GTA Mission Template Description

January 21, 1998

Note on this document

There is NO GUARANTEE the information in this document is correct. Use it at your own risk!!
ALWAYS make back-up copies of your mission.ini file.
Any commands noted as "UNSUPPORTED" may function incorrectly or not at all.

Note on Coordinates

All coordinates in the mission.ini are one of two types

- 0 - 255 (BLOCKS)
  These are block references. (0,0,0) would be the very top corner, (255,255,4) would be the very bottom right.
- 0 - 16320 (PIXELS)
  A more precise way of placing objects etc - this gives you an extra 64 pixels per block to place things.
  To convert between blocks & shifted coords, divide your PIXEL coord by 64.

Mission Object Creation

The basic format for Object creation is

{X,Y,Z} TYPE CLASS ANGLE

However various types use the Class and Angle fields for additional data, below is a description of all the types you can create, and what the parameters mean.

(X,Y,Z} **CAR** <model> <angle>   Creates a car of type <model> at facing <angle>

(X,Y,Z) **PED** <type> <angle> <remap> <enemy>

Creates a pedestrian of type <type> facing <angle> with remap <remap>.

Clarification on <enemy>

This parameter is needed for complicated ped types who 'work on' another ped.
The following types need <enemy>:

FOLLOW	Ped to follow
ESCAPE	Ped to runaway from
GUARD_FOLLOW	Ped to follow whilst in 'guard' mode
KILLER_BLOKE	Ped to try to kill for you

Suitable lines for <enemy> are PLAYER, PED, FUTUREPED. An enemy of -2 will make the ped choose another ped at random. Use at your own risk…
If the ped type is **NOT** one of these, miss it out **COMPLETELY**.

(X,Y,Z) are now PIXEL values like objects. Z value should be PHYSICAL level, e.g. 4 shifted up.

(X,Y,Z) **OBJECT** <type> <angle>

Creates an object of type <type> facing <angle>
Coords are in **pixels**, not blocks.

(X,Y,Z) **PLAYER** <car> <angle>

Creates a player pedestrian who owns <car> where car is the line number of it

(X,Y,Z) **DRIVER** 0 <car>

Creates a player pedestrian inside car <car>

(X,Y,Z) **PARKED** <model> <angle>

Creates a parked car of type <model> facing <angle>

(X,Y,Z) **TELEPHONE** <means> <angle>
Creates a telephone at <angle>. If <means> is greater than 0, then when the player is sent to answer this phone, how he got there will be checked
1 = by car
2 = by train
3 = on foot
Means checking is not guaranteed to work - it isn't used currently.

(X,Y,Z) **TRIGGER** <line> <range>
Creates a trigger that will start <line> line of Targets with a radius of <range> blocks in a 'square' range. A range of 0 means check **just** this block.

(X,Y,Z) **DOOR** <face> <type> <interior>
Creates a door of type <type> on face <face> of a building, with interior block <interior>. Doors are NASTY to place correctly, and they usually have a trigger tied to them.

(X,Y,Z) **TARGET** 0 0
Creates a target area used for explosions, destinations
Targets are normally declared in PIXELS, use a DUMMY for blocks.

(X,Y,Z) **FUTURE** <type> <angle>
Defines an object that can be created in the future (eg. a Suitcase)

(X,Y,Z) **COUNTER** <count> 0
Defines a counter that can be used for looped missions with the DECCOUNT command, e.g. for keeping track of the number of cars you've brought to a specific location.

(X,Y,Z) **SPRAY** <colour> <radius>
Creates a spray shop for <colour>

(X,Y,Z) **CARBOMB** <type> <angle>
Creates a car with a bomb in it.
Type can be the following:-
0 - No Bomb (disarm)
1 - Unarmed '5 second' bomb
2 - Armed '5 second' bomb - i.e. it will blow in five seconds
3 - Model Car Bomb (instant detonation on pressing 'special key')
4 - Dodgy Explosives (take some damage then it will blow)
5 - Unarmed Speed Bomb (Speed >¾ max speed to arm)
6 - Armed Speed Bomb (Speed <½ max speed to detonate)
N.B. This can be used with stationary vehicles to blow them up instantly

UNSUPPORTED!

(X,Y,Z) **BARRIER** <face> <num frames> <type>
Creates a barrier that can only be opened by emergency vehicles. <num frames> is the number of frames of animation. <type> is the interior block to use.

(X,Y,Z) **BOMBSHOP** 0 0
Creates a bomb shop trigger that places a bomb in your current car for a price!

(X,Y,Z) **STOPPED** <type> <angle>
Creates a stopped car, with a driver

(X,Y,Z) **DUMMY** 0 0
Does nothing, allows padding of lines of objects.

Coords are usually in BLOCKS.
This gets used a lot with GET_DRIVER_INFO - by calling a GET_DRIVER_INFO.

(X,Y,Z) **FUTUREPED <**type> <angle>
Data for a ped to be created at a later date.

(X,Y,Z) **CARTRIGGER** <line> <car>
Creates a trigger that will start <line> line when player enters <car> vehicle.

(X,Y,Z) **HELLS** <model> <angle>
Creates parked bike with driver of type <model> facing <angle>.

(X,Y,Z) **DRIVER_PARK** <model> <angle>
Creates a parked vehicle of <model> with driver.

(X,Y,Z) **FUTURECAR <**model> <angle>
Creates a future car of <model>, with driver.

(X,Y,Z) **ONETRIGGER** <line> <radius> <car num>

Creates a trigger of <radius> radius, that toggles the door defined at <line>.
It's triggered when the player drives car <car num> (the line num) to (X,Y,Z).
UNSUPPORTED

(X,Y,Z) **MODEL_BARRIER** <face> <type> <num frames> <car_type>

Creates a barrier that opens only for vehicles of model <car_type>. <num frames> is the number of frames of animation. <type> is the floor block to use.

(0,0,0) **MOVING_TRIG <**car_num> <radius> <hunt type>
Creates a trigger on car defined on line <car_num>. When any player gets within <radius> distance of this car, it will start hunting player down with a hunt of <hunt type>.

(X,Y,Z) **SPECIFIC_BARR** <face> <type> <num frames> <car num> <remap>
Creates a barrier that opens only for a specific vehicle declared in line <car num>. For the barrier to open, the car must have a remap value <remap>, set this to -1 to ignore the car colour. <type> is the interior block to use.

(0,0,0) **MPHONES**   <phone> <radius>
Creates a trigger for Multi-Phones which will set them to on whenever a player approaches the phone in line   <phone>, with radius <radius>.

(X,Y,Z) **MODEL_DOOR** <face> <type> <num frames> <interior> <car_type>
Creates a door that behaves in the same way as **MODEL_BARRIER.** See that for details, with <interior> the interior block to use.

(0,0,0) **GTA_DEMAND** <crane num> <model type> <remap> <number to get>
Creates a 'GTA Demand' for a crane. It basically means the player has to steal
<number to get> cars of model <model> of colour <remap> and take them to crane <crane>
[Crane is (0-3), NOT the line number. Get this by the order in which they're declared in your mission.ini]
[See Liberty Easy MPHONE 4 for example with stealing two taxis]

(0,0,0) **BOMBSHOP_COST** <cost> 0
Sets the cost value of using a bombshop to <cost>. Whenever a player uses a bombshop now, this value will be taken from his/her score.
This command isn't necessary, the game defaults to 5,000.

(X,Y,Z) **SPECIFIC_DOOR** <face> <type> <interior> <car num> <remap>

Creates a door that opens for a specific vehicle declared in line <car num>. For it to open, the car must have a remap value of <remap>, -1 to ignore. <type> is the floor block to use, <interior> is the 'floor' (I think….).

(X,Y,Z) **PHONE_TOGG** <phone num> <range>
This trigger has to be used to get the multiphones to ring again after a phone has been answered. Declare your MPHONES, then somewhere after it use this… <phone num> is the line num of the MPHONES declaration, <range> should be bigger than the MPHONE range by one or two, recommended value is 4/5.

(0,0,0) **TARGET_SCORE** <score> 0
Defines the score the player has to reach, for display on the pause screen.
SCORE_CHECK does the actual checking.

(X,Y,Z) **DUM_MISSION_TRIG** <line> <radius> <vehicle>
Defines a trigger on the block (x,y,z) with range <radius> that goes to line num <line>. It is triggered by the dummy vehicle defined in line <vehicle> entering the range of the trigger.

(X,Y,Z) **CORRECT_MOD_TRIG** <line> <radius> <model>
Defines a trigger on the block (x,y,z) with range <radius> that goes to line num <line> as usual. It is triggered if the player is driving a car with model number <model>.

(X,Y,Z) **CORRECT_CAR_TRIG** <line> <radius> <car>
Defines a trigger on the block (x,y,z) with range <radius> that goes to line num <line> as usual. It is triggered if the player is driving the car created on line <car>. Once triggered, disable it otherwise it will keep 'retriggering'.

(X,Y,Z) **MIDPOINT_MULTI** <next checkpoint> 0
Creates a trigger on block (X,Y,Z) that operates a 'checkpoint' in the multi-player games. First player to cross it gets a congrats message, and it switches on the next checkpoint, declared in line num <next checkpoint>. Further players get failure message and arrow pointing to next checkpoint.

(X,Y,Z) **MID_MULTI_SETUP <**first checkpoint> 0
This object is needed to setup the checkpoints properly, call it *after* you've done your sequence of checkpoints. <first checkpoint> is the line number of the first one, strangely enough.

(X,Y,Z) **FINAL_MULTI** 0 0
The 'finishing line' trigger for the a list of checkpoints…. It will congrat the first player and commiserate the stragglers. It keeps a record of positions, but at the moment does nothing with them.

(X,Y,Z) **POWERUP <**type> <timer>
Defines the location and info for a 'powerup icon'. (X,Y,Z) are in block coords (0-255),
<type> is the powerup type
For a Killfrenzy powerup, timer should be the number of frames (must be > 100)
For a certain amount of ammo, timer > 0 but < 100
For default amount, set timer to 0.

(X,Y,Z) **BLOCK_INFO** <info1> <info2>
This is used to declare the info for the "CHANGE_BLOCK" and "CHANGE_TYPE" commands.
For CHANGE_BLOCK, <info1> is the new value for the block., and <info2> is 0.
For CHANGE_FACE, <info1> is the face to change & <info2> is the new face value.

(X,Y,Z) **SPECIFIC_DOOR** <face> <type> <interior> <car num>
Creates a door that opens for a specific vehicle declared in line <car num>. For it to open, the car must have a bomb planted. <type> is the floor block to use, <interior> is the 'floor' (I

think….).

(X,Y,Z) **SETUP_SPEED** <range> 0

This defines and creates a disarm point on the map at (X,Y,Z) (Logical block level). This will be toggled on and off by the behind the scenes Speed mission…. If you don't have buses on your map, ignore this.

(0,0,0) **CARBOMB_TRIG** <car> <line>

Sets up a trigger that goes to line number <line> when the car defined in line <car> gets a bomb placed on it, e.g. when it enters a bombshop.

(X,Y,Z) **DAMAGE_TRIG** <line> <range>

This sets up a trigger that goes to line <line> when the player drives a car into the location defined with (x,y,z) & <range>, and blows it up. The car doesn't have to blow up with a bomb, it just has to have reached the maximum damage limit (100).
This command is a bit odd in execution and I'm warning you now it may act a little odd.

(X,Y,Z) **GUN_TRIG** <line> <range>

Sets up a trigger that goes to line <line> when the player starts shooting in location (X,Y,Z) & range (you know the score…).

(0,0,0) **GUN_SCREEN** <line> <ped>

Sets up a trigger that goes to line <line> when the player starts shooting and the ped defined in line <ped> is on-screen.

(X,Y,Z) **CARDESTROY_TRIG** <line> <range> <car>

Trigger that goes to line <line> when the car declared in line <car> is destroyed within the area defined by (X,Y,Z) & <range>.

(0,0,0) **CARWAIT_TRIG** <line> <car>

Sets up a trigger that goes to line <line> when the car defined in line <car> is stationary for a set period (currently 40 secs) and on screen.. as asked for by Steve. As soon as it goes offscreen, the counter is reset.

(0,0,0) **PEDCAR_TRIG** <line> <ped> <car>

Sets a trigger that goes to line <line> when the ped defined in line <line> enters the car <car>. This should be used with dummy peds, not the player - use a cartrigger for that.

(X,Y,Z) **PARKED_PIXELS** <model> <angle>

Creates a parked car of type <model> facing <angle>, with (X,Y,Z) in PIXEL coords.

(X,Y,0) **CHOPPER_ENDPOINT** 0 0

Sets the end destination for the end of level chopper. X&Y are in PIXEL coords.
You **MUST** define this line from now on!

UNSUPPORTED! (but should still work with START_CHOPPER….)

(X,Y,Z) **CANNON_START** 0 0

Needed for the cannon-ball runs to work properly - should just be the 'start' square of it.

(X,Y,Z) **DUM_PED_BLOCK_TRIG** <line> <radius> <dummy ped>

Sets up a trigger that goes to line <line> etc, when the dummy ped in line <dummy ped> enters the block/radius defined…

(X,Y,Z) **BASIC_BARRIER** <face> <num frames> <type>

Creates a barrier that will open for ANY vehicle. <num frames> is the number of frames of animation. <type> is the interior block to use.

(0,0,0) **SECRET_MISSION_COUNTER** <total> 0

Operates like a normal counter, but is a count of the **SECRET** missions e.g. kill frenzies etc.

Targets/Commands

Each objective is specified in the following manner

TYPE *Object* *Succeed* *Fail* *Data* *Score*

Below is a list of the Currently Available Mission Objectives and how they are used.

LOCATE <object> <succ> <fail> <timer> <score>

Waits for the player to reach a certain location on foot, or for the timer to run out, if the player reaches the location it goes to the <succ> objective otherwise it goes to the <fail> objective. A timer of 0 will disable the timer. Note: **PLAYER MUST BE ON FOOT** for this to succeed…

DESTROY <object> <succ> <fail> <timer> <score>

Wait for the player to destroy a certain object, or for the timer to run out, if the player destroys the object it goes to the <succ> objective otherwise ot goes to the <fail> objective. A timer of 0 will disable the timer.

ANSWER <phone> <succ> <fail> <rings> <score>

Starts a telephone ringing, usually used after **SURVIVE** to form a delay + telephone ring, but can be triggered by a TRIGGER, the phone will ring rings/2 times. If you answer it goes to <succ> objective otherwise it goes to <fail> objective.

STEAL <car> <succ> <fail> <timer> <score>

Waits for the player to steal the <car> specified, i.e. Enters the car. If timer is set to 0 there is no time limit.

SURVIVE 0 <succ> 0 <timer> <score>

Survive for <timer> and it goes to <succ>, if you dont survive, your dead anyway.

HUNTON <car> <succ> <type> <hunted> <score>

Starts <car> hunting <hunted>, if this car was previously parked, you will need to use DUMMYON to start the car first. It always goes to <succ>. <type> is the type of hunt as listed in the briefing.txt file.

HUNTOFF <car> <succ> 0 0 <score>

Stops <car> hunting, turns it back into a dummy car, it always goes to <succ>.

START_DRIVER <ped> <succ> 0 <car> <score>

Creates driver object and sends him to <car>. It always goes to <succ>.

DUMMYON <car> <succ> 0 0 <score>

Starts <car> driving as a normal dummy car. It always goes to <succ>

SENDTO <car> <succ> 0 <dest> <score>

Sends <car> to <dest>. It always goes to <succ>.
[See notes]

END <code> 0 0 <timer> <score>

Sets the game to terminate after <timer>, returns <code>, 1 - succeed, 2 - failed. No longer needed.

MAKEOBJ <obj> <succ> 0 0 <score>

Creates the FUTURE object <obj> on the map. It always goes to <succ>

EXPLODE <target> <succ> 0 <face> <score>

Causes the building at <target> to explode on its <face> side, It always goes to <succ>

OBTAIN <obj> <succ> 0 0 <score>
Creates the object <obj> in the players hands, useful for bombs, suitcases and other things that need to be thrown. It always goes to <succ>. **THIS COMMAND IS BOLLOCKS**.

THROW <obj> <succ> 0 <car> <score>
Causes the player to throw <obj> at <car>. It always goes to <succ>

BRIEF <target> <succ> <dest_loc> <time> <type>
Produces a message telling the player where to go and how long he's got to get there. It always goes to <succ>. At this stage, you should be using your 'special' briefings, e.g. from 1001 upwards.

MPHONE <phone> <succ> <phones> <rings> <score>
Allows a group of phones to ring simultaneously,, allows the player to answer only one phone, then runs the mission <succ>+<answered>-<phone> (i.e. <succ> plus the number of the phone from the MPHONES point of view <phone> being 0.

DONOWT 0 0 0 0 <score>
The simplest thing is to do nowt, this does nothing at all, it sits there forever, useful for linking several triggers, or waiting for the player to trigger a mission.

DISABLE <trig> <succ> 0 0 <score>
Disables trigger <trig> so that it cannot be used (allows mission areas to be switched off).

ENABLE <trig> <succ> 0 0 <score>
Re-enables trigger <trig> so that the mission can come back on again. This will also 'look up' the relevant info for the trigger if the trigger is using a FUTURECAR, PED or whatever.

In other words, if you're pointing the trigger to a FUTURECAR, do an **ENABLE** once you've done CAR_ON and the correct info will magically appear in the trigger.

DECCOUNT <counter> <succ> <fail> 0 <score>
Decrements the counter <counter> if it reaches zero the it goes to <succ> otherwise it goes to <fail>.

GOTO <target> <succ> <fail> <time> <score>
Waits for the player to get to <target> by any means possible - player can be inside a car and it will succeed. Use LOCATE for 'on foot only'.

CRANE <crane> <succ> <fail> <trigger> <score>
Starts a GTA crane lifting any vehicle that may be beneath its trigger <trigger>, this will end when the crane fails to find a car, or has lifted the car and is about to move it back onto its ferry.

PARK <door> <succ> <COORDS> <face> <score>
Displays animation of your current car parking in the garage at <door>, then of the player walking out again.

<face> is the face that the door is on. Values are
0 - up, 1 - right, 2 - down, 3 - left

CHECK that you have the correct info in this line matching the definition of the doorway in line <door> before telling me that PARK isn't working, please.

<COORDS> is a line number of a DUMMY that is set to the coords of the garage - i.e. the 'location' inside that the actual park command takes place on, that make sense??

If it is set to 0, it will look up the trigger that 'started' this process & use its coords… This it the existing code which is causing all the weird problems.

SETBOMB <car> <succ> <fail> <bomb> <score>

Sets the car bomb value of <car> to <bomb>, some useful values of bomb are

0 - No Bomb (disarm)
1 - Unarmed '5 second' bomb
2 - Armed '5 second' bomb - i.e. it will blow in five seconds
3 - Model Car Bomb (instant detonation on pressing 'special key')
4 - Dodgy Explosives (take some damage then it will blow)
5 - Unarmed Speed Bomb (Speed >¾ max speed to arm)
6 - Armed Speed Bomb (Speed <½ max speed to detonate)
N.B. This can be used with stationary vehicles to blow them up instantly

ARROW <target> <succ> <fail> 0 <score>

Points an arrow at line <target>, this should not be used with moving targets. <target> should have a suitable (X,Y,Z)

EXPL_LAST 0 <succ> <fail> 0 <score>

Explodes the last car the player controlled - at the moment always proceeds to <succ>.

Complaints about the effect of this command, please DO NOT use it. With a bit of planning, you can usually come up with something less 'random' looking.

DROP_ON <object> <succ> <fail> 0 <score>

Creates the 'drop off' point <object>. To use this, create your dropoff point as a FUTURE object, then call this when player reaches final telephone (or whatever). Means player cannot see the drop off until it's time.

KILL_DROP <object> <succ> <fail> 0 <score>

Kills the 'drop off' point <object> that has been turned on with DROP_OFF previously. If successful, goes to <succ> else goes to <fail>, e.g. if not previously turned on, it'll go to fail.

PED_ON <futureped> <succ> <fail> 0 <enemy>

Creates the FUTUREPED futureped.
If ped type is FOLLOW, <enemy> is the ped to follow
DRIVER, <enemy> is the car to drive.
[See **PED** for what <enemy> should be, as <enemy> behaves like <enemy>]

CAR_ON <futurecar> <succ> <fail> 0 <score>

Creates the FUTURECAR futurecar, WITH a driver.

ARMEDMESS 0 <succ> <fail> 0 <score>

Displays the message "Bomb Armed"

DISARMMESS 0 <succ> <fail> 0 <score>

Displays the message "Bomb Disarmed" and disarms any bomb on the player's current car. If there's no bomb or the player isn't in a car, it goes to <fail>

ARROW_OFF 0 <succ> <fail> 0 <score>

Removes the arrow if it's switched on.

P_BRIEF <target> <succ> <dest_loc> <time> <type>

Displays a 'BRIEF' style message on the Pager. Should behave **exactly** as the BRIEF command, so look at that for specific information.

PED_SENDTO <point> <succ> <fail> <ped_num> <score>

Sends the ped defined in line <ped_num> to the point defined in line <point>. This line **doesn't** wait for the ped to reach that point, go to a **WAIT_FOR_PED** to do that.
Uses the 'logical z' value, and sends the ped to the *centre* of the block declared in <dest>.

WAIT_FOR_PED <loc> <succ> <fail> <ped_num> <score>
Waits for the ped defined in line <ped_num> to reach the point defined in line <loc>.

KILL_OBJ <obj_num> <succ> <fail> 0 <score>
Kills the object declared in line num <obj_num>.

PED_BACK **<**car_num> <succ> <fail> <ped_num> <score>
Sends the ped in line <ped_num> to the back door of car <car_num>. Proceeds to <success> without delay.

MISSION_END 0 <succ> 0 0
Explicitly tells the game that this mission has been successfully completed - **IMPORTANT!** You **MUST** do one of these lines at the end of each mission. Always goes to <succ>.

EXPLODE_CAR <car num> <succ> <fail> 0 <score>
Explodes the car defined in line <car num>, proceeding to <succ> if exploded & adding <score>, otherwise, eg if car not created, to <fail>.

ARROWCAR <car num> <succ> 0 0 <score>
Points the arrow to the car defined in line <car num>, following it around if the car moves.

ARROWPED <ped num> <succ> 0 0 <score>
Points the arrow to the ped defined in line <ped num>, following it around if the ped moves.

DO_GTA <gta demand> <succ> 0 <score>
Switches 'on' the GTA demand declared in line <gta demand>. This basically means that the crane involved will be looking for the target, giving a bonus when it's reached. This gives **NO** message upon starting, do one yourself before it. It will do a "Just one more…" when there's one more to, and a "completion" style message when the player's completed the order.

REMAP_PED <ped num> <succ> <fail> <new remap> <score>
Changes the remap value for the ped declared in line <ped num> to value <new remap>. Goes to <succ> if successful, <fail> if the ped hasn't been created.

REMAP_CAR <car num> <succ> <fail> <new remap> <score>
Changes the remap value for the car declared in line <car num> to value <new remap>. Goes to <succ> if successful, <fail> if the car hasn't been created.

THROW_TO_POINT <obj> <succ> 0 <location> <score>
Causes the player to throw <obj> to the (x,y,z) declared in line <location> - these are in pixels… It always goes to <succ>. Doesn't throw very far at the moment - will be worked on shortly. Again, maybe best to avoid this if possible.

ARE_BOTH_ONSCREEN <car one> <succ> <fail> <car two> <score>
Checks to see if the cars declared in lines <car one> & <car two> are both onscreen at this particular moment. If so, it goes to line <succ>. If not, or if either car has not been created, it goes to line <fail>.

This (hopefully) will give you something to use for the end of the taxi following mission - set up a trigger on the block the taxi's going to which will call this, and it'll do a branch to wherever you want, IYKWIM.

GOTO_DROPOFF <drop point> <succ> <fail> <timer> <score>
Makes the player go to the drop off point defined in line <drop point> in <timer> 'ticks'. If

fail, go to <fail> else <succ>.
Will eventually ‘animate’ the drop off point like the phones. Not at the moment though.

MODEL_HUNT <first car> <succ> 0 <number> <score>
This routine handles the ‘Dead Pool’ style attack. Line <first car> is where the model cars are declared, with <number> the number of them. The cars MUST be declared in consecutive lines, with consecutive numbers
eg

(…) MODEL_CAR ….. {296}
(…) MODEL_CAR ….. {297}

It will start them hunting, with them blowing up when they're underneath the player. When all are destroyed, it will proceed to line <succ>.

MODEL_FUTURE <car line> <succ> <fail> 0 <score>
Creates a model car declared as a FUTURECAR in line <car line>, going to line <succ> if created okay, otherwise line <fail>. Use this BEFORE MODEL_HUNT to ‘switch on’ the model cars.

STARTUP <trigger line> <succ> -1 0 0
Special trigger ‘case’ - use this to branch to a part of your code at the start of the game, eg to switch off certain triggers etc. <trigger line> is the line where the ‘startup’ trigger is declared (the one placed where the player starts), <succ> is the line num to branch to. See the note below for more details.

START_HIRED_ESC <car> <succ> 0 <dest> <score>
Orders the dummy vehicle defined in line <car> to go into ‘Hired Gun’ Escape style mode, i.e. the car will attempt to get to a specific location as quickly as possible. Will go to line <succ>. As with all hired escapes, I cannot guarantee that the car will get to its destination, nor the quality of its escape - it depends on the road layout, the traffic, the weather.

POWERUP_ON <powerup> <succ> 0 0 <score>
Creates the ‘power-up’ defined in line <powerup>. Goes to <succ>, adding <score>.

PLAIN_EXPL_BUILDING <target> <succ> 0 <face> <score>
Causes the building at <target> to explode on its <face> side, without producing any rubble or plant-pots. It always goes to <succ>

CHANGE_BLOCK <block def> <succ> 0 0 <score>
Changes the block defined in line <block def> to the new type declared in that line. Goes to <succ>, adding <score> as usual.

CHANGE_TYPE <face def> <succ> 0 0 <score>
Similar in construction to CHANGE_BLOCK…. Changes the face defined in line <block def> to the new face as defined in that line.

These two commands can be used to temporarily manipulate the physical map structure, e.g. to ‘destroy’ bridges.

P_BRIEF_TIMED 0 <succ> 0 <time> <type>
Displays a pager briefing of type <type> that finishes with a countdown of <time> secs (roughly). Now proceeds straight away to <succ>.

Don't use any of the fancy briefing constructions with this, stick to fairly simple ones…

CHANGE_PED_TYPE <ped> <succ> -1 <new type> <data>
Changes the type of the ped declared in line <ped> to type <new type>. If new type is a follow,

<data> should be the line num that the ped to follow was declared on.
Works for all ped types.

DOOR_ON <door num> <succ> 0 0 <score>
Switches the door defined in line <door num> ‘on’, i.e. it will now open/close when its requirements are met. If the door is in the middle of closing, it will wait till the animation is finished, then switch it on.

DOOR_OFF <door num> <succ> 0 0 <score>
Switches the door defined in line <door num> ‘off’, i.e. it will stay closed no matter what. Use this to prevent players entering doors at the wrong time. If the door is in the middle of closing, it will wait till the animation is finished, then switch it off.

WRECK_CURR_TRAIN 0 <succ> <fail> 0 <score>
“Wrecks” the train the player is currently on. If the player isn’t on a train, it goes to <fail>, otherwise to <succ> adding <score>. Not much use to be honest.

PED_OUT_OF_CAR <car num> <succ> 0 0 <score>
Chucks the driver of car <car num> out of the car, going to <succ> when the animation has finished.
If you make a ped a driver, he will go and get into that specific car….

SETUP_REPO <car> <succ> 0 <ped> 0
“Sets up” a repo-man mission, <car> should be the line the car’s declared in, <ped> is the line the future ped’s declared in.

DO_REPO <car> <succ> 0 <timer> <score>
This handles an old style repo-man mission. Initially, it does a timer of <timer> frames, when this finishes it creates the driver ped (declared as a FUTUREPED) and sends him to the car. When he reaches the car, it drives off.
When this is all happening, it’s checking for the player stealing the car - at this point it’ll go to line <succ>, adding <score>.

OPEN_DOOR <door> <succ> <fail> 0 <score>
Manually force the door <door> to start opening. If things go okay, it goes to <succ>, adding <score> as usual, otherwise it goes to <fail>.

CLOSE_DOOR <door> <succ> <fail> 0 <score>
Manually force the door <door> to start closing.

BANK_ROBBERY <car> <succ> 0 0 <score>
Reports a bank robbery crime, with the car defined in line <car> the culprit, e.g. if player has to make it to the bank in car 100, report the crime on that car. This also causes an alarm to

NO LONGER sets off an alarm, use **BANK_ALARM_ON** for that.

GET_DRIVER_INFO <car> <succ> 0 <dummy> <score>
This is used to ‘get’ the info for a driver. It takes the car defined in line <car> and sets the relevant info in line <dummy> which should have been defined as a DUMMY.

[Use this, e.g., when you’ve done a dummy_on and need to refer to the driver ped.]

GET_CAR_INFO <ped> <succ> 0 <dummy> <score>
This is used to ‘get’ the info for a car. It takes the ped defined in line <ped> and sets the relevant info in line <dummy> which should have been defined as a DUMMY.

DELAY_CRIME <criminal> <succ> <type> <delay> <score>
Delays the reporting of a particular crime…. It isn’t very flexible I’m afraid, more to do with the

crime side of things rather than mission code.
<criminal> is the line number of the criminal, NOT the player but the car you expect him to be in, e.g. the line num of a car which he has to use to 'do' a bank job. <type> is the time of crime to watch out for, see below for values. <delay> is the number of frames to wait until the crime gets reported.

Possible 'types' -	CARJACKING	5
	MURDER	6
	BANK ROBBERY	9
	CONVOY ATTACK	10

KILL_CAR <car num> <succ> 0 0 <score>
Kills the car defined in line <car num>.

KILL_PED <ped num> <succ> <fail> 0 <score>
Kills the ped defined in line <ped num>. If it doesn't exist, it will go to <fail>.

GARAGE_SEND <car> <succ> 0 <dest> <score>
Use this for the 'dummy parking in garages' gumpf instead of START_HIRED_ESC.

PLAYER_ARE_BOTH_ONSCREEN <car one> <succ> <fail> 0 <score>
Checks to see if the current player and the car declared on line <car one> are both onscreen at this particular moment…. The player cannot be on a train, only in a car or on foot

CHECK_CAR <car> <succ> <fail> <remap> <score>
Simple command that checks that the player is currently driving the car defined in line <car>. If he is, it goes to <succ> adding <score>, otherwise it goes to <fail>. In addition, if <remap> is greater than -1, it will check that the car's remap value equals <remap>.

WAIT_FOR_PLAYER <location> <succ> 0 <dist> <score>
Waits for the player to leave the immediate area around the block declared in line <location> before proceeding to <succ>.. <dist> is the number of blocks in each direction it checks.

EXPL_PED <pednum> <succ> 0 0 <score>
Explodes the ped defined in line <pednum>. Simple as that!

PARKED_ON   <futurecar> <succ> <fail> 0 <score>
Creates the FUTURECAR futurecar, but without a driver.

CANCEL_BRIEFING <briefing> <succ> <fail> 0 <score>
Cancels the 'timed briefing' which is currently going on. <briefing> should be the line number of the briefing you want cancelled.

This command is here for completion's sake.

FREEZE_TIMED <time> <succ> 0 0 <score>
Freezes the player's controls as discussed for <time> frames.

FREEZE_ENTER 0 <succ> 0 0 <score>
Freezes the 'enter/leave car' key, e.g. preventing the player leaving the car.

UNFREEZE_ENTER 0 <succ> 0 0 <score>
Unfreezes the 'enter/leave car' key. All three of these commands go to <succ> adding <score> right away.

EXPL_NO_FIRE <target> <succ> 0 <face> <score>
Causes the building at <target> to explode on its <face> side, without producing any rubble, plantpots or fire. It always goes to <succ>

KICKSTART <line> <succ> -1 -1 -1

This command is pretty powerful, it allows you to 'kickstart' a new mission process, starting at line <line> with the current player, in tandem with the current one, i.e. you can do two things at once….

Please EXPLICITLY declare <succ>, i.e. don't just put 0, put the proper linenum in.

Use this when you want to do one thing and check for a destroy at the same time, but use it WISELY.

NEXT_KICK <line> <succ> -1 <kickstart> 0

This does a 'second' kickstart in the same block of code, e.g. when you want a destroy check on a car & a ped at the sametime. <line> is the line to start the new process from, <kickstart> is the linenum of the first KICKSTART in this 'section'.

KILL_PROCESS <kickstart> <succ> 0 0 <score>

This is the compliment of the kickstart command, allowing you to stop processes. <kickstart> is the line num of the KICKSTART command, and this will kill ANY process started using KICKSTART & NEXT_KICK in the same 'section' of code.

This command MUST be executed at some point when you've done a load of kickstarts & next_kicks otherwise things could get messy….

KILL_SIDE_PROC <kickstart> <succ> 0 0 <score>

Kills all 'kickstarted' process started on the line <kickstart>, but keeps the 'original' thread running.

SET_KILLTRIG <trigger> <succ> 0 <kickstart> <number>

This does a special 'killer trigger' that will kill a given process when the trigger <trigger> is switched on by its normal conditions. <kickstart> should be the line the kickstart that the process was 'stored' in, and <number> should be the number of the process, e.g. if it's the kickstart it's 1, the first NEXT_KICKSTART is 2, etc.

KILL_SPEC_PROC <kickstart> <succ> -1 <proc> <score>

Kills a 'thread' started by either a kickstart or a NEXT_KICK. Point it to the original kickstart in line <kickstart>. <proc> should be one of 1,2 or 3, depending on the process. The process 'kickstarted' is 1, the process from the first NEXT_KICK is 2, the second NEXT_KICK is 3 etc. Doing a 0 will kill the 'orignal process' that executed the KICKSTART.

This will then proceeed to <succ> adding <score>, unless you've killed this process.

POWERUP_OFF <powerup def> <succ> 0 0 <score>

Deletes the powerup defined in line <powerup def>, going to <succ> etc.

SCORE_CHECK <score> <succ> <fail> 0 <score>

Checks to see if the current player's score is greater than or equal to <score>, going to <succ> if it is adding <score>.
Goes to <fail> if not.

FRENZY_SET <line> <succ> -1 -1 <score>

Sets the score for this player in the correct place in line <line>. This should be a FRENZY_CHECK or else strange things will happen….

FRENZY_CHECK <score diff> <succ> <fail> 0 <score>

This does the actual 'frenzy check' for the player. It takes <score diff>, the score the player should have achieved for this mission and compares it to the player's current score. If okay, goes to <succ> else to <fail>.

GET_CAR_INFO <ped> <succ> 0 <dummy> <score>

This is used to 'get' the info for a car. It takes the ped defined in line <ped> and sets the relevant info in line <dummy> which should have been defined as a DUMMY.

LOCATE_STOPPED <object> <succ> <fail> <timer> <score>

Waits for the player to reach a certain location on foot, or for the timer to run out, if the player reaches the location it goes to the <succ> objective otherwise it goes to the <fail> objective. A timer of 0 will disable the timer. Note: **PLAYER MUST BE ON FOOT & STOPPED** for this to succeed!

MOBILE_BRIEF 0 <succ> 0 0 <brief>

This does a new 'mobile phone' style brief. Similar to pager I think, but puts a mobile phone icon on-screen. Talk to Keith about this & the speech one, nothing to do with me. Both of these new commands 'pause' the game.

SPEECH_BRIEF 0 <succ> 0 0 <brief>

This does a 'speech' brief - similar to old style subtitles, with a speech icon.

MESSAGE_BRIEF 0 <succ> 0 0 <brief>

Does a simple 'message' style briefing at the very top of the screen. Behaves like every other briefing, you know the score..

START_CHOPPER <start loc> <succ> <fail> <player> 0

This will start the 'automatic' end of level pickup. <player> is the line num of the player, <start loc> is the block where the helicopter should be created at. If it fails to create a chopper, it will go to <fail>, otherwise <succ>.

Notes:
For the end of level to look smooth, create the chopper slightly offscreen & at a high z-level. Make sure the player has been 'freezed' to prevent him moving off. The chopper will fly over and drop down, and then zoom out & fly off to the point defined as the end point - where the game will end.

PIXEL_CAR_ON <car> <succ> <fail> <remap> <score>

Creates a car, similar to CAR_ON, but with coords defined in PIXELS, like PARKED_PIXELS. <car> should ideally be a TARGET, with the data similar to a FUTURE_CAR…

```
e.g. 293 (7840,8736,256) TARGET 29 768
       20 PIXEL_CAR_ON 293 100 0 0 0
```

GENERAL_ONSCREEN <goal> <succ> <fail> 0 <score>

Checks to see if the goal <goal> is currently 'on' the current player's screen (ie the player 'doing' this mission.). If it is, goes to <succ> etc etc.

PED_WEAPON <ped> <succ> 0 <weapon> <score>

Gives the dummy ped in line <ped> a weapon of type <weapon>, going to <succ> etc. If the ped has yet to be created, or anything else goes wrong, it will give a fatal error message.

SET_NO_COLLIDE <car> <succ> 0 0 <score>

This will set the car declared in line <car> as a 'non-collidable'. So if bumped, it'll stay stationary & the other car will bounce.

SET_COLLIDE <car> <succ> -1 -1 <score>

Equivalent command to clear the 'non-collide' flag set by SET_NO_COLLIDE

IS_GOAL_DEAD <goal> <succ> <fail> <killer> <score>

Checks the status of goal <goal>. If it is technically 'dead' - i.e. if a car, blown up, if a ped, dead, it goes to <succ>, otherwise proceeds rightaway to <fail>.

New: if <killer> is greater than 0 and <goal> is a ped/player, it will check to see if ped <killer> killed the goal ped. So you'll need a ped health of 0 **AND** this other ped killing it for this to succeed if <killer> is used… That make sense?

IS_PED_IN_CAR <ped> <succ> <fail> <model> <score>
Checks to see if the ped in line <ped> is currently inside a car/on a bike. If <model> is >-1, the ped must be in a car with this model number. Set it to 0 for 'basic' use.
If so, goes to <succ>, otherwise to <fail>.

BANK_ALARM_ON 0 <succ> -1 <location> <score>
This switches on the 'bank alarm' at the coords defined in <location>. This should probably be a dummy as data needs to be stored.

BANK_ALARM_OFF <location> <succ> 0 0 <score>
Switches off the sound effect of a 'bank alarm' at the location declared in line <location>. This should be the line used as <location> when you started the bank alarm.

FREEUP_CAR <car> <succ> -1 -1 <score>
This will 'flag' the car defined in line <car> as no longer need as a mission car.

PARKED_PIXELS_ON <car> <succ> <fail> 0 <score>
Creates the a FUTURECAR <car> in 'pixels', with no driver.
Use this for precise placement of cars.

PED_POLICE <ped> <succ> <fail> -1 <score>
Changes the previously created ped in line <ped> to look like a policeped. Goes to <succ>, blah, blah, blah.

DROP_WANTED_LEVEL 0 <succ> 0 0 <score>
Clears the wanted level for the current player, adding score & going to <succ>

LOCK_DOOR <car> <succ> -1 -1 <score>
Locks the door of a previously created car, stopping ped from entering (and maybe leaving..)

UNLOCK_DOOR <car> <succ> -1 -1 <score>
Unlocks the door of a car, allowing the player to get back out.

IS_PED_ARRESTED <ped> <succ> <fail> -1 <score>
Does a simple check to see if this ped is 'currently' arrested. Goes to <succ> if so, otherwise straight to <fail>

This command isn't foolproof - it really needs to be looped round to 'catch' the arrest 100%

HELL_ON <goal> <succ> <fail> 0 <score>
Creates a Hells Angel which has been declared as a FUTURE_CAR. Remaps the ped etc correctly for a hell's angel ped…

SET_PED_SPEED <ped> <succ> 0 <new speed> <score>
Sets the 'max speed' for a given ped. Means he won't run faster than this speed. The standard ped speed is 4 for guidance. Goes to <succ> etc.
Useful for FOLLOWs where you don't want the player to catch up…

IS_PLAYER_ON_TRAIN <train> <succ> <fail> 0 <score>
If the current player is curerntly in a train, this will go to <succ>. <train> should be the line number of a DUMMY line, this is for storing the train info if you want to blow it up later on.. If the player isn't on a train, goes to <fail> etc.

WRECK_A_TRAIN <train> <succ> -1 -1 <score>
Wrecks a train! <train> should be the <train> line set before in a IS_PLAYER_ON_TRAIN. Goes to <succ> etc.
NOTE - this **PERMANENTLY** wrecks the train system!

INCCOUNT <counter> <succ> <fail> <check> <score>
Increments the given counter in line <counter> by 1. After it does this, it checks to see if the counter is great than or equal to the <check> number… Goes to <succ> if it is, <fail> otherwise. Set check to <-1> if you don't want any sort of check.

COMPARE <counter> <succ> <fail> <check> <score>
Takes the counter <counter> and compares it to the number <check>. If equale, goes to <succ>, otherwise it goes to <fail>.

RESET 0 <succ> -1 -1 <score>
Forces a reset - this will clear off every process that's running (except for this one, and any KEEP_THIS_PROC ones of course), and any objects, peds & cars that have been created without the '1' flag.

KEEP_THIS_PROC 0 <succ> -1 -1 <score>
Use this to 'keep' this process from being stopped when a new mission starts.

IS_PED_STUNNED <ped> <succ> <fail> -1 <score>
If the ped definied in line <ped> is currently 'stunned', it will go to <succ> otherwise to <fail>
A ped is 'stunned' when he is punched.

IS_A_TRAIN_WRECKED -1 <succ> <fail> -1 <score>
Checks to see if there's currently a train wrecked anywhere on the map. Goes to <succ> if there is <fail> if there isn't. You can use this to 'protect' any missions using the trains - check to see if they can complete the mission before you start it for instance.

UNSUPPORTED

INC_HEADS <num heads> <succ> -1 -1 <score>
Simply increases the number of heads by <num heads>, to a maximum of four.

KF_BRIEF_TIMED 0 <succ> -1 <time> <type>
Displays a "KILL FRENZY" briefing of type <type> that finishes with a countdown of <time> secs roughly. Proceeds straightaway to <succ>. This 'switches on' tanks.

KF_CANCEL_BRIEFING <briefing> <succ> <fail> 0 <score>
Cancels the 'timed briefing' which is currently going on. <briefing> should be the line number of the brief you want cancelled. Switches off tank firing.

KF_BRIEF_GENERAL 0 <succ> 0 <time> <type>
Displays a "KILL FRENZY" briefing of type <type> that finishes with a countdown like KF_BRIEF_TIMED.

KF_CANCEL_GENERAL <briefing> <succ> <fail> 0 <score>
Cancles the 'timed briefing' which is currently going on. <briefing> should be the line number of the briefing you want cancelled. Doesn't do anything to tanks.

STOP_FRENZY <weapon> <succ> -1 -1 <score>
This will prematurely stop a kill frenzy started on the weapon declared in line <weapon>.

FRENZY_BRIEF 0 <succ> 0 0 <brief>
This does a standard brief but with a "KILL FRENZY" icon - plus no 'chatter' sound effect.

ADD_A_LIFE 0 <succ> -1 -1 <score>

Simply adds a life to the current player's total, and proceeds to <succ> etc.

DEAD_ARRESTED <ped> <succ> <fail> -1 <score>

Simple command that goes to <succ> if the ped declared in line <ped> is either dead or arrested at that particular moment. Goes to <fail> if not.
Handy when 'looped round'.

IS_POWERUP_DONE <powerup> <succ> <fail> -1 <score>

If the powerup declared in line <powerup> has been collected, it will go line <succ> otherwise, goes to <fail>

START_MODEL <dummy> <succ> <fail> <car to kill> <num of cars>

All singing, all dancing reverse model cars command set. New structure -
<dummy> is a DUMMY line used to 'create' the model cars in, and store some more info in. It should have the (XYZ) to create the cars at, usually the same as the truck the player should be in. Not really important now - gets the coords of the truck at runtime…
<succ> if it successfully creates the car, goes to here
<fail> else to here if ANYTHING goes wrong!!!
<car to kill> Simply the linenumber of a target to kill. Means the DO_MODEL will stop when you destroy it. Set to -1 to ignore it
<num of cars> Number of model cars you get to play with!

DO_MODEL <startmodel> <succ> <fail> <car to kill> 0

This command does the actual processing of the Dead Pool stuff. <startmodel should be the line number of the 'START_MODEL' command. Whenever a model car blows up, this will check whether or not to create a new one. When the player has gone through all his/her cars, it goes to <succ>. <car to kill> is as above - set to -1 to ignore it.
<succ> should go *directly* to a RETURN_CONTROL.

RETURN_CONTROL <start line> <succ> 0 0 <score>

Simple command to wrap up the Dead Pool, it returns control to the ped the player originally controlled, adding <score> and going on to <succ>.
[See Notes]

RESET_WITH_BRIEFS 0 <succ> -1 0 <score>

Does a 'standard' reset but with the added bonus of stopping any briefings that are on-screen except for any 'big font' messages.

RED_ARROW <loc> <succ> -1 -1 <score>

Points a **RED** arrow at a given goal. This arrow will NOT be reset/removed unless a "RED_ARROW_OFF" is called. It does NOT interfere with the standard yellow arrow. It points to the (XYZ) of <loc> and will NOT update if this goal moves.

RED_ARROW_OFF 0 <succ> -1 -1 <score>

Cancels the red arrow. Duh.

PIXEL_CAR_ON <car> <succ> <fail> <remap> <score>

Creates a car, similar in behavour to CAR_ON, but with coords defined in PIXELS for extra careful placing. <Car> should ideally be a TARGET with the data similar to a FUTURE_CAR eg

```
293 (7840,8736,256) TARGET 29 768
20 PIXEL_CAR_ON 293 100 0 3 0
```

Notes on Triggers & Ordering

Any car-triggers must be declared **after** the car that they're referring to. This is very important! Same with drivers for cars, make sure the car is declared first.

This is the same for **ANYTHING** that refers to another line - this 'other' car/ped/object must have been declared before the reference to it.

Notes on Timed Objectives with zero timer.

Every objective that has a timer with the 0 - Infinite option, can now be failed, at the moment only the escort arriving at their destination will cause this to happen, but if you can think of anything else that could be used then please let me know.

Line Numbering

After the effort James went to, to insert proper line numbering, you could at least have the decency to use it in a bit more responsible manner, you have 32768 (0-32767) line numbers to play with, you don't have to squeeze them all together, plus at the moment you're not leaving any room if anything needs to be inserted into them later you'll be completely stuck, just don't say I didn't warn your, and before you ask, I am not going to write a renumber program for you, you'll have to do that yourselves.

Line Number Zero

The target defined with line number zero will be executed **every** time. Use this **very** carefully!

Z-Values

'Physical' Z-Value means it's like the *actual* z-value, e.g. 4
'Logical' Z-Value means it's the game processing value, e.g. 3

Basically, if you're creating something use the physical value, if you're referring to something use the logical value… That's the gist of it.

Startup & Multiphone triggering

Multiphone 'retriggering' now works, if you follow the proper layout for it. It's a bit fiddly but it works. You need to do a MPHONE declaration, followed by a PHONE_TOGG to get the multi-phones to work….
If you want to execute a section of code once on startup, you need to place a trigger on the start location of the player, which calls a line with the STARTUP command in it. This section of code doesn't have to consist of DISABLEs only, you can use it to do an arrow pointing to the multiphones, or a briefing. In fact, it's better to do this with STARTUP.

Here's a little example to show you what I mean.

```
(123,50,4)     MPHONES       10     3              {line 200}
.
.
(123,50,4)     PHONE_TOGG    200    5              {line 203}
.
(120,50,4)     TRIGGER       3000   0              {line 210}

10      {multiphones as normal}
.
.
.
3000    STARTUP       210    0      -1     0      0
3010    DISABLE       {disable triggers etc goes here}
```

.
.
3020	DISABLE	10	-1	0	0	0	{the -1 'kills' this process, **THIS IS IMPORTANT….**}

Doors

Doors now have a default state of 'OFF' to prevent players entering them at the wrong time. Use **DOOR_ON** to turn them 'ON' during particular missions, and use **DOOR_OFF** to switch them back off when you're finished. **DOORS NOW DEFAULT TO OFF**

Typical Errors

If you're having problems getting mission.ini to load, things to check/remember are:

- "Request for an object line number has produced a line_num of -1"

  These tend to mean that a line somewhere above has too little or too many parameters, particular culprits are doors & peds. Look out for driver peds without a car for example.

  Typing errors are also prone to give this - look for brackets with { and (mixed up, or maybe . instead of , … All these can give this error. These can be fiddly to find, but if I can do it, so can you.

  Of course, it could be that you're just referring to an object which doesn't exist.

- "missing '(' in style … "

  This means dino is trying to find a (X,Y,Z) when it shouldn't be. Commonest cause of this is missing out the '-1' at the end of the object list. Something like

  ```
  101 (0,0,0) DUMMY 0 0
  102
  103 (0,0,0) DUMMY 0 0
  ```

  can cause this error too….
- IS_PED_IN_CAR not going to succ

  Check that your model number is -1… If it's 0, it will look for a model num of 0!

GTA PED REMAPS

(These may not be 100% correct)

0	not used
1	darken 50%
2	lighten 50%
3	red -> yellow
4	red -> green
5	red ->light blue
6	lighten 40%
7	darken 40%
8	lighten 30%
9	darken 30%
10	darken 10%
11	darken 20%
12	lighten 10%
13	darken 10%
14	red -> grey
15	ALL -> red
16	ALL -> dark blue
17	ALL -> light green
18	ALL -> light yellow

19	ALL -> bright green
20	ALL -> dark blue (not light brown!)
30	ALL -> yellow
31	ALL -> bright green
32	ALL -> light blue
33	ALL -> grey
34	ALL -> red (not l.blue)
35	ALL -> light blue (not red)
36	ALL -> yellow (not l.green)
37	ALL -> light green (not yellow)
38	ALL -> light brown
128	darken 8%
129	darken 16%
130	darken 26%

A Guide to Ped Types in GTA

NOTE:

- "EXTRA INFO" is the piece of extra info called <enemy> in the command listing above.
  For PED_ONS this should be in the <score> field…
- An 'extra info' of -1 for simple guards means do NOT shoot at any ped/player
- -2 means the ped will latch onto a random ped…

Standing Ped 0
Simple standing ped, that will jostle slightly. Doesn't require any extra info

Basic Guard 1
Basic guard, waits for you to shoot near him. Will not chase.

Driver 2
A driver of a car, he will attempt to get into 'his' car & drive off.
EXTRA INFO: line num of his car.

Walking Ped 4
Simply walks around…

Follow ped 5
Simply follows another ped around
EXTRA INFO: linenum of ped to follow around

Running Ped 6
Simply runs around…

Escaping Ped 7
This ped will try to flee a given ped.
EXTRA INFO: linenum of ped to flee from

"Follow Guard" 8
This ped behaves as a normal stationary guard ped until someone shoots at him, when he starts giving chase, shooting as necessary.
EXTRA INFO: line num of the ped 'expected' to chase - eg the player.

Mayor 9

The Mayor/VIP/President as necessary. Doesn't really do much except run when he can, but is worth a LOT when killed.

Lemming ped 10
Is very thick. So thick in fact that he'll commit suicide by walking off the edges of buildings etc.

"Killer Bloke" 11
A friend to the needy. He'll start shooting peds acting as a kind of bodyguard to you.
EXTRA INFO: linenum of ped to kill.

"Mr Shooty Cars" 12
This guy will shoot at a given car till kingdom comes.
EXTRA INFO: linenum of car to shoot.

"Mr Shooty Follow" 13
This guy will shoot at a given car till kingdom comes like above, but this guy CHASES the car too!
EXTRA INFO: linenum of car to shoot.

(complicated peds now…)

Standing Guard, Crap Shot 21
Stationary guard who is a fairly crap shot

Standing Guard, Good Shot 22
As above, but a pretty good shot.

Standing Guard, Crack Shot 23
As above, but rarely misses

Follow Guard, Crap Shot 24
A "follow guard" like above, but is a fairly crap shot

Follow Guard, Good Shot 25
As above, but a pretty good shot.

Follow Guard, Crack Shot 26
As above, but rarely misses.

Standing Killer Bob, Crap Shot 41
Stationary 'killer bloke' - will shoot at a given ped. Crap shot though.

Standing Killer Bob, Good Shot 42
Stationary 'killer bloke' - will shoot at a given ped. Good shot though.

Standing Killer Bob, Crack Shot 43
Stationary 'killer bloke' - will shoot at a given ped. Rarely misses

Chase Killer Bob, Crap Shot 44
This 'killer bloke' will chase after his target ped, shooting at him. Crap shot though.

Chase Killer Bob, Good Shot 45
As above, but a better shot.

Chase Killer Bob, Crack Shot 46
As above, but rarely misses his target.

By using these more complicated types, you get more control of the actions of these peds.

Object types in GTA

A quick list of suitable 'object types' for the likes of suitcases, telephones etc.

o_umberella 0
o_tyre 1
o_lid 2
o_bin 3
o_cone 4
o_flowers 5
o_barriery 6
o_barrier2 7
o_bench 8
o_gravestone 9
o_smoke 10
o_skid 11
o_bloodskid 12
o_spark 13
o_train_doors 14
o_blood 15
o_tlrbox 16
o_dirtskid 17
o_fire 18
o_fire2 19
o_interior2 21
o_bomb 22
o_barrier 24 //used to be 24, changed due to barriercades creating tombstones!!
o_bumper 25
o_tree1 26
o_tree2 27
o_tree3 28
o_crane 30
o_missile 31
o_turret 33
o_tgun 34
o_zturret 38
o_zgun 39
o_phone 40
o_case 41
o_rubble 43
o_rubble2 44
o_rubble1 45
o_lfire 46
o_dropoff 47
o_fw1 48
o_fw2 49
o_firehose 50
o_big_smoke 51
o_stretch 52
o_exhaust 53
o_splash 54
o_arm1 55
o_arm2 56
o_arm3 57

```
o_swatgun 59
o_checkflag 60
o_first_blood 63
o_body 64
o_inv 65
o_inv2 66
o_smk 67
o_inv3 68
o_barrel2 69
o_camera 70
o_bullet 74
o_fireball 75
o_blood_smear 77

// powerup objects
o_powerup_pistol      78
o_powerup_machinegun  79
o_powerup_rocket      80
o_powerup_flamethrower 81

// crate object that sits on top of powerups
o_crate               84
o_crate_exploding     85

o_beachball           86
o_bollard             87
o_flag_waving         88
o_rubbish             89

// rotor blade
o_blade               92

o_rubbish_exploding         93
o_helipad                   94

o_powerup_info              95
o_powerup_shield            96
o_powerup_light       97
o_powerup_key         98
o_powerup_heart       99
o_powerup_extra       100
o_powerup_coin        101
```