
void Map_SetEnvironmentParticleGravityVelocity(const cVector3f &in
avVelocity)
```

Map_SetEnvironmentParticleRotateVelocity

```
void Map_SetEnvironmentParticleRotateVelocity(const cVector3f &in
avVelocity)
```

Map_SetEnvironmentParticleWindVelocity

```
void Map_SetEnvironmentParticleWindVelocity(const cVector3f &in avVelocity)
```

Map_FadeEnvironmentParticleWindVelocity

```
void Map_FadeEnvironmentParticleWindVelocity(const cVector3f &in avVelocity,
```

```
                                                    float afTime)
```

Map_Preset_SetupFog

```
void Map_Preset_SetupFog(tString asName,
                         bool abActive,
                         float afStart=,
                         float afEnd=,
                         float afFalloffExp=1,
                         cColor aColor=1,
                         float afBrightness=1,
                         bool abUnderwater=false,
                         bool abSkybox=false)
```

Setup fog for the specified preset

- **asName**: Name of the preset
- **abActive**: If fog should be active
- **afStart**: Where fog starts showing
- **afEnd**: Where fog has full effect
- **afFalloffExp**: How fast the fog fades to full effect, lower is faster
- **aColor**: Color of the fog
- **afBrightness**: Brightness of the fog
- **abUnderwater**: If underwater fog should be used
- **abSkybox**: If fog color should be taken from skybox

Map_Preset_SetupSecondaryFog

```
void Map_Preset_SetupSecondaryFog(tString asName,
                                  bool abActive,
                                  float afStart=,
                                  float afEnd=,
                                  float afFalloffExp=1,
                                  cColor aColor=1,
                                  float afBrightness=1)
```

Setup secondary fog for the specified preset

- **asName**: Name of the preset
- **abActive**: If fog should be active
- **afStart**: Where fog starts showing
- **afEnd**: Where fog has full effect
- **afFalloffExp**: How fast the fog fades to full effect, lower is faster
- **aColor**: Color of the fog

- **afBrightness**: Brightness of the fog

Map_Preset_SetupDepthOfField

```
void Map_Preset_SetupDepthOfField(tString asName,
                                  bool abActive,
                                  float afStart=,
                                  float afEnd=,
                                  float afFalloff=1)
```

Setup depth of field for specified preset

- **asName**: Name of the preset
- **abActive**: If DoF should be active
- **afStart**: Where near plane starts
- **afEnd**: Where far plane starts
- **afFalloff**: How fast the depth of field fades in

Map_Preset_SetupBloom

```
void Map_Preset_SetupBloom(tString asName,
                           float afWidth,
                           float afBrightPass,
                           float afFalloff,
                           cColor aTint)
```

Setup bloom for specified preset

- **asName**: Name of the preset
- **afWidth**: If bloom should be active
- **afBrightPass**: How bright
- **afFalloff**: Falloff of the bloom blur
- **aTint**: Color and brightness of bloom

Map_Preset_SetupDirLight

```
void Map_Preset_SetupDirLight(tString asName,
                              bool abActive,
                              cColor aDiffuseColor,
                              float afBrightness,
                              cVector3f avDirection,
                              cColor aGroundColor,
                              cColor aSkyColor)
```

Setup Direcitonal light for specified preset

- **asName**: Name of the preset
- **abActive**: If dir light should be active
- **aDiffuseColor**: Base color of the light
- **afBrightness**: How bright
- **avDirection**: Direction of the light
- **aGroundColor**: Ground color of the ambient light
- **aSkyColor**: Sky color of the ambient light

Map_Preset_SetupToneMapping

```
void Map_Preset_SetupToneMapping(tString asName,
                                 float afToneMappingKey,
                                 float afToneMappingExposure,
                                 float afToneMappingWhiteCut)
```

Setup tonemapping for specified preset

- **asName**: Name of the preset
- **afToneMappingKey**: Multiplier of the light
- **afToneMappingExposure**: Brightness of the tonemapping, value is in log2
- **afToneMappingWhiteCut**: Colors brighter than this value will be clamped to white

---

Map_Preset_SetupColorGrading

```
void Map_Preset_SetupColorGrading(tString asName,
                                  tString asColorGrading)
```

Setup color grading for specified preset

- **asName**: Name of the preset
- **asColorGrading**: Texture to use

Map_Preset_SetupSkybox

```
void Map_Preset_SetupSkybox(tString asName,
                            const cColor &in aColor,
                            float afBrightness)
```

Setup skybox for specified preset

- **asName**: Name of the preset
- **aColor**: Skybox color
- **afBrightness**: Brightness for the skybox color

Map_Preset_Fade

```
void Map_Preset_Fade(tString asName,
                     float afFadeTime)
```

Fade in the preset

- **asName**: Name of the preset
- **afFadeTime**: How long it should take to fade in

Map_CopyGlobalSettings

```
bool Map_CopyGlobalSettings(cLuxMap @apMap,
                            float afFadeTime)
```

Copies and fades in settings of another map

- **apMap**: Map to copy from
- **afFadeTime**: How long it should take to fade in

Map_SetDistanceCullingRange

```
void Map_SetDistanceCullingRange(float afRange,
                                 float afRandomSize=)
```

Actives distance culling and sets the min and max range to the value. Any objects further away then that range will get culled Only use this function on maps that dont have distance culling active from the .map file

- **afRange**: any object further away than this will get culled
- **afRandomSize**: adds a little random distance to each object, so that not all objects are culled at exactly the same distance

---

Map_DisableDistanceCulling

```
void Map_DisableDistanceCulling()
```

Disables distance culling

Map_SetUnderwater

```
void Map_SetUnderwater(bool abX,
                       bool abUseStartAndEndEffects=false)
```

Sets if the map is underwater or not, and the correct properties for the state.

- **abX**: if the map should be underwater or not.
- **abUseStartAndEndEffects**: if special effect should play at start or end.

Map_GetUnderwater

```
bool Map_GetUnderwater()
```

Gets if the map is under water or not.

Returns: true if the map is underwater.

Map_AddTimer

```
void Map_AddTimer(const tString &in asName,
                  float afTime,
                  const tString &in asFunction)
```

Syntax for callback function, void FunctionName(const tString &in asTimer). syntax for the callback function code snippet, clbTimer.

Map_RemoveTimer

```
void Map_RemoveTimer(const tString &in asName)
```

Remove a created timer.

- **asName**: name of the timer to remove. (if called inside a timer callback, the parent timer will be chosen in case there are many with the same name)

Map_GetTimerTime

```
float Map_GetTimerTime(const tString &in asName)
```

Get the current time of a created timer.

- **asName**: name of the timer. (if called inside a timer callback, the parent timer will be chosen in case there are many with the same name)

Map_TimerExists

```
bool Map_TimerExists(const tString &in asName)
```

Get if the specified timer exists.

- **asName**: name of the timer. (if called inside a timer callback, the parent timer will be chosen in case there are many with the same name)

Returns: bool, true if the timer exists.

Map_SetTimerPaused

```
void Map_SetTimerPaused(const tString &in asName,
                        bool abX)
```

Pause/unpause a created timer.

- **asName**: name of the timer to remove. (if called inside a timer callback, the parent timer will be chosen in case there are many with the same name)
- **abX**: true = pause - false = unpause.

Map_TimeHasPassed

```
bool Map_TimeHasPassed(const tString &in asName,
                       float afTime)
```

Checks if a timer with the specified name has reached 0, and if so recreates the timer with the specified time and returns true. Use in if-cases to limit how often the script within the case can run.

- **asName**: name of the timer to check/create. (if called inside a timer callback, the parent timer will be chosen in case there are many with the same name)

- **afTime**: time to set the timer to if created.

Returns: bool, true if the time has passed and a new timer was created, otherwise false.

Map_RestartCurrentTimer

```
void Map_RestartCurrentTimer(float afNewTime=-1)
```

If used inside a timer callback it will restart that the timer that issued that callback.

- **afNewTime**: The time for the new timer to make a callback, if <0 then previous time is used.

Map_SetTimerUserVarFloat

```
void Map_SetTimerUserVarFloat(const tString &in asName,
                              float afX)
```

Map_SetTimerUserVarInt

```
void Map_SetTimerUserVarInt(const tString &in asName,
                            int alX)
```

Map_SetTimerUserVarString

```
void Map_SetTimerUserVarString(const tString &in asName,
                               const tString &in asX)
```

Map_GetTimerUserVarFloat

```
float Map_GetTimerUserVarFloat(const tString &in asName)
```

Map_GetTimerUserVarInt

```
int Map_GetTimerUserVarInt(const tString &in asName)
```

Map_GetTimerUserVarString

```
tString Map_GetTimerUserVarString(const tString &in asName)
```

Map_IncTimerUserVarFloat

```
float Map_IncTimerUserVarFloat(const tString &in asName,
                               int afX)
```

Map_IncTimerUserVarInt

```
int Map_IncTimerUserVarInt(const tString &in asName,
                           int alX)
```

Map_GetEntity

```
iLuxEntity Map_GetEntity(const tString &in asName,
                         eLuxEntityType aType=eLuxEntityType_LastEnum,
                         const tString &in asClass="")
```

Returns the entity with the given name.

- **asName**: the name of the entity.
- **aType**: (optional), the type of entity (as an enum).
- **asClass**: (optional), the class of the entity.

Returns: iLuxEntity @, the requested entity, or null if the entity was not found.

Map_GetEntityArray

```
bool Map_GetEntityArray(const tString &in asName,
                        array< iLuxEntity @> &inout avOutEntities,
                        eLuxEntityType aEntityType=eLuxEntityType_LastEnum)
```

Returns an array of entities with a given name.

- **asName**: name of entities. May contain * as wildcards.
- **avOutEntities**: reference to array that will be filled with lights.
- **aEntityType**: type of entity to filter by - eLuxEntityType_LastEnum for all types

Returns: true if any entities found

Map_GetProp

```
cLuxProp Map_GetProp(const tString &in asName,
                     const tString &in asClass="")
```

Returns the prop with the given name.

- **asName**: the name of the prop.
- **asClass**: (optional), the class of the prop.

Returns: cLuxProp @, the requested prop, or null if the prop was not found.

Map_GetLight

```
iLight Map_GetLight(const tString &in asName)
```

Returns a light with the given name.

- **asName**: name of light

Returns: iLight @, the requested light

Map_GetSoundEntityArray

```
array < cSoundEntity @> Map_GetSoundEntityArray(const tString &in asName,
                                                array< cSoundEntity @>
&inout avOutSoundEntities)
```

Creates an array of sound entities with a given name.

- **asName**: name of the sound entities. May contain * as wildcards.
- **avOutSoundEntities**: reference to array that will be filled with sounds.

Returns: array< cSoundEntity >, array of sound entities found.

Map_GetEntityID

```
tID Map_GetEntityID(const tString &in asName,
                    eLuxEntityType aType=eLuxEntityType_LastEnum,
                    const tString &in asClass="")
```

Returns the entity with the given name.

- **asName**: the name of the entity.
- **aType**: (optional), the type of entity (as an enum).
- **asClass**: (optional), the class of the entity.

Returns: tID , the id to the prop, tID_Invalid returned if not found

---

Map_GetPropID

```
tID Map_GetPropID(const tString &in asName,
                  const tString &in asClass="")
```

Returns the prop with the given name.

- **asName**: the name of the prop.
- **asClass**: (optional), the class of the prop.

Returns: tID , the id to the prop, tID_Invalid returned if not found

---

Map_GetLensFlareIDArray

```
array < tID > Map_GetLensFlareIDArray(const tString &in asName,
                                      array< tID > &inout avOutLensFlares)
```

Creates an array of lens flares with a given name.

- **asName**: name of lens flares. May contain * as wildcards.
- **avOutLensFlares**: reference to array that will be filled with lens flares.

Returns: array<tID>, array of id to lens flares found.

Map_GetSoundEntityIDArray

```
array < tID > Map_GetSoundEntityIDArray(const tString &in asName,
```

```
                                          array< tID > &inout
avOutSoundEntities)
```

Creates an array of sound entities with a given name.

- **asName**: name of the sound entities. May contain * as wildcards.
- **avOutSoundEntities**: reference to array that will be filled with lights.

Returns: array<tID>, array of id to sound entities found.

Map_GetParticleSystemIDArray

```
array < tID > Map_GetParticleSystemIDArray(const tString &in asName,
                                           array< tID > &inout
avOutParticles)
```

Creates an array of particle systems with a given name.

- **asName**: name of particle systems. May contain * as wildcards.
- **avOutParticles**: reference to array that will be filled with particle systems.

Returns: array<tID>, array of id to particle systems found.

Map_ChangeMap

```
void Map_ChangeMap(const tString &in asMapName,
                   const tString &in asStartPos,
                   const tString &in asStartSound,
                   const tString &in asEndSound)
```

Changes the active map to the one specified.

- **asMapName**: map to change to.
- **asStartPos**: name of the player start node that the player should begin at.
- **asStartSound**: sound that plays before loading the map.
- **asEndSound**: sound that plays as the new map starts.

Map_ChangeMap

```
void Map_ChangeMap(const tString &in asMapName,
                   const tString &in asStartPos,
                   const tString &in asTransferArea,
```

```
                    const tString &in asStartSound,
                    const tString &in asEndSound)
```

Changes the active map to the one specified.

- **asMapName**: map to change to.
- **asStartPos**: name of the player start node that the player should begin at.
- **asTransferArea**: name of the area used to transfer objects (and the player) from one map to another
- **asStartSound**: sound that plays before loading the map.
- **asEndSound**: sound that plays as the new map starts.

Map_Preload

```
void Map_Preload(const tString &in asMapName,
                 eWorldStreamPriority aPrio=eWorldStreamPriority_Normal)
```

Starts loading in assets for the next map

- **asMapName**: Map to load data for
- **aPrio**: How intrusive the preload is

Map_SetPreloadPriority

```
void Map_SetPreloadPriority(eWorldStreamPriority aPrio)
```

Prioirty of background download thread

- **aPrio**: How intrusive the preload is

Map_GetPreloadMap

```
cLuxMap Map_GetPreloadMap()
```

Returns the map that is currently being preloaded

Map_Deload

```
void Map_Deload(const tString &in asTransferArea="")
```

Deloads the objects in the current map

- **asTransferArea**: Objects inside this area will not be deloaded

---

Map_FadeOut

```
void Map_FadeOut(float afTime)
```

Fades out the screen and all the sounds. Should be called some second before calling change map

- **afTime**: time to fade out over

Map_IsChanging

```
bool Map_IsChanging()
```

Returns if a map change is occuring

Map_IsPreloading

```
bool Map_IsPreloading()
```

Returns if a map is being streamed in

Map_IsPreloadCompleted

```
bool Map_IsPreloadCompleted()
```

Returns if the map has been fully preloaded

Map_SetInteractionWhiteListActive

```
void Map_SetInteractionWhiteListActive(bool abActive,
                                       bool abClearList)
```

Disables interaction on all entities that are not on the white list

- **abActive**: if the white list should be used

- **abClearList**: if the white list should be reset

Map_AddEntityToInteractionWhiteList

```
void Map_AddEntityToInteractionWhiteList(const tString &in asName)
```

Adds an entity to the white list, meaning that is can be intereacted with

- **asName**: Name of the entity

Map_CheckLineOfSight

```
void Map_CheckLineOfSight(const tString &in asCallbackFunc,
                          const cVector3f &in avStart,
                          const cVector3f &in avEnd,
                          bool abCheckOnlyShadowCasters,
                          bool abCheckOnlyStatic)
```

Checks if the line of sight between two positions is clear

- **asCallbackFunc**: callback function to call the result, syntax: void func(bool abClear)
- **avStart**: origin of the check
- **avEnd**: end of the check
- **abCheckOnlyShadowCasters**: if only shadow casters should be checked
- **abCheckOnlyStatic**: if only static bodies should be checked

---

Map_GetClosestEntity

```
void Map_GetClosestEntity(const tString &in asCallbackFunc,
                          const cVector3f &in avStart,
                          const cVector3f &in avDir,
                          float afRayLength,
                          int alInteractType,
                          bool abCheckLineOfSight)
```

Returns the closest entity in direction

- **asCallbackFunc**: callback function to call the result, syntax: void func(bool abSuccesful, float afDistance,
- **avStart**: origin of the check
- **avDir**: direction to check in
- **afRayLength**: max distance to cast ray
- **alInteractType**: interaction type
- **abCheckLineOfSight**: if the entity has to be in view

Map_GetClosestBody

```
void Map_GetClosestBody(const tString &in asCallbackFunc,
                        const cVector3f &in avStart,
                        const cVector3f &in avDir,
                        float afRayLength)
```

Returns the closest body in direction

- **asCallbackFunc**: callback function to call the result, syntax: void func(bool abSuccesful, float afDistance, const
- **avStart**: origin of the check
- **avDir**: direction to check in
- **afRayLength**: max distance to cast ray

Map_GetClosestCharCollider

```
void Map_GetClosestCharCollider(const tString &in asCallbackFunc,
                                const cVector3f &in avStart,
                                const cVector3f &in avDir,
                                float afRayLength,
                                bool abCheckDynamic)
```

Returns the closest physics body that can collide with a character

- **asCallbackFunc**: callback function to call the result, syntax: void func(bool abSuccesful, float afDistance, const
- **avStart**: origin of the check
- **avDir**: direction to check in
- **afRayLength**: max distance to cast ray
- **abCheckDynamic**: if both dynamic and static physics should be checked against

---

Map_GetLightLevelAtPos

```
void Map_GetLightLevelAtPos(const tString &in asCallbackFunc,
                            const cVector3f &in avPos,
                            iLight @apSkipLight=null,
                            float afRadiusAdd=0.25)
```

Get how bright the position of the map is

- **asCallbackFunc**: callback function to call the result, syntax: void func(float afLightLevel)

- **avPos**: position to check
- **apSkipLight**: light not to use in the check
- **afRadiusAdd**: additional radius to search to simulate a area instead of a point

---

Visibility_SetAreaActive

```
void Visibility_SetAreaActive(const tString &in asName,
                              bool abActive)
```

Set if visibility area is active (visible) or not.

- **asName**: Name of the entity, Wildcard(s) * are supporeed
- **abActive**: If true, area will become active

Visibility_SetMainActive

```
void Visibility_SetMainActive(bool abActive)
```

Set if the main visiblity area should be active

Visibility_SetTerrainActive

```
void Visibility_SetTerrainActive(bool abActive)
```

Set if the terrain should be visible area should be active

Decal_SetVisibleInArea

```
void Decal_SetVisibleInArea(bool abVisible,
                            const cVector3f &in avMin,
                            const cVector3f &in avMax,
                            array< tString > avMaterials)
```

Sets visiblity of all decals with material in the area

- **abVisible**: If visible or not
- **avMin**: Min of area
- **avMax**: Max of area
- **avMaterials**: Array of decals with materials to affect

Decal_SetVisibleInArea

```
void Decal_SetVisibleInArea(bool abVisible,
                            const tString &in asArea,
                            array< tString > avMaterials)
```

Sets visiblity of all decals with material in the area

- **abVisible**: If visible or not
- **asArea**: Name of the area that the decals has to intersect with
- **avMaterials**: Array of decals with materials to affect

Decal_SetVisibleInArea

```
void Decal_SetVisibleInArea(bool abVisible,
                            const tString &in asArea,
                            tString asMaterial)
```

Sets visiblity of all decals with material in the area

- **abVisible**: If visible or not
- **asArea**: Name of the area that the decals has to intersect with
- **asMaterial**: Only decals with this material will be affected

---