helper_effects.hps

Effect_Shake_Start

```
int Effect_Shake_Start(float afAmount,
                       float afTime,
                       float afFadeInTime,
                       float afFadeOutTime,
                       const cVector3f &in avDirAmount=cVector3f(1,
                       1,
                       1),
                       float afFrequency=)
```

Makes the screen shake for a certain amount of time. Returns id to the current shaking. If several calls to shake has been made the largest one is used.

- **afAmount**: The strength of the shake.
- **afTime**: The time it shakes at full intensity (fade in and fade out are added to this).
- **afFadeInTime**: The time before it reaches full strenght.
- **afFadeOutTime**: The time, after afTime is up, that it will take for the shake to reach zero strength.
- **avDirAmount**: The amount it will shake in each direction in camera space
- **afFrequency**: Times per seconds the shake will change offset. 0=every update (ie max).

Effect_Shake_FadeOut

```
void Effect_Shake_FadeOut(int alID,
                          float afFadeOutTime)
```

Fades out one or all shake instances.

- **alID**: The id of the shake (returned by Effect_Shake_Start). -1 = fade out all active shakes.
- **afFadeOutTime**: the time it takes to fade out.

Effect_Shake_SetSourceEntity

```
void Effect_Shake_SetSourceEntity(int alID,
                                  const tString &in asSourceEntity,
                                  float afMinDist,
                                  float afMaxDist,
                                  eEasing aEasingType=eEasing_CubicIn)
```

Sets an entity as the source of a shake instance. The shake instance will increase in strength as the player gets closer to the entity.

- **alID**: The id of the shake (returned by Effect_Shake_Start).
- **asSourceEntity**: entity to be used as source.
- **afMinDist**: distance to entity where the shake will be at its most powerful.
- **afMaxDist**: distance where shake will have faded out completely.
- **aEasingType**: the curve the strength of the shake will follow as you move further from the entity.

Effect_Shake_GetCurrentShakeAmount

```
float Effect_Shake_GetCurrentShakeAmount()
```

Gets the current amount of shake (as in abs larges element in shake vector)

Effect_Shake_EnableAll

```
void Effect_Shake_EnableAll(bool abState)
```

Effect_VideoDistortion_EnableAll

```
void Effect_VideoDistortion_EnableAll(bool abState)
```

Effect_Fade_In

```
void Effect_Fade_In(float afTime)
```

Effect_Fade_Out

```
void Effect_Fade_Out(float afTime)
```

Effect_Fade_IsFading

```
bool Effect_Fade_IsFading()
```

Effect_Fade_GetAlpha

```
float Effect_Fade_GetAlpha()
```

Effect_Glow_AddEntity

```
void Effect_Glow_AddEntity(iLuxEntity @apEntity,
                           float afAlpha,
                           float afY)
```

Adds a glow effect to the object to make it more visible to the player Must be called from OnPostUpdate() every frame the glow should be visible

- **apEntity**: entity to glow
- **afAlpha**: how much it should glow, [0, 1]
- **afY**: The position of the the glow.

Effect_Bloom_SetActive

```
void Effect_Bloom_SetActive(bool abX)
```

Sets if bloom should be active Bloom is an effect that makes bright colors glow in a halo around the object

- **abX**: if bloom should be active

Effect_Bloom_FadeBrightPass

```
void Effect_Bloom_FadeBrightPass(float afBrightPass,
                                 float afTime)
```

Changes the bright pass of the bloom over time

- **afBrightPass**: how bright a color has to be to be able to bloom, this should be a values between 0-1
- **afTime**: time to fade

Effect_Bloom_FadeBloomWidth

```
void Effect_Bloom_FadeBloomWidth(float afBloomWidth,
                                 float afTime)
```

Changes the width of the bloom over time

- **afBloomWidth**: the width in pixels of the bloom halo, works best with a value between 32-512 pixels
- **afTime**: time to fade

Effect_Bloom_FadeBloomFalloff

```
void Effect_Bloom_FadeBloomFalloff(float afBloomFalloff,
                                   float afTime)
```

Changes the sharpness of the bloom over time

- **afBloomFalloff**: how sharp the bloom should be, higher = sharper, 0.5 default
- **afTime**: time to fade

Effect_Bloom_FadeBloomTint

```
void Effect_Bloom_FadeBloomTint(float afR,
                                float afG,
                                float afB,
                                float afTime)
```

Changes the color of the bloom over time

- **(afR**: the color to tint the bloom in
- **afTime**: time to fade

Effect_FilmGrain_SetActive

```
void Effect_FilmGrain_SetActive(bool abX)
```

Sets if film grain should be active Film Grain is an effect that applies animated noise to the image to remove banding and give life to it

- **abX**: if film grain should be active

Effect_FilmGrain_FadeFilmGrainIntensity

```
void Effect_FilmGrain_FadeFilmGrainIntensity(float afIntensity,
                                             float afTime)
```

Changes the amount of noise applied to the screen over time

- **afIntensity**: the amount of noise applied to the screen
- **afTime**: time to fade in

Effect_DoF_Start

```
int Effect_DoF_Start(float afFocusStart,
                     float afFocusEnd,
                     float afFalloff,
                     float afTime)
```

Starts an instance of depth of field and returns the id

- **afFocusStart**: near plane of the instance
- **afFocusEnd**: far plane of the instance
- **afFalloff**: how smooth the transition from sharp to blurr should be
- **afTime**: time to fade to this value

Effect_DoF_FadeFocus

```
void Effect_DoF_FadeFocus(int alID,
                          float afFocusStart,
                          float afFocusEnd,
                          float afTime)
```

Fade the start and end of the focus plane

- **alID**: id of the instance
- **afFocusStart**: near plane of the instance
- **afFocusEnd**: far plane of the instance
- **afTime**: time to fade to this value

Effect_DoF_FadeFalloff

```
void Effect_DoF_FadeFalloff(int alID,
```

```
                                float afFocusFalloff,
                                float afTime)
```

Fade the falloff of the instance

- **alID**: id of the instance
- **afFocusFalloff**: how smooth the transition from sharp to blurr should be
- **afTime**: time to fade in

Effect_DoF_FadeOut

```
void Effect_DoF_FadeOut(int alID,
                        float afTime)
```

Fade out the instance of depth of field, the instance is removed after fading out

- **alID**: id of the instance
- **afTime**: time to fade out

Effect_DoF_Reset

```
void Effect_DoF_Reset()
```

Effect_ToneMapping_UseSRGB

```
void Effect_ToneMapping_UseSRGB(bool abX)
```

If srgb gamma should be used. Default is normal pow(x, 1 / gamma)

- **abX**: If srgb gamma correction should be used

Effect_ToneMapping_FadeExposure

```
void Effect_ToneMapping_FadeExposure(float afExposure,
                                     float afTime)
```

Changes the overall brightness of the image over time

- **afExposure**: how much extra light should be let though the lens, 0 is default
- **afTime**: time to fade in

Effect_ToneMapping_GetExposure

```
float Effect_ToneMapping_GetExposure()
```

Get the current exposure value of the viewport

Returns: , current exposure

Effect_ToneMapping_GetExposureBrightness

```
float Effect_ToneMapping_GetExposureBrightness()
```

Get the current exposure value of the viewport

Returns: , current brightness

Effect_ToneMapping_FadeWhiteCut

```
void Effect_ToneMapping_FadeWhiteCut(float afWhiteCut,
                                     float afTime)
```

Changes the white point of the tone mapper over time Every color brighter then the white cut gets clamped to (1,1,1)

- **afWhiteCut**: the value of the white cut, every brighter color get set to white
- **afTime**: time to fade in

Effect_ToneMapping_FadeGrading

```
void Effect_ToneMapping_FadeGrading(tString asTextureName,
                                    float afTime)
```

Fades in a grading texture which changes the final color of the image

- **asTextureName**: name of the grading texture
- **afTime**: time to fade in

Effect_ImageTrail_Start

```
int Effect_ImageTrail_Start(float afAmount,
                            float afFadeInTime,
                            float afStayTime,
                            float afFadeOutTime)
```

Image trail blends the image of multiple previous frames over time, multiple instances can be active at the same time The strongest instance of image trail will be used

- **afAmount**: how much of the previous frame should be blended, values from 0-inf, 0 = disabled
- **afFadeInTime**: how long it should take to fade to fade in
- **afStayTime**: how long it will stay before starting to fade out. If <0, then it stays for ever.
- **afFadeOutTime**: how long it will take to fade out

Returns: the id to this image trail instance

Effect_ImageTrail_SetDirectAmount

```
void Effect_ImageTrail_SetDirectAmount(float afAmount)
```

Image trail blends the image of multiple previous frames over time. (in case Effect_ImageTrail_Start has also been used, the strongest instance of image trail will be used

- **afAmount**: how much of the previous frame should be blended, values from 0-inf, 0 = disabled

---

Effect_ImageTrail_FadeOut

```
void Effect_ImageTrail_FadeOut(int alID,
                               float afFadeOutTime)
```

Fades out an active instance of image trail

- **alID**: id of the trail instance, returned by _Start
- **afFadeOutTime**: how long it will take to fade out

Effect_ImageTrail_Clear

```
void Effect_ImageTrail_Clear()
```

Clears all effects

Effect_ChromaticAberration_StartAnim

```
void Effect_ChromaticAberration_StartAnim(float afDuration,
                                          float afAmount,
                                          float afRandomness,
                                          cVector2f avDirection)
```

Split the image into three images depending on color and makes rotation animation.

- **afDuration**: how long the effect should be active for, multiple instances can be active at the same time
- **afAmount**: how far the split should go in screen space, values between 0-0.2 look good
- **afRandomness**: the randomness of the splitting, if it should sway when splitting up and going back down, values larger than 1.0 will cause full rotation
- **avDirection**: an additional direction that the screen should move in when splitting, 0 = random

---

Effect_ChromaticAberration_SetDirect

```
void Effect_ChromaticAberration_SetDirect(float afAmount,
                                          float afRotation,
                                          float afHue=,
                                          const cVector2f &in
avOffset=cVector2f())
```

Split the image into three images depending on color

- **afAmount**: The amount of the effect (how much seperation basically). Measured in part of screen. 0 - 0.2 looks good.
- **afRotation**: The rotation of the diffferent splits. 0 - 360
- **afHue**: Which use to split into, there is always a 120 seperation between each split, this sets. where to make these splits. 0 - 360
- **avOffset**: An extra offset.

Effect_ChromaticAberration_CreateInstance

```
int Effect_ChromaticAberration_CreateInstance(float afAmount,
                                              float afRotation,
                                              float afHue=,
                                              const cVector2f &in
avOffset=cVector2f())
```

Effect_ChromaticAberration_SetInstanceValues

```
void Effect_ChromaticAberration_SetInstanceValues(int alID,
                                                  float afAmount,
                                                  float afRotation,
                                                  float afHue=,
                                                  const cVector2f &in
avOffset=cVector2f_Zero)
```

Effect_ChromaticAberration_DestroyInstance

```
void Effect_ChromaticAberration_DestroyInstance(int alID)
```

Effect_Flash_Start

```
void Effect_Flash_Start(float afFadeIn,
                        float afWhite,
                        float afFadeOut)
```

Fades exposure in and out to create a white flash effect.

- **afFadeIn**: time to fade in.
- **afWhite**: time to remain at the max exposure.
- **afFadeOut**: time to fade out.

Effect_Screen_Start

```
int Effect_Screen_Start(const tString &in asMaterial,
                        cVector2f avPosition,
                        cVector2f avSize)
```

Creates a images that shows up on the screen, it must be faded manually Remember that these must work on both 4:3 and 16:9 aspect ratio

- **asMaterial**: material to use
- **avPosition**: position on the screen, values between [0,1]
- **avSize**: size of the texture

Returns: the id to this screen material

Effect_Screen_Start

```
int Effect_Screen_Start(const tString &in asMaterial,
                        cVector2f avPosition,
                        cVector2f avSize,
                        float afTargetAlpha,
                        float afFadeInTime,
                        float afStayTime,
                        float afFadeOutTime)
```

Creates a images that fade in and then out Remember that these must work on both 4:3 and 16:9 aspect ratio

- **asMaterial**: material to use
- **avPosition**: position on the screen, values between [0,1]
- **avSize**: size of the texture
- **afTargetAlpha**: alpha it should stop fading at
- **afFadeInTime**: how fast it should fade in
- **afStayTime**: how long it should stay after fading in
- **afFadeOutTime**: how fast it should fade out again

Returns: the id to this screen material

Effect_Screen_FadeOut

```
void Effect_Screen_FadeOut(int alID,
                           float afFadeTime)
```

Fades out a material

- **alID**: id of the screen material
- **afFadeTime**: time to fade out over

Effect_Screen_FadeLiquidAmount

```
void Effect_Screen_FadeLiquidAmount(int alID,
                                    float afAmount,
                                    float afFadeTime)
```

Fades the liquid amount of a material, this is a variable used by the shader to determine wetness

- **alID**: id of the screen material
- **afAmount**: liquid amount, value between [0,1]

- **afFadeTime**: time to fade over

Effect_Screen_FadeAlpha

```
void Effect_Screen_FadeAlpha(int alID,
                             float afAlpha,
                             float afFadeTime)
```

Fades the transparency of a material

- **alID**: id of the screen material
- **afAlpha**: transparency of the material value between [0,1]
- **afFadeTime**: time to fade over

Effect_RadialBlur_SetDirect

```
void Effect_RadialBlur_SetDirect(float afSize,
                                 float afStartDistance,
                                 float afAlpha)
```

Directly sets the values of radial blur (but will blend with any instnace that has been started)

- **afSize**: size of the blur, screen size, looks good 0.01-0.1
- **afStartDistance**: how far away from the center the blur should start
- **afAlpha**: how visible the blur is

Effect_RadialBlur_Start

```
int Effect_RadialBlur_Start(float afSize,
                            float afAlpha,
                            float afStartDistance,
                            float afTime)
```

Starts a radial blur effect, will be active until

- **afSize**: size of the blur, screen size, looks good 0.01-0.1
- **afAlpha**: how visible the blur is
- **afStartDistance**: how far away from the center the blur should start
- **afTime**: time to fade in

Returns: the id to this radial blur instance

Effect_RadialBlur_FadeSize

```
void Effect_RadialBlur_FadeSize(int alID,
                                float afSize,
                                float afTime)
```

Changes the size of the blur of this instance

- **alID**: id to the instance to change
- **afSize**: size of the blur
- **afTime**: fade time of the blur

Effect_RadialBlur_FadeStartDistance

```
void Effect_RadialBlur_FadeStartDistance(int alID,
                                         float afDistance,
                                         float afTime)
```

Changes how far away from the center to start blurring

- **alID**: id to the instance to change
- **afDistance**: distance from center [0-1]
- **afTime**: fade time

Effect_RadialBlur_FadeAlpha

```
void Effect_RadialBlur_FadeAlpha(int alID,
                                 float afAlpha,
                                 float afTime)
```

Changes how visible the instance should be

- **alID**: id to the instance to change
- **afAlpha**: how visible it should be [0-1]
- **afTime**: fade time

Effect_RadialBlur_FadeOut

```
void Effect_RadialBlur_FadeOut(int alID,
                               float afTime)
```

Fades out this instance and removes it. The ID is invalid after this has been called

- **alID**: id to the instance to fade out
- **afTime**: fade time

Effect_ImageFadeFX_SetAmount

```
void Effect_ImageFadeFX_SetAmount(float afX)
```

Effect_ImageFadeFX_FadeAmount

```
void Effect_ImageFadeFX_FadeAmount(float afAmount,
                                   float afTime)
```

Effect_ImageFadeFX_SetTextures

```
void Effect_ImageFadeFX_SetTextures(const tString &in asFadeTexture,
                                    const tString &in asColorTexture,
                                    const tString &in asOffsetTexture)
```

Effect_VideoDistortion_SetDirectAmount

```
void Effect_VideoDistortion_SetDirectAmount(float afX)
```

Sets the amount of video distortion directly. (if there are instances in play, then the max amount will be used)

- **afX**: The amount of the effect, 0 - 1 are valid values. (0 turns off)

Effect_VideoDistortion_Start

```
int Effect_VideoDistortion_Start(float afAmount,
                                 float afTime,
                                 float afFadeInTime,
                                 float afFadeOutTime,
                                 float afVolume=1.0f)
```

Starts the video distortion effect. If many have been started the max amount is used. Returns ID to this instnace.

- **afAmount**: The amount of effect, 0 - 1 are valid values. (0 turns off)
- **afTime**: The length the effect (after fade in) the effect should last. -1 means it lasts forever.
- **afFadeInTime**: The time it takes for the effect to fade in.
- **afFadeOutTime**: The time it takes for the effect to fade out.
- **afVolume**: Volume of gui sound (will override previous instances. Only one sound at a time)

Effect_VideoDistortion_FadeOut

```
void Effect_VideoDistortion_FadeOut(int alID,
                                    float afFadeOutTime)
```

Fades out a video distortion effect instance.

- **alID**: id to the instance to fade out. If -1, all instances will be faded.
- **afFadeOutTime**: The time it takes for the effect to fade out.

Effect_VideoDistortion_SetMaxAmount

```
void Effect_VideoDistortion_SetMaxAmount(int alID,
                                         float afMaxAmount)
```

Sets the max amount of an instnace (basically changes the amount)

- **alID**: id to the instance to fade out
- **afMaxAmount**: The new max amount to be used, 0 -1 are valid values.

Effect_VideoDistortion_GetAmount

```
float Effect_VideoDistortion_GetAmount()
```

Get the current amount of the distortion

Effect_VideoDistortion_GetEffectAmount

```
float Effect_VideoDistortion_GetEffectAmount()
```

Get the current effect amount of the distortion (the one that is sent to the post effect)

Effect_VideoDistortion_SetSoundEffectsDisabled

```
void Effect_VideoDistortion_SetSoundEffectsDisabled(bool abX)
```

Set if video distortion sounds should be disabled.

Effect_TimeGlitch_Start

```
void Effect_TimeGlitch_Start(float afTimeAdvancement,
                             float afPlayFastSoundTime,
                             float afShakeAmount,
                             bool abPlayEffectSound,
                             float afSoundEffectVolume=1.0f)
```

Starts an effect that makes physics and visual jump forward in time.

- **afTimeAdvancement**: The amount of time physics and visuals should advance.
- **afPlayFastSoundTime**: How long the sounds should run sped up. (sound freq depends on the time advancement + this time)
- **afShakeAmount**: How much the screen should shake.
- **abPlayEffectSound**: If an effect sound should play.
- **afSoundEffectVolume**: Volume of the sound being played.

Effect_Sway_FadeIn

```
void Effect_Sway_FadeIn(float afSize,
                        float afTime,
                        float afRate,
                        float afAngle,
                        float afInitialAmount=-1.0f,
                        float afYMul=0.25f,
                        float afDecay=0.995f,
                        float afMoveFactor=0.4f)
```

- **afSize**: max X size of sway
- **afTime**: fade in time
- **afRate**: how many sways per second
- **afAngle**: sway will be around a line on XZ plane - this sets the y-axis rotation of it (degrees : 0 = along X axis)
- **afInitialAmount**: how big is the initial sway size? (0.0f - 1.0f) If negative, has no effect.
- **afYMul**: multiplier of afSwaySize for y
- **afDecay**: how fast the swaying dies away (1.0f = never dies)

- **afMoveFactor**: how much of the player's move speed to factor in

---

Effect_Sway_SetDirect

```
void Effect_Sway_SetDirect(float afSize,
                           float afRate,
                           float afAngle,
                           float afInitialAmount=-1.0f,
                           float afYMul=0.25f,
                           float afDecay=0.995f,
                           float afMoveFactor=0.4f)
```

- **afSize**: max X size of sway
- **afRate**: how many sways per second
- **afAngle**: sway will be around a line on XZ plane - this sets the y-axis rotation of it (degrees : 0 = along X axis)
- **afInitialAmount**: how big is the initial sway size? (0.0f - 1.0f). If negative, has no effect.
- **afYMul**: multiplier of afSwaySize for y
- **afDecay**: how fast the swaying dies away (1.0f = never dies)
- **afMoveFactor**: how much of the player's move speed to factor in

---

Effect_Sway_FadeOut

```
void Effect_Sway_FadeOut(float afTime)
```

- **afTime**: How long it takes to fade out.

Effect_Sway_Stop

```
void Effect_Sway_Stop()
```

Effect_Sway_GetOffset

```
cVector3f Effect_Sway_GetOffset()
```

Effect_Rumble_Start

```
int Effect_Rumble_Start(float afAmount,
```

```
                              float afTime,
                              float afFadeInTime=,
                              float afFadeOutTime=)
```

Start a gamepad rumble

Effect_GamepadColor_Fade

```
int Effect_GamepadColor_Fade(cColor aColor,
                                float afTime)
```

Fade the gamepad color (the backlight of the ps4 controller)

Effect_Rumble_Stop

```
void Effect_Rumble_Stop(int alID)
```

Stops the rumble from this id

Effect_Rumble_SetScreenShakeMul

```
void Effect_Rumble_SetScreenShakeMul(float afX)
```

Sets a multiplier for how much rumble should be generated from screenshake.

- **afX**: multiplier to set. 1 is default.

Effect_GamepadColor_Stop

```
void Effect_GamepadColor_Stop(int alID)
```

Stops the color fade from this id
