helper_player.hps

Player_GetPosition

```
cVector3f Player_GetPosition()
```

Gets the position of the player.

Returns: cVector3f , The position of the player.

Player_GetFeetPosition

```
cVector3f Player_GetFeetPosition()
```

Gets the feet position of the player.

Returns: cVector3f , The position of the player's feet.

Player_GetDistanceToEntitySquared

```
float Player_GetDistanceToEntitySquared(const tString &in asEntity)
```

- **asEntity**: The entity name

Returns: float, The square of the distance the entity is from the player.

Player_GetDistanceToEntity

```
float Player_GetDistanceToEntity(const tString &in asEntity)
```

- **asEntity**: The entity name

Returns: float, The distance the entity is from the player.

Player_SetUnderwater

```
void Player_SetUnderwater(bool abX)
```

Sets whether player is underwater or not, setting up various properties for this

- **abX**: if underwater or not.

Player_GetUnderwater

```
bool Player_GetUnderwater()
```

Retuns if the player is under water or not.

Player_SetCharacterBodyDefaults

```
void Player_SetCharacterBodyDefaults()
```

Reset the properties of the character body to what they were when the game started

---

Player_SetGravity

```
void Player_SetGravity(const cVector3f &in avGravity)
```

Set the level of gravity affecting the player

- **avGravity**: the gravity vector to use.

Player_SetGravityEnabled

```
void Player_SetGravityEnabled(bool abX)
```

Enable or disable gravity affecting the player

- **abX**: if gravity is enabled or not.

Player_SetFallDamageActive

```
void Player_SetFallDamageActive(bool abX)
```

Sets whether fall damage is allowed or not.

- **abX**: true = can take fall damage. - false = cannot take fall damage.

Player_Jump

```
void Player_Jump()
```

Makes the player jump

Player_SetCrouching

```
void Player_SetCrouching(bool abState,
                         bool abInstant=false,
                         bool abSilent=false)
```

Sets whether player is crouching or not.

- **abState**: true = player crouches - false = player stops crouching.
- **abInstant**: true = camera moves to new position instantly - false = camera moves to new position over time.
- **abSilent**: false = play sounds, true = don't play sounds

---

Player_GetCrouching

```
bool Player_GetCrouching()
```

Gets whether player is crouching or not.

Returns: bool, true = player crouches - false = player is not crouching.

Player_IncCrawlCount

```
void Player_IncCrawlCount()
```

Increaments the crawl counter, which if it is >0and player is crouching it means the player is crawling.

Player_DecCrawlCount

```
void Player_DecCrawlCount()
```

Decrements the crawl counter, which if it is >0 and player is crouching it means the player is

crawling.

Player_SetForceCrawling

```
void Player_SetForceCrawling(bool abX)
```

Forces the player to be in crawling mode no matter if they are standing or not.

Player_SetDisableCrawling

```
void Player_SetDisableCrawling(bool abX)
```

Turns off crawling and forces it not to occur again.

Player_GetHeadBob

```
cVector3f Player_GetHeadBob()
```

Gets the current headbob amount

Player_GetRunning

```
bool Player_GetRunning()
```

Gets whether player is running or not.

Player_SetFootWear

```
void Player_SetFootWear(tString &in asType)
```

Set what type of footwear the player has.

- **asType**: can be: barefoot, sneaker, default. For Depth default will be the soma steps.

Player_SetClothing

```
void Player_SetClothing(tString &in asType)
```

Set what type of clothing the player is wearing. These change the jump and pose sounds.

- **asType**: can be: naked, dressed, default. For Depth default will be the soma body.

Player_SetHeadBobMul

```
void Player_SetHeadBobMul(float afMul)
```

Sets the head bob amount multiplier

- **afMul**: the amount of headbob, 0=none, 1=normal

Player_FadeHeadBobMul

```
void Player_FadeHeadBobMul(float afMul,
                           float afTime=1.0f)
```

Sets the head bob amount multiplier

- **afMul**: the amount of headbob, 0=none, 1=normal
- **afTime**: the time to fade over

Player_GetHeadBobMul

```
float Player_GetHeadBobMul()
```

Gets the head bob amount multiplier

Player_SetFootstepSoundsDisabled

```
void Player_SetFootstepSoundsDisabled(bool abX)
```

Set if foot steps sounds (landing included) should be played or not.

- **abX**: true = steps should be played - false = steps should not be playe

Player_SetRunBreathingDisabled

```
void Player_SetRunBreathingDisabled(bool abX)
```

Player_SetActive

```
void Player_SetActive(bool abX)
```

Set if the player can control the main character.

- **abX**: set controls active or not.

Player_GetLookSpeedMul

```
float Player_GetLookSpeedMul()
```

Gets the player's look speed multiplier

Returns: float, the speed multiplier, where 1.0 is the default value

Player_SetLookSpeedMul

```
void Player_SetLookSpeedMul(float afMul)
```

Sets the player's look speed multiplier

- **afMul**: multiplier value to set, where 1.0 is the default value

Player_FadeLookSpeedMulTo

```
void Player_FadeLookSpeedMulTo(float afMul,
                               float afTime)
```

Fades the player's look speed multiplier over a specified time.

- **afMul**: multiplier value to fade to, where 1.0 is the default value
- **afTime**: time to fade over.

Player_GetMoveSpeedMul

```
float Player_GetMoveSpeedMul()
```

Gets the player's move speed multiplier

Returns: float, the speed multiplier, where 1.0 is the default value

---

Player_SetMoveSpeedMul

```
void Player_SetMoveSpeedMul(float afMul)
```

Sets the player's move speed multiplier

- **afMul**: multiplier value to set, where 1.0 is the default value

---

Player_FadeMoveSpeedMulTo

```
void Player_FadeMoveSpeedMulTo(float afMul,
                               float afTime)
```

Fades the player's move speed multiplier over a specified time.

- **afMul**: multiplier value to fade to, where 1.0 is the default value
- **afTime**: time to fade over.

Player_GetRunSpeedMul

```
float Player_GetRunSpeedMul()
```

Gets the player's script run speed multiplier

Returns: float, the speed multiplier, where 1.0 is the default value

---

Player_SetRunSpeedMul

```
void Player_SetRunSpeedMul(float afMul)
```

Sets the player's script run speed multiplier

- **afMul**: multiplier value to set, where 1.0 is the default value

---

Player_FadeRunSpeedMulTo

```
void Player_FadeRunSpeedMulTo(float afMul,
                              float afTime)
```

Fades the player's run speed multiplier over a specified time.

- **afMul**: multiplier value to fade to, where 1.0 is the default value
- **afTime**: time to fade over.

Player_SetJumpDisabled

```
void Player_SetJumpDisabled(bool abX)
```

Sets whether jumping should be disabled or not.

- **abX**: true = jumping is disabled - false = jumping is enabled

---

Player_CancelJump

```
void Player_CancelJump()
```

If a jump is currently in progress, cancel it.

Player_SetCrouchActionDisabled

```
void Player_SetCrouchActionDisabled(bool abX)
```

Sets whether anything should happen when you press the crouch button

- **abX**:

Player_SetCrouchDisabled

```
void Player_SetCrouchDisabled(bool abX)
```

Sets whether crouching should be disabled or not.

- **abX**: true = crouching is disabled - false = crouching is enabled

---

Player_SetStandDisabled

```
void Player_SetStandDisabled(bool abX)
```

Sets whether standing should be disabled or not.

- **abX**: true = standing is disabled - false = standing is enabled

---

Player_SetAfterDamageCrawl

```
void Player_SetAfterDamageCrawl(bool abX)
```

If the crawling taking place is a special after taking damage

- **abX**: If active or not.

Player_SetSlowStandupMotion

```
void Player_SetSlowStandupMotion(bool abX)
```

If the player should move much slower when rising up from crouch.

- **abX**: If active or not.

Player_WasDamaged

```
bool Player_WasDamaged()
```

Returns true if the player was damaged this update, else false.

Player_GetCurrentStateName

```
tString Player_GetCurrentStateName()
```

Returns the name of the current state.

Player_ChangeStateToNormal

```
void Player_ChangeStateToNormal()
```

Sets the state of the player to normal

Player_ChangeStateToCustomControls

```
void Player_ChangeStateToCustomControls(const tString &in
asLookInputCallback,
                                        const tString &in
asMoveInputCallback,
                                        const tString &in
asActionInputCallback,
                                        const tString &in
asLeanInputCallback="")
```

Changes the player state so that the normal controls are used for something else.

- **asLookInputCallback**: Overrides look controls. If false is returned, normal controls are overridden. Syntax: bool F(const cVector2f &in avLookAmount)
- **asMoveInputCallback**: Overrides move controls. If false is returned, normal controls are overridden. Syntax: bool F(const cVector2f &in avMoveAmount)
- **asActionInputCallback**: Overrides action input. If false is returned, normal controls are overridden. Syntax: bool F(int alAction, bool abPressed)
- **asLeanInputCallback**: Overrides lean input. If false is returned, normal controls are overridden. Syntax: bool F(float afAmount)

Player_GetDefaultEyelineHeight

```
float Player_GetDefaultEyelineHeight()
```

Get the player's eyeline height i.e. the height from the foot position to the camera

Returns: The distance between the feet and the default eyeline

Player_SetCameraUseSmoothing

```
void Player_SetCameraUseSmoothing(bool abValue)
```

Turn on or off camera smoothing. When off, this stops the camera lagging slightly behind the player's movements.

- **abValue**: smooth values of the camera or not

Player_StartLookAt

```
void Player_StartLookAt(const tString &in asEntityName,
                        float afAcc,
                        float afSpeedMul,
                        float afMaxSpeed)
```

Rotate the player camera towards the positon of an entity/area.

- **asEntityName**: name of the entity/area to rotate camera towards.
- **afAcc**: acceleration of the camera movement.
- **afSpeedMul**: distance towards camera multiplied with this value sets the camera speed.
- **afMaxSpeed**: maximum speed of camera movement.

Player_StopLookAt

```
void Player_StopLookAt(float afDeacc)
```

Stop rotating the camera towards a target specified with

- **afDeacc**: deacceleration of the camera movement. 0 means 'stop now'

Player_MoveHeadPos

```
void Player_MoveHeadPos(const cVector3f &in avPos,
                        float afAcc,
                        float afSpeed,
                        float afSlowDownDist)
```

Moves the position of the camera over time.

- **avPos**: target offset from original camera position, relative to the rotation of the player.

- **afAcc**: acceleration of camera movement.
- **afSpeed**: target speed of camera movement.
- **afSlowDownDist**: distance from target where movement starts slowing down.

Player_FadeFOVMulTo

```
void Player_FadeFOVMulTo(float afX,
                         float afSpeed)
```

Fades the FOV Multiplier to a value

- **afX**: The FOV to a value, something like 0.001 - 2 are okay values. 1 = default
- **afSpeed**: The amount of change per second

Player_FadeAspectMulTo

```
void Player_FadeAspectMulTo(float afX,
                            float afSpeed)
```

Fades the Aspect Multiplier to a value. This value represent how much wider the screen than the height.

- **afX**: The Aspect to a value, something like 0.2 - 2 are okay values. 1 = default
- **afSpeed**: The amount of change per second

Player_FadeFOVTo

```
void Player_FadeFOVTo(float afX,
                      float afSpeed)
```

Fades the FOV value. This does NOT affect the FOV multiplier which will still be applied

- **afX**: The vertical FOV to a value in radians, -1 = default (the menu displays horzontal FOV which is not the same)
- **afSpeed**: The amount of change per second

Player_FadeFOVToDefault

```
void Player_FadeFOVToDefault(float afSpeed)
```

Fades the FOV value to default. This does NOT affect the FOV multiplier

- **afSpeed**: The amount of change per second

Player_FadeRollTo

```
void Player_FadeRollTo(float afX,
                       float afSpeedMul,
                       float afMaxSpeed)
```

Fades the camera roll angle.

- **afX**: This is the angle of rotation in angles. 0=normal and 180=upside down.
- **afSpeedMul**: This is used to calculate the speed based on the distance to the goal angle. speed = dist_to_goal * afSpeedMul
- **afMaxSpeed**: The maximum speed the camera is rotated in

Player_SetRoll

```
void Player_SetRoll(float afX)
```

Sets the camera roll angle.

- **afX**: This is the angle of rotation in angles. 0=normal and 180=upside down.

Player_Teleport

```
void Player_Teleport(const tString &in asStartPosName,
                     bool abAlignYaw=true)
```

Teleport the player to a specific PlayerStart Area.

- **asStartPosName**: name of the PlayerStart Area to teleport to.
- **abAlignYaw**: if the player yaw should be aligned with the PlayerStart Area's yaw.

Player_PlaceAtEntity

```
void Player_PlaceAtEntity(const tString &in asEntityName,
                          bool abAlignYaw=true)
```

Teleport the player to a specific entity

- **asEntityName**: name of the entity to teleport to.
- **abAlignYaw**: if the player should be rotated to match the entity's rotation

---

Player_EnableCameraLock

```
void Player_EnableCameraLock(float afLocalYawMin,
                             float afLocalYawMax,
                             float afLocalPitchMin,
                             float afLocalPitchMax)
```

This locks the player's camera based upon the current look direction. Arguments are in degrees and local to the current look direction.

Player_DisableCameraLock

```
void Player_DisableCameraLock()
```

Disables the camera lock and makes it go back to normal

Player_GetCameraPosition

```
cVector3f Player_GetCameraPosition()
```

Returns the position that the camera should be having at this moment (but might not, if it is doing something else)

Player_GetCameraYawPitchRoll

```
cVector3f Player_GetCameraYawPitchRoll()
```

Returns the yaw, pitch and roll for the current player camera

---

Player_AddBodyForce

```
void Player_AddBodyForce(const cVector3f &in avForce,
```

```
                                bool abUseLocalCoords)
```

Adds force to the player's body.

- **avForce**: force added.
- **abUseLocalCoords**: true = force is added relative to the player's rotation - false = force is added relative to the world.

Player_AddContinuosBodyForce

```
void Player_AddContinuosBodyForce(const cVector3f &in avForce,
                                  float afTime)
```

Adds continuous force to the player's body for a certain duration in global space Only support one of this running at one time.

- **avForce**: force added.
- **afTime**: Duration of the force add

_Player_AddContinuosBodyForce_Update

```
void _Player_AddContinuosBodyForce_Update(const tString &in asTimer)
```

Player_AddBodyForceAwayFromEntity

```
void Player_AddBodyForceAwayFromEntity(const tString &in asEntity,
                                       float afForce,
                                       bool abOnly2D=false,
                                       float afMaxForceSpeed=2)
```

Adds force to the player's body away from the specified entity.

- **asEntity**: entity to push away from.
- **afForce**: magnitude of the force added.
- **abOnly2D**: true = only have into account the x and z values
- **afMaxForceSpeed**:

Player_GetSpeed

```
float Player_GetSpeed()
```

Gets the player's speed.

Returns: float, the speed of the player's movement, measured in m/s.

Player_GetVelocity

```
cVector3f Player_GetVelocity()
```

Gets the player's velocity.

Returns: cVector3f , the velocity of the player.

Player_MoveForward

```
void Player_MoveForward(float afAcc)
```

Moves the player character in the direction it's facing.

- **afAcc**: acceleration with which to move forward.

Player_GetFocusEntityName

```
tString Player_GetFocusEntityName()
```

Gets the name of the current entity in focus. Will only returna string when player is in the normal state.

Player_GetFocusEntityID

```
tID Player_GetFocusEntityID()
```

Gets the ID of the current entity in focus. Will return tID_Invalid when player is in the normal state or has no focus entity.

Player_IsInteracting

```
bool Player_IsInteracting()
```

Returns false if the player is grabbing an object, reading a terminal or a piece of paper etc. etc.

Player_SetInteractionAllowed

```
void Player_SetInteractionAllowed(bool abState)
```

Turn on or off the player's ability to interact with the world.

Player_GetInteractionAllowed

```
bool Player_GetInteractionAllowed()
```

Is the player allowed to interact with the world?

Player_IsReading

```
bool Player_IsReading()
```

Return true if the player is reading a note, terminal or zoom area

---

Player_ShowCrossHairIcons

```
void Player_ShowCrossHairIcons(bool abX)
```

Sets whether the crosshair should be drawn or not.

- **abX**: true = draw crosshair - false = hide crosshair.

Player_SetCrossHairState

```
void Player_SetCrossHairState(eCrossHairState alState)
```

Sets the crosshair state for the player.

- **alState**: state to set.

Player_SetFlashlightActive

```
void Player_SetFlashlightActive(bool abX,
                                bool abSuppressSound=false)
```

Turns the flashlight on or off.

- **abX**: true = flashlight on. - false = flashlight off.
- **abSuppressSound**: if true don't make a noise

Player_GetFlashlightActive

```
bool Player_GetFlashlightActive()
```

Returns if the flashlight is turned on or off

Returns: abX, true = flashlight on. - false = flashlight off.

Player_GetFlashlightRange

```
float Player_GetFlashlightRange()
```

Returns the range of the flashlight

Returns: afX, Radius of flashlight.

Player_SetFlashlightFadeOut

```
void Player_SetFlashlightFadeOut(float afSpeed,
                                 eEasing aEasing=eEasing_Linear)
```

Set the flashlight fade out and ease

- **afSpeed**: fade speed
- **aEasing**: easing method to use

Player_SetFlashlightFadeIn

```
void Player_SetFlashlightFadeIn(float afSpeed,
                                eEasing aEasing=eEasing_Linear)
```

Set the flashlight fade in and ease

- **afSpeed**: fade speed
- **aEasing**: easing method to use

Player_SetFlashlightDisabled

```
void Player_SetFlashlightDisabled(bool abX)
```

Enables or disables toggling flashlight on/off

- **abX**: true = flashlight can't be turned on or off. - false = flashlight can be turned on or off.

Player_SetFlashlightOnOffCallback

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

syntax for callback function, FunctionName(bool abLit).

Player_SetFlashlightEnvParticleMul

```
void Player_SetFlashlightEnvParticleMul(float afMul)
```

Set how much the player flashlight should affect env particles

- **afMul**: how bright the flashlight should make the env particles

---

Player_SetAmbientLight

```
void Player_SetAmbientLight(float afRadius,
                            float afBrightness,
                            float afFadeTime)
```

Player_SetAmbientLight_None

```
void Player_SetAmbientLight_None(bool abFade=false)
```

Turns off the ambient light

- **abFade**: if the light should fade

Player_SetAmbientLight_Indoors

```
void Player_SetAmbientLight_Indoors(bool abFade=false)
```

Sets up the ambient light for indoors usage.

- **abFade**: if the light should fade

Player_SetAmbientLight_Outdoors

```
void Player_SetAmbientLight_Outdoors(bool abFade=false)
```

Sets up the ambient light for outdoors usage.

- **abFade**: if the light should fade

CameraAnimation_Begin

```
void CameraAnimation_Begin(const tString &in asAnimationName,
                           const tString &in asInteractCallback,
                           bool abExitOnLastNode)
```

Initializes and starts the interactive camera animation state using all nodes in the specified animation.

- **asAnimationName**: animation name, the first part of the name of all animation node areas that should be included in the animation.
- **asInteractCallback**: name of the interact callback, this method is called whenever player interacts with objects while being in camera state. Syntax: void FuncName(const tString &in asEntity)
- **abExitOnLastNode**: whether the state should automatically end when the animation has hit the final node.

CameraAnimation_End

```
void CameraAnimation_End()
```

Stops the camera animation. Has the same effect as Player_ChangeStateToNormal, but only if the camera animation state is currently active.

CameraAnimation_SetAttachedEntity

```
void CameraAnimation_SetAttachedEntity(const tString &in asEntity,
                                       bool abAlignYaw,
                                       bool abAlignPitch,
                                       bool abAlignRoll)
```

This attacheds a single entity with the basic transform of the camera (that is player movements not included).

CameraAnimation_RemoveAttachedEntity

```
void CameraAnimation_RemoveAttachedEntity()
```

CameraAnimation_GetCurrentPitch

```
float CameraAnimation_GetCurrentPitch()
```

Returns: Current player-applied pitch compared to the viewpoint of the current node.

CameraAnimation_GetCurrentYaw

```
float CameraAnimation_GetCurrentYaw()
```

Returns: Current player-applied yaw compared to the viewpoint of the current node.

Player_ExitTerminal

```
void Player_ExitTerminal()
```

Player_GetCurrentToolEntityName

```
tString Player_GetCurrentToolEntityName()
```

Returns the name of the tool entity currently equipped.

Returns: tString , the name of the tool entity. An empty string if the player isn't carrying a tool or if the tool doesn't have an entity associated with it.

Player_GetCurrentHudEntityName

```
tString Player_GetCurrentHudEntityName()
```

Returns the name of the hud entity currently equipped.

Returns: tString , the name of the hud entity. An empty string if the player doesn't have a hud entity.

Player_ToolIsInInventory

```
bool Player_ToolIsInInventory(const tString &in asTool)
```

Checks if the specified tool is in the player's inventory.

- **asTool**: name of tool entity.

Returns: bool, true if the tool is in the player's inventory.

Player_EquipTool

```
bool Player_EquipTool(const tString &in asTool,
                      const tString &in asLookEntities="")
```

Tries to equip a specific tool in the player's inventory. Returns false if the tool isn't in the inventory.

- **asTool**: the name of the tool entity to equip.
- **asLookEntities**: names of the entities (comma separated) that trigger the equipping of the tool when inside the camera view.

Player_ReEquipTool

```
bool Player_ReEquipTool(float afTime=0.5f)
```

Tries to equip the tool that was last unequipped. Useful when unequipping something temporarily. Returns false if the tool is no longer in the inventory or no tool has been unequipped.

- **afTime**: the time it should take to raise tool. Defaults to 0.5 seconds.

Player_UnequipTool

```
void Player_UnequipTool(bool abRemoveFromInventory=false)
```

Unequips the currently carried tool.

- **abRemoveFromInventory**: if tool should be removed from inventory when hidden. Defaults to false.

Player_RemoveTool

```
void Player_RemoveTool(const tString &in asTool)
```

Removes the specified tool from the player's inventory so it can no longer be equipped.

- **asTool**: the name of the tool entity to remove.

Player_RemoveAllTools

```
void Player_RemoveAllTools()
```

Clears the player's inventory.

Player_PlayCustomToolAnimation

```
void Player_PlayCustomToolAnimation(const tString &in asAnim,
                                    bool abRemoveToolWhenOver,
                                    const tString &in asAnimOverCallback)
```

This only works if the player is drawing or is holding an item. If you want the hide animation to play after this animation is done, call Player_UnequipTool after calling this and amkes sure abRemoveToolWhenOver is false.

- **asAnim**: the name of the animaiton to play
- **abRemoveToolWhenOver**: If the tool should be removed when this animaiton has finished playing.
- **asAnimOverCallback**: played when the animation is over. syntax: void f(const tString& in asAnim)

Player_PickupProp

```
void Player_PickupProp(const tString &in asPropName,
                       const tString &in asCallback,
                       const float afTimeToPickup)
```

Initializes and starts a “pickup” animation for a the prop passed as argument.

- **asPropName**: name of the prop that will be animated.
- **asCallback**: name of the function that will be called when animation is done. Syntax “void MyFunc(const tString &in asEntity)”.
- **afTimeToPickup**: the total time of the animation.

Player_SetHealth

```
void Player_SetHealth(float afX)
```

Sets the player's health. If at 0 or below, then Player becomes dead.

Player_GetHealth

```
float Player_GetHealth()
```

Gets the health of the player

Player_AddHealth

```
void Player_AddHealth(float afX,
```

```
                        float afMinValue=-1)
```

Adds a value to the player's health. If at 0 or below, then Player becomes dead.

Player_SetMaxHealth

```
void Player_SetMaxHealth(float afX)
```

Sets the player's maximum health.

Player_GetMaxHealth

```
float Player_GetMaxHealth()
```

Gets the max health of the player

Player_GiveDamage

```
void Player_GiveDamage(float afAmount,
                       float afMinHealth,
                       eDamageType aType,
                       const tString &in asSource)
```

Gives damage to the player which applies a damage effect

- **afAmount**: how much damage to do to the player
- **afMinHealth**: the minimum health the damage can make
- **aType**: the type of damage, which damage effect should be applied to the screen
- **asSource**: the name of the entity doing the damage

Player_IsDead

```
bool Player_IsDead()
```

Returns true if the player is dead.

Returns: bool, true if player is dead.

Player_SetPitchLimits

```
void Player_SetPitchLimits(float afMin,
                           float afMax)
```

Restrict the camera's pitch (vs. world axes) - use 0,0 to turn off

---

Player_SetYawLimits

```
void Player_SetYawLimits(float afMin,
                         float afMax)
```

Restrict the camera's yaw (vs. world axes) - use 0,0 to turn off

---

Player_GetPitchLimits

```
cVector2f Player_GetPitchLimits()
```

Return the camera's pitch limits (vs. world axes) as a vector - x=min, y=max

---

Player_GetYawLimits

```
cVector2f Player_GetYawLimits()
```

Return the camera's yaw limits (vs. world axes) as a vector - x=min, y=max

---

Player_SetLadderMoveMul

```
void Player_SetLadderMoveMul(float afMul=1.0)
```

Player_StopDatamining

```
void Player_StopDatamining()
```