Script Reference document

This document is not perfectly structured, sometimes a certain script function is perhaps not in the most logical position and so forth. It's recommended you search with the browser for what you need. Almost all scripts are used in Penumbra/Penumbra: Overture so it's always a good idea to locate a script in an actual level .hps file to see how it is used.

Engine scripts

General

```
void Print(std::string asText)
```

Prints a string to the log.

```
std::string FloatToString(float afX)
```

Converts a float to string

```
std::string IntToString(int alX)
```

Converts an integer to string,

```
float RandFloat(float afMin, float afMax)
```

Generates a random float.

```
int RandInt(int alMin, int alMax)
```

Generates a random int.

```
StringContains(std::string asString, std::string asSubString)
```

Checks what a string contains.
asString what string to check
asSubString what to check the string against

Resources

```
static std::string stdcall Translate(std::string asCat, std::string asName)
```

Gets a string in the current language.
asCat The translation category
asName The name of the category entry.

```
static void stdcall PreloadSound(std::string asFile)
```

Preloads the data for a sound.
asFile This can be a wav, ogg, mp3 or snt file.

```
void ResetLogicTimer()
```

Resync screen frames to logic timer.

Particle Systems

```
void SetParticleSystemActive(std::string asName, bool abActive)
```

Set if a particle system should be active or not.
asName The name of the particle system.
abActive If it should be active or not.

```
void CreateParticleSystem(std::string asName, std::string asType,
std::string asArea,
                                                    float afX, float afY,
float afZ)
```

Create a particle system at the position of an area
asName The name of the particle system.
asType The type of aprticle system
asArea The name of the area
X Y and Z the variables of the particle system.

```
CreateParticleSystemOnCamera(string asName ,string asFile)
```

Create an particle on the position of the camera(player).
asName the name of the particle system ingame.
asFile the name of the file for the particle system.

```
void KillParticleSystem(std::string asName)
```

Kill a particle system

asName The name of the particle system.

Beams

```
void CreateBeam(string asName, string asFile, string asStartArea, string
asEndArea)
```

Creates a beam between 2 areas
asName the name of the beam
asFile the name of the .beam file
asStartArea name of the area to start the beam
asEndArea name of the area to end the beam

```
void DestroyBeam(string asName)
```

Destroy a beam
asName name of the beam to destory

Lights

```
void FadeLight3D(std::string asName,
                 float afR,float afG, float afB, float afA, float
afRadius, float afTime)
```

Fades a sound to a color and a radius
asName The name of the light
afR The red channel to fade to.
afG The green channel to fade to.
afB The blue channel to fade to.
afA The alpha channel to fade to.
afRadius The radius to fade to.
afTime The amount of seconds the fade should last.

```
void AttachBillboardToLight3D(std::string asBillboardName, std::string
asLightName,bool abX)
```

Attaches a billboard to a light
asBillboardName The billbaord name
asLightName The light name
abX True if it should be attached, false if you want to remove.

```
void SetLight3DVisible(std::string asName, bool abX)
```

Sets on/off a light
asName The light name
abX if the light should be on or off.

```
void SetLight3DFlickerActive(std::string asName, bool abX)
```

Sets flickering on/off a light
asName The light name
abX if the light flicker should be on or off.

```
void SetLight3DFlicker(std::string asName,
                       float afR,float afG, float afB, float afA,
                       float afRadius,
                       float afOnMinLength, float afOnMaxLength,
                       std::string asOnSound,std::string asOnPS,
                       float afOffMinLength, float afOffMaxLength,
                       std::string asOffSound,std::string asOffPS,
                       bool abFade,
                       float afOnFadeLength, float afOffFadeLength)
```

Sets flickering parameters
asName The light name
abR, afG, afB, afA The color of the light when off
afRadius The radius of the light when off.
afOnMinLength Minimum time before going from off to on.
afOnMaxLength Maximum time before going from off to on.
asOnSound Name of the sound played when going from off to on. _fckg_QUOTfckg_QUOT_ means no sound.
asOnPS Name of the particle system played when going from off to on. _fckg_QUOTfckg_QUOT_ means none.
afOffMinLength Minimum time before going from on to off.
afOffMaxLength Maximum time before going from on to off.
asOffSound Name of the sound played when going from on to off. _fckg_QUOTfckg_QUOT_ means no sound.
asOffPS Name of the particle system played when going from on to off. _fckg_QUOTfckg_QUOT_ means none.
abFade If there should be a fade between off and on.
afOnFadeLength Fade length from off to on.
afOffFadeLength Fade length from on to off.

```
void CreateLightFlashAtArea(  std::string asArea, float afRadius,
                    float afR,float afG, float afB, float afA,
                    float afAddTime, float afNegTime);
```

Creates a flash of light at an area.
asArea Name of the area to create the flash in
afRadius Radius of the flash
afR, afG, afB, afA Red, Green, Blue and alpha of the flash..
afAddTime Time it will take to reach full strenght.
afNegTime Time it will take to end flash, turning black.

```
void SetLight3DOnlyAffectInSector(std::string asName, bool abX);
```

If a light should cast its light and shadow, affecting the surroundings in its sector(_room grouping) or in all sectors it reaches. By default static lights added in the editor to a level is true, the same goes for lights that are part of an Lamp entity.
asName The light name
abX if it should only affect the sector or not.

Audio

```
void CreateSoundEntity(std::string asName, std::string asFile, std::string asArea)
```

Creates a sound entity at the postion of an area.
asName Name of the created sound entity.
asFile The snt file to load.
asArea The area to create at.

```
void CreateSoundEntityAt(std::string asType,std::string asDestName,
                  std::string asSoundName,std::string asSoundFile);
```

Creates a sound entity at the position of an entity.
asType The type of entity. This can be: “Joint”,“Body” or “Entity”.
asDestName The entity/body/joint name
asSoundName The name of created sound entity.
asSoundFile The name of the snt file.

```
void PlaySoundEntity(std::string asName, bool abPlayStart)
```

Play a sound entity
asName The entity name
abPlayStart If the start sound should be played.

```
void StopSoundEntity(std::string asName, bool abPlayEnd)
```

Stop a sound entity
asName The entity name
abPlayEnd If the end sound should be played.

```
void FadeInSoundEntity(std::string asName, float afSpeed)
```

Play a sound entity fading it
asName The entity name
afSpeed Volume increase per second.

```
void FadeOutSoundEntity(std::string asName, float afSpeed)
```

Stop a sound entity fading it
asName The entity name
afSpeed Volume decrease per second.

```
void PlayMusic(std::string asName, float afVol, float afStepSize, bool
abLoop)
```

Play music track.
asName Name of the music file.
afVol Volume of the music
afStepSize The increase in volume per second when fading in.

```
void StopMusic(float afStepSize)
```

Stop music track.
afStepSize The decrease in volume per second when fading out.

```
PlayGameMusic(string asFile, float afVolume,float afFadeStep,bool abLoop,
int alPrio);
```

Play music track with prio, added to episode 1 to be able to play different tracks that mix in and out depending on prio.
asFile The name of the music file.
afVolume The volume to play the music at 0 to 1.
afFadeStep The decrease in volume per second when fading out.
abLoop If the music should loop.
alPrio The priority of the track, 0 to 10. A higher numbers fades a lower number out, if you have an ambient music track using prio 0 it will fade out if you add a music track to an enemy entity with prio 1 and the enemy attacks the player. When the attack is over the ambient track will fade back in.

```
StopGameMusic(float afFadeStep, int alPrio);
```

Stop game music track with prio.
afFadeStepThe decrease in volume per second when fading out.
alPrio priority of the music track.

```
void PlayGuiSound(std::string asName, float afVol)
```

Play a sound gui sound, with out any position.
asName The sound name
afVol Volume of the sound

Physics

```
void SetJointCallback(std::string asJointName, std::string
asType,std::string asFunc)
```

Sets a callback for a joint.
The syntax for the function is: void MyFunction(string asJointName)
asJointName The joint name

asType The type, can be: “OnMax” or “OnMin”.
asFunc The script function to be called. Must be in the current script file. _fckg_QUOTfckg_QUOT_ = disabled.

`void BreakJoint(std::string asJointName)` Breaks a joint.
asJointName The joint name

`void SetJointControllerActive(std::string asJointName,std::string asCtrlName, bool abActive)` Sets if a joint controller is active or not.
asJointName The joint name
asCtrlName The controller name
abActive If the controller is to be active or not.

`SetJointControllerPropertyFloat(std::string asJointName,std::string asCtrlName, std::string asProperty, float afValue)` Sets if a joint controller is active or not.
asJointName The joint name
asCtrlName The controller name
asProperty The property to change.
afValue The value of the property to change.

`void ChangeJointController(std::string asJointName,std::string asCtrlName)` Change the active controller. All other controllers are set to false.
asJointName The joint name
asCtrlName The controller name

`float GetJointProperty(std::string asJointName, std::string asProp)` Gets a property from the joint.
Valid properties are:
“Angle” The angle between the bodies (in degrees) (Not working for ball joint)
“Distance” The distance between the bodies (in meter)
“LinearSpeed” The relative linear speed between the bodies (in m/s)
“AngularSpeed” The relative angular speed between the bodies (in m/s)
“Force” The size of the force (in newton, kg*m/s^2).
“MinLimit” The minimum limit of the joint.
“MaxLimit” The maximum limit of the joint.
asJointName The joint name
asProp The property to get

`SetJointProperty(string asJoint, string asProp, float afVal)` Sets a property to the joint.
asJoint the joint to change property for
asProp what property to change:
“MinLimit” the min limit on joint, does not work on ball joint.
“MaxLimit” the max limit on joint, does not work on ball joint.
afVal the value in float to set.

`float GetBodyProperty(std::string asBodyName, std::string asProp)` Gets a property from the body.
Valid properties are:
“Mass” The mass of the body (in kg)

“LinearSpeed” The linear speed the body has (in m/s)
“AngularSpeed” The angular speed the body has (in m/s)
asBodyName The body name
asProp The property to get

`void SetBodyProperty(std::string asBodyName, std::string asProp, float afVal)`
Sets a property to the body.
Valid properties are:
“Mass” The mass of the body (in kg)
“CollideCharacter” If the body should collide with the character or not, 0=false 1=true.
“HasGravity” If a body is affected by gravity. 0=false 1=true.
asBodyName The body name
asProp The property to get
afVal The new value of the property

`void AddBodyForce(std::string asBodyName, std::string asCoordType, float afX, float afY, float afZ)` Adds a force to the body. This can either be in the bodies local coord system or the world's.
asBodyName The body name
asCoordType The coordinate system type. “World” or “Local”.
afX force in the x direction. (in newton, kg*m/s^2)
afY force in the y direction. (in newton, kg*m/s^2)
afZ force in the z direction. (in newton, kg*m/s^2)

`void AddBodyImpulse(std::string asBodyName, std::string asCoordType, float afX, float afY, float afZ)` Adds an impule (a change in velocity) to the body. This can either be in the bodies local coord system or the world's.
asBodyName The body name
asCoordType The coordinate system type. “World” or “Local”.
afX velocity in the x direction. (in m/s)
afY velocity in the y direction. (in m/s)
afZ velocity in the z direction. (in m/s)

==== Local Variables ==== `void CreateLocalVar(std::string asName, int alVal)` `void SetLocalVar(std::string asName, int alVal)` `void AddLocalVar(std::string asName, int alVal)` `int GetLocalVar(std::string asName)`

==== Global Variables ==== `void CreateGlobalVar(std::string asName, int alVal)` `void SetGlobalVar(std::string asName, int alVal)` `void AddGlobalVar(std::string asName, int alVal)` `int GetGlobalVar(std::string asName)`

===== Game scripts ===== ==== General ==== `void ResetGame();` Resets the game and returns to main menu

`void StartCredits();` Starts the end credits screen, uses .lang Category “MainMenu” and .lang Entry “CreditsText” as well as music file “penumbra_music_E1_E.ogg”

`void StartDemoEndText();` Starts the end images that are used in the demo of episode 1, images are named “demo_end01.jpg” increase the number for each image you want to use.

`string GetActionKeyString(std::string asAction);` Gets the key for a certain action, for example can be used in tutorials to create text that uses the user configured key for “Interact”.
asAction The name of the action

`void AddMessageTrans(std::string asTransCat, std::string asTransName);` Adds on on screen game message.
TransCat The translation category
TransName The translation name

`void AddMessage(std::string asMessage);` Adds on on screen game message.
asMessage The text to be shown.

`void SetMessagesOverCallback(std::string asFunction);` Sets a callback that is called (and removed) when the last message as been shown.
asFunction The function to be called, syntax “MyFunc()“

`AddSubTitleTrans(std::string asTransCat,std::string asTransName, float afTime);` Adds a onscreen message at the bottom, that does not interrupt the gameplay.
asTransCat The translation files category.
asTransName The translation files name of the entry to be displayed.
afTime the length of time it should be displayed.

`AddSubTitle(std::string asMessage, float afTime);` Adds a onscreen message at the bottom, that does not interrupt the gameplay.
asMessage the text you want to appear on screen.
afTime the length of time it should be displayed.

`SetMessageBlackText(bool abX)` Makes the onscreen text invert colours to be easier to read on bright backgrounds.
abX enable or disabled the function.

`void AddRadioMessage( std::string asTransCat,std::string asTransName, std::string asSound);` Plays an audio file and displays subtitles for it.
asTransCat the Category in the translations file
asTransName the name of the entry in the translations file
asSound the sound file to play for the text.

`void SetRadioOnEndCallback(std::string asFunc);` Callback that activates when a Radio message is over
asFunc The name of the callback to activate

VARIOUS TEMP STRINGS
`void AddToTempString(std::string asString);` *A piece of text to add to a string* `void AddToTempStringTrans(std::string asCat,std::string asEntry);` A text from the translations file in Category from Entry `void AddToTempStringAction(string);` *Get user configured key for a certain action* `void AddMessageTempString();` Add the strings to one temporary string to print as a Message on screen `void AddSubTitleTempString(float);` *Add the strings to*

one temporary string to print as a Subtitle on screen Used to create a flow of text that uses various sources, as an example from episode 1:
*AddToTempStringTrans(*GESHI_QUOT*00_01_boat_cabin*GESHI_QUOT*,* GESHI_QUOT*PadLock01*GESHI_QUOT *);*
*AddToTempStringAction(*GESHI_QUOT*Inventory*GESHI_QUOT *);*
*AddToTempStringTrans(*GESHI_QUOT*Misc*GESHI_QUOT*,* GESHI_QUOT*ParentDot*GESHI_QUOT *); AddMessageTempString(); Wrote the message "Always travel with a padlock. And a key, preferably. Mine's in the inventory (TAB)." as a message.*

static void stdcall ChangeMap(std::string asMapFile, std::string asMapPos, std::string asStartSound, std::string asStopSound, float afFadeOutTime, float afFadeInTime, std::string asLoadTextCat, std::string asLoadTextEntry); Changes the map.
asMapFile The file to be loaded
asMapPos The position on the map to be spawned on.
asStartSound The sound that is played when the change starts.
asStopsSound The sound that is played when the new map is loaded
afFadeOutTime Time in seconds it takes for the fade out.
afFadeInTime Time in seconds it takes for the fade in.
asLoadTextCat Category in .lang file to use
asLoadTextEntry Entry in .lang file to use

void AddNotebookTask(std::string asName, std::string asTransCat, std::string asTransEntry); Adds a new task note in the notebook
asName Name of the task
asTransCat Translation category of text.
asTransEntry Translation entry of text.

void AddNotebookTaskText(std::string asName,std::string asText); Adds a new task note in the notebook without using the lang file.
asName Name of the task
asText The text for the task

void AddNotebookNote(std::string asNameCat, std::string asNameEntry, std::string asTextCat, std::string asTextEntry); Adds a note to the note book
asNameCat
asNameEntry Name of the note
asTextCat
asTextEntry The note text.

void RemoveNotebookTask(std::string asName); Removes a task note in the notebook.
asName Name of the task

void StartNumericalPanel(std::string asName,int alCode1,int alCode2,int alCode3, int alCode4, float afDifficulty, std::string asCallback); Starts the numerical panel
asName Name of the panel
alCode1 1st code digit.
alCode2 2nd code digit.

alCode3 3rd code digit.
alCode4 4th code digit.
afDifficulty How difficult it is to hack.
asCallback Called when the code as been entered. Syntax: **MyCallback(string asName, bool abCodeWasCorrect)**

`void SetInventoryActive(bool abX);` Set if the inventory should be active.
abX If it is active or not.

`void FadeIn(float afTime);` Fades the screen to full color.
afTime Time the fade will take in seconds.

`void FadeOut(float afTime);` Fades the screen to black.
afTime Time the fade will take in seconds.

`bool IsFading();` Check if a fade is going on.

`void SetWideScreenActive(bool abActive);` Sets wide screen mode on or off.

`void AutoSave();` Save the game to the auto save.

`void ClearSavedMaps();` Clears the “history” of the save, useful to do when you know the player will not be able to go back anymore. Makes the next save much smaller in size. It's a must to clear the save history a couple of times during the length of a game, in Penumbra: Overture we do it 5-7 times during the whole game.

`SetDepthOfFieldActive(bool abX, float afFadeTime);` Sets depth of field on/off
abX if true/false
afFadeTime how long for it to focus.

`SetupDepthOfField(float afNearPlane, float afFocalPlane, float afFarPlane);` How the depth of field should appear
afNearPlane where the focus should begin
afFocalPlane how large area that is in focus
afFarPlane where the focus should end

`FocusOnEntity(std::string asEntity);` Set focus on a particular entity.
asEntity name of the entity

`SetConstantFocusOnEntity(std::string asEntity);` Set focus on a particular entity and keep focus on it if it moves or the player moves.
asEntity name of the entity

`void StartFlash(float afFadeIn, float afWhite, float afFadeOut);` Creates a white flash on screen.
afFadeIn The time the fade in takes.
afWhite The time it stays white.
afFadeOut The time the fade out takes.

`StartScreenShake(float afAmount, float afTime,float afFadeInTime,float afFadeOutTime);` Makes the screen shake.

afAmount The ammount of shaking.
afTime How long the shake should be.
afFadeInTime How fast the shake should fade in.
afFadeOutTime How fast the shake should fade out.

SetupSaveArea(std::string asName,std::string asMessageCat,std::string asMessageEntry, std::string asSound); Properties for an area to type “save”, eg _fckg_QUOT*area_save_gamename”. asName the name of the area*
asMessageCat The message category in lang file
asMessageEntry The message entry in lang file
asSound The sound to play during the save

==== Attacks ==== void CreateSplashDamage(std::string asAreaName, float afRadius, float afMinDamage, float afMaxDamge, float afMinForce, float afMaxForce, float afMaxImpulse, int alStrength); Creates a ball shaped splash damage at the center of an area.
asAreaName Name of the area.
afRadius Radius of the slash damage in meters.
afMinDamage The minimum damage if in side radius
afMaxDamge The maximum damage, this is when you are at center.
afMinForce The minimum force if you are inside the radius.
afMaxForce The maximum force, this is when you are at center.
afMaxImpulse Impulse is ForceSize / Mass and this is the max value for that. This is useful to insure that samll objects to no gain very high speed.
alStrength The strength of the the damage.

*void SetupDamageArea(string asName, float afDamage, float afUpdatesPerSec, int alStrength, bool abDisableObjects, bool abDisableEnemies, string asDamageType, string asDamageSound); Properties for an area of type “damage”, eg _fckg_QUOT*area_damage_gamename”. Will inflict damage on player and entites while in it.
asName The name of the area
afDamage Damage to do for each update
afUpdatesPerSec How many updates per second
alStrength The strength of each update
abDisableObjects If objects should be disabled on destruction
abDisableEnemies If enemies should be disabled on destruction
asDamageType What sort of damage graphics to use. “BloodSplash”, “Ice”, “Poison”.
asDamageSound What sound to play at each damage.

void DamageEntity(std::string asName, float afDamage, int alStrength); Gives damage to an entity
asName the name of the entity
afDamage the amount of damage to do
alStrength the strength of the damage

==== Game Timer ==== void CreateTimer(std::string asName, float afTime, std::string asCallback, bool abGlobal); Creates a new timer which calls a callback

function when the time is out.
Syntax: **MyCallback(string asTimerName)**, the prefix "@" (ie "@Func()") shows that the function is in the global script
asName Name of the timer
afTime The time til time out in seconds.
asCallback The callback fucntion
abGlobal If the timer is global. Global timers are not deleted at map change.

`void DestroyTimer(std::string asName); void SetTimerPaused(std::string asName, bool abPaused); void SetTimerTime(std::string asName, float afTime); void AddTimerTime(std::string asName, float afTime); float GetTimerTime(std::string asName);`

==== Player ==== `static void GivePlayerDamage(float afAmount, std::string asType);` Gives the player a hit + damage
afAmount The amount of health taken.
asType what sort of damage effect
"BloodSplash"
"Ice"
"Poison"

`void SetPlayerHealth(float afHealth);` Sets the health of the player.
afHealth The amount the health should be set to.

`float GetPlayerHealth();` Gets the current health of the player.

`void SetPlayerPose(std::string asPose,bool abChangeDirectly);` Sets the pose of the player.
asPose The pose, can be "Stand" or "Crouch"
abChangeDirectly, if the pose should change instantly or in a movement

`void SetPlayerActive(bool abActive);` Sets if player can be moved or not.
abActive If the player is active or not.

`void StartPlayerLookAt(std::string asEntityName, float afSpeedMul, float afMaxSpeed);` This turns the players head towards a specific entity.
asEntityName The entity name
afSpeedMul The speed of the head movement is determined by the angle length multiplied by this.
afMaxSpeed The maximum speed the head will turn.

`void StopPlayerLookAt();` Stops the look at motion.

`void StartPlayerFearFilter(float afStrength);` Starts a fear effect filter.
afStrength strength of the filter.

`SetFlashlightDisabled(bool abDisabled);` Disabled the flashlight.
abDisabled true/false for flashlight on/off.

`void StopPlayerFearFilter();` Stops the fear filter.

`void TeleportPlayerToArea(string asArea);` Teleports the player to an area, players feet is

aligned to the center of the area.
asArea the name of the area to teleport to.

`bool GetPlayerHasGasMask();` Get if the player has gasmask or not.

==== Inventory ==== `void AddPickupCallback(std::string asItem, std::string asFunction);` Add a callback that is called when the player picks up an item.
The syntax for the callback function is:
void MyFunction(string asItem), the prefix "@" (ie "@Func()") shows that the function is in the global script
asItem The name of the item.
asFuntion The name of the function called.

`void AddUseCallback(std::string asItem, std::string asEntity, std::string asFunction);` Add a callback that is called when the player uses the item on en entity
The syntax for the callback function is:
void MyFunction(string asItem, string asEntity), the prefix "@" (ie "@Func()") shows that the function is in the global script.
asItem The name of the item.
asEntity The name of the entity.
asFuntion The name of the function called.

`void AddCombineCallback(std::string asItem1,std::string asItem2, std::string asFunction);` Add a callback that is called when the player combines two items
The syntax for the callback function is (alSlotIndex = slot at which combination occurd):
void MyFunction(string asItem1, string asItem2, int alSlotIndex), the prefix "@" (ie "@Func()") shows that the function is in the global script.
asItem1 The name of the first item.
asItem2 The name of the second item.
asFuntion The name of the function called.

`void RemovePickupCallback(std::string asFunction); void RemoveUseCallback(std::string asFunction); void RemoveCombineCallback(std::string asFunction);` Removes the callback.
asFuntion The name of the function used by the callback.

`bool HasItem(std::string asName);` Checks if the player has a certain item.
asName The name of the item.

`void RemoveItem(std::string asName);` Removes an item from the inventory.
asName The name of the item.

`void GiveItem(std::string asName,std::string asEntityFile, int alSlotIndex);` Give an item to the player.
asName The name of the entity
asEntityFile The file to load it from.
lSlotIndex The index of the slot to put the item in. -1 = first empty slot

`SetInventoryMessage(std::string asMessage);` Displays a message in the inventory.
asMessage The message to write in the inventory.

`SetInventoryMessageTrans(std::string asTransCat, std::string asTransName);`