helper_ai.hps

Helper functions for AI

Agent_SetStaticCollider

```
void Agent_SetStaticCollider(const tString &in asAgentName,
                             bool abX)
```

Set if the agent will collide with the environment (objects and stuff will still collide with it!)

- **asAgentName**: name of the agent. Wildcard supported.
- **abX**: If the character should be static and NOT move about.

Agent_SetSensesActive

```
void Agent_SetSensesActive(const tString &in asAgentName,
                           bool abX)
```

Set if the agent should have its senses (hearing/vision/etc) enabled. Usually implemented on a specific basis.

- **asAgentName**: name of the agent. Wildcard supported.
- **abX**: If the senses should be active.

Agent_RevealPlayerPosition

```
void Agent_RevealPlayerPosition(const tString &in asAgentName,
                                bool bForceRedetect=false)
```

Reveals the player's current position to the agent.

- **asAgentName**: name of the agent. Wildcard supported.
- **bForceRedetect**: true = forces the agent to detect the player again.

Agent_SetViewRangeMul

```
void Agent_SetViewRangeMul(const tString &in asAgentName,
                           float afRangeMul)
```

Sets the view range multiplier of the specified agent

- **asAgentName**: name of the agent. Wildcard supported.
- **afRangeMul**: the range multiplier to set.

Agent_FaceEntity

```
void Agent_FaceEntity(const tString &in asAgentName,
                      const tString &in asLookAtTarget)
```

Instantly sets the yaw of the agent to make it look at the specified entity.

- **asAgentName**: name of the agent. Wildcard supported.
- **asLookAtTarget**: name of entity to look at.

Agent_TeleportFeetToEntity

```
void Agent_TeleportFeetToEntity(const tString &in asAgentName,
                                const tString &in asTargetName)
```

Teleport the agent's foot position to a specific other entity.

- **asAgentName**: name of the entity to teleport.
- **asTargetName**: name of the entity to teleport to.

Agent_SetAutoDisableWhenOutOfSightActive

```
void Agent_SetAutoDisableWhenOutOfSightActive(const tString &in asAgentName,
                                              bool abActive,
                                              float afMinDistance)
```

When active, the agent will be a disabled (SetActive(false)) when a certain distance away from the player and no longer seen.

- **asAgentName**: name of the agent.Wildcard supported.
- **abActive**: if the disabling is active
- **afMinDistance**: The minimum distance from the player for this to happen.

Agent_SetAutoDisableCallback

```
void Agent_SetAutoDisableCallback(const tString &in asAgentName,
                                  const tString &in asCallbackFunc)
```

Sets the function that is called when the disabling made by Agent_SetAutoDisableWhenOutOfSightActive happens

- **asAgentName**: name of the agent.Wildcard supported.
- **asCallbackFunc**: Function called when the disabling happens. syntax: void f(const tString&in asEntityName)

Agent_PlayerDetected

```
bool Agent_PlayerDetected(const tString &in asAgentName)
```

Checks if the agent has dected the player

- **asAgentName**: name of the agent.

CharMover_SetDirection

```
void CharMover_SetDirection(const tString &in asEntityName,
                            eLuxCharMoveDirection aDir)
```

Sets the direction used by a character

- **asEntityName**: name of the entity with the CharMover.
- **aDir**: direction that the character will face when walking

---

CharMover_TurnToEntity

```
void CharMover_TurnToEntity(const tString &in asEntityName,
                            const tString &in asTurnToEntityName,
                            const tString &in asTurnedToGoalCallbackFunc="")
```

Turns the character to a specifc entity. Broadcasts the entity message eLuxEntityMessage_TurningDone when done.

- **asEntityName**: name of the entity with the CharMover.
- **asTurnToEntityName**: the entity which center will be turn towards
- **asTurnedToGoalCallbackFunc**: function called when it has turned to the goal. Syntax: void Func(const tString& in asEntityName)

CharMover_TurnInstantlyToEntity

```
void CharMover_TurnInstantlyToEntity(const tString &in asEntityName,
                                     const tString &in asTurnToEntityName)
```

Turns the character directly to a specifc entity. Broadcasts the entity message eLuxEntityMessage_TurningDone when done.

- **asEntityName**: name of the entity with the CharMover.
- **asTurnToEntityName**: the entity which center will be turn towards

---

CharMover_MoveToEntity

```
void CharMover_MoveToEntity(const tString &in asEntityName,
                            const tString &in asMoveToEntityName,
                            const cVector3f &in avOffset=cVector3f())
```

Moves the character to a entity. Note that for non-flying characters, goal position should be the feet postion.

- **asEntityName**: name of the entity with the CharMover.
- **asMoveToEntityName**: the entity which center will be turn towards
- **avOffset**: an offset for the move postion.

CharMover_MoveToPos

```
void CharMover_MoveToPos(const tString &in asEntityName,
                         const cVector3f &in avPosition)
```

Moves the character to a position. Note that for non-flying characters, goal position should be the feet postion.

- **asEntityName**: name of the entity with the CharMover.
- **avPosition**: position to move to

CharMover_SetSpeedState

```
void CharMover_SetSpeedState(const tString &in asEntityName,
                             int alIndex)
```

Sets the speed state of the character.

- **asEntityName**: name of the entity with the CharMover.
- **alIndex**: speed state index

CharMover_SetUseMoveStateAnimations

```
void CharMover_SetUseMoveStateAnimations(const tString &in asEntityName,
                                         bool abX)
```

If the charmover should change animations depending based on movement.

- **asEntityName**: name of the entity with the CharMover.
- **abX**: if charmover shouuld take care of animations

CharMover_PlayAnimation

```
void CharMover_PlayAnimation(const tString &in asEntityName,
                             const tString &in asAnimName,
                             float afFadeTime=0.3f,
                             bool abLoop=false,
                             bool abPlayTransition=true,
                             const tString &in asCallback="")
```

CharMover_StopTurning

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

Stops any turning in the charmover

- **asEntityName**: name of the entity with the CharMover.

CharMover_SetTurnRate

```
void CharMover_SetTurnRate(const tString &in asEntityName,
                           float afMinBreakAngle,
                           float afBreakAngleMul,
                           float afTurnSpeedMul,
                           float afTurnSpeedMax)
```

Sets the turning behavior of the charmover.

- **asEntityName**: name of the entity with the CharMover.

- **afMinBreakAngle**: the minimum angle at which the charmover should slow down when turning.
- **afBreakAngleMul**: value multiplied by the angle to calculate how much the charmover should slow down when turning.
- **afTurnSpeedMul**: value multiplied by the angle to calculate how fast the charmover should turn.
- **afTurnSpeedMax**: the max turn speed.

CharMover_SetMaxForwardSpeed

```
void CharMover_SetMaxForwardSpeed(const tString &in asEntityName,
                                  float afMaxSpeed)
```

Sets the max forward speed of the CharMover.

- **asEntityName**: name of the entity with the CharMover.
- **afMaxSpeed**: max forward speed to set.

CharMover_SetWallAvoidanceActive

```
void CharMover_SetWallAvoidanceActive(const tString &in asEntityName,
                                      bool abX)
```

Enables or disables wall avoidance on the specified CharMover.

- **asEntityName**: name of the entity with the CharMover.
- **abX**: the state to set.

Pathfinder_MoveTo

```
void Pathfinder_MoveTo(const tString &in asEntityName,
                       const cVector3f &in avPos,
                       const tString &in asCallbackFunc,
                       float afUpdateFreq=,
                       bool abExactStopAtEnd=true,
                       const tString &in asResultCallbackFunc="")
```

Moves an entity containing a Pathfinder component to a position. The entity message eLuxEntityMessage_EndOfPath is broadcasted when complete.

- **asEntityName**: name of the entity with the pathfinder.
- **avPos**: position to move to.

- **asCallbackFunc**: The name of callback called when end is reach. Syntax: void Func(const tString& in asEntityName, bool abReachedEnd)
- **afUpdateFreq**: The number of times per second that the path will be updated. O or below means never update.
- **abExactStopAtEnd**: If the goal position should be hit exactly.
- **asResultCallbackFunc**: The name of callback called when the pathfinding has been performed. Syntax: void Func(bool abSuccesful)

Pathfinder_MoveToNode

```
void Pathfinder_MoveToNode(const tString &in asEntityName,
                           const tString &in asNodeName,
                           const tString &in asCallbackFunc="",
                           float afUpdateFreq=,
                           bool abExactStopAtEnd=true,
                           const tString &in asResultCallbackFunc="")
```

Moves an entity containing a Pathfinder component to a node. The entity message eLuxEntityMessage_EndOfPath is broadcasted when complete.

- **asEntityName**: name of the entity with the pathfinder.
- **asNodeName**: name of the node to move to.
- **asCallbackFunc**: The name of callback called when end is reach. Syntax: void Func(const tString& in asEntityName, bool abReachedEnd)
- **afUpdateFreq**: The number of times per second that the path will be updated. O or below means never update.
- **abExactStopAtEnd**: If the goal node should be hit exactly.
- **asResultCallbackFunc**: The name of callback called when the pathfinding has been performed. Syntax: void Func(bool abSuccesful)

Pathfinder_MoveToEntity

```
void Pathfinder_MoveToEntity(const tString &in asPathfinderName,
                             const tString &in asEntityName,
                             const tString &in asCallbackFunc,
                             float afUpdateFreq=,
                             bool abExactStopAtEnd=true,
                             const cVector3f &in
avGoalPosOffset=cVector3f(),
                             const tString &in asResultCallbackFunc="")
```

Moves an entity containing a Pathfinder component to a entity. The entity message eLuxEntityMessage_EndOfPath is broadcasted when complete.

- **asPathfinderName**: name of the entity with the pathfinder.
- **asEntityName**: name of the entity to move to.

- **asCallbackFunc**: The name of callback called when end is reach. Syntax: void Funcconst
- **afUpdateFreq**: The number of times per second that the path will be updated. O or below means never update.
- **abExactStopAtEnd**: If the goal entity should be hit exactly.
- **avGoalPosOffset**: if the goal pos should be offset
- **asResultCallbackFunc**: The name of callback called when the pathfinding has been performed. Syntax: void Func(bool abSuccesful)

Pathfinder_Stop

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

Stops any movement.

- **asEntityName**: name of the entity with the pathfinder.

Pathfinder_SetEndCallback

```
void Pathfinder_SetEndCallback(const tString &in asEntityName,
                               const tString &in asCallbackFunc)
```

Sets the callback for when the pathfinder has reached its goal.

- **asEntityName**: name of the entity with the pathfinder.
- **asCallbackFunc**: The name of callback called when end is reach. Syntax: void Funcconst

---

Pathfinder_IsMoving

```
bool Pathfinder_IsMoving(const tString &in asEntityName)
```

Checks if the pathfinder is currently moving.

- **asEntityName**: name of the entity with the pathfinder.

Pathfinder_Track_Clear

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

Clears all track nodes and stops track.

- **asEntityName**: name of the entity with the pathfinder.

Pathfinder_Track_Add

```
void Pathfinder_Track_Add(const tString &in asEntityName,
                          const tString &in asNodeName,
                          float afMinWaitTime=,
                          float afMaxWaitTime=-1,
                          const tString &in asAnimName="",
                          bool abLoopAnim=false)
```

Adds a track node to be followed.

- **asEntityName**: name of the entity with the pathfinder.
- **asNodeName**: Name of the path node to be added.
- **afMinWaitTime**: minimum time before the entity moves to the next node.
- **afMaxWaitTime**: maximum time before the entity moves to the next node. If <0, then the value will be same as afMinWaitTime
- **asAnimName**: animation played when at node, per entity.
- **abLoopAnim**: if the animtion should be looped, per entity.

Pathfinder_Track_Start

```
void Pathfinder_Track_Start(const tString &in asEntityName,
                            bool abLoop=false,
                            float afUpdateFreq=1.0f,
                            const tString &in asEndOfTrackCallback="")
```

Starts a track

- **asEntityName**: name of the entity with the pathfinder.
- **abLoop**: if the path should loop
- **afUpdateFreq**: the frequency at which the path is recculated
- **asEndOfTrackCallback**: Callback called when path is over. Syntax: void MyFunc(const tString& in asEntityName)

Pathfinder_Track_Stop

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

Stops the current track.

- **asEntityName**: name of the entity with the pathfinder.

BarkMachine_SetActive

```
void BarkMachine_SetActive(const tString &in asEntityName,
                           bool abActive)
```

Sets if the barkmachine is active (this will only affect random sounds)

- **asEntityName**: name of the entity with the barkmachine.
- **abActive**: if active or not.

BarkMachine_PlayVoice

```
void BarkMachine_PlayVoice(const tString &in asEntityName,
                           const tString &in asSubject,
                           int alPrio,
                           float afMinDistance=-1,
                           float afMaxDistance=-1,
                           float afMaxPlayerListeningRange=-1)
```

Plays a voice subject as a bark

- **asEntityName**: name of the entity with the barkmachine.
- **asSubject**: the name of the voice subject
- **alPrio**: the prio of the voice, if there is already one playing in the same scene prio must be higher for this to play.
- **afMinDistance**: The distance where volume starts getting quiet. If below 0, default is used.
- **afMaxDistance**: The maximum hearing range. If below 0, default is used.
- **afMaxPlayerListeningRange**: The maximum that subtitles are displayed. If below 0, default is used.

HeadTracker_SetActive

```
void HeadTracker_SetActive(const tString &in asEntityName,
                           bool abX)
```

Turns on and off head tracking.

- **asEntityName**: name of the entity with the HeadTracker.
- **abX**: if headtracking is enabled or not

HeadTracker_IsActive

```
bool HeadTracker_IsActive(const tString &in asEntityName)
```

If head tracking is active or not

- **asEntityName**: name of the entity with the HeadTracker.

HeadTracker_SetAngleOffset

```
void HeadTracker_SetAngleOffset(const tString &in asEntityName,
                                float afX)
```

Sets the offset of the headtracker angle useful if the character is not facing in the direction of the meshentity forward vector.

- **asEntityName**: name of the entity with the HeadTracker.
- **afX**: The angle offset, in degrees

HeadTracker_SetTargetEntity

```
void HeadTracker_SetTargetEntity(const tString &in asEntityName,
                                 const tString &in asTargetEntityName)
```

Sets the target of the head tracking

- **asEntityName**: name of the entity with the HeadTracker.
- **asTargetEntityName**: The target of the tracking

NPC_SetVoiceName

```
void NPC_SetVoiceName(const tString &in asNpcName,
                      const tString &in asVoiceName)
```

Sets the voice name for an NPC. This also automatically sets the source of the voice as the npc.

- **asNpcName**: name of the NPC.
- **asVoiceName**: name of the voice as defined in the voicehandler file..

NPC_IsTalking

```
bool NPC_IsTalking(const tString &in asNpcName)
```

Checks if the NPC is being talked to.

- **asNpcName**: name of the NPC.

NPC_SetCanBeTalkedTo

```
void NPC_SetCanBeTalkedTo(const tString &in asNpcName,
                          bool abX)
```

Sets if the NPC can be talked to (interacted with)

- **asNpcName**: name of the NPC. Wildcard supported.
- **abX**: if can be talked to.

NPC_SetMainAnimation

```
void NPC_SetMainAnimation(const tString &in asNpcName,
                          const tString &in asAnim,
                          bool abPlayTransition=true,
                          const tString &in asCallbackFunc="",
                          float afFadeTime=0.5,
                          bool abGlobalSpace=false)
```

Sets and plays the main animation of the NPC. This will also turn off any movement animations.

- **asNpcName**: name of the NPC. Wildcard supported.
- **asAnim**: The name of the animation
- **abPlayTransition**: If a transition animation should be played.
- **asCallbackFunc**: name of callback function. Only used if there is a transition animation. Syntax void Func(const tString &in asEntityName, const tString &in asAnimName)
- **afFadeTime**: time it takes to fade to the animation.
- **abGlobalSpace**: if the animation takes place in global space

---

NPC_PlayExtraAnimation

```
void NPC_PlayExtraAnimation(const tString &in asNpcName,
                            const tString &in asAnim,
                            float afFadeTime=0.3,
                            const tString &in asCallbackFunc="",
```

```
                                bool abGlobalSpace=false)
```

Plays an extra animation for the NPC. Will return to main animation (see NPC_SetMainAnimation) when done.

- **asNpcName**: name of the NPC. Wildcard supported.
- **asAnim**: The name of the animation
- **afFadeTime**: The time it takes to fade in the animation.
- **asCallbackFunc**: name of callback function. Syntax void Func(tString &in asEntityName, tString &in asAnimName)
- **abGlobalSpace**: the animation will be played in global space coordinates (?)

NPC_StopAllAnimations

```
void NPC_StopAllAnimations(const tString &in asNpcName,
                           float afFadeTime=0.0f)
```

Stops all animations for the NPC.

- **asNpcName**: name of the NPC. Wildcard supported.
- **afFadeTime**: The time it takes to fade out.

NPC_StopExtraAnimation

```
void NPC_StopExtraAnimation(const tString &in asNpcName,
                            float afFadeTime=0.0f)
```

Stops currently playing extra animation for the NPC.

- **asNpcName**: name of the NPC. Wildcard supported.
- **afFadeTime**: The time it takes to fade out.

NPC_PlayEmotion

```
void NPC_PlayEmotion(const tString &in asNpcName,
                     const tString &in asEmotion,
                     float afDuration,
                     float afWeight=1.0f,
                     float afFadeTime=0.25f)
```

Plays a face emotion for the duration

- **asNpcName**: name of the NPC. Wildcard supported.
- **asEmotion**: name of the emotion,

"Smile/Sad/Happy/Surprised/Wink/Wondering/Angry/Worried/Scared"
- **afDuration**: how long the emotion should be active
- **afWeight**: how strong the emotion should be
- **afFadeTime**: fade in and out time for the emotion

NPC_MoveToNode

```
void NPC_MoveToNode(const tString &in asNpcName,
                    const tString &in asNodeName,
                    const tString &in asCallbackFunc)
```

Moves the NPC to a path node.

- **asNpcName**: name of the NPC. Wildcard supported.
- **asNodeName**: The name of the path node.
- **asCallbackFunc**: The name of callback called when end is reach. Syntax: void Func(const tString& in asEntityName, bool abReachedEnd)

Critter_SetGroupEntity

```
void Critter_SetGroupEntity(const tString &in asCritter,
                            const tString &in asGroupEntity)
```

Sets the entity the critter should flock around

- **asCritter**: name of the Critter. Wildcard supported.
- **asGroupEntity**: The name of the group entity.

Critter_AddExtraEscapeEntity

```
void Critter_AddExtraEscapeEntity(const tString &in asCritter,
                                  const tString &in asEscapeEntity)
```

Adds an entity that the critter should try to avoid the same way it does with the player. (Only implemented in FishSmall right now)

- **asCritter**: name of the Critter. Wildcard supported.
- **asEscapeEntity**: The name of the entity to escape from
