How to Set Up Vehicles for the AI in Battlefield1942™

Author: Tobias Karlsson

Abstract: This document will try to give a brief introduction to how to set up new vehicles for the AI of Battlefield1942. It is by no means exhaustive and is only intended as a support for people building new vehicles.

This document is part of a series of documents on the AI of Battlefield1942.

Setting Up Vehicles

When a new vehicle is created, it is not enough that it works for a human player for a bot to be able to use it. Adapting it for a bot requires three things:

- The right set of behaviours are available
- The right behaviours are assembled into a suitable vehicle type
- An AI Template for the vehicle must be created

When a bot enters a vehicle, it needs a lot of information to be able to correctly use the vehicle. This information is part of the AI Object attached to the vehicle. Possibly the most fundamental piece of information in the AI Object is the vehicle type. The vehicle type decides which set of behaviours the bot is going to use when it is in the vehicle. There is a wide range of vehicle types already defined (they can be found in *Game/AIBehaviours.con*). If no vehicle type suits the new vehicle, one will have to define a new one. There is range of behaviours available, but many of them are custom written for some of the available vehicles. Of course, it is probably still possible to create a number of new vehicle types, but they won't deviate too far from the present ones.

Adapting Individual Vehicles for AI

Most vehicles use most of what the AI Knowledge system has to offer. Since most other objects do not, it would have been a great waste of memory if all AI Objects could hold all the information that the vehicles need. To solve this problem, the AI Objects are able to have optional plug ins. A plug in is additional information that extends that of the AI Object. Many plug ins are only used by vehicles. As an example to illustrate and guide through the following description, we will use a Sherman tank.

Note that the term vehicle will be used as having the more general meaning player controllable position in the text below. This means that a passenger seat, or a static machine gun may and will be referred to as a vehicle.

Creating the Base AI Object

The file in which the AI Object Template is created should be located in a subdirectory to the vehicle called *AI*. The file should be named *Objects.con*. The first thing to create is the AI Object Template itself. Note that this definition is not at the top of the file since the plug ins must be created before they are added to the object template. The template definition for the Sherman looks like this:

```
aiTemplate.create Sherman
aiTemplate.addType ITUnit
aiTemplate.addType ITBiological
aiTemplate.addType ITGround
aiTemplate.addType ITMobile
aiTemplate.addType ITObstructedView
aiTemplate.degeneration 15
aiTemplate.allowedTimeDiff 2
aiTemplate.basicTemp 12
aiTemplate.commonKnowledge 0
aiTemplate.addPlugIn ShermanMobile
aiTemplate.addPlugIn ShermanPhysical
aiTemplate.addPlugIn ShermanCover
aiTemplate.addPlugIn ShermanTurret
aiTemplate.addPlugIn ShermanUnit
aiTemplate.addPlugIn ShermanCtrl
```

aiTemplate.create

- aiTemplate.create *name(string)*

The create command creates a new AI Template with the name *name*. It will fail if such a template already exists and all the following commands will be invalid. In the example, we create a new AI Template called “Sherman”.

aiTemplate.addType

- aiTemplate.addType *flag(enum)*

All the AI Objects have a number of flags that together builds a profile that is called the objects *information type*. The information type of an object is many times used to sieve through a lot of objects to find objects that can potentially fulfil the requirements of the search. There are a number of standards for which objects should have which flags, and if some important flags are set incorrectly, it could result in strange behaviour of the AI regarding that type of object. The possible flags are as follows:

- ITUnit
- ITAir
- ITGround
- ITNaval
- ITAirfield
- ITTransportation
- ITArtillery
- ITFixed
- ITCover
- ITMobile
- ITEditors
- ITUnmanned
- ITLowPriority
- ITStructure
- ITBiological
- ITVegetation
- ITSoldier
- ITObstructedView
- ITNoTemperature
- ITNoRender
- ITNoChildRender
- ITScaleRender

- ITSampling

Some of these flags have a special meaning and will be described in further detail below:

ITAir, *ITGround*, and *ITNaval* are essential for all vehicles (including stationary guns). These flags describe which element the vehicle moves in.

ITAirfield. This flag is used to mark vehicles that spawn aircrafts.

ITTransportation. This flag is used for vehicles that are used to transport troops (e.g. LCVPs and ACPs).

ITArtillery. This flag is used to mark the objects as being able to fire artillery fire.

ITFixed. This flag is used to mark the objects that are fixed to a position (e.g. Coastal Airguns, AA Gun emplacements).

ITCover. This flag marks the object as possible to take cover behind or in.

ITMobile. This flag marks the object as being able to move. If an object has this flag, it will not be rendered on the path finding map.

ITStructure. This flag marks the object as a structure. This must be set for any object that should be rendered as a house on the path finding map.

ITBiological. This flag is not used in the game, though it is usually used to mark the object as able to host a soldier.

ITVegetation. This flag is used to mark an object as part of the vegetation.

ITSoldier. This flag is used to mark an object as a soldier. It is not used for positions on vehicles where the soldier is exposed.

ITObstructedView. This flag is used to mark positions in vehicles where the view is partly obstructed by the vehicle (the driver position of a tank is a good example).

ITNoTemperature. If an object is marked with this flag, the object will not generate any heat in the strategic areas.

ITNoRender. If this flag is used, the object will not be rendered on the path finding maps.

ITNoChildRender. If this flag is used, the object's child object will not be rendered on the path finding maps.

ITScaleRender. If this flag is used, the object will be rendered on the path finding maps with a smaller brush than normal. This is used to make bridges possible/easier to traverse.

aiTemplate.degeneration

- aiTemplate.degeneration *seconds(float)*

This command is used the set how fast the bots forget the object once they cannot see it anymore. It takes 15 seconds for a bot to forget a Sherman.

aiTemplate.allowedTimeDiff

- aiTemplate.allowedTimeDiff *seconds(float)*

This command is used to tell the bots how often they should update the information they have about the object while it is in view. This is an optimisation since it takes some time to update an object, at the

same time, the bot's information should not grow too old, or they will start making erroneous decisions based on it. The Sherman's information must be updated every other second.

aiTemplate.basicTemp

- aiTemplate.basicTemp *temperature(float)*

This command is used to set a temperature that the object has before any modifications that the SAI may do. The temperature does not have a fixed scale; it is relative to any and all of the other temperatures in the world. Look at other objects to get a feeling for where a new object should end up on the temperature scale. The Sherman has a basic temperature of 12.

aiTemplate.commonKnowledge

- aiTemplate.commonKnowledge *bool*

If an AI Object type is of type common knowledge, booth sides will share the same, perfect information about it. This is an optimisation for saving memory. Common knowledge is used for all objects that do not change to any greater degree during play. The only objects that are not common knowledge are usually vehicles. The default value of common knowledge is *false*.

aiTemplate.Secondary

- aiTemplate.Secondary *bool*

This command marks the position as a secondary position. A vehicle can have one primary position, and any number of secondary positions. The primary position is assumed to be the driver of the vehicle, and the first position that is in the object's hierarchy in the objects *Objects.con*. Secondary positions do not have Physical, Cover or Mobile plug ins.

aiTemplate.addPlugIn

- aiTemplate.addPlugIn *name(string)*

This command is used to add a plug in to the AI Object Template. The command accepts all types of plug ins and they are referenced by their name *name*.

Creating a Plug In

Regardless of which plug in should be created the same command is used:

- aiTemplatePlugIn.create *plugInTyp(eum) name(string)*

The parameter *plugInType* decides what kind of plug in is to be created. *PlugInType* can have the following values:

- Armament
- ControlInfo
- ControlInfo3d
- Cover
- Mobile
- Physical
- Unit

The second parameter, *name*, must be a unique string that identifies the plug in.

The Mobile Plug In

The Mobile Plug In is used to add extra information about mobile objects.

```
aiTemplatePlugIn.create Mobile ShermanMobile
aiTemplatePlugIn.vehicleNumber 0
aiTemplatePlugIn.maxSpeed 16.0
aiTemplatePlugIn.turnRadius 5.0
aiTemplatePlugIn.coverSearchRadius 50.0
aiTemplatePlugIn.lodHeight 0.9
aiTemplatePlugIn.isTurnable 1
```

aiTemplatePlugIn.vehicleNumber

- aiTemplatePlugIn.vehicleNumber *vehicle(int)*

This command sets which path finding map that should be used with the vehicle. Exactly which path finding map the vehicle number refers to is defined in *AIPathfinding.con* of the level. The Sherman is a tank and hence it uses the number 0 path finding map. The fact that the meaning of this number is defined locally and the number is set globally for each vehicle is a little unfortunate. However, as long as one is aware of this fact, it should not cause any problems.

aiTemplatePlugIn.maxSpeed

- aiTemplatePlugIn.maxSpeed *maxSpeed(float)*

This command is used for the bots to know what the maximum speed is of a vehicle. Easiest way to find out a good value is to use the console command *console.showStats*, try the vehicle on a flat surface and see how fast it goes. The Sherman has a maximum speed of 16 m/s.

aiTemplatePlugIn.turnRadius

- aiTemplatePlugIn.turnRadius *radius(float)*

To plan their turns, the bots need to know the turning radius of their vehicles. This command set the turning radius to *radius*. A tank is able to turn almost on the spot, and has a small turning radius; the Sherman's turning radius is set to 5.0.

aiTemplatePlugIn.coverSearchRadius

- aiTemplatePlugIn.coverSearchRadius *radius(float)*

The cover search radius is a value that tells the bots how far away they are allowed to search for cover if the need arises when they are in the vehicle. The search radius for the Sherman is 50.0 meters.

aiTemplatePlugIn.lodHeight

- aiTemplatePlugIn.lodHeight *height(float)*

When a vehicle is loded, the AI replaces the physics system with a simpler physics. One thing that is replaced is the physics of the wheels, suspension and gravity that makes a vehicle look credible on the ground. The lod height *height* is an offset that is used to regulate how high a loded vehicle is above the ground. In order for the switching between loded and unloded Sherman to be seamless, the lod height is set to 0.9 meters.

aiTemplatePlugIn.isTurnable

- aiTemplatePlugIn.isTurnable *bool*

Some vehicles can turn without moving, these vehicles are generally tanks. If the vehicle is able to do this, it should be marked for the AI with the above command. The default value is *false*. The Sherman is a tank and is able to turn on the spot.

The Physical Plug In

The physical plug in is used for objects that have a physical representation in the world. This is almost every object.

```
aiTemplatePlugIn.create Physical ShermanPhysical
aiTemplatePlugIn.setStrType HeavyArmour
```

aiTemplatePlugIn.setStrType

- aiTemplatePlugIn.setStrType *armourType(enum)*

This option sets the armour type of the vehicle. The armour type is used to check which weapons that can damage the vehicle. The following armour types are available:

- Infantry
- LightArmour
- HeavyArmour
- NavalArmour
- Submarine
- Air

Note that submarines are not used with the AI in Battlefield1942, and this armour type should not be used either. The Sherman is a tank and therefore its armour is of type *HeavyArmour*.

The Cover Plug In

The cover plug in is used to mark an object as a potential cover.

```
aiTemplatePlugIn.create Cover ShermanCover
aiTemplatePlugIn.coverValue 4.0
```

aiTemplatePlugIn.coverValue

- aiTemplatePlugIn.coverValue *value(float)*

The cover value parameter specifies how good a cover an object is. The scale depends on the other objects, and the only way to get a good feeling for it, is to look at lots of objects.

The Armament Plug In

The armament plug in is used to mark an object as having one or more weapons. It should only be used if the actual position has a weapon (i.e. the driver of a M3A1 should not have an armament plug in, but the machine gunner on top should have one).

```
aiTemplatePlugIn.create Armament ShermanTurret
```

aiTemplatePlugIn.setIsAntiAircraft

- aiTemplatePlugIn.setIsAntiAircraft *bool*

If the weapon(s) of the vehicle can be used for attacking airborne vehicles, this parameter should be set to *true*. The default value is *false*, and as the Sherman is not suitable to attack flying airplane, this parameter is left with its default setting.

The Unit Plug In

The unit plug in is used to store information about how a vehicle is used from an administrative point of view of the AI.

```
aiTemplatePlugIn.create Unit ShermanUnit
aiTemplatePlugIn.equipmentType 0
aiTemplatePlugIn.setStrategicStrength 0 3
aiTemplatePlugIn.setStrategicStrength 1 3
aiTemplatePlugIn.setSelectKey PIMenuSelect1
```

aiTemplatePlugIn.equipmentType

- aiTemplatePlugIn.equipmentType *equipment(int)*

This command is one of the most crucial commands to get right. It is this command that details which set of behaviour that are supposed to be used with the vehicle. The different vehicle types are defined in *AIBehaviour.con*. Currently they are:

- 0 = Tank – This vehicle is a driver and gunner position of a ground vehicle that runs on tracks. Example: Panzer VI Tiger.
- 1 = Plane – This vehicle is a driver and gunner position in a plane. Example: Supermarine Spitfire.
- 2 = Boat – This vehicle is a driver and gunner position for a boat. Example: Yamato class battleship.
- 3 = Infantery – This vehicle is only used by infantry.
- 4 = Fixed – This vehicle is a standard fixed mounted weapon. Also used for tail gunners on airplanes, and secondary gunners on any ground vehicle that is not artillery. Example: Machine gun nests and Achi Val tail gunner.
- 5 = Car – This vehicle is a driver position for a wheeled land vehicle. Example: Kubelwagen.
- 6 = UnarmedTank – This vehicle is a driver position for a tracked ground vehicle. Example: Hanomag.
- 7 = LandingCraft – This vehicle is a driver position for a boat used for beach landings. Example: LCVP.
- 8 = Passenger – This vehicle is an unarmed passenger position for any kind of ground vehicle. Example: M3A1 passenger.
- 9 = BoatFixed – This vehicle is a gunner position for a boat. Example: Prince of Wales AA battery.
- 10 = LandingCraftPassenger – This vehicle is an unarmed passenger position for a boat used for beach landings. Example: LCVP passenger.
- 11 = LandingCraftFixed – This vehicle is a gunner position for a boat used for beach landings. Example: LCVP machine gunner.
- 12 = UnarmedBoat – This vehicle is an unarmed passenger position for boats.
- 13 = FixedLargeBore – This is a fixed mounted weapon with artillery capabilities. It could also be the gunner on a mobile artillery piece. Example: Coastal Gun, Wespe mobile artillery gun.
- 14 = ArtilleryDriver – This is an unarmed driver of a tracked vehicle whose gunner position has artillery abilities. Example: Wespe mobile artillery vehicle.
- 15 = UnarmedPlane – This is an unarmed pilot of an airplane. Example: C-47 Skytrain.

- 16 = AmphibiousCar – This is the driver of an amphibious vehicle. Example: LVT-4 Waterbuffalo.

- 17 = FixedGuidedMissileLauncher – This is a special vehicle for the Wasserfall TV-guided missile. Example: Wasserfall launcher.

The Sherman is a tank and uses the vehicle type 0.

aiTemplatePlugIn.setStrategicStrength

- aiTemplatePlugIn.setStrategicStrength *type(int) strength(int)*

This command sets the strategic strengths of the vehicle. The strategic strength is used by the SAI when it distributes its forces on the different tasks that it needs accomplished. There are two kinds of strategic strengths; offensive and defensive strength. The parameter *type* decides which of the two kinds of strengths that are to be set. *0* is used for denoting the offensive strength, and *1* for the defensive strength. The parameter *strength* is the value of the strength. This value is on a relative scale, and if it is a high or a low value is decided by all the other values on strategic strengths. To learn this scale, the only option available is to look at the other vehicles. Today, this scale is rather coarse. The Sherman is considered being equally good for offence and defence, and has a strategic strength of 3 in both cases.

aiTemplatePlugIn.setSelectKey

- aiTemplatePlugIn.setSelectKey *keyMapping(enum)*

This command is used to set the key that is used to switch to the position from another position in the same vehicle. There are nine possible keys to chose from (num key 1-9), and they are written on the form *PIMenuSelect1*, *PIMenuSelect2*, and so on.

setUseNoPathfindingToGetToObject

- setUseNoPathfindingToGetToObject *bool*

This command has a badly chosen name. It is used to mark vehicles that the bots are only allowed to enter from the rear. Examples of these vehicles are static machine guns and coastal defence guns. This parameter is set to *false* by default.

setHasExposedSoldier

- setHasExposedSoldier *bool*

This command is used to show that the driver of the vehicle is exposed and can be attacked by small arms fire. It is set to *false* by default.

setAlwaysTargetSoldier

- setAlwaysTargetSoldier *bool*

This is used if the vehicle is, or is being close to, indestructible. If it is set, the vehicle will never be targeted, instead only the soldier (that must be exposed) will be attacked. This is today only used for static machine guns. This command is set to *false* by default.

The ControlInfo Plug In

The controlInfo plug in is used to tell the bots some of the details of how the vehicle is controlled.

```
aiTemplatePlugIn.create ControlInfo ShermanCtrl
aiTemplatePlugIn.driveTurnControl          PIYaw
aiTemplatePlugIn.driveThrottleControl      PIThrottle
aiTemplatePlugIn.aimHorizontalControl      PIMouseLookX
aiTemplatePlugIn.aimVerticalControl        PIMouseLookY
aiTemplatePlugIn.lookHorizontalControl     PIMouseLookX
aiTemplatePlugIn.lookVerticalControl       PIMouseLookY
aiTemplatePlugIn.throttleSensitivity      -1.0
aiTemplatePlugIn.pitchSensitivity          0.21817
```

```
aiTemplatePlugIn.rollSensitivity           -0.21817
aiTemplatePlugIn.yawSensitivity            -2.5
aiTemplatePlugIn.lookVerticalSensitivity    0.21817
aiTemplatePlugIn.lookHorizontalSensitivity -0.21817
aiTemplatePlugIn.throttleLookAhead          1.0
aiTemplatePlugIn.pitchLookAhead             1.0
aiTemplatePlugIn.rollLookAhead              1.0
aiTemplatePlugIn.yawLookAhead               1.0
aiTemplatePlugIn.lookVerticalLookAhead      1.0
aiTemplatePlugIn.lookHorizontalLookAhead    1.0
aiTemplatePlugIn.throttleScale              1.0
aiTemplatePlugIn.pitchScale                 5.0
aiTemplatePlugIn.rollScale                  5.0
aiTemplatePlugIn.yawScale                   0.0020
aiTemplatePlugIn.lookVerticalScale          1.0
aiTemplatePlugIn.lookHorizontalScale        1.0
aiTemplatePlugIn.setCameraRelativeMinRotationDeg -360/-20/0
aiTemplatePlugIn.setCameraRelativeMaxRotationDeg 360/5/0
```

The Key Mappings

Several vehicles uses slightly different set-ups for their keys, and exactly what keys are used for what must be written in the con-file. The key mappings are not actual keys, but functionality that would be assigned to a key. The following keys are available:

- PIYaw
- PIPitch
- PIRoll
- PIThrottle
- PIMouseLookX
- PIMouseLookY
- PIFire
- PIAction
- PIUse
- PIMouseLook
- PIMenuSelect1
- PIMenuSelect2
- PIMenuSelect3
- PIMenuSelect4
- PIMenuSelect5
- PIMenuSelect6
- PIMenuSelect7
- PIMenuSelect8

- PIMenuSelect9
- PIAltFire
- PIReload
- PILie
- PICrouch
- PINone

aiTemplatePlugIn.driveTurnControl

- aiTemplatePlugIn.driveTurnControl *keyMapping(enum)*

This command sets which control that is used to turn the vehicle when driving. The Sherman uses the standard *PIYaw* to turn.

aiTemplatePlugIn.driveThrottleControl

- aiTemplatePlugIn.driveThrottleControl *keyMapping(enum)*

This command sets which control that is used to change the speed of the vehicle when driving. The Sherman uses the standard *PIThrottle* to control its speed.

aiTemplatePlugIn.aimHorizontalControl

- aiTemplatePlugIn.aimHorizontalControl *keyMapping(enum)*

This command sets which control that is used to target enemies with its gun on the horizontal axis. The Sherman uses the standard *PIMouseLookX* to do this.

aiTemplatePlugIn. aimVerticalControl

- aiTemplatePlugIn. aimVerticalControl *keyMapping(enum)*

This command sets which control that is used to target enemies with its gun on the vertical axis. The Sherman uses the standard *PIMouseLookY* to do this.

aiTemplatePlugIn.lookHorizontalControl

- aiTemplatePlugIn.aimHorizontalControl *keyMapping(enum)*

This command sets which control that is used to look around on the horizontal axis. The Sherman uses the standard *PIMouseLookX* to do this.

aiTemplatePlugIn.lookVerticalControl

- aiTemplatePlugIn.lookVerticalControl *keyMapping(enum)*

This command sets which control that is used to look around on the vertical axis. The Sherman uses the standard *PIMouseLookY* to do this.

Sensitivity, Look Ahead, and Scale

In many con-files, you may find a number of commands under the ControlInfo plug in whose names end with sensitivity, look ahead, or scale. The exact meaning of these commands is ancient arcane knowledge that has been lost in the mists of time. Their purpose is to let you control how (speed, smoothness, etc.) a bot moves turrets and vehicles. Most vehicles use the same values, and work reasonably well.

aiTemplatePlugIn.setCameraRelativeMinRotationDeg

- aiTemplatePlugIn.setCameraRelativeMinRotationDeg *rotation(Vec3)*

This command sets how much the camera of the vehicle can be rotated in the negative direction from its default position. Note that rotating upwards is in the negative direction. To find these values, look in the normal *object.con* file for the vehicle and at the limitations for the corresponding rotational bundles. The Sherman can rotate to any position horizontally, and a maximum of 20 degrees vertically.

aiTemplatePlugIn.setCameraRelativeMaxRotationDeg

- aiTemplatePlugIn.setCameraRelativeMaxRotationDeg *rotation(Vec3)*

This command sets how much the camera of the vehicle can be rotated in the positive direction from its default position. Note that rotating downwards is in the positive direction. To find these values, look in the normal *object.con* file for the vehicle and at the limitations for the corresponding rotational bundles. The Sherman can rotate to any position horizontally, and a maximum of 5 degrees vertically.

The ControlInfo3d Plug In

Since airplanes move in three dimensions instead of the two dimensions most vehicles use, they need somewhat more controls than a normal vehicle. This information is put into the ControlInfo3d plug in. It inherits the normal ControlInfo plug in, that means that it has the same information as a normal ControlInfo plug in, but it also extends it with some new. As an example, we will use the North American P-51 Mustang:

```
aiTemplatePlugIn.create ControlInfo3d MustangCtrl
aiTemplatePlugIn.driveTurnControl     PIYaw
aiTemplatePlugIn.driveThrottleControl PIThrottle
aiTemplatePlugIn.aimHorizontalControl PIYaw
aiTemplatePlugIn.aimVerticalControl   PIPitch
aiTemplatePlugIn.driveRollControl     PIRoll
aiTemplatePlugIn.drivePitchControl    PIPitch
aiTemplatePlugIn.aimRollControl       PIRoll
aiTemplatePlugIn.aimThrottleControl   PIThrottle
aiTemplatePlugIn.maxRollAngle         0.9900
aiTemplatePlugIn.maxClimbAngle        0.3333
```

Only the new commands, specific to ControlInfo3d, will be described below. For a description of the other commands, see the description of the ControlInfo plug in above.

aiTemplatePlugIn.driveRollControl

- aiTemplatePlugIn.driveRollControl *keyMapping(enum)*

This command specifies which control is used to roll the vehicle (that is to turn it around its direction of flight). The Mustang uses the standard *PIRoll* control.

aiTemplatePlugIn.drivePitchControl

- aiTemplatePlugIn.drivePitchControl *keyMapping(enum)*

This command specifies which control is used to Pitch the vehicle (that is to point the front up or down). The Mustang uses the standard *PIPitch* control.

aiTemplatePlugIn.aimRollControl

- aiTemplatePlugIn.aimRollControl *keyMapping(enum)*

Airplanes usually aim their weapons by positioning themselves so that their weapons point in the right direction. As with the steering, the aiming also needs more controls. This command sets the control for aiming by rolling. The Mustang uses the standard *PIRoll* control.

aiTemplatePlugIn.aimThrottleControl

- aiTemplatePlugIn.aimThrottleControl *keyMapping(enum)*

This command sets the control for controlling the speed of the vehicle while aiming. The Mustang uses the standard *PIThrottle* control.

aiTemplatePlugIn.maxRollAngle

- aiTemplatePlugIn.maxRollAngle *keyMapping(enum)*

This command sets the maximum angle that the vehicle is allowed to roll. If the vehicle is crashing while turning, this angle should be reduced. If it seems like the vehicle is turning too slowly, it might be worth trying to increase this angle. The angle is measured in radians. The Mustang's maximum allowed roll angle is 0.99 radians (about 57 degrees).

aiTemplatePlugIn.maxClimbAngle

- aiTemplatePlugIn.maxClimbAngle *keyMapping(enum)*

This command sets the maximum angle that the vehicle is allowed to climb. If the vehicle is frequently stalling, this angle should be reduced. If it seems like the vehicle is climbing too slowly, it might be worth trying to increase this angle. The angle is measured in radians. The Mustang's maximum allowed climb angle is 0.3333 radians (about 19 degrees).

Setting Up Weapons

The bots also need information about the weapons available in the vehicles in order to be able to use them. This information is created for each weapon and there is no need for double copies in the case of weapons that are shared between several vehicles. The information of the weapons is normally stored in a directory called '*AI*' in the directory that contains the corresponding weapon. To attach the AI information to the right object, the following line has to be in the .con-file in which that weapon is defined:

```
ObjectTemplate.AITemplate aiWeaponName
```

Where '*aiWeaponName'* is the name given to the ai weapon.

The ai weapon for the Sherman's main gun looks like this:

```
weaponTemplate.create ShermanMainGun
weaponTemplate.burst 0
weaponTemplate.indirect 0
weaponTemplate.minRange 2.0
weaponTemplate.maxRange 250.0
weaponTemplate.weaponActivate PINone
weaponTemplate.weaponFire PIFire
weaponTemplate.setStrength Infantery  10.0
weaponTemplate.setStrength LightArmour 7.0
weaponTemplate.setStrength HeavyArmour 2.0
weaponTemplate.setStrength NavalArmour 0.0
weaponTemplate.setStrength Submarine   0.0
weaponTemplate.setStrength Air         1.0
```

weaponTemplate.create

- weaponTemplate.create *name(string)*

This command creates a new AI weapon. *Name*, that has to be a unique identifier among the AI weapons, is the name of that weapon.

weaponTemplate.create

- weaponTemplate.activate *name(string)*

This command activates the AI weapon that is called *name*. If there is no such weapon, no weapon will be activated.

weaponTemplate.burst

- weaponTemplate.burst *bool*

This variable tells the AI if the weapon is possible to fire in automatic mode (if you can keep the button pressed and continue firing). If true, the bots will try and fire the weapon as an automatic weapon.

weaponTemplate.deviation

- weaponTemplate.deviation *deviation(float)*

This is a constant that decides how hard it is for the bots to hit with the weapon. Most weapons use the default deviation 5 meters/100 meter. This deviation will decrease over time and with the skill of the bots.

weaponTemplate.deviationCorrectionTime

- weaponTemplate.deviationCorrectionTime *time(float)*

This value defines how long time it takes a bot to reach maximum accuracy with the weapon after it first started aiming at a target. Most weapons use the default time of 10 seconds.

weaponTemplate.indirect

- weaponTemplate.indirect *bool*

If this value is set to true, the weapon is fired in the 45-90 degree range, if it is set to false, it is fired in the 0-45 degree range. Most weapons use direct firing.

weaponTemplate.healing

- weaponTemplate.healing *bool*

If this value is set to true, the weapon is giving negative damage to its target, i.e. it heals the target.

weaponTemplate.useAimerOnly

- weaponTemplate.useAimerOnly *bool*

This is a setting used for weapons that fire self-propelled projectiles. If it is set to true, the AI uses some special calculations to accommodate for the unusual trajectory when aiming. Note that most rockets (e.g. bazookas) are not self-propelled, they only have visual effects that make them look that way.

weaponTemplate.drag

- weaponTemplate.healing *drag(float)*

Normally, the weapon calculates the drag on its projectiles directly from the projectiles, themselves. However, sometimes this is not possible (e.g. with self-propelled projectiles), and this value has to be tweaked manually.

weaponTemplate.minRange

- weaponTemplate.minRange *range(float)*

This value gives the minimum effective range of the weapon. Bots will not fire the weapon at an opponent that is closer than this range.

weaponTemplate.maxRange

- weaponTemplate.maxRange *range(float)*

This value gives the maximum effective range of the weapon. Bots will not fire the weapon at an opponent that is farther away than this range.

weaponTemplate.weaponActivate

- weaponTemplate.weaponActivate *keyMapping(enum)*

This setting specifies which button is used to select the weapon. The most common mappings are *PINone*, *PIMenuSelect1*, and *PIMenuSelect2*.

weaponTemplate.weaponFire

- weaponTemplate.weaponFire *keyMapping(enum)*

This setting specifies which button is used to fire the weapon. The most common mappings are *PIFire*, and *PIAltFire*.

weaponTemplate.isThrown

- weaponTemplate.isThrown *bool*

This value is exclusively used by the hand grenades and allows for hand grenade specific attacking.

weaponTemplate.soundSphereRadius

- weaponTemplate.soundSphereRadius *radius(float)*

This value specifies how loud the weapon is. It is possible for a bot to hear the weapon in *radius* meters distance.

weaponTemplate.exitVelocity

- weaponTemplate.exitVelocity *velocity(float)*

For some weapons, it is not possible for the AI to calculate a proper exit velocity of the projectiles. This is mostly used for weapons that fire self-propelled projectiles.

weaponTemplate.setStrength

- weaponTemplate.setStrength *strengthType(enum) strength(int)*

This value specifies how effective the weapon is against a particular type of vehicles. The strength *strength* is not an absolute value, but a value that is relative to all the other strength values that are defined for the other vehicles. The possible strength types are the following:

- Infantry
- LightArmour
- HeavyArmour
- NavalArmour
- Submarine
- Air

Creating a New Vehicle

Creating a New Class of Vehicles

In order to create a new class of vehicles, it is necessary to extend *AIBehaviours.con* in order to create a new vehicle type. Defining a new vehicle type requires three things; declaring the vehicle class, adding the behaviours, and adding the plan interpreters.

Declaring the Vehicle Type

The first thing that is needed is to make room for the new vehicle type. The following command controls the number of available vehicle types:

- aiSettings.setNVehiclesTypes *maxTypes(int)*

Make sure that the number *maxTypes* is equal or greater than the number of vehicle types that is needed.

The next step is to declare the new vehicle type using the following command:

- aiSettings.setVehicle *id(int) name(string)*

The parameter *id* needs to be a number between 0 and one less than the number specified by the *setNVehiclesTypes* described above. The parameter *name* is the name that is associated with the new vehicle type. Both *id* and *name* must be unique.

Adding Behaviours to a Vehicle Type

It is possible to add up to eight behaviours to a vehicle type. There are not enough different behaviours available for there to be any sense in adding more behaviours. To add a behaviour, the following command is used:

- aiSettings. setVehicleBehaviour *vehicle (string) behaviour(string) urgencyGenerator(string) planGenerator(string) priority(int) parallellMask(int) urgencyCurve(string) inhibitors(string)*

The parameter *vehicle* is the name given to the vehicle with the *setVehicle* command above. The parameter *behaviour* is the name of a behaviour slot. The possible names are defined with the command *setBehaviour*, and can be find higher up in the *AIBehaviour.con* file. The behaviours currently available are:

- Avoid
- Change
- Fire
- Idle
- MoveTo
- Scout
- Special
- TakeCover

The parameter *urgencyGenerator* is used to define which behaviour implementation is to be used in which slot. The currently available behaviours will be described below:

BBAvoid

This behaviour is used for land and sea vehicles to avoid colliding with each other. It requires the vehicle to be mobile.

BBChange

This behaviour is used for all ground vehicles to decide whether they should change the vehicle or not.

BBChangeLandingCraft

This is a custom change behaviour for landing crafts. The bots will exit the landing craft if it touches land within a landing zone.

BBChangeTeleport

This is a custom change behaviour used by vehicles on ships. If the bot decides to change to a position that is not onboard the ship, it will disappear and spawn like it had died but without losing a ticket.

BBFire3d

This is a behaviour that is used by airplanes to attack enemies.

BBFireDriver

This is a custom behaviour that is used by drivers of self-propelled artillery to positioning their vehicle in a way that allows the gunner to attack.

BBFireInfantery

This is the standard attack behaviour for ground and sea vehicles, and second positions.

BBFireLargeBore

This is a behaviour that is used for attacking enemies for vehicles with artillery capacity. It allows the vehicle to engage enemies that are out of sight.

BBFireGuided

This is a special behaviour to fire guided missiles, and is currently only used by the Wasserfall launcher.

BBFireUnarmed

This is a special behaviour for unarmed vehicles. It is used to calculate how much the occupying bot would have liked to attack its enemies. This need is then used by the change behaviour to see if the bot is considering changing to another vehicle from which it can attack.

BBIdle

This is a default behaviour that is used if no other behaviour is viable. It is the same behaviour for all vehicles.

BBMedicAssist

This is a behaviour used for repairing other units. Both engineers and medics use it.

BBMoveTo

This behaviour is used for land and sea vehicles for following moving directions from the SAI. It requires that the vehicle is mobile.

BBMoveToFixed

This is a special behaviour for fixed vehicles. It is used to calculate how much the occupying bot would have liked to move if it could have moved. This need is then used by the change behaviour to see if the bot is considering changing to another vehicle with which could move.

BBScout

This is a behaviour used by most vehicles in order to keep track of their environment.

BBTakeCover

This is a behaviour used by most ground vehicles for finding and taking cover.

BBTakeCoverInfantry

This is a custom behaviour that lets infantry find and take cover both behind and in objects.

The parameter *planGenerator* is used to define the plan generator, an entity that take the output from a behaviour and generates a plan for how the bot is going to implement the decision of the behaviour. By necessity, each behaviour is tightly coupled to its corresponding behaviour. Even though each behaviour may have several plan generators, each plan generators usually do not suit more than one behaviour. The available plan generators are the presented below:

- BBPAvoidCollision2d – This is a plan generator for the BBAvoid behaviour, used by most round and seas vehicles.
- BBPAvoidCollisionInfantery – This is a custom plan generator for the BBAvoid behaviour used by infantry. The spelling should be incorrect.
- BBPChange – This is the plan generator for the BBChange behaviour.
- BBPChangeLandingCraft – This is the plan generator for the BBChangeLandingCraft behaviour.
- BBPChangeTeleport – This is the plan generator for the BBChangeTeleport behaviour.
- BBPFire2d – This is the plan generator for the BBFireLargeBore behaviour.
- BBPFire2dDriver – This is the plan interpreter for the BBFireDriver behaviour.
- BBPFire3d – This is the plan generator for the BBFire3d behaviour.
- BBPFireInfantery – This is a plan generator for the BBFireInfantery behaviour.
- BBPGotoWaypoint2d – This is a plan generator for the BBMoveTo behaviour that is used by ground and sea vehicles.
- BBPGotoWaypoint3d - This is a plan generator for the BBMoveTo behaviour that is used by airplanes.
- BBPGotoWaypointBoat – This is a plan generator for the BBMoveTo behaviour that is used by landing crafts only (in spite of the name).
- BBPGotoWaypointInfantery – This is a plan generator for the BBMoveTo behaviour that is used by infantry (note the (miss-) spelling).
- BBPIdle2d – This is a plan generator for the BBIdle behaviour used by ground and sea vehicles.
- BBPIdle3d – This is a plan generator for the BBIdle behaviour used by airplanes.
- BBPIdleInfantery – This is a plan generator for the BBIdle behaviour used by infantry (again, note the bad spelling).
- BBPMedicAssist – This is the plan generator used with the MedicAssist behaviour.
- BBPScout – This is a plan generator for the BBScout behaviour used by ground and sea vehicles.
- BBPScoutInfantery – This is a plan generator for the BBScout behaviour used by infantry.

- BBPTakeCover – This is the plan generator for the BBTakeCover behaviour used by ground and sea vehicles.

- BBPTakeCoverInfantry – This is the plan generator for the BBTakeCoverInfantry behaviour used by infantry (note the correct spelling).

- UnRelated – The behaviours BBFireUnarmed and BBMoveToFixed do not require a plan interpreter and take this argument instead.

The parameter *priority* is used to prioritise the behaviour in case that the bot lacks the execution time to examine all behaviours. The priority is relative to the priority to all the other behaviours' priorities. A behaviour with a lower priority is going to be examined more often in case of the system being starved for time.

The parameter *parallellMask* is a piece of legacy code and should be set to *0*.

With the parameter *urgencyCurve*, it is possible to change the behaviour's urgency development over time. The curve specified with *urgencyCurve* takes the time the behaviour has been active and returns a result that is multiplied with the behaviours urgency. If the behaviour is not the active behaviour, the parameter will not make any difference. The following predefined curves are available:

aiSettings.createUCConstant

- aiSettings.createUCConstant *name(string) k(int)*

This command creates an urgency curve named *name* that always returns the constant value *k*.

aiSettings.createUCLinear

- aiSettings.createUCLinear *name(string) k(int) m(int)*

This command creates an urgency curve named *name* that always returns the value $v = kt + m$. Where t is the time the behaviour has been active.

The parameter *inhibitors* is a set of multipliers that will affect the urgencies of each corresponding behaviour.

aiSettings.createBehaviourModifiers

- aiSettings.createBehaviourModifiers *name(string)*

This command creates a new set of inhibitors named *name*.

aiSettings.setBehaviourModifier

aiSettings.setBehaviourModifier *behaviour(string) modifier(float)*

This command sets the modifier for the behaviour placed on the place given by *behaviour*.

aiSettings.setVehicleDefaultBehaviour

- aiSettings.setVehicleDefaultBehaviour *vehicle(string) behaviour(string)*

This command sets the default behaviour for the vehicle *vehicle* to be the behaviour *behaviour*. The *Idle* behaviour is supposed as the default behaviour if no other behaviour is relevant.

aiSettings.addInterpreterEntry and aiSettings.activateDefaultInterpreter

These are settings that define which interpreters are needed for the bots to be able to interpret the plans the plan generators generate. To know exactly which interpreter entries that has to be activated for which plan generator requires in depth knowledge of the implementation of the plan generators. That is beyond the scope of this document, and I recommend looking at other vehicles that use the same plan generators.