Look at neo/d3xp/bot

1. BotBrain
2. BotManager
3. BotPlayer

BuddyBots API Docs

# Bots need 3 things by default:

## A definition in a def file

Example def file:

entityDef AssaultBot {

"inherit" "bot"

"author" "Dylan Rodarte"

"bot\_type" "Script"

"scriptclass" "AssaultBot"

"ui\_name" "AssaultBot"

}

As you can see the def file is a basic object consisting of:

The bots parent:FILLER

The author of the bot: This is just the name of the person that made it

Bot\_Type: FILLER

A scriptclass: This is the name of the script that will be loaded with this bot. For example this bot will have a “AssaultBot\_main.py” script in the same folder

Ui\_name: The name that is used to spawn that bot and the name that will appear when the bot is selected when it is spawned

## A main python file

The python file uses the API from BotBrain. Mainly Think, Spawn, and Restart. Information is accessed through self. See the BotPlayer documentation for more information about self.body

## A state machine file

State machines have 5 states:

1. Update
2. Init
3. Shutdown
4. DetermineNewState
5. Transition

# BotBrain

### Think

**Summary:** Tick function that determines what the bot does

**Input:** self, deltaTimeMS

**Output:** the moves the bot will make as aiInput\_t

**Example Call in script:**

def Think(self, deltaTimeMS) {

botInput = aiInput\_t()

(stuff to determine moves)

return botInput

}

### Spawn

**Summary:** Controls how the bot is spawned

**Input:** self, spawnDict

**Output:** none

**Example Call in script:**

def Spawn(self, spawnDict) {

(stuff)

return

}

### Restart

**Summary:** Controls what happens on restart

**Input:** none

**Output:** none

**Example Call in script:**

### OnPain

**Summary:** Controls what happens when the bot experiences pain

**Input:** idEntity\* inflictor, idEntity\* attacker, vec3 dir, vec3 damage

**Output:** none

**Example Call in script:**

### OnDisconnect

**Summary:** Controls what happens when a client disconnects

**Input:** int for the clientNum

**Output:** none

**Example Call in script:**

### OnKill

**Summary:** Controls what happens when a bot gets a kill

**Input:** idPlayer\* dead, idPlayer\* killer, vec3 dir, int damage

**Output:** none

**Example Call in script:**

### OnDeath

**Summary:** Controls what happens when a bot dies

**Input:** idPlayer\* dead, idPlayer\* killer, vec3 dir, int damage

**Output:** none

**Example Call in script:**

### OnHit

**Summary:** Controls what happens when a bot gets a hit

**Input:** idPlayer\* target, vec3 dir, int damage

**Output:** none

**Example Call in script:**

### OnRespawn

**Summary:** controls what happens when a bot respawns

**Input:** none

**Output:** none

**Example Call in script:**

# BotManager

## Possible Calls on afiBotManager

### GetWinningTeam

**Summary:** Returns an int corresponding to the winning team

**Input:** None

**Output:** an int for the winning team

**Example Call in script:**

winningTeam = afiBotManager.GetWinningTeam()

### GetFlagCarrier

**Summary:** Get the current player(or bot) with the flag

**Input:** int for the team you want to find the flag carrier for

**Output:** An idPlayer\* for the player with the flag for the given team

**Example Call in script:**

player = afiBotManager.GetFlagCarrier(self.myTeam)

### GetFlag

**Summary:** Get the flag for the provided team

**Input:** int for the team you want to find the flag for

**Output:**the flag for the team that you provided

**Example Call in script:**

self.ourFlag = afiBotManager.GetFlag(self.myTeam)

### GetFlagStatus

**Summary:** Get the status of the flag for the provided team

**Input:** int for the team you want to find the flag status for

**Output:** the status of the flag (FLAGSTATUS\_INBASE 0, FLAGSTATUS\_TAKEN 1, FLAGSTATUS\_STRAY 2, FLAGSTATUS\_NONE 3)

**Example Call in script:**

afiBotManager.GetFlagStatus(self.enemyTeam)

### ProcessChat

**Summary:** isn’t implemented right now

**Input:**

**Output:**

**Example Call in script:**

### InitBotsFromMapRestart

**Summary:** Gets called on map load, all queued bots are added

**Input:** None

**Output:** None

**Example Call in script:**

You wouldn’t

### GetPersistentArgs

**Summary:** Get the persistent args

**Input:** the int for the client you want to find args for

**Output:** an idCmdArgs\* with the args fort the client

**Example Call in script:**

You wouldn’t

### GetUsrCmd

**Summary:** get the saved cmd for a particular user

**Input:** the int for the client you want to find cmd for

**Output:** a usercmd\_t

**Example Call in script:**

You wouldn’t

### SetUsrCmd

**Summary:** Set the usrcmd for a particular client

**Input:** the int for the client you want to set the cmd for and the usercmd\_t

**Output:** none

**Example Call in script:**

You wouldn’t

### WriteUserCmdsToSnapshot

**Summary:** Its not implemented

**Input:**

**Output:**

**Example Call in script:**

### ReadUserCmdsFromSnapshot

**Summary:** Its not implemented

**Input:**

**Output:**

**Example Call in script:**

### AddBotInfo

**Summary:** add bot info to a loaded bot

**Input:** botInfo\_t\* for the info you want to add

**Output:** none

**Example Call in script:**

You wouldn’t

### AddRemoveInfo

**Summary:** add bot info to a removed bot

**Input:** botInfo\_t\* for the info you want to add

**Output:** none

**Example Call in script:**

You wouldn’t

### SpawnBrain

**Summary:** give a loaded bot a brain to control its actions

**Input:** idStr botName, int clientNum

**Output:** afiBotBrain\*

**Example Call in script:**

You wouldn’t

### FindBotProfile

**Summary:** get the bot profile for a bot with a certain name

**Input:** idStr botName

**Output:** botInfo\_t

**Example Call in script:**

You wouldn’t

### FindTeamProfile

**Summary:** get the team profile for a team with a certain name

**Input:** idStr teamName

**Output:** teamInfo\_t

**Example Call in script:**

You wouldn’t

### FindBotProfileByIndex

**Summary:** get the bot profile for a bot with a certain index

**Input:** int clientNum

**Output:** botInfo\_t

**Example Call in script:**

You wouldn’t

### FindBotProfileByClassName

**Summary:** get the bot profile for a bot with a certain class name

**Input:** idStr botClassName

**Output:** botInfo\_t

**Example Call in script:**

You wouldn’t

### ReloadPak

**Summary:** Reload a bot with a certain profile and clientNum

**Input:** botInfo\_t botProfile, int clientNum

**Output:** botInfo\_t

**Example Call in script:**

You wouldn’t

### LoadBot

**Summary:** load a bot from a pak

**Input:** idStr brainPakName, botInfo\_t outputBotProfile

**Output:** bool for whether or not the bot was loaded

**Example Call in script:**

You wouldn’t

### LoadTeam

**Summary:** load a bot team from a pak

**Input:** idStr teamPakName, botInfo\_t outputTeamProfile

**Output:** bool for whether or not the bot team was loaded

**Example Call in script:**

You wouldn’t

### LoadAllBots

**Summary:** loades every pak file it can

**Input:** none

**Output:** none

**Example Call in script:**

You wouldn’t

### GetBotNum

**Summary:** get the number of loaded bots

**Input:** none

**Output:** an int for the number of currently loaded bots

**Example Call in script:**

You wouldn’t

### GetBotName

**Summary:** get the name of a bot with a particular index

**Input:** an index for the bot

**Output:** and idStr with the name of the bot

**Example Call in script:**

You wouldn’t

### ParseForBotName

**Summary:** load a bot with the given name

**Input:** void\* defBuffer, unsigned bufferLength, const char\* name, idStr& botName, idStr& authorName, idStr& botType, idStr& botSpawnClass

**Output:** none

**Example Call in script:**

You wouldn’t

### ParseForTeamName

**Summary:** load a team with the given name

**Input:** void \* defBuffer, unsigned bufferLength, const char \* name, idStr & teamName, int & teamSize, idStrList& bots, idList<bool>& used

**Output:** none

**Example Call in script:**

You wouldn’t

# BotPlayer

## Possible Calls on self.body

### FindItem

**Summary:** Give the function the name of an item in string format and it will give you the corresponding game object

**Input:** A const char\*(a string) of the item you want to find

**Output:** idEntitiy\* (the game object corresponding to the item you wanted to find)

**Example Call in script:**

self.body.FindItem(“THING”)

### InView

**Summary:** Returns a boolean for whether not an entity is in the bots field of view

**Input:** An entity (literally any player, item, thing)

**Output:** boolean whether or not that entity is in the field of view

**Example Call in script:**

self.body.InView(enemyBot)

### MoveTo

**Summary:** Moves the bot to a position at a certain speed

**Input:** A position(in the form of a Vec3) and a speed(in the form of a float)

**Output:** None (has the side effect of changing the bots direction and speed)

**Example Call in script:**

self.body.MoveTo(enemyBot.GetPosition(),8)

### MoveToPosition

**Summary:** Move the bot towards a position for a certain distance

**Input:** A position(in the form of a Vec3) and a range(in the form of a float)

**Output:** A boolean whether or not the bot starts moving to the position (if true then it has the side effect of moving the bot)

**Example Call in script:**

self.body.MoveToPosition(enemyBot.GetPosition(),8)

### MoveToEntity

**Summary:** Move the bot to an entity (for a quarter of the radius?????)

**Input:** An entity (literally any player, item, thing)

**Output:** A boolean whether or not the bot starts moving to the entity(if true then it has the side effect of moving the bot)

**Example Call in script:**

self.body.MoveToEntity(enemyBot)

### MoveToPlayer

**Summary:** Move the bot towards the player with predictive pathing

**Input:** A player(or bot)

**Output:** A boolean for whether or not the bot is moving to the player(if true then it has the side effect of moving the bot)

**Example Call in script:**

self.body.MoveToPlayer(enemyBot)

### Attack

**Summary:** Tells the bot to start attacking

**Input:** None

**Output:** None (has the side effect of telling the bot to open fire)

**Example Call in script:**

self.body.Attack()

### StopAttack

**Summary:** Tells the bot to cease fire

**Input:** None

**Output:** None (has the side effect of telling the bot to cease fire)

**Example Call in script:**

self.body.StopAttack()

### Jump

**Summary:** The bot jumps

**Input:** None

**Output:** None (has the side effect of the bot jumping)

**Example Call in script:**

self.body.Jump()

### LookInDirection

**Summary:** The bot looks in the provided direction without focusing on a target

**Input:** A vec3 position

**Output:** None (has the side effect of turning the bot)

**Example Call in script:**

self.body.LookInDirection(enemyBot.GetPosition())

### LookAtPosition

**Summary:** The bot looks directly at the position

**Input:** A vec3 position

**Output:** None (has the side effect of the bot focusing on the position)

**Example Call in script:**

self.body.LookAtPosition(enemyBot.GetPosition())

### MoveToNearest

**Summary:** Move to the nearest instance of the provided game item

**Input:** A const char\*(a string) corresponding to the game object you want to move to

**Output:** An idEntitiy\* corresponding to the instance of the thing youre moving to

**Example Call in script:**

self.body.MoveToNearest(“THING”)

### Wander

**Summary:** Wander around the map

**Input:** Void

**Output:** Boolean if the bot can move

**Example Call in script:**

self.body.Wander()

### PathToGoal

**Summary:** Move the bot along a path towards a goal

**Input:** aasPath\_t &path, int areaNum, const idVec3 &origin, int goalAreaNum, const idVec3 &goalOrigin

**Output:** boolean if the bot is moving or not (side effect of the bot moving if true)

**Example Call in script:**

**?????????????????????????????????????????**

### ReachedPos

**Summary:** When moving towards the position just see if the bounding box touches the destination

**Input:** A position vec3 and a range (range for the bounding box)

**Output:** a boolean if the bot is touching the position or not

**Example Call in script:**

self.body.ReachedPosition(enemyBot.GetPosition(), 1)

### SwitchWeapon

**Summary:** Switch the equipped weapon to the one with the name provided

**Input:** char\* (string) of the weapon the equip

**Output:** boolean for whether or not the weapon was equipped

**Example Call in script:**

self.body.SwitchWeapon(“ROCKET\_LAUCHER)

### HasAmmo

**Summary:** See if the weapon with the provided name has ammo

**Input:** char\* (string) of the weapon to check ammo for

**Output:** and int of the ammount of ammo available

**Example Call in script:**

self.body.HasAmmo(“ROCKET\_LAUCHER)

### AmmoInClip

**Summary:** Reload the current weapon

**Input:** None

**Output:** None (has the side effect of reloading the weapon

**Example Call in script:**

self.body.AmmoInClip()

### FindNearByPlayers

**Summary:** Finds all players(or bots) that the bot can currently see

**Input:** None

**Output:** List of players(or bots) in the bots field of view

**Example Call in script:**

nearbyBotList = self.body.FindNearbyPlayers()

### FindItemsInView

**Summary:** Finds all the items that the bot can currently see

**Input:** None

**Output:** List of items(weapons/ammo/powerups/etc) in the bots field of view

**Example Call in script:**

nearbyItemList = self.body.FindItemsInView()

### GetPosition

**Summary:** Get the current position of the bot

**Input:** None

**Output:** A vec3 of the current position of the bot

**Example Call in script:**

self.body.GetPosition()

### NextWeapon

**Summary:** Equip the next weapon in the weapon selection

**Input:** None

**Output:** None (has the side effect of changing the currently equipped weapon

**Example Call in script:**

self.body.NextWeapon()

### UpdateAIMoveFlag

**Summary:** Change the current move state of the bot

**Input:** An aiMoveFlag\_t (can be NULLMOVE, CROUCH, JUMP, WALK, RUN)

**Output:** None, has the side effect of changing the bots current move state

**Example Call in script:**

self.body.UpdateAIMoveFlag( aiMoveFlag\_t.RUN )

### SaveLastTarget

**Summary:** Save the entity that is passed in for easy retrieval

**Input:** An idEntity that you want to save

**Output:** None (has the side effect of storing the provided entity)

**Example Call in script:**

self.body.SaveLastTarget(enemyBot)

### GetLastTarget

**Summary:** Retrieve the last entity that was save with SaveLastTarget

**Input:** None

**Output:** The idEntity that you previously saved

**Example Call in script:**

enemyBot = self.body.GetLastTarget()

### Health

**Summary:** Get the current health of the bot

**Input:** None

**Output:** The current health of the bot

**Example Call in script:**

Health = self.body.health

### Team

**Summary:** Get the current team classification of the bot

**Input:** None

**Output:** The current team classification of the bot

**Example Call in script:**

Team = self.body.team

### Spectator

**Summary:** See if the current bot is a spectator

**Input:** None

**Output:** If the current bot is a spectator or not

**Example Call in script:**

Spectator = self.body.spectator

# MISC.

## Using weapon calls:

Weapon scripts can be found in the assets folder.  
...buddybots\base\pak006.pk4  
  
In the archive under “script”, all of the weapons can be found for proper names.

weapon\_base.script

weapon\_bfg.script  
weapon\_chaingun.script  
weapon\_chainsaw.script

weapon\_fist.script

weapon\_flashlight.script  
weapon\_handgrenade.script  
weapon\_machinegun.script  
weapon\_pistol.script  
weapon\_plasmagun.script  
weapon\_rocketlauncher.script  
weapon\_shotgun.script

**Example Call in script:**

self.body.SwitchWeapon("weapon\_shotgun")

gun = self.body.FindItem("weapon\_rocketlauncher")